

INFORME SPRINT 3

INTEGRANTES:

Johan Sebastián Robles Rincón

Rominger Buriticá Angulo

Andrés Mateo Franco Reyes

Diego Federico Vásquez Romero

Tutor Scrum: Daniel

7 de octubre de 2022

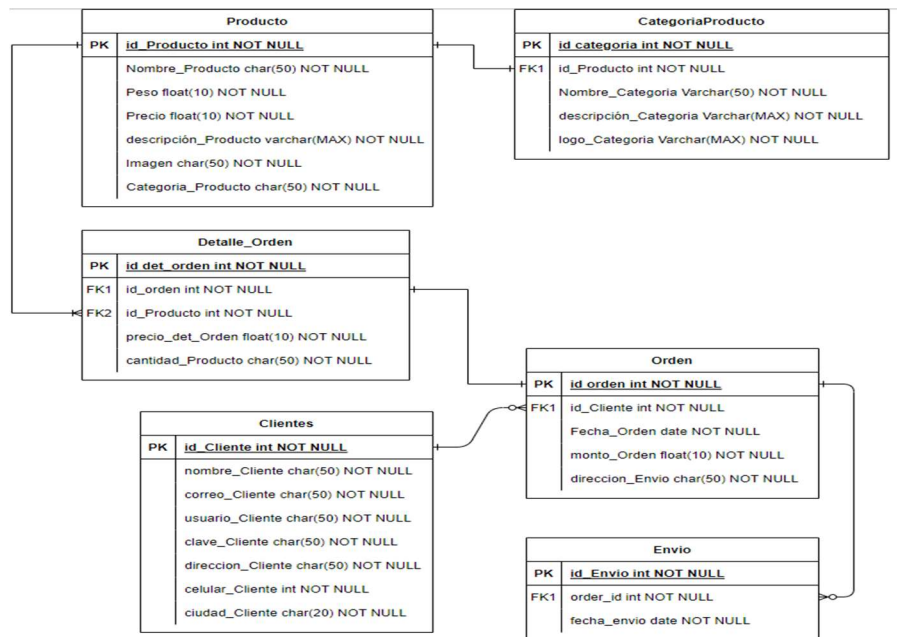
i. Presentación MVC

1. Aplicación con persistencia Relacional

La persistencia relacional se prueba por medio del uso de Java Persistence API, más conocida por sus siglas JPA, que es la API de persistencia desarrollada para la plataforma Java esta API integrada a la herramienta Hibernate nos muestra el Mapeo objeto-relacional (ORM), para la plataforma Java, ayudándonos a que en ejecuciones posteriores se puedan utilizar los datos previamente generados esto lo vemos en las propiedades de la aplicación donde como se observa en la línea 27 de la configuración esta se cambió de Create que es la propiedad que al lanzar cada vez el servidor creara nuevamente todas las entidades de la tabla a una funcionalidad más estable de Update que es la usualmente utilizada para entornos de producción.

```
17
18 spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
19 # spring.datasource.url=jdbc:mysql://<HOST>:<PORT>/<DATABASE>
20 # spring.datasource.url=jdbc:mysql://remotemysql.com:3306/gD0StdLxLC
21 spring.datasource.url=jdbc:mysql://sql5.freesqldatabase.com:3306/sql5524009
22 # spring.datasource.username=gD0StdLxLC
23 spring.datasource.username=sql5524009
24 # spring.datasource.password=Z0G1D4EMyz
25 spring.datasource.password=3LK2E16ImH
26
27 spring.jpa.hibernate.ddl-auto=update
```


En uno de los Diagrama entidad relación se plantearon 6 tablas que finalmente para la ejecución real proyecto se optó por emplear únicamente dos tablas manteniendo en ellas una integridad referencial, con campos identificadores auto numéricos creados por medio de la librería lombook.



2. Pruebas unitarias de la lógica desarrollada

Para el desarrollo de las pruebas unitarias se realizaron validaciones:

Integridad y persistencia relacional con las siguientes tablas:

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado
1	id_producto 	int(11)			No	Ninguna
2	link_imagen	varchar(255)	latin1_swedish_ci		Sí	NULL
3	nombre	varchar(50)	latin1_swedish_ci		No	Ninguna
4	precio_kilo	double			Sí	NULL
5	tipo	varchar(255)	latin1_swedish_ci		No	Ninguna

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado
1	username 	varchar(255)	latin1_swedish_ci		No	Ninguna
2	active	bit(1)			Sí	NULL
3	admin	bit(1)			Sí	NULL
4	contraseña	varchar(255)	latin1_swedish_ci		Sí	NULL
5	correo	varchar(255)	latin1_swedish_ci		Sí	NULL
6	nombre	varchar(255)	latin1_swedish_ci		Sí	NULL

En cada una de las interfaces del proyecto por lo que se probó como la aplicación ejecutaba todos los casos posibles al ingresar un determinado usuario:

Usuario

admin

Contraseña

.....

Usuario Inactivo 

Bienvenido

Usuario

Contraseña

Usuario No Existente ✕

[¿Eres Nuevo? Regístrate aquí](#)

Iniciar Sesión

Bienvenido

Usuario

Contraseña

Contraseña Incorrecta ✕

[¿Eres Nuevo? Regístrate aquí](#)

Iniciar Sesión

En el carrito de compras se validó la funcionalidad de este:

CARRITO

Producto	Precio	Cantidad	
Cebollas \$4500	4500	<input type="text" value="1"/>	<input type="button" value="X"/>
Tomates \$4000	4000	<input type="text" value="1"/>	<input type="button" value="X"/>
Total \$ 8500			<input type="button" value="Comprar"/>

Para asegurarse de la persistencia en las tablas mencionadas se ejecuto el siguiente código con ayuda de la librería lombok y como se observa en las imágenes se asegura persistencia con los id's generados para cada una de las tablas:

```
package com.loscampesinos.loscampesinos.model.entity;

import javax.persistence.Column;
import javax.persistence.Entity; //JPA
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;
import javax.print.DocFlavor.STRING;

import lombok.Data;

@Entity // para indicarle que es una tabla en la base de datos
@Data // me genera por defecto los setter y getter
@Table(name = "Productos") //colocarle el nombre a la tabla
public class producto {
    @Id //para indicar a llaver primaria de la tabla
    @GeneratedValue(strategy = GenerationType.AUTO) // PARA INDICARLE QUE EL AVLOR SE GENERE POR SI SOLO DE MANERA INCREMENTAL
    private Integer id_producto;

    @Column(name = "Nombre", nullable = false, length = 50) // para indicarle características al campo nombre de la tabla
    private String nombre;
    private Double precioKilo;
    private String linkImagen;
    @Column(name = "tipo", nullable = false)
    private String tipo;
    // en id tipo es 1 para verduras, 2 para frutas y 3 para granos.
}
```

```
1 package com.loscampesinos.loscampesinos.model.entity;
2
3 import javax.persistence.Entity;
4 import javax.persistence.Id;
5
6 import lombok.Data;
7
8 @Data
9 @Entity
10 public class usuario {
11     private String nombre;
12     @Id
13     private String username;
14     private String correo;
15     private String contraseña;
16     private Boolean active;
17     private Boolean admin;
18 }
```

Posteriormente se realizaron pruebas al insertar productos en la base de datos:

```
INSERT INTO `productos` (`id_producto`, `link_imagen`, `nombre`,  
`precio_kilo`, `tipo`) VALUES ('18',  
'https://th.bing.com/th/id/R.31951a51b9c70ea455968001bbd6df2b?  
rik=eGZ4Zcc7XZwQYQ&pid=ImgRaw&r=0', 'Peras', '15000', '2')
```

Al tratar de insertar un registro con un id repetido dentro de la base se obtiene la siguiente respuesta por parte del servidor:

```
#1062 - Entrada duplicada '18' para la clave 'PRIMARY'
```

Al insertar un nuevo producto con el id correcto

```
INSERT INTO `productos` (`id_producto`, `link_imagen`, `nombre`, `precio_kilo`, `tipo`) VALUES ('23', 'https://th.bing.com/th/id/R.31951a51b9c70ea455968001bbd6df2b?  
rik=eGZ4Zcc7XZwQYQ&pid=ImgRaw&r=0', 'Peras', '15000', '2')
```

Esta ejecutando perfectamente el script:

✓ 1 fila insertada. (La consulta tardó 0.0721 segundos.)

ii. Informe de retrospectiva

1.- Entregables

- Backups de las bases creadas
- Diseño Front del carrito de compras (Documentos Html y CSS)
- Implementación Backend del registro de usuarios y el carrito de compras (Documentos .java)

2.- Plan de acciones de mejora.

- Recapitular los objetivos del sprint y del equipo para acoplarse mejor a los tiempos de entrega.
- Realizar votaciones más breves de las funcionalidades que podrán ser implementadas dentro del proyecto.

3.- Nuevas best practices.

- Eliminar las ramas del proyecto que ya no estan en entorno de desarrollo
- Disminuir el número de aprobaciones necesarias para aprobar los cambios dentro del repositorio ya que estan abarcando mucho tiempo en revisión.
- Generar un autoincremento de los id's automatico

4.- Acuerdos de equipo actualizados.

- Proponer pruebas para implementar en las funcionalidades implementadas con el fin de ver el comportamiento y comprobar el uso de estas.

5.- Impedimentos a escalar

- Las bases de datos en la nube, solicitan una suscripción luego de pasar una semana de uso para esto evaluar si lo más conveniente es utilizar uno de estos servicios o emplear una base de datos on-premise

iii. Historias de usuario a desarrollar en el sprint 4. (Trello)

<https://trello.com/invite/b/66pkCJ0Q/4623c1c823b502b7cb2fc82f9861a2b5/sprint4>