**Отчёт по научно-исследовательской работе (НИР) по разработке онлайн-сервиса для облачного запуска мобильных приложений**

**1. Описание проблемы, которую должна решать система**

Современные мобильные устройства, особенно бюджетные или устаревшие модели, часто сталкиваются с рядом ограничений, затрудняющих эксплуатацию новейших мобильных приложений:

**- Недостаточная производительность:** Некоторые приложения требуют высокопроизводительной аппаратной платформы для комфортной работы. Клиентские устройства могут не обеспечивать должный уровень производительности.

**- Устаревшая версия ОС Android:** Многие приложения создаются под актуальные версии Android, и при попытке запуска их на более старых ОС возникают проблемы совместимости. Это особенно критично для пользователей, чьи устройства перестали получать обновления.

**- Ограничения по объёму памяти:** Нехватка свободного места на устройстве не позволяет устанавливать или использовать приложения с большим объёмом кэша или медиаконтента.

**- Регулярное обновление приложений:** Пользователям приходится самостоятельно отслеживать и устанавливать обновления приложений, что может быть неудобно.

**- Затраты для разработчиков на поддержку старых версий ОС:** Разработчикам приходится адаптировать приложения под разные версии Android, что увеличивает стоимость и сложность разработки.

Разрабатываемая система – онлайн-сервис облачного запуска приложений – решает эти проблемы. Пользователю достаточно иметь «тонкий клиент» на своём устройстве, а приложения работают на высокопроизводительном сервере с последней версией Android. Данные и кэш хранятся на стороне сервера, обновления выполняются автоматически, а разработчики могут ориентироваться только на актуальную версию Android.

**2. Детальное описание компонентов (органов) системы**

Система состоит из двух ключевых компонентов:

**- Магазин мобильных приложений (аналог AppStore или Google Play):**

**Назначение:** предоставляет пользователю возможность найти, просмотреть, выбрать и «установить» (по сути, активировать облачный запуск) нужные мобильные приложения.

**Принцип работы:**

- Пользователь с помощью клиентского приложения (мобильного или веб) обращается к магазину.

- Происходит аутентификация и авторизация по токенам (Access Token, привязанный к IP, и Refresh Token).

- Пользователь просматривает каталог приложений, фильтрует по категориям и версиям.

- При выборе приложения магазин взаимодействует с сервером баз данных (БД) через REST API для получения метаданных, инициации запуска и привязки приложения к сеансу пользователя.

- Почему именно так: Данный подход упрощает управление контентом, обновлениями и доступом, а также обеспечивает гибкую масштабируемость. Магазин является фронтендом, предоставляющим унифицированный интерфейс к огромному количеству приложений.

- Техническое основание: при разработке использован подход Database-First. Вся бизнес-логика и операции с данными сосредоточены в БД. Магазин выступает в роли интерфейсного слоя, получающего данные через REST API и отображающего их пользователю.

**- Гипервизор (серверная часть для запуска Android-приложений):**

**Назначение:** Отвечает за фактическое исполнение мобильных приложений в облаке. Он эмулирует среду Android последней версии.

**Принцип работы:**

- На сервере установлен гипервизор (либо Docker-контейнеры, либо эмулятор Android – у нас нет докера, мы вряд ли это прикрутим), позволяющий запускать приложения на высокопроизводительной аппаратной платформе.

- Приложения получают доступ к серверному дисковому пространству и работают под управлением актуальной версии Android.

- На клиентское устройство транслируется интерфейс приложения (например, через удалённый стриминг экрана или протокол обмена событиями пользовательского ввода), что превращает устройство пользователя в «тонкий клиент».

- Почему именно так: Такой подход устраняет зависимость от аппаратных характеристик конечного устройства, обеспечивает всегда актуальную ОС Android и упрощает обновления, а также снимает необходимость в множественных версиях приложения для разных ОС.

- Техническое основание: Высокопроизводительный сервер с LAMP-стеком и эмулятором Android. Использование гипервизора позволяет централизованно управлять обновлениями и настройками, оптимизируя ресурсы.

Архитектура серверной части:

- ОС и серверные компоненты:

- Linux Debian Bookworm 12.8.0 – стабильная, безопасная и широко поддерживаемая операционная система.

- Apache/2.4.62 – веб-сервер, работающий в режиме MPM Event для оптимизации под REST-запросы.

Параметры конфигурации:

- StartServers 3

- MinSpareThreads 9

- MaxSpareThreads 54

- ThreadLimit 28

- ThreadsPerChild 18

- MaxRequestWorkers 54

- MaxConnectionsPerChild 0 (неограниченно, для стабильной длительной работы)

- MySQL 8.4.3 – СУБД для централизованного хранения данных.

- PHP 8.2.26 + PHP-FPM – для высокопроизводительной обработки REST-запросов и генерации динамического JSON-контента.

- Архитектурные особенности реализации функционала магазина приложений:

1) База данных (Database-First подход):

Все сущности и связи в БД определены заранее. Вся бизнес-логика (идентификация пользователей, авторизация, проверка прав, операции с приложениями) реализована в хранимых процедурах. Это повышает безопасность (все запросы из PHP идут через EXECUTE) и структурирует код.

2) REST API интерфейс на PHP:

Собственный фреймворк, разрабатываемый под конкретные нужды системы, выполняет:

- Валидацию запросов.

- Маппинг эндпоинтов API на хранимые процедуры в БД. - Привязку входных параметров к параметрам процедур. - Преобразование результатов в JSON.

- Контроль доступа к локальному хранилищу (файлы, медиа).

- Возможность отправлять E-mail и вызывать внешние скрипты Bash.

3) Клиентское мобильное приложение:

Лёгкий клиент, работающий на устройстве пользователя. Он не содержит основной логики работы приложений, а лишь отображает интерфейс и передаёт ввод пользователя. Это снижает требования к устройству по ресурсам и обновлениям.

Реализован на React Native. Выбранная технология обусловлена наличием поддержки кросс-платформенности (Android и IOS). Вся верстка адаптивна и подойдет под любой размер экрана. Все взаимодействие с сервером организовано через REST-API, трансляция работающих приложений с сервера реализована через WEB-RTC протокол (веб-сокетное соединение с соответствующими нодами на сервере).

4) Ядро гипервизора:

Композитный сервис, состоящий из исходного кода операционной системы Android, небольших низкоуровневых скриптов на bash (аналог CRUD запросов на SQL, занимаются менеджментом конфигов, данных и прочих свойств эмулятора), высокоуровневых нод, написанных на Python (маршрутизатор соединений, приемник данных от пользователей) и Node.js (WEB-RTC нода для handshake между сервером и пользователем).

Исходный код Android: представляет собой модифицированный код AOSP (Android open-source project) с возможность создания avd (android virtual device), а также прочих надстроек, которые не позволял сделать AOSP

Скрипты на bash: используются для создания, удаления, редактирования и запуска avd. Также могут устанавливать приложения на эмулятор, выдавать права им и запускать.

Ноды на Python: реализиуют маршрутизацию с внешнего ip адреса, на адреса серверных нод.

Ноды на Node.js: используются для преодоления NAT заслона. Позволяют пользователю и серверу подключиться друг к другу и бесперебойно транслировать поток данных между ними. Для соединения используются STUN сервера Google.

Backend-клиент для эмуляторов: программа с интерфейсом на JAVA, установлена на каждом эмуляторе, устанавливает непосредственное соединение с клиентом.

Механизм работы подключения:

1. Пользователь инициирует запуск приложения запросом к фреймворку.
2. Фреймворк запускает нужный bash скрипт, который в свою очередь запускает эмулятор, устанавливает на него нужное приложение (если его нет у пользователя), запускает backend софт и подключается к ноде WEB-RTC.
3. Сервер отправляет request на клиент пользователя.
4. Клиент отправляет answer.
5. Устанавливается подключение.
6. С этого момента каждое нажатие и свай пользователя отправляются в виде массива координат на сервер, где в свою очередь происходит эмуляция нажатия в том же месте экрана эмулятора (для точности нажатий используется математический пересчет от размеров экрана устройства).
7. Пользователь пользуется приложением, пока не истечет тариф, либо пока он не изъявит желание прервать сессию нажатием на кнопку завершения.

3. Описание того, что было достигнуто В ходе НИР:

- Разработана архитектура облачного онлайн-сервиса, позволяющего запускать Android-приложения на сервере.

- Оптимизирована серверная конфигурация под режим REST API, что даёт высокую производительность обработки запросов и уменьшает задержки.- Реализован подход Database-First: вся бизнес-логика сосредоточена в БД, обеспечивая чёткую структуру и повышенную безопасность.

- Разработан собственный PHP-фреймворк для REST API, упрощающий интеграцию с БД, повышающий безопасность (через работу с хранимыми процедурами) и гибкость (выбор и настройка контента, файлохранилища, внешних вызовов).

- Встроена система авторизации и аутентификации на основе двух токенов, учитывающая особенности динамики IP-адресов.

Разработан адаптивный кроссплатформенный мобильный клиент.

Разработан масштабируемый гипервизор с потенциалом расширения и децентрализации.

4. Общее описание узких мест системы

Несмотря на достигнутый прогресс, в системе присутствуют потенциальные узкие места:

- Нагрузки на БД: Высокая зависимость от хранимых процедур может привести к нагруженности MySQL при интенсивных запросах, особенно при большом количестве пользователей.

- Масштабируемость при пиковой нагрузке: При значительном увеличении числа пользователей и одновременном запуске множества приложений может потребоваться горизонтальное масштабирование или балансировка нагрузки.

- Безопасность хранилища и файловой системы: Необходим жёсткий контроль над путями и типами контента, чтобы избежать несанкционированного доступа или внедрения вредоносных данных.

- Сетевая задержка и пропускная способность: Качество пользовательского опыта может зависеть от пропускной способности сети и задержек при трансляции экрана и взаимодействии.

- Обновление гипервизора и ОС Android: Поддержание актуальности серверной среды требует плановых обновлений и миграций, что может кратковременно снижать доступность сервиса.

- Скорость запуска гипервизора снижена из-за отсутствия специальной облегченной версии и альтернативной структуры ядра.

5. Детальное описание узких мест и предложения по их устранению

1) Нагрузки на БД:

- Проблема: Большое количество запросов к хранимым процедурам может привести к росту времени отклика.

- Решение:

- Внедрение кэширования данных на уровне REST API (например, с помощью Redis или Memcached).

- Репликация БД и разделение нагрузки между несколькими MySQL-серверами (Master-Slave или Master-Master репликации).

- Оптимизация хранимых процедур, индексирование таблиц, анализ запросов с помощью инструментов профилирования.

2) Масштабируемость при пиковой нагрузке:

- Проблема: При резком росте числа пользователей производительность может деградировать.

- Решение:

- Использование балансировщиков нагрузки (NGINX или HAProxy) перед Apache, распределяющих запросы по нескольким инстансам веб-сервера.

- Горизонтальное масштабирование путём развертывания дополнительных серверов с гипервизорами, объединённых в кластер.

- Контейнеризация (Docker/Kubernetes) для быстрой оркестрации и масштабирования.

3) Безопасность хранилища и файловой системы:

- Проблема: Опасность несанкционированного доступа к контенту, выполнение вредоносного кода, атаки через подмену путей.

- Решение:

- Жёсткое разграничение директорий, строгая валидация путей и имён файлов. - Использование SELinux/AppArmor для изоляции процессов.

- Постоянный аудит и обновление ACL (Access Control Lists) для файловой системы.

- Ещё более строгий контроль на уровне PHP-фреймворка, применение White-list подхода (разрешать только известные и проверенные типы контента).

4) Сетевая задержка и пропускная способность:

- Проблема: Длительная задержка в ответе сервера может ухудшить качество взаимодействия пользователя с приложением.

- Решение:

- Распределение CDN для статических ресурсов.

- Оптимизация протокола передачи (использование WebSockets или протокола с низкой задержкой для стриминга экрана).

- Сжатие и оптимизация трафика, адаптация к ширине канала пользователя.

5) Обновление гипервизора и ОС Android:

- Проблема: Обновления серверной части могут требовать краткосрочного простоя.

- Решение:

- Использование Blue-Green Deployment стратегий: параллельный запуск новой версии среды, перевод трафика на неё после тестирования, минимизация времени простоя.

- Автоматизированные скрипты обновления и отката, тестирование обновлений в изолированной среде перед разворачиванием в продакшн.

6) Скорость запуска гипервизора и его архитектура:

- Проблема: некомфортная скорость запуска

- Решение:

- Использование облегченных версий системы Android (cuttlefish для мобильных устройств).

- Модификация уже имеющихся образов под свои нужды.

- Децентрализация архитектуры, добавление распределительных узлов для равномерной нагрузки на все копии гипервизора, облегчение ПО. Улучшение алгоритмов.

Заключение:

Разработанная информационная система представляет собой комплексный онлайн-сервис по облачному запуску мобильных приложений, устраняющий барьеры производительности, совместимости и обновляемости, с которыми сталкиваются конечные пользователи и разработчики. Система основана на чёткой архитектуре, использует LAMP-стек с оптимизированным конфигурированием под REST API и применяет подход Database-First для повышения безопасности и управляемости.

Отдельно выделены потенциальные узкие места: нагрузка на БД, масштабируемость, безопасность файлового хранилища, сетевая задержка и обновления и скорость гипервизора. Для каждого проблемного аспекта представлены конкретные предложения по улучшению. Внедрение этих мер позволит повысить надёжность, отказоустойчивость и качество конечного пользовательского опыта, а также обеспечить гибкость и масштабируемость системы при дальнейшем росте количества пользователей и увеличении функциональности.