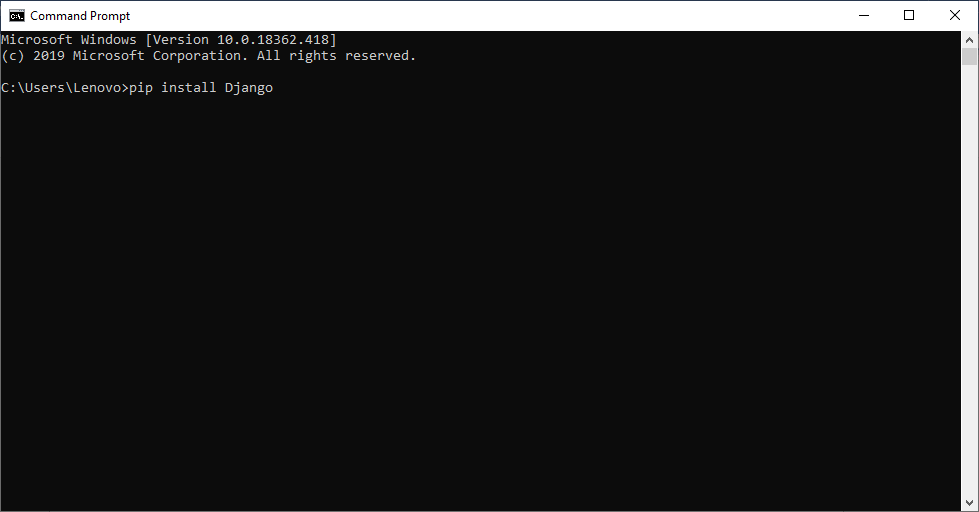
**TUTORIAL DJANGO**

**Instalando Django**

**\*Para la instalación de Django es necesario que tenga instalado Python y pip.**

En la consola de comando (CMD) escribimos el comando:

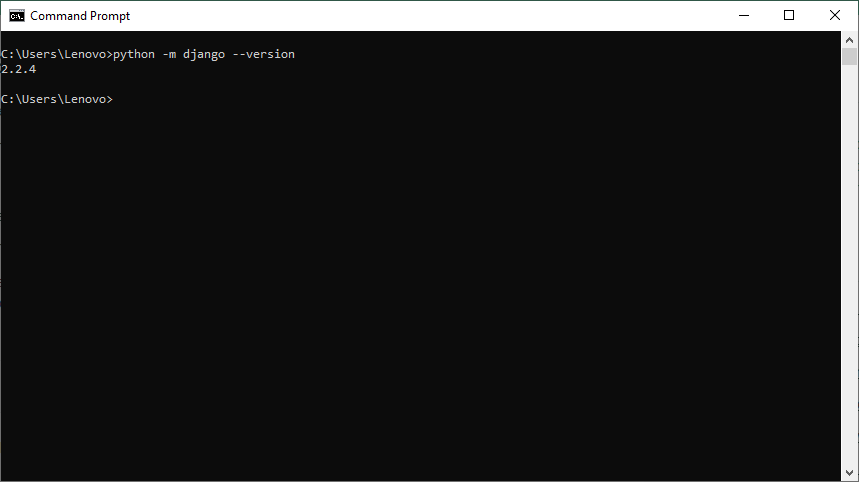
**pip install Django**



Para verificar la instalación correcta de Django, vemos su versión con el comando:

**python –m django –version**

Si está correctamente instalada, se muestra la versión, de caso contrario, saldrá un error de que no existe el módulo Django.



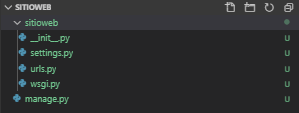
**Creando un proyecto**

Primero nos dirigimos hacia el directorio donde queramos poner nuestro proyecto, una vez ya ubicados, para la creación de un proyecto con Django, en el CMD se escribe el comando:

**django-admin startproject sitioweb**

**\*No poner nombres al proyecto de componentes o módulos reservados por Python o Django, por ejemplo: django, python, test, etc., ya que se puede generar conflictos.**

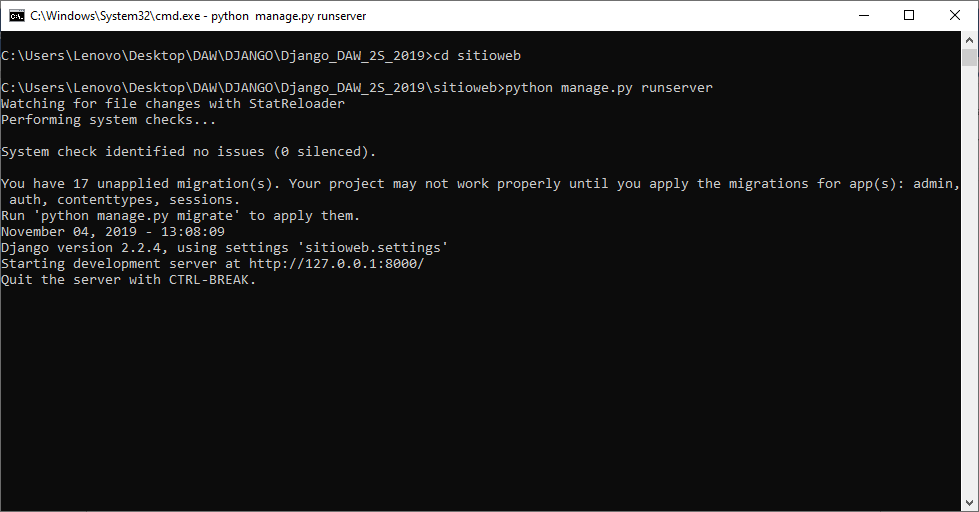
Una vez ejecutado el comando, se nos crea un directorio raíz llamado **sitioweb**, el cual posee un archivo **manage.py** y un subdirectorio llamadao **sitioweb**.

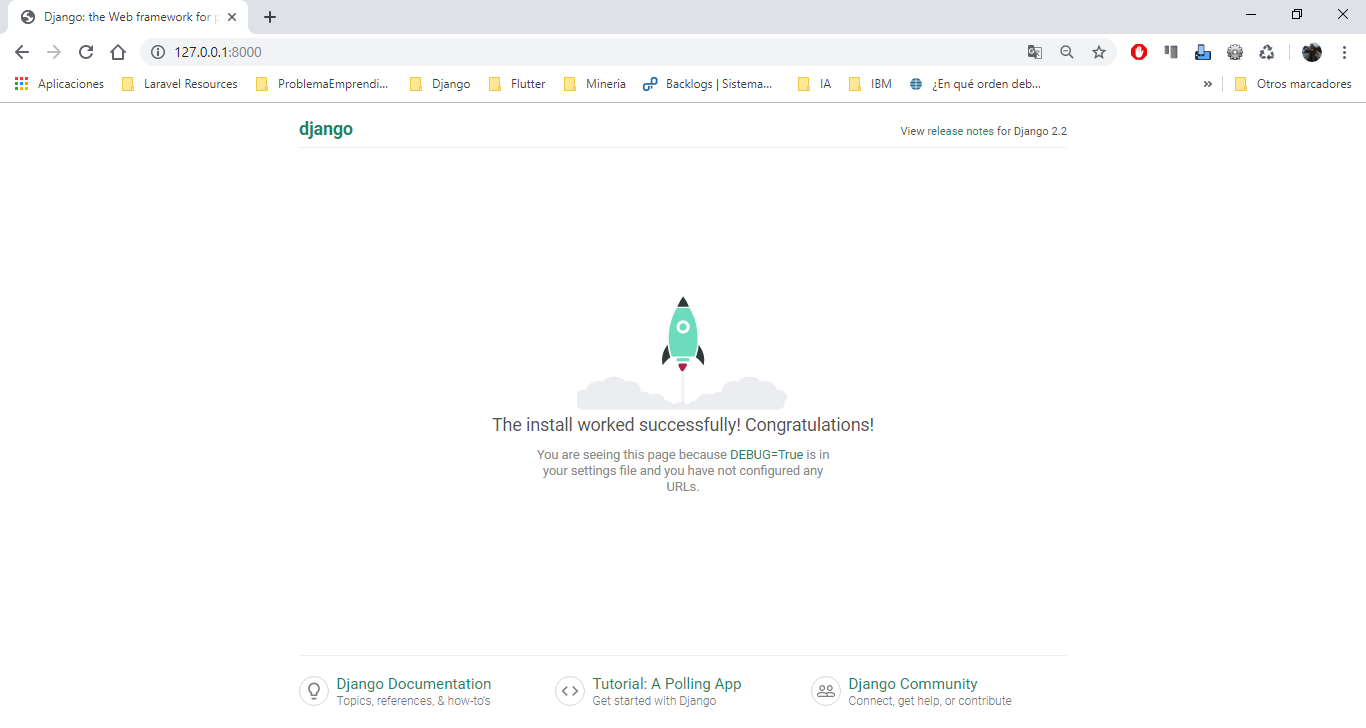


Ahora probaremos a iniciar el servidor de Django de manera local, ubicarse en el directorio raíz **sitioweb**, y escribimos el siguiente comando en el CMD:

**python manager.py runserver**

Nos saldrá una salida como la siguiente, en las que nos indica que la aplicación se encuentra iniciada en **127.0.0.1** con un puerto por defecto, el **8000**. Posteriormente, debemos copiar eso en la barra de búsqueda del navegador y esperemos a ver lo que carga.





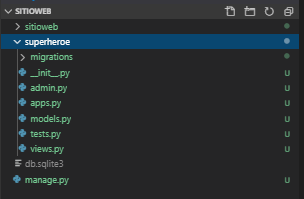
**Creando la aplicación de Superheroes**

Una vez que ya tenemos nuestro proyecto, vamos a crear nuestra primera aplicación de **superheroe**, que consumirá un **servicio Rest**. Para crear la aplicación en el CMD escribimos el comando siguiente:

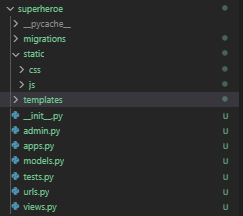
**python manage.py startapp superheroe**

**\* ¿Cuál es la diferencia entre un proyecto y una aplicación? Una app es una aplicación web que hace algo, por ejemplo, un sistema de blog, una base de datos de registros públicos o una aplicación de encuesta simple. Un proyecto es un conjunto de configuraciones y aplicaciones para un sitio web determinado. Un proyecto puede contener aplicaciones múltiples. Una aplicación puede estar en varios proyectos.**

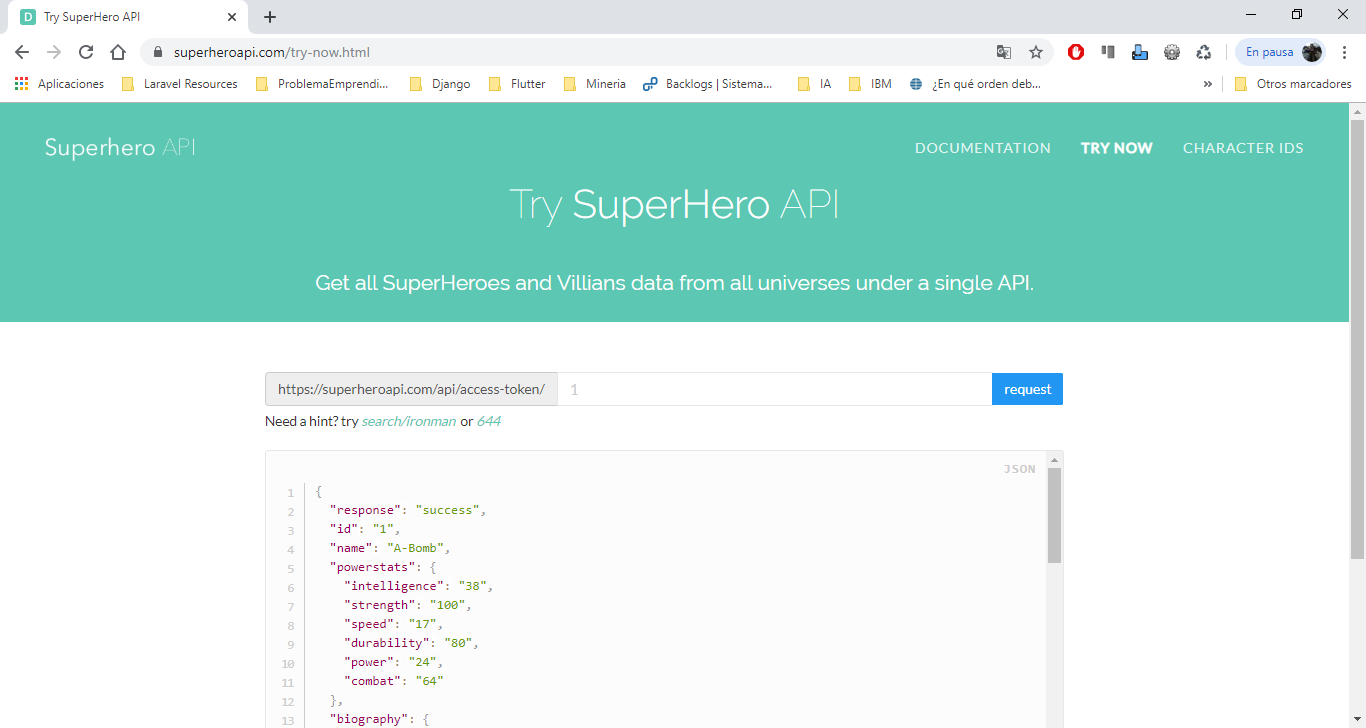
Veremos que se nos crea otro directorio llamado **superheroe** en la raíz **sitioweb**.



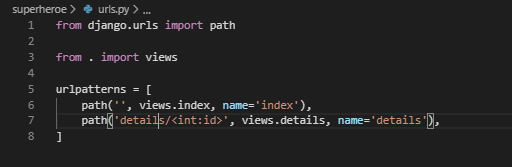
Vamos a crear dos carpetas nuevas dentro de **superheroe**, una llamada **templates**, en la cual estarán los archivos **html** para las vistas; y otra llamada **static**, en la que se ubicaran los archivos estáticos en dos subcarpetas adicionales, una llamada **css**, donde estaran las hojas de estilos **css** y otra llamada **js** donde se encontraran archivos scripts **js**; quedando como la imagen siguiente.



En nuestra aplicación consumiremos un servicio rest que contiene información de varios superheroes, en donde su **endpoint** nos devuelve una respuesta en formato **JSON** con información que cargaremos en nuestra página web.

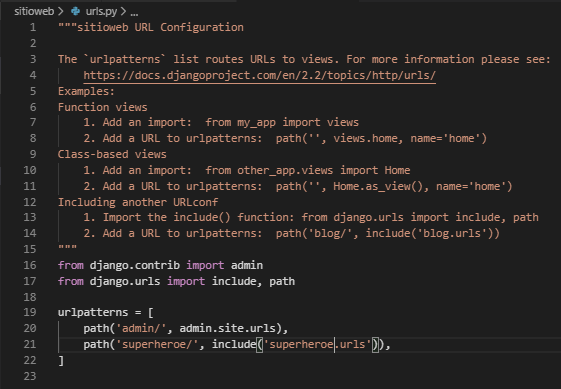


Primero debemos crear nuestra ruta para acceder a la página web, para aquello, en nuestra aplicación **superheroe** creamos un archivo llamado **urls.py**, en el cual tendrá lo siguiente.



Aquí registramos las rutas, la primera **url** será nuestra url raíz, por lo que se pone solo comillas vacías, mientras que la otra url será **details/<int:id>**, luego del slash (**/**) recibe un parámetro, en este caso un **id** de un superhéroe para obtener su información con el servicio rest. **path** está recibiendo 3 parámetros, el primero es la url, el segundo es la función a la cual la url irá para hacer la lógica y el tercer parámetro es un nombre que se le asigna a la ruta para identificar la función de la vista en general.

Como estamos trabajando en la aplicación **superheroe** y nuestro proyecto **sitioweb** puede tener varias aplicaciones, debemos registrar la ruta para cada aplicación creada en el archivo **urls.py** del **sitioweb**.

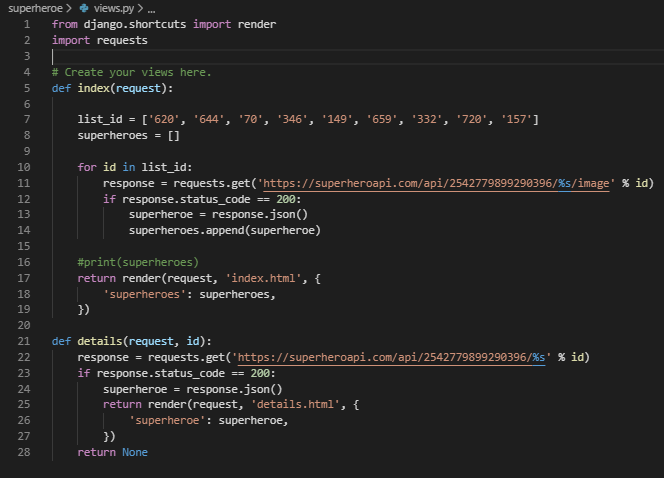


Aquí el **include()** incluye a la url **superheroe/** todas las urls que se ingresaron para esa aplicación, por lo tanto para nuestras urls al final en el navegador serán estas:

**localhost:8000/superheroe/**

**localhost:8000/superheroe/details/10**

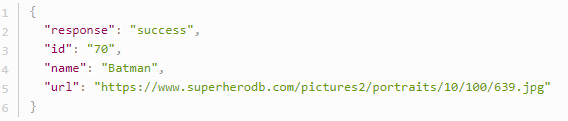
Una vez creado nuestras rutas, debemos ahora crear nuestras funciones en **views** para manejar la lógica y retornar una vista. Primero nos ubicamos en el archivo **views.py** de nuestra aplicación **superheroe** y haremos lo siguiente.



Hay que tener en cuenta que debemos tener instalado el módulo **requests** de Python, en caso de que no se lo tenga, se instala con el siguiente comando:

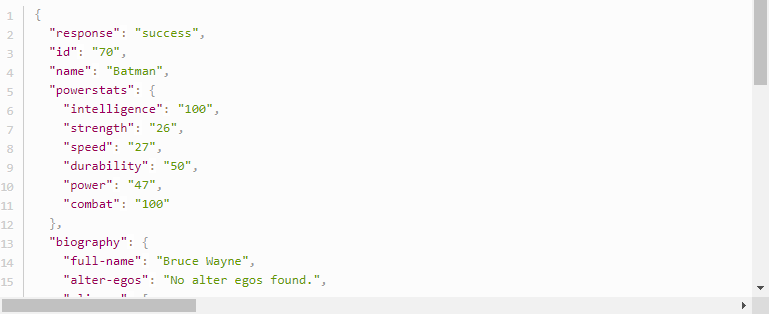
**pip install requests**

Cada función que creamos en **views.py** tiene como primer parámetro un objeto **HttpRequest**, que normalmente de lo denomina **request**. En la función **index** primero tenemos una lista de ids (**list\_id**), luego iteramos esa lista, y en cada iteración hacemos un requerimiento al servicio usando **requests.get()** al **endpoint** [https://superheroapi.com/api/2542779899290396/1/image](https://superheroapi.com/api/2542779899290396/%s/image), y si obtenemos una respuesta válida (**status 200**), la agregamos a la lista **superheroes**. Cada respuesta válida se ve de la siguiente manera en formato **JSON**.

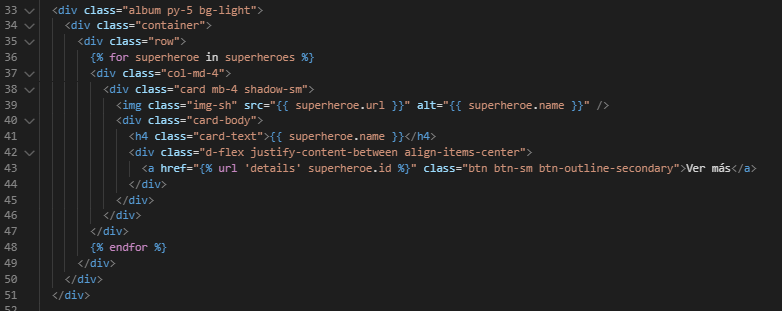


Luego en el **return** hacemos un **render**, el cual nos renderiza nuestra vista **index.html** en el navegador. **render** recibe en este caso, **request** (**objeto HttpRequest**), el nombre de nuestro archivo **html**, y como tercer parámetro un **diccionario**, con clave-valor, donde la clave es **superheroes** y como valor la lista denominada **superheroes**.

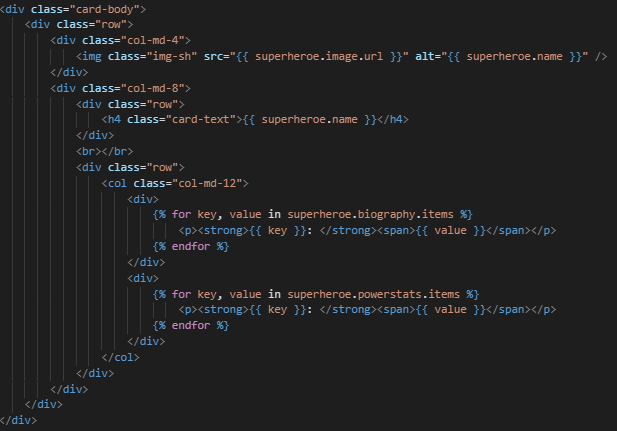
En la functón **details**, a parte del **request**, recibimos un parámetro adicional **id**, porque cuando definimos nuestra **url**, esta recibía aquel id como parámetro, hacemos nuestro requerimiento al **endpoint** https://superheroapi.com/api/2542779899290396/1 y luego dependiendo de que, si obtenemos una respuesta válida, renderizamos ahora nuestro archivo **details.html**, con un **diccionario** que contiene nuestra respuesta **JSON** del servicio.



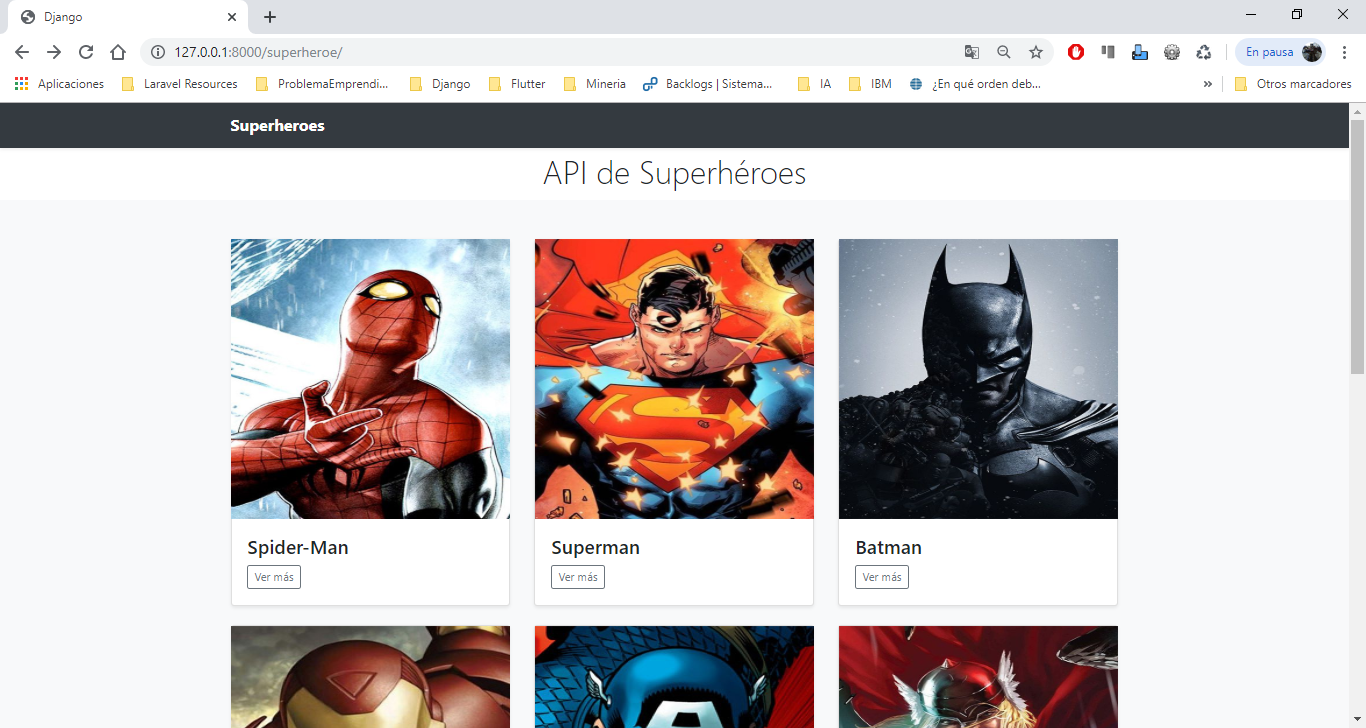
Lo último que nos toca crear es nuestros archivos **html** que son utilizados en el **render** por las funciones en **views**. Para este caso se usaron plantillas de **bootstrap** ya hechas. Por lo tanto, debemos colocar los **.css** y **.js** en nuestra carpeta **static**, y nuestros **html** en nuestra carpeta **templates**. A continuación, se muestra la parte más importante en los **html** que utilizan el **motor de plantillas** propio de **Django**.

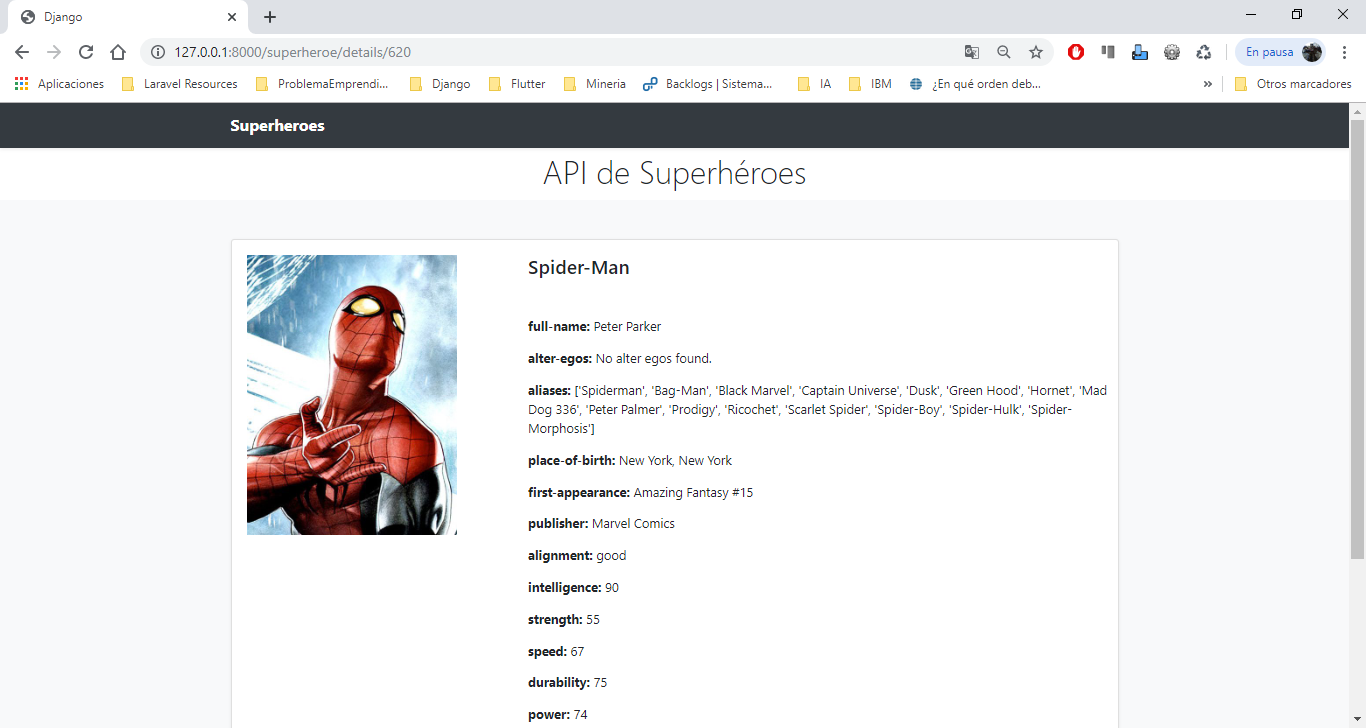


Esta sección es de nuestro archivo **index.html**, como se ve, se está haciendo una iteración **for** de nuestra lista **superheroes** que enviamos en el **render**, y por cada **superheroe**, se van creando el contenedor **div** con una imagen (**img**) y un enlace (**a**), el cual, en su **href**, se pone el nombre del **url details** que creamos, junto con **superheroe.id** que contiene el valor del id del superhéroe para el **servicio rest**. Por lo tanto, al dar clic sobre ese enlace, nos llevaría a la url **/details/id** y nos redireccionaría a esa ruta con otra vista.



En esta vista, ya solo tenemos un superhéroe, el cual viene a hacer un **JSON** y los manejamos como **diccionario**. En **img** colocamos la imagen que nos da el servicio y abajo, recorremos los diccionarios que contienen más información detalla del superhéroe. Al final, en el navegador tendremos las siguientes páginas.





**Repositorio en Git**

<https://github.com/emiliomoran/Django_DAW_2S_2019>