MANUAL TECNICO

REPOSITORIO DIGITAL DEL SISTEMA DE GESTION DE TAREAS PARA LA EMPRESA AGRICOLA FLORCANELA

Manual Tecnico

**Para:**

Sistema AgroTask

Versión 1.0

**Preparado por:**

NOMBRE

**Fecha:** Guayaquil – Enero 2026

Table of Contents

[Introducción 4](#_Toc220941982)

[Informacion general del proyecto 4](#_Toc220941983)

[Descripcion general del sistema 4](#_Toc220941984)

[Requisitos del sistema 5](#_Toc220941985)

[Arquitectura del sistema 6](#_Toc220941986)

[**Componentes principales:** 6](#_Toc220941987)

[Estructura del proyecto 6](#_Toc220941988)

[Base de datos 8](#_Toc220941989)

[Backend (Servidor  / API) 8](#_Toc220941990)

[Frontend 9](#_Toc220941991)

[Seguridad del sistema 10](#_Toc220941992)

[Instalacion y confirguracion 11](#_Toc220941993)

[Despliegue del sistema 11](#_Toc220941994)

[Prueba tecnicas 12](#_Toc220941995)

[Mantenimiento y soporte 12](#_Toc220941996)

[Glosario de terminos 12](#_Toc220941997)

[Anexos 13](#_Toc220941998)

# Introducción

El presente **Manual Técnico** tiene como propósito describir de forma clara y detallada la **arquitectura**, **estructura**, **funcionamiento** y **configuración** del sistema desarrollado, proporcionando la información técnica necesaria para facilitar su correcta comprensión, mantenimiento y posibles ampliaciones futuras.

Este documento está dirigido principalmente a **desarrolladores**, **administradores de sistemas** y **personal técnico**, que requieran realizar tareas de instalación, configuración, operación, mantenimiento o mejora del sistema. De igual manera, constituye un **soporte documental** del proyecto desarrollado dentro del ámbito académico, evidenciando el diseño y la implementación de la solución propuesta.

El sistema está conformado por una **aplicación frontend**, desarrollada utilizando tecnologías web, y un **backend** que expone una **API**, la cual permite la gestión de usuarios, la autenticación mediante correo electrónico y contraseña, así como la interacción con la **base de datos** para el almacenamiento y recuperación de la información.

## Informacion general del proyecto

* + **Nombre del sistema:** Sistema Web de Gestión con Autenticación
  + **Tipo de proyecto:** Académico
  + **Arquitectura:** Cliente – Servidor
  + **Modalidad:** Aplicación Web
  + **Autor:** Rogger Alfredo Lindao Pérez
  + **Institución:** Universidad Israel
  + **Fecha de elaboración:** 2026
  + **Versión del sistema:** 1.0

## Descripcion general del sistema

El sistema desarrollado corresponde a una **aplicación web** orientada a la **gestión de usuarios**, la cual implementa un mecanismo de **autenticación mediante correo electrónico y contraseña**, con el objetivo de garantizar un **acceso controlado y seguro** a las funcionalidades disponibles.

El proyecto se encuentra estructurado en **dos componentes principales**, claramente diferenciados según su responsabilidad dentro de la arquitectura del sistema:

* **Frontend:** Componente encargado de la **interfaz gráfica de usuario**, la presentación visual del sistema, el manejo de los **formularios de acceso** y la comunicación directa con la **API** del backend.
* **Backend:** Componente responsable de la **lógica de negocio**, la **validación de credenciales**, la **gestión de usuarios**y la interacción con la **base de datos**, asegurando la integridad y consistencia de la información.

El sistema ha sido diseñado bajo principios de **escalabilidad**, **modularidad** y **facilidad de mantenimiento**, aplicando **buenas prácticas de desarrollo de software**, lo que permite su evolución y adaptación a futuros requerimientos.

## Requisitos del sistema

* + Requisitos de hardware

Servidor (backend)

* Procesador: 1 núcleo o superior
* Memoria RAM: mínimo 1 GB
* Almacenamiento: mínimo 5 GB disponibles
* Conectividad a Internet

Cliente (Usuario final)

* Computadora o dispositivo movil
* Navegador web actualizado
  + Requisitos de software

Backend:

* Sistema operativo: Linux, Windows o macOS
* Lenguaje de programación: Python
* Framework backend: FastAPI
* Gestor de dependencias: pip / entorno virtual
* Base de datos: MySQL
* Herramientas adicionales:
  + Git
  + Postman (para pruebas de API)

Frontend:

* HTML5
* CSS3
* JavaScript
* Navegadores compatibles:
  + Google Chrome
  + Mozilla Firefox
  + Microsoft Edge

## Arquitectura del sistema

El sistema utiliza una **arquitectura cliente-servidor**, donde el frontend actúa como cliente consumiendo los servicios expuestos por el backend a través de una **API REST**.

**Componentes principales:**

* **Cliente (Frontend):**
  + Interfaz gráfica
  + Formularios de login
  + Validaciones básicas
  + Consumo de la API mediante peticiones HTTP
* **Servidor (Backend):**
  + Lógica de negocio
  + Autenticación y validación de usuarios
  + Gestión de datos
  + Respuesta en formato JSON
* **Base de datos:**
  + Almacenamiento de usuarios y credenciales
  + Persistencia de la información

La comunicación entre el frontend y el backend se realiza mediante el protocolo **HTTP**, utilizando métodos como GET y POST.

## Estructura del proyecto

El proyecto se encuentra organizado de forma modular, separando claramente las responsabilidades del **frontend** y del **backend**, lo que facilita el mantenimiento, la escalabilidad y la comprensión del sistema.

* + Estructura general

/proyecto-raiz

│

├── backend

│ ├── app

│ │ ├── main.py

│ │ ├── routers

│ │ ├── controllers

│ │ ├── models

│ │ ├── schemas

│ │ └── config

│ ├── requirements.txt

│ └── .env

│

├── frontend

│ ├── css

│ ├── js

│ ├── assets

│ ├── pages

│ └── index.html

│

└── README.md

* + Descripcion de carpetas

**Backend**

* **app/**: Contiene el núcleo de la aplicación backend.
* **routers/**: Define las rutas o endpoints de la API.
* **controllers/**: Implementa la lógica de negocio.
* **models/**: Define los modelos de base de datos.
* **schemas/**: Estructuras de datos para validaciones y respuestas.
* **config/**: Configuraciones generales del sistema.
* **requirements.txt**: Lista de dependencias del proyecto.
* **.env**: Variables de entorno (credenciales, configuración).

**Frontend**

* **css/**: Hojas de estilo del sistema.
* **js/**: Lógica JavaScript (login, consumo de API).
* **assets/**: Recursos gráficos.
* **pages/**: Vistas del sistema.
* **index.html**: Página principal del sistema.

## Base de datos

El sistema utiliza una **base de datos relacional**, encargada de almacenar la información de los usuarios y sus credenciales de acceso.

Base de datos:

* **Tipo:** Relacional
* **Motor:** MySQL
  + Modelo de datos

La base de datos está diseñada para garantizar la integridad, seguridad y persistencia de la información.

|  |  |  |
| --- | --- | --- |
| Campo | Tipo dato | Descripcion |
| id\_usuario | Int(PK) | Identificador unico del usuario |
| Nombre | VARCHAR | Nombre de usuario |
| Correo | VARCHAR | Correo electronico |
| Password | VARCHAR | Constraseña cifrada |
| Fecha\_creado | DATETIME | Fecha de registro |

Nota: Las contraseñas no se almacenan en texto plano, sino mediante técnicas de cifrado para garantizar la seguridad de la información.

## Backend (Servidor  / API)

El backend es el encargado de procesar las solicitudes del frontend, aplicar la lógica de negocio y comunicarse con la base de datos.

* + Tecnologias utilizadas
    - Lenguaje: Python
    - Framework: FastAPI
    - Base de datos: MySQL
    - Formato de intercambio de datos: JSON
  + Configuracion del backend

Antes de ejecutar el ervidor es necesario:

1. Crear un entorno virtual
2. Instalar las dependencias desde dependencias.txt
3. Configurar las variables de entorno en el archivo .env
4. Ejecutar el servidor backend
   * Endpoints principales

Autenticacion

|  |  |  |
| --- | --- | --- |
| Metodo | Endpoint | Descripcion |
| POST | /login | Autenticacion de usuario |
| POST | /usuarios | Registro de nuevo usuario |
| GET | /usuarios | Obtener usuario |
| PUT | /usuarios | Editar usuarios |
| DELETE | /usuarios | Eliminar usuario |
| GET | /menu | Obtener los permiss segun el usuario |
| GET | /roles | Obtene los roles |

Ejemplo de solicitud de login

{

"correo": "adminCorreo@gmail.com",

"password": "Admin1234"

}

Ejemplo de Respuesta

{

"access\_token": "Token generado",

"token\_type": "bearer"

}

## Frontend

El frontend corresponde a la interfaz gráfica del sistema y permite al usuario interactuar con las funcionalidades disponibles.

* + Tecnologias utilizadas
    - HTML5
    - CSS3
    - JavaScript
  + Funcionalidades principales
    - Formulario de inicio de sesion(login).
    - Validacion basica de campos.
    - Envio de credenciales al backend.
    - Manejo de respuestas del servidor.
    - Mensajes de error o confirmacion.
  + Flujo de autenticacion
    - El usuario ingresa su correo y contraseña.
    - El frontend valida que los campos no esten vacíos.
    - Se envia una peticion HTTP al backend.
    - El backend valida las credenciales.
    - Se retorna una respuesta indicando exito o error.
    - El frontend muestra el resultado al usuario.

## Seguridad del sistema

La **seguridad del sistema** constituye un aspecto esencial en el desarrollo de aplicaciones web, particularmente en aquellas que gestionan **información sensible**, como las **credenciales de los usuarios**, ya que garantiza la protección de los datos y la confiabilidad del sistema.

* + Autenticacion

El sistema implementa un **mecanismo de autenticación** basado en el uso de **correo electrónico y contraseña**, mediante el cual se verifican las credenciales ingresadas por el usuario, comparándolas con los **registros previamente almacenados en la base de datos**, con el fin de garantizar un acceso seguro y controlado.

* + Manejo de contraseñas
    - Las contraseñas se almacenan de forma cifrada.
    - No se permite el almacenamiento de contraseñas en texto plano.
    - Se aplican validaciones minimas de seguridad en el ingreo de datos.
  + Validaciones de datos
    - Validaciones de campos obligatorios de frontend.
    - Validaciones e estructura de datos en backend.
    - Control de respuestas para evitar exposicion innecesaria de informacion.
  + Control de acceso

El acceso a las funcionalidades del sistema está restringido únicamente a usuarios autenticados, garantizando un uso controlado de la aplicación.

## Instalacion y confirguracion

Esta sección describe los pasos necesarios para instalar y ejecutar el sistema en un entorno local o de pruebas.

* + Instalacion del backend

1. Clonar el proyecto

git clone <https://github.com/Espinoza21268/agrostak.git>

1. Acceder a la carpeta backend

cd backend

1. Crear entorno virtual

Python -m venv venv

1. Activar el entorno virtual

source venv/bin/activate # Linux / macOS

venv\Scripts\activate # Windows

1. Instalar dependencias

pip install -r dependencias.txt

1. Configurar las variables de entorno en el archivo .env
2. Ejecutar el servidor

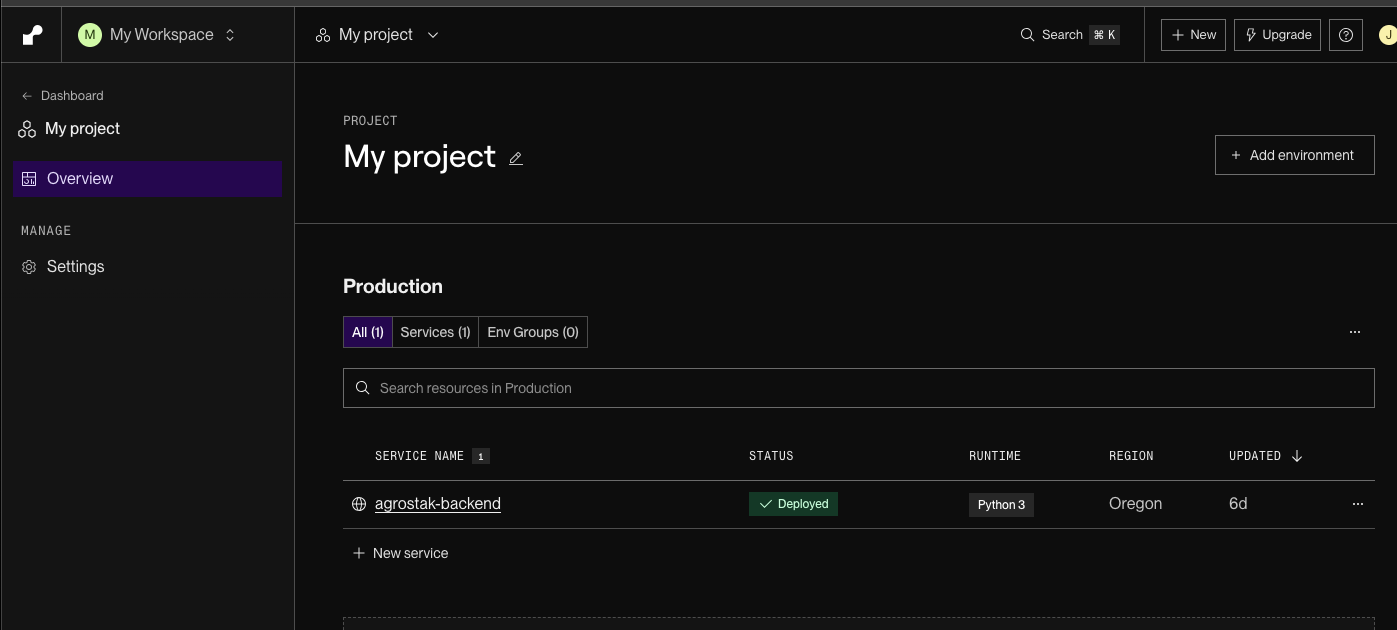
uvicorn app.main:app –reload

* + Instalacion del frontend
  1. Acceder a la carpeta frontend
  2. Verificar la estructura del archivo html, css y JavaScript
  3. Configurar la url de la api en los archivos Javascript configuracion.js
  4. Ejecutar el frontend desde un servidor local o directamente en el navegador.

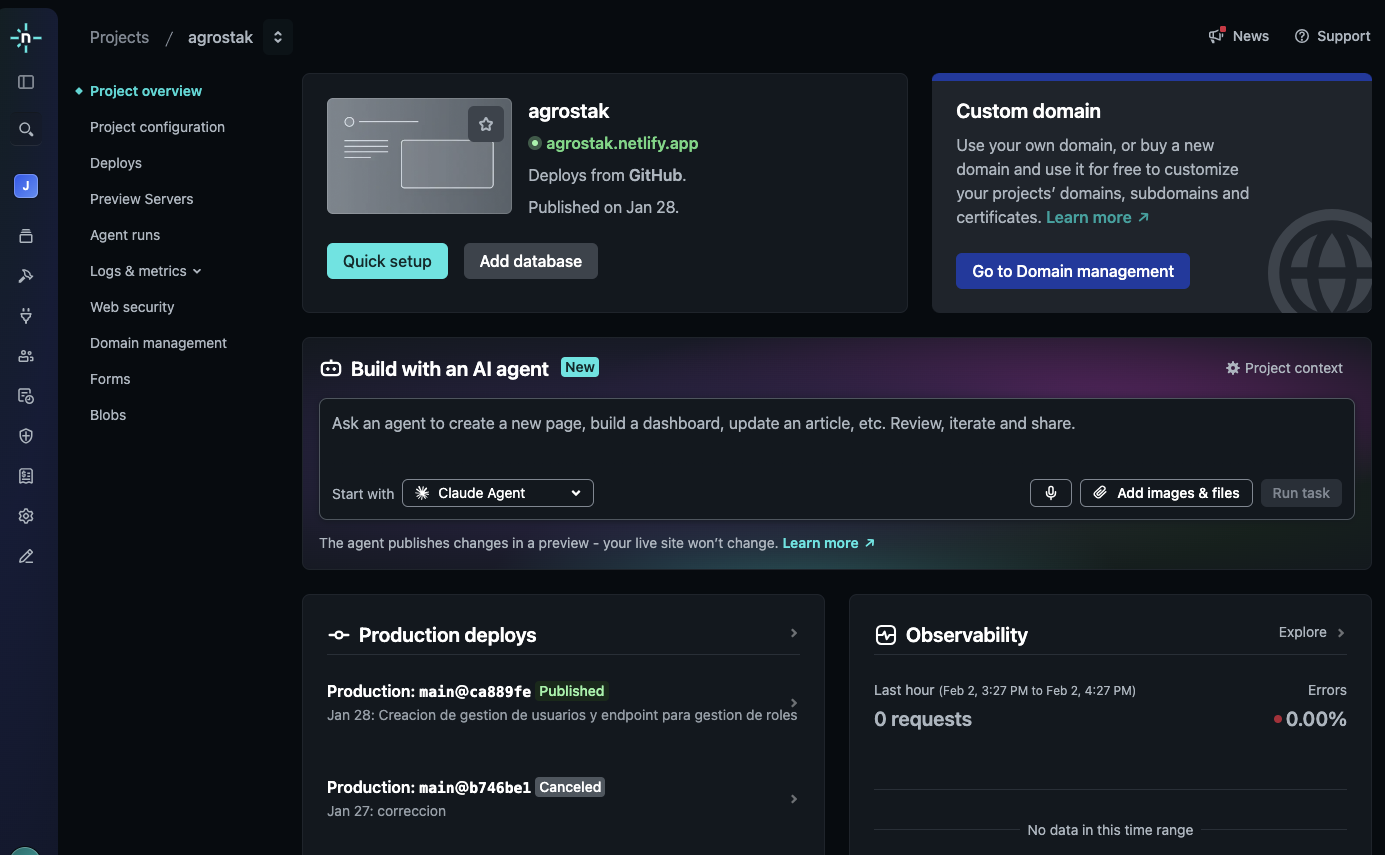
## Despliegue del sistema

El sistema puede ser desplegado utilizando servicios de alojamiento en la nube, permitiendo su acceso desde cualquier ubicación con conexión a Internet.

* + Despliegue del backend
    - Plataforma utilizada: Render
    - Tipo de servicio: WebService
    - Configuracion de variables de entorno
    - Exposicion de la API mediante HTTPS
    - Configurar la direccion del proyecto, debe especificar que es el backend



* + Despliegue del frontend
    - Plataforma utilizada: Netlify
    - Tipo de despliegue: Sitio web estatico
    - Consumo del backend mediante url publica



## Prueba tecnicas

Durante el desarrollo del sistema se realizaron pruebas para verificar el correcto funcionamiento de las funcionalidades implementadas.

* + Tipos de pruebas realizadas
    - Pruebas funcionales
    - Pruebas de autenticacion
    - Pruebas de consumo de API
  + Herramientas utilizadas
    - Navegador web
    - Postman
    - Consola del navegador
    - Visual code

## Mantenimiento y soporte

Para garantizar el correcto funcionammiento del sistema a largo plazo, se recomienda:

* Realizar copoas de seguridad perdiodicas de la base de datos
* Mantener actualizadas las dependencias del sistema
* Monitorear errores del servidor
* Documentar nuevas funcionalidades antes de su implementacion

## Glosario de terminos

* + API: Interfaz que permite la comunicacion entre aplicaciones.
  + Backend: Parte del sistema que gestiona la logica de negocio.
  + Frontendt: Interfaz visual con la que interactua el usuario.
  + JSON: Formato de intercambio de datos.
  + Autenticacion: Proceso de validacion de identidad.

## Anexos

Diagrama de Entidad Relacion de la BDD

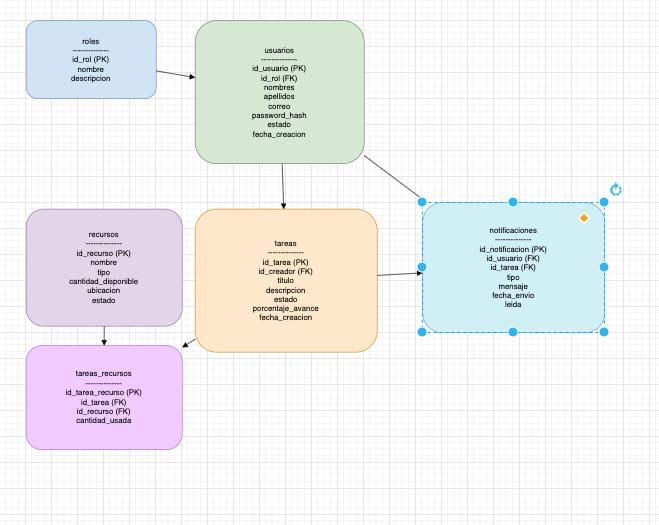


Diagrama de arquitectura de Backend:

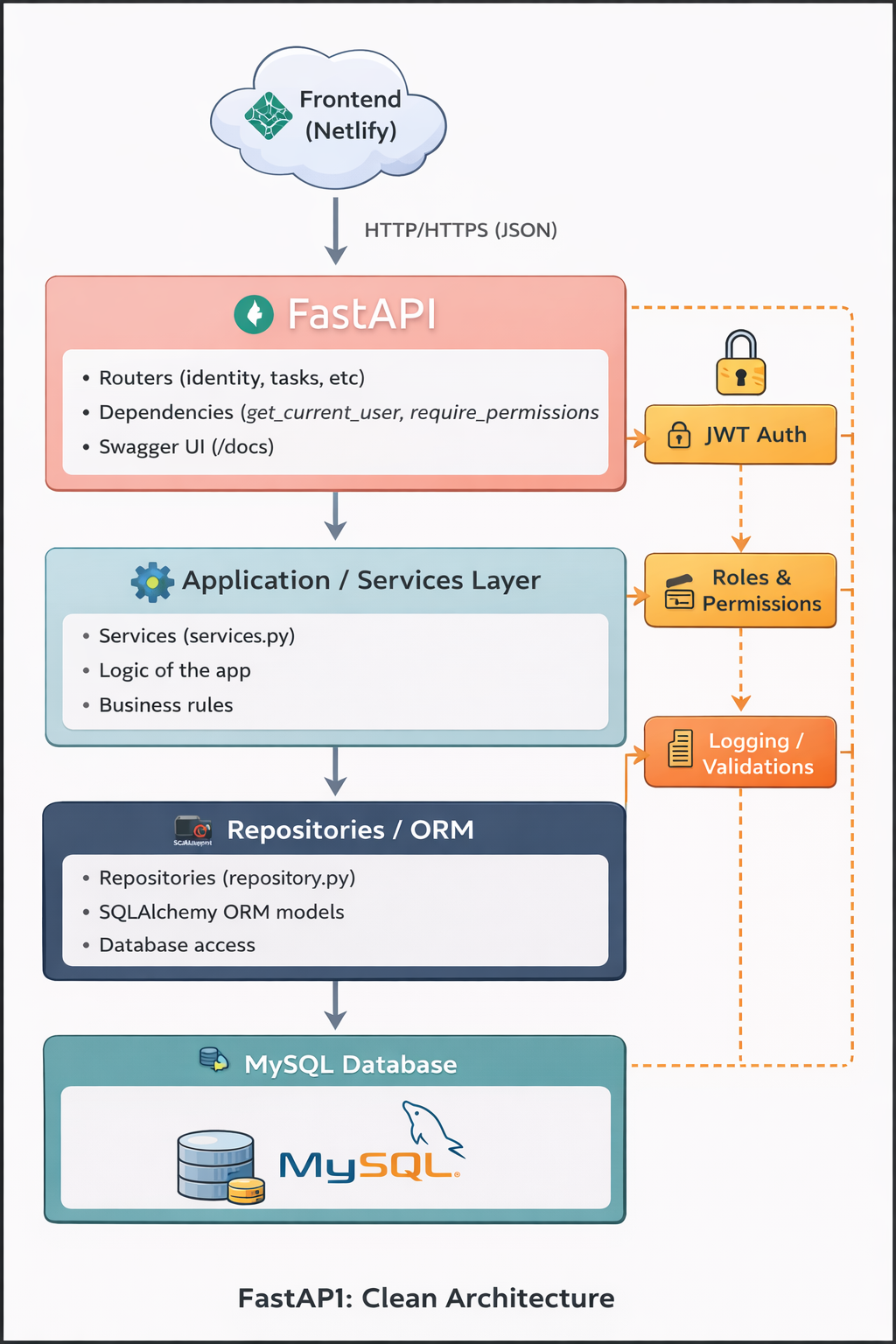
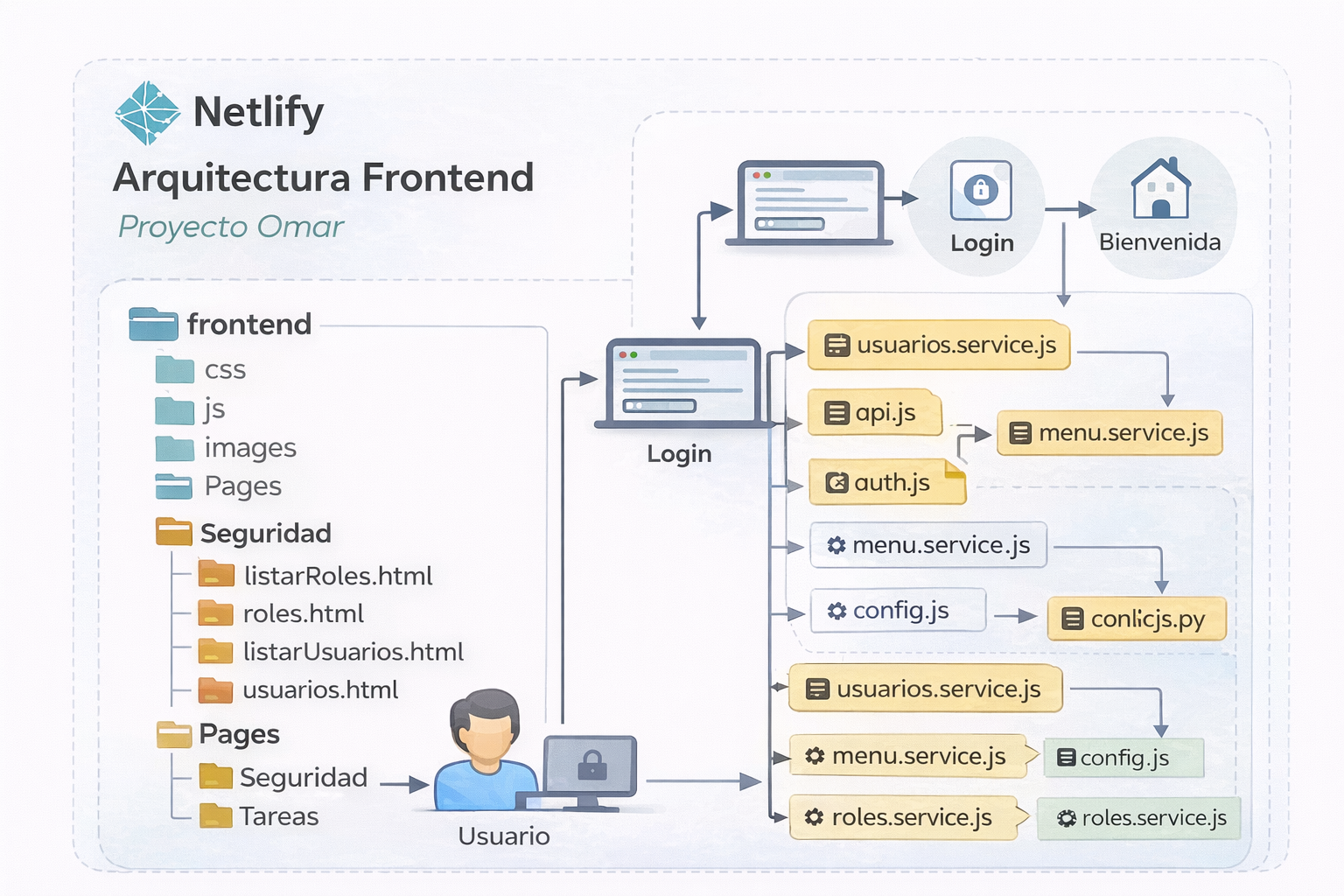


Diagrama del arquitectura del Frontend



Scripts de BDD

Creacion de BDD:

Para crear la BDD vacia ejecutar el siguiente script

<https://github.com/Espinoza21268/agrostak/blob/main/BDD/create-db.sql>

Para Crear los primeros usuarios Administrador, Supervisor y Trabajador ejecutar el siguiente script

<https://github.com/Espinoza21268/agrostak/blob/main/BDD/insetar-roles.sql>

Imagenes del sistema levantado