Crear proyecto de Django con:

*django-admin startproject BCDSaliva*

Después para comprobar que todo funciona, si se quiere que la pagina siga corriendo conforme se hacen cambios, se tiene que dejar el servidor corriendo:

*py manage.py runserver*

Una vez funcionando se crea la app en la cual estará contenida la página web:

*py manage.py startapp webpage*

**En el proyecto general**

En settings.py, en INSTALLED\_APPS se agrega la aplicación creada, en este caso “webpage”, de igual forma en TEMPLATES se agrega “DIRS: [BASE\_DIR / “templates”], esto para que se pueda tomar un html de base y de ahí se puedan derivar los demás, de igual forma en la sección de STATIC debe estar habilitada y configurada para los CSS, js, imágenes, etc., por último en se debe agregar MEDIA\_ROOT = BASE\_DIR / “uploads”, el nombre de la carpeta puede variar, esto es para determinar en donde se guardarán los archivos que se suban a la página.

En urls.py, se agregaron los paths para que se despliegue como BCDSaliva tomando en cuenta la app de webpage, y en caso de que el path esté vació lo redirige de igual forma.

**En la carpeta static**

Se ponen los archivos que comparten todas las app, por ejemplo una fuente específica para todas las páginas en el css, etc.

**En la carpeta templates**

Se ponen los archivos de los cuales se pueden derivar otros, por ejemplo, se pone la estructura del html y solo se modifican los bloques en las apps.

**La carpeta de uploads que contiene la carpeta files**

Aquí se guardarán los archivos que se suban a la página de momento.

**Carpeta de la app webpage**

En la carpeta migrations se guardan los modelos de la base de datos que provee Django, la base de datos puede cambiar a mysql sin problemas modificando la conexión en settings.py

La carpeta de static tiene la misma función que la anterior, sin embargo el scope es dentro de la app y no para todas, de igual forma por convención se pone dentro de static una carpeta con el nombre de la app, donde se pueden poner todos los archivos.

Dentro de la carpeta templates, también por convención se pone una carpeta dentro con el nombre de la app, en este caso se ponen las paginas a ser desplegadas, en este caso el index.html

**INDEX.HMTL**

Se extiende de base.html, igual carga la carpeta de static.

Se modifica el bloque de css\_files para poner el css de la app.

Igual se modifica el bloque del título y finalmente el del contenido.

Para la sección de subida de archivos se incluye el csrf token que provee seguridad para el lado del servidor y la form, la cual se genera automáticamente con base en el archivo forms.

Por último se hace un loop para crear un div para cada uno de los miembros del equipo con el contexto que se pasa a traves de views.py

**FORMS.PY**

Se encuentra la form que se utiliza en la página, el cual especifica el tipo de inputs que contiene, en este caso es de archivos.

**MODELS.PY**

Se encuentra la clase que se creó en la base de datos para almacenar los archivos, en la cual especifica a donde mandar los archivos que se suben a la página. Cuando se cambia el modelo de la clase, se tienen que ejecutar los comandos:

*python manage.py makemigrations*

*python manage.py migrate*

Esto para que los cambios se queden guardados y funcionales

**URLS**

La app también puede manejar urls internos i.e. BCDSaliva/hello, BCDSaliva/bye, en este caso no se utilizan de momento.

**Views.py**

Se encuentra una lista que contiene diccionarios con cada una de la información de los miembros, y también la clase SiteView, la cual renderiza el index.html utilizando el modelo UserFile de models.py, y en caso de que la form se suba correctamente vuelve a cargar la página (esto puede ser cambiado), por ultimo pasa a la página a renderizar una lista con el contexto que se necesita, en este caso la lista de miembros y sus datos.