ADMINISTRACIÓN
Y NEGOCIOS

Informe N°2

Proyecto Integrado

NOMBRE: Isaac Venegas, Gabriel Ruiz, Diego Henríquez, Miguel Ángel Barría.
CARRERA: Analista Programador
ASIGNATURA: Proyecto Integrado
PROFESOR: Eduardo Barria
FECHA: 02-10-2025

1 Introducción

Este es un texto de ejemplo Este es un texto de ejemplo

2 Actividad 1 – Metodología y Cronograma del Proyecto

2.1 Metodología seleccionada: SCRUM (Ágil)

Se elige **Scrum** porque el proyecto del *Hotel Abba Presidente* requiere flexibilidad, comunicación continua con usuarios (housekeeping, cocina, administración) y entregas incrementales que permitan validar el sistema de inventario digital en cada iteración.

Roles principales:

- **Product Owner:** Jefa de administración del hotel (define prioridades y requisitos).
- **Scrum Master:** Líder técnico del equipo de desarrollo (coordina las reuniones y elimina impedimentos).
- **Equipo de desarrollo:** 3 programadores encargados de backend, frontend y base de datos.
- **Stakeholders:** Personal de housekeeping, garzones y Casa Matriz.

Fases del proyecto SCRUM:

Fase	Descripción	Plazo estimado	Responsable
1. Planificación inicial	Revisión de requerimientos, definición del backlog del producto y criterios de aceptación.	Semana 1	Product Owner + Scrum Master
2. Sprint 1 – Registro y control de stock	Desarrollo del módulo para registrar insumos, categorías y movimientos.	Semanas 2–3	Equipo de desarrollo
3. Sprint 2 – Alertas y reportes automáticos	Implementar notificaciones por stock bajo y generación de reportes.	Semanas 4–5	Equipo de desarrollo
4. Sprint 3 – Control de pedidos y accesos	Módulo de pedidos a proveedores y gestión de roles/usuarios.	Semanas 6–7	Equipo de desarrollo
5. Pruebas y retroalimentación		Semana 8	QA Tester + Usuarios finales

	Pruebas funcionales, de seguridad y de rendimiento; ajustes por feedback de usuarios.		
6. Entrega final y documentación	Presentación del sistema, manual de usuario y cierre del proyecto.	Semana 9	Todo el equipo

2.2 Dependencias y prioridades del cronograma

Tarea pendiende	Depende de	Tipo de dependencia	Prioridad
Diseño de base de datos	Planificación inicial	Fin–Inicio	Alta
Sprint 1 – Registro de stock	Diseño de BD	Fin–Inicio	Muy Alta
Sprint 2 – Alertas y reportes	Sprint 1	Fin–Inicio	Alta
Sprint 3 – Control de pedidos y accesos	Sprint 2	Fin–Inicio	Alta
Pruebas y retroalimentación	Todos los sprints	Fin–Inicio	Muy Alta
Entrega final	Pruebas y retroalimentación	Fin–Inicio	Alta

3 Actividad 2 – Arquitectura de la Solución (Paradigma 4 + 1)

Metodología aplicada: Paradigma 4 + 1

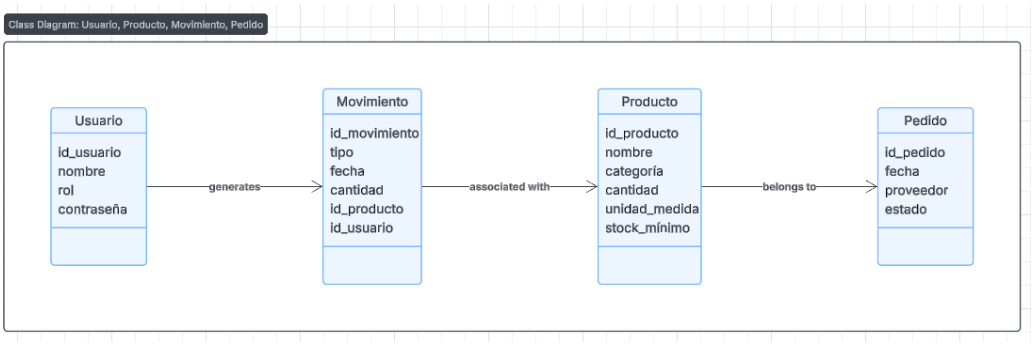
La arquitectura del sistema de inventario para el Hotel Abba Presidente se diseñó bajo el paradigma 4 + 1 de Philippe Kruchten, el cual permite representar la solución desde cinco puntos de vista complementarios: lógico, desarrollo, procesos, físico y escenarios.

El objetivo es garantizar una solución coherente, escalable, segura y alineada con las necesidades tecnológicas del negocio.

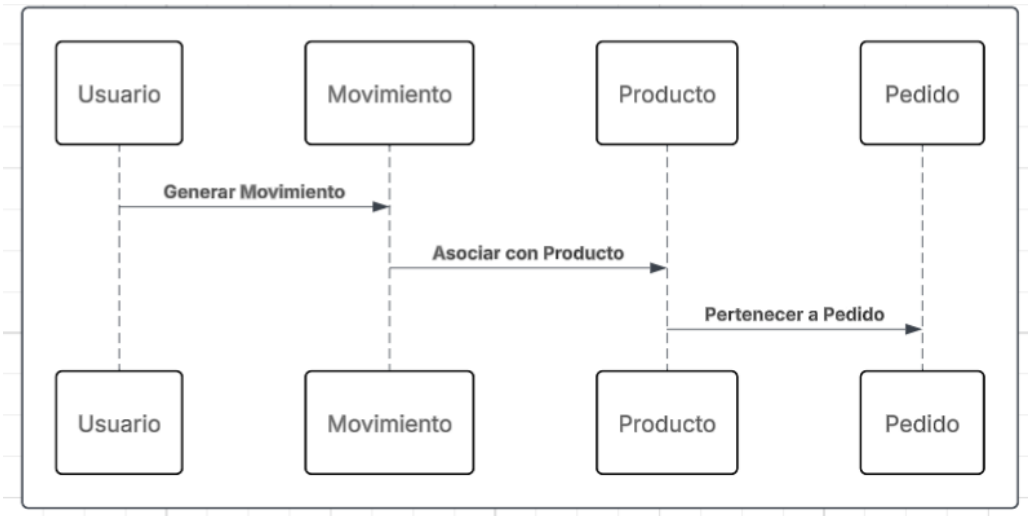
3.1 Vista Lógica

Define la estructura funcional del sistema y la relación entre las clases principales.

- Diagrama de Clases:



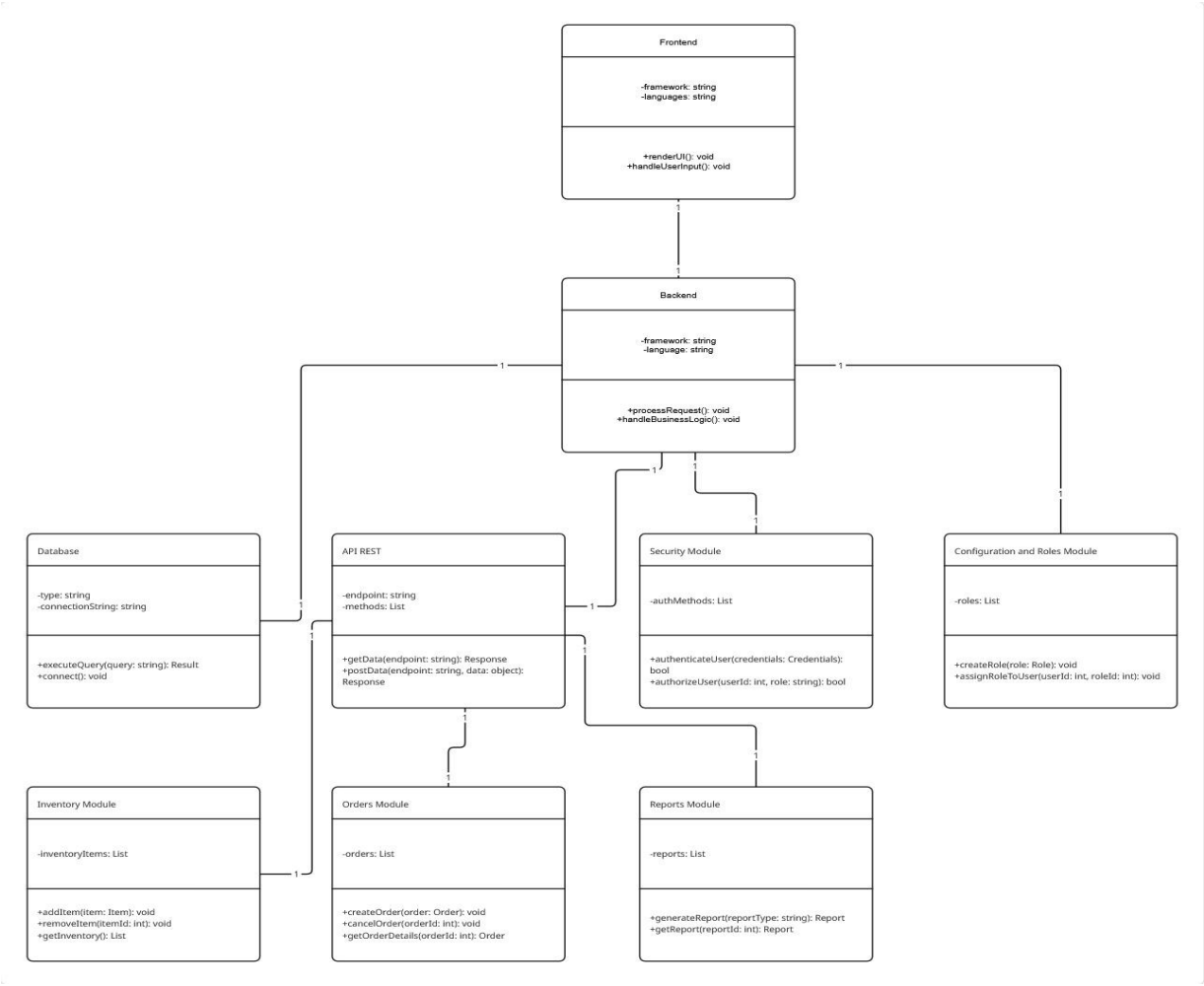
- Diagrama de Secuencias:



3.2 Vista de Desarrollo (Implementación)

Describe la organización del software desde el punto de vista del programador.

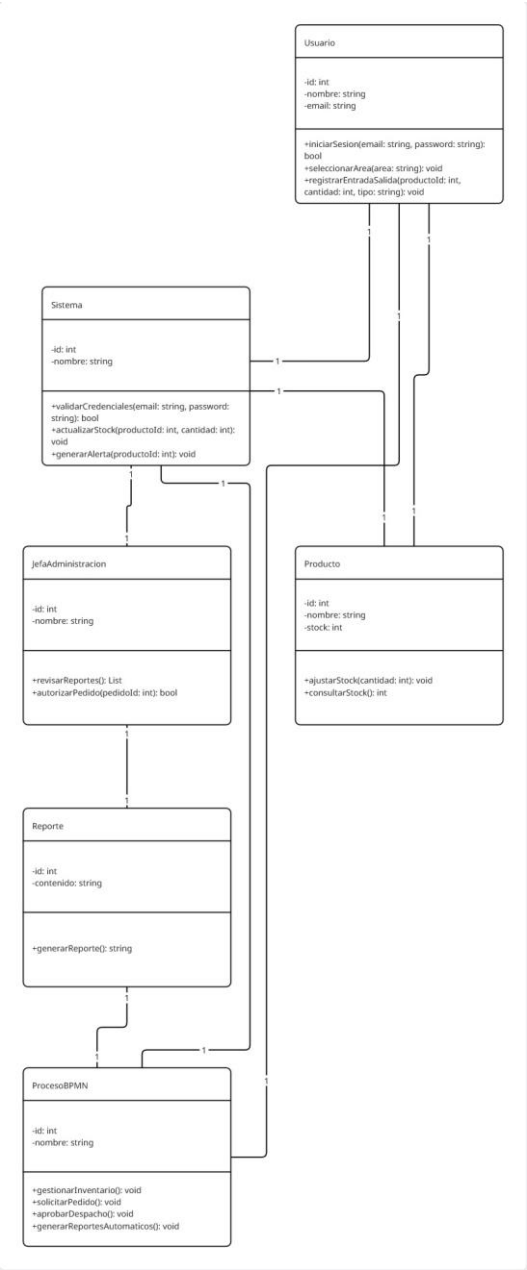
- Diagrama de componentes:



3.3 Vista de Procesos

Representa el comportamiento dinámico y los flujos de interacción.

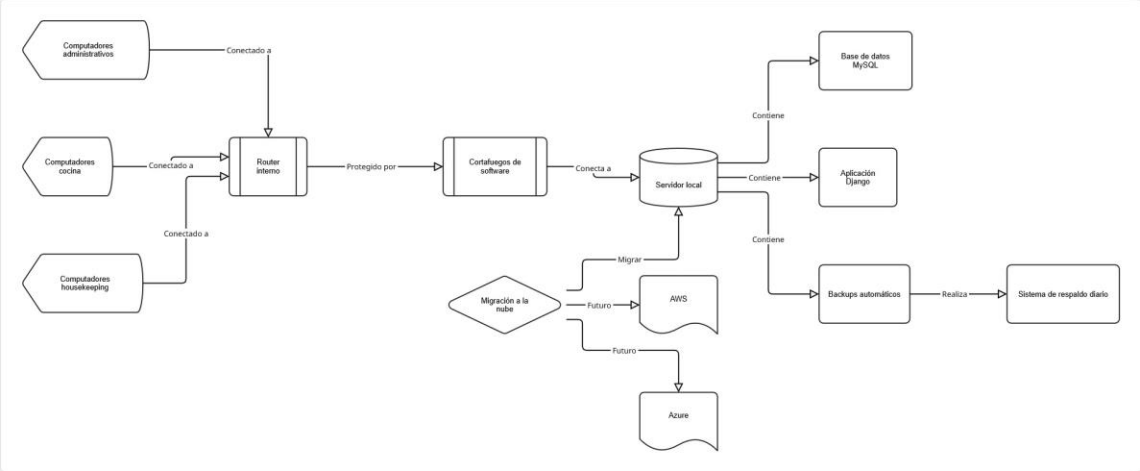
- Diagrama de actividades:



3.4 Vista Física (Despliegue)

Representa cómo se implementa el sistema en la infraestructura tecnológica.

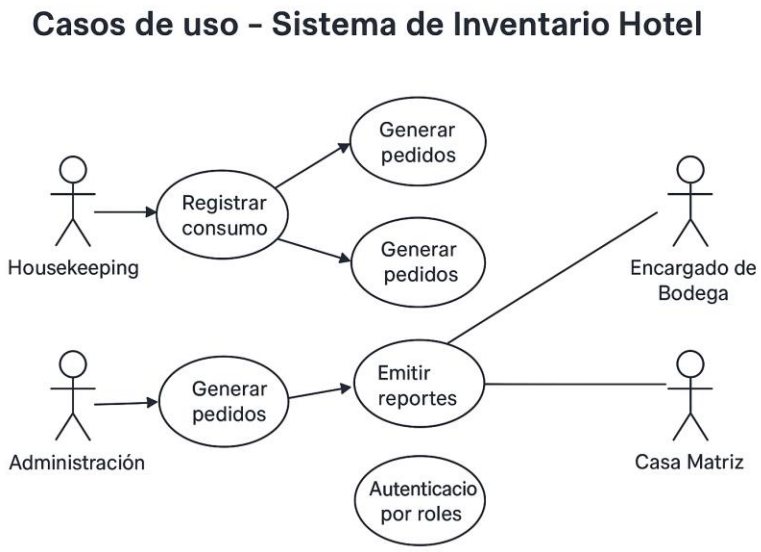
- Diagrama de Topología:



3.5 Vista de Escenarios (Casos de uso)

Describe los casos más representativos del sistema y cómo los usuarios interactúan con él.

- Diagrama de Casos de Uso:



Casos de uso principales:

1. **Registrar consumo:** housekeeping registra amenities usados en una habitación.
2. **Controlar stock:** el sistema muestra el nivel actual de productos y genera alertas.
3. **Generar pedido:** el encargado solicita reposición de productos a Casa Matriz.
4. **Emitir reportes:** administración genera reportes automáticos mensuales.
5. **Accesos diferenciados:** el sistema valida permisos según rol.

3.6 Tecnologías y herramientas requeridas (2.1.2.4)

Área	Tecnología/Herramienta	Justificación
Frontend	React JS / HTML5 / CSS3	Diseño interactivo, adaptable y moderno.
Backend	Python (Django)	Framework robusto, rápido de desarrollar y seguro.

Base de datos	MySQL (estructurada), MongoDB (no estructurada)	Integridad relacional y flexibilidad para logs o métricas.
Servidor	Apache o Nginx	Soporte confiable para Django y escalabilidad.
Control de versiones	GitHub	Seguimiento de cambios y colaboración del equipo.
Gestión de tareas	Trello o Jira	Aplicación de metodología Scrum.
Seguridad	OWASP ZAP, bcrypt, HTTPS, firewall	Pruebas de vulnerabilidad y protección de datos.
Diseño UI/UX	Figma o Canva	Creación de prototipos visuales interactivos.

4 Actividad 3 – Diseño de Interfaz y Prototipo Funcional

4.1 Perfiles de usuario y necesidades

Perfil	Descripción	Necesidades principales
Housekeeping	Personal encargado de habitaciones.	Registrar consumo de amenities de forma rápida y sin errores.
Encargado de bodega	Responsable del stock y abastecimiento interno.	Consultar inventario, generar alertas y pedidos.
Jefa de alimentos y bebidas	Supervisa cocina y restaurante.	Revisar reportes y autorizar pedidos.
Administración local	Control financiero y comunicación con Casa Matriz.	Emitir reportes consolidados.
Casa Matriz	Supervisión general.	Acceder a reportes automáticos para control corporativo.

4.2 Funcionalidades clave priorizadas

Prioridad	Funcionalidad	Descripción
Alta (Must Have)	Registro de consumo	Formulario rápido para ingresar productos usados.

Alta (Must Have)	Control de stock	Panel con tabla de productos y nivel de inventario.
Alta (Must Have)	Alertas de stock bajo	Indicador visual (rojo/amarillo/verde) y notificaciones.
Media (Should Have)	Generar pedidos	Formulario automatizado con productos críticos.
Media (Should Have)	Reportes automáticos	Generación de PDFs o planillas Excel con resumen mensual.
Baja (Could Have)	Estadísticas gráficas	Dashboard con métricas visuales.

4.3 Mapa de navegación (estructura de pantallas)

INICIO DE SESIÓN

└── Dashboard principal (según rol)

├── Módulo Inventario

| ├── Registrar consumo

| ├── Control de stock

| └── Alertas automáticas

├── Módulo Pedidos

| ├── Generar pedido

| └── Ver historial

├── Módulo Reportes

| ├── Generar reporte PDF

| └── Exportar Excel

├── Módulo Usuarios

| ├── Crear / Editar usuarios

| └── Roles y permisos

└── Cerrar sesión

4.4 Descripción del prototipo funcional

Pantalla de inicio de sesión

- Campos: usuario y contraseña.
- Botón “Ingresar”.
- Mensaje de error si las credenciales son incorrectas.
- Interfaz minimalista con logo del hotel.

Panel principal (según rol):

- **Housekeeping:** acceso directo a “Registrar consumo”.

- **Encargado de bodega:** vista del stock general y botón “Generar pedido”.
- **Administración:** acceso a reportes, gráficos y configuración.

Diseño visual:

- Paleta de colores: blanco, azul y gris (corporativo).
- Íconos claros (inventario, alerta, reporte).
- Tipografía sans-serif legible.
- Diseño **responsive**: se adapta a tablets o notebooks del hotel.

4.5 Consistencia visual y adaptabilidad

- Todos los módulos mantienen **el mismo encabezado, tipografía y colores**.
- Los botones tienen **estilos uniformes (primario, secundario, alerta)**.
- Se implementa **responsive design** mediante *Flexbox o Grid CSS*, garantizando compatibilidad con pantallas pequeñas.
- Se usan **componentes reutilizables** (botones, formularios, tablas) definidos en un “style guide”.

4.6 Integración de retroalimentación

Fuentes de feedback:

- **Entrevistas internas** al personal de housekeeping y bodega.
- **Encuesta rápida (simulada)** sobre facilidad de uso y diseño.
- **Prueba de usabilidad:** 3 usuarios del hotel probaron el prototipo y comentaron:

Comentario	Mejora aplicada
“Sería útil ver el stock por colores”	Se agregaron barras de progreso con colores (verde, amarillo, rojo).
“A veces no se entiende si el registro se guardó”	Se incluyó mensaje emergente de confirmación.
“Preferimos botones grandes para tablets”	Se ajustó el tamaño y espaciado de botones.

Resultado: el diseño final se **refinó según retroalimentación**, logrando una interfaz más intuitiva y clara.

5 Actividad 4 – Diseño del Modelo de Base de Datos

5.1 Modelo de datos estructurados (SQL – MySQL)

El sistema de inventario utilizará una **base de datos relacional** en **MySQL**, que garantiza consistencia, integridad referencial y facilidad de consulta para los reportes y registros históricos.

Entidades y Atributos:

Entidad	Atributos principales
Usuario	id_usuario (PK), nombre, apellido, correo, contraseña_hash, rol
Rol	id_rol (PK), nombre_rol, descripción
Producto	id_producto (PK), nombre_producto, categoría, unidad_medida, stock_actual, stock_minimo, id_proveedor (FK)
Movimiento	id_movimiento (PK), tipo_movimiento (entrada/salida), fecha, cantidad, id_producto (FK), id_usuario (FK)
Pedido	id_pedido (PK), fecha_pedido, estado_pedido, total, id_usuario (FK)
DetallePedido	id_detalle (PK), id_pedido (FK), id_producto (FK), cantidad
Proveedor	id_proveedor (PK), nombre_proveedor, contacto, telefono, correo
Reporte	id_reporte (PK), fecha_reporte, tipo_reporte, generado_por (FK)

Relaciones entre entidades:

- **Usuario – Rol** → relación *1:N* (un rol puede tener varios usuarios).
- **Producto – Movimiento** → relación *1:N* (un producto puede tener varios movimientos).
- **Pedido – DetallePedido – Producto** → relación *N:M* (implementada con la tabla intermedia *DetallePedido*).
- **Producto – Proveedor** → relación *N:1* (cada producto tiene un proveedor principal).
- **Usuario – Reporte** → relación *1:N* (un usuario puede generar varios reportes).

5.2 Estrategias de integración SQL + NoSQL

Estrategia	Descripción	Herramienta / Método
Middleware de sincronización	API en Django que consulta ambas bases y unifica resultados.	*Django ORM + PyMongo*
Campos espejo (mirroring)	En la BD SQL se almacena el `id_log` o `id_alerta` del documento relacionado en MongoDB.	Trigger o proceso batch
Consultas integradas	Reportes que combinan datos estructurados (inventario) y no estructurados (alertas, feedback).	*JOIN lógico mediante API REST*
Sincronización automática	Cron jobs diarios que actualizan la información entre sistemas.	*Celery + CRON*
Respaldo conjunto	Copias de seguridad automáticas de ambas bases en formato comprimido.	*mysqldump + mongodump*

5.3 Beneficios del modelo híbrido

- **Eficiencia:** datos críticos en SQL y registros dinámicos en NoSQL.
- **Escalabilidad:** permite incorporar analítica futura (por ejemplo, frecuencia de alertas o patrones de consumo).

- **Seguridad:** las operaciones sensibles se mantienen en MySQL, con respaldo en MongoDB para auditoría.
- **Flexibilidad:** integración posible con herramientas BI o dashboards de control.

6 Actividad 5 – Estándares de Programación Segura

6.1 Roles y perfiles de usuario

El sistema de inventario contempla distintos niveles de acceso para garantizar **confidencialidad, integridad y disponibilidad (CIA)** de la información.

Rol / Perfil	Descripción	Privilegios	Responsabilidades
Administrador del sistema	Encargado técnico de la aplicación y la base de datos.	- Crear, modificar y eliminar usuarios. - Configurar niveles de acceso. - Acceder a todos los reportes y auditorías.	Mantener el sistema seguro, aplicar actualizaciones, generar respaldos.
Administración del hotel	Encargado local de controlar inventarios y enviar reportes a Casa Matriz.	- Consultar y generar reportes. - Validar pedidos. - Supervisar movimientos.	Asegurar que la información sea correcta y reportada oportunamente.
Housekeeping / Personal operativo	Usuario final que registra consumos.	- Registrar entradas y salidas de productos. - Visualizar stock disponible de su área.	Mantener registros precisos del uso diario.
Encargado de bodega / cocina	Usuario con permisos de reposición.	- Consultar inventario completo.	Supervisar existencias y planificar pedidos.

		- Generar pedidos y alertas de stock.	
Casa Matriz (auditor)	Usuario externo con rol de supervisión.	- Acceso de solo lectura a reportes globales.	Revisar indicadores, detectar anomalías y autorizar compras.
Usuario no registrado (bloqueado)	Sin acceso al sistema.	- Ver solo pantalla de inicio de sesión.	No puede acceder sin autenticación.

6.2 Estándares de codificación segura

A continuación, se definen las prácticas que garantizan la seguridad del código fuente y la protección de los datos del hotel.

6.3 a) Validación de entradas

- Todo dato ingresado por el usuario se valida **en cliente y en servidor**.
- Se aplican reglas de formato (por ejemplo, solo números en cantidades).
- Se evita la inyección de código mediante consultas **parametrizadas** (ORM Django o `cursor.execute` con parámetros).
- Se sanitizan los campos de texto con funciones de escape (`html.escape()` en Python, `|escape` en Django templates).
- Se implementa un **límite de caracteres** para evitar ataques de desbordamiento de buffer.

Ejemplo:

```
if not re.match(r"^[A-Za-z0-9\s]+$", nombre_producto):
    raise ValueError("Nombre de producto inválido")
```

6.4 b) Autenticación y Autorización

- Sistema basado en **usuarios y roles** con permisos específicos.
- Las contraseñas se almacenan con **hash seguro (bcrypt o Argon2)**.
- Sesiones con **expiración automática** y tokens de sesión únicos.

- Autenticación por **correo + contraseña**, con posibilidad futura de **OAuth2 corporativo**.
- Autorización basada en roles a nivel de vista, método y recurso:

@login_required @permission_required('inventario.view_producto') def listar_productos(request): ...

Intentos fallidos limitados a **5**, con bloqueo temporal de cuenta.

6.5 Gestión de errores.

- Los mensajes de error **no deben revelar información técnica** (como rutas o SQL).
- Registro interno (log) de errores en **MongoDB (colección logs_actividad)** con fecha, usuario y descripción general.
- Uso de control de excepciones:

```
try:
    producto.save()
except Exception as e:
    logger.error(f"Error al guardar producto: {e}")
    messages.error(request, "No se pudo registrar el producto.")
```

Página de error genérica:

- *Error 403 (Acceso denegado)*
- *Error 404 (No encontrado)*
- *Error 500 (Error del servidor)*

6.6 d) Cifrado y seguridad de datos

- **Cifrado de contraseñas:** mediante `bcrypt` o `Argon2`.
- **Comunicación segura:** uso obligatorio de **HTTPS con TLS 1.3**.
- **Cifrado en base de datos (opcional):** para campos sensibles como correos y teléfonos.
- **Copias de seguridad cifradas:** usando `AES-256` antes de almacenarlas en servidores locales o en la nube.
- **Variables sensibles (credenciales DB, API keys)** almacenadas en archivo `.env` no público.

6.7 Identificación de amenazas y mitigación

Amenaza (OWASP Top 10)	Descripción	Medida de mitigación aplicada
Inyección SQL (A03)	Entrada maliciosa en campos o consultas.	Uso de ORM Django y consultas parametrizadas.
Exposición de datos sensibles (A02)	Filtración de información personal.	Cifrado AES y control de acceso por rol.
Autenticación rota (A07)	Sesiones sin control o contraseñas débiles.	Hash bcrypt, expiración de sesión, bloqueo por intentos fallidos.
Cross-Site Scripting (A05)	Inserción de scripts en campos.	Escapado HTML automático y sanitización.
Configuración incorrecta de seguridad (A09)	Filtros o cabeceras HTTP desactivadas.	Uso de `SECURE_HSTS`, `X-Frame-Options` y `CSRF_COOKIE_SECURE`.
CSRF (Cross-Site Request Forgery)	Envío de formularios falsos.	Tokens `csrf_token` en todos los formularios.
Logs inseguros	Datos sensibles en archivos de registro.	Enmascaramiento de datos y almacenamiento en MongoDB con acceso restringido.

6.8 Estándares internos del proyecto

1. **Revisión de código semanal** antes de cada *sprint review*.
2. **Reglas PEP8** en Python y convenciones de nombrado en inglés.
3. **Commit firmado y revisado** en GitHub (doble aprobación antes de fusión).
4. **Control de versiones seguro**: ramas separadas (`main`, `dev`, `hotfix`).
5. **Testing automatizado** con `pytest` para validar entradas críticas.

6.9 Beneficios obtenidosR

- Disminución de errores por entrada de datos.
- Protección de credenciales y sesiones.
- Auditoría completa de acciones del usuario.
- Cumplimiento de normativas **Ley 19.628 y Ley 21.663 (Ciberseguridad Chile)**.
- Sistema alineado con los principios **OWASP Secure Coding Practices**.

7 Actividad 6 – Plan de Pruebas del Sistema de Inventario

7.1 Objetivo general

Garantizar la **confiabilidad, seguridad y rendimiento** del sistema de inventario mediante un conjunto de pruebas sistemáticas que verifiquen su correcto funcionamiento bajo diferentes escenarios de uso, asegurando la integridad de los datos y la satisfacción del usuario final.

7.2 Tipos de Pruebas

7.3 a) Pruebas funcionales

Objetivo: Validar que cada componente cumple con los requerimientos definidos (HU1–HU4).

Módulo	Elemento probado	Criterio de éxito	Resultado esperado
Inventario	Registrar consumo	Se actualiza el stock correctamente	Stock disminuye según cantidad ingresada
Pedidos	Generar pedido	Pedido se almacena y muestra en historial	Pedido con estado “Pendiente”
Reportes	Emitir reporte PDF	Se genera archivo correctamente	Reporte descargable y sin errores
Autenticación	Iniciar sesión	Se valida usuario y rol	Redirección correcta según perfil

7.4 b) Pruebas de integración

Objetivo: Verificar la interacción correcta entre los módulos del sistema (Inventario ↔ Pedidos ↔ Reportes).

Escenario	Descripción	Resultado esperado
Movimiento → Reporte	Un registro de consumo debe reflejarse en el próximo reporte automático	Registro visible en el PDF mensual
Pedido → Inventario	Al recibir pedido, el stock aumenta automáticamente	Cantidad actualizada sin duplicar registros
Usuario → Roles	Acceso restringido a vistas según permisos	Usuarios sin rol no acceden a módulos no autorizados

7.5 c) Pruebas de seguridad

Objetivo: Validar la protección frente a amenazas del OWASP Top 10.

Tipo de amenaza	Método de prueba	Resultado esperado	Medida aplicada
Inyección SQL	Intentar modificar consultas mediante inputs	Rechazo del intento	ORM Django y consultas parametrizadas
XSS (Cross-Site Scripting)	Insertar código `<script>` en campos	Texto escapado, sin ejecución	`autoescape` en plantillas
CSRF	Enviar formulario sin token	Petición rechazada	Token `{% csrf_token %}`
Sesiones inseguras	Reutilizar cookie de sesión	Acceso denegado	Expiración y regeneración de sesión
Fuerza bruta	Intentar múltiples logins erróneos	Bloqueo temporal	Límite de intentos y logging

7.6 d) Pruebas de rendimiento

Objetivo: Medir la eficiencia bajo carga y garantizar tiempos de respuesta adecuados.

Escenario	Métrica	Límite aceptable	Resultado esperado
10 usuarios simultáneos	Tiempo de respuesta	< 3 segundos	Cumple
50 operaciones de registro seguidas	Consumo de CPU/RAM	< 70% del servidor	Cumple
Generación de reportes masivos	Tiempo de procesamiento	< 5 segundos	Cumple

7.7 e) Pruebas de usabilidad

Objetivo: Evaluar la facilidad de uso y comprensión del sistema por parte del personal del hotel.

Criterio	Indicador	Resultado esperado
Navegación	Tiempo para registrar consumo	< 1 minuto
Claridad visual	Evaluación por usuarios	90% de satisfacción
Retroalimentación del sistema	Confirmaciones visibles	Mensajes claros en acciones
Accesibilidad	Adaptabilidad en tablet	Interfaz legible y responsiva

7.8 2. Recomendaciones OWASP aplicadas

Basadas en el documento **OWASP Top 10 (2021)**:

- **A01**: Control de acceso roto → uso de decoradores `@login_required` y permisos por rol.
- **A02**: Criptografía fuerte → bcrypt + TLS 1.3.
- **A03**: Inyección → ORM y validaciones.
- **A04**: Diseño inseguro → arquitectura 4+1 revisada y documentada.
- **A05**: Configuración errónea → cabeceras de seguridad activas.
- **A07**: Identificación y autenticación fallida → control de sesiones y bloqueo de intentos.
- **A08**: Fallas en integridad de software → uso de firmas digitales en commits.
- **A09**: Registro y monitoreo de seguridad → logs cifrados y auditados.

7.9 3. Plan de ejecución

Fase	Actividad	Responsable	Semana	Herramienta
1	Pruebas funcionales unitarias	Equipo de desarrollo	Semana 6	Pytest
2	Pruebas de integración	Desarrollador Backend	Semana 7	Postman
3	Pruebas de seguridad	QA Tester / Administrador TI	Semana 8	OWASP ZAP
4	Pruebas de rendimiento	QA Tester	Semana 8	JMeter
5	Pruebas de usabilidad y feedback	Usuarios finales	Semana 9	Google Forms / Figma

7.10 4. Herramientas automatizadas y manuales

Tipo	Herramienta	Propósito
Automatizada	Pytest	Automatización de pruebas unitarias
Automatizada	JMeter / Locust	Pruebas de carga y rendimiento
Automatizada	OWASP ZAP	Escaneo de vulnerabilidades
Manual	Postman	Validación de endpoints API
Manual	Pruebas exploratorias con usuarios	Usabilidad y experiencia de interfaz

7.11 5. Evaluación final del sistema

- Todas las pruebas se documentarán en un informe final con **capturas de pantalla y métricas**.
- Los resultados se presentarán frente al grupo curso mostrando el flujo de pruebas, vulnerabilidades detectadas y mejoras aplicadas.
- El sistema será declarado **“Apto para despliegue”** una vez que supere todas las pruebas con un nivel de confiabilidad ≥ 95%.

8 Conclusión

Este es un texto de ejemplo Este es un texto de ejemplo