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

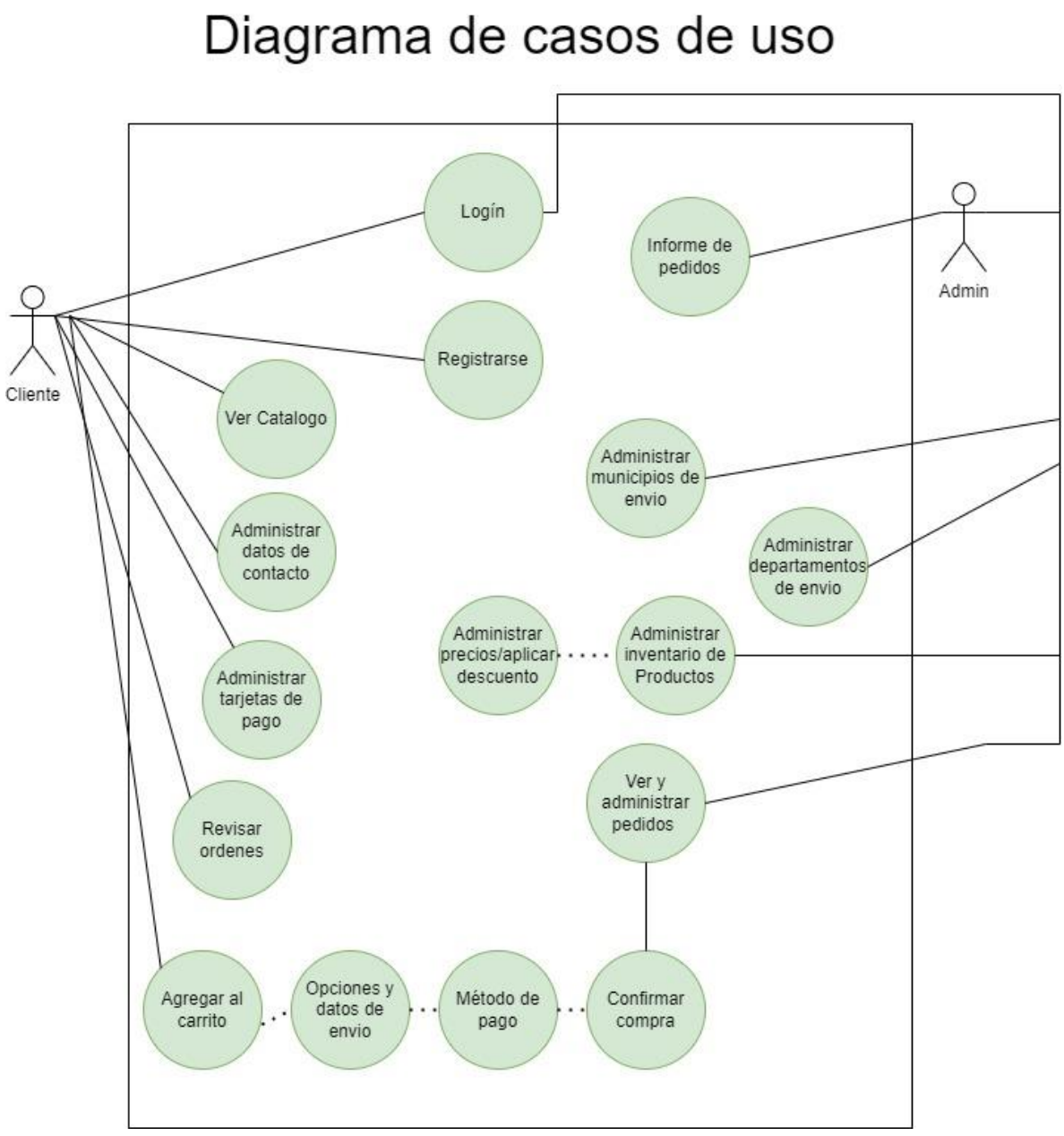
Definición de proyecto y requerimientos

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

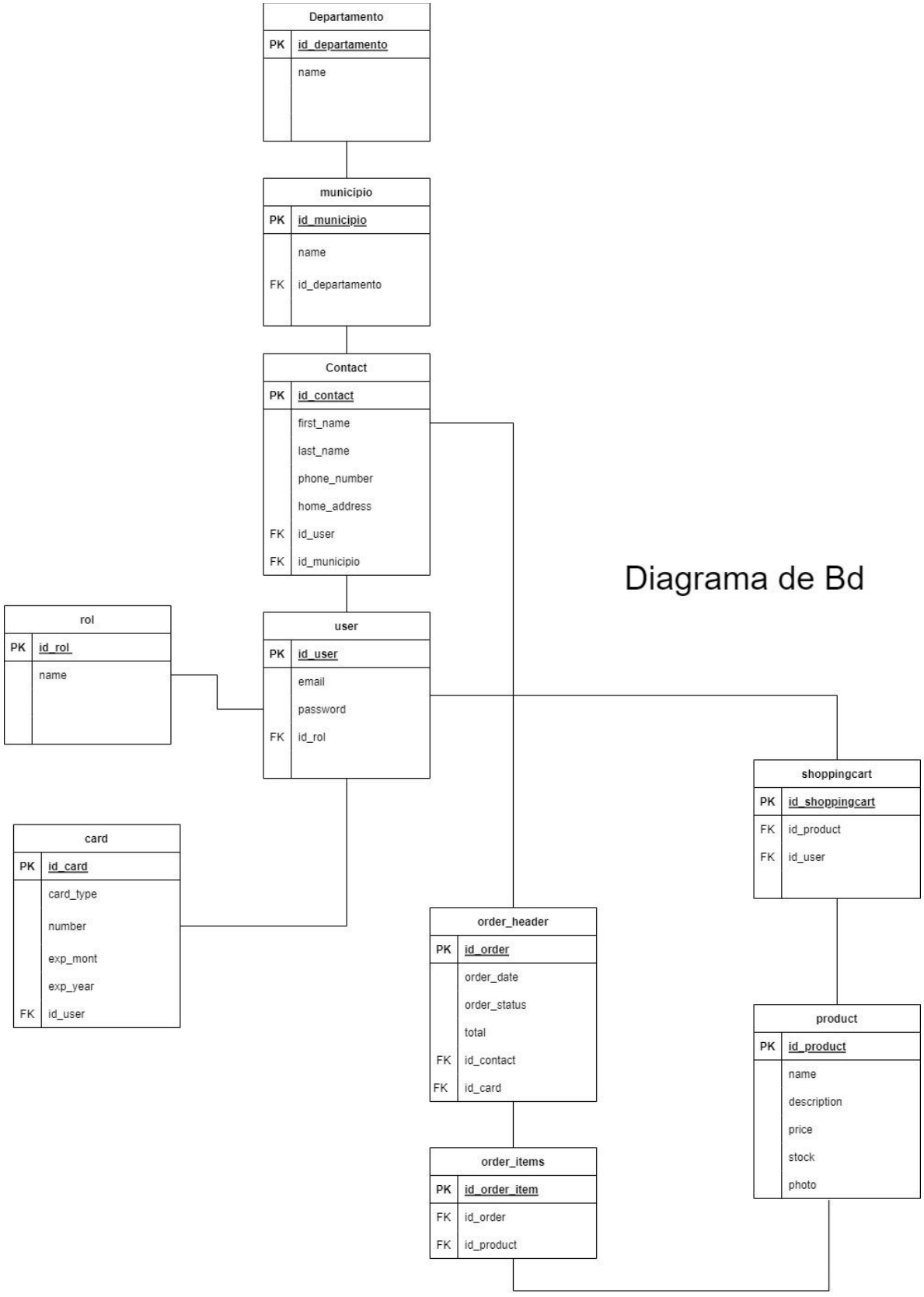
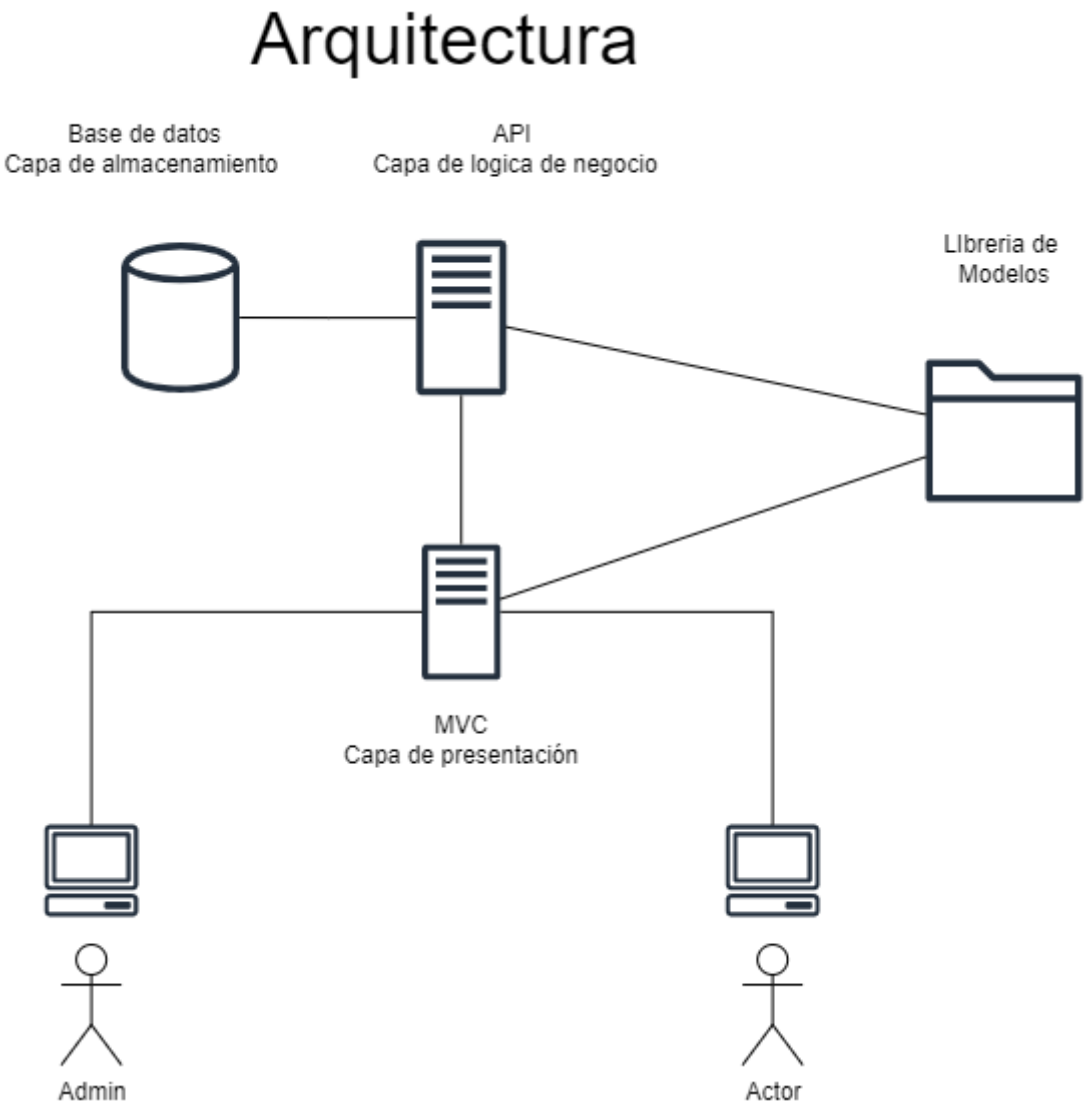


Diagrama de Bd

Arquitectura de la aplicación



API

Endpoints

Admin

- GET /api/Admin

Cards

- GET /api/Cards
- POST /api/Cards
- GET /api/Cards/{id}
- PUT /api/Cards/{id}
- DELETE /api/Cards/{id}

Client

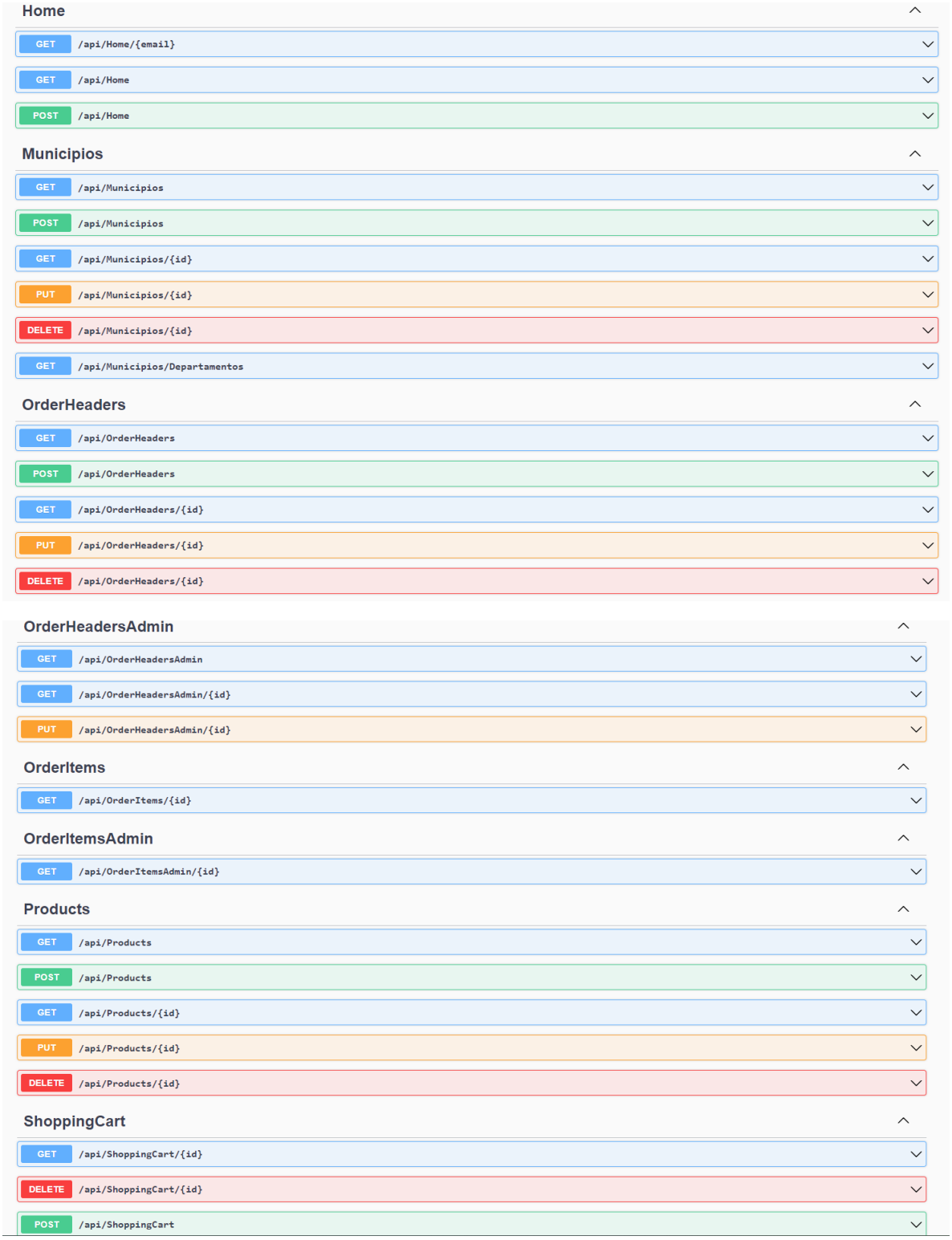
- GET /api/Client
- GET /api/Client/{id}

Contacts

- GET /api/Contacts
- POST /api/Contacts
- GET /api/Contacts/{id}
- PUT /api/Contacts/{id}
- DELETE /api/Contacts/{id}
- GET /api/Contacts/Municipios
- GET /api/Contacts/MunicipiosEdit

Departamentos

- GET /api/Departamentos
- POST /api/Departamentos
- GET /api/Departamentos/{id}
- PUT /api/Departamentos/{id}
- DELETE /api/Departamentos/{id}



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

Para el cifrado de contrasenias se hizo uso de BCrypt con el hash de contrasenias

Para el manejo de JWT se usaron los paquetes

- Microsoft.AspNetCore.Authentication.JwtBearer
- System.IdentityModel.Tokens.Jwt

Base de datos

Tablas

```
drop table order_item;

drop table order_header;

DROP TABLE card;

drop table contact;

drop table municipio;

drop table departamento;

drop table shoppingcart;

DROP TABLE users;

drop table roles;

drop table product;


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(200) 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) 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
);

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 CASCADE,
constraint fk_order_contact foreign key (id_contact) references contact (id_contact) ON DELETE CASCADE
);

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 CASCADE,

constraint fk_item_product foreign key (id_product) references product (id_product)
ON DELETE CASCADE

);



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','$2a$11$oBMx.WdW//tkAlLraK/Hs.kzuyhRjCq1h9n
cCEADGbpg2Y8CCx6mS',1);
```