Week 4.
Android Assignment

안드로이드 4주차 과제

LEVEL1 필수과제

저번주차에 이어서 해주시면 됩니다!

Level1 필수과제

로그인/회원가입 API 연동

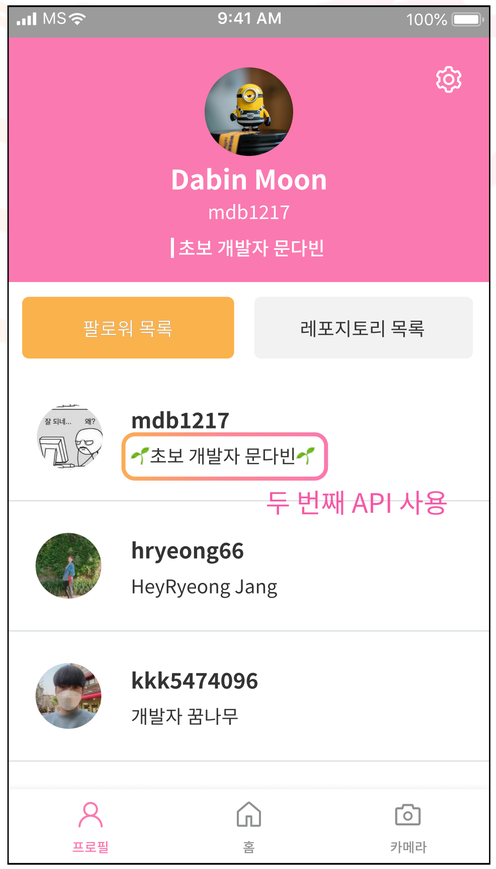
API 문서링크

- 해당 링크로 들어가서 로그인 회원가입 서버 통신을 구현 해보세요.
- POSTMAN 테스트 + 회원가입 완료 + 로그인 완료 이미지를 리드미에 올려주세요.
- retrofit interface와 구현체, Request/Response 객체에 대한 코드 필수
- 그 외 유저조회(+Email로 유저 조회), 유저 수정, 유저 삭제 구현은 자유!(물론 하면 더 좋겠죠 ㅎㅎ)

LEVEL2 도전과제

좀 더 심화적인 성장을 원하는 그대에게~~!!

Level2 도전과제 2 - 1



첫 번째 API 사용

GitHub API 연동해서 리스트로 띄우기 (Follower 리스트 연동하기)

https://docs.github.com/en/rest/reference/users#list-followers-of-a-user
(팔로워 리스트 가져오는 API)

https://docs.github.com/en/rest/reference/users#get-a-user
(유저 정보 가져오는 API)

- 우선 첫 번째 링크의 API로 팔로워 리스트를 불러온다!
- 가져온 팔로워 리스트의 정보(UserName)를 이용해
  두 번째 링크의 API로 유저들의 상세정보도 불러온다.
- 왼쪽과 같은 리스트를 완성한다!

(UserName 하단의 상세정보 구성의 경우 자유입니다~!!)

Level2 도전과제 2 - 2

```
@Headers( ...value: "Content-Type: application/json")
@POST( value: "user/login")
fun postLogin(
    @Body requestLoginData: RequestLoginData
) : Call<ResponseLoginData>

@Headers( ...value: "Content-Type: application/json")
@POST( value: "user/{Id}")
fun getUserInfo(
    @Path( value: "Id") Id: String
) : Call<ResponseLoginData>
```

매번 겹치는 헤더를 꾸역꾸역 넣어줘야하나.. (OkHttp 활용해보기)

- 똑같은 헤더들.. 매번 저렇게 넣어줘야하나..
- OkHttp를 활용하면 겹치는 헤더를 매번 넣어줄 필요가 없다!
- OkHttp를 활용해 겹치는 헤더를 집어넣어주고 retrofit2에 연동시켜보자!

Level2 도전과제 2 - 3

```
package org.sopt.androidseminar

data class ResponseLoginData(
    val status : Int,
    val success : Boolean,
    val message : String,
    val data : Data
) {
    data class Data(
        val id : Int,
        val name : String,
        val email : String
    )
}
```

공통되는 부분 매번 작성해야하나..

- 흠 이부분 다른 Request/Response 데이터에도 들어가던데.. 매번 작성해줘야하나..
- 해당 중복 내용을 어떻게하면 해결할 수 있을지 방법을 찾아 적용해봅시다!
- hint : Wrapper 클래스

강아지 사진 출처는 iOS 파트장님 강아지입니다 ^^

LEVEL3 심화과제

다음 레벨은 처음 안드로이드를 접하시는 분, 개발을 처음 해보시는 분은
외계어 혹은 한글이란 탈을쓴 다른나라 말로 들릴 수 있습니다!!

SOPT 세미나 이상으로 성장하고 싶은 분들에게 추천드립니다!

Level3 심화과제 3 - 1

비동기 처리 라이브러리를 이용해 서버통신을 해볼까?

```
private fun initNetwork() {
    val requestLoginData = RequestLoginData(
        email = binding.etId.text.toString(),
        password = binding.etPass.text.toString()
    )

    val call: Call<ResponseLoginData> = ServiceCreator.sampleService.postLogin(requestLoginData)

    call.enqueue(object : Callback<ResponseLoginData> {
        override fun onResponse(
            call: Call<ResponseLoginData>,
            response: Response<ResponseLoginData>
        ) {
            if(response.isSuccessful) {
                val data = response.body()?.data

                Toast.makeText( context: this@MainActivity,  text: "${data?.email}님 반갑습니다!", Toast.LENGTH_SHORT).show()
                startActivity(Intent( packageContext: this@MainActivity, SecondActivity::class.java))
            } else
                Toast.makeText( context: this@MainActivity,  text: "로그인에 실패하셨습니다", Toast.LENGTH_SHORT).show()
        }

        override fun onFailure(call: Call<ResponseLoginData>, t: Throwable) {
            Log.e( tag: "NetworkTest",  msg: "error:$t")
        }
    })
}
```

길어도 너무 길다..

- 서버 연결마다 Callback 익명 클래스를 제작하기엔.. 길어도 너무 길어요..
- 안드로이드에는 대표적인 2가지 비동기 처리 라이브러리가 있습니다!
- 한 번 그걸 활용해서 왼쪽의 코드를 보다 간결하게 만들어봐요😊

과제 제출 방법

과제 제출 시 필수로 깃허브에 올려주세요🔥🔥
(29기 안팟 오가니제이션에 있는 본인의 레포지토리에 올려주세요!)

- **해당 과제 관련 프로젝트 파일**

- **해당 과제를 어떻게 구현한지 설명하는 README파일**

- 구현한 로직(코드)을 설명하는 내용

- 이번 과제를 통해 배운내용, 성장한 내용

- 실행화면

위 내용들이 모두 들어가야합니다🔥🔥

4차 과제 제출 마감기한

11. 7(일) 23:59