APLICACIONES TELEMÁTICAS

Grado en Ingeniería de Tecnologías y Servicios de Telecomunicación

RED SOCIAL DE VIAJES EN ANDROID STUDIO – THE TRAVEL APP

PABLO ÚCAR GARGALLO

LAURA OREJAS GONZALEZ

HÉCTOR CATALÀ DOMÉNECH

EDUARD LLISTÓ ESTELA

CURSO ACADÉMICO 2024-2025
E.T.S.I. DE TELECOMUNICACIÓN
UNIVERSITAT POLITÈCNICA DE
VALÈNCIA 6 DE JULIO DE 20

1. Introducción

A la hora de realizar una aplicación en Android Studio, y siendo 4 personas en el grupo, decidimos hacer un proyecto con el que estuviéramos contentos trabajando y nos motivara a dedicarle horas. A demás queríamos enfrentarnos a un reto que nos permitiera aprender a fondo el funcionamiento y los recursos de Android Studio. Es por este motivo que nos decidimos a ser bastante ambiciosos a la hora de hacer el proyecto y dedicarle la mayor cantidad de tiempo y esfuerzo posible para lograr un buen resultado. Partiendo de esta base se nos ocurrió la idea de hacer una red social dedicada a los viajes.

En ella, los usuarios tendrían acceso a tres pantallas diferentes con relación entre sí. La primera de estas pantallas sería el menú del usuario, o perfil. En ella, el usuario podría ver los viajes realizados que hubiera decidido subir a la aplicación, así como la opción de subir más viajes. La idea principal era que fuera lo más sencilla e intuitiva posible. Por esta razón, el planteamiento original era que presentara una imagen del usuario, su nombre de perfil y una descripción del usuario. Así mismo se incluirían dos botones, uno que abriría una pestaña para añadir un nuevo viaje y uno que diera acceso a visualizar los viajes ya realizados por el usuario.

La segunda pantalla sería la pantalla social. En esta pantalla, el usuario tendría acceso a los viajes realizados por otros usuarios agregados como amigos. A su vez se podría ver la información de estos amigos y la información básica de estos viajes. La información de cada viaje se presentaría con un título del viaje, dado por el usuario que lo ha planteado, así como una imagen del mismo y una breve descripción de las actividades realizadas en el viaje. Finalmente se planteó la posibilidad de añadir un botón para añadir nuevos usuarios como amigos.

Finalmente se plantearía una pantalla con un mapa. La idea original era integrar Google Maps o un servicio similar para plantear una aplicación directa al mundo real. Pretendemos que esta pantalla sirva para darle al usuario una experiencia más inmersiva y tangible a la hora de utilizar la aplicación. Con el mapa queremos lograr que el usuario vea la ubicación exacta de los viajes realizados por sus amigos y lo que él mismo ha hecho. Para esto emplearemos pines de diferentes colores que señalicen la ubicación de los distintos sitios en los que se han realizado los viajes.

Con todo esto planteado, nos decidimos a empezar la aplicación, pero rápidamente nos topamos con varios contratiempos y dificultades que nos hicieron ver que había partes del proyecto planteado que no era factibles para realizarse en nuestra aplicación. El mayor de estos problemas era la integración del mapa. Para integrar Google Maps es necesario integrar una Api, la cual tiene un coste económico que escapa a nuestras posibilidades, y que se nos hacía muy complicada de integrar. Partiendo de este problema decidimos poner un mapa de La Tierra Media, proveniente del universo de fantasía del Señor de los Anillos, de J.R.R. Tolkien. A partir de aquí decidimos tematizar toda nuestra red social tomando como referencia la obra de Tolkien.

Otro problema al que nos enfrentamos fue afrontar las acciones que tomaba el usuario de cara a crear nuevos viajes o añadir nuevos amigos. Estas acciones al requieren dejar que el usuario creara en base a la plantilla ya hecha y teniendo el mismo usuario que registrar los textos, la ubicación en el mapa y posibles contenidos multimedia, en el caso de añadir un nuevo viaje, y el tener que buscar usuarios entre los ya existentes para añadir nuevos amigos, con el sistema de solicitudes correspondiente. Es por eso que, pese a las funcionalidades que perderíamos, añadir estas funciones suponía mucha complejidad y un reto técnico que no éramos capaces de afrontar.

A raíz de todas estas cosas, decidimos enfocar la aplicación a lo que sería una especie de "versión de prueba". Si bien funcional en el funcionamiento de sus menús y el movimiento a través de ellos, limitando la interacción del usuario a la experiencia de un entorno ya creado, similar en interfaz y experiencia de uso a lo que sería una versión 100% funcional pero adaptada a una interacción simple.

2. Pantalla del Mapa

Nada más abrir la aplicación se nos abre la pantalla del mapa, y en la parte de debajo de la pantalla nos aparece el menú de secciones, Como se ve en la Figura 1. Este menú nos permite viajar cómodamente entre las distintas secciones que tiene nuestra aplicación. Volviendo con el mapa, en este destacan los pines, de azul y amarillo, que resaltan localizaciones marcadas como viajes. Los pines azules referencian a viajes de otros usuarios agregados como amigos y los amarillos indican los viajes marcados por el usuario de la aplicación.

Volviendo con el mapa, este es una imagen de La Tierra Media, de la obra El Señor de los Anillos, escrita por J.R.R. Tolkien, y que se ve en la Figura 2. Queriamos disponer el mapa de tal forma que fuera fácil desplazarse por él, que ocupara toda la pantalla y que fuera ampliable. Después de darle muchas vueltas, probar múltiples opciones de mapas que pudieran adaptarse directamente a la pantalla y buscar soluciones que se adaptasen a nuestro objetivo, decidimos implementar la librería PhotoView. Esta librería se encarga de, partiendo de los píxeles de la imagen, generar una matriz que identifica a cada píxel, así como implementar directamente las funciones para ampliar la imagen del mapa, moverse por ella y adaptar el tamaño del mismo a la pantalla sin que se pierda información de la imagen.

El siguiente paso era implementar los pines que marcan las ubicaciones de los viajes en el mapa. Estos pines tenían que permanecer estáticos en una posición determinada del mapa, independientemente del movimiento del mismo. A su vez deberían tener siempre el mismo tamaño, aunque se ampliara o encogiera el mapa. Para ello implementamos dos imágenes, las vistas en la Figura 3 y la Figura 4, para los pines de los viajes del usuario y de los amigos del mismo respectivamente. Entonces les asignamos una posición fija a cada uno en la imagen, a partir de un float que toma como referencia la matriz que genera PhotoView. A partir de aquí se juega con esta referencia para permitir que los pines se queden fijos en el punto de la imagen que deseamos y que no se reescalen con el mapa.

Finalmente convertimos cada pin en un botón, de manera que, al clickar en ellos, se nos abra un popup con el nombre de usuario que ha hecho el viaje y un botón para ver el viaje. Una vez presionamos el botón, se nos redirige a una pantalla que nos muestra el viaje más a profundidad, con una descripción del mismo y un botón que nos permite volver al mapa. Esta pantalla la conocemos como pantalla de Detalles del viaje y volveremos a ella cada vez que queramos ver la información específica de un viaje en concreto.

En cuanto a código, y a modo de explicarlo todo junto, se ha utilizado la librería PhotoView para realizar las funciones de zoom y desplazamiento de la imagen del mapa. Cuatro pines llevados a la vida por imágenes .png de "chinchetas" son representados encima del mapa en ciertas coordenadas predefinidas mediante ImageView. Al pulsar uno de los pines salta una ventana emergente (PopupWindow), en la cual se ofrece un botón para ver los detalles del viaje en una nueva ventana (DetallesViaje). Estos viajes están predefinidos y por tanto enteramente codificados en el archivo anteriormente nombrado. El paso de objetos entre actividades se hace con Intent y Serializable.

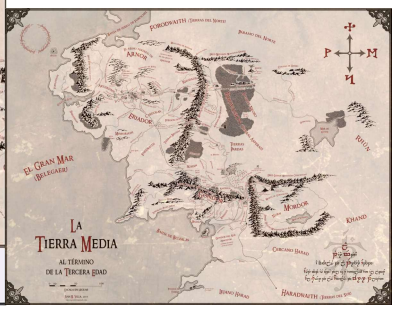

FIGURA 1. Pantalla de Mapa. FIGURA 2. Mapa de La Tierra Media


FIGURA 3. Pin Usuario FIGURA 4. Pin Amigos

3. Pantalla Social

La siguiente pantalla en la que nos vamos a centrar es la pantalla social, que se ve en la Figura 5. En ella podemos ver los viajes que han realizado otros usuarios que tengamos agregados como amigos. Estos viajes se nos presentan con una foto, un título dado por el usuario que ha hecho el viaje y una pequeña descripción del mismo. A la hora de obtener estas imágenes, y debido a la falta de imágenes de los personajes en las situaciones que buscábamos, decidimos generar imágenes con inteligencia artificial que se adaptaran a los viajes planteados. Estas imágenes se ven también en la Figura 5.

Cuando se pulsa en la foto de cualquiera de estos viajes se nos redirige nuevamente a la pantalla de Detalles del Viaje, que ya hemos mencionado en la pantalla del Mapa. En esta pantalla podemos ver la información detallada del viaje que hayamos seleccionado. Podemos ver el nombre de usuario de la persona que ha subido el viaje, la foto del mismo, así como datos adicionales de este, como se ve en la Figura 6. Adicionalmente a esto tenemos un botón que nos permite ir nuevamente a la pantalla del Mapa. Tratemos el código. La vista se infla con ViewBinding, se obtiene la referencia al RecyclerView y se le asigna un LinearLayoutManager para mostrar los ítems (viajes) a modo de desplazamiento vertical. Los viajes en sí son una lista de objetos Trip creados con parámetros predefinidos. Se instancia TripAdapter, enviándole la lista de viajes y un callback que al pulsar un item lanza la actividad DetallesViaje (varios TextView) enviando el objeto Trip seleccionado (un viaje específico) mediante Intent. Los viajes se renderizan mediante RecyclerView.

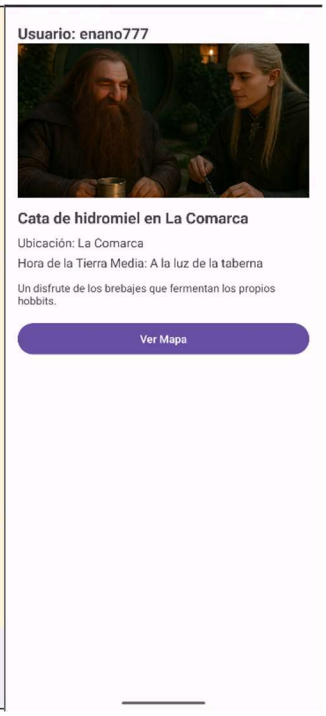

FIGURA 5. Pantalla Social. FIGURA 6. Pantalla de Detalles del Viaje

4. Pantalla Perfil

La tercera y última pantalla de la aplicación es la del perfil del propio usuario. Aparece su fotografía de perfil, su nombre de usuario y una pequeña descripción de su perfil. Bajo esta pequeña "identificación" se encuentra el botón "Añadir viaje", el cual lleva a una pestaña en la que se pueden seleccionar diversas opciones multimedia y redactar el título y descripción de un hipotético viaje. Lamentablemente, y a pesar de nuestros esfuerzos y deseos, esta mecánica no es realmente funcional. Desde la pantalla origen en el perfil del usuario se ve también el viaje realizado por el propio usuario (el que ya aparecía en la pantalla de mapa), siendo esta la otra forma de acceder a él. Un click encima nos llevaría a la pantalla del viaje mediante la misma mecánica que nos permitía acceder a los viajes de amigos en la pestaña social.

De manera técnica, podemos hablar de que la Pantalla Perfil, de forma general (archivo ProfileFragment.java), hace uso de ViewBinding para acceder al XML del Layout, un adaptador para RecyclerView y navega entre varios fragmentos y actividades dedicados a los distintos botones y texto. Binding se inicializa en en paso Create y se libera en Destroy para evitar fugas de memoria. Se usa RecyclerView para mostrar la lista de viajes (al añadido y todos los que se podrían agregar mediante código), que se almacenan de manera vertical por un LinearLayoutManager.

Los viajes, objetos Trip, se implementan mediante un adaptador TripAdapter que mediante un listener maneja los clicks en cada elemento. Para acceder al fragmento del botón "Añadir viaje", AddTripFragment, se usa el botón addTripButton. Los mensajes informativos que aparecen al pulsar botones dentro de la pantalla se logran mediante toast. De otro modo, cuando se quiere acceder a un viaje en concreto, se pulsa sobre él y se despliégala actividad DetallesViaje, a la que se le pasa el objeto Trip mediante Intent.
El resto de los elementos son en su mayoría ImageView y TextView.

FIGURA 7. Pantalla Perfil. FIGURA 8. Pantalla de Añadir Viaje

5. Conclusiones

El desarrollo de la aplicación ha sido para los cuatro un verdadero desafío, pero a su vez, posiblemente, el mayor aprendizaje del que hemos podido disfrutar a lo largo de la asignatura. La idea inicial era ambiciosa, muy probablemente demasiado: crear una aplicación totalmente funcional, atractiva para un hipotético y, por qué no, posible usuario. Hacia mitad de camino (la aplicación se empezó más o menos cuando en clase se trató por primera vez el tema del proyecto) nos dimos cuenta de que, a nuestro nivel de conocimiento del lenguaje, rodeados por el resto de tareas académicas que debíamos de llevar a cabo, y frenados por los recursos económicos de los que disponíamos, decidimos que había que hacer un corte de alguna manera. Sin embargo, este contratiempo no nos frenó, sino que nos obligó a movernos a otro camino por el que correctamente hemos sabido seguir.

Uno de los principales aprendizajes ha sido la importancia en nuestro proyecto de modularizar el diseño, entendiendo de una forma más cercana a la explicada en las clases de teoría la profundidad del ciclo de vida de las distintas actividades y fragmento, el manejo correcto correcto de los elementos visuales mediante librerías como PhotoView o componentes como RecyclerView. A pesar de haberse visto mermadas funcionalidades como la integración de Google Maps o el alta dinámica de usuarios, finalmente logramos construir una aplicación correctamente navegable, con una interfaz más que intuitiva y ambientada de forma original siguiendo la temática de La Tierra Media (idea de Pablo Beneit, cuando consultado por la viabilidad de la implementación del API del mapa de Google, aunque él no lo recuerde).

El trabajo en equipo y la compenetración han sido clave para superar los retos técnicos y de diseño. Los miembros ya habíamos trabajado juntos anteriormente, lo que nos permitió maximizar la eficiencia a la hora de distribuir tareas, recopilar información, integrar distintos módulos y mantener la coherencia del proyecto de una forma lineal, tanto a nivel interfaz como de arquitectura de código. Además, la utilización de herramientas modernas como las IAs y ciertas buenas prácticas recomendadas y recopiladas en exhaustivas búsquedas en páginas web y libros digitales, además de los muy completos recursos de la asignatura, nos han permitido sentar una base sólida para una ilusionante y no tan imposible expansión futura del proyecto.

En resumen: esta aplicación (The Travel App) representa una versión de prueba semifuncional, que simula con éxito la experiencia de uso de una red social de viajes. Aunque no incorpora todas las capacidades interacticas previstas originalmente, sí que transmite el propósito del proyecto y demuestra nuestra capacidad de asumir el desarrollo de una aplicación mediante lenguaje Java en Android Studio. Valoramos muy positivamente el crecimiento que la tarea nos ha permitido adquirir en cuanto a competencias técnicas y capacidad de gestión de proyectos. Gracias a Irene por la idea original.

6. Citas y bibliografía

1. Android Developers. (n.d.). *Recommendations for Android architecture*. https://developer.android.com/topic/architecture/recommendations
2. Android Developers. (n.d.). *Fragment lifecycle*. https://developer.android.com/guide/fragments/lifecycle
3. Android Developers. (n.d.). *Handling lifecycles with lifecycle-aware components*. https://developer.android.com/topic/libraries/architecture/lifecycle
4. Alcérreca, J. (2017, December 5). *The Android Lifecycle cheat sheet — part III: Fragments*. Medium. https://medium.com/androiddevelopers/the-android-lifecycle-cheat-sheet-part-iii-fragments-afc87d4f37fd
5. Android Developers. (n.d.). *R.drawable*. https://developer.android.com/reference/android/R.drawable
6. Lake, I. (2020, August 19). *Fragments: rebuilding the internals*. Medium. https://medium.com/androiddevelopers/fragments-rebuilding-the-internals-61913f8bf48e
7. Beyls, C. (2017, October 24). *Architecture Components pitfalls — Part 1*. Medium. https://bladecoder.medium.com/architecture-components-pitfalls-part-1-9300dd969808
8. Poespas Blog. (2024, April 29). *Mastering Android Fragment Lifecycle Management: Tips and Tricks for Efficient Development*. https://blog.poespas.me/posts/2024/04/29/android-fragment-lifecycle-management-tips
9. Poespas Blog. (2024, May 2). *Mastering Fragment Lifecycle Management in Android Development: Best Practices and Tricks*. https://blog.poespas.me/posts/2024/05/02/android-fragment-lifecycle-management-best-practices
10. Stack Overflow user. (2019). *Best practice to use BottomNavigation in Android — Activity vs Fragment* [Online forum comment]. Stack Overflow. https://stackoverflow.com/questions/57437844
11. Reddit user. (2020, May 2). *A reminder that Single Activity App Architecture has been the official Google recommendation since 2 years ago*. Reddit. https://www.reddit.com/r/androiddev/comments/gc0yw3
12. Reddit user. (2022, April 9). *Example of Lifecycle memory leaks that LiveData prevents*. Reddit. https://www.reddit.com/r/androiddev/comments/fxvezn
13. Reddit user. (2020, March 9). *TIL How to use FragmentStateAdapter with ViewPager2 with Dynamic Fragments*. Reddit. https://www.reddit.com/r/androiddev/comments/ffso1z
14. Reddit user. (2019, August 20). *Is having multiple Fragments sharing a ViewModel a violation of the Single Responsibility Principle?* Reddit. https://www.reddit.com/r/androiddev/comments/ct2bkg
15. Android Developers. (n.d.). *App architecture: UI layer*. https://developer.android.com/topic/architecture/ui-layer
16. Material Design. (n.d.). *Bottom navigation bar*. https://m3.material.io/components/bottom-navigation/overview
17. Android Developers. (n.d.). *Dependency injection with Hilt*. https://developer.android.com/training/dependency-injection/hilt-android
18. Android Developers. (n.d.). *Use Hilt with other Jetpack libraries*. https://developer.android.com/training/dependency-injection/hilt-jetpack

19. Android Developers. (n.d.). *Fundamentals of testing Android apps*. https://developer.android.com/training/testing/fundamentals
20. Android Developers. (n.d.). *Espresso basics*. https://developer.android.com/training/testing/espresso/basics