

# 동네 친구 찾기

Back-End : <https://gitlab.com/rkddbwns123/nearbyfriendssearch>

Front-End : [https://gitlab.com/rkddbwns123/near\\_friends\\_search](https://gitlab.com/rkddbwns123/near_friends_search)

제작기간: 2023-03-06 ~ 2023.03.10  
버전:v0.0.1

강유준



# 목차 table of contents

---

- 1 기획 의도
  - 2 소프트웨어 구성
  - 3 Back-End
  - 4 Front-End
- 



III  
ADVENTURES...  
AND LESSONS  
LEARNED

Part 1

# 기획 의도



## 기획 의도

GPS가 실생활에서 굉장히 많이 사용하는 기술 중 하나라는 건 경험적으로도 알고 있는 정보다.


대부분의 자동차에는 내비게이션이 달려 있고, 사람들이 항상 쥐고 다니는 스마트폰에도 GPS와 그걸 이용한 수많은 앱들이 있기 때문이다.

지도 앱, 당근 마켓 그리고 만남 주선 앱 등등.

자주 쓰이는 기술을 경험하고 다른 Api를 호출해 사용하는 방법도 익히고자 기획하게 되었다.

/ Kakao에서 제공하는 로컬 RestApi를 이용해 등록된 회원들의 주소 좌표로 특정 거리의 회원들을 찾는 Api 기획





Part 2

# 소프트웨어 구성



- 
- Java
  - Spring Boot
  - JPA
  - Postgres / Procedure
  - Rest Api



- 
- Javascript
  - vue.js
  - Nuxt

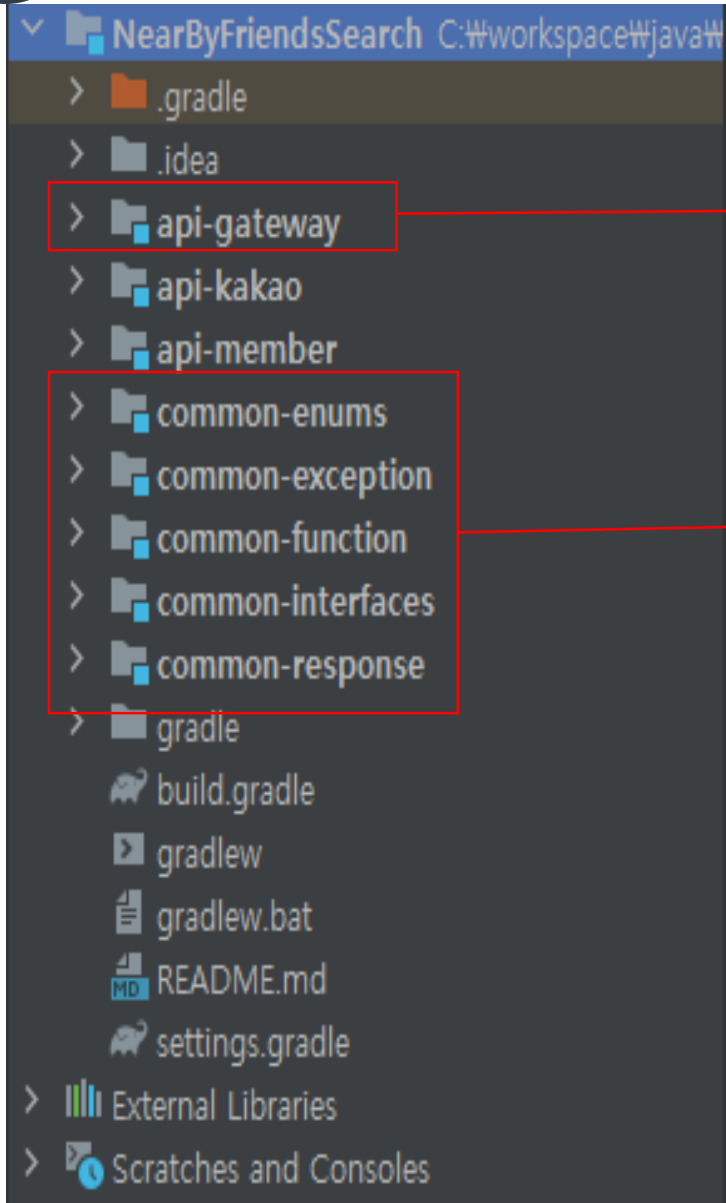




Part 3

# Back-End





여러 API 모듈을 관리하기 위해 Api 게이트웨이 생성

프로젝트에서 공통적으로 사용하는 모듈:  
사용할 API의 Dependencies로 연결시켜 사용



## Kakao-api / Service

```

@Service
public class GeoService {
    @Value("${kakao.api.domain}")
    String KAKAO_API_DOMAIN;
    @Value("${kakao.api.rest-key}")
    String KAKAO_API_REST_KEY;

    public SearchAddressResponse getSearchAddress(String addressValue) {
        String apiFullUri = KAKAO_API_DOMAIN + KakaoUri.SEARCH_ADDRESS.getApiSubUri();

        String queryString = "?query=" + URLEncoder.encode(addressValue, StandardCharsets.UTF_8);

        String resultUrl = apiFullUri + queryString;

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

    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);
    }

    public RegionToAddressResponse getRegionToAddress(String x, String y) {
        String apiFullUri = KAKAO_API_DOMAIN + KakaoUri.REGION_TO_ADDRESS.getApiSubUri();

        String queryString = "?x=" + x + "&y=" + y + "&input_coord=WGS84";

        String resultUrl = apiFullUri + queryString;

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

```

주소를 받아서 좌표로 변환 시키는 메서드

좌표로 행정구역 찾는 메서드

좌표를 받아서 주소로 변환 시키는 메서드

@Value를 이용해 application.yml 파일에 설정해둔 Bean 객체를 서비스에서 사용할 수 있게 호출.

```
SEARCH_ADDRESS( apiName: "주소 검색하기", apiSubUri: "/v2/local/search/address.json");
```

## ENUM

apiFulluri = 카카오 로컬 api의 공통적인 도메인(bean) + Enum으로 설정해둔 기능 별 uri

queryString = '?query=' + 입력한 주소를 UTF-8로 인코딩

제네릭을 사용 (callApi) / Response와 URL만 다르게 하고 공통적인 코드를 제네릭으로 처리

## Kakao-api / Generic

```
public class RestApi {  
    public static <T> T callApi(HttpMethod httpMethod, String apiUrl, String restKey, Class<T>  
    responseModel) {  
        try {  
            URI uri = URI.create(apiUrl);    서비스에서 받은 주소로 URI 생성  
  
            RestTemplate restTemplate = new RestTemplate();  
  
            HttpHeaders headers = new HttpHeaders();  
            headers.add("Authorization", "KakaoAK " + restKey);  
            headers.add("Accept", MediaType.APPLICATION_JSON_VALUE);  
            headers.add("Content-Type", MediaType.APPLICATION_FORM_URLENCODED_VALUE +  
            ";charset=UTF-8");  
  
            RequestEntity<String> requestEntity = new RequestEntity<>(headers, httpMethod, uri);  
            String responseText = restTemplate.exchange(requestEntity, String.class).getBody();  
  
            Gson gson = new Gson();    Gson으로 Json형태의 데이터를 객체로 변환  
  
            return gson.fromJson(responseText, responseModel);  
        } catch (Exception e) {  
            e.printStackTrace();  
            throw new CMissingDataException();  
        }    실패할 경우 Exception 처리  
    }  
}
```

RestTemplate을 이용해 다른 RestApi를 호출

Headers에 값을 추가 (HttpHeaders)

값을 추가한 headers와 httpMethod의 종류, 위에서 만든 uri를 requestEntity에 넣어준다.

restTemplate.exchange로 requestEntity를 넘겨주면서 String 형태의 데이터를 받아온다.



## Kakao-api / Controller

```
public class GeoController {  
    private final GeoService geoService;  
  
    @ApiOperation(value = "주소로 좌표 변환하기")  
    @GetMapping("/search/address")  
    public SingleResult<SearchAddressResponse> getSearchAddress(@RequestParam("searchAddress")  
String searchAddress) {  
        return ResponseService.getSingleResult(geoService.getSearchAddress(searchAddress));  
    }  
    @ApiOperation(value = "좌표로 행정 구역 받기")  
    @GetMapping("/search/region")  
    public SingleResult<SearchRegionResponse> getSearchRegion(@RequestParam("x") String x,  
@RequestParam("y") String y) {  
        return ResponseService.getSingleResult(geoService.getSearchRegion(x, y));  
    }  
    @ApiOperation(value = "좌표로 주소 변환하기")  
    @GetMapping("/region-to/address")  
    public SingleResult<RegionToAddressResponse> getRegionToAddress(@RequestParam("x") String x,  
@RequestParam("y") String y) {  
        return ResponseService.getSingleResult(geoService.getRegionToAddress(x, y));  
    }  
}
```

@RequestParam 어노테이션을 이용해

주소 혹은 좌표 값을 요청 받으면 Service 실행

@ApiOperation 어노테이션으로 Swagger에서  
가독성을 높임

```

create function get_near_friends(positionx double precision, positiony double precision, distance double precision)
returns TABLE(nickname character varying, hobby character varying, gender character varying, distance_m double precision)
language plpgsql
as
$$
DECLARE
    v_record RECORD;
BEGIN
    for v_record in (
        select
            member_info.nick_name, member_info.hobby, member_info.gender, earth_distance(ll_to_earth(member_info.posy, member_info.posx), ll_to_earth(positionY, positionX)) as distance_m
        from member_info
        where earth_distance(ll_to_earth(member_info.posy, member_info.posx), ll_to_earth(positionY, positionX)) <= distance
        order by distance_m asc
    )
    loop
        nickname := v_record.nick_name;
        hobby := v_record.hobby;
        gender := v_record.gender;
        distance_m := v_record.distance_m;
        return next;
    end loop;
END;
$$;

alter function get_near_friends(double precision, double precision, double precision) owner to near_by_friend_member;

```

값을 받을 테이블 지정

For 문을 이용해 DB내의 조건에 부합하는 데이터를 가져와 v-record에 넣는다.

v-record에 저장된 데이터들을 각 테이블에 저장.

## Procedure



## Table 생성

```

@Entity
@Getter
@NoArgsConstructor(access = AccessLevel.PROTECTED)
public class MemberInfo {
    @ApiModelProperty(notes = "시퀀스")
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @ApiModelProperty(notes = "닉네임")
    @Column(nullable = false, length = 20)
    private String nickName;

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

    @ApiModelProperty(notes = "성별")
    @Enumerated(value = EnumType.STRING)
    @Column(nullable = false)
    private Gender gender;

    @ApiModelProperty(notes = "x 좌표값")
    @Column(nullable = false)
    private Double posX;

    @ApiModelProperty(notes = "y 좌표값")
    @Column(nullable = false)
    private Double posY;
}

```

성별 enum

## Builder 패턴

```

private MemberInfo(MemberInfoBuilder builder) {
    this.nickName = builder.nickName;
    this.hobby = builder.hobby;
    this.gender = builder.gender;
    this.posX = builder.posX;
    this.posY = builder.posY;
}

public static class MemberInfoBuilder implements CommonModelBuilder<MemberInfo> {

    private final String nickName;
    private final String hobby;
    private final Gender gender;
    private final Double posX;
    private final Double posY;

    public MemberInfoBuilder(MemberInfoRequest request) {
        this.nickName = request.getNickName();
        this.hobby = request.getHobby();
        this.gender = request.getGender();
        this.posX = request.getPosX();
        this.posY = request.getPosY();
    }

    @Override
    public MemberInfo build() {
        return new MemberInfo(this);
    }
}
}

```

## Member-api / Service

```

@Service
@RequiredArgsConstructor
public class MemberInfoService {

    @PersistenceContext
    EntityManager entityManager;

    private final MemberInfoRepository memberInfoRepository;

    public void setMemberInfo(MemberInfoRequest request) {
        MemberInfo addData = new MemberInfo.MemberInfoBuilder(request).build();

        memberInfoRepository.save(addData);
    }

    public ListResult<NearFriendsItem> getNearFriends(double posY, double posX, int distance) {
        double distanceResult = distance * 1000; // Int 값으로 입력 받는 거리를 미터(m)로 환산

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

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

        return ListConvertService.settingResult(result);
    }
}

```

회원 등록을 위한 메소드  
Request를 받아 빌더 패턴을 이용해 데이터를  
Repository에 저장.

특정 거리에 있는 주변 회원을 찾는 메소드  
자신의 위치(posX, posY)와 지정할 거리를  
받는다.

프로시저 쿼리 값을 String 형태로 만든 뒤  
entityManager를 통해 쿼리 객체로 변환.

nativeQuery를 통해 쿼리를 실행시키고 리스트  
를 가져온다.  
가져온 Object를 List의 index값을 이용해 순  
서대로 넣어주고 String으로 변환시킨다.  
빌더 패턴을 이용해 반환한다.



## Member-api / Controller

```
@Api(tags = "회원 관리")
@RestController
@RequiredArgsConstructor
@RequestMapping("/v1/member-info")
public class MemberInfoController {
    private final MemberInfoService memberInfoService;

    @ApiOperation(value = "회원 등록") RequestBody로 요청 값을 받아오고 Valid로
    @PostMapping("/new") 들어오는 값에 조건을 달아 조건에 부합하는 지 확인
    public CommonResult setMemberInfo(@RequestBody @Valid MemberInfoRequest request) {
        memberInfoService.setMemberInfo(request);

        return ResponseService.getSuccessResult();
    }

    @ApiOperation(value = "주변 친구 찾기")
    @PostMapping("/distance")
    public ListResult<NearFriendsItem> getNearFriends(@RequestBody @Valid NearFriendsSearchRequest request) {
        return ResponseService.getListResult(memberInfoService.getNearFriends(request.getPosY(),
        request.getPosX(), request.getDistance()), true);
    }
}
```

Body가 필요하기 때문에 Read 기능이지만 편의상 PostMapping을 사용.



Part 4

# Front-End

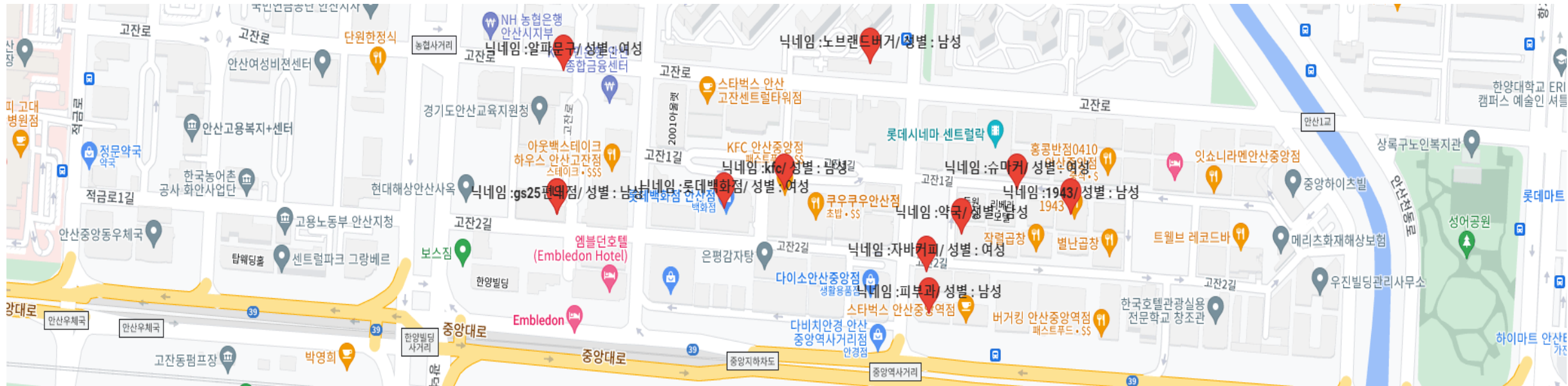
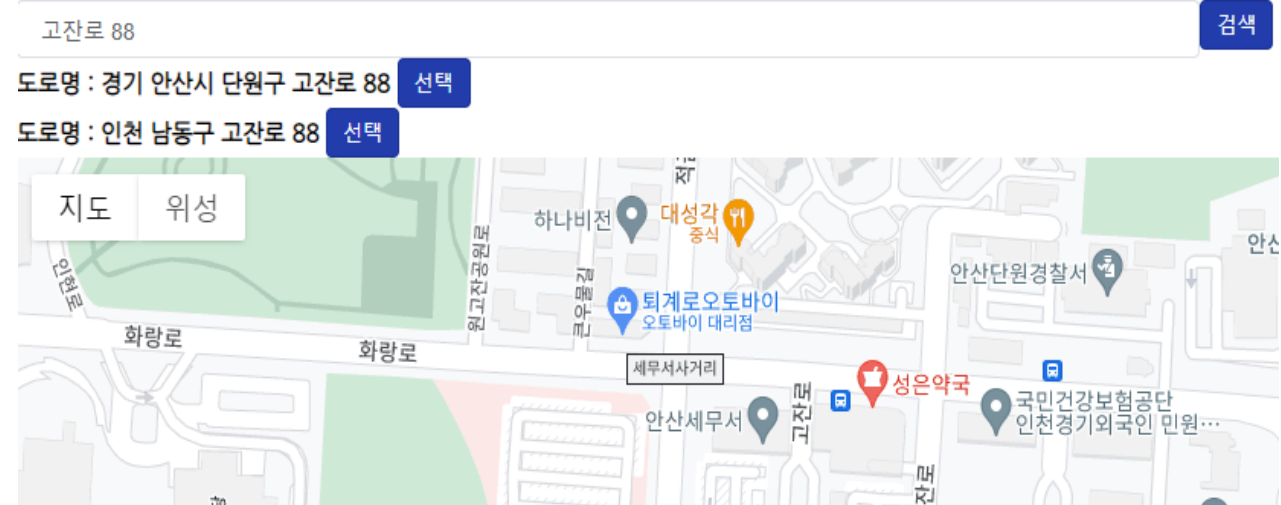




1



2



3

## Template

```

<template>
  <div>
    <el-row>
      <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>
      </el-col>
    </el-row>

    <div v-if="isAddressResultBoxView"> <!-- isAddressResultBoxView
      <div class="address-font" 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 style="width:2000px; height:1050px" id="map"></div>
</div>
</template>

```

구글 지도 api 사용

내 주소 검색 (찾아올 정보의  
기준 좌표)

주소 검색이 성공 했을 때  
(true값)

주소 검색 시 데이터 베이스에  
저장된 좌표에 가져온 정보를  
띄운다.

```
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 답을 리스트 저장 할 변수    map: null,  
    mapCenter: { lat: 37.317957, lng: 126.832261 }, //lat = y, lng = x  
    bands: {  
      lat: null, // 구글 지도 이용을 위한 데이터  
      lng: null, // mapCenter = 처음 웹 사이트 들어갔을  
    }, // 때 기준 좌표. 고정 값으로 설정  
  },  
}
```



```
methods: {
```

```
  getAddress() {
    if (this.searchAddress.length >= 1) {
      let payload = {
        params: {
          searchAddress: this.searchAddress
        }
      }
    }
  }
```

```
    this.$store.commit(this.$customLoadingConstants.FETCH_LOADING_SHOW, true)
    this.$store.dispatch(this.$apiKakaoConstants.DO_SEARCH_ADDRESS, payload)
      .then((res) => {
        this.searchAddressResultList = res.data.data.documents
        this.isAddressResultBoxView = true // 주소 검색 결과 박스 보여주기
        this.$store.commit(this.$customLoadingConstants.FETCH_LOADING_SHOW, false)
      })
      .catch((err) => {
        this.$toast.error(err.response.data.msg)
        this.isAddressResultBoxView = false
        this.$store.commit(this.$customLoadingConstants.FETCH_LOADING_SHOW, false)
      })
  },
},
```

searchAddress(검색 값)이 1글자 이상 이  
라면 searchAddress에 그 데이터를 넣어준  
다.

주소 검색( 주소로 좌표 변환 )Api 연동

.then / 성공 했을 시

1. Api의 documents 결과 값을 가져오고  
리스트에 담는다.
2. 주소 검색 결과 박스를 보여준다.
3. 로딩 제거

.catch / 실패 했을 시

1. 에러 메시지 토스트를 띄운다.
2. 주소 검색 결과 박스를 띄우지 않는다.
- 3.로딩 제거

```
getNearFriends() {
```

```
  let payload = {
    posX: Number(this.choicePositionX),
    posY: Number(this.choicePositionY),
    distance: 3
  }
```

posX, posY에 x, y좌표 값  
저장,  
Distance는 주변 거리(km)  
편의를 위해 3으로 고정

```
  this.$store.commit(this.$customLoadingConstants.FETCH_LOADING_SHOW, true)
  this.$store.dispatch(this.$apiMemberConstants.DO_NEAR_FRIENDS, payload)
```

```
  .then(res => {
    this.nearFriendList = res.data.list
    res.data.list.forEach(item => {
```

```
      let tempPosition = { lat: item.posY, lng: item.posX }
      this.setMarker(tempPosition, '닉네임 : ' + item.nickname + '/ 성별 : ' +
```

```
      item.genderName )
```

```
    })
```

```
    this.$store.commit(this.$customLoadingConstants.FETCH_LOADING_SHOW, false)
```

```
  })
```

```
  .catch(err => {
```

```
    this.$toast.error(err.response.data.msg)
```

```
    this.$store.commit(this.$customLoadingConstants.FETCH_LOADING_SHOW, false)
```

```
  },
```

```
choiceResultPosition(posX, posY) {
  this.choicePositionX = posX
  this.choicePositionY = posY
  this.isAddressResultBoxView = false
  this.getNearFriends()
},
```

선택한 주소의 X, Y값을 저장한 뒤 getNearFriends 메  
소드 실행

가져온 위치 정보에 구글 마커 표시 및 간략하게  
닉네임과 성별을 표시

감사합니다

