

# UNIVERSIDAD PRIVADA DOMINGO SAVIO

INGENIERÍA DE SISTEMAS



Nombres: Cristian Daniel Tito Chince

Angel Andres Rocha Gomez

Materia: Programación IV

Docente: Franklin Riabani Mercado Flores

Cochabamba – Bolivia

Junio 16 de 2025

# Índice

1.	Introducción .....	3
2.	Objetivos .....	3
2.1	Objetivo General .....	3
2.2	Objetivos Específicos.....	3
3.	Alcance del Proyecto .....	3
4.	Tecnologías Utilizadas .....	3
5.	Interfaz de inicio de sesión del sistema .....	4
5.1	Panel principal del sistema (dashboard) .....	4
5.2	Gestión de productos: listado y registro de nuevos artículos.....	4
5.3	Gestión de usuarios y roles .....	5
5.4	Panel de Empleado.....	5
5.5	Protección de Rutas .....	5
5.6	Montar Contenedores en Docker .....	5
5.7	Código Dashboard .....	6
5.8	Conexión Middleware con el frontend y banckend .....	6
5.9	Código Autenticación .....	6
6.	Link Github .....	7
7.	Conclusión .....	8

## 1. Introducción

En la actualidad, el manejo eficiente del inventario es un pilar fundamental para el éxito de cualquier empresa del sector comercial, especialmente en empresas importadoras de productos como electrodomésticos, donde el control de stock, ubicación de productos y trazabilidad son críticos. Con el objetivo de mejorar la eficiencia operativa, se ha decidido desarrollar un sistema de gestión de inventario que permita administrar de forma eficaz los productos en cada una de las sucursales de la empresa.

Este proyecto está siendo desarrollado como parte de una iniciativa tecnológica para digitalizar y automatizar procesos internos, utilizando FastAPI como framework para el backend y una interfaz nativa para el frontend que facilitará la interacción del personal con el sistema.

## 2. Objetivos

### 2.1 Objetivo General

Desarrollar un sistema de gestión de inventario que permita controlar y supervisar el stock de productos en cada sucursal de una empresa importadora de electrodomésticos.

### 2.2 Objetivos Específicos

- Diseñar e implementar una API RESTful utilizando FastAPI que gestione productos, sucursales y movimientos de inventario.
- Registrar el ingreso y salida de productos en tiempo real.
- Asociar productos a diferentes sucursales de forma independiente.
- Garantizar la trazabilidad de cada artículo mediante un sistema de historial.
- Desarrollar una interfaz de usuario amigable, funcional y nativa que permita al personal gestionar fácilmente el inventario.

## 3. Alcance del Proyecto

El sistema permitirá:

- Registrar nuevos productos y categorías.
- Asociar productos a sucursales específicas.
- Registrar entradas y salidas de inventario.
- Consultar el stock disponible por producto y sucursal.
- Generar reportes básicos de movimientos por fecha y por producto.

## 4. Tecnologías Utilizadas

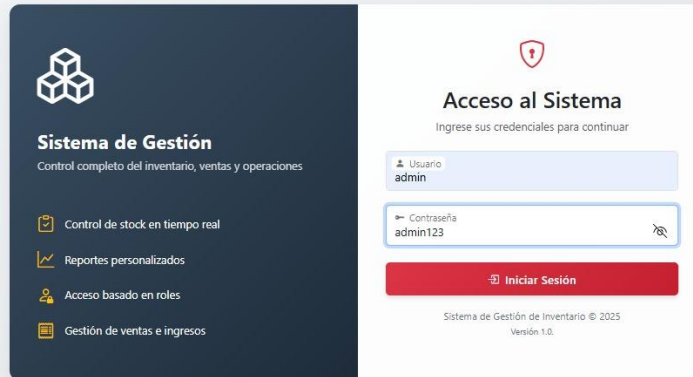
### Backend:

- Lenguaje: Python
- Framework: FastAPI
- Base de datos: MySQL
- ORM: SQLAlchemy
- Autenticación: JWT
- Documentación automática de la API: Swagger UI (integrado en FastAPI)

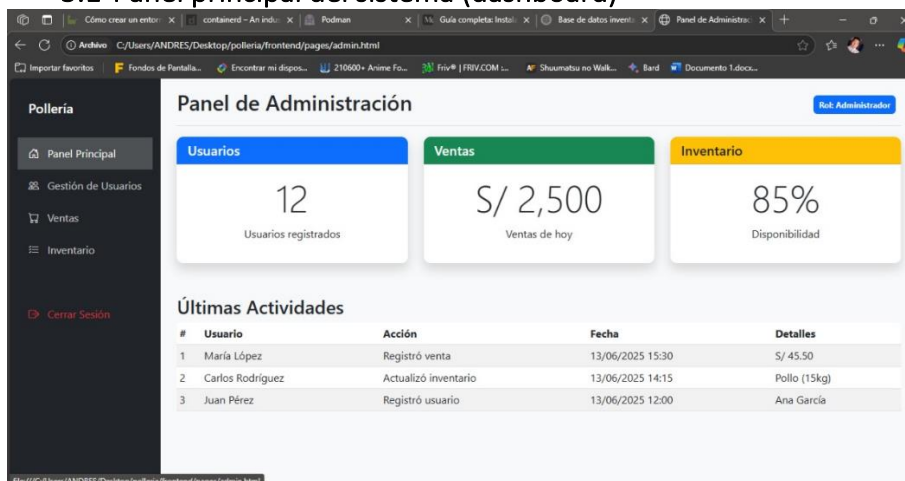
## Frontend:

- Desarrollo nativo (HTML,CSS,JS)
- Conexión a la API a través de peticiones HTTP REST
- Interfaces diseñadas para usabilidad y rendimiento

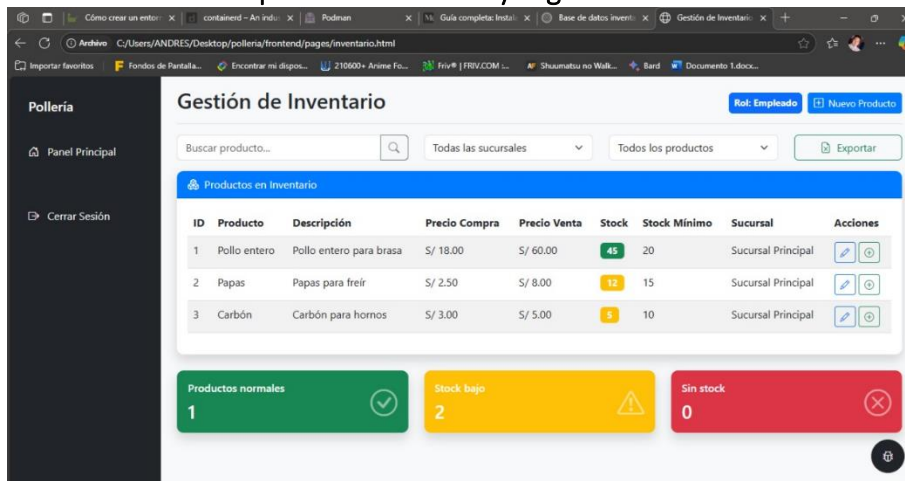
### 5. Interfaz de inicio de sesión del sistema



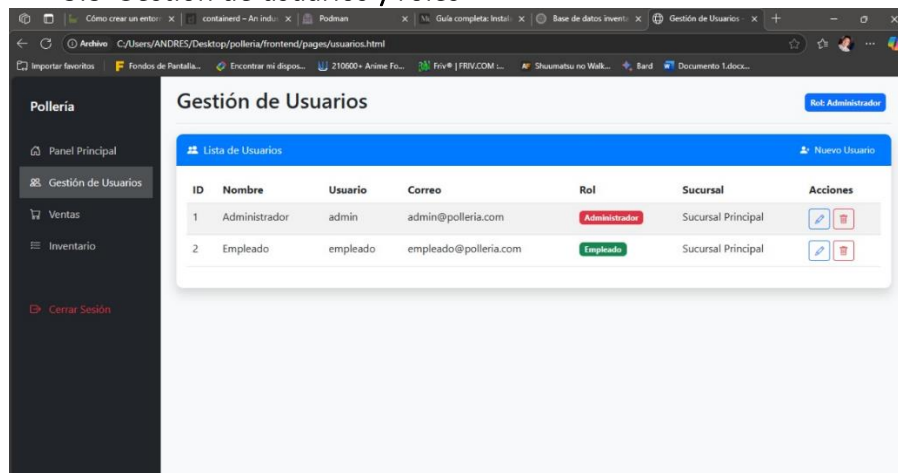
### 5.1 Panel principal del sistema (dashboard)



### 5.2 Gestión de productos: listado y registro de nuevos artículos



## 5.3 Gestión de usuarios y roles



Pollería

Panel Principal

Gestión de Usuarios

Ventas

Inventario

Cerrar Sesión

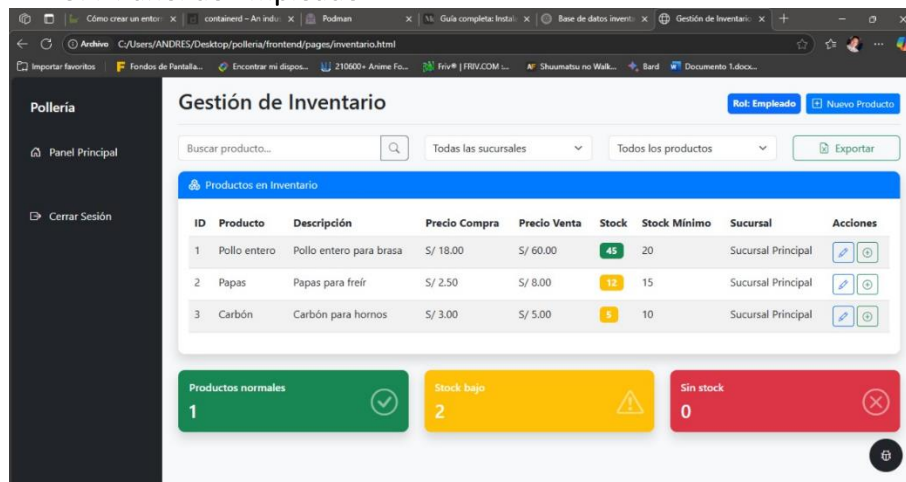
### Gestión de Usuarios

Rol: Administrador

Nuevo Usuario

ID	Nombre	Usuario	Correo	Rol	Sucursal	Acciones
1	Administrador	admin	admin@polleria.com	Administrador	Sucursal Principal	<a href="#">Editar</a> <a href="#">Eliminar</a>
2	Empleado	empleado	empleado@polleria.com	Empleado	Sucursal Principal	<a href="#">Editar</a> <a href="#">Eliminar</a>

## 5.4 Panel de Empleado



Pollería

Panel Principal

Cerrar Sesión

### Gestión de Inventario

Rol: Empleado

Nuevo Producto

Buscar producto...

Todas las sucursales

Todos los productos

Exportar

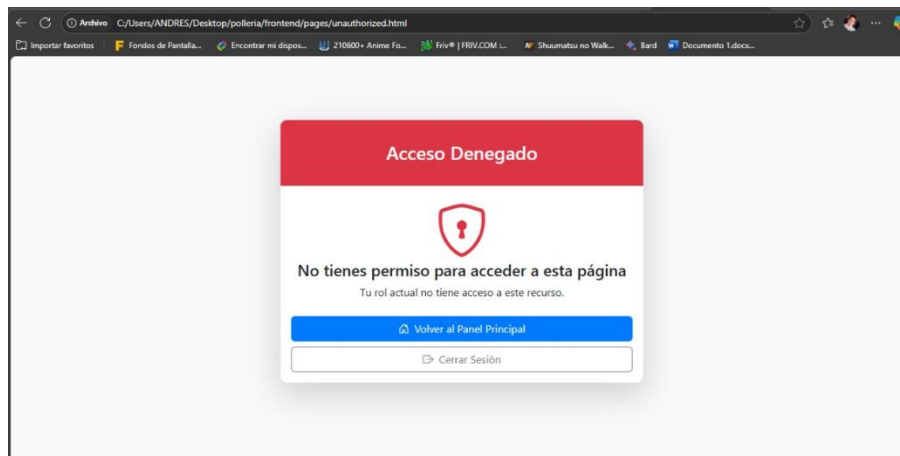
ID	Producto	Descripción	Precio Compra	Precio Venta	Stock	Stock Mínimo	Sucursal	Acciones
1	Pollo entero	Pollo entero para brasa	S/ 18.00	S/ 60.00	45	20	Sucursal Principal	<a href="#">Editar</a> <a href="#">Eliminar</a>
2	Papas	Papas para freír	S/ 2.50	S/ 8.00	12	15	Sucursal Principal	<a href="#">Editar</a> <a href="#">Eliminar</a>
3	Carbón	Carbón para hornos	S/ 3.00	S/ 5.00	5	10	Sucursal Principal	<a href="#">Editar</a> <a href="#">Eliminar</a>

Productos normales: 1

Stock bajo: 2

Sin stock: 0

## 5.5 Protección de Rutas



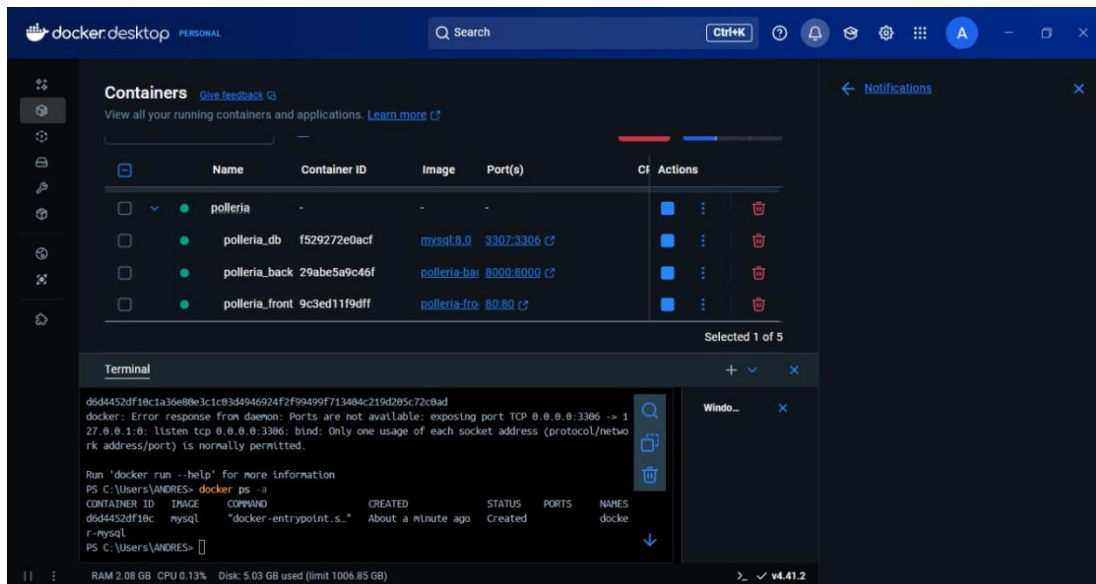
Acceso Denegado

No tienes permiso para acceder a esta página

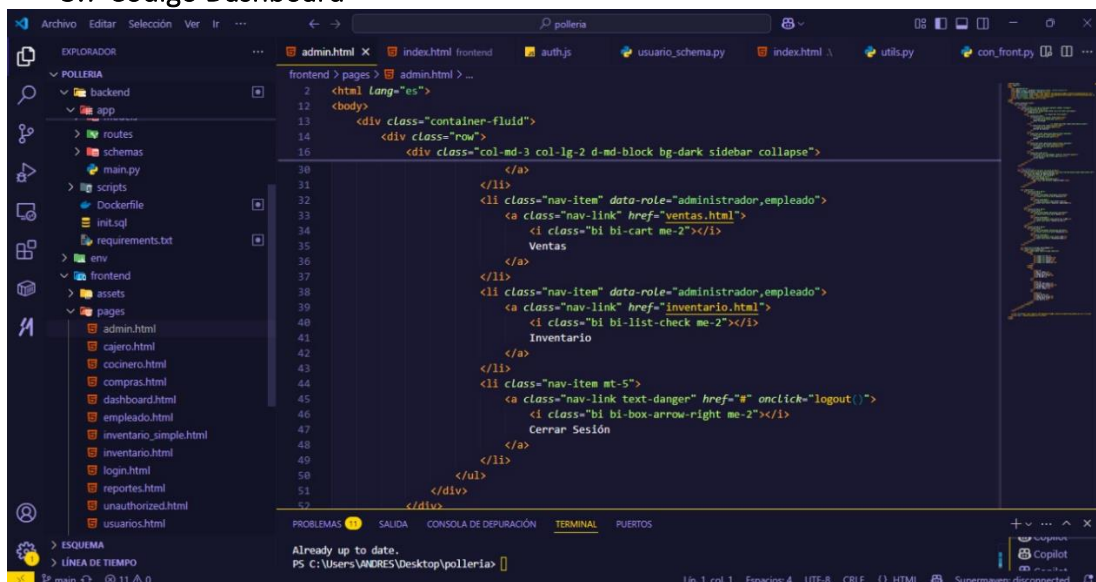
Tu rol actual no tiene acceso a este recurso.

Volver al Panel Principal

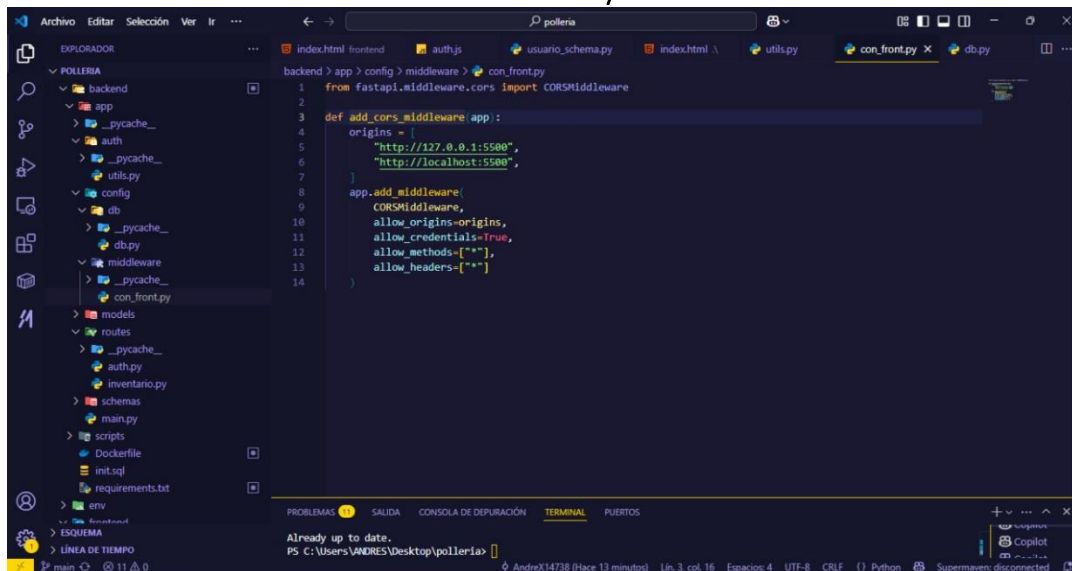
Cerrar Sesión



## 5.7 Código Dashboard



## 5.8 Conexión Middleware con el frontend y backend



## 5.9 Código Autenticación

```
def login(form_data: OAuth2PasswordRequestForm = Depends(), db: Session = Depends(get_db)):
    return {
        "access_token": access_token,
        "token_type": "bearer",
        "user_id": user.id,
        "rol": rol.nombre
    }

@router.get("/usuarios/me", response_model=UsuarioDetalleResponse)
def get_user_me(current_user: Usuario = Depends(get_current_user), db: Session = Depends(get_db)):
    """Endpoint para obtener información del usuario actual"""
    # Conectar el usuario con sus relaciones
    user = db.query(Usuario).filter(Usuario.id == current_user.id).first()
    return user

@router.get("/admin", response_model=UsuarioResponse)
def admin_only_current_user: Usuario = Depends(has_role(["administrador"])):
    """Endpoint de ejemplo que requiere rol de administrador"""
    return current_user

@router.get("/empleado", response_model=UsuarioResponse)
def empleado_admin(current_user: Usuario = Depends(has_role(["empleado", "administrador"])):
    """Endpoint de ejemplo que requiere rol de empleado o administrador"""
    return current_user
```

```
def register(usuario: UsuarioCreate, db: Session = Depends(get_db), current_user: Usuario = Depends(has_role(["administrador"])):
    """Endpoint para registrar un nuevo usuario (solo administradores)"""
    # Verificar si ya existe un usuario con el mismo email o username
    existe_email = db.query(Usuario).filter(Usuario.correo == usuario.correo).first()
    existe_username = db.query(Usuario).filter(Usuario.username == usuario.username).first()

    if existe_email:
        raise HTTPException(
            status_code=status.HTTP_400_BAD_REQUEST,
            detail="El correo ya está registrado"
        )
    if existe_username:
        raise HTTPException(
            status_code=status.HTTP_400_BAD_REQUEST,
            detail="El username ya está registrado"
        )

    # Crear el token JWT
    access_token = create_access_token(
        data={"sub": str(usuario.id)}
    )

    # Retornar el token con información del usuario
    return {
        "access_token": access_token,
        "token_type": "bearer",
    }
```

```
from fastapi import APIRouter, Depends, HTTPException, status
from fastapi.security import OAuth2PasswordRequestForm
from sqlalchemy.orm import Session
from datetime import timedelta

from ..schemas.esquemas import Token, UsuarioCreate, UsuarioResponse, UserLogin, UsuarioDetalleResponse
from ..models.modelos import Usuario, Rol
from ..auth.utils import create_access_token, get_password_hash, verify_password, get_current_user, has_role
from ..config.db import get_db

router = APIRouter(tags=["Autenticación"])

@router.post("/registro", response_model=UsuarioResponse)
def register(usuario: UsuarioCreate, db: Session = Depends(get_db), current_user: Usuario = Depends(has_role(["administrador"])):
    """Endpoint para registrar un nuevo usuario (solo administradores)"""
    # Verificar si ya existe un usuario con el mismo email o username
    existe_email = db.query(Usuario).filter(Usuario.correo == usuario.correo).first()
    existe_username = db.query(Usuario).filter(Usuario.username == usuario.username).first()

    if existe_email:
        raise HTTPException(
            status_code=status.HTTP_400_BAD_REQUEST,
            detail="El correo ya está registrado"
        )
    if existe_username:
        raise HTTPException(
            status_code=status.HTTP_400_BAD_REQUEST,
            detail="El username ya está registrado"
        )

    # Crear el token JWT
    access_token = create_access_token(
        data={"sub": str(usuario.id)}
    )

    # Retornar el token con información del usuario
    return {
        "access_token": access_token,
        "token_type": "bearer",
    }
```

6. Link Github



<https://github.com/AndreX14738/polleria.git>

## 7. Conclusión

El sistema de gestión de inventario propuesto busca resolver un problema real y cotidiano en empresas de distribución de productos físicos. Al usar herramientas modernas como FastAPI y una interfaz nativa, se busca garantizar un sistema eficiente, escalable y fácil de usar. Con su implementación, se espera una mejora significativa en la organización, control y análisis del inventario de cada sucursal.