

□ Module 화 & 폴더 구조

폴더 구조

...

iot-dashboard/

├─ backend/

- | └─ config/db.js # DB 연결 설정 (PostgreSQL 등)
- | └─ controllers/temperatureController.js # 온도 센서 데이터 처리
- | └─ controllers/doorController.js # 출입문 센서 데이터 처리
- | └─ controllers/lightController.js # 조명 센서 데이터 처리
- | └─ routes/apiRoutes.js # API 라우팅 정의
- | └─ server.js # Express 서버 엔트리포인트
- | └─ package.json # Backend 의존성 관리
- | └─ .env.example # 환경변수 예시 파일 (실제 비밀번호 제거됨)

├─ frontend/

- | └─ src/components/Login.jsx # 로그인 컴포넌트
- | └─ src/components/Dashboard.jsx # 메인 대시보드 화면
- | └─ src/components/SensorCard.jsx # 센서 상태 표시 카드 UI
- | └─ App.jsx # React 앱 구조 뼈대

- | └─ main.jsx # React DOM 렌더링 진입점
- | └─ index.css # 글로벌 CSS 스타일
- | └─ tailwind.config.js # TailwindCSS 설정
- | └─ vite.config.js # Vite 빌드 설정
- | └─ package.json # Frontend 의존성 관리
- | └─ public/logo.svg # 프로젝트 로고 이미지

- └─ docs/
- | └─ README_ko.md # 프로젝트 개요 및 실행 방법 (한글)
- | └─ CONFIGURATION_ko.md # 환경 설정 방법 정리
- | └─ MODULES_ko.md # 모듈별 기능 설명
- | └─ API_SPEC_ko.md # API 명세서
- | └─ IoT_Dashboard_Presentation.pptx # 발표용 PPT 슬라이드
- | └─ IoT_Dashboard_Documentation.pdf # 최종 보고서 문서
- | └─ architecture_diagram.png # 전체 아키텍처 다이어그램
- | └─ api_flow.png # API 요청/응답 흐름도

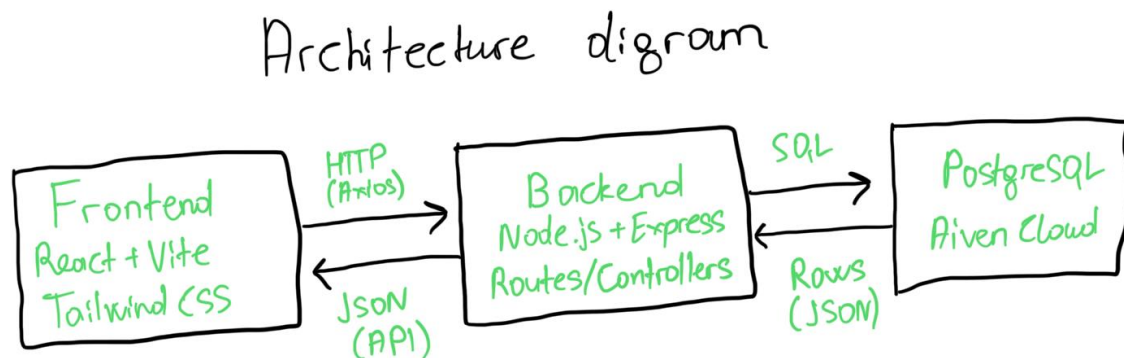
- └─ .gitignore # Git 추적 제외 설정
- ...

주요 모듈

- **backend/config/db.js** — DB 연결 (Pool)
- **backend/controllers/** — 비즈니스 로직 (temperature/door/light)
- **backend/routes/apiRoutes.js** — 라우팅 정의
- **backend/server.js** — 앱 부트스트랩
- **frontend/src/components/** — UI 컴포넌트 (Login, Dashboard, SensorCard)

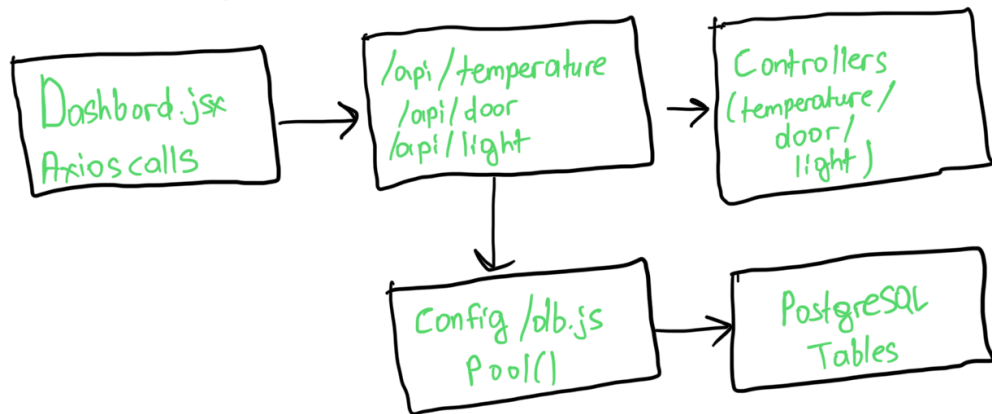
의존성/흐름 다이어그램

- 아키텍처: `architecture_diagram.png`



- API 플로우: `api_flow.png`

API FLOW



흐름 요약

1. 'Dashboard.jsx' → Axios 로 '/api/*' 호출
2. 'routes/apiRoutes.js' → 컨트롤러 연결
3. 컨트롤러 → 'config/db.js'의 Pool 로 SQL 실행
4. PostgreSQL 결과(JSON) → 프론트에 반환

Configuration

1) 환경 변수 (.env 예시)

> 제출에는 `backend/.env.example` 만 포함 (비밀번호 제거)

```
``env
PORT=3001
DB_USER=your_user
DB_PASSWORD=*****
DB_HOST=postgres-smartboy.h.aivencloud.com
DB_PORT=26207
DB_NAME=kafolat
DB_SSL=true
JWT_SECRET=your_jwt_secret  # 로그인 세션용
CORS_ORIGIN=http://localhost:5173
``
```

2) DB 연결 (backend/config/db.js)

- `pg`의 `Pool` 사용, `.env` 로드

- SSL 권장:

```
``js
```

```
ssl: {  
  rejectUnauthorized: false;  
}  
...
```

- 커넥션 풀 개수 제한 권장 (예: `max: 10`)

3) 스타일 (TailwindCSS)

- `src/index.css`:

```
``css  
@tailwind base;  
@tailwind components;  
@tailwind utilities;  
...
```

- `tailwind.config.js`: 필요 시 `darkMode: 'class'` 옵션 추가

4) Vite 개발 서버

- 기본 포트: 5173

- Proxy 미사용 시 → 프론트에서 직접 `http://localhost:3001` 호출

- Proxy 설정 예시 (`vite.config.js`):

```
```js
server: {
 proxy: {
 '/api': 'http://localhost:3001'
 }
}
```
```

5) 보안·배포 체크리스트

- `.env`는 Git 에 커밋 금지
- 최소 권한 DB 계정, SSL 활성화
- **CORS 설정**: 운영 환경에선 특정 도메인만 허용
- **JWT/세션 비밀키**는 강력한 랜덤 값 사용
- **Helmet / Rate limiting** 도입 권장 (Express 보안 강화)
- 프론트엔드 빌드:

```
```bash
npm run build
```
```

→ `/dist` 생성, nginx 등 static server 로 배포

6) Docker (선택 사항)

- `Dockerfile` 및 `docker-compose.yml` 작성 시:

- `backend` + `postgres` 컨테이너 통합 실행 가능

- `.env` 환경 변수 compose 파일에서 주입

☑ 위 설정으로 로컬 개발뿐 아니라, 실제 배포 환경에서도 안정적으로 실행 가능.

📖 API 명세

공통 사항

- Base URL: `http://localhost:3001/api`
- 모든 API 는 ****JSON 응답**** 반환
- 최대 6 개 레코드만 조회 (`NULL` 데이터는 제외)
- 에러 발생 시:

```
```json
```

```
{ "error": "에러 메시지" }
```

```
```
```

```
---
```

🔑 GET /api/temperature

- 설명: 최신 온도/습도 센서 데이터 조회
- 파라미터: 없음
- 응답 예시:

```
```json
```

```
[
```

```
{
```

```
 "s_id": "1",
```

```
 "s_temperature": "29.80",
```

```
 "s_humidity": "88.00"
```

```
}
```

```
]
...
```

- 비교:

- `s\_id`: 센서 ID

- `s\_temperature`: 현재 온도 값 (°C)

- `s\_humidity`: 현재 습도 값 (%)

---

## 🔑 GET /api/door

- 설명: 출입문(잠금) 상태 조회

- 파라미터: 없음

- 응답 예시:

```
```json
```

```
[
  {
    "s_id": "11111111111",
    "s_ip": "172.20.10.2",
    "s_status": "locked"
  }
]
```

...

- 비교:

- `s_id`: 센서 ID

- `s_ip`: 센서가 연결된 IP

- `s_status`: `"locked"` 또는 `"unlocked"`

🔑 GET /api/light

- 설명: 조명 상태 조회

- 파라미터: 없음

- 응답 예시:

```json

```
[
 {
 "s_id": "11111111111",
 "s_ip": null,
 "s_status": null
 }
]
```

```

- 비교:

- `s_id`: 센서 ID

- `s_ip`: 센서 IP (없으면 `null`)

- `s_status`: `"on"`, `"off"`, 또는 `null`


 향후 확장 고려


- `POST /api/door` → 원격으로 문 열기/잠그기

- `POST /api/light` → 조명 ON/OFF 제어

- `GET /api/temperature/:id` → 특정 센서 데이터 상세 조회

README-KR.MD

 IoT 대시보드 (React + Node.js + PostgreSQL)

 프로젝트 개요

온도/습도, 문 잠금, 조명 상태를 ****실시간 시각화****하는 풀스택 IoT 대시보드입니다.

프론트엔드와 백엔드, DB 를 직접 설계하고 연동하여 학습 및 발표용으로 개발했습니다.

- ****Frontend****: React (Vite) + Tailwind CSS

- ****Backend****: Node.js (Express)

- ****DB****: PostgreSQL (Aiven Cloud)

- ****통신****: Axios (HTTP), JSON 응답

📄 실행 방법

```bash

# 1) Backend 실행

cd backend

npm install

node server.js # => http://localhost:3001

# 2) Frontend 실행

cd frontend

npm install

npm run dev # => http://localhost:5173

```

🔑 로그인 (데모)

- ID: `admin`
- PW: `1234`

📁 프로젝트 문서

- ****CONFIGURATION_ko.md**** — 환경설정 및 빌드 도구
- ****MODULES_ko.md**** — 폴더 구조 및 모듈 설명
- ****API_SPEC_ko.md**** — API 명세
- ****그림 자료****: `architecture_diagram.png`, `api_flow.png`

👤 개발자

- 본 프로젝트는 ****제가 직접 기획·개발****한 IoT 학습용 대시보드입니다.
- 프론트엔드, 백엔드, DB 모두 직접 구현하고 문서화하여 발표 자료까지 준비했습니다.

💬 명언 (Quote)

> “미래를 예측하는 가장 좋은 방법은 그것을 발명하는 것이다.”

> — 앨런 케이 (Alan Kay)