

이지호

Full Stack Developer

Computer Science와 Algorithm을 개발에 접목시키는 풀스택 개발자입니다.
시간 복잡도와 빌드 최적화에 집착하지만, 의미있는 현실 세계의 문제를 풀려고 노력합니다.
현재 테스트 자동화를 비롯한 Software Verification에 관심이 많습니다.

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.

Java로 객체지향 분석과 설계를 할 수 있고, 기본적인 Spring Boot 애플리케이션을 작성할 수 있습니다.

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

교육사항

한국산업기술대학교 / 컴퓨터공학과 학사과정 (2021년 휴학)
2019.03 ~

Aurola / Freelancer Task (2021.08 ~)

머신러닝 기반 데이터 라벨링 플랫폼

프로젝트 상세

- 이미지 라벨링을 자동화하고 검증 및 모니터링할 수 있는 웹 기반 플랫폼
- FE: Typescript, React, Jotai, MUI, React-router, React-konva
- BE: Python, Django, DRF, 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 -> 2초 이내 빌드 환경 구성 (23배 향상, M1 Pro 기준)
- 빌드를 최적화한 이유: 단순히 시간 문제가 아닌 프론트엔드 개발 사이클 속도에 정비례하기 때문
- git을 통한 직접적인 패키지 버전 관리 환경을 구성하기 위해 yarn berry를 도입
- react lazy component loading 도입, 유의미한 수준의 초기 렌더링 시간 축소 및 번들 파일 분리로 인한 빌드 성능 추가 향상 (4초 -> 2초)
- react custom hook으로 JWT Access Token, Refresh Token 워크플로우 자동화
- 이미지 캔버스에 렌더링한 뒤 객체 표시와 캔버스 내부 에디터 필요에 따라 react-konva를 이용하여 라벨링된 json을 이미지에 렌더링시켜주는 파서 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 라벨링이 하나 끝날 때마다 워커에서 서버를 거쳐 클라이언트에게 실시간으로 간접 요청
- Inner Join과 Prefetch_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 Actions 사용 로컬 docker registry 사용해서 docker image build & push, kubectl로 쿠버네티스 원격 배포까지 툴링 릴리즈로 자동화
- CI의 이전 조건으로 CI가 돌아가도록 구성, 테스트 스위트 작성으로 의미있는 CI 구성

params-to-querydsl-example (2021.05 ~ 2021.06) [Github](#)

Spring Boot MVC GetMapping 파라미터를 QueryDSL로 파싱하는 PoC

프로젝트 상세

- 당시 필요에 따라 Spring Boot상에서 동적 쿼리처럼 동작하는 GET 메소드가 필요했지만 관련된 코드를 웹에서 간단하게 찾을 수 없어서 오픈소스로 작성해서 공개한 PoC
- 조금 더 편하고 유연한 스트림 역할로 RxJava 사용

사용 기술

- Spring Boot
- Spring Data JPA
- QueryDSL
- RxJava 3 (as better stream)

역할 및 성과

- 개인적 불편 -> 개발 -> 공개로 이루어지는 오픈소스 개발 문화 경험
- 웹에서 바로 정답을 찾을 수 없는 문제에 대해 해결책을 고민하고 공식 문서를 통해 필요한 인터페이스를 찾아내는 과정을 겪음

cors-bypass-api(2021.06~ 2021.07) [Github](#)

프론트엔드 CORS 문제를 해결해주는 Spring Boot 기반 RESTful 프록시 서버

프로젝트 상세

- CORS를 해결하는 두 가지 방법중에 프록시 서버를 구현
- API에 목적지 URL을 넣어서 GET 요청을 하면 서버가 헤더와 URL 파라미터를 파싱하고 목적지에 리퀘스트를 보낸 뒤 대응되는 리스폰스를 클라이언트에게 전달

사용 기술

- Spring Boot
- Spring Data JPA

역할 및 성과

- 다른 CORS 관련 API에 대해 조금 더 구현을 생각해볼 수 있게 됨
- Request / Response 모델에 대한 이해
- Python에서도 기본적인 개념은 변하지 않아 aiohttp나 requests등으로 프록시 로직을 변환해서 사용할 수 있는 방법을 익히게 됨