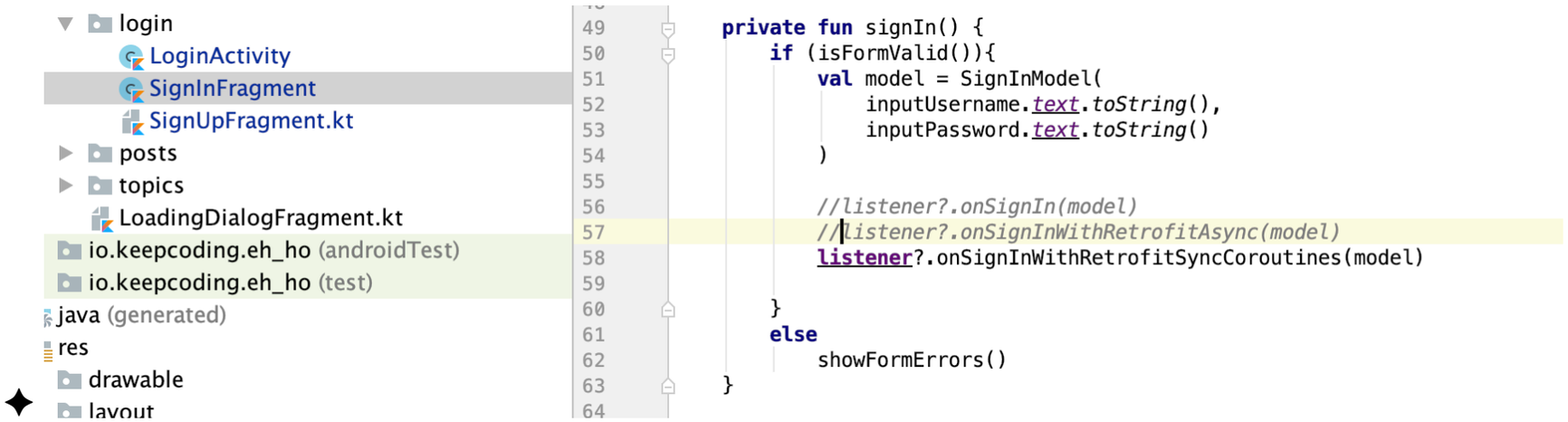
Práctica Módulo Android Avanzado
Mónica Sanmartín Gregorio
Febrero de 2020

- Se ha implementado un NavigationDrawer en la AppBar de la app, con un widget que proporciona las opciones de “Latest Topics” y “Latest Posts”, a modo de sección de latest news de la app.
- En la nueva funcionalidad de “Latest News” se carga un nuevo fragment que permite visualizar un listado de los últimos posts publicados en cada Topic. Se muestran en una lista (*RecyclerView*) cuyos elementos componentes son del tipo *CardView*, siendo éstos clickables, de forma que al seleccionar un elemento de la lista (un “latest post”), se muestra una nueva pantalla con el detalle del Topic “padre” de ese post, es decir, el listado completo de posts de ese topic.
- Las llamadas a la API de Discourse se hacen mediante la librería Volley.
- La librería Retrofit se usa en la llamada a la API de discourse para la funcionalidad de Login de la aplicación.

- Dentro de UserRepo se han implementado los métodos para probar el acceso a la API de discourse para el login, con Retrofit de forma asíncrona (fun *signInWithRetrofitAsynchronously*) o de forma síncrona y utilizando coroutines (suspend fun *signInWithRetrofitSynchronouslyWithinCoroutines(signInModel: SignInModel)*). Se han probado ambas opciones y se pueden seleccionar en el método privado sigIn del SignInFragment.



- El listado de posts recientes se persiste en una BD local creada con Room (PostsDatabase), en cuyo repositorio (PostRepo), se ha intentado, sin éxito, obtener mediante inyección de dependencias la base de datos y el contexto (lateinit var db & ctx). Se ha usado Dagger como inyección de dependencias pero no se ha conseguido que se inicialice la BD mediante este mecanismo. Finalmente, para probar igualmente la base de datos se ha optado por crear la bd en el propio PostRepo, algo poco recomendable pero necesario para al menos probar la BD.

- Para probar la inyección de dependencias habría que comentar la creación de la var db en el método getLatestPosts del PostRepo.

```
object PostsRepo  {

    lateinit var db: PostsDatabase
    lateinit var ctx: Context

    //_____________________________________________

    fun getLatestPosts(
        context: Context,
        onSuccess: (List<LatestPost>) -> Unit,
        onError: (RequestError) -> Unit
    ) {

        Log.d( tag: "GET LATEST POSTS",  msg: "...... POSTS REPO.................

        var db : PostsDatabase = Room.databaseBuilder(
            context, PostsDatabase::class.java,  name: "posts_database"
        ).build()
```

- En cualquier caso, a continuación se detallan las diferentes pruebas y sus resultados con Dagger.

Buid del DaggerApplicationGraph en el onActivityCreated

1) utilsModule : deprecado??
2) Cómo obtener el contexto para el UtilsModule dentro del onActivityCreated del fragmento (LatestPostFragment?

di
- ApplicationGraph
- PostsModule
- UtilsModule

```
override fun onAttach(context: Context) {  context: MainActivity@11407
    //  Log.d("LATEST POST FRAGMENT ","ON ATTACH_____________**********

//    DaggerApplicationGraph.builder().utilsModule(UtilsModule(context)).build().inj
    super.onAttach(context)

    if (context is LatestPostInteractionListener)  context: MainActivity@11407
        listener = context
}
```

```
override fun onActivityCreated(savedInstanceState: Bundle?) {  savedInstanceState: null
    Log.d( tag: "LATEST POST FRAGMENT ", msg: "On activity created_____________******")

var ctx = context!!  ctx: MainActivity@11407
    DaggerApplicationGraph.builder()
        .utilsModule(UtilsModule(ctx)).build()  ctx: MainActivity@11407
        .inject( latestPostsFragment: this)
    super.onActivityCreated(savedInstanceState)
}
```

ApplicationGraph

```
package io.keepcoding.eh_ho.di

import dagger.Component
import io.keepcoding.eh_ho.data.PostsRepo
import io.keepcoding.eh_ho.home.MainActivity
import io.keepcoding.eh_ho.latestposts.LatestPostsActivity
import io.keepcoding.eh_ho.latestposts.LatestPostsFragment

import javax.inject.Singleton

// @Component makes Dagger create a graph of dependencies
@Singleton
@Component(modules = [PostsModule::class, UtilsModule::class])
interface ApplicationGraph {
    // Add functions whose return value indicate what can be provided from this container

    // Add here as well functions whose input argument is the entity in which Dagger can add any
    // dependency you want

    fun inject(latestPostsFragment: LatestPostsFragment)
    //fun inject(latestPostsActivity: LatestPostsActivity)
    //fun inject (mainActivity: MainActivity)
}
```

PostsModule

```
import android.content.Context
import dagger.Module
import dagger.Provides
import io.keepcoding.eh_ho.data.PostsRepo
import io.keepcoding.eh_ho.data.PostsDatabase
import javax.inject.Singleton

@Module
class PostsModule {

    @Singleton
    @Provides

    fun providePostRepo(context: Context, postDatabase: PostsDatabase): PostsRepo {
        PostsRepo.ctx = context
        PostsRepo.db = postDatabase
        return PostsRepo
    }

}
```

UtilsModule

```
package io.keepcoding.eh_ho.di

import android.content.Context
import androidx.room.Room
import dagger.Module
import dagger.Provides
import io.keepcoding.eh_ho.data.Post
import io.keepcoding.eh_ho.data.PostsDatabase
import javax.inject.Singleton

@Module
class UtilsModule(private val context: Context) {

    @Singleton
    @Provides
    fun provideApplicationContext(): Context = context

    @Singleton
    @Provides
    fun providePostDb(ctx: Context): PostsDatabase = Room.databaseBuilder(
        context, PostsDatabase::class.java,  name: "posts_database"
    ).build()


}
```

PostRepo

```
object PostsRepo  {

    lateinit var db: PostsDatabase
    lateinit var ctx: Context

    fun getLatestPosts(
        context: Context,  context: MainActivity@11407
        onSuccess: (List<LatestPost>) -> Unit,  onSuccess: "Function1<java.util.List<? extends io.keepcoding.eh_ho.data.LatestPost>, kotlin.Unit>"
        onError: (RequestError) -> Unit  onError: "Function1<io.keepcoding.eh_ho.data.RequestError, kotlin.Unit>"
    ) {

        Log.d( tag: "GET LATEST POSTS",  msg: "..... POSTS REPO.................")

    /*   var db : PostsDatabase = Room.databaseBuilder(
            context, PostsDatabase::class.java, "posts_database"
        ).build()*/

        val username = UserRepo.getUsername(context)  username: "msanmar"  context: MainActivity@11407

        val request = UserRequest(
            username,  username: "msanmar"
            Request.Method.GET,
            ApiRoutes.getLatestPosts(),
             body: null,
            { it: JSONObject?
                it?.let { it: JSONObject

                    onSuccess.invoke(LatestPost.parseLatestPosts(it))
                    thread {
                        db.postDao().insertAll(LatestPost.parseLatestPosts(it).toEntity())
                        Log.d( tag: "GUARDADOS POSTS EN BBBD",  msg: "................. OK GUARDADOS")
                    } ^let
                }

                if (it == null)
                    onError.invoke(RequestError(messageResId = "Invalid response from server"))
            },
```

1) El context que se inyecta y el que llega como argumento en la fun *getLatestPosts* parecen ser el mismo, MainActivity11407, pero finalmente la BD aparece siempre como no inicializada

- this = {PostsRepo@12063}
- context = {MainActivity@11407}
- onSuccess = {LatestPostsFragment$loadLatestPosts$$inlinedletlambda$1@12064} "Function1<java.uti..
- onError = {LatestPostsFragment$loadLatestPosts$1$2@12065} "Function1<io.keepcoding.eh_ho.data.Requ
- username = "msanmar"

Se recuperan bien los posts pero no se pueden guardar porque db no está inicializada

- onError = {LatestPostsFragment$loadLatestPosts$1$2@12065} "Function1<io.keepcoding.eh_ho.data.Requ
- onSuccess = {LatestPostsFragment$loadLatestPosts$$inlinedletlambda$1@12064} "Function1<java.uti..
- it = {JSONObject@12218} "{"latest_posts":[{"id":529,"name":"Mónica","username":"msanmar","avatar_templ..

Frames | Threads

"Thread-7"@12,230 in group "main": RUNNING

invoke:46, PostsRepo$getLatestPosts$request$1$1$1 *(io.keepcoding.eh_ho.data)*
invoke:16, PostsRepo$getLatestPosts$request$1$1$1 *(io.keepcoding.eh_ho.data)*
run:30, ThreadsKt$thread$thread$1 *(kotlin.concurrent)*

Variables

- it = {JSONObject@12218} "{"latest_posts":[{"id":529,"name":"Mónica","username":"msanmar","avatar_templ...
  - nameValuePairs = {LinkedHashMap@12359} size = 1
    - "latest_posts" -> {JSONArray@12365} "[{"id":529,"name":"Mónica","username":"msanmar","avatar_te...
      - key = "latest_posts"
      - value = {JSONArray@12365} "[{"id":529,"name":"Mónica","username":"msanmar","avatar_template"...
  - shadow$_klass_ = {Class@8036} "class org.json.JSONObject"... Navigate
  - shadow$_monitor_ = 0

```
E/AndroidRuntime: FATAL EXCEPTION: Thread-7
    Process: io.keepcoding.eh_ho, PID: 22504
    kotlin.UninitializedPropertyAccessException: lateinit property db has not been initialized
        at io.keepcoding.eh_ho.data.PostsRepo.getDb(PostsRepo.kt:20)
        at io.keepcoding.eh_ho.data.PostsRepo$getLatestPosts$request$1$1$1.invoke(PostsRepo.kt:87)
        at io.keepcoding.eh_ho.data.PostsRepo$getLatestPosts$request$1$1$1.invoke(PostsRepo.kt:16)
        at kotlin.concurrent.ThreadsKt$thread$thread$1.run(Thread.kt:30)
I/Process: Sending signal. PID: 22504 SIG: 9
Process 22504 terminated.
```