CS 6314 – Web Programming Languages Fall 2018

Course Project-Live Auction

Responsive web site and scalable web application based on the Service Oriented Architecture

BidBig

Group Name:- BidBig

Group Members:-
1.Mareena George Mekkattil
2.Shubhangi Khandelwal
3.Ujjawal Mishra

TABLE OF CONTENTS

I. INTRODUCTION

This report illustrates the implementation details of the Course Project-“BidBig”- Live Auction-a responsive website and scalable web application based on the Service Oriented Architecture. The architectural diagram describing how the components interact with each other is clearly depicted. A brief overview of modules, Web Services, Microservices and technologies used is explained in this report.

II. ARCHITECTURE

We have implemented using Microservice Architecture. BidBig has five core Web services.

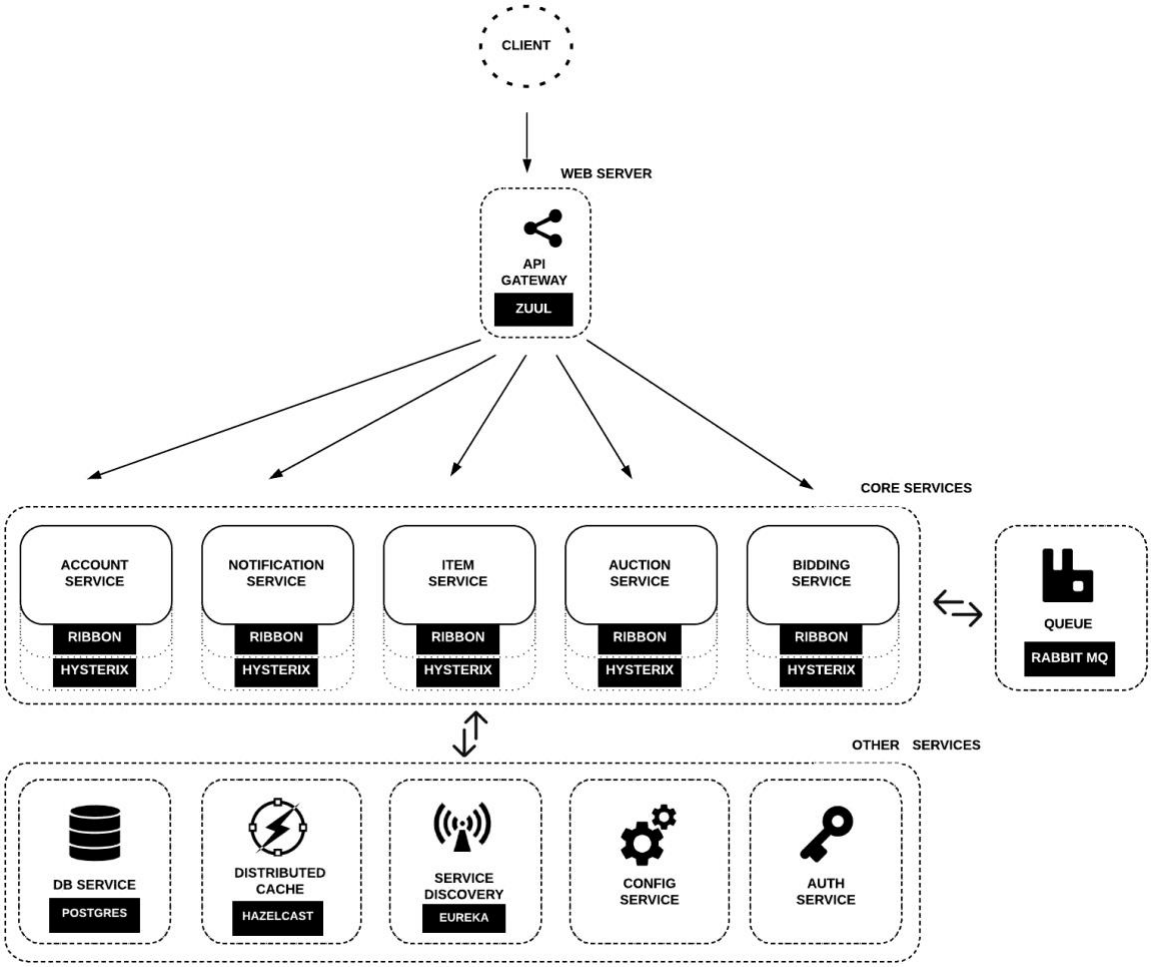
1. Account Service
2. Notification Service
3. Item Service
4. Bidding Service(Microservice)
5. Auction Service

Other services are:-

1. Auth Services
2. Gateway Services
3. Service Discovery
4. Config Service
5. DB Service
6. Queue Service
7. Distribute Cache Service

All the services are independently deployed applications organized with business logic. This project implements the following architectural patterns for distributed systems.

Infrastructure Services



III. MODULES

Core Services

1. **Account Services**
   This service contains general input logic and validation: login, register a new regular user, and account settings.

2. **Notification Services**
   This service contains contact information of the user and notification settings like forgot and reset password). Scheduled workers collect other required information from other services and send notifications.

3. **Item Service**
   This service contains creating, updating, and deleting item details including item image. This also helps in retrieving items of the specific user.

4. **Bidding Service**
   This service contains new bid creation, retrieving all bids of a specific item and highest bid of the specific auction.

5. **Auction Service**
   This service contains creating new auctions by admin. This also enables the user to auction their item.

Other Services

1. **Auth Service**
   Authorization responsibilities are completely extracted to a separate server, which grants OAuth2 tokens for the backend resource services. Auth Server is used for user authorization as well as for secure machine-to-machine communication inside a perimeter. Password credentials grant type for users authorization and Client Credentials grant for microservices authorization is used in this project.

2. **Config Service**
   Spring Cloud Config provides server and client-side support for externalized configuration in a distributed system. With the Config Server, we have a central place to manage external properties for applications across all environments. In this project, we used native profile in the Config Server for loading config files from the local classpath. We can see the shared directory in Config service resources.

3. **Gateway Services**
   In this project, we had implemented an API Gateway that is the single entry point for all clients. This is used to handle requests either by routing them to the appropriate backend service or invoking different services and combining the results together. We have broken down our API into different microservices such as Account service, notification service, item service, bidding service and auction service. We have used Zuul, an edge service from Netflix which enable dynamic routing, monitoring, resiliency and security. We used Zuul for storing static content and for routing requests to appropriate microservices. Zuul uses Service Discovery mechanism to locate different service instances and also Circuit breaker and Load Balancer. All Web Services have authentication/authorization to allow only a valid user to access/modify his/her data.

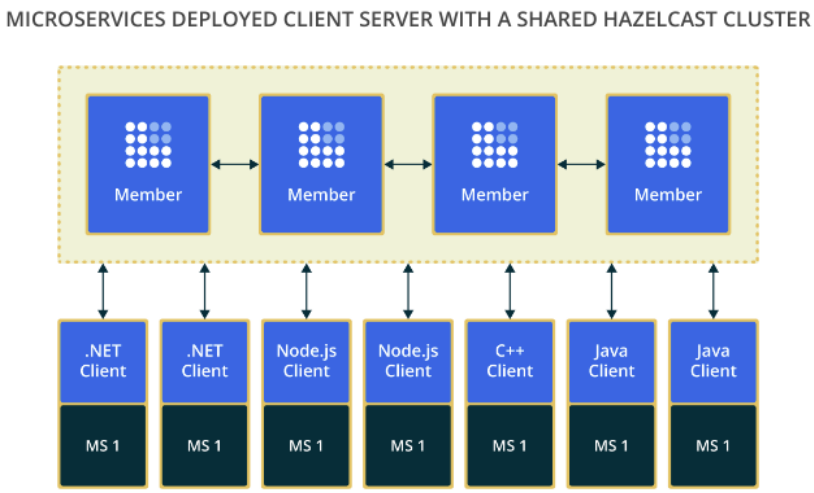
4. **Service Discovery**
   We have used Netflix Service Discovery Eureka in this project. Eureka is a REST based service that is primarily used in the AWS cloud for locating services for the purpose of load balancing and failover of middle-tier servers. On application startup, it registers with Eureka Server and provides metadata and Eureka receives heartbeat messages from each instance of service. The instance is removed from the registry if the heartbeat fails over a configurable timetable.

5. **Database Service**
   This project uses Postgres database to store all data. Single Postgres with multiple databases are used for services.

6. **Queue Service**

This Project uses RabbitMQ for implementing queue. This helps in Bidding Service functioning based on First-come-first-serve-basis. We encapsulate a task such as incremented bids/further bids during the slot as a message and send it to a queue. The worker process pop the tasks and eventually execute the job.

7. **Distribute Cache Service**
   Hazelcast architecture has the features like scalability, in-memory backups, and automatic eviction, which adds the features to Hibernate world. Hibernate ships with a first-level cache associated with the Session object that principally reduces the number of SQL queries within a given transaction. For example, multiple changes will only result in a single update statement. Hibernate first-level accesses the database for every single query. That can be slow and unpredictable depending on the database load and size.



Load balancer, Circuit breaker and Http client
The Netflix OSS also provides tools such as Ribbon, Hystrix, and Feign. Ribbon is a client-side Inter Process Communication library with load balancing, fault tolerance, multiple protocol support, caching and batching. Hystrix is a latency and fault tolerance library designed to isolate points of access to remote systems, services and 3rd party libraries, stop cascading failure and enable resilience in complex distributed systems where failure is inevitable. Feign is a Java to HTTP client binder.

IV. TECHNOLOGIES

- Web Client -React
- Microservice/Service-Spring Cloud
- Distributed Cache-Hazelcast
- Queue- RabbitMQ
- Database-PostgreSQL
- Container- Docker

We have built our website's client-side Graphical User Interface (GUI) using React Redux. We have also used responsive HTML/CSS/JavaScript templated such as Bootstrap in the UI. We have used React because of its single page application feature and its efficient update and render on updated states. Microservices are based on Spring Cloud. Initially we planned to

develop using Django, but since Spring Boot is easy to develop, we moved to Spring Boot. Spring cloud provides powerful tools that enhance Spring Boot applications behavior to implement those patterns.

The backend is done using PostgreSQL. We have used single Postgres and different Schema/DB for both account and Auth services. Docker make it easier to create, deploy, and run applications by using containers. Containers allow a developer to package up an application with all of the parts it needs, such as libraries and other dependencies, and ship it all out as one package. All the services are running on Docker containers. We use two types of services- core services and auxiliary services. Auxiliary services include Gateway, config service, Auth service, Service Discovery. RabbitMQ Docker container is used as queue. Two containers are used for distributed caching. Distributed Cache is implemented using Hazelcast. Hazelcast has the features like NoSQL Key Value Store, Web Session Clustering, Cloud or SaaS app scalability, Oracle Coherence Replacement, etc.

V. FUNCTIONALITIES

We have implemented a real-time online auction web application for selling/purchasing items.

1. New regular user registration
   - A new user can create a new account through Register/ Sign Up Page. User is required to provide username, email and password.

2. Existing user functionalities:-

   - Login - with username and password
   - Logout
   - Edit and view profile information such as address and phone number.
   - Forgot Password – Sending email with Reset Password link to a short-term page that lets them set a new password.
   - User can post items for auctioning with item title, description, initial/minimum bidding price, and item photo.
   - User can view the auction schedule for one month.
   - User can view the different items available for bidding during a particular date.
   - Can delete user's own posted items.
   - Can view the currently active bid item along with current or initial bidding price. User can also see the countdown timer with the remaining time left for bidding that specific item.
   - Can bid for a particular item with incremental amount during the given time slot. The time slot is set default as 10 am to 11 am with 24(maximum) number items for auction. So, each item is available for bidding for 5 minutes.
   - If a user has input incremented price value for an item, it will be served on the basis of first-come-first-serve policy. The highest bid user at the end of the bidding lot will be the winner.
   - All users have access to real-time updates of data on auction board.

3. All unavailable pages are retrieved with a pretty 404 error page with back page link to it.

4. Admin user has the following functionalities:-
    - Login
    - Logout
    - Admin has the provision to specify as default and change the auction time and number of slots.

VI. WEBSERVICE AND MICROSERVICES

Account Services

Method	Path	Description
GET	/accounts/users/{name}	Get account data
GET	/accounts/users/current	Get current account data
PUT	/accounts/users/current	Save current account data
POST	/accounts/users/	Register new account

Notification Services

Method	Path	Description
GET	/notifications/settings/current	Get current account notification settings
PUT	/notifications/settings/current	Save current account notification settings

Item Service

Method	Path	Description
POST	/item	Save item details
PUT	/item/{itemId}	Update item details
DELETE	/item/{itemId}	Delete an item
GET	/page/{userId}	Get iems of a specic user
GET	/downloadImage/{fileName:.+}	Get item image

Bidding Service

Method	Path	Description
GET	/bids/highest/{auctionId}	Get highest bid of specific auction
GET	/bids/all/{itemId}	Get all bids of specific item
POST	/bids/new	Create a new bid on current item

Auction Service

Method	Path	Description
POST	/auctions/create	Admin can create new auctions
POST	/auctions/list	User can list auctions
POST	/auctions/update	User can update auctions

Auth Service

Method	Path	Description
GET	/uaa/users/current	Get token of user
POST	/uaa/users/notifications/reset	Send username using Forgot password link
POST	/uaa/users/notifications/resetted	Reset password

FEATURES

1. High Performance:
   Perform distributed caching. Memcache is used for implementing distributed cache mechanism.

2. Client-Server Communication Encryption:
   The communication channel between the client (i.e. browser), website server, Web Services, and Microservices server are encrypted using TLS/SSL. HTTPS calls are made to the RESTful WebServices and Microservices using browser-based REST clients such as Postman.

3. Request/Response Compression:
   Perform compression of website server's response to the client using gzip.

VII. PROBLEMS ENCOUNTERED

- We faced few challenges while integrating frontend with the backend in terms of the learning curve that came with using react redux. However react redux helped us in building single page application quite easily.
- Challenge faced while integrating frontend with the backend as there was a learning curve involved with using react redux for the first time.
- Inter service SSL integration
- WebSocket OAuth2 integration
- Relational vs Non Relational schemas for scalability
- Relatively high latency due to SSL and distributed architecture

VIII. CONCLUSION

The project was completed successfully with all the required functionalities and features. There were 5 main core services such as Account service, notification service, item service, bidding service and auction service. The project implements distributed caching, Client-Server Communication Encryption, Compression, etc. All the services are independently deployed applications organized with business logic.