Finance Manager

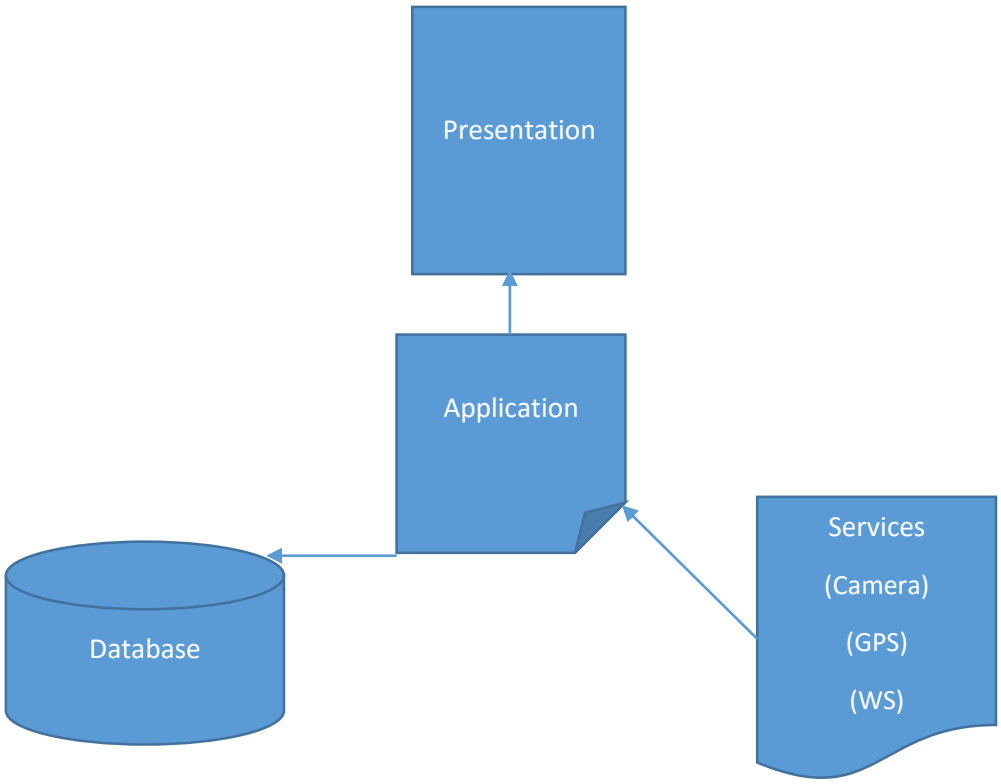
Pedro Nájera, Qiang Wan, Yuanyuan Ma

# Design Document

This document explains the design decisions made for the Finance Manager Android application project. Firstly, the class diagrams will be shown and explained. Secondly, the database schemas will be shown and discussed. Lastly, a brief description of the connection between Server and Client will be discussed.

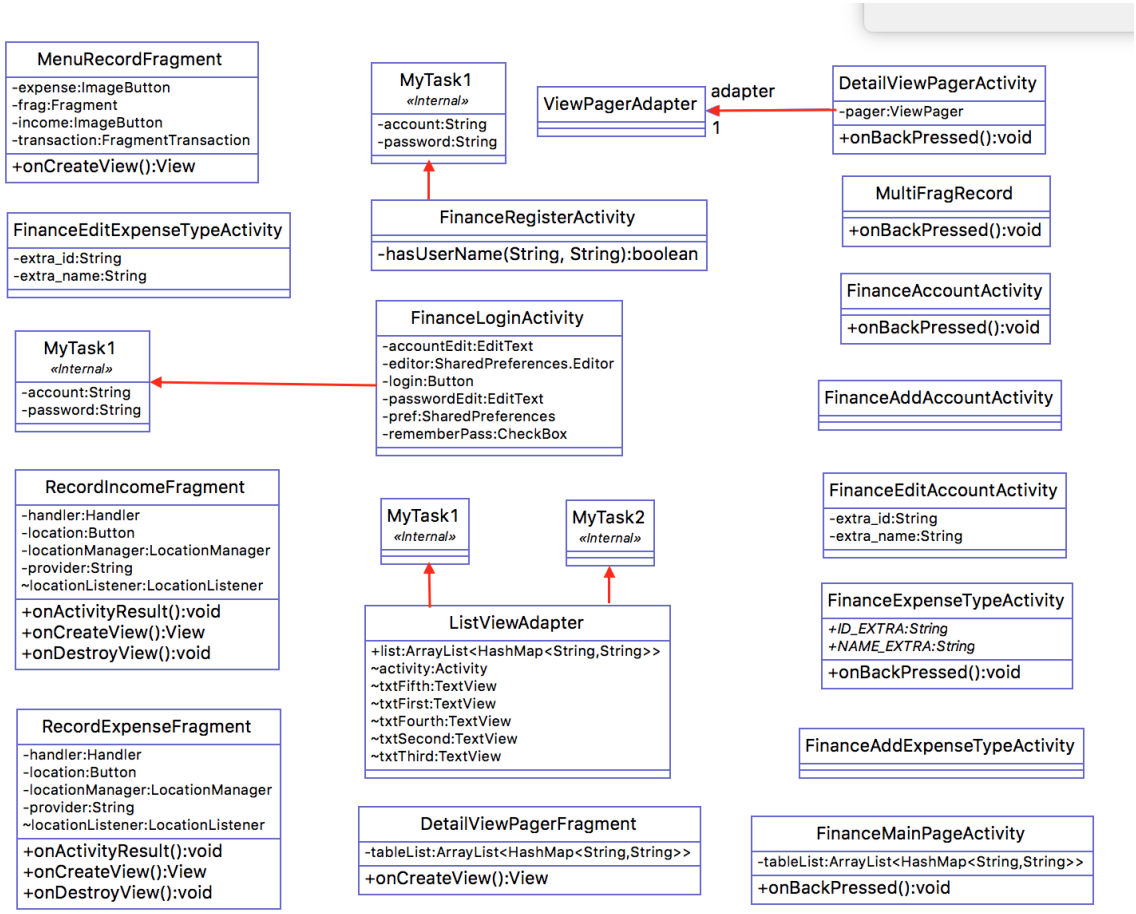
## Class Diagrams

Given the magnitude of Finance Manager, it would be unfeasible to show a single class diagram containing all of the classes used. For this reason, class diagrams were separated using the following model.



## Presentation

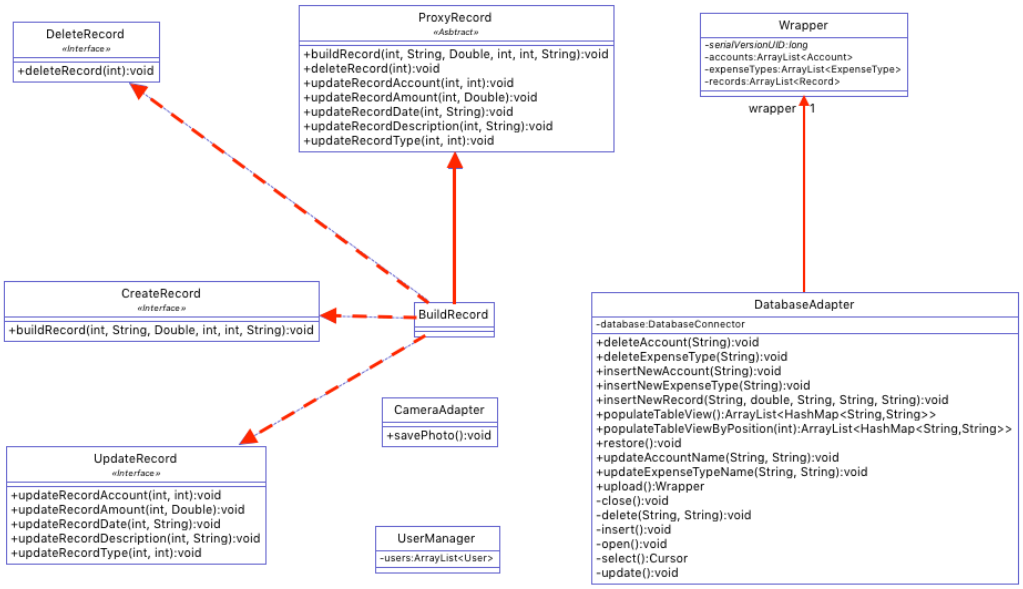
The following figure portrays the class diagram of the presentation layer.

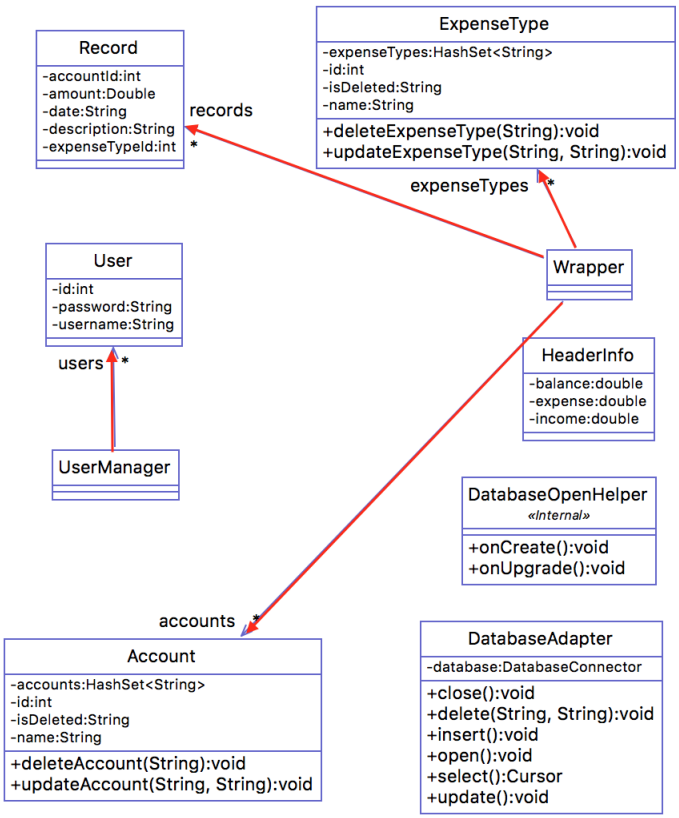



It is visible that this classes are not linked; this is not an error, simply the classes are not instantiated in another classes but rather linked by an Android intent, which are not part of the class diagram design. The Presentation layer interacts with the services using the Business/Application layer.

## Application

The following figure represents the class diagram of the application layer.

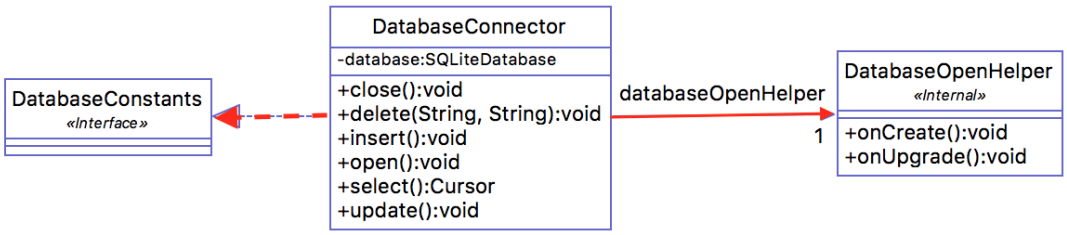


The first thing to comment is that there are two packages here, the first one is the Adapter package and the second one is the Model package. The model package is where the objects are designed and object methods (getter and setters) are used. The adapter package is where most of the logic is implemented and it serves like a bridge between the UI and the services. This is the largest diagram, since it is where most of the work (logical and transactional) is performed. The application layer, also called business layer, is where all the logic and workload should be done. This layer can be further abstracted (as done in the Android packages) by model, adapters, etc.

**Database**

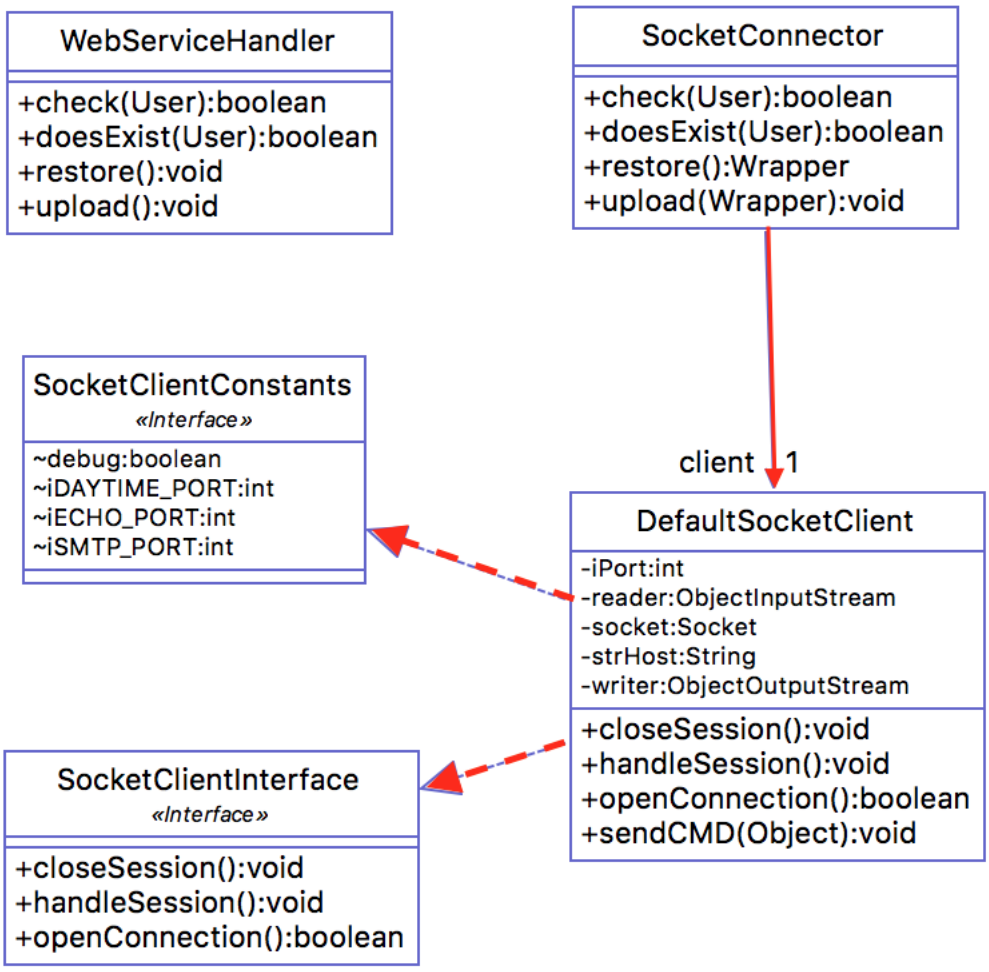
Even though Database is considered a Service, we thought that it is a big part of the implementation and decided to put it in a separate sub-section. The class diagram for the database is as follows.




The most noticeable thing about our design is that the DatabaseConnector class is compact and only contains specific methods to access the SQLite database. The class contains a single select, insert, delete and update methods. Abstraction of this method is done using the DatabaseAdapter class shown in the previous section.

**Services**

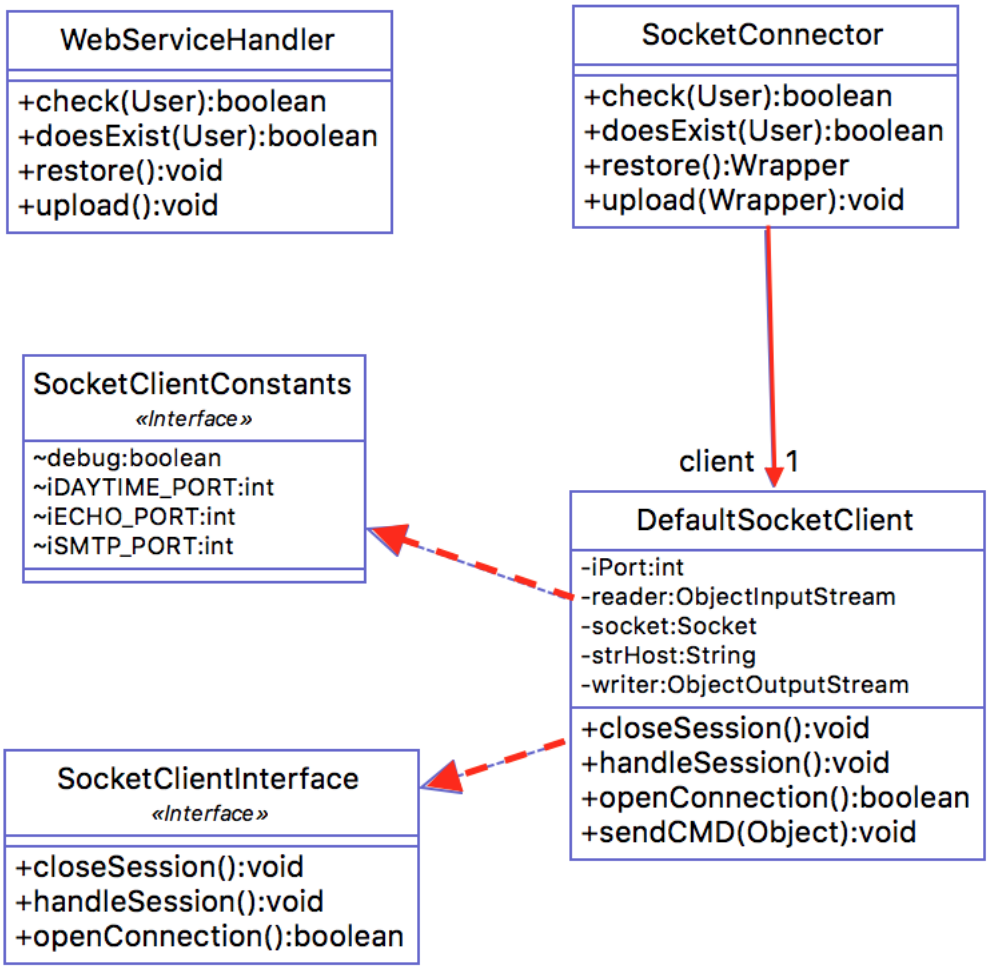
In this layer, the communication with the Server is implemented. Hardware services (as well as the previously discussed Database layer) can be put in this sub-section as well. The following figure is the class diagram for the service layer.




Communication with Server is done in this layer. Either requesting or uploading data is done in this abstraction layer.

**Exception**

With every major application, an exception package is quasi mandatory. Exception handling is located at every layer; each layer will use a different class in this package and ultimately use the appropriate fix method to prevent the application from crashing. The class diagram for this package is shown below.



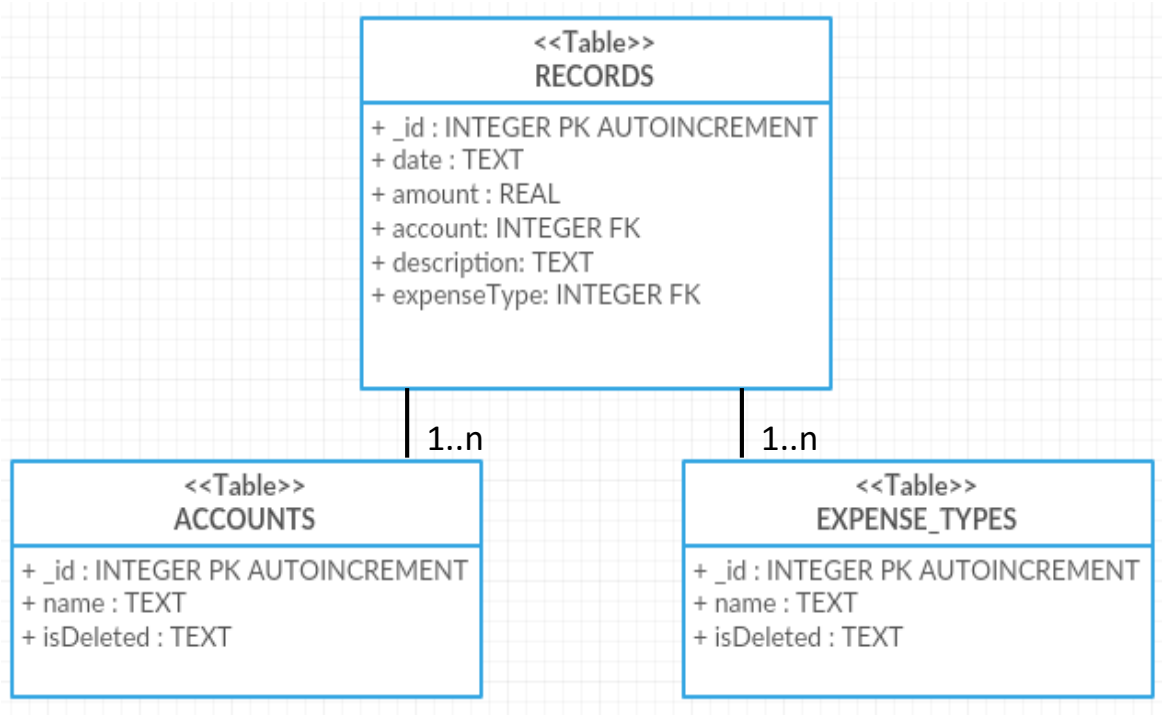
## Database Schemas

As explained before, Finance Manager consists in two parts. The client (Android device) can upload data to the Server and even restore (request) data from the Server. As part of an important feature of Finance Manager, data is stored in the Server and in the Client.

Even though there database schema of both Server and Client are similar, they have its subtle differences; this differences will be explained hereafter.

### Client DB schema

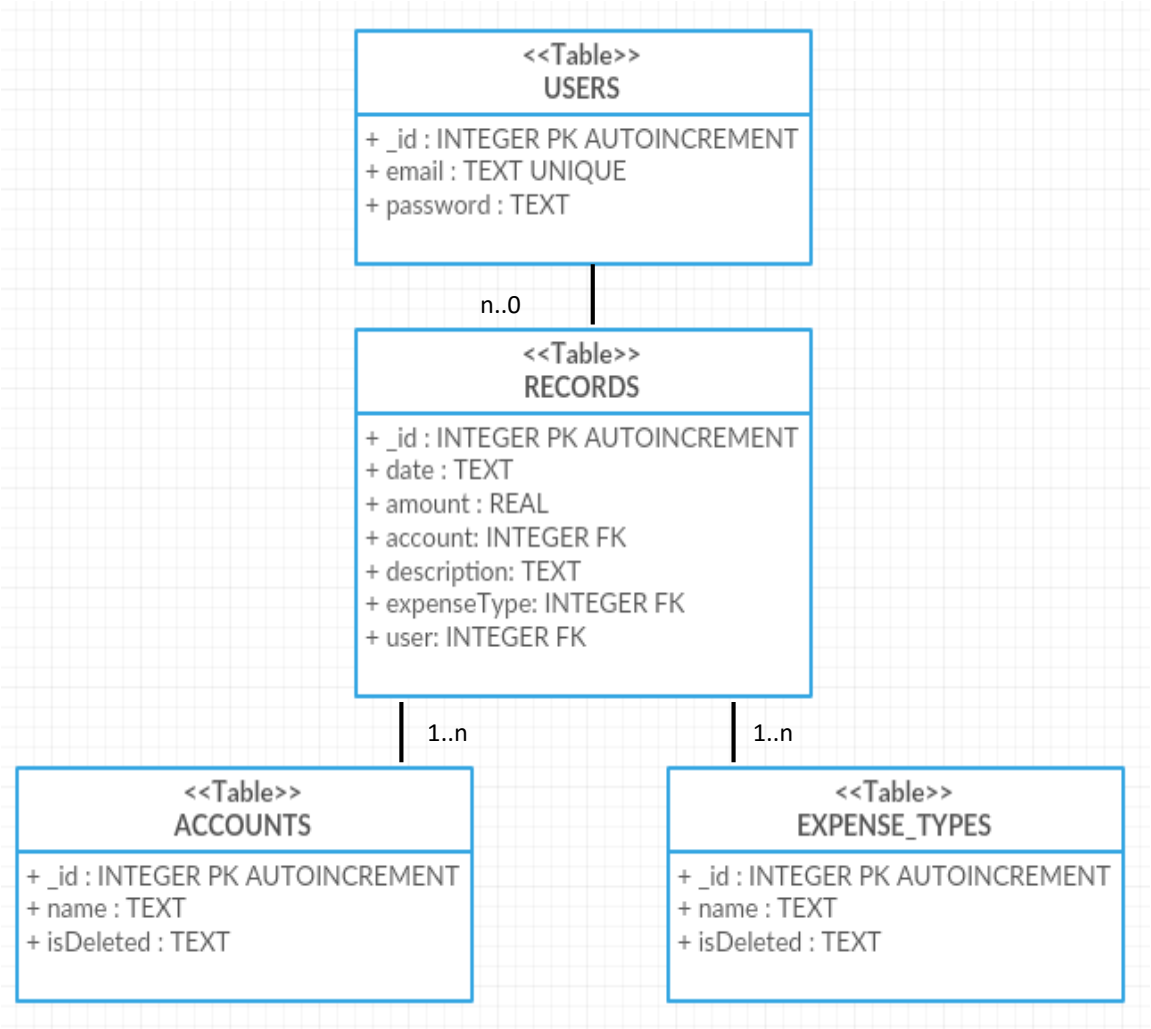
The following figure depicts the database schema used by the Client.




The RECORDS table is the main table and is responsible for storing the heart of the application. The table contains appropriate columns that are relevant to the application user. The ACCOUNTS and EXPENSE_TYPES tables are used to store the customizable accounts and expense types. In order to accommodate normalization in the database, the RECORDS table stores account and expense type as an ID which is the foreign key to the other tables, this way normalization is enforced.

**Server DB schema**

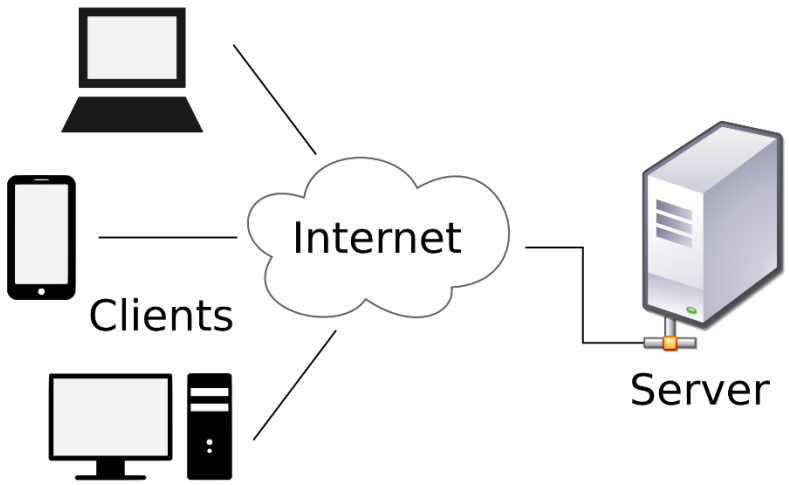
The following figure depicts the database schema used by the Server.




The most noticeable thing here is that the schemas are very similar. This is because they both store the same information. The main difference is that the Server stores data for N users so it is necessary to have a USERS table to store every user as part of the registration process and also each record is identified to correspond to a user, so a “user” foreign key is necessary in the RECORDS table.

## Server-Client interaction

Finance Manager implements a Server using Java in a remote (static) machine. Interaction between Server and Client device was done completely by us, meaning no plugins or third party resources were used. The implementation was done using Java sockets and a direct Server-Client connection was used as seen in the next diagram.




The Server supports multiple clients and is able to serve all of the concurrently. Interaction with Server is transparent to user and the application does not need Server interaction to perform the main tasks; only certain features use the Server.