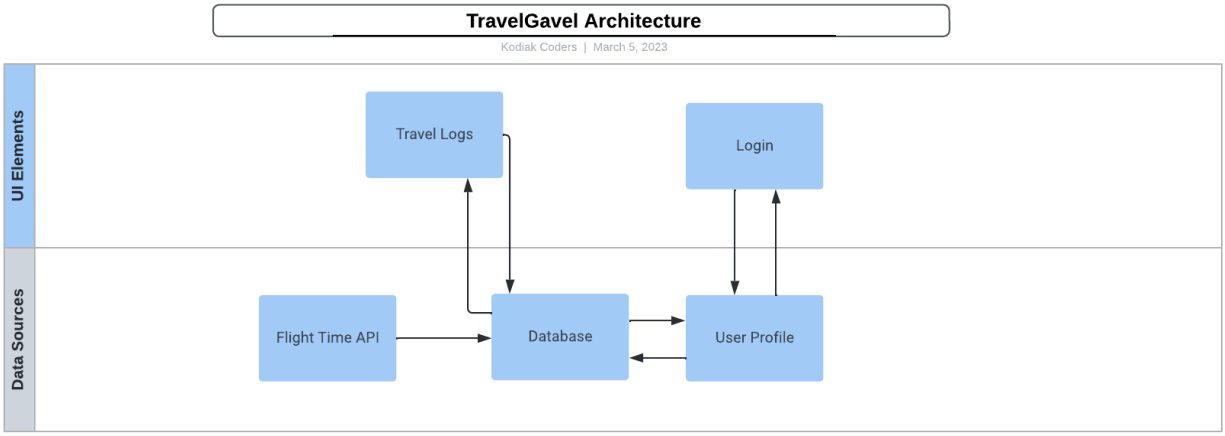
TravelGavel Architecture



Description

Layered architecture is an architecture that models sub-systems and how those subsystems interact with each other, which are called layers. It groups components with similar functionalities in horizontal layers. This results in multiple layers, each with a specific role in the application. This architecture separates the low level aspects of the architecture, such as the database, from the high level aspects of the architecture, such as the user interface. When a layer changes, only layers adjacent to the changed layer can be affected. These layers commonly feature a presentation layer which holds the UI and bridges to the next layer, the business layer which holds the system's core functionality. This layer is connected to the prior and successor layer, the data layer, which accesses data within the system boundaries and any data networked systems expose, as well as generic interfaces for components in the business layer. Layered architecture is commonly used in Android applications. When used in this context, the layers should at least include a UI layer, that displays application data on the screen, and a data layer, that contains the business logic of the app and exposes application data. There is an additional, optional layer that is a domain layer, to simplify and reuse the interactions between the UI and data layers.

Justification

Layered architecture is the recommended architecture for Android applications. The principles guiding this recommendation are: separation of concerns (UI layer should only contain logic that handles UI and operating system interactions), drive UI from data models (persistent models mean users won't lose data if Android OS destroys the app to free up resources and the app continues to work in cases when network connection in unstable or unavailable), single source of truth (centralize changes to a particular type of data in one place, protects the data so that other types cannot tamper with it, and makes changes to the data more traceable), and unidirectional data flow (data should only flow in one direction to guarantee data consistency). Additionally, redundant facilities can be provided in each layer to increase the dependability of the application. This can be used to increase the security of the application, which will be important if the users' store their payment information in the applicatThe organized nature of layered architecture also makes the development of the application easier, as it allows for functionality to be divided into purposeful layers. It makes it easier to divide work between different teammates when asynchronous work is required. There is also the potential to modify specific layers if needed or include certain aspects in multiple layers to ensure functionality. This makes the application more resilient to change, which will be helpful, as we are new to software engineering and are likely to need to make many changes as the application is developed.