

v.0.0.3

최병권

주변 친구 찾기

hello



목차

table of contents

- | | |
|---|-----------|
| 1 | 기획의도 |
| 2 | 구성요소 |
| 3 | Back-End |
| 4 | Front-End |
| 5 | 프론트 화면 |



Part 1

기획 의도



기획 의도

‘주변 친구 찾기 API’는 주변에 친구들이 얼마나 떨어져 있는지 찾기 위해 만든 API이다. 주변 건물을 등록 해놓고 원하는 위치에서 반경 몇 미터 안에 그 건물이 있는지 정확하게 수치로 나타내주는 기능을 가지고 있는데, 주요 기능은 주소를 입력해서 x, y 좌표를 찾아낸 뒤, 주변 주소와 도로명 주소, 행정구역상 주소(동)를 알아 내는 것인데, 내 친구가 어느 건물에 위치해 나와 얼마나 떨어져 있는지 알기 위해서 주변 친구 찾기 API를 만들어 보았다.



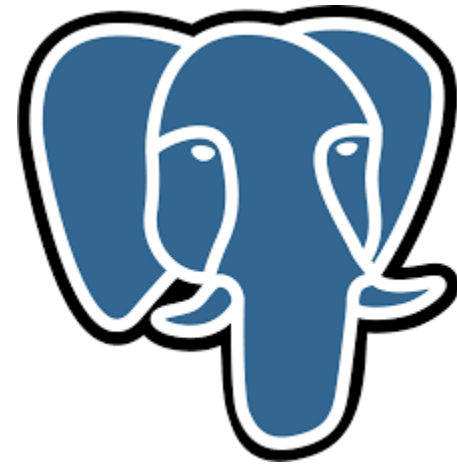
Part 2

구성 요소





- java 17
- Spring Boot



- Postgres

Part 3

Back-End



GeoController

```
@Api(tags = "좌표 관리")
@RestController
@RequiredArgsConstructor
@RequestMapping("/v1/geo")
public class GeoController {

    3 usages
    private final GeoService geoService;

    no usages
    @ApiOperation(value = "좌표를 주소로 변환하기 (좌표를 지번주소, 도로명주소로 변환)")
    @GetMapping("/convert/address")
    public SingleResult<ConvertAddressResponse> getConvertAddress(@RequestParam("x") String x, @RequestParam("y") String y) {
        return ResponseService.getSingleResult(geoService.getConvertAddress(x, y));
    }

    no usages
    @ApiOperation(value = "주소를 좌표로 변환하기 (주소를 좌표로 변환)")
    @GetMapping("/search/address")
    public SingleResult<SearchAddressResponse> getSearchAddress(@RequestParam("searchAddress") String searchAddress) {
        return ResponseService.getSingleResult(geoService.getSearchAddress(searchAddress));
    }

    no usages
    @ApiOperation(value = "행정구역정보 검색하기 (좌표를 지번주소, 행정구역상주소로 변환)")
    @GetMapping("/search/region")
    public SingleResult<SearchRegionResponse> getSearchRegion(@RequestParam("x") String x, @RequestParam("y") String y) {
        return ResponseService.getSingleResult(geoService.getSearchRegion(x, y));
    }
}
```



```
@Getter
@AllArgsConstructor
public enum KakaoUri {
    2 usages
    SEARCH_ADDRESS( apiName: "주소 검색하기", apiSubUri: "/v2/local/search/address.json"),
    2 usages
    SEARCH_REGION( apiName: "행정구역정보 검색하기", apiSubUri: "/v2/local/geo/coord2regioncode.json"),
    2 usages
    CONVERT_ADDRESS( apiName: "주소 변환하기", apiSubUri: "/v2/local/geo/coord2address.json");

    no usages
    private final String apiName;
    no usages
    private final String apiSubUri;
}
```

- 주소 검색, 주소 변환, 행정 구역 정보 검색 키워드를 ENUM에 추가

```

public class RestApi {
    /**
     * api 호출
     * @param httpMethod 매핑 형태.. get, post, update....
     * @param apiUrl api call 할 주소
     * @param restKey rest api 호출 키
     * @param responseModel 반환받을 타입.. 반환받을 그릇 모양
     * @return responseModel 을 통해 알려진 타입을 반환 받는다.
     * @param <T> 반환받을 그릇 모양 */
    3 usages
    public static <T> T callApi(HttpMethod httpMethod, String apiUrl, String restKey, Class<T> responseModel) {
        try {
            URI uri = URI.create(apiUrl); // String으로 받은 url을 URI 객체로 바꿔주기

            RestTemplate restTemplate = new RestTemplate(); // api call 할 전화기 준비

            // 헤더준비
            HttpHeaders headers = new HttpHeaders();
            headers.add( headerName: "Authorization", headerValue: "KakaoAK " + restKey);
            headers.add( headerName: "Accept", MediaType.APPLICATION_JSON_VALUE);
            headers.add( headerName: "Content-Type", headerValue: MediaType.APPLICATION_FORM_URLENCODED_VALUE + ";charset=UTF-8");
            // 헤더준비 끝

            RequestEntity<String> requestEntity = new RequestEntity<>(headers, httpMethod, uri); // api call 하기 위해 필요한 데이터 세팅
            String responseText = restTemplate.exchange(requestEntity, String.class).getBody();
            // api에 필요한 데이터(request) 넘겨주면서 필요한 데이터 바꿔먹기.. 일단 String 으로 바꿔먹는다.

            Gson gson = new Gson(); // json 형태의 string 을 원하는 모델로 바꿔주기 위해 이 기능을 할 수 있는 Gson 이란 놈 한명 불러오기

            return gson.fromJson(responseText, responseModel); // gson 한테 Class<T> responseModel 로 받은..(이 모양으로 바꿔줘) class 모양으로 바꿔달라고 하기
        } catch (Exception e) { // 위에 try 안에 코드 실행하다가 안되면.. 안될때 : api 주소가 잘못되었거나 api 호출 횟수가 소진되었거나 없는키거나..
            e.printStackTrace(); // 에러메세지 로그로 띄워주고
            throw new CMissingDataException(); // advice 를 통해서 커스텀 비상구로 밀어버리기
        }
    }
}

```

- response가 다른 API를 여러 개를 사용하는데
- 이것을 하나의 메서드로 처리하기 위해서 제네릭<T>을 사용하였다.

ConvertAddressItems

```
@Getter
@Setter
public class ConvertAddressDocumentInAddressResponse {
    no usages
    private String address_name;
}
```

ConvertAddressDocumentInAddressResponse

```
@Getter
@Setter
public class ConvertAddressResponse {
    no usages
    private List<ConvertAddressDocumentItem> documents;
}
```

ConvertAddressResponse

```
@Getter
@Setter
public class ConvertAddressDocumentInRoadAddressResponse {
    no usages
    private String address_name;

    no usages
    private String building_name;

    no usages
    private String zone_no;
}
```

ConvertAddressDocumentInAddressResponse

```
@Getter
@Setter
public class ConvertAddressDocumentItem {
    no usages
    private ConvertAddressDocumentInRoadAddressResponse road_address;
    no usages
    private ConvertAddressDocumentInAddressResponse address;
}
```

ConvertAddressDocumentItem

SearchAddressItems

```

@Getter
@Setter
public class SearchAddressDocumentItem {
    no usages
    @ApiModelProperty(notes = "전체 지번 주소 또는 전체 도로명 주소, 입력에 따라 결정됨")
    private String address_name;

    no usages
    @ApiModelProperty(notes = "address_name의 값의 타입(Type)\n" +
        "다음 중 하나:\n" +
        "REGION(지명)\n" +
        "ROAD(도로명)\n" +
        "REGION_ADDR(지번 주소)\n" +
        "ROAD_ADDR(도로명 주소)")
    private String address_type;

    no usages
    @ApiModelProperty(notes = "X 좌표값, 경위도인 경우 경도(longitude)")
    private String x;

    no usages
    @ApiModelProperty(notes = "Y 좌표값, 경위도인 경우 위도(latitude)")
    private String y;
}

```

SearchAddressDocumentItem

```

@Getter
@Setter
public class SearchAddressResponse {
    no usages
    @ApiModelProperty(notes = "결과 아이템들")
    private List<SearchAddressDocumentItem> documents;
}

```

SearchAddressResponse

SearchRegionItems

```
@Getter
@Setter
public class SearchRegionDocumentItem {

    no usages
    @ApiModelProperty(notes = "H(행정동) 또는 B(법정동)")
    private String region_type;

    no usages
    @ApiModelProperty(notes = "지역 1Depth, 시도 단위")
    private String region_1depth_name;

    no usages
    @ApiModelProperty(notes = "지역 2Depth, 구 단위")
    private String region_2depth_name;

    no usages
    @ApiModelProperty(notes = "지역 3Depth, 동 단위")
    private String region_3depth_name;

    no usages
    @ApiModelProperty(notes = "지역 4Depth, region_type이 법정동이며, 리 영역인 경우만 존재")
    private String region_4depth_name;
}
```

SearchRegionDocumentItem


```
@Getter
@Setter
public class SearchRegionResponse {

    no usages
    private List<SearchRegionDocumentItem> documents;
}
```

SearchRegionResponse

GeoService (1)

원래 Value 형태



```
@Service
public class GeoService {
    3 usages
    @Value("https://dapi.kakao.com")
    String KAKAO_API_DOMAIN;

    3 usages
    @Value("25cc7f2338ffac4b7d240103218df07f")
    String KAKAO_API_REST_KEY;

    1 usage
    public SearchAddressResponse getSearchAddress(String searchAddress) {
        String apiFullUri = KAKAO_API_DOMAIN + KakaoUri.SEARCH_ADDRESS.getApiSubUri();
        String queryString = "?query=" + URLEncoder.encode(searchAddress, StandardCharsets.UTF_8);

        String resultUrl = apiFullUri + queryString;

        return RestApi.callApi(HttpMethod.GET, resultUrl, KAKAO_API_REST_KEY, SearchAddressResponse.class);
    }
}
```

```
@Service
public class GeoService {
    3 usages
    @Value("${kakao.api.domain}")
    String KAKAO_API_DOMAIN;

    3 usages
    @Value("${kakao.api.rest-key}")
    String KAKAO_API_REST_KEY;

    1 usage
    public SearchAddressResponse getSearchAddress(String searchAddress) {
        String apiFullUri = KAKAO_API_DOMAIN + KakaoUri.SEARCH_ADDRESS.getApiSubUri();
        String queryString = "?query=" + URLEncoder.encode(searchAddress, StandardCharsets.UTF_8);

        String resultUrl = apiFullUri + queryString;

        return RestApi.callApi(HttpMethod.GET, resultUrl, KAKAO_API_REST_KEY, SearchAddressResponse.class);
    }
}
```

GeoService (2)

```
public SearchRegionResponse getSearchRegion(String x, String y) {  
    String apiFullUri = KAKAO_API_DOMAIN + KakaoUri.SEARCH_REGION.getApiSubUri();  
    String queryString = "?x=" + x + "&y=" + y;  
  
    String resultUrl = apiFullUri + queryString;  
  
    return RestApi.callApi(HttpMethod.GET, resultUrl, KAKAO_API_REST_KEY, SearchRegionResponse.class);  
}
```

1 usage

```
public ConvertAddressResponse getConvertAddress(String x, String y) {  
    String apiFullUri = KAKAO_API_DOMAIN + KakaoUri.CONVERT_ADDRESS.getApiSubUri();  
  
    String coord = "WGS84";  
    String queryString = "?x=" + x + "&y=" + y + "&input_coord=" + coord;  
  
    String resultUrl = apiFullUri + queryString;  
  
    return RestApi.callApi(HttpMethod.GET, resultUrl, KAKAO_API_REST_KEY, ConvertAddressResponse.class);  
}
```

MembersController

```
@Api(tags = "주변 친구 관리")
@RestController
@RequiredArgsConstructor
@RequestMapping("/v1/member")
public class MembersController {
    2 usages
    private final MembersService membersService;

    no usages
    @ApiOperation(value = "주변 친구 등록")
    @PostMapping("/friend")
    public CommonResult setMembers(@RequestBody @Valid MembersCreateRequest createRequest) {
        membersService.setMembers(createRequest);
        return ResponseService.getSuccessResult();
    }

    no usages
    @ApiOperation(value = "내 주변 친구 찾기")
    @PostMapping("/search/friends") // 기능은 R(Get)이지만 body 를 쓰기 위해 PostMapping 사용
    public ListResult<NearFriendItem> getFriends(@RequestBody @Valid NearFriendSearchRequest searchRequest) {
        return ResponseService.getListResult(membersService.getNearFriends
            (searchRequest.getPosX(), searchRequest.getPosY(), searchRequest.getDistance()), isSuccess: true);
    }
}
```


Members

```

@Entity
@Getter
@NoArgsConstructor(access = AccessLevel.PROTECTED)
public class Members {

    no usages
    @ApiModelProperty(notes = "시퀀스")
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    1 usage
    @ApiModelProperty(notes = "별명")
    @Column(nullable = false, length = 20)
    private String nickname;

    1 usage
    @ApiModelProperty(notes = "취미")
    @Column(nullable = false, length = 50)
    private String hobby;

    1 usage
    @ApiModelProperty(notes = "성별")
    @Column(nullable = false, length = 10)
    @Enumerated(EnumType.STRING)
    private Gender gender;

    1 usage
    @ApiModelProperty(notes = "x 좌표")
    @Column(nullable = false)
    private Double posX;

    1 usage
    @ApiModelProperty(notes = "y 좌표")
    @Column(nullable = false)
    private Double posY;

```

```

private Members(Builder builder) {
    this.nickname = builder.nickname;
    this.hobby = builder.hobby;
    this.gender = builder.gender;
    this.posX = builder.posX;
    this.posY = builder.posY;
}

2 usages
public static class Builder implements CommonModelBuilder<Members> {

    2 usages
    private final String nickname;

    2 usages
    private final String hobby;

    2 usages
    private final Gender gender;

    2 usages
    private final Double posX;

    2 usages
    private final Double posY;

    1 usage
    public Builder(MembersCreateRequest createRequest) {
        this.nickname = createRequest.getNickname();
        this.hobby = createRequest.getHobby();
        this.gender = createRequest.getGender();
        this.posX = createRequest.getPosX();
        this.posY = createRequest.getPosY();
    }

    2 usages
    @Override
    public Members build() { return new Members( builder: this); }

}

```

Gender

```
@Getter
@AllArgsConstructor
public enum Gender {
    no usages
    MAN( name: "남자"),
    1 usage
    WOMAN( name: "여자");

    no usages
    private final String name;
}
```

성별 정보를 ENUM에 추가

NearFriendItem

```

@Getter
@NoArgsConstructor(access = AccessLevel.PROTECTED)
public class NearFriendItem {

    1 usage
    @ApiModelProperty(notes = "닉네임")
    private String nickname;

    1 usage
    @ApiModelProperty(notes = "취미")
    private String hobby;

    1 usage
    @ApiModelProperty(notes = "성별(eunm값)")
    private String gender;

    1 usage
    @ApiModelProperty(notes = "성별 한글명")
    private String genderName;

    1 usage
    @ApiModelProperty(notes = "나와의 거리")
    private Double distanceM;

    1 usage
    private NearFriendItem(Builder builder) {
        this.nickname = builder.nickname;
        this.hobby = builder.hobby;
        this.gender = builder.gender;
        this.genderName = builder.genderName;
        this.distanceM = builder.distanceM;
    }
}

```

```

private NearFriendItem(Builder builder) {
    this.nickname = builder.nickname;
    this.hobby = builder.hobby;
    this.gender = builder.gender;
    this.genderName = builder.genderName;
    this.distanceM = builder.distanceM;
}

2 usages
public static class Builder implements CommonModelBuilder<NearFriendItem> {

    2 usages
    private final String nickname;

    2 usages
    private final String hobby;

    2 usages
    private final String gender;

    2 usages
    private final String genderName;

    2 usages
    private final Double distanceM;

    1 usage
    public Builder(String nickname, String hobby, String gender, Double distanceM) {
        this.nickname = nickname;
        this.hobby = hobby;
        this.gender = gender;
        this.genderName = Gender.valueOf(gender).getName();
        this.distanceM = distanceM;
    }

    2 usages
    @Override
    public NearFriendItem build() { return new NearFriendItem( builder, this); }
}
}

```

MembersCreateRequest

```
@Getter
@Setter
public class MembersCreateRequest {

    no usages
    @ApiModelProperty(notes = "별명")
    @NotNull
    @Length(min = 2, max = 20)
    private String nickname;

    no usages
    @ApiModelProperty(notes = "취미")
    @NotNull
    @Length(min = 2, max = 50)
    private String hobby;

    no usages
    @ApiModelProperty(notes = "성별")
    @NotNull
    @Enumerated(EnumType.STRING)
    private Gender gender;

    no usages
    @ApiModelProperty(notes = "x 좌표")
    @NotNull
    private Double posX;

    no usages
    @ApiModelProperty(notes = "y 좌표")
    @NotNull
    private Double posY;
}
```


NearFriendSearchRequest

```
@Getter
@Setter
public class NearFriendSearchRequest {

    no usages
    @ApiModelProperty(notes = "내 위치 x 좌표")
    @NotNull
    private Double posX;

    no usages
    @ApiModelProperty(notes = "내 위치 y 좌표")
    @NotNull
    private Double posY;

    no usages
    @ApiModelProperty(notes = "최대 거리 설정(반경)")
    @NotNull
    private Integer distance;
}
```

MembersRepository

```
public interface MembersRepository extends JpaRepository<Members, Long> {  
    1 usage  
    @Query(  
        value = "select * from members where earth_distance(ll_to_earth(posy, posx), ll_to_earth(:pointY, :pointX)) <= :searchDistance",  
        nativeQuery = true  
    )  
    List<Members> findAllByNearFriends(@Param("pointY") double pointY, @Param("pointX") double pointX, @Param("searchDistance") double searchDistance);  
    1 usage  
    @Query(  
        value = "select * from member where earth_distance(ll_to_earth(posy, posx), ll_to_earth(:pointY, :pointX)) <= :searchDistance and member.gender = :gender",  
        nativeQuery = true  
    )  
    List<Members> findAllByNearFriendsByGender(@Param("pointY") double pointY, @Param("pointX") double pointX, @Param("searchDistance") double searchDistance, @Param("gender") String gender);  
}
```

주변 친구를 찾기 위해 필요한 x, y 좌표와 정렬 순서

MembersService

```

@Service
@RequiredArgsConstructor
public class MembersService {
    1 usage
    private final MembersRepository membersRepository;

    1 usage
    @PersistenceContext
    EntityManager entityManager;

    1 usage
    public void setMembers(MembersCreateRequest createRequest) {
        Members members = new Members.Builder(createRequest).build();
        membersRepository.save(members);
    }

    /**
     * 근처 친구 리스트를 가져옴
     * @param posX ex) 126.xxx
     * @param posY ex) 37.xxx
     * @param distance 몇 km의 친구 리스트를 가져올지 결정 ex) 1,2,...
     * @return
     */
    1 usage
    public ListResult<NearFriendItem> getNearFriends(double posX, double posY, int distance) {
        double distanceResult = distance * 1; // 미터로 변환 ( * 1 = 입력 값[숫자] * 1m)

        String queryString = "select * from public.get_near_friends(" + posX + "," + posY + "," + distanceResult + ")";
        Query nativeQuery = entityManager.createNativeQuery(queryString);
        List<Object[]> resultList = nativeQuery.getResultList(); // 실행해서 가져오기

        List<NearFriendItem> result = new LinkedList<>();
        for(Object[] resultItem : resultList) {
            result.add(
                new NearFriendItem.Builder(
                    resultItem[0].toString(),
                    resultItem[1].toString(),
                    resultItem[2].toString(),
                    Double.parseDouble(resultItem[3].toString()))
                .build()
            );
        }
        return ListConvertService.settingResult(result);
    }
}

```

Procedur e

```
CREATE OR REPLACE FUNCTION public.get_near_friends
    (positionX DOUBLE PRECISION, positionY DOUBLE PRECISION, distance DOUBLE PRECISION)
RETURNS TABLE
(
    nickname varchar,
    hobby varchar,
    gender varchar,
    distance_m double precision
)
as
$$
DECLARE
    v_record RECORD;
BEGIN
    for v_record in (
        select
            members.nickname, members.hobby, members.gender, earth_distance(ll_to_earth(members.posy, members.posx),
                ll_to_earth(positionY, positionX)) as distance_m
        from members
        where earth_distance(ll_to_earth(members.posy, members.posx), ll_to_earth(positionY, positionX)) <= distance
        order by distance_m asc
    )
    loop
        nickname := v_record.nickname;
        hobby := v_record.hobby;
        gender := v_record.gender;
        distance_m := v_record.distance_m;
        return next;
    end loop;
END;
$$
LANGUAGE plpgsql
```


Procedur e

```
select
  nickname,
  gender,
  hobby,
  posx,
  posy,
  earth_distance(ll_to_earth(posy, posx), ll_to_earth(37.3059938196752, 126.83709520524)) as distance_m
from members
where
  earth_distance(ll_to_earth(posy, posx), ll_to_earth(37.3059938196752, 126.83709520524)) <= 2000
order by distance_m asc;
```

1. Members Entity에서 nickname, gender, hobby, meter로 변환할 pos x, pos y (위도, 경도 -> 좌표)를 가져온다.
2. pos x, pos y (위도, 경도 -> 좌표) 에서 2000m (2km) 반경 내에 등록된 모든 친구들을 호출한다.

Part 4

Front-End



Template

```

<template>
  <div class = searchAddress>
    <el-row>
      <div class = "addressType">
        <h1>* 지번주소 또는 도로명주소를 입력해주세요.</h1>
      </div>
      <div class = "address">
        <el-col :span="18">
          <el-input placeholder="내 주소 검색" v-model="searchAddress"></el-input>
        </el-col>
        <el-col :span="6">
          <el-button type="primary" @click="getAddress()">검색</el-button> <!-- 검색버튼 누르면 getAddress() 함수 실행시키기 -->
        </el-col>
      </div>
    </el-row>

    <div class="searchBox">
      <div v-if="isAddressResultBoxView"> <!-- isAddressResultBoxView 값이 true면.. true면 보여주기 false면 감추기 -->
        <div class="inbox" v-for="(item, index) in searchAddressResultList">
          {{ item.address_type === 'ROAD_ADDR' ? '- 도로명 : ' : '- 지번 : ' }} {{ item.address_name }}
          <el-button type="primary" @click="choiceResultPosition(item.x, item.y)">선택</el-button>
        </div>
      </div>
    </div>

    <div>
      <li class = nearFriend v-for="(item, index) in nearFriendList">
        나와의 거리 : {{ Math.round(item.distanceM) }}m, 성별 : {{ item.genderName }} , 취미 : {{ item.hobby }} , 닉네임 : {{ item.nickname }}
      </li>
      <div class="map">
        
      </div>
    </div>
  </div>
</template>

```

```

<script>
no usages
export default {
  data() {
    return {
      searchAddress: '', // 주소 검색 키워드 저장 할 변수
      searchAddressResultList: [], // 주소 검색 결과 리스트 저장 할 변수
      isAddressResultBoxView: false, // 주소 검색 결과 박스 보일지 말지 결정하는 변수
      choicePositionX: null, // 주소 선택 시 좌표 x 값 저장 할 변수
      choicePositionY: null, // 주소 선택 시 좌표 y 값 저장 할 변수
      nearFriendList: [] // 근처 친구 결과 담을 리스트 저장 할 변수
    }
  },
  methods: {
    getAddress() {
      if (this.searchAddress.length >= 1) { // 만약 searchAddress 변수에 글자가 한글자라도 있으면.
        let payload = { // 페이로드 <- 페이로드라는건 그냥 이름일 뿐. request에 쓰이는 값을 그냥 페이로드라고 통상적으로 부름.
          params: { // params 라는 임의의 공간을 만들고
            searchAddress: this.searchAddress // searchAddress라는 칸을 또 만들어 준 후 위에 데이터에서 값을 가져와 넣어준다.
          }
        }

        this.$store.commit(this.$customLoadingConstants.FETCH_LOADING_SHOW, { payload: true }) // api call 해야하니 그동안 사람이 이상한지 못하게 화면 막아주고
        this.$store.dispatch(this.$apiKakaoConstants.DO_SEARCH_ADDRESS, payload) // store api-kakao 모듈에 정의된 액션을 실행시킨다. (api 호출하는게 액션)
        .then(res => { // 만약에 api 호출에 성공했다면
          // res.data 까지가 응답 결과이고 그 응답결과에서 data.documents 부분을 뽑아서 searchAddressResultList에 넣어준다. -> swagger response 참조
          this.searchAddressResultList = res.data.data.documents
          this.isAddressResultBoxView = true // 주소 검색 결과 박스 보여주기
          this.$store.commit(this.$customLoadingConstants.FETCH_LOADING_SHOW, { payload: false }) // 다 끝났으니 화면 막았던거 다시 풀어주기
        })
        .catch((err) => { // 만약에 api 호출에 실패했다면
          this.$toast.error(err.response.data.msg) // 실패 메시지 토스트팝업으로 띄워주고
          this.isAddressResultBoxView = false // 주소 검색 결과 박스 숨기기
          this.$store.commit(this.$customLoadingConstants.FETCH_LOADING_SHOW, { payload: false }) // 다 끝났으니 화면 막았던거 다시 풀어주기
        })
      }
    }
  }
}

```

```

    },
    choiceResultPosition(posX, posY) {
      this.choicePositionX = posX
      this.choicePositionY = posY
      this.isAddressResultBoxView = false // 주소 검색 결과 박스 숨기기
      this.getNearFriends()
    },
    getNearFriends() {
      let payload = {
        posX: Number(this.choicePositionX),
        posY: Number(this.choicePositionY),
        distance: 3 // 반경 3km 이내의 친구 불러오기.. 강제로 3km 설정함.
      }

      this.$store.commit(this.$customLoadingConstants.FETCH_LOADING_SHOW, { payload: true }) // api call 해야하니 그동안 사람이 이상한지 못하게 화면 막아주고
      this.$store.dispatch(this.$apiMemberConstants.DO_NEAR_FRIENDS, payload)
        .then(res => { // 만약에 api 호출에 성공했다면
          this.nearFriendList = res.data.list
          this.$store.commit(this.$customLoadingConstants.FETCH_LOADING_SHOW, { payload: false }) // 다 끝났으니 화면 막았던거 다시 풀어주기
        })
        .catch(err => { // 만약에 api 호출에 실패했다면
          this.$toast.error(err.response.data.msg) // 실패 메시지 토스트팝업으로 띄워주고
          this.$store.commit(this.$customLoadingConstants.FETCH_LOADING_SHOW, { payload: false }) // 다 끝났으니 화면 막았던거 다시 풀어주기
        })
    }
  },
}

```

Part 5

프론트 화면



Front Image (1)

* 지번주소 또는 도로명주소를 입력해주세요.

내 주소 검색

검색



Front Image (2)

* 지번주소 또는 도로명주소를 입력해주세요.

고잔동 780 ← 1. 검색어 입력

검색

- 지번 : 경기 안산시 단원구 고잔동 780

선택

 ← 2. 내 위치 조회

- 지번 : 인천 남동구 고잔동 780

선택



Front Image (3)

* 지번주소 또는 도로명주소를 입력해주세요.

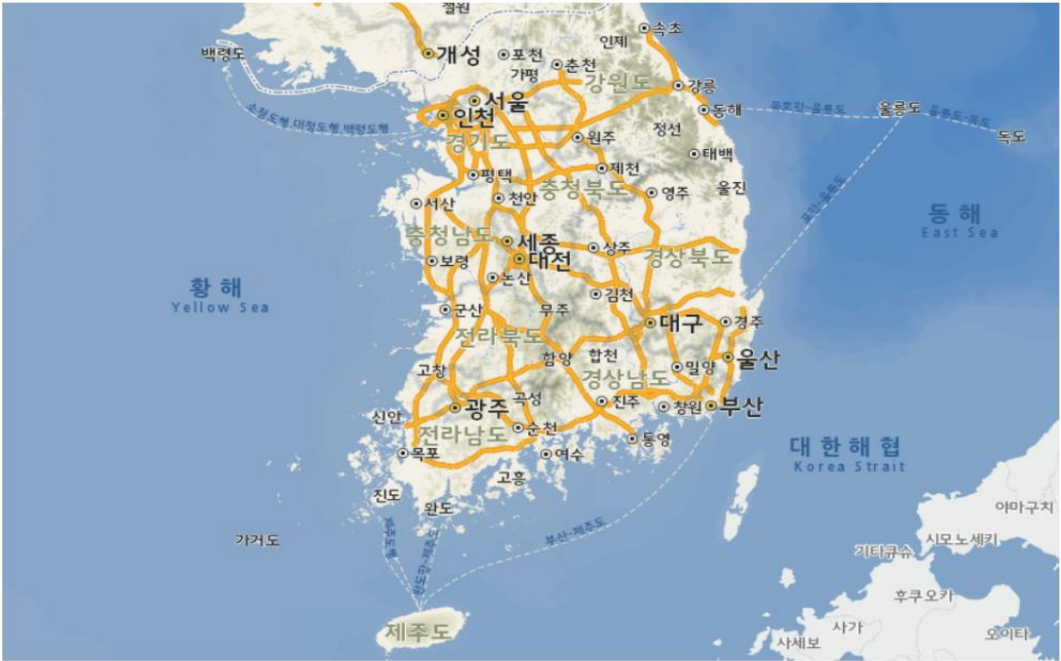
고잔동 780

검색

- 나와의 거리 : 303m, 성별 : 여자 , 취미 : 집에서 티비보기 , 닉네임 : 집순이
- 나와의 거리 : 1259m, 성별 : 남자 , 취미 : 햄버거 먹기 , 닉네임 : 햄돌이
- 나와의 거리 : 1356m, 성별 : 남자 , 취미 : 백화점 쇼핑하기 , 닉네임 : 백화점돌이
- 나와의 거리 : 1403m, 성별 : 남자 , 취미 : 영화보기 , 닉네임 : 영화보기
- 나와의 거리 : 1423m, 성별 : 남자 , 취미 : 치과 치료하기 , 닉네임 : 치과의사임
- 나와의 거리 : 1591m, 성별 : 남자 , 취미 : 일본제품 불매하기 , 닉네임 : 롯데불매하자

← 2. 나와 떨어진 거리 및 기타 정보

↑
1. 검색 버튼 클릭





The End