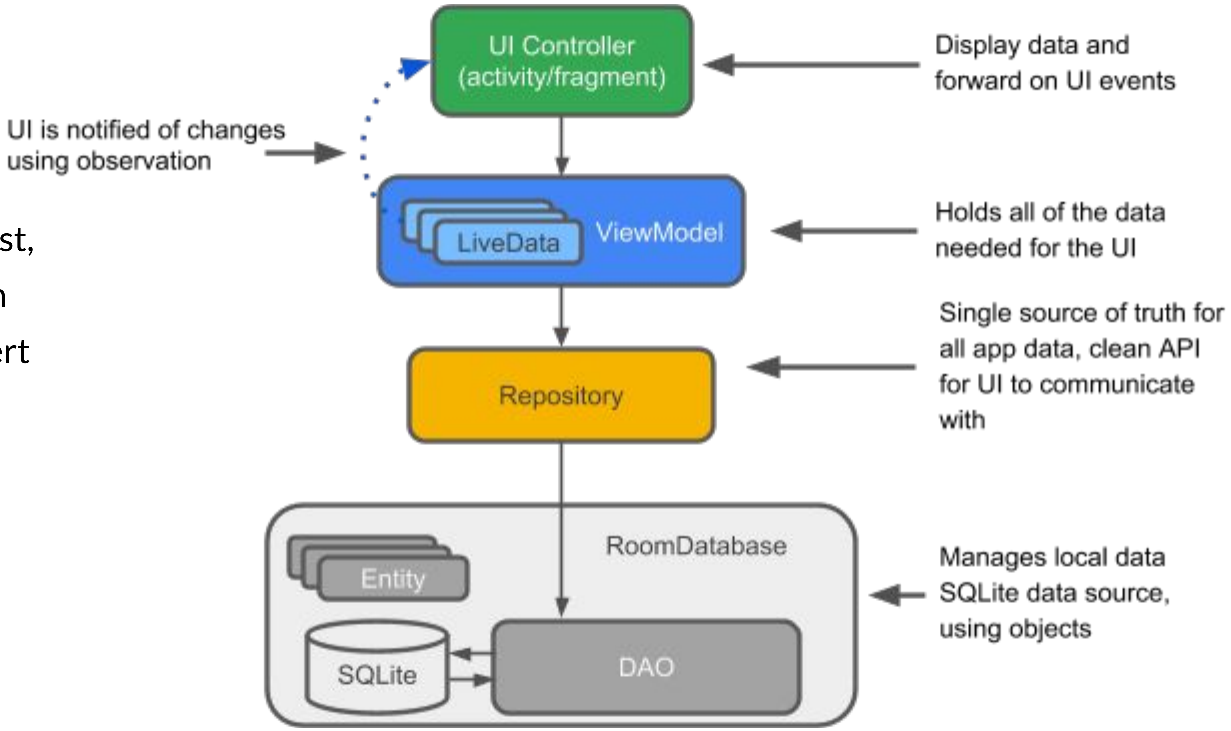
Room with Navigation in Kotlin

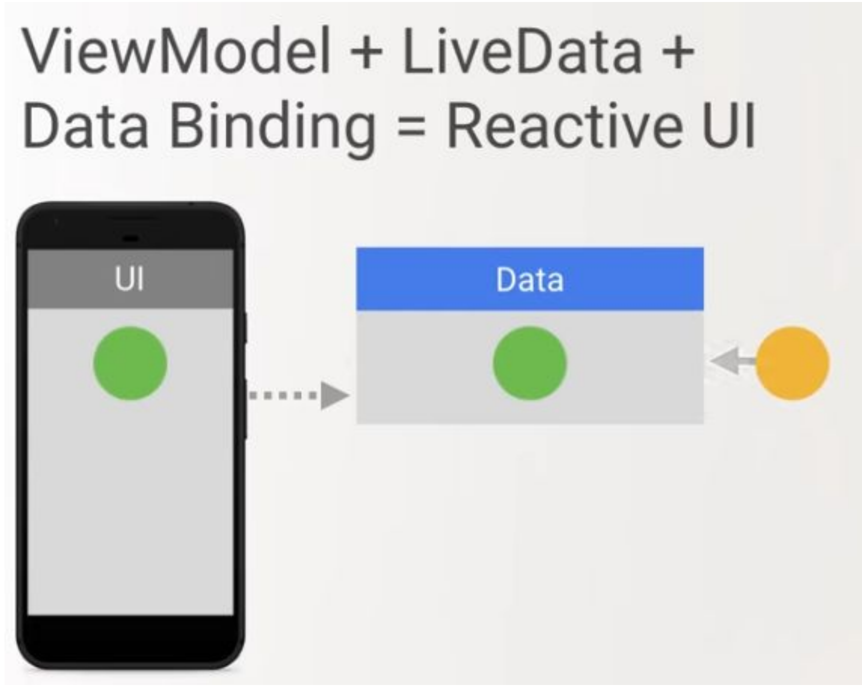
Architektur

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


UI Controller (activity/fragment)
Display data and forward on UI events
UI is notified of changes using observation
LiveData
ViewModel
Holds all of the data needed for the UI
Repository
Single source of truth for all app data, clean API for UI to communicate with
RoomDatabase
Entity
SQLite
DAO
Manages local data SQLite data source, using objects

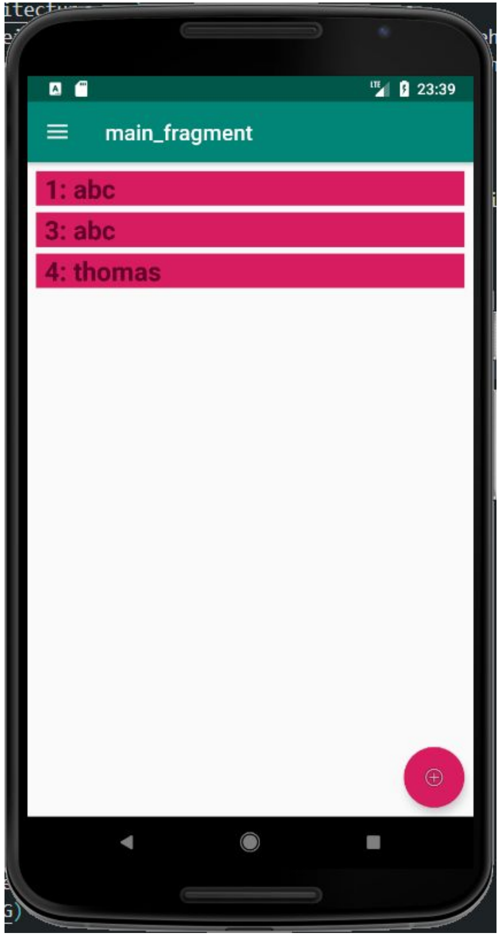
Warum LiveData

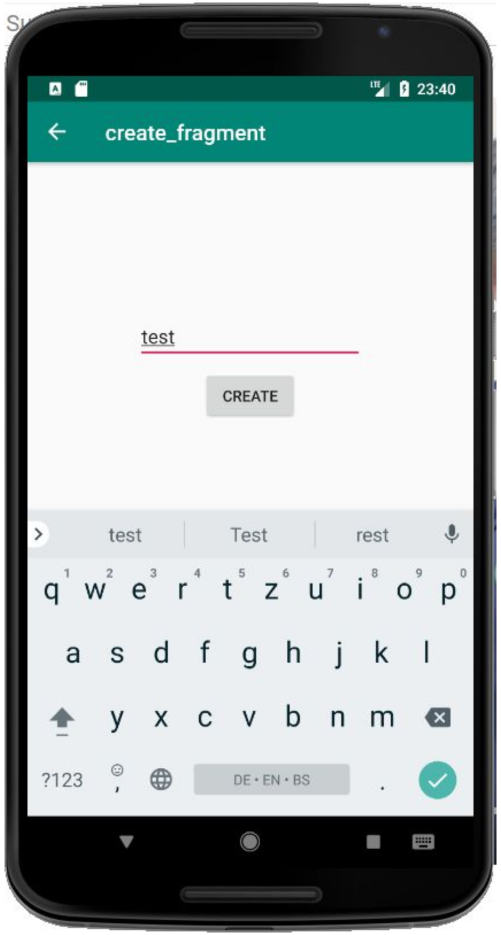
LiveData bekommt immer eine Benachrichtigung, wenn sich was in der Datenbank ändert (vergleichbar mit einer ObservableList). Das Beispiel wäre
noch mit Data Binding erweiterbar!

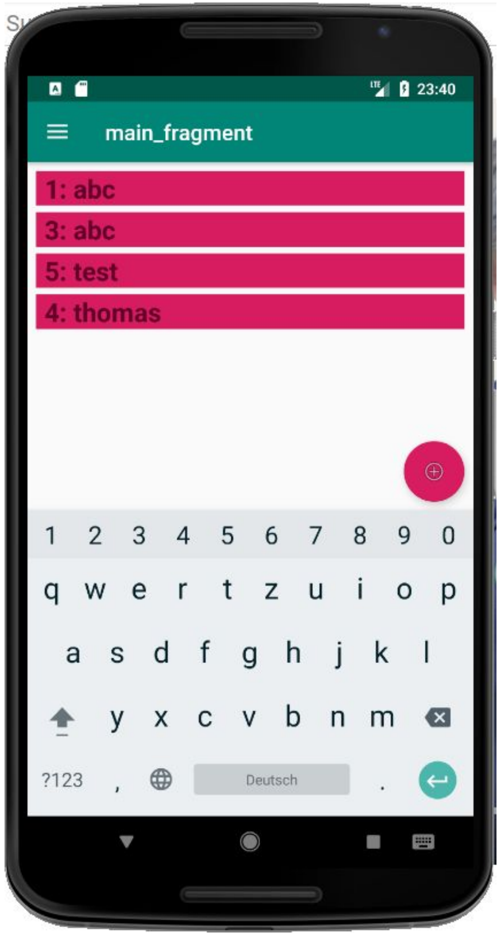


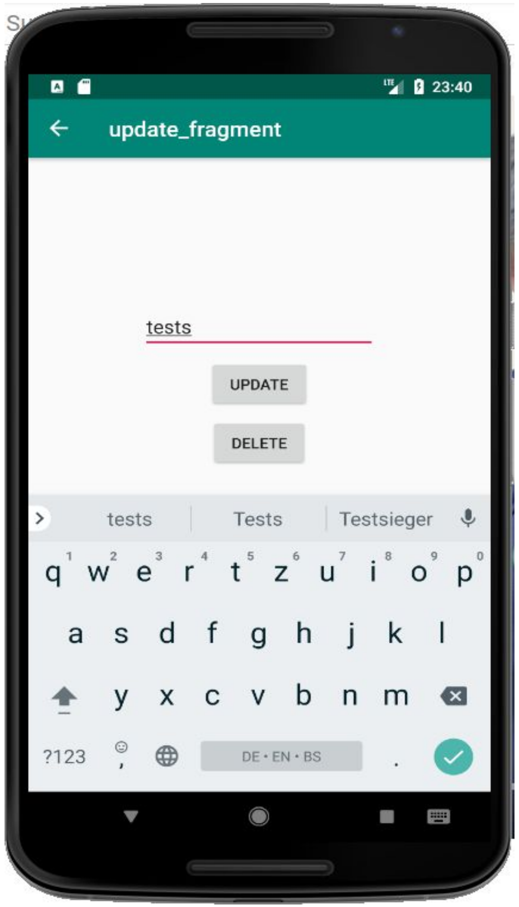
Was wir erreichen wollen

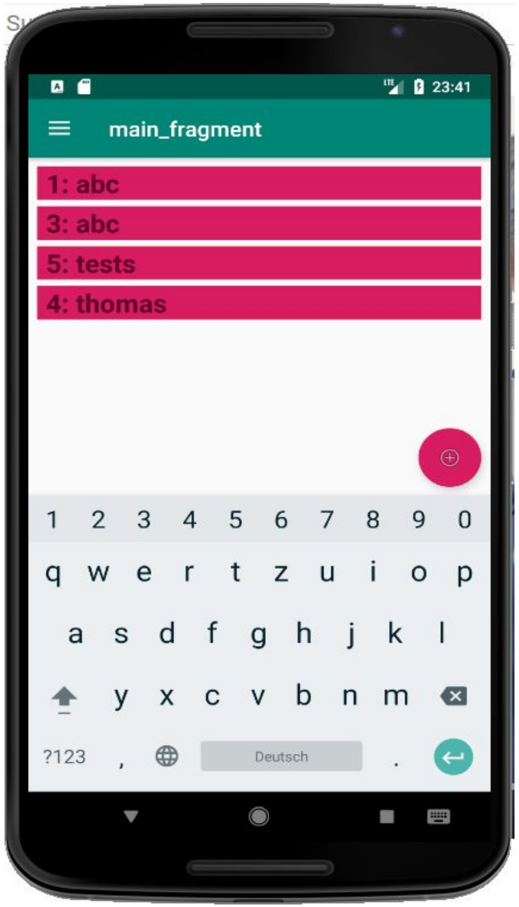
Man soll eine App entwickeln, welche mittels Room auf die SQLLite-Datenbank zugreift. Darauf soll man über die CRUD-Operationen zugreifen können.

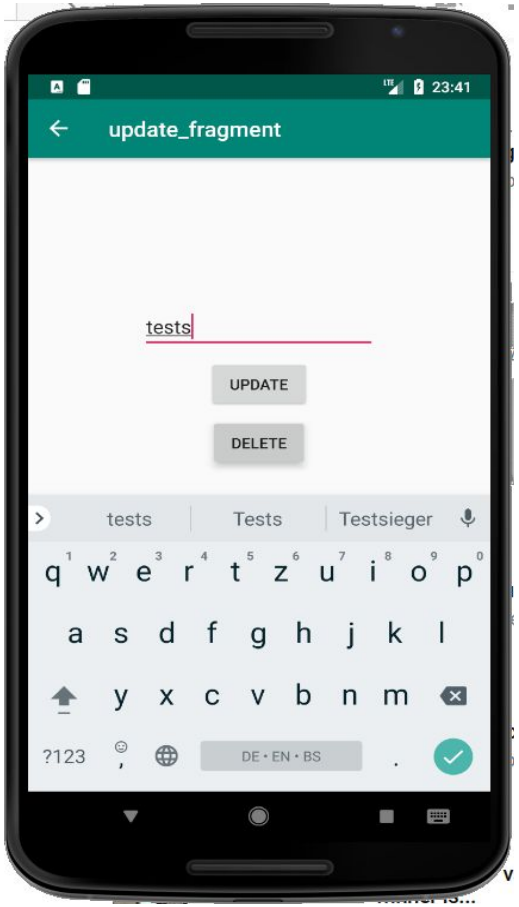

23:39
main_fragment
1: abc
3: abc
4: thomas

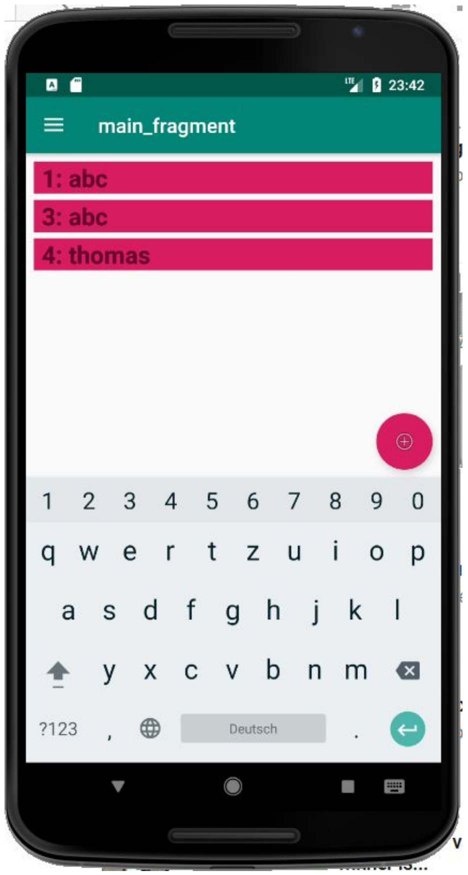

23:40
create_fragment
test
CREATE
test
Test
rest
DE • EN • BS

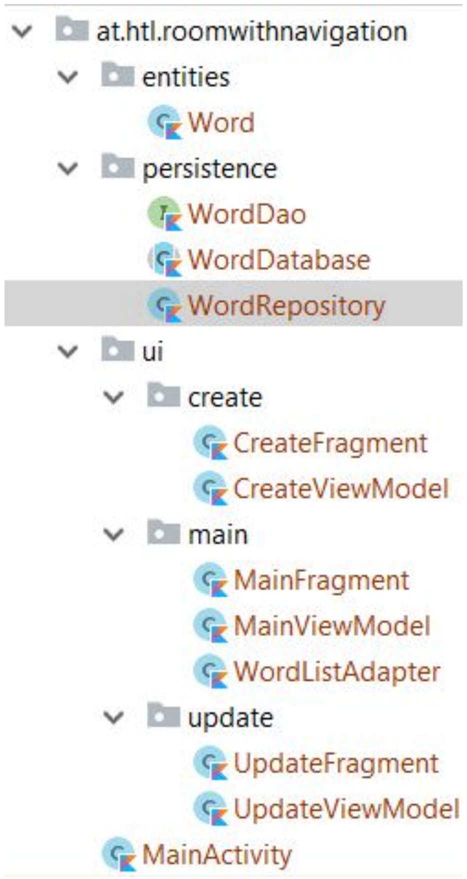

23:40
main_fragment
1: abc
3: abc
5: test
4: thomas
Deutsch


23:40
update_fragment
tests
UPDATE
DELETE


23:41
main_fragment
1: abc
3: abc
5: tests
4: thomas

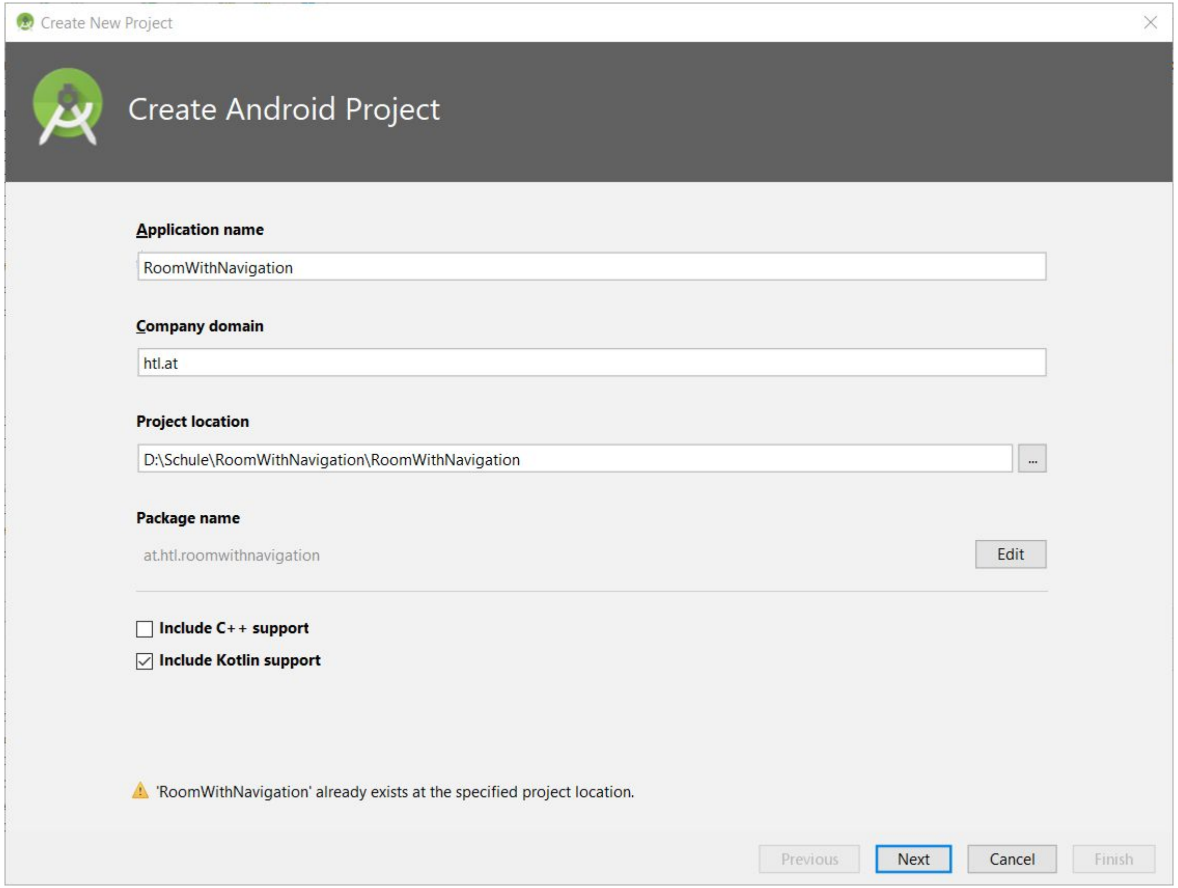

23:41
update_fragment
tests
UPDATE
DELETE





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

Los geht´s



Application name: RoomWithNavigation

Company domain: htl.at

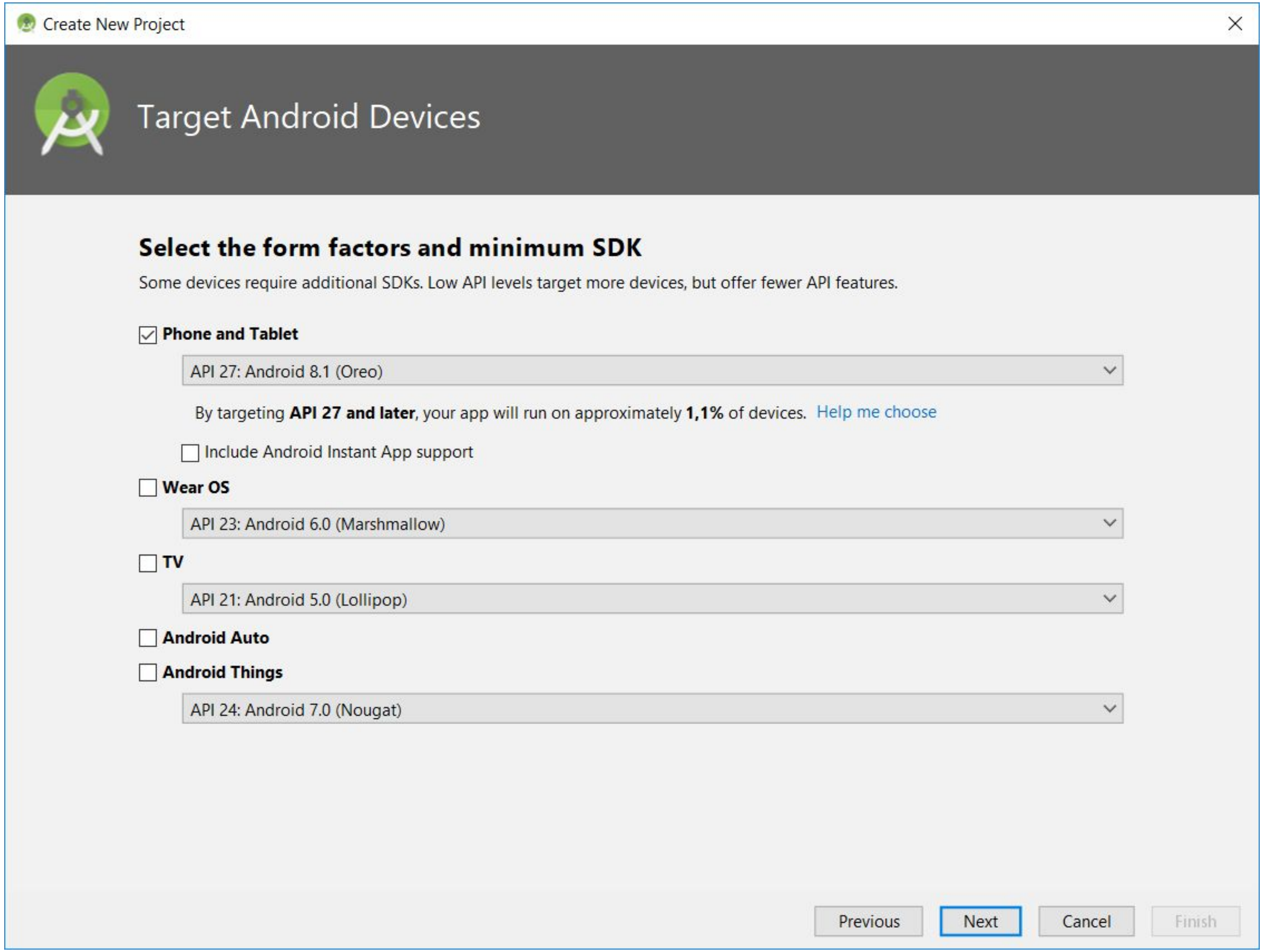
Project location: D:\Schule\RoomWithNavigation\RoomWithNavigation

Package name: at.htl.roomwithnavigation

☐ Include C++ support

☑ Include Kotlin support

'RoomWithNavigation' already exists at the specified project location.


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)

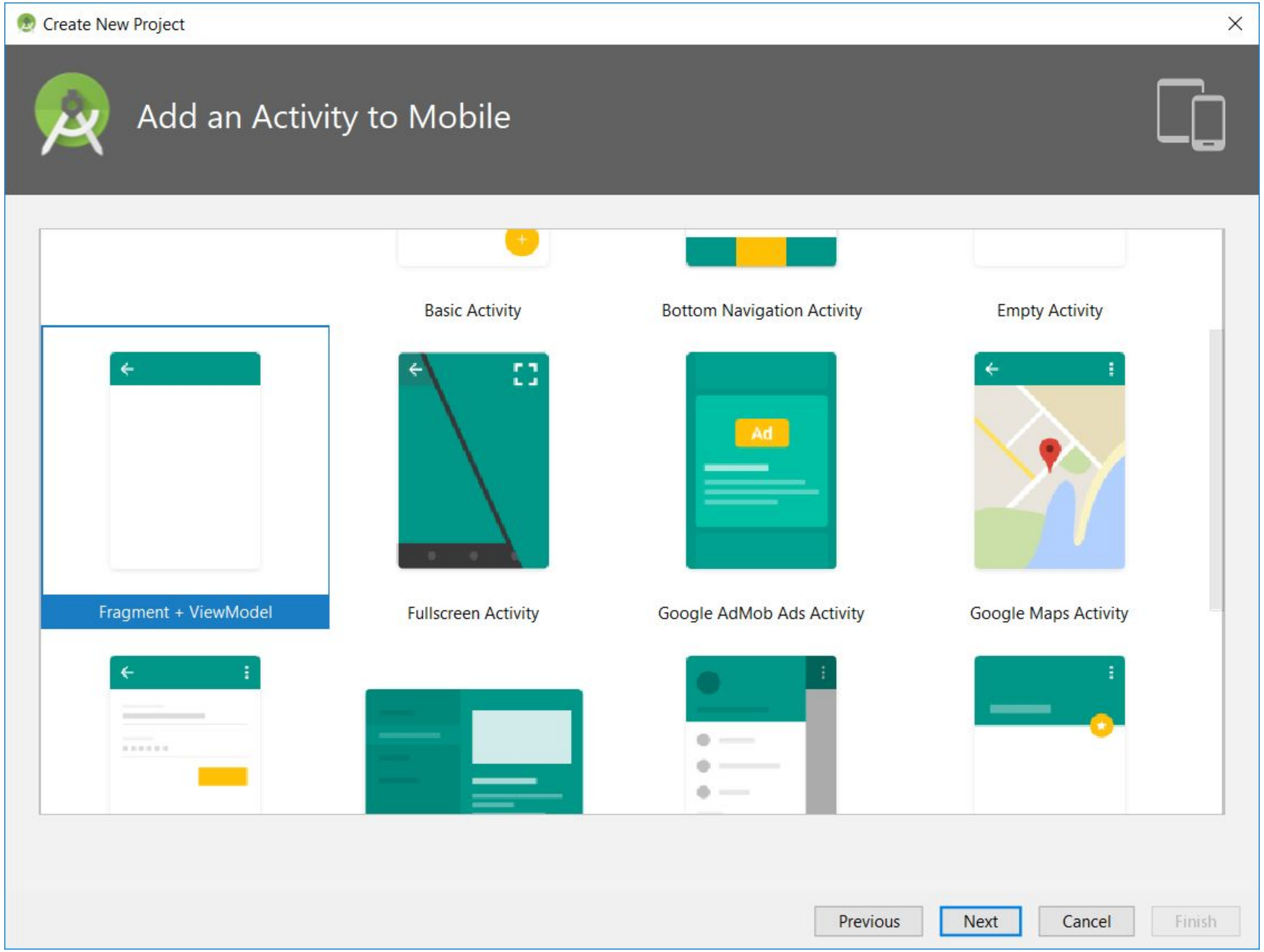
[ ] **TV**

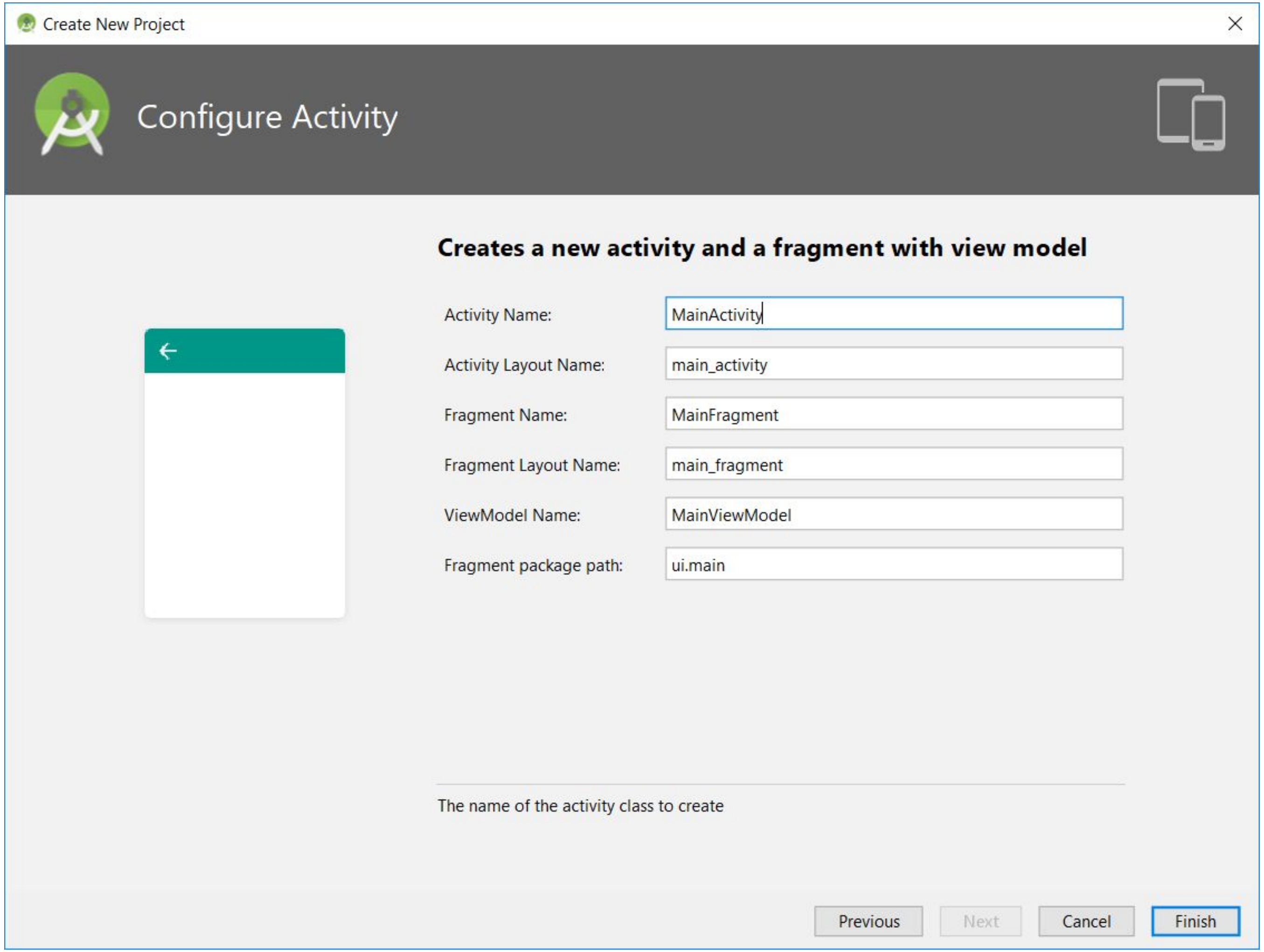
API 21: Android 5.0 (Lollipop)

[ ] **Android Auto**

[ ] **Android Things**

API 24: Android 7.0 (Nougat)


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

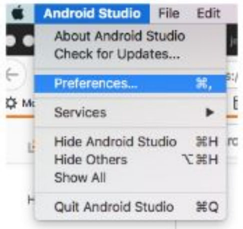


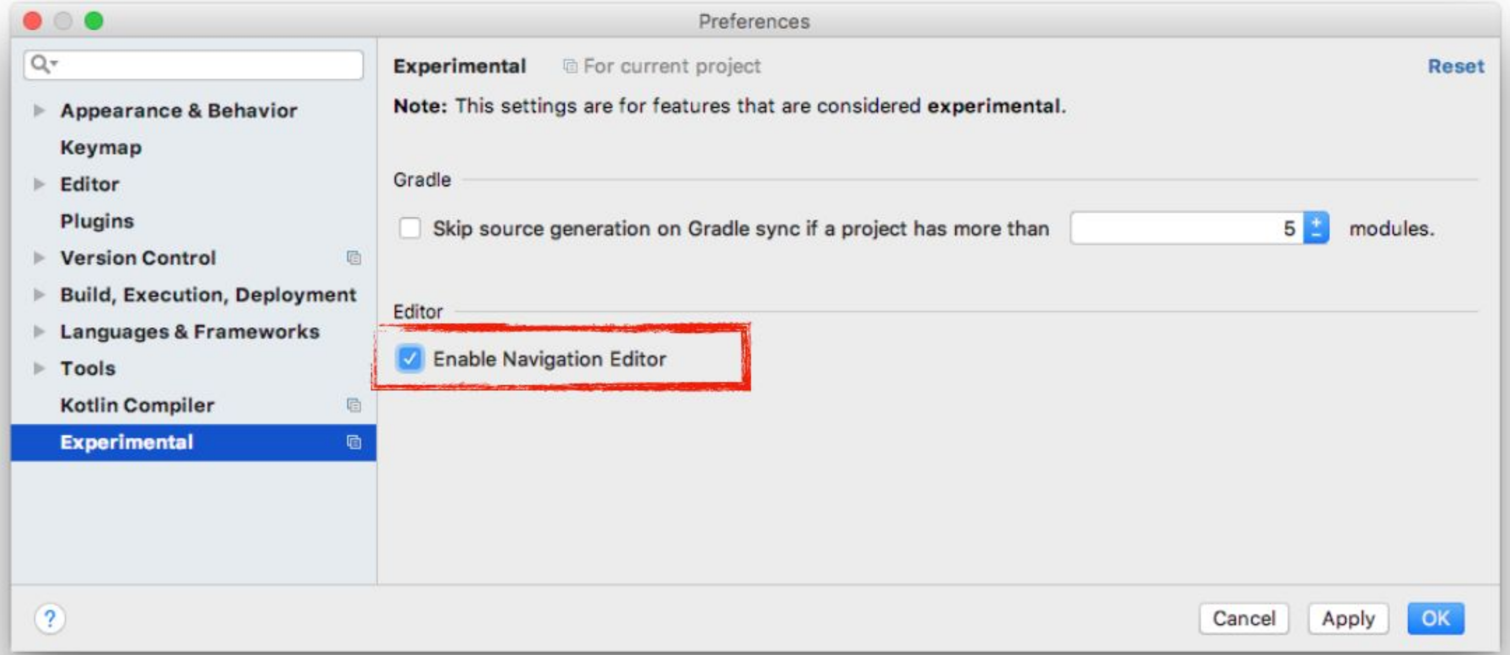
Creates a new activity and a fragment with view model

Activity Name:	MainActivity
Activity Layout Name:	main_activity
Fragment Name:	MainFragment
Fragment Layout Name:	main_fragment
ViewModel Name:	MainViewModel
Fragment package path:	ui.main

The name of the activity class to create

Kontrolliere ...





Experimental For current project

Note: This settings are for features that are considered **experimental**.

Gradle

[ ] Skip source generation on Gradle sync if a project has more than 5 modules.

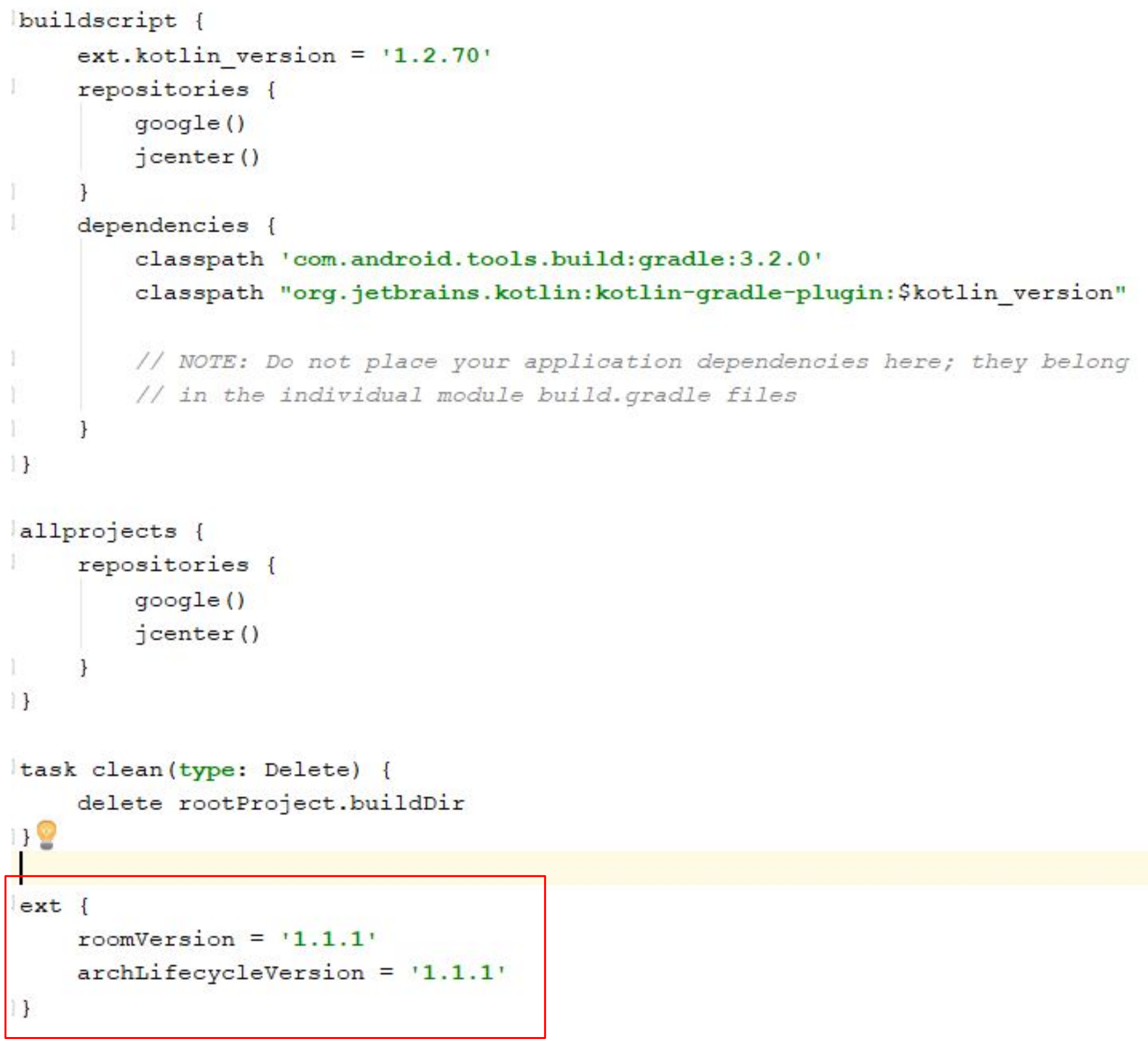
Editor

[✓] Enable Navigation Editor

Im gradle build (Project: ...)

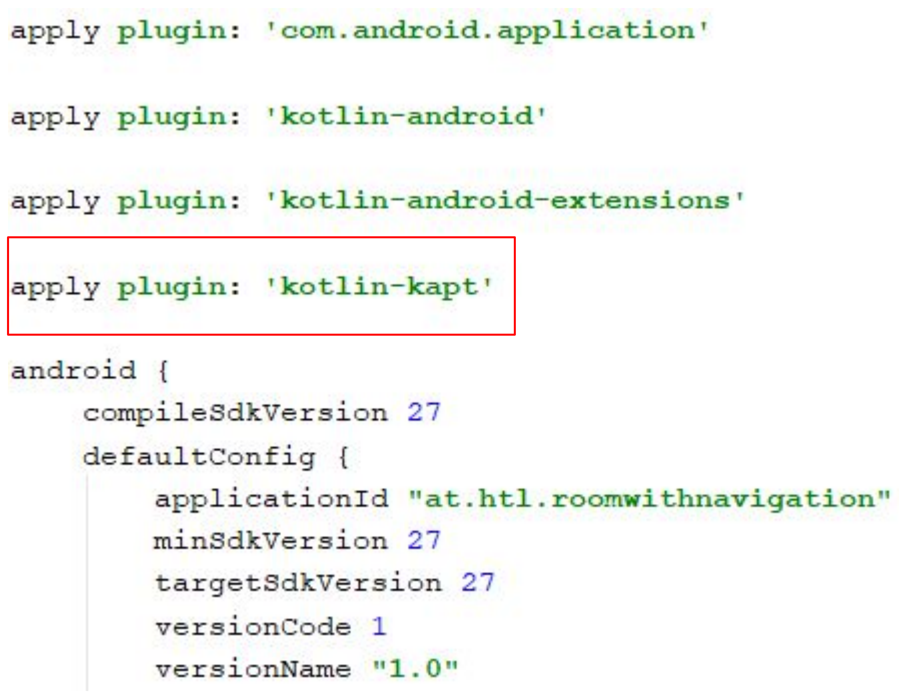
Hier zum Kopieren

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



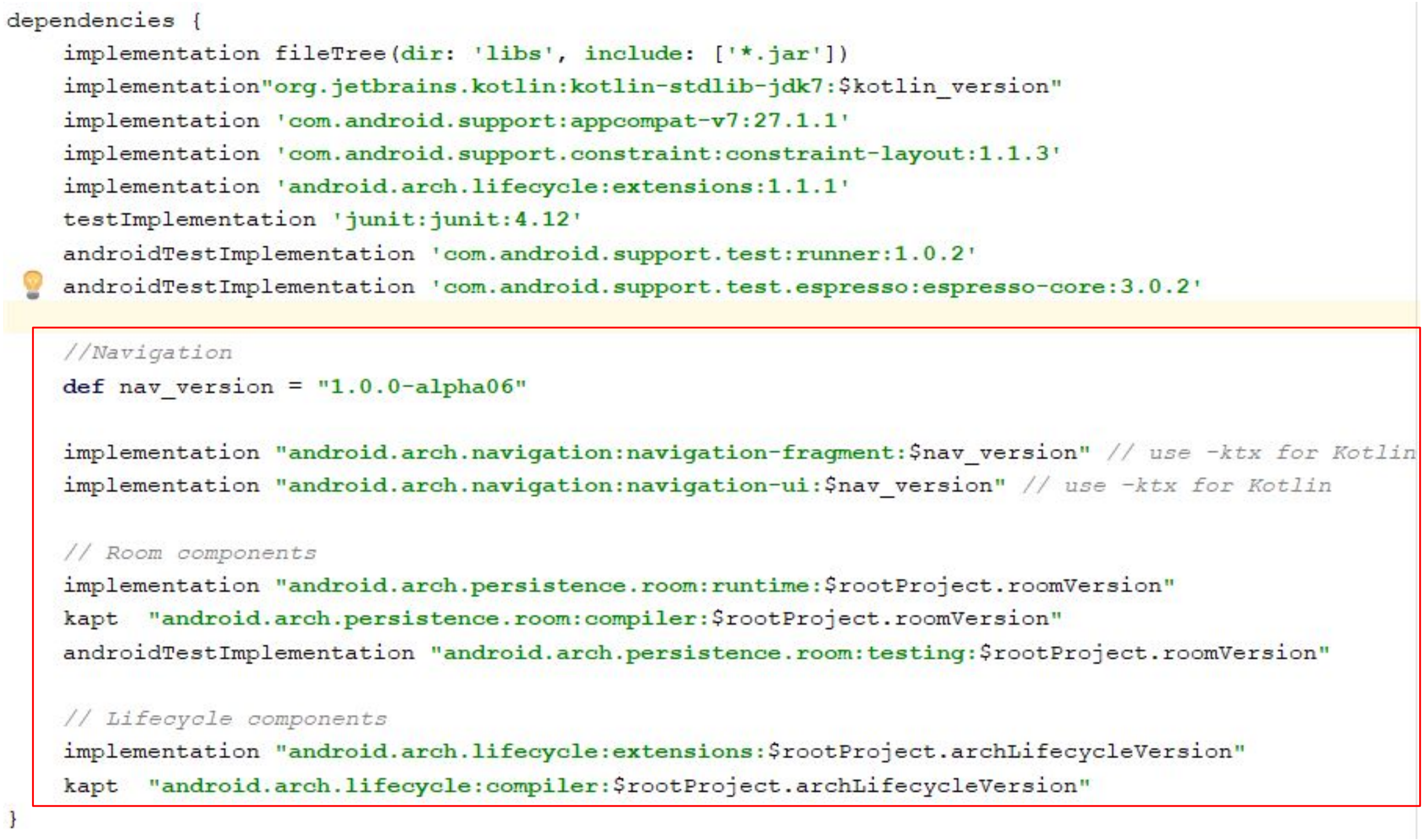
```
buildscript {
    ext.kotlin_version = '1.2.70'
    repositories {
        google()
        jcenter()
    }
    dependencies {
        classpath 'com.android.tools.build:gradle:3.2.0'
        classpath "org.jetbrains.kotlin:kotlin-gradle-plugin:$kotlin_version"

        // 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'
}
```

Im gradle build (app)



```
apply plugin: 'com.android.application'

apply plugin: 'kotlin-android'

apply plugin: 'kotlin-android-extensions'

apply plugin: 'kotlin-kapt'

android {
    compileSdkVersion 27
    defaultConfig {
        applicationId "at.htl.roomwithnavigation"
        minSdkVersion 27
        targetSdkVersion 27
        versionCode 1
        versionName "1.0"
```

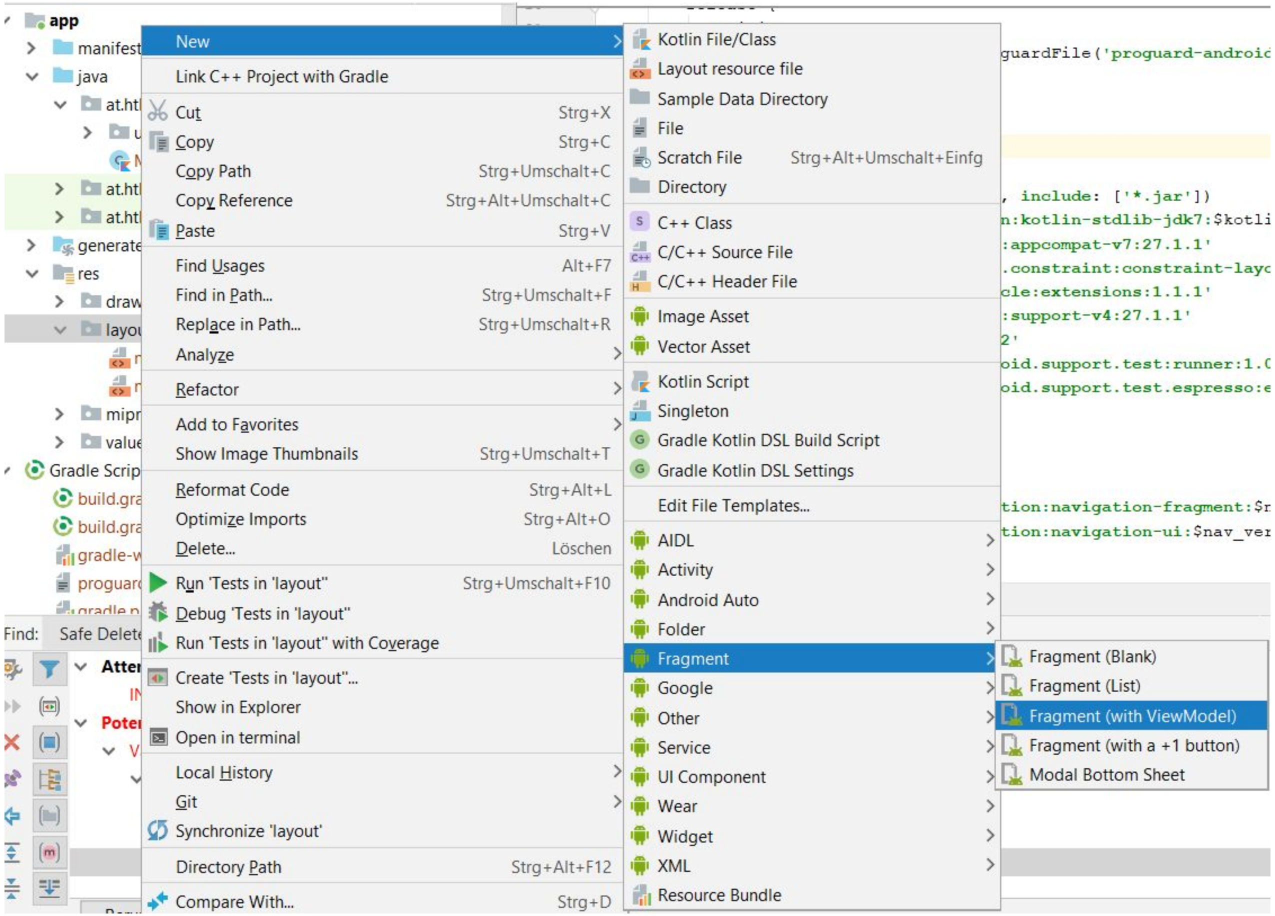
In der nächsten Folie zum Herauskopieren!



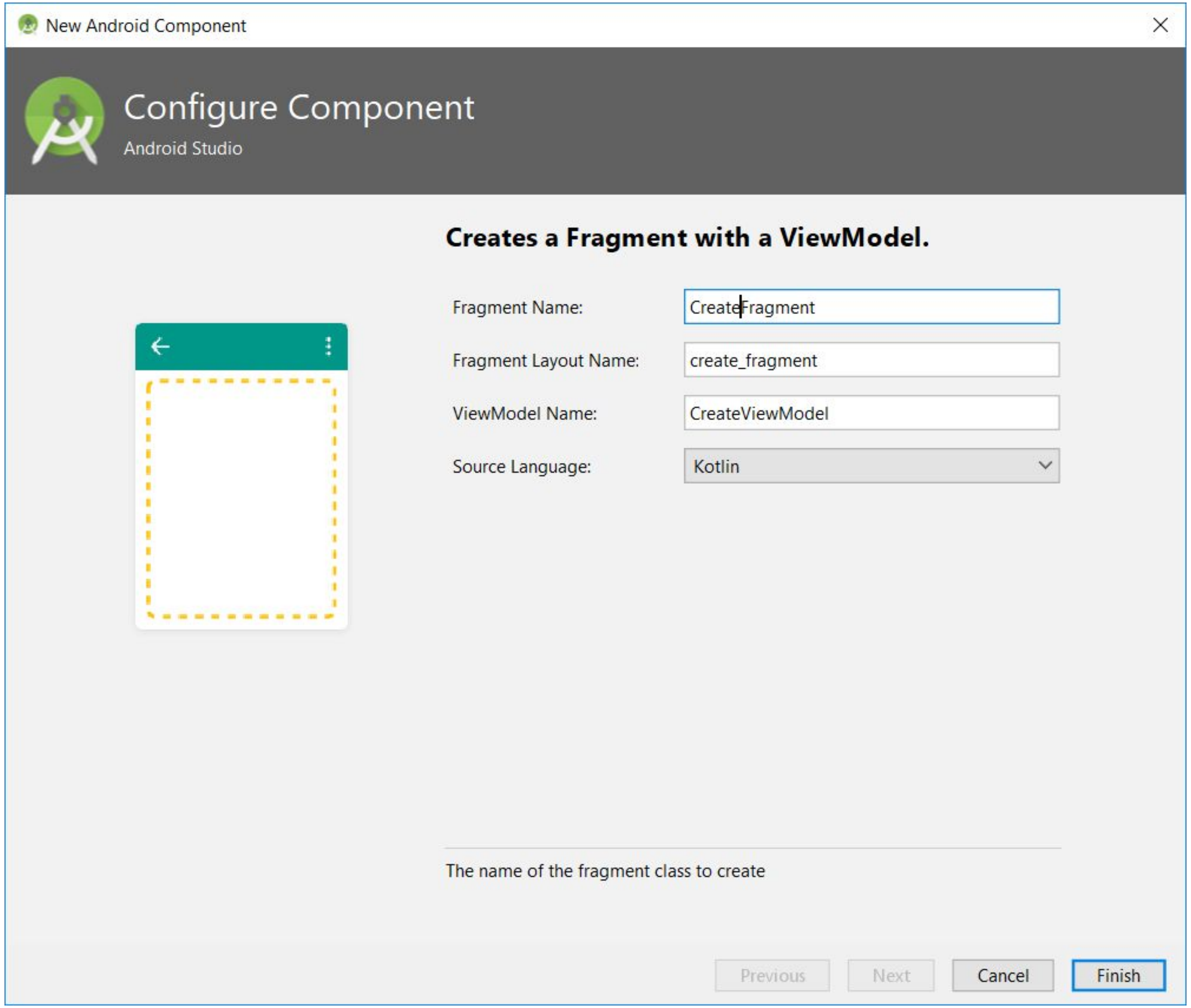
```
dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation"org.jetbrains.kotlin:kotlin-stdlib-jdk7:$kotlin_version"
    implementation 'com.android.support:appcompat-v7:27.1.1'
    implementation 'com.android.support.constraint:constraint-layout:1.1.3'
    implementation '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"
}
```

```
//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"
```

layout > new > fragment > fragment (with viewmodel)



```
guardFile('proguard-androic
, include: ['*.jar'])
n:kotlin-stdlib-jdk7:$kotli
:appcompat-v7:27.1.1'
.constraint:constraint-layc
cle:extensions:1.1.1'
:support-v4:27.1.1'
2'
oid.support.test:runner:1.(
oid.support.test.espresso:e
tion:navigation-fragment:$r
tion:navigation-ui:$nav_ver
```


New Android Component

Configure Component

Android Studio

Creates a Fragment with a ViewModel.

Fragment Name: CreateFragment

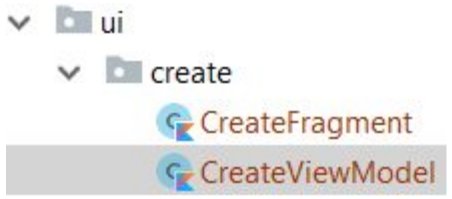
Fragment Layout Name: create_fragment

ViewModel Name: CreateViewModel

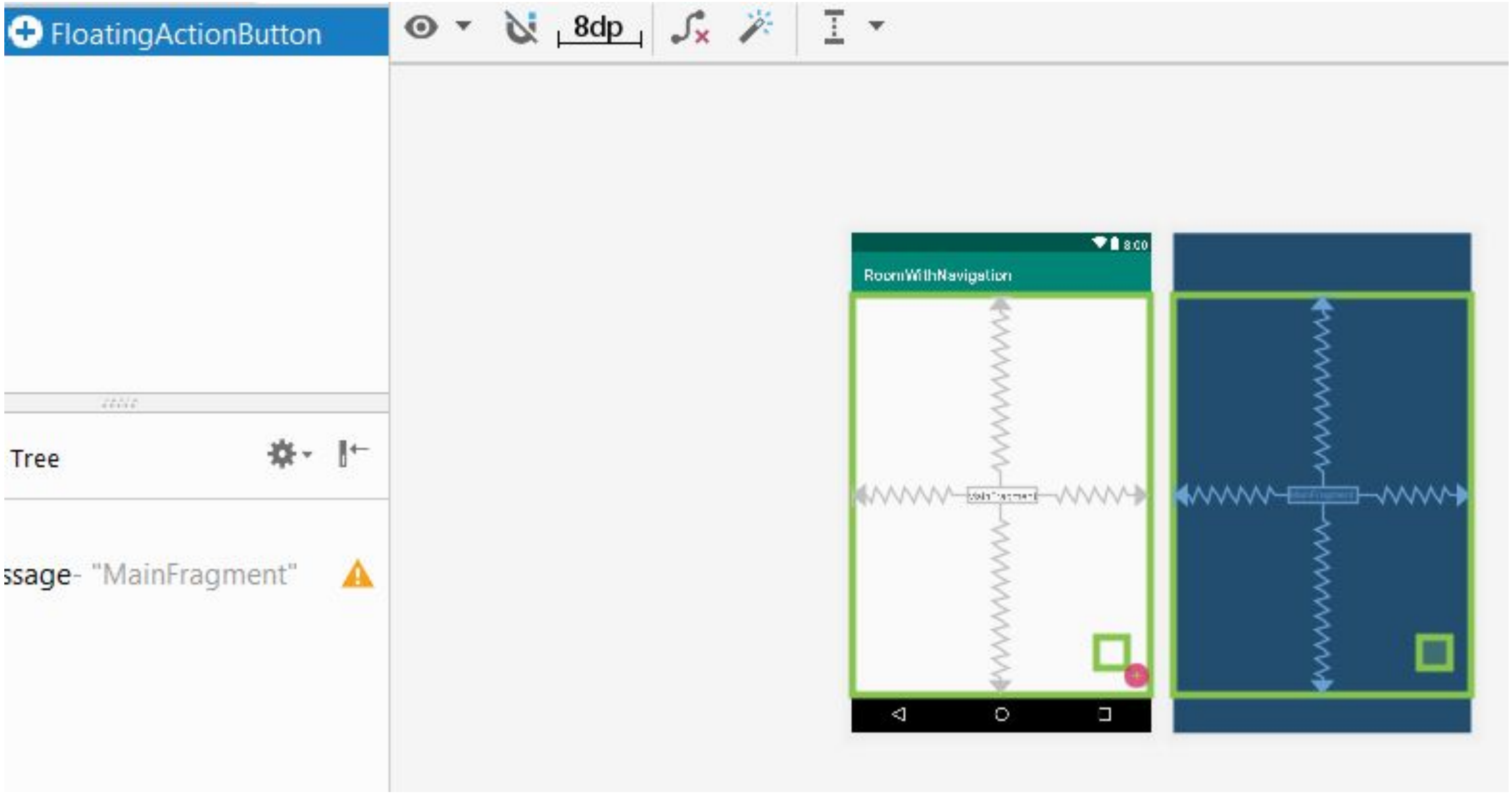
Source Language: Kotlin

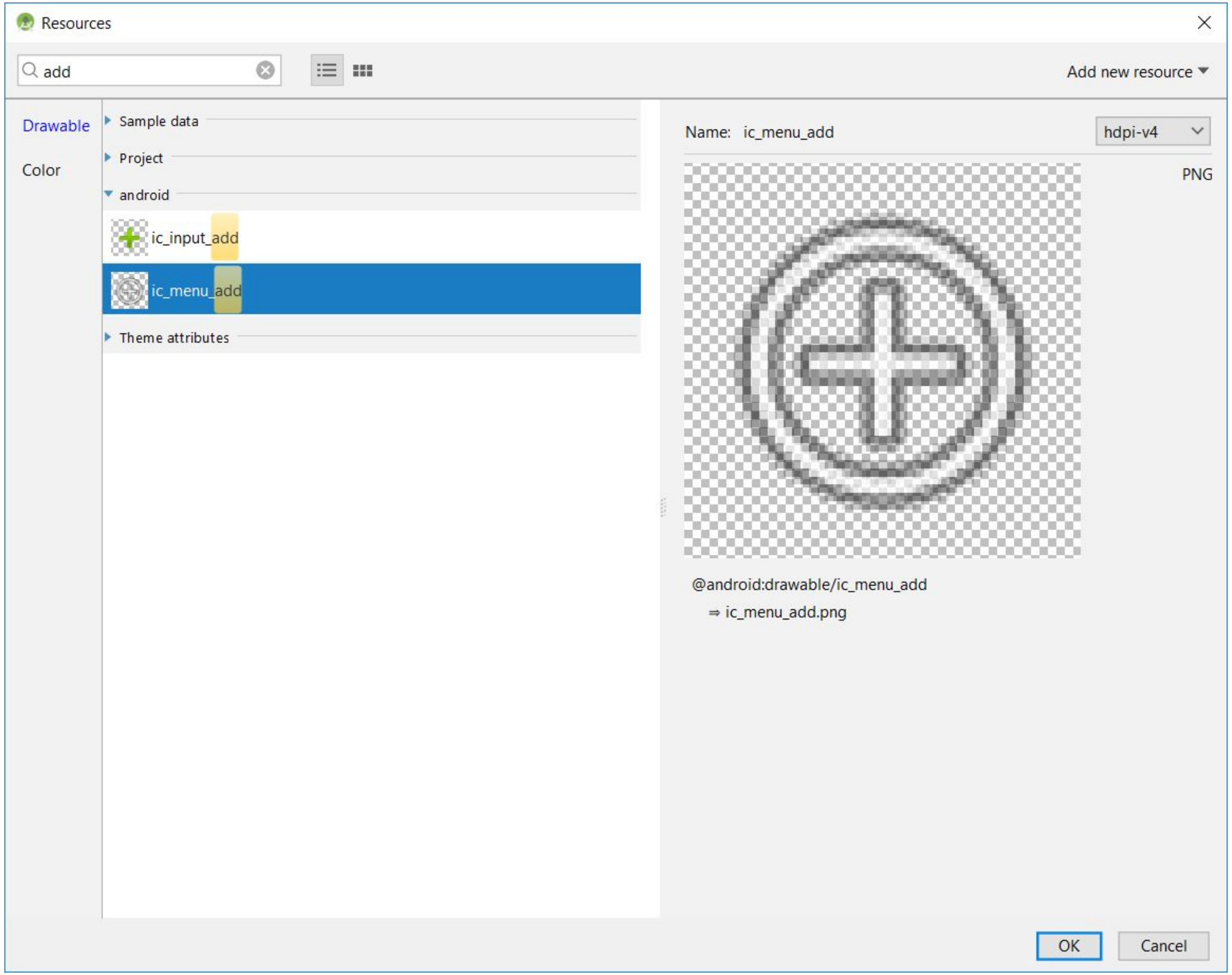
The name of the fragment class to create

Jetzt noch das Fragment in die Ordnerstruktur bringen



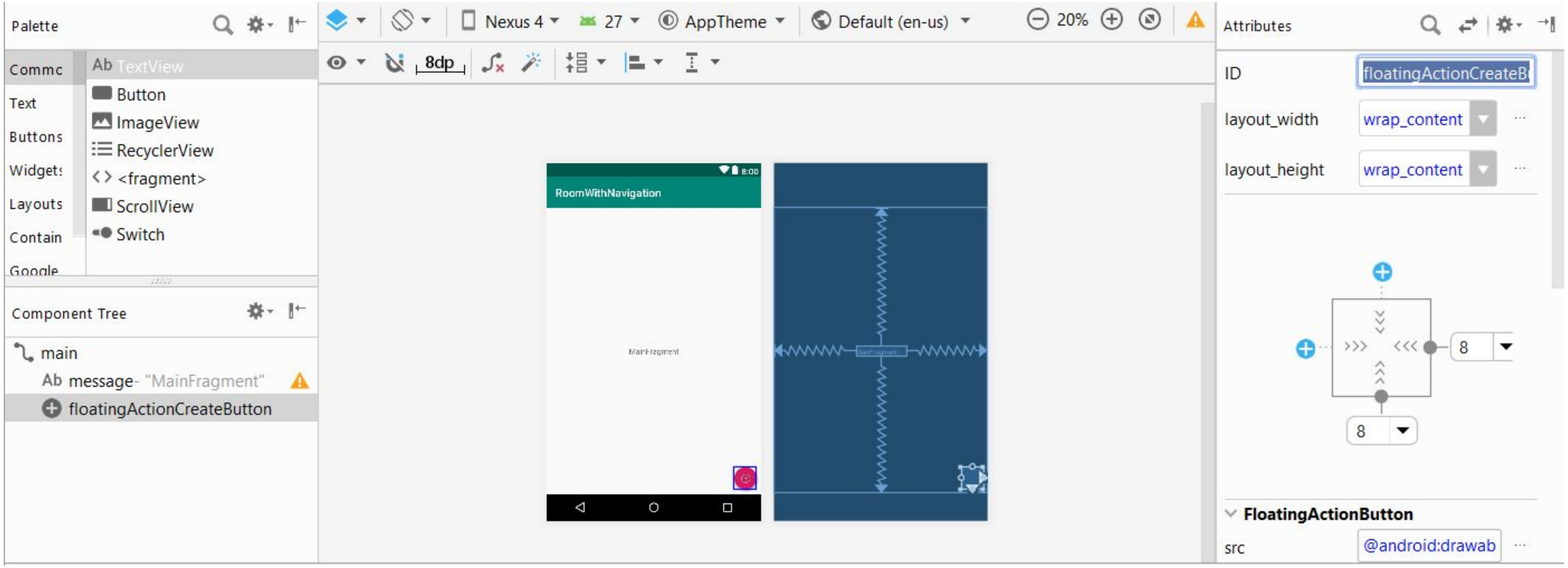
In das Fragment fügt man einen FloatingActionButton ein





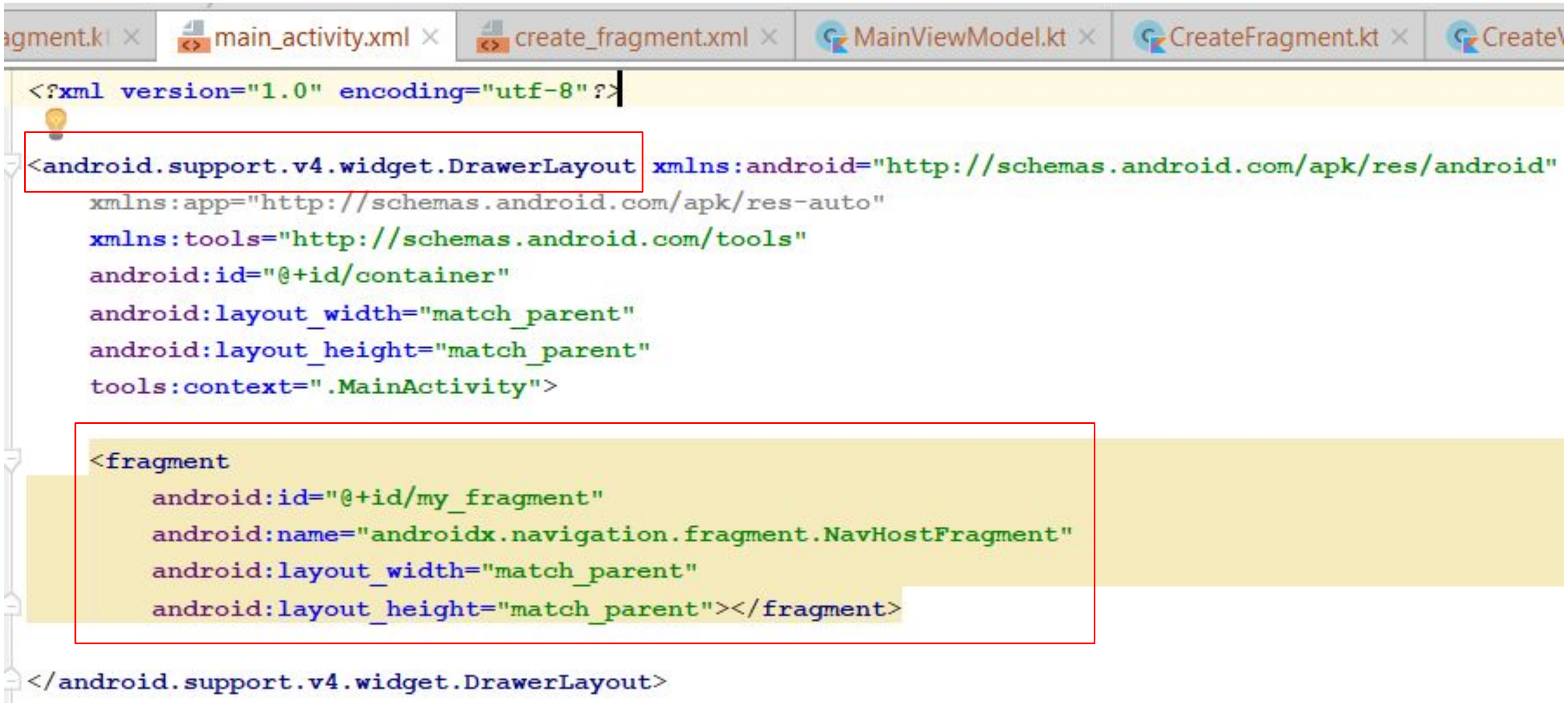
Name: ic_menu_add
hdpi-v4
PNG

@android:drawable/ic_menu_add
⇒ ic_menu_add.png

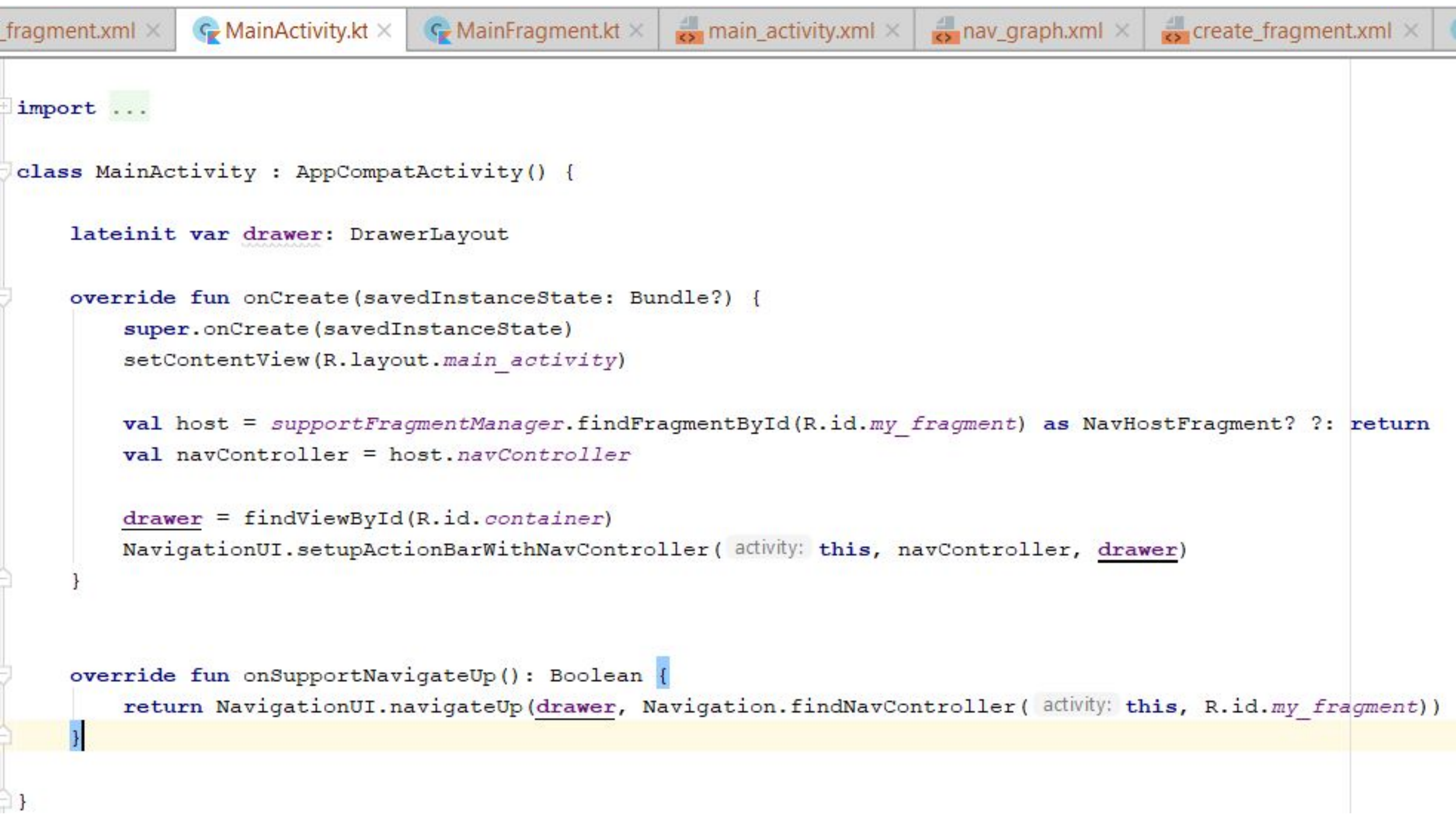


ID: floatingActionCreateB
layout_width: wrap_content
layout_height: wrap_content
src: @android:drawab

In der main_activity ändernt man auf DrawerLayout und fügt ein Fragment-Element hinzu



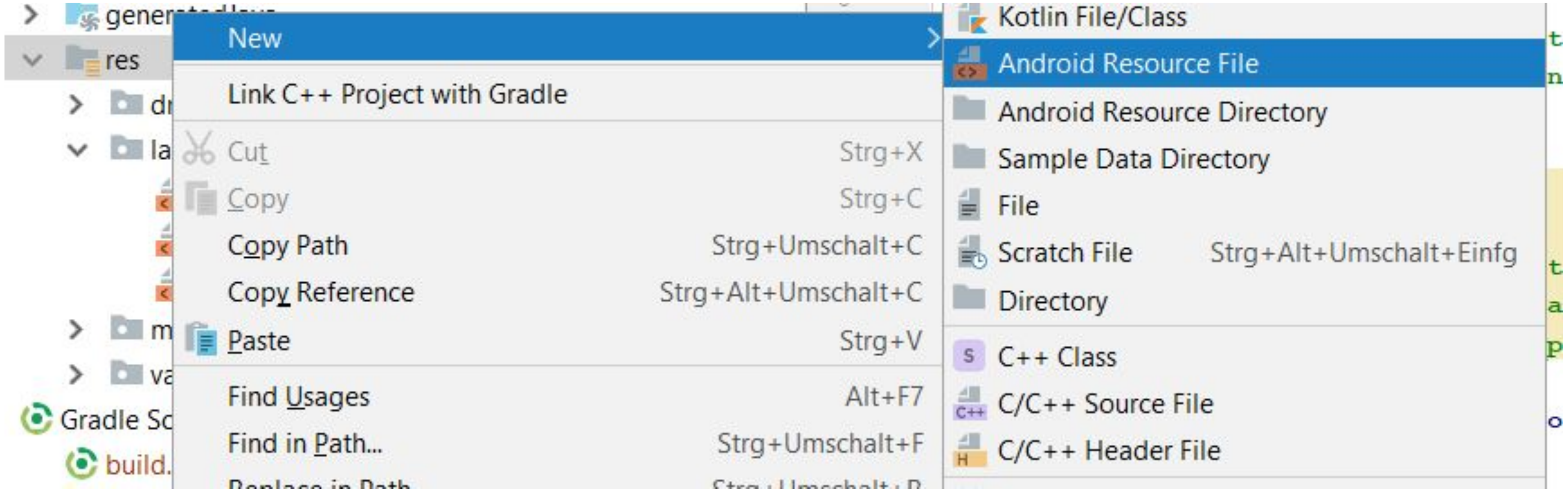
```
<?xml version="1.0" encoding="utf-8"?>

<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>
```

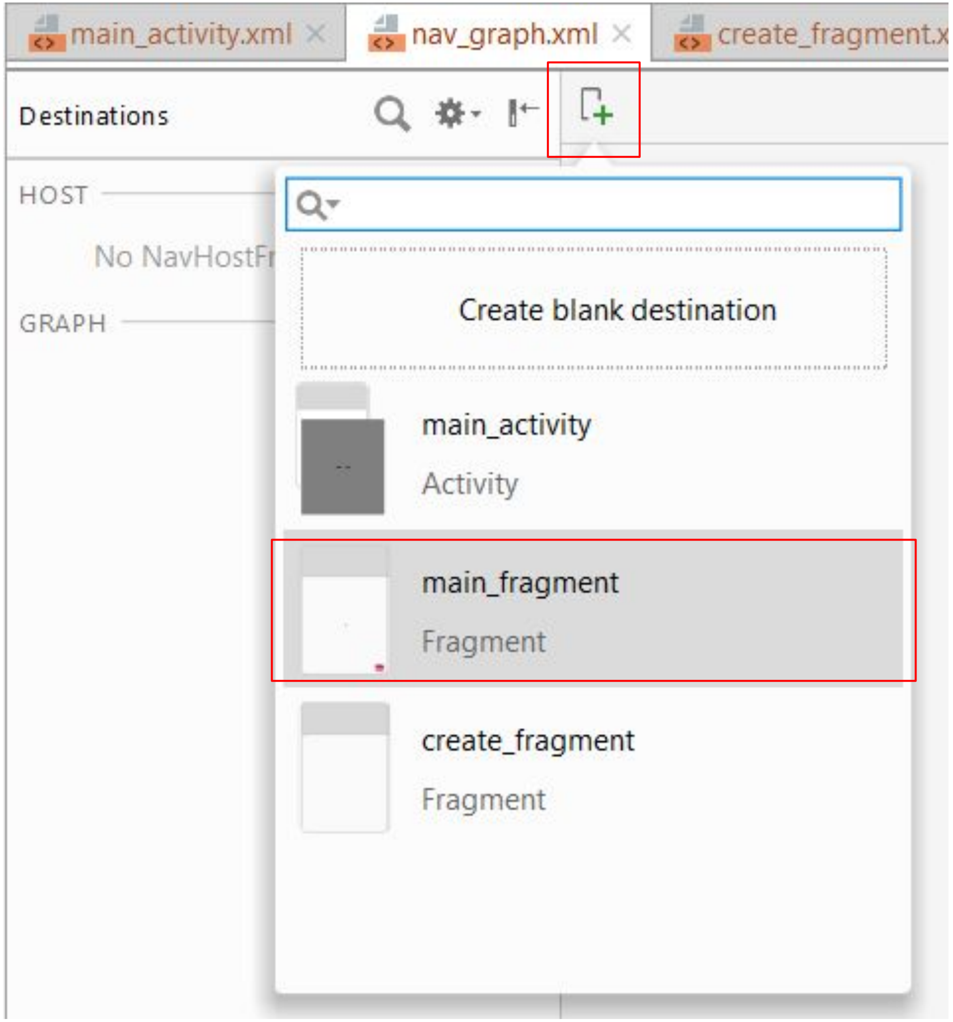


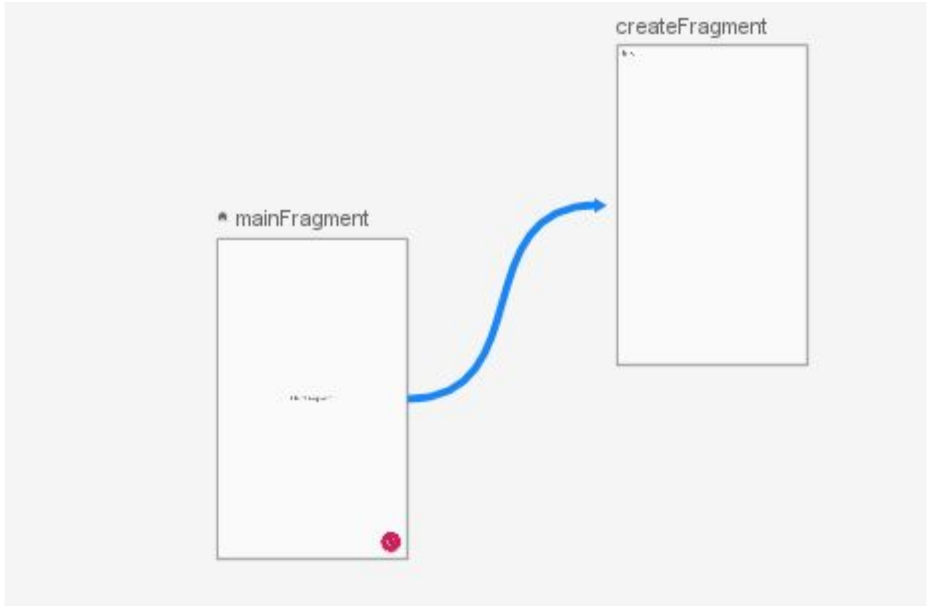
```kotlin
import ...

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))
    }

}
```

Hinzufügen des Navigation Graphs

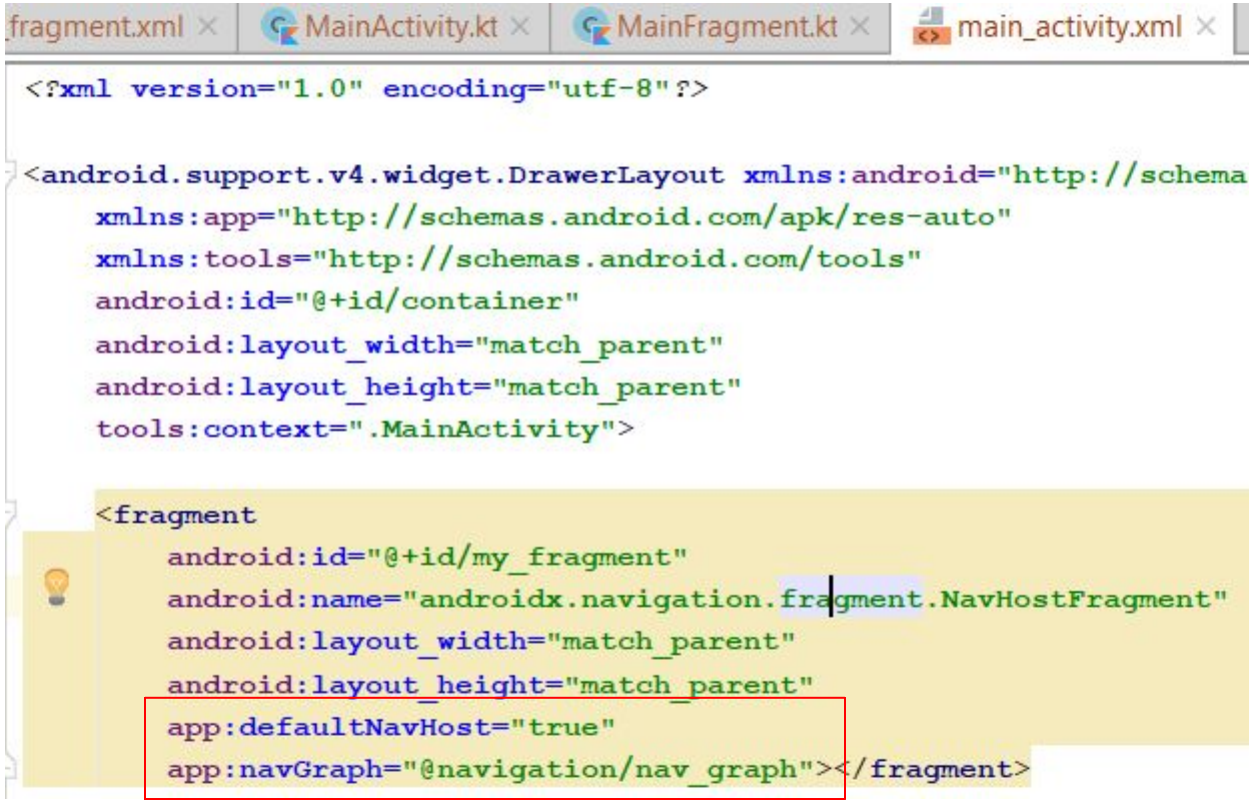
res > new > android resource file




main_activity.xml
nav_graph.xml
create_fragment.x
Destinations
HOST
No NavHostFr
GRAPH
Create blank destination
main_activity
Activity
main_fragment
Fragment
create_fragment
Fragment

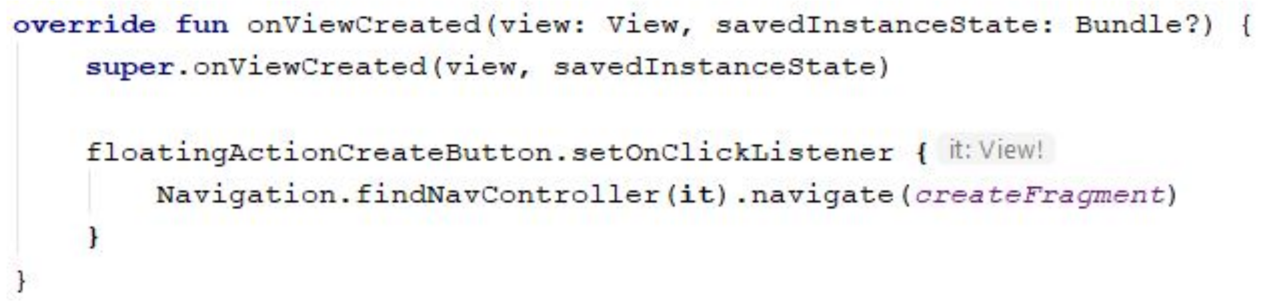

createFragment
mainFragment

In der main_activity setzt man jetzt fest, dass das Fragment der erste Eintrittspunkt des Graphen ist



```
<?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>
```

In MainFragment.kt fügt man noch diese Methode ein
Damit setzt man den onClickListener auf den FloatingActionButton

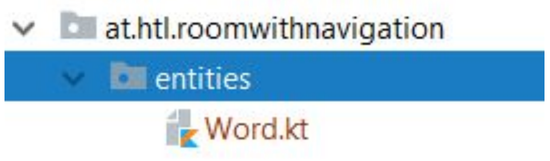


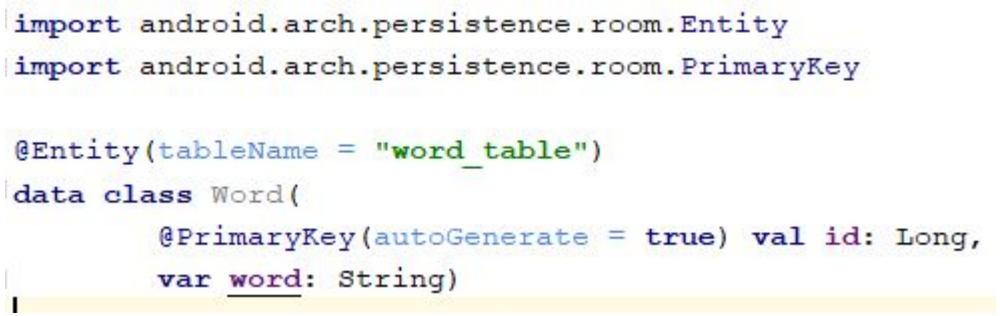
```kotlin
override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
    super.onViewCreated(view, savedInstanceState)

    floatingActionCreateButton.setOnClickListener { it: View!
        Navigation.findNavController(it).navigate(createFragment)
    }
}
```

Jetzt kann man vom main- zum create-Fragment wechseln

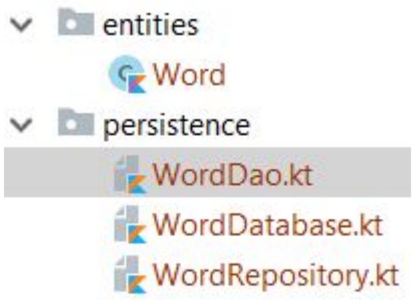
Nun geht es zu der Erstellung der nötigen Files für Room

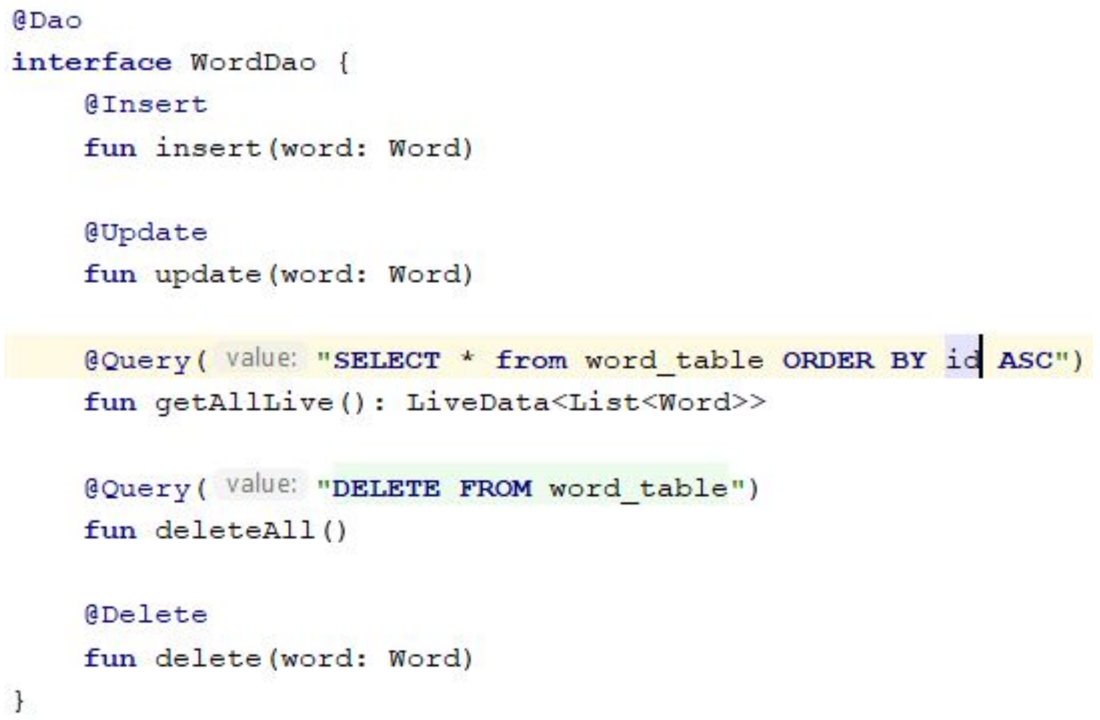
Zuerst die Word Entität





```
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)
```

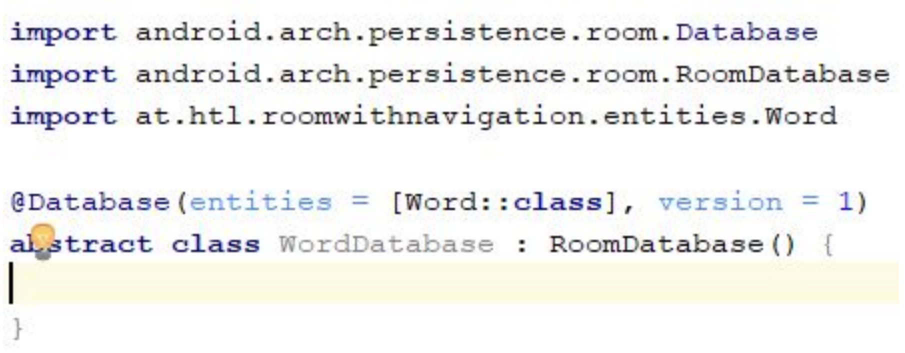




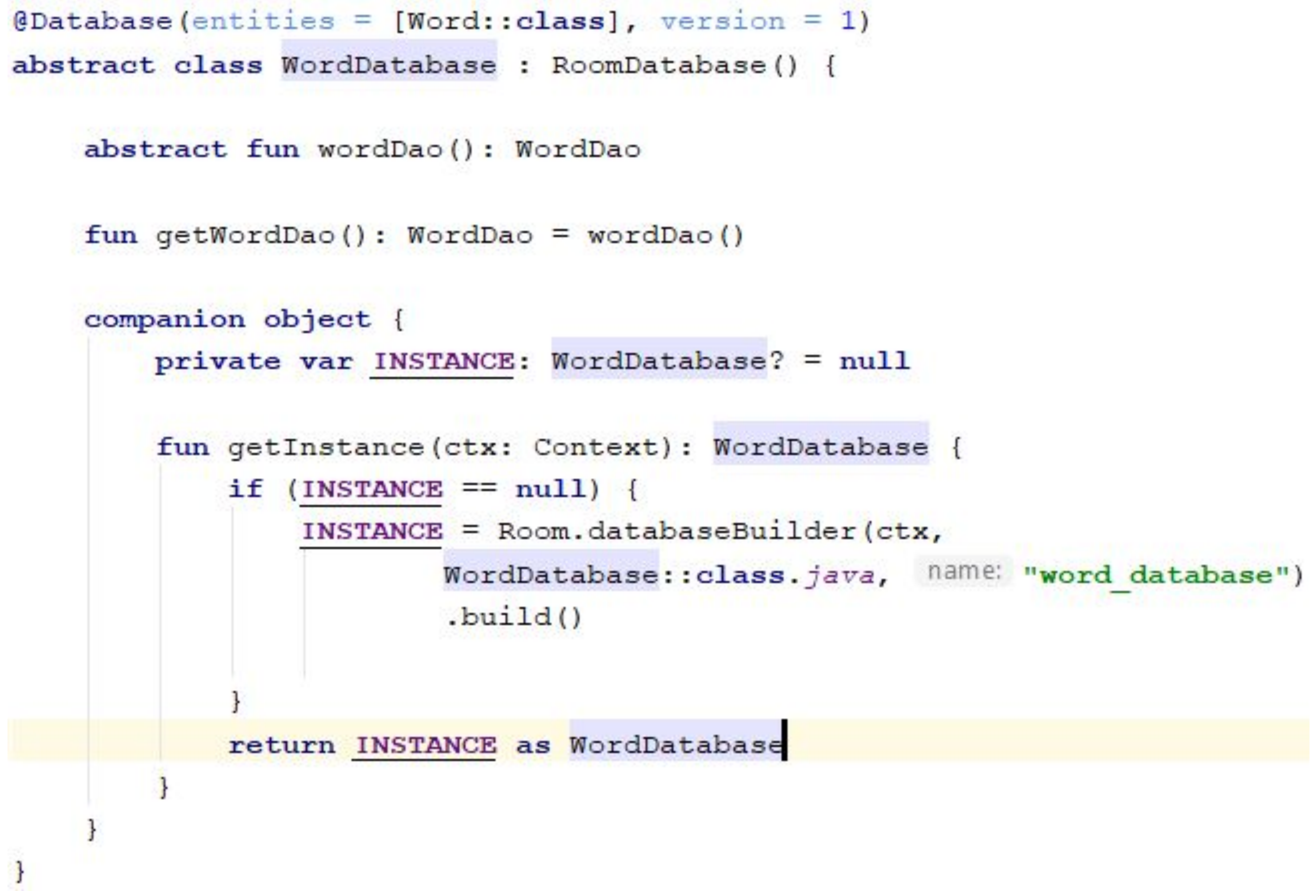
```
@Dao
interface WordDao {
    @Insert
    fun insert(word: Word)

    @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)
}
```

in dem entities gibt man die Entitäten an welche in der Datenbank gespeichert werden

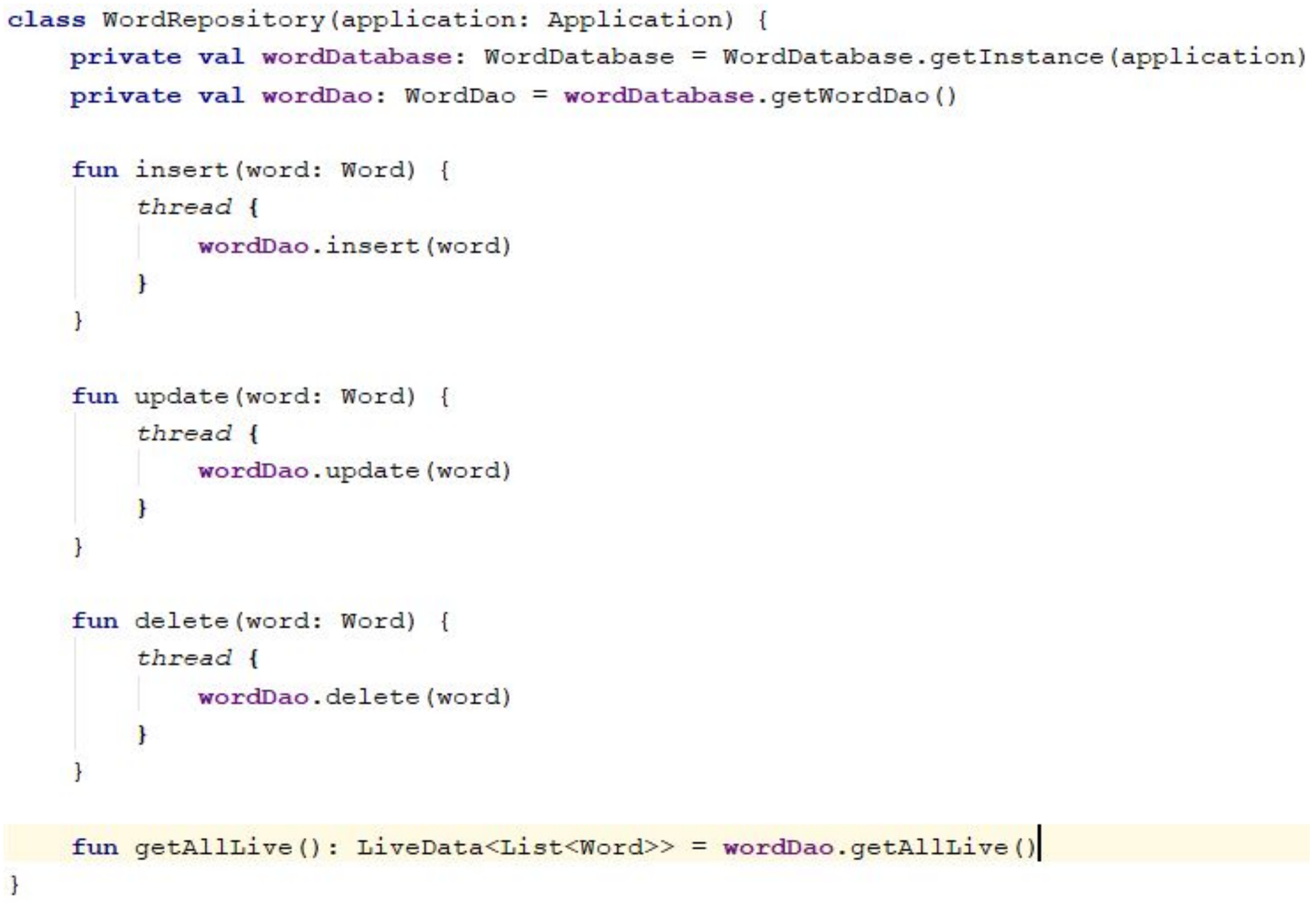
die Version gibt an in welcher Version sich die Datenbank befindet und wenn man die Struktur ändert kann man eine Migrationsfunktion programmieren dass die aktuellen Apps auch auf die aktuelle Struktur geändert werden



```
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() {

}
```

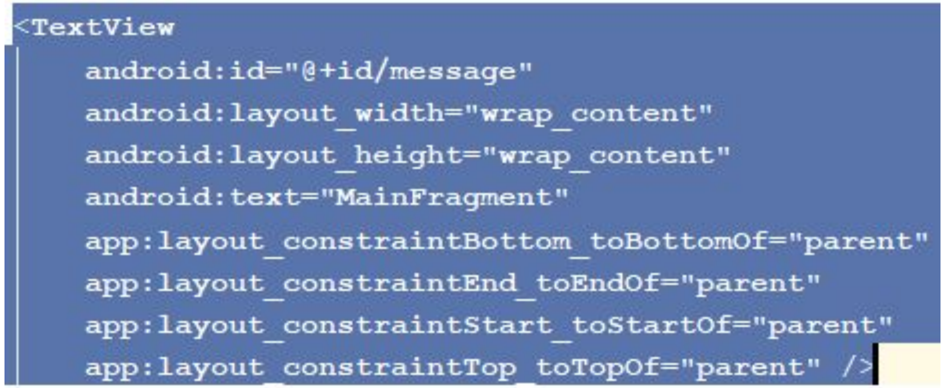


```
@Database(entities = [Word::class], version = 1)
abstract class WordDatabase : RoomDatabase() {

    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
        }
    }
}
```



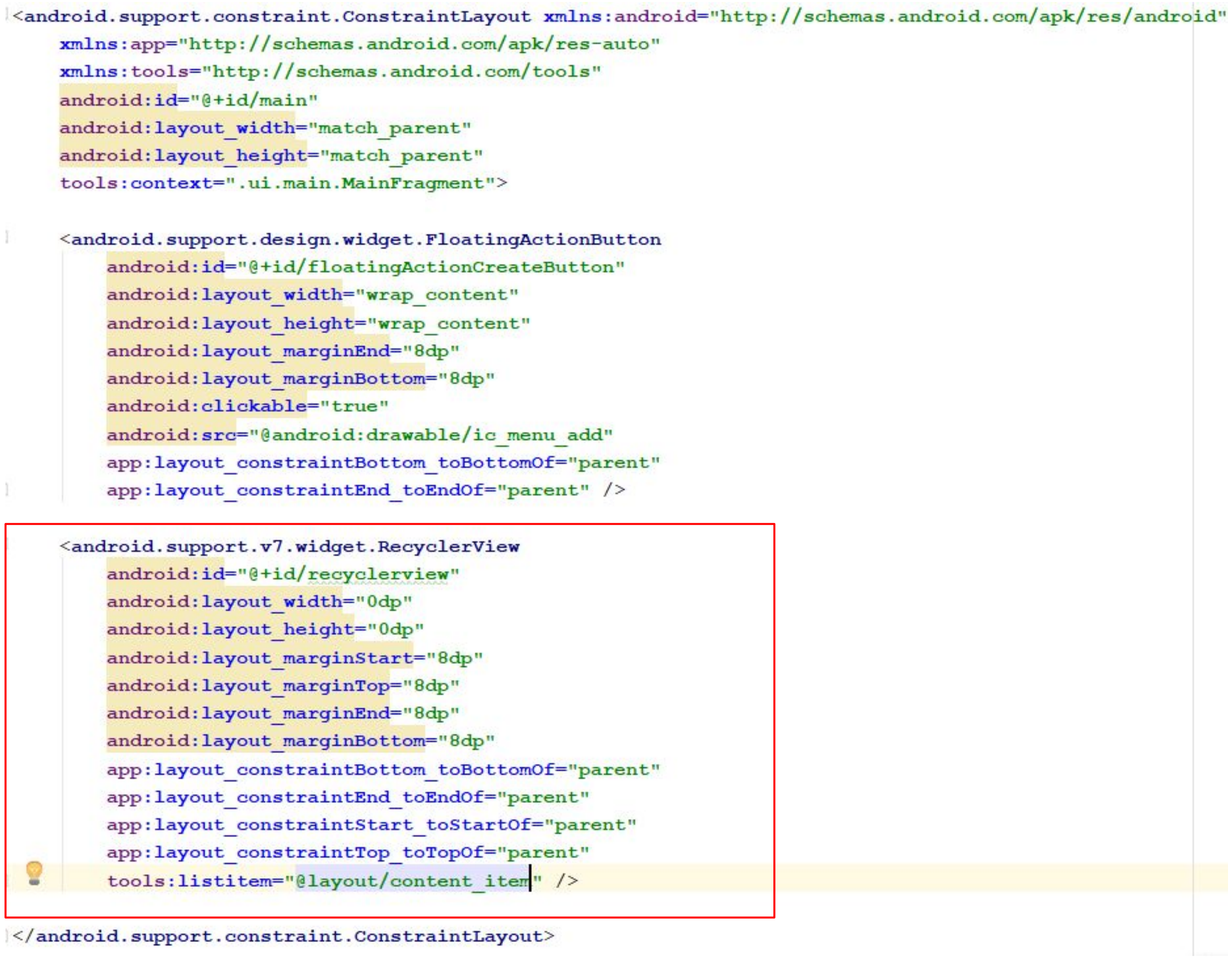
```
class WordRepository(application: Application) {
    private val wordDatabase: WordDatabase = WordDatabase.getInstance(application)
    private val wordDao: WordDao = wordDatabase.getWordDao()

    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()
}
```

Im main-fragment löscht man zuerst die Textview



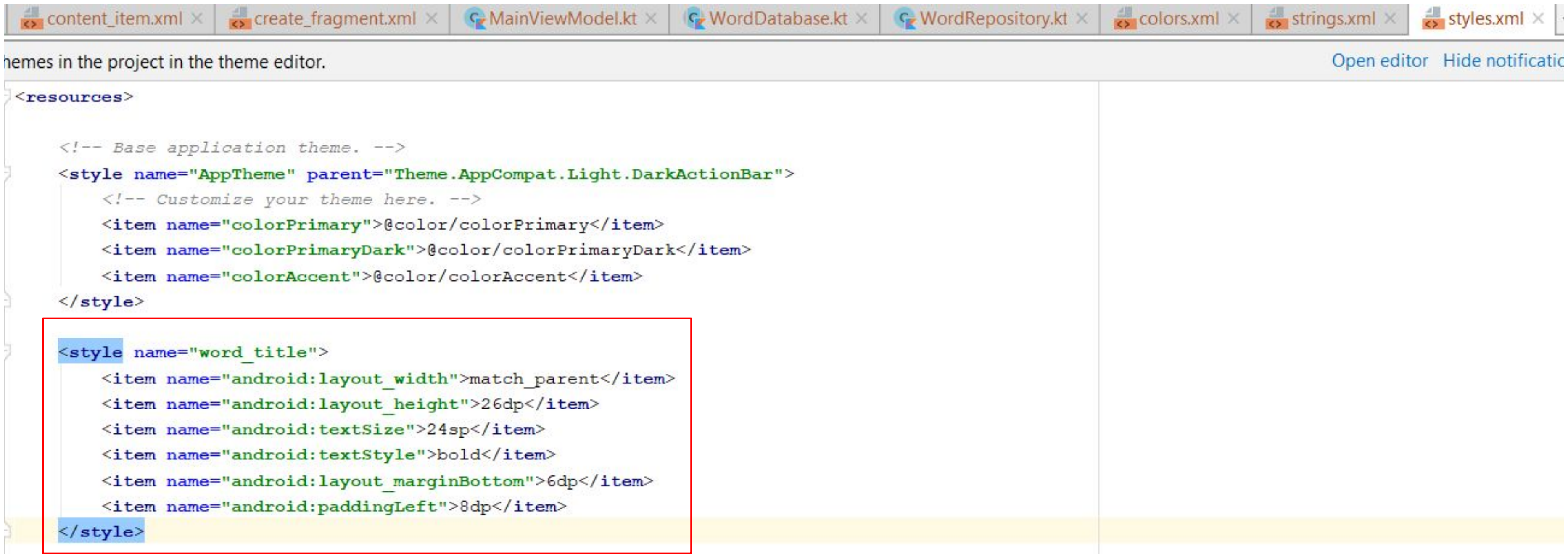
```xml
<TextView
    android:id="@+id/message"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="MainFragment"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"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

im main-fragment



```
<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:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".ui.main.MainFragment">

    <android.support.design.widget.FloatingActionButton
        android:id="@+id/floatingActionCreateButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginEnd="8dp"
        android:layout_marginBottom="8dp"
        android:clickable="true"
        android:src="@android:drawable/ic_menu_add"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent" />

    <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"
        tools:listitem="@layout/content_item" />

</android.support.constraint.ConstraintLayout>
```

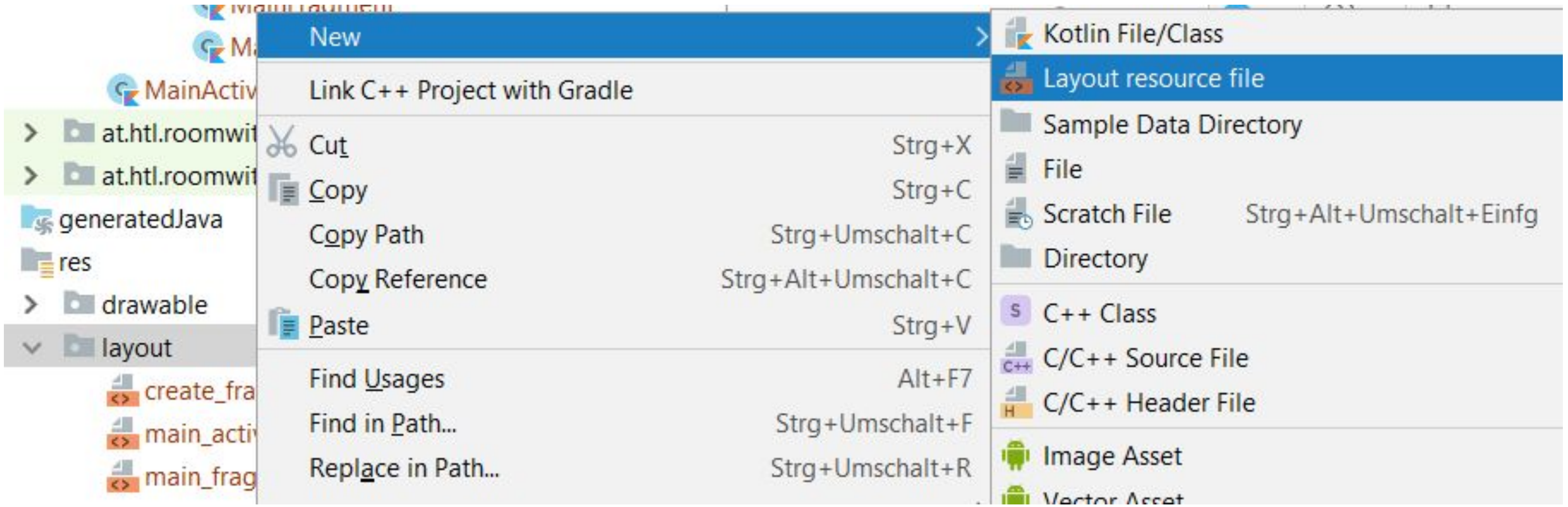
im styles xml fügen wir einen neuen style
den kann man ähnlich wie eine css-klasse verwenden

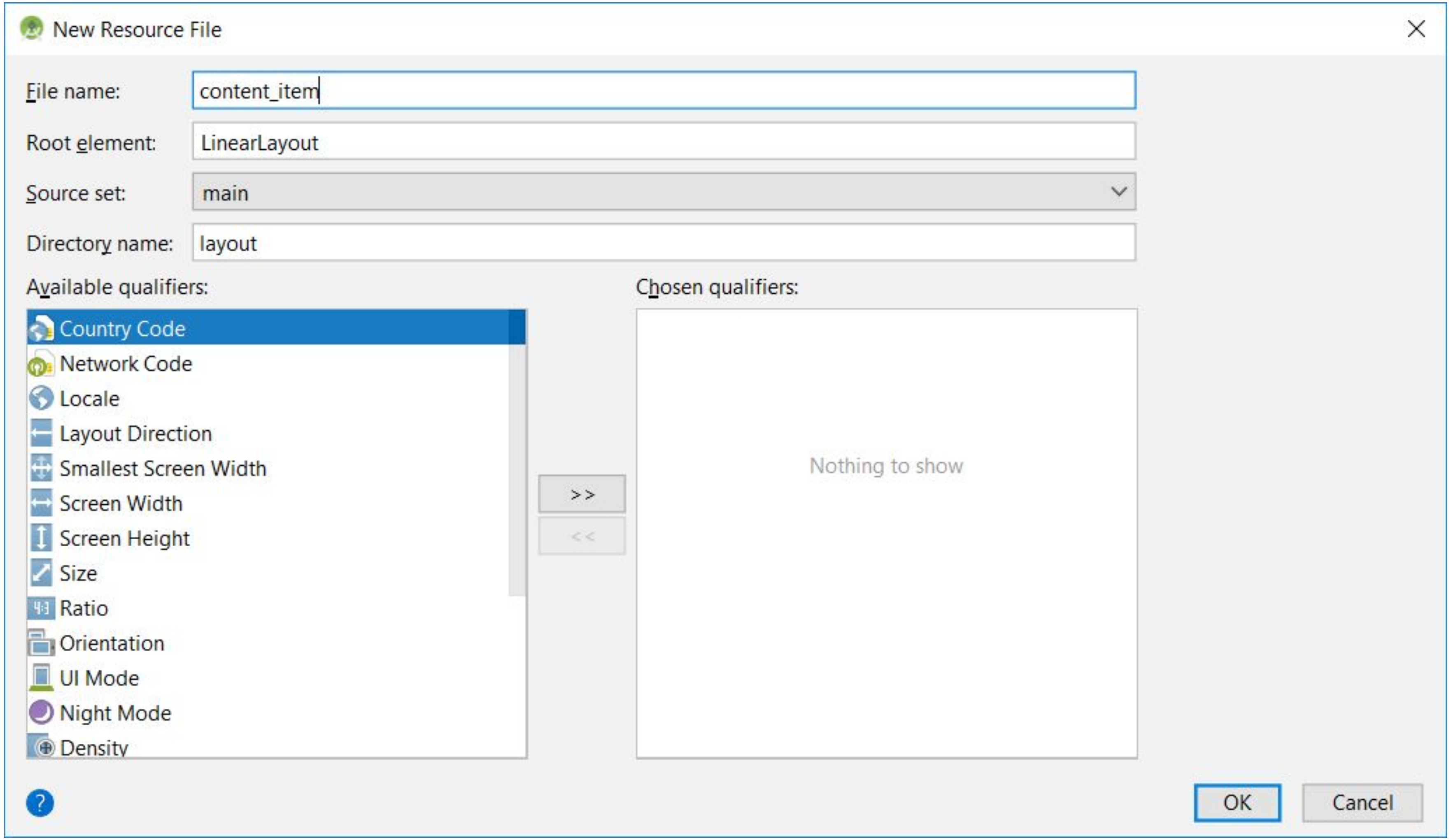


hemes in the project in the theme editor.

```
<resources>

    <!-- Base application theme. -->
    <style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">
        <!-- Customize your theme here. -->
        <item name="colorPrimary">@color/colorPrimary</item>
        <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
        <item name="colorAccent">@color/colorAccent</item>
    </style>

    <style name="word_title">
        <item name="android:layout_width">match_parent</item>
        <item name="android:layout_height">26dp</item>
        <item name="android:textSize">24sp</item>
        <item name="android:textStyle">bold</item>
        <item name="android:layout_marginBottom">6dp</item>
        <item name="android:paddingLeft">8dp</item>
    </style>
```

layout > new > layout resource file



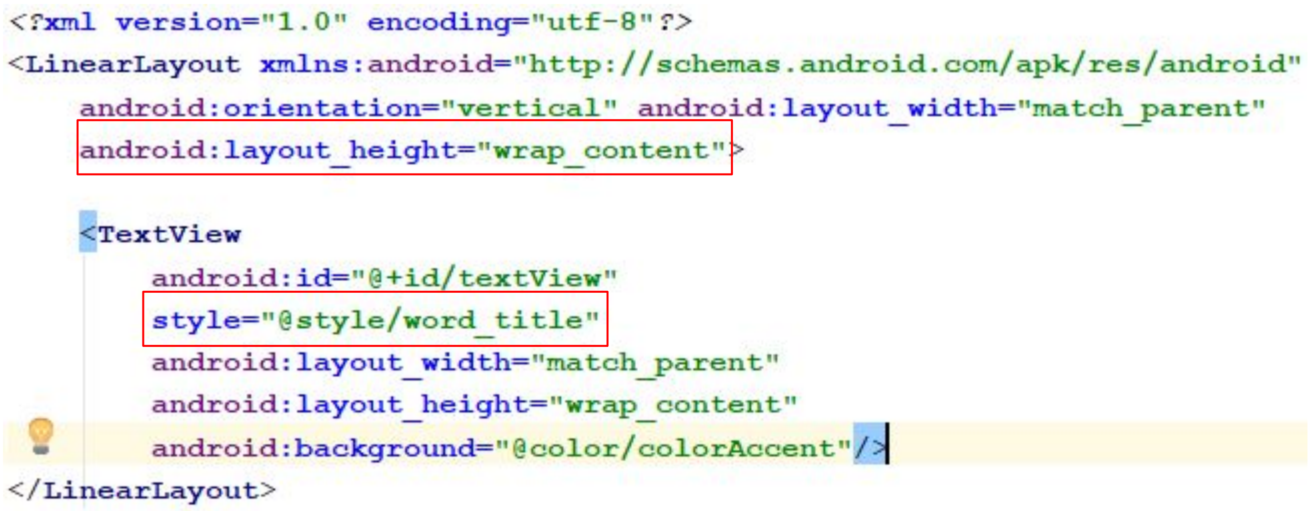

New Resource File

File name: content_item

Root element: LinearLayout

Source set: main

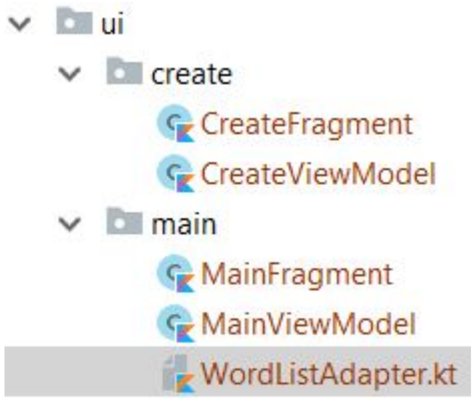
Directory name: layout

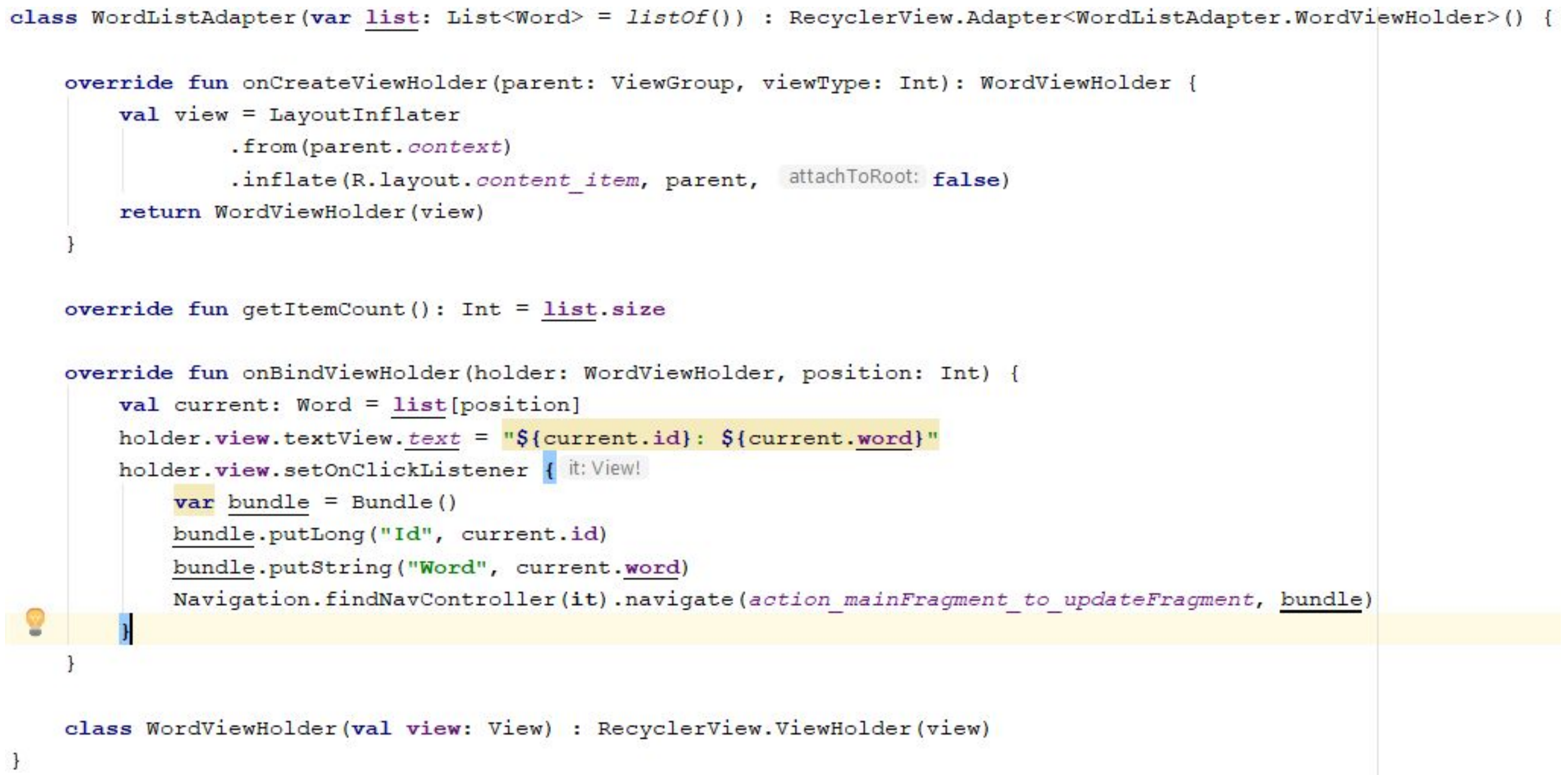


```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="match_parent"
    android:layout_height="wrap_content">

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

Hier setzt man dann den style

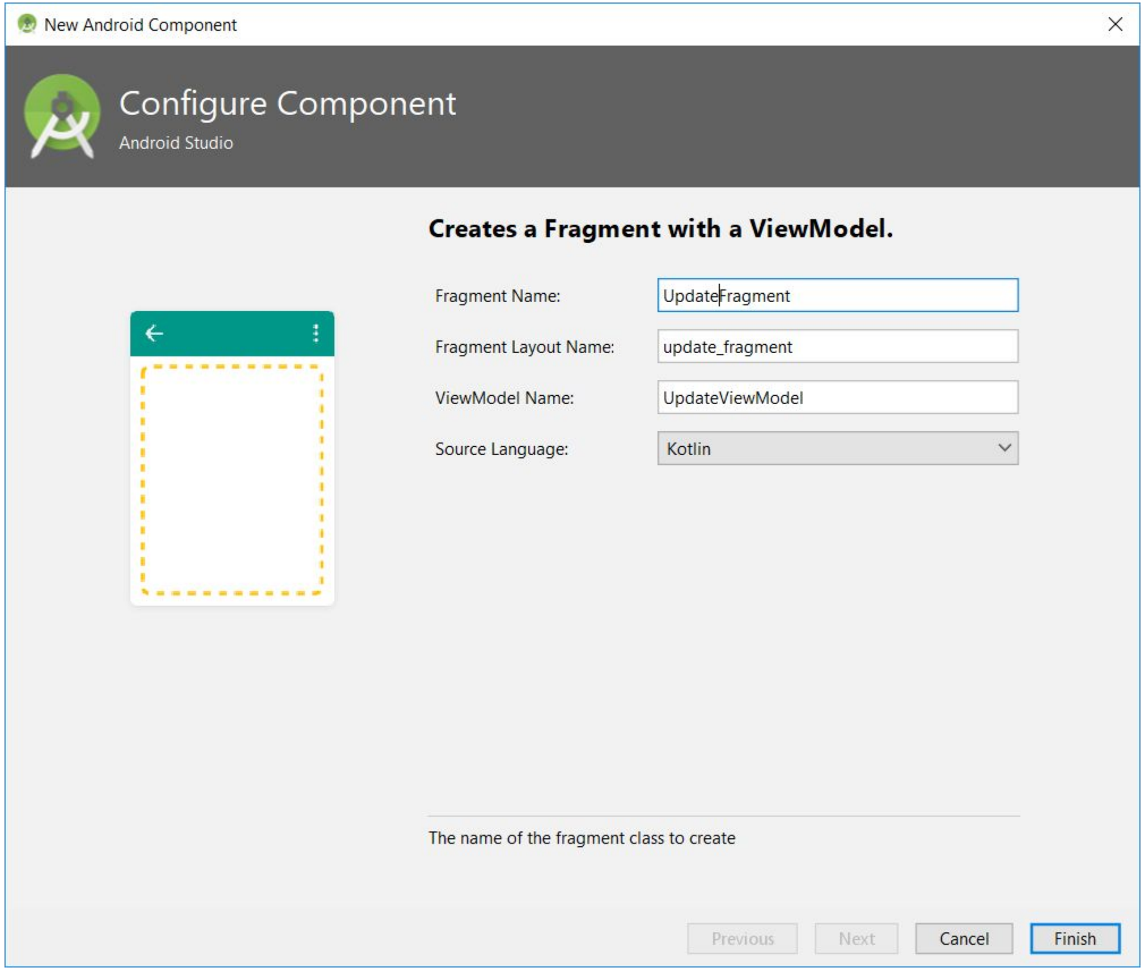
Jetzt müssen wir einen Adpater für die RecyclerView erstellen





```kotlin
class WordListAdapter(var list: List<Word> = listOf()) : RecyclerView.Adapter<WordListAdapter.WordViewHolder>() {

    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): WordViewHolder {
        val view = LayoutInflater
                .from(parent.context)
                .inflate(R.layout.content_item, parent,  attachToRoot: false)
        return WordViewHolder(view)
    }

    override fun getItemCount(): Int = list.size

    override fun onBindViewHolder(holder: WordViewHolder, position: Int) {
        val current: Word = list[position]
        holder.view.textView.text = "${current.id}: ${current.word}"
        holder.view.setOnClickListener { it: View!
            var bundle = Bundle()
            bundle.putLong("Id", current.id)
            bundle.putString("Word", current.word)
            Navigation.findNavController(it).navigate(action_mainFragment_to_updateFragment, bundle)
        }
    }

    class WordViewHolder(val view: View) : RecyclerView.ViewHolder(view)
}
```

layout > new > fragment > fragment (with viewmodel)



Creates a Fragment with a ViewModel.

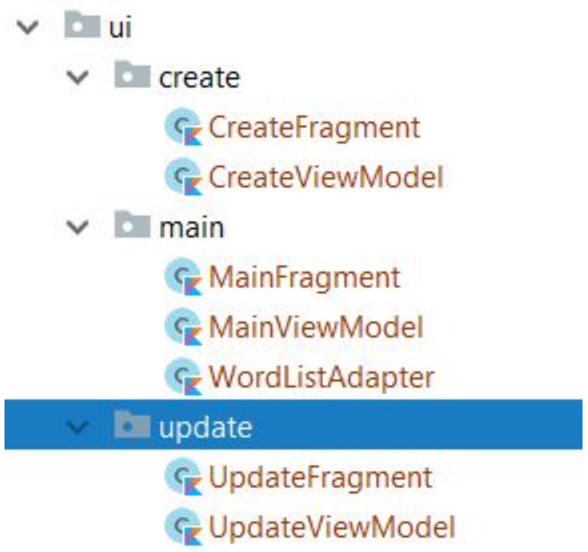
Fragment Name: UpdateFragment

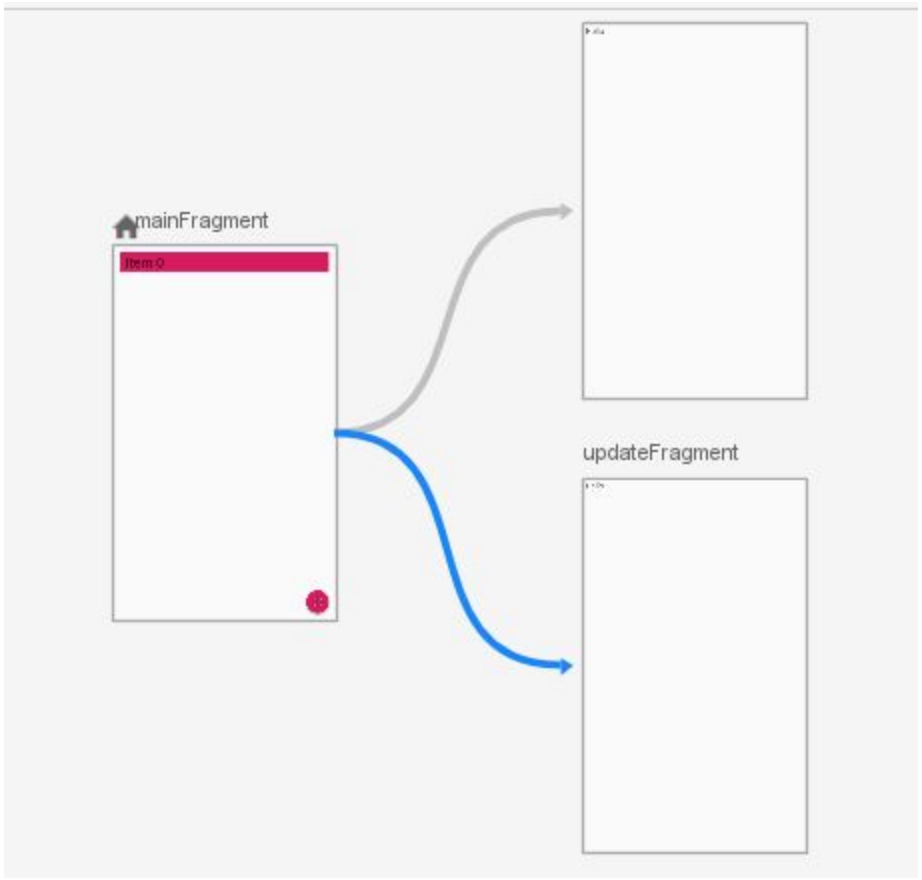
Fragment Layout Name: update_fragment

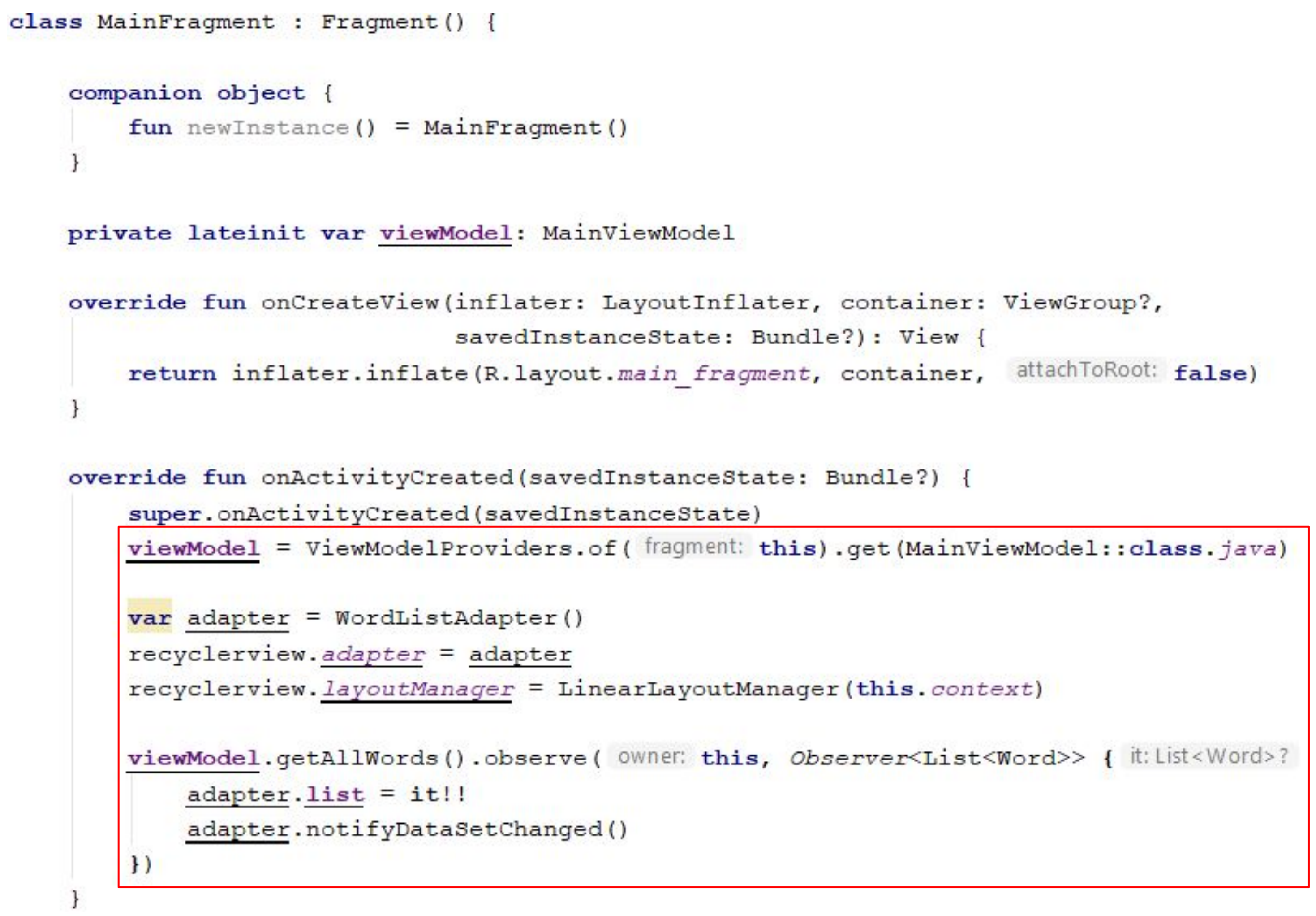
ViewModel Name: UpdateViewModel

Source Language: Kotlin

The name of the fragment class to create

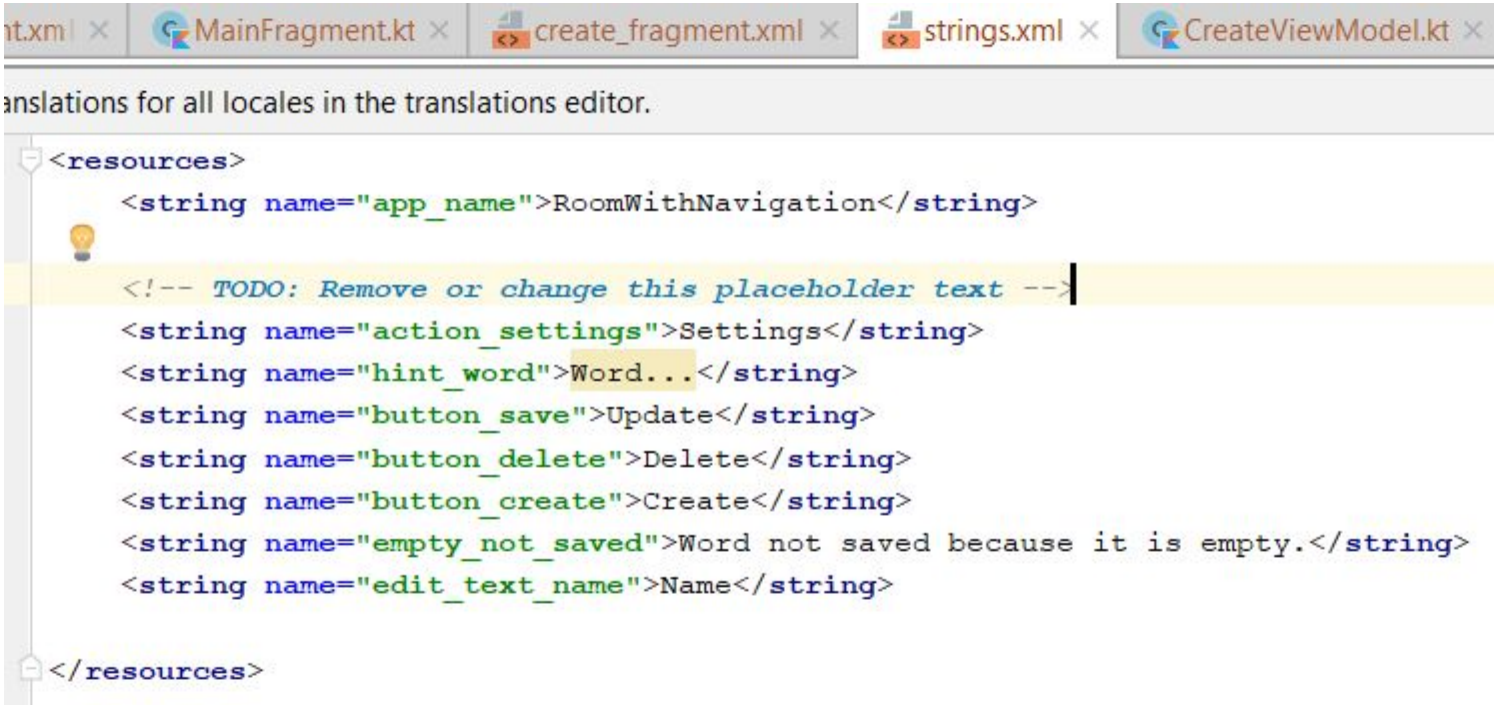

ui
create
CreateFragment
CreateViewModel
main
MainFragment
MainViewModel
WordListAdapter
update
UpdateFragment
UpdateViewModel


mainFragment
Item 0
updateFragment



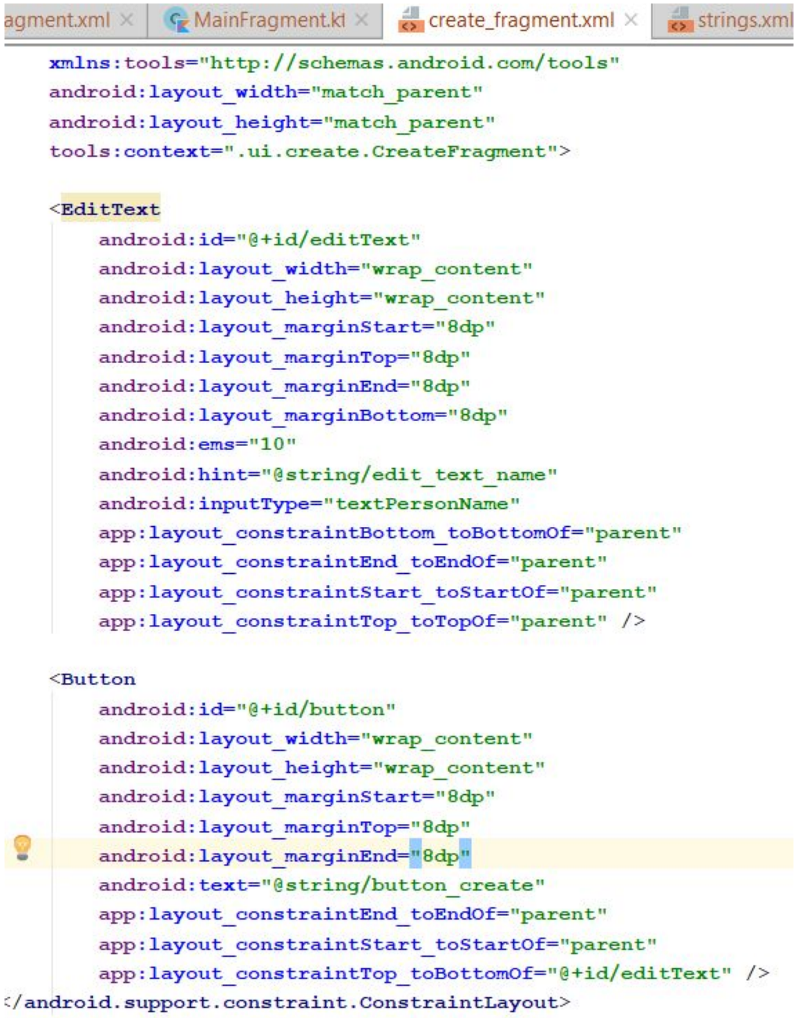
```
class MainFragment : Fragment() {

    companion object {
        fun newInstance() = MainFragment()
    }

    private lateinit var viewModel: MainViewModel

    override fun onCreateView(inflater: LayoutInflater, container: ViewGroup?,
                              savedInstanceState: Bundle?): View {
        return inflater.inflate(R.layout.main_fragment, container,  attachToRoot: false)
    }

    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!!
            adapter.notifyDataSetChanged()
        })
    }
```

in das string xml file fügen wir die notwendigen strings ein

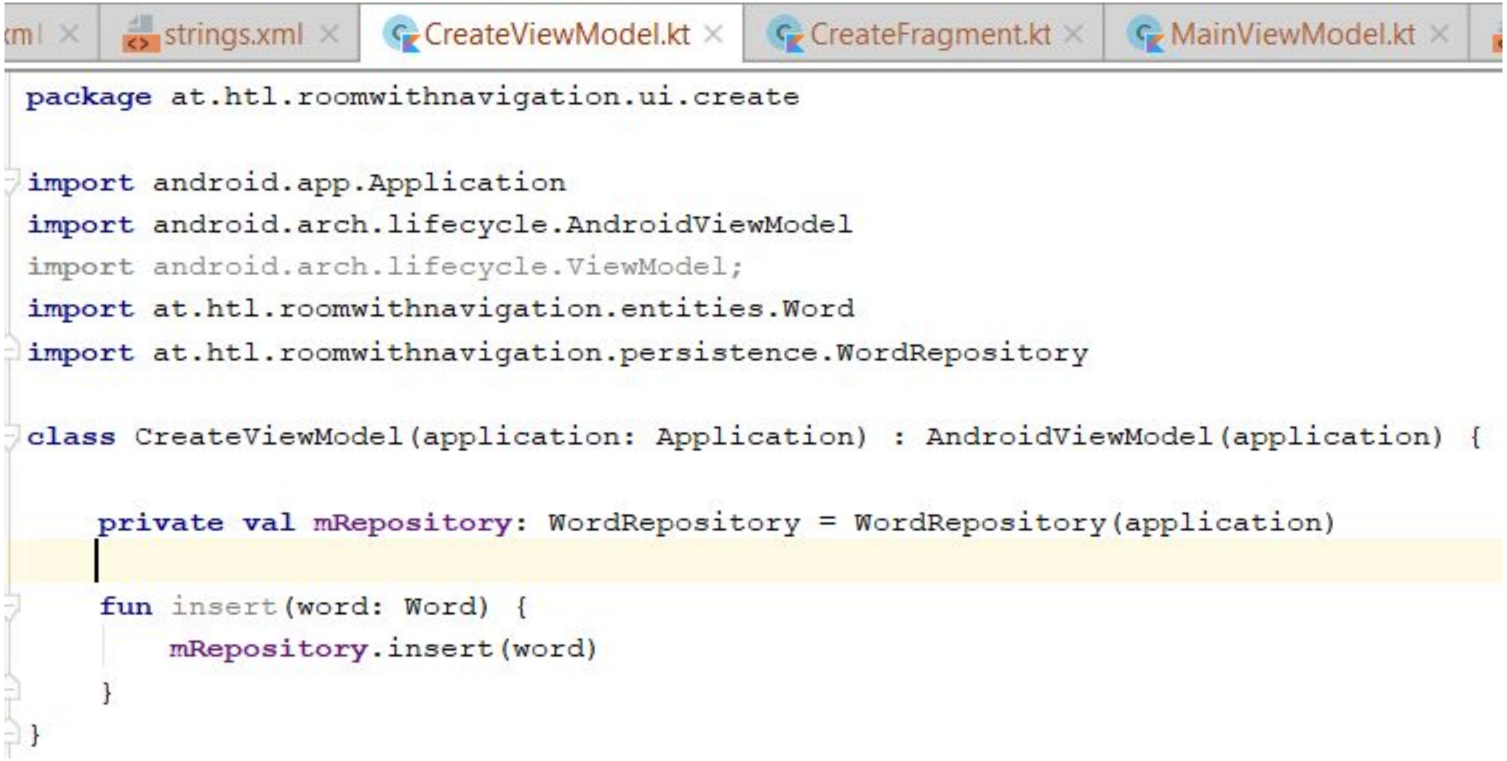


anslations for all locales in the translations editor.

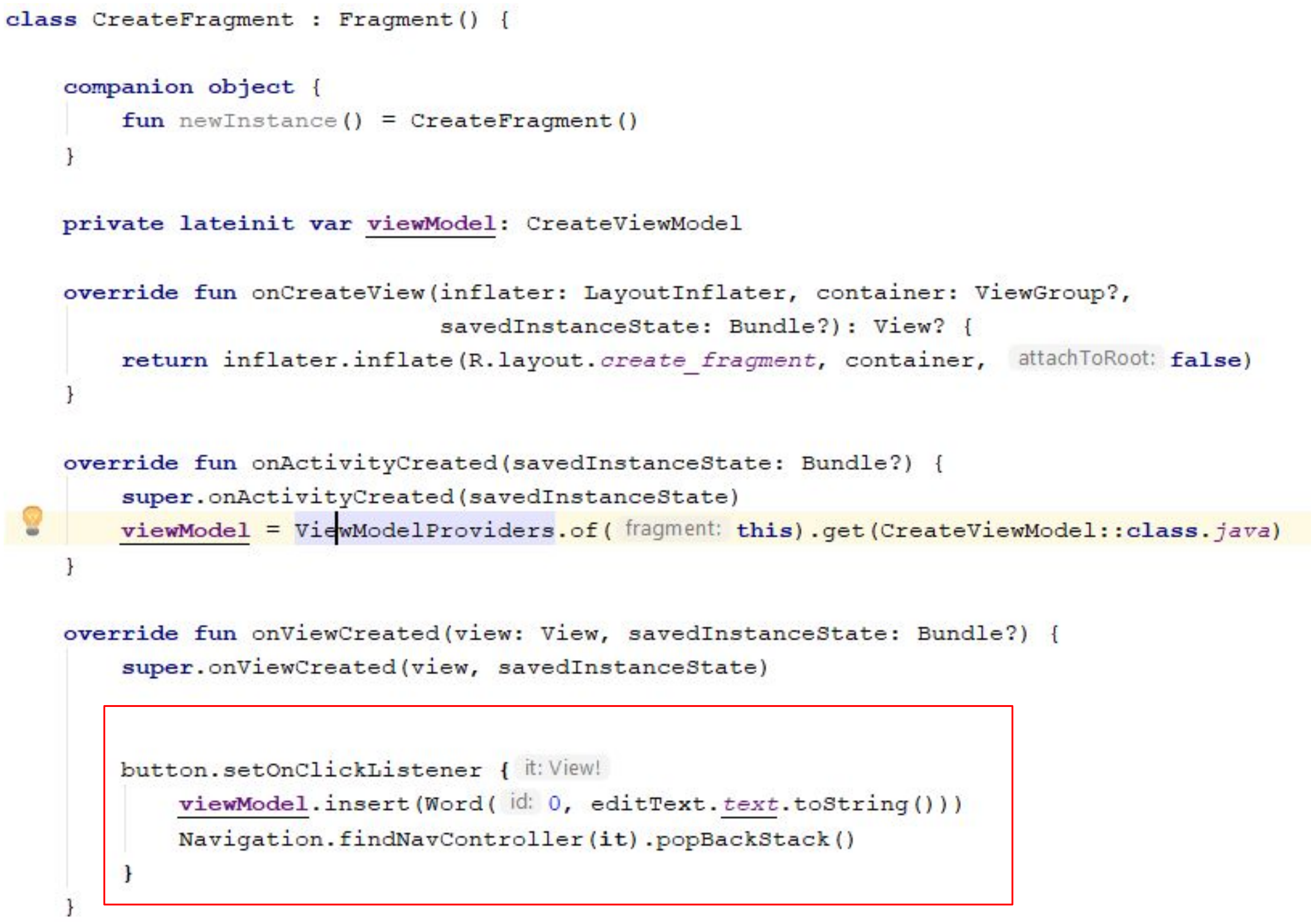
```
<resources>
    <string name="app_name">RoomWithNavigation</string>

    <!-- 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>
```



```
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>
```



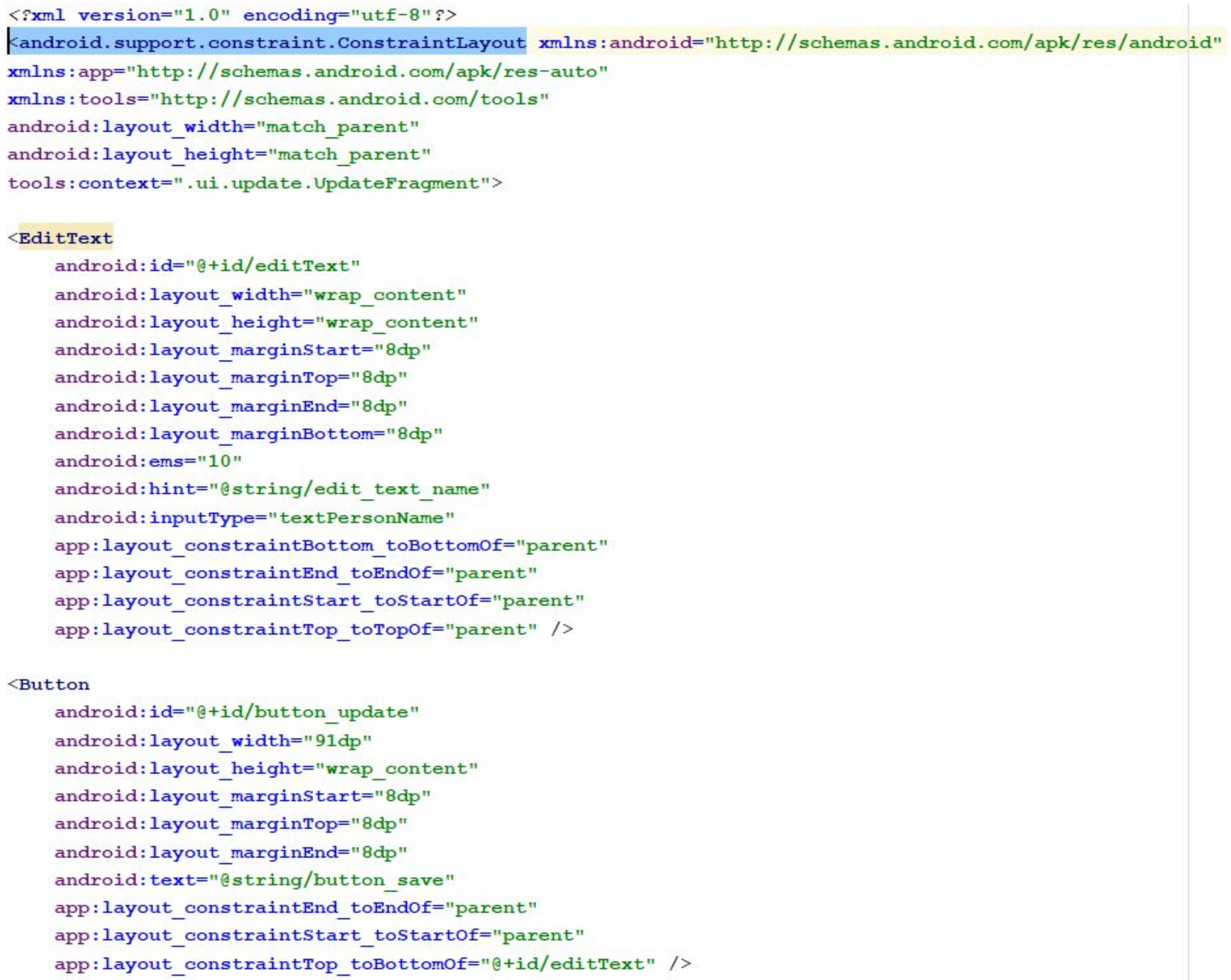
```kotlin
package at.htl.roomwithnavigation.ui.create

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)
    }
}
```



```
class CreateFragment : Fragment() {

    companion object {
        fun newInstance() = CreateFragment()
    }

    private lateinit var viewModel: CreateViewModel

    override fun onCreateView(inflater: LayoutInflater, container: ViewGroup?,
                              savedInstanceState: Bundle?): View? {
        return inflater.inflate(R.layout.create_fragment, container,  attachToRoot: false)
    }

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

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


        button.setOnClickListener {  it: View!
            viewModel.insert(Word( id: 0, editText.text.toString()))
            Navigation.findNavController(it).popBackStack()
        }
    }
```

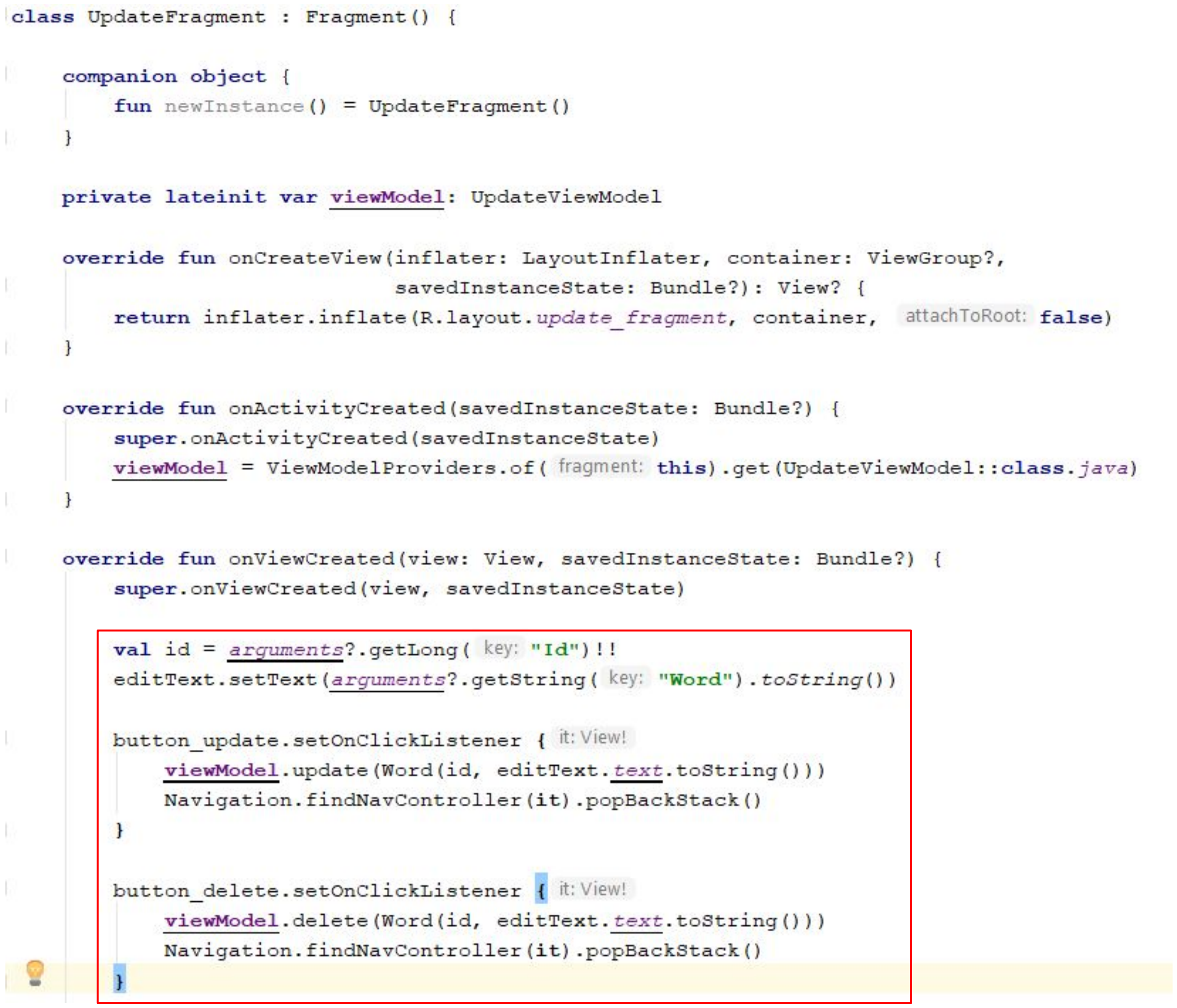
```kotlin
class UpdateViewModel(application: Application) : AndroidViewModel(application) {

    private var mRepository = WordRepository(application)

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

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

im update_fragment



```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" />
```

```xml
<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>
```



```kotlin
class UpdateFragment : Fragment() {

    companion object {
        fun newInstance() = UpdateFragment()
    }

    private lateinit var viewModel: UpdateViewModel

    override fun onCreateView(inflater: LayoutInflater, container: ViewGroup?,
                              savedInstanceState: Bundle?): View? {
        return inflater.inflate(R.layout.update_fragment, container,  attachToRoot: false)
    }

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

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

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

        button_update.setOnClickListener {  it: View!
            viewModel.update(Word(id, editText.text.toString()))
            Navigation.findNavController(it).popBackStack()
        }

        button_delete.setOnClickListener {  it: View!
            viewModel.delete(Word(id, editText.text.toString()))
            Navigation.findNavController(it).popBackStack()
        }
```

Quellen

https://www.youtube.com/watch?v=5qlIPTDE274

https://codelabs.developers.google.com/codelabs/android-room-with-a-view/#0

Github:

https://github.com/ThomasKaar/RoomWithNavigation