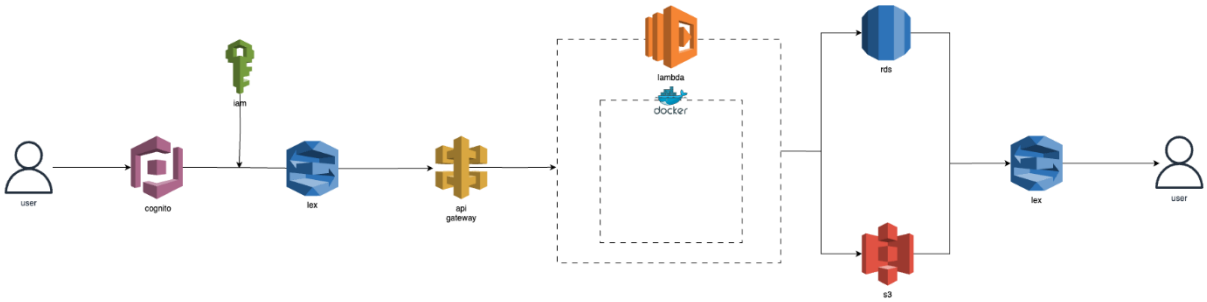
6주차 이전 : 팀빌딩 이후 팀 내에서 한승민 학우와 함께 백엔드 파트를 담당하게 되었다. 팀원 중 AWS 서비스를 사용해 본 경험이 있는 팀원이 없었고, 본인의 희망 분야를 정하지 못하여 어떤 분야든 배우면서 진행해보려는 생각이 있었기에 AWS관련 백엔드 파트를 구현하게 되었다. 6주차 이전까지 팀원들과 주제 선정, 아키텍쳐 설계 등을 하는 과정에서 AWS 서비스, 앱 개발 과정에서의 기초 지식 등이 많이 부족함을 알게 되었고, '업무에 바로 쓰는 AWS 입문'이라는 도서를 구매하여 AWS의 주요 서비스에 대한 기본적인 역할과 사용법 등에 대해 공부하였다. 그 후 기존에 구성했던 Technical Architecture를 재구성하였다. 아키텍쳐는 이후에도 꾸준히 수정 과정을 거쳤다.



RDS, S3 그리고 이후 AWS 서비스를 사용하게 될 사용자를 생성하고, 필요한 역할을 각 사용자에게 부여하였다.

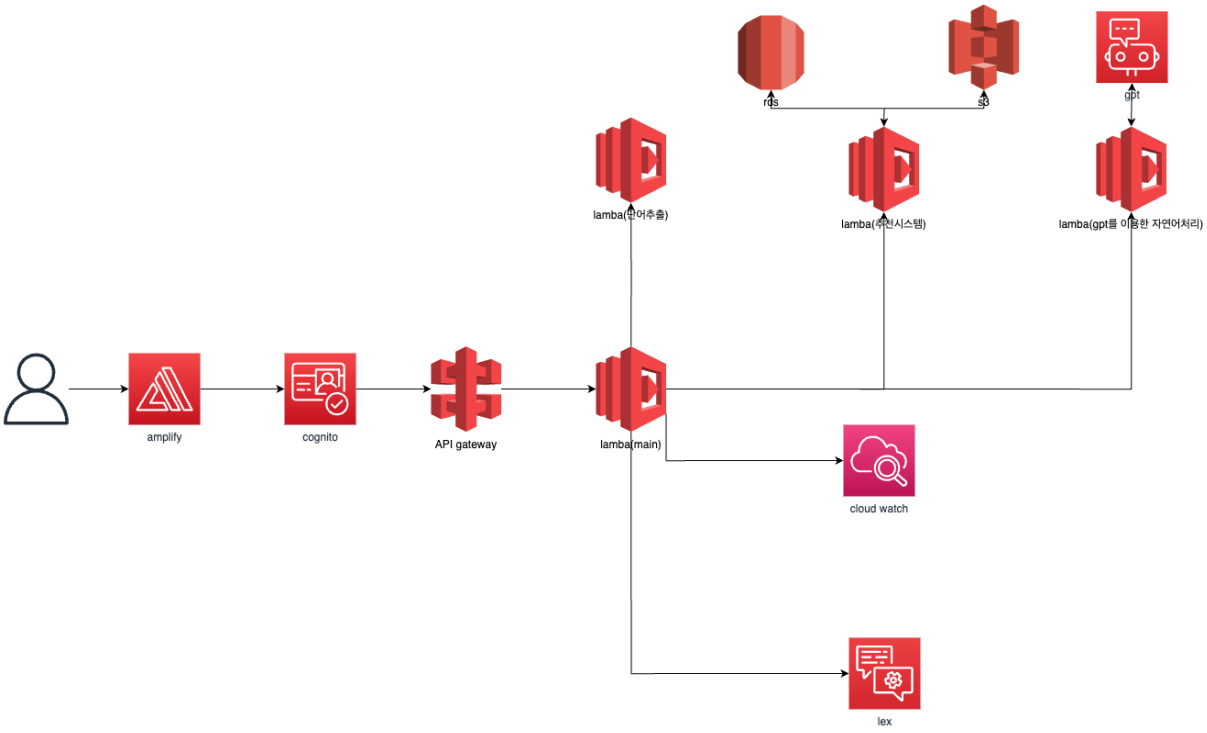
사진을 저장하게 될 S3 스토리지 객체를 만든 뒤 데이터베이스 관리를 위한 RDS 인스턴스를 생성하려 했으나, RDS에도 MySQL 데이터베이스를 연동하는 과정에서 VPC 관련 오류가 발생하여 아직 만들지는 못하였다. 학교 공지사항을 크롤링한 사진이 있다면 S3에 올리고 확인해 보려 했으나 아직 결과로 나온 사진이 없어 임의의 사진으로 업로드만 해 보았다.

발표 이후, 'ios 앱 프론트와 연결할 방안이 아키텍쳐 구성에 포함되어 있지 않다'라는 피드백을 받았다. 이를 해결하기 위해 AWS amplify라는 서비스를 사용하여 서비스를 통합하고 배포하는 데 사용해보기로 했다.

RDS를 VPC와 연결하려 했으나, RDS에서 연동할 VPC가 보이지 않고 기존에 만들었던 VPC를 수정하기도 어려워 결국 RDS와 VPC의 옵션 등을 설정하여 재생성하고 연결하였다.

8주차 발표에서 챗봇기능을 구현하는 데 있어서 QnA maker라는 서비스에 대한 추천을 받았고, QnA maker의 사용을 위해 알아본 결과, 곧 서비스 종료가 예정되어 있어 기존 서비스 이외에 신규 가입자는 사용이 제한된다는 것을 알게 되어, GPT API를 사용하기로 하였다. 이에 맞춰 AWS EC2를 사용하여 구성을 할지 혹은 AWS lambda만을 사용하여 서버리스로 프로그램을 만들 수 있을지에 대한 토의를 하였고, lambda에 올릴 수 없는 크기의 코드는 docker 이미지파일을 람다에 올리는 것으로 하여 진행하기로 하였다. 이에 맞춰 AWS lex를 사용하고, 각각의 기능에 대한 람다

함수를 만들기로 한 것을 바탕으로 아키텍쳐를 재구성하였다. 이 과정에서 사용자의 의도를 어떻게 파악할 것인지, 그 의미를 다른 기능에 어떻게 전달할 것인지에 대해 어려움이 있었다.

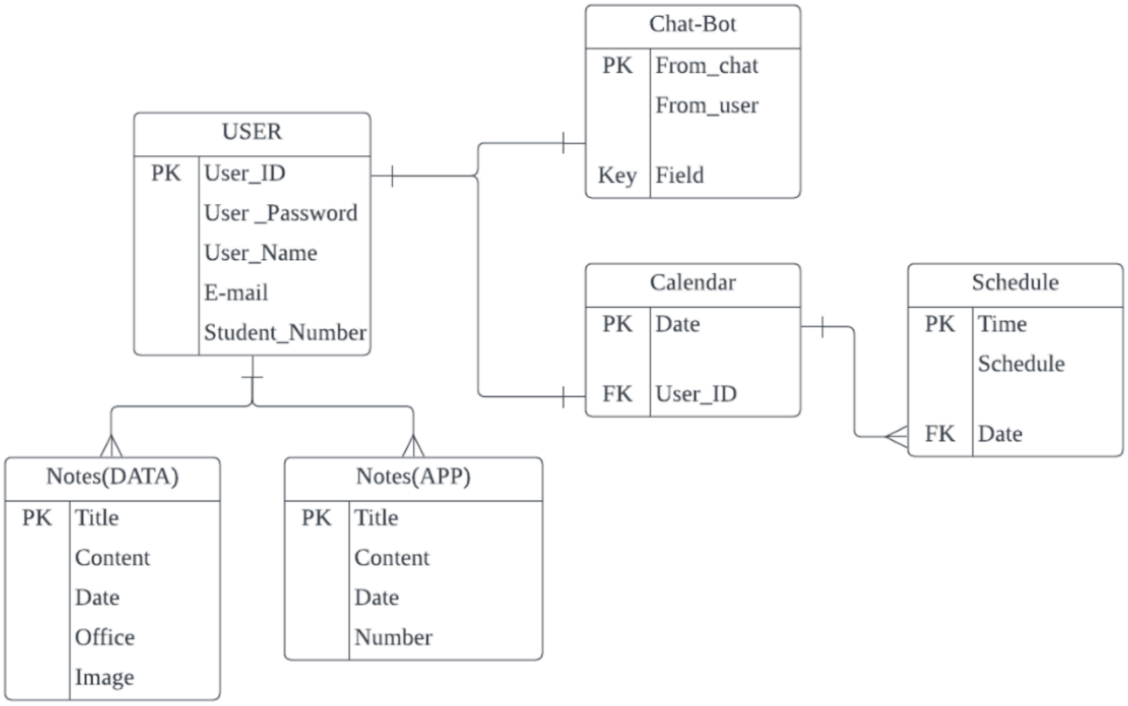


또한 DBeaver 툴을 이용해 RDS와 연동해 놓은 데이터베이스에 우리 학과에 대한 기본적인 정보(이름, 정원, 과사무실 번호 등)을 담은 간단한 더미데이터를 생성한 뒤 람다 함수에서 데이터를 가져오는 함수를 작성하였다.

9주차에 들어서면서, 프론트엔드에서의 요청을 처리하기 위한 API 명세서를 노션을 활용하여 만들고, 회원가입, 로그인 API부터 생성하여 해당 기능들을 구현하려 했으나, API gateway와 람다를 통해 간단한 API를 만드는 것도 하지 못하여 첫 1~2주 동안 API에 대한 기초적인 내용들을 찾아보고, API 호출을 주고받는 원리 등에 대해 알아보는 시간을 많이 가졌다.

회원가입과 로그인 기능은 AWS의 cognito와 amplify를 연동하여 구현하려 했고, cognito에서 사용자 풀을 정의하고 기타 설정을 마친 뒤 로그인/회원가입 과정이 이루어 지는 동안 인증이 이루어지는 방식 등에 대해 알아보면서 시도를 해 보았으나 해결이 되지 않아서 앞으로 개발하는데 더욱 필요한 기능을 먼저 구현하기로 하고 미뤄두었다가 결국 구현하지 못하게 되었다. 6월이 되고 나서 간단하게라도 데이터베이스에 사용자 정보를 저장하고, api 호출을 통해 비교하는 과정으로라도 구현을 해 보았으나, 보안 등의 이유로 넣지 않기로 했다.

데이터들 간의 관계를 정의하는 ERD 설계도 진행하였다. 당시 아키텍쳐 구성을 토대로, 각 기능에서 사용할 데이터들에 대해 정리를 하고 이후에 데이터베이스 테이블을 만들고 기능을 구현하는데 큰 도움이 되었다.



중간발표 이후, 회의 결과 Chat-gpt의 API를 사용하게 되면 AWS LEX를 사용할 필요가 없다는 결론에 도달하였고, LEX를 개발 계획에서 제외하기로 결정했다. API를 계속해서 제작하는 과정에서 GET 호출은 문제없이 진행되었으나, POST 관련 호출을 진행하는 데 있어 HTTPMethod가 전달되지 않는 문제가 지속적으로 발생하는 오류가 있었고, 결국 api gateway에서 lambda proxy 통합 설정을 해제하고 매핑 탬플릿을 직접 지정해줌으로서 이 문제를 해결하였다.

이후 API의 제작 및 수정 작업을 꾸준히 거치면서, 앱의 각 기능에 대한 람다 함수를 구현하기 시작했다. 사용자의 질문을 토대로 단어를 추출하는 코드와, 그 질문을 통해 유사도가 높은 공지

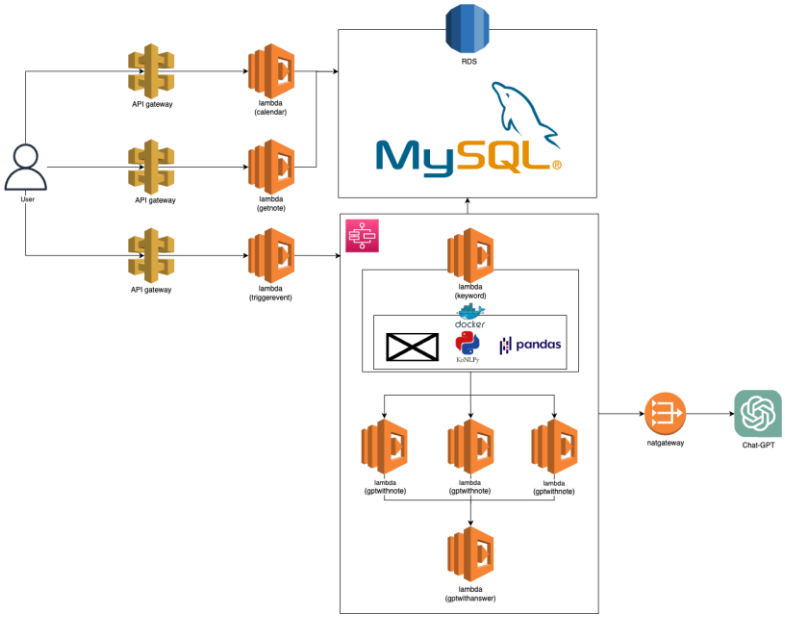
사항에 대한 정보를 반환하는 코드를 작성하였으나 람다 함수에서는 지원하지 않는 패키지를 사용하는 바람에 docker 파일로 이미지를 만들고 이를 람다에 올려서 구현하기로 했다. 그리고 이들 기능을 구현하는 main 역할을 하는 람다 함수를 진행하고, 이 람다 함수에서 각 기능을 호출, 실행시키기로 했다.

후에 이 두 함수를 하나의 도커 파일로 합쳐 하나의 기능으로 통합하였고, 프론트 측 팀원들과 대화를 통해 api의 파라미터와 리턴값에 대해 정리하였다.

GPT 모델이게 질문을 주고 이에 대한 답을 받아오는 함수를 구현하였고, 유사도가 높다고 뽑힌 공지사항 3개의 본문 내용을 입력으로 실행하려 했으나, API 호출 시 전송가능한 토큰의 수를 초과하는 바람에 에러가 나는 경우가 있었다. 이를 해결하기 위해 공지사항의 url 정보도 데이터베이스에 추가하고, GPT 모델에게 url을 보내어 답변 및 요약을 요구했으나, 모델이 url 주소에 접속하지 못하는 경우가 많아서, 공지사항 3개에 대한 질문을 한번씩 총 3번 진행하기로 했다. 이렇게 진행한 경우 질문은 잘 넘길 수 있으나, 모델에게 질문을 던지는 작업을 3번이나 직렬로 수행하다 보니 시간이 너무 오래 걸려 api 응답 대기 시간을 초과하는 경우가 발생하였다. 이를 해결하기 위해 AWS SQS와 AWS EventBridge를 사용하면, 사용자의 질문이 들어오면 이벤트를 발생시키고 메시지를 전달하여 그 메시지 내용을 토대로 각각의 람다 함수들을 병렬적으로 실행할 수 있을 줄 알고 이를 진행하였다. 하지만 SQS와 이벤트브릿지에 대한 이해를 잘못 하였고, 우리가 원하는 방식으로 람다 함수들을 호출할 수 없다는 것을 알게 되어 추가적인 조사를 하였다. 그 결과 AWS Step function이라는 서비스를 알게 되었고, 각각의 람다 함수의 반환값을 다음 단계의 입력으로 하여 여러 단계를 순차적으로 진행할 수 있다고 하여 이 서비스를 사용하기로 하였다. 기존 방식은 사용자의 입력을 api 호출을 통해 람다의 입력으로 들어오면, 이후 함다 함수들을 각각 api로 호출하여 결국 모든 람다 함수가 끝이 나야 처음에 호출한 api가 응답을 하는 과정이었다면, 최종 수정된 방식은 사용자의 입력이 들어오면 step function을 통해 병렬적으로 최대한 빠르게 실행을 하고 step function이 진행된 결과 중 원하는 값을 반환하는 방식이다. 이에 step function의 상태머신 정의를 만들고, 각 람다 함수들을 순차적, 혹은 병렬적으로 진행하게끔 구성하였다. 이후 예외가 발생하는 경우에 대한 예외처리를 진행하였다.

초반 4, 5월 초까지 기초적인 내용에 대한 학습, 준비 등으로 시간을 오래 할애하였고, 이후 개발 과정에도 영향을 미쳐 오랜 시간이 걸리게 되었다. 모르는 것이 많아 생각했던 것들을 모두 구현하진 못했지만, aws 서비스 사용이나 ERD설계, API 등 이번 산학 프로젝트를 통해 처음 접하게 된 작업들로 정말 많이 배울 수 있었고 이를 토대로 앞으로 개발자가 되기 위한 좋은 양분이 되리라 생각한다. 중간에 팀원의 이탈이 있었지만, 남은 팀원들끼리 협동하여 끝까지 더욱 열심히 해서 결과를 맺을 수 있었다고 생각한다.

<최종 아키텍쳐 구성>



<최종 ERD>

calendardata	
PK	idx
	userid
	year
	month
	date
	time

chatdata	
PK	idx
	userid
	sender
	content
	similarity

KAUinfo	
PK	Index
	Noun
	title
	content
	office
	date

bbsoutput	
PK	Index
	Noun
	title
	content
	office
	date

notioutput	
PK	Index
	Noun
	title
	content
	office
	date

soft	
PK	Index
	Noun
	title
	content
	office
	date

totaloutput	
PK	Index
	Noun
	title
	content
	office
	date

{User_ID}_table	
PK	Index
	Noun
	title
	content
	office
	date
	notename