Technical design document

Author: Denny Cox
Class: S6-RB10
Semester: 6

Table of contents

Technical choices

Microservices
The Kwetter application is enterprise software and will be built using a microservices architecture. With this, the functionality of the application must be separated into different services. In the analysis document the project has been worked out with event storming. The different aggregations that came out of this can be used as microservices.

API gateway
Since the application is broken up into smaller services there has to be a way to communicate from the frontend. For this an API gateway has been added. The advantage is that the endpoint of the different APIs in the system will not be exposed and protected from different kinds of attacks. Because every microservice will communicate via a single point of entry with the API gateway, things like authorization using API tokens will be easier to implement. This decreases complexity.

Disadvantages are that the single point of entry could also act as a single point of failure. The routing to the different microservices must also be managed during deployment to ensure the routing works correctly.

A technology has to be chosen to create an API gateway. Following Microsoft Docs example projects, a good and popular choice is Ocelot; an Open Source .NET Core based API Gateway especially made for microservices architecture that need unified points of entry into their system. It is lightweight, fast, scalable, cross-platform and provides routing and authentication among many other features.

Event bus
To keep the data in all the services up to date if data changes, the system uses an event bus. This architecture is called an event-driven microservice architecture.

A technology has to be chosen to create an event bus. Following Microsoft Docs example projects, a good and popular choice is RabbitMQ; an open source message broker that is easy to deploy for distributed systems.

Figure [1] shows the global architecture of the project. As shown, the different components will be able to run in separate Docker containers.
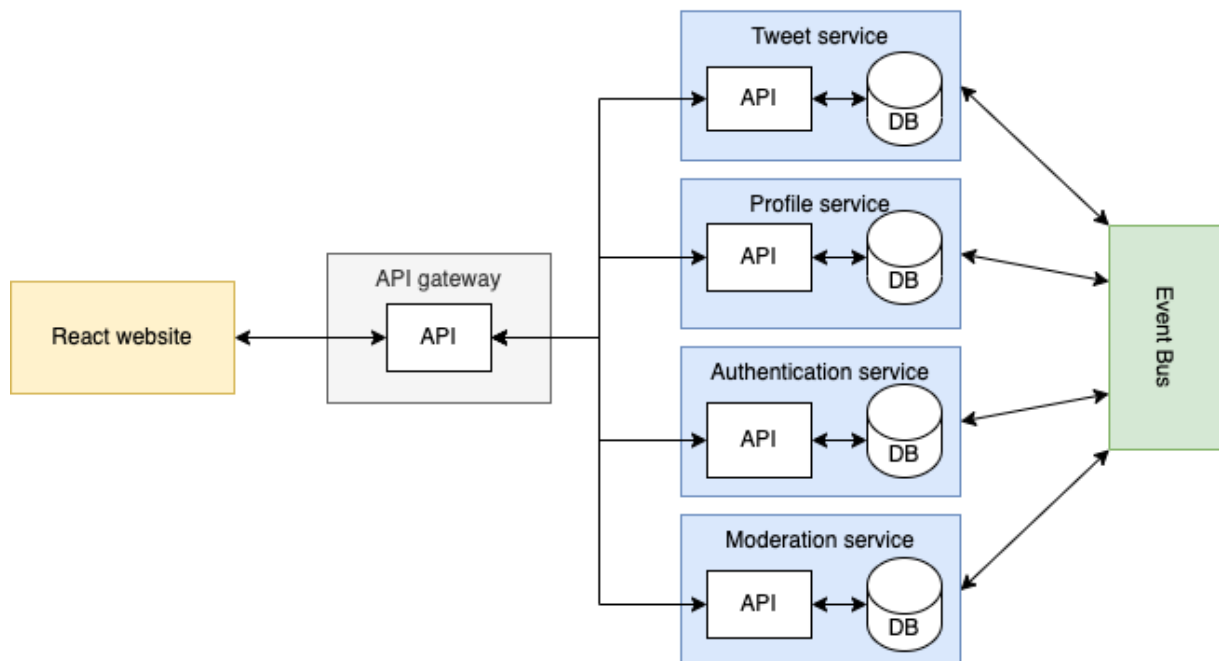
Figure 1 Global architecture

Security by design

Security has to be implemented right from the start. To make sure the security is implemented well only proven and well documented technologies will be used. Because the use of microservices, a microservice can be down without the other services being affected. With the development, the testing and the building of the software will also be automated. The OWASP security design principles will also be followed to make sure the most well-known vulnerability will be covered. A security risk- and impact analysis will also be performed to map out the possible security risks.

C4 models

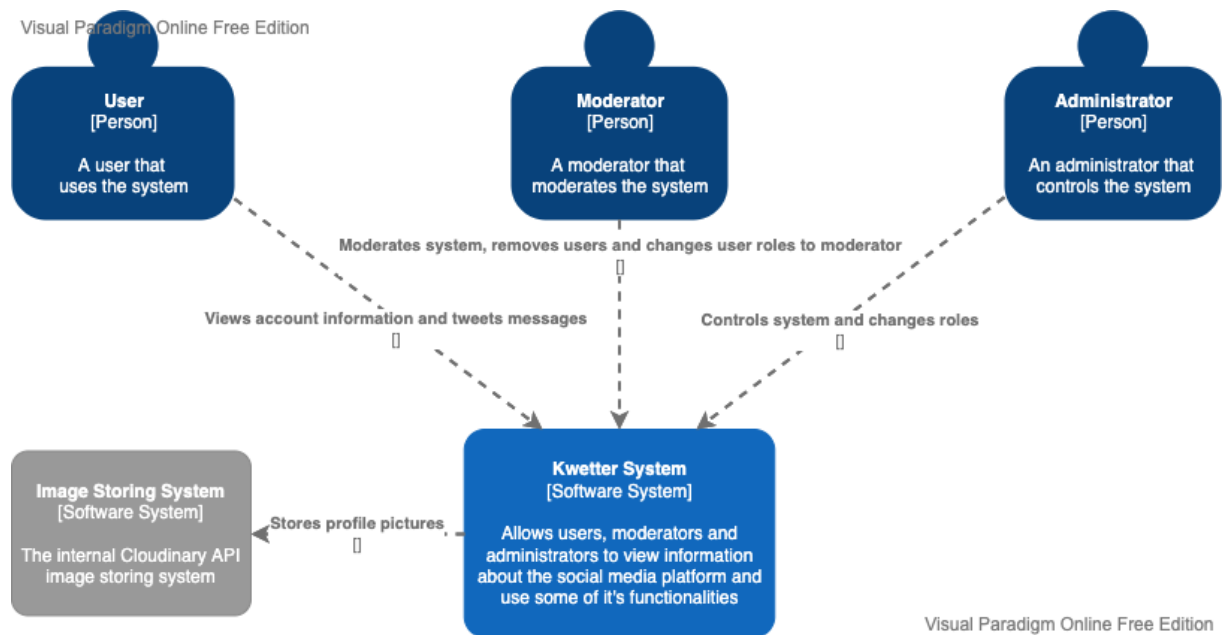## Context diagram

Figure [2] shows the context diagram.

Figure 2 Context diagram

Container diagram

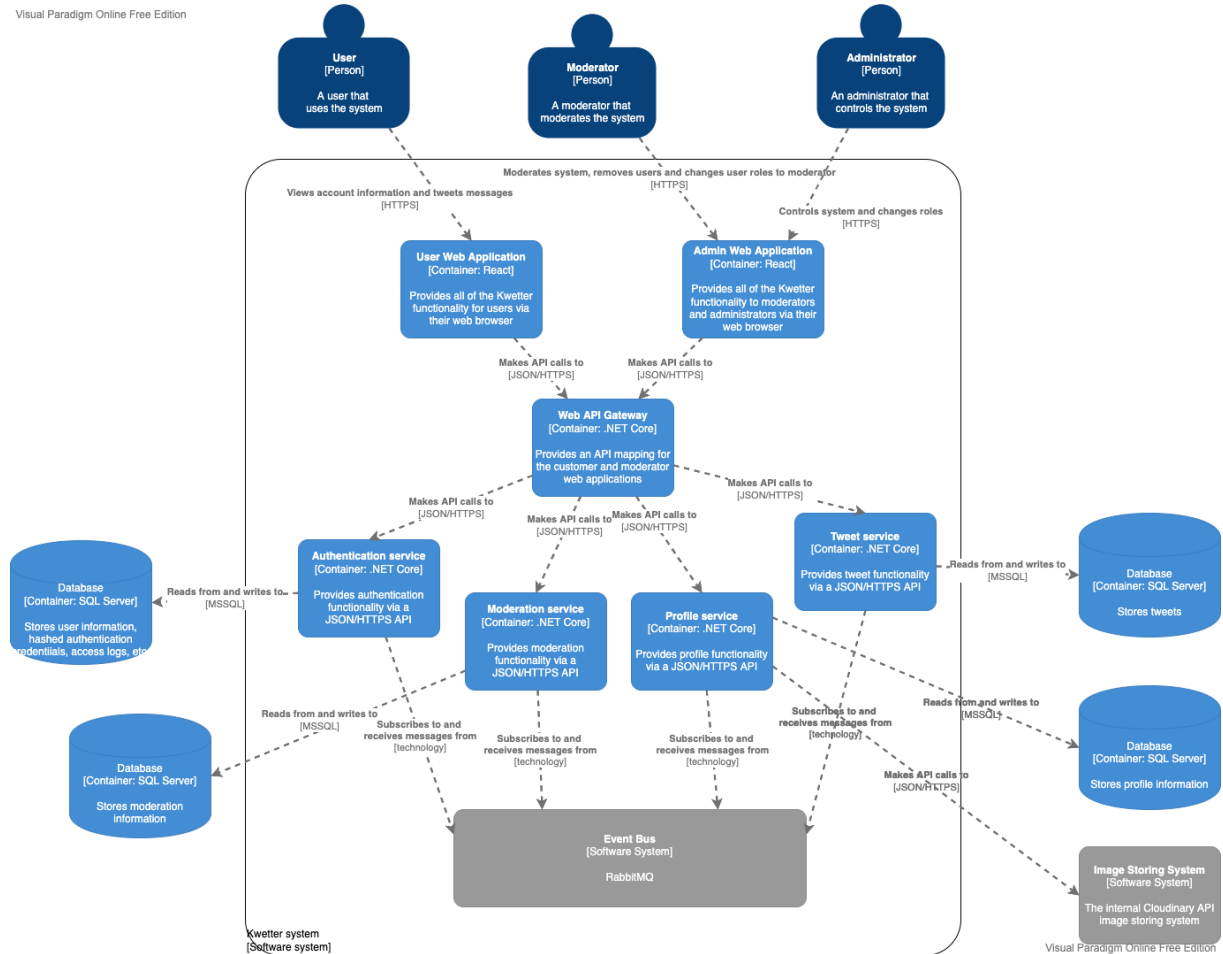Figure [3] shows the container diagram.

Figure 3 Container diagram

Component diagram

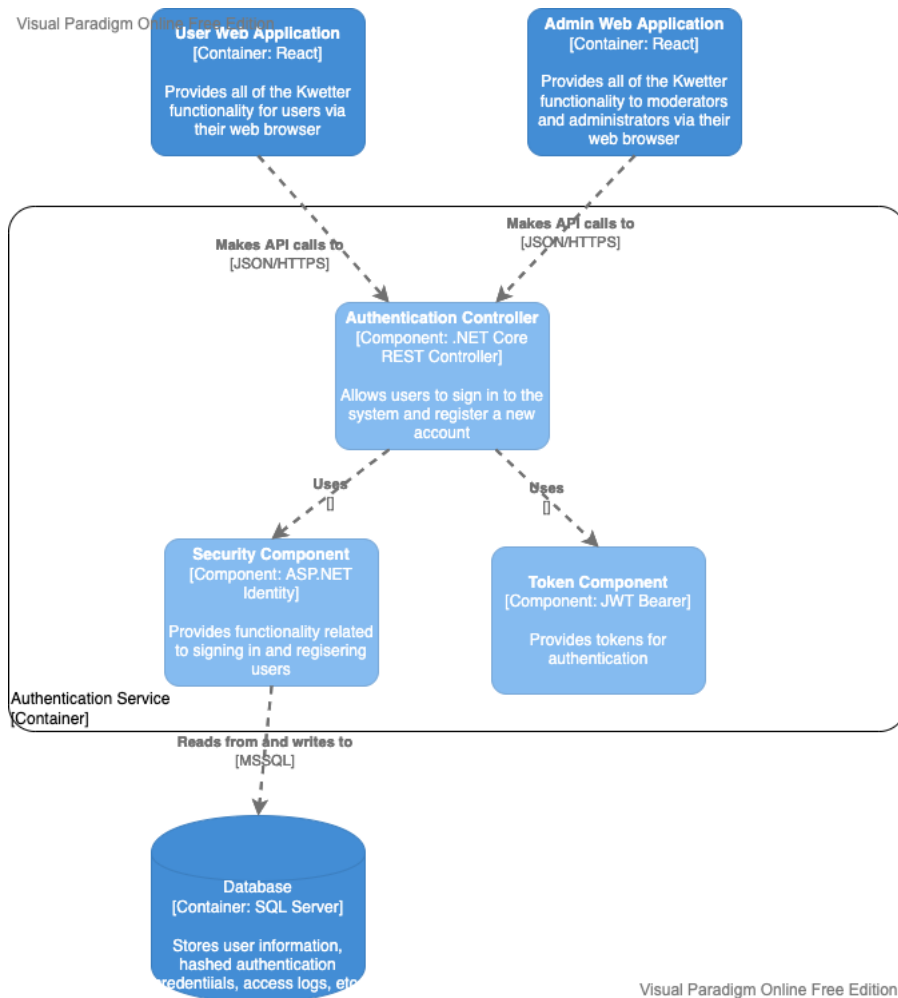Figure [4] shows the component diagram of the authentication service.

Figure 4 Component diagram

Deployment diagram

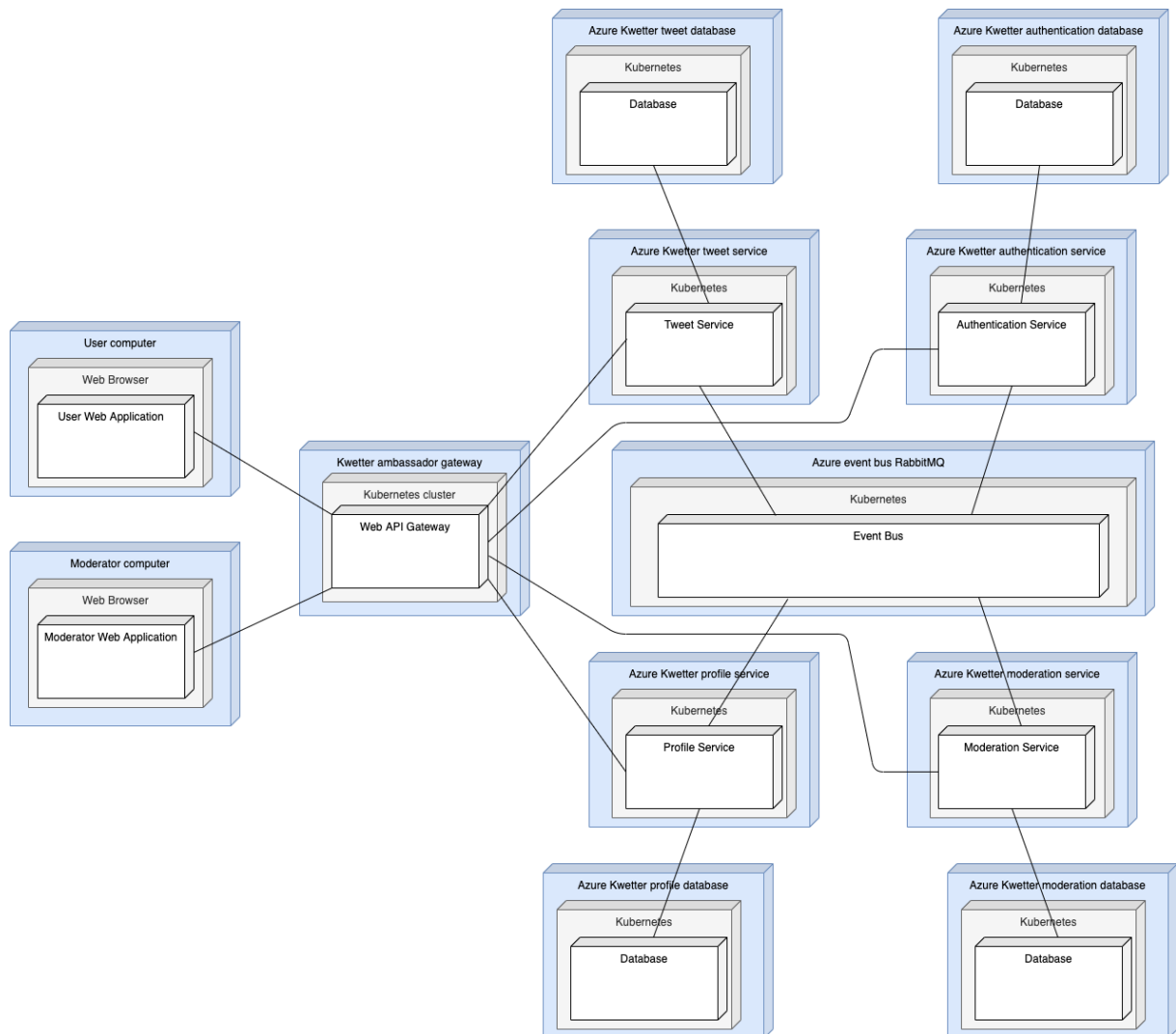Figure [5] shows the deployment diagram of the Kwetter system.

Figure 5 Deployment diagram

Class diagram

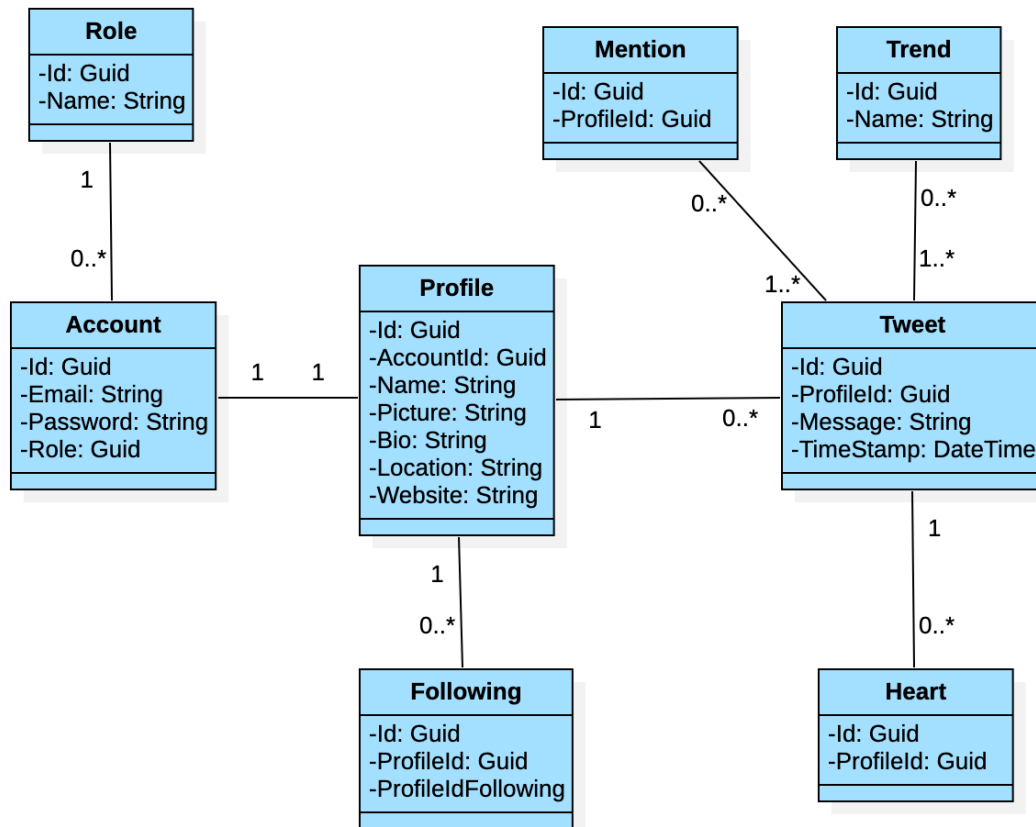Figure [6] shows the class diagram.
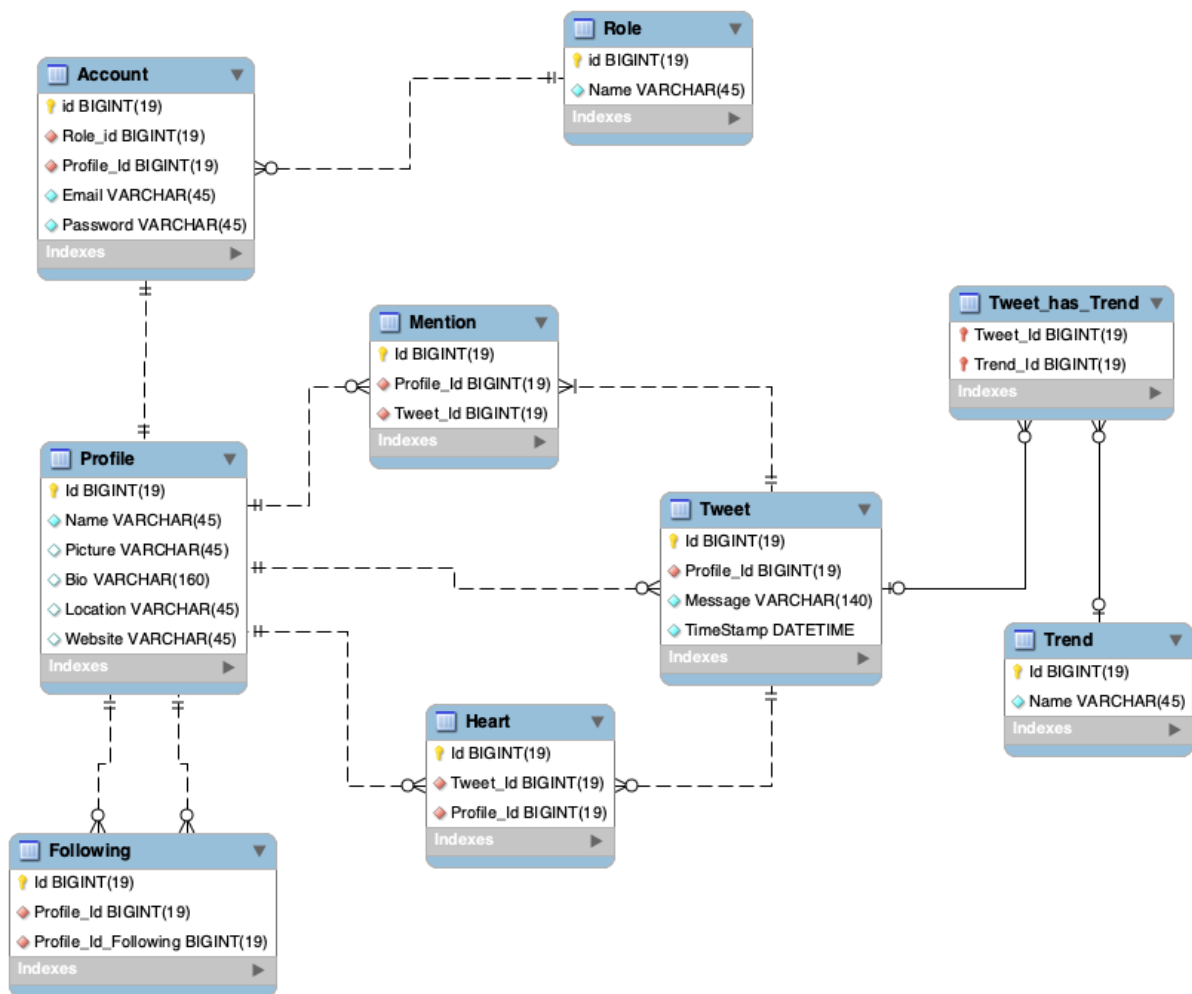

Figure 6 Class diagram

Database diagram

Figure [7] shows the database diagram.

Figure 7 Database diagram