System requirements

Project description

The application is an online wine web shop, where users can buy/order different kind 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

Product idea

- beverage platform for wine.

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 different kinds of wines.
- The web shop have premade basket with a 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, amount 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 whole 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 between 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 admin role.