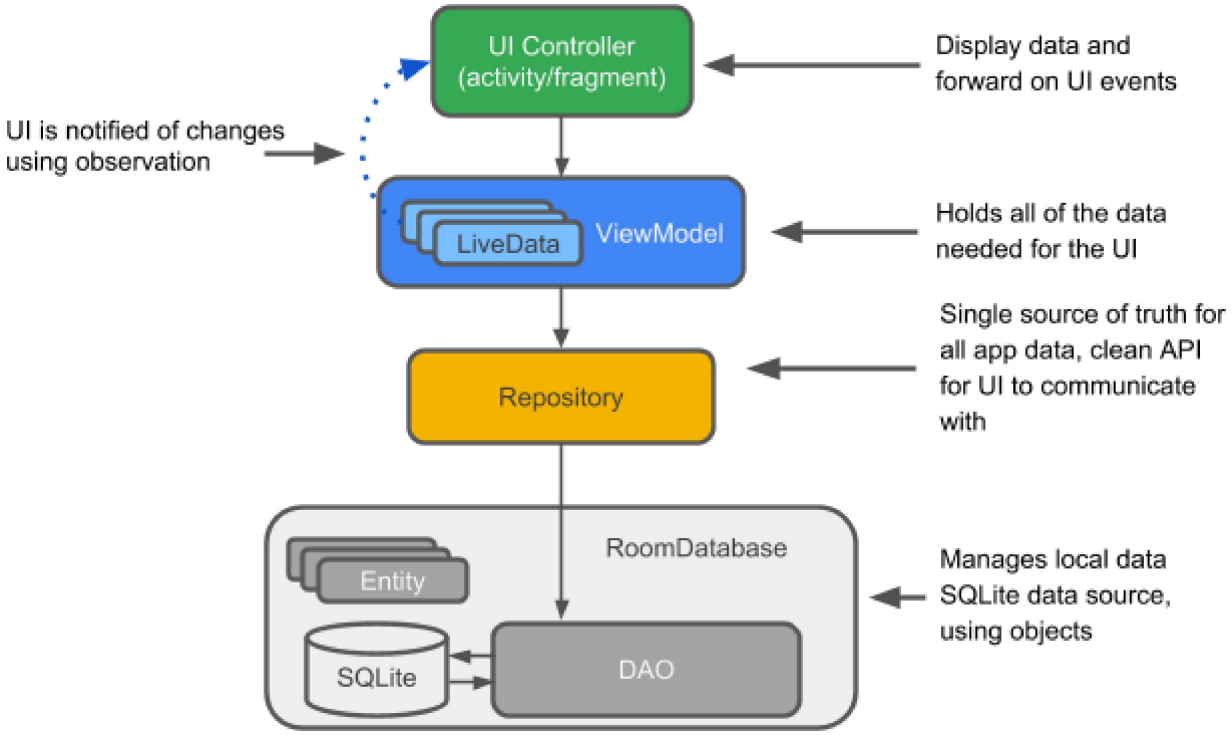
Room with Navigation in Kotlin

https://www.youtube.com/watch?v=5qlIPTDE274
https://codelabs.developers.google.com/codelabs/android-room-with-a-view/#0

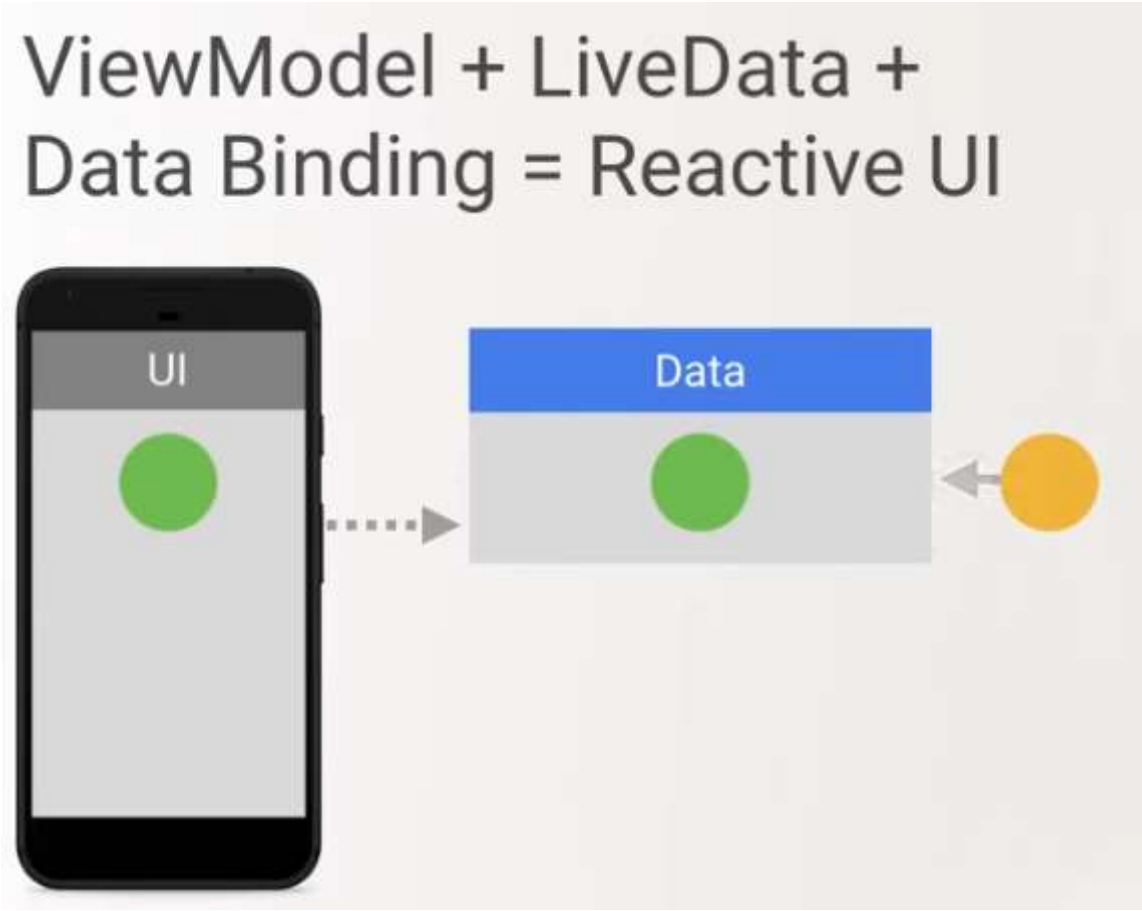
Architektur

MVVM



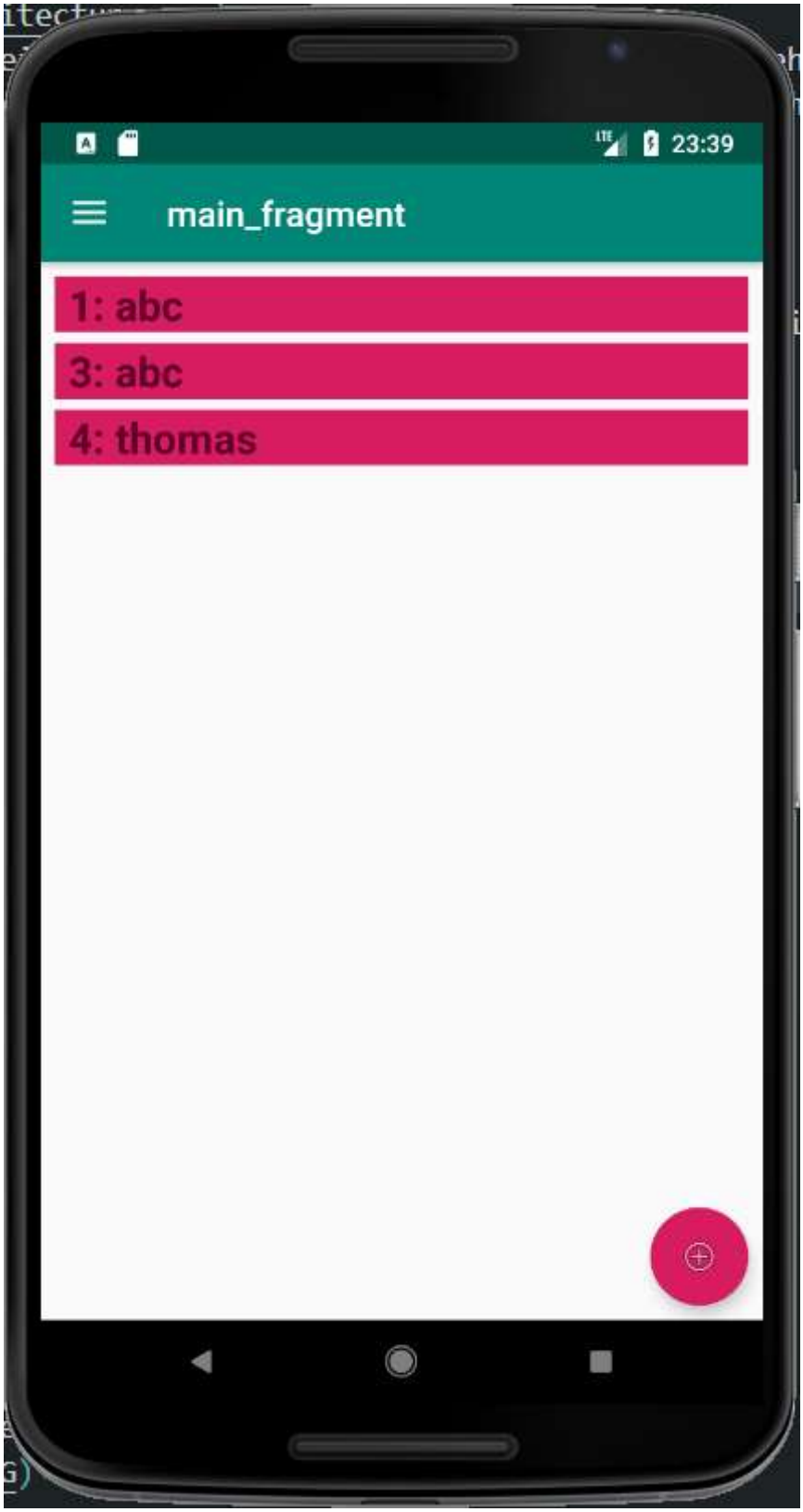
Der Vorteil des ViewModels ist, dass es die Daten behält, auch wenn die Activity neu generiert wird. (z.B.: Rotieren des Telefons)

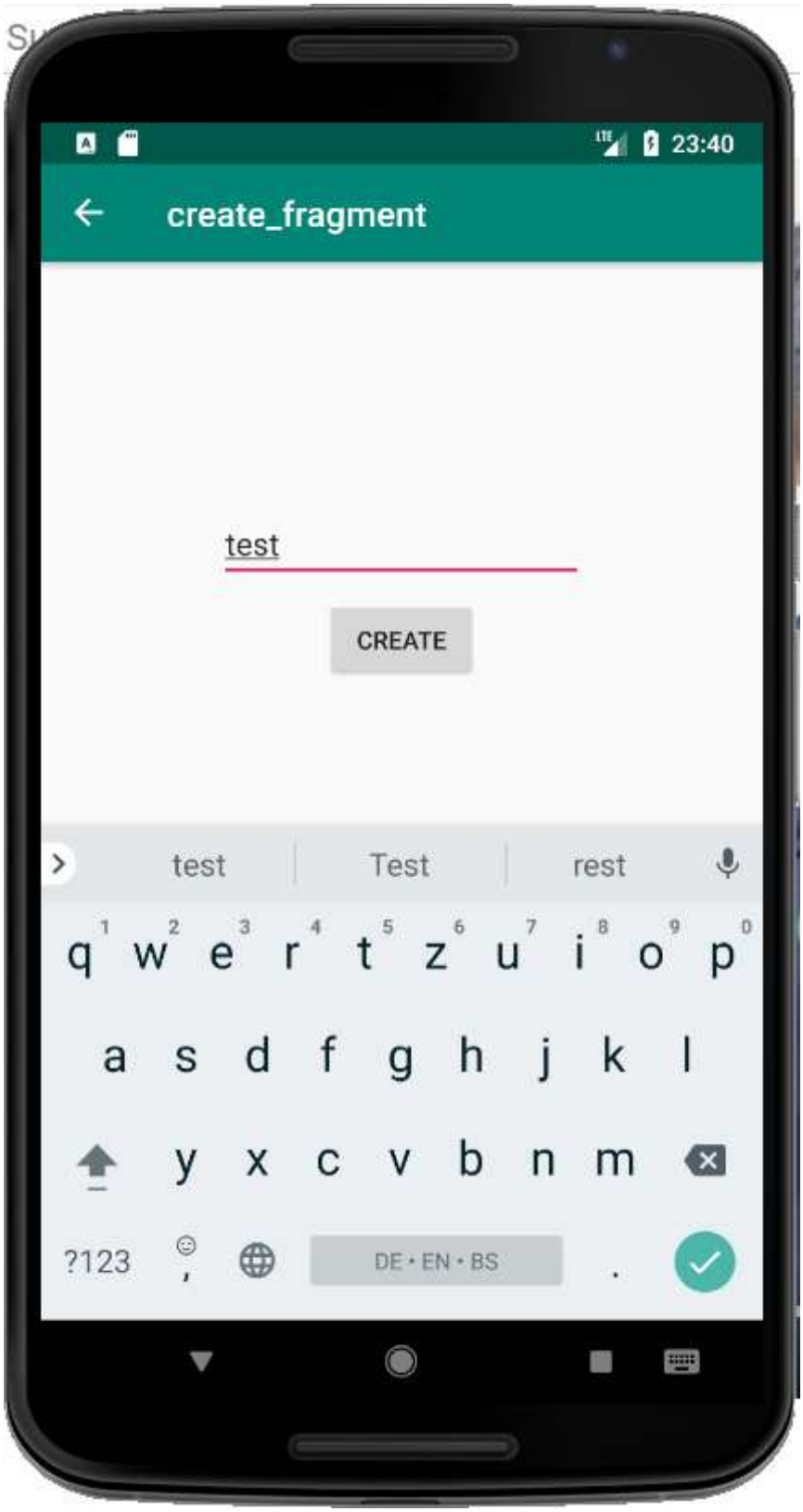
Warum LiveData

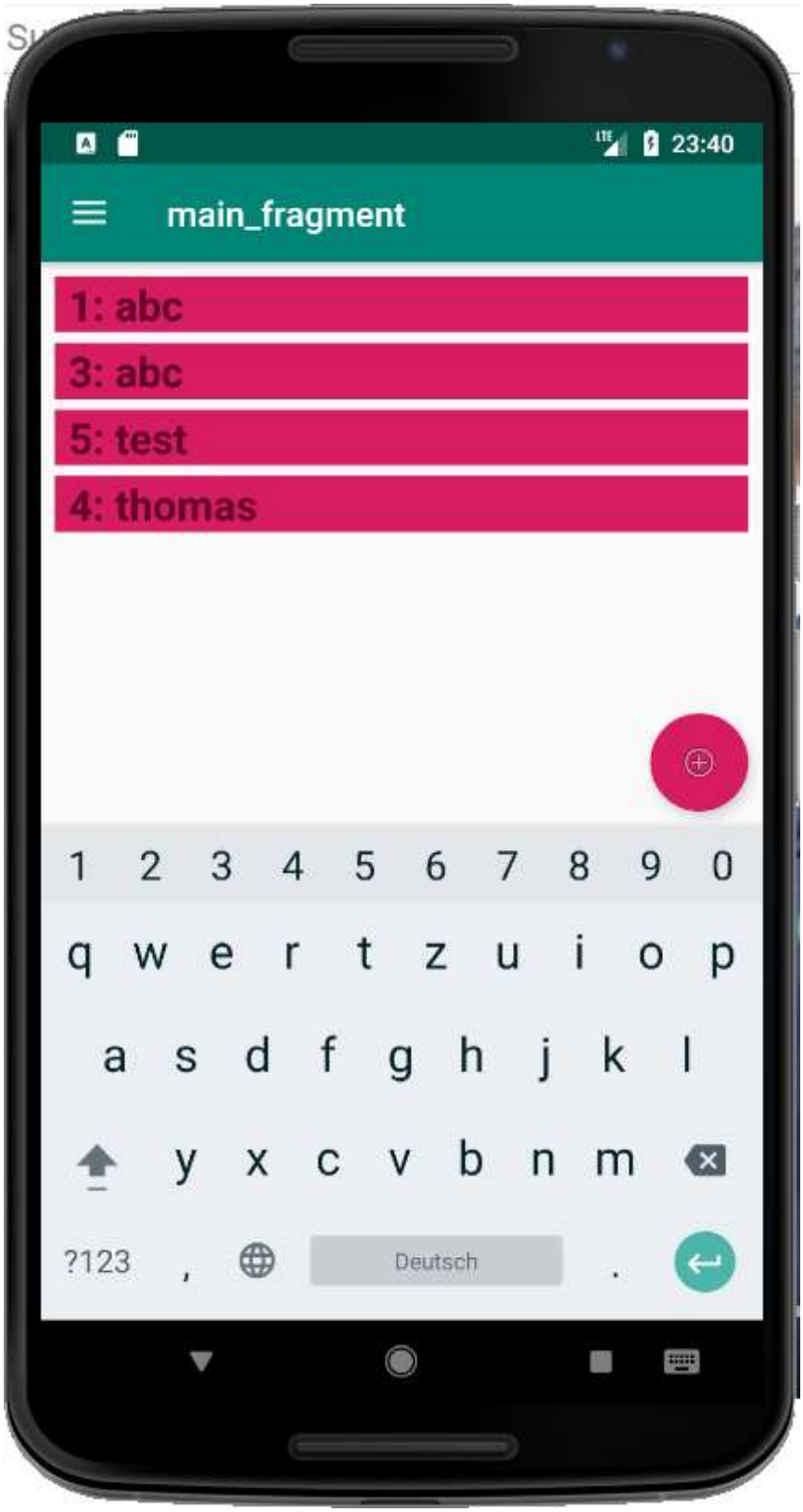


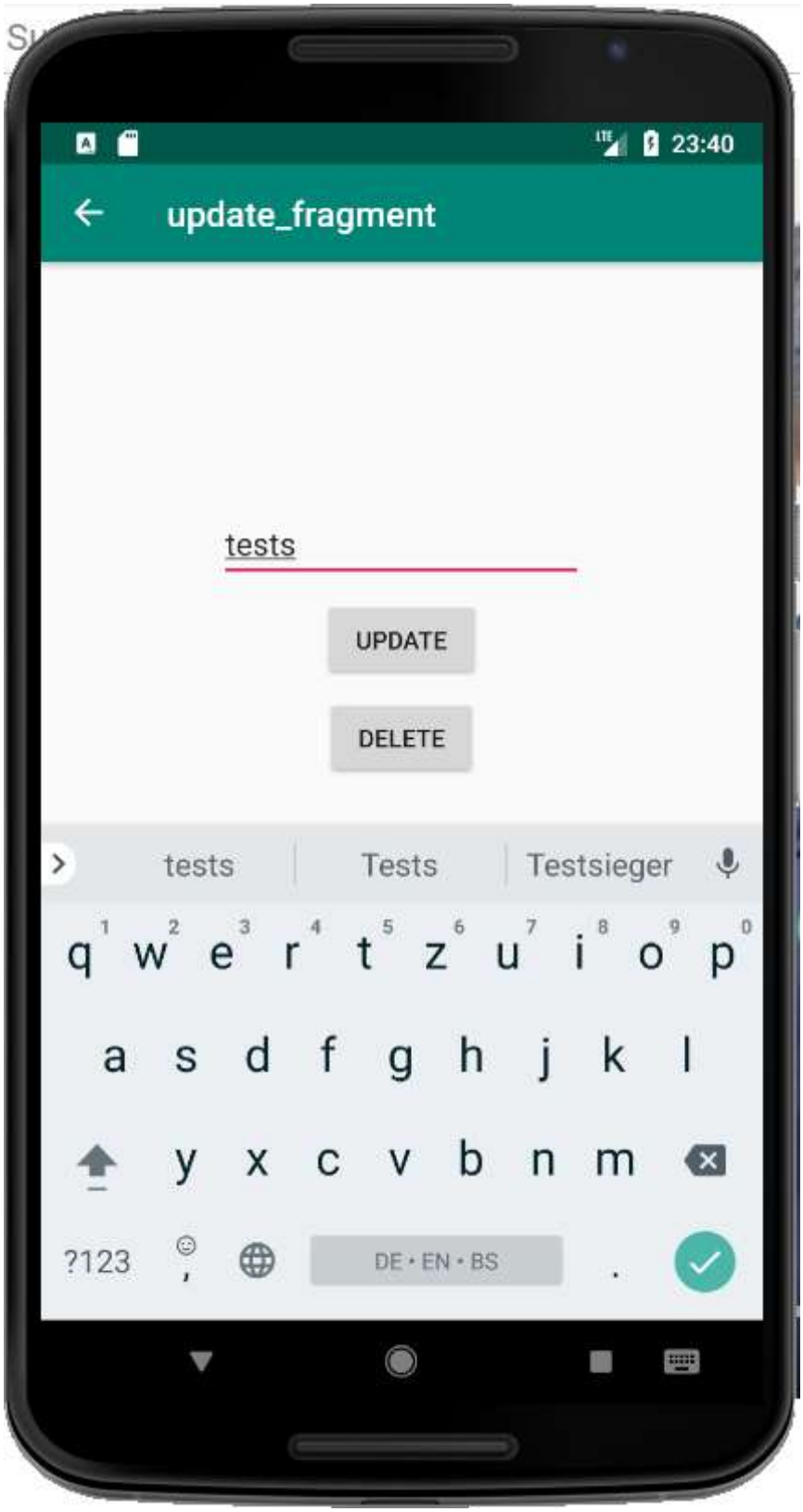
LiveData bekommt immer eine Benachrichtigung, wenn sich was in der Datenbank ändert
Das Beispiel wäre noch mit Data Binding erweiterbar!

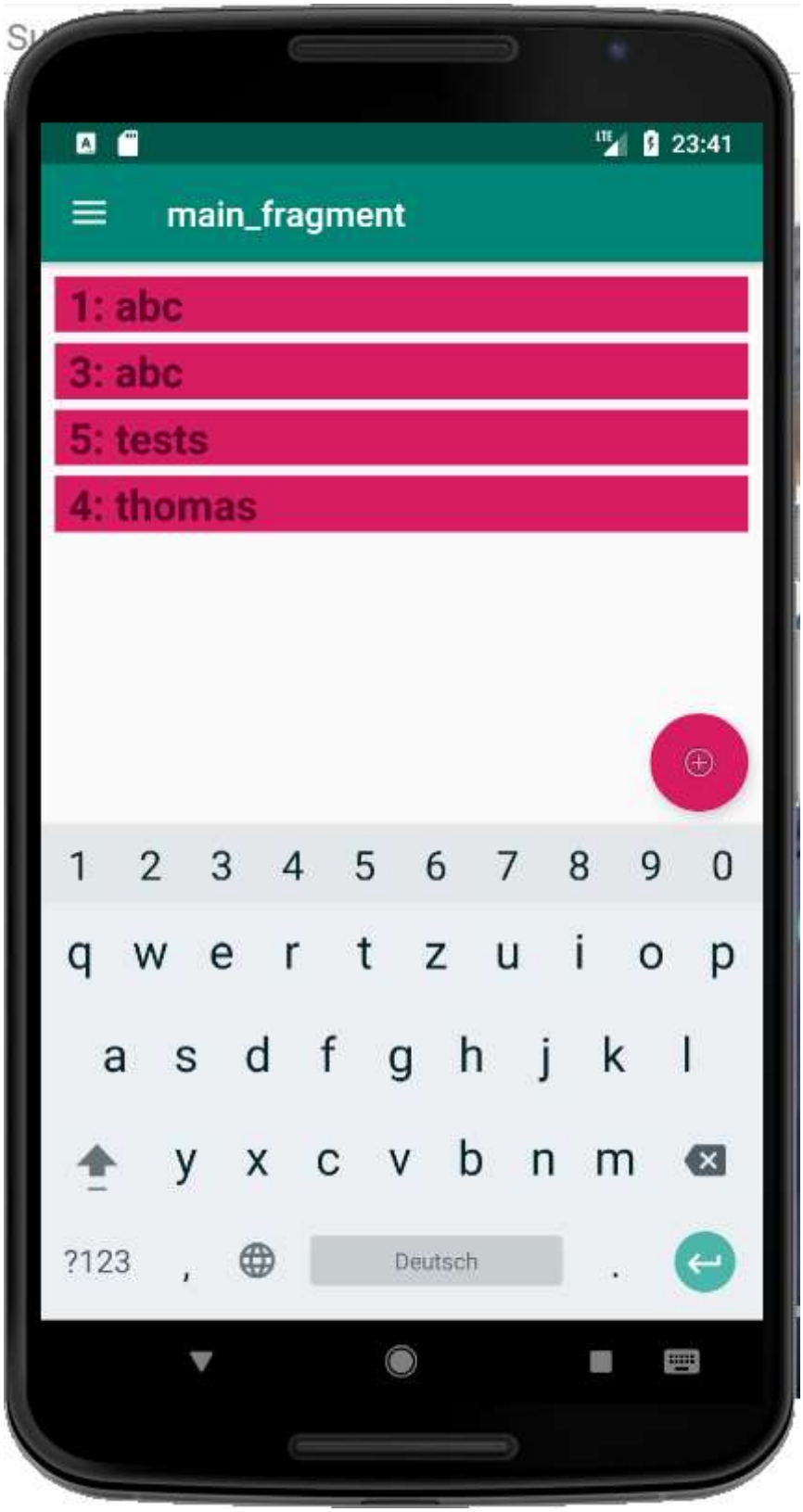
Was wir erreichen wollen

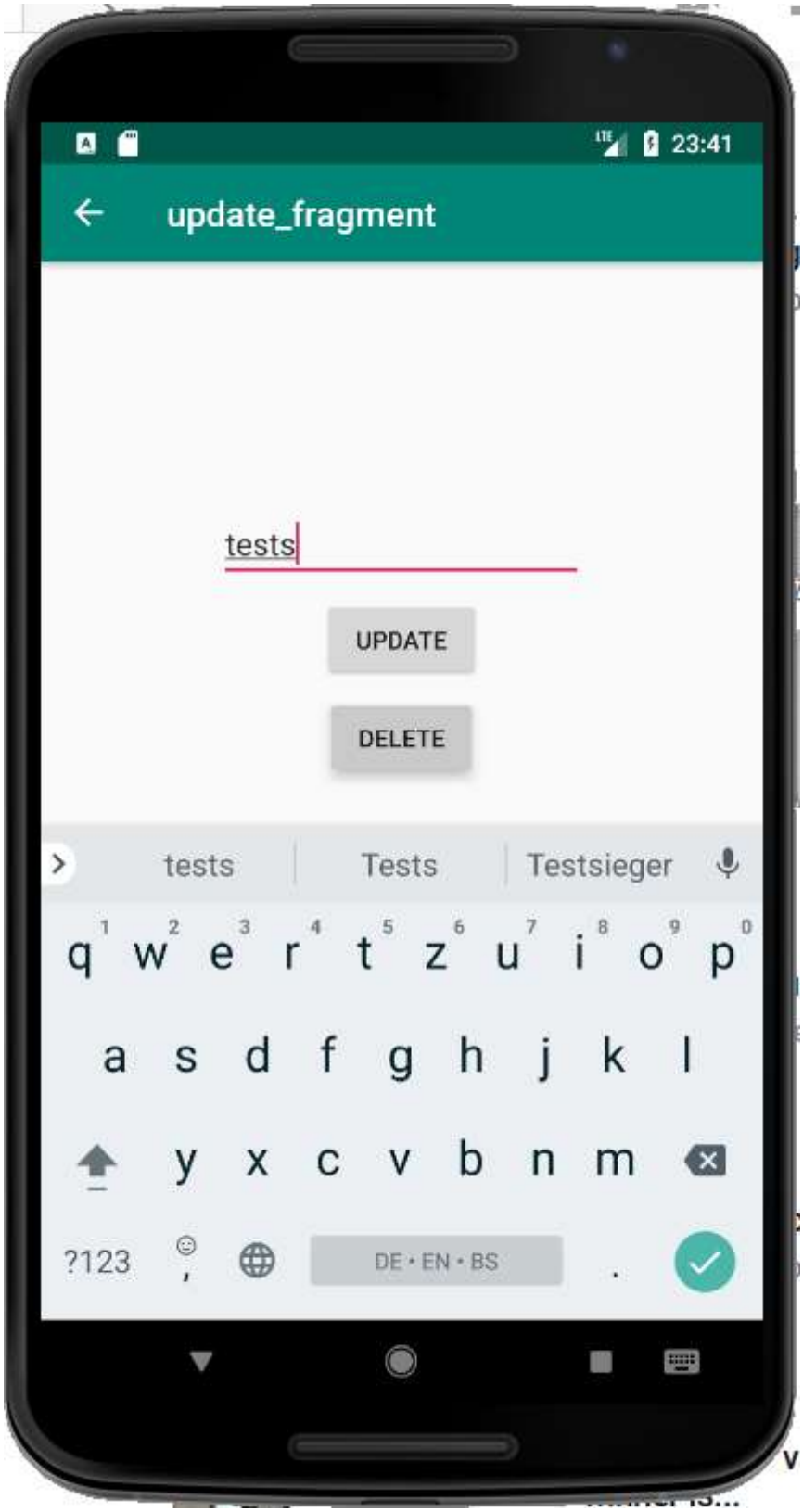

23:39
main_fragment
1: abc
3: abc
4: thomas

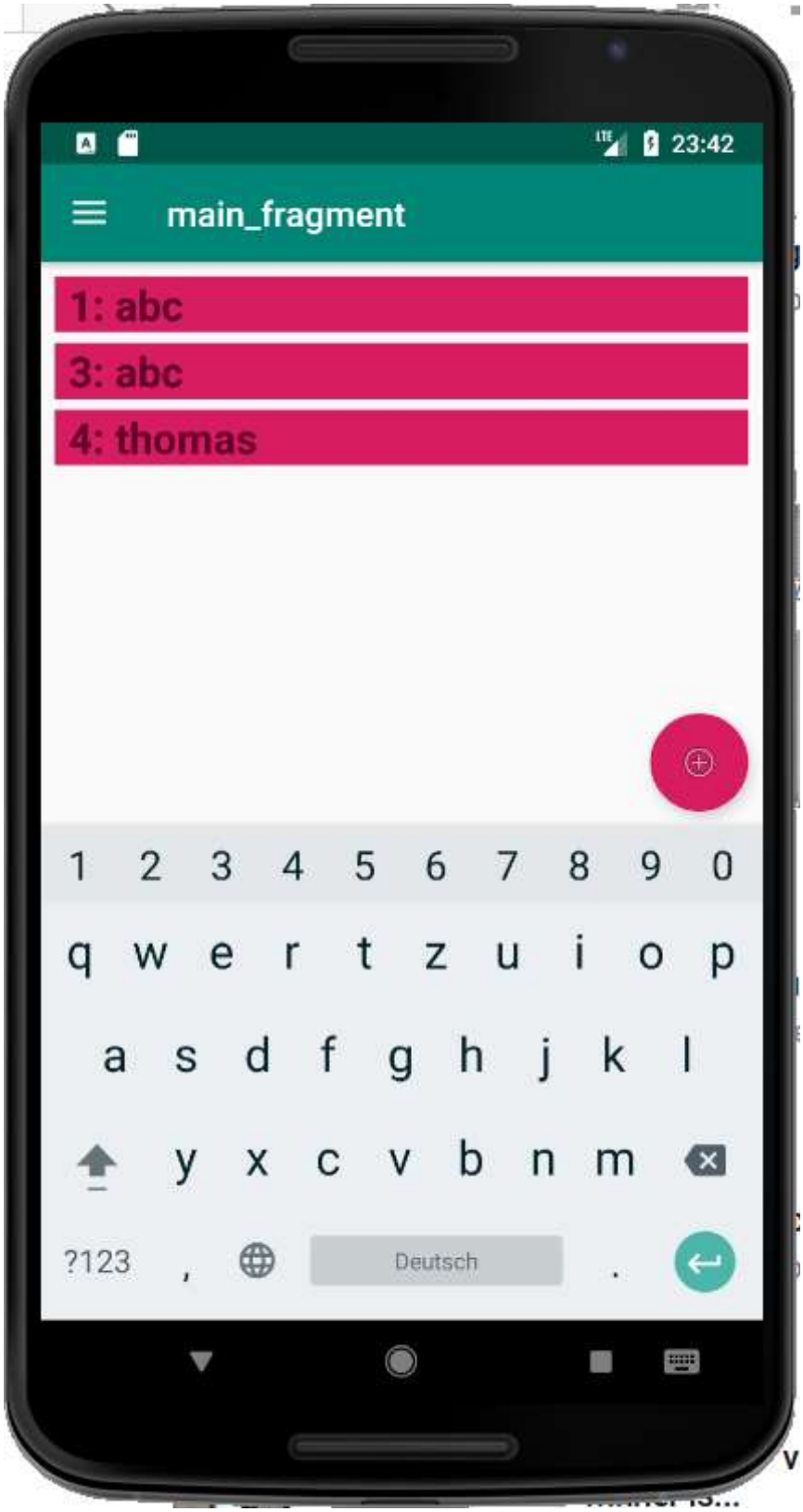

23:40
create_fragment
test
CREATE
test
Test
rest
q w e r t z u i o p
a s d f g h j k l
y x c v b n m
?123
DE • EN • BS

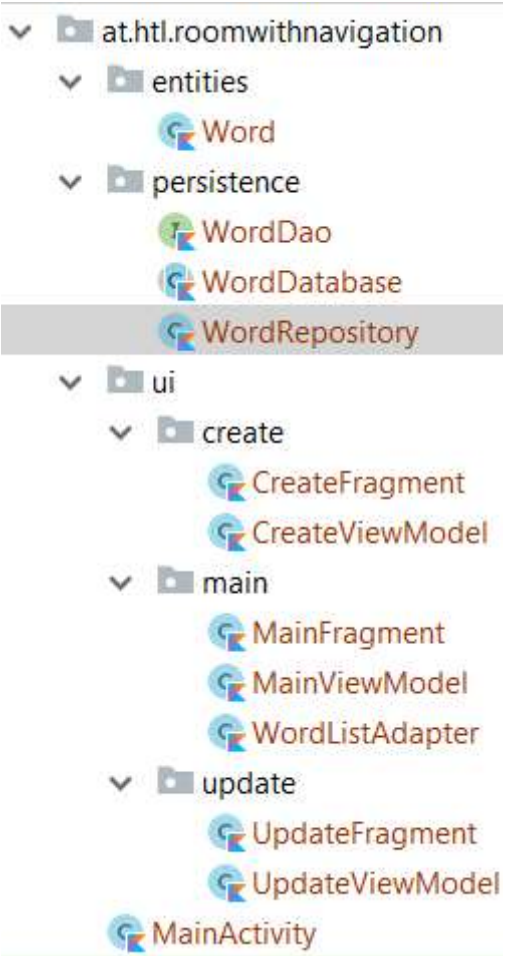

23:40
main_fragment
1: abc
3: abc
5: test
4: thomas
1 2 3 4 5 6 7 8 9 0
q w e r t z u i o p
a s d f g h j k l
y x c v b n m
?123
Deutsch


23:40
update_fragment
tests
UPDATE
DELETE
tests
Tests
Testsieger
q w e r t z u i o p
a s d f g h j k l
y x c v b n m
?123
DE • EN • BS


23:41
main_fragment
1: abc
3: abc
5: tests
4: thomas
1 2 3 4 5 6 7 8 9 0
q w e r t z u i o p
a s d f g h j k l
y x c v b n m
?123
Deutsch


23:41
update_fragment
tests
UPDATE
DELETE
tests
Tests
Testsieger
q w e r t z u i o p
a s d f g h j k l
y x c v b n m
?123
DE • EN • BS


23:42
main_fragment
1: abc
3: abc
4: thomas
1 2 3 4 5 6 7 8 9 0
q w e r t z u i o p
a s d f g h j k l
y x c v b n m
?123
Deutsch


at.htl.roomwithnavigation
entities
Word
persistence
WordDao
WordDatabase
WordRepository
ui
create
CreateFragment
CreateViewModel
main
MainFragment
MainViewModel
WordListAdapter
update
UpdateFragment
UpdateViewModel
MainActivity

Los geht´s

Create New Project

Create Android Project

Application name

RoomWithNavigation

Company domain

htl.at

Project location

D:\Schule\RoomWithNavigation\RoomWithNavigation ...

Package name

at.htl.roomwithnavigation Edit

☐ **Include C++ support**

☑ **Include Kotlin support**

⚠ 'RoomWithNavigation' already exists at the specified project location.

Previous Next Cancel Finish

Create New Project

Target Android Devices

Select the form factors and minimum SDK

Some devices require additional SDKs. Low API levels target more devices, but offer fewer API features.

☑ **Phone and Tablet**

API 27: Android 8.1 (Oreo)

By targeting **API 27 and later**, your app will run on approximately **1,1%** of devices. Help me choose

☐ Include Android Instant App support

☐ **Wear OS**

API 23: Android 6.0 (Marshmallow)

☐ **TV**

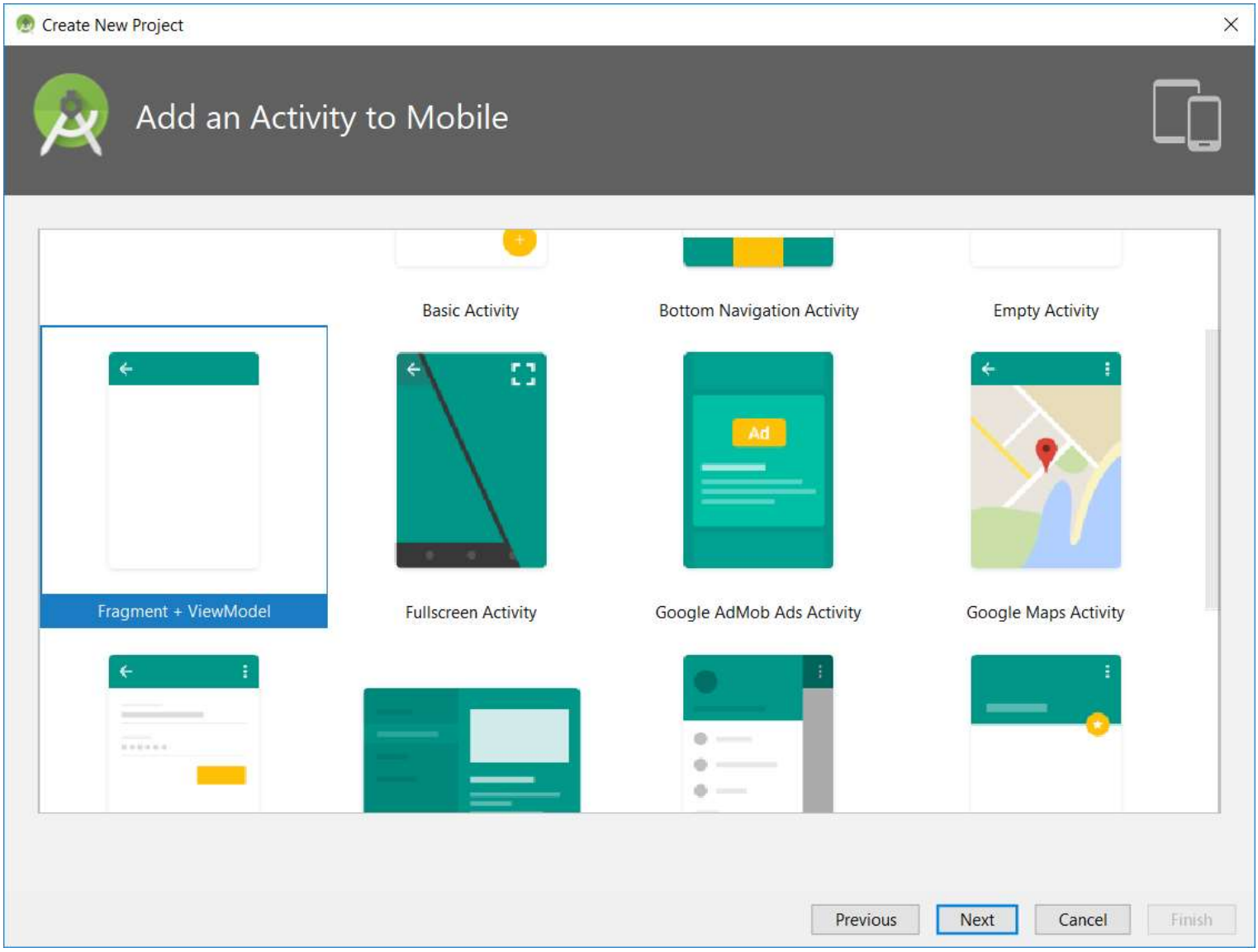
API 21: Android 5.0 (Lollipop)

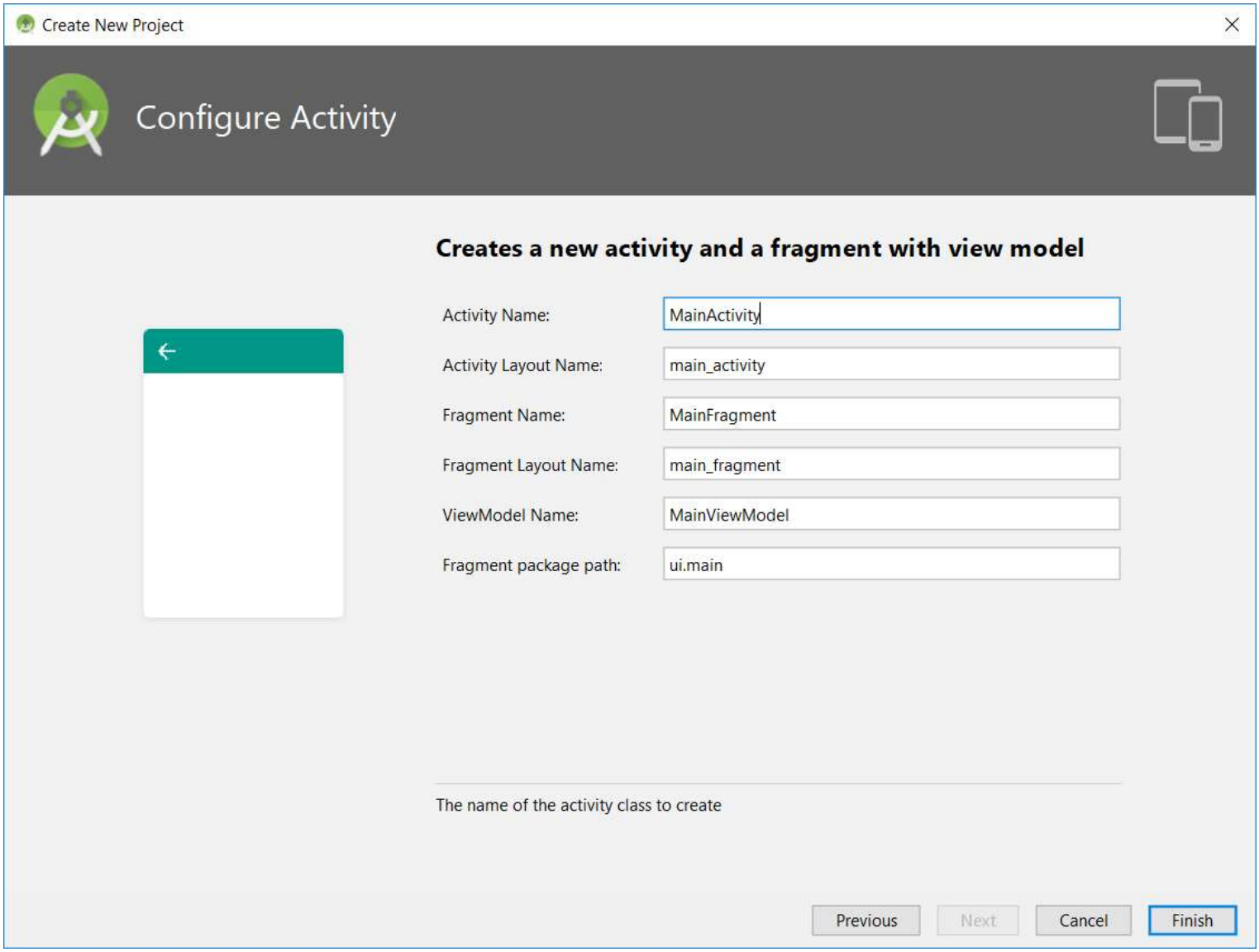
☐ **Android Auto**

☐ **Android Things**

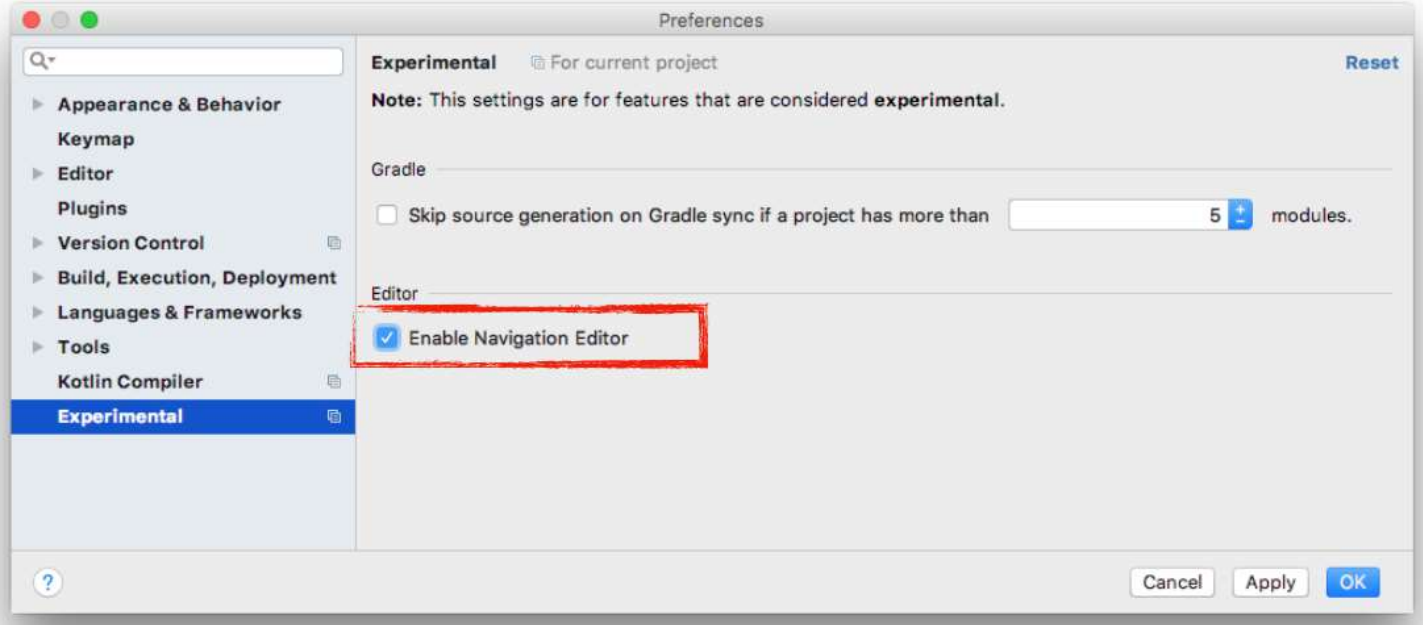
API 24: Android 7.0 (Nougat)

Previous | Next | Cancel | Finish


Create New Project
Add an Activity to Mobile
Basic Activity
Bottom Navigation Activity
Empty Activity
Fragment + ViewModel
Fullscreen Activity
Google AdMob Ads Activity
Google Maps Activity
Ad
Previous
Next
Cancel
Finish



Kontrolliere ...



in gradle project

```
ext {
    roomVersion = '1.1.1'
    archLifecycleVersion = '1.1.1'
}

buildscript {
    ext.kotlin_version = '1.2.70'
    repositories {
        google()
        jcenter()
    }
    dependencies {
        classpath 'com.android.tools.build:gradle:3.2.0'
        classpath "org.jetbrains.kotlin:kotlin-gradle-plugin:$kotlin_version"

        // NOTE: Do not place your application dependencies here; they belong
        // in the individual module build.gradle files
    }
}

allprojects {
    repositories {
        google()
        jcenter()
    }
}

task clean(type: Delete) {
    delete rootProject.buildDir
}
```

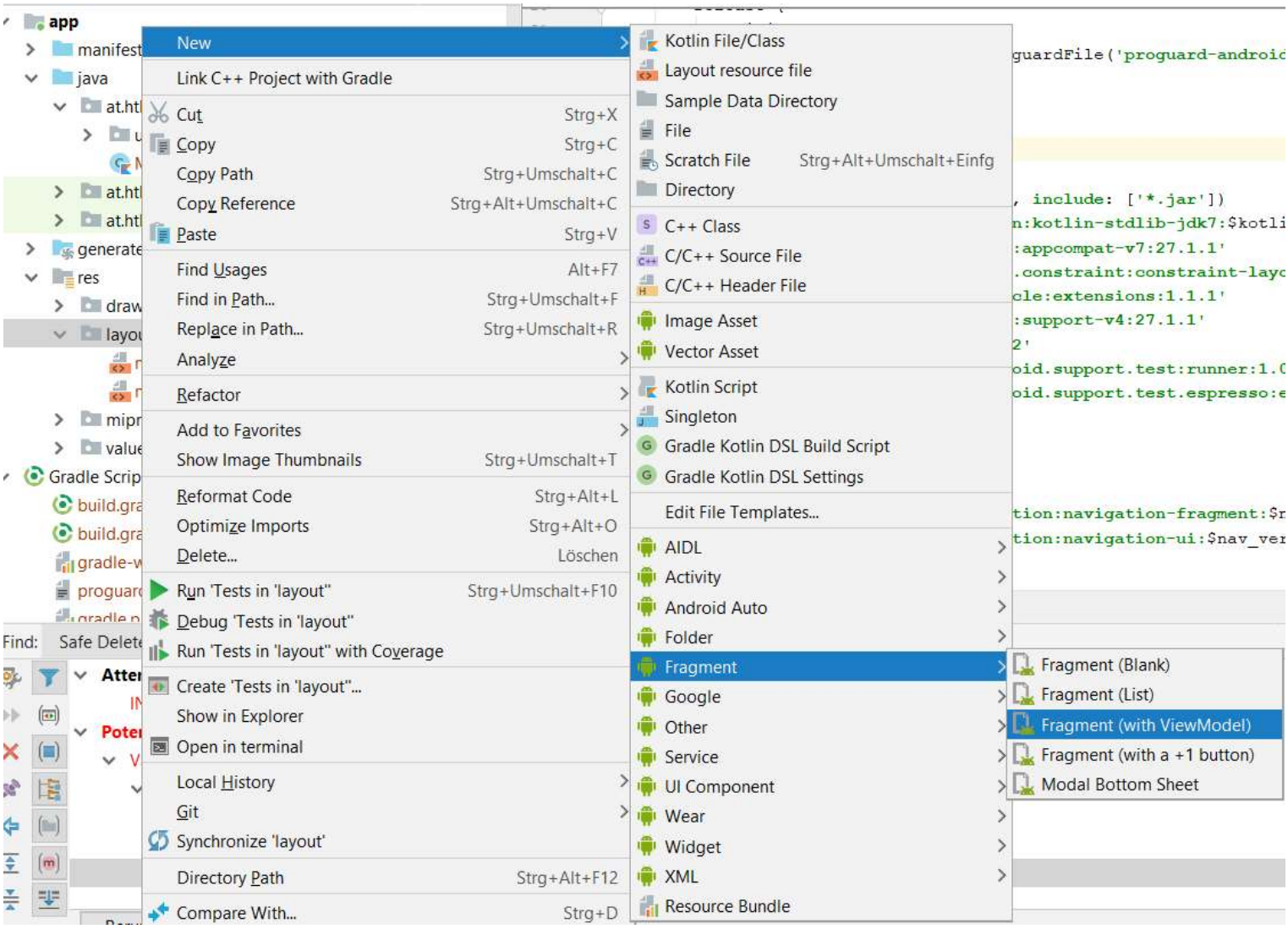
```
ext {
    roomVersion = '1.1.1'
    archLifecycleVersion = '1.1.1'
}
```

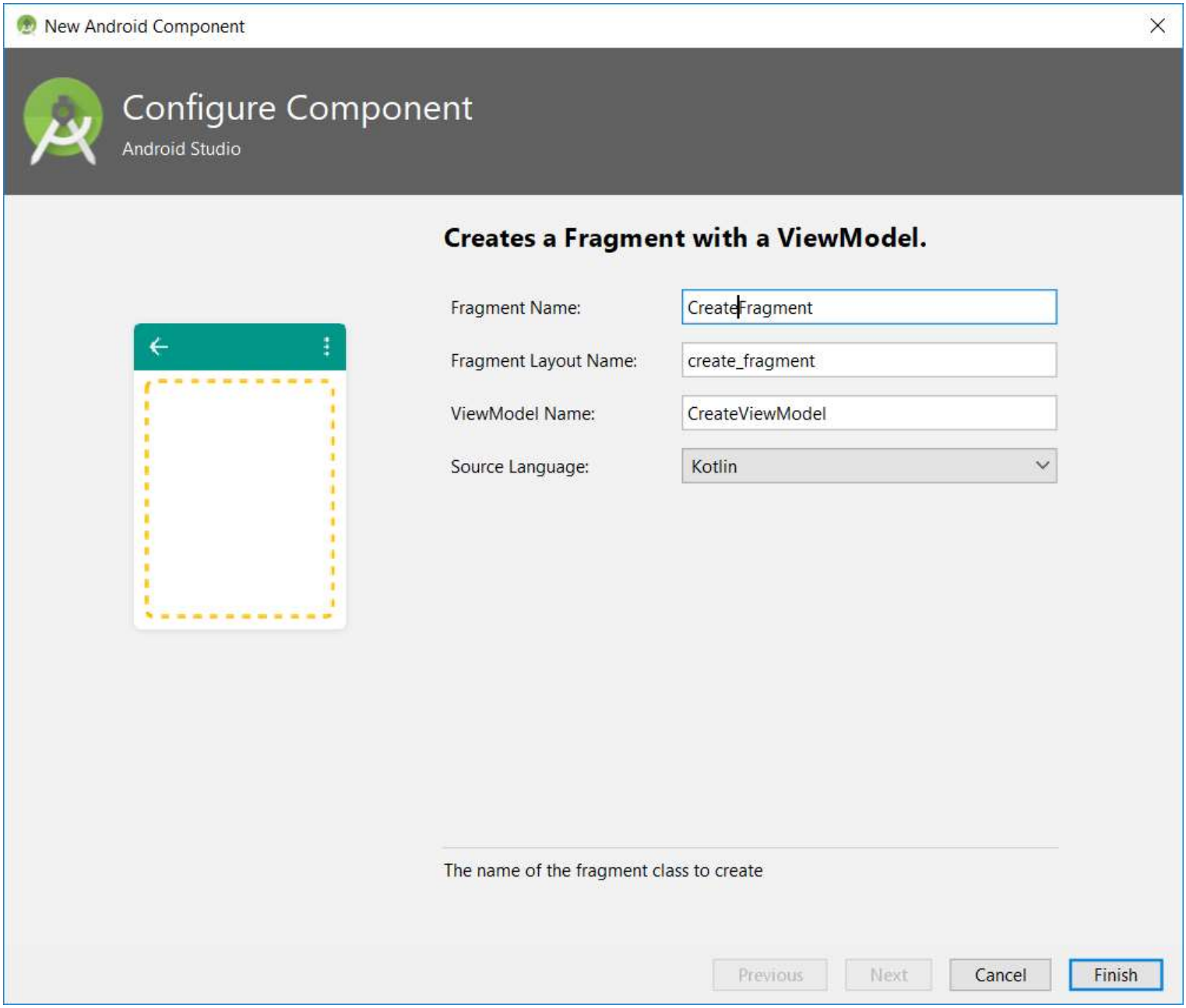
in gradle app

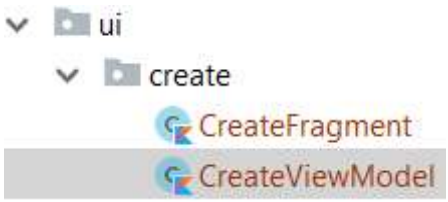
```
apply plugin: 'kotlin-kapt'
```

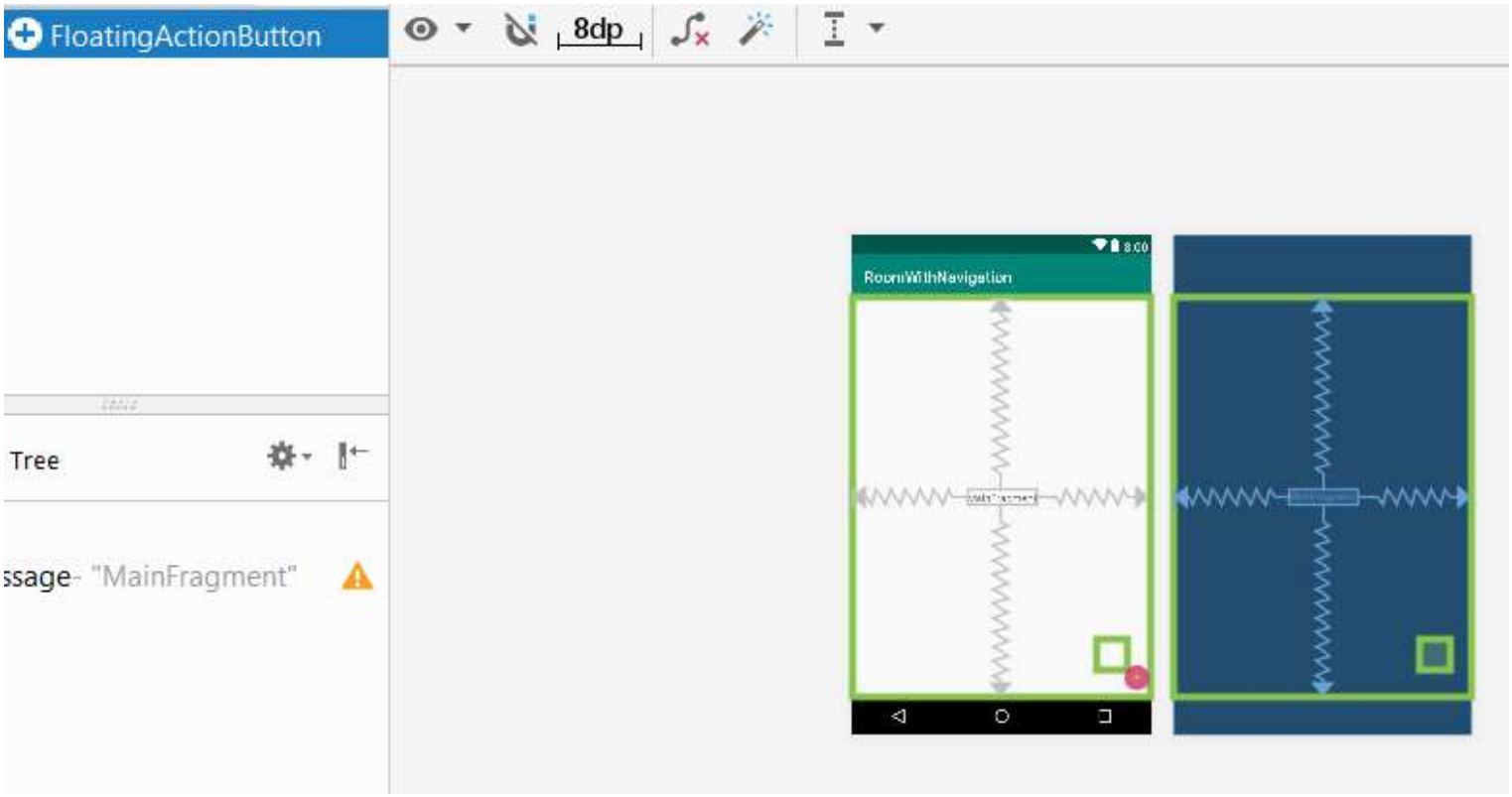
```
//Navigation
def nav_version = "1.0.0-alpha06"

implementation "android.arch.navigation:navigation-fragment:$nav_version" // use -ktx for Kotlin
implementation "android.arch.navigation:navigation-ui:$nav_version" // use -ktx for Kotlin

// Room components
implementation "android.arch.persistence.room:runtime:$rootProject.roomVersion"
kapt  "android.arch.persistence.room:compiler:$rootProject.roomVersion"
androidTestImplementation "android.arch.persistence.room:testing:$rootProject.roomVersion"

// Lifecycle components
implementation "android.arch.lifecycle:extensions:$rootProject.archLifecycleVersion"
kapt  "android.arch.lifecycle:compiler:$rootProject.archLifecycleVersion"
```

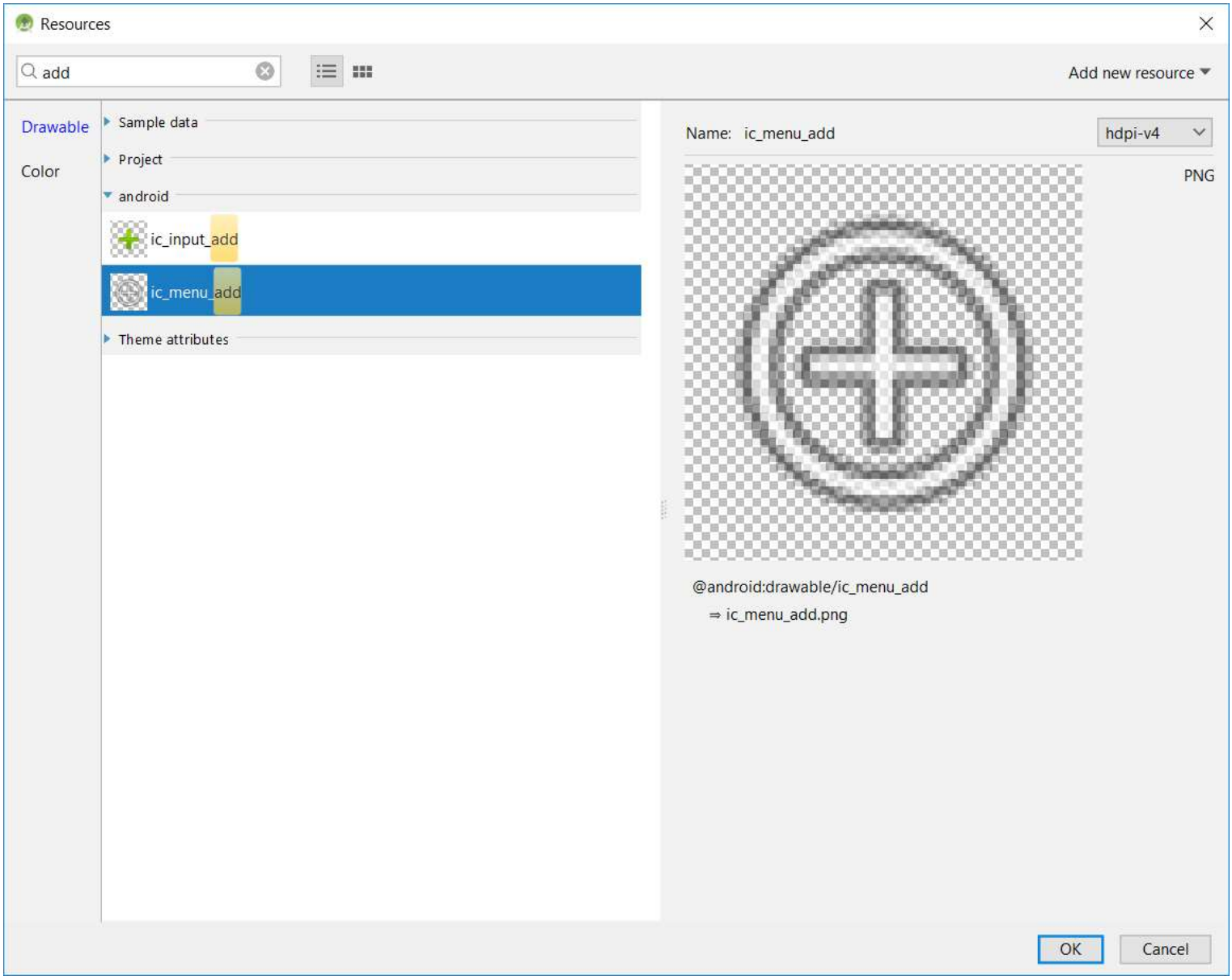
```
dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation"org.jetbrains.kotlin:kotlin-stdlib-jdk7:$kotlin_version"
    implementation 'com.android.support:appcompat-v7:27.1.1'
    implementation 'com.android.support.constraint:constraint-layout:1.1.3'
    implementation 'android.arch.lifecycle:extensions:1.1.1'
    testImplementation 'junit:junit:4.12'
    androidTestImplementation 'com.android.support.test:runner:1.0.2'
    androidTestImplementation 'com.android.support.test.espresso:espresso-core:3.0.2'

    //Navigation
    def nav_version = "1.0.0-alpha06"

    implementation "android.arch.navigation:navigation-fragment:$nav_version" // use -ktx for Kotlin
    implementation "android.arch.navigation:navigation-ui:$nav_version" // use -ktx for Kotlin

    // Room components
    implementation "android.arch.persistence.room:runtime:$rootProject.roomVersion"
    kapt  "android.arch.persistence.room:compiler:$rootProject.roomVersion"
    androidTestImplementation "android.arch.persistence.room:testing:$rootProject.roomVersion"

    // Lifecycle components
    implementation "android.arch.lifecycle:extensions:$rootProject.archLifecycleVersion"
    kapt  "android.arch.lifecycle:compiler:$rootProject.archLifecycleVersion"
}
```

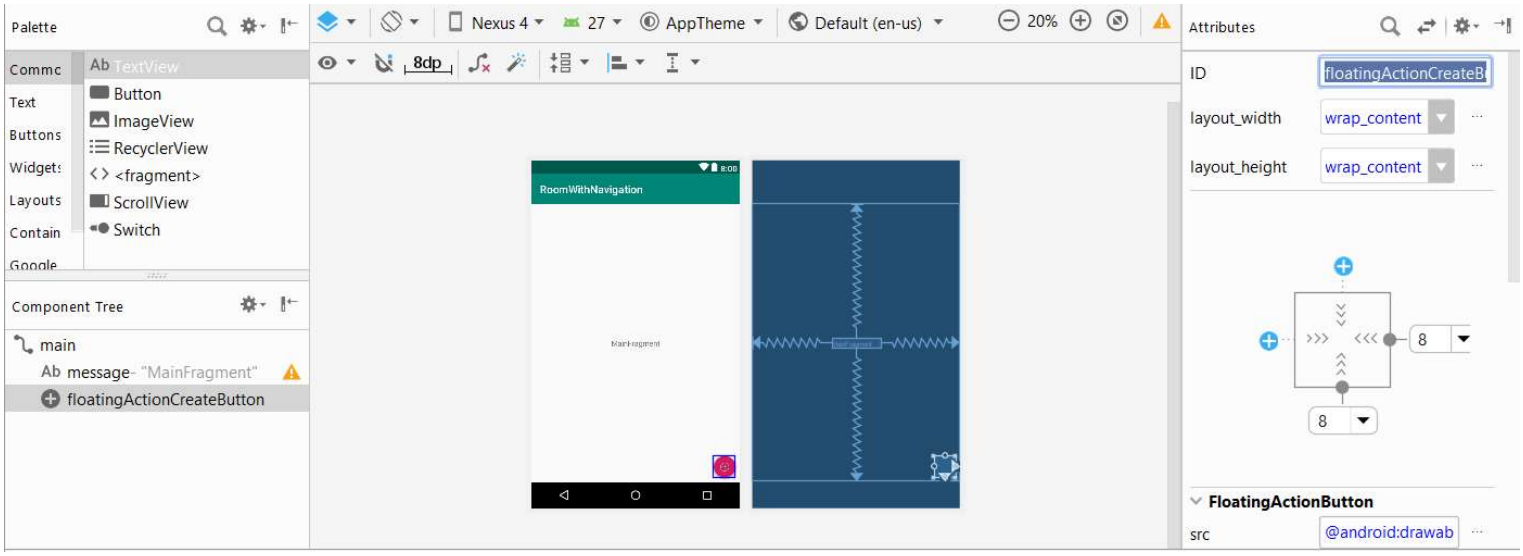

app
manifest
java
generate
res
Gradle Scrip
Find:
New
Link C++ Project with Gradle
Cut Strg+X
Copy Strg+C
Copy Path Strg+Umschalt+C
Copy Reference Strg+Alt+Umschalt+C
Paste Strg+V
Find Usages Alt+F7
Find in Path... Strg+Umschalt+F
Replace in Path... Strg+Umschalt+R
Analyze
Refactor
Add to Favorites
Show Image Thumbnails Strg+Umschalt+T
Reformat Code Strg+Alt+L
Optimize Imports Strg+Alt+O
Delete... Löschen
Run 'Tests in 'layout'' Strg+Umschalt+F10
Debug 'Tests in 'layout''
Run 'Tests in 'layout'' with Coverage
Create 'Tests in 'layout''...
Show in Explorer
Open in terminal
Local History
Git
Synchronize 'layout'
Directory Path Strg+Alt+F12
Compare With... Strg+D
Kotlin File/Class
Layout resource file
Sample Data Directory
File
Scratch File Strg+Alt+Umschalt+Einfg
Directory
C++ Class
C/C++ Source File
C/C++ Header File
Image Asset
Vector Asset
Kotlin Script
Singleton
Gradle Kotlin DSL Build Script
Gradle Kotlin DSL Settings
Edit File Templates...
AIDL
Activity
Android Auto
Folder
Fragment
Google
Other
Service
UI Component
Wear
Widget
XML
Resource Bundle
Fragment (Blank)
Fragment (List)
Fragment (with ViewModel)
Fragment (with a +1 button)
Modal Bottom Sheet


New Android Component
Configure Component
Android Studio
Creates a Fragment with a ViewModel.
Fragment Name:
CreateFragment
Fragment Layout Name:
create_fragment
ViewModel Name:
CreateViewModel
Source Language:
Kotlin
The name of the fragment class to create
Previous
Next
Cancel
Finish


ui
create
CreateFragment
CreateViewModel

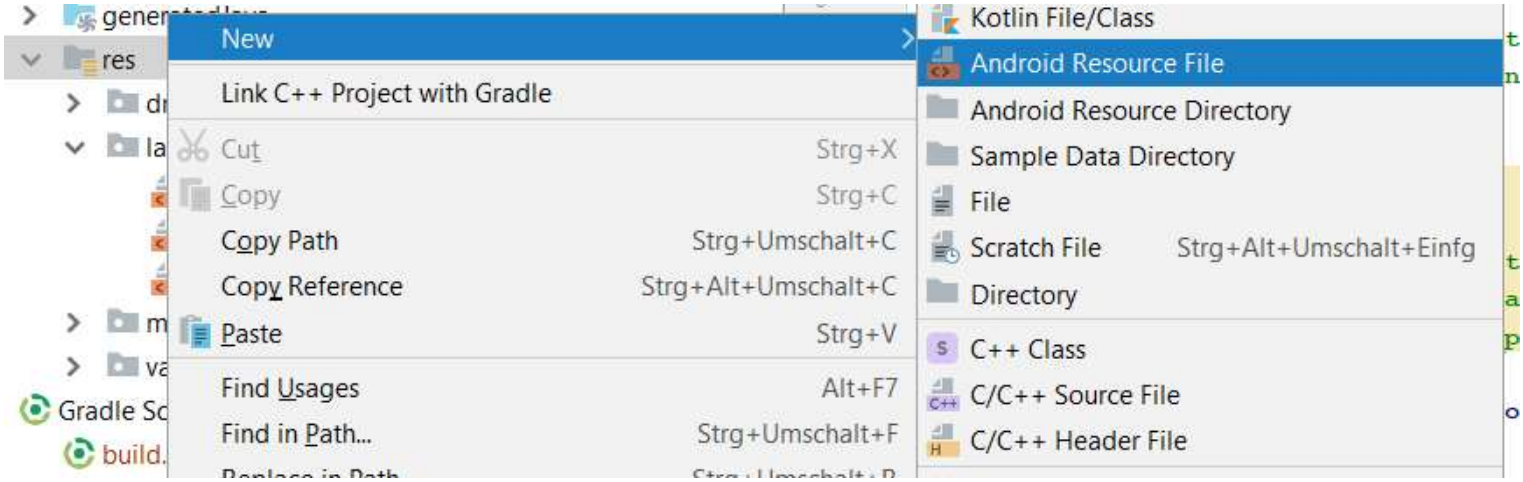

FloatingActionButton
8dp
Tree
ssage- "MainFragment"
RoomWithNavigation

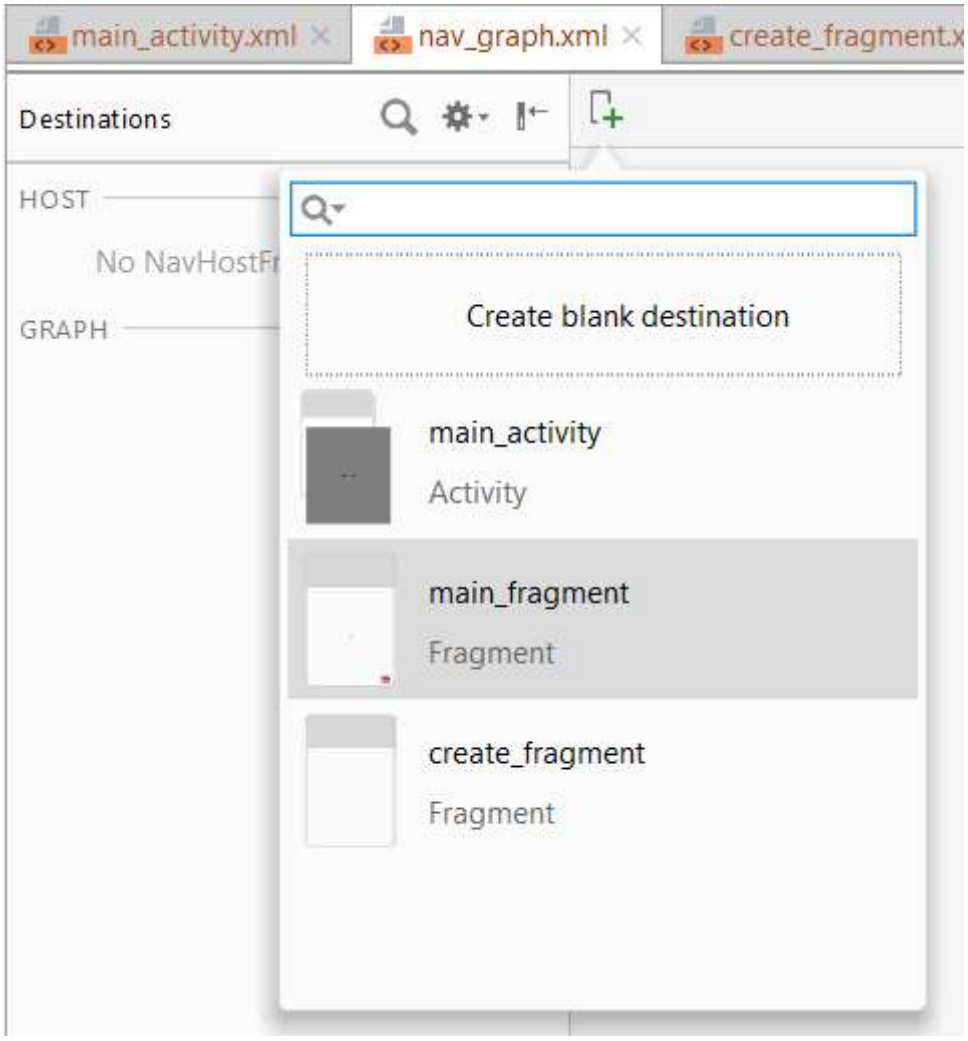

Resources
add
Add new resource
Drawable
Color
Sample data
Project
android
ic_input_add
ic_menu_add
Theme attributes
Name: ic_menu_add
hdpi-v4
PNG
@android:drawable/ic_menu_add
⇒ ic_menu_add.png
OK
Cancel

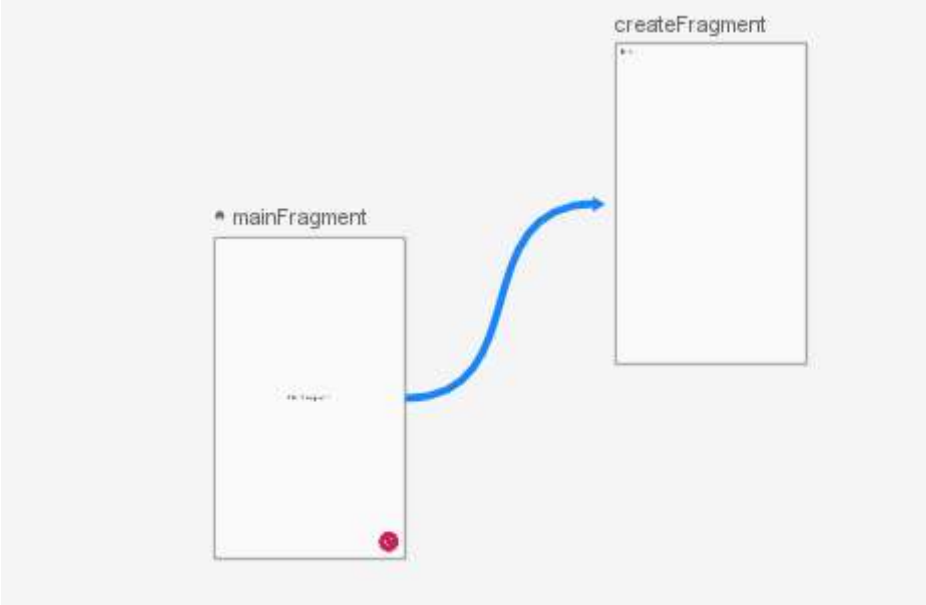


main_activity.xml | create_fragment.xml | MainViewModel.kt | CreateFragment.kt

```xml
<?xml version="1.0" encoding="utf-8"?>

<android.support.v4.widget.DrawerLayout 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:id="@+id/container"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <fragment
        android:id="@+id/my_fragment"
        android:name="androidx.navigation.fragment.NavHostFragment"
        android:layout_width="match_parent"
        android:layout_height="match_parent"></fragment>

</android.support.v4.widget.DrawerLayout>
```







in der main_activity/my_fragment

```
app:defaultNavHost="true"
app:navGraph="@navigation/nav_graph"
```

fragment.xml × | MainActivity.kt × | MainFragment.kt × | main_activity.xml ×

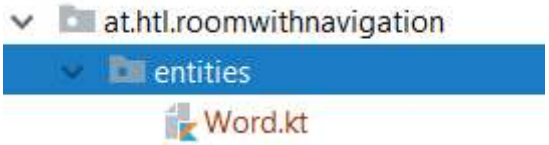
```xml
<?xml version="1.0" encoding="utf-8"?>

<android.support.v4.widget.DrawerLayout xmlns:android="http://schema
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/container"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <fragment
        android:id="@+id/my_fragment"
        android:name="androidx.navigation.fragment.NavHostFragment"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        app:defaultNavHost="true"
        app:navGraph="@navigation/nav_graph"></fragment>
```

fragment.xml × | MainActivity.kt × | MainFragment.kt × | main_activity.xml × | nav_graph.xml × | create_fragment.xml ×

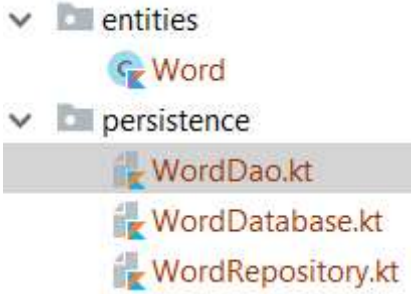
```kotlin
import ...

class MainActivity : AppCompatActivity() {

    lateinit var drawer: DrawerLayout

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.main_activity)

        val host = supportFragmentManager.findFragmentById(R.id.my_fragment) as NavHostFragment? ?: return
        val navController = host.navController

        drawer = findViewById(R.id.container)
        NavigationUI.setupActionBarWithNavController( activity: this, navController, drawer)
    }


    override fun onSupportNavigateUp(): Boolean {
        return NavigationUI.navigateUp(drawer, Navigation.findNavController( activity: this, R.id.my_fragment))
    }

}
```

jetzt können wir vom main- zu create-Fragment wechseln

```kotlin
override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
    super.onViewCreated(view, savedInstanceState)

    view.findViewById<FloatingActionButton>(R.id.floatingActionCreateButton).setOnClickListener { it: View!
        Navigation.findNavController(it).navigate(R.id.createFragment)
    }
}
```




```
import android.arch.persistence.room.Entity
import android.arch.persistence.room.PrimaryKey

@Entity(tableName = "word_table")
data class Word(
        @PrimaryKey(autoGenerate = true) val id: Long,
        var word: String)
```




```
@Dao
interface WordDao {
    @Insert
    fun insert(word: Word)

    @Update
    fun update(word: Word)

    @Query( value: "SELECT * from word_table ORDER BY id ASC")
    fun getAllLive(): LiveData<List<Word>>

    @Query( value: "DELETE FROM word_table")
    fun deleteAll()

    @Delete
    fun delete(word: Word)
}
```

```
import android.arch.persistence.room.Database
import android.arch.persistence.room.RoomDatabase
import at.htl.roomwithnavigation.entities.Word

@Database(entities = [Word::class], version = 1)
abstract class WordDatabase : RoomDatabase() {

}
```

[Word::class]
hier werden die Entities angegeben für die Datenbank

version

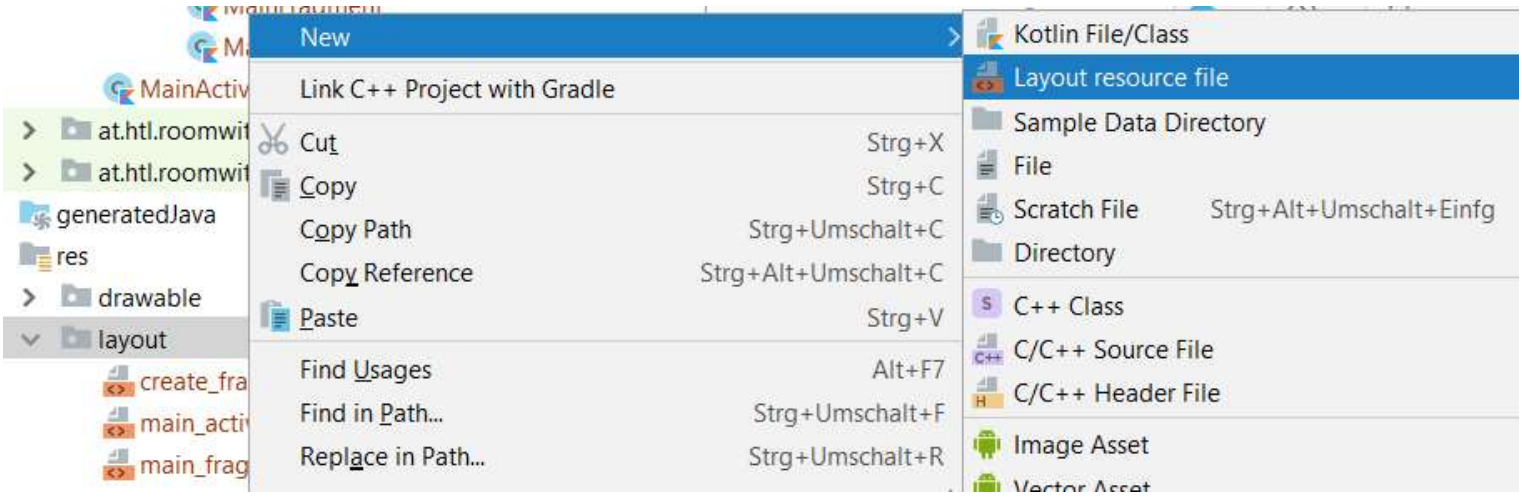
braucht man um Migrationen durchführen zu können

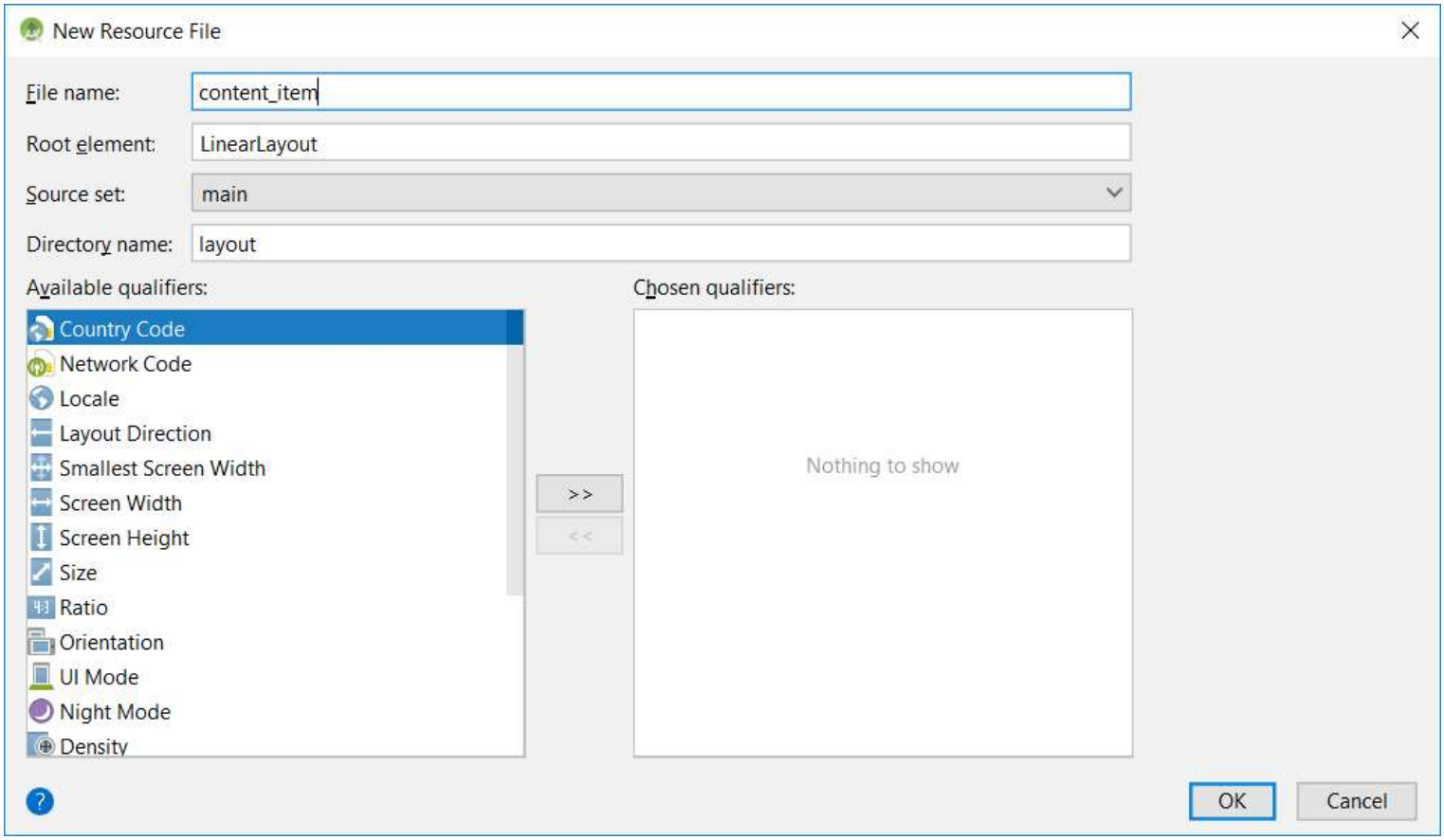
```
@Database(entities = [Word::class], version = 1)
abstract class WordDatabase : RoomDatabase() {

    abstract fun wordDao(): WordDao

    fun getWordDao(): WordDao = wordDao()

    companion object {
        private var INSTANCE: WordDatabase? = null

        fun getInstance(ctx: Context): WordDatabase {
            if (INSTANCE == null) {
                INSTANCE = Room.databaseBuilder(ctx,
                        WordDatabase::class.java,  name: "word_database")
                        .build()

            }
            return INSTANCE as WordDatabase
        }
    }
}

class WordRepository(application: Application) {
    private val wordDatabase: WordDatabase = WordDatabase.getInstance(application)
    private val wordDao: WordDao = wordDatabase.getWordDao()

    fun insert(word: Word) {
        thread {
            wordDao.insert(word)
        }
    }

    fun update(word: Word) {
        thread {
            wordDao.update(word)
        }
    }

    fun delete(word: Word) {
        thread {
            wordDao.delete(word)
        }
    }

    fun getAllLive(): LiveData<List<Word>> = wordDao.getAllLive()
}
```

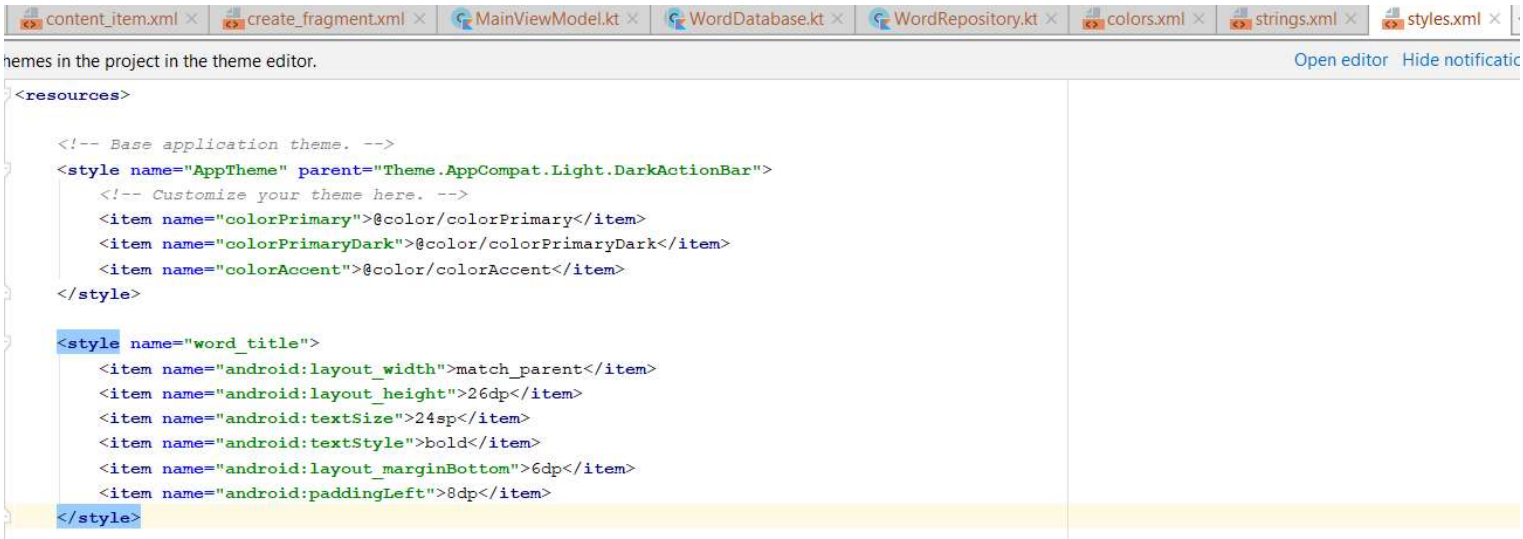
```
<android.support.v7.widget.RecyclerView
    android:id="@+id/recyclerview"
    android:layout_width="0dp"
    android:layout_height="0dp"
    android:layout_marginStart="8dp"
    android:layout_marginTop="8dp"
    android:layout_marginEnd="8dp"
    android:layout_marginBottom="8dp"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
```

löschen

```
<TextView
    android:id="@+id/message"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="MainFragment"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
```





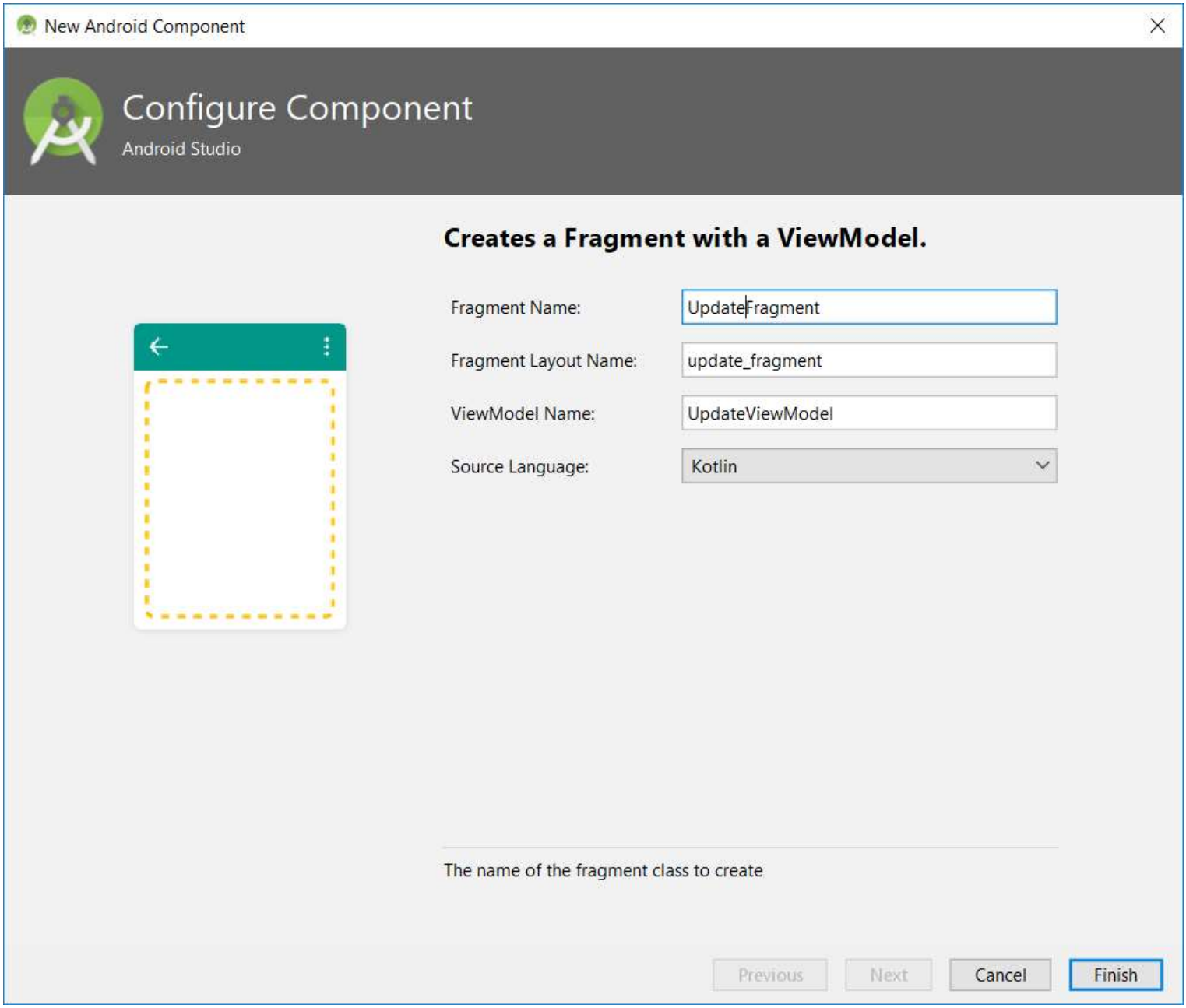


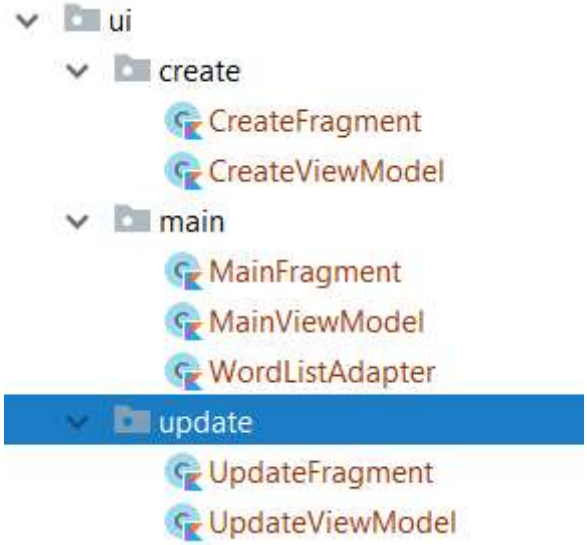
in der recyclerview

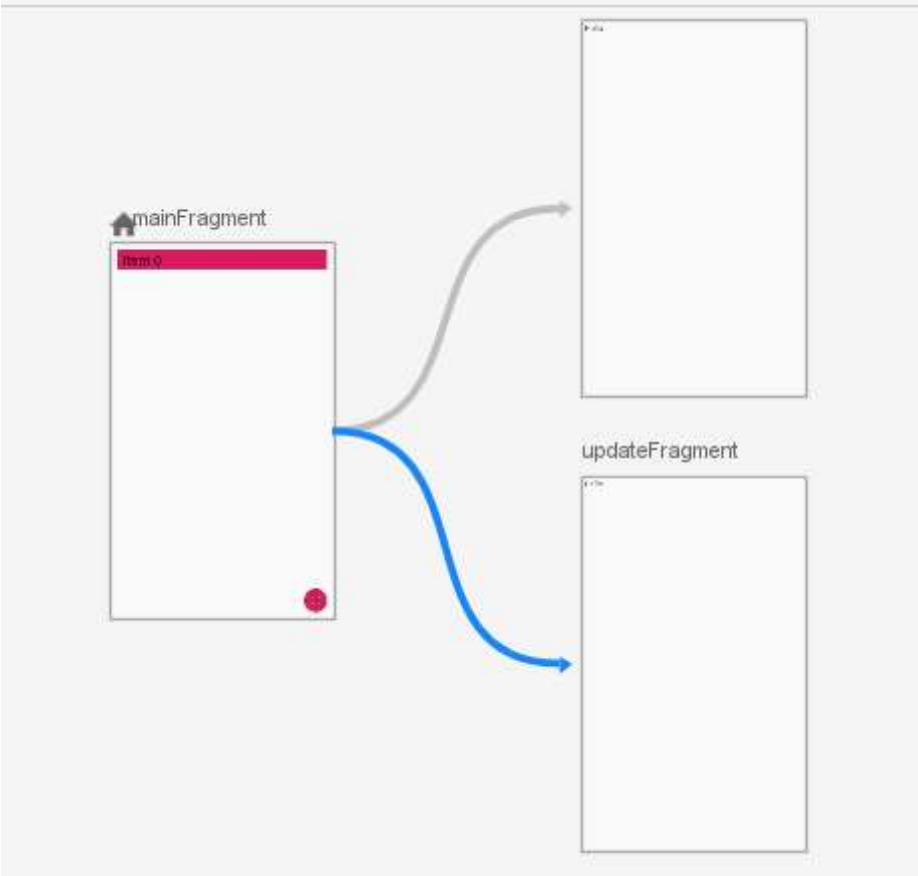
```
tools:listitem="@layout/content_item"
```

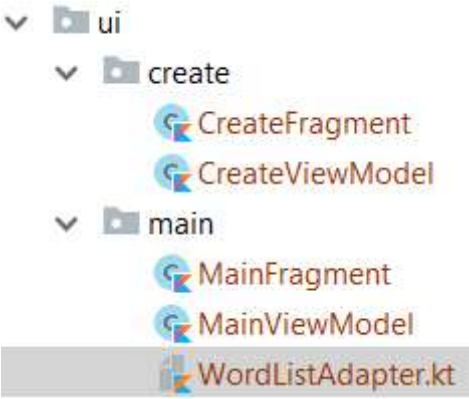
im content_item

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="match_parent"
    android:layout_height="wrap_content">

    <TextView
        android:id="@+id/textView"
        style="@style/word_title"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="@color/colorAccent"/>
</LinearLayout>
```


New Android Component
Configure Component
Android Studio
Creates a Fragment with a ViewModel.
Fragment Name:
UpdateFragment
Fragment Layout Name:
update_fragment
ViewModel Name:
UpdateViewModel
Source Language:
Kotlin
The name of the fragment class to create
Previous
Next
Cancel
Finish


ui
create
CreateFragment
CreateViewModel
main
MainFragment
MainViewModel
WordListAdapter
update
UpdateFragment
UpdateViewModel






```
override fun onActivityCreated(savedInstanceState: Bundle?) {
    super.onActivityCreated(savedInstanceState)
    viewModel = ViewModelProviders.of( fragment: this).get(MainViewModel::class.java)

    var adapter = WordListAdapter()
    recyclerview.adapter = adapter
    recyclerview.layoutManager = LinearLayoutManager(this.context)

    viewModel.getAllWords().observe( owner: this, Observer<List<Word>> { it: List<Word>?
        adapter.list = it!!
        recyclerview.adapter.notifyDataSetChanged()
    })
}
```

nt.xml × | MainFragment.kt × | create_fragment.xml × | strings.xml × | CreateViewModel.kt ×

anslations for all locales in the translations editor.

```
<resources>
    <string name="app_name">RoomWithNavigation</string>

    <!-- TODO: Remove or change this placeholder text -->
    <string name="action_settings">Settings</string>
    <string name="hint_word">Word...</string>
    <string name="button_save">Update</string>
    <string name="button_delete">Delete</string>
    <string name="button_create">Create</string>
    <string name="empty_not_saved">Word not saved because it is empty.</string>
    <string name="edit_text_name">Name</string>

</resources>
```

agment.xml × | MainFragment.kt × | create_fragment.xml × | strings.xml

```
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".ui.create.CreateFragment">

    <EditText
        android:id="@+id/editText"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:layout_marginEnd="8dp"
        android:layout_marginBottom="8dp"
        android:ems="10"
        android:hint="@string/edit_text_name"
        android:inputType="textPersonName"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:layout_marginEnd="8dp"
        android:text="@string/button_create"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/editText" />
</android.support.constraint.ConstraintLayout>
```

xml | strings.xml | CreateViewModel.kt | CreateFragment.kt | MainViewModel.kt

```kotlin
package at.htl.roomwithnavigation.ui.create

import android.app.Application
import android.arch.lifecycle.AndroidViewModel
import android.arch.lifecycle.ViewModel;
import at.htl.roomwithnavigation.entities.Word
import at.htl.roomwithnavigation.persistence.WordRepository

class CreateViewModel(application: Application) : AndroidViewModel(application) {

    private val mRepository: WordRepository = WordRepository(application)

    fun insert(word: Word) {
        mRepository.insert(word)
    }
}

override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
    super.onViewCreated(view, savedInstanceState)

    view.findViewById<Button>(R.id.button)?.setOnClickListener { it: View!
        viewModel.insert(Word( id: 0, editText.text.toString()))
        Navigation.findNavController(it).popBackStack()
    }
}

class UpdateViewModel(application: Application) : AndroidViewModel(application) {

    private var mRepository = WordRepository(application)
    var mWord: LiveData<Word>? = null

    fun setWord(id: Long) {
        mWord = mRepository.getSingleLive(id)
    }

    fun update(word: Word) {
        mRepository.update(word)
    }

    fun delete(word: Word) {
        mRepository.delete(word)
    }}
```

```xml
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.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=".ui.update.UpdateFragment">

<EditText
    android:id="@+id/editText"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginStart="8dp"
    android:layout_marginTop="8dp"
    android:layout_marginEnd="8dp"
    android:layout_marginBottom="8dp"
    android:ems="10"
    android:hint="@string/edit_text_name"
    android:inputType="textPersonName"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />

<Button
    android:id="@+id/button_update"
    android:layout_width="91dp"
    android:layout_height="wrap_content"
    android:layout_marginStart="8dp"
    android:layout_marginTop="8dp"
    android:layout_marginEnd="8dp"
    android:text="@string/button_save"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/editText" />

<Button
    android:id="@+id/button_delete"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginStart="8dp"
    android:layout_marginTop="8dp"
    android:layout_marginEnd="8dp"
    android:text="@string/button_delete"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/button_update" />
</android.support.constraint.ConstraintLayout>
```

```
class UpdateViewModel(application: Application) : AndroidViewModel(application) {

    private var mRepository = WordRepository(application)

    fun update(word: Word) {
        mRepository.update(word)
    }

    fun delete(word: Word) {
        mRepository.delete(word)
    }
}
```

im updateviewmodel

```
override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
    super.onViewCreated(view, savedInstanceState)

    val id = arguments?.getLong( key: "Id")!!
    editText.setText(arguments?.getString( key: "Word").toString())

    view.findViewById<Button>(R.id.button_update)?.setOnClickListener { it: View!
        viewModel.update(Word(id, editText.text.toString()))
        Navigation.findNavController(it).popBackStack()
    }

    view.findViewById<Button>(R.id.button_delete)?.setOnClickListener { it: View!
        viewModel.delete(Word(id, editText.text.toString()))
        Navigation.findNavController(it).popBackStack()
    }

}
```

Damit haben wir es auch schon geschafft!
Noch gutes Gelingen mit Android Jetpack 😉