

SALECOLD

PRUEBAS UNITARIAS: PYTEST



Instalación de Pytest



Configuración Pytest



Pruebas

Instalación de Pytest

1. Activar el virtual enviroment de nuestro proyecto con el siguiente comando:

```
1 source env/bin/activate
```

2.Instalar el gestor de paquetes de python con el siguiente comando:

 SI ya esta instalado omitir este paso.

```
1 sudo apt install python3-pip
```

3.Instalar la libreria pytest en nuestro proyecto con el siguiente comando:

```
1 pip install pytest-django
```

```
daniel ~ (e) env ~ > Escritorio > Proyecto formativo django > SaleCold > pip install pytest-django
Defaulting to user installation because normal site-packages is not writeable
Collecting pytest-django
  Downloading pytest_django-4.5.1-py3-none-any.whl (20 kB)
Collecting pytest>=5.4.0
  Using cached pytest-6.2.5-py3-none-any.whl (280 kB)
Requirement already satisfied: packaging in /home/daniel/.local/lib/python3.8/site-packages (from pytest>=5.4.0->pytest-django) (21.0)
Requirement already satisfied: pluggy<2.0,>=0.12 in /home/daniel/.local/lib/python3.8/site-packages (from pytest>=5.4.0->pytest-django) (1.0.0)
Requirement already satisfied: toml in /home/daniel/.local/lib/python3.8/site-packages (from pytest>=5.4.0->pytest-django) (0.10.2)
Requirement already satisfied: iniconfig in /home/daniel/.local/lib/python3.8/site-packages (from pytest>=5.4.0->pytest-django) (1.1.1)
Requirement already satisfied: py>=1.8.2 in /home/daniel/.local/lib/python3.8/site-packages (from pytest>=5.4.0->pytest-django) (1.10.0)
Requirement already satisfied: attrs>=19.2.0 in /home/daniel/.local/lib/python3.8/site-packages (from pytest>=5.4.0->pytest-django) (21.2.0)
Requirement already satisfied: pyparsing>=2.0.2 in /home/daniel/.local/lib/python3.8/site-packages (from packaging->pytest>=5.4.0->pytest-django) (2.4.7)
Installing collected packages: pytest, pytest-django
Successfully installed pytest-6.2.5 pytest-django-4.5.1
```

Prueba 1
Prueba 2
Prueba 3

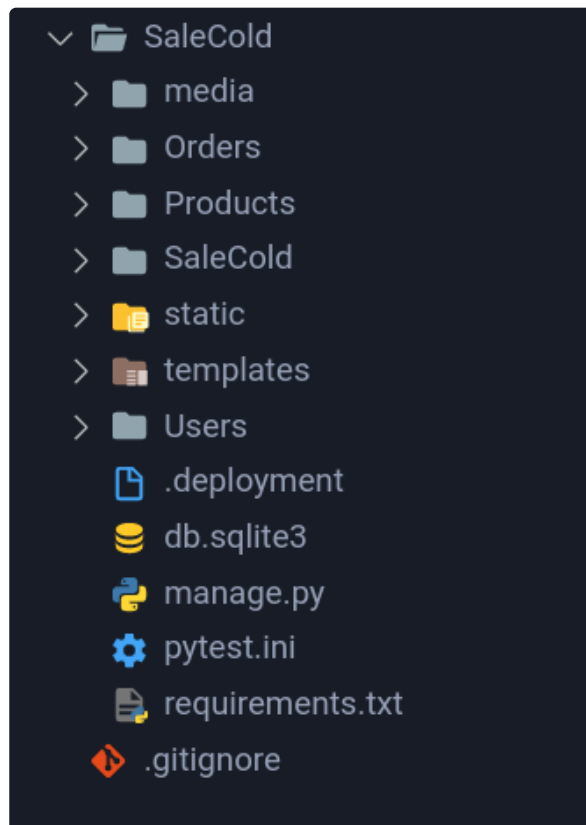
4. Ejecutar el siguiente comando para verificar que la instalación se realizó de manera correcta:

```
1 pip freeze
```

Configuración Pytest

1. Crear un archivo llamado `pytest.ini` en la carpeta principal del proyecto.

 En este archivo se almacenará la configuración de Pytest.



2. En el archivo `pytest.ini` especificar la configuración con los siguientes aspectos:

- **DJANGO_SETTINGS_MODULE** = En esta variable constante se especificará el archivo que contiene la configuración del proyecto.
- **python_files** = En esta variable se especificará el nombre de los archivos que pytest busque para ejecutar los tests.

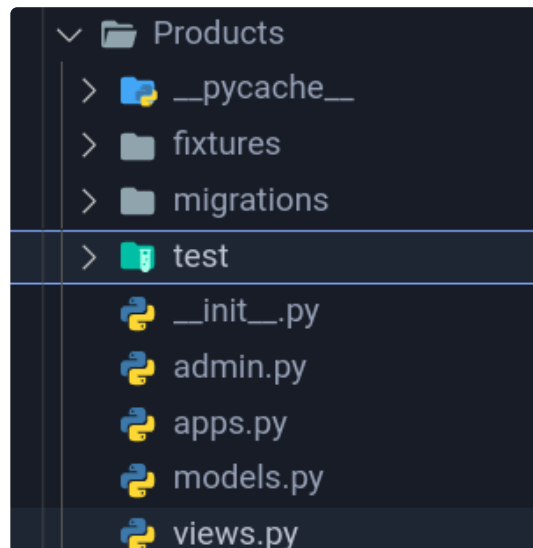
```
1 [pytest]
2
3 DJANGO_SETTINGS_MODULE = SaleCold.settings
4
```

```
5 python_files = test.py test_*.py *_test.py
```

3. Agregar el método create al modelo que le vamos a realizar las pruebas.

```
1 from django.db import models
2
3 class Category(models.Model):
4     id_category = models.AutoField("Id categoria", primary_key = True)
5     name = models.CharField("Nombre categoria", max_length = 55)
6     description = models.CharField("Descripcion", null = True, blank = True, max_length = 100)
7
8     def __str__(self):
9         return self.name
10
11     class Meta:
12         verbose_name = "Categoria"
13         verbose_name_plural = "Categorias"
14         db_table = "category"
15         ordering = ['id_category']
16
17     @classmethod
18     def create(cls, id_category, name, description):
19         category = cls(id_category = id_category, name = name, description = description)
20         return category
```

4. Dentro de la aplicación o módulo crear una carpeta llamada test, en esta carpeta se almacenaran todos los test relacionados a la aplicación.



Pruebas

Se implementaron tres pruebas en la creación de instancias de el modelo Category, cada prueba se represento como una función.

Prueba 1

```
1 import pytest
2 from Products.models import Category
3
4 @pytest.mark.django_db
5 def test_create_category():
6     category = Category.create(
7         id_category = 2,
8         name = "categoria 2",
9         description = ""
10    )
11    assert category.id_category == 2
```

Prueba 2

```
1 import pytest
2 from Products.models import Category
3
4 @pytest.mark.django_db
5 def test_create_category2():
6     category = Category.create(
7         id_category = 3,
8         name = "categoria 3",
9         description = ""
10    )
11    assert category.id_category == 3
```

Prueba 3

```
1 import pytest
2 from Products.models import Category
3
4 @pytest.mark.django_db
5 def test_create_category3():
6     category = Category.create(
7         id_category = 4,
8
9         name = "categoria 4",
10        description = ""
11    )
12    assert category.id_category == 4
```

Resultado

Para ejecutar las pruebas lo haremos con el siguiente comando:



- i) Para conocer mas información sobre que pruebas se ejecutaron, después de pytest colocamos el verbose -vv.

```
1 pytest -vv
```

```
daniel (e) env ~ > Escritorio > Proyecto formativo django > SaleCold > pytest -vv
===== test session starts =====
platform linux -- Python 3.8.10, pytest-6.2.5, py-1.10.0, pluggy-1.0.0 -- /usr/bin/python3
cachedir: .pytest_cache
django: settings: SaleCold.settings (from ini)
rootdir: /home/daniel/Escritorio/Proyecto formativo django/SaleCold, configfile: pytest.ini
plugins: django-4.5.1
collected 3 items

Products/test/test_category.py::test_create_category PASSED [ 33%]
Products/test/test_category.py::test_create_category2 PASSED [ 66%]
Products/test/test_category.py::test_create_category3 PASSED [100%]

===== 3 passed in 1.34s =====
```