CarbonCoder

2019315509 신용호
2019311779 최재경
2018313395 김병무
2019312755 정태문
2018313548 임수민
2021311279 김택림
2018314675 서현원

목차

Roles

신용호	조장, frontend의 코드 분석 페이지 구현, 코드 리뷰/리팩토링, deployment 서버 및 환경 구축
김병무	Frontend 코드 분석 페이지 UI 일부 구현
김택림	유저 제출 코드 실행 및 사용 자원 정보 제공 API 구현
정태문	그린화패턴 데이터 생성 및 그린화패턴 정보 제공 API 구현
최재경	그린화패턴 데이터 생성 및 에너지/탄소 계산 로직 구현
서현원	그린화패턴 카테고리 제공 API 구현
임수민	Frontend 그린화 패턴 페이지 구현

Goal

친환경 소프트웨어를 만들 수 있도록 코드의 탄소 배출량을 계산해주고,

탄소 배출량이 감소되는 그린화 패턴을 보여줌

탄소 배출량 측정

사용자에게 Java 코드를 입력 받아
탄소 배출량을 측정

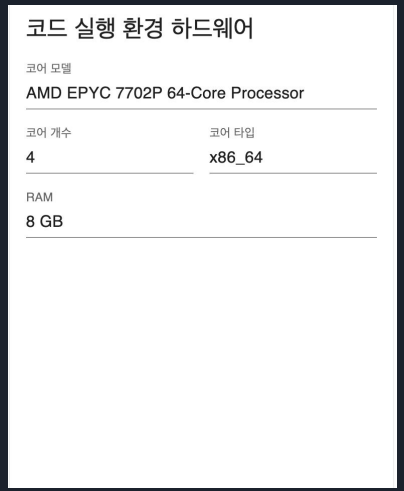
그린화 패턴 예시 조회

카테고리에 따른 다양한 탄소 배출량 감소
코드 패턴을 제시

프로젝트 실행 영상

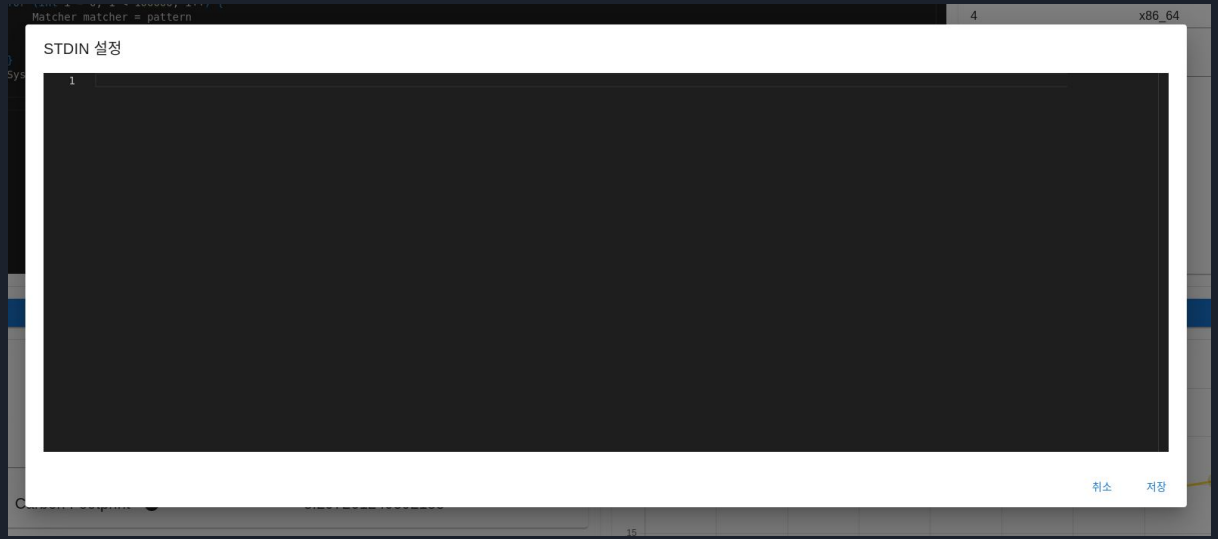
Key Feature - Code Analysis

- VSCode와 유사한 환경에서 코드 입력
- Java syntax highlighting 제공
- 5번 반복 실행 후 평균값을 반환
- 코드가 실행되는 하드웨어 환경 정보 제공 (고정값)



Key Feature - Standard Input

- 코드를 실행할 때 입력값을 설정 가능(STDIN)
- 입력값에 따른 증가폭 확인 가능

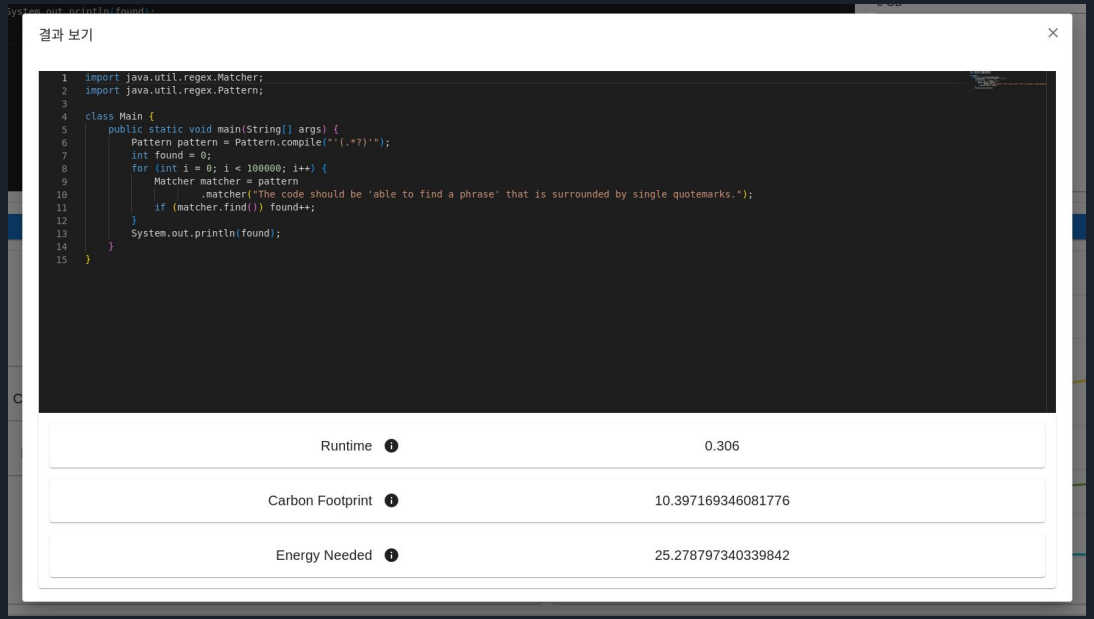


Key Features - Graph Visualisation

- 그래프를 통한 실행시간/탄소배출량/에너지 사용량 비교
- 런타임, 탄소 배출량, 에너지 사용량 증가/감소를 한눈에 파악

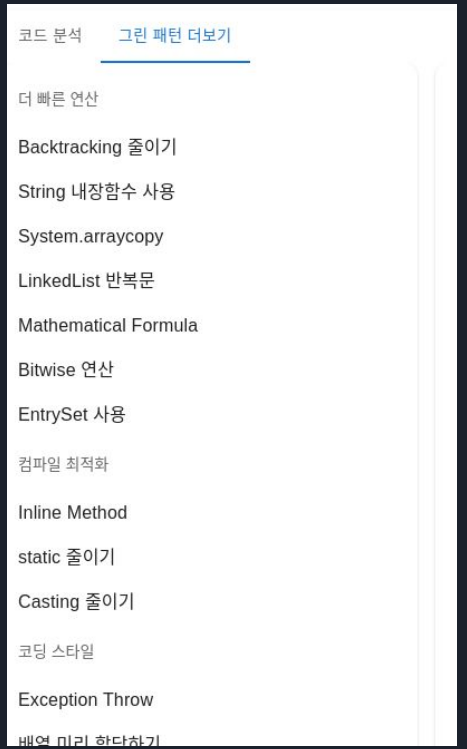
Key Features - Graph Visualisation

- 이전에 실행한 코드를 다시 확인 가능
- 런타임 정보, 탄소 배출량 계산 정보, 에너지 사용량 정보



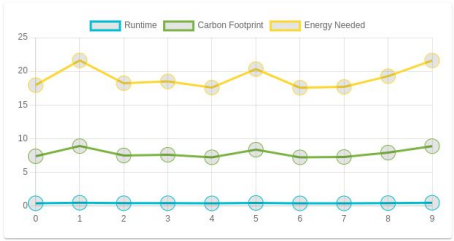
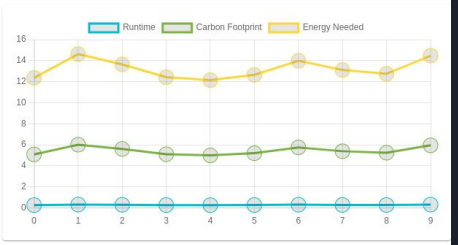
Key Features - Pattern Examples

- 서버에 저장된 예시 패턴 카테고리 별로 확인 가능
- 그린화 패턴 카테고리 정보
  - Java 객체 관련
  - 스레드 동기화
  - 컴파일 최적화
  - 코딩 스타일
  - 더 빠른 연산



Key Features - Pattern Examples

- 패턴 적용 전/후 코드, 변화 및 샘플 데이터 제공

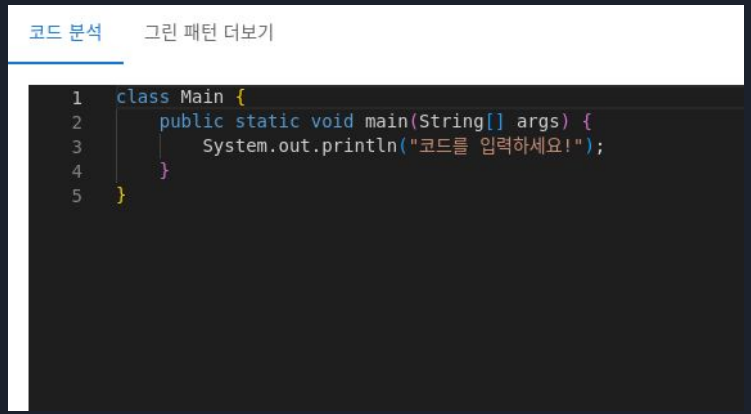
Runtime	0.42230999999999996	0.29343
Carbon Footprint	7.825441080565166	5.436030687408405
Energy Needed	19.02611495396344	13.216704807703408
Graph		

Development Management

`deployment`	Updated 8 minutes ago by MangoCubes	✓	3 | 0	#17 Merged
`pattern`	Updated 3 days ago by jeakyungc	✓	14 | 0	#16 Merged
`green`	Updated 3 days ago by MangoCubes	✓	5 | 0	#15 Merged
`bridging`	Updated last week by MangoCubes	✓	57 | 0	#9 Merged
`thread`	Updated last week by r01ex	✓	70 | 0	#7 Merged
`websocket`	Updated last week by r01ex		84 | 0	New pull request
`calculation`	Updated last week by root	✓	85 | 0	#5 Merged
`frontend-input`	Updated last week by MangoCubes	✓	89 | 0	#8 Merged

Development Management - Frontend

- 각 페이지마다 branch로 관리
  - frontend-input: 코드 입력 및 탄소 배출량 계산 페이지
  - green: 그린 패턴 예시 목록 표시 페이지



Development Management - Frontend

- ESLint를 이용한 코드 정리
- 불필요한 import, 변수 감지
- 코드 유지를 어렵게 하는 타입 사용 금지

```
Failed to compile.

[eslint]
src/analyser/ChartDisplay.tsx
  Line 1:26:  'Stack' is defined but never used  @typescript-eslint/no-u
nused-vars

Search for the keywords to learn more about each error.
```

Development Management - Frontend

- 코드 타입 정보에 주석 추가
- 직접 만든 컴포넌트의 property에 대한 주석 추가
- 리팩토링을 거쳐서 컴포넌트 재사용, 불필요 컴포넌트 삭제

```
type Props = {
  // 분석 중인지 여부
  sending: boolean;
  // 분석 중인지 여부를 설정하는 함수
  setSending(value: boolean): void;
}

// 성공한 분석 결과와 실패한 분석 결과를 합친 타입
export type AnalysisResult = SuccessfulAnalysis | FailedAnalysis;

// 코드 분석 페이지 코드
export function Analyser({ sending, setSending }: Props) {
  // 현재 입력된 코드
  const [code, setCode] = useState(defaultVal);
```

Development Management - Backend

Backend 개발 부분을 3개의 Subcomponents로 나누어 진행하고 이를 각 branch에서 진행

- thread : Code Execution Request system

  Judge0 code 와 통신을 위한 스레드 생성 및 결과 반환 코드 개발

- calculation : Usage Calculation System

  코드 탄소배출량 계산 코드 개발

- pattern : Green Algorithm Response system

  그린화 패턴 정보를 전달하는 API end-point 개발 및 json 파일 업로드

Development Management - Backend

- 기능 설명 및 상세 주석 추가

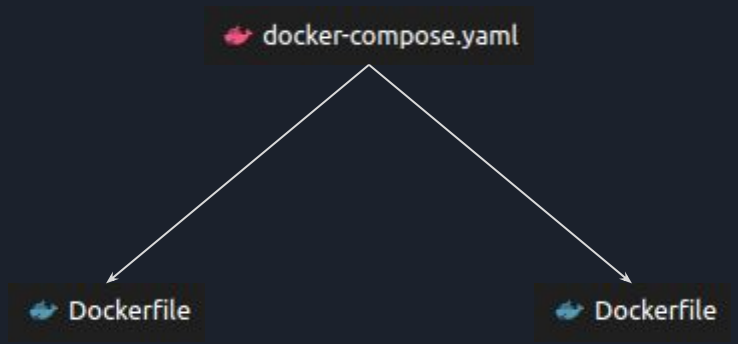
```
# 패턴 리스트와 개별 패턴 정보를 저장하는 JSON 파일 경로
pattern_list_json_file_path = "./pattern_list.json"
patterns_json_file_path = "./patterns.json"

def request_to_judge0(code: str, stdin: str):
    request_json = {
        # Judge0가 UTF-8로 인코딩을 할 수 없을 경우 문제를 발생시키므로 데이터를 송수신할때는 base64로 인코딩하여 사용
        "source_code": base64.b64encode(bytes(code, 'utf-8')).decode('utf-8'),
        "language_id": 62,
        "stdin": stdin,
        "number_of_runs": 5,
        # "cpu_time_limit": null,
        # "cpu_extra_time": null,
```

```
# 에너지와 탄소 발자국을 계산
def calculate_energy_and_carbon(cpu_time: float, memory: int):
    n_core = int(1)  # number of processor
    tdp = 12.5  # TDP(W): Core type: AMD EPYC 7702P 64-core processor, 200W for 64 cores. Our server has 4 cores.
    u_core = int(
        1
    )  # Usage of core(%): when cannot identify use 1 for value, [0,1] : 1 = 100%
```

Development Management - Others

- bridging: 프론트엔드와 백엔드를 연결하기 위한 변경사항 관리
- deployment: 서비스를 실제로 deploy하기 위한 환경변수 설정
- Dockerfile, docker-compose 관리



Finding Green Patterns

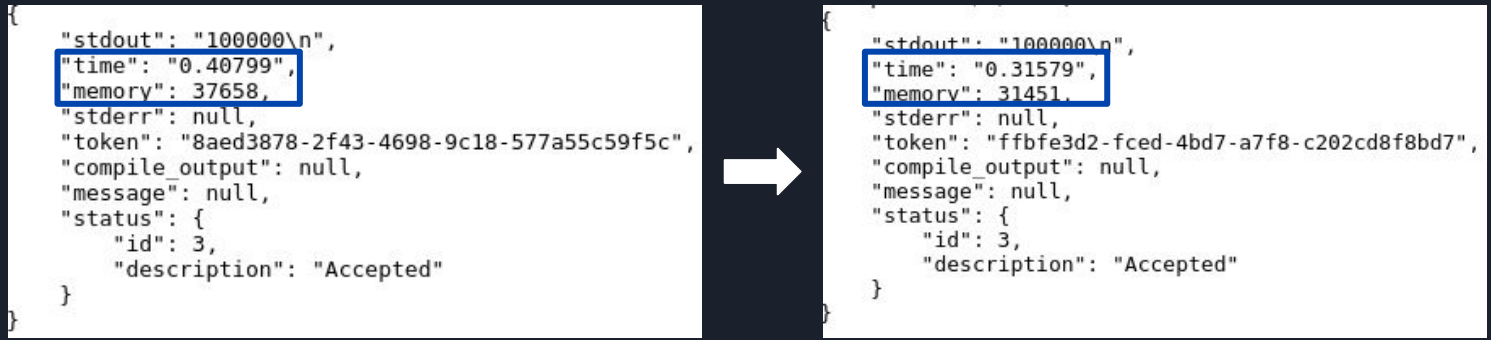
$$E = t \times (n_c \times P_c \times u_c + n_m \times P_m) \times PUE \times PSF$$
$$C = E \times CI$$

- 탄소발자국 C는 에너지 사용량 E에 비례
- 메모리 사용량, 실행시간이 제일 큰 영향을 줌
- 단위 에너지 생산에서 발생하는 탄소량 CI는 상수[1]
- 에너지 사용량 E 계산식에서의 상수 (코드가 변해도 값이 변하지 않거나 변화가 작음)
  - 코어의 수
  - 코어 전력소비량[2]
  - 코어 사용률 (정규화)[3]
  - 메모리 전력 소비량
  - 데이터 센터의 효율성 계수[4]
  - Pragmatic Scaling Factor (반복 실행에 대한 값)

1: "CarbonFootprint", Grid Electricity Conversion Factors Publications, 2023.07,https://www.carbonfootprint.com/, South Korea의 값 사용
2: 열 설계 전력(TDP)를 사용, 코어수에 맞게 값 조정.
3, 4: 측정이 어려우므로 1(100%)로 고정함.

Finding Green Patterns

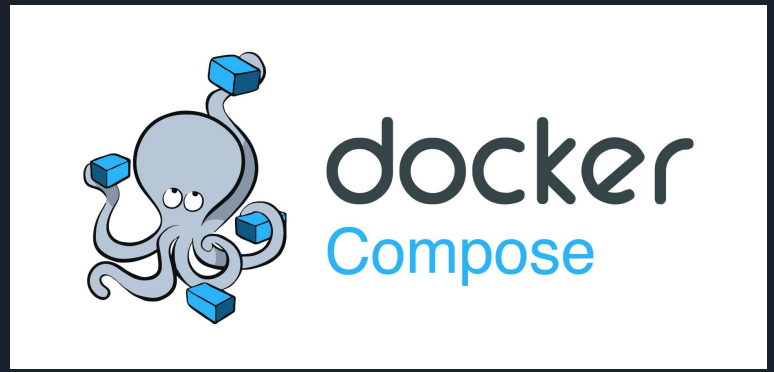
- 프로그램 실행 시간/메모리 사용량이 줄어들 것이라고 예상되는 목록 작성
  - 프로그래밍을 하면서 배운 최적화 방법을 사용한 패턴
  - Stackoverflow에서 추가 패턴 수집
- Judge0에서 코드를 실제로 실행 후 결과 확인
  - 20번 실행 후 평균값을 사용하여 일시적 실행시간 증/감 등의 오류 최소화
- 패턴들 중에서 메모리 사용량 또는 실행 시간이 감소한 패턴 목록 작성



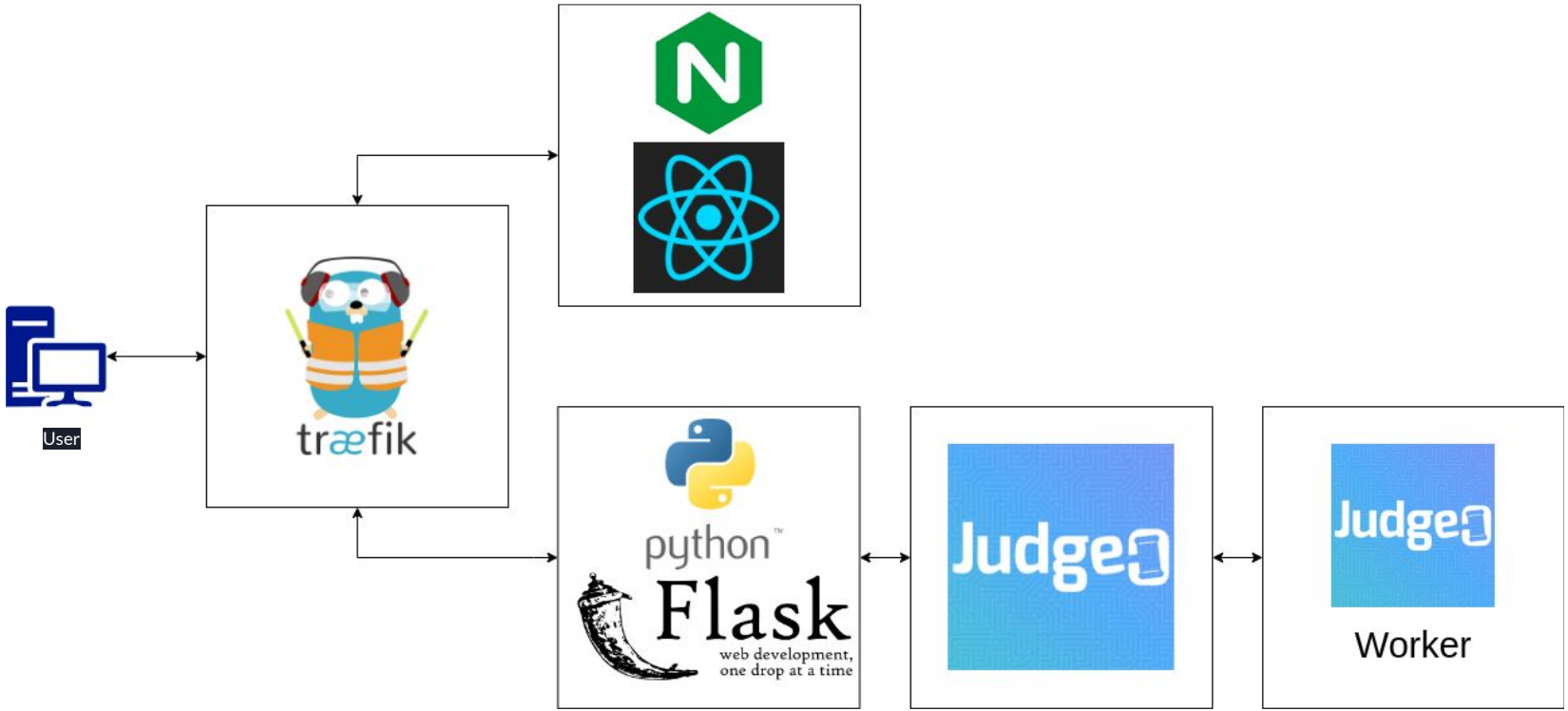
Deployment

- Docker 사용, docker-compose로 관리
- 총 5개의 container로 4개의 서비스 실행
- docker compose up -d --build로 모든 컨테이너 시작 및 연결

```
[+] Running 4/4
 ✔ Container judgeworker   Running
 ✔ Container judge         Running
 ✔ Container frontend      Started
 ✔ Container backend       Running
admin@server:~/docker/dev$
```



Structure


User
træfik
python
Flask
web development,
one drop at a time
Judge
Judge
Worker

Component - Traefik

- 오픈소스 HTTP 리버스 프록시
  - 사용자의 요청을 받고 이를 웹서버로 전달
- TLS와 요청 전달을 투명하게 제공
- 외부에서 컨테이너 접근 허용
- URL의 path로 접근할 컨테이너 라우팅
  - cc.skew.ch -> Nginx
  - cc.skew.ch/api -> Flask
- 사용자가 접근할 수 있는 곳 최소화

Component - Nginx

- 오픈소스 웹 서버
- 웹 프론트에 사용
- 빌드한 React 코드를 클라이언트에게 전달
  - 빌드는 Node.js 컨테이너에서 실행

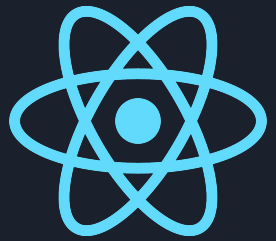
Component - Python Flask

- 오픈소스 REST API 서버
- Judge0와 사용자 요청 사이에서 중재
  - Judge0의 직접적인 접근 제한
  - 과도한 요청 방지 (IP기반 제한)
- 그린화 패턴 예시 API 제공

Component - Judge0

- 오픈소스 코드 실행 서버
- 코드를 5번 실행하여 최적화로 인한 결과 변화 최소화
- 실행 후 실행 시간, 사용한 메모리를 REST API로 반환
- 에러 발생시 이를 대신 반환
- Judge0와 Judge0 worker로 구성됨

Open Source Projects Used - Frontend



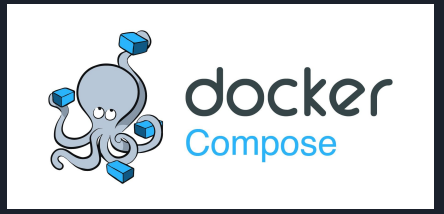
- React: UI 개발 Javascript 라이브러리
- MUI: React 컴포넌트 라이브러리
- Monaco editor: 웹 코드 에디터
- Chart.js: 차트 생성기
- Node.js: React 웹사이트 빌드

Open Source Projects Used - Backend

- Python
- Flask
- Python Request

Open Source Projects Used - Deployment

- Nginx: HTTP 서버
- Docker와 docker-compose: 컨테이너화 및 컨테이너 관리
- traefik: HTTP 리버스 프록시
- Judge0: 코드 실행 서버



감사합니다