La parte de gestión de usuarios ahora mismo la tengo montada de tal forma que en SQL todos los tipos de usuario tienen acceso a todo. Después en la aplicación en Python se hacen las restricciones. Saco de la base de datos el tipo de permisos del usuario y en función de eso oculto las acciones a las que no tiene permiso y en caso de que aun así acceda a alguna cree una función de filtro que se llama antes de cada otra función que bloquea el acceso a una función si no se tiene el permiso mandando un mensaje de error por pantalla y redireccionando la uri a la página principal.

En cuanto al alcance del proyecto las partes en las que lo dividiría serian:

\* **Gestión de usuarios**:

Todo esto está hecho y en términos generales funciona. Aun que me gustaría hacer JUnits para comprobar todos los casos específicos

- **Registro**: Genera una solicitud de admisión y crea el usuario en la base de datos (es obligatorio hacerlo aquí porque para crear el usuario se necesita la contraseña, en el momento en que se acepta la admisión la contraseña no está disponible en ningún sitio)

- **Admisión**: Un usuario administrador acepta (o rechaza) las solicitudes de admisión.

^ Si se acepta la solicitud se crea una copia de la base de datos plantilla “BD\_” utilizando como nombre de la base de datos el nombre de usuario (que es único) “BD\_NombreUsuario” y se habilita el usuario que se había creado en el registro

^ Si se rechaza la solicitud se elimina el usuario que se había creado en el registro

- **Eliminar usuarios:** Elimina el usuario y su base de datos asociada. Un usuario solo se puede eliminar a si mismo y si es administrador puede eliminar también a usuarios con permisos de menor rango

- **Editar permisos**: Modifica los permisos de los usuarios. De momento hay definidos tres rangos, pero lo estaba implementando de modo que sea procedural y sea fácil añadir rangos nuevos. Solo se puede editar los permisos de menor rango que el usuario. Por ejemplo, los usuarios de rango 1 no pueden hacer nada. Los de rango 2 pueden gestionar a los usuarios de rango 1 y los usuarios de rango 3 pueden gestionar a los de rango 1 y 2.

\* **Gestión de datos**:

- **Manual**: Formularios para que cada usuario pueda rellenar su base de datos línea a línea. Esto es en lo que estoy trabajando ahora

- **Fichero**: Es un único formulario que permite a un usuario rellenar su base de datos a granel desde una hoja Excel (Fichero CSV) ordenando los datos en filas y columnas conforme a una plantilla que podrá descargarse desde esta misma página. Esto este hecho y funciona. En vez de JUnits he creado un fichero con todos los casos de datos correctos e incorrectos que se me han ocurrido. Actualmente si una línea tiene un dato con un formato erróneo y no se puede introducir a la base de datos, dicha fila se separa y al finalizar la ejecución de todas las filas las que han fallado se imprimen en un mensaje por pantalla indicado que filas no se han podido introducir y porque motivo (Se diferencian motivos de forma bastante amplia. Por ejemplo, si la fecha no es correcta, si ese dato ya existe en la base de datos, si se esperaba un número y se ha introducido otro tipo de dato. Una funcionalidad adicional que quería añadir, pero la dejare en trabajo futuro, era que en vez de un mensaje por pantalla se añadiesen todas las líneas que han fallado a un fichero CSV que se descargara automáticamente al finalizar la ejecución de todas las filas. De este modo, no es necesario ir a buscar cada fila que ha fallado en el fichero original. Solo hay que corregir los fallos del fichero descargado y subirlo. Mucho más cómodo

\* **Copias de seguridad**:

De esto no tengo nada hecho:

Había pensado en hacer copias de seguridad automáticas en otro servidor, pero algo así se me va a ir de las manos en cuanto a complejidad. Así que había pensado mejor guardar todos los datos de la base de datos del usuario en un fichero XML que se pudiera descargar de forma manual. De tal modo que este mismo fichero se pudiera volver a subir de la misma forma que funciona el CSV del apartado anterior.

Al ser un XML que yo mismo decido su estructura seria más sencillo de implementar que el CSV en el cual tenia que tener en cuenta todas las posibles barbaridades que escribiera el usuario en el fichero (Si me mete símbolos en los números, si usa puntos o comas para separa decimales, si usa puntos o comas para separar miles, el formato con el que escribe la fecha)

No obstante, este sistema de copias de seguridad es super vulnerable porque no puedo obligar al usuario a hacer copias. Si el usuario decide que no vale la pena descargar este fichero periódicamente podría perder todos sus datos.

\* **Interpretación de datos**:

De esto he estado mirando cosas y de la primera versión tengo algo implementado que funcionaba a medias.

Consiste en buscar fuentes de donde pueda obtener datos financieros y hacer las distintas representaciones de la información de los datos del usuario y los datos externos.

Esta es la parte que se supone que es la clave del proyecto. La única que debería ser imprescindible.

\* **Subirlo a un servidor:**

De esto no tengo nada mirado

Pero es una pagina web. Lo suyo es subirla a un servidor web.

Me gustaría que fuera una página correcta, con su correspondiente certificado etc.

\* **Pruebas de campo:**

Consiste en que cuando la aplicación esté terminada pasársela a un conjunto de personas junto con un test con preguntas acerca de su funcionamiento para sacar conclusiones en cuanto a aspectos de mejora y ver que partes son más o menos intuitivas.

Los aspectos de mejora que se saquen se dejan como trabajo futuro y listo.