Los principales ficheros que actúan en el sistema y sus tareas son:

1. **WebServer.py**

   Este fichero contiene la información necesaria para arrancar un servidor web, en cual ofrecerá un punto de interacción con el usuario final. Para que las peticiones web sean asíncronas, se utilizará **gunicorn,** un modulo Python que permite crear varios Workers para procesos web entre otras cosas.
   Ofrecerá las siguientes funcionalidades:

   - **Visualizar el mapa de terremotos de una zona:**
     Para solicitar el mapa de terremotos de una zona se accede a la ruta:
     /zone/*nombreCiudad*

     Esto desencadena internamente una petición **REQ** a **intermediario** de la información de los terremotos cerca de esa zona. Y cuando la recibe, WebServer.py crea el mapa, lo guarda en una carpeta, y luego lo manda lo devuelve al navegador para que se visualice.
     {imagen de comunicación entre webServer.py y intermediario.py a través de REQ/REP mostrando que se pide y que devuelve}
   - **Mostrar listado de mapas de terremotos guardados:**
     Para solicitar el listado de mapas de terremotos guardados en el sistema solicitamos via:
     /getSaveds

     Esto desencadena una petición **REQ** a **intermediario** para que liste los ficheros guardados en Dropbox con la información de los terremotos, y se los mande de vuelta a WebServer.py para que los devuelva al navegador web.
     {imagen de comunicación entre webServer.py y intermediario.py a través de REQ/REP mostrando que se pide y que devuelve}
   - **Buscar una ciudad del mundo por su nombre:**
     Para solicitar un listado de ciudades, según un nombre introducido, lo solicitamos a la dirección:
     /search/nombreCiudad

     Esta solicitud web, realiza la tarea de usar la web de Earthquaketrack.com, para scrapear y devolver un conjunto de ciudades que coincidan con el nombreCiudad introducido.

   - **Solicitar la generación del mapa:**
     Para solicitar al sistema que guarde los datos de terremotos de una ciudad, se realiza mediante la dirección:
     /getzone/nombreCiudad/numeroPaginas

     Esta solicitud web, genera una petición **PUSH** a **StartClients.py**, en la que se indica el nombre de la ciudad de la que tenemos que scrapear toda la info de terremotos en Earthquaketrack.com. Se devuelve al navegador web un mensaje indicando que el mapa estará disponible en breve

{imagen de comunicación entre webServer.py y StartClients.py a través de PUSH/PULL mostrando que info se manda}

- **Acceso a página principal:**
  Para acceder a la página principal de la aplicación:
  /
  Se devuelve el mapa de terremotos de la zona por defecto y los botones con las opciones disponibles.

2. **Intermediario.py**
   Este script se conecta con **DropBox y WebServer.py** y es el encargado de prestar servicio a peticiones de:
   - **Listar ciudades(REP):**
     Devuelve un listado con el nombre de las ciudades de las que hay datos guardados en Dropbox
   - **Devolver datos de terremotos de una ciudad(REP):**
     Devuelve un json leído de Dropbox con los datos de los terremotos de la

   {también se podría meter alguna foto de la comunicación si se ve necesario…}

3. **StartClients.py y Client.py**
   Este script recibe peticiones por un socket **PULL**. En las peticiones se recibe el nombreCiudad y el numero de páginas entre otra información, para instanciar objetos de la Clase Client con las opciones recibida.

   Cada objeto de la clase Client se comunica mediante **REQ** con **Server.py**, le manda el nombreCiudad y numeroPaginas y espera la respuesta de tarea finalizada.

4. **Server.py**
   Script que recibe mediante **ROUTER** una mensaje, y se lo manda por **ROUTER** al primer worker que vea disponible, cuando el Worker termina, manda de vuelta un mensaje de tarea finalizada ,que es mandado al Client que solicito la tarea.
   Este servidor puede aceptar tantos clientes como quiera y creará un pool de Workers, de tal forma que le manda al primer worker disponible la tarea, si un worker se desconecta se elimina del pool, si se agrega otro worker mientras está funcionando se agrega al pool automáticamente.

5. **Worker.py y Spider.py**
   Los Worker.py esperan peticiones mediante **REP** de **Server.py,** devolviendo un mensaje de tarea finalizada al terminar su cometido.
   Reciben la ciudad y el número de paginas de las que hay que obtener datos de terremotos.
   Internamente, ejecuta un comando llamando a Spider.py con los parámetros de nombreCiudad y numeroPaginas, para realizar scraping de earthquaketrack.com.
   Una vez realizado el scraping, el worker guarda la información en **DropBox** y manda mensaje de tarea finalizada.
   {foto de worker haciendo las cosa de los demonio}