**프로젝트 정의서**

20150224 김우석

주제 : 숭실대학교 학식 정보 제공 카카오톡 챗봇

이번 프로젝트로 숭실대학교 학식 정보를 알려주는 카카오톡 챗봇을 개발했습니다. 해당 주제를 프로젝트 주제로 선정하게 된 이유는 불편함을 줄이고 싶었습니다.

**목표 : 학식 정보 제공 간편화  
필요성 : 학생들이 불편함을 느낌**

저를 비롯한 학교 주변에서 거주하는 많은 학생들이 학식을 자주 이용합니다. 하지만 학식 메뉴를 알기 위해서는 ‘에브리타임’이라는 대학생 커뮤니티 어플리케이션을 접속해서 여러 페이지를 거쳐야 합니다. 또, 기숙사 식당의 메뉴는 어플리케이션 내에서 제공되고 있지 않아 유저가 기숙사 홈페이지에 직접 들어가서 캡처를 하고 업로드하고 있습니다. 그래서 유저가 자발적으로 메뉴를 업로드하지 않으면 학생들은 기숙사 메뉴를 보기 위해 직접 홈페이지에 접속하거나, 그냥 식당으로 향하는 수 밖에 없습니다. 누군가 업로드를 했다 하더라도 해당 게시글이 일정 추천수 이상 받지 못한다면 어플리케이션 메인 화면의 ‘화제글’에 보이지 않아 정보 게시판까지 직접 접속해야 하는 번거로움이 있습니다.

저 또한 불편함을 느끼고 ‘누군가 편한 시스템을 개발하지 않았을까’하는 생각으로 검색을 해 본 결과 카카오톡에 ‘숭실대학식’이라는 채널이 있었지만 2019년 12월 31일 부로 카카오측에서 플랫폼 교체 작업을 하여 기존의 API를 카카오 i 오픈빌더라는 인공지능 플랫폼으로 변경했고, 그에 따라 서비스가 중단된 것을 확인할 수 있었습니다.

마땅한 대체재가 없이 약간의 불편함을 감수하고 에브리타임 어플리케이션으로 학식 정보를 확인하고 있던 와중에 해당 주제가 네트워크 프로그래밍의 수업 내용과도 잘 맞고, 학생들의 불편함을 해소할 수 있다는 취지도 의미 있는 것 같아 이번 프로젝트의 주제로 선정하게 되었습니다.

**장점 : 완전한 서비스, 편리한 접근성**

해당 프로젝트의 장점은 완전한 서비스, 편리한 접근성입니다. 에브리타임 어플리케이션도 학식 정보를 제공하고, 어플리케이션이기 때문에 접근성도 좋습니다. 하지만 앞서 말했듯이 기숙사 식당의 메뉴를 자체적으로 제공하지 않는다는 점에서 완전하지 못하고. 여러 페이지를 거쳐야 하기에 약간의 불편함이 있습니다. 따라서 해당 서비스를 완전하게 제공하는 시스템을 개발한다면 학생들이 이용할 가능성이 있다고 생각했고, 많은 학생들이 이용하기 위해서는 어플리케이션보다 접근성이 좋아야 한다고 생각했습니다.

만약 어플리케이션으로 개발한다면 이용하기 위해 우선적으로 설치를 해야 하는데 잘 모르는 어플리케이션을 설치하는 것을 기피하는 사람들도 있다는 생각을 했습니다. 그래서 카카오톡 채널 챗봇으로 구현하는 것을 택했습니다. 대형 서비스인 카카오톡의 채널로 구현한다면 설치할 필요도 없고 채팅 한 번이면 원하는 학식 정보를 얻을 수 있기에 접근성도 어플리케이션보다 좋다고 판단했습니다.

**수업과의 연관성 : DNS, HTTP, 클라이언트-서버 구조**

사용자가 카카오톡 채널에 발화를 입력하면 챗봇이 REST API를 호출하여 http request를 AWS lambda의 함수로 전달하고 이 함수는 학식 정보를 JSON 형식으로 http response에 저장하여 반환합니다. 또한 매주 월요일에 학식 정보를 기숙사 웹사이트와 학교 웹사이트로부터 스크래핑하여 데이터베이스에 저장합니다. 이 과정에서 웹사이트에 도메인 이름으로 접근하여 http를 fetch하는 것이 포함됩니다. 또한 클라이언트-서버 구조로, 채팅을 입력하는 사용자가 클라이언트가 되며 서버는 lambda 함수를 통해 클라이언트에게 서비스를 제공합니다.

**개발 환경 : python 3.7.9, 파이썬 오픈소스 프레임워크 scrapy, AWS lambda, AWS API Gateway, AWS dynamoDB, AWS EC2, cron, 카카오 i 오픈 빌더**

우선 프로젝트는 serverless 구조로 설계했습니다. serverless는 서버가 없는 것이 아니라 서버를 클라우드에서 구동함으로써 관리하지 않아도 서비스를 제공할 수 있는 것을 의미합니다. serverless에는 두 종류가 있는데, BaaS와 FaaS입니다.

BaaS는 Backend as a Service로, 백엔드 자체를 API로 제공해주는 서비스를 의미하며 대표적으로 firebase가 있습니다. FaaS는 Function as a Service로, 이벤트가 발생하면 매핑된 함수가 호출되어 서비스를 제공하는 것을 말합니다. 대표적으로 AWS lambda가 있습니다.  
 제가 FaaS 방식을 선택한 이유는 다른 방식이 서버를 구동하는 시간에 따라 비용을 지불한다면, FaaS의 경우 사용한 양에 따라 비용을 지불합니다. 또 서버 확장이 자동으로 이루어집니다. 그래서 챗봇과 잘 맞을 것이라고 생각되었습니다.

학식 정보를 알려주는 서비스이기 때문에 특정 시간에 트래픽이 몰립니다. 다른 시간에는 상대적으로 호출이 적을 것입니다. 이 때 서버를 구동하는 방식으로 구현한다면 호출되지 않는 시간은 낭비가 될 것입니다. 그래서 사용자가 챗봇에 발화를 입력해 함수를 호출할 때만 서비스를 제공하는 구조를 택했습니다. 사용량의 경우 AWS lambda는 월 100만 건까지는 무료이기 때문에 한 사람이 한 달에 100회를 호출한다고 해도 10000명의 사용자를 수용할 수 있으므로 비용에서 문제가 되지 않을 것이라고 생각했습니다.  
 lambda를 호출하는 이벤트는 AWS API Gateway가 담당합니다. REST API를 지정하고 URL로 요청이 들어오면 lambda 함수를 실행한 뒤 응답을 반환합니다. 요청이 들어오는 경우는 사용자가 카카오톡 채널에 발화를 입력하는 것입니다.

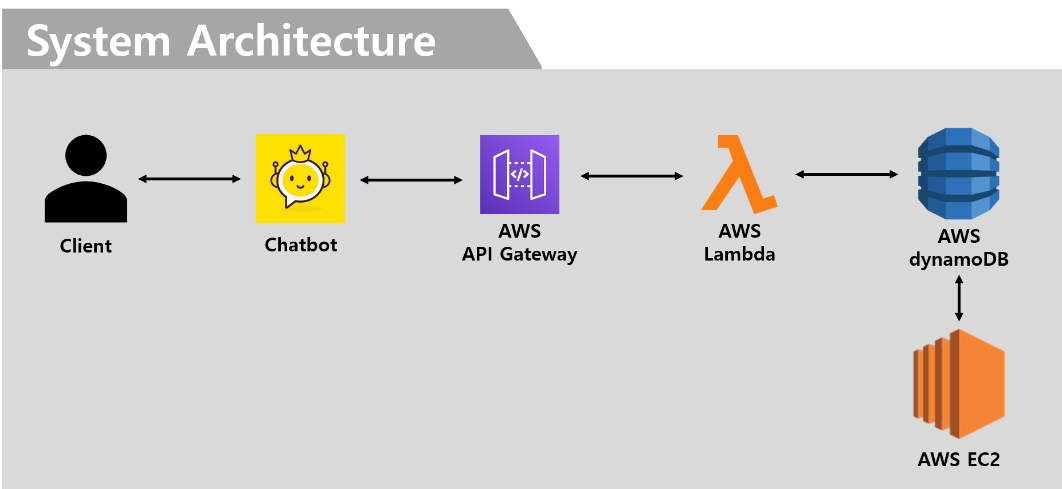
개발 언어로 python을 선택한 가장 큰 이유는 AWS lambda에는 ‘cold start’라는 개념이 존재합니다. lambda 함수를 호출하면 실행 환경으로 라우팅이 된 후 요청이 처리됩니다. 한 동안 함수를 실행하지 않았다면 실행 환경을 구성하기 위해 컨테이너가 생성됩니다. 함수가 호출되고 컨테이너가 생성되기까지의 지연 시간을 ‘cold start’라고 합니다. 그리고 실행 환경이 구성되면 ‘warm start’가 되어 코드를 실행합니다. 그리고 ‘warm’ 상태는 약 10여 분간 지속되다가 다시 ‘cold’ 상태로 돌아갑니다. 이 ‘cold start’의 지연 시간이 가장 짧은 언어가 python입니다.  
 그리고 부수적인 이유로는 python에 웹 크롤링 및 스크래핑을 하기 위한 라이브러리들이 많이 있기 때문이었습니다. 그 중 scrapy를 선택한 이유는 프레임워크라서 사용법은 쉽지 않지만 확장성이 좋아 데이터를 가공하기에 편리했기 때문입니다.

또한 dynamoDB를 데이터베이스로 사용한 이유는 AWS RDS의 경우 lambda에 추가적으로 라이브러리를 설치해줘야 하지만 dynamoDB는 boto3라는 AWS 라이브러리에 이미 내포되어 있어 lambda로 사용하기가 용이했기 때문에 선택하게 되었습니다. 그리고 관계형 데이터베이스는 연결 기반이기 때문에 빠르게 호출되고 종료되는 lambda 함수와 궁합이 맞지 않기 때문에 NoSQL안 dynamoDB를 선택한 점도 있었습니다.

AWS EC2는 cron을 통해 매주 월요일마다 자동으로 학식 정보를 스크래핑하기 위해 사용했습니다. 서버를 하나 구동하고 scrapy 코드를 월요일에 실행해주는 역할을 담당하고 있습니다. 하지만 코드 하나를 돌리기 위해 서버를 구동하는 것은 비효율적이라서 추후 수정해야 할 부분이기도 합니다.

카카오 I 오픈 빌더는 카카오톡으로 학식 정보를 전달하기 위해 사용했습니다. 사용자가 카카오톡 채널에 발화를 입력하면 카카오톡 챗봇이 그것을 lambda의 REST API로 POST 방식으로 전달합니다. 이 때 request body에는 JSON 형태로 다양한 데이터가 오지만 본 프로젝트에서 사용하는 정보는 발화 내용 하나입니다. 전달되면 lambda 함수가 실행되고 발화에 맞는 적절한 응답이 반환됩니다. 카카오톡 챗봇은 이를 가공해서 사용자에게 카카오톡 채팅으로 학식 정보를 전달합니다.

**동작 과정**



전체적인 구조도는 위와 같습니다. 전체 과정을 설명하겠습니다.

우선 사용자가 카카오톡 채널인 ‘슈밥’에 발화를 입력합니다. 챗봇의 패턴으로 등록된 발화는 ‘월’, ‘화’, ‘수’, ‘목’, ‘금’, ‘토’, ‘일’, ‘오늘’, ‘내일’, ‘처음으로’, ‘요일별’의 총 11개입니다. 이들을 제외한 발화를 입력하면 챗봇은 ‘풀백 블록’을 전달하여 이해를 못했다는 의사를 표현합니다.

정해진 발화를 입력하면 챗봇은 등록된 ‘스킬 서버’로 HTTP POST 요청을 합니다. 본 프로젝트의 ‘스킬 서버’는 AWS lambda를 호출하는 AWS API Gateway입니다. body에는 유저나 발화 등의 정보가 포함되어 있습니다.

AWS API Gateway는 API 키를 검사한 뒤 AWS lambda를 실행하면서 input으로 request를 전해줍니다. 만약 한 동안 함수가 실행되지 않았으면 실행 환경을 구성하는 ‘cold start’ 지연 시간이 생길 것입니다. 실행 환경이 구성되었으면 lambda는 사용자의 발화 내용에 맞는 적절한 응답을 챗봇이 이해할 수 있는 JSON 형태로 구성합니다. 응답은 사용자의 발화가 ‘처음으로’나 ‘요일별’일 경우 추가 정보를 요청하는 JSON으로 구성됩니다. 나머지 9개의 발화일 경우 학식 정보를 dynamoDB로부터 가져와서 사용자가 읽기 쉽게 가공한 뒤 JSON에 포함됩니다.

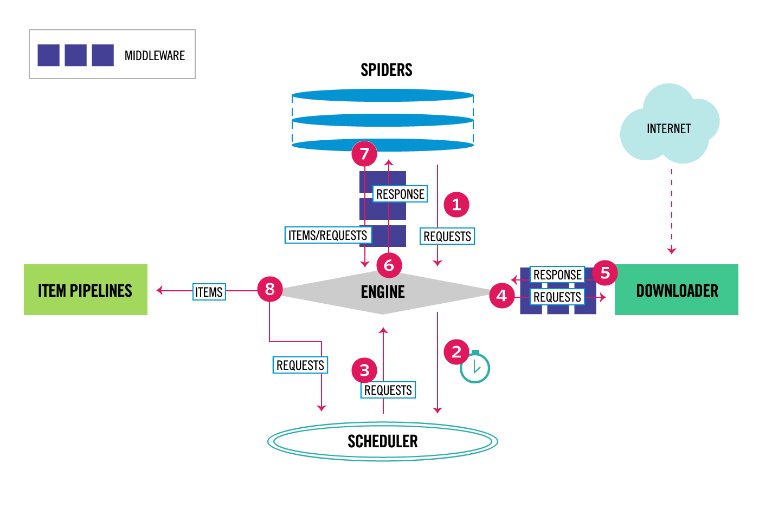
이 학식 정보는 24시간 구동되고 있는 AWS EC2에서 cron을 통해 매주 월요일 02시에 웹사이트를 스크래핑한 뒤 dynamoDB에 저장되도록 설계되어 있습니다. 학교 웹사이트의 학식 정보가 일요일에 업데이트되는데, 그 시간이 일정하지 않아 월요일 자정에 구동될 수 있도록 했습니다.

그렇게 lambda가 응답을 반환하면 챗봇은 응답을 파싱하여 유저에게 전달합니다. 그럼 사용자는 학식 정보를 제공받을 수 있게 됩니다. 전체 구동 과정은 이렇게 끝이 납니다.

**API 설명**

이번엔 사용된 오픈 소스 API를 설명하겠습니다.

**scrapy**



핵심 모듈로는 spider, engine, scheduler, pipeline, middleware, downloader가 있습니다. scrapy의 대략적인 동작 과정은 다음과 같습니다.

1. scrapy engine이 spider로부터 request를 받습니다. spider은 사용자가 작성한 스크래핑 logic에 따라 웹페이지를 스크래핑하는 클래스입니다.   
2. engine이 request를 scheduler에게 등록하고 스크래핑할 request를 요청합니다.  
3. scheduler가 request를 engine에게 전달합니다.  
4. engine은 request를 downloader에게 전달하고 이 과정에서 request는 middleware를 거쳐갑니다. middleware에서 지정된 logic에 따라 가공됩니다.  
5. downloader가 request 안의 URL 페이지를 가져와서 response를 생성한 뒤 engine에게 전달합니다. response는 middleware를 거쳐가면서 가공됩니다. 본 프로젝트에서 접근하는 URL은 기숙사 식단 사이트와 학교 식당 사이트입니다.  
6. engine은 response를 받아 spider에게 전달합니다. response는 middleware를 거쳐갑니다.  
7. spider는 response를 logic에 따라 처리하고 item과 새로운 request를 구성하여 engine에게 전달합니다. item은 스크래핑한 데이터를 저장하는 key-value 형태의 자료구조로, 본 프로젝트에서는 날짜를 나타내는 date, 시간대를 나타내는 time, 장소를 나타내는 location, 학식 정보를 나타내는 menu로 key를 지정하였습니다. 새로운 request는 해당 페이지에 다른 링크가 존재하고, spider가 그 링크에 접근할 경우 생성됩니다.  
8. engine은 item을 pipeline에게 전달하고 pipeline은 사용자가 정의한 logic에 따라 item을 가공합니다. 또, engine은 새로 생긴 request를 scheduler에게 전달하면서 다음 request를 요청하고 이 과정을 반복합니다. 본 프로젝트에서 pipeline은 item을 dynamoDB에 저장하는 역할을 담당하고 있습니다.

dynamoDB에 모든 학식 정보를 저장했으면 scrapy는 종료됩니다. 만약 dynamoDB에 접근하지 못했거나 웹사이트를 제대로 스크래핑하지 못했으면 예외가 발생한 것이므로 즉시 종료됩니다. 학생 식당 사이트의 html 태그가 일정하지 않고 매주 다르기 때문에 해당 부분은 잠재적인 에러로 두고 주시해야 할 것 같습니다.

**카카오톡 챗봇**

카카오톡 챗봇은 AWS lambda와 매핑된 AWS API Gateway의 URL을 스킬 서버로 두고 있습니다. 이 스킬 서버로 요청을 전송할 때 헤더에 ‘x-api-key’라는 key를 포함시켜 전송합니다. 해당 key가 없는 요청은 AWS API Gateway로의 접근이 거부됩니다.

응답으로 돌아오는 JSON의 형식이 정해져 있어서, 이를 지키지 않을 경우 챗봇은 사용자에게 아무런 응답을 하지 않고 종료됩니다. 또한 다양한 출력 형식이 있는데, 본 프로젝트에서 사용하는 출력 형식은 SimpleText와 QuickReplies 형식입니다.

SimpleText는 간단한 문자열이 담긴 블록으로, 카카오톡의 채팅 메시지 형식와 같다고 생각하면 됩니다. QuickReplies는 사용자가 누를 수 있는 블록 형태로, 누를 경우 해당 블록과 매핑된 문자열이 사용자 측에서 보내집니다. 즉, 터치 한 번으로 채팅이 입력되는 블록입니다.

**cron**

AWS EC2에서 실행되는 소프트웨어입니다. 본 프로젝트에서는 매주 월요일 02시에 scrapy 코드를 실행하는 역할을 담당하고 있습니다. 그리고 작동 로그를 이메일로 보내주도록 설계함으로써 혹시나 scrapy 코드가 실행되지 않았거나 웹사이트 스크래핑에 실패했을 경우 조치를 취할 수 있도록 했습니다.