

|  |
| --- |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждение высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** |

Институт Информационных технологий

Кафедра Математического обеспечения и стандартизации информационных технологий

Отчет по практической работе №2

по дисциплине «Разработка клиент-серверных мобильных приложений»

|  |  |
| --- | --- |
| **Выполнил:**  Студент группы ИКБО-28-22 | Некрасов Глеб Андреевич |
| **Проверила:** | Дворникова Екатерина Михайловна |

МОСКВА 2025

СОДЕРЖАНИЕ

[Практическая работа №2 3](#_Toc192622038)

[Выполнение практической работы №2 5](#_Toc192622039)

[Тестирование практической работы 11](#_Toc192622040)

[Вывод 18](#_Toc192622041)

[Список использованной литературы 19](#_Toc192622042)

# Практическая работа №2

Продолжаем работу над разработкой мобильного приложения из практической работы №1, а именно поисковым экраном (поисковой строкой).

**Задание 1**

На поисковой строке реализовать кнопку «Очистить». Обязательные условия:

1. В поле ввода поискового запроса отображается подсказка, если поле ввода пустое.

2. При нажатии на поле ввода появляется клавиатура.

3. Если в поле ввода поискового запроса введён какой-то текст, то отображается кнопка «Очистить».

4. Если в поле ввода поискового запроса нет никакого текста, то кнопка «Очистить» не отображается.

5. Нажатие на кнопку «Очистить» удаляет текст в поле ввода поискового запроса и скрывает клавиатуру.

**Задание 2**

Реализовать сохранение текста поискового запроса в жизненном цикле, то есть если в поле ввода поискового запроса есть какой-то текст, то при повороте устройства в ландшафтную ориентацию и обратно текст в поле ввода поискового запроса не исчезает.

**Задание 3**

Если в приложении для курсовой работы не предполагается подключение API, необходимо подготовить «приложение-заглушку» для демонстрации работы API.

**Задание 4**

Подключить к странице поиска любую открытую API. Ссылку на список открытых API можно найти в СДО. Реализовать корректный поиск согласно теме проекта.

**Задание 5**

Реализовать плейсхолдеры:

1. Если нет результатов поиска.

2. Если поисковый запрос завершился неудачно, при этом на данном плейсхолдере должна быть размещена кнопка «Обновить».

**Задание 6**

Если на плейсхолдере присутствует кнопка «Обновить», то необходимо реализовать повторное отправление последнего запроса при нажатии на нее.

# Выполнение практической работы №2

Дополним нашу разметку activity\_search.xml (см. Листинг 1)

*Листинг 1. activity\_search.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="#222222">   <androidx.cardview.widget.CardView  android:id="@+id/cardView"  android:layout\_width="0dp"  android:layout\_height="0dp"  app:cardCornerRadius="0dp"  app:layout\_constraintBottom\_toBottomOf="parent"  app:layout\_constraintEnd\_toEndOf="parent"  app:layout\_constraintStart\_toStartOf="parent"  app:layout\_constraintTop\_toTopOf="parent">   <androidx.constraintlayout.widget.ConstraintLayout  android:layout\_width="match\_parent"  android:layout\_height="match\_parent"  android:background="#FFFFFF">   <!-- Заголовок поиска -->  <TextView  android:id="@+id/textViewSearchTitle"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:text="Поиск путёвок LuxuryLife"  android:textColor="@android:color/black"  android:textSize="20sp"  app:layout\_constraintTop\_toTopOf="parent"  app:layout\_constraintStart\_toStartOf="parent"  app:layout\_constraintEnd\_toEndOf="parent"  android:layout\_marginTop="32dp"/>   <!-- Поле ввода для поиска -->  <EditText  android:id="@+id/editTextSearch"  android:layout\_width="0dp"  android:layout\_height="wrap\_content"  android:background="@drawable/edit\_text\_background"  android:hint="Введите запрос"  app:layout\_constraintTop\_toBottomOf="@+id/textViewSearchTitle"  app:layout\_constraintStart\_toStartOf="parent"  app:layout\_constraintEnd\_toStartOf="@id/buttonClear"  android:layout\_marginStart="32dp"  android:layout\_marginEnd="8dp"  android:layout\_marginTop="16dp"/>   <!-- Кнопка очистки поля поиска -->  <Button  android:id="@+id/buttonClear"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:text="✖"  android:textSize="18sp"  android:background="@drawable/registration\_button\_background"  android:visibility="gone"  app:layout\_constraintTop\_toTopOf="@id/editTextSearch"  app:layout\_constraintBottom\_toBottomOf="@id/editTextSearch"  app:layout\_constraintEnd\_toEndOf="parent"  android:layout\_marginEnd="32dp"/>   <!-- Плейсхолдер для пустых результатов поиска -->  <TextView  android:id="@+id/textViewNoResults"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:text="Нет результатов поиска"  android:textColor="@android:color/darker\_gray"  android:textSize="18sp"  app:layout\_constraintTop\_toBottomOf="@id/editTextSearch"  app:layout\_constraintStart\_toStartOf="parent"  app:layout\_constraintEnd\_toEndOf="parent"  android:visibility="gone"  android:layout\_marginTop="16dp"/>   <!-- Плейсхолдер для ошибки запроса с кнопкой "Обновить" -->  <TextView  android:id="@+id/textViewError"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:text="Произошла ошибка при запросе"  android:textColor="@android:color/holo\_red\_dark"  android:textSize="18sp"  android:visibility="gone"  app:layout\_constraintTop\_toBottomOf="@id/textViewNoResults"  app:layout\_constraintStart\_toStartOf="parent"  app:layout\_constraintEnd\_toEndOf="parent"  android:layout\_marginTop="16dp"/>   <!-- Кнопка для обновления запроса -->  <Button  android:id="@+id/buttonRetry"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:text="Обновить"  android:visibility="gone"  app:layout\_constraintTop\_toBottomOf="@id/textViewError"  app:layout\_constraintStart\_toStartOf="parent"  app:layout\_constraintEnd\_toEndOf="parent"  android:layout\_marginTop="16dp"  android:background="@drawable/registration\_button\_background"/>   <!-- Кнопка "Назад" -->  <Button  android:id="@+id/buttonBack"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:text="Назад"  android:background="@drawable/registration\_button\_background"  app:layout\_constraintBottom\_toBottomOf="parent"  app:layout\_constraintStart\_toStartOf="parent"  android:layout\_marginBottom="16dp"  android:layout\_marginStart="16dp"/>   <!-- Кнопка "Профиль" -->  <Button  android:id="@+id/buttonProfile"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:text="Профиль"  android:background="@drawable/registration\_button\_background"  app:layout\_constraintBottom\_toBottomOf="parent"  app:layout\_constraintEnd\_toEndOf="parent"  android:layout\_marginBottom="16dp"  android:layout\_marginEnd="16dp"/>   <!-- Результаты поиска -->  <TextView  android:id="@+id/textViewResults"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:text=""  android:textColor="@android:color/black"  android:textSize="16sp"  app:layout\_constraintTop\_toBottomOf="@id/editTextSearch"  app:layout\_constraintStart\_toStartOf="parent"  app:layout\_constraintEnd\_toEndOf="parent"  android:layout\_marginTop="16dp"/>   </androidx.constraintlayout.widget.ConstraintLayout>   </androidx.cardview.widget.CardView>  </androidx.constraintlayout.widget.ConstraintLayout> |

Также для выполнения нашей практической, надо добавить код SearchActivity.kt (см. Листинг 2)

*Листинг 2. SearchActivity.kt*

|  |
| --- |
| package com.example.kp\_luxurylife\_part2  import android.os.Bundle import android.view.KeyEvent import android.view.View import android.view.inputmethod.EditorInfo import android.view.inputmethod.InputMethodManager import android.widget.Toast import androidx.appcompat.app.AppCompatActivity import androidx.core.widget.addTextChangedListener import com.example.kp\_luxurylife\_part2.databinding.ActivitySearchBinding import kotlinx.coroutines.CoroutineScope import kotlinx.coroutines.Dispatchers import kotlinx.coroutines.launch import kotlinx.coroutines.withContext  class SearchActivity : AppCompatActivity() {   private lateinit var binding: ActivitySearchBinding  private var lastSearchQuery: String? = null   override fun onCreate(savedInstanceState: Bundle?) {  super.onCreate(savedInstanceState)  binding = ActivitySearchBinding.inflate(layoutInflater)  setContentView(binding.root)   savedInstanceState?.let {  lastSearchQuery = it.getString("LAST\_SEARCH\_QUERY")  binding.editTextSearch.setText(lastSearchQuery)  }   setupSearchField()  setupButtons()  }   private fun setupSearchField() {  binding.editTextSearch.hint = "Введите запрос"   binding.editTextSearch.setOnFocusChangeListener { \_, hasFocus ->  binding.buttonClear.visibility = if (hasFocus && !binding.editTextSearch.text.isNullOrEmpty()) View.VISIBLE else View.GONE  }   binding.editTextSearch.addTextChangedListener { text ->  binding.buttonClear.visibility = if (!text.isNullOrEmpty()) View.VISIBLE else View.GONE  }   binding.editTextSearch.setOnEditorActionListener { \_, actionId, event ->  if (actionId == EditorInfo.IME\_ACTION\_SEARCH ||  (event?.keyCode == KeyEvent.KEYCODE\_ENTER && event.action == KeyEvent.ACTION\_DOWN)) {  performSearch(binding.editTextSearch.text.toString())  true  } else {  false  }  }  }   private fun setupButtons() {  binding.buttonClear.setOnClickListener {  binding.editTextSearch.text.clear()  binding.buttonClear.visibility = View.GONE  hideKeyboard()  }   binding.buttonBack.setOnClickListener {  finish()  }   binding.buttonRetry.setOnClickListener {  lastSearchQuery?.let {  Toast.makeText(this, "Повторная попытка запроса...", Toast.LENGTH\_SHORT).show()  binding.editTextSearch.onEditorAction(EditorInfo.IME\_ACTION\_SEARCH)  } ?: Toast.makeText(this, "Нет запроса для обновления", Toast.LENGTH\_SHORT).show()  }  }   private fun performSearch(query: String) {  if (query.isNotEmpty()) {  lastSearchQuery = query   // Если введено слово "ошибка", имитируем ошибку  if (query.equals("ошибка", ignoreCase = true)) {  showErrorPlaceholder()  return  }   CoroutineScope(Dispatchers.IO).launch {  try {  val response = FakeApiService.searchDestinations(query)  withContext(Dispatchers.Main) {  if (response.isNotEmpty()) {  displayResults(response)  } else {  showNoResultsPlaceholder()  }  }  } catch (e: Exception) {  withContext(Dispatchers.Main) {  showErrorPlaceholder()  Toast.makeText(this@SearchActivity, "Ошибка при выполнении запроса", Toast.LENGTH\_SHORT).show()  }  }  }  }  }   private fun displayResults(destinations: List<Destination>) {  val resultText = destinations.joinToString("\n") { "${it.name} (${it.country}): ${it.description}" }  binding.textViewResults.text = resultText  binding.textViewResults.visibility = View.VISIBLE   binding.textViewNoResults.visibility = View.GONE  binding.textViewError.visibility = View.GONE  binding.buttonRetry.visibility = View.GONE  }   private fun showNoResultsPlaceholder() {  binding.textViewResults.visibility = View.GONE  binding.textViewNoResults.visibility = View.VISIBLE  binding.textViewError.visibility = View.GONE  binding.buttonRetry.visibility = View.GONE  }   private fun showErrorPlaceholder() {  binding.textViewResults.visibility = View.GONE  binding.textViewNoResults.visibility = View.GONE  binding.textViewError.visibility = View.VISIBLE  binding.buttonRetry.visibility = View.VISIBLE  }   private fun hideKeyboard() {  val imm = getSystemService(INPUT\_METHOD\_SERVICE) as InputMethodManager  imm.hideSoftInputFromWindow(binding.editTextSearch.windowToken, 0)  }   override fun onSaveInstanceState(outState: Bundle) {  super.onSaveInstanceState(outState)  outState.putString("LAST\_SEARCH\_QUERY", lastSearchQuery)  } } |

Добавим класс Destination для хранения данных в определённом формате (см. Листинг 3)

*Листинг 3. Destination.kt*

|  |
| --- |
| package com.example.kp\_luxurylife\_part2  data class Destination(  val name: String,  val country: String,  val description: String ) |

Добавим FakeApiService (см. Листинг 4)

*Листинг 4. FakeApiService.kt*

|  |
| --- |
| package com.example.kp\_luxurylife\_part2  import kotlinx.coroutines.delay  object FakeApiService {   private val fakeDestinations = listOf(  Destination("Москва", "Россия", "Красная площадь, Кремль, музеи"),  Destination("Париж", "Франция", "Эйфелева башня, Лувр, Нотр-Дам"),  Destination("Нью-Йорк", "США", "Таймс-сквер, Центральный парк, Статуя Свободы"),  Destination("Лондон", "Великобритания", "Биг-Бен, Букингемский дворец, Тауэр"),  Destination("Дубай", "ОАЭ", "Бурдж-Халифа, Дубай-Молл, Пальма Джумейра"),  Destination("Токио", "Япония", "Синдзюку, Сибуя, храмы, аниме-культура"),  Destination("Берлин", "Германия", "Бранденбургские ворота, Берлинская стена, музеи"),  Destination("Рим", "Италия", "Колизей, Пантеон, Ватикан"),  Destination("Барселона", "Испания", "Саграда Фамилия, парк Гуэля, пляжи"),  Destination("Амстердам", "Нидерланды", "Каналы, музеи, велосипеды"),  )   suspend fun searchDestinations(query: String): List<Destination> {  delay(2000) // Эмуляция сетевого запроса (2 секунда)  return fakeDestinations.filter {  it.name.contains(query, ignoreCase = true) ||  it.country.contains(query, ignoreCase = true) ||  it.description.contains(query, ignoreCase = true)  }  } } |

# Тестирование практической работы

Протестируем наше приложение (см. Рисунки 1-10)

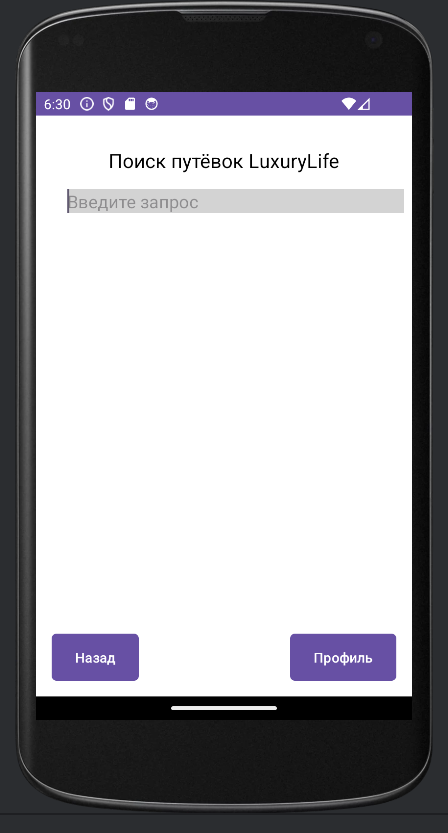


Рисунок 1. Запуск приложения

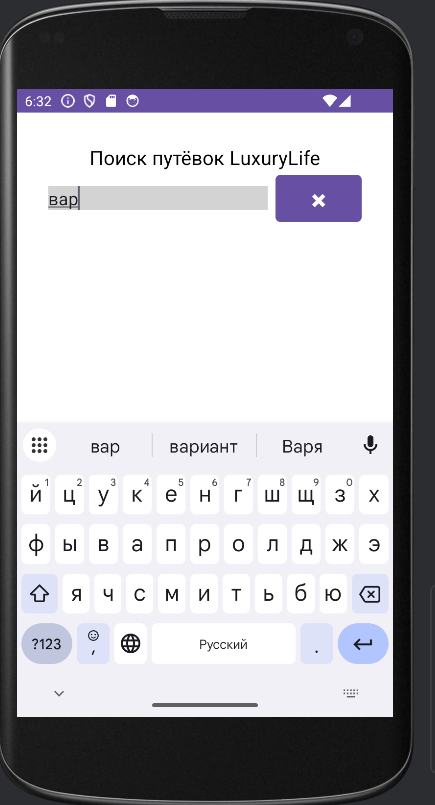


Рисунок 2. Ввод запроса

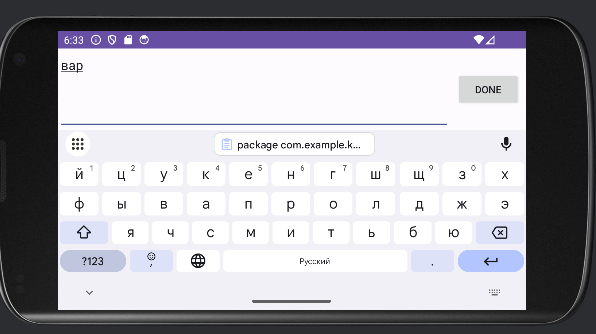


Рисунок 3. Перевернули экран

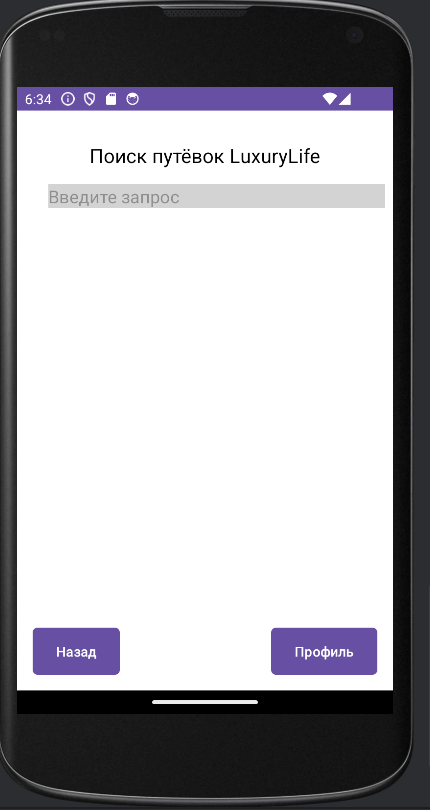


Рисунок 4. Нажали на кнопку очистить

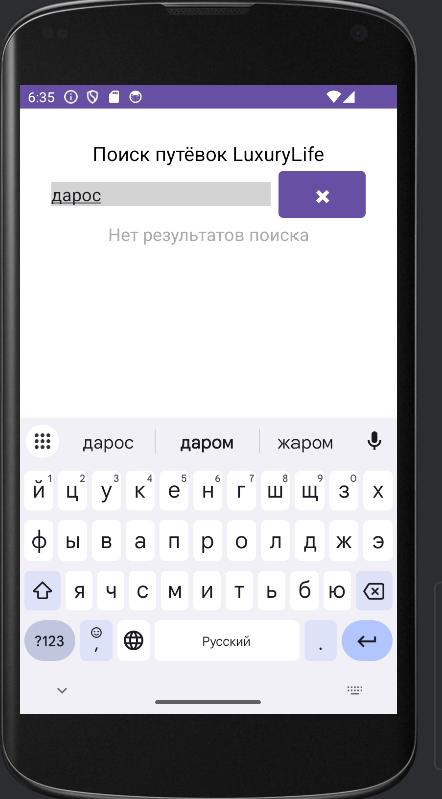


Рисунок 5. Ввели несуществующий запрос

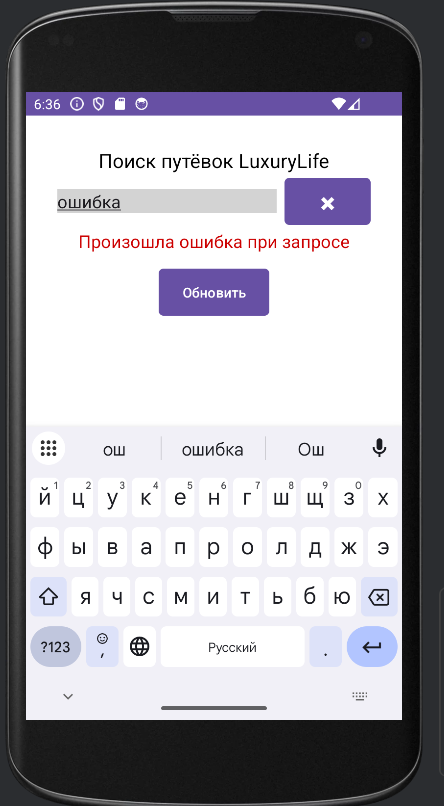


Рисунок 6. Ввели ключевое слово «ошибка» для проверки неудачного запроса

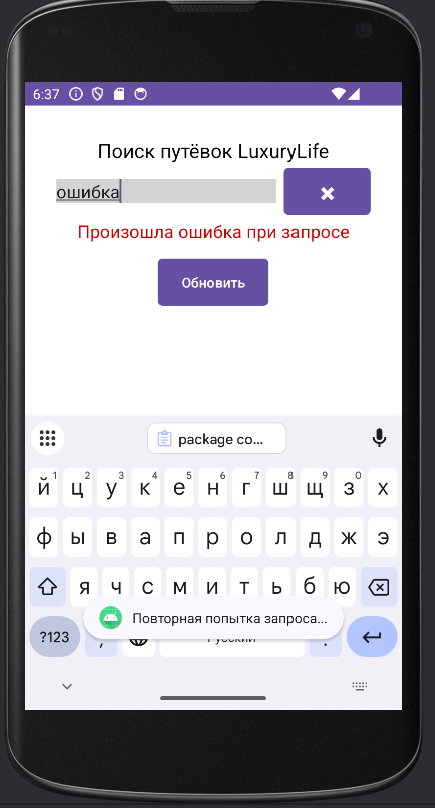


Рисунок 7. Повторная отправка запроса при нажатии на кнопку обновить

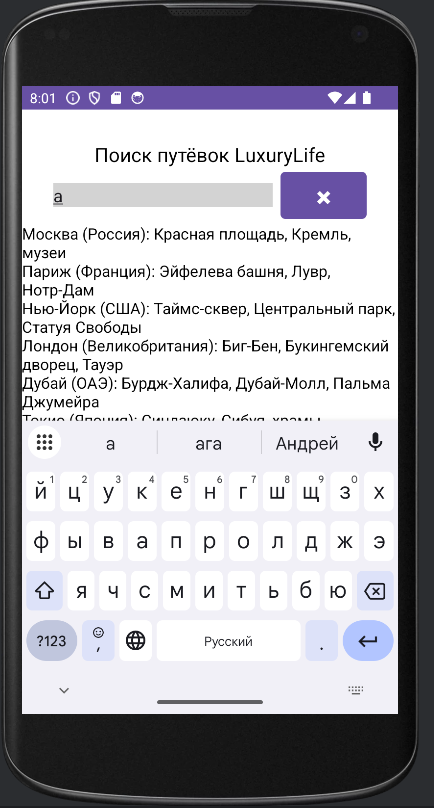


Рисунок 8. Проверка поиска №1

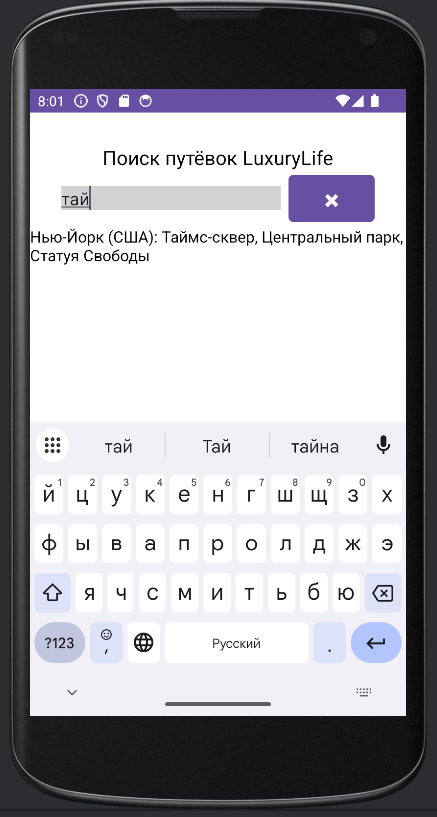


Рисунок 9. Проверка поиска №2

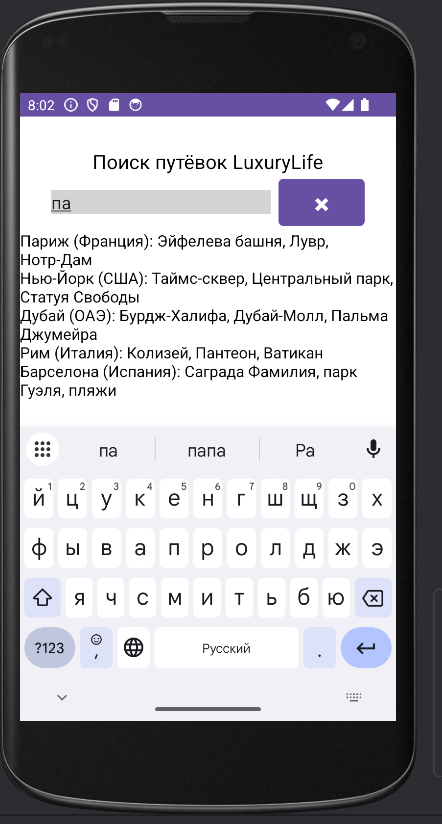


Рисунок 10. Проверка поиска №3

# Вывод

В ходе выполнения практической работы №2, мы узнали, как делать поисковую строку с API, кнопку Очистить, познакомились с плейсхолдерами, а также научились сохранять запроса в жизненном цикле при повороте экрана

# Список использованной литературы

1. Фримен Э., Робсон Э. Изучаем HTML, XHTML и CSS. — СПб.: Питер, 2007. — 656 с.: ил. [Печатное издание] (Общие знания о веб-разработке)
2. Макконнелл С. Совершенный код. Мастер-класс. — СПб.: Питер, 2005. — 896 с.: ил. [Печатное издание] (Основы разработки и архитектуры ПО)
3. Kotlin in Action / Dmitry Jemerov, Svetlana Isakova. — Manning Publications, 2017. — 400 с.: ил. [Печатное издание] (Основы языка Kotlin)
4. Android Programming: The Big Nerd Ranch Guide / Bill Phillips, Chris Stewart, Kristin Marsicano. — Big Nerd Ranch Guides, 2017. — 624 с.: ил. [Печатное издание] (Руководство по разработке Android)
5. Head First Android Development / David Griffiths, Dawn Griffiths. — O’Reilly Media, 2017. — 784 с.: ил. [Печатное издание] (Основы разработки Android)