UNIVERSIDAD DA VINCI DE GUATEMALA

FACULTAD DE INGENIERÍA, INDUSTRIA Y TECNOLOGÍA

CARRERA: LICENCIATURA EN INGENIERÍA EN SISTEMAS

Sistema de Gestión de Inventarios – ElectroDistribuciones S.A

Ingeniería de Software II

Ashley Dayane Alfaro Aguilar

Carné 202301982

Byron Rodolfo Maldonado Palacios

Carné 202300076

Guatemala, 2025.

Contenido

Introducción

El presente documento describe el desarrollo del Sistema de Gestión de Inventarios (SGA) para la empresa ElectroDistribuciones S.A., construido como parte del Proyecto del Semestre. El sistema se diseñó con el objetivo de ofrecer una herramienta tecnológica ligera, segura y fácil de usar para la administración completa del inventario, permitiendo registrar productos, controlar existencias, auditar movimientos y generar reportes críticos para la toma de decisiones.

El SGA integra funcionalidades clave como el registro y edición de productos, la administración de entradas y salidas de inventario, la consulta del historial de movimientos, y reportes especializados como el reporte de stock bajo, que incluye alertas visuales para facilitar el monitoreo del estado del inventario.

La solución fue desarrollada utilizando Python para el backend, CSS para la interfaz gráfica, y SQL Server como motor de base de datos. Estos componentes permiten una arquitectura modular, escalable y con buenas prácticas de programación. Además, se implementó un sistema de autenticación por roles, permitiendo el acceso diferenciado para usuarios ADMIN, SUPERVISOR y OPERADOR, garantizando un control adecuado de privilegios según la responsabilidad del puesto.

ACTA DE CONSTITUCIÓN DE

ElectroDistribuciones S.A.

Nombre del Proyecto: Sistema de Gestión de Inventarios – ElectroDistribuciones S.A.

Lugar y Fecha: Guatemala, 22 de octubre de 2025

Hora: 20:00 p.m.

Integrantes del equipo:

NOMBRES	ROL EN EL PROYECTO	NUMERO DE IDENTIFICACION	CORREO ELECTRÓNICO
Byron Rodolfo Maldonado Palacios	Líder del proyecto	202300076	202300076@estudiante.udv.edu.gt
Ashley Dayane Alfaro Aguilar	Analista de sistemas	202301982	202301982@estudiante.udv.edu.gt
Byron Maldonado y Ashley Alfaro	Desarrollador Backend	202300076 - 202301982	202300076@estudiante.udv.edu.gt - 202301982@estudiante.udv.edu.gt

1. DESIGNACIÓN DE ROLES PRINCIPALES.

Gerente del proyecto: Byron Rodolfo Maldonado Palacios.

Analista de negocios: Ashley Dayane Alfaro Aguilar.

Desarrollador Principal: Byron Maldonado y Ashley Alfaro.

Encargado de pruebas: Byron Maldonado y Ashley Alfaro.

2. JUSTIFICACIÓN DEL PROYECTO

La empresa actualmente administra inventarios con hojas de cálculo, lo que provoca errores en registros, falta de trazabilidad, información desactualizada y riesgo de pérdida de datos. Un sistema moderno reducirá errores, habilitará decisiones en tiempo real y soportará el crecimiento de la empresa como parte de su estrategia de transformación digital.

3. ALCANCE Y OBJETIVOS GENERALES

Alcance:

- Construir una aplicación web responsive para gestionar entradas/salidas, devoluciones y ajustes.

- Generar reportes en tiempo real (stock, historial).
- Integrar con sistemas existentes (POS y compras).
- Implementar alertas de niveles críticos y sugerencias de reabastecimiento.
- Gestión de usuarios y roles (admins, operadores, supervisores).
- Cumplir requisitos no funcionales: seguridad, escalabilidad, usabilidad móvil y disponibilidad con respaldo.
- Entregar un MVP en 4 meses con hitos mensuales (diseño/prototipo; productos/movimientos; reportes/alertas; integración/pruebas).

Objetivo general: Diseñar e implementar un sistema de inventarios que disminuya errores operativos y brinde visibilidad en tiempo real para la toma de decisiones, logrando un MVP funcional en cuatro meses con integraciones esenciales y controles de seguridad alineados a las necesidades del negocio.

Los integrantes del equipo manifiestan su compromiso de desarrollar el Sistema de Gestión de Inventarios, con el fin de automatizar los procesos de control de existencias, entradas, salidas y reportes de productos en ElectroDistribuciones S.A., dentro del marco del proyecto semestral de la carrera de Ingeniería en Sistemas de la Universidad Da Vinci de Guatemala.

4. BREVE DESCRIPCIÓN DEL PRODUCTO FINAL

Aplicación web que permita registrar movimientos (compras, ventas, devoluciones), consultar reportes en tiempo real, configurar alertas inteligentes, administrar usuarios/roles y conectarse con POS y compras. Debe ser usable en escritorio y móvil, con autenticación, autorización y cifrado de datos sensibles**.**

5. PRESUPUESTO ESTIMADO

Presupuesto estimado (Q)		
Concepto	**Supuesto principal**	**Monto (Q)**
Desarrollo (2 devs × Q7,500 × 4 meses)	2 desarrolladores dedicados	60,000
Gestión/PM (0.5 FTE × Q10,000 × 4)	Gerencia de proyecto medio tiempo	20,000
UX/UI	Prototipos + pruebas de usabilidad	8,000
Infraestructura (hosting, dominios, CI)	4 meses ambientes dev/test/prod	4,000

Licencias y herramientas	Repositorios, monitoreo, utilidades	3,000
Capacitación y adopción	Talleres a usuarios (operativos y supervisores)	5,000
Subtotal		**100,000**
Contingencia (10%)	Riesgos técnicos/organizacionales	**10,000**
Total estimado		**110,000**

6. CRITERIOS DE ÉXITO

- MVP entregado en 4 meses con las funcionalidades definidas por hito mensual.
- Reducción de errores de registro ≥ 60% frente a la línea base en Excel.
- Tiempo de consulta de stock ≤ 3 segundos para los 50 productos más consultados.
- Trazabilidad completa de movimientos por producto (auditable).
- Disponibilidad ≥ 99% en horario operativo y respaldo automático verificado semanalmente.
- Adopción del usuario final: ≥ 80% del personal operativo capacitado y usando el sistema en la primera semana de salida a producción.
- Integración funcionando con POS y compras para flujos básicos (compra/venta).

7. SUPUESTOS Y RESTRICCIONES

- **Supuestos:** colaboración de áreas de ventas/compras; acceso a APIs/archivos de POS y compras; disponibilidad de usuarios clave para validaciones.
- **Restricciones:** presupuesto limitado; convivencia con proceso actual durante la transición; priorización estricta del alcance del MVP.

8. LECTURA Y APROBACIÓN DEL ACTA

Sometida a consideración de los constituyentes, la presente acta fue leída y aprobada y en constancia de todo lo anterior se firma por los integrantes de la reunión.

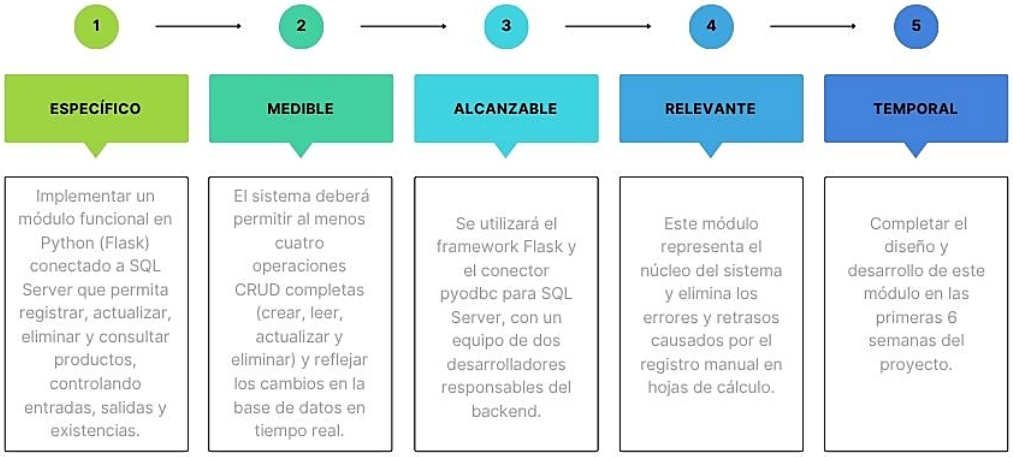
Byron Rodolfo Maldonado Palacios **Ashley Dayane Alfaro Aguilar**

C.C.  **C.C.** 

OBJETIVOS SMART

Objetivo SMART 1

Desarrollar el módulo de gestión de productos e inventarios



Objetivo SMART 2

Implementar una interfaz web moderna y responsiva

Objetivo SMART 3

Integrar reportes automáticos y controles de seguridad

Identificación de Stakeholders

Stakeholder	Rol en el proyecto	Expectativas principales	Nivel de influencia	Plan básico de comunicación
Gerente del Proyecto (Byron Maldonado)	Responsable de la planificación, control y supervisión del desarrollo.	Cumplir los plazos, mantener la calidad del sistema y asegurar la coordinación del equipo.	**Alta**	Reuniones semanales con el equipo, seguimiento diario vía grupo interno o Trello.
Analista de Sistemas (Ashley Alfaro)	Encargado del levantamiento de requerimientos, modelado de base de datos y pruebas técnicas.	Obtener información clara de los usuarios y garantizar la estabilidad de la base de datos SQL Server.	**Alta**	Comunicación continua con el gerente y usuarios para validar requerimientos.
Desarrollador Backend (Byron Y Ashley)	Implementa la lógica del sistema en Python, conecta con la base de datos y asegura la integridad de los datos.	Código funcional, eficiente y seguro; documentación técnica clara.	**Media**	Coordinación técnica con analista y revisiones de código semanales.
Diseñador Frontend (UI/UX) (Byron Y Ashely)	Crea las interfaces web con HTML, CSS y Bootstrap.	Interfaz moderna, intuitiva y adaptable a móviles.	**Media**	Comunicación diaria con el backend y revisiones quincenales con el gerente.
Tester / QA (Byron Maldonado)	Valida funcionalidades, pruebas unitarias y detección de errores.	Asegurar la calidad y funcionalidad del producto antes de su entrega.	**Media**	Reporte de errores mediante hoja de control compartida y reuniones de revisión.
Asesor Técnico (Ashley Alfaro)	Supervisa el avance, brinda asesoría técnica y valida entregas.	Cumplimiento de estándares académicos y técnicos del curso.	**Alta**	Entrega de reportes quincenales, revisiones presenciales o virtuales.
Usuarios finales (Empleados de ElectroDistribuciones S.A.)	Utilizarán el sistema para registrar inventario y generar reportes.	Plataforma fácil de usar, rápida y confiable para el control de existencias.	**Alta**	Encuestas de satisfacción y sesiones de capacitación al cierre del proyecto.
Dirección de la Empresa (Stakeholder institucional)	Cliente principal que aprueba el producto final.	Sistema funcional que mejore la eficiencia y reduzca errores del control manual.	**Muy alta**	Presentaciones mensuales de avance, informe final y demostración funcional.

Plan básico de comunicación				
Tipo de comunicación	**Frecuencia**	**Medio**	**Responsables**	**Propósito principal**
Reunión general del equipo	Semanal	Google Meet / WhatsApp	Gerente del Proyecto	Revisar avances y asignar tareas.
Informe de progreso técnico	Quincenal	Documento compartido (Drive / Trello)	Analista / Desarrollador	Evaluar estado del desarrollo y base de datos.
Revisión con asesor técnico	Cada 2 semanas	Reunión virtual / Presencial	Gerente / Asesor Técnico	Obtener retroalimentación académica.
Actualización al cliente (empresa)	Mensual	Presentación / Correo institucional	Gerente del Proyecto	Mostrar resultados parciales y validar requerimientos.
Capacitación a usuarios finales	Fase final	Sesión presencial o virtual	Analista / Tester	Garantizar la adopción del sistema.

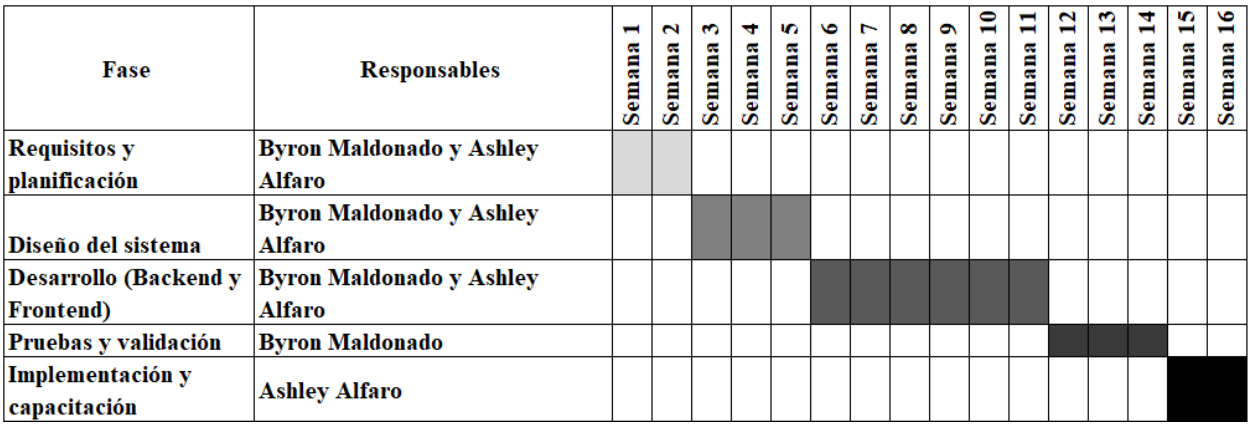
Organigrama del proyecto

Diagrama de Gantt

Duración total estimada: 16 semanas

Periodo del proyecto: Del 28 de octubre de 2025 al 16 de febrero de 2026

Fase	Duración estimada	Fecha de inicio	Fecha de fin	Dependencias	Actividades principales
1. Requisitos y planificación	2 semanas	28-oct-25	10-nov-25	—	Reuniones con stakeholders, levantamiento de requerimientos, definición de alcance y riesgos.
2. Diseño del sistema	3 semanas	11-nov-25	1-dic-25	Fase 1	Creación de diagramas UML, estructura de base de datos SQL Server, diseño de interfaz y arquitectura.
3. Desarrollo (Backend y Frontend)	6 semanas	2-dic-25	13-ene-26	Fase 2	Implementación en Python (Flask), conexión con SQL Server, diseño con CSS y Bootstrap, pruebas parciales.
4. Pruebas y validación	3 semanas	14-ene-26	3-feb-26	Fase 3	Pruebas unitarias, de integración y aceptación. Corrección de errores y ajustes de interfaz.
5. Implementación y capacitación	2 semanas	4-feb-26	16-feb-26	Fase 4	Instalación del sistema, documentación técnica y manual de usuario, capacitación al cliente.



Triple restricción

1. **Gestión del tiempo**
   - Se utilizará un cronograma detallado con responsables, controlado semanalmente.
   - Las actividades se dividirán en entregables cortos basados en avances técnicos (backend, frontend, base de datos).
   - Se aplicará una metodología ágil que permita ajustar prioridades cada sprint.

Herramientas: Trello, GitHub Projects, reuniones semanales de control.

2. **Gestión de alcance**
   - El alcance está basado en los requisitos funcionales y no funcionales anteriormente (registro, reportes, roles, alertas, integración, etc.).
   - Cualquier funcionalidad adicional será registrada como Solicitud de Cambio (RFC).
   - El objetivo es entregar un MVP funcional en 4 meses, por lo que el alcance estará estrictamente protegido.
3. **Gestión del costo**

Aunque el contexto académico no utiliza presupuesto real, este proyecto contempla:

- Horas de dedicación del equipo.
- Recursos tecnológicos (servidor SQL, hosting local, herramientas de desarrollo).
- Posible uso de bibliotecas externas.

Para gestionarlo:

- Se priorizan herramientas gratuitas y open source.
- Se evita el sobre-desarrollo fuera del alcance.
- Se monitorean horas invertidas por integrante.

Por ejemplo: La empresa pide que el sistema incluya un módulo de lectura de códigos QR para el ingreso rápido de inventario.

Impacto:

En el Alcance:

- Se agrega una nueva funcionalidad no incluida originalmente.
- Habría que diseñar interfaz, lógica en Python y pruebas adicionales.

En el Tiempo:

A. Esto podría agregar entre 1 a 2 semanas de trabajo al sprint de desarrollo.

B. Retrasa tareas ya planificadas como los reportes o las alertas.

En el Costo:

- Requiere más horas del equipo.
- Puede necesitar instalar librerías nuevas o integrar hardware adicional.

Metodología Ágil – Planificación del Sprint

Este sprint simula un Sprint de 2 semanas (14 días), correspondiente a la fase inicial del desarrollo del MVP.

A. User Stories (5 historias de usuario)

Construidas bajo el formato: Como [rol], quiero [función], para [beneficio].

1. HU-01 — Registrar productos

Como administrador, quiero registrar nuevos productos con datos completos en el sistema, para mantener actualizado el inventario y evitar errores manuales.

2. HU-02 — Consultar niveles de stock

Como operador, quiero visualizar el stock actualizado por producto, para tomar decisiones rápidas sobre ventas y reabastecimiento.

3. HU-03 — Registrar movimientos

Como operador, quiero ingresar entradas y salidas de inventario, para mantener un control preciso del flujo de productos.

4. HU-04 — Alertas de stock bajo

Como supervisor, quiero recibir alertas cuando un producto llegue a su mínimo, para iniciar procesos de reabastecimiento.

5. HU-05 — Autenticación y roles

Como usuario del sistema, quiero iniciar sesión según mi rol, para acceder únicamente a las funcionalidades autorizadas.

B. Sprint Backlog – Planificación del Sprint (2 semanas)

Duración del sprint: 14 días

Equipo: Backend, Frontend, Analista y Tester.

SPRINT BACKLOG – Tabla de tareas

User Story	Tareas del Sprint	Responsable	Duración estimada
HU-01 Registrar productos	• Crear tabla Productos en SQL Server	Desarrollador Backend + Frontend	3 días
	• Programar endpoint en Python (Flask)		
	• Crear formulario HTML/CSS		
HU-02 Consultar stock	• Crear consulta SQL (SELECT + filtros)	Backend + Frontend	2 días
	• Crear interfaz de consulta		
	• Mostrar resultados con tabla dinámica		
HU-03 Registrar movimientos	• Crear tabla Movimientos	Backend	3 días
	• Programar CRUD de movimientos		
	• Validar actualización de inventario		
HU-04 Alertas	• Configurar trigger o SP para niveles mínimos	Backend	3 días
	• Implementar lógica en Python		
	• Crear notificación visual		
HU-05 Autenticación	• Crear tabla Usuarios y Roles	Backend + Frontend	3 días
	• Implementar login (Python)		
	• Proteger rutas según rol		

Total, estimado: 14 días — Sprint completo

C. Reunión de Revisión del Sprint

Al finalizar el sprint se realiza la Sprint Review, donde:

El equipo presenta:

- CRUD de productos funcionando
- Vista de stock y reportes iniciales
- Movimientos y actualización de inventario
- Alertas básicas y Login con roles

Participan:

- Gerente del Proyecto (Ashley)

- Analista (Byron)
- Docente/Asesor técnico y Todo el equipo de desarrollo

Objetivo de la revisión:

- Validar que lo entregado cumple las User Stories
- Obtener retroalimentación
- Ver si se deben agregar cambios al Product Backlog

D. Reunión de Retrospectiva del Sprint

1. **¿Qué salió bien?**
   - Comunicación fluida entre Backend y Frontend
   - Conexión exitosa con SQL Server
   - Se cumplió el sprint completo sin retrasos
2. **¿Qué se puede mejorar?**
   - Documentar mejor la API y Optimizar consultas SQL pesadas
   - Mejor uso del control de versiones.
3. **¿Qué acciones tomamos para el siguiente sprint?**
   - Aplicar Gitflow de manera más estricta
   - Añadir pruebas unitarias desde el inicio
   - Priorizar tareas complejas en los primeros días del sprint

Gestión de cambios

A. Proceso de Aprobación de Cambios

A continuación, se detalla el procedimiento estándar que utilizará el equipo:

1. Solicitud del Cambio (RFC – Request For Change)

Cualquier stakeholder (clientes, usuarios, docente asesor o equipo técnico) puede solicitar un cambio.
La solicitud debe incluir:

- Descripción del cambio
- Razón o necesidad del cambio
- Impacto esperado en el sistema
- Urgencia o prioridad
- Stakeholder que lo solicita

Formato básico (RFC):

- Código: RFC-001
- Solicitante: Usuario / Supervisor / Docente / Equipo
- Fecha de solicitud
- Descripción del cambio
- Justificación
- Módulos afectados

2. Análisis del Cambio

El equipo técnico evalúa:

- Impacto técnico:
  Si afecta la base de datos, backend, frontend o integración.
- Impacto funcional:
  Si cambia el comportamiento o agrega nuevas reglas.
- Impacto en la Triple Restricción (Tiempo – Costo – Alcance):

Elemento	Evaluación
Tiempo	¿Retrasa alguna fase del Gantt? ¿Requiere horas extra?
Costo	¿Demandará más recursos, herramientas o trabajo adicional?
Alcance	¿Amplía funcionalidades fuera del MVP?

3. Aprobación o Rechazo del Cambio

A cargo del:

- Gerente del Proyecto (Ashley)
- Analista de Sistemas (Byron)
- Docente Asesor (cuando el cambio afecta el MVP o los objetivos académicos)

Se clasifican en:

- Aprobado
- Aprobado con modificaciones
- Rechazado

4. Implementación del Cambio

Si se aprueba:

- Se incluye en el próximo Sprint Backlog
- Se actualiza el Diagrama de Gantt
- Se asigna un responsable (backend, frontend o analista)
- Se actualiza la documentación del sistema

5. Seguimiento y Validación

Una vez implementado:

- El Tester/QA realiza pruebas unitarias
- El analista valida que el cambio cumple el objetivo
- El equipo lo presenta en la Revisión de Sprint

B. Impacto del Cambio en la Triple Restricción (Ejemplo Específico)

Cambio solicitado:

La empresa desea agregar un módulo de auditoría de movimientos, que registre automáticamente quién realiza cada operación y cuando la realiza.

1. **Impacto en el tiempo**
    i. Se agregan aproximadamente 2 semanas adicionales al sprint.
    ii. Podría retrasar las pruebas y la implementación final.
2. Impacto en el costo
    i. Requiere mayor dedicación del equipo.
    ii. Implica más tiempo de pruebas y ajustes.
3. Impacto en el alcance
    i. No estaba incluido en el MVP original.
    ii. Requiere modificar la base de datos (añadir tabla Auditoría).
    iii. Aumenta funciones y complejidad del backend.

Conclusión del análisis del ejemplo:

El cambio solo puede aprobarse si:

- Se extiende el cronograma o
- Se reduce el alcance de otra funcionalidad o
- Se asignan más recursos (horas del equipo)

Esto mantiene el equilibrio del proyecto.

Gestión de riesgos

Los riesgos se clasifican en técnicos, humanos y organizacionales:

4. **Riesgos Técnicos**
   - **Problemas de conexión entre Python y SQL Server**

Probabilidad: Media

Impacto: Alto

Estrategia de mitigación:

- Probar la conexión desde el inicio del proyecto.
- Utilizar el controlador actualizado (pyodbc) y validar credenciales.
- Crear procedimientos almacenados para estandarizar consultas.

Plan de contingencia:

- Migrar temporalmente a una base SQLite local para avanzar el desarrollo.
- Rediseñar la capa de datos usando una clase abstracta (DAO) para permitir cambio rápido de motor.

5. **Riesgos humanos**
   - **Falta de comunicación entre los miembros del equipo**

Probabilidad: Media

Impacto: Medio

Estrategia de mitigación:

- Reuniones breves tres veces por semana (Scrum Diario).
- Uso obligatorio de Trello para asignar y actualizar tareas.
- Acordar canales oficiales (WhatsApp y GitHub).

Plan de contingencia:

- Reunión de emergencia dirigida por el Gerente del Proyecto.
- Redistribución temporal de tareas a otro integrante.

6. **Riesgos organizacionales**
   E. **Requerimientos cambiantes del "cliente"**

Probabilidad: Alta

Impacto: Alto

Estrategia de mitigación:

- Utilizar un documento de requerimientos firmado.
- Incluir todas las solicitudes de cambio en el proceso RFC.
- Actualizar backlog a final de cada sprint.

Plan de contingencia:

- Renegociar alcance para evitar afectar el MVP.
- Ajustar el plan del sprint o extender el cronograma según prioridad del cambio.

Tipo de riesgo	Riesgo	Probabilidad	Impacto	Mitigación	Contingencia
Técnico	Conexión Python–SQL Server	Media	Alto	Configuración correcta, pruebas tempranas	Cambiar temporalmente de motor
Humano	Falta de comunicación	Media	Medio	Reuniones, Trello, roles definidos	Redistribuir tareas
Organizacional	Cambios de requerimientos	Alta	Alto	Control RFC, sprints	Renegociación de alcance

Capacidad de reutilización del proyecto

El sistema de gestión de inventarios para ElectroDistribuciones S.A., desarrollado con Python (backend), HTML/CSS (frontend) y SQL Server (base de datos), se diseña pensando no solo en cubrir las necesidades actuales, sino también en ser reutilizable, escalable y adaptable a futuros requerimientos de la empresa u otros proyectos similares.

1. **Modularidad del código**

El proyecto se estructurará siguiendo una arquitectura en capas, similar a MVC (Modelo–Vista–Controlador):

- Modelo (Model):
    - Clases y funciones que gestionan el acceso a la base de datos (tablas de Productos, Movimientos, Usuarios, Roles, etc.).
    - Uso de módulos específicos para consultas SQL (por ejemplo, db_productos.py, db_movimientos.py).
- Controlador (Controller):
    - Rutas y lógica de negocio en Python (Flask), como /productos, /movimientos, /login.
    - Validación de datos, reglas de negocio (actualizar stock, validar rol del usuario, etc.).
- Vista (View):
    - Plantillas HTML reutilizables (layouts comunes) y archivos CSS para estilos.
    - Formularios reutilizados para altas, bajas, cambios, listas y filtros.

Ventajas de este modularidad:

- Permite reutilizar componentes (por ejemplo, un módulo de autenticación) en otros proyectos.
- Facilita el mantenimiento, ya que los cambios en una parte (vista, controlador o modelo) no obligan a reescribir todo.
- Posibilita que distintos miembros del equipo trabajen en paralelo sin interferirse.

2. **Uso de buenas prácticas de programación**

Para garantizar que el sistema pueda mantenerse y reutilizarse, se adoptarán las siguientes buenas prácticas:

- Control de versiones con Git y Gitflow

    - Ramas feature/, develop, main y hotfix/.
    - Facilita reutilizar partes del código y controlar versiones del sistema.
- Convenciones de código (PEP8 en Python)
    - Nombres claros de variables, clases y funciones.
    - Comentarios solo donde agreguen valor, evitando confusión.
- Documentación técnica básica:
    - Comentarios en funciones clave.
    - Archivo README describiendo módulos, requerimientos e instrucciones de ejecución.
- Separación de configuración:
    - Datos de conexión a SQL Server en un archivo de configuración (.env o config separado).
    - Esto permite reutilizar el proyecto en otros entornos (desarrollo, pruebas, producción) sin cambiar el código fuente.
- Pruebas básicas:
    - Pruebas unitarias y de integración para validar que los módulos sigan funcionando al hacer cambios o reutilizarlos.

Producto mínimo viable funcional (MVP)

El MVP incluirá como mínimo las siguientes funcionalidades:

1. **Gestión de usuarios y autenticación**
    a. Inicio de sesión con usuario y contraseña.
    b. Roles básicos: administrador y operador.
    c. Restricción de acceso según rol (por ejemplo, solo el administrador puede gestionar usuarios).
2. **Gestión de productos**
    a. Registro de productos (código, nombre, categoría, proveedor, precio, stock mínimo).
    b. Edición y eliminación de productos.
    c. Listado de productos con búsqueda y filtros básicos.
3. **Registro de movimientos de** inventario
    a. Registro de entradas (compras o reabastecimiento).
    b. Registro de salidas (ventas o consumo).
    c. Registro de devoluciones y ajustes.
    d. Actualización automática del stock en cada movimiento.
4. **Consulta de inventario**
    a. Pantalla de vista general de stock por producto.
    b. Filtros por categoría o estado (con stock bajo, sin existencias, etc.).
5. **Reportes básicos**
    a. Reporte de movimientos recientes (por rango de fechas).
    b. Reporte de productos con stock bajo (según stock mínimo).
    c. Exportación sencilla (por ejemplo, a tabla HTML imprimible).
6. **Alertas simples de stock bajo**
    a. Indicadores visuales (íconos o colores) para productos con stock por debajo del mínimo.
    b. Mensaje de alerta en la interfaz al iniciar sesión si existen productos críticos.

Demostración del MVP

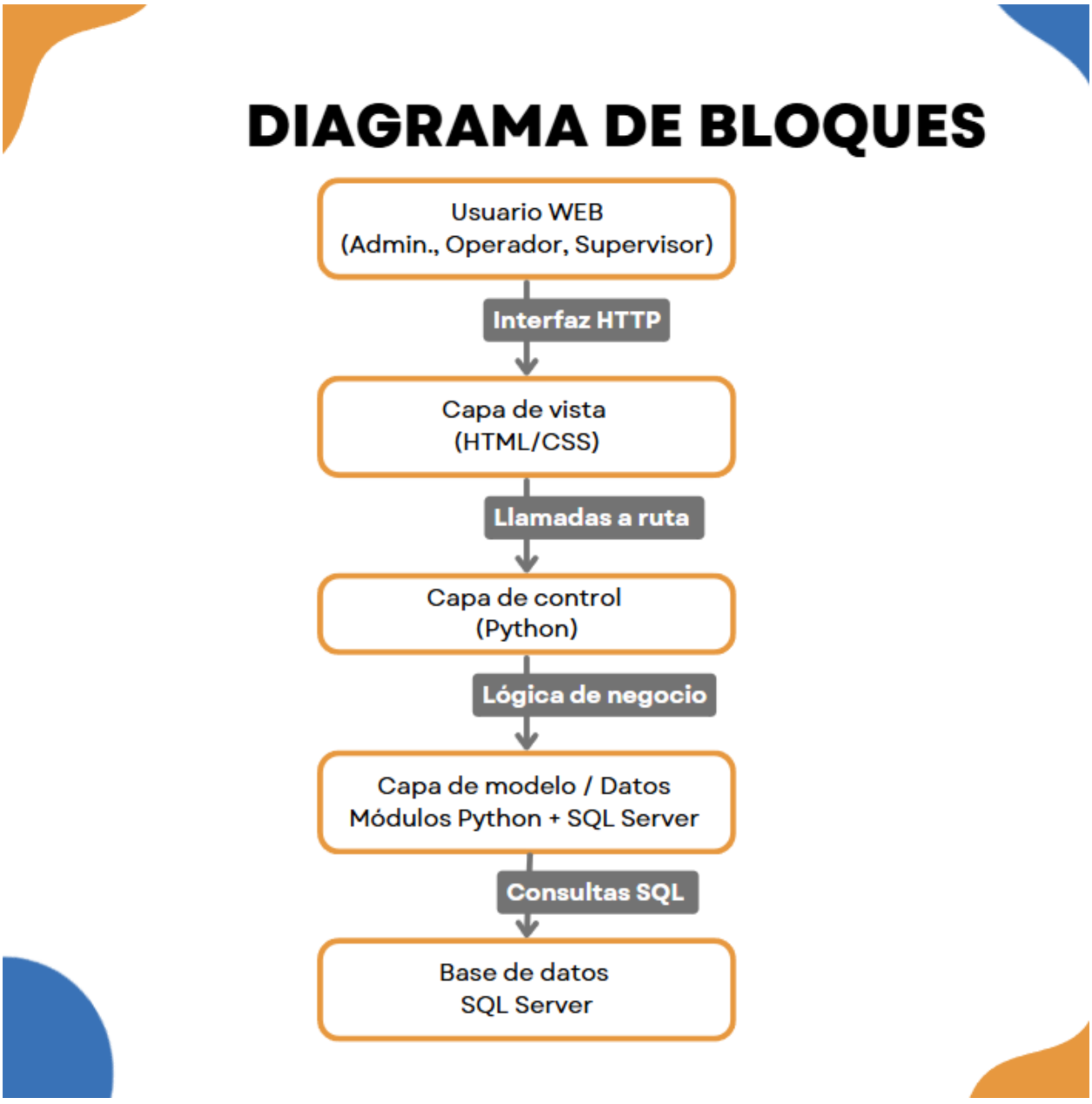
1. **Ingreso al sistema**
    - Inicio de sesión como usuario administrador y como operador.
    - Diferencias de menú según rol.
2. **Alta de productos**
    - Registro de un nuevo producto desde el formulario web.
    - Verificación de su aparición en el listado de inventario.
3. **Registro de movimientos**
    - Registrar una entrada de inventario.
    - Registrar una salida/venta del mismo producto.
    - Mostrar como se actualiza el stock en tiempo real.
4. **Visualización de reportes**
    - Consultar el reporte de movimientos.

- Mostrar el reporte de productos con stock bajo.
- Mostrar alertas visuales en la interfaz.

5. **Escenario sencillo de negocio**
    - Simular ingreso de mercadería, ventas y consultoría de inventario.

Diagrama de bloques

Representa los componentes principales del sistema y como se comunican:



Programas fuentes y ejecutables

El proyecto incluye:

1. Código fuente:
    - App.py
    - Carpeta templates con las vistas HTML.
    - Carpeta static con CSS e imágenes.
    - Módulos de python de lógica y acceso a datos.
    - Archivo requirements.txt con las dependencias del proyecto.
2. Ejecutable:
    - Instrucciones en el README.md para configurar la base de datos en SQL Server, configurar variables de conexión.
    - El sistema se ejecuta con Python app.py

Arquitectura utilizada

La arquitectura adoptada es MVC (Modelo – Vista – Controlador), implementada de manera lógica

- Modelo:
    - Módulos de acceso a datos.
    - Clases que representan entidades: Producto, Movimiento, Usuario, Rol.
- Vista:
    - Plantillas HTML con CSS para mostrar formularios, listados y reportes.
    - Diseño responsivo para visualización correcta en escritorio y dispositivos móviles.
- Controlador:
    - Rutas de Flask que reciben las peticiones del usuario.
    - Implementan la lógica de negocio.
    - Deciden qué vista mostrar y qué datos enviar.

Esta arquitectura permite separar responsabilidades, mejorar el mantenimiento y facilitar la evolución futura del sistema.

Recomendaciones para el mantenimiento del software

Para garantizar la continuidad del sistema y su evolución, se recomienda:

1. **Documentación actualizada**
    a. Mantener el README actualizado con instrucciones de instalación y ejecución.
    b. Documentar los módulos principales y las funciones críticas.
2. **Control de versiones**
    a. Utilizar Git y GitHub con ramas develop, main y feature/.
    b. Registrar cambios con mensajes de commit claros.
3. **Mantenimiento de la base de datos**
    a. Realizar respaldo periódico de la base SQL Server.
    b. Documentar cambios en el esquema (script de migraciones).
4. **Pruebas continuas**
    a. Añadir pruebas unitarias para las funciones más importantes.
    b. Realizar pruebas de regresión antes de hacer cambios mayores.

5. **Gestión de errores**
    a. Registrar excepciones y errores en un log de sistema.
    b. Establecer una rutina para revisar y corregir problemas reportados.
6. **Evolución del sistema**
    a. Analizar nuevas necesidades del negocio antes de modificar el código.
    b. Integrar nuevos módulos de manera gradual.

Link del repositorio de GitHub: https://github.com/AshleyAlfaroAguilar/Sistema-de-Gesti-n-de-Inventarios-SGA.git

Conclusión

El desarrollo del Sistema de Gestión de Inventarios para ElectroDistribuciones S.A. permitió integrar conceptos de análisis, diseño, programación y gestión de proyectos en una solución funcional que atiende necesidades reales de control de inventario dentro de una empresa. El sistema implementado cumple con los requisitos establecidos en el Proyecto del Semestre, incluyendo autenticación segura, gestión por roles, registro de productos, control detallado de movimientos y generación de reportes esenciales.

La implementación del módulo de usuarios con roles permite asegurar que solo personal autorizado pueda modificar información sensible, mientras que el módulo de movimientos garantiza trazabilidad completa de entradas y salidas. Los reportes —especialmente el de stock bajo— fortalecen la capacidad operativa del sistema al anticipar quiebres de inventario mediante alertas visuales claras y oportunas.