2020

NAVER

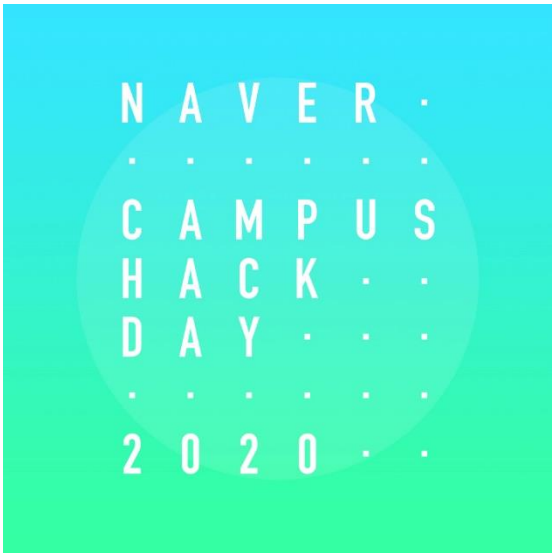
HACK DAY

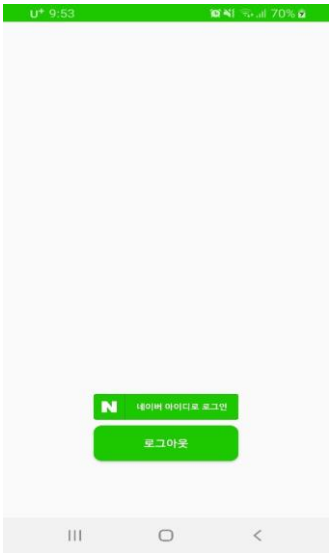
목 차

1. 2020 NAVER HACKDAY

- 기간 : 2020. 5.3(일) ~ 2020. 5. 24(일)
- 파트 : Android
- 내용 : 2020 NAVER HACKDAY에 참여하여 과제 수행 및 멘토링
- 과제 내용 : 네이버 검색 Api를 활용한 쇼핑몰 앱
- GitHub : https://github.com/kangmin1012/Naver_ForestShopping.git
- 간단 후기 : https://kangmin1012.tistory.com/32

2. 후기



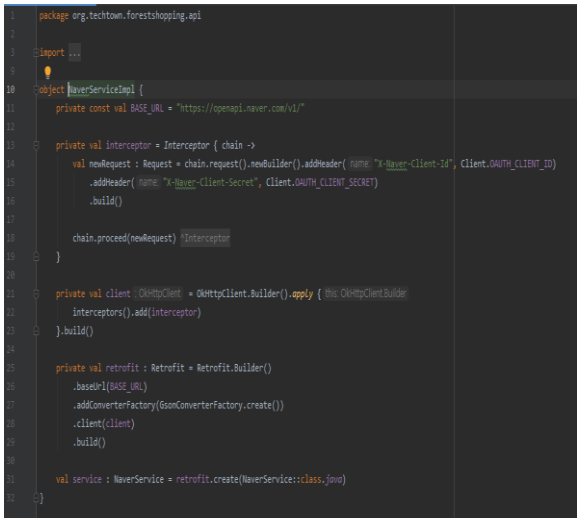


Naver에서 주최하는 1박2일 해커톤 NAVER HACK DAY에 참여했다. 기존 해커톤으로 진행되던 행사가 올해 코로나19의 영향으로 취소되었고, 온라인 멘토링으로 대체되었다.
사전에 본인이 원하는 프로젝트를 선택하고, 해당 프로젝트를 선택한 팀원들과 멘토 한 명으로 구성된 그룹에서 정해진 과제를 수행하고, 지속적인 멘토링을 받는 형식으로 바뀌었다.
나는 여기서 NAVER 쇼핑 검색 API를 이용하여 간단한 쇼핑몰 APP을 만드는 프로젝트를 선택하여, 행사 기간동안 개발에 임했다.
처음 행사에 참여했던 거라 대단한 기술을 사용해서 앱을 멋지고 완벽하게 만들어야 하는 압박감에 사로 잡혀서 내가 만들어낸 과제가 대단하게 느껴지지 않았다. 기본기에 충실한 앱으로 디자인적인 요소는 최대한 배제한 상태로 만들었다. (스스로 디자인을 하기에 어려움이 많다.)

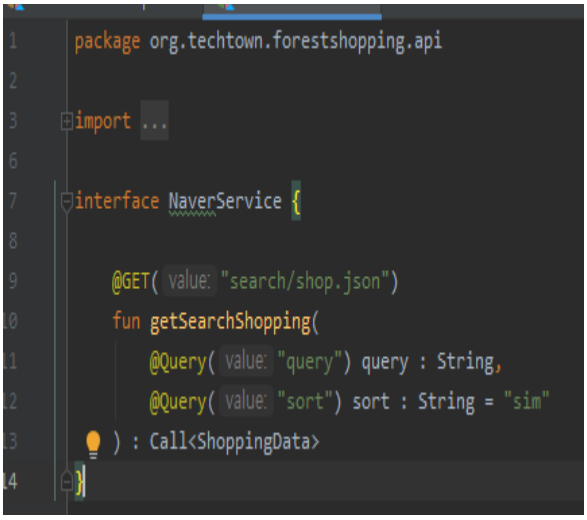
처음 Splash 화면이 지나가면 로그인 창이 나온다. 로그인은 네이버 아이디로 로그인을 해야 하며, 이를 구현하는 과정에서 처음 네이버 api를 사용하는 경험이 되었다. 처음으로 네이버 공식 api 문서를 읽어보면서 구현했으며 크게 어려움을 느끼지는 않았다.

2-1. 메인화면

로그인에 성공하게 되면 Main 화면이 나오는데, 이번 과제에 핵심 기능들은 거의 다 Main 화면에 존재했다. 사용자가 상품을 검색하면 네이버 쇼핑 검색 api를 이용해 상품 목록들을 리스트 형식으로 보여줘야 했다. 여기서 사용자가 상품 목록을 격자 또는 리스트 형식으로 보여줄 수 있어야 했다.



```
package org.techtown.forestshopping.api

import ...

object NaverServiceImpl {
    private const val BASE_URL = "https://openapi.naver.com/v1/"

    private val interceptor = Interceptor { chain ->
        val newRequest : Request = chain.request().newBuilder().addHeader( name: "X-Naver-Client-Id", Client.OAUTH_CLIENT_ID)
            .addHeader( name: "X-Naver-Client-Secret", Client.OAUTH_CLIENT_SECRET)
            .build()

        chain.proceed(newRequest) //Interceptor
    }

    private val client : OkHttpClient = OkHttpClient.Builder().apply { this: OkHttpClient.Builder
        interceptors().add(interceptor)
    }.build()

    private val retrofit : Retrofit = Retrofit.Builder()
        .baseUrl(BASE_URL)
        .addConverterFactory(GsonConverterFactory.create())
        .client(client)
        .build()

    val service : NaverService = retrofit.create(NaverService::class.java)
}
```



```
package org.techtown.forestshopping.api

import ...

interface NaverService {

    @GET( value: "search/shop.json")
    fun getSearchShopping(
        @Query( value: "query") query : String,
        @Query( value: "sort") sort : String = "sim"
    ) : Call<ShoppingData>
}
```

해당 부분은 검색 api를 참조하여 Retrofit2로 변형하여 사용하였다. 기존 api 가이드를 그대로 따라하는데 어려움이 있어서, Retrofit2을 사용하면 편하게 쓸 수 있지 않을까 하는 마음에 내가 잘 사용할 수 있는 기술을 이용하였다.

사용자의 선택에 따라 격자 또는 리스트 형태로 상품 리스트를 보이게 하기 위해서 RadioButton을 만들었고, 어떤 것을 눌렀는지에 따라 보여주는 형식을 다르게 지정했다.

```
private fun initRcv(LAYOUT_CODE : Int){
    shoppingAdapter = ShoppingAdapter( context: this, LAYOUT_CODE)
    if (LAYOUT_CODE == 0 ){
        binding.searchrcv.layoutManager = GridLayoutManager( context: this, spanCount: 2,GridLayoutManager.VERTICAL, reverseLayout: false)
    }
    else {
        binding.searchrcv.layoutManager = LinearLayoutManager( context: this)
    }
    binding.searchrcv.adapter = shoppingAdapter
    if(items.isNotEmpty()) {
        shoppingAdapter.data = items
```

상품 리스트를 보여주는 RecyclerView 선언

initRcv() 함수를 사용해서 RecyclerView를 초기화 시켜주었는데 LAYOUT_CODE 변수에 따라 보여주는 형태를 다르게 가져갔다. 여기서 LAYOUT_CODE는 RadioButton이 선택됨에 따라 다른 값을 가지게 된다. 여기서 리스트 형태가 달라질 때마다 해당 함수가 지속적으로 호출된다는 점에서 initRcv()라는 함수 명을 변경하는 것이 좋다는 조언을 들었다.

```
@SuppressLint( ...value: "ResourceAsColor")
fun initRadio(){
    rg_list.setOnCheckedChangeListener { _, checkedId ->
        when(checkedId){
            R.id.rb_linear -> {
                rb_linear.setTextColor(Color.parseColor( colorString: "#ffffff"))
                rb_grid.setTextColor(R.color.naverColor)
                LAYOUT_CODE = 1
                initRcv(LAYOUT_CODE)
            }
            R.id.rb_grid ->{
                rb_linear.setTextColor(R.color.naverColor)
                rb_grid.setTextColor(Color.parseColor( colorString: "#ffffff"))
                LAYOUT_CODE = 0
                initRcv(LAYOUT_CODE)
            }

        }
    }
}
```

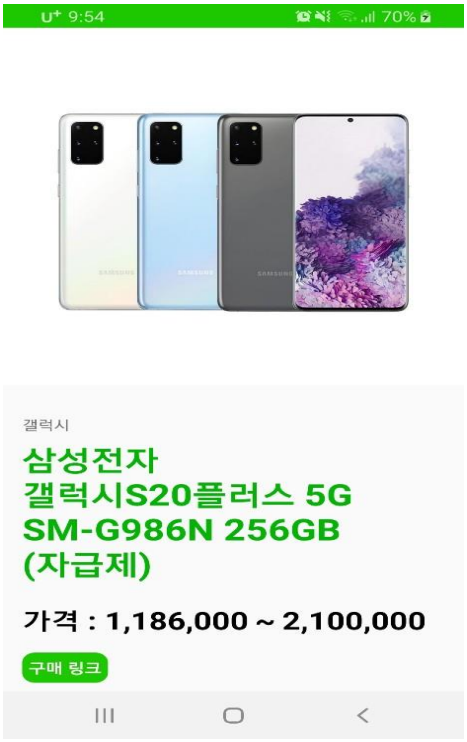
RadioButton 클릭 이벤트 구현 함수. initRcv() 함수가 클릭 시 호출된다.

좌측 '정렬'버튼을 누르면 정확도순, 날짜순, 가격순으로 정렬을 할 수 있다.
이는 BottomSheet를 이용하였으며, 지난 Crecker 프로젝트에서 사용한 코드를 참조하였다.
네이버 쇼핑 검색 api에서는 Query값에 따라 위와 같은 정렬로 데이터를 제공해주기 때문에 정렬을 바꿀 때마다 새롭게 통신을 하는 방향으로 코드를 작성했다.

```
@SuppressLint( ...value: "ResourceAsColor")
fun order(view : View){
    bottomSheetDialog = BottomSheetDialog( context: this)
    bottomSheetDialog.setContentView(R.layout.bottom_sheet)
    bottomSheetDialog.show()

    bottomSheetDialog.bottom_correct_tv.setOnClickListener { it: View!
        mSort = "sim"
        searchClick(view)
        bottomSheetDialog.dismiss()
    }

    bottomSheetDialog.bottom_day_tv.setOnClickListener { it: View!
        mSort = "date"
        searchClick(view)
        bottomSheetDialog.dismiss()


    }

    bottomSheetDialog.bottom_price_tv.setOnClickListener { it: View!
        mSort = "asc"
        searchClick(view)
        bottomSheetDialog.dismiss()

   }

    when(mSort){
        "sim" -> {
            bottomSheetDialog.bottom_correct_tv.setTextColor(Color.parseColor( colorString: "#1EC800"))
            bottomSheetDialog.bottom_correct_tv.typeface = Typeface.DEFAULT_BOLD
        }
        "date" -> {
            bottomSheetDialog.bottom_day_tv.setTextColor(Color.parseColor( colorString: "#1EC800"))
            bottomSheetDialog.bottom_day_tv.typeface = Typeface.DEFAULT_BOLD

        }
        "asc" -> {
            bottomSheetDialog.bottom_price_tv.setTextColor(Color.parseColor( colorString: "#1EC800"))
            bottomSheetDialog.bottom_price_tv.typeface = Typeface.DEFAULT_BOLD

        }
    }
}
```

2-1. 상품 상세 화면

사용자가 상품 이름을 검색해서 리스트가 출력되고, 해당 리스트를 누르면 상세 페이지로 이동하게 된다.
이 화면에서는 상품 이미지, 브랜드명, 상품의 최저가와 최고가, 구매링크를 제공한다.
상품 상세 화면 역시 간단하게 구성했으며, 뷰홀더에서 Intent로 값을 받아와서 화면에 뿌려주는 방식으로 코드를 작성했다. Intent로 데이터를 보낼 때 상품에 대한 정보를 담고 있는 Data Class를 통째로 보내는 부분에서 어려움을 겪었다. String이나 Int형은 많이 보낸 경험은 있지만, Data Class 타입을 전달해 본 적은 없기 때문이다.
이 부분에 관해서는 GitHub Issue를 통해 질문 글을 올리고 해결법을 받아 해결할 수 있었다.
서버로부터 전달 받을 데이터를 저장 할 Data Class에 Serializable을 상속하여 Intent에 넣을 수 있도록 수정 하였다.



GitHub Issue를 통한 질문하기

```
itemView.setOnClickListener { it: View!
    val intent = Intent(itemView.context, DetailActivity::class.java)
    intent.putExtra( name: "data", data)

    itemView.context.startActivity(intent)

}
```

ViewHolder에서 Intent에 DataClass를 넣어주는 모습

3. 배운 점

이번 2020 NAVER HACK DAY를 마무리 하고 나서 원래 행사인 해커톤을 통한 성장은 기대할 수 없었지만, 지속적인 온라인 멘토링을 통해 내 자신이 한층 더 성장하게 되었다. 이번에 겪었던 경험이 좋았던 이유는 기존의 안드로이드 기술 습득이 주가 아닌 모든 개발에 있어서 개발자가 가져야할 마인드와 현업에서 일하고 있는 개발자의 이야기를 들어볼 수 있는 기회였기 때문이다. 행사 기간 중 나는 멘토님과 다른 팀원들 앞에서 코드 리뷰를 진행했었다. 이미지를 넣어줄 땐 Glide를, 통신할 때는 Retrofit2를 사용했다는 얘기를 하던 중 멘토님께서 Glide와 Retrofit2를 사용한 이유가 무엇이냐고 물어보셨다. 이 부분에서 나는 5초 정도 침묵을 지킬 수 밖에 없었다. 지금 활동하고 있는 동아리 세미나에서 배운 기술이기 때문에 사용한 것인데 이를 곧이곧대로 말하면 안될 것 같았기 때문이다. 멘토님께서는 우리가 외부의 라이브러리를 사용하려 할 때, 이 라이브러리를 사용하는 이유가 무엇인지를 명확하게 알고 사용하는 것이 중요하다고 말씀해 주셨다. 그리고 내가 개발을 하면서 코드를 작성하는데 있어서 무엇을 중요시 여기는지 생각해 보라고 하셨다. 앱의 기술적인 부분을 부각시킬 것인지, 내부 흐름의 효율성을 생각할 것인지 등등 본인이 개발할 때 중요하게 생각하는 것이 무엇인지 명확하게 하는 것도 큰 도움이 된다고 들었을 때, 지금까지 내가 개발을 하면서 중요하게 여겼던 점이 무엇인지 생각해 보았다.

그 과정에서 생각났던 부분이 바로 '코드의 간결화'였다. 중복되는 코드를 줄이고, 반복적인 행동을 최소화시키는 것에 초점을 맞추고 싶었다. 이유는 간단하다. '귀찮음' 때문이다.
개발하면서 많은 코드를 작성하게 되는데, 단순 노동 같은 코드는 피로만을 누적시킨다는 것을 동아리 활동을 통해서 경험했기 때문에 이러한 부분에 초점을 맞추는 것 같았다.
그래서 개발을 할 때 DataBinding과 Style을 최대한 이용해 보았다. 이 부분에 대해서는 좋은 평가를 받았고, 좀 더 다듬으면 좋은 코드가 될 거라는 이야기를 들었다.

```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    binding = DataBindingUtil.setContentView(this, R.layout.activity_search)
    binding.activity = this

    initRcv(LAYOUT_CODE)
    initRadio()
}
```

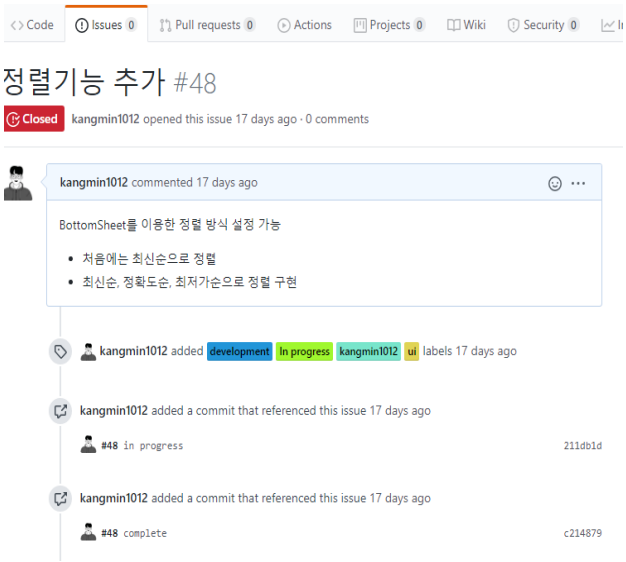
DataBinding을 사용한 모습

```xml
<style name="CustomList">
    <item name="android:textColor">#000000</item>
    <item name="android:textSize">15sp</item>
    <item name="android:textStyle">bold</item>
</style>

<style name="DetailText">
    <item name="android:textColor">#000000</item>
    <item name="android:textSize">25sp</item>
    <item name="android:textStyle">bold</item>
</style>

<style name="RadioButtonPadding">
    <item name="android:padding">10dp</item>
    <item name="android:textStyle">bold</item>
</style>
```

```xml
<TextView
    android:id="@+id/tv_1"
    style="@style/CustomList"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="25dp"
    android:text="최저가 : "
    app:layout_constraintStart_toStartOf="@+id/tv_title"
    app:layout_constraintTop_toBottomOf="@+id/tv_title" />
```

Style 태그를 만들어 반복되는 속성 값 작성 방지

GitHub를 이용하는 법에 대해서도 배웠다. 가장 큰 도움이 되었던 것은 Issue의 사용이었다. 개발을 하는데 있어서 일어나는 모든 일은 Issue가 된다. 내가 모르는 것을 질문하는 것도 Issue고 어떤 기능을 개발하는 것도 Issue다.
이번 행사를 하면서 개발 질문이나 진행 상황은 Issue를 사용했다. 처음 다뤄보는 기능이었기에 사용하는 것에 두려움이 있었지만, 한 번 두 번 사용하다 보니 금방 익숙해져 있는 내 자신을 볼 수 있었다. Issue를 작성하면 나중에 Commit할 때 메시지도 굉장히 수월하게 작성할 수 있었다.



앞으로 안드로이드 공부할 때 무엇을 공부할지에 대한 가이드라인을 알게 된 것 뿐만 아니라, 협업 능력의 심화 및 시야를 넓힐 수 있는 좋은 기회가 되었다. 다음에 참여할 수 있는 기회가 또 생긴다면 지금과는 달리 한 층 더 성장한 모습으로 참여하고 싶다.