2DV609
Project Course in Software Engineering
Assignment D2 - Design Document
Group 6

April 30, 2020

Contents

1 Introduction

1.1 Description

The system is a web based messaging service providing services for sending messages/media between users. The system also supports a database which will store user data as well as conversation data.

1.2 Requirements

The requirements used as source for this document is the full list of functional requirements located in the document *2DV609_D1_RequirementsDocument_Group6*.

1.3 General Priorities

If we had priorities that made an impact on choices made when designing the system.

- **Architecture:** The system should be build by using a known architectual pattern for systems supporting a user interface.

2 Design Details

This section will provide more thorough information on design choices, alternatives and in-depth models of the system.

2.1 Design Choices

A list of all the design choices that was made, a rationale behind each decision and a discussion.

MVC-pattern
The first that was made was to use the *Model-View-Controller*-pattern as inspiration for the design. The alternative to this was to user the *Observer*-pattern for interactions between the view and the rest of the system which also seemed to be a common pattern for this sort of system, but as a majority of the group members has more experience working with the MVC-pattern this was the favourable outcome.

Database Facade
As the majority of this system will be interacting with a database a choice was made to have all of the functionality of the database gathered in a single component based on the *Facade*-pattern. This component should return an instance of a class that provides all of the database functionality required by the rest of the system.

2.2 Issues

Issues that has risen while making design choices and possible alternatives to the design we chose.

2.3 Patterns

A detailed list of all of the patterns that was included in the design and rationale.

- MVC-pattern
- Facade Pattern

2.4 Performance Model

A more detailed performance model and how the model has affected the design. Unclear what to do here as the performance model has not affected the design.

2.5 System Overview

This section will provide a high level overview of the system and how the different parts of the system will interact.

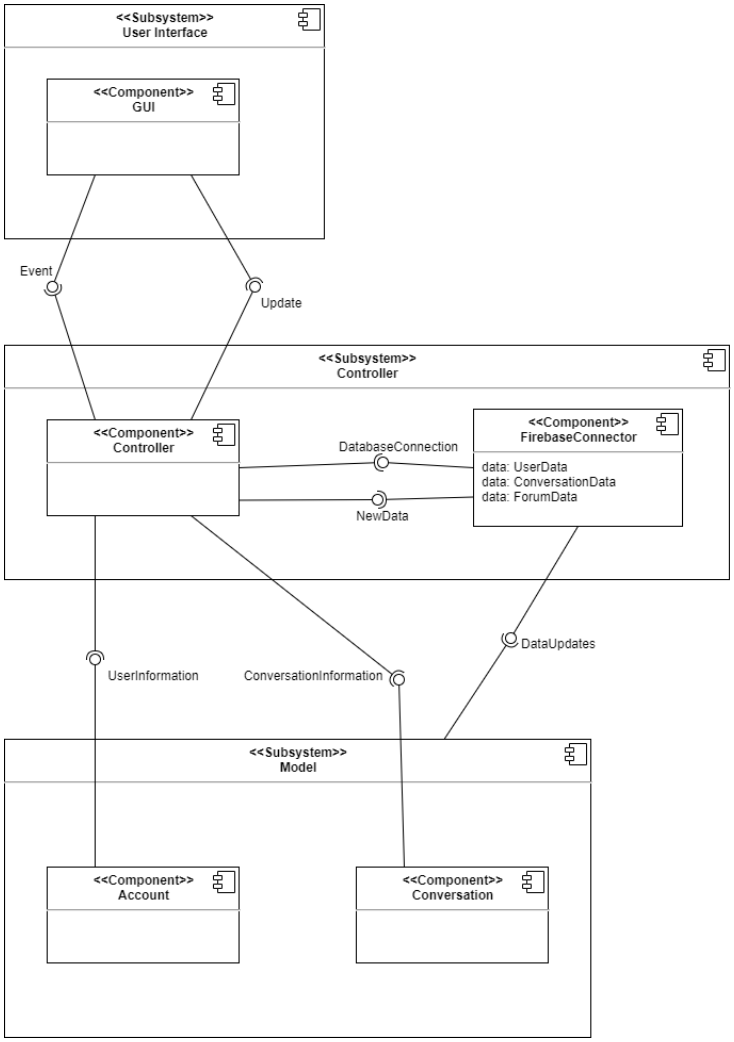

Figure 1: Component Diagram

- **GUI:** The user interface component. This component will include all of the visual elements.
  Interfaces:
  - This component will have an interface which pushes *Events* to the *Controller*-component.

- **Controller:** This component will serve as state-control which receives/delegates information between the view and the rest of the system.
  Interfaces:
  - The interface towards the *GUI*-component will push *Updates* whenever data has been updated.
  - The interface towards the *FirebaseConnector*-component will push *new data* received from the *GUI*-component.

- **FirebaseConnector:** A centralized component that contains everything related to the database.
  Interfaces:
  - This component will have an interface towards the *Controller*-component providing a *DatabaseConnection*.
  - This component will also provide *Data updates* to all of the **model-components** when relevant data has been updated in the database.

- **Account:** A component that contains all classes related to accounting (userinformation, passwords, friends..).
  Interfaces:
  - This component will provide accounting information for the *Controller*-component.

- **Conversation:** A component that contains all classes related to chatting (conversation, message..).
  Interfaces:
  - This component will provide chat-related information for the *Controller*-component.

2.6 Detailed Component Information

This section will list each component and provide detailed information of the internal structure of that component and its interactions.

2.6.1 GUI

The main View-component that includes all classes with viewable elements.

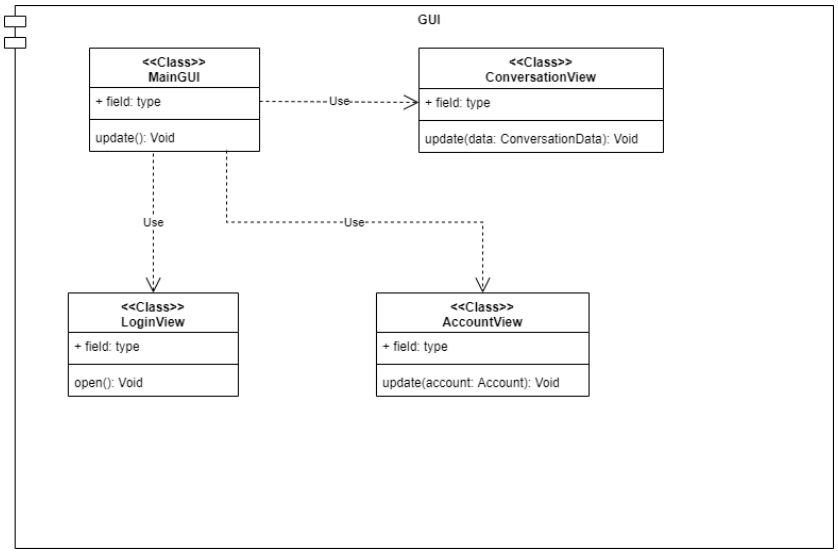

Figure 2: GUI Component

MainGUI
Main responsibility is holding the main interface, navigational menu bars, static visual elements and to display other view classes.

- Display static visual elements
- Display other view classes
- Support means to navigate between different views

LoginView
The class that provides a login view.

- Support means for a user to input login credentials

ConversationView
This class will hold all visual elements needed for chatting with another user.

- Support means for conversations between users

AccountView
This class will hold all visual elements representing a users account information.

- Support means to display account information
- Support means to change account information
- Support means to delete the account

2.6.2 Controller

The main controller that handles interactions between other controller-components, view-components and model-components.

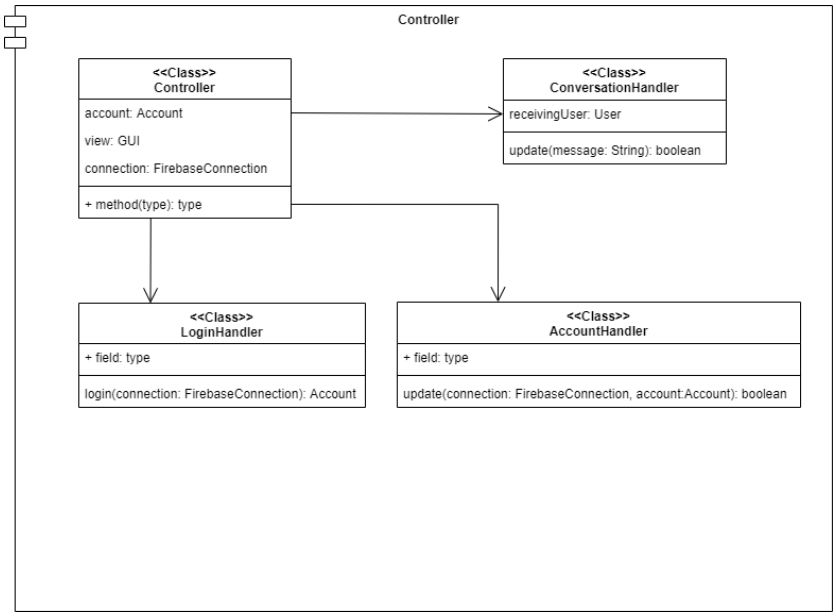

Figure 3: Controller Component

Controller
The main controller that handles communication between the View, the Model and other controllers

- Support communication between other controllers.
- Handle all input from the view-classes
- Update the view when data has been updated
- Send requests to the *FirebaseConnector*

ConversationHandler
This class will handle communication between the *View*-component and the *Conversation*-component

- Control the flow of information between the Conversation-View and Conversation-Model

LoginHandler
This class will handle communication between the *LoginView* and the *FirebaseConnector*.

- Handle input from the *LoginView*
- Send a login request to the database

AccountHandler
This class will handle all communication between the *AccountView* and *AccountModel*

2.6.3 FirebaseConnector

The controller-component responsible for all communication between the main controller and the database.

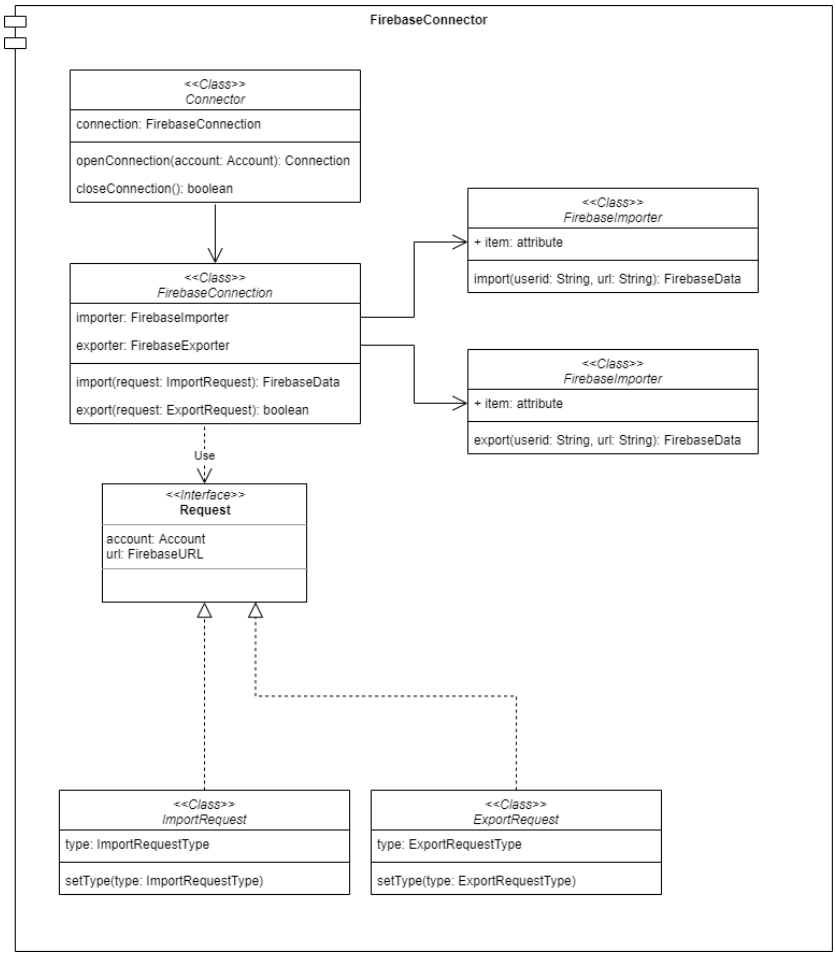

Figure 4: FirebaseConnector Component

Connector
The main class of the *FirebaseConnector*-component which provides a facade with all of the functionality of the component.

- Request for a new Account
- Login a user to Firebase
- Open a connection to Firebase

- Close a connection to Firebase
- Get data from Firebase
- Update data in Firebase
- Push new data to Firebase

FirebaseConnection
The class that will hold all information regarding the firebase connection.

- Open a connection to Firebase
- Close a connection to Firebase
- Request for a new Account
- Login a user to Firebase

FirebaseImporter
The class that will handle all imports from Firebase.

- Get data from Firebase

FirebaseExporter
The class that will handle all Exports to Firebase.

- Update data in Firebase
- Push new data to Firebase

Request
An interface for requests to Firebase

- Any class that implements the Request interface should provide a new way of forming a request that is understood by the database.