**Informe T.P integrador**

**Sistema de Gestión**

**Integrantes del grupo:**

Paladea Antu

Mamami Romina

Celeste Ramos

Cafferata Leandro

Fretes Alejandro

INDICE

[**Introducción/Objetivo** 3](#_Toc203303775)

[**Alcance del sistema** 3](#_Toc203303776)

[**Objetivo general** 3](#_Toc203303777)

[**Requisitos funcionales** 4](#_Toc203303778)

[**Login y permisos** 4](#_Toc203303779)

[**CRUD de productos** 4](#_Toc203303780)

[**Alertas** 4](#_Toc203303781)

[**Los requisitos No Funcionales:** 5](#_Toc203303782)

[**Software** 5](#_Toc203303783)

[**Arquitectura del sistema** 6](#_Toc203303784)

[**Patrón MVC** 6](#_Toc203303785)

[**Diagrama general:** 6](#_Toc203303786)

[**diagrama general del sistema** 7](#_Toc203303787)

[**Diagrama de Gantt** 8](#_Toc203303788)

[**Caso de uso** 9](#_Toc203303789)

[**Diagrama DER** 10](#_Toc203303790)

[**Guía de instalación y configuración** 10](#_Toc203303791)

[**Aplicaciones que se requieren para crear un sistema** 11](#_Toc203303792)

[**Paso 1: Instalar Dependencias del backend (Django), para esto abriremos Visual Studio Code e iremos a la terminal donde descargaremos Django** 11](#_Toc203303793)

[**Paso 2: Para configurar la base de datos local deberemos ir a la carpeta del proyecto y buscar el apartado de setting.py** 13](#_Toc203303794)

[**Paso 3: Crear una tabla para la base de datos** 14](#_Toc203303795)

[**Paso 4: Ahora para poder correr el servidor iremos a la terminal y usaremos un comando para iniciar el servidor** 16](#_Toc203303796)

[**Crendenciales del super usuario/admin** 17](#_Toc203303797)

[**Evaluación del Trabajo** 17](#_Toc203303798)

[**Anexos** 17](#_Toc203303799)

[**Anexo 1:** 17](#_Toc203303800)

[**Anexo 2:** 18](#_Toc203303801)

# **Introducción/Objetivo**

El proyecto tiene como objetivo desarrollar un sistema web funcional para la gestión de productos, centrado en una tienda dedicada a la venta de figuras de la temática Jujutsu Kaizen. Esta aplicación busca simular el funcionamiento de una plataforma de comercio electrónico enfocándose en la experiencia del usuario como en el control del inventario y administración del sistema.

El sistema deberá permitir realizar operaciones como el registro e inicio de sesión de usuarios, visualización del catálogo de productos y una gestión completa de los artículos en venta (Altas, Bajas, Modificaciones y visualización). También se deberá implementar un panel de administración para usuarios que posean el rol de administrador para que así puedan tener una gestión tanto de los productos como de los usuarios.

# **Alcance del sistema**

En este punto se detalla lo que se espera que el sistema va a permitir realizar como también los objetivos del sistema y cuáles serán los requisitos Funcionales y No funcionales

El sistema deberá permitir realizar las siguientes funciones:

* Gestión de usuarios y permisos (Login con roles).
* CRUD de productos (Altas, Bajas, Modificaciones y Visualización).
* Alertas de confirmación y advertencia de stock bajo.
* Interfaz responsiva para múltiples dispositivos.
* Panel de administración para control de inventario y usuarios.

## **Objetivo general**

El objetivo general que se espera es el de poder desarrolla un sistema web de gestión de productos en este caso basado en la franquicia Jujutsu Kaisen, el sistema debera no solo ser funciona si no que también modular y escalable

Mas énfasis en los objetivos que se esperan que pueda cumplir y realizar el sistema:

Implementar funciones de login y permisos por rol.

* Permitir el alta, baja, modificaciones y visualización de los productos.
* Mostrar una alerta de advertencia en el caso de que un producto tenga stock bajo.
* Diseñar una interfaz responsiva utilizando HTML, CSS y Bootstrap.
* Utilizar Django, Python y SQLite para encargarnos de toda la parte lógica del sistema y así pueda funcionar correctamente.

## **Requisitos funcionales**

Los requisitos funcionales son las acciones, comportamientos y funcionalidades que el sistema debe realizar para cumplir con su objetivo

### **Login y permisos**

Una gestión de los usuarios y sus permisos tales como

* Que los usuarios puedan registrarse e iniciar sesión.
* Los usuarios estándar solo podrán visualizar los productos del catálogo.
* Los usuarios con el rol de Administrador podrán gestionar tanto los productos como a los usuarios.

### **CRUD de productos**

Un CRUD de los productos que permita realizar las 4 operaciones básicas en un sistema de gestión de datos

* Altas: Cargar un nombre del producto, una descripción, el stock, el precio y una imagen.
* Bajas: Tener la opción de poder eliminar los productos.
* Modificaciones: Poder editar los datos de cada producto.
* Visualización: Poder Mostrar el catálogo de productos de forma ordenada y poder buscar productos más cómodamente

### **Alertas**

Alertas de advertencia y stock bajo

* Una confirmación visual al eliminar o editar algún producto.
* Una notificación de “Stock Bajo” que nos indique que el stock de algún producto es menor a 5 unidades.

## **Los requisitos No Funcionales:**

* Seguridad: Control de acceso según el rol.
* Interfaz de Usuario: Adaptada a escritorio y móvil.
* Escalabilidad: Preparado para expansión de base de datos y nuevas funciones.

# **Software**

Software y herramientas que se utilizaron para crear y desarrollar el sistema:

Lenguaje backend: Python

Framework: Django

Frontend: HTML, CSS

Base de datos: SQLite (viene predeterminadamente con Django)

Navegadores compatibles: ¿?

Herramientas: Github, Visual Studio Code

# **Arquitectura del sistema**

## **Patrón MVC**

Nuestro sistema web de gestión de productos está estructurado bajo el patrón **MVC (Modelo – Vista – Controlador)**, lo cual permite organizar el código de forma clara, separar responsabilidades y facilitar el trabajo en equipo.

* **Modelo:** Define las propiedades principales del sistema como Productos y Usuarios. Se implementa en models.py y es responsable de la estructura de la base de datos.
* **Vista:** Es la parte visual que ve el usuario. Está compuesta por archivos HTML ubicados en la carpeta templates/, e incluye Bootstrap y CSS para mejorar la estética. Muestra los catálogos, formularios de login, panel de administración, etc.
* **Controlador:** Gestiona la lógica del sistema. Las funciones definidas en views.py reciben las solicitudes del usuario, interactúan con los modelos y devuelven una vista de lo solicitado.

## **Diagrama general:**

El diagrama esta conformado con una arquitectura que se divide en tres roles fundamentales que se dividen en Frontend, Backend y Base De Datos.

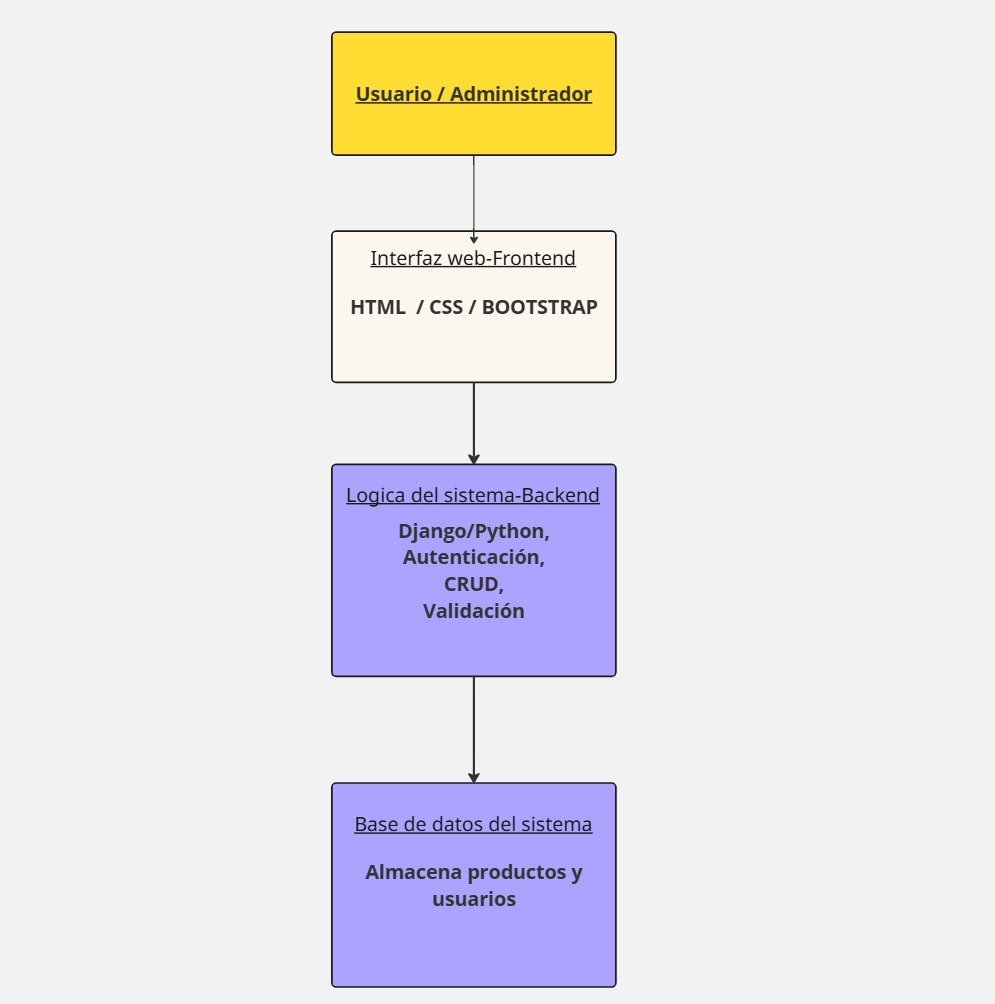
Cada uno de estos roles va a cumplir un funcionamiento específico para el sistema

Frontend: Se encargará de la parte visual y estética del sistema con la que los usuarios y administradores interactúan, en este caso hicimos usos de herramientas como HTML, CSS y Bootstrap para poder lograr una interfaz accesible y atractiva, tanto para el usuario el cual puede encontrar cómodo al momento de ver los productos como también para los administradores al momento de ver formularios y alertas o cargar nuevos productos

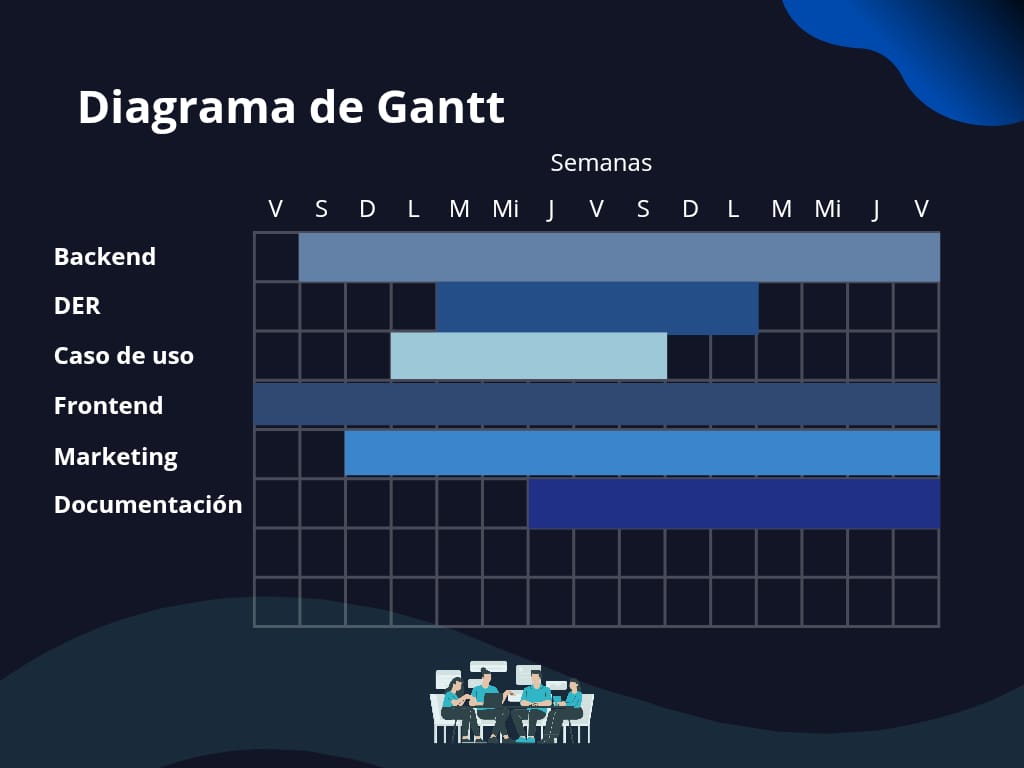
Backend: Se encargo de manejar la lógica del sistema y de que al momento de recibir las solicitudes del Frontend estas puedan se procesadas y así al comunicarse con la base de datos poder realizar la solicitud con éxito. También se encarga de controlar otras cosas como la autenticación, los permisos que tiene cada usuario y la gestión del inventario

Base de datos: Se encarga de almacenar toda la información o datos que le agreguemos como los usuarios, productos, el stock y los roles

## **diagrama general del sistema**



# **Diagrama de Gantt**



# **Caso de uso**



# **Diagrama DER**



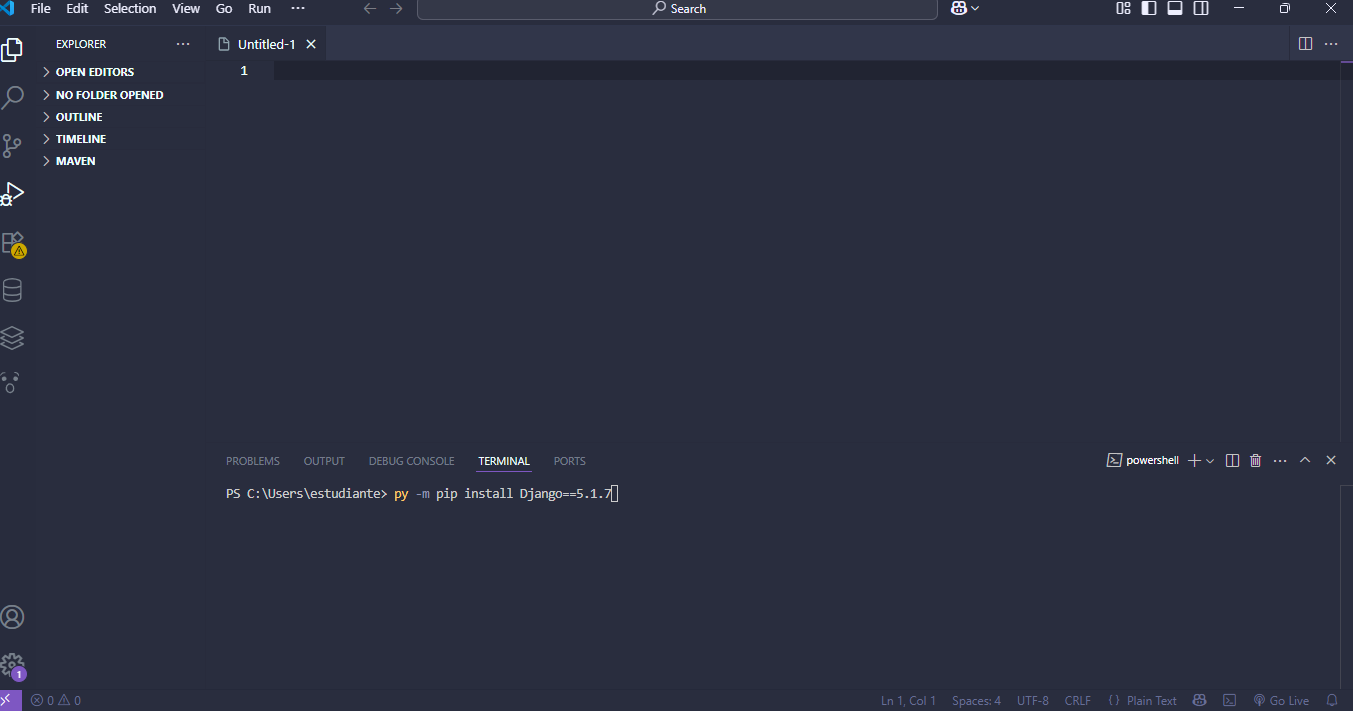
# **Guía de instalación y configuración**

Este punto detallara los pasos para crear y correr un sistema

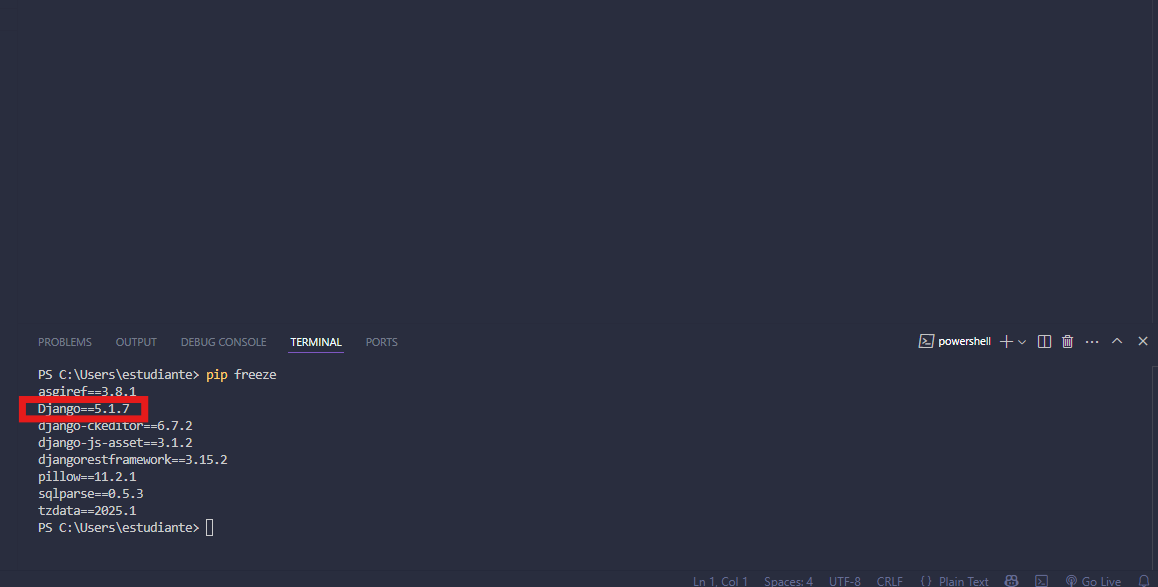
## **Aplicaciones que se requieren para crear un sistema**

* Visual Studio Code
* Python
* Django

## **Paso 1: Instalar Dependencias del backend (Django), para esto abriremos Visual Studio Code e iremos a la terminal donde descargaremos Django**



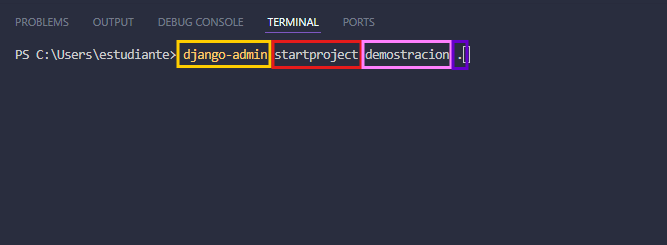
Para comprobar que django se instalo podes utilizar el comando pip freeze que nos mostrara todas las dependencias instaladas y tendremos que fijarnos que aparezca django con su version



Para Iniciar un Proyecto y una aplicación utilizaremos 2 comandos los cuales serán

**django-admin startproject “Nombre del proyecto” .**

**django-admin starapp “Nombre de la aplicación”**

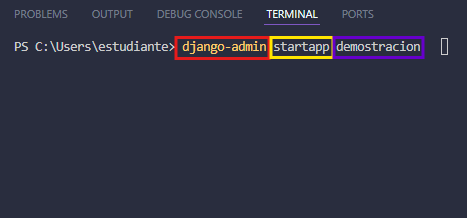
****

**Django-admin:** determina que el proyecto se va a generar desde Admin.

**Startproject**: Comando utilizado para indicar que vamos a crear un proyecto.

**Demostración**: Este será el nombre de la carpeta que contendrá el proyecto, el cual pueden elegir el nombre que quieran.

**El punto:** El punto al final y separado se usa para indicarle que la carpeta del proyecto se cree en la raíz.

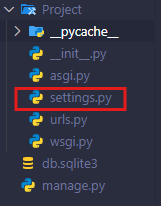


**Django-admin:** termino que utilizamos para generar una aplicación desde Admin.

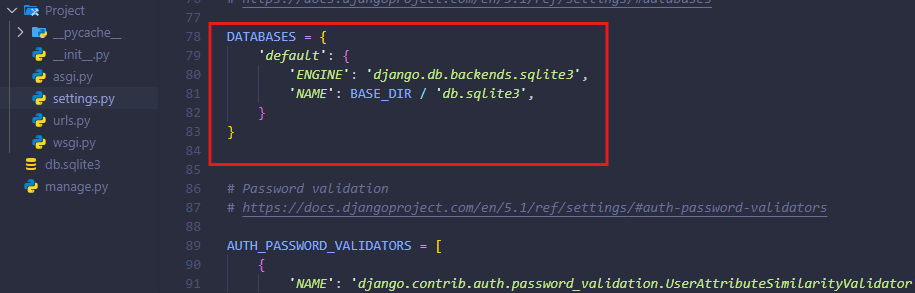
**Startapp:** Comando para crear una aplicación.

**Demostración:** Este será el nombre de la aplicación también pueden elegir el que quieran.

## **Paso 2: Para configurar la base de datos local deberemos ir a la carpeta del proyecto y buscar el apartado de setting.py**



Cuand estemos en settings.py bajaremos hasta que veamos una parte que diga DATABASES



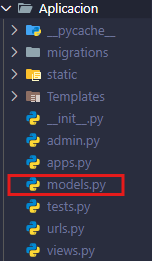
‘ENGINE’ define el motor que se va a usar como base de datos en este caso SQLite viene predeterminado

‘NAME’ declara como se va a llamar el archivo de la base de datos y que estara ubicada en la carpeta raiz del proyecto

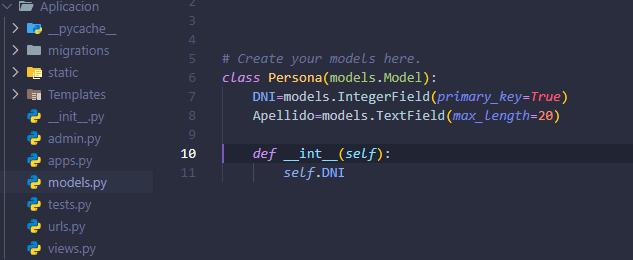
## **Paso 3: Crear una tabla para la base de datos**

**Acá explicaremos como se crear una tabla que se usara para cargar algunos datos.**

**Primero iremos a la carpeta de la aplicación y ahí iremos al apartado de models.py que es donde crearemos las tablas para la base de datos**



Una vez en la parte de models.py podremos crear una tabla

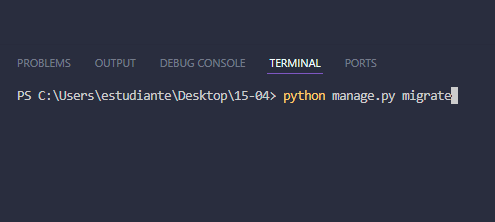


**Class Persona(models.Model)**: Define un **modelo (table)** llamado **Persona**, con el **models.Model** Django entiende que esta clase representa una tabla

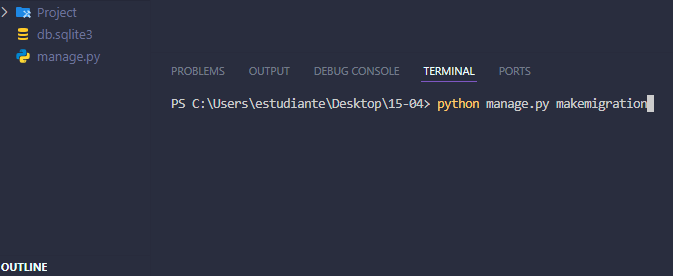
**DNI=models.IntegerField(primary\_key=True)**: con este comando creamos una columna que guarda numeros enteros y con el **Primary\_key** le decimos que **DNI** va a ser la llave primaria de esta tabla

**Apellido=models.TextField(max\_length=20)**: Con esto creamos una columna para texto largo y con el max\_length=20 le declaramos que esta columna solo puede tener un máximo de 20 caracteres

Ahora toca crear el archivo de base de datos

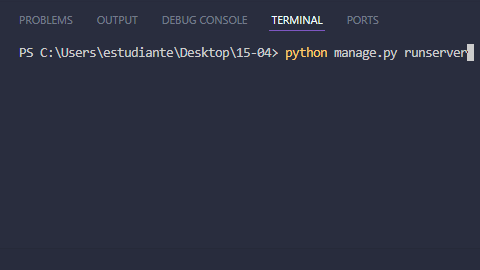


Cuando creemos el archivo de base de dato nos tocara usar el comando makemigrations el cual va a general l tabla que creamos en models.py pero ahora la añadirá a la base de datos

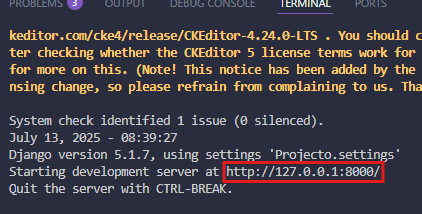


Una vez que hayamos generado la tabla tocara usar el comando mígrate de nuevo para que nuestra tabla aparezca en la base de datos.

## **Paso 4: Ahora para poder correr el servidor iremos a la terminal y usaremos un comando para iniciar el servidor**



Cuando pongamos ese comando cargaran una serie de comando, pero el que importante Sera la dirección de nuestro proyecto



Podemos copiar eso en el buscador de nuestro navegador o podemos hacer control + clic en <http://127.0.0.1:800/>

# **Crendenciales del super usuario/admin**

Usuario: antu

Contraseña: 1234

# **Evaluación del Trabajo**

El desarrollo y trabajo del sistema web desde los distintos roles del equipo fue una buena experiencia para todo el equipo. Logramos cumplir con la mayoría de los objetivos planteados, como la implementación del sistema de login con roles, la gestión de productos mediante operaciones CRUD y el diseño de una interfaz responsiva.

Durante el transcurso del trabajo, cada integrante se enfocó en su rol: backend, frontend, UI/marketing y documentación. Esto facilitó el desarrollo paralelo y permitió avanzar de forma organizada. Entre lo mas desafiante estuvo el desarrollo de marketing.

# **Anexos**

## **Anexo 1:**

Logo de la empresa



## **Anexo 2:**

Banners promocionales

