Migration from Monolithic to Microservices Architecture

Introduction

This document outlines the steps and considerations required to migrate an existing monolithic application to a microservices architecture. The current monolithic application includes the following components:

* **Services**: Student, Course, Enrollment, Department
* **Layers**: Service, Controller, Repository, Model, DTO

Each service is tightly coupled within the monolithic architecture. The goal is to decouple these services into independent, deployable microservices.

Preparation

Project Understanding and Analysis

* **Identify Modules**: Recognize and understand the current modules (Student, Course, Enrollment, Department).
* **Decouple Dependencies**: Comprehend dependencies between modules and data flow.
* **Database Structure**: Examine the current database structure and shared data.

Migration Plan

Define Microservice Boundaries

* **Service Decomposition**: Break down the monolithic application into smaller, manageable microservices:
  + Student Service
  + Course Service
  + Enrollment Service
  + Department Service

Establish Communication Strategies

* **API Gateway**: Implement an API Gateway to handle requests and route them to the appropriate microservice.
* **Inter-Service Communication**: Use REST/HTTP or messaging queues for communication between microservices.

Database Decomposition

* **Database Per Service**: Each microservice should have its own database to ensure data independence.
* **Data Synchronization**: Implement data synchronization mechanisms as needed.

Implementation Steps

Create Project Structure

Each microservice follows a similar structure for better organization and clarity.

Example: Student Service

student-service/

├── src/

│ ├── main/

│ │ ├── java/

│ │ │ ├── com/

│ │ │ │ ├── example/

│ │ │ │ │ ├── student/

│ │ │ │ │ │ ├── controller/

│ │ │ │ │ │ ├── dto/

│ │ │ │ │ │ ├── model/

│ │ │ │ │ │ ├── repository/

│ │ │ │ │ │ ├── service/

│ ├── resources/

│ │ ├── application.properties

├── pom.xml

Similarly, we have the structure for:

* Course Service
* Enrollment Service
* Department Service

Extracting Services

Each service have its own Models, Repositories, and Controllers.

Models

* Extract the model representing the data structure.

Repositories

* Extract the repository interface for database interactions.

Services

* Extract the business logic.

Controllers

* Extract the controller logic for handling HTTP requests.

Configuration and Deployment

1. **Spring Boot Configuration**:
   * Create an application.properties file specific to each microservice with database and server port configurations.

API Gateway and Discovery Service

* **API Gateway**: Implement an API Gateway to route requests.

Testing and Debugging

* **Unit Testing**: Ensure unit tests are written for each microservice.
* **Integration Testing**: Write integration tests to validate end-to-end scenarios.

Monitoring and Logging

* **Logging**: Implement centralized logging
* **Monitoring**: Implement monitoring and alerting

Conclusion

Migrating from a monolithic application to microservices involves careful planning and execution. This document provides a comprehensive guide to achieving a smooth transition. Implementing these steps will result in a scalable, maintainable, and efficient microservices architecture.