Министерство образования и молодежной политики Свердловской области



ГАПОУ СО «Екатеринбургский колледж транспортного строительства»

# Отчёт по программе «Задание 7 - Неявные интенты»

Выполнил: Годовых Егор

Группа: ПР-32

Преподаватель: Мирошниченко Г. В.

2025

**Задание:**

1.Открыть приложение из практической 19-20  
2.Cоздать функциональное приложение по макету (по рис. в практической), используя фрагменты и неявные интенты.

**Входные и выходные данные**

* titleField
* dateButton
* suspectButton
* reportButton

**Листинг программы (если есть)**

Crime

package com.example.criminalintent  
  
import androidx.room.ColumnInfo  
import androidx.room.Entity  
import androidx.room.Ignore  
import androidx.room.PrimaryKey  
import java.util.Date  
import java.util.UUID  
  
@Entity(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 = ""  
) {  
 @Ignore  
 constructor(title: String, date: Date, isSolved: Boolean, suspect: String) : this(  
 UUID.randomUUID(),  
 title,  
 date,  
 isSolved,  
 suspect  
 )  
}

CrimeDao

package com.example.criminalintent  
  
import androidx.lifecycle.LiveData  
import androidx.room.Dao  
import androidx.room.Insert  
import androidx.room.Query  
import androidx.room.Update  
import java.util.UUID  
  
@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)  
}

CrimeDatabase

package com.example.criminalintent  
  
import androidx.room.Database  
import androidx.room.RoomDatabase  
import androidx.room.TypeConverters  
import androidx.room.migration.Migration  
import androidx.sqlite.db.SupportSQLiteDatabase  
  
@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 ''")  
 }  
 }  
 }  
}

CrimeFragment

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.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 androidx.fragment.app.Fragment  
import com.example.criminalintent.Crime  
import com.example.criminalintent.R  
  
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  
  
 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  
  
 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  
 **}** 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*)  
 *isEnabled* = true  
 **}** 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)  
 **}  
 }** }  
  
 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*)  
 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(0)  
 crime.suspect = suspect  
 suspectButton.*text* = suspect  
 }  
 **}** } else {  
 // Обработка случая, когда contactUri равен null  
 }  
 }  
 }  
 }  
  
 private fun getCrimeReport(): String {  
 val solvedString = if (crime.isSolved) {  
 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"  
 }  
}

CrimeRepository

package com.example.criminalintent  
  
import android.content.Context  
import androidx.lifecycle.LiveData  
import androidx.room.Room  
import java.util.UUID  
  
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")  
 }  
 }  
}

CrimeTypeConverters

package com.example.criminalintent  
  
import androidx.room.TypeConverter  
import java.util.Date  
import java.util.UUID  
  
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)  
 }  
}

MainActivity

package com.example.criminalintent  
  
import CrimeFragment  
import android.os.Bundle  
import androidx.appcompat.app.AppCompatActivity  
import com.example.criminalintent.R  
  
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()  
 }  
 }  
}

activity\_main

<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

<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">  
  
 <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" />  
  
 <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:id="@+id/crime\_suspect"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="@string/crime\_suspect\_text" />  
  
 <Button  
 android:id="@+id/crime\_report"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="@string/crime\_report\_text" />  
</LinearLayout>

**Тестовые ситуации**

