

EXERCISES QUE TRABAJAREMOS EN EL CUE

- EXERCISE 1: PRUEBAS DE DJANGO
- EXERCISE 2: CREACIÓN DE PÁGINAS ESTÁTICAS
- EXERCISE 3: CASO DE PRUEBA SIMPLE EN DJANGO
- EXERCISE 4: PRUEBAS A LA APLICACIÓN CRUDAPP AL MODELO EMPLEADO

EXERCISE 1: PRUEBAS DE DJANGO

La prueba es una parte importante, pero a menudo descuidada, de cualquier proyecto de Django. En términos generales, hay dos tipos de pruebas que se deben ejecutar:

- Las pruebas unitarias son pequeñas, aisladas, y se enfocan en una función específica.
- Las pruebas de integración tienen como objetivo imitar el comportamiento del usuario, y combinar múltiples piezas de código y funcionalidad.

Si bien podríamos usar una prueba unitaria para confirmar que la página de inicio devuelve un código de estado HTTP de 200, una de integración podría imitar todo el flujo de registro de un usuario.

Django viene con un pequeño conjunto de sus propias herramientas para escribir pruebas, en particular un cliente de prueba, y cuatro clases de casos de prueba proporcionadas, las cuales se basan en el módulo `unittest` de Python, `TestCase` de Python, y la clase base.

El cliente de prueba de Django se puede usar para actuar como un navegador web ficticio, y verificar las vistas. Esto se puede hacer dentro del shell de Django para pruebas únicas, pero se integra más comúnmente en las pruebas unitarias.

La mayoría de las pruebas unitarias de Django se basan en `TestCase`, pero en ocasiones cuando una aplicación no se basa en una base de datos, se puede usar `SimpleTestCase`.

En esta práctica crearemos dos pruebas básicas, una enfocada a la aplicación con dos páginas estáticas y otra a la vista de la aplicación crudapp que contiene el modelo de `Empleado`. Vamos a utilizar la aplicación crudapp del proyecto `django-orm-postgresql`.

EXERCISE 2: CREACIÓN DE PÁGINAS ESTÁTICAS

Actualizaremos las vistas de la aplicación crudapp, agregando dos clases `InicioPageView` y `AcercaPageView`, que se basarán en las plantillas correspondientes a `inicio.html` y `acerca-de.html`.

Agregue la librería en el archivo `crudapp/views.py`:

```
1 #crudapp/views.py
2 from django.views.generic import TemplateView
```

Adecue al final del archivo `crudapp/views.py` con el siguiente código:

```
1 #crudapp/views.py
2 class InicioPageView(TemplateView):
3     template_name = "inicio.html"
4
5 class AcercaPageView(TemplateView):
6     template_name = "acerca-de.html"
```

Adecue el archivo `crudapp/urls.py` para asignar las rutas correspondientes:

```
1 #crudapp/urls.py
2 from django.urls import path
3 from . import views
4 from .views import InicioPageView, AcercaPageView
5
6 urlpatterns = [
7     path('', views.insertar_emp, name='insertar-emp'),
8     path('mostrar/', views.mostrar_emp, name='mostrar-emp'),
9     path('editar/<int:pk>', views.editar_emp, name='editar-emp'),
10    path('editar/actualizaremppleado/<int:id>', views.actualizaremppleado,
11    name='actualizaremppleado'),
12    path('eliminar/<int:pk>', views.eliminar_emp, name='eliminar-emp'),
13    path("empleado/<int:pk>", views.empleado_detalle, name = "empleado-
14    detalle"),
15    path("inicio/", InicioPageView.as_view(), name="inicio"),
16    path("acerca-de/", AcercaPageView.as_view(), name="acerca-de"),
17 ]
```

Finalmente, creamos las plantillas `inicio.htm` y `acerca-de.html` que hacen referencia a la vista en `crudapp/views.py` en `crudapp/templates/`.

Cree el archivo `inicio.html` dentro de la carpeta `crudapp/templates`:

```
1 #crudapp/templates/inicio.html
2 <h1>Página de inicio</h1>
```

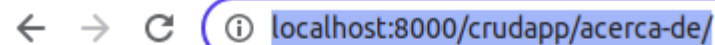
Cree el archivo `acerca-de.html` dentro de la carpeta `crudapp/templates`:

```
1 #crudapp/templates/acerca-de.html
2 <h1>Página Acerca de</h1>
```

Luego, navegue a la página de inicio en <http://localhost:8000/crudapp/inicio/>, y a la página en <http://localhost:8000/crudapp/acerca-de/> para confirmar que todo funciona.



Página de inicio



Página Acerca de

EXERCISE 3: CASO DE PRUEBA SIMPLE EN DJANGO

Como nuestra aplicación crudapp contiene dos páginas estáticas en este momento, podemos usar `SimpleTestCase` en el archivo existente `crudapp/tests.py`.

```
1 #crudapp/tests.py
2 from django.test import TestCase
3 from django.test import SimpleTestCase
4 from django.urls import reverse
5
6 from .models import Empleado
7
8 class InicioTests(SimpleTestCase):
9     def test_url_exists_at_correct_location(self):
```

```
10 response = self.client.get("/crudapp/inicio/")
11 self.assertEqual(response.status_code, 200)
12
13 def test_url_available_by_name(self):
14     response = self.client.get(reverse("inicio"))
15     self.assertEqual(response.status_code, 200)
16
17 def test_template_name_correct(self):
18     response = self.client.get(reverse("inicio"))
19     self.assertTemplateUsed(response, "inicio.html")
20
21 def test_template_content(self):
22     response = self.client.get(reverse("inicio"))
23     self.assertContains(response, "<h1>Página de inicio</h1>")
24     self.assertNotContains(response, "No es la Página")
```

Seguidamente, construimos cuatro pruebas para nuestra página de inicio:

- Primero, probamos que la URL funciona correctamente en `/crudapp/inicio/`, la página de inicio devuelve una respuesta HTTP 200.
- En segundo lugar, confirmamos llamar a la página por su nombre de URL a través de `reverse()`. Es por ello que agregamos `name="inicio"` a la ruta `crudapp/urls.py` de la URL para la página de inicio.
- Tercero, confirmamos que la plantilla utilizada es `inicio.html`.
- Finalmente, verificamos el contenido de la plantilla buscando el fragmento de HTML `<h1>Página de inicio</h1>`, y también que el HTML incorrecto no esté en la página. Siempre es bueno probar tanto el comportamiento esperado como el inesperado.

Ejecutamos la prueba en la terminal:


```
1 $ python manage.py test crudapp
2 Found 4 test(s).
3 System check identified no issues (0 silenced).
4 ....
5 -----
6 Ran 4 tests in 0.115s
7
8 OK
```

Existen otras pruebas que posiblemente podríamos agregar, pero para las páginas web estáticas éstas cubren los aspectos básicos: validar una respuesta HTTP 200, usar la vista y la plantilla correctas, y verificar el contenido de la plantilla.

EXERCISE 4: PRUEBAS A LA APLICACIÓN CRUDAPP AL MODELO EMPLEADO

Ya hemos creado la aplicación crudapp con un modelo de Empleado que se conecta a la base de datos. Al mismo tiempo, hemos realizado el proceso de CRUD al modelo que podemos observar tanto por los templates como por el sitio administrativo de Django.

Observamos en <http://localhost:8000/crudapp/mostrar/>:


localhost:8000/crudapp/mostrar/

Información de Empleados					
Id de Empleado	Nombre	Correo	Designación	Edit	Delete
452	Jose Camargo	jose@test.com	Desarrollador Web	Actualizar	Eliminar
123	Juan Perez	juan@test.com	Project Manager	Actualizar	Eliminar

¿Quieres ingresar más empleadas?

[<-- Ir a la pagina de Inicio](#)

Usted ha visitado esta página 2 veces.

Y en <http://localhost:8000/admin/crudapp/empleado/>:

Django administration
WELCOME, ADMIN VIEW SITE / CHANGE PASSWORD / LOG OUT

Home » Crudapp » Empleados

Start typing to filter...

AUTHENTICATION AND AUTHORIZATION

Groups [+ Add](#)

Users [+ Add](#)

CRUDAPP

Empleados [+ Add](#)

RELACIONES

Automotrizs [+ Add](#)

Modelo carros [+ Add](#)

Select empleado to change

Action: Go 0 of 2 selected

<input type="checkbox"/>	ID	EMP ID	EMP NOMBRE	EMP CORREO	EMP DESIGNACION
<input type="checkbox"/>	10	452	Jose Camargo	jose@test.com	Desarrollador Web
<input type="checkbox"/>	9	123	Juan Perez	juan@test.com	Project Manager

2 empleados

ADD EMPLEADO [+](#)

FILTER

↓ By emp nombre

All

Jose Camargo

Juan Perez

↓ By emp designation

All

Desarrollador Web

Project Manager

TestCase es la clase más común para escribir pruebas en Django. Nos permite simular consultas a la base de datos. **setUpTestData()** nos permite crear datos iniciales una vez, a nivel de clase, para todo el **TestCase**. Esta técnica permite pruebas mucho más rápidas que la creación de datos desde cero para cada prueba de unidad individual dentro de la clase.

Para nuestro modelo de **Empleado** creamos las siguientes pruebas en el archivo **crudapp/tests.py**:

```
1 #crudapp/tests.py
2 from django.test import TestCase
3 from django.test import SimpleTestCase
4 from django.urls import reverse
5
6 from .models import Empleado
7
8
9 class EmpleadoTests(TestCase):
10     @classmethod
11     def setUpTestData(cls):
12         cls.empleado = Empleado.objects.create(emp_id="123",
13                                                emp_nombre='Juan',
14                                                emp_correo='juan@test.com',
15                                                emp_designacion='Desarrollo Web')
16
17     def test_model_content(self):
18         self.assertEqual(self.empleado.emp_id, "123")
19         self.assertEqual(self.empleado.emp_nombre, "Juan")
20         self.assertEqual(self.empleado.emp_correo, "juan@test.com")
21         self.assertEqual(self.empleado.emp_designacion, "Desarrollo
22 Web")
23
24     def test_url_exists_at_correct_location(self):
25         response = self.client.get("/crudapp/")
26         self.assertEqual(response.status_code, 200)
27
28     def test_homepage(self):
29         response = self.client.get(reverse("mostrar-emp"))
30         self.assertEqual(response.status_code, 200)
31         self.assertTemplateUsed(response, "mostrar.html")
32         self.assertContains(response, "Información de Empleados")
33
34
35
```

- La primera prueba unitaria **test_model_content** verifica que los datos en nuestra base de datos simulada coincidan con los que se crearon inicialmente en **setUpTestData**.
- Luego, verificamos la URL para confirmar que devuelve una respuesta HTTP 200.

- Finalmente, `test_homepage` utiliza `reverse` para llamar al nombre de la URL, busca una respuesta HTTP 200, verifica que se use la plantilla correcta, y confirma que el contenido HTML coincide con lo esperado.

Ejecutamos la prueba en la terminal:

```
1 $ python manage.py test crudapp
2 Found 3 test(s).
3 Creating test database for alias 'default'...
4 System check identified no issues (0 silenced).
5 ...
6 -----
7 Ran 3 tests in 0.112s
8
9 OK
10 Destroying test database for alias 'default'...
```

A medida que los proyectos de Django crecen en complejidad, es común eliminar el archivo `app/tests.py` inicial, y reemplazarlo con una carpeta de pruebas a nivel de aplicación que contiene archivos de pruebas individuales para cada área de funcionalidad. Por lo tanto, en lugar de un solo archivo `crudapp/tests.py`, se puede tener un directorio `crudapp/tests/`, y dentro de él los archivos, que por defecto deben comenzar por `test_` para diferentes áreas de la aplicación.

Por ejemplo:

