

TALLER # 2

Frontend y BD

Contenido

INSTRUCCIONES PARA EL TALLER.....	2
INTRODUCCIÓN A BOOTSTRAP	3
¿Qué es Bootstrap?	3
Relación entre Django y Bootstrap.....	3
Inclusión de Bootstrap.....	3
Componente Card	5
CREACIÓN DE LA APP NEWS.....	9
Creación de la app news.....	9
Creación de ruta para la app news	10
Estilo Bootstrap para app news.....	13
POBLANDO AUTOMÁTICAMENTE LA BASE DE DATOS.....	14
Sobre la base de datos.....	15
Definir las librerías necesarias para el proyecto	15
Modificar el modelo Movies.....	16
Descargar y usar un dataset de películas	17
EXTENSIÓN DE TEMPLATE BASE	20
Plantilla base.....	20
Barra de navegación	21
Footer	26
MANEJO DE ARCHIVOS ESTÁTICOS	26
GRÁFICA A PARTIR DE LA INFORMACIÓN DE LA BASE DE DATOS	28
Recopilación de información de la base de datos.....	28
Definición de url	30
Definición de plantilla.....	30
NAVEGACIÓN ENTRE PÁGINAS.....	31
Enviar un formulario a otra página.....	31
Crear un enlace para “ir atrás”	34
BIBLIOGRAFÍA	36

INSTRUCCIONES PARA EL TALLER

- El taller está acompañado de una introducción temática realizada por el profesor. No se trata solo de seguir instrucciones. Comprenda los conceptos mientras sigue las instrucciones. Formule las preguntas que sean necesarias.
- Este taller es individual, evaluativo y presencial.
- El taller corresponde al 5% de la nota de la asignatura.
- La duración del taller es de aproximadamente 2 horas (máximo 3 horas).
- El taller consiste en seguir paso a paso este documento y entregar, antes de finalizar la clase, en el Buzón de Interactiva Virtual las evidencias (imágenes, enlaces, etc.) solicitadas.
- El punto de partida es el Taller 1. Si no lo tiene, puede **descargarlo en formato ZIP** desde: <https://github.com/paolavallejo/Taller-PI1-20241>. **NOTA: No haga un clon del repositorio.**
- **En el buzón de Interactiva Virtual entregue un documento PDF que contenga lo siguiente:**
 1. El enlace de su repositorio en GitHub (actualizado). Puede seguir usando el mismo repositorio de Taller 1 o crear un crear un repositorio específicamente para Taller 2.
 2. Una captura de pantalla con al menos 10 películas cargadas desde el dataset (*movies_initial.csv*), donde todas las películas están en *Cards* de Bootstrap y se vean todos los atributos, incluyendo *genre* y *year*. La captura debe contener el *navbar* (con una imagen estática) y el *footer*.
 3. Una captura de pantalla en la que la lista de películas se ajusta a una ventana de tamaño pequeño (reduzca el tamaño de la ventana del navegador). La captura debe contener el *navbar* (con una imagen estática).
 4. Una captura de pantalla de las “News” donde las novedades estén en *Horizontal Cards* de Bootstrap.
 5. Una captura de pantalla de la gráfica de películas por año.
 6. Una captura de pantalla de la gráfica de películas por género.

INTRODUCCIÓN A BOOTSTRAP

¿Qué es Bootstrap?

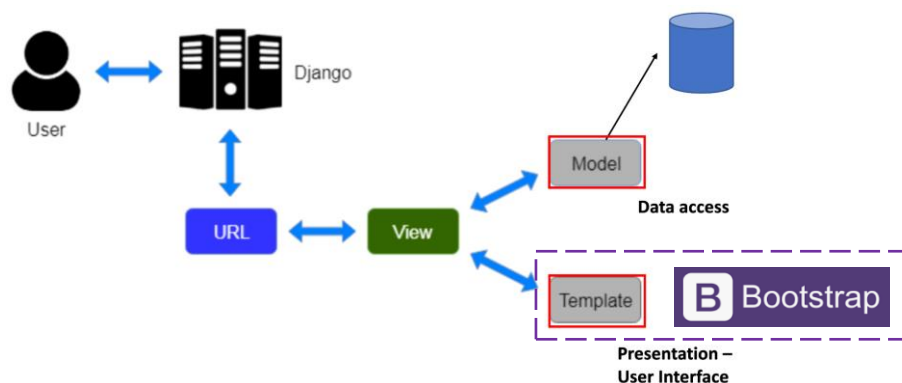
“Bootstrap is a powerful, feature-packed frontend toolkit. Build anything—from prototype to production—in minutes” (<https://getbootstrap.com/docs/5.3/getting-started/introduction/>). Bootstrap es un marco de trabajo (*framework*) de desarrollo *frontend* que se utiliza para crear **sitios web** (estático, informativo) y **aplicaciones web** (dinámico, acciones complejas de parte del usuario). Bootstrap proporciona una colección de herramientas y componentes de diseño basados en HTML, CSS y JavaScript que facilitan la creación de interfaces de usuario modernas y receptivas.

Algunas características Bootstrap son:

- Utiliza un sistema de rejilla (*grid system*) sensible que permite crear diseños flexibles y adaptativos para diferentes tamaños de pantalla y dispositivos.
- Proporciona una amplia gama de componentes preestilizados, como botones, formularios, barras de navegación, tarjetas, modales y más, que puede utilizar fácilmente en sus proyectos. Bootstrap incluye un conjunto de estilos prediseñados y utilidades de CSS que facilitan la personalización y la creación de diseños visualmente atractivos.
- Incluye varios *plugins* de JavaScript, como deslizadores de carrusel y acordeones, que pueden mejorar la funcionalidad del sitio web o aplicación web.

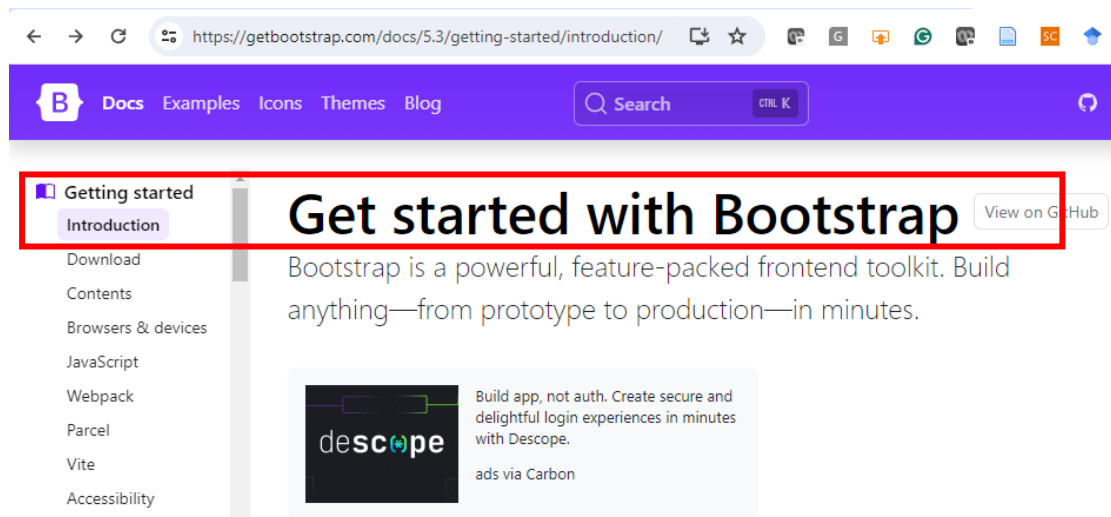
Relación entre Django y Bootstrap

Al combinar Django para la lógica del *backend* y Bootstrap para el diseño del *frontend*, es posible crear aplicaciones web modernas de manera eficiente, aprovechando la potencia de Django en el manejo de datos y la facilidad de uso de Bootstrap en la creación de interfaces de usuario atractivas y funcionales.



Inclusión de Bootstrap

1. Acceda a <https://getbootstrap.com/docs/5.3/getting-started/introduction/>.



2. Copie la etiqueta y el contenido de `<link>`. Copie también la etiqueta y el contenido de `<script>`. Ambas etiquetas están en la sección Quick start / 2. **Include Bootstrap's CSS and JS.**

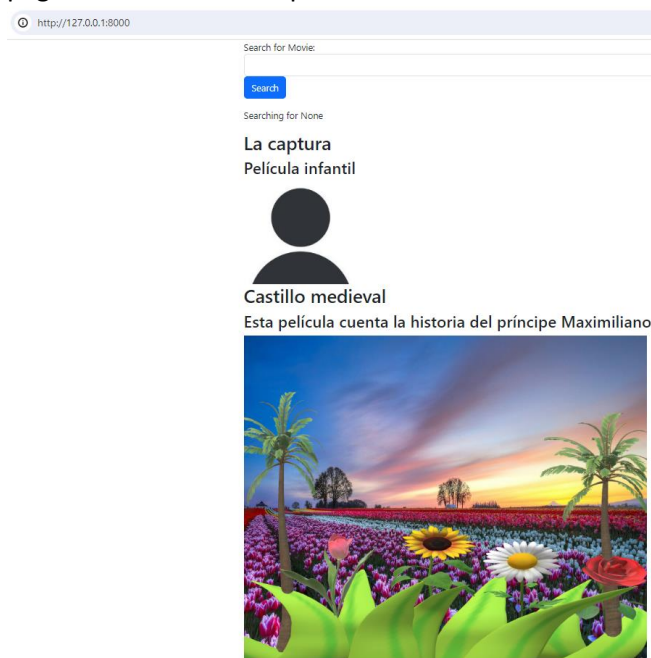
2. **Include Bootstrap's CSS and JS.** Place the `<link>` tag in the `<head>` for our CSS, and the `<script>` tag for our JavaScript bundle (including Popper for positioning dropdowns, poppers, and tooltips) before the closing `</body>`. Learn more about our [CDN links](#).

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Bootstrap demo</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-TW8Jv99CNev68xavRmh/xRmDn18151ra9kE47F3yKxjXg25kbbqD4987thK9" crossorigin="anonymous">
  </head>
  <body>
    <h1>Hello, world!</h1>
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.bundle.min.js" integrity="sha384-C6R5g80d6j94oAnCRpJ+t4YFb496BfCv7t+JHnAJt6Kt6JM9pXKL9mStQhK3" crossorigin="anonymous"></script>
  </body>
</html>
```

3. En el archivo **home.html**, agregue la etiqueta `<head>`, y al interior de esa etiqueta pegue lo que acaba de copiar desde el sitio de Bootstrap. Estas líneas de incluyen una hoja de estilo externa en el encabezado del documento HTML. `<head>` es una etiqueta que marca el inicio del encabezado del documento HTML. El encabezado contiene metadatos, como el título de la página, enlaces a hojas de estilo y scripts, entre otros. `<link>` se utiliza para incluir recursos externos en el documento HTML. `href="..."` especifica la URL del recurso externo que se va a incluir. En este caso, la URL apunta a un archivo CSS de Bootstrap alojado en un CDN (*Content Delivery Network*: red de servidores distribuidos geográficamente que trabajan juntos para proporcionar contenido web de manera eficiente a los usuarios). `rel="stylesheet"` especifica que el recurso externo es una hoja de estilo. `integrity="..."` se utiliza para garantizar que el recurso no se haya modificado en tránsito. En este caso, se proporciona una cadena hash SHA-384. `crossorigin="anonymous"` especifica cómo el navegador debe manejar las solicitudes de recursos cruzados. En este caso, indica que las solicitudes de recursos externos deben realizarse sin enviar credenciales del usuario.

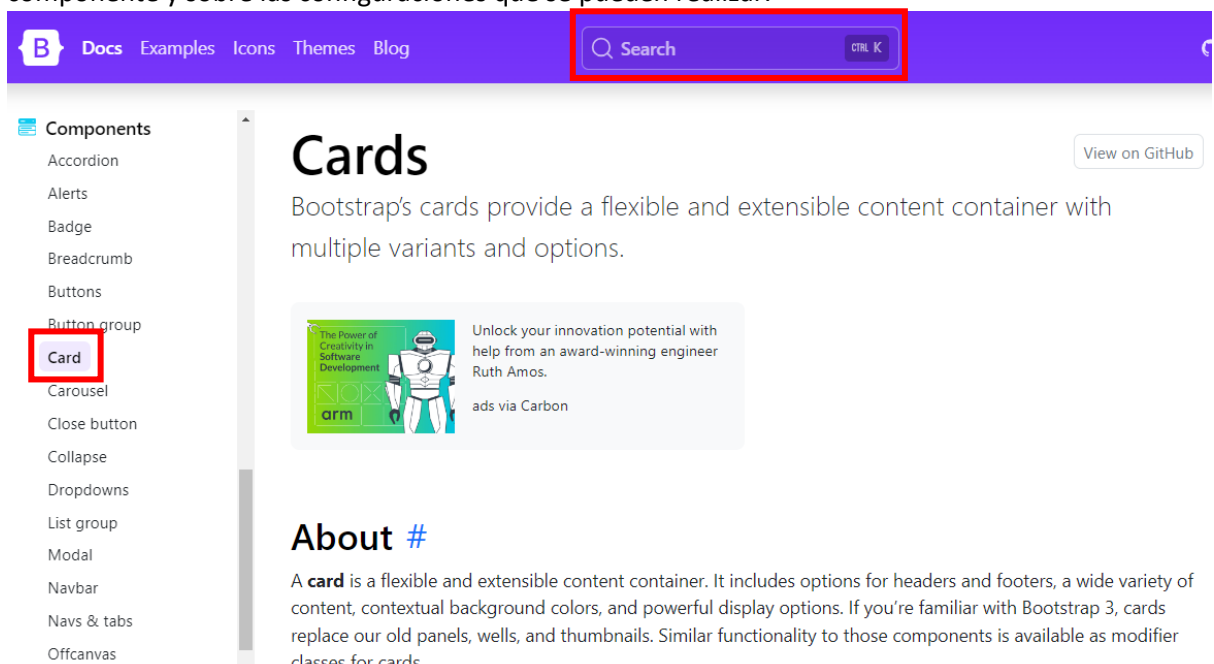
```
<> home.html M X
movie > templates > <> home.html
1 <!--<h1>Welcome to Home Page</h1-->
2 <!--<h1>Welcome to Home Page, <span style="color: blue;">{{ name }}</span></h1-->
3 <!--<h2>This is the full home page</h2-->
4
5 <head>
6   <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-
7   <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.bundle.min.js" crossorigin="anonymous"> </sc
8 </head>
```

4. Ejecute el servidor e ingrese al enlace <http://localhost:8000> para visualizar la información de la página con un estilo un poco diferente.

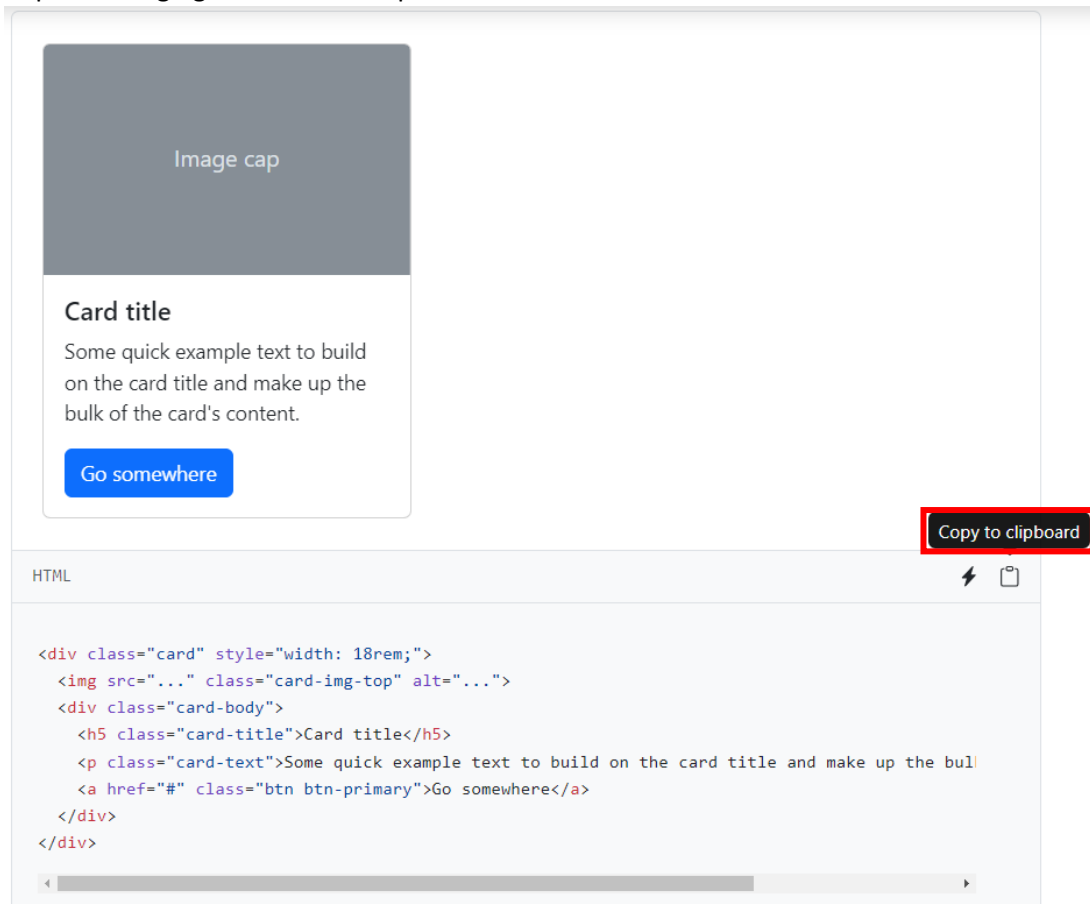


Componente Card

1. Ubique el componente Card, usando la barra de búsqueda o el menú lateral. Lea sobre este componente y sobre las configuraciones que se pueden realizar.



2. Copie el código genérico del componente Card.



The screenshot shows a web browser displaying a generic Card component. The card has a grey header with the text "Image cap", a white body with the title "Card title" and a paragraph of text, and a blue button labeled "Go somewhere". A red box highlights a "Copy to clipboard" button in the bottom right corner. Below the browser window, the HTML code for the card is displayed in a code editor.

```
HTML

<div class="card" style="width: 18rem;">
  
  <div class="card-body">
    <h5 class="card-title">Card title</h5>
    <p class="card-text">Some quick example text to build on the card title and make up the bulk of the card's content.</p>
    <a href="#" class="btn btn-primary">Go somewhere</a>
  </div>
</div>
```

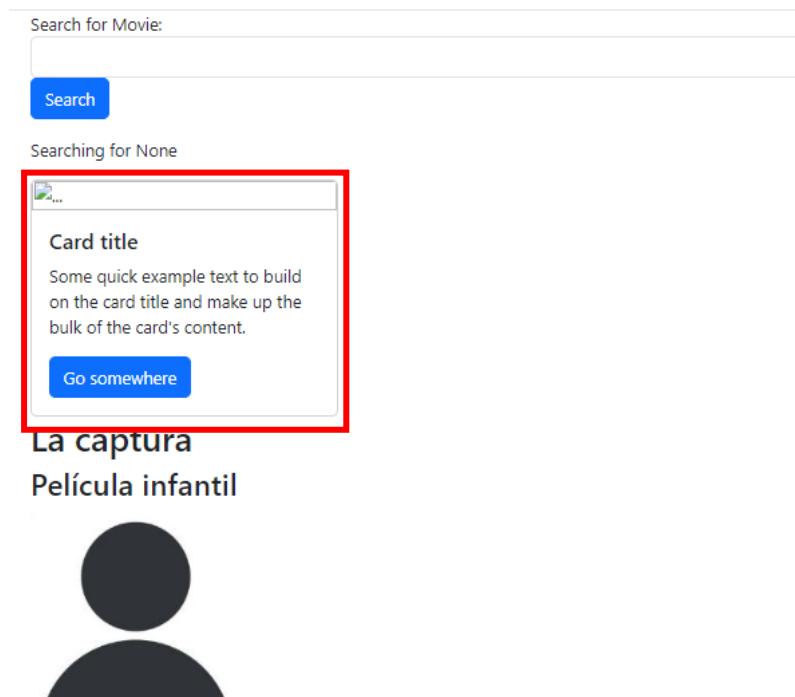
3. Pegue el código del componente Card en el archivo **home.html**.



The screenshot shows a code editor with the file "home.html" open. The code is in Jinja2 template syntax. A red box highlights the HTML code for the Card component, which is being pasted into the file. The code is as follows:

```
<div class="card" style="width: 18rem;">
  
  <div class="card-body">
    <h5 class="card-title">Card title</h5>
    <p class="card-text">Some quick example text to build on the card title and make up the bulk of the card's content.</p>
    <a href="#" class="btn btn-primary">Go somewhere</a>
  </div>
</div>
```

4. En el navegador ingrese al enlace <http://localhost:8000> y visualizará un Card genérico.



5. Edite el código del archivo **home.html** para agregar un Card por cada película, el cual contenga la información específica de dicha película.

```


<> home.html M X
movie > templates > <> home.html
8
9  <div class="container"> <!-- mejora el espaciado de la pagina-->
10  <form action="">
11      Search for Movie:
12      <input type="text" class="form-control" name="searchMovie"/>
13      <button type="submit" class="btn btn-primary">Search</button>
14  </form>
15  <p>Searching for {{searchTerm}}</p>
16  {% for movie in movies %}
17      <div class="card" style="width: 18rem;">
18          
19          <div class="card-body">
20              <h5 class="card-title">{{ movie.title }}</h5>
21              <p class="card-text">{{ movie.description }}</p>
22              {% if movie.url %}
23              <a href="{{ movie.url }}" class="btn btn-primary">Movie Link</a>
24              {% endif %}
25          </div>
26      </div>
27  {% endfor %}
28  <br/>
29  </div>
  
```

6. En el navegador ingrese al enlace <http://localhost:8000> y visualizará un Card por cada película que tenga en la base de datos.


Search for Movie:

Search

Searching for None



La captura
Película infantil



Castillo medieval
Esta película cuenta la historia del príncipe Maximiliano

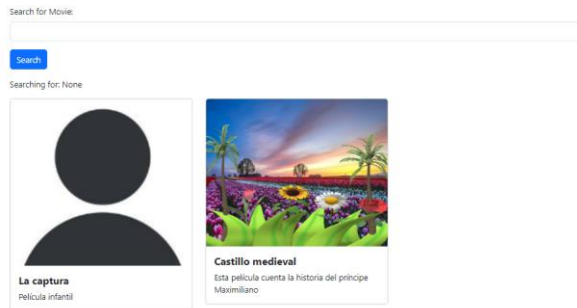
7. Edite el archivo **home.html** para mejorar el estilo. `<div class="mb-3">` agrega un espacio inferior de 3 unidades entre la barra de búsqueda y el botón *Search*. La etiqueta `<label>` se utiliza para asociar un texto descriptivo con un campo de entrada en un formulario. El atributo `for="searchMovie"` especifica qué campo de entrada está etiquetando. `<div class="row row-cols-1 row-cols-md-3 g-4">` define una fila en una cuadrícula de Bootstrap con una sola columna en pantallas extra pequeñas y tres columnas en pantallas medianas y más grandes.

```

< home.html M X
movie > templates > < home.html
9 <div class="container"> <!-- mejora el espaciado de la pagina-->
10 <form action=""> <!-- especifica que al hacer clic en enviar, enviamos el formulario a la misma página-->
11 <div class="mb-3"> <!-- agrega espacio inferior de 3 unidades-->
12 <label for="searchMovie" class="form-label"> <!-- asocia un texto descriptivo con un campo de entrada en un formulario -->
13 Search for Movie:
14 </label>
15 <input type="text" class="form-control" name="searchMovie"/>
16 </div>
17 <button type="submit" class="btn btn-primary">Search</button>
18 </form>
19 <n>Searching for: {{searchTerm}}</n>
20 <div class="row row-cols-1 row-cols-md-3 g-4"> <!-- define una fila en una cuadrícula con una columna en pantallas extra pequeñas y tres columnas en pantallas medianas y más grandes -->
21 {% for movie in movies %}
22 <div v-for="movie in movies" class="col">
23 <div class="card">
24 
25 <div class="card-body">
26 <h5 class="card-title fw-bold">{{ movie.title }}</h5>
27 <p class="card-text">{{ movie.description }}</p>
28 {% if movie.url %}
29 <a href="{{ movie.url }}" class="btn btn-primary">Movie Link</a>
30 {% endif %}
31 </div>
32 </div>
33 </div>
34 {% endfor %}
35 </div>

```


- En el navegador acceda al enlace <http://localhost:8000> para visualizar el detalle de las películas.



- Reduzca el ancho de la ventana del navegador para visualizar el ajuste de los cards. Las películas deben acomodarse en vertical (una bajo la otra).



CREACIÓN DE LA APP NEWS

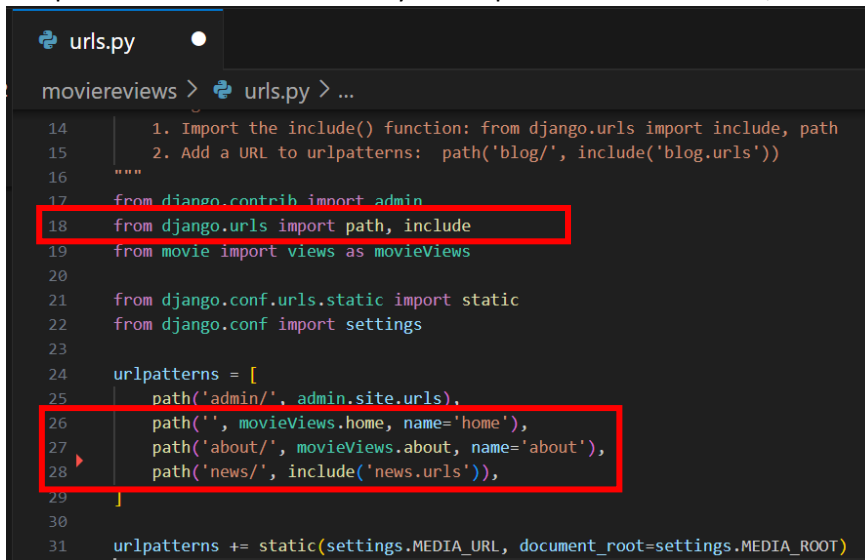
Una *app* en Django es un componente modular que proporciona funcionalidades específicas dentro de un proyecto. Contiene archivos que gestionan modelos de datos, vistas, plantillas y archivos estáticos relacionados con una tarea particular. Esto facilita la organización y reutilización del código.

Creación de la app news

- Hasta ahora tenemos únicamente la app **movie**, crearemos la app **news** para publicar noticias. Para esto debe seguir el mismo proceso de la creación de la app movie. Desde la Terminal, ubíquese en la carpeta del proyecto **moviereviewsproject** (NO en la carpeta interna) y ejecute el comando: `python manage.py startapp news` para crear la nueva app. Si tiene dudas puede ver la sección **Creación de una aplicación con el comando startapp de Taller 1**.
- Verifique que la carpeta **news** se creó dentro de la carpeta del proyecto.
- Agregue la app al archivo **settings.py** de **moviereviews**, en la sección de **INSTALLED_APPS**. Si tiene dudas puede revisar las instrucciones en el **Taller 1**.

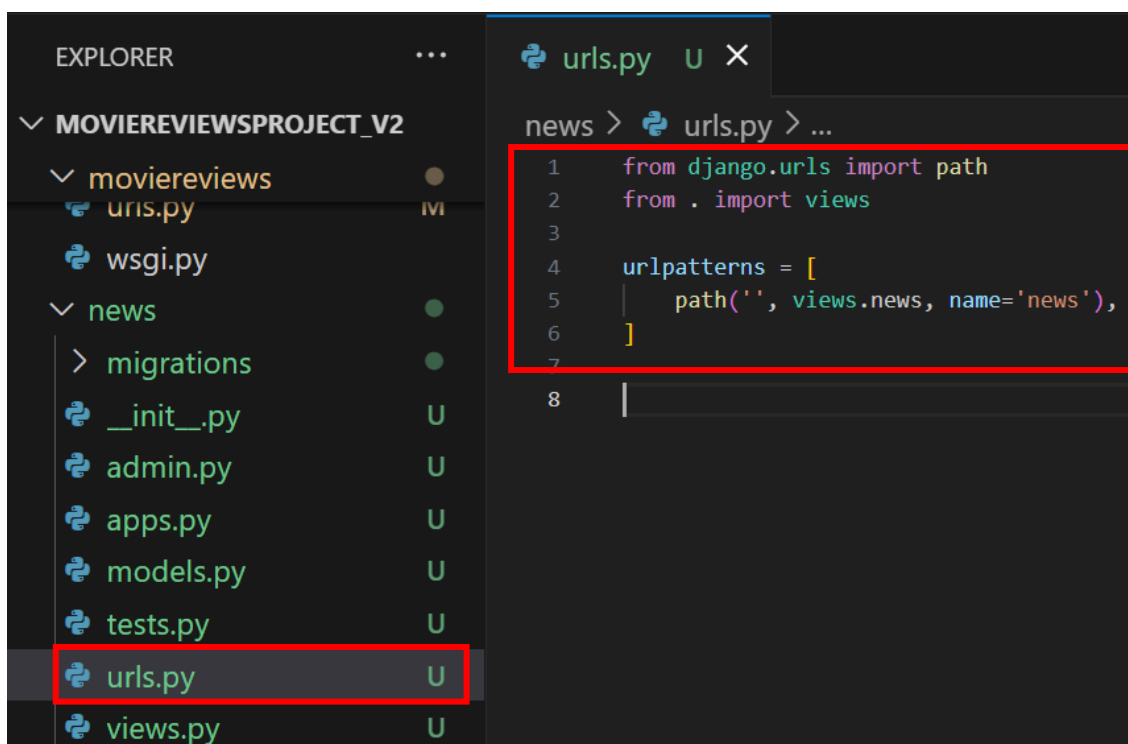
Creación de ruta para la app news

1. Edite el archivo **urls.py** de **moviereviews** para agregar la ruta hacia **news**. Si agregáramos más rutas, pronto sería difícil distinguir qué rutas corresponden a qué aplicación. Para separar mejor las rutas en sus propias aplicaciones, cada aplicación puede tener su propio archivo **urls.py**. La función `include()` importa y vincula las rutas definidas en otro archivo **urls.py**. En este caso, está importando las rutas definidas en el archivo `urls.py` de la aplicación llamada **news**. `path('news/', include('news.urls'))` define una ruta en el proyecto principal que comienza con `'news/'`. Cuando el servidor recibe una solicitud con este patrón de URL, el sistema de enrutamiento de Django buscará coincidencias en las rutas definidas en el archivo **urls.py** de la aplicación **news** y manejará la solicitud según corresponda. Complemente las rutas de **home** y **about** para darles un nombre, lo cual facilitará su referenciación.



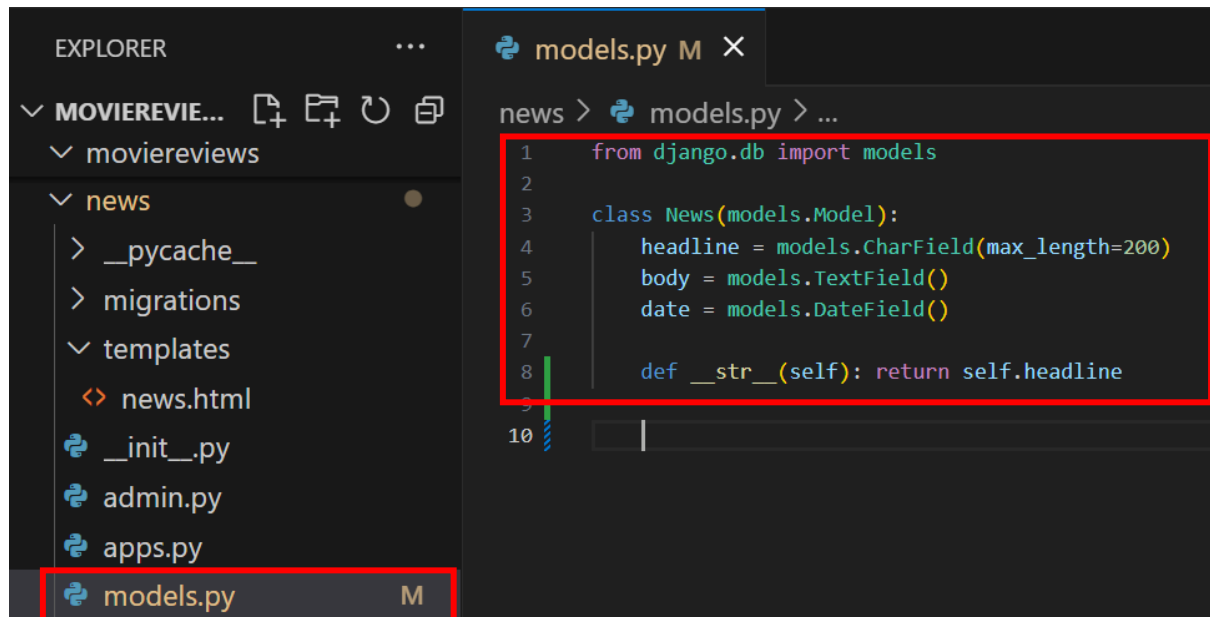
```
14 1. Import the include() function: from django.urls import include, path
15 2. Add a URL to urlpatterns: path('blog/', include('blog.urls'))
16 """
17 from django.contrib import admin
18 from django.urls import path, include
19 from movie import views as movieViews
20
21 from django.conf.urls.static import static
22 from django.conf import settings
23
24 urlpatterns = [
25     path('admin/', admin.site.urls),
26     path('', movieViews.home, name='home'),
27     path('about/', movieViews.about, name='about'),
28     path('news/', include('news.urls')),
29 ]
30
31 urlpatterns += static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

2. Cree el archivo **urls.py** en la carpeta de la app **news** y defina la ruta principal para la app. `from . import views` se utiliza para importar el módulo **views** del mismo directorio en el que se encuentra el archivo actual. La ruta definida redirige una solicitud, por ejemplo, `localhost:8000/news`, a la vista de noticias.



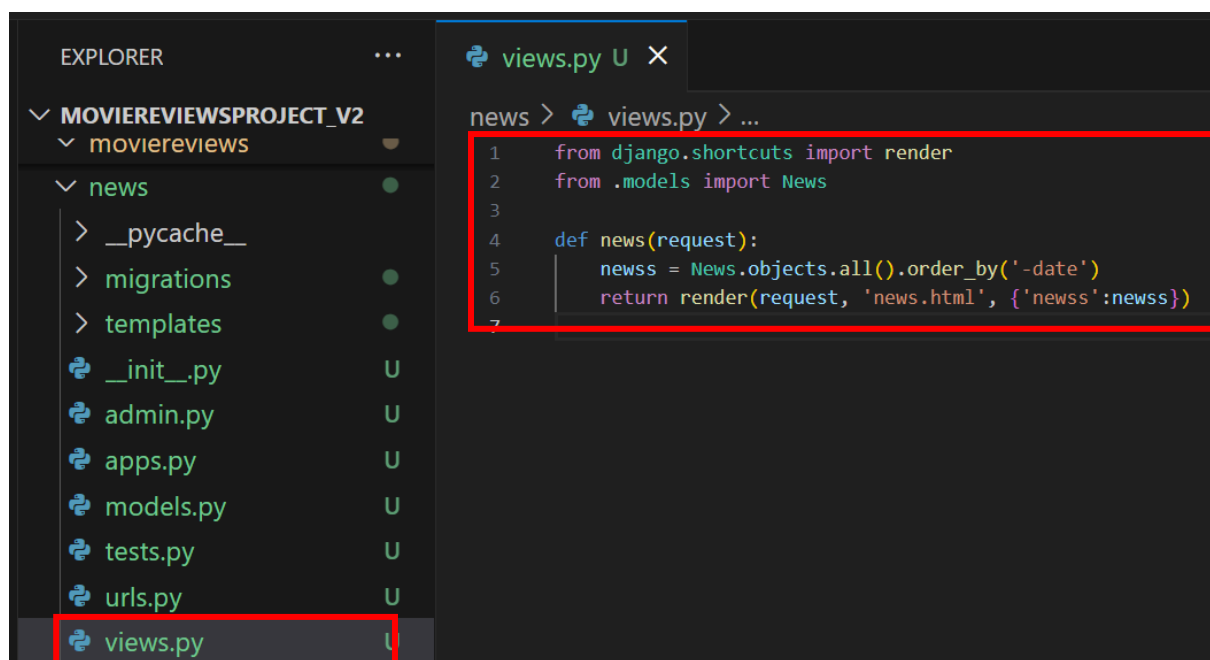
```
1 from django.urls import path
2 from . import views
3
4 urlpatterns = [
5     path('', views.news, name='news'),
6 ]
7
8
```

- Defina el modelo de **news** en el archivo **news/models.py**, este contendrá un encabezado, el cuerpo y la fecha de la noticia. Django automáticamente agrega una clave primaria a cada tabla. `def __str__(self): return self.headline` permite ver el encabezado de la película en la interfaz de Admin, en lugar de ver los elementos como Object (1), Object (2), etc.



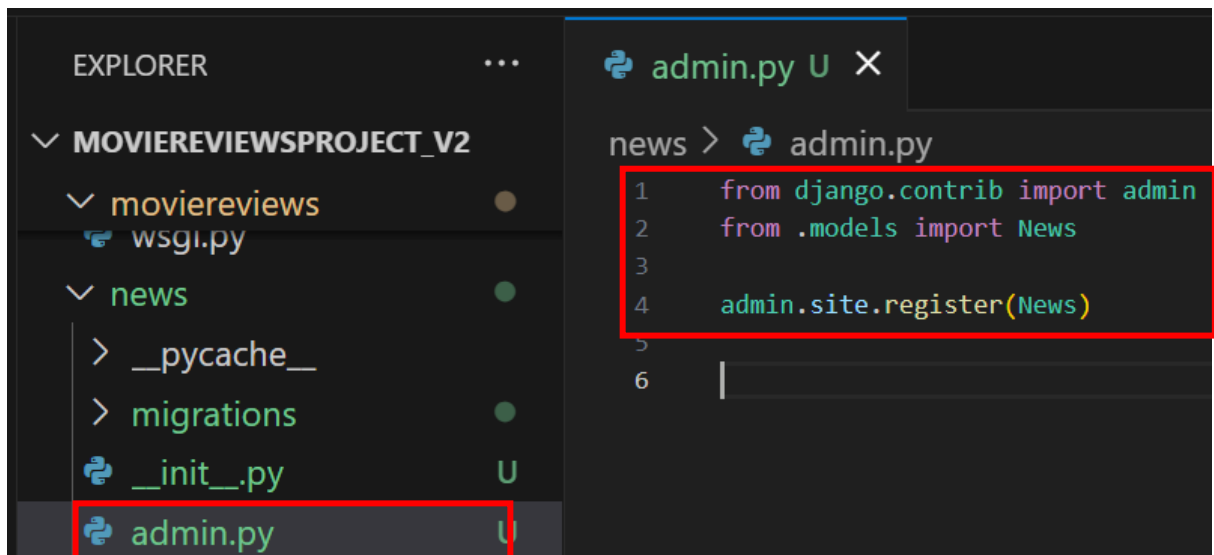
```
1 from django.db import models
2
3 class News(models.Model):
4     headline = models.CharField(max_length=200)
5     body = models.TextField()
6     date = models.DateField()
7
8     def __str__(self): return self.headline
```

- Cree la función **news** en el archivo **views.py** de la app **news** para listar todas las noticias, ordenadas por fecha. `order_by('-date')` ordena los resultados de la consulta según el campo `date` de la clase `News`. El prefijo `-` indica que la ordenación se realizará en orden descendente, es decir, desde la fecha más reciente hasta la más antigua. la consulta SQL equivalente sería algo como esto: `SELECT * FROM News ORDER BY date DESC;`



```
1 from django.shortcuts import render
2 from .models import News
3
4 def news(request):
5     newss = News.objects.all().order_by('-date')
6     return render(request, 'news.html', {'newss':newss})
```

- Registre el modelo en el archivo **admin.py** de **news**.



6. Dado que agregó un nuevo modelo, debe hacer las migraciones usando los comandos: `python manage.py makemigrations` y `python manage.py migrate`. Si tiene dudas puede revisar las instrucciones en el Taller 1.
7. Cree la carpeta **templates** y, dentro de la carpeta cree el archivo **news.html**. Muestre todas las noticias con sus correspondientes atributos.



8. Ejecute el servidor y acceda a la interfaz de Admin para agregar noticias. Si tiene dudas puede revisar las instrucciones en el Taller 1.
9. Acceda a <http://localhost:8000/news/> para visualizar el resultado.



Estilo Bootstrap para app news

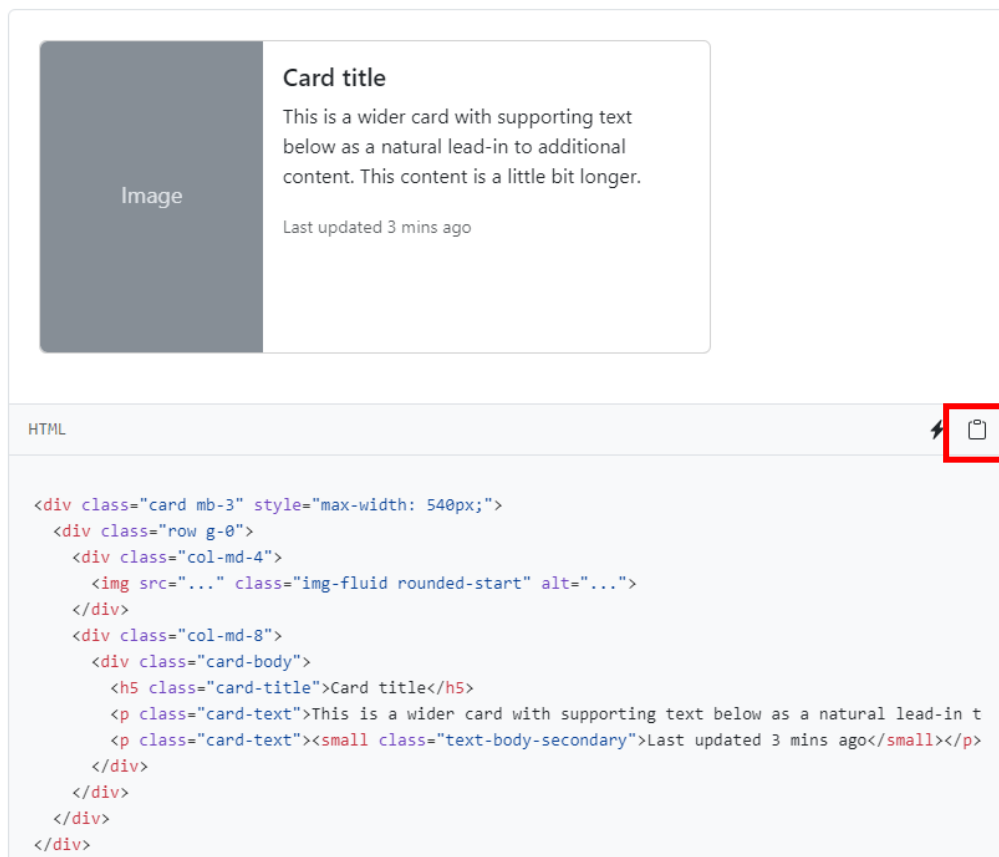
1. Agregue el enlace de Bootstrap al `<head>` de **news.html** (ver sección **Inclusión de Bootstrap**).



2. Mejore el aspecto de la página de noticias usando el componente Horizontal Card de Bootstrap. Copie el ejemplo de este componente desde la página de Bootstrap.

Horizontal

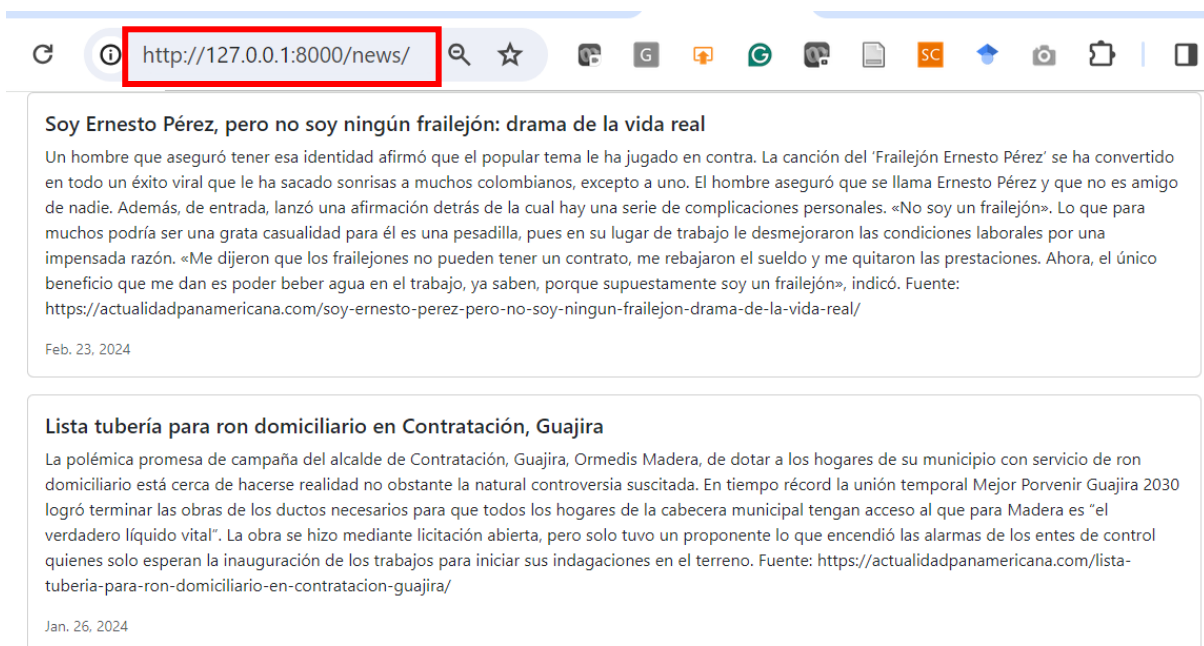
Using a combination of grid and utility classes, cards can be made horizontal in a mobile-friendly and responsive way. In the example below, we remove the grid gutters with `.g-0` and use `.col-md-*` classes to make the card horizontal at the `md` breakpoint. Further adjustments may be needed depending on your card content.



3. Pegue el ejemplo en el archivo **news.html** y edítelo. Considere las mejoras que realizó en la página **home.html**. Elimine la etiqueta de la imagen. `text-muted` se usa para aplicar un estilo de texto gris claro para indicar que el texto es menos relevante o importante.

```
<> news.html X
news > templates > <> news.html
1 <head>
2   <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="styl
3   </head>
4
5 <div class="container">
6   {% for news in newss %}
7   <div class="card mb-3">
8     <div class="row g-0">
9       <div class="col-md-8">
10        <div class="card-body">
11          <h5 class="card-title">{{ news.headline }}</h5>
12          <p class="card-text">{{ news.body }}</p>
13          <p class="card-text"><small class="text-muted">{{ news.date }}</small></p>
14        </div>
15      </div>
16    </div>
17  </div>
18  {% endfor %}
19 </div>
20
```

4. Acceda a <http://localhost:8000/news/> para visualizar el resultado.



POBLANDO AUTOMÁTICAMENTE LA BASE DE DATOS

Poblar automáticamente una base de datos tiene varias ventajas en el desarrollo de software. Primero, ahorra tiempo y esfuerzo al introducir datos de prueba o datos iniciales en la base de datos, lo que acelera

el proceso de desarrollo y pruebas. Además, garantiza la consistencia y la integridad de los datos al eliminar errores humanos en la introducción manual de datos.

Sobre la base de datos

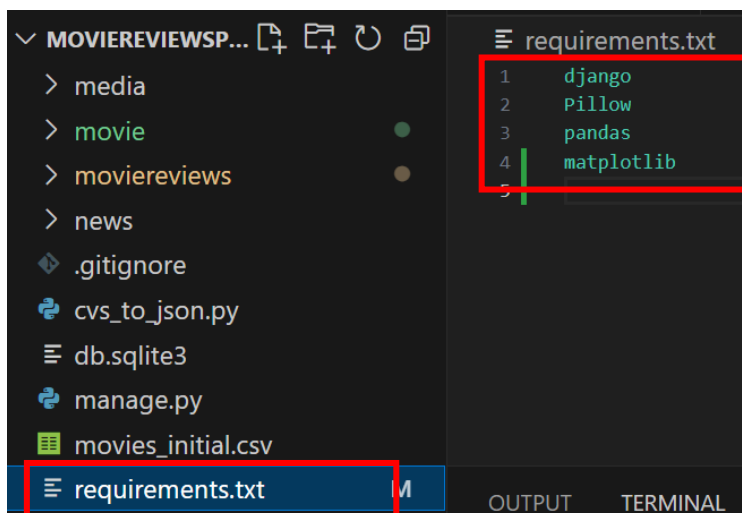
Actualmente, usamos una base de datos SQL, la cual se define en el archivo **settings.py**, en la zona de **DATABASES**. **SQLite** es la base de datos que Django usa por defecto, es muy apropiada para proyectos pequeños. Se ejecuta desde un solo archivo y no requiere una instalación compleja. En contraste, las otras opciones implican cierta complejidad para configurarlas adecuadamente. Si en algún momento necesita cambiar de base de datos, debe modificar los atributos **ENGINE** y **NAME**.

```
83 DATABASES = {
84     'default': {
85         'ENGINE': 'django.db.backends.sqlite3',
86         'NAME': BASE_DIR / 'db.sqlite3',
87     }
88 }
```

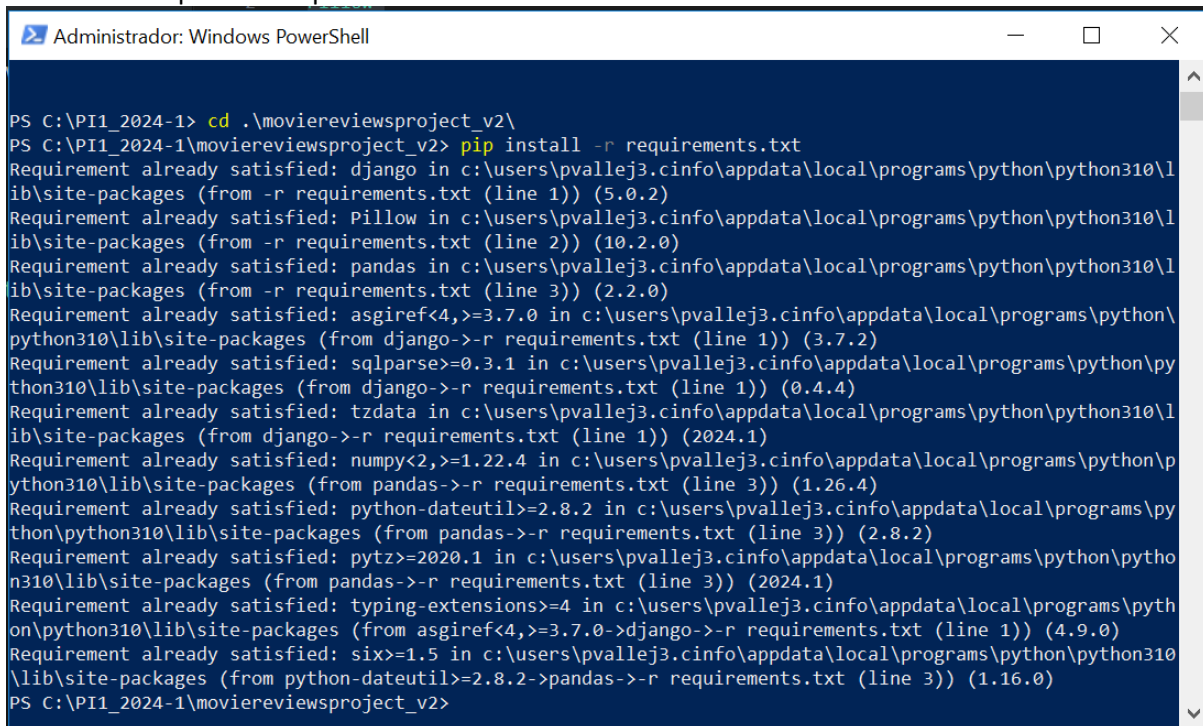
Definir las librerías necesarias para el proyecto

Cuando desarrollamos software es común utilizar un conjunto de librerías, estas librerías se pueden descargar directamente desde el sitio web, clonar desde un repositorio, o instalar directamente usando pip. Sin embargo, este procedimiento manual es muy tedioso e ineficiente. Si las librerías se borran o se necesita utilizar un computador diferente para continuar con el desarrollo del proyecto, es necesario volver a buscar las librerías y descargarlas una por una. El archivo **requirements.txt** permite automatizar la instalación de librerías Python.

1. Cree el archivo **requirements.txt** en la carpeta raíz del proyecto. En ese archivo, debe listar todas las librerías necesarias para el funcionamiento del proyecto. Si se necesita una versión específica de alguna librería se debe especificar de la siguiente forma: `numpy==1.20.1`. Por ahora, dado que no requerimos versiones específicas, se puede dejar el archivo como está. Puede agregar librerías en cualquier momento.



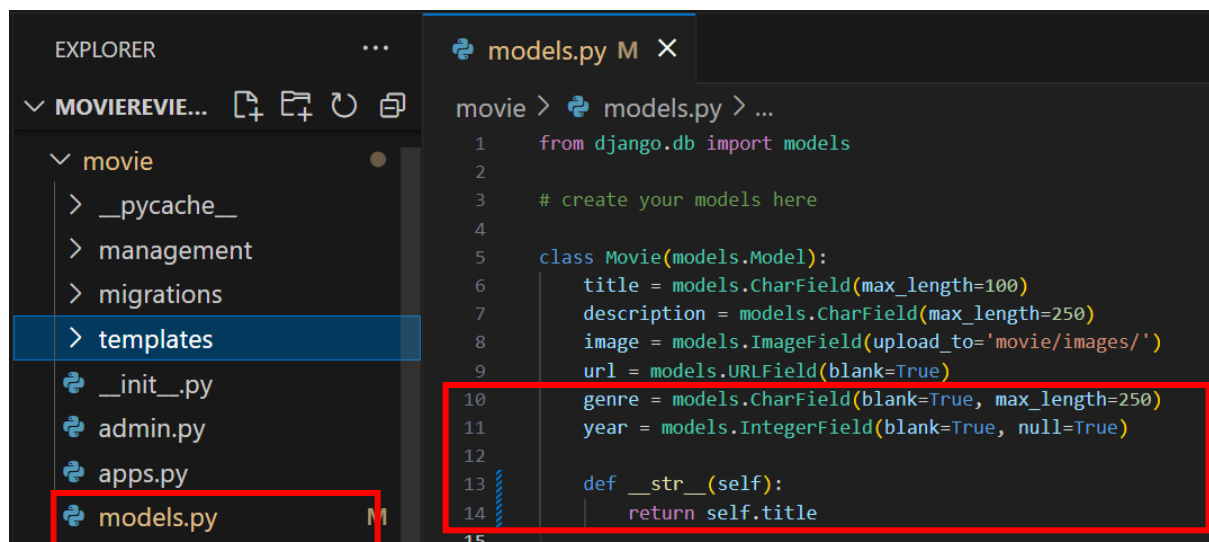
2. Desde la consola ubicada en la carpeta donde se encuentra el archivo `requirements.txt` ejecute el comando `pip install -r requirements.txt`. Después de unos segundos la instalación debe quedar completa.



```
PS C:\PI1_2024-1> cd .\moviereviewsproject_v2\
PS C:\PI1_2024-1\moviereviewsproject_v2> pip install -r requirements.txt
Requirement already satisfied: django in c:\users\pvallej3.cinfo\appdata\local\programs\python\python310\lib\site-packages (from -r requirements.txt (line 1)) (5.0.2)
Requirement already satisfied: Pillow in c:\users\pvallej3.cinfo\appdata\local\programs\python\python310\lib\site-packages (from -r requirements.txt (line 2)) (10.2.0)
Requirement already satisfied: pandas in c:\users\pvallej3.cinfo\appdata\local\programs\python\python310\lib\site-packages (from -r requirements.txt (line 3)) (2.2.0)
Requirement already satisfied: asgiref<4,>=3.7.0 in c:\users\pvallej3.cinfo\appdata\local\programs\python\python310\lib\site-packages (from django->-r requirements.txt (line 1)) (3.7.2)
Requirement already satisfied: sqlparse>=0.3.1 in c:\users\pvallej3.cinfo\appdata\local\programs\python\python310\lib\site-packages (from django->-r requirements.txt (line 1)) (0.4.4)
Requirement already satisfied: tzdata in c:\users\pvallej3.cinfo\appdata\local\programs\python\python310\lib\site-packages (from django->-r requirements.txt (line 1)) (2024.1)
Requirement already satisfied: numpy<2,>=1.22.4 in c:\users\pvallej3.cinfo\appdata\local\programs\python\python310\lib\site-packages (from pandas->-r requirements.txt (line 3)) (1.26.4)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\pvallej3.cinfo\appdata\local\programs\python\python310\lib\site-packages (from pandas->-r requirements.txt (line 3)) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in c:\users\pvallej3.cinfo\appdata\local\programs\python\python310\lib\site-packages (from pandas->-r requirements.txt (line 3)) (2024.1)
Requirement already satisfied: typing-extensions>=4 in c:\users\pvallej3.cinfo\appdata\local\programs\python\python310\lib\site-packages (from asgiref<4,>=3.7.0->django->-r requirements.txt (line 1)) (4.9.0)
Requirement already satisfied: six>=1.5 in c:\users\pvallej3.cinfo\appdata\local\programs\python\python310\lib\site-packages (from python-dateutil>=2.8.2->pandas->-r requirements.txt (line 3)) (1.16.0)
PS C:\PI1_2024-1\moviereviewsproject_v2>
```

Modificar el modelo Movies

1. Agregue los atributos **genre** y **year** al modelo de **Movies**. Luego, realice las migraciones (ya que realizamos cambios en el modelo). El campo **year** será un entero porque un campo *IntegerField* proporciona la flexibilidad necesaria sin tener que preocuparse por el formato de fecha. **Si tiene dudas puede revisar las instrucciones en el Taller 1.**

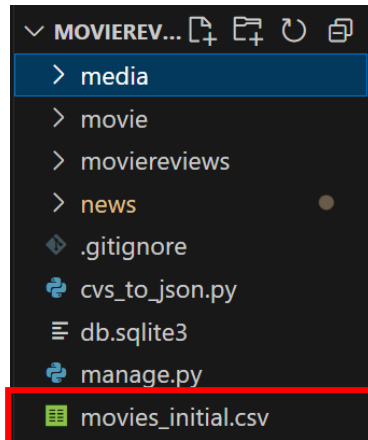


```
EXPLORER
MOVIEREVIEW...
  movie
    __pycache__
    management
    migrations
    templates
  __init__.py
  admin.py
  apps.py
  models.py

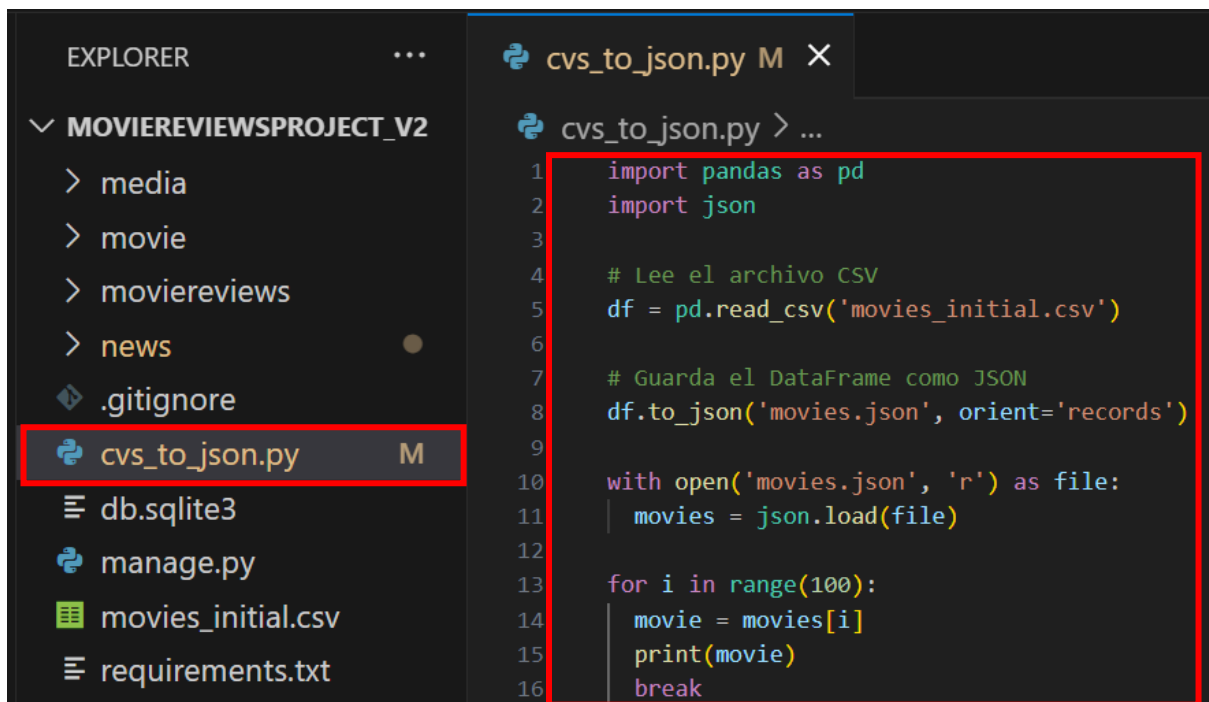
models.py M X
movie > models.py > ...
1  from django.db import models
2
3  # create your models here
4
5  class Movie(models.Model):
6      title = models.CharField(max_length=100)
7      description = models.CharField(max_length=250)
8      image = models.ImageField(upload_to='movie/images/')
9      url = models.URLField(blank=True)
10     genre = models.CharField(blank=True, max_length=250)
11     year = models.IntegerField(blank=True, null=True)
12
13     def __str__(self):
14         return self.title
15
```


Descargar y usar un dataset de películas

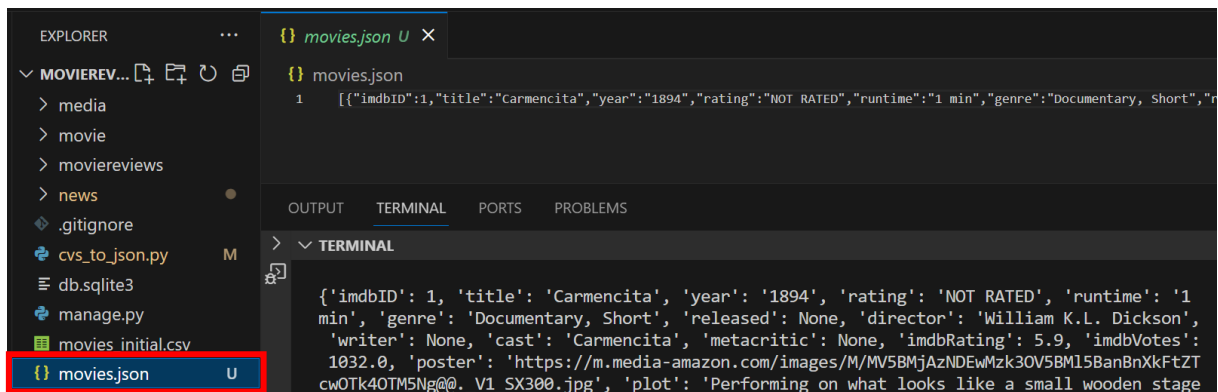
1. Descargue el *dataset* de películas *movies_initial.csv* (será proporcionado junto con el taller) y almacénelo en la carpeta raíz del proyecto. Existen diversos repositorios de *dataset* (por ejemplo, desde <https://www.kaggle.com/datasets>), podría descargar cualquier dataset, siempre y cuando contenga la información que requiere para poblar la base de datos.



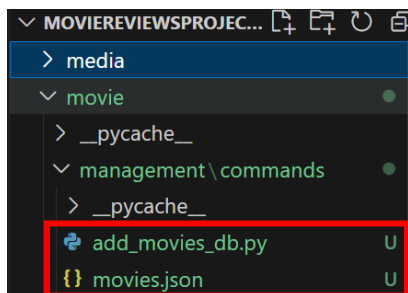
2. Cree el archivo **cvs_to_json.py** en la carpeta raíz del proyecto. Implemente el código para leer el archivo csv y generar un archivo json que contenga la información de las películas. *orient='records'* especifica que se debe guardar cada fila del DataFrame como un registro en el archivo JSON. **NOTA:** Es posible trabajar directamente con el archivo en formato CSV, pero haremos conversión a formato JSON para demostrar que es posible trabajar en diferentes formatos y practicar algunos conceptos de programación.



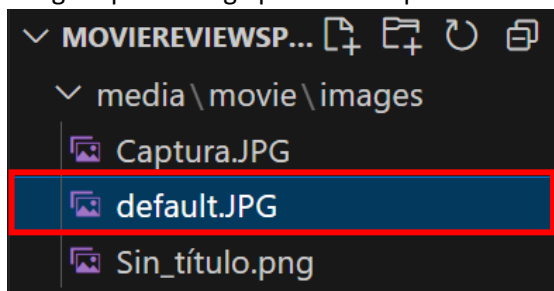
3. Ejecute el archivo **cvs_to_json.py** usando el comando `python cvs_to_json.py`. Luego, verifique que se creó el archivo **movies.json**.



4. Dentro de la carpeta de la app movie, cree la carpeta **management**. Luego, dentro de la carpeta management, cree la carpeta **commands**. Finalmente, dentro de la carpeta commands cree el archivo **add_movies_db.py**. Este archivo se utilizará para pasar la información del archivo json a la base de datos de películas de la aplicación de Django. **NOTA: el archivo debe estar en esta ubicación para tener acceso con permisos de edición a la base de datos, de lo contrario no se podrá modificar automáticamente y tendrá que hacerlo manual desde la interfaz de Admin.** Desplace el archivo **movies.json** a la carpeta **movie/management/commands**.



5. Dentro de la carpeta **media/movie/images** guarde una imagen llamada **default.jpg**. Esta será la imagen que se carga por defecto para todas las películas.



6. Complete el archivo **add_movies_db.py** para extraer la información de 100 películas del archivo json y llevarla a la base de datos (el archivo de código será proporcionado junto con el taller). Concretamente, extraerá el título, el género, el año y la descripción. Se cargará la imagen por defecto (**default.JPG**).

```

add_movies_db.py X
movie > management > commands > add_movies_db.py > ...
1 from django.core.management.base import BaseCommand
2 from movie.models import Movie
3 import os
4 import json
5
6 class Command(BaseCommand):
7     help = 'Load movies from movie_descriptions.json into the Movie model'
8
9     def handle(self, *args, **kwargs):
10         # Construct the full path to the JSON file
11         # Recuerde que la consola está ubicada en la carpeta DjangoProjectBase.
12         # El path del archivo movie_descriptions con respecto a DjangoProjectBase sería la carpeta anterior
13         json_file_path = 'movie/management/commands/movies.json'
14
15         # Load data from the JSON file
16         with open(json_file_path, 'r') as file:
17             movies = json.load(file)
18
19         # Add products to the database
20         for i in range(100):
21             movie = movies[i]
22             exist = Movie.objects.filter(title = movie['title']).first() #Se asegura que la película no exista en la base de datos
23             if not exist:
24                 Movie.objects.create(title = movie['title'],
25                                     image = 'movie/images/default.jpg',
26                                     genre = movie['genre'],
27                                     year = movie['year'],
28                                     description = movie['plot'],)
29

```

- Ejecute el archivo **add_movies_db.py** con el comando **python manage.py add_movies_db**. Acaba de crear un comando personalizado.
- Ejecute el servidor y visualice la lista de películas desde el Admin. Acaba d poblar su base de datos de forma automática, a partir de un *dataset* existente.

☐ MOVIE

☐ The Ocean Waif

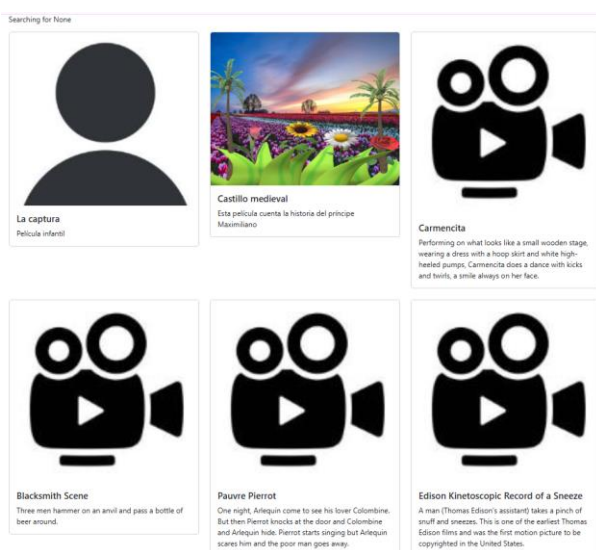
☐ Making a Living

☐ Mabel's Married Life

☐ Mabel at the Wheel

☐ Laughing Gas

- Visualice la lista de películas desde el *home*.



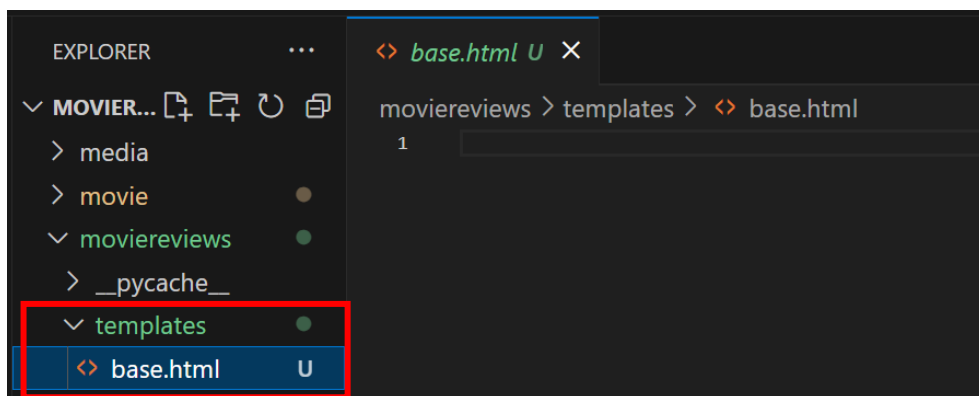
- Edite el archivo **home.html** para mostrar todos los atributos de las películas, incluyendo **genre** y **year**.
- Ejecute el servidor y visualice la lista de películas desde el *home*.

EXTENSIÓN DE TEMPLATE BASE

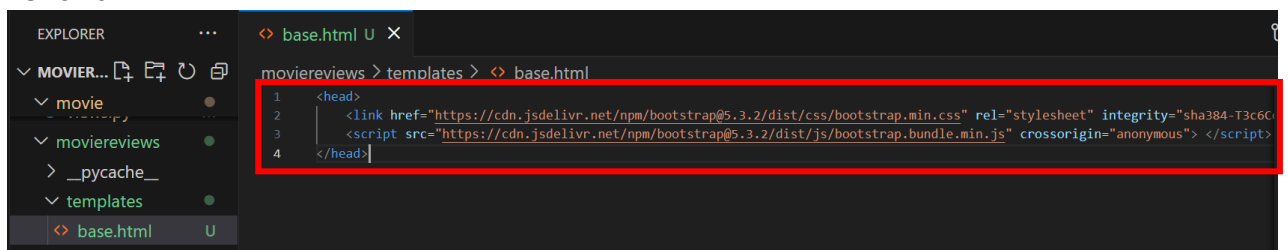
Definir un estilo y copiarlo en todas las páginas es una tarea tediosa, por lo tanto, definiremos un *template* de base, el cual usaremos en las diferentes páginas que tengamos. Utilizaremos plantillas base donde podamos agregar la barra de navegación y el *footer* a cada página individual. Esto nos permite realizar cambios en nuestra barra de navegación en un solo lugar y se aplicarán a cada página. **NOTA: Todos los elementos que sean globales, es decir, que se usarán en todas las páginas individuales, se definirán en la carpeta principal del proyecto (donde se encuentra el archivo *settings.py*), no en las carpetas de las aplicaciones.**

Plantilla base

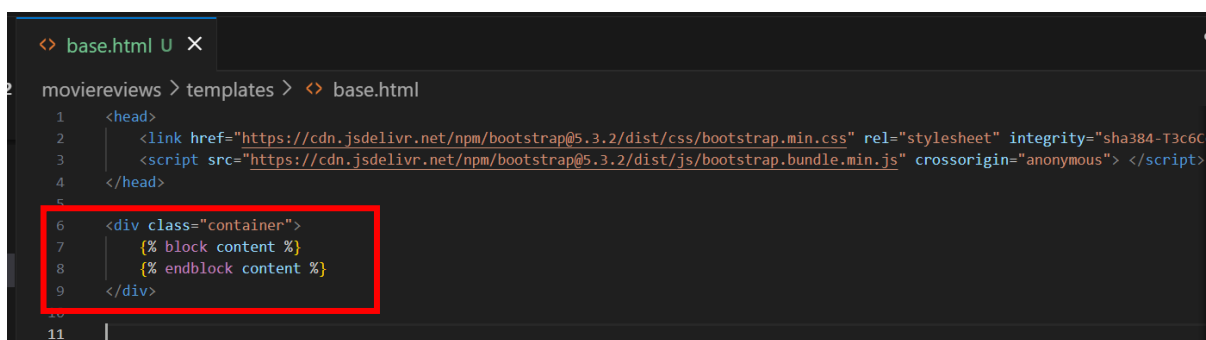
1. En la carpeta **moviereviews**, cree una carpeta llamada **templates**. En esa carpeta cree el archivo **base.html**.



2. Corte todo el **<head>** de **home.html** y péguelo en **base.html**. Además, elimine el **<head>** de **news.html**.



3. Al final de **base.html** agregue el código que se muestra en la imagen. Esta estructura permite definir un área de contenido que puede ser reemplazada por contenido específico en plantillas hijas. **<div class="container">** define un contenedor para el contenido. **{% block content %}** inicia un bloque. **{% endblock content %}** finaliza el bloque.



4. Registre la nueva carpeta **templates** en la zona **TEMPLATES** de **settings.py**. Esto crea una lista que contiene una única ruta de directorio (la ruta completa al directorio **templates** dentro del directorio **moviereviews**) en el proyecto, usando la variable **BASE_DIR** para formar la ruta base.

Barra de navegación

Actualmente tenemos nuestra página de películas, página de registro para la lista de correos y página de noticias. Pero los usuarios tienen que ingresar manualmente la URL para navegar a cada una de las páginas, lo cual no es ideal. Agregaremos una barra de navegación que les permita desplazarse entre las páginas. Utilizamos como base el componente *navbar* de Bootstrap.

1. Busque el componente **navbar** y copie el código.

2. Pegue el código en el archivo **base.html** y elimine los elementos que no usará. Cambie los nombres de los ítems del menú. Puedes copiar y pegar el siguiente código.

```
3. <nav class="navbar navbar-expand-lg bg-body-tertiary">
4.   <div class="container-fluid">
5.     <a class="navbar-brand" href="#">Movies</a>
6.     <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-
target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-expanded="false" aria-
label="Toggle navigation">
7.       <span class="navbar-toggler-icon"></span>
8.     </button>
9.     <div class="collapse navbar-collapse" id="navbarSupportedContent">
10.      <ul class="navbar-nav me-auto mb-2 mb-lg-0">
11.        <li class="nav-item">
12.          <a class="nav-link active" aria-current="page" href="#">News</a>
13.        </li>
14.        <li class="nav-item">
```

```

15.         <a class="nav-link" href="#">Login</a>
16.     </li>
17.     <li class="nav-item">
18.         <a class="nav-link" href="#">Sign Up</a>
19.     </li>
20. </ul>
21. </div>
22. </div>
23. </nav>

```

```

<> base.html U X
moviereviews > templates > <> base.html
4     </head>
5
6     <nav class="navbar navbar-expand-lg bg-body-tertiary">
7         <div class="container-fluid">
8             <a class="navbar-brand" href="#">Movies</a>
9             <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarSupportedContent" aria-controls=
10                 <span class="navbar-toggler-icon"></span>
11             </button>
12             <div class="collapse navbar-collapse" id="navbarSupportedContent">
13                 <ul class="navbar-nav me-auto mb-2 mb-lg-0">
14                     <li class="nav-item">
15                         <a class="nav-link active" aria-current="page" href="#">News</a>
16                     </li>
17                     <li class="nav-item">
18                         <a class="nav-link" href="#">Login</a>
19                     </li>
20                     <li class="nav-item">
21                         <a class="nav-link" href="#">Sign Up</a>
22                     </li>
23                 </ul>
24             </div>
25         </div>
26     </nav>
27
28     <div class="container">
29         {% block content %}
30         {% endblock content %}

```

24. En el archivo **home.html**, agregue el código que permite usar el template **base.html**. `{% extends 'base.html' %}` toma todo el contenido del **home.html** y lo coloca en **base.html**. Ahora, la página principal tiene una barra de navegación.

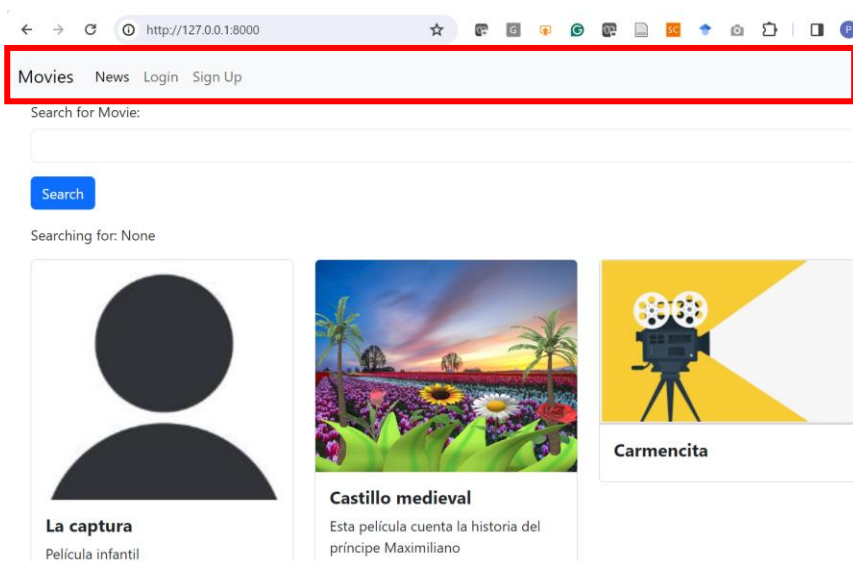
```

1  {% extends 'base.html' %}
2  {% block content %}
3
4  <div class="container"> <!-- mejora el espaciado de la pagina-->
5      <form action=""> <!-- especifica que al hacer clic en enviar, enviamos el formulario a la mis
6          <div class="mb-3"> <!--agrega espacio inferior de 3 unidades-->
7              <label for="searchMovie" class="form-label"> <!--asocia un texto descriptivo con un c
8                  Search for Movie:
9              </label>
10             <input type="text" class="form-control" name="searchMovie"/>
11         </div>
12         <button type="submit" class="btn btn-primary">Search</button>
13     </form>
14     <p>Searching for: {{searchTerm}}</p>
15     <div class="row row-cols-1 row-cols-md-3 g-4"> <!-- define una fila en una cuadrícula con una
16         {% for movie in movies %}
17         <div v-for="movie in movies" class="col">
18             <div class="card">
19                 
20                 <div class="card-body">
21                     <h5 class="card-title fw-bold">{{ movie.title}}</h5>
22                     <p class="card-text">{{ movie.description}}</p>
23                     {% if movie.url %}
24                     <a href="{{ movie.url }}" class="btn btn-primary">Movie Link</a>
25                     {% endif %}
26                 </div>
27             </div>
28         </div>
29     {% endfor %}
30 </div>
31 <br/>
32 <br />
33 <br />
34 <h2>Join our mailing list:</h2>
35 <form action="{% url 'signup' %}">
36     <div class="mb-3">
37         <label for="email" class="form-label"> Enter your email: </label>
38         <input type="email" class="form-control" name="email" />
39     </div>
40     <button type="submit" class="btn btn-primary">Sign Up</button>
41 </form>
42 </div>
43 {% endblock content %}

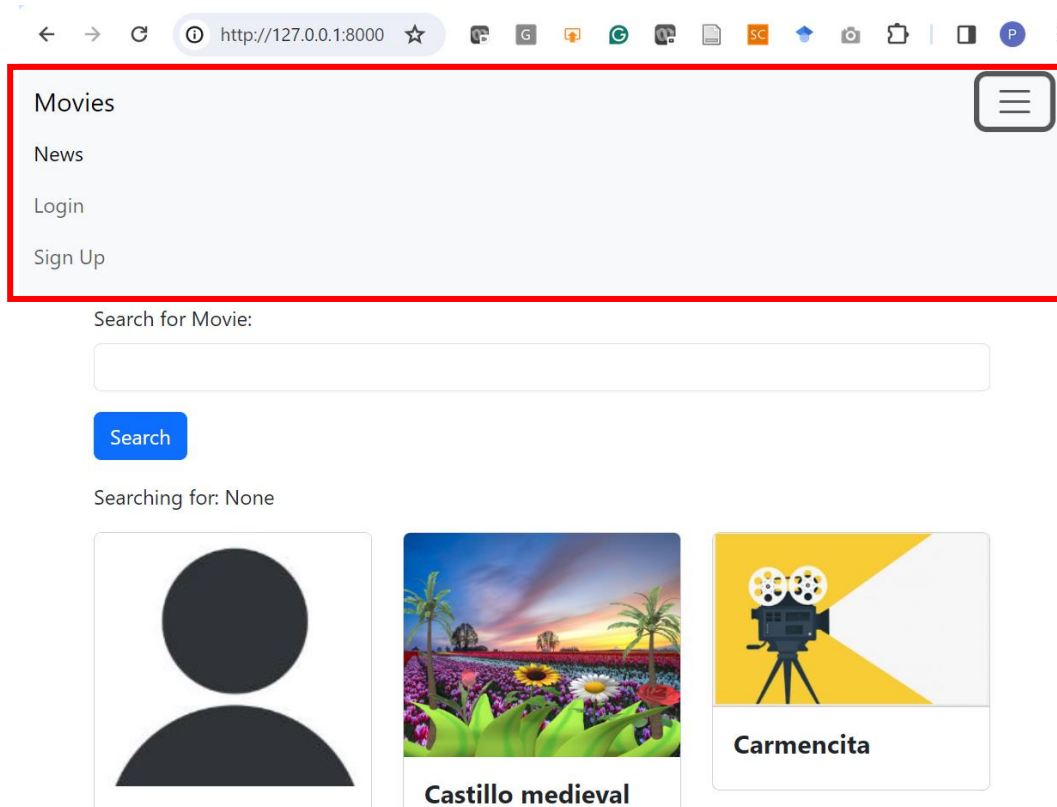
```

4.

Ejecute el servidor y verifique el resultado.



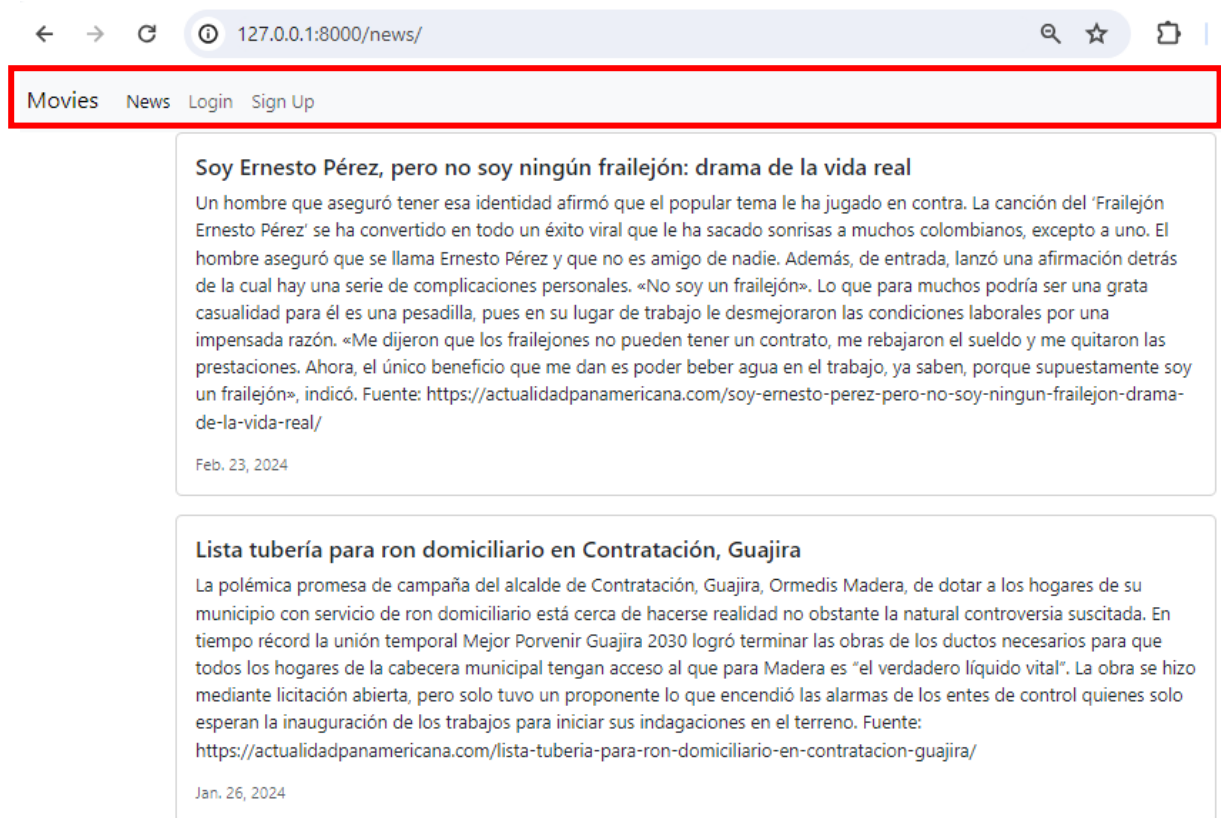
5. Reduzca el tamaño de la pantalla y observe el resultado.



6. Aplique el mismo proceso de extensión de la plantilla base en **news.html**.

```
<> news.html M X
news > templates > <> news.html > ...
1  {% extends 'base.html' %}
2  {% block content %}
3
4  <div class="container">
5      {% for news in newss %}
6      <div class="card mb-3">
7          <div class="row g-0">
8              <div>
9                  <div class="card-body">
10                     <h5 class="card-title">{{ news.headline }}</h5>
11                     <p class="card-text">{{ news.body }}</p>
12                     <p class="card-text"><small class="text-muted"> {{ news.date }} </small></p>
13                 </div>
14             </div>
15         </div>
16     </div>
17     {% endfor %}
18 </div>
19
20 {% endblock content %}
```

7. Ejecute el servidor y verifique el resultado, accediendo a la ruta de news.



8. Para que los enlaces de la barra de navegación funcionen, agregue las rutas correctas al archivo **base.html**. Esto funcionará gracias a que en **urls.py** se definieron las rutas *home* y *news* (ver sección de **urlpatterns**).

```

1  <head>
2  |   <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css" rel="sty
3  |   <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.bundle.min.js" cr
4  </head>
5
6  <nav class="navbar navbar-expand-lg bg-body-tertiary">
7  |   <div class="container-fluid">
8  |       <a class="navbar-brand" href="{% url 'home' %}">Movies</a>
9  |       <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navba
10 |       <span class="navbar-toggler-icon"></span>
11 |   </button>
12 |   <div class="collapse navbar-collapse" id="navbarSupportedContent">
13 |       <ul class="navbar-nav me-auto mb-2 mb-lg-0">
14 |           <li class="nav-item">
15 |               <a class="nav-link active" aria-current="page" href="{% url 'news' %}">News</a>
16 |           </li>
17 |           <li class="nav-item">
18 |               <a class="nav-link" href="#">Login</a>
19 |           </li>
20 |           <li class="nav-item">
21 |               <a class="nav-link" href="#">Sign Up</a>
22 |           </li>
23 |       </ul>
24 |   </div>
25 | </div>
26 </nav>
27
28 <div class="container">
29 |     {% block content %}
30 |     {% endblock content %}
31 </div>

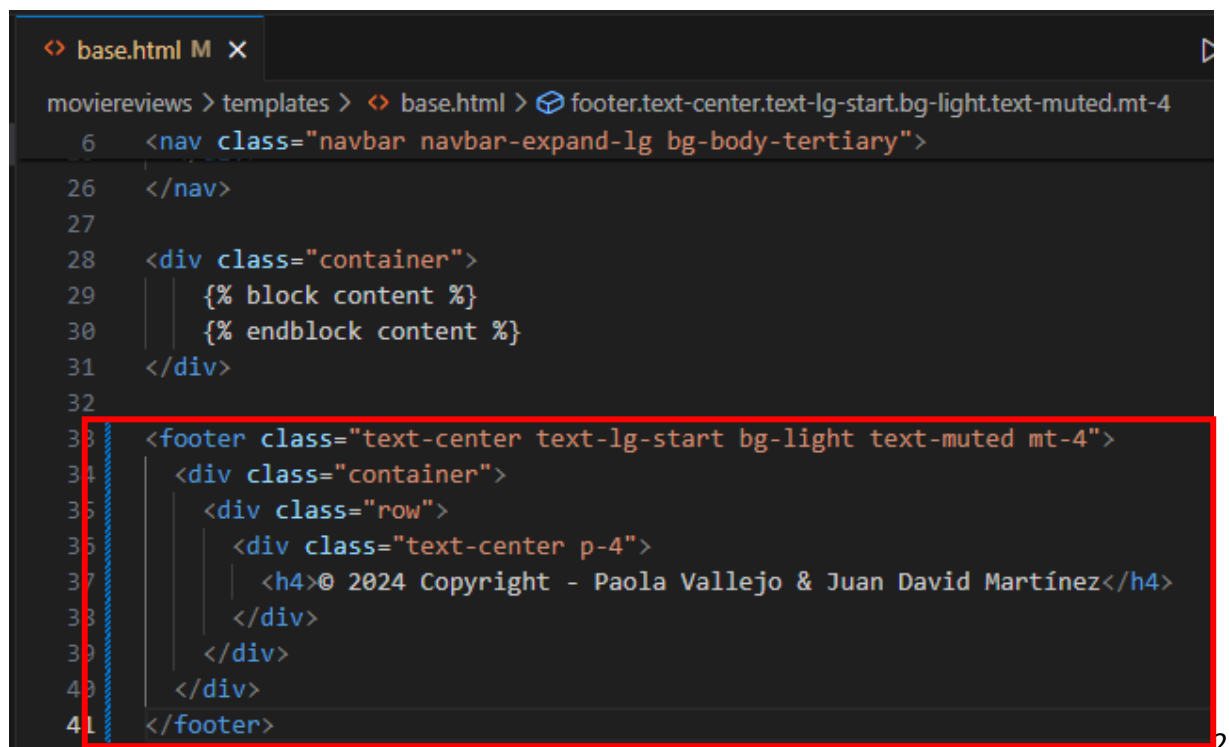
```

9. Ejecute el servidor y verifique que puede usar los ítems del menú de navegación.

Footer

1. Agregue un **footer** en el archivo **base.html**. Puede copiar y pegar el siguiente código.

```
<footer class="text-center text-lg-start bg-light text-muted mt-4">
  <div class="container">
    <div class="row">
      <div class="text-center p-4">
        <h4>© 2025 Copyright - Paola Vallejo & Juan David Martínez</h4>
      </div>
    </div>
  </div>
</footer>
```



```
<> base.html M X
moviereviews > templates > <> base.html > footer.text-center.text-lg-start.bg-light.text-muted.mt-4
6  <nav class="navbar navbar-expand-lg bg-body-tertiary">
26 </nav>
27
28 <div class="container">
29   {% block content %}
30   {% endblock content %}
31 </div>
32
33 <footer class="text-center text-lg-start bg-light text-muted mt-4">
34   <div class="container">
35     <div class="row">
36       <div class="text-center p-4">
37         <h4>© 2024 Copyright - Paola Vallejo & Juan David Martínez</h4>
38       </div>
39     </div>
40   </div>
41 </footer>
```

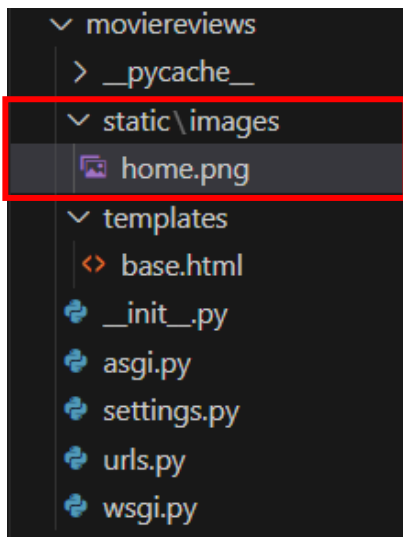
Ejecute el servidor y verifique el resultado en el navegador.

© 2021 Copyright - Paola Vallejo & Juan David Martínez

MANEJO DE ARCHIVOS ESTÁTICOS

Los archivos estáticos son aquellos que se cargan en la página y son fijos, es decir no van a variar.

1. Cree la carpeta **static** en **moviereviews**. Dentro de esa carpeta, cree la carpeta **images**, esta carpeta contendrá las imágenes estáticas que se usarán en el sitio web. En esa carpeta guarde una imagen llamada **home.png**, esta representa el enlace hacia el home de la barra de navegación.



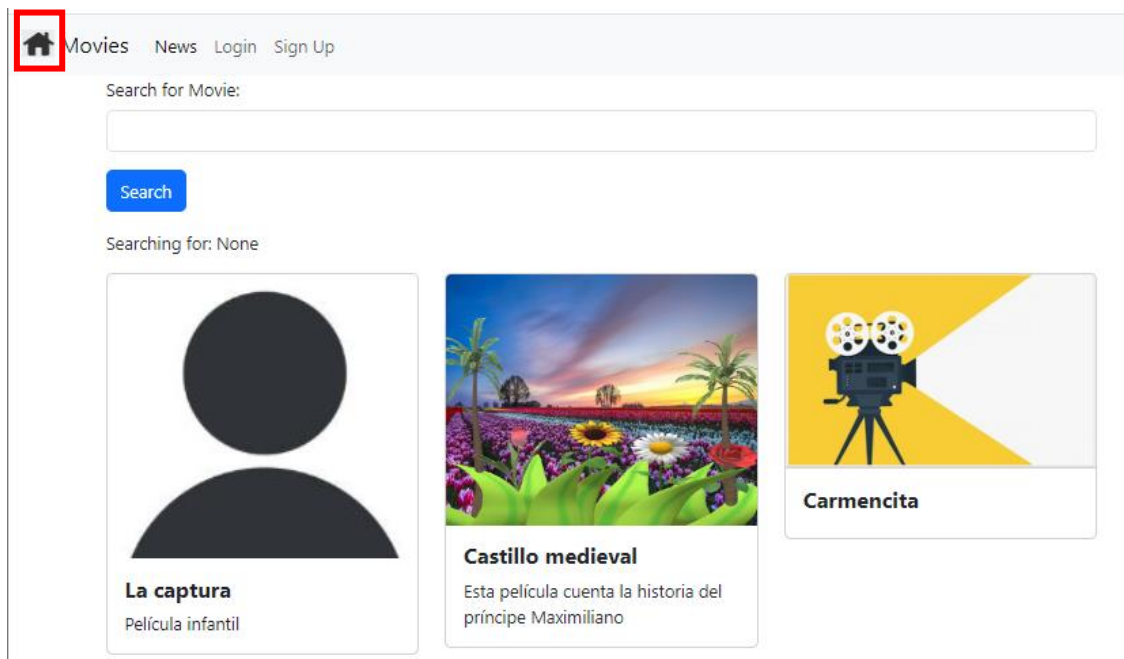
2. Agregue la imagen en la barra de navegación en **base.html**.

```
base.html M X
moviereviews > templates > base.html > div.container
1 <head>
2   <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css" rel="stylesheet">
3   <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.bundle.min.js" rel="script">
4 </head>
5
6 <nav class="navbar navbar-expand-lg bg-body-tertiary">
7   <div class="container-fluid">
8     <a class="navbar-brand" href="{% url 'home' %}">
9       {% load static %}
10      
13      Movies
14    </a>
15    <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarNav"
16      <span class="navbar-toggler-icon"></span>
17    </button>
```

3. En **settings.py** agregue la ruta donde estarán las imágenes estáticas. Agregue la sección **STATICFILES_DIRS** al final del archivo.

```
settings.py M X
moviereviews > settings.py > ...
127 # Default primary key field type
128 # https://docs.djangoproject.com/en/5.0/ref/settings/#default-auto-field
129
130 DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'
131
132 STATICFILES_DIRS = [
133   os.path.join(BASE_DIR, "static"),
134   'moviereviews/static/',
135 ]
```

4. Ejecute el servidor y acceda al navegador para ver el resultado.



GRÁFICA A PARTIR DE LA INFORMACIÓN DE LA BASE DE DATOS

Recopilación de información de la base de datos

1. En el archivo **views.py** de la app movie, agregue la función **statistics_view**. Esta función contará cuantas películas hay por cada año y generará una gráfica de barras con la cantidad de películas por año. No olvide importar todas las librerías necesarias. `matplotlib.use('Agg')` establece el backend de Matplotlib en 'Agg', que es un backend no interactivo que es adecuado para la generación de gráficos sin necesidad de una interfaz gráfica. Puede copiar y pegar el siguiente código.

```
import matplotlib.pyplot as plt
import matplotlib
import io
import urllib, base64
```

```
def statistics_view(request):
    matplotlib.use('Agg')
    years = Movie.objects.values_list('year', flat=True).distinct().order_by('year') # Obtener todos los años de las películas
    movie_counts_by_year = {} # Crear un diccionario para almacenar la cantidad de películas por año
    for year in years: # Contar la cantidad de películas por año
        if year:
            movies_in_year = Movie.objects.filter(year=year)
        else:
            movies_in_year = Movie.objects.filter(year__isnull=True)
            year = "None"
        count = movies_in_year.count()
        movie_counts_by_year[year] = count

    bar_width = 0.5 # Ancho de las barras
    bar_spacing = 0.5 # Separación entre las barras
    bar_positions = range(len(movie_counts_by_year)) # Posiciones de las barras
```

```

# Crear la gráfica de barras
plt.bar(bar_positions, movie_counts_by_year.values(), width=bar_width, align='center')
# Personalizar la gráfica
plt.title('Movies per year')
plt.xlabel('Year')
plt.ylabel('Number of movies')
plt.xticks(bar_positions, movie_counts_by_year.keys(), rotation=90)
# Ajustar el espaciado entre las barras
plt.subplots_adjust(bottom=0.3)
# Guardar la gráfica en un objeto BytesIO
buffer = io.BytesIO()
plt.savefig(buffer, format='png')
buffer.seek(0)
plt.close()

# Convertir la gráfica a base64
image_png = buffer.getvalue()
buffer.close()
graphic = base64.b64encode(image_png)
graphic = graphic.decode('utf-8')

# Renderizar la plantilla statistics.html con la gráfica
return render(request, 'statistics.html', {'graphic': graphic})

```

NOTA: A continuación, encontrará otra versión, más eficiente en cuanto a la cantidad de consultas a la base de datos. Puede experimentar con otras soluciones posteriormente.

```

def statistics_view(request):
    matplotlib.use('Agg')
    # Obtener todas las películas
    all_movies = Movie.objects.all()

    # Crear un diccionario para almacenar la cantidad de películas por año
    movie_counts_by_year = {}

    # Filtrar las películas por año y contar la cantidad de películas por año
    for movie in all_movies:
        year = movie.year if movie.year else "None"
        if year in movie_counts_by_year:
            movie_counts_by_year[year] += 1
        else:
            movie_counts_by_year[year] = 1

    # Ancho de las barras
    bar_width = 0.5
    # Posiciones de las barras
    bar_positions = range(len(movie_counts_by_year))

    # Crear la gráfica de barras
    plt.bar(bar_positions, movie_counts_by_year.values(), width=bar_width, align='center')

    # Personalizar la gráfica
    plt.title('Movies per year')
    plt.xlabel('Year')
    plt.ylabel('Number of movies')
    plt.xticks(bar_positions, movie_counts_by_year.keys(), rotation=90)

    # Ajustar el espaciado entre las barras
    plt.subplots_adjust(bottom=0.3)

    # Guardar la gráfica en un objeto BytesIO
    buffer = io.BytesIO()
    plt.savefig(buffer, format='png')
    buffer.seek(0)
    plt.close()

```

```
# Convertir la gráfica a base64
image_png = buffer.getvalue()
buffer.close()
graphic = base64.b64encode(image_png)
graphic = graphic.decode('utf-8')

# Renderizar la plantilla statistics.html con la gráfica
return render(request, 'statistics.html', {'graphic': graphic})
```

Definición de url

1. En el archivo **urls.py** de **moviereviews**, agregue la ruta para ver la gráfica con la cantidad de películas por año.

```
moviereviews > urls.py > ...
16 """
17 from django.contrib import admin
18 from django.urls import path, include
19 from movie import views as movieViews
20
21 from django.conf.urls.static import static
22 from django.conf import settings
23
24 urlpatterns = [
25     path('admin/', admin.site.urls),
26     path('', movieViews.home, name='home'),
27     path('about/', movieViews.about, name='about'),
28     path('news/', include('news.urls')),
29     path('statistics/', movieViews.statistics_view, name='statistics'),
30 ]
31
32 urlpatterns += static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

Definición de plantilla

1. En la carpeta **movie/templates** cree el archivo **statistics.html**. Edite el código para recibir la gráfica que genera la función **statistics_view** y mostrarla.

```
movie > templates > statistics.html
1 {% extends 'base.html' %}
2 {% block content %}
3
4 <body>
5     <h3>Movies per year chart</h3>
6     
7 </body>
8
9 {% endblock content %}
10
```

2. Edite el navbar de **base.html** para agregar un ítem que permita acceder a Statistics.

```

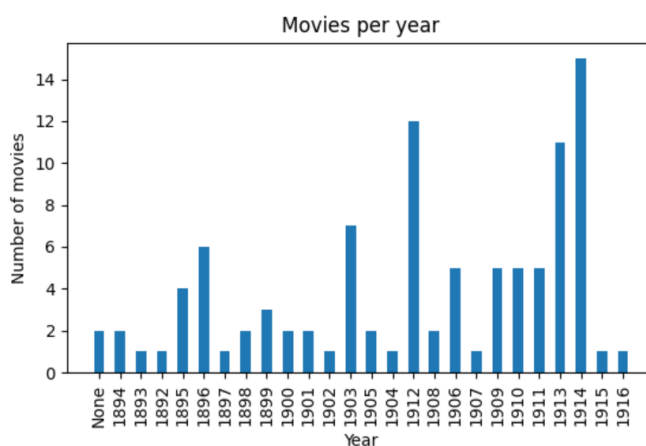
moviereviews > templates > <> base.html
11 height="24"
12 Movies
13 </a>
14 <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarSupportedContent" aria
15 <span class="navbar-toggler-icon"></span>
16 </button>
17 <div class="collapse navbar-collapse" id="navbarSupportedContent">
18 <ul class="navbar-nav me-auto mb-2 mb-lg-0">
19 <li class="nav-item">
20 <a class="nav-link active" aria-current="page" href="{% url 'news' %}">News</a>
21 </li>
22 <li class="nav-item">
23 <a class="nav-link" href="{% url 'statistics' %}">Statistics</a>
24 </li>
25 <li class="nav-item">
26 <a class="nav-link" href="#">Login</a>
27 </li>
28 <li class="nav-item">
29 <a class="nav-link" href="#">Sign Up</a>
30 </li>
31 </ul>
32 </div>
33 </div>
34 </nav>
35
36 <div class="container">
37 {% block content %}
38

```

3. Acceda al navegador y verifique el resultado.



Movies per year chart



4. Agregue los elementos necesarios para generar una gráfica de cantidad de películas por género. Considere únicamente el primer género al que pertenece la película.
5. Acceda al navegador y verifique el resultado.

NAVEGACIÓN ENTRE PÁGINAS

Enviar un formulario a otra página

Actualmente, envía un formulario a la misma página (barra de búsqueda). Ahora enviará un formulario a otra página.

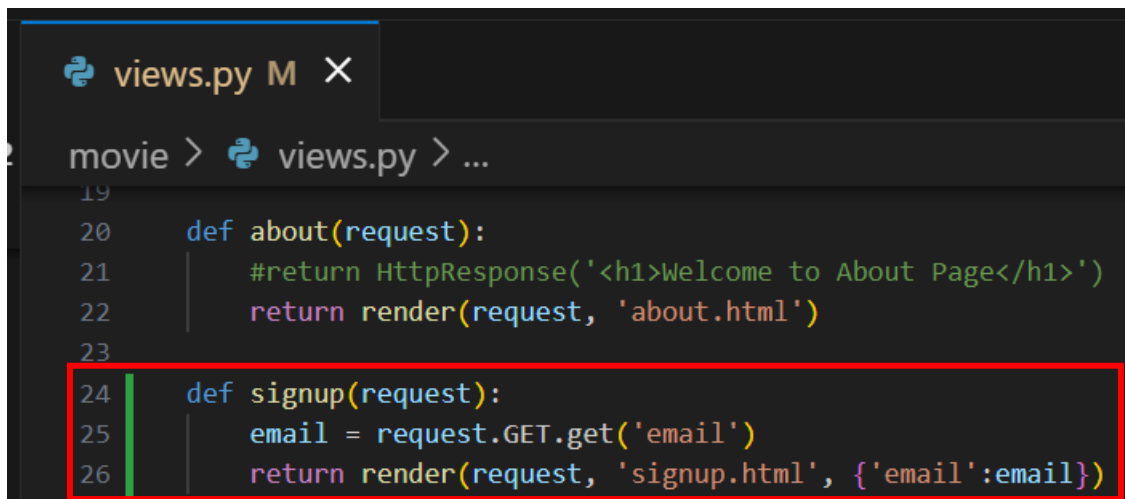
1. En el archivo **home.html**, agregue un formulario para simular la subscripción a una lista de correo. Este formulario es similar al formulario de búsqueda de películas. La diferencia principal está en la acción del formulario, el cual redirecciona a la página de Sign Up. Se toma el patrón *signup* y se retorna una URL.

```
movie > templates > <> home.html
4      <div class="container"> <!-- mejora el espaciado de la pagina-->
15      <div class="row row-cols-1 row-cols-md-3 g-4"> <!-- define una fila en una
17      <div v-for="movie in movies" class="col">
20      </div>
27      </div>
28      </div>
29      {% endfor %}
30      </div>
31      <br/>
32      <br />
33      <br />
34      <h2>Join our mailing list:</h2>
35      <form action="{% url 'signup' %}">
36      <div class="mb-3">
37      <label for="email" class="form-label"> Enter your email: </label>
38      <input type="email" class="form-control" name="email" />
39      </div>
40      <button type="submit" class="btn btn-primary">Sign Up</button>
41      </form>
42      </div>
43
44      {% endblock content %}
```

2. Agregue la ruta **signup** a la lista de patrones en el archivo **urls.py** de **moviereviews**.

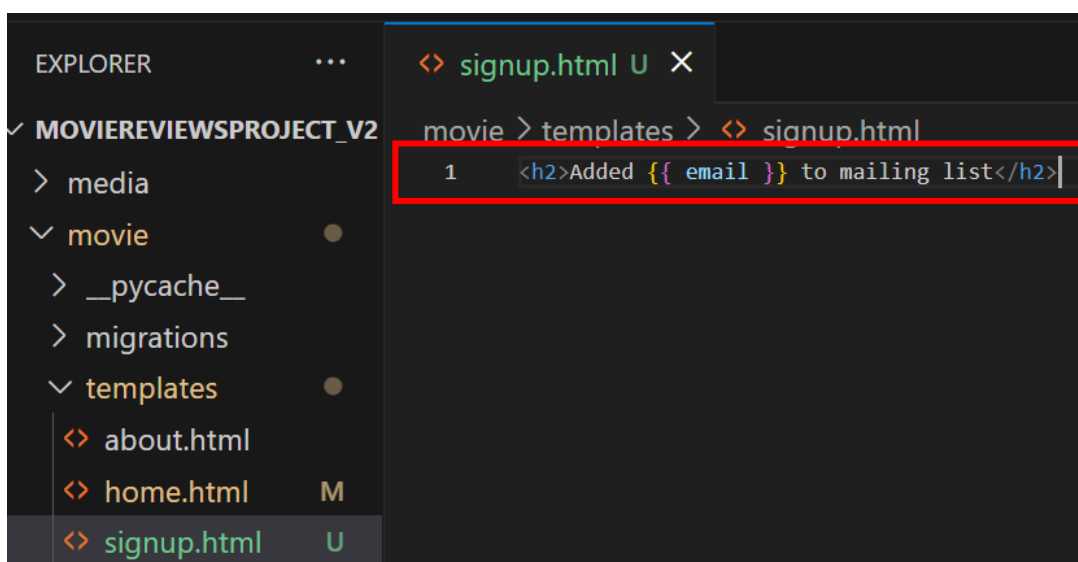
```
urls.py
moviereviews > urls.py > ...
23
24      urlpatterns = [
25      path('admin/', admin.site.urls),
26      path('', movieViews.home, name='home'),
27      path('about/', movieViews.about, name='about'),
28      path('news/', include('news.urls')),
29      path('statistics/', movieViews.statistics view, name='statistics'),
30      path('signup/', movieViews.signup, name='signup'),
31      ]
32
33      urlpatterns += static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

3. Agregue la función **signup** al archivo **views.py**. Se obtiene el email gracias al GET y se envía a **signup.html** pasando el diccionario {'email': email}.



```
views.py M X
movie > views.py > ...
19
20 def about(request):
21     #return HttpResponse('<h1>Welcome to About Page</h1>')
22     return render(request, 'about.html')
23
24 def signup(request):
25     email = request.GET.get('email')
26     return render(request, 'signup.html', {'email':email})
```

4. Agregue el archivo **signup.html** a la carpeta de **templates**.



```
EXPLORER ...
MOVIEREVIEWSPROJECT_V2
> media
> movie
> __pycache__
> migrations
> templates
  <> about.html
  <> home.html M
  <> signup.html U

movie > templates > <> signup.html
1 <h2>Added {{ email }} to mailing list</h2>
```


5. En el navegador acceda al enlace <http://localhost:8000> para visualizar el nuevo formulario. Note que GET pasa los parámetros mediante la URL.

http://127.0.0.1:8000


Search for Movie:

Search

Searching for None



La captura
Película infantil



Castillo medieval
Esta película cuenta la historia del príncipe Maximiliano

Join our mailing list:

Enter your email:

Sign Up

← → ↻ ⓘ http://127.0.0.1:8000/signup/?email=pvallej3%40eafit.edu.co

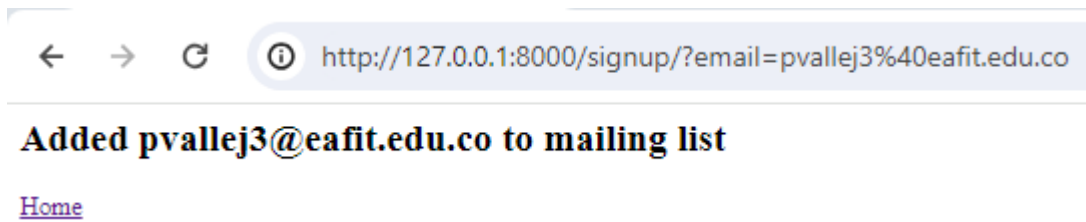
Added pvallej3@eafit.edu.co to mailing list

Crear un enlace para “ir atrás”

1. Cree un enlace para devolverse de **signup** a **home**. Para esto edite el archivo **signup.html** con la etiqueta **<a>**.

```
<> signup.html U X
movie > templates > <> signup.html
1 <h2>Added {{ email }} to mailing list</h2>
2 <a href="{% url 'home' %}">Home</a>
```

2. En el navegador acceda al enlace <http://localhost:8000> para visualizar el nuevo enlace para “ir atrás”.



BIBLIOGRAFÍA

Lim, G., Correa, D. Beginning Django 3 Development. Build Full Stack Python Web Applications. 2021.

<https://github.com/danielgara/bookdjango4.0>

Chacon, S., Straub, B. Pro Git. 2024. <https://git-scm.com/book/en/v2>

Octoverse. 2024. <https://octoverse.github.com/2022/top-programming-languages>

Moure, B. Curso de GIT y GITHUB desde CERO para PRINCIPIANTES.

<https://www.youtube.com/watch?v=3GymExBkKjE>

https://twitter.com/Harsa_Dash/status/1750741820135596036?s=20

<https://blog.hubspot.es/website/como-crear-footer-bootstrap>