EXERCISES QUE TRABAJAREMOS EN EL CUE

- EXERCISE 1: VERIFICAR Y EJECUTAR LA APLICACIÓN EN DJANGO.
- EXERCISE 2: REVISANDO EL PROYECTO WEB DE DJANGO.
- EXERCISE 3: CONFIGURANDO UN PROYECTO.

EXERCISE 1: VERIFICAR Y EJECUTAR LA APLICACIÓN EN DJANGO.

El objetivo del presente ejercicio es plantear una guía paso a paso para observar los distintos archivos de configuración, y adecuar el directorio de plantillas o template.

Requisitos previos:

- Tener conocimiento de una terminal o línea de comando e instalación de paquetes de software en el sistema operativo, en este caso en Linux Ubuntu.
- Tener previamente instalado la versión de Python 3.
- Hacer uso de la herramienta Visual Studio Code.

VERIFICAR Y EJECUTAR LA APLICACIÓN EN DJANGO

Previamente creamos el entorno virtual `project_django`, y dentro el proyecto site_web_django y una aplicación boards. Para abrir el proyecto nos ubicamos en el directorio site_web_django, y lo hacemos con VSC:

```
$cd site_web_django
$code .
```

Activamos el proyecto con `workon`:

```
$workon projects_django
```

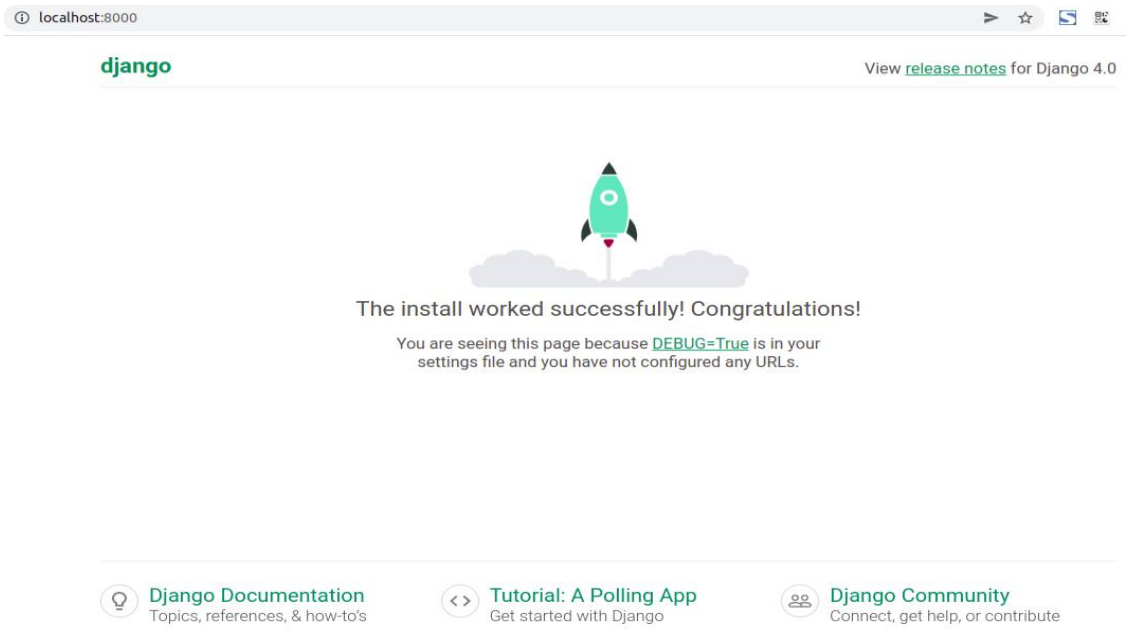
Ejecutamos la aplicación:

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

Observamos en la terminal que se está ejecutando el servidor en http://127.0.0.1:8000/.

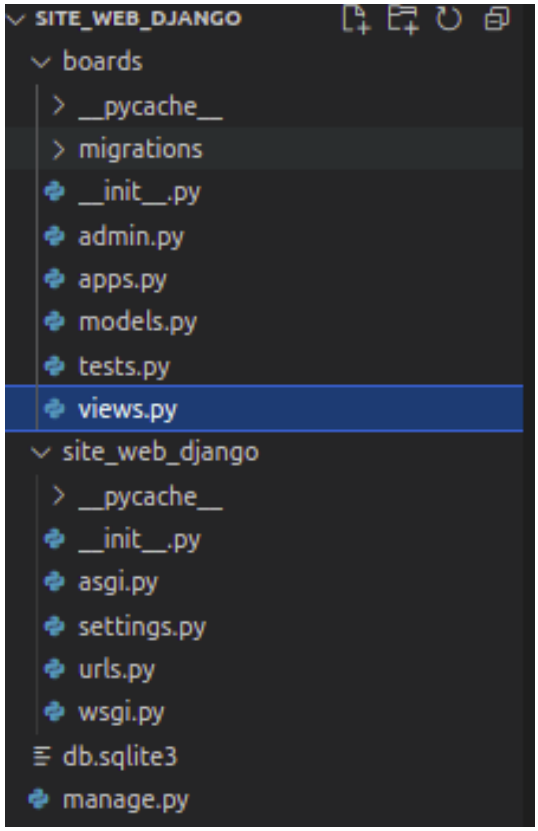
```
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
July 10, 2022 - 20:35:21
Django version 4.0.5, using settings 'site_web_django.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
```

Observamos:



EXERCISE 2: REVISANDO EL PROYECTO WEB DE DJANGO.

Observamos la estructura del proyecto:



EXERCISE 3: CONFIGURANDO UN PROYECTO.

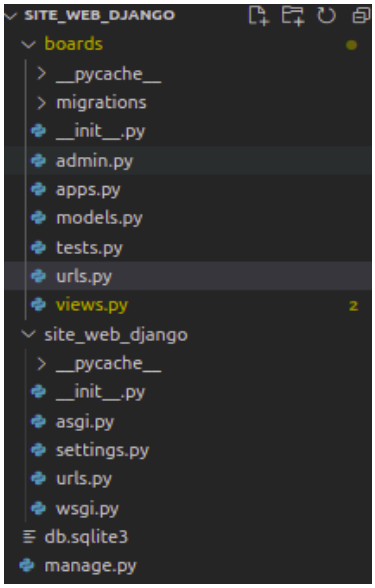
Procedemos a crear una vista y configurar el proyecto. Se procede a crear la primera vista. Inicializamos el archivo `boards/views.py`, y se coloca el siguiente código de Python:

BOARDS/VIEWS.PY

```
1 from django.shortcuts import render
2
3 # Create your views here.
4 from django.http import HttpResponse
5
6
7 def index(request):
8     return HttpResponse("Mensaje, Hola Mundo.")
```

Esta es la vista más simple posible en Django. Para llamarla, necesitamos mapearla a un URL y para esto se requiere de un `URLconf`.

Para crear una `URLconf` en el directorio de boards, se genera un archivo llamado `urls.py`. Su directorio boards de aplicación ahora debería verse así:



CONFIGURACIÓN URLS.PY

Seguidamente, en el archivo recientemente creado boards/urls.py, agregamos el siguiente código: `boards/urls.py`.

```
from django.urls import path

from . import views

urlpatterns = [
    path('', views.index, name='index'),
]
```

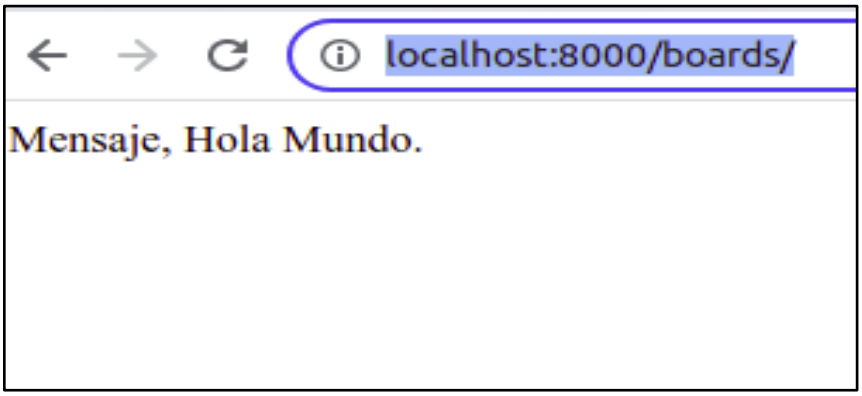
El siguiente paso es apuntar la site_web_django.urls raíz al módulo `boards.urls`. En site_web_django/urls.py, agregue una importación para django.urls.include e inserte un `include()` en la lista `urlpatterns`, para que así obtenga:

SITE_WEB_DJANGO/URLS.PY

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

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

- La función `include()` permite hacer referencia a otros `URLconfs`. Cada vez que Django encuentra `include()`, corta cualquier parte de la URL que coincida hasta ese punto, y envía la cadena restante a la URLconf incluida para su posterior procesamiento.
- La idea detrás de `include()` es facilitar la conexión y reproducción de URL. Dado que las boards están en su propia URLconf (`boards/urls.py`).
- Siempre debe usar `include()` cuando incluya otros patrones de URL. `admin.site.urls` es la única excepción a esto.

Procedemos a visualizar en el navegador, y observamos: http://localhost:8000/boards/



CONFIGURANDO SETTINGS.PY

De forma predeterminada, `INSTALLED_APPS` contiene las siguientes aplicaciones, las cuales vienen con Django:

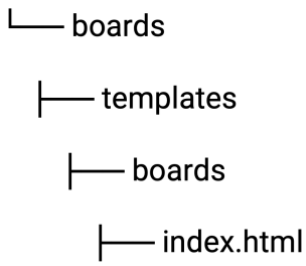
- `django.contrib.admin`: el sitio de administración.
- `django.contrib.auth`: un sistema de autenticación.
- `django.contrib.contenttypes`: un marco para tipos de contenido.
- `django.contrib.sessions`: un marco de sesión.
- `django.contrib.messages`: un marco de mensajería.
- `django.contrib.staticfiles`: un marco para administrar archivos estáticos.
- `boards.apps.BoardsConfig`: aplicación creada para la práctica.

CONFIGURACIÓN DE TEMPLATES

Cada marco web necesita una forma conveniente de generar archivos HTML, y en Django el enfoque es usar plantillas: archivos HTML individuales que se pueden vincular entre sí, y que también incluyen lógica básica.

CONFIGURACIÓN DE PATHS

La primera consideración es dónde ubicar las plantillas dentro de la estructura de un proyecto Django. Esto requiere configurar los paths y para hacer eso tenemos dos opciones. De forma predeterminada, el cargador de plantillas de Django buscará dentro de cada aplicación las plantillas relacionadas. Sin embargo, la estructura es algo confusa: cada aplicación necesita un nuevo directorio de plantillas, otro directorio con el mismo nombre que la aplicación, y luego, el archivo de plantilla.



Esto significa que necesitaríamos crear un nuevo directorio de plantillas, un nuevo directorio con el nombre de la aplicación, las páginas y, finalmente, nuestra propia plantilla, que es `index.html`.

Sin embargo, existe otro enfoque que consiste en crear un único directorio de plantillas a nivel de proyecto, y colocar todas las plantillas allí. Al hacer un pequeño ajuste en nuestro archivo `site_web_django/settings.py`, podemos indicar a Django que también busque plantillas en este directorio. Basado en este principio, configuraremos el directorio de las plantillas.

Para ello, salimos del servidor de desarrollo de Django, presionando Control + C, y seguidamente creamos el directorio templates.

```
$ mkdir templates
```

Ajustamos en el `settings.py` indicando la ubicación de nuestro nuevo directorio de plantillas, y ajustando la siguiente línea en la variable de entorno `TEMPLATES`.

SITE_WEB_DJANGO/SETTINGS.PY

```
TEMPLATES = [
    {
.....
        'DIRS': [BASE_DIR / 'templates'],
.....

]
```

Dentro del directorio templates, se crea un nuevo archivo llamado `index.html` con Visual Studio Code.

TEMPLATES/INDEX.HTML

```
<!-- templates/index.html -->
<h1>Saludos, Hola Mundo</h1>
```

Seguidamente, procedemos a configurar los archivos de URLs y Views, respectivamente:

En nuestra vista, usaremos el `TemplateView` incorporado para mostrar nuestra plantilla. Actualice el archivo `boards/views.py`

BOARDS/VIEWS.PY

```python
# boards/views.py

from django.views.generic import TemplateView


class IndexPageView(TemplateView):
    template_name = "index.html"
```

Luego, procedemos a actualizar las URL. Necesitamos hacer actualizaciones en dos lugares. Primero, actualizamos el archivo `site_web_django/urls.py` para que apunte a nuestra aplicación de boards; y luego, dentro de boards, hacemos coincidir las vistas con las rutas de URL.

Comencemos con el archivo:

SITE_WEB_DJANGO/URLS.PY

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


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

Agregamos include en la segunda línea para apuntar la URL existente a la aplicación de páginas. Posteriormente, editamos el archivo `boards/urls.py`, y se agrega cuando se utilizan vistas basadas en clases, siempre colocando `as_view()` al final del nombre de la vista.

BOARDS/URLS.PY

```python
# boards/urls.py

from django.urls import path
from .views import IndexPageView

urlpatterns = [
    path("", IndexPageView.as_view(), name="index"),
]
```

Finalmente, iniciamos el servidor web local, y visualizamos en el navegador la salida correspondiente.

Terminal:

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