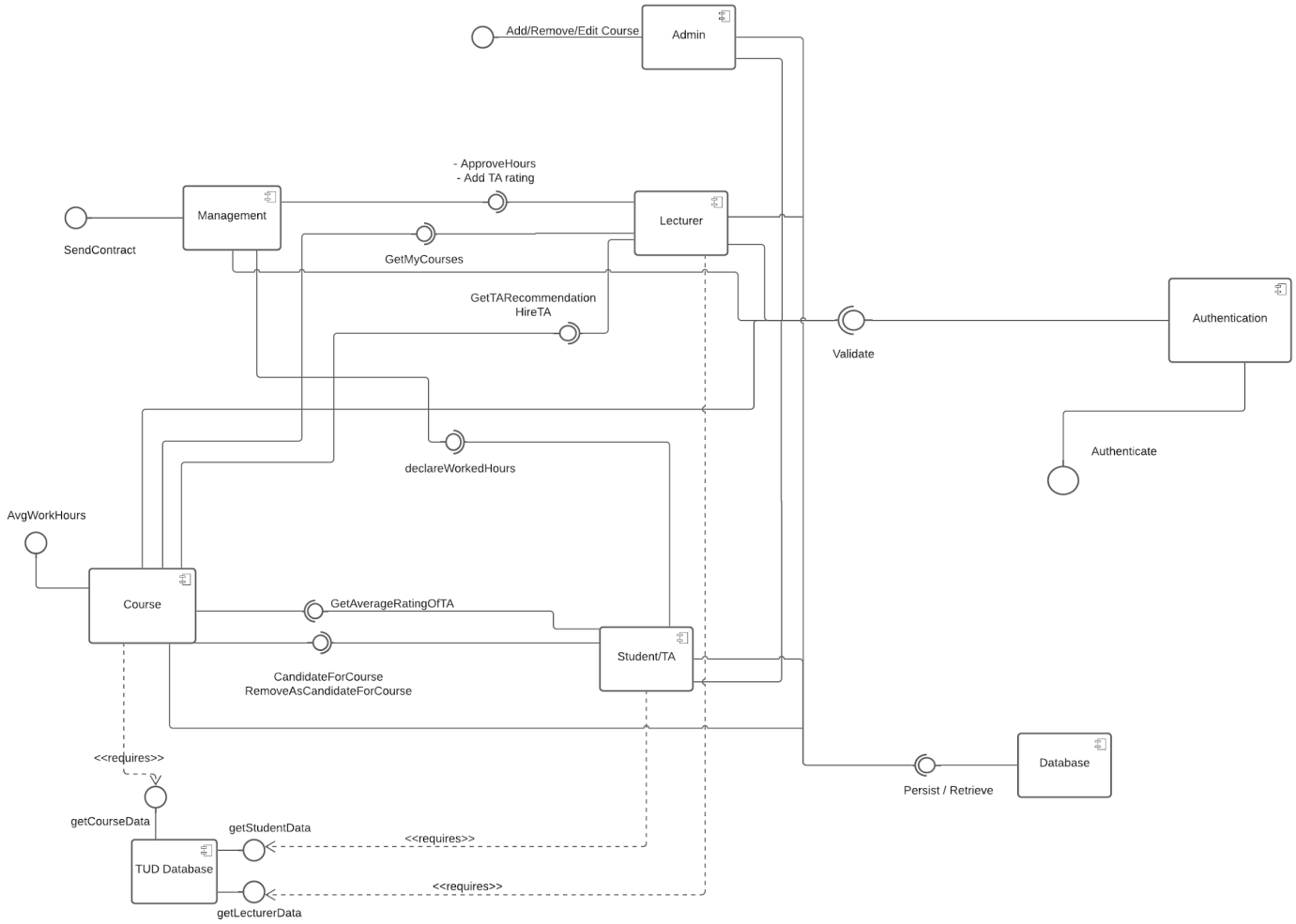
Software Architecture - group 15a

Project architecture



Introduction

Domain driven design (DDD) is a process and domain centered design type, which is compatible with microservices due to the detached architecture of a domain from other domains. Microservices improve scalability, ease testing and help with fault tolerance, which is why this type of an architecture is chosen for this project. The application is a TA Hiring System that has several components that communicate and

exchange information, which consists of Authentication, Course, Lecturer, Student and Management components.

There are several design choices made, while producing these components. For instance, Lecturer and Student have been separated and not used as a single User domain, because the attributes that these domains have and their functionality have significant differences. The Management domain is taking care of hours worked, rating of TAs and sending the contract to the students. The Course domain is responsible for course-related data and also for recommending candidate TAs. Finally, the Authentication domain authenticates credible users and allows them in the system.

Bounded contexts

Users are a core bounded context of the system. It is one of the most important contexts since the system revolves around various types of users interacting with courses through different roles. Users will be subdivided into a Student, Lecturer, and Admin subtypes where each role interacts with User downstream dependencies Courses and Management differently with different privileges. Although each subrole has similarities such as name and netid, they also have differences as outlined above. As such we decided to map each subrole to a separate microservice in order to have higher cohesion.

Course is a bounded context, and it is one of the core domains. The system depends on a course and has no purpose without courses. Course has dependencies with other domains such as lecturer (who leads the course) and students (enrolled in the course). The size of the course depends on students, while the lecturer set depends on lecturers. Course is a separate domain as it provides the functionality of keeping data of courses.

Management is a separate bounded context that stores the related information for a TA for a course. For example, hours worked/declared/approved and the rating of the TA. It is also responsible for sending the contract to the chosen student to be a TA.

The authentication bounded context is a core domain that also corresponds to an independent microservice. It is unique from the others, in that it is the only one that has no dependencies on other bounded contexts, but is entirely responsible for serving other microservices. For this reason, compared to the other ones in the system, it is the most similar one to emulating a client-server design pattern.

Microservices

Course

As a microservice, the course is independent and has its own methods.

The responsibility of the service is

- To connect (candidate) TAs to one course
- Keep data of all courses
- Keep track of TAs for a course
- Keep capacity of a course
- Keep the number of needed TAs for a course
- To request average worked hours during a course
- Create, edit, remove courses (Admin only)

Course microservice dependencies:

- Endpoint for creating Courses (Admin only)

Each course has a unique id. All communication with other services will be done through the controllers.

Management

Management microservice responsibilities:

- Create a TA contract based on information provided
- Track declared, worked and approved hours of TAs
- Track the rating of a TA for a course
- Allow lecturers to approve/disapprove declared hours by TAs
- Send contracts to candidate TAs
- Send emails to candidate TAs who have been picked for the job

Management microservice dependencies:

- Lecturer creates and approves hours of TAs by interacting with the contract
- TAs declare their working hours

Authentication

The main tasks that the authentication microservice is responsible for are:

- Generate secure JWT tokens to different microservices whose instances wish to access other parts of the system and their databases

- Create a “server” that exists in the background and stores all information related to authentication and security
- Validate all requests to the server by determining the validity of the JWT tokens passed along with the requests
- Maintain different access permissions based on the role that each user has
- Provide endpoint for first time authentication (generate JWT)
- Provide endpoint for validating requests

Student

The responsibilities of the student microservice are to:

- Keep track of passed courses student is eligible to become a TA for and corresponding grade achieved
- Keep track of courses student has candidate themselves as TAs
- Keep track of courses student are/have TAed through Management objects
- Provide endpoint for student average TA rating

Dependencies of student microservice are:

- Endpoint to declare worked hours
- Candidate student to a course
- Remove student candidate from course
- Get averaged workload for given course

Lecturer

The Lecturer microservice will have the following responsibilities:

- Keep track of the courses per every lecturer. We have decided to have the constraint that every course can have only one responsible lecturer
- Get a list of recommended students to become Teaching Assistants based on their grades and rating
- Pick a Teaching Assistant for their course
- Approve working hours for current Teaching Assistants
- Rate the current Teaching Assistants

The Lecturer microservice dependencies:

- Create a Management object for a chosen TA with a fixed working hours for a given course