Министерство образования Республики Беларусь

Учреждение образования  
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных машин

Лабораторная работа №5

«Нативное программирование в Android»

Студент Д. А. Снитко

Проверил О. М. Внук

МИНСК 2024

**1 ЦЕЛЬ РАБОТЫ**

1. Перенести часть разработанного функционала в разработанном приложении в библиотеку на языке Си и подключить посредством Android NDK.

2. Продемонстрировать работоспособность всего заявленного функционала в лабораторной работе №1.

**2 ИСХОДНЫЕ ДАННЫЕ**

Среда разработки для Android;

Язык программирования Kotlin;

Источник исходного кода: https://github.com/Luflexia/Currency-Converter

**3 ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ**

Android NDK (Native Development Kit) - это набор инструментов, позволяющий разработчикам Android-приложений использовать код, написанный на языках C и C++. NDK особенно полезен в ситуациях, когда требуется повысить производительность определенных частей приложения или использовать существующие библиотеки, написанные на C/C++.

Основные преимущества использования NDK- повышение производительности. Некоторые алгоритмы могут работать быстрее при реализации на C/C++. Повторное использование существующего кода: Возможность интеграции существующих C/C++ библиотек. Низкоуровневые операции. Прямой доступ к системным ресурсам и аппаратным возможностям устройства. Процесс интеграции нативного кода в Android-приложение включает следующие шаги. Написание нативных функций на C/C++. Создание заголовочных файлов Java для нативных методов. Компиляция нативного кода в разделяемую библиотеку (.so). Загрузка нативной библиотеки в Java-код. Вызов нативных методов из Java/Kotlin кода.

**4 КОД ПРОГРАММЫ**

Файл user\_validation.cpp

#include <jni.h>

#include <string>

#include <android/log.h>

#include <cstring>

#include <vector>

extern "C" {

// Простая функция для поиска подстроки

const char\* strstr\_custom(const char\* haystack, const char\* needle) {

return strstr(haystack, needle);

}

// Простая функция для извлечения значения из JSON строки

std::string extract\_value(const char\* json, const char\* key) {

std::string result;

const char\* start = strstr\_custom(json, key);

if (start) {

start = strstr\_custom(start, ":");

if (start) {

start++; // пропускаем ':'

while (\*start == ' ' || \*start == '"') start++; // пропускаем пробелы и кавычки

const char\* end = start;

while (\*end != '"' && \*end != ',' && \*end != '}') end++;

result = std::string(start, end - start);

}

}

return result;

}

JNIEXPORT jboolean JNICALL

Java\_com\_example\_currencyconverterv2\_activities\_LoginActivity\_validateUserNative(

JNIEnv\* env,

jobject /\* thisObj \*/,

jstring usersJson,

jstring username,

jstring password) {

const char \*usersJsonStr = env->GetStringUTFChars(usersJson, 0);

const char \*usernameStr = env->GetStringUTFChars(username, 0);

const char \*passwordStr = env->GetStringUTFChars(password, 0);

// Ищем пользователя в JSON

std::string userKey = std::string("\"") + usernameStr + "\"";

const char\* userStart = strstr\_custom(usersJsonStr, userKey.c\_str());

jboolean result = JNI\_FALSE;

if (userStart) {

// Извлекаем пароль пользователя

std::string storedPassword = extract\_value(userStart, "password");

// Сравниваем пароли

if (storedPassword == passwordStr) {

result = JNI\_TRUE;

}

}

env->ReleaseStringUTFChars(usersJson, usersJsonStr);

env->ReleaseStringUTFChars(username, usernameStr);

env->ReleaseStringUTFChars(password, passwordStr);

return result;

}

}

Файл LoginActivity.kt

external fun validateUserNative(usersJson: String, username: String, password: String): Boolean

companion object {

init {

System.loadLibrary("user\_validation")

}

}

private external fun validateUserNative(usersJson: String, username: String, password: String): Boolean

companion object {

init {

System.loadLibrary("user\_validation")

}

}

private fun validateUser(username: String, password: String): Boolean {

return validateUserNative(usersJson.toString(), username, password)

}

**5 ВЫВОД**

В ходе выполнения лабораторной работы была успешно реализована интеграция нативного кода C++ в Android-приложение с использованием NDK. Основная функция проверки учетных данных пользователя была перенесена в нативную библиотеку, что потенциально повышает безопасность этой операции.

Использование NDK позволило расширить возможности приложения, добавив низкоуровневый доступ к операции валидации. Это демонстрирует гибкость платформы Android в плане интеграции различных языков программирования и технологий.

Реализация функции проверки учетных данных на нативном уровне может предоставить следующие преимущества:

1) Повышенная производительность при обработке больших объемов данных пользователей.

2) Усложнение анализа и потенциального взлома механизма аутентификации.

3) Возможность использования более сложных алгоритмов шифрования и хеширования, доступных в C++ библиотеках.