Backend Systems Portfolio Assignment 04
Behind The Goal

Kerim Dincer, Yigit Savasir, Cagan Yetis

February 15, 2025

1 Introduction

This project focuses on implementing a distributed system using Hexagonal Architecture and gRPC as the API technology. Our system models football leagues, teams, matches, and statistics. The primary goal is to provide a scalable and modular architecture where different components (e.g., match statistics, league information) can be maintained independently.

Key Use Cases:

- Retrieving league and team information
- Fetching match statistics
- Creating and updating match and team records
- Implementing a paginated match listing
- Reading a League Table

Example: Creating and Deleting Teams via gRPC

- A team can be created using the ‘createTeam‘ method, which takes a team name and league ID and returns the newly created team.
- A team can be deleted using the ‘deleteTeam‘ method, where we provide the league ID, and all associated teams under that league are removed.

2 Software Architecture

Our architecture follows the principles of Hexagonal Architecture, which separates business logic from infrastructure concerns. The main components include:

2.1 Domain Layer

This contains core business logic, such as entities for leagues, teams, matches, match statistics, and team statistics. Each of these entities encapsulates domain-specific rules and behaviors.

2.2 Application Layer

This layer provides API interfaces and handles application logic, such as data transformations and interactions with repositories.

2.3 Infrastructure Layer

This includes database interactions using JPA and Hibernate, along with gRPC service implementations.

2.4 Business Logic

The core business logic includes:

- Calculating team statistics (e.g., points, wins, losses)
- Ensuring valid match entries (e.g., a team cannot play against itself)
- Implementing pagination to efficiently retrieve match data

2.5 Ports and Adapters

Ports define the entry points for external communication (gRPC services), and adapters implement the logic to interact with the database and other external services.

2.6 Challenges in Implementing Hexagonal Architecture

One of the most challenging aspects was correctly structuring the dependencies between the domain, application, and infrastructure layers while maintaining a loosely coupled design. Some specific difficulties included:

- Managing dependencies between gRPC services and JPA repositories.
- Avoiding bidirectional relationships in entities that could cause recursion.
- Ensuring correct unit and integration testing with mock dependencies.

3 API Technology

We chose gRPC due to its efficiency in handling structured data and its performance benefits over REST.

Advantages of gRPC:

- Strongly-typed contracts via Protocol Buffers
- High-performance communication using HTTP/2
- Automatic code generation for client-server communication

Trade-offs:

- More complex setup than REST
- Requires code generation, making iteration slower than JSON-based APIs

4 Implementation Details

Our implementation follows a microservice approach where the gRPC service acts as an intermediary between the clients and the database.

4.1 Pagination in Match Retrieval

Match results for a given league can be retrieved with pagination using:

```
message PaginatedLeagueRequest {
    int64 league_id = 1;
    int32 page = 2;
    int32 size = 3;
}
message PaginatedMatchListResponse {
    repeated Match matches = 1;
    int32 total_pages = 2;
    int32 total_elements = 3;
    int32 current_page = 4;
}
```

4.2 Adapter Mapping

Data transfer objects (DTOs) are mapped to entities using a combination of ModelMapper and manual mappings. This ensures that the application layer remains decoupled from the domain logic.

4.3 Framework Choice: Why Spring Boot?

We chose Spring Boot for several reasons:

- **Ease of Dependency Injection:** Spring Boot simplifies managing dependencies between different layers.
- **Integration with Hibernate and JPA:** Our system relies on JPA for ORM, and Spring Boot provides a seamless way to configure and use Hibernate.
- **gRPC Support:** Spring Boot allows smooth integration with gRPC using additional dependencies.
- **Testing Capabilities:** It provides robust support for unit and integration testing using frameworks like Mockito and Testcontainers.

5 Testing Strategy

We followed a layered testing approach:

5.1 Unit Tests

Mockito was used for mocking dependencies in service layer tests.

5.2 Integration Tests

Testcontainers were used to spin up a PostgreSQL instance for real database interactions.

5.3 Performance Testing

We used JMeter to simulate concurrent requests and measure system performance.

6 Learning Outcomes and Reflection

Throughout this project, we learned about structuring a distributed system with clear separation of concerns. Some key takeaways:

- gRPC provided efficient communication, but debugging required more effort than REST.
- Hexagonal Architecture helped maintain a loosely coupled system, but proper structuring of adapters was a challenge.
- Implementing pagination was essential for optimizing database queries and reducing unnecessary data transmission.
- We realized the importance of robust testing strategies in ensuring application stability.
- Database locking mechanisms (optimistic and pessimistic locking) were explored to handle concurrent updates efficiently.
- Spring Boot's integration with PostgreSQL and Testcontainers helped streamline testing in an isolated environment.
- We faced difficulties in managing bidirectional relationships in JPA but resolved them using proper '@OneToMany' and '@ManyToOne' configurations.

Future Improvements:

- Implement caching mechanisms for frequently accessed queries to improve performance.
- Introduce Kubernetes for service orchestration and scaling.
- Enhance the test coverage to include more edge cases.

Acknowledgment

This document was drafted and refined using GPT-4 based on an outline containing related information. The GPT-4 output was reviewed, revised, and enhanced with additional content. It was then edited for improved readability and active tense.