Edinburgh Napier UNIVERSITY

Coursework Report

Ben Parker
40284706@live.napier.ac.uk
Edinburgh Napier University - Mobile Application Development (SET08114)

Abstract

This assignment was to design, develop and test a mobile application for the android platform, within the allotted time frame. The app needs to be testable, so in my case a sandbox version, non-reliant on production data was required. As we were given the freedom to choose our own app, my app is a Customer Relationship Management app, to aid salesman in their daily schedule.

Keywords – Mobile, App, Coursework

1 Introduction

The mobile application (referenced app throughout) I have created is a tool which is aimed towards salesman within a company for to handle all customer relationship management from activities with the customer, to . Although designed with salesman in mind, this can be utilised by anyone who needs to log their daily activities and work with a team.

My inspiration for this idea came from my place of employment, where I work with salesman on a daily basis who struggle to manage their schedules efficiently. This app was designed the issues I have heard them talk about in mind like : activity logging; customer information; communicating with other reps; and effective communication with customers.

2 Software Design

In order to make the interactions with the database as simple as possible, I create a model class for each database table. This also helps avoid duplicate code. The ERP System which I want this app to integrate with consists of nearly 1000 tables, not all of them relevant to the app but I feel like this design could scale to any level.

The design follows the following structure :

All model classes extend DBObject **IF** the app needs to update/insert records in the local SQLite database. DBObject implements the 'ISaveable' interface which declares a method *save()*. Since DBObject is abstract then all model classes which extend DBObject must provide an implementation for the save() method. I have also provided a method declared in DBObject *fromJson(JSONObject)* which these models must implement as well. This is currently part of the design so I have included it, but not given it a body yet, as this is the sandbox version of the app and does not connect to the web-service.

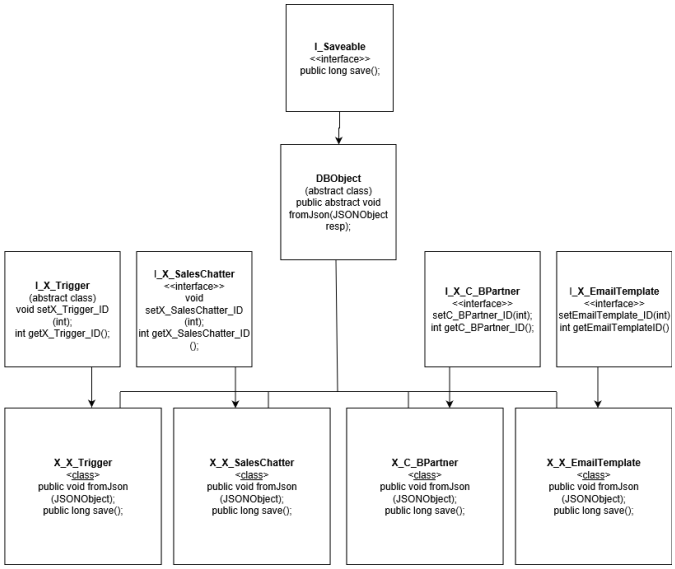

Figure 1: **Class Diagram** - Classes representing database tables architecture

2.1 Database Creation

In order to create the database on load of the app I have created a class **DatabaseCreator** which is launched on load of the **LoginActivity**. The table creation SQL scripts are created in the model classs linked interface *(see Figure 1)* . So XXTrigger implements an interface IXTrigger. Then we can create the database by calling the table creation script from each interface :

Listing 1: Database Creation method

```
1  void onCreate(SQLiteDatabase db) {
2      db.execSQL(I_X_C_BPartner.tableCreationSQL);
3      db.execSQL(I_X_LoginDetail.tableCreationSQL);
4      db.execSQL(I_X_Trigger.tableCreationSQL);
5      db.execSQL(I_X_Action_Purpose.tableCreationSQL);
6      db.execSQL(I_X_Action_Status.tableCreationSQL);
7      db.execSQL(I_X_EmailTemplate.tableCreationSQL);
8      db.execSQL(I_X_SalesChatter.tableCreationSQL);
9  }
```

Important Note Please note that the classes required to interact with the ERP web-service are present in the project. These are not in use in the sandbox version of the app.

Figure 2: Home Page

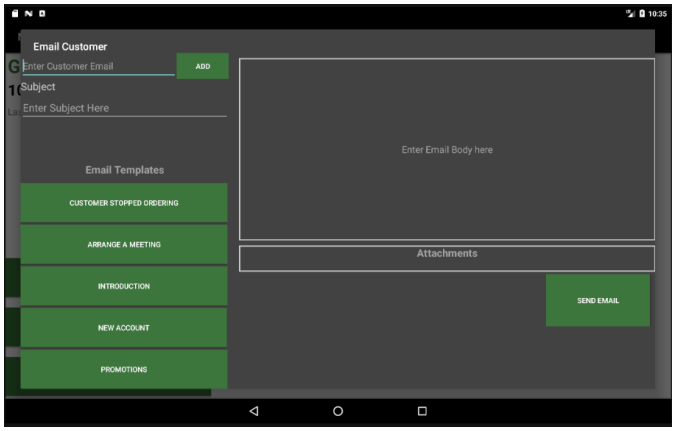

Figure 3: Email Activity

3 Application Implementation

Figure 2 shows the home page of the application. This is the main menu of the app, which allows the user to choose which feature they wish to use.

The graphs displayed on this home page contain test data since the creation of real data (which is derived from historic customer orders / products purchased etc) would require a model for each table etc, and would not suit the scope of this project. Once real data is available these charts will be customisable by the logged in user.

Figure 3 is a screenshot of the Email Activity which the user can access via the Trigger Schedule or the Main Menu. If the user loads from the Trigger Schedule (after selecting a customer) then the email page will load up with the customer email address and name already populated.

The templates buttons on the left allow the user to pre-populate the email body and subject with a message to the customer. I have covered the main scenarios for customers as example data. In future versions of the app the email templates would be customisable by the user.

On click of the send email button the app loads the devices default application for sending emails.

4 Critical Evaluation

My first design of this app was very far out of the scope of this project. I had planned a lot of other features such as : Amazon Lex integration to respond to user queries : 'Give me Aultbea Hotels top 10 products' would trigger an AWS Lambda function to return the data to the production database table, and then read by the web-service and displayed on the app.

These features would of been excellent, and in future versions of the app I hope to expand into these areas, as they are the features that would set this app apart from other similar CRM apps. However I had to scale my design accordingly.

4.1 Competitor App Comparison

This specific market has a number of fortune 500 companies owning the majority share (Microsoft / SAP). Having seen their current mobile apps, which are excellent, they are definitely more feature rich than the current version of this app. However they are not on the bleeding edge of 2018 technology. They do the basics very well with regards to logging activities and customer info.

It is my belief that there is a lot of room for improvement in this area, and although SAP have scratched the surface with their voice assistant, there is more to be achieved.

4.2 Next Release Improvements

In the next version of the app I want to add the following features :

1) Amazon Lex voice assistant for basic queries
2) Activity to add Leads (Most Salesman do not have privileges to update customer records so needs to be leads)
3) Redevelop app to use production data so more complex charts can be made.
4) Orders should be able to be created on the app and synced to the ERP
5) Add Competitor Information activity, which would allow the user to log a customers competitor information
E.g. Walk into a new customers store and see products delivered by another supplier - Take a photo and log it against the customer

5 Personal Evaluation

Over this course the main thing I have learned is that design is more important than I initially gave it credit for. As the lions share of my programming experience is relating to back-end development (non-GUI based), then I did not appreciate the importance of an excellent user interface.

The main challenges I faced with this app revolved mainly around the database interaction and the test data. Due to the nature of the SQLite database (no support for booleans) I had to find workarounds. I think if I were able to integrate fully with the ERP system, it would actually make this app simpler as I could create a fromJson method for each model and not worry about test data integrity.

I faced a lot of issues regarding image paths in android, something which I now understand a lot better. This was overcame by a combination of the Android Documentation and the Napier lecture slides.