S.O.S (소셜 온-오프네트워크 서비스)

요 약

온라인에서 뿐만 아니라 오프라인을 통해서도 사회적 관계망을 형성할 수 있는 S.O.S 를 개발하고자 한다. 이는 기존 SNS 들이 가진 한계점들을 보완하고 장점을 흡수하여 건전한 사회적 관계를 만들기 위한 어플리케이션이다.

1. 서론

1.1. 연구배경

대부분의 사용자들은 온라인 상에서 뿐만 아니라 오프라인상에서도 다양한 사회적 관계형성에 대한 기대를 원하고 있다. 그리고 기존 SNS 에서는 본인과 비슷한 의견과 생각 등을 지속적으로 수집함으로써 편향된 사고를 가질 확률이 매우 높다. 이로 인해 다양한 사람들과 갈등이 생길 확률이 매우 높다. 결과적으로 서로에게 도움을 주고 받는 것에 대해 각박한 사회가 되어가고 있다. 따라서 스타트업 드림의 일환으로 S.O.S 라는 어플리케이션을 개발하여 많은 도움들을 주고 받을 수 있도록 하여 온라인 뿐만 아니라 오프라인에서도 사회적 관계를 만들어 건강한 사회를 만들고자 한다.

1.2. 연구목표

SOS(소셜 온-오프라인 네트워크 서비스)는 온라인상에서 외에도 오프라인에서도 다양한 사회적 관계를 형성하여 서로에게 도움을 주고 받는 것에 각박한 사회에서 벗어나고자 한다. 또한, 기존의 SNS 로 인한 갈등 및 중독 등의 사회적 문제를 해결하고자 한다. 그리고 기존 SNS 에서 본인과 비슷한 의견들을 수집하여 편향적인 사고를 가지는 것에서 벗어나 다양한 생각과 가치관의 컨텐츠를 공유할 수 있는 형태의 SNS 를 개발하고자 한다.

2. 관련연구

2.1. Kotlin

2.1.1. Java 와 Kotlin 의 비교

Java 와 Kotlin 은 모두 정적타입으로 컴파일러에서 Type-Error 를 찾을 수 있다. 또한, 멀티 플랫폼 개발을 할 수 있는 장점을 보유하고 있다. 그리고 모두 객체지향 프로그래밍을 지원하여 class 키워드를 사용할 수 있는 공통점을 가지고 있다.

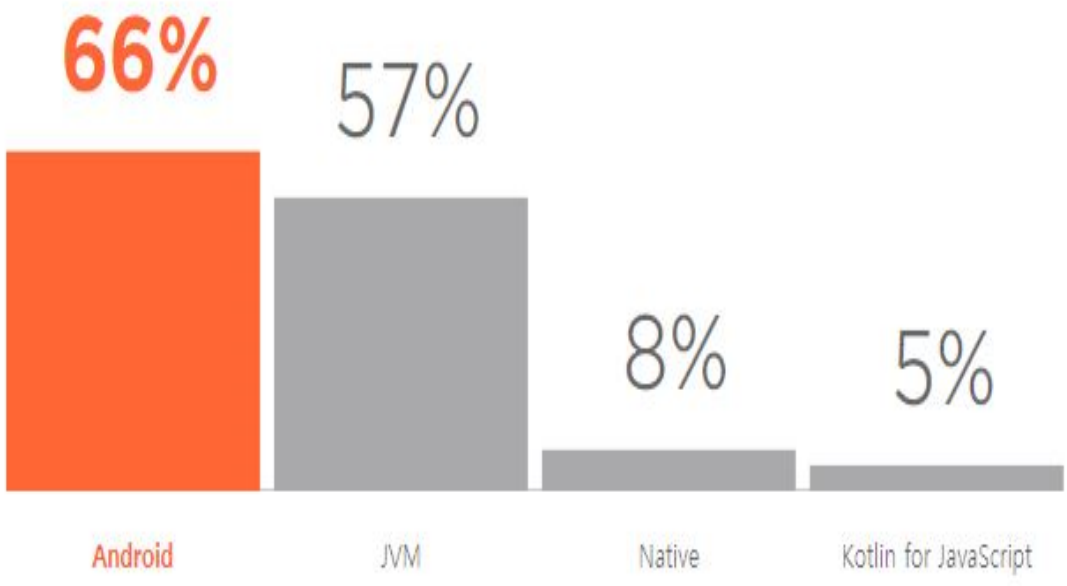
하지만, Java 는 class 가 기본단위로 class 안에서 로직을 만들어야 하는 반면, 코틀린은 class 가 선택사항이다. 따라서 Top-Level 에서 변수, 함수의 선언과 구현이 가능하다. 그리고 자바 9 이하의 버전에서는 타입 추론이 불가능하지만, 코틀린은 타입추론이 가능하다. 마지막으로 Null Point 의 예외처리에서 자바의 경우에는 런타임에 Null 참조시 예외를 발생시킬 수 있다. 그에 반해 코틀린은 Null 을 확인하는 연산자가 존재하여 이를 컴파일 시점에서 방지할 수 있다.

	Type system	Multi-Platform	OOP	FP	Type Inference	Nullable
자바 (Java)	정적 타입	O	O	X	X (9 이하)	X
코틀린 (Kotlin)	정적 타입	O	O	O	O	O

[그림 1] 자바와 Kotlin 비교 표

2.1.2. Kotlin 의 장점

코틀린은 다른 사람이 이해하기 쉽도록 코드를 작성할 때 시간이 단축된다. 이로 인해 코드가 짧으며 가독성이 매우 좋다. 이로 인해 에러발생률을 낮출수 있다. 또한, coroutines, extension funcitons, lambdas 등의 다른 라이브러리와 Android Jetpack 을 지원한다. 그리고 Java 와 호환이 가능하여 기존의 어플리케이션을 모두 Kotlin 으로 손쉽게 바꿀수 있다. 이로 인해 Kotlin 은 2019 년 기준으로 안드로이드나 JVM 에서 대부분 사용하고 있다.

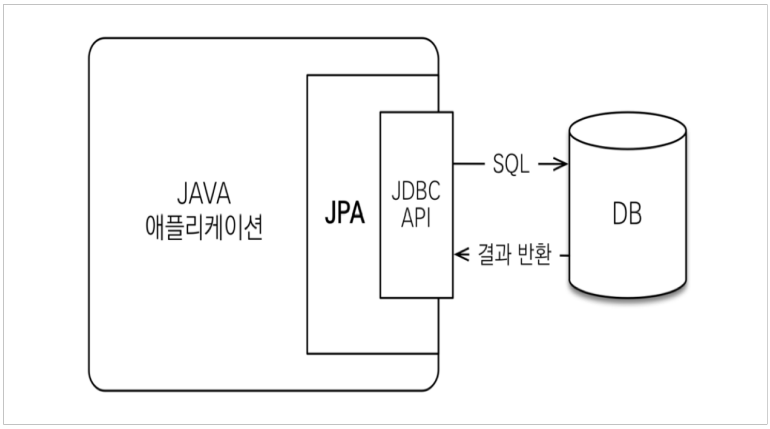


[그림 2] 2019 년 기준 개발자들의 Kotlin 을 언어로 하는 플랫폼 비중

2.2. JPA

JPA는 JAVA의 ORM 기술 표준으로, 인터페이스의 모음을 의미한다. 현재 Hibernate, EclipseLink, DataNucleus의 JPA 2.1 표준 명세를 구현한 3가지 구현체가 존재한다.

JPA는 애플리케이션과 JDBC 사이에서 동작하며, 개발자가 JPA를 사용하면 내부에서 JDBC API로 SQL을 호출하여 DB와 통신하도록 한다.



[그림 3] JPA의 동작과정

JPA는 코드의 무한반복, 객체와 관계 DBMS의 차이 등 SQL 중심적인 개발에서 발생할 수 있는 문제점을 해결하고자 한다. 또한, 간단한 CRUD 구문을 통해 생산성을 높일 수 있고, 그로 인해 유지보수도 용이하게 할 수 있다. 그리고 Object와 RDB간의 패러다임의 불일치를 해결하고, 성능을 최적화 하기 위해 JPA를 사용한다.

2.3. Spring Boot

Spring Boot 는 스프링 포트폴리오를 신속하게, 미리 정의된 방식으로, 이식성 있게, 실제 서비스 환경에 사용할 수 있도록 조립해 놓은 것이다.

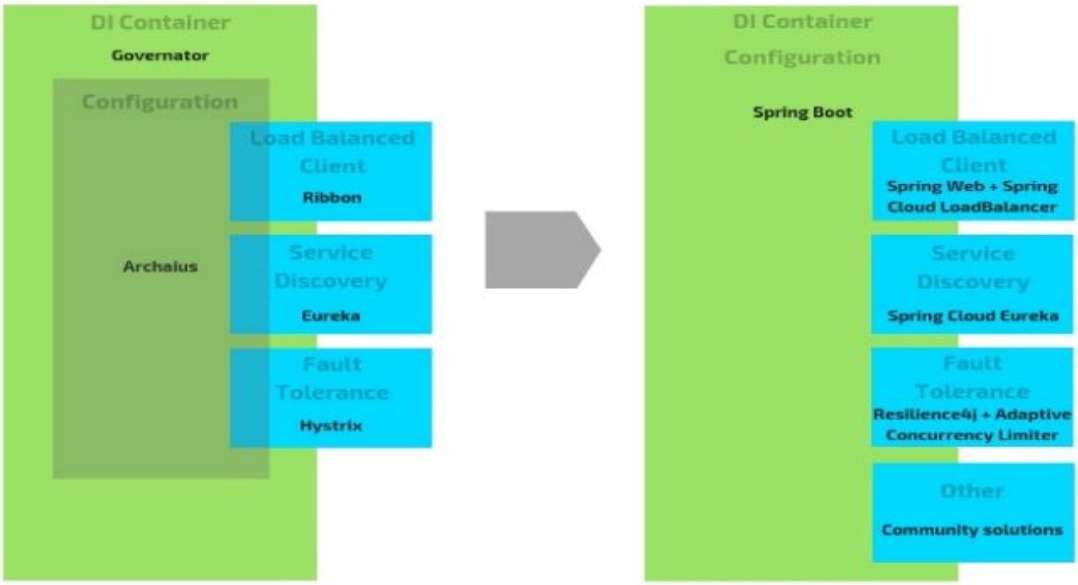
신속성: 의존관계를 포함하여 애플리케이션의 여러 요소에 기반한 의사결정을 신속하게 적용할 수 있게 하여 개발속도를 높인다.

미리 정의된 방식: 스프링 부트의 구성을 정하면, 그 구성에 따라 미리 정의된 방식으로 기본적인 설정을 자동으로 지정해 준다.

이식성: 자바의 표준도구인 스프링 프레임워크를 기반으로 만들어져, JDK 가 있다면 어느 환경에서나 실행할 수 있다. 또한, 특정한 인증을 받은 애플리케이션 서버가 필요하지 않으므로 스프링 부트로 애플리케이션을 만들고 패키지를 만들면 어디에서든 배포하여 실행이 가능하다.

실제 서비스 환경에서 사용 가능: 작은 부분에서 사용해야 하는 제약이 없으며, 실제로도 광범위하게 사용되고 있다. 실제로 넷플릭스의 경우에는 클라우드 인프라의 요구사항인 안정성,

확장성, 효율성, 보안을 충족하기 위해 스프링 부트를 채택하였고, 일부는 넷플릭스 자체의 소프트웨어의 사용 및 조정을 통해 인프라를 발전시켰다.



[그림 4] Netflix 의 Spring Boot 사용 사례

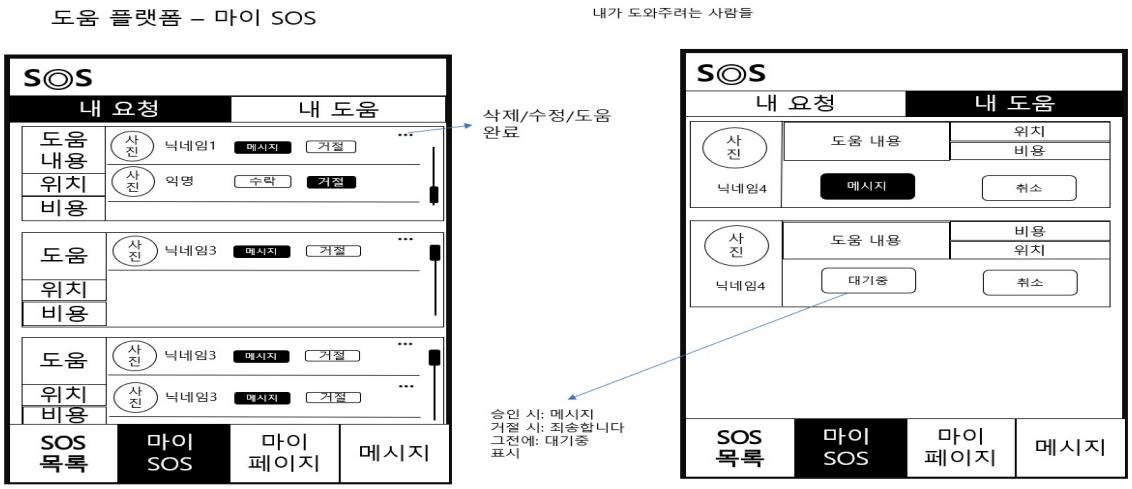
2.4. React

React 는 사용자 인터페이스(UI)를 만드기 위한 JavaScript 라이브러리의 일종이다. 양방향 데이터 바인딩을 사용하면 규모가 커질수록 데이터의 흐름을 추적하기 힘들고 복잡해지는 경향이 있다. 따라서 보다 더 데이터의 변화를 예측가능하도록 데이터의 흐름을 단방향 흐름을 가진다. 또한, content, Text, Button 등을 컴포넌트로 관리하여 코드의 재사용성을 높이고, 이로 인해 유지보수성을 높일 수 있다. 또한, 리액트는 복잡함이 적기 때문에 프로그래머들에게 배우기 쉬우므로 UI 수정을 보다 더 용이하게 할 수 있다. 그리고 React-Native 의 경우에는 모바일 환경에서도 개발이 가능하도록 한 라이브러리로 개발하고자 하는 애플리케이션을 모바일 형태로 배포가 가능하다.

3. 프로젝트 내용

3.1. UI 설계

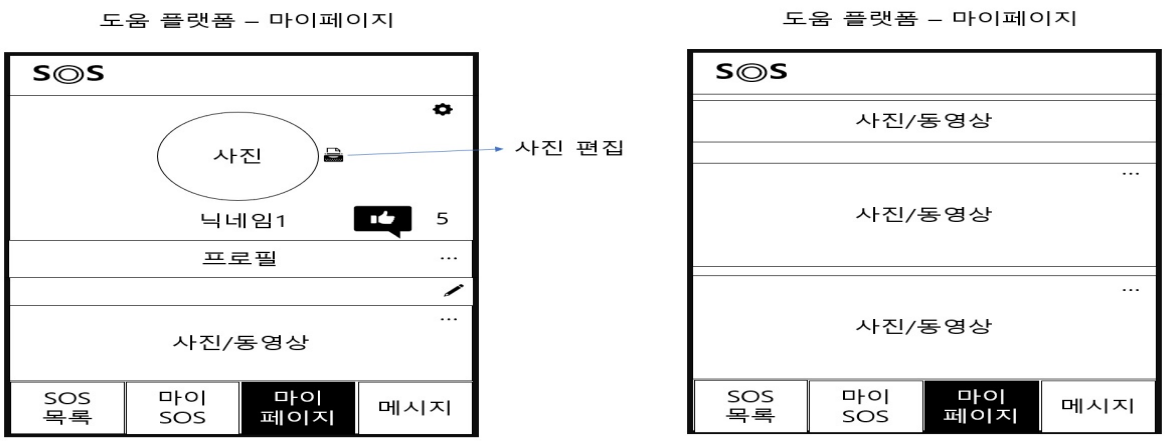
3.1.1. 마이 SOS



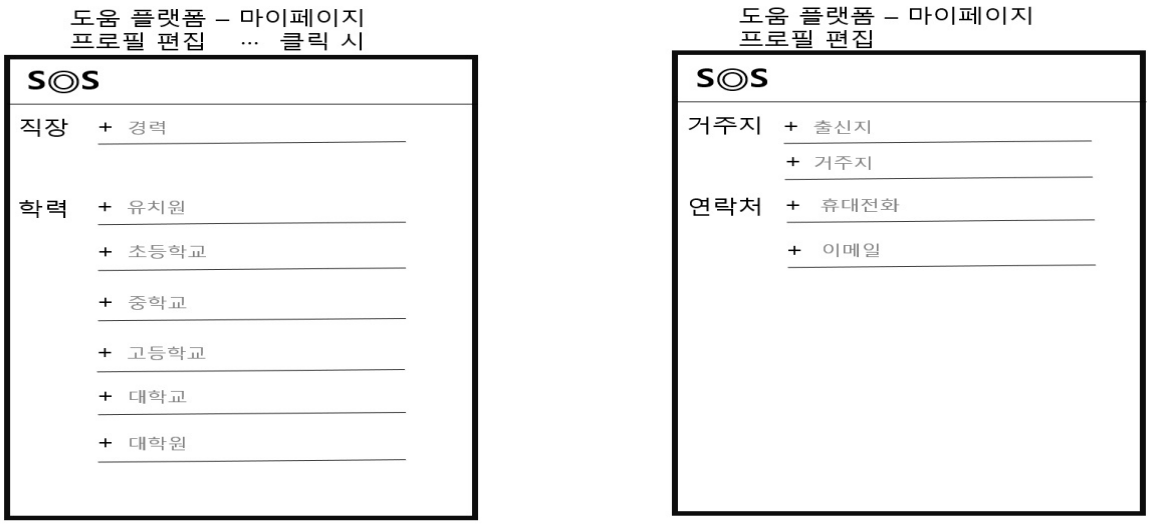
[그림 5] 마이 SOS 설계 UI

- 마이 SOS 는 크게 2 개의 화면으로 구성된다.
  - 사용자가 작성한 SOS 목록
  - 사용자가 도와주려는 사람들의 SOS 목록
- 사용자가 작성한 SOS 목록에는 다음을 포함한다.
  - 도움 내용, 위치 및 비용
  - 도움의 수락 여부 버튼 (수락 또는 거절)
- 사용자가 도와주려는 사람들의 SOS 목록은 다음을 포함한다.
  - 도움 내용, 위치 및 비용
  - 수락 여부 (승인 시: 메시지, 거절 시: 죄송합니다, 처리 X: 대기중)

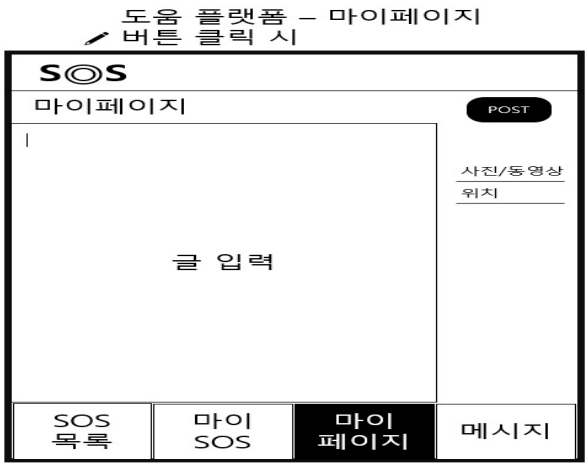
3.1.2. 마이페이지



[그림 6] 마이페이지 Main 설계 UI



[그림 7] 프로필 편집 UI



[그림 8] 게시글 작성 설계 UI

● 마이 페이지는 크게 3 개의 화면으로 구성된다.

- 마이 페이지 Main
- 프로필 편집
- 게시글 작성

● 마이페이지 Main 에는 다음을 포함한다.

- 사용자의 기본 정보 (프로필 사진, 닉네임)
- 프로필 편집 및 게시글 작성 이동 버튼
- 사용자가 작성한 게시글 정보

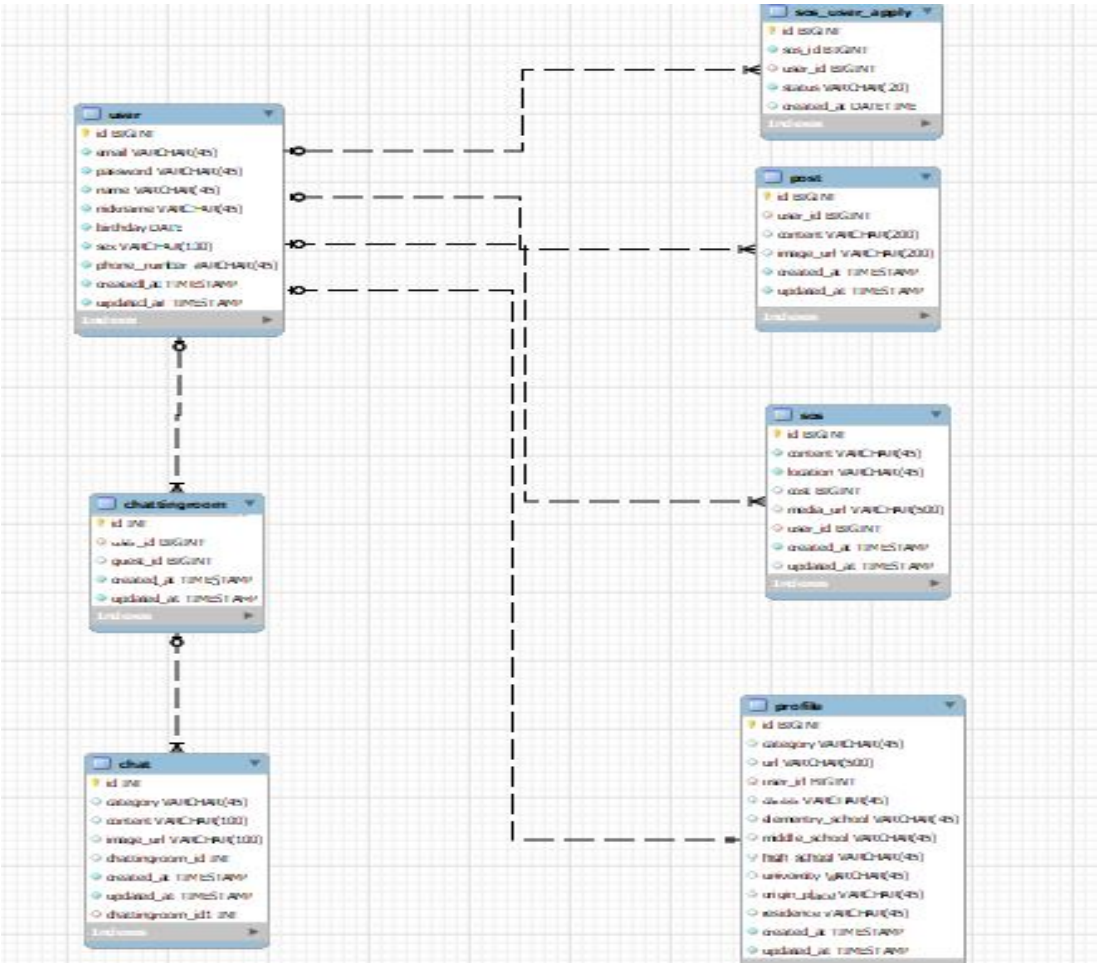
● 프로필 편집 UI 에는 다음을 포함한다.

- 사용자의 경력, 학력, 주소, 연락처 정보 입력 텍스트

● 게시글 작성 UI 에는 다음을 포함한다.

- 게시글 내용을 작성하기 위한 텍스트
- 이미지 첨부와 게시글 등록 버튼

3.2. DB 설계



[그림 9] ER Diagram

[Table 관계도]

- **user 와 Profile (1:1)**

  user 는 하나의 **Profile** 연결을 갖는다.

- **user 와 sos (1:N)**

  user 는 여러개의 **sos** 연결을 갖는다.

- **user 와 sos_user_apply (1:N)**

  user 는 여러개의 **sos_user_apply** 연결을 갖는다.

- **user 와 post (1:N)**

  user 는 여러개의 **post** 연결을 갖는다.

- **user 와 chattingroom (1:N)**

  Stroke activity 는 여러개의 **chattingroom** 연결을 갖는다.

- **chattingroom 과 chat (1:N)**

  chattingroom 은 여러개의 **chat** 연결을 갖는다.

[Table 정보]

- user 테이블 정보

user			
열명	**타입**	**크기**	**옵션**
id	bigint	8	기본키
email	varchar	45	
password	varchar	45	
name	varchar	45	
nickname	varchar	45	
birthday	date		
sex	varchar	100	
phone_number	varchar	45	
created_at	timestamp		
updated_at	timestamp		

- sos 테이블 정보

sos			
열명	**타입**	**크기**	**옵션**
id	bigint	8	기본키
content	varchar	45	
location	varchar	45	
cost	bigint	8	
media_url	varchar	500	
user_id	bigint	8	
created_at	timestamp		
updated_at	timestamp		

- sos_user_apply 테이블 정보

sos_user_apply			
열명	**타입**	**크기**	**옵션**
id	bigint	8	기본키
user_id	bigint	8	
sos_id	bigint	8	
status	bigint	20	
created_at	timestamp		
updated_at	timestamp		

- **profile 테이블 정보**

profile			
열명	**타입**	**크기**	**옵션**
id	bigint	8	기본키
category	varchar	45	
url	varchar	500	
user_id	bigint	8	
career	varchar	45	
elementry_school	varchar	45	
middle_school	varchar	45	
high_school	varchar	45	
university	varchar	45	
origin_place	varchar	45	
residence	varchar	45	
created_at	timestamp		
updated_at	timestamp		

- **post 테이블 정보**

post			
열명	**타입**	**크기**	**옵션**
id	bigint	8	기본키
content	varchar	200	
image_url	varchar	200	
user_id	bigint	8	
created_at	timestamp		
updated_at	timestamp		

- **chattingroom 테이블 정보**

chattingroom			
열명	**타입**	**크기**	**옵션**
id	bigint	8	기본키
user_id	bigint	8	
guest_id	bigint	8	
created_at	timestamp		
updated_at	timestamp		

- chat 테이블 정보

chat			
열명	**타입**	**크기**	**옵션**
id	bigint	8	기본키
category	varchar	45	
content	varchar	100	
image_url	varchar	100	
chattingroom_id	bigint	8	
created_at	timestamp		
updated_at	timestamp		

3.3. API 설계

3.3.1. 마이 SOS

3.3.1.1. 내 요청

	Method	Request	Resopnse
사용자가 도와주려고하는 SOS 목록	GET	{userId: int}	{sosList: list}
해당 SOS 를 요청한 사용자 목록	GET	{sosId: int}	{userList: list}
요청여부 수정	PUT	{sosId: int, userId: int}	{code: int}

3.3.1.2. 내 도움

	Method	Request	Resopnse
사용자가 도와주기를 신청한 SOS 목록	GET	{userId: int}	{ sosList: list}
도와주기 취소	PUT	{sosId: int, userId: int}	{code: int}

3.3.2 마이 페이지

3.3.2.1. 마이페이지 Main

	Method	Request	Resopnse
프로필 기본 정보	GET	{userId: int}	{profile: user}
등록한 게시글 목록	GET	{userId: int}	{postList: list}

3.3.2.2. 프로필 편집

	Method	Request	Resopnse
프로필 편집 수정 완료	PUT	{userId: int, career: string, elemetrySchool: string, middleSchool: string, highSchool: string, university: stirng, originPlace: string, residence: string}	{code: int}

3.3.3.3. 게시글 작성

	Method	Request	Resopnse
게시글 등록	POST	{userId: int, content: string, imageUrl: string}	{code: int}

3.4. 프로젝트 결과

3.4.1. 마이 SOS

3.4.1.1. 내 요청

[그림 ms1] 내 요청 UI

[그림 ms1]는 마이 SOS 중 내 요청 UI 이다. 하단의 마이 SOS 를 클릭 시 또는 [그림 ms2]에서 내 요청을 클릭 시에 이동할 수 있는 페이지이다. 내 요청 UI 는 사용자가 등록한 SOS 들 중 다른 사용자가 도와주기로 신청한 SOS 들의 목록이 화면에 보여진다. 각 사용자별로 신청한 내역들을 수락, 거절 버튼을 이용하여 수락여부를 결정할 수 있다. 이 때, 수락, 거절 여부가 결정되면 변경을 할 수 없도록 메시지를 표시하도록 한다.

3.4.1.2. 내 도움

[그림 ms2] 내 도움 UI

[그림 ms2]는 내 도움 UI 이다. 사용자가 도와주기로 한 SOS 들을 확인할 수 있다. 상단의 내 도움 버튼을 통해 이동할 수 있다. SOS 정보로는 도움을 요청한 사용자의 닉네임과 프로필사진, 도움내용과 장소, 비용이 보여진다. 하단에는 승인.거부 여부와 취소버튼이 있다. 승인 시에는 승인, 거절 시에는 죄송합니다, 수락여부가 결정되지 않았으면 대기중이 보여진다. 취소버튼은 내가 도와주겠다고 하는 요청을 취소할 수 있다.

3.4.2. 마이페이지

3.4.2.1. 프로필 화면

[그림 mp1] 프로필 상단

[그림 mp2] 프로필 하단

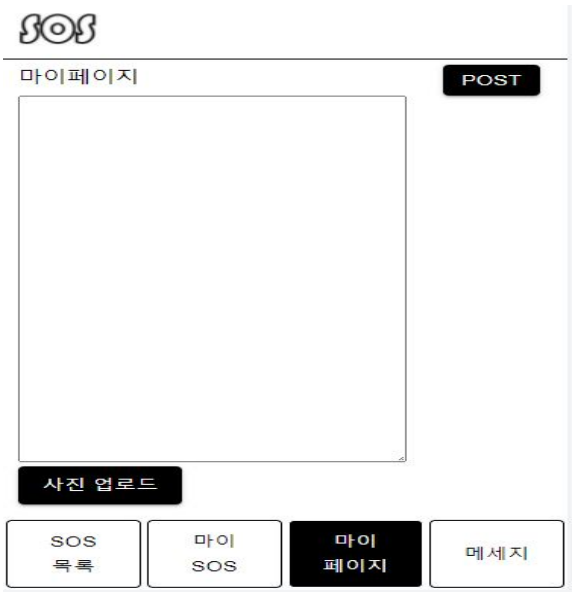
[그림 mp1]과 **[그림 mp2]**는 마이페이지의 메인페이지 UI 이다. **[그림 mp2]**의 맨 하단의 마이페이지 버튼을 클릭할 시 이동할 수 있다. 화면 상단에는 **[그림 mp1]**과 같이 사용자의 프로필 사진과 닉네임이 보여진다. **[그림 mp1]** 우측 하단의 화살표 버튼은 프로필 편집 이동하면으로 버튼을 클릭 시 **[그림 mp3]**로 이동한다. 화면 하단에는 **[그림 mp2]**와 같이 사용자가 등록한 게시물을 확인할 수 있으며 우측 상단의 연필모양의 버튼으로 게시글을 등록할 수 있다. 게시물의 정보로는 첨부한 이미지와 게시글의 내용이 있으며, 이미지가 없을 경우 별도로 지정한 no-image 가 보여진다. 연필모양의 버튼 클릭 시에는 **[그림 mp4]**로 이동한다.

3.4.2.2. 프로필 편집

[그림 mp3] 프로필 편집 화면

[그림 mp3]는 사용자의 프로필 세부 정보를 편집하는 화면이다. **[그림 mp1]**의 화살표 모양의 버튼을 클릭 할 때 나타나는 화면이다. 입력 받는 정보로는 경력(직장)과 학력(초등학교, 중학교, 고등학교 대학교) 주거지 정보(출신지와 거주지)를 입력받는다. 사용자의 선택에 따라 입력을 하지 않을 수 있다. 수정 버튼을 클릭 시에는 수정완료 메시지와 함께 마이페이지 메인 페이지로 이동한다.

3.4.2.3. 게시글 작성



[그림 mp4] 게시글 작성 UI

[그림 mp4]는 게시글을 등록하는 화면이다. [그림 mp2]의 연필모양의 버튼을 클릭 할 때 나타나는 화면이다. 게시글에는 내용 작성과 이미지 첨부를 할 수 있다. 비어있는 텍스트에 내용을 입력할 수 있으며, 반드시 입력해야 하는 필수항목이다. 하단의 사진 업로드 버튼은 이미지를 첨부할 수 있으며, image 형태로만 첨부가 가능하다. 사진업로드는 내용과 달리 사용자의 선택에 따라 등록하지 않을수 있다. 우측 상단의 POST 버튼은 게시글을 등록하는 버튼으로 content 가 입력되지 않으면 글을 입력해야 된다는 메시지가 나온다. 게시글이 등록되면 등록완료의 메시지와 함께 DB 의 POST 테이블에 저장하고, 마이페이지 메인페이지로 이동한다.

4. 결론

4.1. 기대효과

SOS 는 서로 돕고 다양한 가치관이 공존하는 사회를 추구하고, 온-오프라인의 복잡한 사회적 관계망 서비스를 지향하고 있다. 또한, 도움 플랫폼을 통해 팀원 모집, 중고 거래 등의 다양한 서비스들을 중개함으로써 행복한 사회를 꿈꿀 수 있을 것으로 기대된다. 그리고 차후에는 온라인 의류 쇼핑몰이나 맛집 등과 협업을 통해 서비스를 확장하여 다양한 반응형.노출형 광고를 통한 수익을 얻을 수 있을 것으로 예상된다.

4.2. 추후 연구 방향

추후에는 WebSocket 을 활용하여 구현되지 않은 채팅 시스템의 설계하고 이를 구체화 시키는 데에 연구를 진행할것이다. 또한, 프로필 편집 시에는 현재 텍스트 입력으로만 구현이 되어있다. 하지만 경력, 학력, 거주지에 대한 정보들을 수집하여 DB 로 구축하고 이를 검색했을 때 다른 포털사이트의 검색 시 나타나는 연관검색어 처럼 나올 수 있도록 구현하는 것을 연구해볼 것이다. 그리고 현재 웹 형태로 배포되어있는 서비스를 모바일 형태로도 개발하여 많은 사용자들이 이용할 수 있도록 할 것이다.

5. 참고문헌

[1] 스프링 부트와 AWS로 혼자 구현하는 웹 서비스

[2] 스프링 부트 실전 활용 마스터

[3] [Android] 코틀린과 자바의 차이 https://bbaktaeho-95.tistory.com/50

[4] Kotlin 2019 - 2019년 개발자 에코시스템의 현황 인포그래픽
https://www.jetbrains.com/ko-kr/lp/devecosystem-2019/kotlin/

[5] Kotlin 공식 문서 https://kotlinlang.org/docs

[6] Netflix의 스프링 부트 사용 사례
https://netflixtechblog.com/netflix-oss-and-spring-boot-coming-full-circle-4855947713a0

[7] [React] React.js란?
https://velog.io/@jini_eun/React-React.js%EB%9E%80-%EA%B0%84%EB%8B%A8-%EC%A0%95%EB%A6%AC