**컴퓨터공학종합설계**

**최종보고서**

****

**컴퓨터 공학 종합설계 1조**

**12172133 이종혁**

**12172133 임유한**

**12181667 이지율**

**목차**

1. **프로젝트 기획 동기 & 의도**
2. **프로젝트 기본 요구 사항**
3. **프로젝트에 포함된 기능 명세**
4. **주요 키워드 소개**
5. **공통 기능 - 회원가입/로그인/마이페이지**
6. **개요 및 요구사항**
7. **프론트엔드 UI & 로직**

* **기능 고찰**

1. **백엔드 API 구현**

* **기능 고찰**

**C. 일자리**

1. **개요 및 요구사항**
2. **프론트엔드 UI & 로직**

* **기능 고찰**

1. **백엔드 API 구현**

* **기능 고찰**

1. **데이터 크롤링 파트**

**D. 의료/복지 관련**

1. **개요 및 요구사항**
2. **프론트엔드 UI & 로직**

* **기능 고찰**

1. **백엔드 API 구현**

**- 기능 고찰**

**E. 경제적 지원 관련**

1. **개요 및 요구사항**
2. **프론트엔드 UI & 로직**

* **기능 고찰**

1. **백엔드 API 구현**

* **기능 고찰**

1. **데이터 크롤링 파트**

**F. 관계 단절 방지**

1. **개요 및 요구사항**
2. **프론트엔드 UI & 로직**

* **기능 고찰**

1. **백엔드 구현**

* **기능 고찰**

1. **프로젝트 기술 요약, 업무 분담, 개발 기록**
2. **참고자료**

**1. 프로젝트 기획 동기 및 의도**

* **2020 인구주택총조사에 따르면 1인 가구는 664만 가구로 전체 일반 가구의 31.7%를 차지하고 있다. 그 중 노인 1인 가구의 비율은 현재 30%를 웃돌며, 20년 안에 45%까지 치솟을 것으로 보인다.**
* **또한, 대한민국은 유례없는 세계 최저의 출산율을 보이며 초고령화 사회에 진입할 것이 확실시 되었다. 이는 노인층의 케어 서비스의 중요성이 더욱 더 커질 것임을 의미한다.**
* **앞서 말했듯, 노년층의 1인 가구 증가가 큰 사회적 이슈가 되고 있다. 따라서 디지털 이용 및 접근성이 취약한 노년층에게 꼭 필요한 통합 맞춤형 서비스를 제공하여, 혼자 남겨진 노인이 자신의 힘으로 자립할 수 있는 플랫폼을 설계하고, 나아가 고립에서 벗어나게 하는 것을 목표로 한다.**
* **공통된 사유로, 노인의 자립을 위한 결핍 요소 중 정보를 뽑아 노인 대상의 복지 정책 등을 노인의 정보에 맞춰 제공하는 기능을 포함하도록 한다.**

**2. 프로젝트 공통 요구사항**

* **UI의 간소화**
  + **디지털 기기 사용에 취약한 노년층을 타겟팅하고 있는 서비스이기 때문에, 사용자단에서는 사용하기 쉽도록 UI를 최대한 직관적이고 단순하게 만들고자 한다.**
* **데이터의 시각화/필터링**
  + **최대한 직관적으로 알아보기 쉬워야 하기 때문에 지도 API, 도표 등을 이용해서 데이터를 시각화 하도록 한다.**
  + **직접 사용자가 설정하고 검색하는 복잡한 기능 보다는 최대한 백엔드/데이터 단에서 조금 추가적인 작업을 더 하더라도 최대한 유저 맞춤형으로 데이터를 필터링해 필요한 정보만을 제공해야 한다.**

**3. 프로젝트에 포함될 기능 명세**

1. **주요 키워드 소개**

**노년층이 가장 필요로 하는 서비스가 무엇일지를 먼저 토의한 후에, 크게 네 가지 키워드로 분류해서 정의하였다.**

* **일자리**
* **의료/건강**
* **경제적 지원**
* **관계 단절 방지**

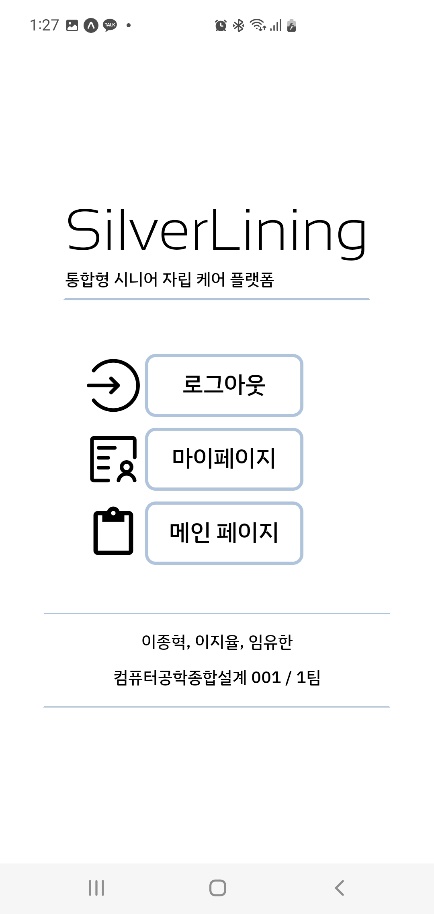
1. **유저 관련 - 회원가입/로그인/마이페이지**
2. **개요 및 요구사항**

* **노년층을 타겟팅한 어플리케이션 서비스이니만큼, 최대한 직관적으로 회원가입 화면을 구성해야 하며, 사용자 맞춤형으로 데이터를 필터링해 전송하기 위해 최대한 회원가입 시에 유저와 관련된 데이터를 세분화해서 입력하도록 할 예정이다.**

1. **프론트엔드 UI & 로직**

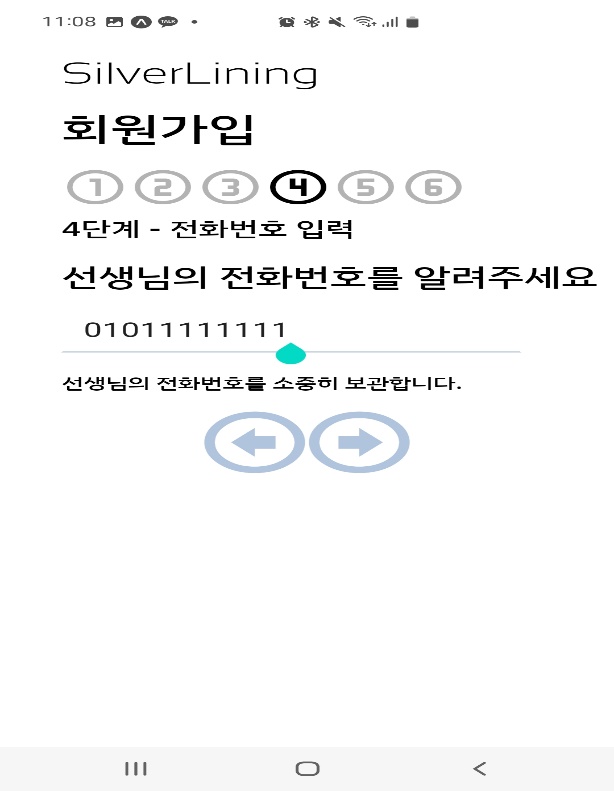
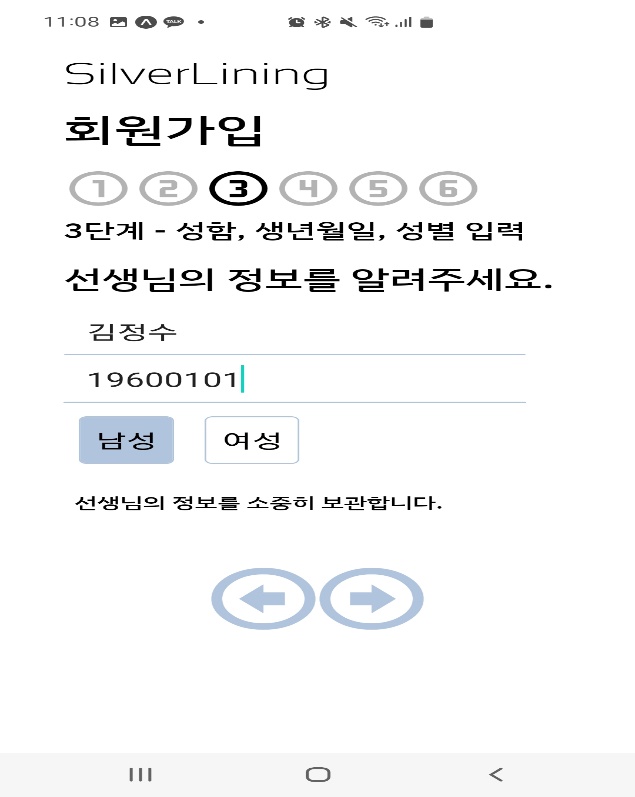
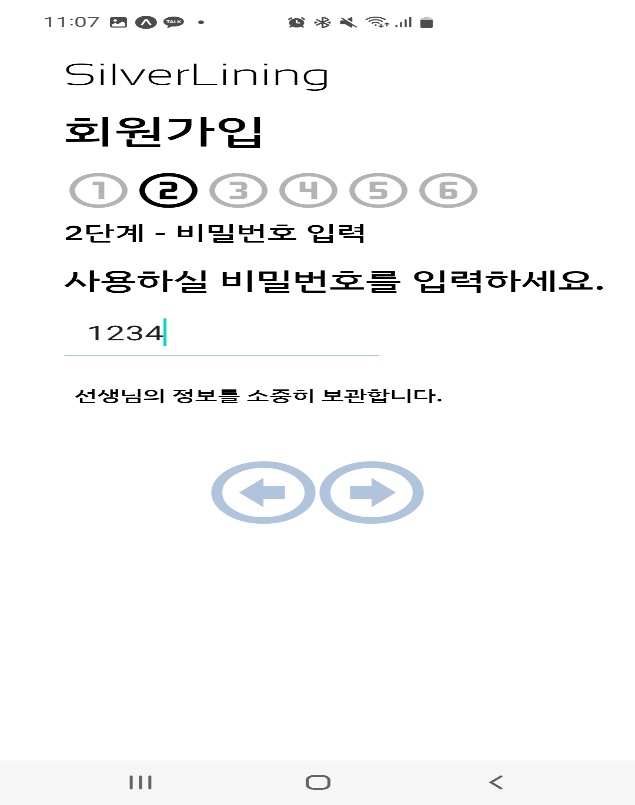
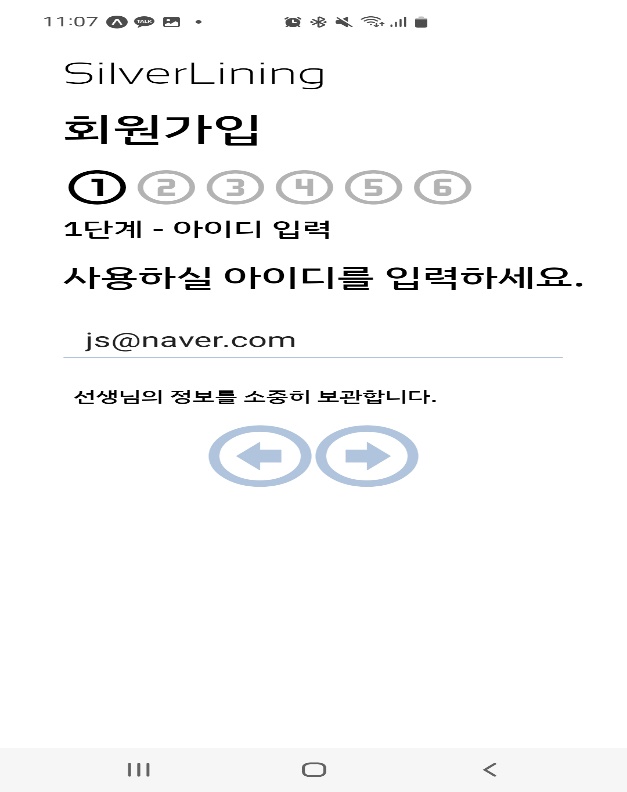
* **해당 기능은 공통 부분, 일반 사용자 부분, 기업 회원 부분으로 구분된다. 기능의 설명은 공통 부분, 일반 부분, 기업 회원 부분 순으로 하기로 한다.**
* **프론트 엔드 - 공통 부분**

1. **타이틀 UI**

****

* **위 페이지는 타이틀 페이지로, 로그인, 회원가입, 메인 페이지로 이동하는 navigation 역할의 버튼이다.**
* **해당 부분의 경우 로그인 시 서버에서 response로 받은 json web token을 1차적으로 AsyncStorage에 저장한다. 이후 AsyncStorage에서 불러온 jwt를 쿠키로 하여, 조건부 렌더링을 통해 로그인 시와 비 로그인 시의 렌더링을 달리 하였다.**

1. **회원가입 UI**



* **상단 페이지는 위와 같이 아이디, 비밀번호, 전화번호, 유저의 개인 정보 등을 입력 받는다. 이중 전화번호 데이터를 이용하여 인증 절차를 진행할 것이다. 하단 이미지에서 인증 ui를 확인할 수 있다.**

**텍스트이(가) 표시된 사진

자동 생성된 설명**

* **해당 기능은 open api 사용 및 유효성 검증을 진행한다. 1차적으로는 입력한 전화번호에 대하여 데이터베이스에 해당 데이터가 있는지 조회하며, 2차적으로는 해당 전화번호에 대하여 api에 대하여 request를 보내도록 한다.**

**텍스트이(가) 표시된 사진

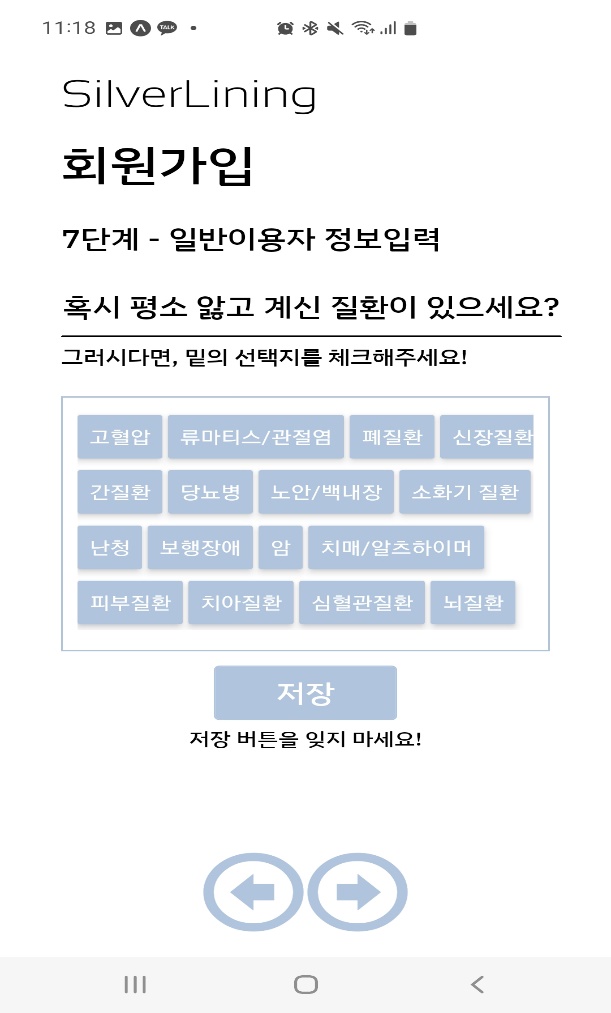
자동 생성된 설명**

* **상단 이미지에서와 같이 사용자의 주소를 입력 받는 부분은 카카오 api를 이용한다. 해당 api의 response data를 추출한 후, 유저 인터페이스를 통해 데이터를 프로젝트의 데이터베이스에 입력한다.**

**텍스트이(가) 표시된 사진

자동 생성된 설명**

* **회원 가입의 공통 부분 마지막 단계에서는 사용자가 일반 사용자인지 기업 사용자인지를 결정할 수 있으며, 어떤 선택지를 고르는 지에 따라 분기를 나눠 회원의 종류에 따라 다른 페이지로 이동하게 된다.**

****

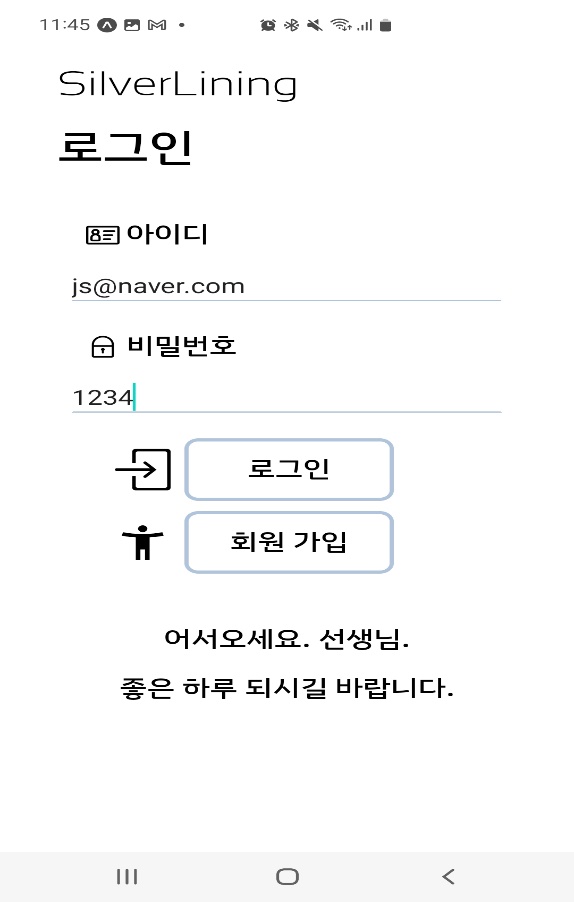
**텍스트이(가) 표시된 사진

자동 생성된 설명**

* **텍스트이(가) 표시된 사진

  자동 생성된 설명유저가 기업사용자인지, 일반 사용자에 따라 입력을 받는 부분이 다르다. 기업 사용자는 추가적으로 데이터를 많이 입력 받지 않으나, 일반 사용자의 경우 위의 이미지와 같이 질병, 보험, 운전 면허 등에 대하여 입력 받는다.**
* **가입이 완료된 경우, 고유한 json web token과 user\_idx를 입력 받는다. 해당 데이터는 백엔드 api를 요청할 때 parameter로 사용된다.**

1. **로그인 UI**

****

* **로그인은 아이디와 비밀번호를 입력 받아 진행되며, 데이터베이스에 존재하는 계정의 정보와 비교하여 일치할 경우 로그인이 수행되게 된다.**

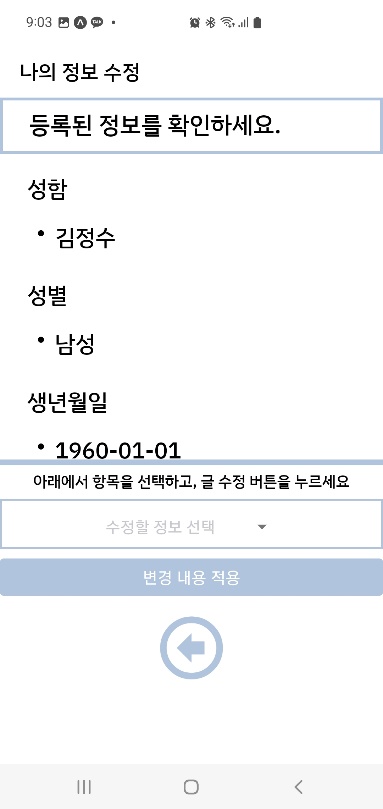
1. **마이페이지**

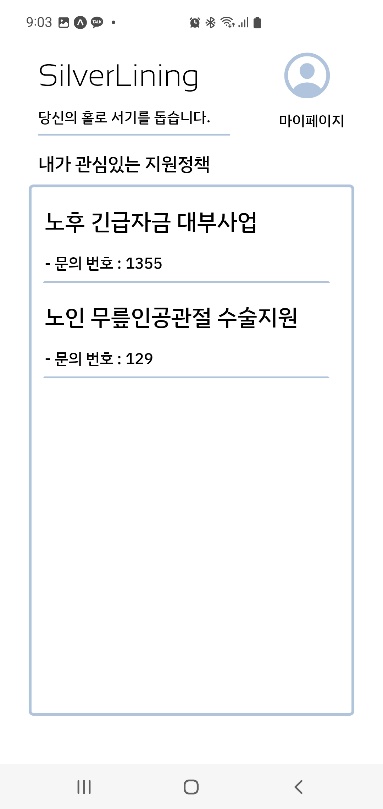
텍스트이(가) 표시된 사진

자동 생성된 설명

* **마이페이지는 일반 사용자와 기업 사용자 간의 기능차이가 있다.**
* **우측은 먼저 일반 사용자의 인터페이스이다.**
* **마이페이지에서는 공통적으로 사용자 자신의 정보를 수정할 수 있도록 하였으며, 또한 자신이 작성한 글을 조회하고 상태를 변화시키는 기능을 포함하였다.**

**일반 사용자의 경우**

텍스트이(가) 표시된 사진

자동 생성된 설명

* **위의 3개의 UI에 접근할 수 있다. 좌상단은 자신의 정보를 수정하는 기능이며, 우측의 두 UI는 자신이 지원했거나 즐겨찾기한 공고, 정책을 조회할 수 있는 기능이다. 해당 로직들은 앞으로 살펴볼 기능들의 로직을 그대로 사용했으므로, 이후 구인구직 플랫폼/정책 추천 기능에서 구체적으로 다룬다.**

**기업 사용자의 경우**

테이블이(가) 표시된 사진

자동 생성된 설명텍스트이(가) 표시된 사진

자동 생성된 설명

* **기업 사용자는 2개의 기능을 이용할 수 있다.**
* **먼저 보이는 하나는 나의 정보를 수정하는 기능으로, 일반 사용자의 그것과 차이가 없다.**
* **내가 등록한 공고 UI의 경우에는 일반 사용자의 내가 지원한 공고와 매칭되는 기능으로, 기업 사용자의 관점에서 지원자 등을 조회할 수 있다.**

**텍스트이(가) 표시된 사진

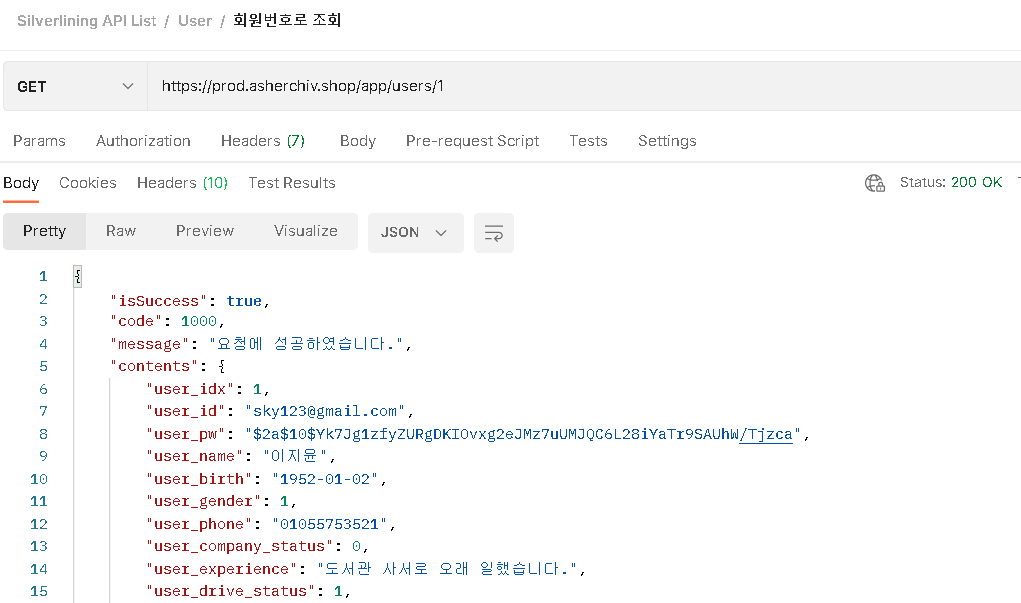
자동 생성된 설명프론트 엔드 기능 고찰**

* **해당 파트는 여러 단계를 거치며 모든 정보를 저장하는 것이 공통의 요구사항이었다.**
* **우측의 질병/질환 선택 ui와 같은 경우, 그리드 형식의 15개 내외의 버튼으로 설계하였다.**
* **해당 버튼은 각각 1번부터 순차적으로 넘버링 되어 state에 담긴 array로 관리하였는데, react-native의 state가 비동기 처리이므로 ui와 상호작용 직후에 이벤트에 의해 다음 페이지로 넘기는 중, 데이터의 저장 처리가 생략되는 현상이 빈번하게 발생했다.**
* **따라서 각각의 데이터를 state에 저장한 후, useEffect hook을 이용해 동기화된 데이터의 업데이트 반영을 확인한 후 다음 페이지로 넘기는 로직이 필요하였다.**

**C) 백엔드 API 구현 내역**

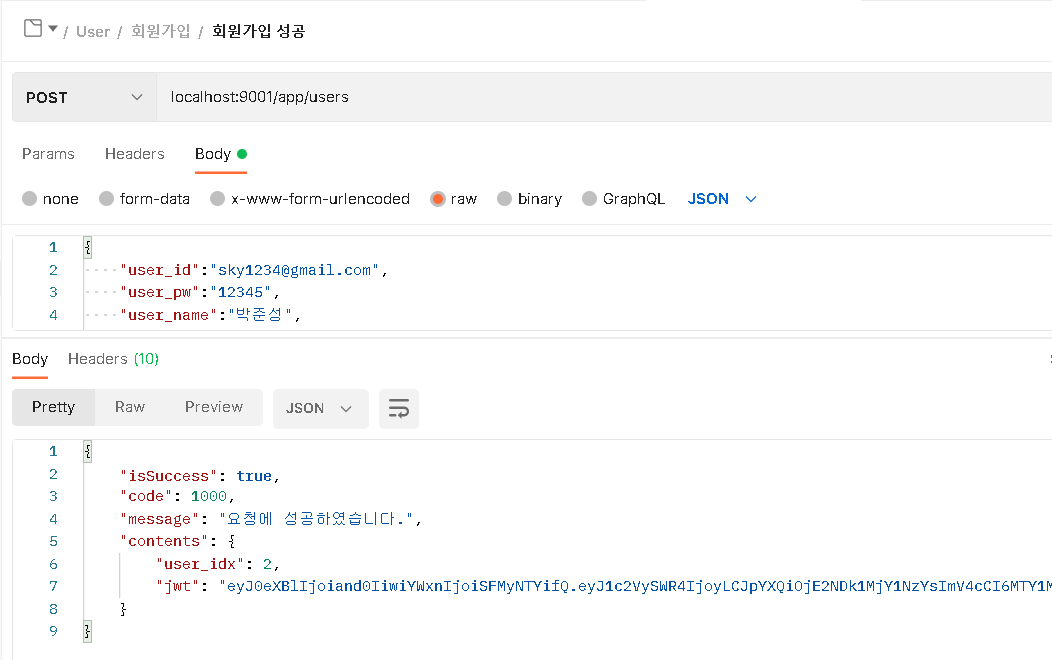
**1) 회원 번호로 조회 API (/app/users/{userNum})**

**: 회원 번호를 request에 포함시키면 DB에 저장된 회원의 모든 정보를 조회하는 기능을 구현함.**

****

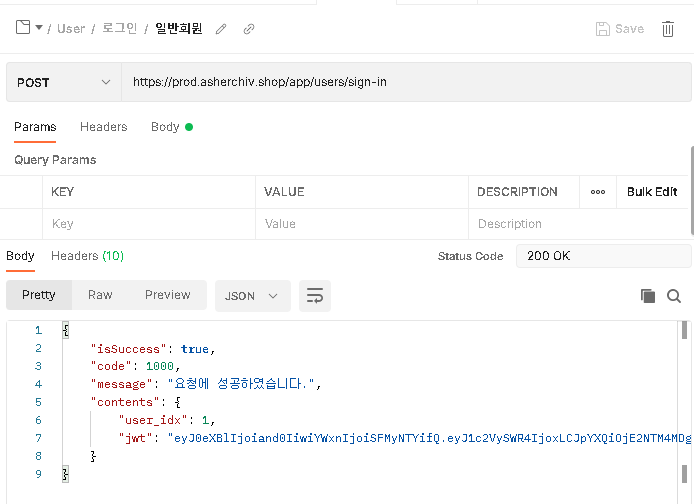
**2) 회원가입 API (/app/users)**

**: 어플리케이션에서 입력받은 회원의 정보를 DB에 일련의 과정을 거쳐 삽입한 뒤, 생성된 회원번호와 jwt를 결과값으로 반환한다.**

****

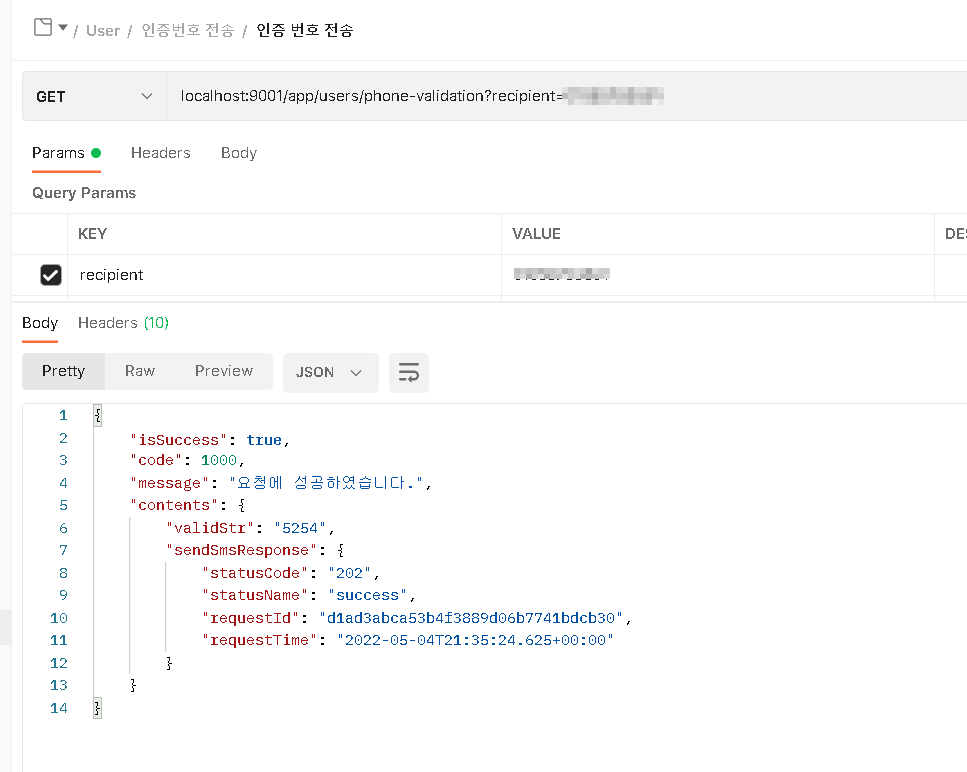
**3) 로그인 API (/app/users/sign-in)**

**: DB에 저장된 id와 pw를 입력하면, 일련의 검증 절차를 거쳐 유저 인덱스와 jwt를 반환하고, 이때 발급된 jwt는 추후 인가가 필요한 API에서 인증시에 사용된다.**

****

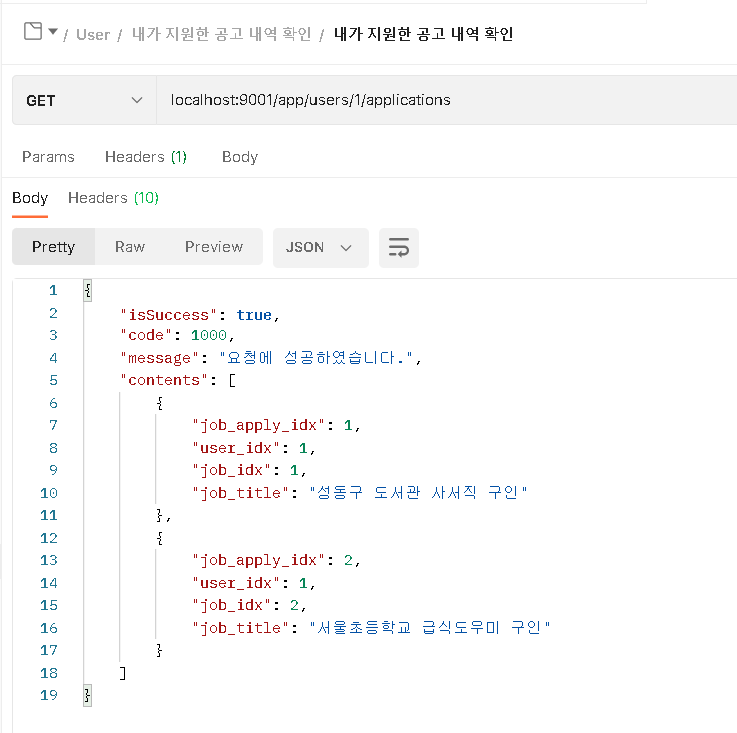
**4) 회원가입 시 문자로 본인인증 API (/app/users/phone-validation?recipient=)**

**: 네이버 SENS 문자 전송 API를 활용하여 서버단에서 네자리 인증 번호를 랜덤으로 생성한뒤, 문자로 비밀번호를 전송하고 이를 어플리케이션에 입력하여 일치하면 나머지 회원가입 절차를 진행할 수 있도록 구현함.**

****

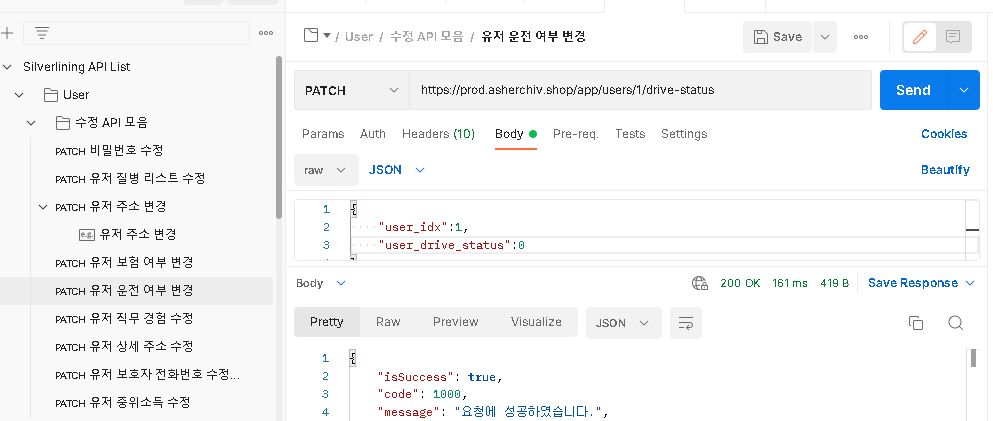
**5) 내가 지원한 공고 내역 확인 API (/app/users/{userNum}/applications)**

**:내가 지원한 구인 공고 내역을 확인할 수 있도록 리스트를 조회하는 API**

****

**6) 유저 정보 수정 API**

**:유저의 비밀번호, 질병리스트, 주소, 보험 여부, 운전 여부, 직무 경험, 상세주소, 보호자 전화번호, 중위소득 등을 수정할 수 있다.**



**- 백엔드 파트 기능 고찰**

**: 유저에게 맞춤형 일자리, 병원, 복지시설, 노인정책 등을 추천하기 위해서 회원가입 시에 입력받고 관리할 데이터가 상당히 많았다. 따라서 모든 데이터를 조회,수정,추가 할 수 있도록 꼼꼼하게 Response/Request를 확인해가면서 구현해야 했다. 또한 모든 앱 내에서의 User Authentication 을 위해 로그인 단계에서 jwt를 발급하고, 인가가 필요한 단계에서 jwt를 통해 유효한 접근인지 확인할 수 있도록 기능을 추가했다.**

1. **일자리- 노년층 맞춤형 구인구직 플랫폼**
2. **개요 및 요구사항**

* **“노인일자리”라 함은 보통 노인이 일을 하는 주체가 아닌, 서비스를 제공하는 대상이 노인인 산업을 일컫는 데에 더 많이 사용되는 말이다. (사회복지사 등) 그러나 초고령화 사회로 진입한 만큼 인구 피라미드가 역삼각형으로 형성될 것이 확실시된 지금, 노동을 원하는 노인 수의 증가 및 노인의 노동력을 활용해야 할 필요성이 증가되고 있다.**
* **현재 대한민국의 대표적인 일자리 플랫폼으로는 ‘알바천국’, ‘알바몬’, ‘잡 코리아’등이 있다. 그러나 대부분 청년층~중년층에 맞춰진 서비스로, 노인의 일자리에 관한 기능은 형식적으로 존재하거나 또는 그 기능이 존재하지 않는다. 정리하자면 노인을 위한, 노인이 주체가 되는 일자리 플랫폼은 매우 부족한 상태이다. 본 프로젝트는 단순히 노인에 대한 일자리 정보 제공 뿐만이 아닌, 노인의 특성에 맞춰 본인이 일자리를 골라 지원하거나, 또는 유저 사전 정보에 맞춘 렌더링 로직을 포함하도록 한다.**
* **플랫폼으로써의 기능을 위하여 사용자 층을 구분할 필요가 있다. 사용자 층을 크게 일반 사용자와 기업으로 나눌 것이며, 각 유저 타입에 따라 사용할 수 있는 기능을 분리한다.**

**일반 사용자**

* + **사용자의 나이, 거주지역, 희망직종, 보유 기능 등을 상세하게 입력받고, 유저의 정보를 토대로 데이터를 필터링해 맞춤형 일자리를 제공한다.**
  + **직관적인 이해가 쉽도록 지도 API를 이용해 거주지역 과 가까운 일자리 정보를 시각화하며, 관심분야에 맞는 일자리 목록을 렌더링**
  + **미리 받아둔 정보를 토대로 지원서 양식을 어플리케이션 상에서 생성해주며, 버튼 하나로 빠른 구직 지원이 가능하도록 함.**

**기업 사용자**

* + **공고를 양식에 맞춰서 등록하면, 지원자 리스트와 각 지원자의 이력서를 한눈에 확인할 수 있도록 함.**
  + **공고에 대한 수정 기능 등을 포함한다.**
* **이상의 내용을 바탕으로, 프로젝트 개발 사항에 대하여 설명하도록 한다.**

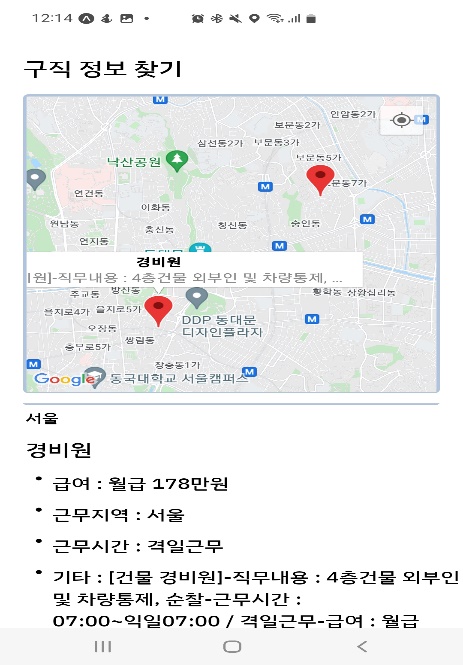
1. **프론트엔드 구현**

**텍스트이(가) 표시된 사진

자동 생성된 설명**

* **앞서 언급했듯이, 구인구직 플랫폼은 일반 회원, 기업 회원에 따라 접근할 수 있는 기능에 차이가 있다.**
* **일반 회원은 구직정보 찾기 기능들을 이용할 수 있으며, 기업 회원은 구직정보 등록 및 지원자 조회 기능 등을 이용할 수 있다.**
* **왼쪽 버튼은 일반 사용자 타입의 계정이 이용할 수 있으며, 우측 버튼은 기업 사용자 타입의 계정이 이용할 수 있다.**

1. **일반 사용자**

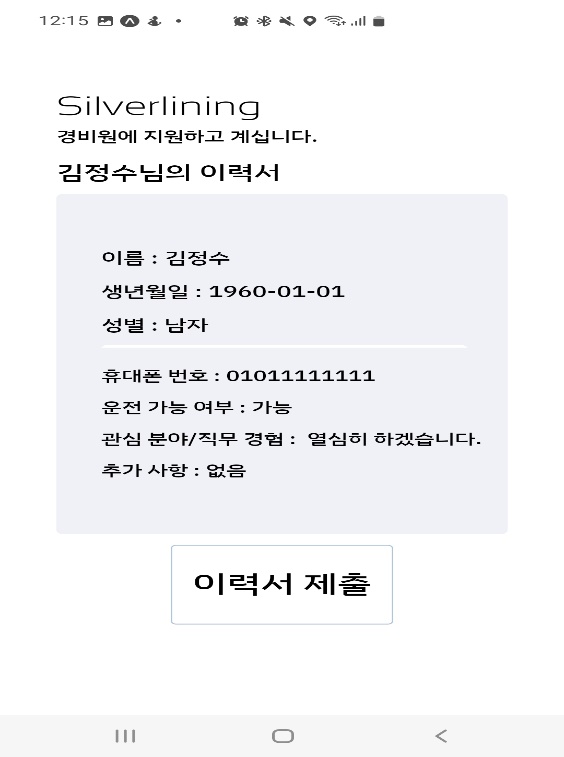
****

* **일반 사용자의 경우, 지도 api 기반으로 유저 근방 1km 이내의 노인 공고를 불러온다. 공고는 유저가 회원가입시에 입력한 정보를 기반으로 불러와진다. 지도에서 마커를 클릭할 경우, 공고의 정보가 리스트 뷰에 불러와지며, 해당 리스트 뷰를 클릭할 경우 아래의 페이지로 이동하게 된다.**

**텍스트이(가) 표시된 사진

자동 생성된 설명**

* **그리드 뷰에서 선택한 공고에 대한 상세 정보를 확인할 수 있다. 해당 정보는 유저가 사전에 입력한 정보를 토대로 렌더링된 공고들이며, 내용으로는 근무조건, 지원조건, 문의정보 등을 포함한다.**

****

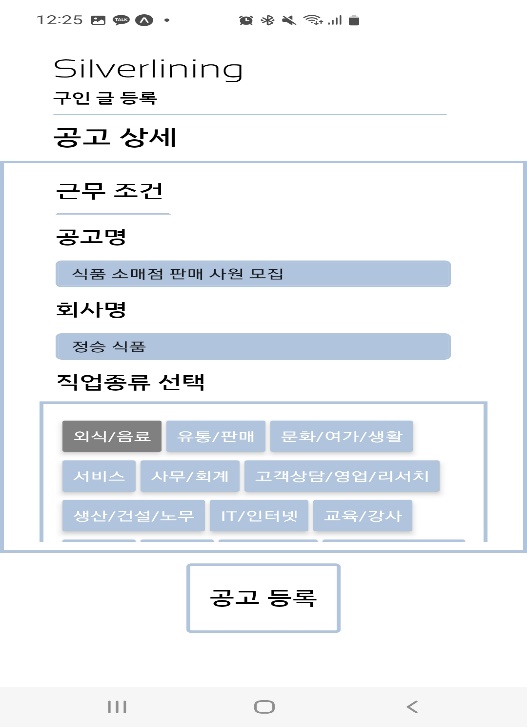
* **상단 이미지는 지원하기 버튼을 클릭할 경우 이동하는 뷰이다. 자동으로 작성된 이력서가 사용자에게 표시되며, 마이페이지에서 해당 지원서에 포함할 내용을 수정할 수 있다.**

1. **기업 사용자의 경우**

**텍스트이(가) 표시된 사진

자동 생성된 설명**

* **기업 사용자는 해당 플랫폼에서 두 개의 기능을 이용할 수 있다.**
* **하나는 구인 글 등록이며, 다른 하나는 작성한 구인 글의 조회 기능이다.**

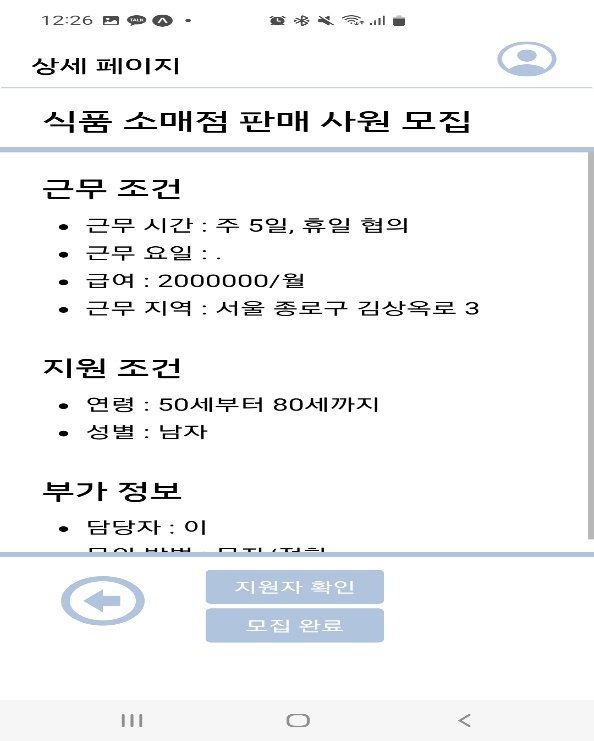
****

* **또한 기업 사용자는 공고를 위처럼 작성할 수 있으며, 공고명, 회사명, 직업 종류, 연령, 성별 등의 정보를 기입하여 요건에 맞는 일반 사용자에게 노출되도록 한다.**
* **작성한 정보는 공고등록 버튼을 통해 데이터베이스에 등록할 수 있다.**

**텍스트이(가) 표시된 사진

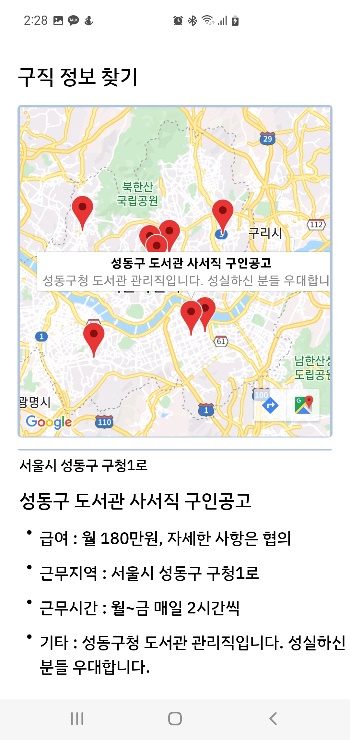
자동 생성된 설명**

* **나의 구인 글 조회 기능을 들어가면, 위처럼 자신이 작성한 글을 조회할 수 있는 페이지로 넘어갈 수 있다.**
* **또한 자신이 작성한 구인 글을 누르면, 아래의 페이지로 넘어가게 된다.**

****

* **해당 페이지에서는 해당 공고의 내용을 확인할 수 있으며, 지원자를 확인하거나 모집완료 처리를 할 수 있다.**

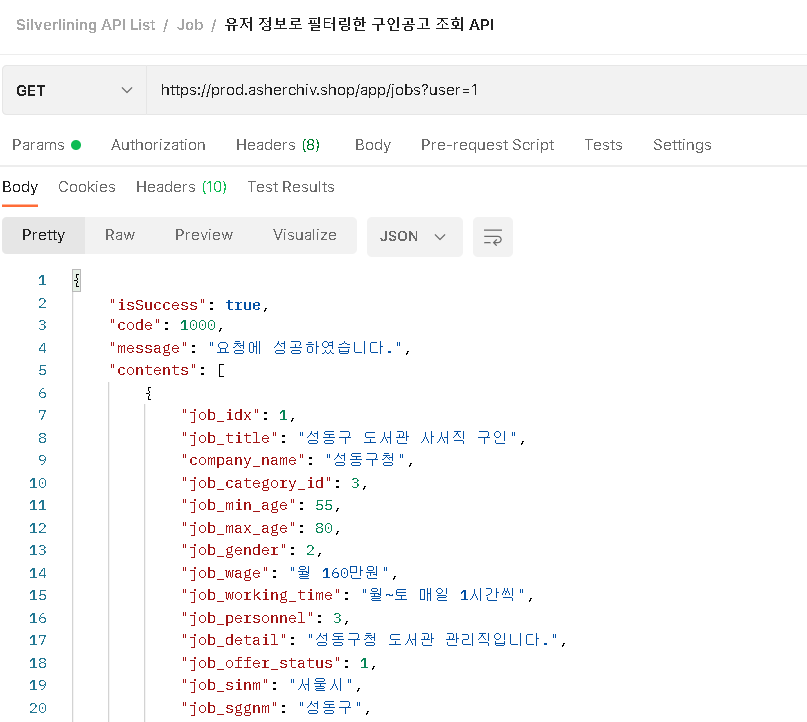
**프론트엔드 기능 고찰**

* **해당 부분은 이번 프로젝트에서 가장 중요한 부분이었다고 해도 과언이 아니다. 그런만큼 다양한 기능이 이용되었는데, Google map의 마커 렌더링용 api를 이용하기 위해 위, 경도 데이터를 추출하여 확보할 필요가 있었다.**
* **해당 부분에서 추출한 위, 경도 데이터를 AsyncStorage에 임시 저장하거나, 또는 데이터베이스에서 입력 받아온 후, 이를 백엔드 api에 요청할 parameter로 이용하였다. 해당 api를 이용해 받아온 response json 데이터를 후처리하여 google map의 Marker api + map method를 이용하여 렌더링하였다.**
* **또한 마커를 정보를 모두 확보한 후 렌더링하는 지연시간이 꽤 걸렸는데, 비동기화 로드를 진행하여 해당 부분을 어느정도 개선하였다.**

**c. 백엔드 API 구현**

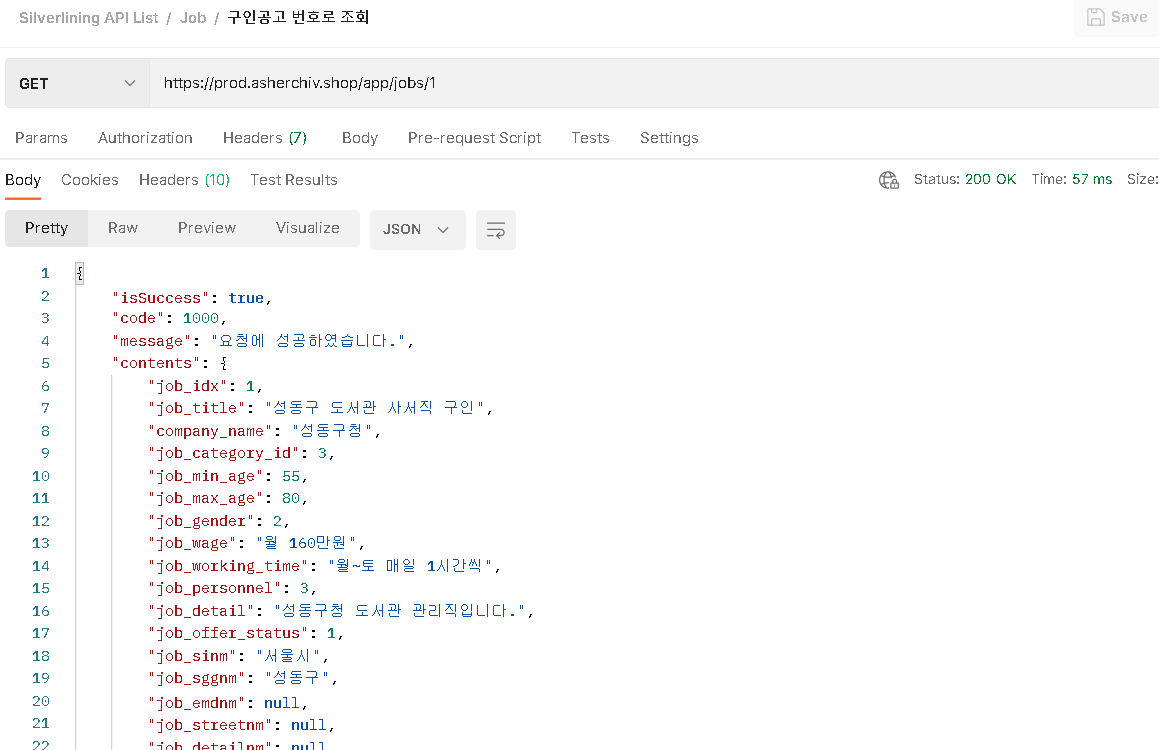
**1) 유저 정보로 필터링한 구인 정보 조회 API (/app/jobs?user=)**

**: 기존에 회원가입시에 입력된 데이터를 통해 구인공고를 필터링하며, 필터링의 기준은 나이, 성별, 직장과 유저간의 거리등을 고려하였다. 조회 시에는 구인 공고의 모든 정보를 response로 출력한다.**

****

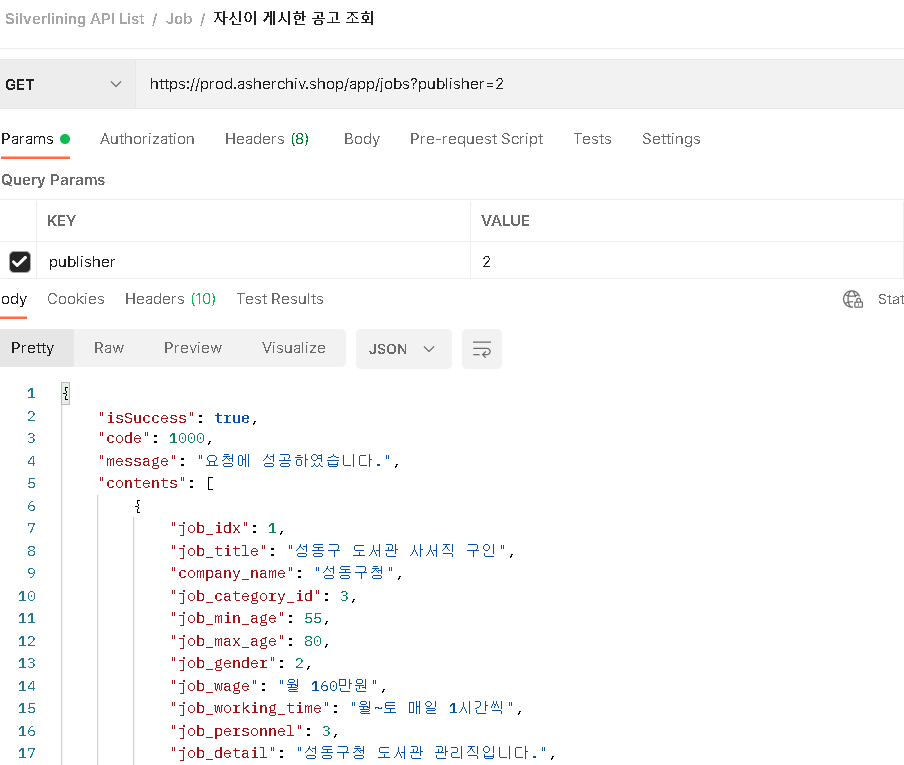
**2) 구인 공고 번호로 조회 API (app/jobs/:jobIdx)**

**: 구인 공고에 부여된 번호를 통해 조회하며, 구인 공고 게시 시에 입력받은 모든 정보를 조회한다.**

****

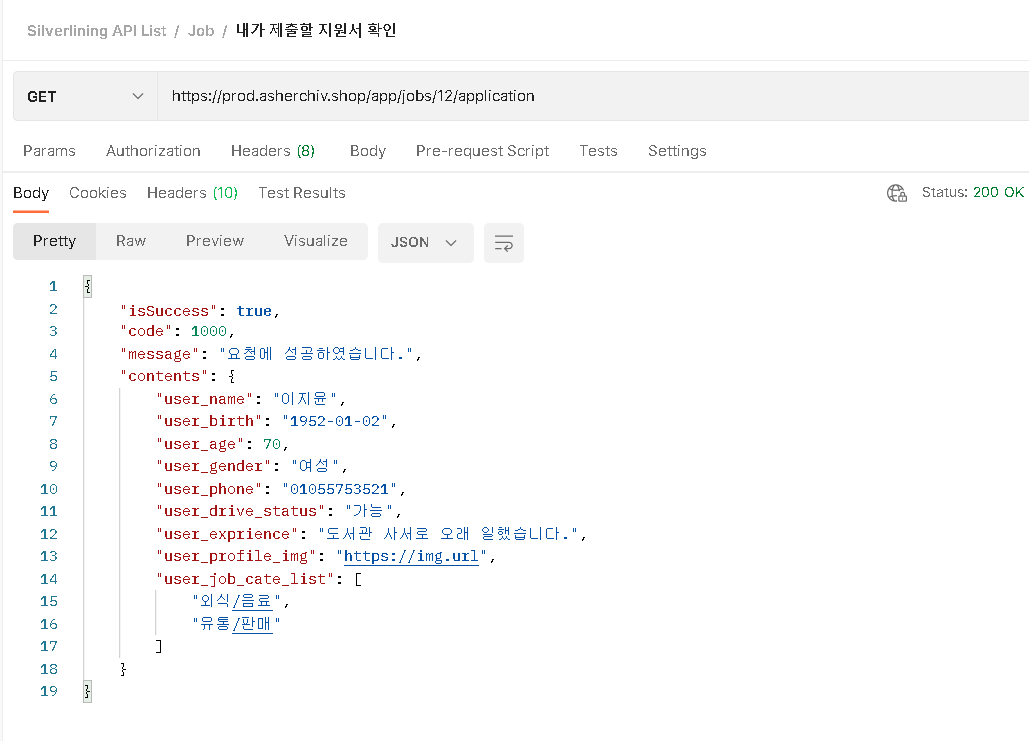
**3) 자신이 게시한 공고 조회 API (/app/jobs?publisher=)**

**: 기업이 게시한 공고에는 회원 번호를 저장하는데, 이를 통해 기업에서는 직접 게시한 공고만을 조회할 수 있다.**

****

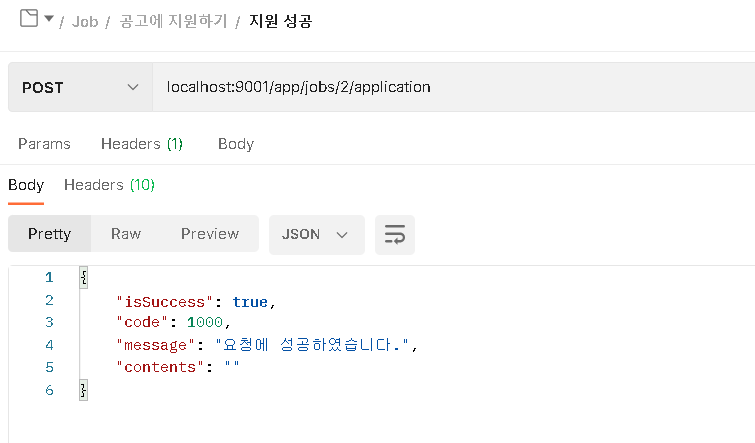
**4) 내가 제출할 지원서 확인 API (/app/jobs/:jobIdx/application)**

**: 유저가 공고에 지원하면, 먼저 유저가 제출할 이력서를 검토할 수 있도록 유저의 관련된 데이터를 조회한다.**

****

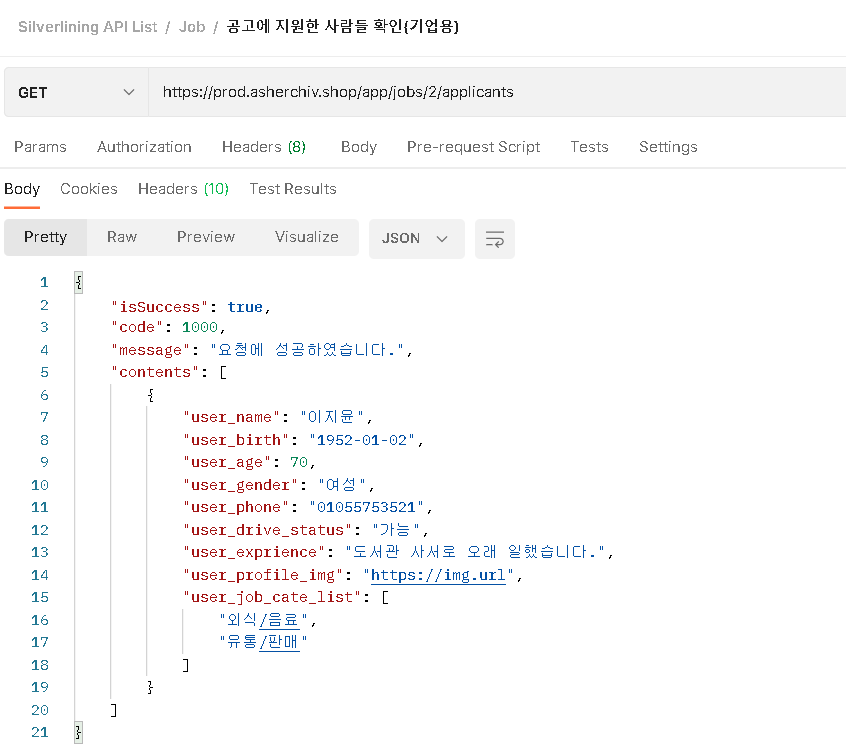
**5) 공고 지원 API (/app/jobs/:jobIdx/application)**

**: 게시된 구인공고에 지원할 수 있도록 하는 기능이다.**

****

**6) 공고에 지원한 사람들 확인 API (/app/jobs/:jobIdx/applicants)**

**: 기업 차원에서 자신의 공고에 지원한 사람들의 정보를 확인할 수 있는 API이다.**

****

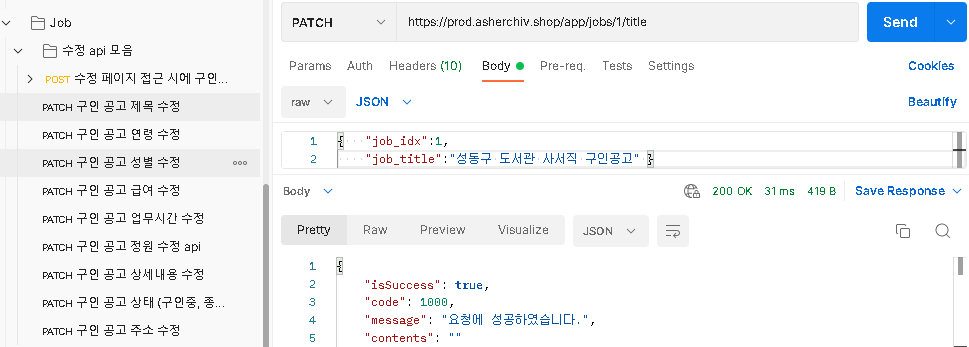
**7) 구인 공고 게시 API (/app/jobs)**

**: 기업 측에서 직접 구인 공고를 게시할 수 있도록 정보를 입력받고, DB에 삽입하는 기능이다.**

****

**8) 구인 공고 내용 수정 api (PATCH)**

**: 구인공고를 게시한 사람만이 페이지에 접근 가능하며, 공고의 제목, 연령, 성별, 급여, 업무시간, 정원, 상세내용, 상태, 주소등을 수정할 수 있도록 하였다.**



**- 백엔드 파트 기능 구현 고찰**

**: 최대한 유저에게 맞춤형 정보를 제공할 수 있도록 구인 공고 게시 시에 필터링에 용이한 명확한 데이터(최소/최대 연령, 성별, 직장 위/경도)로 입력받았고, 유저가 구인 공고를 조회할 때 유저의 연령, 유저 거주지와의 거리, 성별 등을 고려해서 공고를 필터링하였다. 또한 유저가 따로 이력서를 작성할 필요가 없게끔 회원 가입시에 입력받은 데이터를 가공해 앱이 자동으로 지원서를 완성하고 제출할 수 있도록 지원시에 불편할 수 있는 과정을 최소화하였다. 또한 유저가 지원한 공고 또한 바로 조회할 수 있도록 하여 유저 단에서 최대한 구인구직 관련 기능을 쉽게 사용할 수 있도록 신경 썼다.**

**또한 권한이 있는 기업회원만이 공고를 게시하고, 추후에 수정도 할 수 있도록 하였으며, 자신이 게시한 공고와 공고에 지원한 사람들의 이력서를 바로 확인할 수 있도록 하는 여러 조회 기능을 추가했다.**

**d. 데이터 크롤링 파트.**

**1) 데이터 크롤링을 진행한 사이트 링크는 다음과 같다.** [**https://goldenjob.or.kr/job/find-person.asp**](https://goldenjob.or.kr/job/find-person.asp)

**해당 사이트는 서울시 어르신 취업 진행 센터의 웹사이트로, 하단의 페이지 ui에서 구직 중으로 설정되어 있는 페이지만 데이터를 크롤링 하였다.**

**지도이(가) 표시된 사진

자동 생성된 설명**

**2) 크롤링은 주피터 노트북을 사용하여 BeautifulSoup 라이브러리를 활용하여 페이지 정보를 엑셀에 정리하였다.**

**텍스트이(가) 표시된 사진

자동 생성된 설명**

**3) csv파일 형식으로 정리된 정보에서 추가적으로 요구 되는 정보에 한하여 구글링을 통해 엑셀에 추가하여 요구되는 데이터들을 api 명세서에 맞게 정리하였다.**

**테이블이(가) 표시된 사진

자동 생성된 설명**

1. **의료/복지 관련 - 맞춤형 주변 병원/복지시설 조회 기능**
   1. **개요 및 요구사항**
      1. **노년층은 각자가 갖고 있는 질환이나 건강 관리에 대한 관심도가 높다는 점을 반영하여, 사전에 입력한 질병 정보를 적극 활용한다.**
      2. **사전에 입력 받은 정보를 토대로 주변에 위치한 병원 또는 복지시설을 추천해주고, 위치를 지도로 시각화 하여 안내해주는 기능을 개발하고자 한다.**
      3. **국가 공공데이터포털에서 제공하는 병원, 노인 복지/여가/요양 시설 위치 API와 지도 관련 외부 API를 적극 활용해서 개발을 진행하도록 한다.**
   2. **프론트엔드 구현**

* **해당 부분은 유저의 정보를 기반으로 하여, 위도, 경도 데이터를 통해 주변 병원 및 복지시설을 렌더링하는 기능이다. 지도 api 상에서 마커로써 렌더링된 ui를 통해, 병원 및 복지시설 정보를 받아올 수 있다.**

**텍스트이(가) 표시된 사진

자동 생성된 설명**

* **사용자는 위의 두 가지를 사용할 수 있으며, 두 기능이 매우 흡사하므로 위의 기능을 설명하여 아래 기능의 설명을 포함하도록 한다.**

****

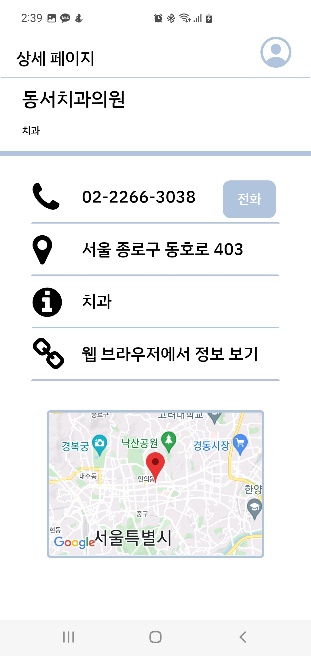
* **자신의 위치를 기반으로 지도의 초기 중심점이 설정되며, 반경 1km내의 병원, 의료복지 시설 정보를 가져온다. 마커를 클릭할 경우, 하단 리스트 뷰에 간략한 정보가 표시된다. 해당 리스트 뷰를 클릭할 경우, 아래 페이지로 이동한다.**

**텍스트이(가) 표시된 사진

자동 생성된 설명**

* **위와 같이 상세한 페이지로 이동하게 되며, 표시되는 url을 클릭하면 웹 뷰로 넘어가게 된다. 웹 뷰 페이지는 아래와 같다.**

****

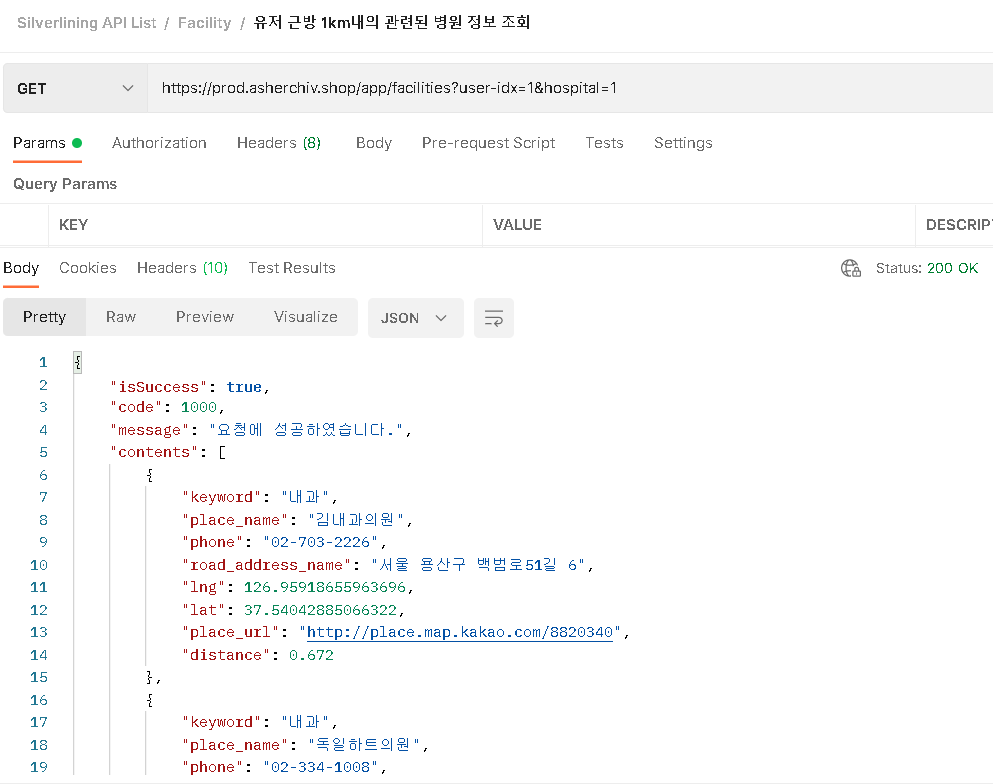
** 프론트엔드 기능 고찰**

* **해당 부분은 구인구직 파트에서 이용한 테크닉을 응용하였다. 지도의 그리드 뷰에서 넘어온 페이지가 바로 우측의 ui인데, 지도 렌더링 부분은 비동기화 처리하여 페이지가 렌더링 되는 속도를 개선하였고, 나머지 전화번호, 주소 등의 데이터는 react-navigation open library의 route 기능을 이용하여 이전 페이지로부터 데이터를 사전에 담아 넘겨줄 수 있도록 하였다.**

**c. 백엔드 API 구현**

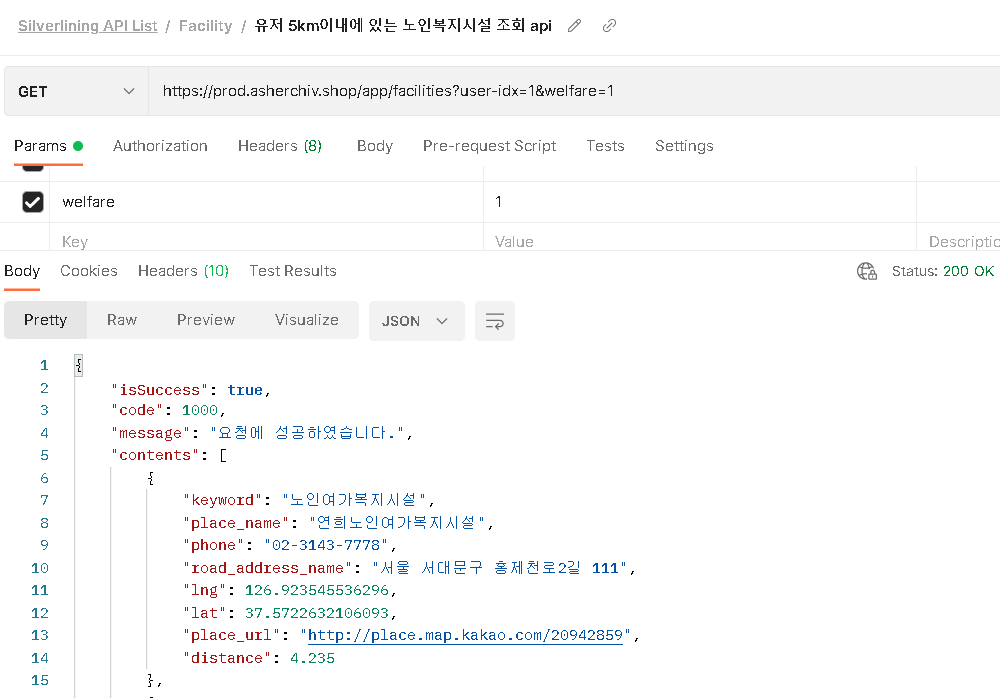
**1) 유저의 기저질환과 관련된 반경 1km 내의 병원 정보 조회API (/app/faclilities?user-idx=1&hospital=1)**

**: 일전에 회원가입시에 유저가 갖고 있는 기저질환과 거주지 정보를 조사받았다. 이를 DB에 저장해두고, 이 질환과 관련된 병원의 진료과 리스트를 추출하였고, 이 리스트를 검색 키워드로 하여 카카오 로컬 API를 통해 유저 근방 1km내의 관련된 병원 정보를 출력했고, 이 결과에서 필요한 정보만을 추출해 가공한 뒤 해당 키워드 별로 분류해 response로 보내게 된다.**

****

**2) 유저 반경 5km 내의 노인 복지 시설 조회 API (/app/faclilities?user-idx=1&welfare=1)**

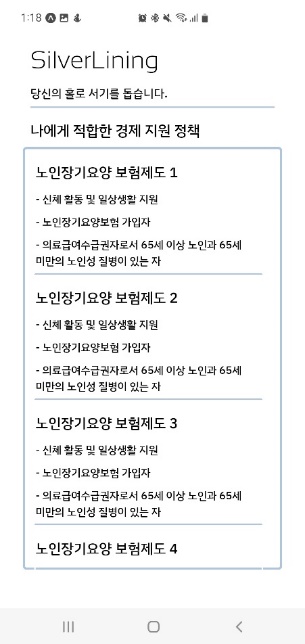
**: 해당 API에서는 앞의 API와 같은 방식으로 키워드와 유저 위치를 통해 시설 정보를 조회하였으며, 이번에는 5대 노인 복지시설 (주거복지시설/ 여가복지시설/ 재가노인복지시설/ 노인보호전문기관/ 노인일자리지원기관)을 키워드로 하여 유저 근방에 있는 노인복지시설 정보를 추출하여 응답으로 전송하였다.**

****

**- 백엔드 파트 기능 구현 고찰**

**: 회원가입시에 입력받은 질병 리스트를 키워드로 하여 유저 거주지 근처에 있는 관련된 의료시설 데이터를 조회 할 수 있도록 하였다. 최대한 많고 상세한 병원/복지시설 관련 데이터를 조회할 수 있도록 카카오 local api를 사용하였고, api를 통해서 조회된 정보를 가공하여 해당 장소의 이름, 전화번호, 주소, 위/경도, 카카오 맵 url, 유저와 시설 사이의 거리 등을 response에 담았고, 이를 목록으로 조회할 수 있도록 하였다.**

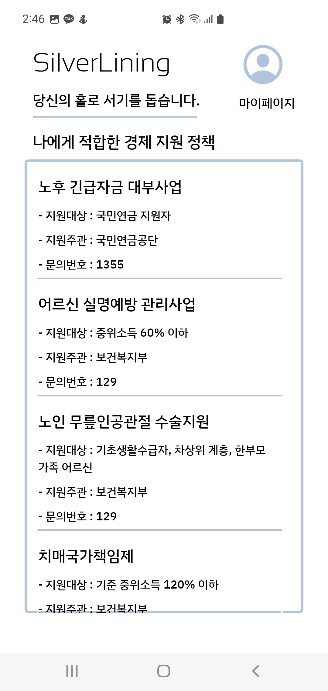
1. **경제적 지원 관련 - 노년층 지원사업/정책 알리미**
2. **개요 및 요구사항**
   * 1. **현재 노인관련 정책, 또는 지원 사업을 정부에서 여러가지 추진하고 있으나, 디지털 기기에 친숙하지 않는 노인분들은 이를 직접 찾아보고 지원하거나 활용하는데 어려움이 많은 것으로 판단된다.**
     2. **해당 유저의 경제적 상태, 질병, 거주 지역 등을 가입 시에 입력 받아 각자에게 적합한 정책을 맞춤형으로 알려주는 서비스의 필요성을 반영하도록 한다.**
     3. **챗봇, 문자 메세지, 푸시 알림 등을 활용해 해당 대상에게 정보를 안내하고 쉽게 신청할 수 있도록 유도하는 방식으로 구현하였다.**
3. **프론트엔드 구현**

****

**텍스트이(가) 표시된 사진

자동 생성된 설명**

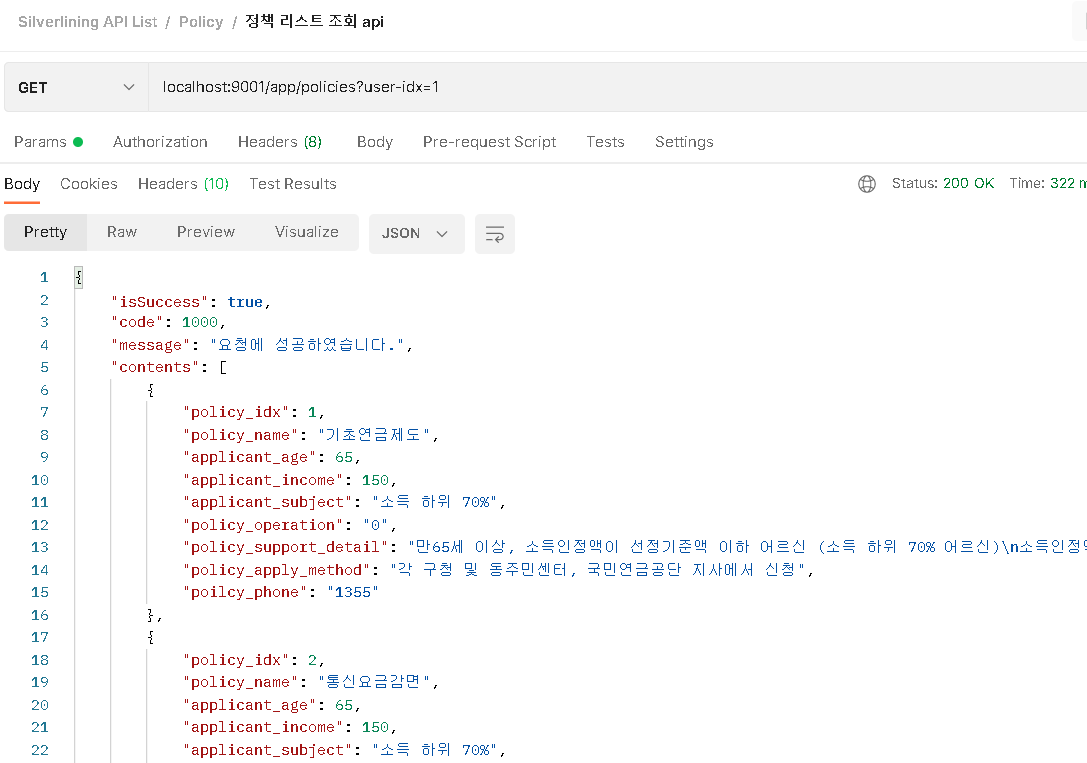
* **해당 부분은 크롤링 데이터를 기반으로, 데이터 베이스에서 유저와 매칭되는 부분이 있는 공고를 렌더링해주는 기능을 한다.**
* **위와 같은 형식의 그리드 형식 뷰로 설계하였으며, 각 그리드 내부 컨텐츠를 누를 시 우측 이미지와 같이 해당 공고의 정보를 조회할 수 있다.**
* **공고의 특성 상 한눈에 여러 개의 정보를 확인해야 적합한 것을 고르기 용이할 것이므로, 횡단으로 UI를 구성하는 것을 지양하였다. 따라서 상단 좌측처럼 UI를 구성하였다.**

**프론트엔드 기능 고찰**

* **해당 페이지의 경우, 크롤링 데이터를 이용하여 이를 유저에게 적합한 형태로 가공하는 것이 주된 과제였다.**
* **json형태로 response로 돌려주는 백엔드 api로부터 데이터를 받아온 후, 해당 데이터들을 ui 내부에 가공하여 map method를 통해 렌더링하였다.**
* **해당 그리드 뷰의 ui를 누르면 상세 페이지로 이동하게 되는데, 그리드 뷰에 노출된 정보를 다시 가공하고, route기능을 이용하여 데이터를 다른 페이지로 전송하는 등의 데이터 전송 관련 로직을 추가로 작성해야 했다.**
  1. **백엔드 API 구현**

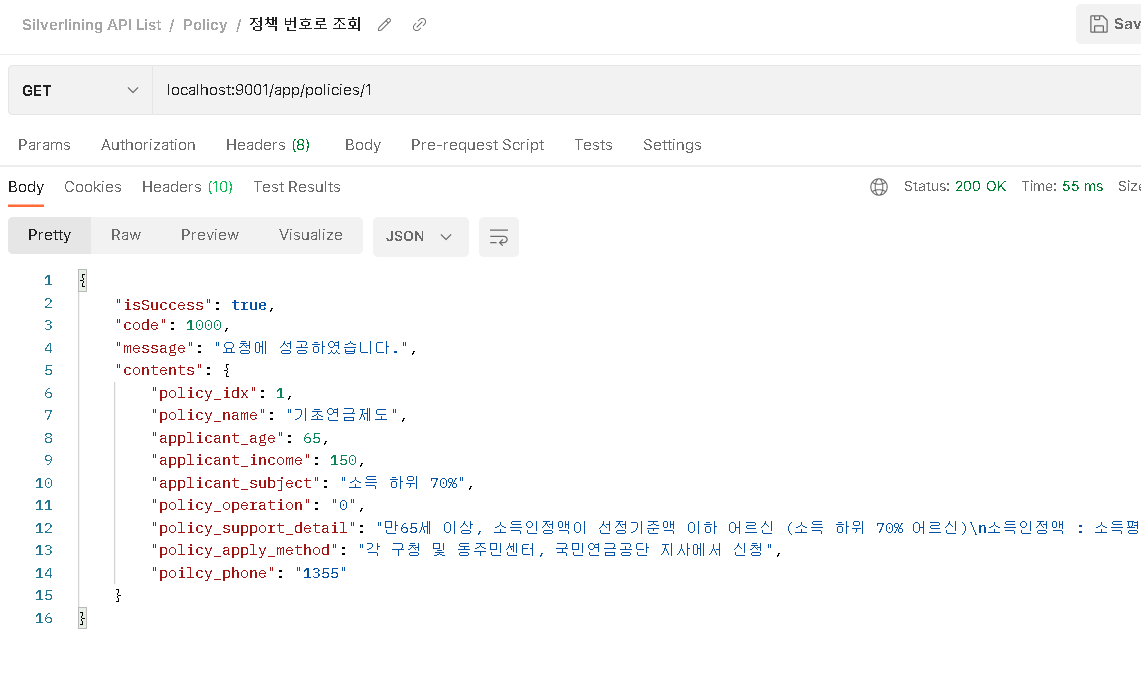
**1) 유저 관련 정책 리스트 조회 API (/app/policies?user-idx=1)**

**: 여러가지 노인관련 복지정책/지원사업에 대한 정보들을 크롤링을 통해 모아왔고, 이를 가공해 DB에 저장해둔 상태이다. 이때 유저와 관련도가 높은 정책을 추천하기 위해서 유저의 나이, 중위소득을 토대로 해당되는 정책의 정보만 조회하도록 하였다.**

****

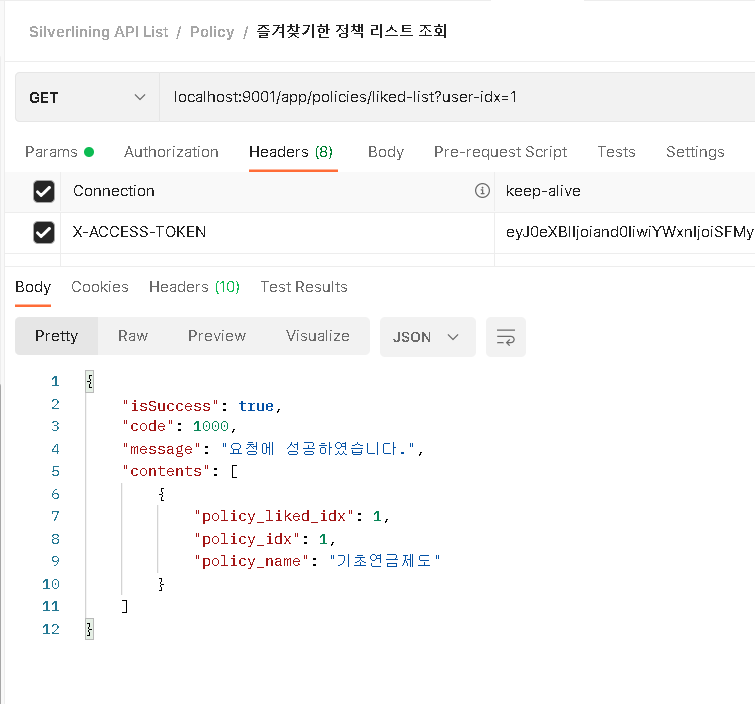
**2) 정책 번호로 조회 API (/app/policies/:policyIdx)**

**: 정책에 모두 인덱스를 부여해놓았는데, 해당 인덱스를 통해 정책의 정보를 조회할 수 있도록 하였다.**

****

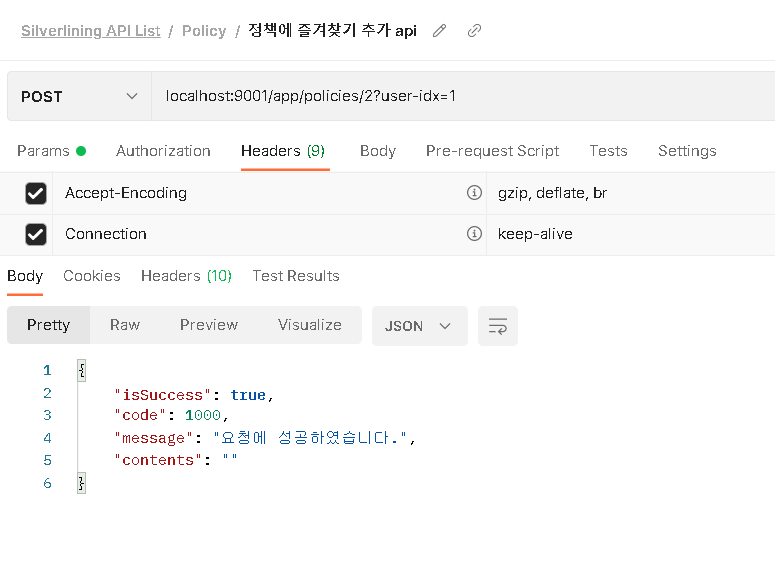
**3) 즐겨찾기한 정책 리스트 조회 API (/app/policies/liked-list?user-idx=1)**

**: 유저가 관심있어하는 정책을 즐겨찾기할 수 있도록 하며, 이를 저장해두고 추후에 유저가 즐겨찾기 해둔 정책의 리스트르 조회할 수 있도록 하였다.**

****

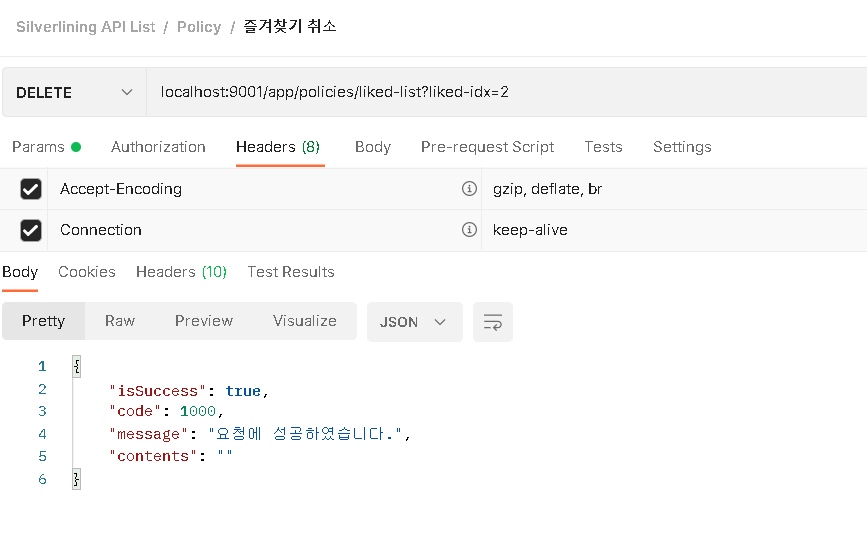
**4) 정책에 즐겨찾기 추가 API (/app/policies/:policyIdx?user-idx=1)**

**: 유저가 관심있어 하는 정책을 즐겨찾기에 추가할 수 있도록 구현하였다.**

****

**5) 정책에 즐겨찾기 취소 API (/app/policies/liked-list?liked-idx=2)**

**: 또한 위에서 즐겨찾기에 추가해둔 목록을 삭제할 수도 있도록 구현하였다.**

****

**- 백엔드 파트 기능 고찰**

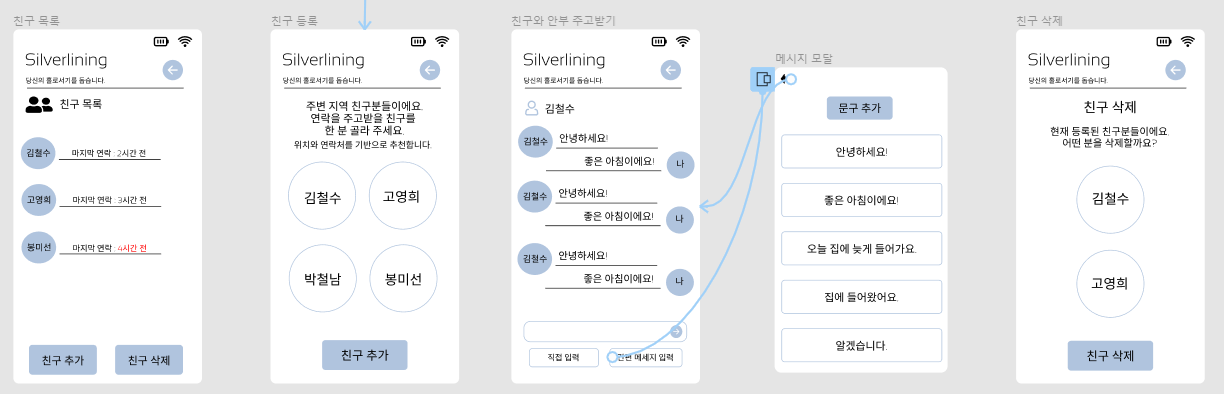
**: 역시 위의 기능들과 비슷하게 유저에게 최대한 맞춤형 정보를 조회할 수 있도록 유저의 연령, 중위소득등을 고려하여 DB에 저장된 정책 데이터를 필터링하였다. 또한 추후에 유저가 관심있는 정책을 다시 조회할 수 있도록 즐겨찾기 추가/삭제 기능을 구현하였다.**

**d. 데이터 크롤링**

**1)경제적 지원에 해당하는 크롤링 데이터는 정리된 웹페이지가 없고, 요구되는 API의 데이터 형식에 맞도록 정리되어 있지 않았으며, 노출되어 있는 데이터가 적다는 사항을 고려하여 여러 경제적 지원 내역들을 구글링을 통해 정리하고 엑셀 파일을 API 명세에 맞게 정리하여 바로 CSV 파일 형태로 제공하여 사용하는 방식으로 데이터 크롤링을 진행하였다.**

****

1. **관계 단절 관련 - 주변 친구 맺기**
2. **개요 및 요구사항**
   1. **본 기능은 노인의 고립으로 인해 위기 상황이 닥칠 경우 제대로 된 대처를 할 수 없는 상황을 방지하고자 한다.**
   2. **따라서, 가까운 주변 유저와 친구를 맺어 서로의 안부를 확인할수 있도록 기능을 개발하도록 한다.**
   3. **안부 연락은 채팅 형식으로 개발하여 유저들 사이에 간단하게 안부를 물을 수 있도록 구현하도록 한다.**
3. **Application 작동 프로세스 - UI/UX 설계도**

**c) 프론트엔드 UI & 로직**

**관계단절 방기 기능 메인 UI & 로직**

****

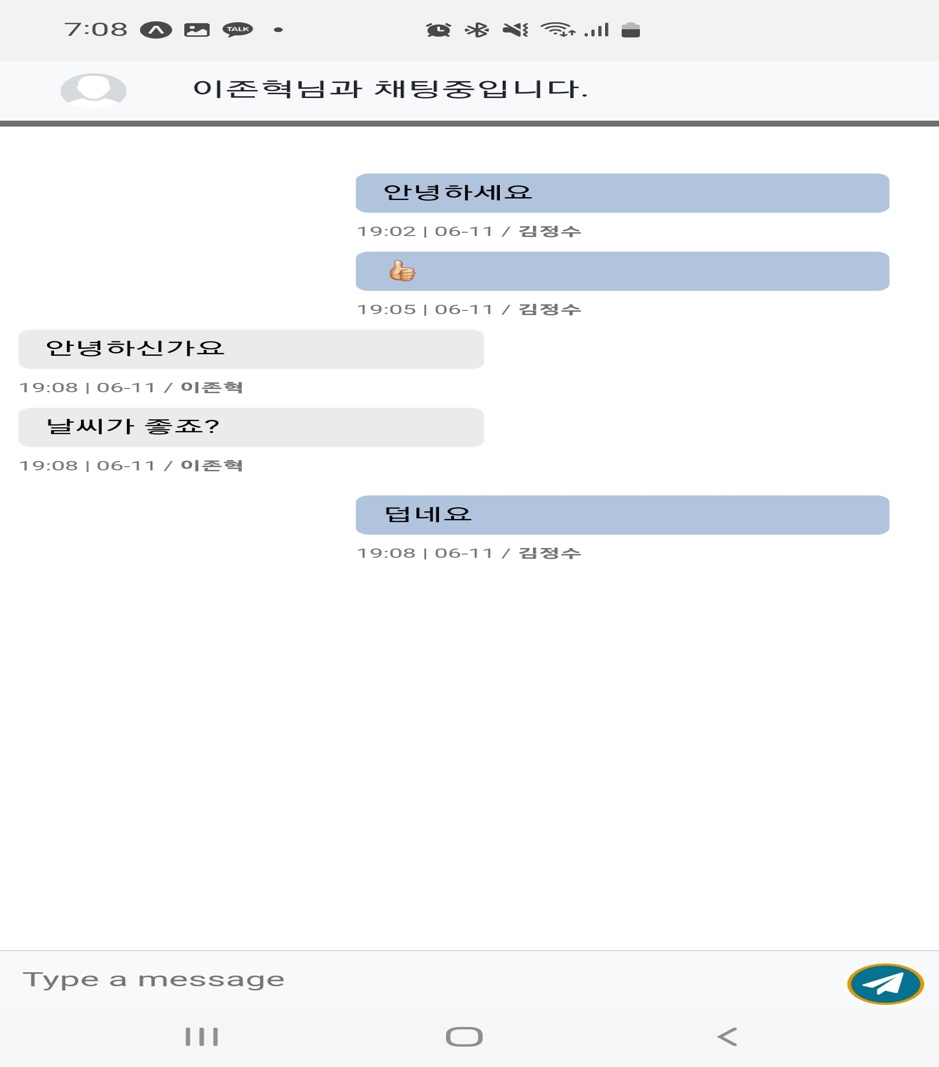
* **위는 채팅방의 메인 페이지 UI이다. 친구로 등록된 사용자는 원형의 인터페이스로 생성되어 렌더링되며, 유저와 상호작용 할 수 있도록 한다.**
* **해당 원형 인터페이스는 백엔드 API로부터 호출해온 JSON데이터를 파싱하여, 리스트 형식으로 나열한 후 이를 최근에 등록된 순으로 화면에 렌더링 한 것이다.**
* **또한 우상단에서 확인할 수 있는 바와 같이, 하단의 친구 추가 버튼을 통해 전화번호를 통해 새로운 사용자를 친구로 추가할 수 있도록 하였다.**

**텍스트이(가) 표시된 사진

자동 생성된 설명텍스트이(가) 표시된 사진

자동 생성된 설명**

* **친구 삭제 버튼을 누른다면 친구가 삭제되고, 친구 목록에서 삭제된다.**
* **즉각적인 로직의 반영을 위하여 해당 이벤트가 발생할 경우 백엔드 API로부터 JSON데이터를 다시 받아왔으며, 해당 데이터를 기존의 렌더링용 데이터에 덮어씌워 출력함으로써 UI의 즉각적인 반영이 일어나도록 유도하였다.**

****

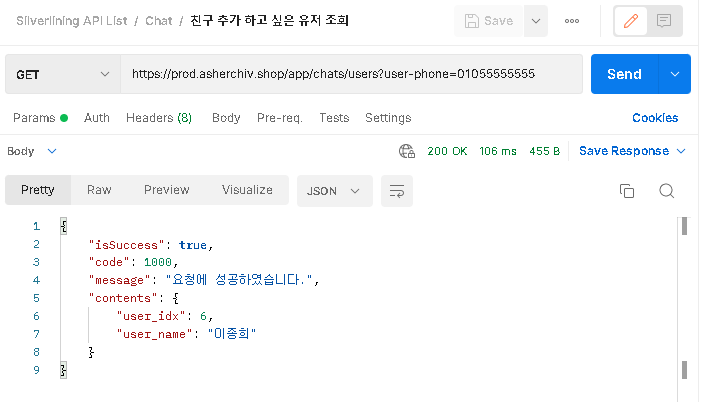
* **친구와 이야기 버튼을 누르면 웹 뷰로 채팅을 할 수 있는 페이지로 넘어가 상대방과 채팅을 주고 받을 수 있게 된다.**
* **채팅 메세지 아래에는 보낸 시각, 보낸 날짜 등이 표시된다.**

**프로트엔드 구현 고찰**

* **본래 모바일 어플리케이션 내에서 모든 기능을 구현하기 위하여 EventSource 등의 라이브러리를 추가로 사용하였으나, 해당 방법은 두 가지 문제가 존재하였다.**
* **첫 번째로는 React Native의 EventSource 라이브러리가 프로젝트 목적에 맞지 않은 채로 작성된 채 개발이 중단되었으며, 두번째로는 유저 피드백 중 채팅 텍스트를 마음껏 늘리고 줄이는 요구사항이 제시되었다는 점이다.**
* **Server Sent Event를 프론트엔드 단에서 이용하기 위해서는 Event Source가 꼭 필요하였는데, 이를 해결하기 위해 web상에서 이를 모바일 해상도에 맞게 구현하고, react native의 web-view라이브러리를 활용해 둘을 연동시키는 방식을 취하였다.**

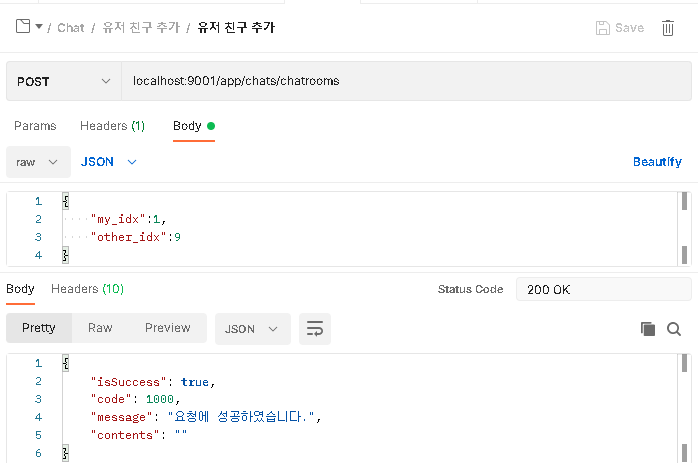
**d)** **백엔드 파트 구현**

**1) 친구 추가하고 싶은 유저 조회 (app/chats/users?user-phone=)**

**: 유저가 친구로 추가하고자 하는 다른 유저의 전화번호를 입력하면 해당 유저의 인덱스와 이름을 조회하며, 만약 앱 유저가 아니면 유저가 아니라는 에러를 발생시킨다.** 

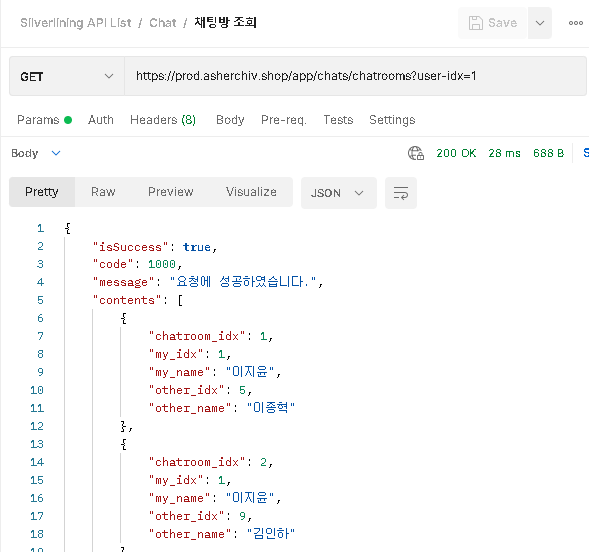
**2) 유저 친구 추가 (app/chats/chatrooms)**

**: 이전에 조회한 유저의 인덱스를 이용해 친구로 추가하며 ,이미 친구가 맺어진 경우와 존재하지 않는 유저로 추가 시도를 하는 경우에는 에러가 발생한다.**



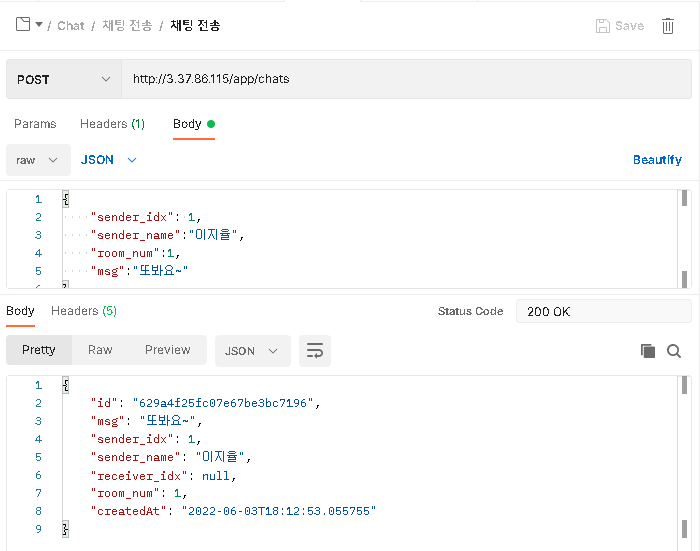
**3) 채팅방 조회 (/app/chats/chatrooms?user-idx=)**

**: 해당 유저와 친구가 되어 채팅을 주고받을 수 있는 모든 채팅방을 조회하며, 추후에 채팅 내역 조회와 저장을 위해서 채팅방 인덱스, 나와 친구의 인덱스와 이름까지 조회하도록 response에 포함시켰다.**



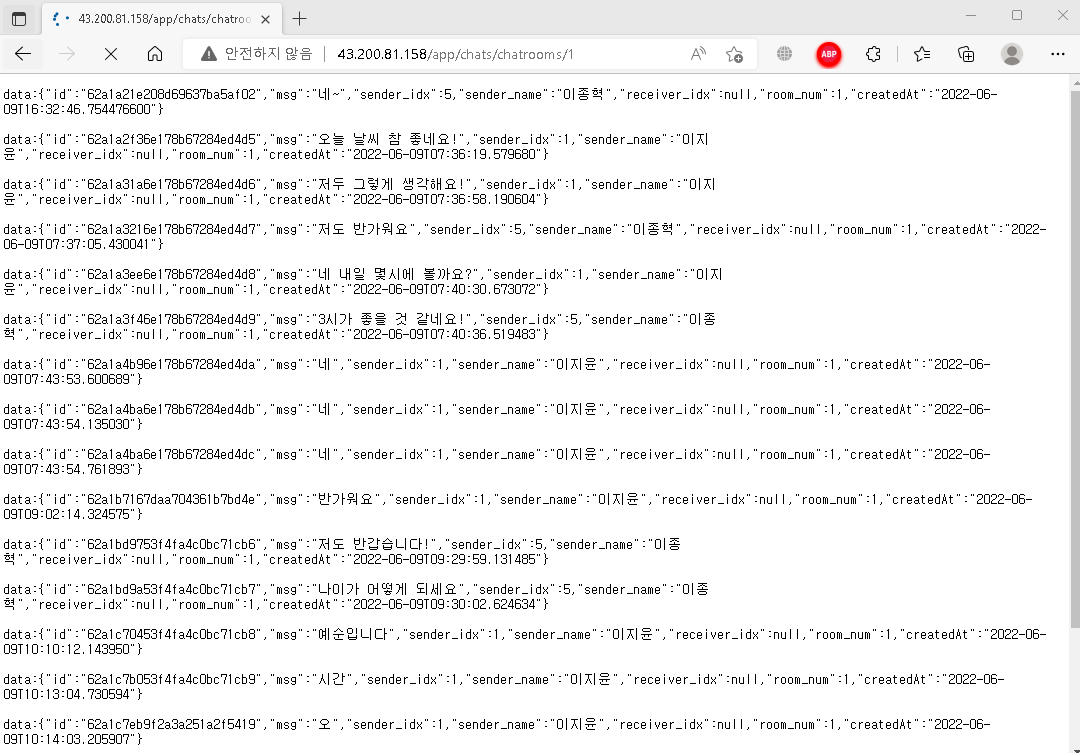
**5) 채팅 전송 (/app/chats)**

**: 채팅 내용, 채팅방 번호, 전송자 idx,이름을 포함해서 request body를 구성해서 채팅을 전송하면 DB에 저장되고, DB에 저장된 내역을 response로 확인할 수 있도록 하였다.**



**6) 이전 채팅 내역 조회 (/app/chats/chatrooms/{idx})**

**: Spring Webflux의 Flux 객체를 사용해서 데이터를 끊임없이 흘러들어오게 하며, 해당 채팅방에 새로운 데이터가 발생해도 지속적으로 채팅 내역을 조회할 수 있도록 구현하였다.**



**7) 채팅방 뷰 구현**



**Api 구현 만으로는 채팅이 제대로 이루어지고 있는지 확인이 어려웠기 때문에 백엔드 파트에서 채팅방 뷰까지 구현하고, api를 연동하면서 테스트를 진행하였다.**

**이전에 채팅방 목록을 조회할 때**

**Response로 받아온 채팅방 index, 나의 index, 이름, 상대방 이름 을 이용하여 이전 채팅 내역을 조회하고, 나와 상대방을 분별하며, 누가 어떤 채팅방에 채팅을 보냈는지 명시하여 DB에 채팅을 저장한다.**

**또한 아래 전송창에서 채팅을 보내면 DB에 채팅이 저장되는데, 이때 채팅방을 조회하는 api(Server)에 새롭게 데이터가 흘러들어와 event가 발생하면 채팅창을 초기화하여 새롭게 렌더링해 새로운 채팅까지 뷰에 추가하여 실시간으로 채팅을 확인할 수 있도록 하였다.**

**- 백엔드 파트 구현 고찰**

**: 채팅 기능이 가장 최종 구현까지 많은 곡절이 있었던 파트이다. 기존에 구현하고자 하는 방식이 상당히 복잡하고 시간이 걸릴 듯해 여러가지 방법을 공부하고 테스트해 보다가 앱에 가장 적합한 방식으로 채팅을 구현하였다.**

**- 채팅할 수 있는 친구를 맺고, 조회하고, 삭제하는 부분은 기존의 유저 database를 사용해야 했기 때문에 이 api는 기존의 프로젝트와 통합하여 구현하였다.**

**- 실제로 채팅을 주고받을 때 사용한 api인 이전 채팅 내역 조회, 채팅 저장은 기존의 Servlet 기반 Spring boot 프로젝트와 분리하여 좀더 경량화되고 빠르게 비동기적으로 작동하는 Netty Server- Reactive web 기반 프로젝트로 구현하였으며, 이를 통해서 데이터를 Non-blocking 형태로 끊임없이 새롭게 발생하는 채팅 데이터가 흘러들어올수 있도록 하였다. DB 역시 기존의 MySQL RDBMS가 아닌 NoSQL, 비동기적으로 작동하는 MongoDB 클러스터를 하나 생성하여 채팅 내역을 저장하였다. 또한 별도의 설정이 필요한 웹소켓보다 간단하게 기존의 HTTP 프로토콜을 기반으로 최초연결 이후에 서버에서 클라이언트로 데이터를 지속적으로 realtime streaming 할 수 있는 SSE 프로토콜을 이용해 DB에 채팅이 저장되는 event가 발생할 때마다 채팅 페이지를 초기화해 방금 전송된 채팅을 렌더링할 수 있도록 구현하였다.**

**4. 어플리케이션 기술 요약 / 업무 분담/개발 일정**

1. **Server 파트 - 이지율**

**: UX/UI 디자인/ 서버-DB 관리/ ERD 작성 / 백엔드 REST API 개발**

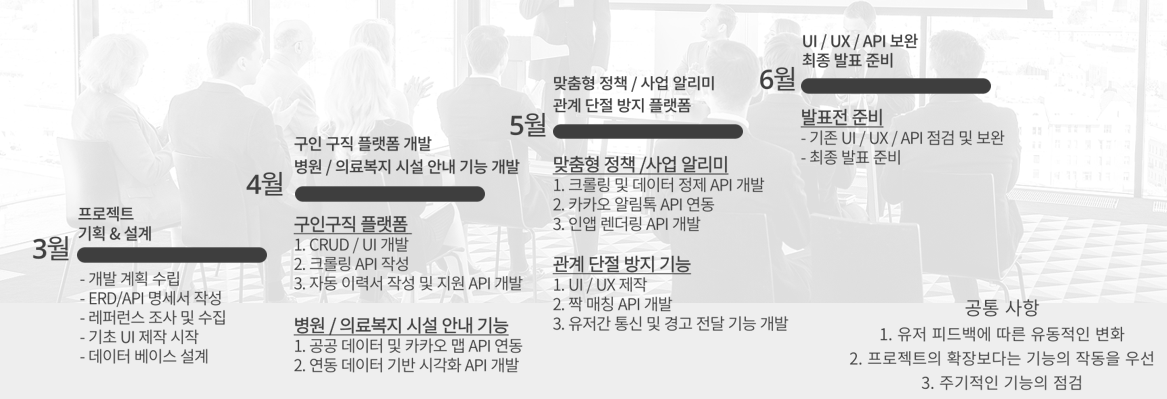
* 1. **Cloud Server - AWS ec2**
  2. **Server Engine- Nginx**
  3. **DB - AWS RDS Database/MySQL**
  4. **Backend Language- Spring Boot**

1. **Front-End 파트 - 이종혁**

**: UI/UX 개발 및 REST API 방식 로직 개발**

1. **Framework - React Native, react, react-native, node.js**
2. **UI Library - styled-components, Material-UI, iconfy...**
3. **Open API : Kakao map, Google map, React-native web-view, EventSource…**
4. **데이터크롤링 - 임유한**
5. **Data crawling - Python (BeautifulSoup)**
6. **Framework - React Native**
7. **Stack - mongoDB**

* **프로젝트 개발 일정**

****

**5. 참고 사이트/ 레퍼런스**

* **전국 병의원 찾기 API**
  + [**https://www.data.go.kr/data/15000736/openapi.do**](https://www.data.go.kr/data/15000736/openapi.do)**: 병‧의원 정보를 시도/시군구/진료요일/기관별/진료과목별로 조회할 수 있다.**
* **서울시 노인 여가시설 데이터**
  + [**https://www.data.go.kr/data/15046517/fileData.do**](https://www.data.go.kr/data/15046517/fileData.do)
* **서울시 노인 주거복지시설 데이터**
  + [**https://www.data.go.kr/data/15046494/fileData.do**](https://www.data.go.kr/data/15046494/fileData.do)
* **국토교통부 노인복지시설 지도 API**
  + [**https://www.data.go.kr/data/15056929/openapi.do**](https://www.data.go.kr/data/15056929/openapi.do)
* **유사사이트 : 굿닥**
  + [**https://www.goodoc.co.kr/**](https://www.goodoc.co.kr/)
  + **질병명을 토대로 가까운 병원을 찾아주는 어플리케이션**

# **이모잡 (**[**https://2mojob.com/**](https://2mojob.com/)**)**

* **중장년층을 위한 일자리 매칭 플랫폼. 특히 은퇴 직후인 50대 후반 ~ 60대 초반 연령대를 타겟으로 함. 회원가입시 자신의 직종이나 신상 명세**
* **서울시 어르신 취업지원센터**

**​​​​**[**https://goldenjob.or.kr/job/find-person.asp?keyfield=&keyword=&p=2**](https://goldenjob.or.kr/job/find-person.asp?keyfield=&keyword=&p=2)