1. Introduction

1.1. Problem description

The application is an online wine web shop, where users can buy/order various kinds of wine from different price points. These wines come in all kinds of categories:

- Red wine
- White wine
- Rosé Wine
- Sparkling Wine
- Dessert Wine
- Organic Wine

1.2. Functional and non-functional requirements

Functional requirements

- As a user you can add & remove wines from the shopping cart.
- As a user you should be able to place and order.
- As a user you can search and find a wine depending on the category, price, and name.
- As a user you should be able to use the site as a signed in user or a guest.
- The web site should have some authentication to check if the user is above the legal limit for buying alcohol.
- As a user, you should be able to see your order history (see previous orders).
- The administrator must be able to add new products & edit old ones.

Nice-to-have

- As a user you can create a gift basket with 5 diverse kinds of wines.
- The web shop has premade baskets with an assortment of different wines (This could be depending on the season).
- The web shop sell wine associated products such as:
    - Food products such as cheese and biscuits
    - Wine coolers
    - Wine glasses

Non-functional requirements

- Scalability:
  databases: amount of data, number of requests, query optimization
    - application: microservices - example: using Kubernetes to orchestrate running of the containerized microservices
    - Portability: using containerized microservices
- Security: Authentication and authorization, VPC, …
- Interoperability: designing proper APIs
- The entire system should be deployed in the cloud. Each service - microservice or database server - can run on a different cloud. We are aiming for characteristics like:
    - elasticity
    - high availability
    - low latency

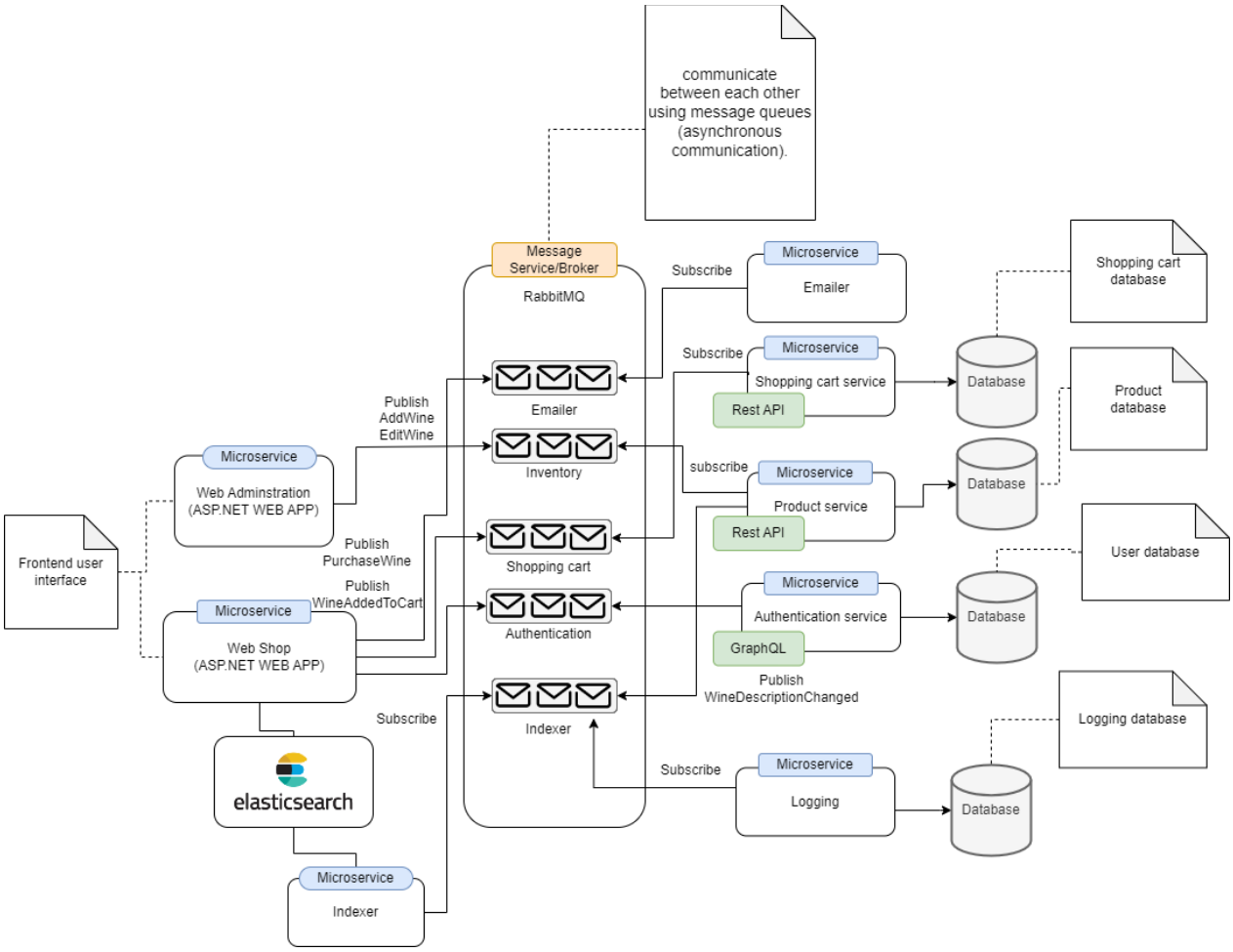
Other requirements

- Microservices should communicate with each other using message queues (asynchronous communication).
- The backend should communicate with the frontend using REST API or GraphQL. You must implement both ways in your project - for example one microservice can use REST API and another one GraphQL, or you can implement both REST API and GraphQL for a single microservice.
- Use serverless functions for some of the tasks (for example image processing after uploading an image)
- Logging and monitoring system - for the production
- Authentication implementation + 3rd party integration (for example Gmail, Facebook)
- Email service (for example for email verification)
- Admin service - there should be some extra GUI and backend logic for the admin role.

1.3. Explanation of choices for the technology stack (databases, programming languages,
frameworks, etc).

- GraphQL (database)
- Trello (managment of project tasks)
- React or angular (frontend)
- C#/.NET (Backend)
- OneDrive (documents)

2. System architecture



2.1. Introduction to the microservices architecture + schema of the whole system

2.2. Microservice 1 description

2.3. Microservice 2 description

2.4. Microservice 3 description

2.5. …

2.6. Communication between microservices

2.7. Description of the patterns and techniques used in the project

2.7.1. CQRS

2.7.2. Immutable data (Tombstone pattern and Snapshot pattern)

2.7.3. Idempotence

2.7.4. Commutative message handlers

2.7.5. Saga pattern

2.7.6. Caching

2.7.7. …

3. Deployment

3.1. Introduction to the cloud deployment

- Forventer at anvende Azure

3.2. Description of used technologies

3.3. CI/CD pipeline description

- Ikke besluttet

3.4. Monitoring and logging of the deployed system

- Ikke besluttet

4. Project management and team collaboration

4.1. Introduction to the project management and team collaboration

4.2. Description of the methods used during the project

4.3. Versioning strategies for the source code, databases, and APIs

4.4. Documentation strategy

5. Conclusion

5.1. Advantages and challenges of the distributed systems (microservices architecture)

5.2. Pros and cons of used patterns like CQRS etc.

5.3. Scalability

5.4. Possible improvements

6. References

7. Appendix