

# ERP 프로젝트 개발 가이드

## 1. 프로젝트 개요 및 목표

### 1.1 목표

본 프로젝트의 최종 목표는 실무 환경에서 사용되는 다양한 통신 방식(REST, gRPC, WebSocket)과 이종 데이터 저장소(MySQL, MongoDB, In-Memory)를 통합하여 하나의 엔터프라이즈 자원 관리(ERP) 시스템처럼 동작하는 유연한 마이크로서비스 구조를 구현하는 것이다.

다음 4가지 핵심 서비스를 독립적으로 개발해야 한다.

서비스명	배점	핵심 기능
Employee Service	20	직원 정보 관리 (CRUD)
Approval Request Service	30	결재 요청 생성 및 결재 단계 관리
Approval Processing Service	30	결재 승인/반려 처리 로직 및 대기열 관리
Notification Service	20	결재 완료/반려 결과 실시간 알림

## 2. 시스템 아키텍처 및 통신 구조

### 2.1 전체 시스템 아키텍처

본 프로젝트는 4개의 독립적인 서비스로 구성된다. 각 서비스는 역할에 맞는 프로토콜과 저장소를 사용한다.

### 2.2 서비스 구성 요소 요약

서비스명	핵심 기능	통신 프로토콜	저장소 유형
Employee Service	직원 CRUD	REST	MySQL
Approval Request Service	결재 요청 생성, 저장, 단계 관리	REST gRPC Client	MongoDB
Approval Processing Service	결재 승인/반려 처리	REST gRPC Server	In-Memory
Notification Service	실시간 알림	WebSocket	없음

### 2.3 실행 흐름 예시 (결재 승인 완료 시나리오)

- Requester**가 Approval Request Service의 POST /approvals를 호출하여 결재를 요청
- Approval Request Service는 MongoDB에 요청을 저장하고, **gRPC**를 통해 Approval Processing Service에 결재 정보를 전달
- Approver 3**이 Approval Processing Service의 POST /process/{approverId}/{requestId}를 호출하여 승인
- Approval Processing Service는 **gRPC**를 통해 Approval Request Service로 승인 결과를 전달
- Approval Request Service는 MongoDB를 업데이트하고, 다음 결재자가 남아있는지 확인
- 다음 결재자가 남아있다면 (예: Approver 7), **gRPC**로 결재 정보를 Approval Processing Service에 **재전달**하여 다음 단계를 시작
- 모든 결재가 완료되면, Approval Request Service는 Notification Service에 알림을 요청하고, Notification Service는 **WebSocket**을 통해 **Requester**에게 최종 완료 알림을 전송

### 3. 서비스 상세 구현 가이드

#### 3.1 Employee Service (REST + MySQL)

##### 3.1.1 테이블 구조 (scripts/init\_mysql.sql)

```
CREATE TABLE employees (  
  id BIGINT PRIMARY KEY AUTO_INCREMENT,  
  name VARCHAR(100) NOT NULL,  
  department VARCHAR(100) NOT NULL,  
  position VARCHAR(100) NOT NULL,  
  created_at DATETIME DEFAULT CURRENT_TIMESTAMP  
);
```

##### 3.1.2 REST API 상세

HTTP Method	URI	설명	요청/응답 예시 및 처리 흐름
POST	/employees	직원 생성	<b>Request:</b> {"name": "Kim", "department": "HR", "position": "Manager"}  <b>Response:</b> {"id": 10}  흐름: 필드 검증 후 INSERT 실행 및 생성된 ID 반환.
GET	/employees	직원 목록 조회	쿼리: GET /employees?department=HR&position=Manager  <b>Response:</b> [{"id": 7, "name": "Kim", "department": "HR", "position": "Manager"}]
GET	/employees/{id}	직원 상세 조회	
PUT	/employees/{id}	직원 수정	<b>Request:</b> {"department": "Finance", "position": "Director"}  제약: department와 position만 수정 가능. 이외 필드 수정 요청 시 에러 처리.
DELETE	/employees/{id}	직원 삭제	

### 3.2 Approval Request Service (REST + MongoDB + gRPC Client)

#### 3.2.1 MongoDB Document 구조

필드명	타입	설명
requestId	Number	자동 생성되는 결재 요청 ID
requesterId	Number	요청한 직원 ID
title	String	결재 제목
content	String	결재 내용
steps	Array of Object	결재자 순서 및 단계별 상태
steps[].step	Number	결재 단계 번호
steps[].approverId	Number	결재자 직원 ID
steps[].status	String	단계 상태 (pending, approved, rejected)
finalStatus	String	최종 상태 (in_progress, approved, rejected)
createdAt	DateTime	요청 생성 시간

**Document 예시:**

```
{
  "requestId": 1,
  "requesterId": 1,
  "title": "Expense Report",
  "content": "Travel expenses",
  "steps": [
    { "step": 1, "approverId": 3, "status": "approved", "updatedAt": "2025-01-01T11:23:11Z" },
    { "step": 2, "approverId": 7, "status": "approved", "updatedAt": "2025-01-01T12:23:11Z" }
  ],
  "finalStatus": "approved",
  "createdAt": "2025-01-01T10:23:11Z",
  "updatedAt": "2025-01-01T12:23:11Z"
}
```

### 3.2.2 REST API 상세

HTTP Method	URI	설명	요청/응답 예시 및 처리 흐름
POST	/approvals	결재 요청 생성	<b>Request:</b> { "requesterId": 1, "title": "Expense Report", "content": "Travel expenses", "steps": [ { "step": 1, "approverId": 3 }, { "step": 2, "approverId": 7 } ] }  <b>Response:</b> {"requestId": 1}
	<b>흐름:</b> 1. Employee Service REST 호출로 requesterId/approverId 존재 여부 검증. 2. steps가 1부터 오름차순인지 검증. 3. 각 steps에 "status": "pending" 추가. 4. MongoDB INSERT. 5. <b>gRPC</b> 를 통해 Approval Processing Service에 RequestApproval 호출.		
GET	/approvals	결재 요청 목록 조회	모든 결재 요청 목록 반환.
GET	/approvals/{requestId}	결재 요청 상세 조회	특정 requestId에 해당하는 Document 반환.
DELETE	-	결재 요청 삭제	기능 없음. 결재자가 <b>Rejected</b> 해야만 처리 종료 가능.

### 3.2.3 gRPC 프로토콜 정의 (proto/approval.proto)

```
syntax = "proto3";

package approval;

// Approval Processing Service와 통신하는 서비스
service Approval {
    // 결재 요청 정보를 Processing Service로 전달
    rpc RequestApproval(ApprovalRequest) returns (ApprovalResponse);
    // Processing Service로부터 결재 결과를 전달받음
    rpc ReturnApprovalResult(ApprovalResultRequest) returns (ApprovalResultResponse);
}

message Step {
    int32 step = 1;
    int32 approverId = 2;
    string status = 3; // pending, approved, rejected
}

message ApprovalRequest {
    int32 requestId = 1;
    int32 requesterId = 2;
    string title = 3;
    string content = 4;
    repeated Step steps = 5;
}

message ApprovalResponse {
    string status = 1; // "received" 등 처리 상태
}

message ApprovalResultRequest {
    int32 requestId = 1;
    int32 step = 2;
    int32 approverId = 3;
    string status = 4; // approved or rejected
}

message ApprovalResultResponse {
    string status = 1;
}
```

### 3.2.4 gRPC 기반 승인/반려 결과 처리 흐름 (Approval Request Service 역할)

Approval Processing Service로부터 ReturnApprovalResult 호출을 받으면:

1. 해당 requestId의 Document를 찾아 승인/반려 결과(status)를 업데이트 + updatedAt 추가
  - {"step": 1, "approverId": 3, "status": "approved", "updatedAt": "2025-01-01T10:23:11Z"}
2. Status가 "rejected"인 경우:
  - finalStatus를 "rejected"로 변경 + updatedAt 추가
  - Notification Service를 호출하여 requester에게 최종 반려 알림을 전송
3. Status가 "approved"인 경우:
  - 다음 **PENDING** 단계가 남아있는 경우:
    - 다음 approverId가 포함된 ApprovalRequest를 다시 구성하여 **gRPC**를 통해 Approval Processing Service에 RequestApproval을 재호출
  - 모든 단계가 완료된 경우:
    - finalStatus를 "approved"로 변경 + updatedAt 추가
    - Notification Service를 호출하여 requester에게 최종 승인 완료 알림을 전송

### 3.3 Approval Processing Service (REST + gRPC Server)

#### 3.3.1 역할 및 데이터 저장 구조

- **역할:** Approval Request Service로부터 gRPC 호출을 받아 결재 대기열을 관리하고, 결재자의 승인/반려 요청을 처리한 후, 그 결과를 다시 gRPC로 회신하는 게이트웨이 역할을 수행
- **저장소:** DB 없이 \*\*인메모리 자료구조(예: Map)\*\*를 사용하여 결재자 ID별 대기 목록을 저장

**In-Memory 자료구조 예시:**

```
{
  "7": [ // 결재자 ID (Approver ID)
    {
      "requestId": 1,
      "requesterId": 1,
      "title": "Expense Report",
      "content": "Travel expenses",
      "steps": [
        { "step": 1, "approverId": 3, "status": "approved" },
        { "step": 2, "approverId": 7, "status": "pending" }
      ]
    }
  ],
  // ... 다른 결재자 목록
}
```

#### 3.3.2 gRPC 서버 처리 흐름 (RequestApproval 호출 수신 시)

1. 수신된 steps 배열에서 상태가 첫 번째 **"pending"**에 해당하는 approverId를 찾음
2. 해당 approverId를 키로 하는 인메모리 대기 리스트에 수신된 결재 정보를 저장
3. ApprovalResponse로 {"status": "received"}를 반환

#### 3.3.3 REST API 상세

HTTP Method	URI	설명	요청/응답 예시 및 처리 흐름
GET	/process/{approverId}	결재자 대기 목록 조회	<b>Response:</b> 해당 approverId의 인메모리 대기 리스트 반환.
POST	/process/{approverId}/{requestId}	승인 또는 반려 처리	<b>Request:</b> {"status": "approved"} 또는 {"status": "rejected"}  <b>흐름:</b> 1. pending 목록에서 해당 결재 건을 찾아 제거. 2. <b>gRPC</b> 를 통해 Approval Request Service의 ReturnApprovalResult()를 호출하여 결과를 전달.

## 3.4 Notification Service (WebSocket)

### 3.4.1 동작 방식

- 연결: ws://[IP]:8080/ws?id={employeeId} 형태로 접속
- 사용자별 WebSocket 세션을 Map 형태로 인메모리에 저장하여 특정 직원에게만 메시지를 전송할 수 있도록 함

### 3.4.2 메시지 예시 (Approval Request Service에서 호출)

최종 승인 완료 알림:

```
{
  "requestId": 1,
  "result": "approved",
  "finalResult": "approved" // 최종 승인 완료
}
```

반려 알림 (첫 번째 반려 발생 시):

```
{
  "requestId": 1,
  "result": "rejected",
  "rejectedBy": 7, // 반려한 결재자 ID
  "finalResult": "rejected" // 최종 반려
}
```

## 4. 확장 및 심화 구현 과제

번호	과제명	배점	상세 내용
4.1	쿠버네티스 배포	10	각 서비스별 <b>Dockerfile</b> 작성 및 <b>K8s Manifest 파일(YAML)</b> 작성 (Deployment, Service, Ingress 등). 실행 방법 및 테스트 시나리오에 배포 내용 추가.
4.2	비동기 메시지 기반 통신 도입	10	현재 gRPC 동기 통신을 <b>Kafka, RabbitMQ 등 메시지 브로커</b> 로 대체. Approval Request Service와 Approval Processing Service 간 통신을 메시지 기반으로 변경하고, 비동기 통신의 이점을 보고서에 설명.
4.3	창의 영역	30	ERP 서비스 확장 관점에서 추가적으로 도입할 만한 기능을 제안하고 구현. (예: 권한 관리, 직원 출퇴근 관리, 연차 관리 등)

## 5. 제출물 및 보고서 가이드

### 5.1 제출물 구조 (학번/ 디렉토리)

```
학번/  
  employee-service/  
  approval-request-service/  
  approval-processing-service/  
  notification-service/  
  proto/  
    approval.proto  
  scripts/  
    init_mysql.sql  
k8s/ (4.1 확장 과제 선택 시)  
  <서비스명>-deployment.yaml  
...
```

학번.zip으로 압축하여 제출

### 5.2 보고서 구성 요소

1. 전체 아키텍처 다이어그램 및 설명
2. 서비스 간 호출 흐름도 (REST, gRPC, WebSocket 명시)
3. REST API 표 정리 (모든 서비스 통합)
4. gRPC proto 파일 내용
5. MySQL 스키마 설명
6. MongoDB 문서 구조 설명
7. WebSocket 메시지 구조
8. 실행 방법 (빌드 및 실행 명령어, 환경 설정 포함)
9. 테스트 시나리오 (결재 승인, 반려, 동시성 등) → 많은 시나리오로 자세히 검증할 수록 높은 점수
10. 실행 화면 스크린샷 → 많은 스크린샷과 이에 대한 설명이 있을 수록 높은 점수
11. 개발 중 문제와 해결 방법 (통신, 자료구조, 저장소 연동 등) → 본인이 설치하지 못하거나 세팅하지 못했던 것은 SoSo

### 5.3 제출일

12월 11일 수업 시간 전(13:29까지)