Architecture Document

Key Watch

BAS World

Date	**7/10/2024**
Version	**0.1**
State	**Set up**
Author	**Nameless**

Contents

Architecture 3
1.1 Design Choices 3
1.1.1 Back-end 3
1.1.2 Front-end 6

Architecture

1.1 Design Choices

This will indicate the choices, tools and technologies that will be used for this project.

1.1.1 Back-end

- **Spring Boot:** Simplifies Java backend development by reducing boilerplate code and offering built-in support for RESTful APIs and microservices, making it ideal for scalable applications.
- **Rest Template:** Facilitates HTTP communication by streamlining data retrieval from external APIs and automatically deserializing responses into Java objects, ensuring seamless integration.
- **Lombok:** Reduces boilerplate in Java code by generating common methods like getters and setters, enhancing code clarity and maintainability, especially in projects with many model classes.
- **JUnit:** Employed for unit testing, ensuring code reliability through automated tests and supporting test-driven development (TDD), which helps maintain high code quality and reduce bugs.
- **Docker:** Used for containerizing applications, allowing for consistent environments across development, testing, and production. It simplifies deployment and scaling.
- **Continuous Integration Runner**: Used to automate the testing and deployment processes, facilitating the integration of changes made by multiple developers. It ensures that the application builds correctly and automates testing to catch issues early, promoting a robust development workflow.
- **Gradle:** Selected as the build automation tool for its flexibility and powerful dependency management capabilities. It simplifies project builds and supports multi-project setups, enabling efficient management of complex applications while enhancing build performance with incremental builds.
- **ModelMapper**: Used for simplifying the object mapping process between different data models, ModelMapper facilitates the conversion of one object type to another, such as DTOs to entity objects and vice versa. By reducing the amount of manual mapping code, it enhances code maintainability and readability, while also supporting complex mappings and configurations to handle nested structures and custom transformations effectively.
- **Jakarta Validation**: Provides a framework for validating JavaBeans, allowing developers to enforce constraints on data models through annotations. It helps ensure data integrity by allowing the definition of validation rules.

Choices

Interfaces within Business Layer: Interfaces are implemented in the logic layer to facilitate dependency inversion for the Data Access Layer (DAL). This organization supports better separation of concerns and enhances the overall architecture by allowing for easier testing and maintainability.

Mapping in Services Instead of Controllers: Services are responsible for processing data and implementing business logic, while controllers focus on exposing data to clients. Therefore, it is more appropriate to handle mapping within the services, as this keeps the controllers streamlined and maintains a clear separation of concerns.

SOLID Framework: A set of principles aimed at creating maintainable and scalable software, emphasizing single responsibility, open/closed design, Liskov substitution, interface segregation, and dependency inversion.

3-Layer Architecture: A software design pattern that divides an application into three distinct layers: the presentation layer for user interaction, the business logic layer for processing and managing data, and the data access layer for interacting with external data sources or databases.

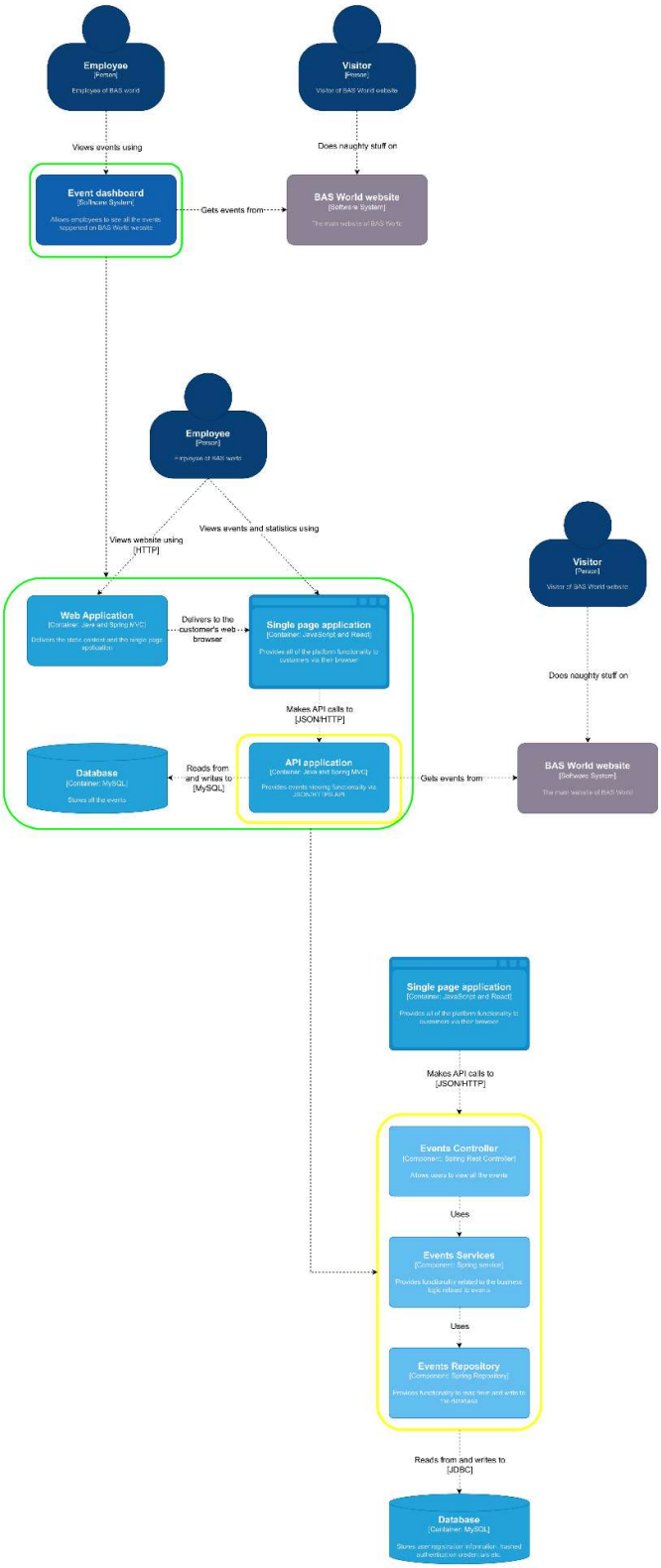
Breakdown

```
src
└── main
    ├── java
    │   └── com
    │       └── fontys
    │           └── nameless_backend
    │               ├── apiclients
    │               │   └── ..
    │               ├── business
    │               │   ├── ..
    │               │   ├── impl
    │               │   │   └── ..
    │               ├── configuration
    │               │   └── ..
    │               ├── controller
    │               │   ├── requests
    │               │   │   └── ..
    │               │   └── responses
    │               │       └── ..
    │               ├── domain
    │               │   ├── definitions
    │               │   │   ├── data
    │               │   │   │   └── ..
    │               │   │   └── ..
    │               │   └── ..
    │               └── persistence
    │                   └── ..
    └── resources
        └── application.properties
```

1.1.2 Front-end

- **React:** Chosen for its efficient component-based architecture, allowing for reusable and maintainable UI code, which enhances interactivity and user experience.
- **Tailwind CSS:** Utilized as a utility-first CSS framework that promotes rapid UI development through its predefined classes. It enables the creation of responsive designs directly in the markup, enhancing maintainability and consistency while reducing the need for custom CSS.
- **ShadCN**: A modern component library for React that offers a collection of customizable, pre-built components. It focuses on accessibility and responsiveness, enhancing UI development with flexible styling options and promoting consistency in design.

Choices

CSS Frameworks: Utilizing a CSS framework enables faster development by providing pre-built styles and components, allowing for quick implementation of design elements without the need to create them from scratch.

1.2 **C4 diagram**