jueves, 24 de noviembre de 2022

Clase 1: Preparar ambiente

Objetivo: Crear un proyecto de Django.

Para que podamos crear un proyecto Django necesitaremos preparar nuestro ambiente con software y algunos archivos.

Ustedes pueden instalar los requerimientos en su propio sistema operativo, pero para utilidad y sentido práctico aquí utilizaremos Docker para poder encapsular todo nuestro ambiente.

Software a instalar

- Docker. Sitio web para descargar Docker: https://www.docker.com/
- Git. Sitio web: https://git-scm.com/

Recuerden que cada vez que vayan a utilizar Docker, necesitan iniciarlo.

Clonar y hacer correr el proyecto localmente

Una vez instalado el software en su sistema podrán clonar el repositorio del taller donde estaremos compartiendo el código de nuestro proyecto.

Repositorio: https://github.com/alexsazo/tallerdjango2022

1. Abrir una consola / terminal / powershell en la carpeta donde quieran dejar el proyecto.
2. Clonar el repositorio. En la página del repo, van a encontrar un botón verde donde les aparecerá una URL de donde hacer la clonación del repositorio. En una carpeta van a ejecutar el siguiente comando:

```
git clone <direccion-url-que-les-muestre-el-repo>
```

3. Ir a la carpeta que se creó con el repositorio.
4. Guarden el archivo <.env> que les envié por correo y guárdenlo dentro de la carpeta que se creó del repositorio.
5. Ahora necesitamos crear el proyecto Django. Esto significa que Django va a crear una carpeta con los archivos esenciales para empezar a trabajar. Vamos a utilizar como nombre de proyecto “taller”.

```
docker compose run web django-admin startproject <nombre-del-proyecto>
```

Recordemos que los archivos Dockerfile y docker-compose.yml que están en el repositorio definen lo que nuestra máquina virtual base tendrá instalado y corriendo. Este comando hace lo siguiente: con <docker compose run web> hace que, utilizando la definición del servicio <web> (que se encuentra en docker-compose.yml local), corra los comandos que le siguen: <django-admin startproject…>.

6. Sólo por comodidad, vamos a hacer un paso más para ubicar los nuevos archivos en la raíz de nuestro repositorio. Debería haberse creado una carpeta llamada <taller> y dentro de ella un archivo <manage.py> y otra carpeta llamada <taller>. Muevan esa carpeta y archivo a la raíz del repositorio (al mismo lugar donde tenían el archivo docker-compose.yml). Es posible que necesiten reemplazar el nombre de la carpeta padre <taller> para que al copiar la carpeta interna que tiene el mismo nombre no hagan conflicto. Debería quedarles esta estructura en el proyecto:

```
/
      - docker-compose.yml
      - Dockerfile
      - LICENSE
      - manage.py
      - README.md
      - requirements.txt
      - start.sh
      - taller / …
```

Configuración inicial

- Abrir archivo para editarlo /taller/settings.py

- Reemplazar la definición de los ALLOWED_HOSTS y DATABASES para usar nuestras variables de entorno (que se encuentran guardadas en el archivo <.env>) como sigue:

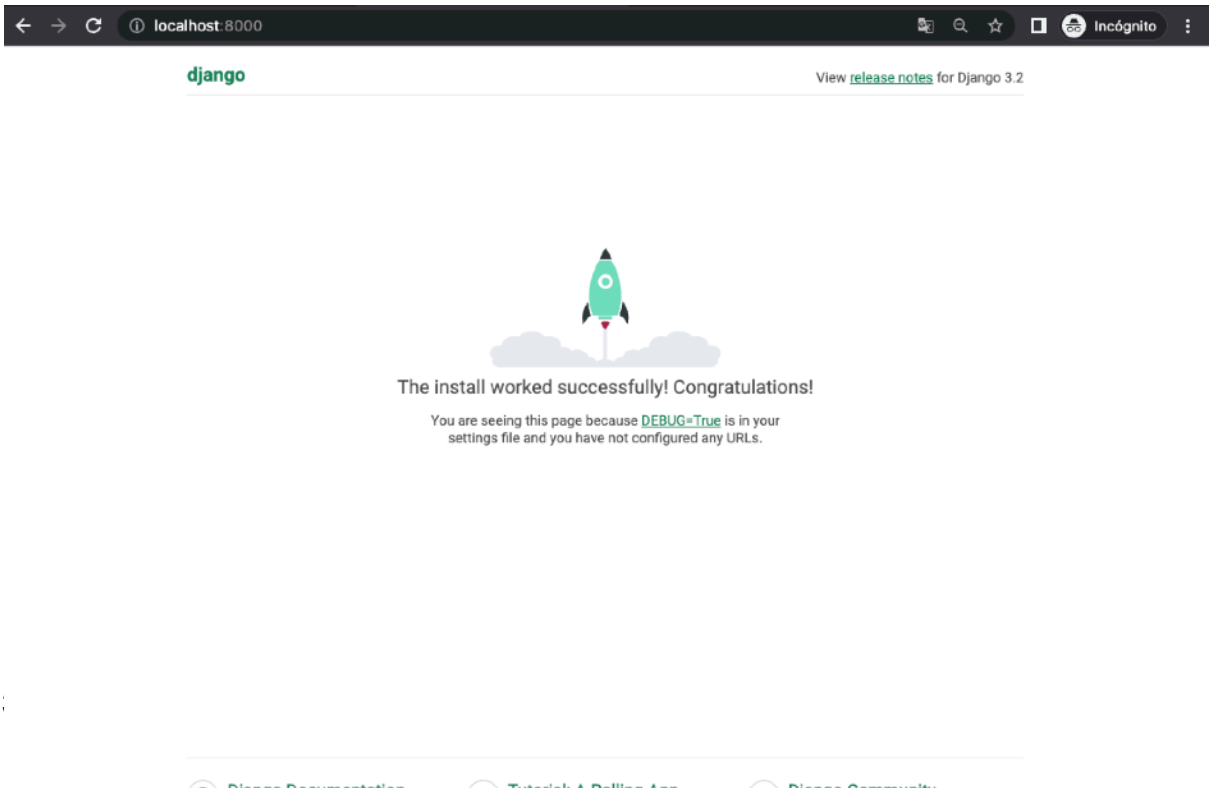
```
ALLOWED_HOSTS = ['*']

DATABASES = {
    'default': {
        'ENGINE': os.environ[‘DB_ENGINE’],
        'NAME': os.environ['DB_NAME'],
        'USER': os.environ['DB_USER'],
        'PASSWORD': os.environ['DB_PASSWORD'],
        'HOST': os.environ['DB_HOST'],
        'PORT': os.environ['DB_PORT'],
    }
}
```

Correr el servidor de pruebas

Para finalizar sólo necesitaremos correr el servidor de pruebas para comprobar que todo esté funcionando sin problemas. Ejecutar siguiente línea (siempre desde la carpeta del proyecto).

```
docker compose up -d
```

Después podremos acceder a: **http://localhost:8000 donde veremos la siguiente página:**



Si llegaron a este punto, todo está funcionando como corresponde.