ESCUELA DE CIENCIAS EXACTAS E INGENIERÍA
CIENCIAS DE LA COMPUTACIÓN E INTELIGENCIA ARTIFICIAL

Arquitectura de Software

DOCUMENTACIÓN DE PROYECTO PARA AVÍCOLA JS

DANIEL SANTIAGO VARELA GUERRERO
GUSTAVO TAKASHI CABRERA ROSALES
ANDRES RICARDO REY AGUDELO
TOMAS DE ANDREIS ROJAS

BOGOTA D.C

14/03/2025

Resumen

El presente documento describe el diseño e implementación de un sistema de gestión de ventas y productos, desarrollado con un backend en FastAPI, frontend en Angular y desplegado utilizando Docker. El sistema permite la gestión de usuarios, productos, ventas y clientes, ofreciendo una arquitectura escalable y modular. Se detalla el estado del arte en sistemas de gestión basados en microservicios, los objetivos del proyecto, la arquitectura propuesta y las tecnologías utilizadas. Además, se justifica la viabilidad del sistema y se identifican los retos técnicos asociados a su implementación.

1. Introducción

1.1 Visión del Proyecto

Avícolajs busca modernizar sus procesos de gestión integrando ventas, gastos y producción en un sistema de información avanzado. En esta primera fase, nos enfocaremos en optimizar la gestión de la producción, asegurando una transición fluida desde métodos tradicionales a soluciones tecnológicas. Posteriormente, en una segunda fase, ampliaremos la integración a todos los procesos empresariales, consolidando la información en un sistema centralizado.

1.2 Justificación

El uso de FastAPI como framework para el backend permite desarrollar una API rápida y eficiente, ideal para sistemas que requieren alta disponibilidad y bajo tiempo de respuesta. La integración con Docker facilita el despliegue en diferentes entornos, garantizando consistencia y portabilidad.

Este sistema ofrece los siguientes beneficios:

- Escalabilidad: La arquitectura basada en microservicios permite agregar nuevas funcionalidades sin afectar el sistema existente.
- Seguridad: Se implementan mecanismos de autenticación y autorización para proteger los datos de usuarios y transacciones.
- Portabilidad: El uso de Docker permite desplegar el sistema en cualquier entorno que soporte contenedores.
- Eficiencia: FastAPI garantiza un alto rendimiento en el procesamiento de solicitudes.

2. Objetivo

2.1 Objetivo General

El objetivo es dejar atrás el uso de Excel y facturas físicas, adoptando tecnologías de vanguardia que permitan una gestión más eficiente y efectiva. Con esta transición, aspiramos a mejorar la precisión de nuestros datos, la rapidez en la toma de decisiones y la capacidad de crecimiento continuo como empresa.

2.2 Objetivo Específico

1. **Análisis de Requerimientos:** Identificar las funcionalidades necesarias para la gestión de usuarios, productos, ventas y clientes.

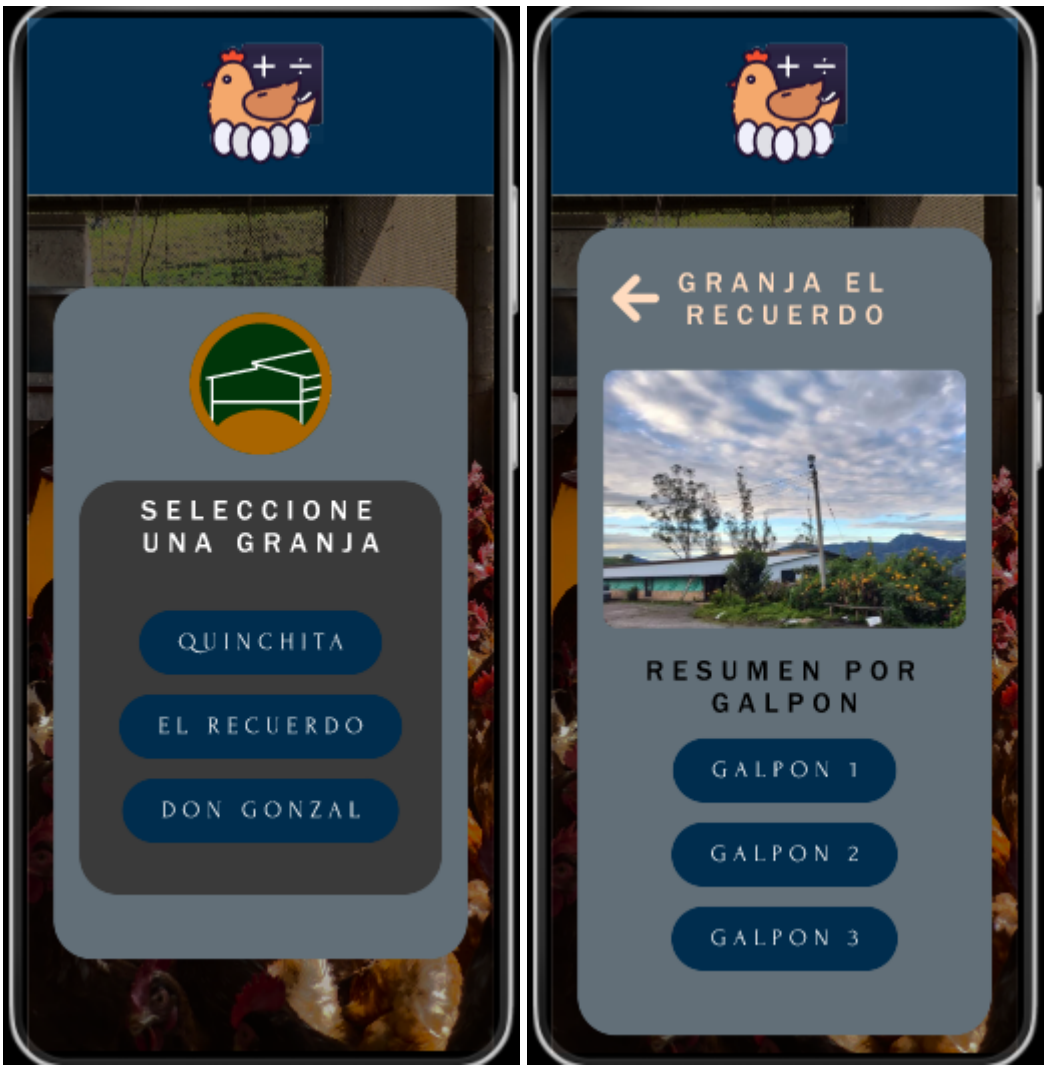
2. **Diseño de la Arquitectura:** Definir la estructura del sistema, incluyendo los modelos de datos, endpoints de la API y la integración con Docker.

3. **Implementación del Backend:** Desarrollar el backend en FastAPI, implementando autenticación, gestión de productos y registro de ventas.

4. **Despliegue con Docker:** Configurar contenedores para el backend y la base de datos, garantizando un despliegue consistente.

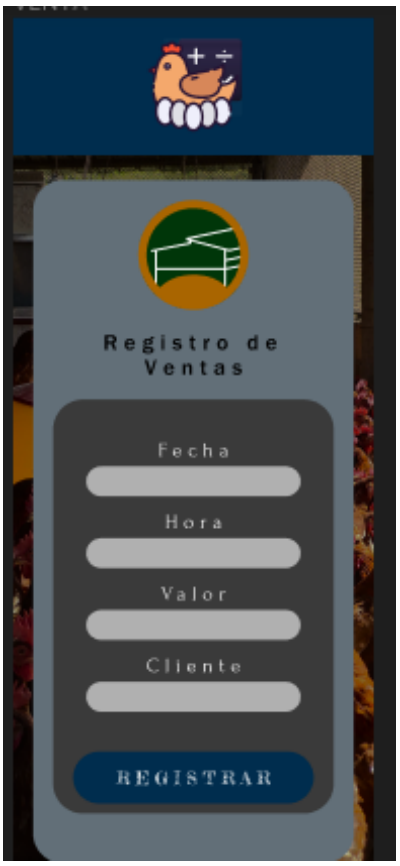
5. **Pruebas y Validación:** Realizar pruebas de funcionalidad, rendimiento y seguridad para asegurar la calidad del sistema.

3. Generación de Layouts (diseño y vista estimada del proyecto)


SELECCIONE UNA GRANJA
QUINCHITA
EL RECUERDO
DON GONZAL
GRANJA EL RECUERDO
RESUMEN POR GALPON
GALPON 1
GALPON 2
GALPON 3


¿Qué desea hacer?
Visualización de datos
Ventas
Gastos


GRANJA EL RECUERDO
GALPON 1
Ventas Semana
<-4/03-10/03->
Fecha Valor Cliente
4-mar-24 $3.000.000 Carlos
4-mar-24 $3.567.800 Andrés
4-mar-24 $2.560.388 Jose
TOTAL $9.128.188
Gastos Semana
<-4/03-10/03->



Metas:

Modernización y Optimización de Procesos Nuestro principal objetivo es modernizar y optimizar los procesos internos de Avícola JS, eliminando la dependencia de herramientas obsoletas como hojas de cálculo de Excel y facturas físicas. Implementaremos un sistema de información avanzado que integrará todas las áreas clave de la empresa, desde ventas y gastos hasta producción. Este sistema permitirá una gestión centralizada y eficiente de los datos, mejorando la precisión, reduciendo el tiempo de procesamiento y aumentando la transparencia en todas las operaciones empresariales.

Diseño de la Arquitectura
4.1 Componentes del Sistema

El sistema se compone de los siguientes módulos:

Módulo de Autenticación: Gestiona el registro y autenticación de usuarios.

Módulo de Productos: Permite la creación, actualización y eliminación de productos.

Módulo de Ventas: Registra transacciones y asocia productos vendidos.

Módulo de Clientes: Gestiona la información de los clientes.

4.2 Tecnologías Utilizadas

Backend: FastAPI (Python).

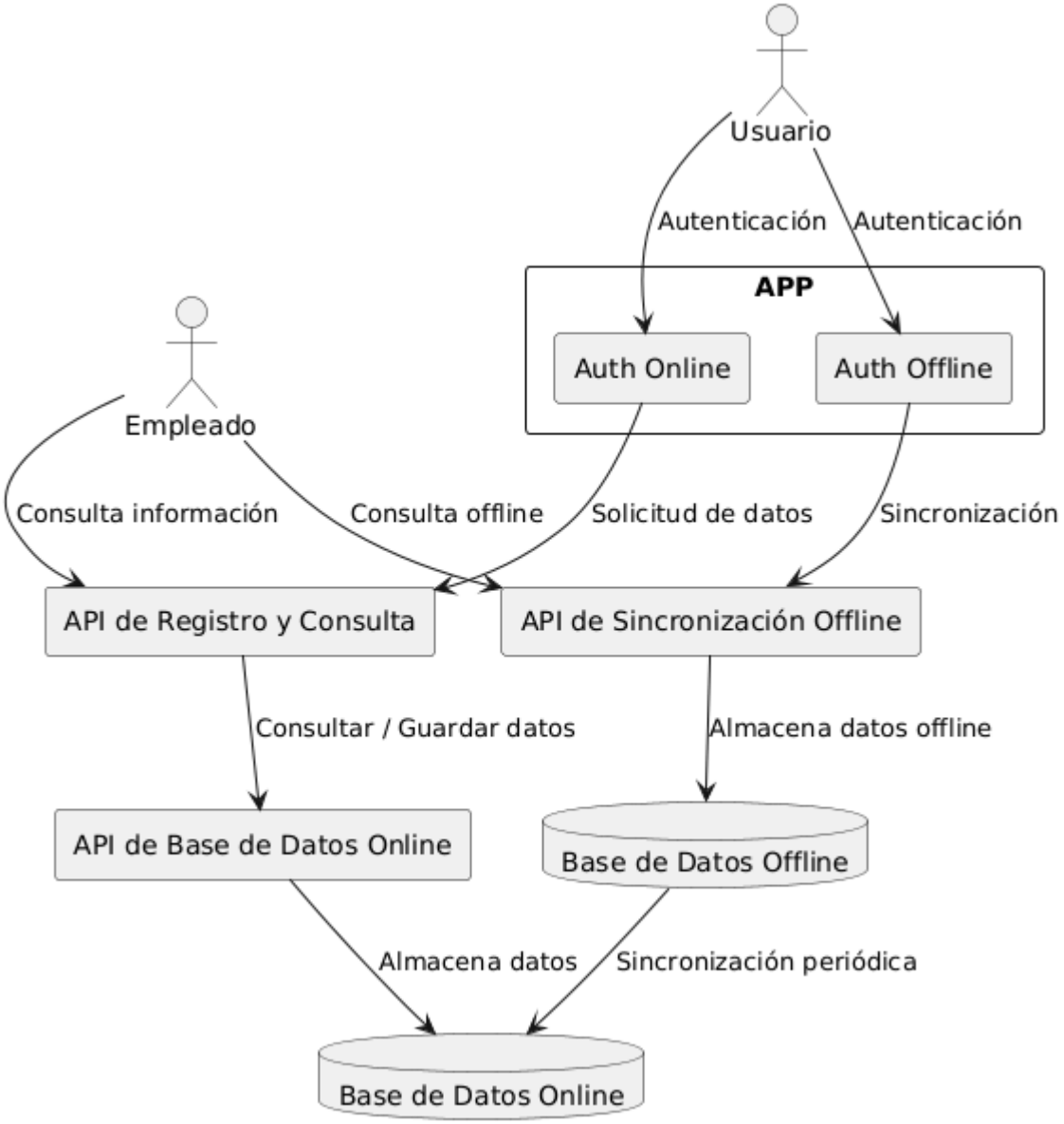
Base de Datos: PostgreSQL.

Autenticación: JWT.

Contenedores: Docker.

Gestión de Dependencias: Poetry.

4.3 Diagrama de Arquitectura



Implementación

5.1 Backend en FastAPI

El backend está desarrollado en FastAPI y ofrece los siguientes endpoints:

Usuarios:

- **POST /users/register/: Registra un nuevo usuario.**
- **POST /users/login/: Autentica a un usuario.**
- **GET /users/: Obtiene una lista de usuarios.**

Productos:

- **POST /{user_id}/productos/create/:** Crea un nuevo producto.
- **GET /productos/:** Obtiene una lista de productos.
- **DELETE /{user_id}/productos/{producto_id}:** Elimina un producto.

Ventas:

- **POST /{user_id}/ventas/create/:** Registra una nueva venta.
- **GET /{user_id}/ventas/:** Obtiene una lista de ventas por usuario.

5.2 Dockerización

El sistema se despliega utilizando Docker. A continuación, se describe la configuración:

Dockerfile: Define la imagen del backend en FastAPI.

docker-compose.yml: Configura los servicios del backend y la base de datos.

```yaml
version: '3.8'
services:
  backend:
    build: .
    ports:
      - "8000:8000"
    environment:
      - DATABASE_URL=postgresql://user:password@db:5432/mydatabase
    depends_on:
      - db

  db:
    image: postgres:13
    environment:
      POSTGRES_USER: user
      POSTGRES_PASSWORD: password
      POSTGRES_DB: mydatabase
    ports:
      - "5432:5432"
```