Bus Pass NFC System

Analysis and Design Document

Student: Adrian Timis

**Group: 30432**

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 19.04.2018 | 1.0 |  | Adrian Timis |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

I. Project Specification 4

II. Elaboration – Iteration 1.1 4

1. Domain Model 4

2. Architectural Design 4

2.1 Conceptual Architecture 4

2.2 Package Design 4

2.3 Component and Deployment Diagrams 4

III. Elaboration – Iteration 1.2 4

1. Design Model 4

1.1 Dynamic Behavior 4

1.2 Class Design 4

2. Data Model 4

3. Unit Testing 4

IV. Elaboration – Iteration 2 4

1. Architectural Design Refinement 4

2. Design Model Refinement 4

V. Construction and Transition 5

1. System Testing 5

2. Future improvements 5

VI. Bibliography 5

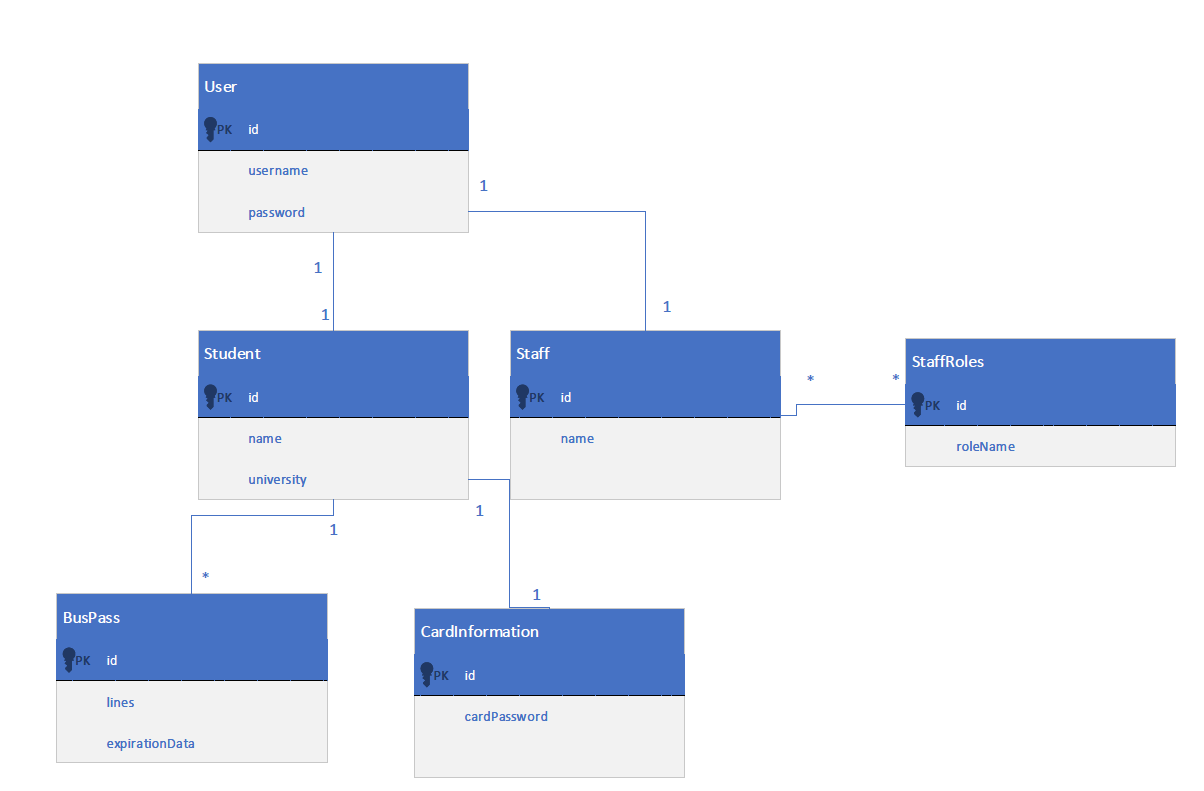
# Project Specification

This project has the sole purpose of providing a better system for students in Cluj-Napoca to manage their monthly free bus passes. The current situation has the student come each moth to the kiosk’s owned by the transportation company, and then wait in a long queue, give the person there a lot of documents that are given each month, and do not change, and then more than often choose the same 2 free lines as the previous month. It does not take too much time to realize that this process can be improved, and this is exactly what this project aims to do.

# Elaboration – Iteration 1.1

# Domain Model

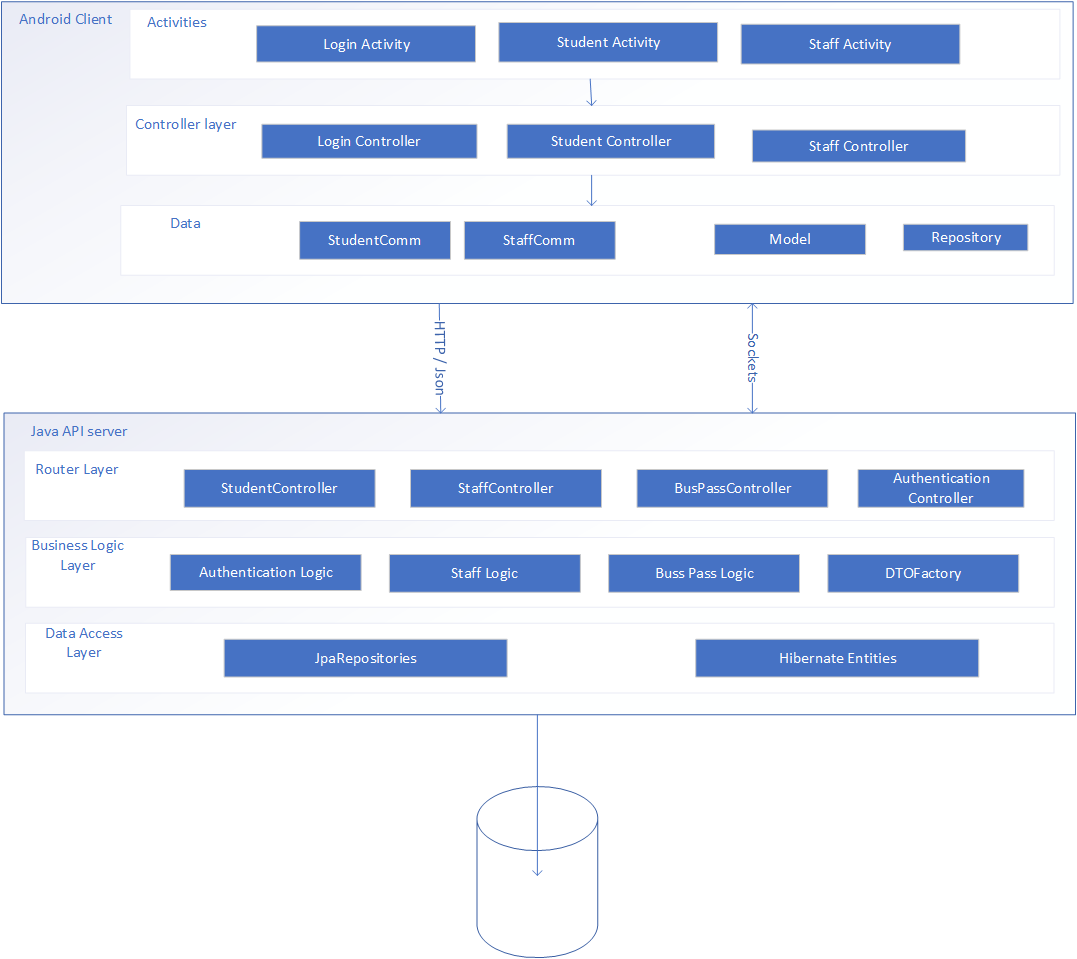
The domain model of this application is composed of the students, staff, and their additional information.



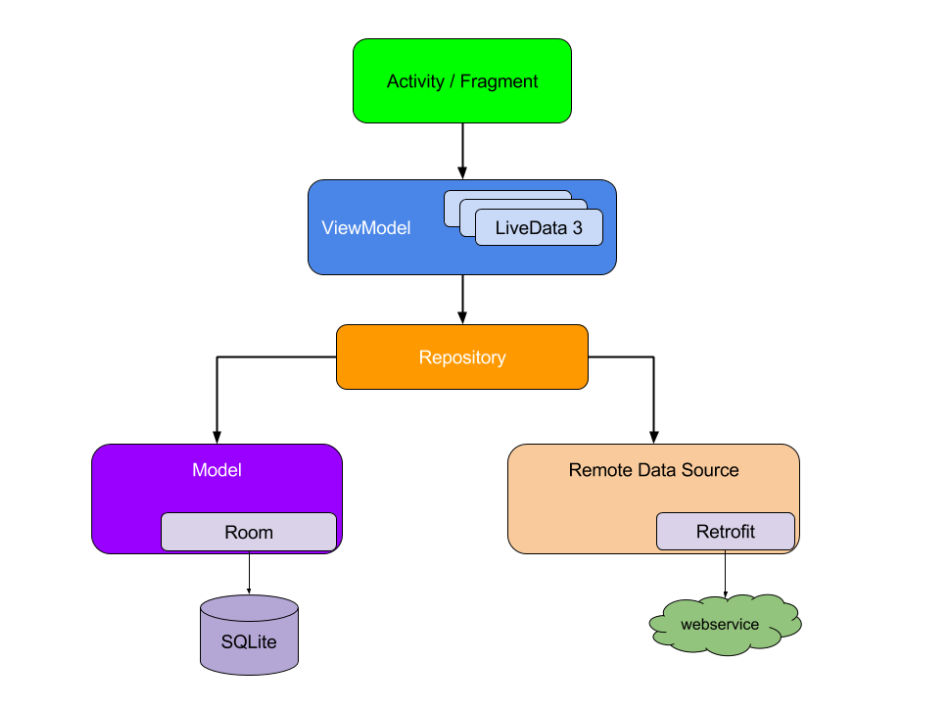
# Architectural Design

## Conceptual Architecture

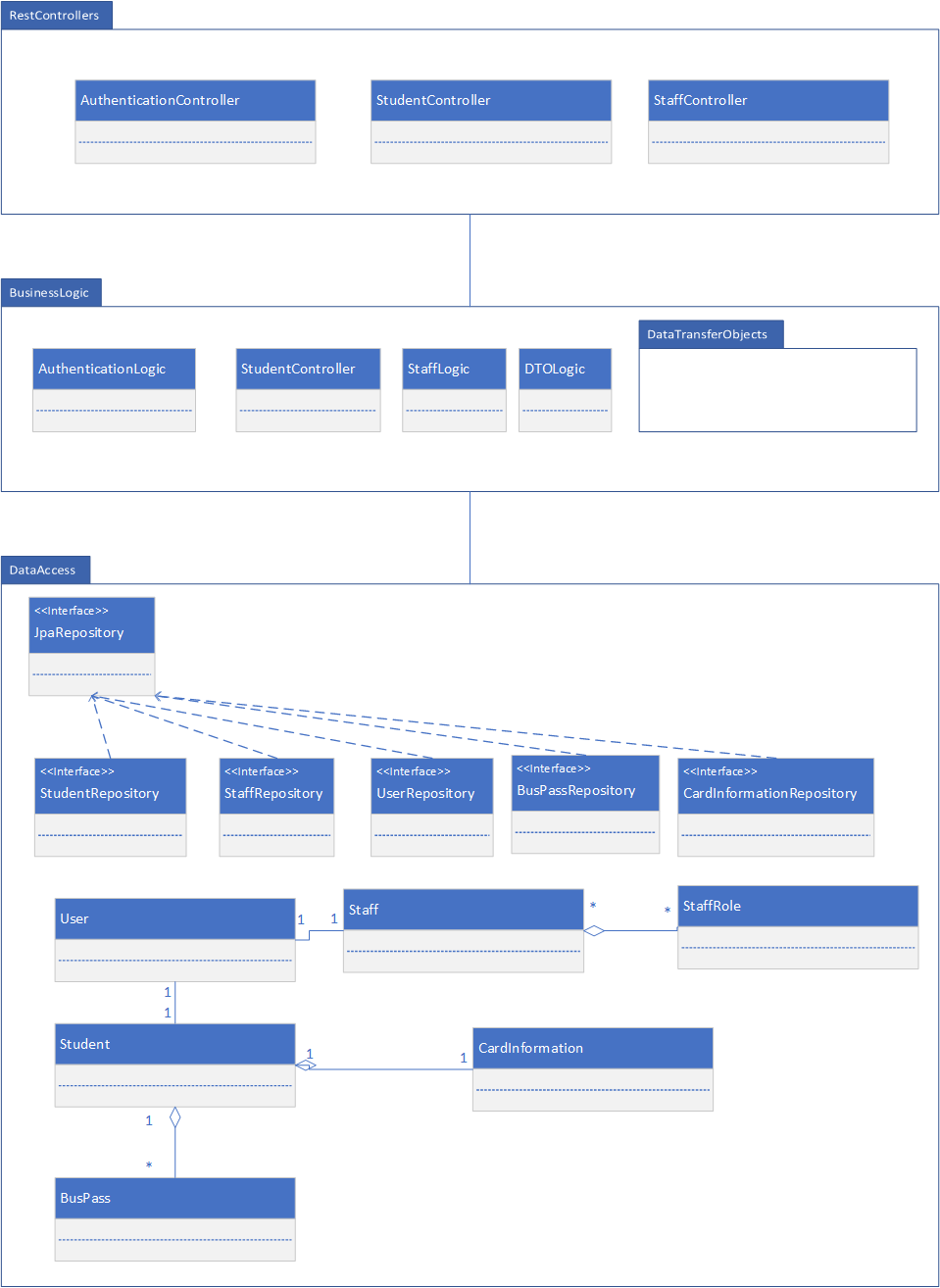
The architecture of the project is based on the Client-Server architectural pattern. The server will be constructed using layered architecture and will feature a REST api so that the client application can communicate with it. The Android application will just feature the presentation layer, which will be done using the MVC model.



This mainly explains the backend part of the application, and not how the android application will be structures. The lack of experience on my part is to blame. However, google provides guides for understanding good architectures in android. The trick is to separate concerns, that is to make sure that a class does only what it should do, and to understand how activity classes work in android, so that you do not store some data in those classes, classes that could be removed by the android OS at any time. Below is a simple diagram provided by google, for an application that has data persistence using caching and a simple form of the observable design patter.

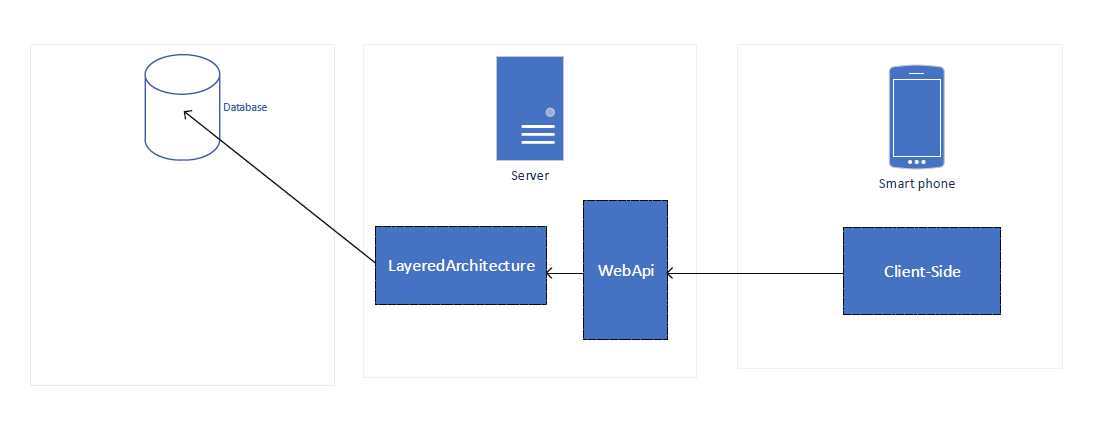


## Package Design

**

## Component and Deployment Diagrams

# 

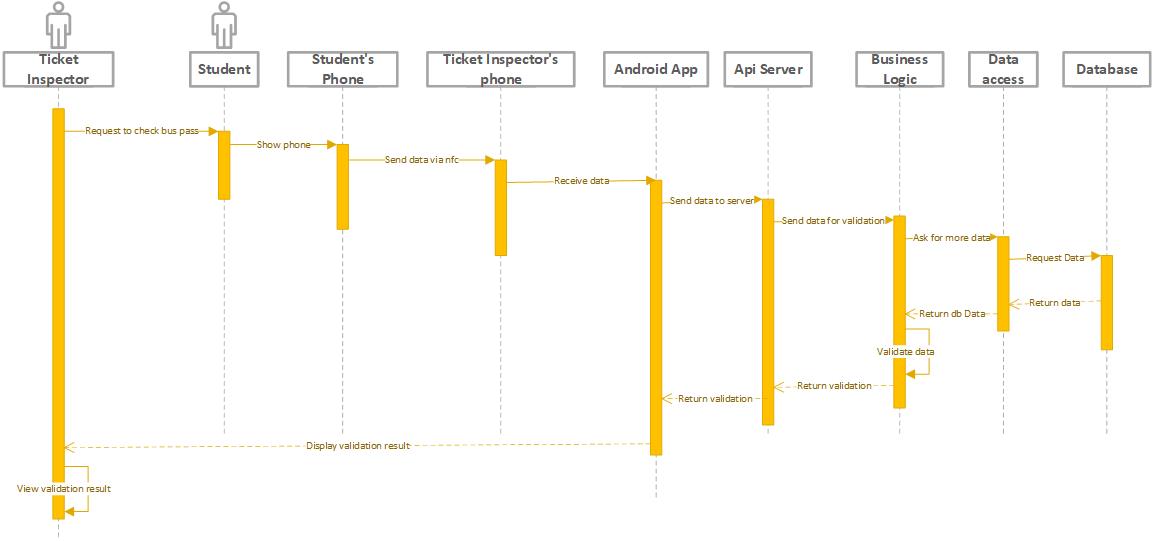


# Elaboration – Iteration 1.2

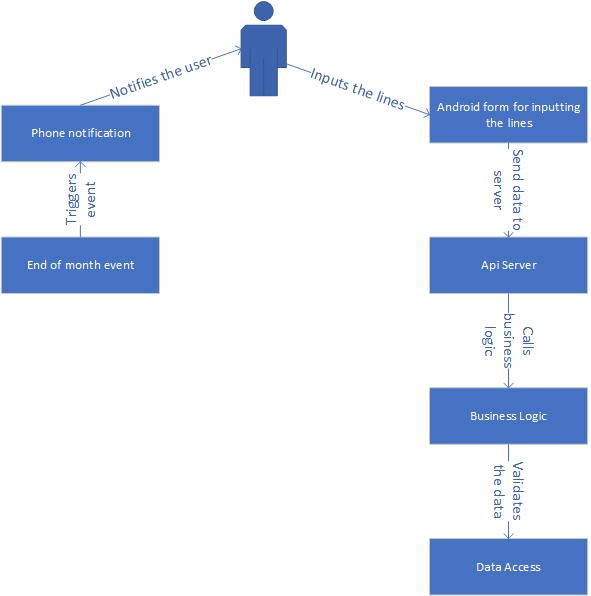
# Design Model

## Dynamic Behavior

The sequence diagram chosen to explain is for the check validation of student’s bus pass by the ticket inspector use case. In this use case, the ticket inspector would like to check whether the student has a bus pass for the line he is currently inspecting. Because users cannot be entirely trustworthy, the validation is done server side, in order to not be tricked by some fake data provided by the student. Both the student and the teacher will have an android application, which will communicate via nfc the data the ticket inspector is required. In this case, since the validation is done server-side, the info needed by the application is just the user id. Because there is not much data needed by the application, the user will also have the option to request an nfc card, which will hold that information.



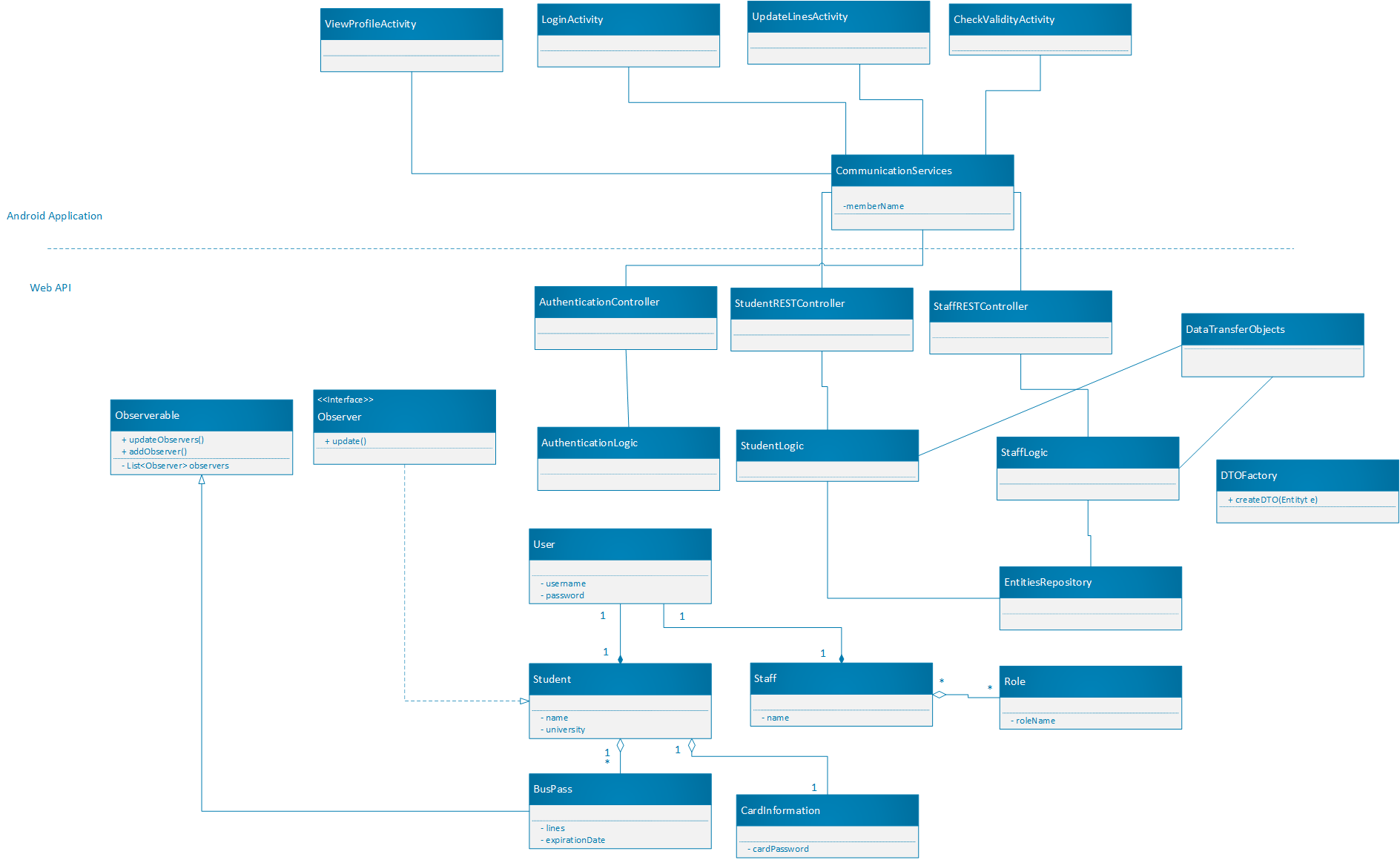
Below is the communication diagram for the scenario where the user must select the bus lines he wishes to have active during the next month. The user has the option for the future months, to automatically set the same lines, removing the need to introduce the same data each month. If the user has not selected this option, then he is alerted via a notification that the bus pass is about to expire and asks him to introduce the bus lines of his choosing.



## Class Design

The UML class diagram presents the major classes that are present in the application. The application will use Hibernate framework for data access, therefore there will be entities and repositories for each entity. The web api part of the application will have a set of controllers, which will act as the communication layer for client applications. The android app will only feature some activity classes.

As for design patterns, as of now only 2 have been introduced in the application. The Observer design pattern will keep a track of the users which need to be notified that their bus pass is about to expire. This will allow some users to opt out of the feature, and choose instead to have their lines updated automatically, without ant form of input being needed from them. The second design pattern is the factory design pattern, which makes data transfer objetcs, which will be transmitted to and from the client side part of the application.



# Data Model

# 

# Unit Testing

There is no clear way as to how the testing will be done. However, unit testing and integration tests will be mainly used. The test cases for which these tests will be performed can be found in the Project\_UseCaseModel file in this folder.

# Elaboration – Iteration 2

# Architectural Design Refinement

*[Refine the architectural design: conceptual architecture, package design (consider package design principles), component and deployment diagrams. Motivate the changes that have been made.]*

# Design Model Refinement

## *[Refine the UML class diagram by applying class design principles and GRASP; motivate your choices. Deliver the updated class diagrams.]*

# Construction and Transition

# System Testing

*[Describe how you applied integration testing and present the associated test case scenarios.]*

# Future improvements

*[Present future improvements for the system]*

# Bibliography