Big Data Processing & Trend Analysis

Big Smile Team

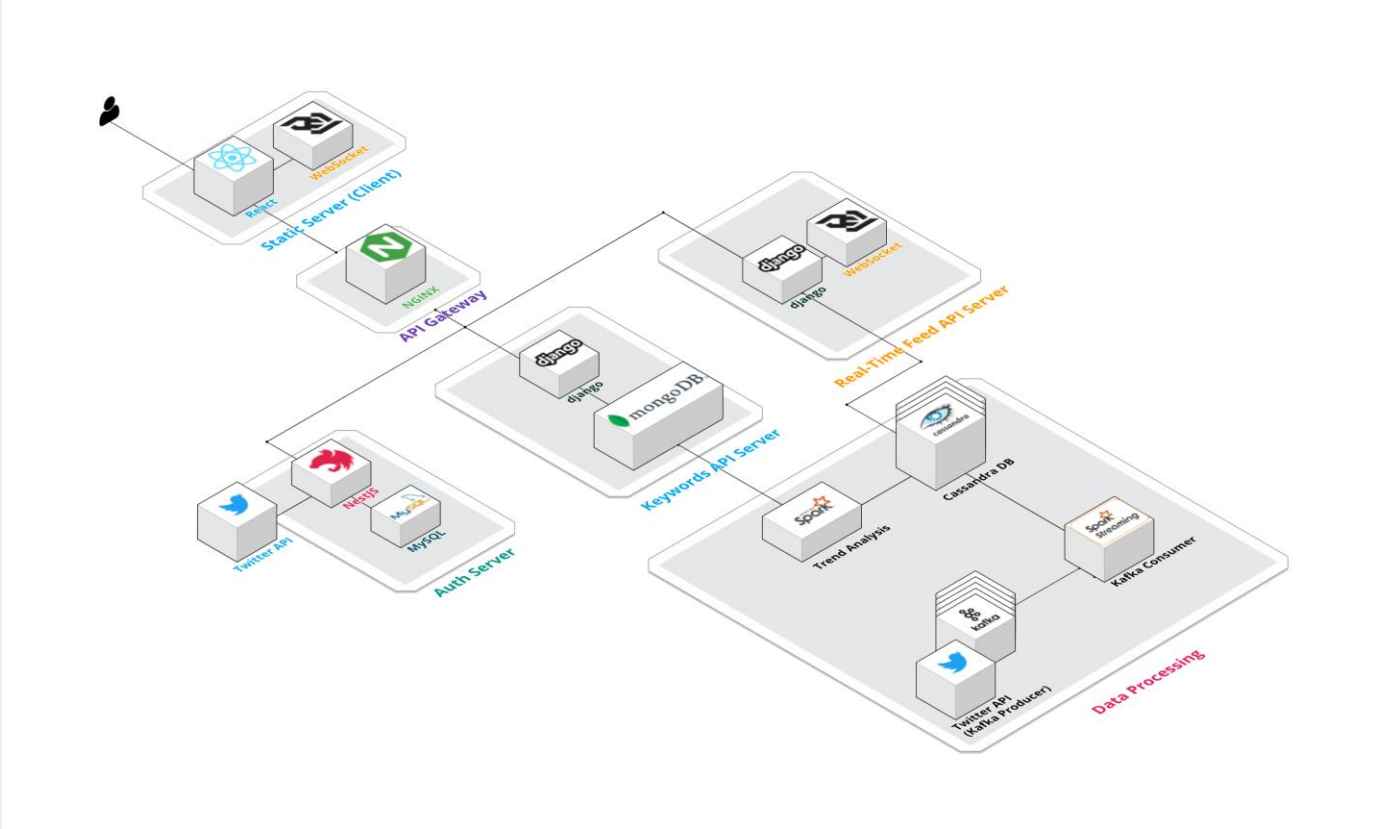
김가윤 이규은 정석준 조인식 추연호

01 프로젝트 팀 목표

TweetDeck과 SomeTrend를 벤치마킹하여 각자 관심있는 기술을 습득하고, 기술을 통합하여 TweeTrend라는 하나의 서비스를 완성한다.

TweetTrend 서비스를 개발하면서 사용한 기술에 대해 서로에게 설명할 수 있을 정도로 관심 분야 기술에 대한 이해를 높이는 것을 목표로 한다.

02 프로젝트 설계 – 아키텍처 설계



02 프로젝트 설계 – 기술 스택

Data Processing

언어 : Python
프레임워크 : Kafka, Spark, Spark Streaming
분산형 DB : Cassandra DB
기타 : Twitter API

BackEnd

언어 : Python, TypeScript & JavaScript
프레임워크 : Django, NestJS
Database : MySQL , MongoDB
API Gateway : Nginx
기타 : Web Socket, uWSGI, Daphne, JWT, Redis

FrontEnd

- 언어 : TypeScript & JavaScript
- 프레임워크 : React (FrontEnd), NestJS (Auth-Server)
- 상태 관리 : React Hooks (필요시 Redux, MobX, Recoil 중 선택하여 추가)
- 스타일링 : Styled-components + UI Library 혼합
- 데이터 시각화 : recharts, visx 혼합
- 기타 : Web Socket, Storybook

02 프로젝트 설계 – 기능 정의서

Category	Feature	Detail
회원가입 & 로그인	일반 회원가입	
회원가입 & 로그인	이메일 인증	
회원가입 & 로그인	소셜 로그인	선택 기능으로 기간에 여유가 있을 때 구현. (트위터 or 구글 or 카카오)
유저 관심 키워드 관리	입력 폼 개발	회원 가입이 완료되면 폼을 제공해 유저에게 관심 주제를 입력 받는다. Ex) 코로나, BTS, 넷플릭스
유저 관심 키워드 관리	관심 키워드 추가/삭제	유저는 관심 있는 주제를 추가하거나 삭제할 수 있다.
유저 관심 키워드 관리	관심 키워드 별 페이지 이동	메인 화면에는 하나의 주제에 대한 통계 데이터가 보여지고 상단 탭으로 유저가 관심 있는 주제별로 이동할 수 있다.
통계 데이터	기간 별 통계 데이터	메인 화면에는 키워드에 대한 여러 통계 데이터를 보여준다. 유저는 일간, 주간, 월간, 3개월간 통계를 선택할 수 있다.
통계 데이터	언급량 추이	해시태그의 언급량 추이 그래프를 보여준다.
통계 데이터	개별 사용자 추이	해시태그를 사용한 개별 사용자 추이 그래프를 보여준다.
통계 데이터	컨텐츠 분석 - 인기 게시물	기간 내에 가장 인기있는 게시물 목록을 보여준다.
통계 데이터	컨텐츠 분석 - 사용자	기간 내에 해당 해시태그를 가장 많이 사용한 사용자 목록을 보여준다.
통계 데이터	연관어 분석 - word cloud	관련 키워드를 word cloud 형태로 보여준다.
통계 데이터	연관어 분석 - 순위	연관어 순위를 보여준다.
통계 데이터	연관어 분석 - 빈도수	연관어 빈도 수를 비교하는 그래프를 보여준다.

03 마일스톤

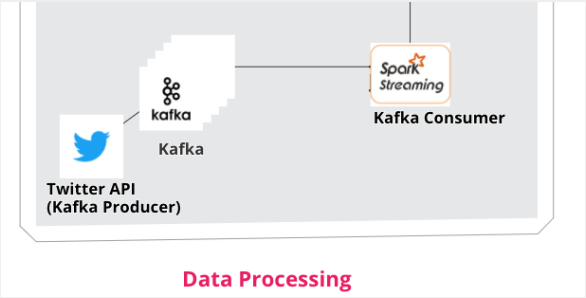
M1	DevCamp 기간 내에 반드시 완료해야하는 목표	- Adobe XD와 Figma를 활용한 디자인 프로토타입 제작 - 코로나 키워드에 대한 트렌드 분석 제공 - 특정 토픽에 대한 트윗을 실시간으로 제공하는 실시간 피드 개발 - 기본적인 트렌드 분석 대시보드 구현 => 총합, 평균 등 단순 통계치 ex) 총 언급량, 언급량 추이, 연관어 분석 - 데이터 처리 Kafka - Spark - Cassandra 연결하여 데이터 통신 파이프라인 구축 - Spark Streaming과 Spark Consumer, Spark Cassandra Connector 구현 - 키워드 수에 따라 Scale-Out 가능한 구조를 설계하여 Kafka broker(topic 및 partition) 구현 - 유지 보수가 원활하도록 분산 DB 구축 및 분석에 필요한 추가적인 데이터 모델링 - 트위터 소셜 로그인을 사용한 유저 인증 (토큰 기반 인증으로 MSA로 동작) - AWS 내에서 개발 환경 구축 및 전체 서비스 통합 - 기간별 데이터 제공 기능 구현
M2	DevCamp 기간 안에 어떻게든 최선을 다한다면 가능할 수 있을 것 같은 목표	- covid 등의 키워드를 추가해 대용량 데이터에 대한 성능 측정 - 추가적으로 분석 기법이 들어간 Trend 컴포넌트들을 완성 (워드 클라우드, 버블 차트 등) - 트위터와 상호작용할 수 있는 기능 구현 (좋아요, 리트윗, 댓글 등) - 데이터 유실이 없도록(또는 속도와의 Trade-Off) Kafka 파라미터 튜닝 및 키워드 확장 적용 - DB 퍼포먼스 향상을 위한 튜닝 - 웹 서버와 웹 인터페이스 연결에 있어 API 게이트웨이를 통해 짜임새 있는 MSA 구조 구축 - 트위터에서 제공하는 데이터 최대한 활용하여 통계 컴포넌트 늘리기 (사용자 위치, 작성 디바이스 등) - 웹페이지 모바일 반응형 지원 - 랜딩 페이지 구축 (홈 화면, 서비스 소개)
M3	DevCamp 기간 내에는 불가능하지만 궁극적으로 만들고자 하는 목표	- 유저가 검색한 검색어 데이터 실시간 반영으로 확장하기 - 유저가 검색한 내용을 twitter API 의 파라미터로 입력해 실시간으로 데이터를 받아옴(유동적) - twitter API의 Sample Stream을 이용해 많은 양의 전체 tweets 의 수를 샘플링하여 감당할 수 있는 양의 데이터에서도 동향을 파악할 수 있도록 함 - 해당 검색어와 코로나와의 상관성을 분석하여 인사이트 제공 - 분석 결과 개인 유저 별 저장 - 트위터 API 뿐만 아니라 다른 데이터 소스까지 확장(인스타그램 등)

04 개인 목표 – 기술 스택 및 R&R

조인식

실시간으로 생산되는 많은 양의 Twitter 데이터를 안정적으로 수집/전달하는 경험을 통해 데이터 엔지니어링에 대한 역량을 높이겠습니다.

기능 및 역할



- Twitter API를 이용해 수집한 실시간 Tweets 데이터를 전달
- 확장될 수 있는 데이터를 고려해 Scale-Out 가능한 모델 설계
- 낮은 유실률과 빠른 속도를 목표로 설정

개인 목표

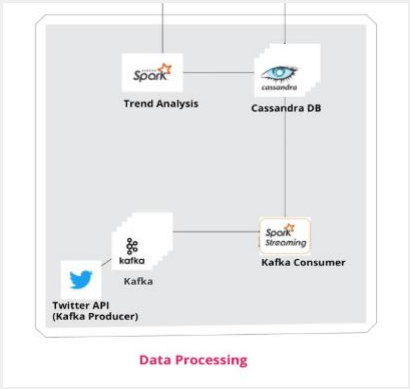
- 데이터 파이프라인에 대한 지식을 쌓고, 프레임워크의 종류와 각 장단점을 조사하여 이를 프로젝트에 반영하겠다.
- 성능 및 효율에 관련된 파라미터들을 파악하여 대용량 데이터에 대비한 확장성을 가지는 파이프라인을 구축하겠다. (다양한 파라미터 및 분산된 서버 환경에서 테스트를 진행해 효율 및 확장성을 판단해볼 것)
- 데이터파이프라인 구축 중 맡은 부분에 대해 타인에게 설명할 수 있을 정도로 익히고, 그 외의 부분에도 의논이나 질문을 통해 흐름 파악 정도는 할 것
- 결과물에 대해 피드백을 받는 것 외에 스스로도 속도와 유실률 등의 측면에서 평가해보겠다.
- 위의 목표를 이루는 과정이 후에 도움이 될 수 있도록 기록을 남기겠다.
  (gitlab data processing repo를 통해 정리)

04 개인 목표 – 기술 스택 및 R&R

김가윤

Spark를 활용한 실시간 데이터 전처리로 실제 서비스 가능한 기능들을 구현하여
대용량 데이터처리 파이프라인과 데이터 가공 및 분석에 대한 이해도를 높이겠습니다.

기능 및 역할



- 데이터 가공, 저장 및 분석 기능 구현
- 실시간 피드를 위한 Spark Streaming과 Trend 분석을 위한 Spark 활용
- 수집된 데이터 전처리를 맡아 Kafka와 Cassandra DB를 연결하는 역할 담당
- Cassandra DB에 저장된 데이터 분석을 맡아 MongoDB에 저장 담당

개인 목표

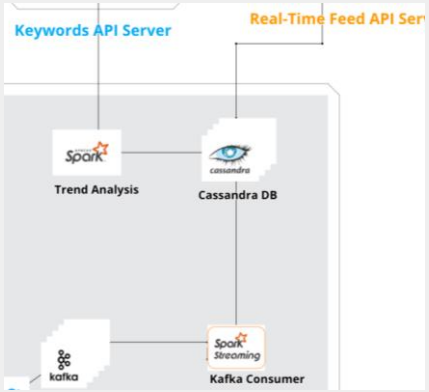
- 데이터 처리 중 Spark Streaming을 활용한 실시간 데이터 수집 및 DB 저장 기능 구현해보기
- Spark 내부 데이터 처리 기능을 활용하여 Kafka에서 가져온 raw data 가공하여 실시간 피드를 위한 Cassandra DB에 저장하기
- Trend 대시보드를 위한 Cassandra DB 데이터 분석 후 Mongo DB에 저장하여 keyword API로 전달하기
- Kafka와 Cassandra DB 사이에서 유실 없는(최대 20%의 유실) 데이터 파이프라인 구축하기
- 개인 목표 달성을 위한 리스크는 데이터 처리에 대한 경험 부족으로, Spark 기술을 찾아보면서 공부했던 내용들 Gitlab에 1일 1commit 하기
- 데이터팀 내 Spark를 담당하지만, 앞단의 데이터 수집과 이후 DB저장까지 데이터 파이프라인의 전체적 흐름을 파악할 것. 나머지 데이터팀 팀원들과 협업하여 각자 맡은 부분이 어떻게 작동하는지와 어떻게 서로 연결하는지에 대해 웹개발 담당 팀원들에게 자신있게 설명할 수 있을 정도로 정확히 숙지할 것.

04 개인 목표 – 기술 스택 및 R&R

이규은

빠른 데이터 CRUD를 가능하게 하여 실시간성을 확보하고,
장애 발생 시에도 안정적으로 데이터를 공급할 수 있도록 한다.

기능 및 역할



- 가공 데이터 카산드라 DB에 분산 저장 및 전달
- 데이터 모델링
- 복제된 데이터의 유효성 검사 및 관리
- 퍼포먼스 향상을 위한 DB 최적화

개인 목표

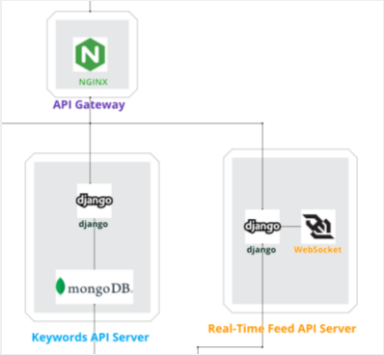
- 분산 데이터베이스를 이해하고 그에 맞는 올바른 데이터 모델링을 통해 빠른 데이터 CRUD가 가능하도록 하여 실시간성을 확보한다.
- 확장성을 고려하여 대용량의 데이터가 들어와도 안정적으로 데이터를 분산 저장하고 빠르게 전달할 수 있도록 DB를 설계한다.
- 복제된 데이터에 대한 유효성을 검사하고 및 복제 계수 등 파라미터를 관리하여 데이터의 상태를 최신으로 유지한다.
- 장애 발생 시에도 복제된 데이터 등을 통해 안정적으로 데이터의 저장 및 공급이 가능하도록 한다.
- 데이터 파이프라인의 전반적인 흐름을 이해하기 위해 카산드라 DB 뿐만 아니라 다른 영역과의 인터페이스 등에 관해서도 꾸준히 공부하여 데이터팀 gitlab에 습득한 내용을 기록한다.

04 개인 목표 – 기술 스택 및 R&R

정석준

실시간 데이터를 제공하는 서비스에 있어서 각 기능을 철저하게 독립적으로 분리하여 안정적인 서비스 환경을 제공할 수 있도록 서버를 구축한다.

기능 및 역할



- Realtime Feed API 서버 구축. 웹소켓을 통한 실시간 데이터 전송 및 API를 통한 과거 트윗 전송 기능 구현
- Keyword API 서버 구축. 키워드별 데이터 전송과 캐싱을 통해 효과적인 실시간 데이터 전달 기능 구현
- 요청에 대한 각 기능별 인증, 권한 검사
- 기능 분리와 API GATEWAY를 통한 MSA 구조 구축

개인 목표

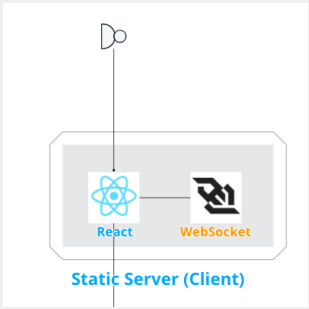
- 목표1. 웹 개발 전반에 있어 백엔드 서버를 성공적으로 구축하여 프론트 서버나 다른 API 서버들과 맞물려 돌아가는 전체적인 시스템 이해하기 → 리스크1. 아직 웹 전반에 대한 개념과 경험이 없다. → 수행1. 장고 웹프레임워크를 기준으로 최대한 다양한 기능 구현을 시도해보며 각 기능을 구현하는 방식과 돌아가는 시스템을 이해할 수 있도록 노력하기 → 관리1. 시도하고 구현한 부분에 대해 문서로 정리해놓기
- 목표2. 다른 부분을 개발하는 개발자와 협업하여 약속된 REST API를 개발하고 성공적으로 전송할 수 있는 역량 갖추기 → 리스크2. REST API에 대한 개념 이해 충분히 없음. 명세에 대해 다른 사람과 협업으로 의견을 맞춰본 경험 없음. → 수행2. DRF을 기반으로 학습하여 하나의 REST API를 성공적으로 개발하는 흐름 익히기. 프론트 개발자와 함께 협업을 통해 해당 과정에서 필요한 사항들을 이해하고 경험을 정리하기 -> 관리2. 명세에 맞추어 모든 API가 계획대로 반영되었는지 테이블로 정리하기
- 목표3. HTTP나 Websocket과 같은 통신 전반에 대한 이해도를 높이기 → 수행3. HTTP 기본 전송 형태에 대해 개념적인 학습도 진행하고 WebSocket과 같은 부분에 있어서 django channels를 활용하여 최대한 많은 부분을 실제로 구현해본다. 실시간 처리에 있어 각 통신 규약별 기능에 대한 차이 이해하기
- 목표4. MSA구조를 구축하여 연동 부분에 있어 문제가 없이 잘 구축해낼 수 있는 능력 갖추기 → 리스크4. MSA 이전에 애초에 기본 단일 구조에 대해서도 개발 경험이나 이해가 없음. → 수행관리4. 개발 과정에서 순서대로 통일된 기능을 구현하고 나눌 수 있는 부분은 최대한 분리된 모듈로 구축해보며 두 방식 모두 이해할 수 있도록 시도해보기

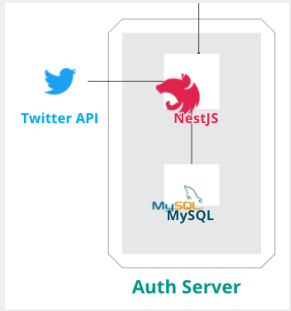
04 개인 목표 – 기술 스택 및 R&R

추연호

> 웹 앱 결과물 뿐만 아니라, 디자인시스템을 함께 만들어 팀 내 소통 가능하고, 쉽게 확장할 수 있도록 개발하기

기능 및 역할





- 트렌드 대시보드 형태의 웹 애플리케이션 구현
- 차트 라이브러리를 사용한 통계 데이터 시각화
- 웹소켓 통신으로 실시간 피드 구현
- JWT 토큰 기반 인증 서버 구현 및 소셜 로그인

개인 목표

- 대시보드 웹 애플리케이션(Web app)을 기획 및 디자인부터, 개발까지 담당하여 완성하기
- 기존 백엔드를 ExpressJS로 구성했을 때 느꼈던 구조를 잡는데 불편했던 경험과 TypeScript로 migration하면서 느낀 불편함을 NestJS를 사용하며 개선할 수 있는지 학습 및 적용하기
- API Gateway에서 Token 기반 인증을 적용하면서 확장가능한 MSA 구조 경험하기
- Adobe XD 또는 Figma 등 프로토타입 툴을 사용하여 컴포넌트 및 테마를 디자인하고, Storybook을 사용해 문서화한 디자인시스템을 개발하여 팀끼리 또는 앞으로 디자이너, 개발자가 협업하는데 도움이 되는 실전 경험 터득하기 → Storybook의 다양한 addon들을 적용해보며 유용한 addon들 파악하기, CSS box-shadow 제대로 이해하기
- 데이터 시각화 - 리액트 차트 라이브러리를 사용하여 꺾은 선 차트, 막대 차트, 도넛 차트 만들기 → 더 나아가서 직접 svg를 활용해 차트 구현해보기
- React에 웹소켓을 접목시켜 실시간 피드를 구현하는 방법을 터득하고, 블로그 글 작성하기
- 상태 관리 라이브러리(Redux or Recoil or MobX)를 각각 사용하여 토이 프로젝트를 해보고 비교하며 장단점 파악 및 마음에 드는 보일러플레이트 구성하기
- TypeScript + React 프로젝트에 ESLint + Prettier + husky 등을 적용한 나만의 react-app boilerplate를 구성하고(오픈소스화하고 블로그 글 작성하기), 팀 내 브랜치 및 Commit 규칙에 따라 GitLab으로 레포지토리 및 소스 코드를 깔끔하게 관리하기
- 이번 프로젝트를 하면서 새롭게 접한 기술이나 어려웠던 이슈 해결에 대해 문서화하기 (최소 5개)

05 프로젝트 관리 계획 – 매니지먼트 요소 및 관리 방안

시간 및 개발 일정

- 팀 프로젝트 기간을 단계별로 나눠 각 단계의 목표를 정하고, 각 목표에 맞춰 각자의 기능을 개발한다.
- 해야 할 일의 우선순위를 파악하고 작은 단위로 일을 나누어 체계적으로 진행한다.
- 각자 약속한 개발 일정을 준수하자. (일정을 지키지 못하면, 납득할 수 있는 이유가 있어야 한다.)
- 매일 스크럼미팅을 통해 각자의 진행 상황을 공유한다.

경험 부족

- 이슈와 학습 내용에 대해 체계적이고 자세하게 기록하고 개발에 적용한다. (JIRA와 gitlab 활용)
- 구조적인 고민, 성능 및 최적화에 대한 고민보다는 우선 구현할 수 있는 것부터 구현하고 나중에 개선한다.
- 리서치를 통해 공유하면 좋은 자료는 함께 공유한다.

인터페이스

- 인터페이스를 맞춰야 하는 당사자간 회의를 통해 의논한다.
- 개발 중에 다른 사람과 협의해야 하는 문제가 있다면, 빨리 문제 상황을 의논한다.

에러 해결

- 혼자 고민하지 말고 팀원의 도움을 구하기 위해 적극적으로 질문한다. (알려주는 사람이 더 배울 수 있도록)
- 에러 해결 후 notion에 공유하여 동일한 문제 해결에 대해 다른 팀원들에게 도움을 제공한다.

06 프로젝트 관리 계획 – 협업 도구

소스 코드 관리

개발한 소스코드 공유 및 형상관리

이슈 관리

개발 일정에 따라 Sprint 생성 후 각자 개발 사항 Issue 등록 후 개발 진행

공유 문서 관리

회의록, PMP Code Convention 등 문서 작성 공유

메신저, 온라인 미팅

오전 Daily Scrum 및 회의 진행