Distributed Systems

Assignment 3:

Web sockets and security

Student: Orfescu Petrisor

Group: 30241

1.Requirements:

Develop a chat microservice and an authorization component for the Energy Management System.

The authorization component should provide secured access of users to systems’ microservices.

The chat microservice should allow communication between the users and the administrator of the

system, allowing them to ask questions and receive answers.

2. Functional Requirements

Chat microservice:

➢ The front-end application displays a chat box where users can type messages.

➢ The message is sent asynchronously to the administrator, that receives the message together

with the user identifier, being able to start a chat with the user.

➢ Messages can be sent back and forth between the user and the administrator during a chat

session.

➢ The administrator can chat with multiple users at once.

➢ A notification is displayed for the user when the other administrator reads the message and

vice versa.

➢ A notification is displayed for the user (e.g., typing) while the administrator from the other

end of communication types of its message and vice versa.

Authorization component:

➢ One of the services is chosen as authorization server (e.g. User Microservice, or a newly

implemented microservice for authorization and authentication). This service generates

access tokens to the client application. The tokens will be used to access other

microservices.

3. Implementation Technologies

➢ Chat component: web sockets technology

➢ Authorization component: JWT based authorization - for user’s authentication and

authorization to all microservices.

o One authorization service that generates tokens that will be recognized by other

microservices which share the same secret key as the authorization service

4.Implementation Details

**The Chat Microservice** is implemented using WebSockets. I implemented a public chat where users can ask both for the opinion of the administrator and other users.

A white background with black text

Description automatically generated

The Backend of the microservice consists of three packages:

* Config
* Controller
* Model

Config contains a class called WebSocketConfig that implements the WebSocketMessageBrokerConfigurer interface and sets the topic and endpoint for the messages.

Controller contains a class called ChatController that implements the logic for message sending.

Model contains the ChatMessage model class, the message structure.

JWT Authentication is used in order to secure the application. Each time a user logs in and the credentials are valid, the Users Microservice returns a token signed with a specific secret key, which is saved in the browser’s localStorage.

public String generateToken(UserDetails userDetails) {  
 Map<String, Object> claims = new HashMap<>();  
 return doGenerateToken(claims, userDetails.getUsername());  
}  
  
private String doGenerateToken(Map<String, Object> claims, String subject) {  
 return Jwts.*builder*().setClaims(claims).setSubject(subject).setIssuedAt(new Date(System.*currentTimeMillis*()))  
 .setExpiration(new Date(System.*currentTimeMillis*() + *JWT\_TOKEN\_VALIDITY* \* 1000))  
 .signWith(SignatureAlgorithm.*HS512*, secret).compact();  
}

Any request that comes next has to contain the header: “Authorization : Bearer {token}”, otherwise the result will be forbiddenAccess.

The other microservices are configured in order to validate the token using the same secret key. That way, only a token generated by the users microservice can return a valid value and let the request trough.

5.Deliverables

5.1 Conceptual architecture of the distributed system.

The Conceptual architecture of the distributed system is illustrated in the diagram below (Conceptual Architecture):

A diagram of a computer program

Description automatically generated with medium confidence

Conceptual architecture

5.2 Deployment.

The project is deployed on Docker; the Deployment diagram of the system is illustrated below (Deployment Diagram).

A diagram of a computer

Description automatically generated

Deployment Diagram