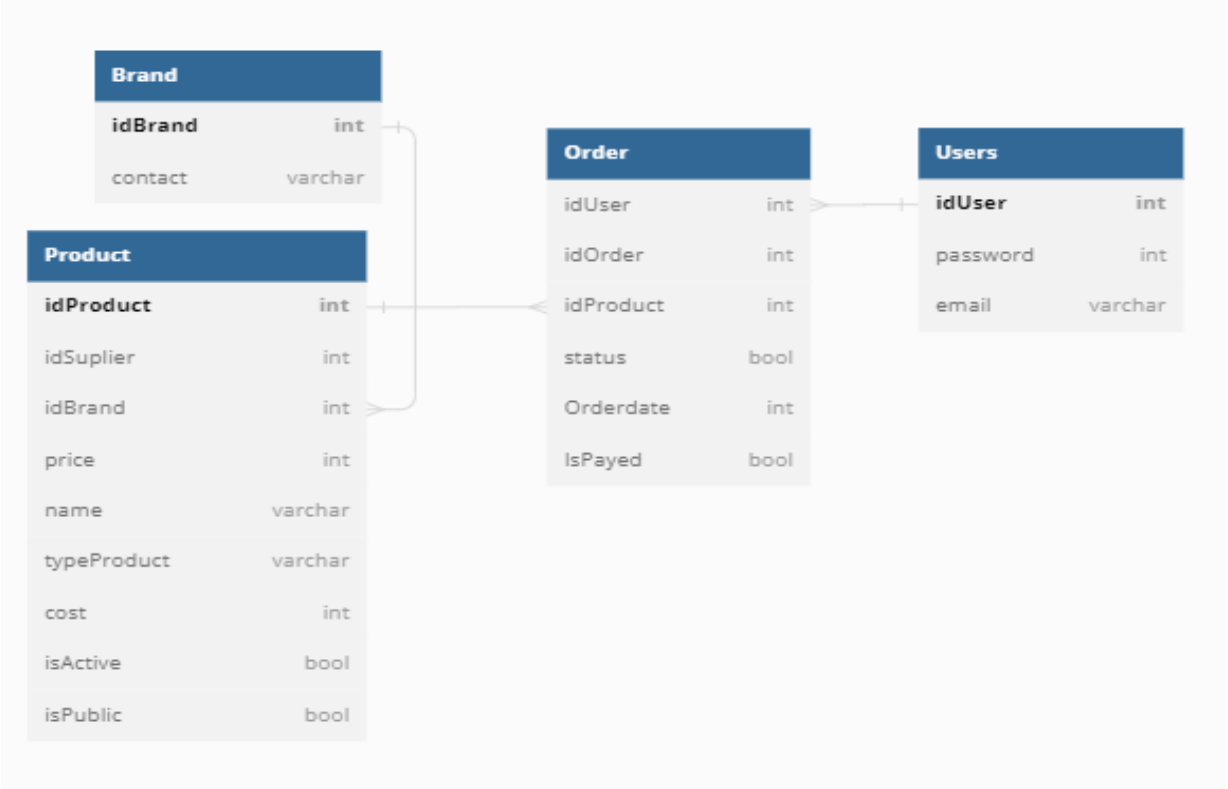
Prototipo Aplicación API

Ainhoa Cala Bustos

1. Negocio elegido y fundamentación.
La aplicación está orientada a un comercio al por menor, en concreto es una tienda de alimentación especializada, que muestran muchos problemas con el programa que utiliza desde hace años, ya que no le permite actualizar los productos de manera sencilla, muestra muchos errores en el stock y en ocasiones se desactualizan los datos.

Por lo que la empresa, en lugar de usar uno de los programas estándar utilizados en el sector, ha optado por el diseño de una aplicación adaptada por completo a sus necesidades.

2. Diseño de tablas.
Por consiguiente, a continuación podemos ver cómo se estructura la API, a través de tablas planteando las conexiones entre sí.



Cada tabla se encuentra vinculada por el id, y nos proporcionará la información presente en la misma, la tabla producto estará vinculada con la tabla marca, por lo que cualquier modificación permitirá la actualización de la información. También contamos con la tabla user, en la que ingresando email y contraseña podremos acceder a diferentes permisos, de igual forma los clientes que ingreses podrán ver sus pedidos y poder consultar el estado en el que se encuentran.

3. Tecnologías, lenguajes y versiones.
Las tecnologías utilizadas para el desarrollo de esta aplicación son:

- Microsoft Visual Studio 2022
- Microsoft SQL Server Management Studio 18
- Lucidchart Diagramas
- Git Hub

Y los lenguajes implementados son:

- C#
- Net.Core
- Entity Framework 7.0
- SQL

4. Documentación funcional:

Objetivos

- Registro de producto, pudiendo realizar las funcionalidades de agregar, modificar y eliminar, tanto el producto, como el stock del mismo, y las marcas o proveedores.
- Registro y gestión de usuarios, administrador y cliente (con los respectivos permisos).
- Gestión de pedidos, pudiendo saber el estado de cada uno.

Arquitectura

El diseño seleccionado es la estructura por capas.

Controlles y métodos

La aplicación cuenta con cuatro entities cada una asociada con una tabla, user, order, product y brand. Los métodos implementados en cada una de ellas con los de POST, GET, PATCH y DELETE.

A continuación pasaremos a especificar la funcionalidad de cada uno de ellos.

Brand

- POST

Request body

```
{
  "id": 0,
  "isActive": true,
  "contact": "string"
}
```

-GET

```
{
  "id": 0,
  "isActive": true,
  "contact": "string"
}
```

-PATCH

Request body

```
{
  "id": 0,
  "isActive": true,
  "contact": "string"
}
```

-DELETE

id

```
{
  "id": 0,
  "isActive": false,
  "contact": "string"
```

}

Order
-POST
Request body

```
{
  "idUser": 0,
  "id": 0,
  "productId": 0,
  "lot": 0,
  "orderDate": "2023-02-05T13:36:47.151Z",
  "status": true,
  "isPayed": true,
  "isActive": true
}
```


-GET

```
{
  "idUser": 0,
  "id": 0,
  "productId": 0,
  "lot": 0,
  "orderDate": "2023-02-05T13:36:47.151Z",
  "status": true,
  "isPayed": true,
  "isActive": true
}
```


-PATCH

```
{
  "idUser": 0,
  "id": 0,
  "productId": 0,
  "lot": 0,
  "orderDate": "2023-02-05T13:36:47.151Z",
  "status": true,
  "isPayed": true,
  "isActive": true
}
```


-DELETE
Id

```
{
  "idUser": 0,
  "id": 0,
  "productId": 0,
```

```
  "lot": 0,
  "orderDate": "2023-02-05T13:36:47.151Z",
  "status": true,
  "isPayed": true,
  "isActive": false
}
```


Product

-POST

Request body


```
{
  "id": 0,
  "idSuplier": 0,
  "idBrand": 0,
  "price": 0,
  "name": "string",
  "cost": 0,
  "typeProduct": "string",
  "isActive": true,
  "isPublic": true
}
```


-GET


```
{
  "id": 0,
  "idSuplier": 0,
  "idBrand": 0,
  "price": 0,
  "name": "string",
  "cost": 0,
  "typeProduct": "string",
  "isActive": true,
  "isPublic": true
}
```


-DELETE

id


```
{
  "id": 0,
  "idSuplier": 0,
  "idBrand": 0,
  "price": 0,
  "name": "string",
  "cost": 0,
  "typeProduct": "string",
  "isActive": true,
  "isPublic": false
```

}

-PATCH
Request body
{
  "id": 0,
  "idSuplier": 0,
  "idBrand": 0,
  "price": 0,
  "name": "string",
  "cost": 0,
  "typeProduct": "string",
  "isActive": true,
  "isPublic": true
}

User
-POST
Request body
{
  "id": 0,
  "email": "string",
  "isActive": true
}

-GET
{
  "id": 0,
  "email": "string",
  "isActive": true
}

-PATCH
Request body
{
  "id": 0,
  "email": "string",
  "isActive": true
}

-DELETE
id
{
  "id": 0,
  "email": "string",
  "isActive": false
}

5. Extensiones y pendientes.

La aplicación aún se encuentra en su estado más primitivo, se debe retocar y perfeccionar la funcionalidad que ya tiene, además de incorporar aspectos como el filtrado de marcas, la identificación de usuario, la realización de pedidos y poder visualizar el estado del mismo, además de posibles modificaciones más avanzadas en referencias a la actualización del stock y a la contabilidad.