Министерство образования и молодежной политики Свердловской области



ГАПОУ СО «Екатеринбургский колледж транспортного строительства»

Отчёт по УП

Задание 4 Неявные интенты.   
Приложение «CriminalImtent»

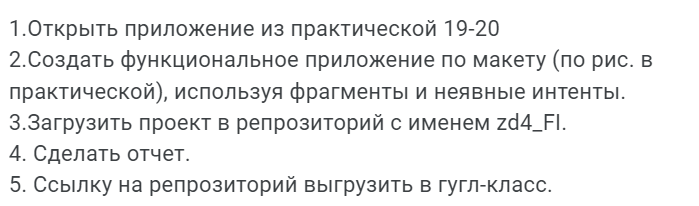
Выполнил: Мартынов Арсений

Группа: ПР-31

Преподаватель: Мирошниченко Г.В

2024

1. **Описание задачи:**

****

**Входные данные:** title - поле для ввода преступления, name - поле для выбора подозреваемых

**Выходные данные:** message – сообщение о преступлении и подозреваемом

1. **Структура проекта:**

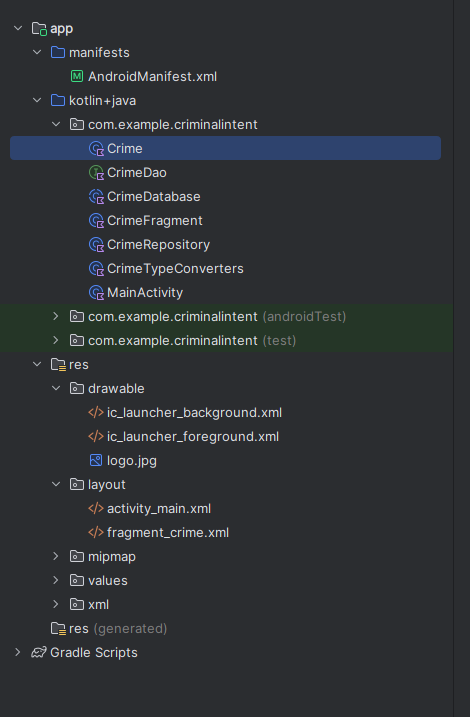


Схема классов

1. **Листинг программы:**

**activity\_main.xml**

<FrameLayout  
 xmlns:android="http://schemas.android.com/apk/res/android"  
 android:id="@+id/fragment\_container"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent" />

**fragment\_crime.xml**

<LinearLayout  
 xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:orientation="vertical"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:layout\_margin="16dp">  
 <ImageView  
 android:layout\_width="match\_parent"  
 android:layout\_height="250dp"  
 android:background="@drawable/iconnn"  
 />  
  
 <TextView  
 style="?android:listSeparatorTextViewStyle"  
 android:textColor="@color/black"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="20dp"  
 android:layout\_marginLeft="20dp"  
 android:layout\_marginRight="20dp"  
 android:text="@string/crime\_title\_label" />  
  
 <EditText  
 android:id="@+id/crime\_title"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:textSize="16dp"  
 android:layout\_marginLeft="20dp"  
 android:layout\_marginRight="20dp"  
 android:hint="@string/crime\_title\_hint" />  
  
 <TextView  
 style="?android:listSeparatorTextViewStyle"  
 android:textColor="@color/black"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginLeft="20dp"  
 android:layout\_marginRight="20dp"  
 android:layout\_marginBottom="8dp"  
 android:text="@string/crime\_details\_label" />  
  
 <Button  
 android:id="@+id/crime\_date"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:background="@drawable/button"  
 android:layout\_marginLeft="20dp"  
 android:layout\_marginRight="20dp"  
 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:layout\_marginBottom="30dp"  
 android:layout\_marginLeft="20dp"  
 android:text="@string/crime\_solved\_label" />  
  
 <Button  
 android:id="@+id/crime\_suspect"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:background="@color/black"  
 android:textColor="@color/white"  
 android:layout\_marginLeft="20dp"  
 android:layout\_marginRight="20dp"  
 android:textSize="12dp"  
 android:text="@string/crime\_suspect\_text"  
 android:layout\_marginBottom="8dp" />  
  
 <Button  
 android:id="@+id/crime\_report"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:background="@color/black"  
 android:textColor="@color/white"  
 android:layout\_marginLeft="20dp"  
 android:layout\_marginRight="20dp"  
 android:textSize="12dp"  
 android:text="@string/crime\_report\_text"  
 android:layout\_marginBottom="8dp"/>  
 <Button  
 android:id="@+id/call\_suspect"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:background="@color/black"  
 android:textSize="12dp"  
 android:textColor="@color/white"  
 android:layout\_marginRight="20dp"  
 android:layout\_marginLeft="20dp"  
 android:text="@string/call\_suspect" />  
</LinearLayout>

**MainActivity.kt**

class MainActivity : AppCompatActivity() {  
  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 setContentView(R.layout.*activity\_main*)  
  
 val currentFragment = *supportFragmentManager*.findFragmentById(R.id.*fragment\_container*)  
  
 if (currentFragment == null) {  
 val fragment = CrimeFragment()  
 *supportFragmentManager*.beginTransaction()  
 .add(R.id.*fragment\_container*, fragment)  
 .commit()  
 }  
 }  
}

**CrimeTypeConverters.kt**

class CrimeTypeConverters {  
 @TypeConverter  
 fun fromDate(date: Date?): Long? {  
 return date?.*time* }  
  
 @TypeConverter  
 fun toDate(millisSinceEpoch: Long?): Date? {  
 return millisSinceEpoch?.*let* **{** Date(**it**) **}** }  
  
 @TypeConverter  
 fun fromUUID(uuid: UUID?): String? {  
 return uuid?.toString()  
 }  
  
 @TypeConverter  
 fun toUUID(uuid: String?): UUID? {  
 return UUID.fromString(uuid)  
 }  
}

**CrimeRepository.kt**

class CrimeRepository private constructor(context: Context) {  
  
 private val database: CrimeDatabase = Room.databaseBuilder(  
 context.*applicationContext*,  
 CrimeDatabase::class.*java*,  
 CrimeDatabase.DATABASE\_NAME  
 )  
 .addMigrations(CrimeDatabase.migration\_1\_2)  
 .build()  
  
 private val crimeDao = database.crimeDao()  
  
 fun getCrimes(): LiveData<List<Crime>> {  
 return crimeDao.getCrimes()  
 }  
  
 fun getCrime(id: UUID): LiveData<Crime?> {  
 return crimeDao.getCrime(id)  
 }  
  
 fun addCrime(crime: Crime) {  
 crimeDao.addCrime(crime)  
 }  
  
 fun updateCrime(crime: Crime) {  
 crimeDao.updateCrime(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")  
 }  
 }  
}

**CrimeFragment.kt**

class CrimeFragment : Fragment() {  
  
 private lateinit var crime: Crime  
 private lateinit var titleField: EditText  
 private lateinit var dateButton: Button  
 private lateinit var solvedCheckBox: CheckBox  
 private lateinit var suspectButton: Button  
 private lateinit var reportButton: Button  
 private lateinit var callButton: Button  
  
 private val REQUEST\_CALL\_PERMISSION = 1  
  
 var phone: String? = null  
  
 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)  
  
 titleField = 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  
 suspectButton = view.findViewById(R.id.*crime\_suspect*) as Button  
 reportButton = view.findViewById(R.id.*crime\_report*) as Button  
 callButton = view.findViewById(R.id.*call\_suspect*) 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?) {}  
 }  
  
 titleField.addTextChangedListener(titleWatcher)  
  
 solvedCheckBox.setOnCheckedChangeListener **{** \_, isChecked **->** crime.isSolved = isChecked  
  
  
 val message = if (isChecked) {  
 "Радость"  
 } else {  
 "Тоска"  
 }  
  
  
 Toast.makeText(*context*, message, Toast.*LENGTH\_SHORT*).show()  
 **}** 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*)  
 **}** 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)  
 **}  
 }** callButton.setOnClickListener **{** val callIntent = Intent(Intent.*ACTION\_CALL*).*apply* **{** *data* = Uri.parse("tel:${phone}")  
 **}** if (requireContext().checkSelfPermission(android.Manifest.permission.*CALL\_PHONE*) == PackageManager.*PERMISSION\_GRANTED*) {  
 startActivity(callIntent)  
 } else {  
 requestPermissions(*arrayOf*(android.Manifest.permission.*CALL\_PHONE*), REQUEST\_CALL\_PERMISSION)  
 }  
 **}** }  
  
 @SuppressLint("Range")  
 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.*\_ID*, ContactsContract.Contacts.*DISPLAY\_NAME*)  
 if (contactUri != null) {  
 val cursor = requireActivity().*contentResolver*.query(  
 contactUri,  
 queryFields,  
 null,  
 null,  
 null  
 )  
 cursor?.*use* **{** if (**it**.*count* > 0) {  
 **it**.moveToFirst()  
 val suspect =  
 **it**.getString(**it**.getColumnIndex(ContactsContract.Contacts.*DISPLAY\_NAME*))  
 crime.suspect = suspect  
 suspectButton.*text* = suspect  
  
  
 val contactId =  
 **it**.getString(**it**.getColumnIndex(ContactsContract.Contacts.*\_ID*))  
 val phoneCursor = requireActivity().*contentResolver*.query(  
 ContactsContract.CommonDataKinds.Phone.*CONTENT\_URI*,  
 null,  
 "${ContactsContract.CommonDataKinds.Phone.*CONTACT\_ID*} = ?",  
 *arrayOf*(contactId),  
 null  
 )  
  
 phoneCursor?.*use* **{** pc **->** if (pc.moveToFirst()) {  
 phone =  
 pc.getString(pc.getColumnIndex(ContactsContract.CommonDataKinds.Phone.*NUMBER*))  
 }  
 **}** }  
 **}** } else {  
 }  
 }  
 }  
 }  
  
 private fun getCrimeReport(): String {  
 val solvedString = if (crime.suspect.*isBlank*()) {  
 getString(R.string.*crime\_report\_solved*)  
 } else {  
 getString(R.string.*crime\_report\_unsolved*)  
 }  
  
 val dateString = DateFormat.format(DATE\_FORMAT, crime.date).toString()  
  
 val 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)  
 }  
  
 companion object {  
 private const val REQUEST\_CONTACT = 1  
 private const val DATE\_FORMAT = "EEE, MMM, dd"  
 }  
}

**CrimeDatabase.kt**

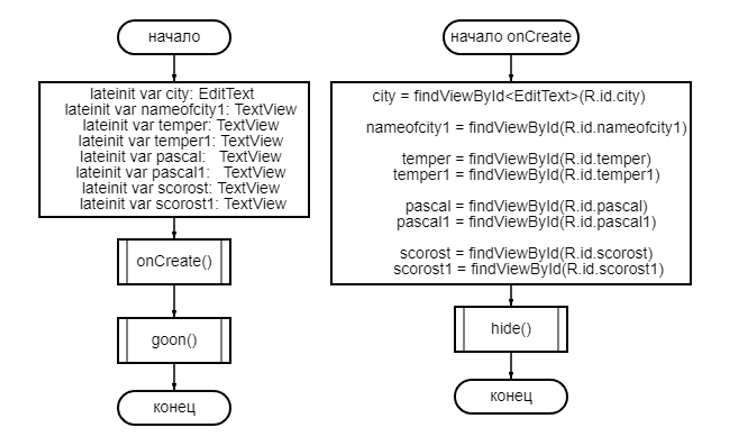
@Database(entities = [Crime::class], version = 2)  
@TypeConverters(CrimeTypeConverters::class)  
abstract class CrimeDatabase : RoomDatabase() {  
 abstract fun crimeDao(): CrimeDao  
  
 companion object {  
 const val DATABASE\_NAME = "crime-database"  
  
 val migration\_1\_2 = object : Migration(1, 2) {  
 override fun migrate(database: SupportSQLiteDatabase) {  
 database.execSQL("ALTER TABLE Crime ADD COLUMN suspect TEXT NOT NULL DEFAULT ''")  
 }  
 }  
 }  
}

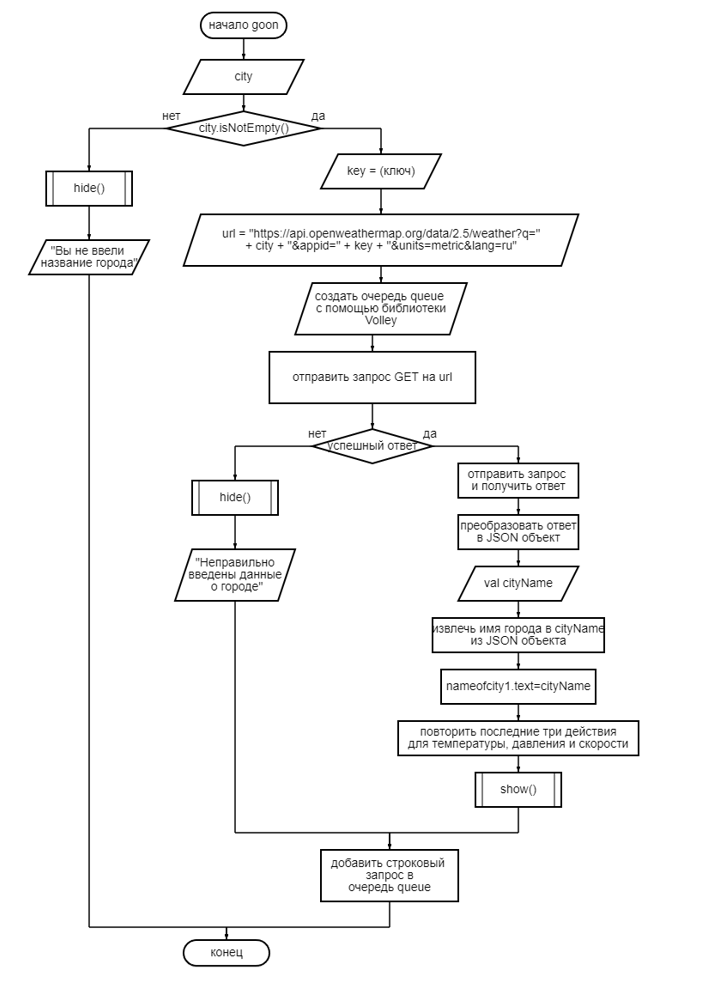
**CrimeDao.kt**

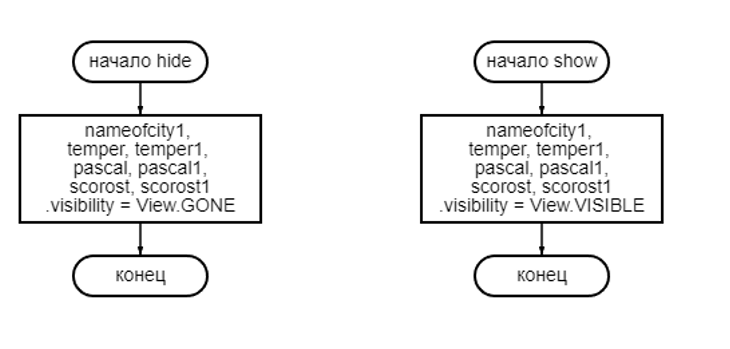
@Dao  
interface CrimeDao {  
 @Query("SELECT \* FROM crime")  
 fun getCrimes(): LiveData<List<Crime>>  
  
 @Query("SELECT \* FROM crime WHERE id=(:id)")  
 fun getCrime(id: UUID): LiveData<Crime?>  
  
 @Insert  
 fun addCrime(crime: Crime)  
  
 @Update  
 fun updateCrime(crime: Crime)  
}

**Crime.kt**

@Dao(tableName = "crime")  
data class Crime(  
 @PrimaryKey var id: UUID = UUID.randomUUID(),  
 @ColumnInfo(name = "title") var title: String = "",  
 @ColumnInfo(name = "date") var date: Date = Date(),  
 @ColumnInfo(name = "is\_solved") var isSolved: Boolean = false,  
 @ColumnInfo(name = "suspect") var suspect: String = "",  
 @ColumnInfo(name = "phone") var phone: String = ""  
) {  
 @Ignore  
 constructor(title: String, date: Date, isSolved: Boolean, suspect: String) : this(  
 UUID.randomUUID(),  
 title,  
 date,  
 isSolved,  
 suspect  
 )  
}

1. **Алгоритм решения:**

****



1. **Используемые библиотеки:**

**Android Аннотации**

* **android.annotation.SuppressLint**  
  Используется для подавления предупреждений компилятора о потенциальных проблемах в коде.

**Android Графика**

* **android.graphics.Color**  
  Предоставляет константы и методы для работы с цветами.

**Android Активности**

* **androidx.appcompat.app.AppCompatActivity**  
  Базовый класс для активностей, который обеспечивает поддержку старых версий Android и позволяет использовать компоненты Material Design.

**Android Системные классы**

* **android.os.Bundle**  
  Используется для передачи данных между активностями и сохранения состояния активности.

**Android Виджеты**

* **android.widget.GridLayout**  
  Контейнер, который позволяет размещать дочерние элементы в виде сетки.
* **android.widget.ImageView**  
  Элемент пользовательского интерфейса для отображения изображений.
* **android.widget.LinearLayout**  
  Контейнер, который размещает дочерние элементы в одном направлении (горизонтально или вертикально).
* **android.widget.TextView**  
  Элемент пользовательского интерфейса для отображения текста.

**Сторонние библиотеки**

* **com.google.gson.Gson**  
  Библиотека для работы с JSON в Java и Kotlin. Позволяет легко сериализовать и десериализовать объекты, преобразуя их в JSON и обратно.
* **com.squareup.picasso.Picasso**  
  Библиотека для загрузки и кэширования изображений в Android. Упрощает процесс загрузки изображений из интернета в ImageView.
* **okhttp3**  
  Мощный HTTP-клиент для Android и Java. Используется для отправки и получения сетевых запросов на основе HTTP.

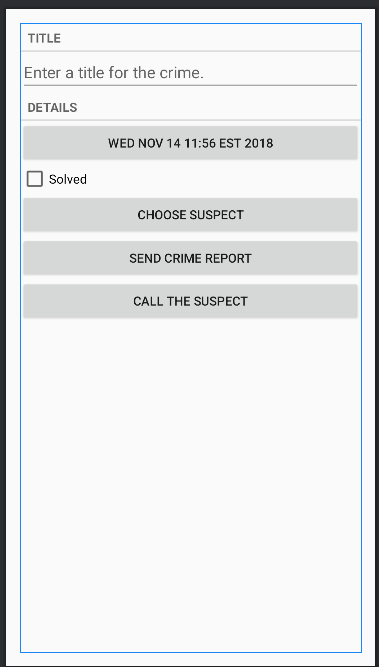
1. **Используемые инструменты:**

Android Studio

Kotlin

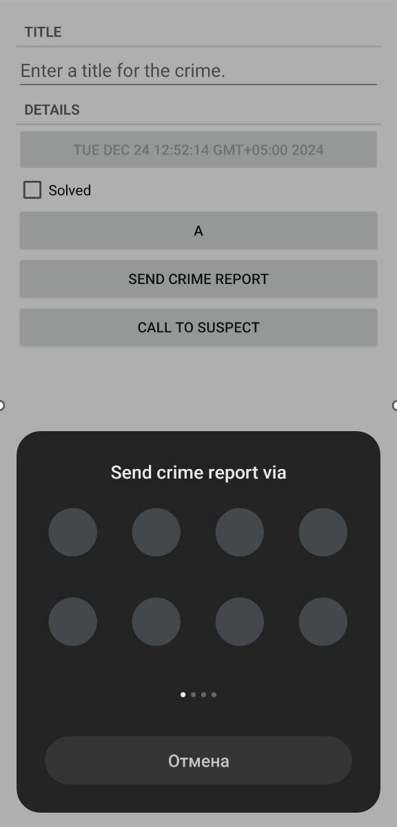
1. **Описание пользовательского интерфейса:**

После входа в приложение, пользователя встречает базовый экран:



Пользователь должен ввести наименование преступления, а затем выбрать подозреваемого из своей телефонной книги:

|  |  |
| --- | --- |
|  |  |



Затем пользователь может отправить сообщение о преступлении:

И / или позвонить подозреваемому:

