Steelgo

***SAM 3.0***

Guía de Desarrollo

Versión 1.0

Septiembre 14, 2015



[Versiones 3](#_Toc430016604)

[1 Desarrollo de FrontEnd 4](#_Toc430016605)

[ApiManager 5](#_Toc430016606)

[DictionaryManager 6](#_Toc430016607)

[UtilitiesManager 6](#_Toc430016608)

[AlertsManager: 6](#_Toc430016609)

[Modelo de Seguridad 6](#_Toc430016610)

[2 Desarrollo BackEnd 9](#_Toc430016611)

[Controladores 9](#_Toc430016613)

[Acceso a datos 9](#_Toc430016614)

[Contexto de base de datos. 10](#_Toc430016615)

[Excepciones y tipos de retorno 11](#_Toc430016616)

[Seguridad en controladores 12](#_Toc430016617)

[Notificaciones 12](#_Toc430016618)

[Database Manager 13](#_Toc430016619)

[4 Configuraciones Básicas de Base de Datos 14](#_Toc430016620)

[Agregar un Usuario 14](#_Toc430016621)

[Añadir Página 15](#_Toc430016622)

[Añadir Perfiles 16](#_Toc430016623)

[Añadir elementos al menú 18](#_Toc430016624)

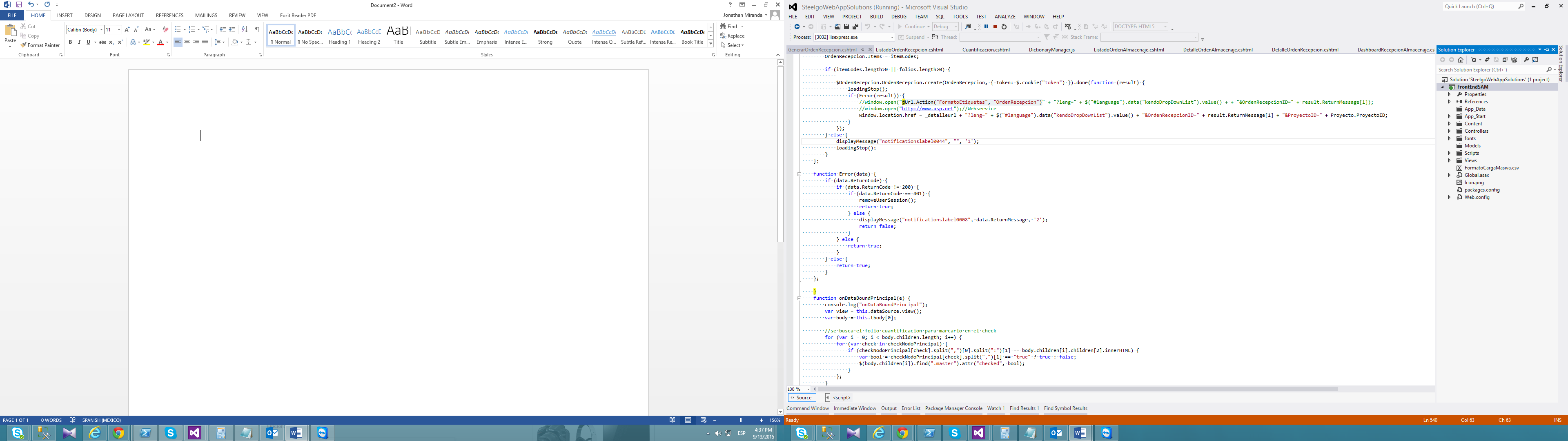
[Envío de notificaciones 21](#_Toc430016625)

# Versiones

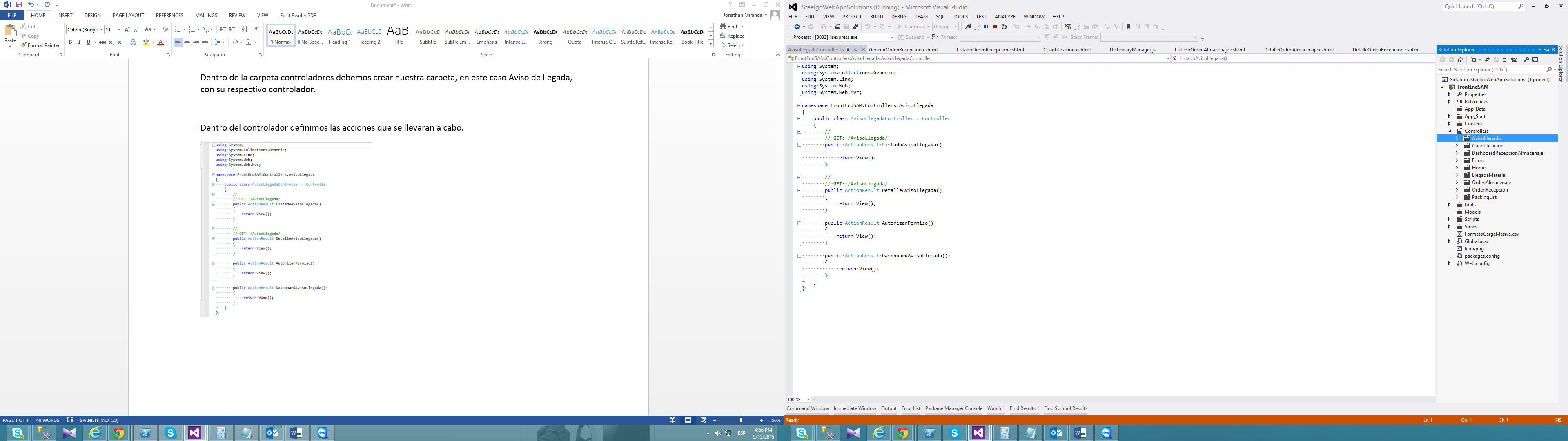
|  |  |  |  |
| --- | --- | --- | --- |
| **Fecha** | **Versión** | **Autor** | **Descripción** |
| 14-sept-2015 | 1.0 | Luis Manríquez | Creación del documento |

# Desarrollo de FrontEnd

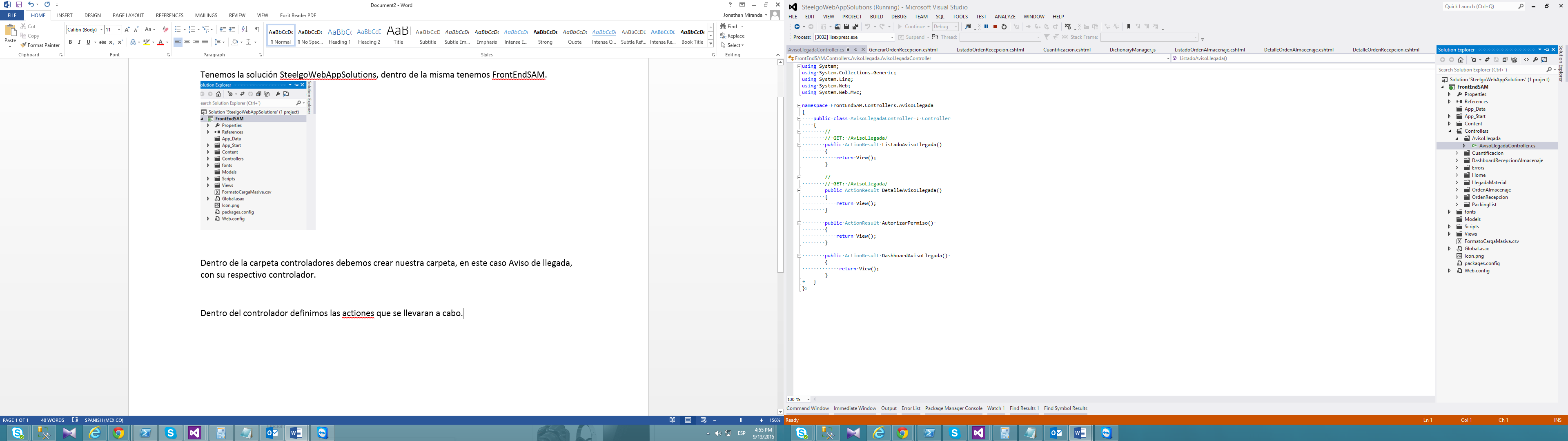
En la solución tenemos FrontEndSAM.



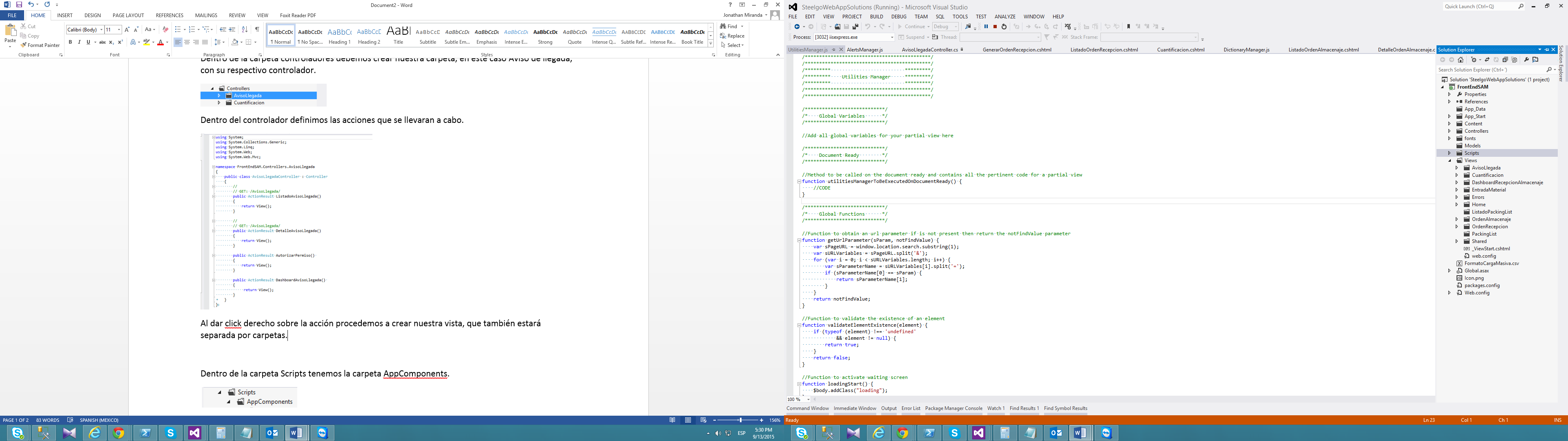
Dentro de la carpeta controladores debemos crear nuestra carpeta, en este caso Aviso de llegada, con su respectivo controlador.



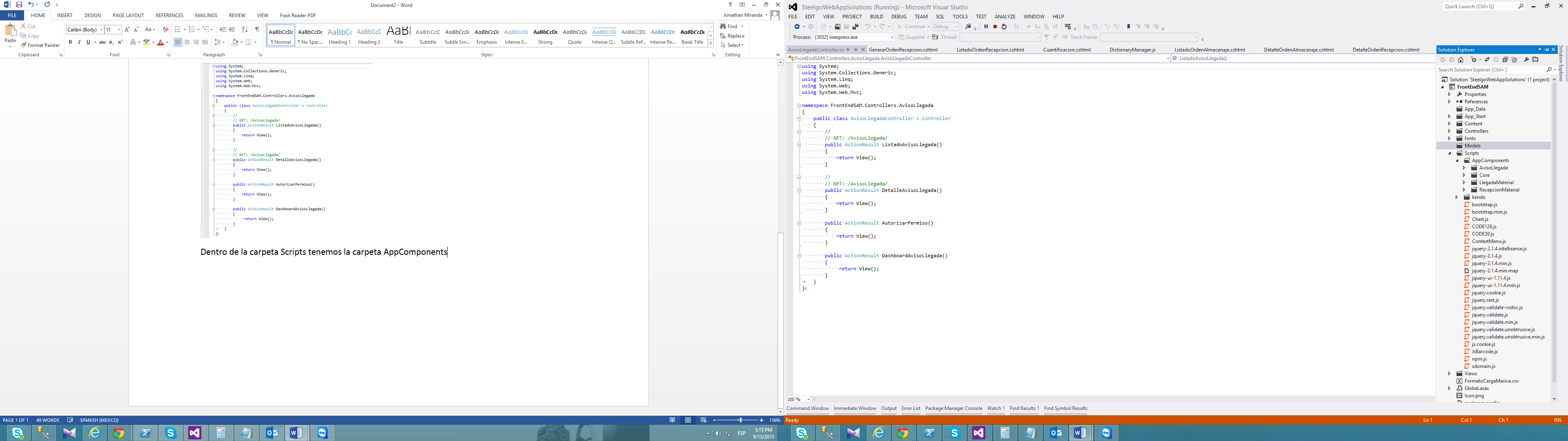
Dentro del controlador definimos las acciones que se llevaran a cabo.



Al dar click derecho sobre la acción procedemos a crear nuestra vista, que también estará separada por carpetas.



Dentro de la carpeta Scripts tenemos la carpeta AppComponents



En AppComponents tenemos la carpeta Core que contiene 4 archivos “js” muy importantes.

## ApiManager

Se encuentran las url de nuestro localhost:

var $BackEndSAMUri = 'http://localhost:60960';

var $SecuritySAMUri = 'http://localhost:60921';

var $MessagesSAMUri = 'http://localhost:60966';

var $SearchSAMUri = 'http://localhost:60921';

var $FilesSAMUri = 'http://localhost:60921';

Para nuestros servicios tenemos que declararla, en nuestro caso los servicios se encuentran en BackendSam:

var $Plana = new $.RestClient($BackEndSAMUri + '/backendsam/api/');

Agregamos nuestro controlador.

$Plana.add("Plana");

Ahora podemos utilizar create, destroy, update, segun sea el caso.

$Plana.Plana.create(PlanaModal, { token: Cookies.get("token") }).done(function (data) {

});

Estas son las dos formas de mandar la informacion:

1.-Mandamos PlanaModal con todas las propiedades.

PlanaModal = {};

PlanaModal = { TipoVehiculoID: "", Placas: };

$Plana.Plana.create(PlanaModal, { token: Cookies.get("token") }).done(function (data) {

});

2.- Mandamos cada uno de los valores:

$Plana.Plana.create({}, {Pruebaid:1,descripcion:”Prueba”, token: Cookies.get("token") }).done(function (data) {

});

## DictionaryManager

Debemos declarar las etiquetas de la siguiente forma:

"loginLabel0001": {

"es-MX": "Inicio de Sesión",

"en-US": "Login"

},

## UtilitiesManager

Se encuentran funciones que estaremos utilizando diariamente.

getUrlParameter: se utiliza para obtener los parámetros enviados en la url.

loadingStart: mostrar el loading para que el usuario no esté dando click varias veces al guardar, le ponemos un velo con la imagen de loading.

loadingStop: Se detiene el loading cuando termina de realizar la acción.

## AlertsManager:

Se encuentra una función “displayMessage”, tenemos opciones para el tipo de error 0 success, 1 warning, 2 error.

(la etiqueta, cualquier string que queramos mostrar,tipo de error);

displayMessage("notificationslabel0014", "", '1');

Dentro de la Vista debemos declarar las 4 secciones.

<script type="text/javascript">

Se declaran las variables globales a utilizar.

@section JavascriptGlobalVariables {

};

Se declaran todas las funciones

@section JavascriptGlobalFunctions {

};

Es la carga inicial de nuestros controles.

@section JavascriptDocumentReadyFunctions {

};

Debemos poner el número de página que le corresponde.

@section JavascriptDocumentReadyHomeCookie {

Cookies.set("home", true, { path: '/' });

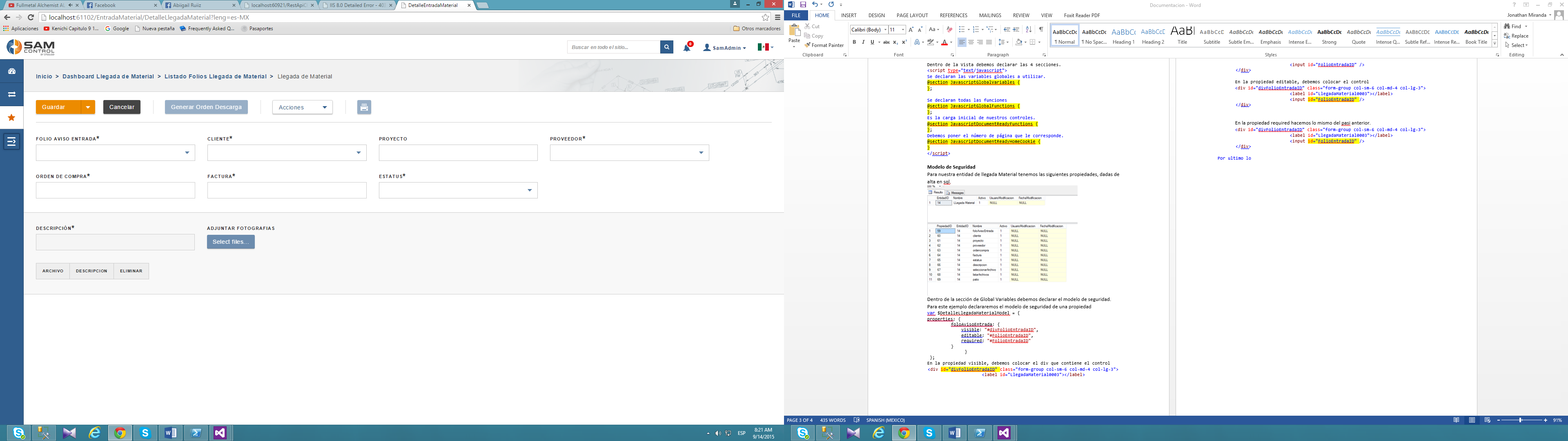
Cookies.set("navegacion", "7", { path: '/' });

}

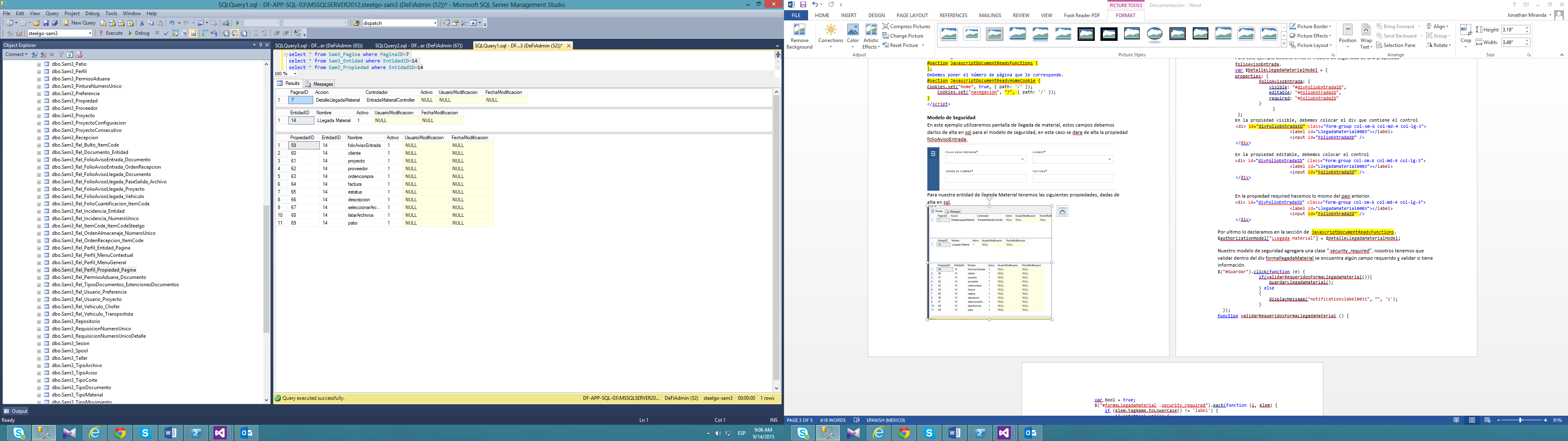
</script>

## Modelo de Seguridad

En este ejemplo utilizaremos pantalla de llegada de material, estos campos debemos darlos de alta en sql para el modelo de seguridad, en este caso se dara de alta la propiedad folioAvisoEntrada,



Para nuestra entidad de llegada Material tenemos las siguientes propiedades, dadas de alta en SQL.



Tambien tenemos que dar de alta nuestra entidad y página en [Sam3\_Rel\_Perfil\_Entidad\_Pagina]

, para nuestras pruebas siempre utilizamos el perfil “1”.

Tambien debemos dar de alta nuestras propiedades por página, para nuestras pruebas perfil “1”.

[Sam3\_Rel\_Perfil\_Propiedad\_Pagina]

Dentro de la sección de Global Variables debemos declarar el modelo de seguridad.

Para este ejemplo declararemos el modelo de seguridad de una propiedad folioAvisoEntrada.

var $DetalleLlegadaMaterialModel = {

properties: {

folioAvisoEntrada: {

visible: "#divFolioEntradaID",

editable: "#FolioEntradaID",

required: "#FolioEntradaID"

}

}

};

En la propiedad visible, debemos colocar el div que contiene el control

<div id="divFolioEntradaID" class="form-group col-sm-6 col-md-4 col-lg-3">

<label id="LlegadaMaterial0003"></label>

<input id="FolioEntradaID" />

</div>

En la propiedad editable, debemos colocar el control

<div id="divFolioEntradaID" class="form-group col-sm-6 col-md-4 col-lg-3">

<label id="LlegadaMaterial0003"></label>

<input id="FolioEntradaID" />

</div>

En la propiedad required hacemos lo mismo del pasi anterior.

<div id="divFolioEntradaID" class="form-group col-sm-6 col-md-4 col-lg-3">

<label id="LlegadaMaterial0003"></label>

<input id="FolioEntradaID" />

</div>

Por ultimo lo declaramos en la sección de JavascriptDocumentReadyFunctions.

$authorizationModel["LLegada Material"] = $DetalleLlegadaMaterialModel;

Nuestro modelo de seguridad agregara una clase “.security\_required”, nosotros tenemos que validar dentro del div formallegadaMaterial se encuentra algún campo requerido y validar si tiene información.

$("#Guardar").click(function (e) {

if(validarRequeridosFormaLlegadaMaterial()){

GuardarLlegadaMaterial();

} else

{

displayMessage("notificationslabel0031", "", '1');

}

});

function validarRequeridosFormaLlegadaMaterial () {

var bool = true;

$("#formaLlegadaMaterial .security\_required").each(function (i, elem) {

if (elem.tagName.toLowerCase() != 'label') {

if (!$(this).val()) {

bool = false;

$(this).closest("div").find("label").addClass("error");

$(this).closest("div").addClass("clearfix");

} else {

$(this).closest("div").find("label").removeClass("error");

$(this).closest("div").removeClass("clearfix");

};

};

});

return bool;

}

Si son requeridos mostrara un asterisco y si al guardar falto algún campo se pondrá en color rojo el asterisco por la clase .error.

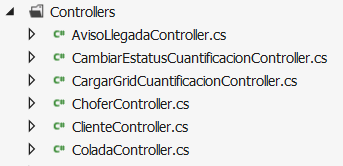
# Desarrollo BackEnd

La arquitectura del back end de SAM 3.0 corresponde al modelo vista controlador (MVC), en el cual tenemos controladores que funcionan como interface entre la entrada y salida de datos, una capa de acceso a datos y modelos de objetos.



## Controladores

Todos los controladores existentes o aquellos que se han de crear para la solución deberán contenerse en la carpeta “Controllers”.



Es necesario el uso de las siguientes referencias para que el controlador pueda atender las peticiones REST enviadas desde el Frontend.



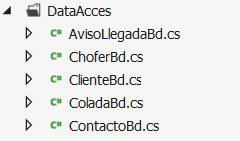
De igual modo la clase del controlador debe heredar la clase ApiController y estar decorada por un clausula EnableCors.



De este modo el controlador podrá escuchar las peticiones REST que se le envíen.

## Acceso a datos

Las clases de acceso a datos se agrupan en la Carpeta DataAccess del proyecto, estas clases agrupan los métodos necesarios para las operaciones sobre una entidad en específico.



Como convención hemos agregado al nombre de la clase el sufijo Bd, para hacer una distinción entre las clases de modelo, y las clases de acceso a datos.

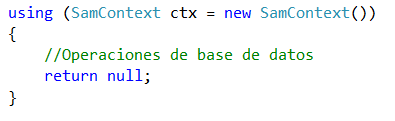
La conexión y manipulación de dato se hace a través del Entity Framework y del contexto correspondiente a la base de datos que se quiera manipular, es necesario agregar a la clase la referencia necesaria ya sea para SAM 3 (DatabaseManager.Sam3) o SAM2 (DatabaseManager.Sam2).

### Contexto de base de datos.

El proyecto ha sido configurado para crear contextos de base de datos, para lo cual habrá que modificar según sea necesario las cadenas de conexión necesarias, para el caso de SAM 3, en el web.config de la solución tenemos la siguiente línea de cadena de conexión.

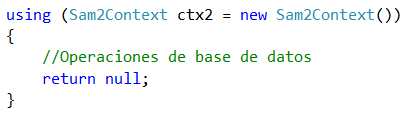
<add name="SamContext" connectionString="metadata=res://\*/Sam3.ModelSam3.csdl|res://\*/Sam3.ModelSam3.ssdl|res://\*/Sam3.ModelSam3.msl;provider=System.Data.SqlClient;provider connection string=&quot;data source=DF-APP-SQL-03\MSSQLSERVER2012;initial catalog=steelgo-sam3;persist security info=True;user id=DeFiAdmin;password=Chelsea102!;MultipleActiveResultSets=True;App=EntityFramework&quot;" providerName="System.Data.EntityClient" />

Para el uso de datos dentro de los métodos abriremos un contexto que nos servirá para rastrear los cambios o generar las consultas necesarias.



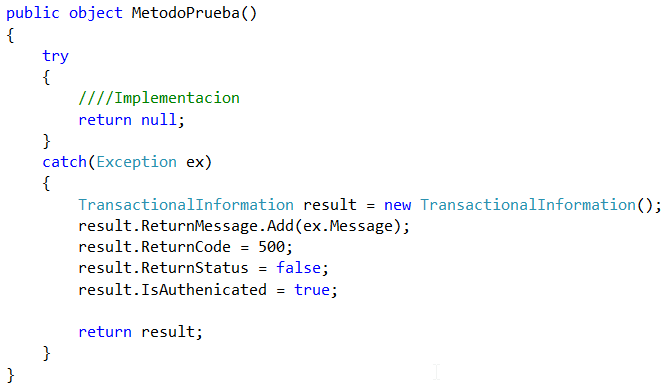
Este proceso es igual para el uso de la base de datos de SAM 2.

<add name="Sam2Context" connectionString="metadata=res://\*/Sam2.ModelSam2.csdl|res://\*/Sam2.ModelSam2.ssdl|res://\*/Sam2.ModelSam2.msl;provider=System.Data.SqlClient;provider connection string=&quot;data source=DF-APP-SQL-03\MSSQLSERVER2012;initial catalog=steelgo-sam2;persist security info=True;user id=DeFiAdmin;password=Chelsea102!;MultipleActiveResultSets=True;App=EntityFramework&quot;" providerName="System.Data.EntityClient" />

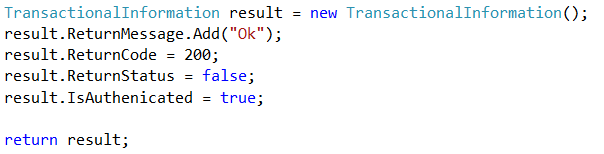


## Excepciones y tipos de retorno

Cada controlador debe poder devolver en todo momento ya sea una respuesta formateada de error o éxito, un set de datos necesario, o cualquier otro tipo de retorno de modo que para obedecer a este requerimiento todos los tipos de retorno son de tipo object.



Las excepciones o las respuestas de éxito que se devuelven al Front End serán tal como se muestra en imagen anterior. Y debajo un ejemplo de un mensaje de éxito.



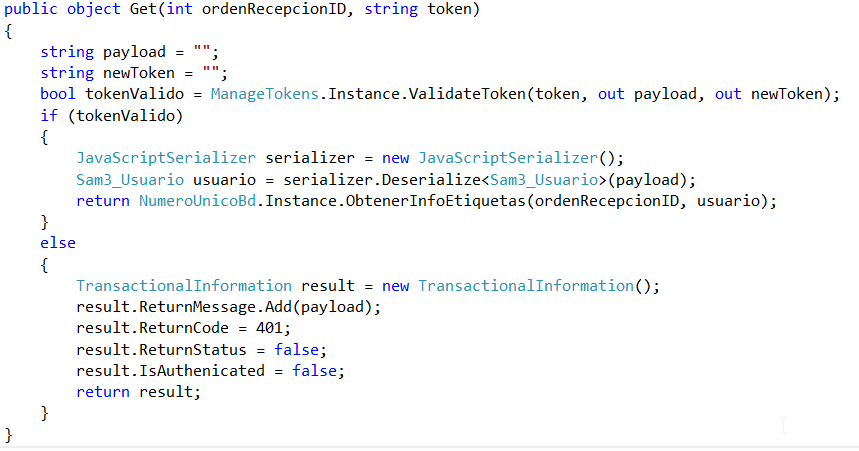
Result.ReturnMessage es un arreglo de strings en que se puede enviar información adicional para la operación del Front end como la información de un nuevo registro generado, etc.

## Seguridad en controladores

Back End no cuenta con sesiones de usuario, la autenticación y seguridad se maneja por medio del token que el usuario obtiene al iniciar sesión, todos los controladores reciben como parámetro el token del usuario, y hacen una llamada al Security Manager, específicamente al método ValidateToken de la clase ManageTokens.

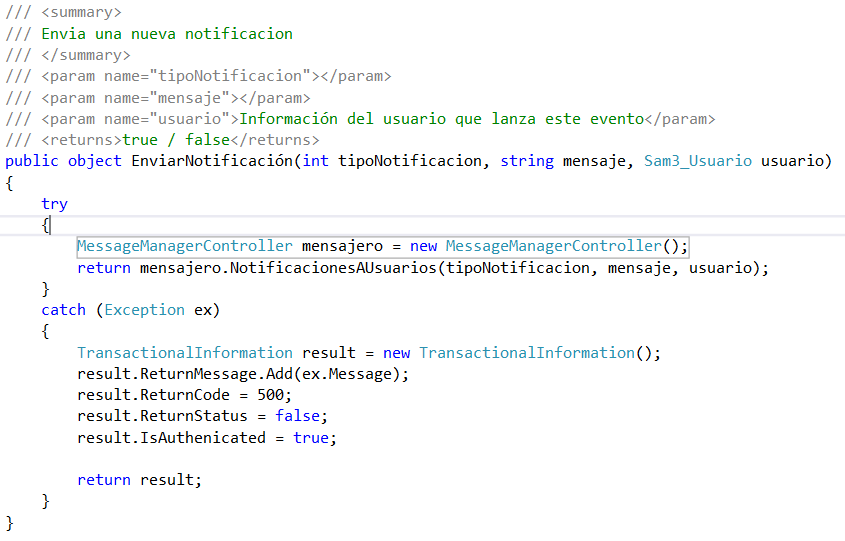
Este método valida que el token sea autentico y se encuentre vigente, devuelve un valor booleano que indica si el token es o no valido, en caso de ser válido el método cuanta con dos variables de salida que son; payload (información del usuario) y newToken (un nuevo token generado para ese usuario).

Esta es la forma básica de un método dentro de un controlador.



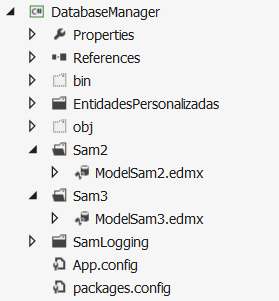
## Notificaciones

El envió de notificaciones hacia el MessageManager se hace a través de la clase EnviarAvisosBd en DataAccess, el método EnviarNotificacion es el encargado de hacer el enlace entre el back end y el proyecto de MessageManager.



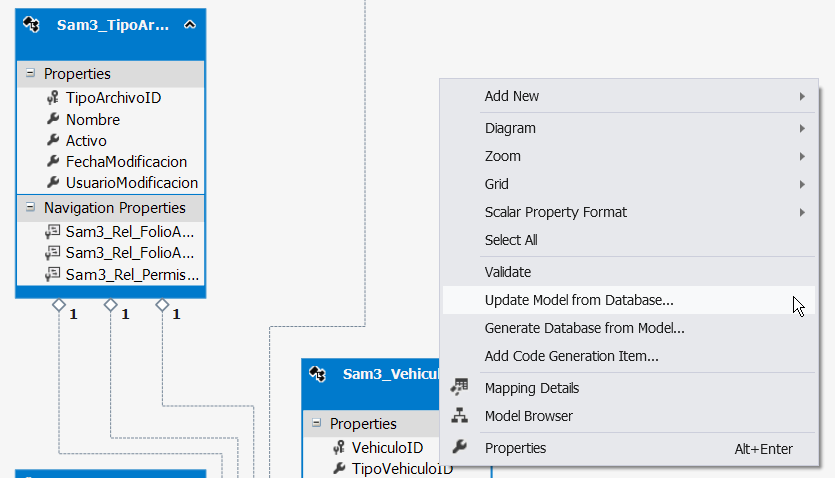
## Database Manager

El proyecto Database Manager es donde encontramos los modelos de bases de datos de SAM 3 y SAM 2, estos modelos fueron generados con Entity Framework 6.



Como se puede ver en la imagen en este proyecto encontramos por separado los modelos de base de datos (.edmx) de cada una de las bases de datos que se manejan en esta solución.

Cualquier cambio que se genere en la base de datos debe ser incluido en el modelo, para esto abrimos el modelo de base de datos que corresponda y sobre el diagrama damos un click derecho y elegimos la opción de actualizar modelo desde base de datos, como se muestra.



Se debe tener especial cuidado de los cambios en el modelo, pues este es suceptible de fallar si se modifica incorrectamente algún elemento de este. Después de cualquier actualización del modelo de base de datos se debe notificar a los demás usuario de la solución, para que hagan la actualización necesaria en sus repositorios locales.

Para esta primera fase el objetivo de Negocio es:

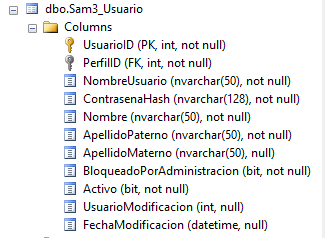
**Desarrollar Módulo de Materiales –** Como primer módulo a construir bajo la nueva versión será el módulo de materiales que incluirá nuevos modelos así como procesos ya existentes.

# Configuraciones Básicas de Base de Datos

## Agregar un Usuario

Para agregar un nuevo usuario a la base de datos nos ubicamos en la tabla *Sam3\_Usuario*

Y se captura la siguiente información:



* PerfilID:

Es el tipo de perfil que tendrá el usuario. Se encuentran en la tabla Sam3\_Perfil  
Cada perfil podrá realizar diferentes acciones y acceder a diferentes paginas

* NombreUsuario
* ContrasenaHash:

Contraseña que tendrá el usuario

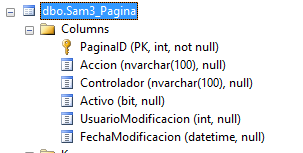
* Nombre
* ApellidoPaterno
* ApellidoMaterno
* BloqueadoPorAdministración:

En caso de que el usuario haya sido bloqueado

* Activo
* UsuarioModificacion
* FechaModificacion

## Añadir Página

Para agregar la dirección de una nueva página en la base de datos, nos ubicamos en la tabla *Sam3\_Pagina*



Se captura la siguiente información:

* Acción
* Controlador:

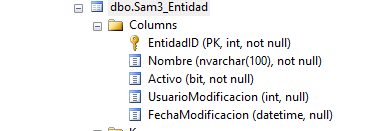
Nombre del controlador como se encuentra en FrontEnd

* Activo: Si la página se encontrará activa
* UsuarioModificacion
* FechaModificacion

**Añadir Entidades**

Las entidades son secciones que contiene cada página las cuales podrán ser vistas dependiendo del perfil que tenga el usuario. Cada entidad contiene Propiedades, que es cada elemento que se encuentre dentro de esa entidad.

Para añadir entidades a la base de datos nos ubicamos en la tabla *Sam3\_Entidad*



Ahí capturamos la siguiente información:

* Nombre:

Nombre de la entidad/sección que será visible para el usuario según los permisos que tenga

* Activo:

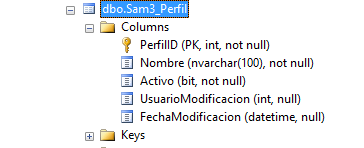
Si la sección se encuentra activa

* UsuarioModificacion
* FechaModificacion

## Añadir Perfiles

Los perfiles son los diferentes tipos de usuarios. Cada perfil tiene diferentes permisos

Para añadir perfiles nos ubicamos en la tabla *Sam3\_Perfil*



Se captura la siguiente información:

* Nombre:

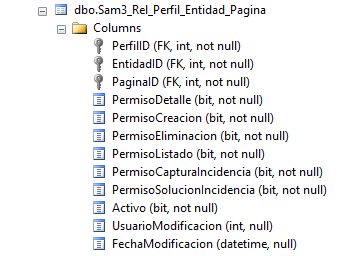
Nombre que tendrá el perfil

* Activo:

Si el perfil está activo

* UsuarioModificacion
* FechaModificacion

Para asignar diferentes permisos a un usuario se deberá capturar en la tabla *Sam3\_Rel\_Perfil\_Entidad\_Pagina* la siguiente información*:*



* PerfilID:

Id del perfil al cual se le otorgarán los permisos

* EntidadID:

Entidad que podrá ser vista por el perfil seleccionado

* PaginaID

Página que podrá ser vista por el perfil seleccionado

* PermisoDetalle

True/False, Si el perfil seleccionado podrá ver el Detalle de una página

* PermisoCreacion

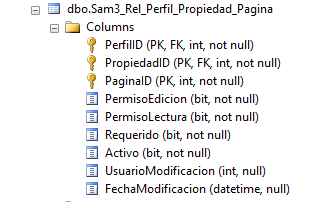
True/False, Si el perfil seleccionado contará con el permiso de Creación en la página

* PermisoEliminacion

True/False, Si el perfil seleccionado contará con el permiso de Eliminación en la página

* PermisoListado  
  True/False, Si el perfil seleccionado podrá ver los diferentes listados de la página
* PermisoCapturaIncidencia  
  True/False, Si el perfil seleccionado podrá capturar Incidencias
* PermisoSolucionIncidencia  
  True/False, Si el perfil seleccionado podrá dar Solución a las Incidencias
* Activo  
  Si el permiso se encuentra activo
* UsuarioModificacion
* FechaModificacion

Para agregar las propiedades que podrá ver cada perfil nos ubicamos en la tabla *Sam3\_Rel\_Perfil\_Propiedad\_Pagina*



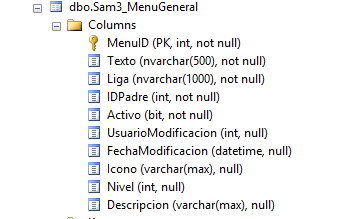
Se captura la siguiente información:

* PerfilID

Id del perfil al cual se le otorgarán los permisos

* PropiedadID
* Propiedad que tendrá los permisos
* PaginaID
* Página en la que se encuentra la propiedad seleccionada
* PermisoEdicion  
  True/False, Si el perfil contará con permiso de Edición
* PermisoLectura  
  True/False, Si el perfil contará con permiso de lectura
* Requerido  
  True/False
* Activo  
  Si el permiso se encuentra Activo
* UsuarioModificacion
* FechaModificacion

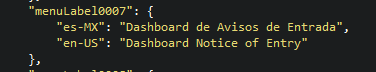
## Añadir elementos al menú

Para añadir elementos al menú nos ubicamos en la tabla *Sam3\_MenuGeneral*

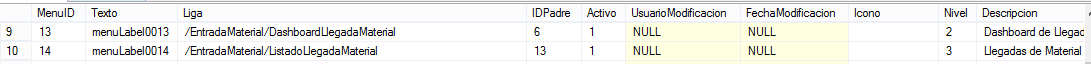
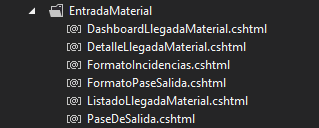
Se captura la siguiente información:

* Texto: Nombre del texto que se mostrará en el menú, el cual deberá existir en FrontEnd en la carpeta Scripts/AppComponents/Core/DictionaryManager.js con su traducción

Ejemplo:

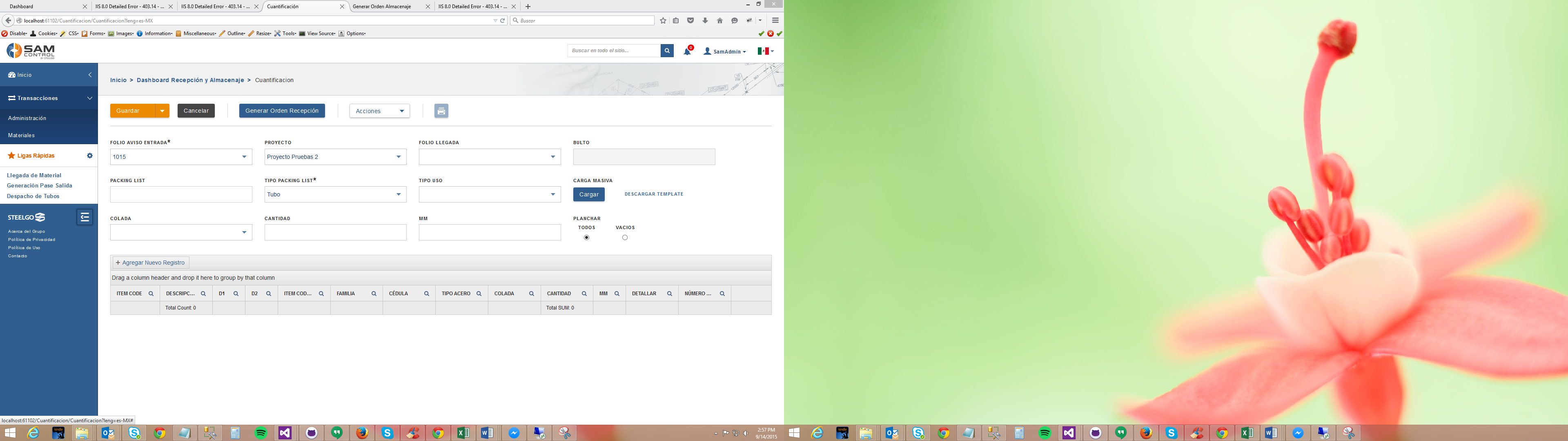
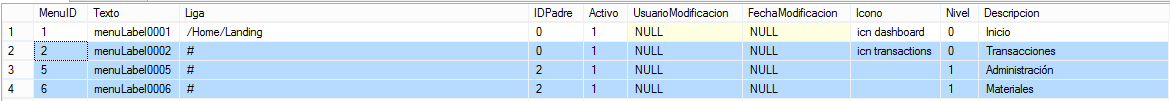


* Liga: Dirección para acceder a la página

Ejemplo

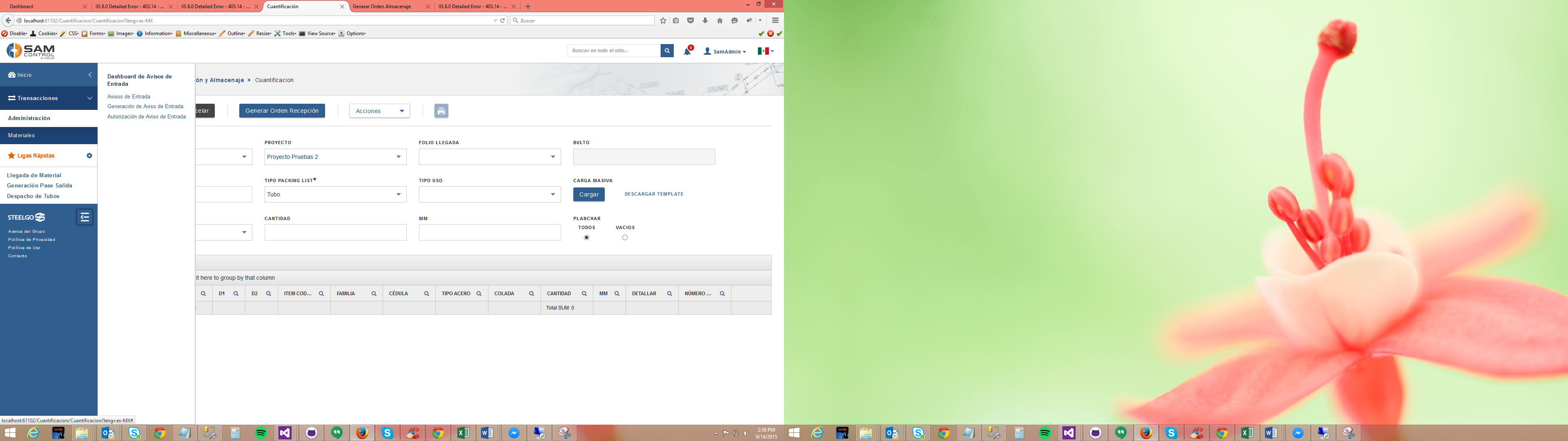
* IDPadre: Sección del menú de la que depende

Ejemplo:



Ejemplos de Padres

Administración y Materiales dependen de Transacciones (Padre)

* Activo:   
  Si la sección del menú será visible
* UsuarioModificacion
* FechaModificacion
* Icono: Ícono que aparecerá a un lado del menú
* Nivel: Nivel en el que aparecerá el menú

Nivel 2

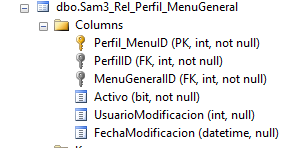
Nivel 1

Nivel 3

* Descripción

Se deberán agregar los permisos al menú

Nos ubicamos en la tabla Sam3\_Rel\_Perfil\_MenuGeneral



Se capturará la siguiente información

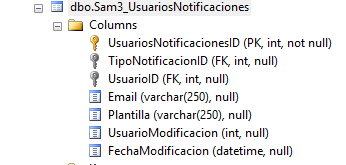
* PerfilID

Perfil que podrá ver el menú

* MenuGeneralID  
  Id del menú que podrá ser visto por el perfil
* Activo
* UsuarioModificacion
* FechaModificacion

## Envío de notificaciones

Para el envío de notificaciones se capturarán los usuarios que podrán recibir los diferentes tipos de notificaciones



* TipoNotificacionID:

ID de la notificación que tendrá cada usuario (Tabla TipoNotificacion)

* UsuarioID:

Id del usuario que tendrá el tipo de notificación seleccionado

* Email:

Email del usuario

* Plantilla:

Texto que tendrá el correo / notificación que se enviará

* UsuarioModificación
* FechaModificación