Integrantes:

Jaime Álvarez

Lucas Fuentes

Javiera Gálvez

ecomarket spa

Experiencia n°2

**Índice**

[**1.** **Introducción** 2](#_Toc199365707)

[**2.** **Diagrama de arquitectura de microservicios** 3](#_Toc199365708)

[**3.** **Estructura del proyecto** 4](#_Toc199365709)

[3.1. Dependencias: 4](#_Toc199365710)

[3.2 Componentes implementados: 4](#_Toc199365711)

[**4.** **Base de datos** 6](#_Toc199365712)

[4.1 Motor utilizado: 6](#_Toc199365713)

[4.2 Estructura de bases de datos utilizadas: 6](#_Toc199365714)

[**5.** **Implementación de los servicios** 8](#_Toc199365715)

[5.1 Servicios REST definidos: 8](#_Toc199365716)

[5.2 Interacción con la Base de Datos (MySQL) 11](#_Toc199365717)

[**6.** **Implementación de vistas** 13](#_Toc199365718)

[6.1 Vista de Usuarios: 13](#_Toc199365719)

[6.2 Vista de Productos: 13](#_Toc199365720)

[6.3 Vista de Carrito: 13](#_Toc199365721)

[**7.** **Uso de Git y GitHub** 14](#_Toc199365722)

[7.1 Control de versiones: 14](#_Toc199365723)

[7.2 Repositorio en GitHub: 15](#_Toc199365724)

[**8.** **Conclusión** 16](#_Toc199365725)

# **Introducción**

EcoMarket SPA es una empresa chilena dedicada a la venta de productos ecológicos y sostenibles, con presencia en distintas regiones del país. A raíz de su crecimiento acelerado y la expansión a nuevas ciudades, la empresa ha comenzado a experimentar serias limitaciones con su sistema monolítico actual, afectando tanto el rendimiento operativo como la experiencia del cliente.

Con el fin de responder a las nuevas demandas de escalabilidad, disponibilidad y eficiencia, se propone el rediseño del sistema utilizando una **arquitectura de microservicios** soportada por el framework **Spring** y un motor de base de datos **MySQL**. Este enfoque permitirá a EcoMarket SPA segmentar sus funcionalidades críticas —como la gestión de usuarios, inventario, ventas, logística y atención al cliente— en servicios independientes pero integrados, facilitando el mantenimiento, el despliegue continuo y la adaptabilidad frente al crecimiento del negocio.

El presente informe documenta el desarrollo técnico de esta solución, abordando desde la configuración inicial del proyecto con herramientas como **Maven** y **Git**, hasta la implementación de servicios **RESTful** y su validación mediante pruebas realizadas con **Postman**. Además, se describe la estructura del sistema, la interacción entre los distintos módulos y las vistas diseñadas para visualizar operaciones del sistema, tanto internas como externas.

Esta solución no solo busca resolver los problemas actuales de EcoMarket SPA, sino también sentar las bases para un sistema moderno, modular y escalable, capaz de sostener su crecimiento futuro y mejorar la experiencia de sus usuarios a todo nivel.

# **Diagrama de arquitectura de microservicios**

El siguiente diagrama representa la arquitectura propuesta del sistema de EcoMarket SPA, basada en una estructura de microservicios independientes que se comunican mediante API REST. Esta arquitectura busca reemplazar el sistema monolítico existente, permitiendo una mayor escalabilidad, mantenibilidad y disponibilidad.

Cada microservicio se encarga de una funcionalidad específica del sistema —como gestión de cuentas, productos, inventario, carrito, compras y envíos—, y está asociado a su propia base de datos para mantener el principio de independencia y desacoplamiento. La interacción del usuario se realiza a través de una interfaz web o móvil, conectada al sistema por una capa de API que distribuye las solicitudes a los servicios correspondientes.

Esta arquitectura permite a EcoMarket SPA responder eficientemente al crecimiento de su operación, mejorar la experiencia del cliente y optimizar la gestión interna mediante una estructura modular y escalable.

Diagrama

El contenido generado por IA puede ser incorrecto.

# **Estructura del proyecto**

El proyecto EcomarketSPA es una aplicación desarrollada en Java 17 utilizando el framework Spring Boot 3.5.0, estructurada bajo el patrón de arquitectura en capas (modelo MVC extendido), que promueve una clara separación de responsabilidades. Esta estructura facilita el mantenimiento, escalabilidad y reutilización del código, permitiendo manejar tanto servicios REST como vistas basadas en HTML.

El código fuente se encuentra organizado en distintos paquetes que representan entidades, controladores, servicios, repositorios y la clase principal de arranque de la aplicación.

## Dependencias:

El archivo pom.xml utiliza Maven como sistema de construcción y define las siguientes dependencias esenciales:

| **Dependencia** | **Funcionalidad** |
| --- | --- |
| **spring-boot-starter-web** | Desarrollo de aplicaciones web y APIs REST. |
| **spring-boot-starter-data-jpa** | Integración con JPA/Hibernate para persistencia en base de datos. |
| **spring-boot-starter-thymeleaf** | Motor de plantillas para vistas HTML dinámicas. |
| **mysql-connector-j** | Driver para conexión con bases de datos MySQL. |
| **spring-boot-devtools** | Soporte para recarga automática durante el desarrollo. |
| **spring-boot-starter-test** | Herramientas para pruebas unitarias y de integración. |

El proyecto utiliza Java 17 y el plugin spring-boot-maven-plugin para compilar y ejecutar la aplicación.

## 3.2 Componentes implementados:

A continuación, se describen los principales componentes del sistema y su función:

**EcomarketspaApplication.java**

* Clase principal del proyecto. Inicia la aplicación mediante la anotación @SpringBootApplication.

**Entities/**

* Contiene las clases que representan las entidades del sistema:
  + Usuario.java
  + Producto.java
  + Carrito.java
* Estas clases están anotadas con @Entity y se utilizan para mapear objetos Java a tablas en la base de datos.

**Repository/**

* Define interfaces que extienden JpaRepository, facilitando operaciones CRUD sobre las entidades:
  + UsuarioRepository
  + ProductoRepository
  + CarritoRepository

**Services/ y ServicesImpl/**

* Capas de lógica de negocio:
  + Las interfaces (UsuarioService, etc.) definen las operaciones disponibles.
  + Las clases Servicelmpl implementan esa lógica y coordinan el flujo de datos.

**RestController/**

* Exponen los servicios como **API RESTful**:
  + UsuarioRestController
  + ProductoRestController
  + CarritoRestController
* Manejan peticiones HTTP (GET, POST, PUT, DELETE) y se comunican con la capa de servicios.

**controllers/**

* Controladores tipo MVC, pensados para renderizar vistas con HTML dinámico, probablemente usando Thymeleaf.

# **Base de datos**

La base de datos del sistema EcomarketSPA fue diseñada para representar los elementos clave de una tienda en línea: usuarios, productos y carritos de compra. Esta base de datos es utilizada por la aplicación Spring Boot para almacenar y gestionar la información relacionada con las operaciones del sistema, como la creación de usuarios, administración de productos y gestión de carritos.

La relación entre las tablas fue diseñada utilizando el enfoque relacional, con claves primarias y foráneas que permiten mantener la integridad referencial. El mapeo objeto-relacional se lleva a cabo mediante las entidades Java anotadas con @Entity, en conjunto con el uso de Spring Data JPA para interactuar con la base de datos.

## 4.1 Motor utilizado:

El sistema utiliza como motor de base de datos:

* **MySQL**
  + Especificado en el proyecto mediante el **driver JDBC** mysql-connector-j.
  + Su integración se realiza a través de **Spring Data JPA**, lo que permite que las entidades se persistan automáticamente sin necesidad de sentencias SQL explícitas.
  + El archivo application.properties definiría la configuración de conexión (url, username, password, y dialecto de Hibernate).

## 4.2 Estructura de bases de datos utilizadas:

A continuación, se describen las principales tablas del sistema y sus relaciones, basadas en las clases de entidad:

**Tabla usuarios**

* **idUsuario** (PK): Identificador único del usuario.
* **nombUsuario**: Nombre del usuario.
* **correo**: Correo electrónico del usuario.

**Tabla productos**

* **id** (PK): Identificador único del producto.
* **nombre**: Nombre del producto.
* **descripcion**: Descripción detallada.
* **precio**: Precio del producto (en valor entero).

**Tabla carrito**

* **idCarrito** (PK): Identificador único del carrito.
* **usuario** (FK): Relación con la tabla usuarios.
* **producto** (FK): Relación con la tabla productos.
* **cantidad**: Cantidad de unidades de ese producto en el carrito.

Estas tres tablas se encuentran relacionadas a través de claves foráneas, permitiendo que un carrito esté asociado a un usuario y a uno o más productos.

En términos de modelado, carrito actúa como una tabla intermedia que representa una relación **muchos a muchos** entre usuarios y productos, incluyendo un atributo adicional (cantidad) que especifica cuántas unidades de un producto ha añadido el usuario.

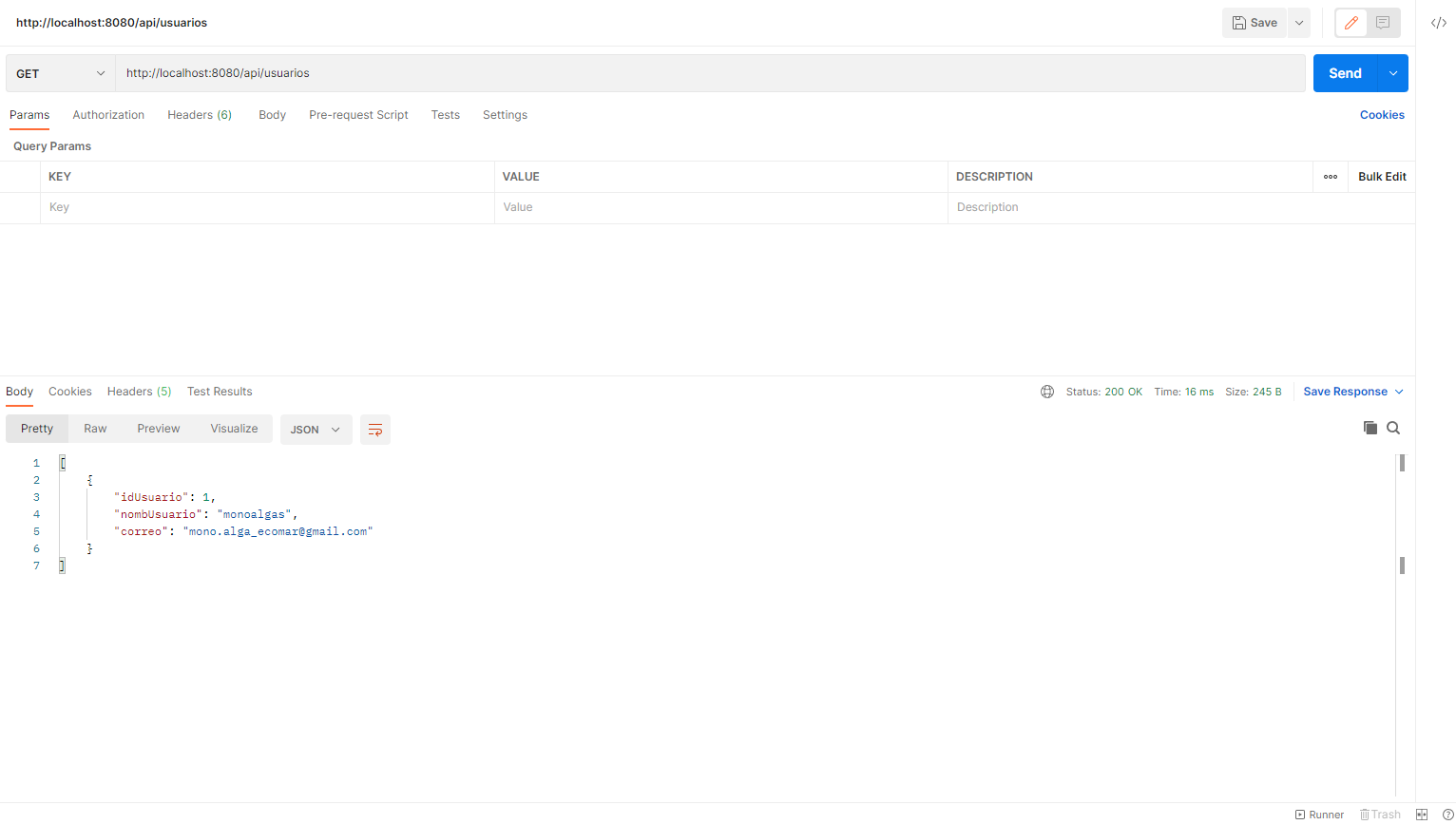
# **Implementación de los servicios**

El sistema expone una API Full REST para gestionar usuarios, productos y carritos de compra, utilizando controladores definidos con la anotación @RestController en Spring Boot. Cada recurso cuenta con rutas que permiten realizar operaciones de tipo GET, POST, PUT y DELETE, cumpliendo con los principios de una arquitectura RESTful.

Estas peticiones permiten la interacción directa con la base de datos base\_ecomarketspa, gestionada mediante JPA y el motor MySQL.

## 5.1 Servicios REST definidos:

| **Método** | **Endpoint** | **Acción** |
| --- | --- | --- |
| GET | /api/usuarios | Obtener todos los usuarios registrados. |
| GET | /api/usuarios/{id} | Obtener un usuario por su ID. |
| POST | /api/usuarios | Crear un nuevo usuario. |
| PUT | /api/usuarios/{id} | Actualizar un usuario existente. |
| DELETE | /api/usuarios/{id} | Eliminar un usuario. |

Evidencia (GET /api/usuarios):

| **Método** | **Endpoint** | **Acción** |
| --- | --- | --- |
| GET | /api/productos | Listar todos los productos. |
| GET | /api/productos/{id} | Ver un producto específico. |
| POST | /api/productos | Agregar un nuevo producto. |
| PUT | /api/productos/{id} | Editar un producto. |
| DELETE | /api/productos/{id} | Eliminar un producto. |

Evidencia (GET /api/productos):

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

El contenido generado por IA puede ser incorrecto.

| **Método** | **Endpoint** | **Acción** |
| --- | --- | --- |
| GET | /api/carritos | Ver todos los carritos. |
| GET | /api/carritos/{id} | Ver detalles de un carrito específico. |
| POST | /api/carritos | Crear un nuevo carrito. |
| DELETE | /api/carritos/{id} | Eliminar un producto del carrito. |

Evidencia (GET /api/carritos):

Interfaz de usuario gráfica, Aplicación, Teams

El contenido generado por IA puede ser incorrecto.

## 5.2 Interacción con la Base de Datos (MySQL)

La base de datos utilizada es base\_ecomarketspa, con las siguientes tablas principales:

* usuario
* producto
* carrito

Estructura de tablas en MySQL Workbench:

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

Consultas SQL en ejecución:

Texto, Aplicación, Correo electrónico

El contenido generado por IA puede ser incorrecto.

# **Implementación de vistas**

La aplicación EcomarketSPA permite acceder y visualizar los datos almacenados en la base de datos a través de servicios GET disponibles en la API REST. Estos servicios entregan la información en formato estructurado, lo que permite que sea interpretada y mostrada de forma clara para el usuario final en cualquier sistema que consuma la API, como aplicaciones móviles, de escritorio o sistemas web.

A continuación, se detallan las vistas simuladas generadas a partir de los datos presentes en el sistema.

## 6.1 Vista de Usuarios:

**Servicio utilizado**: GET /api/usuarios

Este servicio entrega un listado de los usuarios registrados en el sistema. La vista resultante puede mostrarse de forma tabular, incluyendo identificador, nombre y correo electrónico.

## 6.2 Vista de Productos:

**Servicio utilizado**: GET /api/productos

Permite desplegar los productos disponibles en el sistema, incluyendo su nombre, descripción y precio. Esta vista puede ayudar a los usuarios a revisar el catálogo de productos ecológicos.

## 6.3 Vista de Carrito:

**Servicio utilizado**: GET /api/carrito

Esta vista permite mostrar el contenido de los carritos de compra, vinculando usuarios y productos junto con la cantidad seleccionada por cada uno.

Estas vistas permiten visualizar de forma estructurada y comprensible la información contenida en el sistema. Están directamente basadas en los servicios disponibles, y cumplen el objetivo de entregar al usuario final un acceso claro a los datos que forman parte de las operaciones centrales del sistema: usuarios registrados, productos disponibles y carritos de compra.

# **Uso de Git y GitHub**

## 7.1 Control de versiones:

Para el control de versiones del proyecto se utilizó **Git**, un sistema distribuido que permite llevar un historial de cambios, facilitando la colaboración, el trabajo modular y la reversión en caso de errores.

A lo largo del desarrollo se emplearon los siguientes comandos clave:

| **Comando** | **Descripción** |
| --- | --- |
| git init | Inicializa el repositorio local en el proyecto. |
| git add . | Añade todos los archivos al área de preparación. |
| git commit -m "mensaje" | Guarda un cambio con un mensaje descriptivo. |
| git branch -M main | Renombra la rama principal como main. |
| git remote add origin https://github.com/jaimeDalvarez/EVALUACION-EcomarketSPA.git | Asocia el repositorio local al remoto en GitHub. |
| git push -u origin main | Sube el contenido inicial al repositorio remoto. |
| git push | Sube cambios posteriores. |

## 7.2 Repositorio en GitHub:

El proyecto está disponible públicamente en GitHub en el siguiente enlace:

<https://github.com/jaimeDalvarez/EVALUACION-EcomarketSPA>

Este repositorio contiene:

* Todo el código fuente del proyecto Spring Boot.
* Archivos de configuración como pom.xml.
* Capturas de pantalla de ejecución en Postman y base de datos.
* Archivos de prueba en formato .json.

# **Conclusión**

El desarrollo del proyecto EcomarketSPA permitió consolidar conocimientos fundamentales en la creación de aplicaciones backend utilizando el framework Spring Boot. A través de una arquitectura bien definida basada en capas, se logró implementar un sistema robusto, modular y escalable que cubre la gestión de usuarios, productos y carritos de compra mediante una API RESTful.

El uso de herramientas como Spring Data JPA y MySQL facilitó la integración de la lógica de negocio con la persistencia de datos, mientras que el control de versiones mediante Git y GitHub aseguró un flujo de trabajo organizado, documentado y colaborativo.

Si bien el sistema actualmente opera sin una interfaz gráfica de usuario, está preparado para ser consumido por clientes externos (como aplicaciones móviles o frontend en React/Vue), lo cual lo convierte en una base sólida para futuras ampliaciones.

En resumen, EcomarketSPA representa una solución funcional, escalable y bien estructurada, que responde a las necesidades de una tienda ecológica y puede evolucionar fácilmente hacia una aplicación completa de comercio electrónico.