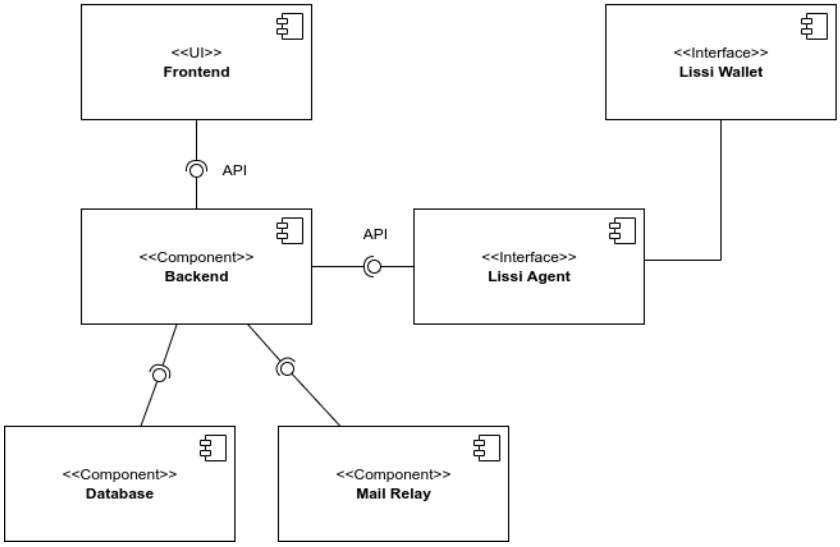
Software Architecture Description

AMOS SS2022

Project 4: Digital Identity

Runtime Components



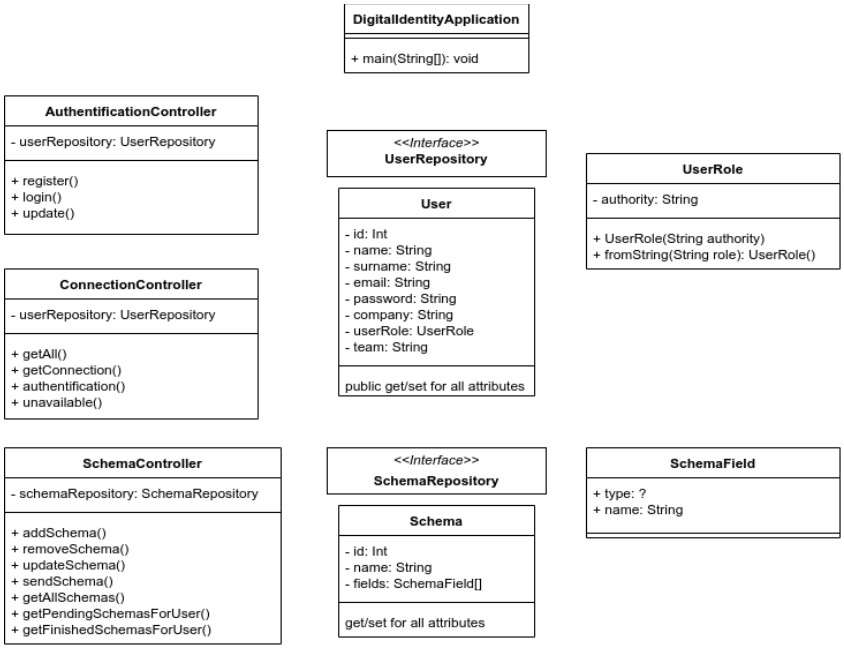
The HR employee manages user identities and certificates via the front end. The frontend communicates with the backend via a REST API. The backend stores the user data and schemas in the connected database and manages them using the lissi agent.

An email can also be sent via the mail relay for certain actions (e.g. user creation/invitation).

A regular employee (user) receives an invitation via email. With this, a QR code can be scanned with the official lissi app. Then a connection between the HR employee and the liss agent is established via the backend. Certificates or surveys can then be exchanged.

Code Components

Backend: Spring

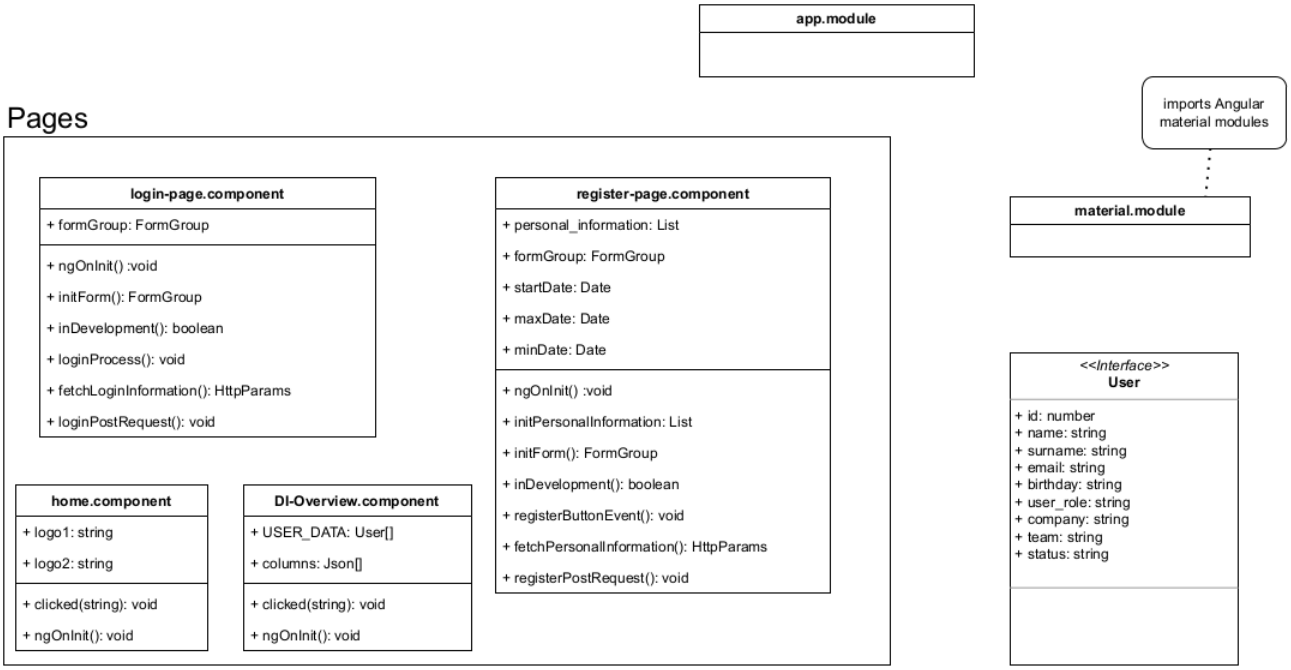


The AuthenticationController handles the login and registration process and also provides an update function for the user data. It also provides the API mapping that spring web implements for us. The AuthenticationController has access to the UserRepository, which is stored in a mysql database in conjunction with Spring Data JPA. The UserRepository is where the user entities are stored. The User entity class stores all private data such as name, email, password, but also the user role. The user role is used for authentication to use various methods.

Each user can optionally be assigned a Lissi connection, via which certificates can be issued. These connections are managed and retrieved in the ConnectionController, which represents an interface to the Lissi Endpoint.

The SchemaController works in the same way as the AuthenticationController. A schema entity stores an enumeration of fields that represent entries of the certificate that can be distributed to a user.

Frontend: Angular



So far, only a simple front-end application is planned. This is also responsible for the corresponding routing. As in every Angular project, it is assembled in the app.module. The material.module is used to bundle the numerous cross-component imports of Angular materials in one place.

In addition, there are individual components that describe individual pages:

- **login-page.component** describes the login page, which gets an email address and a password and should let the user log in.
- **register-page.component** describes the registration page, which accepts a user with the entered data and registers it with the backend.
- **home.component** describes the start page, where the user can navigate to all functions.
- **DI-Overview.component** describes the page that gives an overview of all registered digital identities.

Throughout the model, the user interface is used to store and pass on user data in a bundle.

Technology Overview

Technology Stack

Component	Technology/Tools	Version
Version Control	Github git	-
Frontend	Angular HTML CSS Typescript	13.3.3
Backend	Spring Boot Spring Web Spring Security Spring Data JPA Spring Mysql Connector Maven	2.6.7 2.6.7 2.6.7 2.6.7 3.6.3
Deployment	Docker	20.10.14
Database	MySQL Community Server	8.0.29
API Development	OpenAPI	3.1
Lissi Endpoint	Lissi Institutional Agent	

We chose Angular as our frontend because many team members already have experience with it. Angular can be used to create progressive web apps with the capabilities of a modern web platform. Angular apps are also available through the web browser, but can also be ported to mobile devices very easily. With typescript, html and css as mainstays, proven and well-known technologies are used.

Spring is the world's most popular Java framework and brings many features we need for a backend. It is very flexible and has a comprehensive set of extensions that allow developers to create almost any application imaginable. Spring Security helps us integrate our application with industry standard security schemes and deliver a trusted solution. With Spring Data JPA and Spring MySQL Connector, our database can be integrated very easily and many stumbling blocks can be overcome from the start.

With MySQL Community Server, a strong, widely used and proven relational database management system was chosen.

By using the OpenAPI standard we were able to develop our API very fast and understandable for everyone.