

Proyecto CRUD Rest



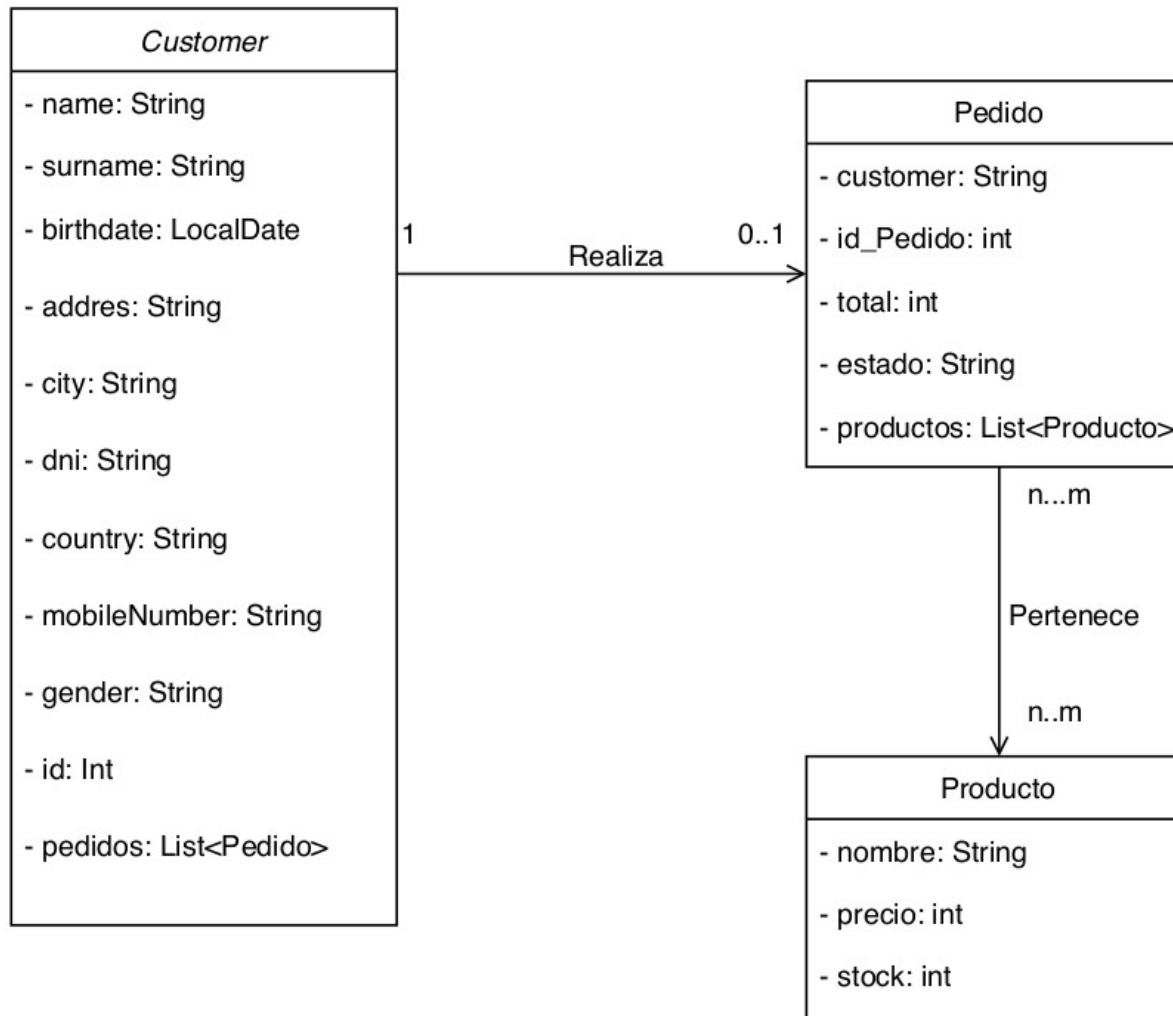
Introducción

- En mi proyecto he intentado recrear una tienda o almacén de productos.

Para ello la elaboración de mi solución consiste en:

- Una clase Customer que representa a los compradores
- Una clase Pedidos para almacenar los pedidos que se realizan
- Una clase Productos para almacenar los productos que hay en el almacén.

UML de las clases del proyecto

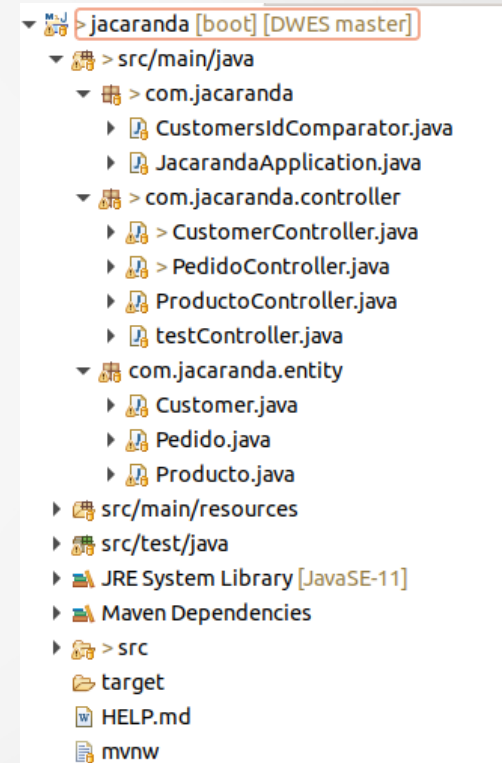


Estructura Java

- La estructura del proyecto en java es sencilla:

- Tenemos una entidad Customer, que además de sus atributos tiene una colección de Pedidos.

Cada pedido a su vez tiene sus propios atributos y una colección de Productos, estos últimos solo tienen sus atributos.



Controladores

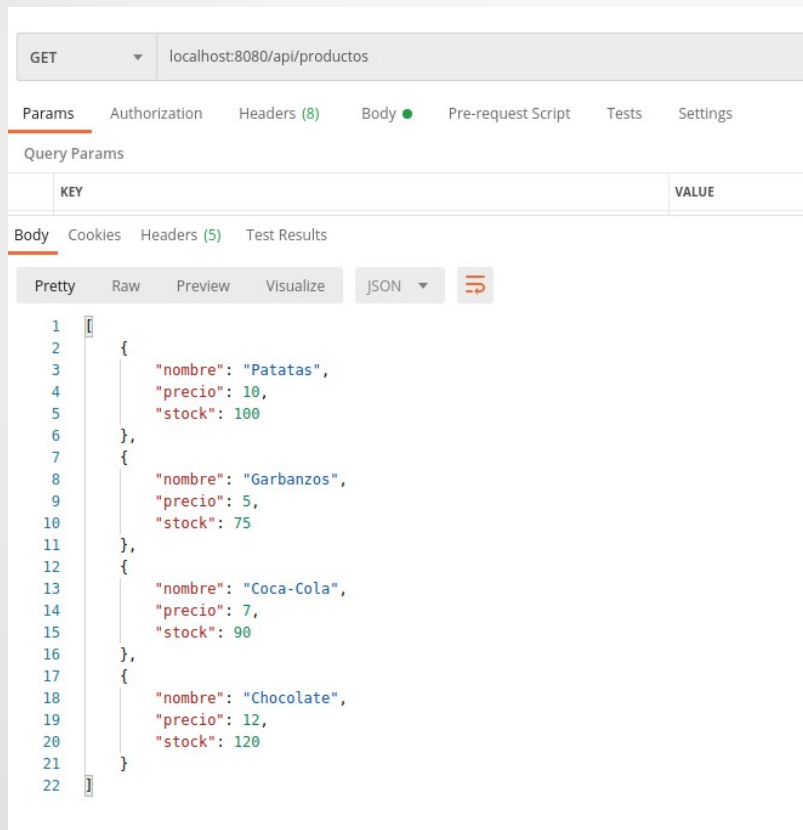
- Cada entidad tiene su controlador, donde se realiza el CRUD (GET, POST, PUT, DELETE). Además los controladores de Customer y Pedidos tienen un método POST específico para añadir pedidos a un customer y productos a un pedido respectivamente.

```
//Petición POST para añadir pedidos al customer indicado
@PostMapping("/customers/{id}")
public ResponseEntity<?> addPedido(@PathVariable int id, @RequestBody Pedido ped){
    ResponseEntity respuesta=ResponseEntity.status(HttpStatus.CONFLICT).body("FAILED");

    boolean encontrado=false;
    Iterator<Customer> custIterator= customers.iterator();
    while(custIterator.hasNext() && !encontrado) {
        Customer c= custIterator.next();
        if(c.getId()==id) {
            encontrado=true;
            c.getPedidos().add(ped);
            ped.setCustomer(c.getName());
            respuesta=ResponseEntity.status(HttpStatus.OK).body("OK");
        }
    }
    return respuesta;
}
```

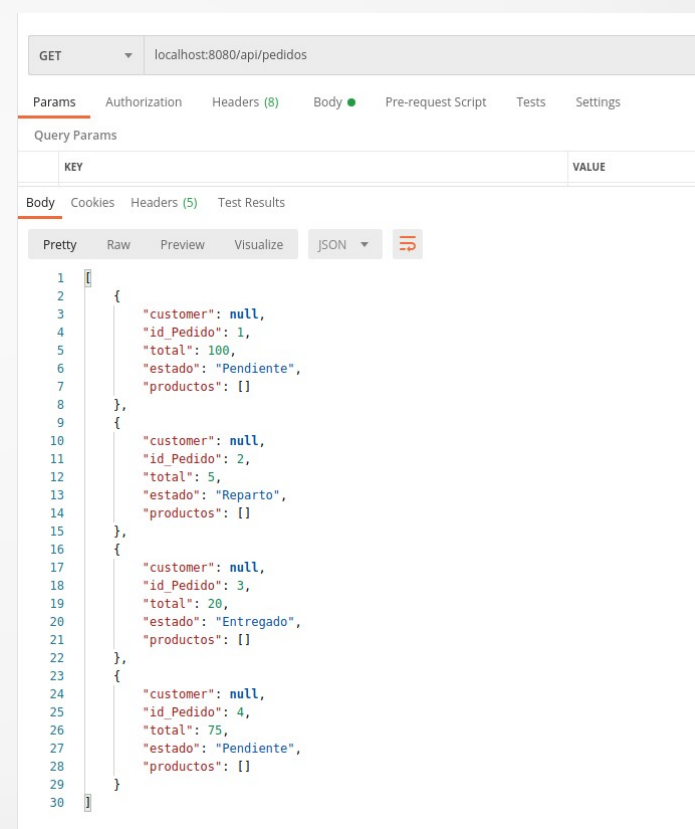
Peticiones

- A continuación pondré algunas capturas de peticiones realizadas desde Postman.



Postman interface showing a GET request to `localhost:8080/api/productos`. The response is a JSON array of product objects, displayed in the Body tab using the Pretty view.

```
[
  {
    "nombre": "Patatas",
    "precio": 10,
    "stock": 100
  },
  {
    "nombre": "Garbanzos",
    "precio": 5,
    "stock": 75
  },
  {
    "nombre": "Coca-Cola",
    "precio": 7,
    "stock": 90
  },
  {
    "nombre": "Chocolate",
    "precio": 12,
    "stock": 120
  }
]
```



Postman interface showing a GET request to `localhost:8080/api/pedidos`. The response is a JSON array of order objects, displayed in the Body tab using the Pretty view.

```
[
  {
    "customer": null,
    "id_Pedido": 1,
    "total": 100,
    "estado": "Pendiente",
    "productos": []
  },
  {
    "customer": null,
    "id_Pedido": 2,
    "total": 5,
    "estado": "Reparto",
    "productos": []
  },
  {
    "customer": null,
    "id_Pedido": 3,
    "total": 20,
    "estado": "Entregado",
    "productos": []
  },
  {
    "customer": null,
    "id_Pedido": 4,
    "total": 75,
    "estado": "Pendiente",
    "productos": []
  }
]
```

Peticiones

POST localhost:8080/api/pedidos/1

Params Authorization Headers (8) **Body** Pre-request Script Tests Settings

● none ● form-data ● x-www-form-urlencoded ● raw ● binary ● GraphQL JSON ▾

```
1 {
2   "nombre": "Chocolate",
3   "precio": 12,
4   "stock": 120
5 }
```

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize Text ▾

1 OK

POST localhost:8080/api/customers/1

Params Authorization Headers (8) **Body** Pre-request Script Tests Settings

● none ● form-data ● x-www-form-urlencoded ● raw ● binary ● GraphQL JSON ▾

```
3 {
4   "id_Pedido": 1,
5   "total": 100,
6   "estado": "Pendiente",
7   "productos": [
8     {
9       "nombre": "Chocolate",
10      "precio": 12,
11      "stock": 120
12     }
13   ]
14 }
```

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize Text ▾

1 OK

GET localhost:8080/api/customers

Params Authorization Headers (8) **Body** Pre-request Script Tests Settings

● none ● form-data ● x-www-form-urlencoded ● raw ● binary ● GraphQL JSON ▾

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize JSON ▾

```
1 [
2   {
3     "name": "Raúl",
4     "surname": "Morales",
5     "birthdate": null,
6     "address": null,
7     "city": "Sevilla",
8     "dni": "7816162",
9     "country": null,
10    "mobilenumber": null,
11    "gender": null,
12    "id": 1,
13    "pedidos": [
14      {
15        "customer": "Raúl",
16        "id_Pedido": 1,
17        "total": 100,
18        "estado": "Pendiente",
19        "productos": [
20          {
21            "nombre": "Chocolate",
22            "precio": 12,
23            "stock": 120
24          }
25        ]
26      }
27    ]
28  },
29 ]
```

Dificultades

- La mayor dificultad que he encontrado ha sido relacionar las entidades mediante las peticiones porque es algo a lo que hasta hoy no me había enfrentado.
- Solución realizar POST específico indicando datos en la url de la petición

Propuestas de mejora

- Aún tengo varios puntos que mejorar entre ellos:
 - Añadir servicios ya que actualmente los métodos del crud realizan todas las acciones.
 - Intentar mejorar los métodos post para hacer las modificaciones usando unicamente ids en el path sin tener que usar el body en las peticiones
 - Optimizar las búsquedas, etc...