Architecture/Design Document

Table of Contents

Change History
Version: 2.7
Date: 12/12/2019
Description of Change: First release of complete application

Introduction

Server and Client

This document describes the architecture and design for the ServerClientChat application being developed for De Carli Lorenzo.

With this server every group's member can chat with another classmate. The advantage of this system is that it's more secure because it's working only on the LAN

The purpose of this document is to describe the architecture and design of the ServerClientChat application in a way that addresses the interests and concerns of all major stakeholders. For this application the major stakeholders are:

1. Users - They want a simply interface to use the client
2. Users and the customer – they want assurances that the server will provide a service simple but that it works well for resttrict use that it will have
3. Developers – they want an architecture that will minimize complexity and development effort.
4. Project Manager – the project manager want a clean work, as simple as possible. They also want a clear everyone's knowledge of the whole project
5. Maintenance Programmers – they want assurance that the system will be easy to modify in case of bugs.

Design Goals

Server and Client

The design priorities for the ServerClientChat application are: • The design should minimize complexity and development effort. • The design should be easy to modify from every programmer. • The design must be follow the PEP-8 coding standards

System Behavior

The use case view is used to both drive the design phase and validate the output of the design phase. The architecture description presented here starts with a review of the expect system behavior in order to set the stage for the architecture description that follows. For a more detailed account of software requirements, see the requirements document.

Server

The Server side behave in this way: 1. Wait the user's logins (after sign in if is a new user) 2. For every user create a new thread (calls Connection) 3. For every 'connections' it wait any request, or send some answers/messages 4. At the logout close the thread, then the connection

Client

The Client side behave in this way (in the case that you send a message): 1. Wait the confirmation that the server have received the request of send a message 2. If the server answere that he have sent the message, the client create a new thread to wait the answere of the other user 3. After the client have received the answere, it close the thread

Logical View

In this section the modules of the system are expressed in terms of components (architecture)

The architecture of our project is very simple, in fact the *Server* can be assigned to the business layer, and we can also have a database for manage our users. For the other side, the *Client* can be assigned to Presentation layer because its purpose is to provide a Graphic Interface

Process View

The **Server** crate a new thread for every users connected. In this way every users can have a "private" space in the Server. The thread will be opened with login and closed with logout

The **Client** crate a new thread for every mex sent. In this way the users have a personal chat until the chat goes on. The thread will be opened with the sent of the first message to another users and closed with the end of the chat

Physical View

The only physical device required is a machine always turned on with a running Server. The clients also need a machine but are not essentially for proper operation of server, in fact the server will work fine also without clients.

Use Case View

For correct use of this application you need to set up your IP address in .py file and after start it.

Coding

Server and Client

Files standards

All the files are encoded with UTF-8 standard.
All the python code was written following PEP-8 standard: you can check the documentation here

Standard of names

Classes: UpperCamelCase
Function: lowerCamelCase
Variables: undercase separating words with “_"