SUSTENTANTE: ERIC ALEJANDRO CHÁVEZ GÓMEZ

CARRERA: TECNOLOGÍAS DE LA INFORMACIÓN Y LA COMUNICACIONES

NO.CONTROL:111050146

PROYECTO: SOFTV

EMPRESA: SISTEMAS ADMINISTRATIVOS PARA TV RESTRINGIDA

ASESOR: MITC RAFAEL PRECIADO GUTIERREZ

TITULACIÓN: OCTUBRE 2017

Tabla de Contenido

Lista de Tablas

{Listar en forma ascendente todos los títulos de las tablas del documento con los respectivos números de página}

Lista de Figuras

Introducción

Las comunicaciones como servicio en algunos casos puede extenderse a niveles insospechados y el alcance de los servicios algunas veces debe de extenderse másallá de lo previsto al planear un proyecto ,este documento expondrá como resolver la escalabilidad de un sistema de escritorio que sobrepasa los niveles de concurrencia no previstos a las mejores y más actuales tecnologías de desarrollo web y como rediseñar la complicada estructura de un sistema en producción sin sacrificar el performance y funcionalidad del sistema existente añadiendo que el sistema de escritorio debe ser migrado a un sistema web en el menor tiempo posible.

¿Podrán las tecnologías más actuales afrontar este problema?

Este es el caso de la empresa Sistemas Administrativos para tv Restringida (SOFTV) es una importante empresa del ramo de las comunicaciones digitales, estase dedica al desarrollo de sistemas de administración de varias compañías y proveedores de televisión cable e internet del territorio mexicano y algunas más del extranjero . la empresa creo como Core un sistema que está desarrollado en Visual Basic el cual contiene todas las funcionalidades que una empresa del sector de tv por cable necesita para su administración y cobranza, es un sistema utilizado por varios de sus clientes con resultados favorables los cual les ha dado prestigio en el sector de las comunicaciones.

Este software es llamado **Softv** por la empresa SATV lleva más de 10 años en el mercado

Su principal característica es su sistema de administración de clientes, su proceso de cobranza donde implementa usuarios de varios niveles que se encargan de varios procesos para la administración. Además de manejar periodos de pago y cobro de servicios de cable y servicios de internet, reportes y estadísticas.

El sistema se divide por un administrador principal (el administrador tiene control y acceso total de todos los procesos) este a su vez puede crear otros usuarios que a su vez funcionan como distribuidores o sucursales los cuales pueden contener técnicos y terminales (cajas).

Una empresa de cable que se administra con SOFTV se divide en varias formas

- Una sucursal principal y un almacén principal independiente ,el almacén principal suministran materiales a todas las sucursales sin que estas almacenen, cada sucursal tiene sus propios técnicos (**figura 1**)

- Una sucursal principal y un almacén principal independiente, los cuales administran material a las demás sucursales, pero estas pueden tener su almacenes operar de manera independiente, realizando órdenes de compra al almacén central y el almacén se encarga de surtir material. Cada sucursal tiene técnicos a su cargo(**figura 2**)

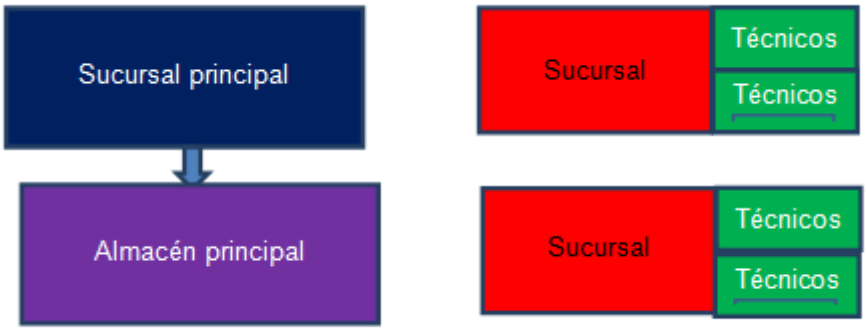


Figura 1 diagrama tipo de administración 1

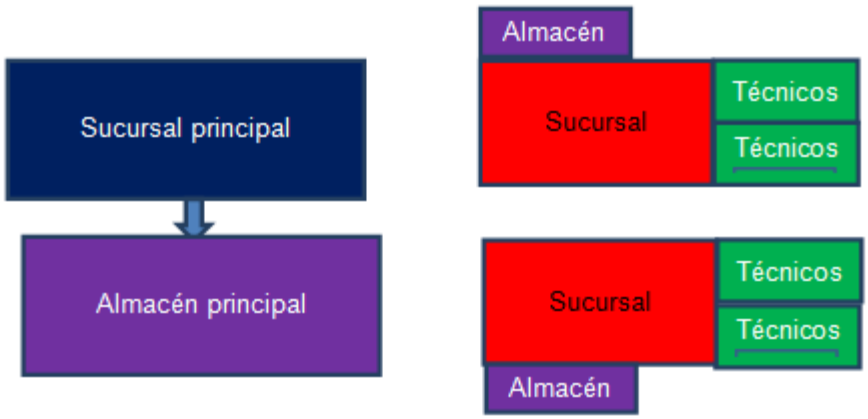


Figura 2

PROCESO DE CONTRATACION, INSTALACION DE SOFTV

El sistema SOFTV trabaja de la siguiente manera

El cliente llega a una caja de alguna sucursal para contratar algún servicio de cable o internet, el pago se registra en el sistema como cliente nuevo, se genera una orden de activación e instalación, se notifica al almacén de la sucursal que se contrató el servicio, Se revisa si en su almacén cuenta con material, si cuenta se realiza un salida de material al técnico, el cual se dirigirá al domicilio del cliente para instalar el servicio.

En caso que la sucursal no cuente con material ni el técnico cliente, se realiza unaorden de compra al almacén principal el cual surtirá material al almacén que se requirió y continua con el proceso de instalación.

El siguiente proceso se explica en el siguiente diagrama:

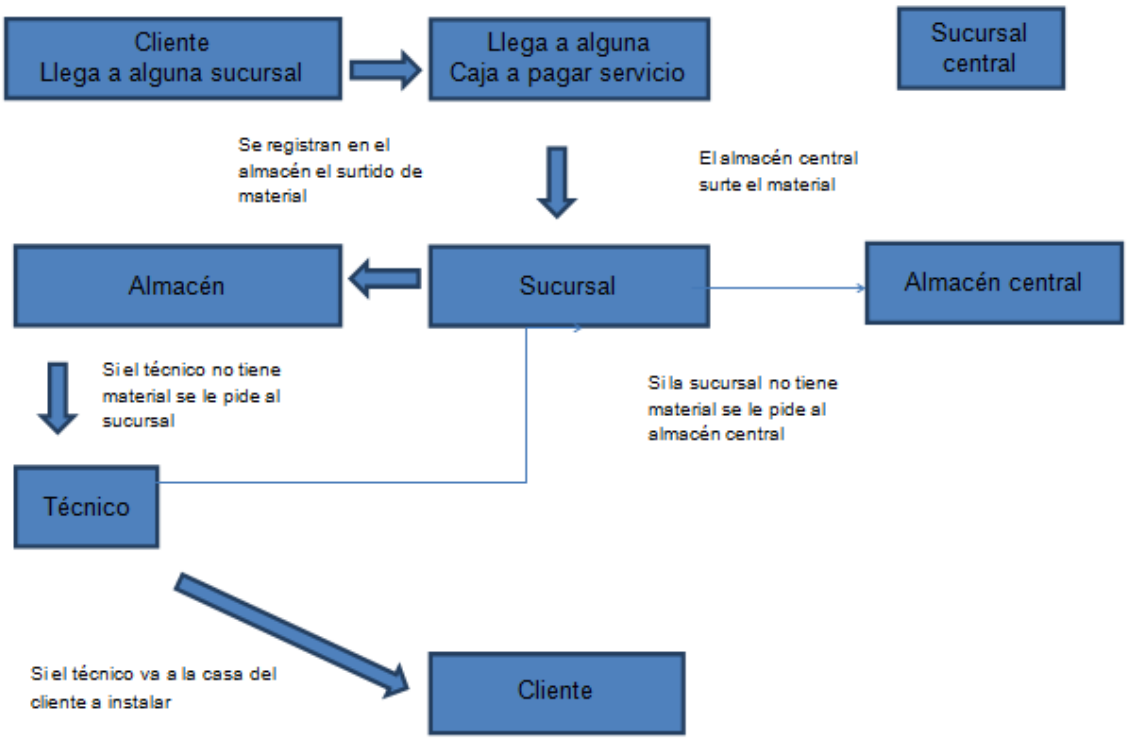


Figura 3

El crecimiento de la empresa y de sus clientes ha estado derivando varios tipos de problemas que desde un principio no se tuvieron en cuenta en el momento de la programación y la estructura de un proyecto que se inició sin contemplar su potencial expansión.

Uno de los varios problemas se debe a la accesibilidad y el performance del software, ya que es necesario que las actividades que ejerce cada uno de los usuarios en SOFTV debe ser lo más rápido posible, lo cual en algunos casos como es el cobro a los clientes o activación de material (cable módems y set top box son lentas y pueden dañar o alterar los ingresos de los clientes por ofrecer servicios que no son los más apropiados).

Los clientes exigen un sistema rápido, que pueda crecer a la par de las necesidades de la empresa sin afectar sus operaciones algo que el sistema actual que están usando no lo puede hacer tan fácilmente y requeriría inversión en tiempo y costos para el desarrollo y los clientes

Algunas de las ventajas de migrar el sistema web son:

- Ahorran costes de hardware y software
- Fáciles de usar
- Facilitan el trabajo colaborativo y a distancia
- Escalables y de rápida actualización
- Provocan menos errores y problemas
- Los datos son más seguros

se cree que las tecnologías web más actuales como Json y Angular JS podrían mejorar la rapidez del procesamiento de la información ,son tecnologías que están siendo utilizadas por empresas en todo el mundo y han tenido protagonismo en estos años, por tal motivo serán las tecnologías elegidas para migrar los datos a un entorno web.

Marco Teórico

Tecnologías y herramientas que las personas utilizan para intercambiar, distribuir y recolectar información y para comunicarse con otras personas. Las TIC pueden agruparse en tres categorías. Las tecnologías de información utilizan computadores, que se han vuelto indispensables en las sociedades modernas para procesar datos y economizar tiempo y esfuerzos. Las tecnologías de telecomunicaciones incluyen teléfonos (con fax) y transmisión de radio y televisión, a menudo a través de satélites. Las redes de tecnologías, de las que la más conocida es internet, también abarcan la tecnología de teléfono celular, la telefonía de voz sobre IP (VoIP), las comunicaciones por satélite y otras formas de comunicación que aún están siendo desarrolladas.

La conectividad está aumentando el flujo de información sobre el mercado, servicios financieros y servicios sanitarios hacia las zonas remotas, contribuyendo así a cambiar la vida de la gente en una forma sin precedentes. Las nuevas tecnologías de la información y de la comunicación (TIC), en particular la Internet de alta velocidad, están modificando el modo en que las empresas hacen negocios, transformando la prestación de servicios públicos y democratizando la innovación.

Antecedentes de c#

C# es un lenguaje de programación que se ha diseñado para compilar diversas aplicaciones que se ejecutan en .NET Framework. C# es simple, eficaz, con seguridad de tipos y orientado a objetos. Las numerosas innovaciones de C# permiten desarrollar aplicaciones rápidamente y mantener la expresividad y elegancia de los lenguajes de estilo de C.

Visual C# es una implementación del lenguaje C# de Microsoft. Visual Studio ofrece compatibilidad con Visual C# con un completo editor de código, un compilador, plantillas de proyecto, diseñadores, asistentes para código, un depurador eficaz y de

fácil uso y otras herramientas. La biblioteca de clases de .NET Framework ofrece acceso a numerosos servicios de sistema operativo y a otras clases útiles y adecuadamente diseñadas que aceleran el ciclo de desarrollo de manera significativa.

Por qué elegir REST Y NO SOAP PARA CONSUMIR SERVICIOS

SOAP es un protocolo de acceso a servicios Web basados en estándares que ha existido durante un tiempo y disfruta de todas las ventajas de su uso a largo plazo. Originalmente desarrollado por Microsoft, SOAP en realidad no es tan simple como el acrónimo podría sugerir.

De SOAP se basa exclusivamente en XML para proporcionar servicios de mensajería. Microsoft desarrolló originalmente SOAP para tomar el lugar de las tecnologías más antiguas que no funcionan bien en Internet, tales como el modelo de objetos componentes distribuidos (DCOM) y CORBA (CORBA). Estas tecnologías fallan porque se basan en la mensajería binaria; la mensajería XML de SOAP que emplea funciona mejor a través de Internet.

La visión general de REST

Muchos desarrolladores de SOAP encontraron engorroso y difícil de usar. Por ejemplo, trabajar con SOAP en JavaScript significa escribir un montón de código para realizar tareas muy simples, ya que debe crear la estructura XML que sea absolutamente necesario en todo momento.

REST proporciona una alternativa de peso ligero. En lugar de utilizar XML para hacer una solicitud, REST se basa en una simple URL en muchos casos. En algunos casos, usted debe proporcionar información adicional de una manera especial, pero la mayoría de los servicios Web REST utilizando confiar exclusivamente en la obtención de la información necesaria utilizando el enfoque de la URL. RESTO puede utilizar cuatro HTTP 1.1 verbos diferentes (GET, POST, PUT y DELETE) para realizar tareas.

SOAP es sin duda la opción de peso pesado de acceso al servicio web. Se ofrece las siguientes ventajas en comparación con los demás:
Idioma, la plataforma y el transporte independiente (REST requiere el uso de HTTP)
Funciona bien en entornos empresariales distribuidos (REST asume la comunicación directa de punto a punto)
Estandarizado
Proporciona significativa extensibilidad pre-estructura en la forma de los estándares WS *

- Construido en el tratamiento de errores
- Automatización cuando se utiliza con ciertos productos de lenguaje

REST es más fácil de usar en su mayor parte y es más flexible. Tiene las siguientes ventajas en comparación con SOAP:

- No se requieren herramientas caras para interactuar con el servicio Web
- curva de aprendizaje más pequeña
- Eficiente (SOAP utiliza XML para todos los mensajes, REST puede utilizar formatos de mensajes más pequeños)
- Fast (sin extensa procesamiento requerido)
- Más cerca de otras tecnologías Web en la filosofía de diseño

Metodología

Control de versiones

GitHub es una plataforma de desarrollo colaborativo de software para alojar proyectos utilizando el sistema de control de versiones Git.
GitHub aloja tu repositorio de código y te brinda herramientas muy útiles para el trabajo en equipo, dentro de un proyecto.

Además de eso, puede contribuir a mejorar el software de los demás. Para poder alcanzar esta meta, GitHub provee de funcionalidades para hacer un fork y solicitar pulls.

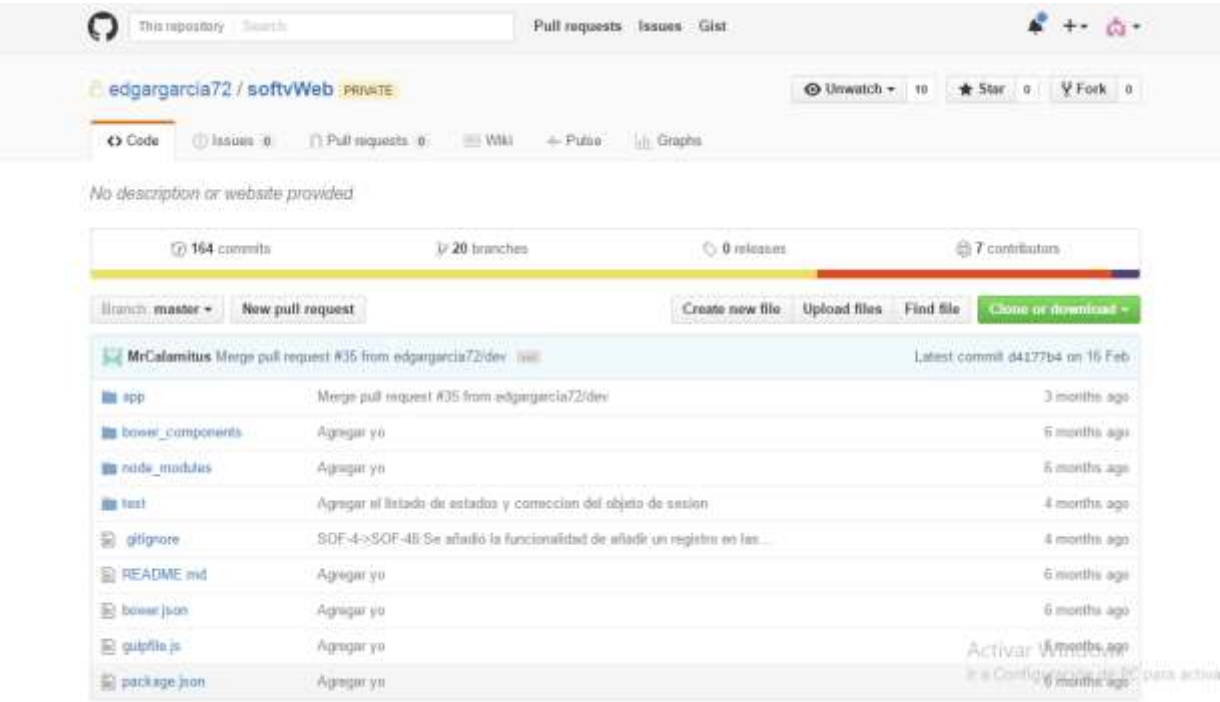


Figura 4

Documentación del software y planificación

Jira

JIRA es una aplicación basada en web para el seguimiento de errores, de incidentes y para la gestión operativa de proyectos. Jira también se utiliza en áreas no técnicas para la administración de tareas. La herramienta fue desarrollada por la empresa australiana Atlassian. Inicialmente Jira se utilizó para el desarrollo de software, sirviendo de apoyo para la gestión de requisitos, seguimiento del estatus y más tarde para el seguimiento de errores. Jira puede ser utilizado para la gestión de procesos y para la mejora de procesos, gracias a sus funciones para la organización de flujos de trabajo.

Figura 5 paneles de administración de JIRA

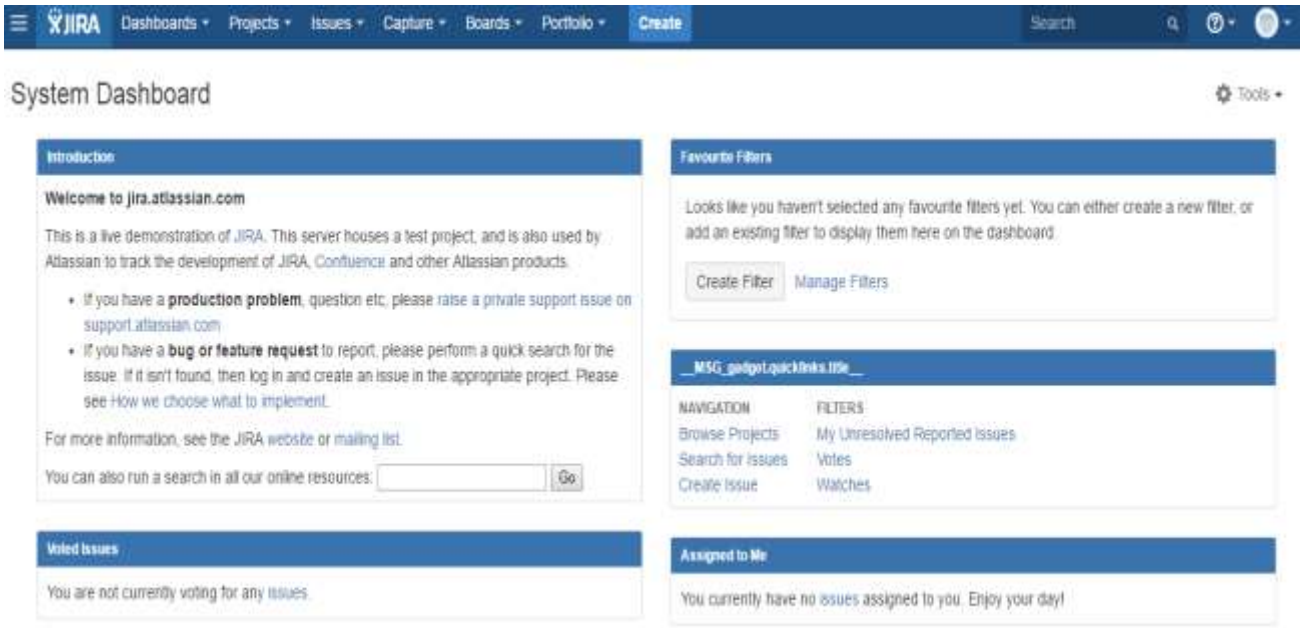


Figura 5

Para iniciar con la migración del sistema primero se recolectara la información de tablas y campos de SQL Server se recolectaran las tablas de bases de datos de “Newsoftv” que consta de alrededor de 970 tablas las cuales se filtraran las que están en funcionamiento y las que podrían ser omitidas para la migración en web, uno de los objetivos de la migración es poder reducir la cantidad de tablas existentes ya que alrededor de la mitad de las tablas ya no son utilizadas por los sistemas instalados actualmente ,al ser un número considerable de tablas se empezara a migrar las más importantes y principales que son el núcleo del sistema estas son los rangos de importancia:

- **INFORMACION DE EMPRESA Y DISTRIBUIDORES**

  Las tablas con información fiscal sobre la empresa y sus terminales, Carteras de clientes, información financiera y fiscal de sucursales y entidades (técnicos, personal, usuarios, permisos), pagos generales, movimientos

- **PROCESOS**

  Información de cobro de servicios, instalaciones, cortes, paquetes, tipos de servicioFacturación

- **CLIENTES**

  Las tablas de clientes, las relaciones con contratos, relaciones de domicilio y clientes, historial de pagos, información personal y crediticia del cliente.

- **INFORMACION GEOGRAFICA**

  La información de estados, calles, ciudades relacionadas con los servicios, ubicación de alambrado.

- **CATALOGOS**

  Sectores, artículos, clientes, técnicos, distribuidores, pagos, periodos cajas, bancos, tipos de procesos.

Una vez que se filtraron de la base de datos las tablas más importantes para el funcionamiento del sistema se crearan procedimientos almacenados por cada tabla que se extraiga los procedimientos almacenados son los más eficientes de SQL Server y evitaran que el cliente tenga una interacción directa con datos del sistema.

Cada procedimiento creado ayudara a la rápida recolección de datos, los procedimientos se llamaran desde el servicio REST para ser consumidos algunos de ellos tendrán que recibir parámetros, se nombraran con el nombre inicial **GET** para obtener información **Add**para agregar información**, UPDATE** donde requiera actualizarse información**, DELETE** para eliminar registros**, GetPagedList**para obtener una lista paginada de información**, GetDeep**para obtener un registro en especifico

Los procedimientos por cada tabla se detallan a continuación:

NOMBRE PROCEDIMIENTO	PARAMETRO	DETALLE
Add+Nombre de tabla	Objeto	Agrega un nuevo registro a la tabla
GetList+ Nombre de tabla	Sin parámetro	Retorna la todos los registros de la tabla
GetPagedList+ Nombre de tabla	Intinicio,int fin ,intcant_tomar	Retorna una lista de registros requeridos por el cliente

GetDeep+ Nombre de tabla	Intid_registro	Retorno un el registro que coincide con el id
Update + Nombre de tabla	Objeto	Retorna el objeto editado
Delete+ Nombre de tabla	Intid_registro	Retorna el objeto eliminado

Figura 6 Detalle y estructura de procedimientos almacenado

Estructura del proyecto

El proyecto se dividirá en varias capas según las buenas prácticas de codificación para capas de acceso a datos de aplicaciones será un modelo de 3 capas el cual Microsoft recomienda en las buenas practicas detallado en el enlace siguiente

https://msdn.microsoft.com/es-es/library/dn151512.aspx

Figura 6

El perfil del desarrollador se encargaría de implementar toda la parte relativa al aplicativo, mientras que el DBA/DBD se encargaría de mantener e implementar procesos a nivel de base de datos. Tal y como se muestra en las figuras 2 y 3, los desarrolladores piensan en los datos que manejan las aplicaciones como una serie de entidades u objetos, con atributos, colecciones de objetos etc. En cambio, el DBA/DBD los ve desde un punto de vista relacional, y piensa en tablas, relaciones, integridad referencial, restricciones etc. Claramente, son dos perspectivas muy distintas sobre una misma aplicación.

El proyecto principal se codificara en Visual Studio 2015 se dividirán las capas y módulos por carpeta para una mayor accesibilidad y rápida codificación

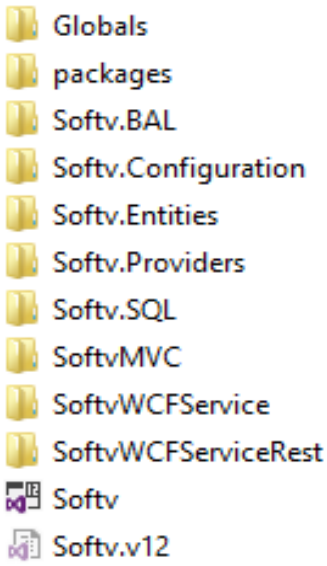


Figura 7

A Continuación se muestra los detalles de la estructura del proyecto:

CARPETA	DETALLES
Globals	Esta carpeta almacenara las clases útiles ,serializadores y Herramientas del proyecto
Packages	Esta carpeta contiene todos las librerías que se utilizan en el proyecto
Softv.BAL	Contiene las clases del modelo de negocios
Softv.Configuration	Contiene las configuraciones del proyecto
Softv.Providers	Contiene la comunicación entre los servicios REST y la capa de modelo de negocios
Softv.SQL	Contiene la conexión entre el manejador de la base de datos y el proyecto
Softv.MVC	Contiene las vistas y controladores del proyecto en general
SoftvWFCService	Servicios Rest de la solución

Softv.Entities	Las entidades atributos y relaciones de la base de datos

Figura 9 Estructura de las capas de desarrollo

Softv.BAL,Softv.SQL , Softv.Providers Y SoftvWFCService son las capas de desarrollo de la solución ,a continuación se detallara estas capas y su interacción e importancia de cada una de ellas en el proyecto.

1. **Business LogicLayer (BAL)**

   Esta capa es la lógica del negocio se llama así porque en esta capa se harán las validaciones, cambios y optimizaciones que los datos que pasan por esta capa sean los requeridos para cada tipo de necesidad.

```
public class CLIENTE
{

    #region Constructors
    public CLIENTE() { }
    #endregion

    /// <summary>
    ///Adds CLIENTE
    ///</summary>
    [DataObjectMethod(DataObjectMethodType.Insert, true)]
    public static int Add(CLIENTEEntity objCLIENTE)
    {
        int result = ProviderSoftv.CLIENTE.AddCLIENTE(objCLIENTE);
        return result;
    }
```

Figura 8

En el proyecto esta capa se definirá como una biblioteca de clases donde se colocara una clase por cada entidad que se encuentre en la base de datos newsoftv

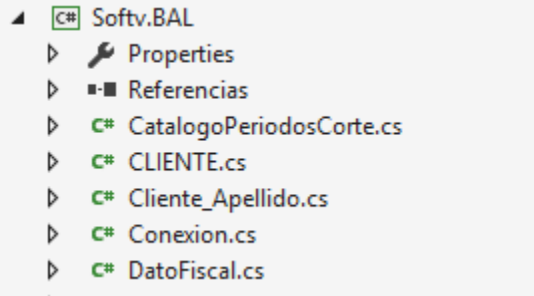


Figura 9

2.-Providers

Las clases de tipo providers se encargan de la conexión de datos entre el servicio y la capa de BAL y capa SQL, el cual valida que la información que se manda al usuario final sea integra e integra clases abstractas para propósitos de herencia.

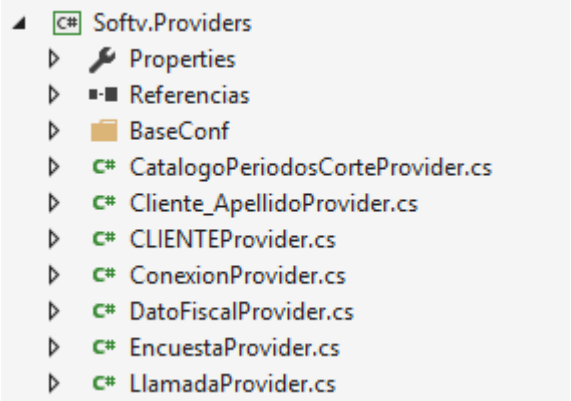


Figura 10

```
/// <summary>
/// Abstract method to delete CatalogoPeriodosCorte
/// </summary>
public abstract int DeleteCatalogoPeriodosCorte(int? Clv_Periodo);
```

Figura 11

3.- Capa SQL

La biblioteca de clases tipo SQL contienen la definición de la conexión de la base de datos, en este proyecto se instanciaran los procedimientos almacenados de la base de datos, gracias a estas clases los datos son enviados hacia el cliente final.

```
public override int AddCatalogoPeriodosCorte(CatalogoPeriodosCorteEntity entity_CatalogoPeriodosCorte)
{
int result=0;
using(SqlConnection connection = new SqlConnection(SoftvSettings.Settings.CatalogoPeriodosCorte.ConnectionString))
{

SqlCommand comandoSql = CreateCommand("Softv_CatalogoPeriodosCorteAdd", connection);

    AssingParameter(comandoSql, "@Clv_Periodo", null, pd: ParameterDirection.Output, IsKey: true);

        AssingParameter(comandoSql,"@Descripcion",entity_CatalogoPeriodosCorte.Descripcion);

        AssingParameter(comandoSql,"@Habilitar",entity_CatalogoPeriodosCorte.Habilitar);

try
```

Figura 12

Figura 14 Ejemplo capa SQL

4.- Entities

Biblioteca de clases el cual se definen los atributos y relaciones de las entidades de la base de datos Su utilidad es la accesibilidad de los datos con EntityFrameWork 5 lo que optimiza el rendimiento y la búsqueda de información con Linq.

Figura 15 Ejemplo

```
public class CatalogoPeriodosCorteEntity : BaseEntity
{
#region Attributes

  /// <summary>
  /// Property Clv_Periodo
  /// </summary>
  [DataMember]
  public int? Clv_Periodo { get; set; }
  /// <summary>
  /// Property Descripcion
  /// </summary>
  [DataMember]
  public String Descripcion { get; set; }
  /// <summary>
  /// Property Habilitar
  /// </summary>
  [DataMember]
  public int? Habilitar { get; set; }
#endregion
}
```

Figura 13

Services

La capa final donde la información que paso en las capas anteriores es mostrada como servicio REST hacia el cliente para consumo de información esta información se mandara al cliente en formato JSON para ser consumida de manera eficaz.

```
public class SoftvWCFService : IBanco
{

    #region Banco
    public BancoEntity GetBanco(int? IdBanco)
    {
        return Banco.GetOne(IdBanco);
    }

    public BancoEntity GetDeepBanco(int? IdBanco)
    {
        return Banco.GetOneDeep(IdBanco);
    }

    public IEnumerable<BancoEntity> GetBancoList()
    {
        return Banco.GetAll();
    }
```

Figura 14

Interacción entre capas

la interacción entrecapas se realiza cuando el cliente realiza una petición al servicio REST ,los datos pasan por la capa BAL donde son procesados para la recepción en la capa de providers,después estos pasan por la capa data la conexión aquí es donde se envían los datos a los procedimientos almacenados de la base de datos ,la base de datos retorna una respuesta de inserción de valores.

La respuesta retorna de manera viceversa hasta llegar al servicio donde se recibe una respuesta OK, o BAD REQUEST según la respuesta dl servicio retornado en formato Json donde en el cliente se transformara.

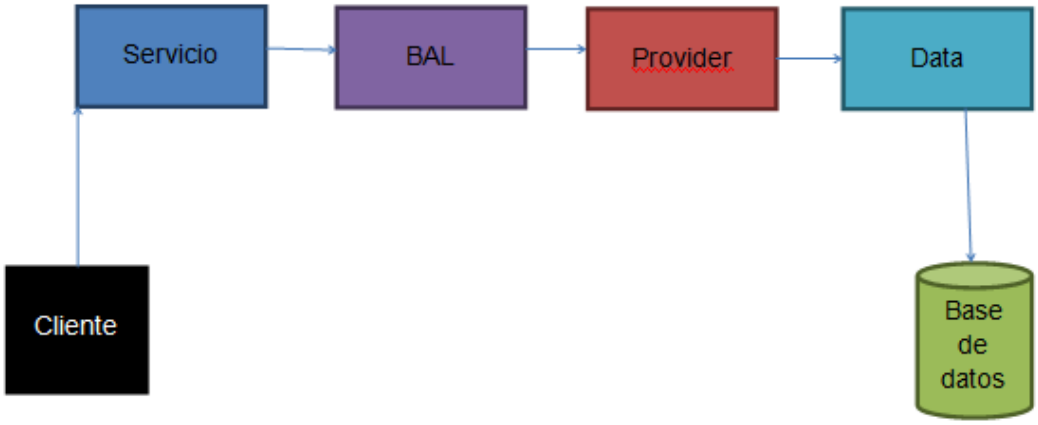


Figura 15

El siguiente esquema muestra el proceso de interacción de las diferentes capas del sistema hasta llegar al usuario final

ESTRUCTURA DE CLIENTE (vistas)

Las vistas de este proyecto están codificadas en HTML5 Y Angular JS, JavaScript
En el diseño está apoyado por el framework Boostrap, Angular JS consumirá los servicios REST que ya se han creado, angular es desarrollado por Google, en los últimos años este framework ha tenido gran impacto en los desarrolladores frontend gracias a su simpleza para codificar e interactuar con el DOM.

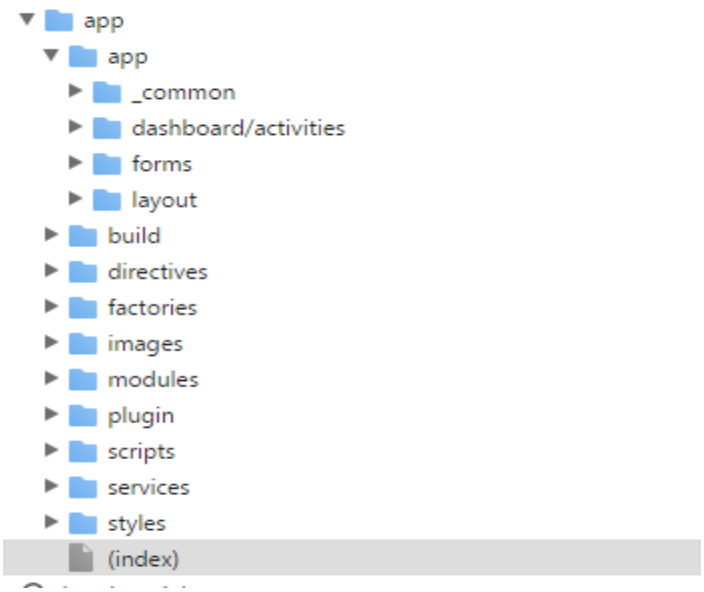
Anteriormente en la parte Front-End de las aplicaciones web sólo se tenía a jQuery (además de otras librerías parecidas como Mootools, Prototype,…) para ayudarnos con el código JavaScript del cliente. Podíamos manipular el DOM de una forma más sencilla, añadir efectos, llamadas AJAX, etc… pero no SE TENIA un patrón a seguir. Todo el código JS iba en funciones que íbamos creando según se necesitara, lo que provocaba que con el tiempo el código fuera difícilmente manejable.

Angular no tiene necesidad de ser instalado ya que cuenta con un CDN que puede ser consumido por cualquier aplicación web

Boostrap es el framework por excelencia por los diseñadores frontend ,Boostrap tiene definido cientos de clases y estilos que hacen el diseño más fácil para desarrolladores no tan expertos ,su sistema de maquetado y su web Responsive Design hace que cualquier aplicación diseñada por Boostrap sea vista desde cualquier dispositivo de cualquier resolución.

ES de Código abierto para diseño de sitios y aplicaciones web. Contiene plantillas de diseño con tipografía, formularios, botones, cuadros, menús de navegación y otros elementos de diseño basado en HTML y CSS, así como, extensiones de JavaScript opcionales adicionales.

Estructura de las vistas



Estructura de las vistas del proyecto

Figura 16

La imagen anterior muestra la estructura del proyecto por el lado del cliente donde los servicios creados serán consumidos, como angular tiene un estilo de una sola vista en

el archivo index será la plantilla base para todas las vistas esta podrá heredar estilos a las demás páginas, además se colocaran aquí todos los scripts necesarios de controladores de angular y referencia a todos los plugins necesarios.

Interacción entre capas MVC de Angular Js

GlobalService: Proporciona la conexión entre el cliente y el servicio

Factories: Es el puente del servicio y los controladores

Controllers: Contiene la definición de todas las funciones del controlador especifico e interacciona con las vistas.

Directiva: Proporciona la interacción con el DOM definiendo reglas, permite agregar pedazos de código o hasta una funcionalidad completa.

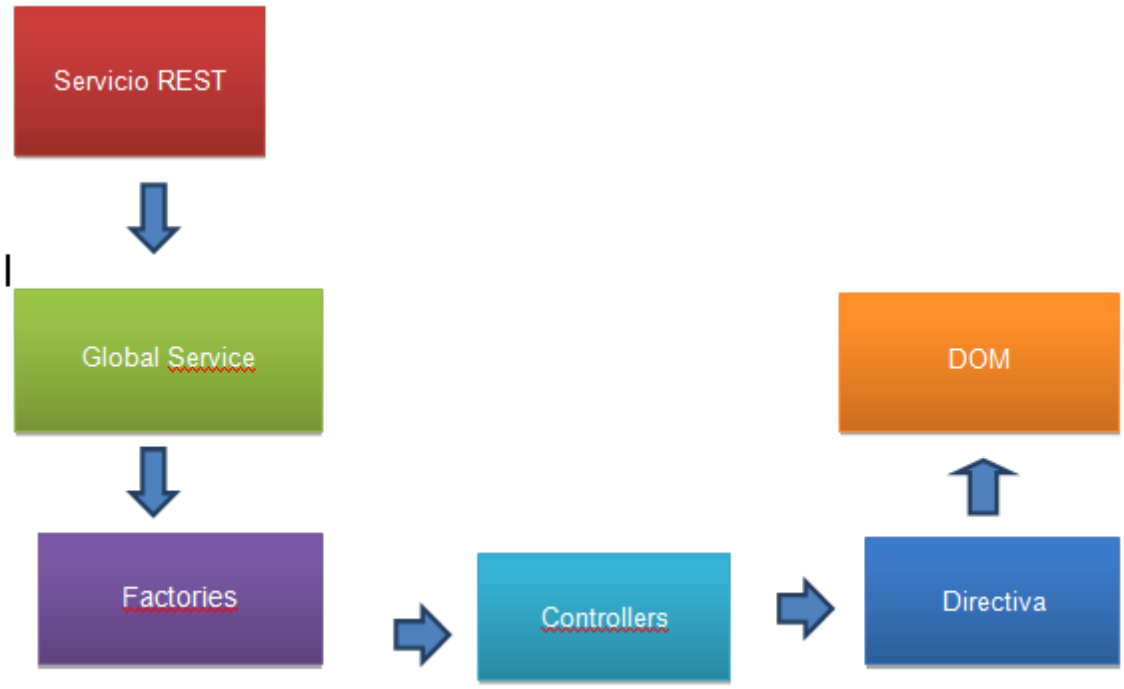


Figura 17

Global service.js

Los Servicios en angular son declarados de la forma **app.service** y son inyectados en los controladores u otros servicios o factorías. Un servicio se declara de la siguiente manera:

```
'use strict';

/**
 * @ngdoc service
 * @name app.GlobalSerivce
 * @description
 * # GlobalSerivce
 * Service in the app.
 */
angular.module('app').service('GlobalService', [function () {
        var svc = {};

        svc.getUrl = function (){
            //return "http://52.35.23.247:64481/SoftvWCFService.svc/";
            return "http://softvaws.zoga.com.mx/SoftvWCFService.svc/";
        };

        return svc;
    }]);
```

Figura 18

En este archivo Javascript se declara el servicio global que consumirá el servicio Rest

Se puede observar la IP del servidor es retornada en una función .la cual será consumida a su vez por los factories.

Factories

Las factorías en AngularJS son declaradas de forma **app.factory** y al igual que los servicios estos pueden tener dependencias de otros servicios y/ factorías. Unafactoría se declara de la siguiente manera:

```
angular.module('app').factory('CitiesFactory', function ($http, $q, GlobalService, $base64, User) {
    var factory = {};
    var paths = {
        list: "Municipio/GetMunicipioList",
        detail: "Municipio/GetDeepMunicipio",
        add: "Municipio/AddRelEstMunL",
        update: "Municipio/UpdateRelEstMunL",
        remove: "Municipio/DeleteMunicipio",
        citiesFromState: "RelMunicipioEst/GetEstadosRelMun"
    };
```

Figura 19

En el script anterior podemos ver la declaración de un Factory en este caso el Factory de ciudades, se puede observar también que el servicio global es heredado y pasado al Factory.

La variable paths contiene todas las definiciones del servicio y son utilizadas en funciones de ese factorie para pasar datos a los controladores

En el siguiente script se pude observar como en la funciónGetList se obtiene mediante un servicio $http donde se pide al global service la IP del servidor, al objeto paths la ruta GetListSeguido de los parámetros que se van a enviar al servicio, en este caso no se enviaran parámetros ya que solo se obtendrá una lista de ciudades.

```
factory.getList = function () {
    var deferred = $q.defer();
    var Parametros = {};
    var config = {
        headers: {
            'Authorization': User.getToken()
        }
    };
    $http.post(GlobalService.getUrl() + paths.list, JSON.stringify(Parametros), config).success(function (data) {
        deferred.resolve(data.GetMunicipioListResult);
    }).error(function (data) {
        deferred.reject(data);
    });
    return deferred.promise;
};
```

Figura 20

Por ultimo si el servicio retorna respuesta satisfactoria se obtendrá la respuesta del servicio Que retornara los listas en formato JSON

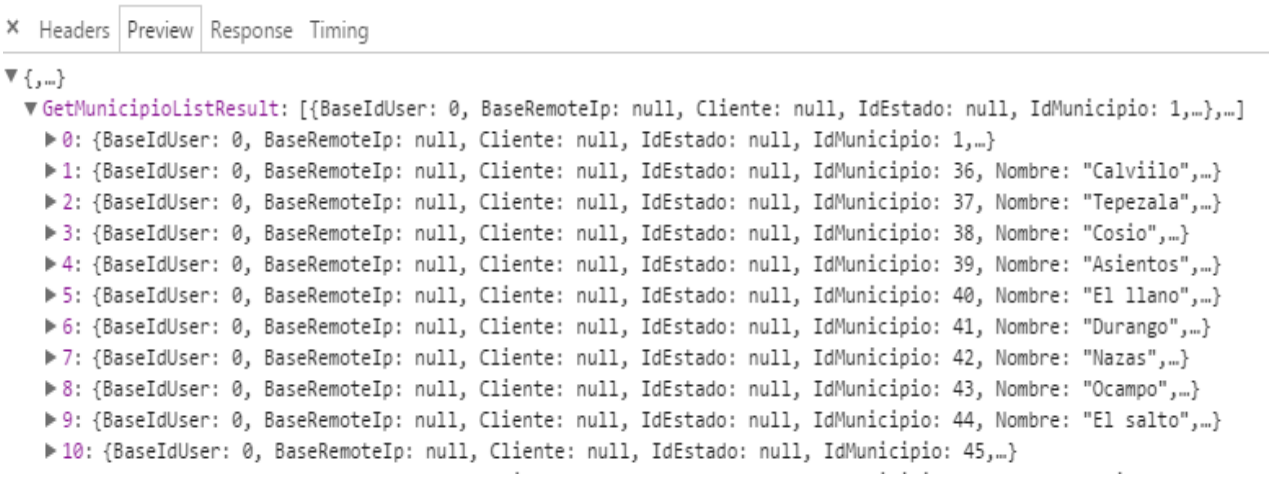


Figura 21

Ejemplo de respuesta del servicio

Controllers

Los controladores en AngularJS se encargar de controlar los datos de las aplicaciones AngularJS. Los controladores son Objetos JavaScript**.**

Las aplicaciones AngularJS son controladas por los controladores. Un controlador no es más que un objeto JavaScript, que se crea con el constructor de objetos de JavaScript. El ámbito de la aplicación está definido por`$scope` y se corresponde con el elemento HTML asociado a la aplicación

```
angular.module('app.catalogs').controller('CitiesCtrl', function ($rootScope, $scope, $timeout, notificationService, CitiesFactory,
    var states = null;
    $scope.dataTable = {
        headers: [{
                "title": "ID",
                "key": "IdMunicipio",
                "class": "text-center"
            }, {
                "title": "Nombre de la Ciudad / Municipio",
                "key": "Nombre",
                "class": "text-left"
            }],
```

Figura 22

```
var getList = function () {
    load.init();
    CitiesFactory.getList().then(function (data) {
        angular.forEach(data, function (value) {
            $scope.entities[value.IdMunicipio] = value;
        });
        $scope.dataTable.dataRows = data;
        load.remove();

    });
};
```

Figura 23

Ejemplo de una función en un controlador

El script anterior se muestra como el controlador acceso a los factories y recupera la información de la función Getlist, en esta última capa los datos pueden ser ahora desplegados hacia el usuario final para su integración

Mostrando datos al usuario

```
<!-- catalogos -->
<script src="modules/catalogs/module.js"></script>
<script src="modules/catalogs/states/controllers/StatesCtrl.js"></script>
<script src="modules/catalogs/states/controllers/ModalDataStateCtrl.js"></script>
<script src="modules/catalogs/cities/controllers/CitiesCtrl.js"></script>
<script src="modules/catalogs/cities/controllers/ModalDataCityCtrl.js"></script>
<script src="modules/catalogs/square/controllers/SquareCtrl.js"></script>
```

Figura 24

Se hace referencia de los archivos JavaScriptcreados. Se coloca el nombre del controlador por **qué** se va a hacer referencia en la vista /archivo HTML

Este puede ser en cualquier parte del archivo HTML, en este ejemplo la directiva ng-clickTraerá los datos del controlador hacia la vista.

```
<div class="col-xs-12">
<div class="box box-success">
            <div class="box-header with-border" ng-controller="CitiesController">
              <h3 class="box-title"><i class="fa fa-tag"></i> Administración de encuestas</h3>
            </div>
            <div class="box-body">
              <h1>Pagina de encuestas</h1>

                <p>{{ message }}</p>
            </div>

    <div ng-controller="Ctrl">
        <button ng-click="addData()">Add Data</button>
         <h2>This table uses columns defined in controller</h2>
        <table my-table options="options" class="table-bordered"></table>
    </div>
```

Figura 25

Aquí termina la interacción de las capas de Angular JS como podemos ver angular hace que el desarrollo del frontend sea rápido y con el modelo de desarrollo que implementa el orden de los scripts y funciones hacen que el código no termine en un completo desorden.

Deploy

La aplicación web será alojada en Amazon Web Services,Amazon Web Services ofrece servicios de cloudcomputing de confianza, escalables y económicos. Se compró un servidor

Con Windows Server 2014 y IIS 8 el deploy es muy parecido a una publicación web cotidiana solo con la diferencia que puede ser accesado en cualquier parte

Figura 26

Figura 27

Resultados

Catálogo de ciudades

El catálogo de ciudades del sistema anterior tenía un diseño muy malo, las características de mejora son que ahora la versión web contiene un filtrado de ciudades y cuanta con una tabal principal donde la edición y eliminación son más sencillos para el usuario, se puede distinguir la mejoría a simple vista, se evita la necesidad de tantos clicks que son molestos para el usuario

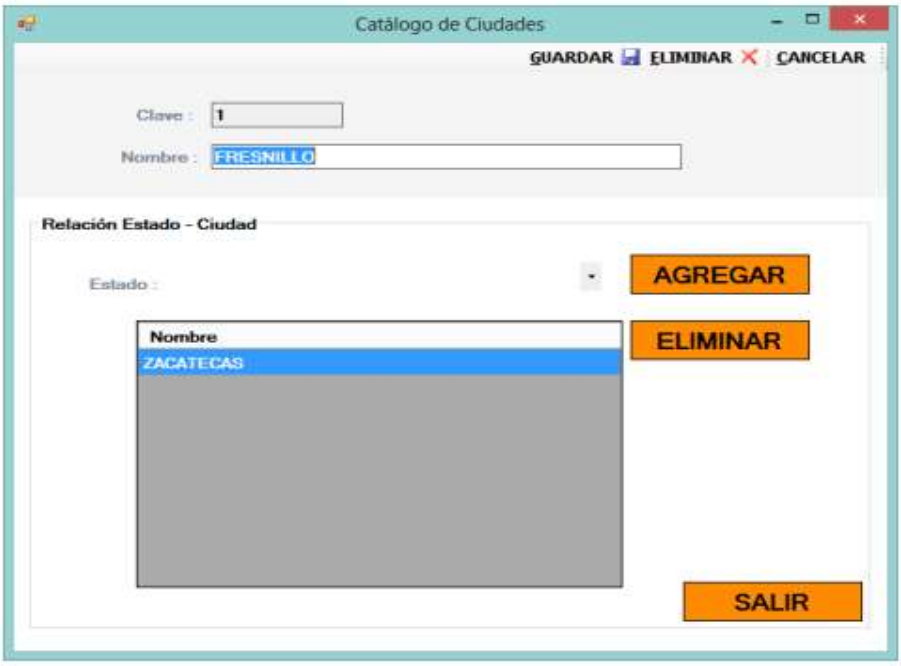


Figura 28

Catalogo de ciudades antes

Figura 29

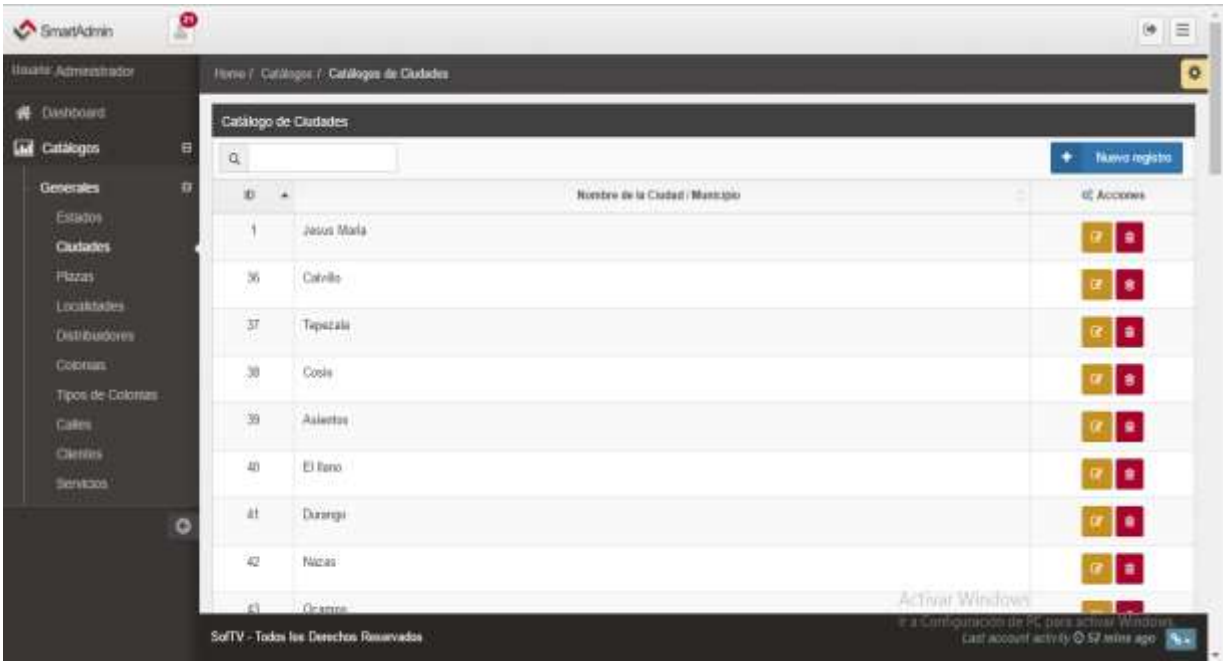


Figura 30

Catalogo de ciudades después

Catálogo de distribuidores

El catálogo de distribuidores se trató de ser lo más claro posible para el usuario ya que se agregaron la ayuda de modales en Boostrap para mostrar la información másdetallada, además que no existía una tabla de contenidos ni se podía filtrar.

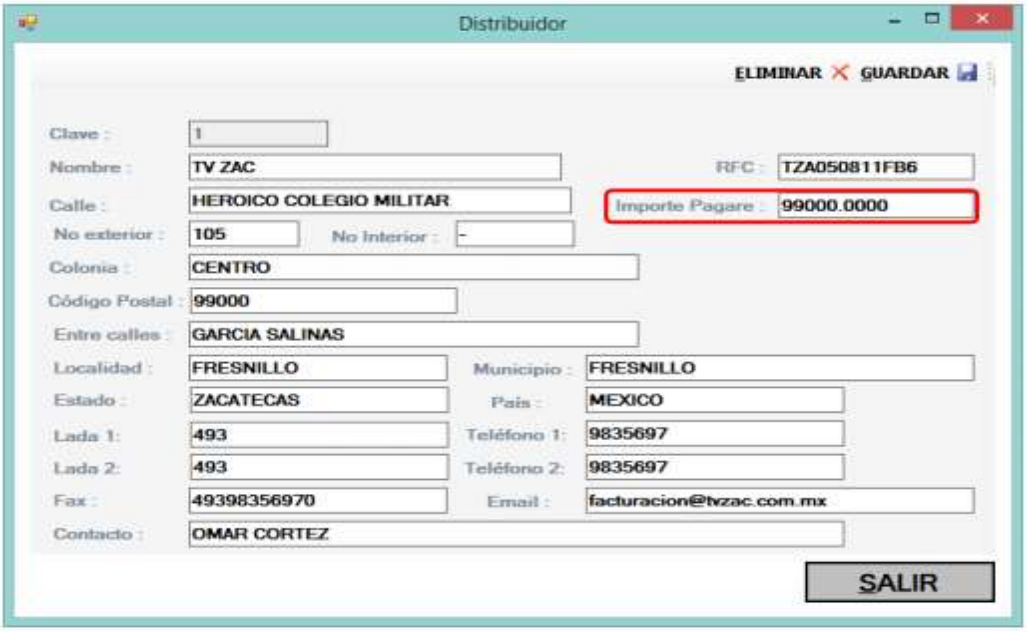


Figura 31

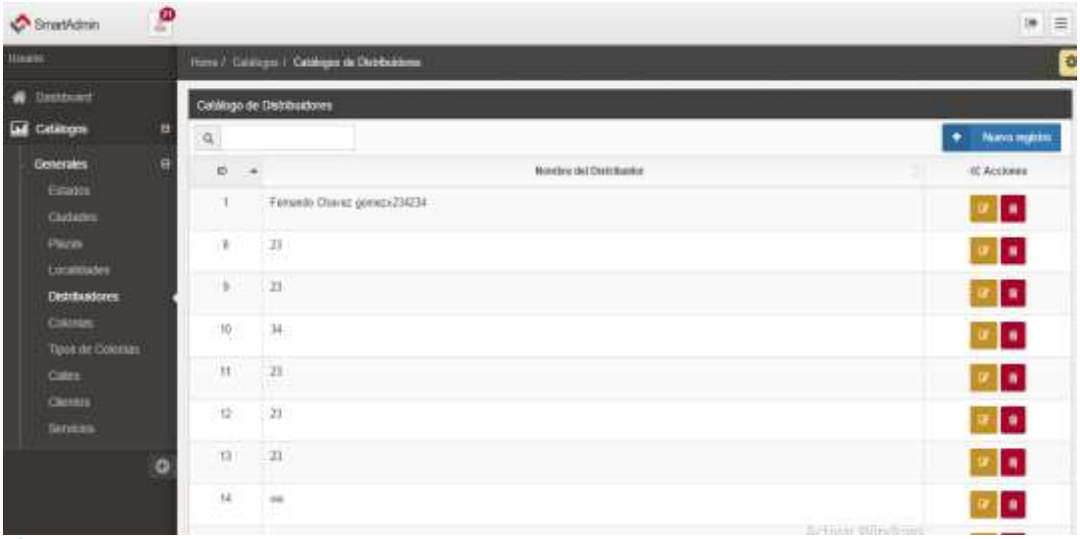


Figura 32

Catálogo de localidades

De igual manera el catálogo de localidades no contaba con un catálogo que pudiera ser visualizado por el usuario, la edición tardaba en procesarse bastante tiempo

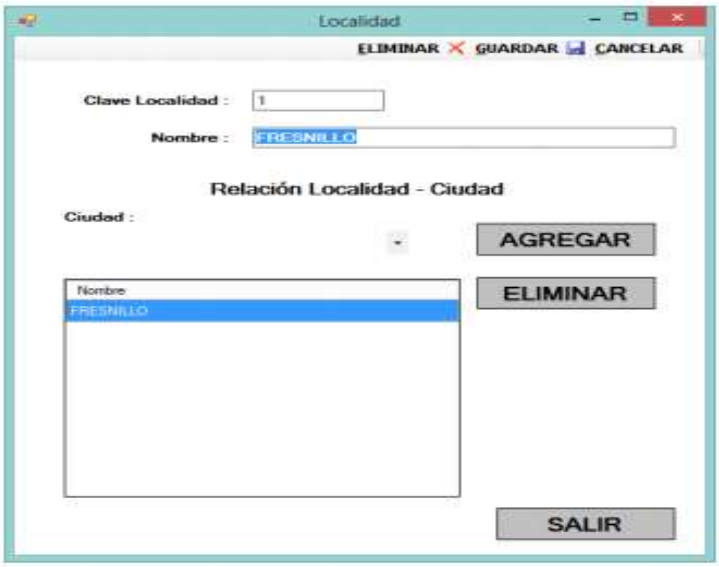


Figura 33

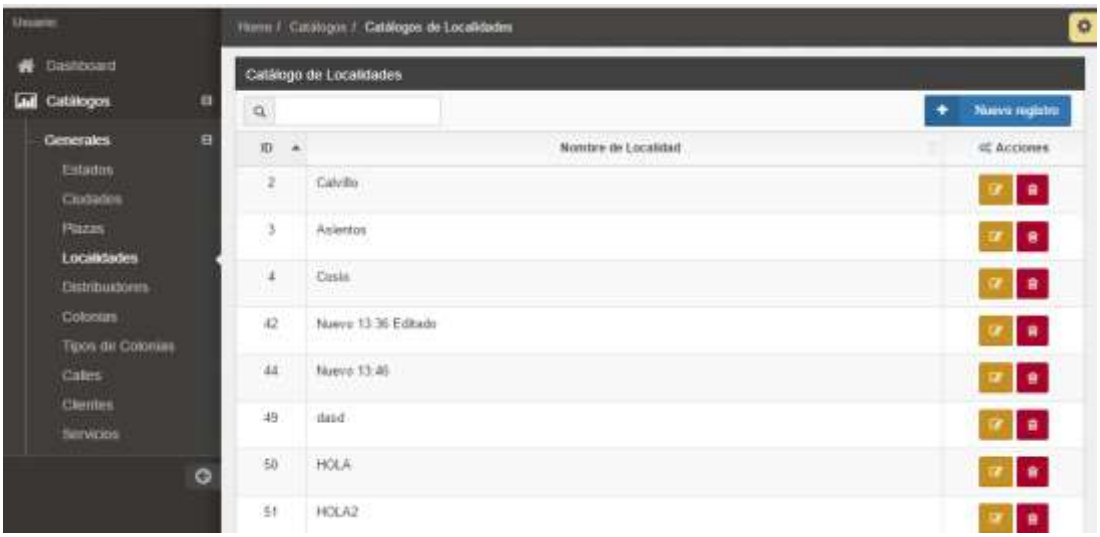


Figura 34

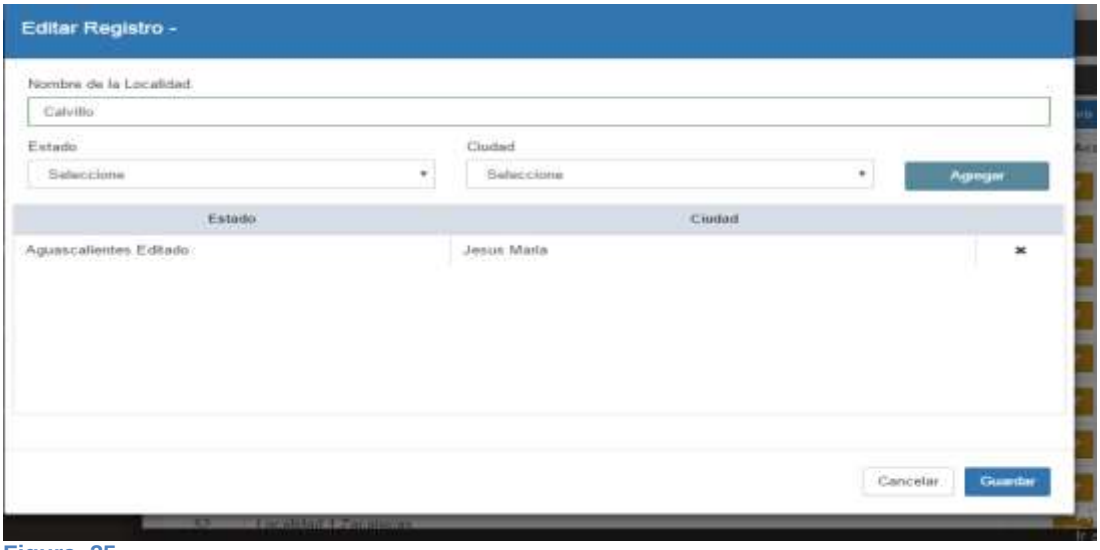


Figura 35

Catalogó de colonias

En el cátalo de colonias se trató de mantener el mismo diseño que en anterior ya que los usuarios les había gustado la forma de trabajar, se implementó una tabla con paginación donde se filtrara y se podrá editar cada registro

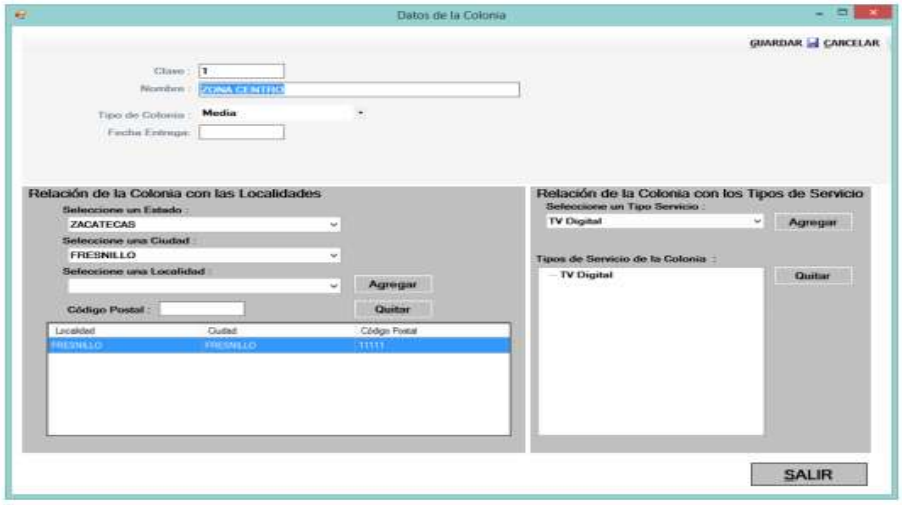


Figura 36

Figura 37

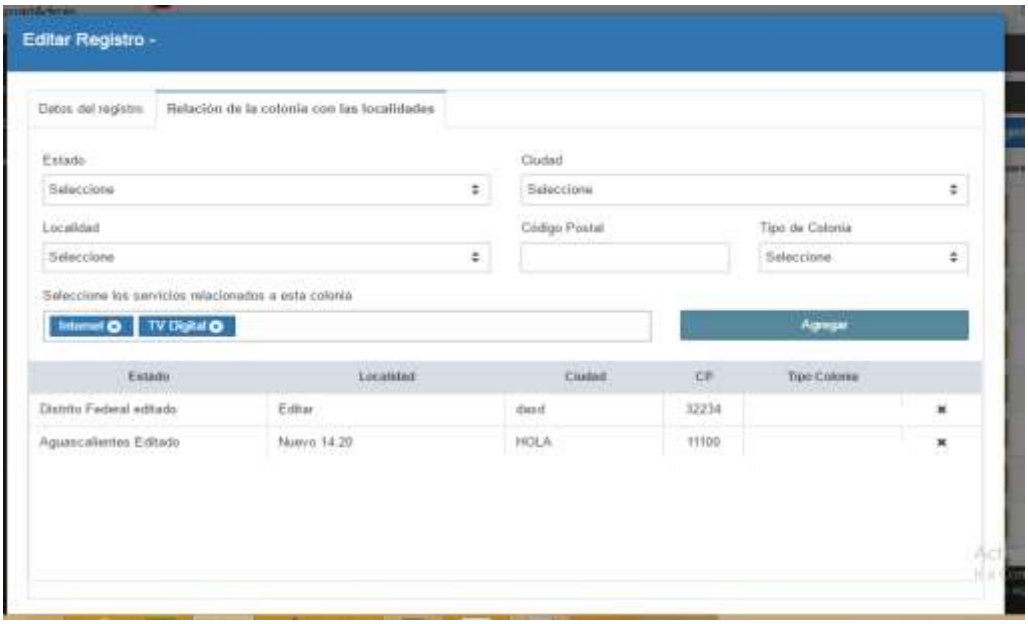


Figura 38

Catálogo de clientes

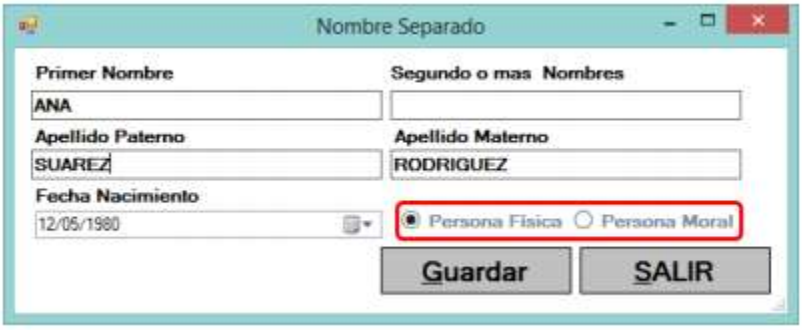


Figura 40

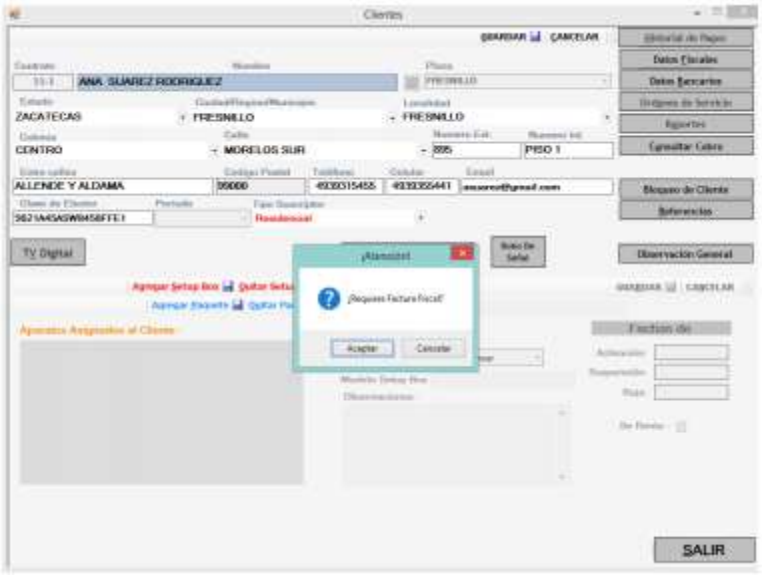


Figura 39

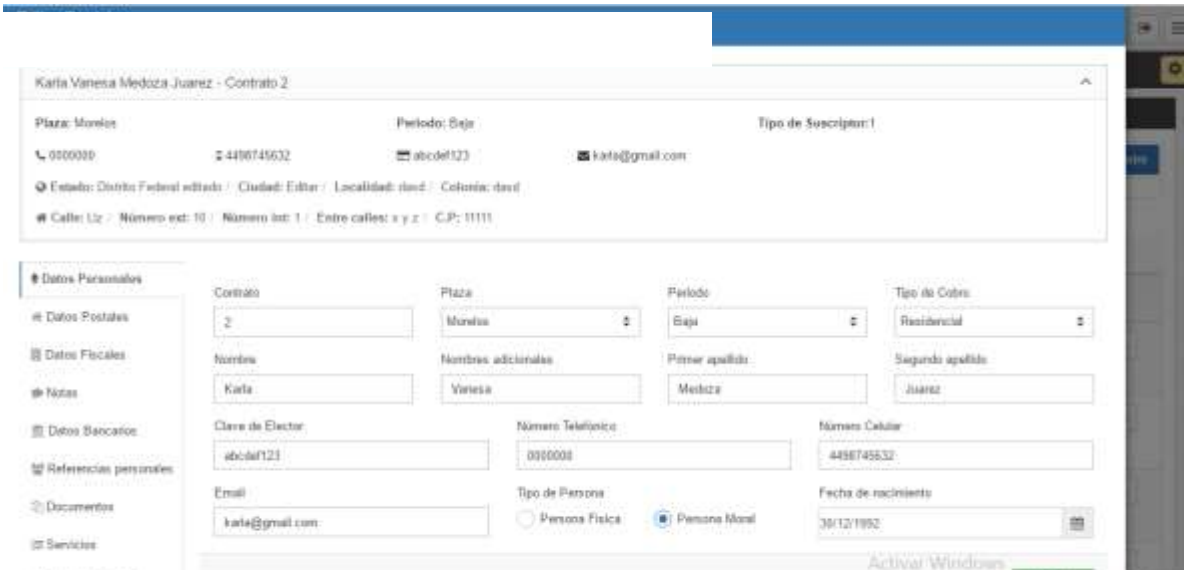


Figura 41

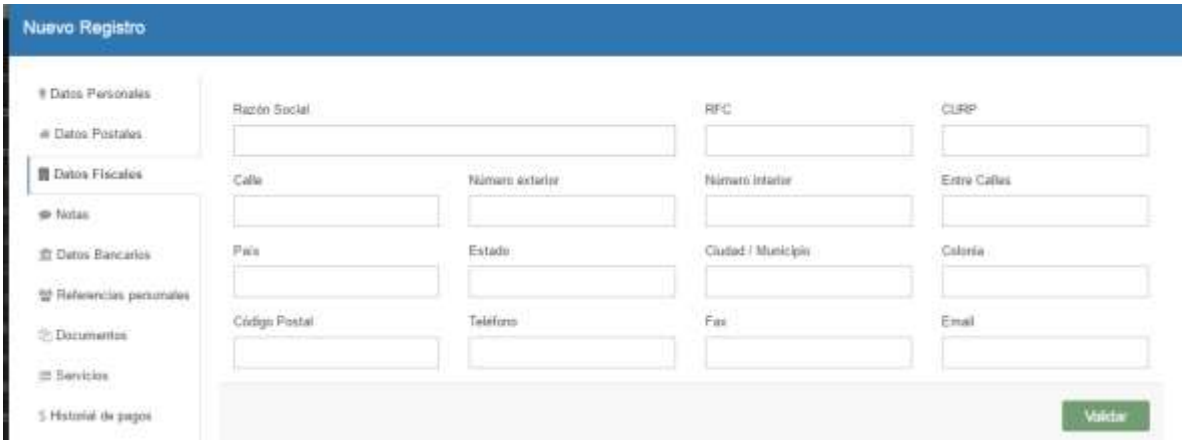


Figura 42

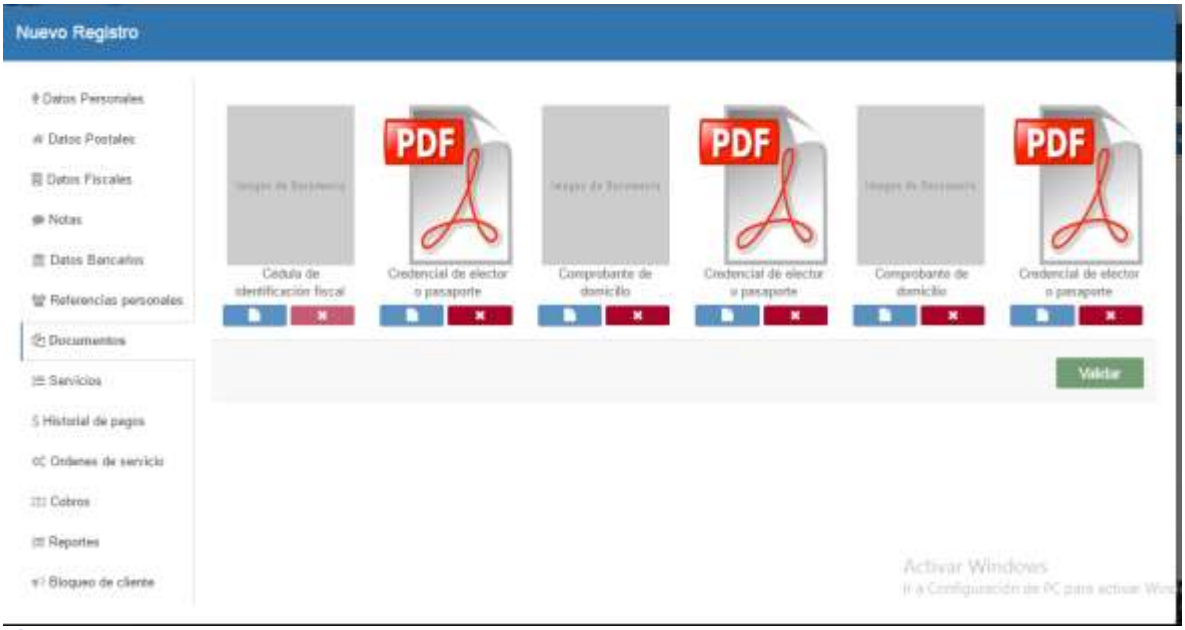


Figura 43

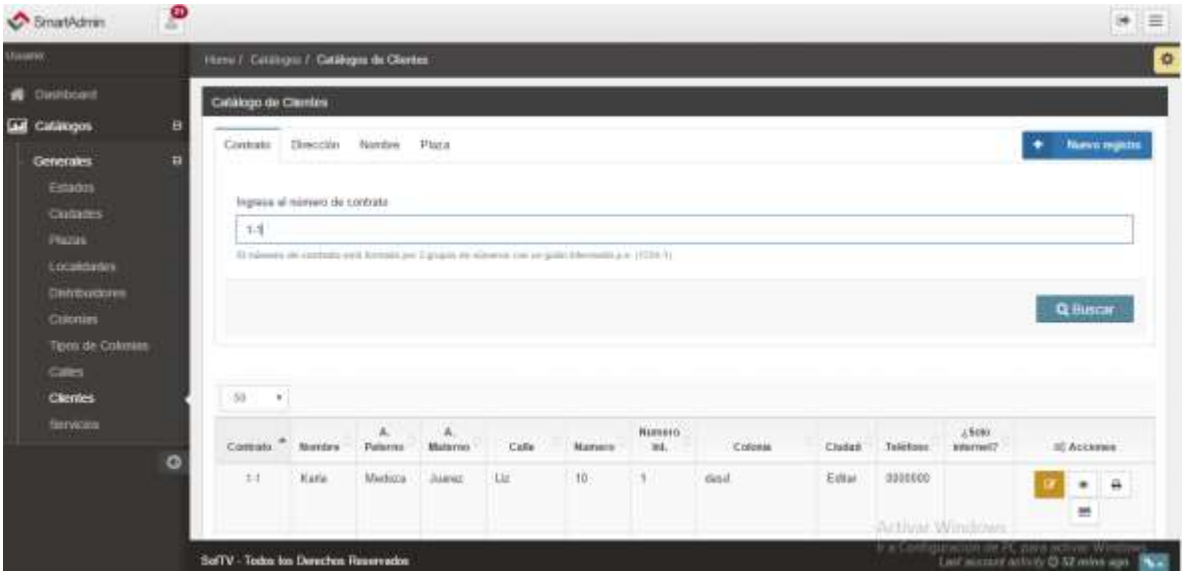


Figura 44

Catálogo de servicios

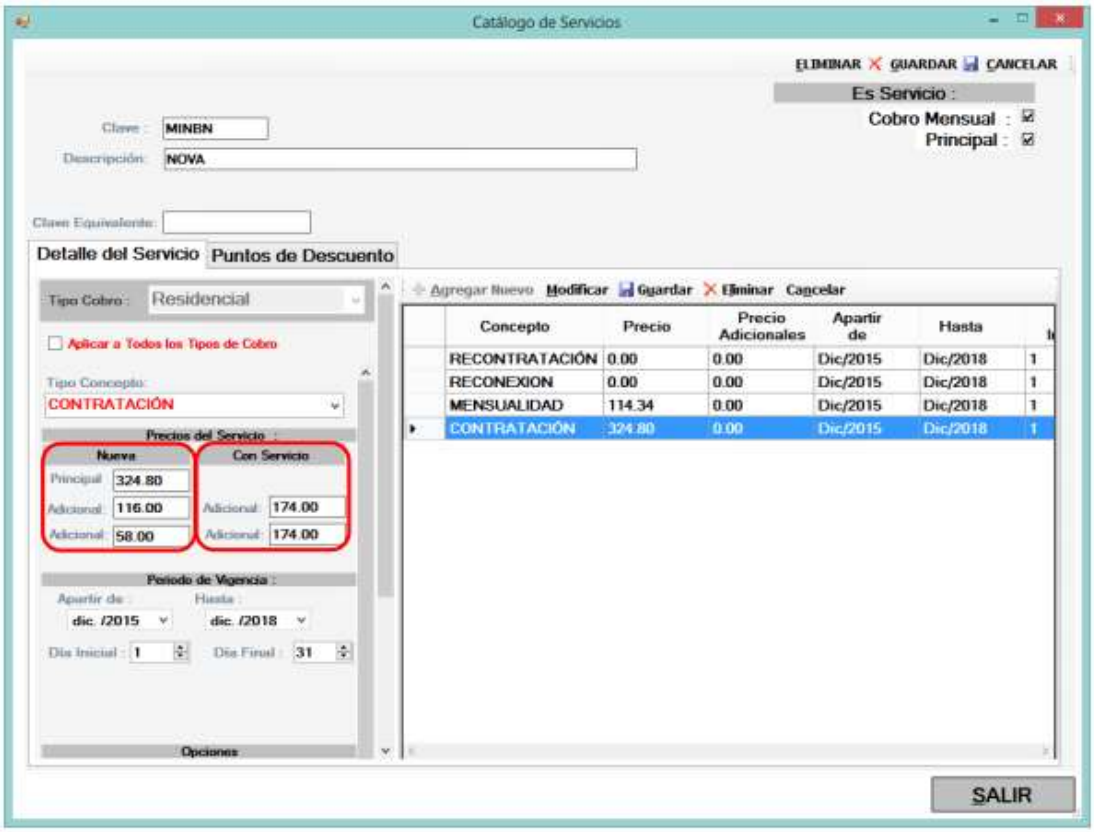


Figura 45

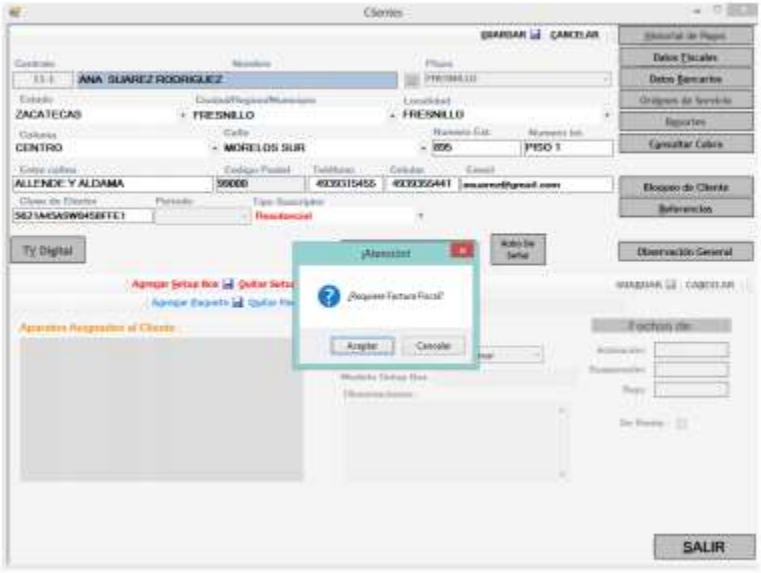


Figura 46

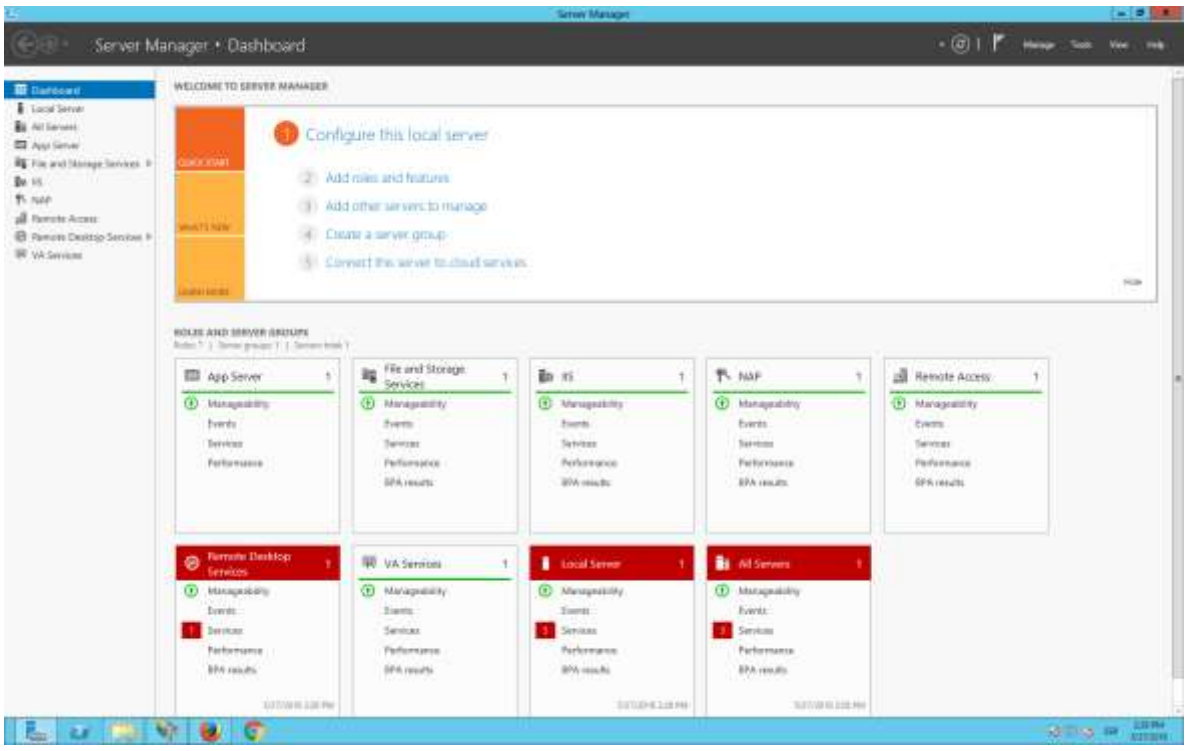


Figura 47

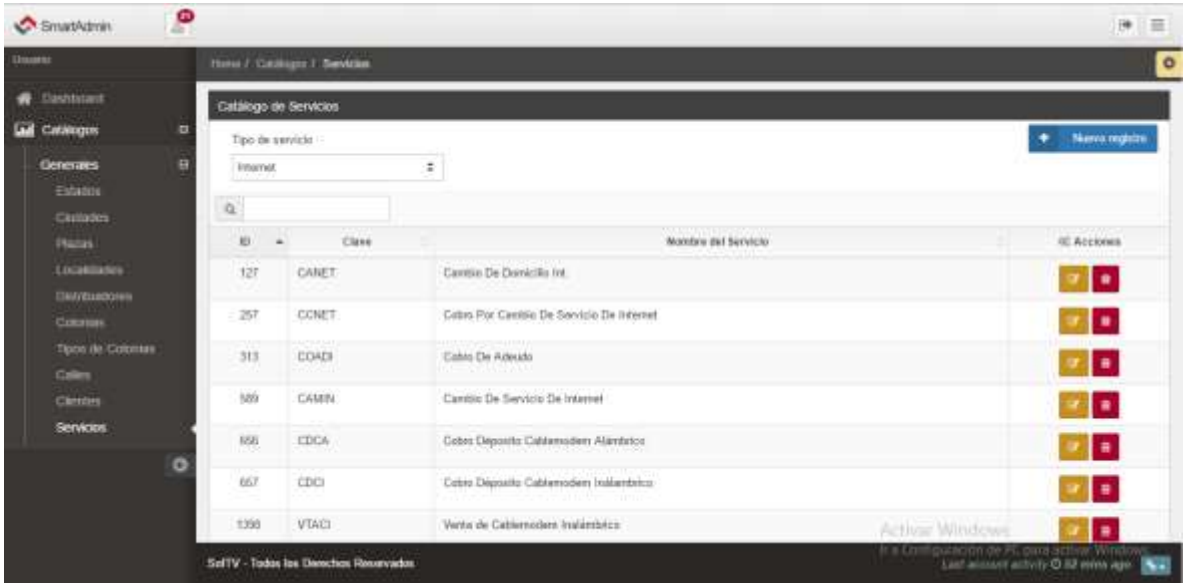


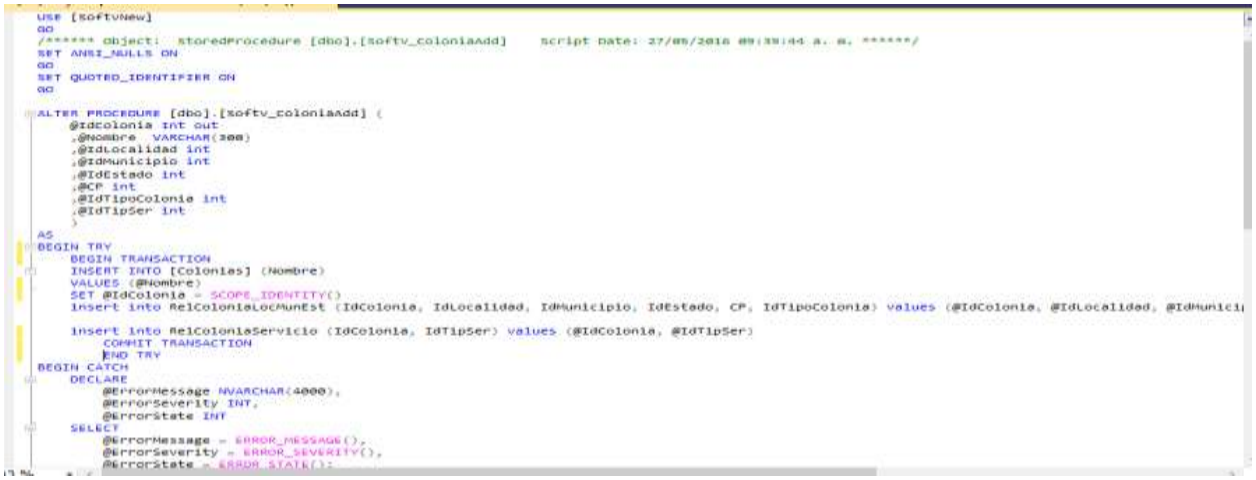
Figura 48

Figura 49

Base de datos NEWSOFTV

En la base de datos de Newsoftv se realizaron los siguientes scripts estos hacen las funciones básicas de CREATE, UPDATE, DELETE, GETLIST

Ejemplo de procedimientos almacenados Colonia_ADD



```
USE [SoftvNew]
GO
/****** Object:  StoredProcedure [dbo].[softv_coloniaAdd]    Script Date: 27/05/2016 09:39:44 a. m. ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO

ALTER PROCEDURE [dbo].[softv_coloniaAdd] (
    @IdColonia INT out
    ,@Nombre  VARCHAR(300)
    ,@IdLocalidad int
    ,@IdMunicipio int
    ,@IdEstado int
    ,@CP int
    ,@IdTipoColonia int
    ,@IdTipSer int
    )
AS
BEGIN TRY
    BEGIN TRANSACTION
    INSERT INTO [Colonias] (Nombre)
    VALUES (@Nombre)
    SET @IdColonia = SCOPE_IDENTITY()
    Insert into RelColoniaLocMunEst (IdColonia, IdLocalidad, IdMunicipio, IdEstado, CP, IdTipoColonia) values (@IdColonia, @IdLocalidad, @IdMunici

    Insert into RelColoniaServicio (IdColonia, IdTipSer) values (@IdColonia, @IdTipSer)
        COMMIT TRANSACTION
        END TRY
BEGIN CATCH
    DECLARE
        @ErrorMessage NVARCHAR(4000),
        @ErrorSeverity INT,
        @ErrorState INT
    SELECT
        @ErrorMessage = ERROR_MESSAGE(),
        @ErrorSeverity = ERROR_SEVERITY(),
        @ErrorState = ERROR_STATE();
```

Figura 50

```
USE [SoftvNew]
GO
/****** Object:  StoredProcedure [dbo].[softv_CNREdit]    Script Date: 27/05/2016 09:42:51 a. m. ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO


ALTER PROCEDURE [dbo].[Softv_CNREdit] (
    @IdConsecutivo BigInt
    ,@IdContrato BigInt
    ,@IDUnicaNet BigInt
    ,@MacAddress  VARCHAR(50)
    ,@Paquete  VARCHAR(250)
    ,@Comando  VARCHAR(50)
    ,@Resultado Int
    ,@Descripcion_transaccion  VARCHAR(max)
    ,@Fechaejecucion DateTime
    ,@IdOrden BigInt
    ,@FechaHabilitar DateTime
    )
AS
UPDATE CNR
SET IdContrato = @IdContrato
    ,IDUnicaNet = @IDUnicaNet
    ,MacAddress = @MacAddress
    ,Paquete = @Paquete
    ,Comando = @Comando
    ,Resultado = @Resultado
    ,Descripcion_transaccion = @Descripcion_transaccion
    ,Fechaejecucion = @Fechaejecucion
    ,IdOrden = @IdOrden
    ,FechaHabilitar = @FechaHabilitar
WHERE IdConsecutivo = @IdConsecutivo
```

Figura 51

Las tablas de SQL SERVER ahora están bien estructuradas, se eliminaron las tablas innecesarias .

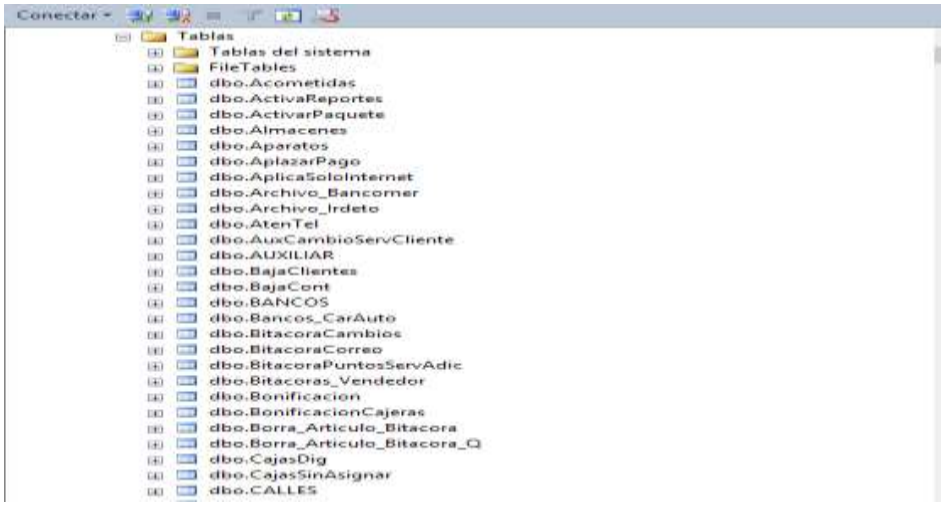


Figura 52

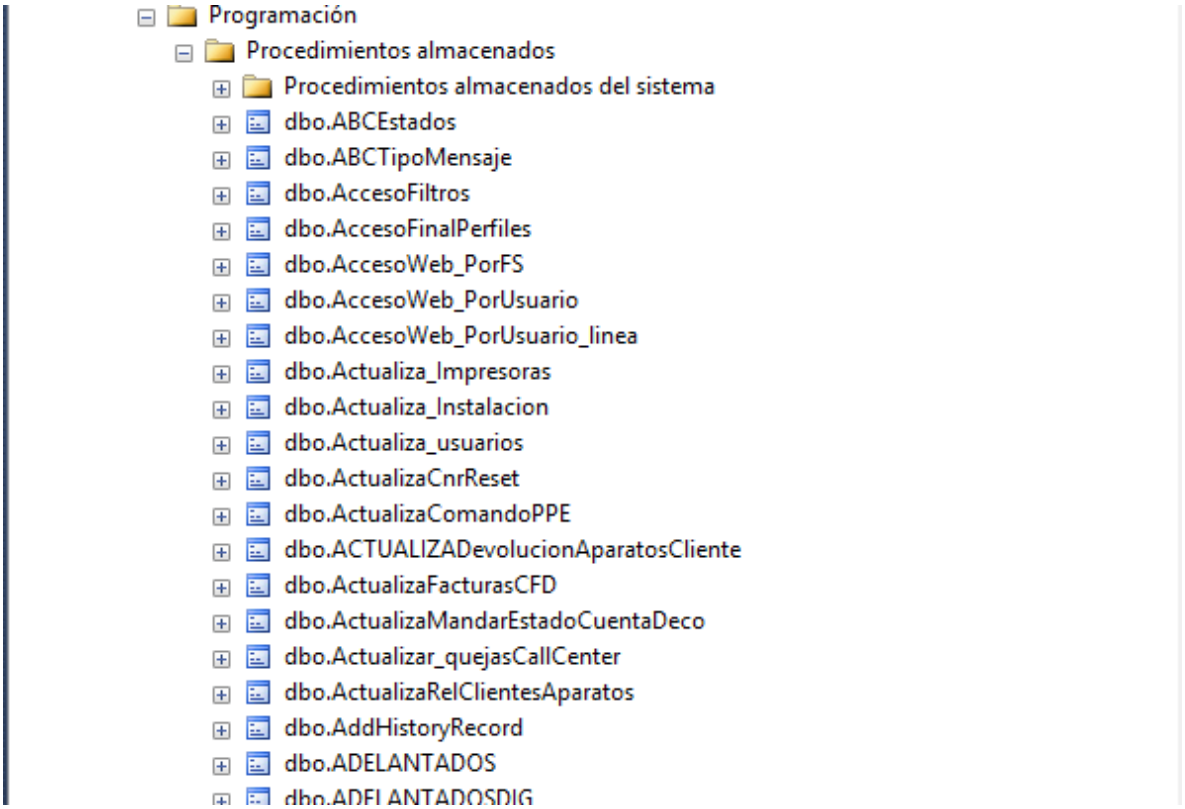


Figura 53

Servicio REST

el servicio REST se realizo en visual estudio se crearon librerías de clase para formar las capas del modelo de programación

Figura 54

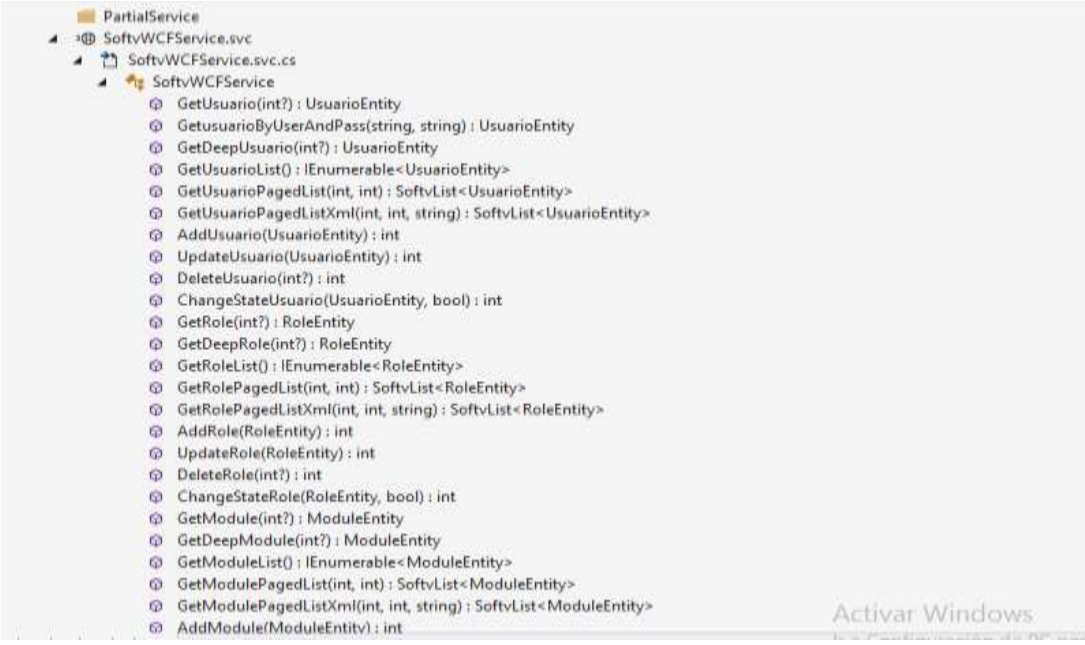


Figura 55

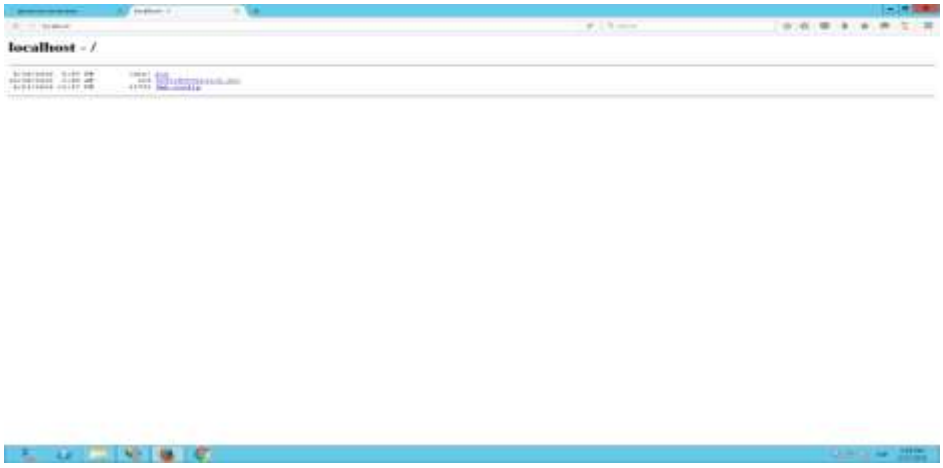


Figura 56

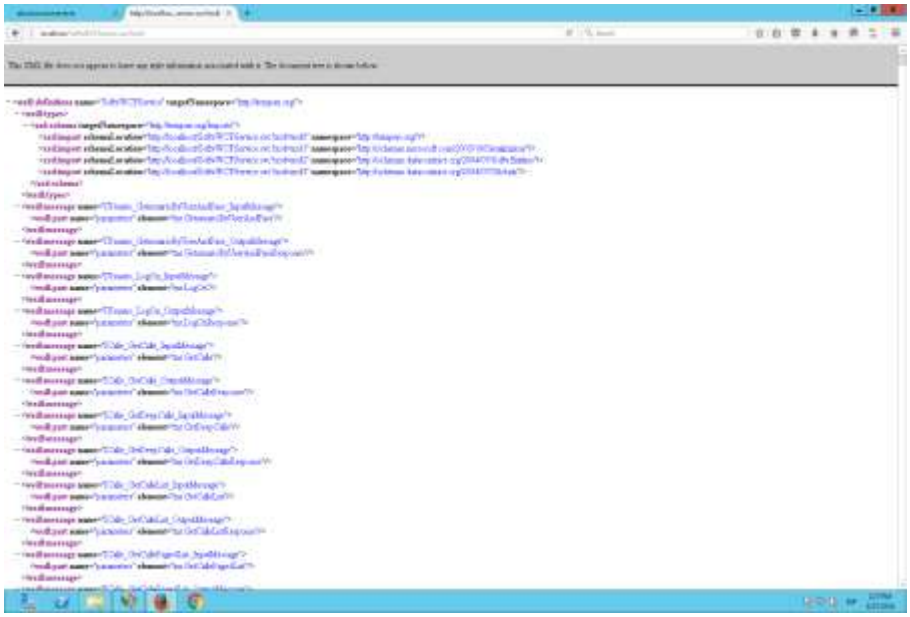


Figura 57

Conclusiones

- la planeación en un proyecto inicial es vital y crucial para la vida del software y su ciclo vital no acabe tan rápido, se debe de revisar la estructura de bases de datos con la finalidad de no hacer tan extensa la información y el espacio de la base de datos se ocupe en registros sin importancia.

- el uso de un control de versiones como Github optimiza el desarrollo de software, ya que implementa el trabajo colaborativo de manera eficiente, ayuda al ahorro de tiempo y errores de código y evita el mal manejo de tareas.

- las nuevas tecnologías web que están surgiendo al paso de los años son una gran herramienta que debemos aprender a utilizar ya que mejoran de una gran manera al desarrollo de software.

- el uso de modelo de capas es de gran ayuda para la optimización del código pero algunas veces puede llegar a ser molesto para desarrolladores que no están familiarizados.

- los servicios REST son muy fáciles de consumir y son bastantes fáciles de los, pero se requiere un conocimiento más avanzado para consumirlos de forma correcta.

- angular js sin duda es uno de los mejores frameworks para Javascript ,su manera de implementar el su modelo es una gran ayuda para la rápido desarrollo e interacción con el DOM, recomendable para próximos proyectos

Glosario

API

La interfaz de programación de aplicaciones, abreviada como API (del inglés: Application Programming Interface), es el conjunto de subrutinas, funciones y procedimientos (o métodos, en la programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción. Son usadas generalmente en las bibliotecas de programación.

MVC

El modelo–vista–controlador (MVC) es un patrón de arquitectura de software que separa los datos y la lógica de negocio de una aplicación de la interfaz de usuario y el módulo encargado de gestionar los eventos y las comunicaciones.

FRONTEND

En diseño de software el front-end es la parte del software que interactúa con el o los usuarios

DEPLOY

Son todas las actividades que hacen a un software disponible para su uso

JSON

JSON, acrónimo de JavaScript Object Notation, es un formato ligero para el intercambio de datos. JSON es un subconjunto de la notación literal de objetos de JavaScript que no requiere el uso de XML

DOM

Document Object Model o DOM ('Modelo de Objetos del Documento' o 'Modelo en Objetos para la Representación de Documentos') es esencialmente una interfaz de plataforma que proporciona un conjunto estándar de objetos para representar documentos HTML, XHTML y XML.

XML

eXtensible Markup Language ("lenguaje de marcas Extensible"), es un lenguaje de marcas desarrollado por el World Wide Web Consortium (W3C) utilizado para almacenar datos en forma legible.

GIT

es un software de control de versiones diseñado por Linus Torvalds, pensando en la eficiencia y la confiabilidad del mantenimiento de versiones de aplicaciones cuando éstas tienen un gran número de archivos de código fuente.

ENTITIES

En bases de datos, una entidad es la representación de un objeto o concepto del mundo real que se describe en una base de datos.
Una entidad se describe en la estructura de la base de datos empleando un modelo de datos.

FRAMEWORK

Un conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problemática particular que sirve como referencia, para enfrentar y resolver nuevos problemas de índole similar.

RESPONSIVE DESIGN

Es una filosofía de diseño y desarrollo cuyo objetivo es adaptar la apariencia de las páginas web al dispositivo que se esté utilizando para visualizarlas. Hoy día las páginas web se visualizan en multitud de dispositivos.

MIGRACIÓN

Consiste en la transferencia de materiales digitales de un origen de datos a otro, transformando la forma lógica del ente digital de modo que el objeto conceptual pueda ser restituido o presentado por un nuevo equipo o programa informático.

SOAP

Es un protocolo estándar que define cómo dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos XML.

WFC

Windows Communication Foundation (WCF) es un marco de trabajo para la creación de aplicaciones orientadas a servicios. Con WCF, es posible enviar datos como mensajes asincrónicos de un extremo de servicio a otro. Un extremo de servicio puede formar parte de un servicio disponible continuamente hospedado por IIS, o puede ser un servicio hospedado en una aplicación.

FORK

Una bifurcación (fork en inglés), en el ámbito del *desarrollo de software*, es la creación de un proyecto en una dirección distinta de la principal u oficial tomando el *código fuente* del proyecto ya existente. Comúnmente se utiliza el término inglés.

Programa de actividades Cronograma de actividades

*{Aquí incluimos un **ejemplo** de cronograma, considerando un semestre, para la ejecución del proyecto}*

Actividades por Quincena	01 -15 enero	16-01 Enero	01-15 Febrero	16-28 febrero	01-15 marzo	16-30 Marzo	01-15 abril	16-31 abril	01-15 mayo	16-30 mayo	01 - 15junio	16-30 junio
Creación de servicios REST												
Elaboración de procedimientos almacenados												
Elaboración de controllers,factories y services de Angular JS												
Diseño de layout y adecuaciones al sistema												
Creación de vistas y templates en boostrap												
Testeo y deploy de aokicación												

Referencias

https://blogs.msdn.microsoft.com/brian_swan/2011/02/16/do-stored-procedures-protect-against-sql-injection/

https://dzone.com/articles/web-services-architecture

http://envivo.bancomundial.org/tecnolog%C3%ADas-de-la-informaci%C3%B3n-y-de-la-comunicaci%C3%B3n-para-el-desarrollo

https://msdn.microsoft.com/es-mx/library/kx37x362.aspx