E-commerce Application Implementation Exercise

Background

You have successfully completed the initial design phase for an e-commerce application with a "rush" delivery feature. We want you to implement a portion of this system, focusing on the frontend, a backend microservice, message queuing, and authentication. This exercise will help you gain hands-on experience with modern web technologies and cloud-native practices.

Objective

Develop a prototype of the e-commerce system that demonstrates the core functionality of the "rush" delivery feature, incorporates real-time updates, and ensures secure user authentication.

Technical Stack

- Frontend: Angular (latest stable version)
- Backend: Spring Boot 3.x
- Message Queue: RabbitMQ
- Authentication: Keycloak
- Containerization: Docker

Requirements

1. Frontend (Angular)

- Implement a responsive user interface for the e-commerce application.
- Create a product listing page with the ability to add items to a cart.
- Add a "rush delivery" option only for orders above $100.
- Integrate Keycloak for user authentication and authorization.

2. Backend (Spring Boot 3.x Microservice)

- Develop a RESTful API to handle product listings, cart management.
- Implement business logic for the "rush delivery" feature, including eligibility checks.
- Use cache system
- Use Spring Security with Keycloak adapter for securing endpoints.
- Use Spring AMQP for interaction with RabbitMQ.

3. Message Queue (RabbitMQ)

- Set up a RabbitMQ instance for asynchronous communication between services.
- Configure queues for order processing and status updates.

4. Authentication (Keycloak)

- Set up a Keycloak server with a realm for the e-commerce application.
- Configure client applications for both frontend and backend.
- Implement role-based access control (RBAC) with at least two roles: "customer" and "admin".

5. Containerization (Docker)

- Create Dockerfiles for the frontend, backend, RabbitMQ, and Keycloak services.
- Develop a docker-compose.yml file to orchestrate all services locally.

Detailed Implementation Guidelines

Frontend (Angular)

1. Use Angular Material or a similar UI library for consistent styling.

2. Implement lazy loading for optimized performance.
3. Use NgRx for state management, particularly for the shopping cart.
4. Implement interceptors for adding authentication tokens to API requests.
5. Use Angular's HttpClient for API communication.

Backend (Spring Boot 3.x)

1. Use Spring Data JPA for database operations.
2. Implement a layered architecture: Controller -> Service -> Repository.
3. Use DTO (Data Transfer Objects) for API requests and responses.
4. Implement custom exceptions and a global exception handler.
5. Use Spring Validation for input validation.
6. Implement unit and integration tests using JUnit and Mockito (nice to have).
7. Use Lombok to reduce boilerplate code.
8. Configure Swagger/OpenAPI for API documentation.

Message Queue (RabbitMQ)

1. Create needed queues .
2. Implement retry mechanism for failed message processing.

Containerization (Docker)

1. Use multi-stage builds for optimized Docker images.
2. Implement health checks for all services.
3. Use environment variables for configuration in Dockerfiles and docker-compose.yml.

Deliverables

1. Source code for the Angular frontend application.
2. Source code for the Spring Boot backend microservice.
3. Docker configuration files (Dockerfiles and docker-compose.yml).
4. README.md with setup instructions and any necessary documentation.

Evaluation Criteria

- Functionality: Does the application meet all the specified requirements?
- Code Quality: Is the code well-structured, documented, and following best practices?
- Security: Are proper security measures implemented, especially regarding authentication and authorization?
- Performance: Is the application optimized for high traffic scenarios?
- Containerization: Are Docker configurations correct and optimized?

Submission

Please submit your work as a Git repository hosted on GitHub. Ensure that your repository is private and provide access to your evaluators.

Good luck with your implementation!