**모바일 앱 프로그래밍2 중간 보고서**

프로젝트명: 자동 번역 다국적 유학생 커뮤니티

참여 인원:

2016117007: 정원준

2018114686: 박준하

2020115974: 한의진

2021113490: 이현서

**시스템 아키텍처와 디자인 패턴**

Diagram

Description automatically generated

우리의 시스템 아키텍처는 기본적으로 POP(Protocol Oriented Programming)기반의 기본적인 MVVM 디자인 패턴을 채택했습니다. 간단히 로그인 기능을 구현할 때의 상황을 적용해보도록 하겠습니다. 첫번째로 SignInView에 로그인 창을 구현합니다. 그리고 Model로는 RDS와 Firebase의 FireAuth를 사용해 Oauth를 구현합니다. 그리고 SignInViewModel을 작성해 두 연결관계를 추상화해서 원천자료의 변경을 감지해 View를 업데이트 하는 작업을 진행합니다.

Diagram

Description automatically generated

우리의 간단한 시스템 아키텍쳐 그림입니다. AWS의 서비스를 적극 사용했으며, VPC내에 우리의 서비스를 구축하였습니다. 기본적으로 Docker, Elastic Container Registry를 통해 CI/CD Pipeline을 구축하여 Application Server의 형상을 관리하였습니다. 또한 이들은 Auto Scaling Group과 Elastic Load Balencer를 활용하여 서버의 부하를 견딜 수 있게 했습니다. 그 외에도 외부 SaaS로 Firebase의 Auth, Store를 사용하고, 외부 Relational DB로 RDS, 캐시 서버로 Redis를 구축했습니다.

**데이터베이스 설계**

Diagram

Description automatically generated

먼저 로그인한 유저의 정보가 users 라는 콜렉션에 개별 document로 저장됩니다. id는 authentication에서 발급받은 uid가 그대로 들어가 swiftUI의 리스트로 표시하기 위한 Identifiable 프로토콜의 구현으로도 사용됩니다.필드는 추후에 필요에 따라 추가할 예정이며 우선은 id와 email만으로 설정하였습니다.

다음으로는 게시글 관련 콜렉션인 posts 입니다. post의 경우 개별 post의 document id, 작성자의 id, 글의 내용 content, 제목 title, 댓글목록인 comments로 구성됩니다.

comments의 경우 서브콜렉션이 필드로 사용되며 개별 comment document에는 document id, author id, content로 구성됩니다.

마찬가지로 좋아요를 눌렀을때 추가되는 데이터인 like또한 서브콜렉션으로써, 자체 document id와 좋아요를 누른 사람의 id가 저장됩니다.

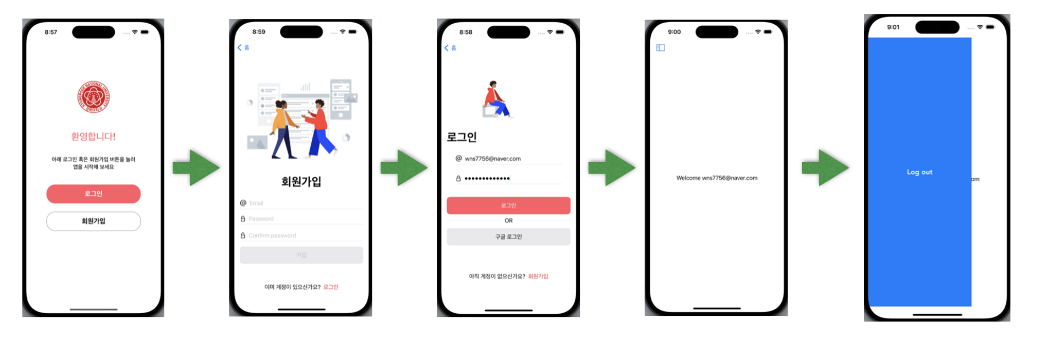
swift 프로젝트에서 firebase Firestore와 통신하기 위해서 FirebaseCore와 FirebaseFirestore라는 패키지를 import하여 해당 패키지에 포함된 API를 사용하였는데, 다른 패키지를 포함하여 모든 의존성은 cocoapods 시스템을 통해 일괄 관리 하였습니다.

저희가 작성한 FirestoreAPI는 Codable이라는 프로토콜을 준수하는 struct를 대상으로 작업을 수행합니다.

Codable은 FirebaseFirestoreSwift라는 패키지에 포함되어 있는 프로토콜로써, 이 프로토콜을 구현하게 되면 firestore와 통신할때 구조체에 들어있는 필드가 자동으로 매핑되어 JSON으로 변환됩니다. 모든 firestore 상에 존재하는 모든 collection은 swift 코드상에서 단순 구조체로 표현하여 사용할 수 있습니다.

마지막으로 swiftUI 상에서 UI가 직접 firebase와 통신할때는 task라는 modifier를 사용하여 비동기로 처리하여 사용자 반응성을 높일 수 있었습니다.

**로그인 창**



로그인 화면입니다.

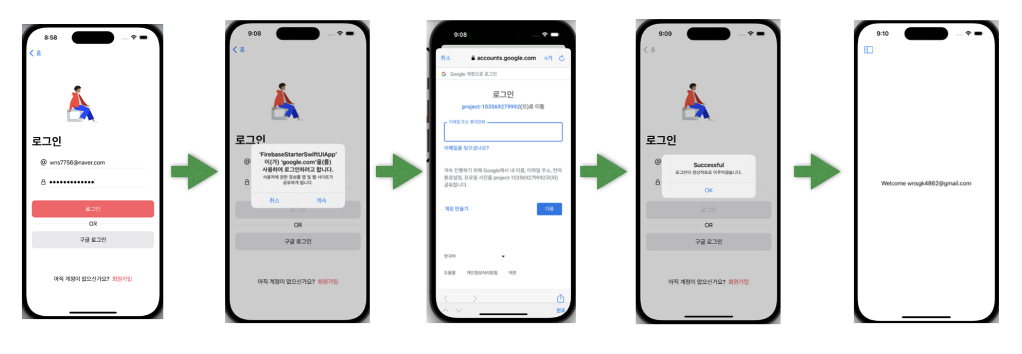
우선 첫화면에 들어가면 경북대학교 로고, 안내문구와 함께 로그인과 회원가입 버튼이 나타납니다.

계정이 없다면 우선 회원가입부터 해주어야 합니다.

회원가입 창으로 들어가면 위 그림과 같이 이메일과 패스워드, 패스워드 확인 필드가 나타며 모든 필드를 올바르게 입력하면 가입이 완료됩니다. 만약 이메일 필드를 형식에 맞지않게 입력하거나 확인 패스워드가 틀릴경우 에러문구와 함께 가입이 진행되지 않습니다.

회원가입을 완료했으면 로그인 화면으로 돌아가서 로그인을 진행할 수 있습니다.

방금 입력한 이메일과 패스워드를 넣고 로그인 버튼을 누르면 계정 확인 화면으로 들어갑니다. 이때 해당 창에서 왼쪽 사이드바를 활성화하여 로그아웃 할수 있습니다.



만약 구글 계정으로 로그인하기를 원할경우 구글 로그인 버튼을 클릭하여 제3사 로그인 서비스를 이용할 수 있습니다. 해당 버튼을 누르면 위 그림처럼 구글 로그인 창이 뜨며 이메일과 패스워드를 올바르게 입력할 시 휴대폰으로 확인 알람이 가고, 해당 알람에서 허용 버튼을 눌러 로그인을 완료할 수 있습니다.

**게시글 창(게시판)**

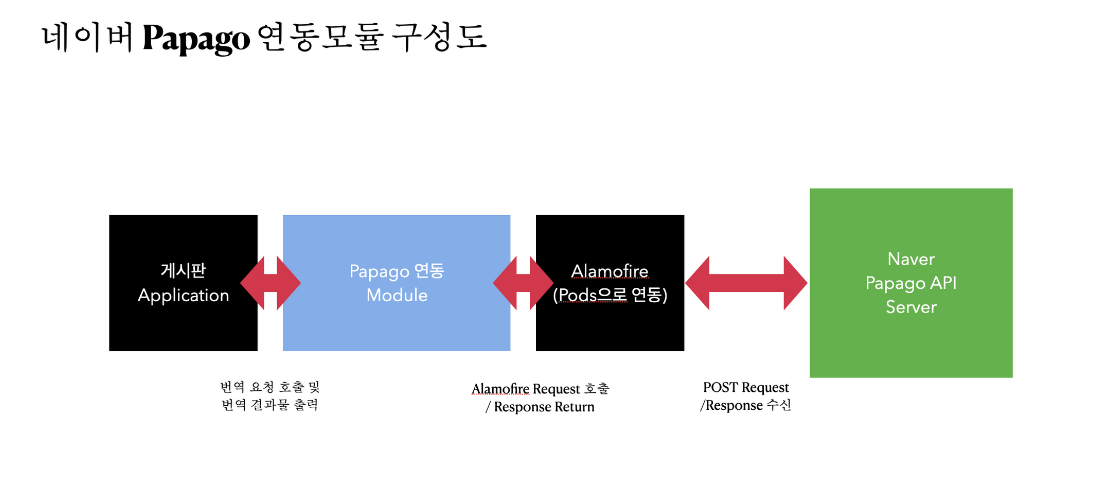
먼저 기본적인 커뮤니티 어플에 맞게 유저들의 게시글을 모아볼 수 있는 게시판UI를 구현하였습니다. 전체적인 UI은 대표적인 대학생 커뮤니티앱인 에브리타임을 벤치마킹하여 구성하였습니다. 여기서 전체 글 목록과 각 게시글의 세부 내용을 보고 댓글을 달 수 있는 세부 페이지를 자유롭게 이동할 수 있게 하기 위해 SwiftUI의 NavigationView를 이용하였고, 상단의 NavigationBar에 세부설정과 검색, 메인 화면으로 이동하기 위한 버튼을 배치하였습니다. 여기서 리스트에 NavigationVie를 적용하게 되면 나타나는 오른쪽의 화살표를 없애기 위해 각 리스트를 ZStack으로 감싸 EmptyView를 적용하였습니다. 각 게시글을 클릭하게 되면 세부 게시글 페이지로 이동하게 되고 세부 게시글 페이지에서는 글의 전체 내용을 확인할 수 있고, 하단의 TextField를 통하여 댓글을 입력할 수 있게 됩니다.



**파파고 번역 기능**

Naver Papago API를 활용하여 한국어→영어 및 영어→한국어 번역 모듈을 개발하였습니다.

* 번역 모듈의 원문 언어 및 번역 언어는 Request의 Header 및 Body 부분에 의해 결정되므로, 영문 및 국문 이외에도 다양한 언어의 번역 기능을 애플리케이션에 제공할 수 있습니다.
* 사용된 오픈 소스 라이브러리는 Alamofire로, HTTP Request를 보내고 Response를 수신하는 코드에 사용됩니다. MIT License가 적용되어 있어 애플리케이션 저작권 문제는 없습니다.
* 메인 애플리케이션(게시판 모듈)에서 연동 모듈을 Request Parameter와 함께 호출하면 연동 모듈에서 Alamofire 라이브러리를 통해 POST Request를 전송합니다. (원문언어, 번역 언어, 원문 전송)
* 번역문을 Alamofire에서 Response로 받으면(Closure 형태로 Return) 이를 메인 애플리케이션의 게시판 모듈에 표시할 수 있도록 연동합니다.



Diagram

Description automatically generated

**중간보고서 발표 영상 유튜브 링크**

https://www.youtube.com/watch?v=scDQJwDEak8