A-SIA 2.0

Documentation

2022

UAS Technikum Wien

Inhalt

[1 System Definition 3](#_Toc107225325)

[1.1 System Boundary 3](#_Toc107225326)

[1.2 System Architecture 4](#_Toc107225327)

[1.2.1 Server Architecture 4](#_Toc107225328)

[1.3 Models 5](#_Toc107225329)

[2 Backend (C#) 5](#_Toc107225330)

[2.1 ASP.Net Server 5](#_Toc107225331)

[2.1.1 Controller 5](#_Toc107225332)

[2.1.2 Authentication 9](#_Toc107225333)

[2.2 Neo4J 9](#_Toc107225334)

[2.2.1 DAL Layer Definition 9](#_Toc107225335)

[2.2.2 Repositories 9](#_Toc107225336)

[2.2.3 Neo4J Engine / Database 9](#_Toc107225337)

[2.3 Graph Calculation 9](#_Toc107225338)

[2.3.1 Role Calculation 9](#_Toc107225339)

[2.3.2 Value Calculation 9](#_Toc107225340)

[2.4 Plugin System 9](#_Toc107225341)

[2.5 Configuration 9](#_Toc107225342)

[2.5.1 ASP.Net 9](#_Toc107225343)

[2.5.2 DAL Layer 9](#_Toc107225344)

[2.6 Deployment 9](#_Toc107225345)

[2.6.1 Docker 9](#_Toc107225346)

[2.6.2 Manual 9](#_Toc107225347)

[3 Frontend (Angular) 9](#_Toc107225348)

[4 Glossary 9](#_Toc107225349)

# System Definition

## System Boundary

The following diagram shows the predefined System Context Boundary. It ensures that only the required functionalities are considered when looking at the development and maintenance of the A-SIA 2.0 application.

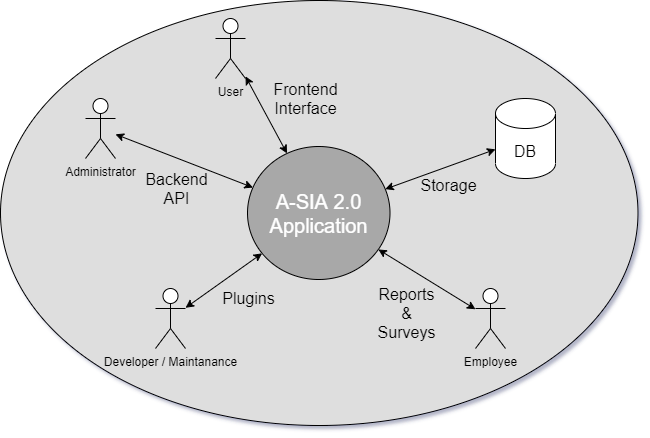


Figure 1 System Boundary Diagram

**Frontend Interface**

<Description>

**Backend API**

The backend API is responsible for receiving calls that affect the calculation and management of the networks and user. It should be an easy-to-use interface like swagger/openapi and should also be used by the frontend as access point to the backend. Additionally, the interface should be defined as REST service.

**Storage**

The storage system needs to save the data that is managed and used by the system in a database. The database can be chosen freely, but it must be possible to display relations and graph like structures in the database. Neo4J would be a good choice for such database.

**Plugins**

The system must be extendable via external plugins. These plugins should be able to be added after compiled and loaded after restarting the application, without changing the code.

**Reports & Surveys**

<Description>

## System Architecture

The following chapter shows the different system architectures of the frontend and backend components of the application. The architecture overviews should be described with words and displayed with an UML diagram for better understanding.

### Server Architecture

The server architecture is split into different parts. First there is the Neo4J database as external system. Then there are the ASP.Net REST server, the calculation system, plugin system and the DAL Layer of the server that connects to the external Neo4J database. The following UML diagram shows the system overviews with focus on the ASP.Net REST server.

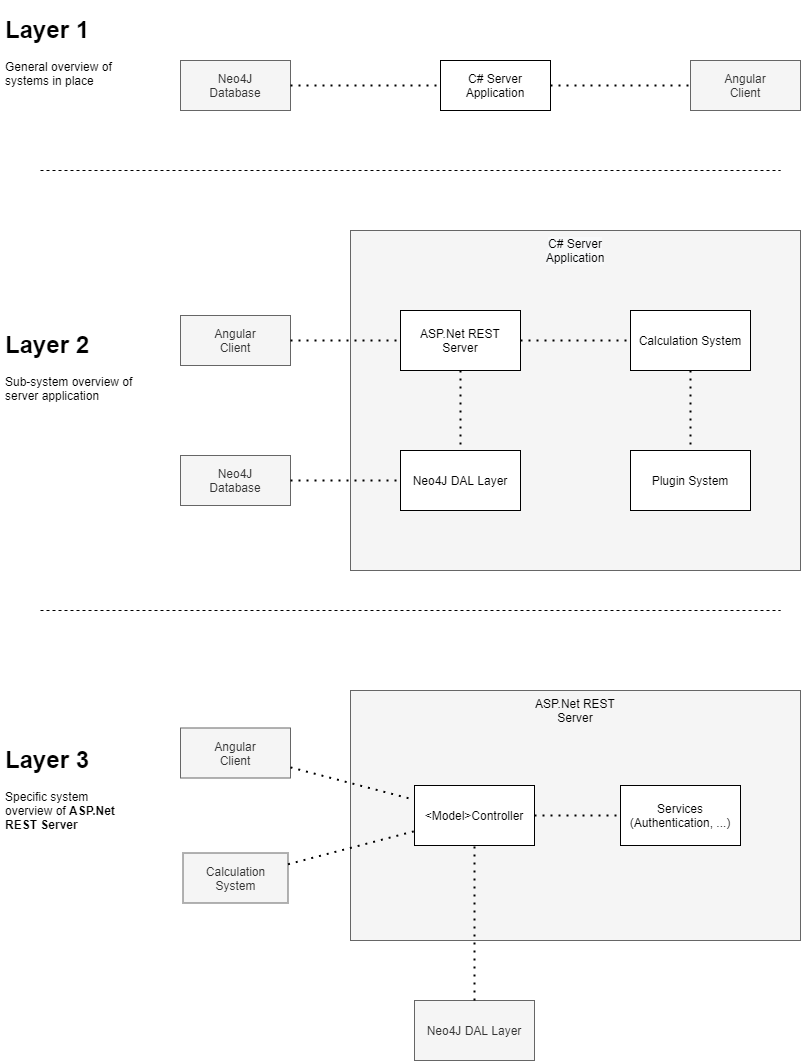


Figure 2 Diagram of backend architecture with detailed view of REST server

## Models

The models of the system can change over the runtime of the project and therefore the following diagram might be outdated.

Graphical user interface

Description automatically generated

Figure 3 System models diagram

# Backend (C#)

## ASP.Net Server

### Controller

The following diagrams show the controller members and methods. It provides a general overview of the controllers in use and their structural information.

#### Extended Controller Base

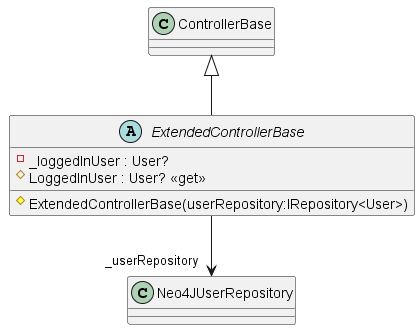


Figure 4 Extended Controller Base class diagram

Extends the controller base by adding a getter for the logged-in user.

#### User Controller

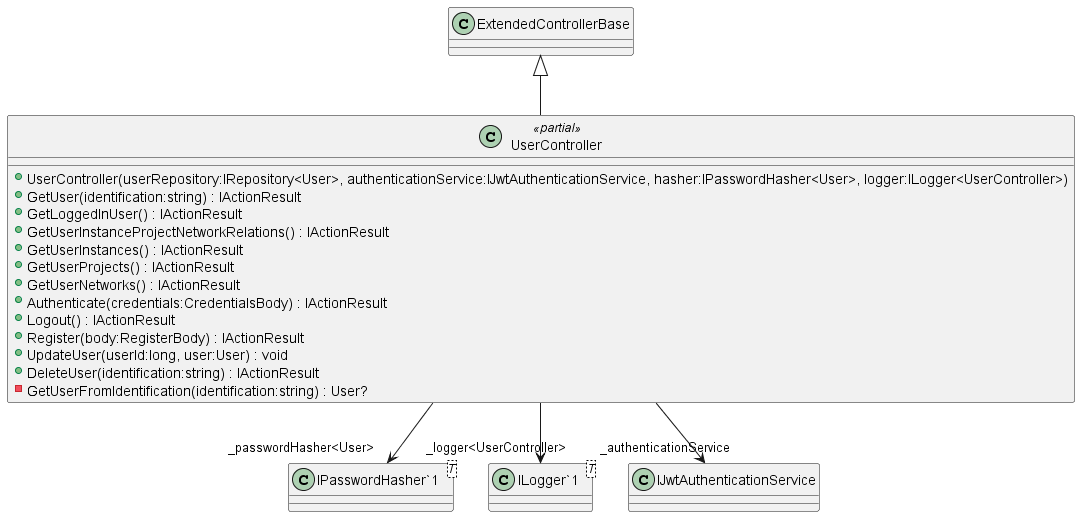


Figure 5 User Controller class diagram

Provides user CRUD operations, as well as the user authentication and user instance management (e.g., GetUserInstances).

#### Instance Controller

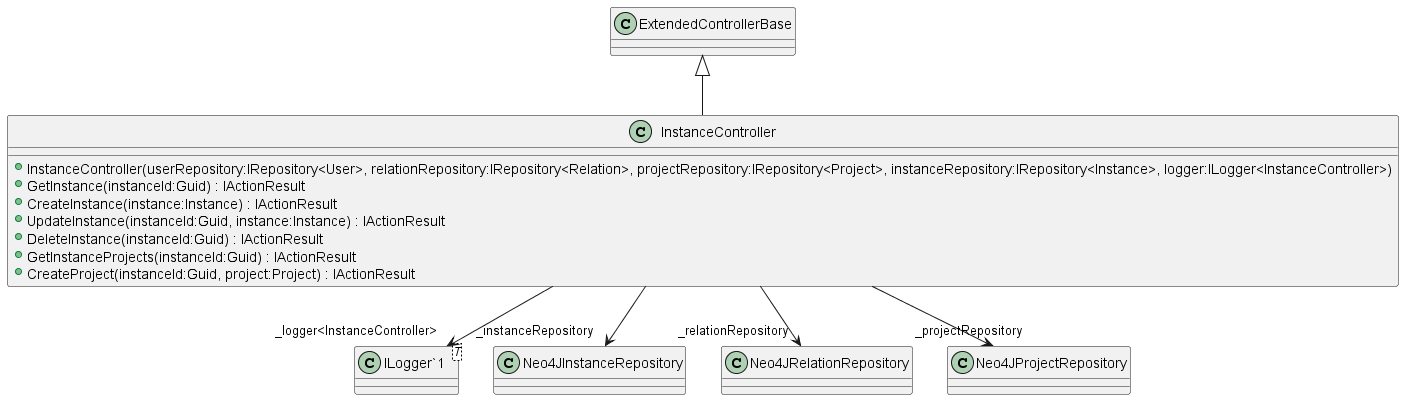


Figure 6 Instance Controller class diagram

Provides instance CRUD operations and project creation.

#### Project Controller

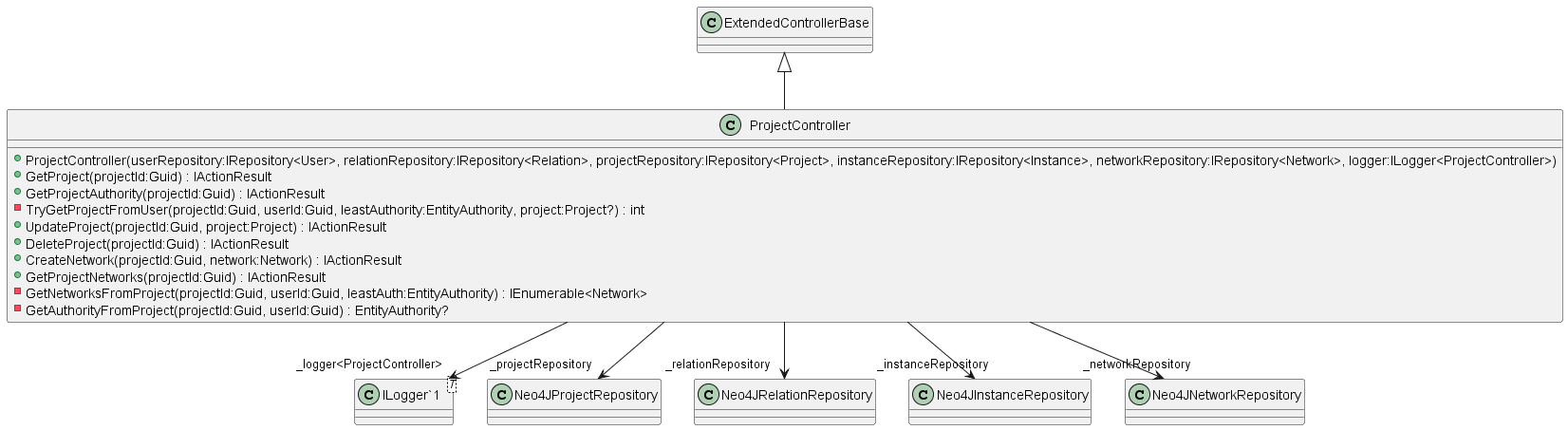


Figure 7 Project Controller class diagram

#### Network Controller

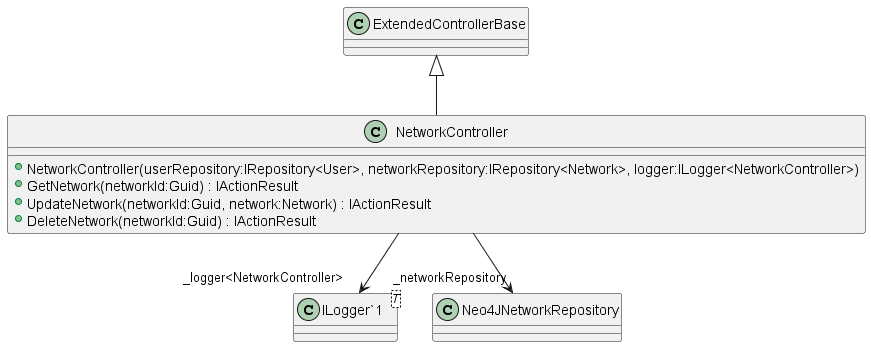


Figure 8 Network Controller class diagram

#### Person Controller

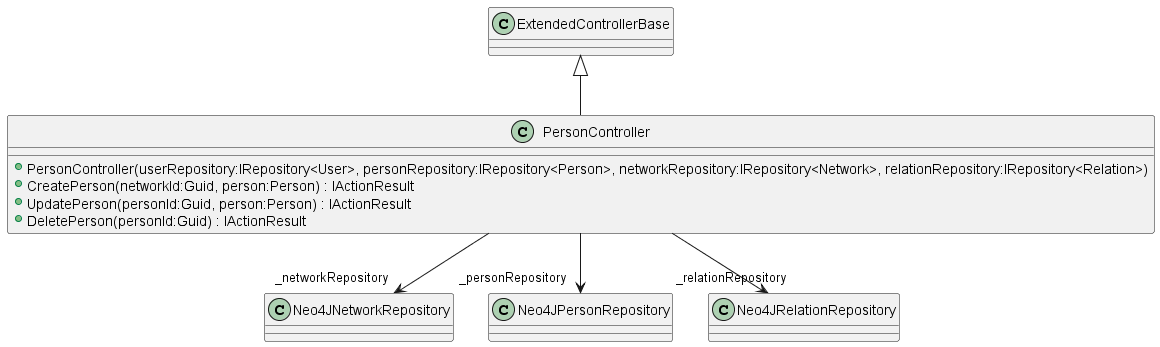


Figure 9 Person Controller class diagram

#### Relation Controller

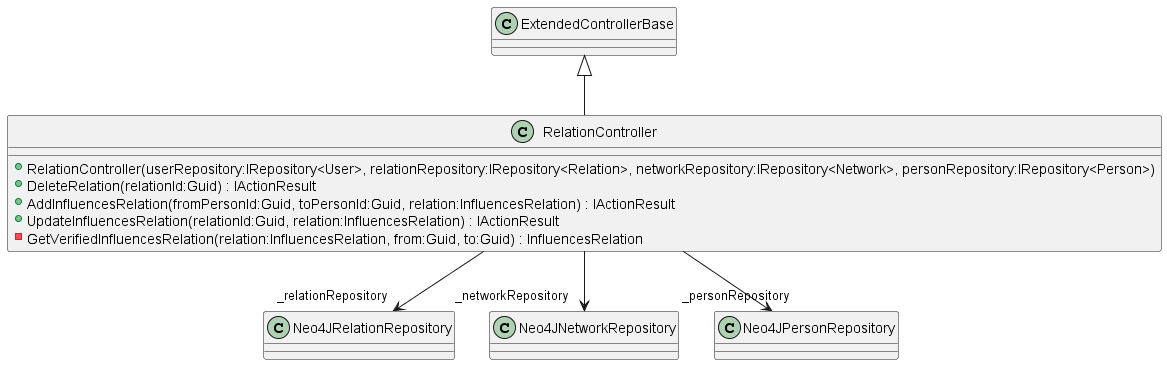


Figure 10 Relation Controller class diagram

#### Group Controller

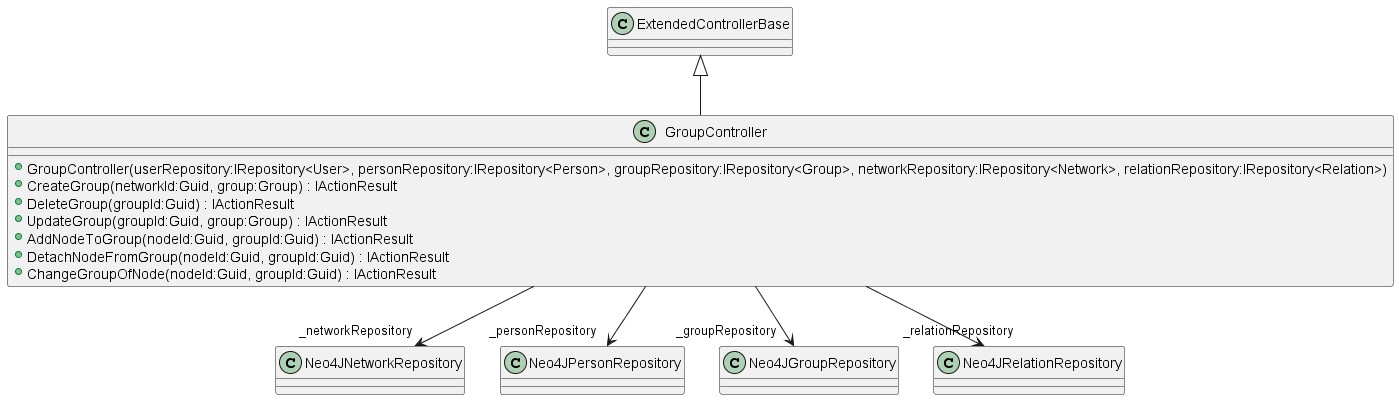


Figure 11Group Controller class diagram

#### Network Structure Controller

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Figure 12 Network Structure Controller class diagram

#### Simulation Controller

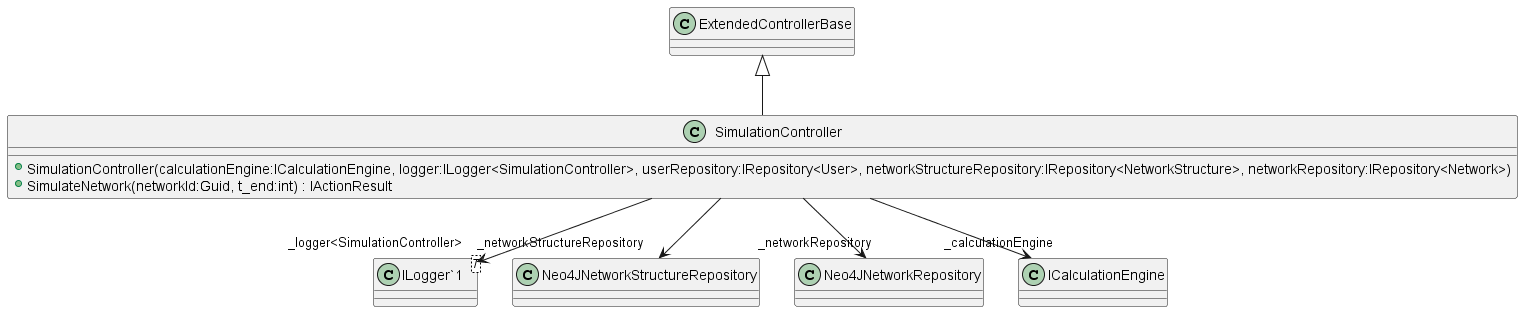


Figure 13 Simulation Controller class diagram

### Authentication

## Neo4J

### DAL Layer Definition

### Repositories

### Neo4J Engine / Database

## Graph Calculation

### Role Calculation

### Value Calculation

## Plugin System

## Configuration

### ASP.Net

### DAL Layer

## Deployment

### Docker

### Manual

# Frontend (Angular)

# Glossary