

TEAM SOCCER



프로젝트 소개

지역 축구팀의 홍보와 선수 모집을 돋는 온라인 플랫폼,
TEAM SOCCER!

지역 축구팀이 더 많은 선수를 모집하고,
쉽게 팀을 홍보할 수 있도록 돋는 것을 목표로 하고 있습니다.



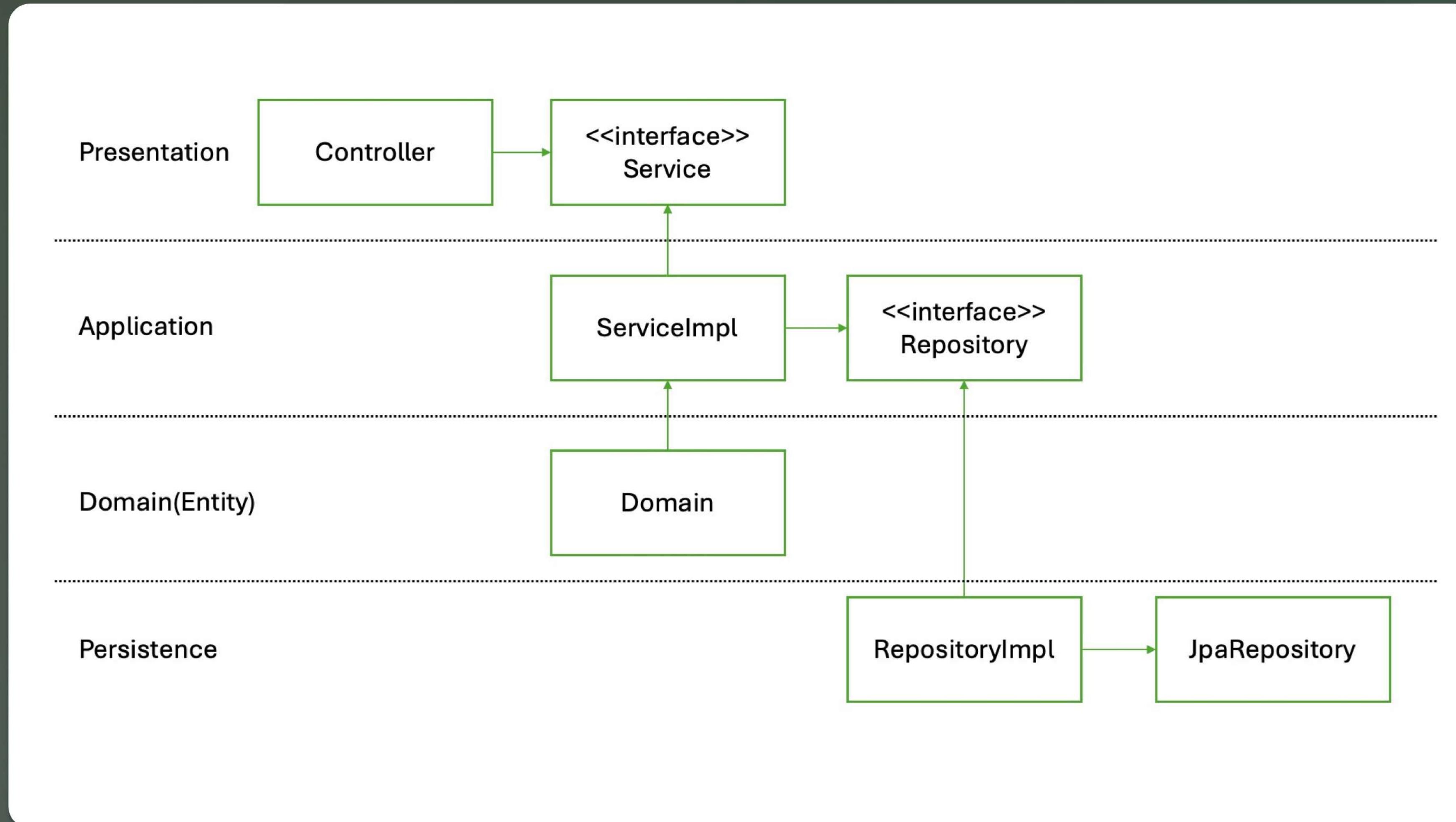
프로젝트 소개

주요 기능

- CRUD를 활용한 팀과 선수 소개 및 게시물 작성
- JWT 인증 기반의 사용자 로그인 및 권한 관리
- 검색
- Docker, Git Action을 활용한 CICD

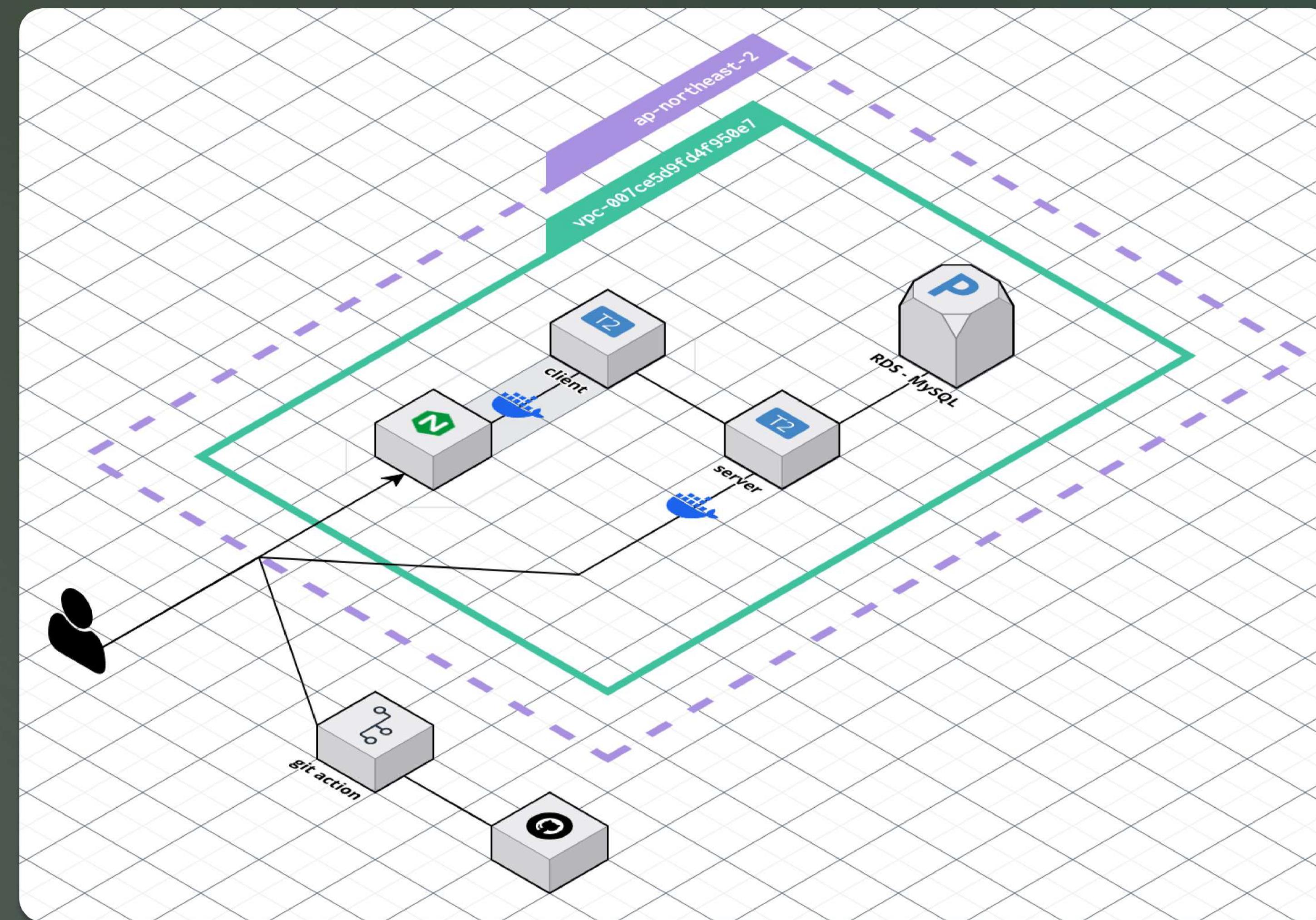


프로젝트 서버 코드 아키텍처





프로젝트 서버 인프라 아키텍처





Git 협업

- Branch 생성, Pull Request 작성

- 코드리뷰
- commit 컨벤션



Git

fix: 게시물 디테일 게시물 소유자 여부 확인

fix: delete error message

feat: 입단 신청 update

ywonchae1 requested changes 2 days ago

ywonchae1 left a comment

수고하셨습니다 !! 수정 조건만 하면 완벽할 것 같네요 🤘

src/main/java/soccerTeam/enroll/dto/EnrollUpdateRequest.java Outdated

Comment on lines 12 to 13

```
12 +     @Schema(description = "팀 ID", example = "1", requiredMode = Schema.RequiredMode.REQUIRED)
13 +     Long teamId,
```

ywonchae1 2 days ago

수정할 때 팀 id까지 필요한가요? enroll id만 있으면 수정 가능할 것 같아요

Resolve conversation

src/main/java/soccerTeam/enroll/dto/EnrollUpdateRequest.java

src/main/java/soccerTeam/enroll/repository/EnrollRepositoryImpl.java Outdated



Swagger

- 읽기 좋은 문서 제공
- API설계를 일관적으로 할 수 있도록 도움

스프링부트 게시판 REST API v1.0 OAS3

</v3/api-docs>

스프링부트 기반의 게시판 REST API 서비스

Apache 2.0

Servers

입단 신청서 API

입단 신청서 API 관련 작업

PUT [/api/enroll](#) 입단 신청서 수정

POST [/api/enroll](#) 입단 신청서 생성

GET [/api/enroll/{enrollId}](#) 입단 신청서 상세 조회



응답 규격화

```

@JsonPropertyOrder({"status", "code", "message", "data"})
public record ApiResponse<T>{
    int status,
    String code,
    String message,
    @JsonInclude(JsonInclude.Include.NON_NULL) T data) {
    }

    public static ApiResponse<?> success(SuccessType successType) {
        return new ApiResponse<>(status: 200, successType.getCode(), successType.getMessage(), data: null);
    }

    public static <T> ApiResponse<T> success(SuccessType successType, T data) {
        return new ApiResponse<>(status: 200, successType.getCode(), successType.getMessage(), data);
    }

    public static ApiResponse<?> fail(BaseException exception) {
        ErrorType errorType = exception.getErrorType();
    }
}

```

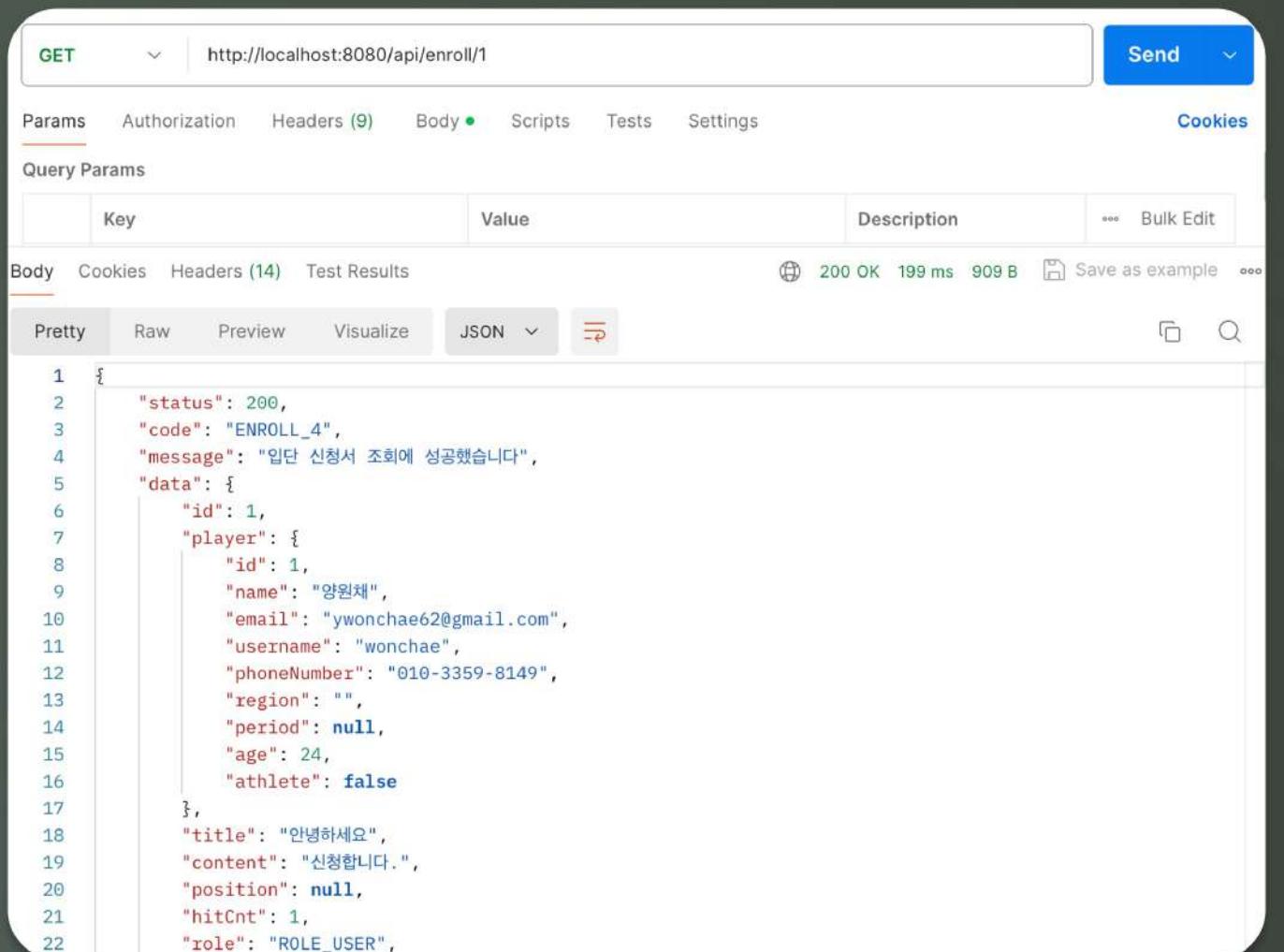
```

public enum EnrollSuccessType implements SuccessType {
    CREATE_SUCCESS(code: "ENROLL_1", message: "팀 가입 신청에 성공하였습니다"),
    LIST_SUCCESS(code: "ENROLL_2", message: "입단 신청서 목록 조회에 성공했습니다"),
    UPDATE_SUCCESS(code: "ENROLL_3", message: "입단 신청 수정이 완료되었습니다."),
    ENROLL_SUCCESS(code: "ENROLL_4", message: "입단 신청서 조회에 성공했습니다"),
    DELETE_SUCCESS(code: "ENROLL_5", message: "입단 신청서 삭제에 성공했습니다");

    private final String code;
    private final String message;

    EnrollSuccessType(String code, String message) {
        this.code = code;
        this.message = message;
    }
}

```



```

@RestController
@RequestMapping("/api/enroll")
@RequiredArgsConstructor
@Tag(name = "입단 신청서 API", description = "입단 신청서 API 관련 작업")
public class EnrollController {

    private final EnrollService enrollService;

    @Operation(summary = "입단 신청서 생성", description = "새로운 신청서를 생성합니다.")
    @PostMapping
    public ApiResponse<EnrollCreateResponse> create(
        @LoginMember String username,
        @Valid @RequestBody EnrollCreateRequest enrollCreateRequest) {
        EnrollCreateResponse response = enrollService.create(username, enrollCreateRequest);
        return ApiResponse.success(EnrollSuccessType.CREATE_SUCCESS, response);
    }
}

```



GlobalExceptionAdvice

```
public class BaseException extends RuntimeException {  
    private final ErrorType errorType;  
    private final HttpStatus httpStatus;  
  
    ▲ ywonchae1  
    public BaseException(ErrorType errorType, HttpStatus httpStatus) {  
        super(errorType.getMessage());  
        this.errorType = errorType;  
        this.httpStatus = httpStatus;  
    }  
  
    ▲ ywonchae1  
    public HttpStatus getHttpStatus() {  
        return this.httpStatus;  
    }  
}
```

```
@Slf4j  
@RestControllerAdvice  
public class GlobalExceptionAdvice {  
    ▲ ywonchae1  
    @ExceptionHandler(MissingServletRequestParameterException.class)  
    public ResponseEntity<ApiResponse<?>> handleMissingServletRequestParameterException(  
        final MissingServletRequestParameterException ex) {  
        String param = ex.getParameterName();  
        Map<String, String> body = new HashMap<>();  
        body.put("param", param);  
        return new ResponseEntity<ApiResponse<?>>(  
            ApiResponse.fail(CommonErrorResponseType.MISSING_PARAM, httpCode: 400, body), HttpStatus.BAD_REQUEST);  
    }  
  
    ▲ ywonchae1  
    @ExceptionHandler(MethodArgumentNotValidException.class)  
    public ResponseEntity<ApiResponse<?>> handleMethodArgumentNotValidException(  
        final MethodArgumentNotValidException ex) {  
        return new ResponseEntity<ApiResponse<?>>(  
            ApiResponse.fail(CommonErrorResponseType.REQUEST_VALIDATION, httpCode: 400),  
            HttpStatus.BAD_REQUEST);  
    }  
}
```

```
exception  
    ⚡ BadRequestException  
    ⚡ BaseException  
    ⚡ ForbiddenException  
    ⚡ InternalServerException  
    ⚡ NotFoundException  
    ⚡ UnauthorizedException
```

```
/** CUSTOM */  
▲ ywonchae1  
@ExceptionHandler(BaseException.class)  
public ResponseEntity<ApiResponse<?>> handleCustomException(BaseException ex) {  
    log.error(ex.getMessage());  
    return new ResponseEntity<ApiResponse<?>>(ApiResponse.fail(ex), ex.getHttpStatus());  
}
```



Java의 스트림 API와 람다 표현식

- 스트림, 람다를 통해 읽기 좋은 코드 작성
- 대규모 데이터 처리 시 매우 유용
- 향후 프로젝트에서도 적용할 예정

```
@Override
public List<SoccerTeamListResponseDto> selectSoccerTeamList() {
    return soccerTeamRepository.findAll().stream() Stream<SoccerTeamEntity>
        .map(SoccerTeamEntity::toDto) Stream<SoccerTeamListResponseDto>
        .toList();
}
```



Validation Check

- **@NotBlank**: 필드가 null이거나 빈 문자열인 경우 유효성 검사를 실패합니다.
- **@Pattern**: 특정 정규 표현식과 일치하는지 확인합니다.
- **@Email**: 유효한 이메일 형식인지 검증합니다.
- **@Size**: 문자열의 길이를 제한합니다.
- **@Min, @Max**: 숫자의 최소 및 최대 값을 제한합니다.

```
@Schema(description = "회원가입 요청 객체")
public class JoinDto {

    @NotBlank(message = "이름을 입력해주세요.")
    @Size(min = 3, max = 17, message = "이름은 실명으로 입력해주세요.")
    @Schema(description = "사용자의 이름", example = "홍길동", required = true)
    private String name;

    @NotBlank(message = "이메일을 입력해주세요.")
    @Email(message = "이메일 형식으로 입력해주세요.")
    @Schema(description = "사용자의 이메일", example = "test@test.com", required = true)
    private String email;

    @NotBlank(message = "ID를 입력해주세요.")
    @Schema(description = "사용자의 아이디", example = "testid", required = true)
    private String username;
```



Flyway

- 데이터베이스 스키마의 버전을 관리
- マイグレーション 자동화



Flyway

```
1 • USE teamsoccerdb;
2 • select * from flyway_schema_history
3
```

Result Grid | Filter Rows: [] | Edit: [] | Export/Import: [] | Wrap Cell Content: []

installed_rank	version	description	type	script	checksum	installed_by	installed_on	execution_time	success
1	1	<< Flyway Baseline >>	BASELINE	<< Flyway Baseline >>	NULL	root	2024-08-27 00:16:47	0	1
2	2	Add testcolumn	SQL	V2__Add_testcolumn.sql	185527866	root	2024-08-27 00:21:28	22	1
3	3	Drop testcolumn	SQL	V3__Drop_testcolumn.sql	-938467939	root	2024-08-27 00:36:59	34	1
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

```
spring:
  flyway:
    enabled: true
    locations: classpath:db/migration
    baseline-on-migrate: true
    baseline-version: 1
```

```
src
  main
    generated
    java
  resources
    db.migration
      V1_InitTables.sql
```



Kafka

- 단순한 메시지 브로커를 넘어 실시간 데이터 스트리밍, 로그 수집 등 다양한 분야에서 활용될 수 있는 도구
- 구성요소를 이해하고 설정하는 데 어려움이 많아 완성하지 못함
- 메시지 브로커 수준의 코드 구현을 위해 노력함





Docker, DockerCompose, Git Action

```
FROM openjdk:21
COPY build/libs/teamsoccer-server.jar teamsoccer-server.jar
ENTRYPOINT ["java", "-jar", "/teamsoccer-server.jar", "--spring.profiles.active=prod"]
```

```
version: "3.8"
```

```
services:
  backend:
    image: ${IMAGE_FULL_URL}
    container_name: ${DOCKERHUB_IMAGE_NAME}
    restart: always
    environment:
      - TZ=Asia/Seoul
    ports:
      - '8080:8080'
    env_file: .env
```

Repository secrets		New repository secret
Name ↗↑	Last updated	
APPLICATION_SECRETS	7 minutes ago	🔗
DOCKERHUB_ACCESS_TOKEN	9 minutes ago	🔗
EC2_HOST	1 minute ago	🔗
EC2_PRIVATE_KEY	now	🔗
EC2_USERNAME	1 minute ago	🔗

```
# Docker 이미지 빌드 및 도커허브로 푸시
- name: Docker Build and Push
  uses: docker/build-push-action@v6.0.1
  with:
    file: scripts/Dockerfile
    context: .
    platforms: linux/amd64, linux/arm64
    push: true
    tags: ${steps.metadata.outputs.tags}
```

```
# 환경변수 적용
- name: Set environment variables
  run:
    - echo "${secrets.APPLICATION_SECRETS}" >> .env
```

```
# 서버로 .env 파일 전송
- name: Copy .env file to EC2
  uses: burnett01/rsync-deployments@7.0.1
  with:
    switches: -avzr --delete
    remote_host: ${secrets.EC2_HOST}
    remote_user: ${secrets.EC2_USERNAME}
    remote_key: ${secrets.EC2_PRIVATE_KEY}
    path: .env
    remote_path: /home/${secrets.EC2_USERNAME}/teamsoccer-app/
```

```
# EC2로 배포
- name: Deploy to EC2 Server
  uses: appleboy/ssh-action@master
  env:
    IMAGE_FULL_URL: ${steps.metadata.outputs.tags}
  with:
    host: ${secrets.EC2_HOST}
    username: ${secrets.EC2_USERNAME}
    key: ${secrets.EC2_PRIVATE_KEY}
    envs: IMAGE_FULL_URL, DOCKERHUB_IMAGE_NAME # docker-compose.yml에서 사용할 환경 변수
    script:
      - cd teamsoccer-app/
      - echo "${secrets.DOCKERHUB_ACCESS_TOKEN}" | docker login -u "${env.DOCKERHUB_USERNAME}" --password-stdin
      - docker compose up --build -d
      - docker image prune -a -f
```



Git Action

main ▾ 3 Branches 0 Tags Go to file

ywonchae1 Merge pull request #32 from TeamSoccer/feat-gitaction-cicd

.github/workflows fix: main 브랜치 병합 시 cicd 동작하도록 수

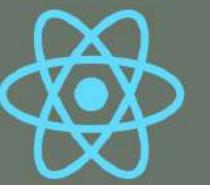
gradle/wrapper BackEnd 업로드

scripts feat: git action cicd 파이프라인 구축

src/main Merge pull request #30 from TeamSoc



Kubernetes



- Deployment
- Service
(LoadBalancer)

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: client-deploy
spec:
  replicas: 3
  selector:
    matchLabels:
      app: client
  template:
    metadata:
      labels:
        app: client
    spec:
      containers:
        - name: client
          image: ywonchae1/teamsoccer-client
          ports:
            - containerPort: 80
          envFrom:
            - configMapRef:
                name: client-configmap

---
apiVersion: v1
kind: Service
metadata:
  name: client
spec:
  type: LoadBalancer
  ports:
    - port: 80
      # targetPort: 3000
  selector:
    app: client
```



- Deployment
- Service
(LoadBalancer)
- ConfigMap
- Secret

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: server-deploy
spec:
  replicas: 3
  selector:
    matchLabels:
      app: server
  template:
    metadata:
      labels:
        app: server
    spec:
      containers:
        - name: server
          image: ywonchae1/teamsoccer-server
          ports:
            - containerPort: 8080
          envFrom:
            - configMapRef:
                name: server-configmap
            - secretRef:
                name: server-credentials

---
apiVersion: v1
kind: Service
metadata:
  name: server
  selector:
    app: server
```



Kubernetes



ConfigMap

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: server-configmap
data:
  MYSQL_URL: jdbc:mysql://database-1.cxd...
  MYSQL_USERNAME: teamsoccer
  FILE_LOCATION: /files
  SERVER_PORT: "8080"
```



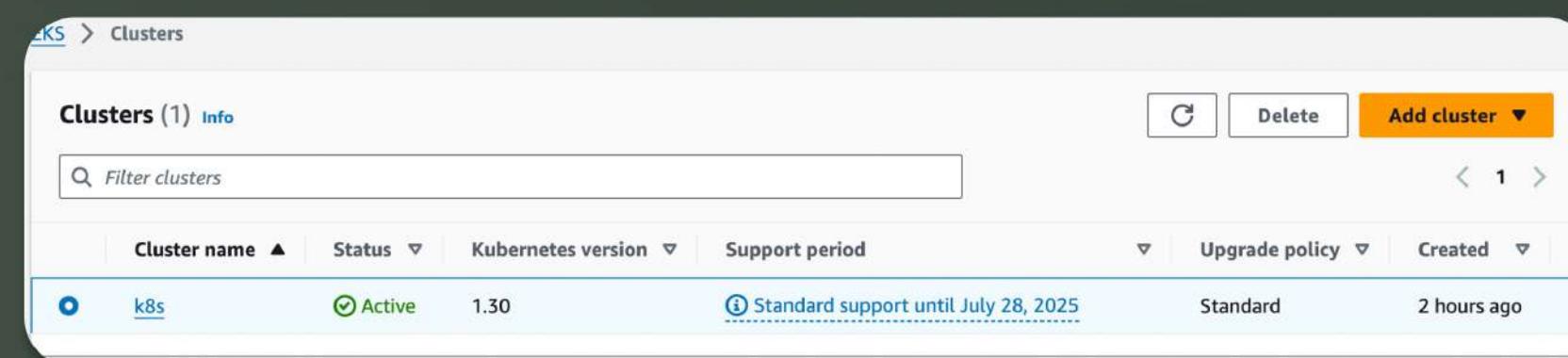
Secret

```
apiVersion: v1
kind: Secret
metadata:
  name: server-credentials
data:
  MYSQL_PASSWORD: c29jY2VydG...
  JWT_SECRET: uVbX3RFJ8Fkj2+
```



Kubernetes

쿠버네티스 클러스터에서 Pod가 적절히 스케줄링 되는 것을 확인
서비스 도메인이 없고, EKS의 LoadBalancerIP 설정 과정이 까다로워 실 배포에 적용하지 못함



```
1chae@Lionel FrontEnd % kubectl get all
NAME                               READY   STATUS    RESTARTS   AGE
pod/client-deploy-776b864554-26llw 1/1     Running   0          80s
pod/client-deploy-776b864554-ht92j 1/1     Running   0          80s
pod/client-deploy-776b864554-ndc7m  1/1     Running   0          80s
pod/server-deploy-74949c9f56-c5hbd 1/1     Running   0          2m38s
pod/server-deploy-74949c9f56-npt8r 1/1     Running   0          2m38s
pod/server-deploy-74949c9f56-pvfd  r 1/1     Running   0          2m38s

NAME          TYPE      CLUSTER-IP      EXTERNAL-IP
service/client LoadBalancer  10.100.200.234  aaba0dddcc445b4173a5bd4f0dac4885f-1818151442.ap-northeast-2.elb.amazonaws.com
service/kubernetes ClusterIP  10.100.0.1    <none>
service/server  LoadBalancer  10.100.194.183 a5c090431880243cba06d09ff509e1a6-1120076585.ap-northeast-2.elb.amazonaws.com

NAME          READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/client-deploy  3/3     3           3           80s
deployment.apps/server-deploy  3/3     3           3           2m38s

NAME          DESIRED   CURRENT   READY   AGE
replicaset.apps/client-deploy-776b864554  3        3        3        80s
replicaset.apps/server-deploy-74949c9f56  3        3        3        2m38s
```

```
1chae@Lionel FrontEnd % kubectl exec client-deploy-776b864554-26llw -- curl http://server:8080/api/soccerTeam
% Total    % Received % Xferd  Average Speed   Time   Time     Time Current
          Dload  Upload Total Spent   Left Speed
100  102    0  102    0      0  482      0  --:--:--  --:--:-- 481
{"status":200,"code":"SOCCER_TEAM_1","message":"팀 목록 조회에 성공하였습니다","data":[]}%
```



양원재

K8s는 GCP에서만 사용을 해봤는데, 이번에 AWS의 EKS를 사용해 볼 수 있어서 좋은 경험이었습니다.

EKS에서는 LoadBalancerIP 적용 방식이 까다로워 시간 관계상 완료하지 못했지만, 꼭 다시 도전해 보고 싶습니다.



정우석

처음 경험하는 것들이 많아 초기에는 어려움이 많았고, 시간이 많이 소요되었습니다.

그럼에도 불구하고 팀원들과의 협업을 통해 많은 성장을 이뤘으며, 팀 프로젝트의 중요성을 깨달았습니다.

다만, 이러한 이유로 후반부에 Kafka나 Kubernetes 같은 배포 기술을 학습하고 적용할 시간이 부족했던 점은 아쉬움으로 남습니다.



김민재

github를 통해 협업하는 일, 백엔드 코드 작업을 할 때에서의 협업도 처음이라 많이 해맸습니다.

하지만 팀원의 도움으로 github의 branch 관리나 pull request, commit message 등을 배워 사용했습니다.

아쉬운 점으로는 팀 협업을 처음 경험하다보니 프로젝트 진행하는 동안 스스로의 생각보다 진도가 굉장히 더디게 나갔다는 겁니다.

이부분은 팀원의 도움으로 해결됐지만 아쉬움을 많이 느꼈습니다.



강민수

이번 프로젝트를 진행하면서 부족한게 많고, 협업도 처음이라 시행착오가 많았지만,

팀원들과의 협업을 통해 발생한 문제들을 함께 해결하며, 각자의 강점을 발휘하고 서로 배울 수 있어 좋았습니다.

프로젝트가 마무리될 때쯤, 완료하지 못한 작업들이 있어 아쉬움이 남았지만 많은 기술적 성장을 느낄 수 있었고, 앞으로의 개발에 큰 자신감을 가지게 되었습니다.