Career

김현철

안드로이드 개발자입니다.

- 링크드인

경력

Qoo10(큐텐테크놀로지)

> 글로벌 오픈마켓 서비스를 제공

> 모바일그룹/대리

> 2017.06 - 2022.04

상단, 하단 툴바 스크롤 컨트롤러 개발

- 기간 : 2017.10 ~ 2017.11
- 목적 : 앱의 모든 페이지에 상단 헤더와 하단 네비게이션바가 고정되어 있었던 것을 화면을 아래로 스크롤 할 때 헤더와 네비게이션바도 위아래로 스크롤되게 하여 페이지에 더 많은 정보가 노출되도록 하고 답답한 좁은 화면을 개선.
- 개발 내용 : 상,하단 툴바가 스크롤에 따라 움직이게 하기 위해 스크롤 정보를 받아와 툴바를 움직이는 컨트롤러를 생성. 앱에 있는 모든 activity에는 recyclerview, listview, webview 중 하나가 존재하기 때문에 스크롤 정보에 관련된 interface를 생성하여 recyclerview, listview, webview에 해당 interface를 상속받아 각 scroll view에서 스크롤 정보를 전달하는 기능이 추가 된 recyclerview, listview, webview 생성. base activity에서는 컨트롤러를 생성하여 스크롤 정보를 전달하는 interface를 상속 한 view를 컨트롤러에 등록하고 컨트롤러는 각 view에서 touch event를 이용하여 전달된 값을 통해 상,하단 툴바의 움직임을 관리. 툴바는 animation을 이용하여 스크롤 되게 개발. 추가로 상단, 하단 툴바와 함께 움직이는 플로팅뷰가 있을 경우를 위해 플로팅뷰를 추가할 수 있는 기능 개발.

RxJava를 통한 API 호출 개선

- 기간 : 2018.04 ~ 2018.05
- 목적 : 앱이 오래동안 개발되면서 필요한 api들이 점점 늘어나 한 화면에 여러 api를 호출하게되는 상황이 발생하여 화면 갱신이 여러번 발생. 초기에는 api response 받았는지에 대한 Boolean 값을 갖고 모든 값이 true일 경우에 갱신하게 하였는데 코드가 복잡하고 효율이 떨어짐.
- 개발 내용 : 이를 해결할 방법을 찾다가 RxJava를 알게되어 개발을 시작. RxJava의 zip을 이용. 각 api를 호출하는 observable을 생성하여 response 데이터를 zip으로 전달하고 zip 로직에 의해 모든 api의 response가 전달되면 데이터 취합하여 UI에 노출.

생체 인증을 이용한 사용자 정보 저장 및 앱 간의 데이터 공유

- 기간 : 2019.10 ~ 2019-11
- 목적 : 결제 및 비밀 번호 찾기를 할 때 생체 인증을 이용하여 간편하게 찾을 수 있는 기능을 개발. 서비스중인 앱이 여러개 있는데 앱 간에 해당 데이터를 공유하도록 개발

- 개발 내용 : (1) FingerPrintManager를 이용하여 지문 인식을 위한 컨트롤러 개발. KeyGenerator로 받은 받은 key를 이용하여 cipher를 초기화하고 해당 cipher로 CryptoObject를 생성하여 FingerPrintManager를 실행. 인증에 성공하면 서버로 부터 전달받은 데이터를 cipher로 암호화 및 복호화.(디바이스에서 cipher를 초기화 할 때 디바이스에 저장된 인증 정보가 변경되면 InvalidKeyException이 발생해야 하는데 갤럭시의 특정 OS버전에서는 해당 Exception이 발생하지 않아 인증 성공 후 cipher의 block 사이즈를 체크하여 변경 여부 확인) (2) 앱 간의 데이터 공유는 ContentProvider와 ContentResolver를 이용하여 데이터 전달. 앱끼리 서명이 같을 경우 ContentProvider의 permission level을 signature로 설정하면 되지만 서명이 다른 앱이 있어 앱의 key hash를 전달하여 api를 통해 해당 key hash의 인증 여부를 체크 한 뒤 데이터를 전달하고 받을 수 있도록 개발.

Kotlin 전환

- 기간 : 2019.10 ~ 2020.12
- 목적 : 구글 안드로이드 공식 언어 지정 및 간결한 코드와 nullable한 변수의 직관적인 표현으로 null 안전성 향상 등의 이점으로 유지 보수 및 효율성 증대
- 개발 내용 : 100% java로 이루어진 프로젝트를 점진적으로 kotlin 전환. 전환 작업 전에 kotlin에 대한 사전 지식 습득 및 gradle 세팅. Android Studio에서 kotlin 자동 변환을 제공하지만 자동 변환 후에도 전체적으로 변경 된 코드를 검토 해야하고 코딩 스타일이나 kotlin에 대한 학습을 위해서 자동 변환을 사용하지 않고 자체적으로 변환.

열리고 접히는 expandable recyclerview adapter 개발

- 기간 : 2020.02 ~ 2020.02
- 목적 : 카테고리나 메뉴 개발을 하다보면 타이틀을 선택했을 때 서브 리스트가 노출되고 사라지는 동작이 종종 있었는데 여러 사람이 개발을 하다보니 코드가 통일되지 않고 복잡하게 개발되어 있어서 유지보수하기에 어려움을 느껴 여러 화면에서 동일하게 사용할 수 있는 adapter 개발
- 개발 내용 : (1) Parent(타이틀)와 Child(서브 리스트)를 저장하는 Group Item class를 생성하고 이 Group Item을 Array로 하는 Group List class 생성. Group List class 에서는 Parent의 open, close에 대한 상태를 저장하는 Array를 Parent의 크기로 생성하고 adpater로 item count와 view type을 전달한 함수 생성. item count는 {(총 parent의 수) + (Boolean Array에서 값이 true인 Parent의 Child 수)}를 전달. view type은 전달받은 position이 Parent인지 Child인지 확인하여 Parent 또는 Child view type을 전달. adapter는 Group List class를 생성하여 전달 받은 데이터를 Group List class로 변환. onCreateViewHolder와 onBindViewHolder에는 Parent, Child view type별로 Parent ViewHolder와 Child ViewHolder를 적용. Parent ViewHolder는 itemView에 onClickListener를 생성하여 abstract 함수를 호출하도록 생성하고 custom adapter에서는 Parent가 눌렸을 때 Group List class의 Parent의 상태 값이 토글되도록 구현. (2) 만들어 놓은 Group List class와 adapter를 이용하여 토글 동작 로직을 구현 할 Controller class를 생성. Group List를 전달 받아 Controller에서 Parent 상태 값을 바꿔주고 listener를 이용하여 adapter에 열렸는지 접혔는지와 해당 position 및 child count를 전달하여 adapter는 notifyItemRangeInserted 또는 notifyItemRangeRemoved를 호출하여 화면 갱신. (3) 구현된 expandable adapter를 상속받은 adapter는 전달받은 data를 Group Item으로 묶어서 expandable adapter에 전달해주고 타이틀 및 서브 리스트의 ViewHolder는 Parent ViewHolder, Child ViewHolder를 상속받아 사용하면 여러 화면에서 공통적으로 사용할 수 있음.

Retrofit2 적용

- 기간 : 2020.05 ~ 2020.05
- 목적 : 기존에 api 통신으로 사용하던 Volley에서 Retrofit2로 전환하기 위한 모듈 생성
- 개발 내용 : Interceptor를 상속받은 Custom Interceptor 생성하여 request에 header 정보 추가 및 response 데이터의 gzip 디코딩 로직 추가. 서버에서 내려주는 데이터가 특정 키에 묶여 내려오고 있어서 특정 키의 내용을 풀어 response 타입으로 변환하여 전달하기 위한 custom converter factory 생성하여 위의 로직 구현

동영상 플레이어 개발

- 기간 : 2022.01 ~ 2022.02
- 목적 : 상품 리스트 중에 동영상 정보가 있는 상품일 경우 등록 된 동영상을 보여주기 위함
- 개발 내용 : 일반 영상과 유튜브 영상 두 종류를 재생해야 해서 ExoPlayer와 YouTubePlayer를 사용하고 이를 통합으로 관리하기 위한 Player를 생성. 동영상 타입에 따라 적절한 PlayerView를 노출해서 동영상에 맞는 Player가 노출될 수 있도록 개발. YouTubePlayer의 경우 Android SDK 버전 관리가 오랫동안 되지 않아 서비스 중인 앱에 적용하기 어려워 IFrame API를 사용하여 html 파일로 YouTubePlayer 화면을 만들고 앱과 통신할 인터페이스를 추가하여 개발. ExoPlayer와 YouTubePlayer의 상태 변화, 에러 핸들링 등의 공통적인 이벤트를 하나의 Listener로 통합. 각 Player에 내장된 컨트롤 UI를 사용하지 않고 custom UI로 만들고 화면 터치나 일정 시간에 따른 UI 노출을 제어하기 위한 컨트롤러를 개발. 영상 위치에 따라 자동 재생 될 수 있는 기능을 위해 컨트롤러 개발. RecyclerView를 파라미터로 전달받아 스크롤 Listener를 추가해서 스크롤 상태가 IDLE일 경우에 화면 중앙과 가장 가까운 child view를 찾고 해당 child가 동영상을 재생할 수 있는 경우 child에서 동영상을 재상하기 위한 컨테이너를 가져와 PlayerView를 컨테이너에 추가하고 화면에서 벗어나거나(detach) 다른 child에 영상을 재생해야 될 경우 컨테이너에서 PlayerView를 제거하고 다른 child에 추가하는 방식으로 동작. child의 경우에는 동영상 재생 기능 구현을 위해 Player 인터페이스를 상속받아 구현하도록 했고 컨트롤러에서도 child가 동영상 재생할 수 있는지 판단하기 위해 Player 인터페이스를 상속받은 child인지 확인하여 판단. 또한, 컨트롤러는 LifecycleObserver를 상속받고 있고 Lifecycle을 파라미터로 전달받아 Lifecycle에 컨트롤러를 추가하여 Lifecycle에 따라 Player pause, release 등의 동작 관리. ExoPlayer의 경우 파라미터로 전달받아 단일 player를 사용하도록 개발.

컬쳐히어로

> 우리의 식탁이라는 음식 레시피 및 커머스 서비스 제공

> 앱개발셀/스탭

> 2022.04 - 2024.04

레거시 코드 개선

- 기간 : 2022.04 ~
- 목적 : deprecate 된 라이브러리, api 등을 변경하고 일부 동일 한 use-case 들이 공통 모듈로써 관리되지 않는 부분들 일부 수정
- 개발 내용 : (1) 앱링크, 푸시, 웹, 서버 data 등에서 화면 이동을 위한 link, link type이 정의되어 있는데 해당 데이터를 핸들링 하는 로직이 화면마다 정의되어 있어서 하나의 class로 정의 함. (2) ButterKnife 사용 중인 부분을 ViewBinding, DataBinding으로 변경 (3) 사용중인 ImageLoader 모듈에서 Glide를 이용하는데 이미지 로드가 완료되면 Bitmap으로 저장해서 특정 비율로 변경하는 등의 로직이 있어서 메모리 이슈가 있었음. Bitmap 저장하는 부분 삭제하고 Glide api를 이용하여 Image 비율 변경 할 수 있도록 수정. (4) module로 import 된 라이브러리를 maven으로 변경 (5) kotlin 코드 적용 (6) JetPack ViewModel 적용 (7) Jcenter(종료) 사용중인 library maven 전환 또는 jar(or aar) 모듈로 추가

외부 라이브러리 연동

- 기간 : 2022.05 ~ 2022-06
- 목적 : logging, 문의, 공유 링크 생성 등의 기능 구현을 위해 외부 라이브러리 연동
- 개발 내용 : (1) Amplitude SDK 연동 (2) Branch를 이용하여 링크 생성 (3) 채널톡 SDK 연동.

본인 인증 화면 개발

- 기간 : 2022.07 ~ 2022.08
- 목적 : 본인 인증을 위한 화면 개발
- 개발 내용 : 본인 인증 url에 사용자 정보를 파라미터로 추가하여 웹뷰로 로드. WebViewClient에서 shouldOverrideLoading override하여 url의 parameter를 통해 성공 여부와 data를 추출. 이를 이용하여 서버와 api 통신을 통해 인증 결과를 전달하고 setResult를 통하여 인증 화면을 호출한 화면으로 결과 전달.

난독화 적용

- 기간 : 2022.08 ~ 2022.08
- 목적 : 난독화 적용
- 개발 내용 : 사용 중인 외부 Library 확인하여 proguard rule 추가. crashlytics mapping file upload 추가

사용자 이벤트 전송 모듈

- 목적 : Firebase Analytics, Branch, Braze를 사용하여 사용자 이벤트를 로깅하고 있었는데 필요한 위치에서 매번 중복되는 코드들을 작성하고 각 라이브러리마다 코드를 작성해서 코드가 길어지고 복잡해짐. 이를 개선하고자 함.
- 개발 내용 : EventLogger라는 클래스를 만들어 이벤트 name, properties, platform(firebase, branch, braze)을 전달 받아 각 platform에 맞게 변형하여 이벤트 전달하는 기능 구현. 프로젝트에 적용 된 모든 이벤트 전송 코드를 변경하여 코드를 단순화 함.

검색 화면 개발

- 기간 : 2022.11 ~ 2022.12
- 목적 : 검색 화면 리뉴얼
- 개발 내용 : 홈, 입력, 결과 화면으로 구성. 홈 화면에서는 최근 검색한 키워드, 인기 키워드, 추천 키워드를 노출. 입력 화면에서는 EditText에 TextWatcher를 추가하여 변경 되는 text를 flow로 변환하고 debounce를 이용하여 자동 완성 api 호출하도록 구현. api 응답 받은 data는 입력 된 키워드와 비교하여 겹치는 부분의 색상은 변경하도록 구현. 검색 결과 화면에서는 5개의 tab을 ViewPager2와 TabLayout으로 구현. 각 화면에는 UiState를 구현해서 flow로 collect하고 화면에서 handling하도록 구현.

HashTag Editor 개발

- 기간 : 2023.03 ~ 2023.04
- 목적 : 게시물 작성 시 해시태그 추가 될 경우 연관 된 해시태그 목록 노출 및 게시물에서 HashTag 색상 표현
- 개발 내용 : TextWatcher를 이용하여 입력 된 text가 해시태그(#)인지 확인하고 해시태그일 경우 해당 text를 listener를 통해 전달. listener를 구현한 곳에서는 전달받은 text를 api를 통해서 연관 태그를 요청하고 화면에 노출하도록 구현. 해시태그 검색 로직은 '#' 체크하고 공백이 나올 때 까지를 해시태그로 판단.

Image Picker 개발

- 기간 : 2023.05 ~ 2023.06
- 목적 : 커스텀한 이미지 피커에 대한 니즈가 있어서 Image Picker 개발
- 개발 내용 : Activity에 이미지 리스트, 앨범 리스트, 프리뷰 화면 Fragment를 Jetpack Navigation을 이용하여 구현. ContentResolver로 이미지에 대한 Cursor를 생성하여 이미지 정보를 받아 온 뒤 이미지들을 Bucket 이름별로 그룹화해서 전달하는 Context 확장 함수를 구현. 이미지 리스트 화면에서는 선택 된 Bucket에 포함 된 이미지들을 grid list 형태로 노출하도록 구현. 앨범 리스트에서는 Bucket 리스트를 노출하고 Bucket을 선택하면 이미지 리스트 화면으로 돌아가서 선택 된 Bucket의 이미지들을 노출하도록 구현.

SSE를 이용한 AI Recipe

- 기간 : 2023.08 ~ 2023.09
- 목적 : AI 레피시 기능을 제공하기 위해 개발
- 개발 내용 : 자사 서버에서 ChatGPT를 base로 하는 3rd-party 라이브러리를 이용하여 recipe 생성하는 기능을 제공. 클라이언트에서는 SSE를 이용하여 자사 서버와 연결하고 전달 받은 데이터를 파싱하여 노출하도록 구현. 데이터는 서버-클라이언트 간에 정의 된 tag를 사용한 xml 양식으로 전달되어 클라이언트에서 각 tag 맞는 양식으로 UI 노출하도록 구현. 각각 SSE를 통해 입력을 받는 class, 입력에서 tag를 찾아 tag 시작, 종료, 컨텐츠를 확인하는 class, 각 tag에 맞는 View를 생성하여 컨텐츠를 입력하고 노출하는 class를 생성하여 구현. SSE 연동하는 Repository에서 서버로부터 데이터가 전달 되면 tag 시작, 입력, 종료 state와 tag type 전달. 입력일 경우 입력 된 data 전달. 전달되는 data들은 flow로 방출되고 collect하는 곳에서는 전달받은 data를 View 생성하는 class로 전달하여 화면에 노출하도록 구현.

JWT 인증 로직 개발

- 기간 : 2023.10 ~ 2023-11
- 목적 : JWT 사용하여 보안 향상
- 개발 내용 : OkHttpClient Interceptor를 이용하여 인증 로직 구현. Interceptor에서 response의 code가 401일 경우 토큰 검증 및 갱신 로직을 실행. 해당 로직은 synchronized로 동기화하며 request전에 사용한 Access Token 값과 현재 클라이언트에서 저장중인 Access Token값을 비교하여 다를 경우 다른 request에서 갱신이 완료됐다고 판단하고 해당 Access Token을 사용하여 다시 request, 같은 경우 Access Token을 갱신하는 request를 실행하고 전달받은 Access Token을 이용하여 원래 request를 다시 실행하도록 구현.

EventBus Flow

- 기간 : 2024.02 - 2024.03
- 목적 : EventBus와 비슷한 기능을 위한 flow 구현
- 개발 내용 : Event를 seal class로 구현해서 필요한 Event들을 정의. SharedFlow를 사용하여 Event를 방출하기 위한 flow 생성. Fragment, ActivityFragment의 확장 함수로 Event를 수집하는 inline fun 생성하여 해당 화면의 lifecycle scope를 이용하여 Event collect 하도록 구현. flow로 emit하기 위한 scope는 Default로 지정하여 해당 scope에서 emit 실행.