Analysis and Design Document

Student: Zavaczki Péter - Tibor
Group: 30433

Revision History

Date	Version	Description	Author
01/Apr/18	1.0	Beginning document	Zavaczki Péter - Tibor

Table of Contents

I. Project Specification

ITA is an Android based application for traffic participants to cooperate by sending traffic alerts to a central database, from where other users can see them in real time so that they can adapt their route and/or driving style accordingly.

II. Elaboration – Iteration 1.1

1. Domain Model

The domain model of package delivery is presented with the conceptual class diagram in Illustration 1 below. The Users are modeled to have a username, an email to register with and a password, additionally, every user can submit a traffic alert from the moment of their creation. The alerts have a type (ex.: police, roadworks, accident etc.) and a coordinate location that is queried from the device when the alert is submitted (ex.: 46°46'20.0"N 23°35'07.8"E).

Illustration 1: Domain model conceptual class diagram

2. Architectural Design

2.1 Conceptual Architecture

The application will be implemented based on the client-server architectural pattern, since the users should be able to access the service provided by the system at all time. This pattern also allows the main functionality of the application to work correctly, allowing multiple users to be simultaneously connected, all of them being able to send and receive alerts in real time. The view layer of the application is present on the user side along with a portion of the controllers. The server connects the database, which may be hosted locally or on a different domain, to the local data access layer and a portion of the controllers. The Controller layer is split into two sublayers: the client side controller layer and the server side controller layer. This conceptual architecture can be seen on Illustration 2.

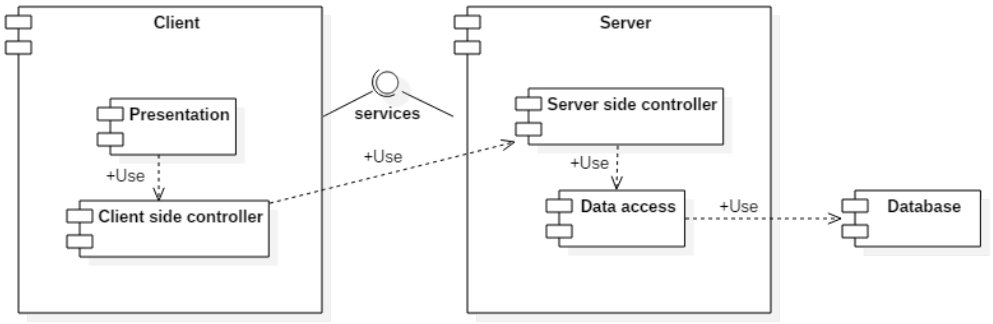

Illustration 2: Conceptual architecture diagram

2.2 Package Design

The package diagram in Illustration 3 represents the structure in which the system's components are organized. Packages encapsulate classes which are semantically connected to a certain extent, as such, for example every data access object will be in the DAOs package.

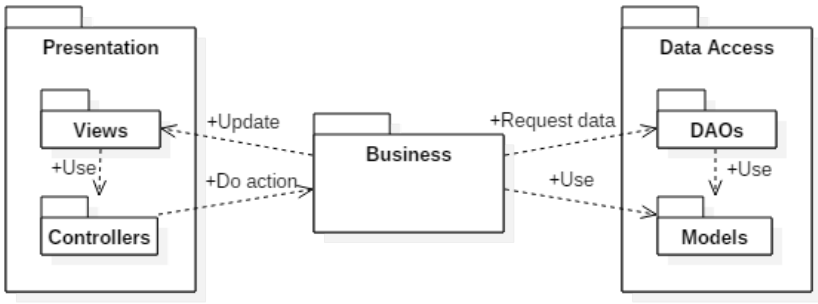

Illustration 3: Conceptal package diagram

2.3 Component and Deployment Diagrams

The project ultimately sums up in two component modules, the visual application, running on the client side device, and the server side logic module. This can be seen on Illustration 4.

Illustration 4: Conceptual component diagram

The application's deployment consists of deploying the app on the execution environment, which then runs on the client's device. The second part of the deployment is deploying the server app on its own execution environment, so that the clients can connect to it. This can be seen in Illustration 5.

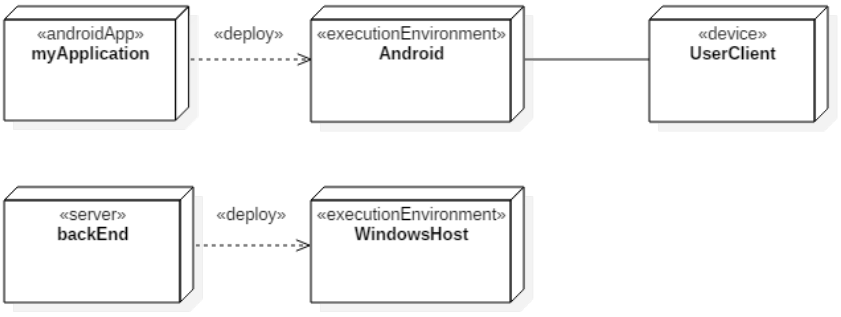

Illustration 5: Conceptual deployment diagram

III. Elaboration – Iteration 1.2

1. Design Model

1.1 Dynamic Behavior

[Create the interaction diagrams (1 sequence, 1 communication diagrams) for 2 relevant scenarios]

1.2 Class Design

[Create the UML class diagram; apply GoF patterns and motivate your choice]

2. Data Model

[Create the data model for the system.]

3. Unit Testing

[Present the used testing methods and the associated test case scenarios.]

IV. Elaboration – Iteration 2

1. 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.]

2. Design Model Refinement

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

V. Construction and Transition

1. System Testing

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

2. Future improvements

[Present future improvements for the system]

VI. Bibliography