Software Architecture & Design Pattern Document

Software Architecture

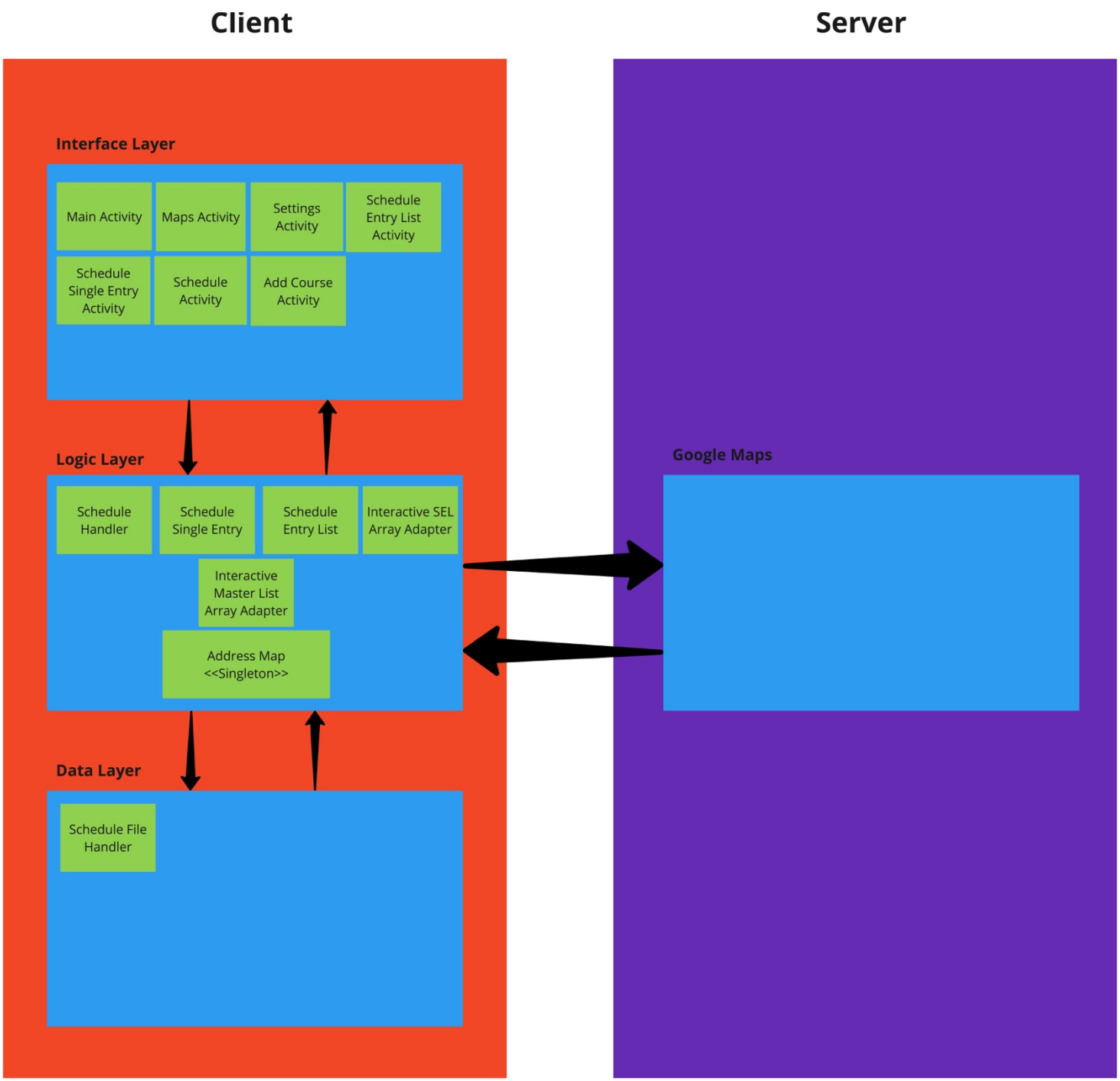
The software architecture for the *RaiderNAV* app is based on a simple hybrid of the layered and client-server architectural patterns as detailed below.

The *RaiderNAV* software located on a target Android device is itself comprised of three layers. The first is an interface layer, composed of Android activities, which facilitates communication between the user and the underlying system. Next is a logic layer, implemented in the form of Java classes, which interprets user input and processes it according to the app's design. Finally, at the lowest level is a data layer, similarly formed by Java classes, that is responsible for manipulation and storage of processed data. True to the layered pattern, the interface and data layers interact directly only with the logic layer. The logic layer, in the middle of the three-tiered design, interacts with both other layers.

Altogether, these three software layers on the device constitute the client component of the encompassing client-server model. The server module of this model is represented in the form of the backend functionality of the Google Maps API, which is necessary for navigation and location features. Communication between server and client components is limited to the logic layer of the client-side implementation. The functionality of *RaiderNAV* is dependent on both the user interaction and information processing done on the client-side as well as the data retrieved from the Google Maps server.

Design Patterns

Several design patterns make appearances throughout *RaiderNAV* codebase. For example, many of the activities included in the interface layer can be considered examples of the observer design pattern. They present information saved on an Android device in a variety of ways and are decoupled from the information itself. The AddressMap class represents a singleton that holds a single, static copy of information that is used throughout the app. As their name suggests, both the InteractiveSELArrayAdapter and InteractiveMasterListArrayAdapter classes are representative of the adapter pattern and transform underlying information into a format usable by display elements. Finally, the ScheduleSingleEntry and ScheduleEntryList class are examples of the composite pattern. Each one encapsulates mulitple data items into a single class which is treated as single unit for use throughout the app.


Client
Server
Interface Layer
Main Activity
Maps Activity
Settings Activity
Schedule Entry List Activity
Schedule Single Entry Activity
Schedule Activity
Add Course Activity
Logic Layer
Schedule Handler
Schedule Single Entry
Schedule Entry List
Interactive SEL Array Adapter
Interactive Master List Array Adapter
Address Map <<Singleton>>
Google Maps
Data Layer
Schedule File Handler