Chat Client

Team 4 Architecture Document

Author: Bachir Karim
E-mail: fakemail@gmail.com
Date: 13/05/2020
Year: 2019/2020

Table of Contents

1.0 Introduction

The Chat Client Team 4 Architecture Document is designed to illustrate and identify the high-level architecture systems used to design and implement the Chat Client application. The document contains an overall view of the system hierarchy, logical views of the system components, and a process view of the system's communication.

2.0 High Level Hierarchy

2.1 Hierarchy Diagram

2.2 Hierarchy Description

The architecture system for the Chat Client application is an n-tier architecture. This architecture system is designed to allow for proper information hiding, modular components, and single system dependencies. The abstraction of the presentation layer, and consequently the Graphical User Interface (GUI), allow for a flexible pipeline for the optimization of the GUI to meet customer needs and expectations. There are multiple layers between the Presentation Layer and the lowest level, due to the significant challenges present in the optimization and control of the Processes design. The Database layer is the lowest level in the hierarchy and is used to save data.

3.0 Components Classification

3.1 Presentation Layer

Purpose: To display buttons, text fields, text area, images and sounds to the user to create a fluid and efficient user experience.

Specific Nature: The presentation layer will be in charge of displaying appropriate images, menus, and sounds to the user. This layer will also be in charge of handling stylus clicks. When a user clicks a button on the GUI, the code corresponding to that event will be called. This layer will also be in charge of the spawning of appropriate threads to increase the performance of the application. The need of spawning extra threads is due to the fact that the main thread of the app will be watching for event clicks, but we also need another thread constantly running to collect all the messages.

3.2 Controller Layer

Purpose: Processes and responds to events, typically user actions, and may invoke changes on the model.

Specific Nature: The controller layer in our program will be in charge of getting the status of the connection with the server. It will do this on a constant interval of k. k is a 5-10 second time step that will be determined through testing.

Associated Constructs: Status Controller

- **Connection Status Controller** – Connection Status Controller class will consist of the current connection status. This class will be updated with the newer status every k seconds. This Status Controller will be executed by the Presentation Thread while the user is chatting.

3.3 Business Layer

Purpose: This layer is in charge of the heavy algorithm business logic found in complex solutions.

Specific Nature: This layer will be used to compute the algorithm for finding the connection status and for detecting any type of attack. The current status algorithm will be located in a class called connection status Manager. The attack detection algorithm will be located in a class called attack detection Manager. This layer will also contain a class called Error Manager. This class is in charge of getting the appropriate error message based on the actions of the user.

3.4 Record Layer

Purpose: This layer is in charge of containing the classes that strictly consist of data. Little to no functional methods will be found in these classes.

Specific Nature: This layer will be used to store User data, connection status data, and Error data. These classes will only contain properties (variables) that describe each data type.

Associated Constructs: User Record, connection status Record, Error Record

- **User Record** – User Record class will consist of only properties describing a user. This class will be static, meaning there is only one copy of thisclass in memory once initialized until the end of the program.
    - _ **User Record** _
        - *User Name*
- **Connection Status Record** – Connection Status Record class will consist of only properties describing a status.
    - _ **Connection Status Record** _
        - *date*
        - *status* – connected, disconnected, waiting connection, in connection, error.
        - *Server IP*
        - *Client IP*
- **Error Record** – Error Record class will consist of only properties describing an error.
    - _ **Error Record** _
        - *Name* – Will hold the name of the error. This property is of type *string*.
        - *Description* – Will hold the description of the error. This property is of type *string*.
        - *Date*

3.5 Data Access Layer

Purpose: This layer is in charge of communicating to the database. This layer should handle all of the database transactions and connectivity.

Specific Nature: This layer will be in charge of communication to our database which will in turn lead to populating the record layer. Our database in this project will consist of XML files. The Connection Status DAL class will be used to read the Connection Status XML file and populate Connection Status Record classes based on the data in the Connection Status XML file. The User DAL class will be used to initialize a User Record class. This User DAL class will only be in charge of initializing the User Record class to null because we will not be storing different user types in an XML file. The Error DAL class will be in charge of reading an Error XML file and populating all of the Error Record classes according to all of the different types of errors the system can throw.

3.6 Database Layer

Purpose: This layer is in charge of storing data in persistent storage.

Specific Nature: This layer will consist of XML files. These together will be our database management system. These XML files will store Users, Errors, and eventually Language Support.

Associated Constructs: Users XML, Errors XML

- **Users XML** – Users XML will be used to store all Users that will login in our system.
    - **_ Users XML _**

```
<Users>
    <User>
        <name>Pippo<name>
    </User>
</Users>
```

- **Errors XML** – Errors XML will be used to store all Errors that could be used in our system.
    - **Errors XML**

```
<Errors>
    <Error>
        <name>undefined</name>
        <description>an undefined error</description>
        <date>13/02/2003</date>
    </Error>
</Errors>
```

4.0 Process View

4.1 Process View Description

The Process View is essential in understanding how the separate components and subcomponents communicate with each other in a concurrent application. By better understanding the necessary paths of communication between the components, it may be possible to optimize the data flow and storage of the application, as well as ensuring thread-safety.

4.2 Application Thread

This thread is the main application thread that is created at runtime of the program. The program creates the thread; this is not a user created thread. This thread handles the basic program flow by controlling navigation between tabs, and processing window events, including the handling of user input GUI.

4.3 Message Collector Thread

This thread collects all the messages that the client receives, creates a new Interpreter thread to read the message.

4.4 Interpreter Thread

This thread reads the message and takes action based on what it contains.