Classes

# Client

## Utility classes

Most of these classes are a collection of static functions used throughout that app.

* **BitmapUtils** – contains function(s) for dealing with bitmaps
* **GlideUtils** – contains function(s) for working with Glide (image processing library)
* **InputValidationUtils** – contains function(s) for validating user input using Regex patterns
* **ListUtils** – contains some custom utility functions and various high level list functions that aren’t available for the minimum SDK the app can be deployed on as such needed custom implementation.
* **UIUtils** - contains function(s) for dealing with UI components such as getting cleaned text from a text field and hiding elements.
* **RequestCodes** – this class contains codes for the activity result and permission result requests the app will make.
* // todo – move this to the ui package.

**AppScreen** – an enum containing the different screens that the app can show. Each of these corresponds to a specific fragment which will be hosted in the single Activity that the app contains. Each enum member contains an instance of its corresponding fragment. These fragment instances are used for navigation within the app.

## Local Data Stores

These classes contain functionality of saving and retrieving local data. Some of them deal with persisted data and some only save to RAM.

* **DataSets** – a singleton class that deals with data kept only in RAM. It includes a reference to the currently logged in user (of class User) and a list of all categories products can be assigned to (list of class ProductCategory). Both these values are received via a login response (through 3 different endpoints). The references to these objects never change, only the internal members do.

This is used throughout the app to get user information and to display list of categories.

* **InitializedStore** – a superclass containing the infrastructure for singleton classes that need initialization before use.
* **TokenStore**, a subclass of InitializedStore – this singleton class deals with persisting the jwt token received via server request to SharedPreferences.

This token is used in most server requests and received from most server responses (only request/responses where the token is unknown will leave it out such as registering)

As the token may refresh at any request (as a security measure), it is returned in server responses and persisted locally.

This token is used to identify the user for most server request and allows such things as automatic login for a user who has logged in in the past and hasn’t logged out.

* **CartTransactable** – an interface describing the basic functionality of the cart.
* **CartDB**, a subclass of SQLiteOpenHelper and implements CartTransactable – this class deals with persisting the cart to SQLite store locally. It saves pairs of product id’s and quantities for them. These pairs are described in the class CartProduct. This class contains various standard database transactions such as insertion, updating, deletion and selection.
* // todo – perhaps make the two above package access
* **CartStore**, a subclass of InitializedStore and implements CartTransactable. This singleton class acts as the public API for the cart transactions for the rest of the app. It holds a reference to an instance of CartDB and performs the various operations through it with some adjustments which increase ease of use.

## Generic Events

These classes allow for defining event with any type of fire method. These classes are used to listen to activity request and permission request results.

* **Event** – this class describes a generic event and contains a list of listeners to said event (of some implementation of EventListener). In addition, a generic fire method is provided which notifies all listeners.
* **EventListener** – an interface describing the minimum for being an even listener. It contains a fire method, which the observed event will call when it fires.
* **OnActivityResultListener, OnRequestPermissionResultListener** – two interfaces that extend the EventListener interface. Each provides a method corresponding to the life cycle event methods onActivtyResult and onRequestPermissionResult belonging to an Activity class. The single Activity class that this app contains, ParentActivity contains Event instances for each and first each even accordingly. These are used in the app when requesting permission for storage and/or camera access and when receiving the result of choosing a picture from storage/taking a picture via the camera.