



ESCUELA DE  
INGENIERÍA EN CIENCIAS Y SISTEMAS  
FACULTAD DE INGENIERÍA  
UNIVERSIDAD DE SAN CARLOS DE GUATEMALA



**Día, Fecha:**

Jueves, 10/10/2024

**Hora de inicio:**

10:40 - 12:20

# Introducción a la Programación y Computación 2 [P]

Denilson Florentín de León Aguilar



Nombre de la actividad:	Calificación DTT - DSI
Cantidad de participantes:	4
Doy fe que esta actividad está planificada en dtt (Sí/No):	<input checked="" type="checkbox"/> Sí

Hora de inicio:	10:50
Hora de fin:	11:00
Duración (min):	10

**Participantes: llenar las siguientes cajas de texto (tomar información del chat del meet)**

ad



# Recordatorio Grabar Clase

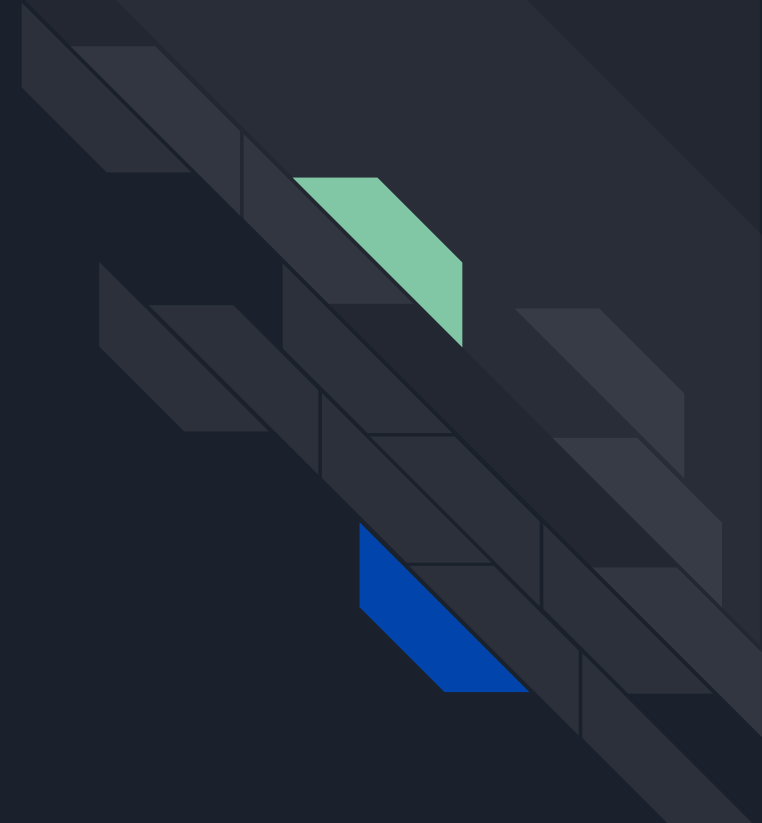




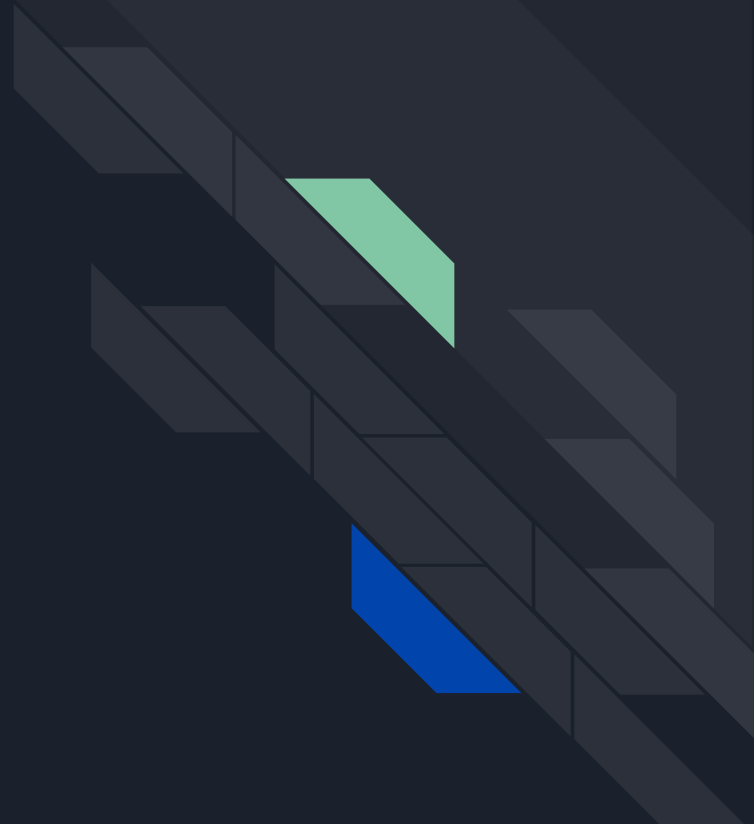
## Contenido clase 11

- Preguntas proyecto 3
- Preguntas práctica 3
- 8. Acceso a Datos Web
  - 8.10. Interfaces de programación de aplicaciones
  - 8.11. API
  - 8.12. Seguridad y uso de API's
- Configuración Django
  - Explicación funcionalidades:
  - vistas
  - url
  - configuración
- Ejemplo de Flask a Django

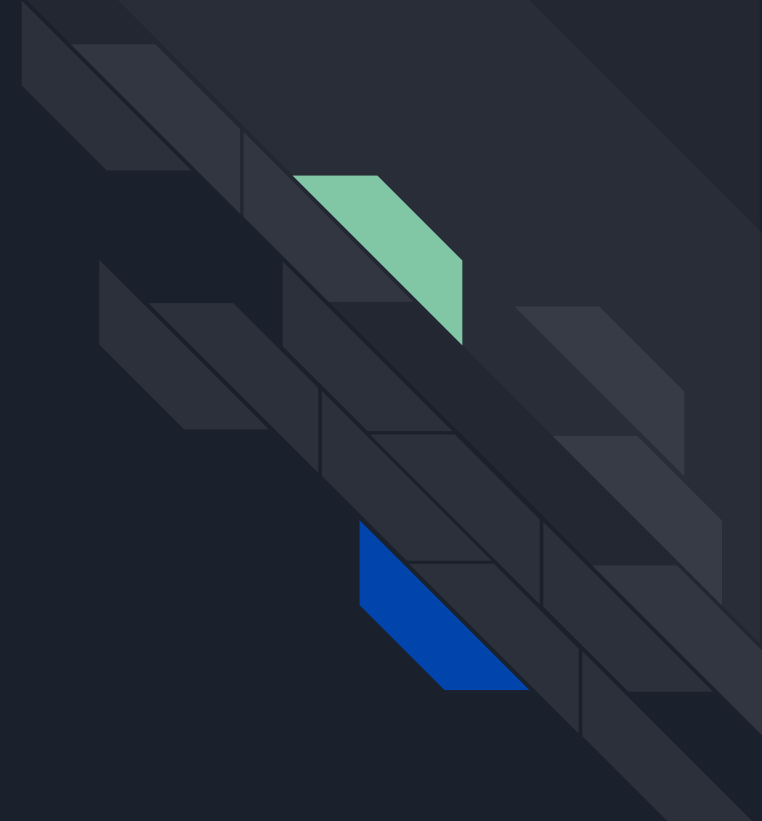
# Preguntas Proyecto 3



# Preguntas práctica 3

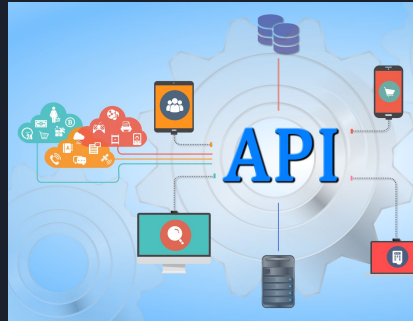


## 8. Acceso a datos web



## 8.10. Interfaces de Programación de Aplicaciones (API)

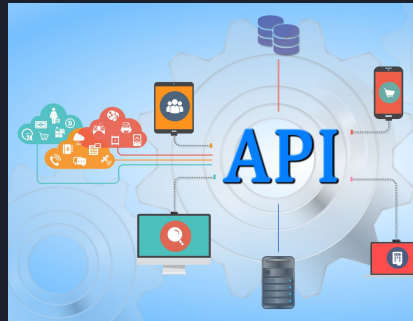
Las Interfaces de Programación de Aplicaciones, comúnmente conocidas como API, son conjuntos de reglas y protocolos que permiten que diferentes aplicaciones se comuniquen entre sí. En el contexto del desarrollo web, una API puede ser proporcionada por un servicio en línea para permitir que los desarrolladores accedan y utilicen ciertas funcionalidades o datos de esa plataforma. Por ejemplo, las redes sociales como Facebook, Twitter o Instagram pudieran ofrecer APIs que permiten a los desarrolladores acceder a datos de usuario, publicar contenido y realizar otras acciones dentro de esas plataformas desde sus propias aplicaciones.



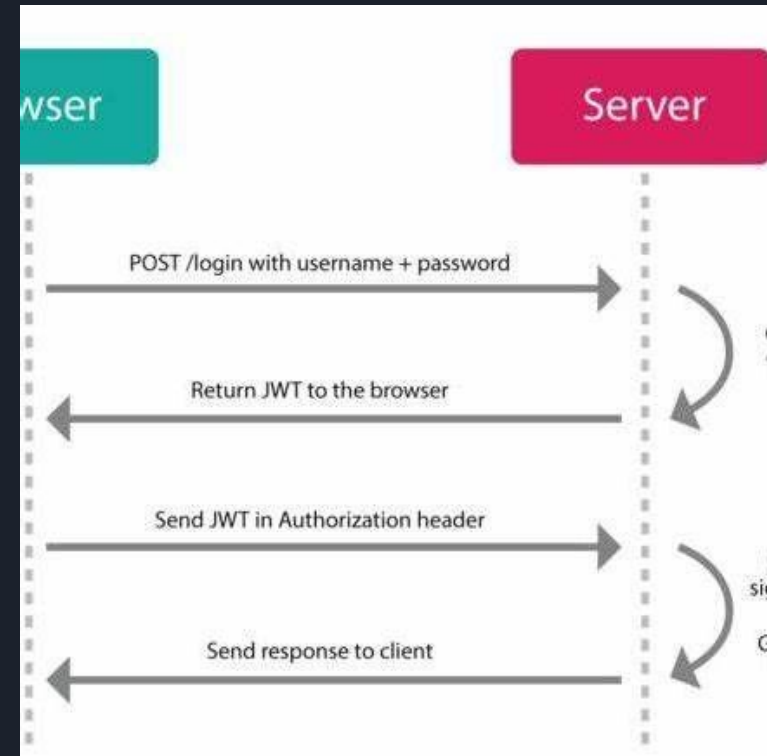
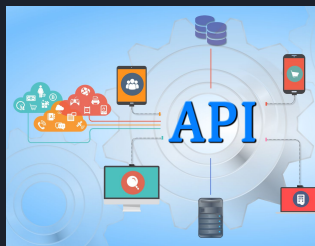
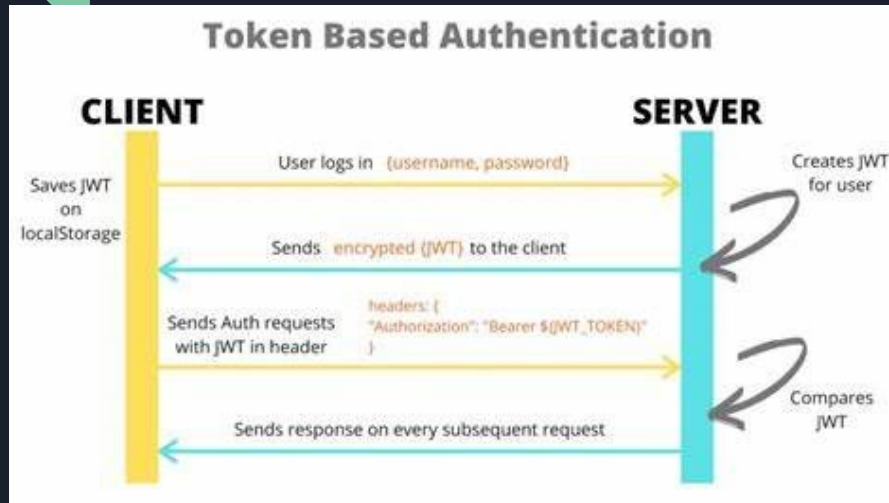


## 8.11. Seguridad y Uso de API's

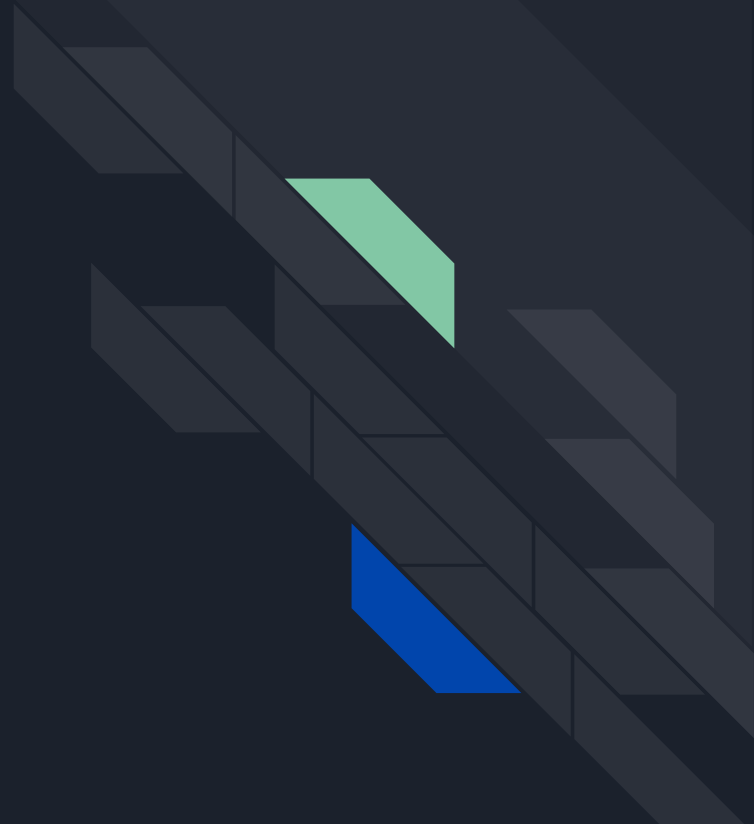
La seguridad en el uso de APIs es fundamental para garantizar la integridad, confidencialidad y disponibilidad de los datos y servicios que proporcionan. Esto implica implementar medidas de autenticación, autorización y cifrado adecuadas para proteger tanto la API como los datos que maneja. Además, es importante seguir buenas prácticas de diseño de API para garantizar que sean fáciles de usar, eficientes y seguras. Algunas medidas comunes de seguridad en el uso de APIs incluyen el uso de tokens de acceso, la limitación de solicitudes por parte de un cliente y la validación de datos de entrada para prevenir ataques como la inyección de código.



## 8.11. Seguridad y Uso de API's



# Ejemplos Flask y Node



# Comunicación entre servicios

<https://genderize.io/documentation#localization>

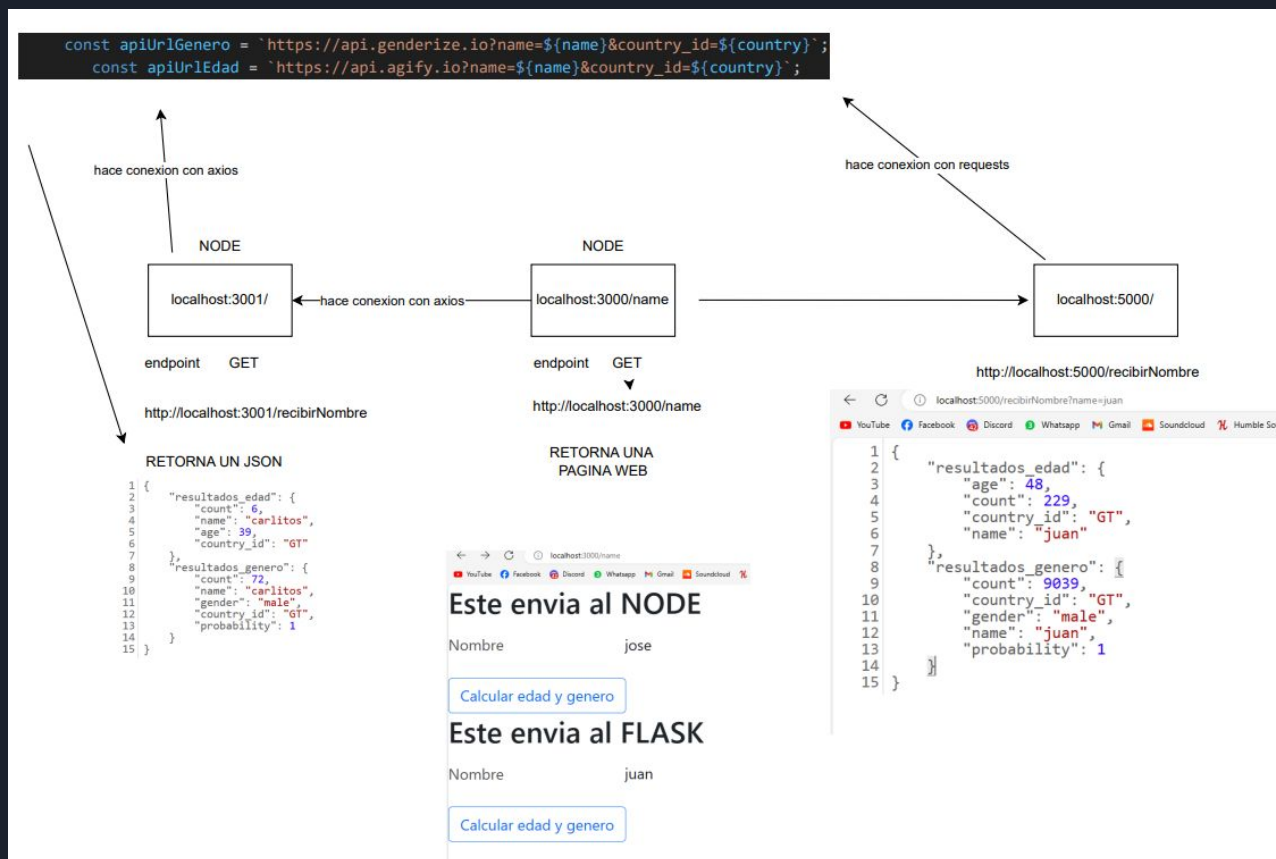
<https://agify.io/documentation#localization>

# Flujo - Salida de las API de flask y node

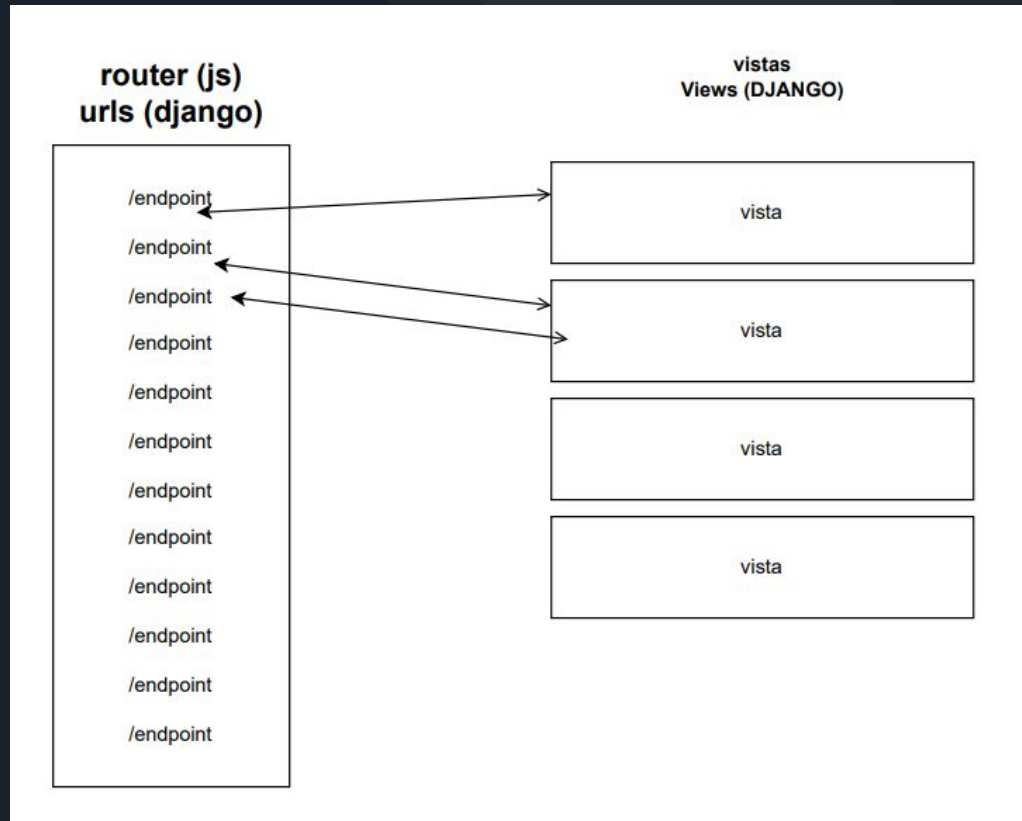
retornan un json

```
1  {  
2    "resultados_edad": {  
3      "age": 48,  
4      "count": 229,  
5      "country_id": "GT",  
6      "name": "juan"  
7    },  
8    "resultados_genero": {  
9      "count": 9039,  
10     "country_id": "GT",  
11     "gender": "male",  
12     "name": "juan",  
13     "probability": 1  
14   }  
15 }
```

# Flujo final



# Django



# Recordatorio Captura de pantalla





# Ejemplos Django Configuration

The background of the slide features a series of dark gray, three-dimensional rectangular planes that recede into the distance, creating a sense of depth. A bright green parallelogram is positioned on one of the middle planes, and a bright blue parallelogram is on a lower plane, both adding a pop of color to the monochromatic scheme.

## 7. DJANGO - Instalación y Creación Proyecto

Preparar virtual environment (venv)

```
python -m venv myVenv
```

Ejecutar en linux y windows

```
# En Linux  
source myVenv/Scripts/activate
```

```
# En windows  
Set-ExecutionPolicy -Scope Process -ExecutionPolicy Bypass  
  
.\myVenv\Scripts\activate
```

```
+ FullyQualifiedPath : UnauthorizedAccess  
PS C:\Users\denii\OneDrive\Documents\Projects\IPC2_S1_2024_Examples\django> Set-ExecutionPolicy -Scope Process -ExecutionPolicy Bypass  
PS C:\Users\denii\OneDrive\Documents\Projects\IPC2_S1_2024_Examples\django> .\myVenv\Scripts\activate  
(myVenv) PS C:\Users\denii\OneDrive\Documents\Projects\IPC2_S1_2024_Examples\django> █
```



## 7. DJANGO - Instalación y Creación Proyecto

### Paso 1: Instalación de Django

Abre tu terminal y ejecuta el siguiente comando para instalar Django utilizando pip:

```
pip install django
```

### Paso 2: Crear un nuevo proyecto Django

Una vez instalado Django, puedes crear un nuevo proyecto ejecutando el siguiente comando en tu terminal:

```
django-admin startproject nombre_del_proyecto
```

```
cd nombre del proyecto  
django-admin startapp nombre_de_app
```



## 7. DJANGO - Creación aplicación en el proyecto

### Paso 3: Crear una aplicación

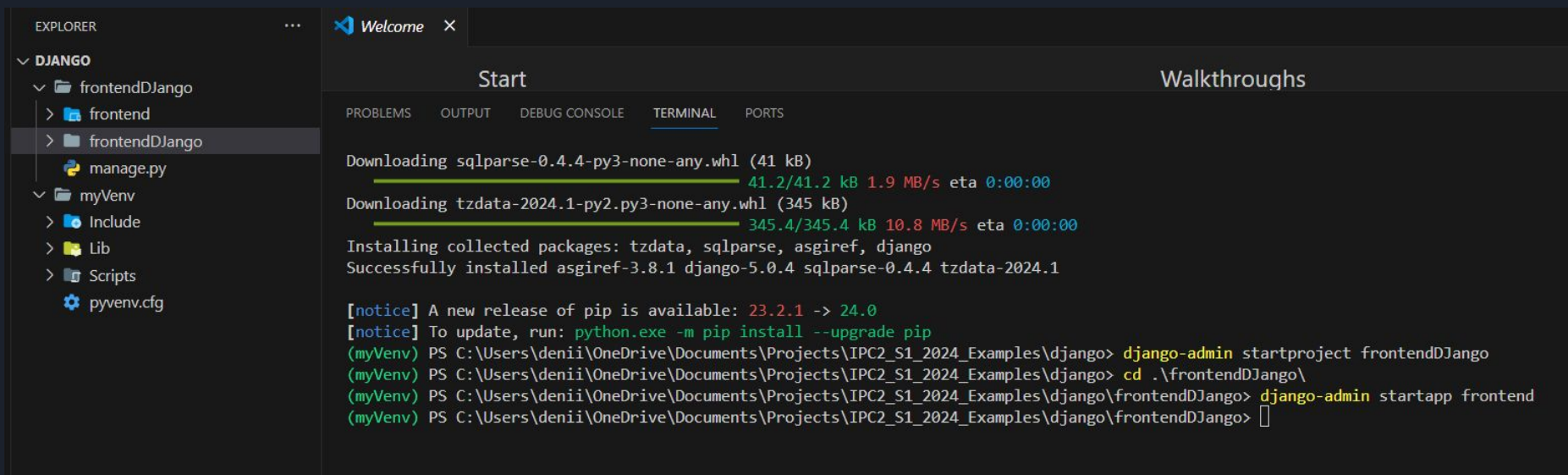
Una vez creado el proyecto Django.

Crea una aplicación Django dentro de tu proyecto ejecutando el siguiente comando en tu terminal (asegúrate de estar en la raíz de tu proyecto Django):

```
django-admin startapp nombre_de_app
```

```
python manage.py startapp nombre_de_app
```

## 7. DJANGO - Instalación y Creación Proyecto



EXPLORER

▼ DJANGO

- ▼ frontendDjango
  - > frontend
  - > frontendDjango
    - manage.py
- ▼ myVenv
  - > Include
  - > Lib
  - > Scripts
  - pyvenv.cfg

Welcome X

Start Walkthroughs

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Downloading sqlparse-0.4.4-py3-none-any.whl (41 kB)  
41.2/41.2 kB 1.9 MB/s eta 0:00:00

Downloading tzdata-2024.1-py2.py3-none-any.whl (345 kB)  
345.4/345.4 kB 10.8 MB/s eta 0:00:00

Installing collected packages: tzdata, sqlparse, asgiref, django  
Successfully installed asgiref-3.8.1 django-5.0.4 sqlparse-0.4.4 tzdata-2024.1

[notice] A new release of pip is available: 23.2.1 -> 24.0  
[notice] To update, run: python.exe -m pip install --upgrade pip

(myVenv) PS C:\Users\denii\OneDrive\Documents\Projects\IPC2\_S1\_2024\_Examples\django> django-admin startproject frontendDjango

(myVenv) PS C:\Users\denii\OneDrive\Documents\Projects\IPC2\_S1\_2024\_Examples\django> cd .\frontendDjango\

(myVenv) PS C:\Users\denii\OneDrive\Documents\Projects\IPC2\_S1\_2024\_Examples\django\frontendDjango> django-admin startapp frontend

(myVenv) PS C:\Users\denii\OneDrive\Documents\Projects\IPC2\_S1\_2024\_Examples\django\frontendDjango>



## 7. DJANGO - Migrate

El comando `python manage.py migrate` en Django se utiliza para aplicar las migraciones pendientes al esquema de la base de datos. Las migraciones son archivos de Python generados automáticamente por Django que representan los cambios en los modelos de datos. Cuando defines modelos en Django y realizas cambios en ellos, como agregar campos o modificar relaciones, necesitas crear migraciones y luego aplicarlas a la base de datos para reflejar esos cambios en el esquema de la base de datos.

```
python manage.py migrate
```



## 7. DJANGO - Migrate

El proceso generalmente involucra los siguientes pasos:

- Crear o modificar modelos en tu aplicación Django.
- Generar migraciones usando el comando `python manage.py makemigrations`. Esto crea archivos de migración en el directorio `migrations` de cada aplicación Django.
- Aplicar las migraciones pendientes a la base de datos usando el comando `python manage.py migrate`.

```
python manage.py migrate
```

## 7. DJANGO - Migrate

```
python manage.py migrate
```

```
(myVenv) PS C:\Users\denii\OneDrive\Documents\Projects\IPC2_S1_2024_Examples\django\fr
ontendDjango> python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying admin.0003_logentry_add_action_flag_choices... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying auth.0007_alter_validators_add_error_messages... OK
  Applying auth.0008_alter_user_username_max_length... OK
  Applying auth.0009_alter_user_last_name_max_length... OK
  Applying auth.0010_alter_group_name_max_length... OK
  Applying auth.0011_update_proxy_permissions... OK
  Applying auth.0012_alter_user_first_name_max_length... OK
  Applying sessions.0001_initial... OK
(myVenv) PS C:\Users\denii\OneDrive\Documents\Projects\IPC2_S1_2024_Examples\django\fr
ontendDjango>
(myVenv) PS C:\Users\denii\OneDrive\Documents\Projects\IPC2_S1_2024_Examples\django\fr
ontendDjango>
```





## 7. DJANGO - Configurar usuario

Configurar el administrador (admin), no es obligatorio, pero es una buena práctica para administrar fácilmente los datos de tu aplicación durante el desarrollo. Configurar el administrador te permite crear, leer, actualizar y eliminar (CRUD) registros de tus modelos directamente desde una interfaz de administración web. Esto puede ser útil, especialmente durante el desarrollo y las pruebas de tu aplicación Django.

```
# Creamos usuario administrador (datos ejemplo, configurar prudentemente)
$ python manage.py createsuperuser
  > Email Address: admin@admin.com
  > Password: admin
  > Repeat Password Again: admin
```

## 7. DJANGO - Configurar usuario

Configurar el administrador (admin), no es obligatorio, pero es una buena práctica para administrar fácilmente los datos de tu aplicación durante el desarrollo. Configurar el administrador te permite crear, leer, actualizar y eliminar (CRUD) registros de tus modelos directamente desde una interfaz de administración web. Esto puede ser útil, especialmente durante el desarrollo y las pruebas de tu aplicación Django.

```
(myVenv) PS C:\Users\denii\OneDrive\Documents\Projects\IPC2_S1_2024_Examples\django\frontendDJango> python manage.py createsuperuser
Username (leave blank to use 'denii'): admin
Email address: admin@admin.com
Password:
Password (again):
The password is too similar to the username.
This password is too short. It must contain at least 8 characters.
This password is too common.
Bypass password validation and create user anyway? [y/N]: y
Superuser created successfully.
(myVenv) PS C:\Users\denii\OneDrive\Documents\Projects\IPC2_S1_2024_Examples\django\frontendDJango>
```



## 7. DJANGO - Ejecutar

### Paso 4: Ejecutar el servidor de desarrollo

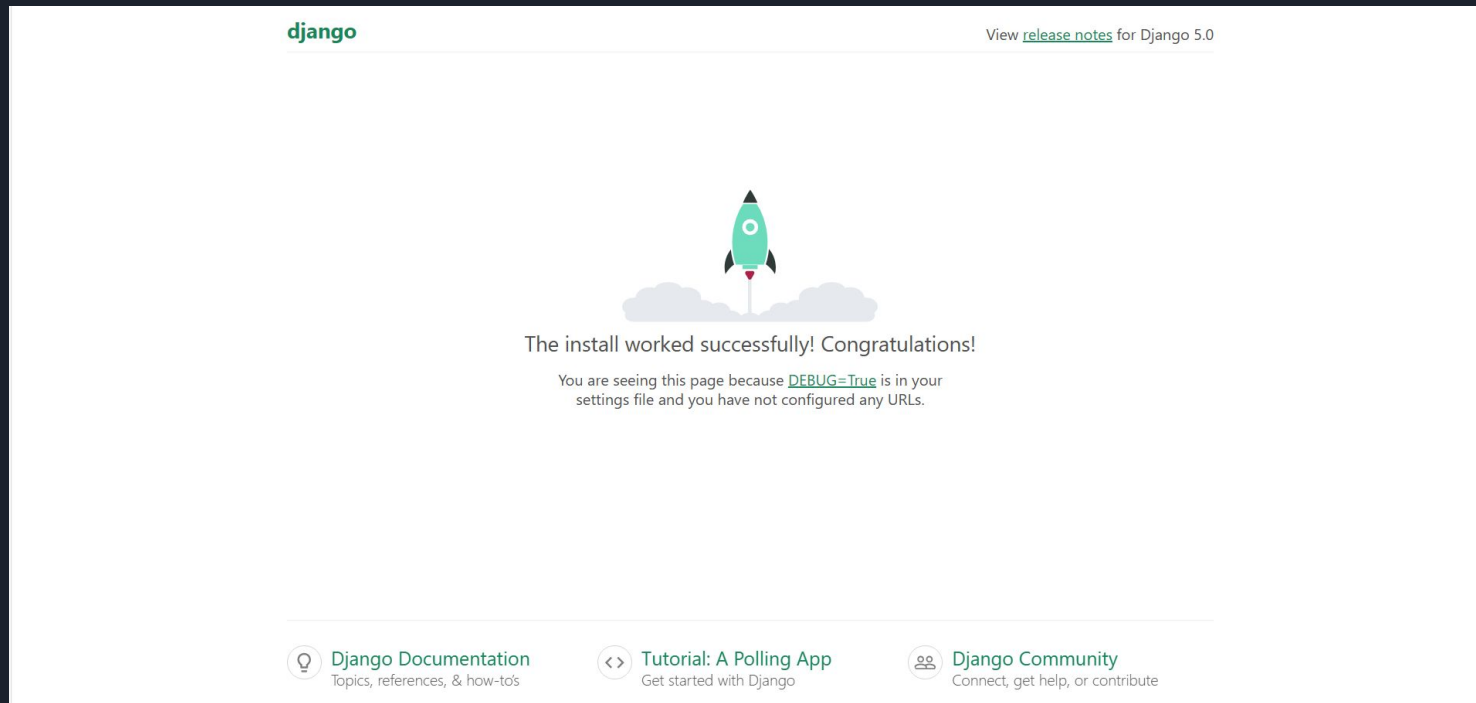
Una vez que hayas migrado tu proyecto Flask a Django, puedes ejecutar el servidor de desarrollo de Django para probar tu aplicación. Ejecuta el siguiente comando en tu terminal (asegúrate de estar en la raíz de tu proyecto Django):

```
python manage.py runserver
```

Esto iniciará el servidor de desarrollo de Django, y podrás acceder a tu aplicación desde tu navegador web visitando la dirección `http://localhost:8000`.

## 7. DJANGO - Ejecutar

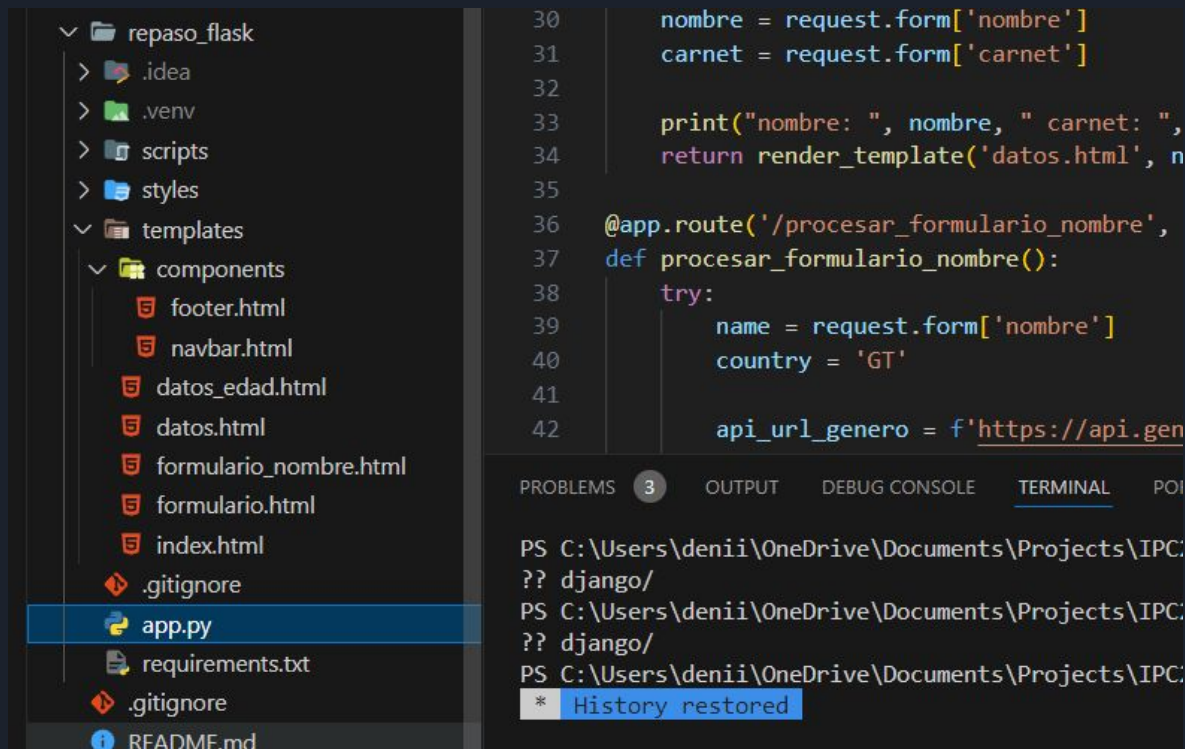
```
python manage.py runserver
```



# Migrar Ejemplo repaso de Flask (frontend) hacia Django



## 7. DJANGO - Flask a migrar



The screenshot displays a code editor interface. On the left, a file explorer shows the project structure for 'repaso\_flask'. The files listed are:

- repaso\_flask
  - .idea
  - .venv
  - scripts
  - styles
  - templates
    - components
      - footer.html
      - navbar.html
      - datos\_edad.html
      - datos.html
      - formulario\_nombre.html
      - formulario.html
      - index.html
    - .gitignore
    - app.py
    - requirements.txt
    - .gitignore
    - README.md

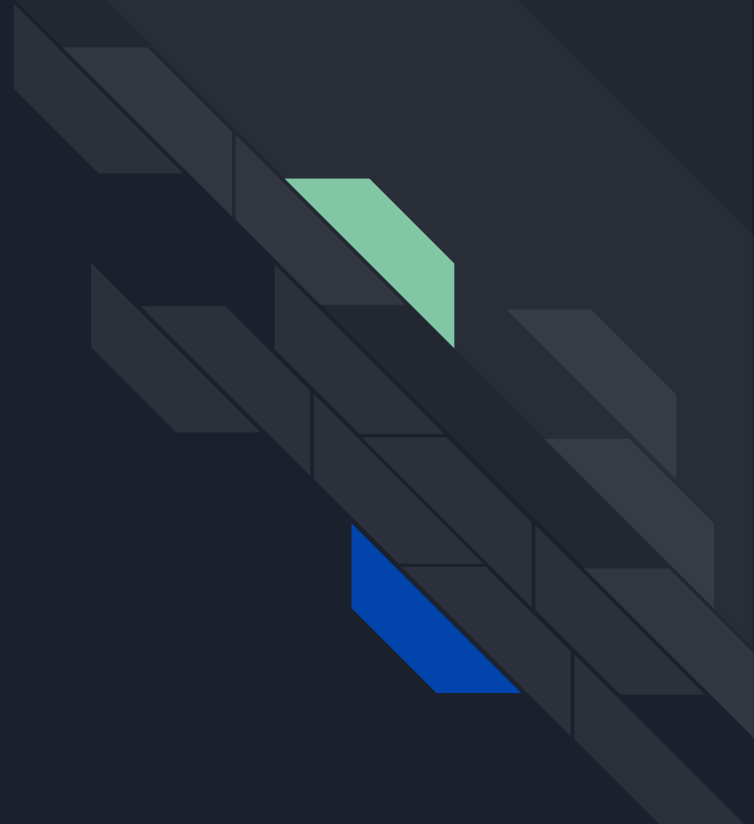
On the right, the code editor shows the following Python code:

```
30 nombre = request.form['nombre']
31 carnet = request.form['carnet']
32
33 print("nombre: ", nombre, " carnet: ",
34       return render_template('datos.html', n
35
36 @app.route('/procesar_formulario_nombre',
37 def procesar_formulario_nombre():
38     try:
39         name = request.form['nombre']
40         country = 'GT'
41
42         api_url_genero = f'https://api.gen
```

At the bottom, the terminal window shows the following commands and output:

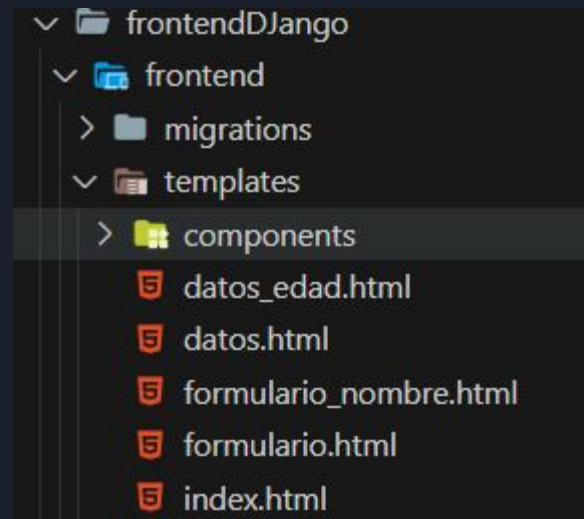
```
PS C:\Users\denii\OneDrive\Documents\Projects\IPC:
?? django/
PS C:\Users\denii\OneDrive\Documents\Projects\IPC:
?? django/
PS C:\Users\denii\OneDrive\Documents\Projects\IPC:
* History restored
```

# Configure Templates



## 7. DJANGO - Templates

Creamos el directorio templates y  
agregamos los archivos html necesarios

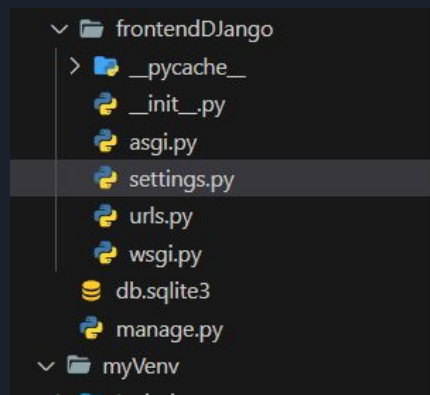




## 7. DJANGO - Settings project

agregamos la app creada en el archivo settings.py de la carpeta project dentro de project

En este ejemplo sería “frontend”

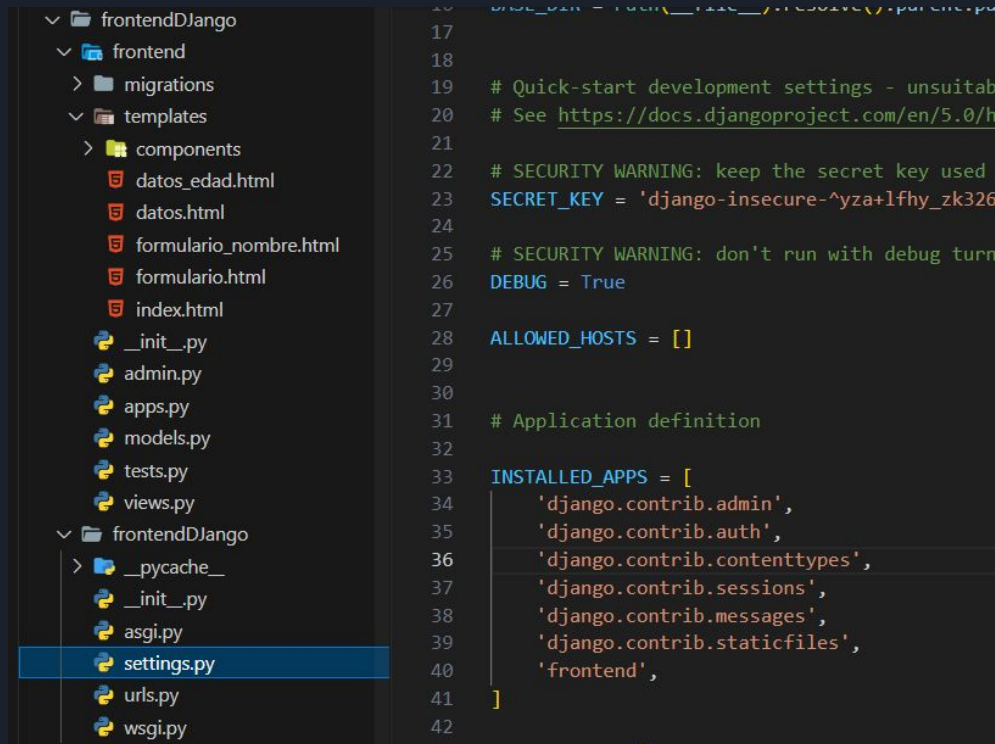


```
30
31 # Application definition
32
33 INSTALLED_APPS = [
34     'django.contrib.admin',
35     'django.contrib.auth',
36     'django.contrib.contenttypes',
37     'django.contrib.sessions',
38     'django.contrib.messages',
39     'django.contrib.staticfiles',
40     'myApp'
41 ]
42
```

## 7. DJANGO - Settings project

agregamos la app creada en el archivo settings.py de la carpeta project dentro de project

En este ejemplo sería “frontend”

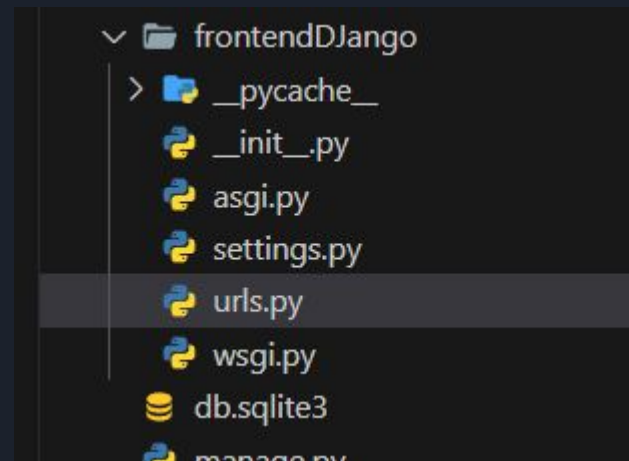


```
16 BASE_DIR = Path(__file__).resolve().parent.parent
17
18
19 # Quick-start development settings - unsuitab
20 # See https://docs.djangoproject.com/en/5.0/h
21
22 # SECURITY WARNING: keep the secret key used
23 SECRET_KEY = 'django-insecure-^yza+lfhy_zk326
24
25 # SECURITY WARNING: don't run with debug turn
26 DEBUG = True
27
28 ALLOWED_HOSTS = []
29
30
31 # Application definition
32
33 INSTALLED_APPS = [
34     'django.contrib.admin',
35     'django.contrib.auth',
36     'django.contrib.contenttypes',
37     'django.contrib.sessions',
38     'django.contrib.messages',
39     'django.contrib.staticfiles',
40     'frontend',
41 ]
42
```

## 7. DJANGO - URLS project

```
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include("myApp.urls"))
]
```



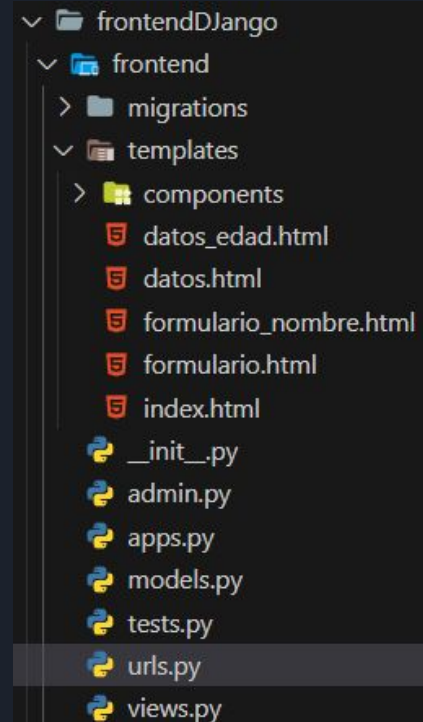
En nuestro ejemplo, myApp se llama “frontend”

## 7. DJANGO - URLS App

```
from . import views
from django.urls import path

urlpatterns = [
    path("", views.viewName, name="home"),
]
```

Creamos el archivo "urls.py" que referenciamos anteriormente. Además, en este caso llamaremos a index.



## 7. DJANGO - VIEWS App

```
from django.shortcuts import render

def index(request):
    context = {}
    return render(request, "index.html", context)
```

La dirección a usar en “index.html” en este caso tiene que ir relativa al directorio /templates

