ASSIGNMENT 1 DRAFT

Software Engineering Methods

Group 11b

Aleksandar Andonov

Bart Coster

Kasper van Duijne

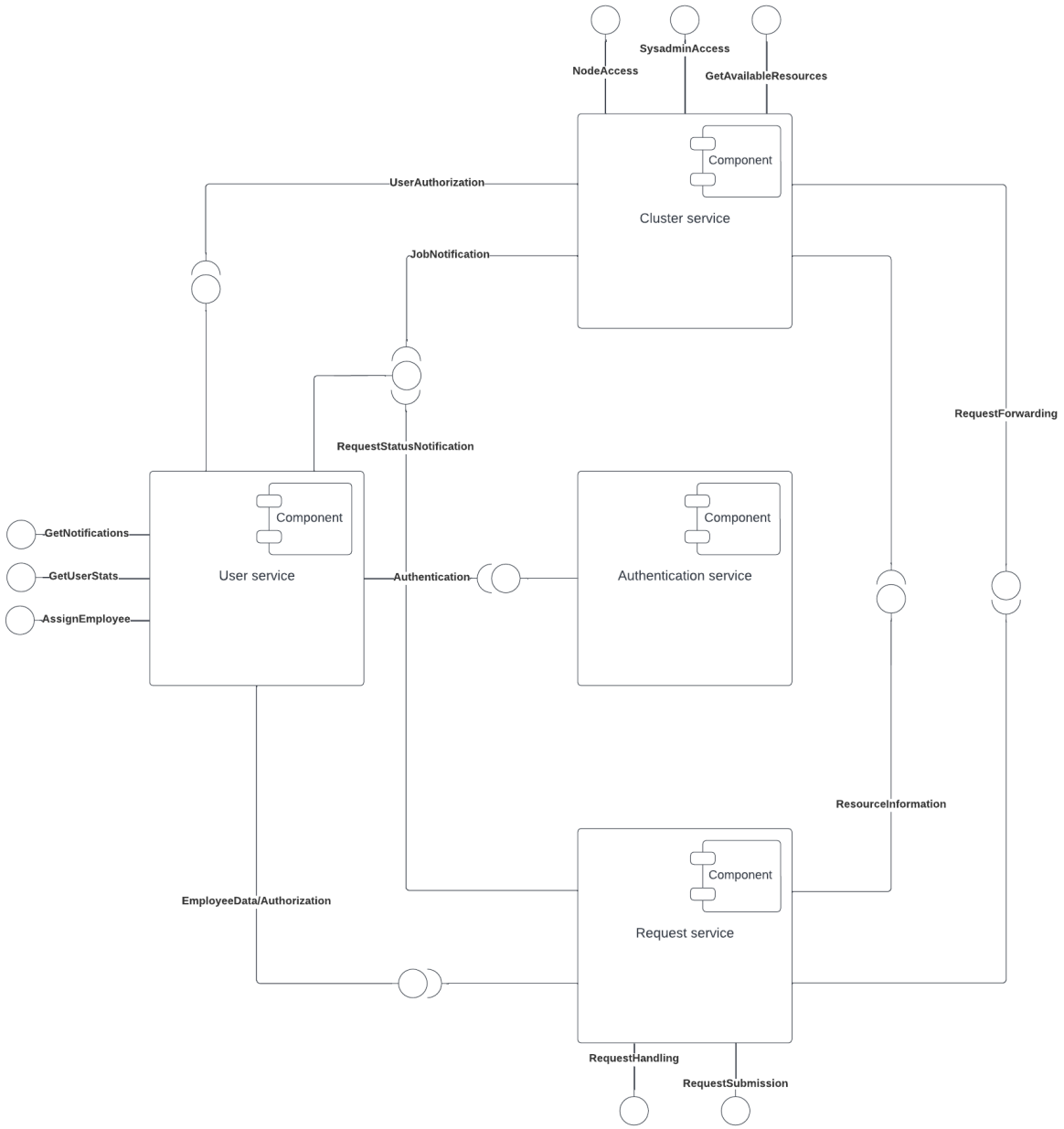
Ariel Potolski Eilat

Alan Kuźnicki

Vladimir Pavlov

1. Contents

2. Overview of the architecture



3. Bounded contexts

3.1. User context

We identified the user as a separate context. The user bounded context is all about the communication with the user and the starting point of interactions. Therefore it handles incoming notifications directed at the user from another context, keeps track of users assigned faculties, having a gateway towards request status and authenticating itself via a separate microservice. Moreover, the user context is a core domain of the system.

3.2. Faculty context

The faculty logic should be classified as a separate bounded context, because it's tangible enough on its own to not be contained within the user and cluster microservice. Since the requests themselves become independent entities in the system, once they are sent by the user, a service that handles them would be required. Furthermore, the faculty is a core domain of the system.

This context is also separable from the supercomputer's scope of functionality, as the faculty is responsible for keeping track of the resources assigned to them, and does not have complete information about the whole capacity, while the supercomputer works with the already created nodes and the management of the overall cluster of nodes with all of the available capacity of the system. The scope of the request handling service is limited to the respective faculties and request access to information about the resources of Delft Blue.

3.3. Cluster context

When dividing the bounded contexts, we identified the cluster as being one of them. We understand cluster to be the supercomputer itself, and therefore it handles the supercomputer functionalities, such as receiving nodes from users, managing and updating the amount of resources available, and managing the schedule of the jobs, as well as granting access to the sysadmin. Moreover, the cluster should output information to the other bounded contexts, such as the amount of resources available, or a job notification. Furthermore, we decided to use partnership as the organizational pattern to communicate with the other bounded contexts, since cluster influences and is influenced by them. The cluster is a core domain of the system, given that all functionalities need to, somehow, go through or interact with it.

3.4. Authentication

We decided that while authentication itself is a much smaller and more cohesive bounded context than the others, it would nevertheless not fit in together with any of the thus far defined areas. We have therefore opted for leaving it as a separate context, focusing on just the authentication of the users of the system. Different from the others, authentication is a generic domain of our system.

4. Mapping to microservices

4.1. User microservice

The bounded context is mapped one-to-one to the microservice. This makes the user a micro service that is responsible for assigning the faculties to the users and keeping track of which user has which faculties. Furthermore it saves all the notifications that the other micro services send. It also makes sure that if a message is sent to another micro service it is accompanied with the proper authorization. It got this authorization by authenticating with the authentication micro service.

4.2. Request microservice

We have decided that the faculty bounding context is dominated by one key group of functionalities - the requests. This is why we chose to represent this context through the request microservice. It is still, nevertheless, a one-to-one mapping from the faculty bounding context. The request service implements the functionality which facilitates the communication between the user and the cluster/supercomputer. This microservice is responsible for taking care of the correct division of resources within the respective factulties and acts as the intermediate interface through which the faculty distributes the resources to the respective users.

The microservice receives data from the authenticated users and information about the available resources that the cluster will allocate to the faculty. Internally, it handles the requests - scheduling and allocating its resources, and then submits them to the cluster.

4.3. Cluster microservice

In order to implement this bounded context, we decided to stick to a one-to-one mapping, and we will, therefore, implement only one microservice that will handle all the functionalities of it. As said before, this microservice will be able to handle requests for posting a node, which adds resources to the cluster, as well as remove a node. Besides that, this microservice will handle the creation and management of a schedule for all jobs that are pending or already approved. Additionally, it should grant special access to the cluster by the sysadmin. Finally, the cluster microservice will handle/support operations related to the resources available.

4.4 Authentication microservice

The authentication microservice implements the only functionality inherent to its bounding context - the authentication itself. Its purpose will be to authenticate users who begin interacting with the system and to provide tokens that the other microservices can send to each other and verify.

5. Responsibility and role in the infrastructure

5.1. User microservice

The user service will store employee faculty relations in a database and is responsible for checking if a user is allowed to make a request to a given faculty. The service will be in direct communication with the authentication service. Furthermore the user service will be responsible for delivering a user's stats when the given user requests them, and deliver notifications and updates to a user when asked. The service will get this data by communicating with the cluster service. Lastly the faculty service will talk to the user service to check authorization of an employee.

5.2. Request microservice

1. Request Handling/Submission - the internal, essential logic of the microservice which processes all faculty requests. During the process, requests are being scheduled, filtered and if approved forwarded to the cluster.
2. Resource information - this is the functionality responsible for querying the cluster for resource allocation, performance issues and the current states of the nodes. Shortly, all information that might influence the reserved resources of the faculty.
3. Request status notification - logic responsible for notifying the user with the current state of his/her job. It should also alert the user if there's a problem with the submitted job.

5.3. Cluster microservice

As explained above, we chose to separate a supercomputer microservice (which we named "cluster service") and let it handle all resource- and job-related tasks. Resources are the CPU, GPU and memory resources that a user can request to use; a "job" is the content of the request, so a task for the cluster to complete. The possible interactions between the service, the user, and the rest of the system are:

1. The node/resource and job data is stored in the service. It also handles the scheduling of incoming jobs.
2. Whenever a job is completed or delayed, the cluster alerts the user microservice, which then stores the notification for the job's requestor to access when they use the system.
3. Whenever a user accesses this microservice or another service forwards a user's request, the cluster reaches out to the user service which is connected to authentication.
4. A user can add and remove nodes from the cluster by directly accessing the cluster service.
5. A sysadmin can access the cluster database through the microservice to acquire the information specified in the requirements.

6. Any user can see the resources available for the following day in this microservice.
7. A request made by a user is handled by the request service and, if approved, is forwarded to the cluster where it is scheduled and the job is completed.
8. As the request microservice has to know whether the user's faculty still has free resources, it asks the cluster service for this information.

5.4. Authentication microservice

The authentication microservice is linked directly to the user service. It is through this connection that users are authenticated - whenever they make an interaction with the system, the authentication microservice is contacted to either provide authorization or verify a token sent with a user request.

6. Rationale and motivation

6.1. User service

The user service is a well needed standalone microservice. This is because employee faculty relation data is now stored in a separate microservice, which in our opinion is useful because functionalities are kept separate. Also the user service is responsible for delivering notifications to the user. We also think this is useful to have 1 microservice be able to deliver all the notifications to the user. When expanding the total project, a new microservice could send new notifications to the user microservice, which can deliver it to the user with a given infrastructure, which means the project is more easily expandable.

6.2. Request service

We have chosen to separate the request service from the rest of the infrastructure, as communication would become extremely cumbersome and convoluted if we lacked this middleman structure, which performs additional scheduling and filtering. Moreover, it would be functionally impossible to integrate this microservice into any other, as it would introduce either a lot of overlap or skew the balance greatly towards one microservice, defeating the purpose of this style of architecture.

6.3. Cluster service

Given that the supercomputer has an important role on the system, and it is responsible for a big share of the system's functionalities, we decided that it should be a bounded context by itself. This is supported by the fact that, although it shares some functionalities with the other bounded contexts, the cluster carries concepts that are unique to it.

Finally, we believe that creating a microservice to handle the supercomputer's functionality will ease the parallelization during the development of our project, as well as provide a better structure and organization for our implementation.

6.4. Authentication service

The authentication is the lynchpin holding a system as critical as a supercomputer together and thus we believe that it would be unwise to assign this functionality to another microservice. Furthermore, authentication being a separate microservice will mean that if, for any reason, it has to be turned off, no other functionality will suffer.