Presentación

(HealthTrackR)

Diapositiva 1/38

Buenas días, soy **Javier Jordán Luque** y les voy a presentar mi Trabajo de Fin de Grado. El trabajo se titula "**Aplicación móvil para la monitorización de tratamientos sanitarios**" y ha sido tutorizado por **Francisco José Jaime Rodríguez**.

Diapositiva 2/38

La **motivación** de este trabajo surge del **desafío que enfrentan muchas personas para seguir adecuadamente los tratamientos médicos, especialmente quienes padecen enfermedades crónicas**.

Esto genera un **impacto** significativo, provocando **errores en la administración de medicamentos, confusiones y preocupaciones tanto para los pacientes como para sus cuidadores**, lo cual **afecta negativamente la efectividad del cuidado médico**.

Por tanto, existe la **necesidad** de **simplificar el seguimiento de tratamientos médicos proporcionando información clara y accesible para los pacientes**.

La **solución** que ha surgido de todo esto es una **aplicación móvil integral llamada HealthTrackR**, la cual explicaré en detalle más adelante.

Diapositiva 3/38

Para el desarrollo de la aplicación, HealthTrackR, se ha llevado a cabo un proceso de desarrollo software completo, siguiendo una **metodología ágil basada en Scrum**. Esta metodología implicaba la realización de **sprints cada dos semanas** durante los cuales se **evaluaba** y **validaba** el trabajo realizado y se planificaba el siguiente sprint. Este enfoque promovía una **entrega incremental de resultados** y permitía **adaptarse a los cambios** de manera efectiva.

Diapositiva 4/38

Para la planificación del trabajo, se ha hecho uso de la herramienta **Trello**, que proporciona un tablero donde se detallan las tareas a realizar en cada sprint. En total el trabajo, ha consistido en **10 sprints** aunque los primeros tuvieron una duración considerablemente mayor debido a la fase inicial de definición de requisitos y diseño del sistema, que son procesos que requieren más tiempo.

Diapositiva 5/38

Aquí pueden ver el **octavo sprint** como ejemplo. Como se puede observar, las tareas se organizan en columnas según su estado. Cada tarea está etiquetada para mantener una organización según su naturaleza, y también se clasifican por colores que indicaban su prioridad: las tareas de color verde eran las menos prioritarias, mientras que las de color rojo eran las más importantes.

Diapositiva 6/38

Como he comentado anteriormente, se ha llevado a cabo un **proceso de desarrollo de software** completo, abordando todas las fases del ciclo de vida del desarrollo de software, comenzando por el **análisis y especificación de**

requisitos, seguido por el **modelado y diseño del sistema**, la **implementación**, las **pruebas** y finalmente el despliegue.

Diapositiva 7/38

Antes de nada, vamos a repasar las tecnologías y herramientas utilizadas, que se muestran en esta diapositiva.

La principal tecnología ha sido **Android**, ya que la aplicación está diseñada exclusivamente para estos dispositivos. La base de datos empleada es **SQLite** y el principal lenguaje de programación es **Java**.

En cuanto a las **herramientas**, se ha utilizado principalmente **Visual Paradigm** para la creación de los diagramas UML, y **Pencil Project** para el diseño de la interfaz de usuario.

Diapositiva 8/38

Vamos a comenzar con el **análisis y especificación de requisitos**. La aplicación, lleva a cabo la **administración de cuentas** y el seguimiento de **tratamientos** a través de **medicamentos**, **pautas** (a las que se le pueden añadir imágenes y vídeos explicativos), **síntomas**, **preguntas para el médico** y **citas médicas**. Además, la aplicación programa **notificaciones** asociadas a los medicamentos y citas médicas para mantener al usuario informado sobre el seguimiento de su tratamiento.

Los requisitos están organizados en tres categorías: **requisitos funcionales**, **no funcionales** y **opcionales,** y todos cuentan con un identificador único.

Diapositiva 9/38

Este requisito funcional que se muestra es el **requisito funcional 24**. Todos los requisitos tienen un título y una descripción que detalla brevemente el requisito en cuestión.

Diapositiva 10/38

Este requisito no funcional establece que el **sistema debe ser seguro.** Los requisitos anidados detallan específicamente los aspectos de esa seguridad, mientras que en los requisitos funcionales se especifica los requisitos que se deben implementar para cumplir con el requisito no funcional.

Diapositiva 11/38

Por último, tenemos los **requisitos opcionales** que son simplemente **requisitos funcionales y no funcionales a realizar en un futuro para aumentar las prestaciones de la aplicación**.

Diapositiva 12/38

Seguimos con el **modelado y diseño del sistema**, que ha consistido en la elaboración de los **casos de uso**, el **modelado estructural**, el de **comportamiento** y el **diseño de la interfaz de usuario**.

Diapositiva 13/38

En cuanto a los **casos de uso**, estos tienen el **identificador único,** al igual que sucedía con los requisitos. Se ha elaborado el diagrama de casos de uso y el documento asociado.

Diapositiva 14/38

El **diagrama de casos de uso** tiene al **usuario** como **único actor**. En este diagrama se han empleado únicamente la **asociación** y la **extensión**.

Diapositiva 15/38

Aquí puede observarse cómo el usuario está asociado varios casos de uso, entre los cuales se incluye "**consultar la información de la cuenta**", que a su vez permite ejecutar otros casos de uso debido a la relación de extensión.

Diapositiva 16/38

Todos los casos de uso del diagrama anterior están descritos en un **documento**. Cada caso de uso tiene una **precondición** que debe cumplir para poder ejecutarse y una **postcondición** que se debe garantizar tras su ejecución. Además, existe un **escenario principal** o de éxito y varios **escenarios alternativos**.

Diapositiva 17/38

En esta imagen se muestra como ejemplo el **caso de uso 1**, "**Registrarse**". Como se puede observar, incluye la **precondición**, la **postcondición**, el **escenario principal** y varios **escenarios alternativos**.

Diapositiva 18/38

Para definir el **modelado estructural** se ha llevado a cabo el **diagrama de clases que muestra la estructura estática del sistema**. En esta imagen se pueden ver sus dimensiones. Vamos a centrarnos en algunas partes del diagrama.

Diapositiva 19/38

Aquí se puede ver cómo está gestionada la interacción con la base de datos. Tenemos la clase abstracta `BaseRepository`, que define los métodos principales acceso a la base de datos. A su alrededor, están todos los repositorios que heredan de él y corresponden a las tablas de la base de datos. `BaseRepository` está también asociada con la clase `DatabaseHelper`, la cual se encarga de inicializar y configurar la base de datos.

Diapositiva 20/38

En esta otra parte, nos encontramos con la clase `Treatment`, que tiene una relación de composición de 1 a muchos con `MedicalAppointment`. A su vez, tiene opcionalmente una notificación `MedicalAppointmentNotification` y una localización `Location`.

Diapositiva 21/38

Continuamos con el **modelado de comportamiento**, donde se han elaborado **diagramas de secuencia** de los principales casos de uso. Estos diagramas nos permiten conocer **qué sucede en el sistema durante su ejecución**. A continuación se presenta el diagrama de secuencia asociado al caso de uso "**Adición de un medicamento**".

Diapositiva 22/38

El diagrama de secuencia comienza con el usuario llamando al constructor de la clase `Medicine`. Entre otras cosas se pasa el tratamiento al que pertenece el medicamento, por lo que, para respetar el diagrama de clases, en el mismo constructor se añade el medicamento a la lista de medicamentos del tratamiento.

Diapositiva 23/38

Después, se comprueba si el medicamento existe o no previamente en la base de datos. Si no existe, se inserta en la base de datos cifrando previamente los datos del medicamento.

Diapositiva 24/38

Una vez el medicamento está almacenado en la base de datos, se establecen las dos notificaciones asociadas al medicamento: la notificación previa de toma y la del momento de toma.

Diapositiva 25/38

Por último, en el modelado y diseño del sistema, se realizó el **diseño de la interfaz de usuario**. Este proceso consistió en la elaboración de **mockups para guiar el proceso de implementación de la interfaz**. Estos diseños permiten **conocer la apariencia de la interfaz** y **cómo interactúa la interfaz**. Además, ayudan a refinar todos los requisitos funcionales del sistema.

Diapositiva 26/38

En este caso se presenta el *mockup* de adición de un tratamiento. Las flechas indican el flujo de acciones a seguir para completar el proceso de **adición de un tratamiento**.

Diapositiva 27/38

La **base de datos**, tal y como se ha mencionado anteriormente, es SQLite. Todos los **datos cifrados** tienen como tipo de datos **BLOB** y en el caso del almacenamiento de **fechas y horas** se ha utilizado el timestamp desde el EPOCH, por lo que se almacenan como **INTEGER**.

El **diagrama entidad-relación** se ha realizado en MySQL Workbench por lo que realmente genera un *script* SQL. Este *script* ha sido transformado a un *script* SQLite y se ha introducido en la carpeta **assets** de la aplicación. La clase `DatabaseHelper` se encarga de ejecutar el *script* para crear la base de datos la primera vez que el usuario interactúe con ella.

Diapositiva 28/38

Este es el **diagrama entidad-relación** de la base de datos en el que podemos observar todas las diferentes tablas y sus relaciones.

Diapositiva 29/38

En cuanto a la implementación, la podríamos dividir en **backend** y **frontend**.

Diapositiva 30/38

En el **backend** nos vamos a centrar en la implementación de la **seguridad**, las **excepciones** y las **notificaciones**.

Diapositiva 31/38

El **almacenamiento seguro de los datos** se lleva a cabo en las clases `SerializationUtils` y `SecurityService`. `SecurityService` contiene los métodos encargados de cifrar y descifrar, y llama a los métodos de `SerializationUtils` para serializar los datos de los objetos a cadenas de bytes.

Las contraseñas no se almacenan en claro, si no que se almacena su valor **HASH**. Se utiliza **SHA-256** junto con un **salt aleatorio de 16 bytes**. Esto permite que, a partir de dos contraseñas iguales, se obtengan valores HASH distintos, previniendo ataques como los de Rainbow Table.

El cifrado de los datos se realiza a partir del **algoritmo AES** en modo **CBC** con un padding **PKCS7**. Se utiliza la misma **clave simétrica** tanto para cifrar como para descifrar y se almacena en el sistema que proporciona Android denominado **Android KeyStore**, que proporciona ciertas medidas de seguridad.

El **proceso de cifrado** consiste en aplicar el algoritmo AES a una cadena de bytes junto con un vector de inicialización aleatorio, obteniendo un criptograma que se almacena en la base de datos junto con el vector. Este vector actúa de forma similar al *salt*, garantizando que cada criptograma sea único, independientemente de los datos de entrada.

Diapositiva 32/38

Se han creado **excepciones personalizadas** para asegurar un **seguimiento estructurado y organizado de las excepciones**. Existen hasta 13 excepciones que se anidan entre sí, facilitando la identificación de la causa de cualquier error en la aplicación. El método `advertiseUI` de la clase `ExceptionManager` se encarga de mostrar estos errores en la interfaz, tal como se observa en la imagen.

Diapositiva 33/38

La aplicación incluye **tres tipos de notificaciones**: una **previa a la toma de medicación**, otra en el **momento de la toma** (que se muestra en la imagen) y una para las **citas médicas**.

Las **notificaciones están precedidas por alarmas que se encargan de activarlas**. Android ofrece dos tipos de alarmas: exactas e **inexactas**. Las **alarmas inexactas** son agrupadas por el sistema operativo para minimizar el despertar del dispositivo. Android recomienda el uso de alarmas inexactas debido a consideraciones sobre el consumo de batería, y para el propósito de la aplicación, se ajustan perfectamente.

La clase `NotificationScheduler` se encarga de programar las alarmas para cada una de las notificaciones, mientras que la clase `NotificationPublisher` recibe la señal y publica la notificación correspondiente.

Por cómo funcionan los dispositivos Android, cuando el **dispositivo se reinicia**, todas las **alarmas programadas se desactivan**. La aplicación está prevista de esto, pero hace falta activar la opción de autoinicio en segundo plano.

Dependiendo de la versión Android del dispositivo es probable que haya que conceder **permiso para recibir notificaciones de la aplicación** y también comentar que la aplicación proporciona **canales de notificaciones** que permiten al usuario configurar las notificaciones de la aplicación a su gusto.

Diapositiva 34/38

En cuanto al **frontend**, este se ha desarrollado basado en los *mockups* elaborados durante el diseño de la interfaz de usuario. La interfaz se ha elaborado con código XML. En el que se establecen etiquetas para cada uno de los componentes y se anidan unos con otros estableciendo una especie de capas.

La interfaz ofrece **feedback visual** constante, informando al usuario sobre las acciones que realiza, tal como se ve en la imagen. El **diseño de la aplicación es responsivo**, permitiendo su uso tanto en orientación vertical como horizontal, y es compatible con múltiples dispositivos como móviles, *tablets* e incluso Smart TV. Además, la aplicación cuenta con una barra de navegación superior e inferior, lo que ayuda a mantener al usuario orientado dentro de la aplicación.

La información del usuario se transfiere entre actividades mediante la sesión `SessionViewModel`, que contiene la instancia del usuario activo en ese momento.

Como se puede observar en la imagen, la aplicación permite configurar un **modo claro u oscuro** y **ofrece hasta 9 idiomas distintos**. Esta configuración puede seguir la configuración establecida en el dispositivo o ser completamente independiente.

Diapositiva 35/38

Tras la implementación, se procedió a realizar **pruebas unitarias** centradas únicamente en las principales funcionalidades de la aplicación, debido al significativo tiempo que requiere una batería completa de pruebas. Se utilizaron **mocks para aislar las unidades de código** a probar, y todos los **tests están parametrizados para ejecutar el mismo test con diferentes inputs**.

En total, se llevaron a cabo **30 pruebas**, entre las que se prueban las clases `AuthenticationService`, `MedicalAppointment`, `Medicine` y `Treatment`.

Diapositiva 36/38

Ahora voy a realizar una breve **demostración de la aplicación**.

Antes solo mencionar que la aplicación está **disponible** exclusivamente **para dispositivos Android con versión 9.0 o superior**, lo que representa aproximadamente el **86.4%** de los dispositivos. Dado que no se encuentra disponible en Google Play Store, es necesario **habilitar instalaciones de orígenes desconocidos**. Además, el **manual de usuario se encuentra accesible en todo momento a través de la aplicación**.

Tengo un vídeo preparado, pero voy a intentar realizar la demostración en vivo. El recorrido va a ser por las principales funcionalidades, pero si tienen **alguna sugerencia** concreta que quieran realizar me la pueden comentar sin problema.

Diapositiva 37/38

Demo: https://youtu.be/H_jH0Lwfrtg

Diapositiva 38/38

Para terminar, quiero comentar que sinceramente he **superado mis expectativas iniciales** con este proyecto. Se ha logrado **cumplir con todos**

los requisitos, incluso varios opcionales. Pero lo más importante es que se ha **desarrollado una aplicación realmente útil** para las personas.

Como con todo proyecto me he enfrentado a varios retos. Una de las principales complicaciones fue la **gestión de las variaciones entre versiones y fabricantes**, lo cual repercutía directamente en el **control de los permisos de la aplicación**. Además, la **gestión de las alarmas y las notificaciones** también resultó difícil debido al desconocimiento inicial de su funcionamiento.

Como líneas futuras, es evidente que la aplicación podría **expandirse con más funcionalidades**, lo cual siempre es beneficioso. Sin embargo, considero que lo más importante es abordar la limitación actual de almacenamiento local de datos en el dispositivo. **Implementar un sistema remoto** que almacene los datos en un servidor ofrecería una mayor flexibilidad, pero también requeriría un aumento significativo en la seguridad debido a la naturaleza sensible de los datos. Afortunadamente, muchas de estas medidas ya están implementadas en la actual versión. Además, otra mejora potencial sería **permitir el acceso a profesionales de la salud** para que ellos puedan prescribir los tratamientos directamente a los usuarios, en lugar de que estos introduzcan los tratamientos por sí mismos. Sin embargo, esto podría implicar cierto papeleo y gestión adicional.

Diapositiva 1/38

Muchas gracias por su atención. Si tienen cualquier pregunta estoy a su total disposición.