GA10-220501097-AA10-EV01

ELABORA DOCUMENTOS TÉCNICOS Y DE USUARIO DEL SOFTWARE

APRENDIZ

ANDRES FELIPE QUIÑONEZ ARREDONDO

Servicio Nacional de Aprendizaje SENA
Análisis y desarrollo de software Ficha 2977402

INSTRUCTOR

EDWARD JAVIER GARCIA CORREDOR

Medellín, Antioquia, Colombia
12 de octubre de 2025

INTRODUCCION

Este manual técnico documenta la arquitectura, datos, instalación, despliegue y operación del Sistema de Inventario (Django + MySQL + Tailwind). Está dirigido a desarrolladores, administradores de base de datos y personal DevOps que deban mantener, modificar o desplegar el sistema.

DESARROLLO

Prerrequisitos de instalación

Hardware (mínimo pruebas): 2 CPU, 4 GB RAM, 30 GB SSD.
SO recomendado: Ubuntu 22.04 LTS (servidor).
Software / herramientas:

- Python 3.10+
- pip
- virtualenv / venv
- MySQL Server 8.0 (InnoDB).
- Node.js 16+ y npm (para Tailwind).
- Git (>=2.30).
- Gunicorn + Nginx (producción).
- Opcional: Docker y docker-compose para contenedores.

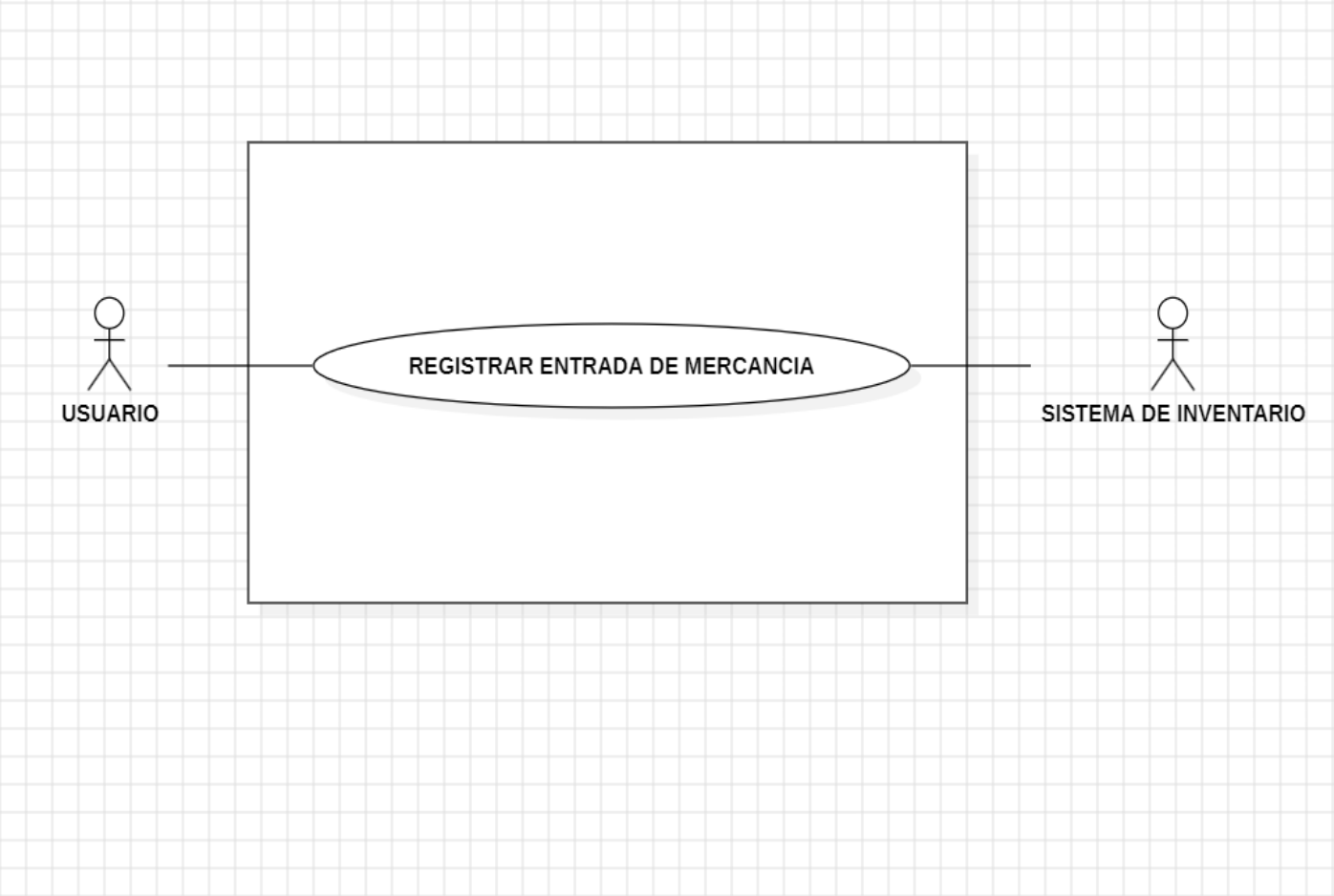
Permisos / red: cuenta con privilegios para crear DB, ejecutar scripts; puertos 80/443 y 3306 según arquitectura.

FRAMEWORKS, LIBRERÍAS Y ESTÁNDARES

- Django 4.x — backend.
- Django REST Framework — APIs.
- MySQL 8.0 — gestor relacional.
- Tailwind CSS v3 — estilos frontend.
- Gunicorn + Nginx — servidor de aplicaciones y proxy.
- GitHub Actions — CI/CD.
- Estándares: RESTful, SemVer para versionado, bcrypt/argon2 para hashing de contraseñas, cifrado para backups.

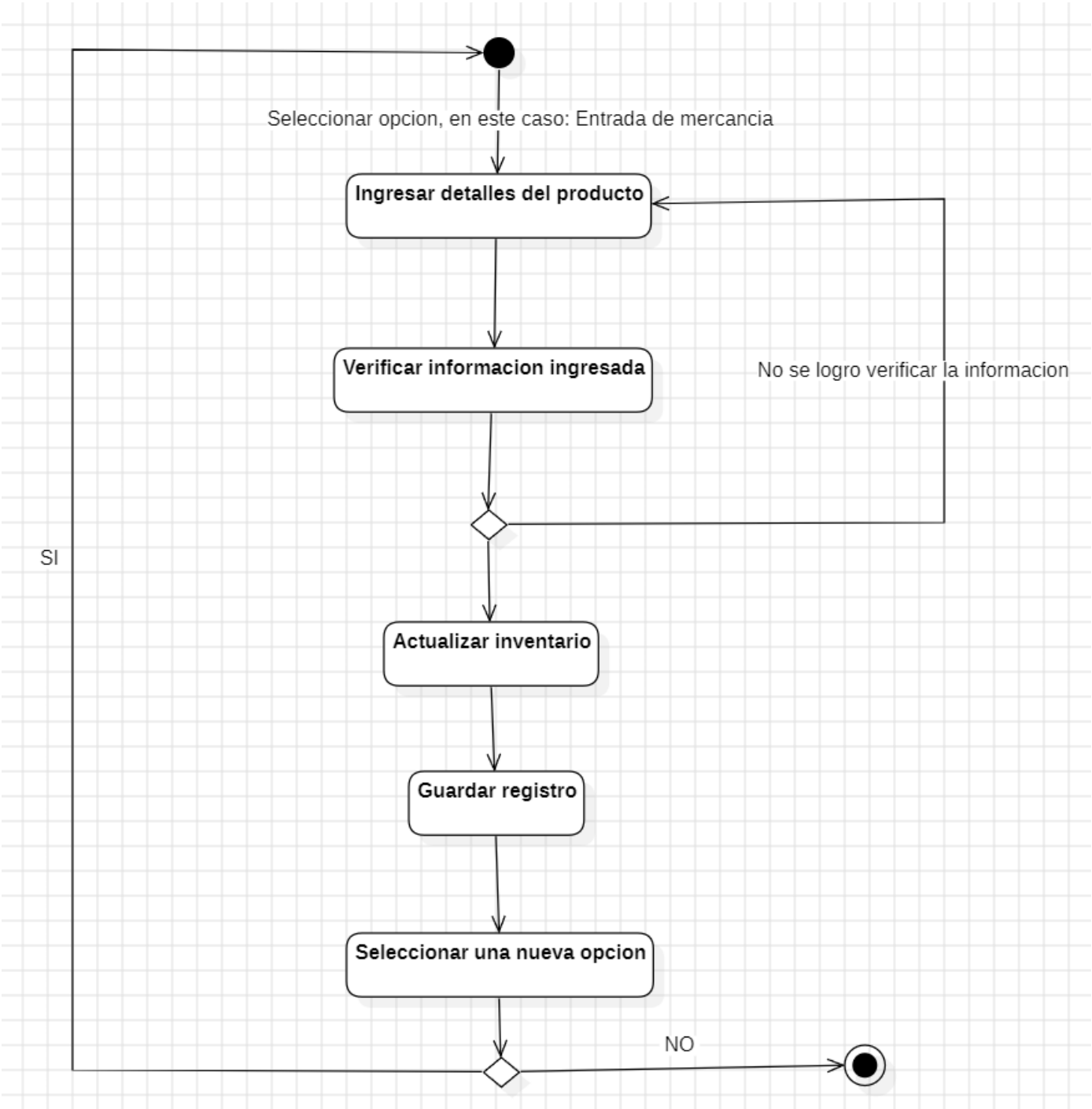
DIAGRAMA DE CASOS DE USO

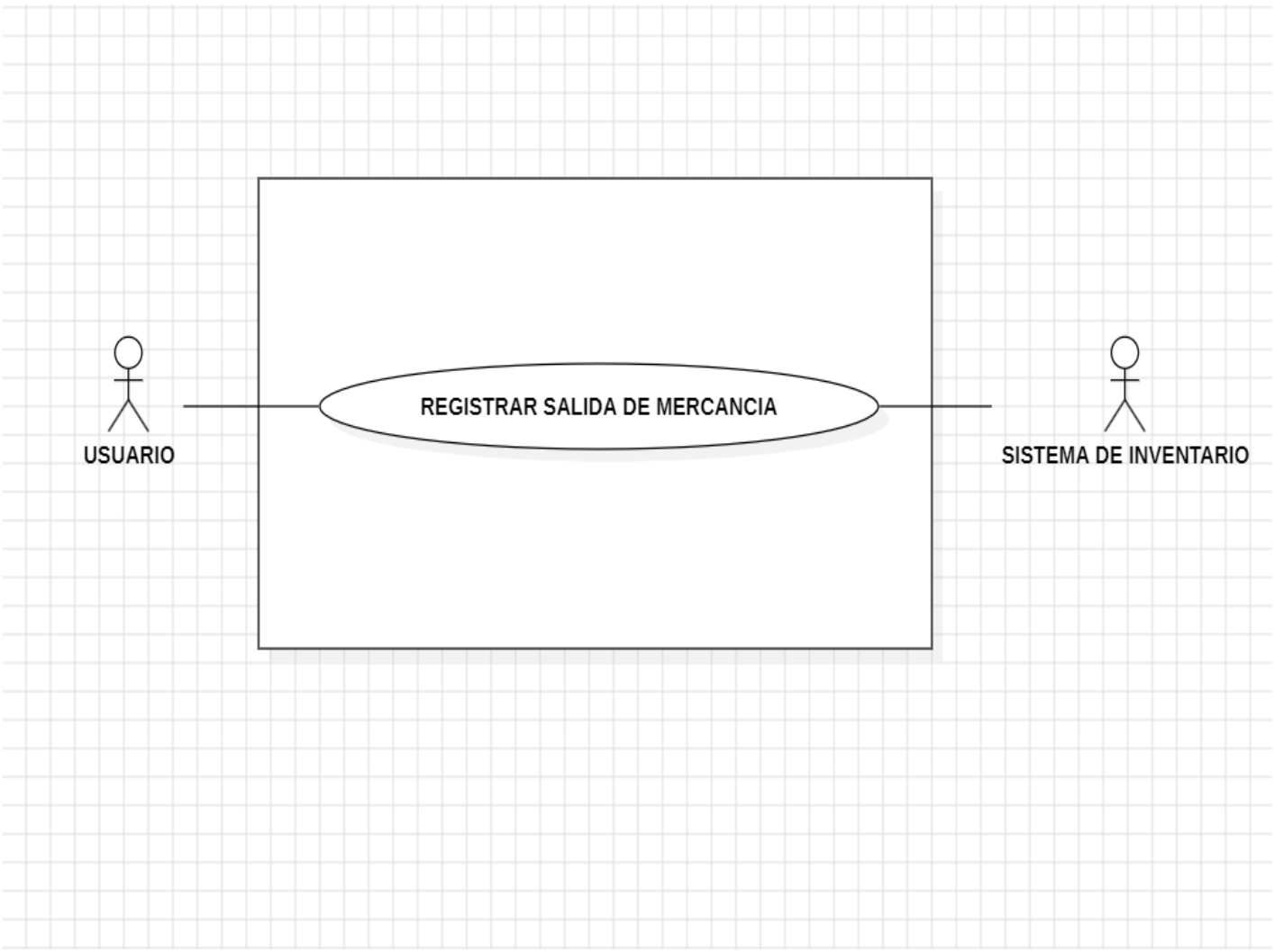
CASOS DE USO, ESPECIFICACION DE CASOS DE USO Y DIAGRAMA DE ACTIVIDADES



1. Caso de Uso: Registrar Entrada de Mercancía

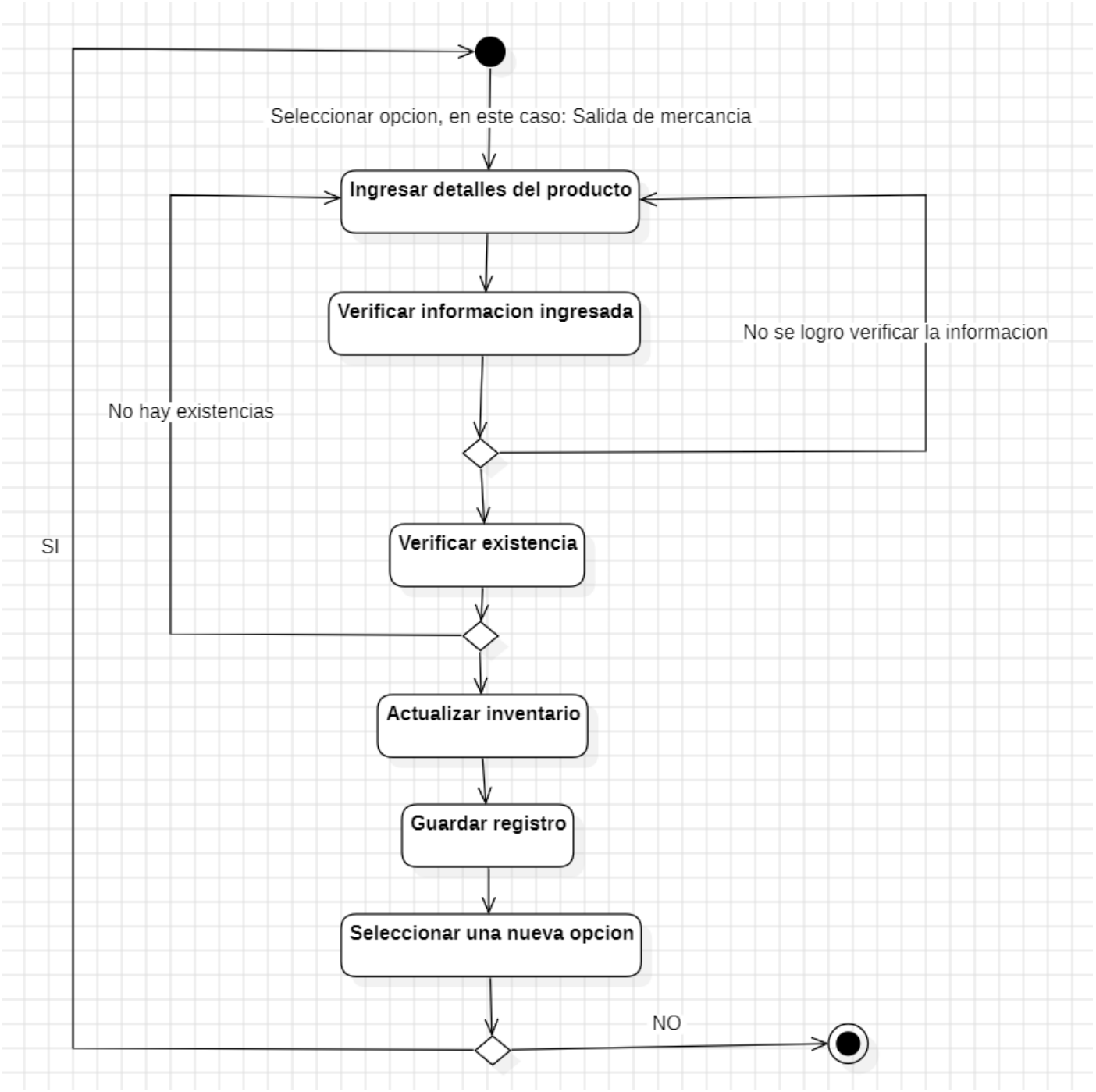
- **Autor**: Andres Felipe Quiñonez Arredondo
- **Fecha**: 04/09/2024
- **Descripción**: Este caso de uso permite a los usuarios registrar la entrada de mercancía en el inventario, actualizando los niveles de stock y guardando los detalles de la transacción.
- **Actores del caso de uso**: Usuario (Administrador de Inventario), Sistema de Inventario
- **Precondiciones**:
    - El usuario debe estar autenticado en el sistema.
    - Los productos que se van a registrar deben estar previamente creados en el catálogo del inventario.
- **Flujo normal**:
    1. El usuario selecciona la opción de "Registrar Entrada de Mercancía" en el sistema.
    2. El sistema solicita al usuario que ingrese los detalles de la mercancía (ID del producto, nombre, cantidad, proveedor, fecha de entrada).
    3. El usuario ingresa la información solicitada.
    4. El sistema verifica la información ingresada (campos obligatorios, formatos, etc.).
    5. El sistema actualiza los niveles de inventario según la cantidad ingresada.
    6. El sistema guarda el registro de entrada de mercancía en la base de datos.
    7. El sistema confirma al usuario que la entrada ha sido registrada exitosamente.
- **Flujo alternativo**:
    - Si la información ingresada es incorrecta o está incompleta, el sistema muestra un mensaje de error y solicita al usuario que ingrese nuevamente los datos del producto antes de continuar.
- **Poscondiciones**:
    - El sistema ha actualizado correctamente los niveles de inventario y ha guardado el registro de la entrada de mercancía.

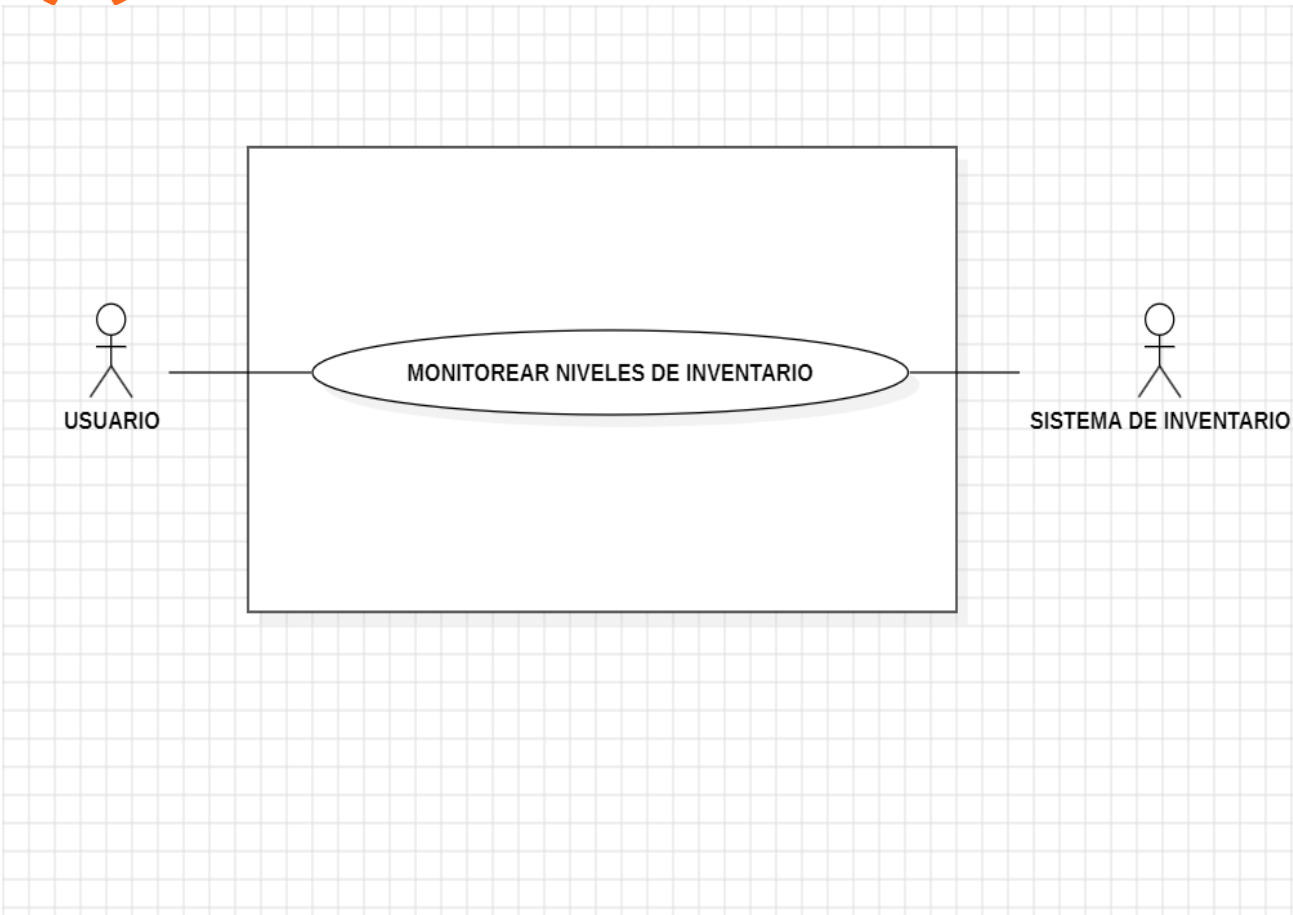

Seleccionar opcion, en este caso: Entrada de mercancia
Ingresar detalles del producto
Verificar informacion ingresada
No se logro verificar la informacion
SI
Actualizar inventario
Guardar registro
Seleccionar una nueva opcion
NO


USUARIO
REGISTRAR SALIDA DE MERCANCIA
SISTEMA DE INVENTARIO

2. Caso de Uso: Registrar Salida de Mercancía

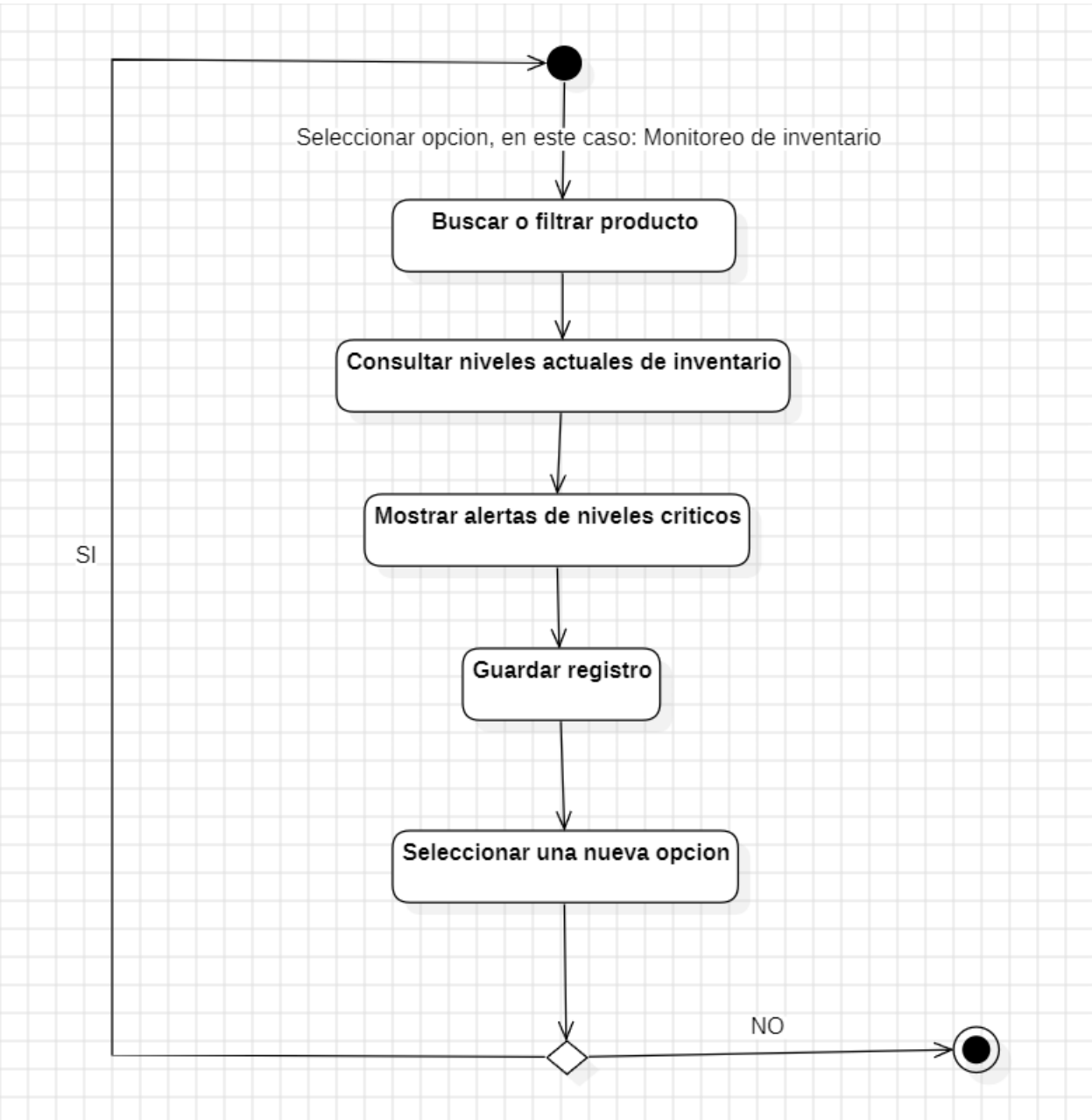
- **Autor**: Andres Felipe Quiñonez Arredondo
- **Fecha**: 04/09/2024
- **Descripción**: Este caso de uso permite a los usuarios registrar la salida de productos del inventario, actualizando los niveles de stock y guardando los detalles de la transacción.
- **Actores del caso de uso**: Usuario (Administrador de Inventario), Sistema de Inventario
- **Precondiciones**:
  - El usuario debe estar autenticado en el sistema.
  - Los productos que se van a retirar deben estar previamente registrados en el inventario.
- **Flujo normal**:
  1. El usuario selecciona la opción de "Registrar Salida de Mercancía" en el sistema.
  2. El sistema solicita al usuario que ingrese los detalles de la salida (ID del producto, nombre, cantidad, motivo de salida, fecha de salida).
  3. El usuario ingresa la información solicitada.
  4. El sistema verifica la información ingresada (campos obligatorios, formatos, disponibilidad de stock, etc.).
  5. El sistema actualiza los niveles de inventario reduciendo la cantidad indicada.
  6. El sistema guarda el registro de salida de mercancía en la base de datos.
  7. El sistema confirma al usuario que la salida ha sido registrada exitosamente.
- **Flujo alternativo**:
  - Si la información ingresada es incorrecta o está incompleta, el sistema muestra un mensaje de error y solicita al usuario que ingrese nuevamente los datos del producto antes de continuar.
  - Si la cantidad solicitada no está disponible en el inventario, el sistema muestra un mensaje de error y solicita al usuario que ajuste la cantidad.
- **Poscondiciones**:
  - El sistema ha actualizado correctamente los niveles de inventario y ha guardado el registro de la salida de mercancía.

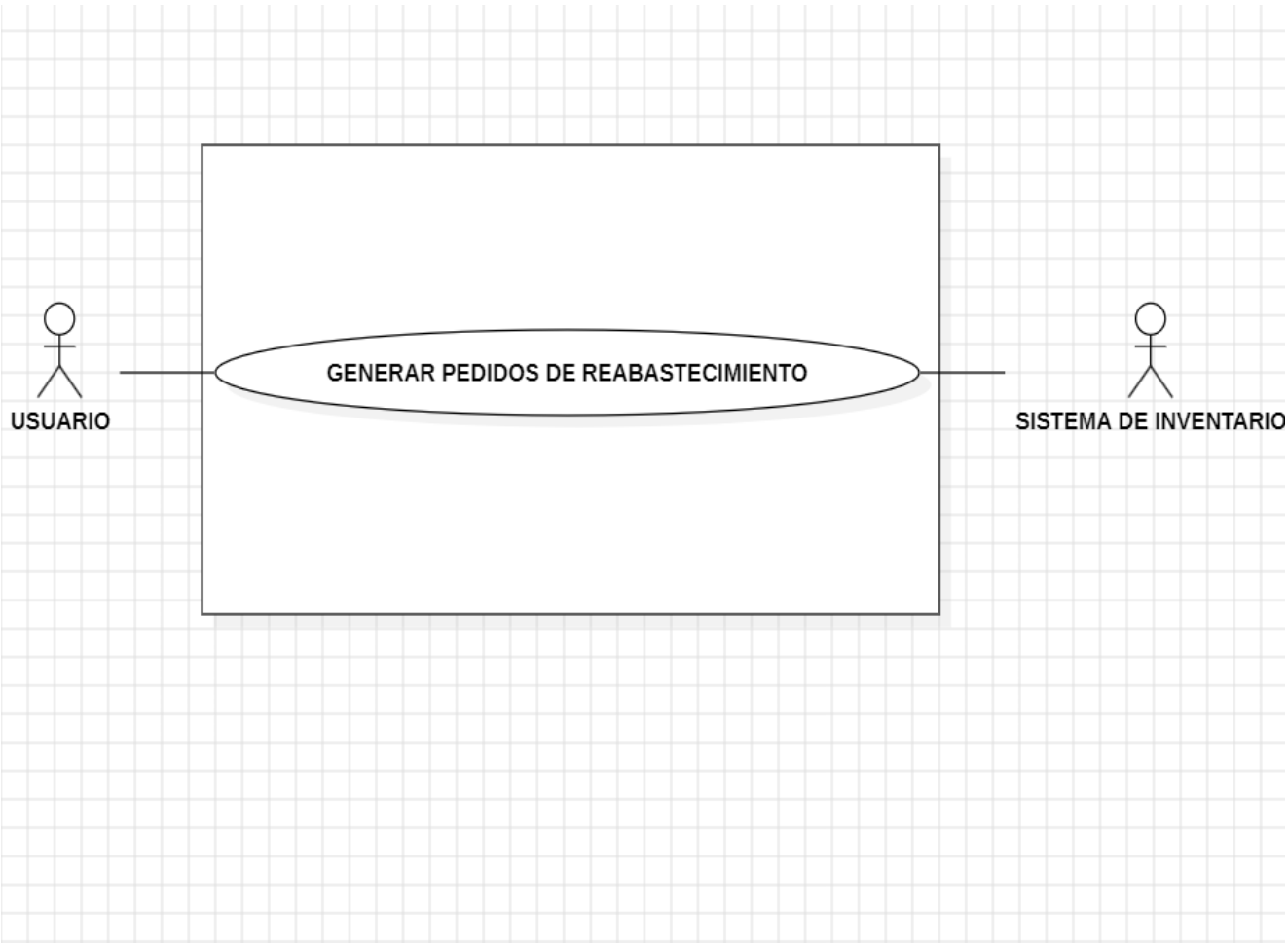

Seleccionar opcion, en este caso: Salida de mercancia
Ingresar detalles del producto
Verificar informacion ingresada
No se logro verificar la informacion
No hay existencias
Verificar existencia
SI
Actualizar inventario
Guardar registro
Seleccionar una nueva opcion
NO


MONITOREAR NIVELES DE INVENTARIO
USUARIO
SISTEMA DE INVENTARIO

3. Caso de Uso: Monitorear Niveles de Inventario

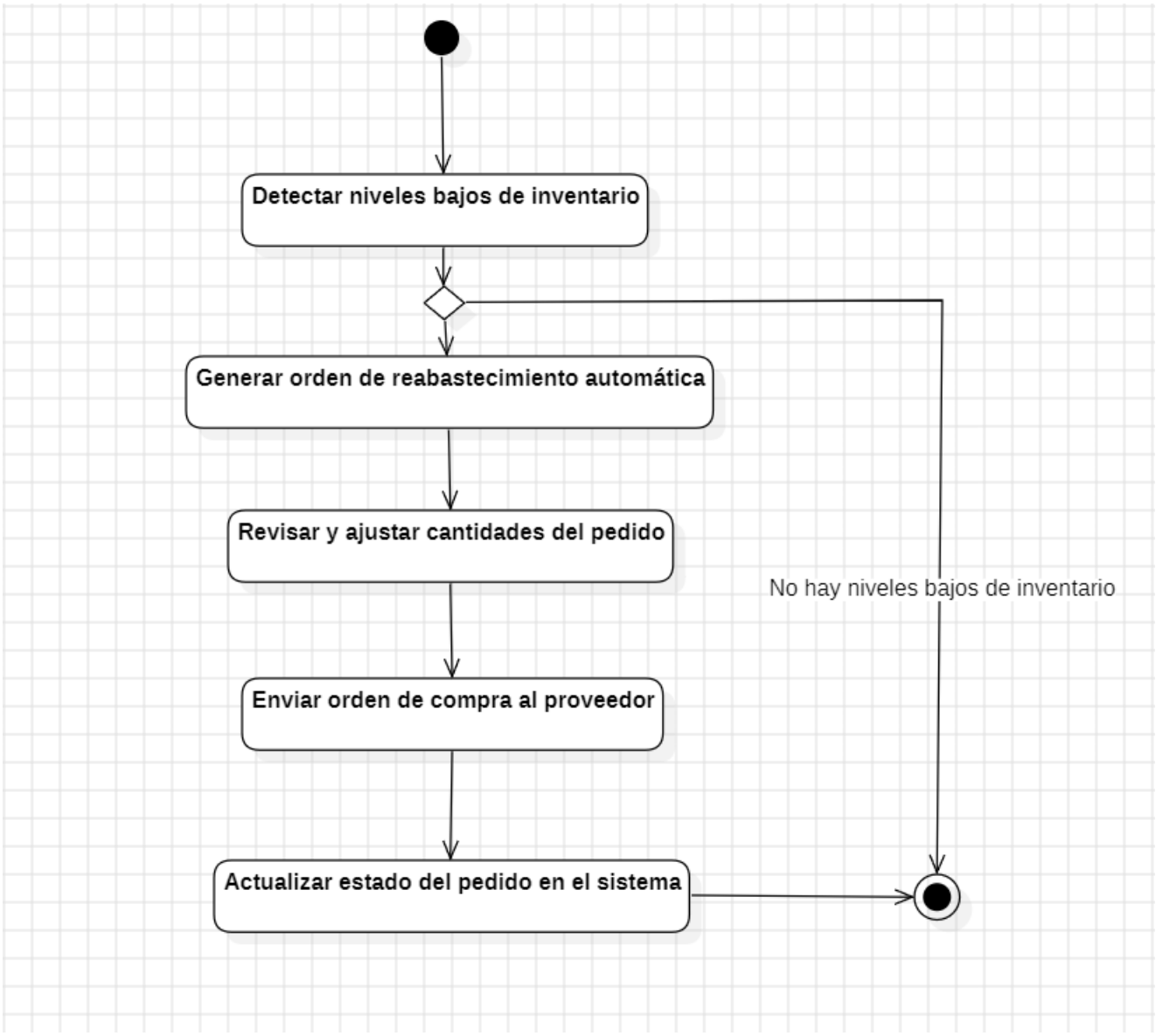
- **Autor**: Andres Felipe Quiñonez Arredondo
- **Fecha**: 04/09/2024
- **Descripción**: Este caso de uso permite a los usuarios consultar y monitorear los niveles actuales de inventario, incluyendo alertas de productos con niveles críticos de stock.
- **Actores del caso de uso**: Usuario (Administrador de Inventario), Sistema de Inventario
- **Precondiciones**:
    - El usuario debe estar autenticado en el sistema.
- **Flujo normal**:
    1. El usuario selecciona la opción de "Monitorear Niveles de Inventario" en el sistema.
    2. El sistema muestra una lista de todos los productos en el inventario con sus niveles actuales de stock.
    3. El usuario puede buscar o filtrar productos específicos.
    4. El sistema muestra las alertas de productos con niveles críticos de stock.
    5. El usuario revisa la información y puede tomar acciones como generar un pedido de reabastecimiento.
- **Flujo alternativo**:
    - No aplica.
- **Poscondiciones**:
    - El usuario ha visualizado los niveles actuales de inventario y ha recibido alertas sobre productos con niveles críticos de stock.

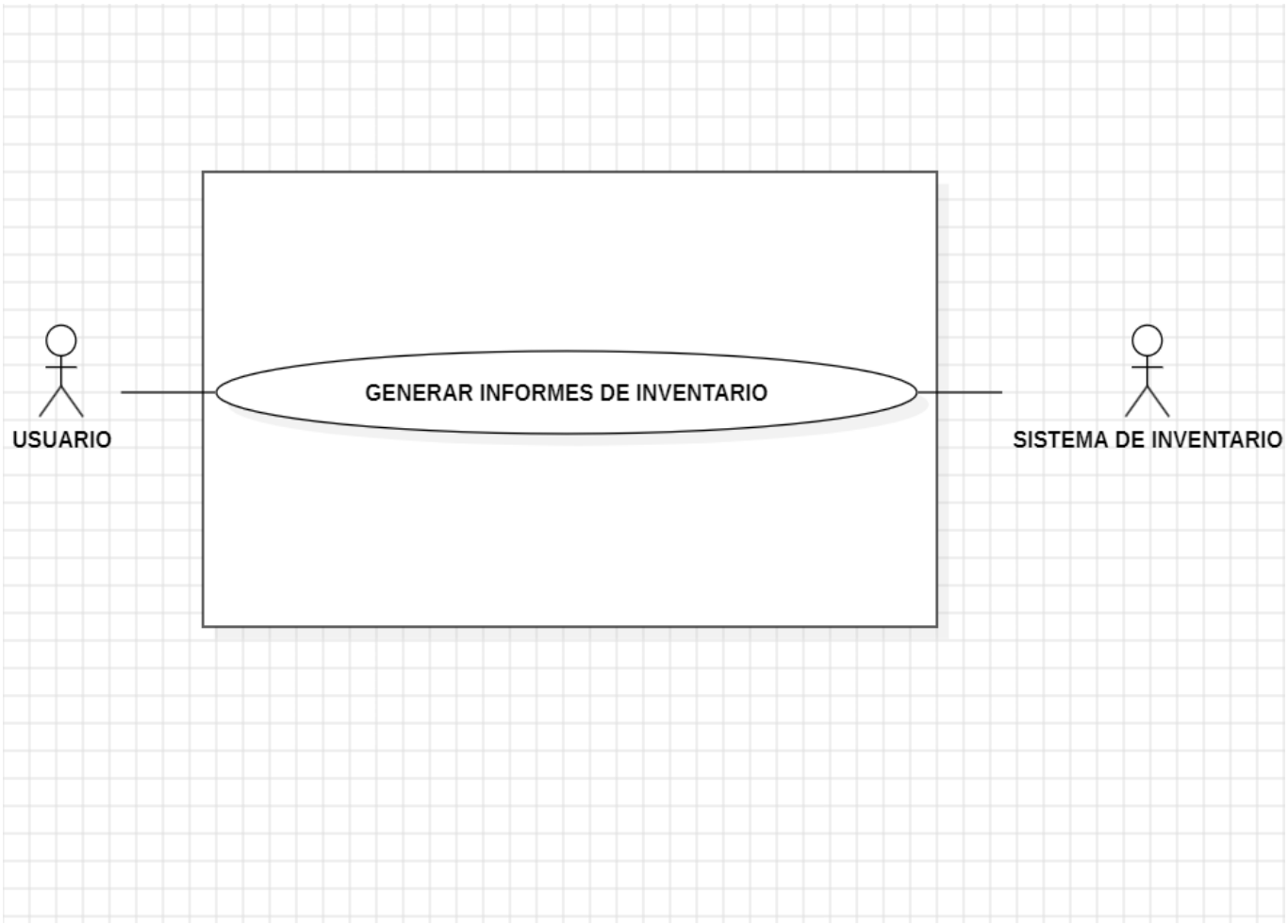

Seleccionar opcion, en este caso: Monitoreo de inventario
Buscar o filtrar producto
Consultar niveles actuales de inventario
Mostrar alertas de niveles criticos
Guardar registro
Seleccionar una nueva opcion
SI
NO


GENERAR PEDIDOS DE REABASTECIMIENTO
USUARIO
SISTEMA DE INVENTARIO

4. Caso de Uso: Generar Pedidos de Reabastecimiento

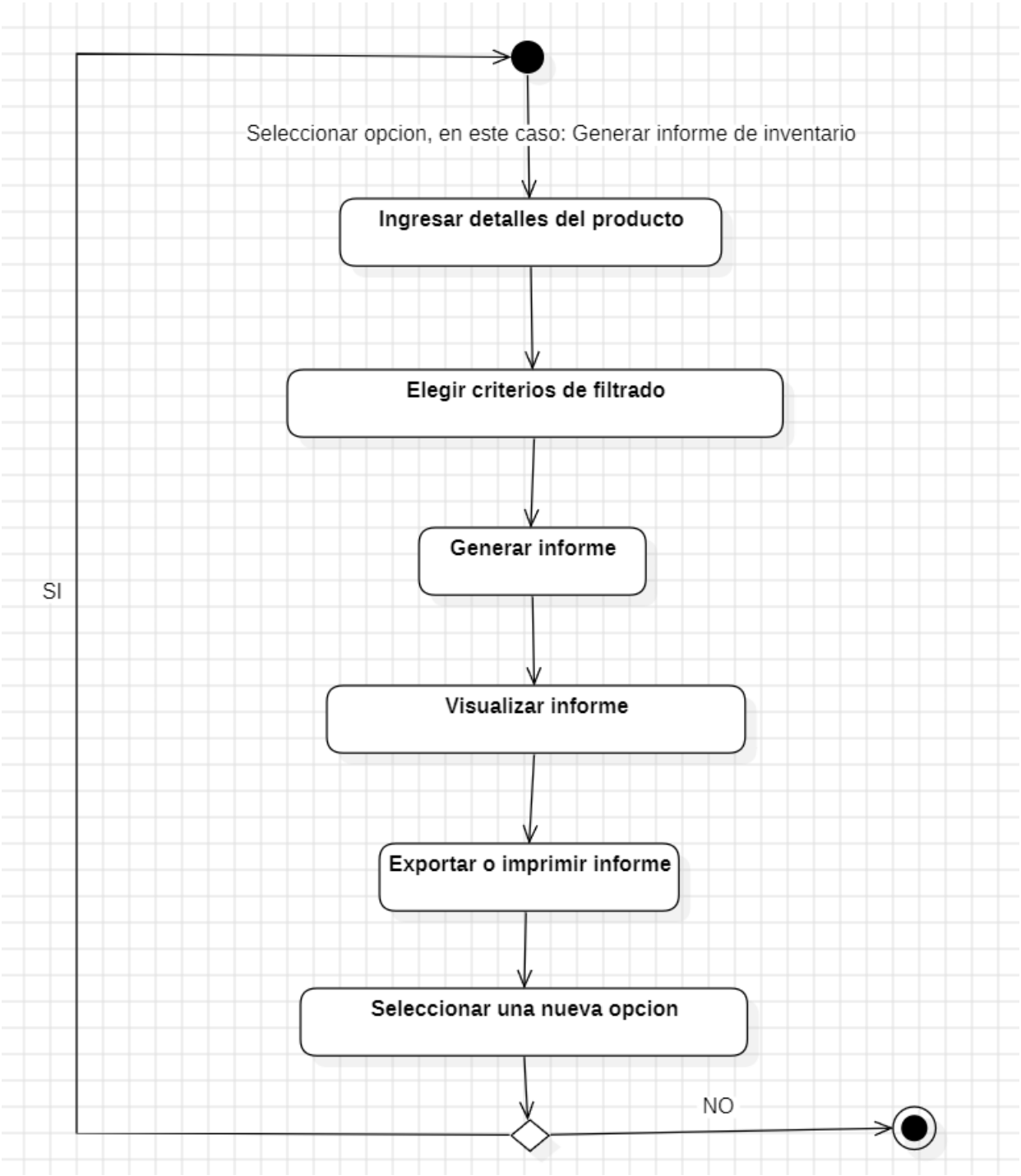
- **Autor**: Andres Felipe Quiñonez Arredondo
- **Fecha**: 04/09/2024
- **Descripción**: Este caso de uso permite al sistema generar automáticamente pedidos de reabastecimiento cuando los niveles de inventario caen por debajo de un umbral definido.
- **Actores del caso de uso**: Usuario (Administrador de Inventario), Sistema de Inventario
- **Precondiciones**:
    - El sistema debe estar configurado con los umbrales mínimos de stock para cada producto.
- **Flujo normal**:
    1. El sistema detecta que los niveles de inventario de un producto han caído por debajo del umbral definido.
    2. El sistema genera automáticamente una orden de reabastecimiento para el producto.
    3. El usuario revisa y ajusta las cantidades del pedido si es necesario.
    4. El usuario confirma y envía la orden de compra al proveedor.
    5. El sistema actualiza el estado del pedido en la base de datos.
- **Flujo alternativo**:
    - Si el usuario decide no proceder con el pedido, el sistema permite cancelar la orden de reabastecimiento.
- **Poscondiciones**:
    - El pedido de reabastecimiento ha sido generado, revisado, y enviado, con el estado del pedido actualizado en el sistema.


Detectar niveles bajos de inventario
Generar orden de reabastecimiento automática
Revisar y ajustar cantidades del pedido
Enviar orden de compra al proveedor
Actualizar estado del pedido en el sistema
No hay niveles bajos de inventario


GENERAR INFORMES DE INVENTARIO
USUARIO
SISTEMA DE INVENTARIO

5. Caso de Uso: Generar Informes de Inventario

- **Autor**: Andres Felipe Quiñonez Arredondo
- **Fecha**: 04/09/2024
- **Descripción**: Este caso de uso permite a los usuarios generar informes detallados sobre el estado del inventario, con opciones para filtrar y exportar los datos.
- **Actores del caso de uso**: Usuario (Administrador de Inventario), Sistema de Inventario
- **Precondiciones**:
    - El usuario debe estar autenticado en el sistema.
- **Flujo normal**:
    1. El usuario selecciona la opción de "Generar Informes de Inventario" en el sistema.
    2. El sistema solicita al usuario que elija los criterios de filtrado (rango de fechas, productos específicos, etc.).
    3. El usuario selecciona los criterios deseados.
    4. El sistema genera el informe basado en los criterios seleccionados.
    5. El usuario visualiza el informe generado.
    6. El usuario puede optar por exportar o imprimir el informe.
- **Flujo alternativo**:
    - No aplica.
- **Poscondiciones**:
    - El informe ha sido generado y está disponible para exportación o impresión según la necesidad del usuario.

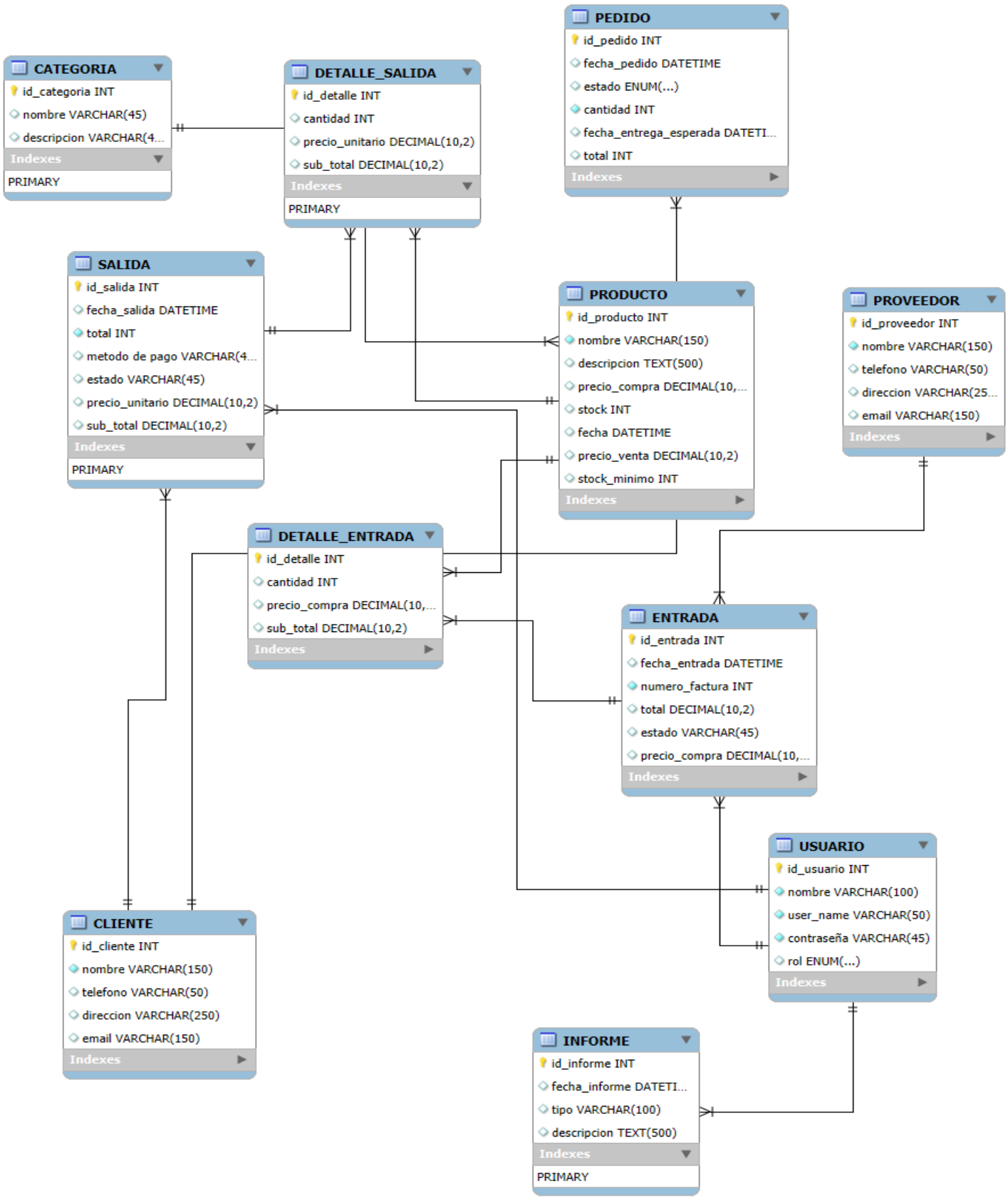

Seleccionar opcion, en este caso: Generar informe de inventario
Ingresar detalles del producto
Elegir criterios de filtrado
Generar informe
Visualizar informe
Exportar o imprimir informe
Seleccionar una nueva opcion
SI
NO

MODELO ENTIDAD–RELACIÓN DEL SISTEMA DE INVENTARIO

Entidades y atributos principales:

- USUARIO (id_usuario PK) — roles (admin, operador)
- CLIENTE (id_cliente PK)
- PROVEEDOR (id_proveedor PK)
- CATEGORIA (id_categoria PK)
- PRODUCTO (id_producto PK) — FK → CATEGORIA
- ENTRADA (id_entrada PK) — FK → PROVEEDOR, USUARIO
- DETALLE_ENTRADA — FK → ENTRADA, PRODUCTO
- SALIDA (id_salida PK) — FK → CLIENTE, USUARIO
- DETALLE_SALIDA — FK → SALIDA, PRODUCTO
- PEDIDO / DETALLE_PEDIDO — pedidos de cliente
- INFORME / AUDITORÍA — logs

Representación Gráfica en MySQL Workbench



DICCIONARIO DE DATOS

Tabla: USUARIO

Campo	Tipo	NULL	PK / AI	Defecto	Observaciones
id_usuario	INT	NO	PK, AUTO_INCREMENT		Identificador usuario
nombre	VARCHAR(100)	NO			Nombre completo
user_name	VARCHAR(50)	NO	UNIQUE		Usuario para login
contraseña	VARCHAR(255)	NO			Hash de contraseña (bcrypt/argon2 recomendado)
rol	ENUM('admin','operador')	SÍ		'operador'	Permisos/rol

Tabla: PROVEEDOR

Campo	Tipo	NULL	PK / AI	Observaciones
id_proveedor	INT	NO	PK, AUTO_INCREMENT	Identificador proveedor
nombre	VARCHAR(150)	NO		Razón social
telefono	VARCHAR(50)	SÍ		
direccion	VARCHAR(250)	SÍ		
email	VARCHAR(150)	SÍ		

Tabla: CATEGORIA

Campo	Tipo	NULL	PK / AI	Observaciones
id_categoria	INT	NO	PK, AUTO_INCREMENT	
nombre	VARCHAR(45)	SÍ		
descripcion	VARCHAR(255)	SÍ		

Tabla: PRODUCTO

Campo	Tipo	NULL	PK / AI	Observaciones
id_producto	INT	NO	PK, AUTO_INCREMENT	Ajustado a AI en patch
nombre	VARCHAR(150)	NO		
descripcion	TEXT	SÍ		
precio_compra	DECIMAL(10,2)	SÍ	DEFAULT 0.00	
precio_venta	DECIMAL(10,2)	SÍ		
stock	INT	SÍ	DEFAULT 0	
stock_minimo	INT	SÍ		
fecha	DATETIME	SÍ	DEFAULT CURRENT_TIMESTAMP	
id_categoria	INT	SÍ	FK → CATEGORIA(id_categoria)	

Tabla: CLIENTE

Campo	Tipo	NULL	PK / AI	Observaciones
id_cliente	INT	NO	PK, AUTO_INCREMENT	
nombre	VARCHAR(150)	NO		
telefono	VARCHAR(50)	SÍ		
direccion	VARCHAR(250)	SÍ		
email	VARCHAR(150)	SÍ		

Tabla: ENTRADA

Campo	Tipo	NULL	PK / AI	Observaciones
id_entrada	INT	NO	PK, AUTO_INCREMENT	
fecha_entrada	DATETIME	SÍ	DEFAULT CURRENT_TIMESTAMP	
numero_factura	VARCHAR(45)	SÍ		
total	DECIMAL(10,2)	SÍ		
id_proveedor	INT	SÍ	FK → PROVEEDOR(id_proveedor)	
id_usuario	INT	SÍ	FK → USUARIO(id_usuario)	

Tabla: DETALLE_ENTRADA

Campo	Tipo	NULL	PK / AI	Observaciones
id_detalle	INT	NO	PK, AUTO_INCREMENT	
id_entrada	INT	SÍ	FK → ENTRADA(id_entrada)	
id_producto	INT	SÍ	FK → PRODUCTO(id_producto)	
cantidad	INT	SÍ		
precio_compra	DECIMAL(10,2)	SÍ		
sub_total	DECIMAL(10,2)	SÍ		

Tabla: SALIDA

Campo	Tipo	NULL	PK / AI	Observaciones
id_salida	INT	NO	PK, AUTO_INCREMENT	
fecha_salida	DATETIME	SÍ	DEFAULT CURRENT_TIMESTAMP	
total	DECIMAL(10,2)	SÍ		
metodo_pago	VARCHAR(45)	SÍ		
id_cliente	INT	SÍ	FK → CLIENTE(id_cliente)	
id_usuario	INT	SÍ	FK → USUARIO(id_usuario)	

Tabla: DETALLE_SALIDA

Campo	Tipo	NULL	PK / AI	Observaciones
id_detalle	INT	NO	PK, AUTO_INCREMENT	
id_salida	INT	SÍ	FK → SALIDA(id_salida)	
id_producto	INT	SÍ	FK → PRODUCTO(id_producto)	
cantidad	INT	SÍ		
precio_unitario	DECIMAL(10,2)	SÍ		
sub_total	DECIMAL(10,2)	SÍ		

Tabla: PEDIDO

Campo	Tipo	NULL	PK / AI	Observaciones
id_pedido	INT	NO	PK, AUTO_INCREMENT	
fecha_pedido	DATETIME	SÍ	DEFAULT CURRENT_TIMESTAMP	
estado	VARCHAR(45)	SÍ		
id_cliente	INT	SÍ	FK → CLIENTE(id_cliente)	
total	DECIMAL(10,2)	SÍ		

Tabla: DETALLE_PEDIDO

Campo	Tipo	NULL	PK / AI	Observaciones
id_detalle	INT	NO	PK, AUTO_INCREMENT	
id_pedido	INT	SÍ	FK → PEDIDO(id_pedido)	
id_producto	INT	SÍ	FK → PRODUCTO(id_producto)	
cantidad	INT	SÍ		

Tabla: INFORME

Campo	Tipo	NULL	PK / AI	Observaciones
id_informe	INT	NO	PK, AUTO_INCREMENT	
fecha_informe	DATETIME	SÍ	DEFAULT CURRENT_TIMESTAMP	
tipo	VARCHAR(100)	SÍ		
descripcion	TEXT	SÍ		
id_usuario	INT	SÍ	FK → USUARIO(id_usuario)	

SCRIPTS DE INSTALACIÓN E IMPORTACIÓN

`setup_env.sh` — preparación del entorno (Ubuntu/Debian)

```
#!/bin/bash
set -e
# setup_env.sh — prepara entorno base en Ubuntu/Debian (dev/prod adaptado)
echo "Actualizando paquetes..."
sudo apt update
echo "Instalando dependencias del SO..."
sudo apt install -y python3-venv python3-dev build-essential libmysqlclient-dev git curl \
                    default-mysql-client nodejs npm unzip
echo "Creando entorno virtual..."
python3 -m venv venv
source venv/bin/activate
echo "Actualizando pip..."
pip install --upgrade pip
if [ -f requirements.txt ]; then
  pip install -r requirements.txt
else
  pip install django djangorestframework mysqlclient python-dotenv gunicorn
fi
echo "Entorno preparado. Activar con: source venv/bin/activate"
```

`create_db_and_user.sh` — crear base de datos y usuario MySQL

```
#!/bin/bash
# Uso: ./create_db_and_user.sh DB_NAME DB_USER DB_PASS
DB_NAME=${1:-inventario_db}
DB_USER=${2:-django_user}
DB_PASS=${3:-CambiarPass123!}

echo "Creando DB '${DB_NAME}' y usuario '${DB_USER}'..."
mysql -u root -p <<SQL
CREATE DATABASE IF NOT EXISTS \`${DB_NAME}\` CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci;
CREATE USER IF NOT EXISTS '${DB_USER}'@'localhost' IDENTIFIED BY '${DB_PASS}';
GRANT ALL PRIVILEGES ON \`${DB_NAME}\`.* TO '${DB_USER}'@'localhost';
FLUSH PRIVILEGES;
SQL
echo "Creación completada."
```

`import_sql.sh` — backup previo e importación del archivo SQL

```
#!/bin/bash
# Uso: ./import_sql.sh DB_NAME SQL_FILE DB_USER
DB_NAME=${1:-inventario_db}
SQL_FILE=${2:-INVENTARIO_X_SCRIPT.sql}
DB_USER=${3:-root}

TIMESTAMP=$(date +%Y%m%d_%H%M%S)
BACKUP_FILE="backup_before_import_${TIMESTAMP}.sql"

if [ -e "${SQL_FILE}" ]; then
  echo "Creando backup previo de la base '${DB_NAME}'..."
```

```
  mysqldump --single-transaction --routines --triggers -u ${DB_USER} -p ${DB_NAME} >
${BACKUP_FILE}
  echo "Backup guardado en: ${BACKUP_FILE}"
else
  echo "Archivo SQL ${SQL_FILE} no encontrado. Colocar INVENTARIO_X_SCRIPT.sql en la carpeta de
trabajo."
  exit 1
fi

echo "Importando ${SQL_FILE} en la base ${DB_NAME}..."
mysql -u ${DB_USER} -p ${DB_NAME} < ${SQL_FILE} 2> import_errors.log || {
  echo "ERROR durante importación. Revisar import_errors.log"
  exit 1
}
echo "Importación finalizada."
echo "Tablas en ${DB_NAME}:"
mysql -u ${DB_USER} -p -e "USE ${DB_NAME}; SHOW TABLES;"

echo "Conteo ejemplo de PRODUCTO (si existe tabla PRODUCTO):"
mysql -u ${DB_USER} -p -e "USE ${DB_NAME}; SELECT COUNT(*) AS productos FROM PRODUCTO;" || true

echo "IMPORT COMPLETADO."
```

`verify_import.sh` — verificación y chequeos básicos después de importar

```
#!/bin/bash
# Uso: ./verify_import.sh DB_NAME DB_USER
DB_NAME=${1:-inventario_db}
DB_USER=${2:-root}

echo "Listado de tablas en DB: ${DB_NAME}"
mysql -u ${DB_USER} -p -e "USE ${DB_NAME}; SHOW TABLES;"

echo "Constraints (FOREIGN KEY) en ${DB_NAME}:"
mysql -u ${DB_USER} -p -e "SELECT TABLE_NAME, CONSTRAINT_NAME FROM
information_schema.TABLE_CONSTRAINTS WHERE CONSTRAINT_TYPE='FOREIGN KEY' AND
TABLE_SCHEMA='${DB_NAME}';"

echo "Conteos básicos (PRODUCTO, PROVEEDOR, USUARIO) si existen:"
mysql -u ${DB_USER} -p -e "USE ${DB_NAME}; SELECT 'producto' AS tabla, COUNT(*) AS filas FROM
PRODUCTO UNION ALL SELECT 'proveedor', COUNT(*) FROM PROVEEDOR UNION ALL SELECT 'usuario',
COUNT(*) FROM USUARIO;" || true

echo "Verificación finalizada."
```

`backup.sh` — generar backup comprimido y checksum

```
#!/bin/bash
# Uso: ./backup.sh DB_NAME DEST_FOLDER DB_USER
DB_NAME=${1:-inventario_db}
DEST_FOLDER=${2:-./backups}
DB_USER=${3:-root}
mkdir -p ${DEST_FOLDER}
TIMESTAMP=$(date +%Y%m%d_%H%M%S)
OUT_FILE="${DEST_FOLDER}/${DB_NAME}_full_${TIMESTAMP}.sql"
echo "Generando backup ${OUT_FILE} ..."
mysqldump --single-transaction --routines --triggers -u ${DB_USER} -p ${DB_NAME} > ${OUT_FILE}
gzip -f ${OUT_FILE}
```

```
echo "Backup comprimido: ${OUT_FILE}.gz"
sha256sum ${OUT_FILE}.gz > ${OUT_FILE}.gz.sha256
echo "Checksum generado: ${OUT_FILE}.gz.sha256"
```

`rollback.sh` — restaurar backup (rollback)

```
#!/bin/bash
# Uso: ./rollback.sh BACKUP_FILE.gz DB_NAME DB_USER
BACKUP_FILE=${1:-backup_before_import.sql.gz}
DB_NAME=${2:-inventario_db}
DB_USER=${3:-root}

if [ ! -f "${BACKUP_FILE}" ]; then
  echo "Backup ${BACKUP_FILE} no encontrado. Abortando."
  exit 1
fi

echo "Restaurando backup ${BACKUP_FILE} en DB ${DB_NAME}..."
gunzip -c ${BACKUP_FILE} | mysql -u ${DB_USER} -p ${DB_NAME} || {
  echo "Error restaurando backup."
  exit 1
}
echo "Restauración completada correctamente."
```

`docker-compose.yml` — opción de despliegue con contenedores

```
version: '3.8'
services:
  db:
    image: mysql:8.0
    container_name: mysql_inventario
    environment:
      MYSQL_DATABASE: inventario_db
      MYSQL_USER: django_user
      MYSQL_PASSWORD: CambiarPass123!
      MYSQL_ROOT_PASSWORD: RootPassSegura!
    volumes:
      - db_data:/var/lib/mysql
    ports:
      - "3306:3306"

  web:
    build: .
    container_name: web_inventario
    command: bash -c "python manage.py migrate --noinput && gunicorn proyecto.wsgi:application
--bind 0.0.0.0:8000"
    volumes:
      - .:/app
    ports:
      - "8000:8000"
    environment:
      - DB_HOST=db
      - DB_NAME=inventario_db
      - DB_USER=django_user
      - DB_PASSWORD=CambiarPass123!
    depends_on:
      - db
```

```
volumes:
  db_data:
```

`Dockerfile` — imagen base para la aplicación

```
FROM python:3.11-slim
ENV PYTHONDONTWRITEBYTECODE=1
ENV PYTHONUNBUFFERED=1
WORKDIR /app
COPY requirements.txt /app/
RUN apt-get update && apt-get install -y build-essential default-libmysqlclient-dev gcc && rm -
rf /var/lib/apt/lists/*
RUN pip install --upgrade pip
RUN pip install -r requirements.txt
COPY . /app/
CMD ["gunicorn", "proyecto.wsgi:application", "--bind", "0.0.0.0:8000"]
```

`.env.template` — plantilla de variables de entorno

```
# .env.template - renombrar a .env y completar valores reales
SECRET_KEY=poner_aqui_un_valor_secreto
DEBUG=False
DB_HOST=localhost
DB_NAME=inventario_db
DB_USER=django_user
DB_PASSWORD=CambiarPass123!
ALLOWED_HOSTS=localhost,127.0.0.1
```

`requirements.txt` — dependencias Python

```
Django>=4.2
djangorestframework
mysqlclient
python-dotenv
gunicorn
```

DIAGRAMA DE COMPONENTES

El diagrama de componentes representa la estructura lógica y física del sistema de inventario, mostrando cómo se relacionan los diferentes módulos, servicios y artefactos del software.

El sistema de inventario desarrollado en Django está compuesto por varios módulos interconectados que permiten gestionar productos, entradas, salidas, usuarios y reportes.

Cada componente cumple una función específica dentro de la arquitectura cliente-servidor con una capa de presentación (frontend), una capa lógica (backend) y una capa de datos (base de datos).

componentes principales

Componente	Descripción técnica	Dependencias	Responsable / Rol
Interfaz de usuario (Frontend)	Proporciona la interfaz visual del sistema (HTML, CSS, JS, Tailwind). Interactúa con el backend mediante peticiones HTTP/REST.	Navegador web, API REST	Desarrollador frontend
API / Lógica de negocio (Backend Django)	Gestiona reglas de negocio, validaciones, autenticación, controladores (views) y comunicación con la base de datos.	Django, DRF, MySQL client	Desarrollador backend
Base de datos (MySQL)	Almacena la información persistente del sistema (productos, usuarios, movimientos, pedidos, etc.).	MySQL Server 8.0	DBA / Administrador
Servidor de aplicaciones (Gunicorn)	Intermediario entre Django y el servidor web (procesa peticiones WSGI).	Python 3.10+, Django	DevOps
Servidor web (Nginx)	Gestiona peticiones HTTP/HTTPS, balanceo y archivos estáticos.	Gunicorn, Django	DevOps
Sistema de almacenamiento (media/static)	Maneja archivos estáticos y multimedia generados por el sistema (imágenes, reportes PDF, etc.).	Sistema de archivos / AWS S3	DevOps
Sistema de control de versiones (Git/GitHub)	Permite la gestión de código, control de versiones y despliegues automatizados (CI/CD).	GitHub Actions	Desarrollador / DevOps
Servicios de monitoreo y logs	Recolecta métricas de rendimiento y errores del sistema.	Sentry / Prometheus (opcional)	Administrador / DevOps

Artefactos

- manage.py — punto de entrada del proyecto Django.
- settings.py — configuración general del proyecto (base de datos, rutas, middleware).
- urls.py — mapeo de rutas y endpoints.
- models.py — definición de entidades y relaciones de datos.
- views.py — lógica de presentación / controladores.
- serializers.py — estructura de datos para API REST.
- requirements.txt — dependencias del proyecto.
- INVENTARIO_X_SCRIPT.sql — estructura base de datos.
- Dockerfile y docker-compose.yml — configuración de contenedores.
- static/ y media/ — archivos estáticos y multimedia.
- nginx.conf — configuración del proxy inverso.
- systemd.service *(si aplica)* — servicio de arranque en servidor.

Conexiones y flujo de comunicación

1. El **usuario** interactúa con la Interfaz web (Frontend).
2. El **Frontend** envía peticiones al Backend Django mediante la API REST.
3. El **Backend** ejecuta la lógica de negocio y realiza consultas a la Base de datos MySQL.
4. El **Servidor Gunicorn** recibe y responde solicitudes bajo el protocolo WSGI.
5. **Nginx** actúa como proxy, distribuyendo peticiones entre Gunicorn y los archivos estáticos.
6. Los registros (logs) se almacenan en un servicio de monitoreo (Sentry / Logs del servidor).
7. Los **artefactos de despliegue** (Docker, systemd) garantizan la portabilidad del sistema.

Consideraciones de diseño

- La arquitectura sigue el patrón MVC (Model–View–Controller) implementado por Django.
- Las dependencias están aisladas mediante entornos virtuales (venv).
- El sistema está preparado para despliegues escalables mediante contenedores Docker o **servidores** Linux tradicionales (Gunicorn + Nginx).

Configuración del entorno y variables de entorno (ENV)

El sistema de inventario utiliza variables de entorno para configurar parámetros sensibles y específicos de cada ambiente (desarrollo, pruebas, producción).
Estas variables se definen en un archivo oculto llamado .env que se ubica en la raíz del proyecto Django.

Ubicación y carga

El archivo .env debe estar en la misma carpeta que el archivo manage.py, y nunca debe subirse al repositorio.

El proyecto usa la librería python-dotenv o el módulo os.environ de Django para leer estas variables desde el entorno del sistema.

Variables principales

Variable	Descripción	Valor de referencia
SECRET_KEY	Clave secreta para Django.	`django-inventario-2025-key`
DEBUG	Modo depuración (True/False).	`False`
ALLOWED_HOSTS	Dominios o IPs permitidas.	`localhost,127.0.0.1,miapp.com`
DB_ENGINE	Motor de base de datos.	`django.db.backends.mysql`
DB_NAME	Nombre de la base de datos.	`inventario_db`
DB_USER	Usuario de conexión.	`django_user`
DB_PASSWORD	Contraseña del usuario de la BD.	`ContraseñaSegura123!`
DB_HOST	Host o IP del servidor de BD.	`localhost`
DB_PORT	Puerto del motor de BD.	`3306`
EMAIL_HOST	Servidor SMTP.	`smtp.gmail.com`
EMAIL_PORT	Puerto SMTP.	`587`
EMAIL_USER	Cuenta de correo del sistema.	`soporte@miapp.com`
EMAIL_PASSWORD	Contraseña o token de aplicación.	`clavecorreo2025`
AWS_S3_BUCKET	Nombre del bucket (si aplica).	`inventario-static`
SENTRY_DSN	URL de integración para reportes de errores.	`https://abc123@sentry.io/5678`

Archivo `.env`

```
# Seguridad
SECRET_KEY=django-inventario-2025-key
DEBUG=False
ALLOWED_HOSTS=localhost,127.0.0.1,miapp.com

# Base de datos MySQL
DB_ENGINE=django.db.backends.mysql
DB_NAME=inventario_db
DB_USER=django_user
DB_PASSWORD=ContraseñaSegura123!
DB_HOST=localhost
DB_PORT=3306

# Correo electrónico
EMAIL_HOST=smtp.gmail.com
EMAIL_PORT=587
EMAIL_USER=soporte@miapp.com
EMAIL_PASSWORD=clavecorreo2025

# Servicios opcionales
AWS_S3_BUCKET=inventario-static
SENTRY_DSN=https://abc123@sentry.io/5678
```

API Y RUTAS DEL SISTEMA

Rutas Web

Estas rutas permiten la interacción con el sistema desde la interfaz web del proyecto

Ruta	Método HTTP	Vista / Función	Descripción
/	GET	`views.inicio`	Página de inicio o panel principal del sistema.
/productos/	GET	`views.productos`	Muestra la lista de productos registrados.
/productos/<int:pk>/edit/	GET / POST	`views.product_edit`	Permite editar la información de un producto específico.
/productos/<int:pk>/delete/	POST	`views.product_delete`	Elimina un producto del inventario.
/api/lista/	GET	`views.lista_productos`	Devuelve la lista de productos en formato JSON (endpoint API auxiliar).
/api/crear-demo/	POST	`views.crear_producto_demo`	Crea un producto de prueba de forma rápida (endpoint auxiliar).

Rutas del Proyecto Principal (INVENTARIOX)

El archivo INVENTARIOX/urls.py orquesta todas las rutas del proyecto y registra los endpoints de la API REST:

Ruta base	Módulo asociado	Descripción
/admin/	`django.contrib.admin`	Panel de administración de Django.
/	`Producto.urls`	Redirección principal a la app de productos.
/api/	`rest_framework.routers.DefaultRouter()`	Enrutamiento automático de los endpoints REST.
/accounts/	`django.contrib.auth.urls`	Rutas integradas de autenticación (login, logout, cambio de contraseña).
/accounts/register/	`views.register`	Registro personalizado de nuevos usuarios.

Endpoints API REST

El router registrado en INVENTARIOX/urls.py gestiona los endpoints de la clase ProductoViewSet, la cual expone operaciones CRUD para la entidad Producto**.**

Endpoint	Método HTTP	Funcionalidad	Autenticación requerida
/api/productos/	GET	Lista todos los productos registrados.	Sí
/api/productos/{id}/	GET	Consulta los detalles de un producto.	Sí
/api/productos/	POST	Crea un nuevo producto.	Sí (rol admin)
/api/productos/{id}/	PUT / PATCH	Actualiza un producto existente.	Sí
/api/productos/{id}/	DELETE	Elimina un producto.	Sí (rol admin)

MIGRACIONES, BACKUPS Y PLAN DE DESPLIEGUE

Migraciones de la base de datos

Las migraciones permiten mantener sincronizado el modelo de datos del sistema con el esquema físico de la base de datos.

Pasos para ejecutar migraciones en Django:

1. Activar el entorno virtual.
2. Ejecutar Python manage.py makemigrations para generar los archivos de migración.
3. Ejecutar Python manage.py migrate para aplicar los cambios a la base de datos.
4. Verificar con SHOW TABLES; en MySQL que las tablas se actualizaron correctamente.
5. Documentar cada migración en el repositorio Git con un commit identificable

Copias de seguridad (Backups)

Los backups garantizan la recuperación de la información en caso de fallos, errores humanos o corrupción de datos.

Política de respaldo recomendada:

- **Frecuencia:** diaria (automática con tarea programada).
- **Tipo:** copia completa de la base de datos (mysqldump) y respaldo incremental semanal del código fuente.
- **Retención:** mantener copias de los últimos 30 días.
- **Ubicación:** servidor seguro y unidad externa o servicio en la nube (Google Drive, AWS S3, Azure Blob).
- **Verificación:** restaurar y validar una copia cada mes para asegurar integridad.

Plan de despliegue (Deployment Plan)

El despliegue define los pasos para trasladar la aplicación del entorno de desarrollo al entorno de producción de forma controlada.

Etapas del plan:

Etapa	Actividad	Responsable	Herramienta	Resultado esperado
1	Crear backup completo de base de datos y código fuente	DBA / DevOps	mysqldump / Git	Copia de seguridad almacenada
2	Clonar proyecto y verificar dependencias	Dev	Git / pip	Entorno listo
3	Ejecutar migraciones	Dev	Django CLI	BD actualizada
4	Compilar assets estáticos (Tailwind)	Dev	npm build	Archivos CSS/JS generados
5	Configurar servidor Gunicorn + Nginx	DevOps	Linux / systemctl	Servicio en ejecución
6	Probar endpoints y vistas principales	QA	Postman / Navegador	Validación funcional
7	Monitorear logs post-despliegue (72h)	DevOps	journalctl / Sentry	Confirmación de estabilidad

Pruebas, CI/CD y calidad de código

Pruebas recomendadas: unitarias (pytest/django.test), integración (endpoints), pruebas de contrato (OpenAPI), pruebas e2e ligeras (Playwright/Cypress).

CI/CD: pipeline que ejecute linting (flake8/black), tests unitarios, construcción assets (npm build), y despliegue condicional.

Cobertura: objetivo mínimo 70% para módulos críticos.

Registro de errores y códigos

Convenciones: Exponer mensajes de error técnicos en logs; respuestas al cliente con códigos claros. Ejemplo de códigos HTTP y significado en app:

- 400 Bad Request — validación fallida.
- 401 Unauthorized — credenciales inválidas.
- 403 Forbidden — falta permisos.
- 404 Not Found — recurso no existente.
- 409 Conflict — operación en conflicto (ej. duplicado).
- 500 Internal — error no controlado; revisar logs (Sentry).

Pasos rápidos para debug de un fallo en producción:

1. Reproducir en staging con dataset representativo.
2. Revisar logs (Gunicorn / Nginx / DB).
3. Ejecutar querys SQL de verificación.
4. Si es incidente crítico: rollback a backup y abrir MR para corrección.

BIBLIOGRAFIA

Django Software Foundation. (2024). *Django documentation (version 4.x): The web framework for perfectionists with deadlines.* Recuperado de https://docs.djangoproject.com/en/4.2/

MySQL. (2024). *MySQL 8.0 Reference Manual.* Oracle Corporation. Recuperado de https://dev.mysql.com/doc/refman/8.0/en/

Tailwind Labs. (2024). *Tailwind CSS Documentation (version 3.x).* Recuperado de https://tailwindcss.com/docs

International Organization for Standardization (ISO). (2006). *ISO/IEC/IEEE 14764:2006 – Software Engineering — Software Life Cycle Processes — Maintenance.* Recuperado de https://www.iso.org/standard/39064.html

Red Hat. (2024, mayo). *Cómo realizar una migración segura de base de datos y planificar respaldos efectivos.* Blog de Red Hat. Recuperado de https://www.redhat.com/es/blog/backup-y-migracion-de-datos

Departamento Nacional de Planeación (DNP). (2022). *Guía para la elaboración de manual técnico y de operaciones de los sistemas de información.* Recuperado de https://bit.ly/3I1439H

SENA. (s. f.). *Actividades de documentación y entrega de software en procesos de implantación*. Zajuna. Recuperado de https://zajuna.sena.edu.co/Repositorio/Titulada/institution/SENA/Tecnologia/228118/Contenido/OVA/CF46/index.html#/