Contenido

Archivo settings.py

Es un archivo de configuración.

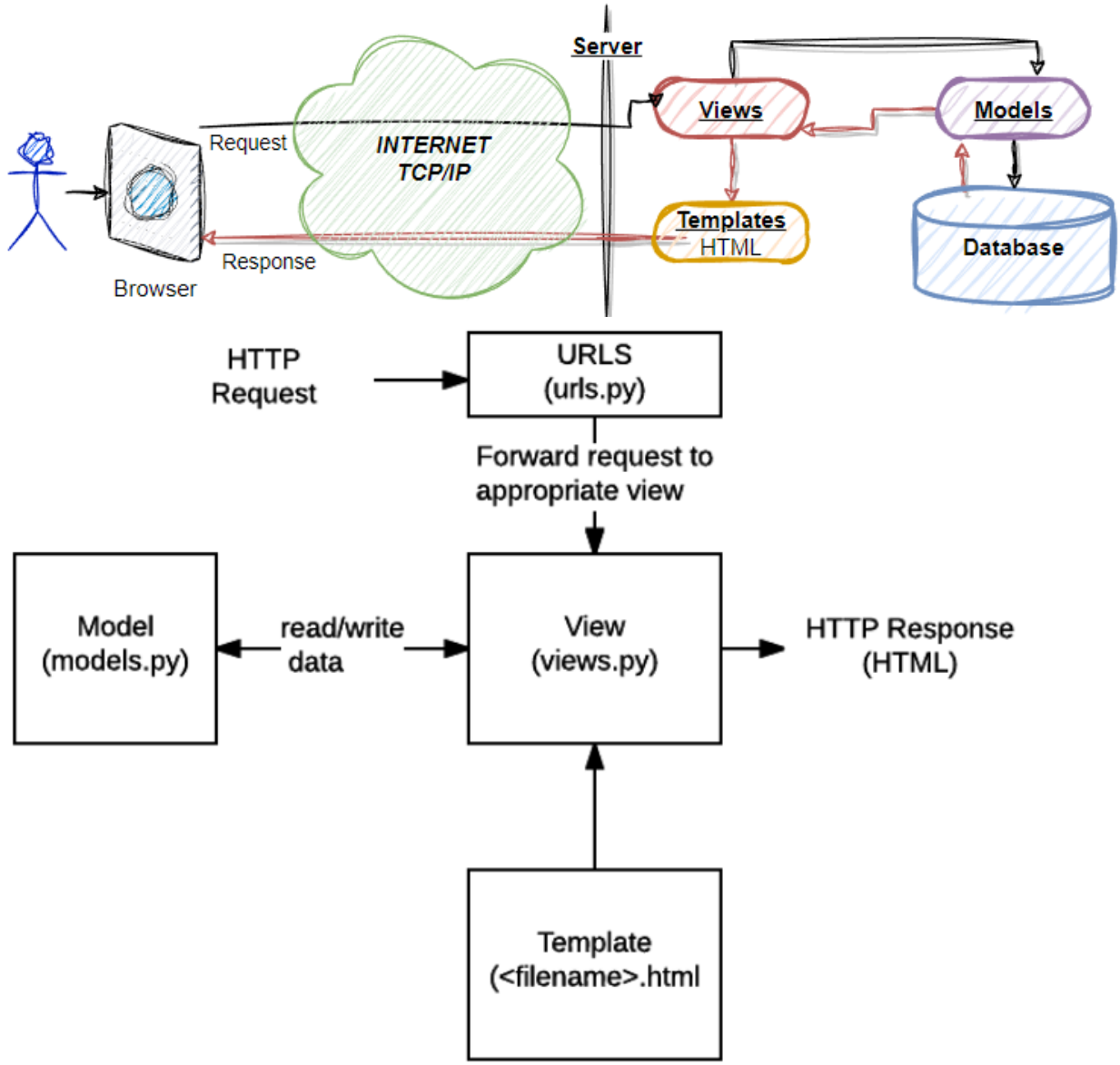
- DEBUG
  - ```
    25  # SECURITY WARNING: d
    26  DEBUG = True
    ```
- LANGUAGE_CODE
  - ```
    106  LANGUAGE_CODE = 'es'
    107
    108  TIME_ZONE = 'America/Mexico_City'
    ```
- DATABASES
  - ```
    76  DATABASES = {
    77      'default': {
    78          'ENGINE': 'django.db.backends.sqlite3',
    79          'NAME': BASE_DIR / 'db.sqlite3',
    80      }
    81  }
    ```
- INSTALLED_APPS
  - ```
    33  INSTALLED_APPS = [
    34      'django.contrib.admin',
    35      'django.contrib.auth',
    36      'django.contrib.contenttypes',
    37      'django.contrib.sessions',
    38      'django.contrib.messages',
    39      'django.contrib.staticfiles',
    40  ]
    ```
  - Sirven para el “Panel de administrador”, “autenticador” de usuarios, entre otras cosas.

FrameWork

¿Por qué Django?

- **<u>Ahorro de tiempo</u>**
- ***El desarrollador debe gastar tiempo enfocado a generar valor, mediante el entendiendo de la “lógica del negocio” y no con el desarrollo de CRUD’s para administración de catálogos, usuarios y privilegios o “inventando el hilo negro”***
- Framework para el desarrollo **Web** en Python
- Versátil / Fácil de usar
- Aplicaciones complejas
- Aplicaciones web **eficientes** y **seguras**
- *Django–rest Framework
  - *Microservicios*
  - *Cómputo en la nube*

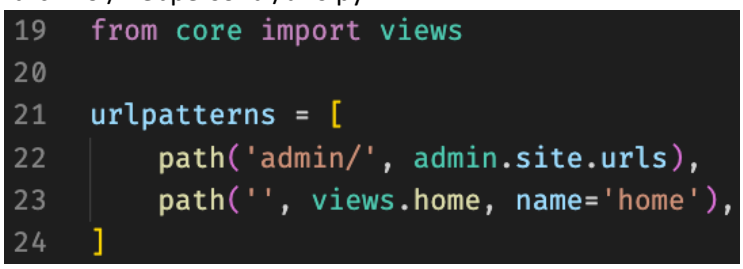
El framework es <Modelo – Vista – Template> y se puede explicar mediante los siguientes diagramas.



Ejemplo Core

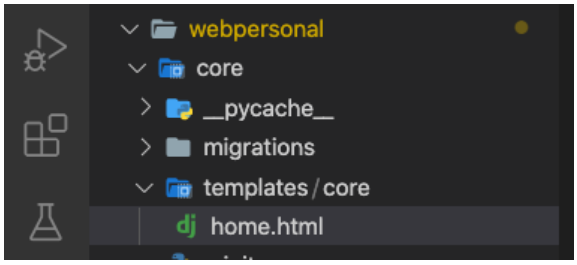
1. Agregar la aplicación “core” dentro del archivo settings.py en la sección correspondiente.
   a.
```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'core'
]
```
2. Editar el archivo core/views.py

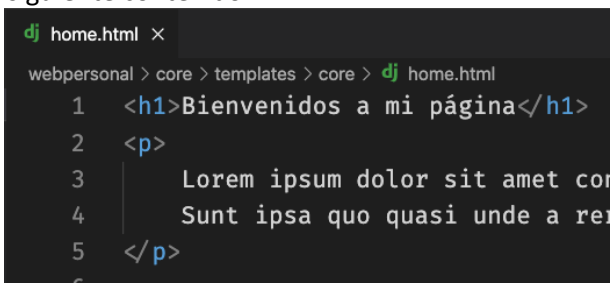
a.

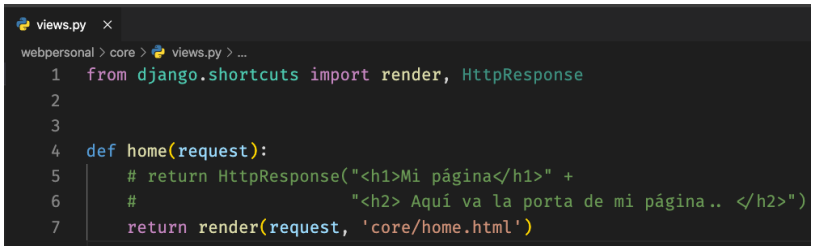

3. Editar el archivo /webpersonal/urls.py
a.


4. Correr el servidor

Templates

1. Crear la siguiente estructura de directorios dentro de “core”
    a. templates\core
2. Crear el archivo home.html
a.


3. Crear el siguiente contenido
a.


4. Dentro del método view
a.


Plantilla WebPersonal

1. Existen varios sitios que ofrecen “templates” gratis basados en Bootstrap, por ejemplo:
    a. https://startbootstrap.com/
2. Verificar el funcionamiento de los html’s estáticos de la carpeta webPersonal.
3. Identificar que parte del código “siempre” se repite en todas las páginas

Herencia en plantillas

Template tag block

1. Dentro de la carpeta templates\core, crear el archivo base.html
2. Copiar el código que se repite y crear un templete tag por cada sección que se requiera incluir en las demás páginas.
    a. {% block content %} {% endblock %}

```
                    <i class="far fa-file-chart-line"></i> Proyec
                </li>
                <li class="navbar-nav">
                <a class="nav-link" href="contact.html">
                    <i class="fal fa-at"></i> Contacto</a>
                </li>
            </ul>
            </div>
        </div>
        </nav>

        {% block content %} {%endblock%}

        <!-- Footer -->
        <footer id="main-footer" class="bg-dark">
            <div class="container">
                <div class="row">
                <div class="col text-center py-2">
                    <p>franvazgom Copyright &copy; <span id="year
```

    b. El “block” sirve para definir un bloque de contenido con un nombre
3. Ahora, dentro del archivo home.html hacer uso de la herencia
    a. {% extends ‘core/base.html’ %}
    b. Colocar el contenido de la página, dentro del bloque..

home.html M

webpersonal > core > templates > core > home.html

You, 1 second ago | 1 author (You)

```
{% extends "core/base.html" %}
{% block content %}

<!-- Home Section -->
<header id="home-section">
  <div class="dark-overlay">
    <div class="home-inner container">
```

Template tag url

1. En el archivo base.html, reemplazar las urls directas haciendo uso del template tag url {% url ‘nombre’ %}

```
<li class="nav-item">
<a class="nav-link active" href=" {% url 'home' %}">
    <i class="far fa-user-secret"></i> Inicio</a>
</li>      You, 13 seconds ago • Uncommitted changes
```

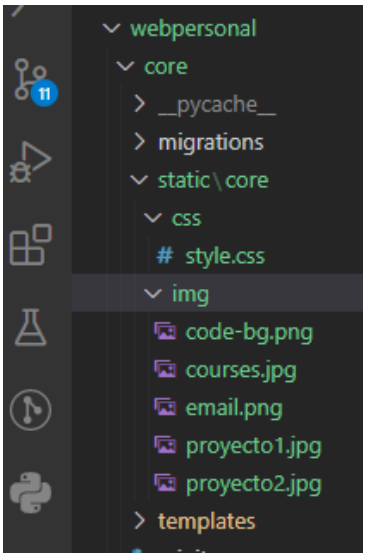
Archivos estáticos

Los recursos estáticos son:

- Archivos js
- Archivos css
- Imágenes

Para poderlos manejar, se debe crear la siguiente estructura:

- Dentro del directorio “core” crear la carpeta static/core
- Copiar los archivos correspondientes



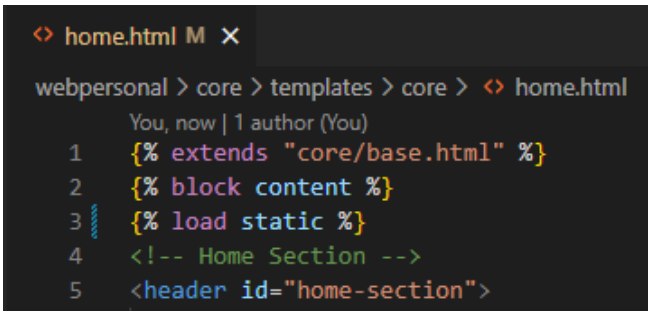
Django buscará todos los directorios estáticos de todas las aplicaciones para poderlos despachar.

En el template Base se debe indicar que “cargue” los archivos estáticos dentro del “head” mediante el siguiente tag

{% load static %}
Ahora para cargar un archivo estático se debe hacer de la siguiente manera (ejemplo):

```
20  </script>
21  {% load static %}
22  <link rel="stylesheet" href=" {% static 'core/css/style.css' %}" />
23  <title>Inicio | Francisco Vázquez</title>
```

En el home..



Reto

1. Dentro de la aplicación “core”, crear el template contact.html
    a. Al igual que se hizo en el template home.html, heredar del template base.html
    b. Colocar el html correspondiente
    c. Cambiar la ruta estática de la imagen a la que hace referencia
2. En el archivo views.py agregar la función contact para que renderice al template contact.html
3. En el archivo urls.py hacer referencia al template
    a.
```
urlpatterns = [
    path('admin/', admin.site.urls),
    path('', views.home, name='home'),
    path('/contact', views.contact, name='contact'),
]
```
4. En el archivo base.html modificar el menú.
5. Verificar su funcionamiento