|  |
| --- |
| FuelcellDr. 프로젝트 인수인계 문서 |
| |  |  | | --- | --- | | **작 성 자** | **구 민 주** | | **작성기간** | **2025.03.18~2024.03.25.** | |
|  |
|  |

The information contained is proprietary and confidential.

Do not duplicate without permission.

NSTEK Technologies Ltd. NSTEK Technologies Inc

Junbgu bldg 5F, 329, Yeongdong-daero, Gangnam-gu,

Seoul, Korea NSTEK Inc

TEL : +82)02.6207.0880 Fax : 82)02.6207.0888

Preface

The 21st century has become an era where a digital network can overcome time and space.

Our current environment is challenging us with a new paradigm and with help of the Internet,

we’re one step closer to reach the goal.

We are making network and Internet based solutions that are high in portability and scalability.

As a young start-up company, NSTEK will not hesitate to face the difficult challenges.

Our belief that the most profound solution is in a human mind will carry us through the 21st century.

NSTEK Technologies Ltd.

5F Jung Bu building, 329 Young-Dong Street, Kang Nam-GU, Seoul, Korea NSTEK Inc.

TEL: + 82)02.6207.0880 Fax: + 82)02.6207.0888

About this Document

This application note is what maps out the \_\_\_ Development Blueprint for replacing \_\_\_

on Traffic Management System.

Related Documents

Reserved

This Document

This following is a brief description of the contents of each section:

Reserved

Table of Contents

[1. 프로젝트 개요 5](#_Toc193813721)

[1.1. 프로젝트 주요 목적 5](#_Toc193813722)

[1.2. 주요 기능 5](#_Toc193813723)

[1.3. 사용 언어 5](#_Toc193813724)

[1.4. 프로젝트 경로 및 문서 위치 5](#_Toc193813725)

[1.5. 주요 DB 6](#_Toc193813726)

[1.5.1. 사용관리자, 발전소, 그룹, 연료전지 ERD 6](#_Toc193813727)

[1.5.2. 테이블 설명: 7](#_Toc193813728)

[1.5.3. 스택진단 관련 테이블 ERD 8](#_Toc193813729)

[1.5.4. 테이블 설명: 9](#_Toc193813730)

[1.6. API 명세 및 Redis 키 구조 문서 위치 9](#_Toc193813731)

[2. 시스템 구조 및 파일 설명 10](#_Toc193813732)

[2.1. 전체 폴더 구조 10](#_Toc193813733)

[2.2. 주요 파일 및 폴더 상세 설명 15](#_Toc193813734)

[3. 주요 기능 및 코드 분석 31](#_Toc193813735)

[3.1. 세션 기반 다중 데이터베이스 접근 시스템 31](#_Toc193813736)

[3.1.1. 통합 작동 원리 31](#_Toc193813737)

[3.1.2. 세션 관리 시스템(get\_session.js) 32](#_Toc193813738)

[3.1.3. Redis 세션 연동(session\_redis.php) 34](#_Toc193813739)

[3.2. 데이터베이스 연결 관리(config.php, update\_db\_config.php, db\_config.php) 35](#_Toc193813740)

[3.2.1. 동작 프로세스 35](#_Toc193813741)

[3.2.2. 기본 설정(config.php) 36](#_Toc193813742)

[3.2.3. 데이터베이스 전환 프로세스(update\_db\_config.php) 36](#_Toc193813743)

[3.2.4. 데이터베이스 연결 관리(db\_config.php) 36](#_Toc193813744)

[3.3. 사이트 구조 및 계층관리 시스템(fuelcellSelector.js) 36](#_Toc193813745)

[3.3.1. 동작 프로세스 36](#_Toc193813746)

[3.3.2. 계층 구조 초기화 및 로드 37](#_Toc193813747)

[3.3.3. 설정 및 선택 관리 시스템 38](#_Toc193813748)

[3.3.4. UI 연동 및 이벤트 처리 39](#_Toc193813749)

[3.4. 스택 검색 시스템(stack\_search.js) 41](#_Toc193813750)

[3.4.1. 검색 유형 전환 및 조건 처리 41](#_Toc193813751)

[3.4.2. 그래프 데이터 처리 및 파일 관리 42](#_Toc193813752)

[3.4.3. 북마크 관리 시스템 44](#_Toc193813753)

[3.5. 유닛컨트롤(unitControl.js) 46](#_Toc193813754)

[3.5.1. 상태 모니터링 46](#_Toc193813755)

[3.5.2. 데이터 갱신 및 표시 47](#_Toc193813756)

[3.5.3. 진단 명령어 제어 시스템 47](#_Toc193813757)

[3.5.4. 시스템 초기화 및 연동 49](#_Toc193813758)

# 프로젝트 개요

## 프로젝트 주요 목적

* 연료전지 발전소 통합 모니터링 및 관리 시스템 구축
* 연료전지 성능 데이터 실시간 분석 및 시각화
* BOP(Balance of Plant) 및 스택(Stack) 진단 기능 제공
* 발전소/그룹/연료전지 계층 구조 기반 관리 시스템 구현
* AI 학습　기반 고장 예측 및 진단 기능 제공

## 주요 기능

* 대시보드 시스템: 실시간 연료전지 운영 정보 및 성능 지표 모니터링
* 사용자 관리: 사용자 계정, 권한, 보안 설정 및 발전소별 접근 제어
* BOP 진단 시스템: 센서 데이터 분석,\87 정상 데이터 학습, 센서 레이블링
* 스택 진단 시스템: 스택 성능 분석, 진단 로그 관리, 북마크 및 태그 관리
* 유닛 컨트롤: 진단 장비 원격 제어 및 다양한 진단 모드 지원
* 실시간 모니터링: 전류/전압/수소/공기 상태 실시간 모니터링
* 사용자 관리: 관리자 권한 설정 및 발전소별 접근 권한 관리
* 다국어 지원: 한국어/영어 언어 전환 기능개발 환경 및 기술 스택

## 사용 언어

* 프론트엔드: JavaScript, HTML5, CSS3
* 백엔드: PHP 7.x
* 데이터베이스: MySQL
* 캐싱 시스템: Redis
* 데이터 형식: JSON, CSV

## 프로젝트 경로 및 문서 위치

* 프로젝트 URL : <http://fuelcelldr.nstek.com:11180/FDC/Proj/TRUNK/>
* 프로젝트 서버 경로: /home/nstek/h2\_system/patch\_active/FDC/Proj/TRUNK
* 인수인계 문서 위치 : /home/nstek/h2\_system/patch\_active/FDC/Proj/TRUNK/doc

　１）FDC 인수인계 문서.docx: 프로젝트의 전체적인 구조와 기능을 상세히 설명하는 메인 문서

　２）FDC\_README\_detail.md: 각 파일과 디렉토리의 상세 설명이 담긴 기술 문서

　３）FDC\_README\_summary.md: 프로젝트의 핵심 기능과 구조를 요약한 개요 문서

## 주요 DB

### 사용관리자, 발전소, 그룹, 연료전지 ERD

텍스트, 도표, 폰트, 평행이(가) 표시된 사진

자동 생성된 설명

### 테이블 설명:

* users: 시스템 사용자 정보 관리
* 관리자(Admin), 로컬 관리자(LocalAdmin), 일반 사용자(User) 권한 구분
* 계정 상태 및 로그인 기록 추적
* powerplants: 발전소 기본 정보
* 발전소별 총 용량 및 연료전지 수량 관리
* 위치 정보 및 등록일 기록
* fuelcell\_groups: 연료전지 그룹 정보
* 발전소 내 그룹 구성 관리
* 그룹별 전기/열 생산 용량 정보
* fuelcells: 개별 연료전지 관리
* 연료전지 상태 추적(normal, watchout, warning, critical, stop)
* 설치일 및 용량 정보 관리
* user\_powerplants/user\_groups: 사용자-발전소/그룹 간 접근 권한 관리
* 사용자별 접근 가능한 발전소 및 그룹 맵핑

### 스택진단 관련 테이블 ERD

텍스트, 스크린샷, 편지, 문서이(가) 표시된 사진

자동 생성된 설명

### 테이블 설명:

* search: 스택 성능 측정 데이터 저장
* SIN/PULSE 모드 측정 결과 저장
* 위치 좌표(X1, X2, Y1, Y2) 및 임피던스 측정값(H-M, M-L 등) 관리
* 측정 시간, 주파수 범위, 오류 코드(MERR) 정보 포함
* 사용자 입력 메모(BIGO) 및 라벨(LABEL) 관리
* bookmark: 북마크 관리
* 관련 연료전지 정보(발전소/그룹/셀)
* 북마크별 색상 코드 관리
* 사용자 정의 북마크 이름
* bmk\_sch: 북마크-검색 결과 연결 테이블
* 다대다 관계 매핑 (북마크와 검색 결과 간)
* 북마크에 포함된 검색 결과 관리
* 스택 진단 데이터 특징:
* 임피던스 측정 결과를 시각화하기 위한 좌표 데이터 저장
* 사용자 북마크 기능으로 중요 측정 결과 분류 및 관리
* 오류 코드 및 메모를 통한 진단 결과 해석 지원
* 복합적인 진단 지표(SQ, BQ, CI, STM 등)를 통한 연료전지 상태 평가

## API 명세 및 Redis 키 구조 문서 위치

* MS Teams > Fuelcell\_Dr > API 명세
* MS Teams > Fuelcell\_Dr > 레디스 키

\* 최신 API 및 Redis 키 정보는 위 Teams 경로에서 확인 가능합니다.

\* 문서 접근 권한이 필요한 경우 관리자에게 문의하세요.

# 시스템 구조 및 파일 설명

## 전체 폴더 구조

├── **/home/nstek/h2\_system/patch\_active/FDC/Proj/TRUNK**

├── common\_modules.php        # 언어 선택, 콘솔로그 관리  모듈

├── navbar.php               # 메인 네비게이션 바 컴포넌트(연료전지 레벨 네비바)

├── primary\_navbar.php       # 상단 네비게이션 바 컴포넌트(발전소 레벨 네비바)

├── index-front.html         # 발전소 대시보드 페이지

├── admin.html               # 관리자 설정 페이지

├── index.html               # 대시보드 페이지

├── bop.html                 # BOP 진단 페이지

├── stack.html               # 스택 진단 페이지

├── aitraining.html          # AI 학습 관리 페이지(개발 중단됨)

│

├── **js**

│   ├── **config**

│   │   ├── [기본 설정]

│   │   ├── config.php # 기본 설정 파일 (DB, 경로, 로그)

│   │   ├── db\_config.php # MySQL 데이터베이스 연결 설정

│   │   ├── update\_db\_config.php # 데이터베이스 선택 업데이트

│   │

│   │   ├── [핵심 모듈]

│   │   ├── fuelcellSelector.js # 연료전지 선택 및 관리 핵심 모듈

│   │   │   - URL 파라미터 처리

│   │   │   - 선택기 UI 관리

│   │   │   - 사이트 구조 로딩

│   │   ├── redis\_dataManager.js # index.html Redis 연료전지 데이터 관리 모듈

│   │   │   - {발전소ID}{연료전지ID}\_dash\_web 형식 섹션 처리

│   │   │   - 실시간 데이터 로딩

│   │   ├── redis\_config.php  # Redis 연결 및 데이터 처리 설정

│   │

│   │   ├── [데이터 조회]

│   │   ├── get\_redis\_data.php # Redis 데이터 조회 처리

│   │   ├── get\_site\_structure.php # 전체 사이트 구조 데이터 조회

│   │   ├── get\_facility\_name.js # 발전소, 그룹, 연료전지 이름 조회

│   │   │   - 화면에 표시할 시설 이름 포맷팅

│   │   ├── get\_powerplant\_info.php # 발전소 정보 조회

│   │   ├── get\_group\_info.php # 그룹 상세 정보

│   │   ├── get\_fuelcell\_info.php # 연료전지 상세 정보

│   │   ├── get\_model\_groups.php # 모델 그룹 조회

│   │

│   │   ├── [세션 관리]

│   │   ├── get\_session.js # 세션 관리 클라이언트 모듈

│   │   ├── session\_redis.php # Redis 세션 데이터 조회

│   │

│   │   ├── [유틸리티]

│   │   ├── modelSelector.js # 모델 그룹 선택 관리

│   │   │   - localStorage: selectedModelGroup

│   │   ├── i18n.js # 다국어 처리 모듈

│   │   │   - 언어 전환 기능

│   │   │

│   ├── **locales**                # 다국어 지원 리소스 디렉토리

│   │   ├── en.json           # 영어 번역 파일

│   │   └── ko.json           # 한국어 번역 파일

│   │

│   ├── **index-front**                # index-front.html(발전소 대시보드) 페이지 모듈

│   │   ├── facility\_status.js #설비 현황

│   │   ├── fuelcell\_group\_list.js # 연료전지 그룹 리스트

│   │   ├── fuelcell\_list.js     # 연료전지 리스트

│   │   ├── event\_list.js        # 알람 로그

│   │   ├── get\_alarm\_log.php # 알람 로그 조회

│   │   ├── realtime\_graph.js # 실시간 그래프

│   │

│   ├── **setting**                   # admin.html(관리자 설정) 페이지 모듈

│   │   │

│   │   ├── [공통 설정]

│   │   ├── setting\_common.js     # 공통 설정 관리(관리자 페이지 모든 조회함수 포함)

│   │   ├── get\_users.php         # 전체 사용자 목록

│   │   ├── get\_powerplants.php   # 발전소 목록 조회

│   │   ├── get\_groups.php    # 그룹 목록 조회

│   │   ├── get\_fuelcell\_groups.php #  연료전지-그룹 관계 목록 조회

│   │   ├── get\_fuelcells.php     # 연료전지 목록 조회

│   │   ├── get\_fuelcells\_by\_powerplant.php # 발전소별 연료전지 조회

│   │   ├── pagination.php        # 페이지네이션 처리

│   │   ├── redis\_admin\_update.php  # 발전소, 그룹, 연료전지 CRUD 발생시 대시보드 데이터 업데이트 요청

│   │   │

│   │   ├── [발전소 관리]

│   │   ├── setting\_powerplant.js # 발전소 설정 관리

│   │   ├── add\_powerplants.php   # 발전소 추가

│   │   ├── update\_powerplant.php # 발전소 수정

│   │   ├── delete\_powerplant.php # 발전소 삭제

│   │   │

│   │   ├── [그룹 관리]

│   │   ├── setting\_group.js  # 그룹 설정 관리

│   │   ├── add\_groups.php    # 그룹 추가

│   │   ├── update\_group.php  # 그룹 수정

│   │   ├── delete\_group.php  # 그룹 삭제

│   │   ├── connect\_group\_fuelcells.php # 그룹-연료전지 연결 위젯

│   │   ├── get\_groups\_by\_powerplant.php # 발전소별 그룹 목록

│   │   │

│   │   ├── [연료전지 관리]

│   │   ├── setting\_fuelcell.js   # 연료전지 설정 관리

│   │   ├── add\_fuelcells.php     # 연료전지 추가

│   │   ├── update\_fuelcell.php   # 연료전지 수정

│   │   ├── delete\_fuelcell.php   # 연료전지 삭제

│   │   │

│   │   ├── [사용자 관리]

│   │   ├── setting\_user.js       # 사용자 설정 관리

│   │   ├── add\_user.php          # 일반 사용자 추가

│   │   ├── add\_local\_admin.php   # 로컬 관리자 추가

│   │   ├── add\_local\_user.php    # 로컬 사용자 추가

│   │   ├── update\_user.php       # 사용자 정보 수정

│   │   ├── delete\_user.php       # 사용자 삭제

│   │   ├── search\_user.php       # 사용자 검색

│   │

│   ├── **dashboard**                  # index.html(연료전지 대시보드) 페이지 관련 모듈

│   │   │

│   │   ├── [연료전지 선택기]

│   │   ├── ../config/fuelcellSelector.js  # 연료전지 선택 관리

│   │   │

│   │   ├── [Redis 데이터 파일]

│   │   ├── ../config/redis\_dataManager.js   # Redis 데이터 관리

│   │   │

│   │   ├── [Redis 데이터 받아와 보여주는 위젯]

│   │   ├── dash\_eventManager.js  # 대시보드 전체 위젯 이벤트 관리

│   │   ├── dash\_system\_info.js   # 시스템 정보

│   │   ├── dash\_operation\_info.js # 운전 정보 표시

│   │   ├── dash\_operation\_rate.js # 발전량/가동률 정보

│   │   ├── dash\_realtime\_production.js # 실시간 생산량

│   │   ├── dash\_bar\_chart.js     # 전기/열 생산량 막대 차트

│   │   ├── dash\_bopDiagram.js    # BOP 다이어그램

│   │   ├── dash\_alarm.js         # 알람 로그

│   │   ├── load\_alarm.php        # 알람 데이터 로드

│   │   ├── dash\_hwAlert.js       # HW 센서정보

│   │   ├── dash\_faultDiag.js     # 고장 진단

│   │   ├── dash\_qoe.js           # Soh

│   │

│   ├── **bop**                        # bop.html 페이지 관련 모듈

│   │   │

│   │   ├── [연료전지 선택기]

│   │   ├── ../config/fuelcellSelector.js  # 연료전지 선택 관리

│   │   │

│   │   ├── [모델 그룹 선택 위젯]

│   │   ├── ../config/modelSelector.js  # 모델링 그룹 선택

│   │   ├── modelName.js               # 현재 적용 모델

│   │   ├── get\_model\_apply.php        # 모델 적용 상태 조회

│   │   │

│   │   ├── [BOP 진단 적용 위젯]

│   │   ├── bopDiagApply.js           # BOP 진단 적용

│   │   ├── get\_model\_status.php       # 모델 상태 조회

│   │   ├── update\_DB\_bop\_diag\_apply.php    # DB 진단 결과 저장

│   │   ├── update\_Redis\_bop\_diag\_apply.php # Redis 진단 결과 저장

│   │   │

│   │   ├── [정상 데이터 학습 위젯]

│   │   ├── normalDataTraining.js      # 정상 데이터 학습

│   │   │

│   │   ├── [BOP 정상 학습 데이터 위젯]

│   │   ├── bopNormalData.js          # BOP 정상 학습 데이터 관리 핵심 모듈

│   │   ├── get\_normal\_data.php        # 정상 데이터 목록 조회

│   │   ├── add\_normal\_data\_time.php   # 시간 범위 정상 데이터 추가

│   │   ├── delete\_normal\_data.php     # 정상 데이터 삭제

│   │   ├── check\_file\_exists.php      # 파일명 중복 검사

│   │   ├── get\_normal\_data\_file\_list.php # LST 파일 목록 조회

│   │   ├── add\_new\_file\_name.php      # 새 LST 파일 데이터 추가

│   │   ├── append\_lst\_into\_data.php   # LST 파일 분석하여 시간 데이터 추출

│   │   │

│   │   ├── [BOP 센서 라벨링 데이터 위젯]

│   │   ├── bopSensorLabelingData.js   # BOP 센서 라벨링 데이터

│   │   ├── get\_event.php             # 이벤트 데이터 조회

│   │   ├── bop\_HWSensorList.js       # HW BOP 센서 리스트

│   │   ├── add\_sensor\_labeling\_data.php    # 레이블링 데이터 추가

│   │   ├── delete\_sensor\_labeling\_data.php # 레이블링 데이터 삭제

│   │   ├── get\_sensor\_labeling\_data.php    # 레이블링 데이터 조회

│   │   ├── get\_sensor\_labeling\_data\_by\_id.php # ID별 레이블링 조회

│   │   ├── update\_sensor\_labeling\_data.php    # 레이블링 데이터 수정

│   │   │

│   │   ├── [기타]

│   │   ├── bop\_soh.js                # Soh 위젯

│   │   ├── get\_bop\_soh.php          # Soh 데이터 조회

│   │

│   │

│   ├── **stack**                      # stack.html 페이지 관련 모듈

│   │   │

│   │   ├── [연료전지 선택기]

│   │   ├── ../config/fuelcellSelector.js  # 연료전지 선택 관리

│   │   │

│   │   ├── [SIN/PULSE 검색 위젯]

│   │   ├── stack\_search.js            # 스택 데이터 검색/표시

│   │   ├── stack\_search.php           # 검색 데이터 처리

│   │   ├── copyFileForGraph.php       # 그래프용 파일 복사

│   │   ├── update\_bigo.php            # 비고 업데이트

│   │   ├── update\_error\_code.php      # 에러 코드 업데이트

│   │   ├── stack\_label\_update.php     # 라벨 업데이트

│   │   ├── delete\_files\_in\_selected.php  # 선택 파일 삭제

│   │   ├── delete\_search\_data.php     # 검색 데이터 삭제

│   │   ├── delete\_selected\_file.php   # 선택 파일 삭제

│   │   ├── get\_pulse\_name.php         # 펄스 이름 조회

│   │   ├── saveMaxValue.php           # 최대값 저장

│   │   ├── get\_data\_detail.php         # 상세 데이터 조회

│   │   │

│   │   ├── [SIN/PULSE 위젯 -  태그 관리]

│   │   ├── stack\_tag\_manager.js        # 태그 관리

│   │   ├── save\_tag.php               # 태그 저장

│   │   ├── load\_tags.php              # 태그 로드

│   │   ├── delete\_tag.php             # 태그 삭제

│   │   ├── update\_tag\_order.php       # 태그 순서 업데이트

│   │   │

│   │   ├── [SIN/PULSE 위젯 - 북마크 관리]

│   │   ├── add\_bookmark.php           # 북마크 추가

│   │   ├── delete\_bookmark.php        # 북마크 삭제

│   │   ├── get\_bookmark.php           # 북마크 조회

│   │   ├── update\_bookmark.php        # 북마크 수정

│   │   ├── save\_bookmark\_data.php     # 북마크(북마크-항목 매핑)를 데이터베이스 저장

│   │   ├── copyFileForBookmark.php    # 실제 파일을 북마크 디렉토리에 복사

│   │

│   │   ├── [유닛 컨트롤 위젯]

│   │   ├── unitControl.js             # 유닛 제어 UI

│   │   ├── unitControl\_redis.php      # Redis 데이터 처리

│   │   ├── search\_updateEIS.js        # EIS 데이터 업데이트

│   │   ├── get\_latest\_data.php        # 최신 데이터 조회

│   │   │

│   │   ├── [AVHA 전료/전압/수소/공기 현황 위젯]

│   │   ├── AVHA\_state.js              # 전류/전압/수소/공기 상태

│   │   ├── get\_AVHA\_value.php         # AVHA 값 조회

│   │   │

│   │   ├── [기타]

│   │   ├── stack\_diagnosis\_log.js      # 진단 로그 위젯

│   │   ├── stack\_stack\_diagnosis.js    # 스택 진단 위젯

│   │   ├── get\_bop\_date\_on\_search.php    # BOP-STACK 페이지 연동

│   │   ├── stack\_dataManager.js        # 진단로그 및 스택진단 데이터 관리

│   │   ├── stack\_eventManager.js       # 스택 이벤트 처리

│   │   ├── color\_map.php             # 색상 맵 조회

│   │

│   ├── **aitraining**                    # aitraining.html  관련 모듈(개발 중단됨)

│   │   │

│   │   ├── ai\_dataManager.js          # AI 데이터 관리

│   │   ├── ai\_eventManager.js         # AI 이벤트 관리

│   │   ├── get\_model\_data.php         # 모델 데이터 조회

│   │   ├── get\_progress.php           # 진행상태 조회

│   │   ├── fault\_label\_list.js        # 학습 모델 검증 - 고장 라벨링 리스트

│   │   ├── faultDiagnosisData.php     # 고장 진단 데이터

│   │   ├── get\_fault\_label\_list.php   # 고장 레이블 조회

│   │   ├── hw\_sensor\_table\_list.js    # HW 센서 테이블

│   │   ├── sensorList.js              # 고장 진단 리스트, 로스프 센서 리스트

│   │   ├── get\_hw\_sensor\_table\_list.php # 센서 테이블 조회

│   │   ├── check\_file\_exists.php      # 파일 존재 확인

## 주요 파일 및 폴더 상세 설명

.

├── **/home/nstek/h2\_system/patch\_active/FDC/Proj/TRUNK**

├── common\_modules.php        # 공통 모듈 파일

│   - 언어 선택기 (한국어/영어) 구현

│   - 세션 관리 및 시설 이름 가져오기 기능

│   - i18n (다국어) 초기화 및 언어 변경 기능

│   - 콘솔 로깅 설정 (개발/프로덕션 모드 true/false)

│

├── navbar.php               # 메인 네비게이션 바 구현(연료전지 레벨 네비바)

│   - 대시보드, BOP, Stack 등 주요 페이지 네비게이션

│   - 발전소, 그룹, 연료전지 선택 드롭다운 메뉴

│   - 페이지 이동 및 URL 파라미터 관리

│   - 동적 네비게이션 상태 관리

│

├── primary\_navbar.php       # 상단 네비게이션 바 구현(발전소 레벨 네비바)

│   - 발전소 연료전지 관제 시스템 타이틀 표시

│   - 단순화된 네비게이션 (대시보드, 관리자 페이지)

│   - 연료전지 선택기 및 언어 선택 기능

│   - 세션 및 사용자 인증 관리

│

├── index-front.html         #  발전소 대시보드 페이지

│   - 실시간 성능 데이터 표시

│   - 알람 로그 관리

│   - 연료전지 그룹 및 목록 표시

│

├── admin.html               # 관리자 설정 페이지

│   - 사용자 관리

│   - 시스템 설정

│   - 권한 관리

│

├── index.html               # 대시보드 페이지

│   - 실시간 운영 정보

│   - 시스템 상태 표시

│   - 성능 지표 모니터링

│

├── bop.html                 # BOP 진단 페이지

│   - BOP 센서 데이터 분석

│   - 정상 데이터 학습

│   - 센서 레이블링

│

├── stack.html               # 스택 진단 페이지

│   - 스택 성능 분석

│   - 진단 로그 관리

│   - 북마크 및 태그 관리

│

├── aitraining.html          # AI 학습 관리 페이지(개발 중단됨)

│   - 고장 레이블 관리

│   - 센서 데이터 학습

│   - 모델 상태 관리

│

├── **js**

│   ├── **config**

│   │   ├── [기본 설정]

│   │   ├── config.php        # 기본 설정 파일

│   │   │   - 데이터베이스 접속 정보

│   │   │   - 파일 시스템 경로 설정

│   │   │   - 로그 파일 경로 설정

│   │   │   - 세션 기반 DB 선택

│   │   ├── db\_config.php        # MySQL 데이터베이스 연결 설정

│   │   │   - PDO 기반 데이터베이스 연결

│   │   │   - 세션 기반 동적 데이터베이스 선택

│   │   │   - 에러 처리 및 로깅

│   │   ├── update\_db\_config.php # 데이터베이스 선택 업데이트

│   │   │   - 회사별 데이터베이스 전환

│   │   │   - 세션에 현재 데이터베이스 저장

│   │   │

│   │   ├── [핵심 모듈]

│   │   ├── fuelcellSelector.js  # 연료전지 선택 및 관리 핵심 모듈

│   │   │   - 사이트 구조 관리

│   │   │   │   ∙ 발전소/그룹/연료전지 계층 구조 관리

│   │   │   │   ∙ 세션 기반 사용자별 발전소 접근 권한 처리

│   │   │   │

│   │   │   - 연료전지 설정 관리

│   │   │   │   ∙ 전역 설정 객체 (fuelcellConfig) 관리

│   │   │   │   ∙ 기본 연료전지 설정

│   │   │   │   ∙ 연료전지별 섹션 ID 생성 및 관리

│   │   │   │

│   │   │   - URL 파라미터 처리

│   │   │   │   ∙ plant/group/fuelcell 파라미터 처리

│   │   │   │   ∙ URL 기반 상태 관리

│   │   │   │   ∙ 히스토리 관리

│   │   │   │

│   │   │   - 이벤트 시스템

│   │   │   │   ∙ fuelcellChanged 이벤트

│   │   │   │   ∙ dataLoaded 이벤트

│   │   │   │   ∙ fuelcellConfigLoaded 이벤트

│   │   │   │

│   │   │   - 데이터 동기화

│   │   │   │   ∙ Redis 데이터 관리자와 연동

│   │   │   │   ∙ 실시간 데이터 업데이트

│   │   │   │   ∙ 모델 이름 업데이트

│   │   │

│   │   ├── redis\_dataManager.js   # index.html Redis 연료전지 데이터 관리 모듈

│   │   │   - Redis 키 패턴: {발전소ID}\_{연료전지ID}\_dash\_web

│   │   │   - 연료전지 실시간 데이터 처리

│   │   │   │   ∙ stack\_info

│   │   │   │   ∙ realtime\_production

│   │   │   │   ∙ real\_per\_production

│   │   │   │   ∙ operation\_rate

│   │   │   │   ∙ e\_production, t\_production

│   │   │   │   ∙ e\_capacity, e\_operation

│   │   │   │   ∙ result\_event

│   │   │   │   ∙ BOP

│   │   │   │   ∙ HW\_Alert

│   │   │

│   │   ├── redis\_config.php  # Redis 연결 및 데이터 처리 설정

│   │   │   - CORS 설정 및 헤더 관리

│   │   │   - Redis 서버 연결 (127.0.0.1:6379)

│   │   │   - 섹션별 데이터 조회

│   │   │   - JSON 응답 처리

│   │   │   - 에러 핸들링 및 로깅

│   │   │

│   │   ├── [데이터 조회]

│   │   ├── get\_redis\_data.php     # Redis 데이터 조회 처리

│   │   │   - Redis 키 패턴: {key} (URL 파라미터로 전달)

│   │   │   - JSON 데이터 유효성 검증

│   │   │   - 에러 처리 및 응답 포맷팅

│   │   │

│   │   ├── get\_site\_structure.php       # 전체 사이트 구조 데이터 조회

│   │   │   - 발전소/그룹/연료전지 계층 구조

│   │   │   - 사용자 권한 기반 필터링

│   │   │   - fuelcellConfig 데이터 구성

│   │   │

│   │   ├── get\_facility\_name.js         # 시설 이름 관리

│   │   │   - 표시 이름 포맷팅

│   │   │

│   │   ├── get\_powerplant\_info.php      # 발전소 정보 조회

│   │   │   - 발전소 기본 정보

│   │   │   - 위치, 용량 등 상세 정보

│   │   │

│   │   ├── get\_group\_info.php           # 그룹 상세 정보

│   │   │   - 그룹 상세 구성 정보

│   │   │   - 소속 연료전지 목록

│   │   │

│   │   ├── get\_fuelcell\_info.php        # 연료전지 상세 정보

│   │   │   - 연료전지 상세 스펙

│   │   │   - 운영 상태 정보

│   │   │

│   │   ├── get\_model\_groups.php         # 모델 그룹 정보 조회

│   │   │   - URL 파라미터: powerplant\_id, fuelcell\_id

│   │   │   -/FDC/{$powerplant\_id}/{$fuelcell\_id}/BOP/MODEL/

│   │   │   - 해당 연료전지의 모델 그룹 목록 반환

│   │   │

│   │   ├── [세션 관리]

│   │   ├── get\_session.js         # 세션 관리 클라이언트 모듈

│   │   │   - 쿠키 키: SESS\_ID

│   │   │   - 세션 데이터 조회 및 관리:

│   │   │   │   ∙ company\_id (회사 DB 선택에 사용)

│   │   │   │   ∙ account\_role (권한 관리)

│   │   │   │   ∙ account\_powerplants (접근 가능 발전소 목록)

│   │   │   - URL 파라미터 'plant' 처리

│   │   │   - localStorage: selected\_powerplant

│   │   ├── session\_redis.php      # Redis 세션 데이터 조회

│   │   │   - Redis 키 패턴: session:{session\_hash}

│   │   │   - 세션 데이터 구조:

│   │   │   │   ∙ company\_id

│   │   │   │   ∙ account\_id

│   │   │   │   ∙ account\_role

│   │   │   │   ∙ session\_ipaddr

│   │   │   │   ∙ session\_created\_at

│   │   │

│   │   ├── [유틸리티]

│   │   ├── modelSelector.js             # 모델 그룹 선택 관리

│   │   │   - 모델 그룹 선택 UI 관리

│   │   │   - localStorage: selectedModelGroup

│   │   │   - 이벤트 처리:

│   │   │   │   ∙ modelGroupChanged (모델 그룹 변경)

│   │   │   │   ∙ fuelcellChanged (연료전지 변경)

│   │   │   - 데이터 연동:

│   │   │   │   ∙ 연료전지별 모델 그룹 로드

│   │   │   │   ∙ 선택된 모델 그룹 데이터 로드

│   │   │   - 자동 초기화 및 상태 복원

│   │   │

│   │   ├── i18n.js           # 다국어 처리 모듈

│   │   │   - 언어 리소스 로드 (loadLocale)

│   │   │   - 번역 적용 (applyTranslations)

│   │   │   - 초기 언어 설정 (initI18n)

│   │   │   - 언어 변경 처리 (changeLanguage)

│   │   │   - 로컬 스토리지 기반 언어 설정 저장

│   │

│   ├── **locales**                # 다국어 지원 리소스 디렉토리

│   │   ├── en.json           # 영어 번역 파일

│   │   └── ko.json           # 한국어 번역 파일

│   │

│   ├── **index-front**             # index-front.html(발전소 대시보드) 페이지 모듈

│   │   │   - 모든 실시간 데이터는 'js/config/get\_redis\_data.php'에서 수신

│   │   │   - fuelcellSelector.js의 선택 상태에 따라 데이터 필터링

│   │

│   │   ├── facility\_status.js # 발전소 시설 상태 관리

│   │   │   - 세션에서 발전소 ID 가져오기

│   │   │   - Redis에서 시설 상태 데이터 조회 ({발전소ID}dash\_total\_web)

│   │   │   - 정상/주의/경고/위험/정지 상태별 시설 수 표시

│   │   │   - 5초 간격 자동 업데이트

│   │

│   │   ├── event\_list.js         # 알람 로그 처리

│   │   │   - 알람 상태 실시간 업데이트

│   │   │

│   │   ├── get\_alarm\_log.php     # 알람 로그 데이터 조회

│   │   │   - 데이터베이스 알람 이력 조회

│   │   │   - 시간별 알람 이력 반환

│   │   │

│   │   ├── fuelcell\_group\_list.js # 연료전지 그룹 목록

│   │   │   - get\_group\_list.php로 그룹 정보 조회

│   │   │   - get\_site\_structure.php로 사이트 구조 조회

│   │   │   - 그룹 계층 구조 표시

│   │   │

│   │   ├── fuelcell\_list.js      # 연료전지 목록

│   │   │   - get\_fuelcell\_info.php로 상세 정보 조회

│   │   │   - 연료전지 상태 표시 및 관리

│   │   │   - 목록 필터링 및 정렬

│   │   │   - 클릭 시 상세 페이지 이동 (URL 파라미터 처리)

│   │   │

│   │   ├── realtime\_graph.js     # 실시간 발전량/가동율, 일별, 월별 발전/열생산 현황 그래프 표시

│   │   │   - 성능/생산량 데이터 그래프 렌더링

│   │   │   - 그래프 스케일 및 범례 관리

│   │   │   - 실시간 데이터 시각화

│   │

│   ├── **setting**                   # admin.html(관리자 설정) 페이지 모듈

│   │   │

│   │   ├── [공통 설정]

│   │   ├── setting\_common.js # 공통 설정 관리

│   │   │   - 관리자 페이지 모든 조회함수 포함

│   │   │   - 모달 컨트롤 기능(modalSW)

│   │   │   - 체크박스 리스너 관리

│   │   │   - 테이블 데이터 업데이트 함수

│   │   ├── get\_users.php         # 전체 사용자 목록

│   │   ├── get\_powerplants.php   # 발전소 목록 조회

│   │   ├── get\_groups.php    # 그룹 목록 조회

│   │   ├── get\_fuelcell\_groups.php #  연료전지-그룹 관계 목록 조회

│   │   ├── get\_fuelcells.php     # 연료전지 목록 조회

│   │   ├── get\_fuelcells\_by\_powerplant.php # 발전소별 연료전지 조회

│   │   ├── pagination.php        # 페이지네이션 처리

│   │   ├── redis\_admin\_update.php # 관리 기능에서 데이터 변경 시 Redis 업데이트

│   │   │   - 발전소, 그룹, 연료전지 CRUD 발생 시 대시보드 데이터 갱신

│   │   │   - Redis 키 형식: {발전소ID}dash\_total\_web, {발전소ID}{연료전지ID}dash\_web

│   │   │   - 데이터베이스와 Redis 캐시 간 동기화 처리

│   │   │

│   │   ├── [발전소 관리]

│   │   ├── setting\_powerplant.js # 발전소 설정 관리

│   │   ├── add\_powerplants.php   # 발전소 추가

│   │   ├── update\_powerplant.php # 발전소 수정

│   │   ├── delete\_powerplant.php # 발전소 삭제

│   │   │

│   │   ├── [그룹 관리]

│   │   ├── setting\_group.js  # 그룹 설정 관리

│   │   ├── add\_groups.php    # 그룹 추가

│   │   ├── update\_group.php  # 그룹 수정

│   │   ├── delete\_group.php  # 그룹 삭제

│   │   ├── connect\_group\_fuelcells.php # 그룹-연료전지 연결

│   │   ├── get\_groups\_by\_powerplant.php # 발전소별 그룹 목록

│   │   │

│   │   ├── [연료전지 관리]

│   │   ├── setting\_fuelcell.js   # 연료전지 설정 관리

│   │   ├── add\_fuelcells.php     # 연료전지 추가

│   │   ├── update\_fuelcell.php   # 연료전지 수정

│   │   ├── delete\_fuelcell.php   # 연료전지 삭제

│   │   │

│   │   ├── [사용자 관리]

│   │   ├── setting\_user.js       # 사용자 설정 관리

│   │   ├── add\_user.php          # 일반 사용자 추가

│   │   ├── add\_local\_admin.php   # 로컬 관리자 추가

│   │   │   - LocalAdmin 권한 계정 생성 전용

│   │   │   - 사용자 역할(role)이 "LocalAdmin"일 때 호출됨

│   │   ├── add\_local\_user.php    # 로컬 사용자 추가

│   │   │   - User 권한 계정 생성 전용

│   │   │   - 사용자 역할(role)이 "User"일 때 호출됨

│   │   ├── update\_user.php       # 사용자 정보 수정

│   │   ├── delete\_user.php       # 사용자 삭제

│   │   ├── search\_user.php       # 사용자 검색

│   │

│   ├── **dashboard**                  # index.html(연료전지 대시보드) 관련 모듈

│   │   │

│   │   ├── [연료전지 선택기]

│   │   ├── ../config/fuelcellSelector.js  # 연료전지 선택 관리

│   │   │

│   │   ├── [Redis 데이터 파일]

│   │   ├── ../config/redis\_dataManager.js   # Redis 데이터 관리

│   │   │   - 연료전지 데이터 캐싱 및 실시간 조회

│   │   │   - 섹션 ID 형식: {발전소ID}{연료전지ID}\_dash\_web

│   │   │

│   │   ├── [Redis 데이터 받아와 보여주는 위젯]

│   │   ├── dash\_dataManager.js   # 대시보드 데이터 관리

│   │   │   - redis\_dataManager.js와 연동

│   │   │   - 실시간 데이터 캐싱 및 배포

│   │   │   - 컴포넌트별 데이터 필터링

│   │   │

│   │   ├── dash\_eventManager.js  # 대시보드 이벤트 관리

│   │   │   - 컴포넌트 초기화 관리

│   │   │   - fuelcellChanged 이벤트 처리

│   │   │   - dataLoaded 이벤트 처리

│   │   │

│   │   ├── dash\_qoe.js           # SoH 지수

│   │   │   - 품질 경험 지표 계산

│   │   │   - 성능 데이터 기반 분석

│   │   │

│   │   ├── dash\_operation\_rate.js # 발전량/가동률 정보

│   │   │   - 시간/일간 가동률 계산

│   │   │   - operation\_rate 데이터 사용

│   │   │

│   │   ├── dash\_realtime\_production.js # 실시간 생산량

│   │   │   - 전기/열 생산량 표시

│   │   │   - realtime\_production 데이터 사용

│   │   │

│   │   ├── dash\_bopDiagram.js    # 시스템 구조도

│   │   │   - BOP 상태 다이어그램

│   │   │   - BOP 데이터 시각화

│   │   │

│   │   ├── dash\_bar\_chart.js     # 성능 차트

│   │   │   - 성능 지표 차트 표시

│   │   │   - e\_production, t\_production 데이터 사용

│   │   │

│   │   ├── dash\_alarm.js         # 알람 로그

│   │   │   - load\_alarm.php와 통신

│   │   │   - 실시간 알람 표시

│   │   │

│   │   ├── load\_alarm.php        # 알람 데이터 로드

│   │   │   - Redis 알람 데이터 조회

│   │   │   - 알람 이력 필터링

│   │   │

│   │   ├── dash\_faultDiag.js     # 고장 진단 리스트

│   │   │   - result\_event 데이터 처리

│   │   │   - 고장 진단 결과 표시

│   │   │

│   │   ├── dash\_hwAlert.js       # HW센서정보

│   │   │   - HW\_Alert 데이터 처리

│   │   │   - 하드웨어 상태 모니터링

│   │   │

│   │   ├── dash\_operation\_info.js # 운전 정보 표시

│   │   │   - 운전 상태 및 모드 표시

│   │   │   - redis\_dataManager.js 실시간 데이터 구독

│   │   │

│   │   ├── dash\_system\_info.js   # 시스템 정보 표시

│   │   │   - 연료전지 기본 정보 표시

│   │   │   - get\_fuelcell\_info.php 데이터 사용

│   │   │

│   ├── **bop**                        # bop.html(BOP 진단) 관련 모듈

│   │

│   │   ├── [연료전지 선택기]

│   │   ├── ../config/fuelcellSelector.js  # 연료전지 선택 관리

│   │   │

│   │   ├── [모델 그룹 선택 위젯]

│   │   ├── ../config/modelSelector.js  # 모델 그룹 선택 핵심 모듈

│   │   │   - 발전소/연료전지별 사용 가능한 모델 그룹 목록 로드

│   │   │   - 모델 그룹 선택 UI 관리 및 드롭다운 컨트롤

│   │   │   - localStorage에 선택된 모델 그룹(selectedModelGroup) 저장

│   │   │   - 모델 변경 시 modelGroupChanged 이벤트 발생

│   │   │   - fuelcellChanged 이벤트 감지 시 모델 목록 자동 갱신

│   │   │

│   │   ├── modelName.js               # 현재 적용된 모델 표시 모듈

│   │   │   - Redis에서 '{plant}\_{fuelcell}\_model\_apply' 키 값 조회

│   │   │   - UI에 현재 적용된 모델 이름 실시간 표시

│   │   │   - 5초마다 자동 갱신으로 모델 적용 상태 모니터링

│   │   │   - 연료전지 변경 시 표시 정보 자동 업데이트

│   │   │

│   │   ├── get\_model\_apply.php        # 모델 적용 상태 조회

│   │   │   - Redis에서 현재 적용된 모델 정보 조회

│   │   │   - plant/fuelcell 파라미터 기반 데이터 필터링

│   │   │   - 적용 상태 및 시간 정보 JSON 형식 반환

│   │   │

│   │   ├── [BOP 진단 적용 위젯]

│   │   ├── bopDiagApply.js           # BOP 진단 모델 적용 핵심 관리 모듈

│   │   │   - 진단 모델 선택 및 적용 UI 구현 (DiagModelSelector 클래스)

│   │   │   - 모델 적용 및 학습 시작/중지 기능 제공

│   │   │   - 날짜 범위 선택기로 학습 데이터 기간 설정

│   │   │

│   │   ├── get\_model\_status.php       # 모델 실행 상태 조회

│   │   │   - Redis에서 '{plant}\_{fuelcell}\_model\_status' 키 값 확인

│   │   │   - 모델의 실행 상태(true/false) 반환

│   │   │   - bopDiagApply.js가 5초마다 호출하여 상태 갱신

│   │   │

│   │   ├── update\_DB\_bop\_diag\_apply.php    # DB 모델 적용 정보 저장

│   │   │   - api\_modelapply 테이블에 모델 적용 정보 저장

│   │   │   - 기존 레코드 존재 시 UPDATE, 없을 경우 INSERT 처리

│   │   │   - powerplant\_id, group\_id, fuelcell\_id, model\_name 저장

│   │   │   - PDO 기반 데이터베이스 트랜잭션 처리

│   │   │

│   │   ├── update\_Redis\_bop\_diag\_apply.php # Redis 모델 적용 정보 저장

│   │   │   - '{plant}\_{fuelcell}\_model\_apply' 키에 모델명 저장

│   │   │   - 다른 모듈(modelName.js)이 이 키를 참조하여 UI 업데이트

│   │   │   - 출력 버퍼 제어로 안정적 JSON 응답 제공

│   │   │

│   │   ├── [정상 데이터 학습 위젯]

│   │   ├── normalDataTraining.js      # 정상 데이터 학습 관리 모듈

│   │   │   - 정상 운전 데이터 기반 모델 학습 기능 제공

│   │   │   - 데이터 유형별 처리 (시간 범위 선택 또는 파일 선택)

│   │   │   - 선택된 데이터를 API 서버로 전송하여 학습 요청

│   │   │   - 날짜와 시간 포맷 처리 및 파일명 생성 로직 구현

│   │   │   - 체크박스 선택 기반의 데이터 수집 및 검증

│   │   │

│   │   ├── [BOP 정상 학습 데이터 위젯]

│   │   ├── bopNormalData.js          # BOP 정상 학습 데이터 관리 핵심 모듈

│   │   │   - 정상 데이터 테이블 초기화 및 이벤트 리스너 설정

│   │   │   - 시간 범위 기반 정상 데이터 추가 기능

│   │   │   - 파일(LST) 추가 및 관리 UI 구현

│   │   │   - 선택된 데이터로 정상 학습 파일 생성 기능

│   │   │   - 체크박스 기반 다중 선택 및 삭제 기능

│   │   │   - API 서버와 통신하여 데이터 병합 요청(nm\_merge)

│   │   │

│   │   ├── get\_normal\_data.php        # 정상 데이터 목록 조회

│   │   │   - 발전소/연료전지 기준 정상 데이터 검색

│   │   │   - type=1(시간 범위) 또는 type=2(파일명) 데이터 필터링

│   │   │   - JSON 형식으로 데이터 반환

│   │   │

│   │   ├── add\_normal\_data\_time.php   # 시간 범위 정상 데이터 추가

│   │   │   - 선택된 날짜/시간 범위 정보 저장

│   │   │   - 중복 데이터 검증 및 오류 처리

│   │   │   - type=1 형식으로 DB 저장

│   │   │

│   │   ├── delete\_normal\_data.php     # 정상 데이터 삭제

│   │   │   - 선택된 항목들의 일괄 삭제 처리

│   │   │   - 데이터 유형(type)에 따른 처리 분기

│   │   │   - 트랜잭션 기반 안전한 삭제 처리

│   │   │

│   │   ├── check\_file\_exists.php      # 파일명 중복 검사

│   │   │   - 정상 데이터 생성 시 파일명 중복 체크

│   │   │   - BOP/MODEL 디렉토리 내 파일 존재 여부 확인

│   │   │   - 파일시스템 및 데이터베이스 기반 검증

│   │   │

│   │   ├── get\_normal\_data\_file\_list.php # LST 파일 목록 조회

│   │   │   - 발전소/연료전지별 사용 가능한 LST 파일 목록 제공

│   │   │   - 파일 추가 및 부가 모달에서 사용

│   │   │   - 디렉토리 기반 파일 목록 수집

│   │   │

│   │   ├── add\_new\_file\_name.php      # 새 LST 파일 데이터 추가

│   │   │   - 선택된 LST 파일들을 데이터베이스에 등록

│   │   │   - type=2 형식으로 파일명만 DB 저장

│   │   │   - 중복 파일명 검사 및 오류 처리

│   │   │

│   │   ├── append\_lst\_into\_data.php   # LST 파일 분석하여 시간 데이터 추출

│   │   │   - LST 파일 내용 분석하여 날짜/시간 정보 추출

│   │   │   - 중복 검사 후 새 시간 범위만 데이터베이스에 추가

│   │   │   - 추가된 항목과 중복 항목 목록 반환

│   │   │

│   │   ├── [BOP 센서 라벨링 데이터 위젯]

│   │   ├── [BOP 센서 라벨링 시스템]

│   │   ├── bopSensorLabelingData.js   # BOP 센서 라벨링 데이터 관리 모듈

│   │   │   - 센서 오류 코드 매핑(errorCodeMapping) 객체 제공

│   │   │   - 라벨링 데이터 목록 표시 및 페이지네이션 처리

│   │   │   - 시간 범위 기반 센서 라벨링 데이터 추가/수정/삭제

│   │   │   - 날짜 클릭 시 이벤트 데이터 로드 및 캘린더 연동

│   │   │   - 모달 기반의 상세 정보 보기 및 수정 인터페이스

│   │   │

│   │   ├── get\_sensor\_labeling\_data.php    # 라벨링 데이터 목록 조회

│   │   │   - 발전소/연료전지 기준 라벨링 데이터 검색

│   │   │   - 페이지네이션 지원(현재 페이지, 항목 수 파라미터)

│   │   │   - 에러 코드 기반 필터링 기능

│   │   │   - JSON 형식으로 데이터 및 페이지 정보 반환

│   │   │

│   │   ├── get\_sensor\_labeling\_data\_by\_id.php # ID별 라벨링 상세 조회

│   │   │   - 특정 ID의 라벨링 데이터 상세 정보 제공

│   │   │   - 시작/종료 시간, 에러 코드, 온습도, 설명 등 포함

│   │   │   - 모달에서 상세 정보 표시 시 사용

│   │   │

│   │   ├── add\_sensor\_labeling\_data.php    # 라벨링 데이터 추가

│   │   │   - 시간 범위, 에러 코드, 온습도, 설명 정보 저장

│   │   │   - 발전소/연료전지 정보와 함께 데이터베이스에 저장

│   │   │   - 중복 검사 및 데이터 유효성 검증

│   │   │

│   │   ├── update\_sensor\_labeling\_data.php    # 라벨링 데이터 수정

│   │   │   - 기존 라벨링 데이터의 모든 필드 업데이트

│   │   │   - 에러 코드, 시간 범위, 온습도, 설명 수정 지원

│   │   │   - 트랜잭션 기반 안전한 데이터 업데이트

│   │   │

│   │   ├── delete\_sensor\_labeling\_data.php # 라벨링 데이터 삭제

│   │   │   - 선택된 라벨링 데이터 ID 기반 삭제

│   │   │   - 단일 또는 다중 항목 삭제 지원

│   │   │   - 삭제 결과 및 성공/실패 메시지 반환

│   │   │

│   │   ├── get\_event.php              # 이벤트 데이터 조회

│   │   │   - 특정 날짜(년/월/일)의 이벤트 데이터 검색

│   │   │   - 발전소/연료전지 기준 필터링

│   │   │   - API 이벤트(type=0)와 검색 이벤트(type=1) 구분 제공

│   │   │   - CSV 형식으로 데이터 반환 (행 단위 이벤트 정보)

│   │   │

│   │   ├── bop\_HWSensorList.js       # HW BOP 센서 리스트

│   │   │   - 하드웨어 센서 목록 UI

│   │   │   - 센서 상태 모니터링

│   │   │   - 센서 선택 기능

│   │   │

│   │   ├── [기타]

│   │   ├── bop\_soh.js                # Soh 위젯

│   │   ├── get\_bop\_soh.php          # Soh 데이터 조회

│   │

│   ├── **stack**                      # stack.html 페이지 관련 모듈

│   │   │

│   │   ├── [연료전지 선택 위젯]

│   │   ├── ../config/fuelcellSelector.js  # 연료전지 선택 관리

│   │   │   - 연료전지 선택 UI 컴포넌트 관리

│   │   │   - localStorage에 선택된 연료전지 정보 저장

│   │   │   - fuelcellChanged 이벤트 발생 및 처리

│   │   │

│   │   ├── [SIN/PULSE 검색 위젯]

│   │   ├── stack\_search.js            # 스택 데이터 검색/표시 (핵심 모듈)

│   │   │   - SIN/PULSE 모드 전환 및 각 모드별 UI 관리

│   │   │   │   ∙ initializeTypeSelector(): 검색 유형 UI 초기화

│   │   │   │   ∙ updateEisOptions(): 선택된 타입에 따라 검색 옵션 변경

│   │   │   │

│   │   │   - 고급 검색 기능

│   │   │   │   ∙ initializeSearchConditions(): 검색 조건 초기화

│   │   │   │   ∙ getSearchDateRange(): 날짜 범위 처리

│   │   │   │   ∙ 필드별 비교 조건(over/under) 지원

│   │   │   │   ∙ resetSearchConditions(): 검색 조건 초기화

│   │   │   │

│   │   │   - 데이터 처리 및 표시

│   │   │   │   ∙ fetchData(): API 호출 및 결과 가져오기

│   │   │   │   ∙ displayResults(): 검색 결과 테이블 렌더링

│   │   │   │   ∙ updateTable(): 테이블 데이터 동적 업데이트

│   │   │   │   ∙ displayPagination(): 페이지네이션 UI 구현

│   │   │   │

│   │   │   - 그래프 관련 기능

│   │   │   │   ∙ copyFilesForGraph(): 그래프 데이터 파일 복사 처리

│   │   │   │   ∙ handleFileOperations(): 파일 작업 처리

│   │   │   │   ∙ saveMaxValueToFile(): 최대값 설정 저장

│   │   │   │   ∙ clearSelectedDirectory(): 임시 파일 폴더 정리

│   │   │   │

│   │   │   - 북마크 및 데이터 관리

│   │   │   │   ∙ getBookmarkTabs(): 북마크 탭 로드

│   │   │   │   ∙ handleTabClick(): 탭 전환 처리

│   │   │   │   ∙ addBookmark(), deleteBookmarks(): 북마크 CRUD

│   │   │   │   ∙ handleDeleteItems(): 선택된 아이템 삭제

│   │   │   │   ∙ updateBigoInDatabase(): 비고 필드 실시간 업데이트

│   │   │   │

│   │   │   - 이벤트 처리 시스템

│   │   │   │   ∙ setupEventListeners(): 전체 이벤트 리스너 등록

│   │   │   │   ∙ initCheckboxStateAndSelectAll(): 체크박스 상태 관리

│   │   │   │   ∙ handleSelectAllChange(): 전체 선택 처리

│   │   │   │   ∙ handleDateCellClick(): 날짜 셀 클릭 이벤트 처리

│   │   │

│   │   ├── stack\_search.php           # 검색 데이터 처리

│   │   │   - SIN/PULSE 데이터 검색 처리

│   │   │   - GET 파라미터: page, perPage, powerplant\_id, fuelcell\_id, type(SIN/PULSE), bookmarkId

│   │   │   - 페이지네이션 지원(기본 100건)

│   │   │   - 검색 조건 지원: 날짜 범위, 라벨, 비고(#키워드 구분), 오류코드(MERR)

│   │   │   - 필드별 상세조건검색 지원(over/under)

│   │   │

│   │   ├── copyFileForGraph.php       # 그래프용 파일 복사

│   │   │   - 검색 결과 파일을 그래프용 디렉토리로 복사

│   │   │   - 파일명에 색상코드 추가 처리

│   │   │   - 검색 테이블의 좌표 데이터(X1, X2, Y1, Y2) 함께 반환

│   │   │   - 원본 파일 경로는 발전소/연료전지 ID와 날짜 기반으로 구성

│   │   │

│   │   ├── update\_bigo.php            # 비고 업데이트

│   │   ├── update\_error\_code.php      # 에러 코드 업데이트

│   │   ├── stack\_label\_update.php     # 라벨 업데이트

│   │   ├── delete\_files\_in\_selected.php  # 그래프용 임시 폴더 초기화

│   │   │   - 그래프 표시를 위해 복사된 모든 파일 삭제

│   │   │   - /home/nstek/h2\_system/patch\_active/FDC/work/bjy/impedance/selected/ 디렉토리 내 파일 제거

│   │   │

│   │   ├── delete\_search\_data.php     # 검색 데이터 삭제

│   │   │   - 북마크/일반 모드별 테이블 삭제 처리

│   │   │   - 선택 항목 영구 제거

│   │   │

│   │   ├── delete\_selected\_file.php   # 그래프 파일 삭제

│   │   │   - 검색 테이블 정보 기반 파일 삭제

│   │   │   - 색상코드 포함 관련 파일 일괄 제거

│   │   │

│   │   ├── saveMaxValue.php           # 그래프 최대값 저장

│   │   │   - 최대값 파라미터 처리 및 저장

│   │   │   - 설정 파일 경로 관리

│   │   │

│   │   ├── get\_data\_detail.php        # 항목 상세 정보 조회

│   │   │   - 번호 기준 데이터 조회

│   │   │   - JSON 형식 결과 반환

│   │   │

│   │   ├── [스택 검색 위젯의 태그 관리]

│   │   ├── stack\_tag\_manager.js        # 태그 관리 UI

│   │   │   - 태그 CRUD 인터페이스

│   │   │   - 태그 드래그앤드롭 정렬

│   │   │   - 태그 필터링 기능

│   │   ├── save\_tag.php               # 태그 저장

│   │   │   - 새 태그 생성

│   │   │   - 태그 정보 업데이트

│   │   ├── load\_tags.php              # 태그 로드

│   │   │   - 저장된 태그 목록 조회

│   │   │   - 태그별 메타데이터 로드

│   │   ├── delete\_tag.php             # 태그 삭제

│   │   │   - 태그 및 연관 데이터 삭제

│   │   │   - 참조 무결성 확인

│   │   ├── update\_tag\_order.php       # 태그 순서 업데이트

│   │   │   - 태그 순서 저장

│   │   │   - 순서 변경 이력 관리

│   │   │

│   │   ├── [스택 검색 위젯의 북마크 관리]

│   │   ├── add\_bookmark.php           # 북마크 추가

│   │   │   - 새 북마크 생성

│   │   ├── delete\_bookmark.php        # 북마크 삭제

│   │   │   - 북마크 제거

│   │   │   - 연관 파일 정리

│   │   ├── get\_bookmark.php           # 북마크 조회

│   │   │   - 북마크 목록 로드

│   │   │   - 상세 정보 조회

│   │   ├── update\_bookmark.php        # 북마크 수정

│   │   ├── save\_bookmark\_data.php     # 북마크(북마크-항목 매핑)를 데이터베이스 저장

│   │   ├── copyFileForBookmark.php    # 북마크용 파일 복사

│   │   │   - 북마크 관련 파일 복제

│   │   │   - 메타데이터 복사

│   │   ├── color\_map.php             # 색상 맵 조회(현재 사용 중단됨)

│   │   │

│   │   ├── [유닛 컨트롤 위젯]

│   │   ├── unitControl.js             # 유닛 제어 UI

│   │   │   - 진단 장비 상태 모니터링

│   │   │   - 다양한 진단 모드(connect, run, pulse) 지원

│   │   │   - 연료전지 변경 시 자동 상태 업데이트

│   │   │

│   │   ├── unitControl\_redis.php      # Redis 데이터 처리

│   │   │   - 게이트웨이 상태 정보 처리

│   │   │   - 데이터 새로고침 시점 추적

│   │   │   │   ∙ {발전소ID}\_{연료전지ID}\_fdu\_prg: 게이트웨이 상태 정보

│   │   │   │   ∙ {발전소ID}\_{연료전지ID}\_fdu\_uploaded: 데이터 새로고침 시점

│   │   │

│   │   ├── search\_updateEIS.js        # 자동 EIS 측정 결과 처리

│   │   │   - Redis를 통한 측정 완료 감지 및 후처리

│   │   │   - EIS 명령 실행 후 자동으로 최신 데이터 체크

│   │   │   - 새 측정 데이터를 SIN 데이터 테이블에서 자동 선택 및 그래프 생성

│   │   ├── get\_latest\_data.php        # 연료전지별 최신 측정 데이터 조회

│   │   │   - 특정 연료전지의 가장 최근 측정값 반환

│   │   │   - search\_updateEIS.js에서 호출되어 자동화 지원

│   │   │   - 새 EIS 측정 완료 시 해당 데이터 식별에 사용

│   │   │

│   │   ├── [AVHA 전료/전압/수소/공기 현황 위젯]

│   │   ├── AVHA\_state.js              # 전류/전압/수소/공기 상태

│   │   │   - 연료전지 주요 운전 파라미터 실시간 모니터링

│   │   │   │   ∙ 전류(Current), 전압(Voltage) 데이터 표시

│   │   │   │   ∙ 수소(MFC1-H2), 공기(MFM3-Air) 유량 정보 시각화

│   │   │

│   │   ├── get\_AVHA\_value.php         # AVHA 값 조회

│   │   │

│   │   ├── [기타/유틸리티]

│   │   ├── stack\_diagnosis\_log.js      # 진단 로그 위젯

│   │   ├── stack\_stack\_diagnosis.js    # 스택 진단 위젯

│   │   ├── get\_bop\_date\_on\_search.php    # BOP 페이지 데이터로 STACK 페이지 그래프 생성

│   │   │   - 타임스탬프 기반 날짜 데이터 검색

│   │   │   - 검색 결과의 순서 정보 제공

│   │   │   - JSON 형식 응답 처리

│   │   │

│   │   ├── stack\_dataManager.js        # 진단로그/스택진단 데이터 관리

│   │   │   - CSV 파일 데이터 로드 및 자동 갱신

│   │   │   - 진단 로그 데이터 파싱 및 변환

│   │   │   - 스택 진단 데이터 구조화

│   │   │

│   │   ├── stack\_eventManager.js       # 스택 이벤트 처리

│   │   │   - 진단 로그 데이터 페이징 표시

│   │   │   - 페이지 네비게이션 인터페이스 관리

│   │   │   - 체크박스 선택 기능 제공

│   │   ├── color\_map.php             # 색상 맵 조회

│   │

│   ├── **aitraining**                         # aitraining.html  관련 모듈(개발 중단됨)

│   │

│   │   ├── [핵심 모듈]

│   │   ├── ../config/stackSelector.js  # 이전 연료전지 선택 관리, 삭제된 파일

│   │   ├── ../config/fuelcellSelector.js  파일 기준으로 리팩토링 필요하지만, 개발 중단돼서 추후 개발계획에 따라 달라짐.

│   │   ├── modelSelector.js     # 모델 그룹 선택 관리 핵심 모듈

│   │   │   - 모델 그룹 선택 UI 관리

│   │   │   - localStorage에 선택된 모델 그룹 저장

│   │   │   - modelGroupChanged 이벤트 발행 및 처리

│   │   │   - 연료전지 변경 시 모델 그룹 목록 자동 갱신

│   │   │

│   │   ├── [학습 모델 검증]

│   │   ├── fault\_label\_list.js        # 고장 라벨 목록 관리 및 모델 검증

│   │   │   - 고장 데이터 목록 표시 및 필터링

│   │   │   - 검증 대상 데이터 선택 및 API 전송

│   │   │   - 날짜 선택 시 그래프 연동

│   │   │   - 에러 코드 선택 이벤트 처리

│   │   │

│   │   ├── get\_fault\_label\_list.php   # 고장 라벨 데이터 조회

│   │   │   - 연료전지별 고장 이벤트 목록 조회

│   │   │   - 모델 그룹별 검증 점수 조회

│   │   │   - 에러 코드 기반 필터링

│   │   │   - 날짜별 정렬 처리

│   │   │

│   │   ├── get\_progress.php           # 검증 진행상태 조회

│   │   │   - Redis에서 검증 진행상황 조회

│   │   │   - 진행률, 완료 항목 수, 상태 정보 제공

│   │   │   - 검증 작업 모니터링 지원

│   │   │

│   │   ├── [센서 데이터 관리 위젯]

│   │   ├── hw\_sensor\_table\_list.js    # 하드웨어 센서 데이터 테이블

│   │   │   - 센서 선택에 따른 데이터 테이블 표시

│   │   │   - 센서별 CSV 파일 로드 및 파싱

│   │   │   - 선택된 센서의 그래프 생성(init\_aitraining\_graph\_2 호출)

│   │   │   - 모델 그룹 및 연료전지 변경 이벤트 처리

│   │   │

│   │   ├── get\_hw\_sensor\_table\_list.php # 센서 데이터 CSV 파일 조회

│   │   │   - 모델 그룹 디렉토리에서 센서별 CSV 파일 로드

│   │   │   - 파일 경로: /home/nstek/h2\_system/FDC/{powerplant\_id}/{fuelcell\_id}/BOP/MODEL/{modelGroup}/{fileName}

│   │   │   - CSV 데이터를 JSON 형식으로 변환하여 반환

│   │   │   - 파일 존재 여부 검증 및 오류 처리

│   │   │

│   │   ├── sensorList.js              # 센서 목록 관리

│   │   │   - 고장진단 결과 표시

│   │   │   - 센서 데이터 표시 및 필터링

│   │   │

│   │   ├── get\_model\_data.php         # 모델 데이터 조회

│   │   │   - 선택된 모델의 상세 정보 조회

│   │   │   - 학습 이력 및 성능 정보 제공

# 주요 기능 및 코드 분석

## 세션 기반 다중 데이터베이스 접근 시스템

세션 기반 인증과 다중 데이터베이스 관리 메커니즘으로, 여러 회사(발전소)의 데이터를 효율적으로 분리하고 접근 권한을 관리

### 통합 작동 원리

1. 사용자 인증:

* 로그인 시 SESS\_ID 쿠키가 브라우저에 저장됨
* 세션 데이터는 Redis에 session:{해시값} 형식으로 저장됨
* 세션 데이터 조회:
* get\_session.js가 쿠키에서 SESS\_ID 추출
* session\_redis.php를 통해 Redis에서 세션 정보 조회
* 세션 정보에는 company\_id, account\_role, account\_powerplants 등이 포함됨

1. 데이터베이스 전환:

* 조회된 company\_id를 기반으로 update\_db\_config.php 호출
* 회사 ID에 맞는 데이터베이스 선택 및 연결 설정
* 세션에 현재 데이터베이스 정보 저장

1. 사이트 구조 로드:

* fuelcellSelector.js가 get\_site\_structure.php 호출
* 현재 데이터베이스에서 발전소/그룹/연료전지 계층 구조 조회
* 사용자 권한에 맞는 발전소만 필터링하여 반환

1. 페이지 상태 동기화:

* URL 파라미터 분석하여 초기 선택 상태 설정
* 사용자 선택에 따라 현재 상태 및 URL 파라미터 업데이트
* fuelcellChanged 이벤트로 모든 컴포넌트에 변경 사항 알림

### 세션 관리 시스템(get\_session.js)

export class SessionService {

// 쿠키에서 세션 해시 가져오기

static getSessionHash() {

const cookies = document.cookie.split(';');

for (let cookie of cookies) {

const [name, value] = cookie.trim().split('=');

if (name === 'SESS\_ID') {

return value;

}

}

console.error('SESS\_ID를 찾을 수 없음. 사용 가능한 쿠키들:', cookies);

throw new Error('세션 ID를 찾을 수 없습니다');

}

// Redis에서 세션 정보 조회

static async getSessionInfo() {

try {

const sessionHash = this.getSessionHash();

const response = await axios.get('js/config/session\_redis.php', {

params: {

session\_hash: sessionHash

}

});

if (!response.data.redis\_connection) {

throw new Error('Redis 연결 실패');

}

if (!response.data.session\_data) {

throw new Error(response.data.error || '세션 데이터를 찾을 수 없습니다');

}

return response.data.session\_data; // 이미 파싱된 세션 데이터

} catch (error) {

console.error('세션 정보 조회 실패:', error);

throw error;

}

}

// 회사 ID 가져오기 및 DB 설정 업데이트

static async getCompanyId() {

try {

const sessionInfo = await this.getSessionInfo();

const companyId = sessionInfo.company\_id;

// company\_id로 DB 설정 업데이트

await axios.post('js/config/update\_db\_config.php', {

company\_id: companyId

});

return companyId;

} catch (error) {

console.error('회사 정보 조회 실패:', error);

throw error;

}

}

// 권한 정보 가져오기

static async getAccountRole() {

try {

const sessionInfo = await this.getSessionInfo();

return sessionInfo.account\_role;

} catch (error) {

console.error('권한 정보 조회 실패:', error);

throw error;

}

}

// 발전소 목록 가져오기

static async getPowerplants() {

try {

const sessionInfo = await this.getSessionInfo();

return sessionInfo.account\_powerplants;

} catch (error) {

console.error('발전소 목록 조회 실패:', error);

throw error;

}

}

// 현재 선택된 발전소 ID 가져오기

static async getCurrentPowerplantId() {

try {

// URL에서 plant 파라미터 확인

const urlParams = new URLSearchParams(window.location.search);

let powerplantId = urlParams.get('plant');

// URL에 없으면 로컬 스토리지에서 확인

if (!powerplantId) {

powerplantId = localStorage.getItem('selected\_powerplant');

}

// 둘 다 없으면 권한 있는 첫 번째 발전소

if (!powerplantId) {

const powerplants = await this.getPowerplants();

powerplantId = powerplants[0];

localStorage.setItem('selected\_powerplant', powerplantId);

}

return powerplantId;

} catch (error) {

console.error('현재 발전소 ID 조회 실패:', error);

throw error;

}

}

}

### Redis 세션 연동(session\_redis.php)

session\_redis.php는 Redis에 저장된 세션 정보를 가져오는 서버 측 스크립트. 클라이언트의 세션 해시를 이용해 Redis에서 세션 데이터를 조회

<?php

header('Content-Type: application/json');

header("Access-Control-Allow-Origin: {$\_SERVER['HTTP\_ORIGIN']}");

header('Access-Control-Allow-Credentials: true');

$response = [

'redis\_connection' => false,

'error' => null,

'session\_data' => null

];

try {

$redis = new Redis();

$redis->connect('127.0.0.1', 6379);

$redis->select(1); // 세션 데이터베이스 선택

$response['redis\_connection'] = true;

if (isset($\_GET['session\_hash'])) {

$sessionKey = 'session:' . $\_GET['session\_hash'];

$sessionData = $redis->get($sessionKey);

if ($sessionData !== false) {

// 세션 데이터가 JSON 형식인지 확인

$decodedData = json\_decode($sessionData, true);

if ($decodedData !== null) {

$response['session\_data'] = $decodedData;

} else {

$response['error'] = 'Invalid session data format';

}

} else {

$response['error'] = 'Session not found';

}

} else {

$response['error'] = 'No session hash provided';

}

} catch (Exception $e) {

$response['error'] = 'Redis connection error: ' . $e->getMessage();

}

echo json\_encode($response);

?>

## 데이터베이스 연결 관리(config.php, update\_db\_config.php, db\_config.php)

### 동작 프로세스

1. 사용자 DB 전환 요청:

* 사용자가 회사/데이터베이스 변경 액션 실행
* 클라이언트에서 AJAX로 update\_db\_config.php에 company\_id 전송

1. 세션 데이터 업데이트:

* update\_db\_config.php가 받은 company\_id를 $\_SESSION['current\_db']에 저장
* 다음 페이지 로드나 API 요청 시 이 세션 값이 사용됨

1. 데이터베이스 연결 설정:

* 각 PHP 파일이 실행될 때 config.php 포함됨
* config.php는 세션에서 저장된 current\_db 값을 읽어 $dbname 변수 설정

만약 세션에 값이 없으면 기본값 'FDC' 사용

1. 실제 DB 연결 생성:

* 필요한 시점에 db\_config.php가 포함됨
* config.php에서 설정된 변수들을 사용하여 PDO 객체 생성

이 PDO 객체가 전환된 데이터베이스에 연결

### 기본 설정(config.php)

$host = 'localhost';

$dbname = $\_SESSION['current\_db'] ?? 'FDC'; // 세션에서 데이터베이스명 가져오거나 기본값 사용

$username = 'root';

$password = 'coffee';

### 데이터베이스 전환 프로세스(update\_db\_config.php)

// AJAX 요청으로 받은 company\_id를 세션에 저장

if (isset($data['company\_id'])) {

$\_SESSION['current\_db'] = $data['company\_id']; // 클라이언트가 선택한 DB를 세션에 저장

}

### 데이터베이스 연결 관리(db\_config.php)

// config.php에서 설정된 변수를 사용하여 실제 데이터베이스 연결 생성

$dsn = 'mysql:host=' . $host . ';dbname=' . $dbname;

try {

$pdo = new PDO($dsn, $username, $password);

// ...

} catch (PDOException $e) {

## 사이트 구조 및 계층관리 시스템(fuelcellSelector.js)

fuelcellSelector.js는 발전소, 그룹, 연료전지의 계층 구조를 관리하는 핵심 모듈. 세션 서비스와 연동하여 사용자가 접근 가능한 발전소 및 연료전지를 로드하고 관리.

### 동작 프로세스

1. **초기화 과정**:

* 페이지 로드 시 SessionService를 통해 사용자가 접근 가능한 발전소 목록 가져옴
* get\_site\_structure.php를 호출하여 발전소/그룹/연료전지 계층 구조 로드
* 로드된 데이터를 siteStructure와 fuelcellConfig 변수에 저장

1. **선택 흐름 관리**:

* URL 파라미터 > 로컬 스토리지 > 기본값 순으로 우선순위 적용
* 선택된 연료전지 정보는 powerplant\_id, group\_id, fuelcell\_id, name 형태로 관리
* 잘못된 선택 또는 오류 발생 시 기본 연료전지로 대체

1. **UI 연동**:

* 발전소/그룹/연료전지 드롭다운 메뉴 동적 구성
* 계층별 선택 변경 시 하위 항목 자동 업데이트
* 선택 변경 시 fuelcellChanged 이벤트 발생으로 연동된 모든 컴포넌트에 통지

1. **데이터 및 상태 동기화**:

* 연료전지 변경 시 loadAllData()로 새 데이터 로드
* URL 파라미터 업데이트로 페이지 새로고침 시에도 상태 유지
* 로컬 스토리지에 선택 상태 저장으로 세션 간 연속성 제공

### 계층 구조 초기화 및 로드

// 동적으로 가져올 데이터를 저장할 변수들

export let siteStructure = null;

export let fuelcellConfig = null;

export let DEFAULT\_FUELCELL = null;

// 전역 변수로 설정

window.fuelcellConfig = null;

window.isFuelcellConfigLoaded = false;

async function fetchSiteStructure() {

try {

// 세션에서 발전소 목록 가져오기

const powerplants = await SessionService.getPowerplants();

console.log("세션에서 가져온 발전소 목록:", powerplants);

if (!powerplants || powerplants.length === 0) {

throw new Error('발전소 목록을 가져올 수 없습니다');

}

// GET 요청으로 사이트 구조 가져오기

const response = await fetch(`js/config/get\_site\_structure.php?powerplants=${powerplants.join(',')}`);

const data = await response.json();

if (data.success) {

siteStructure = data.structure;

fuelcellConfig = data.fuelcellConfig;

window.fuelcellConfig = data.fuelcellConfig;

DEFAULT\_FUELCELL = data.defaultFuelcell;

window.isFuelcellConfigLoaded = true;

// 데이터 로드 완료 이벤트 발생

document.dispatchEvent(new CustomEvent('fuelcellConfigLoaded'));

return true;

}

throw new Error(data.message || '사이트 구조를 가져오는데 실패했습니다');

} catch (error) {

console.error('사이트 구조 로드 실패:', error);

return false;

}

### 설정 및 선택 관리 시스템

// 현재 설정 가져오기

export async function getCurrentConfig() {

try {

await ensureDataLoaded();

// URL에서 파라미터 가져오기

const urlParams = new URLSearchParams(window.location.search);

const plant = urlParams.get('plant');

const group = urlParams.get('group');

const fuelcell = urlParams.get('fuelcell');

// URL에 파라미터가 있으면 사용

if (plant && group && fuelcell) {

return {

powerplant\_id: plant,

group\_id: group,

fuelcell\_id: fuelcell,

name: fuelcellConfig[fuelcell]?.name || fuelcell

};

}

// 없으면 선택된 연료전지 사용

const selectedFuelcell = await getSelectedFuelcell();

if (!selectedFuelcell) {

console.warn('선택된 연료전지가 없습니다. 기본 연료전지를 사용합니다.');

if (!DEFAULT\_FUELCELL) {

throw new Error('기본 연료전지가 설정되지 않았습니다.');

}

const defaultConfig = fuelcellConfig[DEFAULT\_FUELCELL];

return {

powerplant\_id: defaultConfig.plant,

group\_id: defaultConfig.group,

fuelcell\_id: DEFAULT\_FUELCELL,

name: defaultConfig.name

};

}

const config = fuelcellConfig[selectedFuelcell];

if (!config) {

console.warn(`연료전지 설정을 찾을 수 없습니다: ${selectedFuelcell}`);

if (!DEFAULT\_FUELCELL) {

throw new Error('기본 연료전지가 설정되지 않았습니다.');

}

const defaultConfig = fuelcellConfig[DEFAULT\_FUELCELL];

return {

powerplant\_id: defaultConfig.plant,

group\_id: defaultConfig.group,

fuelcell\_id: DEFAULT\_FUELCELL,

name: defaultConfig.name

};

}

return {

powerplant\_id: config.plant,

group\_id: config.group,

fuelcell\_id: config.fuelcell,

name: config.name

};

} catch (error) {

console.error('현재 설정 가져오기 실패:', error);

// 기본값 반환

return {

powerplant\_id: 'SE01',

group\_id: 'GR01',

fuelcell\_id: 'F001',

name: '기본 연료전지'

};

}

}

### UI 연동 및 이벤트 처리

// 연료전지 선택기 설정

export async function setupFuelcellSelector() {

await fetchSiteStructure();

if (!fuelcellConfig) {

throw new Error('사이트 구조를 로드할 수 없습니다');

}

const plantSelect = document.getElementById('plant-select');

const groupSelect = document.getElementById('group-select');

const fuelcellSelect = document.getElementById('fuelcell-select');

if (!plantSelect || !groupSelect || !fuelcellSelect) {

console.error("선택 요소를 찾을 수 없습니다");

return;

}

// 발전소 목록 채우기

plantSelect.innerHTML = '';

getPlants().forEach(plantId => {

const option = document.createElement('option');

option.value = plantId;

option.textContent = siteStructure[plantId].name || plantId;

plantSelect.appendChild(option);

});

// 발전소 변경 시 그룹 목록 업데이트

plantSelect.addEventListener('change', () => {

const selectedPlant = plantSelect.value;

updateGroupSelect(selectedPlant);

updateFuelcellSelect(selectedPlant, groupSelect.value);

updateURL();

});

// 연료전지 변경 시 이벤트 처리

fuelcellSelect.addEventListener('change', async function() {

updateURL();

const newFuelcell = this.value;

const newConfig = fuelcellConfig[newFuelcell];

if (newConfig) {

localStorage.setItem('selectedFuelcell', newFuelcell);

try {

// 이벤트 발생시켜 시스템에 연료전지 변경 알림

const event = new CustomEvent('fuelcellChanged', {

detail: {

plant: newConfig.plant,

group: newConfig.group,

fuelcell: newConfig.fuelcell

}

});

document.dispatchEvent(event);

// 데이터 새로 로드하고 UI 업데이트

const newData = await loadAllData();

if (newData) {

document.dispatchEvent(new CustomEvent('dataLoaded', { detail: newData }));

}

await updateModelName();

} catch (error) {

console.error('연료전지 변경 중 데이터 업데이트 실패:', error);

}

}

});

## 스택 검색 시스템(stack\_search.js)

스택 검색 시스템은 SIN/PULSE 모드 데이터를 검색하고 그래프로 시각화 기능, 북마크 관리, 상세 데이터 제공/다운로드 등 복합적인 기능을 제공.

### 검색 유형 전환 및 조건 처리

// SIN/PULSE 모드 전환 및 UI 업데이트

function updateEisOptions(selectedType) {

    const eisContainer = document.getElementById('eis-options');

    // SIN/PULSE 모드별 옵션 설정

    if (selectedType === 'SIN') {

        eisContainer.innerHTML = `

            <div class="sin-specific">

                <label>주파수 범위</label>

                <input type="number" id="freq-start" min="0.1" max="1000">

                <input type="number" id="freq-end" min="0.1" max="1000">

            </div>`;

    } else {

        eisContainer.innerHTML = `

            <div class="pulse-specific">

                <label>펄스 진폭</label>

                <input type="number" id="pulse-amplitude">

                <label>주기</label>

                <input type="number" id="pulse-period">

            </div>`;

    }

}

// 검색 유형(SIN/PULSE) 선택기 초기화

function initializeTypeSelector() {

const typeSelector = document.getElementById('search-type-selector');

typeSelector.addEventListener('change', function() {

const selectedType = this.value;

updateEisOptions(selectedType);

resetSearchConditions();

// 검색 유형에 따라 UI 업데이트

document.querySelectorAll('.type-specific').forEach(el => {

el.style.display = 'none';

});

document.querySelectorAll(`.${selectedType.toLowerCase()}-specific`).forEach(el => {

el.style.display = 'block';

});

});

// 초기 상태 설정

updateEisOptions(typeSelector.value);

}

// 고급 검색 조건 구성

function setupAdvancedSearch() {

const advancedSearchToggle = document.getElementById('advanced-search-toggle');

const advancedSearchContainer = document.getElementById('advanced-search-container');

advancedSearchToggle.addEventListener('click', function() {

const isVisible = advancedSearchContainer.style.display !== 'none';

advancedSearchContainer.style.display = isVisible ? 'none' : 'block';

this.textContent = isVisible ? '고급 검색 표시' : '고급 검색 숨기기';

});

// 비교 연산자 이벤트 처리

document.querySelectorAll('.comparison-selector').forEach(selector => {

selector.addEventListener('change', function() {

const fieldId = this.dataset.field;

const valueInput = document.getElementById(`${fieldId}-value`);

valueInput.disabled = this.value === 'any';

});

});

}

### 그래프 데이터 처리 및 파일 관리

// 선택된 파일들을 그래프용으로 복사

function copyFilesForGraph() {

const checkedBoxes = document.querySelectorAll('input[name="item"]:checked');

if (checkedBoxes.length === 0) {

alert('그래프로 표시할 항목을 선택하세요.');

return;

}

// 선택된 항목 ID 목록 생성

const selectedIds = Array.from(checkedBoxes).map(cb => cb.value);

// 그래프용 임시 폴더 비우기

clearSelectedDirectory()

.then(() => {

// 선택된 파일들을 그래프용 폴더로 복사

return fetch('copyFileForGraph.php', {

method: 'POST',

headers: { 'Content-Type': 'application/json' },

body: JSON.stringify({ ids: selectedIds })

});

})

.then(response => response.json())

.then(data => {

if (data.success) {

// 그래프 생성 또는 업데이트

updateGraph(data.files, data.coordinates);

// 최대값 저장 (있는 경우)

if (data.maxValue) {

saveMaxValueToFile(data.maxValue);

}

} else {

alert('파일 복사 중 오류가 발생했습니다: ' + data.message);

}

})

.catch(error => console.error('그래프 데이터 처리 오류:', error));

}

// 그래프 생성 및 데이터 처리

export async function handleFileOperations(dataNos, colorInput, fuelcell\_id) {

    try {

        for (const no of dataNos) {

            await copyFilesForGraph(no, colorInput, dateValue, fuelcell\_id, powerplant\_id);

        }

        return true;

    } catch (error) {

        console.error('파일 처리 오류:', error);

        return false;

    }

}

// 날짜 셀 클릭으로 그래프 생성

export function generateGraphFromDateCell(dataNo) {

    const color = getGraphColor(dataNo);

    handleFileOperations([dataNo], color, currentFuelcell);

}

### 북마크 관리 시스템

1. **북마크 데이터 구조**
   1. **bookmark 테이블**

* bookmark\_id: 북마크 고유 식별자
* bookmark\_name: 북마크 이름
* color\_code: 북마크 색상 코드(사용 중단됨)
* powerplant\_id, fuelcell\_id: 소속 정보
  1. **bmk\_sch 테이블**: 북마크와 검색 데이터를 연결하는 중간 테이블**(bmk\_id, sch\_id)**

1. **북마크 작동 방식**

* 검색 데이터 저장: 선택된 데이터 항목들을 북마크에 저장하여 나중에 쉽게 접근
* 빠른 데이터 접근: 저장된 북마크 클릭 시 해당 데이터 자동 검색 및 표시
* 탭 기반 인터페이스: 북마크를 탭으로 표시하여 직관적인 접근성 제공
* 북마크 관리 모달: 북마크 추가, 수정, 삭제를 위한 관리 인터페이스

1. **주의사항**

* 북마크는 발전소/연료전지별로 독립적으로 관리
* 북마크 색상은 그래프 표시 시 해당 데이터의 색상으로 사용
* 북마크 데이터 변경 시 자동으로 UI 업데이트

1. **관련 PHP 파일**

* get\_bookmark.php: 북마크 목록 조회
* add\_bookmark.php: 북마크 추가
* update\_bookmark.php: 북마크 정보 수정
* delete\_bookmark.php: 북마크 삭제
* save\_bookmark\_data.php: 북마크 데이터 등록

// 북마크 탭 로드 및 관리

// 북마크 탭 가져오기 및 표시

function getBookmarkTabs() {

const currentConfig = getCurrentConfig();

const { powerplant\_id, fuelcell\_id } = currentConfig;

// 북마크 목록 가져오기 - 모든 북마크 가져오기 (파라미터 없이 호출)

fetch('js/stack/get\_bookmark.php')

.then(response => response.json())

.then(data => {

// 북마크 탭 UI 생성 및 이벤트 핸들러 등록

var bookmarkList = document.getElementById("bookmark-tab").querySelector("ul");

bookmarkList.innerHTML = ""; // 기존 목록 초기화

// "전체항목" 탭 추가

var allItemsTab = document.createElement("li");

allItemsTab.classList.add("tab-item");

allItemsTab.innerHTML = `<a class="all-item-tab">전체항목</a>`;

bookmarkList.appendChild(allItemsTab);

// 북마크 탭 추가

data.forEach(bookmark => {

var newTabItem = document.createElement("li");

newTabItem.classList.add("tab-item");

newTabItem.innerHTML = `<a data-bookmark-id="${bookmark.id}" class="bookmark-tab">${bookmark.name}</a>`;

bookmarkList.appendChild(newTabItem);

});

// 북마크 관리 버튼 추가

const stackDataMngHead = document.getElementById('stack-data-mng-head');

if (stackDataMngHead) {

const plusButton = document.createElement("div");

plusButton.classList.add("bmk-list-mng-plus-btn");

plusButton.innerHTML = '<a class="plus" title="항목관리">+</a>';

stackDataMngHead.appendChild(plusButton);

}

});

}

// 북마크 데이터 필터링 및 표시

function filterDataByBookmark(bookmarkId, page = 1) {

// 현재 발전소 및 연료전지 ID 가져오기

const currentConfig = getCurrentConfig();

const powerplant\_id = currentConfig.powerplant\_id || 'SE01';

const fuelcell\_id = currentConfig.fuelcell\_id || 'F001';

// 검색 조건 파라미터 생성

const searchParams = new URLSearchParams();

searchParams.append('bookmarkId', bookmarkId);

searchParams.append('page', page);

searchParams.append('perPage', 100);

searchParams.append('type', type);

searchParams.append('powerplant\_id', powerplant\_id);

searchParams.append('fuelcell\_id', fuelcell\_id);

// 데이터 요청 및 표시

fetch(`js/stack/stack\_search.php?${searchParams.toString()}`)

.then(response => response.json())

.then(data => {

displayResults(data.data, page, data.totalRows, type);

displayPagination(data.totalRows, page, bookmarkId);

updateCheckboxesAfterSearch();

});

}

// 북마크 추가

function addBookmark(bookmarkName) {

// URL 파라미터 및 현재 설정에서 ID 값 가져오기

const urlParams = new URLSearchParams(window.location.search);

const currentConfig = getCurrentConfig();

const powerplant\_id = urlParams.get('plant') || currentConfig.powerplant\_id || 'SE01';

const group\_id = urlParams.get('group') || currentConfig.group\_id || 'GR01';

const fuelcell\_id = urlParams.get('fuelcell') || currentConfig.fuelcell\_id || 'F001';

// 북마크 추가 요청

const formData = new FormData();

formData.append('name', bookmarkName);

formData.append('powerplant\_id', powerplant\_id);

formData.append('group\_id', group\_id);

formData.append('fuelcell\_id', fuelcell\_id);

fetch('js/stack/add\_bookmark.php', {

method: 'POST',

body: formData

})

.then(response => response.json())

.then(data => {

if (data.success) {

// 북마크 리스트 새로고침 및 선택된 데이터 등록

getBookmarkTabs();

getTabList();

const checkedBoxes = document.querySelectorAll('input[name="data-checkbox"]:checked');

if (checkedBoxes.length > 0) {

registerBookmarkData(data.bookmarkId, checkedBoxes);

}

}

});

}

// 북마크 데이터 등록

function registerBookmarkData(bookmarkId, checkboxes) {

checkboxes.forEach(checkbox => {

const no = checkbox.getAttribute("data-no");

saveBookmarkData(no, bookmarkId);

});

}

// 북마크 데이터 저장

function saveBookmarkData(dataNo, bookmarkId) {

const formData = new FormData();

formData.append('sch\_id', dataNo);

formData.append('bmk\_id', bookmarkId);

return fetch('js/stack/save\_bookmark\_data.php', {

method: 'POST',

body: formData

})

.then(response => response.json());

}

## 유닛컨트롤(unitControl.js)

진단 장비의 상태를 모니터링하고 다양한 진단 모드를 지원하는 핵심 모듈

1. 실시간 모니터링: Redis를 통한 Gateway 상태와 파일 업로드 상태 모니터링
2. 진단 명령어 제어: 다양한 진단 모드(RUN, PULSE 등) 지원 및 상태에 따른 UI 제어
3. 자동 데이터 갱신: 2초 간격으로 상태 정보 자동 갱신
4. 연료전지 변경 대응: 연료전지 변경 시 자동으로 모니터링 설정 업데이트

### 상태 모니터링

class UnitControl {

constructor() {

this.redisKey = null; // Gateway 상태용

this.refreshKey = null; // 데이터 리프레시 제어용

this.lastRefreshTime = null;

this.consoleSection = document.querySelector(".console-section");

}

async initialize() {

try {

// 1. fuelcellConfig 로딩 대기

if (!window.isFuelcellConfigLoaded) {

await new Promise((resolve) => {

document.addEventListener('fuelcellConfigLoaded', resolve, { once: true });

});

}

// 2. Redis 키 설정 및 초기화

await this.updateRedisKey();

await this.loadAndDisplayRedisData();

this.startPeriodicRefresh();

this.listenForFuelcellChanges();

} catch (error) {

console.error('초기화 중 오류:', error);

this.resetDisplayData();

}

}

}

### 데이터 갱신 및 표시

class UnitControl {

async loadAndDisplayRedisData() {

try {

// 1. 리프레시 키를 통한 업로드 상태 확인

const refreshResponse = await fetch(`unitControl\_redis.php?key=${this.refreshKey}`);

const refreshResult = await refreshResponse.json();

if (refreshResult.data) {

const uploadedTime = this.parseTimeString(refreshResult.data);

if (uploadedTime > this.lastRefreshTime) {

this.lastRefreshTime = uploadedTime;

document.dispatchEvent(new CustomEvent('fileUploadSuccess'));

}

}

// 2. Gateway 상태 정보 업데이트

const statusResponse = await fetch(`unitControl\_redis.php?key=${this.redisKey}`);

const data = JSON.parse(statusResult.data);

// 3. PROGRESS 데이터 업데이트

if (data.PROGRESS) {

document.getElementById('progree-step').innerHTML =

`${data.PROGRESS.STEP}<sup>step</sup>`;

const unitValue = data.PROGRESS.UNIT || '0';

const totalValue = data.PROGRESS.TOTAL || '0';

this.updateProgressBars(unitValue, totalValue);

}

// 4. STATUS 데이터 업데이트

const status = data.STATUS;

this.updateStatusDisplay(status);

} catch (error) {

this.resetDisplayData();

this.consoleSection.innerHTML = `<pre>통신이 끊겼습니다.</pre>`;

}

}

}

### 진단 명령어 제어 시스템

class StartButtonHandler {

constructor() {

this.startButton = document.getElementById('start');

this.cmdSelect = document.getElementById('cmd');

this.isRunning = false;

this.lastCommand = null;

this.allCommands = ['connect', 'help', 'run', 'stop', 'dac\_range', 'pulse', 'disconnect'];

// 명령어 정보 매핑

this.cmdInfo = {

'run': {

desc: 'The RUN command starts a scanning of impedance measurements...',

val1Unit: 'Hz',

val2Unit: 'Hz'

},

'pulse': {

desc: 'The pulse command performs a measurement of transient responses...',

val1Unit: 'Hz',

val2Unit: 'mV'

},

// ... 기타 명령어 정보

};

this.initializeOptions();

}

async handleStartButtonClick() {

const cmd = this.cmdSelect.value;

if (!confirm("시작 하시겠습니까?")) return;

const config = getCurrentConfig();

const data = {

powerplant\_id: config.powerplant\_id,

fuelcell\_id: config.fuelcell\_id,

cmd: cmd,

val1: this.val1Input.value,

val2: this.val2Input.value,

bigo: this.bigoInput.value,

merr: this.merrInput.value

};

await this.sendCommand(data);

}

}

### 시스템 초기화 및 연동

document.addEventListener('DOMContentLoaded', async () => {

try {

const startButtonHandler = new StartButtonHandler();

const unitControl = new UnitControl();

if (window.isFuelcellConfigLoaded) {

await unitControl.initialize();

} else {

document.addEventListener('fuelcellConfigLoaded', async () => {

await unitControl.initialize();

}, { once: true });

}

} catch (error) {

console.error('페이지 초기화 중 오류:', error);

}

});