Introduction

We implemented a simple application to search for food online using Lifesum API. Besides searching, this application also allows the user to store foods for offline access.

Using the application

By opening the application, the user is presented with a text field and a “Search” button. After a search is completed, the user is presented with a list of the foods returned by the server. Each item has a “Save” button and an “Info” button on its right side. By clicking on the Save button, the food is saved for offline access. Clicking on the Info button will show more information about this food.

By pulling the “application drawer”, the user can access the other app section: “Saved Foods”. On this screen, the user can check the foods that he chose to store offline. By clicking on the “Question Mark” button, the user will be taken to another screen with more information about this food (for simplicity's sake, just a subset of the information returned from the server is shown).

Design and Implementation

We chose a very simple design using a DrawerLayout with two views: “Search Foods”, where the user can search for foods online, and “Saved Foods”, where the user can check the foods previously saved. To maintain compatibility with older Android versions, we chose to use the compatibility library provided by Google.

Each screen consists of a ListView with several food items. To make the ListView behave as desired, we chose to implement a custom Adapter, which gives us better control and customizable behavior. We use the same custom adapter and the same layout for the two lists, changing the behavior slightly according to the list in question.

For the icons, we used the Android Action Bar Icon Pack, provided by Google, which provide a familiar look and feel for the user.

All network operations are ran on a separate thread, to avoid ANR problems. The data from the database is also loaded asynchronously with a CursorLoader.

To store the offline data, we used a simple SQLite3 database with Androids APIs. We also implemented a simple ContentProvider to make access easier by the ListViews.

The Activity designed to show more information about the food behaves in the same way, whether the data is stored on the local database (permanent) or was just read from the server (in this case, data is saved temporarily in memory).

Limitations

Due to the limited time available to design, develop and test the application (6 hours), this application contains some limitations and simplifications that should not be made on a real, market-targeted app:

- **Connection handling and checking:** the application doesn't check if the user is online before querying the network.

- **Error handling:** some possible error scenarios are not considered (for instance, server offline, unexpected response, errors while writing database, etc).
- **Test on real devices:** this app was tested on two real devices (Galaxy Nexus and Nexus 7) and a few emulator configurations. A real app should be tested on several kinds of devices.
- **Layout simplifications:** due to lack of time, the layouts designed are very minimalistic and could be improved to provide a better user-experience.
- **Hard-coded values:** some values (like layout margins) are hard-coded on XML files. These values should be declared on a separate XML file and customized according to screen resolution and so on.
- **Unimplemented features:** some important features were not implemented (for example, removing foods from the database).
- **Screen rotation and configuration changes:** activity state is not saved in this case and could be reset. The correct implementation should be made on onSaveInstanceState( ) method.
- **Input verification:** it would be wise to check for potentially malicious search parameters before querying the server but, for simplicity, this is not done here.
- **Unused information:** some information returned from the server is not used in this application, but, to demonstrate the database behavior, everything is stored locally.