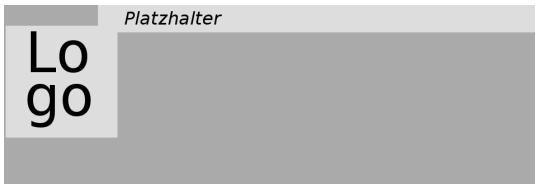


Documentation

Robot Interface
Server Web Application

Author:	Marko Bunteski
Matriculation No.:	xxxxxxx
Course of Studies:	Computer Technologies and Engineering
First examiner:	Prof. Dr. Elmar Wings
Submission date:	August 1, 2024

University of Applied Sciences Emden/Leer · Faculty of Technology · Mechanical Engineering Department
Constantiaplatz 4 · 26723 Emden · `http://www.hs-emden-leer.de`

Contents

List of Figures

1 Introduction

The Robot Interface project is designed to facilitate communication and command execution on a robot via a web application. Built using ASP.NET Core, this application allows users to input the robot's IP address, verify its connectivity, and send commands to be executed on the robot. This document outlines the structure of the project, its main components, and the functionality provided.

2 Project Structure

The project is divided into several key components:

- **Controllers**: Manage HTTP requests and determine the responses to send back to the client.
- **Services**: Contain the core business logic and handle interactions with external systems such as SSH for executing commands on the robot.
- **Models**: Define the data structures used within the application.
- **ViewModels**: Provide data structures used to pass information between views and controllers.
- **Views**: HTML templates rendered by the controllers to present data to the user.

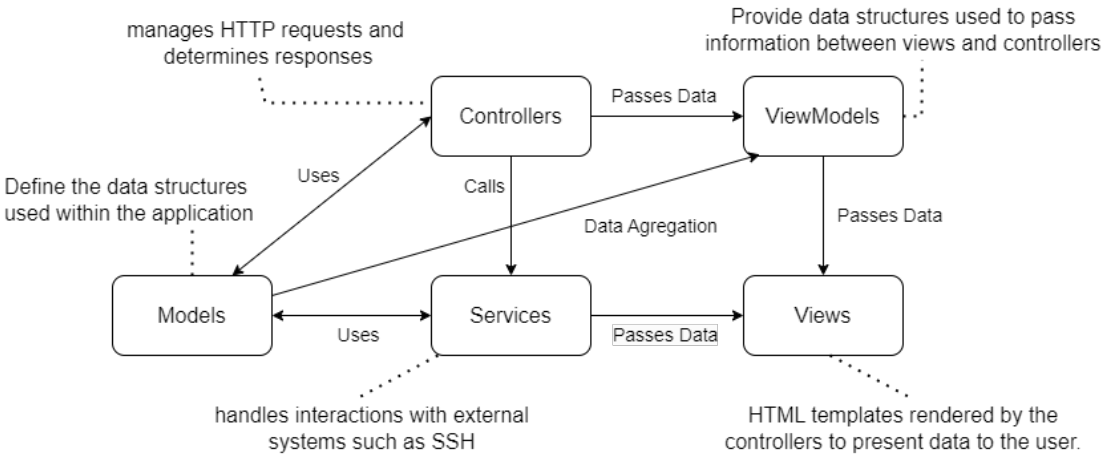


Figure 1: Program Structure

3 Main Components

3.1 Controllers

3.1.1 RobotController

Purpose: Manages the interactions related to setting the robot's IP address and executing commands on the robot.

Key Actions:

- **SetIp**: Allows the user to input the robot's IP address and verifies connectivity by pinging the robot.
- **ExecuteFunction**: Executes a specific command on the robot by function ID. This involves retrieving the command from the database and using the SSH service to execute it on the robot.

3.1.2 DatabaseController

Purpose: Handles database-related operations such as importing data from an Excel file.

Key Actions:

- **Upload**: Provides a web interface for uploading an Excel file.
- **ImportData**: Parses the uploaded Excel file and inserts the data into the database. This is useful for populating the database with predefined functions and commands.

3.1.3 FunctionsController

Purpose: Manages CRUD (Create, Read, Update, Delete) operations for functions and their associated commands.

Key Actions:

- **Index**: Lists all functions in the database, providing options to create, edit, delete, and execute functions.
- **Create**: Displays a form for creating a new function. This includes selecting commands and libraries associated with the function.
- **Edit**: Displays a form for editing an existing function and its associated commands and libraries.
- **Delete**: Deletes a function from the database.
- **Details**: Displays detailed information about a specific function, including its associated commands and libraries.

3.2 Services

3.2.1 SshService

Purpose: Handles SSH connections and command execution on the robot.
Functionality:

- **SetHost**: Sets the IP address of the robot for SSH connections.
- **TestConnection**: Tests the SSH connection to ensure it can successfully connect to the robot.
- **ExecuteCommand**: Executes specified commands on the robot via SSH.

3.2.2 RobotService

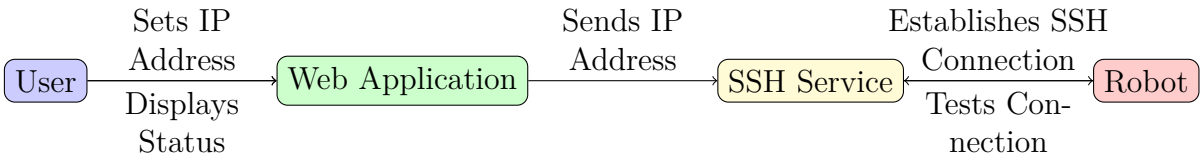
Purpose: Interacts with the SSH service to execute commands on the robot.
Functionality:

- **ExecuteCommand**: Retrieves the command associated with a given function ID from the database and executes it on the robot. If the Docker container is not already running, it starts the container before executing the command.

4 Establishing an SSH Connection

To interact with the robot, the application must establish an SSH connection. The steps involved are:

- **Setting the Host**: The user sets the robot's IP address via the web interface.
- **Connecting to SSH**: The `SshService` uses the provided IP address, along with predefined username and password, to establish an SSH connection.
- **Testing the Connection**: A test command is executed to ensure the connection is successful.



5 Executing Commands in Docker Container

Commands are executed within a Docker container on the robot to ensure they run in the correct environment:

- **Find Latest Container**: The application checks for the latest running Docker container.

- **Start Container if Needed**: If no container is running, a new Docker container is started.
- **Create and Execute Script**: A script is created inside the container, given execution permissions, and then run. This script typically includes sourcing the ROS setup script, navigating to the workspace, sourcing the workspace setup script, and executing the desired command.

6 Database Interaction

The application uses a database to manage functions and commands:

- **CRUD Operations**: The application allows users to Create, Read, Update, and Delete functions and commands through the web interface.
- **Database Population**: Users can populate the database by uploading an Excel document, which is parsed and inserted into the database.

7 Models and Relationships

The application defines several models to represent data:

- **Function**: Represents a function that can be executed on the robot. Each function has a name and belongs to a category.
- **Command**: Represents a command that can be executed on the robot.one or more commands belong in one function which is executed by executing every single function one by one.
- **Category**: Represents a category to which functions belong. (MediaPipe, Visual Interaction, LiDAR)
- **FunctionCommand**: Represents the many-to-many relationship between functions and commands.
- **FunctionLibrary**: Represents the many-to-many relationship between functions and libraries.
- **FunctionCategory**: Represents the many-to-one relationship between functions and Category.

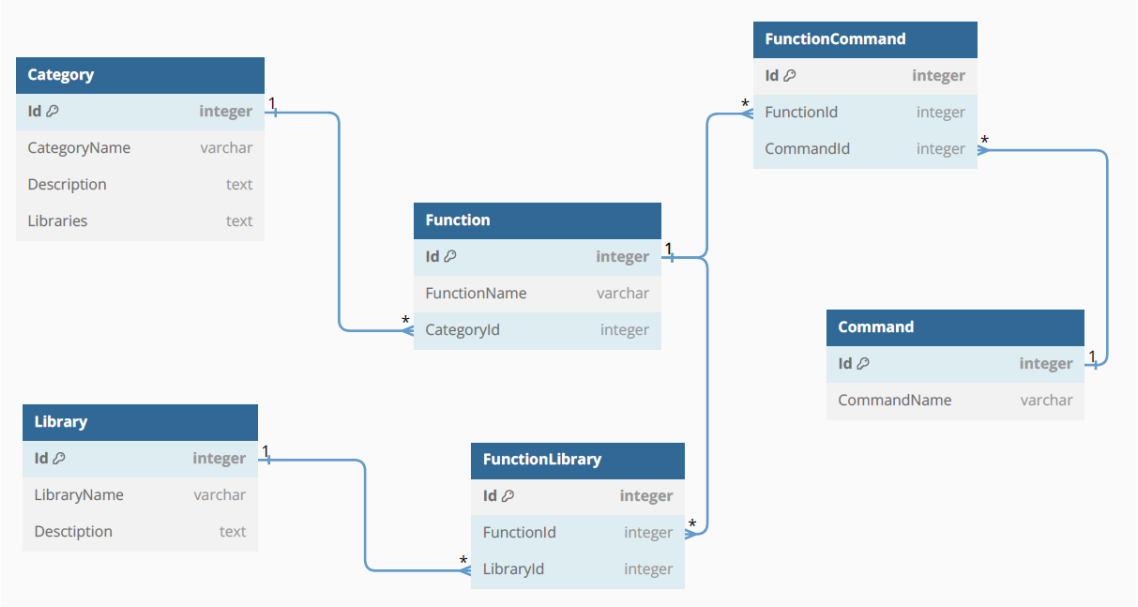


Figure 2: Database Relations

8 Views

The views provide the user interface for interacting with the application:

- **Set IP View**: Allows the user to input and set the robot's IP address.
- **Upload View**: Allows the user to input an Excel document which is used to populate the database.(a prefabricated document will be available containing all the functions with he appropriate relational connections with different models)
- **Function Index View**: Lists all functions and provides options to create, edit, delete, and execute functions. also connect the functions with the needed libraries and Commands as well as add them to the appropriate Category.

9 WebSockets and Real-time Communication

The application uses SignalR to enable real-time communication between the client and server:

- **SignalR Hub**: The `RobotHub` class manages real-time communication.
- **Client-Side Interaction**: JavaScript on the client-side connects to the SignalR hub to send commands and receive responses in real-time.

10 Conclusion

The Robot Interface project provides a user-friendly web interface for interacting with a robot. By leveraging SSH and Docker, it allows users to set the robot's IP address, verify connectivity, and execute commands in a controlled environment. This setup ensures that commands are executed reliably and in the correct context on the robot.

This document should give you a clear understanding of the purpose of the project and how its various components work together to achieve the desired functionality.