모바일 API 활용
지도 애플리케이션

Created by 방형진

ⓒSaebyeol Yu. Saebyeol's PowerPoint

A Table of Contents.

1 프로젝트 개요 Project Summary

2 프로젝트 진행 Project Progress

3 프로젝트 결과 Outcome

Part 1,

프로젝트 개요

Part 1,

프로젝트 개요

Project Summary

001

Android Studio

002

Kotlin

003

Naver Map API

004

공공 데이터 API

Part 1,

프로젝트 개요

Project Summary

- 구글이 JetBrains의 IntelliJ IDEA를 기반으로 만든 IDE
- 안드로이드의 공식언어인 Kotlin을 지원
- UI가 보기 편하고 Git 연동 등 다양한 기능 지원
- 학교 수업에서 사용해 봄

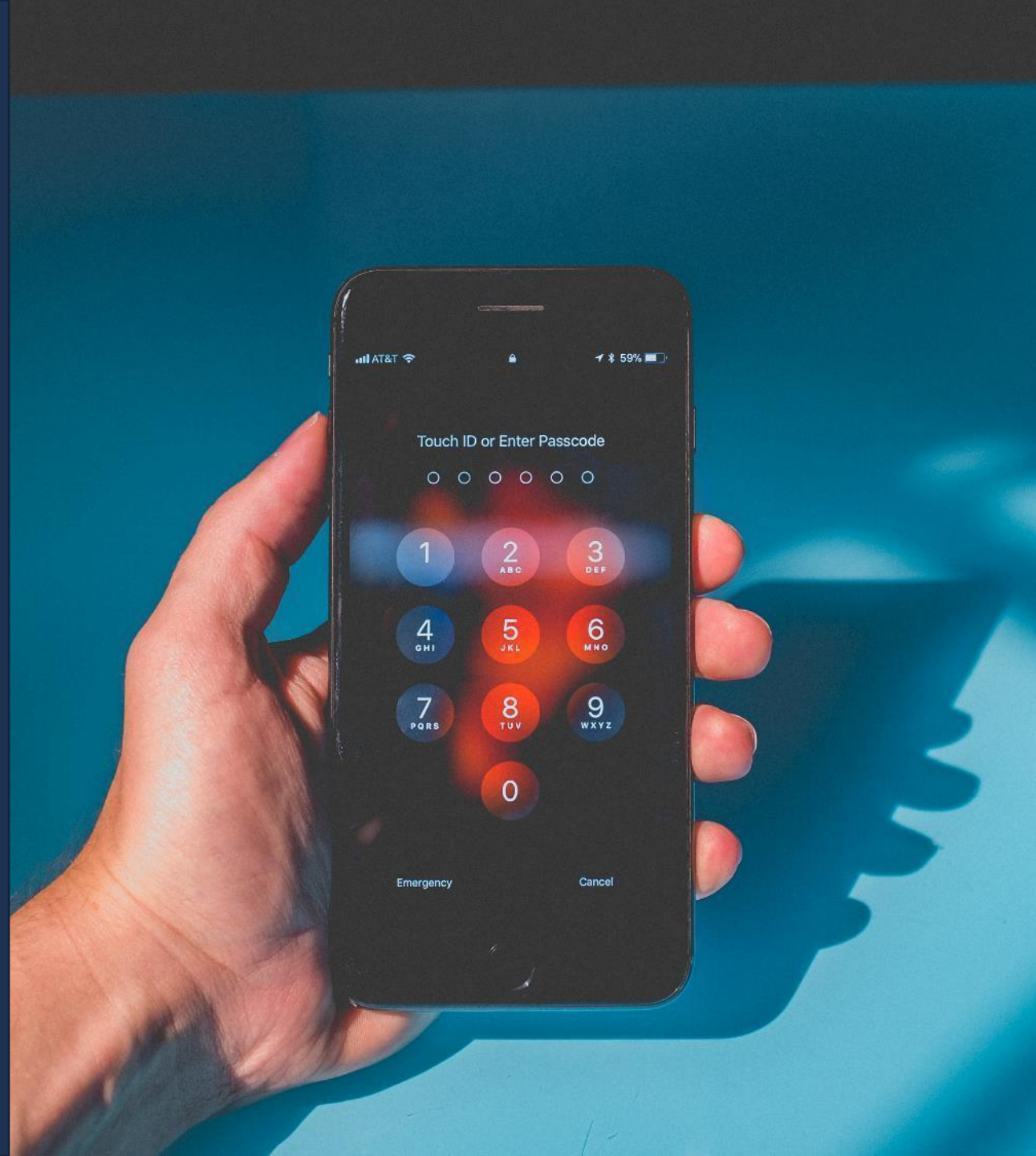
- 2017년에 구글이 안드로이드의 공식 언어로 추가
- Java에 비해 간결한 문법
- Java와 상호 100% 대응
- 새로운 언어를 시도해 보고 싶어서

- Mobile Dynamic Map을 사용해 지도 표현
- 다양한 옵션과 오버레이 등을 지원
- Reverse Geocoding을 사용해 지역 정보 사용
- REST API로 정보의 송수신이 잘 구축되어 있음
- JSON 형태의 API Response를 사용

- 코로나19 관련 공공 데이터 API를 활용
- 누구나 쉽게 이용할 수 있음
- XML 형태의 API Response를 사용

Part 2,

프로젝트 진행



Part 2,

프로젝트 진행

Project Progress

애플리케이션 주요 기능

지도

- 안드로이드 현재 위치 기능을 사용하는 네이버 맵을 프래그먼트에 표현

행정구역 표시

- 대한민국 행정구역 중 광역시도에 해당하는 구역의 경계를 시각화 하여 표현

코로나 19

- 공공 코로나 19관련 데이터 표시
- 오늘의 확진자, 사망자, 환자 유입 등을 각 광역시도에 따라 표시

Part 2,

프로젝트 진행

Project Progress

진행 과정 (시간 순)

시작

05-23

기본 액티비티 제작

API 선택
초기화면 액티비티 제작
맵화면 액티비티 제작

05-24

기능 구현

Json 파일을 읽어오는 기능 구현
읽어온 파일에서 좌표 데이터 추출
네이버맵 폴리곤 객체 생성

05-25

데이터 전처리

행정구역 SHP파일 수집
SHP -> Json 파일 변환
폴리곤 객체 생성 기능 강화

05-26

기능 구현&보수

국외 지역 선택 오류 수정
초기화면 레이아웃 수정

05-28

API 사용 & HTTP 통신

행정구역 이름을 Reverse Geocoding API로 가져옴
Retrofit으로 HTTP 통신

06-01

데이터 전처리 & 레이아웃 수정

Json 파일 통합
모든 행정구역 동시에 레이어 표시 기능 구현
맵 화면 레이아웃 수정

06-02

API 사용

공공데이터 코로나 API 사용
광역시도 별로 코로나 데이터 표시
전국 코로나 데이터 표시

완성

Part 3,

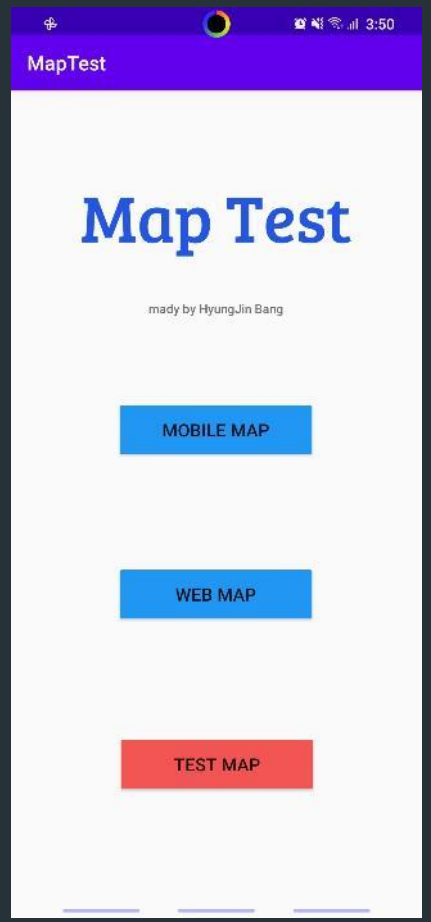
프로젝트 결과

Part 3,

프로젝트 결과

Project Outcome

애플리케이션 실행 스크린샷 및 주요 코드 설명



초기 화면

```
package com.example.maptest

import ...

class MainActivity : AppCompatActivity() { // 초기 화면 설정용

    @RequiresApi(Build.VERSION_CODES.O)
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        val mobile : Button = findViewById(R.id.mobileMap)
        mobile.setOnClickListener {
            val nextIntent = Intent(this, MobileMapActivity::class.java)
            startActivity(nextIntent)
        }
        val web : Button = findViewById(R.id.webMap)
        web.setOnClickListener {
            val nextIntent2 = Intent(this, WebMapActivity::class.java)
            startActivity(nextIntent2)
        }

        val test : Button = findViewById(R.id.testMap)
        test.setOnClickListener {

            val nextIntent3 = Intent(this, MobileMapActivityTest::class.java)
            startActivity(nextIntent3)
        }
    }

    override fun onBackPressed() {  // 뒤로가기 시 alertdialog를 띄워 확실히 종료할 것인지 확인함
        val builder: AlertDialog.Builder = AlertDialog.Builder(this)
        builder.setTitle("종료 확인").setMessage("정말 종료할거에요?")
        builder.setPositiveButton("그래") { dialog, which ->
            finish()
        }
        builder.setNegativeButton("아니") { dialog, which -> }
        builder.show()
    }
}
```

구현 코드

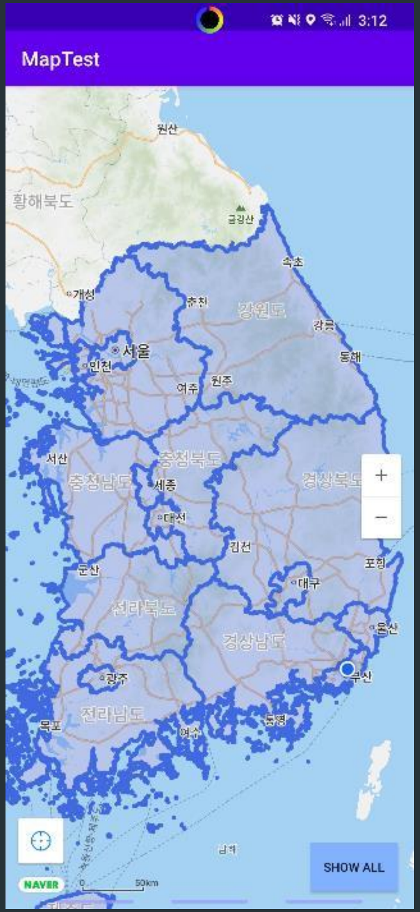
- 앱 실행 시 가장 먼저 표시될 화면 액티비티 코드
- Button 객체를 사용해 다음 액티비티로 이동하는 기능 구현
- 화면에서 스마트폰 '뒤로 가기' 키를 누를 경우 onBackPressed() 함수에서 앱을 바로 종료하지 않고, 다시 한 번 종료할 것인지 물어봄

코드 설명

프로젝트 결과

Project Outcome

애플리케이션 실행 스크린샷 및 주요 코드 설명



행정 구역 경계

```kotlin
inner class MakeLocationArrayThread : Thread() {    // 스레드에서 행정구역의 좌표를 배열에 넣어 줌
    override fun run() {
        try {
            val am = resources.assets   // 에셋 폴더를 사용가능하게 해줌
            val inputStream= am.open("광역시도_변환.json")    // 해당 광역시도의 geojson 파일 열기
            val findName = "CTP_KOR_NM"    // 광역시도 이름 변수 명
            val jsonString = inputStream.bufferedReader().use { it.readText() } // 해당 파일에서 이름이랑 주소 좌표 읽어옴
            val jObject = JSONObject(jsonString)
            val jsonlist = jObject.getJSONArray("features")
            var jsonObjectCoordinatesList : ArrayList<String>
            val jsonRegex = "[\\[\\]]".toRegex()
            for (name in locationArray) {
                for (i in 0 until jsonlist.length()) {  // geojson 파일에서 해당하는 좌표를 읽고 폴리곤을 만들어 폴리곤 리스트에 넣음
                    val jsonObject = jsonlist.getJSONObject(i)
                    val jsonObjectProperties = jsonObject.getJSONObject("properties")
                    val jsonObjectPropertiesNameKor = jsonObjectProperties.getString(findName)
                    val jsonObjectGeometry = jsonObject.getJSONObject("geometry")
                    if (name == jsonObjectPropertiesNameKor) {
                        if (jsonObjectGeometry.getString("coordinates").contains("]]],")) { // 해당 지역이 다수의 폴리곤으로 이루어진 경우
                            val multiPolyList: ArrayList<String> = jsonObjectGeometry.getString("coordinates").split("]]],") as
ArrayList<String>

                            for (j in multiPolyList) {
                                val jsonObjectCoordinates = jsonRegex.replace(j,"")
                                jsonObjectCoordinatesList = jsonObjectCoordinates.split(",") as ArrayList<String>
                                val jsonObjectLatLngList: ArrayList<LatLng> = arrayListOf<LatLng>()
                                var index = 0
                                while(index < jsonObjectCoordinatesList.size) {
                                    jsonObjectLatLngList.add(LatLng(jsonObjectCoordinatesList[index + 1].toDouble(),
                                        jsonObjectCoordinatesList[index].toDouble()))
                                    index += 2
                                }
                                val polygon = PolygonOverlay()
                                polygon.coords = jsonObjectLatLngList
                                if (name == "전라남도" && jsonObjectLatLngList.size > 1000) {  // 전남은 도 중 유일하게 내부에 구멍이 있음
                                    val holeCoordinates = jsonRegex.replace(jsonObjectGeometry.getString("holes"),"")
                                    val holeCoordinatesList = holeCoordinates.split(",") as ArrayList<String>
                                    val holeLatLngList: ArrayList<LatLng> = arrayListOf<LatLng>()
                                    var holeIndex = 0
                                    while(holeIndex < holeCoordinatesList.size) {
                                        holeLatLngList.add(LatLng(holeCoordinatesList[holeIndex + 1].toDouble(),
                                        holeCoordinatesList[holeIndex].toDouble()))
                                        holeIndex += 2
                                    }
                                    polygon.holes = listOf(holeLatLngList) // 광주부분을 전남에서 빼줌
                                }
                                polygonArray.add(polygon)
                            }
                        }
                        else {  // 해당 지역이 폴리곤 1개로만 이루어진 경우
                            val jsonObjectCoordinates = jsonRegex.replace(jsonObjectGeometry.getString("coordinates"),"")
                            jsonObjectCoordinatesList = jsonObjectCoordinates.split(",") as ArrayList<String>
                            val jsonObjectLatLngList: ArrayList<LatLng> = arrayListOf<LatLng>()
                            var index = 0
                            while(index < jsonObjectCoordinatesList.size) {
                                    jsonObjectLatLngList.add(LatLng(jsonObjectCoordinatesList[index + 1].toDouble(),
                                    jsonObjectCoordinatesList[index].toDouble()))
                                index += 2
                            }
                            val polygon = PolygonOverlay()
                            polygon.coords = jsonObjectLatLngList
                            polygonArray.add(polygon)
                        }
                    }
                }
                multiPolygonArray.add(polygonArray)
                polygonArray = arrayListOf()
            }
        }catch (e: Exception){
            e.printStackTrace()
        }
    }
}
```

구현 코드

- 메인 화면에서 맵 화면으로 이동하였을 때 액티비티의 onCreate() 함수에서 바로 실행될 스레드의 코드
- 행정구역의 경계좌표 Json 파일을 읽고 좌표 데이터를 네이버 맵 위에 표시할 폴리곤 데이터로 가공, 폴리곤 배열로 만들어 저장
- 액티비티 시작 시 1번만 실행되므로 추후에 상호작용으로 인해 불필요한 좌표 연산을 하지 않아도 됨

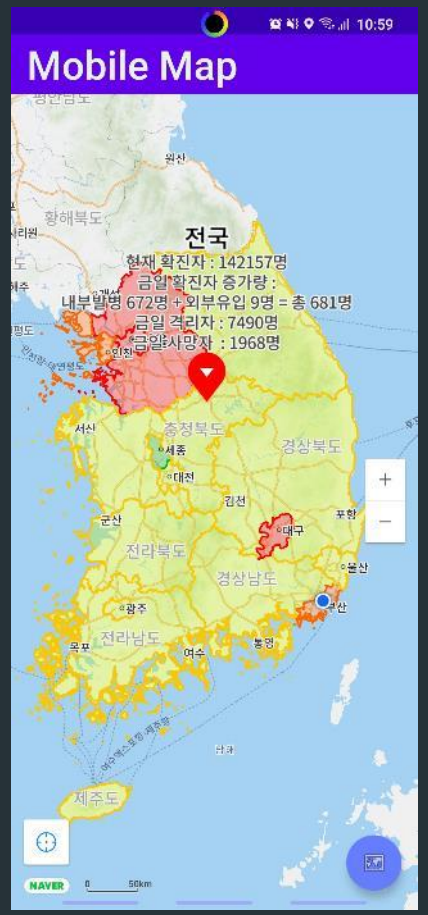
코드 설명

Part 3,

프로젝트 결과

Project Outcome

애플리케이션 실행 스크린샷 및 주요 코드 설명



전국 코로나 현황

```
inner class NetworkThread: Thread(){    // 스레드에서 코로나19 데이터를 공공데이터 api를 사용해서 읽어옴
    @RequiresApi(Build.VERSION_CODES.O)
    override fun run() {
        try {
            // 접속할 페이지의 주소
            val site = "http://openapi.data.go.kr/openapi/service/rest/Covid19/getCovid19SidoInfStateJson?" +
                    "serviceKey=XXXXX"
            val url = URL(site)
            val conn = url.openConnection()
            val input = conn.getInputStream()
            val factory = DocumentBuilderFactory.newInstance()
            val builder = factory.newDocumentBuilder()
            // doc xml문서를 읽도록 자바에서 분석을 준비
            val doc = builder.parse(input)
            // root는 xml 문서의 모든 데이터들을 갖고 있는 객체
            val root = doc.documentElement
            // xml 문서에서 태그 이름이 item인 태그들이 item_node_list에 리스트로 담김
            val itemNodeList = root.getElementsByTagName("item")
            // item_node_list에 들어있는 태그 객체 수만큼 반복함
            for (name in covidLocationArray) {
                for(i in 0 until itemNodeList.length){
                    // i번째 태그 객체를 item_element에 넣음
                    val itemElement = itemNodeList.item(i) as Element
                    // item태그 객체에서 원하는 데이터를 태그이름을 이용해서 데이터를 가져옴
                    // xml 문서는 태그 이름으로 데이터를 가져오면 무조건 리스트로 나옴
                    val gubunList = itemElement.getElementsByTagName("gubun")
                    val gubunNode = gubunList.item(0) as Element
                    val gubun = gubunNode.textContent
                    if (gubun == name) {
                        val defcntList = itemElement.getElementsByTagName("defCnt")
                        val defcntNode = defcntList.item(0) as Element
                        val defCnt = defcntNode.textContent
                        val deathcntList = itemElement.getElementsByTagName("deathCnt")
                        val deathcntNode = deathcntList.item(0) as Element
                        val deathCnt = deathcntNode.textContent
                        val incdecList = itemElement.getElementsByTagName("incDec")
                        val incdecNode = incdecList.item(0) as Element
                        val incDec = incdecNode.textContent
                        val isolingcntList = itemElement.getElementsByTagName("isolIngCnt")
                        val isolingcntNode = isolingcntList.item(0) as Element
                        val isolIngCnt = isolingcntNode.textContent
                        val outsideList = itemElement.getElementsByTagName("overFlowCnt")
                        val outsideNode = outsideList.item(0) as Element
                        val outside = outsideNode.textContent
                        val insideList = itemElement.getElementsByTagName("localOccCnt")
                        val insideNode = insideList.item(0) as Element
                        val inside = insideNode.textContent
                        runOnUiThread {
                            val inputCovidArray : ArrayList<String> = arrayListOf()
                            inputCovidArray.add(defCnt)
                            inputCovidArray.add(inside)
                            inputCovidArray.add(outside)
                            inputCovidArray.add(incDec)
                            inputCovidArray.add(isolIngCnt)
                            inputCovidArray.add(deathCnt)
                            covidNumberArray.add(inputCovidArray)
                        }
                        break
                    }
                }
            }
        }catch (e: Exception){
            e.printStackTrace()
        }
    }
}
```

구현 코드

- HTTP 통신을 통해 공공데이터 코로나 19 API를 사용
- 앞선 지리 정보와 동일하게 onCreate()시 1번만 실행
- 공공데이터 API연결에는 httpurlconnection을 사용
- Xml 형식으로 받아온 데이터를 태그 별로 분류, 필요한 데이터만 뽑아내 배열로 저장

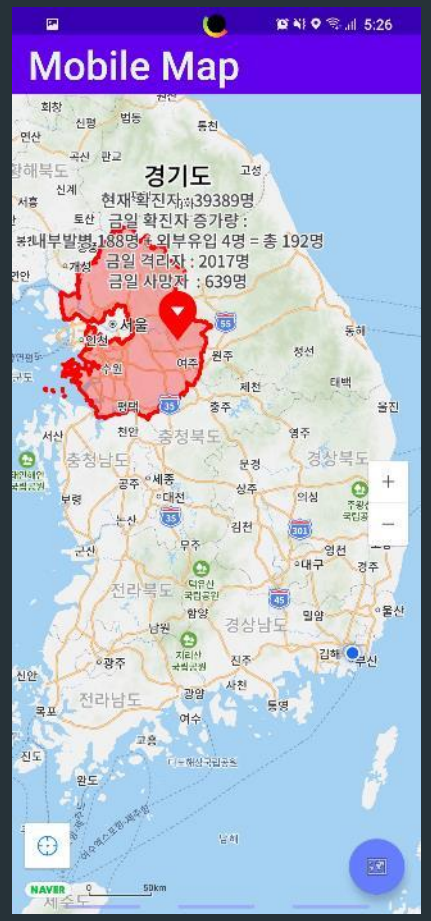
코드 설명

Part 3,

프로젝트 결과

Project Outcome

애플리케이션 실행 스크린샷 및 주요 코드 설명



광역시도 코로나 현황

```
fun returnLocation(coord : LatLng){
    var responseString: ResultGetLocationJson?
    val retrofit = Retrofit.Builder()
        .baseUrl(BASE_URL_NAVER_API)
        .addConverterFactory(GsonConverterFactory.create())
        .build()
    val api = retrofit.create(NaverReverseGeoAPI::class.java)
    val requestCoord : String = coord.longitude.toString() + "," + coord.latitude.toString()
    val callGetLocation = api.getLocation(CLIENT_ID, CLIENT_SECRET, requestCoord)
    callGetLocation.enqueue(object : Callback<ResultGetLocationJson> {
        override fun onResponse(
            call: Call<ResultGetLocationJson>,
            response: Response<ResultGetLocationJson>
        ) {
            responseString = response.body()
            if (responseString?.results?.size != 0 && responseString?.results?.get(0)?.region?.area1?.name != null) {
                val mapString = responseString?.results?.get(0)?.region?.area1?.name     // 선택한 지역 광역시도 이름
                val mapStringIndex = locationArray.indexOf(mapString)
                val covidInfo = covidNumberArray[mapStringIndex]
                val mapCenterCoordLongi = responseString?.results?.get(0)?.region?.area1?.coords?.center?.x  // 선택한 지역 중심지 경도
                val mapCenterCoordLati = responseString?.results?.get(0)?.region?.area1?.coords?.center?.y  // 선택한 지역 중심지 위도
                indexOfLocationArray = locationArray.indexOf(mapString)
                counter = if (covidInfo[0].toInt() > 10000) {
                    3
                } else if (covidInfo[0].toInt() > 5000) {
                    2
                } else if (covidInfo[0].toInt() > 1000) {
                    1
                } else {
                    0;
                }
                if (mapCenterCoordLati != null && mapCenterCoordLongi != null) {    // 선택한 지역의 중심 좌표에 마커를 생성
                    marker.position = LatLng(mapCenterCoordLati.toDouble(),mapCenterCoordLongi.toDouble())  // 마커 위경도 설정
                    marker.captionText = "$mapString"   // 마커 텍스트 설정
                    marker.setCaptionAligns(Align.Top)  // 마커 텍스트 위치
                    marker.captionTextSize = 25f        // 마커 텍스트 크기
                    marker.subCaptionText = "현재 확진자 : ${covidInfo[0]}명" +
                            "\n금일 확진자 증가량 : \n내부발병 ${covidInfo[1]}명" +
                            " + 외부유입 ${covidInfo[2]}명 = " +
                            "총 ${covidInfo[3]}명" +
                            "\n금일 격리자 : ${covidInfo[4]}명" +
                            "\n금일 사망자  : ${covidInfo[5]}명"
                    marker.subCaptionColor = Color.DKGRAY
                    marker.subCaptionTextSize = 17f
                    marker.isHideCollidedSymbols = true
                    marker.icon = MarkerIcons.BLACK
                    marker.iconTintColor = lineColorArray[counter]
                    marker.map = naverMap                // 마커 표시
                }
                //Log.d("결과", "인덱스 : $indexOfLocationArray")
                for (i in multiPolygonArray[indexOfLocationArray]) {
                    i.color = colorArray[counter]  // 폴리곤 내부 색깔 설정
                    i.outlineColor = lineColorArray[counter]  // 폴리곤 외곽선 색깔 설정
                    i.outlineWidth = 10 // 폴리곤 외곽선 굵기 설정
                    i.map = naverMap    // 폴리곤 표시
                }
            }
//            Log.d("결과", "전국 : ${responseString?.results?.size} $requestCoord")
        }
        override fun onFailure(call: Call<ResultGetLocationJson>, t: Throwable) {
//            Log.d("결과:", "실패 : $t")
        }
    })
}
```

구현 코드

- 광역시도 별 코로나 데이터를 표시하기위해 Reverse Geocoding API 사용
- 지도에서 클릭한 부분의 좌표를 네이버 Reverse Geocoding API로 해당하는 지역명으로 변경
- Reverse Geocoding API연결에는 Retrofit을 사용
- Json 형식으로 받아온 데이터를 종류 별로 분류, 필요한 데이터만 뽑아내 배열로 저장

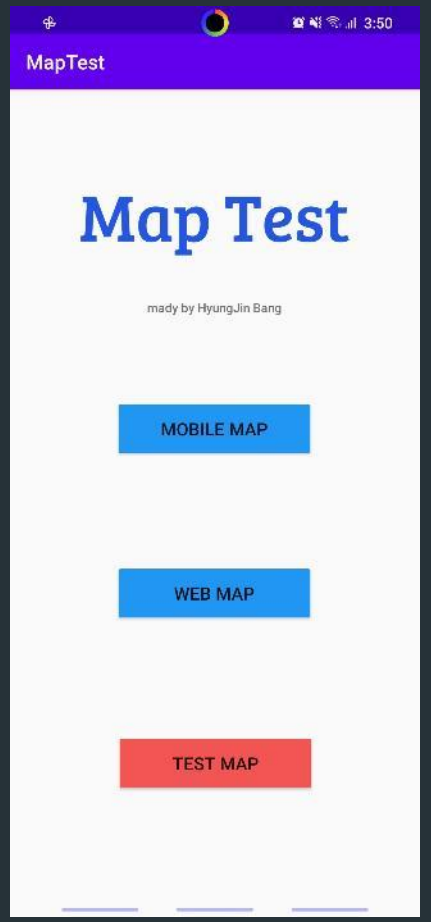
코드 설명

Part 3,

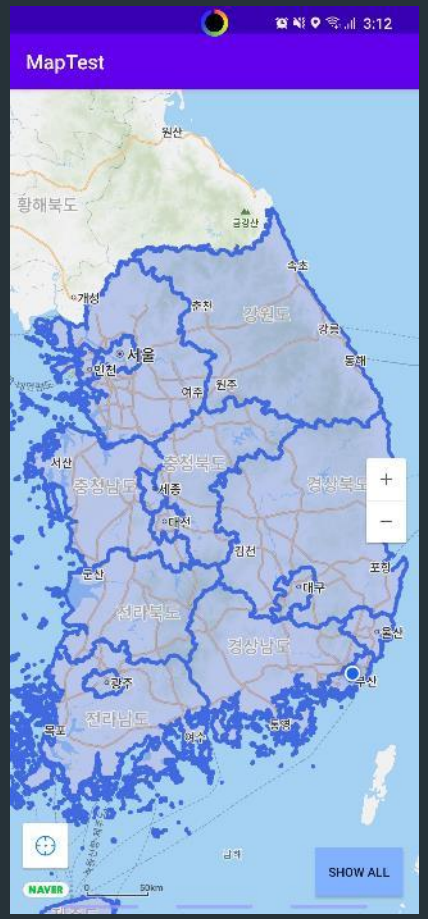
프로젝트 결과

Project Outcome

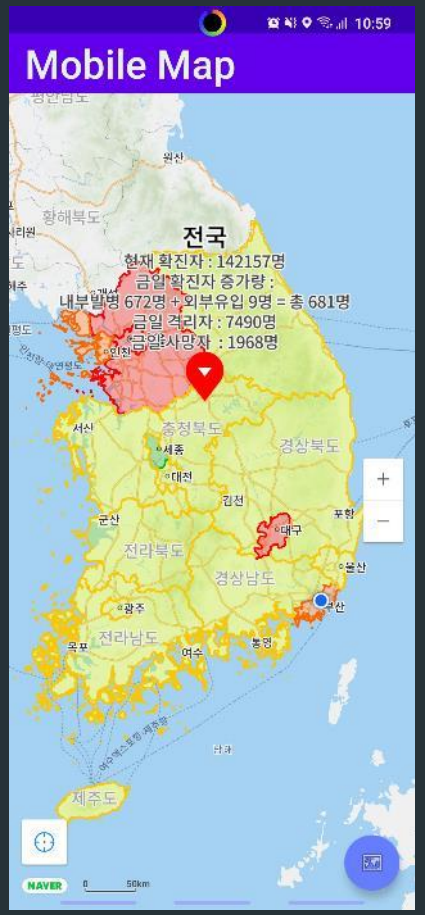
애플리케이션 실행 스크린샷



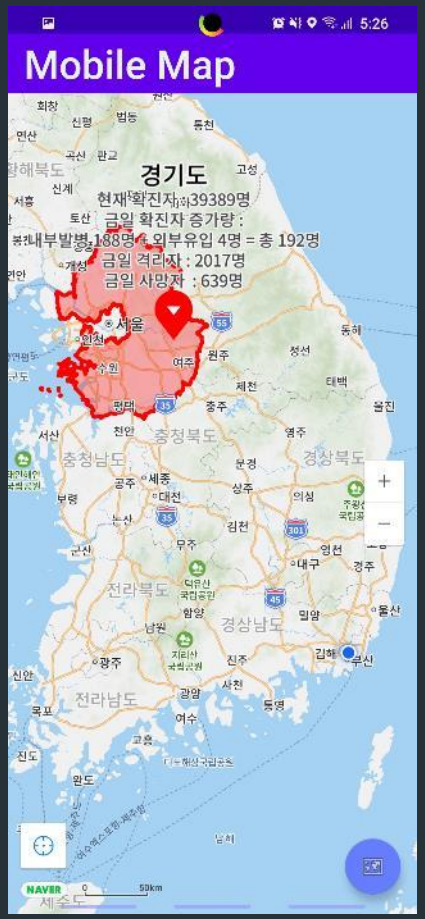
초기 화면



행정구역 경계



전국 코로나 현황



광역시도 코로나 현황

Part 3,

프로젝트 결과

Project Outcome

프로젝트 GitHub Repository

https://github.com/Banghyungjin/MapTest


Q&A

감사합니다

ⓒSaebyeol Yu. Saebyeol's PowerPoint