Servicio Nacional de Aprendizaje (SENA)

Programa: Tecnólogo en Análisis y Desarrollo de Software

Ficha Número: 2775029

Investigadores: Sebastián Urrego y Duber muñoz

Instructor: Abdul

Fecha: 22/10/24

Presentación del caso práctico

-El proyecto tiene como objetivo desarrollar un sistema adaptable para optimizar la gestión de inventarios en pequeñas y medianas empresas (PYMEs). Este sistema se centrará en mejorar la precisión y la eficiencia, al tiempo que facilita su uso. Ofrecerá herramientas para mejorar las operaciones diarias y ayudará a las empresas a tomar decisiones informadas con datos actualizados

- Miembros del equipo : Sebastián Urrego y Duber muñoz

- Asignación de roles:

Sebastián Urrego Cárdenas- front end

Duber muñoz - back-end

Líder de proyecto: Sebastián Urrego

Desarrollador principal: Duber muñoz

Especialista en pruebas: Sebastián Urrego

Especialista en implementación: Duber muñoz

Documentador: Sebastián Urrego
Analizar requisitos dados

1. Requisitos Funcionales

Gestión de Productos:

Añadir, editar y eliminar productos del inventario.
Asignar categorías a los productos (alimentos, limpieza, tecnología, etc.). Actualización automática del stock tras cada compra o venta.
Posibilidad de adjuntar información adicional del producto (descripción, imagen, proveedor, etc.).

Control de Stock:

Control de niveles mínimos y máximos de stock.

Alertas automáticas cuando el stock alcanza un nivel bajo.

Generación de reportes de inventario (productos más vendidos, productos con bajo stock, etc.).

Gestión de Proveedores:

Registro y gestión de proveedores (contactos, productos suministrados). Historial de compras a proveedores.

Alertas para la reposición de productos según acuerdos con proveedores.

Gestión de Compras y Ventas:

Registro de entradas (compras) y salidas (ventas) del inventario.

Facturación de ventas.

Generación de informes de ventas y compras por períodos específicos. **Usuarios y Roles:**

Gestión de usuarios (administradores, empleados, etc.).

Definición de roles y permisos para cada tipo de usuario.

Auditoría de cambios (registro de qué usuario hizo qué acción en el sistema).

Informes y Estadísticas:

Generación de informes personalizados (ventas diarias, rendimiento de productos, etc.).

Visualización de gráficos con el comportamiento del inventario.

Exportación de informes a formatos como PDF o Excel.

2. Requisitos No Funcionales

Seguridad:

Autenticación y autorización de usuarios.

Encriptación de datos sensibles (contraseñas, datos de clientes, etc.). Políticas de acceso seguro y sesiones.

Usabilidad:

Interfaz amigable y fácil de usar para usuarios sin conocimientos técnicos.

Diseño responsive para acceso desde dispositivos móviles.

Documentación de ayuda dentro del sistema.

Escalabilidad:

Capacidad de soportar un número creciente de productos y usuarios sin degradación del rendimiento.

Modularidad para añadir nuevas funcionalidades en el futuro.

Rendimiento:

Respuesta rápida en consultas de inventario y generación de reportes. Capacidad de manejar grandes volúmenes de datos sin afectar la eficiencia.

Compatibilidad e Integración:

Capacidad para integrarse con otros sistemas (contabilidad,

facturación).

Soporte multi-navegador (Chrome, Firefox, Edge).

Backup y Recuperación:

Copias de seguridad automáticas del inventario y datos críticos.

Mecanismo de recuperación ante fallos del sistema.

3. Requisitos de Negocio

Cumplimiento Legal:

Cumplir con las normativas locales sobre gestión de inventarios y facturación.

Mantener un registro adecuado para auditorías.

Soporte Multi-Tienda:

Posibilidad de gestionar inventarios de varias sucursales o tiendas desde una misma plataforma.

Transferencia de stock entre sucursales.

Escalabilidad de Licencias:

El sistema debe poder ofrecer diferentes planes según las necesidades del cliente (pequeñas tiendas vs. cadenas más grandes).

Crear un diseño básico:

El proyecto esta en el siguiente repositorio de github **Desarrollar plan de trabajo**

Fase 1: Análisis y Planificación

Duración: 2 semanas (ambos)

Tareas para ambos:Reuniones iniciales para discutir los requisitos del sistema. Definición clara de los objetivos del proyecto.

Documentar los requisitos funcionales y no funcionales.

Crear un cronograma detallado.

Dividir las responsabilidades en las siguientes fases del proyecto. Entregables:

Documento de requisitos.

Cronograma del proyecto.

Fase 2: Diseño del Sistema

Duración: 2-3 semanas

Persona 1 (Arquitectura y Backend):

Definir la arquitectura del sistema (Django, mysql).

Diseñar el modelo de base de datos.

Planificar la seguridad y escalabilidad.

Persona 2 (Frontend y UI/UX):

Crear los wireframes y prototipos de las pantallas

principales. Diseñar la experiencia de usuario (UX) y la

interfaz (UI).

Entregables:Diagrama de arquitectura y modelo de base de

datos. Prototipos de la interfaz.

Fase 3: Desarrollo del Backend

Duración: 8 semanas

Persona 1:Configurar el entorno de desarrollo en Django.

Implementar el modelo de base de datos y las migraciones.

Desarrollar la API para la gestión de productos, proveedores y

usuarios. Implementar la lógica del inventario, incluyendo control de

stock y alertas.

Persona 2 (Colaboración):Ayudar con la definición de las pruebas unitarias y pruebas de seguridad.

Entregables:API funcional para el backend.

Pruebas unitarias para el backend.

Fase 4: Desarrollo del Frontend

Duración: 8 semanas (paralelo al backend)

Persona 2:Implementar la interfaz de usuario en HTML, CSS y JavaScript (con Bootstrap).

Desarrollar la parte visual de las páginas para la gestión de productos, inventario, y usuarios.

Conectar el frontend con la API del backend.

Persona 1 (Colaboración):Apoyo en la integración del frontend con el

backend. Asegurar que las consultas a la API se ejecuten correctamente

desde el frontend. Entregables:Frontend funcional.

Frontend conectado con el backend.

Fase 5: Pruebas y Depuración

Duración: 4 semanas

Persona 1:Realizar pruebas unitarias y de integración para el

backend. Probar la seguridad del sistema.

Persona 2:Realizar pruebas de la interfaz de usuario (validación de formularios, alertas).

Recoger feedback de usuarios clave para mejorar la UX.

Entregables:

Pruebas completadas.

Corrección de errores.

Fase 6: Implementación y Despliegue

Duración: 2 semanas

Persona 1:

Configurar el entorno de producción (servidores, base de datos). Realizar la migración de datos (si aplica).

Persona 2:

Capacitar a los usuarios finales.

Preparar la documentación de usuario.

Entregables:

Sistema en producción.

Documentación para usuarios.

Fase 7: Mantenimiento y Actualizaciones

Duración: 8 semanas (últimas 2 semanas activas y luego

paralelamente) Ambos:Resolver errores y aplicar actualizaciones de

seguridad.

Monitorear el rendimiento del sistema y realizar ajustes según sea

necesario. Entregables:

Sistema mantenido y actualizado.

Riesgos Técnicos

1. Retrasos en el desarrollo por falta de experiencia en algunas áreas técnicas

Mitigación: Capacitarse en tecnologías clave antes de iniciar el desarrollo, aprovechando recursos en línea.Definir tareas de complejidad adecuada según las fortalezas de cada miembro del equipo.Reservar tiempo en el cronograma para investigación y pruebas.

2.Problemas de integración entre el frontend y backend

Mitigación:Realizar pruebas de integración periódicas, no dejar toda la integración para el final.
Crear un plan de pruebas claro que abarque la comunicación entre ambos sistemas. Usar una metodología ágil (Scrum) para ir entregando partes funcionales cada semana.

Fallas en la arquitectura del sistema (escalabilidad y rendimiento)
Mitigación:Diseñar con la escalabilidad en mente desde el inicio (optimizar la base de datos, evitar código ineficiente).

Realizar pruebas de carga antes del lanzamiento.
Monitorizar el sistema para detectar cuellos de botella tempranamente.

Descripción del Riesgo Principal

Conflictos en la fusión de ramas (merge conflicts)*: Cuando dos desarrolladores realizan cambios en los mismos archivos o funciones y luego intentan fusionar sus ramas, pueden surgir conflictos difíciles de resolver.*

Perdida de cambios importantes: Si no se gestionan adecuadamente las ramas o las fusiones, se pueden perder modificaciones importantes realizadas por uno de los desarrolladores.
Mal uso de la historia de commits: Hacer commits sin mensajes descriptivos o sin seguir una estructura clara puede dificultar la identificación de errores y rastrear cambios importantes.
Sobreescritura de trabajo: Hacer "force push" sin la debida precaución podría sobrescribir trabajo que otro desarrollador ha hecho.

Mitigación

1. **Establecer una Convención de Uso de Git**
   Definir una convención para los nombres de ramas (`feature/nombre feature`, `fix/bug-nombre`).
   Acordar un estándar para los mensajes de commit, por ejemplo:
   ```
   feat: añadir funcionalidad de gestión de productos
   fix: corregir error en la actualización del stock
   ```
2. **Uso de Ramas Separadas**

Cada desarrollador debe trabajar en su propia rama (por ejemplo, `desarrollo`, `feature/nueva-funcionalidad`, etc.).
Mantener una rama principal (`main` o `master`) protegida, que sólo acepte cambios a través de revisiones (pull requests).

3. Revisiones y Pull Requests

Antes de fusionar cambios, utilizar **pull requests** que permitan revisar el código de manera conjunta y detectar posibles conflictos.
Incluir revisiones de código (code reviews) donde ambos desarrolladores verifiquen la calidad del código antes de aceptar los cambios.

4. Realizar Fusiones Frecuentes

Evitar trabajar en ramas demasiado tiempo sin hacer fusiones.
Realizar **fusiones frecuentes** ayuda a reducir los conflictos.
Sincronizar diariamente o semanalmente para reducir la posibilidad de grandes conflictos de fusión al final de las fases.

5. Capacitación y Buenas Prácticas en Git

Asegurarse de que ambos desarrolladores estén familiarizados con las mejores prácticas de Git, como `git pull`, `git rebase`, `git merge`, y el uso adecuado de `git stash`.
Practicar la resolución de conflictos de manera efectiva para no perder tiempo en errores comunes.

6. Uso de Etiquetas y Versiones

Implementar un sistema de etiquetado (`tags`) para marcar versiones estables antes de realizar cambios mayores o experimentos.
Esto permite restaurar versiones anteriores si algo sale mal durante el desarrollo.

7. Backups y Copias de Seguridad

Hacer copias de seguridad frecuentes de los repositorios para asegurarse de que siempre haya una versión estable a la cual regresar en caso de fallos.

Objetivo: El objetivo de StockEase es crear una herramienta adaptable que optimice la gestión de inventarios, mejorando la precisión y eficiencia de las operaciones diarias. Con este sistema, las empresas podrán tomar decisiones informadas basadas en datos actualizados y optimizar recursos de forma automatizada.

Problema que Resuelve

“La gestión de inventarios en pequeñas y medianas empresas suele ser

ineficiente, lo que genera problemas como exceso de stock, falta de productos o errores en la toma de decisiones."

Desafíos actuales:

Falta de visibilidad en tiempo real del inventario.
Procesos manuales que consumen tiempo y son propensos a errores. Dificultad para optimizar el uso de recursos y predecir la demanda.

StockEase aborda estos problemas mediante un sistema que automatiza el control de inventarios, permitiendo a los negocios manejar de manera eficiente su stock.

3. Funcionalidades Clave

"Ahora, hablemos de las principales funcionalidades que ofrece StockEase para mejorar la gestión de inventarios."

Control automatizado del inventario: Los usuarios pueden ver en tiempo real las cantidades de productos disponibles, establecer alertas de stock bajo y evitar errores humanos.
Gestión eficiente de pedidos: El sistema sugiere reabastecimientos basados en datos históricos, lo que permite anticiparse a la demanda.
Informes y análisis: Los gerentes pueden generar informes detallados para analizar el rendimiento de productos y tomar decisiones basadas en datos.
Interfaz intuitiva: Está diseñado pensando en la facilidad de uso, para que cualquier usuario, sin conocimientos técnicos, pueda gestionar el inventario sin problemas.

Desarrollo del Código Base: Actualmente, estamos desarrollando el código de **StockEase** utilizando HTML y CSS. Para futuras fases del proyecto, planeamos incorporar frameworks que nos permitirán agilizar el desarrollo y mejorar la funcionalidad.

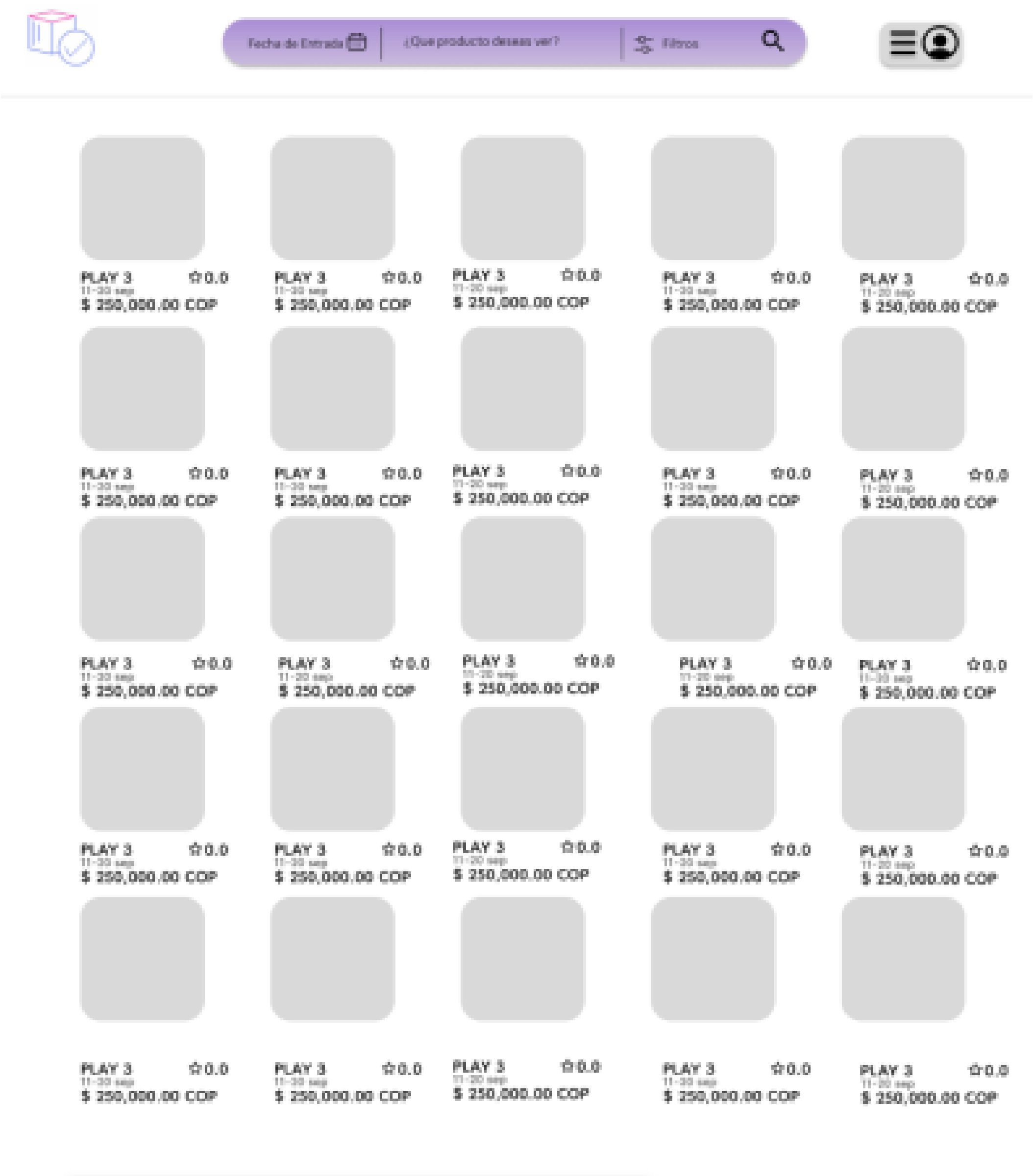
-Creación de interfaces básicas:

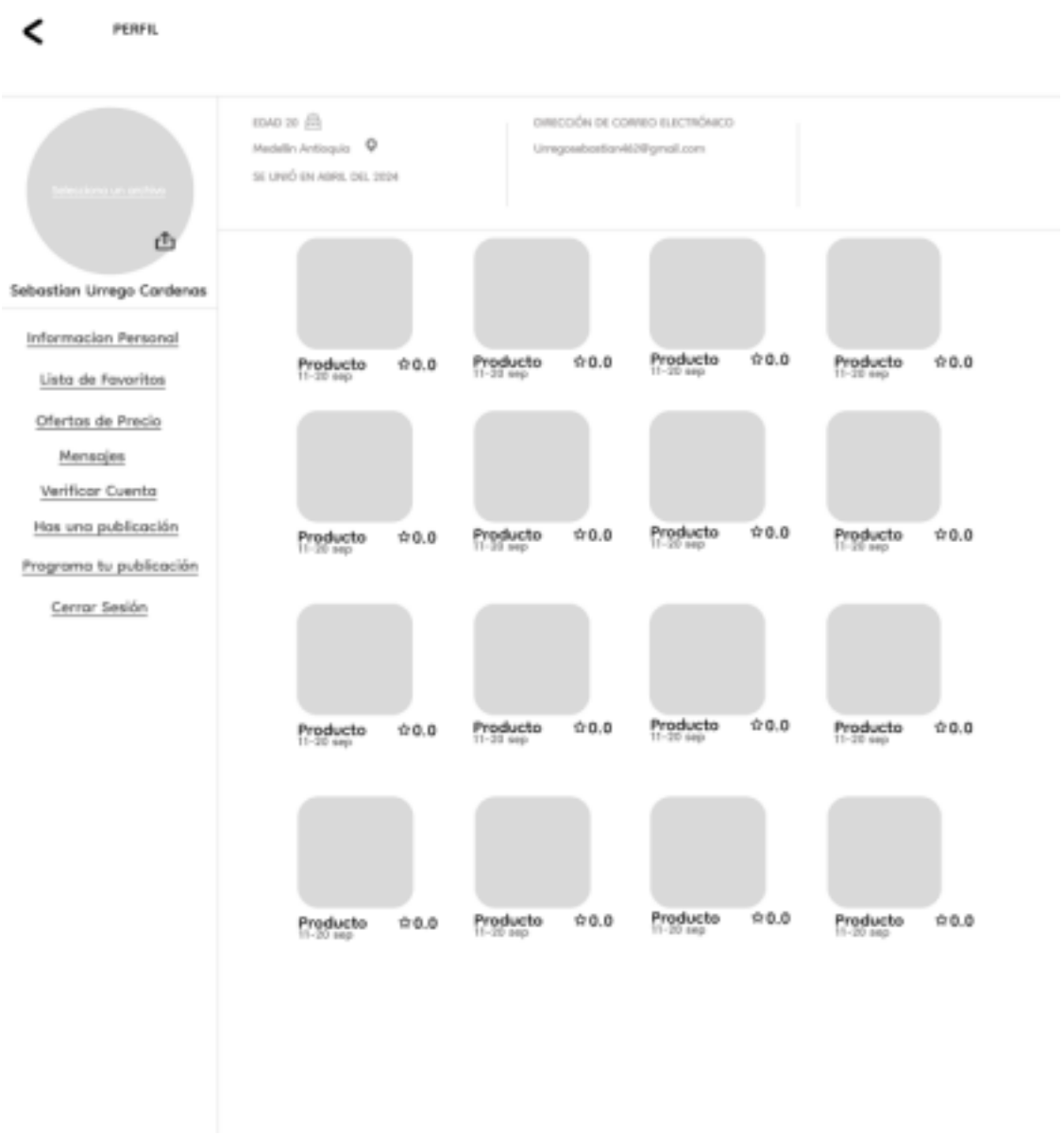
1-Home Inicio

2-pantalla de registeo

3-pantalla Ingreso

INICIO DE SESIÓN


Fecha de Entrada
¿Que producto deseas ver?
Filtros
PLAY 3
☆0.0
$ 250,000.00 COP
PLAY 3
☆0.0
$ 250,000.00 COP
PLAY 3
☆0.0
$ 250,000.00 COP
PLAY 3
☆0.0
$ 250,000.00 COP
PLAY 3
☆0.0
$ 250,000.00 COP
PLAY 3
☆0.0
$ 250,000.00 COP
PLAY 3
☆0.0
$ 250,000.00 COP
PLAY 3
☆0.0
$ 250,000.00 COP
PLAY 3
☆0.0
$ 250,000.00 COP
PLAY 3
☆0.0
$ 250,000.00 COP
PLAY 3
☆0.0
$ 250,000.00 COP
PLAY 3
☆0.0
$ 250,000.00 COP
PLAY 3
☆0.0
$ 250,000.00 COP
PLAY 3
☆0.0
$ 250,000.00 COP
PLAY 3
☆0.0
$ 250,000.00 COP
PLAY 3
☆0.0
$ 250,000.00 COP
PLAY 3
☆0.0
$ 250,000.00 COP
PLAY 3
☆0.0
$ 250,000.00 COP
PLAY 3
☆0.0
$ 250,000.00 COP
PLAY 3
☆0.0
$ 250,000.00 COP
PLAY 3
☆0.0
$ 250,000.00 COP
PLAY 3
☆0.0
$ 250,000.00 COP
PLAY 3
☆0.0
$ 250,000.00 COP
PLAY 3
☆0.0
$ 250,000.00 COP
PLAY 3
☆0.0
$ 250,000.00 COP



Implementación de funcionabilidad: [Github]

https://github.com/CaballeroUrrego/StockEase

- **Gestión del Inventario**: Crear formularios y lógica para añadir, editar y eliminar productos.
- **Alertas de Stock Bajo**: Implementar un sistema de notificaciones cuando el stock esté bajo.
- **Generación de Informes**: Proporcionar herramientas para que los usuarios generen informes sobre el inventario.
- **Control de Acceso**: Implementar roles y permisos para usuarios. •

Actualización en Tiempo Real: Asegurar que los datos del inventario se actualicen sin necesidad de recargar la página.

2. Documentación Paralela

Manual de usuario:

Manual de Usuario de

StockEase 1. Introducción

StockEase es una aplicación de gestión de inventarios diseñada para facilitar la búsqueda y gestión de productos en un entorno amigable y funcional. Este manual tiene como objetivo guiar a los usuarios a través de las diversas características y funcionalidades de la aplicación.

2. Requisitos del Sistema

• Navegador web moderno (Chrome, Firefox, Safari, Edge)

• Conexión a internet

3. Acceso a la Aplicación

1. Abre tu navegador web.
2. Ingresa la URL de **StockEase** en la barra de direcciones.
3. Espera a que se cargue la página de inicio.

4. Página de Inicio

La página de inicio de **StockEase** está diseñada con un encabezado, un área de búsqueda y secciones de productos destacados. A continuación se describen las secciones principales:

4.1 Encabezado

• **Logo**: El logo de la aplicación se encuentra en la parte superior izquierda. Al hacer clic en él, se regresa a la página de inicio.

• **Barra de Búsqueda**: Permite a los usuarios buscar productos ingresando texto en el campo de búsqueda.

• **Filtros**: Al hacer clic en el botón "Filtros", se despliega un menú donde los usuarios pueden seleccionar diferentes criterios de búsqueda, como tipo de producto, características, habitaciones y duración de la estancia.

4.2 Sección de Notificaciones

• **Notificaciones**: Icono de campana que muestra un menú desplegable con

las notificaciones recientes y enlaces a otras secciones como el centro de ayuda.

4.3 Perfil del Usuario

• **Menú de Perfil**: Al hacer clic en el icono de usuario, los usuarios pueden acceder a su perfil, iniciar sesión o cerrar sesión.

5. Navegación de Productos

Los productos se presentan en tarjetas que contienen la siguiente información:

• **Nombre del Producto**: Título del producto que se puede hacer clic para obtener más detalles.

• **Ubicación**: Indica la ubicación del producto.

• **Fechas Disponibles**: Muestra el rango de fechas en que el producto está disponible.

• **Precio**: Indica el precio del producto.

• **Calificación**: Muestra la calificación del producto.

5.1 Interacción con Productos

1. **Visualización de Detalles**: Haz clic en cualquier tarjeta de producto para ver más información sobre el mismo.
2. **Agregar al Carrito**: Si esta funcionalidad está implementada, los usuarios pueden agregar productos a su carrito desde la vista de detalles.

6. Filtros de Búsqueda

Los usuarios pueden refinar su búsqueda utilizando los filtros disponibles:

1. **Tipo de Producto**: Selecciona entre diferentes categorías como "Accesorios para Vehículos", "Animales y Mascotas", etc.
2. **Características**: Marcar casillas para especificar características deseadas. 3. **Habitaciones**: Selecciona el tamaño o tipo de habitación deseada.
4. **Cantidad de Baños**: Indica cuántos baños se requieren.
5. **Duración de Estancia**: Selecciona el tiempo que planeas usar el producto. **7. Soporte**

Si necesitas ayuda, accede al **Centro de Ayuda** desde la sección de notificaciones. Aquí podrás encontrar preguntas frecuentes y contactar al soporte técnico si es necesario.

8. Cierre de Sesión

Para cerrar tu sesión, ve al menú de perfil y selecciona "Cerrar Sesión". **Manual de instalación:** No requiere ya que será web

Plan de Implementación para StockEase

1. Objetivos del Proyecto

Desarrollar un sistema de gestión de inventarios que permita a los usuarios gestionar sus productos de manera eficiente.
Facilitar la búsqueda y filtrado de productos.
Proporcionar una interfaz de usuario amigable y accesible.

2. Alcance del Proyecto

Implementar funcionalidades básicas como el registro de productos, búsqueda, filtrado y gestión de stock.
Incluir un sistema de notificaciones y alertas.
Desarrollar un módulo de informes para analizar el rendimiento del inventario.

3. Cronograma del Proyecto

Mes 1:
Recolección de requisitos y diseño de la interfaz.
Establecimiento de la arquitectura del sistema.
Mes 2:
Desarrollo del backend (API, base de datos).
Configuración del entorno de desarrollo.
Mes 3:
Desarrollo del frontend (interfaz de usuario).
Integración con el backend.
Mes 4:
Pruebas de funcionalidad y corrección de errores.
Feedback de usuarios beta.

- **Mes 5:**

Implementación de mejoras basadas en feedback.
Preparación para el lanzamiento.

• Mes 6:

Lanzamiento oficial del software.
Entrenamiento para usuarios y soporte post-lanzamiento.

4. Equipo de Trabajo

Desarrolladores: 2 personas (tú y un compañero)
Diseñador UI/UX: (si es necesario)
Tester: (puede ser uno de los desarrolladores al inicio)

5. Tecnologías a Utilizar

• **Frontend:** HTML, CSS, JavaScript, posiblemente un framework como React o Vue.js
Base de Datos: PostgreSQL o MySQL.
Herramientas de Control de Versiones: Git y GitHub.

6. Gestión de Riesgos

Identificación de Riesgos:
Retrasos en el desarrollo.
Problemas de integración entre el frontend y el backend.
Plan de Mitigación:
Reuniones semanales para seguimiento.
Pruebas frecuentes durante el desarrollo.

7. Pruebas y Validación

Pruebas unitarias para asegurar que cada módulo funcione correctamente.
Pruebas de integración para comprobar la interacción entre diferentes módulos.
Pruebas de usuario para validar la experiencia del usuario final.

1. Actividades de Preparación
- Pruebas de sistema
- Preparación del ambiente
- Verificación de requisitos