Android

Android Studio 다운로드 - 설치
Do not Import - Don't Send 선택

3개 항목 Accept - Finish - SDK Install이 되는 중

Activity - 화면의 UI를 의미

Kotlin Anonymous Type 서포팅해줌

<새 프로젝트 생성>

```
package com.example.myapplication;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;

public class MainActivity extends AppCompatActivity {

@Override
protected void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  setContentView(R.layout.activity_main);
  }
}
```

Bundle - 어떤 값을 가지고 있는 객체라는 의미
R = Resource

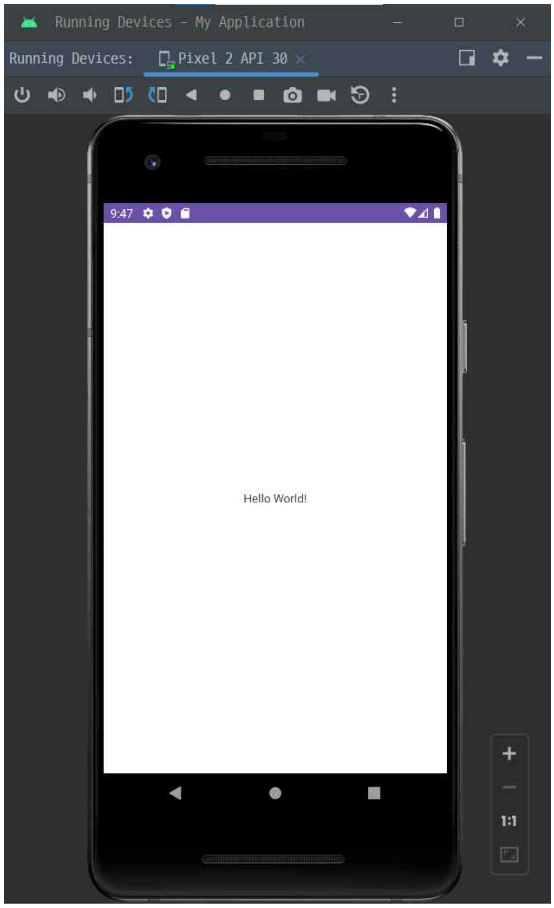
Android - res - activtity_main.xml
파란선 - 가이드 라인

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:app="http://schemas.android.com/apk/res-auto"
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="match_parent"
tools:context=".MainActivity">

<TextView
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Hello World!"
app:layout_constraintBottom_toBottomOf="parent"
app:layout_constraintEnd_toEndOf="parent"
app:layout_constraintStart_toStartOf="parent"
app:layout_constraintTop_toTopOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

API(디바이스) 다운로드
Shift+f10 app 구동하기



Settings - language(ko) - 한국어 설정

<Toast와 Intent>
Toast란?
Toast는 안드로이드 앱에서 간단한 메시지를 사용자에게 잠시 보여주기 위해 사용되는 팝업 형태의 알림입니다. 일반적으로 짧은 시간 동안 화면 상단 또는 하단에 나타나며, 사용자에게 간단한 정보나 액션 결과를 안내하는 용도로 주로 사용됩니다.

Intent란??
Intent(인텐트)는 안드로이드에서 앱 구성 요소 간에 통신하고 상호 작용하는 데 사용되는 메시지 객체입니다. 액티비티(Activity), 서비스(Service), 브로드캐스트 리시버(Broadcast Receiver), 콘텐트 프로바이더(Content Provider) 등 안드로이드의 주요 구성 요소들은 Intent를 통해 다른 구성 요소와 상호 작용하며, 이를 통해 앱 내부나 외부의 다른 앱과도 상호작용할 수 있습니다.

Intent는 다음과 같은 목적으로 사용될 수 있습니다:

액티비티 간 데이터 전달: 액티비티 간에 데이터를 주고받기 위해 Intent를 사용합니다. 예를 들어, 한 액티비티에서 다른 액티비티로 사용자 정보를 전달하거나, 다른 액티비티를 호출하면서 필요한 데이터를 넘겨주는 등의 상황에 사용됩니다.

서비스 호출: 앱 구성 요소가 서비스를 시작하거나 종료할 때 Intent를 사용합니다. 이를 통해 백그라운드에서 작업을 처리하는 서비스를 호출하거나 조작할 수 있습니다.

브로드캐스트: 앱 구성 요소 간에 브로드캐스트 메시지를 보내거나 수신할 때 Intent를 사용합니다. 이를 통해 시스템 또는 다른 앱에서 발생한 이벤트를 수신하고 처리할 수 있습니다.

외부 액티비티 호출: 다른 앱의 액티비티를 호출하거나 외부 앱과의 상호작용을 위해 Intent를 사

용할 수 있습니다. 예를 들어, 전화 걸기, 웹 페이지 열기, 지도 앱 호출 등이 있습니다.

Intent는 명시적 인텐트(Explicit Intent)와 암시적 인텐트(Implicit Intent)로 나눌 수 있습니다. 명시적 인텐트는 목적지 구성 요소의 클래스명을 직접 지정하여 호출하는 방식이며, 암시적 인텐트는 수행하려는 작업과 관련된 액션(action)과 데이터를 지정하여 시스템에 맞는 액티비티를 찾아 호출하는 방식입니다.

안드로이드 앱에서 Intent를 사용하여 구성 요소 간에 효율적으로 상호작용하여 다양한 기능을 구현할 수 있습니다.

<Android UI 구현 - 레이아웃>

LinearLayout: 뷰들을 수평 또는 수직으로 일렬로 배치하는 레이아웃입니다. android:orientation 속성으로 수평(LinearLayout.HORIZONTAL) 또는 수직(LinearLayout.VERTICAL)으로 설정할 수 있습니다.

RelativeLayout: 뷰들을 상대적인 위치로 배치하는 레이아웃입니다. 뷰 간의 상대적인 위치와 규칙(RelativeRule)을 설정하여 UI를 구성할 수 있습니다.

FrameLayout: 뷰들을 겹치게 배치하는 레이아웃입니다. 여러 개의 뷰가 겹쳐져서 표시되며, Z축 방향으로 순서에 따라 표시됩니다.

ConstraintLayout: 제약 조건을 사용하여 뷰들을 배치하는 레이아웃입니다. 뷰들 간의 연결 및 제약 조건을 설정하여 유연하고 복잡한 UI를 구성할 수 있습니다.

GridLayout: 격자 형태로 뷰들을 배치하는 레이아웃입니다. 행과 열을 정의하여 뷰를 배열할 수 있습니다.

CoordinatorLayout: 자식 뷰들 사이의 상호작용을 처리하고 제어하는 레이아웃입니다. 스와이프, 스크롤, 투명도 등과 같은 동작을 지원합니다.

TableLayout: 테이블 형태로 뷰들을 배치하는 레이아웃입니다. 행과 열을 사용하여 뷰를 테이블로 배치할 수 있습니다.

ScrollView: 스크롤 가능한 단일 뷰를 포함하는 레이아웃입니다. 스크롤이 가능한 UI를 구성할 때 사용됩니다.

DrawerLayout: 네비게이션 드로어와 함께 사용하여 사용자 인터페이스를 구성하는 레이아웃입니다.

바인딩
"Binding"은 안드로이드에서 UI 요소와 데이터를 서로 연결하는 메커니즘을 말합니다. 이를 통해 앱의 UI 요소와 데이터 모델을 간편하게 동기화하고 상호작용할 수 있습니다. 바인딩은 앱의 코드에서 XML 레이아웃 파일과 데이터 모델을 더욱 효율적으로 연결하여 UI 업데이트를 쉽게 할 수 있도록 도와줍니다.

안드로이드에서 바인딩에는 두 가지 유형이 있습니다:

데이터 바인딩(Data Binding): 데이터 바인딩은 UI 컴포넌트를 XML 레이아웃과 데이터 모델로 바인딩하는 기술입니다. 이를 통해 UI 요소와 데이터를 뷰 모델 또는 데이터 클래스와 결합시켜 데이터 변경 시 자동으로 UI가 업데이트되도록 할 수 있습니다. 데이터 바인딩을 사용하면 findViewById()를 통해 UI 요소를 찾는 불필요한 코드를 줄일 수 있으며, 더 직관적이고 편리한 방식으로 데이터를 UI에 바인딩할 수 있습니다.

뷰 바인딩(View Binding): 뷰 바인딩은 데이터 바인딩과는 다르게 XML 레이아웃의 뷰 요소를 바인딩합니다. 뷰 바인딩은 findViewById() 대신 각 뷰 요소에 대한 바인딩 클래스를 생성하고 이를 통해 뷰 요소를 쉽게 참조할 수 있도록 도와줍니다. 이는 뷰 요소의 ID를 직접 참조하여 UI 요소를 변경할 때 코드의 안전성을 높여주고, 불필요한 널 체크를 줄여줍니다.

두 바인딩 유형 모두 개발자가 UI 요소를 코드와 더 효율적으로 연결하여 안드로이드 앱 개발을 더욱 편리하고 유지 보수가 용이하게 만들어줍니다. 바인딩은 안드로이드 스튜디오 버전 2.1부터 도입되었으며, 개발자들 사이에서 매우 유용한 도구로 자리 잡았습니다.

Widget

안드로이드에서 위젯(widget)은 런처 또는 잠금 화면 같은 Android 표시 경로에 삽입할 수 있는 소형 애플리케이션 뷰이다. 위젯은 사용자가 자주 사용하는 정보를 빠르게 확인하고 작업을 수행할 수 있도록 도와준다.

AndroidManifest.xml 파일에 위젯을 등록한다.
위젯의 레이아웃과 위젯의 업데이트 주기를 설정한다.
위젯을 등록하려면 AndroidManifest.xml에 추가한다.

자세한 내용은 소스코드 참조

Binding - Gradle에 활성화

```
android {
    ...
    buildFeatures {
        dataBinding true
    }
}
```

Widget UI 구현 하기
<TabHost>
TabHost는 안드로이드에서 탭 기반의 사용자 인터페이스를 구현하기 위해 사용되는 클래스입니다. 탭은 일반적으로 화면 상단 또는 하단에 배치되며, 사용자는 탭을 선택하여 다른 화면이나 콘텐츠를 보거나 앱의 다른 섹션으로 이동할 수 있습니다. 이를 통해 앱의 다양한 기능이나 정보를 구분하여 표시하고 사용자 경험을 향상시키는 데 활용됩니다.

TabHost는 탭을 호스트하는 컨테이너 역할을 하며, 내부에 여러 개의 탭을 추가할 수 있습니다. 각 탭은 TabSpec라고 불리는 객체를 사용하여 정의하고, 해당 탭을 클릭했을 때 표시할 콘텐츠나 레이아웃을 지정합니다.

Theme 변경 - Manifest(AndroidManifest.xml)에서 변경

```
android:theme="@style/Theme.AppCompat.DayNight.DarkActionBar"
```

<ResiterForContextMenu>

```
registerForContextMenu
```

안드로이드 액티비티나 뷰(View)에서 컨텍스트 메뉴(Context Menu)를 사용할 수 있도록 등록하는 메서드입니다. 컨텍스트 메뉴는 사용자가 오랜 클릭(롱 클릭) 또는 뷰를 오래 누르고 있을 때 해당 뷰에 대한 컨텍스트 메뉴가 표시되는 기능을 제공합니다.

<MenuInflater>
MenuInflater는 안드로이드에서 메뉴를 Inflate(생성)하는 데 사용되는 클래스입니다. 메뉴 인플레이터를 사용하면 XML 리소스로 정의된 메뉴를 자바 객체로 변환하여 Activity나 Fragment의 메뉴에 추가할 수 있습니다.

<날짜 및 시간 선택>

RatingBar 활용 및 ImageView 활용

<위치 가져오기 - 지도 맵 구현>

Manifest에서
android.permission.ACCESS_COARSE_LOCATION
android.permission.ACCESS_FINE_LOCATION
사용

<Intent>
명시적 인텐트, 암시적 인텐트

<Activity의 생명주기>

<ListView와 GridView>

<RecylerView - Adapter View>

Interface 활용 Adapter View 만들기

Fragment와 SQLite 개요 및 활용

<Fragment>
프래그먼트(Fragment)는 안드로이드 애플리케이션의 사용자 인터페이스 일부를 나타내는 조각입니다. 프래그먼트는 화면의 작은 부분이나 모듈화된 UI 요소를 나타내며, 액티비티 내에서 여러 개의 프래그먼트를 조합하여 다양한 레이아웃 및 UI를 만들 수 있습니다. 이를 통해 액티비티의 기능을 모듈화하고 재사용성을 높일 수 있습니다.

프래그먼트는 주로 다음과 같은 목적으로 사용됩니다:

유연한 UI 디자인: 프래그먼트를 사용하여 다양한 디바이스의 화면 크기에 대응하거나, 가로 및 세로 방향 전환 시 UI를 효율적으로 관리할 수 있습니다.

모듈화와 재사용성: 여러 액티비티에서 같은 UI 요소를 사용해야 할 때 프래그먼트를 생성하여 재사용하거나 모듈화할 수 있습니다.

다중 작업 및 멀티태스킹: 한 액티비티 내에서 여러 프래그먼트를 사용하여 다중 작업을 수행하거나, 태블릿 등 대형 화면에서 멀티태스킹을 구현할 수 있습니다.

유지보수 및 업데이트 용이성: 각각의 프래그먼트를 독립적으로 관리하므로, 하나의 프래그먼트를 수정해도 다른 부분에 영향을 주지 않습니다.

프래그먼트는 액티비티 내에서 호스팅되며, 각 프래그먼트는 자체 생명주기를 가지고 있습니다. 프래그먼트의 생명주기는 액티비티의 생명주기와 연결되어 있으며, 액티비티와 유사한 메서드들(onCreate, onStart, onResume 등)을 갖습니다.

프래그먼트를 사용하는 데에는 두 가지 주요 방식이 있습니다:

정적 프래그먼트: XML 레이아웃에서 <fragment> 태그를 사용하여 정적으로 프래그먼트를 선언하고 사용합니다.

동적 프래그먼트: 런타임 중에 액티비티 내에서 프래그먼트를 추가, 제거, 교체하거나 상호작용할 수 있습니다. 이를 위해서 FragmentManager를 사용합니다.

프래그먼트를 사용하면 액티비티의 코드를 모듈화하고 복잡한 UI를 관리하는 데 도움이 됩니다. 이는 대규모 애플리케이션에서 특히 유용합니다.

<SQLite>
SQLite는 경량의 관계형 데이터베이스 관리 시스템(RDBMS)입니다. C 라이브러리로 제공되며, 서버가 필요하지 않고 파일 형태로 데이터를 저장하고 관리합니다. SQLite는 소스 코드가 오픈되어 있어서 무료로 사용할 수 있으며, 다양한 플랫폼에서 사용할 수 있는 이식성이 높은 특징을 가지고 있습니다.

주요 특징과 장점:

경량성: SQLite는 작은 라이브러리로서 빠르고 경량입니다. 특히 메모리 사용이 적고, 별도의 서버나 프로세스를 필요로 하지 않습니다.

포터빌리티: SQLite는 다양한 플랫폼에서 사용할 수 있습니다. 데스크탑, 서버, 모바일 디바이스, 임베디드 시스템 등에서 동일한 방식으로 데이터를 다룰 수 있습니다.

자가 포함: SQLite는 데이터베이스 파일 하나에 모든 데이터를 저장합니다. 따라서 별도의 데이터베이스 서버가 필요 없으며, 데이터베이스 파일을 다른 시스템으로 옮겨도 작동합니다.

트랜잭션 지원: SQLite는 ACID(원자성, 일관성, 고립성, 지속성) 트랜잭션을 지원합니다. 이를 통해 데이터의 무결성과 안전성을 보장할 수 있습니다.

SQL 호환성: SQLite는 SQL 데이터베이스 엔진이므로, 대부분의 SQL 명령문을 사용하여 데이터를 조작할 수 있습니다.

오픈 소스: SQLite는 오픈 소스 라이브러리로서, 소스 코드가 공개되어 있습니다. 이는 사용자가 내부 동작을 이해하고 수정할 수 있음을 의미합니다.

SQLite는 주로 모바일 애플리케이션과 임베디드 시스템에서 많이 사용됩니다. 안드로이드 애플리케이션에서도 내장 데이터베이스로서 널리 활용되며, 작은 규모의 프로젝트나 간단한 데이터 저장 용도에 적합합니다. 하지만 대량의 데이터를 다루거나 복잡한 트랜잭션 처리가 필요한 경우에는 다른 데이터베이스 시스템을 고려해야 할 수 있습니다.

<네비게이션 뷰>

NavigationView는 Android의 Material Design 구성 요소 중 하나로, 네비게이션 메뉴를 구현하는 데 사용됩니다. 네비게이션 메뉴는 사용자가 앱 내에서 다양한 섹션 또는 기능으로 이동할 수 있도록 도와주는 중요한 요소입니다. 다양한 종류의 NavigationView가 있으며, 주로 다음과 같이 사용됩니다:

Basic NavigationView: 기본 네비게이션 메뉴로, 드로어 형태로 화면 왼쪽에서 나타나는 구성 요소입니다. 사용자는 메뉴 항목을 선택하여 다른 화면 또는 기능으로 이동할 수 있습니다.

BottomNavigationView: 화면 하단에 위치하는 네비게이션 메뉴로, 주로 탭 형태로 사용됩니다. 사용자는 하단의 탭을 선택하여 다른 섹션 또는 화면으로 빠르게 이동할 수 있습니다.

NavigationView with DrawerLayout: 네비게이션 메뉴를 드로어 레이아웃(DrawerLayout)과 함께 사용하는 방식입니다. 사용자가 드로어를 열면 메뉴 항목이 나타나고, 선택하면 해당 화면으로 이동하거나 기능을 수행할 수 있습니다.

NavigationView with Tabs: 탭 레이아웃(TabLayout)과 함께 사용되는 네비게이션 메뉴입니다. 각 탭은 다른 섹션 또는 화면을 나타내며, 사용자는 탭을 선택하여 해당 섹션으로 이동할 수 있습니다.

NavigationView with ViewPager: ViewPager와 함께 사용되는 네비게이션 메뉴입니다. ViewPager 내부에 다양한 페이지 또는 화면이 있고, 사용자는 네비게이션 메뉴를 통해 페이지 간에 이동할 수 있습니다.

<스레드>

스레드(Thread)는 프로그램 내에서 실행되는 실행 단위로, 프로세스 내에서 작업을 분리하거나 병렬로 실행하기 위해 사용됩니다. 스레드는 프로세스 내에서 독립적으로 실행될 수 있는 작은 코드 블록이며, 하나의 프로세스 내에 여러 개의 스레드를 생성하여 다양한 작업을 동시에 수행할 수 있습니다.

스레드는 프로세스의 자원을 공유하면서 실행되기 때문에, 하나의 프로세스 내에서 여러 스레드를 사용하면 작업의 효율성을 높일 수 있습니다. 스레드를 사용하는 가장 큰 이점은 병렬성(Parallelism)과 응답성(Responsiveness)입니다. 다음은 스레드의 주요 특징과 장점을 요약한 것입니다:

병렬성과 동시성: 스레드를 사용하면 여러 작업을 동시에 실행하여 병렬 처리가 가능합니다. 이로 인해 작업의 처리 시간을 단축하거나 여러 작업을 동시에 수행할 수 있습니다.

응답성: 스레드를 사용하면 사용자 인터페이스(UI)가 더 반응적으로 동작할 수 있습니다. 긴 작업을 수행하는 동안에도 다른 스레드가 UI 이벤트를 처리하면 앱이 더 빠르게 반응하게 됩니다.

효율적인 자원 공유: 하나의 프로세스 내에서 스레드는 메모리 공간 등의 자원을 공유하기 때문에, 데이터를 효율적으로 공유하고 통신할 수 있습니다.

경제성: 프로세스 생성에 비해 스레드 생성이 비교적 적은 시간과 자원을 요구합니다.

멀티코어 활용: 멀티코어 프로세서를 활용하여 병렬로 작업을 처리할 수 있어 성능을 향상시킬 수 있습니다.

그러나 스레드를 사용할 때 주의해야 할 점도 있습니다:

동기화 문제: 여러 스레드가 공유 자원에 동시에 접근할 때 데이터의 일관성과 안정성을 유지하기 위해 동기화 메커니즘이 필요합니다.

경쟁 조건: 스레드 간의 실행 순서가 바뀔 수 있어 예상치 못한 결과가 발생할 수 있습니다.

데드락과 교착상태: 두 스레드가 서로의 작업을 기다리면서 진행이 멈추는 상태를 말합니다.

컨텍스트 스위칭 오버헤드: 스레드 간의 전환(컨텍스트 스위칭)에 오버헤드가 발생할 수 있습니다.

스레드는 프로그래밍 언어와 플랫폼에 따라 다르게 지원되며, 안드로이드 앱 개발에서도 스레드를 사용하여 UI 업데이트, 백그라운드 작업 등 다양한 작업을 처리할 수 있습니다.

안드로이드 앱에서는 스레드(Thread)를 사용하여 백그라운드에서 작업을 수행하거나 UI의 응답성을 향상시키는 등 다양한 작업을 처리할 수 있습니다. 그러나 중요한 점은 메인(UI) 스레드와 백그라운드 스레드 간의 상호작용과 동기화에 주의해야 합니다.

다음은 안드로이드에서의 스레드 사용 방법과 주의사항에 대한 간략한 설명입니다:

메인(UI) 스레드: 안드로이드 앱의 UI는 메인 스레드에서 동작합니다. UI 업데이트 및 사용자 인터페이스와 관련된 작업은 반드시 메인 스레드에서 실행되어야 합니다. 메인 스레드는 앱의 응답성을 유지하기 위해 중요합니다. 긴 시간 동안 실행되는 작업을 메인 스레드에서 실행하면 앱이 느려지거나 응답하지 않을 수 있습니다.

백그라운드 스레드: 메인 스레드 외에도 백그라운드 스레드를 사용하여 오랫동안 실행되는 작업을 처리할 수 있습니다. 네트워크 통신, 파일 다운로드, 데이터 처리 등의 작업을 백그라운드 스레드에서 실행하여 메인 스레드의 부담을 줄일 수 있습니다.

AsyncTask: 안드로이드에서 가장 간단한 백그라운드 스레드 사용 방법 중 하나로, UI 업데이트를 처리하기 쉬운 AsyncTask 클래스를 제공합니다. AsyncTask는 작업을 세분화하여 백그라운드에서 실행하고, 메인 스레드에서 UI 업데이트를 할 수 있도록 지원합니다.

Handler 및 Looper: Handler 클래스를 사용하여 메인 스레드와 백그라운드 스레드 간의 메시지를 주고받을 수 있습니다. Looper는 메인 스레드의 메시지 큐를 처리하는 루프를 제공합니다.

Thread 클래스 및 Runnable 인터페이스: Java의 기본적인 스레드 관련 클래스와 인터페이스를 활용하여 백그라운드 스레드를 생성하고 실행할 수 있습니다.

HandlerThread: HandlerThread는 백그라운드 스레드와 Handler를 결합한 클래스로, 스레드와 메시지 처리를 편리하게 관리할 수 있습니다.

Executor 및 AsyncTask 대체: 안드로이드에서는 AsyncTask의 사용을 권장하지 않으며, 대신 Executor, ThreadPoolExecutor, AsyncTaskCompat 등을 사용하여 백그라운드 작업을 보다 효율적으로 관리할 수 있습니다.

스레드를 사용할 때에는 주의사항을 지켜야 하며, 메인 스레드와 백그라운드 스레드 간의 데이터 동기화, UI 업데이트 등을 신중하게 처리해야 합니다. 또한 Android 8.0 이상에서는 백그라운드 제한 정책으로 인해 백그라운드 작업에 제한이 있으므로 백그라운드 작업을 실행하기 위해서는 작업을 Foreground Service, JobScheduler, WorkManager 등을 활용하여 관리하는 것이 좋습니다.

Application에서의 네트워크에 대한 개요 및 Retrofit 활용하기

<네트워크>
Android에서의 네트워크 통신은 앱이 인터넷을 통해 서버와 통신하거나 외부 리소스에 접근하는 데 사용되는 기술입니다. 안드로이드 앱은 다양한 방법으로 네트워크 통신을 수행할 수 있습니다. 가장 흔한 방법 중 일부는 다음과 같습니다:

HttpURLConnection과 HttpClient: Android에서 기본적으로 제공되는 클래스로, 네트워크 요청과 응답을 처리하는 데 사용됩니다. 단순한 요청을 처리할 때 주로 사용됩니다.

Retrofit: Square에서 제공하는 라이브러리로, 안드로이드 앱에서 강력한 RESTful API 통신을 쉽게 구현할 수 있게 해주는 라이브러리입니다. 주로 JSON 형식의 데이터와 작업할 때 사용됩니다.

Volley: Google에서 제공하는 라이브러리로, HTTP 요청과 응답을 처리하며, 이미지 로딩 및 캐싱 등을 처리하는 데도 사용됩니다.

OkHttp: Square에서 제공하는 라이브러리로, HttpUrlConnection을 대체하거나 보완하여 더 효율적으로 네트워크 요청을 처리할 수 있게 해줍니다.

WebSocket: 실시간 통신을 위한 프로토콜로, 양방향 통신을 지원하며, 앱 내에서 서버와 지속적인 연결을 유지할 때 사용됩니다.

Firebase Cloud Messaging (FCM): Google의 클라우드 메시징 서비스로, 푸시 알림을 전송하거나 서버에서 클라이언트로 메시지를 전달할 때 사용됩니다.

안드로이드에서의 네트워크 통신은 백그라운드 스레드에서 수행되어야 합니다. 메인(UI) 스레드에서 네트워크 요청을 수행하면 앱의 응답성이 저하될 수 있습니다. 따라서 AsyncTask, Thread, Handler, RxJava 등의 메커니즘을 사용하여 네트워크 요청을 비동기적으로 처리해야 합니다.

또한, 보안을 위해 SSL(HTTPS) 연결을 사용하는 것이 좋습니다. 네트워크 통신에서 발생할 수 있는 에러나 예외를 적절히 처리하고, 사용자에게 알리는 방법도 중요한 고려사항입니다.

<네트워크 라이브러리>

```
// https://mvnrepository.com/artifact/com.squareup.retrofit2/retrofit
implementation ("com.squareup.retrofit2:retrofit:2.9.0")
//
https://mvnrepository.com/artifact/com.squareup.retrofit2/converter-gso
n
implementation("com.squareup.retrofit2:converter-gson:2.9.0")
// https://mvnrepository.com/artifact/com.github.bumptech.glide/glide
```

```
implementation("com.github.bumptech.glide:glide:4.11.0")
```

Back-end Server와 Android 연동
Back-end Server로 Spring Boot를 활용하여 Android Studio와 연동을 합니다.

<스프링 부트 활용 안드로이드 스튜디오 연계>
lombok Devtool web JPA SQL
AndroidPhone - StarterProject 생성

Package
Controller Model Repository Service 생성

<properties 수정>
server.port=8899

#database
spring.datasource.dbcp2.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.url=jdbc:mysql://localhost:3306/androiddb?useSSL=false&serverTimezone=Asia/Seoul&characterEncoding=UTF-8
spring.datasource.username=root
spring.datasource.password=root

spring.devtools.livereload.enabled=true

spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true

기타 부족한 부분은 소스 코드를 참고 및 구글링하기 바랍니다.