Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет информатики и радиоэлектроники»

Факультет инфокоммуникаций

Кафедра инфокоммуникационных технологий

Дисциплина: Системы коммутации каналов и пакетов

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе

на тему

**Разработка приложения «Студенты»**

Студент гр.963101 Д.А. Коржовник

Руководитель Н.Н. Ловчий

Минск 2021

**СОДЕРЖАНИЕ**

[Введение 3](#_Toc89253340)

[1. Удаленная база данных firebase 4](#_Toc89253341)

[1.1 Краткие сведение о firebase 4](#_Toc89253342)

[1.2 Настройка firebase для среды разработки android studio 4](#_Toc89253343)

[2. Приложение «студенты» 11](#_Toc89253349)

[2.1 Концепция приложения 11](#_Toc89253350)

[2.2 Описание работы приложения 11](#_Toc89253351)

[Заключение 21](#_Toc89253352)

[Приложение А (обязательное) Блок-схема алгоритма загрузки списка студентов из базы данных на главном экране 22](#_Toc89253353)

[Приложение Б (обязательное) Листинг кода мобильного приложения 23](#_Toc89253354)

# **ВВЕДЕНИЕ**

В данный момент времени существует огромное множество форм представления и передачи информации. Число их форм столь разнообразно, что позволяет нам делать достаточно большой выбор в пользу того, или иного источника информации.

Нужно заметить, что при стремительном развитии в сфере интернета и информационных технологий, все больше становится необходима разработка мобильных приложений, наполненные определенным функционалом.

Актуальность данной темы заключается в том, что работа большинства мобильных приложений, так или иначе, связана с доступом к интернет — соединению. К примеру: социальные сети, мессенджеры, интернет — магазины, а, также, остальные приложения, связанные с работой удаленной базы данных.

Целью данного выбранной темы в рамках данного курсового проекта является исследование предметной области и создание работоспособного мобильного приложения, основанный на работе с облачной (удаленной) базой данных Firebase.

Задачи курсового проекта:

1. Проанализировать данную предметную область.

2. Ознакомиться с современными технологиями в интернете и, по возможности, использовать их.

3. Изучить полный арсенал инструментов, применяемый для создания, разработки и улучшения мобильных приложений.

4. Продумать реализацию приложения.

5. Реализовать мобильное приложение при помощи языка программирования Kotlin, основанное на работе с удаленной базой данных Firebase.

6. Протестировать готовое мобильное приложение на работоспособность.

Главным объектом данного курсового проекта является мобильное приложение «Студенты», работающее на Firebase. А предметом самого курсового проекта является методика создания мобильного приложения.

# **1. УДАЛЕННАЯ БАЗА ДАННЫХ Firebase**

## **1.1 Краткие сведение о Firebase**

**Firebase** — американская компания, поставщик облачных услуг, основанная в 2011 году Эндрю ли и Джейсом Тэмплином, и поглощённая в 2014 году корпорацией [Google](https://ru.wikipedia.org/wiki/Google_(компания)).

Прошла два раунда инвестиций: в мае 2012 года получила $1,4 млн от [Flybridge Capital Partners](https://ru.wikipedia.org/w/index.php?title=Flybridge_Capital_Partners&action=edit&redlink=1), [Greylock Partners](https://ru.wikipedia.org/w/index.php?title=Greylock_Partners&action=edit&redlink=1), [NEA](https://ru.wikipedia.org/w/index.php?title=New_Enterprise_Associates&action=edit&redlink=1), в июне 2013 года привлекла $5,6 миллионов от [Union Square Ventures](https://ru.wikipedia.org/w/index.php?title=Union_Square_Ventures&action=edit&redlink=1) и [Flybridge Capital Partners](https://ru.wikipedia.org/w/index.php?title=Flybridge_Capital_Partners&action=edit&redlink=1).

Основной сервис — [облачная СУБД](https://ru.wikipedia.org/w/index.php?title=Облачная_СУБД&action=edit&redlink=1) класса [NoSQL](https://ru.wikipedia.org/wiki/NoSQL), позволяющая разработчикам приложений хранить и синхронизировать данные между несколькими клиентами. Поддержаны особенности интеграции с приложениями под операционные системы [Android](https://ru.wikipedia.org/wiki/Android_(operating_system)) и [iOS](https://ru.wikipedia.org/wiki/IOS), реализовано [API](https://ru.wikipedia.org/wiki/API) для приложений на [JavaScript](https://ru.wikipedia.org/wiki/JavaScript), [Java](https://ru.wikipedia.org/wiki/Java_(язык_программирования)), [Objective-C](https://ru.wikipedia.org/wiki/Objective-C) и [Node.js](https://ru.wikipedia.org/wiki/Node.js), также возможно работать напрямую с базой данных в стиле [REST](https://ru.wikipedia.org/wiki/REST) из ряда JavaScript-фреймворков, включая [AngularJS](https://ru.wikipedia.org/wiki/AngularJS), [React](https://ru.wikipedia.org/wiki/React), [Vue.js](https://ru.wikipedia.org/wiki/Vue.js), [Ember.js](https://ru.wikipedia.org/wiki/Ember.js).

Firebase – это облачный сервис, сочетающий в себе множество функций: аутентификацию, базу данных в реальном времени, хранение файлов, уведомления и прочие. Данный сервис идеально подойдет для быстрого и простого создания серверной части приложения («back-end»), не требующих больших временных и трудовых затрат.

Разработчики, использующие эту платформу, получают доступ к сервисам, с помощью которых они смогут разрабатывать свои продукты, и это позволяет им сосредоточиться непосредственно на предоставлении качественного продукта.

Некоторые из самых популярных функций платформы Google Firebase включают в себя базы данных, аутентификацию, push-уведомления, аналитику, хранение файлов и многое другое.

Поскольку сервисы находятся в облаке, разработчики могут поэтапно выполнять масштабирование своих продуктов, не испытывая никаких проблем. Firebase на данный момент входит в число лучших платформ для разработки приложений, которым доверяют разработчики по всему миру.

## **1.2 Настройка Firebase для среды разработки Android Studio**

В данном разделе речь пойдет о подключении Firebase к проекту Android Studio - среды разработки мобильных приложений на операционной системе Android. Теперь перейдем непосредственно к настройке базы данных.

Вначале последует настройка самой базы данных Firebase, затем уже подключение Firebase к готовому проекту в Android Studio.

Для этого необходимо:

1. Перейти по ссылке на сайт: firebase.google.com (см. рисунок 1.1)
2. Нажать на кнопку «Get started».

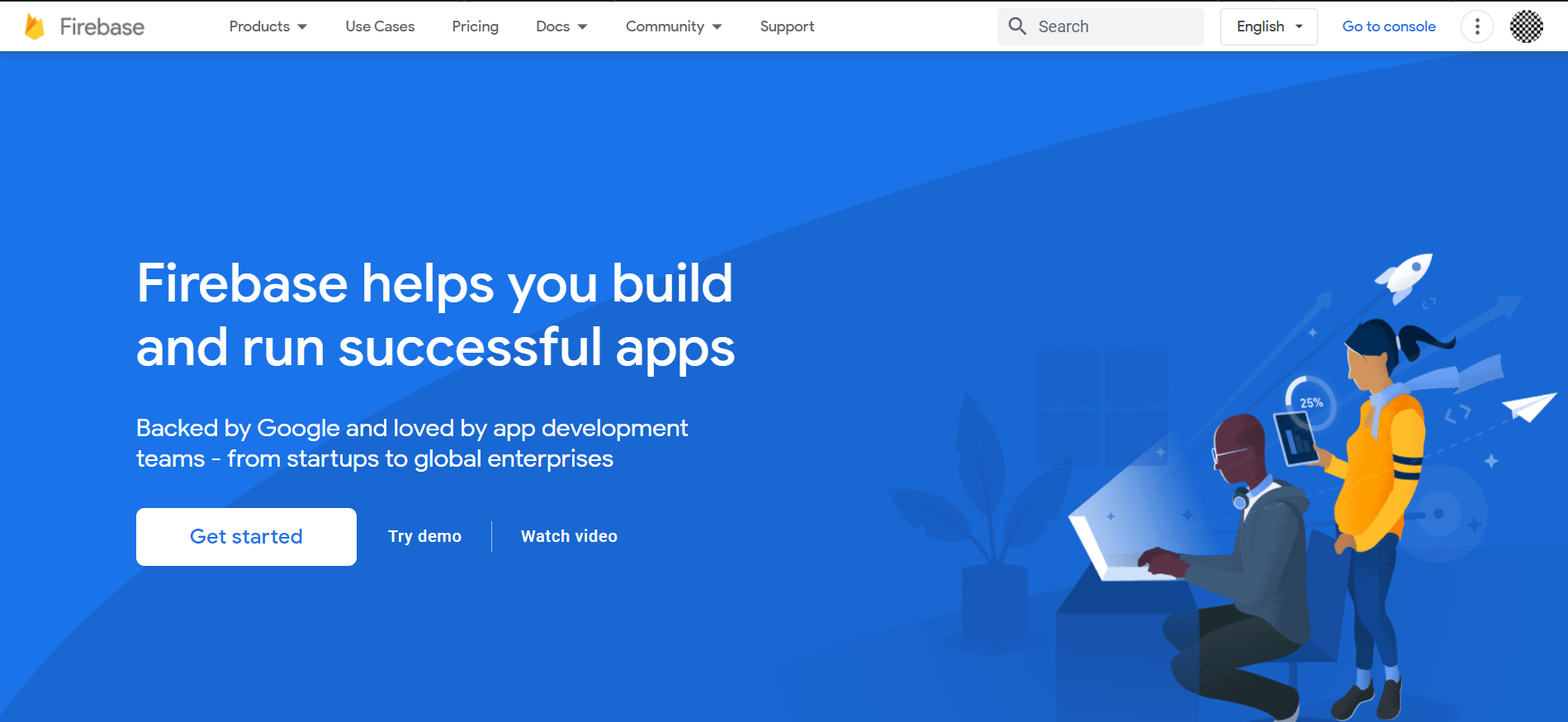


Рисунок 1.1 – Главный вид сайта Firebase

1. Выбрать «Add project» (см. рисунок 1.2).

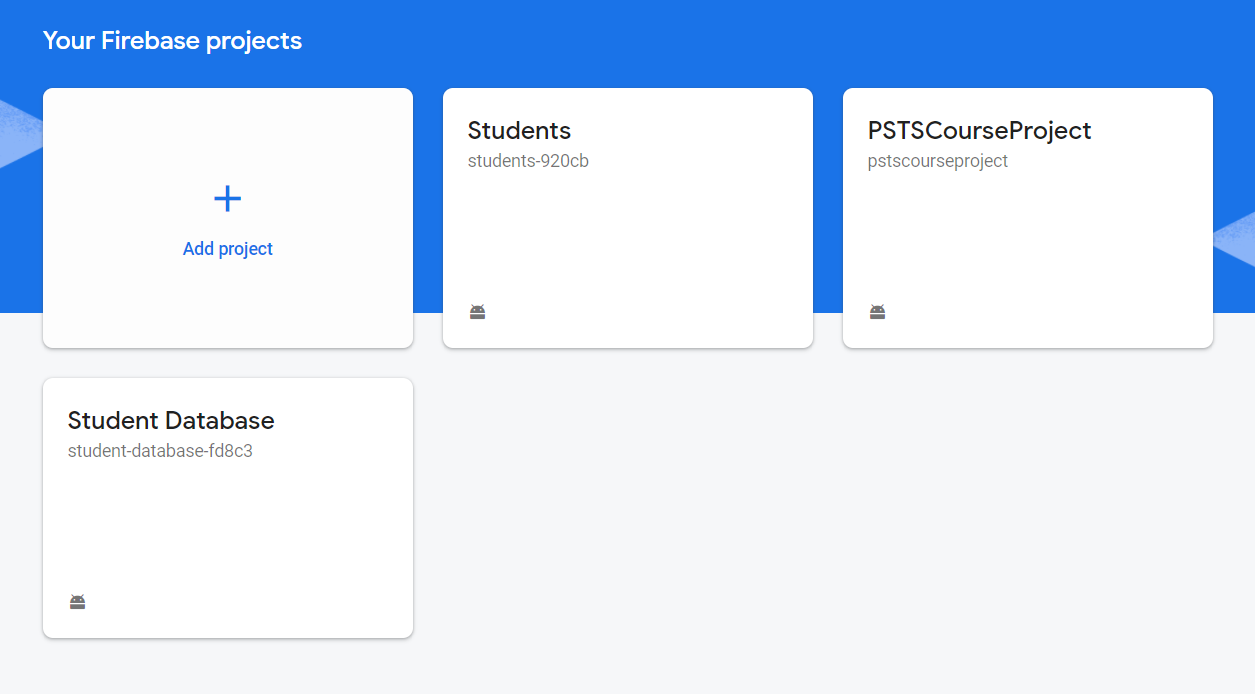


Рисунок 1.2 – Экран проектов

1. Дать название своему проекту (см. рисунок 1.3)

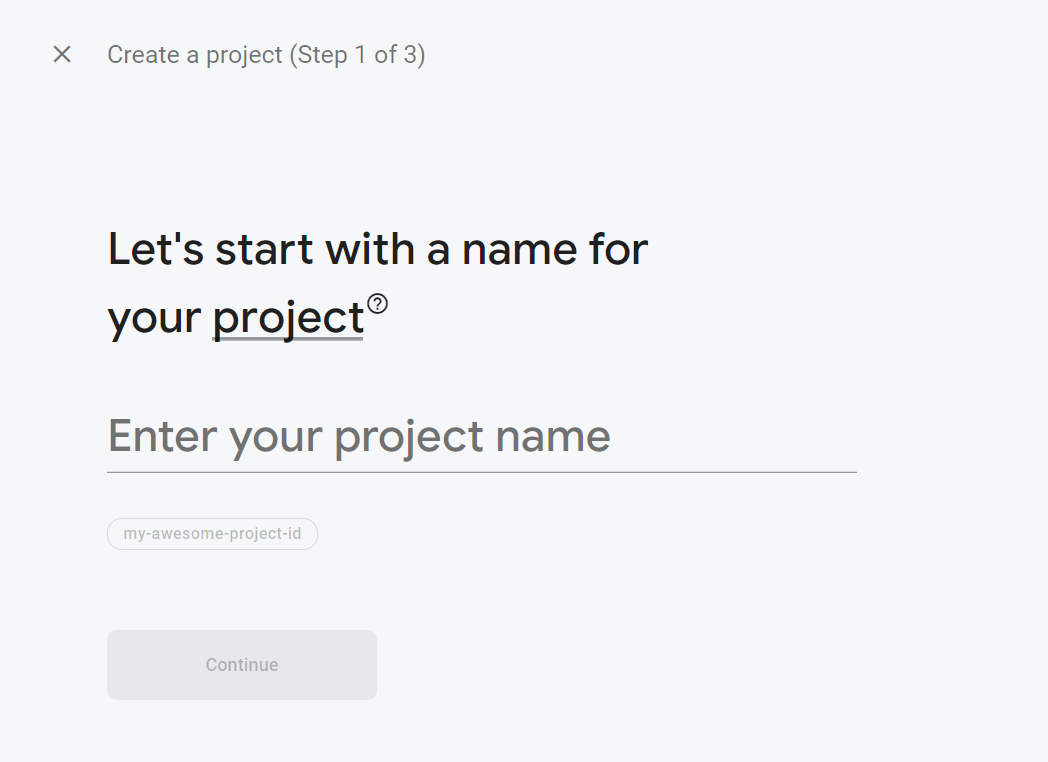


Рисунок 1.3 – Окно названия проекта

1. Выбрать «Default Account for Firebase» и нажать на «Add Firebase» (см. рисунок 1.4). После этого будет загрузка на рисунке 1.5.

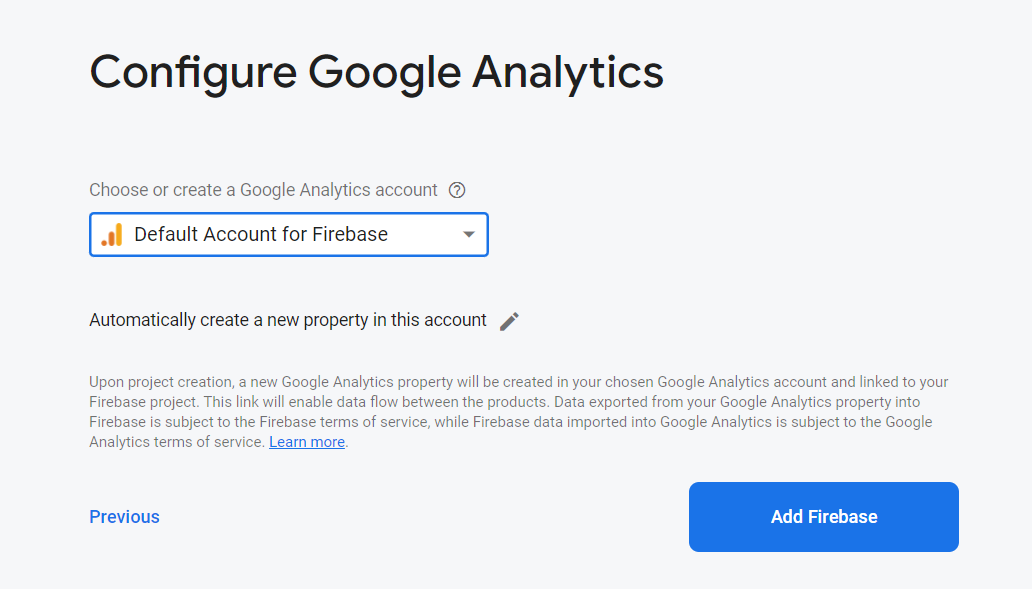


Рисунок 1.4 – Окно настройки аналитики Google

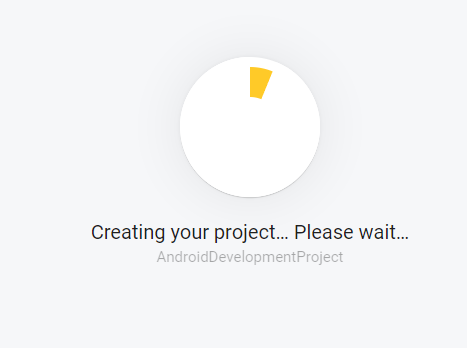


Рисунок 1.5 – Окно создание проекта

1. После этого загрузится окно ниже и необходимо перейти на вкладку «Realtime Database» бокового меню (см. рисунок 1.6).

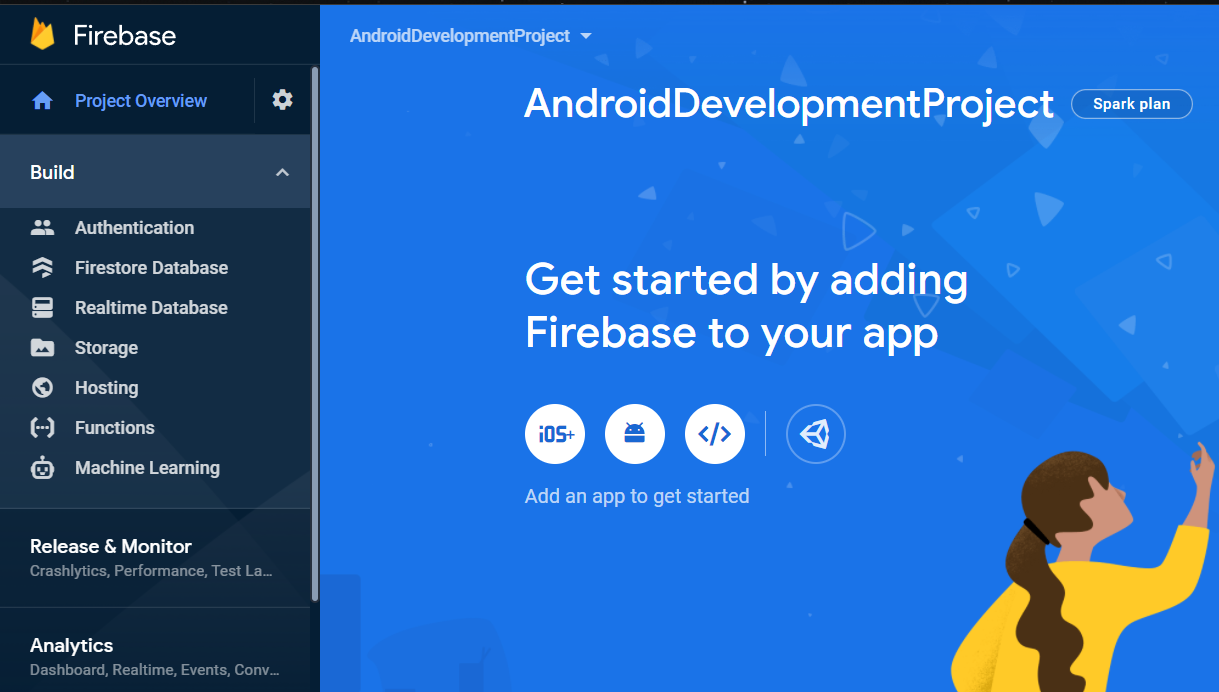


Рисунок 1.6 – Вид проекта

1. Далее, нажать на «Create database» (см. рисунок 1.7).

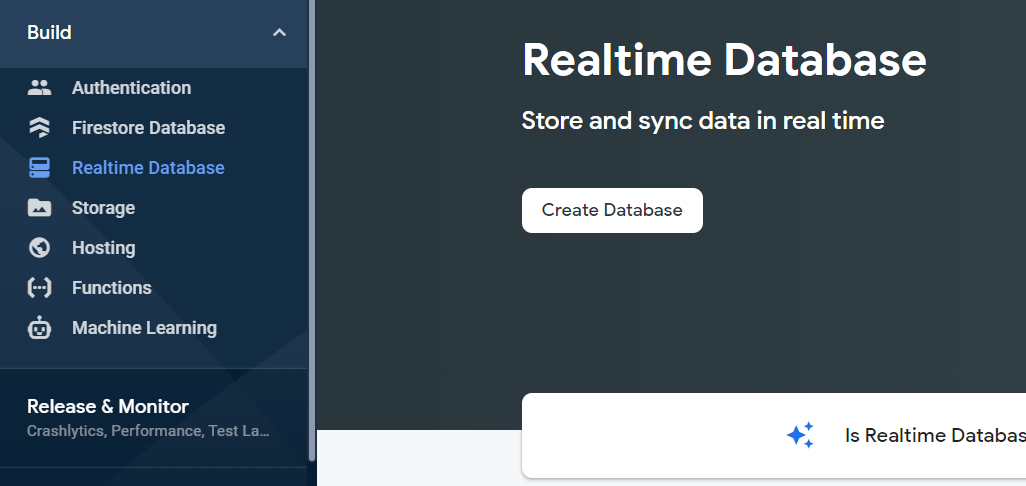


Рисунок 1.7 – Окна создания базы данных

1. На рисунках ниже (1.8 и 1.9) выбрать местоположение базы данных, указанные как на картинке и в правилах безопасности установить radio button «Start in test mode».

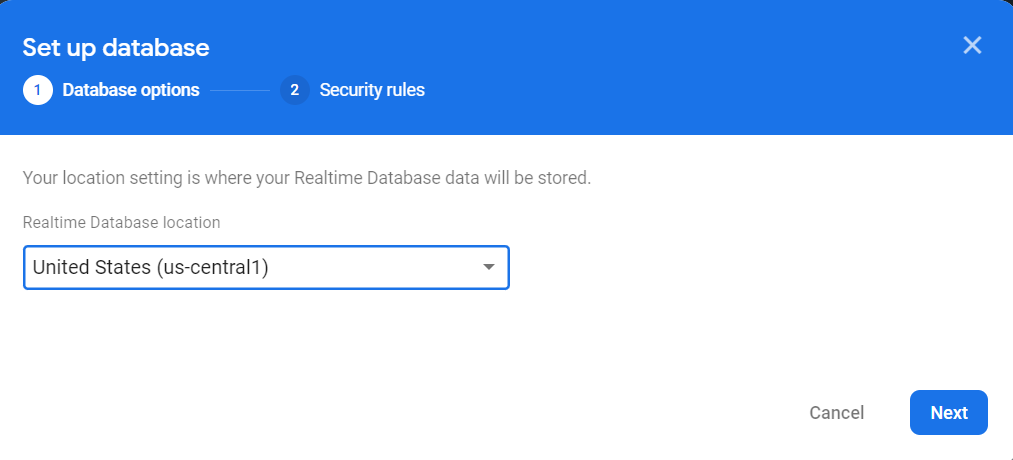


Рисунок 1.8 – Окно настройки базы данных

## 

Рисунок 1.9 – Окно правила безопасности

## После всех вышеописанных действий будет создана база данных в вашей директории. Для присоединения данной базы данных к существующему проекту Android Studio, необходимо:

## В проекте Android Studio, сверху на меню выбрать: Tools -> Firebase, как на рисунке 1.10.

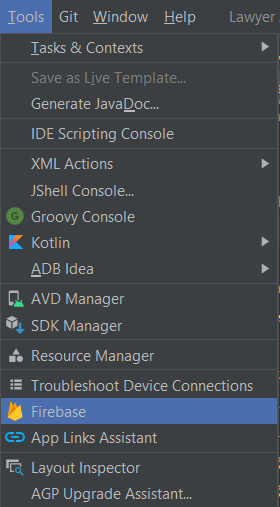


Рисунок 1.10 – Окно меню проекта Android Studio

## Выбрать «Realtime Database» в боковом меню Firebase, изображенном на рисунке 1.11.

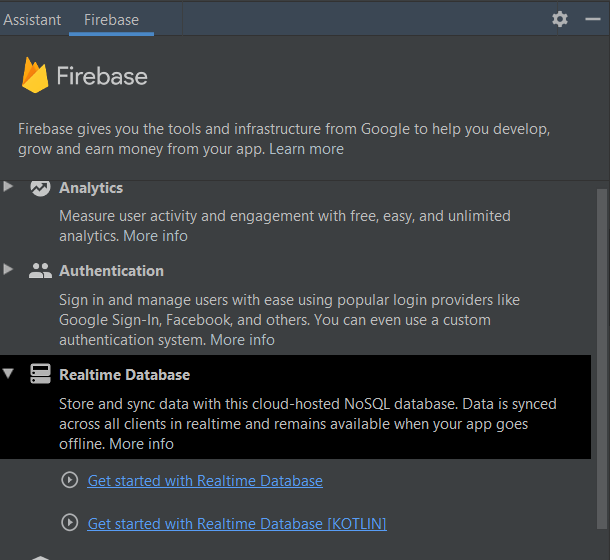


Рисунок 1.11 – Боковое окно Firebase в Android Studio

## В ниже открывшемся окне (см. рисунок 1.12) необходимо нажать на «Connect to Firebase», затем на «Add the Realtime Database SDK to your app».

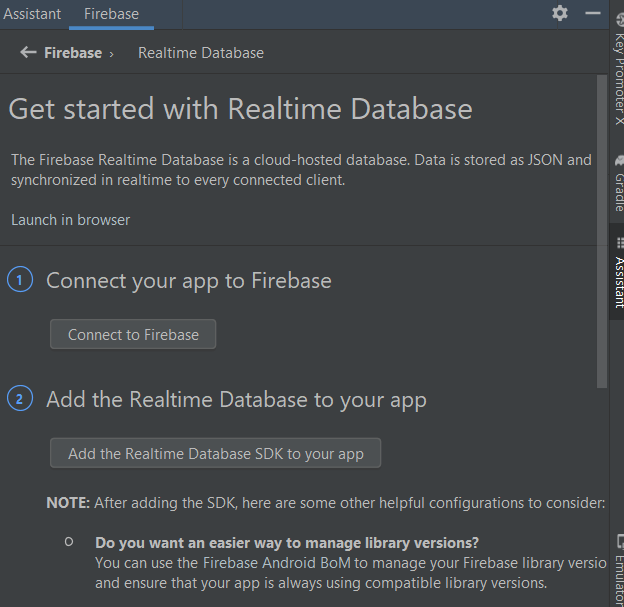


Рисунок 1.12 – Боковое окно подключение Firebase

После выполнения вышеописанных действий, база данных Firebase окончательно подключится к проекту Android Studio. Об этом уведомит окно ниже (см. рисунок 1.13)



Рисунок 1.13 – Окно завершения подключения базы данных

Теперь можно создавать код в проекте и выполнять запросы на базу данных или из базы данных.

# **2 ПРИЛОЖЕНИЕ «СТУДЕНТЫ»**

## **2.1 Концепция приложения**

В рамках данного курсового проекта будет создано приложение «Студенты». Основное предназначение этого приложения – хранить в облачной базе данных список студентов, каждый из которых содержит 3 основных параметра: имя, возраст и рейтинг. Также, еще и четвертый параметр – идентификационный ключ, который абсолютно уникален для каждого студента.

Данное приложение позволяет не только с Firebase загрузить список студентов, но и изменять список. К примеру: добавлять нового студента, редактировать текущего студента, удалять студента, сортировать список и т.д.

Еще, интересной особенностью данного приложения является его открытый доступ ко всем пользователям, которые установят данное приложения ввиду того, что в самой настройке удаленной базы данных уровень доступа указан как «открытый». Для реальных приложений с базами данных, открытая к чтению и редактированию база данных не является безопасной для хранения ценной, конфиденциальной информации. Однако, для демонстрации взаимодействия с Firebase, такой уровень доступа вполне приемлем. В случае, если база данных имела уровень доступа «закрытый», то в таком случае каждому пользователю необходимо будет произвести регистрацию с паролем для получения доступа к базе данных, что увеличивает безопасность хранимой информации.

## **2.2 Описание работы приложения**

В заключительной части главы речь пойдет непосредственно о работе самого мобильного приложения «Студенты» на базе операционной системы Android. Ниже будет изложена работа приложения, полный комплект функционала.

Первое, с чего начинается приложения, это с открытия главного экрана, где отображается список студентов, загруженных с облачной базы данных Firebase (см. рисунок 2.1).

Тем временем, на рисунке 2.2 показано хранение данных в удаленной базе данных Firebase.

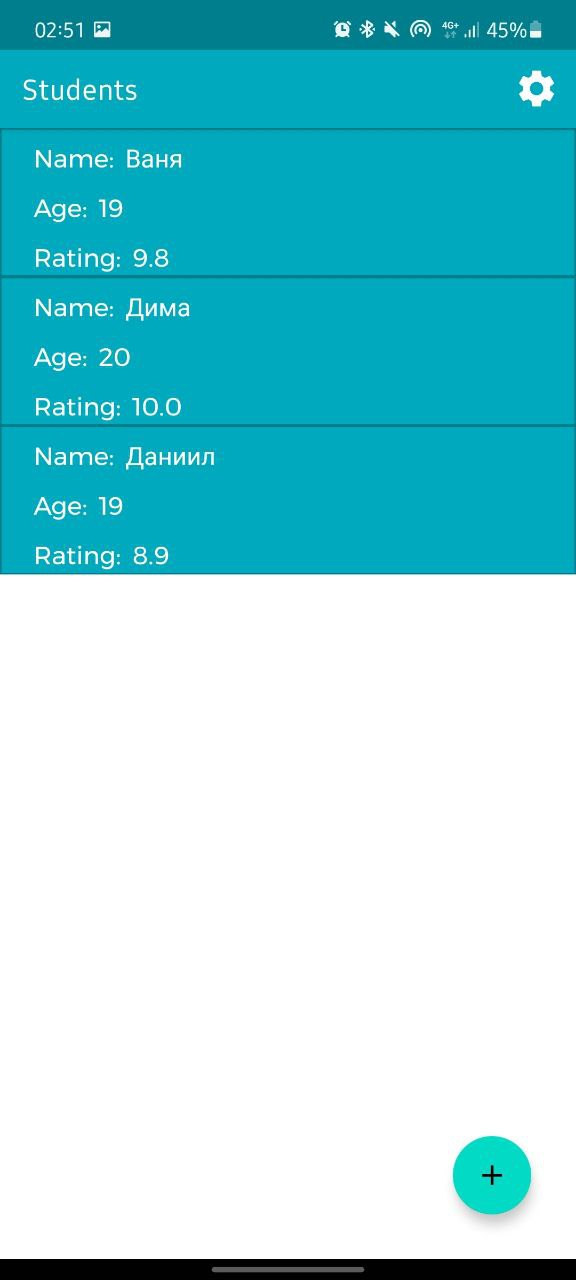


Рисунок 2.1 – Окно главного экрана

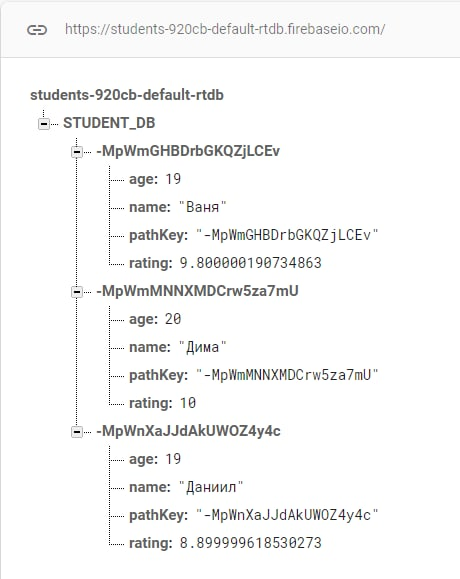


Рисунок 2.2 – Вид базы данных Firebase

Далее, мы можем перейти в окно добавления нового студента в базу данных по нажатии на плавающую кнопку снизу справа. На рисунке 2.3 будет отображен экран добавления студента. В нем содержится три поля редактирования для заполнения данных студента (имя, возраст и рейтинг) и две кнопки, которая возвращает назад и которая добавляет студента при корректном заполнении всех полей.

При некорректном заполнении полей (например, если не все поля будут заполнены) будет отображено всплывающее сообщение об этом (см. рисунок 2.4), а, иначе (см. рисунок 2.5), произойдет добавление нового студента и возврат на основной экран (см. рисунок 2.6), где обновится список студентов. Аналогично, информация будет отображена на сайте firebase, где расположена удаленная база данных (см. рисунки 2.7 и 2.8).

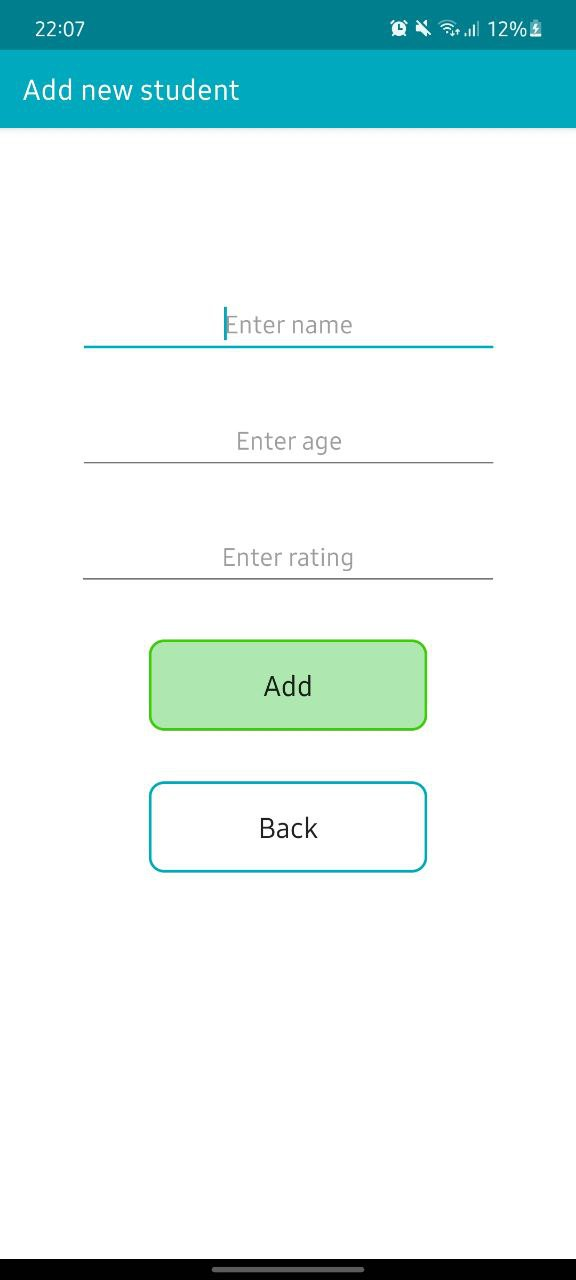


Рисунок 2.3 – Вид экрана добавления нового студента



Рисунок 2.4 – Вид экрана при некорректном заполнении данных

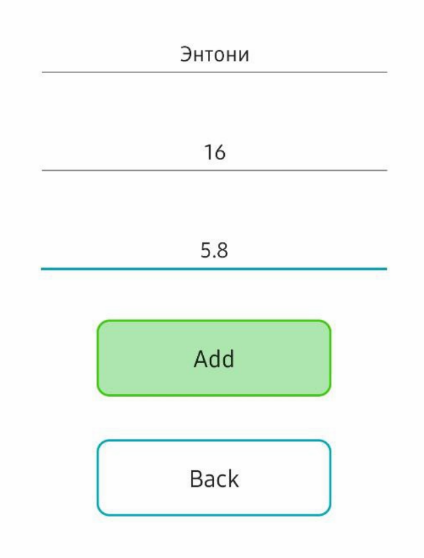


Рисунок 2.5 – Вид экрана корректного заполнения информации

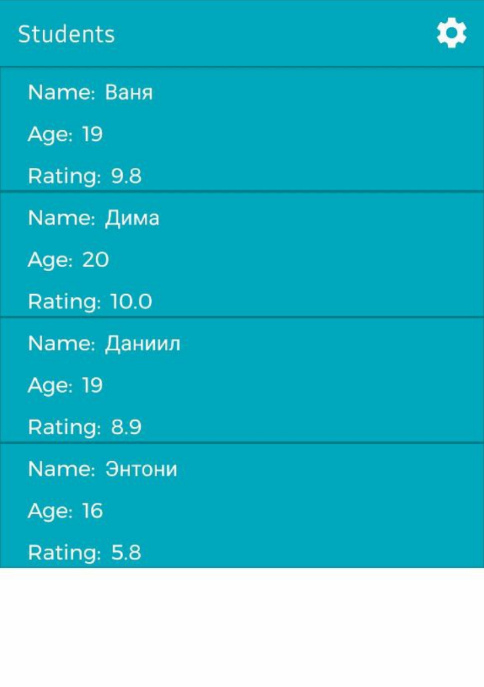


Рисунок 2.6 – Вид главного экрана с обновленной информацией



Рисунок 2.7 – Вид информации на удаленной базе данных

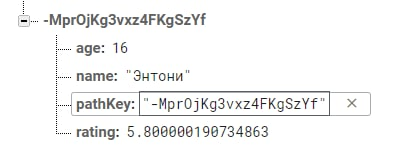


Рисунок 2.8 – Вид подробной информации добавленного студента

Далее, для редактирования информации текущего студента или для его удаления, нужно совершить нажатие по определенному элементу списка. После этого откроется окно редактирования с внешним сходством с экраном добавления. Отличаются лишь тем, что здесь вместо кнопки добавления расположена кнопка удаления и что после редактирования нужно просто вернуться назад на экран, тогда изменения в текущем элементе внесутся.

Ниже будет продемонстрировано предназначение экрана редактирования на рисунке 2.9.

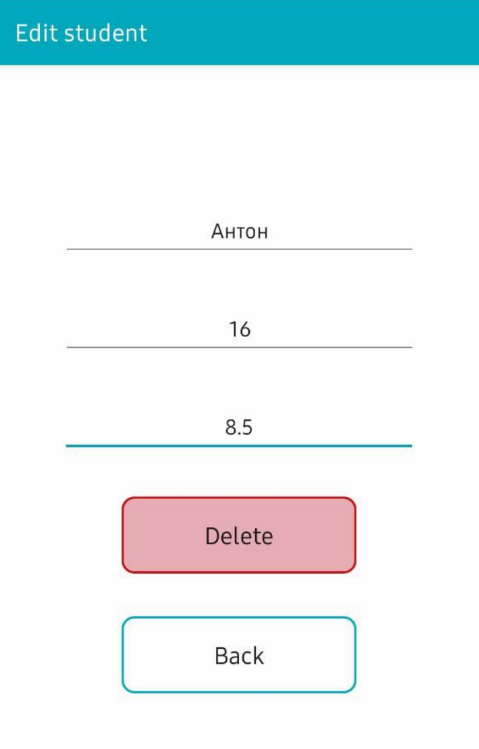


Рисунок 2.9 – Вид экрана редактирования студента

После возвращения на главный экран, изменится информация об текущем студенте. Причем, отображаться изменения будут именно так, в каком поле или в каких полях была изменена информация (см. рисунок 2.10).

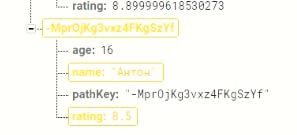


Рисунок 2.10 – Вид измененной информации о студенте

Далее, можно вернуться к экрану, изображенному на рисунке 2.9 и удалить данного студента нажав на кнопку удаления. После этого удаление студента будет отображено в базе данных (см. рисунок 2.11).



Рисунок 2.11 – Вид удаленного студента из базы данных

После тестирования основных методов работы с удаленной базой данных (таких как: добавление, редактирование, удаление и т.д.), можно перейти к экрану настроек по нажатию на иконку «шестеренки», расположенной сверху справа на главном экране.

При нажатии на иконку, мы увидим экран настроек (см. рисунок 2.12), на котором расположены три кнопки, предназначенные для сортировки загруженного списка студентов по трем параметрам: по имени (алфавитному порядку), по возрасту (по возрастанию) и по рейтингу (по убыванию).

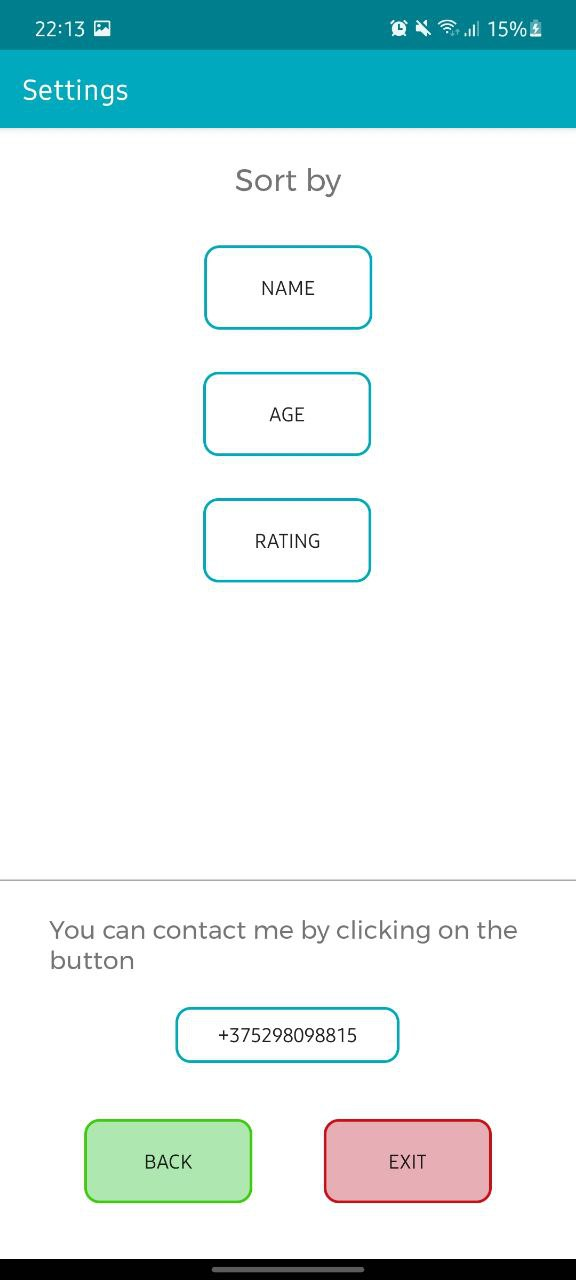


Рисунок 2.12 – Вид удаленного студента из базы данных

Ниже, на рисунках 2.13-15 будут показаны экраны с отсортированными списками. По умолчанию списки отображаются согласно расположению в базе данных: по дате создания студента. Ближайшая аналогия: очередь.

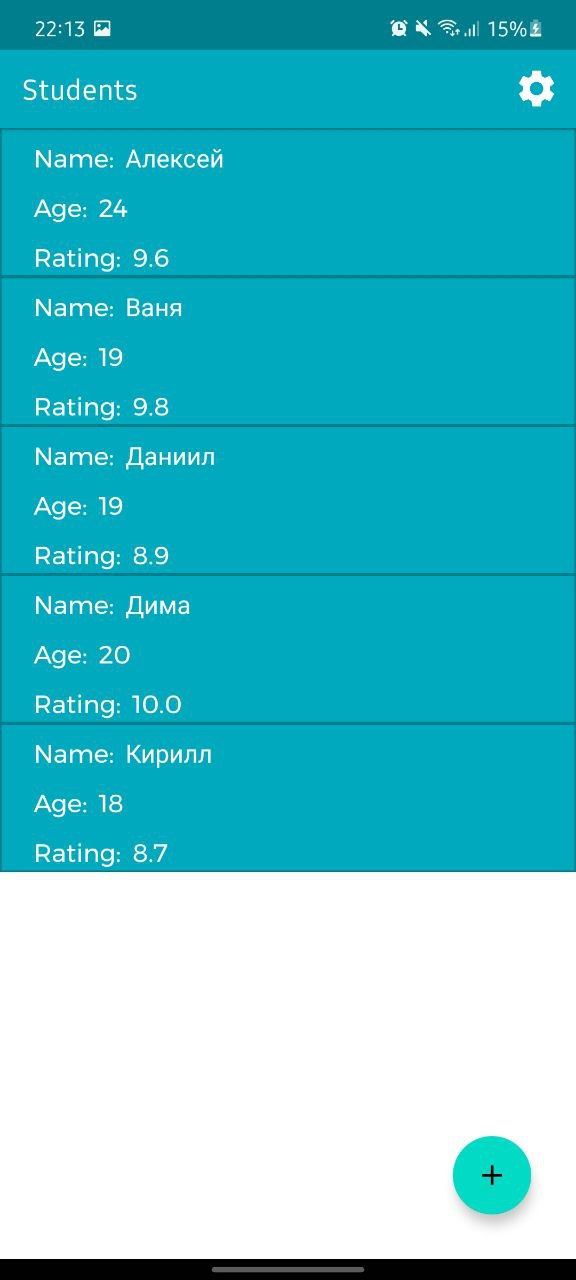


Рисунок 2.13 – Вид отсортированного списка по имени

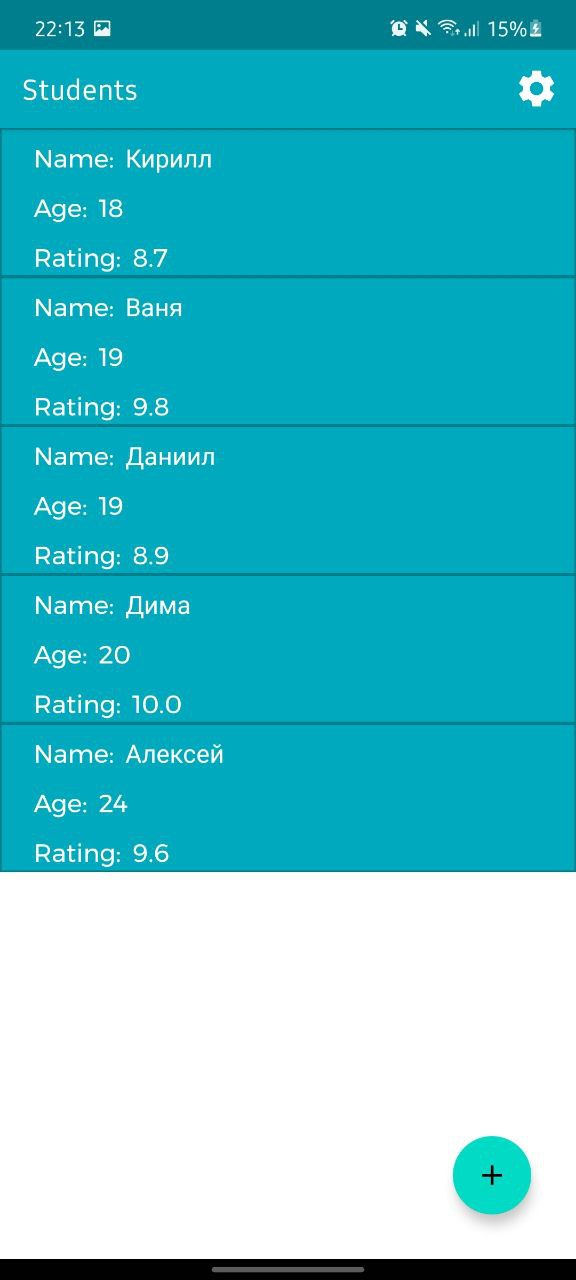


Рисунок 2.14 – Вид отсортированного списка по возрасту

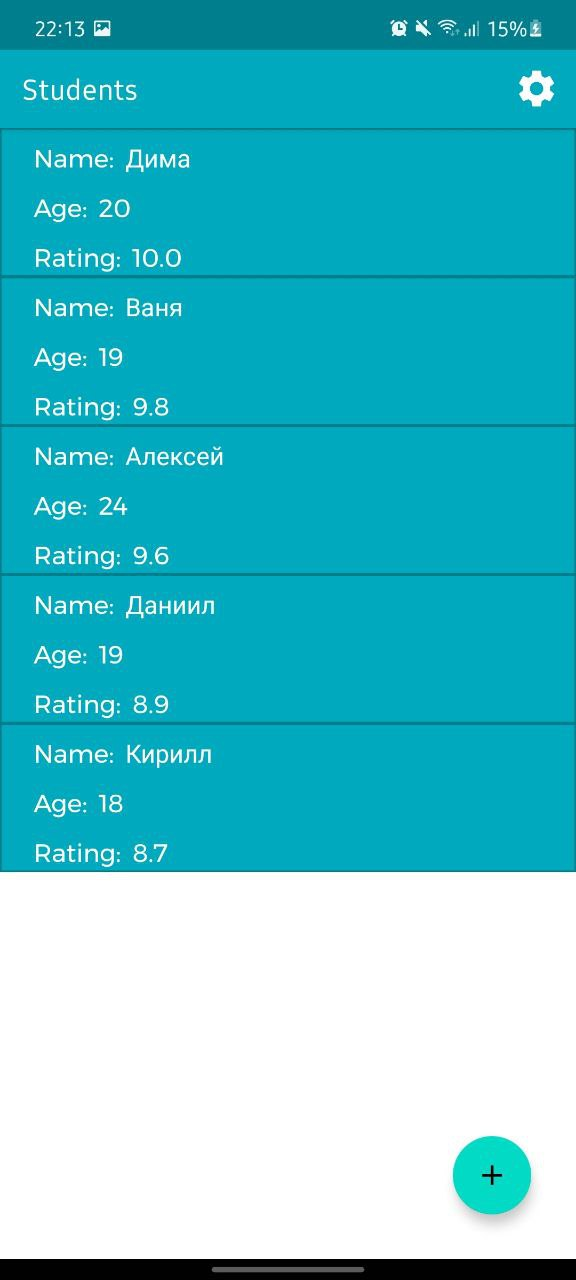


Рисунок 2.15 – Вид отсортированного списка по рейтингу

На экране настроек, ниже горизонтальной разделительной линии расположены три кнопки. Кнопка, расположенная посередине и над надписью «Вы можете связаться со мной, нажав на эту кнопку». При нажатии данной кнопки, при помощи специального Android-класса Intent происходит переход из нашего приложения в контакты, на котором отображен номер для связи (см. рисунок 2.16).

Далее, расположены оставшиеся две кнопки. При нажатии на первую кнопку происходит возвращение на главный экран. А при нажатии на вторую кнопку происходит выход из приложения.

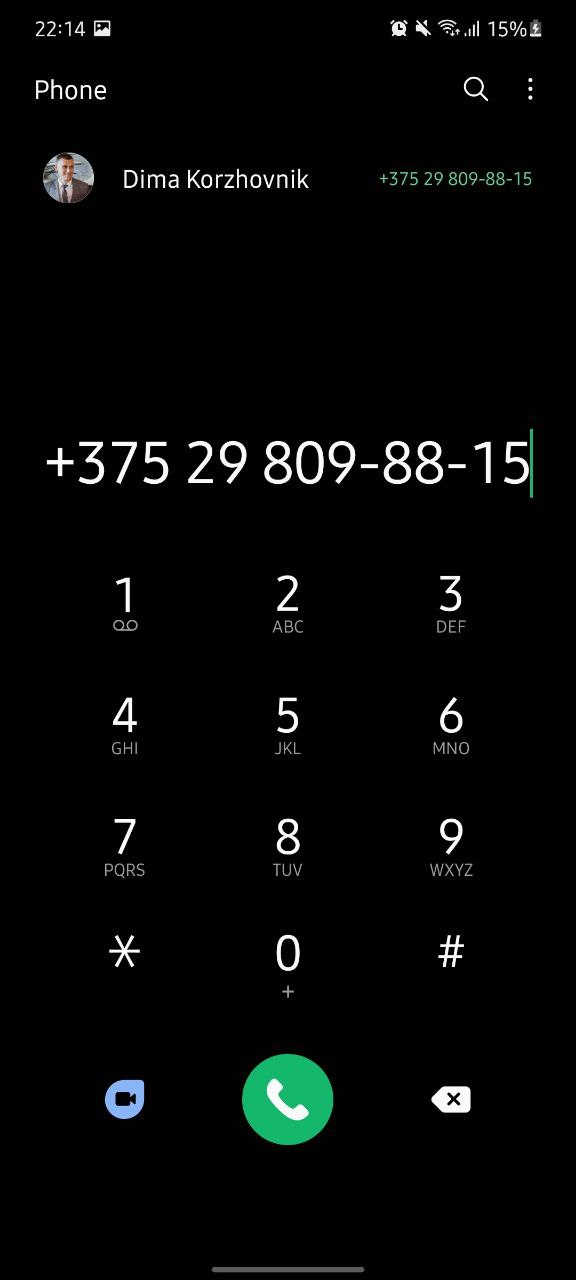


Рисунок 2.16 – Экран контактов после нажатия по кнопке связи

В данном разделе полностью показан весь функционал приложения для более комфортного управления удаленной базой данных через пользовательский интерфейс (UI – user interface). Весь исходный код данного приложения будет расположен на удаленном репозитории GitHub по ссылке: https://github.com/8DietmarBarbarossa8/PSTS\_Course\_Project

# **ЗАКЛЮЧЕНИЕ**

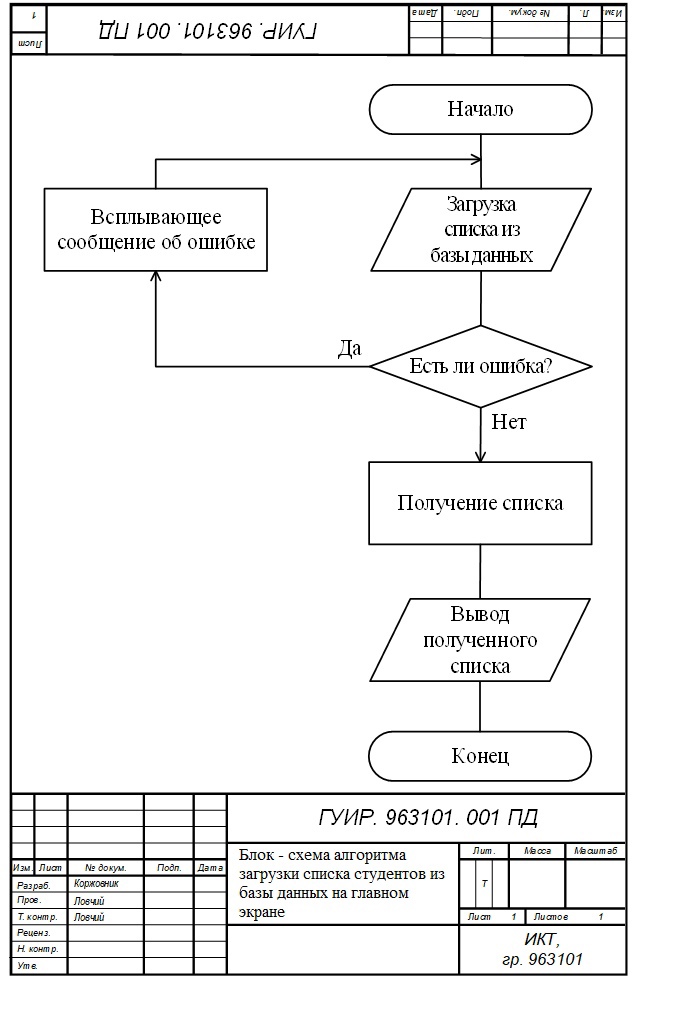
В ходе выполнения курсового проекта было получены сведения насчет работы удаленной базы данных Firebase, его краткого описания и функционала. Также происходило подключение Firebase к проекту мобильного приложения на базе операционной системы Android, создание приложения «Студенты» на базе работы с взаимодействием удаленной базы данных с пользовательским интерфейсом, включение функционала приложения для работы с базой данных.

Данное приложение показало возможность быстрого и простого создания его серверной части и более удобного взаимодействия с удаленной базой данных Firebase.

**ПРИЛОЖЕНИЕ А**

**(обязательное)**

**Блок-схема алгоритма загрузки списка студентов из базы данных на главном экране**



**ПРИЛОЖЕНИЕ Б**

**(обязательное)**

**Листинг кода мобильного приложения**

**MainActivity.kt**

package com.company.dima.studentsfirebase.activity

import android.os.Bundle

import androidx.appcompat.app.AppCompatActivity

import androidx.fragment.app.Fragment

import androidx.fragment.app.FragmentManager

import com.company.dima.studentsfirebase.R

import com.company.dima.studentsfirebase.fragments.StudentFragment

import com.company.dima.studentsfirebase.fragments.StudentListFragment

import com.company.dima.studentsfirebase.fragments.StudentSettingsFragment

import com.company.dima.studentsfirebase.inter\_face.Callbacks

class MainActivity : AppCompatActivity(), Callbacks {

override fun onCreate(savedInstanceState: Bundle?) {

super.onCreate(savedInstanceState)

setContentView(R.layout.activity\_main)

val fm: FragmentManager = supportFragmentManager

val currentFragment = fm.findFragmentById(R.id.fragment\_container)

if (currentFragment == null) onMainScreen()

}

override fun onMainScreen(sortingMode: String) {

this.setTitle(R.string.app\_name)

startFragment(StudentListFragment.newInstance(sortingMode))

}

override fun onCreateNewStudent() {

this.setTitle(R.string.adding\_label)

startFragment(StudentFragment.newInstance())

}

override fun onStudentSelected(pathKey: String) {

this.setTitle(R.string.editing\_label)

startFragment(StudentFragment.newInstance(pathKey))

}

override fun onSettings() {

this.setTitle(R.string.setting\_label)

startFragment(StudentSettingsFragment.newInstance())

}

private fun startFragment(fragment: Fragment) {

supportFragmentManager

.beginTransaction()

.replace(R.id.fragment\_container, fragment)

.commit()

}

}

**StudentFirebase.kt**

package com.company.dima.studentsfirebase.database

import com.company.dima.studentsfirebase.model.Student

import com.google.firebase.database.FirebaseDatabase

object StudentFirebase {

private const val KEY = "STUDENT\_DB"

private val firebase = FirebaseDatabase.getInstance()

val reference = firebase.getReference(KEY)

val students = arrayListOf<Student>()

fun addStudent(student: Student) {

student.pathKey = firebase.reference.push().key ?: ""

firebase.reference.child(KEY).child(student.pathKey).setValue(student)

}

fun deleteStudent(student: Student) {

firebase.reference.child(KEY).child(student.pathKey).removeValue()

}

fun updateStudent(student: Student) {

firebase.reference.child(KEY).child(student.pathKey).setValue(student)

}

fun sortStudents(sortMode: String) {

when (sortMode) {

"name" -> students.sortBy { it.name }

"age" -> students.sortBy { it.age }

"rating" -> students.sortBy { 1.0 / it.rating }

else -> students.sortBy { it.pathKey }

}

}

}

**StudentFragment.kt**

package com.company.dima.studentsfirebase.fragments

import android.content.Context

import android.os.Bundle

import android.view.LayoutInflater

import android.view.View

import android.view.ViewGroup

import android.widget.Button

import android.widget.EditText

import android.widget.Toast

import androidx.activity.addCallback

import androidx.core.content.res.ResourcesCompat

import androidx.fragment.app.Fragment

import com.company.dima.studentsfirebase.inter\_face.Callbacks

import com.company.dima.studentsfirebase.R

import com.company.dima.studentsfirebase.model.Student

import com.company.dima.studentsfirebase.database.StudentFirebase

class StudentFragment : Fragment() {

private lateinit var nameStudent: EditText

private lateinit var ageStudent: EditText

private lateinit var ratingStudent: EditText

private lateinit var functionalButton: Button

private lateinit var neutralButton: Button

private var student: Student = Student()

private var callbacks: Callbacks? = null

override fun onAttach(context: Context) {

super.onAttach(context)

callbacks = context as? Callbacks

}

override fun onCreate(savedInstanceState: Bundle?) {

super.onCreate(savedInstanceState)

requireActivity().onBackPressedDispatcher.addCallback(this) {

if (correctStudentData())

StudentFirebase.updateStudent(student)

snapBackToReality()

}

}

override fun onCreateView(

inflater: LayoutInflater,

container: ViewGroup?,

savedInstanceState: Bundle?

): View? {

val view = inflater.inflate(R.layout.fragment\_student, container, false)

nameStudent = view.findViewById(R.id.name\_student) as EditText

ageStudent = view.findViewById(R.id.age\_student) as EditText

ratingStudent = view.findViewById(R.id.rating\_student) as EditText

functionalButton = view.findViewById(R.id.add\_button) as Button

neutralButton = view.findViewById(R.id.neutral\_button) as Button

return view

}

override fun onViewCreated(view: View, savedInstanceState: Bundle?) {

super.onViewCreated(view, savedInstanceState)

val studentPathKey = arguments?.getSerializable(ARG\_STUDENT\_ID) as String?

if (studentPathKey != null) {

try {

student = StudentFirebase.students.find { it.pathKey == studentPathKey }!!

} catch (e: Exception) {

}

functionalButton.text = resources.getString(R.string.delete)

functionalButton.background = ResourcesCompat

.getDrawable(resources, R.drawable.deleting\_button\_border, null)

nameStudent.setText(student.name)

ageStudent.setText(student.age.toString())

ratingStudent.setText(student.rating.toString())

functionalButton.setOnClickListener {

StudentFirebase.deleteStudent(student)

Toast.makeText(context, R.string.deleting\_notification, Toast.LENGTH\_SHORT).show()

snapBackToReality()

}

} else addStudent()

neutralButton.setOnClickListener {

if (correctStudentData() && studentPathKey != null)

StudentFirebase.updateStudent(student)

snapBackToReality()

}

}

private fun addStudent() {

functionalButton.setOnClickListener {

val name = nameStudent.text.toString()

val age = ageStudent.text.toString().toIntOrNull()

val rating = ratingStudent.text.toString().toFloatOrNull()

if (name.isNotEmpty() && age != null && rating != null) {

StudentFirebase.addStudent(Student(name, age, rating))

Toast.makeText(context, R.string.adding\_notification, Toast.LENGTH\_SHORT).show()

snapBackToReality()

} else Toast.makeText(

context,

R.string.input\_error\_notification, Toast.LENGTH\_SHORT

).show()

}

}

private fun correctStudentData(): Boolean {

val student = Student()

student.name = nameStudent.text.toString()

student.age = try {

ageStudent.text.toString().toInt()

} catch (e: Exception) {

student.age

}

student.rating = try {

ratingStudent.text.toString().toFloat()

} catch (e: Exception) {

student.rating

}

if (!(this.student.name == student.name &&

this.student.age == student.age &&

this.student.rating == student.rating)

) {

this.student.name = student.name

this.student.age = student.age

this.student.rating = student.rating

return true

}

return false

}

private fun snapBackToReality() {

callbacks?.onMainScreen()

}

override fun onDetach() {

super.onDetach()

callbacks = null

}

companion object {

private const val ARG\_STUDENT\_ID = "student\_id"

@JvmStatic

fun newInstance(pathKey: String): StudentFragment {

val args = Bundle().apply { putString(ARG\_STUDENT\_ID, pathKey) }

return StudentFragment().apply { arguments = args }

}

@JvmStatic

fun newInstance() = StudentFragment()

}

}

**StudentListFragment.kt**

package com.company.dima.studentsfirebase.fragments

import android.content.Context

import android.os.Bundle

import android.view.\*

import android.widget.TextView

import android.widget.Toast

import androidx.core.os.bundleOf

import androidx.fragment.app.Fragment

import androidx.recyclerview.widget.LinearLayoutManager

import androidx.recyclerview.widget.RecyclerView

import androidx.swiperefreshlayout.widget.SwipeRefreshLayout

import com.company.dima.studentsfirebase.inter\_face.Callbacks

import com.company.dima.studentsfirebase.R

import com.company.dima.studentsfirebase.model.Student

import com.company.dima.studentsfirebase.database.StudentFirebase

import com.google.android.material.floatingactionbutton.FloatingActionButton

import com.google.firebase.database.DataSnapshot

import com.google.firebase.database.DatabaseError

import com.google.firebase.database.ValueEventListener

class StudentListFragment : Fragment() {

private lateinit var studentRecyclerView: RecyclerView

private var adapter: StudentAdapter = StudentAdapter(emptyList())

private var callbacks: Callbacks? = null

private lateinit var menu: Menu

private var addingButton: FloatingActionButton? = null

override fun onAttach(context: Context) {

super.onAttach(context)

callbacks = context as? Callbacks

StudentFirebase.students.clear()

}

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\_student\_list, container, false)

studentRecyclerView = view.findViewById(R.id.student\_recycler\_view) as RecyclerView

studentRecyclerView.layoutManager = LinearLayoutManager(context)

studentRecyclerView.adapter = adapter

return view

}

override fun onCreateOptionsMenu(menu: Menu, inflater: MenuInflater) {

super.onCreateOptionsMenu(menu, inflater)

inflater.inflate(R.menu.fragment\_student\_list, menu)

this.menu = menu

}

override fun onOptionsItemSelected(item: MenuItem): Boolean {

return when (item.itemId) {

R.id.sorting\_student -> {

callbacks?.onSettings()

true

}

else -> return super.onOptionsItemSelected(item)

}

}

override fun onViewCreated(view: View, savedInstanceState: Bundle?) {

super.onViewCreated(view, savedInstanceState)

val swipeRefreshLayout: SwipeRefreshLayout = view.findViewById(R.id.refresh)

swipeRefreshLayout.setOnRefreshListener {

callbacks?.onMainScreen()

}

StudentFirebase.reference.addValueEventListener(object : ValueEventListener {

override fun onDataChange(dataSnapshot: DataSnapshot) {

for (ds in dataSnapshot.children)

try {

StudentFirebase.students.add(requireNotNull(ds.getValue(Student::class.java)))

} catch (e: Exception) {

continue

}

val set = HashSet<Student>(StudentFirebase.students)

StudentFirebase.students.clear()

StudentFirebase.students.addAll(set)

StudentFirebase.sortStudents(requireNotNull(arguments?.getString(SORTING\_MODE)))

adapter.students = StudentFirebase.students

studentRecyclerView.adapter = adapter

}

override fun onCancelled(databaseError: DatabaseError) {

Toast.makeText(context, "$databaseError", Toast.LENGTH\_SHORT).show()

}

})

addingButton = view.findViewById(R.id.floatingActionButton)

addingButton?.setOnClickListener {

callbacks?.onCreateNewStudent()

}

}

override fun onDetach() {

super.onDetach()

callbacks = null

}

private inner class StudentHolder(view: View) : RecyclerView.ViewHolder(view),

View.OnClickListener {

private lateinit var student: Student

private val nameTextView: TextView = itemView.findViewById(R.id.name\_student)

private val ageTextView: TextView = itemView.findViewById(R.id.age\_student)

private val ratingTextView: TextView = itemView.findViewById(R.id.rating\_student)

init {

itemView.setOnClickListener(this)

}

fun bind(student: Student) {

this.student = student

nameTextView.text = this.student.name

ageTextView.text = this.student.age.toString()

ratingTextView.text = this.student.rating.toString()

}

override fun onClick(v: View) {

callbacks?.onStudentSelected(student.pathKey)

}

}

private inner class StudentAdapter(var students: List<Student>) :

RecyclerView.Adapter<StudentHolder>() {

override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): StudentHolder {

val layoutInflater = LayoutInflater.from(context)

val view = layoutInflater.inflate(R.layout.list\_item\_student, parent, false)

return StudentHolder(view)

}

override fun onBindViewHolder(holder: StudentHolder, position: Int) {

holder.bind(students[position])

}

override fun getItemCount() = students.size

}

companion object {

private const val SORTING\_MODE = "sorting mode"

@JvmStatic

fun newInstance(sortingMode: String = ""):

StudentListFragment = StudentListFragment().apply {

arguments = bundleOf(SORTING\_MODE to sortingMode)

}

}

}

**StudentSettingsFragment.kt**

package com.company.dima.studentsfirebase.fragments

import android.Manifest

import android.app.Activity

import android.content.Context

import android.content.Intent

import android.content.pm.PackageManager

import android.net.Uri

import android.os.Bundle

import android.view.LayoutInflater

import android.view.View

import android.view.ViewGroup

import android.widget.Button

import androidx.fragment.app.Fragment

import androidx.activity.addCallback

import androidx.core.app.ActivityCompat

import androidx.core.content.ContextCompat

import com.company.dima.studentsfirebase.inter\_face.Callbacks

import com.company.dima.studentsfirebase.R

import kotlin.system.exitProcess

class StudentSettingsFragment : Fragment() {

private var buttons: Array<Button?> = Array(3) { null }

private lateinit var contactButton: Button

private lateinit var backButton: Button

private lateinit var exitButton: Button

private var callbacks: Callbacks? = null

override fun onAttach(context: Context) {

super.onAttach(context)

callbacks = context as? Callbacks

}

override fun onCreate(savedInstanceState: Bundle?) {

super.onCreate(savedInstanceState)

requireActivity().onBackPressedDispatcher.addCallback(this) {

callbacks?.onMainScreen()

}

}

override fun onCreateView(

inflater: LayoutInflater, container: ViewGroup?,

savedInstanceState: Bundle?

): View? {

return inflater.inflate(R.layout.fragment\_student\_settings, container, false)

}

override fun onViewCreated(view: View, savedInstanceState: Bundle?) {

super.onViewCreated(view, savedInstanceState)

val idButtonArray = arrayOf(R.id.nameButton, R.id.ageButton, R.id.ratingButton)

for (i in idButtonArray.indices) {

buttons[i] = view.findViewById(idButtonArray[i])

buttons[i]?.setOnClickListener {

callbacks?.onMainScreen(switchSortParameter(i))

}

}

contactButton = view.findViewById(R.id.contact\_button)

contactButton.setOnClickListener {

if (isCallPermissionGranted()) callMe()

else requestCallPermission()

}

backButton = view.findViewById(R.id.neutral\_button)

backButton.setOnClickListener {

callbacks?.onMainScreen()

}

exitButton = view.findViewById(R.id.exit\_button)

exitButton.setOnClickListener {

exitProcess(0)

}

}

private fun switchSortParameter(number: Int): String =

when (number) {

0 -> "name"

1 -> "age"

2 -> "rating"

else -> ""

}

private fun isCallPermissionGranted(): Boolean =

context?.let {

ContextCompat.checkSelfPermission(

it,

Manifest.permission.CALL\_PHONE

)

} == PackageManager.PERMISSION\_GRANTED

private fun requestCallPermission() {

ActivityCompat.requestPermissions(

context as Activity,

arrayOf(Manifest.permission.CALL\_PHONE), REQUEST\_CODE

)

}

private fun callMe() {

val dial = "tel: +${resources.getString(R.string.phone)}"

val intent = Intent(Intent.ACTION\_DIAL, Uri.parse(dial))

startActivity(intent)

}

override fun onDetach() {

super.onDetach()

callbacks = null

}

companion object {

private const val REQUEST\_CODE = 100

@JvmStatic

fun newInstance() = StudentSettingsFragment()

}

}

**Callbacks.kt**

package com.company.dima.studentsfirebase.inter\_face

interface Callbacks {

fun onCreateNewStudent()

fun onStudentSelected(pathKey: String)

fun onSettings()

fun onMainScreen(sortingMode: String = "")

}

**Student.kt**

package com.company.dima.studentsfirebase.model

import android.os.Parcelable

import kotlinx.parcelize.Parcelize

@Parcelize

data class Student(

var name: String = "",

var age: Int = 0,

var rating: Float = 0.0f,

var pathKey: String = ""

) : Parcelable