Министерство образования и молодежной политики Свердловской области



ГАПОУ СО «Екатеринбургский колледж транспортного строительства»

Отчёт по программе «**Приложение – умные часы**»

Выполнил: Ковязин А.М

Группа: ПР-31

Преподаватель: Мирошниченко Г.В

2023

Вариант 7

**Входные данные:**

nameArtist(Strung) – название исполнителя

**Выходные данные:**

**recyclerView –** список

**Toast – сообщение  
AlertDialog - сообщение**

**Листинг программы:**

class Add : Activity() {  
  
 private lateinit var binding: ActivityAddBinding  
 private lateinit var nameArtist:EditText  
  
 private lateinit var titleArtist:TextView  
 private lateinit var singer:TextView  
 private lateinit var add:Button  
  
 private lateinit var database: PlaylistDatabase  
  
  
  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
  
 binding = ActivityAddBinding.inflate(*layoutInflater*)  
 setContentView(binding.*root*)  
  
 nameArtist = findViewById(R.id.*nameArtist*)  
  
 titleArtist = findViewById(R.id.*titleArtist*)  
 singer = findViewById(R.id.*Singer*)  
 add = findViewById(R.id.*add*)  
  
  
 }  
 fun newplaylist(view: View) {  
 database = Room.databaseBuilder(  
 *applicationContext*,  
 PlaylistDatabase::class.*java*,"playlist-database"  
 ).build()  
  
 val nameArtist = nameArtist.*text*.toString()  
 if(nameArtist.*isNotBlank*()) {  
  
 searchArtistId(nameArtist) **{** id **->** getResualt(id)  
 **}** }  
 else  
 {  
 Toast.makeText(this, "Проверьте ввод данных :(", Toast.*LENGTH\_SHORT*).show()  
 }  
 }  
  
  
  
 fun searchArtistId(artistName: String, callback: (String) -> Unit) {  
 val apiKey = "VbokmShmIcueAGWMQNxFTIwylPfNdEVDHnsqxZWW"  
 val searchUrl = "https://api.discogs.com/database/search?q=$artistName&type=artist&token=$apiKey"  
  
 val queue = Volley.newRequestQueue(this)  
  
 val stringRequest = StringRequest(  
 com.android.volley.Request.Method.*GET*, searchUrl,  
 **{** response **->** try {  
 val jsonObject = JSONObject(response)  
 val resultsArray = jsonObject.getJSONArray("results")  
  
 if (resultsArray.length() > 0) {  
 val artistInfo = resultsArray.getJSONObject(0)  
 val artistId = artistInfo.optInt("id")  
 callback(artistId.toString())  
 } else {  
 callback("")  
 }  
 } catch (e: JSONException) {  
 Toast.makeText(this,"Ошибка при загрузке данных",Toast.*LENGTH\_SHORT*).show()  
 callback("")  
  
 }  
 **}**,  
 **{** Log.d("MyLog", "Volley error: ${**it**.message}")  
 callback("") // Если произошла ошибка  
 **}**)  
  
 queue.add(stringRequest)  
 }  
  
  
 fun getResualt(idArtist: String) {  
 if (idArtist.*isEmpty*()) {  
 Toast.makeText(this,"Ошибка при загрузке данных",Toast.*LENGTH\_SHORT*).show()  
  
 } else {  
  
 val url = "https://api.discogs.com/artists/$idArtist?callback=Nirvana"  
 val queue = Volley.newRequestQueue(this)  
  
 val stringRequest = StringRequest(  
 com.android.volley.Request.Method.*GET*, url,  
 **{** response **->** // Обрабатываем JSONP-ответ  
 val jsonpData =  
 response.*substring*(response.*indexOf*("(") + 1, response.*lastIndexOf*(")"))  
 try {  
 val jsonObject = JSONObject(jsonpData)  
 val artistData = jsonObject.getJSONObject("data")  
 val name = artistData.getString("name")  
 val profile = artistData.getString("profile")  
  
  
 var data = Playlist(name = name, singer = profile)  
 GlobalScope.*launch* (Dispatchers.IO) **{** database.playlistDao().insertPlaylist(data)  
 **}** titleArtist.*text* = name  
 singer.*text* = profile  
  
  
  
 } catch (e: JSONException) {  
 Toast.makeText(this,"Ошибка при загрузке данных",Toast.*LENGTH\_SHORT*).show()  
 }  
 **}**,  
 **{** Log.d("MyLog", "Volley error hetresult: ${**it**.message}")  
 **}**)  
  
 queue.add(stringRequest)  
 }  
 }

class HistoryList : Activity() {  
  
 private lateinit var binding: ActivityHistoryListBinding  
 private lateinit var dobavit:Button  
 private lateinit var udalit:Button  
 private lateinit var list:RecyclerView  
 private lateinit var database:PlaylistDatabase  
 private lateinit var adapter: PlaylistAdapter  
  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
  
 binding = ActivityHistoryListBinding.inflate(*layoutInflater*)  
 setContentView(binding.*root*)  
  
 database = Room.databaseBuilder(  
 *applicationContext*,  
 PlaylistDatabase::class.*java*,"playlist-database"  
 )  
 .build()  
  
  
 list = findViewById(R.id.*list*)  
 dobavit = findViewById(R.id.*dob*)  
 udalit = findViewById(R.id.*udal*)  
  
  
 list.*layoutManager* = LinearLayoutManager(this)  
 adapter = PlaylistAdapter()  
 list.*adapter* = adapter  
  
 GlobalScope.*launch* (Dispatchers.IO) **{** val data = database.playlistDao().getRecentPlaylist()  
 withContext(Dispatchers.Main)**{** adapter.submitList(data)  
 **}  
 }** adapter.setOnDeleteClickListener **{** playlist **->** GlobalScope.*launch* (Dispatchers.IO) **{** database.playlistDao().deletePlaylist(playlist)  
  
 val data = database.playlistDao().getRecentPlaylist()  
 withContext(Dispatchers.Main)**{** adapter.submitList(data)  
 **}  
 }  
 }** adapter.setOnEditClickListener **{** playlist **->** EditMenu(playlist) **}** }  
  
 private fun EditMenu(playlist:Playlist)  
 {  
 val mes = AlertDialog.Builder(this)  
 val lay = LinearLayout(this)  
  
 lay.*orientation* = LinearLayout.*VERTICAL* val name = playlist.name  
 val singer = playlist.singer  
 mes.setTitle("Редактор")  
 val ed1 = EditText(this)  
 val ed2 = EditText(this)  
 ed1.*hint* = "Имя артиста"  
 ed2.*hint* = "Описание"  
 ed1.setText("${name}")  
 ed2.setText("${singer}")  
 lay.addView(ed1)  
 lay.addView(ed2)  
 mes.setView(lay)  
 mes.setPositiveButton("Сохранить") **{**\_,\_ **->** val newname = ed1.*text*.toString()  
 val newsinger = ed2.*text*.toString()  
 if(newname.*isEmpty*() || newsinger.*isEmpty*() )  
 {  
 val alertDialog1 = AlertDialog.Builder(this)  
 alertDialog1.setTitle("Ошибка")  
 alertDialog1.setMessage("Заполните все поля")  
 alertDialog1.setPositiveButton("ОК") **{**\_,\_ **->** showAddDialog()  
 **}** val alertDialog2 = alertDialog1.create()  
 alertDialog2.show()  
  
 }  
 else  
 {  
 val update = playlist.copy(name = newname, singer = newsinger)  
 GlobalScope.*launch*(Dispatchers.IO) **{** database.playlistDao().updatePlayslists(update)  
 val data = database.playlistDao().getRecentPlaylist()  
 withContext(Dispatchers.Main) **{** adapter.submitList(data)  
 **}  
 }** }  
 **}** mes.setNegativeButton("Нет", DialogInterface.OnClickListener**{** dialog, which**->** Toast.makeText(this,"Окс", Toast.*LENGTH\_LONG*).show()  
 **}**)  
 mes.show()  
  
 }  
  
 fun dobavit(view: View) {  
 showAddDialog()  
  
 }  
 private fun showAddDialog(){  
 val alertDialog = AlertDialog.Builder(this)  
 alertDialog.setTitle("Добавление")  
 val input = LinearLayout(this)  
 input.*orientation* = LinearLayout.*VERTICAL* val nameEdit = EditText(this)  
 nameEdit.*hint* = "Название"  
 val singerEdit = EditText(this)  
 singerEdit.*hint* = "Описание"  
  
 input.addView(nameEdit)  
 input.addView(singerEdit)  
 alertDialog.setView(input)  
  
 alertDialog.setPositiveButton("Добавить") **{**\_,\_ **->** val nameSt = nameEdit.*text*.toString()  
 val singerSt = singerEdit.*text*.toString()  
  
 if(nameSt.*isNotBlank*() || singerSt.*isNotBlank*())  
 {  
 val newString = Playlist(name = nameSt, singer = singerSt)  
 GlobalScope.*launch*(Dispatchers.IO) **{** database.playlistDao().insertPlaylist(newString)  
 val playlists = database.playlistDao().getRecentPlaylist()  
 withContext(Dispatchers.Main) **{** adapter.submitList(playlists)  
 **}  
 }** }  
 else  
 {  
 val alertDialog1 = AlertDialog.Builder(this)  
 alertDialog1.setTitle("Ошибка")  
 alertDialog1.setMessage("Заполните все поля")  
 alertDialog1.setPositiveButton("ОК") **{**\_,\_ **->** showAddDialog()  
 **}** val alertDialog2 = alertDialog1.create()  
 alertDialog2.show()  
 }  
  
 **}** alertDialog.setNegativeButton("Отмена") **{**dialog, \_ **->** dialog.dismiss()  
 **}** val alertDialog2 = alertDialog.create()  
 alertDialog2.show()  
  
 }  
 fun udalit(view: View) {  
 val mes = AlertDialog.Builder(this)  
 mes.setTitle("Вопрос")  
 mes.setMessage("Вы хотите удалить все данные?")  
 mes.setPositiveButton("YES OF COURSE", DialogInterface.OnClickListener **{** dialog, which**->** GlobalScope.*launch* (Dispatchers.Main) **{** database.playlistDao().deleteAllPlayslists()  
  
 val data = database.playlistDao().getRecentPlaylist()  
 withContext(Dispatchers.Main)  
 **{** adapter.submitList(data)  
 **}  
 }  
 }**)  
 mes.setNegativeButton("Не надо", DialogInterface.OnClickListener**{** dialog, which**->** Snackbar.make(view,"Окс",Snackbar.*LENGTH\_LONG*)  
 .show()  
 **}**)  
 mes.show()  
 }  
  
}

class MainActivity : Activity() {  
  
 private lateinit var binding: ActivityMainBinding  
 lateinit var login: EditText  
 lateinit var password: EditText  
 lateinit var shar: SharedPreferences  
  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
  
 binding = ActivityMainBinding.inflate(*layoutInflater*)  
 setContentView(binding.*root*)  
 login = findViewById(R.id.*login*)  
 password = findViewById(R.id.*password*)  
 shar = getSharedPreferences("data", *MODE\_PRIVATE*)  
 var correctlogin = shar.getString("login", null)  
 var correctpassword = shar.getString("password", null)  
 if (correctlogin != null && correctpassword != null)  
 {  
 login.setText(correctlogin)  
 password.setText(correctpassword)  
 }  
 }  
  
 fun toScreen(view: View) {  
 var log = login.*text*.toString()  
 var pass = password.*text*.toString()  
 shar = getSharedPreferences("data", *MODE\_PRIVATE*)  
 if (log.*isEmpty*() || pass.*isEmpty*())  
 {  
 Toast.makeText(this, "Введите логин и пароль", Toast.*LENGTH\_SHORT*).show()  
 }  
 else if (pass.length < 5)  
 {  
 Toast.makeText(this, "Для ввода пароля необходимо минимум 5 символов", Toast.*LENGTH\_SHORT*).show()  
 }  
 else {  
 shar = getSharedPreferences("data", *MODE\_PRIVATE*)  
 var correctlogin = shar.getString("login", null)  
 var correctpassword = shar.getString("password", null)  
 if (correctlogin != null && correctpassword != null) {  
 if (correctlogin == log && correctpassword == pass) {  
 val intent = Intent(this, MainScreen::class.*java*)  
 startActivity(intent)  
 } else {  
 Toast.makeText(this, "Неверный логин или пароль", Toast.*LENGTH\_SHORT*).show()  
 }  
 } else {  
 var edit = shar.edit()  
 edit.putString("login", log)  
 edit.putString("password", pass)  
 edit.apply()  
 }  
 }  
 }  
}

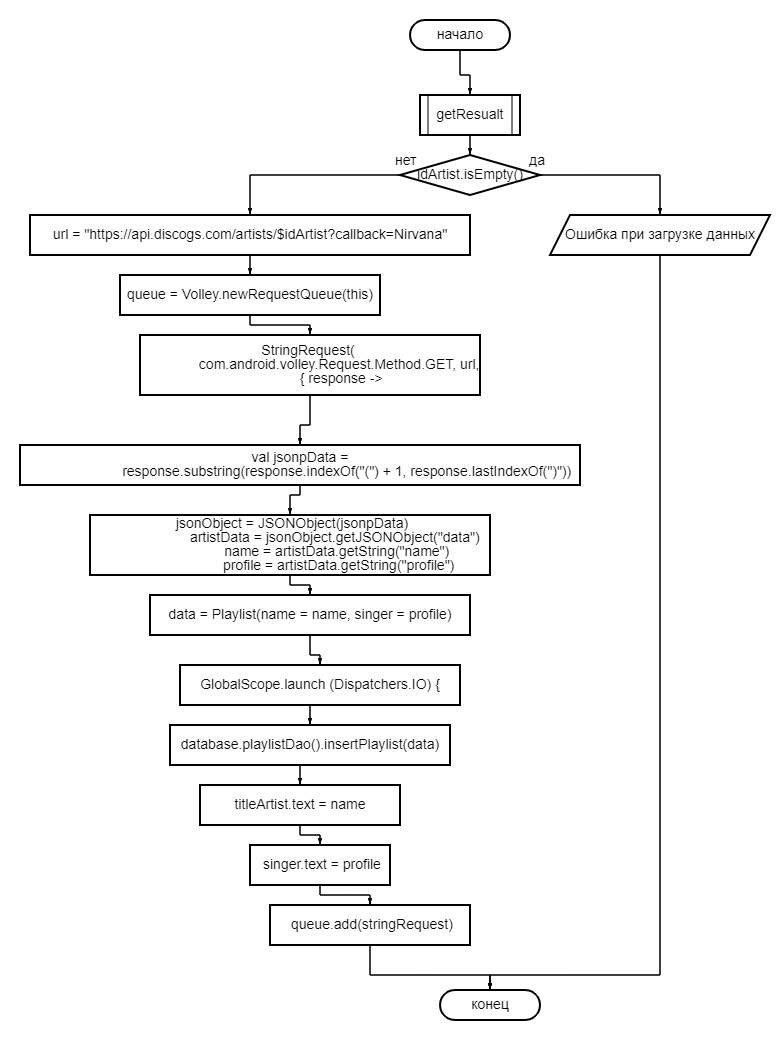
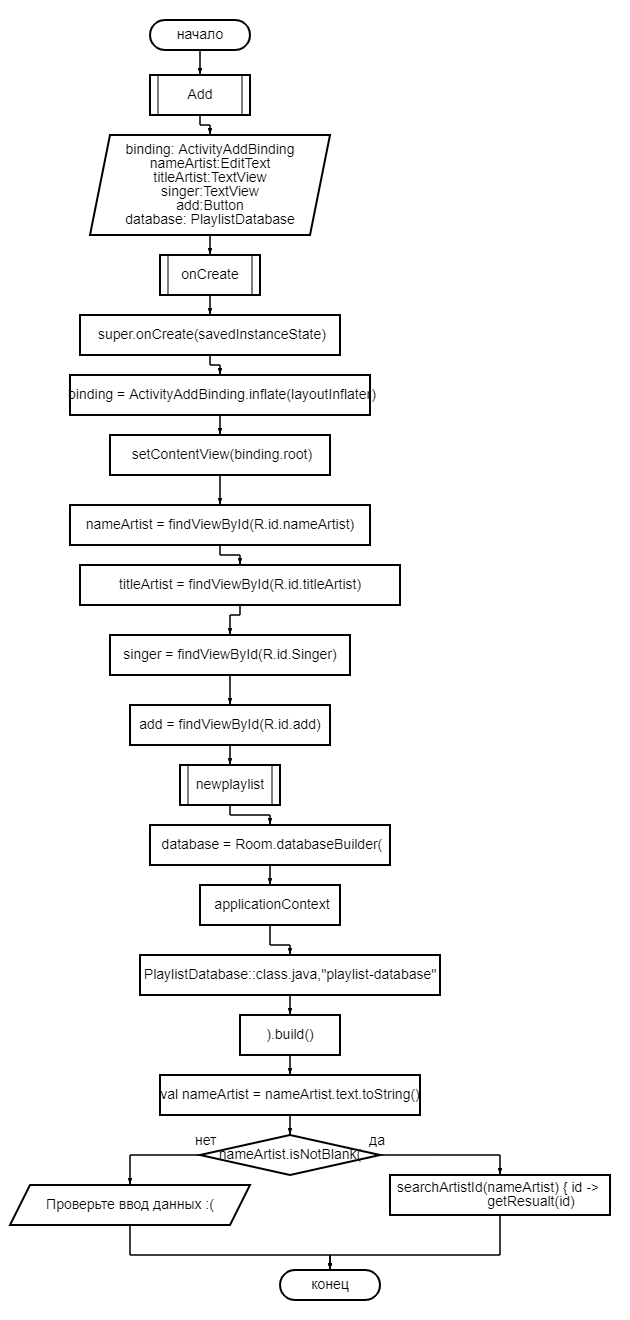
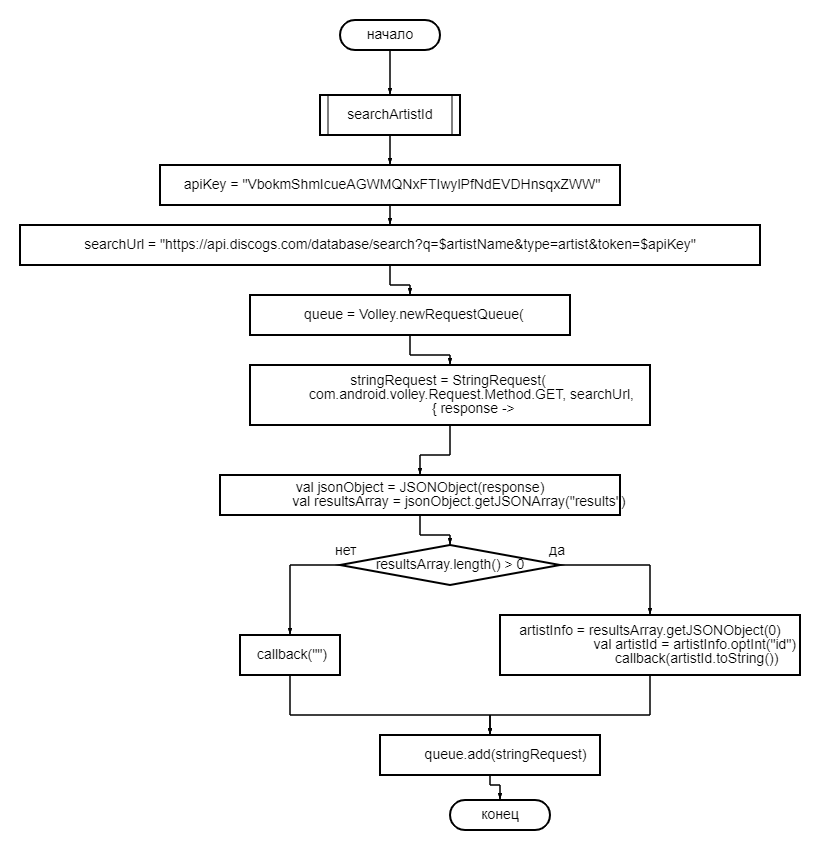
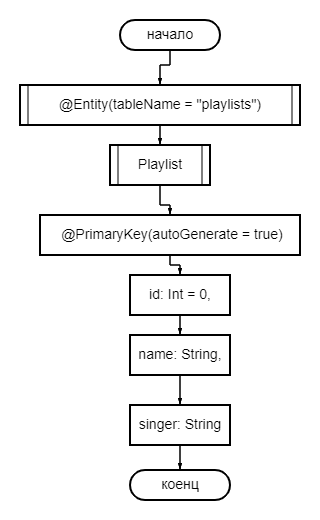
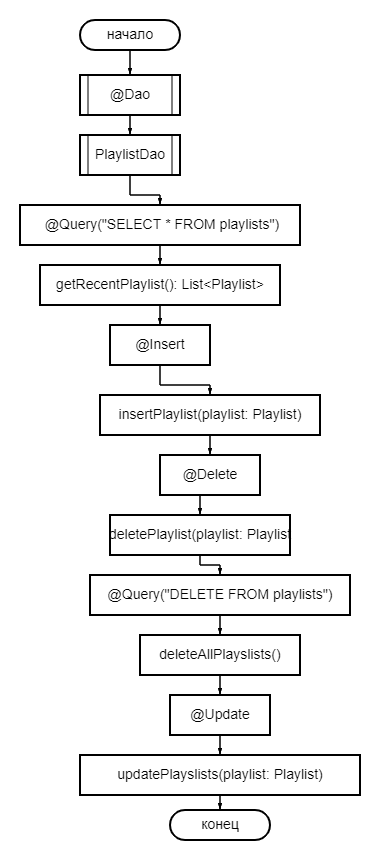
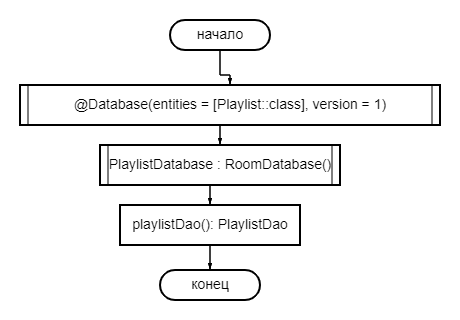
class MainScreen : Activity() {  
  
 private lateinit var binding: ActivityMainScreenBinding  
  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
  
 binding = ActivityMainScreenBinding.inflate(*layoutInflater*)  
 setContentView(binding.*root*)  
  
 }  
  
 fun add(view: View) {  
 val intent = Intent(this,Add::class.*java*)  
 startActivity(intent)  
 }  
 fun history(view: View) {  
 val intent = Intent(this,HistoryList::class.*java*)  
 startActivity(intent)  
 }  
}

@Entity(tableName = "playlists")  
data class Playlist(  
 @PrimaryKey(autoGenerate = true)  
 val id: Int = 0,  
 val name: String,  
 val singer: String  
  
)

class PlaylistAdapter: ListAdapter<Playlist, PlaylistAdapter.PlaylistViewHolder>(DiffCallback()) {  
 private var onDeleteClickListener: ((Playlist) -> Unit)? = null  
 private var onEditClickListener: ((Playlist) -> Unit)? = null  
  
 override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): PlaylistViewHolder {  
 val inflater = LayoutInflater.from(parent.*context*)  
 val view = inflater.inflate(R.layout.*item*, parent, false)  
 return PlaylistViewHolder(view)  
 }  
  
 override fun onBindViewHolder(holder: PlaylistViewHolder, position: Int) {  
 val playlist = getItem(position)  
 holder.bind(playlist, onDeleteClickListener, onEditClickListener)  
  
 }  
 fun setOnDeleteClickListener(listener: (Playlist) -> Unit) {  
 onDeleteClickListener = listener  
 }  
  
 fun setOnEditClickListener(listener: (Playlist) -> Unit) {  
 onEditClickListener = listener  
 }  
 class PlaylistViewHolder(itemView: View) : RecyclerView.ViewHolder(itemView) {  
 private val info1: TextView = itemView.findViewById(R.id.*title*)  
 private val info2: TextView = itemView.findViewById(R.id.*Singer*)  
  
 private val deleteButton: Button = itemView.findViewById(R.id.*delete*)  
 private val redactorButton: Button = itemView.findViewById(R.id.*edit*)  
  
  
 fun bind(playlist: Playlist, onDeleteClickListener: ((Playlist) -> Unit)?, onEditClickListener: ((Playlist) -> Unit)?) {  
 info1.*text* = "${playlist.name}"  
 info2.*text* = "${playlist.singer}"  
  
 deleteButton.setOnClickListener **{** onDeleteClickListener?.invoke(playlist)  
 **}** redactorButton.setOnClickListener **{** onEditClickListener?.invoke(playlist)  
 **}** }  
 }  
  
 private class DiffCallback : DiffUtil.ItemCallback<Playlist>() {  
 override fun areItemsTheSame(oldItem: Playlist, newItem: Playlist): Boolean {  
 return oldItem.id == newItem.id  
 }  
  
 override fun areContentsTheSame(oldItem: Playlist, newItem: Playlist): Boolean {  
 return oldItem == newItem  
 }  
 }

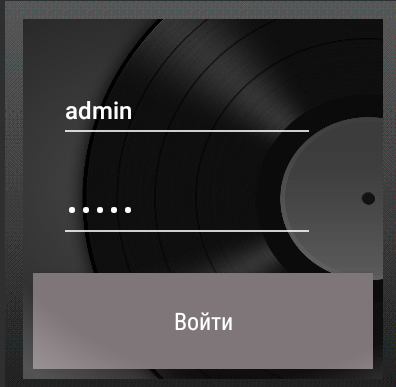
@Dao  
interface PlaylistDao {  
 @Query("SELECT \* FROM playlists")  
 suspend fun getRecentPlaylist(): List<Playlist> //выборка всех записей  
  
 @Insert  
 suspend fun insertPlaylist(movie: Playlist) //вставка в базу данных  
  
 @Delete  
 suspend fun deletePlaylist(movie: Playlist) //удаление из базы данных  
  
 @Query("DELETE FROM playlists")  
 suspend fun deleteAllPlayslists() //удаление всех записей  
  
 @Update  
 suspend fun updatePlayslists(movie: Playlist) //обновление базы данных  
  
  
}

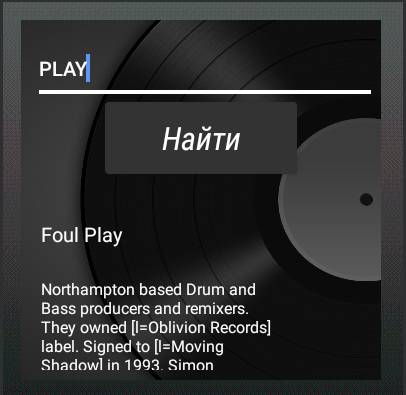
@Database(entities = [Playlist::class], version = 1)  
abstract class PlaylistDatabase : RoomDatabase() {  
 abstract fun playlistDao(): PlaylistDao  
}

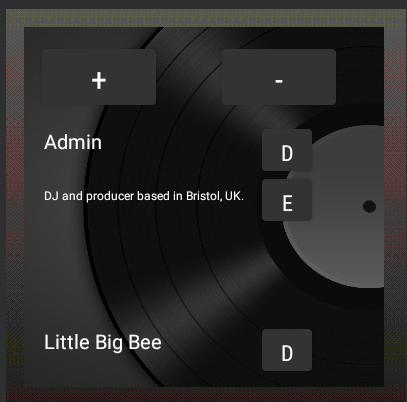
Блок-схема:  
  


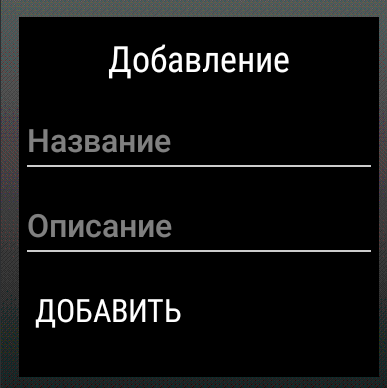


**Программа:**



****

****

****

****

**Тестовые ситуации:**

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

**Вывод:** При выполнении данной практической работы я научился работать с умными часами.