ИССЛЕДОВАНИЕ ВОЗМОЖНОСТЕЙ СИСТЕМЫ ДЛЯ АДМИНИСТРИРОВАНИЯ АВТОНОМНЫХ VR-ГАРНИТУР

Куртуков Д.А. [1], *Лоскутов В.В.* [2]
[1] *НИ ТПУ, г. Томск (студент гр. 8ИМ31 ОИТ, ИШИТР), e-mail: dak87@tpu.ru*
[2] *НИ ТПУ, г. Томск (старший преподаватель ОИТ, ИШИТР), e-mail: rewenger@tpu.ru*

Аннотация

В докладе рассматривается способ удаленного администрирования автономных гарнитур виртуальной реальности. Был проведен анализ существующих решений, поддерживающих устройства нескольких производителей. Проведены анализы подходящих сетевых протоколов для передачи данных и возможности прав Device Owner для контроля системы Android. Анализы сопровождались практическим тестированием на реальных устройствах.

Ключевые слова: виртуальная реальность (VR).

Введение

С каждым годом технологии виртуальной реальности становятся всё более распространёнными и востребованными в образовании. Актуальность их применения в этой сфере обусловлена множеством факторов. Прежде всего, виртуальная реальность создаёт уникальную возможность для создания интерактивных образовательных сред с глубоким погружением, которые могут значительно повысить интерес студентов к обучению [1]. Виртуальные учебные классы позволяют моделировать различные сценарии и ситуации, которые могут быть либо труднодоступными, либо опасными в реальной жизни. Например, обучение медицинским профессиям или техническим специальностям может быть существенно облегчено за счет симуляций, предлагаемых VR.

Рост использования виртуальной реальности в образовании также связывается с глобальными трендами цифровизации и дистанционного обучения. В условиях пандемии COVID-19 многие образовательные учреждения столкнулись с необходимостью перейти на удалённые форматы обучения, что привело к ускоренному внедрению технологий VR и смешанного обучения [2]. Это расширило доступ к качественным образовательным ресурсам и дало возможность студентам получать знания из любой точки мира.

Однако растущее количество пользователей виртуальных технологий ставит перед образовательными учреждениями новые вызовы [3]. Необходимость эффективного управления и контроля над VR-технологиями в учебном процессе становится критически важной. Системы автоматизированного контроля играют ключевую роль в обеспечении как безопасности, так и эффективности образовательного процесса. Без должного управления возможности VR могут быть использованы не по назначению или неэффективно, что может негативно повлиять на обучение и участие студентов [4].

Эффективное управление включает в себя мониторинг взаимодействия студентов с виртуальными ресурсами, оценку их прогресса и предоставление обратной связи. Это также связано с обеспечением соблюдения норм безопасности, защиты данных и поддержания академической честности. Система автоматизированного контроля может помочь преподавателям отслеживать достижения студентов в реальном времени, выявлять пробелы в знаниях и адаптировать учебные материалы под потребности конкретного класса или отдельного студента.

Обзор существующих решений

В таблице 1-2 приведено описание двух систем – ManageXR и ArborXR, которые работают на устройствах от разных производителей [5-6].

Таблица 1. Анализ разделов ManageXR

Раздел	Функция	Возможность реализации
Dashboard	Вывод информации по аккаунту	+
Analytics	Отслеживание времени использования гарнитур	+
	Отслеживание времени использования приложений	Можно лучше (получение информации по использованию приложений на конкретной гарнитуре)
	Отслеживание времени использования устрйств по конфигурации (парку)	+
	Создание собственных диаграмм	Web
	Выгрузка данных в csv формате	Web
Devices	Вывод информации по гарнитурам:	+
	Управление установленными приложениями	Нуждается в дополнительном исследовании
	Просмотр файловой системы	+
	Просмотр логов	+
	Запуск VR контента, загруженного с помощью ManageXR	Возможно реализовать запуск любого контента
	Перезагрузка гарнитуры	+
	Синхронизация данных	+
	On/Off режима Kiosk	+
	On/Off режима Tutorial	Легкий Kiosk
	Сброс до заводских настроек	+
	Удалить гарнитуру из системы	+
Configuration	Управление VR контентом, который доступен гарнитурам с данной конфигурацией	+
	Настройка параметров пользовательского окружения	Собственный пользовательский интерфейс - нуждается в дополнительном исследовании
	Управление режимом отладки по USB (режим разработчика)	-
	Управление возможностью обмена данных по USB	-
	Нстройка как и когда пользователь будет получать уведомление о предоставлении разрешений приложению	-
	Настройка приложения для обучающего режима	Легкий Kiosk
	Oculus: on/off режим сна	Возможно даже для Pico
	Pico: смена языка системы	Не работает +
	Pico: off возможность сброса настроек	+
	Pico: настройка режима производительности	Не работает -
	Pico: отслеживание перемещений	-
	Pico: управление временем перехода в режим ожидания/выключения экрана	+
	Pico: быстрая настройка границ	не работает -
	Pico: режим защиты глаз	не работает -
	Pico: режим защиты от рассеивания	-
	Задержка/отмена обновлений	+

Таблица 2. Дополнительные функции ArborXR

Раздел	Доп функционал ArborXR	Возможность реализации
Dashboard	Вывод данных по батареи: температура + состояние	+
Configuration	Возможность задать режим авторизации в системе при включении гарнитуры	+
	Возможность ввода заданного пин кода для выхода из Kiosk	+
	Изменение времени нажатия кнопки "Домой" на контроллере для его отключения	+
	Pico: Автоматическая настройка IPD (on/off)	-
Analytics	Есть статистика по отдельной гарнитуре	+

Выбор протокола передачи данных

Для контроля над VR-гарнитурами, работающих на мобильной ОС важно выбрать подходящий протокол обмена данными, который позволить отправлять запросы в обе стороны. WebSocket [7] легко интегрируется в Android с помощью библиотеки OkHttp. Соединение устанавливается один раз и поддерживается на протяжении всей сессии. Сообщения отправляются и принимаются в реальном времени, что делает этот протокол удобным для чатоподобных или событийных интерфейсов. Однако WebSocket не обеспечивает строгой структуры данных, требует ручной сериализации и десериализации, а также не всегда стабильно восстанавливает соединение при сетевых сбоях.

MQTT [8] используется через библиотеку Eclipse Paho. Клиент подключается к брокеру, подписывается на нужные топики и получает сообщения. Этот протокол отличается высокой устойчивостью и минимальным потреблением ресурсов, что особенно ценно для фоновых задач и нестабильных сетей. Тем не менее, MQTT ориентирован на простые сообщения и не предоставляет широких возможностей для формализации структур данных.

gRPC [9] – это современный высокопроизводительный RPC-протокол, использующий Protobuf для описания контрактов между клиентом и сервером. В Android gRPC интегрируется через автоматическую генерацию клиентского кода на Kotlin на основе .proto-файлов. Он поддерживает как обычные вызовы, так и потоковую передачу данных в обе стороны. Отлично сочетается с Kotlin Coroutines. gRPC обеспечивает строгость, контроль типов, масштабируемость, безопасность (встроенная поддержка TLS) и полную совместимость с микросервисной архитектурой. При росте требований к производительности и архитектурной строгости он становится оптимальным выбором.

AMQP [10], чаще всего представленный через RabbitMQ, используется в мобильной ОС через промежуточные REST или gRPC-сервисы, так как прямое подключение к AMQP с мобильного клиента не является рекомендуемой практикой. Такой подход добавляет надёжность и очередь доставки, но увеличивает архитектурную сложность и задержки.

XMPP [11] применяется только в специфических случаях, например, если в инфраструктуре уже присутствует XMPP-сервер. В мобильной ОС реализуется через библиотеки вроде Smack, позволяющие устанавливать соединение и обмениваться сообщениями. Протокол поддерживает множество функций – присутствие, статусы, подписки – но имеет высокий порог входа и избыточен для большинства мобильных задач.

По совокупности характеристик, gRPC демонстрирует наилучший баланс между строгостью, производительностью и масштабируемостью. Он особенно эффективен при разработке мобильных клиентов для микросервисной архитектуры, обеспечивая безопасность, четкость интерфейсов и удобство поддержки. WebSocket и MQTT остаются хорошими альтернативами для более простых и лёгких решений, но в контексте полноценных корпоративных систем именно gRPC обеспечивает уровень качества и контроля.

Права Device Owner в Android

Для создания полноценной системы администрирования VR-гарнитур, приложение должно быть назначено как Device Owner [12]. Это специальный режим, при котором приложение получает расширенные права на управление устройством. В отличие от обычных приложений или даже старого режима Device Admin, Device Owner предоставляет полный контроль над системой. Для его использования приложение должно иметь компонент класса DeviceAdminReceiver (листинг 1), быть объявлен в манифесте (листинг 2) и содержать набор политик (листинг 3).

Листинг 1. Компонент MyDeviceAdminReceiver

```
class MyDeviceAdminReceiver : DeviceAdminReceiver() {
    override fun onEnabled(context: Context, intent: Intent) {
        super.onEnabled(context, intent)
        Log.d("log", "Device Admin enabled")
    }
    override fun onDisabled(context: Context, intent: Intent) {
        super.onDisabled(context, intent)
        Log.d("log", "Device Admin disabled")
    }
}
```

Листинг 2. Регистрация в манифесте приложения

```
<receiver
    android:name=".MyDeviceAdminReceiver"
    android:permission="android.permission.BIND_DEVICE_ADMIN">
    <meta-data
        android:name="android.app.device_admin"
        android:resource="@xml/device_admin_policies" />
    <intent-filter>
        <action android:name="android.app.action.DEVICE_ADMIN_ENABLED" />
    </intent-filter>
</receiver>
```

Листинг 3. Набор политик

```
<device-admin xmlns:android="http://schemas.android.com/apk/res/android">
    <uses-policies>
        <force-lock />
        <wipe-data />
        <disable-camera />
        <limit-password />
        <set-global-proxy />
    </uses-policies>
</device-admin>
```

Присвоить права Device Owner можно ADB-командой с указанием пакета приложения и компонента DeviceAdminReceiver (adb shell dpm set-device-owner com.example.app/.DeviceAdminReceiver). Предварительно необходимо выйти на устройстве из всех авторизованных аккаунтов. В сравнении с традиционным подходом Device Admin, этот режим предлагает более богатый и безопасный функционал, требуемый для корпоративного использования Android-устройств. В режиме Device Owner доступны критически важные функции управления, которые недоступны обычным приложениям. Примеры продемонстрированы на листингах 4-8.

Лиситинг 4. Тихая установка APK

```
val dpm = getSystemService(DevicePolicyManager::class.java)
val admin = ComponentName(this, MyDeviceAdminReceiver::class.java)
val apkFile = File("/sdcard/Download/app.apk")

dpm.installExistingPackage(admin, "com.example.app")
```

Лиситинг 5. Режим Kiosk - блокировка устройства для одного или более приложений

```
val dpm = context.getSystemService(DEVICE_POLICY_SERVICE) as DevicePolicyManager
```

```
val adminName = ComponentName(context, DeviceAdminReceiver::class.java)
val APP_PACKAGES = kioskApps

val activityManager = context.getSystemService(Context.ACTIVITY_SERVICE) as
ActivityManager

dpm.setLockTaskPackages(adminName, APP_PACKAGES)
dpm.setLockTaskFeatures(adminName, DevicePolicyManager.LOCK_TASK_FEATURE_HOME)

val options = ActivityOptions.makeBasic()
options.setLockTaskEnabled(true)
val packageManager = context.packageManager
val launchIntent = packageManager.getLaunchIntentForPackage(startApp)
if (launchIntent != null) {
    context.startActivity(launchIntent, options.toBundle())
}
```

Листинг 6. Запрет установки приложений пользователем

```
val dpm = getSystemService(DevicePolicyManager::class.java)
val admin = ComponentName(this, MyDeviceAdminReceiver::class.java)
dpm.setUserRestriction(admin, UserManager.DISALLOW_INSTALL_UNKNOWN_SOURCES, true)
```

Листинг 7. Удаленный сброс устройства

```
val dpm = getSystemService(DevicePolicyManager::class.java)
dpm.wipeData(DevicePolicyManager.WIPE_EXTERNAL_STORAGE or
DevicePolicyManager.WIPE_RESET_PROTECTION_DATA)
```

Листинг 8. Блокировка системных функций

```
val dpm = getSystemService(DevicePolicyManager::class.java)
val admin = ComponentName(this, MyDeviceAdminReceiver::class.java)
dpm.addUserRestriction(admin, UserManager.DISALLOW_CAMERA)
dpm.addUserRestriction(admin, UserManager.DISALLOW_USB_FILE_TRANSFER)
```

Заключение

В ходе работы была рассмотрена возможность реализации удалённого администрирования автономных гарнитур виртуальной реальности на базе Android. Проведён анализ существующих решений для VR-гарнитур, таких как ManageXR и ArborXR, с выявлением их функциональных преимуществ и ограничений. Были сопоставлены различные сетевые протоколы передачи данных, из которых gRPC был выделен как наиболее надёжный и масштабируемый вариант для построения систем централизованного управления.

Особое внимание было уделено возможностям системы Android при назначении приложения в статус Device Owner. Продемонстрированы основные административные функции, доступные при таком подходе, включая тихую установку приложений, активацию режима киоска, удалённый сброс и блокировку пользовательских функций. Реализация указанных механизмов была проверена в ходе практических экспериментов на реальных VR-устройствах.

Полученные результаты подтверждают, что использование Device Owner в сочетании с современными протоколами связи позволяет реализовать надёжную, гибкую и безопасную систему удалённого администрирования VR-гарнитур. Это решение может быть эффективно использовано в образовательной и корпоративной среде для управления парком автономных устройств, обеспечивая контроль, безопасность и удобство эксплуатации.

Список ресурсы удаленного доступа

1. Высшая школа экономики. The Impact of Digitalization on the Socio-Economic Development of Russia // HSE Publications. – 2021. [Электронный ресурс]. – URL: publications.hse.ru/pubs/share/direct/464964807.pdf (дата обращения: 15.02.2025).

2. Лазутина Т. Влияние пандемии COVID-19 на процессы цифровизации в мире // CyberLeninka. – 2021. [Электронный ресурс]. – URL: cyberleninka.ru/article/n/vliyanie-pandemii-covid-19-na-protsessy-tsifrovizatsii-v-mire (дата обращения: 15.02.2025).

3. Левичев А. Прошлое и будущее VR-индустрии // DTF. – 2021. [Электронный ресурс]. – URL: dtf.ru/gameindustry/206605-proshloe-i-budushee-vr-industrii#parks (дата обращения: 15.02.2025).

4. Справочник по автоматизации. Системы автоматического контроля // Справочник. [Электронный ресурс]. – URL: spravochnick.ru/avtomatizaciya_tehnologicheskih_processov/sistemy_avtomaticheskogo_kon trolya (дата обращения: 15.02.2025).

5. ManageXR. Управление устройствами виртуальной реальности // ManageXR. [Электронный ресурс]. – URL: managexr.com/ (дата обращения: 15.02.2025).

6. ArborXR. Управление виртуальной реальностью для бизнеса // ArborXR. [Электронный ресурс]. – URL: arborxr.com (дата обращения: 15.02.2025).

7. WebSocket. Официальный сайт // WebSocket. [Электронный ресурс]. – URL: websocket.org (дата обращения: 15.02.2025).

8. MQTT. Официальный сайт // MQTT. [Электронный ресурс]. – URL: mqtt.org (дата обращения: 15.02.2025).

9. gRPC. Официальный сайт // gRPC. [Электронный ресурс]. – URL: grpc.io (дата обращения: 15.02.2025).

10. AMQP. Официальный сайт // AMQP. [Электронный ресурс]. – URL: amqp.org (дата обращения: 15.02.2025).

11. XMPP. Официальный сайт // XMPP. [Электронный ресурс]. – URL: xmpp.org (дата обращения: 15.02.2025).

12. Реализация приложения – device owner-а под Android // Habr. [Электронный ресурс]. – URL: habr.com/ru/articles/263977 (дата обращения: 15.02.2025).