Initiatives Platform

Analysis and Design Document

Robert Varadi

**Ieremias Viorel**

**Group: 30432**

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 04/Apr/2018 | 1.0 | First Draft | Robert Varadi  Ieremias Viorel |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

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

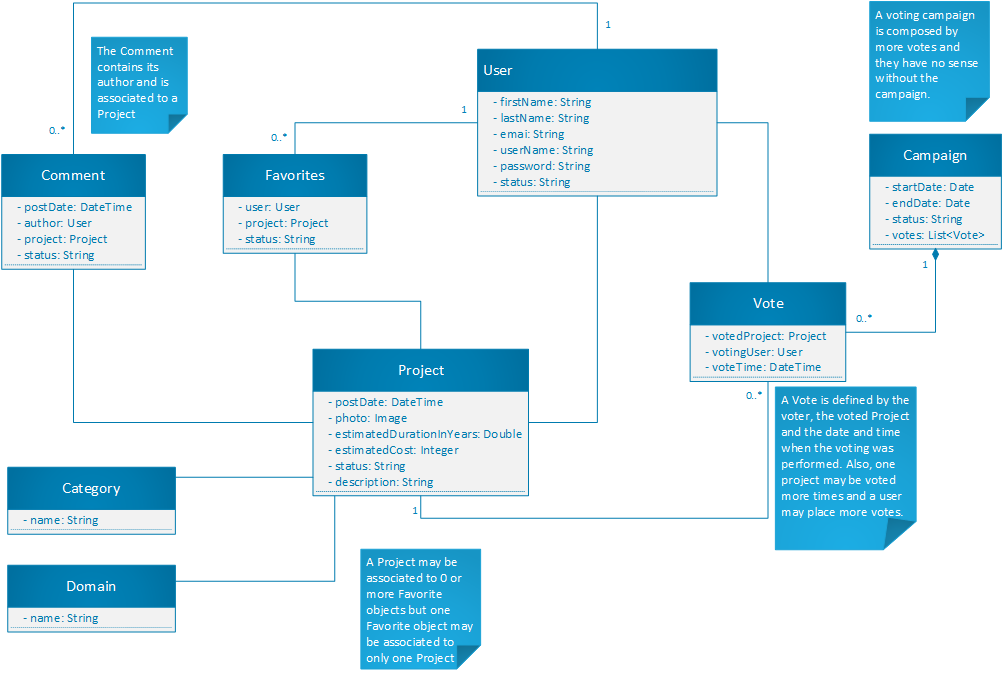
VI. Bibliography 5

# Project Specification

Implement a system where regular people can propose plans they would like to see implemented in their city. Other people, also users of the system, vote for the project they consider most useful. Projects are divided in some categories, on domain. This can however be extended on budget, for example, or implementation time (short-term, medium-term, long-term, etc). Each person is able to vote for a limited number of projects and choosing the best project may take place in more than one round. The application must contain a comments section, too so people will be able to discuss about the different projects. Each person must have a “favorites” corner from where he/she can choose the projects he/she likes the most.

# Elaboration – Iteration 1.1

# Domain Model

The main entities involved in the system are the User, the Project and the Vote. A User entity abstracts personal and account information of a real user of the system. A Project entity contains relevant information related to the initiatives and ideas proposed by the citizens. In order to provide a useful classification for the projects, each one has attached a Domain and Category attribute. Furthermore, to provide a communication feature between users of the system, a Comment entity must be added, while a Favorite entity is used to provide the user with the ability to track the projects he/she is most interested in. The central purpose of the application is to determine the needs of most of the people so a voting mechanism is built around the Vote entity. A more accurate result is obtained if the projects are filtered and eliminated in stages so campaigns are added in the system in order to achieve this purpose.

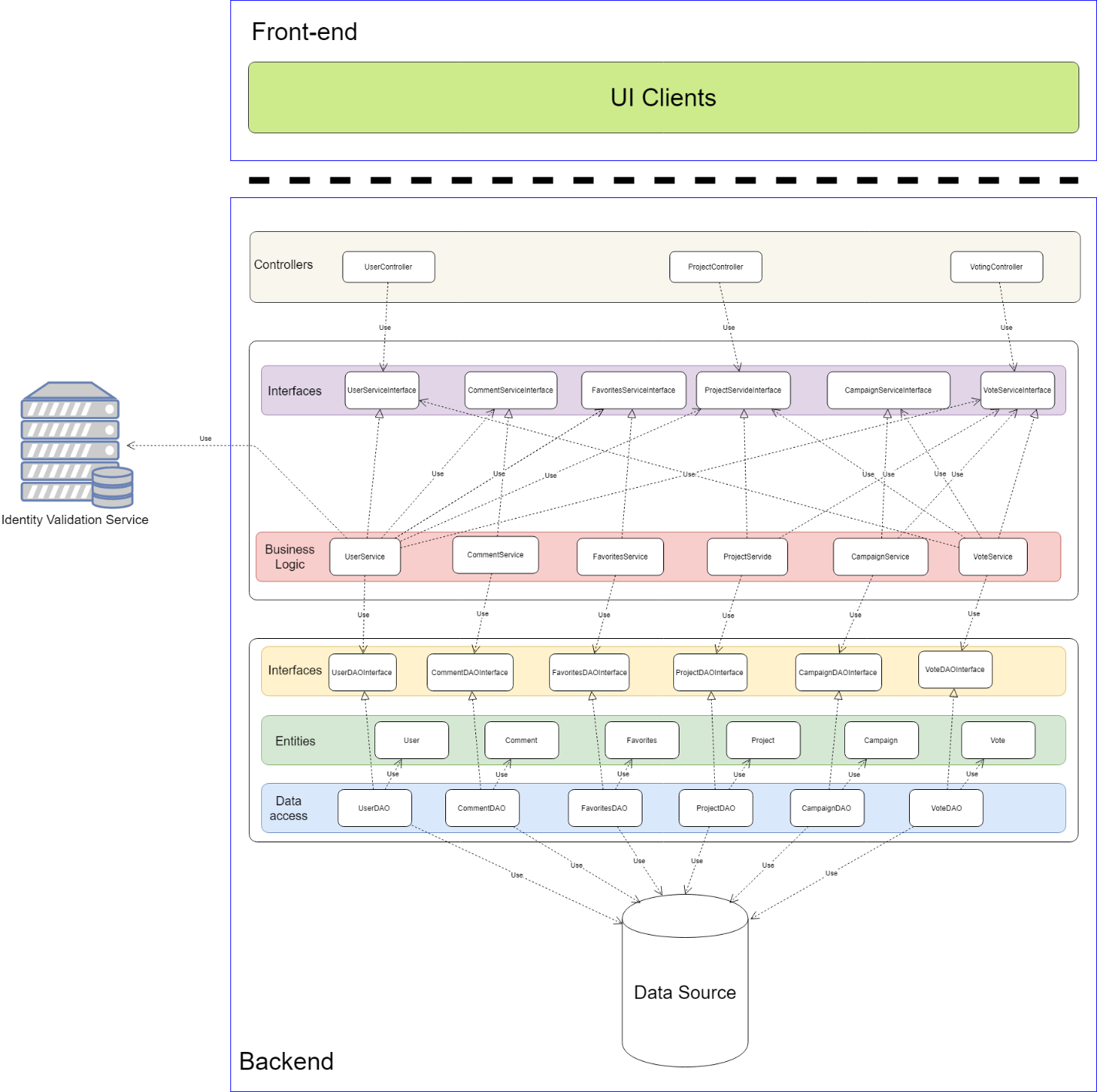
# Architectural Design

## Conceptual Architecture

The application is built as a decoupled system, composed of a REST API and a ReactJS – based Web Application client. The core API receives HTTP requests coming from the client, processes them and returns the HTTP response which will be displayed, in some format, by the client. From this perspective, the application is a distributed client – server system.

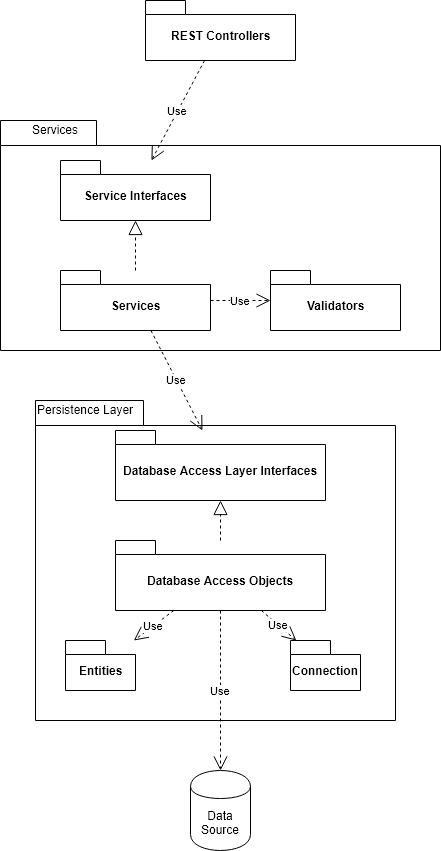
The back-end is structured using the layers pattern, which has the advantage of isolating specific processing steps into independent components which can be tested, replaced, maintained and extended separately. Also, the layers will communicate with each other only through services which are exposed from one to another through interfaces which hide the specific implementation.

At the application level, we use MVVM (Model View View – Model). The Model component is represented by the persistence layer and the business logic layer, where the state of the application is stored and the corresponding processing logic is implemented. The View component is the graphical user interface itself, which is only responsible for the “look” of the application, namely what the user sees and interacts with. The View – Model mechanism is specific to the ReactJS implementations, in which the view stores its own state from which it renders for the user. The advantage of this approach is that the view and the model are decoupled and any number of clients can be added or replaced without affecting the application.



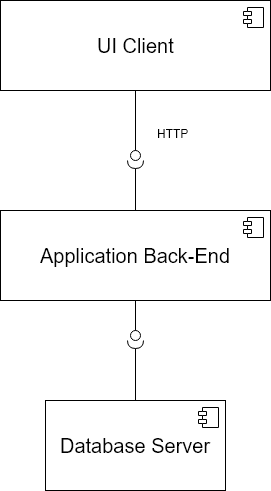
## Package Design

At package level design level, each layer is assigned a package, which is responsible for the whole functionality of the layer. The coupling between the layers is low, because the only way a higher – level layer communicates with a lower – level one is through the interface exposed by the latter one.

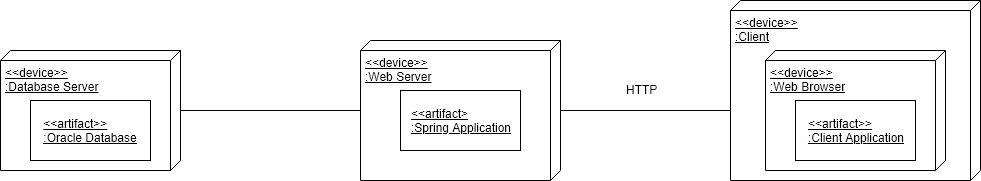


## Component and Deployment Diagrams

# The component diagram contains the layout of the system at physical component level, that is, files, libraries, executables. From this perspective, we can draw the following component diagram:



The deployment diagram contains the physical layout of the system grouped by tiers. We might have the Database Server on a tier, the Application Server on a different one and the UI Client on a third one. Thus, the deployment diagram is the following one:



# Elaboration – Iteration 1.2

# Design Model

## Dynamic Behavior

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

## Class Design

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

# Data Model

*[Create the data model for the system.]*

# Unit Testing

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

# 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