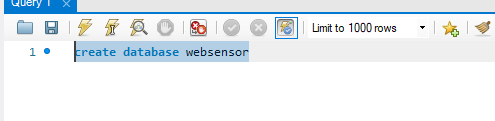
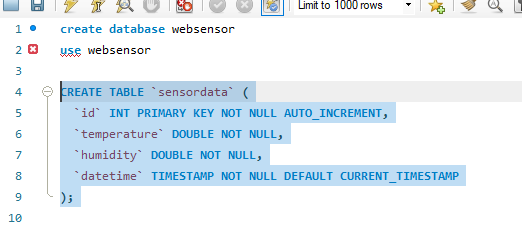
# Examen 2

NOMBRE: CAYLLAHUA CHIRE CARMEN ROSA GRUPO: A

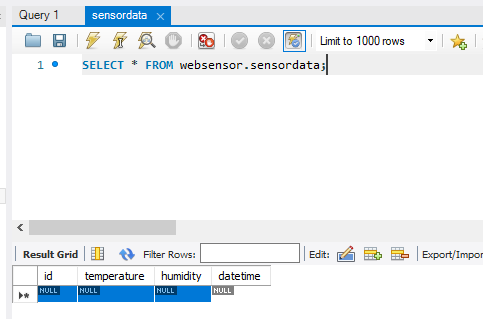
1. Creacion de la base de datos con el nombre ‘websensor’ en MySQL Workbench



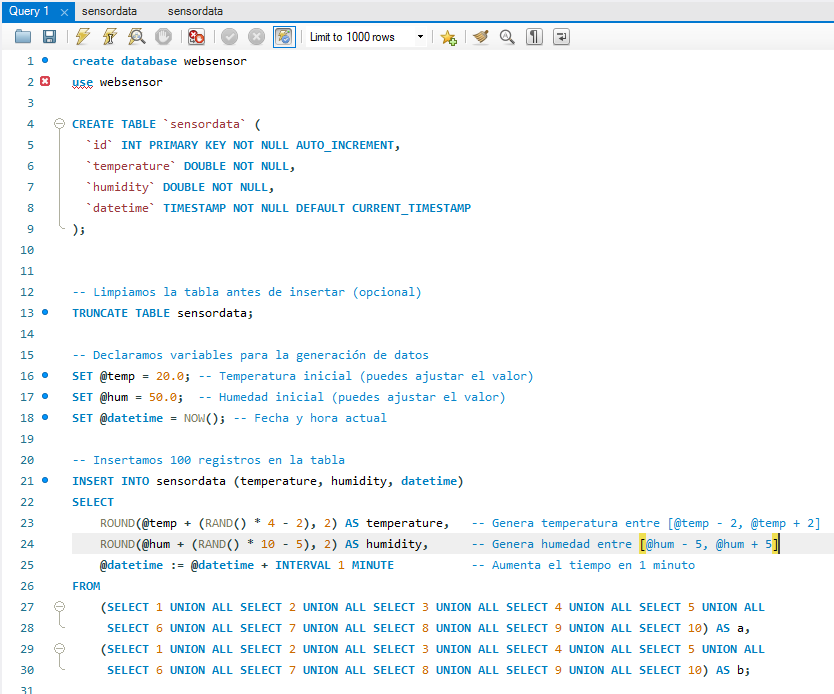
1. Seleccion de la base de datos y abajo se colocó el codigo SQL para crear la tabla

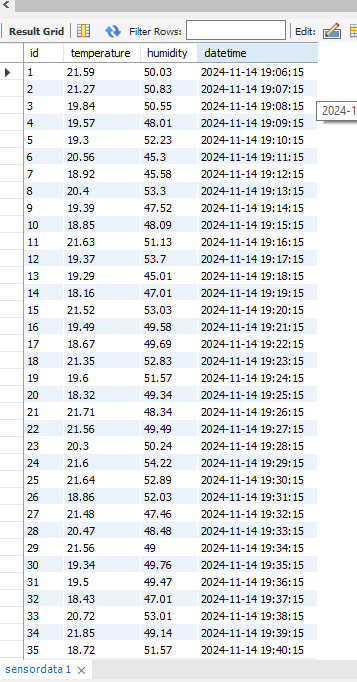


Selecciopnamos las filas de la tabla

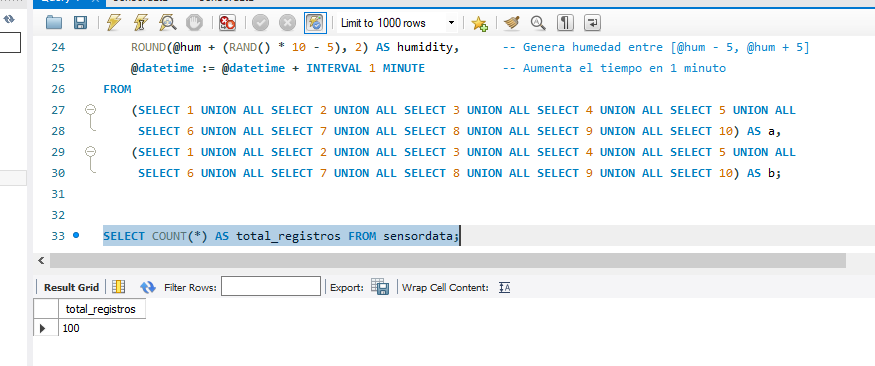


Debajo de truncate que ez para limpiar la tabla generamos los datos y los gruarmos y ugeo los insertamos





En aca utilzamos el comando count para contar lasifals de al tablas y ver que son 100 regsitros

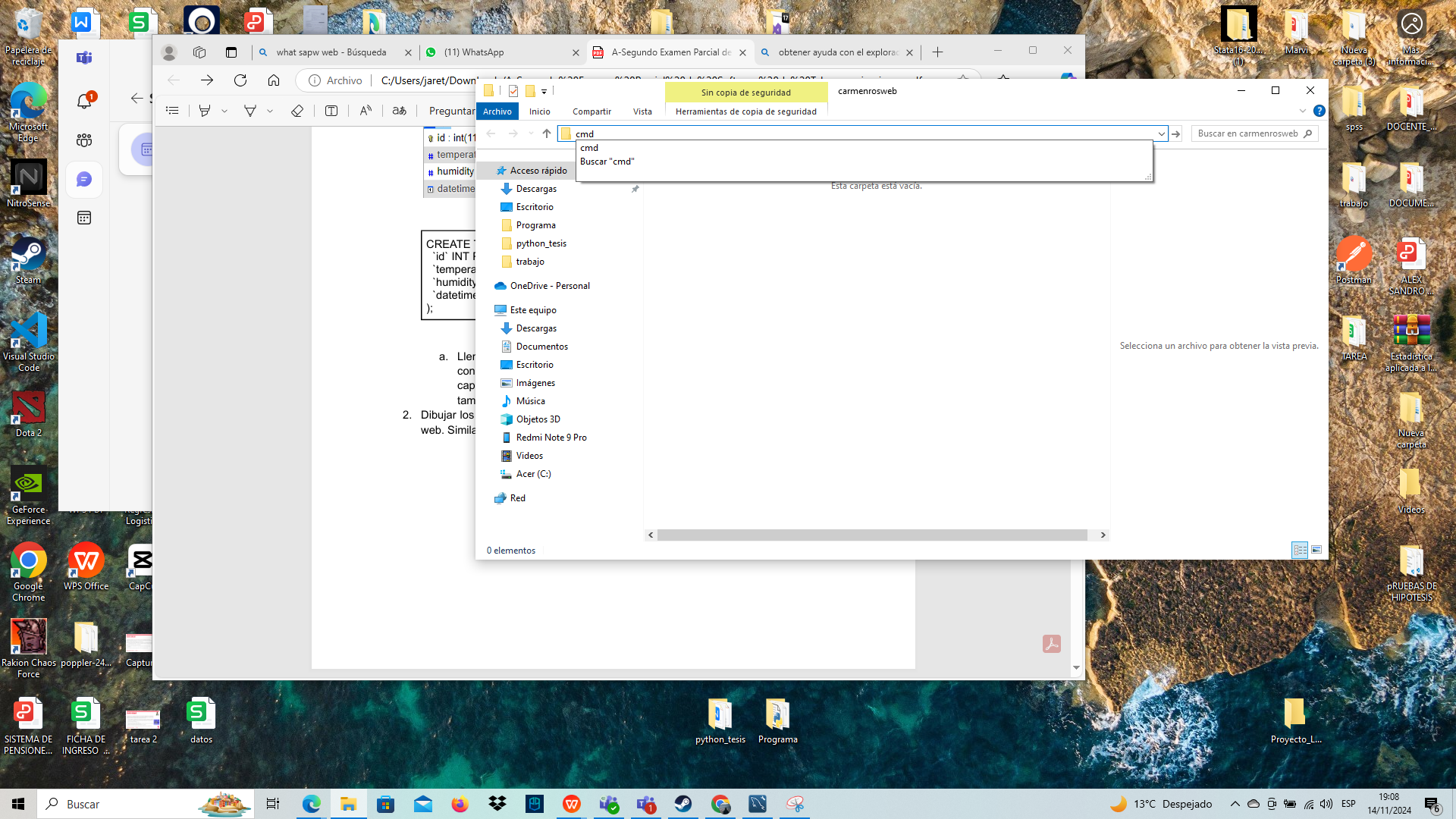


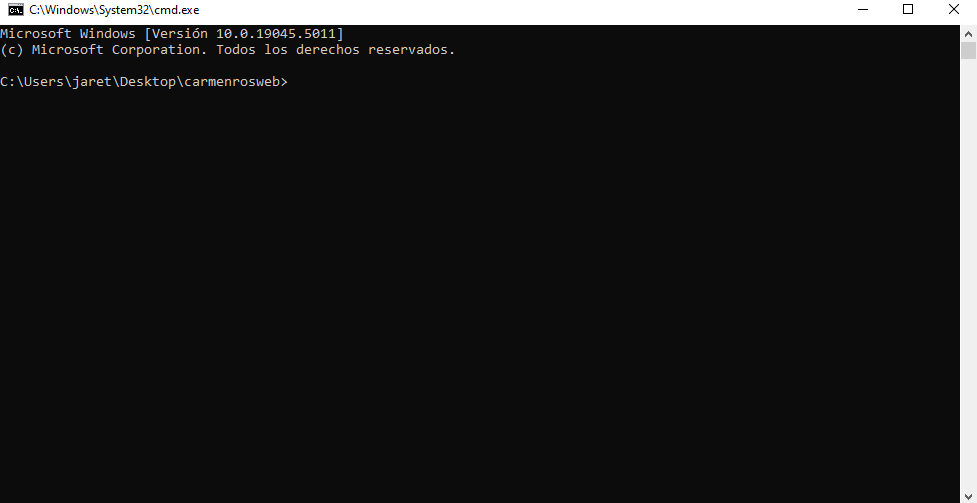
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%pregunta 2

Creamos la carpeta donde se alojara el proeyctop

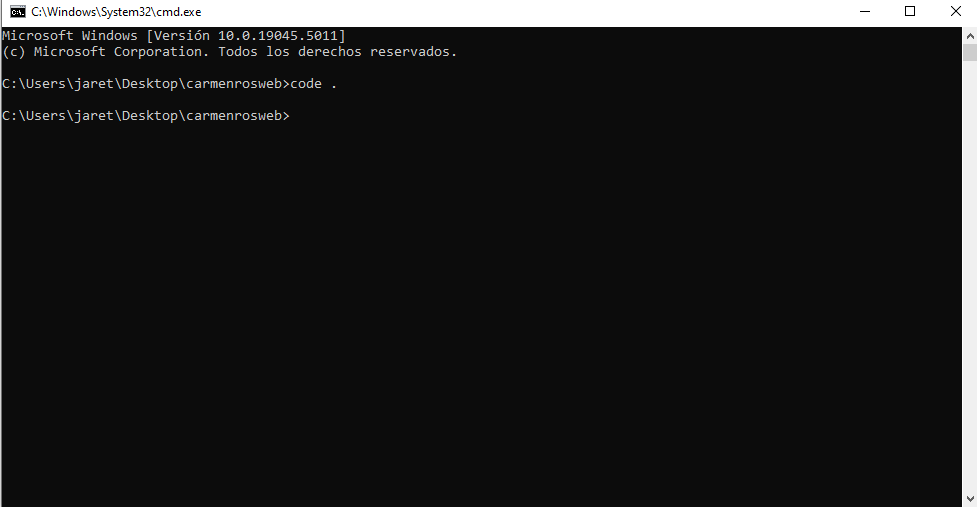


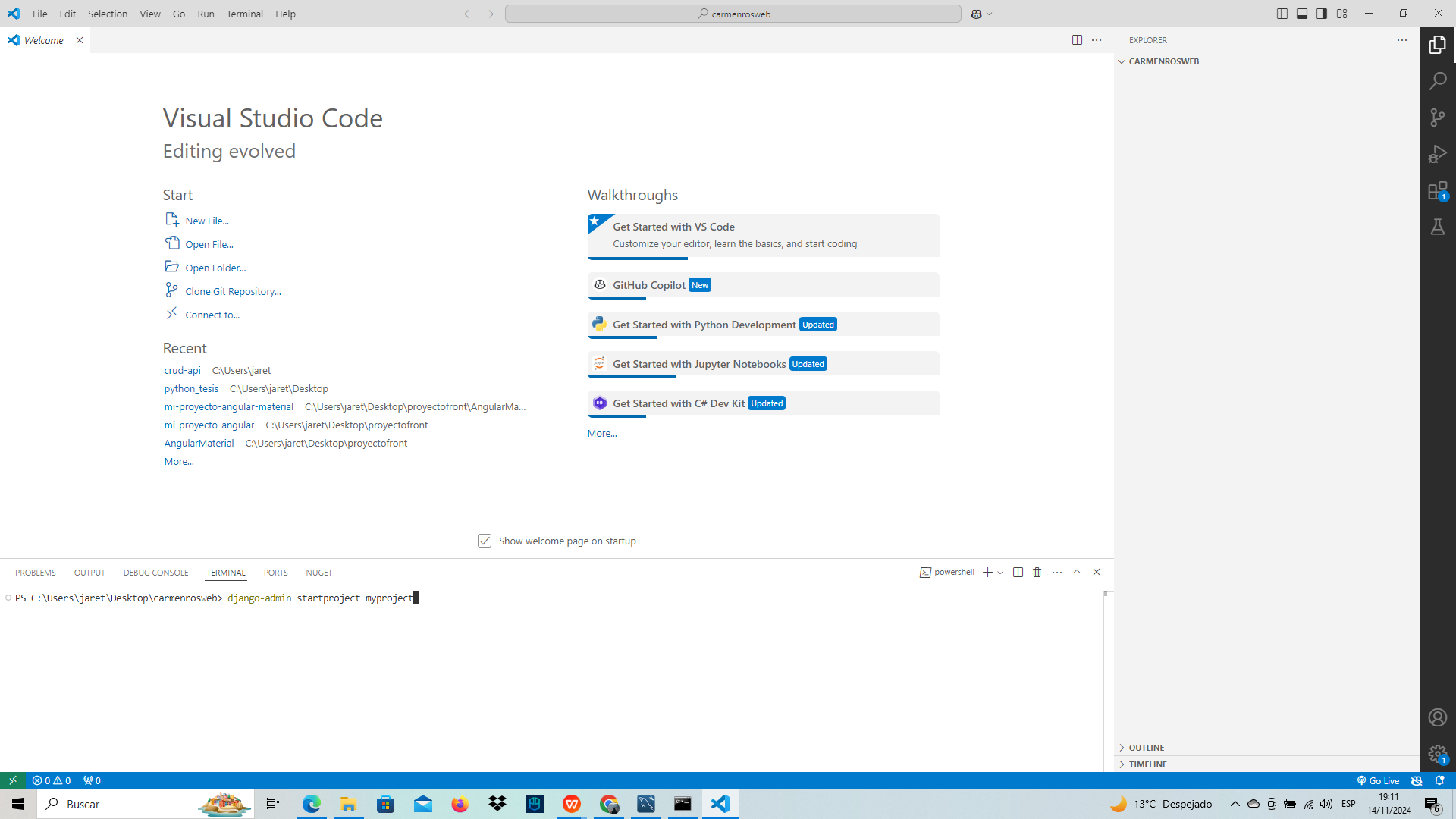
Usamos el cmd para ver la ruta del prueycto



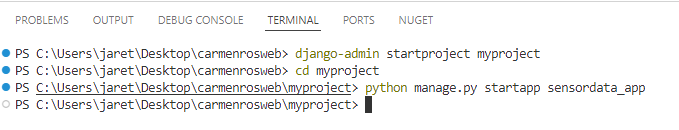


Abrimos el visual estucio doce que es un ediutor de codigo fuente para abir el visual estudo code en esa ruta



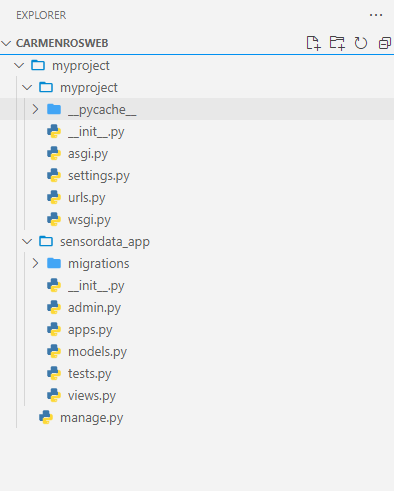


Con estos comando creamos la aplciacion yque se llamara sensordata\_\_app

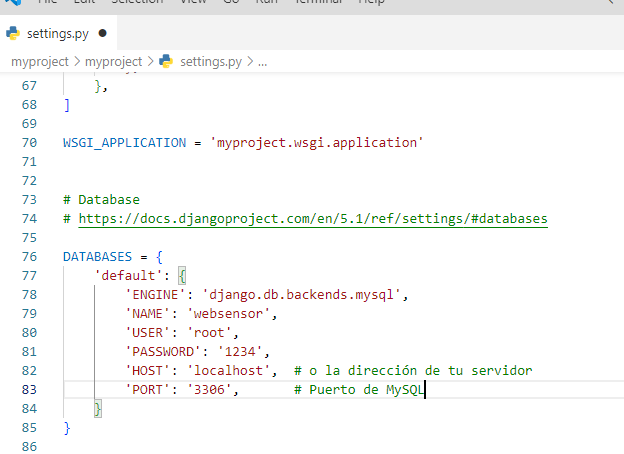


Aca vemos la estructura del proyecot donde son dos carpetas una principal y otra para la aplicacion y el de principal tiene ajuste spara configurar

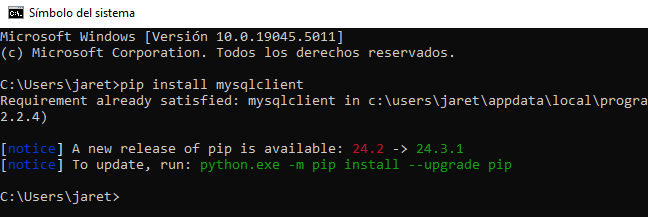
Y tambien hay carpeta de migfraciones para que la aplicacionse conecte con la base de datos pero en este caso no es necesario por que ya emos creado la tabla en la base de datos



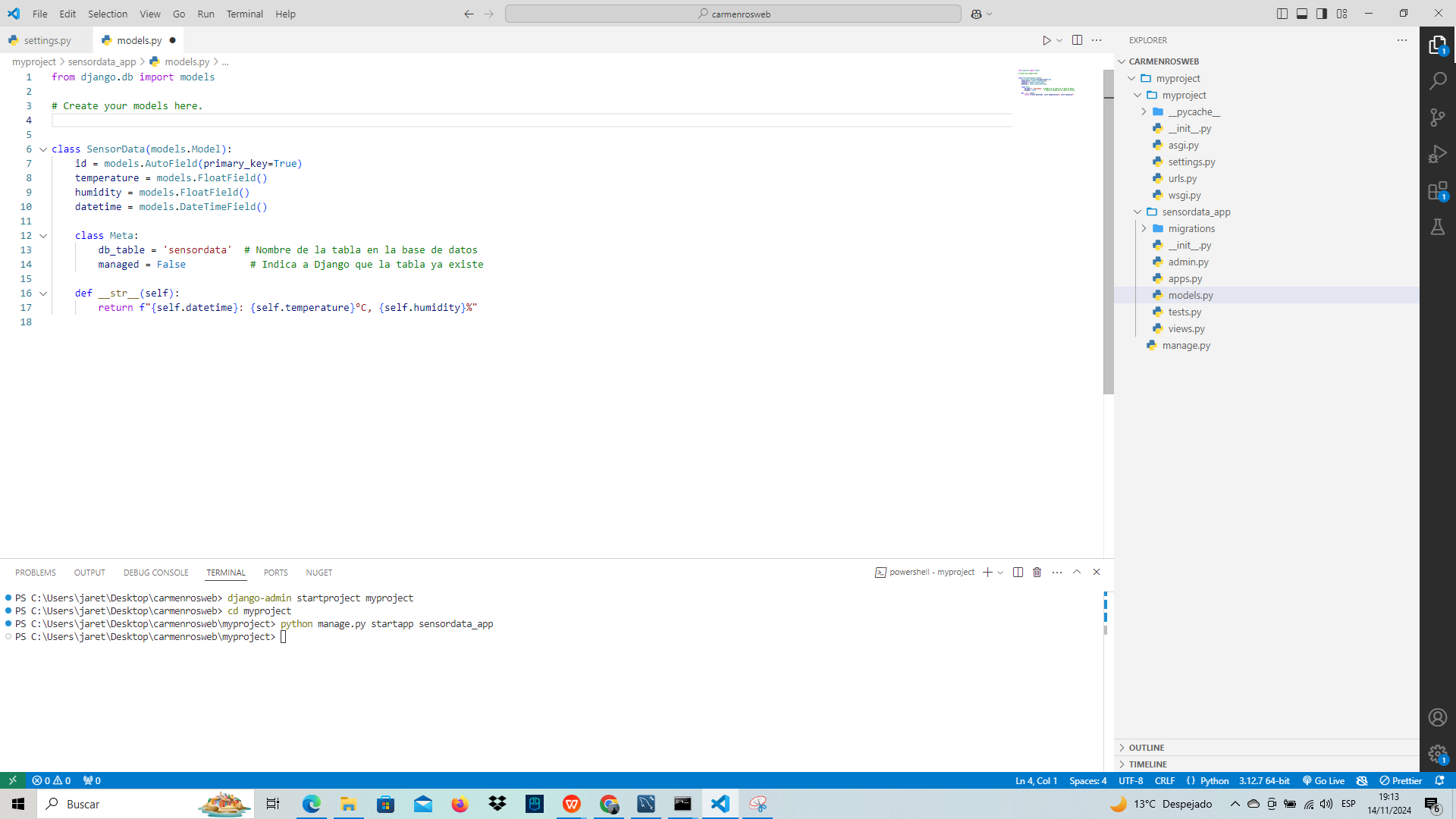
En el setting del a carptea princiuñal tenemos la configuracion ade al base dade datos donde le damos el usurioa el nombre de la base datos y la contrasena y el puerto



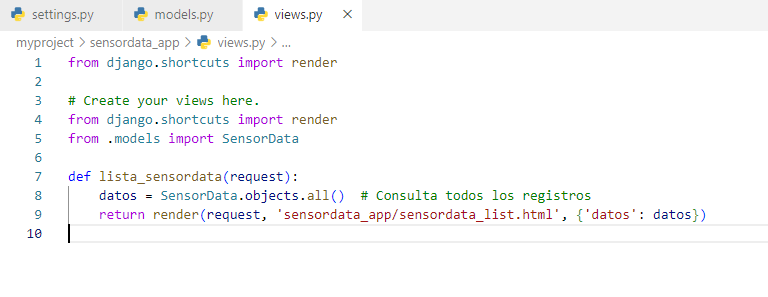
Instalamos la libreria de mysql en pyhtn



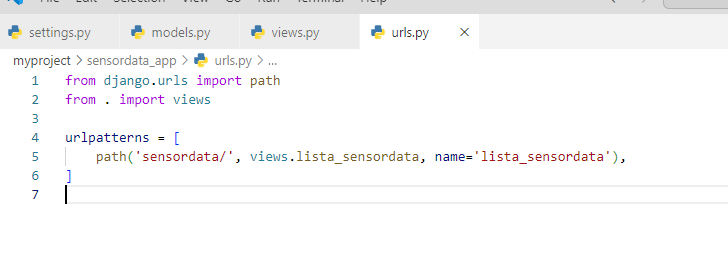
Creamos al entidad o modelo que debesse de ser similar ala tabla que ehemos creado en la base de datos



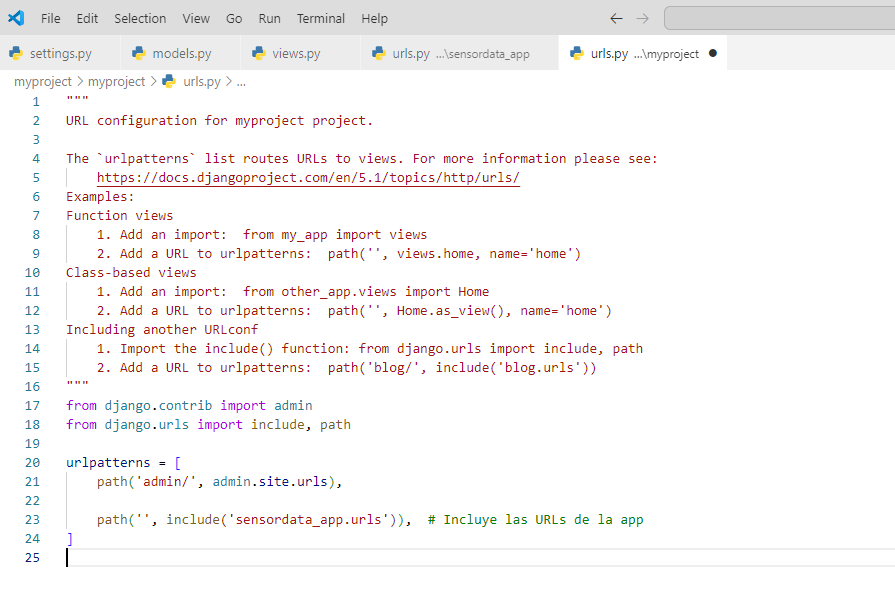
Creamos la vista del a apop,la vista se encarga de la funcionalida dde selccionar todos los datos de la base dedatos



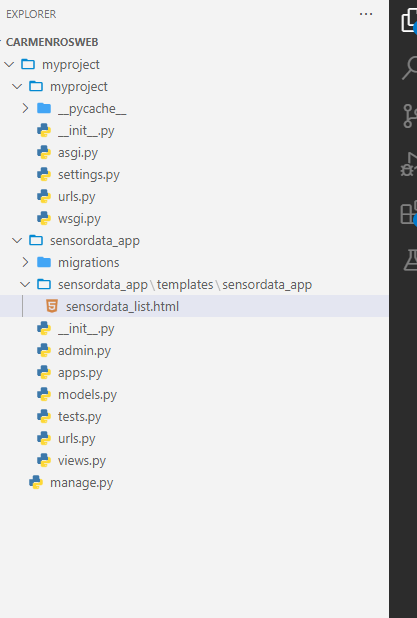
Creamos la ruta donde decimos que en aca dentrara la url de aplicacion



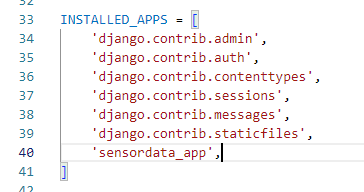
Deifnimos rutas tambien en la aplicaicon principal y llamaos en esta ruta la placiacion del sensor



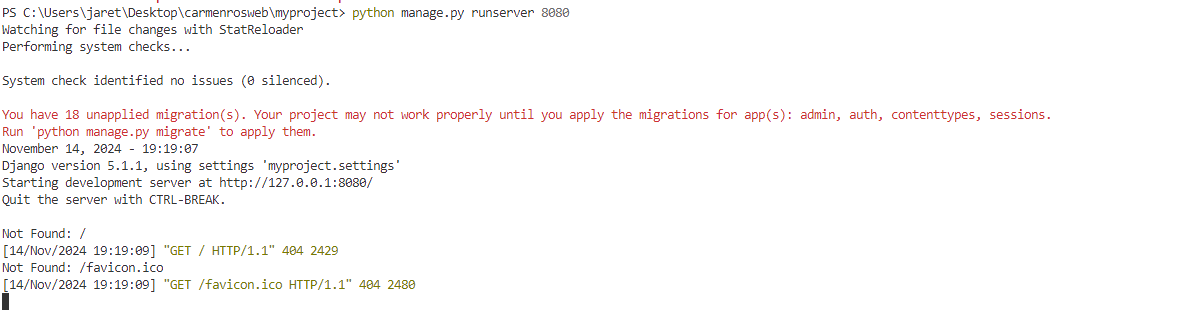
Cremos los htmls



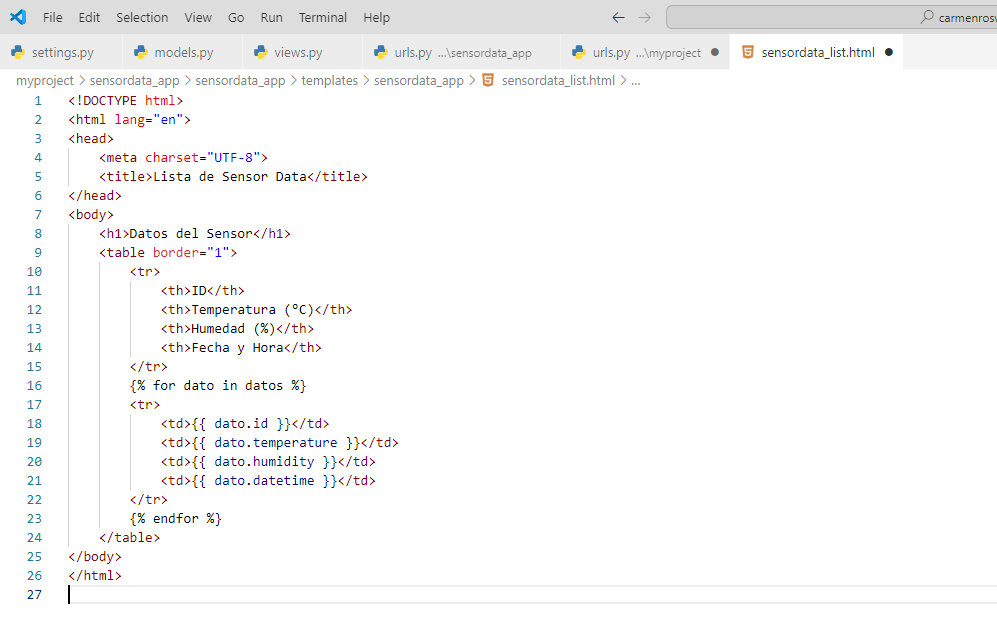
Declaramos el nombre de la aplicacion en el setting dela aplciacion principal

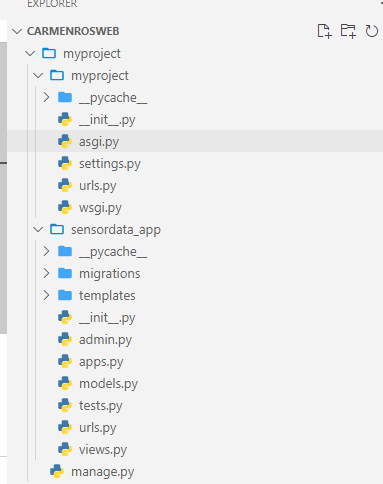


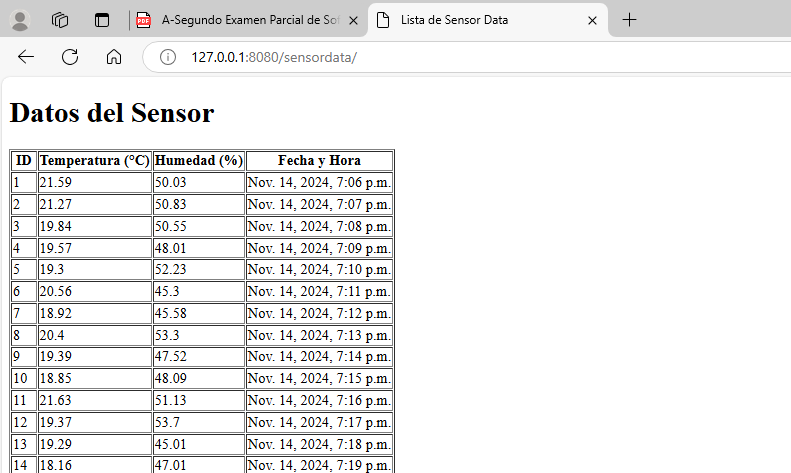
Le damos rtun server



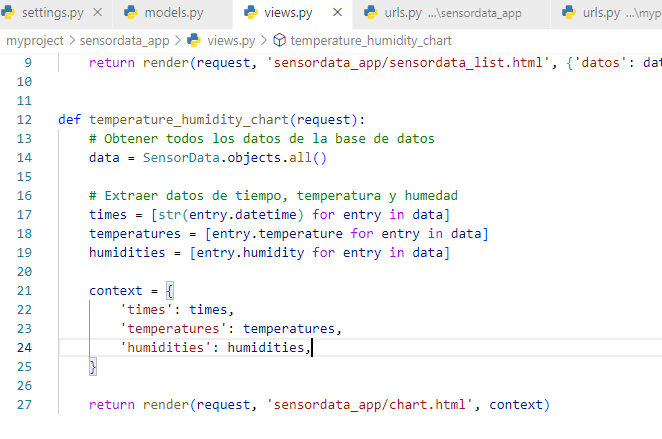
En aqui creamos una talba apra los fdaots de la base dedetaos







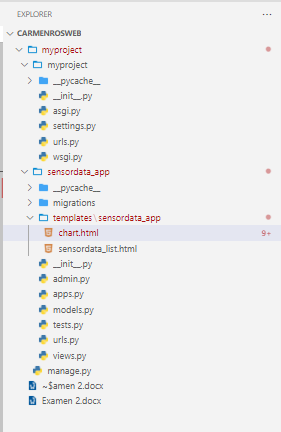
Mejoramos la viista para obtener un grafico



Mnejoramos el html para mejorar el grafico



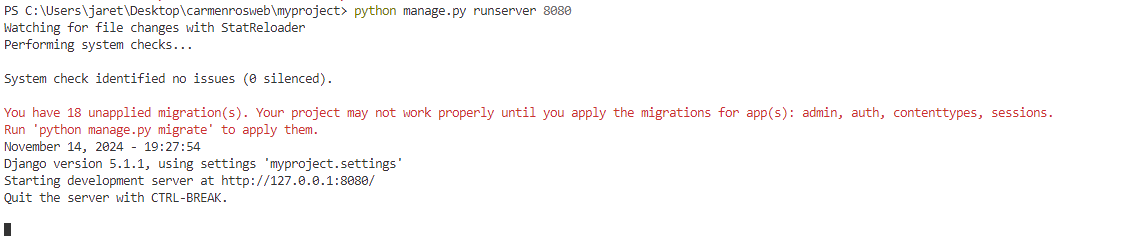
Creamos un nuevo template llamod chart.html apra ver la grafica



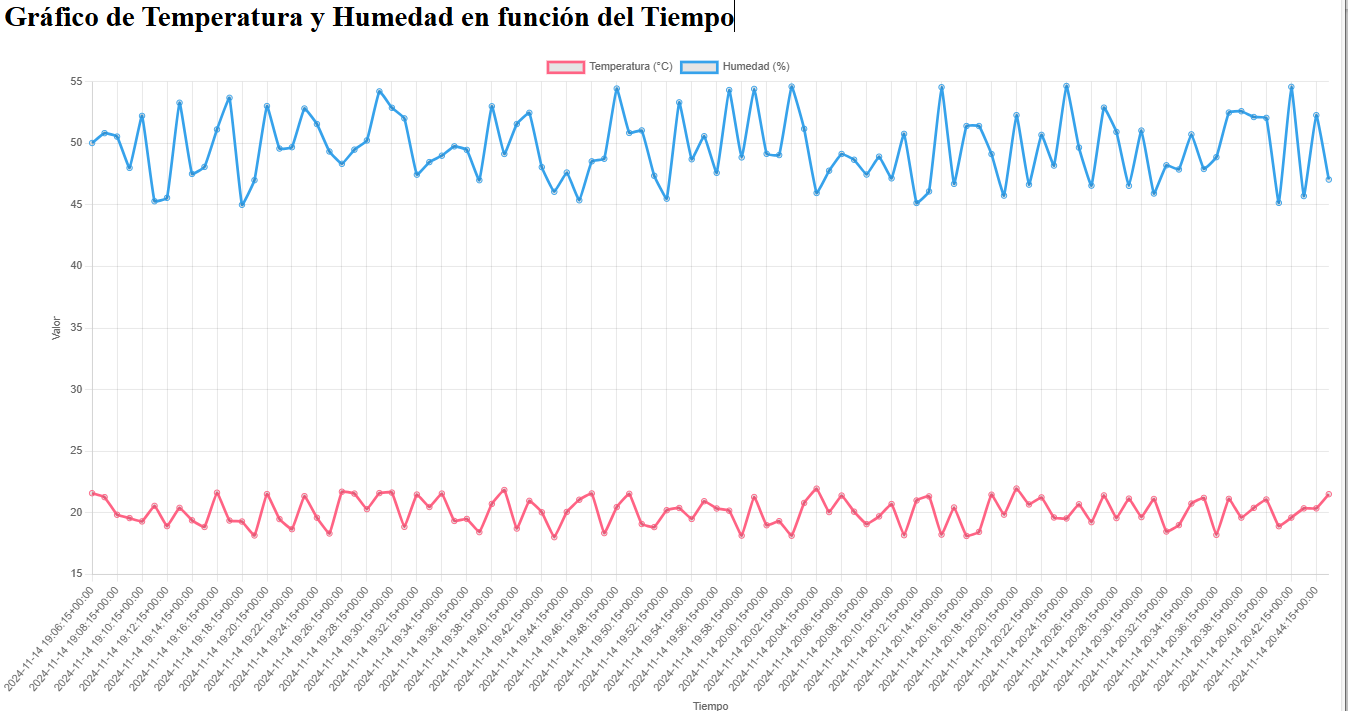
Agregamos su nueva ruta



Despues :le damios run server

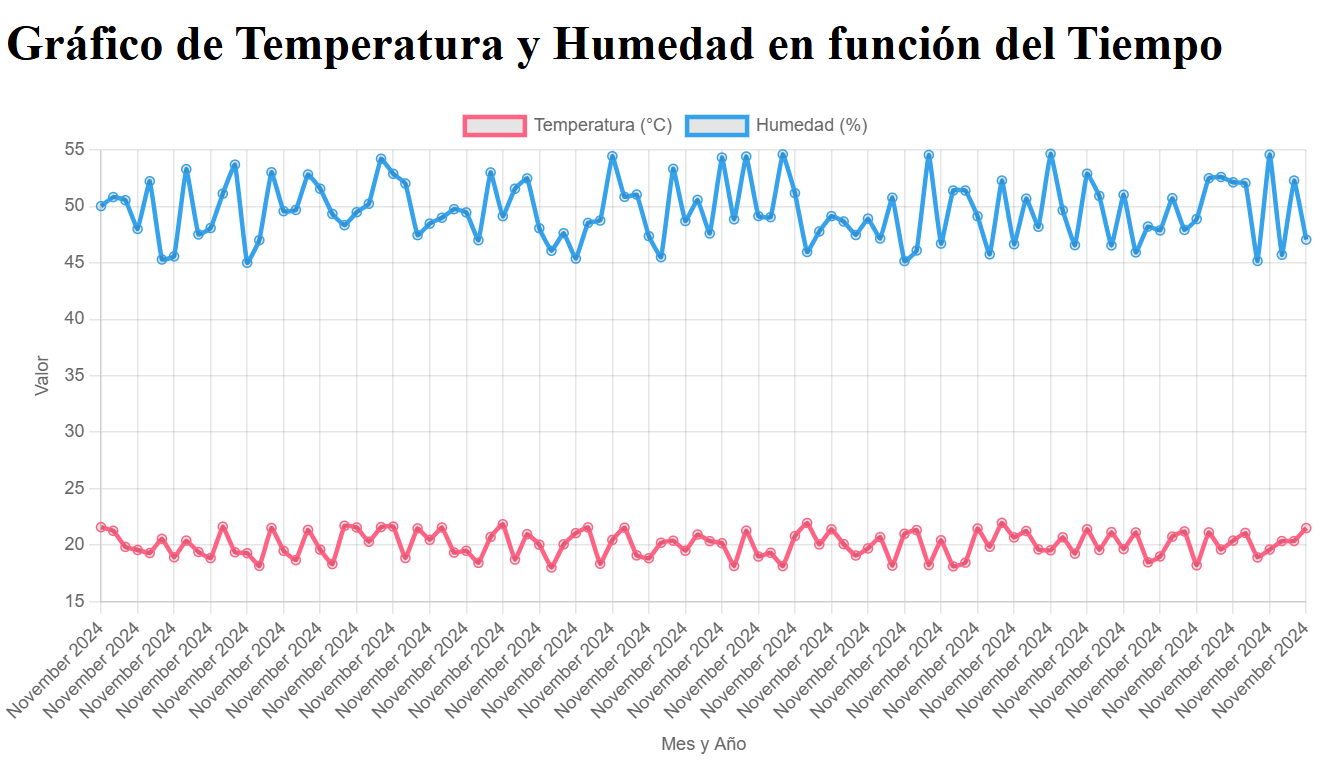


Obsermvamos la grafica mejorada



Tenemos el unevo codifo





# Codigo Final con grafico Final

<!DOCTYPE html>

<html lang="es">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Gráfico de Temperatura y Humedad</title>

    <!-- Cargar Chart.js desde un CDN -->

    <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>

    <!-- Cargar Moment.js para formatear las fechas -->

    <script src="https://cdn.jsdelivr.net/npm/moment@2.29.1/moment.min.js"></script>

</head>

<body>

    <h1>Gráfico de Temperatura y Humedad en función del Tiempo</h1>

    <!-- Crear un contenedor para el gráfico -->

    <canvas id="temperatureHumidityChart" width="800" height="400"></canvas>

    <script>

        var ctx = document.getElementById('temperatureHumidityChart').getContext('2d');

        // Obtener los datos desde Django

        var times = {{ times|safe }};

        var temperatures = {{ temperatures|safe }};

        var humidities = {{ humidities|safe }};

        // Formatear las fechas para mostrar solo mes y año

        var formattedTimes = times.map(function(time) {

            return moment(time).format('MMMM YYYY');  // Formato: Mes Año (ej. 'Marzo 2024')

        });

        // Crear el gráfico

        var temperatureHumidityChart = new Chart(ctx, {

            type: 'line',  // Tipo de gráfico

            data: {

                labels: formattedTimes,  // Ejes X: tiempo (formateado)

                datasets: [{

                    label: 'Temperatura (°C)',  // Etiqueta del gráfico

                    data: temperatures,  // Datos de temperatura

                    borderColor: 'rgb(255, 99, 132)',  // Color de la línea de temperatura

                    backgroundColor: 'rgba(255, 99, 132, 0.2)',  // Relleno debajo de la línea (color suave)

                    fill: true,  // Rellenar el área debajo de la línea

                    tension: 0.1  // Suavizar la curva de la línea

                }, {

                    label: 'Humedad (%)',  // Etiqueta del gráfico

                    data: humidities,  // Datos de humedad

                    borderColor: 'rgb(54, 162, 235)',  // Color de la línea de humedad

                    backgroundColor: 'rgba(54, 162, 235, 0.2)',  // Relleno debajo de la línea (color suave)

                    fill: true,  // Rellenar el área debajo de la línea

                    tension: 0.1  // Suavizar la curva de la línea

                }]

            },

            options: {

                responsive: true,  // Hacer el gráfico responsive (se adapta al tamaño de la pantalla)

                scales: {

                    x: {

                        title: {

                            display: true,

                            text: 'Mes y Año'

                        },

                        ticks: {

                            autoSkip: true,  // Evitar que se sobrepongan las etiquetas

                            maxRotation: 45,  // Rota las etiquetas para que no se apilen

                            minRotation: 30

                        }

                    },

                    y: {

                        title: {

                            display: true,

                            text: 'Valor'

                        }

                    }

                }

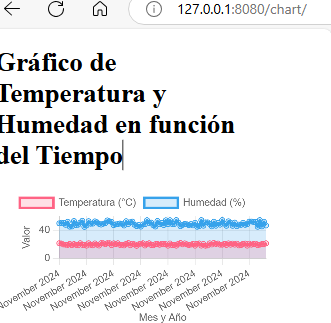
            }

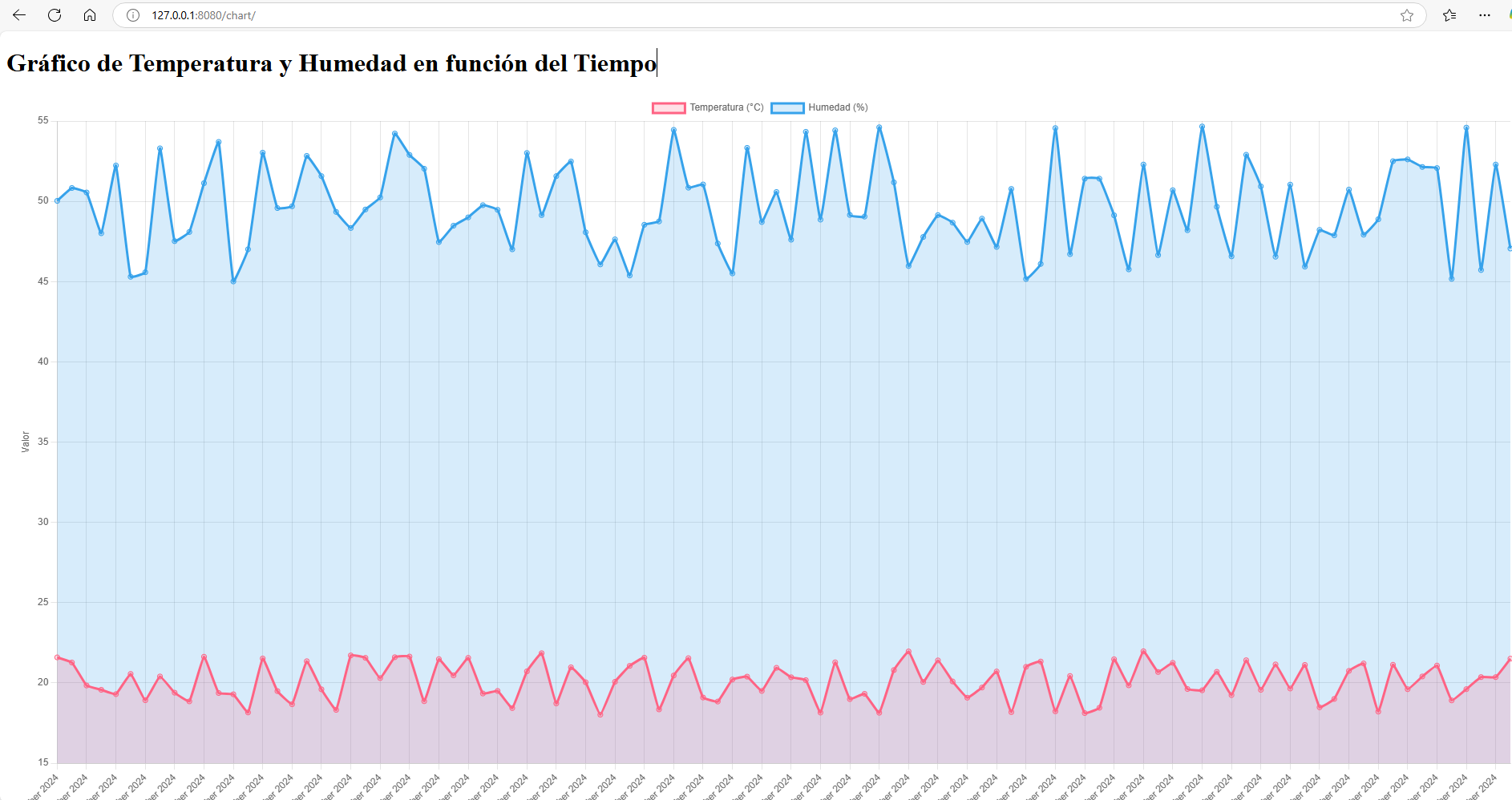
        });

    </script>

</body>

</html>





Descripcion del framework utilizado :

¡Por supuesto! Aquí te dejo una descripción detallada de todas las tecnologías y librerías utilizadas en tu proyecto, ideal para incluir en tu examen:

---

### Tecnologías y Herramientas Utilizadas

1. MySQL:

- Versión: 8.0.x (puedes verificar tu versión con el comando `mysql --version`)

- Descripción: MySQL es un sistema de gestión de bases de datos relacional, que utilizamos para almacenar y consultar los datos de temperatura y humedad. La base de datos `sensordata` contiene tres columnas principales: `id`, `temperature`, `humidity`, y `datetime` para registrar las lecturas de los sensores.

2. Django:

- Versión: 5.1.1 (puedes verificar la versión de Django con el comando `django-admin --version`)

- Descripción: Django es un framework web de alto nivel para Python que facilita la creación de aplicaciones web robustas y escalables. Utilizamos Django para desarrollar la aplicación web que consulta y visualiza los datos de MySQL. En particular, usamos el modelo de datos de Django para interactuar con la base de datos.

3. Python:

- Versión: 3.9.x o superior (verifica la versión con `python --version`)

- Descripción: Python es un lenguaje de programación utilizado para desarrollar la lógica del backend de la aplicación web. Django está construido sobre Python, y utilizamos este lenguaje para gestionar las vistas, los modelos y la configuración de la aplicación web.

4. Chart.js (JavaScript):

- Versión: 3.x (última versión estable en el momento de la implementación)

- Descripción: Chart.js es una librería de JavaScript que se utiliza para crear gráficos interactivos y visualizaciones en aplicaciones web. Usamos esta librería para representar gráficamente los datos de temperatura y humedad a lo largo del tiempo en un gráfico de líneas. Se han utilizado características como el relleno debajo de las líneas (`fill: true`) para crear un área sombreada debajo de las curvas de temperatura y humedad.

5. jQuery (JavaScript):

- Versión: 3.6.x (si se utiliza)

- Descripción: jQuery es una librería de JavaScript que simplifica la manipulación de elementos del DOM (Document Object Model) y la gestión de eventos. Aunque en este caso específico no hemos utilizado jQuery en gran medida, en algunos casos puede ser útil para la interacción con el DOM, especialmente si se requieren efectos o actualizaciones dinámicas.

6. Moment.js (JavaScript):

- Versión: 2.29.1

- Descripción: Moment.js es una librería de JavaScript para el manejo de fechas y horas. La utilizamos para formatear las fechas de las lecturas de temperatura y humedad antes de mostrarlas en el gráfico. Usamos Moment.js para convertir las fechas en un formato adecuado (`Mes Año`), lo que facilita la interpretación del gráfico por parte del usuario.

---

### Proceso de Implementación

1. Base de Datos MySQL:

- Creamos una base de datos llamada `sensordata` para almacenar las lecturas de los sensores. La tabla `sensordata` tiene las siguientes columnas:

- `id`: un identificador único para cada registro (autoincrementable).

- `temperature`: la lectura de la temperatura (tipo `DOUBLE`).

- `humidity`: la lectura de la humedad (tipo `DOUBLE`).

- `datetime`: la fecha y hora de la lectura, con un valor predeterminado de la hora actual (`TIMESTAMP`).

2. Django Backend:

- Creamos un proyecto de Django, configurando una aplicación llamada `sensordata\_app`.

- Dentro de la aplicación, definimos un modelo `SensorData` que representa la tabla de la base de datos MySQL. Django ORM (Object Relational Mapper) facilita la interacción con la base de datos, permitiéndonos realizar consultas sin escribir SQL directamente.

- Implementamos vistas para recuperar los datos de la base de datos y pasarlos a las plantillas HTML para su visualización.

3. Frontend con Chart.js y Moment.js:

- Creamos una página web en HTML donde cargamos los datos obtenidos del servidor Django. Utilizamos Chart.js para generar el gráfico de líneas que muestra la evolución de la temperatura y la humedad a lo largo del tiempo.

- Usamos Moment.js para formatear las fechas y mostrarlas en un formato legible (`Mes Año`), evitando la visualización de las fechas en formato completo, lo que mejora la claridad del gráfico.

---

### Flujo de Trabajo Completo

1. Base de Datos:

- Insertamos 100 registros de datos ficticios de temperatura y humedad con variaciones de tiempo de 1 minuto entre cada lectura. Los datos se guardan en la base de datos `sensordata`.

2. Backend (Django):

- Configuramos el proyecto Django para conectarse a la base de datos MySQL.

- Creamos un modelo `SensorData` que define la estructura de los datos y utilizamos Django ORM para interactuar con la base de datos.

- Creamos vistas que consultan los datos de temperatura y humedad desde la base de datos y los envían a las plantillas HTML en formato JSON.

3. Frontend (HTML, JavaScript, Chart.js):

- Creamos una plantilla HTML que incluye el gráfico generado por Chart.js.

- Utilizamos Moment.js para formatear las fechas de las lecturas antes de enviarlas al gráfico.

- Mostramos los datos en un gráfico interactivo con líneas de temperatura y humedad, con áreas sombreadas debajo de las líneas para mejorar la visualización.

---

### Resúmen de Librerías y Frameworks Utilizados:

1. MySQL: Para almacenar los datos de temperatura y humedad.

2. Django: Framework web de Python para manejar el backend de la aplicación.

3. Python: Lenguaje de programación utilizado para desarrollar el backend de la aplicación.

4. Chart.js: Librería JavaScript para la visualización gráfica de los datos.

5. Moment.js: Librería JavaScript para formatear las fechas y horas.

6. jQuery (opcional): Si es necesario, puede ayudar a interactuar con el DOM.

---

Este desglose te ayudará a mostrar de manera clara y estructurada el flujo de trabajo y las tecnologías utilizadas en tu proyecto, lo cual es perfecto para tu examen.

# PASO PARA CONFIGURAR:

Para configurar correctamente el framework Django y llevar a cabo el proyecto de visualización de datos, los pasos que se llevaron a cabo fueron los siguientes:

### \*\*Pasos para Configurar el Framework Django y el Proyecto\*\*

1. \*\*Instalación de Django y Requisitos Previos\*\*

- \*\*Instalar Python\*\*: Si no se tiene Python instalado, es necesario instalarlo (preferiblemente la versión 3.9 o superior).

- \*\*Instalar Django\*\*:

Ejecutamos el siguiente comando en la terminal para instalar Django:

```bash

pip install django

```

- \*\*Instalar MySQL y MySQL Connector\*\*:

Instalamos MySQL (si no está instalado) y el conector de Python para MySQL para que Django pueda interactuar con la base de datos:

```bash

pip install mysqlclient

```

(Dependiendo del sistema operativo, puede requerir herramientas adicionales para compilar).

2. \*\*Crear un Proyecto Django\*\*

- Creamos el proyecto Django con el siguiente comando:

```bash

django-admin startproject sensordata\_project

```

Esto crea la estructura básica de un proyecto Django, con un directorio principal y los archivos de configuración iniciales.

3. \*\*Configuración de la Base de Datos MySQL\*\*

- En el archivo `settings.py` del proyecto Django, configuramos la conexión a la base de datos MySQL.

- Modificamos la sección de \*\*DATABASES\*\* para apuntar a nuestra base de datos MySQL:

```python

DATABASES = {

'default': {

'ENGINE': 'django.db.backends.mysql',

'NAME': 'sensordata', # Nombre de la base de datos

'USER': 'root', # Usuario de la base de datos

'PASSWORD': 'your\_password', # Contraseña de la base de datos

'HOST': 'localhost', # Host del servidor MySQL

'PORT': '3306', # Puerto de MySQL (puede variar)

}

}

```

- \*\*Crear la base de datos\*\* en MySQL si aún no está creada:

```sql

CREATE DATABASE sensordata;

```

4. \*\*Crear la Aplicación Django (sensordata\_app)\*\*

- Dentro del proyecto Django, creamos una aplicación llamada `sensordata\_app` para gestionar los datos:

```bash

python manage.py startapp sensordata\_app

```

- Registramos esta aplicación en el archivo `settings.py`, bajo la sección `INSTALLED\_APPS`:

```python

INSTALLED\_APPS = [

'django.contrib.admin',

'django.contrib.auth',

'django.contrib.contenttypes',

'django.contrib.sessions',

'django.contrib.messages',

'django.contrib.staticfiles',

'sensordata\_app', # Añadir la nueva app

]

```

5. \*\*Crear el Modelo para la Tabla `sensordata`\*\*

- En el archivo `models.py` de la aplicación `sensordata\_app`, definimos el modelo para los datos de temperatura y humedad que se almacenarán en la tabla `sensordata` de MySQL:

```python

from django.db import models

class SensorData(models.Model):

temperature = models.FloatField()

humidity = models.FloatField()

datetime = models.DateTimeField(auto\_now\_add=True)

def \_\_str\_\_(self):

return f"SensorData(id={self.id}, temperature={self.temperature}, humidity={self.humidity}, datetime={self.datetime})"

```

6. \*\*Migrar el Modelo a la Base de Datos\*\*

- Generamos las migraciones para crear la tabla correspondiente en MySQL:

```bash

python manage.py makemigrations

```

- Aplicamos las migraciones a la base de datos:

```bash

python manage.py migrate

```

7. \*\*Crear una Vista para Mostrar los Datos\*\*

- En el archivo `views.py` de la aplicación, creamos una vista que obtenga los datos de la base de datos y los pase a una plantilla HTML para la visualización:

```python

from django.shortcuts import render

from .models import SensorData

import json

def sensor\_data\_view(request):

data = SensorData.objects.all()

context = {

'data': data,

}

return render(request, 'sensordata\_app/sensor\_data.html', context)

```

8. \*\*Configurar las URLs para la Vista\*\*

- En el archivo `urls.py` de la aplicación `sensordata\_app`, definimos la URL para acceder a la vista de los datos:

```python

from django.urls import path

from . import views

urlpatterns = [

path('', views.sensor\_data\_view, name='sensor\_data'),

]

```

- Luego, en el archivo `urls.py` principal del proyecto, agregamos la aplicación a las rutas del proyecto:

```python

from django.contrib import admin

from django.urls import path, include

urlpatterns = [

path('admin/', admin.site.urls),

path('', include('sensordata\_app.urls')), # Incluir las URLs de la app

]

```

9. \*\*Crear la Plantilla HTML para Mostrar los Datos\*\*

- En el directorio `templates/sensordata\_app/`, creamos el archivo `sensor\_data.html` donde se visualizarán los datos y el gráfico interactivo generado con \*\*Chart.js\*\*:

```html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Sensor Data</title>

<script src="https://cdn.jsdelivr.net/npm/chart.js"></script>

</head>

<body>

<h1>Temperatura y Humedad</h1>

<canvas id="sensorChart" width="400" height="200"></canvas>

<script>

const data = {{ data | json\_script:"sensor\_data" }};

const ctx = document.getElementById('sensorChart').getContext('2d');

const sensorChart = new Chart(ctx, {

type: 'line',

data: {

labels: data.map(d => d.datetime), // Fechas de las lecturas

datasets: [{

label: 'Temperatura',

data: data.map(d => d.temperature),

borderColor: 'rgba(255, 99, 132, 1)',

fill: false,

}, {

label: 'Humedad',

data: data.map(d => d.humidity),

borderColor: 'rgba(54, 162, 235, 1)',

fill: false,

}]

},

options: {

responsive: true,

plugins: {

legend: {

position: 'top',

},

tooltip: {

callbacks: {

title: (tooltipItem) => {

return tooltipItem[0].label;

},

label: (tooltipItem) => {

return tooltipItem.dataset.label + ': ' + tooltipItem.raw;

}

}

}

},

scales: {

x: {

type: 'time',

time: {

unit: 'minute',

tooltipFormat: 'll HH:mm',

}

}

}

}

});

</script>

</body>

</html>

```

10. \*\*Probar el Proyecto\*\*

- Finalmente, ejecutamos el servidor de desarrollo de Django para verificar que todo esté funcionando correctamente:

```bash

python manage.py runserver

```

- Accedemos a la aplicación desde el navegador usando `http://127.0.0.1:8000/` para ver el gráfico interactivo con los datos de temperatura y humedad.

---

### \*\*Resumen de los Pasos Realizados\*\*

1. \*\*Instalación de Python, Django y dependencias\*\*.

2. \*\*Creación de un proyecto Django y configuración de la base de datos MySQL\*\*.

3. \*\*Creación de una aplicación Django (`sensordata\_app`)\*\* y definición del modelo `SensorData`.

4. \*\*Migración de la base de datos y creación de la tabla `sensordata` en MySQL\*\*.

5. \*\*Desarrollo de la vista para mostrar los datos de la base de datos en una página web\*\*.

6. \*\*Configuración de las URLs y plantillas HTML\*\* para mostrar los datos en un gráfico interactivo con \*\*Chart.js\*\*.

7. \*\*Pruebas del servidor\*\* y acceso a la aplicación web.

---

Este conjunto de pasos proporciona una descripción completa del proceso de configuración del proyecto Django con MySQL y la visualización de datos utilizando \*\*Chart.js\*\*.

# PASOS PARA REALIZAR EL GRAFICO\_:

Para realizar el gráfico interactivo en la aplicación web, seguimos una serie de pasos que incluyen la configuración de los datos, la integración de \*\*Chart.js\*\* y la personalización del gráfico para que sea visualmente atractivo y funcione de manera adecuada. A continuación, se detallan los pasos seguidos para implementar el gráfico:

---

### \*\*Pasos para Realizar el Gráfico Interactivo con Chart.js\*\*

1. \*\*Preparar los Datos en Django\*\*

- \*\*Obtener los datos desde la base de datos\*\*:

En primer lugar, consultamos los datos de la tabla `SensorData` desde la base de datos MySQL. Usamos el modelo `SensorData` creado previamente para obtener todas las lecturas de temperatura y humedad, junto con su marca de tiempo.

```python

from django.shortcuts import render

from .models import SensorData

def sensor\_data\_view(request):

data = SensorData.objects.all() # Obtener todos los datos

context = {

'data': data,

}

return render(request, 'sensordata\_app/sensor\_data.html', context)

```

2. \*\*Convertir los Datos a un Formato Usable por JavaScript\*\*

- Para que JavaScript pueda procesar correctamente los datos en la plantilla, los convertimos a formato JSON.

- Utilizamos la función `json\_script` de Django, que convierte los datos en un formato JSON seguro para pasarlos a JavaScript:

```python

context = {

'data': data,

}

return render(request, 'sensordata\_app/sensor\_data.html', context)

```

3. \*\*Crear la Plantilla HTML y Agregar el Elemento del Gráfico\*\*

- En el archivo `sensor\_data.html`, creamos una estructura básica para mostrar el gráfico.

- Usamos un `<canvas>` de HTML, que es donde \*\*Chart.js\*\* dibujará el gráfico:

```html

<canvas id="sensorChart" width="400" height="200"></canvas>

```

4. \*\*Incluir Chart.js en la Plantilla\*\*

- Para crear el gráfico interactivo, agregamos \*\*Chart.js\*\* en nuestra plantilla HTML. Usamos el siguiente CDN para cargar Chart.js:

```html

<script src="https://cdn.jsdelivr.net/npm/chart.js"></script>

```

5. \*\*Configuración del Gráfico en JavaScript\*\*

- En el archivo JavaScript de la plantilla, obtenemos los datos pasados desde Django a través de la variable `data` y los procesamos para que Chart.js pueda utilizarlos.

- Configuramos un gráfico de tipo `line` (línea) para mostrar las lecturas de temperatura y humedad a lo largo del tiempo.

- Creamos el gráfico con la siguiente estructura en JavaScript:

```html

<script>

const data = {{ data | json\_script:"sensor\_data" }}; // Obtener los datos de Django

const ctx = document.getElementById('sensorChart').getContext('2d'); // Obtener el contexto del canvas

const sensorChart = new Chart(ctx, {

type: 'line', // Tipo de gráfico: línea

data: {

labels: data.map(d => d.datetime), // Etiquetas en el eje X (fechas)

datasets: [{

label: 'Temperatura', // Etiqueta para la serie de datos de temperatura

data: data.map(d => d.temperature), // Datos de temperatura

borderColor: 'rgba(255, 99, 132, 1)', // Color de la línea

fill: false, // No llenar el área bajo la línea

}, {

label: 'Humedad', // Etiqueta para la serie de datos de humedad

data: data.map(d => d.humidity), // Datos de humedad

borderColor: 'rgba(54, 162, 235, 1)', // Color de la línea

fill: false, // No llenar el área bajo la línea

}]

},

options: {

responsive: true, // Hacer el gráfico responsive

plugins: {

legend: {

position: 'top', // Posición de la leyenda

},

tooltip: {

callbacks: {

title: (tooltipItem) => {

return tooltipItem[0].label; // Mostrar la fecha en el tooltip

},

label: (tooltipItem) => {

return tooltipItem.dataset.label + ': ' + tooltipItem.raw; // Mostrar el valor de la temperatura o humedad en el tooltip

}

}

}

},

scales: {

x: {

type: 'time', // Eje X tipo tiempo

time: {

unit: 'minute', // Unidad de tiempo: minuto

tooltipFormat: 'll HH:mm', // Formato del tooltip en el eje X

}

}

}

}

});

</script>

```

6. \*\*Configuración del Eje X con Fechas\*\*

- Configuramos el eje X como un eje de tiempo (`type: 'time'`), de modo que cada punto de datos en el gráfico esté asociado con una fecha y hora específica.

- Usamos la opción `time.unit: 'minute'` para mostrar las fechas y horas en intervalos de minutos.

7. \*\*Rellenar el Área Bajo la Curva\*\*

- Para mejorar la visualización del gráfico, habilitamos el relleno del área bajo las curvas de temperatura y humedad, lo cual hace que el gráfico sea más visual y fácil de interpretar.

```javascript

fill: true, // Rellenar el área bajo la curva

```

8. \*\*Hacer el Gráfico Responsive\*\*

- Para que el gráfico se ajuste automáticamente al tamaño de la pantalla en diferentes dispositivos (responsive), configuramos la opción `responsive: true` dentro de las opciones del gráfico.

9. \*\*Probar el Gráfico en el Navegador\*\*

- Finalmente, después de haber configurado el gráfico, ejecutamos el servidor Django (`python manage.py runserver`) y accedemos al navegador para visualizar el gráfico generado.

- En el gráfico, podemos ver las lecturas de \*\*temperatura\*\* y \*\*humedad\*\* a lo largo del tiempo, con las fechas en el eje X y las mediciones de temperatura y humedad en el eje Y.

---

### \*\*Resumen del Proceso para Crear el Gráfico\*\*

1. \*\*Obtener y procesar los datos\*\* desde la base de datos usando Django.

2. \*\*Convertir los datos a formato JSON\*\* y pasarlos a JavaScript para su procesamiento.

3. \*\*Crear el gráfico interactivo\*\* usando \*\*Chart.js\*\*.

4. \*\*Configurar el gráfico\*\* para que sea de tipo línea, con el área bajo la curva rellenada y ajustado para ser responsive.

5. \*\*Configurar los ejes\*\*: eje X como tiempo y eje Y para las mediciones de temperatura y humedad.

6. \*\*Incluir las opciones de leyenda, tooltip y ajuste de escala\*\* para mejorar la visualización.

Con estos pasos, logramos un gráfico interactivo que muestra las variaciones de temperatura y humedad a lo largo del tiempo de manera visualmente atractiva y fácil de entender.