Universidad Rafael Landivar

Facultad de ingeniería

Programación Web

Anleu Rosales Samuel Aarón

Documentación técnica del proyecto de curso

Edgar Sebastian Milián Mejía

1161918

Guatemala 26, abril del 2023

Contenido

[Análisis y diseño 3](#_Toc133362346)

[Definición de proyecto y requerimientos 3](#_Toc133362347)

[Diagrama de casos de uso 4](#_Toc133362348)

[Actividades 5](#_Toc133362349)

[Diagrama de base de datos 7](#_Toc133362350)

[Arquitectura de la aplicación 8](#_Toc133362351)

[API 8](#_Toc133362352)

[Endpoints 8](#_Toc133362353)

[Mensajes (objetos) 10](#_Toc133362354)

[Controladores 14](#_Toc133362355)

[Capa de presentación MVC 15](#_Toc133362356)

[Controladores 15](#_Toc133362357)

[Librería de modelos 15](#_Toc133362358)

[Paquetes Nugget 16](#_Toc133362359)

[Base de datos 16](#_Toc133362360)

[Tablas 16](#_Toc133362361)

# Análisis y diseño

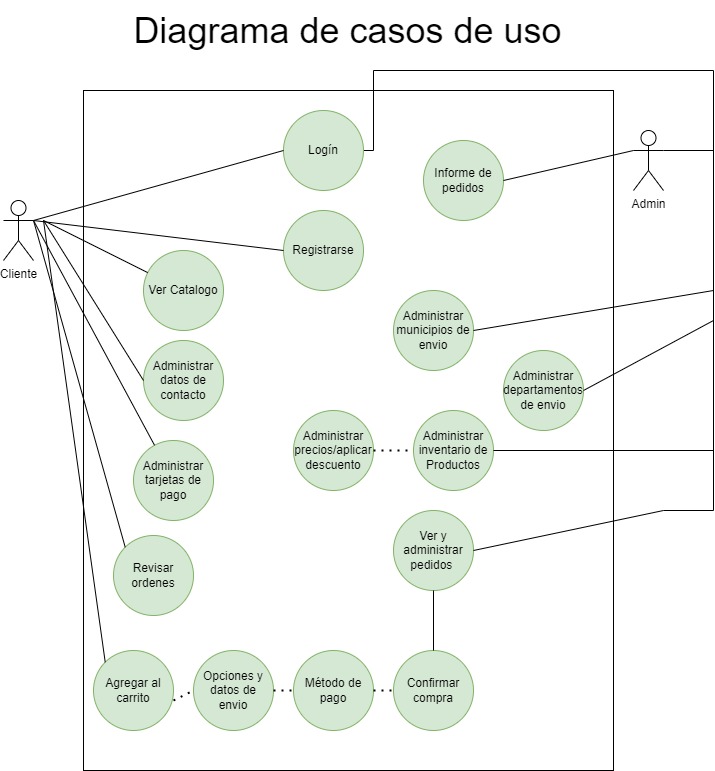
## Definición de proyecto y requerimientos

Sistema de tienda digital

Un almacén de productos nos ha contratado para la elaboración de una página web que les permita entrar al comercio digital, permitiendo a los usuarios hacer sus compras en línea de los productos, gestionar direcciones y sus métodos de pago por el lado de los clientes para realizar pedidos, además de ser lo más autoadministrable posible de acuerdo con las transacciones a realizar. Por el lado administrativo de los empleados y el gerente encargado administrar los productos del catálogo, el inventario, cambiar precios y hacer algunos reportes y/o estadísticas de ser posible que les informe sobre nuevos pedidos, también poder gestionar hacía donde se pueden realizar los envíos por departamento y municipio y que el usuario pueda agregar su dirección personalizada, el cliente también debe contar con un carrito de compra y poder seguir el estado de sus órdenes y a los administradores poder ver las órdenes y administrarlas. (Por el momento el almacén es de productos Varios sin una temática específica por lo que se debe poder agregar productos de cualquier índole.)

(El diagrama de casos de uso que se encuentra en la siguiente página puede estar sujeto a cambios del alcance original ampliando sus funciones.)

## Diagrama de casos de uso



## Actividades

Ambos (Cliente y admin):

Login: El cliente ingresa al inicio del sistema, donde encuentra un formulario donde debe ingresar su correo y contraseñas registrados para poder entrar a su portal de cliente.

Registro: El cliente que quiere registrarse en el sistema ingresa el correo que va a utilizar para ingresar al sistema y también la contraseña para poder entrar a su portal.

Cliente:

Ver catalogo: Deslizarse por el portal al momento de haber entrado luego del login.

Administrar direcciones de contacto: En la barra lateral debe encontrar una sección donde acceder a los CRUD para administrar sus direcciones de contacto para crear, editar y eliminar sus direcciones de contacto.

Administrar tarjetas de pago: En la barra lateral debe encontrar una sección donde acceder a los CRUD para administrar las tarjetas con las que va a realizar las órdenes.

Agregar al carrito: En la sección de la tienda, la pagina principal al iniciar sesión debe poder ingresar a un producto, ver algunos detalles que sean propiedades del producto y agregarlo a un carrito de compra el cual también es accesible desde la barra lateral del sitio donde puede administrarlo para eliminar objetos o realizar la compra. Al realizar la compra debe seleccionar la dirección de contacto para enviar, seleccionar la tarjeta de pago y crear la compra.

Ver Ordenes: En la barra lateral la opción de ordenes permite al usuario ver las ordenes realizadas y los objetos asociados en la orden.

Administrador:

Administrar departamentos: En la barra lateral encontrará la sección de departamentos donde accederá al CRUD para administrar los departamentos para crear, editar y eliminar.

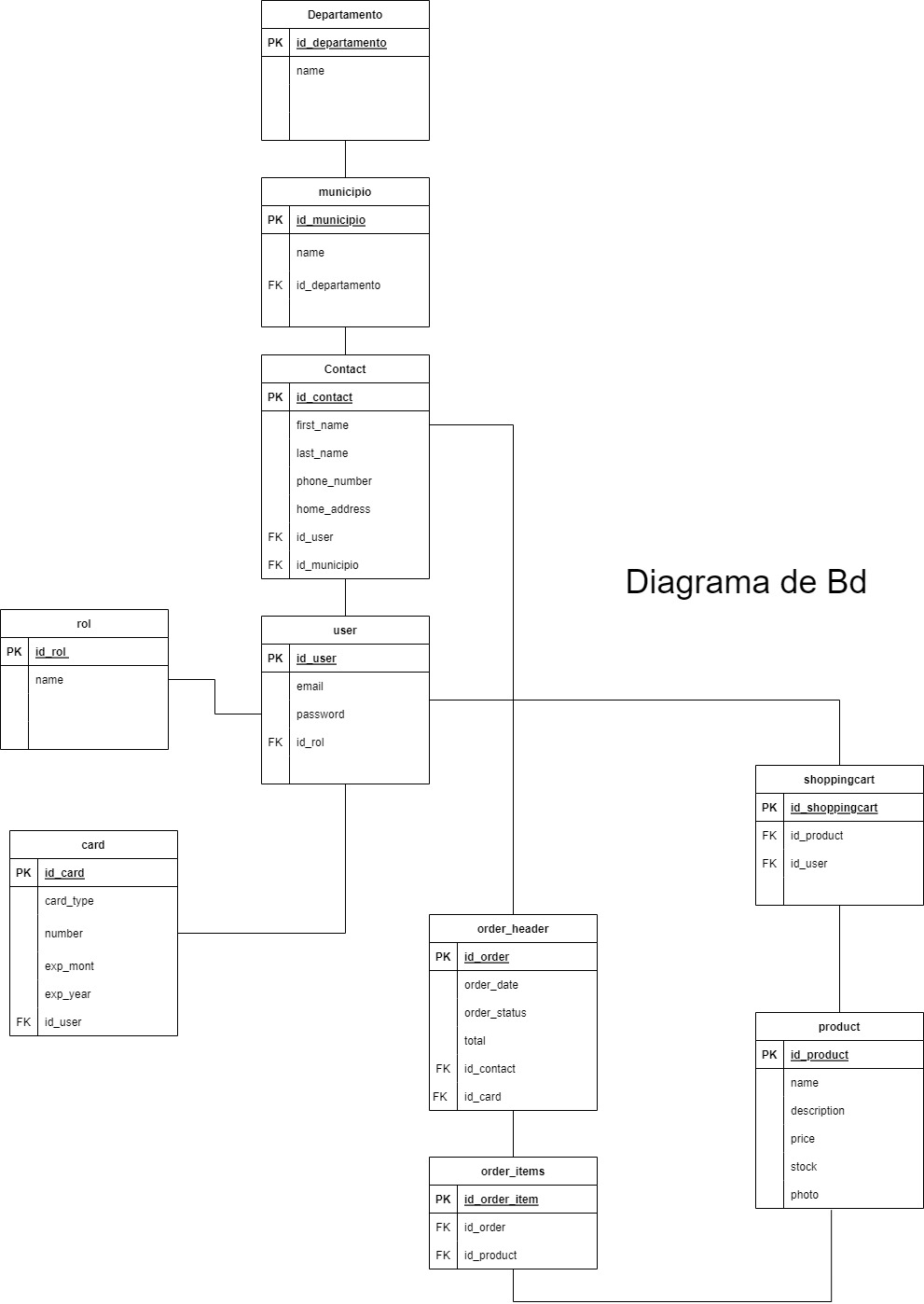
Administrar municipios: En la barra lateral encontrará la sección de departamentos donde accederá al CRUD para administrar los departamentos para crear, editar y eliminar.

Informe de pedidos: En la pagina principal del Amin al iniciar sesión debe poder encontrar los datos de como van los pedidos, pedidos nuevos, confirmados, enviados y entregados.

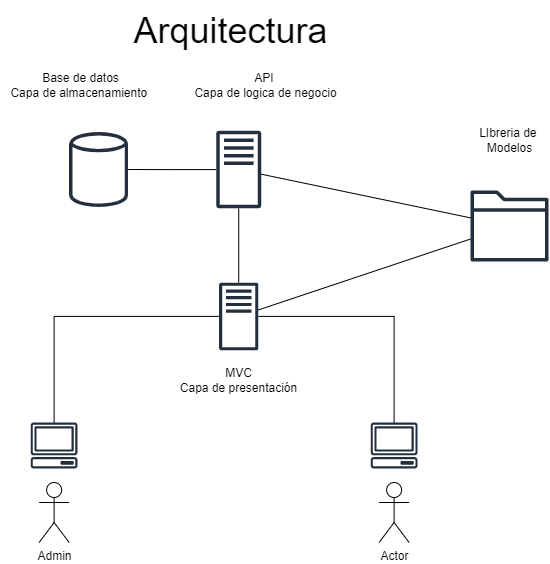
Productos: En la barra lateral se debe encontrar el acceso a los CRUD de los productos para crear, eliminar, editar, permite fotografías.

Ordenes: En el menú de barra lateral se accede a las ordenes que realizan los clientes y se puede administrar el estatus de la orden para confirmarla, enviarla o darla de entregada.

## Diagrama de base de datos

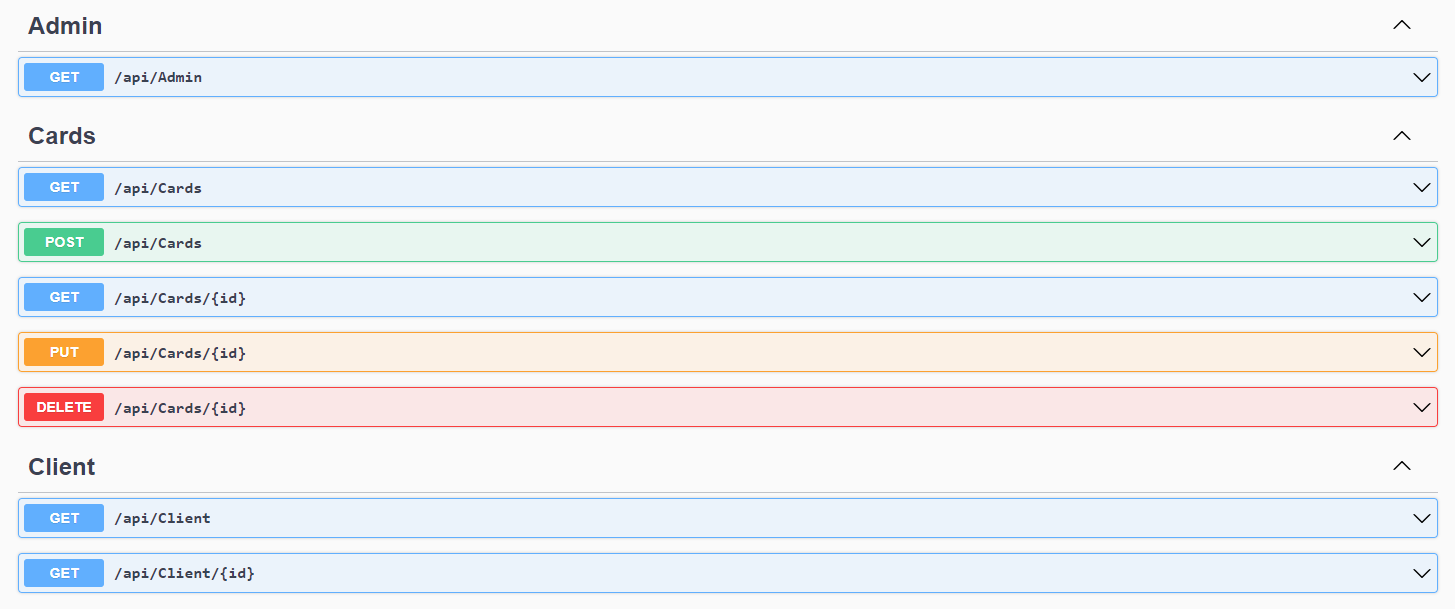


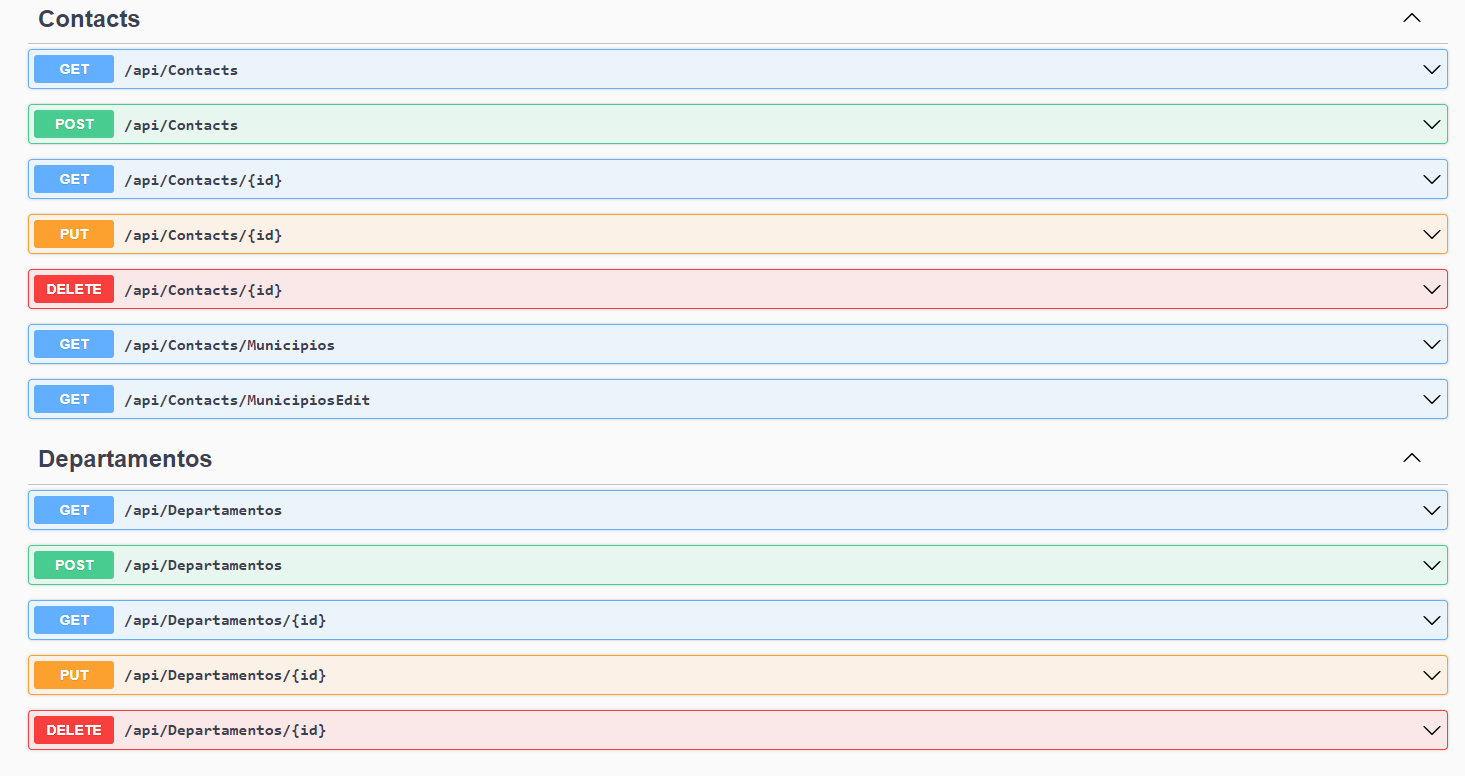
## Arquitectura de la aplicación



# API

## Endpoints









## Mensajes (objetos)

Departamento**{**

|  |  |
| --- | --- |
| **idDepartamento** | **integer($int32)** |
| **name** | **string*: true*** |
|  |  |

**}**

Card**{**

|  |  |
| --- | --- |
| **idCard** | **integer($int32)** |
| **cardtype** | **string *nullable: true*** |
| **number** | **string *nullable: true*** |
| **expMonth** | **string *nullable: true*** |
| **expYear** | **string *nullable: true*** |
| **idUser** | **integer($int32) *nullable: true*** |
|  |  |
|  |  |

**}**

Contact**{**

|  |  |
| --- | --- |
| **idContact** | **integer($int32)** |
| **firstName** | **string *nullable: true*** |
| **lastName** | **string *nullable: true*** |
| **phoneNumber** | **string *nullable: true*** |
| **homeAddress** | **string *nullable: true*** |
| **idUser** | **integer($int32) *nullable: true*** |
| **idMunicipio** | **integer($int32) *nullable: true*** |

**}**

ContactInfo**{**

|  |  |
| --- | --- |
| **contact** | **Contact{...}** |
| **municipios** | **[...]** |
| **departamento** | **Departamento{...}** |
| **municipio** | **Municipio{...}** |

**}**

Kpis**{**

|  |  |
| --- | --- |
| **porConfirmar** | **integer($int32)** |
| **confirmadas** | **integer($int32)** |
| **enviadas** | **integer($int32)** |
| **entregadas** | **integer($int32)** |

**}**

Municipio**{**

|  |  |
| --- | --- |
| **idMunicipio** | **integer($int32)** |
| **name** | **string *nullable: true*** |
| **idDepartamento** | **integer($int32) *nullable: true*** |

**}**

OrderHeader**{**

|  |  |
| --- | --- |
| **idOrder** | **integer($int32)** |
| **orderDate** | **string($date-time) *nullable: true*** |
| **orderStatus** | **string *nullable: true*** |
| **total** | **string *nullable: true*** |
| **idCard** | **integer($int32) *nullable: true*** |
| **idContact** | **integer($int32) *nullable: true*** |

**}**

OrderItem**{**

|  |  |
| --- | --- |
| **idOrderItem** | **integer($int32)** |
| **idProduct** | **integer($int32) *nullable: true*** |
| **idOrder** | **integer($int32) *nullable: true*** |

**}**

Product**{**

|  |  |
| --- | --- |
| **idProduct** | **integer($int32)** |
| **name** | **string *nullable: true*** |
| **description** | **string *nullable: true*** |
| **price** | **string *nullable: true*** |
| **stock** | **integer($int32)** |
| **photo** | **string *nullable: true*** |

**}**

Shoppingcart**{**

|  |  |
| --- | --- |
| **idShoppingcart** | **integer($int32)** |
| **idProduct** | **integer($int32) *nullable: true*** |
| **idUser** | **integer($int32) *nullable: true*** |

**}**

User**{**

|  |  |
| --- | --- |
| **idUser** | **integer($int32)** |
| **email** | **string *nullable: true*** |
| **password** | **string *nullable: true*** |
| **idRole** | **integer($int32) *nullable: true*** |

**}**

## Controladores

AdminController: Este controlador se encarga de devolver KPIS sobre las ordenes que se han hecho, enviado, confirmado y entregado no recibe ningún objeto solo devuelve sobre lo calculado en la bd devuelve un listado de productos.

CardsController: Este controlador permite realizar las operaciones CRUD de las tarjetas de pago permite crear, eliminar y editar recibiendo objetos de tipo CARD.

ClientController: Este controlador devuelve todos los productos de la tienda para mostrarlos en el portal del cliente en el portal inicial devuelve listado de productos solo aparecen los productos cuyo stock sea mayor a 0.

Contact controller: Este controlador se encarga de realizar las operaciones CRUD en las direcciones de entrega de los clientes pueden crear, editar y eliminar sus direcciones.

DepartamentoController: Este controlador permite a los administradores realizar las operaciones CRUD de departamentos crear, eliminar y editar los departamentos a los que se puede hacer envíos.

HomeController: Se encarga del inicio de sesión y registro de los clientes e inicio de sesión de administradores.

MunicipiosController: Este controlador permite a los administradores realizar las operaciones CRUD de municipios crear, eliminar y editar los departamentos a los que se puede hacer envíos.

OrderHeaders: Este controlador maneja las ordenes del lado del cliente por lo que solo obtiene las ordenes realizadas y también realiza la creación de la orden al comprar en el carrito.

OrderHeadersAdminController: Este controlador permite a los administradores ver las ordenes los ítems y también cambiar el estado de la orden que en consecuencia le aparece al cliente también.

OrderItemController: Este controlador únicamente tienen el objetivo de retornar los objetos de una orden para verlos en pantalla.

OrderItemAdminController: Este controlador igual tiene el objetivo de retornar objetos de orden pero para la pantalla del administrador.

ProductController: Este controlador permite a los administradores manejar el inventario de productos para crear nuevos productos, editar precios y stock como también eliminar productos.

ShoppingCartController: Este controlador se encarga de almacenar los productos que un cliente agrega al carrito y mostrarlos para proceder a la compra.

# Capa de presentación MVC

## Controladores

AdminController: Este controlador maneja la vista principal con la información de los estatus de las ordenes.

Clientcontroller: Este controlador maneja la vista principal de la tienda a los clientes trayendo los productos para que los vean y puedan agregarlos al carrito También maneja las tarjetas y direcciones de contacto de los clientes.

DepartamentosController: Maneja las vistas para administrar departamentos, con operaciones CRUD.

MunicipiosController: Maneja las vistas CRUD de los municipios para administrarlos.

OrderHeadersAdminController: Maneja las vistas de las ordenes que hacen los clientes por parte del administrador.

OrderHeaderController: Maneja las vistas de las ordenes de los clientes para que puedan monitoreando.

OrderItemsController: Maneja la vista de los objetos de las ordenes.

ProductController: Este controlador maneja la vista de administración de inventario o productos para crear productos, eliminarlos o editar su stock.

ShoppingCart: Maneja la vista para objetos a ordenar.

# Librería de modelos

La API y MVC comparten una librería de clases que utilizan las aplicaciones para validar y utilizar de referencia además de contar con algunos modelos que agrupan otros modelos la lista de modelos es la siguiente

* Cards
* Contact
* Departamento
* Municipio
* OrderHeader
* OrderItem
* Product
* Role
* ShoppingCart
* StoreCOntext
* User

# Paquetes Nugget

Para conexión a la base de datos se utilizó Entity Framework como ORM el cual hace uso de tres paquetes nugget

* MySql.Data
* MySql.EntityFrameworkCore
* Microsoft.EntityFrameworkCore.Tools

Para realizar peticiones HTTP con formato json de forma más simple

* Microsoft.AspNet.WebApi.Client

# Base de datos

## Tablas

CREATE TABLE roles (

id\_role int primary key auto\_increment,

name varchar (30) not null unique

);

CREATE TABLE users(

id\_user int primary key auto\_increment,

email varchar (50) not null unique,

password varchar(50) not null,

id\_role int,

constraint fk\_user\_rol foreign key (id\_role) references roles(id\_role)

);

CREATE TABLE departamento(

id\_departamento int primary key auto\_increment,

name varchar(50) not null

);

CREATE TABLE municipio (

id\_municipio int primary key auto\_increment,

name varchar(50) not null,

id\_departamento int,

constraint fk\_municipio\_departamento foreign key (id\_departamento) references departamento(id\_departamento) ON DELETE CASCADE

);

CREATE TABLE contact(

id\_contact int primary key auto\_increment,

first\_name varchar(50) not null,

last\_name varchar(50) not null,

phone\_number varchar(8) not null,

home\_address varchar(50) not null,

id\_user int,

id\_municipio int,

constraint fk\_contacts\_user foreign key (id\_user) references users(id\_user) ON DELETE CASCADE,

constraint fk\_contacts\_municipio foreign key (id\_municipio) references municipio(id\_municipio)ON DELETE CASCADE

);

CREATE TABLE card(

id\_card int primary key auto\_increment,

cardtype varchar(20) not null,

number varchar(16) not null,

exp\_month varchar(2) not null,

exp\_year varchar(2) not null,

id\_user int,

constraint fk\_card\_user foreign key (id\_user) references users(id\_user) ON DELETE CASCADE

);

CREATE TABLE product(

id\_product int primary key auto\_increment,

name varchar(50) not null,

description varchar (300) not null,

price varchar(50) not null,

stock int not null,

photo varchar(500) not null

);

CREATE TABLE shoppingcart (

id\_shoppingcart int primary key auto\_increment,

id\_product int,

id\_user int,

constraint fk\_cart\_user foreign key (id\_user) references users (id\_user) ON DELETE CASCADE,

constraint fk\_cart\_product foreign key (id\_product) references product(id\_product) ON DELETE CASCADE

);

drop table order\_item;

drop table order\_header;

CREATE TABLE order\_header (

id\_order int primary key auto\_increment,

order\_date datetime,

order\_status varchar(50),

total varchar(50),

id\_card int,

id\_contact int,

constraint fk\_order\_card foreign key (id\_card) references card (id\_card) ON DELETE SET NULL,

constraint fk\_order\_contact foreign key (id\_contact) references contact (id\_contact) ON DELETE SET NULL

);

CREATE TABLE order\_item (

id\_order\_item int primary key auto\_increment,

id\_product int,

id\_order int,

constraint fk\_item\_order foreign key(id\_order) references order\_header(id\_order) ON DELETE SET NULL,

constraint fk\_item\_product foreign key (id\_product) references product (id\_product) ON DELETE SET NULL

);

insert into roles (name) values ('admin');

insert into roles (name) values ('user');

insert into roles (name) values ('client');

insert into users (email, password, id\_role) values('adminuser@gmail.com','test123',1);