**Architecture constraints, design decisions and C4**

***Emergency Web App***

*Netherlands Citizens*

Incorporate draw.io project: <https://drive.google.com/file/d/1g5n0cqVyDJoiDgYN8PKMzS_iAOfFgl2X/view?usp=sharing>

|  |
| --- |
| **Date : 10 October** |
| **Version : 1** |
| **State : Ongoing** |
| **Author : Jennifer Oentoro** |

Contents

[1. Architecture constraints and design decisions 3](#_Toc42673512)

[2. C4 Model diagrams 6](#_Toc42673519)

# Architecture constraints and design decisions

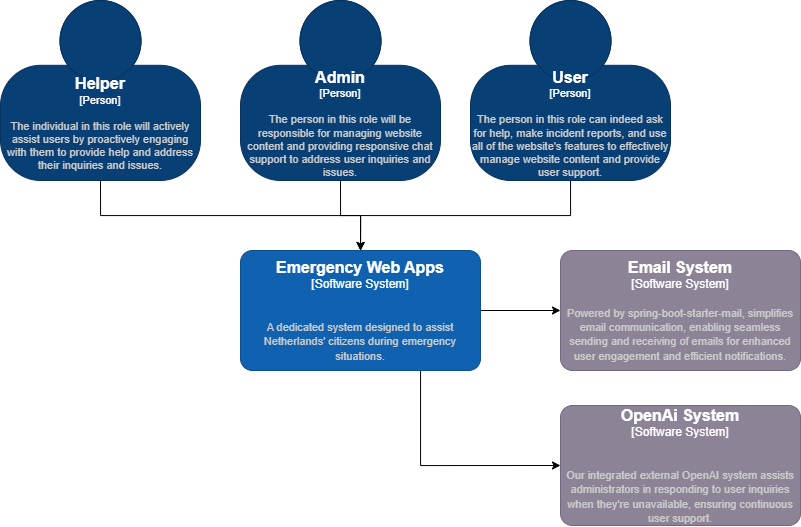
1. **Spring Boot (Backend Framework):**
   * Pro:
     + **Scalability:** Spring Boot is a popular choice for building backend services due to its scalability. It allows for the easy creation of microservices, which can be individually scaled to handle increased load as the application grows.
     + **Robustness:** Spring Boot provides a comprehensive set of tools for building robust and maintainable applications. It offers features like dependency injection, aspect-oriented programming, and built-in support for handling common concerns like security and transactions.
     + **Community and Ecosystem:** The Spring framework has a large and active community, which means there are plenty of resources, tutorials, and third-party libraries available. This can speed up development and troubleshooting.
     + **Integration:** Spring Boot is well-suited for integrating with various data sources and services, making it a good choice if your application needs to interact with different databases or external APIs.
   * Architecture:
     + Using 3 layers:
       1. Controller:
          - The controller layer is responsible for handling incoming HTTP requests, routing them to the appropriate business logic, and returning HTTP responses to the client. It acts as the entry point for client interactions with the application.
          - Use data transfer object to pass in / out the controllers.
       2. Business:
          - The business layer, also known as the service layer, contains the core application logic. It encapsulates the business rules, processes, and operations of the application. This layer is responsible for processing data and performing any necessary business logic.
       3. Persistence:
          - Repository: A repository is an interface that extends Spring Data JPA's JpaRepository or a similar interface from Spring Data for other data storage technologies. It provides high-level, abstracted methods for performing database operations related to the entity.
          - Entity: An entity is a Java class that represents a persistent object in the application. Each entity typically corresponds to a table in the database. Entities are used to model and map the structure of data in the database.
   * Using security JWT & Oauth2
2. **React (Frontend Library):**
   * Pro:
     + **User Experience:** React is known for creating highly responsive and interactive user interfaces. Its virtual DOM and component-based architecture make it easy to develop complex UIs that provide a smooth user experience.
     + **Component Reusability:** React promotes the reusability of UI components. This can save development time and maintainability by allowing you to build a consistent UI across the application.
     + **Performance:** React's virtual DOM efficiently updates only the parts of the UI that have changed, improving application performance. It's a good choice for single-page applications (SPAs) where quick updates are essential.
     + **Developer Productivity:** React has a large developer community, a rich ecosystem of libraries, and tools like Redux for state management. This can boost developer productivity and simplify development tasks.
   * Architecture:
     + APIs: APIs, in the context of a React application, refer to the external interfaces or services used to fetch and send data to and from a backend server or third-party services. These APIs provide the data necessary to render the user interface and manage application state.
     + Components: Components are the building blocks of a React application. They represent reusable, self-contained UI elements that can be combined to create complex user interfaces. Components are at the core of React's component-based architecture.
     + Pages: Pages in a React application represent the different views or routes that users can navigate to. Each page is typically composed of one or more components and is responsible for defining the layout and structure of a specific part of the application.
3. **MySQL (Database):**
   * **Relational Data:** MySQL is a relational database management system (RDBMS) that excels at handling structured data. If your application requires complex queries or transactions, MySQL is a suitable choice.
   * **Data Integrity:** MySQL provides features such as foreign key constraints, transactions, and ACID compliance, ensuring data integrity and consistency.
   * **Scalability:** While MySQL can be used for small to medium-sized applications, it can also scale up to handle larger datasets and high traffic with proper optimization and sharding strategies.
   * **Community and Support:** MySQL has a large user base and a well-established support ecosystem, including documentation, forums, and third-party tools.
   * **Architecture (temporary for now):**

A screenshot of a computer

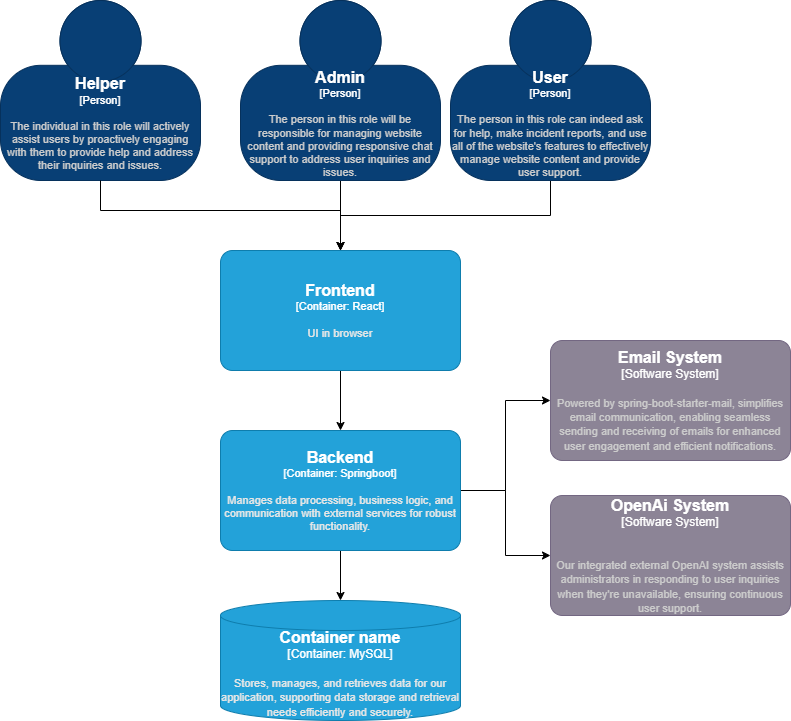
Description automatically generated

# C4 Model Diagrams

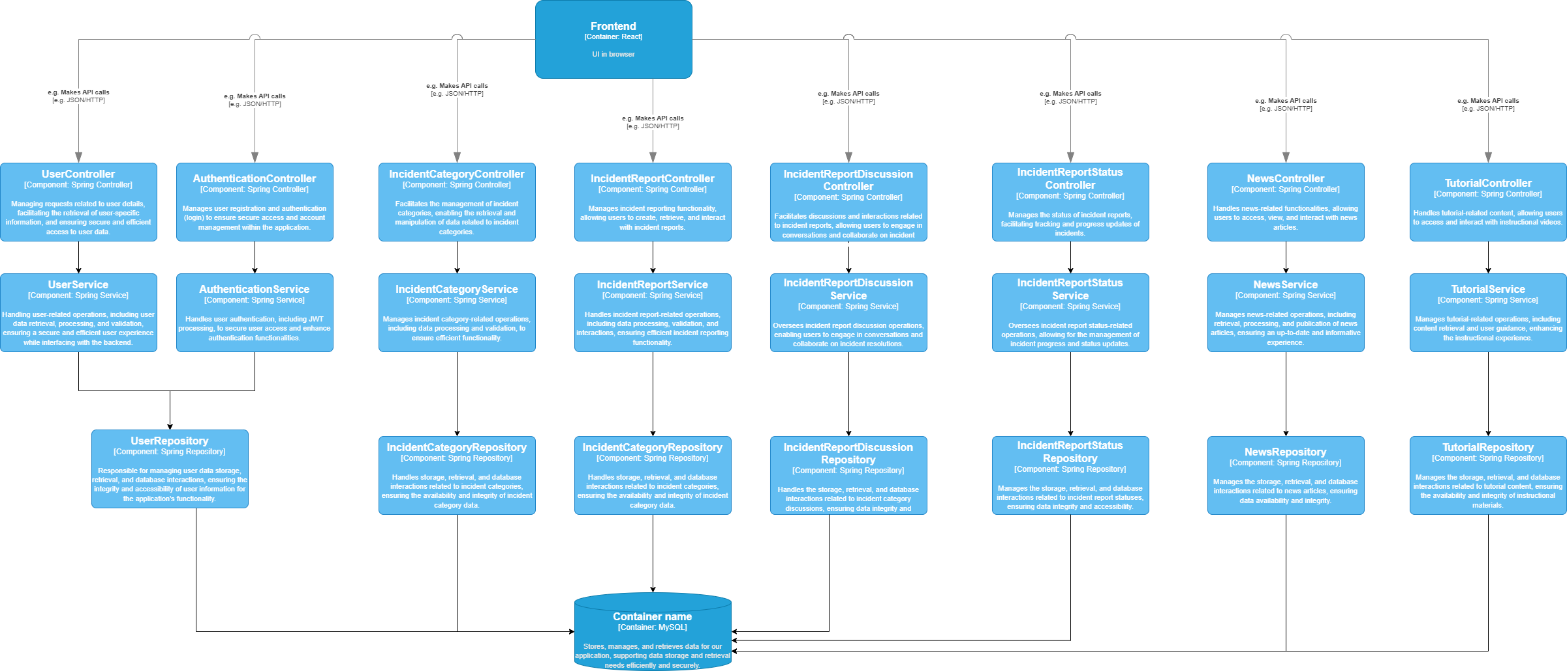
* + - 1. **C1 – Context**



* + - 1. **C2 – Container**



* + - 1. **C3 – Component Backend**

****