1. **Confección de la base de datos.**

Para la actividad final se ha realizado una aplicación que permite gestionar a diversos doctores los diferentes pacientes que atiende y manipular sus historias clínicas. A cada doctor le será asignado un identificador único y se conocerá su nombre, apellido, contraseña para acceder a la gestión de sus pacientes, teléfono, correo electrónico y si poseen derechos de administrador para agregar doctores nuevos a la base de datos.

Se ha seleccionado una sola especialidad para los doctores, en este caso, los médicos de familia o doctores de cabecera. La base de datos contendrá diversos pacientes que van a sus consultas para ser atendidos y de cada uno se almacenará, nombre, apellidos, teléfono, email y fecha de nacimiento. Cada paciente, a su vez, será identificado por una clave que será única. En resumen, un paciente tendrá asignado un único doctor (que luego de consultarlo, decidirá si es necesario que sea visto por un especialista) y cada doctor podrá tener muchos pacientes (relación 1:N).

Por otro lado, cada paciente puede tener diversas historias clínicas resultado de varias consultas que haya tenido con su médico de cabecera, sin embargo, una historia clínica será única para un paciente (relación 1:N). En dichas históricas clínicas se almacenará el motivo de consulta, la fecha de consulta, historia familiar, alergias, diagnóstico, tratamiento y evolución clínica de cada paciente. Además, cada historia será identificada a través de una clave única.

1. **Desarrollo del proyecto.**

Para desarrollar el proyecto se utilizaron las diversas funcionalidades de la librería PyQt6, que permite la manipulación de elementos para crear la interfaz gráfica. La base de datos fue realizada en sqlite a través de QSqlDataBase y las sentencias SQL fueron ejecutadas con QSqlQuery. Ambas pertenecen al módulo QtSql de la librería antes mencionada.

Para la ejecución del proyecto, se entra a través del script hospital\_main.py, que se encarga de llamar a la función main(), que a su vez, contiene el código para las conexiones con la base de datos y la ejecución de la aplicación.

* 1. Secuencia de la aplicación.

A la aplicación se entra a través de la ventana de autenticación. Esta permite a cada doctor introducir sus credenciales (nombre y contraseña en este caso) para acceder a sus pacientes. Luego, aparece la interfaz principal que contiene la tabla con la información de los pacientes del doctor autenticado.

A su vez, a la derecha existen diversos botones para interactuar con la base de datos. Cada uno lleva a una nueva ventana de dialogo donde se podrán hacer varias acciones, como crear nuevos pacientes, consultar historias clínicas del paciente seleccionado, eliminar un paciente seleccionado, cambiar contraseña de doctor, eliminar todos los pacientes, gestionar los doctores existentes en la base de datos (solo con permisos de administrador) y cerrar sesión del médico actual. También, se ha implementado el botón *consulta general* que permite consultar todos los datos de los pacientes del doctor que tengan historia clínica (incluidos los campos de su historia y el nombre de su doctor) y guardar la tabla resultante con formato CSV o filtrar un paciente en específico.

En todas las tablas que pueden consultarse en el proyecto, se ha implementado que sea posible modificar los valores ya introducidos a través del doble click del mouse. A su vez, no es posible modificar el campo ID de cada entidad. Esto se ha logrado gracias a la modificación de banderas en el módulo QSqlTableModel, que es el utilizado para el trabajo con QSqlTableView.

1. **Operaciones con la base de datos.**

Se creó el fichero database.py que contiene el código para la creación de las tablas de la base de datos, la activación del uso de claves foráneas y el método para conectarse a la misma.

* 1. Acciones con la base de datos.

Autentication.py: Se accede a la tabla DOCTORES de la base de datos y se filtra por nombre y contraseña para ser comparados con los campos de la interfaz gráfica y determinar si las credenciales son correctas.

View.py: Se accede a la tabla PACIENTES de la base de datos y se filtran todos los valores que posean como clave foránea el número de identificación del doctor autenticado.

Agregar\_pacientes.py: Ventana de dialogo que accede a la tabla PACIENTES y en dependencia de lo que se ha introducido en los campos de la UI, se utiliza el comando INSERT INTO para agregarlo a la base de datos.

Filtrar\_pacientes.py: En dependencia de los valores de búsqueda introducidos por el usuario, se modifica la sentencia SQL para filtrar en una tabla general que contiene información de las 3 entidades de la base de datos.

Consultar\_historia\_clinica.py: Se consulta la tabla HISTORIAS\_CLINICAS y se filtra por la identificación del paciente que ha sido seleccionado. Estos valores se muestran en una tabla en la UI.

Crear\_historia\_clinica.py: Esta ventana de dialogo se abre al presionar uno de los botones de la ventana de dialogo que se menciona arriba. Posee varios campos para insertar valores que serán añadidos a la tabla HISTORIAS\_CLINICAS.

Tabla\_doctores.py y agregar\_doctores.py: En el primero se consulta la tabla DOCTORES y se muestra todo su contenido (solo pueden acceder doctores con el campo administradores en True). El segundo permite instertar en la tabla un nuevo doctor con todos (se utiliza sentencia INSERT INTO de SQL).

Cambiar\_password.py: Ventana que permite introducir una contraseña nueva y se hace una modificación (con UPDATE) en la base de datos. En este caso para modificar contraseña.

Consulta\_general.py: Se realiza la unión de las 3 tablas de la base de datos a través de las relaciones establecidas y se muestran datos generales de los pacientes. En este caso se filtran los que tengan una historia clínica asociada a su numero de ID de paciente y se muestra el nombre de su doctor.

Además, cada una de las ventanas que poseen una tabla en su interfaz gráfica, poseen un botón para eliminar una fila. En cada caso se filtra por el ID de la fila seleccionada y se elimina en esa posición de la base de datos. Se utiliza el comando DELETE de SQL.

Se añadió la sentencia ON CASCADE para las claves foráneas en cada caso. Si se elimina un doctor, se eliminan sus pacientes, lo mismo si se borra un paciente con sus historias clínicas.