Proyecto final

Ingeniería de software I

Kevin David Rodriguez Riveros

Frank Sebastian Pardo Amaya

Jorge Andrés Torres Leal

Departamento de Ingeniería de Sistemas e Industrial

Universidad Nacional de Colombia - Sede Bogotá

Facultad de Ingeniería

31 de Enero de 2025

1. Levantamiento de Requerimientos

En principio con el equipo tuvimos varias ideas a desarrollar, concordamos en un problema común en pequeñas y medianas empresas con: la gestión de sus inventarios. Muchas empresas aún dependen de hojas de cálculo o registros manuales, nosotros mismos conocemos gente que lo sigue haciendo a día de hoy. Sabemos que eso puede generar errores, pérdida de información, falta de precisión en la misma y también dificultades en el control de productos. Viéndolo así, decidimos desarrollar StockEase, como sistema de gestión de inventarios que facilita el registro, control y seguimiento de productos de manera sencilla e intuitiva.

Pensamos en un sistema que permitiera registrar cada producto con detalles clave como nombre, código, precio de compra y venta, cantidad, lote y fecha de vencimiento. Además ofrece una interfaz de reportes donde los usuarios pueden filtrar productos por nombre, código o rango de fechas, asegurando un acceso rápido a la información. Hasta el momento hemos implementado la funcionalidad de registro y visualización de productos y planeamos también incluir módulos para gestión de ventas y alertas automáticas para productos próximos a vencer o con bajo stock.

En lo técnico trabajamos con Java para el desarrollo del software y definimos la estructura de la base de datos para garantizar un almacenamiento eficiente y seguro. Nuestro objetivo es que StockEase sea una solución accesible y práctica para cualquier chuzo que necesite mejorar su gestión de inventarios sin depender de sistemas complejos y caros.

2. Análisis de Requerimientos

- **Clasificación MoSCoW:**

Funcionalidad	**Clasificación**	**Tiempo Estimado**
Gestionar Perfiles y Credenciales de acceso	Must	3 días
Registro de productos	Must	3 días
Gestión de Ventas	Must	5 días
Sistema de autenticación de usuario	Must	5 días
Editar/Eliminar productos	Must	8 días
Informe de ventas	Must	8 días
Alertas de stock bajo	Could	21 días
Modo Oscuro	Should	13 días

Alerta de vencimiento de productos	Should	13 días
Factura Electrónica	Won’t	N/A
App Móvil	Won’t	N/A

3. Análisis de gestión de Software

El desarrollo de StockEase se divide en fases con tiempos estimados para garantizar un desarrollo eficiente y bien estructurado.

Fase	Actividad	Duración
Diseño y planificación.	Definición de requerimientos y definir qué utilizar para el proyecto	2 semanas
Desarrollo del MVP .	Implementación de funcionalidades clasificadas como "Must"	4 semanas
Pruebas y ajustes .	Testing, corrección de problemas encontrados	2 semanas

Recursos Humanos: El equipo de desarrollo está conformado por tres programadores que trabajarán en esta implementación de gestor de inventarios.

Elemento	**Costo**
Programador Junior*2	$6’000,000 COP
Diseñador Interfaz	$5’000,000 COP.

Costo estimado total: $25,000,000 COP.

MVP: $17,000,000 COP.

- Interfaz básica de búsqueda.
- Registro, visualización, edición y eliminación de productos en el inventario con nombre, cantidad, precio entre otros detalles.
- Base de datos con almacenamiento de inventario.

4. Diseño y Arquitectura

Arquitectura del Sistema: Se utilizó una arquitectura MVC (Modelo Vista Controlador)

Modelo (Model)

- Representa la **lógica de negocio** y la gestión de datos.
- Maneja la interacción con la **base de datos** y las reglas de negocio.
- No depende directamente de la interfaz gráfica.
- Puede notificar a la vista sobre cambios en los datos.

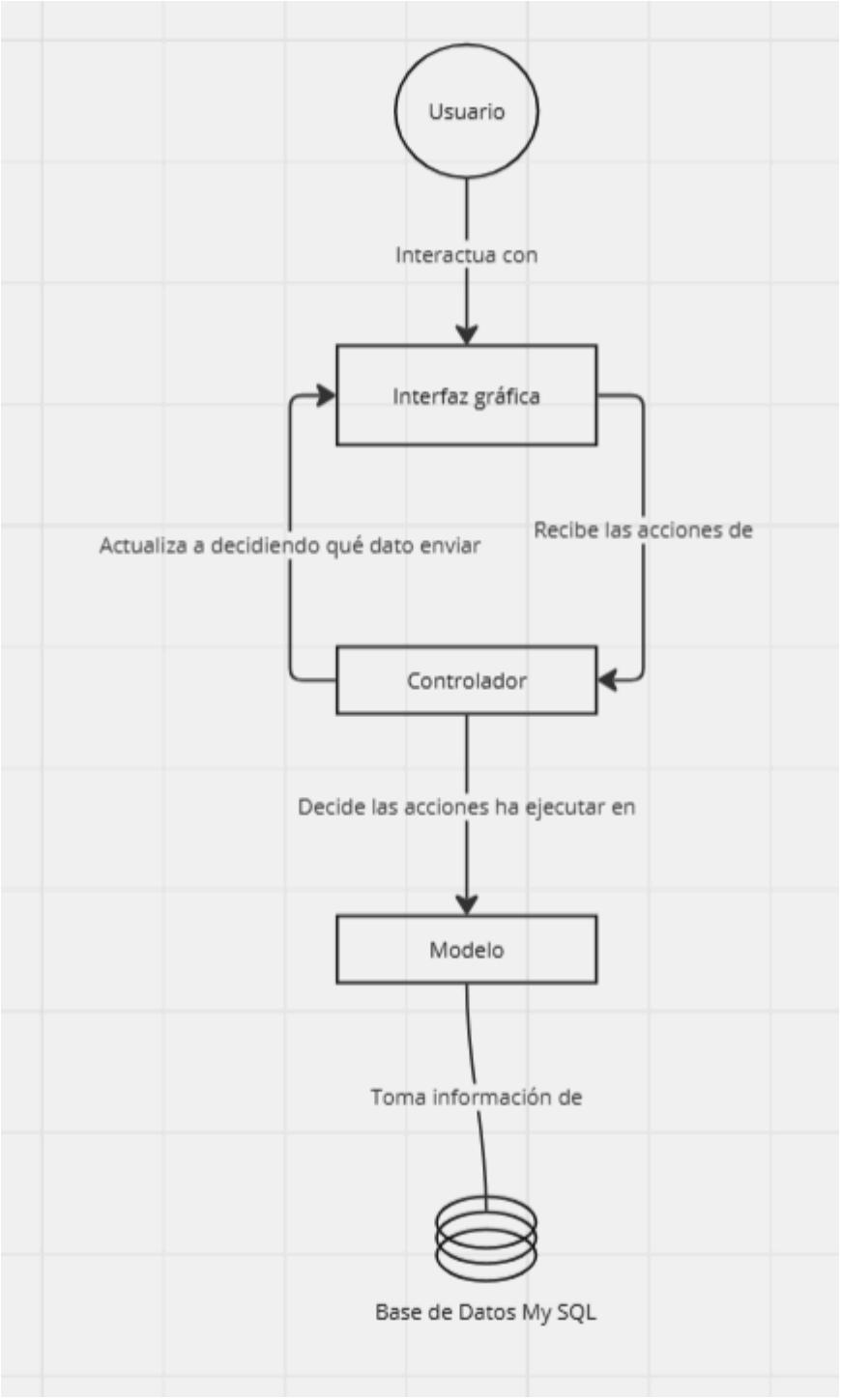
Vista (View)

- Se encarga de la **interfaz de usuario** y la presentación de los datos.
- No tiene lógica de negocio, solo muestra información obtenida del modelo.
- Puede ser una interfaz gráfica (JavaFX, Swing) o una página web (HTML, CSS, JavaScript).

Controlador (Controller)

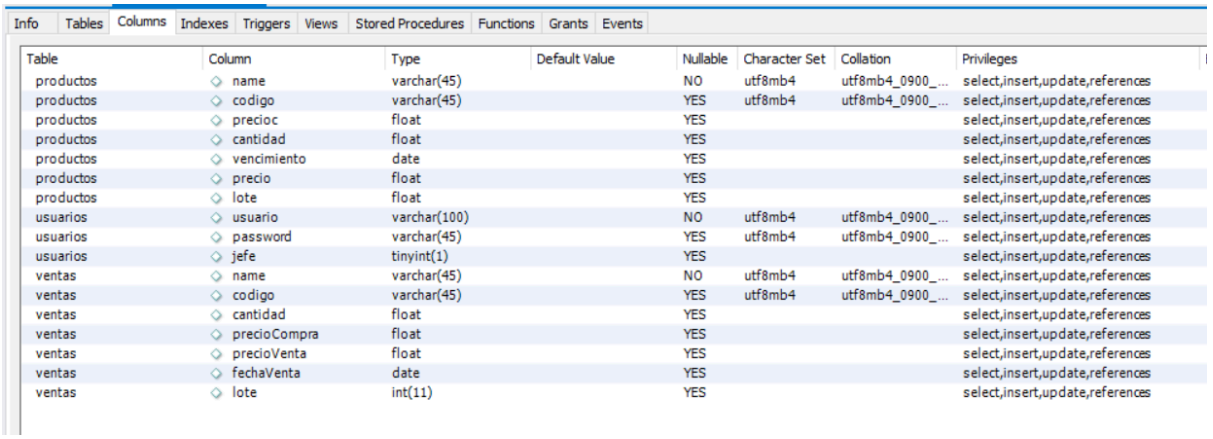
- Actúa como intermediario entre la vista y el modelo.
- Recibe las acciones del usuario desde la vista y actualiza el modelo.
- Decide qué datos enviar a la vista y qué acciones ejecutar en el modelo.

Se eligió este modelo ya que la parte gráfica se pudo aislar a algo más low-code por medio de hacerlo con JavaFX y SceneBuilder, además se pueden realizar pruebas unitarias en el modelo sin depender de la interfaz gráfica, se pueden simular vistas o modelos en entornos de prueba sin afectar el resto del sistema, soporta múltiples vistas sin modificar la lógica central del modelo.



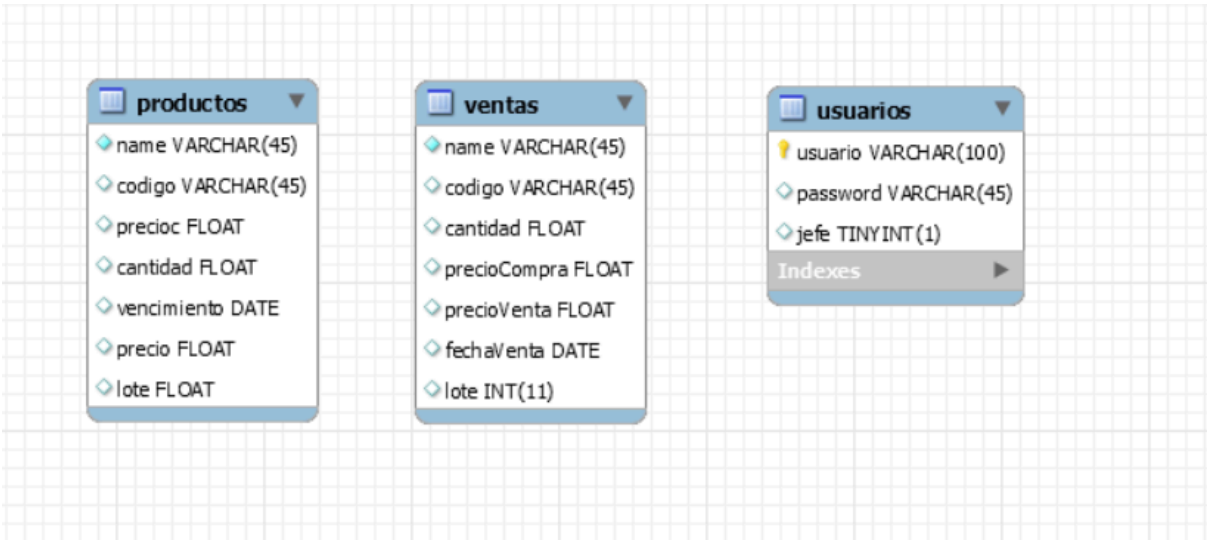
Diseño de Base de Datos: La base de datos tiene tres tablas las cuales son productos, usuarios y ventas, básicamente la de usuarios es una tabla individual sin ningún tipo de conexión con las demás ya que solo se encarga de administrar los nombres, contraseñas y niveles de los empleados (Gerentes o empleados), luego sí existe una relación de uno a muchos entre productos y ventas ya que un producto puede tener varias ventas, sin embargo esto no se ha implementado, además se piensa añadir en un futuro una relación en la cual se refleje quién realizó x o y venta, lo cual implica relacionar usuarios y ventas, como claves primarias se tendría en productos el código junto con el lote, y como llave foránea se tendría en venta el lote y el código del producto.

Se eligió una base de datos SQL ya que es importante para futuros desarrollos el que exista una estructura bien definida de las relaciones entre tablas, incluso si aún no existe completamente.



Info | Tables | Columns | Indexes | Triggers | Views | Stored Procedures | Functions | Grants | Events

Table	Column	Type	Default Value	Nullable	Character Set	Collation	Privileges
productos	name	varchar(45)		NO	utf8mb4	utf8mb4_0900_...	select,insert,update,references
productos	codigo	varchar(45)		YES	utf8mb4	utf8mb4_0900_...	select,insert,update,references
productos	precioc	float		YES			select,insert,update,references
productos	cantidad	float		YES			select,insert,update,references
productos	vencimiento	date		YES			select,insert,update,references
productos	precio	float		YES			select,insert,update,references
productos	lote	float		YES			select,insert,update,references
usuarios	usuario	varchar(100)		NO	utf8mb4	utf8mb4_0900_...	select,insert,update,references
usuarios	password	varchar(45)		YES	utf8mb4	utf8mb4_0900_...	select,insert,update,references
usuarios	jefe	tinyint(1)		YES			select,insert,update,references
ventas	name	varchar(45)		NO	utf8mb4	utf8mb4_0900_...	select,insert,update,references
ventas	codigo	varchar(45)		YES	utf8mb4	utf8mb4_0900_...	select,insert,update,references
ventas	cantidad	float		YES			select,insert,update,references
ventas	precioCompra	float		YES			select,insert,update,references
ventas	precioVenta	float		YES			select,insert,update,references
ventas	fechaVenta	date		YES			select,insert,update,references
ventas	lote	int(11)		YES			select,insert,update,references



5. Patrones de diseño.

Aún no hemos empezado a utilizar un patrón de diseño claro ya que aún no lo sentimos necesario dentro de la estructura de un proyecto relativamente pequeño.