

Proyecto final

Materia:
ingeniería de software I


Presentado por:
Adrian Ramirez Gonzalez
Adrian Alexander Benavides
Andrés Hernando Borda Muñoz
Brayan Alejandro Muñoz Pérez



Profesor:
Oscar Eduardo Álvarez Rodríguez

Universidad Nacional de Colombia
Facultad de ingeniería
Viernes 31 de enero del 2025

Punto 1: Pre requisitos

README.md

 README



Repositorio grupal - Ingeniería de software 1 - 2024-2

Grupo 3

Integrantes:

- Adrian Alexander Benavides Caguasango
- Brayan Alejandro Muñoz Pérez
- Adrian Ramirez Gonzalez
- Andrés Hernando Borda Muñoz

Nombre del proyecto: Boxy

Descripcion del objetivo del proyecto

El objetivo de Boxy es desarrollar una solución integral de gestión de inventarios que permita a empresas optimizar y controlar el almacenamiento de sus productos de manera eficiente. Boxy busca combinar una interfaz sencilla y accesible con funcionalidades interesantes como el seguimiento de inventarios, identificación de stock bajo, generación de informes y registro de entradas y salidas de productos.

Tecnologias usadas

Lenguajes de programacion

- Python
- JavaScript








Frameworks

- Backend: django REST
- Frontend: Vuejs

Base de datos

- SQLite

.gitignore

 adrianram21 Subida taller 1 y 2			a37f297 · 13 minutes ago	 27 Commits
 Asignaciones	Subida Taller 1 - Requerimientos	last month		
 Documentacion	Subida taller 1 y 2	13 minutes ago		
 Proyecto	Subida taller 1 y 2	13 minutes ago		
 .gitignore	Adicion git ignore	last month		
 README.md	Actualizacion del README	5 days ago		

Punto 2: Levantamiento de requerimientos

Nombre del proyecto: Boxy

Se quiere desarrollar una aplicación web para la gestión de inventarios en empresas, con el objetivo de optimizar y controlar el almacenamiento de productos de manera eficiente. Este proyecto surge de la necesidad de abordar los desafíos y problemas asociados con la gestión de inventarios donde cualquier error durante este proceso podría generar pérdidas económicas significativas, haciendo que sea vital contar con herramientas que faciliten el manejo de productos a mediana y gran escala para garantizar el correcto funcionamiento de las empresas. Por esta razón, como equipo quisimos desarrollar una herramienta que permita facilitar este aspecto tan esencial en una gran cantidad de empresas, con la expectativa de poder generar un impacto positivo dentro de estas.

El sistema deberá ofrecer una interfaz intuitiva, enfocada en accesibilidad y usabilidad. Además, este sistema deberá contar con los siguientes conjuntos de funcionalidades:

1. Gestion de usuarios

El sistema debe poder gestionar usuarios los cuales pueden tener dos roles distintos:

- **Usuario:** Persona trabajadora de la empresa la cual tiene acceso a solo algunas de las funcionalidades del sistema.
- **Administrador:** Persona principal encargada de la gestión del inventario la cual tendrá acceso a todas las funcionalidades que tienen los usuarios, además de algunas funcionalidades extra.

Para acceder al sistema será necesario registrarse previamente dentro de la aplicación. Durante este proceso se le pedirá a la persona registrar la siguiente información:

- a. Nombre
- b. Correo electrónico
- c. Contraseña
- d. Rol (Usuario o administrador)
- e. Nombre de la organización

Al escoger registrarse como administrador, el usuario debe crear la organización para la cual se realizará la gestión del inventario, mientras que, si se escoge el rol de usuario, la persona deberá escoger entre las organizaciones que se encuentran actualmente registradas dentro del sistema.

Una vez registrado, el usuario podrá acceder al sistema usando la contraseña y correo electrónico registrados para posteriormente poder hacer uso de las funcionalidades que tenga habilitadas para su rol, incluyendo un servicio de modificación de datos personales en caso de ser necesario. Además, en caso de que el usuario haya olvidado su contraseña, podrá recuperarla haciendo uso del correo electrónico que registró.

2. Gestión de inventarios

El sistema debe permitir a usuarios y administradores visualizar una lista con todos los productos del inventario de la empresa, mostrando de forma clara la información principal de cada producto. Sin embargo, solo los administradores tendrán la capacidad de registrar, modificar y eliminar productos del inventario. Para registrar un nuevo producto, será necesario completar la siguiente información:

- a. Nombre.
- b. Categoría.
- c. Cantidad.
- d. Precio por unidad.
- e. Stock mínimo del producto.

Además, el administrador también tendrá la posibilidad de visualizar el historial de precios que ha tenido cualquier producto de interés.

3. Control de stock bajo

Dentro de la aplicación los administradores deben poder visualizar de manera sencilla los productos que se encuentran por debajo del stock mínimo definido durante el proceso de registro del producto.

4. Registro de movimientos (entradas y salidas)

La aplicación debe permitir que tanto usuarios como administradores puedan registrar salidas y entradas de productos. Para realizar este registro se deberá completar la siguiente información:

- a. Tipo de movimiento (entrada o salida).
- b. Cantidad de producto que sale o entra.
- c. Fecha de realización del movimiento.
- d. Descripción.

Además, tanto los usuarios como los administradores deberán poder visualizar un historial de los movimientos asociados a cualquier producto de interés.

5. Reportes

Los administradores podrán generar reportes donde se podrá ver la cantidad de productos actuales dentro del inventario. Además, el administrador podrá tener la posibilidad de exportar este informe en formato PDF, seleccionado, si desea, columnas específicas.

Respecto al diseño y estética de la aplicación web, se definen los siguientes colores para ser usados al momento de la creación de la interfaz:

Colores primarios

- Azul oscuro (#007BFF) para encabezados, botones principales, y elementos destacados.
- Blanco (FFFFFF) para fondo principal.

Colores secundarios

- Amarillo Mostaza (#ECC94B) para alertas de stock bajo o advertencias menores.

- Gris Claro (#EDF2F7) para fondos secundarios, tarjetas y elementos separados.
- Negro (#1A202C) para textos.

Adicionalmente, se usará la tipografía Verdana en el desarrollo de todas las interfaces debido a su simpleza y fácil lectura.

Por último, con la realización de este proyecto se espera poder tener un gran proceso de aprendizaje no solo de la parte técnica del desarrollo de software, sino también de la parte documental del desarrollo, permitiendo entender de manera más amplia lo que implica desarrollar software de manera profesional.

Punto 3: Análisis de requerimientos

A partir del levantamiento de requerimientos, se define la siguiente lista de funcionalidades junto con los roles que podrán acceder a cada una:

Módulo de registro y autenticación de usuarios

1. Registro de usuarios (administrador y usuario).
2. Inicio de sesión (administrador y usuario).
3. Modificación de datos personales (administrador y usuario).
4. Recuperación de contraseña usando el correo electrónico (administrador y usuario).

Módulo de gestión de inventarios

1. Visualización de inventarios (administrador y usuario).
2. Registro, modificación y eliminación de productos (administrador).
3. Visualización del historial de precios (administrador).

Módulo de control de stock bajo

1. Visualización de productos con stock bajo (administrador).

Módulo de registro de movimientos (entradas y salidas)

1. Visualización del historial de movimientos (administrador y usuario).
2. Registro de entradas y salidas de productos (usuario).

Módulo de reportes

1. Generación de informes del inventario actual (administrador).
2. Exportación de informes en formato PDF (administrador).
3. Personalización de campos en Informes (administrador).

Teniendo en cuenta esta lista de funcionalidades, se realiza el siguiente análisis usando el método **MoSCoW** y la secuencia de Fibonacci para asignar los días de trabajo:

Funcionalidad	Prioridad	Estimación (días)	Justificación
Registro de usuarios	Must have	3 días	Esta funcionalidad es crucial para el acceso al sistema, por lo cual es una prioridad dentro del sistema. Además, este proceso de registro no

			debería tomar mucho tiempo en su desarrollo, por lo cual 3 días es suficiente para su construcción.
Inicio de sesión	Must have	3 días	Al igual que la funcionalidad de registro, el inicio de sesión es crucial para que las personas puedan usar el sistema, volviéndolo una prioridad en la construcción del sistema. Se espera que el proceso de construcción de esta funcionalidad tome 3 días debido a su simpleza.
Modificación de datos personales	Should have	3 días	Aunque es una función muy útil para las personas, esta no implica una grave afectación en el uso de las funcionalidades principales del sistema, volviéndola una funcionalidad de menor prioridad. Por otro lado, se estiman 3 días para su construcción debido a que esta funcionalidad solo implica realizar modificaciones sencillas en la base de datos.
Recuperación de contraseña	Must have	5 días	Es habitual que los usuarios pierdan sus contraseñas o las olviden, implicando que estos no puedan acceder al sistema. Es por esto que consideramos que esta funcionalidad debe tener una alta prioridad. Respecto al tiempo estimado para su desarrollo, se asignaron 5 días de trabajo debido a que esta funcionalidad implica la creación de un correo electrónico que envíe los mensajes a los usuarios, haciendo que sea esencial la correcta configuración de este, lo cual podría tomar un poco más de tiempo debido a nuestra inexperiencia.
Visualización de inventarios	Must have	5 días	Esta funcionalidad es parte fundamental del sistema ya que tiene que ver directamente con la consulta de todos los productos que estén registrados dentro de la empresa. Respecto al tiempo estimado, se asignaron 5 días ya que, aunque esta funcionalidad implica realizar consultas sencillas a la base de datos, es necesario definir cómo esta información va a ser mostrada al usuario, lo cual podría tomar un poco más de tiempo
Registro, modificación y eliminación de productos	Must have	8 días	Al igual que la visualización de inventarios, esta funcionalidad tiene que ver directamente con el objetivo de este sistema (la gestión de inventarios), siendo esencial su construcción. Para esta funcionalidad se asignan 8 días ya que esta se tienen que realizar distintos tipos de acciones dentro de la base de datos, lo cual podría tomar una cantidad considerable de tiempo.
Visualización del historial de precios	Should have	3 días	Aunque esta funcionalidad puede ayudar a ver tendencias de precios dentro de las empresas, su implementación no es tan prioritaria ya que se podría considerar como una funcionalidad secundaria de la gestión de inventarios. Respecto al tiempo estimado, se le asigna un tiempo de 3 días ya que solo habría que realizar consultas a la base de datos, siendo más importante la interfaz

			que se cree alrededor de esta funcionalidad y la forma de mostrar estos datos
Visualización de productos con stock bajo	Must have	3 días	Es una funcionalidad muy importante ya que ayuda a prevenir desabastecimiento, lo cual es muy importante al interior de una empresa. Por otro lado, se asigna un tiempo estimado de 5 días para su construcción ya que esta funcionalidad únicamente se basa en buscar en la base de datos los productos cuyo stock está por debajo del valor mínimo registrado
Visualización del historial de movimientos	Must have	3 días	Esta es una funcionalidad muy importante para poder llevar un mejor trazabilidad del inventario, por lo cual es obligatoria la creación de esta funcionalidad. Al igual que para el historial de precios, se estima que en 3 días se puede crear esta funcionalidad ya que la complejidad de la creación de esta funcionalidad está únicamente relacionada a la forma en la que se mostrarán los datos.
Registro de entradas y salidas de productos	Must have	5 días	El registro de entradas y salidas de productos es una funcionalidad esencial para garantizar un control preciso del inventario, prevenir errores y servir como base para otras funciones críticas del sistema. Su implementación es obligatoria, ya que sin ella no sería posible gestionar correctamente el stock. Se estima un tiempo de desarrollo de 5 días, considerando la implementación de las búsquedas dentro de la base de datos y la verificación del correcto desempeño de esta funcionalidad
Generación de informes del inventario	Must have	5 días	La generación de informes de inventario es una funcionalidad importante ya que mejora la toma de decisiones y optimiza la gestión del stock. Se estima un tiempo de 5 días para su realización ya que es necesario determinar la mejor forma de mostrar esta gran cantidad de información, lo cual podría llevar un tiempo considerable
Exportación de informes del inventario en formato PDF	Should have	3 días	La exportación de informes en PDF es una funcionalidad importante, pero no esencial para el funcionamiento inicial del sistema ya que los administradores podrán acceder a los informes directamente en el sistema. Se estima un tiempo de desarrollo de 3 días con el fin de tener tiempo de aprender sobre reportlab (herramienta para la generación de PDF's) y la integración con el frontend
Personalización de campos en informes	Should have	3 días	Mejora la usabilidad pero no es obligatoria: Permite a los administradores filtrar datos irrelevantes y enfocarse en métricas clave (ej: cantidad y stock mínimo). El tiempo estimado cubre la

			implementación de checkboxes en el frontend, ajustes en las consultas del backend para filtrar campos dinámicamente, y pruebas de todas las combinaciones posibles.
--	--	--	---

Punto 4: Análisis gestión de software

Triada de la gestión de proyectos

- **Alcance**

Para la construcción del MVP de este proyecto se tendrá en cuenta el desarrollo de las funcionalidades clasificadas como Must Have dentro del análisis de requerimientos. Estas funcionalidades son las siguientes:

1. Registro de usuarios.
2. Inicio de sesión.
3. Recuperación de contraseña.
4. Visualización de inventarios.
5. Registro, modificación y eliminación de productos.
6. Visualización de productos con stock bajo.
7. Visualización del historial de movimientos de productos.
8. Registro de entradas y salidas de productos.
9. Generación de informes.

Con la implementación de estas funcionalidades podemos cubrir los principales objetivos del sistema relacionados con la gestión de inventarios y el correcto manejo de usuarios, permitiéndonos tener una versión funcional y útil del software que pueda servir como base para un posterior desarrollo de las funcionalidades clasificadas como Should Have dentro del análisis de requerimientos.

- **Tiempo**

Para el desarrollo de este proyecto se tendrán 4 desarrolladores junior los cuales se repartirán las 9 funcionalidades descritas para el MVP. Esto hará que algunos de los desarrolladores tengan más trabajo que otros, sin embargo, se estima que la construcción del MVP tardará aproximadamente 13 días.

- **Costo**

Para el desarrollo de la aplicación únicamente se utilizaran herramientas gratuitas, siendo el único gasto importante el asociado a los salarios de los 4 desarrolladores Junior. Según talent.com ([link](#)), el sueldo diario de un desarrollador junior es \$115.385 COP. Teniendo esto en cuenta, podemos realizar el siguiente cálculo para ver el gasto aproximado que se realizará para la construcción del sistema:

*Gasto total = Número de desarrolladores junior * días de trabajo * sueldo diario*

$$4 * 13 * \$115.385 = \$6.000.020 \text{ COP}$$

En total, se estima que el costo de desarrollo de la aplicación web será de aproximadamente \$6.000.020 COP.

Punto 8: Diseño y arquitectura

Arquitectura del sistema

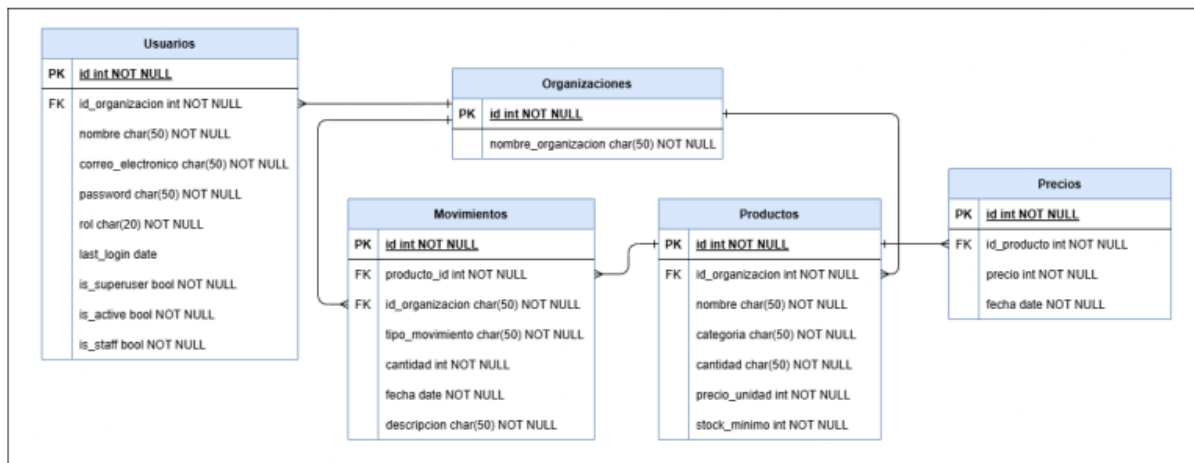
Este sistema utiliza una arquitectura cliente-servidor, donde el backend está desarrollado con Django REST Framework (DRF) y el frontend con Vue.js. Además, para la gestión de la base de datos se emplea SQLite.

En esta arquitectura, Vue.js (cliente) es responsable de generar la interfaz de usuario, permitiendo la interacción del usuario y la visualización de la información. Para obtener los datos, Vue.js realiza peticiones al backend (Django REST Framework), el cual procesa las solicitudes, accede a la base de datos y envía la información al frontend.



La principal razón por la que se escogió esta estructura es debido a su simplicidad de manejo, ya que al tener la parte lógica y la interfaz separadas es más cómodo e intuitivo trabajar en cada funcionalidad, permitiendo incluso realizar modificación de manera muy sencilla. Además, ya que el proyecto a realizar es un gestor de inventarios, la comunicación entre el servidor y la base de datos hace más sencilla la obtención de la distinta información que se necesite mostrar al usuario.

Diseño de la base de datos



La base de datos relacional cuenta con 5 tablas donde cada una tiene asociada una identificación la cual será su llave primaria. Estas identificaciones serán generadas de manera automática por django cada vez que se genere un nuevo registro en cada una de las tablas. Dentro de este modelo de base de datos la principales tablas son:

- **Usuarios:** Aquí se guarda toda la información relacionada con el usuario incluyendo un llave foránea hacia la organización con la que está relacionada el usuario. Como se puede ver, además de contener los campos requeridos en el levantamiento de requerimientos también incluye otros los cuales son requeridos por django para el manejo de usuarios.
- **Organizaciones:** Aquí se almacenan todas las organizaciones.

- **Productos:** Aquí se almacena toda la información de los productos que se definió durante el levantamiento de requerimientos además de incluir una llave foránea hacia la organización a la que pertenece.

Adicionalmente se encuentran las tablas **precios** y **movimientos**. La tabla movimientos es la encargada de registrar las salidas y entradas de productos en el inventario incluyendo la fecha en la que fueron realizados estos movimientos. Esta tabla es fundamental para poder construir la funcionalidad de visualización del historial de movimientos. Por otro lado, la tabla de precios se encarga de registrar todos los cambios que se realicen en el precio de los productos junto con la fecha en la que se realizan esos cambios. Esta tabla es importante para la visualización del historial de precios. Ambas tablas tienen una llave foránea hacia la tabla productos con el fin de poder identificar qué producto sufrió un cambio en su precio o estuvo involucrado en un movimiento.

La razón por la que se escogió una base de datos relacional es que gracias a estas asociaciones entre las distintas entidades se hace más fácil realizar la construcción de funcionalidades como la visualización del historial de precios y el historial de movimientos o para productos en específico la visualización del inventario. Además, la base de datos relacional construida también es útil durante el proceso de registro para verificar que la organización a la que quieren asociarse los usuarios realmente existe.

Punto 9: Patrones de diseño

Al estar usando django y Vue estamos parcialmente condicionados a usar los patrones de diseño definidos dentro de estos frameworks. Para el caso de django, estaremos usando el patrón de diseño MTV (Model-Template-View), el cual es una variación o reinterpretación del conocido patrón de diseño MVC (Model-View-Controller). En el MTV, la vista (View) define qué dato se muestra, pero no como se ve (principal diferencia respecto al tradicional MVC), siendo la plantilla (Template) quien se encarga de mostrar el dato, aunque en este proyecto es realmente Vuejs (frontend) quien se encarga de mostrar los datos. Por otro lado, el modelo (Model) dentro del MTV sigue teniendo el mismo propósito definido en MVC. Teniendo todo esto en cuenta, podemos afirmar que MTV resuelve problemas muy similares a los que resuelve MVC, siendo los siguientes algunos de los más importantes:

- Independencia entre las distintas partes del código, lo cual facilita el desarrollo.
- Posibilidad de reusar código.

Respecto a Vuejs, este usa un patrón de diseño llamado MVVM (Model-View-ViewModel) donde el ViewModel se encarga de establecer una conexión entre Model (datos) y View (interfaz gráfica), siendo su principal beneficio la reactividad de los datos.