Министерство образования и молодежной политики Свердловской области



ГАПОУ СО «Екатеринбургский колледж транспортного строительства»

Отчёт по программе «Неявные интенты»

Выполнил: Максимов К.Э.

Группа: ПР-31

Преподаватель: Мирошниченко Г.В

2023

**Входные данные:**

title: String (название преступления)

date: Date (дата преступления)

isSolved: Boolean (закончено или нет преступление)

suspect: String (подозреваемый)

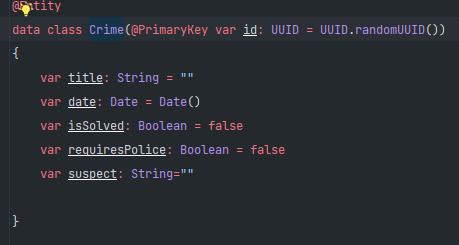
requiresPolice: Int

**Выходные данные:**

**Toast** (сообщения пользователю)

**Листинг программы:**

Crime.kt



CrimeDatailModel.kt



CrimeFragment.kt

package com.criminal\_intent.android.pract\_19\_1  
  
  
import android.app.Activity  
import android.content.Intent  
import android.content.pm.PackageManager  
import android.content.pm.ResolveInfo  
import android.net.Uri  
import android.os.Build  
import android.os.Bundle  
import android.provider.ContactsContract  
  
import android.text.*Editable*import android.text.*TextWatcher*import android.text.format.DateFormat  
import android.view.LayoutInflater  
import android.view.View  
import android.view.ViewGroup  
import android.widget.Button  
import android.widget.CheckBox  
import android.widget.EditText  
import android.widget.Toast  
import androidx.annotation.RequiresApi  
import androidx.fragment.app.Fragment  
import androidx.lifecycle.ViewModelProviders  
import kotlinx.coroutines.CoroutineScope  
import kotlinx.coroutines.Dispatchers  
import kotlinx.coroutines.GlobalScope  
import kotlinx.coroutines.launch  
import java.util.\*  
  
  
private const val *DATE\_FORMAT* = "EEE, MMM, dd"  
private const val *REQUEST\_CONTACT*=1  
class CrimeFragment : Fragment() {  
  
 private lateinit var crime: Crime  
 private lateinit var title: EditText  
 private lateinit var dateButton: Button  
 private lateinit var solvedCheckBox: CheckBox  
 private lateinit var reportButton: Button  
 private lateinit var suspectButton: Button  
 private lateinit var listButton: Button  
 private lateinit var addButoon: Button  
 private val crimeDetailViewModel:CrimeDatailModel by *lazy*{  
 ViewModelProviders.of(this).get(CrimeDatailModel::class.*java*)  
 }  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 crime = Crime()  
 }  
  
 override fun onCreateView(inflater: LayoutInflater, container: ViewGroup?,  
 savedInstanceState: Bundle?): View? {  
 val view = inflater.inflate(R.layout.*fragment\_crime*, container, false)  
 title = view.findViewById(R.id.*crime\_title*) as EditText  
 dateButton = view.findViewById(R.id.*crime\_date*) as Button  
 solvedCheckBox = view.findViewById(R.id.*crime\_solved*) as CheckBox  
 reportButton=view.findViewById(R.id.*crime\_report*) as Button  
 suspectButton=view.findViewById(R.id.*crime\_suspect*) as Button  
// listButton = view.findViewById(R.id.listButton) as Button  
 addButoon = view.findViewById(R.id.*addButton*) as Button  
  
 dateButton.*apply* { *text* = crime.date.toString(); *isEnabled* = false }  
 return view  
 }  
  
  
 override fun onStart() {  
 super.onStart()  
 val titleWatcher = object : *TextWatcher* {  
 override fun beforeTextChanged(sequence: *CharSequence*?, start: Int, count: Int,  
 after: Int) {}  
  
 override fun onTextChanged(sequence: *CharSequence*?, start: Int, before: Int,  
 count: Int) {  
 crime.title = sequence.*toString*()  
 }  
  
 override fun afterTextChanged(sequence: *Editable*?) {}  
 }  
 solvedCheckBox.setOnClickListener(){  
 suspectButton.*isEnabled* = solvedCheckBox.*isChecked* reportButton.*isEnabled* = solvedCheckBox.*isChecked* }  
 title.addTextChangedListener(titleWatcher)  
 solvedCheckBox.*apply* {  
 setOnCheckedChangeListener { \_, isChecked ->  
 crime.isSolved = isChecked  
 }  
 }  
 reportButton.setOnClickListener {  
 Intent(Intent.*ACTION\_SEND*).*apply*{  
 *type*="text/plain"  
 putExtra(Intent.*EXTRA\_TEXT*,getCrimeReport())  
 putExtra(Intent.*EXTRA\_SUBJECT*,getString(R.string.*crime\_report\_subject*))  
 }.*also*{  
 intent ->  
 val chooserIntent= Intent.createChooser(intent,getString(R.string.*send\_report*))  
 startActivity(chooserIntent)  
 }  
  
 }  
 suspectButton.*apply*{  
 val pickContactIntent=Intent(Intent.*ACTION\_PICK*, ContactsContract.Contacts.*CONTENT\_URI*)  
 setOnClickListener {  
 startActivityForResult(pickContactIntent, *REQUEST\_CONTACT*)  
 }  
 val packageManager: PackageManager =requireActivity().*packageManager* val resolvedActivity: ResolveInfo?=packageManager.resolveActivity(pickContactIntent, PackageManager.*MATCH\_DEFAULT\_ONLY*)  
 if (resolvedActivity==null){  
 *isEnabled*=false  
 }  
 }  
 addButoon.setOnClickListener(){  
 var existingCrime:Crime?  
  
 if (title.*text*.*isNullOrEmpty*()) {  
 Toast.makeText(requireActivity(), "Введите преступление", Toast.*LENGTH\_SHORT*).show()  
 } else {  
 val crimeTitle = title.*text*.toString()  
  
  
 // Проверка наличия преступления с таким же заголовком в базе данных  
 GlobalScope.*launch*(Dispatchers.IO) {  
 existingCrime = crimeDetailViewModel.getCrimeByTitle(title.*text*.toString())  
 *CoroutineScope*(Dispatchers.Main).*launch* {  
  
  
 if (existingCrime == null) {  
 // Преступления с таким заголовком нет, добавляем новое  
 val crime = Crime()  
 crime.title = crimeTitle  
 crime.date = Date()  
 crime.isSolved = solvedCheckBox.*isChecked* crimeDetailViewModel.add(crime)  
  
 title.*text* = null  
 Toast.makeText(  
 requireActivity(),  
 "Преступление добавлено",  
 Toast.*LENGTH\_SHORT* ).show()  
 } else {  
 Toast.makeText(  
 requireActivity(),  
 "Преступление с таким заголовком уже существует",  
 Toast.*LENGTH\_SHORT* ).show()  
 }  
 }  
 }  
 }  
 }  
 }  
 private fun updateUI(){  
 title.setText(crime.title)  
 solvedCheckBox.*apply* {  
 *isChecked*=crime.isSolved!!  
 jumpDrawablesToCurrentState()  
 }  
 if (crime.suspect.*isNotEmpty*()){  
 suspectButton.*text*=crime.suspect  
 }  
 }  
 private fun getCrimeReport(): String{  
 val solvedString = if (crime.isSolved == true){  
 getString(R.string.*crime\_report\_solved*)  
 }  
 else{  
 getString(R.string.*crime\_report\_unsolved*)  
 }  
 val dateString= DateFormat.format(*DATE\_FORMAT*,crime.date).toString()  
 var suspect=if (crime.suspect.*isBlank*()){  
 getString(R.string.*crime\_report\_no\_suspect*)  
 }  
 else{  
 getString(R.string.*crime\_report\_suspect*, crime.suspect)  
 }  
 return getString(R.string.*crime\_report*,crime.title,dateString,solvedString,suspect)  
 }  
 @RequiresApi(Build.VERSION\_CODES.*O*)  
 override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {  
 when{  
 resultCode!= Activity.*RESULT\_OK* -> return  
 requestCode== *REQUEST\_CONTACT* && data !=null ->{  
 val contactUri: Uri? =data.*data* val queryFields=*arrayOf*(ContactsContract.Contacts.*DISPLAY\_NAME*)  
 val cursor =  
 contactUri?.*let* {  
 requireActivity().*contentResolver*.query(**it**,queryFields,null,  
 null)  
 }  
 cursor?.*use* {  
 if (**it**.*count*==0){  
 return  
 }  
 **it**.moveToFirst()  
 val suspect=**it**.getString(0)  
 crime.suspect=suspect  
 crimeDetailViewModel.save(crime)  
 suspectButton.*text*=suspect  
 }  
  
 }  
 }  
 }  
 companion object{  
 fun newInstance() : CrimeListFragment{  
 return CrimeListFragment()  
 }  
 }  
}

CrimeListFragment.kt

package com.criminal\_intent.android.pract\_19\_1  
  
import android.os.Bundle  
import android.util.Log  
import android.view.LayoutInflater  
import android.view.View  
import android.view.ViewGroup  
import android.widget.TextView  
import androidx.fragment.app.Fragment  
import androidx.lifecycle.ViewModelProviders  
  
import androidx.recyclerview.widget.LinearLayoutManager  
import androidx.recyclerview.widget.RecyclerView  
  
private const val *TAG* = "CrimeListFragment"  
class CrimeListFragment : Fragment() {  
  
 private val common = 0  
 private val serious = 3  
 private lateinit var recyclerView: RecyclerView  
 private var adapter: CrimeAdapter? = null  
 private val crimeListViewModel: CrimeListViewModel by *lazy* {  
 ViewModelProviders.of(this).get(CrimeListViewModel::class.*java*)  
 }  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 setHasOptionsMenu(true)  
 }  
  
 override fun onCreateView(  
 inflater: LayoutInflater,  
 container: ViewGroup?,  
 savedInstanceState: Bundle?): View? {  
 val view = inflater.inflate(R.layout.*fragment\_crime\_list*, container, false)  
 recyclerView = view.findViewById(R.id.*crime\_recycler\_view*) as RecyclerView  
 recyclerView.*layoutManager* = LinearLayoutManager(*context*)  
 recyclerView.*adapter*=adapter  
 return view  
 }  
 override fun onViewCreated(view: View, savedInstanceState: Bundle?) {  
 super.onViewCreated(view, savedInstanceState)  
 crimeListViewModel.crimeListLiveData?.observe(  
 *viewLifecycleOwner*,  
 androidx.lifecycle.*Observer* {  
 crimes->crimes.*let* { Log.i(*TAG*,"Количество: ${crimes.size}")  
 updateUI(crimes) }  
 }  
 )  
 }  
 private fun updateUI(crimes:*List*<Crime>){  
 adapter=CrimeAdapter(crimes)  
 recyclerView.*adapter*=adapter  
 }  
  
 companion object {  
 fun newInstance(): CrimeListFragment {  
 return CrimeListFragment()  
 }  
 }  
 */\*private inner class CommonCrimeHolder(view: View): RecyclerView.ViewHolder(view), View.OnClickListener {  
 private lateinit var crime: Crime  
  
 private val titleTextView: TextView = itemView.findViewById(R.id.crime\_title)  
 private val dateTextView: TextView = itemView.findViewById(R.id.crime\_date)  
 init {  
 itemView.setOnClickListener(this)  
 }  
  
 fun bind(crime: Crime) {  
 this.crime = crime  
 titleTextView.text = this.crime.title  
 dateTextView.text = this.crime.date.toString()  
 }  
 override fun onClick(v: View) {  
 Toast.makeText(context, "${crime.title} pressed!", Toast.LENGTH\_SHORT)  
 .show()  
 }  
 }  
 private inner class SeriousCrimeHolder(view: View) : RecyclerView.ViewHolder(view),  
 View.OnClickListener {  
  
 private lateinit var crime: Crime  
  
 private val callPolice: Button = itemView.findViewById(R.id.call\_police)  
 private val titleTextView: TextView = itemView.findViewById(R.id.crime\_title)  
 private val dateTextView: TextView = itemView.findViewById(R.id.crime\_date)  
  
 init {  
 itemView.setOnClickListener(this)  
 callPolice.setOnClickListener {  
 val builder = AlertDialog.Builder(context)  
 builder.setTitle("Warning")  
 builder.setMessage("Are you sure you want to contact the police?")  
 builder.setPositiveButton("Yes")  
 { \_, \_ ->  
 Toast.makeText(context, "You called the police", Toast.LENGTH\_SHORT).show()  
 }  
 builder.setNegativeButton("Cancel", null)  
 builder.create()  
 builder.show()  
 }  
 }  
  
 fun bind(crime: Crime) {  
 this.crime = crime  
 titleTextView.text = this.crime.title  
 dateTextView.text = this.crime.date.toString()  
 }  
  
 override fun onClick(v: View) {  
 Toast.makeText(context, "${crime.title} pressed!", Toast.LENGTH\_SHORT).show()  
 }  
 }\*/* private inner class CrimeAdapter(var crimes: *List*<Crime>):RecyclerView.Adapter<CrimeHolder>(){  
 override fun onCreateViewHolder(parent: ViewGroup, viewType: Int):CrimeHolder {  
 val view=*layoutInflater*.inflate(R.layout.*list\_item\_common\_crime*,parent,false)  
 return CrimeHolder(view)  
 }  
  
  
 override fun onBindViewHolder(holder: CrimeHolder, position: Int) {  
  
 val crime=crimes[position]  
 holder.bind(crime)  
 }  
  
 override fun getItemCount(): Int {  
 return crimes.size  
 }  
 }  
 private inner class CrimeHolder(view: View):RecyclerView.ViewHolder(view), View.OnClickListener {  
 private lateinit var crime: Crime  
 val titleTextView: TextView =itemView.findViewById(R.id.crime\_title)  
 val dateTextView: TextView = itemView.findViewById(R.id.crime\_date)  
 init {  
 itemView.setOnClickListener(this)  
 }  
 fun bind(crime:Crime){  
 this.crime=crime  
 titleTextView.text=this.crime.title  
 dateTextView.text=this.crime.date.toString()  
 }  
 override fun onClick(v: View) {  
  
 }  
  
  
 }  
  
  
}

CrimeListViewModel.kt

package com.criminal\_intent.android.pract\_19\_1  
  
import androidx.lifecycle.ViewModel  
  
class CrimeListViewModel : ViewModel() {  
 */\*val crimes = mutableListOf<Crime>()  
 init {  
 for (i in 0..100){  
 val crime = Crime()  
 crime.title = "Crime #$i"  
 crime.isSolved = i % 2 == 0  
 crime.requiresPolice = i % 7 == 0  
 crimes.add(crime)  
 }  
  
 }\*/* val crimeRepository=CrimeRepository.get()  
 val crimeListLiveData=crimeRepository.getCrimes()  
}

CrimeRepository.kt

package com.criminal\_intent.android.pract\_19\_1  
  
import android.content.Context  
import androidx.lifecycle.LiveData  
import androidx.room.Room  
import com.criminal\_intent.android.pract\_19\_1.database.CrimeDatabase  
import com.criminal\_intent.android.pract\_19\_1.database.*migration\_1\_2*import kotlinx.coroutines.Dispatchers  
import kotlinx.coroutines.withContext  
import java.util.\*  
import java.util.concurrent.Executors  
  
private const val *DATABASE\_NAME* = "crime-database"  
  
class CrimeRepository private constructor(context: Context) {  
 private val database: CrimeDatabase = Room.databaseBuilder(  
 context.*applicationContext*,  
 CrimeDatabase::class.*java*,  
 *DATABASE\_NAME* ).addMigrations(*migration\_1\_2*).build()  
 private val crimeDao = database.crimeDao()  
 private val executor = Executors.newSingleThreadExecutor()  
 fun getCrimes(): LiveData<*List*<Crime>>? = crimeDao.getCrimes()  
 fun getCrime(id: UUID): LiveData<Crime?>? = crimeDao.getCrime(id)  
 fun updateCrime(crime: Crime) {  
 executor.execute {  
 crimeDao.getCrime(crime.id)  
 }  
 }  
  
 suspend fun getCrimeByTitle(title: String): Crime? {  
 return withContext(Dispatchers.IO) {  
 crimeDao.getCrimeByTitle(title)  
 }  
 }  
  
 fun addCrime(crime: Crime) {  
 executor.execute {  
 crimeDao.addCrime(crime)  
 }  
 }  
  
 companion object {  
 private var INSTANCE: CrimeRepository? = null  
 fun initialize(context: Context) {  
 if (INSTANCE == null) {  
 INSTANCE = CrimeRepository(context)  
 }  
 }  
  
 fun get(): CrimeRepository {  
 return INSTANCE ?: throw IllegalStateException("CrimeRepository must be initialized")  
 }  
 }  
}

CriminalIntentApplication.kt

package com.criminal\_intent.android.pract\_19\_1  
  
import android.app.Application  
  
class CriminalIntentApplication : Application()  
{  
 override fun onCreate() {  
 super.onCreate()  
 CrimeRepository.initialize(this)  
 }  
}

MainActivity.kt

package com.criminal\_intent.android.pract\_19\_1  
  
  
import android.os.Bundle  
import androidx.appcompat.app.AppCompatActivity  
import androidx.fragment.app.Fragment  
import com.google.android.material.bottomnavigation.BottomNavigationView  
  
  
class MainActivity : AppCompatActivity() {  
 private lateinit var nav: BottomNavigationView  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 setContentView(R.layout.activity\_main)  
 nav = findViewById(R.id.bottomNavigationView)  
 nav.setOnNavigationItemSelectedListener { item ->  
 when (item.itemId) {  
 R.id.addCrime -> {  
 loadFragment(CrimeFragment())  
 return@setOnNavigationItemSelectedListener true  
 }  
 R.id.listCrimes -> {  
// val selectedValue = "action"  
// val fragment = MoviesByGenreFragment.newInstance(selectedValue)  
 loadFragment(CrimeListFragment())  
 return@setOnNavigationItemSelectedListener true  
 }  
 }  
 false  
 }  
 loadFragment(CrimeFragment())  
// val currentFragment = supportFragmentManager.findFragmentById(R.id.fragment\_container)  
// if (currentFragment == null){  
// val fragment = CrimeFragment()  
// supportFragmentManager  
// .beginTransaction()  
// .add(R.id.fragment\_container, fragment)  
// .commit()  
// }  
  
 }  
 private fun loadFragment(fragment: Fragment) {  
 supportFragmentManager.beginTransaction()  
 .replace(R.id.fragment\_container, fragment)  
 .commit()  
 }  
}

CrimeDao.kt

package com.criminal\_intent.android.pract\_19\_1.database  
  
import androidx.lifecycle.LiveData  
import androidx.room.Dao  
import androidx.room.Insert  
import androidx.room.Query  
import androidx.room.Update  
import com.criminal\_intent.android.pract\_19\_1.Crime  
import java.util.\*  
  
@Dao  
interface *CrimeDao* {  
 @Query("SELECT \* FROM crime")  
 fun getCrimes(): LiveData<*List*<Crime>>  
 @Query("SELECT \* FROM crime WHERE id=(:id)")  
 fun getCrime(id: UUID): LiveData<Crime?>  
 @Update  
 fun updateCrime(crime: Crime)  
 @Insert  
 fun addCrime(crime: Crime)  
 @Query("SELECT \* FROM crime WHERE title = :title")  
 suspend fun getCrimeByTitle(title: String): Crime?  
}

CrimeDataBase.kt

package com.criminal\_intent.android.pract\_19\_1.database  
  
import androidx.room.Database  
import androidx.room.RoomDatabase  
import androidx.room.TypeConverters  
import androidx.room.migration.Migration  
import androidx.sqlite.db.*SupportSQLiteDatabase*import com.criminal\_intent.android.pract\_19\_1.Crime  
  
@Database(entities = [ Crime::class ],  
 version=1)  
@TypeConverters(CrimeTypeConverters::class)  
abstract class CrimeDatabase : RoomDatabase() {  
 abstract fun crimeDao(): *CrimeDao*}  
val *migration\_1\_2* = object : Migration(1,2){  
 override fun migrate(database: *SupportSQLiteDatabase*) {  
 database.execSQL("ALTER TABLE Crime ADD COLUMN suspect TEXT NOT NULL DEFAULT ''")  
 }  
}

CrimeTypeConverters.kt

package com.criminal\_intent.android.pract\_19\_1.database  
  
import androidx.room.TypeConverter  
import java.util.\*  
  
class CrimeTypeConverters {  
 @TypeConverter  
 fun fromDate(date: Date?): Long? {  
 return date?.*time* }  
  
 @TypeConverter  
 fun toDate(millisSinceEpoch: Long?): Date?  
 {  
 return millisSinceEpoch?.*let* {  
 Date(**it**)  
 }  
 }  
  
 @TypeConverter  
 fun toUUID(uuid: String?): UUID? {  
 return UUID.fromString(uuid)  
 }  
  
 @TypeConverter  
 fun fromUUID(uuid: UUID?): String? {  
 return uuid?.toString()  
 }  
}

fragment\_crime.xml

<?xml version="1.0" encoding="utf-8"?>  
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:tools="http://schemas.android.com/tools"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 android:orientation="vertical"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_margin="16dp">  
 <TextView  
 style="?android:listSeparatorTextViewStyle"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="@string/crime\_title\_label"/>  
 <EditText  
 android:id="@+id/crime\_title"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:hint="@string/crime\_title\_hint"/>  
 <Button  
 android:textSize="16dp"  
 android:textAllCaps="false"  
 android:id="@+id/addButton"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="Добавить в базу данных"/>  
<!-- <Button-->  
<!-- android:textSize="16dp"-->  
<!-- android:id="@+id/listButton"-->  
<!-- android:layout\_width="match\_parent"-->  
<!-- android:layout\_height="wrap\_content"-->  
<!-- android:textAllCaps="false"-->  
<!-- android:text="Просмотр преступлений"/>-->  
 <TextView  
 style="?android:listSeparatorTextViewStyle"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="@string/crime\_details\_label"/>  
 <Button  
 android:id="@+id/crime\_date"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 tools:text="Wed Nov 14 11:56 EST 2018"/>  
  
 <CheckBox  
 android:id="@+id/crime\_solved"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="@string/crime\_solved\_label"/>  
 <Button  
 android:enabled="false"  
 android:id="@+id/crime\_suspect"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="@string/crime\_suspect\_text"/>  
 <Button  
 android:enabled="false"  
 android:id="@+id/crime\_report"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="@string/crime\_report\_text"/>  
  
 <com.google.android.material.bottomnavigation.BottomNavigationView  
 android:id="@+id/bottomNavigationView"  
 android:layout\_width="0dp"  
 android:background="#55555555"  
 android:layout\_height="wrap\_content"  
 android:layout\_gravity="bottom"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:itemTextColor="@color/white"  
 app:menu="@menu/menu" />  
</LinearLayout>

activity\_main.xml

<?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"  
 android:background="@color/white">  
  
 <FrameLayout  
 android:id="@+id/fragment\_container"  
 android:layout\_width="0dp"  
 android:layout\_height="0dp"  
 app:layout\_constraintBottom\_toTopOf="@id/bottomNavigationView"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent" />  
  
 <com.google.android.material.bottomnavigation.BottomNavigationView  
 android:id="@+id/bottomNavigationView"  
 android:layout\_width="0dp"  
 android:background="#55555555"  
 android:layout\_height="wrap\_content"  
 android:layout\_gravity="bottom"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:itemTextColor="@color/white"  
 app:menu="@menu/menu" />  
  
</androidx.constraintlayout.widget.ConstraintLayout>

fragment\_crime\_list.xml

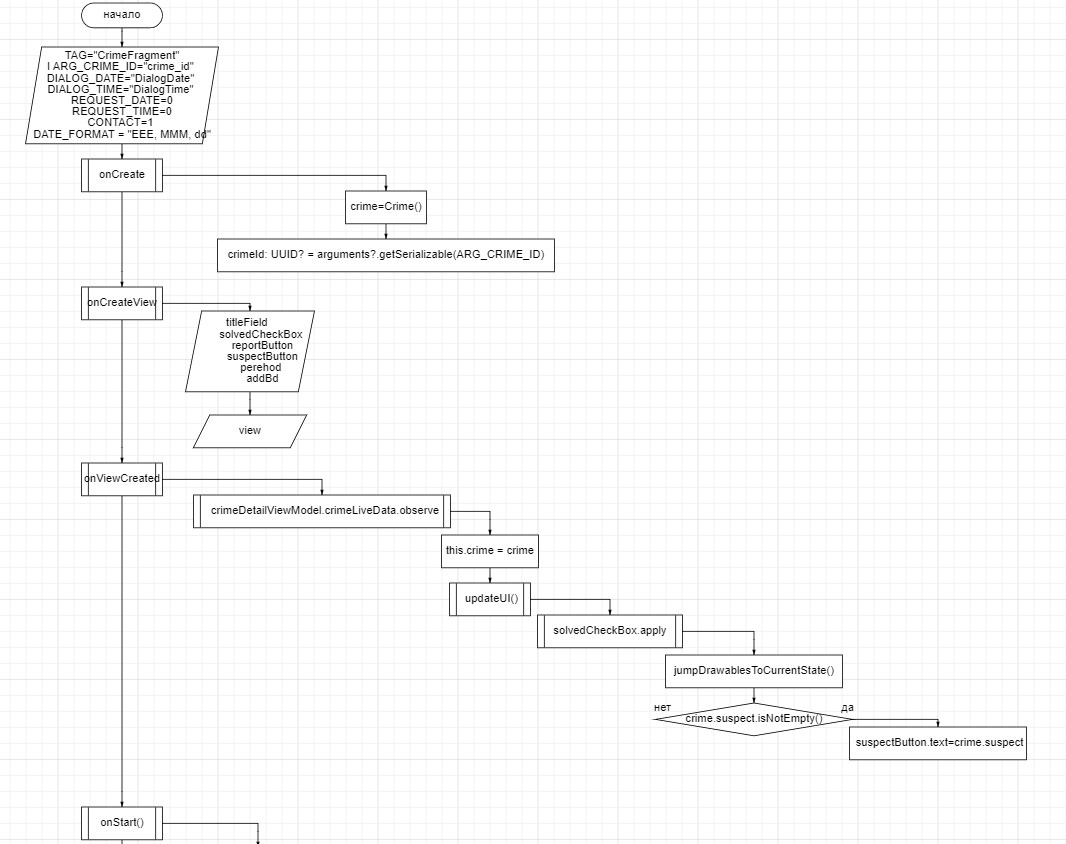
<?xml version="1.0" encoding="utf-8"?>  
<androidx.recyclerview.widget.RecyclerView  
 xmlns:android="http://schemas.android.com/apk/res/android"  
 android:id="@+id/crime\_recycler\_view"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 />

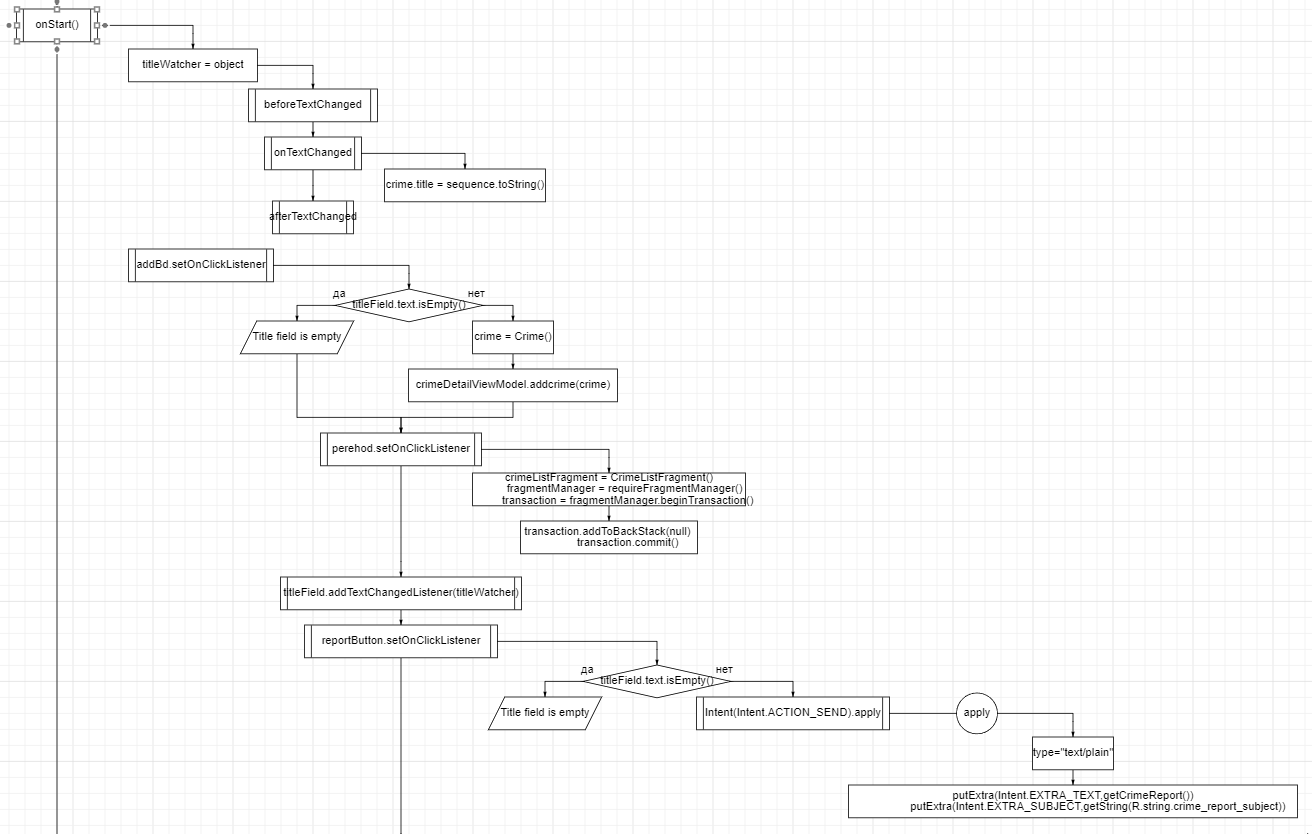
list\_item\_crime.xml

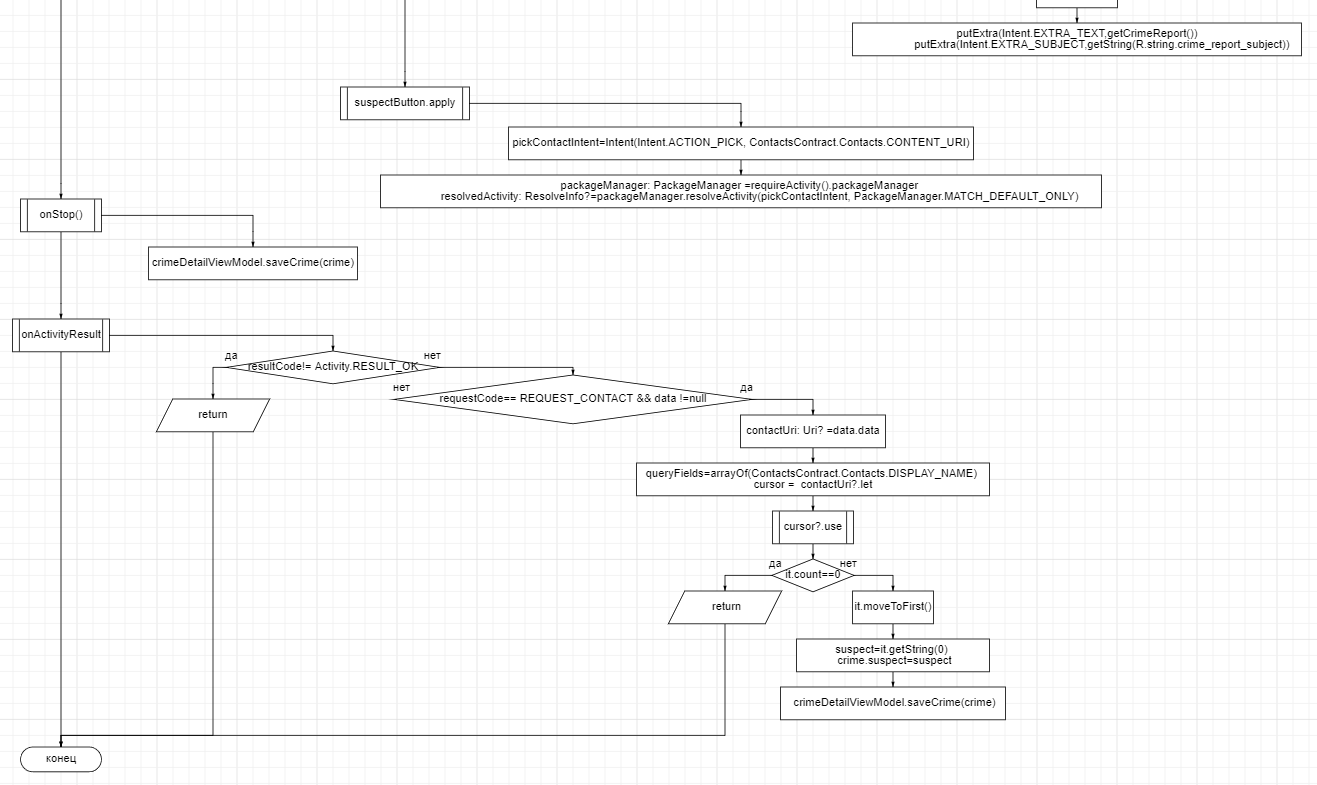
<?xml version="1.0" encoding="utf-8"?>  
<LinearLayout  
 xmlns:android="http://schemas.android.com/apk/res/android"  
 android:orientation="vertical"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:padding="8dp">  
 <TextView  
 android:id="@+id/crime\_title"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="@string/Title2"/>  
 <TextView  
 android:id="@+id/crime\_date"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="@string/Date"/>  
 <TextView  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="@string/Title2"  
 android:textColor="@color/black"  
 android:textStyle="bold"/>  
 <Button  
 android:id="@+id/call\_police"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="40dp"  
 android:layout\_marginStart="20dp"  
 android:textAllCaps="false"  
 android:text="@string/Call"/>  
  
</LinearLayout>

Блок-схема

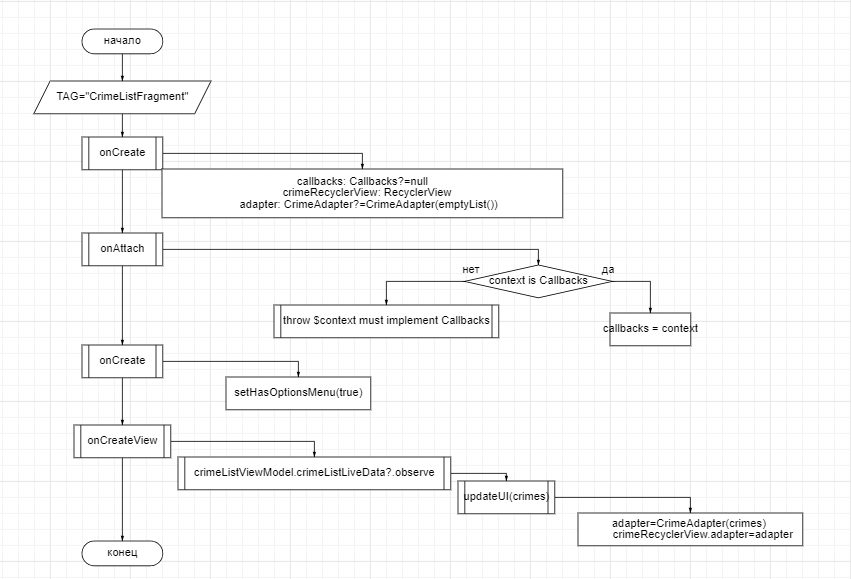
CrimeFragment.kt



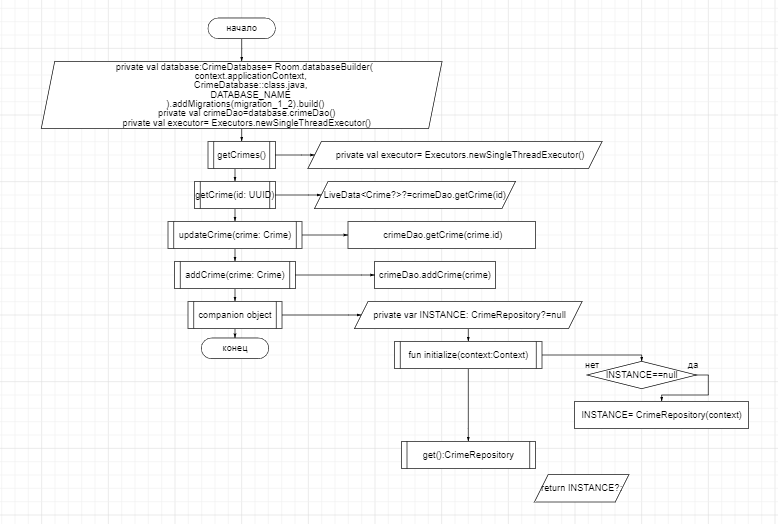




CrimeListFragment.kt



CrimeRepository.kt



**Тестовые ситуации:**

1. Проверка на пустоту строк
2. Проверки на добавлению в базу