Universidad de San Carlos de Guatemala
Centro Universitario de Occidente
División de Ciencias de la Ingeniería

Manejo e Implementación de Archivos
Laboratorio

Proyecto final
E-commercegt
Manual Técnico

Jorge Anibal Bravo Rodríguez
202131782

HERRAMIENTAS USADAS

Solus linux

Es un sistema operativo (SO) de escritorio diseñado para la programación y el uso diario, conocido por su velocidad y estabilidad. Utiliza un modelo de desarrollo rolling release (actualizaciones continuas) y su propio entorno de escritorio llamado Budgie. Está optimizado para proveer un stack de desarrollo moderno y herramientas actualizadas, siendo una alternativa eficiente a otros sistemas basados en Linux para ejecutar tu entorno de desarrollo.

Visual studio code

Es un editor de código fuente ligero y de código abierto utilizado principalmente para el desarrollo de la aplicación Vue (Frontend) y la configuración general del proyecto. Su principal ventaja es su alto rendimiento y la vasta cantidad de extensiones que permiten la depuración, el highlighting de sintaxis y la autocompletación inteligente para JavaScript y archivos de configuración.

IntelliJ IDEA Community

Es el Entorno de Desarrollo Integrado (IDE) elegido para el desarrollo del servidor Spring Boot (Backend). Es una herramienta poderosa, especializada en Java, que ofrece funcionalidades avanzadas como la refactorización inteligente, el debugging eficiente y la integración nativa con herramientas de construcción como Maven o Gradle, siendo clave para la productividad en el desarrollo del servidor.

Temurin JDK 21

Es el Kit de Desarrollo de Java (JDK) de código abierto que proporciona el runtime necesario (la Máquina Virtual de Java o JVM) para compilar y ejecutar el servidor Spring Boot. Al ser una versión de Soporte a Largo Plazo (LTS), garantiza estabilidad y seguridad para la ejecución del backend del sistema.

Spring boot

Es el framework de Java empleado para construir la API RESTful del servidor (Backend). Spring Boot simplifica radicalmente la configuración, permitiendo crear una aplicación ejecutable y standalone (JAR). Es responsable de la lógica de negocio, la seguridad y la exposición de los endpoints a los que se conecta la aplicación frontend.

Postgresql

Es el sistema gestor de bases de datos relacional (ORDBMS) que almacena toda la información persistente del proyecto. Es reconocido por su robustez, su estricto cumplimiento del estándar SQL, su capacidad de manejar grandes volúmenes de datos de manera segura (transacciones ACID) y su alto rendimiento, asegurando la integridad de los datos.

Beekeeper studio
Es un cliente de bases de datos con interfaz gráfica (GUI). Esta herramienta se utiliza para visualizar, explorar y modificar los datos dentro de PostgreSQL de manera intuitiva. Es esencial para el mantenimiento y la depuración de la capa de persistencia durante y después del desarrollo.

Git
Es el sistema de control de versiones distribuido utilizado para rastrear y gestionar los cambios en el código fuente del proyecto a lo largo del tiempo. Permite a los desarrolladores colaborar eficientemente, gestionar distintas ramas de desarrollo (branches) y revertir a versiones anteriores del código si es necesario.

Github

Es la plataforma de alojamiento en la nube para los repositorios de Git. Funciona como el repositorio remoto central del proyecto, facilitando la colaboración, la revisión de código (Pull Requests) y sirviendo como la fuente de código de la que las herramientas de despliegue (Netlify) obtienen la versión lista para producción.

Vue

Es el framework de JavaScript utilizado para construir la interfaz de usuario (Frontend) como una Single Page Application (SPA). Su enfoque en componentes reactivos y la gestión de estado permite crear una experiencia de usuario moderna, modular y altamente eficiente, comunicándose con el backend a través de la API REST.

Netlify

Es la plataforma de hosting y despliegue continuo (CD) elegida para la aplicación Vue (Frontend). Netlify se integra directamente con GitHub para automatizar el proceso de construcción y despliegue, asegurando que cada actualización al código frontend se ponga en producción de manera rápida y eficiente.

Ngrok

Es una utilidad de tunelización segura que se usa durante la fase de desarrollo. Su función es crear un enlace público (HTTPS) que apunta al servidor Spring Boot que se está ejecutando localmente. Esto es crucial para que la aplicación Vue, ya desplegada en Netlify, pueda acceder al backend de desarrollo para pruebas de integración.

Draw.io

Es la herramienta de diagramación en línea utilizada para crear todos los diagramas técnicos del proyecto, incluyendo la arquitectura, los diagramas UML (de componentes, despliegue) y

los flujos del sistema. Su facilidad de uso y las plantillas estandarizadas aseguran que los gráficos del manual sean profesionales.

Zoho mail

Es la solución de correo electrónico empresarial utilizada como el canal de comunicación formal del proyecto. A nivel técnico, se configura como el servidor SMTP que utiliza el servidor Spring Boot para enviar correos electrónicos automatizados del sistema.

Descripción del proyecto

Objetivo General:

Desarrollar una aplicación en Java para la gestión de una biblioteca, que permita agregar, buscar, prestar y devolver libros, manteniendo la persistencia de datos mediante serialización de objetos y registrando un historial de operaciones con fecha, hora y usuario para garantizar trazabilidad.

Objetivos Específicos:

- Implementar la serialización de objetos para guardar y recuperar listas de libros y registros de préstamos.
- Mantener la persistencia de datos entre ejecuciones mediante archivos binarios.
- Gestionar colecciones de objetos serializables (listas de libros y préstamos).
- Manejar excepciones de entrada y salida (I/O) de manera adecuada.
- Diseñar menús de consola interactivos para realizar operaciones de la biblioteca.
- Aplicar identificadores únicos (UUID) para asegurar consistencia y trazabilidad de los datos.

Descripción de la aplicación:

La aplicación es un sistema de consola orientado a objetos que permite la gestión básica de una biblioteca:

- **Libros:** Cada libro tiene un UUID único, título, autor, año de publicación y estado de disponibilidad.
- **Préstamos:** Cada operación de préstamo o devolución se registra como un objeto Prestamo con información del libro, usuario, acción realizada y fecha/hora.
- **Biblioteca:** La clase principal centraliza la gestión de libros y del historial de operaciones, actualizando automáticamente los archivos binarios correspondientes.

Funcionalidades principales:

1. Agregar y buscar libros por título o UUID.
2. Prestar y devolver libros, actualizando su disponibilidad.
3. Registrar todas las operaciones en un historial cronológico con fecha, hora, libro y usuario.
4. Cargar libros desde archivos CSV, validando datos y evitando duplicados.
5. Generar reportes:
    - Libros completos, disponibles o prestados.
    - Historial completo o filtrado por libro/usuario.
    - Frecuencia de préstamos por libro y libros nunca prestados.

Consideraciones técnicas:

- Persistencia de datos mediante serialización de objetos.
- Validaciones de integridad: evitar duplicados, prestar libros no disponibles o devolver libros ya devueltos.
- Interfaz de consola clara y modular, con retroalimentación al usuario.
- Código organizado, limpio y documentado siguiendo buenas prácticas de programación.

Descripcion del proyecto

El proyecto E-COMMERCE GT es una plataforma de comercio electrónico consumidor a consumidor diseñada para automatizar la compra y venta de productos en línea, con enfoque en usuarios residentes de Guatemala.

El sistema está construido siguiendo una arquitectura de doble capa (Cliente-Servidor). El backend es una API RESTful desarrollada con Spring Boot y Java que gestiona la lógica de negocio, la seguridad (mediante JWT) y la persistencia de datos en una base de datos PostgreSQL normalizada. El frontend es una Single Page Application (SPA) desarrollada con Vue que ofrece una interfaz de usuario amigable y segmentada por roles.

El objetivo principal es proveer un entorno seguro y eficiente que soporte la gestión de múltiples roles de usuario (Común, Moderador, Logística, Administrador) y automatice procesos clave como la distribución de ganancias (95% para el vendedor, 5% para la plataforma) y el sistema de notificaciones por correo electrónico en tiempo real.

Funcionalidades

1. Funcionalidades del Usuario Común (Vendedor/Comprador)

- **Gestión de Ventas:**
    - Realizar la solicitud de puesta en venta de un nuevo producto (con validación de propiedades como Nombre, Descripción, Precio, Stock, etc.).
    - Actualizar la información de un producto en venta (requiere reingresar a un estado de revisión).
- **Gestión de Compras y Carrito:**
    - Acceso a un **Carrito de Compras** con funciones de: Agregar/Eliminar/Actualizar cantidades de productos, y Borrar Carrito completo.
    - **Comprar artículos** en el carrito con la única modalidad de pago de tarjeta de crédito.
    - Capacidad de **guardar la información de la tarjeta de crédito** para futuras compras.
- **Interacción y Seguimiento:**
    - **Calificar** (**Rating 1-5 estrellas**) **y dejar comentarios** en productos comprados.
    - Visualizar el **promedio de calificaciones** en la vista de detalle de un producto.
    - **Seguimiento de Pedidos:** Ver la lista de pedidos realizados, su estado actual ("**en curso**" **o** "**entregado**") y la fecha estimada de llegada (5 días después de la compra).

2. Funcionalidades del Usuario Moderador

- **Revisión de Productos:**
    - Visualizar y gestionar las **solicitudes de ingreso** de nuevos productos.
    - **Aceptar o Rechazar** la publicación de productos que los usuarios comunes ponen en venta.
- **Seguridad y Sanciones:**
    - **Gestionar Sanciones:** Suspender temporalmente cuentas de vendedores por fraudes o incumplimientos.
    - Registrar el motivo, fecha y estado de cada sanción para mantener un historial de seguridad.

3. Funcionalidades del Usuario de Logística

- **Gestión de Pedidos en Curso:**
    - Visualizar una lista completa de **todos los pedidos que están en estado "en curso"**.
    - **Actualizar el estado de los pedidos** a "entregado".
    - Modificar la fecha de entrega de cualquier pedido.

4. Funcionalidades del Usuario Administrador

- **Gestión de Empleados:**
    - **Registro de Nuevos Empleados:** Agregar usuarios de tipo Moderador, Logística o Administrador.
    - **Actualización de Información:** Modificar la información de cualquier empleado registrado en el sistema.
    - Visualizar el historial de empleados.
- **Generación de Reportes:**
    - Generar **7 reportes analíticos** en intervalos de tiempo definidos, incluyendo: Top 10 productos más vendidos, Top 5 clientes que más ganancias generan, Historial de sanciones, y Historial de notificaciones.

5. Funcionalidades Transversales del Sistema

- **Gestión de Roles y Autenticación:**
    - **Registro de Usuarios Comunes.**
    - **Inicio de Sesión** y gestión de acceso basada en **Token Web JSON** (**JWT**).
    - **Vistas Protegidas:** Restricción de acceso a vistas según el rol del usuario (Común, Moderador, Logística, Administrador).
- **Notificaciones Automatizadas** (**Zoho Mail**)**:**

    - Generar y enviar **notificaciones por correo electrónico** en tiempo real cuando:
        1. El pedido de un usuario cambie de estado.
        2. El producto en venta de un usuario sea aprobado o rechazado.
    - Almacenar un **historial de notificaciones** en la base de datos.
- **Economía y Financiero:**
    - **Distribución de Ganancias** por cada venta (95% para el vendedor, 5% para la plataforma).
- **Integración y Despliegue:**
    - **Despliegue del Cliente** (**Vue**) en **Netlify**.
    - Exposición del **Servidor** (**Spring Boot**) vía **HTTPS** utilizando **Ngrok** (para demostración/pruebas).

Organización del proyecto

A) BACKEND CON SPRING

El proyecto E-COMMERCE GT API sigue una arquitectura de tres capas basada en el patrón Modelo-Vista-Controlador (MVC), adaptada al desarrollo de una API REST, con una clara separación de responsabilidades:

1. **Capa de Presentación** (**Controladores**)
    - Paquete: com.jbrod.ecommerce_api.controladores
    - Responsabilidad: Manejar las peticiones HTTP entrantes (GET, POST, etc.), validar los datos de entrada a través de DTOs y delegar la lógica de negocio a la capa de servicios. Utiliza Spring Security (Principal) para identificar al usuario autenticado a través del token JWT.

2. **Capa de Lógica de Negocio** (**Servicios**)
    - Paquete: com.jbrod.ecommerce_api.servicios
    - Responsabilidad: Contiene la lógica de negocio del sistema. Esta capa es transaccional y coordina múltiples operaciones de base de datos para asegurar la consistencia. Aquí se implementan requisitos clave como:
        - Procesamiento de *checkout* (creación de Pedido, descuento de *stock* y registro de DetalleVentaVendedor).
        - Cálculo y distribución de ganancias (95% Vendedor, 5% Plataforma).
        - Gestión de errores de negocio (ej. *Stock* insuficiente).

3. **Capa de Persistencia** (**Modelos y Repositorios**)
    - Modelos (Entidades): com.jbrod.ecommerce_api.modelos
        - Responsabilidad: Mapear las tablas de la base de datos PostgreSQL a objetos Java utilizando JPA/Hibernate. Los modelos están organizados por dominio (/pedidos, /usuario, /productos).
        - Modelos Clave de Pedidos: Pedido, ListaProductoPedido (detalle del pedido), DetalleVentaVendedor (registro de comisión) y RecaudacionPlataforma (registro del 5% de la comisión).
    - Repositorios (DAO): com.jbrod.ecommerce_api.repositorios
        - Responsabilidad: Proporcionar los métodos para interactuar directamente con la base de datos (CRUD) utilizando Spring Data JPA.

4. **Organización de Datos** (**DTOs**)

- Paquete: com.jbrod.ecommerce_api.dto
- Responsabilidad: Definir los Objetos de Transferencia de Datos (DTOs). Estos objetos son la única estructura de datos que se permite ingresar o salir de la API para garantizar la seguridad, la validación y el desacoplamiento de las entidades de la base de datos.

B) FRONTEND CON VUE

El cliente del proyecto E-COMMERCE GT es una Single Page Application (SPA) construida con Vue.js, diseñada para ser rápida, reactiva y fácil de mantener. Su estructura se basa en el principio de separación de preocupaciones por módulos.

Estructura de Directorios Principales (src/)

Directorio	**Propósito y Responsabilidad**
api/	Capa de Comunicación (Data Access Layer): Contiene archivos con funciones específicas para interactuar con la API REST de Spring Boot. Centraliza la lógica de las peticiones HTTP (URL, parámetros, método) para que las vistas y *stores* no se mezclen con el acceso a datos.
components/	Componentes Reutilizables: Aloja todos los componentes de la interfaz de usuario. Están organizados por rol (/comun, /administrador, etc.) para agrupar elementos específicos de cada tipo de usuario.
layouts/	Estructura de Vistas especiales para cada usuario.
views/	Vistas de la Aplicación: Contiene los componentes que representan las páginas completas. Organizado jerárquicamente por rol (/comun,

	/administrador) y luego por funcionalidad (/pedidos, /productos), facilitando la navegación lógica.
stores/ (Pinia)	Gestión Global de Estado: Utiliza Pinia (la librería de gestión de estado de Vue) para almacenar y gestionar los datos de la aplicación (ej. datos de usuario, estado del carrito). Contiene la lógica centralizada para la gestión de datos clave.
plugins/	Configuración de Herramientas: Configuración global de utilidades de terceros. El archivo axios.js es clave para configurar la URL base dinámica de Ngrok y los interceptores que gestionan el token JWT.