

Documentación del Proyecto Web con Spring Boot

1. Carátula

Campo	Valor
Nombre del Proyecto	Cyber Kiosco
Asignatura / Curso	Programación II
Profesor / Tutor	Carlos Cimino
Estudiante(s)	Santiago Margarit, Nicolas Speratti
Fecha de Presentación	4/11/2025
Tecnologías Principales	Java, Spring Boot, Thymeleaf, HTML, CSS, MySQL

2. Componentes de Spring Boot Utilizados

Explicación: Esta sección detalla los módulos, dependencias (Starters) o características específicas de Spring Boot que fueron esenciales para construir la aplicación, y justifica por qué se eligieron.

Dependencia / Característica	Propósito / Uso en el Proyecto
Spring Web (spring-boot-starter-web)	Permite el desarrollo de aplicaciones web RESTful y MVC, manejando las peticiones HTTP.
Spring Data JPA (spring-boot-starter-data-jpa)	Se usó para la persistencia de datos y la comunicación con la base de datos a través de Repositorios e interfaces ORM (Hibernate).
[Motor de Base de Datos] (mysql-connector-j, h2 para utilizar con los tests, es una base de datos en memoria.)	Proporciona el <i>driver</i> y configuración automática.
Thymeleaf (spring-boot-starter-thymeleaf)	Usado para renderizar las vistas (HTML).
Spring DevTools (spring-boot-devtools)	Para facilitar el desarrollo (recarga en caliente, etc.).
Spring Test (spring-boot-starter-test)	Para realizar los tests para verificar que todo funcione.
Lombok	Para reducir el código boilerplate.

3. Rutas (Endpoints) de la Aplicación Web

Explicación: Muestra el mapeo de todas las URLs principales que la aplicación expone y la acción que realiza cada una. Esto es crucial para entender la funcionalidad de la aplicación desde una perspectiva de usuario o cliente.

Método HTTP	Ruta (URL)	Descripción de la Acción	Controlador (Clase)
GET	/	Redirecciona al controller de productos	IndexController
GET	/productos	Lista todos los productos y también se encarga de la búsqueda en la página.	ProductoController
GET	/producto_detalle/{id}	Muestra un producto determinado en detalle..	ProductoController
GET	/contactos	lleva a la página de contactos	IndexController
POST	/carrito/agregar	Permite agregar productos al carrito, modificando los registros de carrito y carrito_producto	CarritoController
GET	/carrito	lleva a la página de carrito.	CarritoController
Post	/carrito/sacar	permite eliminar un producto del carrito_producto	CarritoController
Post	/carrito/comprar	realiza la compra	CarritoController

		final del carrito, verificando que haya stock antes de confirmarla.	
--	--	--	--

4. Arquitectura de Tres Capas (MVC y Persistencia)

La arquitectura del proyecto sigue el patrón de **Modelo-Vista-Controlador (MVC)**, extendido a un diseño de **Tres Capas Lógicas** (**Vista**, **Negocio/Servicio** y **Datos/Persistencia**) para garantizar la **separación de responsabilidades**, la modularidad y la facilidad de mantenimiento.

4.1. 🖼️ Capa de Vista (View)

Objetivo Principal	Implementación en Spring Boot (Tecnología)
Responsable de presentar la interfaz de usuario (UI) al cliente.	Utiliza Thymeleaf para el <i>templating</i> y el renderizado dinámico de HTML.

Componentes Clave:

- **Vistas Principales:** Archivos de plantillas HTML principales ubicados en `src/main/resources/templates/`.
 - `index.html`
 - `producto_detalle.html`
 - `carrito.html`
 - `contactos.html`
- **Reutilización de Código:** Uso de **Fragments** de Thymeleaf para reducir el código repetido (ej. *headers, footers*).
- **Páginas de Error Personalizadas:** Incluye vistas específicas para manejar errores y mejorar la experiencia del usuario.
 - `4xx.html` (errores del lado del cliente)
 - `5xx.html` (errores del lado del servidor)
 - `400.html` (Bad Request)
 - `404.html` (Not Found)

4.2. 🚦 Capa de Controlador (Controller)

Objetivo Principal	Implementación en Spring Boot (Anotación)

Manejar las peticiones HTTP entrantes, invocar la lógica de la Capa de Negocio , y seleccionar la vista a renderizar.	Clases anotadas con @Controller que definen los <i>mappings</i> de las rutas de la aplicación.
---	---

Controladores Implementados:

- **IndexController** (Manejo de rutas principales y redirecciones).
- **ProductoController** (Gestión de productos y navegación al detalle).
- **CarritoController** (Gestión de la lógica de la cesta de compra).

4.3. Capa de Negocio (Service)

Objetivo Principal	Implementación en Spring Boot (Anotación)
Contener la lógica de negocio central , las validaciones y gestionar las transacciones entre las capas de presentación y datos.	Clases anotadas con @Service para gestionar la inyección de dependencias.

Servicios Clave:

La lógica de negocio se distribuye en servicios especializados para cada entidad:

- **ProductoService**
- **CarritoService**
- **UsuarioService, PerfilService, RolService** (Gestión de seguridad y usuarios)
- **MarcaService, CategoriaService**
- **CarritoProductoService** (Gestión de la relación entre Carrito y Producto)

4.4. Capa de Acceso a Datos (Repository)

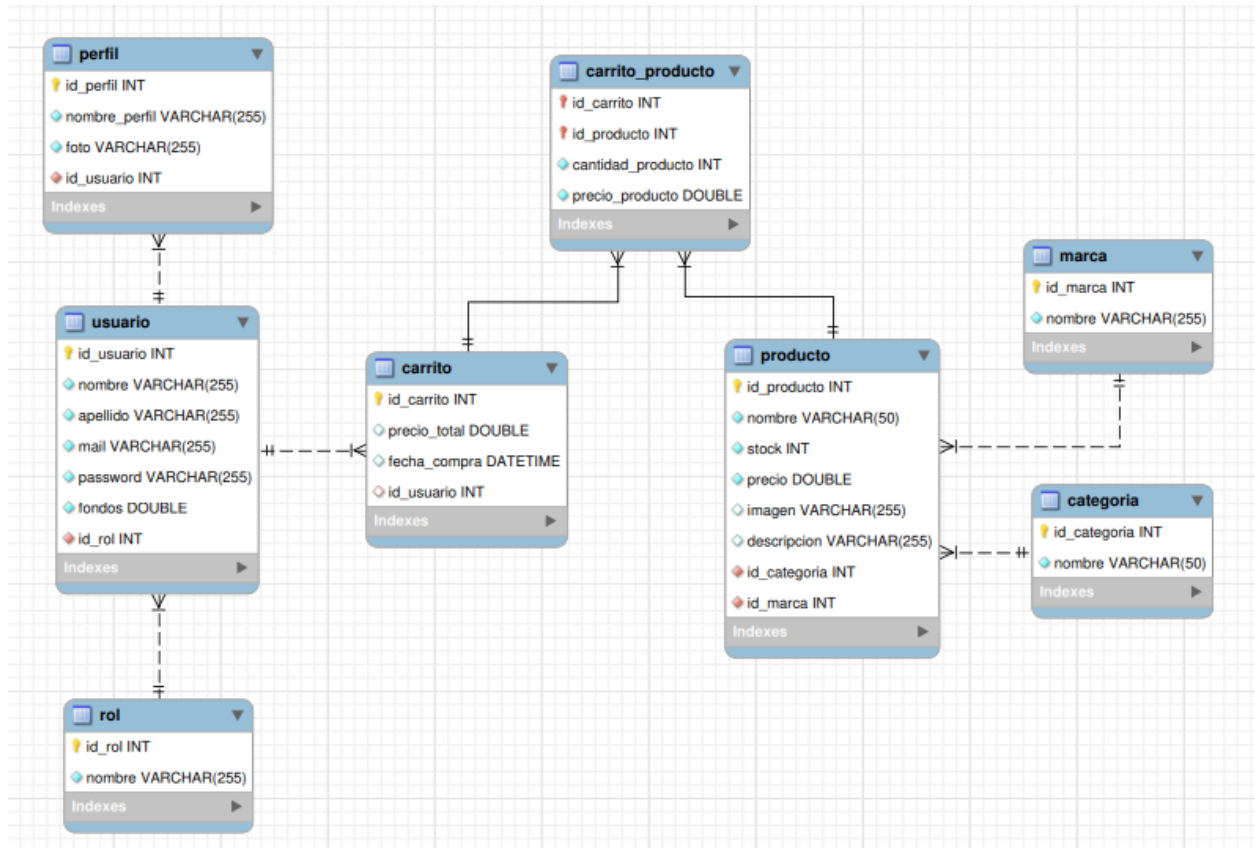
Objetivo Principal	Implementación en Spring Boot (Tecnología)
Gestionar la interacción directa con la base de datos (Persistencia) , realizando las	Interfaces que extienden de JpaRepository (usando Spring

operaciones CRUD.	Data JPA).
-------------------	-------------------

Repositorios Implementados:

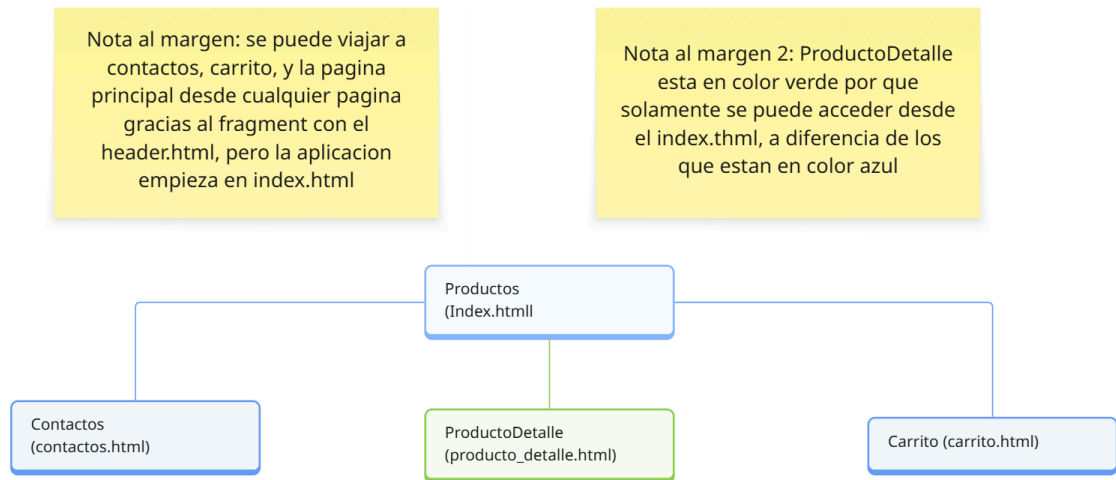
- ProductoRepository
- CarritoRepository
- UsuarioRepository, PerfilRepository, RolRepository
- MarcaRepository, CategoriaRepository
- CarritoProductoRepository

5. Diagrama de la Base de Datos (Modelo Entidad-Relación)



- **Descripción del Diagrama:**
 - **Entidades/Tablas Principales:** producto, usuario, carrito, marca, categoría.
 - **Relaciones Clave:** relación muchos a muchos entre carrito y producto, con una tabla intermedia para representarla, relación 1 a muchos entre producto y marca y entre producto y categoría. relación 1 a muchos entre usuario y carrito.
 - **Propósito:** La estructura modelada permite almacenar productos, marcas, categorías, crear compras (carritos), usuarios, al igual que editar los registros o eliminarlos

6. Sitemap / Mapa de la aplicación web:



miro