Escape Doom - Software Architecture

Kambal, Nowak, Perov, Selbach, Winter

June 6, 2025

Contents

1 Project-Vision and Goals

The Escape Doom system aims to provide FH Campus Wien with a platform for creating educational escape rooms. In the first semester of our Bachelor's program, we participated in a programming escape room that offered an engaging, interactive learning experience. Inspired by this positive experience, we want to create a platform that allows students to easily access and play similar escape rooms.

1.1 Motivation

Initially, instructors of FH Campus Wien manually created and managed escape rooms by coding each one from scratch. This approach allows for unique, tailored experiences, but it is labor-intensive and difficult to maintain over time, especially as riddles need to be refreshed every one to two years to prevent sharing of solutions between cohorts. This inspired us to develop our own version of an escape room to explore the potential for creating a scalable, reusable platform.

In the current situation, our project—while functional—still requires participants to use an IDE (with the code accessible) which limits usability and makes solutions to code-based challenges more easily discoverable. Our goal is to address these limitations by transforming the project into a streamlined, maintainable system—or in other words, a product—that reinforces the educational value of the escape room.

1.2 Product Vision

Our target groups include both students and instructors at FH Campus Wien. For students, we aim to create an escape room platform that enhances learning by providing a fun, interactive environment with near real-time collaboration and an intuitive, browser-based coding experience. For instructors, we aim to provide a maintainable system that allows them to use escape rooms as an educational tool, with the future goal of enabling them to design and configure their own escape rooms.

Our vision is to develop a tool that enables instructors to create and manage user-friendly and accessible point-and-click escape rooms for their courses, with functionality similar to CodinGame escape rooms.[1]

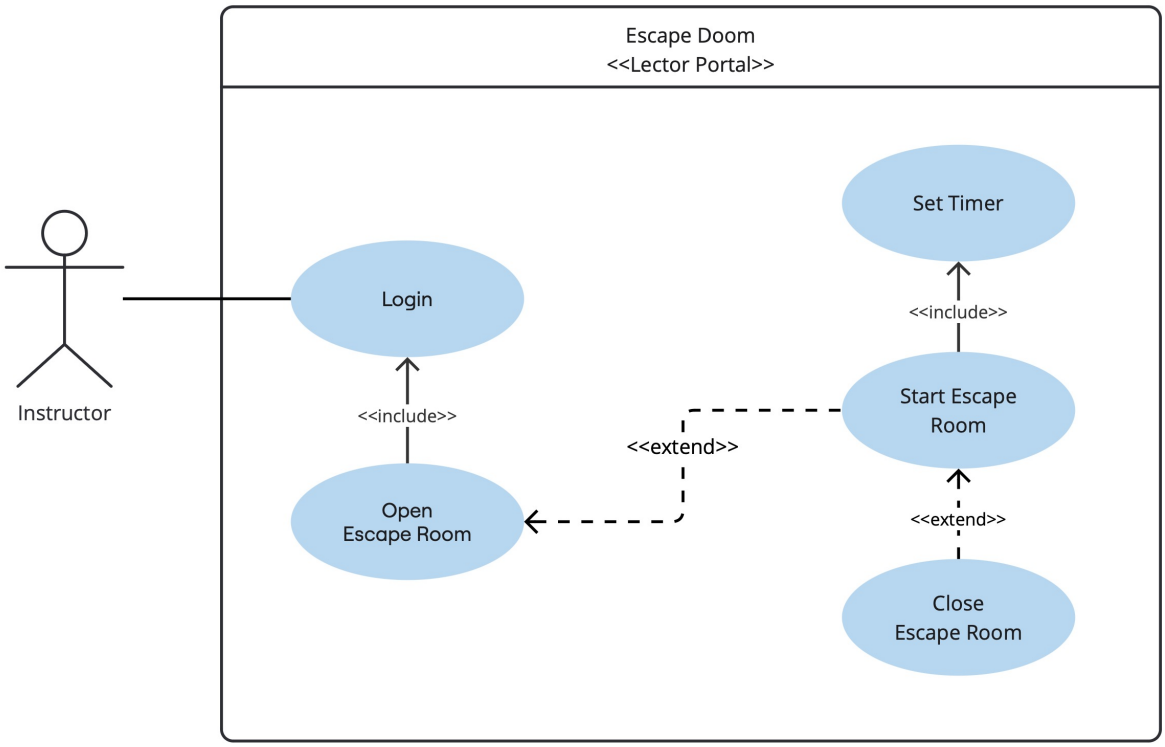


Figure 1: UC01 - Lector Portal

[1] https://escape.codingame.com/

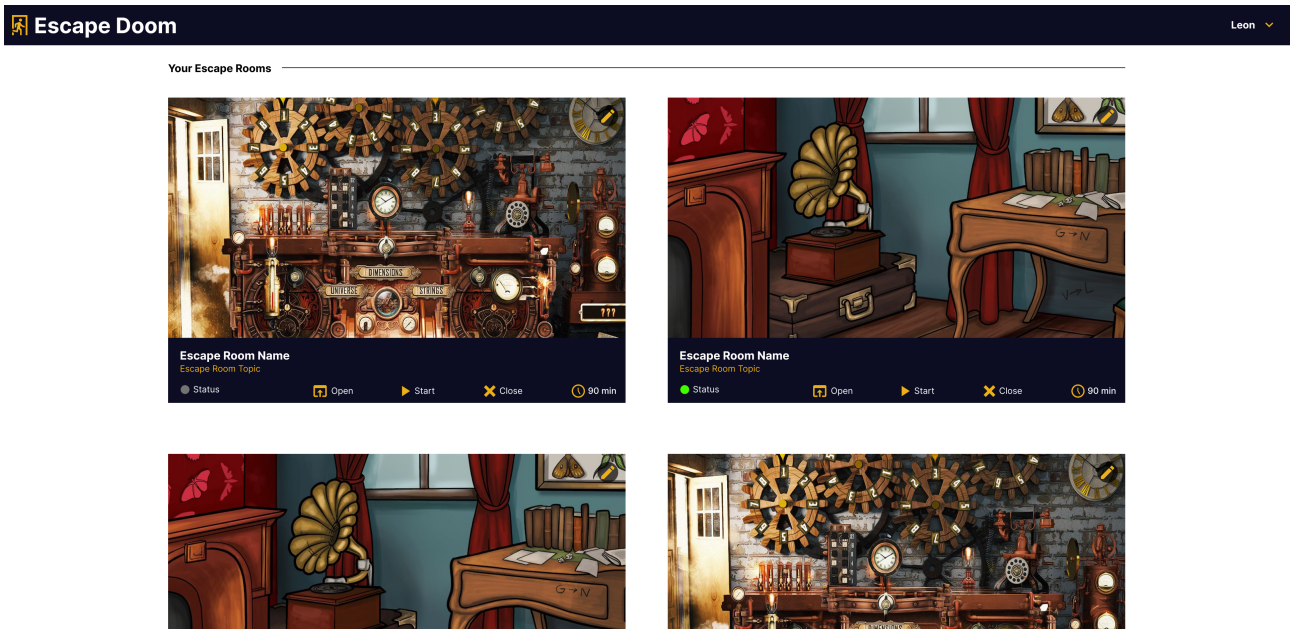


Figure 2: Lector Portal Mockup

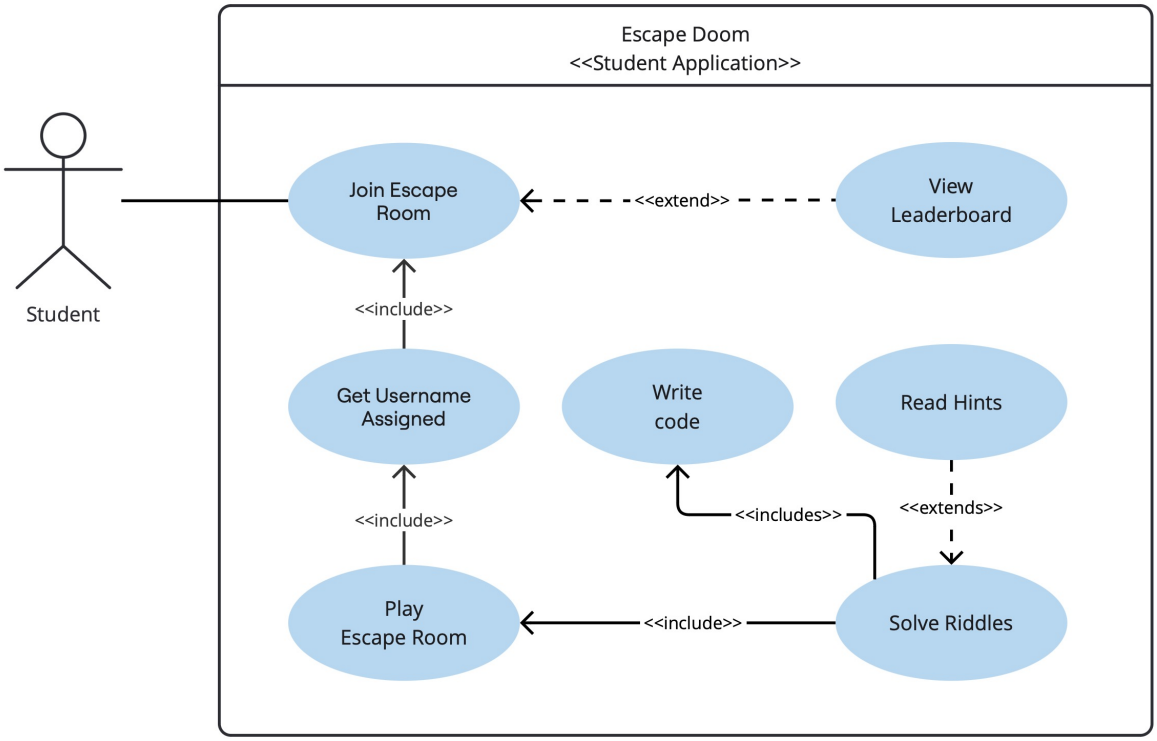


Figure 3: UC02 - Student Application

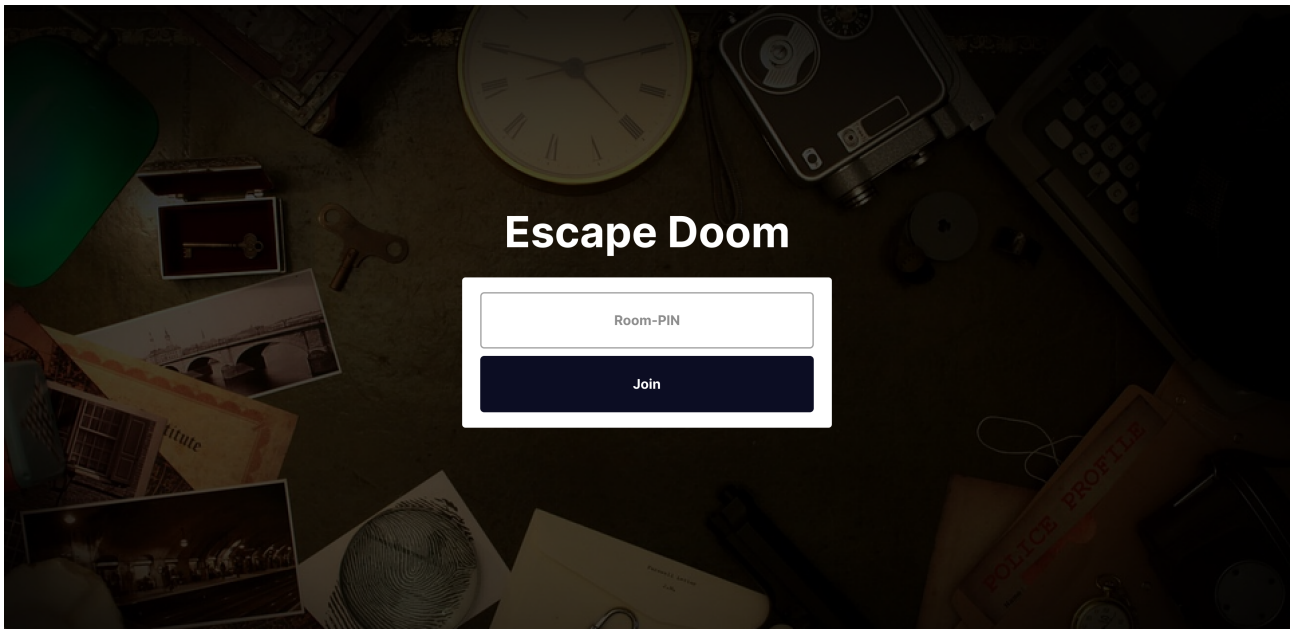


Figure 4: Student Application Mockup

2 Context and Scope

2.1 Business context

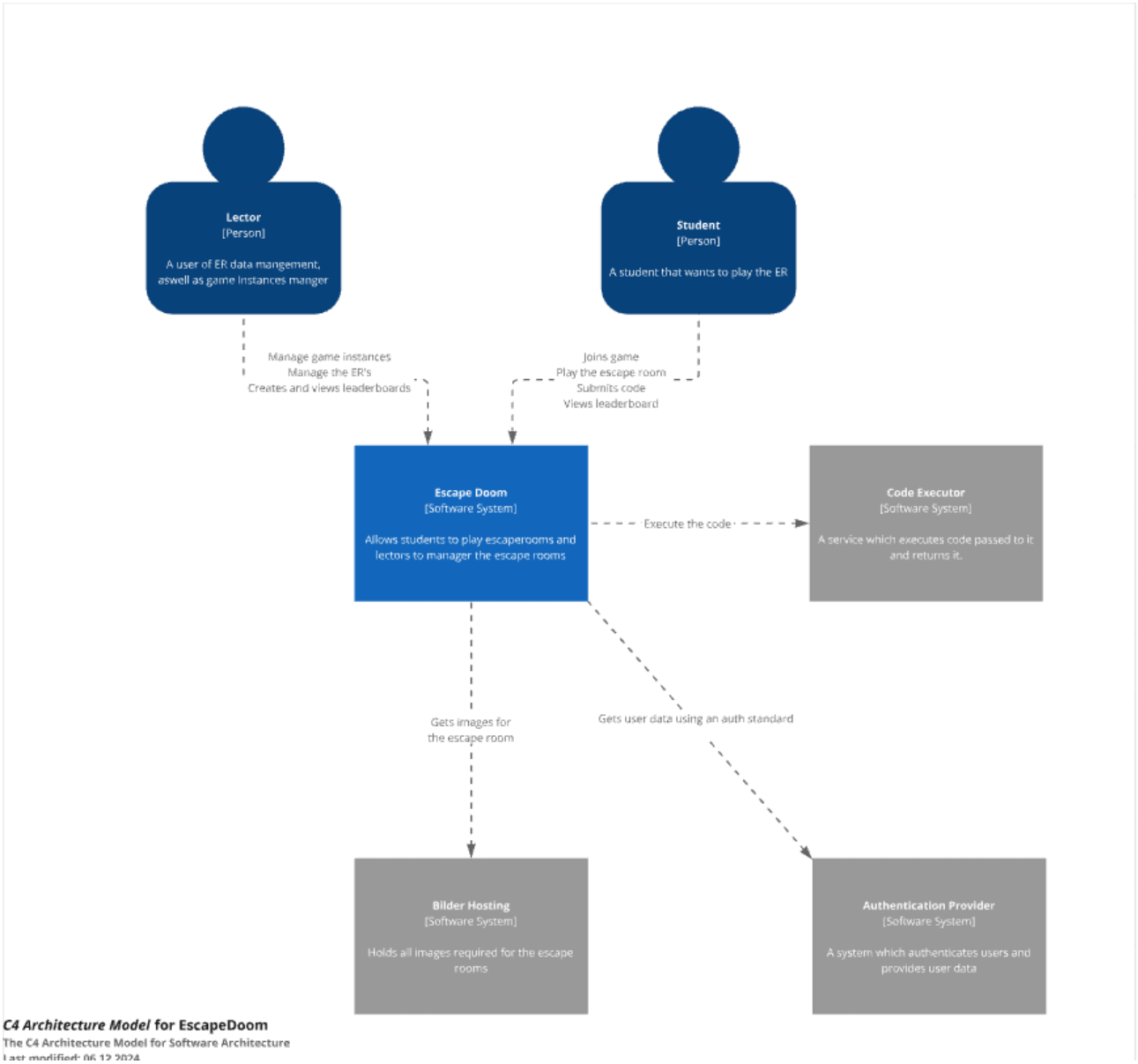


Figure 5: Context diagram with C4 Notation

3 Solution Strategy

To fulfill the task of providing a microservices system which is also maintainable, the team constructed some guiding principles.

1. All backend services from the escape room domain will be written in Spring Boot. To achieve the quality goal of easy maintainability. Because only framework needs to be understood when developing in the system.
2. Since the focus of our project is to build a microservice system the frontend is a single Next.js application, which serves as our only client for students as well as lectors.
3. The system is split into microservices by grouping the business needs in scalable units of compute. This was the choice to enjoy the rapid elasticity and scalability of the architecture pattern.
4. The system itself is being split by business needs, and therefore runtime dependency is a side effect the team is accepting for faster iterations and architecture overhead.
5. The cloud technology used to enable microservices development with the largest velocity possible is the spring cloud ecosystem with a strong emphasis on the Spring cloud Gateway. For efficent and easy custmizable service routing.
6. The deployment is on a Kubernetes cluster and, therefore all systems are built stateless and the state is externalized in systems like redis and Postgres-sql.
7. The Code Executor is a system developed by us but seen as an external system since it is an already solved sub-domain and also could easily be swapped with a finished project. Instead of the own implementation. To support this flexibility, we have also implemented an interface that allows integration with external APIs capable of compiling and executing code.

4 Building Block View

4.1 Level 1 - Context

EscapeDoom is developed with the intent to not create/maintain all parts it requires. Systems such as image hosting or the authentication provider are not part of the EscapeDoom software system, as can be seen in figure ?? below.

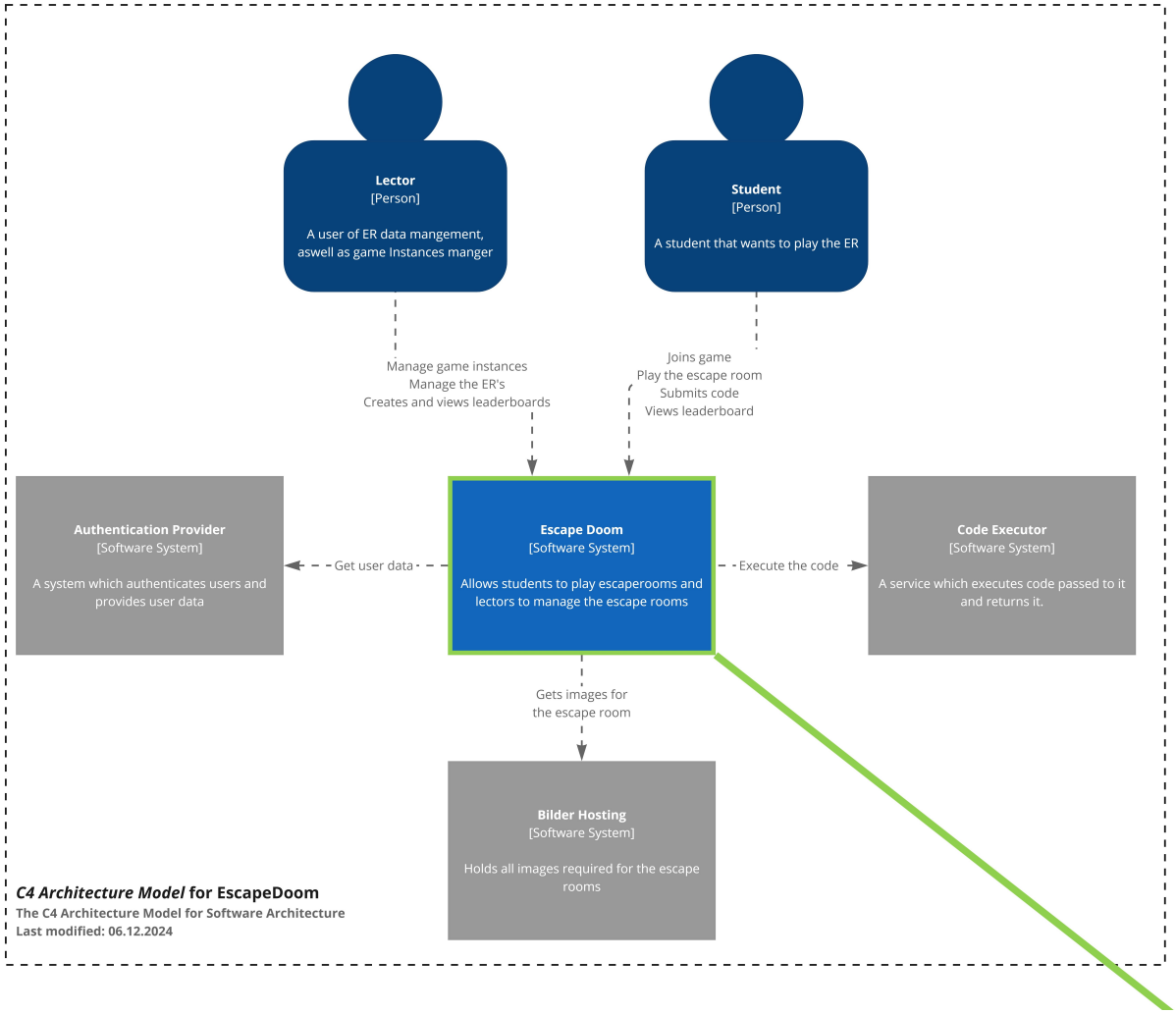


Figure 6: Level 1 - Escape Doom

Authentication Provider	Authenticates users to manage access
Escpae Doom	The main system, with which the lectors and students interact with
Code Executor	Responsible for executing code submitted by students
Bilder Hosting	Host images that are displayed in the escape rooms

Table 1: Explanation of the individual blocks displayed in the Level 1 Context

4.2 Level 2 - Containers

The EscapeDoom application itself can be segmented into multiple subsystems as depicted in **??**. Each dotted line indicates communication, and thus a technical dependency. As can be seen, the front-end is maintained as one application while the back-end is split into microservices. The are then grouped into three separate domains as can be seen by the dotted lines.

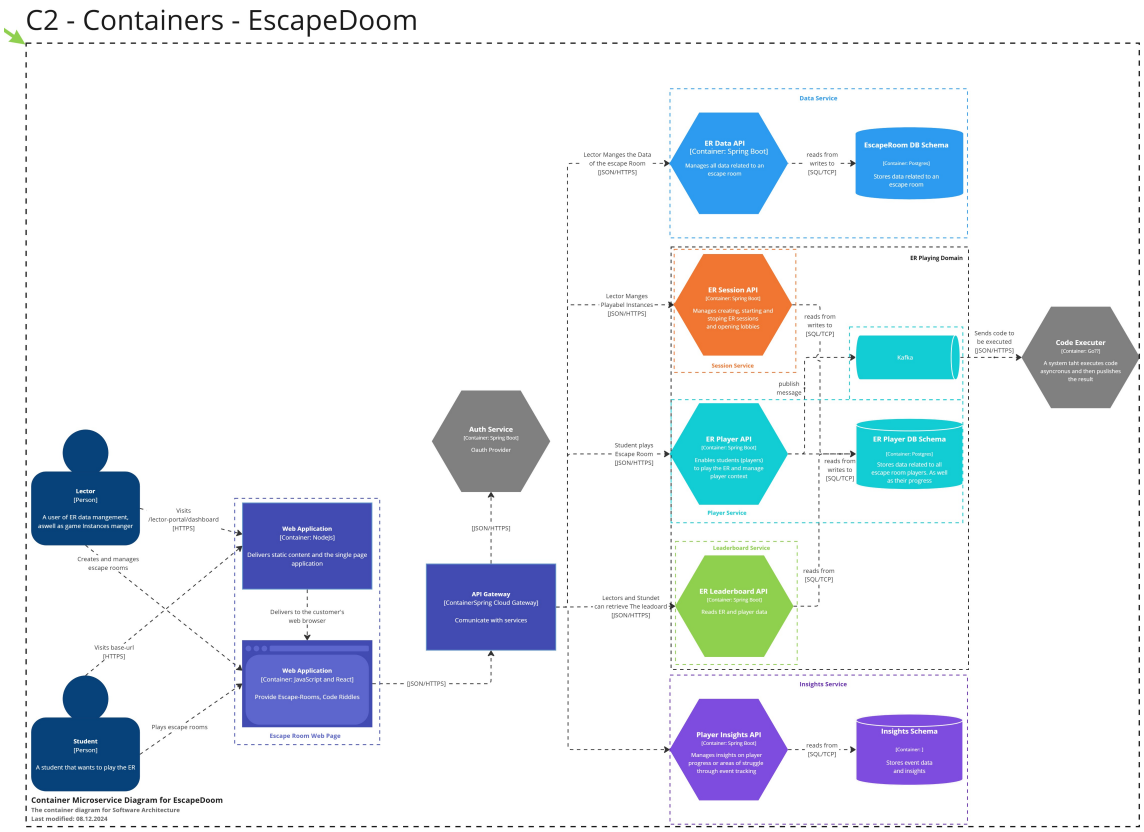
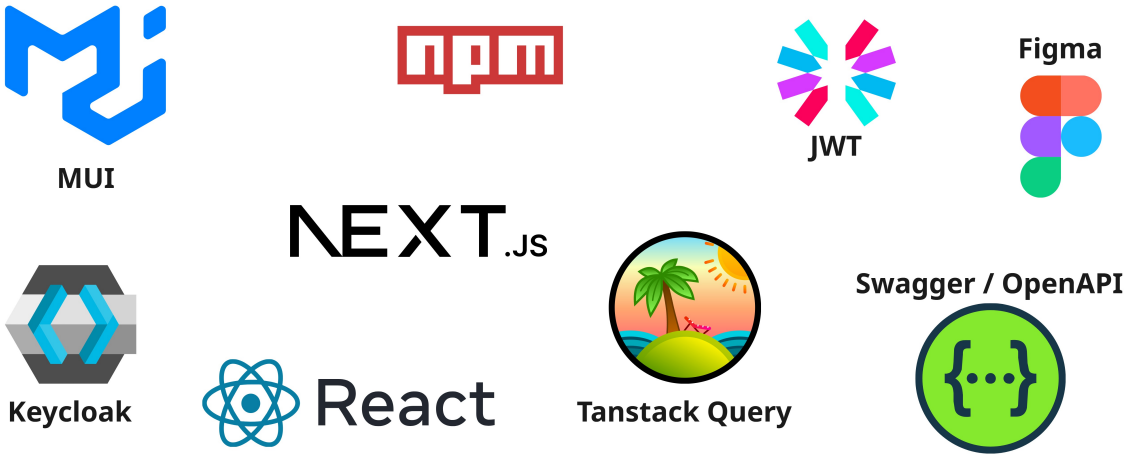


Figure 7: Level 2 - Containers

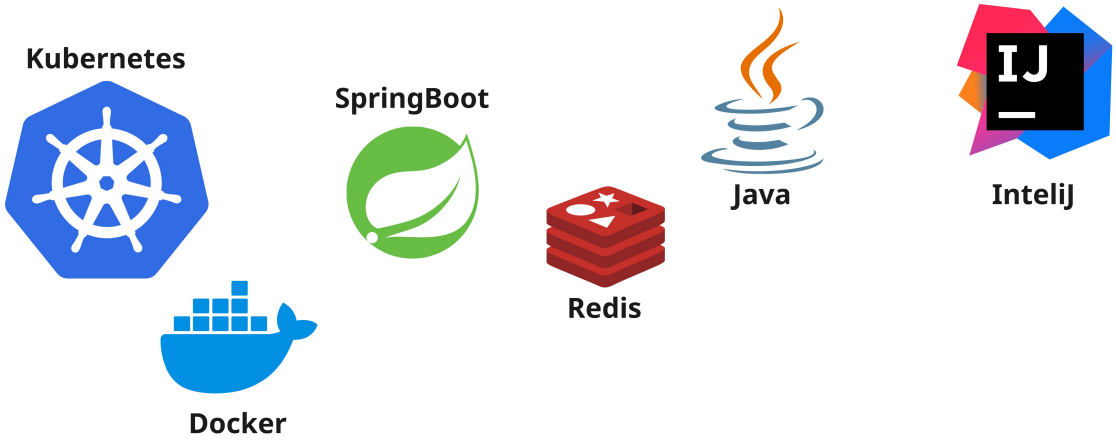
5 Project Tech-Stack

This project was split into a frontend, based on React and NextJS and a backend based on Java and SpringBoot. Below are two images listing most relevant technologies / tools per Stack:

5.1 Frontend



5.2 Backend



5.3 Milestones and current state

Since this project is a continuation of a bachelor project the first semester of the master, the goals this year was to split the application into microservices, add features and make it a deployable app.

The milestones thus can be split into 5 steps: