Evidencia día 05 Semana 14

Se debe continuar trabajando en el ejercicio de ayer:

Ejercicio:

Se requiere desarrollar una aplicación móvil para Android en el lenguaje Kotlin o Java, esta aplicación móvil debe realizar las siguientes acciones:

1. La aplicación se conectará a un API Rest del cual obtendrá un listado de libros mostrando su título, autor y una miniatura de su portada.
2. La aplicación permitirá seleccionar libros y mostrar su detalle.
3. La aplicación permitirá ver los libros favoritos sin conexión.
4. La aplicación contará con un botón que permite enviar correo electrónico utilizando alguna aplicación de correo.
5. Agregar 4 nuevas funcionalidades a la aplicación.
6. Elaborar diagramas de casos de uso y detallar requerimientos funcionales

Desarrollo:

5. Agregar 4 nuevas funcionalidades a la aplicación.

- Que la aplicación se pueda configurar a idioma Inglés.
- un componente me gusta y un contador para cada libro
- un fragmento donde se pueda escribir, editar y eliminar un comentario del libro.
- Que la aplicación pueda cambiar de tema a modo nocturno.

1. La aplicación se conectará a un API Rest del cual obtendrá un listado de libros mostrando su título, autor y una miniatura de su portada.

Para desarrollar este ítem, se requiere de conocimientos que no vimos en clases por lo que se investigó tutoriales y material en google.

APIRest

¿Qué es un API REST?
Podríamos definir API REST como un servicio que nos provee de las funciones que necesitamos para poder obtener información de un cliente externo, como por ejemplo, una base de datos alojada en cualquier parte del mundo desde dentro de nuestra propia aplicación.

Vamos a pensar en Instagram, una aplicación con millones de usuarios. Es inviable tener la información de cada usuario dentro de la aplicación ¿verdad? Pues para solventar el problema usan servicios API REST. Lo primero que hacemos al entrar en la app es un login, esto sería el primero de los servicios, pues nosotros le mandamos al servidor el usuario y contraseña y este nos devolvería la información que debemos mostrar en la app.

Disponemos de cuatro tipos distintos de peticiones como norma general.

Get: Son las peticiones más sencillas, solo nos devuelven información. Si necesitamos pasarle un parámetro a la petición será a través de la url. Es decir si por ejemplo tenemos que hacer una petición que depende de una id (ej la identificación del usuario) la url se formaría así https://ejemplo.com/informacion/1, siendo 1 el parámetro que le pasamos. El problema de esto es que es poco seguro para pasar información delicada.

Post: Similar a Get pero los parámetros no se pasan por url, por lo que es más seguro para mandar información.

Put: Se suele usar para crear la entidad, es decir, si pensamos en un servicio como el acceso a una base de datos, este crearía el usuario por ejemplo.

Delete: Sería el último de los cuatro que nos permitiría borrar los registros de la base de datos.

La información suele venir en dos formatos distintos, XML o JSON. Para no meternos mucho en el tema, solo hablaremos del JSON que es el formato más habitual y con el que nosotros vamos a trabajar.

Formato JSON

Json es un formato de texto simple, es el acrónimo de JavaScript Object Notation. Se trata de uno de los estándar para el traspaso de información entre plataformas, tiene una forma muy legible que nos permite entender su contenido sin problema. Un ejemplo sencillo sería este.

```
{
  "employees": {
    "employee": [
      {
        "id": "1",
        "firstName": "Tom",
        "lastName": "Cruise",
        "photo": "https://jsonformatter.org/img/tom-cruise.jpg"
      },
      {
        "id": "2",
```

```
      "firstName": "Maria",
      "lastName": "Sharapova",
      "photo": "https://jsonformatter.org/img/Maria-Sharapova.jpg"
    },
    {
      "id": "3",
      "firstName": "Robert",
      "lastName": "Downey Jr.",
      "photo": "https://jsonformatter.org/img/Robert-Downey-Jr.jpg"
    }
  ]
 }
}
```


Todo formato Json empieza y termina con llaves y tiene una clave-valor. La clave employees contiene a su vez una lista de employee (fijaros que en vez de llaves tiene corchetes), que este almacena id, firstName, lastName y photo. Así podemos pasarnos gran cantidad de información de una plataforma a otra con unos estándar que nos ayudan a simplificar el proceso.

Si aún así se os complica la lectura de estos ficheros al principio, podemos hacer uso de multitud de webs que simplifican la forma de verlo, como por ejemplo JsonEditOnline.

Reflexión:
Seguimos trabajando en el ejercicio de ayer, desarrollamos el punto de las nuevas funcionalidades, diagrama de casos de uso y los requerimientos funcionales. De forma individual se revisa contenido en internet referente a APIrest y sus aplicaciones y se practica con un tutorial que al estar desactualizado no pude hacer funcionar en el emulador.

Fuentes:
https://public-apis.io/
https://cursokotlin.com/capitulo-15-recyclerview-kotlin/
https://cursokotlin.com/tutorial-retrofit-2-en-kotlin-con-corrutinas-consumiendo-api-capitulo-20-v2/