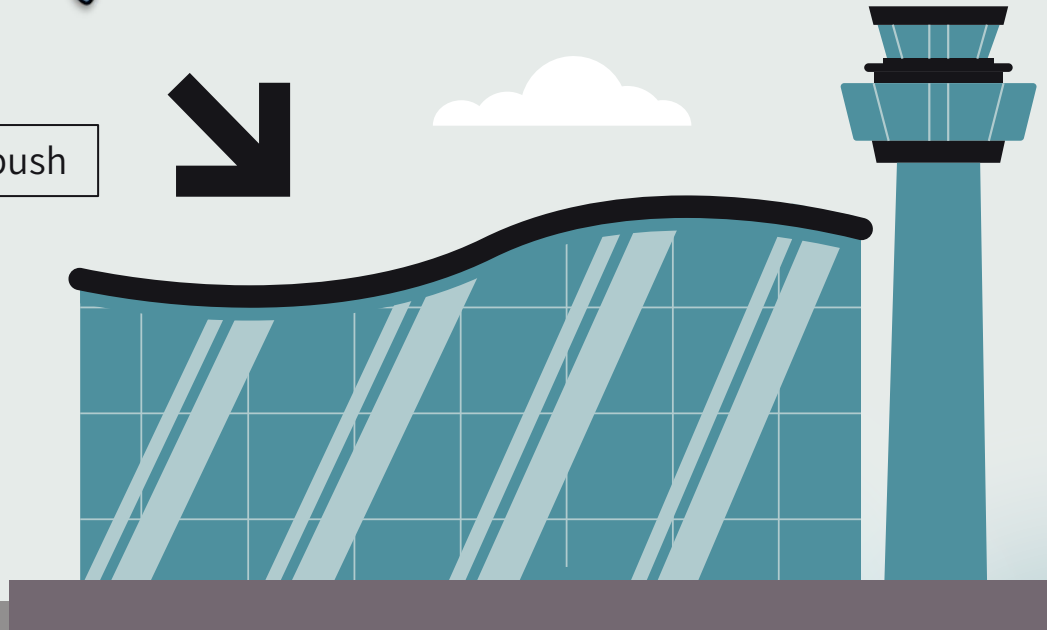
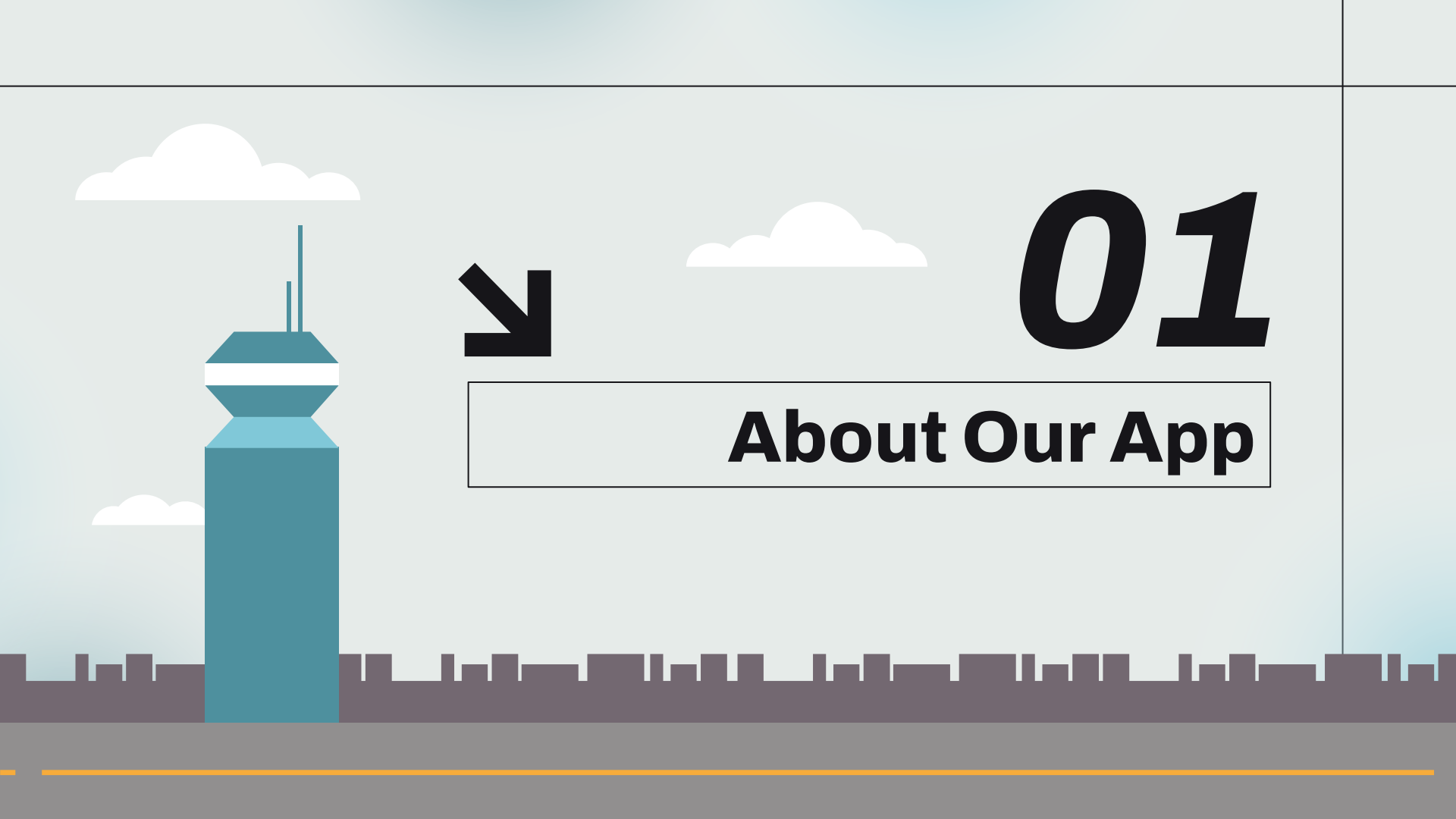
SkyStar

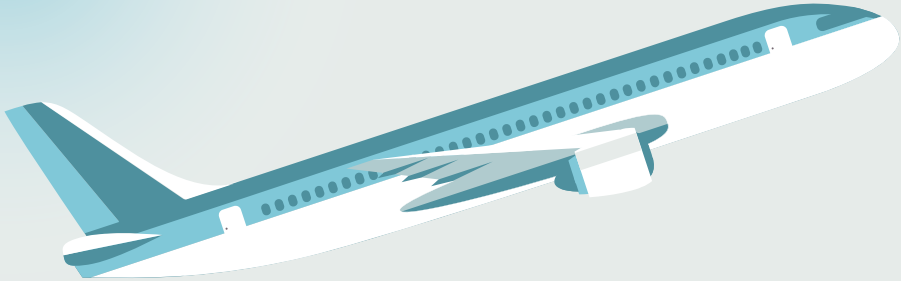
Roi Katz, Aran Ben Ezra & Daniel Dabush





01

About Our App

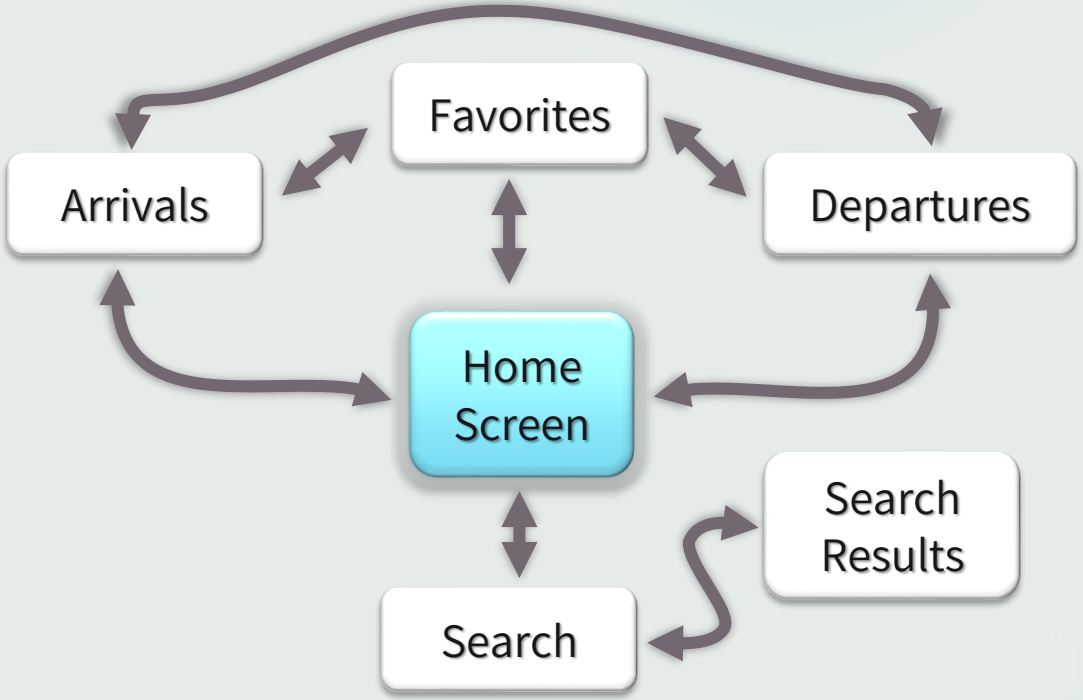


What Is It About

Have you ever had a flight and wanted to know if it was delayed? Or perhaps you waited for your loved one to land back home and wanted to know when to pick them up?
SkyStar provides the user an easy and straightforward way to access the information of upcoming and recent flights at Ben Gurion Airport, Israel – with an option to track specific flights of the user's choice with an up-to-date information

How Does It Look Like?

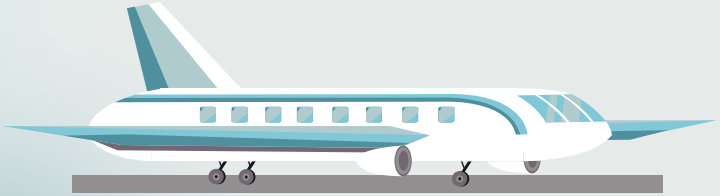
Application's fragment flow chart:



How Does It Look Like?

The users are introduced with a main menu, where they can access the different screens and functions of the app:

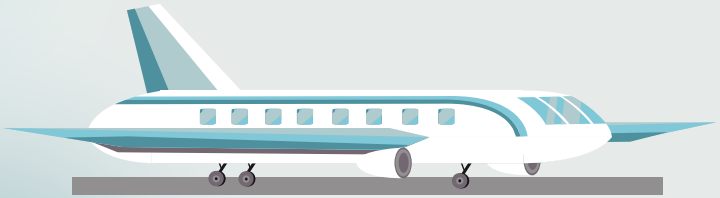
- Departures
- Arrivals
- Favorites
- Search
  - From within a ‘Search Results’ screen is accessed

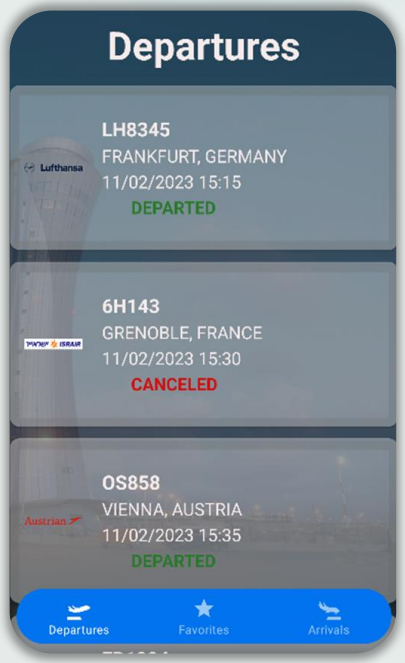


How Does It Look Like?

Departures

Shows the users all departures in the range of 5 days (24-hour prior to loading until 96-hours after loading)

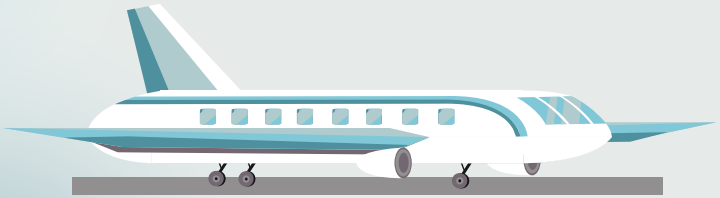


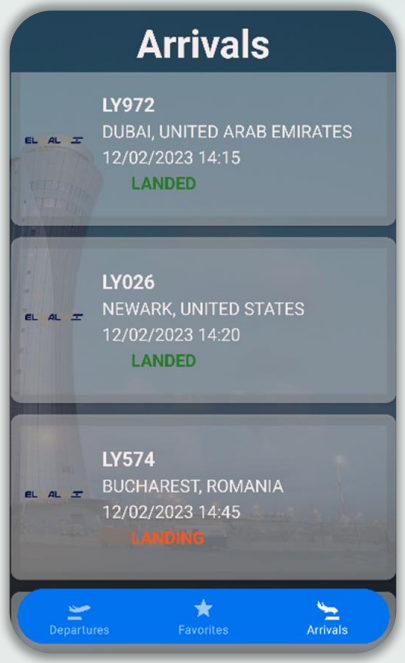


How Does It Look Like?

Arrivals

Shows the users all arrivals in the range of 5 days (24-hour prior to loading until 96-hours after loading)

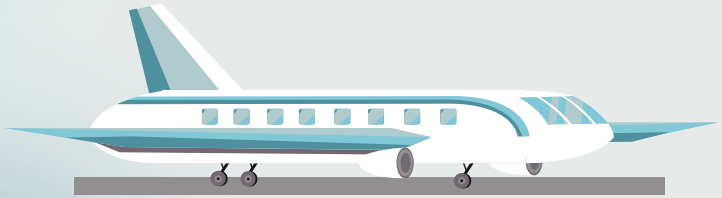


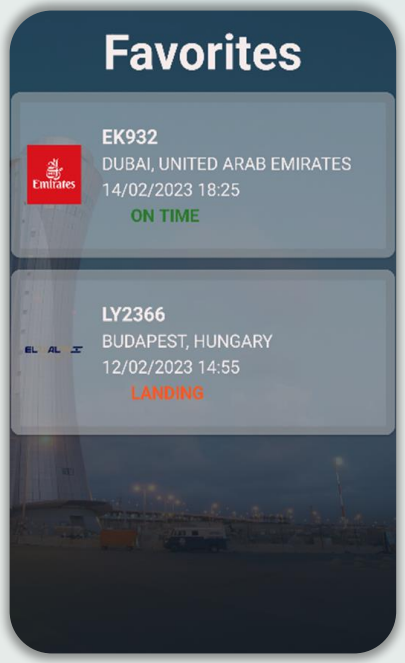


How Does It Look Like?

Favorites

Shows the users a list of all the active flights that they marked as favorites

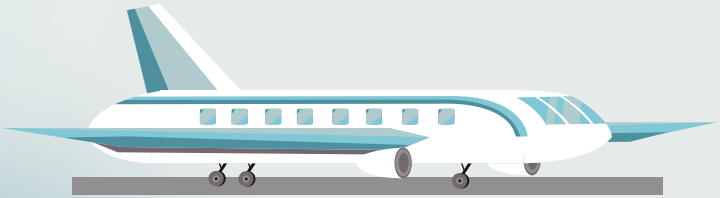


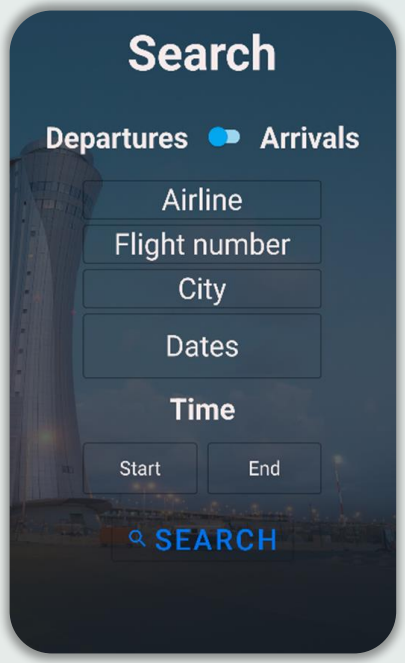


How Does It Look Like?

Search

Shows the users an interface where they can mention various criteria for searching individual flights:
Departure/Arrival, Airline, Flight Number, City, Date Range and Hour Range

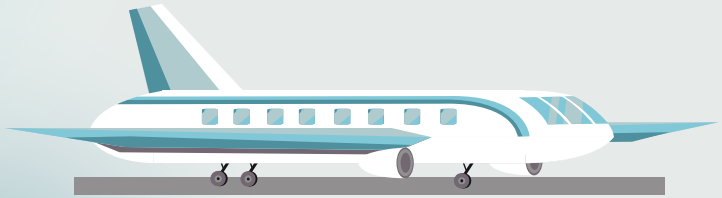




How Does It Look Like?

Flight Details

A dialog box is opened when tapping on a certain flight. It shows further information about the flight: Flight number, Airport, City, Country, Airline, Scheduled Time of Flight and Actual Time of Flight.



How Does It Look Like?

Navigation Bar

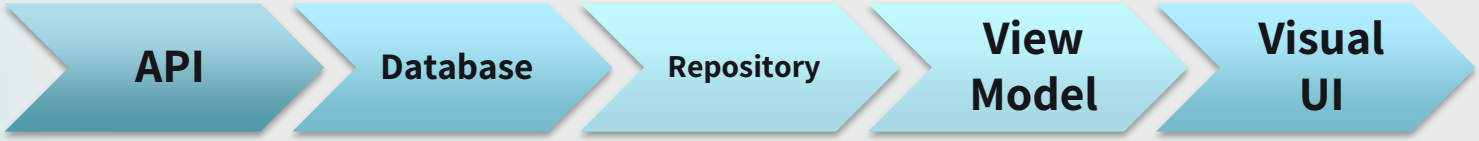
The departures menu, arrivals menu and the favorites menu include a bottom navigation bar that allows the user to easily navigate between the three menus without the need to pass through the home screen

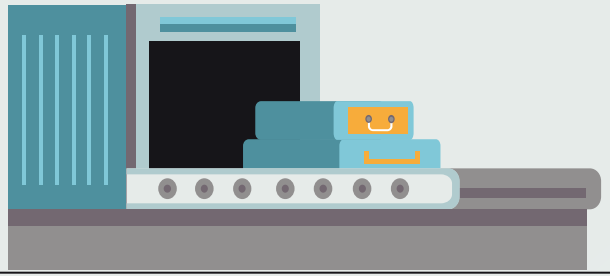
02

How Does It Work

Architecture

The data used for the flights' information in the app is fetched from an online API, managed by the Israeli Government and is available for the general public to use.
The app uses various methods and libraries to fetch, arrange and display the information to the user. The data flows through several stages before being shown to the user:





API

We used the ‘Retrofit2’ library to send ‘GET’ requests to the ‘gov.il’ server.
Because of its free availability, there is no need for a token to receive responses from the API.
There are several queries that the API allows:

- RESOURCE_ID – Identifies the API we want to send the request to, from the many databases the government holds
- Filters – Allows to send filtering queries to the API by a specific criteria, and it will return a response with the corresponding data
- Limit – The number of records the data returns per request

API

For our app's needs, we specified two types of 'GET' request from the API:

- getRecords – fetches data from the API without filters, but since the 'limit' query must be mentioned we specified a limit of 5,000, based on the calculation of the amount of annual flight at Ben Gurion Airport, dividing it by 365 and multiplying the result by 5.
  It is used in the following cases:
  - When we want to filter search results with more sophisticated filters than the API allows
  - When we want to compare the favorites to the most up-to-date data in order to update it

- getFiliteredRecords – fetches data from the API with a single filter. It is used in the following cases:
  - When the user visits or refreshes the departures screen
  - When the user visits or refreshes the arrivals screen

Database

We used the ‘Room’ library to create and manage our databases. We created two databases for the purpose of this application:

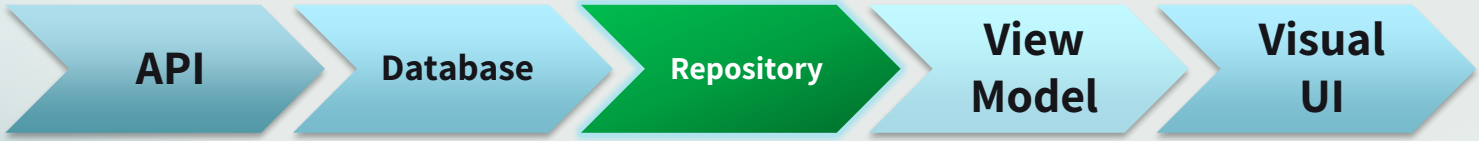
- Records Database – Stores all data received from the API. For every ‘Get’ requests, the database is wiped and the new data replaces it
- Favorites Database – Used to store the user’s favorites. Every time a user adds a favorite, a copy of the record from the previous database is created and stored in the favorites database, so the data there is stored and managed independently.

Repository

The repository is used to manage the function calls to the Data Access Object (DAO) and specifying the database we want to apply the result into. There are several functions that are called from within the repository and they are:

- Getting data from the API (With/out filters)
- Accessing the favorites database
- Adding/Removing a favorite
- Counting the number of favorites
- Check if a flight is in marked as a favorite
- Check if a flight is still active
- Update details about a flight (Specifically, a favorite flight)
- Search flight by various criteria

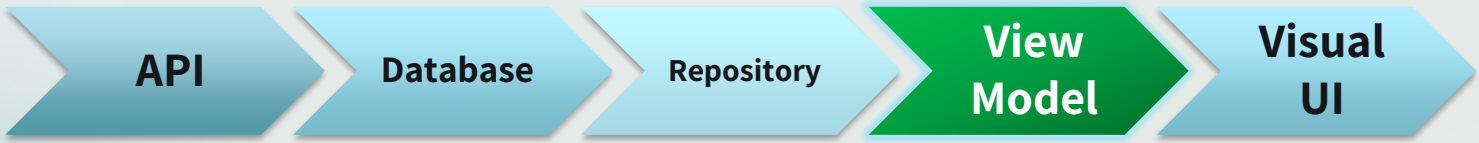
All functions that access databases are called asynchronously in a co-routine, in order to prevent the app from being unresponsive



View Model

The View Model provides an interface to call the functions that are defined in the repository, as mentioned in the slide about the repository.

In addition, the View Model stores the data received from the API in local variables, so the RecyclerView Adapter will be able to access it.



Visual UI

Every UI fragment in the project has an instance of the View Model, so it can access its functions and variables.

Dependent on the need, each fragment calls a different function from the View Model, and if needed (in the case of API calls) stores the results in a variable within the View Model.
That variable is accessed in the respective fragment, where errors are being handled in cases where:

- Server is unreachable (When the user has no internet connection or when the server is down)
- Search results return an empty database
- The favorites list is empty or automatically emptied (When flights are out of the date range)

Visual UI

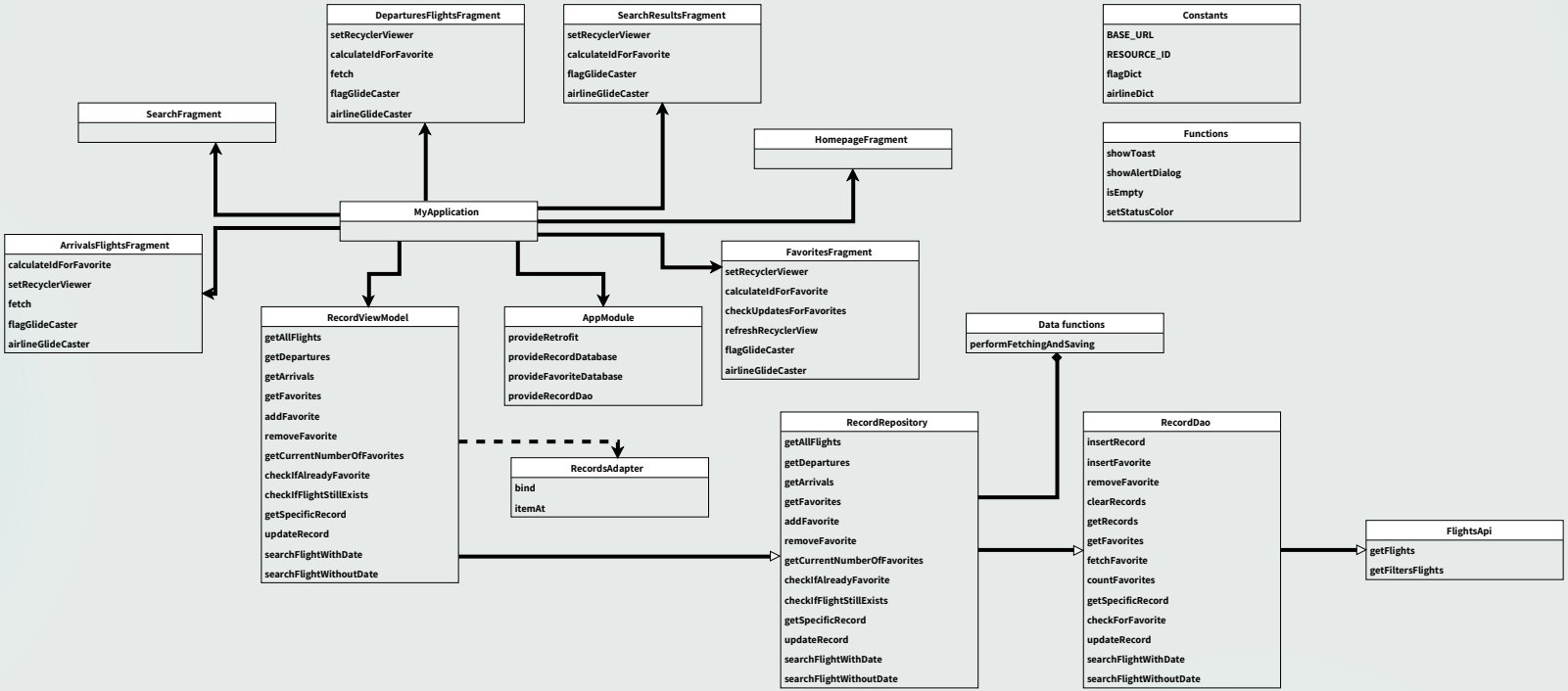
Finally, our general RecyclerView adapter handles the visual representation of the retrieved data.

With the use of the ‘Glide’ library, each entry is able to display the airline’s logo and the destination/origin country’s flag in the entry’s details dialog.

By swiping on an entry from left to right, the flight is added to the favorites database
By making the same gesture at the favorites fragment, the flight is removed from the list.

API → Database → Repository → View Model → Visual UI

Class Diagram



Libraries We Used

Retrofit2

Hilt

ROOM

glide

Thank You For Listening