Instituto Tecnológico de Costa Rica

Manual Técnico para Sistema de Gestión de Entrada de Documentos.

Creado por:

Nelson Abarca Quirós 2013105521

Amanda Solano Astorga 2013100025

Yasiell Vallejos Gómez 2013094179

Versión 1.0

21/01/2017

Contenido

Presentación

Este documento fue creado por estudiantes del Tecnológico de Costa Rica como parte del curso de Proyecto de Ingeniería de Software en el periodo de verano 2016-2017. como finalización del proceso del desarrollo del Sistema de Gestión de Entrada de Documentos de la Municipalidad de Alajuelita.

El manual fue creado con el propósito de brindar la mayor información posible a futuros desarrolladores y personas con el suficiente nivel técnico para entender el funcionamiento interno del sistema, para ello se dará una descripción detallada de las funcionalidades del este.

El sistema de Gestión de Entrada de Documento se encuentra dividido en tres módulos principales: la página o módulo web, la aplicación o módulo móvil y el servicio web. Para dar brindar el mayor entendimiento posible del sistema cada uno de ellos será explicado de forma explícita y con el mayor detalle posible.

Resumen Ejecutivo

Este manual técnico fue creado con el fin de mostrar detalladamente el funcionamiento del Sistema para la Gestión de Entrada de Documentos. Esta dividido en cuatros partes: Aplicación Móvil, Servicio Web o web Service, Plataforma Web y Base de Datos, en todas ellas se encuentra definido las funciones, variables y elementos gráficos que componen el sistema y permiten su funcionamiento.

Objetivo

Describir de forma clara y concisa el funcionamiento del Sistema de Gestión de Entrada de Documentos para que pueda ser entendido de manera exitosa por cualquier persona que así lo precise.

Alcance

Este documento está dirigido al personal del departamento de Informática de la Municipalidad de Alajuelita, quienes serán los encargados de dar el mantenimiento futuro al sistema. Para comprender el sistema se necesita tener conocimiento de bases de datos, programación usando patrón de diseño MVC, funcionamiento de web services por llamadas de SOAP y desarrollo de páginas web con .NET. Además de conocimiento de lenguajes de programación como JAVA y C#.

Documentos de Referencia

Para comprender mejor el sistema se recomienda la lectura de los siguientes documentos:

- Documento de Requerimientos de Software
- Documento de Arquitectura de Software.

En dichos documentos se encuentra explicado ampliamente el funcionamiento del sistema desde una perspectiva más general.

Definiciones Importantes

- **Asincrónico :** Proceso no simultáneo.
- **Biblioteca:** Conjunto de implementaciones funcionales.
- **Clase :** Declaración o abstracción de un objeto.
- **EditText :** Elemento gráfico de Android utilizado para ingresar texto.
- **RelativeLayout:** Es un grupo de vista que muestra los elementos en posición relativa a otro elemento.
- **Spinner :** Conocidos en otros lenguajes de programación como dropdown o comboBox. Pemite seleccionar un dato de una lista.
- **SOAP :** Protocolo para realizar intercambio de información entre aplicaciones.
- **ScrollView :** Vista que permite que el contenido de la pantalla se ajuste mejor.
- **TextView :** Elemento gráfico de Android utilizado para mostrar texto no editable.
- **TableRow :** Fila de una tabla en Android.
- **Web Services :** Tecnológia utlizada para intercambiar datos entre aplicaciones.
- **XML :** Formato de archivo para almacenar texto.

Aplicación Móvil

En esta sección del documento se explicará el funcionamiento de la aplicación para dispositivos móviles Android. Para mejor entendimiento es necesario resaltar que en Android Nativo, toda "pantalla" está conectada por una clase en JAVA, a esta unión se le conoce como un activity.

Descripción de Interfaz Gráfica

En esta sección se explicará las pantallas de la aplicación y los elementos gráficos que las componen. Las pantallas están desarrolladas en archivo de extensión XML.

Todas ellas tienen como contenedor un ScrollView que contiene un RelativeLayout. Dentro del RelativeLayout están elementos como botones, tablas, visualizadores de texto(TextView) y campos para la edición de texto(EditText). Dos visualizadores de texto importante son

- txtWelcome : identificador con que es reconocido el campo donde se muestra el mensaje de bienvenida y el nombre de la persona que ingresó al sistema.
- txtStatus: identificador con que es reconocido el campo donde se muestra el estado de conexión con el servidor.

activity_enter_procedure.xml

Pantalla donde se encuentra el formulario que debe de llenar el Plataformista. Los siguientes son campos importantes de considerar, ya que son aquellos que el usuario utiliza para ingresar la información:

- txtDate: Campo que almacena la fecha. Tiene una función que será explicada más adelante que permite que cuando se toque el campo este despliegue un calendario.
- spinnerDepartment: Despliega una lista con todos los departamentos del sistema.
- spinnerType: Despliega una lista con los tipos de identificación que puede brindar el usuario.

- txtID: Campo que permite ingresar la identificación.
- txtName: Campo que permite ingresar el nombre del ciudadano
- txtContact: Campo que permite ingresar el teléfono o correo del ciudadano
- spinnerProcedure: Despliega una lista con los tipos de trámite que procesa la municipalidad
- txtDetail: Campo para ingresar el detalle.
- btnRegister: Botón que permite el envío de información a la parte lógica para su validación y envío a la base de datos.

activity_main.xml

Pantalla principal, es donde se encuentra el log in. Cuenta con dos campos para ingresar texto y un botón para enviar la información.

activity_menu.xml

Está compuesta por tres botones que re direccionan a otras pantallas.

- btnEnter: Llama al activity_enter_procedure
- btnSearch:Llama al activity_menu_search. Se desactiva si no existe conexión con el servidor.
- btnExit: Sale de la aplicación y regresa al activity_main.

activity_search_code.xml

La funcionalidad de esta pantalla es la de realizar la búsqueda por código. Cuenta con elementos visibles y otros que se muestran solo en ciertas ocasiones. Los elementos visibles son:

- txtCodeSearch : campo para ingresar el código del trámite a buscar
- btnSearchCode: botón que llamará a una función para realizar la búsqueda.

Dentro de los elementos invisibles tenemos:

- txtSearchResult: Etiqueta que se muestra cuando existen resultados de la búsqueda
- tableCode: Tabla donde se muestra el resultado de la búsqueda. Internamente está compuesta por 6 etiquetas que forman las columnas.
- txtError: Etiqueta que se despliega cuando no se encuentra registros en la búsqueda

activity_search_date.xml

La funcionalidad de esta pantalla es la de realizar la búsqueda por fecha. Cuenta con elementos visibles y otros que se muestran solo en ciertas ocasiones. Los elementos visibles son:

- txtFrom: Campo para ingresar la fecha de inicio de búsqueda. Al igual que en enter_procedure_activity, tiene una función que permite que se muestre un calendario al ser tocado
- txtTo: Campo para ingresar la fecha de finalización de búsqueda. Tiene la misma propiedad que el txtFrom
- btnSearchDate: botón que llamará a una función para realizar la búsqueda.

Dentro de los elementos invisibles tenemos:

- txtSearchResult: Etiqueta que se muestra cuando existen resultados de la búsqueda

- tableDate: Tabla donde se muestra el resultado de la búsqueda. Internamente está compuesta por 6 etiquetas que forman las columnas.
- txtError: Etiqueta que se despliega cuando no se encuentra registros en la búsqueda

activity_search_department.xml

La funcionalidad de esta pantalla es la de realizar la búsqueda por código. Cuenta con elementos visibles y otros que se muestran solo en ciertas ocasiones. Los elementos visibles son:

- spinnerDepartmentSearch : campo que despliega una lista con los departamentos existentes.
- btnSearchDepartment: botón que llamará a una función para realizar la búsqueda.

Dentro de los elementos invisibles tenemos:

- txtSearchResult: Etiqueta que se muestra cuando existen resultados de la búsqueda
- tableDepartment Tabla donde se muestra el resultado de la búsqueda. Internamente está compuesta por 6 etiquetas que forman las columnas.
- txtError: Etiqueta que se despliega cuando no se encuentra registros en la búsqueda

activity_search_menu.xml

Al igual que el activity_menu cuenta con botones que realizan distintas funciones:

- btnDate: Muestra el activity_search_date
- btnCode:Muestra el activity_search_department
- btnPlatformist: Muestra el activity_search_plaformer
- btnDepartment: Muestra el activity_search_department

activity_search_platformer.xml

La funcionalidad de esta pantalla es la de realizar la búsqueda por código. Cuenta con elementos visibles y otros que se muestran solo en ciertas ocasiones. Los elementos visibles son:

- spinnerPlatformerSearch : campo que despliega una lista con los plataformistas existentes.
- btnSearchPlatformer: botón que llamará a una función para realizar la búsqueda.

Dentro de los elementos invisibles tenemos:

- txtSearchResult: Etiqueta que se muestra cuando existen resultados de la búsqueda
- tablePlataformist Tabla donde se muestra el resultado de la búsqueda. Internamente está compuesta por 6 etiquetas que forman las columnas.
- txtError: Etiqueta que se despliega cuando no se encuentra registros en la búsqueda

activity_show_code

Esta pantalla cuenta con dos TextView y dos botones. Un TexView es el encargado de indicar al usuario que cual es su código si existe conexión, sin conexión a internet mostrará un mensaje informando que se almacenado el registro localmente y se guardará en el próximo inicio de sesión. Con respecto a los botones:

- btnGoBack: Regresa al activity_menu
- btnNew: Regresa al activity_enter_procedure.

Descripción de clases para GUI

Las clases que serán explicadas a continuación están asociadas a alguna pantalla explicada anteriormente. Para entender mejor el funcionamiento y no entrar en cuestiones repetitivas es importante mencionar los siguientes aspectos:

- OnCreate : Función que aparece en todas las clases. Es la función principal de las activity en Android. En ella se declaran todas las variables y llamadas a otras funciones.
- Los botones,spinner,textView y EditText tienen una propiedad llamada setOnClickListener que permite definir su comportamiento cuando son tocados.
- En Android todos los elementos gráficos deben ser declarados como variable en el código. Para ello se utiliza el siguiente código

  (ElementoGráfico) findViewbyId(R.id.iddelElemento)
- Alguna de estas clases tiene declaradas clases asincrónicas que serán explicadas más adelante.

enterProcedure

Clase asociada a activity_enter_procedure. La mayor parte de sus variables se encuentra asociados a los elementos gráficos del archivo XML. En la imagen 1 se detalla cada una de las variables:

String pattern : Variable utilizada para guardar el patrón de la identificación que seleccione el Plataformista.
dataBase dataBase: Objeto utilizado para llamar a la base de datos local que tiene la aplicación
int idPlataformist: Variable que almacena el identificador del Plataformista.
Boolean status: Variable utilizada para almacenar el estado de la conexión con el servidor.
SimpleDateFormat sdf: Variable utilizada para definir el formato de la fecha como día-mes-año

EditText date: Variable donde se ingresa la fecha
DatePickerDialog changeDate: Variable que muestra el calendario
Spinner department: Variable donde se despliegan los departamentos
Spinner spinnerProcedure: Variable donde se despliegan los tipos de trámites
Spinner spinnerType:Variable donde se despliega los tipos de identificación
EditText ID: Variable donde se ingresa la identificación de esta.
Button register: Variable que hace referencia la botón de registrar
EditText details: Variable donde se ingresa los detalles del trámite
EditText nameClient: Variable donde va el nombre del ciudadano
EditText contact: Variable donde se ingresa el correo o teléfono del ciudadano

Imagen 1

Con respecto a las funciones, la Tabla 1 explica cada una de ellas:

Función	Parámetros Recibidos	Descripción
`loadProcedureSpinner`		Función que llama a la base de datos local para obtener los tipos de trámites y asignárselos al SpinnerProcedure
`loadIdentifySpinner`		Función que llama a la base de datos local para obtener los tipos de identificación y asignárselos al spinnerType
`openDatePicker`		Función que crea y muestra el calendario

`reviewConnection`		Revisa el estado de la conexión y cambia el texto y el color del campo de texto que muestra la conexión
`changePattern`	• Entero con la posición del tipo de identificación	Cambia la variable pattern por el formato que corresponde y además cambia el mensaje de ejemplo de la variable ID
`registerDocument`		Verifica que todos los campos estén completos y el patrón de la cédula sea correcto. De ser así llama a la función registerwithConnection o registerwithoutConnection según corresponda.
`registerwithConnection`	• Fecha del registro • Identificador del departamento • Nombre del Ciudadano • Contacto del ciudadano • Identificador del tipo de trámite	Llama a una clase asincrónica para conectar con el servidor y enviar los datos.
`registerwithoutConnection`	• Fecha del registro • Identificador del departamento • Nombre del Ciudadano • Contacto del ciudadano Identificador del tipo de trámite	Mediante el objeto dataBase ingresa los datos dentro de la base de datos local, para que sean sincronizados la próxima vez que se ingrese al sistema

Tabla 1

MainActivity

Clase que se encarga de realizar la validación de credenciales para ingresar al sistema. Sus variables son explicadas en la imagen 2.

EditText emailText: Variable utilizada para obtener el correo ingresado
EditText passwordText: Variable utilizada para obtener la contraseña ingresada.
Button enter: Variable que hace referencia la botón de ingresar al sistema.
int idPlataformist: Variable para almacenar el id del Plataformista que va a ingresar

Imagen 2

Por otro lado, las funciones que posee se explican en la tabla 2:

Función	**Parámetros Recibidos**	**Descripción**
`loginwithConnection`		Función que llama a una clase asincrónica para conectar con el web services y comparar las credenciales. Si todo está correcto además de permitir el ingreso al sistema, actualizará los datos de la base de datos local y guardará los nuevos registros en el sistema.

`loginwithoutConnection`		Función que llama a la base de datos local para comprar las credenciales del usuario.
`storeCredentials`	• El nombre del Plataformista que está ingresando • El identificador del Plataformista que está ingresando	Función que almacena las credenciales como preferencias para ser utilizadas en las futuras clases.

Tabla 2

Menu

Esta clase está compuesta por tres variables que serán explicadas en la imagen 3. Estas corresponden a botones, por lo tanto, solo tiene una función asociadas que es *reviewConnection,* que funciona igual que el enterProcedure, además de bloquear el botón que dirige a las búsquedas.

Button EnterProcess: Variable que hace referencia al botón de ingresar nuevo documento
Button menuSearch: Variable que hace referencia al botón de búsqueda de oficios
Button exit: Variable que hace referencia al botón de salir.

Imagen 3

searchCode

Esta es una clase que es utilizada para realizar búsquedas. Al igual que las anteriores está compuesta en su mayoría por variables que corresponde a elementos gráficos, estos son explicados en la imagen 4.

EditText code: Variable que obtiene el código ingresado por el usuario
Button search: Variable que hace referencia al botón de buscar
TableLayout showInformation: Variable que hace referencia a la tabla que se creará para la búsqueda.
TextView errorMessage: Variable que hace referencia a la etiqueta que se muestra cuando no hay información
TextView headerMessage: Variable que muestra el texto que va sobre la tabla cuando hay resultados

Imagen 4

Por otro lado, las funciones que contiene, son explicadas a detalle en la tabla 3.

Función	**Salida**	**Descripción**
`reviewConnection`		Revisa la conexión con el servidor. Si existe muestra el mensaje de conectado, de no existir, llama a showErrorConnection
`createHeader`	Un objeto de tipo TableRow	Devuelve un objeto con el encabezado de la tabla
`goSearch`		Valida que los campos no estén vacíos y realiza la búsqueda llamando a una clase asincrónica. Si existen dato crea una tabla y si no muestra un mensaje informando al usuario de esto.

`showErrorConnection`		Muestra un cuadro de dialogo que indica que no existe conexión y la búsqueda no se puede ejecutar y redirección al activity_menu

Tabla 3

searchDate

Esta clase además de las variables que hacen referencias a los objetos gráficos, también cuenta con otros tipos de variables más especializadas. Están son explicadas en la imagen 5.

DatePickerDialog changeDate: Variable que abre el cuadro de dialogo que muestra la fechas.
SimpleDateFormat sdf: Variable que almacena el formato de la fecha
EditText txtFrom: Variable para obtener la fecha de inicio.
EditText txtTo: Variable para obtener la fecha de fin.
Button search: Variable que hace referencia al botón de buscar.
TableLayout showInformation: Variable que hace referencia a la tabla que se creará para la búsqueda.
TextView errorMessage: Variable que hace referencia a la etiqueta que se muestra cuando no hay información
TextView headerMessage: Variable que muestra el texto que va sobre la tabla cuando hay resultados

Imagen 5

Con respecto a las funciones, estás se pueden encontrar en la Tabla 3. Además de la función *openDatePicker,* explicada en la tabla 1.

searchDepartment

Está clase, al igual que searchCode está compuesta por elementos gráficos que serán explicados en la imagen 6. Para referirse a sus funciones se puede ver la Tabla 3 y la función *loadDepartmentSpinner* en la tabla 1.

Button search: Variable que hace referencia al botón de buscar.
TableLayout showInformation: Variable que hace referencia a la tabla que se creará para la búsqueda.
TextView errorMessage: Variable que hace referencia a la etiqueta que se muestra cuando no hay información
TextView headerMessage: Variable que muestra el texto que va sobre la tabla cuando hay resultados
Spinner department: Despliega todos los departamentos

Imagen 6

searchMenu

Esta clase está compuesta igual que la clase Menu. Por lo tanto, solamente contiene botones y no funciones importantes.

searchPlatformer

Está clase, al igual que searchCode está compuesta por elementos gráficos que serán explicados en la imagen 7.

Button search: Variable que hace referencia al botón de buscar.
TableLayout showInformation: Variable que hace referencia a la tabla que se creará para la búsqueda.
TextView errorMessage: Variable que hace referencia a la etiqueta que se muestra cuando no hay información
TextView headerMessage: Variable que muestra el texto que va sobre la tabla cuando hay resultados
Spinner plataformist: Despliega todos los plataformistas

Imagen 7

Para referirse a sus funciones se puede ver la Tabla 3 completa y la tabla 4 que muestra la función extra que posee.

Función	Descripción
`loadPlataformersSpinner`	Función que llama a la base de datos local para obtener el nombre de los plataformistas y asignárselos al Spinner plataformist.

Tabla 4

showCode

Clase cuyo propósito es mostrar el mensaje que informa del código de seguimiento al Plataformista o que el documento fue registrado localmente.

Descripción de clases auxiliares

Para el correcto funcionamiento de la aplicación se cuenta con una serie de clases que ayudan a implementar las funcionalidades. En la tabla 5 se muestra en resumen las clases que son usadas solamente para declarar objetos que son de ayuda para cargar los datos provenientes del web Services.

Clase	Descripción	Variables
Type	Clase utilizada para crear los objetos relacionados con los tipos de trámites y tipos de identificación	Id: Identificador del tipo de trámite o tipo de identificación Name: Nombre del tipo de trámite o tipo de identificación
SearchData	Clase utilizada en las búsquedas para almacenar los datos a mostrar	date: Fecha en que fue ingresado el trámite consecutive: el número consecutivo del trámite detail : Detalle ingresado por el plataformista identification: cédula, pasaporte del ciudadano state : Estado en que se encuentra el trámite type : Tipo de trámite que se realizó plataformer: Nombre del Plataformista que realizo el trámite
Credentials	Clase utiliza para ingresar la información a la tabla login y plataformist en la base de datos local.	idLoggin: Identificador asignado automáticamente cuando se registra un nuevo usuario email: Correo del empleado password: Contraseña del empleado plataformers: ID del plataformista name: Nombre del plataformista isBoss: Se indica si es jefe o no

Department	Clase utilizada para almacenar en objetos los datos de la tabla Departments de la base de datos	id: identificador de los departamentos. code: Código asignado al departamento. Department: Nombre del departamento

Tabla 5

Clase dataBase

Esta clase funciona como intermedia entre la aplicación y la base de datos local. Aquí es donde se crea y llena la base de datos con la información que le brinda el servidor. En la imagen 8 se muestran las variables que utiliza la clase.

Las funciones que son ejecutadas por esta clase son mostradas en la tabla 6.

Función	Descripción	Parámetros de Entrada	Salida
`createTable`	Crea todas las tablas de la base de datos		Booleano para validar que no existan errores en la creación de tablas
`syncBase`	Conecta con el web services para obtener todos los datos de la base de datos		
`exist`	Comprueba que el usuario que ingresa al sistema sea Plataformista	Email: Correo del Plataformista Password: Contraseña del plataformista	Entero con el identificador del plataformistas
`getDepartments`	Realiza una consulta a la base de datos local para traer el nombre de los departamentos		Lista de String con el nombre de los departamentos
`getProcedures`	Realiza una consulta a la base de datos local para traer el nombre de los tipos de trámites		Lista de String con los tipos de Trámites
`getDepartmentID`	Realiza una consulta a la base de datos para obtener el id del departamento buscado	Department: Nombre del departamento	Entero con el código del departamento
`getProcedureID`	Realiza una consulta a la base de datos para obtener el id del tipo de trámite buscado	Name: Nombre del tipo de trámite	Entero con el código del tipo de trámite
`getName`	Obtiene el nombre del Plataformista con el código de log in	ID: Entero que corresponde al identificador del Plataformista	String con el nombre del Plataformista
`getIdentify`	Realiza una consulta a la base de datos local para traer el nombre de los tipos de identificación		Lista de String con los tipos de identificación

`getPlaformers`	Realiza una consulta a la base de datos local para traer el nombre de los plataformistas		Lista de String con el nombre de los plataformistas
`encriptyon`	Encripta la clave del usuario para ser comparada con los datos de la base	Password: Contraseña de Plataformista	Regresa un String con la contraseña encriptada.
`insertProcedure`	Inserta en la base de datos local el trámite ingresado por el Plataformista	finalDate:Fecha de ingreso del trámite. departmentID:Identificador del departamento TypeIdentification: Identificador del tipo de identificación personID: Identificación de la persona name: nombre del ciudadano. Contact: Número o correo del ciudadano typeProcedure: Identificador del tipo de trámite. details: Observación que escribe el Plataformista idPlataformist: Identificador del plataformista que realiza el trámite	
`getProcedureSize`	Obtiene la cantidad de registros que posee la tabla de procedimientos		Regresa el total de registros que tiene la tabla de procedimientos
`syncProcedureTable`	Sincroniza la tabla local de procedimientos con el servidor		

Tabla 6

Descripción de clases asincrónicas

Las clases asincrónicas son usadas para enviar y recibir información el servicio web. Todas ellas tienen las mismas características: una serie de variables fijas que son mostradas en la tabla 7, variables que cambian según la clase que se ejecute y una función llamada doInBackground, que se ejecuta automáticamente al llamar el objeto con el método execute().

Variable	**Descripción**
METHOD_NAME	Nombre del método del web service que se va utilizar
NAMESPACE	Nombre del dominio del web service

SOAP_ACTION	Combinación del NAMESPACE y el METHOD_NAME
URL	Dirección con IP del web Service

Tabla 7

A continuación, se explicará cada clase con más detalle.

connection

Clase que se encarga de revisar la conexión entre la aplicación y el servidor. El método doInBackground regresa un booleano como resultado. No recibe ningún parámetro.

webServiceGetDepartment

Clase que conecta con el web services para obtener una lista con la información de los departamentos. Estos datos son almacenados en la tabla departments en la base de datos local de la aplicación. El método doInBackground regresa una lista con objetos del tipo department.

webServiceGetProcedures

Clase que conecta con el web services para obtener una lista con la información de los tipos de trámites. Estos datos son almacenados en la tabla TypeOfProcedures en la base de datos local de la aplicación. El método doInBackground regresa una lista con objetos del tipo type-

webServiceGetTypes

Clase que conecta con el web services para obtener una lista con la información de los tipos de identificación. Estos datos son almacenados en la tabla typesOfIdentify en la base de datos local de la aplicación. El método doInBackground regresa una lista con objetos del tipo type.

webServiceGetCredentials

Clase que conecta con el web services para obtener una lista con la información de log in de los plataformista. Además de esto también obtiene toda la información relacionada con los plataformistas. Estos datos son almacenados en la tabla loggin y plataformist en la base de datos local de la aplicación. El método doInBackground regresa una lista con objetos del tipo credentials.

webServiceInsertProcedure

Esta clase se utiliza para ingresar los procedimientos dentro de la base de datos. Recibe los mismos parámetros que la función insertProcedure, explicados en la tabla 6. Además, devuelve el código del trámite que se realizó.

webServiceLoggin

Como su nombre lo indica, es utilizada para comprobar las credenciales del usuario. Recibe como parámetro el correo y contraseña del usuario y regresa el código de log in en caso de que el proceso sea exitoso, de no serlo, regresa un 0.

webServiceSearchByCode

Esta clase es usada para consultar en la base de datos por código. Recibe como parámetro el código a buscar y regresa una tabla con los datos encontrados, estos datos son convertidos en una lista de tipo searchData para ser procesa después por la clase que lo contiene: searchCode

webServiceSearchByDepartment

Esta clase es usada para consultar en la base de datos por código. Recibe como parámetro el nombre del departamento a buscar y regresa una tabla con los datos encontrados, estos datos son convertidos en una lista de tipo searchData para ser procesa después por la clase que lo contiene: searchDepartment

webServiceSearchByDate

Esta clase es usada para consultar en la base de datos por código. Recibe como parámetros la fecha de inicio y la fecha de fin a buscar y regresa una tabla con los datos encontrados, estos datos son convertidos en una lista de tipo searchData para ser procesa después por la clase que lo contiene: searchDate

webServiceSearchByPlataformist

Esta clase es usada para consultar en la base de datos por código. Recibe como parámetro el nombre del plataformista a buscar y regresa una tabla con los datos encontrados, estos datos son convertidos en una lista de tipo searchData para ser procesa después por la clase que lo contiene: searchPlataformist

Bibliotecas

La aplicación utiliza la biblioteca KSOP 2 para realizar la conexión con el web services para recibir y enviar datos a la base de datos. El formato de salida que utiliza es XML. Para descargarla puede ingresar aquí.

Instalación y configuración del entorno

Para poder ejecutar el código de la aplicación es necesario contar con Android Studio, este puede ser descargado desde la página oficial de Android Studio.

Una vez descargado, debe dirigirse a la siguiente dirección de GitHub y descargar el proyecto. Para poder realizar modificaciones se debe de abrir en Archivo -> Open. Ahí se debe de seleccionar el archivo que aparecen en la imagen 8 y dar abrir.

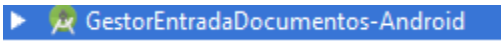

Imagen 8

Una vez abierto, hay que revisar si la biblioteca se encuentra agregada. Para saber esto hay que cambiar la visualización del proyecto a "*Proyecto",* una vez hecho esto, se revisa la carpeta libs, si existe y tiene el archivo de la biblioteca está listo para ejecutarse.

De no ser así, hay que mover el archivo a la carpeta y realizar el proceso que describe la imagen 9.

Web Services

El web services fue programado utilizando .NET y C#. Todas las funciones que realiza retornan un valor, ya sea una lista con información o un dato especifico de la base de datos. En esta sección se especificará con detalle el funcionamiento del web services y como ejecutarlo de manera que funcione en la red local.

Estructura del Web Services

El servicio web está compuesto por cuatro clases y un archivo ServicioClientes.asxm. Las clases están diseñadas para extraer la información de la base de datos en listas y enviarlas en archivos XML que son consumidas por la aplicación. Dichas clases son similares a las clases que son descritas en la tabla 5 en la sección de clases auxiliares de la aplicación, con la diferencia que la clase credentials en el web services se llama login y la clase searchData lleva el nombre de Searchs, por lo demás tienen los mismos parámetros y realizan las mismas funciones.

La explicación de las funciones que utiliza el archivo ServicioClientes.asxm serán explicadas más adelante en Descripción de funciones, en importante aclarar que todas ellas utilizan un String llamada connection donde están los parámetros para entrar en la base de datos, es importante cambiarlos cuando se cambie de servidor o las contraseñas de acceso.

Descripción de funciones

La mayoría de las funciones del web services son utilizadas para conectarse con la base de datos y extraer los datos para ser enviados a la aplicación, por lo tanto, están declaradas como Web Method. A continuación, se explicará cada una de ellas:

DepartmentList

Realiza una consulta a la base de datos para obtener los datos de los departamentos registrados, una vez hecho esto, devuelve una lista con el tipo de datos department que es utilizada en la aplicación para llenar la tabla departament.

LoginList

La aplicación permite ingresar sin conexión con el servidor, esta función le brinda los insumos para realizar dicha tarea. La contraseña que recibe del usuario se decodifica para luego ser nuevamente codificada con un método de encriptación compatible entre Android y C#, al finalizar este proceso regresa una lista con un tipo de dato login.

TypeIdentifyList, TypeProcedureList

Al igual que al departmentList estos dos métodos regresan una lista de objetos type, con la información referente a Tipo de Identificación y Tipos de Trámites.

insertProcedure

Este método, como su nombre lo indica es utilizado para insertar los nuevos documentos a la base de datos. La cantidad y tipo de parámetro que recibe es igual que los de la función insertProcedure que se detallado ampliamente en la tabla 6, la única diferencia es que este método retorna el código de seguimiento del trámite.

loginVerification

Se utiliza para verificar que las credenciales del usuario coinciden y que es un plataformista. Cuando esto ocurre regresa el id del plataformista para ser utilizado en la aplicación, de no ser así regresa un 0.

Métodos de Búsqueda

En la tabla 8, se indica los parámetros que reciben estos métodos y la descripción de ellos. Todos ellos regresan una lista de objetos tipo Searchs

Método	Descripción	Parámetros
searchByCode	Búsqueda en la base de datos por el código ingresado	Un String con el código
searchByDate	Búsqueda en la base de datos en un rango de fechas	Dos Strings, uno con la fecha de inicio y el otro con la fecha de fin
searchByDepartment	Búsqueda en la base de datos por el nombre del departamento ingresado	Un String con el nombre del departamento
searchByPlataformer	Búsqueda en la base de datos por el nombre del plataformista ingresado	Un String con el nombre del plataformista

Tabla 8

Configuración de entorno

Para poder ejecutar el web services de manera exitosa y que la aplicación se conecte con él. Se debe de realizar los siguientes pasos:

1. Instalar Visual Studio 2012.
2. Ejecutar como administrador
3. Dirigirse a ServicioCliente.asxm y modificar los parámetros de conexión con la base.
4. Ir a Compilar ->Publicar y seguir los siguientes pasos:
    a. Seleccionar un perfil, si no existe se debe de crear uno en "*Agregar nuevo perfil*"
    b. En conexión poner los datos como se indica en la imagen 9

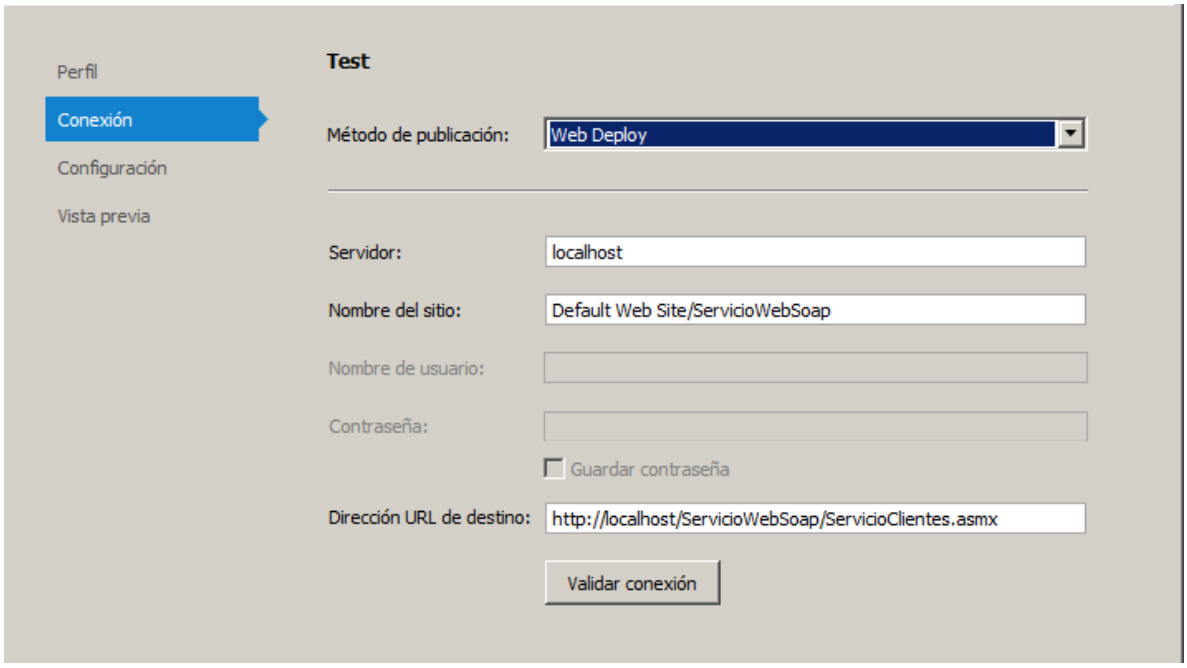

Imagen 9

    c. Poner la configuración en releáse
    d. En vista previa se presiona "*Comenzar Visualización*".
    e. Al finalizar de ejecutarse el paso 4 se debe de dar Publicar y listo ya el web service está listo para ser usado.

Página Web

Modelo

En esta sección se explicará el único modelo creada en la plataforma web, en la cual se establece la conexión con la base de datos y cada una de las funciones utilizadas para interactuar con ella. En la siguiente tabla se detallan las funciones implementadas en este modelo y su funcionalidad.

Función	Descripción	Parámetros de Entrada	Salida
`getDepartments`	Conecta con la base de datos para obtener un dataset con los departamentos registrados en la base de datos		Dataset con la lista de departamentos registrados
`getProcedureType`	Conecta con la base de datos para obtener un dataset con los tipos de trámites registrados en la base de datos		DataSet con la lista de tipos de trámites registrados
`getIdentifyType`	Conecta con la base de datos para obtener un dataset con los tipos de identificación registrados en la base de datos		Dataset con la lista de tipos de identificación registrados
`insertProcedure`	Conecta con la base de datos y registra el ingreso de un nuevo trámite	DateTime date: fecha de ingreso del documento int departmentId: indentificador del departamento seleccionado int idTypeOfIdentify: identificador del tipo de identificación seleccionado String personID: número de identificación del solicitante String personName: nombre del solicitante String personContact: correo electrónico o número telefónico del solicitante	String que corresponde al consecutivo asignado al trámite registrado

		int idTypeOfProcedure: identificador del tipo de trámite seleccionado String detail: descripción del trámite a registrar int userId: identificar del plataformista que ingresa el trámite	
`getProcedure`	Conecta con la base de datos y obtiene un trámite registrado en la base de datos	String procedureCode: consecutivo asignado a un trámite	Dataset que contiene la información registrada del trámite seleccionado
`updateProcedure`	Conecta con la base de datos y actualiza un trámite anteriormente registrado	String code: consecutivo asignado a un trámite int idTypeOfIdentify: identificador del tipo de identificación seleccionado String personID: número de identificación del solicitante String personName: nombre del solicitante String personContact: correo electrónico o número telefónico del solicitante int idTypeOfProcedure: identificador del tipo de trámite seleccionado String detail: descripción del trámite a registrar	
`getPlatformers`	Conecta con la base de datos para obtener un dataset con los plataformistas registrados en la base de datos		DataSet con la lista de plataformistas registrados

`getSearchDep`	Conecta con la base de datos para obtener un dataset con los trámites asignados al departamento que se desea buscar	String “dep” representa el nombre del Departamento al cual los trámites deben pertenecer en la busqueda	DataSet con la tabla de resultados de trámites para ese departamento
`getSearchPlat`	Conecta con la base de datos para obtener un dataset con los trámites asignados al Plataformista que se desea buscar	String “plat” representa el nombre del Plataformista al cual los trámites deben pertenecer en la busqueda	DataSet con la tabla de resultados de trámites para ese plataformista
`getSearchDate`	Conecta con la base de datos para obtener un dataset con los trámites asignados al rango de fechas que se desea buscar	DateTime “from” representa la fecha de inicio del rango de busqueda, DateTime “to” representa el fin del rangp	DataSet con la tabla de resultados de trámites para ese rango de fechas
`getSearchCode`	Conecta con la base de datos para obtener un dataset con el trámite asignado al codigo	String “code” representa el código consecutivo a buscar	DataSet con el resultado del trámite para ese código
`getBinnacle`	Conecta con la base de datos y obtiene una lista completa de los trámites registrados en un periodo establecido	DateTime from: fecha de inicio en la que se obtendrán los trámites DateTime to: fecha final en la que se obtendrán los trámites	Dataset con la lista de trámites registrados en el periodo establecido
`getIDLogin`	Conecta con la base de datos para obtener un int con el ID de la tabla dbo.loggin para un correo y contraseña específico	string “email” representa el email del usuario, string “pwd” respresenta la contraseña del usuario	Un Int con el ID de la tabla dbo.loggin pare ese email y contraseña, retorna -1 en caso de que no haya conexión o no exista el ID

`getIDPlatformer`	Conecta con la base de datos para obtener un int con el ID de la tabla dbo.Plataformers para un ID de login específico	int “idLog” representa el ID del login al que pertenece el plataformista	Un Int con el ID de la tabla dbo.Plataformers, retorna -1 en caso de que no haya conexión o no exista el ID en la tabla
`getNamePlatformer`	Conecta con la base de datos para obtener un String con el nombre del plataformista de la tabla dbo.Plataformers para un ID de plataformista específico	int “platformertId” representa el ID del plataformista	Un String con el nombre del plataformista al que pertenece el ID
`IsABoss`	Conecta con la base de datos para obtener un int para saber si ese plataformista es jefe o no	int “idPlat” representa el ID del plataformista que deseamos saber si es jefe	Un Int que en caso de ser 0 no es jefe de plataforma y de ser 1 es jefe de plataforma
`getIDSecretary`	Conecta con la base de datos para obtener un int con el ID de la tabla dbo.Secretary para un ID de login específico	int “idLog” representa el ID del login al que pertenece el secretaRIO	Un Int con el ID de la tabla dbo.Secretary retorna -1 en caso de que no haya conexión o no exista el ID en la tabla
`getNameSecretary`	Conecta con la base de datos para obtener un String con el nombre del secretario de la tabla dbo.Secretary para un ID de secretaria específico	int “secretarytId” representa el ID delo la secretaria	Un String con el nombre del o la secretaria
`getViewUsers`	Conecta con la base de datos para obtener un dataset con los usuarios del sistema		DataSet con la tabla donde se muestran todos los usuarios del sistema
`existEmail`	Conecta con la base de datos para obtener un int que nos dice en caso de ser 1 que el email, ya está registrado o 0 que aún no	String “email” representa el correo electrónico a saber si ya está registrado o no	Un int que nos dice en caso de ser 1 que el email, ya está registrado o 0

	se ha registrado en la tabla dbo.loggin		que aún no se ha registrado
`insertLoggin`	Conecta con la base de datos para registrar en la tabla dbo.logging el correo y contraseña dada	String “email” representa el email a guardar, String “pwd” representa la contraseña a guardar	Un Int que en caso de ser 0 significa que la operación fue un éxito
`searchInLogging`	Conecta con la base de datos para obtener un int con el ID de la tabla dbo.logging para un correo sin necesidad la contraseña	string “email” representa el email del usuario	Un int con el ID de la tabla dbo.logging
`insertPlatfor`	Conecta con la base de datos para registrar en la tabla dbo. Plataformers al plataformista	String “email” representa el email del usuario, String “pwd” representa la contraseña del usuario, String “name” representa el nombre completo del usuario, int boss representa si es jefe de plataforma o no	Un Int que en caso de ser 0 significa que la operación fue un éxito
`insertSecre`	Conecta con la base de datos para registrar en la tabla dbo.Secretary al secretario	String “email” representa el email del usuario, String “pwd” representa la contraseña del usuario, String “name” representa el nombre completo del usuario, String “dep” representa el nombre del departamento al que se le asignara	Un Int que en caso de ser 0 significa que la operación fue un éxito
`test`	Conecta con la base de datos para obtener un dataset con los trámites nuevos para un departamento en específico	int “departmentId” representa el ID del departamento del cual queremos los trámites nuevos	DataSet con el resultado del trámite para ese departamento
`getIDDepartmentBySec`	Conecta con la base de datos para obtener un int con el ID del departamento al que pertenece una secretaria	int “id” representa el ID de la tabla dbo.Secretary de la secretaria que	Un int con el ID del departamento al que

		deseamos saber el departamento	pertenece una secretaria
`getNameDep`	Conecta con la base de datos para obtener un String con el nombre del departamento al que pertenece el ID	int “departmentId” representa el ID del cual se desea saber el nombre del departamento	Un String con el nombre del departamento
`getViewProcType`	Conecta con la base de datos para obtener un dataset con los tipos de trámites registrados		DataSet con la tabla donde se muestran todos los tipos de trámites registrados
`TransferProcedure`	Conecta con la base de datos para transferir los trámites de un departamento a otro y registrarlo en una tabla	int “idProc”representa el ID del trámite en la table dbo.procedure, Int “recive” reprecenta el ID del departamento que recibe el document ,int “send” representa el ID del o la secretaria que envía el trámite ,String “code” representa el consecutivo que posee el trámite ,String”justi” representa la justificación que da el usuario del porque transfiere el trámite	Un Int que en caso de ser 0 significa que la operación fue un éxito
`init`	Conecta con la base de datos para inicializar un trámite esto es que pasé de nuevo a En proceso	String “code” representa el consecutivo del trámite al que se desea inicializar	Un Int que en caso de ser 0 significa que la operación fue un éxito
`eliminateUser`	Conecta con la base de datos para poner los campos vacíos en la tabla dbo.logging en donde se desea eliminar el usuario	String “email” representa el correo de usuario que buscaremos dejar en vacio	Un Int que en caso de ser 0 significa que la operación fue un éxito

Vistas

En esta sección se explicarán las vistas creadas en la plataforma web, las cuales serán explicadas elemento por elemento y cada una de las funciones utilizadas para interactuar con ella. A

continuación, se detalla cada vista implementada las cuales se agruparán según el controlador al cual pertenecen para un mayor control y entendimiento.

Procedure

Create.cshtml

La funcionalidad de esta pantalla es ingresar un nuevo documento en el sistema, se conforma de un elemento principal que es el Form en el cual se encuentran todos los campos requeridos para el registro. A continuación, se detalla cada uno de los elementos que conforman el Form.

- procedureDate: elemento HTML INPUT de tipo date en el cual tiene como valor predeterminado la fecha actual para el ingreso del documento
- department: elemento HTML SELECT en el cual se despliegan todos los departamentos registrados en la base de datos
- idType: elemento HTML SELECT en el cual se despliegan los tipos de identificación que se encuentran registrados en la base de datos, los cuales son: Pasaporte, Nacional y Extranjero
- personID: elemento HTML INPUT en el cual se ingresa la cédula de la persona solicitante del trámite, este elemento cambia su formato a partir del elemento seleccionado en el idType
- personName: elemento HTML INPUT en el cual se ingresa el nombre de la persona solicitante del trámite
- personContact: elemento HTML INPUT en el cual se ingresa ya sea el correo electrónico o número telefónico de la persona solicitante del trámite
- procedureType: elemento HTML SELECT en el cual se despliegan los tipos de trámites registrados en la base de datos para la asignación del trámite

- procedureDetail: elemento HTML TEXTAREA en el cual se ingresa una descripción breve del trámite, el cual tiene un límite de 300 palabras

Además de los campos mencionados anteriormente se cuenta con dos botones cuyas funcionalidades son:

- saveRegister: elemento HTML BUTTON que se encarga de enviar el form a la función addProcedure cuando todos sus campos han sido ingresados correctamente.
- Regresar: elemento HTML BUTTON que se encarga de regresar al usuario al menú correspondiente.

Esta página cuenta con las siguientes funciones JAVASCRIPT para su funcionalidad

Función	Descripción	Parámetros de Entrada	Salida
`$('#idType').change`	Se encarga de obtener el formato del elemento personID a partir del cambio en el idType	Entero haciendo referencia al id del tipo de identificación	String de formato y de placeholder para el elemento personID

AddProcedure.cshtml

La funcionalidad de esta página es mostrarle al usuario el consecutivo asignado al trámite que acaba de ingresar. Se conforma de los siguientes elementos:

- HTML H3 que contiene el mensaje enviado desde el controlador con el consecutivo asignado al trámite en una variable ViewBag.code
- HTML BUTTON “Finalizar” se encarga de enviar al usuario al menú principal correspondiente.
- HTML BUTTON “Agregar Otro Trámite” se encarga de enviar al usuario a la página de ingresar nuevo trámite para que pueda realizar algún otro registro.

Edit.cshtml

La funcionalidad de esta página es permitirle al jefe de plataforma la posibilidad de editar un trámite que ha sido ingresado al sistema. Esta página se conforma de dos elementos importantes de los cuales uno se encarga del ingreso del código para la búsqueda del trámite y un Form que despliega los datos asignados al trámite. Al inicio el Form se encuentra oculto a la vista del usuario y se despliega una vez ingresado un código correcto.

El form contiene los siguientes campos editables una vez que se obtiene los datos correspondientes al trámite obtenido.

- idType: elemento HTML SELECT en el cual se despliegan los tipos de identificación que se encuentran registrados en la base de datos, los cuales son: Pasaporte, Nacional y Extranjero
- personID: elemento HTML INPUT en el cual se ingresa la cédula de la persona solicitante del trámite, este elemento cambia su formato a partir del elemento seleccionado en el idType
- personName: elemento HTML INPUT en el cual se ingresa el nombre de la persona solicitante del trámite
- personContact: elemento HTML INPUT en el cual se ingresa ya sea el correo electrónico o número telefónico de la persona solicitante del trámite
- procedureType: elemento HTML SELECT en el cual se despliegan los tipos de trámites registrados en la base de datos para la asignación del trámite

- procedureDetail: elemento HTML TEXTAREA en el cual se ingresa una descripción breve del trámite, el cual tiene un límite de 300 palabras

Esta página cuenta con las siguientes funciones JAVASCRIPT para su funcionalidad

Función	Descripción	Parámetros de Entrada	Salida
`$('#getProcedureData').click`	Se encarga de obtener el trámite registrado en la base de datos a partir del código ingresado mediante la utilización de una función AJAX	String que hace referencia al código del trámite	Muestra al usuario el Form con todos los datos del trámite encontrado

`$('#idType').change`	Se encarga de obtener el formato del elemento personID a partir del cambio en el idType	Entero haciendo referencia al id del tipo de identificación	String de formato y de placeholder para el elemento personID

saveChangedProcedure.cshtml

La funcionalidad de esta página es mostrarle al usuario un mensaje en el cual se indica que sus cambios fueron guardados exitosamente. Se conforma de los siguientes elementos:

- HTML H3 que contiene el mensaje enviado desde el controlador con el consecutivo del trámite en una variable ViewBag.code para indicar el cambio
- HTML BUTTON “Finalizar” se encarga de enviar al usuario al menú principal correspondiente.
- HTML BUTTON “Editar Otro Trámite” se encarga de enviar al usuario a la página de editar trámite para que pueda realizar algún otro cambio en un trámite ya registrado.

SearchProcedure.cshtml

La funcionalidad de esta página es permitirle al plataformista buscar trámites por rango de fechas, plataformista, departamento y código de consecutivo. Esta página está conformada por

- HTML SELECT: por medio de un ViewBag.deparmentTable se muestra todos los departamentos extraídos de la tabla para poder buscar por departamento
- HTML SELECT: por medio de un ViewBag.platformersTables muestra todos los plataformistas extraídos de la tabla para poder buscar por plataformista
- HTML INPUT: por medio de un type date se puede seleccionar el tipo de fecha de inicio para la búsqueda por rango de fechas
- HTML INPUT: por medio de un type date se puede seleccionar el tipo de fecha final para la búsqueda por rango de fechas
- HTML INPUT: por medio de un type text se puede escribir el código del consecutivo que se desea buscar

También existe una ventana emergente que se mostrara en caso de cualquier error

Función	Descripción	Parámetros de Entrada	Salida
$('#department').change()	Se encarga de enviarle al controlador el departamento que se ha seleccionado en el Select de departamento	Un String con el nombre del departamento	Recarga la página agregando una vista parcial con el resultado de la búsqueda
$('#platformist').change(	Se encarga de enviarle al controlador el departamento que se ha seleccionado en el	Un String con el nombre del plataformista	Recarga la página agregando una vista parcial con el

	Select de departamento		resultado de la búsqueda
$('#byDate').click()	Se encarga de enviarle al controlador el rango de fechas por el que se desea hacer la búsqueda	Dos datos de tipo date que entran en el formato “yyyy-MM-dd'” para los rango de fechas	Recarga la página agregando una vista parcial con el resultado de la búsqueda
$('#byCode').click()	Se encarga de enviarle al controlador el código del consecutivo por el que se desea hacer la búsqueda	Un string con el código del consecutivo	Recarga la página agregando una vista parcial con el resultado de la búsqueda
$('#btnShowDate').click()	Se encarga de mostrar los campos ocultos para cargar las fechas		Mostrar los campos ocultos para cargar las fechas
$('#btnShowCode').click()	Se encarga de mostrar los campos ocultos para escribir el código		Mostrar los campos ocultos para escribir el código

searchDepartment.cshtml

La funcionalidad de esta página es ser una página parcial donde se carga el resultado de la búsqueda por Departamento. Su parte más importante es el HTML SELECT donde se carga un “ViewBag.searchTable” que es donde se encuentra el resultado de la búsqueda

searchPlatformist.cshtml

La funcionalidad de esta página es ser una página parcial donde se carga el resultado de la búsqueda por Plataformista. Su parte más importante es el HTML SELECT donde se carga un “ViewBag.searchTable” que es donde se encuentra el resultado de la búsqueda

searchByDate.cshtml

La funcionalidad de esta página es ser una página parcial donde se carga el resultado de la búsqueda por rango de fecha. Su parte más importante es el HTML SELECT donde se carga un “ViewBag.searchTable” que es donde se encuentra el resultado de la búsqueda

searchByCode.cshtml

La funcionalidad de esta página es ser una página parcial donde se carga el resultado de la búsqueda por código. Su parte más importante es el HTML SELECT donde se carga un “ViewBag.searchTable” que es donde se encuentra el resultado de la búsqueda.

dailyProcedure.cshtml

La funcionalidad de esta página es mostrar todos los trámites nuevos para el departamento y además ofrece la opción de transferirlos o marcarlos como recibidos. Esta página cuenta con varias partes importantes como:

- HTML SELECT: por medio de un ViewBag.dailyProcedure se carga la tabla con todos los procedimientos nuevos
- HTML INPUT: con un type checkbox se encarga de que el usuario pueda marcar cuales trámites recibieron físicamente
- HTML BUTTON: “Transferir” se encarga de transferirnos a otra página en donde el usuario pueda realizar la transferencia
- HTML BUTTON: “Guardar” se encarga de inicializar los trámites que el usuario marco en los checkbox como recibido.
- También existe una ventana emergente que se mostrara en caso de cualquier error

Función	Descripción	Parámetros de Entrada	Salida
function transferProcedure	Se encarga de enviarle al controlador de la página de transferir documentos, los datos del consecutivo y el ID del procedimiento	Un String que es el código del consecutivo, un int que es el ID del procedimiento	Recarga la página de Transferir Procedimiento
function init()	Se encarga de captar todos los checkbox ue fueron marcados para enviarlos al controlador y este los inicialice	Todos los checkbox marcados	Refresca la página actual pero sin mostrar los trámites que se inicializaron

transferProcedure.cshtml

La funcionalidad de esta página es mostrar un formulario que el secretario debe llenar para transferir el documento a otro departamento. La parte principal de esta página es el HTMLForm que cuenta con un HTML INPUT tipo date para la fecha en que se realiza la transferencia, otro tipo text para el consecutivo y otro tipo number que se encuentra escondido donde se carga el Id, un HTML SELECT donde se carga todos los departamentos a donde se podría enviar el trámite, un HTML TEXT AREA para escribir la justificación y el HTML BUTTON transferir que se encarga de enviar los datos.

dailyProcedure.cshtml

La funcionalidad de esta página es mostrar todos los trámites nuevos para el departamento y además ofrece la opción de transferirlos o marcarlos como recibidos. Esta página cuenta con varias partes importantes como:

LogIn

Index.cshtml

Esta es la página de inicio del programa donde el usuario se logea para ingresar al sistema. La parte principal de la página está formada por un HTML Form que cuenta con dos HTML INPUT de tipo texto, uno para poder ingresar el correo electrónico y otro para poder ingresar la contraseña; también cuenta con un HTML BUTTON para enviar estos datos al controlador y validarlos

Administrator

createUser.cshtml

La funcionalidad de esta página es mostrar un formulario para poder ingresar un nuevo usuario. Las principales partes de este son:

- userName: HTML INPUT de tipo texto para escribir el nombre completo
- Un HTML SELECT para seleccionar el departamento al que va a pertenecer
- userIsBoss: HTML INPUT tipo checkbox para el caso que sea un departamento sea Plataforma, pueda escoger si es jefe de plataforma o no.
- userEmail: HTML INPUT tipo texto donde va el correo del usuario
- userPassword: HTML INPUT tipo texto para la contraseña

Además, posee un un @ViewBag.messege en caso de que sea necesario mostrar algún mensaje de error

procedureTypes.cshtml

La funcionalidad de esta página es mostrar una tabla tipo HTML TABLE con todos los tipos de trámites registrados hasta el momento además de poseer un HTML INPUT para agregar nuevos tipos de trámites cuando sean necesarios.

UserList.cshtml

La funcionalidad de esta página es mostrar una tabla HTML TABLE con todos los usuarios registrados hasta el momento además de poseer un HTML BUTTON que nos traslada a la página de crear usuarios, y un HTML ICON para eliminar usuarios y una ventana emergente de confirmación

Función	Descripción	Parámetros de Entrada	Salida
$('#table').on('click', '#remove',)	Se encarga de enviarle al controlador de la página los		Recarga la página sin

	datos del usuario que se desea eliminar además de mostrar una ventana de confirmación		mostrar al usuario en la lista de usuarios

Controladores

En esta sección se explicarán cada uno de los controladores creados en la plataforma web, en la cual se establece la conexión entre la vista y el modelo. En las siguientes tablas se detallarán cada una de las funciones implementadas en cada controlador y su funcionalidad.

AdministratorController.cs

Función	Descripción	Parámetros de Entrada	Salida
UserList	Esta función se encarga de cargar del modelo la lista de usuarios y enviarla a la vista de UserList		ActionResult hacia la vista de UserList con la lista de usuarios
CreateUser	Esta función se encarga de cargar del modelo la lista de departamentos y enviarla a la vista de CretaUser		ActionResult hacia la vista de CreateUser con la lista de departamento
CreateUser	Esta función recibe un FormCollection "form" que trae los datos del formulario que lleno el usuaro en la vista CreateUser y envía esos datos al modelo para crear el usuario, al final retorna al UserList	FormCollection "form" que trae los datos del formulario que lleno el usuaro en la vista CreateUser	ActionResult hacia el UserList que mostraría en la lista al nuevo usuario
procedureTypes	Esta función se encarga de cargar del modelo la lista de tipos de trámites y enviarla a la vista de procedureTypes		ActionResult hacia la vista de procedureTypes con la lista de tipos de trámites
procedureTypes	Esta función recibe un FormCollection "form" que trae los datos del formulario que lleno el usuaro en la vista procedureTypes y envía esos datos al modelo para crear el tipo de trámite	FormCollection "form" que trae los datos del formulario que lleno el usuaro en la vista procedureTypes	ActionResult hacia la vista de procedureTypes con la lista de tipos de trámites

LogInController.cs

Función	Descripción	Parámetros de Entrada	Salida
Index	Limpia todas las sesiones iniciadas		ActionResult hacia la vista de Index
Index	Esta función recibe un FormCollection "form" que trae los datos del formulario que lleno el usuaro en la vista Index para iniciar Sesión. Los datos se envían al modelo para saber si la sesión es válida o no y cuales variables "Session" se deben inicializar	FormCollection "form" que trae los datos del formulario que lleno el usuaro en la vista Index para iniciar sesión	ActionResult hacia el menú correspondiente a la sesión que se desea iniciar o un ActionResult hacia la vista Index en caso de error.

ProcedureController.cs

Función	Descripción	Parámetros de Entrada	Salida
searchProcedure	Esta función se encarga de cargar la lista de departamentos y la lista de plataformistas para ser cargada en la vista de searchProcedure		ActionResult hacia la vista searchProcedure
searchDepartment	Esta función se encarga de recibir el nombre de un departamento enviarlo al modelo y devolver por medio de un ViewBag a la vista searchDepartment los resultados de la busqueda	String dep que representa el nombre del departamento	ActionResult hacia la vista parcial de searchDepartment con los resultados de la busqueda
searchPlarformist	Esta función se encarga de recibir el nombre de un plataformista, enviarlo al modelo y devolver por medio de un ViewBag a la vista searchPlatformist los resultados de la busqueda	String plat representa el nombre de un plataformista	ActionResult hacia la vista parcial de searchPlatformist con los resultados de la busqueda
searchByDate	Esta función se encarga de recibir un rango de fechas, enviarlo al modelo y devolver por medio de un ViewBag a la vista	DateTime from representa la fecha de inicop del rango, DateTime to	ActionResult hacia la vista parcial de searchByDate con los resultados de la busqueda

	searchByDate los resultados de la busqueda	representa la fecha del fin del rango	
searchByCode	Esta función se encarga de recibir un código, enviarlo al modelo y devolver por medio de un ViewBag a la vista searchByCode los resultados de la busqueda	String code representa el código con el que se ha de buscar	ActionResult hacia la vista parcial de searchByCode con los resultados de la busqueda
dailyProcedure	Esta función carga a del modelo a un viewbag todos los trámites por departamento que estén en estado de "Nuevo", utilizando la variable de sesión "idDepartment"		ActionResult hacia la vista de dailyProcedure con la lista de todos los trámites nuevos del departamento
dailyProcedure	Esta función se encarga de inicializar o pasar el trámite de Nuevo a En proceso, recibiendo el código del trámite	String code representa el código consecutivo del procedimiento a inicializar	ActionResult hacia la vista de dailyProcedure con la lista de todos los trámites nuevos, eliminando los inicializados del departamento
transferProcedure	Esta función se encarga de recibir los datos del trámite que se va a transferir y mostrarlos en la vista del transferProcedure	String codeProcedure representa el consecutivo que tiene el trámite, int idProcedure representa el ID que tiene el trámite	ActionResult hacia la vista transferProcedure.
saveTransferProcedure	Esta función recibe un FormCollection donde están los datos necesarios para la transferencia del trámite, la función se encarga de mandar los al modelo para que pueda ser transferido	FormCollection form trae los datos necesarios para la transferencia del trámite	ActionResult hacia transferProcedure con la lista actualizada de los datos que pertenecen a ese departamento

Configuración de entorno

Referirse a configuración de entorno de la plataforma web.

Base de Datos

En esta sección del documento se describe la estructura de la base de datos y los pasos a seguir para implementarla en el sistema. Es importante recargar que se considera que ya existe un servidor de SQL Server 2008 instalado.

Modelo de Base de Datos

En la imagen 10 se puede visualizar el esquema de la base de datos que utiliza el Sistema de Gestión de Entrada de Documentos para su funcionamiento.

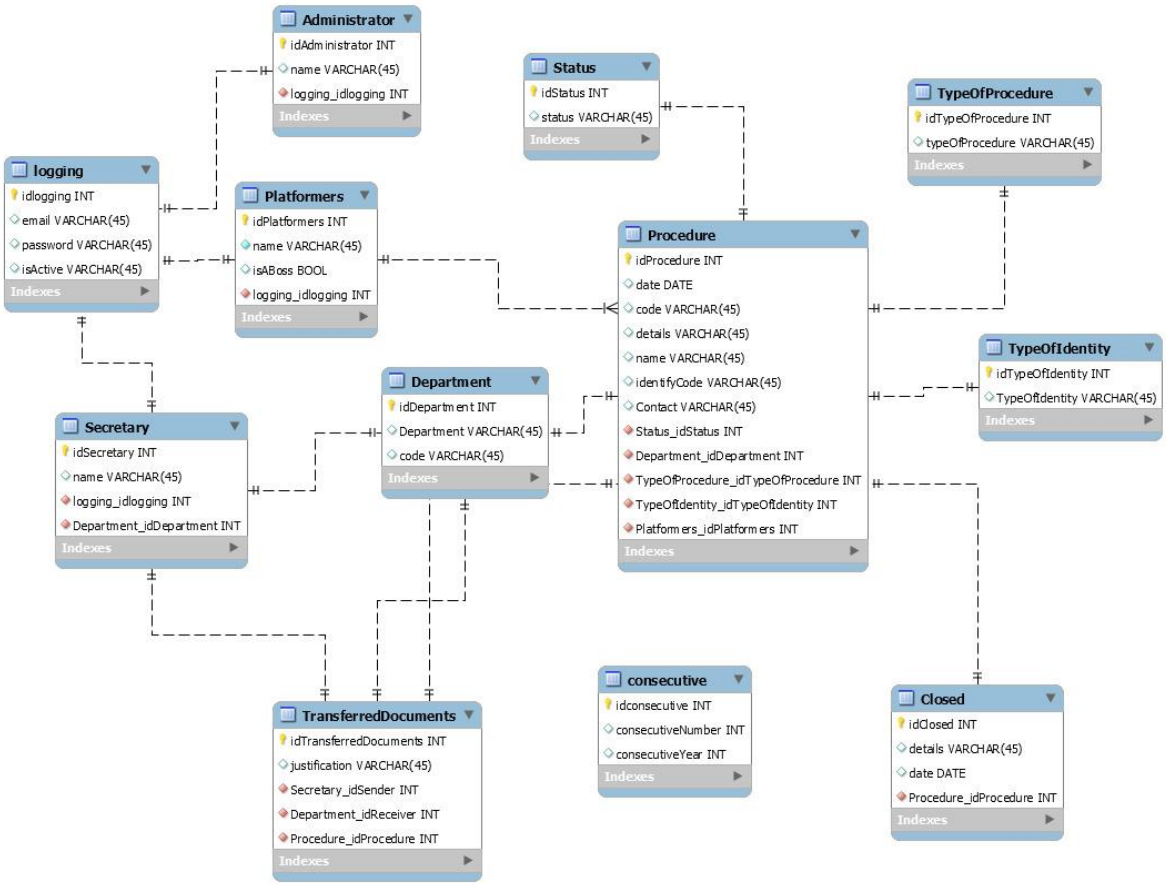

Imagen 10

Diccionario de Datos

Como se puede apreciar en la imagen 10 la base de datos está compuesta por 11 tablas. Las cuáles serán detalladas a continuación

Closed

Tabla donde se almacenan los trámites que han sido finalizado su ciclo.

Nombre	Tipo de Dato	Descripción
idClosed	int	Llave primaria para la tabla
details	nvarchar(300)	Detalle sobre cómo se cerró o terminó el trámite
date	date	Fecha en la que terminó el trámite

idProcedure	int	Llave foránea que conecta con el trámite que se está cerrando

Consecutive

Tabla donde se crean y almacenan el consecutivo para formar el número de seguimiento

Nombre	Tipo de Dato	Descripción
consecutiveId	int	Llave primaria de la tabla
consecutiveNumber	int	Número de consecutivo
consecutiveYear	int	Año en el que se está creando el número del consecutivo

Department

Tabla donde se almacena los datos de los departamentos.

Nombre	Tipo de Dato	Descripción
idDepartment	int	Llave primaria de la tabla
department	nvarchar(50)	Nombre del departamento
code	nvarchar(50)	Código del departamento

Logging

Tabla donde se almacenan las credenciales de ingreso al sistema.

Nombre	Tipo de Dato	Descripción
idLogging	int	Llave primaria de la tabla
email	Nvarchar (50)	Correo institucional del usuario del sistema
password	Nvarchar (300)	Contraseña del usuario del sistema, esta se guarda encriptada
isActive	int	Bandera sobre si el usuario está activo o no.

Plataformers

Tabla donde se almacenan los datos de los plataformistas.

Nombre	Tipo de Dato	Descripción
idPlataformers	int	Llave primaria de la tabla
name	Nvarchar (50)	Nombre completo del plataformista
isABoss	int	Variable para saber si el plataformista es jefe o no (1 si es jefe, 0 no es jefe)
idLogging	int	Llave foránea con la tabla logging para saber el correo electrónico y la contraseña del plataformista

Procedure

Está tabla es el corazón del sistema, ya que aquí es donde se encuentra toda la información de los trámites que los ciudadanos realizan en la municipalidad.

Nombre	Tipo de Dato	Descripción
idProcedure	int	Llave primaria de la tabla
date	date	Fecha en la que se recibió el trámite en plataforma
code	Nvarchar (50)	Número de consecutivo que se le asigna al trámite
details	Nvarchar (300)	Detalles del trámite
identifyCode	Nvarchar (50)	Número de Identificación del ciudadano
Name	Nvarchar (50)	Nombre completo del ciudadano
contact	Nvarchar (50)	Forma de Contacto del ciudadano, puede ser correo o número de teléfono
idStatus	int	Llave foránea con la tabla status para saber el estado del trámite, por defecto sería el estado "Nuevo"
idDepartment	int	Llave foránea con la tabla Department para saber a cuál departamento pertenece
idTypeOfProcedure	int	Llave foránea con la tabla typeOfProcedure para saber qué tipo de trámite es
idTypeOfIdentify	int	Llave foránea con la tabla typeOfIdentify para saber qué tipo de Número de identidad posee el ciudadano (Cédula, Pasaporte, Extranjera)
idPlatformers	int	Llave foránea con la tabla Platformers para saber cuál plataformista ingreso el trámite

Secretary

Tabla donde la está la información de los usuarios del sistema que no son plataformistas.

Nombre	Tipo de Dato	Descripción
idSecretary	int	Llave primaria de la tabla
name	Nvarchar (50)	Nombre completo de la secretaria
idLogging	int	Llave foránea con la tabla logging para saber el correo electrónico y la contraseña de la secretaria
idDepartment	int	Llave foránea con la tabla Department para saber a cuál departamento pertenece la secretaria

Status

Tabla donde se almacenan los tipos de estados que puede tomar un trámite.

Nombre	Tipo de Dato	Descripción
idStatus	int	Llave primaria de la tabla
status	Nvarchar (50)	Nombre de los estados que puede tener un trámite

transferredDocuments

Tabla donde se almacenan los archivos transferidos de los departamentos.

Nombre	Tipo de Dato	Descripción
idTransferredDocuments	int	Llave primaria de la tabla

justification	Nvarchar (50)	Justificación por la cual el trámite es transferido a otro departamento
idSender_Secretary	int	Llave foránea con la tabla Secretary para saber cuál secretaria lo envía
idReceiver_Department	int	Llave foránea con la tabla Department para saber cuál departamento lo recibe
idProcedure	int	Llave foránea con la tabla procedure para saber cuál es el trámite que se está transfiriendo

typeOfIdentify

Tabla donde se encuentran los tipos de identificación que pueden existir.

Nombre	Tipo de Dato	Descripción
idTypeOfIdentify	int	Llave primaria de la tabla
TypeOfIdentify	Nvarchar (50)	Nombre de los tipos de identificación que puede poseer un ciudadano

typeOfProcedure

Tabla donde están los tipos de trámites que procesa que la Municipalidad.

Nombre	Tipo de Dato	Descripción
idTypeOfProcedure	int	Llave primaria de la tabla
TypeOfProcedure	Nvarchar (50)	Nombre de los tipos de trámites que puede hacer el ciudadano

Configuración de la base de datos.

Para poder ejecutar la base de datos, se deben de correr los scripts que se encuentran en la dirección de github https://github.com/aifos29/Verano2016-2017, en la carpeta DataBase. Dichos scripts se encuentran enumerados en el orden en que deben de ser ejecutados.