Software Design Document

WatchDog server monitoring web application
Version: 1.0
Release Date: 11/5/2023

1. Introduction

1.1 Purpose

The purpose of this document is to provide a detailed description of the design of the WatchDog server monitoring Application. This document is intended to help developers, and stakeholders for a better understanding of this project.

1.2 Scope

WatchDog server monitoring application is a web-based application that intends to help the target users manage their servers. The core features of this application include showing statistics information of servers in line graphs and pie charts.

2. Architecture Overview

2.1 Application Architecture

WatchDog Application follows a three-layer architecture:

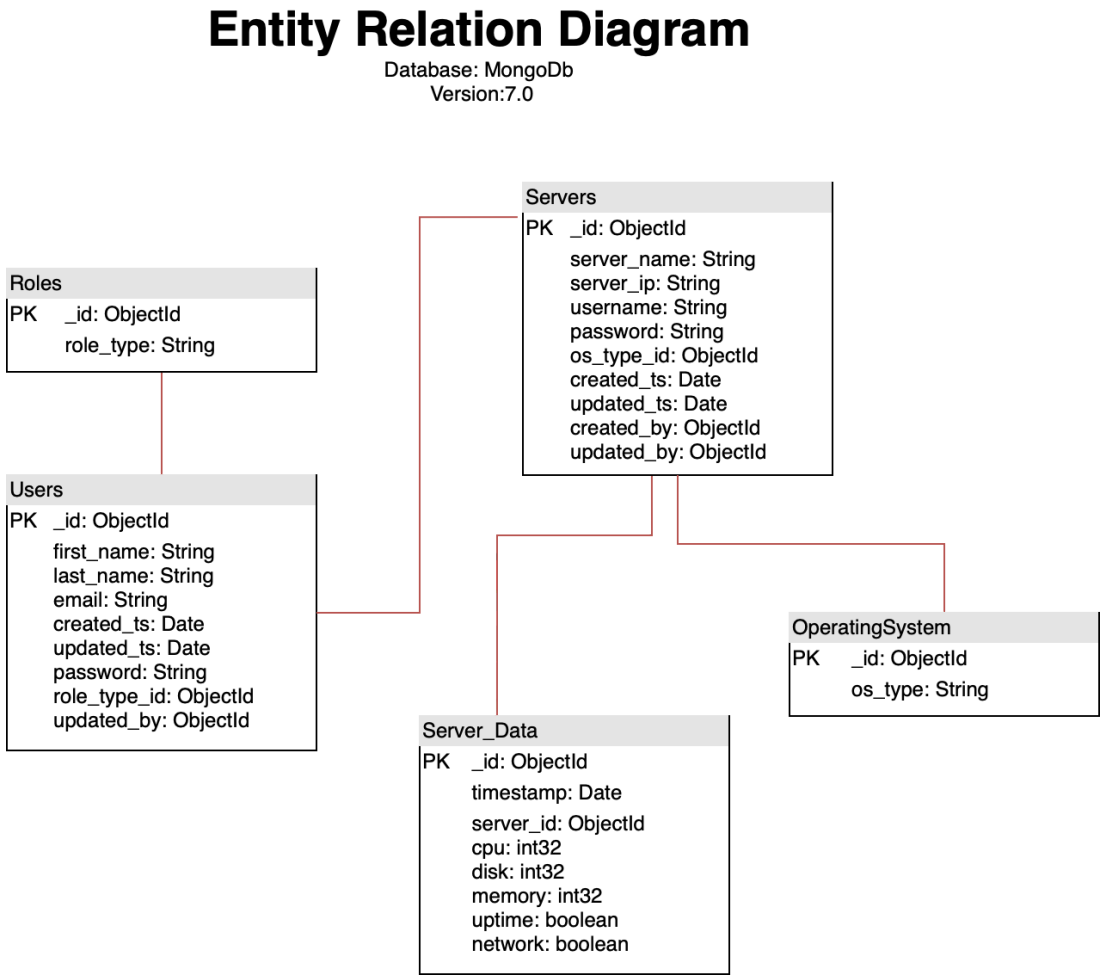
- Application UI Layer: This layer contains the web-based user interface that allows users to control the components to perform the core functionality of this application.
- Server Layer: This layer contains all business logic, and handles data stored in the database.
- Data Layer: This layer contains the database to store all user and server information.

2.2 Technologies

Database: MongoDB
Backend Server: Express.JS, Node.JS
Front-End UI: React.JS
UI Design: Figma
Cloud: AWS
Languages: Javascript, HTML/CSS

3. Database Design

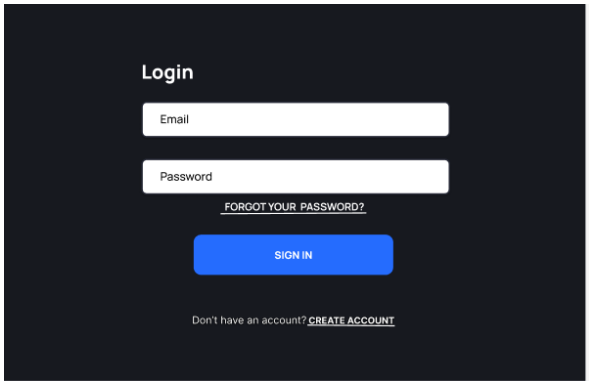
3.1 Entity Relationship Diagram

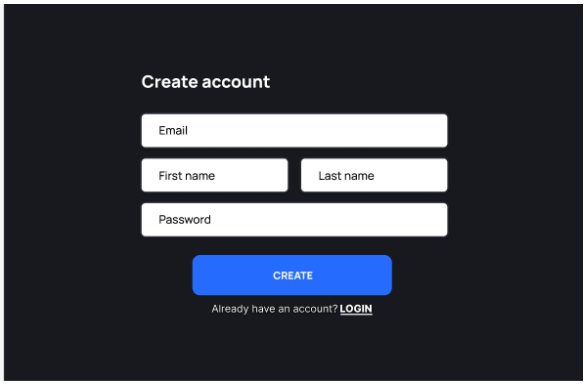


4. User Interface Design

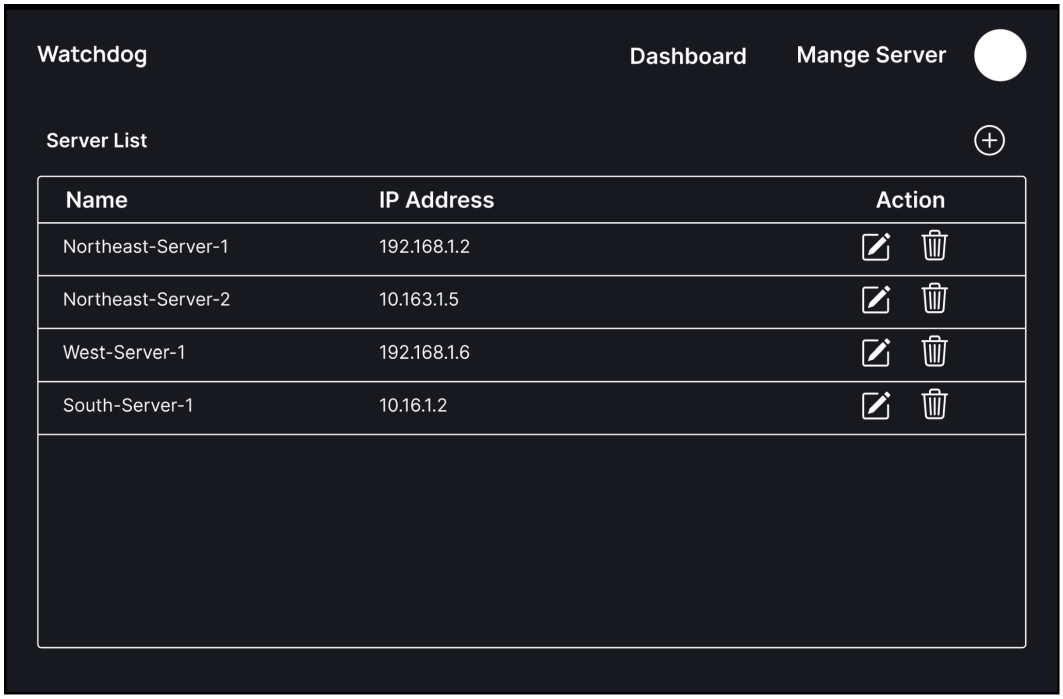
4.1 Main Dashboard

4.2 Login and Sign up

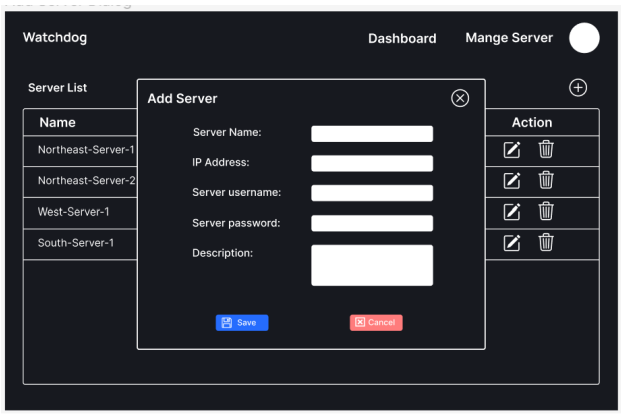


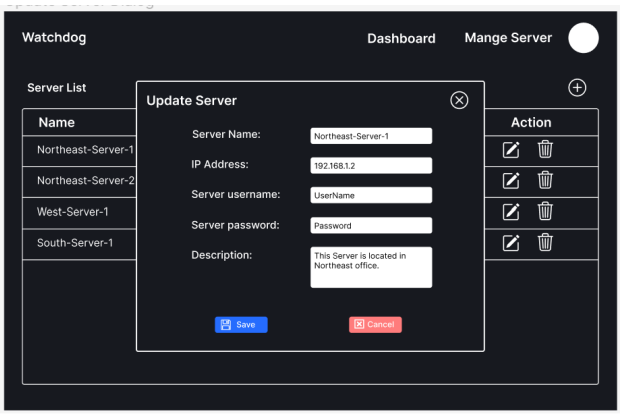


4.3 Server List

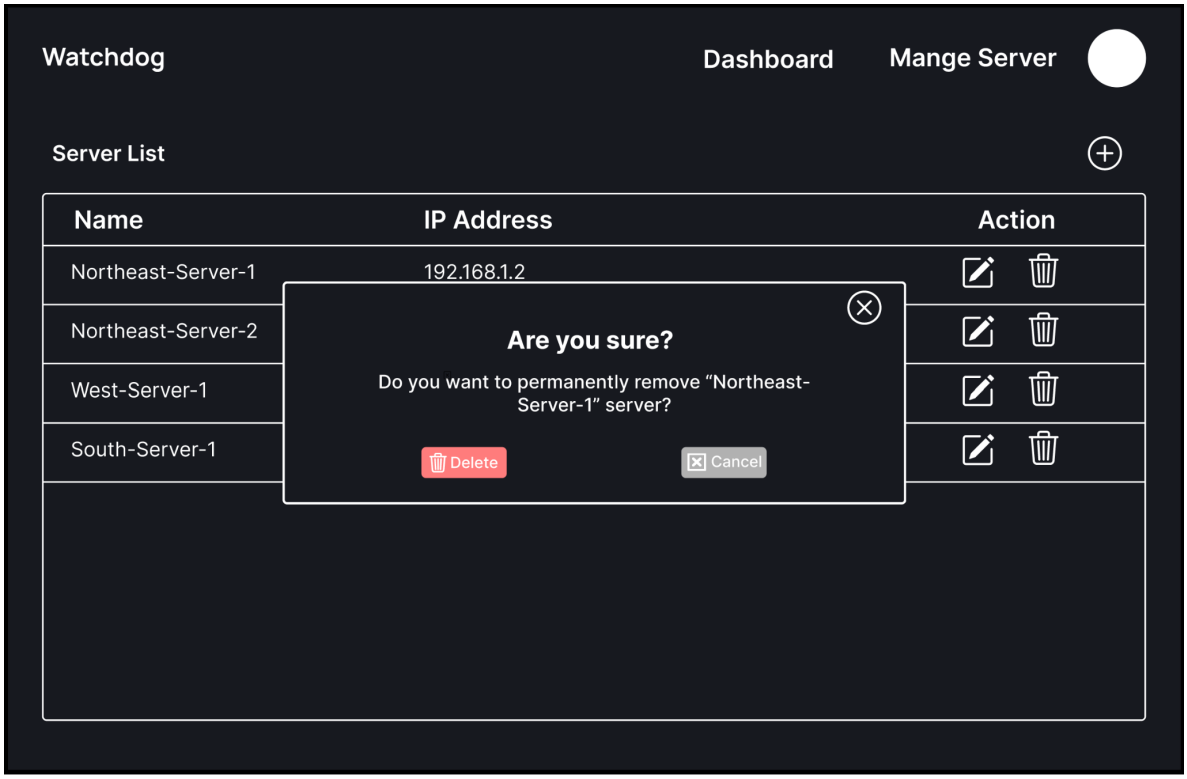


4.4 Add and Update Server



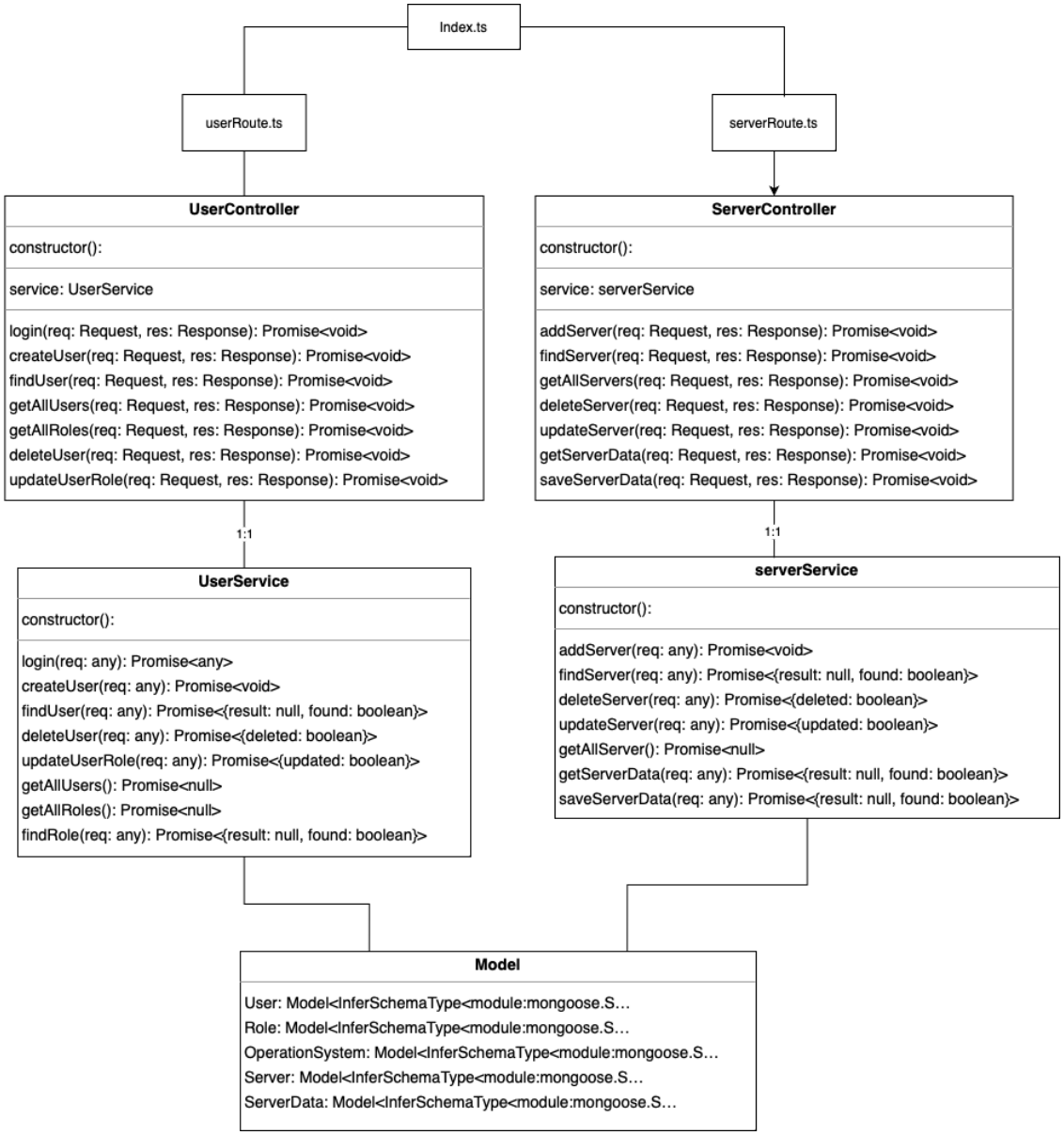


4.5 Delete Server Dialog



5. Back-End Design

5.1 UML DIAGRAM

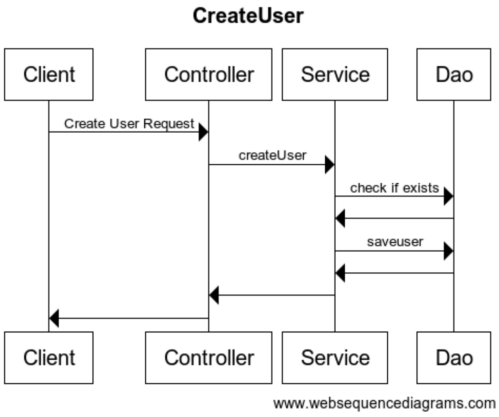


5.2 Sequence Diagram

5.2.1 Create User

Methods: POST

- **API Design** - Sequence Diagram

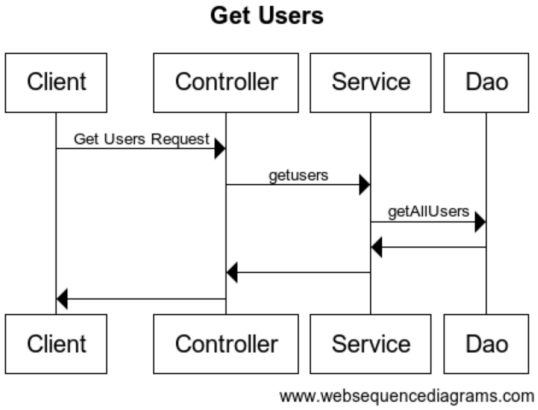


- **Description**: User sign-up will create a user in the database.
- **Flow**:
    - Users enter all information into the sign-up form
    - Users click “Create”
    - Back-end check if the user already exists,
    - If yes, return a message stating that the user has already existed.
    - If not, save the user to the database, and return a success message.

5.2.2 Get Users

Methods: GET

- API Design - Sequence Diagram

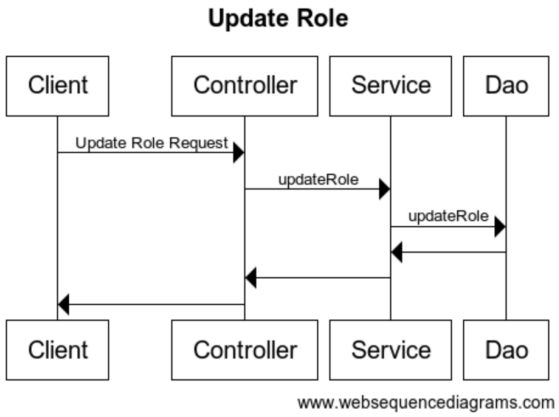


- **Description**: Get all basic information for all current users
- **Flow**:
    - When getting the request to get all users, the backend invokes certain queries to fetch data from the database.
    - The database returns all user data
    - The backend returns to the UI and displays on the page.

5.2.3 Update Role

Methods: POST

- API Design - Sequence Diagram

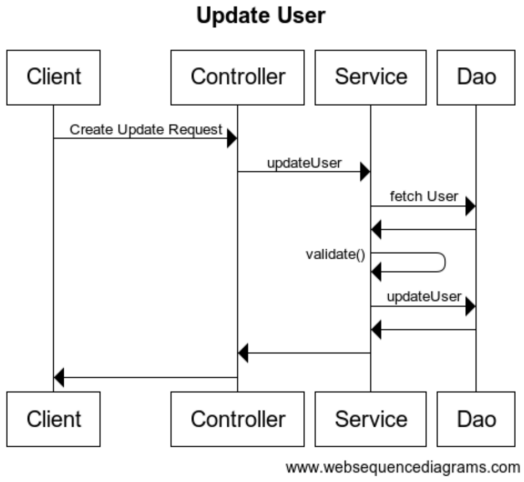


- **Description**: Update the user account's role
- **Flow**:
    - The user clicks "Save"
    - UI sends updated role data to the backend.
    - The backend updates the database and returns a success message.

5.2.4 Update User

Methods: POST

- API Design - Sequence Diagram



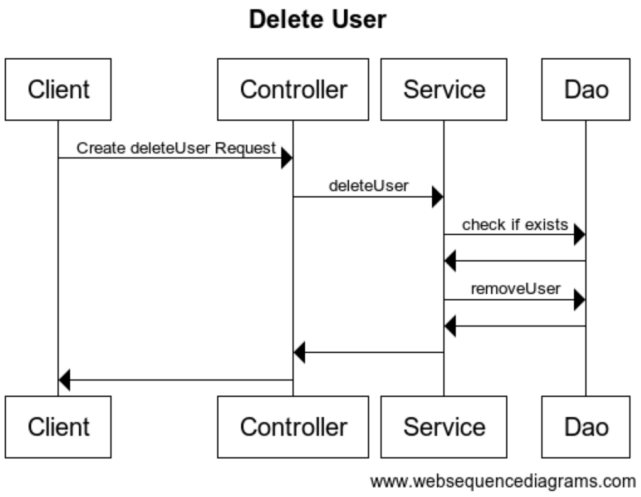
- **Description**: Update the user with all information
- **Flow**:
    - The user clicks "Save"
    - UI sends all entered data to the backend

- The backend will fetch the user from the database and validate the user's information

5.2.5 Delete User

Methods: DELETE

- API Design - Sequence Diagram

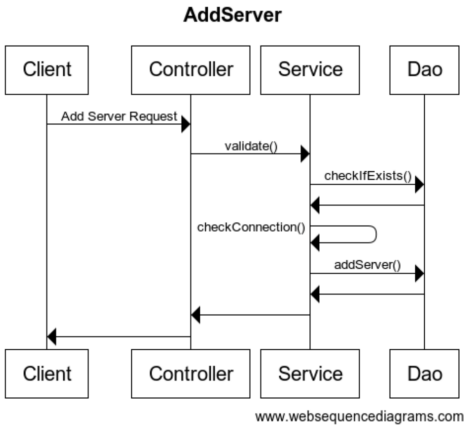


- **Description**: Delete User data from the database
- **Flow**:
  - The back-end gets the delete user request.
  - Back-end check if the user has already exited,
  - If yes, delete the user and return a success message.
  - If not, return a message stating that the user does not exist.

5.2.6 Add Server

Methods: DELETE

- API Design - Sequence Diagram

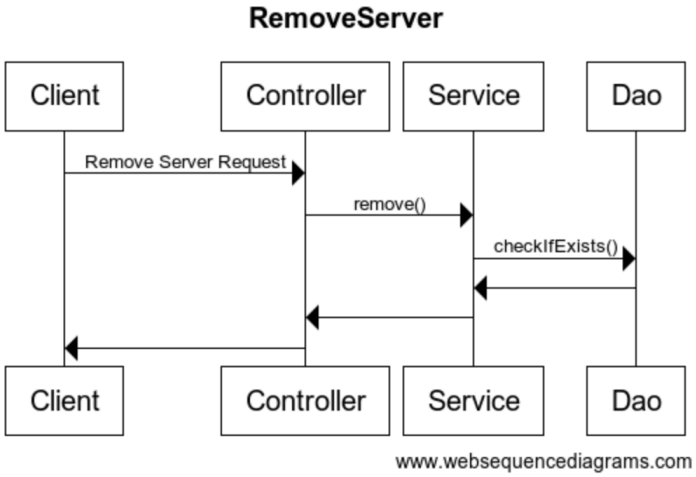


- **Description**: Users can add server information to the database
- **Flow**:
    - The user enters all server information and clicks “Save”
    - The back end will check if the server already exists,
    - If yes, return a message stating the server already exists.
    - If not, check the connection between the application and the server.
    - Then, store the server information in the database.

5.2.7 Remove Server

Methods: DELETE

- API Design - Sequence Diagram



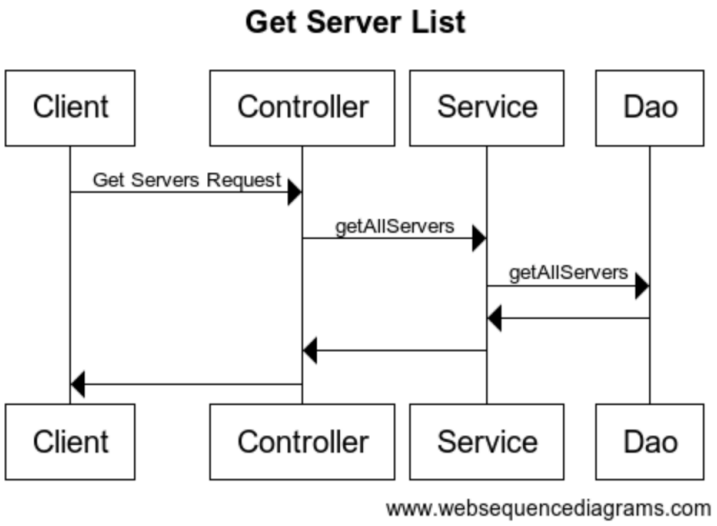
- **Description**: Users can remove servers
- **Flow**:
    - The user clicks “Remove”

- The back end receives the request and checks if the server is stored in the database.
- If yes, remove all information for that server.
- If not, return a message stating that the server does not exist.

5.2.8 Get Server List

Methods: GET

- API Design - Sequence Diagram

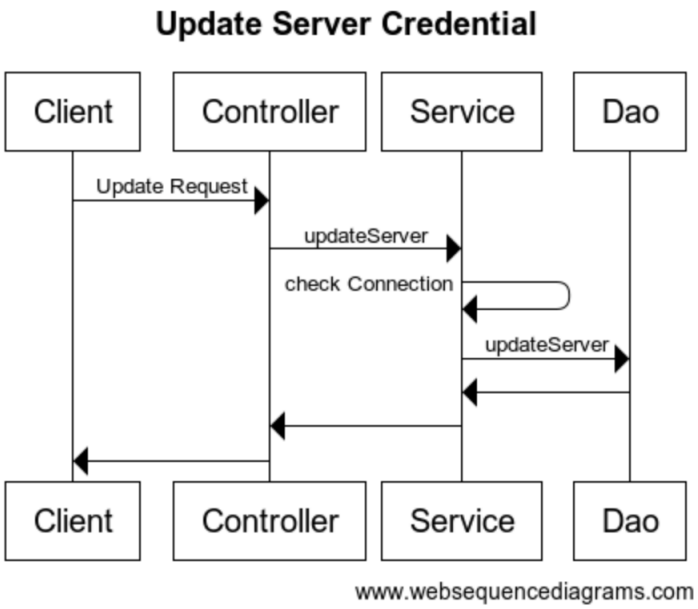


- **Description**: Users can view all servers that have been added as a list.
- **Flow**:
  - The user views the “Server List” page
  - The back end receives the request and fetches all servers’ information.
  - Return to the client to display.

5.2.9 Update Server Credentials

Methods: POST

- API Design - Sequence Diagram

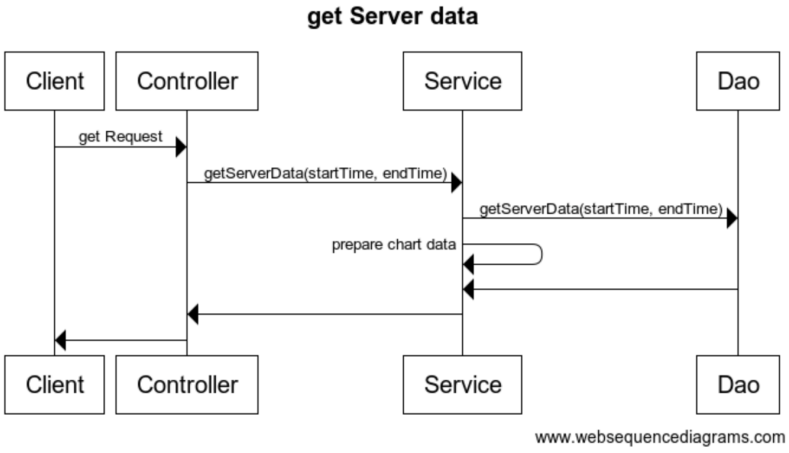


- **Description**: Users can change server credentials when needed.
- **Flow**:
    - The client sends all updated credentials and the request to the back end.
    - The back end will check the connection between the application and the server.
    - The back end will update new credentials and store them in the database.

5.2.10 Get Server Data

Methods: POST

- API Design - Sequence Diagram



- **Description**: Users can get server information.
- **Flow**:

- After the back end receives the request, the back end will call a function “getServerDate(startTime, endTime)”, in which startTime and endTime are the time range for the server data.
- Then, the server will return its data.
- Then the back end will prepare the chart and server data, and send it back to the client.