View

HostActivity

AddTransactionFragment	TransactionListFragment
BudgetEditFragment	ManageCategoryFragment
HomeScreenFragment	OptionsFragment

Controller

HomeScreenController	OptionsController
BudgetEditController	ManageCategoryController
TransactionController	

Model

| BudgetEditModel | ManageCategoryModel |
| HomeScreenModel | |

Account

| Transaction | Category | BudgetItem |

Database

| DataAccessor | DataOpenHelper |

Utilities

Constants

Settings

Settings

Application Structure explanation:

The application is structured around a model-view-controller design pattern. The activities and fragments acts as views where the graphics are displayed and user input is handled. Each fragment owns its own controller and model. The controller receives relevant user input and uses it to change the model. The model then stores all data in the database. When the model is updated, the view is noticed and updated accordingly.

Activities and fragments:

HostActivity is the host of all fragments, it uses an action bar with tabs that links to its respective fragment. The HostActivity keeps its current fragment in a variable and when a tab is clicked the HostActivity switches to the clicked tabs corresponding fragment.

The fragments kept within the HostActivity fetches data from the model and displays it, usually in lists. Most fragments has their own controller that speaks with its corresponding model.

Controller:

The Controller recieves user action calls from its fragments, and calls the appropriate methods in the model to perform the expected behaviour.

Model:

The model package includes both seperate models for each fragment and specific model classes that represents data in the database. The fragment model keeps track of its fragments state, gets commands from the controller and updates its model classes.

The model classes consists of Transactions and BudgetItems that has Categories. These objects are held and by an Account.
Account is the main model class that keeps lists of Transactions, Categories and BudgetItems that it sets and gets from the database.

Database:

The application uses Android's built in SQLite database access. Database tables are created at the first launch of the application.

The tables are constructed as the diagram below.
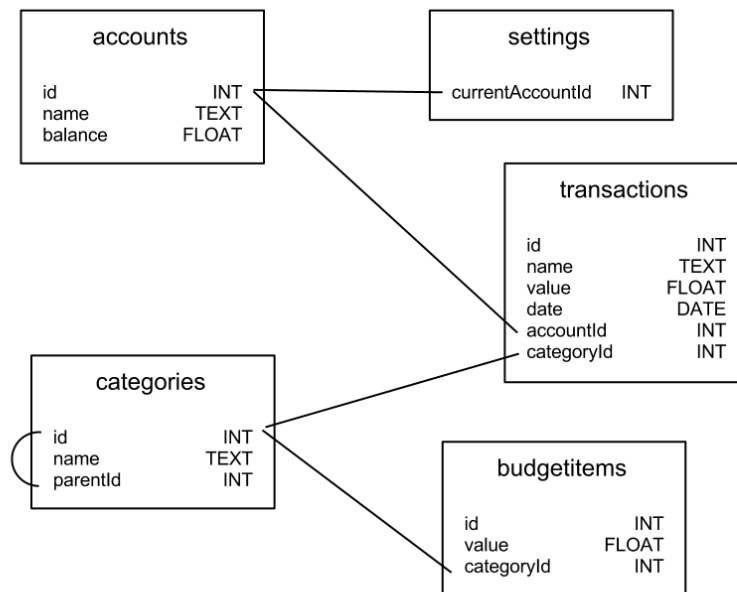
Relationships are made by providing a unique id to each database table item which can be referenced on other table items. Such as "transactions" which has "accountId" and "categoryId" which represents the "id" fields in "accounts" and "categories". These will be connected when running commands to the database.