ESCUELA POLITÉCNICA DE PACHUCA

Materia: Arquitectura de Software

Tema: Reporte de Proyecto Final

Maestro: Jazmin Rodriguez Flores

Alumnos:

Jesús Alejandro Roldán Ortega 2311122575

Jose Pablo Espinoza Lopez 2311122696

Gael Rojo Fuentes 2311122574

Carlos Isaac Leal Rodriguez 2311122567

09/08/2024

Índice

Introducción

En el mundo del comercio de calzado, la gestión eficiente del inventario es crucial para asegurar la disponibilidad de productos, minimizar pérdidas y maximizar las ganancias. El presente proyecto se centra en el desarrollo de un gestor de inventario específicamente diseñado para una zapatería. La herramienta permitirá un control detallado y en tiempo real del stock de productos, facilitando la toma de decisiones informadas y mejorando la experiencia del cliente. El gestor de inventario propuesto incorporará funcionalidades clave como la entrada y salida de productos y el seguimiento de niveles de stock.

Listado de requisitos

Requisitos Funcionales

1. **Gestión de Productos**
    - Añadir, editar, eliminar y ver productos.
2. **Gestión de Ventas**
    - Registrar ventas, ver historial de ventas y generar informes.
3. **Gestión de Inventario**
    - Monitoreo de stock y alertas de bajo inventario.
4. **Usuarios y Roles**
    - Autenticación y roles para diferentes niveles de acceso (administrador, usuario).
5. **Reportes**
    - Generación de reportes de ventas e inventario.

Requisitos No Funcionales

1. **Usabilidad**
    - Interfaz intuitiva y fácil de usar.
    - Entrada de datos rápida y eficiente.
    - Soporte multilenguaje (principalmente español).
2. **Rendimiento**
    - Carga del formulario en menos de 2 segundos.
    - Manejo de hasta 1000 productos y 100 ventas diarias sin degradación significativa.
3. **Seguridad**
    - Autenticación segura con HTTPS y contraseñas cifradas.

- Control de acceso basado en roles.
- Protección contra SQL Injection y Cross-Site Scripting (XSS).

4. **Compatibilidad**
    - Compatible con navegadores modernos y versiones recientes de Windows.
    - Adaptable para diferentes resoluciones de pantalla.
5. **Mantenibilidad**
    - Código bien documentado para facilitar actualizaciones.
    - Diseño modular para añadir nuevas funcionalidades sin afectar las existentes.
6. **Disponibilidad**
    - Disponibilidad del sistema al menos el 99.9% del tiempo, con mantenimiento fuera de horas laborales.
7. **Escalabilidad**
    - Capacidad de expansión para soportar más usuarios o productos sin necesidad de reestructuración mayor.
8. **Accesibilidad**
    - Cumplimiento con las pautas de accesibilidad web (WCAG) para personas con discapacidades.

Herramientas y Lenguajes

- FrameWork ASP.NET Core
- MySQL 20.1
- Google Cloud services
- Visual Studio Code 2022
- C#
- JavaScript
- CCS
- Tallwin
- PC
- GitHub
- Trello

Representación de las historias de usuario

HU 1

Registrar Productos

- **Qué**: Registrar nuevos productos en el sistema.
- **Como**: Como administrador.
- **Para**: Para mantener un catálogo actualizado y permitir la venta de nuevos productos.

HU 2

Actualizar Inventario

- **Qué**: Actualizar el inventario existente con nuevas unidades.
- **Como**: Como administrador.
- **Para**: Para reflejar correctamente la cantidad de productos disponibles y asegurar una gestión precisa del stock.

HU 3

Buscar Productos

- **Qué**: Buscar productos específicos en el inventario.
- **Como**: Como administrador o empleado.
- **Para**: Para localizar rápidamente productos en el sistema y agilizar las consultas de inventario.

HU 4

Filtrar Productos por Categoría

- **Qué**: Filtrar productos por categoría.
- **Como**: Como administrador o cliente.
- **Para**: Para encontrar productos de una categoría específica fácilmente y mejorar la experiencia de búsqueda.

HU 5

Notificar Stock Bajo

- **Qué**: Recibir notificaciones cuando el stock de un producto esté bajo.
- **Como**: Como administrador.

- **Para**: Para tomar acciones oportunas y reabastecer productos antes de que se agoten.

HU 6

Eliminar Productos

- **Qué**: Eliminar productos del inventario.
- **Como**: Como administrador.
- **Para**: Para retirar productos obsoletos o no deseados del catálogo y mantener un inventario actualizado.

HU 7

Vista de Inventario en Tiempo Real

- **Qué**: Ver el estado del inventario en tiempo real.
- **Como**: Como administrador.
- **Para**: Para tomar decisiones informadas sobre la gestión de inventario basadas en datos actuales.

HU 8

Registrar Productos con Foto

- **Qué**: Registrar productos incluyendo una foto.
- **Como**: Como administrador.
- **Para**: Para proporcionar una referencia visual a los clientes y mejorar la experiencia de compra.

HU 9

Control de Acceso y Permisos

- **Qué**: Definir niveles de acceso y permisos para los empleados.
- **Como**: Como administrador.
- **Para**: Para asegurar que solo personas autorizadas puedan realizar cambios críticos en el sistema.

HU 10

Inicio de sesión

- **Qué**: Iniciar sesión en el sistema.
- **Cómo**: Como usuario registrado, quiero ingresar mi nombre de usuario y contraseña.

- **Para**: Acceder a mis funcionalidades asignadas según mi rol (administrador, usuario).

HU 11

Registro

- **Qué**: Registrar un nuevo usuario en el sistema.
- **Cómo**: Como nuevo usuario, quiero poder registrarme proporcionando mis datos personales y de acceso.
- **Para**: Crear una cuenta que me permita utilizar el sistema según mi rol asignado.

HU 12

Cerrar sesión

- **Qué**: Cerrar sesión en el sistema.
- **Cómo**: Como usuario autenticado, quiero poder cerrar sesión de manera segura.
- **Para**: Finalizar mi sesión y proteger mi cuenta de accesos no autorizados.

Asignación de trabajo y actividades

- **Alex, Gael:** Desarrollo del frontend con HTML/CSS, JavaScript y frameworks de CSS.
- **Alex, Pablo:** Desarrollo del backend con JavaScript y C#
- **Pablo:** Configuración y manejo de la base de datos en SQL Google Cloud
- **Gael, Alex:** Implementación de la autenticación con JWT y gestión de usuarios.

Tabla de quemado

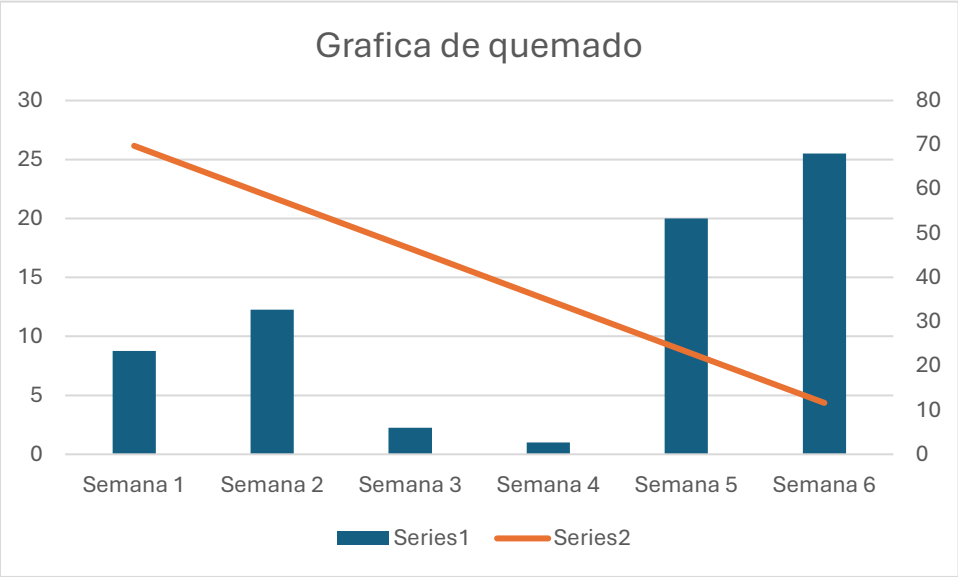
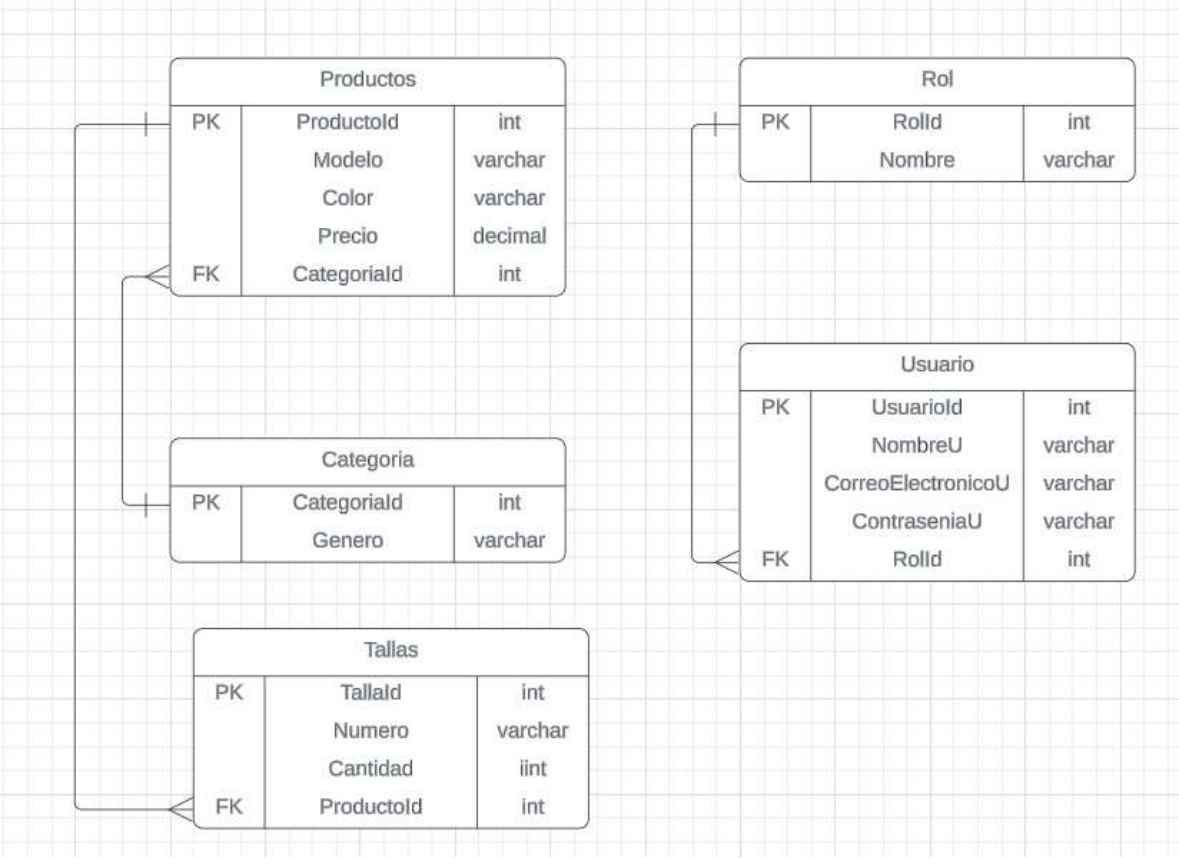


Tabla de quemado 1.1

Esquema de datos



Esquema de datos grafica 1.1

Descripción e implementación de patrones de diseño

Dentro del Modelo:

- **Singleton:** Para garantizar que solo exista una instancia de un objeto, como una conexión a base de datos.
- **Observer:** Para notificar a las vistas cuando los datos del modelo cambian.
- **Strategy:** Para encapsular diferentes algoritmos para realizar una tarea, como diferentes formas de calcular un total.

Dentro de la Vista:

- **Composite:** Para representar una jerarquía de componentes de la interfaz de usuario.
- **Decorator:** Para agregar funcionalidades a los componentes de la vista de forma dinámica.
- **Strategy:** Para cambiar la apariencia de una vista en función de diferentes criterios.

Dentro del Controlador:

- **Command:** Para encapsular una solicitud como un objeto, permitiendo su deshacer, rehacer y registro.
- **Front Controller:** Para manejar todas las solicitudes de la aplicación y delegarlas a controladores más específicos.
- **Interceptor:** Para interceptar las solicitudes y realizar acciones antes o después de que se procese la solicitud.

Entre Modelo, Vista y Controlador:

- **Facade:** Para proporcionar una interfaz simplificada a un subsistema complejo (por ejemplo, un conjunto de clases del modelo).
- **Mediator:** Para desacoplar las vistas y el controlador, permitiendo una comunicación más flexible entre ellos.

Descripción e implementación de principios de diseño

- **Alineación:** Todos los elementos de un diseño deben estar alineados de alguna manera, ya sea a un borde, a un centro o a una cuadrícula. Esto crea una sensación de orden y conexión.
- **Proximidad:** Los elementos relacionados deben estar agrupados visualmente para indicar su relación.
- **Contraste:** El contraste entre elementos (color, tamaño, forma) atrae la atención y crea énfasis.
- **Jerarquía:** Los elementos de un diseño deben tener una jerarquía visual, lo que significa que algunos elementos deben destacar más que otros.
- **Espacio en blanco:** El espacio en blanco (o negativo) es tan importante como los elementos positivos. Ayuda a separar los elementos y a crear una sensación de aire.
- **Equilibrio:** El equilibrio visual se logra distribuyendo los elementos de manera que el diseño se sienta estable y armonioso.
- **Patrones:** Los patrones son elementos repetitivos que pueden crear una sensación de orden y complejidad.

Descripción de arquitectura implementada

- **Arquitectura MVC**
    1. **Frontend**: Laravel para una interfaz de usuario interactiva.
    2. **Backend**: Node.js con Express.js para manejar las operaciones de servidor.
    3. **Base de Datos**: Management Studio para almacenar datos de productos y ventas.
    4. **Autenticación**: JWT (JSON Web Tokens) para la gestión de usuarios y autenticación.

Justificación de la arquitectura implementada

La elección de la arquitectura Modelo-Vista-Controlador (MVC) para el desarrollo del formulario de la zapatería está justificada por varias razones clave que contribuyen a la eficiencia, escalabilidad y mantenibilidad del sistema.

La facilidad de separación de responsabilidades que ofrece el MVC es fundamental. El **Modelo** maneja toda la lógica relacionada con los datos y las reglas del negocio, como la gestión de la base de datos y las operaciones CRUD (Crear, Leer, Actualizar, Eliminar). Esto asegura que la lógica de datos sea coherente y

reutilizable, reduciendo el riesgo de errores y facilitando las modificaciones futuras. La **Vista** se encarga de la presentación, mostrando al usuario los datos a través de formularios, tablas y otros elementos de la interfaz. Esto permite que el diseño de la interfaz de usuario se mantenga independiente de la lógica de negocio. El **Controlador**, por su parte, actúa como un intermediario entre el Modelo y la Vista, gestionando las solicitudes del usuario y determinando qué datos se deben mostrar y cómo interactúan con la lógica de negocio.

Además, la arquitectura MVC facilita la **mantenibilidad** del sistema. Al tener el código organizado en módulos separados, las actualizaciones o mejoras en una parte del sistema se pueden realizar sin afectar las demás. Esto es especialmente útil para la evolución de la aplicación a lo largo del tiempo, permitiendo incorporar nuevas funcionalidades o cambiar las existentes de manera más ágil.

Otra ventaja es la **escalabilidad**. MVC permite que la aplicación crezca en complejidad sin perder orden ni estructura, lo que es crucial para un sistema que puede necesitar añadir nuevas funcionalidades como reportes avanzados, gestión de usuarios o integración con otros sistemas.

MVC promueve la **reutilización de código**. El Modelo puede ser reutilizado en diferentes partes de la aplicación o incluso en otras aplicaciones que compartan la misma lógica de negocio, lo que incrementa la eficiencia en el desarrollo.

Plan de riesgos

1. Identificación de Riesgos

ID	Riesgo	Descripción
R01	Falta de Requisitos Claros	Los requisitos del sistema no están bien definidos, lo que lleva a malentendidos y errores en el desarrollo.
R02	Fallos de Seguridad	La base de datos y el sistema pueden ser vulnerables a ataques, comprometiendo la información sensible.
R03	Errores en la Integración	Problemas al integrar el sistema CRUD con la base de datos en Google Cloud SQL.
R04	Baja Performance	El sistema puede volverse lento o ineficiente con un gran volumen de datos.
R05	Pérdida de Datos	Posibles pérdidas de datos debido a fallos del sistema o errores humanos.
R06	Falta de Capacitación	El personal no está adecuadamente capacitado para usar el nuevo sistema.
R07	Problemas de Compatibilidad	El sistema puede no ser compatible con otros sistemas existentes en la zapatería.
R08	Retrasos en el Cronograma	El proyecto puede sufrir retrasos debido a problemas imprevistos.

R09	Exceder el Presupuesto	El proyecto puede exceder el presupuesto asignado debido a costos imprevistos.
R10	Dependencia de Proveedores	Problemas con proveedores de software o servicios pueden afectar el desarrollo.

Plan de riesgos tabla 1.1

2. Evaluación de Riesgos

Cada riesgo fue evaluado en términos de probabilidad (P) y impacto (I) en una escala de 1 a 5 (1 = bajo, 5 = alto).

ID	Riesgo	P	I	Severidad (P x I)
R01	Falta de Requisitos Claros	3	4	12
R02	Fallos de Seguridad	2	5	10
R03	Errores en la Integración	3	4	12
R04	Baja Performance	2	4	8
R05	Pérdida de Datos	2	5	10
R06	Falta de Capacitación	3	3	9
R07	Problemas de Compatibilidad	2	4	8
R08	Retrasos en el Cronograma	3	3	9
R09	Exceder el Presupuesto	2	4	8
R10	Dependencia de Proveedores	3	4	12

Plan de riesgos tabla 1.2

3. Estrategias de Mitigación

ID	Riesgo	Estrategia de Mitigación
R01	Falta de Requisitos Claros	Realizar sesiones de análisis detallado con los stakeholders y documentar claramente todos los requisitos.
R02	Fallos de Seguridad	Implementar medidas de seguridad como encriptación, autenticación y realizar pruebas de penetración regularmente.
R03	Errores en la Integración	Realizar pruebas de integración frecuentes y contar con documentación detallada de la base de datos y el sistema.
R04	Baja Performance	Optimizar el diseño de la base de datos y el código del sistema, y realizar pruebas de carga.
R05	Pérdida de Datos	Implementar políticas de backup y recuperación de datos, y realizar pruebas periódicas de restauración.
R06	Falta de Capacitación	Proveer capacitación adecuada y recursos de soporte continuo para el personal.
R07	Problemas de Compatibilidad	Evaluar la compatibilidad del sistema con otros sistemas existentes antes de la implementación.
R08	Retrasos en el Cronograma	Establecer un cronograma realista y hacer un seguimiento continuo del progreso del proyecto.
R09	Exceder el Presupuesto	Monitorear continuamente los costos del proyecto y ajustar el alcance si es necesario.

R10	Dependencia de Proveedores	Tener planes de contingencia y alternativas para proveedores críticos.

Plan de riesgos tabla 1.3

Monitoreo y Revisión

El equipo de gestión de riesgos se reunió de manera regular para evaluar el estado de los riesgos y realizar ajustes pertinentes en las estrategias de mitigación. Esto incluyó la actualización de la evaluación de riesgos, la identificación de nuevos riesgos y la evaluación de la efectividad de las estrategias implementadas. El cual se derivo de la siguiente manera:

Semana 1

Se gestiono el proyecto con el cliente y sus necesidades, las actividades, materiales a utilizar e historias de usuario, listado de requisitos funcionales y no funcionales

Semana 2

Se evaluaron las HU y CU

Semana 5

Se gestiono el presupuesto, plan de riesgos patrones de diseño y arquitectura implementada y la asignación de trabajo.

Semana 9

Se creo el Backend, Frontend, Base de Datos, documentación, monitoreo, revisión y poner en producción BD y Formulario

Costo del proyecto

Para la realización del gestor de inventario para zapaterías, se realizó una estimación de costos involucrados, incluyendo salarios de personal, así como gastos de software y Hardware. A continuación, se desglosan los componentes del costo total del proyecto:

Sueldo de los trabajadores	**Pago semanal**	**Pago total**
Jesus Alegando	2,222	20,000
Jose Pablo	2,088	18,800
Gael	1,111	10,000
Carlos Isaac	2,777	-25,000
Material para desarrollo	**Pago unico**	
Computadores	$60,000	

Servicios	$1,200
Software de desarrollo	**Pago por mantenimiento**
Visual Studio Code	$0
Google Cloud Services	$50
Total	$135,050

Tabla de costo del proyecto 1

Sumando dichos componentes, el costo total estimado para la creación del gestor de inventario es de $135,050. Esta cifra proporciona una visión clara de la inversión requerida y asegura que todos los aspectos del desarrollo y la implementación estén adecuadamente financiados.

En conclusión, la estimación de costos incluye tanto los gastos de recursos humanos como los materiales, asegurando que el proyecto pueda ser llevado a cabo de manera efectiva y con los medios necesarios.

Conclusión

El desarrollo del gestor de inventario para zapaterías ha logrado proporcionar una solución robusta y eficiente para la gestión de productos en el comercio de calzado. El proyecto ha sido diseñado con una arquitectura MVC que asegura una separación clara de responsabilidades, facilitando el mantenimiento y la escalabilidad del sistema.

La implementación de funcionalidades clave, como la administración de productos, el registro y seguimiento de ventas, y el control detallado del inventario, ha optimizado las operaciones y mejorado la experiencia del cliente. Además, la incorporación de patrones de diseño y principios de diseño ha contribuido a la creación de una interfaz de usuario intuitiva y un sistema bien estructurado.

La evaluación y mitigación de riesgos ha sido fundamental para garantizar la ejecución exitosa del proyecto, abordando posibles problemas antes de que afecten el desarrollo. La estimación de costos ha proporcionado una visión clara del presupuesto necesario, asegurando que todos los recursos necesarios estén disponibles para una implementación efectiva.

En resumen, el proyecto cumple con los objetivos propuestos y demuestra la importancia de una planificación y ejecución meticulosa en el desarrollo de sistemas de gestión. La herramienta creada no solo optimiza las operaciones internas, sino que también mejora significativamente la experiencia de los clientes y contribuye al éxito general de la zapatería.

Anexos

UNIVERSIDAD POLITÉCNICA DE PACHUCA

ASIGNATURA: Arquitectura de software

LISTA DE COTEJO DE PROYECTO FINAL

ALUMNOS:

INSTRUCCIONES: **Revisar las actividades que se solicitan y marque en los apartados** correspondientes de acuerdo al desempeño del alumno. Si no participan todos los integrantes se penaliza a todo el equipo. Todos deben desarrollar y programar su parte.

Código	*Característica a cumplir (Reactivo)*	*Puntaje*			**OBSERVACIONES**
		Claramente (3)	*Faltan algunos elementos(2)*	*No es claro (0)*	
1	Listado de requisitos. (5)				
2	Representa los CU o HU (5)				
3	Se encuentra dividido el proyecto entre los integrantes equitativamente (5)				
3.1	Identifica claramente las herramientas y lenguajes a usar (5)				
4	Modelo de BD. Contiene un modelo de datos normalizado (5)				
5	Cada integrante realizó los módulos que les fueron asignados (10)				
6	Se describe e implementa al menos un patrón de diseño (5)				
7	Se describe e implementa al menos un principio de diseño (5)				
8	Describe el tipo de arquitectura implementada (10) MVC, MVV				
9	Justifica la aplicación de la arquitectura implementada (5)				
11	AE2. Aplicar los procesos de desarrollo de software para resolver proyectos que cumplen con los requisitos de los clientes (Metodología de desarrollo 10 explicada y con sus etapas EN EL DOCUMENTO)				
12	El programa servidor es funcional (10)				
13	El programa cliente es funcional (5)				
14	Estableció el plan de riesgos para el proyecto (5)				
15	Establece un costo para el proyecto (5)				
16	El proyecto se entregó en tiempo (3)				
17	Autoevaluación (2)				
	CALIFICACIÓN:				