IITP DABT API

설계서

문서 버전: 1.0.0
작성일: 2025-10-30

(주)스위트케이

문서 History

버전	날짜	변경 내용	작성자
v1.0.0	2025-10-30	초기 버전	(주)스위트케이

목차

1. 개요

1.1 프로젝트 소개

IITP API는 장애인 관련 통계 데이터와 POI(Point of Interest), 고용 데이터 정보등을 제공하는 RESTful API 서버입니다. 다양한 데이터 소스로부터 수집된 정보를 표준화된 API 형태로 제공하여 장애인 관련 데이터를 이용할 수 있게 지원합니다.

1.2 핵심 기능 요약

- **기초 통계 데이터**: 장애인 건강, 교육, 고용, 주거, 사회망, 편의시설, 보조기기 등 7개 카테고리 통계 정보
- **POI 정보**: 관광지, 편의시설, 무장애 시설 등 위치 기반 정보
- **POI 위치 기반 검색** : 위치 기반 검색 및 반경 거리 정보 제공
- **고용 관련 데이터**: 장애인 구직자, 구인 정보, 훈련센터 등 15종 고용 관련 정보
- **보안**: API Key 인증, Rate Limiting, 암호화된 설정 관리

---

2. 소프트웨어 아키텍처

2.1 기술 스택

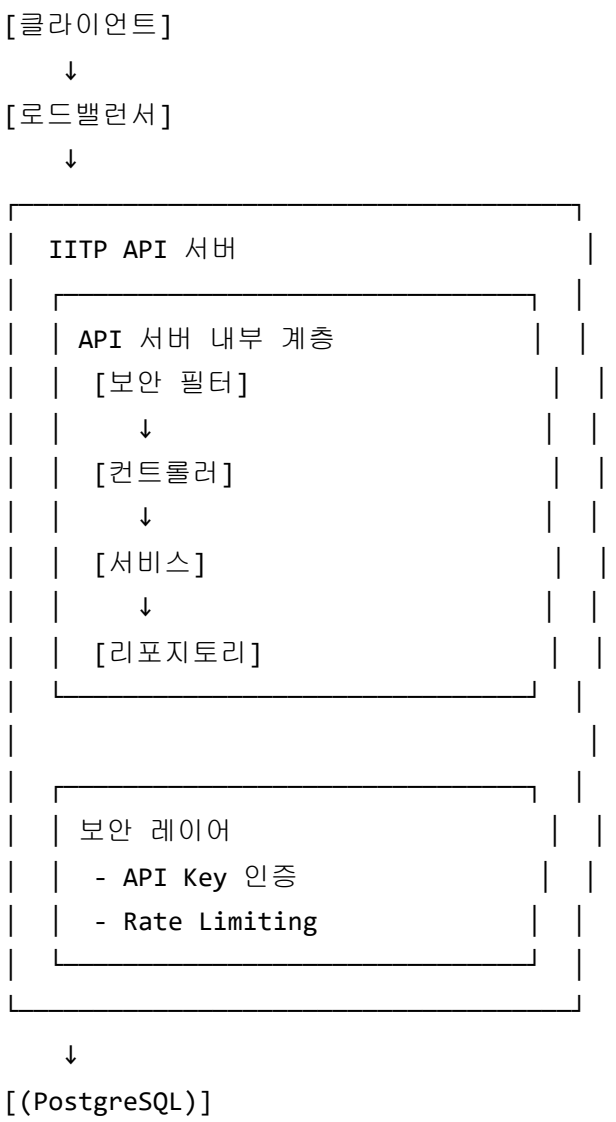
카테고리	기술	버전
언어	Java	21
프레임워크	Spring Boot	3.2.5
데이터베이스	PostgreSQL	16.9
ORM	Spring Data JPA, QueryDSL	5.0.0
보안	Spring Security	-
문서화	SpringDoc OpenAPI	2.3.0
빌드 도구	Gradle	8+

2.2 주요 의존성

```
// 핵심 프레임워크
implementation 'org.springframework.boot:spring-boot-starter-web'
implementation 'org.springframework.boot:spring-boot-starter-data-jpa'
implementation 'org.springframework.boot:spring-boot-starter-security'

// QueryDSL
implementation "com.querydsl:querydsl-jpa:5.0.0:jakarta"

// Rate Limiting
implementation 'com.bucket4j:bucket4j-core:8.7.0'

// OpenAPI 문서화
implementation 'org.springdoc:springdoc-openapi-starter-webmvc-api:2.3.0'

// 암호화
implementation 'com.github.ulisesbocchio:jasypt-spring-boot-starter:3.0.5'
```

2.3 시스템 구성도

시스템 아키텍처



3. 소스 구조 및 설명

3.1 패키지 구조

```
src/main/java/com/sweetk/iitp/api/
├── controller/          # API 컨트롤러
│   └── v1/
│       ├── basic/       # 기초 통계 데이터 API (7개 컨트롤러)
│       ├── poi/         # POI 관련 API (4개 컨트롤러)
│       ├── emp/         # 고용 관련 API (1개 컨트롤러)
│       ├── common/      # 공통 API
│       └── openapi/     # OpenAPI 관리
├── service/             # 비즈니스 로직
│   ├── basic/           # 기초 데이터 서비스 (7개 서비스)
│   ├── poi/             # POI 서비스 (4개 서비스)
│   ├── emp/             # 고용 데이터 서비스 (1개 서비스)
│   └── openapi/         # OpenAPI 서비스
├── repository/          # 데이터 접근 계층
│   ├── basic/           # 기초 데이터 리포지토리
│   ├── poi/             # POI 리포지토리
│   ├── emp/             # 고용 데이터 리포지토리
│   └── openapi/         # OpenAPI 리포지토리
├── entity/              # JPA 엔티티
│   ├── basic/           # 기초 데이터 엔티티
│   ├── poi/             # POI 엔티티
│   ├── emp/             # 고용 데이터 엔티티
│   └── openapi/         # OpenAPI 엔티티
├── dto/                 # Data Transfer Objects
│   ├── basic/           # 기초 데이터 DTO
│   ├── poi/             # POI DTO
│   ├── emp/             # 고용 데이터 DTO
│   └── common/          # 공통 DTO
├── config/              # 설정 클래스
├── security/            # 보안 관련 클래스
├── util/                # 유틸리티 클래스
├── constant/            # 상수 정의
├── exception/           # 예외 처리
└── validator/           # 검증 관련 클래스
```

3.2 주요 모듈 설명

3.2.1 Controller

- **Basic 컨트롤러**: 7개 카테고리의 기초 통계 데이터 API 제공
- **POI 컨트롤러**: 4개 유형의 위치 기반 POI 정보 API 제공
- **EMP 컨트롤러**: 15종의 장애인 고용 관련 데이터 API 제공
- **Common 컨트롤러**: 공통 기능 API 제공

3.2.2 Service

- 비즈니스 로직 처리
- 데이터 변환 및 검증
- 트랜잭션 관리
- 통계 데이터 제한 기능 (limitStatsDataList)

3.2.3 Repository

- JPA 기반 데이터 접근
- QueryDSL을 활용한 동적 쿼리
- 커스텀 쿼리 구현

3.3 설정 파일 구조 및 관리 방식

3.3.1 환경별 설정 파일

- `application.yml` : 공통 설정
- `application-{env}.yml` : 환경별 특화 설정 (local, dev, stage, prod)

3.3.2 주요 설정 구조

기본 설정 구조

```yaml
# 공통 설정 예시
spring:
  profiles:
    active: local
  application:
    name: iitp-api

app:
  version: @version@
  build:
    date: @build.time@
  distance-calculation:
    method: POSTGIS_SPHERE  # PostGIS의 ST_Distance_Sphere 함수 사용 (권장)

api:
  version:
    current: v1
  rate-limit:
    enabled: true
    capacity: 100
    time-window: 60
  stats_data:
    limit_count: 50000
```

플레이스홀더 치환 정보

이 프로젝트는 Gradle 빌드 시 설정 파일의 플레이스홀더를 실제 값으로 치환합니다.

a) 플레이스홀더 항목

- `@version@` : 애플리케이션 버전 (기본값: 0.0.5)
- `@build.time@` : 빌드 시간 (yyyyMMddHHmmss 형식)
- `@springdoc.version@` : OpenAPI 문서 버전 (기본값: v0.0.4)

b) 치환 설정 위치

`build.gradle` 파일에 다음과 같이 설정되어 있습니다:

```
tasks.named('processResources') {
    filesMatching('application*.yml') {
        filter {
            it.replace('@build.time@', buildTime)
              .replace('@version@', jarVersion)
              .replace('@springdoc.version@', apiDocVersion)
        }
    }
}
```

c) 치환 시점 및 절차

- **시점**: `processResources` 태스크 실행 시
- **절차**:
  i. 빌드 시작 시 `buildTime` 과 `jarVersion` 생성
  ii. `processResources` 태스크 실행
  iii. `application*.yml` 파일에서 플레이스홀더 검색 및 치환
  iv. 치환된 파일을 `build/resources/main` 에 복사
  v. JAR 파일에 포함

d) 환경별 설정 파일 관리

- 환경별 설정 파일: `application-{env}.yml` (local, dev, stage, prod)
- 빌드 시 해당 환경의 설정 파일만 포함
- 빌드 명령어:

```
./gradlew buildDev   # dev 환경 설정 파일 사용
./gradlew buildProd  # prod 환경 설정 파일 사용
```

e) 실제 빌드 결과 예시

```yaml
# application.yml 원본
app:
  version: @version@
  build:
    date: @build.time@

# build/resources/main/application.yml (빌드 후)
app:
  version: 0.0.5
  build:
    date: 20250127150230
```

3.3.3 거리 계산 방식 설정

PostGIS 기반 거리 계산

POI API의 위치 기반 검색에서 사용되는 거리 계산 방식을 설정합니다.

지원하는 거리 계산 방식:

- **POSTGIS_SPHERE** (권장): PostGIS의 `ST_Distance_Sphere` 함수 사용
  - 지구의 타원체 형태를 고려한 정확한 거리 계산
  - 장거리 계산에도 높은 정확도 유지
  - PostGIS 설치 필요
- **POSTGIS_PLANAR**: PostGIS의 `ST_Distance` 함수 사용
  - 평면 좌표계 기반 계산으로 빠른 성능
  - 단거리 계산에 적합 (수백 미터 이내)
  - PostGIS 설치 필요
- **POSTGRESQL_EARTH**: PostgreSQL의 `earth_distance` 함수 사용
  - PostGIS 없이도 사용 가능
  - 지구 타원체 기반 계산
  - PostgreSQL 기본 기능만 필요

거리 계산 방식 비교표:

방식	함수	정확도	성능	요구사항	권장 환경
POSTGIS_SPHERE	`ST_Distance_Sphere`	매우 좋음	좋음	PostGIS 확장	운영 환경 (권장)

방식	함수	정확도	성능	요구사항	권장 환경
POSTGIS_PLANAR	`ST_Distance`	양호	매우 좋음	PostGIS 확장	단거리 검색
POSTGRESQL_EARTH	`earth_distance`	좋음	좋음	PostgreSQL 기본	PostGIS 미설치 환경

설정 방법 (`application.yml`):

```yaml
app:
  distance-calculation:
    method: POSTGIS_SPHERE  # POSTGIS_SPHERE, POSTGIS_PLANAR, POSTGRESQL_EARTH
```

PostGIS 사용 방식:

- `method` 설정에 따라 자동으로 PostGIS 사용 여부가 결정됩니다
- `POSTGIS_SPHERE` 또는 `POSTGIS_PLANAR` 선택 시 → PostGIS 확장 사용 (PostGIS 설치 필요)
- `POSTGRESQL_EARTH` 선택 시 → PostGIS 확장 미사용 (PostgreSQL 기본 함수만 사용)

환경별 권장 설정:

- **개발/로컬 환경**: `POSTGRESQL_EARTH` (PostGIS 설치 없이 사용 가능)
- **운영 환경**: `POSTGIS_SPHERE` (가장 정확한 거리 계산, PostGIS 자동 사용)

PostGIS 확장 요구사항 (`POSTGIS_SPHERE` , `POSTGIS_PLANAR` 사용 시):

- PostgreSQL 16+ 버전
- PostGIS 3.0+ 확장 설치 및 활성화
- 데이터베이스에 `CREATE EXTENSION postgis;` 실행 필요
- 자세한 설치 방법은 4.2 PostGIS 확장 확인 및 설치 참조

4.3.4 암호화 설정 방식

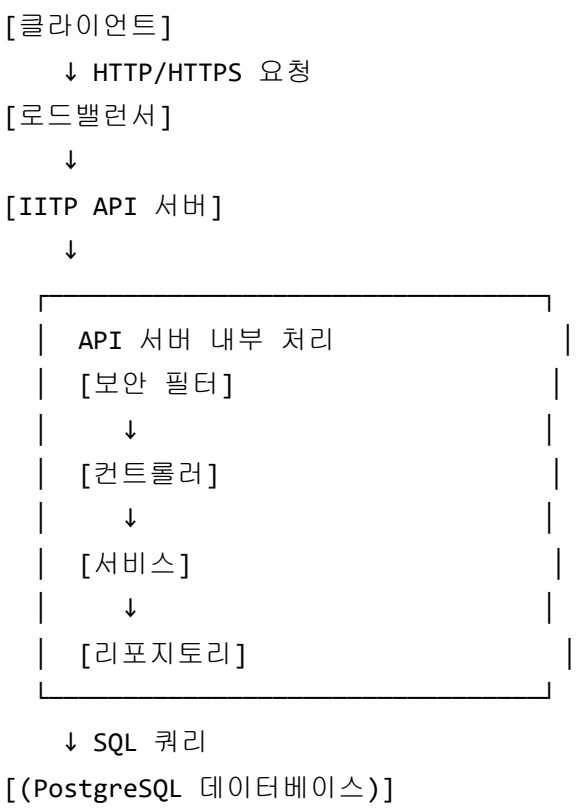
- Jasypt를 사용한 민감 정보 암호화
- ENC() 형태로 암호화된 값 저장
- 환경 변수로 복호화 키 관리

---

4. API 처리

4.1 API 처리 Flow

4.1.1 시스템 연동 Flow



참고: 데이터는 별도의 ETL 프로세스를 통해 PostgreSQL에 저장되며, IITP API 서버는 저장된 데이터를 조회하여 제공합니다.

4.1.2 API 요청-응답 Flow

```
1. Client
   └─> API 요청 전송 (X-API-KEY 헤더 포함)

2. Filter (보안 필터)
   ├─> API Key 인증 검증
   └─> Rate Limiting 체크

3. Controller
   └─> 요청 경로 및 매개변수 파싱

4. Service
   ├─> 비즈니스 로직 처리
   └─> 데이터 변환 및 검증

5. Repository
   └─> 데이터베이스 쿼리 실행

6. Database
   └─> 데이터 조회 및 반환

7. Repository <- Database
   └─> 엔티티 반환

8. Service <- Repository
   └─> DTO 변환

9. Controller <- Service
   └─> JSON 응답 생성

10. Client <- Controller
    └─> 최종 JSON 응답 반환
```

4.2 API 헤더 형식

4.2.1 필수 헤더

```
X-API-KEY: {API_KEY}
Content-Type: application/json
```

4.2.2 선택적 헤더

```
Accept: application/json
User-Agent: {클라이언트_정보}
```

4.2.3 인증 제외 경로

다음 경로는 API Key 인증이 필요하지 않습니다:

- `/v3/api-docs` - OpenAPI 문서
- `/docs/**` - API 문서
- `/api/v1/comm/**` - 공통 API
- `/favicon.ico` - 파비콘

4.3 요청 형식

4.3.1 기본 요청 구조

```json
{
  "page": 0,
  "size": 20,
  "searchParams": {
    "keyword": "검색어",
    "category": "카테고리"
  }
}
```

4.3.2 위치 기반 검색 요청

```json
{
  "latitude": 37.5665,
  "longitude": 126.9780,
  "radius": 1000,
  "category": "관광지"
}
```

4.4 응답 형식

4.4.1 성공 응답

```
{
  "success": true,
  "code": "SUCCESS",
  "message": "요청이 성공적으로 처리되었습니다.",
  "data": {
    "content": [...],
    "page": {
      "number": 0,
      "size": 20,
      "totalElements": 100,
      "totalPages": 5
    }
  },
  "timestamp": "2025-01-27T10:00:00Z"
}
```

4.4.2 에러 응답

```
{
  "success": false,
  "code": "UNAUTHORIZED",
  "message": "유효하지 않은 API Key 입니다.",
  "data": null,
  "timestamp": "2025-01-27T10:00:00Z"
}
```

---

5. 기능별 상세 설명

5.1 기초 통계 데이터 (Basic)

5.1.1 제공 기능 (7개 카테고리)

각 카테고리별로 별도의 컨트롤러와 서비스를 제공합니다:

- **건강 관리 현황** (`BasicHealthController`): 의료이용 현황, 진료비 현황, 생활체육 실행 유형, 운동 지원 사항
- **진로 교육 현황** (`BasicEduController`): 교육 관련 통계
- **고용 현황** (`BasicEmpController`): 근로자 고용 현황
- **주거 자립 현황** (`BasicHousingController`): 신규등록 장애인 현황
- **사회망 현황** (`BasicSocialController`): 사회망 관련 통계
- **편의 시설 제공 현황** (`BasicFacilityController`): 편의시설 관련 통계
- **보조기기 사용 현황** (`BasicAidController`): 보조기기 관련 통계

Basic API 공통 패턴

모든 Basic 카테고리의 API는 다음과 같은 3가지 공통 패턴을 제공합니다:

1. **Info API** (`/info`)
   - 용도: 데이터 소스 정보 조회
   - 예시: `GET /api/v1/basic/health/medical-usage/info`
   - 반환: 데이터 출처, 수집 방법, 데이터 설명 등 메타데이터
2. **Latest API** (`/latest?from=YYYY&to=YYYY`)
   - 용도: 최신 데이터 조회 (기간별)
   - 예시: `GET /api/v1/basic/health/medical-usage/latest?from=2020&to=2023`
   - 파라미터:
     - `from` : 시작 연도
     - `to` : 종료 연도 (선택)
   - 반환: 지정 기간의 최신 통계 데이터
3. **Year API** (`/{statYear}`)
   - 용도: 특정 연도 데이터 조회
   - 예시: `GET /api/v1/basic/health/medical-usage/2023`
   - 파라미터: `statYear` (경로 변수)
   - 반환: 해당 연도의 통계 데이터

공통 기능

- **데이터 제한**: `app.api.stats_data.limit_count` 설정값(기본 50,000건)으로 데이터 응답 크기 제한
- **메타데이터 포함**: 분류 코드(Classification), 항목 코드(Item) 정보 제공
- **데이터 변환**: DB 엔티티를 표준화된 DTO로 변환하여 반환

5.1.2 주요 API 엔드포인트 패턴

각 카테고리별로 동일한 패턴의 API를 제공합니다:
자세한 API 규격은 API 규격서를 참고합니다.

건강 관리 현황 (`/api/v1/basic/health`)

```
GET /api/v1/basic/health/medical-usage/info
GET /api/v1/basic/health/medical-usage/latest
GET /api/v1/basic/health/medical-usage/{statYear}
GET /api/v1/basic/health/disease-cost-sub/info
GET /api/v1/basic/health/disease-cost-sub/latest
GET /api/v1/basic/health/disease-cost-sub/{statYear}
GET /api/v1/basic/health/sport-exec-type/info
GET /api/v1/basic/health/sport-exec-type/latest
GET /api/v1/basic/health/sport-exec-type/{statYear}
GET /api/v1/basic/health/exrc-best-aid/info
GET /api/v1/basic/health/exrc-best-aid/latest
GET /api/v1/basic/health/exrc-best-aid/{statYear}
```

진로 교육 현황 (`/api/v1/basic/edu`)

```
GET /api/v1/basic/edu/voca-exec/info
GET /api/v1/basic/edu/voca-exec/latest
GET /api/v1/basic/edu/voca-exec/{statYear}
GET /api/v1/basic/edu/voca-exec-way/info
GET /api/v1/basic/edu/voca-exec-way/latest
GET /api/v1/basic/edu/voca-exec-way/{statYear}
```

고용 현황 (`/api/v1/basic/emp`)

```
GET /api/v1/basic/emp/natl/info
GET /api/v1/basic/emp/natl/latest
GET /api/v1/basic/emp/natl/{statYear}
GET /api/v1/basic/emp/natl-private/info
GET /api/v1/basic/emp/natl-private/latest
GET /api/v1/basic/emp/natl-private/{statYear}
GET /api/v1/basic/emp/natl-gov-org/info
GET /api/v1/basic/emp/natl-gov-org/latest
GET /api/v1/basic/emp/natl-gov-org/{statYear}
GET /api/v1/basic/emp/natl-public/info
GET /api/v1/basic/emp/natl-public/latest
GET /api/v1/basic/emp/natl-public/{statYear}
GET /api/v1/basic/emp/natl-dis-type-sev/info
GET /api/v1/basic/emp/natl-dis-type-sev/latest
GET /api/v1/basic/emp/natl-dis-type-sev/{statYear}
GET /api/v1/basic/emp/natl-dis-type-indust/info
GET /api/v1/basic/emp/natl-dis-type-indust/latest
GET /api/v1/basic/emp/natl-dis-type-indust/{statYear}
```

주거 자립 현황 (`/api/v1/basic/housing`)

```
GET /api/v1/basic/housing/reg-natl-by-new/info
GET /api/v1/basic/housing/reg-natl-by-new/latest
GET /api/v1/basic/housing/reg-natl-by-new/{statYear}
GET /api/v1/basic/housing/reg-natl-by-age-type-sev-gen/info
GET /api/v1/basic/housing/reg-natl-by-age-type-sev-gen/latest
GET /api/v1/basic/housing/reg-natl-by-age-type-sev-gen/{statYear}
GET /api/v1/basic/housing/reg-sido-by-type-sev-gen/info
GET /api/v1/basic/housing/reg-sido-by-type-sev-gen/latest
GET /api/v1/basic/housing/reg-sido-by-type-sev-gen/{statYear}
GET /api/v1/basic/housing/life-supp-need-lvl/info
GET /api/v1/basic/housing/life-supp-need-lvl/latest
GET /api/v1/basic/housing/life-supp-need-lvl/{statYear}
GET /api/v1/basic/housing/life-primcarer/info
GET /api/v1/basic/housing/life-primcarer/latest
GET /api/v1/basic/housing/life-primcarer/{statYear}
GET /api/v1/basic/housing/life-maincarer/info
GET /api/v1/basic/housing/life-maincarer/latest
GET /api/v1/basic/housing/life-maincarer/{statYear}
GET /api/v1/basic/housing/life-supp-field/info
GET /api/v1/basic/housing/life-supp-field/latest
GET /api/v1/basic/housing/life-supp-field/{statYear}
```

사회망 현황 (`/api/v1/basic/social`)

```
GET /api/v1/basic/social/partic-freq/info
GET /api/v1/basic/social/partic-freq/latest
GET /api/v1/basic/social/partic-freq/{statYear}
GET /api/v1/basic/social/contact-cntfreq/info
GET /api/v1/basic/social/contact-cntfreq/latest
GET /api/v1/basic/social/contact-cntfreq/{statYear}
```

편의 시설 제공 현황 (`/api/v1/basic/facility`)

```
GET /api/v1/basic/facility/welfare-usage/info
GET /api/v1/basic/facility/welfare-usage/latest
GET /api/v1/basic/facility/welfare-usage/{statYear}
```

보조기기 사용 현황 (`/api/v1/basic/aid`)

```
GET /api/v1/basic/aid/device-usage/info
GET /api/v1/basic/aid/device-usage/latest
GET /api/v1/basic/aid/device-usage/{statYear}
GET /api/v1/basic/aid/device-need/info
GET /api/v1/basic/aid/device-need/latest
GET /api/v1/basic/aid/device-need/{statYear}
```

5.1.3 참조 DB 테이블

공통 테이블 (Info API용)

- `stats_src_data_info` - 통계 데이터 소스 정보 (모든 info API에서 참조)

메타데이터 테이블 (Latest/Year API용)

- `stats_kosis_metadata_code` - KOSIS 원천 통계 데이터의 분류/항목 코드 정보 (메타데이터)

건강 관리 현황

- `stats_dis_hlth_medical_usage` - 의료이용 현황
- `stats_dis_hlth_disease_cost_sub` - 진료비 현황
- `stats_dis_hlth_sport_exec_type` - 생활체육 실행 유형
- `stats_dis_hlth_exrc_best_aid` - 운동 지원 사항

진로 교육 현황

- `stats_dis_edu_voca_exec` - 교육 관련 통계
- `stats_dis_edu_voca_exec_way` - 교육 관련 통계

고용 현황

- `stats_dis_emp_natl` - 근로자 고용 현황
- `stats_dis_emp_natl_private` - 민간부문 고용 현황
- `stats_dis_emp_natl_gov_org` - 정부기관 고용 현황
- `stats_dis_emp_natl_public` - 공공부문 고용 현황
- `stats_dis_emp_natl_dis_type_sev` - 장애유형별 고용 현황
- `stats_dis_emp_natl_dis_type_indust` - 산업별 고용 현황

주거 자립 현황

- `stats_dis_reg_natl_by_new` - 신규등록 장애인 현황
- `stats_dis_reg_natl_by_age_type_sev_gen` - 연령별 신규등록 현황

- `stats_dis_reg_sido_by_type_sev_gen` - 시도별 신규등록 현황
- `stats_dis_life_supp_need_lvl` - 생활지원 필요 수준
- `stats_dis_life_primcarer` - 주 양육자
- `stats_dis_life_maincarer` - 주 돌봄 제공자
- `stats_dis_life_supp_field` - 생활지원 분야

사회망 현황

- `stats_dis_soc_partic_freq` - 사회참여 빈도
- `stats_dis_soc_contact_cntfreq` - 사회적 접촉 빈도

편의 시설 제공 현황

- `stats_dis_fclty_welfare_usage` - 편의시설 관련 통계

보조기기 사용 현황

- `stats_dis_aid_device_usage` - 보조기기 관련 통계
- `stats_dis_aid_device_need` - 보조기기 필요 현황

5.2 POI 정보 (이동형)

5.2.1 제공 기능 (4개 컨트롤러)

각 POI 유형별로 별도의 컨트롤러와 서비스를 제공합니다:

- **관광 POI** (`MvPoiController`): 관광지, 식당, 쇼핑, 숙박 시설
- **지하철 엘리베이터** (`PoiSubwayElevatorController`): 지하철 엘리베이터 위치 정보
- **무장애 관광지 시설** (`PoiTourBfFacilityController`): Barrier Free 관광지 시설
- **공중 화장실** (`PoiPublicToiletInfoController`): 공중 화장실 위치 및 시설 정보

POI API 공통 패턴

1. **검색 API** (`/search`)
   - 용도: 일반 키워드 검색
   - 지원 기능: 이름, 주소, 설명 등 다양한 필드 검색
   - 페이징 지원: `/search/paging`
2. **위치 기반 검색 API** (`/search/location`)
   - 용도: 위도/경도 기반 반경 검색
   - 파라미터:
     - `latitude` : 위도
     - `longitude` : 경도

    - `radius` : 검색 반경 (미터 단위)
  - 거리 계산: PostGIS Sphere 방식 사용
  - 페이징 지원: `/search/location/paging`
3. **카테고리별 조회 API** (`/category/{categoryType}`)
  - 용도: POI 카테고리별 조회
  - 카테고리: 관광지, 식당, 쇼핑, 숙박 등
  - 페이징 지원: `/category/{categoryType}/paging`

공통 기능

- **페이징**: 모든 목록 조회 API는 페이징 지원
- **거리 계산**: 위치 기반 검색 시 거리 정보 포함 (PostGIS의 ST_Distance_Sphere 함수 사용)
- **좌표 정보**: 모든 POI 데이터에 위도/경도 포함
- **PostGIS 활용**:
  - 공간 데이터 처리를 위한 PostGIS 확장 사용
  - 위치 기반 반경 검색 시 PostGIS의 `ST_Distance_Sphere` 함수로 정확한 거리 계산
  - 지구 타원체 형태를 고려한 높은 정확도의 거리 계산
  - 설정값 (`app.distance-calculation.method`)에 따라 거리 계산 방식 선택 가능

5.2.2 주요 API 엔드포인트

관광 POI (`/api/v1/poi`)

```
GET /api/v1/poi/{poiId}                          # POI 상세 조회
GET /api/v1/poi/category/{categoryType}          # 카테고리별 조회
GET /api/v1/poi/category/{categoryType}/paging   # 카테고리별 조회 (페이징)
GET /api/v1/poi/search                           # POI 검색
GET /api/v1/poi/search/paging                    # POI 검색 (페이징)
GET /api/v1/poi/search/location                  # 위치 기반 검색
GET /api/v1/poi/search/location/paging           # 위치 기반 검색 (페이징)
```

지하철 엘리베이터 (`/api/v1/poi/elevator/subway`)

```
GET /api/v1/poi/elevator/subway/search           # 지하철 엘리베이터 검색
GET /api/v1/poi/elevator/subway/search/paging    # 지하철 엘리베이터 검색 (페이징)
```

무장애 관광지 시설 (`/api/v1/poi/barrier-free/tour`)

```
GET /api/v1/poi/barrier-free/tour/search         # 무장애 관광지 검색
GET /api/v1/poi/barrier-free/tour/search/paging  # 무장애 관광지 검색 (페이징)
```

공중 화장실 (`/api/v1/poi/public-toilet`)

```
GET /api/v1/poi/public-toilet/search             # 공중 화장실 검색
GET /api/v1/poi/public-toilet/search/paging      # 공중 화장실 검색 (페이징)
```

5.2.3 참조 DB 테이블

- `mv_poi` - 관광 POI 정보
- `poi_subway_elevator` - 지하철 엘리베이터 정보
- `poi_tour_bf_facility` - 무장애 관광지 시설 정보
- `poi_public_toilet_info` - 공중 화장실 정보

5.3 고용 관련 데이터 (EMP)

5.3.1 제공 기능 (15종 데이터)

`EmpDisController` 에서 15종의 장애인 고용 관련 데이터를 제공합니다:

고용 현황 데이터

- 장애인 구직자 현황: 구직자 인원, 구직률 등의 현황 정보
- 장애인 구인 정보: 채용 공고, 구인 수 등의 구인 정보
- 발달장애인훈련센터 이용자 현황: 훈련센터 이용 통계
- 지역별 장애인 고용 현황: 시도별 장애인 고용 현황

고용 지원 정책 데이터

- 신규고용장려금 지급 현황: 장려금 지급 내역 및 통계
- 부담금감면 연계고용사업장 정보: 감면 대상 사업장 정보
- 발달장애인 지원 기관 및 제공서비스: 지원 기관 현황

의무고용 관련 데이터

- 장애인 의무고용 이행 현황: 의무고용 이행률, 현황
- 장애인 의무고용 의무이행 현황: 의무 이행 현황
- 장애인 의무고용 유형별 현황: 유형별 의무고용 통계
- 장애인 의무고용 산업별 현황: 산업별 의무고용 통계

EMP API 공통 패턴

모든 EMP API는 단순 조회형 API로 제공됩니다:

1. **조회 API** (`GET`)
    - 용도: 각 데이터 유형별 현황 정보 조회
    - 반환: 고용 관련 통계 및 현황 데이터
    - 특징:
        - 필터링: 지역, 기간, 유형 등 다양한 필터 지원
        - 정렬: 다양한 정렬 옵션 제공
        - 페이징: 대량 데이터에 대한 페이징 지원

공통 기능

- **다양한 필터**: 지역, 기간, 장애유형 등 다양한 필터 조건 지원
- **정렬 기능**: 다중 컬럼 정렬 지원
- **페이징**: 대량 데이터 처리를 위한 페이징 지원
- **검색**: 키워드 기반 검색 기능

5.3.2 주요 API 엔드포인트

```
GET /api/v1/emp/job/seeker                  # 구직자 현황
GET /api/v1/emp/job/posting                 # 구인 정보
GET /api/v1/emp/center/usage                # 훈련센터 이용자 현황
GET /api/v1/emp/region                      # 지역별 고용 현황
GET /api/v1/emp/incentive                   # 신규고용장려금 현황
GET /api/v1/emp/workplace/burden-redct      # 부담금감면 연계고용사업장
GET /api/v1/emp/dev/support-org             # 발달장애인 지원기관
GET /api/v1/emp/obligation/status           # 장애인 의무고용 이행 현황
GET /api/v1/emp/obligation/fulfillment      # 장애인 의무고용 의무이행 현황
GET /api/v1/emp/obligation/by-type          # 장애인 의무고용 유형별 현황
GET /api/v1/emp/obligation/by-indust        # 장애인 의무고용 산업별 현황
GET /api/v1/emp/std/workplace               # 장애인 의무고용 현황
GET /api/v1/emp/startup/lecture             # 장애인 의무고용 의무이행 현황
GET /api/v1/emp/staff/train-crs             # 장애인 의무고용 유형별 현황
GET /api/v1/emp/consulting/his              # 장애인 의무고용 산업별 현황
```

5.3.3 참조 DB 테이블

- `emp_dis_jobseeker_status` - 장애인 구직자 현황
- `emp_dis_job_posting` - 장애인 구인 정보
- `emp_dis_center_usage` - 발달장애인훈련센터 이용자 현황
- `emp_dis_regional_status` - 지역별 장애인 고용 현황
- `emp_dis_emp_incentive` - 신규고용장려금 지급 현황
- `emp_dis_burden_workplace` - 부담금감면 연계고용사업장 정보

- `emp_dis_dev_support_org` - 발달장애인 지원 기관 및 제공서비스
- `emp_dis_obligation_status` - 장애인 의무고용 이행 현황
- `emp_dis_obligation_fulfillment` - 장애인 의무고용 의무이행 현황
- `emp_dis_obligation_by_type` - 장애인 의무고용 유형별 현황
- `emp_dis_obligation_by_indust` - 장애인 의무고용 산업별 현황
- `emp_dis_std_workplace` - 장애인 의무고용 현황
- `emp_dis_startup_lecture` - 장애인 의무고용 의무이행 현황
- `emp_dis_staff_train_crs` - 장애인 의무고용 유형별 현황
- `emp_dis_consulting_his` - 장애인 의무고용 산업별 현황

---

6. OpenAPI 문서 관련

6.1 문서 생성 및 배포

6.1.1 문서 생성 명령어

```
# OpenAPI 문서 생성
./gradlew generateOpenApiDocs

# 문서 생성 및 복사
./gradlew copyDocsToResources
```

6.1.2 환경별 문서 생성 설정

- 로컬 환경에서 문서 생성 (포트: 28081)
- SpringDoc OpenAPI 자동 스캔
- 컨트롤러 패키지 기반 문서 생성

6.2 문서 자동화 프로세스

6.2.1 copyDocsToResources 태스크 동작 흐름

```
1. generateOpenApiDocs
   └─> OpenAPI 문서 생성 (JSON/YAML)

2. copyVersionedDoc
   └─> 버전별 문서 복사 (openapi-v0.0.4.yaml)
       └─> docs-dist/
       └─> src/main/resources/static/docs/

3. copyLatestDoc
   └─> 최신 문서 복사 (latest.yaml)
       └─> src/main/resources/static/docs/

4. add-x-tagGroups
   └─> Python 스크립트 실행
       └─> x-tagGroups 자동 추가

5. 문서 완성
   └─> JAR 내부에 포함 완료
```

6.2.2 관련 디렉토리 구조

```
docs-dist/                          # OpenAPI 문서 출력 디렉토리
├── openapi-v0.0.4.yaml             # 버전별 문서

src/main/resources/static/docs/     # JAR 내부 문서 디렉토리
├── latest.yaml                     # 최신 문서
└── openapi-v0.0.4.yaml             # 버전별 문서

scripts/
├── add-x-tagGroups.py              # 그룹핑 스크립트
└── api-doc-filename.txt            # 현재 버전 파일명
```

6.2.3 그룹핑 자동화 (x-tagGroups)

- Python 스크립트로 x-tagGroups 자동 추가
- Stoplight Elements 호환성 보장
- 실제 태그와 일치하지 않는 그룹 자동 필터링

6.3 문서 활용 방법

6.3.1 API 문서 접근

- **JSON 형태**: `/v3/api-docs`
- **YAML 형태**: `/docs/latest.yaml`
- **버전별 문서**: `/docs/openapi-v0.0.4.yaml`

6.3.2 Stoplight Elements 연동

- x-tagGroups를 통한 API 그룹핑
- 태그 기반 API 분류
- 자동 그룹핑으로 문서 구조화

---

7. 데이터 수집 방식 및 출처

참고: IITP API 서버는 데이터를 제공하는 API 서버이며, 실제 데이터 수집은 별도의 ETL 프로세스에서 수행됩니다.

7.1 기초 통계 데이터 (Base)

- **데이터 출처**: KOSIS 통계 데이터
- **갱신 주기**: 분기별 또는 연간
- **데이터 형태**: 구조화된 통계 데이터

7.2 POI 데이터

- **데이터 출처**: 관광공사, 지하철공사, 지자체
- **갱신 주기**: 월별 또는 분기별
- **데이터 형태**: 위치 정보 포함 POI 데이터

7.3 고용 관련 데이터 (EMP)

- **데이터 출처**: 한국장애인고용공단, 한국장애인개발원, 나라 HR
- **갱신 주기**: 실시간 또는 일별
- **데이터 형태**: 고용 관련 구조화 데이터

---

8. 로깅

8.1 로깅 설정

8.1.1 기본 설정

- Log4j2 기반 로깅
- 환경별 로그 설정 (`log4j2-{profile}.xml`)
- 구조화된 로그 출력

8.1.2 로그 패턴

기본 로그 패턴: `%d{yyyy-MM-dd HH:mm:ss.SSS} [%thread] %-5level %logger{36} - %msg%n`

패턴 구성 요소

- `%d{yyyy-MM-dd HH:mm:ss.SSS}` : 타임스탬프 (년-월-일 시:분:초.밀리초)
- `[%thread]` : 스레드명 (대괄호로 감쌈)
- `%-5level` : 로그 레벨 (INFO, DEBUG, ERROR 등, 5칸 왼쪽 정렬)
- `%logger{36}` : 로거명 (최대 36자, 패키지명 축약)
- `%msg` : 로그 메시지
- `%n` : 줄바꿈

로그 출력 예시

```
2025-01-27 15:02:30.123 [http-nio-8080-exec-1] INFO  c.s.iitp.api.controller.v1.basic.BasicHeal
```

8.1.3 로그 파일 관리

- 로그 파일 저장 경로: `logs/`
- 파일명: `iitp-api.log`
- 롤링 정책:
  - 일별 롤링: 매일 자정에 새 파일 생성
  - 크기 제한: 파일당 10MB 초과 시 롤링
  - 보관 기간: 최근 10개 파일 유지

8.2 로그 레벨

- **Root**: INFO
- **Application** (`com.sweetk.iitp`): DEBUG
- **QueryDSL** (`com.querydsl`): DEBUG
- **Hibernate SQL** (`org.hibernate.SQL`): DEBUG
- **Hibernate Binder** (`org.hibernate.type.descriptor.sql.BasicBinder`): TRACE

Appendix A: 보안 및 설정 상세

A.1 API Key 인증 방식

A.1.1 인증 프로세스

1. 클라이언트가 X-API-KEY 헤더로 API Key 전송
2. 서버에서 API Key 유효성 검증
3. 활성 상태 및 유효 기간 확인
4. 인증 성공 시 사용자 정보 로드

A.1.2 API Key 관리

- `open_api_auth_key` 테이블에서 관리
- 활성 상태 (`active_yn`)
- 유효 기간 (`start_dt` , `end_dt`)
- 삭제 상태 (`del_yn`)

A.2 Rate Limiting 설정

A.2.1 기본 설정

```yaml
api:
  rate-limit:
    enabled: true
    capacity: 100        # 시간당 요청 수
    time-window: 60      # 시간 윈도우 (초)
```

A.2.2 구현 방식

- Bucket4j 라이브러리 사용
- 토큰 버킷 알고리즘
- 메모리 기반 제한

A.3 암호화 설정 방법

A.3.1 Jasypt 설정

```yaml
spring:
  datasource:
    password: ENC({암호화된_비밀번호})
```

A.3.2 암호화 명령어

```
# 암호화
./gradlew encrypt --plain-text "평문" --password "암호화키"

# 암호화 스크립트 사용
./encrypt.sh "평문" "암호화키"
```

A.3.3 환경 변수 설정

```
export JASYPT_ENCRYPTOR_PASSWORD=암호화키
```