Министерство образования и молодежной политики Свердловской области



ГАПОУ СО «Екатеринбургский колледж транспортного строительства»

# Отчёт по программе «Задание 2. Приложение - умные часы.»

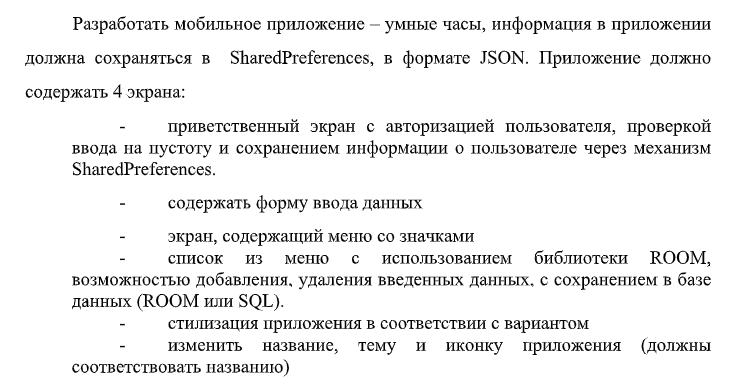
Выполнила: Форрат Лия Константиновна

Группа: ПР-31

Преподаватель: Мирошниченко Г.В

2024

**Вариант 7. Приложение «Исполнители и альбомы»:**



**Входные данные:**

Login – поле для ввода логина

Password – поле для ввода пароля

**Выходные данные**

listContainer – контейнер куда выводится список избранных альбомов

**Описание разработанных функций и методов:**

Методы класса Login:

|  |  |
| --- | --- |
| fun next (view: View) | Метод проверки полученных логина и пароля. Переход на основной экран. |

Методы класса Main:

|  |  |
| --- | --- |
| fun list (view: View) | Метод перехода на страницу с избранным. Получение информации по технологии SharedPreferences |
| fun albums(view: View) | Метод перехода на страницу с альбомами. |
| fun login (view: View) | Метод выхода из приложения, переход на страницу входа. |

Методы класса Albums:

|  |  |
| --- | --- |
| fun fetchAlbums() | Выполняет запрос к API Discogs для получения списка альбомов. |
| fun addToFavorites(album: Album) | Добавляет альбом в избранное (в базу данных). |

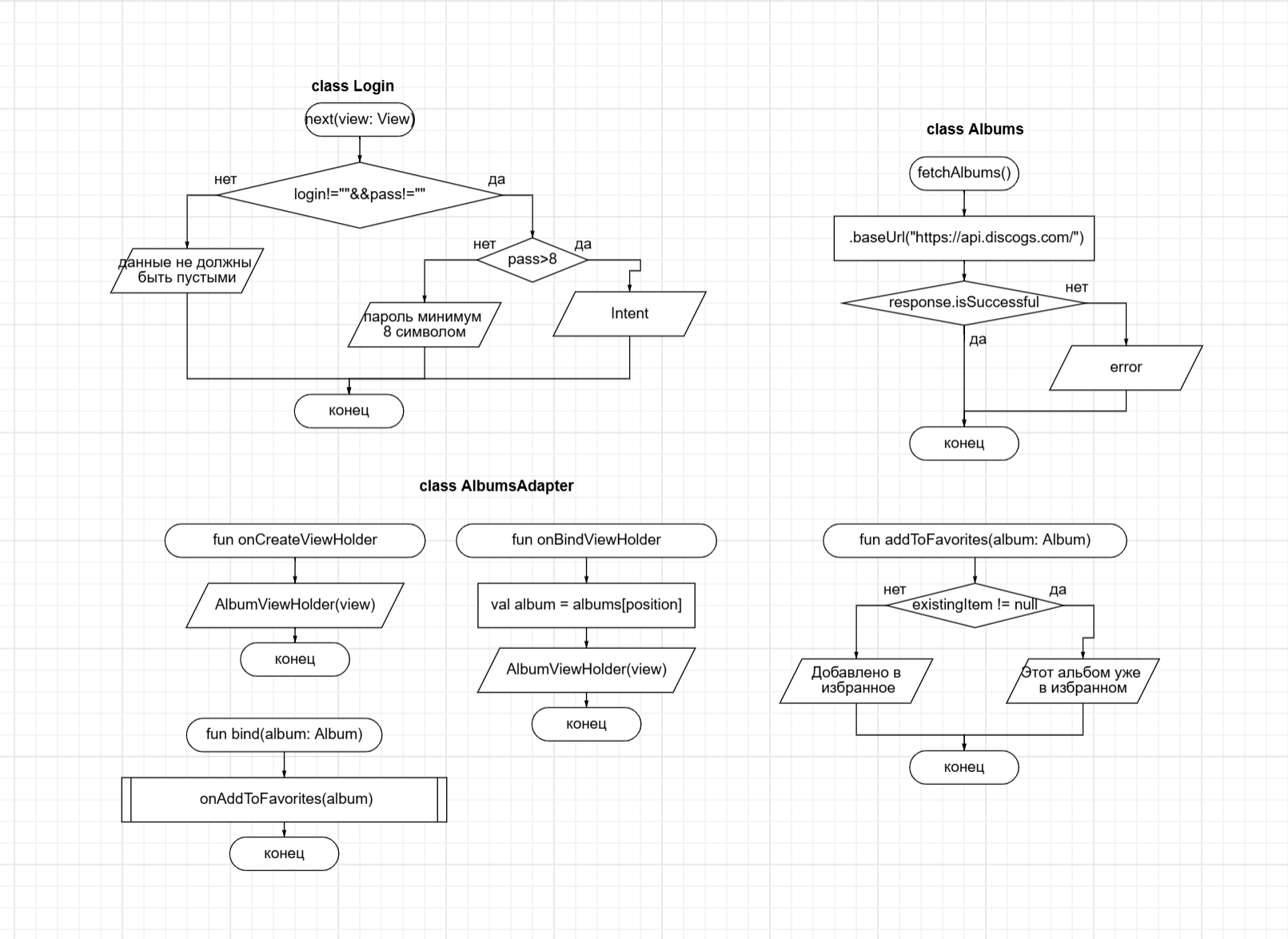
Методы класса AlbumsAdapter:

|  |  |
| --- | --- |
| fun onCreateViewHolder() | Создает новый элемент списка. |
| fun onBindviewholder() | Заполняет элемент списка данными альбома. |
| fun getItemCount() | Возвращает количество альбомов. |
| bind | Привязывает данные альбома к элементу списка. |

Методы класса ListActivity:

|  |  |
| --- | --- |
| private fun LoadItemsFromDatabase() | Получение элементов из базы данных. |
| private fun updateList () | Обновление списка |

**Блок-схема**

****

**Листинг программы:**

**Login.kt**

class Login : AppCompatActivity() {  
 private lateinit var login: EditText  
 private lateinit var pass: EditText  
  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 setContentView(R.layout.*activity\_login*)  
 login = findViewById(R.id.*login*)  
 pass = findViewById(R.id.*password*)  
  
 }  
  
 fun next(view: View) {  
 if (login.*text*.toString().*isNotEmpty*() && pass.*text*.toString().*isNotEmpty*())  
 {  
 if (pass.length() > 8)  
 {  
 val sharedPreferences = getSharedPreferences("UserPrefs", Context.*MODE\_PRIVATE*)  
 val editor = sharedPreferences.edit()  
 editor.putString("username", login.*text*.toString())  
 editor.putString("password", pass.*text*.toString())  
 editor.apply()  
 val intent = Intent(this, Main::class.*java*)  
 startActivity(intent)  
 }  
 else  
 {  
 Snackbar.make(view, "Пароль минимум 8 символов", Snackbar.*LENGTH\_LONG*)  
 .show();  
 }  
  
 }  
 else  
 {  
 Snackbar.make(view, "Данные не должны быть пустые", Snackbar.*LENGTH\_LONG*)  
 .show();  
 }  
  
 }  
}

**Main.kt**

class Main : AppCompatActivity() {  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 setContentView(R.layout.*activity\_main*)  
 }  
  
 fun list(view: View) {  
 val text2 = *intent*.getStringExtra("ITEM\_TEXT")  
 if (text2 != null) {  
 // Сохраняем текст в SharedPreferences  
 val sharedPreferences = getSharedPreferences("MyPrefs", *MODE\_PRIVATE*)  
 val editor = sharedPreferences.edit()  
 val currentText = sharedPreferences.getString("ITEM\_TEXT", "") ?: ""  
 editor.putString("ITEM\_TEXT", currentText + text2 + "\n")  
 editor.apply()  
 }  
  
 // Переход на экран списка  
 val intent = Intent(this, ListActivity::class.*java*)  
 startActivity(intent)  
 }  
  
 fun images(view: View) {  
 val intent = Intent(this, Albums::class.*java*)  
 startActivity(intent)  
 }  
  
  
 fun login(view: View) {  
  
 val intent = Intent(this, Login::class.*java*)  
 startActivity(intent)  
 }  
  
 companion object {  
 var text = ""  
 }  
}

**AlbumsAdapter.kt**

class AlbumsAdapter(  
 private val context: Context,  
 private val albums: List<Album>,  
 private val onAddToFavorites: (Album) -> Unit  
) : RecyclerView.Adapter<AlbumsAdapter.AlbumViewHolder>() {  
  
 override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): AlbumViewHolder {  
 val view = LayoutInflater.from(context).inflate(R.layout.*item\_album*, parent, false)  
 return AlbumViewHolder(view)  
 }  
  
 override fun onBindViewHolder(holder: AlbumViewHolder, position: Int) {  
 val album = albums[position]  
 holder.bind(album)  
 }  
  
 override fun getItemCount(): Int = albums.size  
  
 inner class AlbumViewHolder(itemView: View) : RecyclerView.ViewHolder(itemView) {  
 private val albumTitle: TextView = itemView.findViewById(R.id.*albumTitle*)  
 private val albumArtist: TextView = itemView.findViewById(R.id.*albumArtist*)  
 private val albumCover: ImageView = itemView.findViewById(R.id.*albumCover*)  
 private val btnAddToFavorites: Button = itemView.findViewById(R.id.*btnAddToFavorites*)  
  
 fun bind(album: Album) {  
 albumTitle.*text* = album.title  
 albumArtist.*text* = album.artist  
 Glide.with(context).load(album.coverImage).into(albumCover)  
  
 btnAddToFavorites.setOnClickListener **{** onAddToFavorites(album)  
 **}** }  
 }  
}

**Albums.kt**

class Albums : AppCompatActivity() {  
  
 private lateinit var recyclerView: RecyclerView  
 private lateinit var adapter: AlbumsAdapter  
 private val albums = *mutableListOf*<Album>()  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 setContentView(R.layout.*activity\_albums*)  
  
 recyclerView = findViewById(R.id.*recyclerViewAlbums*)  
 recyclerView.*layoutManager* = LinearLayoutManager(this)  
  
 adapter = AlbumsAdapter(this, albums) **{** album **->** addToFavorites(album)  
 **}** recyclerView.*adapter* = adapter  
  
 fetchAlbums()  
  
 }  
  
 private fun fetchAlbums() {  
 val retrofit = Retrofit.Builder()  
 .baseUrl("https://api.discogs.com/")  
 .addConverterFactory(GsonConverterFactory.create())  
 .build()  
  
 val api = retrofit.create(DiscogsApi::class.*java*)  
 api.searchAlbums("rock").enqueue(object : Callback<DiscogsResponse> {  
 override fun onResponse(call: Call<DiscogsResponse>, response: Response<DiscogsResponse>) {  
 if (response.*isSuccessful*) {  
 val results = response.body()?.results ?: *emptyList*()  
 albums.clear()  
 albums.addAll(results.*map* **{** Album(**it**.id, **it**.title, "Unknown Artist", **it**.cover\_image)  
 **}**)  
 adapter.notifyDataSetChanged()  
 } else {  
 Toast.makeText(this@Albums, "Ошибка: ${response.code()}", Toast.*LENGTH\_SHORT*).show()  
 }  
 }  
  
 override fun onFailure(call: Call<DiscogsResponse>, t: Throwable) {  
 Toast.makeText(this@Albums, "Ошибка загрузки данных", Toast.*LENGTH\_SHORT*).show()  
 }  
 })  
 }  
 private fun addToFavorites(album: Album) {  
 // Инициализация базы данных  
 val database = Room.databaseBuilder(this, AppDatabase::class.*java*, "app\_database")  
 .allowMainThreadQueries()  
 .build()  
  
 // Проверяем, есть ли альбом в избранном  
 val existingItem = database.itemDao().getByName(album.title)  
 if (existingItem != null) {  
 // Если альбом уже есть, показываем сообщение  
 Toast.makeText(this, "Этот альбом уже в избранном", Toast.*LENGTH\_SHORT*).show()  
 } else {  
 // Если альбома нет, добавляем его в избранное  
 database.itemDao().insert(Item(name = album.title))  
 Toast.makeText(this, "Добавлено в избранное", Toast.*LENGTH\_SHORT*).show()  
 }  
 }  
  
  
}

**Album.kt**

data class Album(  
 val id: Int,  
 val title: String,  
 val artist: String,  
 val coverImage: String  
)

**ListActivity.kt**

class ListActivity : AppCompatActivity() {  
 private lateinit var listContainer: LinearLayout  
 private lateinit var database: AppDatabase // Room Database  
 private val items = ArrayList<String>() // Список для хранения элементов  
  
 @SuppressLint("MissingInflatedId")  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 setContentView(R.layout.*activity\_list*)  
  
 listContainer = findViewById(R.id.*listContainer*)  
  
 // Инициализация Room Database  
 database = Room.databaseBuilder(this, AppDatabase::class.*java*, "app\_database").allowMainThreadQueries().build()  
  
 // Получаем все элементы из базы данных  
 loadItemsFromDatabase()  
  
 // Обновляем представление списка  
 updateList()  
 }  
  
 private fun loadItemsFromDatabase() {  
 val itemList = database.itemDao().getAll() // Получаем все элементы из базы данных  
 items.clear()  
 items.addAll(itemList.*map* **{ it**.name **}**) // Добавляем текст в список  
 }  
  
 private fun updateList() {  
 listContainer.removeAllViews() // Очищаем контейнер перед обновлением  
 for (item in items) {  
 val textView = TextView(this)  
 textView.*text* = item  
 textView.setTextColor(*resources*.getColor(android.R.color.*white*))  
 textView.*textSize* = 18f  
 listContainer.addView(textView)  
 }  
 }  
}

**ItemDao.kt**

@Dao  
interface ItemDao {  
 @Insert  
 fun insert(item: Item)  
  
 @Query("SELECT \* FROM items")  
 fun getAll(): List<Item>  
  
 @Query("SELECT \* FROM items WHERE name = :name LIMIT 1")  
 fun getByName(name: String): Item?  
}

**Item.kt**

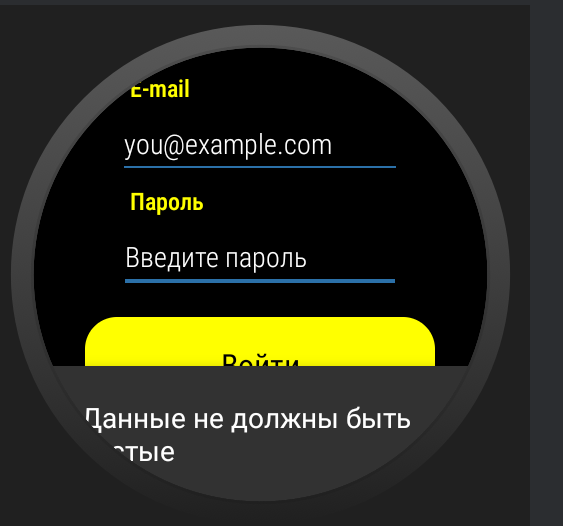
@Entity(tableName = "items")  
data class Item(  
 @PrimaryKey(autoGenerate = true) val id: Long = 0,  
 val name: String  
)

**AppDatabase**

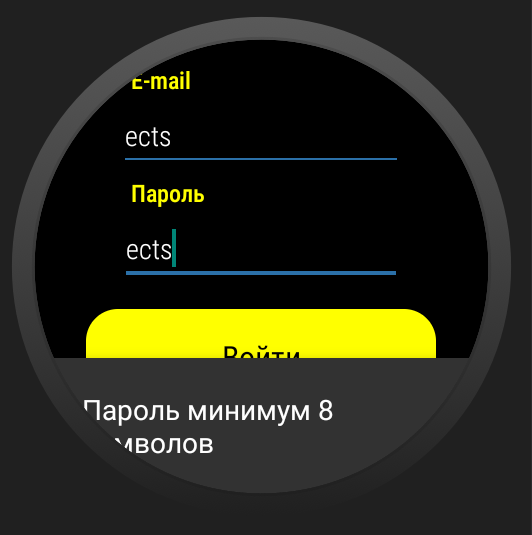
@Database(entities = [Item::class], version = 1)  
abstract class AppDatabase : RoomDatabase() {  
 abstract fun itemDao(): ItemDao  
}

**Тестовые ситуации**

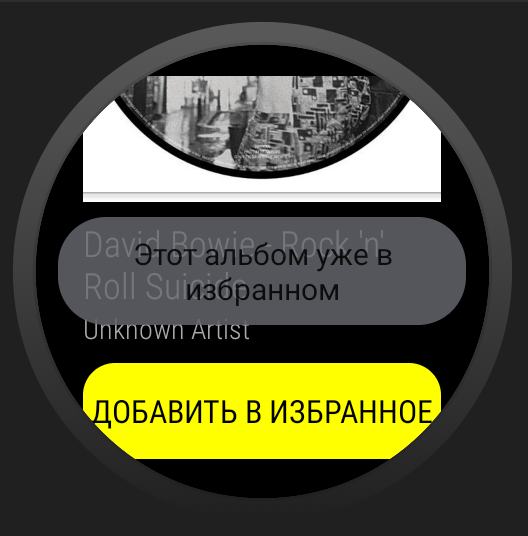
Если пользователь ничего не ввел:

****

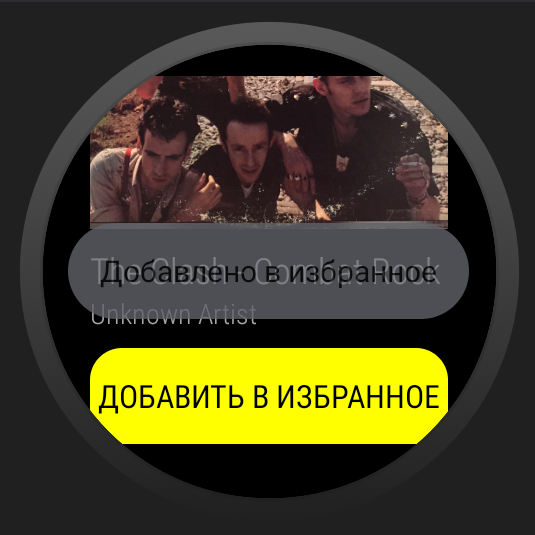
Если пользователь ввел пароль менее 8 символов:



Если пользователь пытается добавить альбом в избранное, который уже есть в избранном:

****

Если пользователь добавляет альбом в избранное, которого еще нет в избранном:

****

**Используемые библиотеки:**

|  |  |
| --- | --- |
| **androidx.room** | Библиотека Room предназначена для хранения данных в Android и работает поверх SQLite, упрощая взаимодействие с базами данных и обеспечивая удобный доступ к данным через объектно-ориентированный подход |
| **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.content** | Этот пакет включает классы, которые предоставляют доступ к различным компонентам приложения, таким как ресурсы, базы данных и системные службы. Он также управляет жизненным циклом приложения и его компонентами. |
| **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. |

**Используемые инструменты:** Android Studio Kotlin.