**Software Architecture Document**

**Afternoon Meetings on A.I.**

|  |
| --- |
| **Date : 16.04.2024** |
| **Version : 1.0** |
| **State : In progress** |
| **Author : Group 1** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Version** | **Date** | **Author(s)** | **Changes** | **State** |
| 1.0 | 16.04.2024 | Catalin Mihai Popoiu | Added initial version and added C1 – C3 level diagrams | In progress |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**Table of contents:**

[1. Introduction 3](#_Toc145410716)

[1.1 Purpose 3](#_Toc145410717)

[1.2 Scope 3](#_Toc145410718)

[2. System Context 3](#_Toc145410722)

[2.1 Business Context 4](#_Toc145410723)

[3. Containers and Technology Choices 5](#_Toc145410725)

[3.1 Backend Container 5](#_Toc145410726)

[3.2 Frontend Container 5](#_Toc145410727)

[4. Components 6](#_Toc145410729)

[4.1 Backend Components 6](#_Toc145410730)

[4.2 Frontend Components](#_Toc145410730) 8

[5. Conclusion](#_Toc145410729) 8

**1. Introduction**

***1.1 Purpose***

This Software Architecture Document provides a comprehensive architectural overview of the AMAI system, outlining the structure, components, and technology stack used to support AI enthusiasts in their engagement with events, discussions, and presentations.

***1.2 Scope***

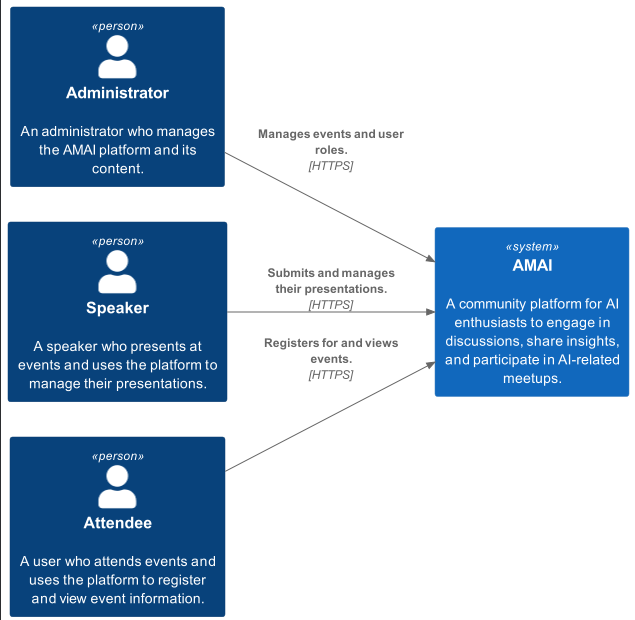
AMAI is designed as a community platform that facilitates interaction among AI enthusiasts, offering features for event management, presentation handling, and user interactions. It integrates various services for a cohesive user experience, leveraging modern web technologies for scalability and maintainability.

**2. System Context**

***2.1 Business Context***

Based in Eindhoven, The Netherlands, AMAI seeks to bridge the gap between AI professionals and enthusiasts by providing a robust platform where users can share knowledge, schedule events, and enhance their professional network.

***2.2 System Overview***

The system is structured around a central web application supported by a backend API, both interfacing with a relational database and various external services for notifications and authentication.

**3. Containers and Technology Choices**

***3.1 Backend Container***

***3.1.1 Spring Boot Application***

Description: Manages core business logic and data interactions for AMAI.

Technology Choices:  
- Spring Boot: Selected for its rapid development features, auto-configuration, and extensive Spring ecosystem support, making it ideal for creating microservices.

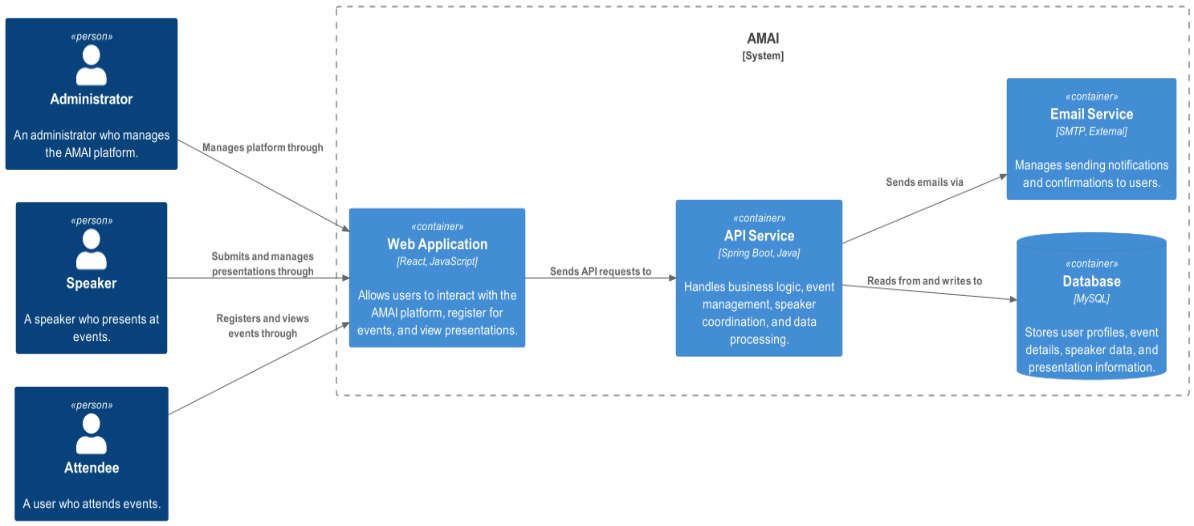
- RESTful API: Ensures statelessness and a uniform interface, facilitating easy integration and communication between frontend and backend components.

***3.2 Frontend Container***

***3.2.1 React Application***

Description: Provides an interactive and responsive user interface.

Technology Choices:

- React: Chosen for its efficiency in updating and rendering components, which is crucial for real-time interactions on the platform. The decision is also influenced by its component-based architecture, facilitating easier development and maintenance.

**4. Components**

**4.1 Backend Components**

The backend architecture consists of three layers, Persistence, Business, and Controller. Their description is as follows:

**Persistence:**

Responsibility: Manages data storage and retrieval, interacting with the MySQL database to ensure efficient and secure data handling.

Components:

* Entity Classes: Map to MySQL tables for data representation.
* Repositories: Leverage JpaRepository for ORM, simplifying database interactions.

**Business:**

Responsibility: Houses the application's core logic, processing data from the Persistence layer for use in the application.

Components:

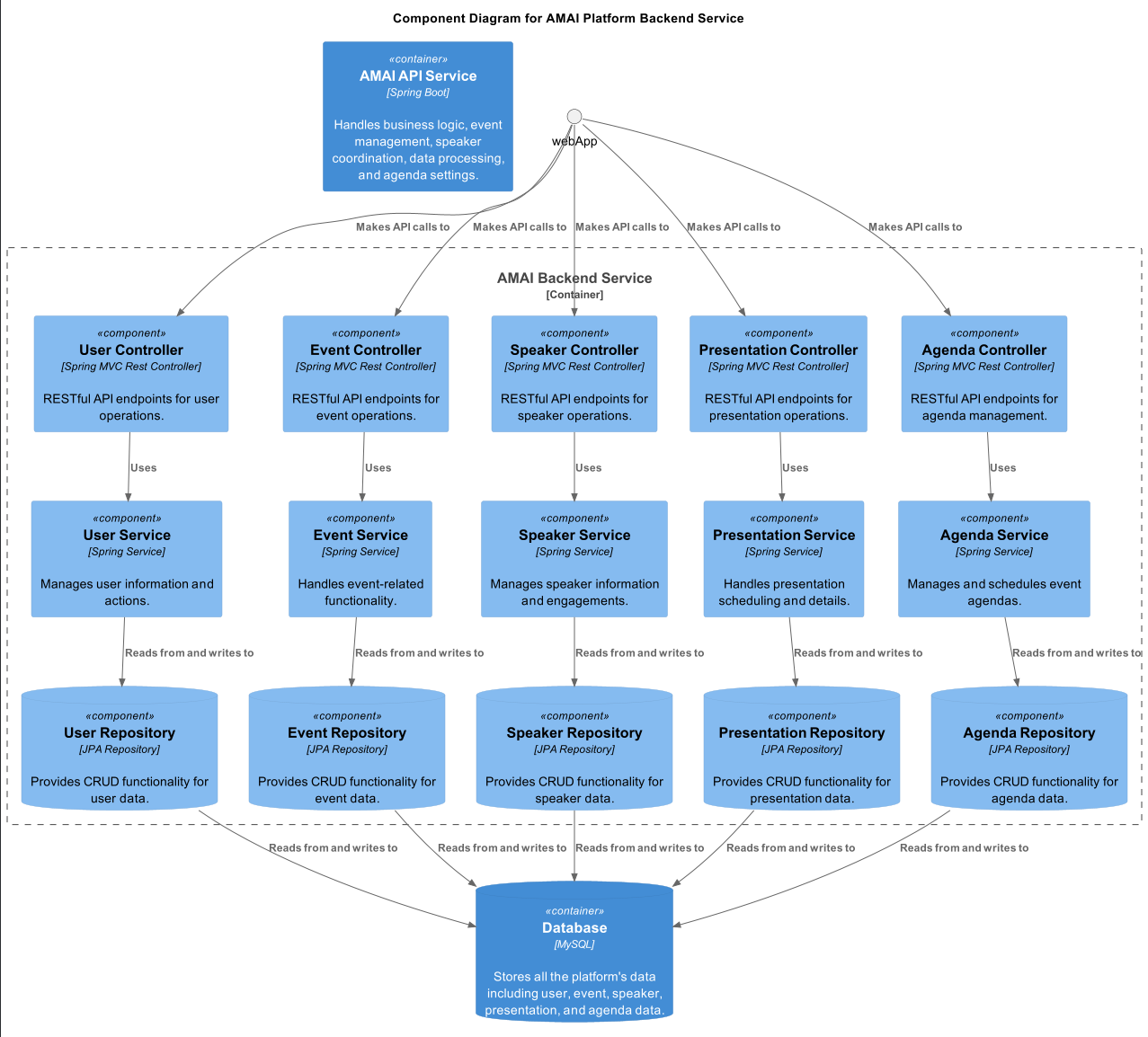
* Service Classes: Contain business logic, transforming data for the Controller layer.
* DTOs (Data Transfer Objects): Facilitate data transfer within the application, adhering to the YAGNI principle by avoiding unnecessary base classes.

**Controller:**

Responsibility: Manages data flow between the UI and business logic, processing user inputs and returning the appropriate responses.

Components:

* Controllers: Interface with both the business layer and the frontend, orchestrating the application's response to user actions.



**4.2 Frontend Components**

**API:**

Responsibility: Handles API calls to the backend, employing HTTP methods to interact with the server.

**Components:**

Responsibility: Serves as the reusable building blocks of the UI, facilitating the creation of dynamic and interactive pages.

**Pages:**

Responsibility: Renders the application's various views, utilizing components for displaying content and interfacing with the user.

**5. Conclusion**

This document provides a detailed description of the AMAI platform's architecture, emphasizing the use of modern, scalable technologies such as Spring Boot and React. By adhering to well-established architectural principles, AMAI aims to offer a robust and user-friendly platform that fosters the AI community's growth and engagement.