

Министерство образования и молодежной политики Свердловской области

ГАПОУ СО «Екатеринбургский колледж транспортного строительства»

Отчёт по практическому заданию 22.2

Выполнил: Скворцов И.С.

Группа: ПР-32

Преподаватель: Мирошниченко Г.В.

2023

Содержание

[1. **Задание №1 Приложение с использованием API. Задание 22** 3](#_Toc149554920)

[1.1 Описание задачи 3](#_Toc149554921)

[1.2 Структура проекта 3](#_Toc149554922)

[1.3 Алгоритм решения 11](#_Toc149554923)

[1.5 Используемые библиотеки 17](#_Toc149554924)

[1.6 Тестовые случаи 17](#_Toc149554925)

[1.7 Используемые инструменты 17](#_Toc149554926)

[1.8 Приложение (pr screen экранов) 18](#_Toc149554927)

# 1. **Задание №1 Приложение с использованием API. Задание 22**

## Описание задачи

Разработка приложения работающие с помощью API  ключей.

## Структура проекта

class DatabaseHelper(context: Context) : SQLiteOpenHelper(context, DATABASE\_NAME, null, DATABASE\_VERSION) {

companion object {

private const val DATABASE\_NAME = "UserDatabase"

private const val DATABASE\_VERSION = 1

private const val TABLE\_NAME = "Users"

private const val COLUMN\_ID = "id"

private const val COLUMN\_USERNAME = "username"

private const val COLUMN\_PASSWORD = "password"

}

override fun onCreate(db: SQLiteDatabase) {

val createTableQuery = "CREATE TABLE $TABLE\_NAME ($COLUMN\_ID INTEGER PRIMARY KEY AUTOINCREMENT, $COLUMN\_USERNAME TEXT, $COLUMN\_PASSWORD TEXT);"

db.execSQL(createTableQuery)

}

override fun onUpgrade(db: SQLiteDatabase, oldVersion: Int, newVersion: Int) {

val dropTableQuery = "DROP TABLE IF EXISTS $TABLE\_NAME"

db.execSQL(dropTableQuery)

onCreate(db)

}

fun addUser(username: String, password: String): Long {

val db = this.writableDatabase

val values = ContentValues()

values.put(COLUMN\_USERNAME, username)

values.put(COLUMN\_PASSWORD, password)

return db.insert(TABLE\_NAME, null, values)

}

fun checkUser(username: String, password: String): Boolean {

val db = this.readableDatabase

val selection = "$COLUMN\_USERNAME = ? AND $COLUMN\_PASSWORD = ?"

val selectionArgs = arrayOf(username, password)

val cursor: Cursor = db.query(TABLE\_NAME, null, selection, selectionArgs, null, null, null)

val count = cursor.count

cursor.close()

return count > 0

}

}

class MainActivity : AppCompatActivity() {

private lateinit var usernameEditText: EditText

private lateinit var passwordEditText: EditText

private lateinit var loginButton: Button

private lateinit var registerButton: Button

private lateinit var dbHelper: DatabaseHelper

private lateinit var imageView: ImageView

private lateinit var randomNumbers: IntArray

private val random = Random()

override fun onCreate(savedInstanceState: Bundle?) {

super.onCreate(savedInstanceState)

setContentView(R.layout.activity\_main)

imageView = findViewById(R.id.photopokemonsrandom)

randomNumbers = resources.getIntArray(R.array.random\_numbers)

val randomIndex = random.nextInt(randomNumbers.size)

val randomNumber = randomNumbers[randomIndex]

val imageName = "photo$randomNumber"

val imageResource = resources.getIdentifier(imageName, "drawable", packageName)

if (imageResource != 0) {

imageView.setImageResource(imageResource)

}

dbHelper = DatabaseHelper(this)

usernameEditText = findViewById(R.id.editText)

passwordEditText = findViewById(R.id.editText2)

loginButton = findViewById(R.id.button2)

registerButton = findViewById(R.id.button)

loginButton.setOnClickListener {

val username = usernameEditText.text.toString()

val password = passwordEditText.text.toString()

if (dbHelper.checkUser(username, password)) {

// Пользователь существует и введенные данные правильные

// Вы можете выполнить вход

val intent = Intent(this, MainActivity2::class.java)

startActivity(intent)

} else {

// Пользователь не существует или введены неверные данные

Toast.makeText(this, "Неверный логин или пароль", Toast.LENGTH\_SHORT).show()

}

}

registerButton.setOnClickListener {

val username = usernameEditText.text.toString()

val password = passwordEditText.text.toString()

if (dbHelper.checkUser(username, password)) {

// Пользователь уже существует

Toast.makeText(this, "Пользователь уже существует", Toast.LENGTH\_SHORT).show()

} else {

val inserted = dbHelper.addUser(username, password)

if (inserted != -1L) {

// Пользователь успешно зарегистрирован

Toast.makeText(this, "Пользователь успешно зарегистрирован", Toast.LENGTH\_SHORT).show()

} else {

// Произошла ошибка при регистрации

Toast.makeText(this, "Ошибка при регистрации", Toast.LENGTH\_SHORT).show()

}

}

}

}

class MainActivity2 : AppCompatActivity() {

private lateinit var editTextPokemon: EditText

private lateinit var buttonSearch: Button

private lateinit var textViewResult: TextView

private lateinit var imageViewPokemon: ImageView

override fun onCreate(savedInstanceState: Bundle?) {

super.onCreate(savedInstanceState)

setContentView(R.layout.activity\_main2)

editTextPokemon = findViewById(R.id.editText3)

buttonSearch = findViewById(R.id.button\_screath)

textViewResult = findViewById(R.id.vivod)

imageViewPokemon = findViewById(R.id.iamgepoklen)

buttonSearch.setOnClickListener {

val pokemonNameOrId = editTextPokemon.text.toString()

searchPokemon(pokemonNameOrId)

}

}

private fun searchPokemon(nameOrId: String) {

GlobalScope.launch(Dispatchers.IO) {

val apiUrl = "https://pokeapi.co/api/v2/pokemon/$nameOrId"

try {

val url = URL(apiUrl)

val connection = url.openConnection() as HttpURLConnection

connection.requestMethod = "GET"

val responseCode = connection.responseCode

if (responseCode == HttpURLConnection.HTTP\_OK) {

val reader = BufferedReader(InputStreamReader(connection.inputStream))

val response = StringBuilder()

var line: String?

while (reader.readLine().also { line = it } != null) {

response.append(line)

}

reader.close()

val jsonObject = JSONObject(response.toString())

val name = jsonObject.getString("name")

val id = jsonObject.getInt("id")

val types = jsonObject.getJSONArray("types")

val typeList = mutableListOf<String>()

for (i in 0 until types.length()) {

val typeObj = types.getJSONObject(i)

val typeName = typeObj.getJSONObject("type").getString("name")

typeList.add(typeName)

}

val abilities = jsonObject.getJSONArray("abilities")

val abilityList = mutableListOf<String>()

for (i in 0 until abilities.length()) {

val abilityObj = abilities.getJSONObject(i)

val abilityName = abilityObj.getJSONObject("ability").getString("name")

abilityList.add(abilityName)

}

val speciesUrl = jsonObject.getJSONObject("species").getString("url")

val speciesInfo = fetchSpeciesInfo(speciesUrl)

val imageUrl = jsonObject.getJSONObject("sprites").getString("front\_default")

runOnUiThread {

val resultText = "Имя: $name\nID: $id\nТипы: ${typeList.joinToString()}\nСпособности: ${abilityList.joinToString()}\nХарактеристики: $speciesInfo"

textViewResult.text = resultText

Picasso.get().load(imageUrl).into(imageViewPokemon)

}

} else {

runOnUiThread {

textViewResult.text = "Покемон не найден"

}

}

connection.disconnect()

} catch (e: Exception) {

runOnUiThread {

textViewResult.text = "Ошибка при выполнении запроса"

}

}

}

}

private fun fetchSpeciesInfo(speciesUrl: String): String {

val connection = URL(speciesUrl).openConnection() as HttpURLConnection

connection.requestMethod = "GET"

val responseCode = connection.responseCode

return if (responseCode == HttpURLConnection.HTTP\_OK) {

val reader = BufferedReader(InputStreamReader(connection.inputStream))

val response = StringBuilder()

var line: String?

while (reader.readLine().also { line = it } != null) {

response.append(line)

}

reader.close()

val jsonObject = JSONObject(response.toString())

val flavorTextEntries = jsonObject.getJSONArray("flavor\_text\_entries")

var englishFlavorText = "Информация о характеристиках не найдена"

for (i in 0 until flavorTextEntries.length()) {

val entry = flavorTextEntries.getJSONObject(i)

val language = entry.getJSONObject("language").getString("name")

val flavorText = entry.getString("flavor\_text")

if (language == "en") {

englishFlavorText = flavorText

break

}

}

englishFlavorText

} else {

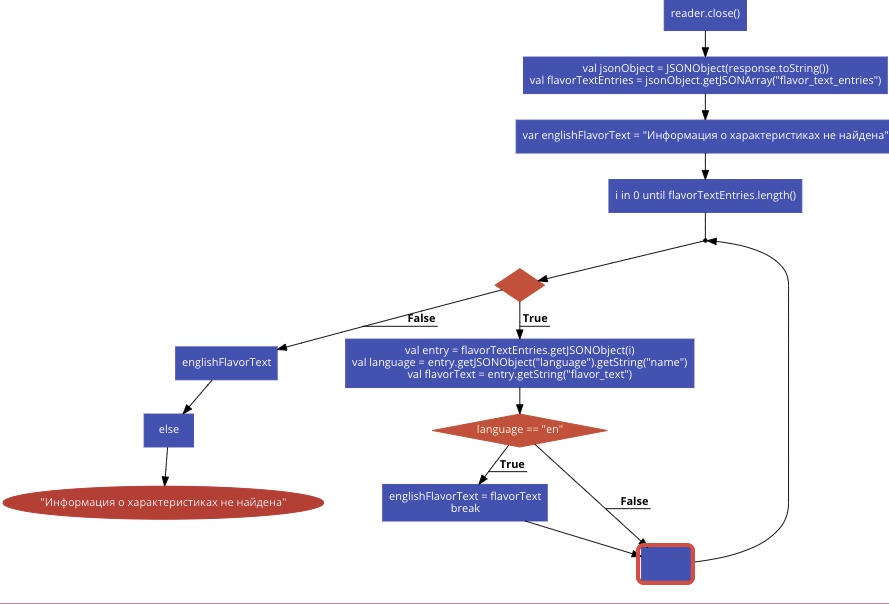
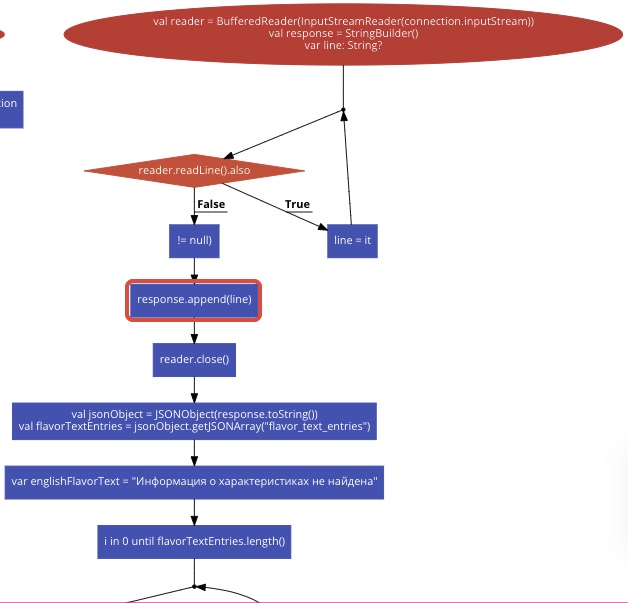
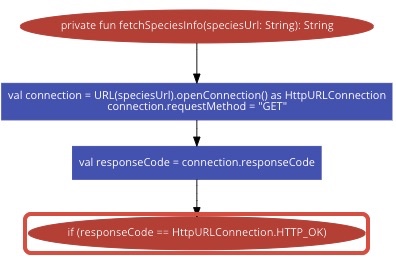
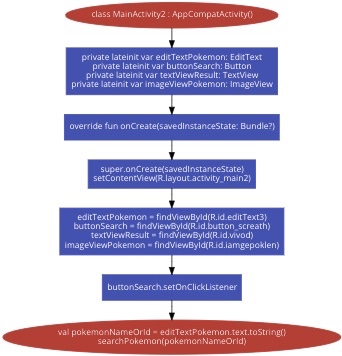
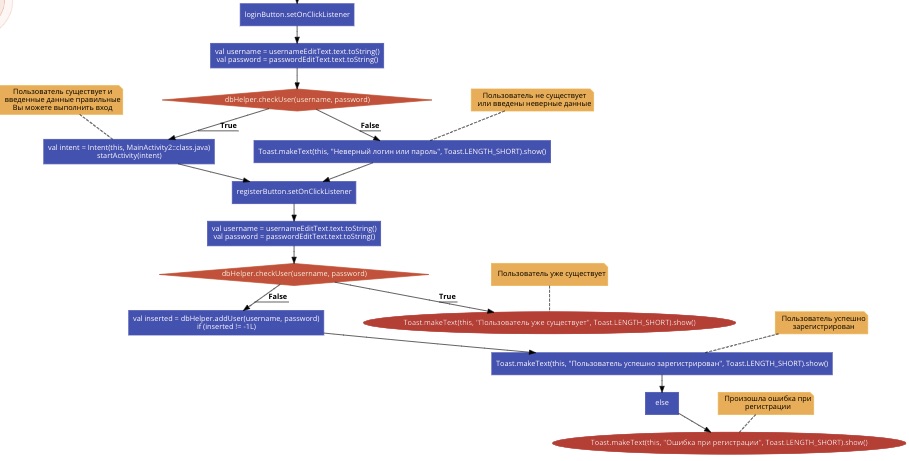
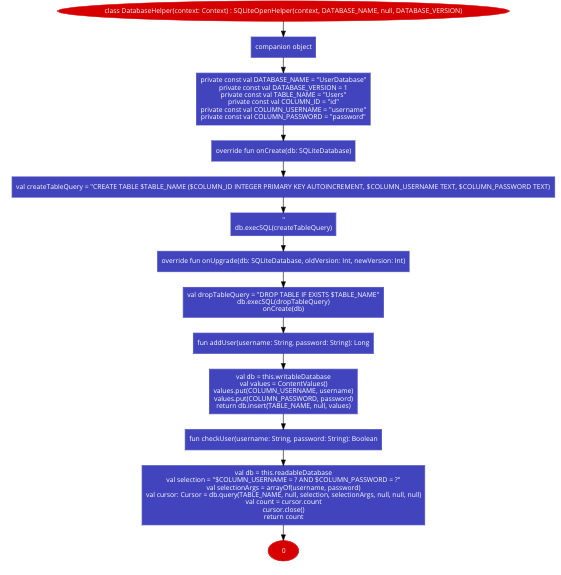
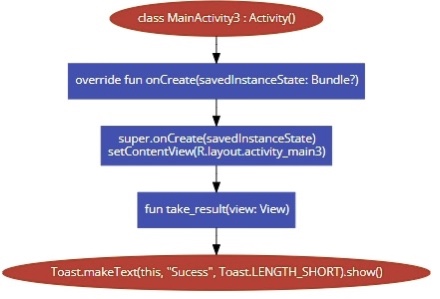
"Информация о характеристиках не найдена"

}

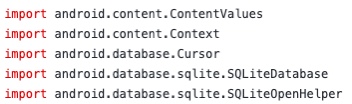
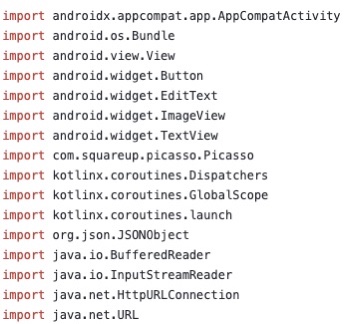
}

}

## Алгоритм решения



## 1.5 Используемые библиотеки



## 1.6 Тестовые случаи

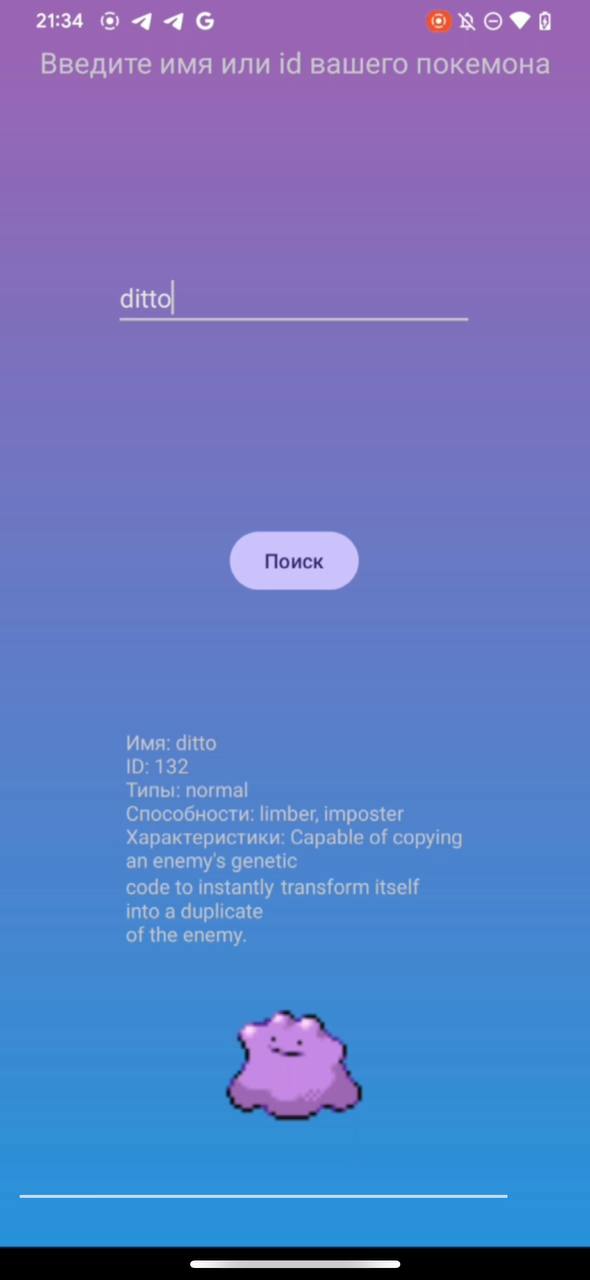
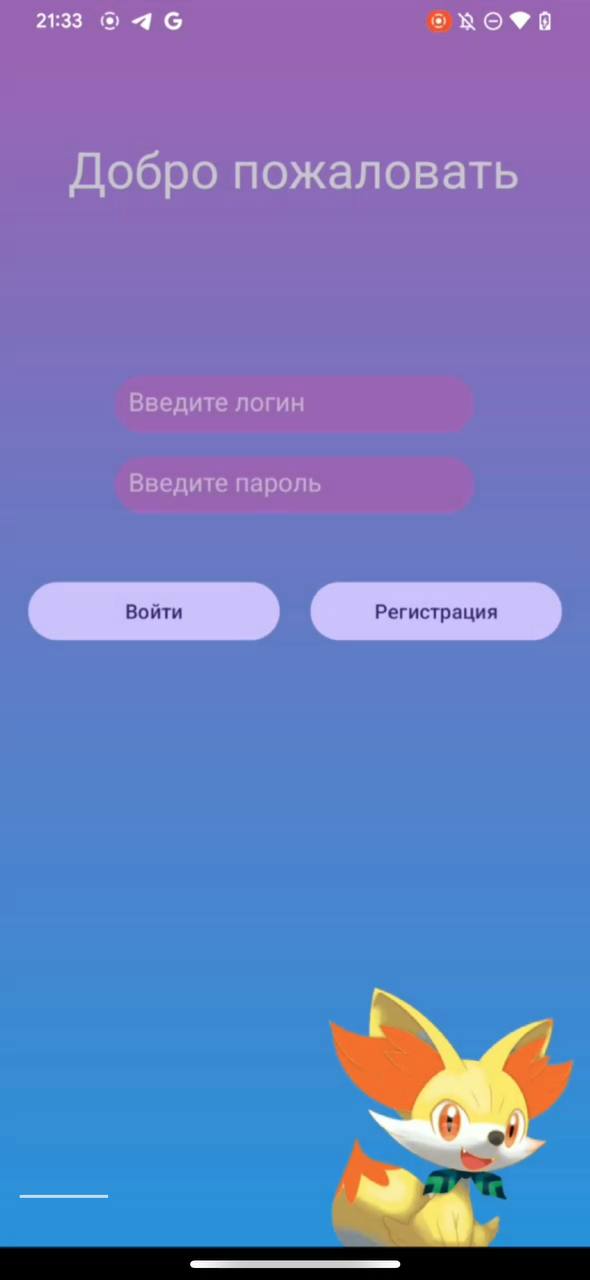
При вводе неправильного логина и пароля – уведомление об ошибке.

При вводе неправильного покемона – ошибка что покемон не найден.

## 1.7 Используемые инструменты

Kotlin, Android Studio

## 1.8 Приложение (pr screen экранов)

****