

-Sistemas Informáticos-

Administración de Base de Datos

Profesor: Miriam Magaly Canche Caamal

**Integrantes:**

José André Salazar Figueroa

José Luis Puc Chan

Yair Vega Gamboa

Diosemir Nah May

Aaron Peña Martin

4° Cuatrimestre Grupo B

Mérida Yucatán, 16 de noviembre de 2016

# Planteamiento del problema



Nombre del Proyecto:

MCTuristic

Descripción:

Mérida también conocida como "la ciudad blanca" se distingue por ser tranquila y sus habitantes amables y hospitalarios. Es el lugar ideal para la gente que desea pasar unas vacaciones en un ambiente relajado y con un clima cálido. Mérida fue fundada el 6 de enero de 1542, y construida sobre la ciudad Maya de Toh.

Esta ciudad ha creado su historia a lo largo de muchos años, por lo cual en su mayoría tiene una historia diferente la cuál contar. Esto hace que Mérida sea una ciudad cultural más importante de Yucatán.

Para comenzar el recorrido por el centro histórico de Mérida lo recomendable sería hacerlo en el Zócalo, también llamado Plaza Grande, lugar de encuentro entre locales y extranjeros, por sus amplios espacios verdes y hermosos jardines, alrededor del cual se encuentran los principales edificios de valor histórico construidos en la época colonial.

En esta plaza central se llevan a cabo numerosos eventos, festivales callejeros, música y bailes típicos, además de diferentes puestos de comida típica para disfrutar de los sabores típicos de la región.

La casa de Montejo es un edificio característico, con mucha historia que fue construida para ser el lugar de residencia del fundador de Mérida y su familia. Su estilo renacentista se ha conservado y actualmente es propiedad de la Fundación Fomento Cultural Banamex y funciona como una casa de cultura y museo. Se exhiben numerosos objetos de la época colonial, la arquitectura original del inmueble, así como exposiciones de obras de arte y distintas expresiones artísticas.

Podemos encontrar el Palacio de Gobierno, inaugurado en 1892 luego de su reconstrucción total. En un principio estaban las oficinas de todas las dependencias del gobierno regional, actualmente estas se han distribuido a otros edificios.

La arquitectura que ostenta el edificio es imponente, en sus espacios se puede encontrar pinturas del afamado pintor, escultor y muralista Fernando Castro Pacheco que representan la historia del estado de Yucatán y de la ciudad.

Entre otros lugares importantes que hacen de Yucatán uno de los mejores lugares para conocer y visitar.

Por lo tanto se te pide que realices lo siguiente:

Con intenciones de fomentar la investigación sobre la historia de nuestros espacios en el centro histórico, se requiere realizar un proyecto en .NET y plataforma Android, conformado por dos aplicaciones una solución web y una para móviles, es necesario manejar conexión a una base de datos desarrollada en SQL Server para almacenar la información de la historia de los puntos más importantes del centro histórico de Mérida y dar a conocer los acontecimientos de estos.

El objetivo es fomentar el conocimiento sobre nuestra historia al igual que los usuarios conozcas sus raíces y los acontecimientos importantes de nuestro estado.

Almacenamiento de los datos de los sitios:

Nombre: Zócalo

Nombre Local: Plaza Grande

Otros Nombres: ….

Descripción: este lugar ha sido testigo una gran parte de la vida de la península de Yucatán, pues aquí desde épocas coloniales se llevaban a cabo una gran gama de eventos que reunían a todas las clases sociales del estado pues aquí se llevaban a cabo torneos de palo encebado, corridas de toros y por supuesto la fiesta del carnaval.

Sucesos relevantes: Durante el período colonial, la Plaza Principal fue mercado público, lugar de corridas de toros, escenario para aplicar penas corporales y ajusticiar cuando era necesario, además de ser un sitio donde se realizaban torneos y festejos de interés general.

Foto**:**



Almacenamiento de los datos de lugares cercanos:

Nombre: Hotel Ricardo

Tipo de Lugar: Descanso

Horarios de apertura y de cierre: 6:00 AM – 11:00 PM

Descripción de servicios: Cuarto desde 160.00 La noche

Página Facebook: HotelRicadoMerida

Teléfono: 9932452335

# Justificación del proyecto

Ante los constantes retos de la era moderna, tenemos la oportunidad de hacer una gran aportación para la sociedad, una de las aportaciones que tenemos es el propósito de adaptar las nuevas tecnologías  para el fomento turístico e informativo de nuestros espacios históricos en el centro de nuestra ciudad.

Se detectó que al viajar como turista al centro histórico de la ciudad, este mismo carece del fomento informativo de zonas culturales e históricas que se encuentran en la zona, el proyecto propone una aportación tecnológica para solventar el problema y promover servicios y eventos culturales que estas zonas ofrecen , la aplicación móvil ofrecerá interactividad con el usuario al reaccionar a ciertas necesidades tales como la hora del día para recomendar donde comer , recomendar eventos realizados por los centros culturales del centro histórico.

La aplicación web administrara el contenido a ser mostrado por la aplicación móvil, se administraran los sitios que serán geo ubicados y definirá las reglas de interactividad con el usuario, se gestionara e implementara las rutas recomendadas de paseo por el centro histórico de la ciudad de Mérida , administrara los eventos que se realizaran por los centros culturales del centro histórico , tendrá un apartado para gestionar los establecimientos privados que requieran suscribirse para aparecer en la aplicación móvil.

.

La aplicación móvil ofrecerá al turista información relevante sobre diversos sitios históricos culturales del centro histórico de Mérida incluyendo esquinas, iglesias, parques, museos, teatros, etc. Se contempla incluir sitios comerciales que requerirán de suscripciones de pago y tendrán tiempo de vigencia para ser visibles en la aplicación.

# Requerimientos aplicación web

En la aplicación web se podrá gestionar el módulo de sitios que serán geo ubicados el cual requerirá de datos específicos como son (dirección , Nombre , Descripción , Historial y su respectiva imagen ) y, definirá las reglas de interactividad con el usuario, se gestionara e implementara las rutas recomendadas de paseo por el centro histórico de la ciudad de Mérida . En el módulo de eventos en la aplicación web se pondrán dar de alta dichos eventos los cuales requerirán de datos respectivos como (horarios, Fechas de Inicio y Finalización, etc.) que se realizaran por los centros culturales del centro histórico, tendrá un apartado para gestionar los establecimientos privados el cual solicitara datos de este mismo como son (Nombre, Dirección, Teléfono, etc.) Que requieran suscribirse para aparecer en la aplicación móvil.

En la aplicación web al igual se podrá gestionar dichas suscripciones que los establecimientos requieran para aparecer en el sistema, para este apartado las suscripciones la web requerirá solicitar datos del suscriptor que serían (Tipo de suscripción, Tiempo de Suscripción, Nombre, etc.)

# Requerimientos aplicación móvil

La aplicación móvil requerirá de consumir datos de la aplicación web, esta se encargara de mostrar la información obtenía ya se han establecimientos, sitios, eventos, etc., En la aplicación móvil se podrá dar de alta usuarios que requerirá de datos de este mismo como son (Nombre, Apellidos, Teléfono, Correo, Fecha de Nacimiento y Contraseña). En la aplicación móvil el usuario podrá gestionar su perfil otorgado.

# Modelo Entidad-Relación

### [Modelo Dia](1.Modelo%20E-R.dia)

# Modelo Relacional

# SCRIPT BASE DE DATOS

## [Base datos generado en Embarcadero](Contenido/3.Script_Base%20de%20Datos.sql)

# Respaldo base datos

### [Respaldo con extensión (.bak)](Contenido/4.%20MCTuristic.bak)

# Archivo de embarcadero E-R Studio

## [(Modelo lógico, físico)](Contenido/5.Modelo%20Lógico%20y%20Físico.DM1)

# Diccionario de datos

## [Diccionario de datos generado desde SQL Server](Contenido/6.Diccionario%20de%20datos.sql)

# Manejo de restricciones

## [Restricciones script](Contenido/7.Restricciones.sql)

# Procedimientos almacenados

-- Este procedimiento sirve para agregar un nuevo administrador en la base de datos

CREATE PROC insertar\_administrador

(

@Nombre varchar(25),

@Apellidos varchar(25),

@Email varchar(50),

@Contrasena varchar(20),

@TelefonoCelular varchar(25),

@FechaNacimiento varchar(25)

)

as

INSERT INTO Administrador (Nombre,Apellidos,Email,Contrasena,TelefonoCelular,FechaNacimiento)

VALUES (@Nombre,@Apellidos,@Email,@Contrasena,@TelefonoCelular,@FechaNacimiento)

Go

-- Este procedimiento sirve para actualizar los datos de un administrador previamente registrado

CREATE PROC actualizar\_administrador

(

@IdAdminstrador int,

@Nombre varchar(25),

@Apellidos varchar(25),

@Email varchar(50),

@Contrasena varchar(20),

@TelefonoCelular varchar(25),

@FechaNacimiento varchar(25)

)

as

UPDATE Administrador

SET Nombre = @Nombre,Apellidos= @Apellidos,Email = @Email,Contrasena = @Contrasena,TelefonoCelular = @TelefonoCelular,FechaNacimiento= @FechaNacimiento

WHERE IdAdministrador = @IdAdminstrador

Go

-- Este procedimiento sirve para eliminar en su totalidad a un administrador que se encuentre en la base de datos

Create proc eliminar\_administrador

(

@IdAdministrador int

)

as

DELETE FROM Administrador WHERE IdAdministrador = @IdAdministrador

GO

-- Este procedimiento sirve para visualizar los datos de un administrador junto con la dirección que le pertenece

Create procedure ver\_administrador

as

SELECT dbo.Administrador.IdAdministrador, dbo.Administrador.Nombre, dbo.Administrador.Apellidos, dbo.Administrador.Email, dbo.Administrador.TelefonoCelular, dbo.Administrador.FechaNacimiento, dbo.Direccion.idDireccion ,dbo.Direccion.Calle, dbo.Direccion.Cruzamiento,

dbo.Direccion.Numero, dbo.Direccion.Colonia, dbo.Direccion.Descripcion, dbo.Direccion.Estado, dbo.Direccion.CodPostal

FROM dbo.Administrador INNER JOIN

dbo.Direccion ON dbo.Administrador.IdAdministrador = dbo.Direccion.IdAdministrador

GO

-- Este procedimiento sirve para seleccionar el último registro ingresado en la tabla de usuario

create procedure usuario\_reciente

as

select MAX([IdUsuario]) as Maximo from Usuario

go

-- Este procedimiento sirve para agregar un nuevo usuario en la base de datos

CREATE PROC insertar\_usuario

(

@Nombre varchar(25),

@Apellidos varchar(25),

@Email varchar(50),

@Contrasena varchar(20),

@TelefonoCelular varchar(25),

@FechaNacimiento varchar(25),

@Foto image

)

as

INSERT INTO Usuario(Nombre,Apellidos,Email,Contrasena,TelefonoCelular,FechaNacimiento,Foto)

VALUES (@Nombre,@Apellidos,@Email,@Contrasena,@TelefonoCelular,@FechaNacimiento,@Foto)

Go

-- Este procedimiento sirve para actualizar los datos de un usuario previamente registrado

CREATE PROC actualizar\_usuario

(

@IdUsuario int,

@Nombre varchar(50),

@Apellidos varchar(50),

@Email varchar(50),

@Contrasena varchar(20),

@TelefonoCelular varchar(50),

@FechaNacimiento varchar(25),

@Foto image

)

as

UPDATE Usuario

SET Nombre = @Nombre,Apellidos= @Apellidos,Email = @Email,Contrasena = @Contrasena,TelefonoCelular = @TelefonoCelular,FechaNacimiento= @FechaNacimiento, Foto = @Foto

WHERE IdUsuario = @IdUsuario

Go

-- Este procedimiento sirve para eliminar en su totalidad a un usuario que se encuentre en la base de datos

Create proc eliminar\_usuario

(

@IdUsuario int

)

as

DELETE FROM Usuario WHERE IdUsuario = @IdUsuario

GO

-- Este procedimiento sirve para visualizar los datos de un administrador junto con la dirección que le pertenece

create procedure ver\_usuario

as

SELECT dbo.Usuario.IdUsuario, dbo.Usuario.Nombre, dbo.Usuario.Apellidos, dbo.Usuario.Email, dbo.Usuario.TelefonoCelular, dbo.Usuario.FechaNacimiento, dbo.Direccion.Calle, dbo.Direccion.idDireccion,

dbo.Direccion.Cruzamiento, dbo.Direccion.Numero, dbo.Direccion.Descripcion, dbo.Direccion.Estado, dbo.Direccion.CodPostal

FROM dbo.Direccion INNER JOIN

dbo.Usuario ON dbo.Direccion.IdUsuario = dbo.Usuario.IdUsuario

GO

-- Este procedimiento sirve para agregar una nueva dirección de un usuario en la base de datos

Create procedure insertar\_direccion\_usuario

(

@Calle varchar(50),

@Cruzamiento varchar(50),

@Numero varchar(50),

@Descripcion varchar(50),

@Colonia varchar(50),

@Estado varchar(50),

@CodPostal varchar(50),

@IdUsuario int

)

as

INSERT INTO Direccion (Calle,Cruzamiento,Numero,Descripcion,Colonia,Estado,CodPostal,IdUsuario,IdAdministrador)

VALUES (@Calle,@Cruzamiento,@Numero,@Descripcion,@Colonia,@Estado,@CodPostal,@IdUsuario,null)

Go

-- Este procedimiento sirve para actualizar los datos de una dirección de un usuario previamente registrada

Create procedure actualizar\_direccion\_usuario

(

@idDireccion int,

@Calle varchar(50),

@Cruzamiento varchar(50),

@Numero varchar(50),

@Descripcion varchar(50),

@Colonia varchar(50),

@Estado varchar(50),

@CodPostal varchar(50),

@IdUsuario int

)

as

UPDATE Direccion SET Calle = @Calle,Cruzamiento = @Cruzamiento,Numero = @Numero,Descripcion = @Descripcion,Colonia = @Colonia,Estado = @Estado,CodPostal = @CodPostal,IdUsuario = @IdUsuario,IdAdministrador = null

where idDireccion = @idDireccion

go

-- Este procedimiento sirve para agregar una nueva dirección de un administrador en la base de datos

Create procedure insertar\_direccion

(

@Calle varchar(50),

@Cruzamiento varchar(50),

@Numero varchar(50),

@Descripcion varchar(50),

@Colonia varchar(50),

@Estado varchar(50),

@CodPostal varchar(50),

@IdUsuario int,

@IdAdministrador int

)

as

INSERT INTO Direccion (Calle,Cruzamiento,Numero,Descripcion,Colonia,Estado,CodPostal,IdUsuario,IdAdministrador)

VALUES (@Calle,@Cruzamiento,@Numero,@Descripcion,@Colonia,@Estado,@CodPostal,@IdUsuario,@IdAdministrador)

go

-- Este procedimiento sirve para actualizar los datos de una dirección de un administrador previamente registrada

Create procedure actualizar\_direccion

(

@idDireccion int,

@Calle varchar(50),

@Cruzamiento varchar(50),

@Numero varchar(50),

@Descripcion varchar(50),

@Colonia varchar(50),

@Estado varchar(50),

@CodPostal varchar(50),

@IdUsuario int,

@IdAdministrador int

)

as

UPDATE Direccion SET Calle = @Calle,Cruzamiento = @Cruzamiento,Numero = @Numero,Descripcion = @Descripcion,Colonia = @Colonia,Estado = @Estado,CodPostal = @CodPostal,IdUsuario = @IdUsuario,IdAdministrador = @IdAdministrador

where idDireccion = @idDireccion

go

-- Este procedimiento sirve para eliminar en su totalidad una dirección que se encuentre en la base de datos

create procedure eliminar\_direccion

(

@idDireccion int

)

as

Delete from Direccion where idDireccion = @idDireccion

go

-- Este procedimiento sirve para agregar un nuevo establecimiento en la base de datos

Create procedure insertar\_establecimiento

(

@NomEstable varchar(10),

@Telefono varchar(50),

@HoraInicio varchar(50),

@HoraCierre varchar(50),

@PagFacebook varchar(50),

@Longitud Decimal(10,2),

@Latitud Decimal(10,2),

@Foto image,

@IdUsuario int

)

as

INSERT INTO Establecimiento (NomEstable,Telefono,HoraInicio,HoraCierre,PagFacebook,Latitud,Longitud,Foto,IdUsuario)

VALUES(@NomEstable,@Telefono,@HoraInicio,@HoraCierre,@PagFacebook,@Latitud,@Longitud,@Foto,@IdUsuario)

Go

-- Este procedimiento sirve para actualizar los datos de un establecimiento previamente registrado

Create procedure actualizar\_establecimiento

(

@IdEstablecimiento int,

@NomEstable varchar(10),

@Telefono varchar(50),

@HoraInicio varchar(50),

@HoraCierre varchar(50),

@PagFacebook varchar(50),

@Longitud Decimal(10,2),

@Latitud Decimal(10,2),

@Foto image,

@IdUsuario int

)

as

UPDATE Establecimiento SET NomEstable =@NomEstable,Telefono =@Telefono,HoraInicio = @HoraInicio,HoraCierre =@HoraCierre

,PagFacebook =@PagFacebook ,Latitud = @Latitud,Longitud = @Longitud,Foto = @Foto,IdUsuario = @IdUsuario

WHERE idEstablecimiento = @IdEstablecimiento

Go

-- Este procedimiento sirve para eliminar en su totalidad un establecimiento que se encuentre en la base de datos

create procedure eliminar\_establecimiento

(

@IdEstablecimiento int

)

as

DELETE FROM Establecimiento WHERE idEstablecimiento = @IdEstablecimiento

go

-- Este procedimiento sirve para agregar un nuevo tipo de servicio en la base de datos

create procedure insertar\_tiposervicio

(

@NombreServicio varchar(50)

)

as

insert into TipoServicio (NombreServicio) values (@NombreServicio)

go

-- Este procedimiento sirve para actualizar los datos de un tipo de servicio previamente registrado

create procedure actualizar\_tiposervicio

(

@idTipoServ int,

@NombreServicio varchar(50)

)

as

Update TipoServicio SET NombreServicio= @NombreServicio

WHERE idTipoServ = @idTipoServ

go

-- Este procedimiento sirve para eliminar en su totalidad un tipo de servicio que se encuentre en la base de datos

create procedure eliminar\_tiposervicio

(

@idTipoServ int

)

as

DELETE FROM TipoServicio WHERE idTipoServ = @idTipoServ

Go

-- Este procedimiento sirve para agregar un nuevo servicio en la base de datos

create procedure insertar\_servicios

(

@NombreServ varchar(30),

@OfertaServicio varchar(50),

@PrecioServicio decimal(10,2),

@DescripServicio varchar(50),

@Foto image,

@Popularidad int,

@idEstablecimiento int,

@idTipoServi int

)

as

Insert INTO Servicios (NombreServ,OfertaServicio,PreciosServicio,DescripServicio,Foto,Popularidad,idEstablecimiento,idTipoServ)

VALUES(@NombreServ,@OfertaServicio,@PrecioServicio,@DescripServicio,@Foto,@Popularidad,@idEstablecimiento,@idTipoServi)

go

-- Este procedimiento sirve para actualizar los datos de un servicio previamente registrado

create procedure actualizar\_servicios

(

@idServicio int,

@NombreServ varchar(50),

@OfertaServicio varchar(50),

@PrecioServicio decimal(10,2),

@DescripServicio varchar(50),

@Foto image,

@Popularidad int,

@idEstablecimiento int,

@idTipoServi int

)

as

UPDATE Servicios SET NombreServ =@NombreServ, OfertaServicio = @OfertaServicio,PreciosServicio = @PrecioServicio,

DescripServicio = @DescripServicio,Foto=@Foto,Popularidad=@Popularidad,idEstablecimiento= @idEstablecimiento,idTipoServ = @idTipoServi

where idServicio = @idServicio

go

-- Este procedimiento sirve para eliminar en su totalidad un servicio que se encuentre en la base de datos

create procedure eliminar\_servicios

(

@idServicio int

)

as

delete from Servicios where idServicio =@idServicio

GO

-- Este procedimiento sirve para agregar un nuevo tipo de suscripción en la base de datos

create procedure insertar\_tiposuscripcion

(

@Nombre varchar(50),

@Monto decimal(10,2),

@TiempoSuscripcion int

)

as

insert into TipoSuscripcion (Nombre,Monto,TiempoSuscripcion)

VALUES (@Nombre,@Monto,@TiempoSuscripcion)

go

-- Este procedimiento sirve para actualizar los datos de un tipo de suscripción previamente registrado

create procedure actualizar\_tiposuscripcion

(

@idSuscripcion int,

@Nombre varchar(50),

@Monto decimal(10,2),

@TiempoSuscripcion int

)

as

update TipoSuscripcion set Nombre = @Nombre,Monto = @Monto,TiempoSuscripcion = @TiempoSuscripcion

where idSuscripcion = @idSuscripcion

go

-- Este procedimiento sirve para eliminar en su totalidad un tipo de suscripción que se encuentre en la base de datos

create procedure eliminar\_tiposuscripcion

(

@idSuscripcion int

)

as

delete from TipoSuscripcion WHERE idSuscripcion = @idSuscripcion

go

-- Este procedimiento sirve para agregar un nuevo soporte en la base de datos

create procedure insertar\_soporte

(

@Comentario varchar(500),

@Asunto varchar(50),

@IdUsuario int

)

as

insert into Soporte (Comentario,Asunto,IdUsuario)

VALUES (@Comentario,@Asunto,@IdUsuario)

GO

-- Este procedimiento sirve para actualizar los datos de un soporte previamente registrado

Create procedure actualizar\_soporte

(

@idSoporte int,

@Comentario varchar(500),

@Asunto varchar(50),

@IdUsuario int

)

as

UPDATE Soporte SET Comentario = @Comentario,Asunto= @Asunto,IdUsuario= @IdUsuario

WHERE idSoporte = @IdUsuario

GO

-- Este procedimiento sirve para eliminar en su totalidad un soporte que se encuentre en la base de datos

Create procedure eliminar\_soporte

(

@idSoporte int

)

as

delete from Soporte where idSoporte = @idSoporte

go

-- Este procedimiento sirve para agregar un nuevo tipo de sitio en la base de datos

create procedure insertar\_tipositio

(

@NombreSitio varchar(50),

@DescripcionSitio varchar(50)

)

as

insert into TipoSitio (NombreSitio,DescripcionSitio) VALUES

(@NombreSitio,@DescripcionSitio)

go

-- Este procedimiento sirve para actualizar los datos de un tipo de sitio previamente registrado

create procedure actualizar\_tipositio

(

@idTipoSitio int ,

@NombreSitio varchar(50),

@DescripcionSitio varchar(50)

)

as

UPDATE TipoSitio SET NombreSitio = @NombreSitio,DescripcionSitio = @DescripcionSitio WHERE

idTipoSitio = @idTipoSitio

go

-- Este procedimiento sirve para eliminar en su totalidad un tipo de sitio que se encuentre en la base de datos

create procedure eliminar\_tipositio

(

@idTipoServicio int

)

as

DELETE FROM TipoSitio where idTipoSitio = @idTipoServicio

go

-- Este procedimiento sirve para agregar un nuevo sitio en la base de datos

create procedure insertar\_sitio

(

@Nombre varchar(50),

@Descripcion varchar(400),

@Historia varchar(500),

@Direccion varchar(50),

@Longitud decimal(10,2),

@Latitud decimal(10,2),

@Foto image,

@SucesosImp varchar(50),

@idTipoSitio int

)

as

insert into Sitio (Nombre,Descripcion,Historia,Direccion,Longitud,Latitud,Foto,SucesosImp,idTipoSitio)

VALUES (@Nombre,@Descripcion,@Historia,@Direccion,@Longitud,@Latitud,@Foto,@SucesosImp,@idTipoSitio)

go

-- Este procedimiento sirve para actualizar los datos de un sitio previamente registrado

create procedure actualizar\_sitio

(

@IdSitio int,

@Nombre varchar(50),

@Descripcion varchar(400),

@Historia varchar(500),

@Direccion varchar(50),

@Longitud decimal(10,2),

@Latitud decimal(10,2),

@Foto image,

@SucesosImp varchar(50),

@idTipoSitio int

)

as

UPDATE Sitio SET Nombre = @Nombre,Descripcion = @Descripcion

,Historia = @Historia,Direccion = @Direccion,Longitud = @Longitud,Latitud = @Latitud,Foto= @Foto,SucesosImp = @SucesosImp

,idTipoSitio = @idTipoSitio where IdSitio = @IdSitio

go

-- Este procedimiento sirve para eliminar en su totalidad un sitio que se encuentre en la base de datos

create procedure eliminar\_sitio

(

@idSitio int

)

as

delete from Sitio where IdSitio = @idSitio

go

-- Este procedimiento sirve para agregar un nuevo evento en la base de datos

create procedure insertar\_evento

(

@NombreEvent varchar(50),

@HoraIncio varchar(50),

@HoraFinalizacion varchar(50),

@FechaIncio date,

@FechaFinalizacion date,

@Foto image,

@IdUsuario int,

@IdSitio int

)

as

insert into Eventos (NombreEvent,HoroInicio,HoraFinalizacion,FechaIncio,FechaFinalizacion,Foto,IdUsuario,IdSitio)

VALUES (@NombreEvent,@HoraIncio,@HoraFinalizacion,@FechaIncio,@FechaFinalizacion,@Foto,@IdUsuario,@IdSitio)

go

-- Este procedimiento sirve para actualizar los datos de un evento previamente registrado

create procedure actualizar\_evento

(

@idEvento int ,

@NombreEvent varchar(50),

@HoraIncio varchar(50),

@HoraFinalizacion varchar(50),

@FechaIncio date,

@FechaFinalizacion date,

@Foto image,

@IdUsuario int,

@IdSitio int

)

as

UPDATE Eventos SET

NombreEvent = @NombreEvent,HoroInicio = @HoraIncio,

HoraFinalizacion = @HoraFinalizacion,FechaIncio = @FechaIncio,FechaFinalizacion = @FechaFinalizacion,Foto=@Foto

,IdUsuario = @IdUsuario,IdSitio = @IdSitio WHERE idEvento= @idEvento

go

-- Este procedimiento sirve para eliminar en su totalidad un evento que se encuentre en la base de datos

create procedure eliminar\_evento

(

@idEvento int

)

as

delete from Eventos where idEvento =@idEvento

go

-- Este procedimiento sirve para agregar una nueva notificación en la base de datos

create procedure insertar\_notificaciones

(

@idEvento int

)

as

insert into Notificaciones (idEvento) values (@idEvento)

go

-- Este procedimiento sirve para actualizar los datos de una notificación previamente registrado

create procedure actualizar\_notificaciones

(

@idNotificaciones int,

@idEvento int

)

as

UPDATE Notificaciones SET idEvento = @idEvento WHERE idNotificacion = @idNotificaciones

go

-- Este procedimiento sirve para eliminar en su totalidad una notificación que se encuentre en la base de datos

create procedure eliminar\_notificaciones

(

@idNotificaciones int

)

as

delete from Notificaciones where idNotificacion = @idNotificaciones

go

-- Este procedimiento sirve para agregar un nuevo registro de contacto en la base de datos

create proc insertar\_contactanos

(

@Nombre varchar(20),

@Apellidos varchar(20),

@Email varchar(40),

@Asunto varchar(30),

@Contenido varchar(200)

)

as

insert into Contactanos (Nombre,Apellido,Email,Asunto,Contenido)

VALUES (@Nombre,@Apellidos,@Email,@Asunto,@Contenido)

GO

-- Este procedimiento sirve para actualizar los datos de un contacto previamente registrado

Create proc actualizar\_contactanos

(

@IdContactanos int,

@Nombre varchar(20),

@Apellidos varchar(20),

@Email varchar(40),

@Asunto varchar(30),

@Contenido varchar(200)

)

as

UPDATE Contactanos SET Nombre = @Nombre , Apellido = @Apellidos, Email = @Email,Asunto =@Asunto,

Contenido = @Contenido WHERE IdContactanos = @IdContactanos

GO

-- Este procedimiento sirve para eliminar en su totalidad un registro de contacto que se encuentre en la base de datos

Create proc eliminar\_contantanos

(

@IdContactanos int

)

as

Delete from Contactanos where IdContactanos = @IdContactanos

go

-- Este procedimiento sirve para agregar una nueva suscripción en la base de datos

create proc insertar\_Suscripcion

(

@idSuscripcion int,

@FechaPago date,

@IdUsuario int

)

as

insert into Suscripcion (idSuscripcion,FechaPago,IdUsuario)

VALUES (@idSuscripcion,@FechaPago,@IdUsuario)

GO

-- Este procedimiento sirve para actualizar los datos de una suscripción previamente registrada

Create proc actualizar\_suscripcion

(

@idSuscripcion int,

@FechaPago date,

@IdUsuario int

)

as

UPDATE Suscripcion SET FechaPago = @FechaPago, IdUsuario =@IdUsuario WHERE idSuscripcion = @idSuscripcion

GO

-- Este procedimiento sirve para eliminar en su totalidad una suscripción que se encuentre en la base de datos

Create proc eliminar\_suscripcion

(

@idSuscripcion int

)

as

Delete from Suscripcion where idSuscripcion = @idSuscripcion

go

-- Este procedimiento sirve para agregar un nuevo comentario en la base de datos

create proc insertar\_comentarios

(

@Nombre varchar(50),

@Contenido varchar(100),

@Status int,

@IdUsuario int

)

as

insert into Comentarios (Nombre,Contenido,Status,IdUsuario)

VALUES (@Nombre,@Contenido,@Status,@IdUsuario)

GO

-- Este procedimiento sirve para actualizar los datos de un comentario previamente registrado

Create proc actualizar\_comentarios

(

@IdComentario int,

@Nombre varchar(50),

@Contenido varchar(100),

@Status int,

@IdUsuario int

)

as

UPDATE Comentarios SET Nombre=@Nombre, Contenido=@Contenido, Status = @Status, IdUsuario = @IdUsuario WHERE IdComentario = @IdComentario

GO

-- Este procedimiento sirve para eliminar en su totalidad un comentario que se encuentre en la base de datos

Create proc eliminar\_comentario

(

@IdComentario int

)

as

Delete from Comentarios where IdComentario = @IdComentario

go

-- Este procedimiento sirve para identificar qué administrador está logueandose en el sistema

create proc login\_user

@email varchar(20),

@password varchar(20)

as

select Nombre , IdAdministrador

from Administrador

where Email=@email and Contrasena=@password

go

## [En su totalidad se debe hacer uso de procedimientos almacenados](8.Procedimientos%20Almacenados.sql)

## Introducción

### Ester procedimiento se encarga de insertar los datos de los administradores, recibe parámetros necesarios para la inserción en la base de datos

## Resultados

### Al final podemos observar como los datos se agregaron de manera correcta a la base de datos.

# dISPARADORES

## [Mínimo 3 disparadores](Contenido/9.%20Disparadores.sql)

## Introducción

### Este disparador se encarga de verificar que el nuevo administrador que se quiere registrar no repita el correo electrónico que es único, como se pudo observar anteriormente se agregó un usuario con un procedimiento almacenado que inserto ciertos datos al servido, para este ejemplo se corrió de nuevo el procedimiento pero esta vez el resultado fue diferente.

## Resultados

### La base de datos ejecuta el disparador al momento de hacer el insert, este se activa y hace un método de búsqueda para saber si existen coincidencia, al efectuar retorna un mensaje.

------------------Verficar si existe el Administrador

CREATE TRIGGER InsertaAdmin ON Administrador

AFTER INSERT

AS

IF EXISTS (SELECT \*

FROM Administrador AS p

JOIN inserted AS i

ON p.Email = i.Email

)

BEGIN

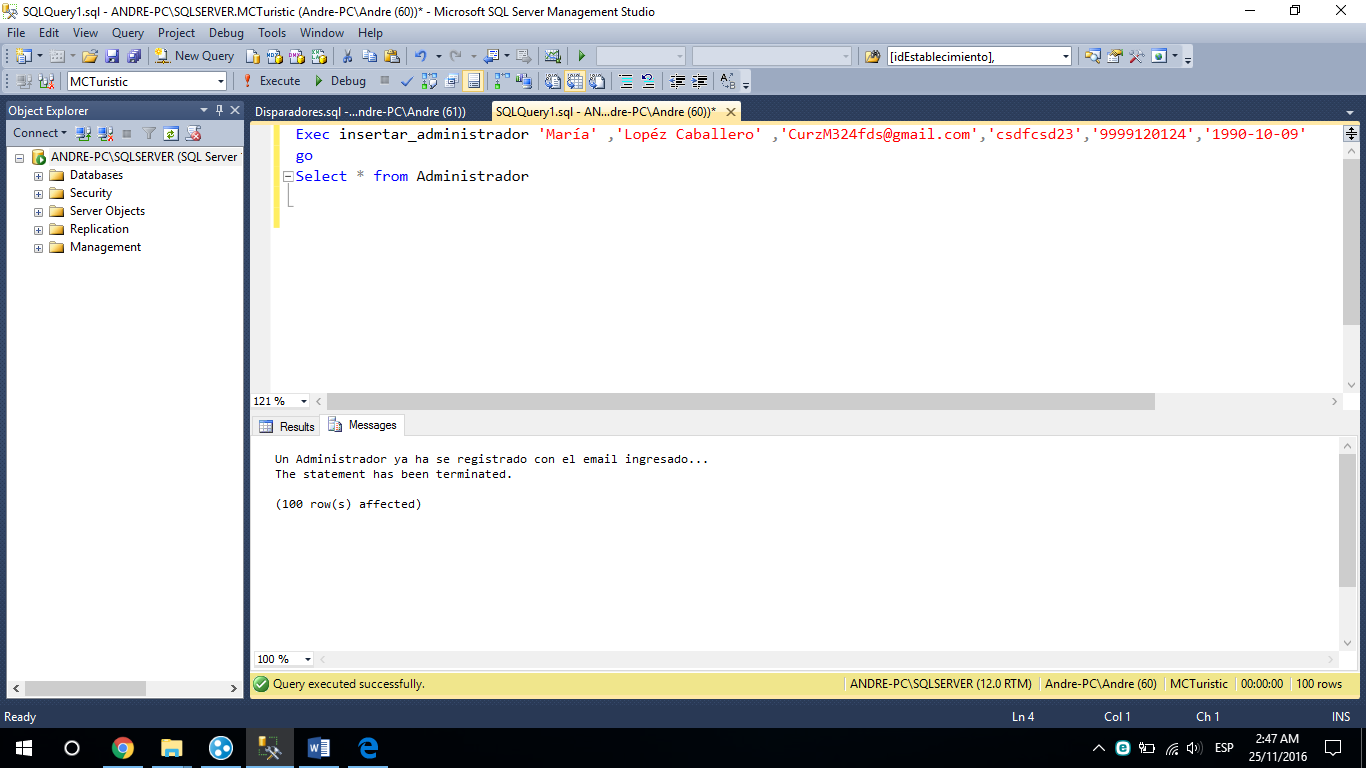
RAISERROR ('Un Administrador ya ha se registrado con el email ingresado', 16, 1);

ROLLBACK TRANSACTION;

RETURN

END;

GO



# iNDICES

### [**Manejo de indices**](Contenido/10.%20Indices.sql)

create index NombreEventos

on Eventos(NombreEvent)

create index NombreUsuario

on Usuario(Nombre ,Apellidos)

create index servicios

on Servicios(NombreServ,OfertaServicio)

create index sp\_establecimientos

on Establecimiento(NomEstable,Telefono)

# vISTAS

## [Manejo de 5 Vistas como mínimo](Contenido/11.%20Vistas.sql)

CREATE VIEW mostrar\_Administrador

as

SELECT Adm.IdAdministrador, Adm.Nombre, Adm.Apellidos,

Adm.Email, Adm.TelefonoCelular, Adm.FechaNacimiento,

D.idDireccion ,

D.Calle, D.Cruzamiento,

D.Numero, D.Colonia,

D.Descripcion, D.Estado, D.CodPostal

FROM Administrador as Adm

INNER JOIN

Direccion as D ON D.IdAdministrador = D.IdAdministrador

GO

CREATE VIEW mostrar\_usuarios

as

SELECT Us.IdUsuario, Us.Nombre,

Us.Apellidos, Us.Email,

Us.TelefonoCelular, Us.FechaNacimiento,

D.Calle,D.idDireccion,

D.Cruzamiento, D.Numero, D.Descripcion,

D.Estado, D.CodPostal

FROM Direccion AS D INNER JOIN

Usuario as Us ON D.IdUsuario = Us.IdUsuario

GO

CREATE VIEW mostrar\_productos

as

SELECT Es.idEstablecimiento,

Es.NomEstable, Es.HoraInicio,

Es.HoraCierre, Ser.NombreServ, Ser.OfertaServicio,

Ser.PreciosServicio, TipSer.NombreServicio

FROM Establecimiento as Es INNER JOIN

Servicios as Ser ON Es.idEstablecimiento = Ser.idEstablecimiento INNER JOIN

TipoServicio as TipSer ON Ser.idTipoServ = TipSer.idTipoServ

GO

CREATE VIEW mostrar\_suscripciones

as

SELECT us.IdUsuario, us.Nombre,us.Apellidos, us.Email, Sp.FechaPago,

TpS.Nombre AS TipoSuscripcion,

TpS.Monto,

TpS.TiempoSuscripcion

FROM dbo.Usuario us INNER JOIN

Suscripcion Sp ON us.IdUsuario = Sp.IdUsuario INNER JOIN

TipoSuscripcion as TpS ON Sp.idSuscripcion = TpS.idSuscripcion

GO

CREATE VIEW mostrar\_eventos

as

SELECT St.Nombre,St.Descripcion,St.Direccion, Ev.NombreEvent, Ev.HoroInicio,

Ev.HoraFinalizacion, Ev.FechaIncio, Ev.FechaFinalizacion,

TpS.NombreSitio

FROM dbo.Sitio St INNER JOIN

TipoSitio as TpS ON St.idTipoSitio = TpS.idTipoSitio INNER JOIN

Eventos as Ev ON St.IdSitio = Ev.IdSitio

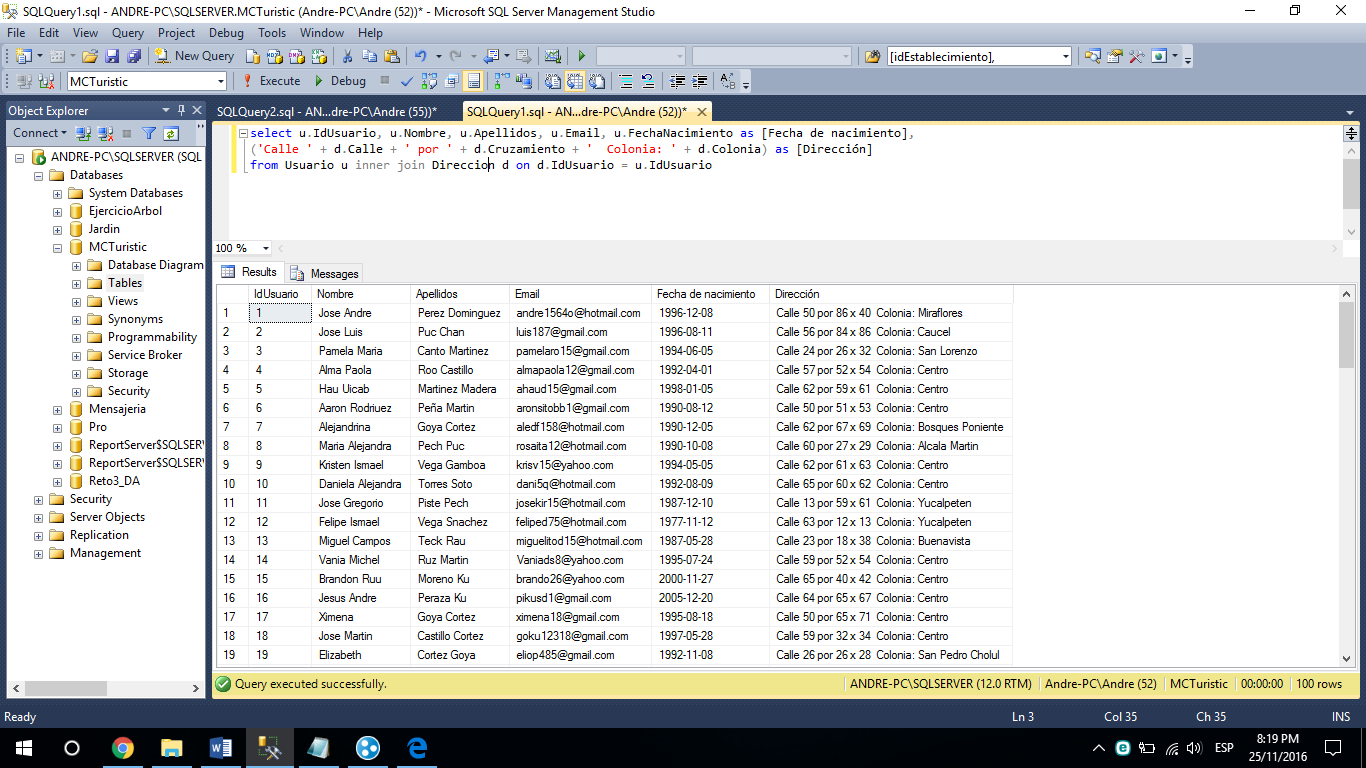
GO

# Joins

### **Selección de datos del usuario con su respectiva dirección**

select u.IdUsuario, u.Nombre, u.Apellidos, u.Email, u.FechaNacimiento as [Fecha de nacimiento],

('Calle ' + d.Calle + ' por ' + d.Cruzamiento + ' Colonia: ' + d.Colonia) as [Dirección]

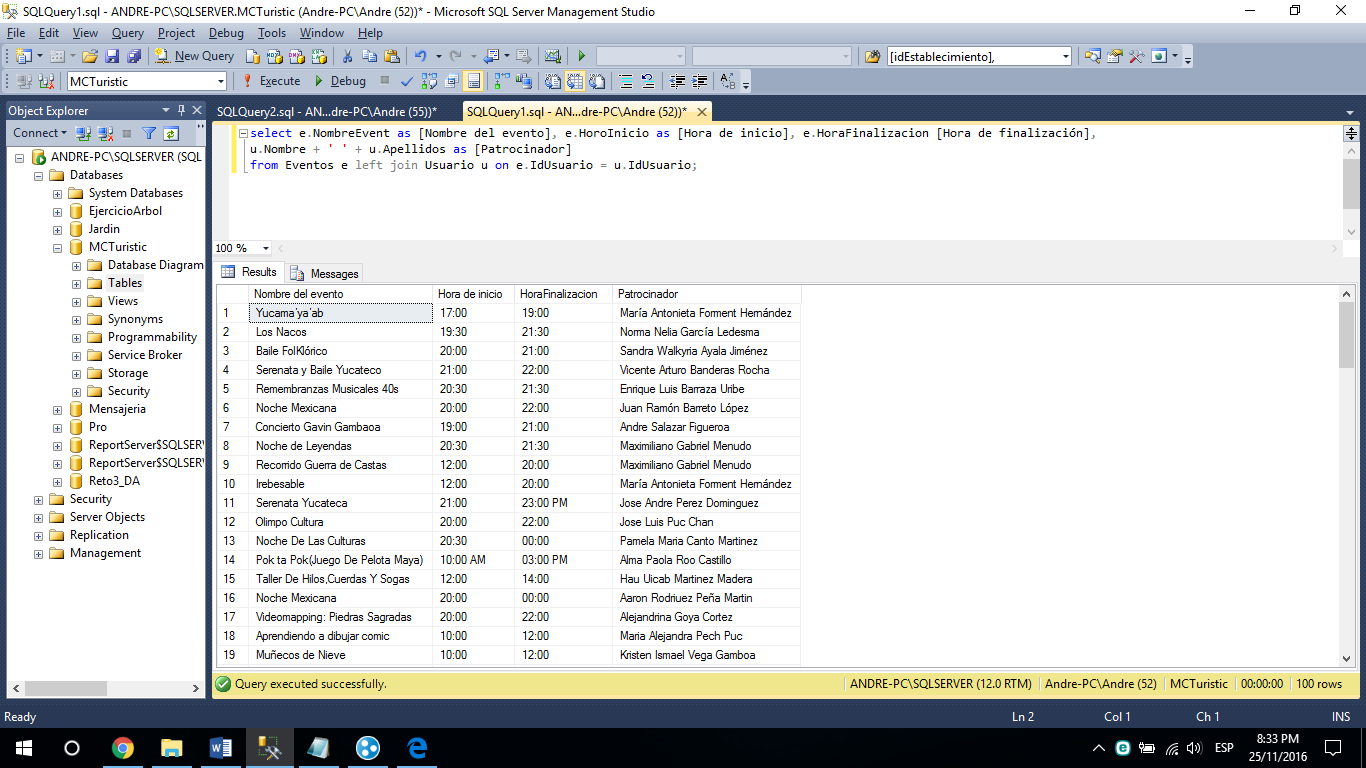
from Usuario u inner join Direccion d on d.IdUsuario = u.IdUsuario

### El resultado ayuda a visualizar con detalle los datos del usuario y así poder manipularlos al momento de llamarlos en el programa

### **Selección de datos de los eventos y sus patrocinadores**

select e.NombreEvent as [Nombre del evento], e.HoroInicio as [Hora de inicio], e.HoraFinalizacion [Hora de finalización],

u.Nombre + ' ' + u.Apellidos as [Patrocinador]

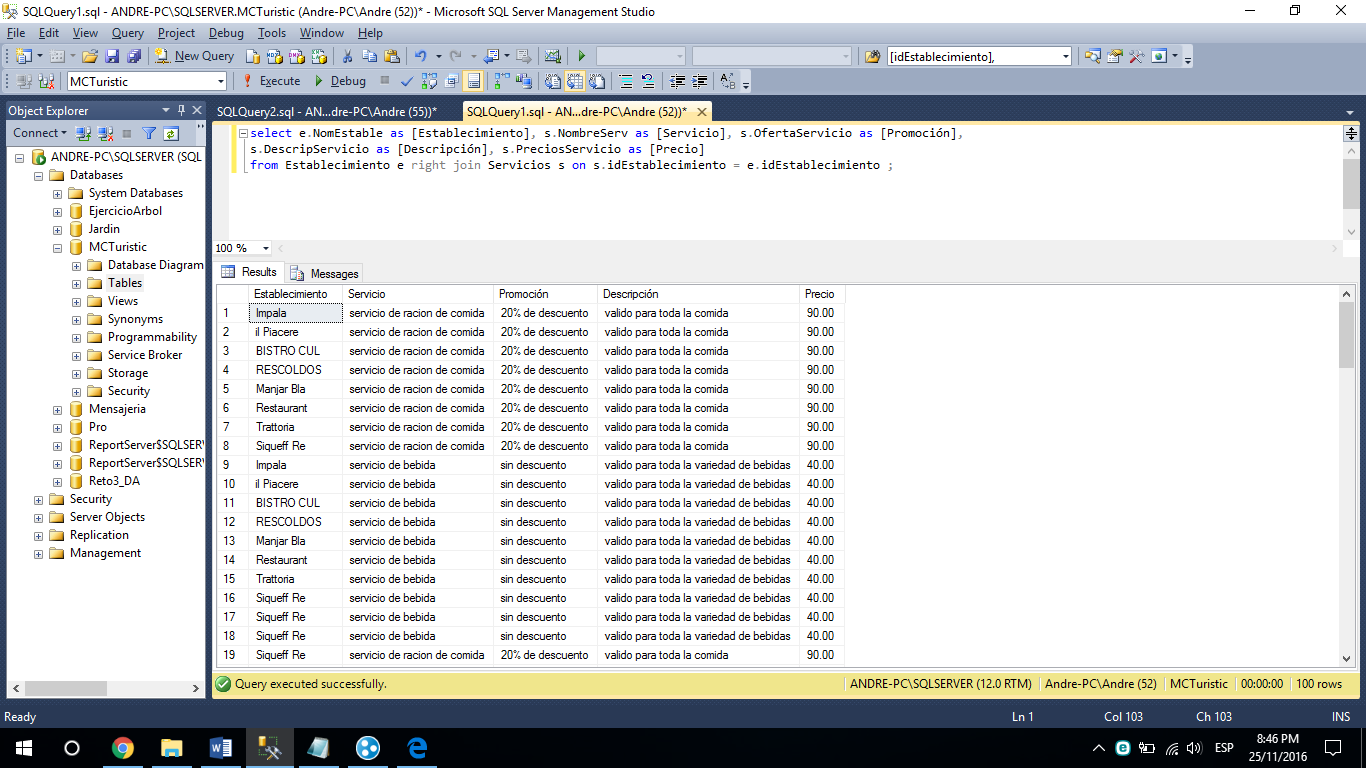
from Eventos e left join Usuario u on e.IdUsuario = u.IdUsuario;

### El resultado servirá para visualizar el nombre, la hora de inicio y finalización de los eventos y quién los patrocina, por medio de estos datos se podrán complementar las notificaciones para los usuarios del sistema.

### **Selección de servicios que ofrecen los establecimientos**

select e.NomEstable as [Establecimiento], s.NombreServ as [Servicio], s.OfertaServicio as [Promoción],

s.DescripServicio as [Descripción], s.PreciosServicio as [Precio]

from Establecimiento e right join Servicios s on s.idEstablecimiento = e.idEstablecimiento ;

### El resultado servirá para promocionar cada servicio ofertado por los establecimientos en la plataforma web. Así cada usuario podrá visualizarlos y seleccionar la opción que desee para más detalles.

# Conclusiones individúales

## José Luis Puc Chan

### Una de las partes importantes para que nuestro proyecto funcionara fue la base de datos con todos los beneficios que esta nos ofrece, al explotar los recursos de nuestro gestor de datos me puede dar cuenta que todo los datos son importantes tanto para el usuario como para el sistema, se implementó disparadores para determinar si existía algún usuario con el mismo registro entre varias otras cosas, de esta manera pudimos enriquecer más nuestra aplicación y de esta manera pueda ser más redituable.

## Diosemir Isael Nah May

### En conclusión con referencia a este documento puedo dar mi punto de visto de lo fácil y rápido que se puede trabajar en equipo, este documento fue realizado de una manera bien organizada donde cada integrante se pudo desenvolver en los apartados que se les hacía más rápidos, de esta manera este documento se fue estructurando cuidando los puntos que nos fueron establecidos.

### Este documento contiene la mayoría de información con respecto a ADB (Administración De Base De Datos) sobre nuestro proyecto llamado “MCTuristic” un desarrollo creado por nuestro equipo “Master Coder” donde podemos darnos cuenta de la capacidad de datos de nuestro desarrollo web. Así como también se verán los cambios evolutivos que se realizara por medio de un modelo evolutivo que nos llevó a una solo definición como equipo que es la continuación del proyecto que nosotros desarrollaremos para nuestra aplicación móvil. De esta manera darle una forma más fácil y útil a un consumidor o usuario de poder interactuar con nuestro servicio, tomando así sus datos de referencia para alimentar nuestra gran base de datos.

## Yair Roberto Vega Gamboa

### En particular en mi conclusión recalco que esta elaboración de documento y secuencias de SQL server nos sirve de mucho aprendizaje, ya que hacemos práctica de la teoría vista en clase y logramos comprender su funcionamiento que tiene, con la elaboración de las vistas, índices, manejo de restricciones y procedimientos almacenados doy por concluido el entendimiento de cómo funcionan y para qué sirven. Se finaliza al igual la documentación del proyecto que nos sirve como base de los requerimientos que pide la aplicación web y móvil para su realización. En este periodo de realización logre entender la importancia de hacer respaldos de la base de datos y registros para su posterior uso o requerimiento.

## José André Salazar Figueroa

### El uso de cada una de las herramientas del gestor de base de datos nos permitió declarar cada elemento y llenarlos de datos relacionados a cada uno. Cuando se realizó la creación de la base de datos se hicieron restricciones que nos ayudarán a delimitar los datos ingresados, al igual que se realizaron vistas que permitirán visualizar una o más tablas en una sola solicitud. Se utilizaron todos los elementos necesarios para optimizar y dejar lo más completa la base de datos. Todo esto nos servirá mucho para el desarrollo de la plataforma web y la aplicación móvil.

## Aarón Jesús Peña Martín

### Con SQL nos permite ingresar comandos o sentencias de tal manera que podemos administrar o crear una base de datos. Esta variedad de comandos nos permite generar información o datos desde la creación, modificación o mantenimiento a tablas las cuales también nos permite recuperar datos o importarlos de varias maneras.

### En este proyecto he aprendido a implementar varias funciones en la base de datos como disparadores, vistas, índices, diccionario de datos, etc. La seguridad en las bases de datos es muy importante debido a que garantiza la integridad física y lógica de los datos. La migración o exportación de datos se puede dar de varias maneras, ejemplo como: Archivos de tipo Texto, hojas de cálculo, archivo de texto sin formato, etc. Dentro de la nube de internet debido a la confiabilidad de las bases de datos. Estas mismas pueden ser manipuladas desde aplicaciones de internet de forma eficiente y segura.