

이지호

Full Stack Developer

Computer Science와 Algorithm을 개발에 접목시키는 풀스택 개발자입니다.

모던 프론트엔드와 관련된 툴체인에 관심이 많아, React 웹앱을 개발하며 연관된 빌드 시스템을 boilerplate 없이 end-to-end로 구성할 수 있습니다.

백엔드에서 Python, Django, DRF, Django ORM과 Celery를 주로 다룹니다.
백엔드 개발에서 유닛 테스트와 Property Based Testing을 hypothesis 라이브러리를 통해 사용하고 있습니다.

Infrastructure as Code 시스템을 Kubernetes, Docker로 설계할 수 있고 배포 자동화 시스템을 Github Actions와 shell의 조합으로 작성할 수 있습니다.

2022년 현재 보충역 산업기능요원으로 신규 편입이 가능합니다.

보유 기술

Front-end Engineering

- Typescript를 주 언어로 사용
- React, Jotai, Material UI등 리액트 기반 프레임워크에 능숙
- React-Konva 프레임워크를 통해 캔버스 기반 인터랙티브 툴 개발 경험

Back-end Engineering

- Django, DRF 기반 백엔드 애플리케이션 작성 경험 다수
- drf-yasg를 사용한 swagger 명세 관리
- Uvicorn을 사용한 Django multiple worker 서버 구축
- Celery와 Channel 이용 리얼타임 WebSocket 서버 / 워커 / 클라이언트 작성
- Celery 환경에서 Pytorch 작업 내장 multiprocessing으로 병렬화 ([관련 이슈](#))

Infrastructure as Code

- 프론트엔드 빌드 툴체인에 대한 이해와 최적화에 능숙
- create-react-app에 의존하지 않고 더 최적화된 리액트 웹앱 구축 가능
- Github Actions 기반 Docker, Kubernetes 빌드 및 배포 자동화 구축 경험 다수

경력

AUTOCRYPT / 프론트엔드 개발자 / 인턴

2021.04 ~ 2021.07

- Vehicle Security Operations Center 프론트엔드 PoC 작업
- 공개 키 기반 암호화 jQuery 기반으로 이원화된 웹앱을 하나의 Vue 앱으로 재작성

주식회사 에스로드 / 풀스택 개발자 / 정규직

2021.07 ~

- 머신러닝 기반 데이터 라벨링 플랫폼 메인 개발자

교육사항

한국산업기술대학교 / 컴퓨터공학과 학사과정

2019.03 ~

프로젝트 상세

- 이미지 라벨링을 자동화하고 검증 및 모니터링할 수 있는 웹 기반 플랫폼
- FE: Typescript, React, Jotai, MUI, React-router, React-Konva, Yarn Berry
- BE: Python, Django, DRF(APIView, Serializer), drf-yasg(Swagger), Uvicorn, Celery, Channel(WebSocket Server), Redis(as MQ)
- CI/CD: Github Actions, Docker(+ Private Registry), Kubernetes
- Github Flow로 주별 스프린트 관리

역할 및 성과

- 팀 내 프론트엔드, 백엔드 및 배포 관련 메인 개발자 역할
- webpack 빌드 최적화로 46초 -> 4초 이내 빌드 환경 구성 (11배 향상, M1 Pro 기준)
- react custom hook으로 JWT Access Token, Refresh Token 워크플로우 자동화
- 라벨링된 json을 이미지에 렌더링시켜주는 파서 함수 작성
- yolov5, resnet50 모델 사용해서 오토 라벨링 프로토타입 구현
- 이미지 검증자의 편의를 위한 배치 라벨링 작업을 Worker Job으로 Pytorch 내장 pool 사용해서 병렬화
- Channel과 WebSocket 프로토콜을 사용해 배치 라벨링 중 실시간으로 갱신 요청
- Inner Join과 Prefrech_related 메소드 사용함으로써 ORM N + 1 문제 해결
- 오픈소스 웹 크롤러인 AutoCrawler 기반으로 Celery + Docker 환경으로 포팅, Django ORM + Boto3 사용하여 DB 업로드 방식으로 변경
(도커라이즈 관련 호스트/컨테이너 아키텍처 관련 문제 트러블슈팅 StackOverflow 작성 기록)
- 점진적으로 백엔드 단위 테스트, Property Based Testing 도입
- Django ORM을 사용한 데이터베이스 스키마 공동 설계
- Sliding Window 알고리즘 사용한 N:M 3배수 교차 검증 분배 알고리즘 작성
- Kubernetes Deployment 내부에서 Active / Active Pod 이중화로 DB 제외 SPOF 제거
- Github Actions를 사용하여 docker image build & push, kubectl로 쿠버네티스 릴리즈 배포 자동화
- github actions상에서 Docker Compose로 컨테이너 환경에서 CI 구축