AxAi documentation

Richard Sindely

2025

Contents

1 Introduction

This program is a task assigned by AgileXpert, this includes the basic tasks and the assignments for AI role applicants. You can access the tasks description here.

1.1 Technical information

The application is written in Java using the Spring Boot framework and communicates with the user through a Command-Line Interface (CLI). It includes an AI feature that allows the user to request command execution through natural language.

The backend uses a MySQL database. Hibernate (JPA) is used to handle data storage, so there is no need to write raw SQL queries manually.

The data model is implemented using classes annotated with @Entity, which marks them as JPA entities mapped to the database tables. In some cases, the @Transactional annotation was necessary to access data from related tables due to issues with lazy loading. Using eager fetching would be a solution, but it would load unnecessary data that could lead to inefficient resource usage. UUIDs are used in the project because the chance of generating duplicates is extremely low. UUIDs are automatically generated for all entities.

The SQL command for user creation can be found in the README.md. It is required in order to connect to the database. The application automatically creates a database named axaidb if it does not already exist.

The AI feature calls the OpenAI API, which returns a command to be executed. The application requires an API KEY, which can be added to the root directory in a file named "openai.env".

1.2 Setup

- Clone the project.
- Install MySQL.
- Create a MySQL user (read the README.md for the command).
- Run the application by executing mvn spring-boot:run in the root directory of the project.

2 Database

2.1 Entities

User entity:

- id (PK)
- username
- password (Passwords are stored in a hashed format.)
- theme_id (User's theme)
- selected_background_id (FK)
- menu_id (FK)

Background entity

- id (PK)
- name (background's name)
- user_id (FK)

Menu entity

- id (PK)
- name

SubMenu entity

- id (PK)
- name (Submenu's name)
- menu_id (FK)

Submenu-App relationship table:

- submenu_id (FK)
- app_id (FK)

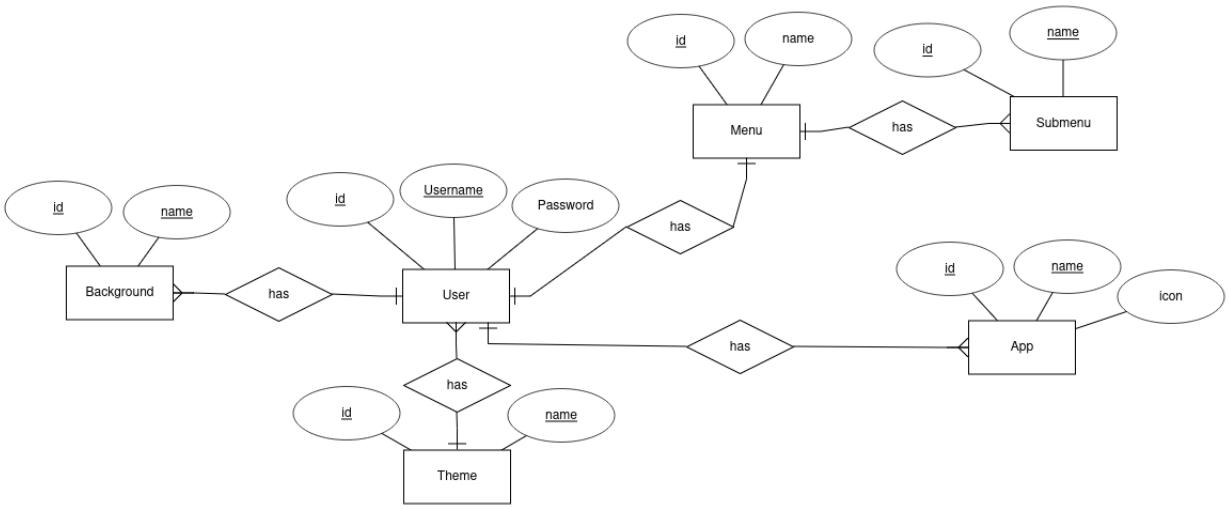
Theme entity

- id (PK)
- name (Theme's name)

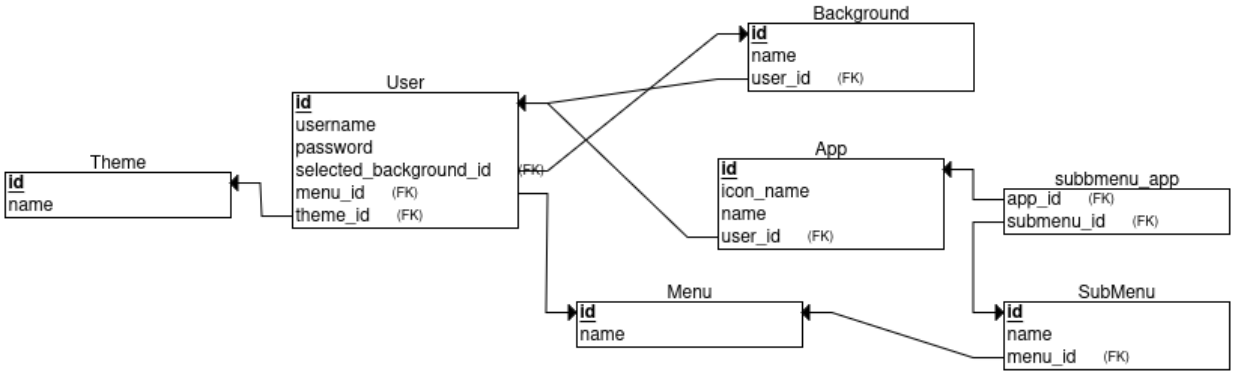
App entity

- id (PK)
- icon_name
- name (App's name)
- user_id (FK)

2.2 ER diagram



2.3 RS diagram



3 Structure

The project uses a repository layer to interact with the database. The service layer communicates with the repositories to retrieve, update, and manage data. These repositories are annotated with @Repository,the services with @Service, and each entity has a corresponding repository interface.

The application currently uses a CLI interface, but a controller can be added later if a frontend is needed.

4 Commands

If the user is not logged-in:

- Create a user: create-user userName password
- Login: login
- Exit: exit:

If the user is logged-in:

- Logout: logout
- Exit: exit
- Edit your username: rename-user newName passwd
- Change your password: change-password password newPassword
- Delete your user: delete-user passwd
- List your menu:menu
- Create new submenu: create-menu menuName
- Rename submenu: rename-menu oldName NewName
- Delete submenu: delete-menu menuName
- Add app to submenu: add-app menuName appName
- Remove app from submenu: remove-app-menu menuName appName
- Create app: download-app appName iconName
- Delete app: delete-app appName
- Update app icon: update-app-icon appName newIconName
- Delete app icon: delete-app-icon appName
- Run an app: run-app appName
- Set your theme: set-theme themeName

- Add theme: add-theme themeName
- Add background: add-background backgroundName
- Choose background: set-background backgroundName