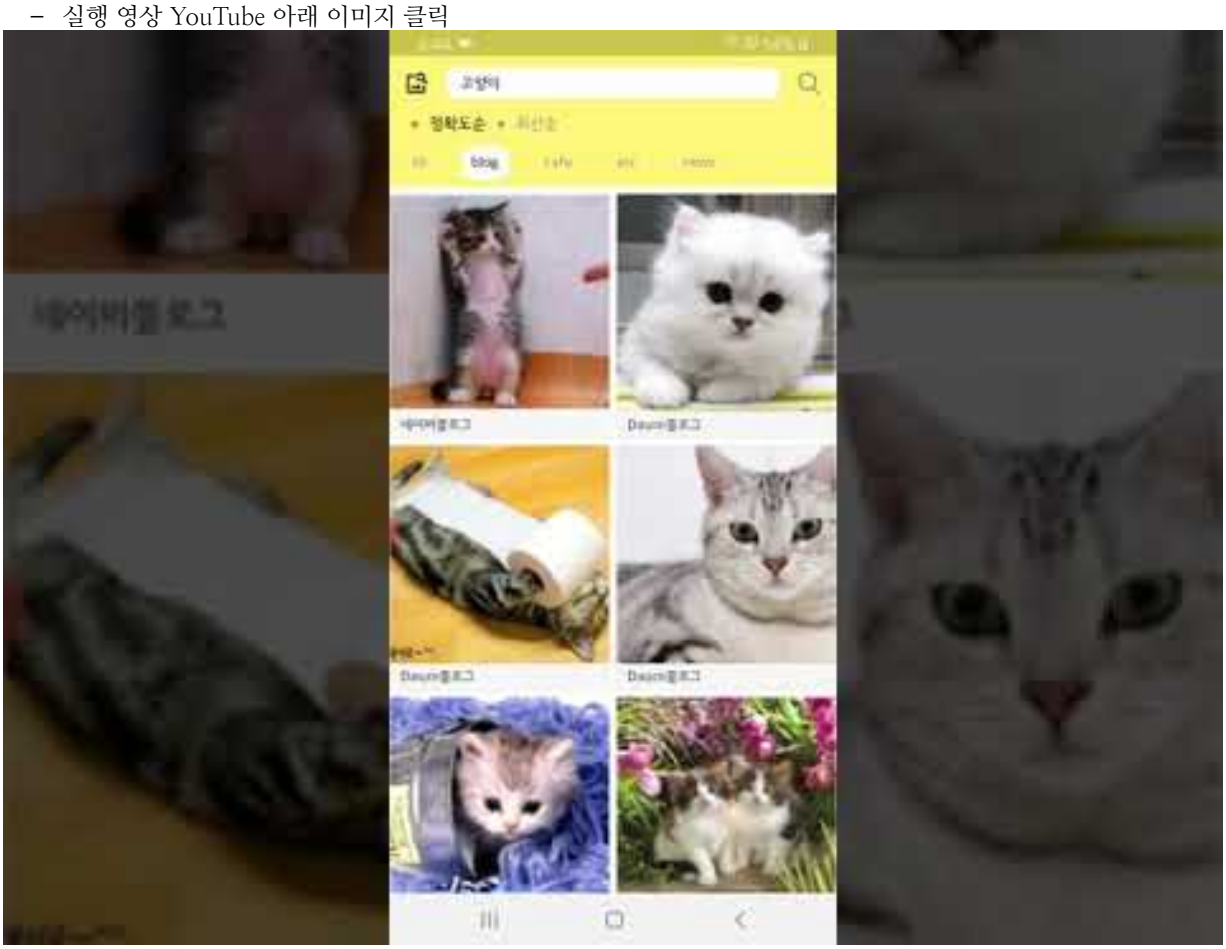
# ImageSearchAndroid

Android application for searching image with daum API.

GitHub URL : https://github.com/meuus90/ImageSearchAndroid

- Product APK 둘러보기
  - APK 링크 Github
  - 실행 영상 YouTube 아래 이미지 클릭



## 목차



## 개발 환경

- 기본 환경
  - JVM target : Java-1.8
  - Kotlin 1.4.10
  - Gradle : 4.0.1
- AndroidX 라이브러리
  - Core 1.3.1

  - Coroutine 1.3.8
  - MultiDex 2.0.1
  - Lifecycle 2.2.0
  - Room 2.3.0-alpha02
  - Paging 3.0.0-alpha06
- 기타 라이브러리
  - Dagger 2.29.1 // Dependency Injection Tool
  - Retrofit 2.9.0 // REST API Tool
  - OkHttp 4.9.0 // HTTP Client Tool
  - Glide 4.11.0 // Image Loading Tool
  - Timber 4.7.1 // Logging Tool
  - Logger 2.2.0 // Logging Tool
- 이미지 출처
  - 앱 아이콘 : 자체 제작
  - 카카오 캐릭터 : 카카오 프렌즈 월페이퍼
  - 기타 아이콘 : MATERIAL DESIGN

## 프로젝트 구성

1. Directory

```
/com/meuus90/imagesearch

  base             ---------> # base package
     arch/util        ---------> # architecture util source
     common/util      ---------> # common util source
     constant         ---------> # constant source
     view             ---------> # custom view source

  di              ---------> # dependency injection modules
  model           ---------> # medel sources
     data/source
        api         ----------> # remote server api
        local       ----------> # local room dao
        repository  ----------> # repository source
     paging       ---------> # paging util
     schema       ---------> # schema data
  viewmodel    ---------> # viewmodel sources
  view         ---------> # view sources
     activity     ---------> # activity source
     dialog       ---------> # dialog source
     fragment     ---------> # fragment source
  ImageSearch   ---------> # main application class
```

2. Architecture Design Pattern and Paging

- 아키텍쳐 디자인 패턴은 MVVM 패턴을 적용하였다.
  - 각 컴포넌트들은 필요시 다른 컴포넌트를 Inject하여 사용하였다. 3. Dependency Injection
- ImageListFragment RecyclerView에 페이징 기능을 적용하였다.
  - AndroidX Paging 3 라이브러리를 사용하였다.
  - Paging 처리 방식은 'Network Storage -> Local Storage -> Repository -> Adapter'로 구성하였다.

3. Dependency Injection

**각 컴포넌트들을 모듈화 하여 컴포넌트간 종속성을 제거하였다.**

**이를 통해 개발 퍼포먼스가 향상되었고 단위 테스트를 수행하기 쉬워졌으며 코드 재사용성이 늘어났다.**

- Fragment를 각각 모듈화 하였고, Activity도 각각 모듈화하여 사용할 Fragment들을 서브모듈로 등록하였다. MainActivityModule

```
@Module
abstract class MainActivityModule {
    @ContributesAndroidInjector(
        modules = [
            IntroFragmentModule::class,
            ImageListFragmentModule::class
        ]
    )
    internal abstract fun contributeMainActivity(): MainActivity
}
```

- AppModule에서는 Application Context, 네트워크 API, Room Database 등을 모듈화하였다. AppModule

```
@Provides
@Singleton
fun appContext(application: Application): Context {
    return application
}

@Provides
@Singleton
fun provideOkHttpClient(interceptor: Interceptor): OkHttpClient {
    val ok = OkHttpClient.Builder()
        .connectTimeout(timeout_connect, TimeUnit.SECONDS)
        .readTimeout(timeout_read, TimeUnit.SECONDS)
        .writeTimeout(timeout_write, TimeUnit.SECONDS)

                .
                .
                .

    ok.addInterceptor(interceptor)
    return ok.build()
}
```

- 생성된 컴포넌트 모듈들은 AppComponent로 등록하여 AppInjector를 통해 Application에 주입하였다. AppComponent

```
@Singleton
@Component(
    modules = [
        AndroidInjectionModule::class,
        AppModule::class,
        MainActivityModule::class
    ]
)

interface AppComponent {
    @Component.Factory
    interface Factory {
        fun create(@BindsInstance app: Application): AppComponent
    }

    fun inject(app: ImageSearch)
}
```

4. CI

- Github Actions Workflow를 이용해 테스트 자동화를 등록하였다. Github Actions
- 주요 기능
  - develop branch에서 commit push 완료시 실행

- JDK 1.8 테스트 환경 셋업
- Kotlin linter 체크
- Android linter 체크
- Test code Unit test 실시

## 화면 구성

1. IntroFragment

- 로컬 캐시 데이터를 초기화한다.
- EditText에 검색어를 입력한 뒤 IME actionSearch 버튼을 클릭하거나 검색 아이콘 클릭 시 이미지 검색 화면으로 이동한다.

2. ImageListFragment

- 디바운싱 검색 기능을 제공한다. 입력창에 텍스트를 입력하고 500ms가 지나면 최종 입력된 문자열로 조회한다.
- 리스트를 아래로 스크롤 시 정해진 페이징 처리방식에 따라 아이템을 자동으로 추가한다.
- appbar_scrolling_view_behavior를 이용하여 AppBar에 RecyclerView 스크롤 이벤트를 적용하였다.
- RecyclerView 아이템은 Glide를 이용하여 thumbnail_url 필드의 이미지 URL을 로딩한다.
- 정확도순이나 최신순(기본 정확도순)에 따른 정렬방식 선택 버튼을 구성하였다.
- 현재 캐싱된 로컬 리스트의 collection 종류를 종합하여 동적으로 추가되는 검색 필터 버튼을 구성하였다.
- 좌측 상단 아이콘을 클릭하면 리스트 최상단으로 이동한다.
- 아이템을 클릭하면 ImageDialog으로 이동한다.

3. ImageDialog

- API 검색으로 받은 이미지를 표현한다.
- 하단 텍스트 클릭 시 브라우저를 열어 링크 페이지로 보낸다.
- Back 버튼을 클릭하거나 좌측 상단 버튼 클릭 시 다이얼로그가 즉시 종료된다.
- Dialog에 이미지를 움직여 다이얼로그를 종료하는 기능을 추가하였다.

## 작업 계획

- [x] 프로젝트 세팅
- [x] 스키마 디자인
- [x] Model 세팅
  - [x] Repository 세팅
  - [x] Room 세팅
  - [x] Paging Data 세팅
- [x] ViewModel 세팅
- [x] Unit Test 테스트코드 작성
- [x] UI 디자인
- [x] API 에러 타입 별 대응
- [x] 애니메이션 등 UX 설정
- [x] 디바이스 퍼포먼스 체크
- [x] UI 테스트 및 기타 버그 픽스
- [x] Release

## License