Universidad Nacional
SAN LUIS GONZAGA

UNIVERSIDAD NACIONAL "SAN LUIS GONZAGA"

FACULTAD DE INGENIERÍA DE SISTEMAS

DIRECCIÓN DE INVESTIGACIÓN

EVALUACIÓN DE ORIGINALIDAD

CONSTANCIA

El que suscribe, deja constancia que se ha realizado el análisis con el software de verificación de similitud al documento cuyo título es:

AUTOMATIZACION DEL PROCESO DE PRESENTACIÓN DE EXPEDIENTES Y SOLICITUDES DE PROYECTOS Y OBRAS EN EMPRESA DE DISTRIBUCIÓN ELÉCTRICA

Presentado por:

- **VALENZUELA RAMOS YORGHINIO ALDAIR**

BACHILLER en **PREGRADO** de la facultad de Ingeniería de Sistemas. El resultado obtenido es **(porcentaje de similitud 1%)** por el cual se otorga el calificativo de:

APROBADO, según el Reglamento de Evaluación de la Originalidad.

Se adjunta al presente el reporte de evaluación con el software de verificación de originalidad.

Ica, 10 de abril del 2024

Dr. JAVIER ORLANDO GUTIÉRREZ FERREYRA
Director de la Unidad de Investigación
Facultad de Ingeniería de Sistemas

UNIVERSIDAD NACIONAL “SAN LUIS GONZAGA”

VICERRECTORADO DE INVESTIGACIÓN

FACULTAD Ingeniería de Sistemas

Línea de investigación: Ciencias naturales, ingeniería y tecnologías sostenibles

AUTOMATIZACION DEL PROCESO DE PRESENTACIÓN DE EXPEDIENTES Y SOLICITUDES DE PROYECTOS Y OBRAS EN EMPRESA DE DISTRIBUCIÓN ELÉCTRICA

Autor: BACHILLER YORGHINIO ALDAIR VALENZUELA RAMOS

Asesor: DR. EDGAR LEONARDO PEÑA CASAS

INFORME FINAL DE TRABAJO DE SUFICIENCIA PROFESIONAL

Ica, Perú

2024

Dedicatoria

El presente informe es dedicado para todas las personas que me han acompañado en todo este trayecto, en especial a las dos personas que iniciaron junto a mi este camino y que hoy no podrán estar físicamente para ver el culmen del mismo (mi abuela Zoila Rosa Mayurí Vda. de Ramos y mi madre Blanca Luisa Ramos Mayurí de Valenzuela), a mi padre, hermanos y a la familia Tito Altamirano por formarme, enseñarme, por estar siempre dispuestos a escucharme y aconsejarme en cada momento; a mi compañera de vida, a mi amada esposa Natalia Shirley Tito Altamirano que siempre me da su apoyo incondicional para que pueda superarme y ser mejor persona cada día y al nuevo ser que está creciendo en su seno por seguir complementando esta felicidad.

También es dedicado para aquellas personas que no creyeron que haciendo lo necesario, luego lo que es posible estarían haciendo lo que creían imposible (San Francisco de Asís).

Agradecimientos

Agradecer a Dios por el don de la vida, la salud y la familia, también agradecer a la Facultad de Ingeniería de Sistemas y a su plana docente y administrativa por ser parte fundamental en el aprendizaje profesional y personal.

Asimismo, agradezco a mis padres, hermanos, a la familia Tito Altamirano y mi esposa Natalia Tito quienes me guiaron y siempre estuvieron dispuestos a ayudarme.

También quiero agradecer a cada empresa en la que he laborado por confiar en mí y en especial a la empresa Electro Dunas S.A.A. por brindarme esta oportunidad de seguir compartiendo y ampliando mis conocimientos.

Índice

INDICE DE TABLAS

INDICE DE FIGURAS

RESUMEN

El presente trabajo de suficiencia profesional detalla la problemática que empezó a tener Electro Dunas S. A.A. frente a la pandemia de COVID – 19, esta problemática es que los clientes no podían presentar sus solicitudes de proyectos y obras de manera presencial ya que la documentación se tenía que entregar era de manera física, este proceso demandaba un gran desgaste de tiempo así como el desgaste de hojas, es por ello que se implementó una herramienta de centralización de información y de ayuda al cliente en que pueda presentar su documentación de manera virtual, logrando la automatización de este proceso. Para poder lograr este objetivo se utilizó la metodología ágil SCRUM para poder gestionar los recursos y tiempos, gestor de base de datos PL/SQL Oracle vinculado a la herramienta .Net, los resultados que se dieron luego de la implementación del sistema fueron los satisfactorios ya que se reduzco los tiempos de atención y la disposición de información a evaluar.

Palabras clave: SCRUM, PL/SQL Oracle, automatización

ABSTRACT

The present work of professional sufficiency details the problems that Electro Dunas S. A.A. began to have. Faced with the COVID - 19 pandemic, this problem is that clients could not submit their requests for projects and works in person since the documentation had to be delivered physically, this process demanded a great deal of time as well as the wear and tear of sheets, which is why a tool for centralizing information and customer support was implemented in which they can present their documentation virtually, achieving the automation of this process. In order to achieve this objective, the agile SCRUM methodology was used to manage resources and times, a PL/SQL Oracle database manager linked to the .Net tool, the results that occurred after the implementation of the system were already satisfactory. that the attention times and the disposition of information to be evaluated are reduced.

Keywords: SCRUM, PL/SQL Oracle, automation

INTRODUCCION

La empresa Electro Dunas S.A.A. es una de las principales empresas de distribución de energía eléctrica en la región de Ica, es por ello que diferentes empresas presentan ante Electro Dunas los expedientes y solicitudes de los proyectos y obras a realizar dentro de su concesión, como parte del mejoramiento de atención al cliente y el seguimiento de estas solicitudes se solicitó la participación de la Gerencia de Tecnología Informática (TI).

Antecedentes: La presentación de los expedientes y solicitudes de los proyectos y obra se realizaban de manera presencial, por motivos de la pandemia COVID – 19 esta actividad ya no se podía realizar por las medidas de salubridad dadas por el gobierno peruano, al tener esta restricción involucró que varias empresas no puedan continuar con sus proyectos u obras tales como: habilitación urbana de una nueva residencial, construcción de una planta de oxígeno en algunas provincias de la región Ica, etc.

Asimismo, al tener la documentación de manera física involucraba un tiempo de búsqueda de esta información para poder seguir con la atención de una solicitud, aumentando el tiempo de atención, así como la sobre carga del trabajo en algunos trabajadores de Electro Dunas.

Estos inconvenientes se presentaron desde la Gerencia Comercial y la Gerencia Técnica es por ello que se vio la necesidad de involucrar a la Gerencia de Tecnología Informática para la implementación de un portal web que ayude en la presentación de estos expedientes y solicitudes, así como el trámite de atención correspondiente.

El objetivo: Proporcionar una herramienta de ayuda al cliente para que pueda presentar los requisitos de los expedientes y solicitudes del proyecto u obra a realizar, asimismo esta herramienta ayudará en el seguimiento y la atención de estas solicitudes, adicionalmente le permitirá reconocer a Electro Dunas aquellas empresas que no tengan un personal especializado que elabore este proyecto, siendo una ventana comercial para Electro Dunas.

Alcance: la implementación de este portal web tiene como alcance la gestión de los requisitos que el cliente presentará para su determinado proyecto u obra, el seguimiento de las solicitudes del cliente, la configuración de las solicitudes a presentar de un determinado tipo de proyecto, mejorar la comunicación para la atención de la solicitud, para poder realizar la correcta implementación de este portal web están involucradas de manera adicional la Gerencia Comercial, Gerencia Técnica.

La implementación de este portal se justifica en el cumplimiento de los objetivos estratégicos de la Gerencia de Tecnología Informática y en la transformación digital de Electro Dunas.

Capítulo I: INFORMACION DE LA INSTITUCION DONDE SE DESARROLLO LA EXPERIENCIA.

Electro Dunas S.A.A. es una empresa privada perteneciente al Grupo Energía Bogotá (GEB) cuyo objetivo es dedicarse a la prestación de servicio de distribución y comercialización de energía eléctrica, con carácter de servicio público o libre contratación, dentro de su área de concesión. Opera en la región sur medio del Perú, específicamente en el departamento de Ica y parte de los departamentos de Huancavelica y Ayacucho.

La empresa inició sus operaciones el 30 de enero de 1912, bajo la denominación de Sociedad Anónima de Electricidad Limitada. Después de sucesivos cambios de denominación, el 04 de abril de 1987 pasó a llamarse Empresa Regional de Servicio Público de Electricidad del Sur Medio S.A.A. – Electro Sur Medio, y a partir del 15 de noviembre de 2009 tomó el nombre de Electro Dunas S.A.A.

Electro Dunas S.A.A. tiene como misión "comprar, transmitir, distribuir y comercializar energía eléctrica con la calidad y en la oportunidad requerida, maximizando la rentabilidad de los accionistas, con responsabilidad social, protegiendo el medio ambiente y contribuyendo al progreso y desarrollo del país", además tiene como meta "ser líderes en el mercado de distribución eléctrica del Perú, logrando óptimos niveles de excelencia y calidad en nuestro servicio. Estar comprometidos con nuestros clientes, ser fuente de orgullo para nuestros colaboradores y ser rentables para los accionistas, promoviendo a su vez el desarrollo sostenible y eficiente en nuestro ámbito de influencia" y para poder lograrlo cuenta con los siguientes valores institucionales:

- Primero la vida
- Integridad
- Trabajo en equipo con responsabilidad individual
- Enfoque a resultados
- Empatía

Adicional a los valores institucionales, la empresa en el Código de Ética y Conducta en la página 06, invita tanto a administradores como colaboradores a actuar bajo los siguientes principios éticos:

- Transparencia
- Respeto
- Equidad
- Legalidad
- Responsabilidad

En la memoria anual de Electro Dunas del año 2022 desde la página 15 hasta la página 17 se detallan los miembros del directorio, siendo:

- Juan Ricardo Ortega López
- Andrés Baracaldo Sarmiento
- Jorge Andres Tabares Angel
- Lucía Cayetana Aljovín Gazzani
- Christian Thomas Laub Benavides

Asimismo, desde la página 18 hasta la página 21 de la memoria anual de Electro Dunas del año 2022 se detallan los miembros de la gerencia de la compañía.

Tabla 1.Miembros de la gerencia de Electro Dunas

Nombre	Cargo
Walter Nestor Sciutto Brattoli	Gerente General
Roxana Palomino Briones	Gerente de Administración y Finanzas
David Chala	Gerente de Tarifas y Compra de Energía
Ernesto San Miguel	Gerente Técnico
Gustavo Javier Michelena	Gerente Comercial
Jorge Santivañez Seminario	Gerente Legal
Angela María Acevedo	Gerente de Gestión del Talento
Eduardo Miranda	Gerente de Tecnología Informática

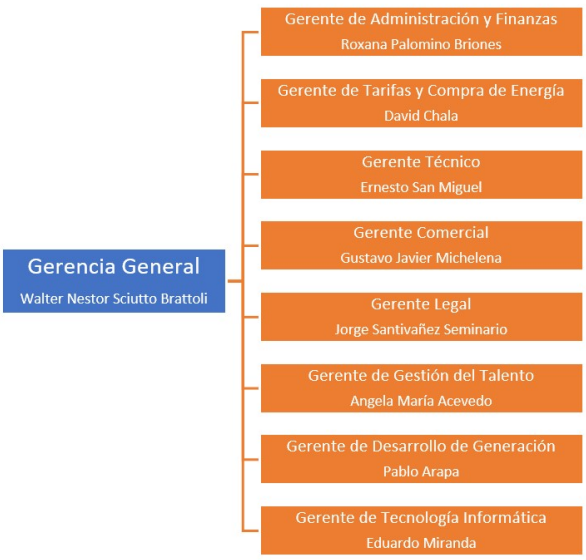


Figura 1. Organigrama de Gerencias en Electro Dunas

Como se aprecia en la figura 1 Electro Dunas cuenta con la Gerencia de Tecnología Informática (TI), la cual tiene como uno de sus principales objetivos estratégicos la transformación digital de Electro Dunas, esta transformación digital va de la mano de las diferentes necesidades y automatización de procesos que requieren las áreas que conforman las demás gerencias y necesidades propias de la gerencia de TI, para ello es donde la gerencia de TI releva toda la información necesaria, aprueba y prioriza el desarrollo de un nuevo proyecto en base a un plan de trabajo, este es el caso de la automatización del proceso de presentación de expedientes y solicitudes de proyectos y obras, este proceso antes del proyecto era de manera física y manual ocasionando una ineficiencia en la identificación de información así como la creación de identificadores redundantes para la empresa.

Capítulo II: TRAYECTORIA PROFESIONAL

El autor del presente trabajo inicia a laborar la consultora **SOEN S.A.C.** (Soluciones Estratégicas de Negocio) como **Analista Programador** desde enero del año 2019 hasta enero del año 2020, donde participó de diferentes proyectos para poder mejorar los procesos de atenciones a los clientes de la consultoría, tuvo participación en:

- Desarrollo e Implementación del Sistema Mesa de Partes para la empresa RANSA del grupo Romero.
- Desarrollo e Implementación del Sistema de Gestión Documental DocuBlock para la empresa RANSA del grupo Romero.
- Desarrollo e Implementación del Sistema de Gestión para la comunidad católica La Barca.

Las actividades que ha realizado y están desarrolladas con la especialidad son el análisis de procesos, desarrollo, pruebas unitarias, pruebas de usuario, desplegar y documentación de las aplicaciones desarrolladas en .NET, usando como gestor de base de datos SQL Server, las aplicaciones han sido desarrolladas usando la estructura MVC, N Capas, Web Api y Micro Servicios en ASP .Net en el lenguaje de programación C#.

También ha desarrollado aplicaciones web con React para las interfaces.

La metodología ágil usada para llevar la gestión recursos, tiempos de desarrollo e hitos de entrega de los proyectos fue SCRUM.

Luego el autor inicia labores en **CONSATEL S.A.** como **Analista de Sistemas** desde febrero del año 2020 hasta setiembre del año 2020, donde participó del proyecto Desarrollo e Implementación del Sistema de Gestión de Colas – ICLASSQ, las actividades que ha desarrollado y están vinculadas a la especialidad han sido el análisis de procesos, desarrollo, pruebas unitarias, pruebas de usuario, implementación y documentación de la aplicación desarrollada en el lenguaje de programación Java, usando como base de datos MySql, la arquitectura usada es MVC y la metodología ágil que se usó para gestionar los recursos, tiempos de desarrollo e hitos de entrega fue SCRUM.

Posteriormente el autor inició labores en la empresa **Electro Dunas S.A.A** desde octubre 2020 hasta la actualidad como **Analista de Sistemas,** en la empresa donde labora actualmente ha participado en diferentes proyectos que han surgido a la necesidad de Electro Dunas, en la mejora de automatización de procesos y mejora continua tanto en la gestión interna de la empresa como para la atención al cliente, estos proyectos se detallan a continuación:

- Automatización del proceso de presentación de expedientes y solicitudes de Proyectos y Obras de Terceros (proyecto seleccionado para la titulación por la modalidad de suficiencia profesional)
- Implementación de la Afiliación al Recibo Digital para los clientes.
- Automatización de la presentación de solicitud de nuevo suministro.
- Implementación del Portal para reportar deficiencias técnicas al servicio que ofrece la empresa.
- Implementación del Portal para reportar emergencias.
- Implementación del portal de Mesa de Partes.
- Automatización del proceso de envío de comprobantes electrónicos para validación ante la OSE y SUNAT.
- Ajuste en el portal de consulta de las fotos de lectura del cliente.
- Implementación del portal de asignación de actividades y gestión documental del área de Desarrollo de la Gerencia de Tecnología Informática.
- Ajuste en el proceso de carga de archivos de las entidades financieras para la conciliación del área de Tesorería.

Las actividades que ha tenido que desarrollar y están relacionadas con la especialidad son las de analizar, desarrollar, pruebas unitarias, pruebas de usuario, desplegar y documentar aplicaciones desarrolladas en .NET, usando como gestor de base de datos Oracle, las aplicaciones han sido desarrolladas usando la estructura MVC, Web Api en ASP .Net en el lenguaje de programación C#.

La metodología ágil usada para llevar la gestión recursos, tiempos de desarrollo e hitos de entrega de los proyectos fue SCRUM.

Capítulo III: APLICACIÓN PROFESIONAL

Situación problemática

Los clientes interesados en realizar algún proyecto u obra de electrificación dentro de la concesión que tiene Electro Dunas tenía que dirigirse de manera presencial a la oficina comercial de Ica para poder presentar la solicitud del proyecto u obra que deseaba realizar, adjuntando de manera física o en algún dispositivo de almacenamiento todos los requisitos necesarios para que pueda ser atendido, estos requisitos están detallados en la Resolución Directoral N° 018-2022-EM-DGE publicado por OSINERGMIN en setiembre del 2002.

La comunicación que se daba entre el cliente interesado y Electro Dunas (y en viceversa) era a través del envío de correo electrónico o llamada telefónica, donde se consultaba el estado actual de su solicitud, el cambio de algún ingeniero residente o proyectista, la solicitud de algunos requisitos que se omitió o se equivocó en la entrega inicial, esta comunicación demandaba un tiempo determinado para el colaborador (trabajador) de Electro Dunas.

Así como la creación del Número de Identificador de Suministro (NIS) por cada solicitud que el cliente interesado presentaba, ocasionando tener más de 4 NIS para un determinado proyecto donde lo ideal es solo tener 1 NIS.

Adicionalmente dado el contexto social que vivimos por la pandemia del COVID – 19 en nuestro país, se suspendieron toda actividad presencial en la oficina comercial de Ica de Electro Dunas, ocasionando un retraso en la atención y recepción de estas solicitudes trayendo consigo que el cliente interesado no pueda realizar algún proyecto u obra.

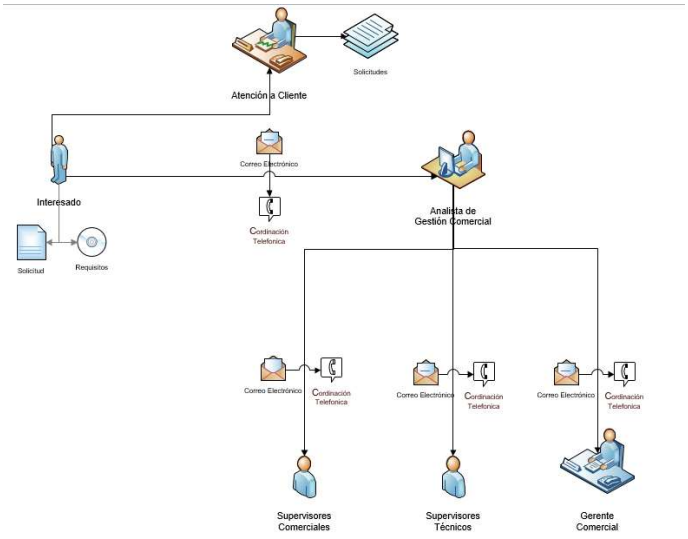


Figura 2. Flujo de Atención Presencial

Alternativa de Solución

Se desarrolló e implementó un sistema web para poder configurar los requisitos de los tipos de solicitud que se presentan en la creación de un expediente para un proyecto u obra del cliente interesado, para este desarrollo se usó la metodología SCRUM como metodología ágil.

En la Guía para el Cuerpo de Conocimiento de Scrum (Guía SBOK) [1] nos detalla que "SCRUM es uno de los métodos ágiles más populares. Es una metodología de adaptación, iterativa, rápida, flexible y eficaz, diseñada para ofrecer un valor significativo de forma rápida en todo el proyecto. SCRUM garantiza transparencia en la comunicación y crea un ambiente de responsabilidad colectiva y de progreso continuo".

Asimismo en el informe Collabnet Versiosone del año 2018 comentan que un 56% de los proyectos actuales el 56% es un mix con otras metodologías, es por ello que en el año 2020 los autores Solis Vera Gabriel Leonardo y Zamora Granados Joel Alfredo en su investigación [2] nos indican que en el desarrollo de su proyecto aplicaron la metodología ágil XP, utilizando el marco de trabajo SCRUM, apoyándose en este enfoque ágil que permite obtener los requerimientos de forma más clara y precisa, debido a las reuniones con el cliente; también realizaron pruebas automatizadas mejorando la calidad del sistema a lo largo del desarrollo de cada sprint.

La solución está guiada al desarrollo de un sistema web, es por ello que en el año 2018 el autor Mancheno Segovia Leonardo David en su proyecto de investigación [3] describe la problemática en que los procesos llevados para la gestión de soporte y mantenimiento no se ajustan a los requisitos necesarios para que se puedan ejecutar las actividades inmersas del negocio de la empresa TELNET SOPORTE, es por eso que nació la necesidad del desarrollo de una aplicación web para que se pueda disponer la información y datos que se generan dentro de la gestión del soporte técnico de la empresa, la información obtenida es vital para la toma de decisiones, brindándonos la seguridad que un sistema web brinda la información necesaria para que se pueda tomar buenas decisiones, así como el sistema web es adaptable a los recursos informáticos ya existentes, en este mismo año los investigadores Bravo Veliz Shirley Estephany y Sánchez Aranda Alonso Alberto [4] desarrollaron un sistema web cuyo resultado fue que representó el 44.7 % de la variación sobre una de las variables de la investigación (gestión de historias clínicas), de esta manera asegura que el desarrollo de un sistema web influye positivamente en los procesos.

El autor Mariano Gendra [5] en la publicación de un artículo en su página web nos brinda un cuadro comparativo entre las aplicaciones de escritorio y sistema web.

Tabla 2 Cuadro comparativo aplicaciones web vs aplicaciones de escritorio

INDICADOR	APLICACIONES DE ESCRITORIO	APLICACIONES WEB
Portabilidad	Limitada, ya que se utiliza únicamente en el ambiente en que se instaló.	Óptima, ya que se puede acceder desde cualquier dispositivo con acceso a Internet.
Actualizaciones	Implica más tiempo y trabajo, ya que se deben recorrer todas las estaciones de trabajo.	Se realizan de forma automática y simultáneamente en todos los equipos desde una única ubicación.
Incompatibilidad de versiones	Puede presentarse si no se actualizan los equipos al mismo tiempo.	No se presenta, ya que las actualizaciones se realizan simultáneamente.
Requerimiento de software	Se requiere la adquisición e instalación de un software.	No requiere.
Instalación	Sí requiere.	No requiere.
Interfaz	Amigable con el usuario.	Amigable con el usuario.
Respaldos	Se encuentran en varias estaciones de trabajo.	Están centralizados.
Sistemas operativos	Sólo se puede utilizar en dispositivos que cuenten con determinado Sistema Operativo.	No importa el Sistema Operativo instalado en el dispositivo, sólo se requiere conexión a Internet.
Conexión a Internet	Se requiere.	Es indispensable para su uso.
Tiempo de desarrollo	Mayor tiempo de desarrollo.	menor tiempo de desarrollo, hoy día se cuenta con herramientas (frameworks) que pueden acelerarlo.
Tiempo de respuesta	Más rápido.	Más lento aunque se puede mejorar con tecnologías como AJAX.
Seguridad	Es muy segura.	Es muy segura.

Concluyendo que las aplicaciones web son la mejor opción ya que desde cualquier lugar se puede tener acceso a los programas, imágenes, documentos y todo tipo de información.

Como política del área de desarrollo de la gerencia de TI la información que se registrará será gestionada por el motor de base de datos Oracle, Javier Morales Correa en su libro “Optimización SQL en Oracle” [6] nos detalla una guía completa de como explotar bases de datos Oracle de forma eficiente, así como la ejecución de SQL sea óptima. Steven Feuerstein en el libro “Oracle PL/SQL Best Practices” [7] nos ofrece respuestas prácticas a las preguntas más frecuentes y complejas que surgen en los desarrolladores de PL/SQL, asimismo incluye una recomendación de las mejores prácticas y estándares de programación.

CEPAL [8] comenta que se mejoró la economía digital como resultado de la expansión del so de plataformas digitales como modelos de negocios de oferta de bienes y servicios, al día de hoy se sigue impulsando hacia una economía digitalizada basada en modelos de producción y consumo en la incorporación de tecnologías digitales en todas las dimensiones económicas, sociales y medioambientales.

Contando con las herramientas necesarias y la optimización de procesos, se desarrolla este sistema web donde al cliente y a los trabajadores de Electro Dunas significa una gran ayuda para poder presentar y dar seguimiento a sus trámites de proyectos reduciendo los tiempos de atención y revisión de información.

Conceptualización del proceso

Para el desarrollo de la solución se hizo un relevamiento de información con las áreas usuarias las cuales son:

- Gerencia Técnica – Supervisores de Proyectos y Obras
- Gerencia Comercial – Proyectos y Obras Comercial

Esta información ha sido dividida en 03 subprocesos, los cuales son:

a. **Subproceso de configuración de expedientes y solicitudes,** este sub proceso agrupa diferentes módulos del sistema web creados para la administración de las áreas de proyectos y obras comercial, así como el área de proyectos y obras de la gerencia técnica; los módulos creados hacen referencia a la administración de los sistemas eléctricos y típicos constructivos que puede brindar la instalación Electro Dunas, así como también se administra los tipos de expedientes (proyectos a crear), también las solicitudes que tendrán que presentarse para que pueda ser atendido un determinado expediente y la configuración de los requisitos a presentar en una solicitud, así como los formatos de documentos permitidos.

   En el módulo de configuración de las solicitudes a presentar para la atención de un determinado expediente, también se configura aquellas solicitudes que implican un cambio de estado en el expediente.

b. **Subproceso de generación de expedientes y solicitudes**, este sub proceso emboca 03 módulos que han sido desarrollados para el usuario externo (interesado), el primero módulo es para que el interesado pueda crear su usuario de acceso al sistema, validando el correo electrónico ingresado.

   El segundo módulo va guiado a la generación del expediente donde el interesado podrá seleccionar el tipo de sistema eléctrico y el tipo de expediente (proyecto) que presentará hacia Electro Dunas, asimismo ingresará un nombre del proyecto como

una descripción que será un dato informativo para el trabajador de Electro Dunas, también podrá seleccionar el departamento, provincia y distrito donde se realizará este proyecto, el sistema solo mostrará lo que esté dentro de la concesión de Electro Dunas, el interesado también ingresará la demanda en kW que estará solicitando para su proyecto, también si tiene un representante del proyecto, este representante puede ser: ingeniero proyectista o ingeniero residente, de tener un representante se tendrá que elegir el departamento, provincia, distrito y la dirección fiscal del mismo, el correo electrónico, celular y número CIP.

El tercer módulo que es habilitado al cliente es para que pueda crear la solicitud correspondiente al proyecto creado inicialmente, donde puede seleccionar el tipo de solicitud que le corresponde para atenderlo, así como poder subir los requisitos configurados anteriormente en el sistema y poder enviarlo a Electro Dunas para que pueda ser revisado.

c. **Subproceso de atención de las solicitudes,** este sub proceso contiene 02 módulos y se basan en la revisión de la información ingresada por el interesado.

   El primer módulo va guiado a la revisión de información de expediente y el ingreso de los datos técnicos del proyecto, tales como:

   - Código de Troncal
   - Punto de Alimentación
   - Potencia de Corto Circuito
   - Tipo de Suministro (trifásico o monofásico)
   - Máxima demanda
   - Tensión en Kv
   - Tipo de conexión del transformador
   - Código de la SET
   - Nombre de la SET
   - Número de usuarios o de lotes
   - Moneda del VNR proyectado
   - VNR proyectado
   - UTM (X) de las estructuras
   - UTM (Y) de las estructuras
   - Zona corrosiva
   - Uso de aguas subterráneas

   El segundo módulo está guiado a la atención de la solicitud del interesado, donde en una primera instancia el "Gestor Comercial" validará la información adjuntada en los requisitos de la solicitud y de no estar correcta podrá rechazar la solicitud para que el

interesado vuelva a hacer la solicitud, pero de estar todo conforme seguirá en el flujo de atención asignándole al "Revisor Técnico" donde ellos evalúan toda la información y emiten una respuesta al interesado también pueden adjuntar recomendaciones para que el proyecto pueda ser mejorado, esta información va redireccionada a la "Asistencia de Gerencia Técnica" el cual valida que todos los documentos adjuntos por el "Revisor Técnico" sean los correctos, de no serlos retorna para que el "Revisor Técnico" lo pueda revisar, pero de ser correcto lo derivará al "Gestor Comercial" para que pueda gestionar la validez y aprobación de los documentos, derivándolos al estado "Revisión de Firmas" para que puedan emitir los documentos con la firma del Gerente Comercial y tenga validez, de una vez que tenga los documentos validados se retorna a "Gestor Comercial" para que pueda dar una respuesta al interesado, esta respuesta puede ser "Solicitud Rechazada" o "Solicitud Aceptada", de ser una solicitud rechazada se le notificará al interesado para que lo vuelva a ingresar al sistema y de ser una solicitud aceptada también se le notificará al interesado y este podrá seguir con la próxima solicitud que corresponda al flujo de atención del proyecto hasta que este pueda ser atendido en totalidad.

Desarrollo de la solución

Análisis de requerimientos

Según los procedimientos de la gerencia de tecnología informática (GTI) de Electro Dunas todo requerimiento por parte del área usuaria se debe de solicitar mediante una solicitud de "Requerimiento de Nuevo Sistema" (RNS) donde detallan el objetivo principal del sistema, la situación actual por el cual se da esta solicitud y cuáles serían los resultados esperados al momento de implementar este nuevo sistema.

Luego de la recepción de este formato, se procede a programar las reuniones de relevamiento de información y de consultas al usuario de un determinado tema, estas sesiones se documentan en el documento llamado "Formato N° 2 – Acta de Reunión" , luego de validar toda la información se procede con la creación del documento llamado "Formato N° 3 – Documento Visión", también se detalla el documento "Formato N° 4 – Metodología de desarrollo" donde se encuentran las historias de usuarios, cronograma del proyecto, plan de entregables, pila del producto, etc. Asimismo, se realiza el documento "Formato N° 5 – Casos de usos" donde se detallan las reglas de negocio y el diccionario de datos por cada subproceso que se tenga del portal.

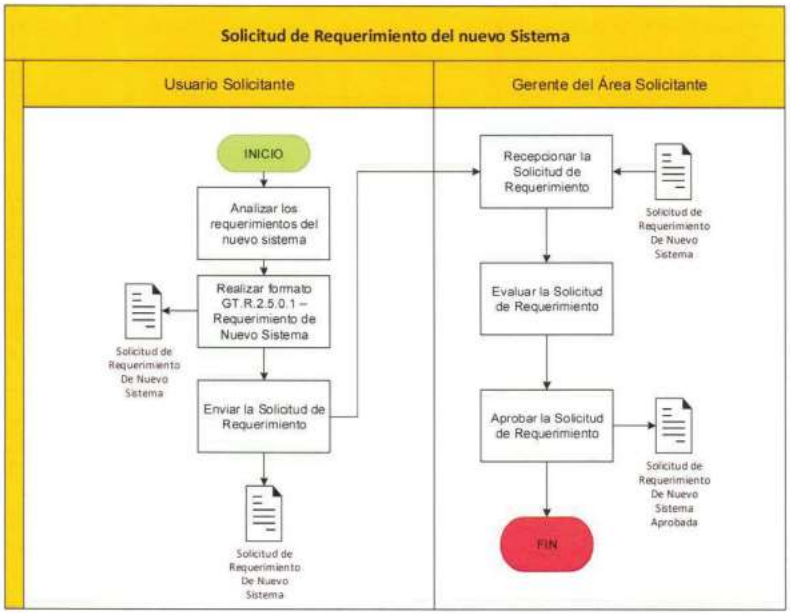


Figura 3 Flujograma de presentación de Solicitud de Requerimiento del nuevo sistema

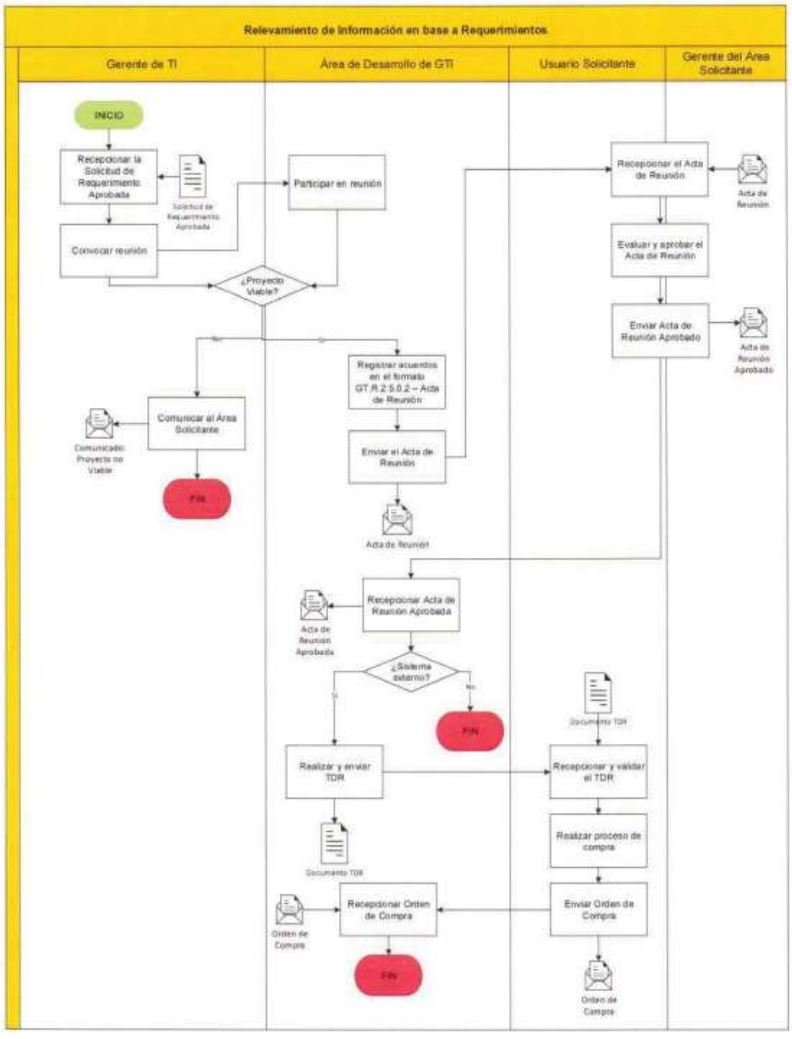


Figura 4 Flujograma del Relevamiento de Información en base a Requerimientos

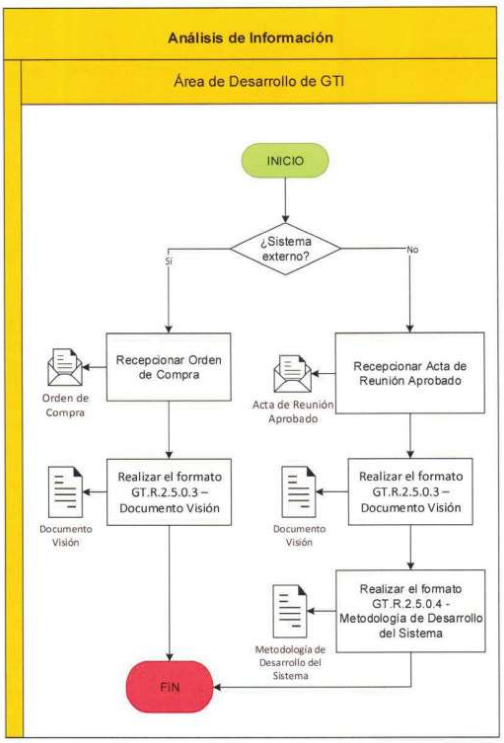


Figura 5 Flujograma de Análisis de Información

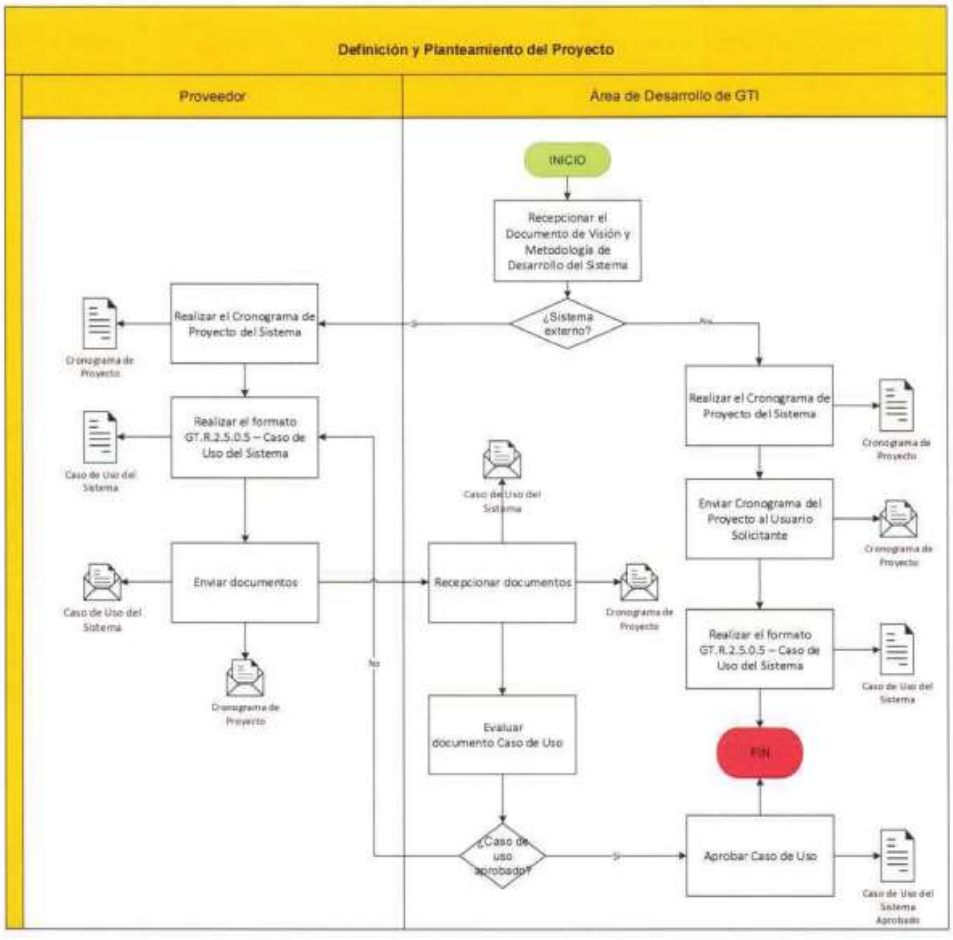


Figura 6 Flujograma de Definición y Planteamiento del Proyecto

Los requerimientos funcionales que se plantean son las que se describen a continuación

Tabla 3 Tabla de funcionalidades a implementar en el sistema

Código	Descripción
GTI-PYO-0001	Registro de usuarios internos y usuarios terceros
GTI-PYO-0002	Funcionalidad para poder administrar los módulos del portal
GTI-PYO-0003	Funcionalidad para poder crear los perfiles del portal y asignarlos a los usuarios
GTI-PYO-0004	Funcionalidad para administrar los tipos de sistemas eléctricos
GTI-PYO-0005	Funcionalidad para administrar los tipos de solicitudes
GTI-PYO-0006	Funcionalidad para administrar los tipos de proyectos
GTI-PYO-0007	Funcionalidad para administrar los típicos constructivos
GTI-PYO-0008	Funcionalidad para poder consultar los diferentes tipos de proyectos, requisitos a presentar en las etapas de un proyecto (tipo de solicitudes), así como los típicos constructivos
GTI-PYO-0009	Funcionalidad para que el usuario externo (tercero) cree y de seguimiento a sus expedientes
GTI-PYO-0010	Funcionalidad para que el usuario externo (tercero) cree y de seguimiento a sus solicitudes
GTI-PYO-0011	Funcionalidad para que personal de Electro Dunas pueda revisar expedientes
GTI-PYO-0012	Funcionalidad para que personal de Electro Dunas pueda revisar solicitudes y posteriormente derivarlo para que pueda ser atendido
GTI-PYO-0013	Funcionalidad para que el personal de Electro Dunas pueda generar los reportes

Para cada uno de las funcionalidades requeridas para el sistema se realizó el análisis respectivo para poder determinar los datos necesarios y el detalle de cada uno de ellos.

Para la primera funcionalidad GTI–PYO–0001 “Registro de usuarios internos y usuarios externos”, esta opción tiene como finalidad contar con una funcionalidad donde se administren los Usuarios que interactúan con el Portal. Los Usuarios pueden ser creados de dos formas: directamente por esta funcionalidad si es el usuario es interno a la organización y por la funcionalidad del Registro del Interesado (tercero), a este tipo de registro se cataloga como un Usuario externo.

Cuando un usuario interno se crea, por defecto debe crearse una contraseña aleatoria y debe ser enviado al correo que está registrando para que luego cuando el usuario ingrese le solicite cambiar la contraseña.

Cuando un Usuario no se acuerde de la contraseña, debe tener una funcionalidad que le permita pedir un correo para el envío de la contraseña temporal para que luego proceda a cambiarla y ese dato se refleje en este Mantenimiento.

Cuando un Usuario es Interno se debe colocar en forma obligatoria la Gerencia y Jefatura a la que pertenece. Este dato se podría tomar de una vista del Sistema Máximo (sistema interno de Electro Dunas).

Como parte de la información se encuentran los datos de auditoría que al momento de crear se llenan los 04 campos que son: Usuario creación, fecha creación, usuario modificación y fecha de modificación. Cuando se actualiza o elimina lógicamente el registro solo se actualiza los últimos campos.

Los datos para registrar son:

Tabla 4 Datos para registro de usuarios internos: GTI-PYO-0001

Campo	Descripción	Editable (SI/NO)	Tipo Dato	Tamaño	Obligatorio (SI / NO)
Código	Código de Usuario	SI	Cadena	25	SI
Nombres	Primer y Segundo Nombre del Interesado	SI	Cadena	100	SI
Apellido Paterno	Apellido Paterno Interesado	SI	Cadena	100	SI
Apellido Materno	Apellido Materno del Interesado	SI	Cadena	100	SI
Correo Electrónico	Correo Electrónico	SI	Cadena	100	SI
Clave	Contraseña que le permitirá el acceso al Portal	SI	Cadena	50	SI
Confirmar la Clave	Que le permita repetir la contraseña para validación	SI	Cadena	50	SI
Tipo de Documento	Tipo de Documento de Identidad	SI	Cadena	3	NO
Número de Identidad	Número de Identidad	SI	Cadena	11	NO
País de Residencia	País donde reside el Interesado	SI	Cadena	50	NO
Ubigeo	Código del Departamento, Provincia y Distrito de Residencia.	SI	Cadena	35	NO

Dirección Postal	Dirección de Residencia del Interesado	SI	Cadena	250	NO
Teléfono Fijo	Número de Teléfono Fijo del lugar de Residencia.	SI	Cadena	50	NO
Teléfono Celular	Número del Teléfono Celular del Interesado	SI	Cadena	50	NO
Teléfono en caso de Emergencia	Número de Teléfono alterno para poder ubicar al interesado	SI	Cadena	50	NO
Tipo Usuario	Tipo de Usuario. Por defecto debe ir el valor Interno	NO	Cadena	10	SI
Gerencia	Gerencia a la que pertenece el Usuario	SI	Cadena	02	SI
Jefatura	Jefatura a la que pertenece el Usuario	SI	Cadena	02	SI
Estado	Estado Activo o Inactivo	SI	Lógico	1	SI
Acepta condiciones generales	Indicador que el Interesado acepta que sus Datos sean usados por las empresas vinculadas.	NO	Cadena	1	NO
Acepta las condiciones para la subscripción	Indicador que el Interesado acepta los términos y condiciones y protección de datos	NO	Cadena	1	NO
Usuario Creación	Código del usuario que genero la transacción	NO	Cadena	10	SI
Fecha de Creación	Fecha que se generó la transacción	NO	Fecha		SI
Usuario Última Modificación	Código del usuario que realizó la última modificación de la transacción	NO	Cadena	10	SI

Fecha de Ultima Modificación	Fecha que se realizó la última modificación a la transacción.	NO	Fecha		SI

La opción para poder crear un usuario externo será utilizada al momento de crear un interesado y cuando se desee modificar cierta información del mismo.

Cuando un interesado se crea, por defecto debe crearse una contraseña aleatoria y debe ser enviado al correo electrónico que está registrando para que luego cuando el usuario ingrese le solicite cambiar la contraseña.

Cuando el interesado no se acuerde de la contraseña en la ventana del login le debe de pedir un correo para el envío de la contraseña temporal para que luego proceda a cambiarla y ese dato se refleje en este mantenimiento.

La información que se debe de gestionar en esta opción es la siguiente:

Tabla 5 Datos para registro de usuarios externos: GTI-PYO-0001

Campo	Descripción	Editable (SI/NO)	Tipo Dato	Tamaño	Obligatorio (SI / NO)
Código	Código de Usuario	SI	Cadena	25	SI
Nombres	Primer y Segundo Nombre del Interesado	SI	Cadena	100	SI
Apellido Paterno	Apellido Paterno Interesado	SI	Cadena	100	SI
Apellido Materno	Apellido Materno del Interesado	SI	Cadena	100	SI
Correo Electrónico	Correo Electrónico	SI	Cadena	100	SI
Clave	Contraseña que le permitirá el acceso al Portal	SI	Cadena	50	SI
Confirmar la Clave	Que le permita repetir la contraseña para validación	SI	Cadena	50	SI
Tipo de Documento	Tipo de Documento de Identidad	SI	Cadena	3	SI

Número de Identidad	Número de Identidad	SI	Cadena	11	SI
País de Residencia	País donde reside el Interesado	SI	Cadena	50	SI
Ubigeo	Código del Departamento, Provincia y Distrito de Residencia.	SI	Cadena	35	SI
Dirección Postal	Dirección de Residencia del Interesado	SI	Cadena	250	SI
Teléfono Fijo	Número de Teléfono Fijo del lugar de Residencia.	SI	Cadena	50	NO
Teléfono Celular	Número del Teléfono Celular del Interesado	SI	Cadena	50	SI
Teléfono en caso de Emergencia	Número de Teléfono alterno para poder ubicar al interesado	SI	Cadena	50	SI
Tipo Usuario	Tipo de Usuario. Por defecto debe ir el valor Externo	NO	Cadena	10	SI
Estado	Estado Activo o Inactivo	SI	Lógico	1	SI
Acepta condiciones generales	Indicador que el Interesado acepta que sus Datos sean usados por las empresas vinculadas.	NO	Cadena	1	SI
Acepta las condiciones para la subscripción	Indicador que el Interesado acepta los términos y condiciones y protección de datos	NO	Cadena	1	SI
Usuario Creación	Código del usuario que genero la transacción	NO	Cadena	10	SI
Fecha de Creación	Fecha que se generó la transacción	NO	Fecha		SI

Usuario Última Modificación	Código del usuario que realizó la última modificación de la transacción	NO	Cadena	10	SI
Fecha de Ultima Modificación	Fecha que se realizó la última modificación a la transacción.	NO	Fecha		SI

Luego del registro del interesado se le debe asociar un rol predefinido con los siguientes accesos al sistema:

- Registrar como usuario externo
- Cambiar la contraseña
- Accesos a las consultas del portal
- Acceso a creación y seguimiento de expedientes
- Acceso a creación y seguimiento de solicitudes

En ambos casos al momento de crear un usuario se enviará mediante correo electrónico las credenciales de acceso.

Para la funcionalidad GTI–PYO–0002 "Funcionalidad para poder administrar los módulos del portal", esta opción tiene como finalidad contar con un mantenimiento donde se administre las opciones o funcionalidades que los usuarios (internos o externos) van a tener acceso en el portal.

Como parte de la información se encuentre los 04 datos de auditoría que son: usuario creación, fecha creación, usuario modificación y fecha de modificación. Cuando se actualiza o elimina lógicamente el registro solo actualizará los últimos 02 campos.

Tabla 6 Datos a Ingresar para los módulos: GTI-PYO-0002

Campo	Descripción	Editable (SI/NO)	Tipo Dato	Tamaño	Obligatorio (SI / NO)
Código	Código de la funcionalidad u opción	SI	Cadena	25	SI
Descripción	Descripción de la Funcionalidad	SI	Cadena	250	SI
Estado	Activo / Inactivo	SI	Cadena	1	SI

Usuario Creación	Código del usuario que genero la transacción	NO	Cadena	10	SI
Fecha de Creación	Fecha que se generó la transacción	NO	Fecha		SI
Usuario Última Modificación	Código del usuario que realizó la última modificación de la transacción	NO	Cadena	10	SI
Fecha de Ultima Modificación	Fecha que se realizó la última modificación a la transacción.	NO	Fecha		SI

Para la funcionalidad correspondiente a GTI-PYO-0003 "Funcionalidad para poder crear los perfiles del portal y asignarlos a los usuarios" se considera para poder administrar todos los roles o perfiles que deben asignarse a los usuarios (internos o externos) que interactúan con el portal, cuando se cree un rol se debe asociar uno más módulos y este rol puede ser asignado a uno o más usuarios.

Como parte de la información se encuentran los datos de auditoría que al momento de crear se llenan los cuatro campos que son: usuario creación, fecha creación, usuario modificación, fecha modificación, cuando se actualiza o elimine lógicamente el registro solo se actualiza los últimos campos.

Se plantea la pre condicional en que exista primero los módulos

Tabla 7 Datos a Ingresar para los perfiles: GTI-PYO-0003

Campo	Descripción	Editable (SI/NO)	Tipo Dato	Tamaño	Obligatorio (SI / NO)
Código	Código del Grupo o Rol	SI	Cadena	25	SI
Descripción	Descripción del Grupo o Rol	SI	Cadena	250	SI
Estado	Activo / Inactivo	SI	Cadena	1	SI
Usuario Creación	Código del usuario que	NO	Cadena	10	SI

	genero la transacción				
Fecha de Creación	Fecha que se generó la transacción	NO	Fecha		SI
Usuario Última Modificación	Código del usuario que realizó la última modificación de la transacción	NO	Cadena	10	SI
Fecha de Ultima Modificación	Fecha que se realizó la última modificación a la transacción.	NO	Fecha		SI

La relación de un módulo con el rol o perfil debe tener los siguientes datos:

Tabla 8 Datos a guardar de relación entre Perfil y Módulo: GTI-PYO-0003

Campo	Descripción	Editable (SI/NO)	Tipo Dato	Tamaño	Obligatorio (SI / NO)
Código	Código del Grupo o Rol	NO	Cadena	25	SI
Código Opción o Funcionalidad	Código Opción o Funcionalidad	SI	Cadena	25	SI
Estado	Activo / Inactivo	SI	Cadena	1	SI
Usuario Creación	Código del usuario que genero la transacción	NO	Cadena	10	SI
Fecha de Creación	Fecha que se generó la transacción	NO	Fecha		SI
Usuario Última Modificación	Código del usuario que realizó la última modificación de la transacción	NO	Cadena	10	SI
Fecha de Ultima Modificación	Fecha que se realizó la última modificación a la transacción.	NO	Fecha		SI

La relación que debe de tener el rol con el usuario (interno o externo) debe de tener los siguientes datos:

Tabla 9 Datos a guardar de relación entre Perfil y Usuario: GTI-PYO-0003

Campo	Descripción	Editable (SI/NO)	Tipo Dato	Tamaño	Obligatorio (SI / NO)
Código	Código del Grupo o Rol	NO	Cadena	25	SI
Código Usuario	Código del Usuario	SI	Cadena	25	SI
Estado	Activo / Inactivo	SI	Cadena	1	SI
Usuario Creación	Código del usuario que genero la transacción	NO	Cadena	10	SI
Fecha de Creación	Fecha que se generó la transacción	NO	Fecha		SI
Usuario Última Modificación	Código del usuario que realizó la última modificación de la transacción	NO	Cadena	10	SI
Fecha de Ultima Modificación	Fecha que se realizó la última modificación a la transacción.	NO	Fecha		SI

Para la funcionalidad GTI-PYO-0004 “Funcionalidad para administrar el tipo de sistema eléctrico” se considera como requisito para poder crear los Tipos de Proyectos y que se puedan ejecutar, como parte de la información se encuentran los datos de auditoría que al momento de crear se llenan los 04 campos que son: usuario creación, fecha creación, usuario modificación y fecha de modificación, cuando se actualiza o elimina lógicamente el registro solo se actualiza los últimos campos.

Los datos a registrar en esta funcionalidad son:

Tabla 10 Datos a Ingresar para los tipo de sistema eléctrico: GTI-PYO-0004

Campo	Descripción	Editable (SI/NO)	Tipo Dato	Tamaño	Obligatorio (SI / NO)
Código del Tipo de Sistema Eléctrico	Código del Tipo de Sistema Eléctrico	SI	Cadena	25	SI
Descripción del Tipo de Sistema Eléctrico	Descripción del Tipo de Sistema Eléctrico	SI	Cadena	250	SI
Estado	Activo / Inactivo	SI	Cadena	1	SI
Usuario Creación	Código del usuario que genero la transacción	NO	Cadena	10	SI
Fecha de Creación	Fecha que se generó la transacción	NO	Fecha		SI
Usuario Última Modificación	Código del usuario que realizó la última modificación de la transacción	NO	Cadena	10	SI
Fecha de Ultima Modificación	Fecha que se realizó la última modificación a la transacción.	NO	Fecha		SI

La funcionalidad GTI-PYO-0005 “Funcionalidad para administrar los tipos de solicitudes” se considera para que el usuario externo o interesado pueda presentar su documentación en el portal y pueda crear su expediente.

Este mantenimiento debe de permitir asociar los requisitos a presentar por un interesado y cada requisito debe de estar asociado al tipo de sistema eléctrico donde desarrollará su proyecto u obra. Por lo tanto, tendrá que ingresar datos en la cabecera y detalle.

Como parte de la información se encuentran los datos de auditoría que al momento de crear se llenan los 04 campos que son: usuario creación, fecha creación, usuario de modificación y fecha de modificación. Cuando se actualiza o elimina lógicamente el registro solo se actualiza los últimos campos.

La pre condición para esta funcionalidad es que exista tipo de sistema eléctrico.

Los datos a registrar en la cabecera del mantenimiento son:

Tabla 11 Datos a Ingresar para los Tipos de Solicitudes: GTI-PYO-0005

Campo	Descripción	Editable (SI/NO)	Tipo Dato	Tamaño	Obligatorio (SI / NO)
Código del Tipo de Solicitud	Código del Tipo de Solicitud	SI	Cadena	25	SI
Descripción	Descripción del Tipo de Solicitud	SI	Cadena	50	SI
Detalle del Tipo de Solicitud	Descripción completa del Tipo de Solicitud	SI	Cadena	250	SI
Plazo de Atención	Número de días para la atención.	NO	Numérico		SI
Estado	Activo / Inactivo	SI	Cadena	1	SI
Usuario Creación	Código del usuario que genero la transacción	NO	Cadena	10	SI
Fecha de Creación	Fecha que se generó la transacción	NO	Fecha		SI
Usuario Última Modificación	Código del usuario que realizó la última modificación de la transacción	NO	Cadena	10	SI
Fecha de Ultima Modificación	Fecha que se realizó la última modificación a la transacción.	NO	Fecha		SI

Los datos del detalle a registrar son:

Tabla 12 Datos a Ingresar en el Detalle de los Tipos de Solicitudes: GTI-PYO-0005

Campo	Descripción	Editable (SI/NO)	Tipo Dato	Tamaño	Obligatorio (SI / NO)

Código del Tipo de Solicitud	Código del Tipo de Solicitud	SI	Cadena	25	SI
Código del Tipo de Sistema Eléctrico	Código del Tipo de Sistema Eléctrico	SI	Cadena	25	SI
Secuencia del Registro	Correlativo del registro (autogenerado)	NO	Numérico	-	SI
Descripción del Requisito	descripción del Requisito	SI	Cadena	80000	SI
Estado	Activo / Inactivo	SI	Cadena	1	SI
Usuario Creación	Código del usuario que genero la transacción	NO	Cadena	10	SI
Fecha de Creación	Fecha que se generó la transacción	NO	Fecha		SI
Usuario Última Modificación	Código del usuario que realizó la última modificación de la transacción	NO	Cadena	10	SI
Fecha de Ultima Modificación	Fecha que se realizó la última modificación a la transacción.	NO	Fecha		SI

Asimismo, para la funcionalidad GTI-PYO-0006 “Funcionalidad para administrar los tipos de proyectos” debe permitir ingresar los tipos de solicitud que son necesarios para que un proyecto avance en el portal mediante la funcionalidad de un flujo de trabajo. Por lo tanto, este mantenimiento debe tener una cabecera y detalle.

Como parte de la información se encuentran los datos de auditoría que al momento de crear se llenan los 04 campos que son: usuario creación, fecha creación, usuario modificación y fecha modificación. Cuando se actualiza o elimina lógicamente el registro solo se actualiza los últimos campos.

Las pre condiciones para esta funcionalidad es la existencia de tipo de sistemas eléctricos y tipo de solicitudes en el portal.

Los datos a registrar en la cabecera del mantenimiento son:

Tabla 13 Datos a Ingresar para los Tipos de Proyectos: GTI-PYO-0006

Campo	Descripción	Editable (SI/NO)	Tipo Dato	Tamaño	Obligatorio (SI / NO)
Código del Tipo de Proyecto	Código del Proyecto	SI	Cadena	25	SI
Descripción del Tipo de Proyecto	Descripción del Proyecto	SI	Cadena	250	SI
Se relaciona con Proyecto Anterior	Debe mostrarse el valor de SI o NO.	SI	Cadena	2	SI
Código de Tipo de Proyecto Relacionado	Código de Tipo de Proyecto Relacionado	SI	Cadena	25	NO
Código del Sistema Eléctrico	Código del Sistema Eléctrico.	SI	Cadena	25	SI
Estado	Activo / inactivo	SI	Cadena	1	SI
Usuario Creación	Código del usuario que genero la transacción	NO	Cadena	10	SI
Fecha de Creación	Fecha que se generó la transacción	NO	Fecha		SI
Usuario Última Modificación	Código del usuario que realizó la última modificación de la transacción	NO	Cadena	10	SI
Fecha de Ultima Modificación	Fecha que se realizó la última modificación a la transacción.	NO	Fecha		SI

Los datos a registrar en el detalle del mantenimiento son:

Tabla 14 Datos a Ingresar en la relación de Tipo de Solicitud y Tipos de Proyectos: GTI-PYO-0006

Campo	Descripción	Editable (SI/NO)	Tipo Dato	Tamaño	Obligatorio (SI / NO)
Código del Tipo de Proyecto	Descripción del Proyecto	NO	Cadena	25	SI
Código del tipo de Solicitud que debe tener el Proyecto	Código del Tipo de Solicitud	SI	Cadena	25	SI
Orden	Orden en que se deba tramitar cada una de los Tipos de Solicitudes.	SI	Numérico		SI
Estado	Activo / inactivo	SI	Cadena	1	SI
Usuario Creación	Código del usuario que genero la transacción	NO	Cadena	10	SI
Fecha de Creación	Fecha que se generó la transacción	NO	Fecha		SI
Usuario Última Modificación	Código del usuario que realizó la última modificación de la transacción	NO	Cadena	10	SI
Fecha de Ultima Modificación	Fecha que se realizó la última modificación a la transacción.	NO	Fecha		SI

Se debe de contar con la funcionalidad GTI-PYO-0007 “Funcionalidad para administrar los típicos constructivos” para que pueda ser consultado por un interesado en el portal, esta funcionalidad debe permitir anexar archivos que corresponden a los diseños, materiales y dimensiones que deben de tener los típicos constructivos.

Como parte de la información se encuentran los datos de auditoría que al momento de crear se llenan los 04 campos que son: usuario creación, fecha creación, usuario modificación y fecha modificación. Cuando se actualiza o elimina lógicamente el registro solo se actualiza los últimos campos.

Los datos a registrar son:

Tabla 15 Datos a Ingresar para los Típicos Constructivos: GTI-PYO-0007

Campo	Descripción	Editable (SI/NO)	Tipo Dato	Tamaño	Obligatorio (SI / NO)
Código	Código del Típico Constructivo	SI	Cadena	25	SI
Descripción	Descripción del Típico Constructivo	SI	Cadena	250	SI
Estado	Activo / inactivo	SI	Cadena	1	SI
Usuario Creación	Código del usuario que genero la transacción	NO	Cadena	10	SI
Fecha de Creación	Fecha que se generó la transacción	NO	Fecha		SI
Usuario Última Modificación	Código del usuario que realizó la última modificación de la transacción	NO	Cadena	10	SI
Fecha de Ultima Modificación	Fecha que se realizó la última modificación a la transacción.	NO	Fecha		SI

Se puede asociar una o más archivos en formato PDF en esta funcionalidad por lo que debe de tener una información de detalle y los datos a guardar son:

Tabla 16 Datos a Ingresar al guardar el archivo del Típico Constructivo: GTI-PYO-0007

Campo	Descripción	Editable (SI/NO)	Tipo Dato	Tamaño	Obligatorio (SI / NO)
Código	Código del Típico Constructivo	SI	Cadena	25	SI
Contenido	Contenido del archivo	SI	Imagen		SI

Estado	Activo / inactivo	SI	Cadena	1	SI
Usuario Creación	Código del usuario que genero la transacción	NO	Cadena	10	SI
Fecha de Creación	Fecha que se generó la transacción	NO	Fecha		SI
Usuario Última Modificación	Código del usuario que realizó la última modificación de la transacción	NO	Cadena	10	SI
Fecha de Ultima Modificación	Fecha que se realizó la última modificación a la transacción.	NO	Fecha		SI

Entre las opciones del portal de proyectos y obras de terceros debe de tener la funcionalidad de poder realizar consultas de los diferentes tipos de proyectos, requisitos a presentar en las etapas de un proyecto (tipo de solicitudes), así como los típicos constructivos (GTI-PYO-0008), para que esta funcionalidad se pueda mostrar de manera correcta se tiene las siguientes pre condiciones:

- Contar con los tipos de proyectos y sus etapas relacionadas en el portal.
- Contar con los requisitos a presentar por etapa de un determinado tipo de proyecto.
- Contar con los sistemas eléctricos.
- Contar con la información de los típicos constructivos.

El interesado tendrá una funcionalidad que le permita revisar en el portal los diferentes tipos de proyecto que se pueden desarrollar en los sistemas de distribución y utilización y pueda con base presentar los documentos que demanda cada uno de ellos.

Esta consulta le debe de permitir ver por cada proyecto las etapas por las que debe de pasar, en esta funcionalidad le debe permitir al tercero exportar en PDF y Excel.

La información que se debe poder visualizar en esta opción es la siguiente:

Tabla 17 Información a mostrar en la consulta de los Tipos de Proyectos: GTI-PYO-0008

Campo	Descripción	Editable (SI/NO)	Tipo Dato	Tamaño	Obligatorio (SI / NO)

Descripción del Proyecto	Descripción del Proyecto	NO	Cadena	250	SI
Se relaciona con Proyecto Anterior	Debe mostrarse el valor de SI o NO.	NO	Cadena	2	SI
Tipo de Sistema Eléctrico	Descripción del Tipo de Sistema Eléctrico relacionado al Proyecto	NO	Cadena	250	
Etapas del Proyecto o Tipos Solicitudes relacionadas al Tipo de Proyecto	Debe mostrar una X si el tipo de proyecto necesita que se presente una solicitud para esta etapa del proyecto.	NO	Cadena	1	SI

De esta manera el interesado tomará de conocimiento las etapas o los tipos de solicitud que debe de presentar para el tipo de proyecto a desarrollar, asimismo contará con la funcionalidad de poder ver los requisitos a presentar en cada una de estas etapas, para ello el portal le debe de permitir al usuario exportar en PDF y Excel.

Esta funcionalidad mostrará la siguiente información:

Tabla 18 Estructura de información a mostrar en la consulta de los Tipos de Proyectos: GTI-PYO-0008

Campo	Descripción	Editable (SI/NO)	Tipo Dato	Tamaño	Obligatorio (SI / NO)
Columnas					
Descripción de los Sistemas Eléctricos	Descripción de los Sistemas Eléctricos	NO	Cadena	250	SI
Filas					
Descripción de la etapa del tipo de Proyecto	Descripción de la etapa del tipo de Proyecto	NO	Cadena	250	SI
Contenido					
Requisito por cada Sistema Eléctrico y Etapa de cada	Requisito por cada Sistema Eléctrico y Etapa	NO	Cadena	250	SI

Tipo de Proyecto	de cada Tipo de Proyecto				

Para poder consultar los típicos constructivos el portal contará con la funcionalidad en la que el interesado pueda revisar los diferentes tipos constructivos que se puedan construir en los sistemas de distribución y utilización, de esta manera tomará conocimiento de las medidas y materiales que se deben de emplear. La opción le debe permitir descargar el archivo que se muestra.

La información que se debe poder visualizar en esta opción es la siguiente:

Tabla 19 Información a mostrar en la consulta del Típico Constructivo: GTI-PYO-0008

Campo	Descripción	Editable (SI/NO)	Tipo Dato	Tamaño	Obligatorio (SI / NO)
Descripción	Descripción del típico constructivo	NO	Cadena	250	SI
Contenido	Mostrar el contenido del archivo	NO			SI

El portal contará con la funcionalidad GTI-PYO-0009 "Funcionalidad para que el usuario externo (tercero) cree y de seguimiento a sus expedientes", esta opción tiene como finalidad que el tercero pueda registrar un expediente para iniciar el trámite de un Proyecto y/o Obra, en esta funcionalidad dependiendo del Tipo de Proyecto a elegir debe mostrar las etapas que le correspondería tramitar mediante solicitudes.

También, en caso de que un expediente dependa de otro proyecto anteriormente presentado debe poder permitir relacionar el último expediente siempre y cuando el Tipo de Proyecto relacionado al Proyecto haya sido generado por el interesado y el expediente anterior se encuentre dentro del plazo.

Un expediente debe tener relacionado los proyectistas o ingenieros residentes encargados del proyecto y que el interesado declara van a interactuar con los supervisores técnicos en Electro Dunas. Esta información no es obligatoria y solo se debe llenar siempre y cuando haya marcado el Check que cuenta con ellos. Un interesado a lo largo del Proyecto y/o Obra puede cambiar de proyectistas e ingenieros residentes por lo que es necesario generar una tabla histórica de los cambios con los datos de auditoría y cuando suceda el

cambio se debe de comunicar vía correo electrónico al interesado, el gestor comercial que este asociado al Expediente y el Supervisor Técnicos.

Cuando se ree un expediente debe tener la siguiente máscara para el código: EXP-YYYY-NNNNNN, donde:

- EXP: valor fijo y corresponde a las siglas de expediente.
- YYYY: año en que el expediente ha sido creado
- NNNNNN: Correlativo numérico que se debe incrementar cada vez que se cree un expediente y se debe reiniciar cuando empieza el año.

Debe permitir al interesado, grabar y eliminar un expediente, esta última acción solo procederá si el expediente no tiene una solicitud en proceso de admisión.

Un expediente tendrá varios estados los cuales cambiarán mediante los flujos de trabajo que se desarrollen en este sistema. Cuando un expediente tenga una o más solicitudes asociadas la información que contenga no puede ser modificar.

Los estados por los que pasará un expediente son:

Tabla 20 Estados de expediente: GTI-PYO-0009

Estado	**Actividades que lo origina**
Creado	Cuando un expediente es creado por in Interesado
Anulado	Cuando un expediente es anulado lógicamente por un Interesado y cuando no tiene una solicitud pendiente de respuesta o Cuando el tiempo o plazo que debió tramitar sus solicitudes a caducado.
Aprobado	Cuando la solicitud de la etapa de aprobación de proyecto es Aceptada.
Desarrollado	Cuando la solicitud de la etapa de inicio de obra del Proyecto ha sido aceptada.
Cerrado	Cuando la solicitud de la etapa de conformidad de obra ha sido Aceptada.

Los datos que se deben de registrar en un expediente son:

Tabla 21 Información a guardar cuando se crea un expediente: GTI-PYO-0009

Campo	**Descripción**	**Editable (SI/NO)**	**Tipo Dato**	**Tamaño**	**Obligatorio (SI / NO)**
Código	Código de Expediente	NO	Cadena	25	SI

	(Autogenerado: EXP-YYYY - NNNNNN)				
Descripción	Descripción del Proyecto y/o Obra a presentar	SI	Cadena	250	SI
Tipo de Proyecto	Código del Tipo de Proyecto y/o Obra a realizarse	NO	Cadena	25	SI
Código de Expediente Relacionado	Se debe colocar el código del Expediente relacionado.	SI	Cadena	25	NO
Fecha de Expediente	Fecha en que se crea el Expediente en el Portal	NO	Fecha		SI
Fecha Inicio	Fecha Inicio del Expediente. Día en que lo creó.	NO	Fecha		SI
Fecha Fin del Expediente	Fecha Inicio + 02 años	NO	Fecha		SI
Estado	Estado del Expediente con base al Flujo de Estados.	NO	Cadena	25	SI
Ubicación geográfica	Ubigeo donde se llevará a cabo el proyecto y/o Obra	SI	Cadena	6	SI
Coordenada X	Posición geográfica del Proyecto (X)	SI	Decimal		SI
Coordinada Y	Posición geográfica del Proyecto (Y)	SI	Decimal		SI
Objetivo del Proyecto	Breve descripción del	SI	Cadena	500	SI

	Objetivo del Proyecto				
Máxima Demanda Kw	Maxima Demanda que solicita el Interesado	SI	Entero	18	SI
Tiene Representante	Debe poder seleccionar el valor SI/ NO	SI	Cadena	2	SI
Usuario Creación	Código del usuario que genero la transacción	NO	Cadena	10	SI
Fecha de Creación	Fecha que se generó la transacción	NO	Fecha		SI
Usuario Última Modificación	Código del usuario que realizó la última modificación de la transacción	NO	Cadena	10	SI
Fecha de Ultima Modificación	Fecha que se realizó la última modificación a la transacción.	NO	Fecha		SI

La información que debe tener para almacenar los Proyectistas o ingenieros residentes son:

Tabla 22 Información a guardar del proyectista asignado a un expediente: GTI-PYO-0009

Campo	Descripción	Editable (SI/NO)	Tipo Dato	Tamaño	Obligatorio (SI / NO)
Código	Código de Expediente (Autogenerado: EXP-YYYY - NNNNNN)	NO	Cadena	25	SI

Correlativo	Secuencia de Registro a generar	NO	Numérico		SI
Tipo de Represéntate	Debe poder elegir los valores de Proyectista o Ingeniero Residente o Interesado	SI	Cadena	25	NO
Nombre del Representante	Nombres y Apellidos del Ingeniero Proyectista o Residente	SI	Cadena	250	NO
Número C.I. P	Número de Registro en el Colegios de Ingenieros del Perú	SI	Cadena	25	NO
Teléfono Fijo	Número de teléfono Fijo	SI	Cadena	25	NO
Teléfono Celular	Número de teléfono celular	SI	Cadena	25	NO
Correo Electrónico	Correo Electrónico para contacto	SI	Cadena	100	Si
Ubigeo	Código del Departamento, Provincia y Distrito de Residencia.	SI	Cadena	35	SI
Dirección Postal	Dirección de Residencia del Interesado	SI	Cadena	250	SI
Usuario Creación	Código del usuario que genero la transacción	NO	Cadena	10	SI
Fecha de Creación	Fecha que se generó la transacción	NO	Fecha		SI
Usuario Última Modificación	Código del usuario que realizó la última	NO	Cadena	10	SI

	modificación de la transacción				
Fecha de Ultima Modificación	Fecha que se realizó la última modificación a la transacción.	NO	Fecha		SI

La información que debe tener para almacenar cuando ocurren cambios de proyectista es:

Tabla 23 Información a guardar cuando se cambia de proyectista en un expediente: GTI-PYO-0009

Campo	Descripción	Editable (SI/NO)	Tipo Dato	Tamaño	Obligatorio (SI / NO)
Código	Código de Expediente (Autogenerado: EXP-YYYY - NNNNNN)	NO	Cadena	25	SI
Correlativo	Secuencia de Registro a generar	NO	Numérico		SI
Nombre del Representante	Nombres y Apellidos del Ingeniero Proyectista o Residente	SI	Cadena	250	NO
Número C.I. P	Número de Registro en el Colegios de Ingenieros del Perú	SI	Cadena	25	NO
Teléfono Celular	Número de teléfono celular	SI	Cadena	25	NO
Correo Electrónico	Correo Electrónico para contacto	SI	Cadena	100	Si
Usuario Creación	Código del usuario que genero la transacción	NO	Cadena	10	SI
Fecha de Creación	Fecha que se generó la transacción	NO	Fecha		SI

Usuario Última Modificación	Código del usuario que realizó la última modificación de la transacción	NO	Cadena	10	SI
Fecha de Ultima Modificación	Fecha que se realizó la última modificación a la transacción.	NO	Fecha		SI

Cuando un expediente pase de un estado a otro, debe almacenarse estos cambios como históricos y debe contar con la siguiente información:

Tabla 24 Información a guardar en el cambio de estado de un expediente: GTI-PYO-0009

Campo	Descripción	Editable (SI/NO)	Tipo Dato	Tamaño	Obligatorio (SI / NO)
Código	Código del Expediente (Autogenerado: EXP-YYYY - NNNNNN)	NO	Cadena	25	SI
Correlativo	Secuencia de Registro a generar	NO	Numérico		SI
Estado	Estado que tenía antes del cambio	NO	Cadena	25	SI
Usuario Creación	Código del usuario que genero la transacción	NO	Cadena	10	SI
Fecha de Creación	Fecha que se generó la transacción	NO	Fecha		SI
Usuario Última Modificación	Código del usuario que realizó la última modificación de la transacción	NO	Cadena	10	SI

Fecha de Ultima Modificación	Fecha que se realizó la última modificación a la transacción.	NO	Fecha		SI

El portal tendrá la funcionalidad GTI-PYO-0010 que es la “Funcionalidad para que el usuario externo (tercero) cree y de seguimiento a sus solicitudes” y tendrá como finalidad registrar las solicitudes que representan las diferentes etapas del Tipo de Proyecto y/o Obra. Para que se pueda generar una solicitud en el portal primero debe de existir un expediente. Caso contrario no es posible crearlos, por lo que el interesado debe ingresar al expediente para relacionar las solicitudes que están relacionados al tipo de proyecto relacionado al expediente.

También, esta solicitud debe de mostrarlos todos los requisitos que deben adjuntar para que se pueda crear. Y por cada requisito le debe permitir adjuntar un archivo, estos archivos pueden ser de la extensión PDF, XLS, DOC y extensiones de imágenes. En caso uno de los requisitos no tenga un archivo adjunto le debe enviar un mensaje indicando que el requisito no se grabará si no completa la completa solicitada.

Para que una funcionalidad avance en su proceso es necesario que el portal cuente con un flujo de trabajo que vaya cambiando el estado de la solicitud. Y esta funcionalidad debe ser visible para que el interesado, gestor comercial o supervisor técnico para que al revisarla puedan hacer avanzar en su proceso. También se debe de tener en cuenta que a partir del estado enviado hacia adelante cuando una solicitud cambie de estado le debe llegar un correo al Gestor Comercial o Supervisor Técnico con base a la información que se llena en la solicitud del Usuario Interno a la que se está derivando. Cuando un usuario interno esté atendiendo la solicitud le debe permitir anexar los documentos o las cartas de respuesta una sección donde dejen un breve comentario de los archivos que están adjuntando, este comentario debe de viajar por correo al usuario interno o externo (interesado), también, se debe tener en cuenta que cuando la solicitud pase a estado Rechazado o Aprobado le debe enviar un mensaje al Interesado para que ingrese al portal y verifique la solicitud del expediente asociado.

Los estados por lo que pasa una solicitud son:

Tabla 25 Estado de Solicitud: GTI-PYO-0010

Estado	**Actividades que lo origina**
Enviado	Cuando el Interesado activa la funcionalidad de enviar al Gestor Comercial para su Aprobación o Rechazo
Revisión Comercial	Cuando el Gestor Comercial recibe la solicitud y procede con la revisión. Esta solicitud puede pasar a estado Rechazado si así lo decide. De estar todo conforme pasa al Estado de Revisión Técnica. Y cuando regresa de este estado puede pasar a estado Aprobado o Rechazado
Revisión Técnica	Cuando el Supervisor Técnico revisa la Solicitud. Esta puede pasar a estado Asistente Gerencia Técnica o derivárselo a otro Supervisor Técnico
Aprobado	Cuando el Gestor Comercial da por Aprobado la Solicitud.
Rechazado	Cuando el Gestor Comercial da por Rechazada la solicitud
Revisión de Firmas	Cuando el Gestor Comercial gestiona la firma del Gerente Comercial para poder dar una respuesta válida al interesado
Asistente Gerencia Técnica	Cuando el Revisor Técnico deriva con documentos adjuntos y el asistente de gerencia técnica los revise y los apruebe, puede pasar al Gestor Comercial, Contribución Reembolsable o regresarlo a Revisión Técnica

Los estados de las solicitudes impactan en el estado de los expedientes con base al tipo de solicitud o etapa del proyecto y obra, por lo tanto, debemos de considerar lo siguiente:

Tabla 26 Relación de estado de Solicitud con estado de Expediente: GTI-PYO-0010

Tipo Solicitud	**Estado de Solicitud**	**Estado Expediente**	**Actividades que lo origina**
Aprobación de Proyecto	Aprobado	Aprobado	Cuando la solicitud de la etapa de aprobación de proyecto es Aprobada cambia el estado del Expediente a Aprobado.
Inicio de Obra	Aprobado	Desarrollado	Cuando la solicitud de la etapa de inicio de obra del Proyecto ha sido Aprobada el Estado del Expediente pasa a Desarrollada.

Conformidad o Aceptación de la Obra	Aprobado	Cerrado	Cuando la solicitud de la etapa de conformidad de obra ha sido Aprobado el estado del Expediente pasa a Cerrado.

Cuando se genere una solicitud debe tener la siguiente máscara para el código: SOL-YYYY-NNNNNN, donde:

- SOL: valor fijo y corresponde a las siglas de Solicitud.
- YYYY: año en que la solicitud ha sido creada
- NNNNNN: correlativo numérico que se debe incrementar cada vez que se cree una solicitud y se debe reiniciar cuando empieza el año.

Un interesado debe seleccionar una solicitud que le corresponda tramitar y esto dependerá del Tipo de Proyecto que ha elegido en el expediente, este tipo de proyecto está relacionada a las etapas o tipos de solicitudes que se deben de presentar. Entonces al interesado le debe mostrar solo el tipo de solicitud que le corresponde presentar, de tal forma que tenga un orden en la presentación de las solicitudes. Solo puede pasar a la siguiente solicitud siempre y cuando la anterior se encuentra en estado Aprobado y no le debe permitir elegirla. En caso una Etapa o Tipo de Solicitud haya sido rechazada debe permitirle generar nuevamente la misma solicitud tantas veces se rechace.

Para la funcionalidad GTI-PYO-0011 “Funcionalidad para que el personal de Electro Dunas pueda revisar expedientes” solo comprende que el usuario interno pueda tener una vista en la que se liste todos los expedientes creadas y poder revisar su detalle y obtener los datos del proyectista, de esta manera le va a permitir identificar qué proyectos u expedientes no cuentan con uno y de esta manera se pueda convertir en un cliente para el servicio de asesoría para proyectos y obras para Electro Dunas.

Para la funcionalidad GTI-PYO-0012 “Funcionalidad para que el personal de Electro Dunas pueda revisar solicitudes y posteriormente derivarlo para que pueda ser atendido” hace referencia cuando un usuario interno revisa las solicitudes al ingresar debe visualizar todas las solicitudes con indicador tipo semáforo basado en el campo KPI de la solicitud, este indicativo solo debe ser de uso del usuario interno y no para el usuario externo.

Este valor debe llenarse de la siguiente manera:

Días KPI = Fecha Atención – (Fecha Solicitud + Plazo de Atención)

- Días KPI < 0, indica que la solicitud está fuera de plazo de atención y debe mostrar un ícono de color rojo, este color representa en el portal que está retrasada la atención.
- Días KPI = 0, indica que estamos en el último día de plazo de atención, debe de mostrar un ícono de color amarillo, este color representa en el portal que el tiempo está ajustado.
- Días KPI > 0, indica que la solicitud está dentro del plazo de atención y debe de mostrar un ícono de color ver, este color representa en el portal que está dentro del plazo de atención.

La información que se debe mostrar para una solicitud es:

Tabla 27 Información a mostrar por cada Solicitud a atender: GTI-PYO-0012

Campo	Descripción	Editable (SI/NO)	Tipo Dato	Tamaño	Obligatorio (SI / NO)
Código	Código de la Solicitud (Autogenerado: SOL-YYYY - NNNNNN)	NO	Cadena	25	SI
Asunto	Breve descripción de los que están solicitando	SI	Cadena	250	SI
Fecha Solicitud	Fecha del día que se creó la Solicitud	NO	Fecha		SI
Plazo de Atención	Número de días para la atención y es el dato que trae del tipo de solicitud o etapa del Proyecto.	NO	Numérico		SI
Código de Expediente	Código del Expediente que se ha seleccionado para generar las solicitudes	NO	Cadena	25	SI
Tipo de Solicitud o	Debe elegir solo el tipo de Proyecto que le	SI	Cadena	25	SI

Etapa de Proyecto	muestra para ese Proyecto.				
Tipo de Sistema Eléctrico	Al elegir el tipo de Proyecto viene relacionado al Tipo de Solicitud.	SI	Cadena	25	SI
Comentarios	Breves Comentarios que deben de colocar el Usuario Interno	SI	Cadena	8000	SI
KPI	Indicador que la Solicitud esta fuera de Plazo de Atención	NO	Numérico		SI
Estado	Estado de la Solicitud con base al Flujo de Estados.				

La información que debe mostrar por cada archivo adjunto en cada tipo de requisito relacionado al tipo de solicitud o etapa del proyecto:

Tabla 28 Información a mostrar por cada archivo adjunto de requisitos de una solicitud: GTI-PYO-0012

Campo	Descripción	Editable (SI/NO)	Tipo Dato	Tamaño	Obligatorio (SI / NO)
Correlativo	Secuencia de Registro a generar	NO	Numérico		SI
Contenido	Grabar el contenido del archivo	NO			SI
Estado	Estado Activo / Inactivo. Por defecto debe ser activo	NO	Cadena	1	SI

Cuando una solicitud pase de un estado a otro, debe almacenarse estos cambios como históricos y debe contar con la siguiente información:

Tabla 29 Información a guardar en el cambio de estado de una solicitud: GTI-PYO-0012

Campo	Descripción	Editable (SI/NO)	Tipo Dato	Tamaño	Obligatorio (SI / NO)
Código	Código de la Solicitud (Autogenerado: SOL-YYYY - NNNNNN)	NO	Cadena	25	SI
Correlativo	Secuencia de Registro a generar	NO	Numérico		SI
Estado	Estado que tenía antes del cambio	NO	Cadena	25	SI
Usuario Creación	Código del usuario que genero la transacción	NO	Cadena	10	SI
Fecha de Creación	Fecha que se generó la transacción	NO	Fecha		SI
Usuario Última Modificación	Código del usuario que realizó la última modificación de la transacción	NO	Cadena	10	SI
Fecha de Ultima Modificación	Fecha que se realizó la última modificación a la transacción.	NO	Fecha		SI

En la última funcionalidad GTI-PYO-0013 "Funcionalidad para que el personal de Electro Dunas pueda generar los reportes" le debe permitir al colaborador exportar la información que se está registrando en el portal para una posterior toma de decisiones, estos reportes son:

- Reporte para la gestión comercial
- Reportes de Número de Expedientes Registrados
- Reporte de Número de Solicitudes Registradas

Reporte para la Gestión Comercial

Este reporte tiene como finalidad listar todos lo expedientes y solicitudes en un rango de fechas.

Los filtros que se deben de considerar en este reporte son:

- Rango de fechas compuesto por dos campos: fecha desde y fecha hasta cuya condición en el reporte sea la fecha inicio del expediente.
- Poder seleccionar más de un estado del expediente que se desee listar.
- Poder seleccionar más de un estado de la solicitud que se desee listar.

Los campos que se deben de mostrar en el reporte son:

Tabla 30 Información a mostrar en el Reporte de Gestión Comercial: GTI-PYO-0013

Campo	Descripción	Tipo Dato	Tamaño
N° Expediente	Código de Expediente (Autogenerado: EXP-YYYY - NNNNNN)	Cadena	25
Descripción del Proyecto	Descripción del Proyecto y/o Obra a presentar	Cadena	250
Máxima Demanda del Proyecto (KW)	Máxima Demanda que gestiona el Interesado	Entero	18
Fecha Inicio Expediente	Fecha Inicio del Expediente. Día en que lo creó.	Fecha	
Estado de Expediente	Estado del Expediente con base al Flujo de Estados.	Cadena	25
N° de Solicitud	Código de la Solicitud (Autogenerado: SOL-YYYY - NNNNNN)	Cadena	25
Asunto Solicitud	Breve descripción de los que están solicitando	Cadena	250
Fecha Ultimo Estado (workflow)	Fecha del Ultimo Estado de la Solicitud	Fecha	
Estado de Solicitud (workflow)	Estado de la Solicitud con base al Flujo de Estados.	Cadena	25

El diseño del reporte será el siguiente:

Figura 7 Diseño de Reporte de Gestión Comercial

Reporte de Número de Expedientes Registrados

Este reporte tiene como finalidad listar la cantidad de expedientes registrados por departamento y provincia de la concesión de Electro Dunas.

Los filtros que se deben de considerar en este reporte son:

- Filtrar por año de la fecha de inicio del expediente.
- Poder seleccionar el departamento registrado en el expediente a través del campo ubigeo.
- Poder seleccionar la provincia registrado en el expediente a través del campo ubigeo.

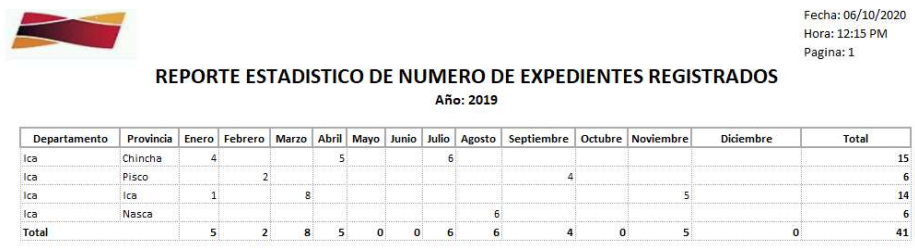
Los campos que se deben de mostrar en el reporte son:

Tabla 31 Información a mostrar en el Reporte de Expedientes Registrados: GTI-PYO-0013

Campo	Descripción	Tipo Dato	Tamaño
Departamento	Descripción del Ubigeo. Se toma los dos primeros dígitos del campo Ubicación geográfica (posición 1 y 2 comenzando por la izquierda) y se extrae la descripción de la tabla del maestro de Ubigeos	Cadena	250
Provincia	Descripción del Ubigeo. Se toma los siguientes dos primeros dígitos del campo Ubicación geográfica (posición 3 y 4 comenzando por la izquierda) y se extrae la descripción de la tabla del maestro de Ubigeos	Cadena	250
Enero	Cantidad de Expedientes registrados en el mes y año de la fecha inicio del Expediente	Entero	18
Febrero	Cantidad de Expedientes registrados en el mes y año de la fecha inicio del Expediente	Entero	18
Marzo	Cantidad de Expedientes registrados en el mes y año de la fecha inicio del Expediente	Entero	18
Abril	Cantidad de Expedientes registrados en el mes y año de la fecha inicio del Expediente	Entero	18

Mayo	Cantidad de Expedientes registrados en el mes y año de la fecha inicio del Expediente	Entero	18
Junio	Cantidad de Expedientes registrados en el mes y año de la fecha inicio del Expediente	Entero	18
Julio	Cantidad de Expedientes registrados en el mes y año de la fecha inicio del Expediente	Entero	18
Agosto	Cantidad de Expedientes registrados en el mes y año de la fecha inicio del Expediente	Entero	18
Septiembre	Cantidad de Expedientes registrados en el mes y año de la fecha inicio del Expediente	Entero	18
Octubre	Cantidad de Expedientes registrados en el mes y año de la fecha inicio del Expediente	Entero	18
Noviembre	Cantidad de Expedientes registrados en el mes y año de la fecha inicio del Expediente	Entero	18
Diciembre	Cantidad de Expedientes registrados en el mes y año de la fecha inicio del Expediente	Entero	18
Total, de Columnas	Sumar las cantidades de los expedientes registrados de todos los meses por cada Fila.	Entero	18
Total, de Filas	Sumar en cada una de las columnas de enero a diciembre la cantidad de Expedientes registrados.	Entero	18

El diseño del reporte será el siguiente:



Fecha: 06/10/2020
Hora: 12:15 PM
Pagina: 1

REPORTE ESTADISTICO DE NUMERO DE EXPEDIENTES REGISTRADOS
Año: 2019

Departamento	Provincia	Enero	Febrero	Marzo	Abril	Mayo	Junio	Julio	Agosto	Septiembre	Octubre	Noviembre	Diciembre	Total
Ica	Chincha	4			5			6						15
Ica	Pisco		2							4				6
Ica	Ica	1		8								5		14
Ica	Nasca								6					6
Total		5	2	8	5	0	0	6	6	4	0	5	0	41

Figura 8 Diseño de Reporte de Expedientes Registrados

Reporte de Número de Solicitudes Registradas

Este reporte tiene como finalidad listar la cantidad de solicitudes registrados por tipo de sistema eléctrico y tipo de solicitud.

Los filtros que se deben de considerar en este reporte son:

- Filtrar por año de la fecha de la solicitud
- Poder seleccionar el tipo de sistema eléctrico relacionado a la solicitud.
- Poder seleccionar el tipo de solicitud relacionado a la solicitud.
- Poder seleccionar el departamento registrado en el expediente a través del campo ubigeo.
- Poder seleccionar la provincia registrado en el expediente a través del campo ubigeo.

Los campos que se deben de mostrar en el reporte son:

Tabla 32 Información a mostrar en el Reporte de Solicitudes Registradas: GTI-PYO-0013

Campo	Descripción	Tipo Dato	Tamaño
Tipo de Sistema eléctrico	Descripción del Tipo de Sistema eléctrico. El código del tipo de sistema eléctrico está relacionado a la solicitud.	Cadena	250
Tipo de Solicitud	descripción del Tipo de Solicitud. El código del tipo de solicitud está relacionado a la solicitud.	Cadena	250
Enero	Cantidad de Expedientes registrados en el mes y año de la fecha inicio del Expediente	Entero	18
Febrero	Cantidad de Expedientes registrados en el mes y año de la fecha inicio del Expediente	Entero	18
Marzo	Cantidad de Expedientes registrados en el mes y año de la fecha inicio del Expediente	Entero	18
Abril	Cantidad de Expedientes registrados en el mes y año de la fecha inicio del Expediente	Entero	18
Mayo	Cantidad de Expedientes registrados en el mes y año de la fecha inicio del Expediente	Entero	18

Junio	Cantidad de Expedientes registrados en el mes y año de la fecha inicio del Expediente	Entero	18
Julio	Cantidad de Expedientes registrados en el mes y año de la fecha inicio del Expediente	Entero	18
Agosto	Cantidad de Expedientes registrados en el mes y año de la fecha inicio del Expediente	Entero	18
Septiembre	Cantidad de Expedientes registrados en el mes y año de la fecha inicio del Expediente	Entero	18
Octubre	Cantidad de Expedientes registrados en el mes y año de la fecha inicio del Expediente	Entero	18
Noviembre	Cantidad de Expedientes registrados en el mes y año de la fecha inicio del Expediente	Entero	18
Diciembre	Cantidad de Expedientes registrados en el mes y año de la fecha inicio del Expediente	Entero	18
Total, de Columnas	Sumar las cantidades de las Solicitudes registrados de todos los meses por cada Fila.	Entero	18
Total, de Filas	Sumar en cada una de las columnas de enero a diciembre la cantidad de Solicitudes registrados.	Entero	18

El diseño del reporte será el siguiente:

Fecha: 06/10/2020
Hora: 12:15 PM
Pagina: 1

REPORTE ESTADISTICO DE SOLICITUDES

Departamento: Ica | Provincia: Chincha
Año: 2019

Tipo de Sistema	Tipo Solicitud	Enero	Febrero	Marzo	Abril	Mayo	Junio	Julio	Agosto	Septiembre	Octubre	Noviembre	Diciembre	Total
Sistema de Utilizacion	Factibilidad	1			3			1						5
	Punto Diseño		1	1		1			1					4
	Revision Proyecto	1					1					2		4
Sistema Distribucion	Inicio de Obra								1					1
	Solicitud de Conexión												1	1
Totales		2	1	1	3	1	1	1	2	0	0	2	0	15

Figura 9 Diseño de Reporte de Solicitudes Registradas

Modelado del proceso de los requerimientos

En la elaboración del RNS se utilizó la herramienta de modelado BIZAGI para detallar los diagramas del proceso.

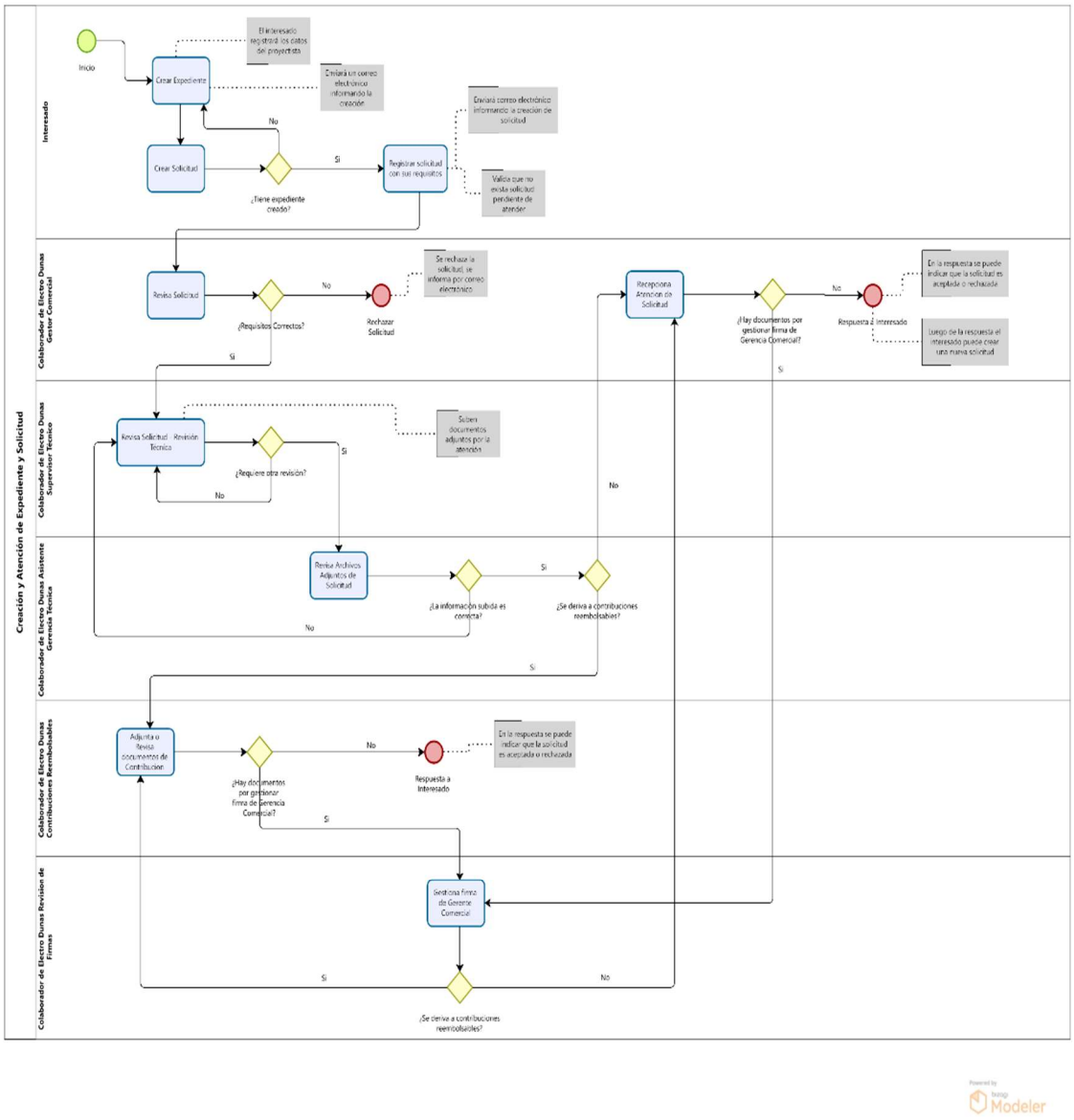


Figura 10 Modelamiento de Procesos

Diseño de los prototipos

Como parte del proceso en el área de desarrollo, se debe de presentar los prototipos al área usuaria antes de iniciar un desarrollo, estos prototipos se presentaron en una sesión remota, se usó la herramienta Balsamiq Mockup.

Los prototipos son organizados por funcionalidad o módulo al que puedan pertenecer:

Como primer prototipo tenemos la ventana login del sistema, donde se encuentra la funcionalidad de poder registrarse como un usuario externo (tercero) en el portal.

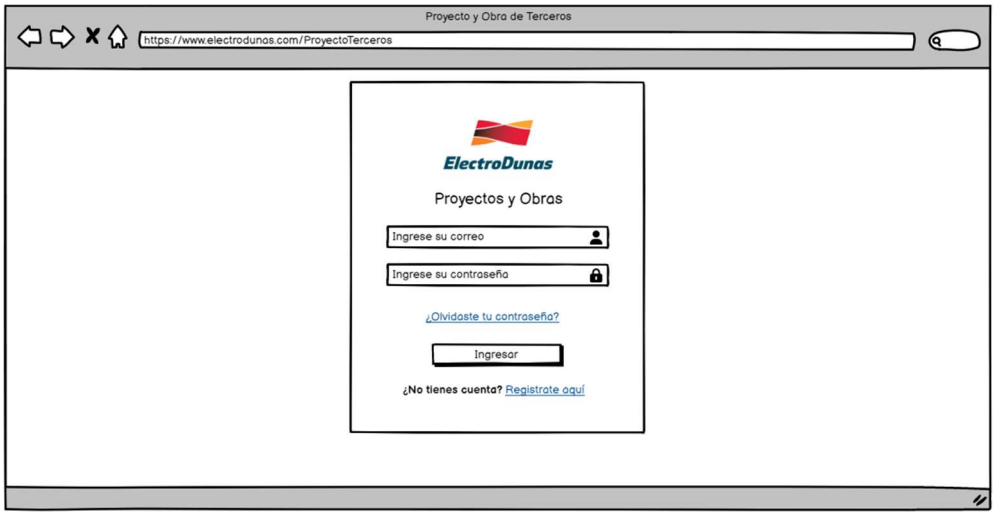


Figura 11 Prototipo de Login del Portal Web

Luego tenemos el prototipo de formulario de registro de un tercero

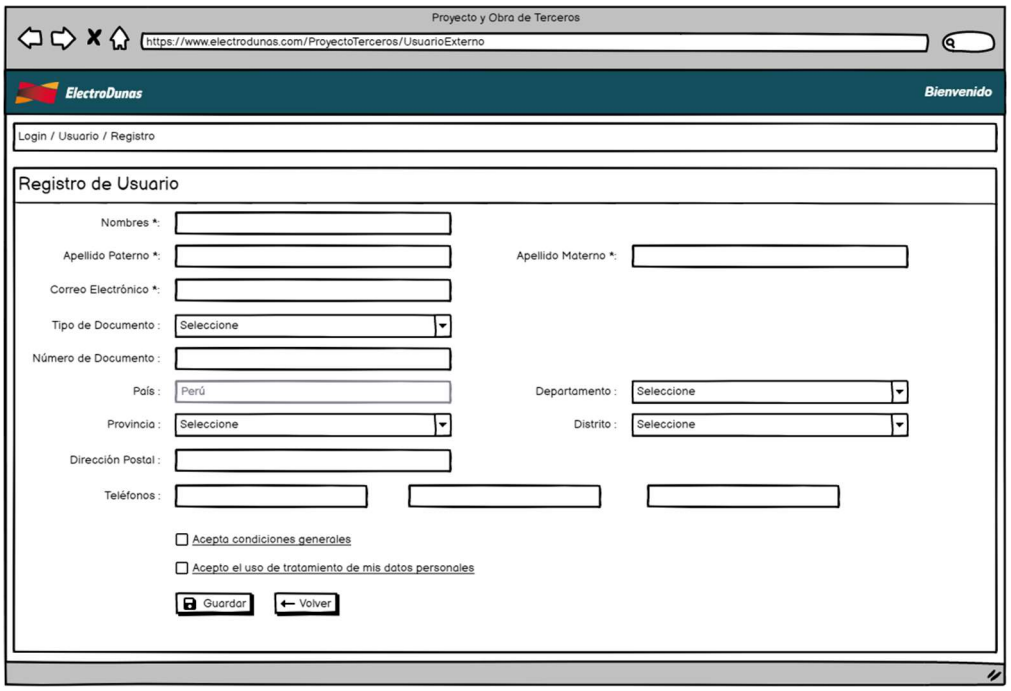


Figura 12 Prototipo de formulario de registro de Usuario Externo (Tercero)

Desde la ventana de login podremos observar la funcionalidad de recuperación de contraseña el cual será activa tanto para el usuario interno (colaborador de Electro Dunas) como el usuario de un tercero, el siguiente prototipo muestra la funcionalidad.

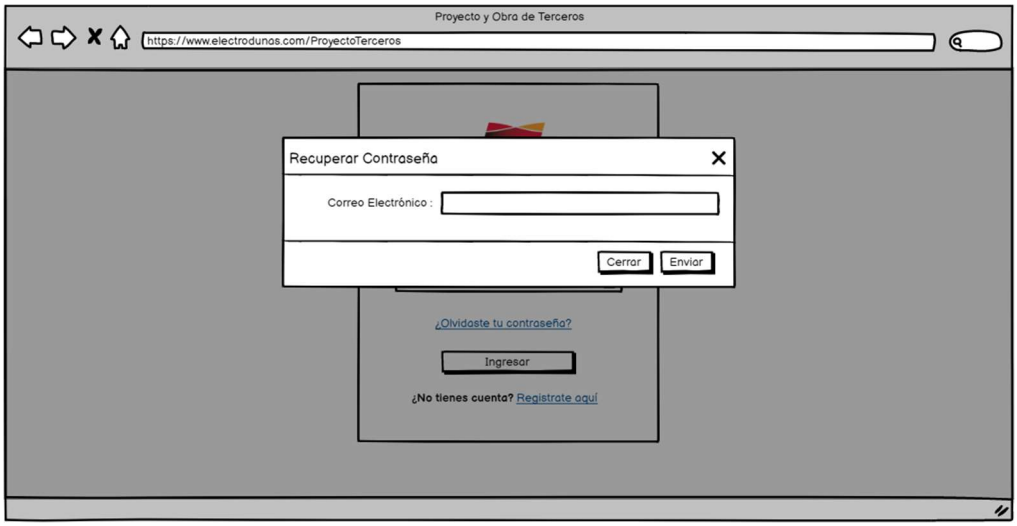


Figura 13 Prototipo de recuperación de contraseña

Luego del primer inicio de sesión, luego de la recuperación la contraseña o cuando se cree el usuario, pedirá la actualización de contraseña, asimismo se usa la misma pantalla cuando el usuario quiera cambiar su contraseña desde una opción en el menú que se le muestre, el siguiente prototipo muestra la funcionalidad detallada.

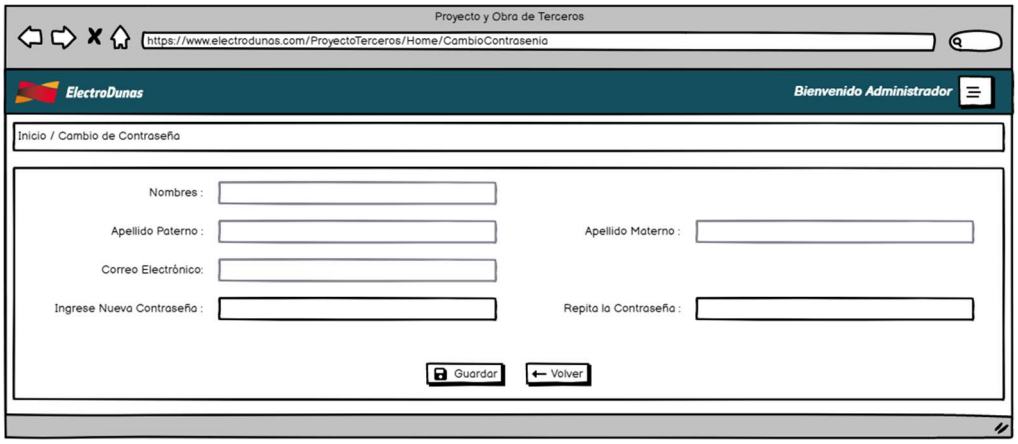


Figura 14 Prototipo de formulario de cambio de contraseña

Cuando el tercero o el colaborador de Electro Dunas ingresa de manera satisfactoria, se mostrará una pantalla de inicio y el listado de opciones del menú al que tenga acceso, serán vistas realizadas como el prototipo lo detalla

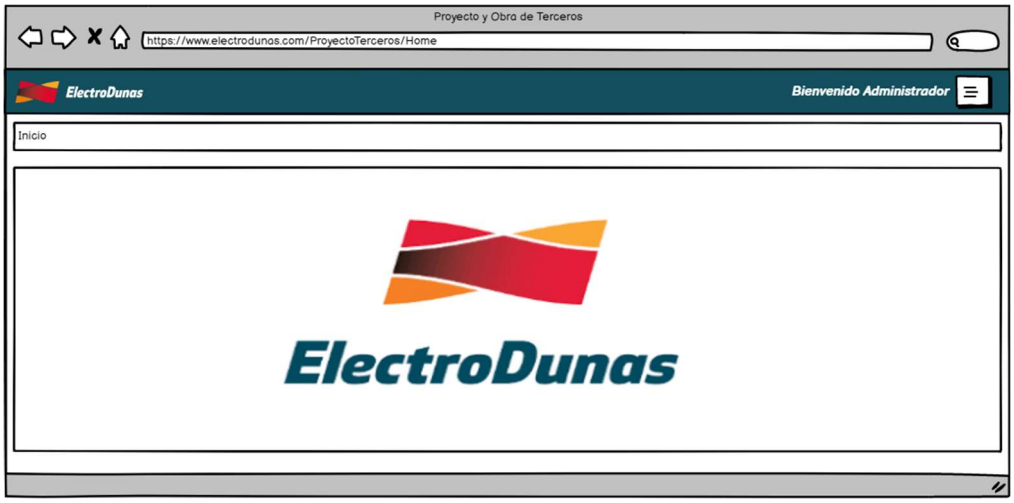


Figura 15 Prototipo de pantalla de inicio

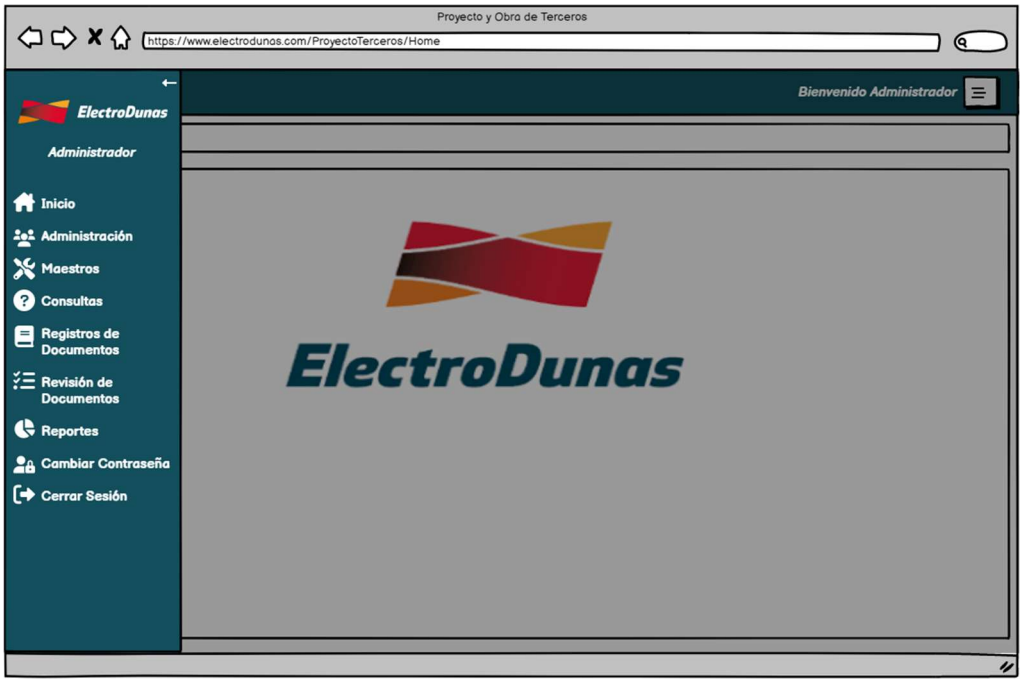


Figura 16 Prototipo del menú del portal web

Para la elaboración de los prototipos siguientes se tomó el orden de módulos que se muestra en la imagen anterior, es por ello que se elaboró el prototipo del módulo Administración.

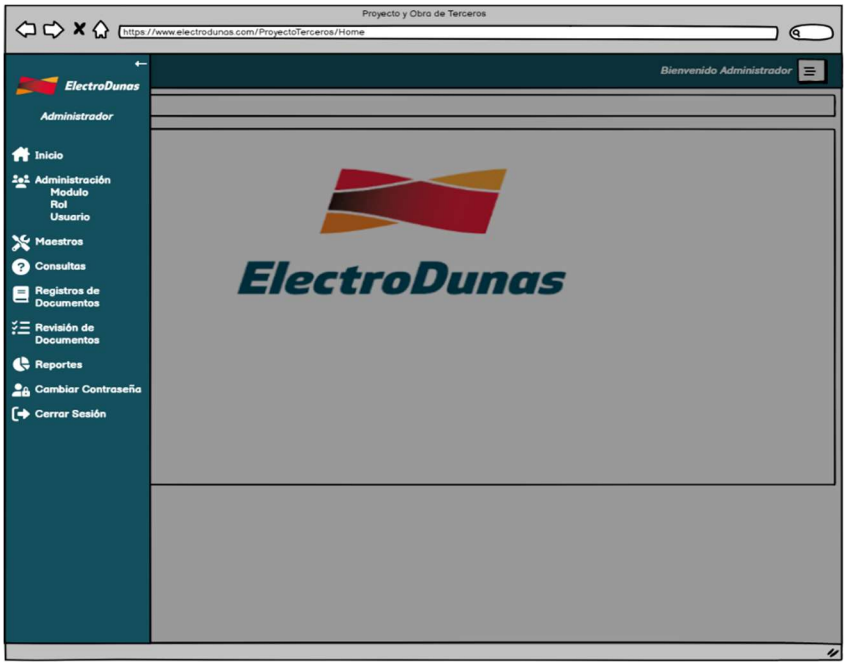


Figura 17 Prototipo del módulo Administración del portal

Como primera opción del módulo se elaboraron los prototipos de la creación, actualización, listado y eliminación de la opción Módulo.

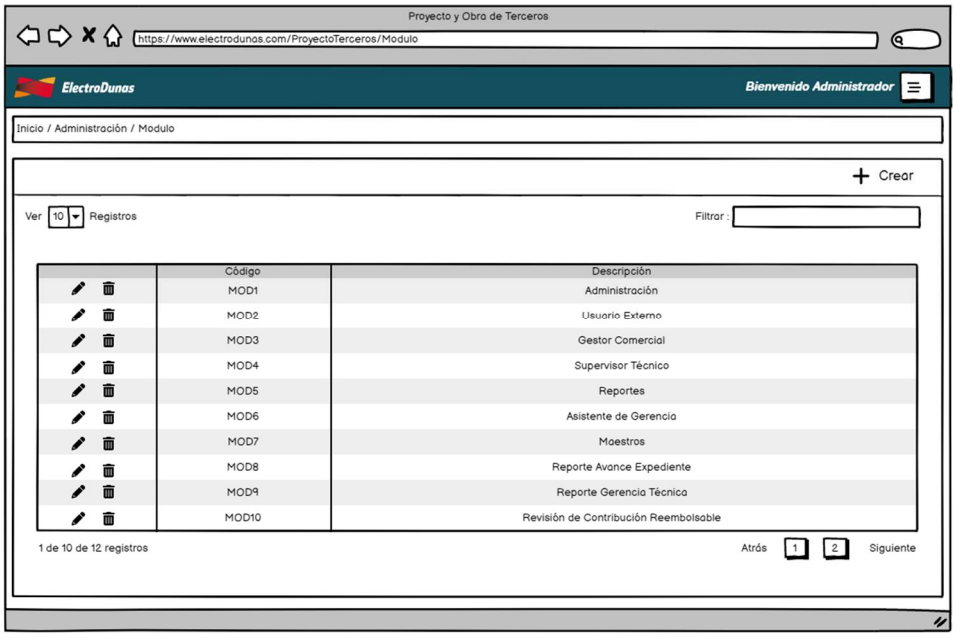


Figura 18 Prototipo de pantalla de listado de Módulos

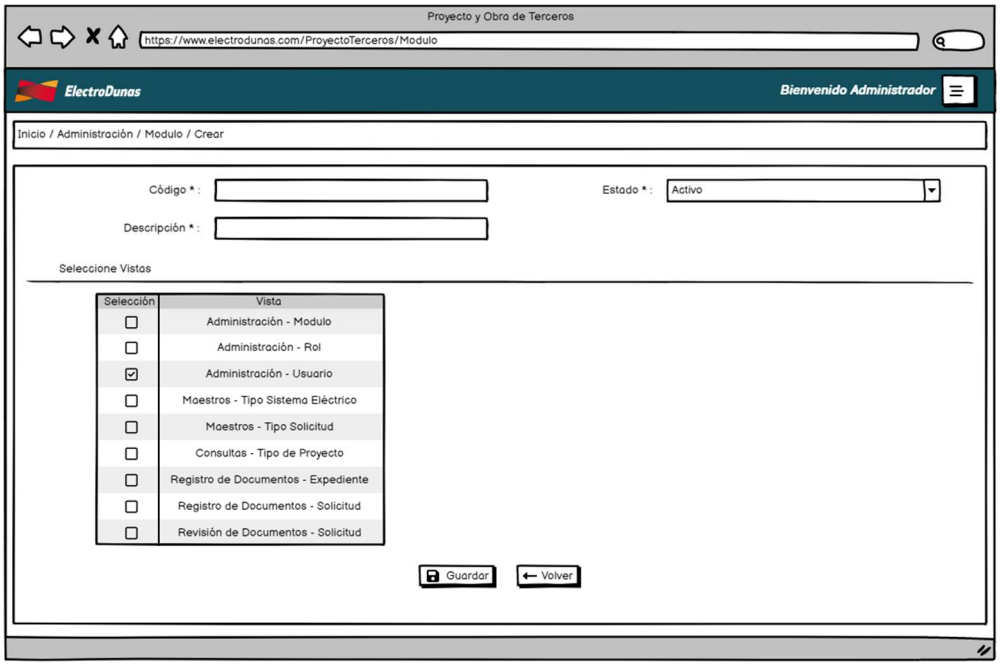


Figura 19 Prototipo de pantalla de creación de módulo

Asimismo, se crearon los prototipos de creación, actualización, listado y eliminación de la opción Rol

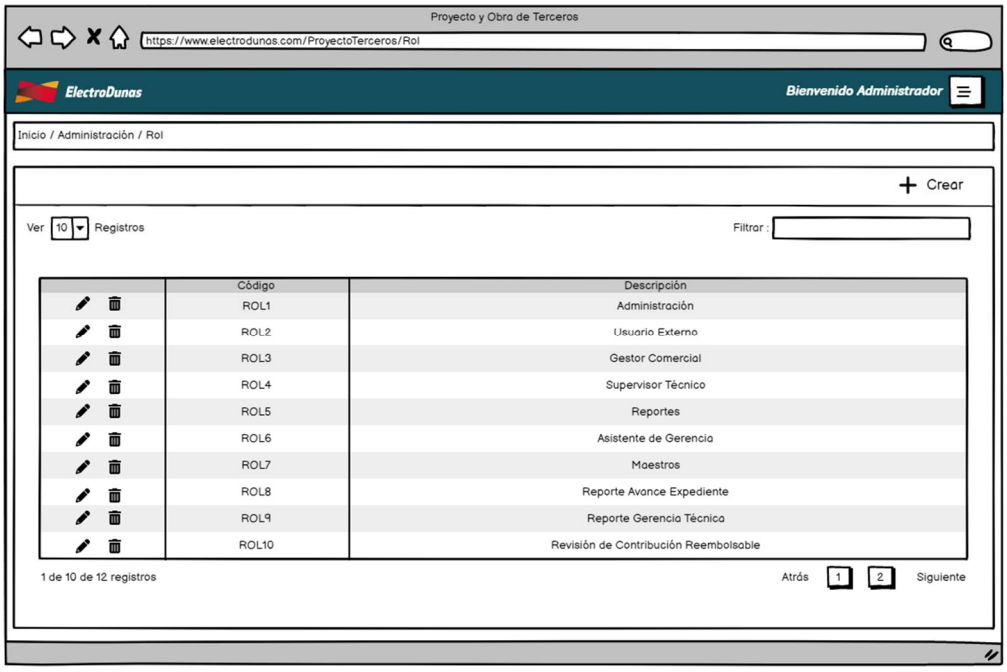


Figura 20 Prototipo de pantalla de listado de Roles

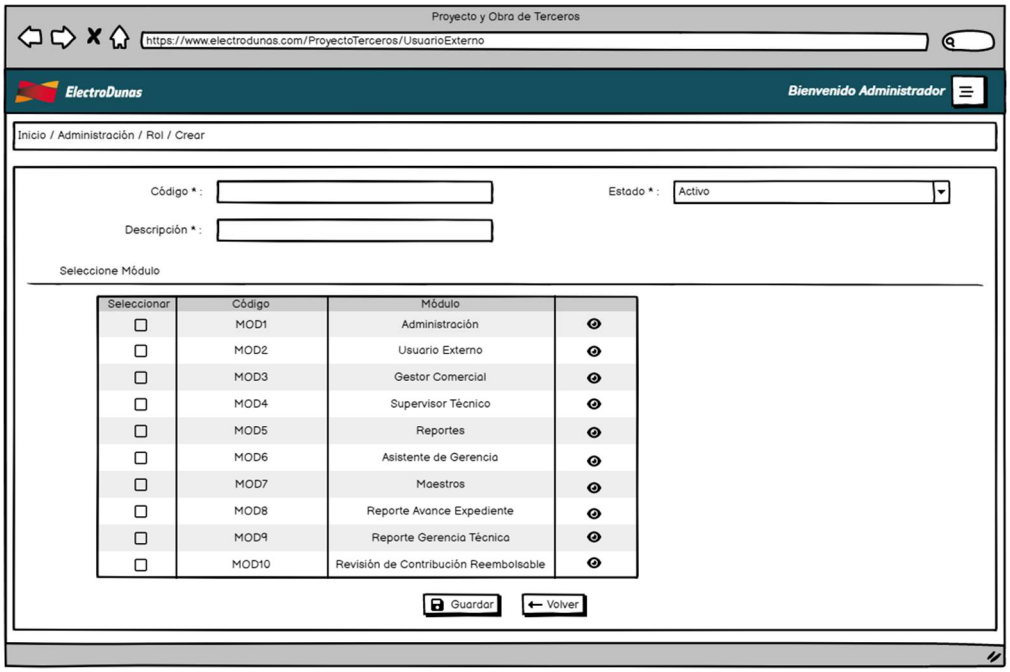


Figura 21 Prototipo de pantalla de creación de Rol

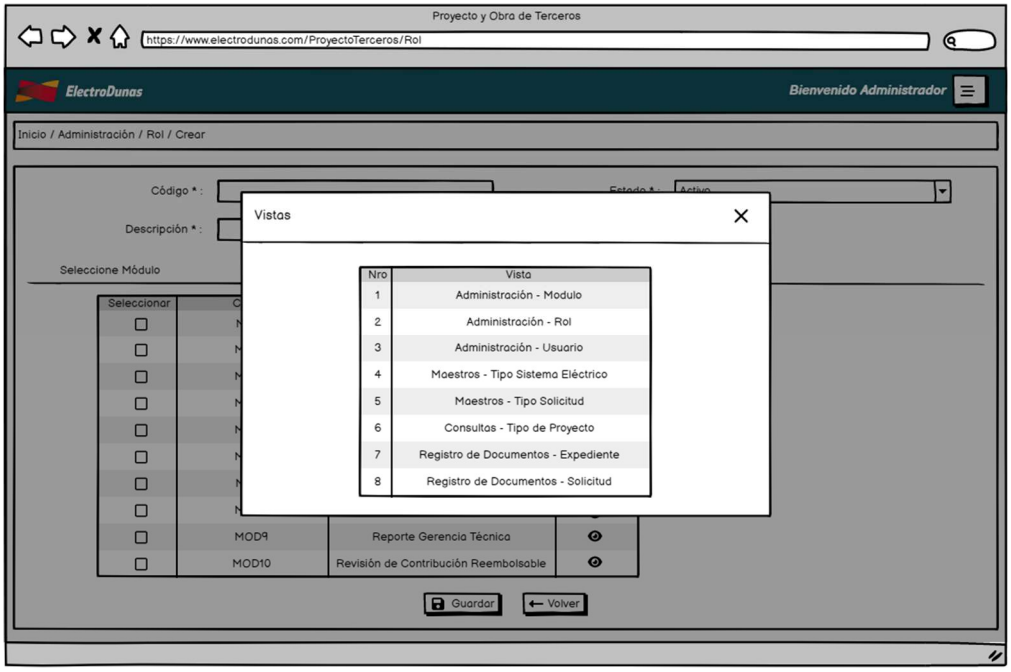


Figura 22 Prototipo de consulta de vistas a mostrar en cada módulo

En este módulo de administración también se ve reflejado los prototipos de la creación, listado, actualización y eliminación de un usuario interno (colaborador de Electro Dunas)

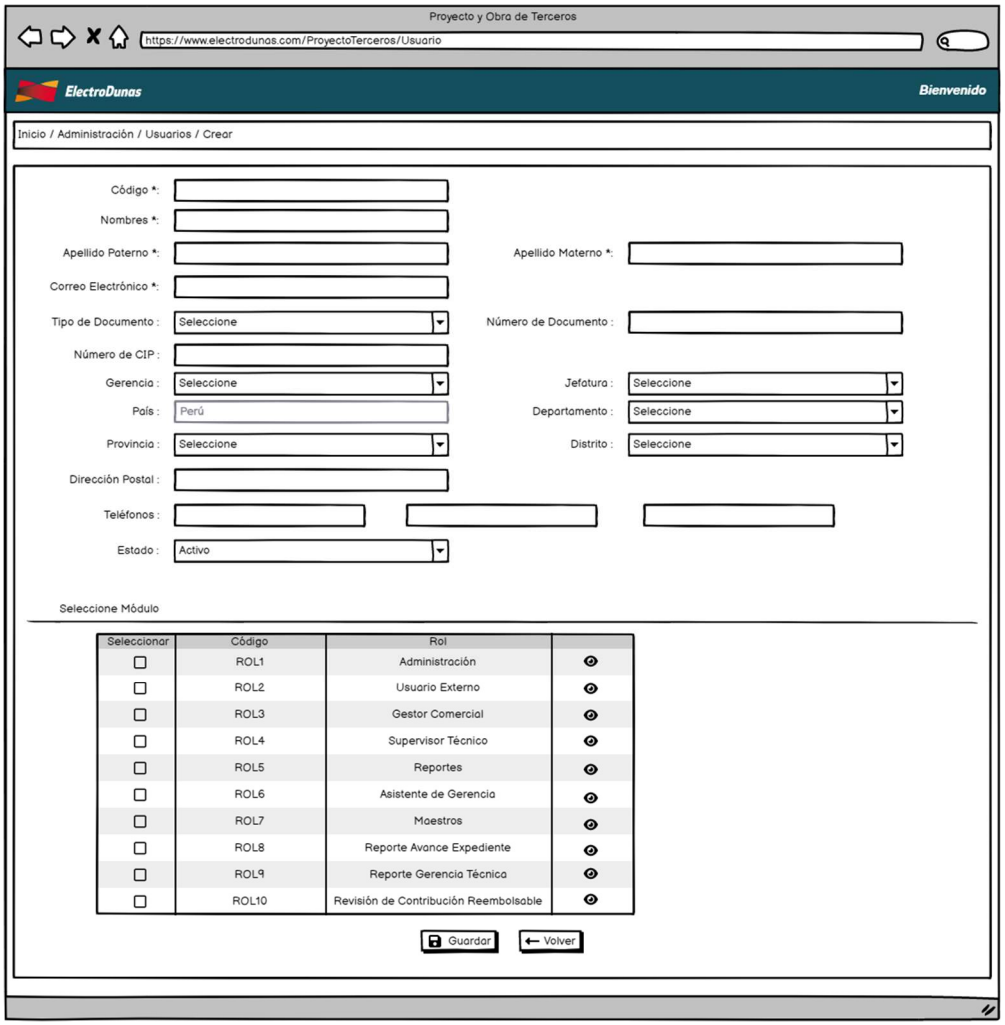


Figura 23 Prototipo de pantalla de creación de Usuario

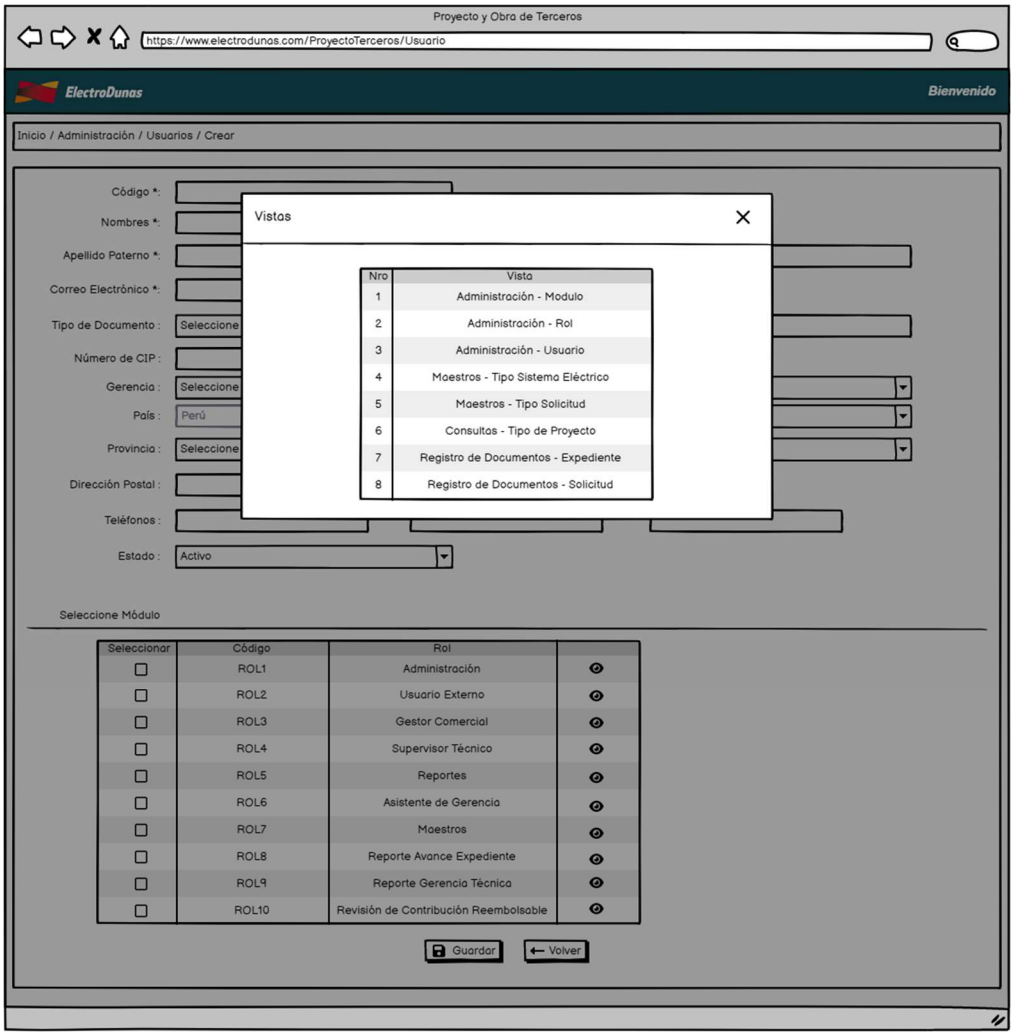


Figura 24 Prototipo de consulta de vistas a mostrar por Rol

Luego de culminar con los prototipos del módulo Administración, continuamos con los prototipos del módulo Maestros.

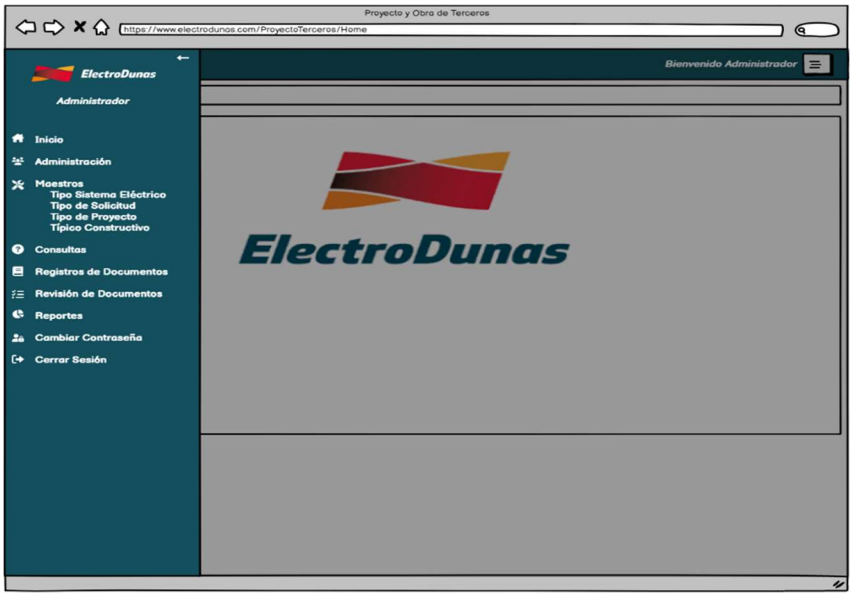


Figura 25 Prototipo del módulo Maestros del portal

En el módulo de Maestro tendremos la opción de la creación, actualización, listado y eliminación del tipo de sistema eléctrico al cual pertenece los siguientes prototipos.

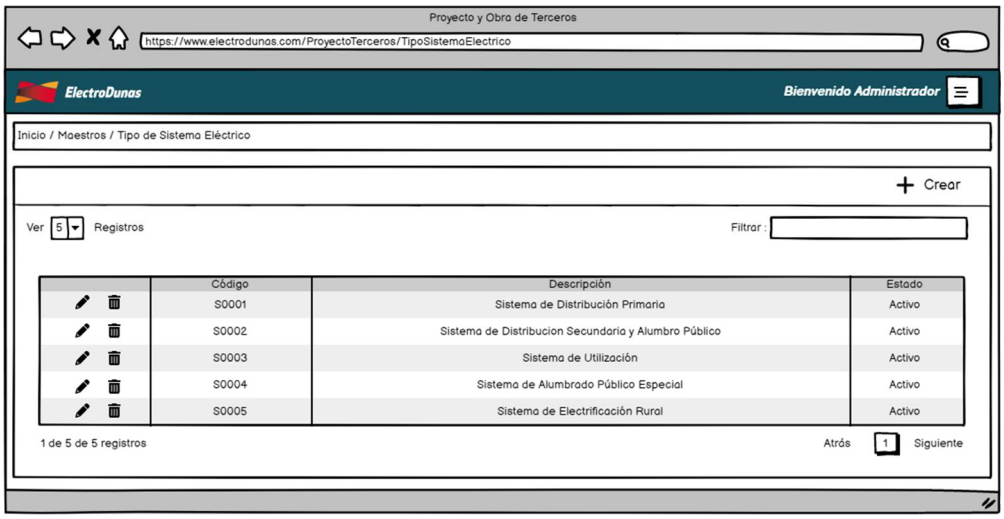


Figura 26 Prototipo de pantalla de listado de Tipo de Sistema Eléctrico

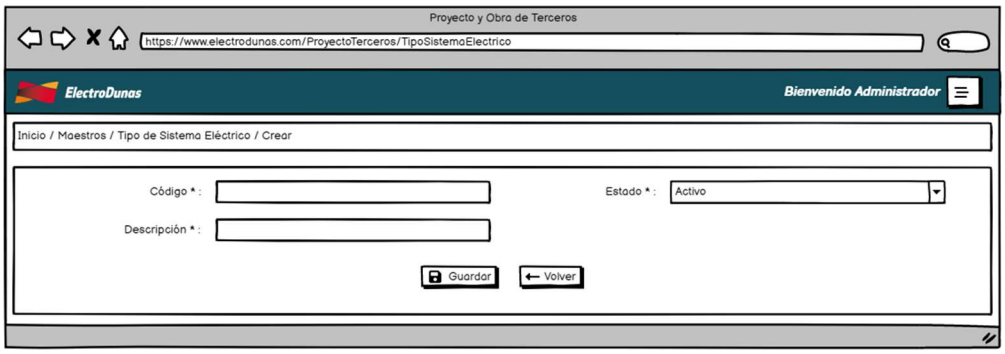


Figura 27 Prototipo de pantalla de creación de Tipo de Sistema Eléctrico

Asimismo, se consideran los prototipos de creación, actualización, listado y eliminación del típico constructivo.

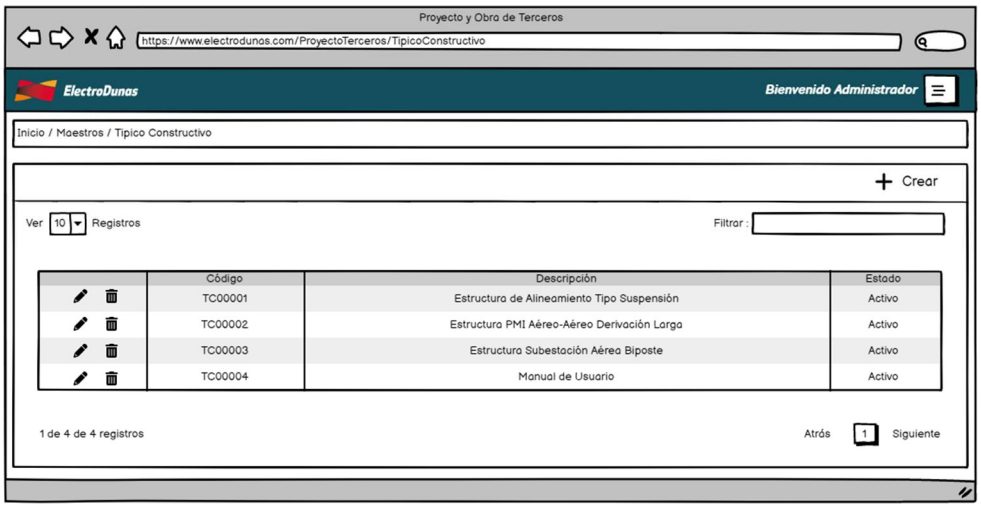


Figura 28 Prototipo de pantalla de listado de Típico Constructivo

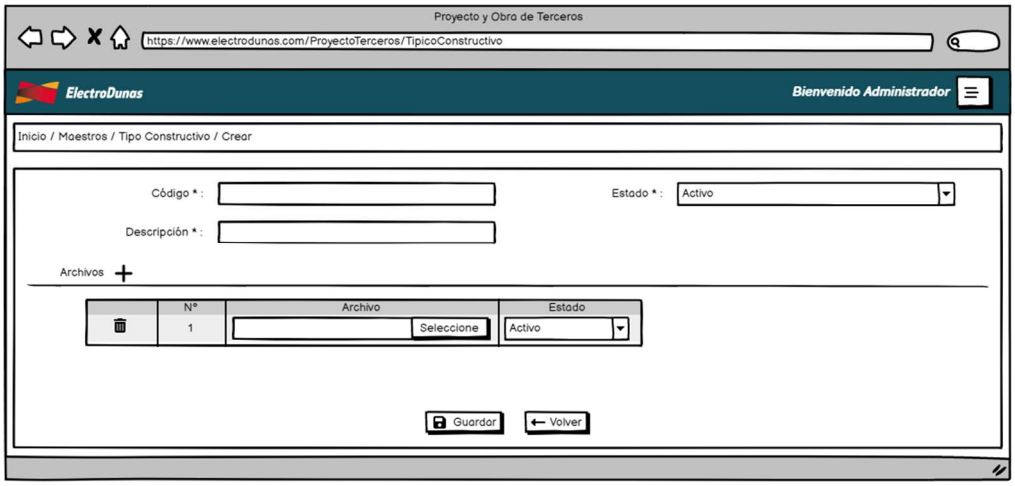


Figura 29 Prototipo de pantalla de creación de Típico Constructivo

A continuación, se tienen los prototipos de creación, listado, actualización y eliminación de los tipos de solicitud.

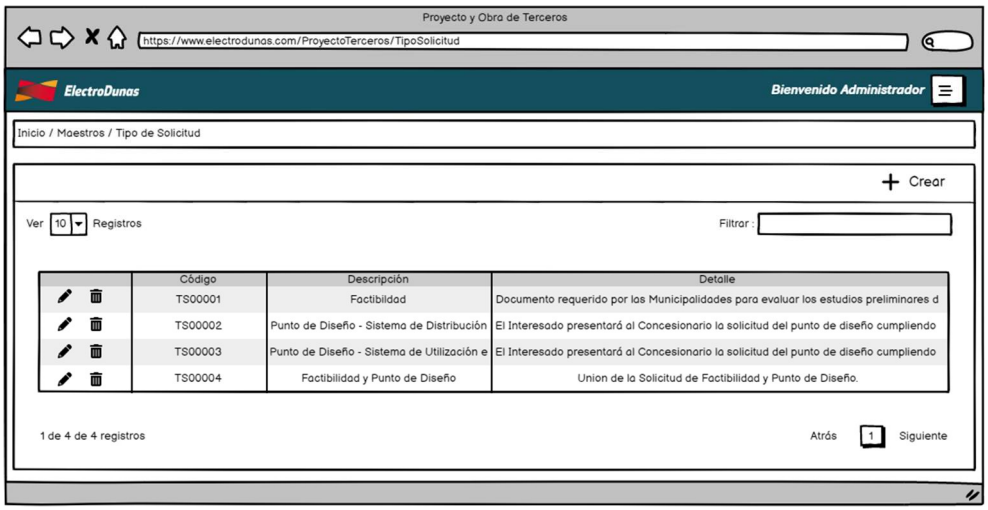


Figura 30 Prototipo de pantalla de listado de Tipo de Solicitud

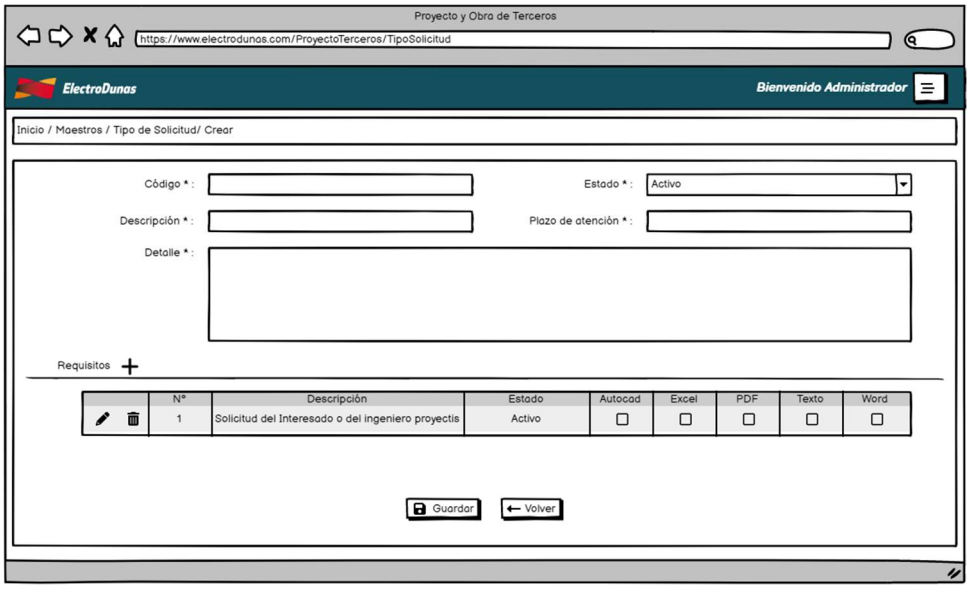


Figura 31 Prototipo de pantalla de creación de Tipo de Solicitud

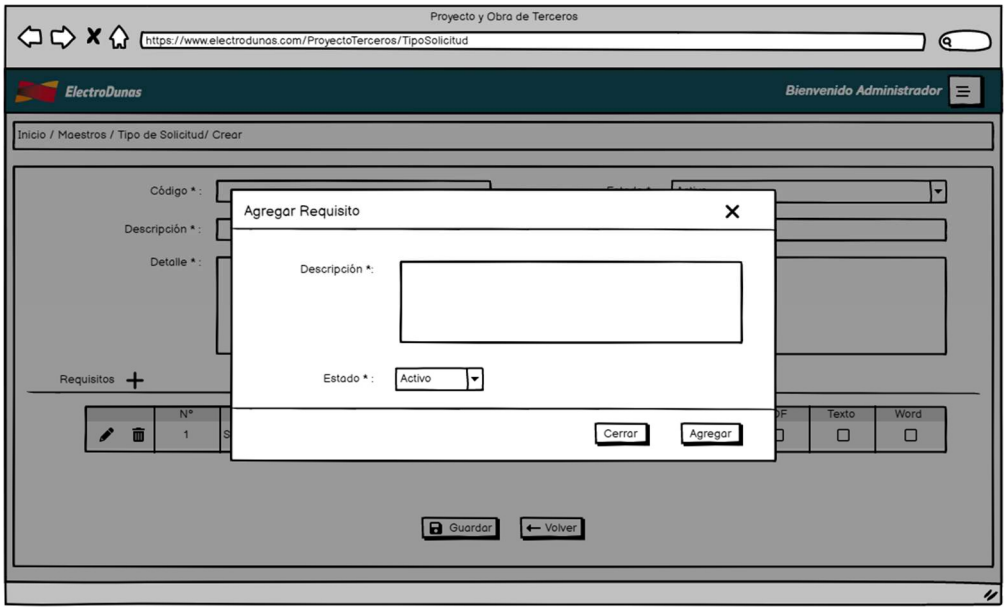


Figura 32 Prototipo de opción para agregar requisito en la creación de un Tipo de Solicitud

Por último, se tienen los prototipos de listado, eliminación, creación y actualización del tipo de proyecto.

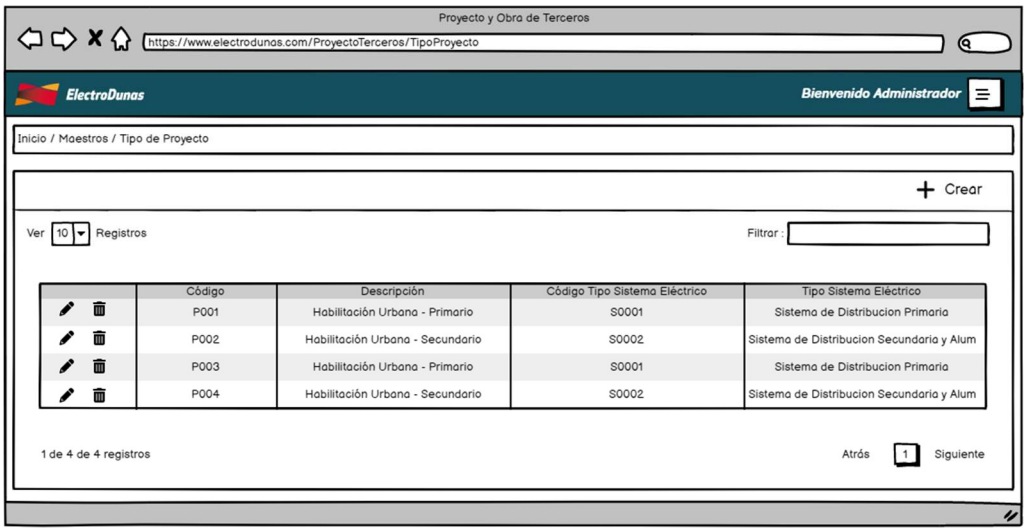


Figura 33 Prototipo de pantalla de listado de Tipo de Proyecto

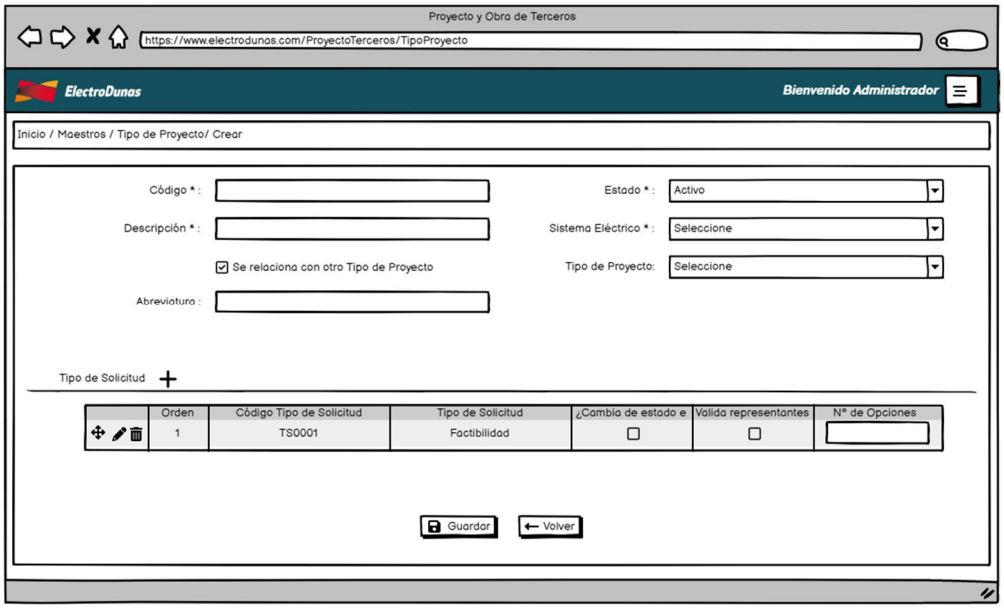


Figura 34 Prototipo de pantalla de creación de Tipo de Proyecto

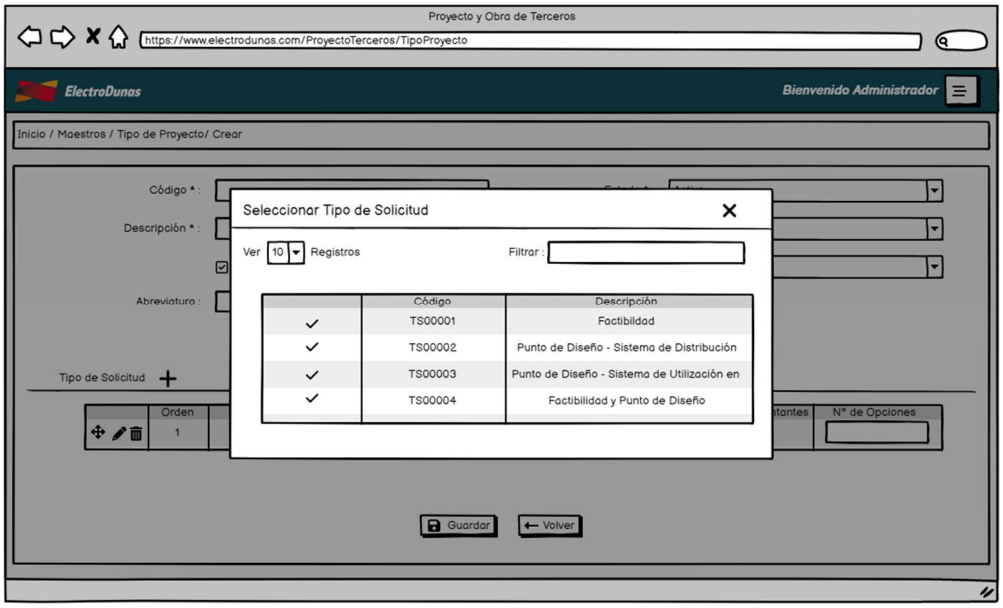


Figura 35 Prototipo de opción para agregar el Tipo de Solicitud en la creación de un Tipo de Proyecto

A continuación, tenemos los prototipos del módulo de Consultas, el cual el tercero puede consultas los requisitos para un típico constructivo y para la presentación de un proyecto determinado, consultando los tipos de solicitudes a presentar como los requisitos en cada uno de ellas.

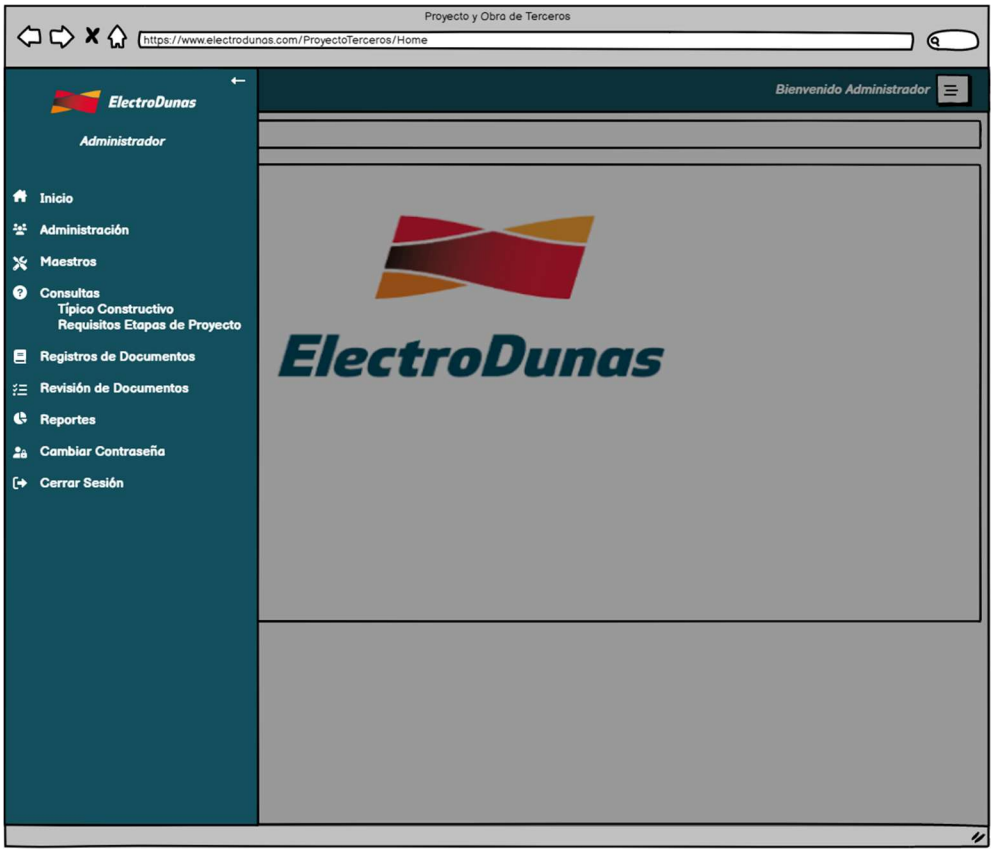


Figura 36 Prototipo del módulo de Consultas del portal

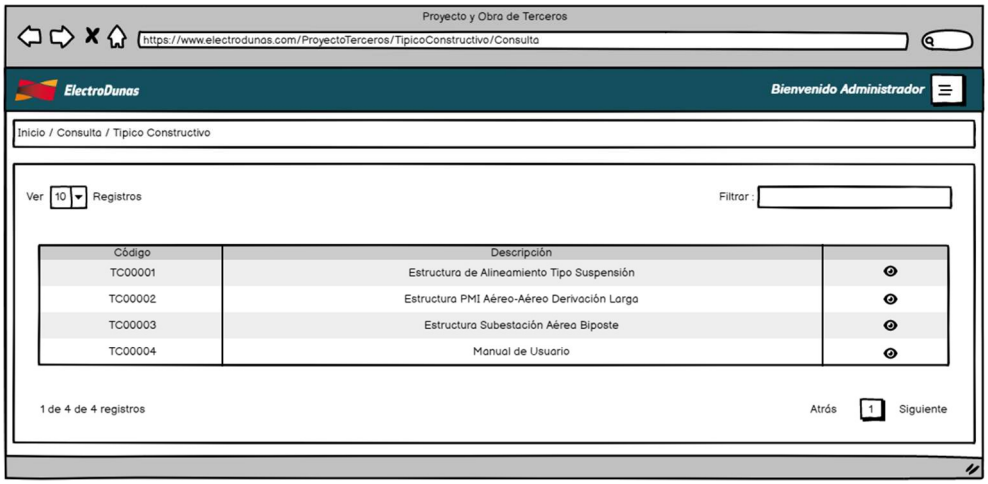


Figura 37 Prototipo de pantalla de consulta de Típico Constructivo

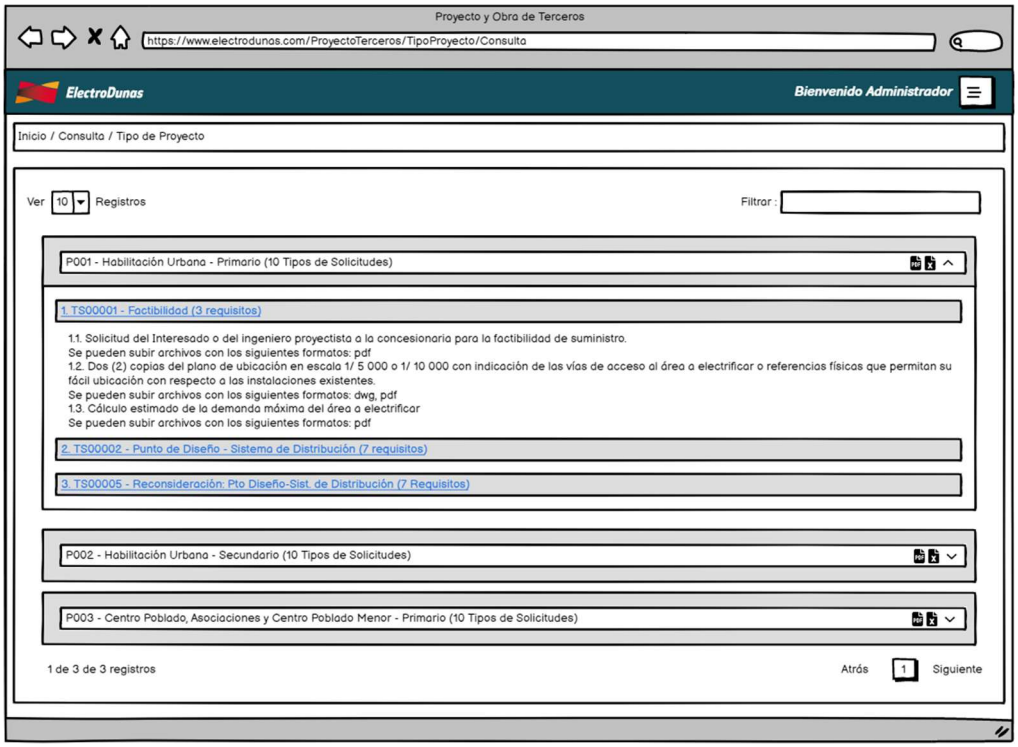


Figura 38 Prototipo de pantalla de consulta de Tipo de Proyecto

El tercero también podrá ingresar al módulo de registro de documentos el cual las opciones se muestran en el siguiente prototipo.

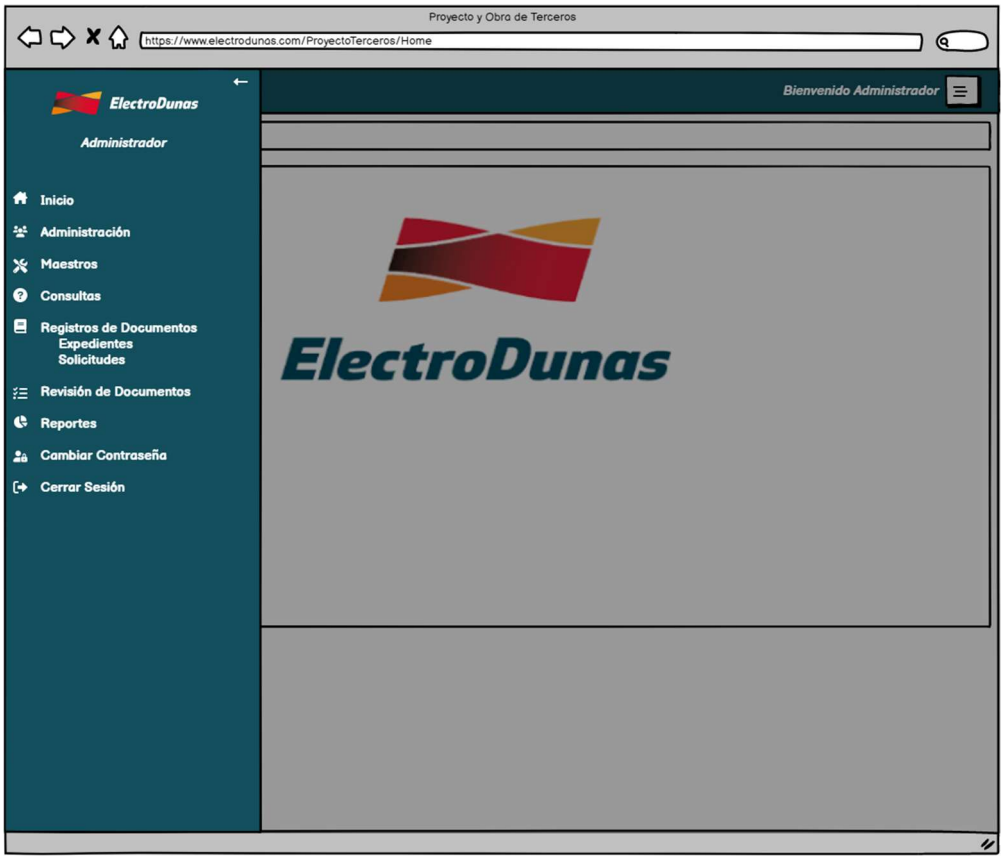


Figura 39 Prototipo de módulo de Registros de Documentos del portal

El tercero contará con una pantalla en la que podrá registrar y dar seguimiento a su expediente creado en el portal, el siguiente prototipo muestra cómo será visible por parte del tercero.

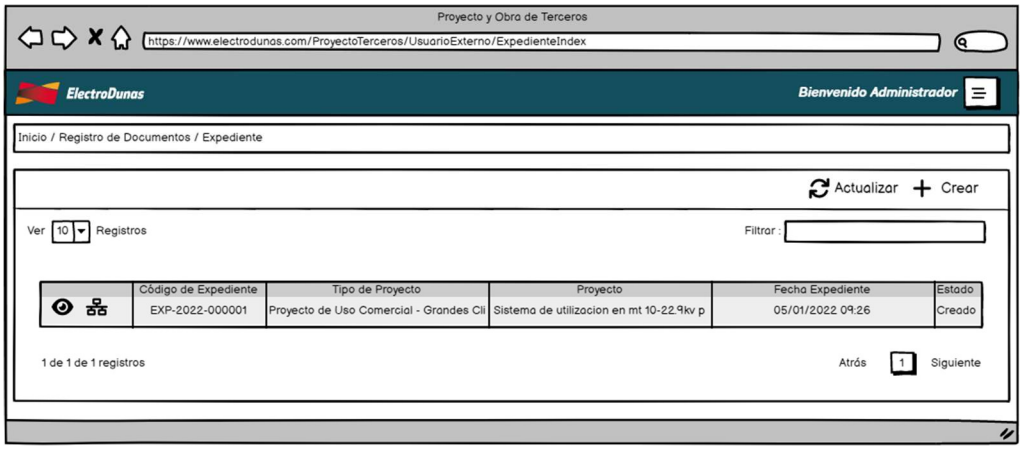


Figura 40 Prototipo de pantalla de listado de Expedientes - Vista del Tercero

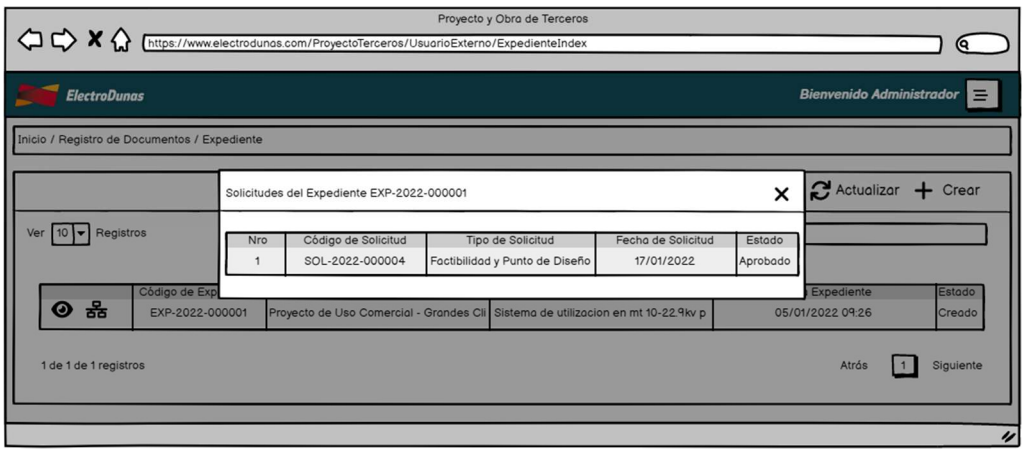


Figura 41 Prototipo de opción de visualización de solicitudes creadas en un Expediente - Vista del Tercero

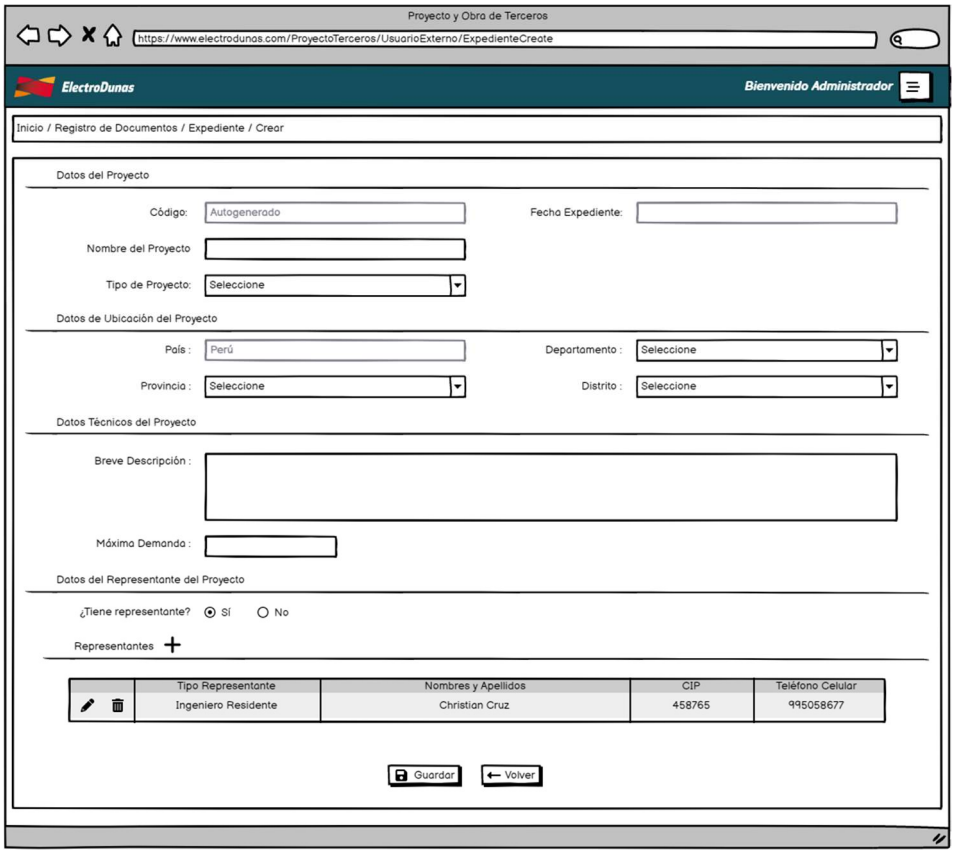


Figura 42 Prototipo de pantalla de creación de Expediente - Vista del Tercero

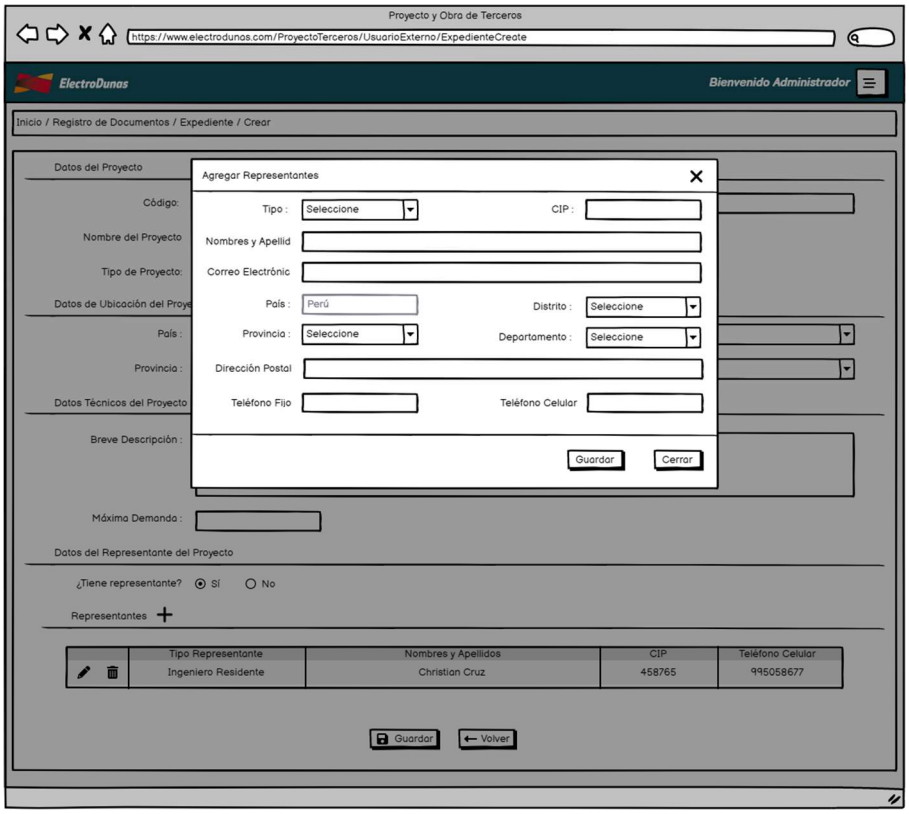


Figura 43 Prototipo de opción de agregar representante en la creación de Expediente - Vista del Tercero

A continuación, se muestra el prototipo de la creación y seguimiento de una solicitud, esta funcionalidad podrá ser usada por el tercero y solo se realizará cuando el sistema identifique que tiene un expediente creado en el sistema.

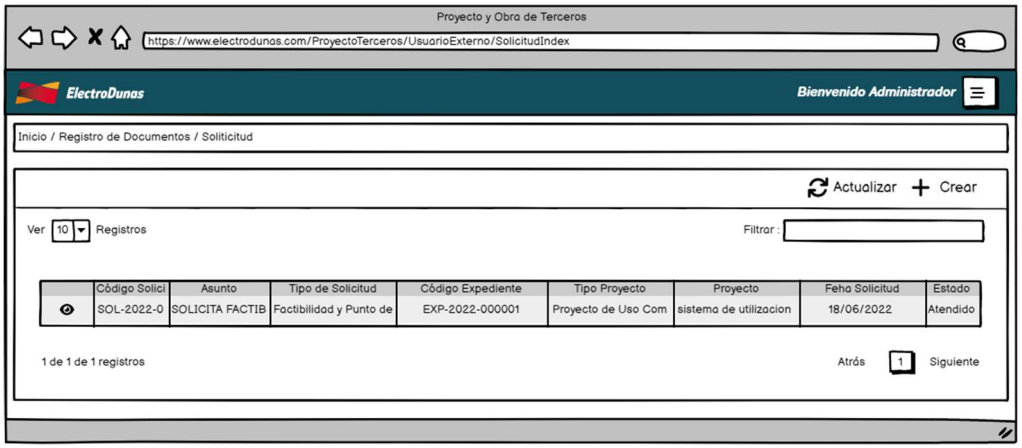


Figura 44 Prototipo de pantalla de listado de Solicitudes - Vista del Tercero

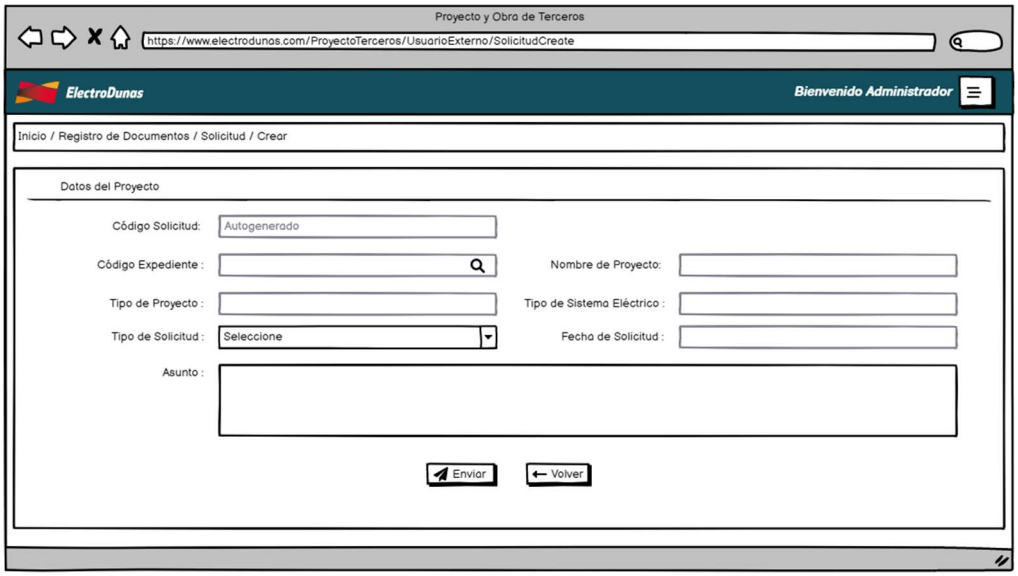


Figura 45 Prototipo de pantalla de creación de Solicitud - Vista del Tercero

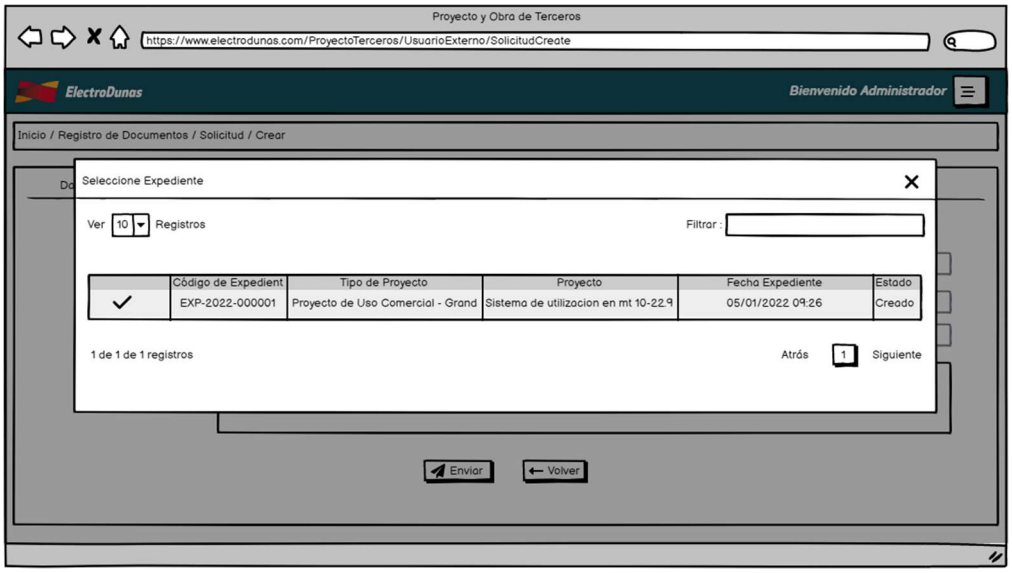


Figura 46 Prototipo de opción de selección de expediente en la creación de Solicitud - Vista del Tercero

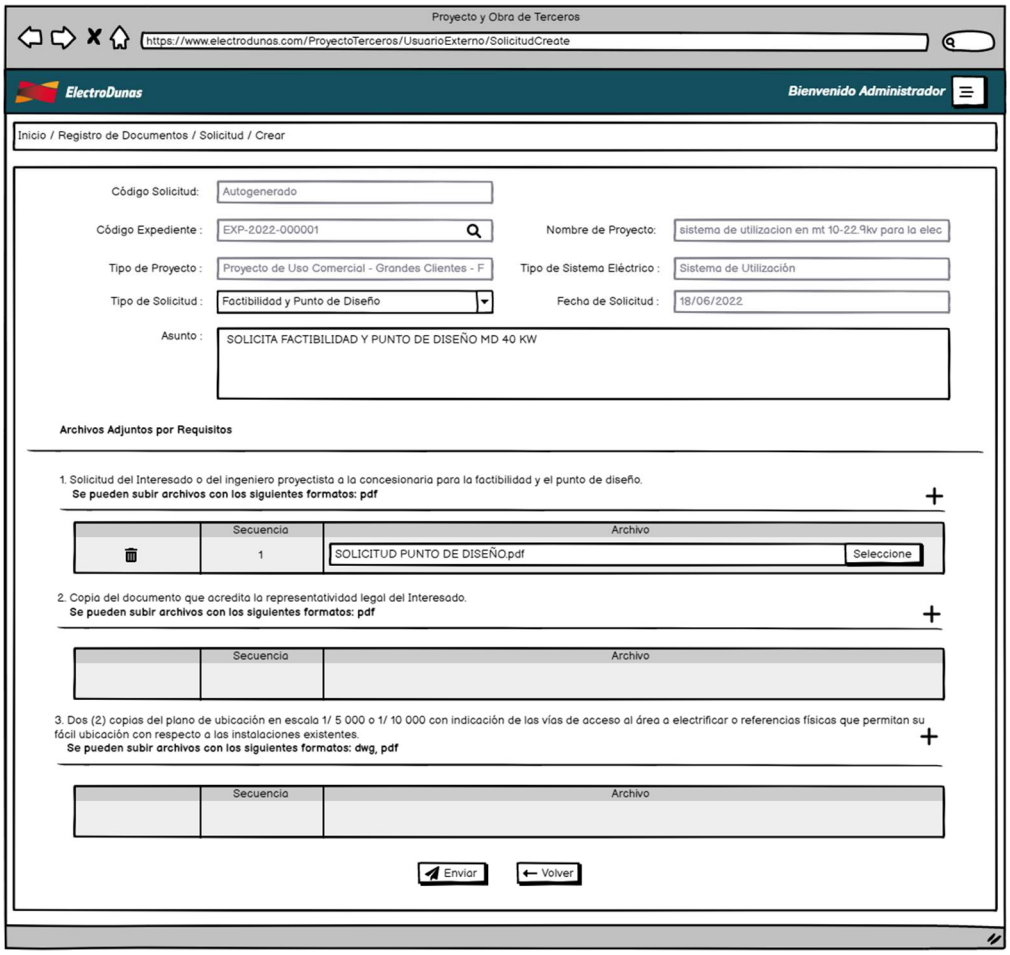


Figura 47 Prototipo de opción de adjuntar archivo en los requisitos en la creación de Solicitud - Vista del Tercero

Por parte del colaborador, podrá ver el módulo de revisión de documentos el cual en el siguiente prototipo muestra las opciones a presentarse.

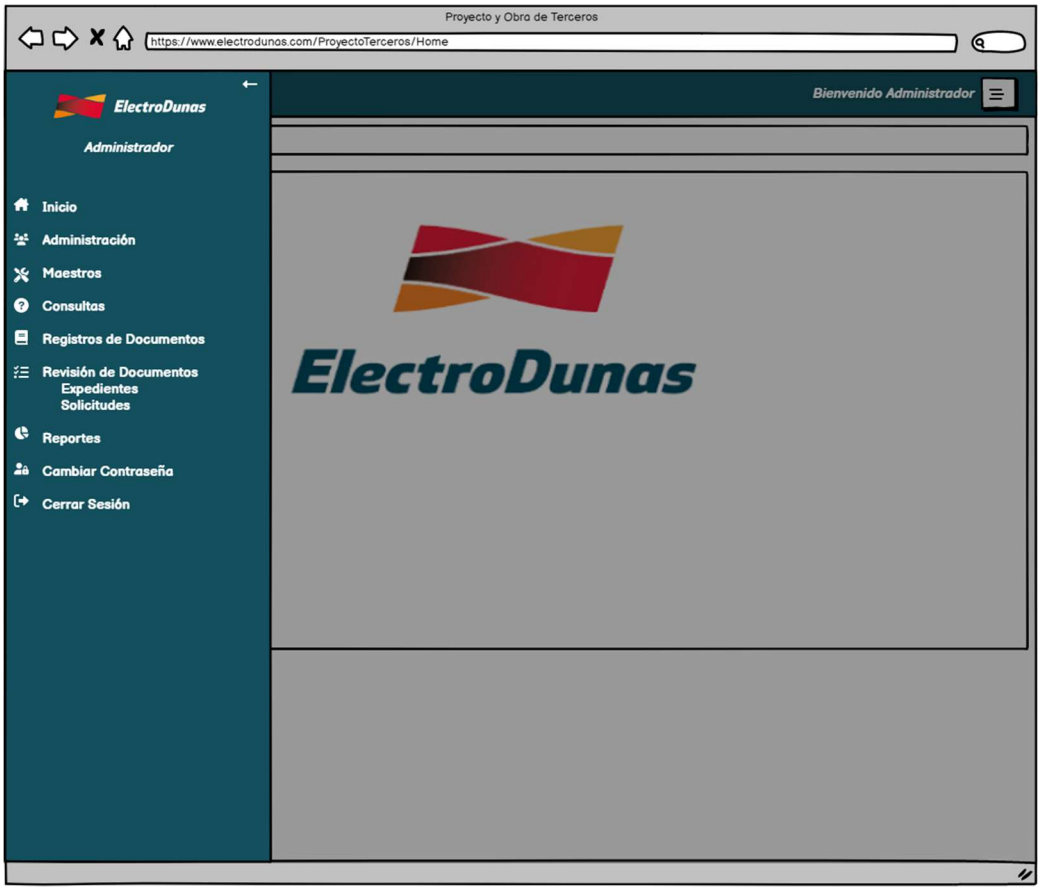


Figura 48 Prototipo de módulo de Revisión de Documentos del Portal

El colaborador como primera opción podrá ver el expediente creado por el tercero, el siguiente prototipo muestra cómo sería visible por parte del colaborador.

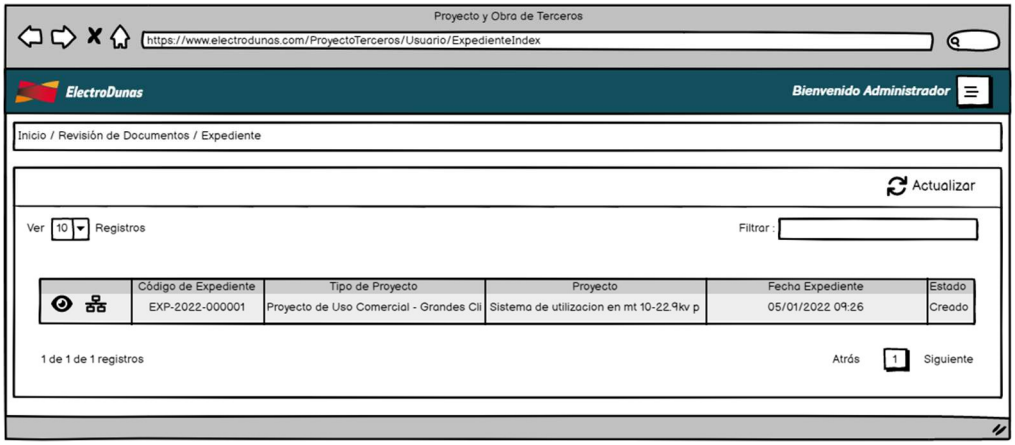


Figura 49 Prototipo de pantalla de listado de Expedientes - Vista Colaborador Interno

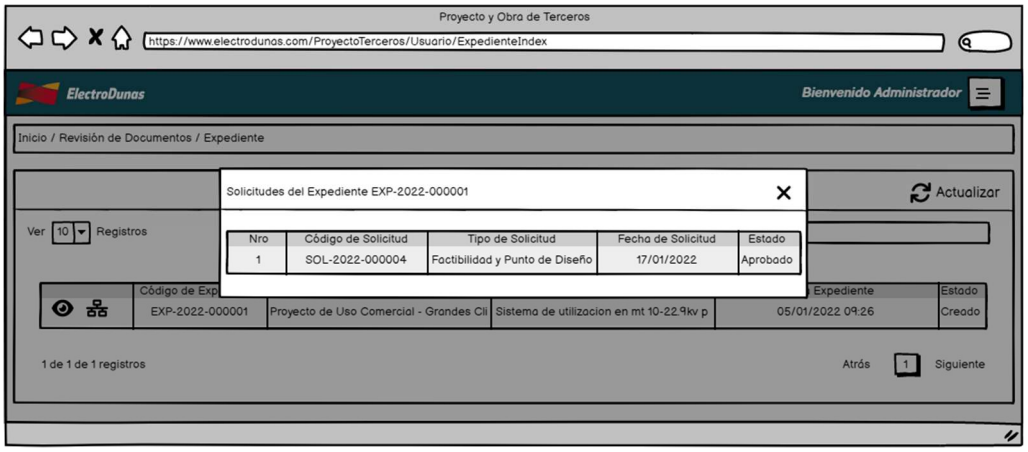


Figura 50 Prototipo de opción de consulta de solicitudes creadas de un Expediente - Vista Colaborador Interno

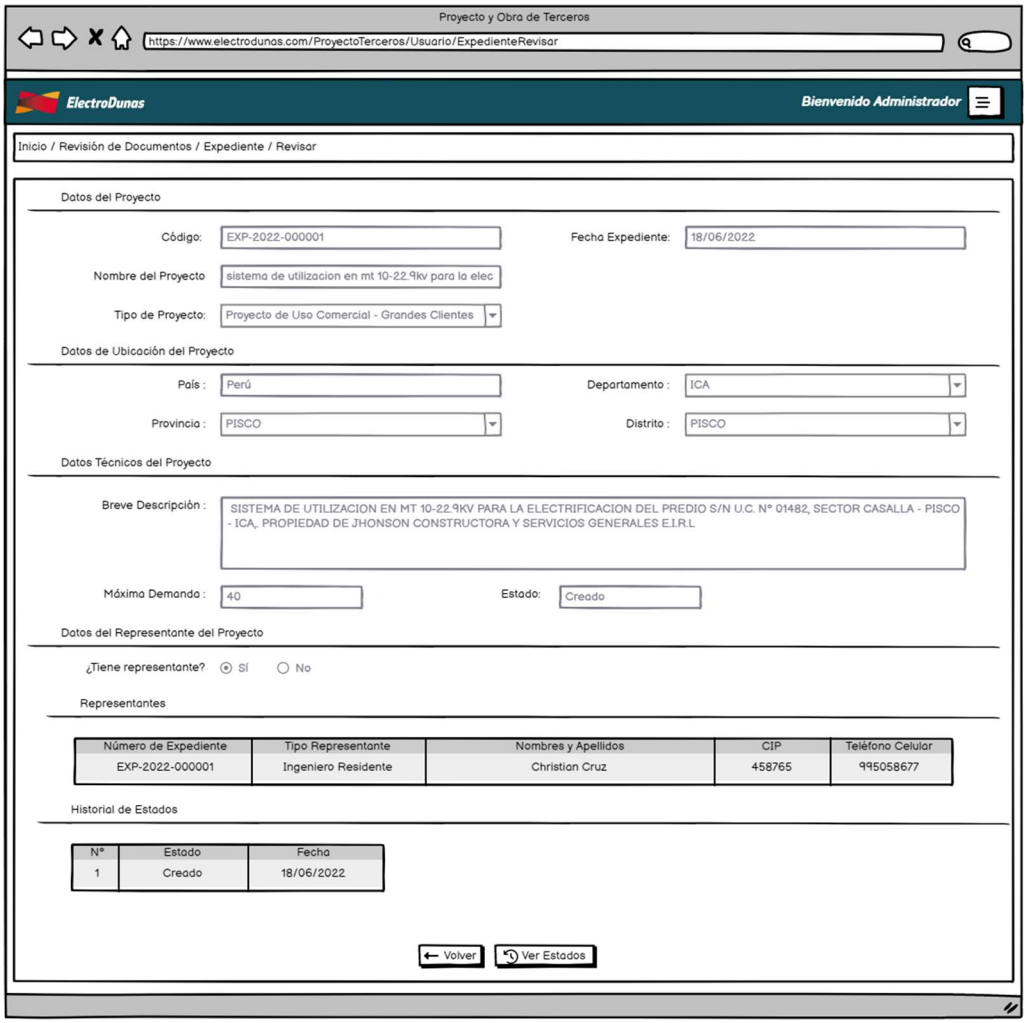


Figura 51 Prototipo de pantalla de revisión de Expediente - Vista Colaborador Interno

Asimismo, el colaborador podrá ver la solicitud creada por el tercero y podrá hacer la derivación correspondiente hasta concluir la atención del mismo, el siguiente prototipo muestra como estaría habilitada esta funcionalidad.

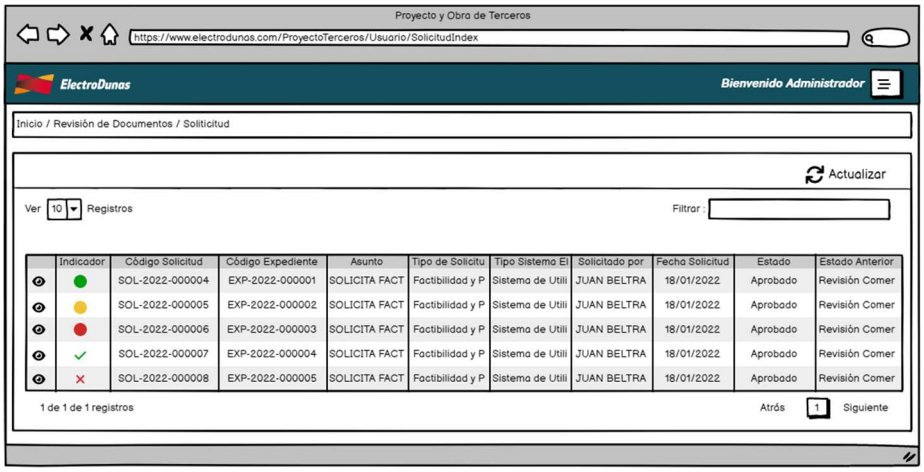


Figura 52 Prototipo de pantalla de consulta de Solicitudes - Vista Colaborador Interno

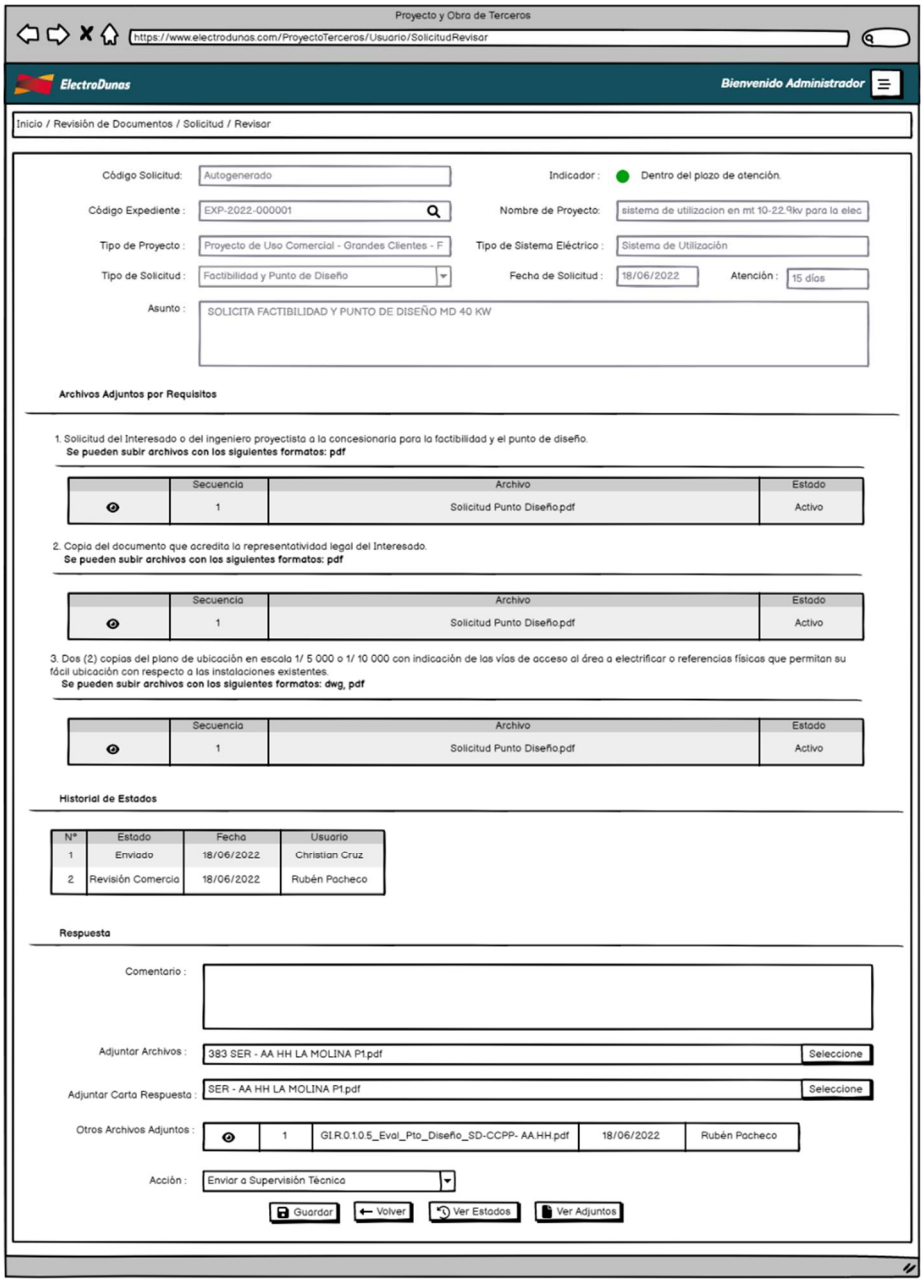


Figura 53 Prototipo de pantalla de revisión de Solicitud - Vista Colaborador Interno

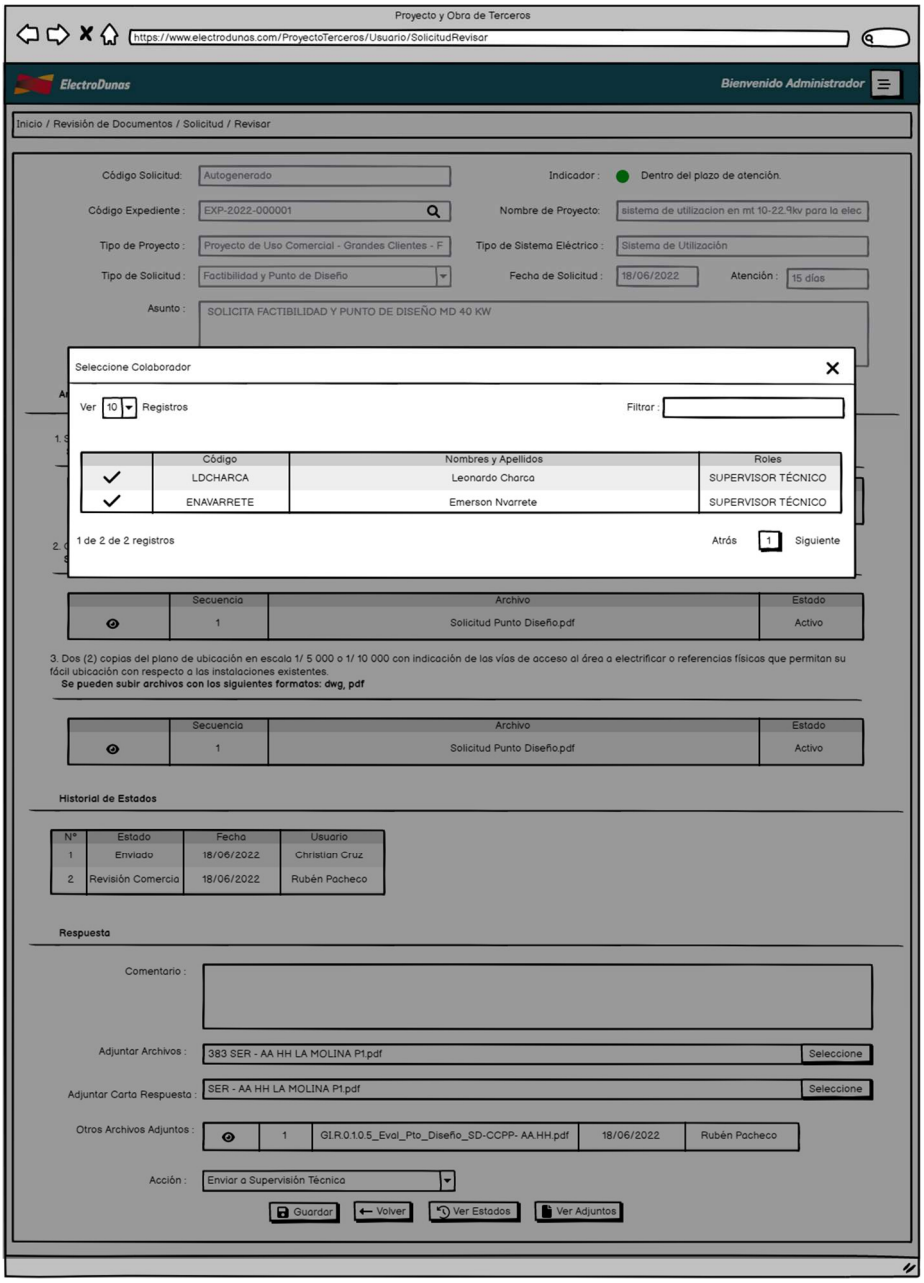


Figura 54 Prototipo de opción de derivación para la atención de una Solicitud - Vista Colaborador Interno

El colaborador podrá ver el módulo de reportes el cual el siguiente prototipo muestra las opciones habilitadas.

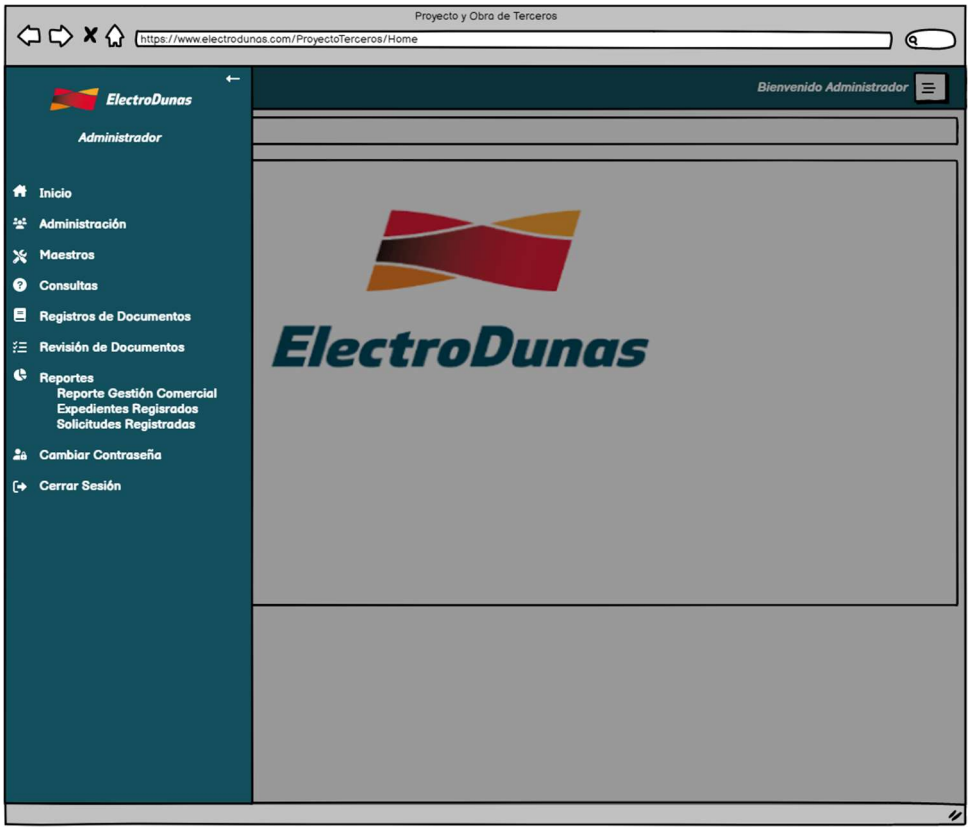


Figura 55 Prototipo de módulo de Reportes del portal

El colaborador podrá ingresar a los tres reportes que se muestra como opción en el módulo, en los siguientes prototipos se muestra como estarían conformadas las tres pantallas a consultar.

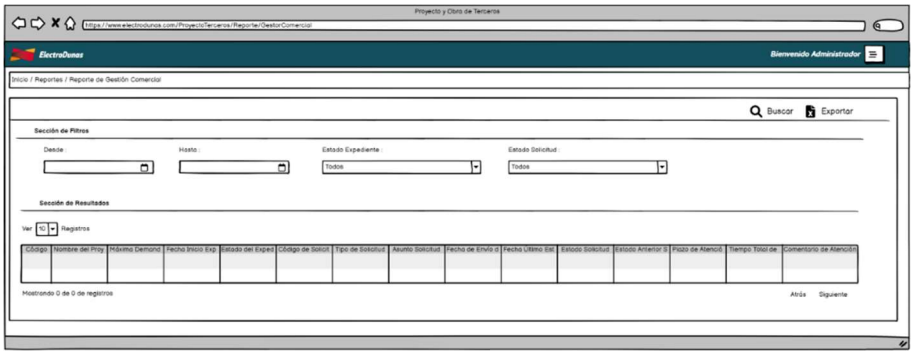


Figura 56 Prototipo de pantalla del Reporte de Gestión Comercial

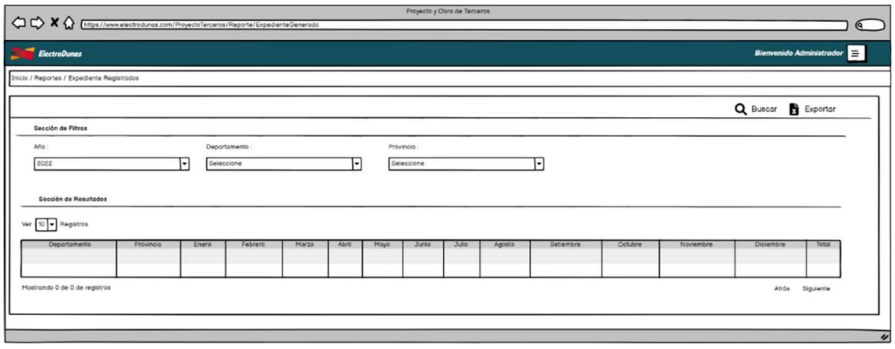


Figura 57 Prototipo de pantalla de Reporte de Expedientes Registrados

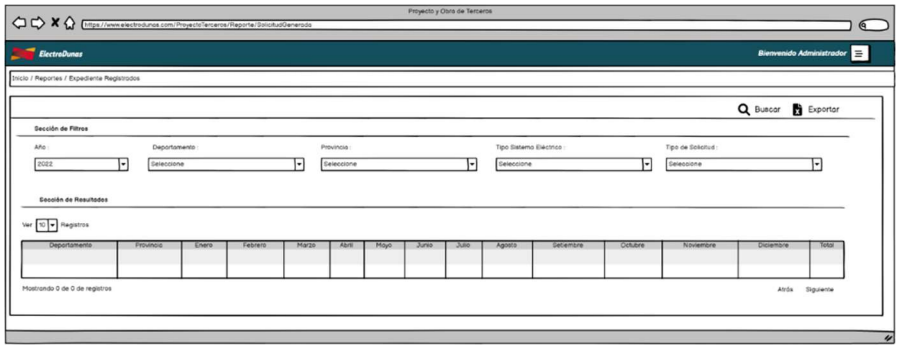


Figura 58 Prototipo de pantalla de Reporte de Solicitudes Registradas

Pruebas unitarias

Las pruebas unitarias se realizan luego del desarrollo del sistema, para el seguimiento de las pruebas unitarias se elaboró un listado de los diferentes casos de uso que se definieron al iniciar con el proyecto.

Tabla 33 Listado de Casos de Uso del Portal

N°	Descripción del Caso de Uso	¿Es resultado satisfactorio?
1	El sistema deberá permitir crear a un tercero su usuario, enviándole por correo electrónico sus credenciales de acceso, la contraseña será autogenerada.	Sí
2	El sistema deberá permitir asociar los módulos de acceso con las opciones y funcionalidades que tendrá el portal	Si
3	El sistema deberá permitir crear los diferentes perfiles asociando el módulo que verá cada uno de estos.	Si
4	El sistema deberá permitir crear a un colaborador (usuario interno), donde se permitirá asociar el perfil con el que podrá ingresar, su usuario y contraseña (autogenerada) será notificado al correo electrónico que se ingresará en la creación del mismo.	Si
5	El sistema deberá tener una opción por el cual el usuario podrá recuperar su contraseña, enviando una nueva contraseña autogenerada.	Si

6	El sistema deberá de solicitar el cambio de contraseña luego del primer inicio de sesión luego de crear el usuario o del restablecimiento de contraseña	Si
7	El sistema deberá permitir la creación, edición de los tipos de sistemas eléctricos	Si
8	El sistema deberá permitir la creación, edición de los típicos constructivos teniendo la opción de poder compartir un archivo en formato PDF para que pueda ser visto por el cliente	Si
9	El sistema deberá permitir la creación y actualización de los tipos de solicitud detallando los requisitos que se tendrá que presentar en cada una de ellos seleccionando el tipo de formato que aceptará, tales como: PDF, imagen, AutoCad, txt, Excel.	Si
10	El sistema deberá permitir la creación y actualización de los tipos de proyectos, asociando los tipos de solicitud en presentar en cada tipo de proyecto, así como la determinación del orden de presentar cada solicitud	Si
11	El sistema deberá de contar con una vista en la que el tercero pueda consultar el archivo subido por cada típico constructivo y poder descargarlo	Si
12	El sistema deberá de contar con una vista en la que el tercero pueda consultar las solicitudes a presentar por cada tipo de proyecto, así como los requisitos a considerar en cada tipo de solicitud, esta información se podrá descargar en archivo PDF y Excel.	Si
13	El sistema deberá permitir la creación de un expediente (proyecto) por parte del tercero, esta creación se le notificará al tercero y	Si

	todos los colaboradores mediante correo electrónico.	
14	El sistema deberá tener una vista en la que el tercero podrá consultar todas las solicitudes que se han creado y en el estado en la que se encuentran de un determinado expediente	Si
15	El sistema deberá permitir al tercero editar o agregar un representante del expediente (proyecto)	Si
16	El sistema deberá permitir la creación de una solicitud por parte del tercero y subir los archivos necesarios a presentar por cada requisito solicitado, esta creación notificará al tercero como a los colaboradores mediante correo electrónico.	Si
17	El sistema deberá de validar que un tercero no tenga una solicitud rechazada para poder continuar con el flujo de creación de solicitudes de un determinado expediente, de tener una solicitud en estado rechazada el sistema le pedirá que presente la misma solicitud hasta que esta pueda ser aprobada	Si
18	El sistema deberá tener una vista en la que el colaborador podrá realizar la consulta de todos los expedientes generados, así como el listado de las solicitudes asociadas y el estado en el que se encuentra.	Si
19	El sistema deberá tener una vista en la que el colaborador pueda ver el detalle de las solicitudes generadas, así como los archivos cargados en cada requisito	Si
20	El sistema deberá permitir que un colaborador pueda asignar la solicitud a otro colaborador para que pueda ser atendido, pudiendo adjuntar archivos y comentarios,	Si

	esta asignación notificará por correo electrónico la acción realizada.	
21	El sistema deberá tener una vista para que se pueda revisar los reportes generados, así como la descarga en formato Excel (.xlsx)	Si

Luego de realizar las pruebas unitarias se notifica al jefe del área de desarrollo, el cual autoriza la publicación del sistema en un entorno de pruebas, el cual llamamos entorno de capacitación, luego de esta publicación se procede a realizar las coordinaciones con el área solicitante para programar las pruebas de usuario.

Las pruebas de usuario se desarrollaron mediante sesiones en teams con acompañamiento del área de desarrollo de la gerencia de TI, el cual todos los resultados, incidentes y observaciones se evidencian en un documento llamado "Formato N° 2 – Acta de reunión" donde se detalla la minuta de la sesión los compromisos por parte de los presentes.

Luego de culminar las pruebas de usuario se solicita la emisión de la conformidad para poder pasar a producción el sistema, para lo cual se elabora el documento llamado "Formato N° 7 – Pase a Producción" donde se indica en donde se realizaron las pruebas y la ruta donde se está guardando toda la información.

Implementación (código fuente)

Registro de usuarios internos (Colaborador)

```
using ProyectoTerceros.Models;
using ProyectoTerceros.Models.Registro.UsuarioExterno;
using ProyectoTerceros.Models.Registro.UsuarioInterno;
using ProyectoTerceros.Models.Seguridad;
using System;
using System.Collections.Generic;
using System.Configuration;
using System.IO;
using System.Linq;
using System.Net;
using System.Net.Http;
using System.Net.Http.Headers;
using System.Net.Mail;
using System.Text.RegularExpressions;
using System.Threading.Tasks;
using System.Web;
using System.Web.Mvc;
```

```
namespace ProyectoTerceros.Controllers.Seguridad
{
   public class UsuarioController : Controller
   {
      // GET: Usuario
      private ProyectoTercerosEntities db = new ProyectoTercerosEntities();
      #region VISTA DE USUARIO
      //[Authorize]
      public ActionResult Index()
      {
         if (Session["Usuario"] == null)
         {
            return RedirectToAction("Index", "Login");
         }
         else
         {
            return View();
         }
      }
      public JsonResult List()
      {
         List<UsuarioQuery> usuarioQuery = new List<UsuarioQuery>();
         List<USUARIO> usuarios = db.USUARIO.Where(p => p.ESTADO == 1 ||
p.ESTADO == 0 || p.ESTADO == 4).ToList();
         foreach (var usuario in usuarios)
         {
            UsuarioQuery uq = new UsuarioQuery();
            uq.CODIGO_USUARIO = usuario.CODIGO_USUARIO;
            uq.NOMBRE = usuario.NOMBRE;
            uq.ESTADO = (Int32)usuario.ESTADO;
            uq.APELLIDO_MATERNO = usuario.APELLIDO_MATERNO;
            uq.APELLIDO_PATERNO = usuario.APELLIDO_PATERNO;
            uq.TIPO_USUARIO = usuario.TIPO_USUARIO;
            usuarioQuery.Add(uq);
         }
         return Json(usuarioQuery);
      }
```

```
public JsonResult GetUsuariosInternos(int accion)
{
  // ROLES CON LA VISTA DE SUPERVISOR TECNICO Y SUPERVISOR COMERCIAL
  string codigo_usuario_login = Session["Codigo"].ToString();
  List<MODULO_OPCION> listModuloOpcion = new List<MODULO_OPCION>();
  if (accion == 8)
  {
    listModuloOpcion = db.MODULO_OPCION.Where(p => p.CODIGO_OPCION == "OP017" && p.ESTADO == 1).ToList();
  }
  if (accion == 13)
  {
    listModuloOpcion = db.MODULO_OPCION.Where(p => p.CODIGO_OPCION == "OP027" && p.ESTADO == 1).ToList();
  }
  if (accion == 14)
  {
    listModuloOpcion = db.MODULO_OPCION.Where(p => p.CODIGO_OPCION == "OP026" && p.ESTADO == 1).ToList();
  }
  else if (accion == 4 || accion == 5)
  {
    listModuloOpcion = db.MODULO_OPCION.Where(p => p.CODIGO_OPCION == "OP013" && p.ESTADO == 1).ToList();
  }
  // OBTENER LOS ROLES QUE TIENEN ESOS MODULOS
  List<String> codigoRoles = new List<String>();
  foreach (var list in listModuloOpcion)
  {
    List<MODULO_ROL> listModuloRol = db.MODULO_ROL.Where(p => p.CODIGO_MODULO == list.CODIGO_MODULO && p.ESTADO == 1).ToList();
    foreach (var listMR in listModuloRol)
    {
      string codigoRol = "";
      codigoRol = listMR.CODIGO_ROL;
```

```
                if (codigoRoles.Where(p => p == codigoRol).ToList().Count == 0)
                {
                    codigoRoles.Add(codigoRol);
                }
            }
        }
        List<UsuarioQuery> usuarioQuery = new List<UsuarioQuery>();
        foreach (var codRol in codigoRoles)
        {
            // VALIDAR QUE USUARIO TOMAMOS
            List<OPCION_REVISION_DETALLE> opcionRevisionDetalle = new
List<OPCION_REVISION_DETALLE>();
            if (accion == 4) // GESTOR COMERCIAL
            {
                opcionRevisionDetalle = db.OPCION_REVISION_DETALLE.Where(p =>
p.CODIGO_ROL == codRol && p.ES_GESTOR_COMERCIAL == 1 && p.ESTADO
== 1).ToList();
            }
            if (accion == 5) // SUPERVISOR TECNICO
            {
                opcionRevisionDetalle = db.OPCION_REVISION_DETALLE.Where(p =>
p.CODIGO_ROL == codRol && p.ES_SUPERVISOR_TECNICO == 1 &&
p.ESTADO == 1).ToList();
            }
            if (opcionRevisionDetalle.Count != 0 || accion == 8 || accion == 13 || accion ==
14)
            {
                List<USUARIO_ROL> usuRol = db.USUARIO_ROL.Where(p =>
p.CODIGO_ROL == codRol && p.ESTADO == 1).ToList();
                foreach (var usu in usuRol)
                {
                    if (usuarioQuery.Where(p => p.CODIGO_USUARIO ==
usu.CODIGO_USUARIO).ToList().Count == 0)
                    {
                        List<USUARIO> usuarios = db.USUARIO.Where(p =>
(p.CODIGO_USUARIO == usu.CODIGO_USUARIO && p.CODIGO_USUARIO !=
```

```
codigo_usuario_login) && p.ESTADO == 1 && p.TIPO_USUARIO == "Interno").ToList();
                    foreach (var usuario in usuarios)
                    {
                        UsuarioQuery uq = new UsuarioQuery();
                        uq.CODIGO_USUARIO = usuario.CODIGO_USUARIO;
                        uq.NOMBRE = usuario.NOMBRE;
                        uq.APELLIDO_MATERNO = usuario.APELLIDO_MATERNO;
                        uq.APELLIDO_PATERNO = usuario.APELLIDO_PATERNO;
                        uq.TIPO_USUARIO = usuario.TIPO_USUARIO;
                        List<USUARIO_ROL> usuarioRol = db.USUARIO_ROL.Where(p => p.CODIGO_USUARIO == usuario.CODIGO_USUARIO && p.ESTADO == 1).ToList();
                        List<UsuarioRolQuery> urquery = new List<UsuarioRolQuery>();
                        int contador = 0;
                        string roles = "";
                        foreach (var ur in usuarioRol)
                        {
                            contador++;
                            if (contador == 1)
                            {
                                roles = roles + ur.ROL.DESCRIPCION;
                            }
                            else
                            {
                                roles = roles + ", " + ur.ROL.DESCRIPCION;
                            }
                        }
                        uq.roles = roles;
                        //roles
                        usuarioQuery.Add(uq);
                    }
                }
            }
        }
        return Json(usuarioQuery, JsonRequestBehavior.AllowGet);
```

```
}
// GET: Usuario/Details/5
public ActionResult Details(int id)
{
    return View();
}
[Route("/Usuario/Editar/{CODIGO_USUARIO}/{TIPO_USUARIO}")]
public ActionResult Editar(string CODIGO_USUARIO, string TIPO_USUARIO)
{
    if (Session["Usuario"] == null)
    {
        return RedirectToAction("Index", "Login");
    }
    else
    {
        if (TIPO_USUARIO == "Interno")
        {
            List<EstadoQuery> estados = new List<EstadoQuery>();
            estados.Add(new EstadoQuery(1, "Activo"));
            estados.Add(new EstadoQuery(0, "Inactivo"));
            ViewBag.Estados = estados;
            ViewBag.TipoDocumento = ListarTipoDocumento();
            ViewBag.Codigo = CODIGO_USUARIO;
            return View("Create");
        }
        else
        {
            List<EstadoQuery> estados = new List<EstadoQuery>();
            estados.Add(new EstadoQuery(1, "Activo"));
            estados.Add(new EstadoQuery(0, "Inactivo"));
            ViewBag.Estados = estados;
            ViewBag.TipoDocumento = ListarTipoDocumento();
            ViewBag.Codigo = CODIGO_USUARIO;
            return View("EditarUsuarioExterno");
        }
    }
}
```

```
[HttpPost]
public ActionResult Eliminar(string codigo_usuario)
{
   USUARIO usuario = db.USUARIO.Find(codigo_usuario);
   usuario.ESTADO = 2;
   usuario.FECHA_EDICION = DateTime.Now;
   usuario.USUARIO_EDICION = Session["Codigo"].ToString();
   db.SaveChanges();
   return Json("Ok");
}
// GET: Usuario/Create
public ActionResult Create()
{
   if (Session["Usuario"] == null)
   {
      return RedirectToAction("Index", "Login");
   }
   else
   {
      List<EstadoQuery> estados = new List<EstadoQuery>();
      estados.Add(new EstadoQuery(1, "Activo"));
      estados.Add(new EstadoQuery(0, "Inactivo"));
      ViewBag.Estados = estados;
      ViewBag.TipoDocumento = ListarTipoDocumento();
      return View();
   }
}
private List<TipoDocumentoQuery> ListarTipoDocumento()
{
   IList<TIPO_DOCUMENTO> tiposDocumentos =
db.TIPO_DOCUMENTO.ToList();
   List<TipoDocumentoQuery> tdq = new List<TipoDocumentoQuery>();
   TipoDocumentoQuery tdqu = new TipoDocumentoQuery();
   tdqu.codigo_tipo_documento = "";
   tdqu.tipo_documento = "Seleccione";
   tdq.Add(tdqu);
   foreach (var td in tiposDocumentos)
```

```
    {
        tdqu = new TipoDocumentoQuery();
        tdqu.codigo_tipo_documento = td.CODIGO_TIPO_DOCUMENTO;
        tdqu.tipo_documento = td.DESCRIPCION;
        tdq.Add(tdqu);
    }
    return tdq;
}
// POST: Usuario/Create
[HttpPost]
public ActionResult Create(FormCollection collection)
{
    if (Session["Usuario"] == null)
    {
        return RedirectToAction("Index", "Login");
    }
    else
    {
        try
        {
            // TODO: Add insert logic here
            string nombre = collection["nombre"];
            return RedirectToAction("Index");
        }
        catch
        {
            return View();
        }           }       }
public JsonResult Guardar(UsuarioInput input)
{
    Status status = new Status();
    try
    {
        // TODO: Add insert logic here
        if (input.tipo_documento == null)
        {
            input.tipo_documento = string.Empty;
```

```
            }
            if (input.numero_documento == null)
            {
               input.numero_documento = string.Empty;
            }
            if (input.pais == null)
            {
               input.pais = string.Empty;
            }
            if (input.ubigeo == null)
            {
               input.ubigeo = string.Empty;
            }
            if (input.direccion_postal == null)
            {
               input.direccion_postal = string.Empty;
            }
            if (input.telefono_fijo == null)
            {
               input.telefono_fijo = string.Empty;
            }
            if (input.telefono_celular == null)
            {
               input.telefono_celular = string.Empty;
            }
            if (input.telefono_emergencia == null)
            {
               input.telefono_emergencia = string.Empty;
            }
            string mensaje = "";
            string estado = "Ok";
            // VALIDAR QUE NO EXISTA OTRO USUARIO CON EL MISMO
CORREO
            if (input.codigo_inicial == null)
            {
               // CREAR
```

```
            List<USUARIO> usuarioCorreo = db.USUARIO.Where(p =>
p.CORREO_ELECTRONICO.ToUpper().Trim() ==
input.correo_electronico.ToUpper().Trim() && (p.ESTADO == 1 || p.ESTADO ==
0)).ToList();
            if (usuarioCorreo.Count != 0)
            {
              estado = "Error";
              mensaje = mensaje + "Ya existe un Usuario con el mismo 'Correo
Electrónico'.";
            }          }
          else
          {
            // EDITAR
            List<USUARIO> usuarioCorreo = db.USUARIO.Where(p =>
p.CORREO_ELECTRONICO.ToUpper().Trim() ==
input.correo_electronico.ToUpper().Trim() && p.CODIGO_USUARIO !=
input.codigo_inicial && (p.ESTADO == 1 || p.ESTADO == 0)).ToList();
            if (usuarioCorreo.Count != 0)
            {
              estado = "Error";
              mensaje = mensaje + "Ya existe un Usuario con el mismo 'Correo
Electrónico'.";
            }          }
          if (estado.Equals("Ok"))
          {
            if (input.codigo_inicial == null)
            {
              try
              {
                // CREAR USUARIO
                USUARIO usuario = new USUARIO();
                string contrasenia = GenerarContrasenia();
                usuario.CODIGO_USUARIO = input.codigo_usuario;
                usuario.NOMBRE = input.nombre;
                usuario.APELLIDO_PATERNO = input.ap_paterno;
                usuario.APELLIDO_MATERNO = input.ap_materno;
                usuario.CORREO_ELECTRONICO = input.correo_electronico;
```

```
                    usuario.CLAVE = contrasenia;
                    usuario.CONFIRMAR_CLAVE = "";
                    usuario.EDITA_DATOS_PROYECTO = esEditaDatosProyecto;
                    usuario.EDITA_DATOS_OBRA = esEditaDatosObra;
                    usuario.GENERA_REPORTE_VNR = esReporteVNR;
                    // CAMPOS NO OBLIGATORIO
                    usuario.CODIGO_TIPO_DOCUMENTO = input.tipo_documento;
                    usuario.NUMERO_CIP = input.numero_cip;
                    usuario.NUMERO_DOCUMENTO = input.numero_documento;
                    usuario.CODIGO_PAIS = input.pais;
                    usuario.UBIGEO = input.ubigeo;
                    usuario.DIRECCION_POSTAL = input.direccion_postal;
                    usuario.TELEFONO_FIJO = input.telefono_fijo;
                    usuario.TELEFONO_CELULAR = input.telefono_celular;
                    usuario.TELEFONO_EMERGENCIA = input.telefono_emergencia;
                    // FIN DE CAMPOS DE NO OBLIGATORIO
                    usuario.TIPO_USUARIO = "Interno";
                    usuario.GERENCIA = input.gerencia;
                    usuario.JEFATURA = input.jefatura;
                    usuario.ESTADO = input.estado;
                    // INICIO DE CAMPOS NO OLIGATORIOS
                    usuario.ACEPTA_CONDICIONES_GENERALES = 1;
                    usuario.ACEPTA_CONDICIONES_SUSCRIPCION = 1;
                    // FIN DE CAMPOS NO OBLIGATORIOS
                    usuario.USUARIO_CREACION = Session["Codigo"].ToString();
                    usuario.USUARIO_EDICION = Session["Codigo"].ToString();
                    usuario.FECHA_CREACION = DateTime.Now;
                    usuario.FECHA_EDICION = DateTime.Now;
                    db.USUARIO.Add(usuario);
                    db.SaveChanges();
                    // ENVIAR CORREO CON LA CONTRASEÑA
                    string nombreCompleto = input.nombre.ToUpper() + " " +
input.ap_paterno.ToUpper() + " " + input.ap_materno.ToUpper();
                    Status statusCorreoElectronico =
EnviarMailContrasenia(input.correo_electronico, contrasenia, nombreCompleto);
                    // CREAR ROL - USUARIO
                    IList<UsuarioRolInput> roles = input.roles;
```

```
        if (roles.Count != 0)
        {
            foreach (var rol in roles)
            {
                USUARIO_ROL ur = new USUARIO_ROL();
                ur.CODIGO_ROL = rol.codigo_rol;
                ur.CODIGO_USUARIO = input.codigo_usuario;
                ur.ESTADO = 1;
                ur.USUARIO_CREACION = Session["Codigo"].ToString();
                ur.FECHA_CREACION = DateTime.Now;
                ur.USUARIO_EDICION = Session["Codigo"].ToString();
                ur.FECHA_EDICION = DateTime.Now;
                db.USUARIO_ROL.Add(ur);
                db.SaveChanges();
            }                       }
        estado = "Ok";
        mensaje = "Se creó Usuario satisfactoriamente.";
    }
    catch (Exception ex)
    {
        estado = "Error";
        mensaje = "Comuníquese con Soporte Técnico.";
    }                   }
else
{
    try
    {
        // EDITAR USUARIO
        USUARIO usuario = new USUARIO();
        usuario = db.USUARIO.Find(input.codigo_inicial);
        usuario.CODIGO_USUARIO = input.codigo_usuario;
        usuario.NOMBRE = input.nombre;
        usuario.APELLIDO_PATERNO = input.ap_paterno;
        usuario.APELLIDO_MATERNO = input.ap_materno;
        usuario.CORREO_ELECTRONICO = input.correo_electronico;
        // CAMPOS NO OBLIGATORIO
        usuario.CODIGO_TIPO_DOCUMENTO = input.tipo_documento;
```

```
                    usuario.NUMERO_CIP = input.numero_cip;
                    usuario.NUMERO_DOCUMENTO = input.numero_documento;
                    usuario.CODIGO_PAIS = input.pais;
                    usuario.UBIGEO = input.ubigeo;
                    usuario.DIRECCION_POSTAL = input.direccion_postal;
                    usuario.TELEFONO_FIJO = input.telefono_fijo;
                    usuario.TELEFONO_CELULAR = input.telefono_celular;
                    usuario.TELEFONO_EMERGENCIA = input.telefono_emergencia;
                    // FIN DE CAMPOS DE NO OBLIGATORIO
                    usuario.EDITA_DATOS_PROYECTO = esEditaDatosProyecto;
                    usuario.EDITA_DATOS_OBRA = esEditaDatosObra;
                    usuario.GENERA_REPORTE_VNR = esReporteVNR;
                    usuario.TIPO_USUARIO = "Interno";
                    usuario.GERENCIA = input.gerencia;
                    usuario.JEFATURA = input.jefatura;
                    usuario.ESTADO = input.estado;
                    usuario.USUARIO_EDICION = Session["Codigo"].ToString();
                    usuario.FECHA_EDICION = DateTime.Now;
                    // db.USUARIO.Add(usuario);
                    db.SaveChanges();
                    // EDITAR ROLES
                    IList<USUARIO_ROL> usuarioroles = db.USUARIO_ROL.Where(p
=> p.CODIGO_USUARIO == input.codigo_inicial).ToList();
                    IList<UsuarioRolInput> roles = input.roles;
                    foreach (var usurol in usuarioroles)
                    {
                        usurol.ESTADO = 0;
                        usurol.CODIGO_USUARIO = input.codigo_usuario;
                        usurol.USUARIO_EDICION = Session["Codigo"].ToString();
                        usurol.FECHA_EDICION = DateTime.Now;
                        db.SaveChanges();
                    }
                    if (roles.Count != 0)
                    {
                        try
                        {
                            foreach (var rol in roles)
```

```
                    {
                        IList<USUARIO_ROL> usuariorolInd = db.USUARIO_ROL.Where(p => p.CODIGO_USUARIO == input.codigo_usuario && p.CODIGO_ROL == rol.codigo_rol).ToList();
                        if (usuariorolInd.Count == 0)
                        {
                            USUARIO_ROL ur = new USUARIO_ROL();
                            ur.CODIGO_ROL = rol.codigo_rol;
                            ur.CODIGO_USUARIO = input.codigo_usuario;
                            ur.ESTADO = 1;
                            ur.USUARIO_CREACION = Session["Codigo"].ToString();
                            ur.FECHA_CREACION = DateTime.Now;
                            ur.USUARIO_EDICION = Session["Codigo"].ToString();
                            ur.FECHA_EDICION = DateTime.Now;
                            db.USUARIO_ROL.Add(ur);
                            db.SaveChanges();
                        }
                        else
                        {
                            foreach (var urol in usuariorolInd)
                            {
                                urol.ESTADO = 1;
                                urol.USUARIO_EDICION = Session["Codigo"].ToString();
                                urol.FECHA_EDICION = DateTime.Now;
                                db.SaveChanges();                                }
}                        }                    }
                catch (Exception ex)
                {
                    throw;
                }                }
            estado = "Ok";
            mensaje = "Se éditó Usuario satisfactoriamente.";
        }
        catch (Exception ex)
        {
            estado = "Error";
```

```
                throw;
            }           }           }
        status.rpta_estado = estado;
        status.rpta_mensaje = mensaje;
        return Json(status);
    }
    catch (Exception ex)
    {
        status.rpta_estado = "Error";
        status.rpta_mensaje = "Comuníquese con Soporte Técnico";
        return Json(status);
    }       }
public JsonResult EscribirLog(string motivo, string mensaje)
{
    Log log = new Log();
    log.Add(motivo); log.Add(mensaje);
    return Json("Ok");
}
 public JsonResult Leer(string codigo)
{
    USUARIO usuario = db.USUARIO.Find(codigo);
    UsuarioInd usuarioInd = new UsuarioInd();
    if (usuario != null)
    {
        if (usuario.ESTADO == 0 || usuario.ESTADO == 1 || usuario.ESTADO == 4)
        {
            usuarioInd.codigo_usuario = usuario.CODIGO_USUARIO;
            usuarioInd.nombre = usuario.NOMBRE;
            usuarioInd.ap_paterno = usuario.APELLIDO_PATERNO;
            usuarioInd.ap_materno = usuario.APELLIDO_MATERNO;
            usuarioInd.correo_electronico = usuario.CORREO_ELECTRONICO;
            usuarioInd.tipo_documento = usuario.CODIGO_TIPO_DOCUMENTO;
            usuarioInd.numero_documento = usuario.NUMERO_DOCUMENTO;
            usuarioInd.pais = usuario.CODIGO_PAIS;
            usuarioInd.ubigeo = usuario.UBIGEO;
            usuarioInd.direccion_postal = usuario.DIRECCION_POSTAL;
            usuarioInd.telefono_fijo = usuario.TELEFONO_FIJO;
```

```
            usuarioInd.telefono_celular = usuario.TELEFONO_CELULAR;
            usuarioInd.telefono_emergencia = usuario.TELEFONO_EMERGENCIA;
            usuarioInd.numero_cip = usuario.NUMERO_CIP;
            usuarioInd.gerencia = usuario.GERENCIA;
            usuarioInd.jefatura = usuario.JEFATURA;
            usuarioInd.estado = Int32.Parse(usuario.ESTADO.ToString());
            List<USUARIO_ROL> usuRol = db.USUARIO_ROL.Where(p =>
p.CODIGO_USUARIO == codigo && p.ESTADO == 1).ToList();
            List<UsuarioRolQuery> urquery = new List<UsuarioRolQuery>();
            foreach (var ur in usuRol)
            {
              UsuarioRolQuery urq = new UsuarioRolQuery();
              urq.codigo_rol = ur.CODIGO_ROL;
              urquery.Add(urq);
            }
            List<USUARIO_ADJUNTO> adjuntos =
db.USUARIO_ADJUNTO.Where(p => p.ID_USUARIO == codigo && p.ELIMINADO
== 0).ToList();
            List<UsuarioDetalleAdjuntoQuery> detalles = new
List<UsuarioDetalleAdjuntoQuery>();
            foreach (var adj in adjuntos)
            {
              UsuarioDetalleAdjuntoQuery det = new UsuarioDetalleAdjuntoQuery();
              det.contenido = adj.NOMBRE_ARCHIVO;
              detalles.Add(det);
            }
            usuarioInd.roles = urquery;
            usuarioInd.adjuntos = detalles;
            usuarioInd.rpta_estado = "Ok";
            usuarioInd.rpta_mensaje = "Usuario existe";
          }
          else
          {
            usuarioInd.rpta_estado = "Error";
            usuarioInd.rpta_mensaje = "Usuario no existe";
          }
        }
```

```
    else
    {
        usuarioInd.rpta_estado = "Error";
        usuarioInd.rpta_mensaje = "Usuario no existe";
    }
    return Json(usuarioInd);
}
// GET: Usuario/Edit/5
//[Authorize]
public ActionResult Edit(int id)
{
    if (Session["Usuario"] == null)
    {
        return RedirectToAction("Index", "Login");
    }
    else
    {
        return View();
    }    }
[HttpPost]
public ActionResult Edit(int id, FormCollection collection)
{
    if (Session["Usuario"] == null)
    {
        return RedirectToAction("Index", "Login");
    }
    else
    {
        try
        {
            // TODO: Add update logic here
            return RedirectToAction("Index");
        }
        catch
        {
            return View();
        }          }       }
```

```
        // POST: Usuario/Delete/5
        [HttpPost]
        public ActionResult Delete(int id, FormCollection collection)
        {
            if (Session["Usuario"] == null)
            {
                return RedirectToAction("Index", "Login");
            }
            else
            {
                try
                {
                    // TODO: Add delete logic here
                    return RedirectToAction("Index");
                }
                catch
                {
                    return View();
                }           }       }
        #endregion
        private string GenerarContrasenia()
        {
            Random rdn = new Random();
            string                                caracteres                                =
"abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890
%$#@";
            int longitud = caracteres.Length;
            char letra;
            int longitudContrasenia = 10;
            string contraseniaAleatoria = string.Empty;
            for (int i = 0; i < longitudContrasenia; i++)
            {
                letra = caracteres[rdn.Next(longitud)];
                contraseniaAleatoria += letra.ToString();
            }
            return contraseniaAleatoria;        }
        private class Log
```

```
    {
        private string path = Path.Combine(ConfigurationManager.AppSettings["RutaGuardarArchivos"].ToString() + "\\Logs\\Usuario-Solicitud");
        private string bringo = "\n";
        public void Add(string mensaje)
        {
            CreateDirectory();
            string nombre = GetNameFile();
            string cadena = "";
            cadena = DateTime.Now + " ==========> " + mensaje + Environment.NewLine;
            StreamWriter sw = new StreamWriter(path + "/" + nombre, true);
            sw.Write(cadena);
            sw.Close();
        }
        private void CreateDirectory()
        {
            try
            {
                if (!Directory.Exists(path))
                    Directory.CreateDirectory(path);
            }
            catch (DirectoryNotFoundException ex)
            {
                throw new Exception(ex.Message);
            }           }
        private string GetNameFile()
        {
            string nombre = "";

            nombre = "log_" + DateTime.Now.Day + "_" + DateTime.Now.Month + "_" + DateTime.Now.Year + ".txt";
            return nombre;
        }       }   }
}
```

Registro de usuarios terceros (Tercero)

```
using ProyectoTerceros.Models;
using ProyectoTerceros.Models.Administracion;
using ProyectoTerceros.Models.Registro.UsuarioExterno;
using ProyectoTerceros.Models.Registro.UsuarioInterno;
using ProyectoTerceros.Models.Seguridad;
using System;
using System.Collections.Generic;
using System.Configuration;
using System.IO;
using System.Linq;
using System.Net.Http;
using System.Net.Http.Headers;
using System.Text.RegularExpressions;
using System.Threading.Tasks;
using System.Web;
using System.Web.Mvc;
namespace ProyectoTerceros.Controllers.Seguridad
{
   public class UsuarioExternoController : Controller
   {
      // GET: Usuario
      private ProyectoTercerosEntities db = new ProyectoTercerosEntities();
      // GET: UsuarioExterno
      public ActionResult Index()
      {
         if (Session["Usuario"] == null)
         {
            return RedirectToAction("Index", "Login");
         }
         else
         {
            return View();
         }       }
      // GET: UsuarioExterno/Create
      public ActionResult Create()
      {
```

```
    List<EstadoQuery> estados = new List<EstadoQuery>();
    estados.Add(new EstadoQuery(1, "Activo"));
    estados.Add(new EstadoQuery(0, "Inactivo"));
    ViewBag.Estados = estados;
    ViewBag.TipoDocumento = ListarTipoDocumento();
    return View();
}
        // POST: UsuarioExterno/Create
[HttpPost]
public ActionResult Create(FormCollection collection)
{
    if (Session["Usuario"] == null)
    {
        return RedirectToAction("Index", "Login");
    }
    else
    {
        try
        {
            return RedirectToAction("Index");
        }
        catch
        {
            return View();
        }           }       }
// GET: UsuarioExterno/Edit/5
public ActionResult Edit(int id)
{
    if (Session["Usuario"] == null)
    {
        return RedirectToAction("Index", "Login");
    }
    else
    {
        return View();
    }       }
```

```
// POST: UsuarioExterno/Edit/5
[HttpPost]
public ActionResult Edit(int id, FormCollection collection)
{
   if (Session["Usuario"] == null)
   {
      return RedirectToAction("Index", "Login");
   }
   else
   {
      try
      {
         return RedirectToAction("Index");
      }
      catch
      {
         return View();
      }          }       }

public JsonResult Guardar(UsuarioInput input)
{
   try
   {
      // TODO: Add insert logic here
      Status status = new Status();
      if (estado.Equals("Ok"))
      {
         if (input.codigo_inicial == null)
         {
            try
            {
               // CREAR USUARIO
               USUARIO usuario = new USUARIO();
               string contrasenia = GenerarContrasenia();
               input.codigo_usuario = GenerarCodigo();
               usuario.CODIGO_USUARIO = input.codigo_usuario;
```

```
                    //usuario.CODIGO_USUARIO = input.codigo_usuario;
                    usuario.NOMBRE = input.nombre.ToUpper();
                    usuario.APELLIDO_PATERNO = input.ap_paterno.ToUpper();
                    usuario.APELLIDO_MATERNO = input.ap_materno.ToUpper();
                    usuario.CORREO_ELECTRONICO                                    =
input.correo_electronico.ToLower();
                    usuario.CLAVE = contrasenia;
                    usuario.CONFIRMAR_CLAVE = "";
                    // CAMPOS NO OBLIGATORIO
                    usuario.CODIGO_TIPO_DOCUMENTO = input.tipo_documento;
                    usuario.NUMERO_DOCUMENTO = input.numero_documento;
                    usuario.CODIGO_PAIS = input.pais.ToString();
                    usuario.UBIGEO = input.ubigeo;
                    usuario.DIRECCION_POSTAL = input.direccion_postal;
                    usuario.TELEFONO_FIJO = input.telefono_fijo;
                    usuario.TELEFONO_CELULAR = input.telefono_celular;
                    usuario.TELEFONO_EMERGENCIA = input.telefono_emergencia;
                    // FIN DE CAMPOS DE NO OBLIGATORIO
                    usuario.TIPO_USUARIO = "Externo";
                    usuario.GERENCIA = input.gerencia;
                    usuario.JEFATURA = input.jefatura;
                    usuario.ESTADO = input.estado;
                    usuario.EDITA_DATOS_PROYECTO = 0;
                    usuario.EDITA_DATOS_OBRA = 0;
                    // INICIO DE CAMPOS NO OLIGATORIOS
                    usuario.ACEPTA_CONDICIONES_GENERALES = 1;
                    usuario.ACEPTA_CONDICIONES_SUSCRIPCION = 1;
                    // FIN DE CAMPOS NO OBLIGATORIOS
                    usuario.USUARIO_CREACION      =      "GTIPROYECTOS-USU-
EXTERNO";
                    usuario.USUARIO_EDICION      =      "GTIPROYECTOS-USU-
EXTERNO";
                    usuario.FECHA_CREACION = DateTime.Now;
                    usuario.FECHA_EDICION = DateTime.Now;
                    db.USUARIO.Add(usuario);
                    db.SaveChanges();
```

```
        // GUARDAR ROL
        USUARIO_ROL ur = new USUARIO_ROL();
        ur.CODIGO_ROL = "ROL02";
        ur.CODIGO_USUARIO = input.codigo_usuario;
        ur.ESTADO = 1;
        ur.USUARIO_CREACION = "GTIPROYECTOS-USU-EXTERNO";
        ur.FECHA_CREACION = DateTime.Now;
        ur.USUARIO_EDICION = "GTIPROYECTOS-USU-EXTERNO";
        ur.FECHA_EDICION = DateTime.Now;
        db.USUARIO_ROL.Add(ur);
        db.SaveChanges();
        estado = "Ok";
        mensaje = "Se creó Usuario satisfactoriamente.";
      }
      catch (Exception ex)
      {
        estado = "Error";
        mensaje = "Comuníquese con Soporte Técnico.";
      }                }
else
{
  try
  {
    // EDITAR USUARIO
    USUARIO usuario = new USUARIO();
    usuario = db.USUARIO.Find(input.codigo_inicial);
    usuario.CODIGO_USUARIO = GenerarCodigo();
    usuario.NOMBRE = input.nombre;
    usuario.APELLIDO_PATERNO = input.ap_paterno;
    usuario.APELLIDO_MATERNO = input.ap_materno;
    usuario.CORREO_ELECTRONICO = input.correo_electronico;
    // CAMPOS NO OBLIGATORIO
    usuario.CODIGO_TIPO_DOCUMENTO = input.tipo_documento;
    usuario.NUMERO_DOCUMENTO = input.numero_documento;
    usuario.CODIGO_PAIS = input.pais;
    usuario.UBIGEO = input.ubigeo;
    usuario.DIRECCION_POSTAL = input.direccion_postal;
```

```
                    usuario.TELEFONO_FIJO = input.telefono_fijo;
                    usuario.TELEFONO_CELULAR = input.telefono_celular;
                    usuario.TELEFONO_EMERGENCIA = input.telefono_emergencia;
                    // FIN DE CAMPOS DE NO OBLIGATORIO
                    usuario.TIPO_USUARIO = "Externo";
                    usuario.GERENCIA = input.gerencia;
                    usuario.JEFATURA = input.jefatura;
                    usuario.ESTADO = input.estado;
                    usuario.USUARIO_EDICION = Session["Codigo"].ToString();
                    usuario.FECHA_EDICION = DateTime.Now;
                    db.USUARIO.Add(usuario);
                    db.SaveChanges();
                    estado = "Ok";
                    mensaje = "Se creó Usuario satisfactoriamente.";
                }
                catch (Exception ex)
                {
                    estado = "Error";
                    mensaje = mensaje + "El campo Gerencia es obligatorio.";
                }               }           }
            status.rpta_estado = estado;
            status.rpta_mensaje = mensaje;
            return Json(status);
        }
        catch (Exception ex)
        {
            return Json("Error");
        }
    }
    private string GenerarContrasenia()
    {
        Random rdn = new Random();
        string                                  caracteres                                  =
"abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890
%$#@";
        int longitud = caracteres.Length;
        char letra;
```

```
            int longitudContrasenia = 10;
            string contraseniaAleatoria = string.Empty;
            for (int i = 0; i < longitudContrasenia; i++)
            {
                letra = caracteres[rdn.Next(longitud)];
                contraseniaAleatoria += letra.ToString();
            }
            return contraseniaAleatoria;
        }
        public JsonResult EscribirLog(string motivo, string mensaje)
        {
            Log log = new Log();
            log.Add(motivo);
            log.Add(mensaje);
            return Json("Ok");
        }
        private class Log
        {
            private string path = Path.Combine(ConfigurationManager.AppSettings["RutaGuardarArchivos"].ToString() + "\\Logs\\UsuarioExterno-Solicitud");
            private string bringo = "\n";
            public void Add(string mensaje)
            {
                CreateDirectory();
                string nombre = GetNameFile();
                string cadena = "";
                cadena = DateTime.Now + " ==========> " + mensaje + Environment.NewLine;
                StreamWriter sw = new StreamWriter(path + "/" + nombre, true);
                sw.Write(cadena);
                sw.Close();
            }
            private void CreateDirectory()
            {
                try
                {
```

```
                if (!Directory.Exists(path))
                    Directory.CreateDirectory(path);
            }
            catch (DirectoryNotFoundException ex)
            {
                throw new Exception(ex.Message);
            }
        }
        private string GetNameFile()
        {
            string nombre = "";
            nombre = "log_" + DateTime.Now.Day + "_" + DateTime.Now.Month + "_" +
DateTime.Now.Year + ".txt";
            return nombre;
        }      }  }}
```

Registro y Administración de módulos del portal

```
using ProyectoTerceros.Models;
using ProyectoTerceros.Models.Seguridad;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
namespace ProyectoTerceros.Controllers.Seguridad
{
   public class ModuloController : Controller
   {
      private ProyectoTercerosEntities db = new ProyectoTercerosEntities();
      // GET: Modulo
      public ActionResult Index()
      {
         if (Session["Usuario"] == null)
         {
            return RedirectToAction("Index", "Login");
         }
         else
         {
```

```
            return View();
        }       }
    public JsonResult List()
    {
        List<ModuloQueryList> usuarioQuery = new List<ModuloQueryList>();
        List<MODULO> usuarios = db.MODULO.Where(p => p.ESTADO == 1 ||
p.ESTADO == 0).ToList();
        foreach (var usuario in usuarios)
        {
            ModuloQueryList uq = new ModuloQueryList();
            uq.CODIGO_MODULO = usuario.CODIGO_MODULO;
            uq.DESCRIPCION = usuario.DESCRIPCION;
            uq.ESTADO = usuario.ESTADO;
            usuarioQuery.Add(uq);
        }
        return Json(usuarioQuery);
    }
    // GET: Modulo/Editar/5
    [Route("/Modulo/Editar/{CODIGO_MODULO}")]
    public ActionResult Editar(string CODIGO_MODULO)
    {
        if (Session["Usuario"] == null)
        {
            return RedirectToAction("Index", "Login");
        }
        else
        {
            ViewBag.Codigo = CODIGO_MODULO;
            return View("Create");
        }       }
    // GET: Modulo/Create
    public ActionResult Create()
    {
        if (Session["Usuario"] == null)
        {
            return RedirectToAction("Index", "Login");
        }
```

```
    else
    {
        ViewBag.Codigo = "";
        return View();
    }       }
// POST: Modulo/Create
[HttpPost]
public ActionResult Create(FormCollection collection)
{
    if (Session["Usuario"] == null)
    {
        return RedirectToAction("Index", "Login");
    }
    else
    {
        try
        {
            // TODO: Add insert logic here
            string descripcion = collection["descripcion"];
            string codigo = collection["codigo"];
            string estado = collection["estado"];
            MODULO b = new MODULO();
            b.CODIGO_MODULO = codigo;
            b.DESCRIPCION = descripcion;
            b.ESTADO = Int32.Parse(estado);
            b.USUARIO_CREACION = "SISTEMAS";
            b.FECHA_CREACION = DateTime.Now;
            b.USUARIO_EDICION = "Sistemas";
            b.FECHA_EDICION = DateTime.Now;
            db.MODULO.Add(b);
            db.SaveChanges();
            return RedirectToAction("Index");
        }
        catch
        {
            return View();              }           }       }
```

```
public JsonResult Save(ModuloInput input)
{
    Status status = new Status();
    string codigo_entidad = "";
    string rpta_estado = "Ok";
    string rpta_mensaje = "";
    string descripcion = input.descripcion;
    string codigo = input.codigo;
    string estado = input.estado;
    string codigo_inicial = input.codigo_inicial;
    List<VistaQuery> vistas = input.vistas;
    if (rpta_estado == "Ok")
    {
        MODULO mod = new MODULO();
        mod = db.MODULO.Find(codigo_inicial);
        if (mod == null)
        {
            // GUARDAMOS EL MODULO
            MODULO b = new MODULO();
            b.CODIGO_MODULO = codigo;
            b.DESCRIPCION = descripcion;
            b.ESTADO = Int32.Parse(estado);
            b.USUARIO_CREACION = Session["Codigo"].ToString();
            b.FECHA_CREACION = DateTime.Now;
            b.USUARIO_EDICION = Session["Codigo"].ToString();
            b.FECHA_EDICION = DateTime.Now;
            db.MODULO.Add(b);
            db.SaveChanges();
            if (vistas.Count != 0)
            {
                // GUARDAMOS LAS VISTAS
                foreach (var vista in vistas)
                {
                    // PREGUNTAMOS EL CORRELATIVO
                    int correlativo = db.MODULO_OPCION.Count();
                    if (correlativo == 0)
                    {
```

```
                    correlativo = 1;
                }
                else
                {
                    correlativo = db.MODULO_OPCION.Max(p => p.ID) + 1;
                }
                MODULO_OPCION moduloOpcion = new MODULO_OPCION();
                moduloOpcion.ID = correlativo;
                moduloOpcion.CODIGO_MODULO = codigo;
                moduloOpcion.CODIGO_OPCION = vista.codigo_vista;
                moduloOpcion.ESTADO = 1;
                moduloOpcion.USUARIO_CREACION =
Session["Codigo"].ToString();
                moduloOpcion.USUARIO_EDICION =
Session["Codigo"].ToString();
                moduloOpcion.FECHA_CREACION = DateTime.Now;
                moduloOpcion.FECHA_EDICION = DateTime.Now;
                db.MODULO_OPCION.Add(moduloOpcion);
                db.SaveChanges();
            }            }
            codigo_entidad = codigo;
            rpta_estado = "Ok";
            rpta_mensaje = "Se guardó 'Módulo' satisfactoriamente.";
        }
        else
        {
            mod.CODIGO_MODULO = codigo;
            mod.DESCRIPCION = descripcion;
            mod.ESTADO = Int32.Parse(estado);
            mod.USUARIO_CREACION = Session["Codigo"].ToString();
            mod.FECHA_CREACION = DateTime.Now;
            mod.USUARIO_EDICION = Session["Codigo"].ToString();
            mod.FECHA_EDICION = DateTime.Now;
            db.SaveChanges();
            //CAMBIAMOS EL ESTADO A TODAS LAS VISTAS
SELECCIONADAS
```

```
            List<MODULO_OPCION> vistasMarcadas = new
List<MODULO_OPCION>();
            vistasMarcadas = db.MODULO_OPCION.Where(p =>
p.CODIGO_MODULO == codigo_inicial).ToList();
            // ACTUALIZAMOS A ESTADO = 0, INACTIVO
            foreach (var vistaMarcada in vistasMarcadas)
            {
               vistaMarcada.ESTADO = 0;
               vistaMarcada.CODIGO_MODULO = codigo;
               vistaMarcada.USUARIO_EDICION = Session["Codigo"].ToString();
               vistaMarcada.FECHA_EDICION = DateTime.Now;
               db.SaveChanges();
            }
            if (vistas.Count != 0)
            {
               foreach (var vista in vistas)
               {
                  List<MODULO_OPCION> opcionInd =
db.MODULO_OPCION.Where(p => p.CODIGO_MODULO == codigo &&
p.CODIGO_OPCION == vista.codigo_vista).ToList();
                  if (opcionInd.Count == 0)
                  {
                     // CREAR
                     int correlativo = db.MODULO_OPCION.Count();
                     if (correlativo == 0)
                     {
                        correlativo = 1;
                     }
                     else
                     {
                        correlativo = db.MODULO_OPCION.Max(p => p.ID) + 1;
                     }
                        MODULO_OPCION moduloOpcion = new
MODULO_OPCION();
                        moduloOpcion.ID = correlativo;
                        moduloOpcion.CODIGO_MODULO = codigo;
                        moduloOpcion.CODIGO_OPCION = vista.codigo_vista;
```

```
                        moduloOpcion.ESTADO = 1;
                        moduloOpcion.USUARIO_CREACION =
Session["Codigo"].ToString();
                        moduloOpcion.USUARIO_EDICION =
Session["Codigo"].ToString();
                        moduloOpcion.FECHA_CREACION = DateTime.Now;
                        moduloOpcion.FECHA_EDICION = DateTime.Now;
                        db.MODULO_OPCION.Add(moduloOpcion);
                        db.SaveChanges();
                    }
                    else
                    {
                      // EDITAR
                      foreach (var vistaMarcada in opcionInd)
                      {
                        vistaMarcada.ESTADO = 1;
                        vistaMarcada.USUARIO_EDICION =
Session["Codigo"].ToString();
                        vistaMarcada.FECHA_EDICION = DateTime.Now;
                        db.SaveChanges();
                      }                  }                  }                }
                codigo_entidad = codigo;
                rpta_estado = "Ok";
                rpta_mensaje = "Se editó 'Módulo' satisfactoriamente.";
              }          }
            status.rpta_estado = rpta_estado;
            status.rpta_mensaje = rpta_mensaje;
            status.codigo_entidad = codigo;
            return Json(status);
        }
        public JsonResult Leer(string codigo)
        {
            ModuloInd rq = new ModuloInd();
            MODULO b = db.MODULO.Find(codigo);
            if (b != null)
            {
              if (b.ESTADO == 0 || b.ESTADO == 1)
```

```
            {
                rq.CODIGO_MODULO = codigo;
                rq.DESCRIPCION = b.DESCRIPCION;
                rq.ESTADO = Convert.ToInt32(b.ESTADO);
                List<VistaQuery> vistas = new List<VistaQuery>();
                List<MODULO_OPCION> moquery = db.MODULO_OPCION.Where(p
=> p.CODIGO_MODULO == codigo && p.ESTADO == 1).ToList();
                foreach (var mquery in moquery)
                {
                    VistaQuery vista = new VistaQuery();
                    vista.codigo_vista = mquery.CODIGO_OPCION;
                    vista.descripcion = mquery.OPCION.DESCRIPCION;
                    vista.menu_principal = mquery.OPCION.MENU_PRINCIPAL;
                    vistas.Add(vista);
                }
                rq.vistas = vistas;
                rq.codigo_entidad = codigo;
                rq.rpta_estado = "Ok";
                rq.rpta_mensaje = "Módulo existe.";
            }
             else
             {
                rq.codigo_entidad = codigo;
                rq.rpta_estado = "Ok";
                rq.rpta_mensaje = "Módulo no existe.";
             }          }
          else
          {
             rq.codigo_entidad = "";
             rq.rpta_estado = "Error";
             rq.rpta_mensaje = "Módulo no existe.";
          }
          return Json(rq);
       }
       [HttpGet]
       public JsonResult Listar()
       {
```

```
            IList<MODULO> modulos = db.MODULO.ToList();
            List<ModuloQuery> mq = new List<ModuloQuery>();
            foreach (var md in modulos)
            {
                ModuloQuery mdq = new ModuloQuery();
                mdq.codigo_modulo = md.CODIGO_MODULO;
                mdq.descripcion = md.DESCRIPCION;
                mq.Add(mdq);
            }
            return Json(mq, JsonRequestBehavior.AllowGet);
        }
        [HttpGet]
        public JsonResult ModuloListarVista()
        {
            IList<MODULO> modulos = db.MODULO.Where(p=> p.ESTADO ==
1).ToList();
            List<ModuloQuery> mq = new List<ModuloQuery>();
            foreach (var md in modulos)
            {
                ModuloQuery mdq = new ModuloQuery();
                mdq.codigo_modulo = md.CODIGO_MODULO;
                mdq.descripcion = md.DESCRIPCION;
                List<VistaQuery> vistas = new List<VistaQuery>();
                List<MODULO_OPCION> moquery = db.MODULO_OPCION.Where(p =>
p.CODIGO_MODULO == md.CODIGO_MODULO && p.ESTADO == 1).ToList();
                foreach (var mquery in moquery)
                {
                    VistaQuery vista = new VistaQuery();
                    vista.codigo_vista = mquery.CODIGO_OPCION;
                    vista.descripcion = mquery.OPCION.DESCRIPCION;
                    vista.menu_principal = mquery.OPCION.MENU_PRINCIPAL;
                    vistas.Add(vista);
                }
                mdq.vistas = vistas;
                mq.Add(mdq);
            }
            return Json(mq, JsonRequestBehavior.AllowGet);        }
```

```
// POST: Modulo/Edit/5
[HttpPost]
public ActionResult Edit(int id, FormCollection collection)
{
   if (Session["Usuario"] == null)
   {
      return RedirectToAction("Index", "Login");
   }
   else
   {            try
      {
         return RedirectToAction("Index");
      }
      catch
      {
         return View();
      }          }       }
[HttpPost]
public ActionResult Eliminar(string codigo_modulo)
{
   MODULO usuario = db.MODULO.Find(codigo_modulo);
   usuario.ESTADO = 2;
   usuario.FECHA_EDICION = DateTime.Now;
   usuario.USUARIO_EDICION = Session["Codigo"].ToString();
   db.SaveChanges();
   return Json("Ok");
}
```

Registro y Administración de Perfiles

```
using ProyectoTerceros.Models;
using ProyectoTerceros.Models.Seguridad;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
```

```
namespace ProyectoTerceros.Controllers.Seguridad
{
   public class RolController : Controller
   {
      private ProyectoTercerosEntities db = new ProyectoTercerosEntities();
      // GET: Rol
      public ActionResult Index()
      {
         if (Session["Usuario"] == null)
         {
            return RedirectToAction("Index", "Login");
         }
         else
         {
            return View();
         }       }
      public JsonResult List()
      {
         List<RolQueryList> usuarioQuery = new List<RolQueryList>();
         List<ROL> usuarios = db.ROL.Where(p => p.ESTADO == 1 || p.ESTADO ==
0).ToList();
         foreach (var usuario in usuarios)
         {
            RolQueryList uq = new RolQueryList();
            uq.CODIGO_ROL = usuario.CODIGO_ROL;
            uq.DESCRIPCION = usuario.DESCRIPCION;
            uq.ESTADO = usuario.ESTADO;
              usuarioQuery.Add(uq);
         }
         return Json(usuarioQuery);
      }
      public JsonResult ListTipoCartaRespuesta()
      {
         List<TipoCartaRespuestaQuery>          usuarioQuery          =          new
List<TipoCartaRespuestaQuery>();
```

```
        List<TIPO_CARTA_RESPUESTA> usuarios =
db.TIPO_CARTA_RESPUESTA.Where(p => p.ESTADO == 0).ToList();
        foreach (var usuario in usuarios)
        {
            TipoCartaRespuestaQuery uq = new TipoCartaRespuestaQuery();
            uq.codigo = usuario.CODIGO_CARTA_RESPUESTA;
            uq.descripcion = usuario.DESCRIPCION_CARTA_RESPUESTA;
            usuarioQuery.Add(uq);
        }
        return Json(usuarioQuery, JsonRequestBehavior.AllowGet);
    }
    [Route("/Rol/Editar/{CODIGO_ROL}")]
    public ActionResult Editar(string CODIGO_ROL)
    {
        if (Session["Usuario"] == null)
        {
            return RedirectToAction("Index", "Login");
        }
        else
        {
            ViewBag.Codigo = CODIGO_ROL;
            return View("Create");
        }       }
    // GET: Rol/Create
    public ActionResult Create()
    {
        if (Session["Usuario"] == null)
        {
            return RedirectToAction("Index", "Login");
        }
        else
        {
            ViewBag.Codigo = "";
            return View();
        }       }
    public JsonResult Save(FormCollection collection)
    {
```

```
    string descripcion = collection["descripcion"];
    string codigo = collection["codigo"];
    string estado = collection["estado"];
    string codigo_inicial = collection["codigo_inicial"];
    IList<char> str = collection["menus"].ToList();
    ROL mod = new ROL();
    mod = db.ROL.Find(codigo_inicial);
    if (mod == null)
    {
        ROL b = new ROL();
        b.CODIGO_ROL = codigo;
        b.DESCRIPCION = descripcion;
        b.ESTADO = Int32.Parse(estado);
        b.USUARIO_CREACION = Session["Codigo"].ToString();
        b.FECHA_CREACION = DateTime.Now;
        b.USUARIO_EDICION = Session["Codigo"].ToString();
        b.FECHA_EDICION = DateTime.Now;
        db.ROL.Add(b);
    }
    else
    {
        mod.CODIGO_ROL = codigo;
        mod.DESCRIPCION = descripcion;
        mod.ESTADO = Int32.Parse(estado);
        mod.USUARIO_CREACION = Session["Codigo"].ToString();
        mod.FECHA_CREACION = DateTime.Now;
        mod.USUARIO_EDICION = Session["Codigo"].ToString();
        mod.FECHA_EDICION = DateTime.Now;
    }
    db.SaveChanges();
    return Json("Ok");
}
public JsonResult Guardar(RolInput input)
{
    Status status = new Status();
    string codigo_entidad = "";
    string rpta_estado = "Ok";
```

```
string rpta_mensaje = "";
string descripcion = input.descripcion;
string codigo = input.codigo_rol;
int estado = input.estado;
string codigo_inicial = input.codigo_inicial;
IList<ModuloRolInput> modulo = input.menus;
if (rpta_estado == "Ok")
{
   ROL mod = new ROL();
   mod = db.ROL.Find(codigo_inicial);
   if (mod == null)
   {
      ROL b = new ROL();
      b.CODIGO_ROL = codigo;
      b.DESCRIPCION = descripcion;
      b.ESTADO = estado;
      b.USUARIO_CREACION = Session["Codigo"].ToString();
      b.FECHA_CREACION = DateTime.Now;
      b.USUARIO_EDICION = Session["Codigo"].ToString();
      b.FECHA_EDICION = DateTime.Now;
      db.ROL.Add(b);
      db.SaveChanges();
      if (modulo.Count != 0)
      {
         foreach (var m in modulo)
         {
            MODULO_ROL mr = new MODULO_ROL();
            mr.ID = db.MODULO_ROL.Count() + 1;
            mr.CODIGO_MODULO_ROL = (db.MODULO_ROL.Count() +
1).ToString();
            mr.CODIGO_MODULO = m.codigo_modulo;
            mr.CODIGO_ROL = codigo;
            mr.ESTADO = 1;
            mr.USUARIO_CREACION = Session["Codigo"].ToString();
            mr.USUARIO_EDICION = Session["Codigo"].ToString();
            mr.FECHA_CREACION = DateTime.Now;
            mr.FECHA_EDICION = DateTime.Now;
```

```
                db.MODULO_ROL.Add(mr);
                db.SaveChanges();
            }
            if (input.tieneRevision == true)
            {
                int idRevision = 0;
                int cantidadDatos = db.OPCION_REVISION_DETALLE.Count();
                if (cantidadDatos == 0)
                {
                    idRevision = 1;
                }
                else if (cantidadDatos != 0)
                {
                    idRevision = db.OPCION_REVISION_DETALLE.Max(p => p.ID)
+ 1;
                }
                int esGestorComercial = 0;
                int esSupervisorTecnico = 0;
                if (input.esGestorComercial)
                {                   esGestorComercial = 1;                  }
                if (input.esSupervisorTecnico)
                {                   esSupervisorTecnico = 1;                    }
                int esEmisionCartas = 0;
                if (input.tieneEmisionCarta == true)
                {
                    if (input.esEmisionCartas)
                    {                   esEmisionCartas = 1;                    }
                    else
                    {                   esEmisionCartas = 0;                    }
                }
                else
                {                   esEmisionCartas = 0;                }
                OPCION_REVISION_DETALLE     oprevdetalle     =     new
OPCION_REVISION_DETALLE();
                oprevdetalle.ID = idRevision;
                oprevdetalle.CODIGO_ROL = codigo;
                oprevdetalle.ES_GESTOR_COMERCIAL = esGestorComercial;
```

```
                    oprevdetalle.ES_SUPERVISOR_TECNICO = esSupervisorTecnico;
                    oprevdetalle.EMITE_CARTA = esEmisionCartas;
                    oprevdetalle.ESTADO = 1;
                    oprevdetalle.USUARIO_CREACION = Session["Codigo"].ToString();
                    oprevdetalle.USUARIO_EDICION = Session["Codigo"].ToString();
                    oprevdetalle.FECHA_CREACION = DateTime.Now;
                    oprevdetalle.FECHA_EDICION = DateTime.Now;
                    db.OPCION_REVISION_DETALLE.Add(oprevdetalle);
                    db.SaveChanges();
                }
                else if(input.tieneRevision == false)
                {
                    if (input.tieneEmisionCarta == true)
                    {
                        int idRevision = 0;
                    int cantidadDatos = db.OPCION_REVISION_DETALLE.Count();
                    if (cantidadDatos == 0)
                    {                       idRevision = 1;                     }
                    else if (cantidadDatos != 0)
                    {                                                       idRevision =
db.OPCION_REVISION_DETALLE.Max(p => p.ID) + 1;                      }
                    int esGestorComercial = 0;
                    int esSupervisorTecnico = 0;
                    int esEmisionCartas = 0;
                        if (input.esEmisionCartas)
                        {                       esEmisionCartas = 1;                    }
                        else
                        {                       esEmisionCartas = 0;                    }
                    OPCION_REVISION_DETALLE      oprevdetalle      =      new
OPCION_REVISION_DETALLE();
                    oprevdetalle.ID = idRevision;
                    oprevdetalle.CODIGO_ROL = codigo;
                    oprevdetalle.ES_GESTOR_COMERCIAL = esGestorComercial;
                    oprevdetalle.ES_SUPERVISOR_TECNICO = esSupervisorTecnico;
                    oprevdetalle.EMITE_CARTA = esEmisionCartas;
                    oprevdetalle.ESTADO = 1;
                    oprevdetalle.USUARIO_CREACION = Session["Codigo"].ToString();
```

```
                    oprevdetalle.USUARIO_EDICION = Session["Codigo"].ToString();
                    oprevdetalle.FECHA_CREACION = DateTime.Now;
                    oprevdetalle.FECHA_EDICION = DateTime.Now;
                    db.OPCION_REVISION_DETALLE.Add(oprevdetalle);
                    db.SaveChanges();
                    }                    }                }
                codigo_entidad = codigo;
                rpta_estado = "Ok";
                rpta_mensaje = "Se guardó 'Rol' satisfactoriamente.";
            }
            else
            {
                mod.CODIGO_ROL = codigo;
                mod.DESCRIPCION = descripcion;
                mod.ESTADO = estado;
                mod.USUARIO_EDICION = Session["Codigo"].ToString();
                mod.FECHA_EDICION = DateTime.Now;
                db.SaveChanges();
                MODULO_ROL r = new MODULO_ROL();
                IList<MODULO_ROL> modulorol = db.MODULO_ROL.Where(p =>
p.CODIGO_ROL == codigo_inicial).ToList();
                foreach (var mdrol in modulorol)
                {
                    mdrol.ESTADO = 0;
                    mdrol.CODIGO_ROL = codigo;
                    mdrol.USUARIO_EDICION = Session["Codigo"].ToString();
                    mdrol.FECHA_EDICION = DateTime.Now;
                    db.SaveChanges();
                }
                if (modulo.Count != 0)
                {
                    try
                    {
                        foreach (var m in modulo)
                        {
```

```
IList<MODULO_ROL> modulorolind = db.MODULO_ROL.Where(p => p.CODIGO_ROL == codigo && p.CODIGO_MODULO == m.codigo_modulo).ToList();
                    if (modulorolind.Count == 0)
                    {
                        MODULO_ROL mr = new MODULO_ROL();
                        mr.ID = db.MODULO_ROL.Count() + 1;
                        mr.CODIGO_MODULO_ROL = (db.MODULO_ROL.Count() + 1).ToString();
                        mr.CODIGO_MODULO = m.codigo_modulo;
                        mr.CODIGO_ROL = codigo;
                        mr.ESTADO = 1;
                        mr.USUARIO_CREACION = Session["Codigo"].ToString();
                        mr.USUARIO_EDICION = Session["Codigo"].ToString();
                        mr.FECHA_CREACION = DateTime.Now;
                        mr.FECHA_EDICION = DateTime.Now;
                        db.MODULO_ROL.Add(mr);
                        db.SaveChanges();
                    }
                    else
                    {
                        foreach (var mrol in modulorolind)
                        {
                            mrol.ESTADO = 1;
                            mrol.USUARIO_EDICION = Session["Codigo"].ToString();
                            mrol.FECHA_EDICION = DateTime.Now;
                            db.SaveChanges();
                        }                    }                    }
                codigo_entidad = codigo;
                rpta_estado = "Ok";
                rpta_mensaje = "Se editó 'Rol' satisfactoriamente.";
            }
            catch (Exception ex)
            {                    throw;                }
            if (input.tieneRevision == true)
            {
                int idRevision = 0;
```

```
int cantidadDatos = db.OPCION_REVISION_DETALLE.Count();
if (cantidadDatos == 0)
{ idRevision = 1; }
else if (cantidadDatos != 0)
{ idRevision = db.OPCION_REVISION_DETALLE.Max(p => p.ID) + 1; }
int esGestorComercial = 0;
int esSupervisorTecnico = 0;
if (input.esGestorComercial)
{ esGestorComercial = 1; }
if (input.esSupervisorTecnico)
{ esSupervisorTecnico = 1; }
int esEmisionCartas = 0;
if (input.tieneEmisionCarta == true)
{
  if (input.esEmisionCartas)
  { esEmisionCartas = 1; }
  else
  { esEmisionCartas = 0; }
}
else
{ esEmisionCartas = 0; }
// OBTENER DATOS GUARDADOS EN LA DB
List<OPCION_REVISION_DETALLE> det = db.OPCION_REVISION_DETALLE.Where(p => p.CODIGO_ROL == codigo).ToList();
foreach (var d in det)
{
  d.ESTADO = 0;
  d.USUARIO_EDICION = Session["Codigo"].ToString();
  d.FECHA_EDICION = DateTime.Now;
  db.SaveChanges();
}
// CONSULTAR SI EXISTE UN REGISTRO
```

```
                List<OPCION_REVISION_DETALLE> detalle = db.OPCION_REVISION_DETALLE.Where(p => p.CODIGO_ROL == codigo && p.ES_GESTOR_COMERCIAL == esGestorComercial && p.ES_SUPERVISOR_TECNICO == esSupervisorTecnico && p.EMITE_CARTA == esEmisionCartas).ToList();
                if (detalle.Count != 0)
                {
                    foreach (var deta in detalle)
                    {
                        deta.ESTADO = 1;
                        deta.ES_GESTOR_COMERCIAL = esGestorComercial;
                        deta.ES_SUPERVISOR_TECNICO = esSupervisorTecnico;
                        deta.EMITE_CARTA = esEmisionCartas;
                        deta.USUARIO_EDICION = Session["Codigo"].ToString();
                        deta.FECHA_EDICION = DateTime.Now;
                        db.SaveChanges();
                    }                    }
                else
                {
                    OPCION_REVISION_DETALLE oprevdetalle = new OPCION_REVISION_DETALLE();
                    oprevdetalle.ID = idRevision;
                    oprevdetalle.CODIGO_ROL = codigo;
                    oprevdetalle.ES_GESTOR_COMERCIAL = esGestorComercial;
                    oprevdetalle.ES_SUPERVISOR_TECNICO = esSupervisorTecnico;
                    oprevdetalle.EMITE_CARTA = esEmisionCartas;
                    oprevdetalle.ESTADO = 1;
                    oprevdetalle.USUARIO_CREACION = Session["Codigo"].ToString();
                    oprevdetalle.USUARIO_EDICION = Session["Codigo"].ToString();
                    oprevdetalle.FECHA_CREACION = DateTime.Now;
                    oprevdetalle.FECHA_EDICION = DateTime.Now;
                    db.OPCION_REVISION_DETALLE.Add(oprevdetalle);
                    db.SaveChanges();
                }                }
            else if (input.tieneRevision == false)
            {
```

```
                List<OPCION_REVISION_DETALLE> det = db.OPCION_REVISION_DETALLE.Where(p => p.CODIGO_ROL == codigo).ToList();
                int esEmisionCartas = 0;
                int esGestorComercial = 0;
                int esSupervisorTecnico = 0;
                foreach (var deta in det)
                {
                    deta.ESTADO = 0;
                    deta.USUARIO_EDICION = Session["Codigo"].ToString();
                    deta.FECHA_EDICION = DateTime.Now;
                    db.SaveChanges();
                }
                if (input.tieneEmisionCarta == true)
                {
                    if (input.esEmisionCartas)
                    {                       esEmisionCartas = 1;                    }
                    else
                    {                       esEmisionCartas = 0;                    }
                    int idRevision = 0;
                    int cantidadDatos = db.OPCION_REVISION_DETALLE.Count();
                    if (cantidadDatos == 0)
                    {                       idRevision = 1;                     }
                    else if (cantidadDatos != 0)
                    {                                                   idRevision = db.OPCION_REVISION_DETALLE.Max(p => p.ID) + 1;                     }
                    List<OPCION_REVISION_DETALLE> detalle = db.OPCION_REVISION_DETALLE.Where(p => p.CODIGO_ROL == codigo && p.ES_GESTOR_COMERCIAL == esGestorComercial && p.ES_SUPERVISOR_TECNICO == esSupervisorTecnico && p.EMITE_CARTA == esEmisionCartas).ToList();
                    if (detalle.Count != 0)
                    {
                        foreach (var deta in detalle)
                        {
                            deta.ESTADO = 1;
                            deta.ES_GESTOR_COMERCIAL = esGestorComercial;
```

```
                    deta.ES_SUPERVISOR_TECNICO = esSupervisorTecnico;
                    deta.EMITE_CARTA = esEmisionCartas;
                    deta.USUARIO_EDICION = Session["Codigo"].ToString();
                    deta.FECHA_EDICION = DateTime.Now;
                    db.SaveChanges();
                }                    }
                else
                {
                    OPCION_REVISION_DETALLE oprevdetalle = new
OPCION_REVISION_DETALLE();
                    oprevdetalle.ID = idRevision;
                    oprevdetalle.CODIGO_ROL = codigo;
                    oprevdetalle.ES_GESTOR_COMERCIAL = esGestorComercial;
                    oprevdetalle.ES_SUPERVISOR_TECNICO =
esSupervisorTecnico;
                    oprevdetalle.EMITE_CARTA = esEmisionCartas;
                    oprevdetalle.ESTADO = 1;
                    oprevdetalle.USUARIO_CREACION =
Session["Codigo"].ToString();
                    oprevdetalle.USUARIO_EDICION =
Session["Codigo"].ToString();
                    oprevdetalle.FECHA_CREACION = DateTime.Now;
                    oprevdetalle.FECHA_EDICION = DateTime.Now;
                    db.OPCION_REVISION_DETALLE.Add(oprevdetalle);
                    db.SaveChanges();
                }                    }                }
            }            }          }
        status.rpta_estado = rpta_estado;
        status.rpta_mensaje = rpta_mensaje;
        status.codigo_entidad = codigo;
        return Json(status);
    }
    // [HttpGet]
    public JsonResult Leer(string codigo)
    {
        RolQuery rq = new RolQuery();
```

```
ROL b = db.ROL.Find(codigo);
if (b != null)
{
  if (b.ESTADO == 0 || b.ESTADO == 1)
  {
    rq.codigo_rol = b.CODIGO_ROL;
    rq.descripcion = b.DESCRIPCION;
    rq.estado = Convert.ToInt32(b.ESTADO);
    List<MODULO_ROL> mr = db.MODULO_ROL.Where(p => p.CODIGO_ROL == codigo && p.ESTADO == 1).ToList();
    //List<ROL_TIPO_CARTA_RESPUESTA> crr = db.ROL_TIPO_CARTA_RESPUESTA.Where(p => p.CODIGO_ROL == codigo && p.ESTADO == 1).ToList();
    List<ModuloRolQuery> mrquery = new List<ModuloRolQuery>();
    List<TipoCartaRespuestaQuery> cquery = new List<TipoCartaRespuestaQuery>();
    foreach (var m in mr)
    {
      ModuloRolQuery mrq = new ModuloRolQuery();
      mrq.codigo_modulo = m.CODIGO_MODULO;
      mrquery.Add(mrq);
    }
    // OBTENER DATOS GUARDADOS EN LA DB
    List<OPCION_REVISION_DETALLE> det = db.OPCION_REVISION_DETALLE.Where(p => p.CODIGO_ROL == codigo && p.ESTADO == 1).ToList();
    foreach (var d in det)
    {
      if (d.ES_GESTOR_COMERCIAL == 1)
      { rq.esGestorComercial = true; }
      else if (d.ES_GESTOR_COMERCIAL == 0)
      { rq.esGestorComercial = false; }
      if (d.ES_SUPERVISOR_TECNICO == 1)
      { rq.esSupervisorTecnico = true; }
      else if (d.ES_SUPERVISOR_TECNICO == 0)
      { rq.esSupervisorTecnico = false; }
```

```
                if (d.ES_GESTOR_COMERCIAL == 1 ||
d.ES_SUPERVISOR_TECNICO == 1)
                { rq.tieneRevision = true; }
                else
                { rq.tieneRevision = false; }
                if (d.EMITE_CARTA == 1)
                { rq.esEmisionCartas = true;
rq.tieneEmisionCarta = true; } break; }
            rq.modulos = mrquery;
            rq.codigo_entidad = codigo;
            rq.rpta_estado = "Ok";
            rq.rpta_mensaje = "Rol existe.";
          }
          else
          {
            rq.codigo_entidad = codigo;
            rq.rpta_estado = "Error";
            rq.rpta_mensaje = "Rol no existe.";
          }      }
        else
        {
          rq.codigo_entidad = "";
          rq.rpta_estado = "Error";
          rq.rpta_mensaje = "Rol no existe.";
        }
        return Json(rq);
      }
      public JsonResult ListarModulo()
      {
        List<MODULO> modulos = db.MODULO.ToList();
        return Json(modulos);
      }
      [HttpGet]
      public JsonResult ListRolWithEmisionCarta()
      {
        IList<RolQuery> roles = new List<RolQuery>();
```

```
        List<OPCION_REVISION_DETALLE> opciones = db.OPCION_REVISION_DETALLE.Where(p => p.ESTADO == 1 && p.EMITE_CARTA == 1).ToList();
        foreach (var op in opciones)
        {
            ROL rol = db.ROL.Find(op.CODIGO_ROL);
            RolQuery rolQuery = new RolQuery();
            rolQuery.descripcion = rol.DESCRIPCION;
            rolQuery.codigo_rol = rol.CODIGO_ROL;
            roles.Add(rolQuery);
        }
        return Json(roles, JsonRequestBehavior.AllowGet);
    }
    [HttpGet]
    public JsonResult Listar()
    {
        IList<ROL> modulos = db.ROL.ToList();
        List<RolQueryforUsuario> mq = new List<RolQueryforUsuario>();
        foreach (var md in modulos)
        {
            RolQueryforUsuario mdq = new RolQueryforUsuario();
            mdq.codigo_rol = md.CODIGO_ROL;
            mdq.descripcion = md.DESCRIPCION;
            mq.Add(mdq);
        }
        return Json(mq, JsonRequestBehavior.AllowGet);
    }
    [HttpGet]
    public JsonResult RolListar()
    {
        IList<ROL> roles = db.ROL.Where(p => p.ESTADO == 1).ToList();
        List<RolQueryforUsuario> rolList = new List<RolQueryforUsuario>();
        foreach (var rol in roles)
        {
            RolQueryforUsuario rolElement = new RolQueryforUsuario();
            rolElement.codigo_rol = rol.CODIGO_ROL;
            rolElement.descripcion = rol.DESCRIPCION;
```

```
            IList<MODULO_ROL> modulosRol = db.MODULO_ROL.Where(p =>
p.ESTADO == 1 && p.CODIGO_ROL == rol.CODIGO_ROL).ToList();
            List<ModuloQuery> modulos = new List<ModuloQuery>();
            foreach (var moduloRol in modulosRol)
            {
               ModuloQuery m = new ModuloQuery();
               m.codigo_modulo = moduloRol.CODIGO_MODULO;
               m.descripcion = moduloRol.MODULO.DESCRIPCION;
               List<VistaQuery> vistas = new List<VistaQuery>();
               List<MODULO_OPCION> moquery = db.MODULO_OPCION.Where(p
=> p.CODIGO_MODULO == moduloRol.CODIGO_MODULO && p.ESTADO ==
1).ToList();
                  foreach (var mquery in moquery)
                  {
                     VistaQuery vista = new VistaQuery();
                     vista.codigo_vista = mquery.CODIGO_OPCION;
                     vista.descripcion = mquery.OPCION.DESCRIPCION;
                     vista.menu_principal = mquery.OPCION.MENU_PRINCIPAL;
                    vistas.Add(vista);
                  }
                  m.vistas = vistas;
                  modulos.Add(m);
               }
               rolElement.modulos = modulos;
               rolList.Add(rolElement);
            }
            return Json(rolList, JsonRequestBehavior.AllowGet);
         }
         [HttpPost]
         public ActionResult Eliminar(string codigo_rol)
         {
            ROL usuario = db.ROL.Find(codigo_rol);
            usuario.ESTADO = 2;
            usuario.FECHA_EDICION = DateTime.Now;
            usuario.USUARIO_EDICION = Session["Codigo"].ToString();
```

```
            db.SaveChanges();
            return Json("Ok");
        }
```

Registro y Administración de los Tipos de Sistemas Eléctricos

```
using ProyectoTerceros.Models;
using ProyectoTerceros.Models.Administracion;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
namespace ProyectoTerceros.Controllers.Administracion
{
   public class TipoSistemaElectricoController : Controller
   {
      private ProyectoTercerosEntities db = new ProyectoTercerosEntities();
      // GET: TipoSistemaElectrico
      public ActionResult Index()
      {
         if (Session["Usuario"] == null)
         {
            return RedirectToAction("Index", "Login");
         }
         else
         {
            return View();          }       }
      public JsonResult List()
      {
         List<TipoSistemaElectricoQuery> usuarioQuery = new
List<TipoSistemaElectricoQuery>();
         List<TIPO_SISTEMA_ELECTRICO> usuarios =
db.TIPO_SISTEMA_ELECTRICO.Where(p => p.ESTADO == 1 || p.ESTADO ==
0).ToList();

         foreach (var usuario in usuarios)
         {
            TipoSistemaElectricoQuery uq = new TipoSistemaElectricoQuery();
            uq.CODIGO_TIPO_SISTEMA_ELECTRICO =
usuario.CODIGO_TIPO_SISTEMA_ELECTRICO;
            uq.DESCRIPCION = usuario.DESCRIPCION;
            uq.ESTADO = usuario.ESTADO;
            usuarioQuery.Add(uq);
         }
         return Json(usuarioQuery);       }

[Route("/TipoSistemaElectrico/Editar/{CODIGO_TIPO_SISTEMA_ELECTRICO}")]
      public ActionResult Editar(string CODIGO_TIPO_SISTEMA_ELECTRICO)
      {
         if (Session["Usuario"] == null)
         {              return RedirectToAction("Index", "Login");          }
         else
         {
            ViewBag.Codigo = CODIGO_TIPO_SISTEMA_ELECTRICO;
```

```
            return View("Create");
        }       }
    [HttpPost]
    public ActionResult Eliminar(string codigo_tipo_sistema_electrico)
    {
        TIPO_SISTEMA_ELECTRICO usuario =
db.TIPO_SISTEMA_ELECTRICO.Find(codigo_tipo_sistema_electrico);
        usuario.ESTADO = 2;
        usuario.FECHA_EDICION = DateTime.Now;
        usuario.USUARIO_EDICION = Session["Codigo"].ToString();
        db.SaveChanges();
        return Json("Ok");
    }
// GET: TipoSistemaElectrico/Create
    public ActionResult Create()
    {
        if (Session["Usuario"] == null)
        {
            return RedirectToAction("Index", "Login");
        }
        else
        {           ViewBag.Codigo = "";           return View();          }
    }

    // POST: TipoSistemaElectrico/Create
    [HttpPost]
    public ActionResult Create(FormCollection collection)
    {
        if (Session["Usuario"] == null)
        {
            return RedirectToAction("Index", "Login");
        }
        else
        {
            try
            {               return RedirectToAction("Index");           }
            catch
            {               return View();          }
        }       }
    public JsonResult Save(FormCollection collection)
    {
        string descripcion = collection["descripcion"];
        string codigo = collection["codigo"];
        string estado = collection["estado"];
        string codigo_inicial = collection["codigo_inicial"];
        TIPO_SISTEMA_ELECTRICO mod = new TIPO_SISTEMA_ELECTRICO();
        mod = db.TIPO_SISTEMA_ELECTRICO.Find(codigo_inicial);
        if (mod == null)
        {
            TIPO_SISTEMA_ELECTRICO b = new TIPO_SISTEMA_ELECTRICO();
            b.CODIGO_TIPO_SISTEMA_ELECTRICO = codigo;
            b.DESCRIPCION = descripcion;
            b.ESTADO = Int32.Parse(estado);
            b.USUARIO_CREACION = Session["Codigo"].ToString();
            b.FECHA_CREACION = DateTime.Now;
```

```
            b.USUARIO_EDICION = Session["Codigo"].ToString();
            b.FECHA_EDICION = DateTime.Now;
            db.TIPO_SISTEMA_ELECTRICO.Add(b);
        }
        else
        {
            mod.CODIGO_TIPO_SISTEMA_ELECTRICO = codigo;
            mod.DESCRIPCION = descripcion;
            mod.ESTADO = Int32.Parse(estado);
            mod.USUARIO_EDICION = Session["Codigo"].ToString();
            mod.FECHA_EDICION = DateTime.Now;
        }
        db.SaveChanges();
        return Json("Ok");
    }
    public JsonResult Leer(string codigo)
    {
        TIPO_SISTEMA_ELECTRICO b =
db.TIPO_SISTEMA_ELECTRICO.Find(codigo);
        TipoSistemaElectricoInd tseInd = new TipoSistemaElectricoInd();
        tseInd.CODIGO_TIPO_SISTEMA_ELECTRICO =
b.CODIGO_TIPO_SISTEMA_ELECTRICO;
        tseInd.DESCRIPCION = b.DESCRIPCION;
        tseInd.ESTADO = b.ESTADO;
        return Json(tseInd);
    }
    public JsonResult GetTipoSistemaElectrico()
    {
        List<TipoSistemaElectricoGet> listTSE = new
List<TipoSistemaElectricoGet>();

        IList<TIPO_SISTEMA_ELECTRICO> tsqList =
db.TIPO_SISTEMA_ELECTRICO.Where(p => p.ESTADO == 1).ToList();

        foreach (var tsq in tsqList) {
            TipoSistemaElectricoGet tse = new TipoSistemaElectricoGet();
            tse.codigo_tipo_sistema_electrico =
tsq.CODIGO_TIPO_SISTEMA_ELECTRICO;
            tse.descripcion_tipo_sistema_electrico = tsq.DESCRIPCION;
            listTSE.Add(tse);
        }
        return Json(listTSE);
    }
    public ActionResult Edit(int id)
    {
        if (Session["Usuario"] == null)
        {           return RedirectToAction("Index", "Login");          }
        else
        {           return View();          }       }

        }
}
```

Registro y Administración de los Tipos de Solicitudes

```
using ProyectoTerceros.Models;
using ProyectoTerceros.Models.Administracion;
using ProyectoTerceros.Models.Seguridad;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
namespace ProyectoTerceros.Controllers.Administracion
{
   public class TipoSolicitudController : Controller
   {
      private ProyectoTercerosEntities db = new ProyectoTercerosEntities();
      // GET: TipoSolicitud
      public ActionResult Index()
      {
         if (Session["Usuario"] == null)
         {              return RedirectToAction("Index", "Login");
         }
         else
         {              return View();
         }       }
      public JsonResult List()
      {
         List<TipoSolicitudQueryList>              usuarioQuery              =              new
List<TipoSolicitudQueryList>();
         List<TIPO_SOLICITUD>  usuarios  =  db.TIPO_SOLICITUD.Where(p  =>
p.ESTADO == 1 || p.ESTADO == 0).ToList();
         foreach (var usuario in usuarios)
         {
            TipoSolicitudQueryList uq = new TipoSolicitudQueryList();
            uq.CODIGO_TIPO_SOLICITUD = usuario.CODIGO_TIPO_SOLICITUD;
            uq.DESCRIPCION = usuario.DESCRIPCION;
            uq.ESTADO = usuario.ESTADO;
            uq.DETALLE = usuario.DETALLE;
            usuarioQuery.Add(uq);
```

```
    }           return Json(usuarioQuery);
  }
  [Route("/TipoSolicitud/Editar/{CODIGO_TIPO_SOLICITUD}")]
  public ActionResult Editar(string CODIGO_TIPO_SOLICITUD)
  {
    if (Session["Usuario"] == null)
    {
      return RedirectToAction("Index", "Login");
    }
    else
    {
      List<EstadoQuery> estados = new List<EstadoQuery>();
      estados.Add(new EstadoQuery(1, "Activo"));
      estados.Add(new EstadoQuery(0, "Inactivo"));
      ViewBag.Estados = estados;
      ViewBag.Codigo = CODIGO_TIPO_SOLICITUD;
      return View("Create");
    }       }
  [HttpPost]
  public ActionResult Eliminar(string codigo_tipo_solicitud)
  {
    TIPO_SOLICITUD                    usuario                    =
db.TIPO_SOLICITUD.Find(codigo_tipo_solicitud);
    usuario.ESTADO = 2;
    usuario.FECHA_EDICION = DateTime.Now;
    usuario.USUARIO_EDICION = Session["Codigo"].ToString();
    db.SaveChanges();
    return Json("Ok");
  }
     // GET: TipoSolicitud/Create
  public ActionResult Create()
  {
    if (Session["Usuario"] == null)
    {
      return RedirectToAction("Index", "Login");
    }
    else
```

```
    {
        List<EstadoQuery> estados = new List<EstadoQuery>();
        estados.Add(new EstadoQuery(1, "Activo"));
        estados.Add(new EstadoQuery(0, "Inactivo"));
        ViewBag.Estados = estados;
        ViewBag.Codigo = "";
        return View();
    }       }

// POST: TipoSolicitud/Create
[HttpPost]
public ActionResult Create(FormCollection collection)
{
    if (Session["Usuario"] == null)
    {
        return RedirectToAction("Index", "Login");
    }
    else
    {
        try
        {                   return RedirectToAction("Index");               }
        catch
        {                   return View();              }
    }       }
public ActionResult Edit(int id)
{
    if (Session["Usuario"] == null)
    {               return RedirectToAction("Index", "Login");          }
    else
    {               return View();          }
}
[HttpPost]
public ActionResult Delete(int id, FormCollection collection)
{
    try
    {               return RedirectToAction("Index");           }
    catch
```

```
{ return View(); } }

public JsonResult Guardar(TipoSolicitudInput input)
{
  // GUARDAR EL TIPO DE SOLICITUD
  if (input.codigo_inicial == null || input.codigo_inicial == "")
  {
    TIPO_SOLICITUD tipoSolicitud = new TIPO_SOLICITUD();
    tipoSolicitud.CODIGO_TIPO_SOLICITUD = input.codigo_tipo_solicitud;
    tipoSolicitud.DESCRIPCION = input.descripcion;
    tipoSolicitud.DETALLE = input.detalle;
    tipoSolicitud.PLAZO_ATENCION = input.plazo_atencion;
    tipoSolicitud.ESTADO = input.estado;
    tipoSolicitud.USUARIO_CREACION = Session["Codigo"].ToString();
    tipoSolicitud.FECHA_CREACION = DateTime.Now;
    tipoSolicitud.USUARIO_EDICION = Session["Codigo"].ToString();
    tipoSolicitud.FECHA_EDICION = DateTime.Now;
    tipoSolicitud.EMITE_CARTA = input.emite_carta.ToString();
    db.TIPO_SOLICITUD.Add(tipoSolicitud);
    db.SaveChanges();
   // EMITE CARTA DE RESPUESTA
    List<TipoSolicitudPlantillaCartaQuery> cartasRespuestas = new
List<TipoSolicitudPlantillaCartaQuery>();
    cartasRespuestas = input.plantillas;
    if (cartasRespuestas.Count != 0)
    {
      foreach (var carta in cartasRespuestas)
      {
        int correlativo = 0;
        if (db.PLANTILLA_TIPO_SOLICITUD.Count() == 0)
        { correlativo = correlativo + 1; }
        else
        { correlativo = b.PLANTILLA_TIPO_SOLICITUD.Count()
+ 1; }
        PLANTILLA_TIPO_SOLICITUD formato = new
PLANTILLA_TIPO_SOLICITUD();
        formato.CORRELATIVO = correlativo;
```

```
                formato.CODIGO_PLANTILLA = carta.codigo_plantilla;
                formato.CODIGO_TIPO_SOLICITUD = input.codigo_tipo_solicitud;
                formato.ESTADO = 1;
                formato.USUARIO_CREACION = Session["Codigo"].ToString();
                formato.FECHA_CREACION = DateTime.Now;
                formato.USUARIO_EDICION = Session["Codigo"].ToString();
                formato.FECHA_EDICION = DateTime.Now;
                db.PLANTILLA_TIPO_SOLICITUD.Add(formato);
                db.SaveChanges();
            }        }
        List<RolTipoCartaRespuestaInput> roles = new
List<RolTipoCartaRespuestaInput>();
        roles = input.relacionados;
        if (roles.Count != 0)
        {
          foreach (var rol in roles)
          {
            int correlativo = 0;
            if (db.ROL_TIPO_CARTA_RESPUESTA.Count() == 0)
            {              correlativo = correlativo + 1;              }
            else
            {                                                    correlativo =
db.ROL_TIPO_CARTA_RESPUESTA.Count() + 1;              }
            ROL_TIPO_CARTA_RESPUESTA formato = new
ROL_TIPO_CARTA_RESPUESTA();
            formato.CORRELATIVO = correlativo;
            formato.CODIGO_ROL = rol.codigo_rol;
            formato.CODIGO_TIPO_CARTA_RESPUESTA = rol.codigo_plantilla;
            formato.CODIGO_TIPO_SOLICITUD = input.codigo_tipo_solicitud;
            formato.ESTADO = 1;
            formato.USUARIO_CREACION = Session["Codigo"].ToString();
            formato.FECHA_CREACION  = DateTime.Now;
            formato.USUARIO_EDICION = Session["Codigo"].ToString();
            formato.FECHA_EDICION = DateTime.Now;
            db.ROL_TIPO_CARTA_RESPUESTA.Add(formato);
            db.SaveChanges();
          }        }
```

```
        List<RequisitoTipoSolicitudInput> requisitos = new
List<RequisitoTipoSolicitudInput>();
        requisitos = input.requisitos;
        requisitos.OrderBy(p => p.secuencia);
        if (requisitos.Count != 0)
        {
          foreach (var requisito in requisitos)
          {
            REQUISITO_TIPO_SOLICITUD requisitoTipoSolicitud = new
REQUISITO_TIPO_SOLICITUD();
            requisitoTipoSolicitud.COD_REQUISITO_TIPO_SOLICITUD =
input.codigo_tipo_solicitud;
            requisitoTipoSolicitud.SECUENCIA = requisito.secuencia;
            requisitoTipoSolicitud.DESCRIPCION = requisito.descripcion;
            requisitoTipoSolicitud.ESTADO = requisito.estado;
            requisitoTipoSolicitud.USUARIO_CREACION =
Session["Codigo"].ToString();
            requisitoTipoSolicitud.FECHA_CREACION = DateTime.Now;
            requisitoTipoSolicitud.USUARIO_EDICION =
Session["Codigo"].ToString();
            requisitoTipoSolicitud.FECHA_EDICION = DateTime.Now;
           try
            {
              db.REQUISITO_TIPO_SOLICITUD.Add(requisitoTipoSolicitud);
              db.SaveChanges();
              // FORMATOS
              int correlativo = 0;
              if (db.TIPO_SOLICITUD_REQ_FORMATO.Count() == 0)
              { correlativo = correlativo + 1; }
              else
              { correlativo =
db.TIPO_SOLICITUD_REQ_FORMATO.Count() + 1; }
              TIPO_SOLICITUD_REQ_FORMATO formato = new
TIPO_SOLICITUD_REQ_FORMATO();
              formato.CORRELATIVO = correlativo;
              formato.CODIGO_REQUISITO = requisito.secuencia.ToString();
              formato.CODIGO_TIPO_SOLICITUD = input.codigo_tipo_solicitud;
```

```
                    formato.ESTADO = 1;
                    formato.ES_AUTOCAD = requisito.esAutocad;
                    formato.ES_EXCEL = requisito.esExcel;
                    formato.ES_PDF = requisito.esPdf;
                    formato.ES_TEXTO = requisito.esTexto;
                    formato.ES_WORD = requisito.esWord;
                    formato.USUARIO_CREACION = Session["Codigo"].ToString();
                    formato.FECHA_CREACION = DateTime.Now;
                    formato.USUARIO_EDICION = Session["Codigo"].ToString();
                    formato.FECHA_EDICION = DateTime.Now;
                    db.TIPO_SOLICITUD_REQ_FORMATO.Add(formato);
                    db.SaveChanges();
                }
                catch (Exception EX)
                {                    throw;                    }
            }           }          }
        else
        {
            TIPO_SOLICITUD tipoSolicitud = new TIPO_SOLICITUD();
            tipoSolicitud = db.TIPO_SOLICITUD.Find(input.codigo_inicial);
            tipoSolicitud.CODIGO_TIPO_SOLICITUD = input.codigo_tipo_solicitud;
            tipoSolicitud.DESCRIPCION = input.descripcion;
            tipoSolicitud.DETALLE = input.detalle;
            tipoSolicitud.PLAZO_ATENCION = input.plazo_atencion;
            tipoSolicitud.ESTADO = input.estado;
            tipoSolicitud.EMITE_CARTA = input.emite_carta.ToString();
            tipoSolicitud.USUARIO_EDICION = Session["Codigo"].ToString();
            tipoSolicitud.FECHA_EDICION = DateTime.Now;
           db.SaveChanges();
            // CARTA DE RESPUESTA
            IList<PLANTILLA_TIPO_SOLICITUD>        plantillaTipoSolicitud        =
db.PLANTILLA_TIPO_SOLICITUD.Where(p => p.CODIGO_TIPO_SOLICITUD ==
input.codigo_inicial).ToList();
            foreach (var plantilla in plantillaTipoSolicitud)
            {
                plantilla.ESTADO = 0;
                plantilla.USUARIO_EDICION = Session["Codigo"].ToString();
```

```
            plantilla.FECHA_EDICION = DateTime.Now;
            db.SaveChanges();
        }
        IList<ROL_TIPO_CARTA_RESPUESTA> rolCartas = db.ROL_TIPO_CARTA_RESPUESTA.Where(p => p.CODIGO_TIPO_SOLICITUD == input.codigo_inicial).ToList();
        foreach (var plantilla in rolCartas)
        {
            plantilla.ESTADO = 0;
            plantilla.USUARIO_EDICION = Session["Codigo"].ToString();
            plantilla.FECHA_EDICION = DateTime.Now;
            db.SaveChanges();
        }
        if (input.emite_carta == 1)
        {
            IList<TipoSolicitudPlantillaCartaQuery> plantillas = input.plantillas;
            IList<RolTipoCartaRespuestaInput> relacionados = input.relacionados;
            foreach (var relacion in relacionados)
            {
                IList<ROL_TIPO_CARTA_RESPUESTA> plantillaInd = db.ROL_TIPO_CARTA_RESPUESTA.Where(p => p.CODIGO_TIPO_SOLICITUD == input.codigo_inicial && p.CODIGO_TIPO_CARTA_RESPUESTA == relacion.codigo_plantilla&& p.CODIGO_ROL == relacion.codigo_rol).ToList();
                if (plantillaInd.Count == 0)
                {
                    int correlativo = 0;
                    if (db.ROL_TIPO_CARTA_RESPUESTA.Count() == 0)
                    { correlativo = correlativo + 1; }
                    else
                    { correlativo = db.ROL_TIPO_CARTA_RESPUESTA.Count() + 1; }
                    ROL_TIPO_CARTA_RESPUESTA formato = new ROL_TIPO_CARTA_RESPUESTA();
                    formato.CORRELATIVO = correlativo;
                    formato.CODIGO_ROL = relacion.codigo_rol;
                    formato.CODIGO_TIPO_CARTA_RESPUESTA = relacion.codigo_plantilla;
```

```
                formato.CODIGO_TIPO_SOLICITUD = input.codigo_tipo_solicitud;
                formato.ESTADO = 1;
                formato.USUARIO_CREACION = Session["Codigo"].ToString();
                formato.FECHA_CREACION = DateTime.Now;
                formato.USUARIO_EDICION = Session["Codigo"].ToString();
                formato.FECHA_EDICION = DateTime.Now;
                db.ROL_TIPO_CARTA_RESPUESTA.Add(formato);
                db.SaveChanges();
            }
            else
            {
                foreach (var plt in plantillaInd)
                {
                    plt.ESTADO = 1;
                    plt.USUARIO_EDICION = Session["Codigo"].ToString();
                    plt.FECHA_EDICION = DateTime.Now;
                    db.SaveChanges();
                }           }           }
        foreach (var plantilla in plantillas)
        {
            IList<PLANTILLA_TIPO_SOLICITUD> plantillaInd =
db.PLANTILLA_TIPO_SOLICITUD.Where(p => p.CODIGO_TIPO_SOLICITUD ==
input.codigo_inicial && p.CODIGO_PLANTILLA ==
plantilla.codigo_plantilla).ToList();
            if (plantillaInd.Count == 0)
            {
                int correlativo = 0;
                if (db.PLANTILLA_TIPO_SOLICITUD.Count() == 0)
                {           correlativo = correlativo + 1;           }
                else
                {                                   correlativo =
db.PLANTILLA_TIPO_SOLICITUD.Count() + 1;           }
                PLANTILLA_TIPO_SOLICITUD formato = new
PLANTILLA_TIPO_SOLICITUD();
                formato.CORRELATIVO = correlativo;
                formato.CODIGO_PLANTILLA = plantilla.codigo_plantilla;
                formato.CODIGO_TIPO_SOLICITUD = input.codigo_tipo_solicitud;
```

```
                formato.ESTADO = 1;
                formato.USUARIO_CREACION = Session["Codigo"].ToString();
                formato.FECHA_CREACION = DateTime.Now;
                formato.USUARIO_EDICION = Session["Codigo"].ToString();
                formato.FECHA_EDICION = DateTime.Now;
                db.PLANTILLA_TIPO_SOLICITUD.Add(formato);
                db.SaveChanges();
            }
            else {
                foreach (var plt in plantillaInd)
                {
                    plt.ESTADO = 1;
                    plt.USUARIO_EDICION = Session["Codigo"].ToString();
                    plt.FECHA_EDICION = DateTime.Now;
                    db.SaveChanges();
                } } } }

        // EDITAR REQUISITOS
        IList<REQUISITO_TIPO_SOLICITUD> requisitoTipoSolicitud =
db.REQUISITO_TIPO_SOLICITUD.Where(p =>
p.COD_REQUISITO_TIPO_SOLICITUD == input.codigo_inicial).ToList();
        IList<RequisitoTipoSolicitudInput> requisitos = input.requisitos;
        foreach (var req in requisitoTipoSolicitud)
        {
            req.ESTADO = 0;
            req.COD_REQUISITO_TIPO_SOLICITUD = input.codigo_tipo_solicitud;
            req.USUARIO_EDICION = Session["Codigo"].ToString();
            req.FECHA_EDICION = DateTime.Now;
            db.SaveChanges();
           string secuencia_requisito = req.SECUENCIA.ToString();
            if (db.TIPO_SOLICITUD_REQ_FORMATO.Where(p =>
p.CODIGO_TIPO_SOLICITUD == input.codigo_inicial && p.CODIGO_REQUISITO
== secuencia_requisito).Count() != 0)
            {
                IList<TIPO_SOLICITUD_REQ_FORMATO> formatos =
db.TIPO_SOLICITUD_REQ_FORMATO.Where(p =>
```

```
p.CODIGO_TIPO_SOLICITUD == input.codigo_inicial && p.CODIGO_REQUISITO
== secuencia_requisito).ToList();
                foreach (var formato in formatos)
                {
                    formato.ESTADO = 0;
                    formato.CODIGO_TIPO_SOLICITUD = input.codigo_tipo_solicitud;
                    formato.USUARIO_EDICION = Session["Codigo"].ToString();
                    formato.FECHA_EDICION = DateTime.Now;
                    db.SaveChanges();
                }               }           }
            if (requisitos.Count != 0)
            {
                foreach (var requisito in requisitos)
                {
                    IList<REQUISITO_TIPO_SOLICITUD> requisitotipoSolicitudInd =
db.REQUISITO_TIPO_SOLICITUD.Where(p                                    =>
p.COD_REQUISITO_TIPO_SOLICITUD == input.codigo_tipo_solicitud &&
p.SECUENCIA == requisito.secuencia).ToList();
                    if (requisitotipoSolicitudInd.Count == 0)
                    {               REQUISITO_TIPO_SOLICITUD requisitoTipoSolicitudG
= new REQUISITO_TIPO_SOLICITUD();
                        requisitoTipoSolicitudG.COD_REQUISITO_TIPO_SOLICITUD =
input.codigo_tipo_solicitud;
                        requisitoTipoSolicitudG.SECUENCIA = requisito.secuencia;
                        requisitoTipoSolicitudG.DESCRIPCION = requisito.descripcion;
                        requisitoTipoSolicitudG.ESTADO = requisito.estado;
                        requisitoTipoSolicitudG.USUARIO_CREACION                =
Session["Codigo"].ToString();
                        requisitoTipoSolicitudG.FECHA_CREACION = DateTime.Now;
                        requisitoTipoSolicitudG.USUARIO_EDICION                 =
Session["Codigo"].ToString();
                        requisitoTipoSolicitudG.FECHA_EDICION = DateTime.Now;
                        db.REQUISITO_TIPO_SOLICITUD.Add(requisitoTipoSolicitudG);
                        db.SaveChanges();
                    }
                    else
                    {
```

```
                    foreach (var rq in requisitotipoSolicitudInd)
                    {
                        rq.DESCRIPCION = requisito.descripcion;
                        rq.ESTADO = requisito.estado;
                        rq.USUARIO_EDICION = Session["Codigo"].ToString();
                        rq.FECHA_EDICION = DateTime.Now;
                       db.SaveChanges();
                    }
                }
                string secuencia_requisito = requisito.secuencia.ToString();
                if (db.TIPO_SOLICITUD_REQ_FORMATO.Where(p => p.CODIGO_TIPO_SOLICITUD == input.codigo_inicial && p.CODIGO_REQUISITO == secuencia_requisito).Count() != 0)
                {
                    // EVALUAR LOS FORMATOS
                    IList<TIPO_SOLICITUD_REQ_FORMATO> formatos = db.TIPO_SOLICITUD_REQ_FORMATO.Where(p => p.CODIGO_TIPO_SOLICITUD == input.codigo_inicial && p.CODIGO_REQUISITO == secuencia_requisito).ToList();
                    if (formatos.Count != 0)
                    {
                        foreach (var formato in formatos)
                        {
                            formato.ESTADO = 1;
                            formato.ES_AUTOCAD = requisito.esAutocad;
                            formato.ES_EXCEL = requisito.esExcel;
                            formato.ES_PDF = requisito.esPdf;
                            formato.ES_TEXTO = requisito.esTexto;
                            formato.ES_WORD = requisito.esWord;
                            formato.USUARIO_EDICION = Session["Codigo"].ToString();
                            formato.FECHA_EDICION = DateTime.Now;
                            db.SaveChanges();
                        }
                    }
                    else
                    {
                        // INSERTAR
```

```
                    int correlativo = 0;
                    if (db.TIPO_SOLICITUD_REQ_FORMATO.Count() == 0)
                    {                       correlativo = correlativo + 1;                    }
                    else
                    {                                                       correlativo =
db.TIPO_SOLICITUD_REQ_FORMATO.Count() + 1;                    }
                    TIPO_SOLICITUD_REQ_FORMATO      formato      =      new
TIPO_SOLICITUD_REQ_FORMATO();
                    formato.CORRELATIVO = correlativo;
                    formato.CODIGO_REQUISITO = requisito.secuencia.ToString();
                    formato.CODIGO_TIPO_SOLICITUD                                =
input.codigo_tipo_solicitud;
                    formato.ESTADO = 1;
                    formato.ES_AUTOCAD = requisito.esAutocad;
                    formato.ES_EXCEL = requisito.esExcel;
                    formato.ES_PDF = requisito.esPdf;
                    formato.ES_TEXTO = requisito.esTexto;
                    formato.ES_WORD = requisito.esWord;
                    formato.USUARIO_CREACION = Session["Codigo"].ToString();
                    formato.FECHA_CREACION = DateTime.Now;
                    formato.USUARIO_EDICION = Session["Codigo"].ToString();
                    formato.FECHA_EDICION = DateTime.Now;
                    db.TIPO_SOLICITUD_REQ_FORMATO.Add(formato);
                    db.SaveChanges();
                }                }
                else
                {
                    int correlativo = 0;
                    if (db.TIPO_SOLICITUD_REQ_FORMATO.Count() == 0)
                    {                       correlativo = correlativo + 1;                    }
                    else
                    {                                                       correlativo =
db.TIPO_SOLICITUD_REQ_FORMATO.Count() + 1;                  }
                    TIPO_SOLICITUD_REQ_FORMATO      formato      =      new
TIPO_SOLICITUD_REQ_FORMATO();
                    formato.CORRELATIVO = correlativo;
```

```
                    formato.CODIGO_REQUISITO = requisito.secuencia.ToString();
                    formato.CODIGO_TIPO_SOLICITUD = input.codigo_tipo_solicitud;
                    formato.ESTADO = 1;
                    formato.ES_AUTOCAD = requisito.esAutocad;
                    formato.ES_EXCEL = requisito.esExcel;
                    formato.ES_PDF = requisito.esPdf;
                    formato.ES_TEXTO = requisito.esTexto;
                    formato.ES_WORD = requisito.esWord;
                    formato.USUARIO_CREACION = Session["Codigo"].ToString();
                    formato.FECHA_CREACION = DateTime.Now;
                    formato.USUARIO_EDICION = Session["Codigo"].ToString();
                    formato.FECHA_EDICION = DateTime.Now;
                    db.TIPO_SOLICITUD_REQ_FORMATO.Add(formato);
                    db.SaveChanges();
                }           }           }           }           return Json("OK");
        }
        public JsonResult Leer(string codigo)
        {
            TIPO_SOLICITUD tipo_solicitud = db.TIPO_SOLICITUD.Find(codigo);
            TipoSolicitudInd tipoSolicitudInd = new TipoSolicitudInd();
            if (tipo_solicitud != null)
            {
                if (tipo_solicitud.EMITE_CARTA is null)
                {               tipo_solicitud.EMITE_CARTA = 0.ToString();          }
                tipoSolicitudInd.codigo_tipo_solicitud                              =
tipo_solicitud.CODIGO_TIPO_SOLICITUD;
                tipoSolicitudInd.descripcion = tipo_solicitud.DESCRIPCION;
                tipoSolicitudInd.detalle = tipo_solicitud.DETALLE;
                tipoSolicitudInd.plazo_atencion = tipo_solicitud.PLAZO_ATENCION;
                tipoSolicitudInd.estado = tipo_solicitud.ESTADO;
                tipoSolicitudInd.emite_carta = Int32.Parse(tipo_solicitud.EMITE_CARTA);
                tipoSolicitudInd.codigo_aprobacion                                  =
tipo_solicitud.CODIGO_PLANTILLA_APROBACION;
                tipoSolicitudInd.codigo_rechazo                                     =
tipo_solicitud.CODIGO_PLANTILLA_RECHAZO;
                tipoSolicitudInd.informe_tecnico                                    =
tipo_solicitud.CODIGO_PLANTILLA_INFO_TEC;
```

```
            tipoSolicitudInd.ficha_tecnica =
tipo_solicitud.CODIGO_PLANTILLA_FIC_TE;
            tipoSolicitudInd.resolucion =
tipo_solicitud.CODIGO_PLANTILLA_RESOLUCION;
            tipoSolicitudInd.cuaderno_obra =
tipo_solicitud.CODIGO_PLANTILLA_CUADERNO_OBRA;
            List<REQUISITO_TIPO_SOLICITUD> requisitosTiposolicitud =
db.REQUISITO_TIPO_SOLICITUD.Where(p =>
p.COD_REQUISITO_TIPO_SOLICITUD == codigo && p.ESTADO == 1).OrderBy(p
=> p.SECUENCIA).ToList();
            List<RequisitoTipoSolicitudQuery> requisitos = new
List<RequisitoTipoSolicitudQuery>();
            foreach (var requisitoTiposolicitud in requisitosTiposolicitud)
            {
              RequisitoTipoSolicitudQuery requisito = new
RequisitoTipoSolicitudQuery();
              requisito.secuencia = requisitoTiposolicitud.SECUENCIA;
              requisito.descripcion = requisitoTiposolicitud.DESCRIPCION;
              requisito.estado = requisitoTiposolicitud.ESTADO;
              var secuencia = requisitoTiposolicitud.SECUENCIA.ToString();
              List<TIPO_SOLICITUD_REQ_FORMATO> formatos =
db.TIPO_SOLICITUD_REQ_FORMATO.Where(p => p.CODIGO_REQUISITO ==
secuencia && p.ESTADO == 1 && p.CODIGO_TIPO_SOLICITUD ==
codigo).ToList();
              foreach (var formato in formatos)
              {
                requisito.esAutocad = formato.ES_AUTOCAD;
                requisito.esExcel = formato.ES_EXCEL ;
                requisito.esPdf = formato.ES_PDF;
                requisito.esTexto = formato.ES_TEXTO;
                requisito.esWord = formato.ES_WORD;
                break;
              }
              requisitos.Add(requisito);
            }
```

```
            IList<PLANTILLA_TIPO_SOLICITUD> plantillaTipoSolicitud =
db.PLANTILLA_TIPO_SOLICITUD.Where(p => p.CODIGO_TIPO_SOLICITUD ==
codigo && p.ESTADO == 1).ToList();
            List<TipoSolicitudPlantillaCartaQuery> plantillas = new
List<TipoSolicitudPlantillaCartaQuery>();
            foreach (var plantilla in plantillaTipoSolicitud)
            {
              TipoSolicitudPlantillaCartaQuery plt = new
TipoSolicitudPlantillaCartaQuery();
              plt.codigo_plantilla = plantilla.CODIGO_PLANTILLA;
              plantillas.Add(plt);
            }
            IList<ROL_TIPO_CARTA_RESPUESTA> rolCartas =
db.ROL_TIPO_CARTA_RESPUESTA.Where(p => p.CODIGO_TIPO_SOLICITUD
== codigo && p.ESTADO == 1).ToList();
            List<RolTipoCartaRespuestaInput> relacionados = new
List<RolTipoCartaRespuestaInput>();
            foreach (var plantilla in rolCartas)
            { RolTipoCartaRespuestaInput plt = new
RolTipoCartaRespuestaInput();
              plt.codigo_plantilla = plantilla.CODIGO_TIPO_CARTA_RESPUESTA;
              plt.codigo_rol = plantilla.CODIGO_ROL;
              relacionados.Add(plt);
            }
            tipoSolicitudInd.requisitos = requisitos;
            tipoSolicitudInd.plantillas = plantillas;
            tipoSolicitudInd.relacionados = relacionados;
            tipoSolicitudInd.rpta_estado = "Ok";
            tipoSolicitudInd.rpta_mensaje = "Tipo Solicitud encontrado";
          }
          else
          {
            tipoSolicitudInd.rpta_estado = "Error";
            tipoSolicitudInd.rpta_mensaje = "Tipo Solicitud no existe";
          }          return Json(tipoSolicitudInd);       }
        public JsonResult GetTipoSolicitud()
        {
```

```
        List<TipoSolicitudGet> listTS = new List<TipoSolicitudGet>();
        IList<TIPO_SOLICITUD> tsList = db.TIPO_SOLICITUD.Where(p =>
p.ESTADO == 1).ToList();
        foreach (var tsq in tsList)
        {
           TipoSolicitudGet tse = new TipoSolicitudGet();
           tse.codigo_tipo_solicitud = tsq.CODIGO_TIPO_SOLICITUD;
           tse.descripcion_tipo_solicitud = tsq.DESCRIPCION;
           listTS.Add(tse);
        }          return Json(listTS);       }    }}
```

Registro y Administración de los Tipos de Proyectos

```
using iTextSharp.text;
using iTextSharp.text.pdf;
using OfficeOpenXml;
using OfficeOpenXml.Style;
using ProyectoTerceros.Models;
using ProyectoTerceros.Models.Administracion;
using ProyectoTerceros.Models.Consultas;
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Threading.Tasks;
using System.Web;
using System.Web.Mvc;
namespace ProyectoTerceros.Controllers.Administracion
{
   public class TipoProyectoController : Controller
   {
      private ProyectoTercerosEntities db = new ProyectoTercerosEntities();
      public ActionResult Index()
      {
         if (Session["Usuario"] == null)
         {             return RedirectToAction("Index", "Login");          }
         else          {             return View();          }
      }
      public ActionResult Consulta()
```

```
        {
            if (Session["Usuario"] == null)
            {               return RedirectToAction("Index", "Login");          }
            else          {               return View();          }
        }
        public JsonResult List()
        {
            List<TipoProyectoQuery> tpqs = new List<TipoProyectoQuery>();
            IList<TIPO_PROYECTO> tipoProyectos = db.TIPO_PROYECTO.Where(p=>
p.ESTADO != 2).ToList();
            foreach (var tipoProyecto in tipoProyectos)
            {
                TipoProyectoQuery tpq = new TipoProyectoQuery();
                tpq.codigo_tipo_proyecto = tipoProyecto.CODIGO_TIPO_PROYECTO;
                tpq.descripcion = tipoProyecto.DESCRIPCION;
                tpq.codigo_tipo_sistema_electrico                                    =
tipoProyecto.CODIGO_TIPO_SISTEMA_ELECTRICO;
                tpq.tipo_sistema_electrico                                           =
tipoProyecto.TIPO_SISTEMA_ELECTRICO.DESCRIPCION;
                tpqs.Add(tpq);
            }
            return Json(tpqs);
        }
        public JsonResult GetTipoProyecto()
        {
            List<TipoProyectoGet> tiposProyectos = new List<TipoProyectoGet>();
            IList<TIPO_PROYECTO> tipoProyectoList = db.TIPO_PROYECTO.Where(p
=> p.ESTADO == 1).ToList();
            foreach (var tsq in tipoProyectoList)
            {
                List<TIPO_PROYECTO_TIPO_SOLICITUD>              tpts              =
db.TIPO_PROYECTO_TIPO_SOLICITUD.Where(p                                  =>
p.CODIGO_TIPO_PROYECTO == tsq.CODIGO_TIPO_PROYECTO && p.ESTADO
== 1).ToList();
                if (tpts.Count != 0)
                {
                    TipoProyectoGet tipoProyecto = new TipoProyectoGet();
```

```
                tipoProyecto.codigo_tipo_proyecto = tsq.CODIGO_TIPO_PROYECTO;
                tipoProyecto.descripcion_tipo_proyecto = tsq.DESCRIPCION;
                if (tsq.TIPO_PROYECTO_RELACIONADO == null)
                {       tipoProyecto.codigo_tipo_proyecto_relacionado = "";                 }
                else          {
                   tipoProyecto.codigo_tipo_proyecto_relacionado                                =
tsq.TIPO_PROYECTO_RELACIONADO;
                }
                if (tsq.SE_RELACIONA_CON_PROYECTO == null)
                {
                   tipoProyecto.es_relacionado_tipo_proyecto = 0;
                }
                else
                {
                   tipoProyecto.es_relacionado_tipo_proyecto                                    =
tsq.SE_RELACIONA_CON_PROYECTO;
                }
                tipoProyecto.codigo_tipo_proyecto_relacionado                                   =
tsq.TIPO_PROYECTO_RELACIONADO;
                tiposProyectos.Add(tipoProyecto);
            }          }
         return Json(tiposProyectos,JsonRequestBehavior.AllowGet);
      }
      [Route("/TipoSolicitud/Editar/{CODIGO_TIPO_PROYECTO}")]
      public ActionResult Editar(string CODIGO_TIPO_PROYECTO)
      {
         if (Session["Usuario"] == null)
         {              return RedirectToAction("Index", "Login");          }
         else          {
            List<EstadoQuery> estados = new List<EstadoQuery>();
            estados.Add(new EstadoQuery(1, "Activo"));
            estados.Add(new EstadoQuery(0, "Inactivo"));
            ViewBag.Estados = estados;
            ViewBag.Codigo = CODIGO_TIPO_PROYECTO;
            return View("Create");
         }
      }
```

```
[HttpPost]
public ActionResult Eliminar(string codigo_tipo_proyecto)
{
   TIPO_PROYECTO                                usuario                                =
db.TIPO_PROYECTO.Find(codigo_tipo_proyecto);
   usuario.ESTADO = 2;
   usuario.FECHA_EDICION = DateTime.Now;
   usuario.USUARIO_EDICION = Session["Codigo"].ToString();
   db.SaveChanges();
   return Json("Ok");
}
// GET: TipoProyecto/Create
public ActionResult Create()
{
   if (Session["Usuario"] == null)
   {            return RedirectToAction("Index", "Login");          }
   else
   {
      List<EstadoQuery> estados = new List<EstadoQuery>();
      estados.Add(new EstadoQuery(1, "Activo"));
      estados.Add(new EstadoQuery(0, "Inactivo"));
      ViewBag.Estados = estados;
      return View();
   }      }
// POST: TipoProyecto/Edit/5
[HttpPost]
public ActionResult Edit(int id, FormCollection collection)
{
   if (Session["Usuario"] == null)
   {        return RedirectToAction("Index", "Login");          }
   else
   {
      try
      {
            return RedirectToAction("Index");
      }
      catch
```

```
            {                return View();            }        }      }
        public JsonResult Guardar(TipoProyectoInput input)
        {
            Status status = new Status();
            status.rpta_estado = "Ok";
            status.rpta_mensaje = "";
            // GUARDAR EL TIPO DE SOLICITUD
            if (input.codigo_inicial == null || input.codigo_inicial == "")
            {
                TIPO_PROYECTO tipoProyecto = new TIPO_PROYECTO();
                tipoProyecto.CODIGO_TIPO_PROYECTO = input.codigo_tipo_proyecto;
                tipoProyecto.CODIGO_TIPO_SISTEMA_ELECTRICO                    =
input.codigo_tipo_sistema_electrico;
                tipoProyecto.DESCRIPCION = input.descripcion_tipo_proyecto;
                tipoProyecto.ESTADO = input.estado;
                tipoProyecto.ABREVIATURA_TIPO_PROYECTO                        =
input.abreviatura_tipo_proyecto;
                tipoProyecto.SE_RELACIONA_CON_PROYECTO                        =
input.esRelacionadoTipoproyecto;
                if (input.esRelacionadoTipoproyecto == 1)
                {
                    tipoProyecto.TIPO_PROYECTO_RELACIONADO                    =
input.codigo_tipo_proyecto_relacionado;
                }
                else
                {
                    tipoProyecto.TIPO_PROYECTO_RELACIONADO                    =
input.codigo_tipo_proyecto_relacionado;
                }
                tipoProyecto.USUARIO_CREACION = Session["Codigo"].ToString();
                tipoProyecto.USUARIO_EDICION = Session["Codigo"].ToString();
                tipoProyecto.FECHA_CREACION = DateTime.Now;
                tipoProyecto.FECHA_EDICION = DateTime.Now;
                db.TIPO_PROYECTO.Add(tipoProyecto);
                db.SaveChanges();
```

```
            List<TipoSolicitudByTipoProyectoQuery> tipos_solicitudes = new
List<TipoSolicitudByTipoProyectoQuery>();
            tipos_solicitudes = input.tipos_solicitudes;
            tipos_solicitudes.OrderBy(p => p.orden_tipo_solicitud);
            if (tipos_solicitudes.Count != 0)
            {
               foreach (var tipo_solicitud in tipos_solicitudes)
               {
                  TIPO_PROYECTO_TIPO_SOLICITUD tipoProyectoByTipoSolicitud =
new TIPO_PROYECTO_TIPO_SOLICITUD();
                  tipoProyectoByTipoSolicitud.CODIGO_TIPO_PROYECTO          =
input.codigo_tipo_proyecto;
                  tipoProyectoByTipoSolicitud.CODIGO_TIPO_SOLICITUD         =
tipo_solicitud.codigo_tipo_solicitud;
                  tipoProyectoByTipoSolicitud.ORDEN                         =
tipo_solicitud.orden_tipo_solicitud;
                  tipoProyectoByTipoSolicitud.CANTIDAD_SOLICITUD_SALTO      =
tipo_solicitud.cantidad_solicitud_salto;
                  tipoProyectoByTipoSolicitud.ESTADO = 1;
                  tipoProyectoByTipoSolicitud.ESCAMBIOESTADO                =
tipo_solicitud.esCambioEstado;
                  tipoProyectoByTipoSolicitud.ESVALIDAREPRESENTANTE         =
tipo_solicitud.esValidaRepresentante;
                  tipoProyectoByTipoSolicitud.USUARIO_CREACION              =
Session["Codigo"].ToString();
                  tipoProyectoByTipoSolicitud.USUARIO_EDICION               =
Session["Codigo"].ToString();
                  tipoProyectoByTipoSolicitud.FECHA_CREACION = DateTime.Now;
                  tipoProyectoByTipoSolicitud.FECHA_EDICION = DateTime.Now;
db.TIPO_PROYECTO_TIPO_SOLICITUD.Add(tipoProyectoByTipoSolicitud);
                  db.SaveChanges();
               }          }          }
         else
         {
            TIPO_PROYECTO tipoProyecto = new TIPO_PROYECTO();
            tipoProyecto = db.TIPO_PROYECTO.Find(input.codigo_inicial);
            tipoProyecto.CODIGO_TIPO_PROYECTO = input.codigo_tipo_proyecto;
```

```
            tipoProyecto.CODIGO_TIPO_SISTEMA_ELECTRICO                    =
input.codigo_tipo_sistema_electrico;
            tipoProyecto.DESCRIPCION = input.descripcion_tipo_proyecto;
            tipoProyecto.ABREVIATURA_TIPO_PROYECTO                        =
input.abreviatura_tipo_proyecto;
            tipoProyecto.ESTADO = input.estado;
            tipoProyecto.SE_RELACIONA_CON_PROYECTO                        =
input.esRelacionadoTipoproyecto;
            if (input.esRelacionadoTipoproyecto == 1)
            {                           tipoProyecto.TIPO_PROYECTO_RELACIONADO =
input.codigo_tipo_proyecto_relacionado;
            }
            else
            {
              tipoProyecto.TIPO_PROYECTO_RELACIONADO                      =
input.codigo_tipo_proyecto_relacionado;
            }
            tipoProyecto.USUARIO_EDICION = Session["Codigo"].ToString();
            tipoProyecto.FECHA_EDICION = DateTime.Now;
            db.SaveChanges();
            // EDITAR TIPOS DE SOLICITUD
            IList<TIPO_PROYECTO_TIPO_SOLICITUD>
tipoProyectoByTipoSolicitudes = db.TIPO_PROYECTO_TIPO_SOLICITUD.Where(p
=> p.CODIGO_TIPO_PROYECTO == input.codigo_inicial && (p.ESTADO == 1 ||
p.ESTADO == 0)).ToList();
            IList<TipoSolicitudByTipoProyectoQuery>          tipos_solicitudes          =
input.tipos_solicitudes;
            foreach (var tp in tipoProyectoByTipoSolicitudes)
            {
              tp.ESTADO = 0;
              tp.CODIGO_TIPO_PROYECTO = input.codigo_tipo_proyecto;
              tp.USUARIO_EDICION = Session["Codigo"].ToString();
              tp.FECHA_EDICION = DateTime.Now;
              db.SaveChanges();
            }
            if (tipos_solicitudes.Count != 0)
            {
```

```
                foreach (var tipo_solicitud in tipos_solicitudes)
                {
                  IList<TIPO_PROYECTO_TIPO_SOLICITUD>
tipoProyectoTipoSolicitudInd = db.TIPO_PROYECTO_TIPO_SOLICITUD.Where(p
=> p.CODIGO_TIPO_SOLICITUD == tipo_solicitud.codigo_tipo_solicitud &&
p.CODIGO_TIPO_PROYECTO == input.codigo_tipo_proyecto  && (p.ESTADO ==
0)).ToList();
                  if (tipoProyectoTipoSolicitudInd.Count == 0)
                  {
                    TIPO_PROYECTO_TIPO_SOLICITUD
tipoProyectoByTipoSolicitud = new TIPO_PROYECTO_TIPO_SOLICITUD();
                    tipoProyectoByTipoSolicitud.CODIGO_TIPO_PROYECTO          =
input.codigo_tipo_proyecto;
                    tipoProyectoByTipoSolicitud.CODIGO_TIPO_SOLICITUD         =
tipo_solicitud.codigo_tipo_solicitud;
                    tipoProyectoByTipoSolicitud.ORDEN                         =
tipo_solicitud.orden_tipo_solicitud;
                    tipoProyectoByTipoSolicitud.CANTIDAD_SOLICITUD_SALTO      =
tipo_solicitud.cantidad_solicitud_salto;
                    tipoProyectoByTipoSolicitud.ESTADO = 1;
                    tipoProyectoByTipoSolicitud.ESCAMBIOESTADO                =
tipo_solicitud.esCambioEstado;
                    tipoProyectoByTipoSolicitud.ESVALIDAREPRESENTANTE         =
tipo_solicitud.esValidaRepresentante;
                    tipoProyectoByTipoSolicitud.USUARIO_CREACION              =
Session["Codigo"].ToString();
                    tipoProyectoByTipoSolicitud.USUARIO_EDICION               =
Session["Codigo"].ToString();
                    tipoProyectoByTipoSolicitud.FECHA_CREACION = DateTime.Now;
                    tipoProyectoByTipoSolicitud.FECHA_EDICION = DateTime.Now;
db.TIPO_PROYECTO_TIPO_SOLICITUD.Add(tipoProyectoByTipoSolicitud);
                    db.SaveChanges();
                  }
                  else
                  {
                    foreach (var tp in tipoProyectoTipoSolicitudInd)
                    {
```

```
                    tp.ORDEN = tipo_solicitud.orden_tipo_solicitud;
                    tp.ESTADO = 1;
                    tp.CANTIDAD_SOLICITUD_SALTO =
tipo_solicitud.cantidad_solicitud_salto;
                    tp.ESCAMBIOESTADO = tipo_solicitud.esCambioEstado;
                    tp.ESVALIDAREPRESENTANTE =
tipo_solicitud.esValidaRepresentante;
                    tp.USUARIO_EDICION = Session["Codigo"].ToString();
                    tp.FECHA_EDICION = DateTime.Now;
                    db.SaveChanges();
                }               }
            }           }       }
        return Json("Ok");
    }
    public JsonResult Leer(string codigo)
    {
        TIPO_PROYECTO tipo_proyecto = db.TIPO_PROYECTO.Find(codigo);
        TipoProyectoInd tipoProyectoInd = new TipoProyectoInd();
        if (tipo_proyecto != null)
        {
            tipoProyectoInd.codigo_tipo_proyecto =
tipo_proyecto.CODIGO_TIPO_PROYECTO;
            tipoProyectoInd.descripcion_tipo_proyecto = tipo_proyecto.DESCRIPCION;
            tipoProyectoInd.codigo_tipo_sistema_electrico =
tipo_proyecto.CODIGO_TIPO_SISTEMA_ELECTRICO;
            tipoProyectoInd.estado = tipo_proyecto.ESTADO;
            tipoProyectoInd.abreviatura_tipo_proyecto =
tipo_proyecto.ABREVIATURA_TIPO_PROYECTO;
            tipoProyectoInd.esRelacionadoTipoProyecto =
tipo_proyecto.SE_RELACIONA_CON_PROYECTO;
            tipoProyectoInd.codigo_tipo_proyecto_relacionado =
tipo_proyecto.TIPO_PROYECTO_RELACIONADO;
            List<TIPO_PROYECTO_TIPO_SOLICITUD> tipoProyectoTipoSolicitudes
=           db.TIPO_PROYECTO_TIPO_SOLICITUD.Where(p =>
p.CODIGO_TIPO_PROYECTO == codigo && p.ESTADO == 1).OrderBy(p=>
p.ORDEN).ToList();
```

```
            List<TipoSolicitudByTipoProyectoQuery> tiposProyectosTiposSolicitudes =
new List<TipoSolicitudByTipoProyectoQuery>();
            foreach (var tipoSolicitudByTipoProyecto in tipoProyectoTipoSolicitudes)
            {
              TipoSolicitudByTipoProyectoQuery tipoSolicitud = new
TipoSolicitudByTipoProyectoQuery();
              tipoSolicitud.codigo_tipo_solicitud =
tipoSolicitudByTipoProyecto.CODIGO_TIPO_SOLICITUD;
              tipoSolicitud.descripcion_tipo_solicitud =
tipoSolicitudByTipoProyecto.TIPO_SOLICITUD.DESCRIPCION;
              tipoSolicitud.orden_tipo_solicitud = tipoSolicitudByTipoProyecto.ORDEN;
              tipoSolicitud.cantidad_solicitud_salto =
tipoSolicitudByTipoProyecto.CANTIDAD_SOLICITUD_SALTO;
              tipoSolicitud.estado = tipoSolicitudByTipoProyecto.ESTADO;
              tipoSolicitud.esCambioEstado =
tipoSolicitudByTipoProyecto.ESCAMBIOESTADO;
              tipoSolicitud.esValidaRepresentante =
tipoSolicitudByTipoProyecto.ESVALIDAREPRESENTANTE;
              tiposProyectosTiposSolicitudes.Add(tipoSolicitud);
            }
            tipoProyectoInd.tipos_solicitudes = tiposProyectosTiposSolicitudes;
            tipoProyectoInd.rpta_estado = "Ok";
            tipoProyectoInd.rpta_mensaje = "Usuario encontrado";
          }
          else
          {
            tipoProyectoInd.rpta_estado = "Error";
            tipoProyectoInd.rpta_mensaje = "Usuario no existe";
          }
          return Json(tipoProyectoInd);
        }
```

Registro y Administración de los Típicos Constructivos y

Consulta de información de los Requisitos de los Tipos de Proyecto y Típicos Constructivos

```
using iTextSharp.text;
using iTextSharp.text.pdf;
using OfficeOpenXml;
```

```
using OfficeOpenXml.Style;
using ProyectoTerceros.Models;
using ProyectoTerceros.Models.Administracion;
using ProyectoTerceros.Models.Consultas;
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Threading.Tasks;
using System.Web;
using System.Web.Mvc;
namespace ProyectoTerceros.Controllers.Administracion
{
   public class TipoProyectoController : Controller
   {
      private ProyectoTercerosEntities db = new ProyectoTercerosEntities();
      public ActionResult Consulta()
      {
         if (Session["Usuario"] == null)
         {            return RedirectToAction("Index", "Login");          }
         else          {              return View();          }
      }
      public JsonResult ListConsulta()
      {
         List<TipoProyectoConsultQuery>                tpcq                =                new
List<TipoProyectoConsultQuery>();
         IList<TIPO_PROYECTO> tipoProyectos = db.TIPO_PROYECTO.Where(p =>
p.ESTADO == 1).ToList();
         foreach (var tipoProyecto in tipoProyectos)
         {
            TipoProyectoConsultQuery tpc = new TipoProyectoConsultQuery();
            tpc.codigo_tipo_proyecto = tipoProyecto.CODIGO_TIPO_PROYECTO;
            tpc.descripcion_tipo_proyecto = tipoProyecto.DESCRIPCION;
            IList<TIPO_PROYECTO_TIPO_SOLICITUD>
tipoProyectoByTipoSolicitud   =   db.TIPO_PROYECTO_TIPO_SOLICITUD.Where(p
=>       p.ESTADO       ==       1       &&       p.CODIGO_TIPO_PROYECTO       ==
tipoProyecto.CODIGO_TIPO_PROYECTO).ToList();
```

```
            IList<TipoSolicitudByTipoProyectoConsultQuery> solicitudes = new
List<TipoSolicitudByTipoProyectoConsultQuery>();
            foreach (var tipoproyectobytiposoli in tipoProyectoByTipoSolicitud)
            {
              IList<TIPO_SOLICITUD> tipoSolicitudes =
db.TIPO_SOLICITUD.Where(p => p.ESTADO == 1 &&
p.CODIGO_TIPO_SOLICITUD ==
tipoproyectobytiposoli.CODIGO_TIPO_SOLICITUD).ToList();
              foreach (var tipoSolicitud in tipoSolicitudes)
              {
                TipoSolicitudByTipoProyectoConsultQuery solicitud = new
TipoSolicitudByTipoProyectoConsultQuery();
                solicitud.codigo_tipo_solicitud =
tipoSolicitud.CODIGO_TIPO_SOLICITUD;
                solicitud.descripcion_tipo_solicitud = tipoSolicitud.DESCRIPCION;
                IList<REQUISITO_TIPO_SOLICITUD> requisitosTipoSolicitud =
db.REQUISITO_TIPO_SOLICITUD.Where(p => p.ESTADO == 1 &&
p.COD_REQUISITO_TIPO_SOLICITUD ==
tipoSolicitud.CODIGO_TIPO_SOLICITUD).ToList();
                IList<RequisitoByTipoSolicitudByTipoProyectoConsultaQuery>
requisitos = new List<RequisitoByTipoSolicitudByTipoProyectoConsultaQuery>();
                foreach (var requisitoTipoSolicitud in requisitosTipoSolicitud)
                {
                  RequisitoByTipoSolicitudByTipoProyectoConsultaQuery requisito =
new RequisitoByTipoSolicitudByTipoProyectoConsultaQuery();
                  requisito.codigo_requisito =
requisitoTipoSolicitud.COD_REQUISITO_TIPO_SOLICITUD;
                  requisito.descripcion_requisito =
requisitoTipoSolicitud.DESCRIPCION;
                  var secuencia = requisitoTipoSolicitud.SECUENCIA.ToString();
                  List<TIPO_SOLICITUD_REQ_FORMATO> formatos =
db.TIPO_SOLICITUD_REQ_FORMATO.Where(p => p.CODIGO_REQUISITO ==
secuencia && p.ESTADO == 1 && p.CODIGO_TIPO_SOLICITUD ==
tipoSolicitud.CODIGO_TIPO_SOLICITUD).ToList();
                  foreach (var formato in formatos)
                  {
                    requisito.esAutocad = formato.ES_AUTOCAD;
```

```
                    requisito.esExcel = formato.ES_EXCEL;
                    requisito.esPdf = formato.ES_PDF;
                    requisito.esTexto = formato.ES_TEXTO;
                    requisito.esWord = formato.ES_WORD;
                    break;
                }
                requisitos.Add(requisito);
            }
            solicitud.requisitos = requisitos;
            solicitudes.Add(solicitud);
        }        }
        tpc.tipos_solicitudes = solicitudes;
        tpcq.Add(tpc);
      }
      return Json(tpcq);
    }
    [Route("/TipoProyecto/DescargarExcel/{codigo_tipo_proyecto}")]
    public async Task<FileResult> DescargarExcel(string codigo_tipo_proyecto)
    {
      TIPO_PROYECTO tipoProyectos = db.TIPO_PROYECTO.Find(codigo_tipo_proyecto);
      string descripcion_tipo_proyecto = tipoProyectos.DESCRIPCION;
      try
      {
        var stream = new System.IO.MemoryStream();
        using (ExcelPackage package = new ExcelPackage(stream))
        {
          IList<TIPO_PROYECTO_TIPO_SOLICITUD> tipoProyectoByTipoSolicitud = db.TIPO_PROYECTO_TIPO_SOLICITUD.Where(p => p.ESTADO == 1 && p.CODIGO_TIPO_PROYECTO == codigo_tipo_proyecto).ToList();
          IList<TipoSolicitudByTipoProyectoConsultQuery> solicitudes = new List<TipoSolicitudByTipoProyectoConsultQuery>();
          foreach (var tipoproyectobytiposoli in tipoProyectoByTipoSolicitud)
          {
            IList<TIPO_SOLICITUD> tipoSolicitudes = db.TIPO_SOLICITUD.Where(p => p.ESTADO == 1 &&
```

```
p.CODIGO_TIPO_SOLICITUD ==
tipoproyectobytiposoli.CODIGO_TIPO_SOLICITUD).ToList();
                foreach (var tipoSolicitud in tipoSolicitudes)
                {
                  TipoSolicitudByTipoProyectoConsultQuery solicitud = new
TipoSolicitudByTipoProyectoConsultQuery();
                  solicitud.codigo_tipo_solicitud =
tipoSolicitud.CODIGO_TIPO_SOLICITUD;
                  solicitud.descripcion_tipo_solicitud = tipoSolicitud.DESCRIPCION;
                  IList<REQUISITO_TIPO_SOLICITUD> requisitosTipoSolicitud =
db.REQUISITO_TIPO_SOLICITUD.Where(p => p.ESTADO == 1 &&
p.COD_REQUISITO_TIPO_SOLICITUD ==
tipoSolicitud.CODIGO_TIPO_SOLICITUD).ToList();
                  IList<RequisitoByTipoSolicitudByTipoProyectoConsultaQuery>
requisitos = new List<RequisitoByTipoSolicitudByTipoProyectoConsultaQuery>();
                  foreach (var requisitoTipoSolicitud in requisitosTipoSolicitud)
                  {
                    RequisitoByTipoSolicitudByTipoProyectoConsultaQuery requisito
= new RequisitoByTipoSolicitudByTipoProyectoConsultaQuery();
                    requisito.codigo_requisito =
requisitoTipoSolicitud.COD_REQUISITO_TIPO_SOLICITUD;
                    requisito.descripcion_requisito =
requisitoTipoSolicitud.DESCRIPCION;
                    var secuencia = requisitoTipoSolicitud.SECUENCIA.ToString();
                    List<TIPO_SOLICITUD_REQ_FORMATO> formatos =
db.TIPO_SOLICITUD_REQ_FORMATO.Where(p => p.CODIGO_REQUISITO ==
secuencia && p.ESTADO == 1 && p.CODIGO_TIPO_SOLICITUD ==
tipoSolicitud.CODIGO_TIPO_SOLICITUD).ToList();
                    foreach (var formato in formatos)
                    {
                      requisito.esAutocad = formato.ES_AUTOCAD;
                      requisito.esExcel = formato.ES_EXCEL;
                      requisito.esPdf = formato.ES_PDF;
                      requisito.esTexto = formato.ES_TEXTO;
                      requisito.esWord = formato.ES_WORD;
                      break;
                    }
```

```
                    requisitos.Add(requisito);
                }
                solicitud.requisitos = requisitos;
                solicitudes.Add(solicitud);
            }
        }
        if (solicitudes.Count != 0)
        {
            foreach (var solicitud in solicitudes)
            {
                ExcelWorksheet worksheet = package.Workbook.Worksheets.Add(solicitud.codigo_tipo_solicitud + " - " + solicitud.descripcion_tipo_solicitud);
                worksheet.View.ShowGridLines = false;
                worksheet.Cells[1, 1].Value = "TIPO DE PROYECTO";
                worksheet.Cells["A1:B1"].Merge = true;
                worksheet.Cells["C1:D1"].Merge = true;
                worksheet.Cells[1, 3].Value = descripcion_tipo_proyecto;
                worksheet.Cells[1, 3].Style.WrapText = true;
                worksheet.Cells[2, 1].Value = "TIPO DE SOLICITUD";
                worksheet.Cells["A2:B2"].Merge = true;
                worksheet.Cells["C2:D2"].Merge = true;
                worksheet.Cells[2, 3].Value = solicitud.codigo_tipo_solicitud + " - " + solicitud.descripcion_tipo_solicitud;
                worksheet.Cells[2, 3].Style.WrapText = true;
                worksheet.Row(1).Style.Font.Bold = true;
                worksheet.Row(1).Style.Font.Size = 12;
                worksheet.Row(1).Style.HorizontalAlignment = ExcelHorizontalAlignment.Center;
                worksheet.Row(1).Style.VerticalAlignment = ExcelVerticalAlignment.Center;

                worksheet.Row(2).Style.Font.Bold = true;
                worksheet.Row(2).Style.Font.Size = 12;
```

```
                worksheet.Row(2).Style.HorizontalAlignment =
ExcelHorizontalAlignment.Center;
                worksheet.Row(2).Style.VerticalAlignment =
ExcelVerticalAlignment.Center;
                worksheet.Column(1).Width = 8;
                worksheet.Column(2).Width = 27;
                worksheet.Column(3).Width = 45;
                worksheet.Column(1).Style.HorizontalAlignment =
ExcelHorizontalAlignment.Center;
                worksheet.Column(1).Style.VerticalAlignment =
ExcelVerticalAlignment.Center;
                worksheet.Column(2).Style.HorizontalAlignment =
ExcelHorizontalAlignment.Center;
                worksheet.Column(2).Style.VerticalAlignment =
ExcelVerticalAlignment.Center;
                worksheet.Cells[3, 1].Value = "";
                worksheet.Cells[4, 1].Value = "REQUISITOS";
                worksheet.Cells["A4:D4"].Merge = true;
                worksheet.Row(4).Style.Font.Bold = true;
                worksheet.Row(4).Style.Font.Size = 12;
                worksheet.Row(4).Style.HorizontalAlignment =
ExcelHorizontalAlignment.Center;
                worksheet.Cells[5, 1].Value = "ITEM";
                worksheet.Row(5).Style.Font.Bold = true;
                worksheet.Row(5).Style.Font.Size = 12;
                worksheet.Row(5).Style.HorizontalAlignment =
ExcelHorizontalAlignment.Center;
                worksheet.Column(5).Width = 10;
                worksheet.Cells[5, 2].Value = "DESCRIPCION";
                worksheet.Cells["B5:C5"].Merge = true;
                worksheet.Row(5).Style.Font.Bold = true;
                worksheet.Row(5).Style.Font.Size = 12;
                worksheet.Row(5).Style.HorizontalAlignment =
ExcelHorizontalAlignment.Center;
                worksheet.Column(5).Width = 5;
                worksheet.Cells[5, 4].Value = "FORMATO";
                worksheet.Row(5).Style.Font.Bold = true;
```

```
                worksheet.Row(5).Style.Font.Size = 12;
                worksheet.Row(5).Style.HorizontalAlignment =
ExcelHorizontalAlignment.Center;
                worksheet.Column(5).Width =10;
                worksheet.Cells[1, 1].Style.Border.Top.Style =
ExcelBorderStyle.Thin;
                worksheet.Cells[1, 1].Style.Border.Left.Style =
ExcelBorderStyle.Thin;
                worksheet.Cells[1, 1].Style.Border.Right.Style =
ExcelBorderStyle.Thin;
                worksheet.Cells[1, 1].Style.Border.Bottom.Style =
ExcelBorderStyle.Thin;
                worksheet.Cells[1, 2].Style.Border.Top.Style =
ExcelBorderStyle.Thin;
                worksheet.Cells[1, 2].Style.Border.Left.Style =
ExcelBorderStyle.Thin;
                worksheet.Cells[1, 2].Style.Border.Right.Style =
ExcelBorderStyle.Thin;
                worksheet.Cells[1, 2].Style.Border.Bottom.Style =
ExcelBorderStyle.Thin;
                worksheet.Cells[1, 3].Style.Border.Top.Style =
ExcelBorderStyle.Thin;
                worksheet.Cells[1, 3].Style.Border.Left.Style =
ExcelBorderStyle.Thin;
                worksheet.Cells[1, 3].Style.Border.Right.Style =
ExcelBorderStyle.Thin;
                worksheet.Cells[1, 3].Style.Border.Bottom.Style =
ExcelBorderStyle.Thin;
                worksheet.Cells[1, 4].Style.Border.Top.Style =
ExcelBorderStyle.Thin;
                worksheet.Cells[1, 4].Style.Border.Left.Style =
ExcelBorderStyle.Thin;
                worksheet.Cells[1, 4].Style.Border.Right.Style =
ExcelBorderStyle.Thin;
                worksheet.Cells[1, 4].Style.Border.Bottom.Style =
ExcelBorderStyle.Thin;
```

```
                worksheet.Cells[2,  1].Style.Border.Top.Style  =
ExcelBorderStyle.Thin;
                worksheet.Cells[2,  1].Style.Border.Left.Style  =
ExcelBorderStyle.Thin;
                worksheet.Cells[2,  1].Style.Border.Right.Style  =
ExcelBorderStyle.Thin;
                worksheet.Cells[2,  1].Style.Border.Bottom.Style  =
ExcelBorderStyle.Thin;
                // Assign borders
                worksheet.Cells[2,  2].Style.Border.Top.Style  =
ExcelBorderStyle.Thin;
                worksheet.Cells[2,  2].Style.Border.Left.Style  =
ExcelBorderStyle.Thin;
                worksheet.Cells[2,  2].Style.Border.Right.Style  =
ExcelBorderStyle.Thin;
                worksheet.Cells[2,  2].Style.Border.Bottom.Style  =
ExcelBorderStyle.Thin;
                worksheet.Cells[2,  3].Style.Border.Top.Style  =
ExcelBorderStyle.Thin;
                worksheet.Cells[2,  3].Style.Border.Left.Style  =
ExcelBorderStyle.Thin;
                worksheet.Cells[2,  3].Style.Border.Right.Style  =
ExcelBorderStyle.Thin;
                worksheet.Cells[2,  3].Style.Border.Bottom.Style  =
ExcelBorderStyle.Thin;
                worksheet.Cells[2,  4].Style.Border.Top.Style  =
ExcelBorderStyle.Thin;
                worksheet.Cells[2,  4].Style.Border.Left.Style  =
ExcelBorderStyle.Thin;
                worksheet.Cells[2,  4].Style.Border.Right.Style  =
ExcelBorderStyle.Thin;
                worksheet.Cells[2,  4].Style.Border.Bottom.Style  =
ExcelBorderStyle.Thin;
                worksheet.Cells[4,  1].Style.Border.Top.Style  =
ExcelBorderStyle.Thin;
                worksheet.Cells[4,  1].Style.Border.Left.Style  =
ExcelBorderStyle.Thin;
```

```
                worksheet.Cells[4,  1].Style.Border.Right.Style  =
ExcelBorderStyle.Thin;
                worksheet.Cells[4,  1].Style.Border.Bottom.Style  =
ExcelBorderStyle.Thin;
                worksheet.Cells[4,  2].Style.Border.Top.Style  =
ExcelBorderStyle.Thin;
                worksheet.Cells[4,  2].Style.Border.Left.Style  =
ExcelBorderStyle.Thin;
                worksheet.Cells[4,  2].Style.Border.Right.Style  =
ExcelBorderStyle.Thin;
                worksheet.Cells[4,  2].Style.Border.Bottom.Style  =
ExcelBorderStyle.Thin;
                worksheet.Cells[4,  3].Style.Border.Top.Style  =
ExcelBorderStyle.Thin;
                worksheet.Cells[4,  3].Style.Border.Left.Style  =
ExcelBorderStyle.Thin;
                worksheet.Cells[4,  3].Style.Border.Right.Style  =
ExcelBorderStyle.Thin;
                worksheet.Cells[4,  3].Style.Border.Bottom.Style  =
ExcelBorderStyle.Thin;
                worksheet.Cells[4,  4].Style.Border.Top.Style  =
ExcelBorderStyle.Thin;
                worksheet.Cells[4,  4].Style.Border.Left.Style  =
ExcelBorderStyle.Thin;
                worksheet.Cells[4,  4].Style.Border.Right.Style  =
ExcelBorderStyle.Thin;
                worksheet.Cells[4,  4].Style.Border.Bottom.Style  =
ExcelBorderStyle.Thin;
                worksheet.Cells[5,  1].Style.Border.Top.Style  =
ExcelBorderStyle.Thin;
                worksheet.Cells[5,  1].Style.Border.Left.Style  =
ExcelBorderStyle.Thin;
                worksheet.Cells[5,  1].Style.Border.Right.Style  =
ExcelBorderStyle.Thin;
                worksheet.Cells[5,  1].Style.Border.Bottom.Style  =
ExcelBorderStyle.Thin;
                worksheet.Cells[5, 2].Style.Border.Top.Style = ExcelBorderStyle.Thin;
```

```
                worksheet.Cells[5,          2].Style.Border.Left.Style          =
ExcelBorderStyle.Thin;
                worksheet.Cells[5,          2].Style.Border.Right.Style         =
ExcelBorderStyle.Thin;
                worksheet.Cells[5,          2].Style.Border.Bottom.Style        =
ExcelBorderStyle.Thin;
                worksheet.Cells[5,          3].Style.Border.Top.Style           =
ExcelBorderStyle.Thin;
                worksheet.Cells[5,          3].Style.Border.Left.Style          =
ExcelBorderStyle.Thin;
                worksheet.Cells[5,          3].Style.Border.Right.Style         =
ExcelBorderStyle.Thin;
                worksheet.Cells[5,          3].Style.Border.Bottom.Style        =
ExcelBorderStyle.Thin;
                worksheet.Cells[5,          4].Style.Border.Top.Style           =
ExcelBorderStyle.Thin;
                worksheet.Cells[5,          4].Style.Border.Left.Style          =
ExcelBorderStyle.Thin;
                worksheet.Cells[5,          4].Style.Border.Right.Style         =
ExcelBorderStyle.Thin;
                worksheet.Cells[5,          4].Style.Border.Bottom.Style        =
ExcelBorderStyle.Thin;
                var requisitos = solicitud.requisitos;
                var contador = 0;
                for (int i = 0; i < requisitos.Count; i++)
                {
                   try
                   {                        contador++;
                      string formatos = "";
                      if (requisitos[i].esAutocad == 1)
                      {                          if (formatos == "")
                         {formatos = formatos + "dwg";}
                         else   {   formatos = formatos + ", dwg";     }                  }
                      if (requisitos[i].esExcel == 1)
                      {
                         if (formatos == "")
                         {  formatos = formatos + "xlsx";         }
```

```
                    else  {     formatos = formatos + ", xlsx"; }            }
                if (requisitos[i].esPdf == 1)
                {
                    if (formatos == "")
                    { formatos = formatos + "pdf";}
                    else{formatos = formatos + ", pdf";}
                }
                if (requisitos[i].esWord == 1)
                {
                    if (formatos == "")
                    {  formatos = formatos + "docx";}
                    else
                    {formatos = formatos + ", docx";}
                }
                worksheet.Cells[i + 6, 1].Value = contador.ToString();
                worksheet.Cells[i + 6, 2].Value =
requisitos[i].descripcion_requisito.ToString();
                worksheet.Cells[i + 6, 4].Value = formatos.ToString();
                worksheet.Cells[i + 6, 1].Style.Border.Top.Style =
ExcelBorderStyle.Thin;
                worksheet.Cells[i + 6, 1].Style.Border.Left.Style =
ExcelBorderStyle.Thin;
                worksheet.Cells[i + 6, 1].Style.Border.Right.Style =
ExcelBorderStyle.Thin;
                worksheet.Cells[i + 6, 1].Style.Border.Bottom.Style =
ExcelBorderStyle.Thin;
                worksheet.Cells[i + 6, 2].Style.Border.Top.Style =
ExcelBorderStyle.Thin;
                worksheet.Cells[i + 6, 2].Style.Border.Left.Style =
ExcelBorderStyle.Thin;
                worksheet.Cells[i + 6, 2].Style.Border.Right.Style =
ExcelBorderStyle.Thin;
                worksheet.Cells[i + 6, 2].Style.Border.Bottom.Style =
ExcelBorderStyle.Thin;
                // Assign borders
                worksheet.Cells[i + 6, 3].Style.Border.Top.Style =
ExcelBorderStyle.Thin;
```

```
                    worksheet.Cells[i + 6, 3].Style.Border.Left.Style = ExcelBorderStyle.Thin;
                    worksheet.Cells[i + 6, 3].Style.Border.Right.Style = ExcelBorderStyle.Thin;
                    worksheet.Cells[i + 6, 3].Style.Border.Bottom.Style = ExcelBorderStyle.Thin;
                    worksheet.Cells[i + 6, 4].Style.Border.Top.Style = ExcelBorderStyle.Thin;
                    worksheet.Cells[i + 6, 4].Style.Border.Left.Style = ExcelBorderStyle.Thin;
                    worksheet.Cells[i + 6, 4].Style.Border.Right.Style = ExcelBorderStyle.Thin;
                    worksheet.Cells[i + 6, 4].Style.Border.Bottom.Style = ExcelBorderStyle.Thin;
                    int celda = i + 6;
                    worksheet.Cells["B" + celda + ":C" + celda].Merge = true;
                    worksheet.Cells["B" + celda + ":C" + celda].Style.WrapText = true;
 }

                    catch (Exception ex)
                    {
                    } } } }
               package.Save();
               stream.Position = 0;
              string fileName = @descripcion_tipo_proyecto + ".xlsx";
               using (Stream responseStream = stream)
               using (var memoryStream = new MemoryStream())
               {
                  Response.ContentType = "application/vnd.openxmlformats-officedocument.spreadsheetml.sheet";
                  Response.AddHeader("content-disposition", "attachment; filename=" + @fileName);
                  package.SaveAs(memoryStream);
                  memoryStream.WriteTo(Response.OutputStream);
                  Response.Flush();
                  Response.End();
```

```
            }            }
            return null;
        }
        catch (Exception ex)
        {throw;          }       }
```

Registro de Expedientes – Tercero

```
using ProyectoTerceros.Models;
using ProyectoTerceros.Models.Administracion;
using ProyectoTerceros.Models.Registro.UsuarioExterno;
using ProyectoTerceros.Models.Registro.UsuarioInterno;
using ProyectoTerceros.Models.Seguridad;
using System;
using System.Collections.Generic;
using System.Configuration;
using System.IO;
using System.Linq;
using System.Net.Http;
using System.Net.Http.Headers;
using System.Text.RegularExpressions;
using System.Threading.Tasks;
using System.Web;
using System.Web.Mvc;
namespace ProyectoTerceros.Controllers.Seguridad
{
  public class UsuarioExternoController : Controller
  {
    private ProyectoTercerosEntities db = new ProyectoTercerosEntities();
    public ActionResult ExpedienteCreate()
    {
      if (Session["Usuario"] == null)
      {return RedirectToAction("Index", "Login");}
      else
      {string codigoSession = Session["Codigo"].ToString();
        List<EXPEDIENTE> expedientes = db.EXPEDIENTE.Where(p =>
p.USUARIO_CREACION == codigoSession && (p.ESTADO != "2" && p.ESTADO
!= "0")).ToList();
        ViewBag.CantidadExpedientes = expedientes.Count();
```

```
            return View();} }
        public JsonResult GetExpedientesAnteriores(string codigo_expediente)
        {
            List<GetExpedientesAnterioresList> expedienteList = new
List<GetExpedientesAnterioresList>();
            string codigoSession = Session["Codigo"].ToString();
            List<EXPEDIENTE> expedientes = db.EXPEDIENTE.Where(p =>
p.FECHA_EXPEDIENTE <= DateTime.Now && p.USUARIO_CREACION ==
codigoSession && p.CODIGO_EXPEDIENTE != codigo_expediente && (p.ESTADO
== "3" || p.ESTADO == "4" || p.ESTADO == "5")).ToList();
            foreach (var expe in expedientes)
            {
                GetExpedientesAnterioresList det = new GetExpedientesAnterioresList();
                det.codigo_expediente = expe.CODIGO_EXPEDIENTE;
                det.codigo_tipo_proyecto = expe.CODIGO_TIPO_PROYECTO;
                det.descripcion_tipo_proyecto = expe.TIPO_PROYECTO.DESCRIPCION;
                det.descripcion_tipo_proyecto_ingresado = expe.DESCRIPCION;
                expedienteList.Add(det);
            }
            return Json(expedienteList, JsonRequestBehavior.AllowGet);
        }
        public ActionResult ExpedienteIndex()
        {
            if (Session["Usuario"] == null)
            {return RedirectToAction("Index", "Login");}
            else
            {return View();}
        }
        public ActionResult ExpedienteEditar(string codigo_expediente)
        {
            if (Session["Usuario"] == null)
            {return RedirectToAction("Index", "Login");  }
            else
            {ViewBag.Codigo = codigo_expediente;
                string codigoSession = Session["Codigo"].ToString();
```

```
            List<EXPEDIENTE> expedientes = db.EXPEDIENTE.Where(p =>
p.CODIGO_EXPEDIENTE != codigo_expediente && p.USUARIO_CREACION ==
codigoSession && (p.ESTADO != "2" && p.ESTADO != "0")).ToList();
            ViewBag.CantidadExpedientes = expedientes.Count();
            return View("ExpedienteCreate");
        }       }
    public JsonResult ExpedienteList()
    {
        List<ExpedienteQuery> expedientesQuery = new List<ExpedienteQuery>();
        string codigo = Session["Codigo"].ToString();
        List<EXPEDIENTE> expedientes = db.EXPEDIENTE.Where(p => p.ESTADO
!= "0" && p.USUARIO_CREACION == codigo).ToList();
        foreach (var expediente in expedientes)
        {
            ExpedienteQuery eq = new ExpedienteQuery();
            eq.codigo_expediente = expediente.CODIGO_EXPEDIENTE;
            eq.tipo_proyecto = expediente.TIPO_PROYECTO.DESCRIPCION;
            eq.proyecto = expediente.DESCRIPCION;
            eq.fecha_expediente                                          =
expediente.FECHA_EXPEDIENTE.ToShortDateString()      +      "      "      +
expediente.FECHA_EXPEDIENTE.ToShortTimeString();
            eq.estado = expediente.ESTADO;
            List<ExpedienteSolicitudGeneradaQuery> solicitudGeneradaList = new
List<ExpedienteSolicitudGeneradaQuery>();
            List<SOLICITUD> solicitud = db.SOLICITUD.Where(p =>
p.CODIGO_EXPEDIENTE == expediente.CODIGO_EXPEDIENTE && (p.ESTADO
!= "0" && p.ESTADO != "9")).ToList();
            foreach (var soli in solicitud)
            {
                ExpedienteSolicitudGeneradaQuery solicitudGenerada = new
ExpedienteSolicitudGeneradaQuery();
                solicitudGenerada.codigo_expediente                          =
expediente.CODIGO_EXPEDIENTE;
                solicitudGenerada.codigo_solicitud = soli.CODIGO_SOLICITUD;
                solicitudGenerada.tipo_solicitud                             =
soli.TIPO_SOLICITUD.DESCRIPCION;
```

```
                solicitudGenerada.estado_solicitud                                    =
EstadoSolicitudDevolver(soli.ESTADO);
                solicitudGenerada.fecha_solicitud                                     =
soli.FECHA_ULTIMO_ESTADO.ToShortDateString();
                solicitudGeneradaList.Add(solicitudGenerada);
            }
            eq.solicitudes = solicitudGeneradaList;
            if ((expediente.FECHA_FIN < DateTime.Now) && (expediente.ESTADO ==
"3" || expediente.ESTADO == "4" || expediente.ESTADO == "1"))
            {
                string codigo_expediente = expediente.CODIGO_EXPEDIENTE;
                EXPEDIENTE                              exp                           =
db.EXPEDIENTE.Find(expediente.CODIGO_EXPEDIENTE);
                exp.ESTADO = 2.ToString(); // ELIMINADO - ANULADO
                exp.FECHA_EDICION = DateTime.Now;
                exp.USUARIO_EDICION = "GTIPROYECTOS";
                db.SaveChanges();
                List<EXPEDIENTE_DETALLE_REPRESENTAN>  detRepresentante  =
db.EXPEDIENTE_DETALLE_REPRESENTAN.Where(p                                     =>
p.CODIGO_EXPEDIENTE == codigo_expediente).ToList();
                foreach (var det in detRepresentante)
                {
                  det.ESTADO = 0;
                  det.USUARIO_EDICION = "GTIPROYECTOS";
                  det.FECHA_EDICION = DateTime.Now;
                  db.SaveChanges();
                }
                List<EXPEDIENTE_DET_REPRE_HISTORICO>
detRepresentanteHistorico  =  db.EXPEDIENTE_DET_REPRE_HISTORICO.Where(p
=> p.CODIGO_EXPEDIENTE == codigo_expediente && p.ESTADO == 1).ToList();
                foreach (var det in detRepresentanteHistorico)
                {
                  det.ESTADO = 0;
                  det.FECHA_EDICION = DateTime.Now;
                  det.USUARIO_EDICION = "GTIPROYECTOS";
                  db.SaveChanges();
                }
```

```
                // ASIGNAR ARCHIVOS
                List<EXPEDIENTE_DETALLE_ADJUNTO> detArchivos = db.EXPEDIENTE_DETALLE_ADJUNTO.Where(p => p.CODIGO_EXPEDIENTE == codigo_expediente && p.ESTADO == 1).ToList();
                foreach (var det in detArchivos)
                {
                    det.ESTADO = 0;
                    det.USUARIO_EDICION = "GTIPROYECTOS";
                    det.FECHA_EDICION = DateTime.Now;
                    db.SaveChanges();
                }               }
            expedientesQuery.Add(eq);
        }           return Json(expedientesQuery);
    }
    public JsonResult ExpedienteListSolicitudCreate()
    {
        List<ExpedienteQuery> expedientesQuery = new List<ExpedienteQuery>();
        string codigoSession = Session["Codigo"].ToString();
        List<EXPEDIENTE> expedientes = db.EXPEDIENTE.Where(p => p.ESTADO != "5" && p.ESTADO != "0" && p.ESTADO != "2" && p.USUARIO_CREACION == codigoSession).ToList();
        foreach (var expediente in expedientes)
        {
            ExpedienteQuery eq = new ExpedienteQuery();
            eq.codigo_expediente = expediente.CODIGO_EXPEDIENTE;
            eq.tipo_proyecto = expediente.TIPO_PROYECTO.DESCRIPCION;
            eq.proyecto = expediente.DESCRIPCION;
            eq.fecha_expediente = expediente.FECHA_EXPEDIENTE.ToShortDateString();
            eq.estado = expediente.ESTADO;
            expedientesQuery.Add(eq);
        }
        return Json(expedientesQuery);
    }
    [HttpPost]
    public JsonResult ExpedientesCrear(ExpedienteInput input)
```

```
{
    Status status = new Status();
    List<RepresentanteInput> representanteInput = input.representantes;
    string rpta_estado = "Ok";
    string rpta_mensaje = "";
    foreach (var representante in representanteInput)
    {
    if (rpta_estado.Equals("Ok"))
    {
      if (input.codigo_inicial == null)
      {
        EXPEDIENTE expediente = new EXPEDIENTE();
        string codigo_expediente = GenerarCodigoExpediente();
        DateTime fechaActual = DateTime.Now;
        expediente.CODIGO_EXPEDIENTE = codigo_expediente;
        expediente.DESCRIPCION = input.descripcion;
        expediente.CODIGO_TIPO_PROYECTO = input.codigo_tipo_proyecto;
        expediente.FECHA_EXPEDIENTE = fechaActual;
        expediente.FECHA_INICIO = fechaActual; //DATOS DE INICIO PARA
NO GUARDARLO EN NULO
        expediente.FECHA_FIN = fechaActual.AddYears(3); // DATOS DE
INICIO PARA NO GUARDARLO EN NULO
        expediente.ESTADO = 1.ToString();          // ESTADO = 1 - CREADO
        expediente.UBICACION_GEOGRAFICA = input.ubigeo;
        expediente.COORDENADA_X = 0;
        expediente.COORDENADA_Y = 0;
        expediente.OBJETIVO_PROYECTO = input.objetivo;
        expediente.MAXIMA_DEMANDA = input.maxima_demanda;
        expediente.TIENE_REPRESENTANTE = input.tiene_representante;
        expediente.EXPEDIENTE_RELACIONADO                              =
input.codigo_expediente_relacionado;
        expediente.USUARIO_CREACION = Session["Codigo"].ToString();
        expediente.FECHA_CREACION = DateTime.Now;
        expediente.USUARIO_EDICION = Session["Codigo"].ToString();
        expediente.FECHA_EDICION = DateTime.Now;
        try
        {
```

```
                db.EXPEDIENTE.Add(expediente);
                db.SaveChanges();
                string codigo_usuario = Session["Codigo"].ToString();
                USUARIO usuario = db.USUARIO.Find(codigo_usuario);
                string nombreCompleto = usuario.NOMBRE + " " +
usuario.APELLIDO_PATERNO + " " + usuario.APELLIDO_MATERNO;
                string descripcion_tipo_proyecto = "";
                TIPO_PROYECTO tipoProyecto =
db.TIPO_PROYECTO.Find(input.codigo_tipo_proyecto);
                descripcion_tipo_proyecto = tipoProyecto.DESCRIPCION;
                 EnviarMailExpedienteCreado(usuario.CORREO_ELECTRONICO,
nombreCompleto, codigo_expediente, descripcion_tipo_proyecto);
                string mensaje_representante = "";
                if (input.tiene_representante == 1)
                {
                  List<RepresentanteInput> listRepresentantes = new
List<RepresentanteInput>();
                  listRepresentantes = input.representantes;
                  int correlativo =
db.EXPEDIENTE_DETALLE_REPRESENTAN.Count();
                  if (correlativo == 0)
                  {correlativo = 0; }
                  else if (correlativo != 0)
                  {correlativo = db.EXPEDIENTE_DET_REPRE_HISTORICO.Max(p
=> p.CORRELATIVO); }
                  foreach (var represen in listRepresentantes)
                  {
                    EXPEDIENTE_DETALLE_REPRESENTAN detRepresentante =
new EXPEDIENTE_DETALLE_REPRESENTAN();
                    correlativo++;
                    if (represen.telefono_fijo == null)
                    {represen.telefono_fijo = string.Empty;
                    detRepresentante.CODIGO_EXPEDIENTE = codigo_expediente;
                    detRepresentante.CORRELATIVO = correlativo;
                    detRepresentante.TIPO_REPRESENTANTE =
represen.tipo_representante;
```

```
                    detRepresentante.NOMBRE_REPRESENTANTE =
represen.nombre_representante;
                    detRepresentante.NUMERO_CIP = represen.numero_cip;
                    detRepresentante.TELEFONO_FIJO = represen.telefono_fijo;
                    detRepresentante.TELEFONO_CELULAR =
represen.telefono_celular;
                    detRepresentante.CORREO_ELECTRONICO =
represen.correo_electronico;
                    detRepresentante.ESTADO = 1;
                    detRepresentante.UBIGEO = represen.ubigeo;
                    detRepresentante.DIRECCION_POSTAL =
represen.direccion_postal;
                    detRepresentante.USUARIO_CREACION =
Session["Codigo"].ToString();
                    detRepresentante.FECHA_CREACION = DateTime.Now;
                    detRepresentante.USUARIO_EDICION =
Session["Codigo"].ToString();
                    detRepresentante.FECHA_EDICION = DateTime.Now;
db.EXPEDIENTE_DETALLE_REPRESENTAN.Add(detRepresentante);
                    db.SaveChanges();
                    mensaje_representante = ". Representante_Guardado";
                    EXPEDIENTE_DET_REPRE_HISTORICO
detRepresentanteHistorico = new EXPEDIENTE_DET_REPRE_HISTORICO();
                    detRepresentanteHistorico.CODIGO_EXPEDIENTE =
codigo_expediente;
                    detRepresentanteHistorico.CORRELATIVO = correlativo;
                    detRepresentanteHistorico.NOMBRE_REPRESENTANTE =
represen.nombre_representante;
                    detRepresentanteHistorico.NUMERO_CIP = represen.numero_cip;
                    detRepresentanteHistorico.TELEFONO_CELULAR =
represen.telefono_celular;
                    detRepresentanteHistorico.CORREO_ELECTRONICO =
represen.correo_electronico;
                    detRepresentanteHistorico.ESTADO = 1; // ACTIVO
                    detRepresentanteHistorico.USUARIO_CREACION =
Session["Codigo"].ToString();
                    detRepresentanteHistorico.FECHA_CREACION = DateTime.Now;
```

```
                    detRepresentanteHistorico.USUARIO_EDICION =
Session["Codigo"].ToString();
                    detRepresentanteHistorico.FECHA_EDICION = DateTime.Now;
db.EXPEDIENTE_DET_REPRE_HISTORICO.Add(detRepresentanteHistorico);
                    db.SaveChanges();
                }
            }
            string mensaje_estado_expediente = "";
            EXPEDIENTE_ESTADO_HISTORICO estado = new
EXPEDIENTE_ESTADO_HISTORICO();
            int correlativoEstado =
db.EXPEDIENTE_ESTADO_HISTORICO.Count();
            if (correlativoEstado == 0)
            {correlativoEstado = correlativoEstado + 1; }
            else
            {correlativoEstado = db.EXPEDIENTE_ESTADO_HISTORICO.Max(p
=> p.CORRELATIVO) + 1;
            }
            estado.CODIGO_EXPEDIENTE = codigo_expediente;
            estado.CORRELATIVO = correlativoEstado;
            estado.ESTADO_ANTERIOR = "Creado";
            estado.USUARIO_CREACION = Session["Codigo"].ToString();
            estado.FECHA_CREACION = DateTime.Now;
            estado.USUARIO_EDICION = Session["Codigo"].ToString();
            estado.FECHA_EDICION = DateTime.Now;
            db.EXPEDIENTE_ESTADO_HISTORICO.Add(estado);
            db.SaveChanges();
            mensaje_estado_expediente = ". Expediente Estado Guardado";
            string codigo_proyecto = input.codigo_tipo_proyecto;
            int id = db.EXPEDIENTE_DETALLE_PROCESO.Count();
            if (id == 0)
            {id = 0; }
            else
            {id = db.EXPEDIENTE_DETALLE_PROCESO.Max(p =>
p.ID_EXPEDIENTE_DETALLE_PROCESO); }
            List<TIPO_PROYECTO_TIPO_SOLICITUD> tipos_solicitudes = new
List<TIPO_PROYECTO_TIPO_SOLICITUD>();
```

```
                tipos_solicitudes = db.TIPO_PROYECTO_TIPO_SOLICITUD.Where(p
=> p.ESTADO == 1 && p.CODIGO_TIPO_PROYECTO ==
input.codigo_tipo_proyecto).OrderBy(p => p.ORDEN).ToList();
                string cantidad_solicitudes = tipos_solicitudes.Count.ToString();
                int primerRegistroAtender = 0;
string detalle_proceso = "";
                foreach (var tipo_solicitud in tipos_solicitudes)
                {
                   id++;
                   primerRegistroAtender++;
                   EXPEDIENTE_DETALLE_PROCESO expedienteDetalleProceso =
new EXPEDIENTE_DETALLE_PROCESO();
                   expedienteDetalleProceso.ID_EXPEDIENTE_DETALLE_PROCESO
= id;
                   expedienteDetalleProceso.CODIGO_EXPEDIENTE =
codigo_expediente;
                   expedienteDetalleProceso.CODIGO_TIPO_PROYECTO =
input.codigo_tipo_proyecto;
                   expedienteDetalleProceso.ES_OMITIDO = 0;
                   expedienteDetalleProceso.USUARIO_OMITE =
Session["Codigo"].ToString();
                   expedienteDetalleProceso.USUARIO_ATIENDE =
Session["Codigo"].ToString();
                   expedienteDetalleProceso.USUARIO_ASIGNA =
Session["Codigo"].ToString();
                   if (primerRegistroAtender == 1)
                   {
                      expedienteDetalleProceso.ESTADO = 1; // 1: SOLICITUD
PENDIENTE -  ES UN ESTADO INTERNO, PROPIO DEL SISTEMA
                   }
                   else
                   {expedienteDetalleProceso.ESTADO = 0; }
                   expedienteDetalleProceso.CODIGO_TIPO_SOLICITUD =
tipo_solicitud.CODIGO_TIPO_SOLICITUD;
                   expedienteDetalleProceso.CANTIDAD_OPCION =
tipo_solicitud.CANTIDAD_SOLICITUD_SALTO;
```

```
                    expedienteDetalleProceso.ORDEN = tipo_solicitud.ORDEN;
                    expedienteDetalleProceso.USUARIO_CREACION =
Session["Codigo"].ToString();
                    expedienteDetalleProceso.FECHA_CREACION = DateTime.Now;
                    expedienteDetalleProceso.USUARIO_EDICION =
Session["Codigo"].ToString();
                    expedienteDetalleProceso.FECHA_EDICION = DateTime.Now;
                    detalle_proceso = ".Se guardó
EXPEDIENTE_DETALLE_PROCESO";
db.EXPEDIENTE_DETALLE_PROCESO.Add(expedienteDetalleProceso);
                    db.SaveChanges();
                }
                status.codigo_entidad = codigo_expediente;
                status.rpta_estado = "Ok";
                status.rpta_mensaje = "Se creó Expediente satisfactoriamente.";
            }
            catch (Exception ex)
            {
                status.rpta_estado = "Error";
                status.rpta_mensaje = "Error al crear expediente.";
            }          }
        else
        {
            EXPEDIENTE expediente = db.EXPEDIENTE.Find(input.codigo_inicial);
            string codigo_expediente = input.codigo_inicial;
            DateTime fechaActual = DateTime.Now;
            try
            {
                if (input.tiene_representante == 1)
                {
                    Int correlativo =
db.EXPEDIENTE_DET_REPRE_HISTORICO.Max(p => p.CORRELATIVO);
                    List<RepresentanteInput> listRepresentantes = new
List<RepresentanteInput>();
                    listRepresentantes = input.representantes;
List<EXPEDIENTE_DETALLE_REPRESENTAN> listarRepresentantes =
```

```
db.EXPEDIENTE_DETALLE_REPRESENTAN.Where(p =>
p.CODIGO_EXPEDIENTE == codigo_expediente).ToList();
                    foreach (var rep in listarRepresentantes)
                    {
                        int registros = listRepresentantes.Where(p => p.codigo_expediente
== codigo_expediente && p.correlativo == rep.CORRELATIVO).Count();
                        if (registros == 0)
                        {
                            rep.ESTADO = 0;
                            rep.USUARIO_EDICION = Session["Codigo"].ToString();
                            rep.FECHA_EDICION = DateTime.Now;
                            db.SaveChanges();
                            string codigo_usuario = Session["Codigo"].ToString();
                            USUARIO usuario = db.USUARIO.Find(codigo_usuario);
                            string nombreCompleto = usuario.NOMBRE + " " +
usuario.APELLIDO_PATERNO + " " + usuario.APELLIDO_MATERNO;
                            string representante = rep.NOMBRE_REPRESENTANTE;
                            List<EXPEDIENTE_DET_REPRE_HISTORICO> historico =
new List<EXPEDIENTE_DET_REPRE_HISTORICO>();
                            historico =
db.EXPEDIENTE_DET_REPRE_HISTORICO.Where(p =>
p.CODIGO_EXPEDIENTE == codigo_expediente && p.CORRELATIVO ==
rep.CORRELATIVO).ToList();
                            foreach (var his in historico)
                            {
                                his.ESTADO = 0; // INACTIVO
                                his.USUARIO_EDICION = Session["Codigo"].ToString();
                                his.FECHA_EDICION = DateTime.Now;
                                db.SaveChanges();
                            }                        }                    }
                    foreach (var listint in listRepresentantes)
                    {
if (listint.correlativo != 0)
                        {
                            List<EXPEDIENTE_DETALLE_REPRESENTAN>
listintRepresentantes = db.EXPEDIENTE_DETALLE_REPRESENTAN.Where(p =>
```

```
p.CODIGO_EXPEDIENTE == codigo_expediente && p.CORRELATIVO ==
listint.correlativo && listint.correlativo != 0).ToList();
                        if (listintRepresentantes.Count == 0)
                        {
                            List<EXPEDIENTE_DET_REPRE_HISTORICO> historico =
new List<EXPEDIENTE_DET_REPRE_HISTORICO>();
                            historico                                        =
db.EXPEDIENTE_DET_REPRE_HISTORICO.Where(p                                    =>
p.CODIGO_EXPEDIENTE == codigo_expediente && p.CORRELATIVO ==
listint.correlativo).ToList();

                            foreach (var his in historico)
                            {
his.NOMBRE_REPRESENTANTE = listint.nombre_representante;
                                his.NUMERO_CIP = listint.numero_cip;
                                his.TELEFONO_CELULAR = listint.telefono_celular;
                                his.CORREO_ELECTRONICO = listint.correo_electronico;
                                his.ESTADO = 0; // INACTIVO
                                his.USUARIO_EDICION = Session["Codigo"].ToString();
                                his.FECHA_EDICION = DateTime.Now;
db.SaveChanges();
                            }                           }
                        else
                        {
                            foreach (var lisRep in listintRepresentantes)
                            {
                                if (listint.telefono_fijo == null)
                                {listint.telefono_fijo = string.Empty; }
                                lisRep.TIPO_REPRESENTANTE                        =
listint.tipo_representante;
                                lisRep.NOMBRE_REPRESENTANTE                      =
listint.nombre_representante;
                                lisRep.NUMERO_CIP = listint.numero_cip;
                                lisRep.TELEFONO_FIJO = listint.telefono_fijo;
                                lisRep.TELEFONO_CELULAR = listint.telefono_celular;
                                lisRep.CORREO_ELECTRONICO                        =
listint.correo_electronico;
```

```
                    lisRep.UBIGEO = listint.ubigeo;
                    lisRep.ESTADO = 1;
                    lisRep.DIRECCION_POSTAL = listint.direccion_postal;
                    lisRep.USUARIO_EDICION =
Session["Codigo"].ToString();
                    lisRep.FECHA_EDICION = DateTime.Now;
                    db.SaveChanges();
                    List<EXPEDIENTE_DET_REPRE_HISTORICO> historico
= new List<EXPEDIENTE_DET_REPRE_HISTORICO>();
                    historico =
db.EXPEDIENTE_DET_REPRE_HISTORICO.Where(p =>
p.CODIGO_EXPEDIENTE == codigo_expediente && p.CORRELATIVO ==
listint.correlativo).ToList();

                    foreach (var his in historico)
                    {
                      his.NOMBRE_REPRESENTANTE =
listint.nombre_representante;
                      his.NUMERO_CIP = listint.numero_cip;
                      his.TELEFONO_CELULAR = listint.telefono_celular;
                      his.CORREO_ELECTRONICO =
listint.correo_electronico;
                      his.ESTADO = 1;
                      his.USUARIO_EDICION =
Session["Codigo"].ToString();
                      his.FECHA_EDICION = DateTime.Now;
                      db.SaveChanges();
                    }     }          }                    }
                else
                {
                  EXPEDIENTE_DETALLE_REPRESENTAN detRepresentante
= new EXPEDIENTE_DETALLE_REPRESENTAN();
                  correlativo++;
                  if (listint.telefono_fijo == null)
                  {   listint.telefono_fijo = string.Empty;                }
                  detRepresentante.CODIGO_EXPEDIENTE = codigo_expediente;
                  detRepresentante.CORRELATIVO = correlativo;
```

```
                    detRepresentante.TIPO_REPRESENTANTE = listint.tipo_representante;
                    detRepresentante.NOMBRE_REPRESENTANTE = listint.nombre_representante;
                    detRepresentante.NUMERO_CIP = listint.numero_cip;
                    detRepresentante.TELEFONO_FIJO = listint.telefono_fijo;
                    detRepresentante.TELEFONO_CELULAR = listint.telefono_celular;
                    detRepresentante.CORREO_ELECTRONICO = listint.correo_electronico;
                    detRepresentante.ESTADO = 1;
                    detRepresentante.UBIGEO = listint.ubigeo;
                    detRepresentante.DIRECCION_POSTAL = listint.direccion_postal;
                    detRepresentante.USUARIO_CREACION = Session["Codigo"].ToString();
                    detRepresentante.FECHA_CREACION = DateTime.Now;
                    detRepresentante.USUARIO_EDICION = Session["Codigo"].ToString();
                    detRepresentante.FECHA_EDICION = DateTime.Now;
db.EXPEDIENTE_DETALLE_REPRESENTAN.Add(detRepresentante);
                    db.SaveChanges();
                    EXPEDIENTE_DET_REPRE_HISTORICO detRepresentanteHistorico = new EXPEDIENTE_DET_REPRE_HISTORICO();
                    detRepresentanteHistorico = new EXPEDIENTE_DET_REPRE_HISTORICO();
                    detRepresentanteHistorico.CODIGO_EXPEDIENTE = codigo_expediente;
                    detRepresentanteHistorico.CORRELATIVO = correlativo;
                    detRepresentanteHistorico.NOMBRE_REPRESENTANTE = listint.nombre_representante;
                    detRepresentanteHistorico.NUMERO_CIP = listint.numero_cip;
                    detRepresentanteHistorico.TELEFONO_CELULAR = listint.telefono_celular;
                    detRepresentanteHistorico.CORREO_ELECTRONICO = listint.correo_electronico;
                    detRepresentanteHistorico.ESTADO = 1;
```

```
                    detRepresentanteHistorico.USUARIO_CREACION =
Session["Codigo"].ToString();
                    detRepresentanteHistorico.FECHA_CREACION =
DateTime.Now;
                    detRepresentanteHistorico.USUARIO_EDICION =
Session["Codigo"].ToString();
                    detRepresentanteHistorico.FECHA_EDICION = DateTime.Now;

db.EXPEDIENTE_DET_REPRE_HISTORICO.Add(detRepresentanteHistorico);
                    db.SaveChanges();
                }               }               }               else
            {
                List<EXPEDIENTE_DETALLE_REPRESENTAN> detRepresentante
= new List<EXPEDIENTE_DETALLE_REPRESENTAN>();
                detRepresentante =
db.EXPEDIENTE_DETALLE_REPRESENTAN.Where(p =>
p.CODIGO_EXPEDIENTE == input.codigo_inicial).ToList();
                foreach (var det in detRepresentante)
                {
                    string codigo_usuario = Session["Codigo"].ToString();
                    USUARIO usuario = db.USUARIO.Find(codigo_usuario);
                    string nombreCompleto = usuario.NOMBRE + " " +
usuario.APELLIDO_PATERNO + " " + usuario.APELLIDO_MATERNO;
                    string representante = det.NOMBRE_REPRESENTANTE;
                    det.ESTADO = 0;
                    det.USUARIO_EDICION = Session["Codigo"].ToString();
                    det.FECHA_EDICION = DateTime.Now;
                    db.SaveChanges();
                }               }       }
    // VISTA INDIVIDUAL
    //[HttpGet]
    public JsonResult ExpedienteLeer(string codigo)
    {
        EXPEDIENTE exp = db.EXPEDIENTE.Find(codigo);
        ExpedienteInd expedienteInd = new ExpedienteInd();
        String codigoSession = Session["Codigo"].ToString();
        if (exp.USUARIO_CREACION == codigoSession)
```

```
        {
            if (exp.ESTADO == "0")
            {
                expedienteInd.rpta_estado = "Error";
                expedienteInd.rpta_mensaje = "Usted no puede ver este Expediente.";
            }
            else
            {
                expedienteInd.codigo_expediente = exp.CODIGO_EXPEDIENTE;
                expedienteInd.descripcion = exp.DESCRIPCION;
                expedienteInd.codigo_tipo_proyecto = exp.CODIGO_TIPO_PROYECTO;
                expedienteInd.ubigeo = exp.UBICACION_GEOGRAFICA;
                expedienteInd.latitud = 0;
                expedienteInd.longitud = 0;
                expedienteInd.objetivo = exp.OBJETIVO_PROYECTO;
                expedienteInd.maxima_demanda =
Int32.Parse(exp.MAXIMA_DEMANDA.ToString());
                expedienteInd.tiene_representante =
Int32.Parse(exp.TIENE_REPRESENTANTE.ToString());
                expedienteInd.fechaInicio = exp.FECHA_INICIO.ToShortDateString();
                expedienteInd.fechaFin = exp.FECHA_FIN.ToShortDateString();
                expedienteInd.fechaExpediente =
exp.FECHA_EXPEDIENTE.ToShortDateString() + " " +
exp.FECHA_EXPEDIENTE.ToShortTimeString();
                expedienteInd.estadoExpediente = exp.ESTADO;
                expedienteInd.codigo_expediente_relacionado =
exp.EXPEDIENTE_RELACIONADO;
                expedienteInd.cantidad_solicitud_creada = db.SOLICITUD.Where(p =>
p.CODIGO_EXPEDIENTE == codigo && (p.ESTADO != "0" && p.ESTADO !=
"9")).Count();
                List<RepresentanteExpedienteQuery> representantes = new
List<RepresentanteExpedienteQuery>();
                List<ArchivoExpedienteQuery> archivos = new
List<ArchivoExpedienteQuery>();
                List<EstadoExpedienteQuery> estados = new
List<EstadoExpedienteQuery>();
```

```
            List<UsuarioInternoNotaQuery> notasUsuario = new
List<UsuarioInternoNotaQuery>();
            List<EXPEDIENTE_DETALLE_REPRESENTAN> detRepresentante =
db.EXPEDIENTE_DETALLE_REPRESENTAN.Where(p =>
p.CODIGO_EXPEDIENTE == codigo).ToList();
            foreach (var det in detRepresentante)
            {
              RepresentanteExpedienteQuery representante = new
RepresentanteExpedienteQuery();
              representante.codigo_expediente = det.CODIGO_EXPEDIENTE;
              representante.correlativo = det.CORRELATIVO;
              representante.tipo_representante = det.TIPO_REPRESENTANTE;
              representante.nombre_representante =
det.NOMBRE_REPRESENTANTE;
              representante.numero_cip = det.NUMERO_CIP;
              representante.telefono_celular = det.TELEFONO_CELULAR;
              representante.telefono_fijo = det.TELEFONO_FIJO;
              representante.correo_electronico = det.CORREO_ELECTRONICO;
              //representante.codigo_pais = det.UBIGEO;
              representante.ubigeo = det.UBIGEO;
              representante.direccion_postal = det.DIRECCION_POSTAL;
              representantes.Add(representante);
            }
            // ASIGNAR ARCHIVOS
            List<EXPEDIENTE_DETALLE_ADJUNTO> detArchivos =
db.EXPEDIENTE_DETALLE_ADJUNTO.Where(p => p.CODIGO_EXPEDIENTE ==
codigo && p.ESTADO == 1).ToList();
            string rutaLectura =
ConfigurationManager.AppSettings["RutaLeerArchivos"].ToString();
            foreach (var det in detArchivos)
            {
              ArchivoExpedienteQuery archivo = new ArchivoExpedienteQuery();
              archivo.contenido = rutaLectura + "/" + det.CONTENIDO;
              archivo.secuencia = det.CORRELATIVO;
              archivo.estado = det.ESTADO;
              archivo.nombre_archivo = det.NOMBRE_ARCHIVO;
              archivos.Add(archivo);
```

```
            }
            // ASIGNAR ESTADOS
            List<EXPEDIENTE_ESTADO_HISTORICO> detEstados = db.EXPEDIENTE_ESTADO_HISTORICO.Where(p => p.CODIGO_EXPEDIENTE == codigo).ToList();
            foreach (var det in detEstados)
            {
               EstadoExpedienteQuery estado = new EstadoExpedienteQuery();
               estado.codigo_expediente = det.CODIGO_EXPEDIENTE;
               estado.fecha_estado = det.FECHA_CREACION.ToShortDateString();
               estado.estado = det.ESTADO_ANTERIOR;
               estados.Add(estado);
            }
          return Json(expedienteInd);
        }
        else
        {
          expedienteInd.rpta_estado = "Error";
          expedienteInd.rpta_mensaje = "Usted no tiene permisos para ver este Expediente.";
           return Json(expedienteInd);
        }      }
    private string GenerarCodigoExpediente()
    {
        string codigo = "";
        string prefijo = "EXP";
        int anio = DateTime.Now.Year;
        int contador = db.EXPEDIENTE.Where(p => p.FECHA_CREACION.Year == DateTime.Now.Year).ToList().Count + 1;

        codigo = prefijo + "-" + anio + "-" + contador.ToString("D6");
        return codigo;
    }
```

Registro de Solicitudes – Tercero

```
using ProyectoTerceros.Models;
using ProyectoTerceros.Models.Administracion;
using ProyectoTerceros.Models.Registro.UsuarioExterno;
```

```
using ProyectoTerceros.Models.Registro.UsuarioInterno;
using ProyectoTerceros.Models.Seguridad;
using System;
using System.Collections.Generic;
using System.Configuration;
using System.IO;
using System.Linq;
using System.Net.Http;
using System.Net.Http.Headers;
using System.Text.RegularExpressions;
using System.Threading.Tasks;
using System.Web;
using System.Web.Mvc;
namespace ProyectoTerceros.Controllers.Seguridad
{
   public class UsuarioExternoController : Controller
   {
      // GET: Usuario
      private ProyectoTercerosEntities db = new ProyectoTercerosEntities();
      #region SOLICITUD
      public ActionResult SolicitudCreate()
      {
         if (Session["Usuario"] == null)
         {return RedirectToAction("Index", "Login");}
         else
         {ViewBag.Codigo = ""; return View();           }
      }
      public ActionResult SolicitudIndex()
      {
         if (Session["Usuario"] == null)
         {return RedirectToAction("Index", "Login");           }
         else
         {return View();}
      }
      public ActionResult SolicitudEditar(string codigo_solicitud)
      {
         if (Session["Usuario"] == null)
```

```
            {return RedirectToAction("Index", "Login");}
            else
            {ViewBag.Codigo = codigo_solicitud; return View("SolicitudCreate");          }
        }
        public JsonResult SolicitudList()
        {
            List<SolicitudQuery> solicitudesQuery = new List<SolicitudQuery>();
            string codigo = Session["Codigo"].ToString();
            List<SOLICITUD> solicitudes = db.SOLICITUD.Where(p => p.ESTADO != "0"
&& p.USUARIO_CREACION == codigo).ToList();
            foreach (var solicitud in solicitudes)
            {
                SolicitudQuery solQuery = new SolicitudQuery();
                solQuery.codigo_solicitud = solicitud.CODIGO_SOLICITUD;
                solQuery.asunto_solicitud = solicitud.ASUNTO;
                solQuery.codigo_expediente = solicitud.CODIGO_EXPEDIENTE;
                solQuery.tipo_proyecto_expediente                                    =
solicitud.EXPEDIENTE.TIPO_PROYECTO.DESCRIPCION;
                solQuery.descripcion_expediente = solicitud.EXPEDIENTE.DESCRIPCION;
                solQuery.descripcion_tipo_solicitud                                  =
solicitud.TIPO_SOLICITUD.DESCRIPCION;
                solQuery.fecha_solicitud                                             =
solicitud.FECHA_SOLICITUD.ToShortDateString()        +        "        "        +
solicitud.FECHA_SOLICITUD.ToShortTimeString();
                solQuery.estado = Int32.Parse(solicitud.ESTADO);
                solicitudesQuery.Add(solQuery);
            }
            return Json(solicitudesQuery);
        }
        public JsonResult GetSolicitudCorrespondiente(string codigo_expediente)
        {
            // EXPEDIENTE
            UsuarioExternoSolicitudAtenderGetList usuarioExternoSolicitudAtenderGetList
= new UsuarioExternoSolicitudAtenderGetList();
            EXPEDIENTE expediente = db.EXPEDIENTE.Find(codigo_expediente);
            // OBTENER CODIGO DE LA SOLICITUD
            // SOLICITUD RECHAZADA
```

```
        List<EXPEDIENTE_DETALLE_PROCESO> listDetalle =
db.EXPEDIENTE_DETALLE_PROCESO.Where(p => p.CODIGO_EXPEDIENTE ==
codigo_expediente && p.ESTADO == 7).OrderBy(p => p.ORDEN).ToList();
        if (listDetalle.Count == 0)
        {
          listDetalle = db.EXPEDIENTE_DETALLE_PROCESO.Where(p =>
p.CODIGO_EXPEDIENTE == codigo_expediente && p.ESTADO == 1).OrderBy(p =>
p.ORDEN).ToList();
        }
if (listDetalle.Count != 0)
        {
          string codigo_solicitud_correspondiente = "";
          List<SolicitudAtenderGet> solicitudes = new List<SolicitudAtenderGet>();
          foreach (var list in listDetalle)
          {
            codigo_solicitud_correspondiente = list.CODIGO_TIPO_SOLICITUD;
            List<REQUISITO_TIPO_SOLICITUD> requisitos =
db.REQUISITO_TIPO_SOLICITUD.Where(p =>
p.COD_REQUISITO_TIPO_SOLICITUD == codigo_solicitud_correspondiente &&
p.ESTADO == 1).OrderBy(p => p.SECUENCIA).ToList();
            TIPO_SOLICITUD tipoSolicitud =
db.TIPO_SOLICITUD.Find(codigo_solicitud_correspondiente);
            SolicitudAtenderGet solicitudAtenderGet = new SolicitudAtenderGet();
            solicitudAtenderGet.codigo_expediente = codigo_expediente;
            solicitudAtenderGet.descripcion_expediente = expediente.DESCRIPCION;
            solicitudAtenderGet.tipo_proyecto_de_expediente =
expediente.TIPO_PROYECTO.DESCRIPCION;
            solicitudAtenderGet.codigo_solicitud_atender =
codigo_solicitud_correspondiente;
            solicitudAtenderGet.codigo_tipo_sistema_electrico_solicitud_atender =
expediente.TIPO_PROYECTO.TIPO_SISTEMA_ELECTRICO.CODIGO_TIPO_SIST
EMA_ELECTRICO;
            solicitudAtenderGet.tipo_sistema_electrico_solicitud_atender =
expediente.TIPO_PROYECTO.TIPO_SISTEMA_ELECTRICO.DESCRIPCION;
            solicitudAtenderGet.plazo_atencion = tipoSolicitud.PLAZO_ATENCION;
            solicitudAtenderGet.descripcion_solicitud_atender =
tipoSolicitud.DESCRIPCION;
```

```
            List<RequisitoTipoSolicitudQuery> requisitosAtender = new
List<RequisitoTipoSolicitudQuery>();
            foreach (var requisito in requisitos)
            {
              RequisitoTipoSolicitudQuery raAtender = new
RequisitoTipoSolicitudQuery();
              raAtender.secuencia = requisito.SECUENCIA;
              raAtender.descripcion = requisito.DESCRIPCION;
              raAtender.estado = requisito.ESTADO;
              var secuencia = requisito.SECUENCIA.ToString();
              List<TIPO_SOLICITUD_REQ_FORMATO> formatos =
db.TIPO_SOLICITUD_REQ_FORMATO.Where(p => p.CODIGO_REQUISITO ==
secuencia && p.ESTADO == 1 && p.CODIGO_TIPO_SOLICITUD ==
codigo_solicitud_correspondiente).ToList();
              foreach (var formato in formatos)
              {
                raAtender.esAutocad = formato.ES_AUTOCAD;
                raAtender.esExcel = formato.ES_EXCEL;
                raAtender.esPdf = formato.ES_PDF;
                raAtender.esTexto = formato.ES_TEXTO;
                raAtender.esWord = formato.ES_WORD;
                break;
              }
              requisitosAtender.Add(raAtender);
            }
            solicitudAtenderGet.requisitos = requisitosAtender;
            solicitudes.Add(solicitudAtenderGet);
          }
          usuarioExternoSolicitudAtenderGetList.solicitudes = solicitudes;
          return Json(usuarioExternoSolicitudAtenderGetList);
        }
        else
        {
          string rpta_estado = "Error";
          string rpta_mensaje = "";
```

```
            List<EXPEDIENTE_DETALLE_PROCESO> detalle = db.EXPEDIENTE_DETALLE_PROCESO.Where(p => p.CODIGO_EXPEDIENTE == codigo_expediente && (p.ESTADO != 0 && p.ESTADO != 6)).OrderByDescending(p => p.ORDEN).ToList();
            foreach (var det in detalle)
            {
              List<SOLICITUD> solicitud = db.SOLICITUD.Where(p => p.CODIGO_EXPEDIENTE == codigo_expediente && p.ESTADO != "6" && p.CODIGO_TIPO_SOLICITUD == det.CODIGO_TIPO_SOLICITUD).ToList();
              foreach (var sol in solicitud)
              {
                rpta_mensaje = "Usted no puede crear otra solicitud, porque la solicitud '" + sol.CODIGO_SOLICITUD + "' está en el estado '" + EstadoSolicitudDevolver(sol.ESTADO) + "'.";
              }
              break;
            }
            usuarioExternoSolicitudAtenderGetList.rpta_estado = rpta_estado;
            usuarioExternoSolicitudAtenderGetList.rpta_mensaje = rpta_mensaje;
            return Json(usuarioExternoSolicitudAtenderGetList);
          }
        }
        [HttpPost]
        public JsonResult SolicitudCrear(SolicitudInput input)
        {
          Status status = new Status();
          if (status.rpta_estado.ToUpper().Equals("OK"))
          {
            if (input.codigo_inicial == null)
            {
              input.estado = 3;
              SOLICITUD solicitud = new SOLICITUD();
              string codigo_solicitud = GenerarCodigoSolicitud();
              DateTime fechaActual = DateTime.Now;
              // GUARDAR UN EXPEDIENTE
              DateTime fechaMaximaAtencion = DateTime.Now;
              int plazo_atencion = 0;
```

```
            List<TIPO_SOLICITUD> tipoSolicitudPlazoAtencion =
db.TIPO_SOLICITUD.Where(p => p.CODIGO_TIPO_SOLICITUD ==
input.codigo_tipo_solicitud).ToList();
            foreach (var tip in tipoSolicitudPlazoAtencion)
            {
               plazo_atencion = tip.PLAZO_ATENCION;
               break;
            }
            int contador = 1;
            DateTime fechaInterna = DateTime.Now;
            TimeSpan ts = new TimeSpan(0, 0, 0);
            fechaInterna = fechaInterna.Date + ts;
            DateTime fechaActualEvalua = DateTime.Now;
            fechaActualEvalua = fechaActualEvalua.Date + ts;
               fechaInterna = fechaInterna.AddDays(1);
            int kpi = 0;
            kpi = fechaMaximaAtencion.Subtract(fechaActual).Days;
            solicitud.CODIGO_SOLICITUD = codigo_solicitud;
            solicitud.ASUNTO = input.asunto_solicitud;
            solicitud.FECHA_SOLICITUD = fechaActual;
            solicitud.FECHA_MAXIMA_ATENCION = fechaMaximaAtencion;
            solicitud.PLAZO_ATENCION = input.plazo_atencion; ;
            solicitud.CODIGO_EXPEDIENTE = input.codigo_expediente; ;
            solicitud.CODIGO_TIPO_SOLICITUD = input.codigo_tipo_solicitud;
            solicitud.CODIGO_TIPO_SISTEMA_ELECTRICO =
input.codigo_tipo_sistema_electrico;
            solicitud.TRAMITADO_POR = Session["Codigo"].ToString();
            solicitud.TRAMITADO_POR_ORIGINAL =
Session["Codigo"].ToString();
            solicitud.COMENTARIOS = "Sin comentarios";
            solicitud.FECHA_ULTIMO_ESTADO = fechaActual;
            solicitud.KPI = kpi;
            solicitud.ESTADO = input.estado.ToString();
            solicitud.USUARIO_CREACION = Session["Codigo"].ToString();
            solicitud.FECHA_CREACION = DateTime.Now;
            solicitud.USUARIO_EDICION = Session["Codigo"].ToString();
```

```
                solicitud.FECHA_EDICION = DateTime.Now;
                db.SOLICITUD.Add(solicitud);
                db.SaveChanges();
            return Json(status);
        }
```

Revisión de Expedientes – Colaborador

```
using ProyectoTerceros.Models;
using ProyectoTerceros.Models.Registro.UsuarioExterno;
using ProyectoTerceros.Models.Registro.UsuarioInterno;
using ProyectoTerceros.Models.Seguridad;
using System;
using System.Collections.Generic;
using System.Configuration;
using System.IO;
using System.Linq;
using System.Net;
using System.Net.Http;
using System.Net.Http.Headers;
using System.Net.Mail;
using System.Text.RegularExpressions;
using System.Threading.Tasks;
using System.Web;
using System.Web.Mvc;
namespace ProyectoTerceros.Controllers.Seguridad
{
    public class UsuarioController : Controller
    {
        private ProyectoTercerosEntities db = new ProyectoTercerosEntities();
        #region EXPEDIENTE
        public ActionResult ExpedienteCreate()
        {
            if (Session["Usuario"] == null)
            {return RedirectToAction("Index", "Login");}
            else
            {return View();           }
        }
        public ActionResult ExpedienteIndex()
```

```
        {
            if (Session["Usuario"] == null)
            {return RedirectToAction("Index", "Login");}
            else
            {return View();}
        }
        public ActionResult ExpedienteRevisar(string codigo_expediente)
        {
            if (Session["Usuario"] == null)
            {return RedirectToAction("Index", "Login");}
            else
            {ViewBag.Codigo = codigo_expediente;
                return View("ExpedienteRevisar");
            }
        }
        public JsonResult ExpedienteList()
        {
            List<ExpedienteQuery> expedientesQuery = new List<ExpedienteQuery>();
            string codigo = Session["Codigo"].ToString();
            List<EXPEDIENTE> expedientes = db.EXPEDIENTE.Where(p => p.ESTADO
!= "0").ToList();
            foreach (var expediente in expedientes)
            {
                ExpedienteQuery eq = new ExpedienteQuery();
                eq.codigo_expediente = expediente.CODIGO_EXPEDIENTE;
                eq.tipo_proyecto = expediente.TIPO_PROYECTO.DESCRIPCION;
                eq.proyecto = expediente.DESCRIPCION;
                eq.fecha_expediente                                          =
expediente.FECHA_EXPEDIENTE.ToShortDateString()      +      "      "      +
expediente.FECHA_EXPEDIENTE.ToShortTimeString();
                eq.estado = expediente.ESTADO;
                List<ExpedienteSolicitudGeneradaQuery>  solicitudGeneradaList  =  new
List<ExpedienteSolicitudGeneradaQuery>();
                List<SOLICITUD>  solicitud  =  db.SOLICITUD.Where(p  =>
p.CODIGO_EXPEDIENTE == expediente.CODIGO_EXPEDIENTE && (p.ESTADO
!= "0" && p.ESTADO != "1" && p.ESTADO != "8")).ToList();
                foreach (var soli in solicitud)
```

```
            {
                ExpedienteSolicitudGeneradaQuery solicitudGenerada = new
ExpedienteSolicitudGeneradaQuery();
                solicitudGenerada.codigo_expediente =
expediente.CODIGO_EXPEDIENTE;
                solicitudGenerada.codigo_solicitud = soli.CODIGO_SOLICITUD;
                solicitudGenerada.tipo_solicitud =
soli.TIPO_SOLICITUD.DESCRIPCION;
                solicitudGenerada.estado_solicitud =
EstadoSolicitudDevolver(soli.ESTADO);
                solicitudGenerada.fecha_solicitud =
soli.FECHA_ULTIMO_ESTADO.ToShortDateString();
                solicitudGeneradaList.Add(solicitudGenerada);
            }
            eq.solicitudes = solicitudGeneradaList;
return Json(expedientesQuery);
 }
      public JsonResult ExpedienteLeer(string codigo)
      {EXPEDIENTE exp = db.EXPEDIENTE.Find(codigo);
        ExpedienteInd expedienteInd = new ExpedienteInd();
expedienteInd.codigo_expediente = exp.CODIGO_EXPEDIENTE;
        expedienteInd.descripcion = exp.DESCRIPCION;
        expedienteInd.codigo_tipo_proyecto = exp.CODIGO_TIPO_PROYECTO;
        expedienteInd.ubigeo = exp.UBICACION_GEOGRAFICA;
        expedienteInd.latitud = exp.COORDENADA_Y;
        expedienteInd.longitud = exp.COORDENADA_X;
        expedienteInd.objetivo = exp.OBJETIVO_PROYECTO;
        expedienteInd.maxima_demanda = (Int32)exp.MAXIMA_DEMANDA;
        expedienteInd.tiene_representante = (Int32)exp.TIENE_REPRESENTANTE;
        expedienteInd.codigo_expediente_relacionado =
exp.EXPEDIENTE_RELACIONADO;
        if (exp.EXPEDIENTE_RELACIONADO != "" &&
exp.EXPEDIENTE_RELACIONADO != null)
        {
          EXPEDIENTE expedienteRelacionado =
db.EXPEDIENTE.Find(exp.EXPEDIENTE_RELACIONADO);
```

```
            expedienteInd.tipo_proyecto_relacionado =
expedienteRelacionado.TIPO_PROYECTO.DESCRIPCION;
            expedienteInd.descripcion_tipo_proyecto_ingresado =
expedienteRelacionado.DESCRIPCION;
        }
        expedienteInd.fechaInicio = exp.FECHA_INICIO.ToShortDateString();
        expedienteInd.fechaFin = exp.FECHA_FIN.ToShortDateString();
        expedienteInd.fechaExpediente =
exp.FECHA_EXPEDIENTE.ToShortDateString() + " " +
exp.FECHA_EXPEDIENTE.ToShortTimeString();
        expedienteInd.estadoExpediente = exp.ESTADO;
        List<RepresentanteExpedienteQuery> representantes = new
List<RepresentanteExpedienteQuery>();
        List<ArchivoExpedienteQuery> archivos = new
List<ArchivoExpedienteQuery>();
        List<EstadoExpedienteQuery> estados = new List<EstadoExpedienteQuery>();
List<EXPEDIENTE_DET_REPRE_HISTORICO> detRepresentanteHistorico =
db.EXPEDIENTE_DET_REPRE_HISTORICO.Where(p =>
p.CODIGO_EXPEDIENTE == codigo).ToList();
        foreach (var det in detRepresentanteHistorico)
        {
          RepresentanteExpedienteQuery representante = new
RepresentanteExpedienteQuery();
          representante.codigo_expediente = det.CODIGO_EXPEDIENTE;
          representante.correlativo = det.CORRELATIVO;
          representante.nombre_representante = det.NOMBRE_REPRESENTANTE;
          representante.numero_cip = det.NUMERO_CIP;
          representante.telefono_celular = det.TELEFONO_CELULAR;
          representante.correo_electronico = det.CORREO_ELECTRONICO;
          if (det.ESTADO == 1)
          {representante.estado_representante = "Activo";}
          else
          {representante.estado_representante = "Inactivo";}
         representantes.Add(representante);
        }
        return Json(expedienteInd);
    }
```

Revisión y Derivación de Solicitudes – Colaborador

```
using ProyectoTerceros.Models;
using ProyectoTerceros.Models.Registro.UsuarioExterno;
using ProyectoTerceros.Models.Registro.UsuarioInterno;
using ProyectoTerceros.Models.Seguridad;
using System;
using System.Collections.Generic;
using System.Configuration;
using System.IO;
using System.Linq;
using System.Net;
using System.Net.Http;
using System.Net.Http.Headers;
using System.Net.Mail;
using System.Text.RegularExpressions;
using System.Threading.Tasks;
using System.Web;
using System.Web.Mvc;
namespace ProyectoTerceros.Controllers.Seguridad
{
   public class UsuarioController : Controller
   {
      private ProyectoTercerosEntities db = new ProyectoTercerosEntities();
      public ActionResult SolicitudIndex()
      {
         if (Session["Usuario"] == null)
         { return RedirectToAction("Index", "Login");}
         else
         {return View();}
      }
      public ActionResult SolicitudRevisar(string codigo_solicitud)
      {
         if (Session["Usuario"] == null)
         {  return RedirectToAction("Index", "Login"); }
         else
         {ViewBag.Codigo = codigo_solicitud; return View("SolicitudRevisar");
         }       }
```

```
public JsonResult SolicitudList()
{
  List<UsuarioInternoSolicitudQuery> solicitudesQuery = new List<UsuarioInternoSolicitudQuery>();
  string codigoSession = Session["Codigo"].ToString();
  string esGestorcomercial = Session["esGestorComercial"].ToString();
  string esSupervisorTecnico = Session["esSupervisorTecnico"].ToString();
  List<SOLICITUD> solicitudes = db.SOLICITUD.Where(p => ((p.ESTADO == "3" && esGestorcomercial == "1") || ((p.ESTADO == "4" || p.ESTADO == "11" || p.ESTADO == "12" || p.ESTADO == "15") && p.TRAMITADO_POR == codigoSession && esGestorcomercial == "1")) || (p.ESTADO == "5" && p.TRAMITADO_POR == codigoSession && esSupervisorTecnico == "1") || p.ESTADO == "6" || p.ESTADO == "7").ToList();
  foreach (var solicitud in solicitudes)
  {
    UsuarioInternoSolicitudQuery solQuery = new UsuarioInternoSolicitudQuery();
    string tramitadoPor = "";
    USUARIO usu = db.USUARIO.Find(solicitud.TRAMITADO_POR_ORIGINAL);
    tramitadoPor = usu.NOMBRE + " " + usu.APELLIDO_PATERNO + " " + usu.APELLIDO_MATERNO;
    int kpi = 0;
    DateTime fechaActual = DateTime.Now;
    DateTime fechaEvaluacion = (DateTime)solicitud.FECHA_MAXIMA_ATENCION;
    solQuery.codigo_solicitud = solicitud.CODIGO_SOLICITUD;
    solQuery.asunto_solicitud = solicitud.ASUNTO;
    solQuery.codigo_expediente = solicitud.CODIGO_EXPEDIENTE;
    solQuery.descripcion_tipo_solicitud = solicitud.TIPO_SOLICITUD.DESCRIPCION;
    solQuery.descripcion_tipo_sistema_electrico = solicitud.EXPEDIENTE.TIPO_PROYECTO.TIPO_SISTEMA_ELECTRICO.DESCRIPCION;
    solQuery.tramitado_por = tramitadoPor;
```

```
            solQuery.fecha_solicitud =
solicitud.FECHA_SOLICITUD.ToShortDateString() + " " +
solicitud.FECHA_SOLICITUD.ToShortTimeString();
            solQuery.estado = solicitud.ESTADO;
            solQuery.kpi = kpi;
            List<SOLICITUD_HISTORIAL> historial =
db.SOLICITUD_HISTORIAL.Where(p => p.CODIGO_SOLICITUD ==
solicitud.CODIGO_SOLICITUD).OrderByDescending(p =>
p.CORRELATIVO).ToList();
            foreach (var hist in historial)
            {solQuery.estado_anterior = hist.ESTADO_ANTERIOR; break;  }
solicitudesQuery.Add(solQuery);
solicitud.KPI = kpi;
            db.SaveChanges();
        }
        return Json(solicitudesQuery);
    }
    public JsonResult SolicitudLeer(string codigo)
    {
        UsuarioInternoSolicitudInd solicitudInd = new UsuarioInternoSolicitudInd();
        SOLICITUD solicitud = db.SOLICITUD.Find(codigo);
        if (solicitud != null)
        {
            solicitudInd.codigo_solicitud = solicitud.CODIGO_SOLICITUD;
            solicitudInd.asunto_solicitud = solicitud.ASUNTO;
            solicitudInd.fecha_solicitud =
solicitud.FECHA_SOLICITUD.ToShortDateString() + " " +
solicitud.FECHA_SOLICITUD.ToShortTimeString();
            solicitudInd.plazo_atencion = solicitud.PLAZO_ATENCION;
            solicitudInd.codigo_expediente = solicitud.CODIGO_EXPEDIENTE;
            solicitudInd.codigo_solicitud_correspondiente =
solicitud.CODIGO_TIPO_SOLICITUD;
            solicitudInd.descripcion_expediente =
solicitud.EXPEDIENTE.DESCRIPCION;
            solicitudInd.descripcion_tipo_proyecto =
solicitud.EXPEDIENTE.TIPO_PROYECTO.DESCRIPCION;
```

```
            solicitudInd.descripcion_tipo_solicitud =
solicitud.TIPO_SOLICITUD.DESCRIPCION;
            solicitudInd.descripcion_tipo_sistema_electrico =
solicitud.EXPEDIENTE.TIPO_PROYECTO.TIPO_SISTEMA_ELECTRICO.DESCRI
PCION;
            solicitudInd.kpi = solicitud.KPI;
            EXPEDIENTE expediente =
db.EXPEDIENTE.Find(solicitud.CODIGO_EXPEDIENTE);
            if (expediente.EDITO_DATOS_TECNICOS == null)
            {solicitudInd.edito_datos_tecnicos = 0; }
            else
            {solicitudInd.edito_datos_tecnicos =
(decimal)expediente.EDITO_DATOS_TECNICOS;
            }
            string estado_anterior = "";
            List<SOLICITUD_HISTORIAL> solicitudHistorialUltimo =
db.SOLICITUD_HISTORIAL.Where(p => p.CODIGO_SOLICITUD ==
codigo).OrderByDescending(p => p.CORRELATIVO).ToList();
            foreach (var sol in solicitudHistorialUltimo)
            {estado_anterior = sol.ESTADO_ANTERIOR; break; }
            List<ArchivoSolicitudQuery> archivosAdjuntos = new
List<ArchivoSolicitudQuery>();
            List<UsuarioInternoArchivoRespuesta> archivosRespuesta = new
List<UsuarioInternoArchivoRespuesta>();
            List<UsuarioInternoArchivoRespuesta> archivosUsuarioRespuesta = new
List<UsuarioInternoArchivoRespuesta>();
            List<UsuarioInternoArchivoRespuesta> otrosArchivos = new
List<UsuarioInternoArchivoRespuesta>();
            List<SOLICITUD_DETALLE_ADJUNTO> archivosSolicitud =
db.SOLICITUD_DETALLE_ADJUNTO.Where(p => p.CODIGO_SOLICITUD ==
codigo).ToList();
            string rutaLectura =
ConfigurationManager.AppSettings["RutaLeerArchivos"].ToString();
            foreach (var archivoSolicitud in archivosSolicitud)
            {
               ArchivoSolicitudQuery archivo = new ArchivoSolicitudQuery();
```

```
                archivo.contenido = rutaLectura + "/" + archivoSolicitud.CONTENIDO;
                archivo.nombre_archivo = archivoSolicitud.NOMBRE_ARCHIVO;
                archivo.secuencia = archivoSolicitud.CORRELATIVO;
                archivo.codigo_solicitud = archivoSolicitud.CODIGO_SOLICITUD;
                archivo.codigo_requisito_tipo_solicitud =
archivoSolicitud.CODIG_REQUISITO_TIPO_SOLICITUD;
                archivo.estado = archivoSolicitud.ESTADO;
                archivosAdjuntos.Add(archivo);
            }        }
        else
        {  solicitudInd.rpta_estado = "Error";solicitudInd.rpta_mensaje = "Solicitud no
válida";}
return Json(solicitudInd);
    }
    public JsonResult SolicitudEnviar(UsuarioInternoSolicitudInput input)
    {
        Status status = new Status();
string codigo_entidad = input.codigo_inicial;
        string rpta_estado = "Ok";
        string rpta_mensaje = string.Empty;
SOLICITUD solicitud = db.SOLICITUD.Find(input.codigo_inicial);
        if (solicitud.ESTADO == "0")
        {rpta_estado = "Error";
            rpta_mensaje = "La Solicitud ha sido eliminada";
        }
        else
        {
            SOLICITUD_HISTORIAL solicitudHistorial = new
SOLICITUD_HISTORIAL();
            int correlativoHistorial = 0;
            correlativoHistorial = db.SOLICITUD_HISTORIAL.Count();
            if (correlativoHistorial == 0)
            {correlativoHistorial = 1; }
            else
            {correlativoHistorial = db.SOLICITUD_HISTORIAL.Max(p =>
p.CORRELATIVO) + 1; }
```

```
            //USUARIO QUIEN ESTÁ ATENDIENDO LA SOLICITUD
            string usuario_atiende = Session["Codigo"].ToString();
            solicitud.COMENTARIOS = input.comentario_respuesta_solicitud;
            solicitud.FECHA_ULTIMO_ESTADO = DateTime.Now;
            solicitud.FECHA_EDICION = DateTime.Now;
            solicitud.USUARIO_EDICION = Session["Codigo"].ToString();
            db.SaveChanges();
            }
              // OBTENGO LA CANTIDAD DE SOLICITUDES QUE PUEDE
MOSTRAR LUEGO
              int? numeroOpciones = 0;
              int? orden = 0;
              foreach (var list in listDetalle)
              {numeroOpciones = list.CANTIDAD_OPCION;
                orden = list.ORDEN;
                break; }
}
```

Generación de Reportes – Colaborador

```
using OfficeOpenXml;
using OfficeOpenXml.Style;
using ProyectoTerceros.Models;
using ProyectoTerceros.Models.Reporte;
using ProyectoTerceros.Models.Seguridad;
using System;
using System.Collections.Generic;
using System.Configuration;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Threading.Tasks;
using System.Web;
using System.Web.Mvc;
namespace ProyectoTerceros.Controllers.Reporte
{
  public class ReporteController : Controller
  {
    private ProyectoTercerosEntities db = new ProyectoTercerosEntities();
```

```
private string oradb =
ConfigurationManager.ConnectionStrings["CNX_ORC_PYO"].ConnectionString;
public ActionResult GestorComercial()
{
   if (Session["Usuario"] == null)
   {return RedirectToAction("Index", "Login");}
   else
   {return View();}
}
public ActionResult ExpedienteGenerada()
{
   if (Session["Usuario"] == null)
   {return RedirectToAction("Index", "Login");}
   else
   {return View();}
}
public ActionResult GerenciaTecnica()
{
   if (Session["Usuario"] == null)
   {return RedirectToAction("Index", "Login");}
   else
   {return View();}
}
public ActionResult AvanceExpediente()
{
   if (Session["Usuario"] == null)
   {return RedirectToAction("Index", "Login");}
   else
   {return View();}
}
public ActionResult SolicitudGenerada()
{
   if (Session["Usuario"] == null)
   {return RedirectToAction("Index", "Login");}
   else
   {return View();}
}
```

```
public ActionResult SeguimientoSolicitud()
{
   if (Session["Usuario"] == null)
   {return RedirectToAction("Index", "Login");}
   else
   {return View();}
}
public JsonResult GetReporteGestionComercial(ReporteGestionComercialQueryInput input)
{
   string fechaInicio = input.fechaInicio;
   string fechaFin = input.fechaFin;
   string expedienteEstado = input.expedienteEstado;
   string solicitudEstado = input.solicitudEstado;
   String query = @"select expe.Codigo_Expediente as codigo_expediente,
expe.descripcion as descripcion_proyecto,
expe.maxima_demanda as maxima_demanda,
to_char(expe.fecha_inicio,'DD/MM/YYYY') as fecha_inicio_expediente,
expe.estado as estado_expediente,
sol.codigo_solicitud as codigo_solicitud,
ts.descripcion as tipo_solicitud,
 sol.asunto as asunto_solicitud,
 to_char(sol.fecha_ultimo_estado,'DD/MM/YYYY HH24:MI:SS') as
fecha_ultimo_estado,
  sol.estado as estado_solicitud , '' as ultimo_estado ,
  to_char(sol.fecha_solicitud,'DD/MM/YYYY HH24:MI:SS')  as fecha_solicitud,
  to_char(sol.fecha_maxima_atencion,'DD/MM/YYYY') as fecha_maxima_atencion,
  ( trunc(sol.fecha_ultimo_estado - sol.fecha_solicitud) || ' d ' ||
       TRUNC(MOD((sol.fecha_ultimo_estado - sol.fecha_solicitud) * 24, 24)) || ' h ' ||
 TRUNC(MOD((sol.fecha_ultimo_estado - sol.fecha_solicitud) * (60 * 24), 60)) || ' min '
||
 TRUNC(MOD((sol.fecha_ultimo_estado - sol.fecha_solicitud) * (60 * 60 * 24), 60)) || '
s ' ) as TIEMPO_ATENCION ,
        CASE sol.estado
         when '7' then
```

```
        CASE                                                    WHEN
(TO_DATE(TO_CHAR(sol.fecha_ultimo_estado,'DD/MM/YYYY'))                 >
sol.fecha_maxima_atencion) THEN
            'SE RESPONDIÓ DESPUÉS DE: '   ||
            trunc(sol.fecha_ultimo_estado - sol.fecha_maxima_atencion) || ' d ' ||
            TRUNC(MOD((sol.fecha_ultimo_estado - sol.fecha_maxima_atencion) * 24,
24)) || ' h ' ||
      TRUNC(MOD((sol.fecha_ultimo_estado  -  sol.fecha_maxima_atencion)  *  (60  *
24), 60)) || ' min ' ||
      TRUNC(MOD((sol.fecha_ultimo_estado - sol.fecha_maxima_atencion) * (60 * 60
* 24), 60)) || ' s ' ||
                              'DE LA FECHA PREVISTA DE ATENCIÓN'
         ELSE
             'SE RESPONDIÓ ANTES DE: '   ||
        trunc(sol.fecha_maxima_atencion - sol.fecha_ultimo_estado) || ' d ' ||
      TRUNC(MOD((sol.fecha_maxima_atencion - sol.fecha_ultimo_estado) * 24, 24))
|| ' h ' ||
 TRUNC(MOD((sol.fecha_maxima_atencion  -  sol.fecha_ultimo_estado)  *  (60  *  24),
60)) || ' min ' ||
 TRUNC(MOD((sol.fecha_maxima_atencion  -  sol.fecha_ultimo_estado)  *  (60  *  60  *
24), 60)) || ' s '     || 'DE LA FECHA PREVISTA DE ATENCIÓN'
         END

       WHEN '6' THEN
         CASE
           WHEN   (TO_DATE(TO_CHAR(sol.fecha_ultimo_estado,'DD/MM/YYYY'))
> sol.fecha_maxima_atencion) THEN
         'SE RESPONDIÓ DESPUÉS DE: '   ||
            trunc(sol.fecha_ultimo_estado - sol.fecha_maxima_atencion) || ' d ' ||
            TRUNC(MOD((sol.fecha_ultimo_estado - sol.fecha_maxima_atencion) * 24,
24)) || ' h ' ||
      TRUNC(MOD((sol.fecha_ultimo_estado  -  sol.fecha_maxima_atencion)  *  (60  *
24), 60)) || ' min ' ||
      TRUNC(MOD((sol.fecha_ultimo_estado - sol.fecha_maxima_atencion) * (60 * 60
* 24), 60)) || ' s ' ||
                              'DE LA FECHA PREVISTA DE ATENCIÓN'
         ELSE
```

```
            'SE RESPONDIÓ ANTES DE: '   ||
          trunc(sol.fecha_maxima_atencion - sol.fecha_ultimo_estado) || ' d ' ||
       TRUNC(MOD((sol.fecha_maxima_atencion - sol.fecha_ultimo_estado) * 24, 24))
|| ' h ' ||
 TRUNC(MOD((sol.fecha_maxima_atencion  -  sol.fecha_ultimo_estado)  *  (60  *  24),
60)) || ' min ' ||
 TRUNC(MOD((sol.fecha_maxima_atencion  -  sol.fecha_ultimo_estado)  *  (60  *  60  *
24), 60)) || ' s '      || 'DE LA FECHA PREVISTA DE ATENCIÓN'
           END
         ELSE
           CASE
             WHEN  (SYSDATE > sol.fecha_maxima_atencion) THEN
           'EL TIEMPO EXCEDENTE ES DE: '   ||
              trunc(SYSDATE - sol.fecha_maxima_atencion) || ' d ' ||
              TRUNC(MOD((SYSDATE - sol.fecha_maxima_atencion) * 24, 24)) || ' h ' ||
       TRUNC(MOD((SYSDATE - sol.fecha_maxima_atencion) * (60 * 24), 60)) || ' min
' ||
       TRUNC(MOD((SYSDATE - sol.fecha_maxima_atencion) * (60 * 60 * 24), 60)) ||
' s ' ||
                                  'DE LA FECHA PREVISTA DE ATENCIÓN'
           ELSE
             'EL TIEMPO FALTANTE ES DE: '   ||
          trunc(sol.fecha_maxima_atencion - sol.fecha_ultimo_estado) || ' d ' ||
       TRUNC(MOD((sol.fecha_maxima_atencion - sol.fecha_ultimo_estado) * 24, 24))
|| ' h ' ||
 TRUNC(MOD((sol.fecha_maxima_atencion  -  sol.fecha_ultimo_estado)  *  (60  *  24),
60)) || ' min ' ||
 TRUNC(MOD((sol.fecha_maxima_atencion  -  sol.fecha_ultimo_estado)  *  (60  *  60  *
24), 60)) || ' s '      || 'PARA CUMPLIR LA FECHA MAXIMA DE ATENCION'
           END
                END AS TIEMPO_DURACION,
         CASE sol.estado when '7' then 'SOLICITUD RECHAZADA'
    WHEN '6' THEN
       CASE
        WHEN    (TO_DATE(TO_CHAR(sol.fecha_ultimo_estado,'DD/MM/YYYY')) >
sol.fecha_maxima_atencion) THEN
       'ATENDIDA FUERA DE PLAZO'
```

```
        ELSE
          'ATENDIDA DENTRO DE PLAZO'
        END
      ELSE
       CASE
         WHEN  (SYSDATE > sol.fecha_maxima_atencion) THEN
        'SOLICITUD SERÁ ATENDIDA FUERA DE PLAZO'
        ELSE
         'SOLICITUD SERÁ ATENDIDA DENTRO DE PLAZO'
        END
        END AS COMENTARIO_SOLICITUD,
        sol.plazo_atencion || ' días' AS PLAZO_ATENCION
  from GTIPROYECTOS.SOLICITUD sol
inner join GTIPROYECTOS.EXPEDIENTE expe
on sol.Codigo_Expediente = expe.Codigo_Expediente
inner join GTIPROYECTOS.TIPO_SOLICITUD ts
on sol.codigo_tipo_solicitud = ts.codigo_tipo_solicitud
where sol.estado in (" + solicitudEstado + @") and expe.estado in (" + expedienteEstado
+ @") and
(to_date(to_char(expe.fecha_inicio,'DD/MM/YYYY'), 'DD/MM/YYYY') >=
TO_DATE('" + fechaInicio + @"','DD/MM/YYYY') and
to_date(to_char(expe.fecha_inicio,'DD/MM/YYYY'), 'DD/MM/YYYY') <=
TO_DATE('" + fechaFin + @"', 'DD/MM/YYYY'))";
          IList<ReporteGestionComercial> reporte =
db.Database.SqlQuery<ReporteGestionComercial>(query).ToList();
          foreach (var report in reporte)
          {
            List<SOLICITUD_HISTORIAL> historial =
db.SOLICITUD_HISTORIAL.Where(p => p.CODIGO_SOLICITUD ==
report.codigo_solicitud).OrderByDescending(p => p.CORRELATIVO).ToList();
              foreach (var hist in historial)
              { report.ultimo_estado = hist.ESTADO_ANTERIOR;
                break;
              }          }
          return Json(reporte);                                                    }
[Route("/Reporte/GestionComercialDescargarExcel/{fechaInicio}/{fechaFin}/{expedie
nteEstado}/{solicitudEstado}")]
```

```
public JsonResult GetReporteExpedienteGenerada(ReporteExpedienteregistradoQueryInput input)
{
    string departamento = input.departamento;
    string provincia = input.provincia;
    string anio = input.anio;
    String query = @"select distinct
(
SELECT ener.entity_name from ENERGIA.GCCOM_GEOGRAPHIC_ENTITY ener
WHERE ubigeo.DEPARTAMENTO = ener.Id_Geo_Entity
) as nombre_departamento,
(
SELECT ener.entity_name from ENERGIA.GCCOM_GEOGRAPHIC_ENTITY ener
WHERE ubigeo.PROVINCIA = ener.Id_Geo_Entity
) as nombre_provincia,
(
SELECT COUNT(*) FROM GTIPROYECTOS.EXPEDIENTE ex
where CONCAT(SUBSTR(ex.UBICACION_GEOGRAFICA, 1 ,4), '000000000000000') = ubigeo.DEPARTAMENTO
and CONCAT(SUBSTR(ex.UBICACION_GEOGRAFICA, 1 ,9), '0000000000') = ubigeo.PROVINCIA
and to_char(ex.fecha_inicio, 'MM') = '01' or to_char(ex.fecha_inicio, 'MM') = '1'
) as mes_enero ,
(
SELECT COUNT(*) FROM GTIPROYECTOS.EXPEDIENTE ex
where CONCAT(SUBSTR(ex.UBICACION_GEOGRAFICA, 1 ,4), '000000000000000') = ubigeo.DEPARTAMENTO
and CONCAT(SUBSTR(ex.UBICACION_GEOGRAFICA, 1 ,9), '0000000000') = ubigeo.PROVINCIA
and to_char(ex.fecha_inicio, 'MM') = '02'  or to_char(ex.fecha_inicio, 'MM') = '2'
) as mes_febrero
,
(
SELECT COUNT(*) FROM GTIPROYECTOS.EXPEDIENTE ex
where CONCAT(SUBSTR(ex.UBICACION_GEOGRAFICA, 1 ,4), '000000000000000') = ubigeo.DEPARTAMENTO
```

```
and CONCAT(SUBSTR(ex.UBICACION_GEOGRAFICA, 1 ,9), '0000000000') =
ubigeo.PROVINCIA
and to_char(ex.fecha_inicio, 'MM') = '03'  or to_char(ex.fecha_inicio, 'MM') = '3'
) as mes_marzo
,
(
SELECT COUNT(*) FROM GTIPROYECTOS.EXPEDIENTE ex
where CONCAT(SUBSTR(ex.UBICACION_GEOGRAFICA, 1 ,4),
'000000000000000') = ubigeo.DEPARTAMENTO
and CONCAT(SUBSTR(ex.UBICACION_GEOGRAFICA, 1 ,9), '0000000000') =
ubigeo.PROVINCIA
and to_char(ex.fecha_inicio, 'MM') = '04'  or to_char(ex.fecha_inicio, 'MM') = '4'
) as mes_abril
,
(
SELECT COUNT(*) FROM GTIPROYECTOS.EXPEDIENTE ex
where CONCAT(SUBSTR(ex.UBICACION_GEOGRAFICA, 1 ,4),
'000000000000000') = ubigeo.DEPARTAMENTO
and CONCAT(SUBSTR(ex.UBICACION_GEOGRAFICA, 1 ,9), '0000000000') =
ubigeo.PROVINCIA
and to_char(ex.fecha_inicio, 'MM') = '05'  or to_char(ex.fecha_inicio, 'MM') = '5'
) as mes_mayo
,
(
SELECT COUNT(*) FROM GTIPROYECTOS.EXPEDIENTE ex
where CONCAT(SUBSTR(ex.UBICACION_GEOGRAFICA, 1 ,4),
'000000000000000') = ubigeo.DEPARTAMENTO
and CONCAT(SUBSTR(ex.UBICACION_GEOGRAFICA, 1 ,9), '0000000000') =
ubigeo.PROVINCIA
and to_char(ex.fecha_inicio, 'MM') = '06'  or to_char(ex.fecha_inicio, 'MM') = '6'
) as mes_junio
,
(
SELECT COUNT(*) FROM GTIPROYECTOS.EXPEDIENTE ex
where CONCAT(SUBSTR(ex.UBICACION_GEOGRAFICA, 1 ,4),
'000000000000000') = ubigeo.DEPARTAMENTO
```

```
and CONCAT(SUBSTR(ex.UBICACION_GEOGRAFICA, 1 ,9), '0000000000') =
ubigeo.PROVINCIA
and to_char(ex.fecha_inicio, 'MM') = '07'  or to_char(ex.fecha_inicio, 'MM') = '7'
) as mes_julio
,
(
SELECT COUNT(*) FROM GTIPROYECTOS.EXPEDIENTE ex
where CONCAT(SUBSTR(ex.UBICACION_GEOGRAFICA, 1 ,4),
'000000000000000') = ubigeo.DEPARTAMENTO
and CONCAT(SUBSTR(ex.UBICACION_GEOGRAFICA, 1 ,9), '0000000000') =
ubigeo.PROVINCIA
and to_char(ex.fecha_inicio, 'MM') = '08'  or to_char(ex.fecha_inicio, 'MM') = '8'
) as mes_agosto
,
(
SELECT COUNT(*) FROM GTIPROYECTOS.EXPEDIENTE ex
where CONCAT(SUBSTR(ex.UBICACION_GEOGRAFICA, 1 ,4),
'000000000000000') = ubigeo.DEPARTAMENTO
and CONCAT(SUBSTR(ex.UBICACION_GEOGRAFICA, 1 ,9), '0000000000') =
ubigeo.PROVINCIA
and to_char(ex.fecha_inicio, 'MM') = '09'  or to_char(ex.fecha_inicio, 'MM') = '9'
) as mes_setiembre
,
(
SELECT COUNT(*) FROM GTIPROYECTOS.EXPEDIENTE ex
where CONCAT(SUBSTR(ex.UBICACION_GEOGRAFICA, 1 ,4),
'000000000000000') = ubigeo.DEPARTAMENTO
and CONCAT(SUBSTR(ex.UBICACION_GEOGRAFICA, 1 ,9), '0000000000') =
ubigeo.PROVINCIA
and to_char(ex.fecha_inicio, 'MM') = '10'
) as mes_octubre
,
(
SELECT COUNT(*) FROM GTIPROYECTOS.EXPEDIENTE ex
where CONCAT(SUBSTR(ex.UBICACION_GEOGRAFICA, 1 ,4),
'000000000000000') = ubigeo.DEPARTAMENTO
```

```
and CONCAT(SUBSTR(ex.UBICACION_GEOGRAFICA, 1 ,9), '0000000000') =
ubigeo.PROVINCIA
and to_char(ex.fecha_inicio, 'MM') = '11'
) as mes_noviembre
,
(
SELECT COUNT(*) FROM GTIPROYECTOS.EXPEDIENTE ex
where CONCAT(SUBSTR(ex.UBICACION_GEOGRAFICA, 1 ,4),
'000000000000000') = ubigeo.DEPARTAMENTO
and CONCAT(SUBSTR(ex.UBICACION_GEOGRAFICA, 1 ,9), '0000000000') =
ubigeo.PROVINCIA
and to_char(ex.fecha_inicio, 'MM') = '12'
) as mes_diciembre
 from (
 select expe.UBICACION_GEOGRAFICA as UBIGEO,
CONCAT(SUBSTR(expe.UBICACION_GEOGRAFICA, 1 ,4), '000000000000000')
AS DEPARTAMENTO,
CONCAT(SUBSTR(expe.UBICACION_GEOGRAFICA, 1 ,9), '0000000000') AS
PROVINCIA
  from GTIPROYECTOS.EXPEDIENTE expe
  where to_char(expe.fecha_inicio, 'YYYY') = '" + anio + @"'
   and ( '0' = '" + departamento + @"' or
(CONCAT(SUBSTR(expe.UBICACION_GEOGRAFICA, 1 ,4), '000000000000000') =
'" + departamento + @"'))
   and ('0' = '" + provincia + @"' or
CONCAT(SUBSTR(expe.UBICACION_GEOGRAFICA, 1 ,9), '0000000000') = '" +
provincia + @"')
  ) ubigeo
";
            IList<ReporteExpedienteRegistrado> reporte =
db.Database.SqlQuery<ReporteExpedienteRegistrado>(query).ToList();
            return Json(reporte);
        }
        public JsonResult
GetReporteSolicitudGenerada(ReporteSolicitudGeneradaQueryInput input)
        {
            string departamento = input.departamento;
```

```
        string provincia = input.provincia;
        string anio = input.anio;
        string codigo_tipo_sistema_electrico = input.codigo_tipo_sistema_electrico;
        string codigo_tipo_solicitud = input.codigo_tipo_solicitud;
        String query = @"select distinct
(
SELECT ener.entity_name from ENERGIA.GCCOM_GEOGRAPHIC_ENTITY ener
WHERE general.DEPARTAMENTO = ener.Id_Geo_Entity
) as nombre_departamento,
(
SELECT ener.entity_name from ENERGIA.GCCOM_GEOGRAPHIC_ENTITY ener
WHERE general.PROVINCIA = ener.Id_Geo_Entity
) as nombre_provincia,

(
SELECT distinct tipoSolicitud.Descripcion from GTIPROYECTOS.TIPO_SOLICITUD
tipoSolicitud
WHERE tipoSolicitud.Codigo_Tipo_Solicitud = general.codigo_tipo_solicitud
) as tipo_solicitud,
(
SELECT distinct tipoSistElec.Descripcion from
GTIPROYECTOS.TIPO_SISTEMA_ELECTRICO tipoSistElec
WHERE tipoSistElec.Codigo_Tipo_Sistema_Electrico =
general.CODIGO_TIPO_SISTEMA_ELECTRICO
) as tipo_sistema_electrico,
(
SELECT COUNT(*) FROM GTIPROYECTOS.EXPEDIENTE ex
  inner join GTIPROYECTOS.SOLICITUD soli
  on ex.codigo_expediente = soli.codigo_expediente
where CONCAT(SUBSTR(ex.UBICACION_GEOGRAFICA, 1 ,4),
'000000000000000') = general.DEPARTAMENTO
and CONCAT(SUBSTR(ex.UBICACION_GEOGRAFICA, 1 ,9), '0000000000') =
general.PROVINCIA
and to_char(soli.fecha_solicitud, 'YYYY') = general.ANIO

and soli.codigo_tipo_solicitud = general.CODIGO_TIPO_SOLICITUD
```

```
and soli.codigo_tipo_sistema_electrico =
general.CODIGO_TIPO_SISTEMA_ELECTRICO
and to_char(ex.fecha_inicio, 'MM') = '01' or to_char(ex.fecha_inicio, 'MM') = '1'
) as mes_enero ,
(
SELECT COUNT(*) FROM GTIPROYECTOS.EXPEDIENTE ex
 inner join GTIPROYECTOS.SOLICITUD soli
 on ex.codigo_expediente = soli.codigo_expediente
where CONCAT(SUBSTR(ex.UBICACION_GEOGRAFICA, 1 ,4),
'000000000000000') = general.DEPARTAMENTO
and CONCAT(SUBSTR(ex.UBICACION_GEOGRAFICA, 1 ,9), '0000000000') =
general.PROVINCIA
and to_char(soli.fecha_solicitud, 'YYYY') = general.ANIO

and soli.codigo_tipo_solicitud = general.CODIGO_TIPO_SOLICITUD
and soli.codigo_tipo_sistema_electrico =
general.CODIGO_TIPO_SISTEMA_ELECTRICO
and to_char(ex.fecha_inicio, 'MM') = '02'  or to_char(ex.fecha_inicio, 'MM') = '2'
) as mes_febrero
,
(
SELECT COUNT(*) FROM GTIPROYECTOS.EXPEDIENTE ex
 inner join GTIPROYECTOS.SOLICITUD soli
 on ex.codigo_expediente = soli.codigo_expediente
where CONCAT(SUBSTR(ex.UBICACION_GEOGRAFICA, 1 ,4),
'000000000000000') = general.DEPARTAMENTO
and CONCAT(SUBSTR(ex.UBICACION_GEOGRAFICA, 1 ,9), '0000000000') =
general.PROVINCIA
and to_char(soli.fecha_solicitud, 'YYYY') = general.ANIO

and soli.codigo_tipo_solicitud = general.CODIGO_TIPO_SOLICITUD
and soli.codigo_tipo_sistema_electrico =
general.CODIGO_TIPO_SISTEMA_ELECTRICO
and to_char(ex.fecha_inicio, 'MM') = '03'  or to_char(ex.fecha_inicio, 'MM') = '3'
) as mes_marzo
,
(
```

```
SELECT COUNT(*) FROM GTIPROYECTOS.EXPEDIENTE ex
 inner join GTIPROYECTOS.SOLICITUD soli
 on ex.codigo_expediente = soli.codigo_expediente
where CONCAT(SUBSTR(ex.UBICACION_GEOGRAFICA, 1 ,4),
'000000000000000') = general.DEPARTAMENTO
and CONCAT(SUBSTR(ex.UBICACION_GEOGRAFICA, 1 ,9), '0000000000') =
general.PROVINCIA
and to_char(soli.fecha_solicitud, 'YYYY') = general.ANIO
and soli.codigo_tipo_solicitud = general.CODIGO_TIPO_SOLICITUD
and soli.codigo_tipo_sistema_electrico =
general.CODIGO_TIPO_SISTEMA_ELECTRICO
and to_char(ex.fecha_inicio, 'MM') = '04'  or to_char(ex.fecha_inicio, 'MM') = '4'
) as mes_abril
,
(
SELECT COUNT(*) FROM GTIPROYECTOS.EXPEDIENTE ex
 inner join GTIPROYECTOS.SOLICITUD soli
 on ex.codigo_expediente = soli.codigo_expediente
where CONCAT(SUBSTR(ex.UBICACION_GEOGRAFICA, 1 ,4),
'000000000000000') = general.DEPARTAMENTO
and CONCAT(SUBSTR(ex.UBICACION_GEOGRAFICA, 1 ,9), '0000000000') =
general.PROVINCIA
and to_char(soli.fecha_solicitud, 'YYYY') = general.ANIO
and soli.codigo_tipo_solicitud = general.CODIGO_TIPO_SOLICITUD
and soli.codigo_tipo_sistema_electrico =
general.CODIGO_TIPO_SISTEMA_ELECTRICO
and to_char(ex.fecha_inicio, 'MM') = '05'  or to_char(ex.fecha_inicio, 'MM') = '5'
) as mes_mayo
,
(
SELECT COUNT(*) FROM GTIPROYECTOS.EXPEDIENTE ex
 inner join GTIPROYECTOS.SOLICITUD soli
 on ex.codigo_expediente = soli.codigo_expediente
where CONCAT(SUBSTR(ex.UBICACION_GEOGRAFICA, 1 ,4),
'000000000000000') = general.DEPARTAMENTO
and CONCAT(SUBSTR(ex.UBICACION_GEOGRAFICA, 1 ,9), '0000000000') =
general.PROVINCIA
```

```
and to_char(soli.fecha_solicitud, 'YYYY') = general.ANIO

and soli.codigo_tipo_solicitud = general.CODIGO_TIPO_SOLICITUD
and soli.codigo_tipo_sistema_electrico =
general.CODIGO_TIPO_SISTEMA_ELECTRICO
and to_char(ex.fecha_inicio, 'MM') = '06'  or to_char(ex.fecha_inicio, 'MM') = '6'
) as mes_junio
,
(
SELECT COUNT(*) FROM GTIPROYECTOS.EXPEDIENTE ex
  inner join GTIPROYECTOS.SOLICITUD soli
  on ex.codigo_expediente = soli.codigo_expediente
where CONCAT(SUBSTR(ex.UBICACION_GEOGRAFICA, 1 ,4),
'000000000000000') = general.DEPARTAMENTO
and CONCAT(SUBSTR(ex.UBICACION_GEOGRAFICA, 1 ,9), '0000000000') =
general.PROVINCIA
and to_char(soli.fecha_solicitud, 'YYYY') = general.ANIO

and soli.codigo_tipo_solicitud = general.CODIGO_TIPO_SOLICITUD
and soli.codigo_tipo_sistema_electrico =
general.CODIGO_TIPO_SISTEMA_ELECTRICO
and to_char(ex.fecha_inicio, 'MM') = '07'  or to_char(ex.fecha_inicio, 'MM') = '7'
) as mes_julio
,
(
SELECT COUNT(*) FROM GTIPROYECTOS.EXPEDIENTE ex
  inner join GTIPROYECTOS.SOLICITUD soli
  on ex.codigo_expediente = soli.codigo_expediente
where CONCAT(SUBSTR(ex.UBICACION_GEOGRAFICA, 1 ,4),
'000000000000000') = general.DEPARTAMENTO
and CONCAT(SUBSTR(ex.UBICACION_GEOGRAFICA, 1 ,9), '0000000000') =
general.PROVINCIA
and to_char(soli.fecha_solicitud, 'YYYY') = general.ANIO

and soli.codigo_tipo_solicitud = general.CODIGO_TIPO_SOLICITUD
and soli.codigo_tipo_sistema_electrico =
general.CODIGO_TIPO_SISTEMA_ELECTRICO
```

```
and to_char(ex.fecha_inicio, 'MM') = '08'  or to_char(ex.fecha_inicio, 'MM') = '8'
) as mes_agosto
,
(
SELECT COUNT(*) FROM GTIPROYECTOS.EXPEDIENTE ex
  inner join GTIPROYECTOS.SOLICITUD soli
  on ex.codigo_expediente = soli.codigo_expediente
where CONCAT(SUBSTR(ex.UBICACION_GEOGRAFICA, 1 ,4),
'000000000000000') = general.DEPARTAMENTO
and CONCAT(SUBSTR(ex.UBICACION_GEOGRAFICA, 1 ,9), '0000000000') =
general.PROVINCIA
and to_char(soli.fecha_solicitud, 'YYYY') = general.ANIO

and soli.codigo_tipo_solicitud = general.CODIGO_TIPO_SOLICITUD
and soli.codigo_tipo_sistema_electrico =
general.CODIGO_TIPO_SISTEMA_ELECTRICO
and to_char(ex.fecha_inicio, 'MM') = '09'  or to_char(ex.fecha_inicio, 'MM') = '9'
) as mes_setiembre
,
(
SELECT COUNT(*) FROM GTIPROYECTOS.EXPEDIENTE ex
  inner join GTIPROYECTOS.SOLICITUD soli
  on ex.codigo_expediente = soli.codigo_expediente
where CONCAT(SUBSTR(ex.UBICACION_GEOGRAFICA, 1 ,4),
'000000000000000') = general.DEPARTAMENTO
and CONCAT(SUBSTR(ex.UBICACION_GEOGRAFICA, 1 ,9), '0000000000') =
general.PROVINCIA
and to_char(soli.fecha_solicitud, 'YYYY') = general.ANIO

and soli.codigo_tipo_solicitud = general.CODIGO_TIPO_SOLICITUD
and soli.codigo_tipo_sistema_electrico =
general.CODIGO_TIPO_SISTEMA_ELECTRICO
and to_char(ex.fecha_inicio, 'MM') = '10'
) as mes_octubre
,
(
SELECT COUNT(*) FROM GTIPROYECTOS.EXPEDIENTE ex
```

```
 inner join GTIPROYECTOS.SOLICITUD soli
 on ex.codigo_expediente = soli.codigo_expediente
where CONCAT(SUBSTR(ex.UBICACION_GEOGRAFICA, 1 ,4),
'000000000000000') = general.DEPARTAMENTO
and CONCAT(SUBSTR(ex.UBICACION_GEOGRAFICA, 1 ,9), '0000000000') =
general.PROVINCIA
and to_char(soli.fecha_solicitud, 'YYYY') = general.ANIO

and soli.codigo_tipo_solicitud = general.CODIGO_TIPO_SOLICITUD
and soli.codigo_tipo_sistema_electrico =
general.CODIGO_TIPO_SISTEMA_ELECTRICO
and to_char(ex.fecha_inicio, 'MM') = '11'
) as mes_noviembre
,
(
SELECT COUNT(*) FROM GTIPROYECTOS.EXPEDIENTE ex
 inner join GTIPROYECTOS.SOLICITUD soli
 on ex.codigo_expediente = soli.codigo_expediente
where CONCAT(SUBSTR(ex.UBICACION_GEOGRAFICA, 1 ,4),
'000000000000000') = general.DEPARTAMENTO
and CONCAT(SUBSTR(ex.UBICACION_GEOGRAFICA, 1 ,9), '0000000000') =
general.PROVINCIA
and to_char(soli.fecha_solicitud, 'YYYY') = general.ANIO

and soli.codigo_tipo_solicitud = general.CODIGO_TIPO_SOLICITUD
and soli.codigo_tipo_sistema_electrico =
general.CODIGO_TIPO_SISTEMA_ELECTRICO
and to_char(ex.fecha_inicio, 'MM') = '12'
) as mes_diciembre
 from (
 select distinct expe.UBICACION_GEOGRAFICA as UBIGEO,
CONCAT(SUBSTR(expe.UBICACION_GEOGRAFICA, 1 ,4), '000000000000000')
AS DEPARTAMENTO,
CONCAT(SUBSTR(expe.UBICACION_GEOGRAFICA, 1 ,9), '0000000000') AS
PROVINCIA,
to_char(sol.fecha_solicitud, 'YYYY') AS ANIO,
sol.codigo_tipo_solicitud AS CODIGO_TIPO_SOLICITUD,
```

```
sol.codigo_tipo_sistema_electrico AS CODIGO_TIPO_SISTEMA_ELECTRICO
  from GTIPROYECTOS.EXPEDIENTE expe
  inner join GTIPROYECTOS.SOLICITUD sol
  on expe.codigo_expediente = sol.codigo_expediente
  where to_char(sol.fecha_solicitud, 'YYYY') = '" + anio + @"'
  and ('0' = '" + codigo_tipo_solicitud + @"' or sol.codigo_tipo_solicitud = '" +
codigo_tipo_solicitud + @"')
   and ('0' = '" + codigo_tipo_sistema_electrico + @"' or
sol.codigo_tipo_sistema_electrico = '" + codigo_tipo_sistema_electrico + @"')
   and ( '0' = '" + departamento + @"' or
(CONCAT(SUBSTR(expe.UBICACION_GEOGRAFICA, 1 ,4), '000000000000000') =
'" + departamento + @"'))
    and ('0' = '" + provincia + @"' or
CONCAT(SUBSTR(expe.UBICACION_GEOGRAFICA, 1 ,9), '0000000000') = '" +
provincia + @"')
  ) general
";
        IList<ReporteSolicitudRegistrado> reporte =
db.Database.SqlQuery<ReporteSolicitudRegistrado>(query).ToList();
        return Json(reporte);
      }
```

Capítulo IV: APORTES A LA INSTITUCION

Luego de la publicación del sistema de Proyectos y Obra de Terceros, se llegó a lograr varios objetivos propuestos y que muy importantes tanto para la Gerencia de TI, Gerencia Comercial y Gerencia Técnica, se mencionan alguno de estos objetivos:

1. Se pudo centralizar la información que envían los clientes cuando se apertura un expediente para un proyecto u obra, reduciendo el tiempo de búsqueda y de consulta de información a la Gerencia Comercial y a la Gerencia Técnica.
2. Se reduzco la generación innecesaria del Número de Identificador de Suministro (NIS) en un 100%
3. Se mejoró la comunicación entre el cliente y Electro Dunas, así como la comunicación interna de la misma empresa.
4. Se reduzco el uso del papel al 100% contribuyendo al cuidado del medio ambiente, colaborando al objetivo de ser una empresa sostenible.
5. Se logró aportar al objetivo estratégico de transformación digital que tiene la Gerencia TI.

CONCLUSIONES

Cumplido el objetivo estratégico de la Gerencia de TI, al desarrollar e implementar el Portal de Proyecto y Obras de Terceros, se llega a las siguientes conclusiones que aportan al ejercicio profesional de mi carrera.

1. El cumplimiento de los procedimientos de la Gerencia de TI ha sido un punto clave en este desarrollo e implementación, puesto que permite tener una mejor visibilidad del objetivo principal del proyecto.
2. El buen uso de la estructura MVC (Modelo – Vista – Controlador) permitió que se pueda tener una mejor organización el desarrollo del proyecto, asimismo esta estructura facilitó la implementación del sistema en el entorno web.
3. El buen uso de la metodología ágil SCRUM, permitió mejorar la gestión de tiempo, la gestión de recursos, así como mejorar la comunicación y coordinación entre los diferentes roles.
4. La automatización del proceso de presentación de expedientes y solicitudes para un proyecto u obra y el buen funcionamiento del sistema respalda la importancia de la transformación digital en una empresa para poder tener un flujo más ordenado y con información inmediata para la toma de decisiones necesarias.

RECOMENDACIONES

Con la experiencia obtenida al desarrollar e implementar el Portal de Proyectos y Obras de Terceros, presento las siguientes recomendaciones para que pueda ayudar otros egresados.

1. Antes de iniciar un proyecto se debe de revisar si la Gerencia de TI tiene procedimientos para el desarrollo e implementación de un sistema, ya que estos ayudarán en el tiempo de desarrollo del mismo.
2. El buen empleo de una arquitectura de desarrollo de un sistema asegura el orden para el buen desarrollo del sistema, así como una mejor organización para su implementación.
3. El empleo de una metodología ágil para que se pueda organizar las tareas a desarrollar y los roles que tendrán cada uno de los usuarios es de suma importancia, ya que se asegura que se tenga una participación constante y activa en el tiempo de desarrollo de un sistema.
4. El tiempo invertido en un análisis del proceso a automatizar acorta el tiempo de desarrollo de un sistema.
5. La correcta construcción y relación de objetos de base de datos que se tenga para el desarrollo de un sistema, acorta el impacto ante un nuevo pedido o cambio del usuario, así como salvaguardar la integración de la información.

REFERENCIAS BIBLIOGRÁFICAS

[1] Scrum Study, UNA GUÍA PARA EL CUERPO DE CONOCIMIENTO DE SCRUM (GUIA SBOK), 410 N 44th Street, Suite 240: SCRUMstudy™, una marca de VMEdu, Inc., 2013.

[2] S. V. G. Leonardo y J. A. Zamora Granados, «“Sistema informático web de control de recorrido de la empresa “El,» Lambayeque, 2020.

[3] M. S. L. David, «Aplicación web para la gestión de información del departamento tecnico de reparacion y garantias de la empresa TELNET SOPORTE de la ciudad de Ibarra,» Ecuador, 2018.

[4] B. V. S. Estephany y A. A. Sanchez Aranda, «Desarrollo de un sistema web y su influencia en el proceso de gestión de historias clinias del centro odontologico Ilumiden, 2018,» Perú, 2018.

[5] M. Gendra, «Mariano Gendra,» [En línea]. Available: https://marianogendra.com.ar/Articulos/aplicaciones-web-vs-escritorio.

[6] J. M. Carreras, Optimización SQL en Oracle, CreateSpace Independent Publishing Platform, 2013.

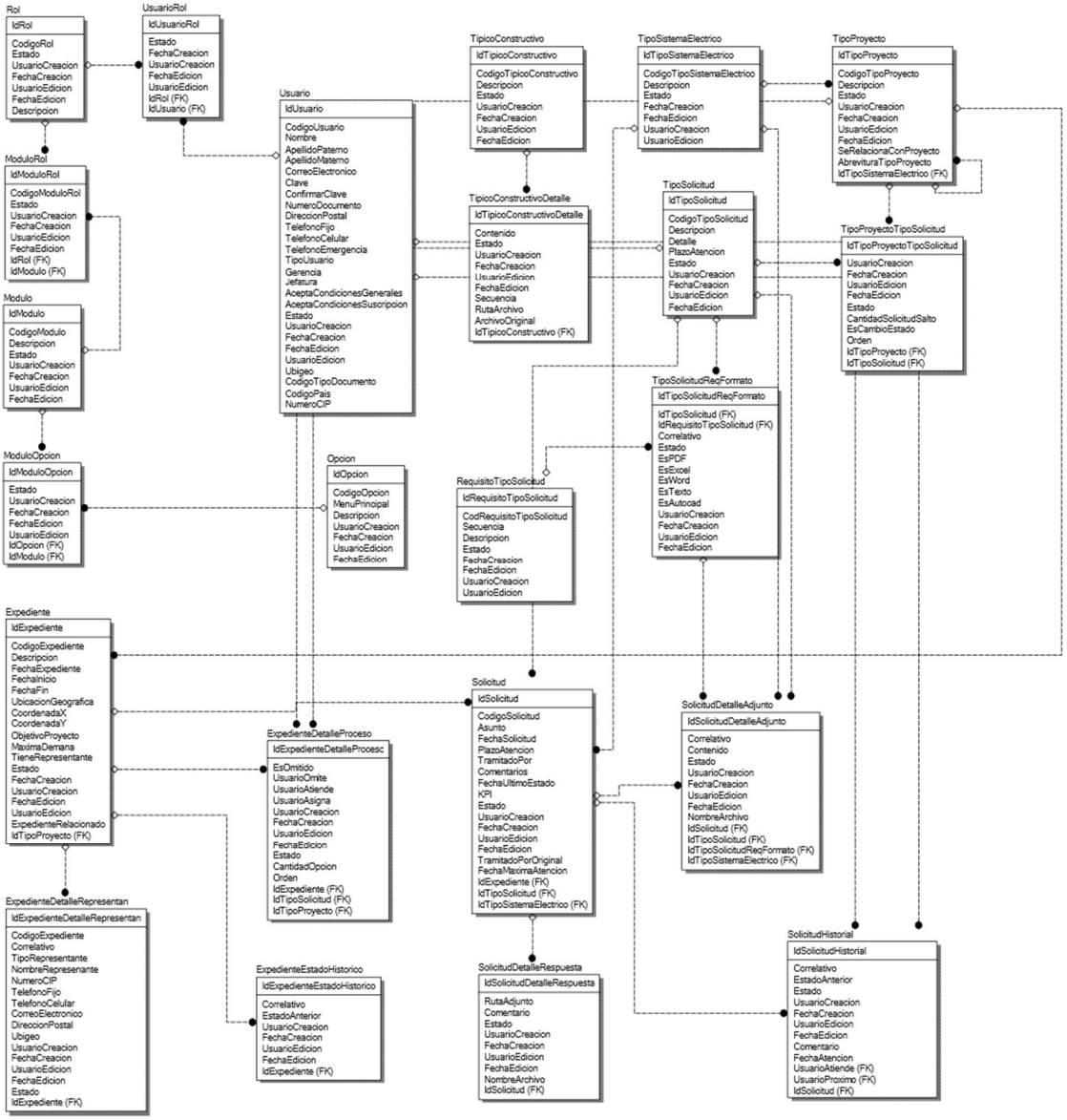
[7] S. Feuerstein, Oracle PL/SQL Best Practices, O'Reilly.

[8] CEPAL, «Agenda Digital para America Latina y el Caribe,» 2021.

ANEXOS

ANEXO I

Modelamiento de base de datos



ANEXO II

Solicitud de Nuevo Sistema de Información

ElectroDunas	REQUERIMIENTO DE NUEVO SISTEMA DE INFORMACIÓN	Versión 1
		GT-R-2-6-0-1

SOLICITUD DE NUEVO SISTEMA DE INFORMACIÓN

NOMBRE DEL PROYECTO: PORTAL DE PROYECTOS Y OBRAS DE TERCEROS	
GERENCIA SOLICITANTE: GERENCIA COMERCIAL	**JEFATURA SOLICITANTE:** GESTIÓN DE PROYECTOS Y OBRAS
LUGAR DE EJECUCIÓN DEL PROYECTO: REMOTO	
SEDE: REMOTO	
OBJETIVO PRINCIPAL: Atención de solicitudes de por cada una de las etapas de un Proyecto y Obras para ser tramitados por los Interesados en forma digital y virtual.	
DESCRIPCION DE LA SITUACIÓN ACTUAL: - Dado el contexto de la Pandemia el área de Gestión de Proyectos y Obras no están atendiendo en forma presencial en las Unidades Comerciales. - Todas las comunicaciones con los Interesados se están realizando vía correo y se registra en el SGC cada solicitud que ingresan generan un nuevo suministro que no culmina su ciclo de instalación en el Sistema Comercial.	
RESULTADOS ESPERADOS: (indicar las variables medibles antes y después del proyecto). - Contar con una herramienta informática que permita la gestión de los expedientes de proyectos y obras, las solicitudes relacionadas y los documentos asociados en forma digital. - Registrar a los Interesados que se le calculará la Contribución Reembolsable y los documentos asociados a este proceso, de forma digital. - Contar con una base de datos de Interesados que podrían ser potenciales Clientes de la Empresa Vinculada Cantalloc.	

OTRAS ALTERNATIVAS DE SOLUCIÓN:
Alternativa 1: Gestionar vía correo electrónico y celular el contacto con los Interesados y potenciales interesados de los Proyectos y Obras.
Alternativa 2: Continuar gestionando la información en el Sistema Comercial generando suministros sin culminar el proceso.

Presup. Estimado (US$): **Presup. Estimado (US$) (Altern. 1):** **Presup. Estimado (US$) (Altern. 2):**	**Incluido en el CAPEX:** **SI** **NO**

Prioridad	1	2	3	4

Usuario solicitante: Ruben Pacheco V°B° Gerente Área Proponente: _____

Fecha: 22/05/2020

ANEXO III

Acta de Reunión – Relevamiento de Información

ElectroDunas	ACTA DE REUNIÓN	Versión 1
		GT-R-2-6-0-2

Información del Proyecto

Nombre del Proyecto:	**PRIMERA - PORTAL DE PROYECTOS COMERCIALES**		
Código del Proyecto:		Acta Nº:	
Preparado Por:		Fecha:	21/05/2020
Propósito:	RELEVAMIENTO DE INFORMACIÓN.		

Datos de Reunión

Tipo:	TOMAR CONOCIMIENTO DEL PROYECTO				
Alcance:	PROYECTOS COMERCIALES				
Fecha de Reunión:	21/05/2020	Hora Inicio:	11:30	Hora Fin:	12:30
Lugar de Reunión:	Webex				
Coordinador Reunión:	Eduardo Miranda				

Participantes

Nombre del Participante	Gerencia / Empresa	Firma
Gustavo Michelena (GM)	G. Comercial	
Antonio Chocano (AC)	G. Comercial	
Jhan Carlos Sanchez (JCS)	G. Comercial	
Rubén Pacheco (RP)	G. Comercial	
Eduardo Miranda (EM)	G. Tecnología Informática	
Arturo Quintero (AQ)	G. Tecnología Informática	
Cristhian Vila (CV)	G. Tecnología Informática	
Cecilia Cruzado C.	G. Tecnología Informática	

Minuta de Reunión

Se llevó a cabo la reunión con la G. Comercial con la finalidad de tener un primer alcance de la solicitud de desarrollo de un portal de Proyectos Comerciales.

Los puntos que podemos resaltar de este reunión son:

1. ***Objetivo del Portal de Proyectos Comerciales.***
   Contar con un aplicativo para el que el Cliente pueda gestionar sus Proyectos y solicitudes sin necesidad de tener que ir físicamente a una Oficina Comercial y el área de Proyecto pueda atenderlos e interactuar con ellos sin la necesidad de gestionar físicamente un documento (cero papel).

2. ***Roles en el Portal:***
   - Acceso a los Clientes para que puedan registrar sus Proyectos y solicitudes adjuntando los requisitos necesarios en cada una de las etapas del Proyecto.
   - Acceso al área de Proyectos para que les permite evaluar los documentos dejados por el Cliente en cada una de sus etapas.
   - Acceso al área de Ingeniería para que les permita evaluar técnicamente los documentos y puedan enviar sus observaciones o conformidad.
   - Acceso a la G. Comercial para que apruebe u observe lo evaluado por el área de ingeniería.

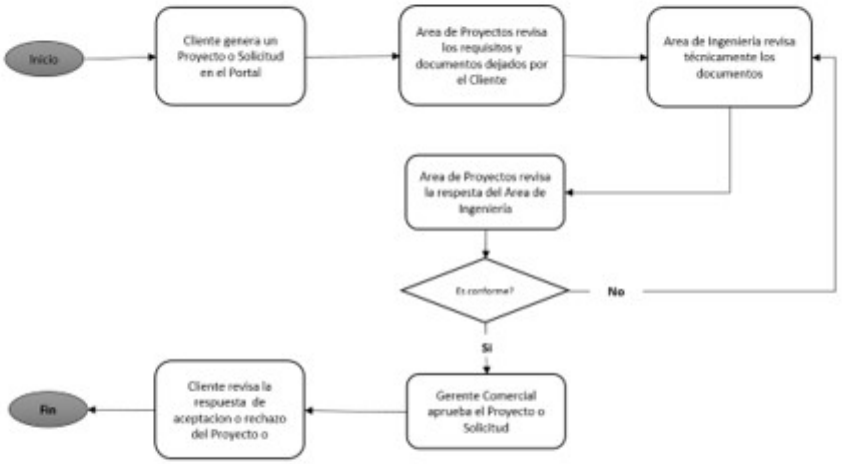
3. ***Documentos para el relevamiento:***
   - MENU DEL PORTAL WEB GESTION DE PROYECTOS DE TERCEROS.docx → Contiene las posibles funcionalidades con que debe de contar el portal a desarrollar.

- GI-P-00X_Proyectos Digitales de Terceros (corregido2).docx → Contiene los lineamientos para la atención de solicitudes de factibilidad de suministro y fijación de puntos de diseño, revisión y aprobación de proyectos tramitados por terceros en forma digital. Este documento aún se encuentra en revisión por parte de la G. Técnica.

4. ***Otros Proceso que se pueden gestionar en el portal:***
   Los proceso que se comentaron que se podría gestionar en el portal son:

   - Contribuciones reembolsables → Proceso documentario y pago al Cliente. En este proceso se interactúa con el área de Finanzas.
   - Proceso de activación del Activo → Cuando culmina la obra y se energiza y entra en operación se debe dibujar en IGEA y posteriormente en Maximo.

5. ***Flujo del Proceso:***
   A continuación se bosqueja el flujo que nos comentaron en la reunión y que se bajará de nivel luego que revisemos la documentación del punto 3 y de todos los documentos u observaciones que nos envíen respecto de este proceso.



6. ***Consideraciones:***
   - En este portal no se gestionará informes regulatorios.
   - Se llevará el conteo de los plazos de atención para la gestión interna de Electro Dunas y no debe ser visible para el Cliente.

7. ***Responsable del Proyecto por parte de la Gerencia Técnica.***
   - Jhan Carlos Sanchez y Rubén Pacheco → Coordinaciones para el proyecto del portal
   - Antonio Chocano → Consultas del proceso.

Compromisos:

No.	Título / Descripción	Responsable	Fecha	Estado
1.	Revisión de los documentos entregados	GTI	No Definida.	Pendiente
2.	Diagrama del Flujo del Proceso en el Portal y Pantallas de cómo se gestionaría la información en el aplicativo	GTI	No Definida	Pendiente
3.	Reuniones de Coordinación para obtener más detalle del desarrollo	GTI	No Definida.	Pendiente

ANEXO IV

Acta de Reunión – Revisión de Flujo de Proceso

ElectroDunas	ACTA DE REUNIÓN	Versión 1
		GT-R-2-6-0-2

Información del Proyecto

Nombre del Proyecto:	**TERCERA REUNIÓN - PORTAL DE PROYECTOS Y OBRAS DE CLIENTES**		
Código del Proyecto:		Acta Nº:	
Preparado Por:		Fecha:	04/08/2020
Propósito:	Revisión del Flujo de Proceso de elaboración de los presupuestos durante el proceso de un Expediente		

Datos de Reunión

Tipo:	Revisión del Flujo de Proceso de elaboración de los presupuestos durante el proceso de un Expediente				
Alcance:	GESTION DE EXPEDIENTES Y SOLICITUDES POR LA GERENCIA COMERCIAL Y GERENCIA TÉCNICA.				
Fecha de Reunión:	04/08/2020	Hora Inicio:	08:00	Hora Fin:	09.30
Lugar de Reunión:	Teams				
Coordinador Reunión:	Cecilia Cruzado C.				

Participantes

Nombre del Participante	Gerencia / Empresa	Nombre del Participante	Gerencia / Empresa
Jhan Carlos Sanchez (JCS)	Gerencia Comercial	**Melissa Trujillo (MT)**	G. Tecnología Informática
Rubén Pacheco (RP)	Gerencia Comercial	**Cecilia Cruzado C (CCC).**	G. Tecnología Informática
José Torres (JT)	Gerencia Comercial	**Arturo Quintero (AQ)**	G. Tecnología Informática
Eddy Matias Rojas (EMR)	Gerencia Técnica	**Cristhian Vila (CV)**	G. Tecnología Informática
Javier Cornejo Tuccno (JC)	Gerencia Técnica	**Cristhian Cahua (CC)**	G. Tecnología Informática

Minuta de Reunión

Objetivo: Revisar el flujo del proceso de la elaboración de un presupuesto producto de una solicitud ingresada por el Usuario solicitante y como se relaciona con los Sistemas SGC y Máximo.

Desarrollo de la Explicación del Flujo del Proceso: EMR, explico que para elaborar el presupuesto ate una solicitud de un Usuario Externo lo realizan en una hoja Excel. El contenido de las hojas Excel son:

1. Planilla de Materiales y Servicios por tipo de Red → En esta planilla contienen la relación de materiales separado por familia o subfamilia del material. La codificación que se les ha colocado no corresponde al maestro de materiales que se gestiona en Máximo. El precio asociado a estos materiales es un precio que se ha establecido cobrar al Cliente por cada unidad.

   En cada una de las planillas de materiales o servicios tienen como columnas los números de los puntos donde han identificado que vana realizar el trabajo y allí colocan la cantidad de los materiales a utilizar.

2. Resumen → Toma la información de las planillas antes mencionadas para totalizar los costos y hallar costos asociados al presupuesto.

3. Formato del Presupuesto → En este documento tiene el formato de una proforma donde indican la condición de pago, el tipo de documento a emitir y el modo de pago.

Al revisar el ejemplo que EMR nos mostraba, revisamos el flujo que se seguía en el SGC y Máximo y nos comentó lo siguiente:

1. Cuando el Usuario Externo ingresa la solicitud por atención a Cliente ya sea para un Cliente de Electro Dunas existe o no, le generan una solicitud que da origen a una NIS que llamaremos "A".
2. El área de gestión de proyectos le solicita al área de proyectos y obras de la Gerencia Técnica realizar el presupuesto para la solicitud ingresada por el cliente y en el documento ingresado hacen referencia al NIS "A".

3. El área de proyectos y obras de la Gerencia Técnica elabora el presupuesto en la hoja Excel descrita líneas arriba y en el formato de presupuesto hacen referencia al NIS "A".
4. El usuario externo se acera a la unidad comercial de ElectroDunas para pagar el presupuesto y para el ejemplo revisado no le emiten el documento por cobrar al NIS "A" sino a un NIS "B". En este punto la relación del presupuesto con el documento en el SGC ya se pierde.
5. Cuando el área de Grandes Clientes se entera del pago por parte del Usuario Externo procede a generar una nueva solicitud del tipo "Solicitud de Obras" y subtipo "Obras por Encargo" generando un NIS "C".
6. Al generarse la O/T en el punto anterior viaja a Maximo donde finalmente sigue su flujo hasta su culminación.
7. EMR, comentó que existen solicitudes de presupuestos de Usuarios Externo que no necesariamente tienen un Proyecto u Obra sino que son pedidos puntuales y que se necesita generarles un presupuesto.

Qué se concluyó:

1. Que en el SGC se están generando tantos códigos NIS como solicitudes se ingresan y que finalmente no tienen relación con la solicitud del Cliente.
2. Revisar la factibilidad, a partir de la fecha de:
   2.1. La generación de los presupuestos se siga realizando como hasta ahora, sin relacionarlo a un numero de NIS sino a los datos del cliente que finalmente van a Cancelar el Presupuesto. (Este punto hasta cuando el Portal se encuentre en Producción).
   2.2. Cuando el Cliente se acerque a atención del Cliente a Cancelar en caso tenga un NIS generado pague por ese documento o le generen uno nuevo y único para todos sus trámites.
   2.3. Generar las O/Ts para la ejecución del presupuesto directo en Máximo como OT cuyo tipo es de Obras por Encargo que no este relacionado a un Proyecto CAPEX sino direccionado a otros APIS de otras áreas. Con este punto evitamos que se generen códigos NIS que finalmente no se cierran su proceso.
   → EMR, revisará en Máximo si es posible generar las OT del punto 2.3 en Producción por que el entorno de PRUEBAS o CAPACITACIÓN no está disponible.
3. De ser factible el punto 2.3 los pasos a seguir serian:
   3.1. Reunión con las áreas involucradas y responsables para formalizar el flujo de trabajo
   3.2. Incluir este proceso dentro del desarrollo del portal.
   3.3. La funcionalidad de elaboración de un presupuesto será desarrollada para ser usada para un usuarios tenga o no un expediente relacionado.

Compromisos:

No.	Título / Descripción	Responsable	Fecha	Estado
1.	Revisar en Máximo si es posible crear Ordenes de Trabajo del Tipo Obras por encargo y si puede generarse todo el flujo.	Eddy Matias	Esta semana	Pendiente
2.	Enviar la plantilla de presupuestos para analizar y ver la factibilidad de adicionar como una opción dentro del Portal.	Eddy Matias	Esta semana	Pendiente
3.	Enviar las cartas de respuesta a los Usuarios Externos en formato PDF.	Eddy Matias	Esta semana	Pendiente

ANEXO V

Documentación Visión del Sistema

ElectroDunas	DOCUMENTO VISIÓN	Versión 1
		GT-R-2-6-0-3

1. INTRODUCCIÓN

Dado el contexto de Pandemia que vivimos en nuestro País, el área de Gestión de Proyectos y Obras de la Gerencia Comercial tiene la necesidad de contar con un Portal donde pueda gestionar los expedientes y solicitudes de los Interesados en la elaboración de Proyectos y Obras de Sistema de Distribución y Utilización dentro de la concesión de Electrodunas.

1.1 Propósito

- Contar con una herramienta informática que permita la comunicación virtual entre el área de Gestión de Proyectos y Obras con los Interesados para gestionar los expedientes de proyectos y obras, las solicitudes relacionadas a los expedientes y los documentos asociados de forma digital.
- Registrar a los Interesados que se le calculará la Contribución Reembolsable y los documentos asociados a este proceso, de forma digital.
- Contar con una base de datos de Interesados que podrían ser potenciales Clientes de la Empresa Vinculada Cantalloc.

1.2 Alcance

La herramienta informática servirá para el trámite de los Expedientes y Solicitudes de Proyectos y obras del Sistema de Distribución y Sistemas de Utilización elaborados por Terceros dentro del Área de Concesión de Electro Dunas y será el medio de comunicación entre el Interesado y Electro Dunas. También, nos permita suprimir el papel a cero por que la comunicación y documentos será virtual.

2. TERMINOLOGÍA

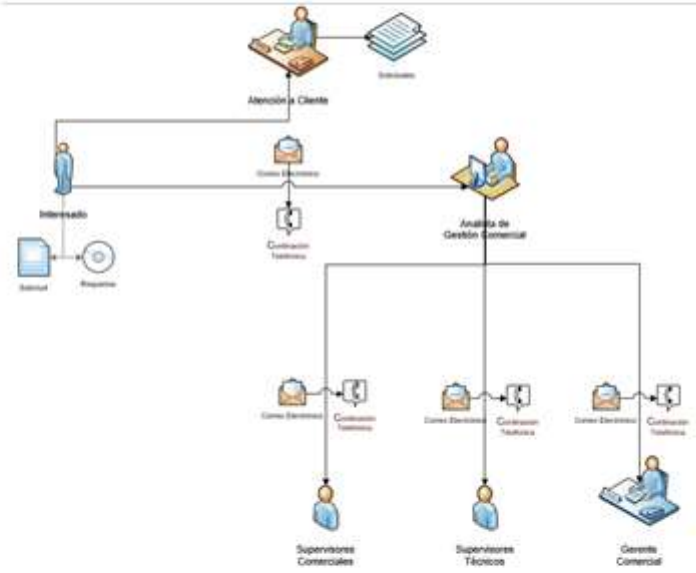
Sistema de Distribución (SD)	Es el conjunto de instalaciones eléctricas comprendidas desde un sistema de generación o transformación a media tensión, hasta los puntos de entrega de los usuarios de media o baja tensión, inclusive las unidades de alumbrado público.
Sistema de Distribución Primaria (SDP)	Son las redes y subestaciones cuyas tensiones de servicio son mayores de 1 kV y menores de 30 kV.
Interesado	Persona natural o jurídica debidamente identificada, encargada de la gestión ante el Concesionario para la dotación y uso del suministro de energía eléctrica en un predio o conjunto de predios o lotes.
Usuario Interno	Persona natural que labora en la Empresa que es dueña del Portal y que gestiona los expedientes de los proyectos y obras de los Interesados
Usuario Externo	Es un Interesado.
Analista Gestor Comercial	Persona natural que labora en la Gerencia Comercial

	de la Empresa.
Supervisor Técnico	Persona natural que labora en la Gerencia de Operaciones de la Empresa.

3. DEFINICIÓN DEL PROBLEMA

3.1 Breve Descripción

Los Interesados dado el contexto en que vive el País no pueden tramitar sus Expedientes y solicitudes de manera presencial en las Unidades Comerciales por lo que se hace necesario el desarrollo de un portal para que puedan gestionar estos documentos y envíen los requisitos digitalmente.

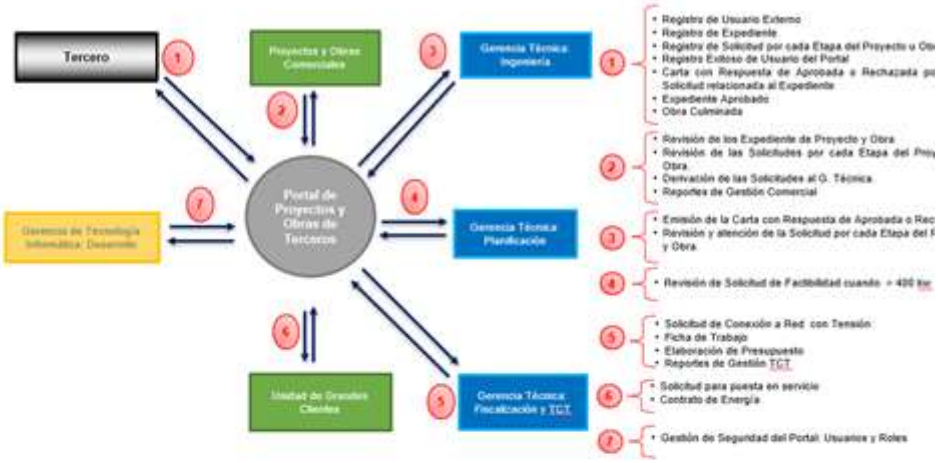


3.2 Oportunidad de Negocio

Registrar y generar el repositorio de información de los Interesados de los Proyectos y Obras de Terceros. De este universo se podrá conocer aquellos que no cuentan con un Proyectista y podría ser oportunidad de negocio para la empresa vinculada Cantalloc.

4. DIAGRAMA DE CONTEXTO DEL SISTEMA

El presente Diagrama de Contexto de Sistema define parte de las funcionalidades del sistema y muestra las entidades que interactuaran con él.



4. DESCRIPCIÓN DE STAKEHOLDERS Y USUARIOS

4.1 Resumen de Stakeholders

Nombre	Descripción	Responsabilidades
Gustavo Michelena	Gerente Comercial	• Solicitar la herramienta tecnológica para poder automatizar las actividades del área de gestión de proyectos y obras de su Gerencia. • Aprobar las respuestas remitidas por la Gerencia Técnica – área de proyectos y obras
Eduardo Miranda	Gerente de Tecnología Informática	Poner a disposición la herramienta tecnológica solicitada por la Gerencia Comercial.

4.2 Resumen de Usuarios

Nombre	Descripción	Responsabilidades
Ruben Pacheco	Analista de Gestión Comercial	Responsable de gestionar los Expedientes y Solicitudes de los Interesados de Proyectos y Obras de Terceros.
Jhan Carlos Sanchez	Supervisor	Responsable de gestionar la información de los Interesados para obtener Proyectos u Obras para Cantalloc

Gerencia Técnica - Proyectos y Obras	Supervisor	Responsable de revisar y gestionar las solicitudes de las diferentes etapas de un Proyecto u Obra. También es responsable de un Supervisor para la Obra a ejecutarse.
Gerencia Técnica - Planificación	Supervisor	Responsable de revisar la solicitud de factibilidad de un Sistema de Distribución cuando un proyecto tiene una demanda mayor de 400 kw.
Gerencia Comercial – Contribución Reembolsable	Analista de Gestión Comercial	Responsable de recibir la conformidad de Obra y de subir a la herramienta el cálculo del VNR.

5. DESCRIPCIÓN DE LOS PERFILES DE STAKEHOLDERS Y USUARIOS

5.1 Perfil de los Stakeholders

5.1.1 Stakeholder 1

Área Funcional	Gerencia Comercial
Descripción	Gerente Comercial
Tipo	~~StakeHolder~~ / Usuario Interno
Responsabilidades	• Solicitar la herramienta tecnológica para poder automatizar las actividades del área de gestión de proyectos y obras de su Gerencia. • Aprobar las respuestas remitidas por la Gerencia Técnica – área de proyectos y obras
Grado de participación	Aprobador
Comentarios	

Área Funcional	Gerencia de Tecnología Informática
Descripción	Gerente de Tecnología Informática
Tipo	~~StakeHolder~~
Responsabilidades	Poner a disposición la herramienta tecnológica solicitada por la Gerencia Comercial.
Grado de participación	Apoyo para el desarrollo de la herramienta tecnológica.
Comentarios	

5.2 Perfil de los Usuarios

5.2.1 Usuario Interno

Área Funcional	Gerencia Comercial
Descripción	Gestor de Proyectos y Obras Comercial / Analista de Grandes Clientes
Tipo	Usuario Interno
Responsabilidades	Gestionar en el portal los expedientes y solicitudes de los Usuarios Externos.
Grado de participación	Revisa y Deriva las Transacciones en el Portal.
Comentarios	Facilita el tramite de las solicitudes de los expedientes de los usuarios externos.

Área Funcional	Gerencia Comercial
Descripción	Jefe de Proyectos y Obras Comercial
Tipo	Usuario Interno
Responsabilidades	Consultar los potenciales Clientes para la empresa vinculada Cantalloc.
Grado de participación	Consulta de Clientes, expedientes y solicitudes.
Comentarios	

Área Funcional	Gerencia Técnica
Descripción	Supervisor de Proyectos y Obras / Supervisor de TCT / Supervisor de Planificación
Tipo	Usuario Interno
Responsabilidades	Revisa y resuelve las Solicitudes presentadas por el Usuario Externo
Grado de participación	Consulta de Clientes, expedientes y solicitudes.
Comentarios	

5.2.2 Usuario Externo

Representante	Interesados
Descripción	Usuarios Externos que se registraran en el Portal y que están interesados en desarrollar sus Proyectos y Obras dentro de la concesión de Electro Dunas.
Tipo	Usuario Externo
Responsabilidades	Registrarse en el Portal y gestionar sus Expedientes y Solicitudes. También consultara los tipos de proyectos, requisitos y armados constructivos.
Grado de participación	Generará transacciones en el Portal.
Comentarios	

5.2.3 Usuario Administrador

Área Funcional	Gerencia de Tecnología Informática
Descripción	Analista de Desarrollo
Tipo	Usuario Interno
Responsabilidades	Gestionara los usuarios Internos y Externos, así como la asignación de grupos o Roles de los mismos. Tendrá acceso a toda la funcionalidad que ofrece el Portal.
Grado de participación	Generará transacciones en el Portal.
Comentarios	Se recomienda que sea del área de Tecnología Informática.

6. NECESIDADES CLAVES DE LOS STAKEHOLDERS Y USUARIOS

Necesidad	Prioridad	Concierne a:	Solución actual	Solución propuesta
Atender los Expedientes y Solicitudes de los Proyectos y Obras de Terceros	Alta	Gerencia Comercial	Correo Electrónico y Llamadas por Celular.	Portal para la Gestión de Expedientes y Solicitudes de Proyectos y Obras de Terceros.
Reducir el papel en la presentación de los Requisitos de las Solicitudes	Alta	Gerencia Comercial	Entrega de Requisitos en mesa de partes.	Digitalizar la entrega de requisitos y guardarlos en un repositorio.
Generar Clientes para la empresa	Alta	Gerencia Comercial	Con base a la experiencia se sabe	El Interesado al registrase en el

vinculada Cantalloc			que Clientes cuentan o no con un Proyectista o Constructor.	portal se sabrá esta información generando una Cartera de Clientes.
Revisión de las respuestas de la Gerencia Técnica	Alta	Gerencia Comercial	Coordinación por Correo.	Esta área interactuará en el Portal donde podrá atender cada solicitud.

ANEXO VI

Acta de Reunión – Pruebas de usuario

ElectroDunas	ACTA DE REUNIÓN	Versión 1
		GT-R-2-6-0-2

Información del Proyecto

Nombre del Proyecto:	**CUARTA REUNIÓN - PORTAL DE PROYECTOS Y OBRAS DE CLIENTES – SEGUNDA PRUEBA CON USUARIOS**		
Código del Proyecto:		Acta Nº:	
Preparado Por:		Fecha:	16/12/2020
Propósito:	Pruebas con Usuario de la G. Comercial y técnica		

Datos de Reunión

Tipo:	Coordinaciones para el Desarrollo				
Alcance:	PRUEBAS TRANSACCIONALES DE LOS EXPEDIENTES Y SOLICITUDES POR LA GERENCIA COMERCIAL Y GERENCIA TÉCNICA.				
Fecha de Reunión:	16/12/2020	Hora Inicio:	15:00	Hora Fin:	17:00
Lugar de Reunión:	Teams				
Coordinador Reunión:	Cecilia Cruzado				

Participantes

Nombre del Participante	Gerencia / Empresa	Nombre del Participante	Gerencia / Empresa
Jhan Carlos Sanchez (JCS)	Gerencia Comercial	**Eduardo Miranda (EM)**	G. Tecnología Informática
Rubén Pacheco (RP)	Gerencia Comercial	**Cecilia Cruzado C (CCC).**	G. Tecnología Informática
Elizabeth Goicochea (EG)	Gerencia Comercial	**Arturo Quintero (AQ)**	G. Tecnología Informática
Leonardo Charcas	Gerencia Técnica	**Yorghinio Valenzuela (YV)**	G. Tecnología Informática
Emerson Navarrete Sotelo	Gerencia Técnica		

Minuta de Reunión

1. **Pantalla Inicio del Portal** →
   - CC, coordinara con Ronny Romero de la Gerencia Legal para revisar el texto por la protección de datos personales.
   - JCS, debe generar el texto de las consideraciones que acepta el Cliente al registrarse al Portal.

2. **Expedientes**:
   - Retirar del Expediente la funcionalidad del histórico de usuarios cuando ingresa un Usuario Externo. Solo es visible para Usuario Interno.
   - La Fecha Inicio y Fin solo se debe llena al Aprobar la solicitud de Aprobación de Proyectos. En ese momento la Fecha Inicio será igual al día que se aprobó la solicitud y la Fecha Fin será la Fecha inicio más dos (02) años.
   - En la pantalla principal de los expedientes cuando se aplica la funcionalidad de ver las solicitudes relacionadas al Expediente la columna estado debe estar centrada al igual que la data.

3. **Solicitudes**:
   - Retirar de la Solicitud la funcionalidad de ver el historio de estados del usuario externo. Solo debe ser visible para el usuario Interno.

Compromisos:

No.	Título / Descripción	Responsable	Fecha	Estado
1.	Envío de correo a Legal por Ley de Protección de Datos	Cecilia Cruzado	17/12/2020	Terminado

No.	Título / Descripción	Responsable	Fecha	Estado
2.	Envío de las consideraciones para el Registro de un Clientes Externo con base a la norma de Proyectos y Obras y comunicaciones que tienen los Clientes.	Jhan Carlo Sanchez	Esta semana	Pendiente
3.	Agendar reunión para las próximas dos (02) sesiones	Cecilia Cruzado	16/12/2020	Terminado

ElectroDunas	ACTA DE REUNIÓN	Versión 1
		GT-R-2-6-0-2

Información del Proyecto

Nombre del Proyecto:	**PORTAL DE PROYECTOS Y OBRAS DE CLIENTES – TERCERA PRUEBA CON USUARIOS**		
Código del Proyecto:		Acta NR:	
Preparado Por:		Fecha:	18/12/2020
Propósito:	Pruebas con Usuario de la G. Comercial y técnica		

Datos de Reunión

Tipo:	Coordinaciones para el Desarrollo				
Alcance:	PRUEBAS TRANSACCIONALES DE LOS EXPEDIENTES Y SOLICITUDES POR LA GERENCIA COMERCIAL Y GERENCIA TÉCNICA.				
Fecha de Reunión:	17/12/2020 18/12/2020	Hora Inicio:	15:00	Hora Fin:	17:00
Lugar de Reunión:	Teams				
Coordinador Reunión:	Cecilia Cruzado				

Participantes

Nombre del Participante	Gerencia / Empresa	Nombre del Participante	Gerencia / Empresa
Jhan Carlos Sanchez (JCS)	Gerencia Comercial	**Cecilia Cruzado C (CCC).**	G. Tecnología Informática
Elizabeth Goicochea (EG)	Gerencia Comercial	**Yorghinio Valenzuela (YV)**	G. Tecnología Informática
Leonardo Charcas	Gerencia Técnica		
Emerson Navarrete Sotelo	Gerencia Técnica		
Javier Cornejo (JC)	Gerencia Técnica		
Angel Garcia (AG)	Gerencia Técnica		
José Mendoza (JM)	Gerencia Técnica		

Minuta de Reunión

Pantalla Inicio del Portal →

- Se explico a JCS, que debe enviar el Texto de las consideraciones generales de la inscripción del usuario externo al Portal.

Expedientes:

- Cuando un Usuario externo crea un Expediente el correo que le llega debe estar en copia los Usuarios Internos cuyo rol es "Revisor Comercial".
- En la pantalla principal de los Expedientes debe haber un botón refrescar para que actualicen las ultimas solicitudes ingresadas o que hayan cambiado de estado.
- Se debe habilitar la funcionalidad que en un Expediente se relaciones con un expediente anterior. Y solo se debe habilitar cuando el Tipo de Proyecto se haya parametrizado de ese modo.

Solicitudes:

- Cuando un Usuario externo crea una Solicitud el correo que le llega debe estar en copia los Usuarios Internos cuyo rol es "Revisor Comercial".
- En la pantalla principal de las Solicitudes debe haber un botón refrescar para que actualicen las ultimas solicitudes ingresadas o que hayan cambiado de estado.
- Cuando la solicitud esta siendo atendido por el Rol de "Para Firmas" no le debe permitir grabar la solicitud o enviarla sino ha subido la carta con las firmas.
- Cuando el Usuario Externo este subiendo lo requisitos debe existir una validación de extensión de archivo. Y en caso no cumpla que le envíe un mensaje de advertencia que no podrá subir el archivo ni permitir grabar la solicitud.

- Cuando un "Revisor técnico" esta atendiendo una solicitud al momento de enviar debe permitirle elegir entre dos Roles: "Revisor Comercial" y "Revisor Técnico". El motivo es que los revisores técnicos se pueden reenviar entre ellos.
- Y Cuando elija uno de los dos roles en la ventana emergente para seleccionar los usuarios debe listar solo a las personas del Rol elegido para el envío. Ej: Si elige "Revisor Comercial" deben listarse los usuarios Jhan Carlos Sanchez y Ruben Pacheco.

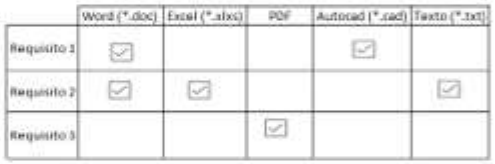
Carga Inicial:

- Se insistió que se debe de revisar la carga inicial de los Requisitos de los Tipos de solicitudes y de los Tipos de Solicitudes por cada Tipo de Proyecto.

Reportes Gerencia Técnica – Etapa I:

- Javier Cornejo comentó que necesitaría reportes en esta primera etapa y enviará el formato de reportes.

Resumen de las Observaciones:

Funcionalidad	Detalle de los cambios
Inicio de Sesión	• Cuando JCS, envíe el texto de las consideraciones debe haber un link o ventana emergente que las muestre y el usuario externo las pueda leer.
Tipo de Proyecto	• Se debe colocar el check que un Tipo de Proyecto se puede relacionar con otro conforme se ha colocado en la especificación funcional.
Requisitos	• Se debe colocar en cada uno de los requisitos de los Tipos de Solicitud el tipo de formato que acepta. Le debe permitir colocar mas de una selección. 
Expedientes	• Cuando un Usuario externo crea un Expediente el correo que le llega debe estar en copia los Usuarios Internos cuyo rol es "Revisor Comercial". • En la pantalla principal de los Expedientes debe haber un botón refrescar para que actualicen las ultimas solicitudes ingresadas o que hayan cambiado de estado. • Se debe habilitar la funcionalidad que en un Expediente se relaciones con un expediente anterior. Y solo se debe habilitar cuando el Tipo de Proyecto se haya parametrizado de ese modo.
Representantes	Ninguno.
Solicitudes	• Cuando un Usuario externo crea una Solicitud el correo que le llega debe estar en copia los Usuarios Internos cuyo rol es "Revisor Comercial". • En la pantalla principal de las Solicitudes debe haber un botón refrescar para que actualicen las ultimas solicitudes ingresadas o que hayan cambiado de estado. • Cuando la solicitud está siendo atendido por el Rol de "Para Firmas" no le debe permitir grabar la solicitud o enviarla sino ha subido la carta con las firmas. • Cuando el Usuario Externo este subiendo lo requisitos debe existir una validación de extensión de archivo. Y en caso no cumpla que le envíe un mensaje de advertencia que no podrá subir el archivo ni permitir grabar la solicitud. • Cuando un "Revisor técnico" está atendiendo una solicitud al momento de enviar debe permitirle elegir entre dos Roles: "Revisor Comercial" y "Revisor Técnico". El motivo es que los revisores técnicos se pueden reenviar entre ellos. • Y Cuando elija uno de los dos roles en la ventana emergente para seleccionar los usuarios debe listar solo a las personas del Rol elegido para el envío. Ej: Si elige "Revisor Comercial" deben listarse los usuarios Jhan Carlos Sanchez y Ruben Pacheco.
Adjuntos	• Cuando suba un adjunto de un requisito debe de validar la extensión y si no cumple debe enviar un mensaje de advertencia.

	Word (*.doc)	Excel (*.xlxs)	PDF	Autocad (*.cad)	Texto (*.txt)
Requisito 1	☑			☑	
Requisito 2	☑	☑			☑
Requisito 3			☑		

ElectroDunas	ACTA DE REUNIÓN	Versión 1
		GT-R-2-6-0-2

Compromisos:

No.	Título / Descripción	Responsable	Fecha	Estado
1.	Envío de las consideraciones para el Registro de un Clientes Externo con base a la norma de Proyectos y Obras y comunicaciones que tienen los Clientes.	Jhan Carlo Sanchez	21/12/2020	Pendiente
2.	Revisión de la Carga inicial de los Tipos de Solicitudes y de los Requisitos por cada una de ellas.	Emerson Navarrete	21/12/2020	Pendiente
3.	Deben de enviar las extensiones de archivos que deben de enviar los requisitos. Esto debe configurarse por cada requisito.	Emerson Navarrete Jhan Carlo Sanchez	21/12/2020	Pendiente
4.	Envío de formatos de Reportes de la G. técnica para esta primera etapa.	Javier Cornejo	21/12/2020	Pendiente
5.	Crear 04 usuarios para la G. técnica con el Rol "Revisor técnico" para: • Javier Cornejo • Angel Garcia • José Mendoza • Leonardo Charca	Yorghinio Valenzuela Cecilia Cruzado	21/12/2020	Pendiente
6.	Agendar las reuniones del 22/12 y 23/12	Cecilia Cruzado	18/12/2020	Terminado.

ElectroDunas	ACTA DE REUNIÓN	Versión 1
		GT-R-2-6-0-2

Información del Proyecto

Nombre del Proyecto:	**PORTAL DE PROYECTOS Y OBRAS DE CLIENTES – TERCERA PRUEBA CON USUARIOS**		
Código del Proyecto:		Acta Nº:	
Preparado Por:		Fecha:	22/12/2020
Propósito:	Pruebas con Usuario de la G. Comercial y técnica		

Datos de Reunión

Tipo:	Coordinaciones para el Desarrollo				
Alcance:	PRUEBAS TRANSACCIONALES DE LOS EXPEDIENTES Y SOLICITUDES POR LA GERENCIA COMERCIAL Y GERENCIA TÉCNICA.				
Fecha de Reunión:	22/12/2020	Hora Inicio:	15:00	Hora Fin:	17:00
Lugar de Reunión:	Teams				
Coordinador Reunión:	Cecilia Cruzado				

Participantes

Nombre del Participante	Gerencia / Empresa	Nombre del Participante	Gerencia / Empresa
Jhan Carlos Sanchez (JCS)	Gerencia Comercial	**Eduardo Miranda (EM).**	G. Tecnología Informática
Ruben Pacheco (RP)	Gerencia Comercial	**Cecilia Cruzado C (CCC).**	G. Tecnología Informática
Elizabeth Goicochea (EG)	Gerencia Comercial	**Yorghinio Valenzuela (YV)**	G. Tecnología Informática
Leonardo Charcas	Gerencia Técnica		
Emerson Navarrete Sotelo	Gerencia Técnica		
Javier Cornejo (JC)	Gerencia Técnica		
Angel Garcia (AG)	Gerencia Técnica		
José Mendoza (JM)	Gerencia Técnica		
César Saravia (CS)	Gerencia Técnica		

Minuta de Reunión

Pantalla Inicio del Portal →

- Se solicitó a JCS que envíe el texto para esta sección.

Expedientes:

- Cuando se coloque el expediente relacionado debe colocarse en no editable la descripción del proyecto y el sistema eléctrico relacionado.

Solicitudes:

- En la descripción de los requisitos se debe concatenar las extensiones de los archivos que el usuario externo puede subir. Esto debe estar en negritas.
- Cuando el Revisor Comercial Aprueba o Rechaza una solicitud debe llegar un correo al Revisor Tecnico.
- La alerta de no dejar grabar la solicitud cuando el Rol para Firmas no sube la carta de respuesta debe salir luego que da al botón grabar.

Campaña Publicitaria del Portal:

- JCS, coordinará con EM cual será el contenido de la campaña del portal en la página web.
- El portal de proyecto y obras tendrá un link dentro de la pagina web de ELD.

Resumen de las Observaciones:

Funcionalidad	Detalle de los cambios

ElectroDunas	ACTA DE REUNIÓN	Versión 1
		GT-R-2-6-0-2

Inicio de Sesión	• Cuando JCS, envíe el texto de las consideraciones debe haber un link o ventana emergente que las muestre y el usuario externo las pueda leer.
Tipo de Proyecto	Ninguno
Requisitos	Ninguno.
Expedientes	• Cuando se coloque el expediente relacionado debe colocarse en no editable la descripción del proyecto y el sistema eléctrico relacionado
Representantes	Ninguno.
Solicitudes	• En la descripción de los requisitos se debe concatenar las extensiones de los archivos que el usuario externo puede subir. Esto debe estar en negritas. • Cuando el Revisor Comercisl Aprueba o Rechaza una solicitud debe llegar un correo al Revisor técnico. • La alerta de no dejar grabar la solicitud cuando el Rol para Firmas no sube la carta de respuesta debe salir luego que da al botón grabar.
Adjuntos	Ninguno.

Compromisos:

No.	Título / Descripción	Responsable	Fecha	Estado
1.	Envío de las consideraciones para el Registro de un Clientes Externo con base a la norma de Proyectos y Obras y comunicaciones que tienen los Clientes.	Jhan Carlo Sanchez	Esta Semana	Pendiente
2.	Revisión de la Carga inicial de los Tipos de Solicitudes y de los Requisitos por cada una de ellas.	Emerson Navarrete	21/12/2020	Retrasado
3.	Deben de enviar las extensiones de archivos que deben de enviar los requisitos. Esto debe configurarse por cada requisito.	Emerson Navarrete Jhan Carlo Sanchez	21/12/2020	Retrasado
4.	Envío de formatos de Reportes de la G. técnica para esta primera etapa.	Javier Cornejo	21/12/2020	Retrasado
5.	Cambios en el Portal luego de las Pruebas	Yorghinio Valenzuela	28/12/2020	Pendiente
6.	Campaña Publicitaria del Portal en la Web	Jhan Carlos Sanchez	Esta Semana	Pendiente

ElectroDunas	ACTA DE REUNIÓN	Versión 1
		GT-R-2-6-0-2

Información del Proyecto

Nombre del Proyecto:	**PORTAL DE PROYECTOS Y OBRAS DE CLIENTES – QUINTA PRUEBA CON USUARIOS**		
Código del Proyecto:		Acta Nº:	
Preparado Por:		Fecha:	29/12/2020
Propósito:	Pruebas con Usuario de la G. Comercial y técnica		

Datos de Reunión

Tipo:	Coordinaciones para el Desarrollo				
Alcance:	PRUEBAS TRANSACCIONALES DE LOS EXPEDIENTES Y SOLICITUDES POR LA GERENCIA COMERCIAL Y GERENCIA TÉCNICA.				
Fecha de Reunión:	29/12/2020	Hora Inicio:	15:00	Hora Fin:	19:00
Lugar de Reunión:	Teams				
Coordinador Reunión:	Cecilia Cruzado				

Participantes

Nombre del Participante	Gerencia / Empresa	Nombre del Participante	Gerencia / Empresa
Jhan Carlos Sanchez (JCS)	Gerencia Comercial	**Eduardo Miranda (EM).**	G. Tecnología Informática
Ruben Pacheco (RP)	Gerencia Comercial	**Cecilia Cruzado C (CCC).**	G. Tecnología Informática
Elizabeth Goicochea (EG)	Gerencia Comercial	**Yorghinio Valenzuela (YV)**	G. Tecnología Informática
Eddy Matias (EM)	Gerencia Técnica		
Emerson Navarrete Sotelo	Gerencia Técnica		
Ida Echevarria (IE)	Gerencia Técnica		

Minuta de Reunión

Página Web:
- EM, comentó que JCS debe de coordinar con su Gerencia la confección de la presentación que verá el Usuario Externo luego que da clic a la opción sugerida por JCS.

Campaña Publicitaria del Portal:
- JCS, comentó que la difusión será a través del Sr. Pacheco quien tiene contacto con los usuarios externos.
- EM, solicitó que coordiné con su Gerencia porqué sugiere que se debe realizar a través de la página web.

Pantalla Inicio del Portal →
- Se debe colocar el Texto enviado por JCS.
- CC, comentó que el texto de datos personales de usuarios externos se ha realizado la consulta a la G. Legal.

Expedientes:
- Se debe colocar agrupadores o secciones para ordenar los campos de: Datos del Proyecto, Datos del Proyecto relacionado, Datos de Ubicación, Datos de representantes.
- Cambiar el "label" de Descripción por Nombre de Proyecto.
- En los campos del Expediente relacionado colocar: N° de Expediente Relacionado, Nombre de Proyecto Relacionado y Tipo de Sistema eléctrico Relacionado.
- Al seleccionar un numero de expediente relacionado por defecto debe mostrar el valor "seleccionar".
- Al crear un expediente retirar la funcionalidad de adjuntar archivos.
- En el campo Fecha de Expediente debe mostrar la Fecha y la Hora.

Solicitudes:
- En la descripción de los requisitos se debe concatenar las extensiones de los archivos que el usuario externo puede subir. Esto debe estar en negritas y en todas las funcionalidades: al crear, editar y visualizar.

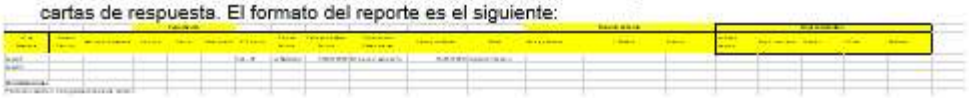
- Al derivar la solicitud a un Rol, luego de grabar debe figurar la lista de las personas que pertenecen a ese rol.
- En el campo Fecha de Solicitud debe mostrar la Fecha y la Hora.
- En la grilla principal los títulos de los campos deben estar centrados.
- Cuando el usuario externo visualiza la solicitud debe poder visualizar los archivos que ha subido el rol de firmas y esto lo debe de ver cuando una solicitud ha sido aprobada o rechazada.
- Debe existir un botón de Adjuntos que permita solo a los usuarios externos (todos los roles) ver todos los archivos que se han ido cargando conforme la solicitud ha sido revisada.

Requisitos:
- JCS, solicito acceso a los maestros para los roles revisor comercial y técnico y revisar la configuración del tipo de archivo que debe subir el Usuario Externo para adjuntar los documentos.

Reportes:
- IE, solicitó un reporte que servirá para informar a su Gerencia el avance de los expedientes y solicitudes. El formato del Reporte es el siguiente:

Consideraciones:
  - Los datos de las solicitudes deben ser del ultimo estado de la solicitud
  - Debe permitir exportar el reporte en PDF y Excel
  - Debe tener filtros por: Tipo de expediente, tipo de solicitud, tipo de sistema eléctrico, estado del expediente, estado de la solicitud, Revisor Técnico.
  - Rango de fechas de las solicitudes (No considerar la hora)

- EM, solicitó un reporte con datos que le servirá para alimentar la hoja de Excel mientras se automatiza las cartas de respuesta. El formato del reporte es el siguiente:



Consideraciones:
  - Los datos de las solicitudes deben de ser del primer envío a Revisión Técnica.
  - Debe permitir exportar el reporte en PDF y Excel
  - Debe tener filtros por: Tipo de expediente, tipo de solicitud, tipo de sistema eléctrico, estado del expediente, estado de la solicitud y rango de fechas de las solicitudes (No considerar la hora).

Procedimiento:
- JCS, debe actualizar, revisar y circular el procedimiento de Proyectos y Obras para conocimiento de las áreas usuarias.

Manual y Video:
- Se realizará el video para tomar conocimiento de la funcionalidad del portal. Desde el registro de un usuario, la creación de un expediente y flujo completo de una solicitud. Esto se realizará en la sesión del 31/12/2020.
- JCS, solicitó que el manual de usuario lo realice TI.

II Etapa:
- Considerar en el flujo del proceso al rol que ejecuta Ida Echevarria. Este rol debe de adjuntar los documentos de respuesta que se debe pasar al revisor comercial.
- Considerar que el Revisor comercial cuando una carta se va a aprobar o rechazar se pueda seleccionar que documentos adjuntos debe visualizar y viajar por correo al usuario externo.

Resumen de las Observaciones:

Funcionalidad	Detalle de los cambios
Inicio de Sesión	• Colocar el texto enviado por JCS en las consideraciones y debe haber un link o ventana emergente que las muestre y el usuario externo las pueda leer.

Tipo de Proyecto	Ninguno
Requisitos	Ninguno.
Expedientes	• Se debe colocar agrupadores o secciones para ordenar los campos de: Datos del Proyecto, Datos del Proyecto relacionado, Datos de Ubicación, Datos de representantes. • Cambiar el "label" de Descripción por Nombre de Proyecto. • En los campos del Expediente relacionado colocar: N° de Expediente Relacionado, Nombre de Proyecto Relacionado y Tipo de Sistema eléctrico Relacionado. • Al seleccionar un número de expediente relacionado por defecto debe mostrar el valor "seleccionar". • Al crear un expediente retirar la funcionalidad de adjuntar archivos. • En el campo Fecha de Expediente debe mostrar la Fecha y la Hora.
Representantes	Ninguno.
Solicitudes	• En la descripción de los requisitos se debe concatenar las extensiones de los archivos que el usuario externo puede subir. Esto debe estar en negritas y en todas las funcionalidades: al crear, editar y visualizar. • Al derivar la solicitud a un Rol, luego de grabar debe figurar la lista de las personas que pertenecen a ese rol. • En el campo Fecha de Solicitud debe mostrar la Fecha y la Hora. • En la grilla principal los títulos de los campos deben estar centrados. • Cuando el usuario externo visualiza la solicitud debe poder visualizar los archivos que ha subido el rol de firmas y esto lo debe de ver cuando una solicitud ha sido aprobada o rechazada. • Debe existir un botón de Adjuntos que permita solo a los usuarios externos (todos los roles) ver todos los archivos que se han ido cargando conforme la solicitud ha sido revisada.
Adjuntos	Ninguno.
Reportes	• Confeccionar los dos reportes solicitados por la Gerencia Técnica.

Compromisos:

No.	Título / Descripción	Responsable	Fecha	Estado
1.	Enviar la presentación que verá el usuario externo cuando de clic al link del Portal de Proyectos y Obras	Jhan Carlo Sanchez	Esta Semana	Pendiente
2.	Revisar con la G. Comercial como se llevará a cabo la campaña al usuario externo del portal y por qué método.	Jhan Carlos Sanchez	Esta Semana	Retrasado
3.	Revisión de la Carga inicial de los Tipos de Solicitudes y de los Requisitos por cada una de ellas.	Eddy Matias Jhan Carlos Sanchez	Esta Semana	Retrasado
4.	Deben de enviar las extensiones de archivos que deben de enviar los requisitos. Esto debe configurarse por cada requisito.	Eddy Matias Jhan Carlos Sanchez	Esta Semana	Retrasado
5.	Envío de formatos de Reportes de la G. técnica para esta primera etapa.	Javier Cornejo	Esta Semana	Retrasado
6.	Revisar la configuración del tipo de archivo que debe subir el Usuario Externo para adjuntar los documentos en los requisitos	Jhan Carlos Sanchez Eddy Matias	Esta Semana	Pendiente
7.	Cambios en el Portal luego de las Pruebas	Yorghinio Valenzuela	31/12/2020	Pendiente
8.	Video de la funcionalidad del Portal durante las pruebas del 31/12/2020.	Jhan Carlos Sanchez	31/12/2020	Pendiente
9.	Actualizar, revisar y difundir el procedimiento de Proyectos y Obras Comerciales	Jhan Carlo Sanchez	Por Definir	Pendiente
10.	Manual de Usuario	Yorghinio Valenzuela Cecilia Cruzado	Por Definir	Pendiente

ElectroDunas	ACTA DE REUNIÓN	Versión 1
		GT-R-2-6-0-2

Información del Proyecto

Nombre del Proyecto:	**PORTAL DE PROYECTOS Y OBRAS DE CLIENTES – SEXTA PRUEBA CON USUARIOS**		
Código del Proyecto:		Acta NR:	
Preparado Por:		Fecha:	30/12/2020
Propósito:	Pruebas con Usuario de la G. Comercial y técnica		

Datos de Reunión

Tipo:	Coordinaciones para el Desarrollo				
Alcance:	PRUEBAS TRANSACCIONALES DE LOS EXPEDIENTES Y SOLICITUDES POR LA GERENCIA COMERCIAL Y GERENCIA TÉCNICA.				
Fecha de Reunión:	30/12/2020	Hora Inicio:	15:00	Hora Fin:	17:30
Lugar de Reunión:	Teams				
Coordinador Reunión:	Cecilia Cruzado				

Participantes

Nombre del Participante	Gerencia / Empresa	Nombre del Participante	Gerencia / Empresa
Jhan Carlos Sanchez (JCS)	Gerencia Comercial	**Cecilia Cruzado C (CCC).**	G. Tecnología Informática
Ruben Pacheco (RP)	Gerencia Comercial	**Yorghinio Valenzuela (YV)**	G. Tecnología Informática
Elizabeth Goicochea (EG)	Gerencia Comercial		
Eddy Matias (EM)	Gerencia Técnica		
Ida Echevarria (IE)	Gerencia Técnica		

Minuta de Reunión

Página Web:

- JCS, comentó que la versión final para la campaña y link del portal ha sido enviada por correo.
- CC, comentó que enviaría la constancia de pase a producción para la conformidad de JCS y luego para la autorización de la GTI.

Campaña Publicitaria del Portal:

- JCS, envió por correo que la difusión será de la siguiente manera: (i) envío de correo a todos los grandes clientes, (ii) impresión de texto enviado para que se difunda en los centros de atención al cliente y (iii) a través de Ruben Pacheco difundiendo vía mail el texto mencionado.
- RP, solicitó a JCS coordinar con el jefe de las Unidades Comerciales y con el personal de Grandes Clientes que a partir del 04.01.2021 entra a Producción el Portal de Proyectos y Obras y que todo tramite será por ese medio.

Pantalla Inicio del Portal →

- CC, comentó que el texto de datos personales de usuarios externos se ha realizado la consulta a la G. Legal y que a la fecha no han respondido.
- Cuando el usuario externo termina de ingresar sus datos para el registro al dar grabar debe llevarlo a la pantalla principal del LOGIN.
- Hay que asegurar que en las rutas que tiene el portal para saber en funcionalidad estamos al dar clic para retroceder validar que te lleve a la ruta correcta. Esto es muy amigable sin tener que ir al menú.

Expedientes:

- Hay que asegurar que al seleccionar el código del expediente relacionado solo se listen aquellos que han sido aprobados.
- Cambiar el label "Objetivo" por "Breve Descripción" y que no sea un campo obligatorio para grabar el expediente.

- El campo máxima demanda debe ser un campo obligatorio.
- En la grilla principal en la columna que dice: "Tramitado por" cambiarlo por "Solicitado por".

Representante:
- Que el campo teléfono fijo no sea obligatorio y colocar validaciones sino colocó información en los campos obligatorios (*).

Solicitudes:
- Hay que asegurar que el Usuario Externo pueda visualizar los adjuntos en el estado Aprobado y Rechazado de la Solicitud.
- En los adjuntos de la solicitud debe mostrar además del nombre del archivo, la fecha y hora que se anexo el archivo, así como el usuario que lo hizo.
- En la grilla principal en la columna que dice: "Tramitado por" cambiarlo por "Solicitado por".

Reportes:
- YV, comentó que lo reportes estarán listos para el 05.01.2021 y se avisará para las pruebas y revisión. Cabe mencionar que estos formatos de reportes recién se entregaron el 29.12.2020.

Manual y Video:
- Se realizará el video para tomar conocimiento de la funcionalidad del portal. Desde el registro de un usuario, la creación de un expediente y flujo completo de una solicitud. Esto se realizará en una próxima sesión con el apoyo de comercial y la gerencia técnica.
- JCS, solicitó que el manual de usuario lo realice TI.

Resumen de las Observaciones:

Funcionalidad	Detalle de los cambios
Inicio de Sesión	• Cuando el usuario externo termina de ingresar sus datos para el registro al dar grabar debe llevarlo a la pantalla principal del LOGIN. • Hay que asegurar que en las rutas que tiene el portal para saber en funcionalidad estamos al dar clic para retroceder validar que te lleve a la ruta correcta. Esto es muy amigable sin tener que ir al menú.
Tipo de Proyecto	Ninguno
Requisitos	Ninguno.
Expedientes	• Hay que asegurar que al seleccionar el código del expediente relacionado solo se listen aquellos que han sido aprobados. • Cambiar el label "Objetivo" por "Breve Descripción" y que no sea un campo obligatorio para grabar el expediente. • El campo máxima demanda debe ser un campo obligatorio. • En la grilla principal en la columna que dice: "Tramitado por" cambiarlo por "Solicitado por".
Representantes	• Que el campo teléfono fijo no sea obligatorio y colocar validaciones sino colocó información en los campos obligatorios (*).
Solicitudes	• Hay que asegurar que el Usuario Externo pueda visualizar los adjuntos en el estado Aprobado y Rechazado de la Solicitud. • En la grilla principal en la columna que dice: "Tramitado por" cambiarlo por "Solicitado por".
Adjuntos	• En los adjuntos de la solicitud debe mostrar además del nombre del archivo, la fecha y hora que se anexo el archivo, así como el usuario que lo hizo.
Reportes	• Confeccionar los dos reportes solicitados por la Gerencia Técnica.

Compromisos:

No.	Titulo / Descripción	Responsable	Fecha	Estado
1.	Envío de formatos de Reportes de la G. técnica para esta primera etapa.	Javier Cornejo	Etapa II	Pendiente

ElectroDunas	ACTA DE REUNIÓN	Versión 1
		GT-R-2-6-0-2

No.	Título / Descripción	Responsable	Fecha	Estado
2.	Últimos Cambios en el Portal luego de las Pruebas – Etapa I	Yorghinio Valenzuela	03/01/2021	Pendiente
3.	Video de la funcionalidad del Portal con apoyo de la G. Comercial y Técnica.	Jhan Carlos Sanchez	Próxima Semana	Pendiente
4.	Manual de Usuario	Yorghinio Valenzuela Cecilia Cruzado	Por Definir	Pendiente
5.	Constancia de Pase a Producción	Cecilia Cruzado Jhan Carlo Sanchez	30/12/2020	Terminado
6.	Coordinar con el jefe de las Unidades Comerciales y con el personal de Grandes Clientes que a partir del 04.01.2021 entra a Producción el Portal de Proyectos y Obras y que todo tramite será por ese medio.	Jhan Carlo Sanchez	Esta Semana	Pendiente
7.	Reportes solicitados por Ida Echevarria y Eddy Matias estarán listos para pruebas la próxima semana por que fueron solicitados el 29.01.2020	Yorghinio Valenzuela	05/01/2021	Pendiente

ANEXO VII

Constancia de Pase a Producción

ElectroDunas	CONSTANCIA DE PASE A PRODUCCIÓN	Versión 1
		GT-R-2-6-0-7

FECHA CONSTANCIA:	30/12/2020

La Gerencia Comercial deja constancia que la siguiente persona:

Nombres:	Jhan Carlo Sanchez
Área:	Proyectos y Obras Comerciales
Cargos:	Jefe de Proyectos y Obras Comerciales

Ha cumplido con realizar las pruebas en QA al Portal de Proyectos y Obras que su área es responsable, de acuerdo con el siguiente detalle:

Módulos	Fechas del Control de Calidad
• Registro Usuario Externo	14/12/2020
• Maestros	15/12/2020
• Transacciones de Expedientes y Solicitudes	17/12/2020
	18/12/2020
• Consultas	22/12/2020
• Reportes	29/12/2020
	30/12/2020

Las pruebas a cada uno de los módulos se han realizado en el ambiente de calidad http://192.168.40.113/ProyectoTerceros y los documentos evidencia de las pruebas se encuentran en la siguiente ruta: \\192.168.40.21\GTI\Desarrollo\02. Proyectos\35.Portal de Expedientes de Proyectos de los Clientes\3. Documentos del Proyecto\1.Formatos según Procedimiento GTI\Etapa 1

Por lo tanto, se otorga la conformidad, para que la Gerencia de Tecnología Informática, realice el pase a producción el Portal de Proyectos y Obras que se ha validado.

Eduardo Miranda Salas
Gerente de TI

Gustavo Michelena
Gerente Comercial

Jhan Carlo Sanchez
Jefe de Proyectos y Obras Comerciales

Cecilia Cruzado
Responsable del Proyecto

Yorghinio Valenzuela
Responsable del Pase a Producción