Министерство образования и молодежной политики Свердловской области



ГАПОУ СО «Екатеринбургский колледж транспортного строительства»

# Отчёт по программе «Базы данных в Android Studio»

Выполнила: Мезенцева Полина Сергеевна

Группа: ПР-31

Преподаватель: Мирошниченко Г.В

2024

# Вариант 6:

1. Cоздать приложение для База данных «Колледж»

В этой базе заказчик хотел бы хранить информацию

 номер студента (является первичным ключом и изменяется на 1, при

добавлении нового студента)

 наименование группы с возможностью группировки по наименованию

группы (выводится список студентов)

 курс

 номер специальности (с возможностью группировки по специальности

– выдается количество студентов на специальности)

 ФИО студента (фото студента) – загрузить любые фото в проект (или

запрашивать сохраненные фото в яндекс диске)

• дата рождения студента, (datepicker с ограничением)

• бюджет / внебюджет

 преподаватели

Некоторые условия работы колледжа, существенные для проектирования

базы данных:

• каждый студент может учиться на более чем одной специальности, но

только 1 специальность бюджет

• один преподаватель может вести на нескольких специальностях, но

ограничение нагрузки (не более 1440 часов/год), более, начисляется

дополнительная премия к зарплате.

На главной странице должна быть реализована авторизация и выбор режима

работы (для приемной комиссии, для преподавателя, для студента), с

проверкой на правильность введенных данных (проверка на правильность

ввода почты, логина и пароля) и сохранением через Shared Preference в базе

данных.

В приложении должно быть организовано не менее 4-х – 5 окон

Навигацию осуществить наиболее оптимальным образом для приложения (с

помощью Carousel with MotionLayout, Navigation) – должны быть

использованы не менее 2-х видов навигации.

)

В приложении должен быть реализован следующий функционал:

Редактирование, удаление, добавление студентов и преподавателей.

На главной странице поиск по наименованию группы или ФИО студента.

2. Создайте и заполните готовую базу и подключите его в проекте через

ROOM.

# Структура проекта

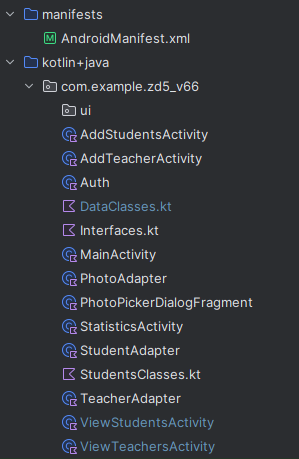
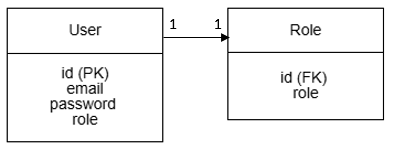
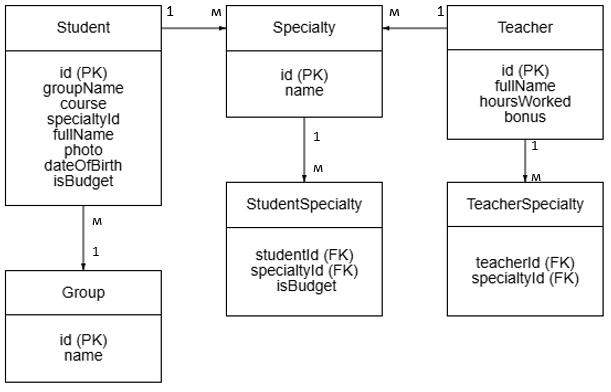
**

Схема классов



ER-диаграмма №1



ER-диаграмма №2

# Входные данные:

*Экран авторизации*: Поля для ввода логина, пароля, выбора роли.

*Экран учителя*: Поля для ввода имени, фамилии, курса, специальности, выбор даты рождения.

*Экраны комиссии*: Поля для ввода ФИО студента/учителя, группы, курса, выбор специальности студента/учителя, выбор даты рождения, выбор фото студента/учителя, нагрузка для учителя

**Выходные данные**

Списки RecycleView для учителей и студентов, с информацией, упомянутой выше.

# Описание разработанных функций и методов:

Методы класса AddStudentsActivity:

|  |  |
| --- | --- |
| deleteSpeciality(view: View) | Сбрасывает выбранные специальности, очищает текстовое поле и показывает уведомление о сбросе. |

Методы класса AddTeacherActivity:

|  |  |
| --- | --- |
| selectedDate(view: View) | Открывает диалог выбора даты с ограничениями по минимальной и максимальной дате. |

Методы класса PhotoPickerDialogFragment:

|  |  |
| --- | --- |
| onCreateDialog(savedInstanceState: Bundle?): Dialog | Создает диалог с выбором фотографии из списка, используя RecyclerView. |

Методы класса StatisticsActivity:

|  |  |
| --- | --- |
| loadStatistics() | Загружает статистику (количество студентов и преподавателей) и обновляет адаптер. |

Методы класса ViewStudentsActivity:

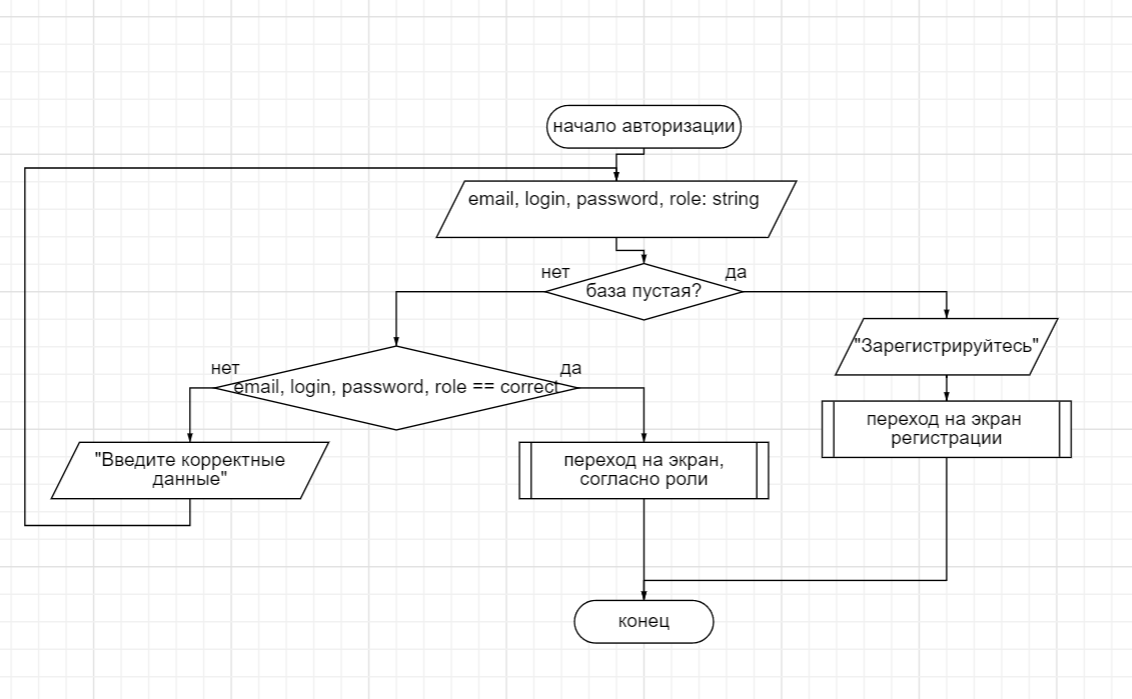
|  |  |
| --- | --- |
| loadStudents() | Загружает список студентов из базы данных и обновляет адаптер. |
| searchStudents(name: String) | Ищет студентов по имени и обновляет список в адаптере. |

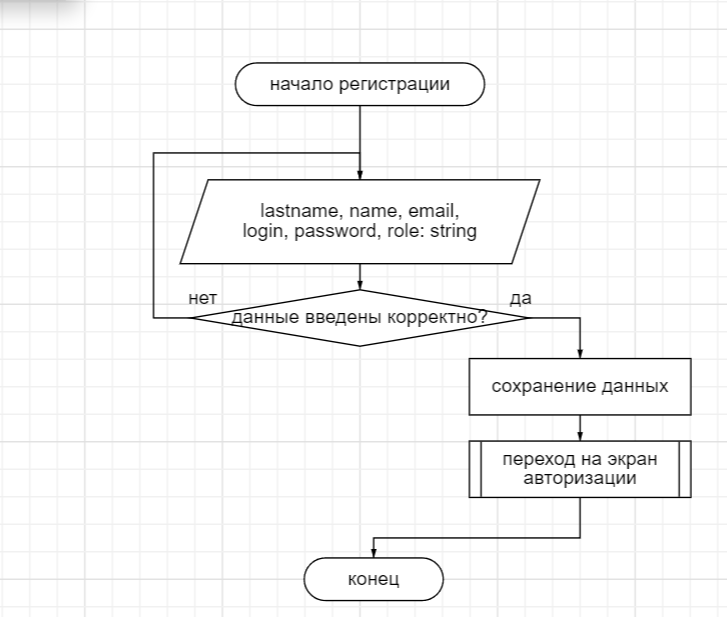
Методы класса ViewTeachersActivity:

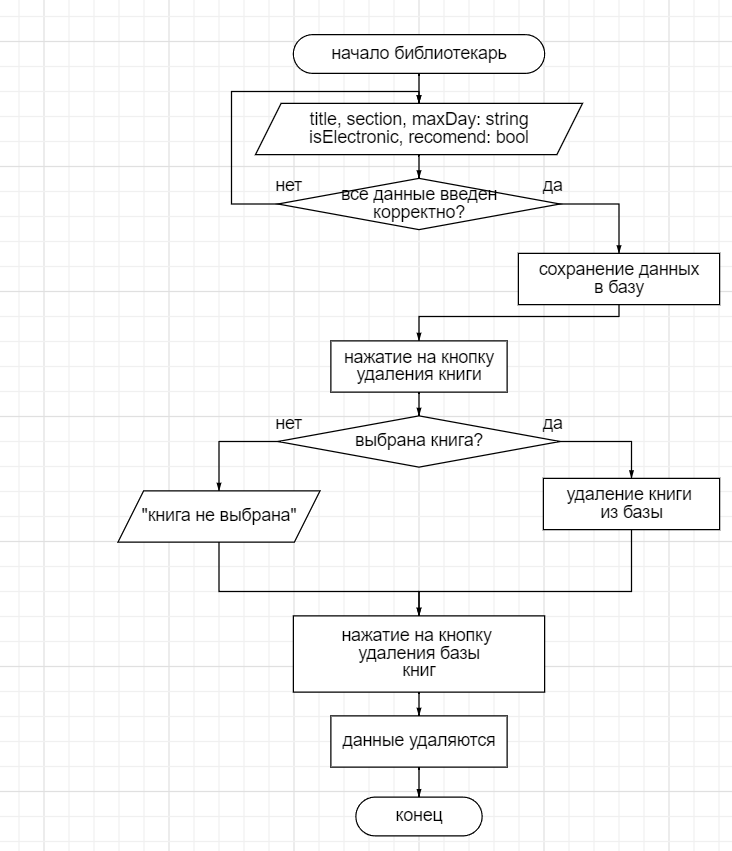
|  |  |
| --- | --- |
| loadTeachers() | Загружает список преподавателей из базы данных и обновляет адаптер. |
| searchTeachers(name: String) | Ищет преподавателей по имени и обновляет список в адаптере. |

# Блок-схемы

**Алгоритм решения:**







# Листинг программы

## AddStudentsActivity.kt

import android.annotation.SuppressLint  
import android.app.DatePickerDialog  
import androidx.appcompat.app.AppCompatActivity  
import android.os.Bundle  
import android.util.Log  
import android.view.View  
import android.widget.AdapterView  
import android.widget.ArrayAdapter  
import android.widget.Button  
import android.widget.DatePicker  
import android.widget.EditText  
import android.widget.ImageView  
import android.widget.RadioGroup  
import android.widget.Spinner  
import android.widget.TextView  
import android.widget.Toast  
import kotlinx.coroutines.CoroutineScope  
import kotlinx.coroutines.Dispatchers  
import kotlinx.coroutines.launch  
import java.util.Calendar  
  
class AddStudentsActivity : AppCompatActivity(),  
 PhotoPickerDialogFragment.OnPhotoSelectedListener {  
 private lateinit var studentDao: StudentDao  
 private lateinit var etFullName: EditText  
 private lateinit var etGroupId: EditText  
 private lateinit var etDateOfBirth: TextView  
 private lateinit var btnSaveStudent: Button  
  
 private lateinit var spinner: Spinner  
 private lateinit var tv\_user\_speciality: TextView  
 private lateinit var course: EditText  
  
 var selectedDate: String = ""  
 var selectedSpecialties: MutableSet<String> = *mutableSetOf*()  
 var budgetSpecialtySelected: Boolean = false  
 private lateinit var selectedPhotoImageView: ImageView  
 private var selectedPhoto: String? = null  
  
 @SuppressLint("MissingInflatedId", "CutPasteId")  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 setContentView(R.layout.*activity\_add\_students*)  
  
 etFullName = findViewById(R.id.*et\_full\_name*)  
 etGroupId = findViewById(R.id.*et\_group\_id*)  
 btnSaveStudent = findViewById(R.id.*btn\_save\_student*)  
 spinner = findViewById(R.id.*spinner*)  
 tv\_user\_speciality = findViewById(R.id.*tv\_user\_speciality*)  
 course = findViewById(R.id.*et\_course*)  
  
 etDateOfBirth = findViewById(R.id.*set\_date*)  
 selectedPhotoImageView = findViewById(R.id.*selected\_photo\_image\_view*)  
  
  
 studentDao = CollegeDatabase.getDatabase(*applicationContext*).studentDao()  
 tv\_user\_speciality.*text* = ""  
  
 val filmSpinner = findViewById<Spinner>(R.id.*spinner*)  
 val adapter = ArrayAdapter.createFromResource(  
 this,  
 R.array.*spec\_types*, android.R.layout.*simple\_spinner\_item* )  
 adapter.setDropDownViewResource(android.R.layout.*simple\_spinner\_dropdown\_item*)  
 filmSpinner.*adapter* = adapter  
  
 filmSpinner.*onItemSelectedListener* = object : AdapterView.OnItemSelectedListener {  
 override fun onItemSelected(  
 parent: AdapterView<\*>,  
 view: View,  
 position: Int,  
 id: Long  
 ) {  
 val selectedItem = parent.getItemAtPosition(position).toString()  
  
 if (selectedSpecialties.size < 3) {  
 if (selectedItem.*endsWith*("(бюджет)")) {  
 if (!budgetSpecialtySelected) {  
 selectedSpecialties.add(selectedItem)  
 budgetSpecialtySelected = true  
 } else {  
 Toast.makeText(  
 this@AddStudentsActivity,  
 "Вы уже выбрали бюджетную специальность",  
 Toast.*LENGTH\_SHORT* ).show()  
 return  
 }  
 } else {  
 if (!selectedSpecialties.contains(selectedItem)) {  
 selectedSpecialties.add(selectedItem)  
 } else {  
 Toast.makeText(  
 this@AddStudentsActivity,  
 "Эта специальность уже выбрана",  
 Toast.*LENGTH\_SHORT* ).show()  
 return  
 }  
 }  
 tv\_user\_speciality.*text* = selectedSpecialties.*joinToString*("\n")  
 } else {  
 Toast.makeText(  
 this@AddStudentsActivity,  
 "Вы можете выбрать максимум 3 специальности",  
 Toast.*LENGTH\_SHORT* ).show()  
 }  
 }  
  
 override fun onNothingSelected(parent: AdapterView<\*>) {  
 Toast.makeText(  
 this@AddStudentsActivity,  
 "Выберите хотя бы одну специальность",  
 Toast.*LENGTH\_SHORT* )  
 }  
 }  
  
 btnSaveStudent.setOnClickListener **{** val fullName = etFullName.*text*.toString()  
 val group1 = etGroupId.*text*.toString()  
 val courseAgain = course.*text*.toString()  
 val specialties = tv\_user\_speciality.*text*.toString()  
 val dateOfBirth = etDateOfBirth.*text*.toString()  
  
 if (selectedPhoto == null) {  
 Toast.makeText(this, "Выберите фотографию!", Toast.*LENGTH\_SHORT*).show()  
 return@setOnClickListener  
 }  
  
 *CoroutineScope*(Dispatchers.IO).*launch* **{** try {  
 val existingCount = studentDao.countStudentsByNameAndDOB(fullName, dateOfBirth)  
  
 if (existingCount > 0) {  
 runOnUiThread **{** Toast.makeText(this@AddStudentsActivity, "Такой студент уже существует!", Toast.*LENGTH\_SHORT*).show()  
 **}** return@launch  
 }  
  
 val currentCount = studentDao.getStudentCountByGroup(group1)  
 if (currentCount >= 25) {  
 runOnUiThread **{** Toast.makeText(this@AddStudentsActivity, "Лимит студентов в группе - 25 человек!", Toast.*LENGTH\_SHORT*).show()  
 **}** return@launch  
 }  
  
 val student = Student(  
 fullName = fullName,  
 group1 = group1,  
 course = courseAgain,  
 specialty = specialties,  
 dateOfBirth = dateOfBirth,  
 photoPath = selectedPhoto!!  
 )  
  
 val id = studentDao.insertStudent(student)  
 runOnUiThread **{** Toast.makeText(  
 this@AddStudentsActivity,  
 "Студент добавлен",  
 Toast.*LENGTH\_SHORT* ).show()  
 finish()  
 **}** } catch (e: Exception) {  
 Log.e("AddStudentsActivity", "Ошибка при добавлении студента", e)  
 runOnUiThread **{** Toast.makeText(  
 this@AddStudentsActivity,  
 "Ошибка при добавлении студента: ${e.message}",  
 Toast.*LENGTH\_SHORT* ).show()  
 **}** }  
 **}  
 }** findViewById<Button>(R.id.*button\_select\_photo*).setOnClickListener **{** val dialog = PhotoPickerDialogFragment()  
 dialog.show(*supportFragmentManager*, "PhotoPicker")  
 **}** }  
  
 override fun onPhotoSelected(photo: String) {  
 val resId = *resources*.getIdentifier(photo, "drawable", *packageName*)  
 if (resId != 0) {  
 selectedPhotoImageView.setImageResource(resId)  
 selectedPhoto = *resources*.getResourceEntryName(resId)  
 } else {  
 Log.e("AddStudentsActivity", "Image resource not found: $photo")  
 }  
 }  
  
 fun deleteSpeciality(view: View) {  
 tv\_user\_speciality.*text* = ""  
  
 selectedSpecialties.clear()  
 budgetSpecialtySelected = false  
  
 Toast.makeText(this, "Специальности сброшены. Выберите заново.", Toast.*LENGTH\_SHORT*)  
 .show()  
 }  
  
 fun selectedDate(view: View) {  
 val defaultYear = 2005  
 val defaultMonth = 0  
 val defaultDay = 1  
  
 val minDateCalendar = Calendar.getInstance()  
 minDateCalendar.set(1970, Calendar.*JANUARY*, 1)  
 val minDate = minDateCalendar.*timeInMillis* val maxDateCalendar = Calendar.getInstance()  
 maxDateCalendar.set(Calendar.*YEAR*, maxDateCalendar.get(Calendar.*YEAR*) - 15)  
 val maxDate = maxDateCalendar.*timeInMillis* etDateOfBirth = findViewById(R.id.*set\_date*)  
 val datePickerDialog = DatePickerDialog(  
 this,  
 **{** \_: DatePicker, selectedYear: Int, selectedMonth: Int, selectedDay: Int **->** selectedDate =  
 String.*format*("%02d-%02d-%d", selectedDay, selectedMonth + 1, selectedYear)  
 etDateOfBirth.*text* = selectedDate  
 **}**,  
 defaultYear,  
 defaultMonth,  
 defaultDay  
 )  
  
 datePickerDialog.*datePicker*.*minDate* = minDate  
 datePickerDialog.*datePicker*.*maxDate* = maxDate  
 datePickerDialog.show()  
 }

## AddTeacherActivity.kt

import android.annotation.SuppressLint  
import androidx.appcompat.app.AppCompatActivity  
import android.os.Bundle  
import android.util.Log  
import android.widget.Button  
import android.widget.EditText  
import android.widget.ImageView  
import android.widget.Toast  
import kotlinx.coroutines.CoroutineScope  
import kotlinx.coroutines.Dispatchers  
import kotlinx.coroutines.launch  
  
class AddTeacherActivity : AppCompatActivity(),  
 PhotoPickerDialogFragment.OnPhotoSelectedListener {  
 private lateinit var teacherDao: TeacherDao  
 private lateinit var selectedPhotoImageView: ImageView  
 private var selectedPhoto: String? = null  
  
 @SuppressLint("MissingInflatedId")  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 setContentView(R.layout.*activity\_add\_teacher*)  
  
 val etTeacherFullName: EditText = findViewById(R.id.*et\_teacher\_full\_name*)  
 val etSpecialtyId: EditText = findViewById(R.id.*et\_specialty\_id*)  
 val etWorkload: EditText = findViewById(R.id.*et\_workload*)  
 val btnSaveTeacher: Button = findViewById(R.id.*btn\_save\_teacher*)  
  
 selectedPhotoImageView = findViewById(R.id.*selected\_photo\_image\_view1*)  
 teacherDao = CollegeDatabase.getDatabase(*applicationContext*).teacherDao()  
  
 btnSaveTeacher.setOnClickListener **{** val teacherFullName = etTeacherFullName.*text*.toString()  
 val specialtyId = etSpecialtyId.*text*.toString()  
 val workload = etWorkload.*text*.toString().*toInt*()  
  
 val teacher = Teacher(fullName = teacherFullName, specialty = specialtyId, workload = workload, photoPath = selectedPhoto!! )  
 *CoroutineScope*(Dispatchers.IO).*launch* **{** teacherDao.insertTeacher(teacher)  
 runOnUiThread **{** Toast.makeText(this@AddTeacherActivity, "Преподаватель добавлен", Toast.*LENGTH\_SHORT*).show()  
 finish()  
 **}  
 }  
 }** findViewById<Button>(R.id.*button\_select\_photo1*).setOnClickListener **{** val dialog = PhotoPickerDialogFragment()  
 dialog.show(*supportFragmentManager*, "PhotoPicker")  
 **}** }  
  
 override fun onPhotoSelected(photo: String) {  
 val resId = *resources*.getIdentifier(photo, "drawable", *packageName*)  
 if (resId != 0) {  
 selectedPhotoImageView.setImageResource(resId)  
 selectedPhoto = *resources*.getResourceEntryName(resId)  
 } else {  
 Log.e("AddStudentsActivity", "Image resource not found: $photo")  
 }  
 }

## Auth.kt

import android.content.Intent  
import android.content.SharedPreferences  
import android.os.Bundle  
import android.view.View  
import android.widget.Button  
import android.widget.EditText  
import android.widget.Spinner  
import android.widget.Toast.*LENGTH\_LONG*import androidx.appcompat.app.AlertDialog  
import androidx.appcompat.app.AppCompatActivity  
import com.google.android.material.snackbar.Snackbar  
import com.google.android.material.snackbar.Snackbar.make  
  
class Auth : AppCompatActivity() {  
 private lateinit var email: EditText  
 private lateinit var pass: EditText  
 private lateinit var spinner: Spinner  
 private lateinit var pref: SharedPreferences  
 private lateinit var button: Button  
  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 setContentView(R.layout.*activity\_auth*)  
 pref = getPreferences(*MODE\_PRIVATE*)  
  
 email = findViewById(R.id.*email*);  
 pass = findViewById(R.id.*password*);  
 spinner = findViewById(R.id.*spinner*)  
 button = findViewById(R.id.*auth\_button*)  
  
 button.setOnClickListener(View.OnClickListener **{** if (email.*text*.toString().*isNotEmpty*() && pass.*text*.toString().*isNotEmpty*()) {  
 if (pass.*text*.toString().length >= 8) {  
 when(spinner.*selectedItem*.toString())  
 {  
 "Преподаватель" -> {  
  
 if (((pref.getString("emailTeacher", "")) == "") && ((pref.getString("passwordTeacher", "")) == "")  
 ) {  
 val ed = pref.edit()  
 ed.putString("emailTeacher", email.*text*.toString())  
 ed.putString("passwordTeacher", pass.*text*.toString())  
 ed.apply()  
  
 val intent = Intent(this, ViewStudentsActivity::class.*java*)  
 startActivity(intent)  
  
 } else if (((pref.getString(  
 "emailTeacher",  
 ""  
 )) == email.*text*.toString()) && ((pref.getString(  
 "passwordTeacher",  
 ""  
 )) == pass.*text*.toString())  
 ) {  
  
 val intent = Intent(this, ViewStudentsActivity::class.*java*)  
 startActivity(intent)  
 } else {  
 val alert = AlertDialog.Builder(this)  
 .setTitle("Неверно")  
 .setMessage("Данные введены неверно")  
 .setPositiveButton("Попробовать еще раз",null)  
 .create()  
 .show()  
 }  
 }  
 "Студент" -> {  
  
 if (((pref.getString("emailStudent", "")) == "") && ((pref.getString("passwordStudent", "")) == "")  
 ) {  
 val ed = pref.edit()  
 ed.putString("emailStudent", email.*text*.toString())  
 ed.putString("passwordStudent", pass.*text*.toString())  
 ed.apply()  
  
 val intent = Intent(this, ViewTeachersActivity::class.*java*)  
 startActivity(intent)  
  
 } else if (((pref.getString(  
 "emailStudent",  
 ""  
 )) == email.*text*.toString()) && ((pref.getString(  
 "passwordStudent",  
 ""  
 )) == pass.*text*.toString())  
 ) {  
  
 val intent = Intent(this, ViewTeachersActivity::class.*java*)  
 startActivity(intent)  
  
 } else {  
 val alert = AlertDialog.Builder(this)  
 .setTitle("Неверно")  
 .setMessage("Данные введены неверно")  
 .setPositiveButton("Попробовать еще раз",null)  
 .create()  
 .show()  
 }  
 }  
 "Приёмная комиссия" -> {  
 if (((pref.getString("emailComission", "")) == "") && ((pref.getString("passwordComission", "")) == "")  
 ) {  
 val ed = pref.edit()  
 ed.putString("emailComission", email.*text*.toString())  
 ed.putString("passwordComission", pass.*text*.toString())  
 ed.apply()  
  
 val intent = Intent(this, MainActivity::class.*java*)  
 startActivity(intent)  
  
 } else if (((pref.getString(  
 "emailComission",  
 ""  
 )) == email.*text*.toString()) && ((pref.getString(  
 "passwordComission",  
 ""  
 )) == pass.*text*.toString())  
 ) {  
 val intent = Intent(this, MainActivity::class.*java*)  
 startActivity(intent)  
 } else {  
 val alert = AlertDialog.Builder(this)  
 .setTitle("Ошибка")  
 .setMessage("Данные введены неверно")  
 .setPositiveButton("Попробовать еще раз",null)  
 .create()  
 .show()  
 }  
 }  
 }  
  
 } else {  
  
 val alert = AlertDialog.Builder(this)  
 .setTitle("Ошибка")  
 .setMessage("Пароль должен быть не менее 8 символов")  
 .setPositiveButton("Попробовать еще раз",null)  
 .create()  
 .show()  
 }  
  
 } else {  
  
 val alert = AlertDialog.Builder(this)  
 .setTitle("Ошибка")  
 .setMessage("Введите почту и пароль")  
 .setPositiveButton("Попробовать еще раз",null)  
 .create()  
 .show()  
  
 }  
 **}**)  
 }  
}

## DataClasses.kt

import androidx.room.Entity  
import androidx.room.PrimaryKey  
  
@Entity(tableName = "students")  
data class Student(  
 @PrimaryKey(autoGenerate = true) val id: Long = 0,  
 val fullName: String,  
 val group1: String,  
 val course: String,  
 val specialty: String,  
 val dateOfBirth: String,  
 val photoPath: String  
){  
 override fun toString(): String {  
 return "$id. $fullName, $group1, курс: $course. №Специальности: $specialty. Дата рождения: $dateOfBirth"  
 }  
}  
  
@Entity(tableName = "teachers")  
data class Teacher(  
 @PrimaryKey(autoGenerate = true) val teacherId: Long = 0,  
 val fullName: String,  
 val specialty: String,  
 val workload: Int,  
 val photoPath: String  
)

## Interfaces.kt

import androidx.room.Dao  
import androidx.room.Delete  
import androidx.room.Insert  
import androidx.room.Query  
import androidx.room.Update  
  
@Dao  
interface StudentDao {  
 @Insert  
 suspend fun insertStudent(student: Student): Long  
  
 @Delete  
 suspend fun deleteStudent(student: Student): Int  
  
 @Update  
 fun updateStudent(student: Student): Int  
  
 @Query("SELECT \* FROM students WHERE fullName LIKE :name")  
 suspend fun searchStudentsByName(name: String): MutableList<Student>  
  
 @Query("SELECT \* FROM students")  
 suspend fun getAllStudentsByGroup(): MutableList<Student>  
  
 @Query("DELETE FROM students WHERE fullName = :name")  
 suspend fun deleteStudentByName(name: String): Int  
  
 @Query("SELECT COUNT(\*) FROM students WHERE group1 = :groupId")  
 fun getStudentCountByGroup(groupId: String): Int  
  
 @Query("SELECT COUNT(\*) FROM students WHERE fullName = :fullName AND dateOfBirth = :dateOfBirth")  
 fun countStudentsByNameAndDOB(fullName: String, dateOfBirth: String): Int  
}  
  
@Dao  
interface TeacherDao {  
 @Insert  
 suspend fun insertTeacher(teacher: Teacher)  
  
 @Update  
 suspend fun updateTeacher(teacher: Teacher)  
  
 @Delete  
 suspend fun deleteTeacher(teacher: Teacher)  
  
 @Query("SELECT \* FROM teachers WHERE fullName LIKE :name")  
 suspend fun searchTeachersByName(name: String): MutableList<Teacher>  
  
 @Query("SELECT \* FROM teachers")  
 suspend fun getAllTeachersBySpecialty(): MutableList<Teacher>  
}

## MainActivity.kt

import android.content.Intent  
import androidx.appcompat.app.AppCompatActivity  
import android.os.Bundle  
import android.widget.Button  
  
class MainActivity : AppCompatActivity() {  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 setContentView(R.layout.*activity\_main*)  
  
 val btnAddStudent: Button = findViewById(R.id.*btn\_add\_student*)  
 val btnViewStudents: Button = findViewById(R.id.*btn\_view\_students*)  
 val btnAddTeacher: Button = findViewById(R.id.*btn\_add\_teacher*)  
 val btnStatistics: Button = findViewById(R.id.*btn\_statistics*)  
  
 btnAddStudent.setOnClickListener **{** startActivity(Intent(this, AddStudentsActivity::class.*java*))  
 **}** btnViewStudents.setOnClickListener **{** startActivity(Intent(this, ViewStudentsActivity::class.*java*))  
 **}** btnAddTeacher.setOnClickListener **{** startActivity(Intent(this, AddTeacherActivity::class.*java*))  
 **}** btnStatistics.setOnClickListener **{** startActivity(Intent(this, StatisticsActivity::class.*java*))  
 **}** }  
}

## PhotoAdapter.kt

class PhotoAdapter(private val photos: List<Int>, private val onClick: (Int) -> Unit) :  
 RecyclerView.Adapter<PhotoAdapter.PhotoViewHolder>() {  
  
 class PhotoViewHolder(itemView: View) : RecyclerView.ViewHolder(itemView) {  
 val imageView: ImageView = itemView.findViewById(R.id.*photo\_image\_view*)  
 }  
  
 override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): PhotoViewHolder {  
 val view = LayoutInflater.from(parent.*context*).inflate(R.layout.*item\_photo*, parent, false)  
 return PhotoViewHolder(view)  
 }  
  
 override fun onBindViewHolder(holder: PhotoViewHolder, position: Int) {  
 val photo = photos[position]  
 holder.imageView.setImageResource(photo)  
  
 holder.itemView.setOnClickListener **{** onClick(photo)  
 **}** }  
  
 override fun getItemCount(): Int {  
 return photos.size  
 }  
}

## PhotoPickerDialogFragment.kt

import android.app.AlertDialog  
import android.app.Dialog  
import android.content.Context  
import android.os.Bundle  
import androidx.fragment.app.DialogFragment  
import androidx.recyclerview.widget.GridLayoutManager  
import androidx.recyclerview.widget.RecyclerView  
  
class PhotoPickerDialogFragment : DialogFragment() {  
  
 private var listener: OnPhotoSelectedListener? = null  
  
 interface OnPhotoSelectedListener {  
 fun onPhotoSelected(photo: String)  
 }  
  
 override fun onAttach(context: Context) {  
 super.onAttach(context)  
 listener = context as? OnPhotoSelectedListener  
 }  
  
 override fun onCreateDialog(savedInstanceState: Bundle?): Dialog {  
 val builder = AlertDialog.Builder(requireActivity())  
 val recyclerView = RecyclerView(requireContext())  
 recyclerView.*layoutManager* = GridLayoutManager(requireContext(), 3)  
  
 val photos = *listOf*(  
 R.drawable.*photo1*,  
 R.drawable.*photo2*,  
 R.drawable.*photo3*,  
 R.drawable.*photo4*,  
 R.drawable.*photo5* )  
  
 val adapter = PhotoAdapter(photos) **{** photoResId **->** val photoName = *resources*.getResourceEntryName(photoResId)  
 listener?.onPhotoSelected(photoName)  
 dismiss()  
 **}** recyclerView.*adapter* = adapter  
  
 builder.setView(recyclerView)  
 .setTitle("Выберите фотографию")  
 .setNegativeButton("Отмена") **{** dialog, \_ **->** dialog.dismiss() **}** return builder.create()  
 }  
}

## StatistickActivity.kt

import androidx.appcompat.app.AppCompatActivity  
import android.os.Bundle  
import android.widget.ArrayAdapter  
import android.widget.ListView  
import kotlinx.coroutines.CoroutineScope  
import kotlinx.coroutines.Dispatchers  
import kotlinx.coroutines.launch  
import kotlinx.coroutines.withContext  
  
class StatisticsActivity : AppCompatActivity() {  
 private lateinit var teacherDao: TeacherDao  
 private lateinit var studentDao: StudentDao  
 private lateinit var statisticsAdapter: ArrayAdapter<String>  
  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 setContentView(R.layout.*activity\_statistics*)  
  
 val lvStatistics: ListView = findViewById(R.id.*lv\_statistics*)  
 statisticsAdapter = ArrayAdapter(this, android.R.layout.*simple\_list\_item\_1*, *mutableListOf*())  
 lvStatistics.*adapter* = statisticsAdapter  
  
 studentDao = CollegeDatabase.getDatabase(*applicationContext*).studentDao()  
 teacherDao = CollegeDatabase.getDatabase(*applicationContext*).teacherDao()  
  
 loadStatistics()  
 }  
  
 private fun loadStatistics() {  
 *CoroutineScope*(Dispatchers.IO).*launch* **{** val studentCount = studentDao.getAllStudentsByGroup().size  
 val teacherCount = teacherDao.getAllTeachersBySpecialty().size  
 val statistics = *listOf*(  
 "Количество студентов: $studentCount",  
 "Количество преподавателей: $teacherCount"  
 )  
  
 withContext(Dispatchers.Main) **{** statisticsAdapter.clear()  
 statisticsAdapter.addAll(statistics)  
 statisticsAdapter.notifyDataSetChanged()  
 **}  
 }** }  
}

## StudentAdapter.kt

import android.view.LayoutInflater  
import android.view.View  
import android.view.ViewGroup  
import android.widget.ImageView  
import android.widget.TextView  
import androidx.recyclerview.widget.RecyclerView  
  
class StudentAdapter(private val students: List<Student>, private val itemClickListener: (Int) -> Unit) : RecyclerView.Adapter<StudentAdapter.StudentViewHolder>() {  
  
 inner class StudentViewHolder(view: View) : RecyclerView.ViewHolder(view) {  
 val studentName: TextView = view.findViewById(R.id.*tvStudentName*)  
 val groupId: TextView = view.findViewById(R.id.*tvGroupId*)  
 val course: TextView = view.findViewById(R.id.*tvCourse*)  
 val specialty: TextView = view.findViewById(R.id.*tvSpecialty*)  
 val dateOfBirthday: TextView = view.findViewById(R.id.*tvDateOfBirth*)  
 val photoImageView: ImageView = itemView.findViewById(R.id.*photo\_image\_view*)  
 init {  
 itemView.setOnClickListener **{** itemClickListener(*adapterPosition*)  
 **}** }  
 }  
  
 override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): StudentViewHolder {  
 val view = LayoutInflater.from(parent.*context*).inflate(R.layout.*item\_student*, parent, false)  
 return StudentViewHolder(view)  
 }  
  
 override fun onBindViewHolder(holder: StudentViewHolder, position: Int) {  
 val student = students[position]  
 holder.studentName.*text* = student.fullName  
 holder.groupId.*text* = student.group1  
 holder.course.*text* = student.course  
 holder.specialty.*text* = student.specialty  
 holder.dateOfBirthday.*text* = student.dateOfBirth  
  
 val resId = holder.itemView.*context*.*resources*.getIdentifier(student.photoPath, "drawable", holder.itemView.*context*.*packageName*)  
 if (resId != 0) {  
 holder.photoImageView.setImageResource(resId)  
 } else {  
 holder.photoImageView.setImageResource(R.drawable.*defoult*)  
 }  
 }  
  
 override fun getItemCount(): Int {  
 return students.size  
 }  
}

## StudentsClasses.kt

import android.content.Context  
import androidx.room.Database  
import androidx.room.Room  
import androidx.room.RoomDatabase  
  
@Database(entities = [Student::class, Teacher::class], version = 3)  
abstract class CollegeDatabase : RoomDatabase() {  
 abstract fun studentDao(): StudentDao  
 abstract fun teacherDao(): TeacherDao  
  
 companion object {  
 @Volatile  
 private var instance: CollegeDatabase? = null  
  
 fun getDatabase(context: Context): CollegeDatabase {  
 return instance ?: *synchronized*(this) **{** val db = Room.databaseBuilder(  
 context.*applicationContext*,  
 CollegeDatabase::class.*java*,  
 "college\_database"  
 ).build()  
 instance = db  
 db  
 **}** }  
 }  
}

## TeacherAdapter.kt

import android.view.LayoutInflater  
import android.view.View  
import android.view.ViewGroup  
import android.widget.ImageView  
import android.widget.TextView  
import androidx.recyclerview.widget.RecyclerView  
  
class TeacherAdapter(private val teachers: List<Teacher>, private val itemClickListener: (Int) -> Unit) : RecyclerView.Adapter<TeacherAdapter.TeacherViewHolder>() {  
  
 inner class TeacherViewHolder(view: View) : RecyclerView.ViewHolder(view) {  
 val teacherName: TextView = view.findViewById(R.id.*tvTeacherName*)  
 val specialty: TextView = view.findViewById(R.id.*tvTeacherSpecialty*)  
 val workload:TextView = view.findViewById(R.id.*tvTeacherWorkload*)  
 val photoImageView: ImageView = itemView.findViewById(R.id.*photo\_image\_view1*)  
 init {  
 itemView.setOnClickListener **{** itemClickListener(*adapterPosition*)  
 **}** }  
 }  
  
 override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): TeacherViewHolder {  
 val view = LayoutInflater.from(parent.*context*).inflate(R.layout.*item\_teacher*, parent, false)  
 return TeacherViewHolder(view)  
 }  
  
 override fun onBindViewHolder(holder: TeacherViewHolder, position: Int) {  
 val teacher = teachers[position]  
 holder.teacherName.*text* = teacher.fullName  
 holder.specialty.*text* = teacher.specialty  
 holder.workload.*text* = teacher.workload.toString()  
  
 val resId = holder.itemView.*context*.*resources*.getIdentifier(teacher.photoPath, "drawable", holder.itemView.*context*.*packageName*)  
 if (resId != 0) {  
 holder.photoImageView.setImageResource(resId)  
 } else {  
 holder.photoImageView.setImageResource(R.drawable.*defoult*)  
 }  
 }  
  
 override fun getItemCount(): Int {  
 return teachers.size  
 }  
}

## ViewStudentsActivity.kt

import android.annotation.SuppressLint  
import android.app.DatePickerDialog  
import androidx.appcompat.app.AppCompatActivity  
import android.os.Bundle  
import android.util.Log  
import android.view.View  
import android.widget.ArrayAdapter  
import android.widget.Button  
import android.widget.DatePicker  
import android.widget.EditText  
import android.widget.ListView  
import android.widget.SearchView  
import android.widget.TextView  
import android.widget.Toast  
import androidx.recyclerview.widget.LinearLayoutManager  
import androidx.recyclerview.widget.RecyclerView  
import kotlinx.coroutines.CoroutineScope  
import kotlinx.coroutines.Dispatchers  
import kotlinx.coroutines.launch  
import kotlinx.coroutines.withContext  
import java.text.ParseException  
import java.text.SimpleDateFormat  
import java.util.Calendar  
import java.util.Locale  
  
class ViewStudentsActivity : AppCompatActivity() {  
 private lateinit var studentDao: StudentDao  
 private lateinit var studentAdapter: StudentAdapter  
  
 private lateinit var nameEditText: EditText  
 private lateinit var groupEditText: EditText  
 private lateinit var specialityEditText: EditText  
 private lateinit var courseEditTExt:EditText  
  
 private lateinit var editButton: Button  
 private lateinit var deleteButton: Button  
 private lateinit var etDateOfBirth: TextView  
 private lateinit var selDatBtn: Button  
 private var currentPosition: Int = -1  
 private var students: MutableList<Student> = *mutableListOf*()  
 var selectedDate: String = ""  
  
 @SuppressLint("MissingInflatedId")  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 setContentView(R.layout.*activity\_view\_students*)  
  
 nameEditText = findViewById(R.id.*name*)  
 groupEditText = findViewById(R.id.*group*)  
 specialityEditText = findViewById(R.id.*speciality*)  
 courseEditTExt = findViewById(R.id.*course*)  
  
 editButton = findViewById(R.id.*edit*)  
 deleteButton = findViewById(R.id.*delete*)  
 etDateOfBirth = findViewById(R.id.*set\_date\_view*)  
 selDatBtn = findViewById(R.id.*selectedDateBtn*)  
  
 val searchView: SearchView = findViewById(R.id.*search\_view*)  
 val recyclerView: RecyclerView = findViewById(R.id.*recycler\_view*)  
  
 studentDao = CollegeDatabase.getDatabase(*applicationContext*).studentDao()  
 recyclerView.*layoutManager* = LinearLayoutManager(this)  
 loadStudents()  
  
 studentAdapter = StudentAdapter(students) **{** position **->** currentPosition = position  
 val selectedItem = students[position]  
  
 nameEditText.setText(selectedItem.fullName)  
 groupEditText.setText(selectedItem.group1)  
 specialityEditText.setText(selectedItem.specialty)  
 courseEditTExt.setText(selectedItem.course)  
 **}** recyclerView.*adapter* = studentAdapter  
  
 selDatBtn.setOnClickListener**{** val defaultYear = 2005  
 val defaultMonth = 0  
 val defaultDay = 1  
  
 val minDateCalendar = Calendar.getInstance()  
 minDateCalendar.set(1970, Calendar.*JANUARY*, 1)  
 val minDate = minDateCalendar.*timeInMillis* val maxDateCalendar = Calendar.getInstance()  
 maxDateCalendar.set(Calendar.*YEAR*, maxDateCalendar.get(Calendar.*YEAR*) - 15)  
 val maxDate = maxDateCalendar.*timeInMillis* etDateOfBirth = findViewById(R.id.*set\_date\_view*)  
 val datePickerDialog = DatePickerDialog(this, **{** \_: DatePicker, selectedYear: Int, selectedMonth: Int, selectedDay: Int **->** selectedDate = String.*format*("%02d-%02d-%d", selectedDay, selectedMonth + 1, selectedYear)  
 etDateOfBirth.*text* = selectedDate  
 **}**, defaultYear, defaultMonth, defaultDay)  
  
 datePickerDialog.*datePicker*.*minDate* = minDate  
 datePickerDialog.*datePicker*.*maxDate* = maxDate  
 datePickerDialog.show()  
 **}** editButton.setOnClickListener **{** if (currentPosition != -1) {  
 val selectedStudent = students[currentPosition]  
  
 val updatedName = nameEditText.*text*.toString()  
 val updatedGroup = groupEditText.*text*.toString()  
 val updatedSpecialty = specialityEditText.*text*.toString()  
 val updatedDate = selectedDate  
  
 if (updatedName.*isBlank*() || updatedGroup.*isBlank*() || updatedSpecialty == null) {  
 Toast.makeText(this, "Пожалуйста, заполните все поля корректно", Toast.*LENGTH\_SHORT*).show()  
 return@setOnClickListener  
 }  
  
 val updatedStudent = selectedStudent.copy(  
 fullName = updatedName,  
 group1 = updatedGroup,  
 specialty = updatedSpecialty,  
 dateOfBirth = updatedDate  
 )  
  
 Thread **{** studentDao.updateStudent(updatedStudent)  
  
 runOnUiThread **{** students[currentPosition] = updatedStudent  
 studentAdapter.notifyItemChanged(currentPosition)  
 Toast.makeText(this, "Данные обновлены", Toast.*LENGTH\_SHORT*).show()  
 **}  
 }**.start()  
 } else {  
 Toast.makeText(this, "Выберите студента для редактирования", Toast.*LENGTH\_SHORT*).show()  
 }  
 **}** deleteButton.setOnClickListener **{** val nameToDelete = nameEditText.*text*.toString()  
 deleteStudentByName(nameToDelete)  
 **}** searchView.setOnQueryTextListener(object : SearchView.OnQueryTextListener {  
 override fun onQueryTextSubmit(query: String?): Boolean {  
 query?.*let* **{** searchStudents(**it**)  
 **}** return true  
 }  
  
 override fun onQueryTextChange(newText: String?): Boolean {  
 if (newText.*isNullOrEmpty*()) {  
 loadStudents()  
 }  
 return true  
 }  
 })  
 }  
  
 private fun deleteStudentByName(name: String) {  
 *CoroutineScope*(Dispatchers.IO).*launch* **{** try {  
 val deletedRows = studentDao.deleteStudentByName(name) // Удаляем студента по имени  
 if (deletedRows > 0) {  
 withContext(Dispatchers.Main) **{** Toast.makeText(this@ViewStudentsActivity, "Студент удален", Toast.*LENGTH\_SHORT*).show()  
 loadStudents()  
 **}** } else {  
 withContext(Dispatchers.Main) **{** Toast.makeText(this@ViewStudentsActivity, "Студент не найден", Toast.*LENGTH\_SHORT*).show()  
 **}** }  
 } catch (e: Exception) {  
 Log.e("EditStudentActivity", "Ошибка при удалении студента", e)  
 withContext(Dispatchers.Main) **{** Toast.makeText(this@ViewStudentsActivity, "Ошибка при удалении студента: ${e.message}", Toast.*LENGTH\_SHORT*).show()  
 **}** }  
 **}** }  
 @SuppressLint("NotifyDataSetChanged")  
 private fun loadStudents() {  
 *CoroutineScope*(Dispatchers.IO).*launch* **{** val newStudents = studentDao.getAllStudentsByGroup() // Получаем студентов  
 withContext(Dispatchers.Main) **{** students.clear() // Очищаем текущий список  
 students.addAll(newStudents) // Добавляем новых студентов  
 studentAdapter.notifyDataSetChanged() // Уведомляем адаптер об изменениях  
 **}  
 }** }  
  
 @SuppressLint("NotifyDataSetChanged")  
 private fun searchStudents(name: String) {  
 *CoroutineScope*(Dispatchers.IO).*launch* **{** val result = studentDao.searchStudentsByName("%$name%")  
 withContext(Dispatchers.Main) **{** students.clear()  
 students.addAll(result)  
 studentAdapter.notifyDataSetChanged()  
 **}  
 }** }  
}

## ViewTeachersActivity.kt

import android.annotation.SuppressLint  
import androidx.appcompat.app.AppCompatActivity  
import android.os.Bundle  
import android.widget.EditText  
import androidx.appcompat.widget.SearchView  
import androidx.recyclerview.widget.LinearLayoutManager  
import androidx.recyclerview.widget.RecyclerView  
import kotlinx.coroutines.CoroutineScope  
import kotlinx.coroutines.Dispatchers  
import kotlinx.coroutines.launch  
import kotlinx.coroutines.withContext  
  
class ViewTeachersActivity : AppCompatActivity() {  
  
 private lateinit var teacherDao: TeacherDao  
 private lateinit var teacherAdapter: TeacherAdapter  
 private var currentPosition: Int = -1  
 private var teachers: MutableList<Teacher> = *mutableListOf*()  
 @SuppressLint("MissingInflatedId")  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 setContentView(R.layout.*activity\_view\_teachers*)  
  
 val searchView: android.widget.SearchView = findViewById(R.id.*search\_view1*)  
 val recyclerView: RecyclerView = findViewById(R.id.*recycler\_view1*)  
 teacherDao = CollegeDatabase.getDatabase(*applicationContext*).teacherDao()  
 recyclerView.*layoutManager* = LinearLayoutManager(this)  
 loadTeachers()  
  
 teacherAdapter = TeacherAdapter(teachers) **{** position **->** currentPosition = position  
 val selectedItem = teachers[position]  
 **}** recyclerView.*adapter* = teacherAdapter  
  
 searchView.setOnQueryTextListener(object : android.widget.SearchView.OnQueryTextListener {  
 override fun onQueryTextSubmit(query: String?): Boolean {  
 query?.*let* **{** searchTeachers(**it**)  
 **}** return true  
 }  
  
 override fun onQueryTextChange(newText: String?): Boolean {  
 if (newText.*isNullOrEmpty*()) {  
 loadTeachers()  
 }  
 return true  
 }  
 })  
 }  
 @SuppressLint("NotifyDataSetChanged")  
 private fun loadTeachers() {  
 *CoroutineScope*(Dispatchers.IO).*launch* **{** val newTeacher = teacherDao.getAllTeachersBySpecialty()  
 withContext(Dispatchers.Main) **{** teachers.clear()  
 teachers.addAll(newTeacher)  
 teacherAdapter.notifyDataSetChanged()  
 **}  
 }** }  
  
 @SuppressLint("NotifyDataSetChanged")  
 private fun searchTeachers(name: String) {  
 *CoroutineScope*(Dispatchers.IO).*launch* **{** val result = teacherDao.searchTeachersByName("%$name%")  
 withContext(Dispatchers.Main) **{** teachers.clear()  
 teachers.addAll(result)  
 teacherAdapter.notifyDataSetChanged()  
 **}  
 }** }  
}

# Тестовые ситуации

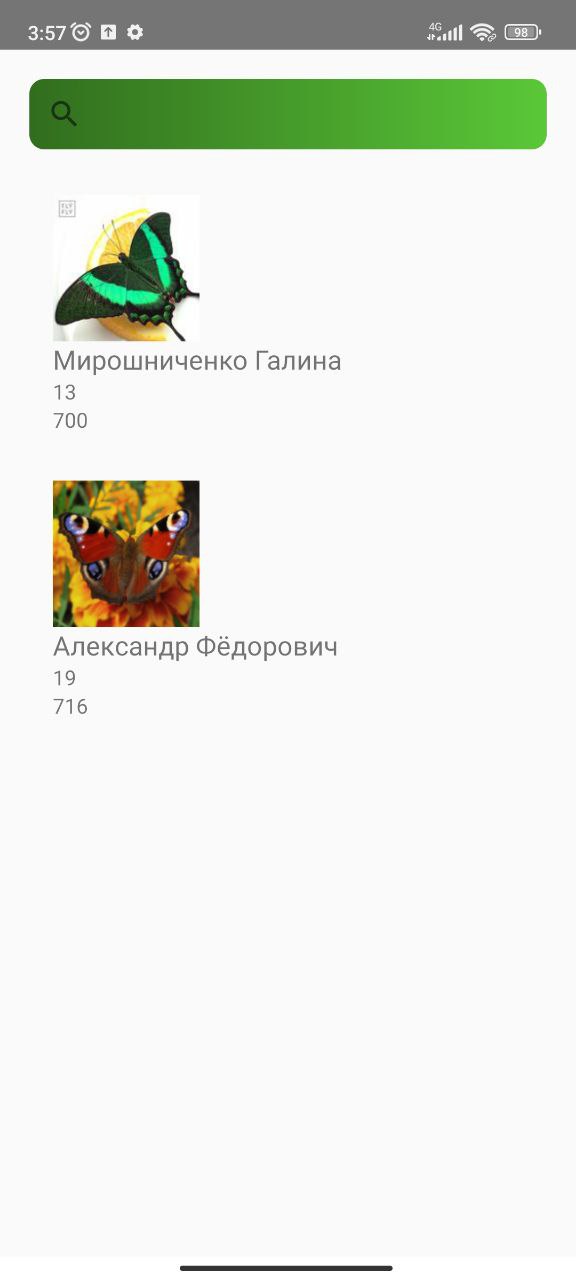
1. **Экран авторизации. При вводе данных в первый раз происходит регистрация.**

|  |  |
| --- | --- |
|  |  |

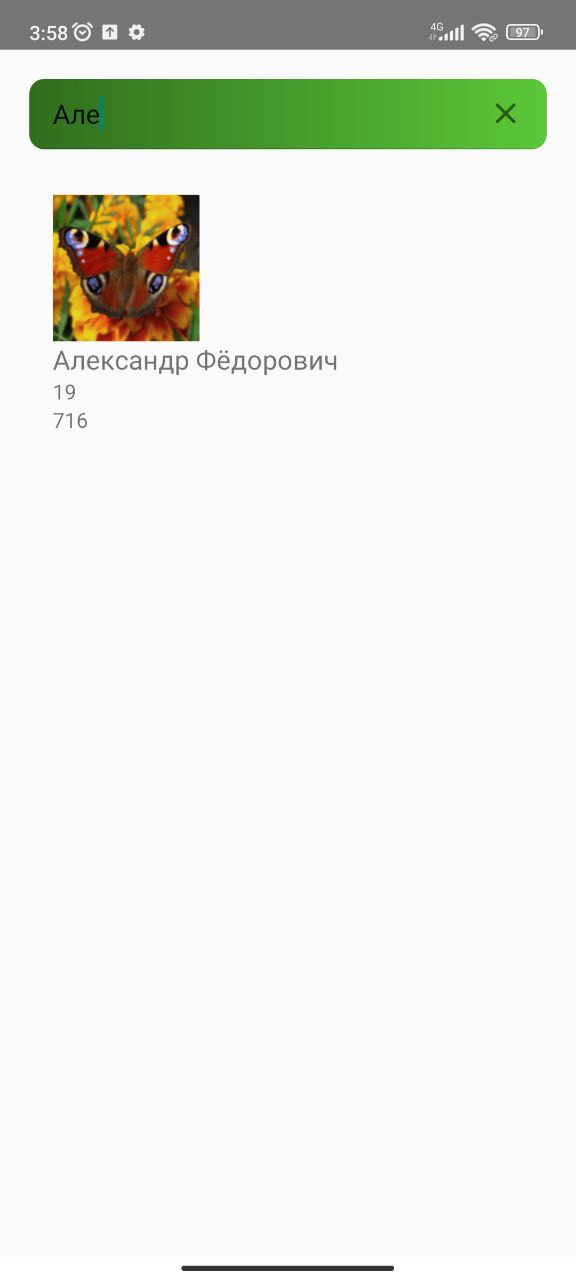
1. **Проверка, верно ли введены данные пользователя.**

|  |  |
| --- | --- |
|  |  |

1. **Экран студента. Выводится список учителей, хранимый в базе данных.**

****

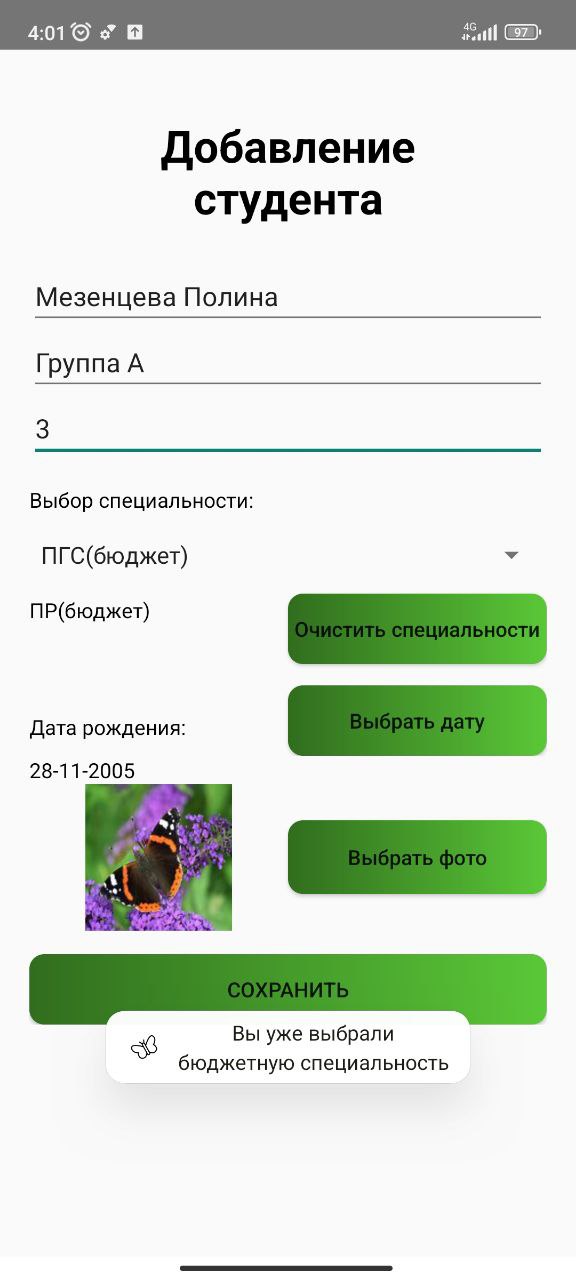
1. **Поиск учителей. Учитель/я нашлись.**

****

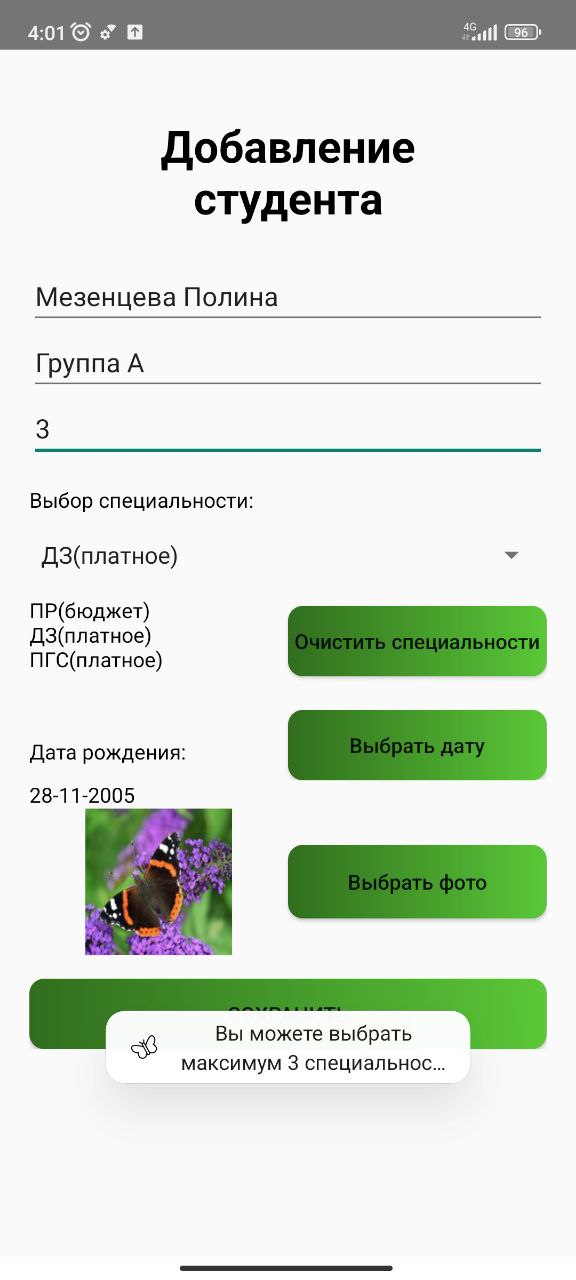
1. **Поиск учителей. Учитель/я не нашлись.**

****

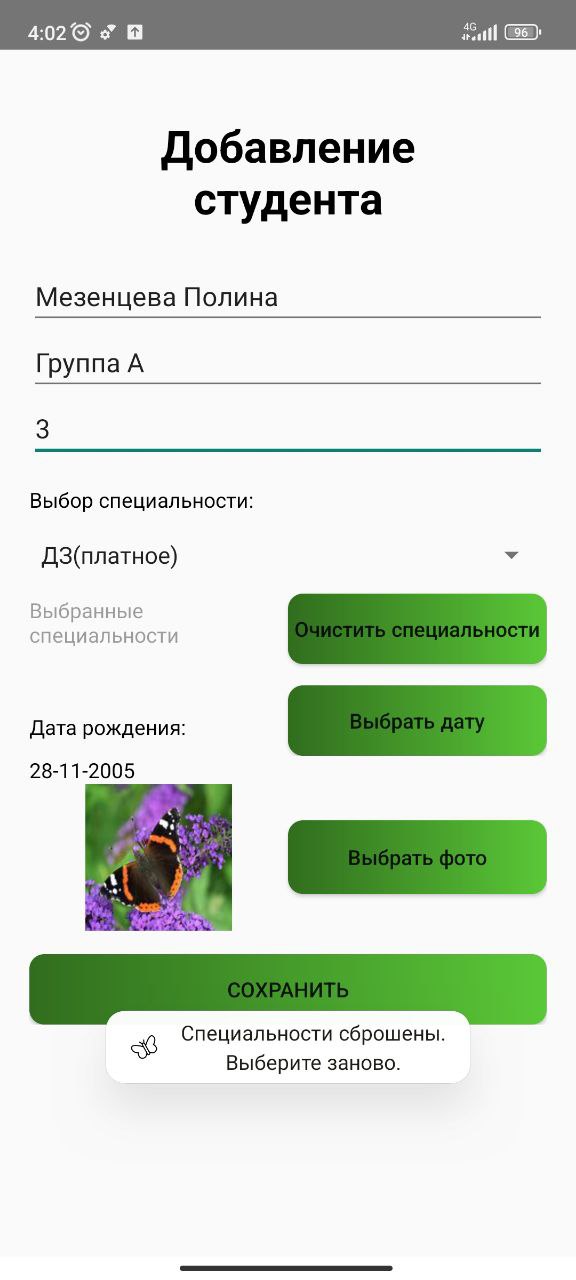
1. **Экран учителя (Экран комиссии -> Добавление студента). Добавление студента в базу данных. При выборе специальности выбрали больше одной бюджетной.**

****

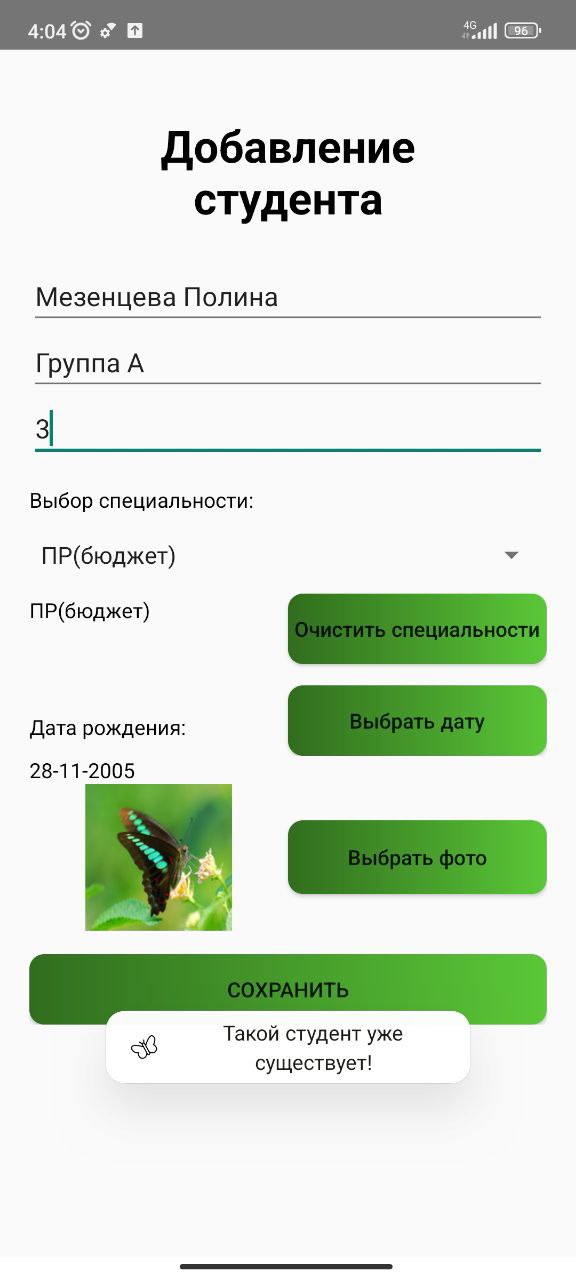
1. **Максимум можно выбрать не больше 3 специальностей.**

****

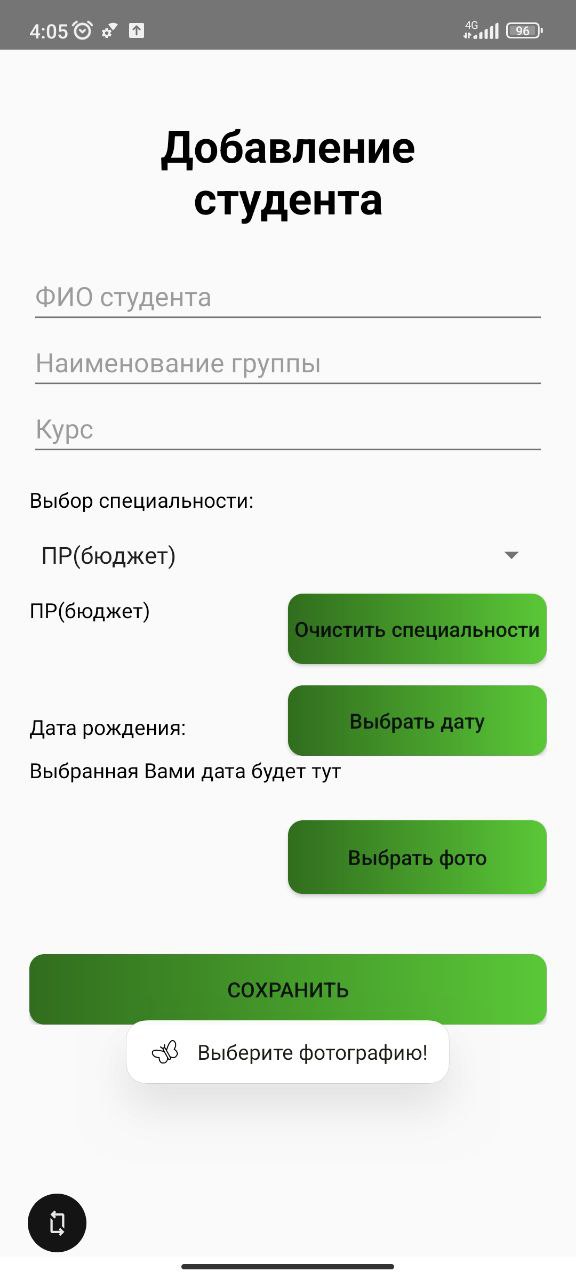
1. **Пользователь сбросил специальности.**

****

1. **Студент/ка с такими ФИО и датой рождения уже существует**

****

1. **Ничего не было введено.**

****

1. **На экране комиссии при добавлении переходит на экран навигации. На экране учителя подсказка вылазит на этом же экране.**

|  |  |
| --- | --- |
|  |  |

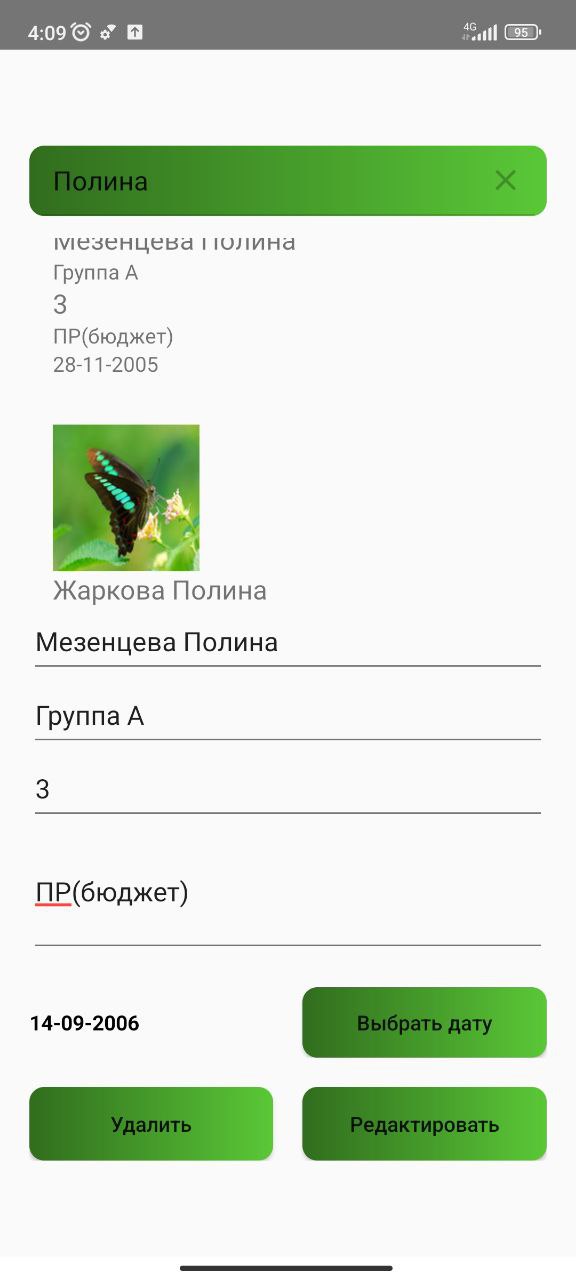
1. **Навигация экрана комиссии. Переход на экран Список студентов. (Экран Добавить студента был разобран ранее)**

|  |  |
| --- | --- |
|  |  |

1. **Выбрали студента и изменили информацию о нем.**

|  |  |
| --- | --- |
|  |  |

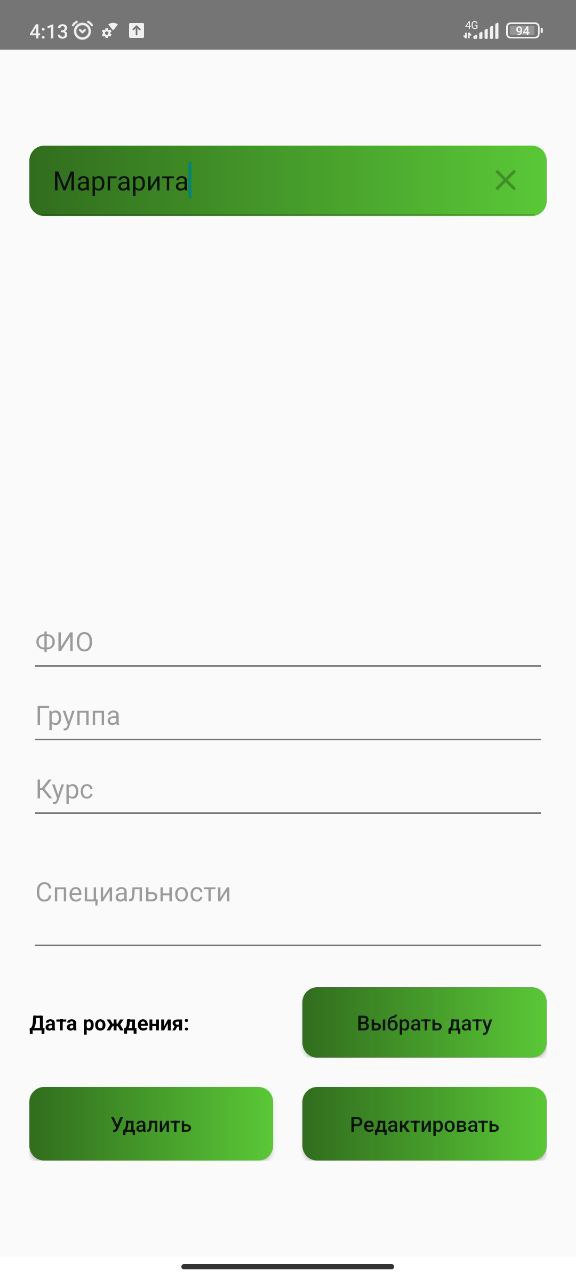
1. **Нашли студенток с именем Полина**

****

1. **Нашли Маргариту, удалили её.**

|  |  |
| --- | --- |
|  |  |

1. **На экране учителя Маргариты также нет**

****

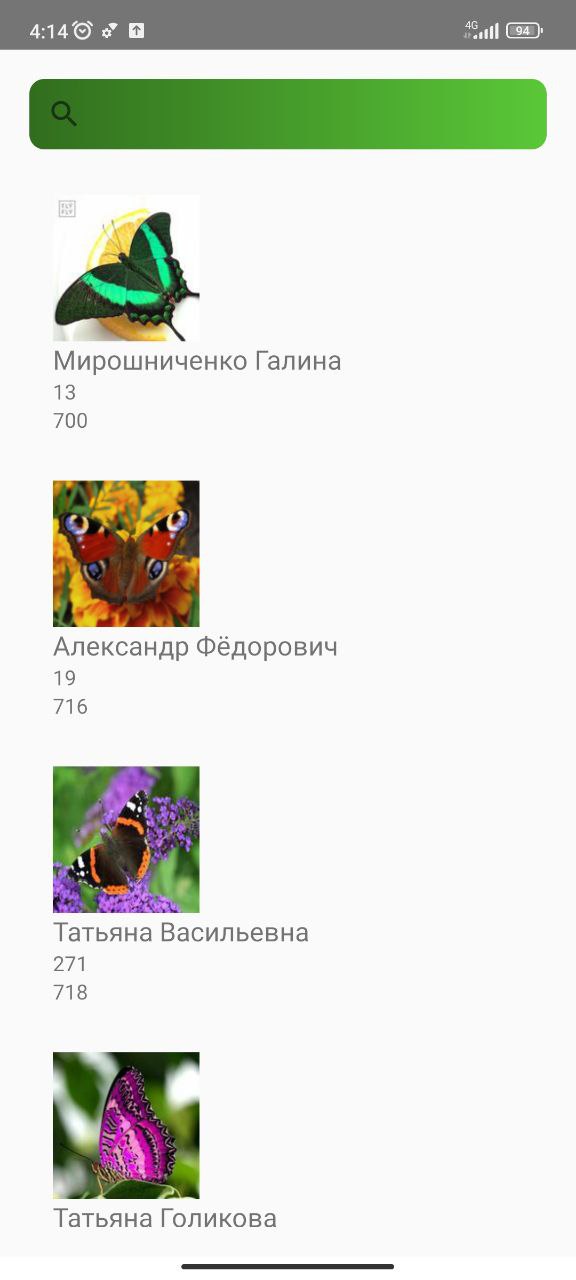
1. **Статистика до и после удаления студентки.**

|  |  |
| --- | --- |
|  |  |

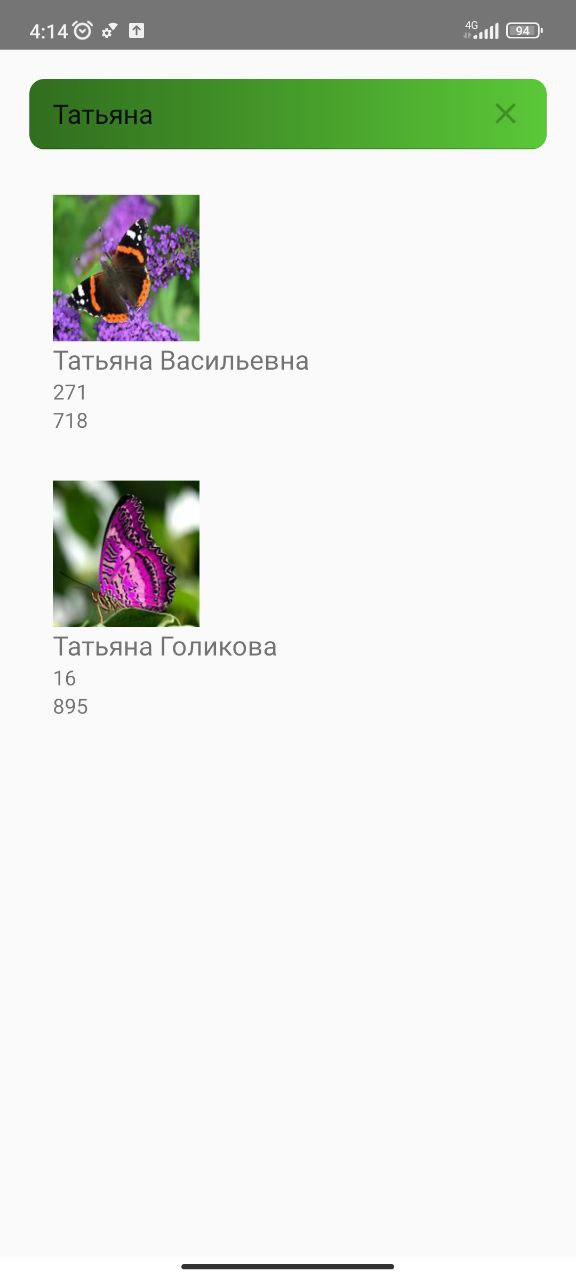
1. **Добавили преподавателей.**

|  |  |  |
| --- | --- | --- |
|  |  |  |

1. **Список преподавателей на Экране студента**

****

1. **Студент нашел преподавателей с именем Татьяна.**

****

1. **Не нашли преподавателей с именем Елизавета**

****

# Используемые библиотеки:

|  |  |
| --- | --- |
| **android.widget.Toast** | Класс для отображения коротких всплывающих уведомлений (тостов) на экране. Toast используется для вывода сообщений пользователю, которые автоматически исчезают через несколько секунд. Например, уведомления об успешном выполнении операции или ошибках. |
| **android.widget** | Пакет, содержащий различные виджеты для создания пользовательского интерфейса в Android, такие как кнопки, текстовые поля и списки. Он предоставляет компоненты для взаимодействия с пользователем. |
| **android.app.DatePickerDialog** | Диалоговое окно с календарем, позволяющее пользователю выбрать дату. Этот компонент часто используется в формах, где требуется ввод даты (например, дата рождения). Поддерживает настройку минимальной и максимальной даты, а также начальной даты по умолчанию. |
| **android.view** | Пакет, который содержит классы для создания и управления пользовательским интерфейсом, включая элементы управления и обработку событий касания. Он отвечает за отображение интерфейса и взаимодействие с пользователем. |
| **androidx.recyclerview** | Мощная библиотека для отображения больших наборов данных в виде списков или сеток. RecyclerView оптимизирован для работы с большими объемами данных, поддерживает анимации, различные типы layout (линейный, сетка) и позволяет создавать сложные интерфейсы с минимальными затратами ресурсов. |
| **androidx.fragment** | Библиотека для работы с фрагментами — модульными компонентами UI, которые могут быть использованы в разных Activity. Фрагменты позволяют создавать гибкие и адаптивные интерфейсы, особенно для планшетов и устройств с большими экранами. Они имеют собственный жизненный цикл и могут быть добавлены, удалены или заменены динамически. |
| **kotlinx.coroutines** | Библиотека для работы с асинхронным программированием в Kotlin. Корутины позволяют писать асинхронный код в синхронном стиле, что делает его более читаемым и удобным для работы. Поддерживает отмену задач, работу с потоками (Dispatchers) и интеграцию с другими библиотеками, такими как Room и Retrofit. |
| **androidx.room.Room** | Библиотека для работы с базой данных SQLite в Android. Room предоставляет абстракцию над SQLite, что упрощает работу с базой данных. Она автоматически генерирует код для выполнения запросов, проверяет их на этапе компиляции и поддерживает LiveData и Flow для наблюдения за изменениями данных. |
| **androidx.room.Dao** | Интерфейс, который определяет методы для доступа к данным в базе данных Room. DAO (Data Access Object) содержит методы для выполнения операций CRUD (Create, Read, Update, Delete) и может возвращать данные в виде объектов, LiveData или Flow. Room автоматически генерирует реализацию DAO на основе аннотаций. |
| **com.squareup.picasso.Picasso** | Picasso — это библиотека для загрузки и кэширования изображений для Android. Она упрощает процесс загрузки изображений из интернета в ImageView |

**Используемые инструменты:** Android Studio Kotlin.