# Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

**Национальный исследовательский**

**Томский политехнический университет**

Инженерная школа информационных технологий и робототехники Отделение информационных технологий

Отчет по лабораторной работе №19 по дисциплине

**«Мобильная разработка. Хранение данных и сетевое взаимодействие»**

Разрешения

Выполнил:

Студент группы 8B21 П. А. Жердева

Проверил:

Ст. преп. ОИТ ИШИТР В. А. Дорофеев

# Задание

Разработать приложение для получения списка контактов из телефонной книги пользователя. Для этого необходимо разрешение READ\_CONTACTS.

По нажатию кнопки «Получить контакты» приложение должно проверить наличие разрешения, и, если оно отсутствует – начать процесс его получения:

* Если система считает, что нужно показать пользователю обоснование необходимости разрешения – показать такое обоснование в какой-либо форме.
* Если пользователь отказал в разрешении – показать ему информацию о том, что без разрешения функционал приложения ограничен, и дать возможность перейти в системные настройки приложения.
* Если разрешение получено – считать имена контактов из адресной книги и отобразить их на экране в какой-либо форме.

Экраны приложения могут выглядеть примерно таким образом:

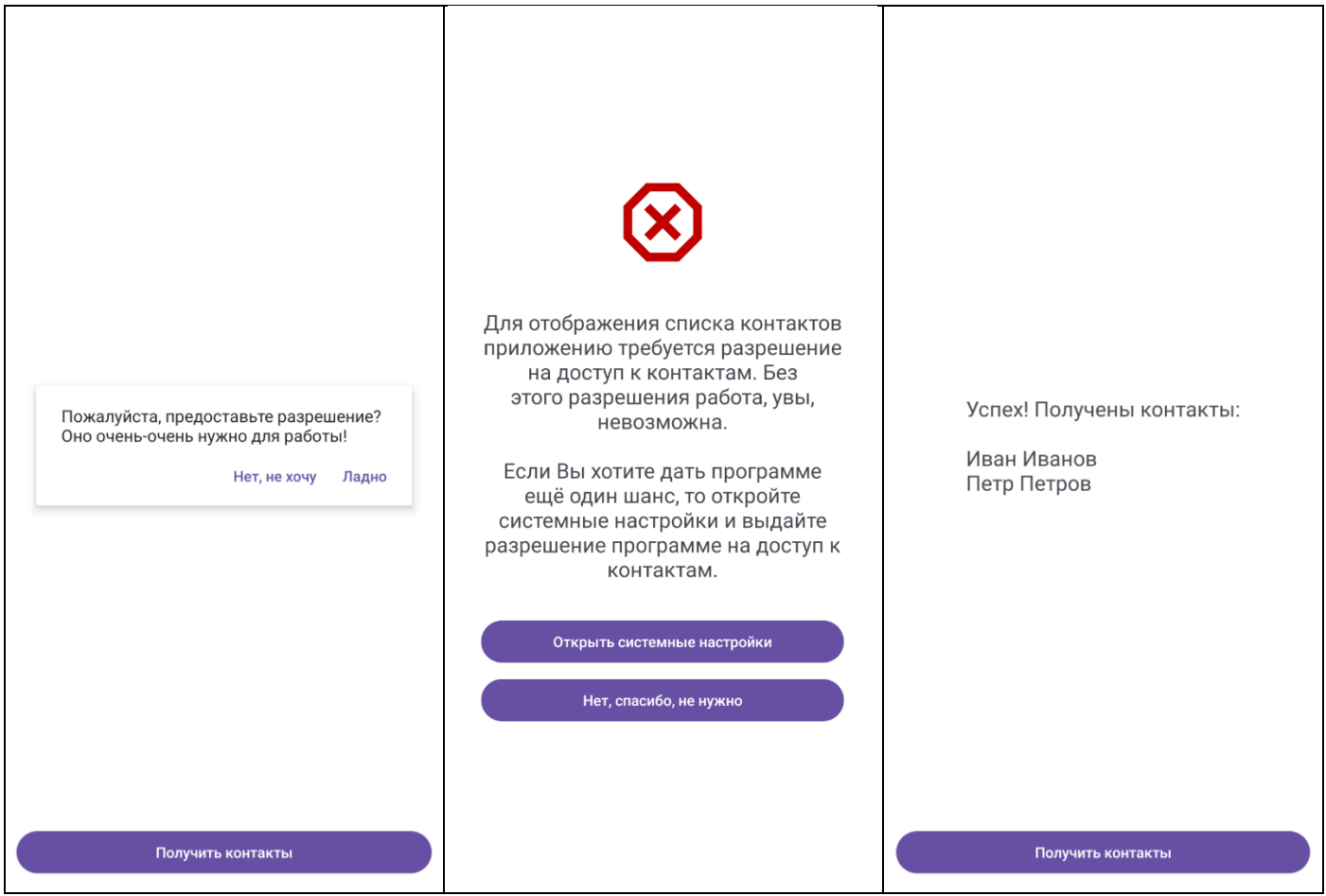


Рис. 1 Примерный вид экранов приложения

Чтение контактов

Для чтения контактов (и многих других операций в Android) используется механизм контентпровайдеров. Он позволяет обмениваться данными между разными приложениями, и при этом не вникать в то, где и как именно эти данные хранятся – в базе данных, в виде локальных файлов или вообще в облаке. Данные запрашиваются при помощи метода query(), и возвращаются в табличном виде, почти как в базе данных – строки с данными, столбцы определённого типа.

Изучение контент-провайдеров выходит за рамки данной темы, поэтому можно просто использовать следующую функцию для получения контактов:

@SuppressLint("Range")

fun getContacts(): List<String> {

val result = mutableListOf<String>()

// Запрос списка контактов

val cur = contentResolver.query(ContactsContract.Contacts.CONTENT\_URI,

null, null, null, null)

if (cur != null) {

// Определение номера столбца, содержащего имя контакта

val colName = cur.getColumnIndex(ContactsContract.Contacts.DISPLAY\_NAME)

while (cur.moveToNext()) {

val name = cur.getString(colName)

result.add(name)

}

cur.close()

}

return result

}

Функция предполагает, что разрешение READ\_CONTACTS уже получено, в противном случае будет выброшено исключение.

**Ход выполнения**

**Как устроено приложение:**

1. Проверяет, есть ли у него разрешение на чтение контактов (READ\_CONTACTS).
2. Если разрешения нет — просит его у пользователя.
3. Если пользователь отказал — показывает экран с предложением открыть настройки.
4. Если разрешение выдано — читает список контактов из телефонной книги и выводит их на экран.

Проект построен по модели MVVM (Model–View–ViewModel):

1. View (представления) - XML-файлы фрагментов и MainActivity.
2. ViewModel - PermissionViewModel.kt — хранит логику проверки и запроса разрешений.
3. Model - Контент-провайдер контактов (ContactsContract), из которого читаются данные.

Для начала в файле AndroidManifest.xml добавляем разрешение для работы с контактами:

<uses-permission android:name="android.permission.READ\_CONTACTS" />

Затем были созданы несколько фрагментов (запрос на получение контактов, фрагмент с запретом получить контакты и фрагмент с контактами), которые отображаются в главной активности приложения.

**Файл MainActivity.kt – главная активность приложения**

Что делает:

1. Отображает стартовый фрагмент RequestPermissionFragment.
2. Создаёт **Activity Result Launcher**, чтобы безопасно запрашивать разрешения.
3. Реагирует на результат — если разрешение выдано, открывает ContactsListFragment, если нет — DeniedPermissionFragment.
4. Позволяет PermissionViewModel обращаться к этому лаунчеру (объект для запроса разрешения у пользователя) (через метод getLauncher()).

Взаимодействие:

* Получает сигнал от PermissionViewModel, что разрешение выдано.
* Меняет фрагменты в контейнере (fragmentContainer).

**Файл PermissionViewModel.kt**

Что делает:

1. Проверяет, выдано ли разрешение (checkPermission()).
2. Если нет — запрашивает через переданный ActivityResultLauncher.
3. Если пользователь отказал, можно открыть системные настройки через openSettings().

Почему это удобно:

* Логика централизована — не нужно дублировать код проверки и запроса в каждом фрагменте.
* ViewModel переживает пересоздание фрагментов (например, при повороте экрана).

**Файл RequestPermissionFragment.kt -** первый экран.

Что делает:

1. Показывает кнопку «Получить контакты».
2. При нажатии:

* Проверяет, есть ли разрешение (viewModel.checkPermission()).
* Запускает запрос (viewModel.requestPermission()).

1. Если разрешение уже есть — MainActivity сразу откроет список контактов.

**Файл DeniedPermissionFragment.kt -** появляется, если пользователь отказал в разрешении.

Что делает**:**

1. Показывает сообщение: без разрешения на доступ к контактам работа приложения невозможна.
2. Кнопка «Открыть системные настройки» вызывает viewModel.openSettings() — открывает страницу приложения в настройках.
3. Кнопка «Нет, спасибо» закрывает приложение.

**Файл ContactsListFragment.kt**

Фрагмент, который показывает контакты.

Что делает:

1. С помощью contentResolver.query() получает список имён контактов.
2. Выводит их в TextView.
3. Если контакты отсутствуют — показывает сообщение “Контакты не найдены”.

**Текст программы**

**ContactsListFragment.kt:**

package ru.polinazherdeva.lr19  
  
import android.annotation.SuppressLint  
import android.os.Bundle  
import android.provider.ContactsContract  
import android.view.LayoutInflater  
import android.view.View  
import android.view.ViewGroup  
import androidx.fragment.app.Fragment  
import ru.polinazherdeva.lr19.databinding.FragmentContactsListBinding  
  
class ContactsListFragment : Fragment() {  
  
 private lateinit var binding: FragmentContactsListBinding *// Переменная для доступа к элементам интерфейса через View Binding* override fun onCreateView(  
 inflater: LayoutInflater, *// объект для раздувания (создания) разметки из XML* container: ViewGroup?, *// контейнер, в который будет вставлен фрагмент* savedInstanceState: Bundle?  
 ): View {  
 *// Создаём binding, который связывает XML с Kotlin-кодом* binding = FragmentContactsListBinding.inflate(inflater, container, false)  
 showContacts()  
 return binding.*root // Возвращаем корневое представление фрагмента (для отображения на экране)* }  
  
 @SuppressLint("Range") *// Подавление предупреждения* private fun showContacts() {  
 val contacts = *mutableListOf*<String>() *// Список для имен контактов* val resolver = requireContext().*contentResolver // Получаем contentResolver - через него можно обращаться к контактам  
  
 // Делаем запрос ко всем контактам в телефоне* resolver.query(  
 ContactsContract.Contacts.*CONTENT\_URI*, *// URI всех контактов* null, *// какие столбцы брать (null = все)* null, null, null *// условия выборки и сортировки* )?.*use* **{** cursor **->** *// Получаем индекс столбца, где хранится имя контакта* val nameColumn = cursor.getColumnIndex(ContactsContract.Contacts.*DISPLAY\_NAME*)  
 *// Проходим по всем строкам результата* while (cursor.moveToNext()) {  
 val name = cursor.getString(nameColumn) *// Извлекаем имя контакта из текущей строки* contacts.add(name) *// Добавляем имя в список* }  
 **}** *// Определяем, какой текст показать пользователю* val text = if (contacts.isEmpty()) {  
 getString(R.string.*contacts\_not\_found*) *// Если контактов нет* } else {  
 *// Если контакты есть — добавляем заголовок и список имён* "${getString(R.string.*contacts\_success*)}\n\n${contacts.*joinToString*("\n")}"  
 }  
  
 *// Устанавливаем готовый текст в TextView на экране* binding.tvContacts.*text* = text  
 }  
}

**DeniedPermissionFragment.kt:**

package ru.polinazherdeva.lr19  
  
import android.os.Bundle  
import android.view.LayoutInflater  
import android.view.View  
import android.view.ViewGroup  
import androidx.fragment.app.Fragment  
import androidx.fragment.app.activityViewModels  
import ru.polinazherdeva.lr19.databinding.FragmentDeniedPermissionBinding  
  
class DeniedPermissionFragment : Fragment() {  
  
 *// View Binding для доступа к элементам интерфейса* private lateinit var binding: FragmentDeniedPermissionBinding  
  
 *// Получаем ViewModel, общую для Activity и всех фрагментов* private val viewModel: PermissionViewModel by *activityViewModels*()  
  
 override fun onCreateView(  
 inflater: LayoutInflater, container: ViewGroup?,  
 savedInstanceState: Bundle?  
 ): View {  
 binding = FragmentDeniedPermissionBinding.inflate(inflater, container, false)  
  
 *// Обработчик нажатия на кнопку «Открыть настройки»* binding.btnOpenSettings.setOnClickListener **{** viewModel.openSettings() *// Открытие системных настроек приложения* **}** *// Обработчик нажатия на кнопку «Нет, спасибо»* binding.btnNoThanks.setOnClickListener **{** requireActivity().finish() *// Закрываем Activity и выходим из приложения* **}** return binding.*root // Возвращаем корневое представление фрагмента* }  
}

**MainActivity.kt:**

package ru.polinazherdeva.lr19  
  
import android.os.Bundle  
import androidx.activity.result.contract.ActivityResultContracts  
import androidx.activity.viewModels  
import androidx.appcompat.app.AppCompatActivity  
import androidx.fragment.app.commit  
import ru.polinazherdeva.lr19.databinding.ActivityMainBinding  
  
class MainActivity : AppCompatActivity() {  
  
 private lateinit var binding: ActivityMainBinding *// View Binding для доступа к элементам activity\_main.xml* private val viewModel: PermissionViewModel by *viewModels*() *// ViewModel, которая управляет разрешениями  
  
 // Регистрация "лаунчера" — объекта для запроса разрешения у пользователя* private val permissionLauncher =  
 *// // Вызывается, когда пользователь ответит на запрос разрешения* registerForActivityResult(ActivityResultContracts.RequestPermission()) **{** isGranted **->** if (isGranted) {  
 openContactsFragment() *// Если пользователь разрешил — открываем фрагмент с контактами* } else {  
 openDeniedPermissionFragment() *// Если отказал — открываем фрагмент с уведомлением об отказе* }  
 **}** override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 binding = ActivityMainBinding.inflate(*layoutInflater*)  
 setContentView(binding.*root*)  
  
 *// Загружаем первый фрагмент — запрос разрешения  
 supportFragmentManager*.*commit* **{** replace(R.id.*fragmentContainer*, RequestPermissionFragment())  
 **}** *// Подписываемся на изменение значения LiveData (из ViewModel)  
 // Когда разрешение будет выдано, автоматически откроется ContactsListFragment* viewModel.permissionGranted.observe(this) **{** granted **->** if (granted) openContactsFragment()  
 **}** }  
  
 *// Функция для открытия фрагмента со списком контактов* fun openContactsFragment() {  
 *supportFragmentManager*.*commit* **{** replace(R.id.*fragmentContainer*, ContactsListFragment())  
 **}** }  
  
 *// Функция для открытия фрагмента, если пользователь отказал в разрешении* fun openDeniedPermissionFragment() {  
 *supportFragmentManager*.*commit* **{** replace(R.id.*fragmentContainer*, DeniedPermissionFragment())  
 **}** }  
  
 *// Функция для передачи лаунчера в другие фрагменты (например, RequestPermissionFragment)* fun getLauncher() = permissionLauncher  
}

**PermissionViewModel.kt:**

package ru.polinazherdeva.lr19  
  
import android.Manifest  
import android.app.Application  
import android.content.Intent  
import android.content.pm.PackageManager  
import android.net.Uri  
import android.provider.Settings  
import androidx.activity.result.ActivityResultLauncher  
import androidx.core.content.ContextCompat  
import androidx.lifecycle.AndroidViewModel  
import androidx.lifecycle.MutableLiveData  
  
class PermissionViewModel(app: Application) : AndroidViewModel(app) {  
  
 *// LiveData, хранящая состояние: выдано ли разрешение (true/false)  
 // Активность и фрагменты могут наблюдать за этим значением* val permissionGranted = MutableLiveData<Boolean>()  
 *// Получаем контекст приложения, чтобы можно было вызывать системные функции* private val context = app.*applicationContext  
  
 // Проверяем, есть ли у приложения разрешение на чтение контактов* fun checkPermission() {  
 val granted = ContextCompat.checkSelfPermission(  
 context, Manifest.permission.*READ\_CONTACTS* ) == PackageManager.*PERMISSION\_GRANTED* permissionGranted.*value* = granted *// Обновляем LiveData* }  
  
 *// Метод для запроса разрешения через специальный launcher  
 // launcher передается из Activity* fun requestPermission(launcher: ActivityResultLauncher<String>) {  
 launcher.launch(Manifest.permission.*READ\_CONTACTS*) *// Запускаем системное окно с просьбой разрешить доступ к контактам* }  
  
 *// Метод открывает настройки приложения, если пользователь запретил разрешение* fun openSettings() {  
 val intent = Intent(  
 Settings.*ACTION\_APPLICATION\_DETAILS\_SETTINGS*,  
 *// Открываем страницу настроек конкретного приложения* Uri.parse("package:${context.*packageName*}") *// Адрес с именем пакета текущего приложения* ).*apply* **{** *flags* = Intent.*FLAG\_ACTIVITY\_NEW\_TASK // Добавляем флаг, чтобы запуск происходил вне текущего стека (нужно для контекста приложения)* **}** *// Запускаем активность настроек* context.startActivity(intent)  
 }  
}

**RequestPermissionFragment.kt:**

package ru.polinazherdeva.lr19  
  
import android.os.Bundle  
import android.view.LayoutInflater  
import android.view.View  
import android.view.ViewGroup  
import androidx.fragment.app.Fragment  
import androidx.fragment.app.activityViewModels  
import ru.polinazherdeva.lr19.databinding.FragmentRequestPermissionBinding  
  
class RequestPermissionFragment : Fragment() {  
  
 *// ViewBinding для доступа к элементам интерфейса из XML* private lateinit var binding: FragmentRequestPermissionBinding  
 *// Получаем ViewModel, общую с активностью (MainActivity),  
 // чтобы можно было вызывать методы проверки и запроса разрешения* private val viewModel: PermissionViewModel by *activityViewModels*()  
  
 override fun onCreateView(  
 inflater: LayoutInflater, container: ViewGroup?,  
 savedInstanceState: Bundle?  
 ): View {  
 *// Создаем объект binding, чтобы связать XML с кодом* binding = FragmentRequestPermissionBinding.inflate(inflater, container, false)  
 *// Обработчик нажатия кнопки "Получить контакты"* binding.btnGetContacts.setOnClickListener **{** viewModel.checkPermission() *// Проверка на наличие разрешения  
 // Получаем ссылку на MainActivity, чтобы вызвать её launcher* val activity = requireActivity() as MainActivity  
 *// Запускаем запрос разрешения на чтение контактов* viewModel.requestPermission(activity.getLauncher())  
 **}** return binding.*root* }  
}

**Activity\_main.xml:**

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

**Fragment\_contacts\_list.xml:**

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 android:orientation="vertical"  
 android:gravity="center\_horizontal"  
 android:padding="24dp"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent">  
  
 <TextView  
 android:id="@+id/tvContacts"  
 android:text="@string/contacts\_title"  
 android:textSize="16sp"  
 android:layout\_marginBottom="24dp"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content" />  
</LinearLayout>

**Fragment\_denied\_permission.xml:**

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 android:orientation="vertical"  
 android:gravity="center\_horizontal"  
 android:padding="24dp"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent">  
  
 <ImageView  
 android:src="@android:drawable/ic\_delete"  
 android:layout\_width="48dp"  
 android:layout\_height="48dp"  
 android:layout\_marginBottom="16dp" />  
  
 <TextView  
 android:text="@string/permission\_denied\_message"  
 android:textSize="16sp"  
 android:textAlignment="center"  
 android:layout\_marginBottom="24dp"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content" />  
  
 <Button  
 android:id="@+id/btnOpenSettings"  
 android:text="@string/button\_open\_settings"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content" />  
  
 <Button  
 android:id="@+id/btnNoThanks"  
 android:text="@string/button\_no\_thanks"  
 android:layout\_marginTop="8dp"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content" />  
</LinearLayout>

**Fragment\_request\_permission.xml:**

*<?*xml version="1.0" encoding="utf-8"*?>*<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:orientation="vertical"  
 android:gravity="center"  
 android:padding="24dp">  
  
 <Button  
 android:id="@+id/btnGetContacts"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="@string/button\_get\_contacts" />  
</LinearLayout>

**Strings.xml:**

<resources>  
 <string name="app\_name">lr19</string>  
  
 *<!-- RequestPermissionFragment -->* <string name="permission\_request\_message">  
 Пожалуйста, предоставьте разрешение на чтение контактов.  
 </string>  
 <string name="button\_get\_contacts">Получить контакты</string>  
  
 *<!-- DeniedPermissionFragment -->* <string name="permission\_denied\_message">  
 Без разрешения на доступ к контактам работа приложения невозможна.  
 </string>  
 <string name="button\_open\_settings">Открыть системные настройки</string>  
 <string name="button\_no\_thanks">Нет, спасибо</string>  
  
 *<!-- ContactsListFragment -->* <string name="contacts\_not\_found">Контакты не найдены</string>  
 <string name="contacts\_success">Успех! Получены контакты:</string>  
   
 <string name="contacts\_title">Контакты</string>  
</resources>

**Результат работы:**

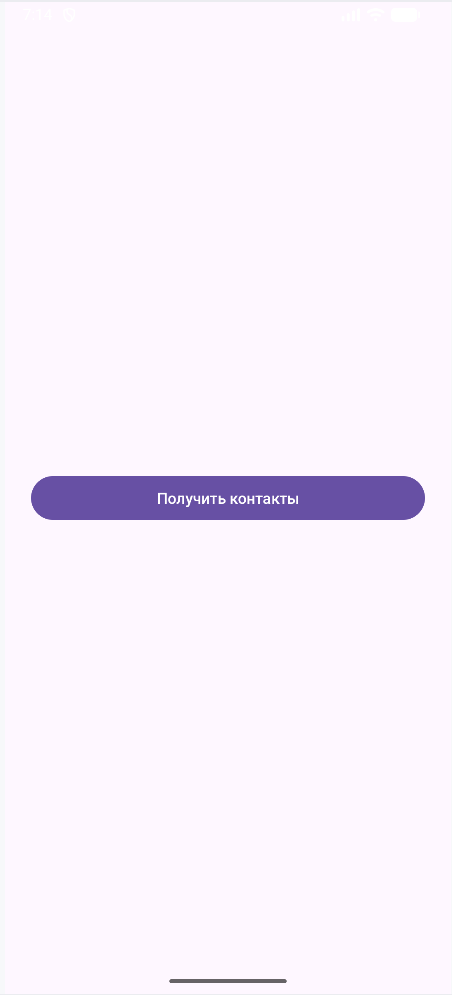


Рис. 2. Стартовый экран с кнопкой «Получить контакты»

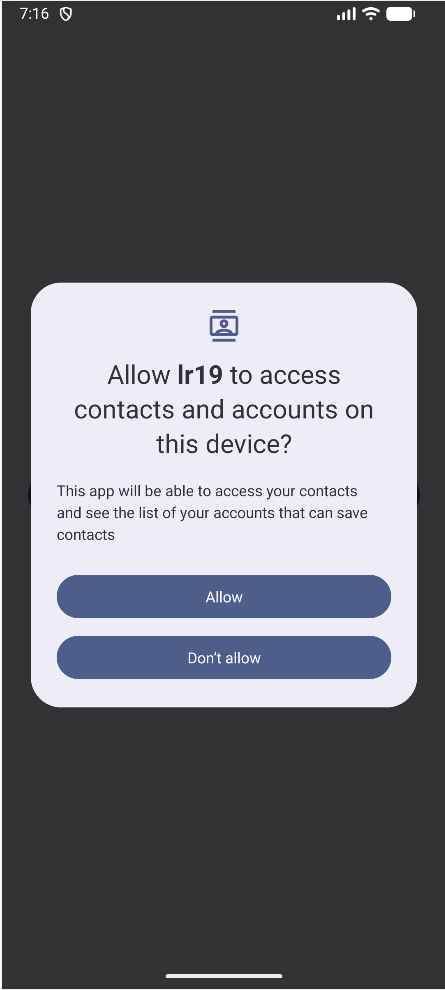


Рис. 3. Запрос разрешения доступа к контактам пользователя

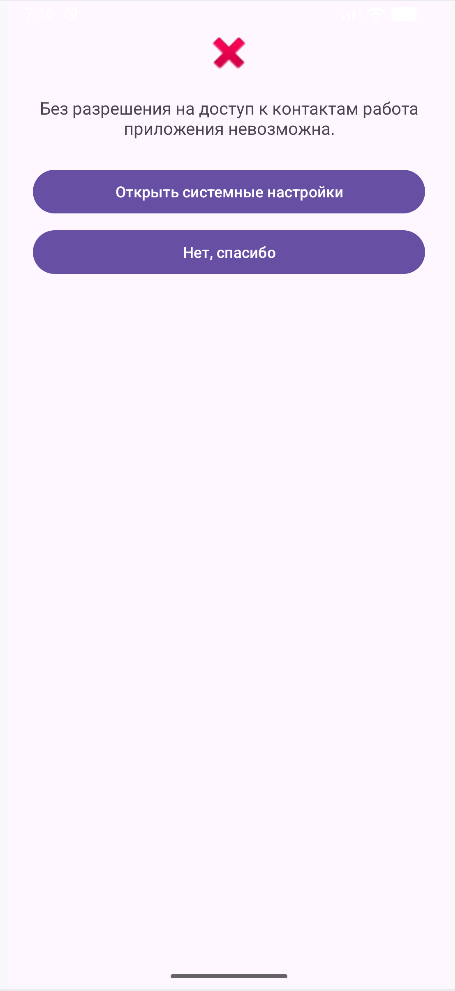


Рис. 4. Фрагмент, появляющийся при отказе пользователя в доступе



Рис. 5. Фрагмент с контактами при разрешении доступа

# Выводы

В ходе лабораторной работы было разработано Android-приложение, которое получает список контактов пользователя с использованием разрешения READ\_CONTACTS.

Реализована проверка наличия разрешения, его запрос во время выполнения и обработка возможных сценариев:

— получение разрешения и вывод контактов,

— отказ пользователя с предложением открыть настройки.

Для реализации использованы ViewModel, LiveData, ContentResolver и ActivityResultContracts.RequestPermission.