

ORGANIZACIÓN CONTABLE

TRABAJO PRÁCTICO N°3

“Django”

GRUPO: ERROR-404

INTEGRANTES:

- Ahumada, Brian
 - Alancay, Abel Matias
 - Alsina, Maximiliano Gabriel
 - Berrini, Alejandro
 - Calle, Sonia
 - Chavez, Rodrigo
 - Costa, Maria Eugenia
 - Navarro, Lucas
 - Sanguinetti Flores ,Pablo
-

TRABAJO A REALIZAR EN EQUIPO Y ENTREGAR

CAPTURA DE PANTALLAS

PRIMERA PARTE:

- 1 - Instalación de DJANGO
 - 2 - Introducciones a todas las partes clave de Django que necesitará saber:
 - 3 - ¿ Cómo instalar Django?
 - a. Instalar Python
 - b. Instala Apache y mod_wsgi
 - c. Pon en marcha tu base de datos
 - d. Instalar el código Django
-

2 - Introducciones a todas las partes clave de Django que necesitará saber:

1. **Modelos:** Los modelos son la representación de las tablas en la base de datos y definen los campos y relaciones entre ellos.
2. **Vistas:** Las vistas son las funciones que procesan las solicitudes del usuario y devuelve una respuesta.
3. **Plantillas:** Las plantillas son archivos HTML que se utilizan para renderizar la información en el navegador.

4. **URLconf**: El archivo URLconf es donde se definen las rutas de URL para cada vista.

5. **Middleware**: El middleware es una capa intermedia entre el servidor web y la aplicación Django que permite realizar tareas como autenticación, compresión de respuestas, etc.

6. **Formularios**: Los formularios permiten a los usuarios enviar datos a través de la aplicación y son utilizados para validar y procesar los datos recibidos.

7. **Administrador**: El administrador es una interfaz web preconstruida que permite gestionar los modelos de la base de datos desde el navegador.


8. **ORM (Object-Relational Mapping)**: El ORM es una herramienta que permite interactuar con la base de datos utilizando objetos Python en lugar de SQL directamente.

9. **Migraciones**: Las migraciones son archivos generados automáticamente por Django que permiten actualizar la estructura de la base de datos sin perder los datos existentes.

10. **Contexto**: El contexto es un diccionario que contiene variables que se pasan a las plantillas para ser renderizadas en el navegador

3 - ¿ Cómo instalar Django?

3 - A - Instalar Python



The screenshot shows the Python.org website. The top navigation bar includes links for Python, PSF, Docs, PyPI, Jobs, and Community. Below this is a search bar and a 'Socialize' button. The main content area features a large banner with the text 'Download the latest version for Windows' and a button to 'Download Python 3.11.3'. To the right of the banner is an illustration of two parachutes carrying boxes. Below the banner, there is a section titled 'Active Python Releases' with a link to the Python Developer's Guide. This section contains a table with the following data:

Python version	Maintenance status	First released	End of support	Release schedule
3.11	bugfix	2022-10-24	2027-10	PEP 664
3.10	bugfix	2021-10-04	2026-10	PEP 619
3.9	security	2020-10-05	2025-10	PEP 596
3.8	security	2019-10-14	2024-10	PEP 569
3.7	security	2018-06-27	2023-06-27	PEP 537

Below the table, there is a link 'Looking for a specific release?' and a note 'Python releases by version number'.

Desde la página de descargas de Python, elegimos el instalador para la versión de 32 bits o 64 bits según corresponda. El instalador web es una pequeña descarga inicial y descargará automáticamente los componentes requeridos según sea necesario.

El instalador fuera de línea incluye los componentes necesarios para una instalación predeterminada y solo requiere una conexión a Internet para funciones opcionales.

Después de iniciar el instalador, se puede seleccionar una de dos opciones:



Si selecciona '**Instalar ahora**':

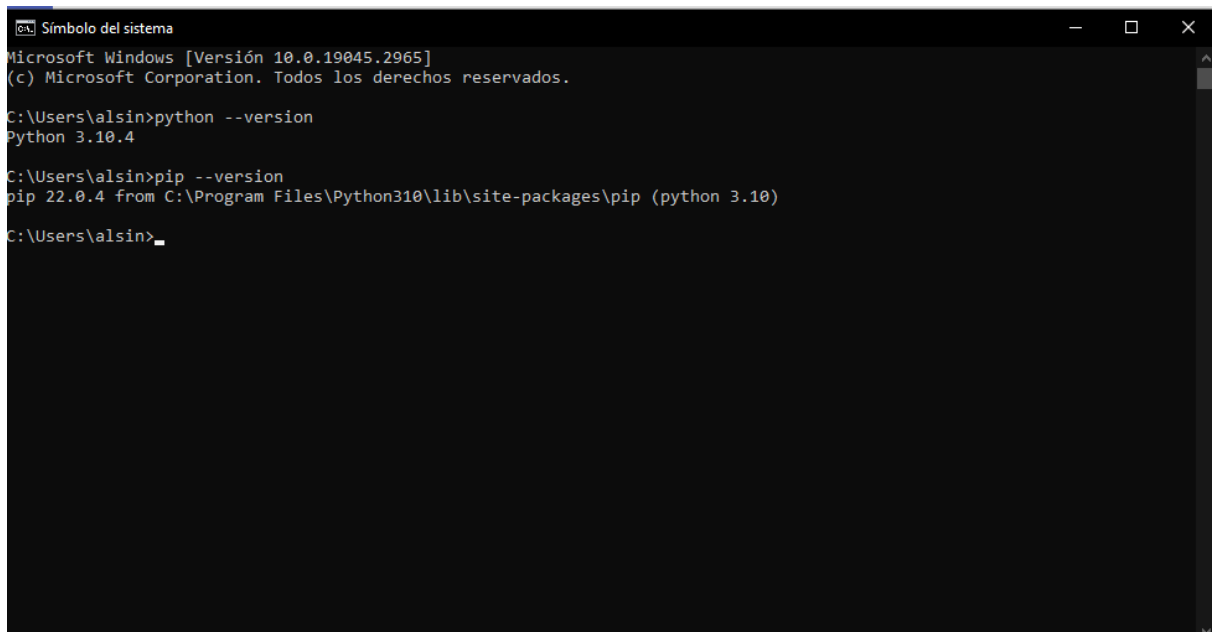
- No necesitará ser administrador (a menos que se requiera una actualización del sistema para C Runtime Library o instale Python Launcher para Windows para todos los usuarios)
- Python se instalará en su directorio de usuario
- Python Launcher para Windows se instalará de acuerdo con la opción en la parte inferior de la primera página
- Se instalarán la biblioteca estándar, el conjunto de pruebas, el lanzador y pip
- Si se selecciona, el directorio de instalación se agregará a su RUTA(PATH)

- Los accesos directos solo serán visibles para el usuario actual

Si selecciona '**Personalizar instalación**', podrá seleccionar las funciones que desea instalar, la ubicación de la instalación y otras opciones o acciones posteriores a la instalación. Para instalar símbolos de depuración o binarios, deberá utilizar esta opción.

Para realizar una instalación para todos los usuarios, debe seleccionar '**Personalizar instalación**'. En este caso:

- Es posible que deba proporcionar credenciales administrativas o aprobación
- Python se instalará en el directorio Archivos de programa
- Python Launcher para Windows se instalará en el directorio de Windows
- Se pueden seleccionar características opcionales durante la instalación
- La biblioteca estándar se puede precompilar en bytecode
- Si se selecciona, el directorio de instalación se agregará a la RUTA del sistema
- Los accesos directos están disponibles para todos los usuarios

A screenshot of a Windows Command Prompt window titled 'Símbolo del sistema'. The window shows the following text: 'Microsoft Windows [Versión 10.0.19045.2965]', '(c) Microsoft Corporation. Todos los derechos reservados.', 'C:\Users\alsin>python --version', 'Python 3.10.4', 'C:\Users\alsin>pip --version', 'pip 22.0.4 from C:\Program Files\Python310\lib\site-packages\pip (python 3.10)', and 'C:\Users\alsin>'.

```
Símbolo del sistema
Microsoft Windows [Versión 10.0.19045.2965]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\alsin>python --version
Python 3.10.4

C:\Users\alsin>pip --version
pip 22.0.4 from C:\Program Files\Python310\lib\site-packages\pip (python 3.10)

C:\Users\alsin>
```

Python ya instalado junto con Pip(otro programa necesario) que esta contenido dentro de los archivos que se instalaron con python.

3 - B - Instalar Apache

Usaremos la versión **Apache 2.4** de la página del lounge, descargable desde [aquí](#).

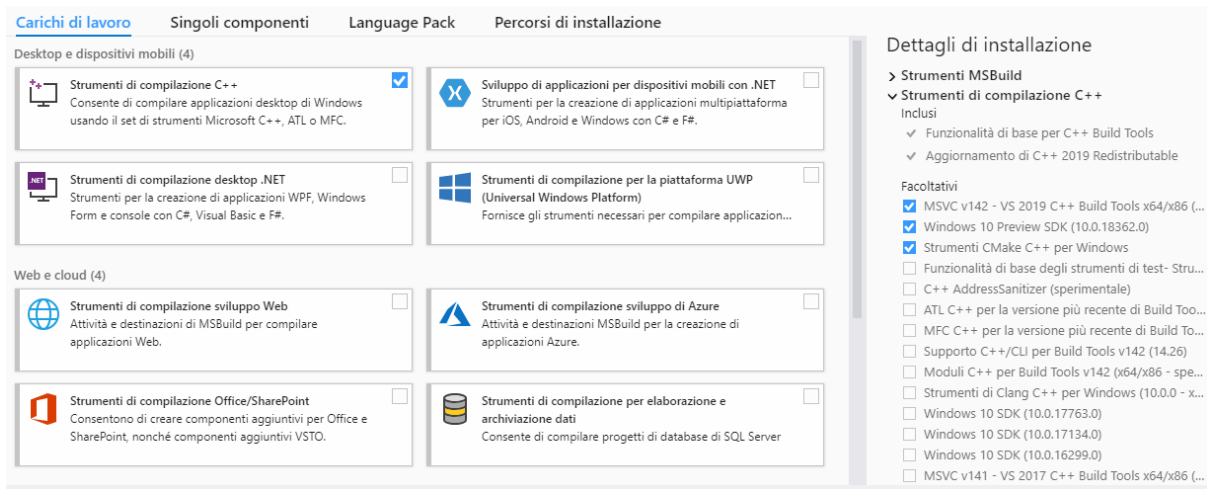
Extraiga el archivo **ZIP** a la raíz del disco **C:/**, de modo que obtenga este resultado **C:/Apache24**.

Instalar Microsoft Build Tools

En Windows es necesario compilar el módulo **mod_wsgi** para poder utilizarlo y para ello es necesario tener instalado Microsoft Build Tools.

La instalación requiere aproximadamente 4 gigabytes y puede descargar el instalador [aquí](#).

Screenshot de la configuración requerida durante la instalación.



3 - B - Instalar mod_wsgi

La instalación del modelo **mod_wsgi** y su configuración utilizando el método **WSGIDaemonProcess** de Apache disponible en Linux no es compatible con Windows.

Windows espera que mod_wsgi se compile utilizando **Microsoft Build Tools**.

Por eso es necesario seguir estos pasos desde el CMD:

- Establecer MOD_WSGI_APACHE_ROOTDIR como variable del environment

```
set "MOD_WSGI_APACHE_ROOTDIR=C:\Apache24"
```

- Activar el entorno virtual de Python

```
"D:\path\to\project-root\venv\Scripts\activate"
```

- Instale el módulo mod_wsgi en el entorno virtual

```
pip install mod_wsgi
```

- Copie el output para obtener la configuración requerida por Apache

```
mod_wsgi-express module-config
```

- Instale todas las dependencias de la librería del proyecto Django

```
pip install module-name
```

U obtenga un archivo de requisitos con `pip freeze > requirements.txt` y luego instalelo con

```
pip install -r requirements.txt
```

- Desactive el entorno virtual.

```
deactivate
```

Configuración

- Los entornos virtuales de Python no incluyen archivos dll y al menos uno de estos archivos es una dependencia para la configuración de Apache. Es posible copiar manualmente el archivo dll requerido en la carpeta `project-root\venv\Scripts` desde la carpeta de instalación de Python que generalmente se encuentra en `%username%\AppData\Local\Programs\Python\Python38`. El archivo principal a copiar es `python39.dll` y opcionalmente todos los otros dll que están junto a él y todo el contenido de la carpeta `Programs\Python\Python38\DLLs`.
- Edite el archivo host en `C:\Windows\System32\drivers\etc` agregando esta línea al final

```
127.0.0.2    www.app-name.com    app-name.com
```

- Actualice la lista de hosts permitidos para la aplicación en **settings.py**.

```
ALLOWED_HOSTS = ['www.app-name.com', 'app-name.com']
```

- Actualice el archivo **wsgi.py** del proyecto Django de la siguiente manera.

```
import os
import sys
from django.core.wsgi import get_wsgi_application
from pathlib import Path

# Add project directory to the sys.path
path_home = str(Path(__file__).parents[1])
if path_home not in sys.path:
    sys.path.append(path_home)

os.environ['DJANGO_SETTINGS_MODULE'] = 'main.settings'

application = get_wsgi_application()
```

- Soluciona un bug de Python que causa un error 500 cada vez que se realiza una consulta a la base de datos.

Edite el archivo **__init__.py** en

project-root\venv\Lib\site-packages\asgiref agregando lo siguiente.

```
# PATCH that fix a Python Bug:
import sys
import asyncio

if sys.platform == "win32" and sys.version_info >= (3, 8, 0):
    asyncio.set_event_loop_policy(asyncio.WindowsSelectorEventLoopPolicy())
```

- Configure Apache modificando el archivo httpd.conf en **C:\Apache24\conf**, agregando al final todo lo que sigue, pero reemplazando los directorios con los de su sistema.

```

LoadFile "S:/path/to/project-root/venv/Scripts/python38.dll"
LoadModule wsgi_module "S:/path/to/project-root/venv/lib/site-packages/mod_wsgi/server
WSGIPythonHome "C:/Users/User-name/AppData/Local/Programs/Python/Python38"
WSGIPythonPath "S:/path/to/project-root/venv/Lib/site-packages"

<VirtualHost *:80>
ServerAlias www.app-name.com
ServerName app-name.com
ServerAdmin info@admin.com
WSGIScriptAlias / "S:/path/to/project-root/project-name/wsgi.py"
    <Directory "S:/path/to/project-root/project-name">
        <Files wsgi.py>
            Require all granted
        </Files>
    </Directory>

Alias /static/ "S:/path/to/project-root/static/"
    <Directory "S:/path/to/project-root/static">
        Require all granted
    </Directory>

ErrorLog "S:/path/to/project-root/logs/apache.error.log"
CustomLog "S:/path/to/project-root/logs/apache.custom.log" common
</VirtualHost>

```

- Verifique la sintaxis de los archivos de configuración de apache.

```
"C:\Apache24\bin\httpd.exe" -t
```

- Ejecute el servidor Apache.

```
"C:\Apache24\bin\httpd.exe" -k start
```

- Verifique la aplicación Django desde el navegador visitando app-name.com.
- Si falta algún módulo, se recomienda usar este comando para asegurarse de que esté instalado en la Python Home configurada en Apache.

```
bash
pip install --target="C:/Users/User-name/AppData/Local/Programs/Python/Python38" libra
```

3 - C - Poner en Marcha una base de datos

Django admite varios servidores de bases de datos de forma oficial, incluyendo **PostgreSQL**, **MySQL**, **Oracle** y **SQLite**. Si planeas utilizar la funcionalidad de la API de base de datos de Django, debes asegurarte de que se esté ejecutando un servidor de base de datos compatible.

Comencemos con el gestor de MySQL.

- Lo primero que haremos será instalar la biblioteca ***mysqlclient***.

```
pip install mysqlclient
```

- Ahora procedemos a crear nuestra **base de datos**.

```
mysql -u root -p
```

```
CREATE DATABASE hackathon;
```

```
USE hackathon;
```

- Una vez con la base de datos creada, procedemos a modificar nuestro archivo **settings.py**. Indicamos que haremos uso del gestor MySQL sobre sqlite.

```
DATABASES = {  
    'default': {  
        'ENGINE': 'django.db.backends.mysql',  
        'NAME': 'hackathon',  
        'USER': 'root',  
        'PASSWORD': '',  
        'HOST': 'localhost',  
        'PORT': '',  
    }  
}
```

En mi caso mi **base de datos** tiene por nombre **hackathon**, el **user** de mi base de datos es **root** y **no posee una contraseña**.

Ejecutamos las migraciones.

```
python manage migrate
```

En nuestro servidor de MySQL listamos todas las tablas:

```
mysql> show tables;
+-----+
| Tables_in_hackathon |
+-----+
| auth_group           |
| auth_group_permissions |
| auth_permission      |
| auth_user            |
| auth_user_groups     |
| auth_user_user_permissions |
| django_admin_log     |
| django_content_type  |
| django_migrations    |
| django_session       |
+-----+
```

Perfecto, ahora estaremos haciendo uso de MySQL en nuestro proyecto.

Ahora continuamos con PostgreSQL.

Lo primero que debemos hacer será instalar la **biblioteca *psycopg2***.

```
pip install psycopg2-binary
```

Una vez la biblioteca haya sido instalada procedemos a crear nuestra **base de datos en PostgreSQL**.

Ejecutamos los siguientes comandos en la terminal.

```
psql postgres
```

```
CREATE DATABASE hackathon;
```

```
\connect hackathon;
```

Listo, ya tenemos nuestra base de datos.

El siguiente paso será **configurar nuestra aplicación**. Para ello modificamos el archivo settings.py.

```
DATABASES = {  
    'default': {  
        'ENGINE': 'django.db.backends.postgresql_psycopg2',  
        'NAME': 'hackathon',  
        'USER': 'eduardo',  
        'PASSWORD': '',  
        'HOST': 'localhost',  
        'PORT': '',  
    }  
}
```

En mi caso mi **base de datos** tiene por nombre *hackathon*, el **user** de mi base de datos es *eduardo* y **no posee una contraseña**.

Perfecto, una vez con la configuración hecha ya seremos capaces de utilizar nuestra base de datos con **PostgreSQL**.

```
python manage.py migrate
```

En nuestro servidor de PostgreSQL ejecutamos **dt** y deberíamos visualizar todas nuestras tablas.

```
hackathon=# \dt
```

```
public | auth_group | table | eduardo
```

public		auth_group_permissions		table		eduardo
public		auth_permission		table		eduardo
public		auth_user		table		eduardo
public		auth_user_groups		table		eduardo
public		auth_user_user_permissions		table		eduardo
public		django_admin_log		table		eduardo
public		django_content_type		table		eduardo
public		django_migrations		table		eduardo
public		django_session		table		eduardo

3 - D - Instalación de Django:

1. Ejecutamos el siguiente comando
 - pip install django


```
Símbolo del sistema
Microsoft Windows [Versión 10.0.19045.2965]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\alsin>pip install django
Defaulting to user installation because normal site-packages is not writeable
Collecting django
  Downloading Django-4.2.1-py3-none-any.whl (8.0 MB)
----- 8.0/8.0 MB 10.4 MB/s eta 0:00:00
Collecting asgiref<4,>=3.6.0
  Downloading asgiref-3.6.0-py3-none-any.whl (23 kB)
Collecting tzdata
  Downloading tzdata-2023.3-py2.py3-none-any.whl (341 kB)
----- 341.8/341.8 KB 10.7 MB/s eta 0:00:00
Collecting sqlparse<=0.3.1
  Downloading sqlparse-0.4.4-py3-none-any.whl (41 kB)
----- 41.2/41.2 KB ? eta 0:00:00
Installing collected packages: tzdata, sqlparse, asgiref, django
  WARNING: The script sqlformat.exe is installed in 'C:\Users\alsin\AppData\Roaming\Python\Python310\Scripts' which is not on PATH.
  Consider adding this directory to PATH or, if you prefer to suppress this warning, use --no-warn-script-location.
  WARNING: The script django-admin.exe is installed in 'C:\Users\alsin\AppData\Roaming\Python\Python310\Scripts' which is not on PATH.
  Consider adding this directory to PATH or, if you prefer to suppress this warning, use --no-warn-script-location.
Successfully installed asgiref-3.6.0 django-4.2.1 sqlparse-0.4.4 tzdata-2023.3
WARNING: You are using pip version 22.0.4; however, version 23.1.2 is available.
You should consider upgrading via the 'C:\Program Files\Python310\python.exe -m pip install --upgrade pip' command.

C:\Users\alsin>_
```

```
File Edit Selection View Go Run Terminal Help
manage.py - c11-17-ft-django - Visual Studio Code

EXPLORER
... manage.py X
OPEN EDITORS
X manage.py
C11-17-FT-DJANGO
  petpal
    > __pycache__
    > __init__.py
    > asgi.py
    > settings.py
    > urls.py
    > wsgi.py
  > venv
  > Lib
  > Scripts
  > .gitignore
  > pyenv.cfg
  > .gitignore
  > db.sqlite3
  > manage.py

manage.py
1 |!usr/bin/env python
2 |"""Django's command-line utility for administrative tasks."""
3 |import os
4 |import sys
5 |
6 |
7 |def main():
8 |    """Run administrative tasks."""
9 |    os.environ.setdefault("DJANGO_SETTINGS_MODULE", "petpal.settings")
10 |    try:
11 |        from django.core.management import execute_from_command_line
12 |    except ImportError as exc:
13 |        raise ImportError(
14 |            "Couldn't import Django. Are you sure it's installed and "
15 |            "available on your PYTHONPATH environment variable? Did you "
16 |            "forget to activate a virtual environment?"
17 |        ) from exc
18 |    execute_from_command_line(sys.argv)
19 |
20 |
21 |if __name__ == "__main__":
22 |    main()
```

Ejemplo:

Consistirá de dos partes:

- Un **sitio público** que le permite a las personas ver sondeos y votar en ellos.
- Un **sitio admin** que le permite añadir, modificar y borrar sondeos.

Si esta es la primera vez que utilizamos Django, tendremos que hacernos cargo de ciertas configuraciones iniciales. Concretamente, tendremos que autogenerar un código que establezca un Django **project** – un conjunto de ajustes para una instancia de Django, incluida la configuración de la base de datos, opciones específicas de Django y configuraciones específicas de la aplicación.

proyecto

Un paquete Python – e.g: una carpeta de código – que contiene todos los ajustes para una instancia de Django. Esto podría incluir la configuración de la base de datos, opciones específicas de Django y ajustes específicos de las aplicaciones.

Desde la línea de comandos, cambiamos a un directorio donde nos gustaría almacenar nuestro código, luego, ejecutamos el siguiente comando:

```
$ django-admin startproject mysite
```

Esto creará un directorio **mysite** en nuestro directorio actual.

El comando **startproject** creó:

```
mysite/  
  manage.py  
  mysite/  
    __init__.py  
    settings.py  
    urls.py  
    wsgi.py
```

Estos archivos son:

- El directorio raíz externo **mysite/** solo es un contenedor de su proyecto.
- **manage.py**: Una utilidad de la línea de comandos que nos permite interactuar con este proyecto Django de diferentes formas.

- En el interior del directorio **mysite/** es el propio paquete de Python para nuestro proyecto. Su nombre es el nombre del paquete de Python que tendremos que utilizar para importar todo dentro de este (por ejemplo, **mysite.urls**).
- **mysite/__init__.py**: Un archivo vacío que le indica a Python que este directorio debería ser considerado como un paquete Python.
- **mysite/settings.py**: Ajustes/configuración para este proyecto Django. [Django settings](#) le indicará todo sobre cómo funciona la configuración.
- **mysite/urls.py**: Las declaraciones URL para este proyecto Django; una «tabla de contenidos» de su sitio basado en Django.
- **mysite/wsgi.py**: Un punto de entrada para que los servidores web compatibles con WSGI puedan servir su proyecto.

Comprobamos que el proyecto Django funciona. Cambiamos el directorio externo **mysite**, si todavía no lo ha hecho, y ejecute los siguientes comandos:

```
$ python manage.py runserver
```

Veremos la siguiente línea de comandos:

```
Performing system checks...

System check identified no issues (0 silenced).

You have unapplied migrations; your app may not work properly until
they are applied.
Run 'python manage.py migrate' to apply them.


diciembre 02, 2019 - 15:50:53
Django version 2.2, using settings 'mysite.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
```

Y por último vemos que se instaló todo correctamente en nuestra ip
<http://127.0.0.1:8000/>

