PokemonPokedex Application

Filipa Ivanković, Lucijan Pavić

Product Definition Statement

This application utilizes the PokeApi (<https://pokeapi.co/>) to showcase a variety of Pokemon and provide useful information about each one. It offers users the ability to tag Pokemon as "Favorite/Collected" for easy reference and tracking, whether for their personal favorites or as a way to keep track of Pokemon caught in the video games.

In this app, users can explore different Pokemon and learn about their characteristics, strengths, and weaknesses. By leveraging the PokeApi, the app displays relevant information such as each Pokemon's name, type, abilities, stats, “shiny” form, weight, height and more.

The app's tagging system allows users to create their own personalized collection of favorite or collected Pokemon. This feature not only provides a convenient way to keep track of the user's favorite Pokemon, but also serves as a tool to document their progress in catching them in the video games.

Furthermore, the app supports filtering through the vast array of Pokemon with a convenient search function. The Application holds data for every single Pokemon currently in existence, as well as each form for each Pokemon.

Overall, this app offers a fun and interactive way for users to explore the world of Pokemon and learn more about their favorite creatures. Whether the user is a seasoned Pokemon fan or just starting their journey, this app provides a useful resource for discovering new Pokemon and tracking their progress in catching them all.

Features Implemented

* Fetch Pokemon data from external API
* Display a big number of Pokemons and display them
* The Api displays every single Pokemon currently in existence
* Pokemon can be clicked on which will open a individual page for that particular
* Each Pokemon can be clicked on and a detail page with that particular Pokemon information will be displayed
* Pokemon will relevant information about them
* Pokemon’s name, pokedex id, typings, moves, stats, abilities, weight, height, shiny form and more are all available in the app
* Pokemon can be added to the Favorite page
* Working Favorite page
* Pokemon can be removed from Favorite/Collection page
* Extra info can be accessed from the Favorite/Collection page
* Pokemon that are added to Favorite/Collection page will be stored in the cache
* Using a third party library for animations (such as LottieFiles)
* Animation plays when clicking the “Add to Favorites” button

Additional Features

* Bottom Navigation between “Pokedex” and “Favorites” screens
* TopAppBar that holds the button for switching between dark and light mode - it will hide itself when going to the Pokemon Details screen
* Navigate to the next/last Pokemon from the details screen with the press of a button
* Custom app icon implemented
* Search functionality - Pokemon can be searched by inputting their name. Additional forms of Pokemon are displayed if the Pokemon has any
* Dark mode - the entire application supports easy switching between light and dark mode

Description of the code

MainActivity.kt

The entry point to the project. The composable “MainScreen” creates a scaffold for the top and bottom navigation. The Navigation.kt is called within the inner part of the scaffold, which is responsible for displaying the rest of the screens. The default starting screen is the Pokedex screen that displays all the Pokemon

PokemonPokedexApplication class

Nothing of note here. This is just needed for the sake of the DaggerHilt plugin.

Data package

AppDataBase.kt

Abstract class creates the Room database and defines all of the entities.

Api package

Contains all of the auto generated data classes for the json data.

APIService.kt in which we define all of the api calls.

Data class PokedexListEntry.kt to help with reading of the data from the api.

Dao package

FavoriteDao.kt interface with all of the queries to the room database.

Entity package

Favorite.kt data class that defines the favorite table and all of the columns.

Module package

AppModule.kt object that converts the json data into gson data that it got from the endpoints in APIService.

GlideModule.kt is a class that uses AppGlideModule to remove errors with undefined glide modules.

Repository

FavoriteRepository.kt

Class and methods for displaying the data for the Favorites screen. There are methods for reading, inserting and deleting favorite Pokemon

PokemonRepository.kt

The main repository for retrieving data about the Pokemon from the Api. The getPokemonList gets the entire list of the Pokemons, while the getPokemonInfo gets information about a specific Pokemon. The getAbilityDetails is used for the alertdialog on the PokemonDetailsScreen that displays information about that Pokemon’s abilities. Each Pokemon can have up to 3 different abilities.

Favorite package

FavoritesScreen.kt

Function FavoriteScreen that creates the favorite page and sets the design.

FavoriteViewModel

Class that contains all of the functions for interaction with the database. There are methods for reading, inserting and deleting favorite Pokemon. They return functions from PokemonRepository.

Navigation package

Navigation

Composable with the Navhost. Contains the composables for the Navigation routes. Three routes: Pokedex Screen, Favorites and PokemonDetails Screen which is dynamically generated for each Pokemon from the Api.

BottomNavigation.kt

The primary navigation feature of the project. Holds the navigation for the “Pokedex” and “Favorites” screen. It is created in the MainActivity.kt within the Scaffold. Works with the NavItem sealed class to get the route, ImageVector and title of the Navigation item.

NavItem

A sealed class file that holds the navigation objects for the Pokedex and Favorites Screen. Each screen has a route, ImageVector and title.

PokedexTopBar.kt

Composable for the TopAppBar for the Scaffold in the MainActivity.kt. The name of the application and a switch for switching between light/dark mode is contained in this TopAppBar. The “Globals” object class is used to remember the state of the light/dark mode

Util package

Constants object file

Contains two constants: the base url of the api and PAGE\_SIZE which is the total number of Pokemon that are being fetched from the API.

Globals object file

Holds global variables that are used throughout the application. The “darkmode” variable is used to determine whether to display light/dark mode. “showTopAppBar” is used to check whether or not to display the PokedexTopBar.kt. It is set to false when going to the Pokedex Details screen, and true for any other screen.

PokemonDataParser.kt

Multiple methods which transform the typings and stats and link them to their corresponding colors from the “ui/Color.kt”

Resource class file

Sealed class that is used for Loading, Error message handling and checking if the Api fetching was successful or not. Each class returns a Resource<T>

Pokedex package

PokedexScreen.kt

Contains functions PokedexScreen, hwScaffold, PokemonEntry and RetrySection. These functions are used to create the pokedex page. They contain all of the design and functionality of the pokedex page.

PokedexViewModel.kt

The main view model for the Pokedex screen. It is used to parse the pokemon list from the list and dynamically fill out the Pokedex screen with all the Pokemon retrieved.

Searchbar.kt

Search functionality and the search bar itself for the Pokedex screen.

PokemonDetail package

PokemonDetailScreen.kt

The main screen for the Pokemon details. The entire structure of the details view is contained here, while sections of the detail view are contained in other kotlin files that have composables for each section.This kotlin class is the main class that uses the getPokemonInfo models. At the bottom of the detail screen are two navigation arrows. Pressing them will send the user to the next Pokemon in the api (example: Going back on Charmander will lead to Venusaur, while going forward on Charmander will lead to Charmeleon). At the top there is a “Star” button which will swap between the Pokemon’s regular and so-called “Shiny” coloring. The background is dynamically colored, as the colors will depend on the Pokemon’s typing. Depending on the typing the gradient will be different.

PokemonAbilities.kt

The Abilities section of the detail view. Pressing the dialog alert button will pop-up an alert with that ability's information. This information is fetched with an additional Api call as the data for the abilities is contained in another section of the PokeApi. The api call is made when the button is pressed. Various Pokemon have a varying number of abilities, but the maximum number of them is 3.

PokemonDataSection.kt

Displays data about the Pokemon’s weight and height, and also displays the icons for the weight and height.

PokemonDetailViewModel

The view model for the detail screen. Two methods: one for getting the Pokemon’s data, and one for getting the data about the Pokemon’s abilities. There is also a var which keeps track which ability is pressed on the details screen.

PokemonMoves.kt

Provides a clickable box/button which will pop-up an alertdialog that holds the list of all the moves the particular Pokemon can learn and use. Unlike the abilities, these moves are from the Pokemon’s data.

PokemonStats.kt

Displays the total stats and distribution of the Pokemon. Plays an animation when the detail screen is visited.

PokemonTyping.kt

Displays the two types for the Pokemon, and displays the appropriate colors for each type from the “ui/Colors.kt” file. Pokemon can have up to two types.

Above and Beyond

We would say that the most impressive part of this application is that we actually managed to make it work. Third time was really the charm, as it took 3 different implementations of the project to finally get the whole api working. At one point, we even wondered if we should continue using the PokeApi, as we had so many issues with it at the beginning. We did end up having to resort to online tutorials, but even with those, we had a lot of trouble, and we needed to rewrite a lot of the code from that tutorial as a lot of it was deprecated and frankly unusable.

We are very proud of the overall design and visual look of the project, as everything manages to look simple, yet pretty.

Third-party frameworks

LottieFiles

Implementation : “com.airbnb.android:lottie-compose:6.0.0”

LottieFiles are used to display a heart animation when the user clicks on the “Add to Favorites” heart icon present on the PokedexScreen. Other animations in the project are used with Kotlin’s default capabilities.

Retrofit

Implementation: “com.squareup.retrofit2:retrofit:2.9.0

Implementation: “com.squareup.retrofit2:converter-gson:2.9.0

Used for reading the JSON-GSON models, and also fetching the data from the PokeApi.

Navigation

Implementation: “androidx.navigation:navigation-compose:2.6.0-beta01”

Used for the bottom navigation and the ability to navigate to the PokemonDetails screens.

GlideImage

Implementation: “com.github.skydoves:landscapist-glide:2.1.11”

Used to display the images of the Pokemon

DaggerHilt

Implementation: “com.google.dagger:hilt-android:2.44”

Implementation: “androidx.hilt:hilt-navigation-compose:1.0.0”

Plugin that was used in conjunction with Retrofit to fetch the data from the PokeApi. It was used to better map out the extensive Api, as its structure was rather complicated. In the project, DaggerHilt annotation can be found over various composables, classes, objects and methods.

Room

Implementation: “androidx.room:room-runtime:2.4.2”

Implementation: “androidx.compose.runtime:runtime-livedata:1.2.0-beta03”

Used for the Favorites feature to locally store the user’s selected favorite Pokemon.