Министерство образования и молодежной политики Свердловской области



ГАПОУ СО «Екатеринбургский колледж транспортного строительства»

Отчёт по программе «**Базы данных в Android Studio**»

Выполнил: Ковязин А.М

Группа: ПР-31

Преподаватель: Мирошниченко Г.В

2023

**Вариант: 7**

**Входные данные:**

login(String) – логин

password(String) – пароль  
search1(String) – название для поиска преподавателя

search2(String) – название для поиска студента

studentnameEditText(String) – ФИО студжента  
imageEditText(String) – изображение студента  
birthdayEditText(String) – дата рождения студента  
cursEditText(Int) - курс  
teachernameEditText(String) – название преподавателя  
hourstechersr(Int) – часы работы преподавателя

specialityEditText(String) – название специальности

**Выходные данные:**

**recyclerView –** список

**Toast – сообщение  
AlertDialog - сообщение**

**Листинг программы:**

class SpecialitiesAdapter : RecyclerView.Adapter<SpecialitiesAdapter.SpecialitiesViewHolder>() {  
 private var specialitiesList:List<Speciality> = *emptyList*()  
 private var onDeleteClickListener: ((Int) -> Unit)? = null  
 private var onEditClickListener:((Int)->Unit)? = null  
 override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): SpecialitiesViewHolder {  
 val itemView = LayoutInflater.from(parent.*context*)  
 .inflate(R.layout.*speciatiesitem*,parent,false)  
 return SpecialitiesViewHolder(itemView,onDeleteClickListener,onEditClickListener)  
 }  
 override fun onBindViewHolder(holder: SpecialitiesViewHolder, position: Int) {  
 holder.bind(specialitiesList[position], position)  
 }  
  
  
 override fun getItemCount(): Int =specialitiesList.size  
 fun setData(newSpecialities: List<Speciality>) {  
 specialitiesList = newSpecialities  
 notifyDataSetChanged()  
 }  
 fun setOnDeleteClickListener(listener: (Int) -> Unit) {  
 onDeleteClickListener = listener  
 }  
 fun setOnEditClickListener(listener: (Int) -> Unit){  
 onEditClickListener = listener  
 }  
 fun getSpecialitiesList():List<Speciality>{  
 return specialitiesList  
 }  
 class SpecialitiesViewHolder(  
 itemView: View,  
 private val onDeleteClickListener:((Int)->Unit)?=null,  
 private val onEditClickListener:((Int)->Unit)?=null  
 ):RecyclerView.ViewHolder(itemView) {  
 private val specialitytext:TextView = itemView.findViewById(R.id.*specialitytext*)  
 private val deleteButton: Button = itemView.findViewById(R.id.*delbuttonspeciality*)  
 private val editButton:Button = itemView.findViewById(R.id.*editbuttonspiciality*)  
 fun bind(speciality: Speciality,position:Int){  
 specialitytext.*text* = "Специальность: ${speciality.speciality}"  
  
  
 deleteButton.setOnClickListener**{** onDeleteClickListener?.invoke(position)  
 **}** editButton.setOnClickListener**{** onEditClickListener?.invoke(position)  
 **}** }  
 }

class StudentsAdapter : RecyclerView.Adapter<StudentsAdapter.StudentsViewHolder>() {  
 private var studentsList: List<StudentWithSpeciality> = *emptyList*()  
 private var onDeleteClickListener: ((Int) -> Unit)? = null  
 private var onEditClickListener:((Int)->Unit)? = null  
  
 override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): StudentsViewHolder {  
 val itemView = LayoutInflater.from(parent.*context*)  
 .inflate(R.layout.*studentitem*, parent, false)  
 return StudentsViewHolder(itemView, onDeleteClickListener,onEditClickListener)  
 }  
  
 override fun onBindViewHolder(holder: StudentsViewHolder, position: Int) {  
  
 holder.bind(studentsList[position], position)  
  
 }  
  
 override fun getItemCount(): Int = studentsList.size  
  
 fun setData(newStudents: List<StudentWithSpeciality>) {  
 studentsList = newStudents  
 notifyDataSetChanged()  
 }  
  
 fun getStudentsList(): List<StudentWithSpeciality> {  
 return studentsList  
 }  
  
  
 fun setOnDeleteClickListener(listener: (Int) -> Unit) {  
 onDeleteClickListener = listener  
 }  
 fun setOnEditClickListener(listener: (Int) -> Unit){  
 onEditClickListener = listener  
 }  
 class StudentsViewHolder(  
 itemView: View,  
 private val onDeleteClickListener: ((Int) -> Unit)? = null,  
 private val onEditClickListener:((Int)->Unit)? = null  
 ) : RecyclerView.ViewHolder(itemView) {  
 private val studentNameTextView: TextView = itemView.findViewById(R.id.*studentnametext*)  
 private val studentBirthdayTextView: TextView = itemView.findViewById(R.id.*studentbirthdaytext*)  
 private val studentCoursesTextView: TextView = itemView.findViewById(R.id.*studentcurstext*)  
 private val studentSpeciality:TextView = itemView.findViewById(R.id.*studentspecialitytext*)  
 private val budgetTextView:TextView = itemView.findViewById(R.id.*byudjettext*)  
 private val studentImage:ImageView = itemView.findViewById(R.id.*studentimage*)  
 private val deleteStudentButton: Button = itemView.findViewById(R.id.*delbuttonstudent*)  
 private val editSpecialityButton:Button = itemView.findViewById(R.id.*editbuttonstudent*)  
  
  
  
  
  
 fun bind(studentWithSpeciality: StudentWithSpeciality, position: Int) {  
 studentNameTextView.*text* = "Студент: ${studentWithSpeciality.student.name}"  
 studentBirthdayTextView.*text* = "Дата рождения:${studentWithSpeciality.student.birthday}"  
 studentCoursesTextView.*text* = "Курс: ${studentWithSpeciality.student.course}"  
 studentSpeciality.*text* = "Специальность: ${studentWithSpeciality.studentSpeciality}"  
 if(studentWithSpeciality.student.isBudget){  
 budgetTextView.*text* = "Бюджет"  
 }  
 else{  
 budgetTextView.*text* = "Не бюджет"  
 }  
 Picasso.get()  
 .load(studentWithSpeciality.student.photo)  
 .into(studentImage)  
  
  
 deleteStudentButton.setOnClickListener **{** onDeleteClickListener?.invoke(position)  
 **}** editSpecialityButton.setOnClickListener**{** onEditClickListener?.invoke(position)  
 **}** }

class StudentsInfoAdapter: RecyclerView.Adapter<StudentsInfoAdapter.StudentsInfoViewHolder>() {  
 private var studentsList: List<StudentWithSpeciality> = *emptyList*()  
  
 override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): StudentsInfoViewHolder {  
 val itemView = LayoutInflater.from(parent.*context*)  
 .inflate(R.layout.*studentinfoitem*, parent, false)  
 return StudentsInfoViewHolder(itemView)  
 }  
  
 override fun onBindViewHolder(holder: StudentsInfoViewHolder, position: Int) {  
 holder.bind(studentsList[position])  
 }  
  
 override fun getItemCount(): Int = studentsList.size  
  
 fun setData(newStudents: List<StudentWithSpeciality>) {  
 studentsList = newStudents  
 notifyDataSetChanged()  
 }  
  
 fun getStudentsList(): List<StudentWithSpeciality> {  
 return studentsList  
 }  
 class StudentsInfoViewHolder(  
 itemView: View  
 ) : RecyclerView.ViewHolder(itemView) {  
 private val studentNameTextView: TextView = itemView.findViewById(R.id.*studentnametext*)  
 private val studentBirthdayTextView: TextView = itemView.findViewById(R.id.*studentbirthdaytext*)  
 private val studentCoursesTextView: TextView = itemView.findViewById(R.id.*studentcurstext*)  
 private val studentSpeciality:TextView = itemView.findViewById(R.id.*studentspecialitytext*)  
 private val budgetTextView:TextView = itemView.findViewById(R.id.*byudjettext*)  
 private val studentImage:ImageView = itemView.findViewById(R.id.*studentinfoimage*)  
  
  
  
  
 fun bind(studentWithSpeciality: StudentWithSpeciality) {  
 studentNameTextView.*text* = "Студент: ${studentWithSpeciality.student.name}"  
 studentBirthdayTextView.*text* = "Дата рождения:${studentWithSpeciality.student.birthday}"  
 studentCoursesTextView.*text* = "Курс: ${studentWithSpeciality.student.course}"  
 studentSpeciality.*text* = "Специальность: ${studentWithSpeciality.studentSpeciality}"  
 if(studentWithSpeciality.student.isBudget){  
 budgetTextView.*text* = "Бюджет"  
 }  
 else{  
 budgetTextView.*text* = "Не бюджет"  
 }  
 Picasso.get().load(studentWithSpeciality.student.photo).into(studentImage)  
  
  
 }

class TeachersAdapter:RecyclerView.Adapter<TeachersAdapter.TeacherViewHolder>() {  
  
 private var teachersList: List<Teacher> = *emptyList*()  
 private var onDeleteClickListener: ((Int) -> Unit)? = null  
 private var onEditClickListener:((Int) ->Unit)?=null  
  
 override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): TeacherViewHolder {  
 val itemView = LayoutInflater.from(parent.*context*)  
 .inflate(R.layout.*teacheritem*, parent, false)  
 return TeacherViewHolder(itemView,onDeleteClickListener,onEditClickListener)  
 }  
  
 override fun onBindViewHolder(holder: TeacherViewHolder, position: Int) {  
 holder.bind(teachersList[position] ,position,teachersList)  
 }  
  
 override fun getItemCount(): Int = teachersList.size  
  
 fun setData(newTeachers: List<Teacher>) {  
 teachersList = newTeachers  
 notifyDataSetChanged()  
 }  
 fun setOnDeleteClickListener(listener: (Int) -> Unit) {  
 onDeleteClickListener = listener  
 }  
 fun setOnEditClickListener(listener: (Int) -> Unit) {  
 onEditClickListener = listener  
 }  
 fun getTeachersList(): List<Teacher> {  
 return teachersList  
 }  
 class TeacherViewHolder(  
 itemView: View,  
 private val onDeleteClickListener: ((Int) -> Unit)? = null,  
 private val onEditClickListener: ((Int) -> Unit)? = null  
 ) : RecyclerView.ViewHolder(itemView) {  
 private val teacherNameTextView: TextView = itemView.findViewById(R.id.*teachernametext*)  
 private val teacherSalaryTextView: TextView = itemView.findViewById(R.id.*teachersalarytext*)  
 private val teacherHours: TextView = itemView.findViewById(R.id.*teacherperyeartext*)  
 private val teacherSpeciality: TextView = itemView.findViewById(R.id.*teacherspecialitutext*)  
 private val deleteButton: Button = itemView.findViewById(R.id.*delbuttonteacher*)  
 private val editButton: Button = itemView.findViewById(R.id.*editbuttonteacher*)  
  
 fun bind(  
 teacher: Teacher,  
 position: Int,  
 allTeachers: List<Teacher> // Добавляем список всех преподавателей  
 ) {  
 teacherNameTextView.*text* = "Преподаватель: ${teacher.name}"  
  
 var totalHours = 0  
 for(item in allTeachers){  
 if(teacher.name==item.name)  
 totalHours+=teacher.hoursPerYear;  
 }  
 if(totalHours>1440){  
 teacherSalaryTextView.*text* = "Зарплата : "+((totalHours\*150)+((totalHours-1440)\*150)).toString()  
 }  
 else{  
 teacherSalaryTextView.*text* = "Зарплата :" +(totalHours\*150).toString()  
 }  
  
 teacherHours.*text* = "Количество часов: ${teacher.hoursPerYear}"  
 teacherSpeciality.*text* = "Специальность: ${teacher.speciality}"  
  
  
 deleteButton.setOnClickListener **{** onDeleteClickListener?.invoke(position)  
 **}** editButton.setOnClickListener **{** onEditClickListener?.invoke(position)  
 **}** }

class TeachersInfoAdapter: RecyclerView.Adapter<TeachersInfoAdapter.TeachersInfoViewHolder>() {  
  
 private var teachersList: List<TeacherWithSpeciality> = *emptyList*()  
  
 override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): TeachersInfoViewHolder {  
 val itemView = LayoutInflater.from(parent.*context*)  
 .inflate(R.layout.*teacherinfoitem*, parent, false)  
 return TeachersInfoViewHolder(itemView)  
 }  
  
 override fun onBindViewHolder(holder: TeachersInfoViewHolder, position: Int) {  
 holder.bind(teachersList[position])  
 }  
  
 override fun getItemCount(): Int = teachersList.size  
  
 fun setData(newTeachers: List<TeacherWithSpeciality>) {  
 teachersList = newTeachers  
 notifyDataSetChanged()  
 }  
 fun getTeachersList(): List<TeacherWithSpeciality> {  
 return teachersList  
 }  
 class TeachersInfoViewHolder(itemView:View): RecyclerView.ViewHolder(itemView) {  
 private val teacherNameTextView: TextView = itemView.findViewById(R.id.*teachernametext*)  
 private val teacherSpeciality: TextView = itemView.findViewById(R.id.*teacherspecialitutext*)  
  
 private val teacherHours: TextView = itemView.findViewById(R.id.*teacherperyeartext*)  
  
 fun bind(  
 teacherWithSpeciality: TeacherWithSpeciality,  
  
 ) {  
 teacherNameTextView.*text* = "Преподаватель: ${teacherWithSpeciality.teacher.name}"  
 teacherHours.*text* = "Количество часов: ${teacherWithSpeciality.teacher.hoursPerYear}"  
 teacherSpeciality.*text* = "Специальность: ${teacherWithSpeciality.teacher.speciality}"  
  
 }  
 }

@Database(entities = [Student::class, Teacher::class, Speciality::class], version = 2)  
abstract class CollegeDatabase : RoomDatabase(){  
  
 abstract fun databaseDao():DatabaseDao  
  
 companion object {  
 @Volatile  
 public var INSTANCE: CollegeDatabase? = null  
  
 fun getDatabase(context: Context): CollegeDatabase {  
 return INSTANCE ?: *synchronized*(this) **{** val instance = Room.databaseBuilder(  
 context.*applicationContext*,  
 CollegeDatabase::class.*java*,  
 "college\_database4"  
 ).allowMainThreadQueries()  
 .build()  
  
 INSTANCE = instance  
 instance  
 **}** }  
  
  
 }

@Dao  
interface DatabaseDao {  
  
 @Query("SELECT \* FROM students")  
 suspend fun getAllStudents(): List<Student>  
  
 @Insert(onConflict = OnConflictStrategy.REPLACE)  
 suspend fun insertStudent(student: Student)  
  
 @Query("SELECT \* FROM teachers")  
 suspend fun getAllTeachers(): List<Teacher>  
  
 @Insert(onConflict = OnConflictStrategy.REPLACE)  
 suspend fun insertTeacher(teacher: Teacher)  
 @Transaction  
 @Query("SELECT students.\*, Speciality.speciality AS studentSpeciality FROM students INNER JOIN Speciality ON students.speciality = Speciality.speciality")  
 suspend fun getStudentsWithSpeciality(): List<StudentWithSpeciality>  
 @Transaction  
 @Query("SELECT students.\*, Speciality.speciality AS studentSpeciality FROM students INNER JOIN Speciality ON students.speciality = Speciality.speciality WHERE name = :name")  
 suspend fun getStudentsWithSpecialityByName(name:String): List<StudentWithSpeciality>  
 @Query("DELETE FROM students")  
 suspend fun deleteAllStudents()  
 @Insert(onConflict = OnConflictStrategy.REPLACE)  
 suspend fun insertStudents(students: List<Student>)  
 @Query("SELECT \* FROM students WHERE name LIKE '%' || :searchName || '%'")  
 suspend fun searchStudentsByName(searchName: String): List<Student>  
 @Query("SELECT \* FROM teachers WHERE name LIKE '%' || :searchName || '%'")  
 suspend fun searchTeachersByName(searchName: String): List<Teacher>  
 //@Query("UPDATE speciality SET speciality = :newSpeciality WHERE speciality = :oldSpeciality")  
 //suspend fun updateSpecialityName(oldSpeciality: String, newSpeciality: String)  
 @Query("UPDATE students SET name = :name, birthday = :birthday, speciality = :speciality, course = :course, isBudget = :isBudget, photo = :photo WHERE studentId = :studentId")  
 suspend fun updateStudent(  
 studentId: Long,  
 name: String,  
 birthday: String,  
 speciality: String,  
 course: Int,  
 isBudget: Boolean,  
 photo: String  
 )  
 @Query("SELECT SUM(hoursPerYear) FROM teachers WHERE name = :teacherName")  
 suspend fun getTotalHoursForTeacher(teacherName: String): Int  
  
 @Query("UPDATE teachers SET name = :name, salary = :salary,hoursPerYear=:hoursPerYear, speciality = :speciality WHERE teacherId = :teacherId")  
 suspend fun updateTeacher(  
 teacherId: Long,  
 name: String,  
 salary: Double,  
 hoursPerYear:Int,  
 speciality: String  
 )  
  
 @Delete  
 suspend fun deleteStudent(student: Student)  
 @Delete  
 suspend fun deleteTeacher(teacher:Teacher)  
 @Query("Select \* from speciality")  
 suspend fun getAllSpecialities():List<Speciality>  
 @Delete  
 suspend fun deleteSpeciality(speciality: Speciality)  
 @Insert(onConflict = OnConflictStrategy.REPLACE)  
 suspend fun insertSpeciality(speciality: Speciality)  
  
 @Query("SELECT teachers.\*, Speciality.speciality AS teacherSpeciality FROM teachers INNER JOIN Speciality ON teachers.speciality = Speciality.speciality WHERE name = :name")  
 suspend fun getTeachersWithSpecialityByName(name:String): List<TeacherWithSpeciality>  
  
 @Query("SELECT teachers.\*, Speciality.speciality AS teacherSpeciality FROM teachers INNER JOIN Speciality ON teachers.speciality = Speciality.speciality")  
 suspend fun getTeachersWithSpeciality(): List<TeacherWithSpeciality>  
  
 @Query("SELECT \* FROM speciality WHERE speciality = :name")  
 suspend fun getSpecialityByName(name: String): Speciality?  
  
 @Query("UPDATE speciality SET speciality = :newSpeciality WHERE speciality = :oldSpeciality")  
 suspend fun updateSpecialityName(oldSpeciality: String, newSpeciality: String): Int  
  
 @Query("SELECT speciality FROM speciality")  
 fun getAllSpecialityNames(): List<String>  
  
  
 @Update  
 fun updateStudent(student: Student)  
  
 @Update  
 fun updateTeacher(teacher: Teacher)  
  
  
}

@Entity(tableName = "speciality")  
data class Speciality(  
 @PrimaryKey(autoGenerate = false)  
 var speciality: String  
)  
  
@Entity(tableName = "students")  
data class Student(  
 @PrimaryKey(autoGenerate = true)  
 val studentId: Long = 0,  
 val name: String,  
 val birthday: String,  
 var speciality: String,  
 val course: Int,  
 val isBudget: Boolean,  
 val photo: String  
)  
@Entity(  
 tableName = "teachers",  
 foreignKeys = [  
 ForeignKey(  
 entity = Speciality::class,  
 parentColumns = ["speciality"],  
 childColumns = ["speciality"],  
 onDelete = ForeignKey.CASCADE  
 )  
 ]  
)  
data class Teacher(  
 @PrimaryKey(autoGenerate = true)  
 val teacherId: Long = 0,  
 val name: String,  
 val salary: Double,  
 val speciality: String, //связываем по этому полю с таблицей Speciality  
 val hoursPerYear: Int //добавляем новое поле  
)  
  
  
data class StudentWithSpeciality(  
 @Embedded val student: Student,  
 val studentSpeciality: String  
)  
data class TeacherWithSpeciality(  
 @Embedded val teacher: Teacher,  
 val teacherSpeciality: String  
)

class SpecialtiesFragment : Fragment() {  
 private lateinit var recycleview:RecyclerView  
 private lateinit var addSpecialities: Button  
 private lateinit var adapter: SpecialitiesAdapter  
 private lateinit var databaseDao: DatabaseDao  
 // *TODO: Rename and change types of parameters* private var param1: String? = null  
 private var param2: String? = null  
  
  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 *arguments*?.*let* **{** param1 = **it**.getString(*ARG\_PARAM1*)  
 param2 = **it**.getString(*ARG\_PARAM2*)  
  
  
  
 **}** }  
  
 override fun onCreateView(  
 inflater: LayoutInflater, container: ViewGroup?,  
 savedInstanceState: Bundle?  
 ): View? {  
 var view = inflater.inflate(R.layout.*fragment\_specialties*, container, false)  
 recycleview = view.findViewById(R.id.*recycleviewspecialities*)  
 addSpecialities = view.findViewById(R.id.*addspecialities*)  
 databaseDao = CollegeDatabase.getDatabase(requireContext()).databaseDao()  
 adapter = SpecialitiesAdapter()  
 recycleview.*layoutManager* = LinearLayoutManager(requireContext())  
 recycleview.*adapter* = adapter  
  
 *lifecycleScope*.*launch*(Dispatchers.IO) **{** val specialitiesList = databaseDao.getAllSpecialities()  
 withContext(Dispatchers.Main) **{** adapter.setData(specialitiesList)  
 **}  
 }** adapter.setOnDeleteClickListener **{** position **->** onDeleteSpeciality(position)  
 **}** adapter.setOnEditClickListener **{** position **->** val speciality = adapter.getSpecialitiesList()[position]  
 showEditDialog(speciality)  
 **}** addSpecialities.setOnClickListener**{** val intent = Intent(requireContext(),AddSpecialities::class.*java*)  
 startActivity(intent)  
 **}** return view  
 }  
  
 private fun showEditDialog(speciality: Speciality) {  
 val dialogBuilder = AlertDialog.Builder(requireContext())  
 val editText = EditText(requireContext())  
 editText.setText(speciality.speciality)  
  
 dialogBuilder  
 .setTitle("Редактирование")  
 .setView(editText)  
 .setPositiveButton("Сохранить") **{** dialog, \_ **->** val newSpecialityName = editText.*text*.toString()  
 if (newSpecialityName.*isNotEmpty*() && newSpecialityName != speciality.speciality) {  
 *lifecycleScope*.*launch*(Dispatchers.IO) **{** val existingSpeciality = databaseDao.getSpecialityByName(newSpecialityName)  
 if (existingSpeciality == null && existingSpeciality != speciality) {  
 // Если новой специальности нет в базе данных, обновляем  
 databaseDao.updateSpecialityName(speciality.speciality, newSpecialityName)  
 val updatedSpecialities = databaseDao.getAllSpecialities()  
 withContext(Dispatchers.Main) **{** adapter.setData(updatedSpecialities)  
 **}** }  
 **}** }  
 dialog.dismiss()  
 **}** .setNegativeButton("Отмена") **{** dialog, \_ **->** dialog.dismiss()  
 **}** .create()  
 .show()  
 }  
  
 private fun onDeleteSpeciality(position: Int) {  
 *lifecycleScope*.*launch*(Dispatchers.IO) **{** val speciality = adapter.getSpecialitiesList().*getOrNull*(position)  
 speciality?.*let* **{** databaseDao.deleteSpeciality(speciality)  
 val updatedSpecialities = databaseDao.getAllSpecialities()  
 withContext(Dispatchers.Main) **{** adapter.setData(updatedSpecialities)  
 **}  
 }  
 }** }

class StudentsFragment : Fragment() {  
private lateinit var recycleView: RecyclerView  
private lateinit var addStudents: Button  
private lateinit var databaseDao: DatabaseDao  
private lateinit var adapter: StudentsAdapter  
private lateinit var buttonexit:Button  
  
// *TODO: Rename and change types of parameters*private var param1: String? = null  
private var param2: String? = null  
  
override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 *arguments*?.*let* **{** param1 = **it**.getString(*ARG\_PARAM1*)  
 param2 = **it**.getString(*ARG\_PARAM2*)  
 **}**}  
  
@SuppressLint("MissingInflatedId")  
override fun onCreateView(  
 inflater: LayoutInflater, container: ViewGroup?,  
 savedInstanceState: Bundle?  
): View? {  
 var view = inflater.inflate(R.layout.*fragment\_students*, container, false)  
 recycleView = view.findViewById(R.id.*recycleviewstudents*)  
 addStudents = view.findViewById(R.id.*addstudent*)  
 databaseDao = CollegeDatabase.getDatabase(requireContext()).databaseDao()  
 adapter = StudentsAdapter()  
 recycleView.*layoutManager* = LinearLayoutManager(requireContext())  
 recycleView.*adapter* = adapter  
 buttonexit = view.findViewById(R.id.*buttonexit*)  
  
 *lifecycleScope*.*launch*(Dispatchers.IO) **{** val studentsList = databaseDao.getStudentsWithSpeciality()  
 withContext(Dispatchers.Main) **{** adapter.setData(studentsList)  
 **}  
 }** adapter.setOnDeleteClickListener **{** position **->** onDeleteStudent(position)  
 **}** adapter.setOnEditClickListener **{** position **->** val student = adapter.getStudentsList()[position]  
 showEditDialog(student)  
  
 **}** buttonexit.setOnClickListener **{** val intent = Intent(requireContext(), SignInActivity::class.*java*)  
 startActivity(intent)  
 **}** addStudents.setOnClickListener **{** val intent = Intent(requireContext(), AddStudent::class.*java*)  
 startActivity(intent)  
 **}** return view  
}  
  
private fun onDeleteStudent(position: Int) {  
 *lifecycleScope*.*launch*(Dispatchers.IO) **{** val student = adapter.getStudentsList().*getOrNull*(position)?.student  
 student?.*let* **{** databaseDao.deleteStudent(**it**)  
 val updatedStudents = databaseDao.getStudentsWithSpeciality()  
 withContext(Dispatchers.Main) **{** adapter.setData(updatedStudents)  
 **}  
 }  
 }**}  
  
  
private fun showEditDialog(student: StudentWithSpeciality) {  
 val builder = AlertDialog.Builder(requireContext())  
 val specialityNames = databaseDao.getAllSpecialityNames()  
 val layout = LinearLayout(requireContext())  
 layout.*orientation* = LinearLayout.*VERTICAL* var name = student.student.name  
 var photo = student.student.photo  
 var birthday = student.student.birthday  
 var curs = student.student.course  
 var budjet = student.student.isBudget  
 var speciality = student.student.speciality  
  
 builder.setTitle("Редактор данных")  
 var ed1 = EditText(requireContext())  
 var ed2 = EditText(requireContext())  
 var ed3 = EditText(requireContext())  
 var ed4 = EditText(requireContext())  
 var checkbox = CheckBox(requireContext())  
 var spinner = Spinner(requireContext())  
  
 ed3.*inputType* = InputType.*TYPE\_DATETIME\_VARIATION\_DATE* checkbox.*text* = "Бюджет"  
 ed1.*hint* = "Введите ФИО студента"  
 val specialityAdapter = ArrayAdapter<String>(  
 requireContext(), android.R.layout.*simple\_spinner\_item*, specialityNames  
 )  
 specialityAdapter.setDropDownViewResource(android.R.layout.*simple\_spinner\_dropdown\_item*)  
 spinner.*adapter* = specialityAdapter  
 ed2.*hint* = "Введите ссылка на фото"  
 ed3.*hint* = "Введите дату рождения"  
 ed4.*hint* = "Введите курс"  
 ed1.setText("${name}")  
 ed2.setText("${photo}")  
 ed3.setText("${birthday}")  
 ed4.setText("${curs}")  
 layout.addView(ed1)  
 layout.addView(ed2)  
 layout.addView(ed3)  
 layout.addView(ed4)  
 layout.addView(checkbox)  
 layout.addView(spinner)  
 builder.setView(layout)  
 builder.setPositiveButton("Сохранить") **{** \_, \_ **->** val newinf1 = ed1.*text*.toString()  
 val newinf2 = ed2.*text*.toString()  
 val newinf3 = ed3.*text*.toString()  
 val newinf4 = ed4.*text*.toString().*toInt*()  
 val budjetinfo = checkbox.*isChecked* val spinerinfo = spinner.*selectedItem* if (newinf1.*isNotEmpty*() && newinf2.*isNotEmpty*() && newinf3.*isNotEmpty*()) {  
 if (isValidURL(newinf2)) {  
 val updatedStudent = student.student.copy(  
 name = newinf1,  
 photo = newinf2,  
 birthday = newinf3,  
 course = newinf4,  
 speciality = spinerinfo.toString(),  
 isBudget = budjetinfo  
 )  
 *lifecycleScope*.*launch*(Dispatchers.IO) **{** val existingStudent = databaseDao.getStudentsWithSpecialityByName(newinf1)  
 if (existingStudent.isEmpty() || existingStudent.*first*().student.studentId == student.student.studentId) {  
 databaseDao.updateStudent(updatedStudent)  
 val students = databaseDao.getStudentsWithSpeciality()  
 withContext(Dispatchers.Main) **{** adapter.setData(students)  
 **}** }  
 **}** } else {  
 // В случае неправильной ссылки, используем заглушку для фото из интернета  
 val placeholderURL = "https://cdn-icons-png.flaticon.com/512/4054/4054617.png"  
 val updatedStudent = student.student.copy(  
 name = newinf1,  
 photo = placeholderURL,  
 birthday = newinf3,  
 course = newinf4,  
 speciality = spinerinfo.toString(),  
 isBudget = budjetinfo  
 )  
 *lifecycleScope*.*launch*(Dispatchers.IO) **{** val existingStudent = databaseDao.getStudentsWithSpecialityByName(newinf1)  
 if (existingStudent.isEmpty() || existingStudent.*first*().student.studentId == student.student.studentId) {  
 databaseDao.updateStudent(updatedStudent)  
 val students = databaseDao.getStudentsWithSpeciality()  
 withContext(Dispatchers.Main) **{** adapter.setData(students)  
 **}** }  
 **}** Toast.makeText(  
 requireContext(),  
 "Неправильная ссылка на фото. Использована заглушка",  
 Toast.*LENGTH\_SHORT* ).show()  
 }  
 } else {  
  
 Toast.makeText(  
 requireContext(),  
 "Пожалуйста, заполните все поля",  
 Toast.*LENGTH\_SHORT* ).show()  
 }  
 **}** builder.setNegativeButton("Отмена") **{** dialog, \_ **->** dialog.dismiss()  
 **}** builder.show()  
  
}  
  
fun isValidURL(url: String): Boolean {  
 return try {  
 URL(url).toURI()  
 true  
 } catch (e: Exception) {  
 false  
 }  
}

class TeachersFragment : Fragment() {  
 private lateinit var recycleView:RecyclerView  
 private lateinit var addTeacher:Button  
 private lateinit var adapter: TeachersAdapter  
 private lateinit var newadapter:TeachersAdapterNew  
  
 private lateinit var database: DatabaseDao  
 // *TODO: Rename and change types of parameters* private var param1: String? = null  
 private var param2: String? = null  
  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 *arguments*?.*let* **{** param1 = **it**.getString(*ARG\_PARAM1*)  
 param2 = **it**.getString(*ARG\_PARAM2*)  
 **}** }  
  
  
 override fun onCreateView(  
 inflater: LayoutInflater, container: ViewGroup?,  
 savedInstanceState: Bundle?  
 ): View? {  
 var view = inflater.inflate(R.layout.*fragment\_teachers*, container, false)  
 recycleView = view.findViewById(R.id.*recycleviewteachers*)  
 addTeacher = view.findViewById(R.id.*addteacher*)  
 database = CollegeDatabase.getDatabase(requireContext()).databaseDao()  
 adapter = TeachersAdapter()  
  
 recycleView.*layoutManager* = LinearLayoutManager(requireContext())  
 recycleView.*adapter* = adapter  
  
 *lifecycleScope*.*launch*(Dispatchers.IO) **{** val studentsList = database.getAllTeachers()  
 withContext(Dispatchers.Main) **{** adapter.setData(studentsList)  
 **}** adapter.setOnDeleteClickListener **{** position **->** onDeleteTeacher(position)  
 **}** adapter.setOnEditClickListener **{** position **->** val teacher = adapter.getTeachersList()[position]  
 showEditDialog(teacher)  
 **}  
 }** addTeacher.setOnClickListener**{** val intent = Intent(requireContext(), AddTeacher::class.*java*)  
 startActivity(intent)  
 **}** return view  
 }  
 private fun onDeleteTeacher(position: Int) {  
 *lifecycleScope*.*launch*(Dispatchers.IO) **{** val teacher = adapter.getTeachersList().*getOrNull*(position)  
 teacher?.*let* **{** database.deleteTeacher(**it**)  
 val updatedTeachers = database.getAllTeachers()  
 withContext(Dispatchers.Main) **{** adapter.setData(updatedTeachers)  
 **}  
 }  
 }** }  
 private fun showEditDialog(teacher: Teacher) {  
 val builder = AlertDialog.Builder(requireContext())  
 val specialityNames = database.getAllSpecialityNames()  
 val layout = LinearLayout(requireContext())  
 layout.*orientation* = LinearLayout.*VERTICAL* var name = teacher.name  
 var speciality = teacher.speciality  
 var salary = teacher.salary  
 var hours = teacher.hoursPerYear  
  
 builder.setTitle("Редактор данных")  
 var ed1 = EditText(requireContext())  
 var spinner = Spinner(requireContext())  
 var ed2 = EditText(requireContext())  
 var ed3 = EditText(requireContext())  
  
 ed2.*inputType* = InputType.*TYPE\_CLASS\_NUMBER* ed3.*inputType* = InputType.*TYPE\_CLASS\_NUMBER* ed1.*hint* = "Введите ФИО преподавателя"  
 val specialityAdapter = ArrayAdapter<String>(  
 requireContext(), android.R.layout.*simple\_spinner\_item*, specialityNames  
 )  
 specialityAdapter.setDropDownViewResource(android.R.layout.*simple\_spinner\_dropdown\_item*)  
 spinner.*adapter* = specialityAdapter  
 ed2.*hint* = "Введите заработную плату"  
 ed3.*hint* = "Введите количество часов"  
 ed1.setText("${name}")  
 ed2.setText("${salary}")  
 ed3.setText("${hours}")  
  
 layout.addView(ed1)  
 layout.addView(ed2)  
 layout.addView(ed3)  
 layout.addView(spinner)  
 builder.setView(layout)  
 builder.setPositiveButton("Сохранить") **{** \_, \_ **->** val newinf1 = ed1.*text*.toString()  
 val newinf2 = ed2.*text*.toString().*toDouble*()  
 val newinf3 = ed3.*text*.toString().*toInt*()  
  
 val spinerinfo = spinner.*selectedItem* if (newinf1.*isNotEmpty*() && newinf2.toString().*isNotEmpty*() && newinf3.toString().*isNotEmpty*()) {  
 val updatedTeacher = teacher.copy(  
 name = newinf1,  
 salary = newinf2.toDouble(),  
 hoursPerYear = newinf3,  
 speciality = spinerinfo.toString(),  
 )  
  
 *lifecycleScope*.*launch*(Dispatchers.IO) **{** val existingTeacher = database.getTeachersWithSpecialityByName(newinf1)  
 if (existingTeacher.isEmpty() || existingTeacher.*first*().teacher.teacherId == teacher.teacherId) {  
  
 database.updateTeacher(updatedTeacher)  
  
  
 val teachers = database.getAllTeachers()  
 withContext(Dispatchers.Main) **{** adapter.setData(teachers)  
 **}** }  
 **}** } else {  
 Toast.makeText(requireContext(), "Пожалуйста, заполните все поля", Toast.*LENGTH\_SHORT*).show()  
 }  
 **}** builder.setNegativeButton("Отмена") **{** dialog,\_ **->** dialog.dismiss()  
  
 **}** builder.show()  
  
  
  
  
  
 }

class AddSpecialities : AppCompatActivity() {  
 private lateinit var specialityEditText:EditText  
 private lateinit var addSpecialitiybase: Button  
 private lateinit var databaseDao: DatabaseDao  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 setContentView(R.layout.*activity\_add\_specialities*)  
 databaseDao = CollegeDatabase.getDatabase(this).databaseDao()  
 specialityEditText = findViewById(R.id.*specialityEditText*)  
 addSpecialitiybase = findViewById(R.id.*addSpecialitybase*)  
 var specialityName = specialityEditText  
  
  
 addSpecialitiybase.setOnClickListener**{** try {  
 if (specialityName.*text*.*isNotBlank*()) {  
 var specialityName = specialityEditText.*text*.toString()  
 *lifecycleScope*.*launch*(Dispatchers.IO) **{** val speciality = Speciality(specialityName)  
 databaseDao.insertSpeciality(speciality)  
 runOnUiThread **{** Toast.makeText(  
 this@AddSpecialities,  
 "Вы добавили специальность",  
 Toast.*LENGTH\_SHORT* ).show()  
 val intent =  
 Intent(this@AddSpecialities, AdmissionCommittee::class.*java*)  
 startActivity(intent)  
 **}  
  
 }** }  
 }catch (ex:Exception){  
 Toast.makeText(this,"Ошибка в вводе",Toast.*LENGTH\_SHORT*).show()  
 }

class AddStudent : AppCompatActivity() {  
 private lateinit var studentnameEditText:EditText  
 private lateinit var imageEditText:EditText  
 private lateinit var birthdayEditText:EditText  
 private lateinit var calendarButton:Button  
 private lateinit var groupSpinner:Spinner  
 private lateinit var cursEditText:EditText  
 private lateinit var byudjetcheckbox:CheckBox  
 private lateinit var addStudentButton:Button  
 private lateinit var databaseDao:DatabaseDao  
 private lateinit var binding: ActivityAddStudentBinding  
 private lateinit var error:String  
  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 binding = ActivityAddStudentBinding.inflate(*layoutInflater*)  
 setContentView(binding.*root*)  
 databaseDao = CollegeDatabase.getDatabase(this).databaseDao()  
  
 val groupAdapter = ArrayAdapter(  
 this,  
 R.layout.*simple\_spinner\_item*,  
 *emptyArray*<String>()  
 ).*apply* **{** setDropDownViewResource(R.layout.*simple\_spinner\_dropdown\_item*)  
 **}** binding.groupSpinner.*adapter* = groupAdapter  
 loadGroups()  
  
  
 binding.calendarButton.setOnClickListener **{** val calendar = Calendar.getInstance()  
 val year = calendar.get(Calendar.*YEAR*)  
 val month = calendar.get(Calendar.*MONTH*)  
 val day = calendar.get(Calendar.*DAY\_OF\_MONTH*)  
  
 val datePicker = DatePickerDialog(  
 this,  
 **{** view, selectedYear, selectedMonth, selectedDay **->** // Проверяем, что выбранная дата входит в диапазон от 1990 до 2007 года  
 val selectedDate = Calendar.getInstance()  
 selectedDate.set(Calendar.*YEAR*, selectedYear)  
 selectedDate.set(Calendar.*MONTH*, selectedMonth)  
 selectedDate.set(Calendar.*DAY\_OF\_MONTH*, selectedDay)  
  
 val minDate = Calendar.getInstance()  
 minDate.set(1990, 0, 1)  
  
 val maxDate = Calendar.getInstance()  
 maxDate.set(2007, 11, 31)  
  
 if (selectedDate.before(minDate) || selectedDate.after(maxDate)) {  
 Toast.makeText(  
 this,  
 "Пожалуйста, выберите дату между 1990 и 2007 годом",  
 Toast.*LENGTH\_SHORT* ).show()  
 } else {  
 binding.birthdayEditText.setText(  
 SimpleDateFormat(  
 "dd.MM.yyyy",  
 Locale.getDefault()  
 ).format(selectedDate.*time*)  
 )  
 }  
 **}**,  
 year, month, day  
 )  
 datePicker.show()  
 **}** suspend fun CheckImageAvailability(imageUrl: String): Boolean {  
 val client = *HttpClient*()  
 return try {  
 val response: io.ktor.client.statement.HttpResponse = client.get(imageUrl)  
 true  
 } catch (e: Exception) {  
 false  
 } finally {  
 client.close()  
 }  
 }  
  
 binding.addStudentButton.setOnClickListener **{** try {  
  
  
 val studentName = binding.studentnameEdiText.*text*.toString()  
 val groupName = binding.groupSpinner.*selectedItem*.toString()  
 val coursText = binding.cursEditText.*text*.toString()  
 val birthday = binding.birthdayEditText.*text*.toString()  
 var image = binding.imageEditText.*text*.toString()  
 val isBudget = binding.byudjetcheckbox.*isChecked* if (studentName.*isNotBlank*() && groupName.*isNotBlank*() && coursText.*isNotBlank*() && image.*isNotBlank*() && birthday != "Нажмите на кнопку справа, чтобы выбрать дату:")  
 {  
 val coursText = coursText.*toInt*()  
 if (coursText > 0 && coursText < 5) {  
  
 val imageAvailable = *runBlocking* **{** CheckImageAvailability(image) **}** if (!imageAvailable) {  
 image = "https://cdn-icons-png.flaticon.com/512/4054/4054617.png"  
 }  
 *lifecycleScope*.*launch*(Dispatchers.IO) **{** val student = Student(name = studentName, birthday = birthday, speciality = groupName, course = coursText.toInt(), isBudget = isBudget, photo = image)  
 if(isStudentUnique(student)) {  
 databaseDao.insertStudent(student)  
 finish()  
 }  
 else{  
 runOnUiThread**{** Toast.makeText(this@AddStudent,error,Toast.*LENGTH\_SHORT*).show()  
 **}** }  
 **}** Toast.makeText(this@AddStudent,"Вы успешно добавили студента",Toast.*LENGTH\_SHORT*).show()  
 val intent = Intent(this,AdmissionCommittee::class.*java*)  
 startActivity(intent)  
  
 } else {  
 Toast.makeText(  
 this,  
 "Курс должен быть больше нуля и не больше 4",  
 Toast.*LENGTH\_SHORT* ).show()  
 }  
 } else {  
 Toast.makeText(this, "Введите все данные", Toast.*LENGTH\_SHORT*).show()  
 }  
  
 }catch (ex:Exception)  
 {  
 Toast.makeText(this,"Проверьте правильность вводимых сиволов. Убедитесь, что есть выбор специальности",Toast.*LENGTH\_SHORT*).show()  
 }  
 **}** }  
 private fun loadGroups() {  
 *lifecycleScope*.*launch*(Dispatchers.IO) **{** val groupsFromDb = databaseDao.getAllSpecialities()  
 val groupNames = groupsFromDb.*map* **{ it**.speciality **}** val groupNamesList = ArrayList<String>(groupNames)  
  
 withContext(Dispatchers.Main) **{** val newAdapter = ArrayAdapter(  
 this@AddStudent,  
 R.layout.*simple\_spinner\_item*,  
 groupNamesList  
 ).*apply* **{** setDropDownViewResource(R.layout.*simple\_spinner\_dropdown\_item*)  
 **}** binding.groupSpinner.*adapter* = newAdapter  
 **}  
  
 }** }  
  
 private suspend fun isStudentUnique(student: Student): Boolean {  
 val studentsList = databaseDao.searchStudentsByName(student.name)  
 var hasBudgetPlace = false  
  
 for (existingStudent in studentsList) {  
 if(existingStudent.speciality==student.speciality) {  
 error = "Студент уже учится на этой специальности"  
 return false  
 }  
 if(existingStudent.isBudget)  
 hasBudgetPlace=true;  
 }  
 if(student.isBudget && hasBudgetPlace){  
 error = "У студента уже бюджетное место"  
 return false  
 }  
  
 // Нет совпадений, студент уникален  
 return true  
 }

class AddTeacher : AppCompatActivity() {  
 private lateinit var teachernameEditText:EditText  
 private lateinit var specialitiesSpinner:Spinner  
 private lateinit var hourstechersr:EditText  
 private lateinit var addTecherButton:Button  
 private lateinit var databaseDao: DatabaseDao  
 private lateinit var binding: ActivityAddTeacherBinding  
 private lateinit var error:String  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 binding = ActivityAddTeacherBinding.inflate(*layoutInflater*)  
 setContentView(binding.*root*)  
 databaseDao = CollegeDatabase.getDatabase(this).databaseDao()  
  
  
 val groupAdapter = ArrayAdapter(  
 this,  
 R.layout.*simple\_spinner\_item*,  
 *emptyArray*<String>()  
 ).*apply* **{** setDropDownViewResource(R.layout.*simple\_spinner\_dropdown\_item*)  
 **}** binding.specialitiesSpinner.*adapter* = groupAdapter  
 loadGroups()  
  
  
  
 binding.addTecherButton.setOnClickListener**{** try{  
  
  
 if(binding.teachernameEditText.*text*.*isNotBlank*() && binding.hourstechers.*text*.*isNotBlank*())  
 {  
 if(binding.hourstechers.*text*.*isNotBlank*() && binding.hourstechers.*text*.toString().*toInt*() <= 2 || binding.hourstechers.*text*.toString().*toInt*() > 4000)  
 {  
 Toast.makeText(this,"Введите корректные часы",Toast.*LENGTH\_SHORT*).show()  
 }  
 else  
 {  
 addTeacher()  
 Toast.makeText(this,"Вы добавили преподавателя",Toast.*LENGTH\_SHORT*).show()  
 val intent = Intent(this,AdmissionCommittee::class.*java*)  
 startActivity(intent)  
 }  
 }  
 else  
 {  
 Toast.makeText(this,"Введите все данные",Toast.*LENGTH\_SHORT*).show()  
 }  
 }catch (ex:Exception){  
 Toast.makeText(this,"Такой преподаватель c такой же специальностью уже есть",Toast.*LENGTH\_SHORT*).show()  
 }  
 **}** }  
  
  
  
  
 fun loadGroups() {  
 *lifecycleScope*.*launch*(Dispatchers.IO) **{** val groupsFromDb = databaseDao.getAllSpecialities()  
 val groupNames = groupsFromDb.*map* **{ it**.speciality **}** val groupNamesList = ArrayList<String>(groupNames)  
  
  
 withContext(Dispatchers.Main) **{** val newAdapter = ArrayAdapter(  
 this@AddTeacher,  
 R.layout.*simple\_spinner\_item*,  
 groupNamesList  
 ).*apply* **{** setDropDownViewResource(R.layout.*simple\_spinner\_dropdown\_item*)  
 **}** binding.specialitiesSpinner.*adapter* = newAdapter  
 **}  
 }** }  
  
 private fun addTeacher() {  
 try {  
 if (binding.specialitiesSpinner.*selectedItemPosition* == -1 || binding.teachernameEditText.*text*.toString()  
 .*isEmpty*() || binding.hourstechers.*text*.*isEmpty*()  
 ) {  
 Toast.makeText(  
 this@AddTeacher,  
 "Вы не заполнили все поля",  
 Toast.*LENGTH\_SHORT* )  
 .show()  
 return  
 }  
 if (binding.hourstechers.*text*.toString().*toInt*() < 0) {  
 Toast.makeText(  
 this@AddTeacher,  
 "Количество часов должно быть положительным",  
 Toast.*LENGTH\_SHORT* ).show()  
 return  
 }  
 *lifecycleScope*.*launch*(Dispatchers.IO) **{** val teacher = Teacher(  
 name = binding.teachernameEditText.*text*.toString(),  
 speciality = binding.specialitiesSpinner.*selectedItem*.toString(),  
 salary = 0.0,  
 hoursPerYear = binding.hourstechers.*text*.toString().*toInt*()  
 )  
 if (isTeacherUnique(teacher)) {  
 databaseDao.insertTeacher(teacher)  
 finish()  
 } else {  
 runOnUiThread **{** Toast.makeText(this@AddTeacher, error, Toast.*LENGTH\_SHORT*).show()  
 **}** }  
 **}** } catch (e: Exception) {  
 Toast.makeText(this@AddTeacher, "Вы ввели не число", Toast.*LENGTH\_SHORT*).show()  
 }  
  
 }  
  
 private suspend fun isTeacherUnique(teacher: Teacher): Boolean {  
 val teacherList = databaseDao.searchTeachersByName(teacher.name)  
 for (item in teacherList) {  
 if (item.speciality == teacher.speciality) {  
 error = "Преподаватель уже ведет на эктой специальности"  
 return false  
 }  
 }  
 return true;  
 }

class AdmissionCommittee : AppCompatActivity() {  
 private lateinit var bottomNavigation: BottomNavigationView  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 setContentView(R.layout.*activity\_admission\_committee*)  
  
 bottomNavigation = findViewById(R.id.*bottomNavigationView*)  
 loadFragment(StudentsFragment())  
 bottomNavigation.setOnNavigationItemSelectedListener **{** item **->** when (item.*itemId*) {  
 R.id.*navigation\_students* -> loadFragment(StudentsFragment())  
 R.id.*navigation\_teachers* -> loadFragment(TeachersFragment())  
 R.id.*navigation\_specialities*->loadFragment(SpecialtiesFragment())  
 }  
 true  
 **}** }  
 private fun loadFragment(fragment: Fragment) {  
 *supportFragmentManager*.beginTransaction()  
 .replace(R.id.*fragment\_container*, fragment)  
 .commit()  
 }

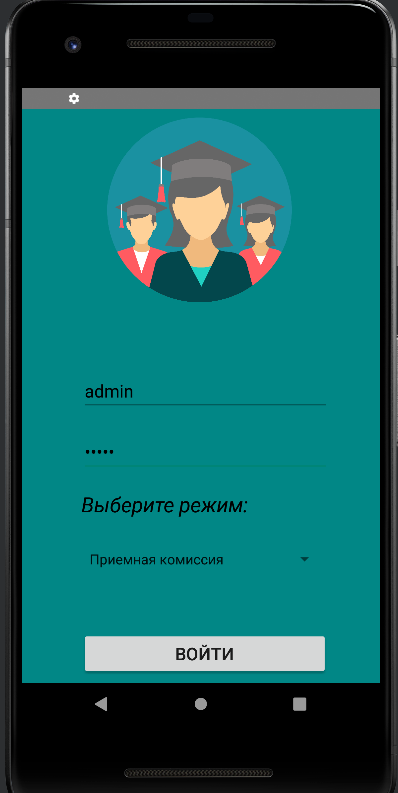
class MainActivity : AppCompatActivity() {  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 setContentView(R.layout.*activity\_main*)  
  
  
 object : CountDownTimer(5000, 1000) {  
 override fun onTick(millisUntilFinished: Long) {  
  
 }  
  
 override fun onFinish() {  
 // Создание Intent для перехода на "SecondActivity"  
 val intent = Intent(this@MainActivity, SignInActivity::class.*java*)  
 startActivity(intent)  
 }  
 }.start()  
 }  
}

class SignInActivity : AppCompatActivity() {  
 private lateinit var login: EditText  
 private lateinit var password: EditText  
 private lateinit var signin: Button  
 private lateinit var spinner: Spinner  
 private lateinit var sharedPref: SharedPreferences  
 private lateinit var editor: SharedPreferences.Editor  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 setContentView(R.layout.*activity\_sign\_in*)  
  
  
 login = findViewById(R.id.*login*)  
 password = findViewById(R.id.*password*)  
 signin = findViewById(R.id.*buttonsignin*)  
 spinner = findViewById(R.id.*spinner*)  
  
 sharedPref = getSharedPreferences("myPrefs", Context.*MODE\_PRIVATE*)  
 editor = sharedPref.edit()  
 // Сохранение логина и пароля для админа  
 editor.putString("admin\_login", "admin")  
 editor.putString("admin\_password", "admin")  
  
 // Сохранение логина и пароля для преподавателя  
 editor.putString("teacher\_login", "teacher")  
 editor.putString("teacher\_password", "teacher")  
  
 // Сохранение логина и пароля для студента  
 editor.putString("student\_login", "student")  
 editor.putString("student\_password", "student")  
  
 editor.apply()  
  
  
  
  
  
 signin.setOnClickListener **{** val enteredLogin = login.*text*.toString()  
 val enteredPassword = password.*text*.toString()  
 //var name = ""  
  
 val userType = spinner.*selectedItem*.toString()  
// when(userType)  
// {  
// "Приемная комиссия" -> name = "admin"  
// "Преподаватель" -> name = "teacher"  
// "Студент" -> name = "student"  
// }  
 val savedLogin = sharedPref.getString("${enteredLogin}\_login", "")?.*trim*()  
 val savedPassword = sharedPref.getString("${enteredPassword}\_password", "")?.*trim*()  
  
  
 if(enteredLogin.*isNotBlank*() || enteredPassword.*isNotBlank*()) {  
 if (enteredLogin.*trim*() == savedLogin && enteredPassword.*trim*() == savedPassword) {  
 val lastUserType = sharedPref.getString("last\_user\_type", "")  
  
 when (enteredLogin) {  
 "admin" -> {  
 // Для администратора разрешаем доступ к любому типу активности  
 val intent = when (userType) {  
 "Приемная комиссия" -> Intent(this, AdmissionCommittee::class.*java*)  
 "Преподаватель" -> Intent(this, TeacherActivity::class.*java*)  
 "Студент" -> Intent(this, StudentActivity::class.*java*)  
 else -> null  
 }  
 if (intent != null) {  
 startActivity(intent)  
 editor.putString("last\_user\_type", userType)  
 editor.apply()  
 } else {  
 Toast.makeText(  
 this,  
 "Ошибка в выборе режима",  
 Toast.*LENGTH\_SHORT* )  
 .show()  
 }  
 }  
 "teacher" -> {  
 // Для преподавателя разрешаем доступ только к экрану преподавателя  
 if (userType == "Преподаватель") {  
 startActivity(Intent(this, TeacherActivity::class.*java*))  
 editor.putString("last\_user\_type", userType)  
 editor.apply()  
 } else {  
 Toast.makeText(  
 this,  
 "Ошибка в выборе режима",  
 Toast.*LENGTH\_SHORT* )  
 .show()  
 }  
 }  
 "student" -> {  
 // Для студента разрешаем доступ только к экрану студента  
 if (userType == "Студент") {  
 startActivity(Intent(this, StudentActivity::class.*java*))  
 editor.putString("last\_user\_type", userType)  
 editor.apply()  
 } else {  
 Toast.makeText(  
 this,  
 "Ошибка в выборе режима",  
 Toast.*LENGTH\_SHORT* )  
 .show()  
 }  
 }  
 else -> {  
 // Предыдущий тип пользователя не найден  
 Toast.makeText(  
 this,  
 "Ошибка в определении предыдущего типа пользователя",  
 Toast.*LENGTH\_SHORT* ).show()  
  
  
 }  
 }  
 } else {  
 Toast.makeText(this, "Введен неправильный логин или пароль", Toast.*LENGTH\_SHORT*)  
 .show()  
 }  
 }else{  
 Toast.makeText(this, "Введите все данные", Toast.*LENGTH\_SHORT*)  
 .show()  
 }  
  
 **}** }

class StudentActivity : AppCompatActivity() {  
 private lateinit var edittext: EditText  
 private lateinit var studentInfo: RecyclerView  
 private lateinit var databaseDao: DatabaseDao  
 private lateinit var adapter: TeachersInfoAdapter  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 setContentView(R.layout.*activity\_student*)  
 edittext = findViewById(R.id.*editextpoiskprepodovateley*)  
 studentInfo = findViewById(R.id.*studentInfo*)  
  
  
 databaseDao = CollegeDatabase.getDatabase(this).databaseDao()  
 adapter = TeachersInfoAdapter()  
 studentInfo.*layoutManager* = LinearLayoutManager(this)  
 studentInfo.*adapter* = adapter  
  
 *lifecycleScope*.*launch*(Dispatchers.IO) **{** val studentsList = databaseDao.getTeachersWithSpeciality()  
 withContext(Dispatchers.Main) **{** adapter.setData(studentsList)  
 **}  
 }** }  
  
  
 fun searchbutton(view: View) {  
 val fullName = edittext.*text*.toString().*trim*()  
  
 if (!fullName.*isNullOrEmpty*()) {  
 *lifecycleScope*.*launch*(Dispatchers.IO) **{** val studentsList = databaseDao.getTeachersWithSpecialityByName(fullName.toString())  
  
 withContext(Dispatchers.Main) **{** adapter.setData(studentsList)  
 **}  
 }** } else {  
 Toast.makeText(this, "Такого преподавателя нет", Toast.*LENGTH\_SHORT*).show()  
 }  
  
 if (!fullName.*isNotBlank*()) {  
 *lifecycleScope*.*launch*(Dispatchers.IO) **{** val studentsList = databaseDao.getTeachersWithSpeciality()  
 withContext(Dispatchers.Main) **{** adapter.setData(studentsList)  
 **}  
 }** }  
  
  
 }  
  
 fun exit(view: View) {  
 val intent = Intent(this, SignInActivity::class.*java*)  
 startActivity(intent)  
 }  
}

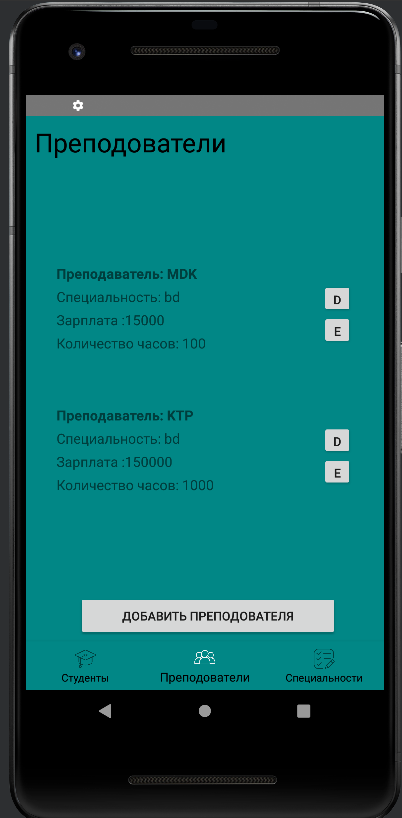
class TeacherActivity : AppCompatActivity() {  
 private lateinit var edittext: EditText  
 private lateinit var teacherInfo: RecyclerView  
 private lateinit var databaseDao: DatabaseDao  
 private lateinit var adapter: StudentsInfoAdapter  
  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 setContentView(R.layout.*activity\_teacher*)  
 edittext = findViewById(R.id.*editTextfiopoiskstudent*)  
 teacherInfo = findViewById(R.id.*teacherInfo*)  
  
  
 databaseDao = CollegeDatabase.getDatabase(this).databaseDao()  
 adapter = StudentsInfoAdapter()  
 teacherInfo.*layoutManager* = LinearLayoutManager(this)  
 teacherInfo.*adapter* = adapter  
  
 *lifecycleScope*.*launch*(Dispatchers.IO) **{** val studentsList = databaseDao.getStudentsWithSpeciality()  
 withContext(Dispatchers.Main) **{** adapter.setData(studentsList)  
 **}  
 }** }  
  
 fun searchstudent(view: View) {  
 val fullName = edittext.*text*.toString().*trim*()  
  
 if (!fullName.*isNullOrEmpty*()) {  
 *lifecycleScope*.*launch*(Dispatchers.IO) **{** val studentsList = databaseDao.getStudentsWithSpecialityByName(fullName.toString())  
  
 withContext(Dispatchers.Main) **{** adapter.setData(studentsList)  
 **}  
 }** } else {  
 Toast.makeText(this@TeacherActivity, "Такого студента нет", Toast.*LENGTH\_SHORT*).show()  
 }  
  
 if (!fullName.*isNotBlank*()) {  
 *lifecycleScope*.*launch*(Dispatchers.IO) **{** val studentsList = databaseDao.getStudentsWithSpeciality()  
 withContext(Dispatchers.Main) **{** adapter.setData(studentsList)  
 **}  
 }** }  
 }  
  
 fun exit(view: View) {  
 val intent = Intent(this, SignInActivity::class.*java*)  
 startActivity(intent)  
 }  
}

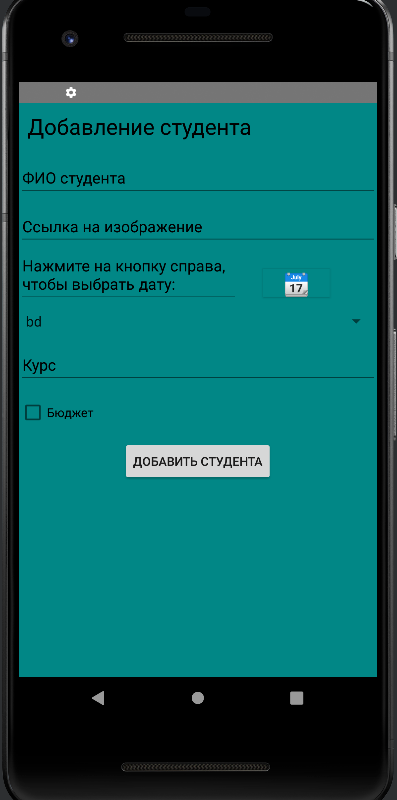
Приложение:

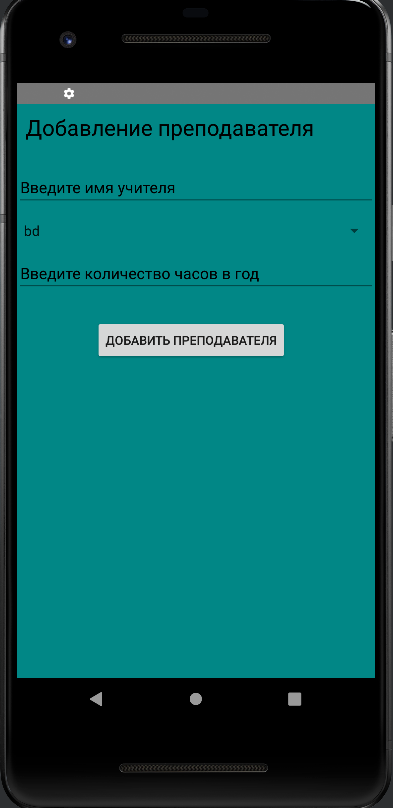
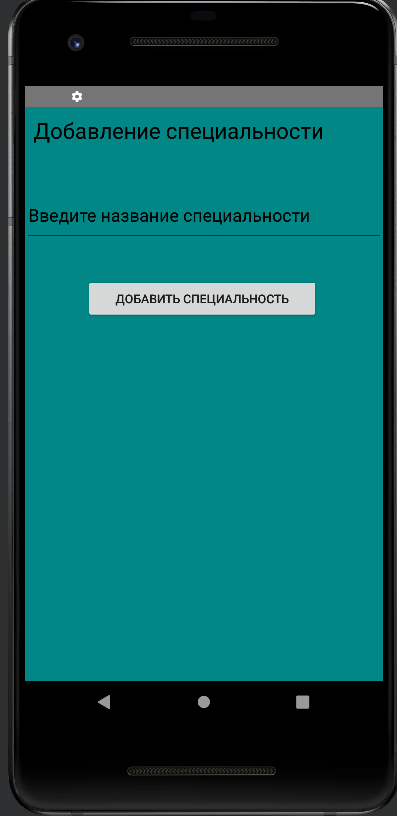


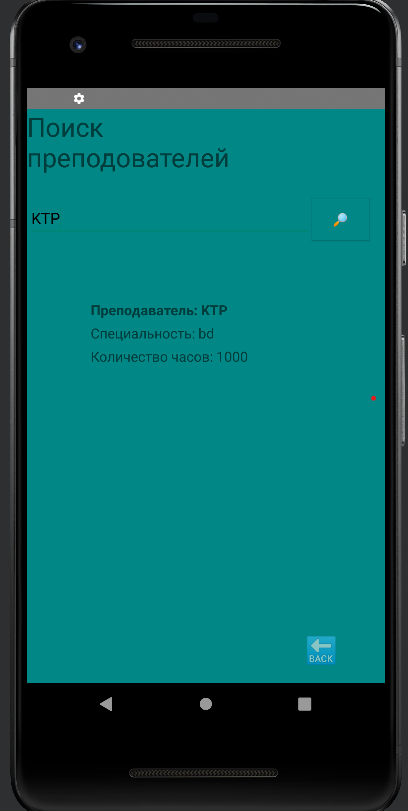
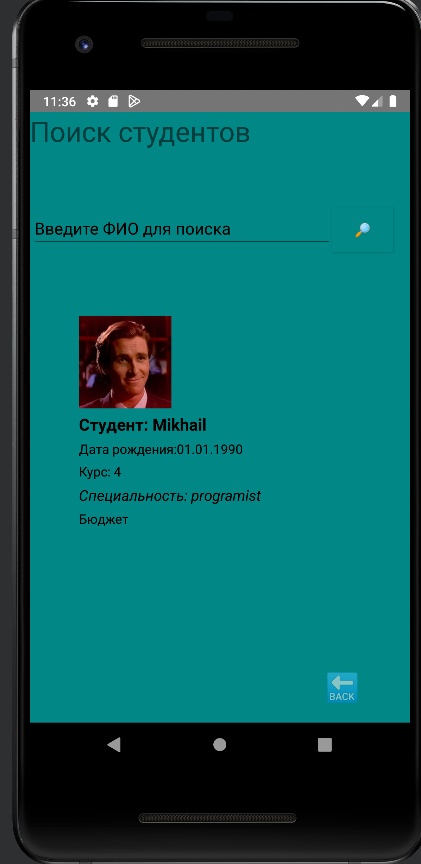


**Тестовые ситуации:**

****

****

****

**** 

**Проверки:**

**1)** Проверка на пустоту (if else)  
**2)** Проверка (try catch)  
**3)** Проверка на правильность ввода(if else)  
**4)** Проверка на правильность выбранной даты

**Вывод:** При выполнении данной практической работы я научился делать большие программы с базой данных.