

## Proyecto Tienda Carrito de Compras

### Descripción

Se tiene una tienda con N cantidad de productos, los cuales se podrán añadir, eliminar, actualizar en el carro de compras y en caso de que haya añadido toda la cantidad de un producto, al intentar añadir de nuevo ese producto deberá indicarle al usuario que no hay disponibilidad de ese producto.

Cada usuario podrá hacer un registro básico (nombre/email/password) para poder realizar la compra de dichos productos.

### Requerimiento

Se requiere la implementación de un API que permita:

#### Mínimos

- ✓ Listar los productos
- ✓ Añadir un producto al carro
- ✓ Actualizar un producto existente en el carro
- ✓ Eliminar un producto del carro
- ✓ Validar el stock del producto, en caso que llegue a 0 deberá informar
- ✓ El código debe estar debidamente comentado

Esta prueba puede ser desarrollada en el lenguaje de programación que usted elija y sienta que tiene mayor conocimiento. No hay restricciones en el framework o tecnología de almacenamiento de datos. Se espera documentación de cómo fue abordada la solución, con información del diseño y/o patrones implementados y con las instrucciones necesarias para montar el proyecto y probar las funcionalidades requeridas, por ejemplo, migraciones de base de datos, esquemas de base de datos, colecciones en Postman, etc...

Se tendrá en alta estima el uso de buenas prácticas de desarrollo según el lenguaje y la escritura de pruebas automatizadas determinísticas

#### Nice to have (suma puntos en orden de prioridad)

- ✓ Pruebas unitarias
- ✓ Autenticación de usuario

#### Entrega:

- ✓ Dejar en Repositorio público.
- ✓ Código fuente de la implementación
- ✓ README correspondiente con la información necesaria del proyecto
- ✓ Debe incluir como ejecutar el proyecto al igual que las pruebas
- ✓ Commits acordes al proceso de implementación

### Herramientas utilizadas:

- ✓ **Java JDK** Versión 11.0.13
- ✓ **IDE:** IntelliJ IDEA Community Edition 2022.1.1
- ✓ **Base de Datos:** PostgreSQL Versión 11.16
- ✓ **Pruebas Unitarias:** Postman
- ✓ **Framework Spring:** Spring Framework, Spring Boot, Spring Data JPA
- ✓ **Front-end:** Angular Versión 13.3.2

### LINK de Descargas Herramientas:

Instalación de ambiente de desarrollo: Windows

#### ✓ **JDK Java**

<https://www.oracle.com/co/java/technologies/javase/jdk11-archive-downloads.html>

#### ✓ **IDE IntelliJ IDEA**

<https://www.jetbrains.com/idea/download/#section=windows>

#### ✓ **PostgresSQL**

<https://www.enterprisedb.com/downloads/postgres-postgresql-downloads>

#### ✓ **Postman**

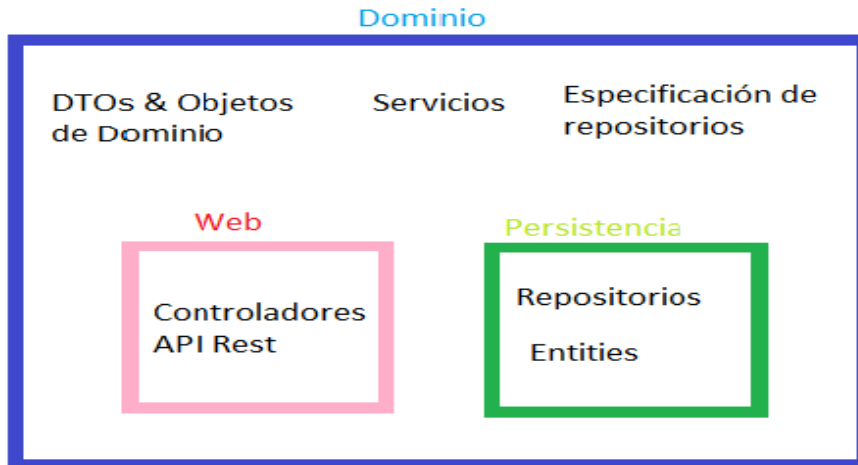
<https://www.postman.com/downloads/>

#### ✓ **Framework Spring**

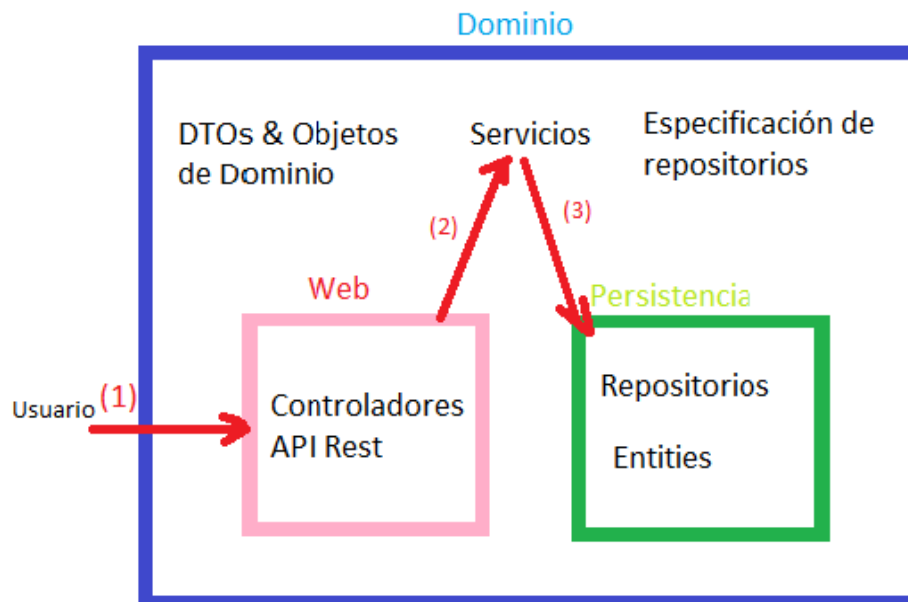
<https://start.spring.io/>

## Arquitectura por capas orientada al dominio

La arquitectura para darle solución al requerimiento está conformada por DTO y objetos de dominio, Servicios, especificación de Repositorios, una capa Web con los controladores API Rest y una capa de Persistencia (Repositorios Entities).



## Flujo normal de la App

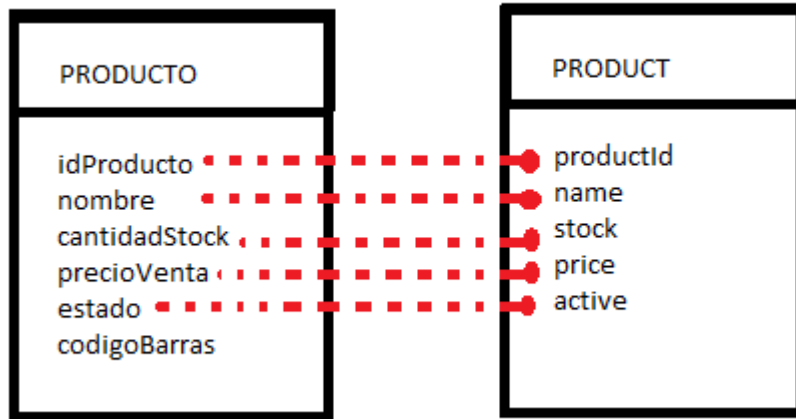


- (1) Un Usuario hace el llamado a un controlador de nuestro API.
- (2) El controlador de nuestra API va al servicio que contiene todo lo necesario para intervenir en esa operación.
- (3) El servicio va al repositorio específico que necesite y es allí donde si tenemos que hacer algún movimiento o gestión en la base de datos, es donde ocurre.

## Patrón Data Mapper

Consiste en convertir o traducir dos objetos que pueden hacer una misma labor. Entre las características más importantes están:

- ✓ No exponer la base datos en la API.
- ✓ Evitar campos innecesarios en la API.
- ✓ Evitar mezclar idiomas en la aplicación.

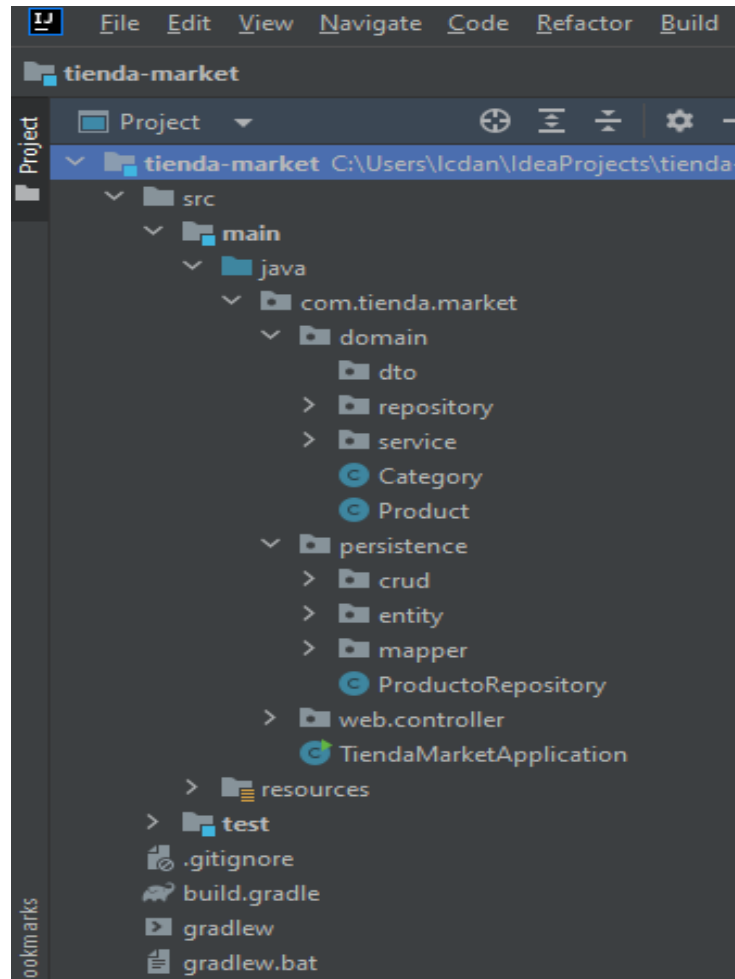


## Generación Proyecto con Spring initializr

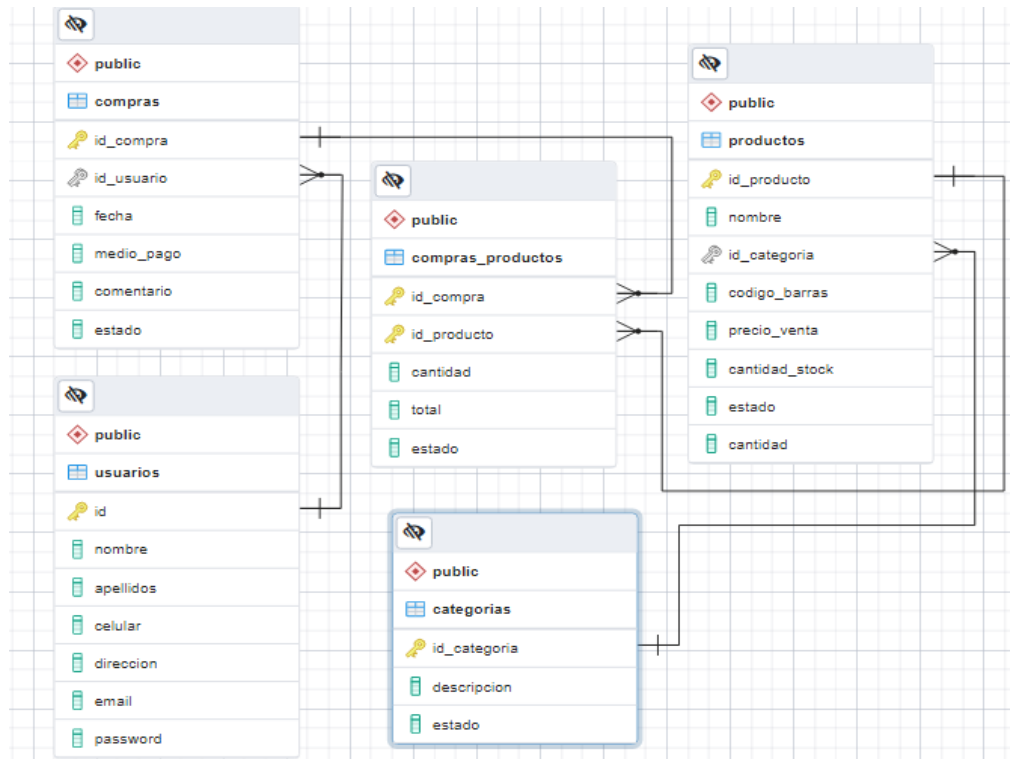
The screenshot displays the Spring Initializr web application interface. At the top, the 'spring initializr' logo is visible. Below the logo, the 'Project' section has radio buttons for 'Maven Project' and 'Gradle Project', with 'Gradle Project' selected. The 'Language' section has radio buttons for 'Java', 'Kotlin', and 'Groovy', with 'Java' selected. The 'Spring Boot' section has radio buttons for '3.0.0 (SNAPSHOT)', '3.0.0 (M4)', '2.7.3 (SNAPSHOT)', '2.7.2', '2.6.11 (SNAPSHOT)', and '2.6.10', with '2.7.2' selected. The 'Project Metadata' section contains input fields for 'Group' (com.tienda-market), 'Artifact' (tienda-market), 'Name' (tienda-market), 'Description' (Proyecto), and 'Package name' (com.tienda-market). The 'Packaging' section has radio buttons for 'Jar' and 'War', with 'Jar' selected. The 'Java' section has radio buttons for '18', '17', '11', and '8', with '11' selected. At the bottom right, there is a 'GENERATE' button with a keyboard shortcut 'CTRL + ⌘'.

## Estructura del proyecto

Inicialmente se creo el proyecto con Spring initializr, se descarga y se configura en el IDE IntelliJ IDEA Community Edition



## Modelo E-R

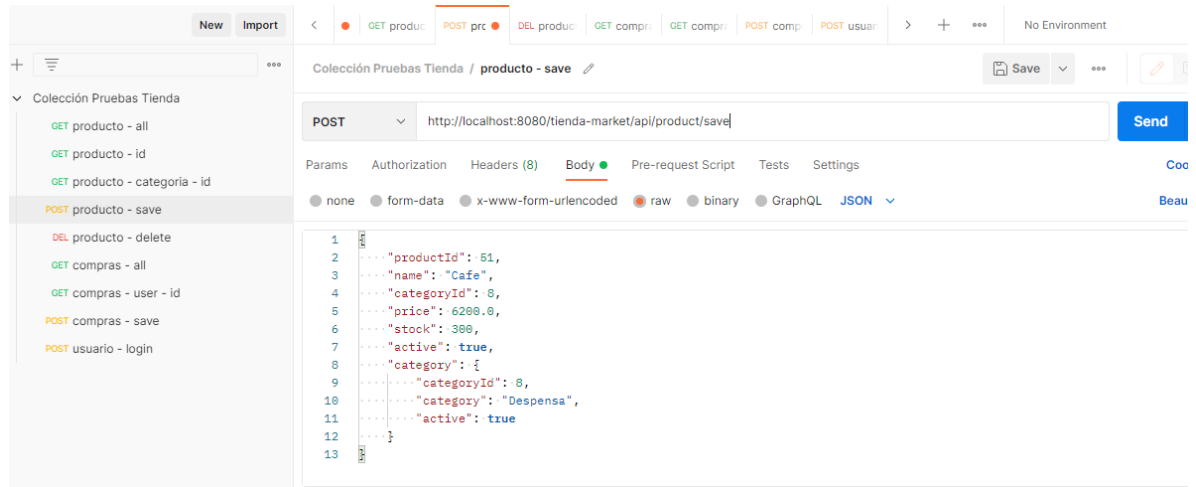


Creada la base datos en PosgreSQL, se ejecutan los scripts contenidos en los archivos, en el siguiente orden:

Crear tablas: 1\_DDL\_Tienda.sql

Crear registros en tablas: 2\_DatosBD\_Tienda.sql

## Pruebas del API tienda-market/api realizadas en Posman



## Registro Usuario

Iniciar Sesión

Registro

Cédula

412356

Nombres

Pepito

Apellidos

Perez

Correo Electrónico

pepe@gmail.com

Contraseña

...

Guardar

## Interfaz Login

Iniciar Sesión

Registro




Correo Electrónico  
pepe@gmail.com

Contraseña

\*\*\*

Ingresar

## Dashboard Tienda

pepe@gmail.com	Pedidos	Categorías	Tienda	Pepito
<b>Aguacate</b> \$2,500.00 98	<b>Apio</b> \$150.00 114	<b>Guayaba Feijoa</b> \$300.00 499		
				
<a href="#">Comprar</a> <a href="#">Ver</a>	<a href="#">Comprar</a> <a href="#">Ver</a>	<a href="#">Comprar</a> <a href="#">Ver</a>		
Lechuga	Limón	Mango		



