9/1/2022

# EAuction Project

Software Architecture Document

Thaw Dar Shwe, Ngu War (Cognizant)
NGUWAR.THAWDARSHWE@COGNIZANT.COM

## Table of Contents

# 1. Introduction

## 1.1. Purpose

This document provides a comprehensive architecture and design overview of the EAuction project, using a number of design considerations to depict different aspects of microservice architecture. It is intended to capture and convey the significant architectural decisions made on the system.

## 1.2. Scope

This document provides an architecture and design overview of the seller and buyer functions of the EAuction project.

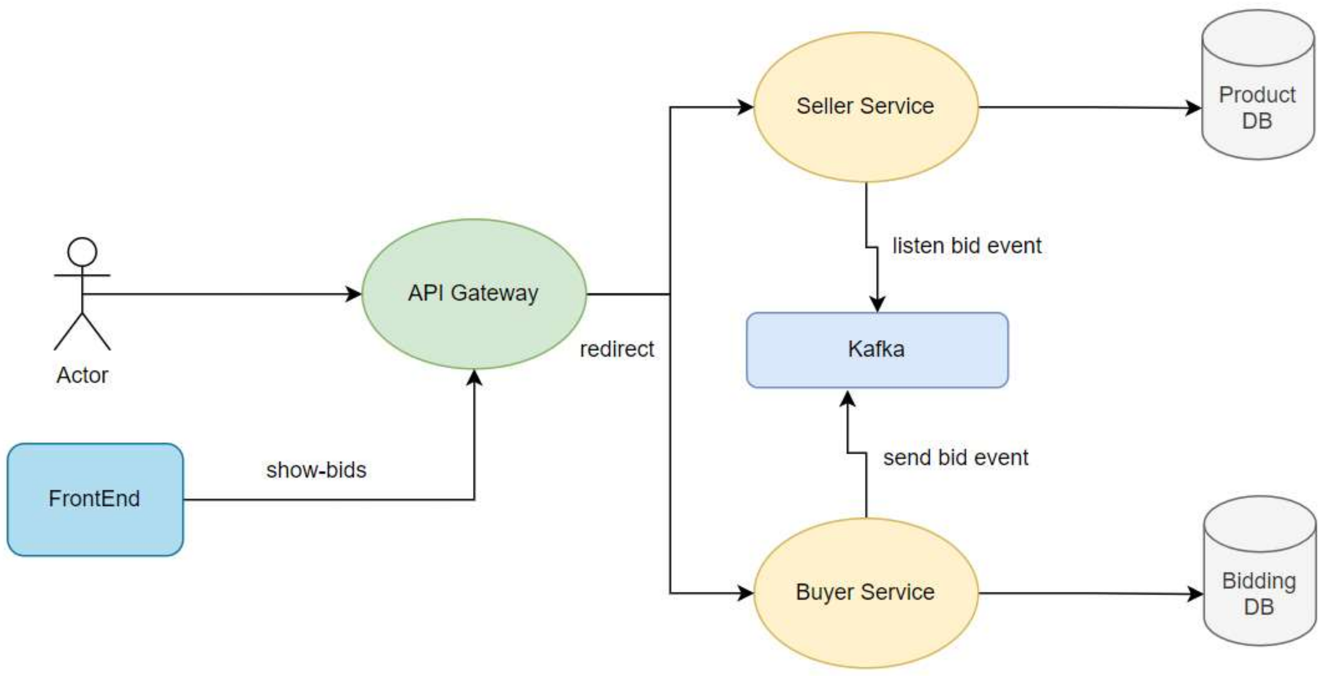
## 1.3. Design goals and constraints

The following are key requirements and constraints that have a significant bearing on the architecture.

1. The system should implement event-driven architecture for communication between microservices.
2. Performance on retrieving product details from frontend application must be taken into consideration as the architecture is being developed.
3. All APIs should be protected and redirected from the API gateway.
4. All APIs should be scalable independently using cloud technologies.
5. The system should implement prevention based on security considerations.

# 2. Architecture

## 2.1. Architecture diagram



## 2.2. Design implementation

EAuction project will decompose the microservices by business capabilities: Seller and Buyer functions. This approach will give easier maintainability of microservices by focus and independence in scaling them as required without interfering with each other.

Each microservice will connect to PostgreSQL for own database, which will eventually be deployed to Amazon RDS in project deployment. This database per service approach will give loosely coupled architecture so that they can be developed, deployed and scaled independently. Kafka message broker will be used for event-driven architecture which will communicate events between microservices.

The following are the detailed description of each component in the diagram.

- Seller Service

  This is the microservice developed for Seller functions such as creating a new product, deletion, and viewing product information.

- Buyer Service

  This is the microservice developed for Buyer functions such as placing a new bid, and upating the bid amount.

- Kafka Message Broker

  This is the message broker which is used for sending bid event from Buyer service to Seller service in an async manner.

- Product Database

  This is the database to store product-related information used by Seller microservice.

- Bidding Database

  This is the database to store bid-related information used by Buyer microservice.

- API Gateway

  This is the API Gateway to protect internal rest endpoints and redirect the call to them from the frontend application.

## 2.3. Non-functional requirements considerations

- Seller Service has to be scaled up/down as demand during the bidding period for the products as it exposes the rest endpoint to view product details and its related bid information.
- Buyer Service should be able to scale up/down independently as it exposes the rest endpoint to bid on the product by many potential buyers. Kafka message broker will be used for bid creation events for better performance, and will follow the eventual consistency model.
- Internal rest endpoints will be protected by API Gateway and it will redirect the call from external to microservices accordingly.
- The system will validate user inputs at the backend for injection attack and SQL injection by using JPQL and @param on queries.
- Microservices will be deployed to Amazon ECS to achieve fault tolerance and resiliency.

## 2.4. Technologies to be used

The following are the technologies to be used in Backend Microservice Applications.

- Spring Boot Application
  - Eureka Service Discovery for API service discovery
  - Resilience4j for circuit breaker
  - Spring Cloud API Gateway for redirecting microservice calls
  - Mapstruct for mapper
  - OpenFeign for internal microservice call
  - Junit Mockito for unit testing
- PostgreSQL for database
- Kafka for message broker
- ELK for logging and monitoring
- Swagger4 for API documentation
- SonarQube for code coverage
- Docker

## 2.5. Git

https://git03.iiht.tech/nguwar.thawdarshwe/eauction.git

Kindly request the access to this private repository for code review. Thank you.