StarWarsWiki

Balint Cristian

February 2023

Contents

1 Introducere

In domeniul dezvoltarii aplicatiilor mobile exista anumite trenduri de tipuri de aplicatii: social media, educational, product store, mobile banking, games etc. acest domeniu este un mediu foarte competitiv si dinamic in care este important sa participam. Pentru ca niciodata nu stim daca dam sau nu lovitura.

2 Scop si obiective

Aplicatia este asemanatoare unui wiki pentru fanaticii renumitei serii starwars, o serie de filme si jocuri de natura science fiction apreciata de oameni de toate varstele. Scopul aplicatiei este unul informativ pentru persoanele care vor sa afle mai multe despre filmele, personajele si planetele acestui univers fictional.

Obiective:

- Meniu principal selectare subiect
- Obtinere date de la un api cu ajutorul retrofit
- Utilizare GSON pentru serializarea obiectelor in JSON si vice versa
- Lista pentru fiecare subiect care sa contina multiple personaje, filme sau planete

3 Context de utilizare

Contextul de utilizare al aplicatiei este cel informativ, atractiv atat fanaticilor veterani cat si noilor fani fascinati de universul Star Wars.

ex: Un domn mai inaintat in varsta vrea sa citeasca aceasta serie, pentru ca i-a fost recomandata de nepoata.

4 Implementare

Pentru construirea aplicatie am folosit:

- retrofit:2.9.0
- converter-gson:2.9.0
- lifecycle_version = 2.7.0
- lifecycle-viewmodel-ktx:2.7.0
- lifecycle-livedata-ktx:2.7.0
- kapt lifecycle-compiler:2.7.0

Aplicatia utilizeaza un api pentru obtinerea informatiilor ce vrem sa le afisam. Api-ul se numeste swapi si are la baza django. Acesta poate fi important ca o librarie in mediul python, dar noi l-am folosit in alt mediu.

4.1 Interpretare json in modele de date

Pentru a putea utiliza interfete retrofit pentru obtinerea datelor sub format json si deserializarea json-ului in obiecte java, mai intai trebuie sa creeam modelarea teoretica a acestor date care in cazul nostru nu sunt defapt altecva decat niste clase de date. Avem trei modele de acest tip: FilmData, PlanetData si PeopleData. Majoritatea continand defapt Stringuri si liste de Stringuri. Pentru fiecare model avem cate o clasa specifica care contine din acel json rubrica de rezultate ale interogarii unei paginii de pe un anumit subiect (ex: get swapi.dev/api/people/?page_number=1). Interogarea unui endpoint al unui subiect contine printre chei, o cheie de rezultate care ca valoarea este defapt un array de obiecte de tip Film, Personaj, Planeta sau altele. Noi am decis sa nu punem si celelalte chei in clasa sub forma unor atribute de obiecte deoarece am considerat ca nu ne sunt utile.

4.2 Instanta Retrofit

Clasa RetrofitInstance contine declaria tia unui obiect cu un base_url si o functie care returneaza instante retrofit pentru construirea cererilor in cazul nostru doar de tip get.

4.3 Servicii

Fiecare serviciu are la baza o adnotarea @get("/endpoint") si o metoda getEndpoint care returneaza un raspuns de tipul endpointului ¡Film¿,¡Planet¿ sau ¡People¿ in functie de serviciul necesar.

4.4 Activitatea prinicipala

Activitatea Main, la crearea instantei activitatii principale se initializeaza toate serviciile de realizare a cereriilor, se realizeaza cereriile si ulterior se populeaza listview-urile subiectelor cu datele respective. Activitatea prinicipala are si un list view afisat initial care functioneaza defapt ca un mediu de navigare spre subiectele(sau topicurile) acestui wiki.

```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    val planetService = RetrofitInstance
        .getRetrofitInstace()
        .create(PlanetService::class.java)
    val filmService = RetrofitInstance
        .getRetrofitInstace()
        .create(FilmService::class.java)
    val peopleService = RetrofitInstance
        .getRetrofitInstace()
        .create(PeopleService::class.java)
    val responsePlanetData: LiveData<Response<Planet>> =
        liveData { this: LiveDataScope<Response<Planet>>
            val response = planetService.getPlanets()
            Log.i( tag: "StarWars_InfoTag",  msg: "Planet Service response: ${response.body()?.results}")
            emit(response)
        }
    val responseFilmData: LiveData<Response<Film>> =
        liveData { this: LiveDataScope<Response<Film>>
            val response = filmService.getFilms()
            Log.i( tag: "StarWars_InfoTag",  msg: "Film Service response: ${response.body()?.results}")
            emit(response)
        }
    val responsePeopleData: LiveData<Response<People>> =
        liveData { this: LiveDataScope<Response<People>>
            val response = peopleService.getPeople()
            Log.i( tag: "StarWars_InfoTag",  msg: "People Service response: ${response.body()?.results}")
            emit(response)
        }
```

Figure 1: obtinere instante servicii si ulterior realizarea unor cereri cu acestea

```
@Composable
fun MainScreen(navController: NavController) {
    val items = listOf("People", "Films", "Planets")

    LazyColumn() { this: LazyListScope
        items(items.size) { this: LazyItemScope index ->
            Text(
                text = items[index],
                fontSize = 24.sp,
                fontWeight = FontWeight.Bold,
                textAlign = TextAlign.Center,
                modifier = Modifier
                    .fillMaxWidth()
                    .padding(vertical = 24.dp)
                    .clickable {
                        when (index) {
                            0 -> navController.navigate( route: "People")
                            1 -> navController.navigate( route: "Films")
                            2 -> navController.navigate( route: "Planets")
                        }
                    }
            )
        }
    }
}
```

Figure 2: Meniu Principal

```
@Composable
fun PeopleScreen(peopleList: List<PeopleData>) {
    LazyColumn() { this: LazyListScope
        itemsIndexed(peopleList) { this: LazyItemScope index, people ->
            Text(
                text = people.name,
                fontSize = 30.sp,
                modifier = Modifier.padding(10.dp)
            )
        }
    }
}

@Composable
fun FilmsScreen(filmList:List<FilmData>) {
    LazyColumn() { this: LazyListScope
        itemsIndexed(filmList) { this: LazyItemScope index, film ->
            Text(
                text = film.title,
                fontSize = 30.sp,
                modifier = Modifier.padding(10.dp)
            )
        }
    }        }
@Composable
fun PlanetsScreen(planetList: List<PlanetData>) {
    LazyColumn() { this: LazyListScope
        itemsIndexed(planetList) { this: LazyItemScope index, planet ->
            Text(
                text = planet.name,
                fontSize = 30.sp,
                modifier = Modifier.padding(10.dp)
            )
        }
    }
}
```

Figure 3: Liste cu date in functie de subiecte

```
setContent {
    val navController = rememberNavController()
    val list1= responsePeopleData.value?.body()?.results?: emptyList()
    val list2= responseFilmData.value?.body()?.results?: emptyList()
    val list3= responsePlanetData.value?.body()?.results ?: emptyList()
    NavHost(navController, startDestination = "main") { this: NavGraphBuilder
        composable( route: "main") { this: AnimatedContentScope  it: NavBackStackEntry
            MainScreen(navController)
        }
        composable( route: "People") { this: AnimatedContentScope  it: NavBackStackEntry
            PeopleScreen(list1)
        }
        composable( route: "Films") { this: AnimatedContentScope  it: NavBackStackEntry
            FilmsScreen(list2)
        }
        composable( route: "Planets") { this: AnimatedContentScope  it: NavBackStackEntry
            PlanetsScreen(list3)
        }
    }
}
```

Figure 4: Componenta de navigatie

5 Concluzie

Din pacate aplicatia nu o putem considera functionala intrucat nu am reusit sa obtinem datele de pe api, motivul este legat de permisiunile de utilizare a internetului in cadrul aplicatiei. Totusi in ciuda nereusitei, am invatat tehnlologii si concepte noi utilizate in dezvoltarea aplicatiilor mobile. Astfel am aflat ca orice idee pusa la test poate fi un beneficiu.