

Министерство образования и молодежной политики Свердловской области

ГАПОУ СО «Екатеринбургский колледж транспортного строительства»

Отчёт по учебной практике

УП 01.01

«Неявные интенты»

Выполнила:Оберюхтина Елена

Группа: ПР-31

Преподаватель: Мирошниченко Г.В.

2023

**Задание:**

Cоздать функциональное приложение по макету, используя фрагменты и неявные интенты.

**Входные данные:**

crime\_title:пользователь вводит название дела.

**Листинг программы:**

CrimeDao:

package com.example.practica20.database  
  
import androidx.lifecycle.LiveData  
import androidx.room.Dao  
import androidx.room.Insert  
import androidx.room.Query  
import androidx.room.Update  
import com.example.practica20.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)  
}

CrimeDatabase:

package com.example.practica20.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.example.crimese.database.CrimeTypeConverters  
import com.example.practica20.Crime  
  
  
@Database(entities = [Crime::class], version = 1, exportSchema = false)  
@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:

package com.example.practica20.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()  
 }  
}

Crime:

package com.example.practica20  
  
import androidx.room.Entity  
import androidx.room.PrimaryKey  
import java.util.\*  
  
@Entity  
data class Crime (@PrimaryKey var id: UUID = UUID.randomUUID()) {  
 var title:String = ""  
 var date: Date? = Date()  
 var isSolved: Boolean? = false  
 var suspect: String=""  
 var requiresPolice: Int?=0  
 constructor(id: UUID, title: String, date: Date, isSolved:Boolean, suspect:String):this(id){  
 this.title = title  
 this.date = date  
 this.id = id  
 this.isSolved=isSolved  
 this.suspect=suspect  
 //this.requiresPolice=requiresPolice  
 }  
}

CrimeDatailModel:

package com.example.practica20  
  
import androidx.lifecycle.LiveData  
import androidx.lifecycle.MutableLiveData  
import androidx.lifecycle.Transformations  
import androidx.lifecycle.ViewModel  
import java.util.\*  
  
class CrimeDatailModel(): ViewModel() {  
 private val crimeRepository=CrimeRepository.get()  
 private val crimeIdLiveData= MutableLiveData<UUID>()  
 var crimeLiveData: LiveData<Crime?> =  
 Transformations.switchMap(crimeIdLiveData)**{** crimeRepository.getCrime(**it**)  
 **}** fun loadCrime(crimeId: UUID) {  
 crimeIdLiveData.*value* = crimeId  
 }  
  
 fun saveCrime(crime: Crime) {  
 crimeRepository.updateCrime(crime)  
 }  
 fun addcrime(crime: Crime){  
 crimeRepository.addCrime(crime)  
 }  
}

CrimeFragment:

package com.example.practica20  
  
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.util.Log  
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.annotation.RequiresApi  
import androidx.core.view.ViewCompat.jumpDrawablesToCurrentState  
import androidx.fragment.app.Fragment  
import androidx.lifecycle.ViewModelProviders  
import java.util.\*  
  
private const val *TAG*="CrimeFragment"  
private const val *ARG\_CRIME\_ID*="crime\_id"  
private const val *REQUEST\_CONTACT*=1  
private const val *DATE\_FORMAT* = "EEE, MMM, dd"  
class CrimeFragment : Fragment(),DatePickerFragment.Callbacks, TimePickerFragment.Callbacks, CrimeListFragment.Callbacks {  
 private lateinit var crime: Crime  
 private lateinit var titleField: EditText  
 private lateinit var solvedCheckBox: CheckBox  
 private lateinit var reportButton: Button  
 private lateinit var suspectButton: Button  
 private lateinit var perehod: Button  
 private lateinit var addBd: Button  
 private lateinit var dateButton: Button  
  
 private val crimeDetailViewModel:CrimeDatailModel by *lazy***{** ViewModelProviders.of(this).get(CrimeDatailModel::class.*java*)  
 **}** override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 crime=Crime()  
 val crimeId: UUID? = *arguments*?.getSerializable(*ARG\_CRIME\_ID*) as? UUID  
  
 Log.d(*TAG*, "args bundle crime ID: $crimeId")  
  
 }  
  
  
 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  
 reportButton=view.findViewById(R.id.*crime\_report*) as Button  
 suspectButton=view.findViewById(R.id.*crime\_suspect*) as Button  
 perehod = view.findViewById(R.id.*perehodList*) as Button  
 addBd = view.findViewById(R.id.*addBaza*) as Button  
 dateButton.*apply* **{** *text* = crime.date.*toString*(); *isEnabled* = false **}** return view  
 }  
 override fun onViewCreated(view: View, savedInstanceState: Bundle?){  
 super.onViewCreated(view, savedInstanceState)  
 crimeDetailViewModel.crimeLiveData.observe(  
 *viewLifecycleOwner*,  
 androidx.lifecycle.*Observer* **{** crime **->** crime?.*let* **{** this.crime=crime  
 updateUI()  
 **}  
 }** )  
 }  
  
 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?) {  
  
 }  
 }  
 //сохранение в бд  
 addBd.setOnClickListener()**{** var crime = Crime()  
 crime.title = titleField.*text*.toString()  
 crime.date = Date()  
 crime.isSolved = solvedCheckBox.*isChecked* crimeDetailViewModel.addcrime(crime)  
 **}** //доступность кнопок  
 solvedCheckBox.setOnClickListener()**{** suspectButton.*isEnabled* = solvedCheckBox.*isChecked* reportButton.*isEnabled* = solvedCheckBox.*isChecked* **}** //просмотр листа  
 perehod.setOnClickListener()**{**// Создаем новый экземпляр вашего фрагмента CrimeListFragment  
 val crimeListFragment = CrimeListFragment()  
  
 // Получаем объект FragmentManager  
 val fragmentManager = requireFragmentManager()  
  
 // Начинаем транзакцию фрагментов  
 val transaction = fragmentManager.beginTransaction()  
  
 // Заменяем текущий фрагмент на CrimeListFragment  
 transaction.replace(R.id.*fragment\_container*, crimeListFragment)  
  
 // Добавляем транзакцию в стек возврата, чтобы пользователь мог вернуться к предыдущему фрагменту  
 transaction.addToBackStack(null)  
  
 // Применяем транзакцию  
 transaction.commit()  
 **}** titleField.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*)  
 **}** //pickContactIntent.addCategory(Intent.CATEGORY\_HOME)  
 val packageManager: PackageManager =requireActivity().*packageManager* val resolvedActivity: ResolveInfo?=packageManager.resolveActivity(pickContactIntent, PackageManager.*MATCH\_DEFAULT\_ONLY*)  
 if (resolvedActivity==null){  
 *isEnabled*=false  
 }  
 **}** }  
  
 override fun onStop(){  
 super.onStop()  
 crimeDetailViewModel.saveCrime(crime)  
 }  
  
 override fun onDateSelected(date: Date?) {  
 crime.date=date  
 updateUI()  
 }  
  
 override fun onTimeSelected(time: Date) {  
 crime.date=time  
 updateUI()  
 }  
 private fun updateUI(){  
 titleField.setText(crime.title)  
 solvedCheckBox.*apply* **{** *isChecked*=crime.isSolved!!  
 jumpDrawablesToCurrentState()  
 **}** if (crime.suspect.*isNotEmpty*()){  
 suspectButton.*text*=crime.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.saveCrime(crime)  
 suspectButton.*text*=suspect  
 **}** }  
 }  
 //updateUI()  
 }  
 private fun getCrimeReport(): String{//7 задание  
 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)  
 }  
 companion object{  
 fun newInstance(crimeId:UUID):CrimeFragment{  
 val args=Bundle().*apply* **{** putSerializable(*ARG\_CRIME\_ID*, crimeId) **}** return CrimeFragment().*apply* **{** *arguments*=args  
 **}** }  
 }  
  
 override fun onCrimeSelected(crimeId: UUID) {  
 *TODO*("Not yet implemented")  
 }  
}

CrimeListFragment:

package com.example.practica20  
  
import android.content.Context  
import android.os.Bundle  
import android.util.Log  
import android.view.\*  
import android.widget.TextView  
import android.widget.Toast  
import androidx.fragment.app.Fragment  
import androidx.lifecycle.ViewModelProviders  
import androidx.recyclerview.widget.LinearLayoutManager  
import androidx.recyclerview.widget.RecyclerView  
import java.util.\*  
  
private const val *TAG*="CrimeListFragment"  
class CrimeListFragment : Fragment() {  
 interface Callbacks{  
 fun onCrimeSelected(crimeId: UUID)  
 }  
 private var callbacks: Callbacks?=null  
 private lateinit var crimeRecyclerView: RecyclerView  
 private var adapter: CrimeAdapter?=CrimeAdapter(*emptyList*())  
 private val crimeListViewModel:CrimeListViewModel by *lazy***{** ViewModelProviders.of(this).get(CrimeListViewModel::class.*java*)  
  
 **}** override fun onAttach(context: Context?) {  
 super.onAttach(context)  
 if (context is Callbacks) {  
 callbacks = context  
 } else {  
 throw ClassCastException("$context must implement Callbacks")  
 }  
 }  
  
 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)  
 crimeRecyclerView=view.findViewById(R.id.*crime\_recycler\_view*) as RecyclerView  
 crimeRecyclerView.*layoutManager*= LinearLayoutManager(*context*)  
 crimeRecyclerView.*adapter*=adapter  
 //updateUI()  
 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*,"Got crimes ${crimes.size}")  
 updateUI(crimes) **}  
 }** )  
 }  
  
 override fun onDetach() {  
 super.onDetach()  
 callbacks=null  
 }  
  
  
  
  
 private fun updateUI(crimes:List<Crime>){  
 adapter=CrimeAdapter(crimes)  
 crimeRecyclerView.*adapter*=adapter  
 }  
 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\_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){  
 callbacks?.onCrimeSelected(crime.id)  
 }  
 }  
 companion object{  
 fun newInstance() : CrimeListFragment{  
 return CrimeListFragment()  
 }  
 }  
}

CrimeListViewModel:

package com.example.practica20  
  
import androidx.lifecycle.ViewModel  
  
class CrimeListViewModel : ViewModel() {  
 private val crimeRepository=CrimeRepository.get()  
 val crimeListLiveData=crimeRepository.getCrimes()  
 fun addCrime(crime:Crime){  
 crimeRepository.addCrime(crime)  
 }  
  
}

CrimeRepository:

package com.example.practica20  
  
import android.content.Context  
import androidx.lifecycle.LiveData  
import androidx.room.Room  
import com.example.practica20.database.CrimeDatabase  
import com.example.practica20.database.*migration\_1\_2*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)  
 **}** }  
 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:

package com.example.practica20  
  
import android.app.Application  
  
class CriminalIntentApplication : Application() {  
 override fun onCreate(){  
 super.onCreate()  
 CrimeRepository.initialize(this)  
 }  
}

DatePickerFragment:

package com.example.practica20  
  
import android.app.DatePickerDialog  
import android.app.Dialog  
import android.os.Build  
import android.os.Bundle  
import android.widget.DatePicker  
import androidx.annotation.RequiresApi  
import androidx.fragment.app.DialogFragment  
import java.util.\*  
  
private const val *ARG\_DATE*="date"  
class DatePickerFragment : DialogFragment() {  
 interface Callbacks{  
 fun onDateSelected(date: Date?)  
 }  
 @RequiresApi(Build.VERSION\_CODES.*N*)  
 override fun onCreateDialog(savedInstanceState: Bundle?): Dialog {  
 val dateListener= DatePickerDialog.OnDateSetListener **{** \_: DatePicker, year: Int,  
 month: Int, day: Int **->** val resultDate : Date =  
 GregorianCalendar(year, month, day).*time  
  
 targetFragment*?.*let* **{** fragment **->** (fragment as  
 Callbacks).onDateSelected(resultDate)**}  
 }** val date= *arguments*?.getSerializable(*ARG\_DATE*) as Date  
 var calendar = Calendar.getInstance()  
 calendar.*time*=date  
 var initialYear=calendar.get(Calendar.*YEAR*)  
 var initialMonth=calendar.get(Calendar.*MONTH*)  
 var initialDay=calendar.get(Calendar.*DAY\_OF\_MONTH*)  
 return DatePickerDialog(requireContext(),dateListener,initialYear,initialMonth,initialDay)  
 }  
 companion object{  
 fun newInstance(date: Date?):DatePickerFragment{  
 val args=Bundle().*apply* **{** putSerializable(*ARG\_DATE*,date)  
 **}** return DatePickerFragment().*apply* **{** *arguments*=args  
 **}** }  
 }  
}

MainActivity:

package com.example.practica20  
  
import androidx.appcompat.app.AppCompatActivity  
import android.os.Bundle  
import java.util.\*  
private const val *TAG*="MainActivity"  
class MainActivity : AppCompatActivity(), CrimeListFragment.Callbacks {  
 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()  
 }  
 }  
 override fun onCrimeSelected(crimeId: UUID) {  
 *TODO*("Not yet implemented")  
 }  
}

TimePickerFragment:

package com.example.practica20  
  
import android.app.Dialog  
import android.app.TimePickerDialog  
import android.os.Build  
import android.os.Bundle  
import android.widget.TimePicker  
import androidx.annotation.RequiresApi  
import androidx.fragment.app.DialogFragment  
import java.util.\*  
  
private const val *ARG\_TIME*="time"  
class TimePickerFragment : DialogFragment() {  
 interface Callbacks{  
 fun onTimeSelected(time: Date)  
 }  
 @RequiresApi(Build.VERSION\_CODES.*N*)  
 override fun onCreateDialog(savedInstanceState: Bundle?): Dialog {  
 val timeListener= TimePickerDialog.OnTimeSetListener **{** \_: TimePicker, hour: Int, minute: Int **->** val resultTime : Date =  
 GregorianCalendar(0, 0, 0,hour,minute).*time  
 targetFragment*?.*let* **{** fragment **->** (fragment as  
 Callbacks).onTimeSelected(resultTime)**}  
 }** val time=*arguments*?.getSerializable(*ARG\_TIME*) as Date  
 val calendar=Calendar.getInstance()  
 calendar.*time*=time  
 var initialHour=calendar.get(Calendar.*HOUR*)  
 var initialMinute=calendar.get(Calendar.*MINUTE*)  
 return TimePickerDialog(requireContext(),timeListener,initialHour,initialMinute, true)  
 }  
 companion object{  
 fun newInstance(time: Date?):TimePickerFragment{  
 var args=Bundle().*apply* **{** putSerializable(*ARG\_TIME*,time)  
 **}** return TimePickerFragment().*apply* **{** *arguments*=args  
 **}** }  
 }  
}

activity\_main:

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

fragment\_crime:

<?xml version="1.0" encoding="utf-8"?>  
<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"></TextView>  
 <EditText  
 android:id="@+id/crime\_title"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:hint="@string/crime\_title\_hint"></EditText>  
 <TextView  
 style="?android:listSeparatorTextViewStyle"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="@string/crime\_details\_label"></TextView>  
  
 <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" />  
 <Button  
 android:enabled="true"  
 android:id="@+id/addBaza"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="SAVE"></Button>  
 <Button  
 android:enabled="true"  
 android:id="@+id/perehodList"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="LIST"></Button>  
 <CheckBox  
 android:id="@+id/crime\_solved"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="@string/crime\_solved\_label"></CheckBox>  
 <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>  
 <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"></Button>  
  
</LinearLayout>

fragment\_crime\_list:

<?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">  
  
</androidx.recyclerview.widget.RecyclerView>

list\_item\_crime:

<?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="Crime Title"></TextView>  
 <TextView  
 android:id="@+id/crime\_date"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="Crime Date"></TextView>  
</LinearLayout>

list\_item\_crime\_police:

<?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\_police"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="Crime Title"></TextView>  
 <TextView  
 android:id="@+id/crime\_date\_police"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="Crime Date"></TextView>  
  
 <Button  
 android:layout\_width="155dp"  
 android:layout\_height="wrap\_content"  
 android:text="@string/for\_police"  
 android:layout\_gravity="center"></Button>  
</LinearLayout>

AndroidManifest:

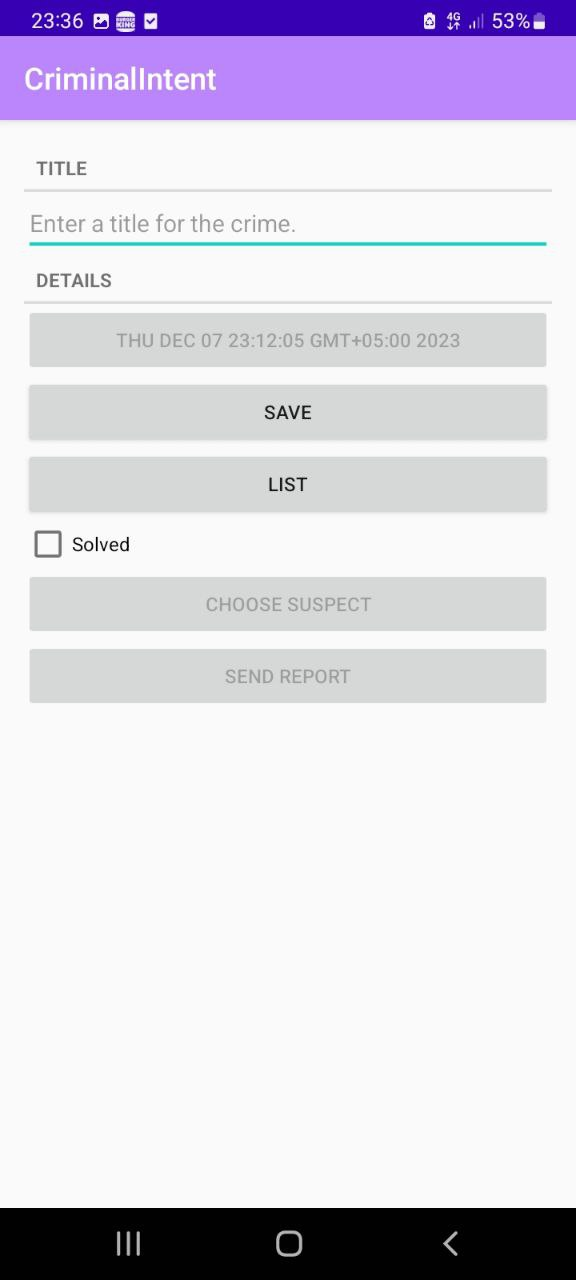
<?xml version="1.0" encoding="utf-8"?>  
<manifest xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:tools="http://schemas.android.com/tools"  
 package="com.example.practica20">  
  
 <application  
 android:name=".CriminalIntentApplication"  
 android:allowBackup="true"  
 android:dataExtractionRules="@xml/data\_extraction\_rules"  
 android:fullBackupContent="@xml/backup\_rules"  
 android:icon="@mipmap/ic\_launcher"  
 android:label="@string/app\_name"  
 android:roundIcon="@mipmap/ic\_launcher\_round"  
 android:supportsRtl="true"  
 android:theme="@style/Theme.Practica20"  
 tools:targetApi="31">  
 <activity  
 android:name=".MainActivity"  
 android:exported="true">  
 <intent-filter>  
 <action android:name="android.intent.action.MAIN" />  
  
 <category android:name="android.intent.category.LAUNCHER" />  
 </intent-filter>  
 </activity>  
 </application>  
  
</manifest>

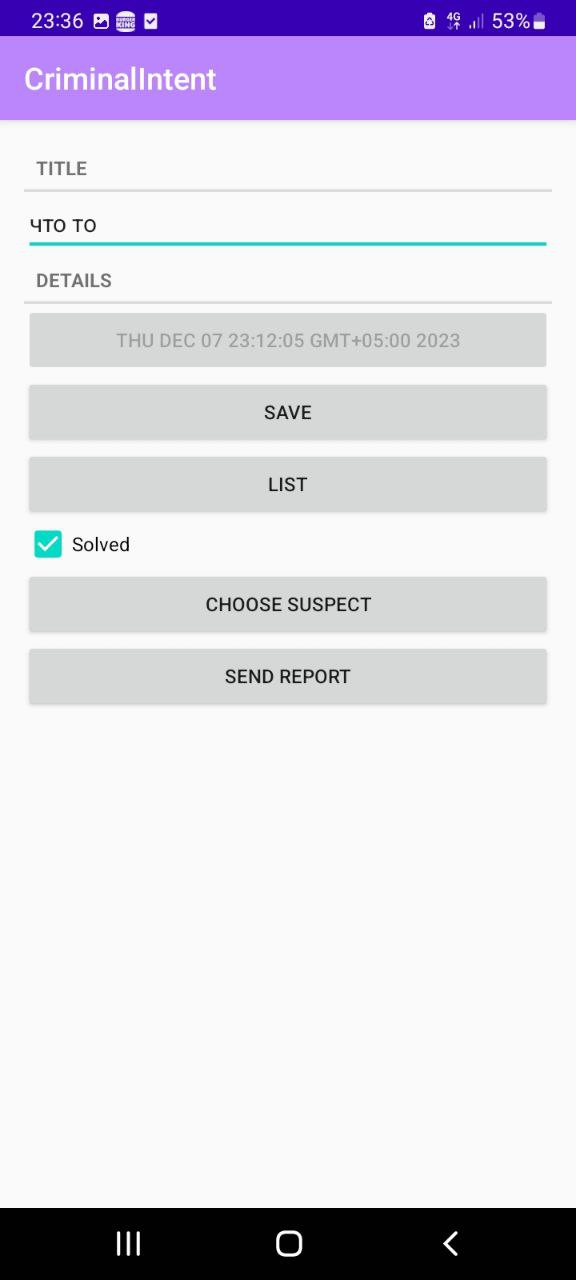
build.gradle:

dependencies **{** //noinspection GradleCompatible  
 implementation 'com.android.support:appcompat-v7:28.0.0'  
 implementation 'com.android.support.constraint:constraint-layout:2.0.4'  
 testImplementation 'junit:junit:4.13.2'  
 androidTestImplementation 'com.android.support.test:runner:1.0.2'  
 androidTestImplementation 'com.android.support.test.espresso:espresso-core:3.0.2'  
 implementation 'androidx.recyclerview:recyclerview:1.0.0'  
 implementation 'android.arch.lifecycle:extensions:1.1.0'  
 implementation 'android.arch.lifecycle:viewmodel:1.1.0'  
 implementation 'androidx.room:room-runtime:2.4.3'  
 kapt 'androidx.room:room-compiler:2.4.3'  
 implementation 'androidx.room:room-ktx:2.4.3'  
 implementation 'androidx.appcompat:appcompat:1.0.0-beta01'  
**}**

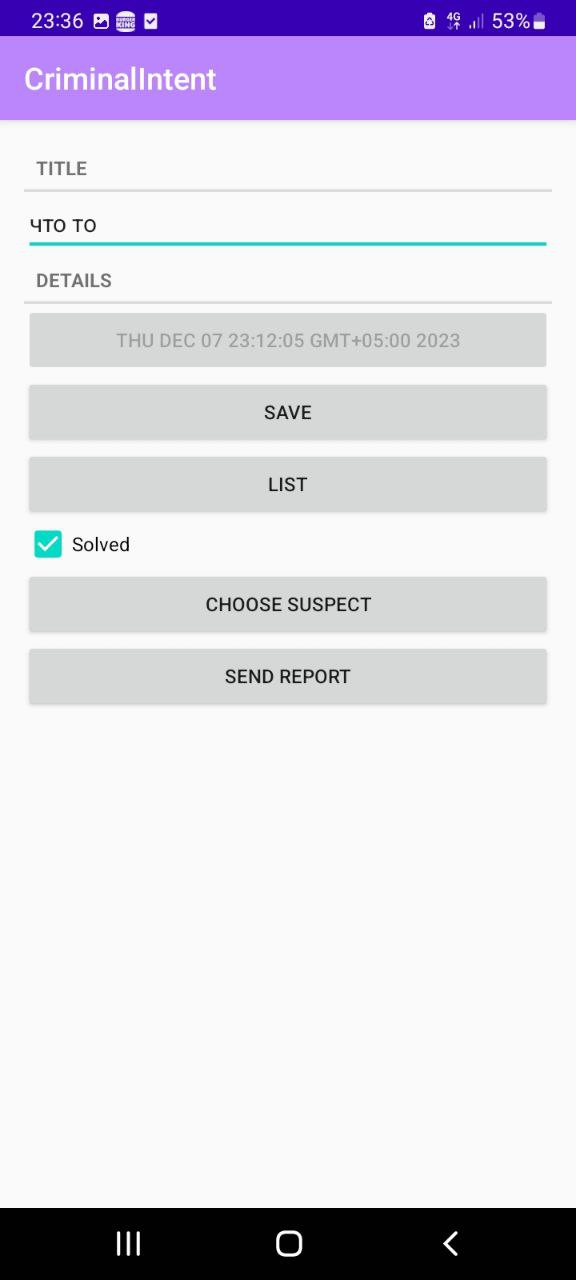
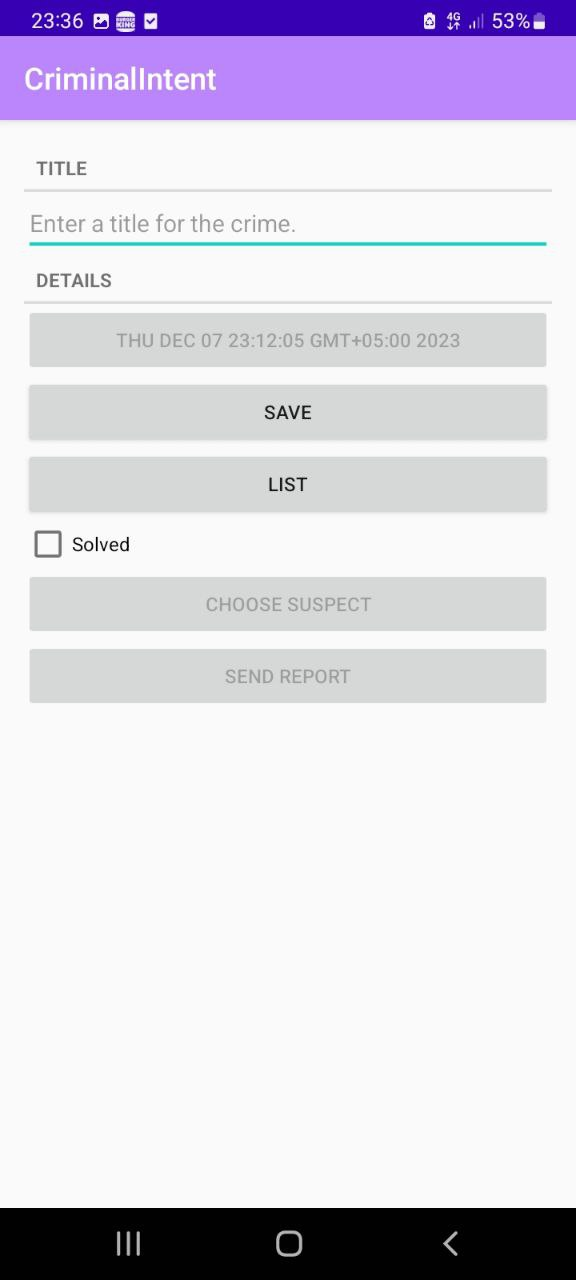
**Проверки:**

Кнопки будут заблокированы пока не будет галочки:



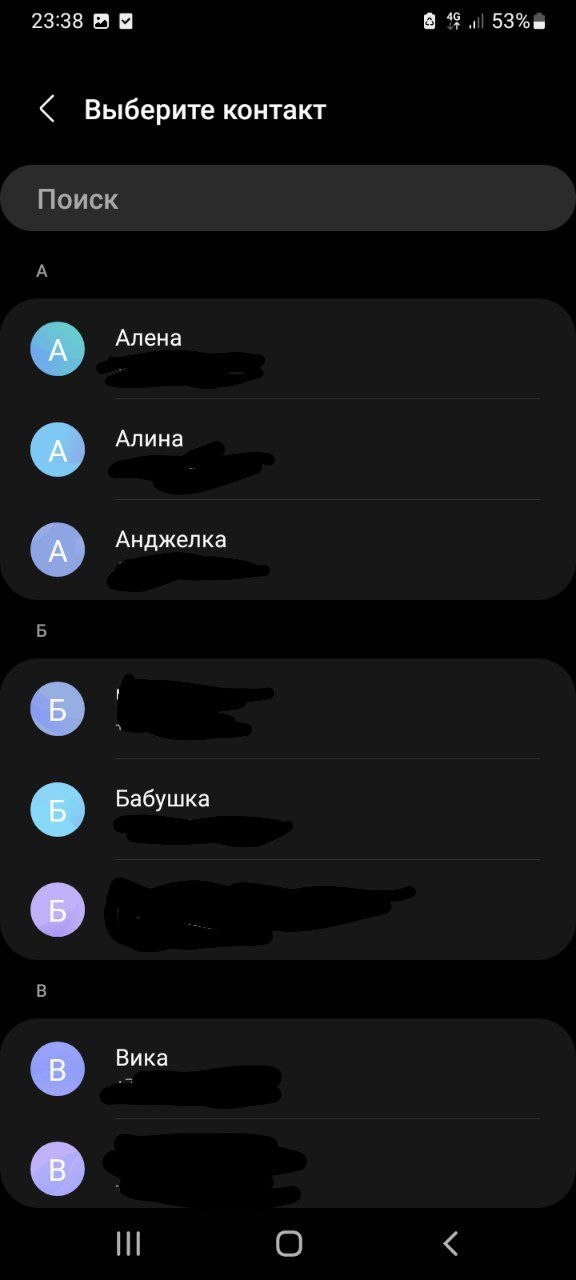


**Экраны:**

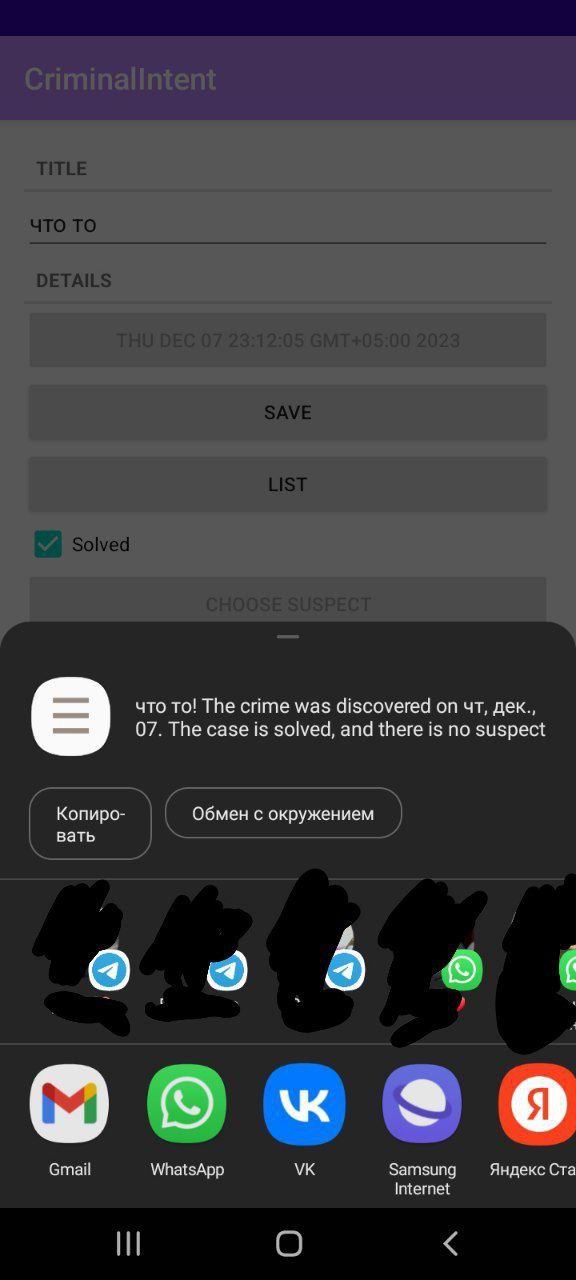
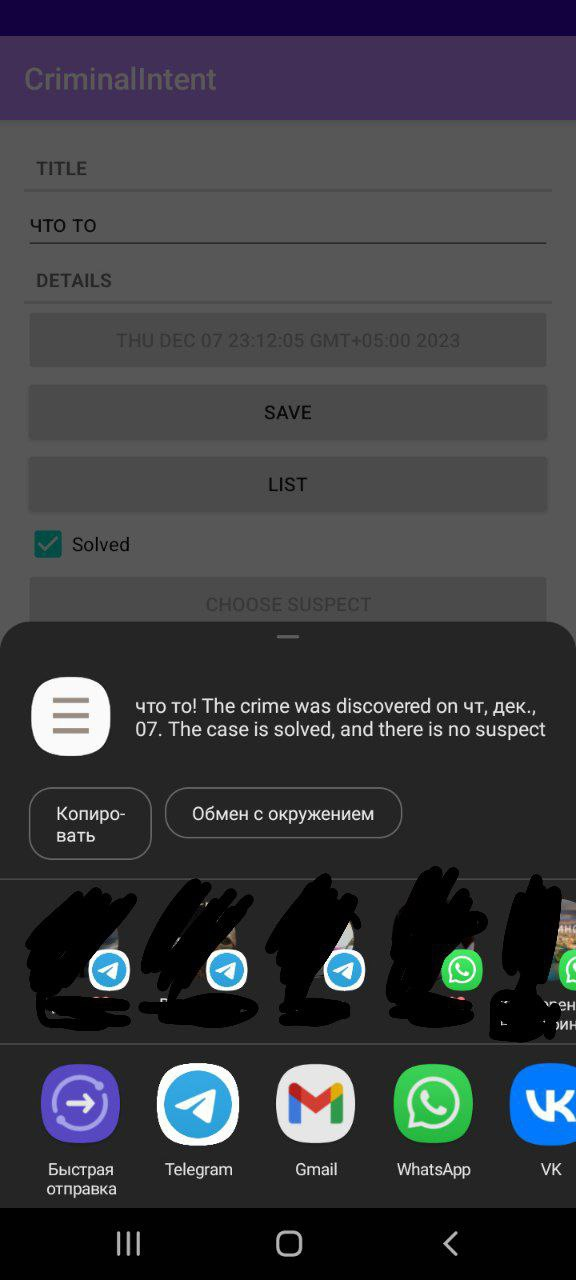


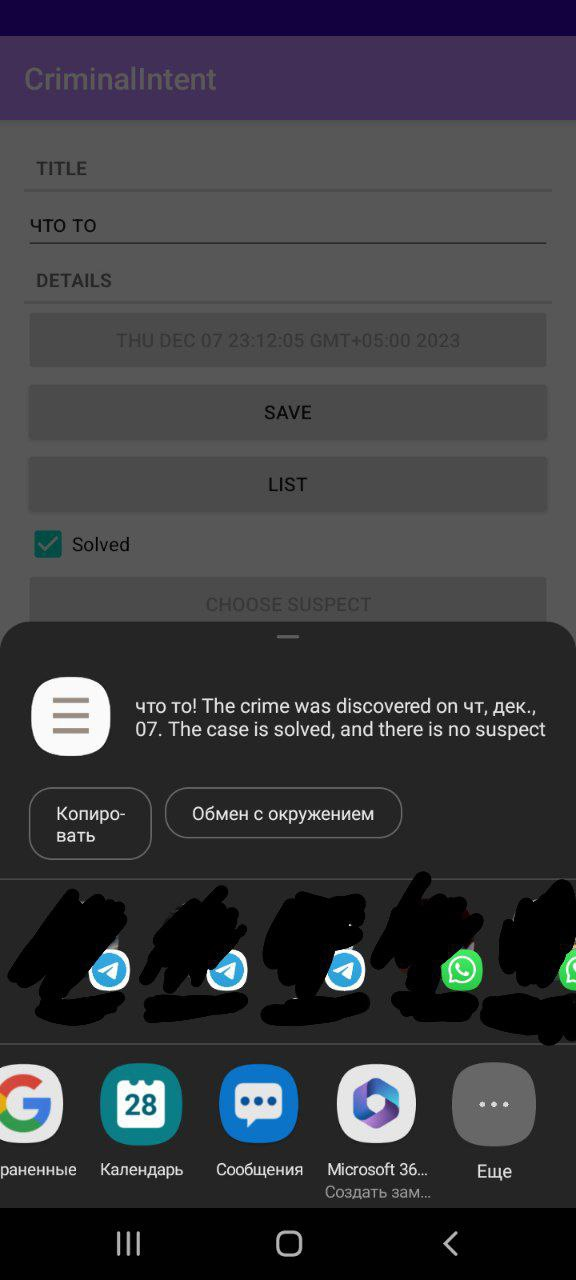


При нажатии на кнопку CHOOSE SUSPECT:



При нажатии на кнопку SEND REPORT:





**Вывод:**Научилась работать с неявными интентами.