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.

Frontend framework

For the frontend React was chosen because it is most familiar frontend framework for me and with microservices it doesn't really matter what frontend technology you use. Alternatives like Vue or Angular would take too much time and effort to learn for this project since it is not the focus.

Backend framework

For the backend C# with the ASP.NET Core framework was chosen because it is most familiar backend framework to me and with the microservices that will be made for Kwetter it can be used for of them, since Identity and Entity Framework can easily be implemented. Alternatives like Java or PHP but these would take too much time to relearn and implement for this project since this is not the focus.

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 must be chosen to create an API gateway. Important for the gateway is that it is well suited for a microservices architecture that need unified points of entry into their system. It should be fast, scalable, cross-platform and provides routing and authentication among many other features. The Following messaging technologies have been investigated: Ocelot, Kong, Azure API management and Nginx. The comparison can be seen in the table [1] below. In the end the choice was made to use Nginx.

Table 1 API gateway comparison

	Weight	Ocelot	Kong	Azure API management	Nginx
Integration	0.13	🟩	🟩	🟩	🟩
Docker support	0.10	🟨	🟩	🟩	🟩
Documentation	0.20	🟩	🟩	🟩	🟩
Community	0.15	🟩	🟩	🟩	🟩
Open-source	0.12	🟩	🟩	🟨	🟩
Monitoring	0,15	🟨	🟩	🟩	🟩
Compatibility	0,15	🟨	🟨	🟩	🟩
Total	1.00	0.60	0.85	0.88	1.00

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 must be chosen to create an event bus. The Following messaging technologies have been investigated: RabbitMQ, Kafka, ActiveMQ and NATS. It was important to have a good and popular choice for an open-source message broker that is easy to deploy for distributed systems. The comparison can be seen in the table [2] below. In the end the choice was made to use RabbitMQ.

Table 2 Event bus comparison

	Weight	RabbitMQ	Kafka	ActiveMQ	NATS
Integration	0.13	🟩	🟩	🟩	🟩
Docker support	0.10	🟩	🟩	🟩	🟩
Documentation	0.20	🟩	🟩	🟨	🟩
Community	0.15	🟩	🟩	🟨	🟩
Open-source	0.12	🟩	🟩	🟩	🟩
Monitoring	0,15	🟩	🟩	🟩	🟩
Compatibility	0,15	🟩	🟨	🟩	🟩
Total	1.00	1.00	0.85	0.65	1.00

Cloud service

For hosting the system into cloud, cloud services offered by Microsoft, Amazon and Google have been investigated to get a clear view on what is available and to decide on what to use for the Kwetter project.

Azure

Worldwide Azure offers more than 600 services. It offers services such as Database services like SQL databases, container services like an Azure Kubernetes Cluster, Identity services like the Azure Active Directory which provides single sign-on, multifactor authentication, and conditional access. Azure also offers Compute services like Virtual machines to run Linux or Windows, Networking services like load balancers for high availability and scalability of applications, Web services like an App Service to host web and mobile apps, Analytics services like the Azure Monitor to monitor performance hosted applications, health and metrics of an app, AI + machine learning services like Metrics advisor to proactively monitor metrics and diagnose issues, Internet of things (IoT) services like IoT Hub to connect, monitor, and manage IoT assets, Integration services like Logic apps for automating things like sending email notifications, and much more. With Azure services are free or cheap to implement and pricing scales with use, every student also gets $100 worth of credit.

Amazon Web Services
Worldwide Amazon Web Services (AWS) offers more then 200 fully featured services. It offers services such as Compute services like Amazon EC2 virtual servers for secure and resizable compute capacity for workloads, Storage services such as Amazon Simple Storage Service (S3) as object storage to retrieve any amount of data from anywhere, Database services such as Amazon Aurora for high performance relational databases, Networking & Content Delivery services such as Amazon API Gateway for building, deploying and managing APIs, Analytics services such as Amazon OpenSearch Service to search, visualize, and analyze large amounts of text and unstructured data, Machine Learning services such as Amazon Lookout for Metrics to automatically detect anomalies in metrics and identify their root cause, Security, Identity, & Compliance services such as Amazon Cognito for identity management for apps, and much more. Some of the AWS services have a (12 month) free trial.

Google Cloud
Worldwide Google Cloud offers more then 100 products. It offers services such as Compute services, Storage services, Database services, Networking services, Operation services, Developer Tools services, Data Analytics services, AI and Machine Learning services, Vertex AI, AI Platform, and Accelerators services, Industry Solutions services, API Management services, Hybrid and Multi-cloud services, Migration services, Security and Identity services, Identity & Access services, Google Distributed Cloud services, User Protection Services, Serverless Computing services, Internet of Things (IoT) services, Management Tools services, Healthcare and Life Sciences services, Media and Gaming services, and much more. For new customers of Google Cloud, a free $300 credit is available.

For the Kwetter system, the Database service from Azure to host the SQL databases was chosen, the Container service from Azure to host the Kubernetes cluster with the Azure Kubernetes Service and the Analytics service to host the Application Insights was chosen. The credit is enough for this. If the usage of the application will grow tremendously, these services can automatically scale up with the demand, within the configuration the hardware can also be upgraded.

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

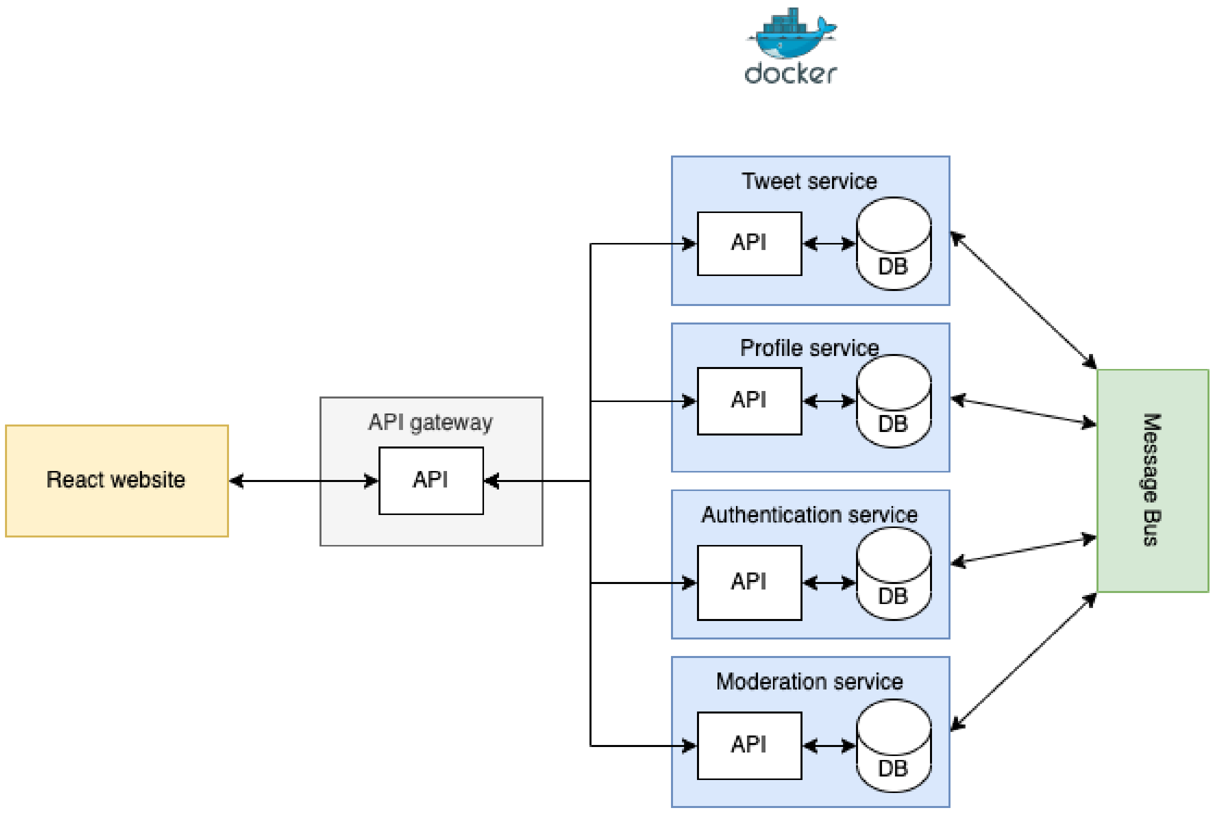


Figure 1 Global architecture

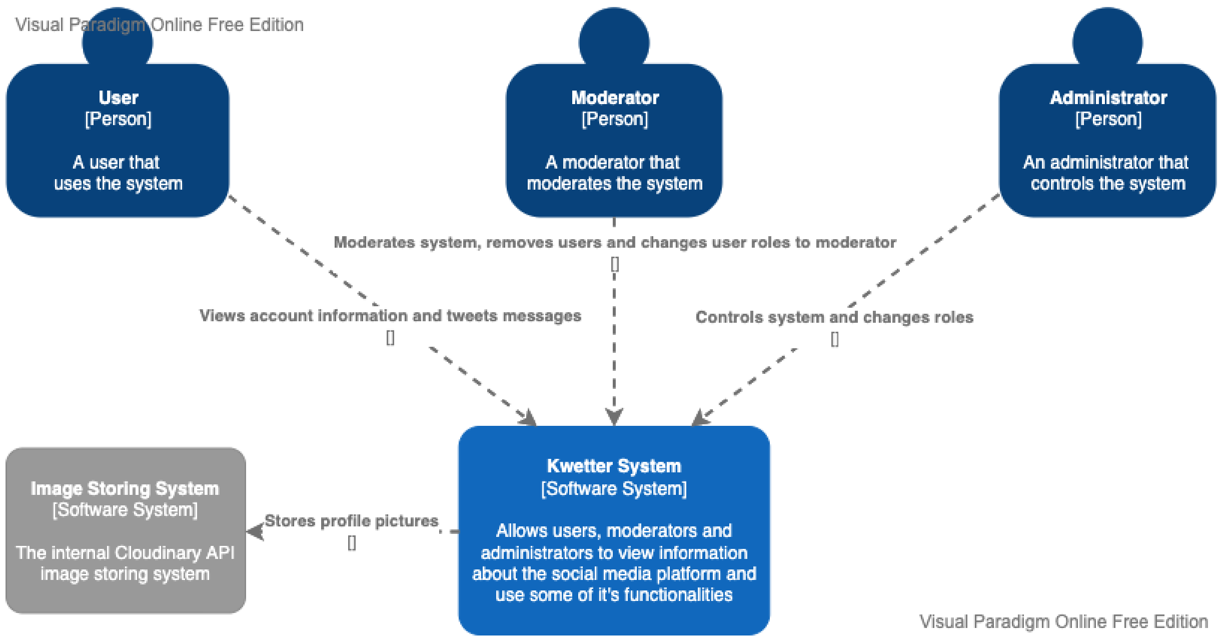
Security by design

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.

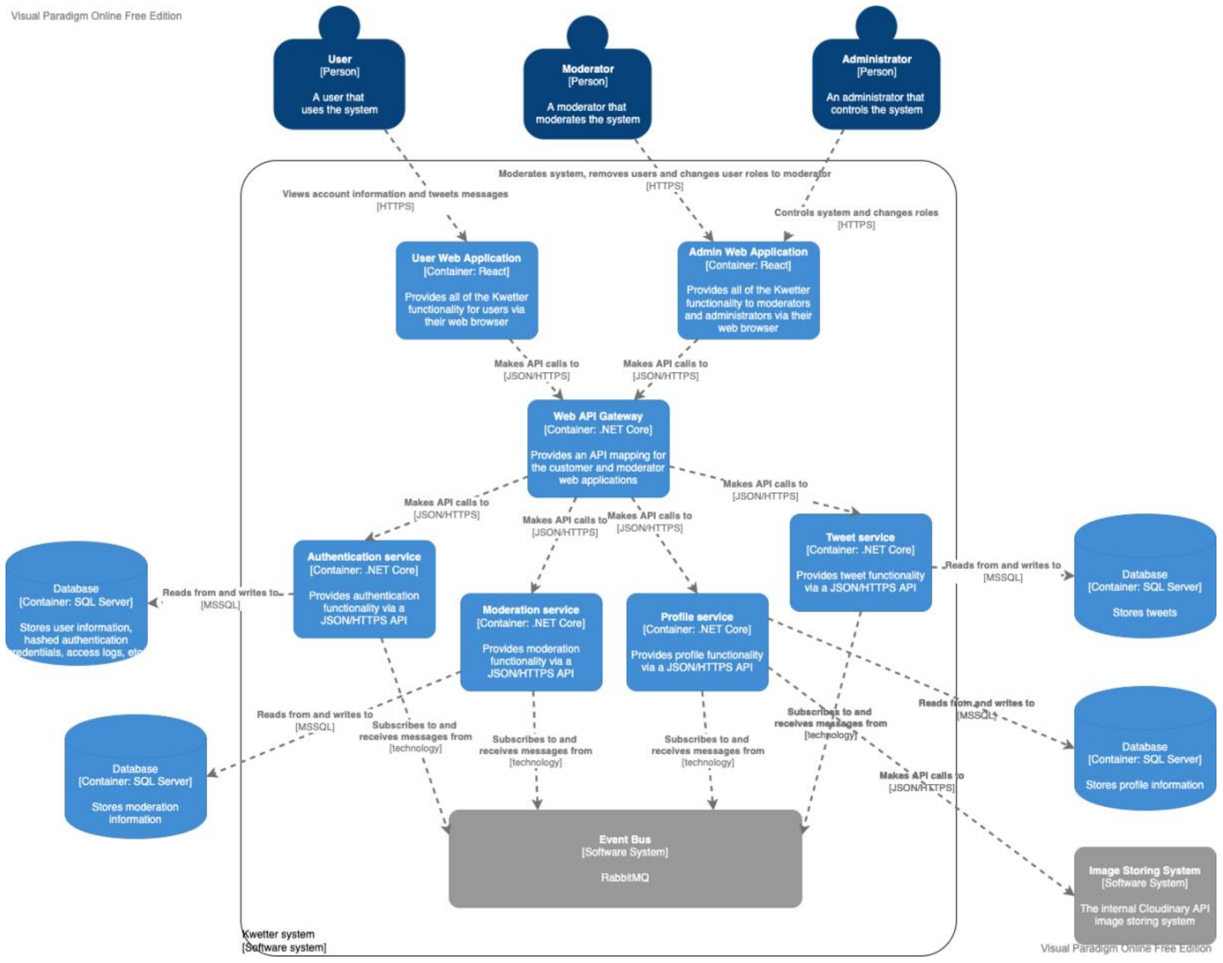


Visual Paradigm Online Free Edition
User
[Person]
A user that uses the system
Moderator
[Person]
A moderator that moderates the system
Administrator
[Person]
An administrator that controls the system
Moderates system, removes users and changes user roles to moderator
[]
Views account information and tweets messages
[]
Controls system and changes roles
[]
Image Storing System
[Software System]
The internal Cloudinary API image storing system
Stores profile pictures
[]
Kwetter System
[Software System]
Allows users, moderators and administrators to view information about the social media platform and use some of it's functionalities
Visual Paradigm Online Free Edition

Figure 2 Context diagram

Container diagram

Figure [3] shows the container diagram.

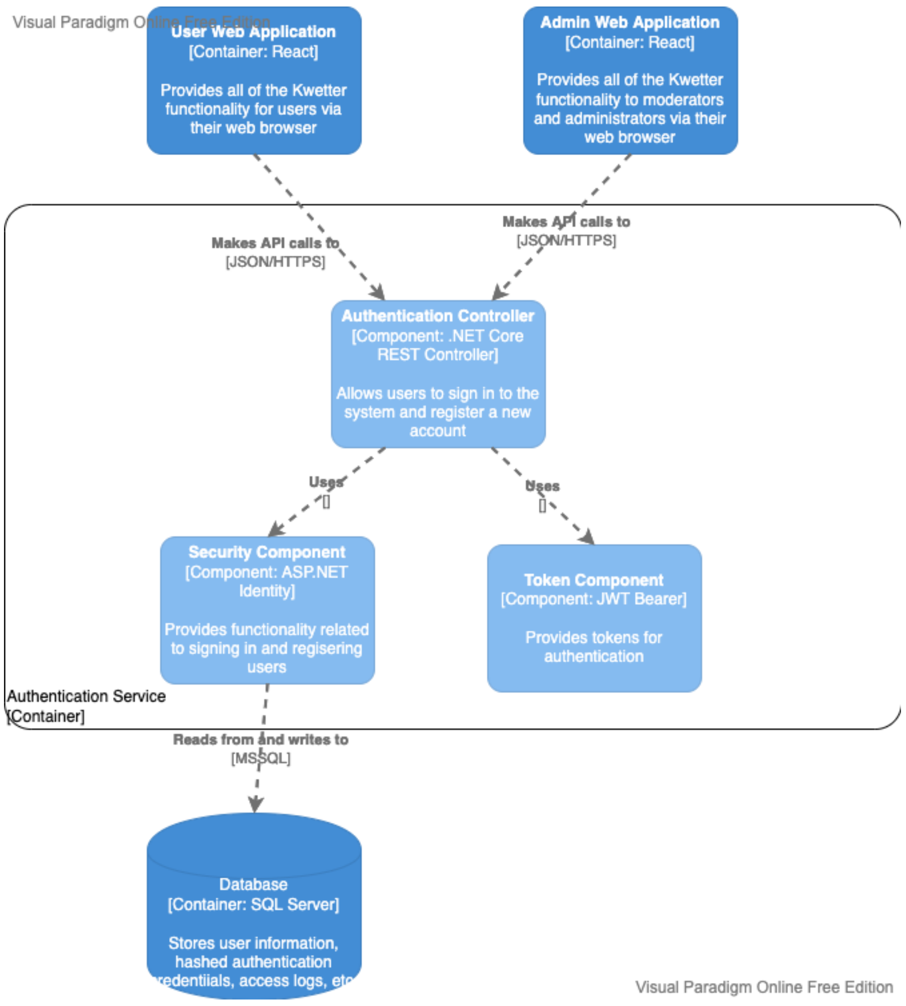


Visual Paradigm Online Free Edition
User [Person] A user that uses the system
Moderator [Person] A moderator that moderates the system
Administrator [Person] An administrator that controls the system
Moderates system, removes users and changes user roles to moderator [HTTPS]
Views account information and tweets messages [HTTPS]
Controls system and changes roles [HTTPS]
User Web Application [Container: React] Provides all of the Kwetter functionality for users via their web browser
Admin Web Application [Container: React] Provides all of the Kwetter functionality to moderators and administrators via their web browser
Makes API calls to [JSON/HTTPS]
Makes API calls to [JSON/HTTPS]
Web API Gateway [Container: .NET Core] Provides an API mapping for the customer and moderator web applications
Makes API calls to [JSON/HTTPS]
Makes API calls to [JSON/HTTPS]
Makes API calls to [JSON/HTTPS]
Makes API calls to [JSON/HTTPS]
Authentication service [Container: .NET Core] Provides authentication functionality via a JSON/HTTPS API
Moderation service [Container: .NET Core] Provides moderation functionality via a JSON/HTTPS API
Profile service [Container: .NET Core] Provides profile functionality via a JSON/HTTPS API
Tweet service [Container: .NET Core] Provides tweet functionality via a JSON/HTTPS API
Database [Container: SQL Server] Stores user information, hashed authentication credentials, access logs, etc
Reads from and writes to [MSSQL]
Database [Container: SQL Server] Stores tweets
Reads from and writes to [MSSQL]
Database [Container: SQL Server] Stores moderation information
Reads from and writes to [MSSQL]
Database [Container: SQL Server] Stores profile information
Reads from and writes to [MSSQL]
Subscribes to and receives messages from [technology]
Subscribes to and receives messages from [technology]
Subscribes to and receives messages from [technology]
Subscribes to and receives messages from [technology]
Makes API calls to [JSON/HTTPS]
Event Bus [Software System] RabbitMQ
Image Storing System [Software System] The internal Cloudinary API image storing system
Kwetter system [Software system]
Visual Paradigm Online Free Edition

Figure 3 Container diagram

Component diagram

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



Visual Paradigm Online Free Edition
User Web Application
[Container: React]
Provides all of the Kwetter functionality for users via their web browser
Admin Web Application
[Container: React]
Provides all of the Kwetter functionality to moderators and administrators via their web browser
Makes API calls to
[JSON/HTTPS]
Makes API calls to
[JSON/HTTPS]
Authentication Controller
[Component: .NET Core REST Controller]
Allows users to sign in to the system and register a new account
Uses
[]
Uses
[]
Security Component
[Component: ASP.NET Identity]
Provides functionality related to signing in and regisering users
Token Component
[Component: JWT Bearer]
Provides tokens for authentication
Authentication Service
[Container]
Reads from and writes to
[MSSQL]
Database
[Container: SQL Server]
Stores user information, hashed authentication credentiials, access logs, etc
Visual Paradigm Online Free Edition

Figure 4 Component diagram

Deployment diagram

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

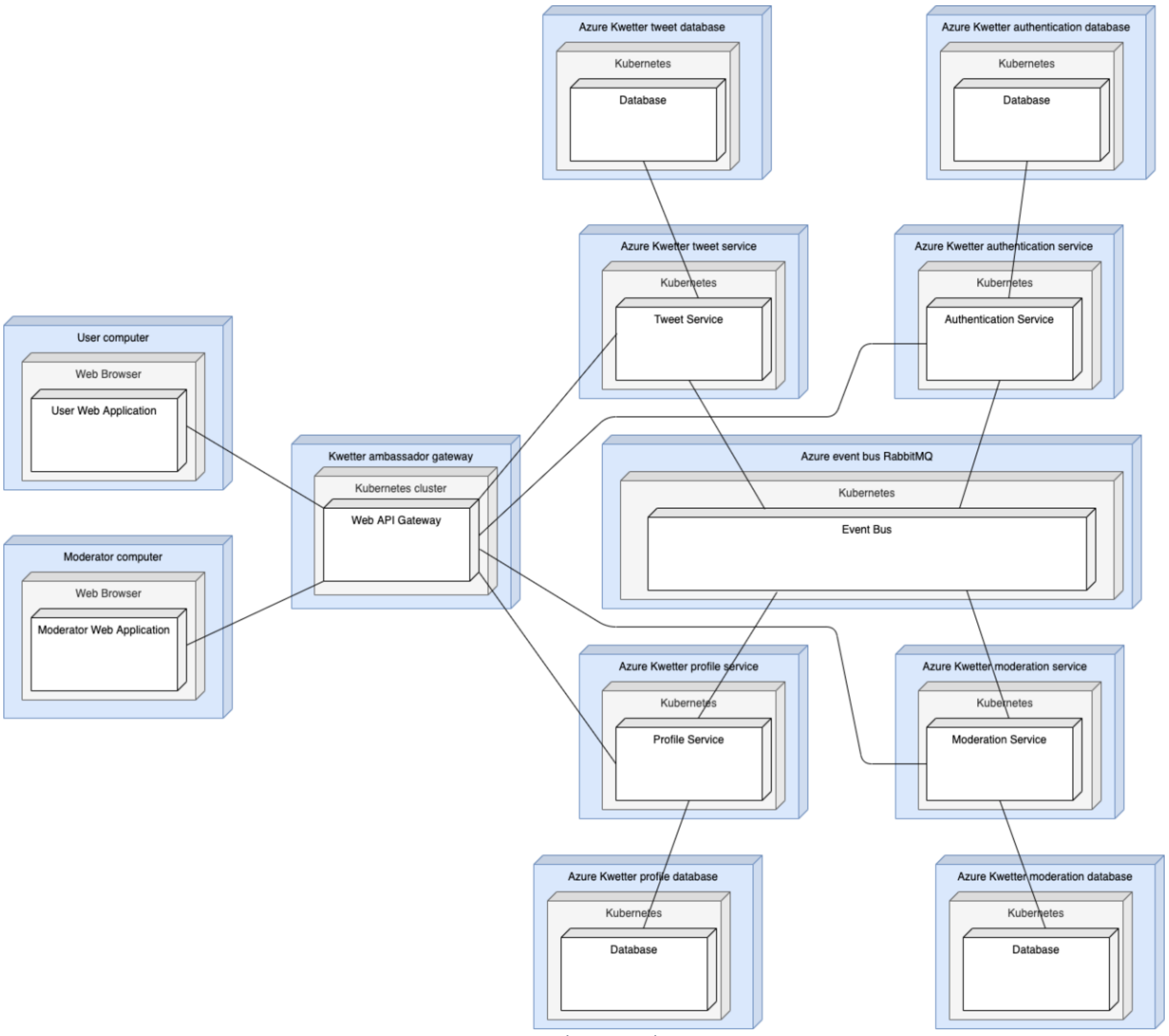


Figure 5 Deployment diagram

Class diagram

Figure [6] shows the class diagram.

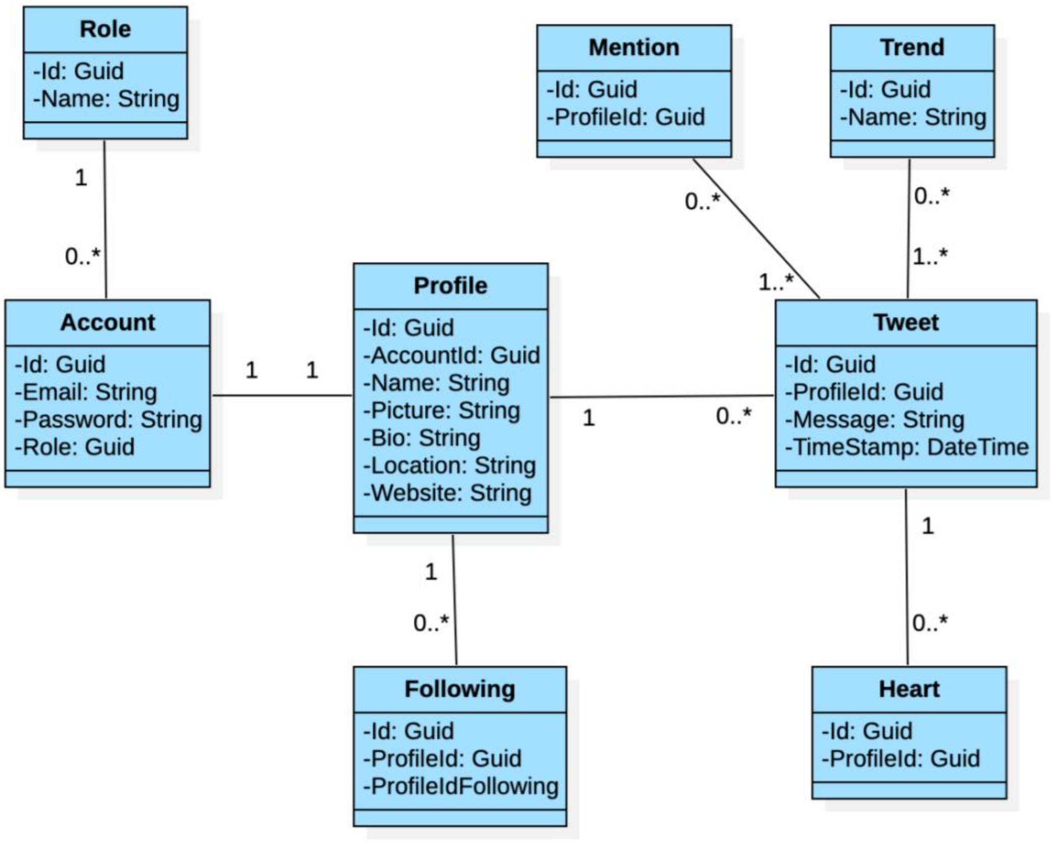


Figure 6 Class diagram

Database diagram

Figure [7] shows the database diagram.

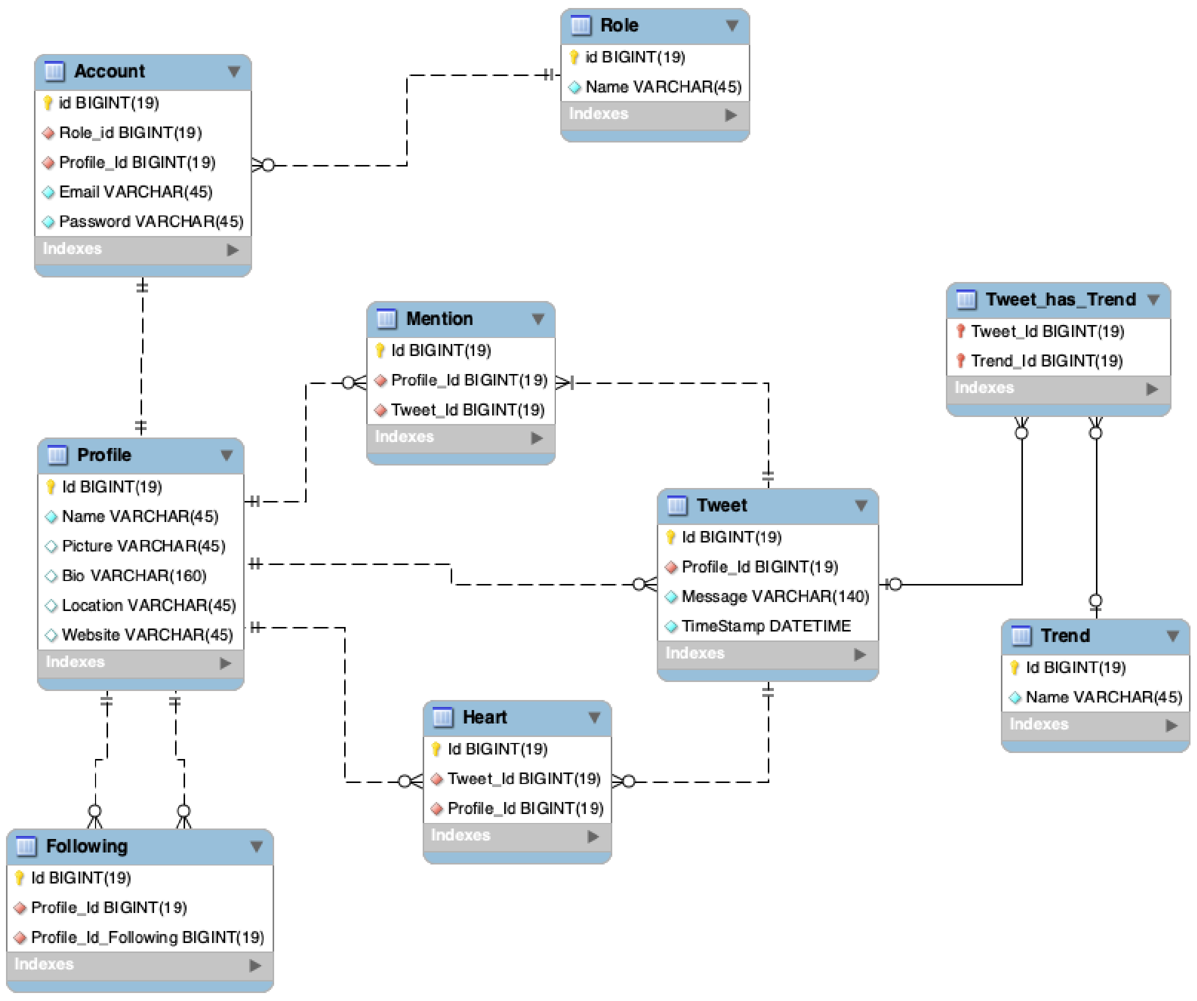


Figure 7 Database diagram