

# 체육인재개발원 교육관리 시스템 백엔드 프로그램 설계서

FASTAPI

SQLALCHEMY

POSTGRESQL

PYDANTIC

# 1. 시스템 개요

본 문서는 체육인재개발원 교육관리 시스템의 백엔드 설계에 관한 상세 내용을 담고 있습니다. 본 시스템은 체육 인재 개발을 위한 교육 과정을 효율적으로 관리하고, 사용자들에게 편리한 교육 서비스를 제공하기 위한 플랫폼입니다.

## 1.1 주요 기능



### 사용자 인증 및 권한 관리

JWT 기반 안전한 사용자 인증과 역할 기반 권한 관리 시스템



### 교육과정 관리 및 수강신청

교육 과정 등록, 수정, 조회 및 수강생 신청 관리



### 교육 이수 관리

교육 이수 처리 및 이수증 발급 관리



## 설문조사 시스템

교육 만족도 및 품질 평가를 위한 설문조사 기능



## 교육장 시설 관리 및 예약

교육 시설 관리 및 실시간 예약 시스템



## 통계 데이터 조회

교육 과정 통계 및 분석 대시보드 제공



## 외부 시스템 연동

체육 관련 외부 시스템과의 API 연동 인터페이스



## 데이터베이스 관리



## 2. 시스템 아키텍처

---

본 시스템은 FastAPI 프레임워크를 기반으로 구축되었으며, 클린 아키텍처 패턴을 적용하여 유지보수성과 확장성을 높였습니다.

## 2.1 아키텍처 구조

```
backend/
├── app/
│   ├── main.py                # FastAPI 앱 실행 엔트리포인트
│   ├── core/
│   │   ├── config.py          # 앱 설정, 보안 관련
│   │   └── security.py         # 환경변수, 설정값 로딩
│   │                               # 인증/인가, JWT 처리
│   ├── api/
│   │   └── # API 라우터
│   │       ├── v1/
│   │       │   ├── user.py    # 회원가입, 로그인 등
│   │       │   ├── education.py # 수강신청, 이수관리
│   │       │   ├── survey.py  # 설문조사 관련
│   │       │   ├── statistics.py # 통계 조회 API
│   │       │   ├── facility.py # 교육장 관리
│   │       │   ├── portal.py  # 체육인재개발원 포털
│   │       │   ├── integration.py # 외부 시스템 연동
│   │       └── admin.py       # 게시판, 권한 등 공통 관리
│   ├── models/
│   │   ├── # SQLAlchemy 모델 정의
│   │   ├── user.py           # User, Role, Permission 모델
│   │   └── education.py      # Course, Enrollment, Completion
│   └── 등
├── schemas/
│   └── # Pydantic 스키마
│       (Request/Response DTO)
│       ├── user.py          # LoginRequest, UserCreate 등
│       └── education.py     # CourseCreate, CourseResponse 등
├── services/
│   ├── # 비즈니스 로직 분리
│   ├── user_service.py     # 회원 가입/수정 처리 로직
│   └── education_service.py # 수강신청, 수료 처리 로직
├── crud/
│   ├── # DB 접근 레이어 (Repository)
│   ├── user_crud.py        # 사용자 관련 DB 로직
│   └── education_crud.py   # 교육과정 관련 DB 로직
├── utils/
│   ├── # 공통 유틸리티
│   └── file.py             # 파일 업로드, 저장 등
```

```
| | └─ notification.py          # SMS/메일 발송 기능
|
| └─ database.py                # DB 세션, 엔진 초기화
|
└─ tests/                      # 테스트 코드
    └─ test_user.py            # 회원가입/로그인 테스트
    └─ test_education.py       # 수강신청/수료 테스트
```

## 2.2 계층 구조

본 시스템은 다음과 같은 계층으로 구성되어 있습니다:

- **API 계층:** HTTP 요청을 처리하고 클라이언트와 통신
- **서비스 계층:** 비즈니스 로직을 담당
- **CRUD 계층:** 데이터베이스와의 통신을 담당
- **모델 계층:** 데이터베이스 테이블과 매핑되는 객체 정의
- **스키마 계층:** API 요청/응답 데이터 구조 정의

### 3. 기술 스택

구분	기술	버전	용도
프레임워크	FastAPI	0.109.0 이상	API 서버 구현
WSGI 서버	Uvicorn	0.27.0 이상	ASGI 서버 실행
데이터 검증	Pydantic	2.6.0 이상	데이터 검증 및 직렬화
ORM	SQLAlchemy	2.0.25 이상	데이터베이스 접근
인증	python-jose, PyJWT	3.3.0, 2.8.0 이상	JWT 기반 인증 처리
암호화	passlib[bcrypt]	1.7.4 이상	패스워드 해싱
환경 변수	python-dotenv	1.0.0 이상	환경변수 관리
폼 처리	python-multipart	0.0.6 이상	폼 데이터 처리
이메일 검증	email-validator	2.1.0 이상	이메일 주소 검증



## 4. 데이터베이스 설계

---

본 시스템은 SQLAlchemy ORM을 사용하여 PostgreSQL 데이터베이스와 상호작용합니다. 데이터베이스 관리를 위한 별도의 `database` 모듈을 구현하여 확장성과 유지보수성을 높였습니다.

### 4.1 데이터베이스 구조

```
database/
|
├── __init__.py          # 패키지 초기화 및 주요 함수 export
├── database.py          # 데이터베이스 연결 설정, 기본 함수 제공
├── db_manager.py        # DB 관리 인터페이스 - 중앙 관리 클래스
└── db_engine.py         # 데이터베이스 엔진 구현체
```

## 4.2 주요 엔티티

엔티티	설명	주요 속성
User	시스템 사용자	id, email, password, name, role, created_at, updated_at
Role	사용자 역할	id, name, permissions
Permission	권한	id, name, description
Course	교육 과정	id, title, description, start_date, end_date, capacity, status
Enrollment	수강 신청	id, user_id, course_id, status, created_at
Completion	교육 이수	id, enrollment_id, completion_date, score, certificate_id
Facility	교육장 시설	id, name, type, description, location, capacity, status
Reservation	시설 예약	id, facility_id, user_id, purpose, start_time, end_time, status
Survey	설문조사	id, title, description, start_date, end_date, status
Question	설문 문항	id, survey_id, content, type, order
Response	설문 응답	id, survey_id, user_id, question_id, answer, created_at

## 4.3 데이터베이스 연결 설정

PostgreSQL 데이터베이스 연결은 `database.py` 파일에서 관리하며, 팀별 데이터베이스 분리 및 세션 관리 기능을 제공합니다.

```

# PostgreSQL 연결 정보
POSTGRES_SERVER = os.getenv("POSTGRES_SERVER", "skcc-tools-aihack25-
postgres.postgres.database.azure.com")
POSTGRES_USER = os.getenv("POSTGRES_USER", "postgresuser15")
POSTGRES_PASSWORD = os.getenv("POSTGRES_PASSWORD", "postgresuser15")
POSTGRES_DB = os.getenv("POSTGRES_DB", "postgresdb15")

# 기본 데이터베이스 URI
DEFAULT_DATABASE_URI = f"postgresql://{POSTGRES_USER}:
{POSTGRES_PASSWORD}@{POSTGRES_SERVER}/{POSTGRES_DB}"

# 팀별 데이터베이스 URI 매핑
team_database_uris: Dict[str, str] = {}

def get_database_uri(team_name: str = None) → str:
    """
    팀 이름에 따른 데이터베이스 URI를 반환합니다.
    팀 이름이 주어지지 않으면 기본 데이터베이스 URI를 반환합니다.
    """
    if team_name and team_name in team_database_uris:
        return team_database_uris[team_name]
    return DEFAULT_DATABASE_URI

# 기본 엔진 생성
engine = create_engine(DEFAULT_DATABASE_URI)
SessionLocal = sessionmaker(autocommit=False, autoflush=False,
bind=engine)

Base = declarative_base()

# Dependency
def get_db():
    db = SessionLocal()
    try:
        yield db
    finally:
        db.close()

# 팀별 엔진 및 세션 생성 함수
def get_team_engine(team_name: str):
    """팀 이름에 해당하는 엔진을 반환합니다."""
    uri = get_database_uri(team_name)
    return create_engine(uri)







```

```
def get_team_session(team_name: str):
    """팀 이름에 해당하는 세션을 생성합니다."""
    team_engine = get_team_engine(team_name)
    team_session = sessionmaker(autocommit=False, autoflush=False,
                                bind=team_engine)
    return team_session
```

## 4.4 데이터베이스 관리 인터페이스

시스템에서는 `DBManager` 클래스를 통해 데이터베이스 관리 기능을 중앙화하여 제공합니다. 이 클래스는 팀별 데이터베이스 생성, 백업/복구, 품질 관리, 데이터 이관 등의 기능을 담당합니다.

### 4.4.1 주요 기능

-  **팀별 데이터베이스 관리**: 팀별 격리된 데이터베이스 생성 및 관리
-  **백업 및 복구**: 주기적인 데이터베이스 백업 및 필요시 복구
-  **데이터 품질 관리**: Null 값, 중복값 검사 등 데이터 품질 진단
-  **데이터 이관**: 서로 다른 데이터베이스 간 데이터 이관 기능
-  **개인정보 보호**: 개인정보 비식별화 처리 기능
-  **성능 진단**: 데이터베이스 성능 모니터링 및 진단

### 4.4.2 DBManager 구현

```

class DBManager:
    """데이터베이스 관리 인터페이스 클래스"""

    def __init__(self):
        self.team_db_engine = team_db_engine
        self.backup_restore = db_backup_restore
        self.quality_manager = data_quality_manager
        self.migration_manager = data_migration_manager

    def create_team_database(self, team_name: str) → bool:
        """팀별 데이터베이스 생성"""
        return self.team_db_engine.create_team_database(team_name)

    def get_team_database_info(self, team_name: str) → Dict[str, Any]:
        """팀 데이터베이스 정보 조회"""
        return self.team_db_engine.get_team_database_info(team_name)

    def backup_database(self, team_name: str = None, backup_type: str =
"full") → str:
        """데이터베이스 백업"""
        return self.backup_restore.backup_database(team_name,
backup_type)

    def restore_database(self, backup_file: str, team_name: str = None) -
> bool:
        """데이터베이스 복구"""
        return self.backup_restore.restore_database(backup_file,
team_name)

    # 데이터 품질 관리 기능
    def check_data_quality(self, team_name: str, table_name: str) →
Dict[str, Any]:
        """데이터 품질 검사"""
        engine = get_team_engine(team_name)

        # NULL 값 및 중복값 검사 등 품질 진단 수행
        # ...

    # 개인정보 비식별화 처리
    def anonymize_personal_data(self, team_name: str, table_name: str,
                                columns: Dict[str, str]) → Dict[str, Any]:
        """
        개인정보 비식별화 처리
        - 방법: "mask" (일부 마스킹), "encrypt" (암호화), "replace" (가상데이터로

```

대체)

```
"""
    engine = get_team_engine(team_name)
    return self.quality_manager.anonymize_personal_data(engine,
table_name, columns)
```

## 4.5 데이터베이스 사용 예시

FastAPI 애플리케이션에서는 dependency injection을 통해 데이터베이스 세션을 주입받아 사용합니다:

```
from fastapi import Depends, FastAPI
from sqlalchemy.orm import Session
from database import get_db, db_manager

app = FastAPI()

@app.get("/users/")
def read_users(db: Session = Depends(get_db)):
    # db 세션을 사용하여 사용자 정보 조회
    users = db.query(User).all()
    return users

@app.get("/team/{team_name}/database-info")
def get_team_db_info(team_name: str):
    # DBManager를 통한 팀별 데이터베이스 정보 조회
    return db_manager.get_team_database_info(team_name)
```

## 5. API 설계

---

본 시스템의 API는 RESTful 원칙을 따르며, 다음과 같은 주요 엔드포인트로 구성됩니다.

### 5.1 사용자 관리 API

**POST** /api/v1/users/register - 사용자 회원가입

**POST** /api/v1/auth/login - 로그인 및 토큰 발급

**GET** /api/v1/users/me - 현재 사용자 정보 조회

**PUT** /api/v1/users/me - 사용자 정보 수정

**POST** /api/v1/auth/refresh - 토큰 갱신

### 5.2 교육과정 관리 API

**GET** /api/v1/courses - 교육과정 목록 조회



**GET**

/api/v1/courses/{course\_id} - 교육과정 상세 조회

**POST**

/api/v1/courses - 교육과정 생성

**PUT**

/api/v1/courses/{course\_id} - 교육과정 수정

**DELETE**

/api/v1/courses/{course\_id} - 교육과정 삭제

## 5.3 수강신청 및 이수 API

**POST**

/api/v1/enrollments - 수강신청

**GET**

/api/v1/users/me/enrollments - 내 수강신청 목록

**PUT**

/api/v1/enrollments/{enrollment\_id}/status - 수강신청 상태 변경

**POST**

/api/v1/enrollments/{enrollment\_id}/complete - 교육 이수 처리

**GET**

/api/v1/users/me/completions - 내 이수 내역 조회

## 5.4 시설 관리 및 예약 API

**GET**

/api/v1/facilities - 시설 목록 조회

**GET**

/api/v1/facilities/{facility\_id} - 시설 상세 조회

**POST**

/api/v1/admin/facilities - 시설 생성 (관리자용)

**PUT**

/api/v1/admin/facilities/{facility\_id} - 시설 수정 (관리자용)

**GET**

/api/v1/facilities/{facility\_id}/availability - 시설 가용성 조회

**POST**

/api/v1/reservations - 예약 생성

**GET**

/api/v1/users/{user\_id}/reservations - 사용자 예약 목록 조회

**POST**

/api/v1/reservations/{reservation\_id}/cancel - 예약 취소

## 5.5 설문조사 API

**GET**

/api/v1/surveys - 설문조사 목록 조회

**GET**

/api/v1/surveys/{survey\_id} - 설문조사 상세 조회

**POST**

/api/v1/surveys/{survey\_id}/responses - 설문 응답 제출

**GET**

/api/v1/surveys/{survey\_id}/results - 설문 결과 조회 (관리자용)

## 5.6 통계 API

**GET**

/api/v1/statistics/courses - 교육과정 관련 통계

**GET**

/api/v1/statistics/enrollments - 수강신청 관련 통계

**GET**

/api/v1/statistics/facilities - 시설 이용 관련 통계

**GET**

/api/v1/statistics/surveys - 설문조사 관련 통계

## 6. 인증 및 권한 관리

---

본 시스템은 JWT(JSON Web Token) 기반의 인증 시스템을 사용합니다.

### 6.1 인증 프로세스

1. 사용자가 이메일과 패스워드로 로그인 요청
2. 서버에서 자격 증명 검증 후, 액세스 토큰과 리프레시 토큰 발급
3. 이후 API 요청 시 액세스 토큰을 Authorization 헤더에 포함하여 전송
4. 액세스 토큰 만료 시 리프레시 토큰을 사용하여 새 토큰 발급

### 6.2 권한 관리

사용자 권한은 다음과 같은 역할로 구분됩니다:

- **ADMIN**: 시스템 관리자. 모든 기능에 접근 가능
- **MANAGER**: 교육 관리자. 교육과정, 시설 관리 권한
- **INSTRUCTOR**: 강사. 자신의 교육과정 관리 권한
- **STUDENT**: 교육생. 수강신청, 설문 참여 등 기본 권한
- **GUEST**: 미인증 사용자. 제한된 정보 조회만 가능

### 6.3 인증 관련 코드 구현

인증 관련 코드는 `core/security.py` 파일에 구현되어 있으며, 주요 함수는 다음과 같습니다:

- `get_password_hash(password: str) → str` : 비밀번호 해싱
- `verify_password(plain_password: str, hashed_password: str) → bool` : 비밀번호 검증
- `create_access_token(data: dict, expires_delta: timedelta) → str` : 액세스 토큰 생성
- `create_refresh_token(data: dict) → str` : 리프레시 토큰 생성

- `get_current_user(token: str = Depends(oauth2_scheme)) → User` : 현재 사용자 가져 오기
- `get_current_active_user(current_user: User = Depends(get_current_user)) → User` : 활성 사용자 확인

## 7. 배포 및 실행 환경

---

### 7.1 로컬 개발 환경 설정

```
# 가상환경 생성 및 활성화
python -m venv venv
source venv/bin/activate # Linux/Mac
.\venv\Scripts\activate  # Windows

# 의존성 설치
pip install -r requirements.txt

# 서버 실행
uvicorn app.main:app --reload
```

### 7.2 Docker를 이용한 배포

본 시스템은 Docker를 이용한 컨테이너화 배포를 지원합니다:

```
# 이미지 빌드
docker build -t aisr-backend .

# 컨테이너 실행
docker run -d -p 8000:80 --name aisr-backend-container aisr-backend
```

### 7.3 환경 변수 설정

다음 환경 변수를 설정하여 시스템 구성을 변경할 수 있습니다:

환경 변수	설명	기본값
DATABASE_URL	데이터베이스 연결 URL	sqlite:///./sql_app.db
SECRET_KEY	JWT 시크릿 키	-
ACCESS_TOKEN_EXPIRE_MINUTES	액세스 토큰 만료 시간 (분)	30
REFRESH_TOKEN_EXPIRE_DAYS	리프레시 토큰 만료 시간(일)	7
SMTP_SERVER	이메일 서버 주소	-
SMTP_PORT	이메일 서버 포트	587
SMTP_USER	이메일 사용자 계정	-
SMTP_PASSWORD	이메일 사용자 비밀번호	-



## 8. API 문서화

---

FastAPI는 자동으로 API 문서를 생성합니다. 서버 실행 후 다음 URL을 통해 API 문서에 접근할 수 있습니다.

### 8.1 Swagger UI

서버 실행 후 /docs 경로에서 Swagger UI를 통해 API를 테스트할 수 있습니다.

### 8.2 ReDoc

서버 실행 후 /redoc 경로에서 ReDoc을 통해 보다 자세한 API 문서를 확인할 수 있습니다.

## 9. 테스트

---

본 시스템은 유닛 테스트와 통합 테스트를 통해 안정성을 확보합니다.

### 9.1 테스트 실행 방법

```
# 전체 테스트 실행
pytest

# 특정 모듈 테스트
pytest tests/test_user.py

# 테스트 커버리지 확인
pytest --cov=app tests/
```

### 9.2 주요 테스트 케이스

- 사용자 인증 테스트
- 교육과정 CRUD 테스트
- 수강신청 및 이수 처리 테스트
- 시설 예약 시스템 테스트
- 권한 관리 테스트

## 10. 결론 및 향후 계획

---

본 문서는 체육인재개발원 교육관리 시스템의 백엔드 설계를 설명하고 있습니다. 시스템은 FastAPI 기반으로 개발되었으며, 확장성과 유지보수성을 고려한 구조를 갖추고 있습니다.

### 10.1 향후 개선 사항

- 실시간 알림 시스템 추가
- 데이터 분석 기능 강화
- 모바일 앱 API 확장
- AI 기반 교육 추천 시스템 구현
- 포인트/리워드 시스템 도입