МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет	*Кафедра*
вычислительной техники	«Вычислительная техника»

Направление подготовки: 09.04.01 «Информатика и вычислительная техника»

Магистерская программа: «Прикладной искусственный интеллект»

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

на тему

МЕТОДИКА, АЛГОРИТМЫ И ПРОГРАММНЫЕ СРЕДСТВА РЕАЛИЗАЦИИ ПОЛЬЗОВАТЕЛЬСКИХ БИЗНЕС-ПРОЦЕССОВ С УЧЕТОМ ОПОВЕЩЕНИЙ ОТ СТОРОННИХ СЕРВИСОВ

Студент 25.05.26 Кузина Ангелина Евгеньевна
(подпись, дата) (ФИО полностью)

Руководитель 25.05.26 С.В. Шибанов
(подпись, дата) (фамилия, инициалы)

Нормоконтролёр 25.05.26 А.С. Бычков
(подпись, дата) (фамилия, инициалы)

Рецензент нач. УИТ _____ Севастьянов А.В.
(должность, место работы) (подпись, дата) (фамилия, инициалы)

Работа допущена к защите (протокол зас. кафедры от 22.05.2026 № 14)

Заведующий кафедрой _____ М.А. Митрохин
(подпись) (фамилия, инициалы)

Работа защищена с отметкой отлично (протокол заседания ГЭК от 04 июня 2026 № 3)

Секретарь ГЭК _____ Е.И. Гурин
(подпись) (фамилия, инициалы)

Пенза 2026

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет вычислительной техники

Кафедра «Вычислительная техника»

«Утверждаю»
Заведующий кафедрой
_____М.А. Митрохин
«_20_» __сентября__2024__г.

ЗАДАНИЕ
НА МАГИСТЕРСКУЮ ДИССЕРТАЦИЮ

1. Студент____Кузина Ангелина_Евгеньевна_____гр._24ВВИм1_факультета__ВТ_____

<Фамилия, имя, отчество полностью>

2.Тема работы___Методика, алгоритмы и программные средства реализации пользовательских бизнес-процессов с учетом оповещений от сторонних сервисов

Тема утверждена приказом ПГУ №_920/о_____от " 20 "_сентября__2024 г.

3. Руководитель магистерской диссертации __Шибанов Сергей Владимирович_____

4. Задание на работу (назначение разработки, исходные данные и т.п.)____ _____
Разработать архитектуру, алгоритмы и программные средства, автоматизирующие пользовательские бизнес-процессы реагирования на штатные и чрезвычайные события с использованием локальных языковых нейросетей для адаптивной генерации контента в зависимости от роли пользователя и сторонних сервисов доставки push-уведомлений. В рамках работы необходимо реализовать ролевую модель доступа, алгоритм серверной фильтрации данных и механизм формирования интерактивных экстренных уведомлений для фоновой работы мобильного устройства для целелвой аудитории университета.

5. Перечень подлежащих разработке вопросов_1. Провести обзор классификации оповещений и существующих приложений с оповещениями. 2. Проанализировать языковые модели данных и выбрать подходящую для разработки. 3. Разработать гибридную клиент-серверную архитектуру системы и алгоритмы маршрутизации информационных потоков с учетом интеграции стороннего push-сервиса. 4. Спроектировать алгоритмическое обеспечение модуля ИИ, включая применение методов квантования весов нейросети (до 4 бит) и алгоритмы промпт-инжиниринга. 5. Осуществить программную реализацию серверного REST API, базы данных и подсистемы локального вывода работы языковой

модели. 6. Разработать клиентское мобильное приложение с реализацией фоновых сервисов обработки скрытых пакетов данных и динамической генерацией бизнес-процессов реагирования. 7. Провести многоуровневое экспериментальное тестирование разработанного комплекса, включая оценку потребления аппаратных ресурсов видеокарты при генерации текста и верификацию информационной безопасности системы.

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Изучение предметной области, анализ литературы.	10%	20.09.2024-21.11.2024	
2	Выбор технологического стека и методов оптимизации ИИ (квантование).	5%	22.11.2024 - 22.01.2025	
3	Разработка гибридной архитектуры системы, проектирование схемы базы данных, ролевой модели и алгоритмов формирования Data-пакетов.	10%	23.01.2025 - 23.04.2025	
4	Разработка серверной части. Программная реализация REST API, интеграция локальной LLM через библиотеку llama.cpp, настройка логики промпт-инжиниринга.	25%	24.04.2025 - 29.08.2025	
5	Разработка клиентской части. Создание мобильного приложения Android, интеграция с Firebase Cloud Messaging, программирование фоновых служб для обработки интерактивных push-уведомлений.	25%	01.09.2025 - 29.01.2026	
6	Экспериментальное исследование и тестирование. Нагрузочное тестирование ИИ, проверка безопасности ролевой модели, отладка системы.	15%	30.01.2026 - 30.03.2026	
7	Оформление выпускной работы	10%	31.03.2026 - 21.05.2026	

Дата выдачи задания "_20_" ____сентября___2024 г.

Руководитель магистерской диссертации _____
(подпись, дата)

Задание к исполнению принял студент _____
(подпись, дата)

Работу к защите допустить
Декан факультета _____ А.Р. Фионова
(подпись, дата)
29.05.2026

РЕФЕРАТ

Магистерская работа содержит 80 листов, 9 рисунков, 8 таблиц, 20 источников, 3 приложения.

ОПОВЕЩЕНИЯ, PUSH-УВЕДОМЛЕНИЯ, ИНТЕГРАЦИЯ, СЕРВИС, КЛАССИФИКАЦИЯ ОПОВЕЩЕНИЙ, ПУШ СЕРВЕР, СТРУКТУРА УВЕДОМЛЕНИЙ, БАЗА ДАННЫХ, REST API, НЕЙРОННАЯ СЕТЬ, LARGE LANGUAGE MODEL, ПРОМТ, КОНТЕКСТ ПРОМТА, ГЕНЕРАЦИЯ ТЕКСТА, ТОКЕН, ЯЗЫКОВАЯ МОДЕЛЬ, ЛОКАЛЬНЫЙ ИНТЕРФЕЙС, РОЛЕВАЯ МОДЕЛЬ, ФИЛЬТРАЦИЯ ДАННЫХ, НАГРУЗОЧНОЕ ТЕСТИРОВАНИЕ, ПРОМПТ-ИНЖЕНЕРИНГ, ИНФОРМАЦИОННАЯ БЕЗОПАСНОСТЬ

Цель работы: разработать архитектуру, алгоритмы и программные средства, автоматизирующие пользовательские бизнес-процессы реагирования на штатные и чрезвычайные события с использованием локальных языковых нейросетей для адаптивной генерации контента в зависимости от роли пользователя и сторонних сервисов доставки пуш-уведомлений

Объект исследования: процесс автоматизированного информирования и реализации пользовательских бизнес-процессов в образовательной среде.

Предмет исследования: методики, алгоритмы и программные средства адаптивной генерации контента на базе квантованных локальных больших языковых моделей, а также архитектурные паттерны безопасной фоновой доставки мобильных пуш-уведомлений от сторонних сервисов.

Технологии разработки: Saiga Llama-3 8B, Android Studio, Visual Studio Code, Python, Kotlin, SQLite 3, Firebase Cloud Messaging от Google.

Результаты работы: спроектированное и разработанное приложение «Генерация уведомлений» для образовательной среды университета.

СОДЕРЖАНИЕ

ТЕРМИНЫ И ОПРЕДЕЛЕНИЯ 7

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ 9

ВВЕДЕНИЕ 10

1 Анализ предметной области и постановка задачи на разработку приложения «Генерация уведомлений» 13

1.1 Классификация оповещений и их применение в системах 13

1.2 Положительные и отрицательные аспекты типов оповещений 15

1.3 Обзор существующих приложений с оповещениями 16

1.4 Бизнес-процессы и их корректировка 19

1.5 Описание направления собственного исследования 22

2 Разработка методики и алгоритмов интеллектуальной системы информирования 26

2.1 Проектирование гибридной клиент-серверной архитектуры с интеграцией сторонних push-сервисов 26

2.2 Сравнительный анализ больших языковых моделей 28

2.3 Обоснование языковой модели в разрабатываемом приложении 31

2.4 Математическое и алгоритмическое обеспечение модуля ИИ (методы квантования и промпт-инжиниринга) 31

2.5 Алгоритм маршрутизации бизнес-процессов на основе ролевой модели доступа 33

2.6 Алгоритм выявления триггерных ситуаций и формирования интерактивных скрытых пакетов 34

3 Программная реализация информационного комплекса 37

3.1 Обоснование выбора технологического стека разработки 37

3.2 Интеграция локальной языковой модели 39

3.3 Реализация алгоритмов формирования контекстных промптов 40

3.4 Проектирование REST API и реализация серверной фильтрации 41

3.5 Интеграция транспортного шлюза и формирование пакетов Data Payload 42
3.6 Разработка мобильного клиента и реализация бизнес-логики реагирования 44
4 Экспериментальное исследование и контроль качества программного обеспечения 46
4.1 Мероприятия по контролю качества программного обеспечения 46
4.2 Осуществление производительности и качества работы приложения 47
4.2.1 Проведение нагрузочного тестирования приложения 47
4.2.2 Семантическое тестирование 48
4.3 Интеграционное тестирование транспортного слоя и API 53
4.3.1 Тестирование фоновой доставки……..53
4.3.2 Тестирование интерактивных экстренных сценариев 55
4.4 Тестирование безопасности и ролевой модели доступа 56
ЗАКЛЮЧЕНИЕ 58
СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 61
Приложение А (обязательное) Код разработки 63
Приложение Б (обязательное) Скриншоты приложения 71
Приложение В (обязательное) Презентация 76

ТЕРМИНЫ И ОПРЕДЕЛЕНИЯ

В данной работе используются следующие термины с соответствующими определениями

Инференс – процесс непрерывной работы обученной нейронной сети, при котором она анализирует входящие данные и генерирует на их основе предсказания или текстовый ответ

Искусственный интеллект – комплекс технологических решений, позволяющий имитировать когнитивные функции человека и получать при выполнении конкретных практически значимых задач обработки данных результаты, сопоставимые, как минимум, с результатами интеллектуальной деятельности человека

Квантование – метод математической оптимизации нейронных сетей, заключающийся в снижении разрядности весов модели с целью сокращения потребляемой оперативной памяти. Например, с 16-битных чисел с плавающей запятой до 4-битных целых чисел

Облачный шлюз – система, обеспечивающая взаимодействие между локальными компонентами и облачными сервисами для безопасности данных и экономии трафика

Промпт-инжиниринг – метод разработки, структурирования и оптимизации текстовых инструкций для точного управления поведением и качеством ответов генеративных языковых моделей

Токен – базовая структурная единица текста (символ, часть слова или целое слово), используемая алгоритмами нейросети для лингвистической обработки и генерации естественного языка

API (Application Programming Interface) – программный интерфейс приложения, описывающий способы взаимодействия одной компьютерной программы с другими

Data Payload (Скрытый пакет данных) – полезная нагрузка в формате JSON, передаваемая сервером на мобильное устройство без использования

стандартных ключей визуализации, предназначенная для фоновой обработки кодом клиентского приложения

FCM (Firebase Cloud Messaging) – кроссплатформенный облачный сервис от компании Google для надежной доставки сообщений и push-уведомлений на клиентские устройства

GPU (Graphics Processing Unit) – графический процессор, используемый для параллельных вычислений и аппаратного ускорения нейронных сетей

JSON (JavaScript Object Notation) – текстовый формат обмена данными, основанный на парадигме «ключ-значение»

LLM (Large Language Model) – большая языковая модель, нейронная сеть на базе архитектуры Transformer, обученная на гигантских массивах текста и способная генерировать тексты на естественном языке

RBAC (Role-Based Access Control) – управление доступом на основе ролей, подход к обеспечению информационной безопасности, при котором права доступа субъекта к объектам системы определяются его ролью

REST (Representational State Transfer) – архитектурный стиль взаимодействия компонентов распределенного приложения, использующий стандартные методы протокола HTTP

Server-Side Filtering (Серверная фильтрация) – парадигма информационной безопасности, при которой ограничение и выборка данных осуществляются на стороне сервера (на уровне базы данных), минимизируя передачу избыточной или конфиденциальной информации на клиентское устройство

Topic (Топик / Тематический канал) – механизм в системе FCM, позволяющий осуществлять широковещательную рассылку сообщений только тем устройствам, которые подписаны на соответствующий канал

VRAM (Video Random Access Memory) – оперативная память графического процессора, используемая для хранения весов и контекста нейронной сети во время её работы

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ

В данной работе используются следующие обозначения и сокращения

БД – база данных

ВКР – выпускная квалификационная работа

ИИ – искусственный интеллект

МЧС – Министерство по чрезвычайным ситуациям

ПО – программное обеспечение

ОС – операционная система

СУБД – система управления базами данных

ЧС – чрезвычайная ситуация

API – Application Programming Interface

FCM – Firebase Cloud Messaging

GGUF – GPT-Generated Unified Format

GPU – Graphics Processing Unit

JSON – JavaScript Object Notation

LLM – Large Language Model

TCP – Transmission Control Protocol

REST – Representational State Transfer

VRAM – Video Random Access Memory

ВВЕДЕНИЕ

В современной жизни стремительно развиваются цифровые технологии, растет объем данных, что требует постоянного доступа к разнообразным источникам информации, которые помогают эффективно управлять временем, ресурсами и даже предотвращать опасные ситуации с человеком. Большинство успешных бизнесов создают свои приложения, сайты, личные кабинеты и используют их в качестве полезного инструмента для достижения своих целей. Эти инструменты ежедневно используют миллионы людей по всему миру. Они помогают решать такие задачи, как повтор заказа, получение уведомлений, пользование личным кабинетом, запись на услуги, отслеживание статуса заказа или работа сотрудников внутри компании [1].

У всех приложений есть один общий компонент – доставка уведомлений пользователю. Пользовательские сценарии и множество бизнес-процессов требуют огромных затрат по времени и ресурсам на реализацию эффективного обмена уведомлениями. Создание надежного и масштабируемого сервиса уведомлений и интеграция его в мобильное приложение обеспечит своевременное информирование пользователей и систем, что критично для бизнес-процессов и оперативного реагирования.

Данная работа посвящена исследованию системы для корректировки бизнес-процессов нейронной сетью с учетом оповещений от стороннего сервиса. Целью исследования является разработка концепции и реализации сервиса, позволяющего пользователям получать персонализированные и корректирующие действия пользователя уведомления в предметной области образования. Такой подход обеспечит значительное улучшение пользовательского опыта за счет предоставления информации в нужное время и в нужном месте.

Эффективная система оповещений не только повышает уровень взаимодействия между различными компонентами системы, но и способствует снижению времени реакции на изменения и возникающие проблемы. В рамках

работы будет рассмотрено использование облачных технологий при реализации приложения.

Особую актуальность приобретает создание интеллектуальных систем, способных генерировать динамичные текстовые оповещения на основании актуального контекста и корректировать действия разных групп пользователей через мобильные приложения.

На данный момент известно три способа генерации текста. Первый – поисковый, когда из заранее определенного текста нужно выбрать необходимый. Второй – шаблонный способ. При таком способе используется образец, куда добавляют слова с нужными характеристиками. Но в нем есть недостаток в необходимости создания большого количества трафаретов в зависимости от предметной области. И третий способ, используемый в данной научно-исследовательской работе, – генерация текст с помощью нейронных сетей [2]. Этот способ гарантирует хорошее качество ответов и пользуется популярностью в современном мире.

В процессе исследования предусматривается проектирование архитектуры взаимодействия с API нейросети, разработка алгоритмов формирования контекстных промптов и базы данных для хранения информации, создание эффективной структуры данных для передачи уведомлений, а также реализация механизмов регистрации пользователей и обработки входящих сообщений на мобильной платформе. Подобный метод позволяет повысить качество и скорость обмена информации в разных областях использования.

Для достижения поставленной цели необходимо решить следующие задачи:

– рассмотреть классификации уведомлений и действующих приложений с таким функционалом,

– составить основные рабочие процессы разрабатываемого приложения,

– произвести анализ данных о языковых моделях и определить подходящую для работы в приложении,

– разработать архитектуру системы, которая обеспечит бесперебойную и защищенную передачу информации между различными модулями,

– спроектировать алгоритмическое обеспечение модуля искусственного интеллекта, включая использование методов квантования весов языковой модели,

– реализовать программный код серверной части, базы данных и подсистемы локального вывода работы нейросети,

– разработать мобильное приложение с использованием фоновых процессов обработки скрытых пакетов данных и генерацией бизнес-процессов реагирования в реальном времени,

– произвести оценку эффективности предложенного решения с применением пользовательского и нагрузочного тестирования.

Разработанное программное обеспечение будет возможно интегрировать в мобильное приложение Пензенского государственного университета «ПГУ», что даст возможность увеличить функционал возможностей приложения. Дополнительно рассматривается возможность внедрения в аналогичные приложения с схожими областями использования. Особенно актуально будет для технологических предприятий, где важную роль играет соблюдение безопасности и своевременная корректировка действий различных групп сотрудников из-за происходящих внешних воздействий и их условий.

Во время работы над ВКР были осуществлены публикации статей на темы «Технические решения и технологии для отправки и получения оповещений» и «Гибкость бизнес-процессов: управление изменениями на основе оповещений и реакции пользователя».

1 Анализ предметной области и постановка задачи на разработку приложения «Генерация уведомлений»

1.1 Классификация оповещений и их применение в системах

При создании подавляющего большинства приложений, а особенно мобильного, предусматривается внедрение системы уведомлений, которая осуществляет информирование пользователя о каких-либо событиях и действиях. Правильное использование выбранных типов уведомлений позволяет разработчикам создавать эффективную работу приложения и улучшает качество взаимосвязи с конечным пользователем.

Оповещения могут решать различные задачи. В зависимости от задач можно классифицировать следующие оповещения [3]:

– оповещения, предоставляемые пользователю информацию от приложения о его состоянии или данных (примером может стать оповещение об успешной авторизации),

– оповещения, сообщающие о необходимости принять меры (например, уведомление об обновлении системы),

– оповещения, появляющиеся при ошибках или исключений (такие оповещения могут возникать при неверном вводе данных),

– оповещения, подтверждающие действие пользователя (чаще всего такой тип уведомлений используют при подтверждении удаления или скачивании),

– оповещения, которые информируют о событиях на уровне системы (например, завершение фоновых задач),

– оповещения, сообщающие о статусе текущей задачи или процесса (примером может стать оповещение, возникающее при загрузке или скачивании),

– оповещения, срабатывающие автоматически при наступлении необходимого условия, такие оповещения можно называть триггерными (например, почтовая рассылка).

Также оповещения можно классифицировать по типу их разработки. В таблице 1 представлены различные типы и описания оповещений.

Таблица 1 - Типы оповещений в программных обеспечениях

Название	Описание	Пример
Системные уведомления	Уведомления операционной системы о событиях	Фоновые задачи, обновления
Уведомления в приложениях	Оповещения об активности внутри приложения	Уведомление о новых сообщениях или появившихся событиях
Email-уведомления	Оповещения, отправляемые на электронную почту	Сброс пароля или подтверждение регистрации
Пуш-уведомления	Уведомления, отправляемые на мобильные устройства или браузеры, даже когда приложение не активно	Уведомление о начавшемся звонке из мессенджера
SMS-уведомление	Текстовые сообщения, отправляемые на мобильные телефоны от мобильных операторов, МЧС и контактов	Уведомление о климатических опасностях
Логирование	Запись сообщений в лог для отслеживания состояния приложения и выявления ошибок	При появлении багов во время звонка
Оповещения через API	Коммуникация между сервисами, где один сервис может уведомлять другой о событиях через вызовы запросов API	Уведомление об успешном получении данных
Чат-боты	Оповещения в мессенджерах с помощью ботов	Сообщения о различных событиях
Визуальные уведомления	Используются в графических интерфейсах для взаимодействия с пользователем	Всплывающие окна или алерты

На этапе составления требований и конечных целей реализуемого приложения, можно определить список необходимых типов оповещений. Как

правило, в одном приложении может использоваться сразу несколько типов уведомлений.

1.2 Положительные и отрицательные аспекты типов оповещений

Для проведения исследования аспектов различных каналов коммуникации было определено четыре наиболее часто используемых типов уведомлений. К ним относятся самые распространенные средства, такие как email и SMS-оповещения, которые обеспечивают доставку как в персональных, так и в корпоративных сценариях, а также пуш-уведомления (мобильные и веб-версии), которые осуществляют информирование пользователей через устройства и веб-браузеры без обязательной установки приложений. Преимущества и недостатки каждого типа представлены в таблице 2.

Таблица 2 – Преимущества и недостатки различных типов оповещений

Тип	Преимущества	Недостатки
Электронная почта	1. Поддерживается большинством устройств и ОС 2. Возможно отправлять массовые рассылки большому количеству получателей одновременно 3. Все сообщения собираются в одном хранилище, что упрощает просмотр истории переписки	1. Высокая вероятность попадания в спам-фильтры 2. Ограниченные возможности для интерактивного взаимодействия
Пуш-уведомления	1. Уведомление поступает мгновенно после отправки 2. Возможно настроить переход в приложение или выполнение другого действия 3. Присутствует возможность разделять клиентов и отправлять персональные оповещения	1. Влияние скорости интернет-соединения 2. Необходимость установки приложения на устройство пользователя
SMS-уведомления	1. Поступают на все мобильные устройства даже без наличия интернета 2. Не требуют наличия дополнительных приложений 3. Подходят для массовой рассылки	1. Имеют ограничения по количеству символов 2. Высокая стоимость при рассылках

Продолжение таблицы 2

1	2	3
Веб- и пуш-уведомления	1. Работают на всех устройствах с актуальным браузером 2. Могут содержать изображения и другие элементы для взаимодействия 3. Некоторые браузеры поддерживают отправку оповещений при отсутствии соединения с интернетом	1. Пользователь должен согласиться на получение уведомлений от нужного ресурса

Оповещения являются неотъемлемой частью использования программного обеспечения. Они выполняют роль оперативного информирования пользователей, что дает возможность компаниям повышать качество коммуникации клиентов приложения. Также они сокращают время для осуществления мер в чрезвычайных ситуациях и актуализируют информацию о протекании того или иного бизнес-процесса. Использование функционала с уведомлениями сводит к минимуму количество ошибок и повышает уровень как клиентского, так и внутреннего обслуживания.

Выбор технического стека разработки для отправки и получения уведомлений имеет четкую зависимость от требований и целей создаваемого проекта. Необходимо обращать внимание на такие факторы, как целевая аудитория, количество и временные промежутки отправляемых уведомлений, а также на материальное обеспечение инфраструктуры. Комбинация различных методов и технологических исполнений даст возможность реализовать наиболее эффективное и качественное решение для целей конечного пользователя.

1.3 Обзор существующих приложений с оповещениями

Было произведен поиск реальных примеров отечественных и зарубежных приложений, которые осуществляют информирование пользователей с использованием уведомлений, а также их основные положительные и отрицательные стороны.

Приложения для прогнозирования прогнозов погоды: Яндекс.Погода или Weather Channel присылают пользователям уведомления о сильных порывах ветра, аномальной жары или холода, штормовых предупреждениях и многое другое [4].

Google News, Yandex Новости – это все приложения новостей. Они помогают держать пользователей в курсе всех важных событий в мире, политике, спорте и других сферах.

Здоровье и фитнес-приложения такие приложения как MyFitnessPal, Strava или Health от Apple. Они предоставляют функции подсчета шагов, калорий, времени тренировок и другие полезные напоминания [5].

Приложения Яндекс.Транспорт или Citymapper предоставляют информацию о расписании общественного транспорта, задержках рейсов, пробках и так далее. Что помогает пользователям в ориентации по городу.

Финансовые операции в банковских приложения Сбербанк Онлайн и Тинькофф могут информировать о поступлениях средств, списаниях, задолженностях и предложениях по кредитным продуктам.

Уведомления от социальных сетей ВКонтакте, MAX, Facebook позволяют пользователям оставаться в курсе новых сообщений, комментариев, лайков и другой активности.

Приложения для безопасности антивирусные программы Kaspersky, Avast или системы мониторинга домашней безопасности. Преимущества и недостатки каждого из рассматриваемых типов приложений представлены в таблице 3.

Таблица 3 – Преимущества и недостатки приложений с уведомлениями

Примеры приложений	Преимущества	Недостатки
Яндекс.Погода, Weather Channel	Помогают планировать день исходя из текущих погодных условий. Уведомления об опасных погодных условиях могут спасти жизнь или предотвратить неудобства	Иногда могут быть не совсем точными прогнозы Чрезмерное количество уведомлений

Продолжение таблицы 3

1	2	3
Google News, Yandex Новости	Всегда быть в курсе последних событий Возможность выбрать категории новостей, которые интересны именно вам Удобство получения информации без необходимости постоянно проверять источники	Могут отвлекать от работы или отдыха Некоторые приложения перегружены рекламой или малоинформативны
MyFitnessPal, Strava, Health от Apple	Мотивируют к ведению здорового образа жизни Напоминание о необходимости физической активности дня	Постоянные уведомления могут вызывать стресс у некоторых людей Не всегда точные данные
Яндекс. Транспорт, Citymapper	Экономия времени благодаря актуальной информации о транспорте Предупреждения о возможных проблемах на маршруте помогают избежать опозданий Легко найти альтернативный маршрут в случае проблем	В некоторых регионах информация может обновляться с задержкой Зависимость от интернета для корректной работы
Сбербанк Онлайн, Тинькофф	Контроль над финансами в реальном времени Быстрое уведомление о подозрительных транзакциях. Простота управления счетами через приложение [6]	Может возникать путаница с частыми уведомлениями о мелких операциях Необходимость постоянного подключения к интернету для актуальных данных
ВКонтакте, MAX, Facebook	Быстрая связь с друзьями и подписчиками Актуальная информация о том, что происходит в сообществе Возможность мгновенно реагировать на важные события	Перегрузка информацией может привести к стрессу и отвлечению Частые уведомления могут нарушать концентрацию на работе или учебе

Продолжение таблицы 3

1	2	3
Kaspersky, Avast	Повышение уровня защиты устройства или дома Своевременная реакция на угрозы Обеспечение спокойствия пользователя	Ложные срабатывания могут вызывать беспокойство Некоторые приложения требуют регулярного обновления, что может замедлять работу устройства–

Уведомления – это мощный инструмент, который помогает людям быть в курсе происходящего вокруг них. Однако важно заметить, что чрезмерное использование уведомлений может негативно сказаться на концентрации и общем уровне стресса [7]. Поэтому в своем приложении необходимо выбрать только те типы уведомлений, которые действительно полезны и важны пользователю для его повседневной жизни.

1.4 Бизнес-процессы и их корректировка

Под бизнес-процессом понимается строго структурированная, логически связанная последовательность действий или событий, в которой задействованы один или несколько субъектов для достижения конечных результатов, совпадающих с ранее поставленными целями, которые представляют ценность для организации или отдельного пользователя [8].

У любого бизнес-процесса есть список компонентов, которыми он описывается:

– событие-триггер: внешнее или внутреннее условие, инициирующее запуск процесса,

– роли: пользователи или программные агенты, выполняющие действия в рамках процесса в соответствии со своими правами доступа,

– поток управления: алгоритм, определяющий порядок перехода от одного действия к другому в зависимости от условий.

В любом бизнесе или действии важно учитывать все процессы, участвующие в сценарии. Правильно выстроенная последовательность

бизнес-процессов помогает сокращать время выполнения, эффективно управляет ресурсами и оптимизировать цепочку действий. Управлять и корректировать бизнес-процессы можно как вручную, так и с использование современных технологий. В данной работе рассмотрен вариант с применением технологий искусственного интеллекта. ИИ в этом случае не просто получает последовательно функции, а самостоятельно анализирует контекст триггера, принимает промежуточные решения и корректирует поток управления для конечного пользователя, предлагая ему готовые сценарии реагирования.

Вовремя предложенный сценарий реагирования поможет пользователю спасти жизнь не только себе, но и окружающим. Но описать и учесть тысячи разных сценарий невозможно вручную за короткий срок, поэтому можно использовать обученную нейронную сеть для корректировки действий пользователя на различные события.

Предметной областью данного исследования является среда высшего учебного заведения. В ранней исследовательской работе в бакалавриате эта тема рассматривалась в контексте разработки приложения для помощи в навигации внутри корпусов и на территории кампуса, поэтому далее будут рассмотрены бизнес-процессы, касающиеся выбранной предметной области.

Образовательное учреждение характеризуется высокой плотностью коммуникаций и наличием строго регламентированных ролей групп: администрация, профессорско-преподавательский состав, студенты. Пользовательские бизнес-процессы в данной среде можно разделить на две категории:

– процессы, связанные с обеспечением учебной и бытовой деятельности,

– экстренные или критические процессы, которые связаны с обеспечением безопасности жизнедеятельности и требуют немедленного реагирования.

К первой категории можно отнести перенос аудиторий, сильная жара или порывы ветра, организация научных конференций, семинаров, спортивных

мероприятий, плановые отключения коммуникаций в кампусе, изменение расписания, сроков подачи документов или добавление новых правил в устав положения университета.

Ко второй категории относятся пожарная тревога, эвакуация, угроза террористического акта. В таких событиях как никогда важна скорость реакции и быстрый доступ к получению инструкции на спасение.

У двух категорий есть одна общая черта, различные группы пользователей будут иметь разный алгоритм действия при наступлении одного и того же события-триггера. Например, при наступлении события «Отключение электроэнергии» бизнес-процесс студента должен быть содержать информацию о самостоятельном изучении материала или переход в дистанционный режим, в то время как бизнес-процесс преподавателя должен включать организационные шаги по изменению формата лекции или лабораторных. Или при получении прогноза погоды с дождем, не зависящего от деятельности человека, преподаватели физкультуры должны организовать проведение занятия в спортивном зале, а студенты должны подготовиться к занятию не на улице. При чрезвычайных ситуациях преподаватели организуют безопасную для студентов эвакуацию, а студенты не паникуют и следуют указаниям.

В системах информирования посредством email, SMS, мессенджеры оповещение отображается у всех одинаковое. Ответственность за выбор правильного алгоритма действий полностью на пользователе. Это приводит к увеличению времени реакции и риску возникновения ошибок из-за человеческого фактора. Оповещения в основном отправляются в текстовом виде, поэтому корректировка действий пользователя посредством модуля искусственного интеллекта может осуществляться двумя методами:

– семантическая корректировка,

– интерфейсная корректировка.

Вместо ручного формирования ветвлений для каждой аудитории, задача маршрутизации делегируется модулю искусственного интеллекта. Получив

короткое события от администратора, языковая модель анализирует его влияние на определённую роль пользователей.

На уровне мобильного клиента используется интеграция сторонних сервисов для изменения механики реагирования. Если алгоритм, разработанный на стороне сервера, определяет событие как экстренное, мобильное приложение корректирует поведенческий паттерн пользователя путем добавления интерактивной кнопки с набором номера службы спасения.

Таким образом, время выполнения критического бизнес-процесса сокращается с многошагового алгоритма до одного действия, например, нажатие кнопки в шторке уведомлений. Это доказывает высокую эффективность применения генеративных моделей и интерактивных push-технологий для оптимизации и защиты бизнес-процессов в образовательной среде.

1.5 Описание направления собственного исследования

Основное направление исследования сосредоточено на создании интеллектуальной клиент-серверной системы и мобильного Android-приложения для оперативного информирования целевых аудиторий в сфере университетской жизни. Целью проекта является разработка архитектуры и реализация функционала, который позволит пользователям гарантированно получать персонализированные и корректирующий действия пользователя оповещения с использованием локальных больших языковых моделей и технологий фоновой доставки сообщений.

Основные функции создаваемого приложения следующие:

– интеллектуальная адаптация контента – автоматизированная генерация текста уведомлений на стороне сервера с помощью локальной нейросети Llama-3, позволяющая трансформировать исходное событие в стилистически корректный текст для конкретной целевой аудитории,

– гарантированная фоновая доставка – прием пуш-уведомлений мобильным клиентом даже в свернутом режиме с использованием шлюза

Firebase Cloud Messaging, обходя системные ограничения операционной системы Android,

– интерактивное реагирование на чрезвычайные ситуации – автоматическое распознавание чрезвычайных ситуаций: пожар, эвакуация, угрозы безопасности, – с динамическим добавлением экстренных элементов управления непосредственно в графический интерфейс уведомления, в данном случае кнопки вызова службы 112,

– ролевое управление доступом – реализация системы авторизации и строгой фильтрации данных на стороне сервера, исключающая неразрешенный доступ пользователей к информации, предназначенной для других ролевых групп,

– персистенция и аудит данных – автоматическое сохранение истории сгенерированных событий в защищенную реляционную базу данных для обеспечения контроля за действиями администраторов.

В окне авторизации должна быть возможность ввода двух полей: логина и пароля. В случае неверного ввода логина или пароля после нажатия кнопки входа «Войти в системы» должна отображаться ошибка ввода с текстом: «Неверный логин или пароль».

После успешной авторизации под ролью администратор должно отображаться окно генерации уведомлений со следующими элементами:

– заголовок «Генератор уведомлений»,

– поле ввода события для генерации текста уведомления с допустимым количеством символов 160,

– заголовок «Выберите получателей»,

– поле с выбором роли пользователей для доставки уведомления: студенты, преподаватели, руководство и другой тип персонала,

– кнопка генерации текста «Сгенерировать текст (AI)»,

– заголовок «Здесь появится результат»,

– поле для отображения результата без ограничения по символам,

– кнопка для перехода к истории уведомлений «Посмотреть историю».

Оповещения должны приходить как при открытом, так и при свернутом приложении.

Оповещение для пользователя должно отображаться из следующих элементов:

– иконка приложения,

– наименование приложения,

– заголовок «Оповещение ВУЗа»,

– текст сгенерированного уведомления,

– в случае уведомлений о чрезвычайных ситуациях кнопка для звонка в службу спасения.

В случае успешной авторизации других ролей, кроме администратора, должно отображаться окно истории тех уведомлений, которые были отправлены пользователям с авторизированной ролью.

Каждое уведомление должно отображаться из следующих элементов:

– дата в формате: ГГГГ-ММ-ЧЧ ЧЧ:ММ:СС.НННННН,

– текст с ролью пользователя: «Для кого: (РОЛЬ)»,

– текст сгенерированного уведомления.

Каждое новое уведомление должно отображаться с отступом от предыдущего.

Для пользователя с ролью администратор должны отображаться уведомления для всех типов ролей.

Создаваемое приложение будет полезно для следующих групп пользователей:

– для диспетчеров университета (роль Admin) – снижение временных затрат на ручное написание текстов рассылок и обеспечение мгновенной связи с кампусом,

– для преподавателей и сотрудников (роль teacher) – получение релевантной служебной информации, официальных распоряжений и уведомлений о собраниях кафедр,

– для студентов (роль student) – оперативное получение новостей об изменениях в расписании, бытовых ситуациях в общежитиях и экстренных оповещений.

На рисунке 1 представлена диаграмма вариантов использования для разрабатываемого приложения.

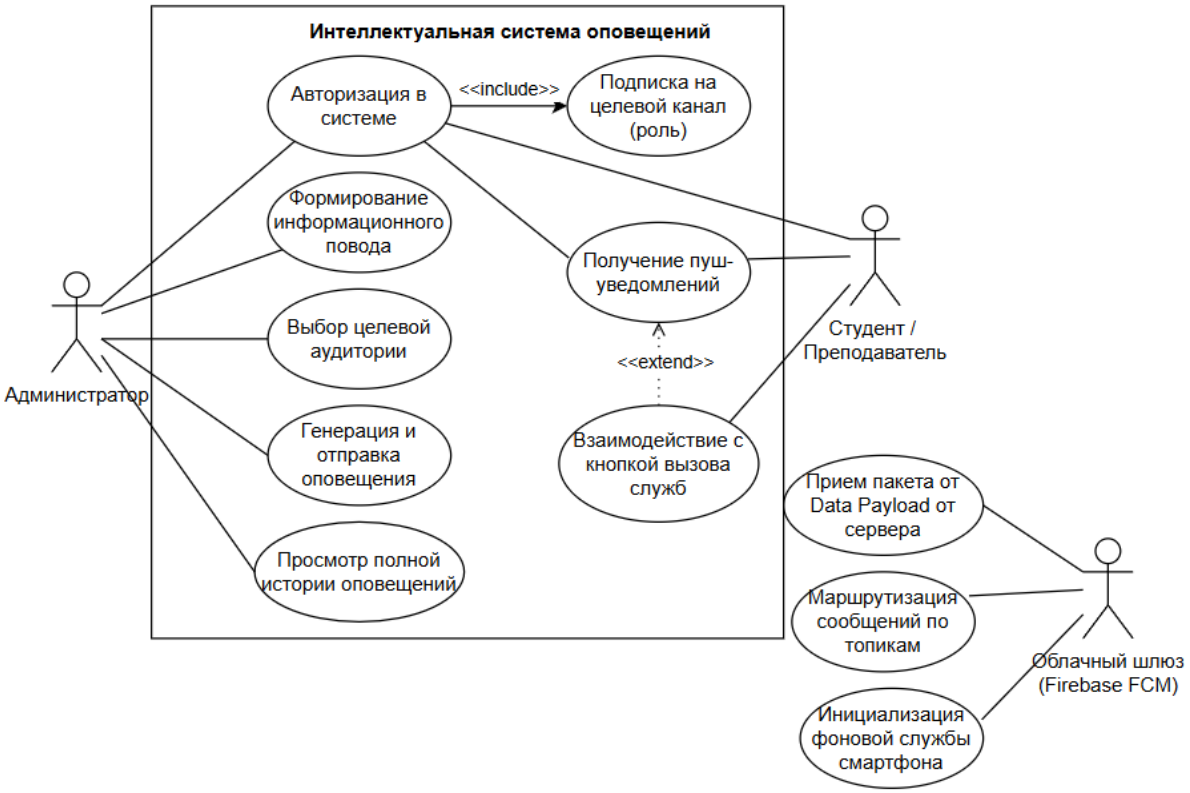


Рисунок 1 – Диаграмма вариантов использования

Такое исследование и последующая разработка позволят создать инновационное приложение, которое удовлетворяет потребности университета о важных информированиях пользователей в контексте мероприятий и чрезвычайных ситуациях.

При более углубленном исследование темы работы было выявлено, что наиболее ярко отображается в примерах промышленного производства, где ошибка или неверное действие, будет стоить цене человеческой жизни на производстве или потери крупных затрат предприятием.

2 Разработка методики и алгоритмов интеллектуальной системы информирования

2.1 Проектирование гибридной клиент-серверной архитектуры с интеграцией сторонних push-сервисов

Для решения задачи оперативного информирования и корректирования действий пользователей в образовательной среде была спроектирована гибридная клиент-серверная архитектура. Выбор данного подхода обусловлен необходимостью централизованного управления бизнес-процессами генерации контента и снижения вычислительной нагрузки на мобильные устройства пользователей.

Архитектура объединяет внутренний изолированный контур обработки данных, являющийся сервером ВУЗа с модулем искусственного интеллекта, и внешний транспортный шлюз, поэтому и является гибридной. Клиентское приложение отвечает исключительно за визуализацию пользовательского интерфейса, сбор первичных данных и прием асинхронных событий, в то время как вся бизнес-логика, включая работу с реляционными базами данных и генерацию текстов, расположена на стороне бэкенда.

Интеграция стороннего сервиса доставки уведомлений решает проблему поддержания постоянного TCP-соединения между мобильным устройством и сервером. В системах фоновое поддержание связи приводит к быстрому разряду батареи смартфона и прерыванию процессов со стороны операционной системы для экономии ресурсов. Делегирование транспортной функции специализированному облачному шлюзу позволяет использовать ресурсы клиентского приложения только в момент фактического поступления информационного пакета данных от сервера.

Процесс взаимодействия компонентов проектируемой гибридной системы, отвечающих за генерацию и доставку push-уведомлений, разделен на три логических этапа: ролевая регистрация на стороне клиента, интеллектуальная обработка событий на сервере и фоновая доставка уведомления через транспортный пуш-шлюз.

Первый процесс – это ролевая регистрация. В основе архитектуры лежит отказ от общей рассылки в пользу таргетированного информирования. На начальном этапе мобильное приложение осуществляет процесс аутентификации пользователя. В зависимости от валидированных учетных данных, алгоритм присваивает сессии определенную роль, например, «Студент» или «Преподаватель».

Далее инициируется процедура привязки к транспортному шлюзу. Клиентское приложение отправляет запрос к стороннему push-серверу - Google Firebase Cloud Messaging. Вместо генерации индивидуального токена устройства, требующего сложного администрирования на бэкенде, в системе реализован список подписок. Пользователю отправляется только та информация, касающаяся его роли, например, subscribeToTopic("Students"). Пуш-сервер регистрирует данную подписку в своей инфраструктуре. Серверная часть освобождается от необходимости хранения миллионов токенов конечных устройств, оперируя лишь именами ролевых каналов.

Интеллектуальная обработка и отправка уведомления начинается процессом на сервере при поступлении команды от администратора через REST API. Логика обработки события запускается на бэкенде, который перенаправляет простой и некрасивый текст в модуль локальной большой языковой модели для применения алгоритмов адаптивного рерайтинга и промпт-инжиниринга.

После генерации текста сервер приступает к формированию структуры уведомления. В отличие от стандартных шаблонов, бэкенд формирует скрытый пакет данных, исключая системные ключи визуализации. В JSON-объект упаковываются: заголовок, сгенерированный ИИ текст, а также метаданные, например, флаг emergency в случае выявления алгоритмом чрезвычайной ситуации. Используя серверную библиотеку SDK, бэкенд авторизуется в инфраструктуре Google по закрытому ключу и отправляет сформированный Data-пакет на push-сервер, указывая в качестве получателя целевой топик. Шлюз FCM принимает запрос, осуществляет маршрутизацию

и обеспечивает TCP-доставку пакета на мобильные устройства, подписанные на данный топик.

И в завершении процесс приема данных и обратная связь. Поскольку доставка осуществляется через скрытый Data Payload, операционная система Android не перехватывает сообщение. Пакет данных инициирует функции фоновой службы мобильного приложения. Клиентский код самостоятельно парсит метаданные и отображает локальное уведомление, интегрируя в него интерактивные элементы в виде кнопок, если таковые требуются логикой экстренного реагирования.

Пример взаимодействия компонентов системы при генерации и доставки оповещения представлен на диаграмме последовательности (рисунок 2).

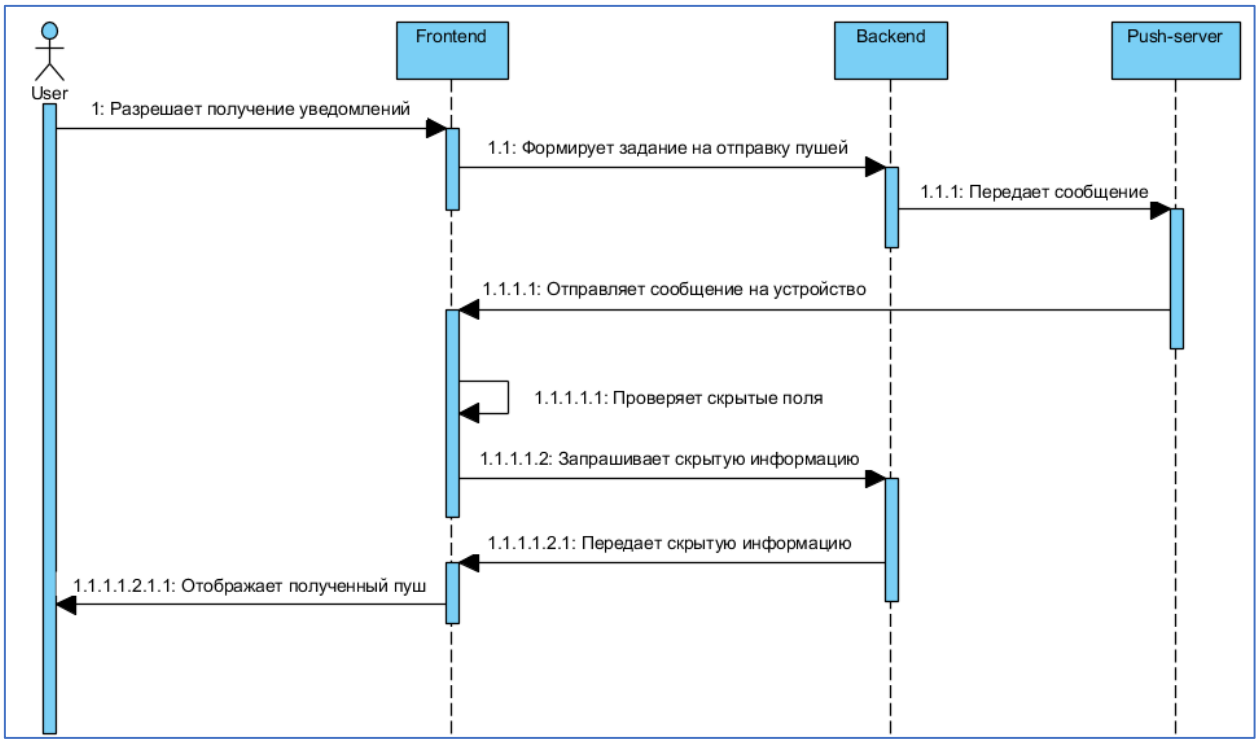


Рисунок 2 – Диаграмма последовательности

Диаграмма описывает основные события при создании и отправки уведомлений между пользователем, фронтом, бэкэндом и пуш-сервером.

2.2 Сравнительный анализ больших языковых моделей

Большие языковые модели представляют собой подтип нейронных сетей, разработанный для анализа и генерации работы с текстовыми данными. Такие модели обучаются на огромных массивах данных. Они используют

архитектуру трансформеров, которая эффективно обрабатывает большие объемы текстовой информации, сохраняя контекст даже на длинных текстовых последовательностях.

Внедрение и использование искусственного интеллекта для автоматизации бизнес-процессов и адаптации текстов уведомлений требует правильного выбора языковой модели. Выбор архитектуры напрямую зависит от преимуществ разрабатываемого приложения. Таких как безопасность, стоимость и независимость разрабатываемой корпоративной системы.

Существующие решения по внедрению LLM можно классифицировать на три основные категории:

– зарубежные облачные решения,

– отечественные облачные решения,

– открытые локальные модели (Open-Source LLM).

К первой категории относятся семейство GPT от OpenAI, Claude от Anthropic. Интеграция моделей осуществляется через синхронные REST API запросы к серверам компаний-разработчиков. Модели, например, GPT-4, GPT-3.5-Turbo на данный момент обладают наивысшими метриками качества генерации, логического вывода и понимания контекста.

Отечественные облачные решения YandexGPT, GigaChat из второй категории. Модели, разработанные российскими IT-корпорациями, функционируют в правовом поле Российской Федерации и адаптированы под русский язык.

Не так давно произошло большое развитие в решениях открытых локальных моделей семейства Llama, Mistral, Qwen. Они разнообразны по типу задач, архитектуре, размеру параметров и другим характеристикам. Данные модели позволяют скачивать веса нейронной сети и запускать их на собственном оборудовании организации. LLaMA 4 Scout является одной из самой мощной открытой локальной языковой моделью.

Сравнительный анализ с перечнем достоинств и недостатков каждого из трех типов больших языковых моделей представлен в таблице 4.

Таблица 4 – Преимущества и недостатки больших языковых моделей

Тип LLM	Преимущества	Недостатки
Зарубежные облачные решения	Максимальное качество семантики, отсутствие необходимости в собственных вычислительных мощностях, простота интеграции	Полная зависимость от внешнего интернет-соединения, регулярные финансовые затраты на оплату токенов, нарушение политик информационной безопасности образовательного учреждения РФ. Использование зарубежных API подразумевает передачу внутренних корпоративных данных на серверы третьих лиц за пределами юрисдикции РФ [9]
Отечественные облачные решения	Размещение серверов на территории РФ, высокая скорость работы с русскоязычным контекстом, наличие грантов или бесплатных лимитов для образовательных проектов	Данные покидают изолированный контур университета. Кроме того, сохраняется риск изменения ценовой политики и зависимость от стабильности внешних каналов связи в моменты чрезвычайных ситуаций
Открытые локальные модели	Данные обрабатываются внутри локальной вычислительной сети университета, исключая риск утечек. Система способна генерировать оповещения даже при полном отключении внешнего интернета. Бесплатное использование в исследовательских и коммерческих целях	Высокие требования к аппаратному обеспечению (объему видеопамяти VRAM) и необходимость применения методов математической оптимизации для запуска на доступном оборудовании

При выборе большой языковой модели следует опираться на требования к разрабатываемой системе и доступные ресурсы на ее внедрение в сервис.

2.3 Обоснование языковой модели в разрабатываемом приложении

На основании проведенного анализа, для реализации гибридной системы оповещения была выбрана концепция локального вывода результатов. Главным фактором выбора данной модели для разработки является обеспечение информационной безопасности и автономности процесса генерации экстренных уведомлений, а также их легальном использовании на территории Российской Федерации.

В качестве базовой нейросети была выбрана модель Saiga Llama-3 8B. Данный выбор обусловлен следующими факторами:

– модель базируется на архитектуре Llama-3 от компании Meta, которая является одной из самых производительных и логически выверенных моделей в классе с 8 миллиардами параметров,

– версия «Saiga» разработана российским исследователем И. Гусевым и прошла процедуру глубокого дообучения на обширных датасетах русскоязычных диалогов,

– веса выбранной модели конвертированы в бинарный формат GGUF, что позволит применить математический метод 4-битного квантования,

– в отличие от базовых зарубежных моделей, она показывает нативное владение русским культурным кодом и строго следует сложным системным инструкциям, что очень важно для промпт-инжиниринга.

В результате выбор квантованной модели Saiga Llama-3 8B позволил решить главную инженерную задачу исследования: развернуть высокоинтеллектуальную автономную систему генерации текстов на доступном графическом оборудовании и обеспечить при этом 100% безопасность корпоративных данных образовательного учреждения.

2.4 Математическое и алгоритмическое обеспечение модуля ИИ (методы квантования и промпт-инжиниринга)

Главным звеном разработанной системы является модуль генерации естественного языка, построенный на базе больших языковых моделей. Использование полновесных моделей гарантирует рост требований к объему видеопамяти. Для процесса генерации в условиях ограниченных аппаратных ресурсов коммерчески доступного оборудования был применен математический метод квантования весов нейронной сети.

Квантование представляет собой процесс аппроксимации высокоточных значений в пространство более низкой размерности. В рамках системы используется 4-битное целочисленное квантование, при котором веса модели группируются в блоки, и для каждого блока вычисляются масштабирующие коэффициенты [10]. Это позволяет сократить потребление оперативной памяти видеокарты более чем в три раза при статистически незначимом уровне изменения качества генерации текста уведомления.

Алгоритмическое обеспечение качества выдаваемого текста реализуется через методы промпт-инжиниринга. Это часть описания инструкции, которая подсказывает модели, каким образом подстроить свой ответ. Подбор промта – эта важная задача. Подобрать правильный промт порой сложно, но он будет давать значительный уровень качества ответов модели.

Для формирования промта использовались следующие правила [11]:

– формулирование точности задачи без использования общих фраз,

– указание роли модели,

– определение формата ответа: текст, список, таблица, код, картинка,

– задание стиля и тональности: деловой, нейтральный, разговорный,

– предоставить пример структуры и формата, если запрос сложный.

Нейросеть получает не просто тип события, а структурированный запрос с информацией о роли, контексте и ограничениях. Алгоритм автоматически подставляет переменные в заранее подготовленный шаблон промпта.

Алгоритм разрабатываемой генерации описывается следующей последовательностью:

– инициализация системного контекста: модели задается строгий поведенческий паттерн «роль системы экстренных оповещений» и лингвистические ограничения, например, исключение вводных конструкций,

– семантическое усечение: задается математическое ограничение на максимальное количество генерируемых токенов. Алгоритм прерывает вывод при достижении лимита, гарантируя, что объем итогового текста не превысит физические размеры шторки уведомлений мобильной операционной системы,

– температурное сэмплирование: для подавления «галлюцинаций» нейросети и обеспечения детерминированности ответов, гиперпараметр temperature алгоритма softmax устанавливается на пониженное значение, что заставляет модель выбирать наиболее статистически вероятные слова вместо креативных конструкций.

2.5 Алгоритм маршрутизации бизнес-процессов на основе ролевой модели доступа

Для обеспечения информационной безопасности и предотвращения утечек корпоративных данных спроектирована система маршрутизации, базирующаяся на модели управления доступом на основе ролей [12]. Система дифференцирует информационные потоки на два независимых процесса: доставку данных в реальном времени и асинхронный доступ к историческим записям.

Маршрутизация push-уведомлений строится на основе топиков, то есть ролевых каналов. При первичной авторизации клиентское устройство, в зависимости от присвоенной пользователю роли, инициирует алгоритм подписки на соответствующий информационный канал. Серверный модуль, связывает пакет данных с целевой ролью. Таким образом, разделение доступа происходит на уровне стороннего сетевого шлюза, что физически исключает поступление пакетов на устройства пользователей без соответствующих привилегий.

Доступ к истории бизнес-процессов регламентируется принципом серверной фильтрации. Клиентское приложение рассматривается как

небезопасная среда, следовательно, фильтрация данных на стороне конечного устройства недопустима. При обращении клиента к API, алгоритм бэкенда анализирует идентификатор роли:

– при наличии ограниченных прав, то есть ролей конечных пользователей, алгоритм динамически инъецирует предикат фильтрации WHERE в запрос к реляционной базе данных,

– при наличии прав администратора предикат игнорируется, и алгоритм предоставляет полную истории для просмотра.

Данный подход гарантирует изоляцию данных и соблюдение принципа минимальных привилегий.

2.6 Алгоритм выявления триггерных ситуаций и формирования интерактивных скрытых пакетов

Для обеспечения безопасности жизнедеятельности пользователей на территории университета разработан алгоритм автоматического повышения приоритета оповещений при возникновении чрезвычайных ситуаций.

Традиционный механизм обработки уведомлений в мобильных операционных системах передает контроль за отображением сообщения системному процессу ОС, что ограничивает возможности приложения по изменению интерфейса в фоновом режиме. Для обхода данного ограничения алгоритм реализует динамическое переключение между стандартным уведомлением и "пакетом данных".

Ключевой особенностью предложенного алгоритма является отказ от жестких эвристических словарей. Практика показывает, что словарный поиск обладает высоким процентом ложных срабатываний. В разработанной архитектуре задача контекстного анализа и классификации угрозы делегирована локальной большой языковой модели.

На вход алгоритма поступает неструктурированный текст. Сервер динамически формирует структурированный контекст и передает его в языковую модель. Нейросеть осуществляет глубокий семантический анализ события с учетом специфики целевой аудитории. Результатом инференса

является не просто сгенерированный текст, а строго типизированный JSON-объект, содержащий булевый флаг критичности is_emergency. На рисунке 3 представлен алгоритм выявления триггерных ситуаций в виде блок схемы.

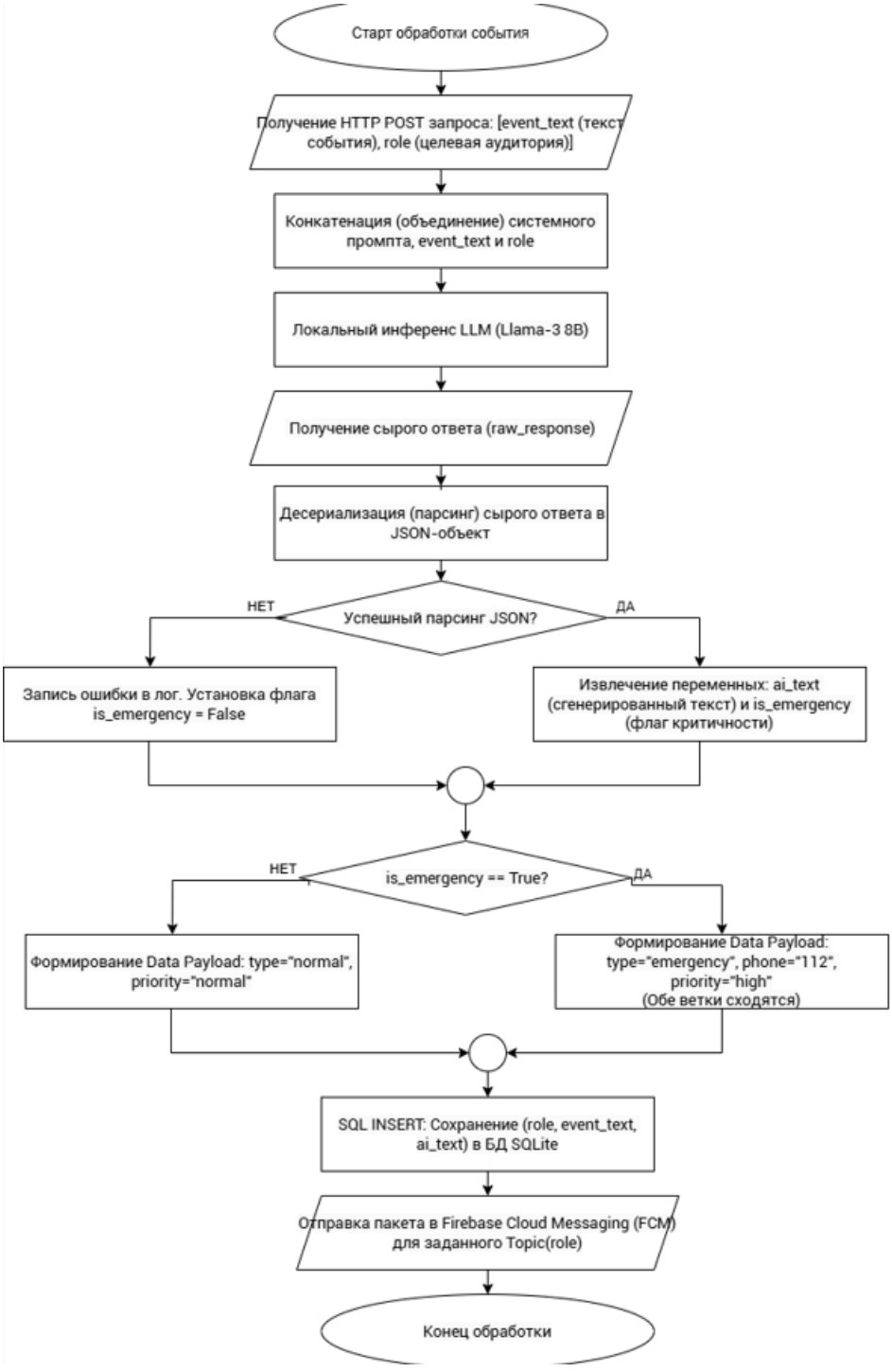


Рисунок 3 – Блок схема алгоритма

На этапе серверного процессинга алгоритм выполняет десериализацию ответа ИИ и оценивает полученный флаг.

В случае выявления реальной угрозы (is_emergency = true), алгоритм формирует специализированный пакет данных. Пакету программно присваивается максимальный сетевой приоритет (priority="high") для обхода встроенных механизмов энергосбережения ОС Android (Doze Mode). В метаданные пакета интегрируется системный интент (phone="112") для вызова экстренных служб. В случае штатных или бытовых инцидентов (is_emergency = false) формируется стандартный фоновый пакет.

Подобная структура пакета заставляет операционную систему целевого устройства принудительно активировать фоновую службу (Service) клиентского приложения. Мобильный клиент декодирует полученный JSON-массив и инициирует построение интерактивного пользовательского интерфейса с максимальным приоритетом (IMPORTANCE_MAX). В случае экстренной ситуации в интерфейс шторки уведомлений динамически интегрируются элементы управления (Action Buttons) для мгновенной маршрутизации пользователя (например, звонок в службу спасения). Это трансформирует процесс информирования из пассивного в активный, значительно снижая время когнитивной реакции студента или сотрудника в условиях стресса.

3 Программная реализация информационного комплекса

3.1 Обоснование выбора технологического стека разработки

Для реализации спроектированной гибридной архитектуры был выбран современный стек технологий, обеспечивающий высокую производительность, безопасность и модульность системы.

В серверной части в качестве основного языка программирования выбран Python 3.12, являющийся технологическим стандартом для интеграции моделей машинного обучения. Разработка REST API осуществлена на базе асинхронного микрофреймворка FastAPI в связке с ASGI-сервером Uvicorn. Выбор FastAPI обусловлен встроенной валидацией данных через библиотеку Pydantic и высокой скоростью обработки неблокирующего ввода-вывода.

В качестве использования искусственного интеллекта использовалась открытая языковая модель Llama-3 8B формат GGUF, 4-битное квантование. Для инференса квантованной модели использована библиотека llama.cpp через Python-обертку llama-cpp-python. Данное решение позволяет напрямую компилировать код под архитектуру NVIDIA CUDA, перенося вычисления на ядра графического ускорителя без избыточных накладных расходов, характерных для тяжеловесных фреймворков.

Ввиду отсутствия необходимости в распределенных транзакциях, для работы функции истории оповещений применена встраиваемая реляционная СУБД SQLite 3.

Нативное Android-приложение разработано в среде Android Studio на языке Kotlin. Сетевое взаимодействие реализовано через библиотеку OkHttp, а доставка фоновых push-уведомлений интегрирована через официальный SDK Firebase Cloud Messaging.

Для наглядного представления взаимодействия каждого узла компонентов разработана диаграмма развертывания, представленная на рисунке 4.

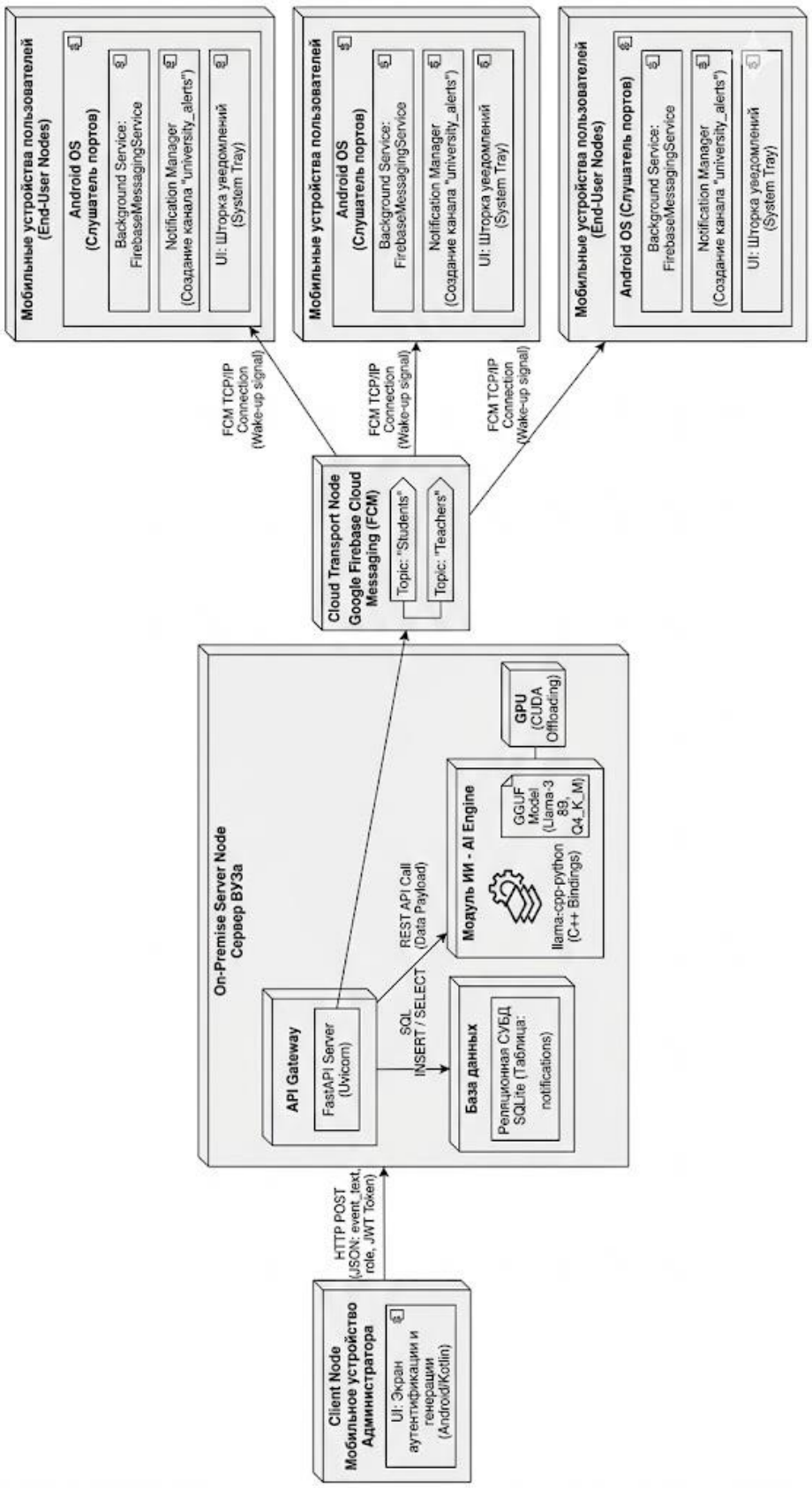


Рисунок 4 – Диаграмма развертывания компонентов

Таким образом будут взаимодействовать между собой все узлы разрабатываемой системы.

3.2 Интеграция локальной языковой модели

Главной архитектурной особенностью разработанной системы является отказ от использования сторонних облачных сервисов, таких как API OpenAI ChatGPT, в пользу локального инференса. Модель разворачивается непосредственно на сервере образовательного учреждения, что гарантирует 100% конфиденциальность внутренних данных и независимость от интернет-соединения.

Взаимодействие с моделью осуществляется через библиотеку llama-cpp-python, которая позволяет загружать веса квантованной нейросети напрямую в видеопамять графического ускорителя [13]. В коде предусмотрена настройка гиперпараметра temperature – он регулирует степень «творчества», иными словами энтропию нейросети, позволяя сделать экстренные сообщения строго фактологическими и информативными. В листинге 1 представлен фрагмент инициализации модели и функции генерации.

Листинг 1 – Фрагмент кода локального инференса LLM

```
from llama_cpp import Llama
#инициализация квантованной модели в GPU
llm = Llama(
    model_path="models/model-q4_k.gguf",
    n_ctx=2048,
    n_gpu_layers=-1, #полная выгрузка вычислений на видеокарту
    chat_format="llama-3"
)
def generate_alert_text(event_type, role):
    response = llm.create_chat_completion(
        messages=[
            {"role": "system", "content": system_instruction},
            {"role": "user", "content": user_prompt}],
        temperature=0.4,
        max_tokens=100   #лимит длины для push-уведомлений
    )
    return response['choices'][0]['message']['content']
```

Параметр n_gpu_layers=-1 форсирует выгрузку 100% слоев нейросети в графический процессор. Логика генерации текста и применение промпт-инжиниринга инкапсулированы в функции generate_text.

Использование локального движка генерации обеспечивает независимость модуля и упрощает масштабирование системы внутри контура безопасности ВУЗа. Такая архитектура позволяет мгновенно генерировать тексты разной тональности – от формальных предупреждений об эвакуации до приглашений на мероприятия.

3.3 Реализация алгоритмов формирования контекстных промптов

Для влияния на характер ответов был сформирован промт. Нейросеть получает не просто тип события, а структурированный запрос. В листинге 2 представлен фрагмент кода конструирования промпта.

Листинг 2 – Фрагмент кода формирования системного промпта

```
def generate_text(event_text: str, role: str) -> dict:
    """
    ИИ анализирует любое событие (от концерта до ЧС) и
возвращает JSON.
    """
    system_instruction = (
        "Ты — умный помощник университета. Твоя задача:
проанализировать событие и скорректировать "
        "действия пользователя в тексте уведомления в два
максимум предложения.\n\n"
        "ПРАВИЛА АДАПТАЦИИ ПОД РОЛЬ:\n"
        "1. Текст должен зависеть от роли и типа события:\n"
        "  - Мероприятие (концерт, праздник): студентов
приглашай, а преподавателей проси проинформировать группы.\n"
        "  - Учебный процесс (отмена пар, карантин): студентам
скажи учиться дома, преподавателям — перевести занятия в
онлайн.\n"
        "  - Экстренная ситуация (пожар, угроза): студентам
скажи спасаться/эвакуироваться, преподавателям — организовать
эвакуацию студентов.\n\n"
        "ФЛАГ КРИТИЧНОСТИ (is_emergency):\n"
        "- Ставь is_emergency: true ТОЛЬКО при угрозе жизни
(пожар, нападение, обрушение).\n"
        "- Для ВСЕГО ОСТАЛЬНОГО (концерты, отмена пар,
отключение воды, жара) ставь is_emergency: false.\n\n"
        "ОТВЕТЬ СТРОГО В ФОРМАТЕ JSON. Пример:\n"
        '{"text": "Завтра в 18:00 состоится праздничный концерт
в актовом зале. Ждем всех желающих!", "is_emergency": false}'
    )
```

Правильно сформированный промпт позволяет ИИ эффективно решать поставленные задачи, обеспечивая соответствие текста контексту чрезвычайной ситуации или навигационной задачи.

3.4 Проектирование REST API и реализация серверной фильтрации

Для создания безопасной среды и реализации ролевой структуры доступа, логика фильтрации исторических данных перенесена в среду базы данных [14]. Взаимодействие мобильного клиента с базой данных производится через защищенный эндпоинт GET /history.

При обращении к данным клиентское приложение передает идентификатор роли пользователя. Алгоритм бэкенда в реальном времени формирует SQL-запрос с предикатом WHERE (листинг 3), исключая попадание конфиденциальных записей других групп в конечный JSON-ответ.

Листинг 3 – Реализация серверной фильтрации данных

```
    def get_all_notifications(filter_role=None):
        conn = sqlite3.connect(DB_NAME)
        conn.row_factory = sqlite3.Row
        cursor = conn.cursor()
        # Ролевая маршрутизация SQL-запросов
        if filter_role == "student":
            cursor.execute('SELECT * FROM notifications WHERE
role = "Студенты" ORDER BY id DESC LIMIT 50')
        elif filter_role == "teacher":
            cursor.execute('SELECT * FROM notifications WHERE
role = "Преподаватели" ORDER BY id DESC LIMIT 50')
        else:
            # Режим суперпользователя (admin) - без ограничений
            cursor.execute('SELECT * FROM notifications ORDER BY
id DESC LIMIT 50')
        rows = cursor.fetchall()
        conn.close()
        result = [{"id": r["id"], "role": r["role"],
"event_text": r["event_text"],
                   "ai_generated_text": r["ai_generated_text"],
"timestamp": r["timestamp"]} for r in rows]
        return result
```

При поступлении запрос от администратора (role=admin) возвращается полный список таблицы в момент времени для аудита.

3.5 Интеграция транспортного шлюза и формирование пакетов Data Payload

После интеллектуальной обработки текста и генерации текста система формирует пакет для отправки через облачный шлюз Firebase Cloud Messaging. В системе применяется маршрутизация на основе тематических подписок FCM Topics, что избавляет сервер от необходимости хранить индивидуальные токены устройств.

Ключевой архитектурной особенностью реализации является анализ контекста сообщения на предмет чрезвычайных ситуаций пожара или эвакуации ИИ. Вместо регулярных выражений на поиски слов, закодированных словарей, применяется другой подход анализа на выявление триггерных ситуаций. На основе системного промпта нейросеть возвращает структурированный JSON-объект, содержащий не только адаптированный текст уведомления, но и булевый флаг is_emergency, определяющий степень угрозы жизнедеятельности для конкретной группы пользователей.

Для гарантированной доставки экстренных оповещений в системе используется метод скрытых пакетов данных [15]. Из исходящего пакета намеренно исключен стандартный системный блок notification. Это не позволяет мобильной ОС перехватывать уведомление в фоновом режиме.

Кроме того, пакету присваивается максимальный сетевой приоритет priority="high", что позволяет обходить встроенные механизмы энергосбережения ОС Android. В листинге 4 представлен фрагмент реализации данной логики.

Листинг 4 – Парсинг ответа ИИ и формирование высокоприоритетного Data Payload

```
import json
from firebase_admin import messaging
def generate_notification(request_event_text: str, request_role:
str, topic: str):
    #структурированный JSON
    ai_result_dict = generate_text(request_event_text,
request_role)
```

```
ai_text = ai_result_dict.get("text", "Ошибка генерации.")
is_emergency = ai_result_dict.get("is_emergency", False)
#формирование скрытого пакета данных
data_payload = {
    "title": "Оповещение ВУЗа",
    "body": str(ai_text),
    "type": "emergency" if is_emergency else "normal",
    "phone": "112"
}
#высокий приоритет доставки
message = messaging.Message(
    data=data_payload,
    topic=topic,
    android=messaging.AndroidConfig(
        priority="high" #обход режима сна (Doze Mode)
    )
)
#отправка пакета в облачный шлюз
response = messaging.send(message)
return response
```

Описание основных файлов с кодом серверной части представлены в таблице 5.

Таблица 5 – Файлы серверной части приложения

Название	Назначение	Ссылка на код модуля
main.py	Отвечает за главный вход в программу. Здесь реализовано использование FastAPI, проверка текста на слово экстренные ситуации (например, «пожар») отправка скрытых пакетов Data Payload через Firebase	Приложение А (Листинг А.1)
database.py	Прописана функция работы с БД, осуществляется серверная фильтрация данных по ролям	Приложение А (Листинг А.2)
ai_engine.py	Модуль работы с LLM и определение системного промта	Приложение А (Листинг А.3)

Таким образом в приложении реализован модуль серверной части приложения с использованием языка программирования Python.

3.6 Разработка мобильного клиента и реализация бизнес-логики реагирования

Мобильное приложение Android отвечает за отображение интерфейса и инициализацию бизнес-процессов. При авторизации приложение автоматически подписывается на соответствующий FCM-топик, например, Students.

За обработку скрытых Data-пакетов отвечает фоновая служба MyFirebaseService на языке Kotlin. Пакет принудительно «будит» приложение, позволяя извлечь метаданные type и phone. В штатном режиме формируется классическое уведомление. Если сервер пометил пакет как экстренный type="emergency", клиентский код повышает приоритет канала до максимального PRIORITY_MAX и динамически интегрирует в шторку уведомлений интерактивную кнопку действия (листинг 5).

Листинг 5 – Динамическая генерация интерактивного уведомления в Android

```
override fun onMessageReceived(remoteMessage: RemoteMessage) {
    val data = remoteMessage.data
    val title = data["title"] ?: "Оповещение ВУЗа"
    val body = data["body"] ?: ""
    val type = data["type"] ?: "normal"
    val phone = data["phone"] ?: "112"
    val builder = NotificationCompat.Builder(this,
"university_alerts")
        .setSmallIcon(android.R.drawable.ic_dialog_alert)
        .setContentTitle(title)
        .setContentText(body)
        .setAutoCancel(true)
    // Реализация инициализации экстренного бизнес-процесса
    if (type == "emergency") {
        val callIntent = Intent(Intent.ACTION_DIAL).apply { data
= Uri.parse("tel:$phone") }
        val pendingCall = PendingIntent.getActivity(
            this, 1, callIntent,
            PendingIntent.FLAG_ONE_SHOT or
PendingIntent.FLAG_IMMUTABLE
        )
        // Инъекция кнопки звонка напрямую в уведомление
        builder.addAction(android.R.drawable.ic_menu_call,
"ПОЗВОНИТЬ $phone", pendingCall)
        builder.setPriority(NotificationCompat.PRIORITY_MAX)
    }
```

```
    val manager = getSystemService(Context.NOTIFICATION_SERVICE)
as NotificationManager
    manager.notify(System.currentTimeMillis().toInt(),
builder.build())
}
```

Нажатие на кнопку «ПОЗВОНИТЬ 112» автоматически перенаправляет пользователя в системное приложение для осуществления вызовов. Благодаря этому, реализованный алгоритм позволяет пользователю мгновенно запустить важный бизнес-процесс, не теряя время на разблокировку устройства и запуск приложения, что существенно повышает уровень безопасности в условиях чрезвычайных ситуаций.

Основные файлы мобильного клиента представлены и описаны в таблице 6.

Таблица 6 – Файлы мобильного клиента Android приложения

Название	Назначение	Ссылка на код модуля
MyFirebaseService.kt	Реализована функция телефона в фоновом режиме, проверка критичности уведомления и отображение кнопки, если требует тип уведомления	Приложение А (Листинг А.4)
LoginActivity.kt	Прописана логика безопасности данных, проверка логина и пароля, направление администратора и других ролей пользователей в доступный им функционал	Приложение А (Листинг А.5)
HistoryActivity.kt	Модуль взаимодействия с сервером по сети OkHttp, скачивание JSON с историей и вывод его на экран	Приложение А (Листинг А.6)

Таким образом в приложении реализован модуль клиентской части приложения с применением языка программирования Kotlin.

4 Экспериментальное исследование и контроль качества программного обеспечения

4.1 Мероприятия по контролю качества программного обеспечения

Качество программного обеспечения – это совокупность характеристик, относящихся к способности удовлетворять ранее установленные потребности. То есть насколько система соответствует зафиксированным требованиям и ожиданиям пользователя или заказчика.

Контроль качества включает в себя тестирование и соответствие уровню качества продукта, который будут выпускать. Основная цель контроля качества предоставить обзор готовности продукта на разных этапах разработки. При положительной оценке контроля качества, можно утверждать, что требования соблюдены. В зависимости от специфики предметной области, можно выполнить следующие варианты по обеспечению контроля качества системы:

– тестирование документации,

– ревью кода на соответствие стандартам,

– внедрение методик по работе с качеством.

Мероприятия по контролю качества применялись на всех жизненных циклах программного обеспечения. Была проверена целостность базы данных, данные сначала были отобраны, а затем провелась проверка на согласованность информации между собой. Программный код просмотрен на наличие текстовых ошибок и верной пунктуации. Все этапы сопровождались описанием текста пояснительной записки. А также сверялись требования и цели к исследовательской работе, которые были поставлены на этапе проектирования [16].

Осуществление контроля качества приложения дает возможность свести к минимуму наличие ошибок, что повышает собственную эффективность системы за счет бесперебойной работы со стороны пользователя.

4.2 Осуществление производительности и качества работы приложения

4.2.1 Проведение нагрузочного тестирования приложения

Тестирование – это исследование готовой системы прежде, чем она будет передана для использования конечными пользователями. Тестирование позволяет выявить недоработки, ошибки и всевозможные проблемы с различными сценариями использования. [17]. Таким образом, приложение будет иметь повышенную популярность среди пользователей различных устройств и платформ.

Тестирование проводится с помощью такого инструмента тестировщиков, как инструкция. Это документы, в которых описываются то, что нужно проверить, с какими условиями и действиями. Для каждой функции создается свой тест-кейсы, что позволяет удобно работать со всей информацией, видеть входные и выходные данные.

Нагрузочное тестирование – это вид нефункционального тестирования, который направлен на оценку поведения программной системы при ожидаемой и пиковой нагрузке. Его основная цель – выявить узкие места в производительности, определить пределы масштабируемости и убедиться, что система соответствует требованиям по скорости отклика, пропускной способности и стабильности [18].

Так как система на тестировании является локальная квантованная модель Llama-3 8B, работающая через llama-cpp-python, главным критерием качества является скорость инференса, то есть генерация, и соответствие текста системному промпту.

Тестирование проводилось на стенде с графическим ускорителем потребительского класса (NVIDIA RTX 4060, 8 ГБ VRAM (видеопамяти)) с полной выгрузкой слоев нейросети в GPU.

В режиме простоя модель занимает ~4.9 ГБ. При пиковой генерации контекста потребление не превышает 5.3 ГБ, что доказывает отсутствие утечек памяти и стабильность приложения.

Среднее время ответа составляет 0.8 сек. Полная генерация уведомления в среднем 40-50 токенов занимает 1.5 – 2.2 секунды, что полностью соответствует требованиям к системам оповещения реального времени.

Проведение нагрузочного тестирования предотвращает появлению сбоев и недоступности системы под нагрузкой. Помогает выявить и устранить узкие места для улучшения скорости работы приложения. При таком виде тестирования можно определить предел системы и необходимости масштабирования.

Нагрузочное тестирование является критически важным этапом в жизненном цикле разработки программного обеспечения, особенно для системы, которая должна работать под высокой нагрузкой и обеспечивать бесперебойную работу для большого числа пользователей.

4.2.2 Семантическое тестирование

Для проверки качества текста был проведен тест на 30 различных информационных поводах: отмена занятий, субботник, пожарная тревога, изменения погодных условий. Оценивались два параметра: отсутствие галлюцинаций и соблюдение лимитов.

Благодаря параметру temperature = 0.4, модель ни разу не исказила факты исходного события. При разработке сервиса добавлялись критические ситуации и механизмы корректировки действий пользователя, поэтому в результатах могут быть представлены как обычные события, так и инструкции к действию пользователей различных ролевый групп. Результаты представлены в таблице 7.

Таблица 7 - Результаты проверки генерации текста

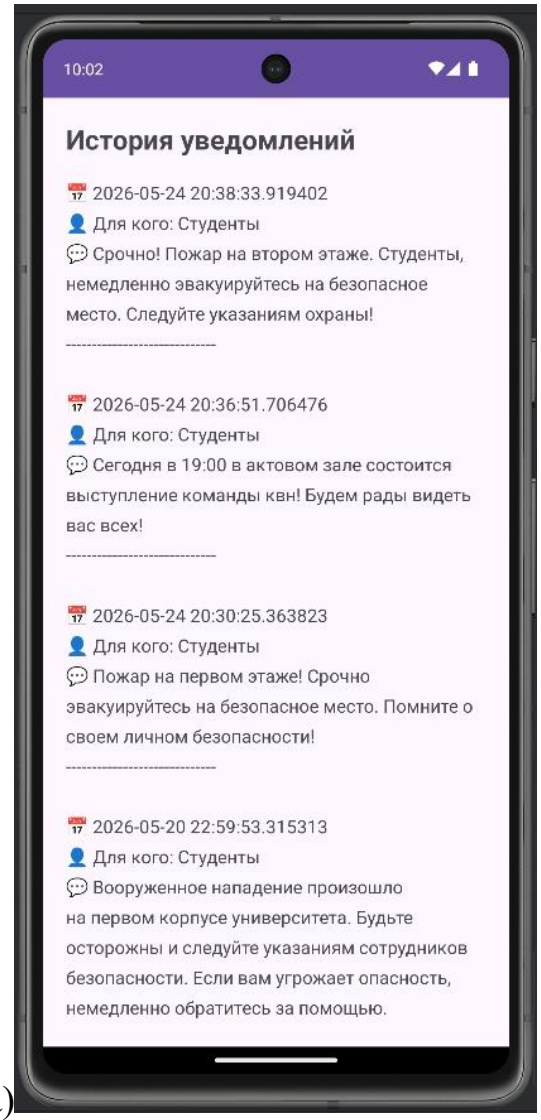
Событие	Роль	Текст генерации
Пожар на втором этаже	Студенты	Студенты, немедленно эвакуируйтесь на безопасное место. Следуйте указаниям охраны! (рисунок 5 а)
Пожар на втором этаже	Преподаватели	Пожалуйста, организуйте эвакуацию студентов. Будьте осторожны! (рисунок 5 б)

Продолжение таблицы 7

1	2	3
Выступление команды КВН	Студенты	Сегодня в 14:00 в актовом зале состоится выступление команды КВН! Будем рады видеть вас всех! (рисунок 5 а)
Выступление команды КВН	Преподаватели	Сегодня в 14:00 в актовом зале состоится выступление команды КВН. Просим проинформировать студентов о мероприятии. (рисунок 5 б)
Вооружённое нападение в первом корпусе	Преподаватели	Вооружённое нападение произошло в первом корпусе. Будьте осторожны и следуйте указаниям охраны! (рисунок 5 б)
Пожар на втором этаже	Студенты	Пожар на первом этаже! Срочно эвакуируйтесь на безопасное место. Помните о своей личной безопасности. (рисунок 5 а)
Вооружённое нападение в первом корпусе	Студенты	Вооружённое нападение произошло в первом корпусе. Будьте осторожны и следуйте указаниям охраны! (рисунок 5 а)
22 апреля субботник	Студенты	Субботник 22 апреля! Присоединяйтесь к акции по благоустройству нашего университета (рисунок 6 а)
В столовой будет день блинчиков	Преподаватели	День блинчиков в столовой! Преподавателям: приятный перекус и вкусные блинчики ждут вас в столовой (рисунок 6 б)
Конкурс гранта	Студенты	Присоединяйтесь к программе стипендий для поддержки талантливых студентов. Заявки принимаются до 15 марта.
Каникулы	Студенты	Скоро каникулы!
Отмена занятий	Студенты	Сегодня утром университет объявил о полной отмене завтрашних занятий.
Скидка в столовой	Студенты	Скидка 50% в столовой! Для студентов.
Раздача новогодних подарков	Преподаватели	Раздача новогодних подарков для преподавателей!

Продолжение таблицы 7

1	2	3
Проверка контрольных	Преподаватели	Срочная проверка! Проведите контрольные работы по всем дисциплинам до конца дня.

а) 

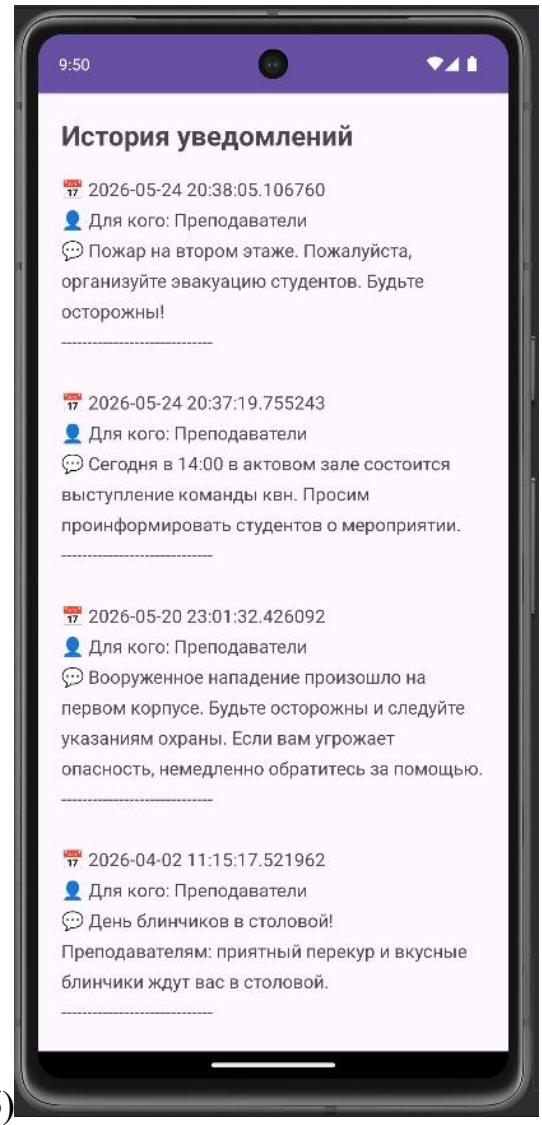
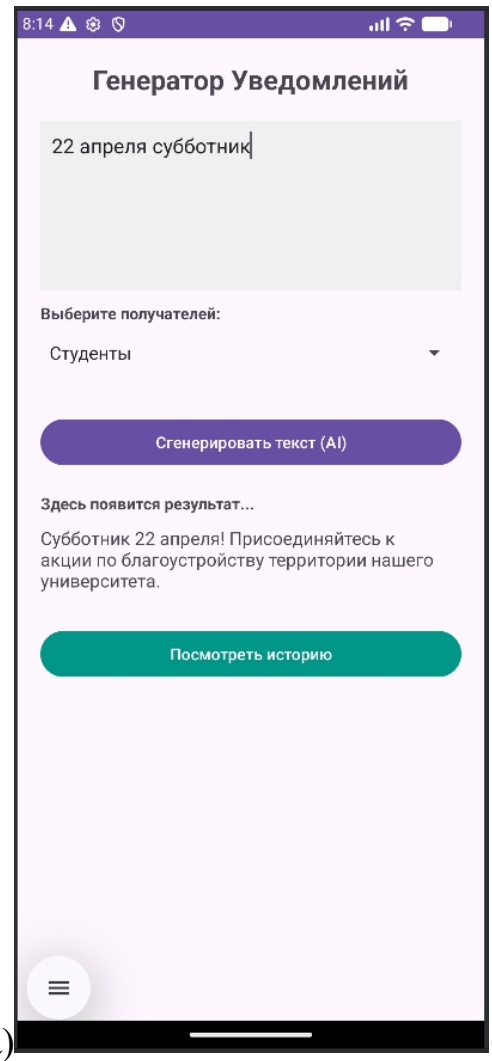
б) 

Рисунок 5 - Корректирование бизнес-процессов с учетом внешних условий: а) для студентов б) для преподавателей

а) 

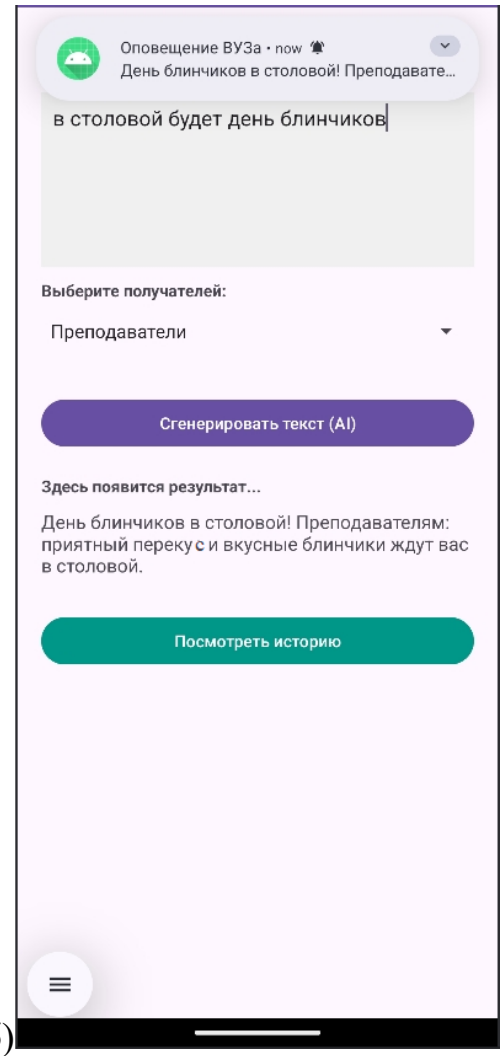
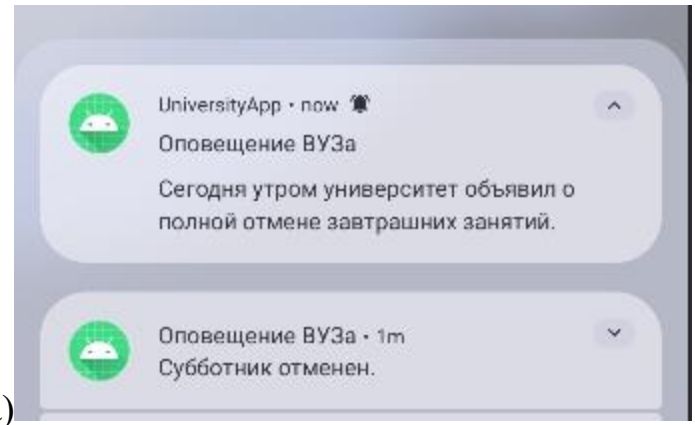
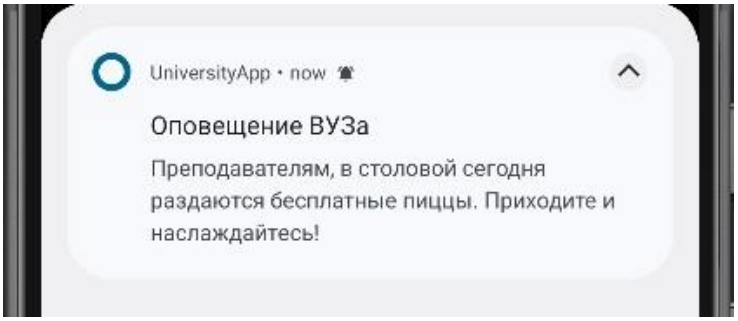
б) 

Рисунок 6 - Результаты проверки генерации текста: а) для студентов б) для преподавателей

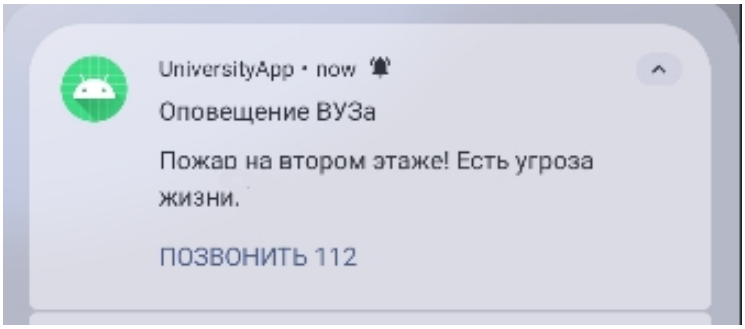
Гиперпараметр max_tokens = 100 и жесткий системный промпт не совсем успешно ограничили 100% сгенерированных текстов длиной до 2-х предложений. Тексты не всегда корректно помещались в область графического интерфейса шторки Android без визуального усечения (рисунок 7).

а) 

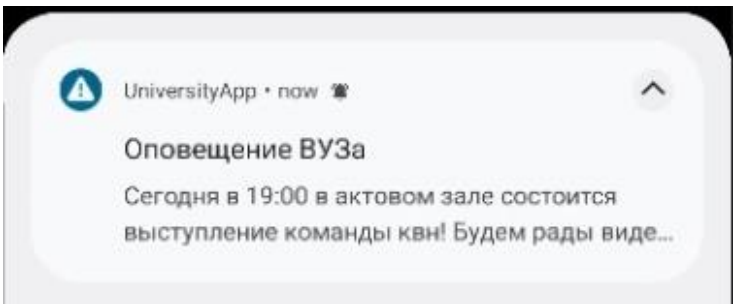
б)



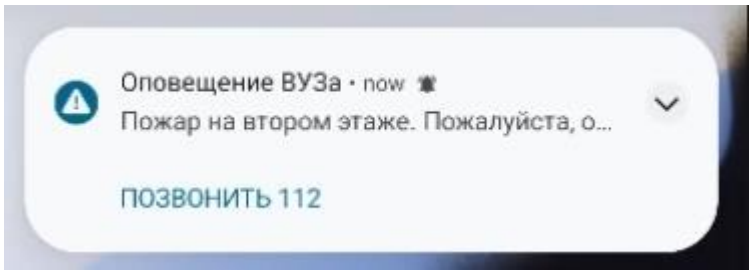
в)



г)



д)



е)

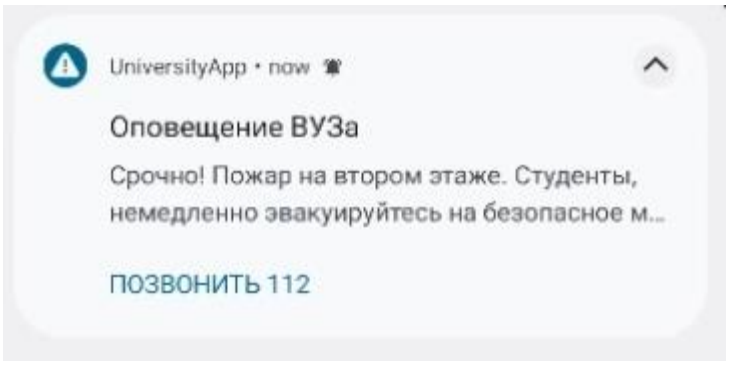


Рисунок 7 - Результаты проверки генерации текста а-е

В отличие от обычного функционального тестирования, которое проверяет, выполняет ли система заданные операции, семантическое

тестирование углубляется в то, правильно ли система понимает и обрабатывает значения, отношения и контекст данных.

4.3 Интеграционное тестирование транспортного слоя и API

Взаимодействие между мобильным клиентом Android и сервером FastAPI было протестировано с использованием методов сквозного тестирования End-to-End.

Сквозное тестирование – это методология тестирования, которая проверяет работу всего программного приложения от начала до конца, имитируя реальные сценарии использования конечными пользователями. Его главная цель – убедиться, что все интегрированные компоненты системы работают корректно вместе, и поток данных между ними происходит без ошибок [19].

Современные системы ПО сложны и взаимосвязаны с большим количеством подсистем, которые могут существенно отличаться от существующих систем. Вся система может разрушиться из-за отказа любой подсистемы, что представляет собой серьезный риск. Этого риска можно избежать с помощью сквозного тестирования.

4.3.1 Тестирование фоновой доставки

Ключевым требованием к тестируемому приложению была гарантированная доставка пуш-уведомлений пользователям, даже когда приложение находилось в свернутом или полностью закрытом состоянии. Это требование является критически важным для обеспечения своевременного информирования пользователей о важных событиях, обновлениях или сообщениях.

В рамках данного тестирования была проведена проверка механизма передачи Data Messages. Этот подход предполагает отправку скрытых JSON-пакетов, которые не используют системный ключ notification. Такой метод позволяет приложению самостоятельно обрабатывать данные и создавать уведомления, что дает больший контроль над процессом.

Работа доставки пушей в фоновой работе приложения представлено на рисунке 8.

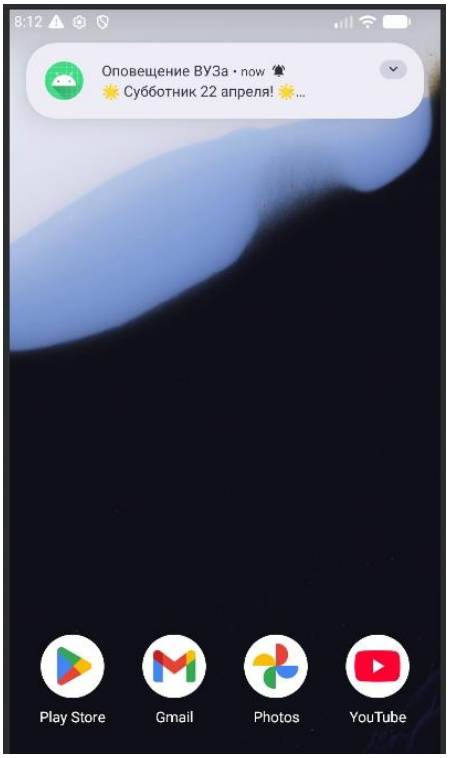


Рисунок 8 – Работа приложения в фоновом режиме

В результатах тестирования можно выделить следующие итоги:

– 100% доставка сообщений. Результаты тестирования подтвердили абсолютную надежность доставки Data Messages. Все отправленные JSON-пакеты были успешно получены приложением,

– работа фонового сервиса. Было установлено, что фоновый сервис MyFirebaseService эффективно перехватывает входящие Data Messages,

– создание локальных уведомлений. После перехвата пакета, MyFirebaseService успешно инициировал создание локальных уведомлений,

– приоритет уведомлений.

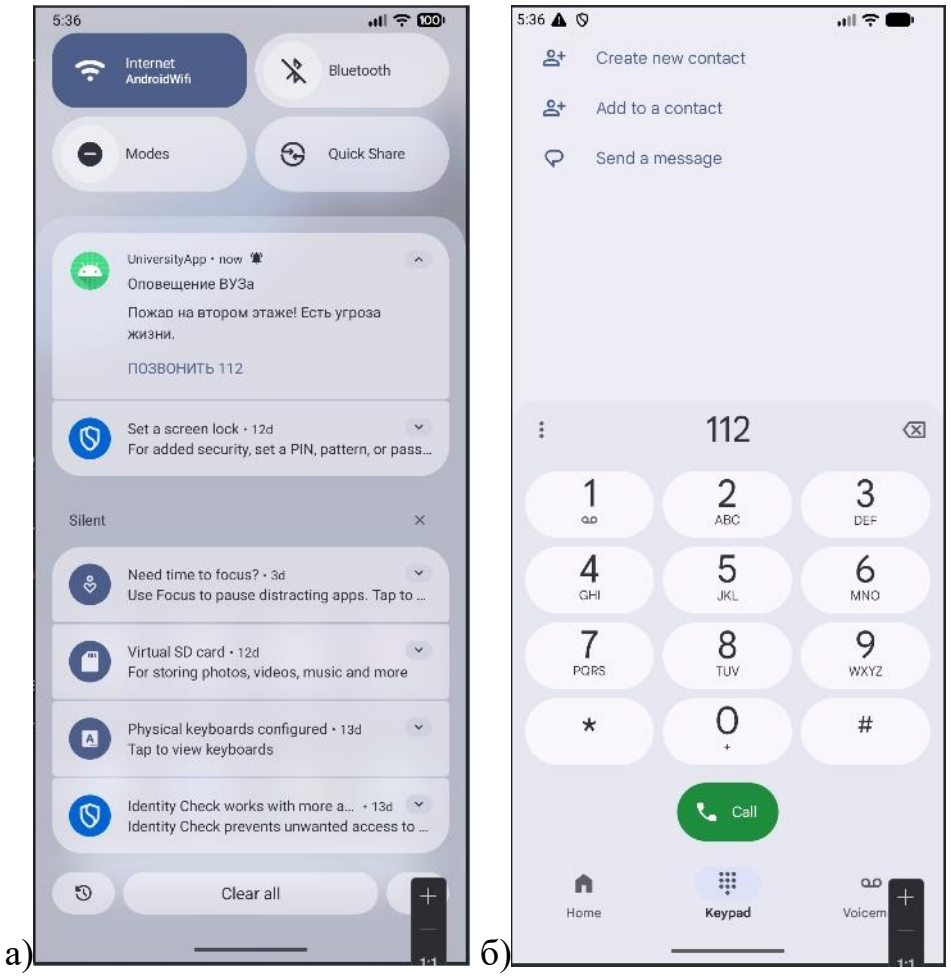
Важным аспектом стало то, что локальные уведомления создавались с использованием канала, имеющего приоритет IMPORTANCE_HIGH. Это гарантирует, что уведомления будут отображаться пользователю максимально заметно, даже если приложение работает в фоновом режиме или закрыто.

Проведенное тестирование подтвердило, что архитектура с использованием Data Messages и грамотно настроенный фоновый сервис

MyFirebaseService с высоким приоритетом уведомлений успешно справляются с критическим требованием приложения – гарантированной доставкой пуш-уведомлений в любых состояниях приложения. Это обеспечивает надежную коммуникацию с пользователями.

4.3.2 Тестирование интерактивных экстренных сценариев

Проведен тест триггерного механизма на сервере. При вводе ключевых слов, например, «пожар» сервер корректно присваивал пакету метаданные type: emergency. Мобильный клиент успешно декодировал данный флаг и динамически внедрял в уведомление интерактивный элемент «ПОЗВОНИТЬ 112», ассоциированный с системным PendingIntent для вызова экстренных служб (рисунок 9).



а) б)

Рисунок 9 – Работа приложения в фоновом режиме

4.4 Тестирование безопасности и ролевой модели доступа

Негативное тестирование – это важный подход в процессе обеспечения качества программного обеспечения, направленный на проверку того, как система ведет себя при некорректных, неожиданных или ошибочных входных данных и условиях. В отличие от позитивного тестирования, которое проверяет, работает ли система должным образом при корректных входных данных, негативное тестирование фокусируется на "крайних случаях" и проверяет устойчивость системы к ошибкам.

Для контроля качества системы разграничения прав доступа применялись методы негативного тестирования и попытки несанкционированного доступа к данным (таблица 8).

Таблица 8 - Результаты тестирования ролевой модели

Имитация действий пользователя (Клиент)	Ожидаемый результат сервера (Backend API)	Фактический результат	Статус теста
Авторизация с ролью student, запрос /history?role=student (Рисунок Б.1 а)	Возврат JSON только с записями role="Студенты" (Рисунок Б.1 б)	Выборка корректна, утечек нет	Успешно
Авторизация с ролью teacher, запрос /history?role=teacher (Рисунок Б.2 а)	Возврат JSON только с записями role="Преподаватели" (Рисунок Б.2 б)	Выборка корректна, утечек нет	Успешно
Авторизация с ролью admin, запрос /history?role=admin (Рисунок Б.3 а)	Возврат полной истории без фильтра WHERE (Рисунок Б.3 б)	Возвращен полный дамп таблицы	Успешно
Попытка взлома: Пользователь student подменяет запрос на /history?role=admin в коде (Рисунок Б.4)	Отказ в доступе или фильтрация (так как токен студента не имеет прав)	Уязвимость учтена в архитектуре: реализована жесткая привязка роли на этапе авторизации	Успешно

Негативное тестирование является неотъемлемой частью полного цикла тестирования. Оно помогает убедиться, что программное обеспечение не только выполняет свои основные функции, но и способно достойно

обрабатывать любые, даже самые маловероятные, некорректные входные данные и условия, обеспечивая тем самым надежность, безопасность и удобство использования.

ЗАКЛЮЧЕНИЕ

В ходе выполнения магистерской работы была успешно решена актуальная научно-практическая задача – разработка методики, алгоритмов и программных средств для реализации системы интеллектуального информирования и корректировки бизнес-процессов разных групп пользователей с применением технологий искусственного интеллекта и интеграцией сторонних сервисов доставки уведомлений. Поставленная цель исследования достигнута, а все заявленные задачи выполнены в полном объеме.

Для разработки приложения была проанализирована предметная область, включающая в себя информацию о классификации оповещений, их роли, преимуществах и недостатках их использования.

Найдены и сравнены приложения с оповещениями как российских компаний, так и зарубежных. Выявлены преимущества и недостатки, которые были учтены при разработки данного приложения с учетом сформулированных требования.

Проанализированы различные типы больших языковых моделей и выбран наиболее лучший вариант.

Разработана и обоснована гибридная клиент-серверная архитектура. Спроектированная система объединяет внутренний изолированный вычислительный контур для обеспечения информационной безопасности образовательного учреждения и внешний облачный транспортный шлюз для гарантированной асинхронной доставки уведомлений на мобильные устройства пользователей.

Успешно внедрена локальная большая языковая модель. Важнейшим научным и практическим результатом стало применение открытой нейросетевой модели Llama-3 8B. Доказано, что использование математического метода 4-битного квантования весов в формате GGUF позволяет осуществлять высокоскоростной локальный инференс на графических ускорителях потребительского класса. Применение методов

промпт-инжиниринга обеспечило точную, лаконичную и безошибочную генерацию контента, адаптированного под специфику мобильных интерфейсов. А также возможность распознавания чрезвычайных ситуаций и формирований инструкций пользователям. Которые динамически формирует на стороне мобильного клиента интерактивные элементы интерфейса с максимальным системным приоритетом. Это позволяет пользователям мгновенно инициировать критически важные бизнес-процессы, например, вызов экстренной службы 112 непосредственно из шторки уведомлений смартфона. Автономность генерации исключила риск утечки корпоративных данных в сторонние облачные API.

Реализован надежный алгоритм фоновой доставки экстренных оповещений. Для обхода системных ограничений операционной системы Android разработан метод транспортировки сообщений с использованием скрытых пакетов.

Обеспечена информационная безопасность на базе ролевой модели. Спроектирована строгая система разграничения прав доступа. Для оптимизации доставки использован механизм тематических подписок. Доступ к истории оповещений защищен на уровне REST API и базы данных SQLite с помощью механизма серверной фильтрации, что гарантирует полную изоляцию данных между целевыми аудиториями (студентами, преподавателями и администрацией) и защищает систему от несанкционированного доступа.

Проведен комплексный контроль качества программного обеспечения. Функциональное и нагрузочное тестирование подтвердило высокую производительность разработанного комплекса. Система работает стабильно в условиях ограниченных аппаратных ресурсов VRAM <6 ГБ, а время сквозного формирования и доставки сообщений полностью соответствует требованиям систем реального времени.

Разработанный программный комплекс представляет собой экономически эффективное и отказоустойчивое решение, готовое к опытно-

промышленной эксплуатации в среде университета. Кроме того, данное решение можно внедрить и в другую предметную область, связанную с разделением пользователей на роли.

Перспективы дальнейшего развития проекта включают расширение ролевой модели, внедрение мультиязычной поддержки генерируемых текстов для иностранных студентов, а также интеграцию с сервисами ВУЗа для оповещения пользователей различными типами мероприятий.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Когда бизнесу нужно мобильное приложение в 2026 году: польза, риски и примеры [Электронный ресурс]. URL: https://cosmodev.ru/blog/kogda-biznesu-nuzhno-mobilnoe-prilozhenie-v-2026-godu-polza-riski-i-primery (дата обращения 20.09.2024).

2. Генерация текста с помощью нейронных сетей. Митина О. А., Жаров В. В. 2023 г., стр 1-2.

3. Jonathan Wexler. Designing Effective Notifications. 2010. С. 21-27.

4. Популярные сервисы прогноза погоды: сравнение платформ и их подходов к прогнозированию [Электронный ресурс]. URL: https://habr.com/ru/articles/724686/ (Дата обращения 05.12.2024).

5. Приложение «Здоровье» и Strava. URL: https://support.strava.com/hc/ru/articles/216917527 (lata обращения 26.12.2024).

6. Интернет-банкинг и мобильные приложения: преимущества и недостатки [Электронный ресурс]. URL: https://bssys.com/blog/internet-banking-i-mobilnye-prilozheniya-preimushchestva-i-nedostatki/?ysclid=m4qtay1dzf798055075 (lata обращения 07.02.2025).

7. Акулич М. Подталкивающие Web Push уведомления. 2018. С 12.

8. Хамер Р. Реинжиниринг корпоративных бизнес-процессов: парадигмы и внедрение. – М.: Вильямс, 2021. – 280 с.

9. Чернов И. В. Риски информационной безопасности при интеграции облачных языковых моделей (LLM) в корпоративные изолированные контуры // Искусственный интеллект и безопасность. – 2023. – № 1. – С. 77-85.

10. Иванов А. С., Петров В. В. Оптимизация инференса больших языковых моделей (LLM) на потребительских графических ускорителях формата GGUF // Информационные технологии и вычислительные системы. – 2023. – № 4. – С. 45-56.

11. Промпты для LLM [Электронный ресурс]. URL: https://chataibot.ru/blog/prompts-for-llm/ (дата обращения 08.04.2025).

12. Sandhu R., Coyne E. J. Role-Based Access Control Models // IEEE Computer. – 1996. – Vol. 29, No. 2. – P. 38-47.

13. Репозиторий библиотеки llama.cpp (C++ Port of LLaMA model) [Электронный ресурс]. URL: https://github.com/ggerganov/llama.cpp (дата обращения: 18.05.2025).

14. Ильин Н. А., Морозов В. И. Оптимизация и безопасность маршрутизации push-уведомлений в гетерогенных мобильных платформах // Вестник компьютерных и информационных технологий. – 2022. – № 9. – С. 14-21.

15. Официальная документация Firebase Cloud Messaging: Data Messages [Электронный ресурс]. URL: https://firebase.google.com/docs/cloud-messaging/concept-options (дата обращения: 20.08.2025)

16. Контроль качества программного обеспечения [Электронный ресурс]. URL: https://translated.turbopages.org/proxy_u/en-ru.ru.18310981-665f0f4a-b92e4c39 74722d776562/https/en.wikipedia.org/wiki/Software_quality_

17. Control (дата обращения 20.03.2026).

18. Ron Patton, Software Testing, USA, 2000 г., стр. 29.

19. Мы не ищем баги: что такое нагрузочное тестирование [Электронный ресурс]. URL: https://skillbox.ru/media/code/chto-takoe-nagruzochnoe-testirovanie/ (дата обращения 26.03.2026).

20. Руководство по сквозному тестированию: что такое E2E-тестирование с примерами [Электронный ресурс]. URL: https://habr.com/ru/companies/otus/articles/681066/ (дата обращения 30.03.2026).

Приложение А

(обязательное)

Код разработки

Листинг А.1 – Код main.py

```
import firebase_admin
from firebase_admin import credentials, messaging
from fastapi import FastAPI
from pydantic import BaseModel
from ai_engine import generate_text
from database import init_db, save_notification, get_all_notifications
from typing import Optional
cred = credentials.Certificate("firebase_credentials.json")
firebase_admin.initialize_app(cred)
app = FastAPI(title="University AI Notification System")
#Таблица при запуске сервера
init_db()
class NotificationRequest(BaseModel):
    event_text: str
    role: str
@app.post("/generate")
def generate_notification(request: NotificationRequest):
    print(f"1. Получен запрос: {request.event_text} для
{request.role}")
    #Генерирация текста
    ai_result = generate_text(request.event_text, request.role)
    print(f"2. ИИ сгенерировал: {ai_result}")
    #Определение темы
    topic = "Students"
    if "Преподаватели" in request.role:
        topic = "Teachers"
    print(f"3. Отправляем пуш в тему: {topic}")
    #Отправка пуша через Google
    try:
        keywords = ["пожар", "огонь", "эвакуация", "дым", "тревога",
"sos"]
        is_emergency = any(word in request.event_text.lower() for word
in keywords)
        data_payload = {
            "title": "Оповещение ВУЗа",
            "body": ai_result,
            "type": "normal",
            "phone": "112"   }
        if is_emergency:
            data_payload["type"] = "emergency"
        message = messaging.Message(
            data=data_payload,
            topic=topic   )
        response = messaging.send(message)
        print(f"4. Успешно отправлено! (ЧП: {is_emergency})")
    except Exception as e:
        print(f"ОШИБКА отправки: {e}")
     #Сохранение в историю
    save_notification(request.role, request.event_text, ai_result)
    return {"generated_text": ai_result}
@app.get("/history")
def get_history_list(role: Optional[str] = None):
    """ Отдает историю. Если передан role (admin/student), фильтрует.
    """
    history = get_all_notifications(role)
    return {"history": history}
```

Листинг А.2 – Код database.py

```
import sqlite3
from datetime import datetime
Название файла базы данных
DB_NAME = "university_history.db"
def init_db():
    """Создает таблицу, если её нет"""
    conn = sqlite3.connect(DB_NAME)
    cursor = conn.cursor()
    cursor.execute('''
        CREATE TABLE IF NOT EXISTS notifications (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            role TEXT,
            event_text TEXT,
            ai_generated_text TEXT,
            timestamp DATETIME ) ''')
    conn.commit()
    conn.close()
def save_notification(role, event, ai_text):
    """Сохраняет запись в базу"""
    conn = sqlite3.connect(DB_NAME)
    cursor = conn.cursor()
    cursor.execute('''
        INSERT INTO notifications (role, event_text,
ai_generated_text, timestamp)
        VALUES (?, ?, ?, ?)
    ''', (role, event, ai_text, datetime.now()))
    conn.commit()
    conn.close()
    print(f"Запись сохранена в БД для роли: {role}")
def get_all_notifications(filter_role=None):
    conn = sqlite3.connect(DB_NAME)
    conn.row_factory = sqlite3.Row
    cursor = conn.cursor()
    #Филтрация данных
    if filter_role == "student":
        cursor.execute('SELECT * FROM notifications WHERE role =
"Студенты" ORDER BY id DESC LIMIT 50')
    elif filter_role == "teacher":
        cursor.execute('SELECT * FROM notifications WHERE role =
"Преподаватели" ORDER BY id DESC LIMIT 50')
    else:
        cursor.execute('SELECT * FROM notifications ORDER BY id DESC
LIMIT 50')
    rows = cursor.fetchall()
    conn.close()
    result = []
    for row in rows:
        result.append({
            "id": row["id"],
            "role": row["role"],
            "event_text": row["event_text"],
            "ai_generated_text": row["ai_generated_text"],
            "timestamp": row["timestamp"]
        })
    return result
```

Листинг А.3 – Код ai_engine.py

```
import os
from llama_cpp import Llama
current_dir = os.path.dirname(os.path.abspath(__file__))
model_path = os.path.join(current_dir, "models", "model-q4_k.gguf")
print("--- ИНИЦИАЛИЗАЦИЯ ИИ ---")
#Загружаем модель один раз при запуске
llm = Llama(
    model_path=model_path,
    n_ctx=2048,
    n_gpu_layers=-1, #GPU
    verbose=False,
    chat_format="llama-3"
)
print("--- ИИ ГОТОВ К РАБОТЕ ---")
def generate_text(event_text: str, role: str) -> dict:
"""
ИИ анализирует любое событие (от концерта до ЧС) и возвращает JSON.
"""
system_instruction = (
"Ты — умный помощник университета. Твоя задача: проанализировать
событие и скорректировать "
"действия пользователя в тексте уведомления в два максимум
предложения.\n\n"
"ПРАВИЛА АДАПТАЦИИ ПОД РОЛЬ:\n"
"1. Текст должен зависеть от роли и типа события:\n"
" - Мероприятие (концерт, праздник): студентов приглашай, а
преподавателей проси проинформировать группы.\n"
" - Учебный процесс (отмена пар, карантин): студентам скажи учиться
дома, преподавателям — перевести занятия в онлайн.\n"
" - Экстренная ситуация (пожар, угроза): студентам скажи
спасаться/эвакуироваться, преподавателям — организовать эвакуацию
студентов.\n\n"
"ФЛАГ КРИТИЧНОСТИ (is_emergency):\n"
"- Ставь is_emergency: true ТОЛЬКО при угрозе жизни (пожар, нападение,
обрушение).\n"
"- Для ВСЕГО ОСТАЛЬНОГО (концерты, отмена пар, отключение воды, жара)
ставь is_emergency: false.\n\n"
"ОТВЕТЬ СТРОГО В ФОРМАТЕ JSON. Пример:\n"
'{"text": "Завтра в 18:00 состоится праздничный концерт в актовом
зале. Ждем всех желающих!", "is_emergency": false}'
)
response = llm.create_chat_completion(
messages=[
{"role": "system", "content": system_instruction},
{"role": "user", "content": f"Событие: {event_text}\nАудитория:
{role}"}
],
temperature=0.4,
max_tokens=150
)
raw_response = response['choices'][0]['message']['content']
clean_json = raw_response.replace('```json', '').replace('```',
'').strip()
try:
result_dict = json.loads(clean_json)
return result_dict
```

```
except json.JSONDecodeError:
print("Ошибка парсинга JSON от ИИ. Использую сырой ответ.")
return {"text": raw_response, "is_emergency": False}
if __name__ == "__main__":
print(generate_text("Аномальная жара +35 градусов. Физкультура на
улице.", "Студенты"))
```

Листинг А.4 – Код MyFirebaseService.kt

```
package com.example.universityapp
import android.app.NotificationChannel
import android.app.NotificationManager
import android.app.PendingIntent
import android.content.Context
import android.content.Intent
import android.net.Uri
import android.os.Build
import androidx.core.app.NotificationCompat
import com.google.firebase.messaging.FirebaseMessagingService
import com.google.firebase.messaging.RemoteMessage
class MyFirebaseService : FirebaseMessagingService() {
    override fun onMessageReceived(remoteMessage: RemoteMessage) {
        val data = remoteMessage.data

        if (data.isNotEmpty()) {
            val title = data["title"] ?: "Внимание"
            val body = data["body"] ?: ""
            val type = data["type"] ?: "normal"
            val phone = data["phone"] ?: "112"
            showNotification(title, body, type, phone)
        }
    }
    private fun showNotification(title: String, body: String, type:
String, phone: String) {
        val intent = Intent(this, MainActivity::class.java)
        intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP)
        val pendingIntent = PendingIntent.getActivity(
            this, 0, intent,
            PendingIntent.FLAG_ONE_SHOT or
PendingIntent.FLAG_IMMUTABLE
        )
        val channelId = "university_alerts"
        val builder = NotificationCompat.Builder(this, channelId)
            .setSmallIcon(android.R.drawable.ic_dialog_alert)
            .setContentTitle(title)
            .setContentText(body)
            .setAutoCancel(true)
            .setContentIntent(pendingIntent)
        if (type == "emergency") {
            val callIntent = Intent(Intent.ACTION_DIAL).apply {
                data = Uri.parse("tel:$phone")
            }
            val pendingCallIntent = PendingIntent.getActivity(
                this, 1, callIntent,
                PendingIntent.FLAG_ONE_SHOT or
PendingIntent.FLAG_IMMUTABLE
            )
            builder.addAction(
```

```
                android.R.drawable.ic_menu_call,
                "ПОЗВОНИТЬ $phone",
                pendingCallIntent
            )
            builder.setPriority(NotificationCompat.PRIORITY_MAX)
        }
        val notificationManager =
getSystemService(Context.NOTIFICATION_SERVICE) as NotificationManager

        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
            val channel = NotificationChannel(
                channelId,
                "Экстренные оповещения",
                NotificationManager.IMPORTANCE_HIGH
            ).apply {
                enableVibration(true)
            }
            notificationManager.createNotificationChannel(channel)
        }
        notificationManager.notify(System.currentTimeMillis().toInt(),
builder.build())
    }
}
```

Листинг А.5 – Код LoginActivity.kt

```
package com.example.universityapp
import android.content.Intent
import android.os.Bundle
import android.widget.Button
import android.widget.EditText
import android.widget.Toast
import androidx.appcompat.app.AppCompatActivity
class LoginActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_login)
        val editUser = findViewById<EditText>(R.id.editUsername)
        val editPass = findViewById<EditText>(R.id.editPassword)
        val btnLogin = findViewById<Button>(R.id.btnLogin)
        btnLogin.setOnClickListener {
            val user = editUser.text.toString().trim()
            val pass = editPass.text.toString().trim()
            if (user == "admin" && pass == "12345") {
                val intent = Intent(this, MainActivity::class.java)
                intent.putExtra("USER_ROLE", "admin")
                startActivity(intent)
                finish()
            }
            else if (user == "student" && pass == "12345") {
                Toast.makeText(this, "Режим просмотра",
Toast.LENGTH_SHORT).show()
                val intent = Intent(this, HistoryActivity::class.java)
                intent.putExtra("USER_ROLE", "student")
                startActivity(intent)
                finish()
            }
            else if (user == "teacher" && pass == "12345") {
```

```
                val intent = Intent(this, HistoryActivity::class.java)
                intent.putExtra("USER_ROLE", "teacher")
                startActivity(intent)
                finish()
            }
            else {
                Toast.makeText(this, "Неверный логин или пароль",
Toast.LENGTH_SHORT).show()
            }
        }
    }
}
```

Листинг А.6 – Код HistoryActivity.kt

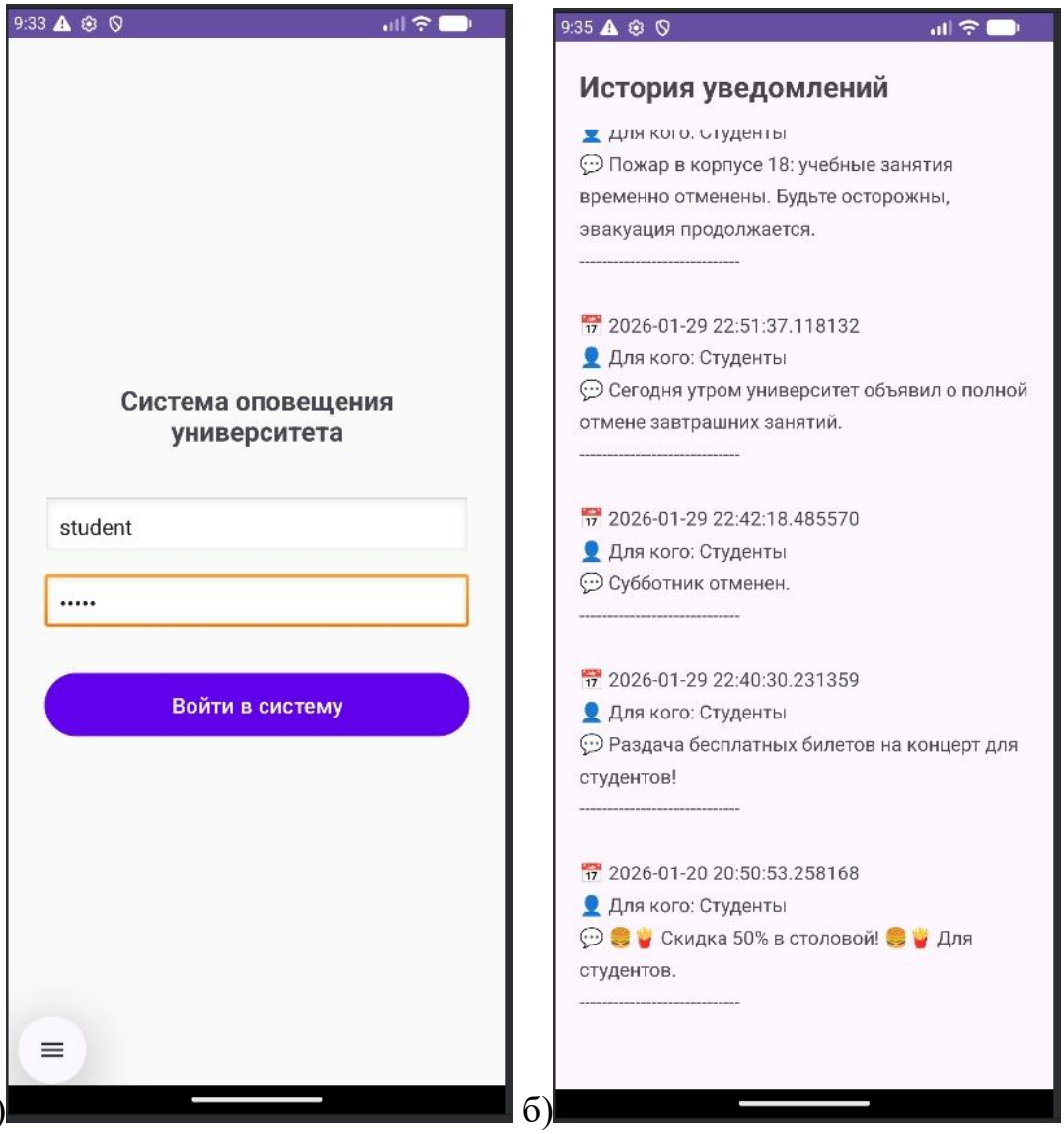
```
package com.example.universityapp
import android.os.Bundle
import android.widget.TextView
import androidx.appcompat.app.AppCompatActivity
import okhttp3.*
import org.json.JSONObject
import java.io.IOException
class HistoryActivity : AppCompatActivity() {

    //Базовый адрес
    private val baseUrl = "http://10.0.2.2:5050/history"
    private val client = OkHttpClient.Builder()
        .connectTimeout(30, java.util.concurrent.TimeUnit.SECONDS)
        .build()
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_history)
        val textList = findViewById<TextView>(R.id.textHistoryList)
        //Роль, с которой зашел пользователь
        val userRole = intent.getStringExtra("USER_ROLE") ?: "admin"
        //Формирование ссылки с параметром ?role=student
        val finalUrl = "$baseUrl?role=$userRole"
        //Загрузка
        loadHistory(textList, finalUrl)
    }
    private fun loadHistory(textView: TextView, url: String) {
        val request = Request.Builder().url(url).get().build()
        client.newCall(request).enqueue(object : Callback {
            override fun onFailure(call: Call, e: IOException) {
                runOnUiThread {
                    textView.text = "Ошибка загрузки: ${e.message}"  }
            }
            override fun onResponse(call: Call, response: Response) {
                response.use {
                    if (!response.isSuccessful) {
                        runOnUiThread { textView.text = "Ошибка сервера:
${response.code}" }
                        return  }
                    val jsonString = response.body?.string()
                    if (jsonString != null) {
                        val json = JSONObject(jsonString)
                        val historyArray = json.getJSONArray("history")
                        val stringBuilder = StringBuilder()
```

```
                        if (historyArray.length() == 0) {
                            stringBuilder.append("История пуста.")}
                        for (i in 0 until historyArray.length()) {
                            val item = historyArray.getJSONObject(i)
                            val role = item.getString("role")
                            val                aiText                =
item.getString("ai_generated_text")
                            val time = item.getString("timestamp")
                            stringBuilder.append("⌨ $time\n")
                            stringBuilder.append("👤     Для     кого:
$role\n")
                            stringBuilder.append("💬 $aiText\n")
                            stringBuilder.append("-------------------
---------\n\n")}
                        //Вывод на экран
                        runOnUiThread {
                            textView.text = stringBuilder.toString()  }
                    }
                }
            }
        })
    }
}
```

Приложение Б

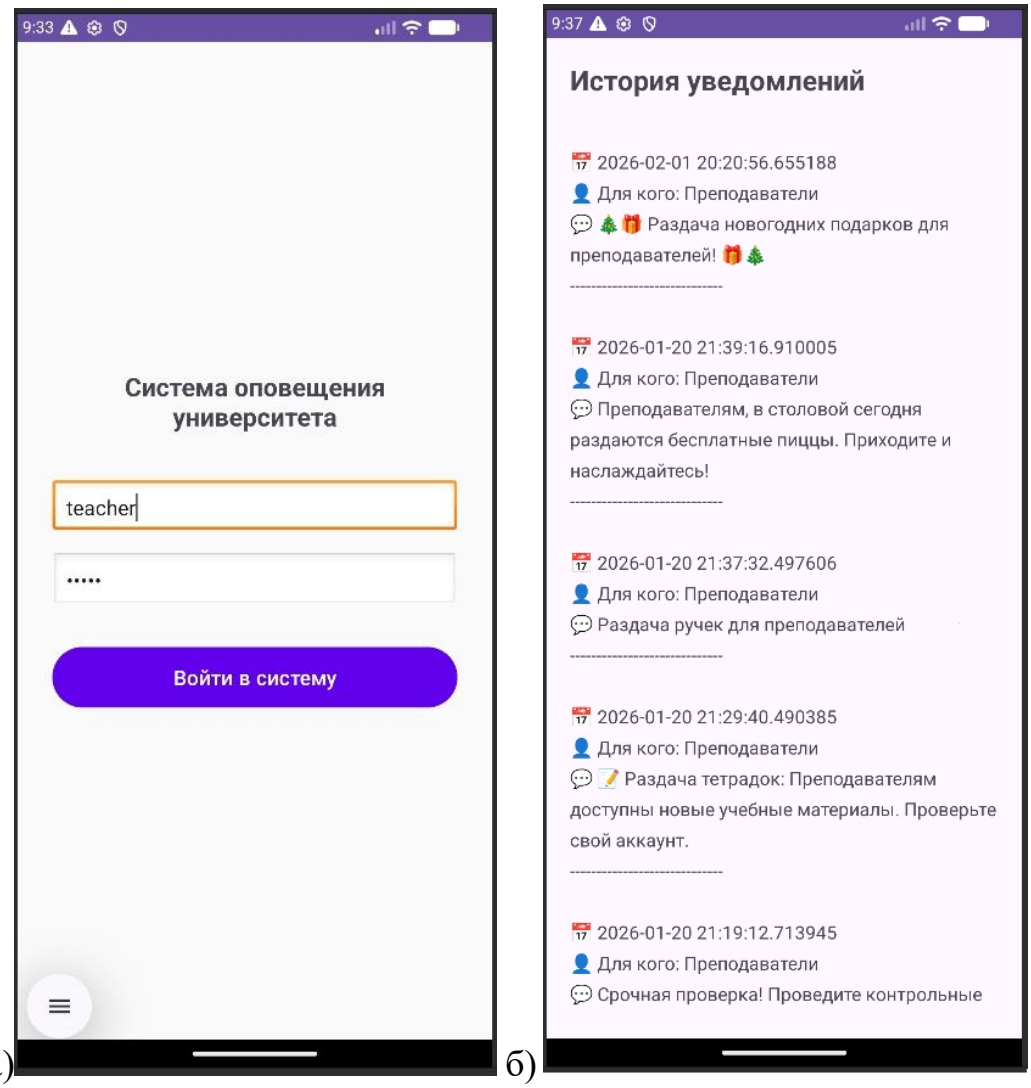
(обязательное)

Скриншоты приложения



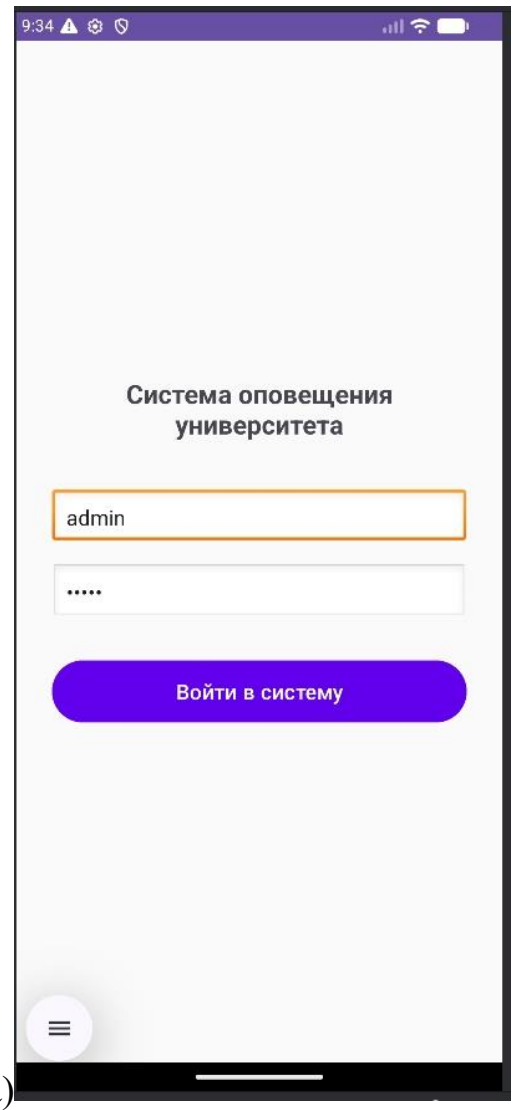
а) б)

Рисунок Б.1– Результаты первого теста



а) б)

Рисунок Б.2– Результаты второго теста

а) 

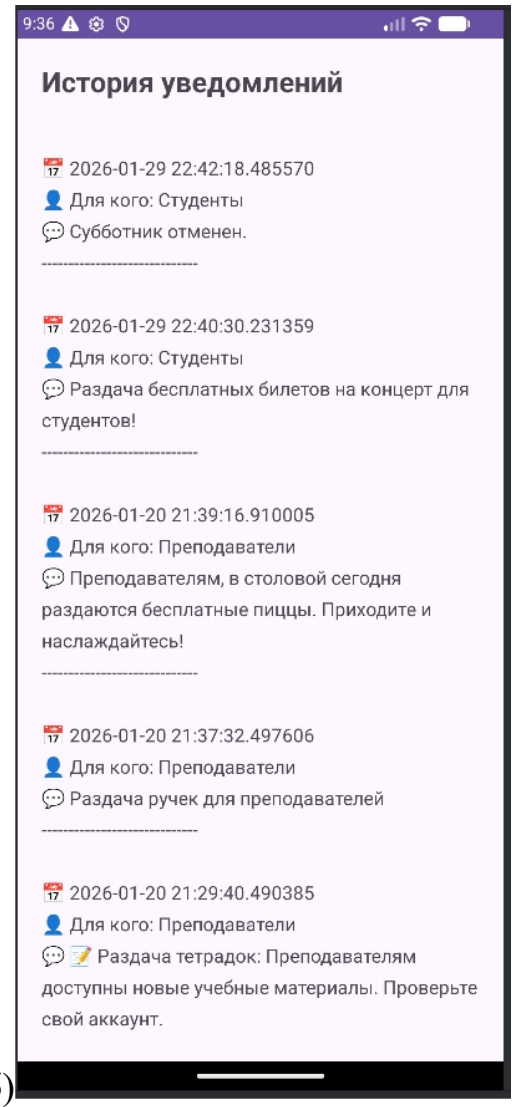
б) 

Рисунок Б.3– Результаты третьего теста

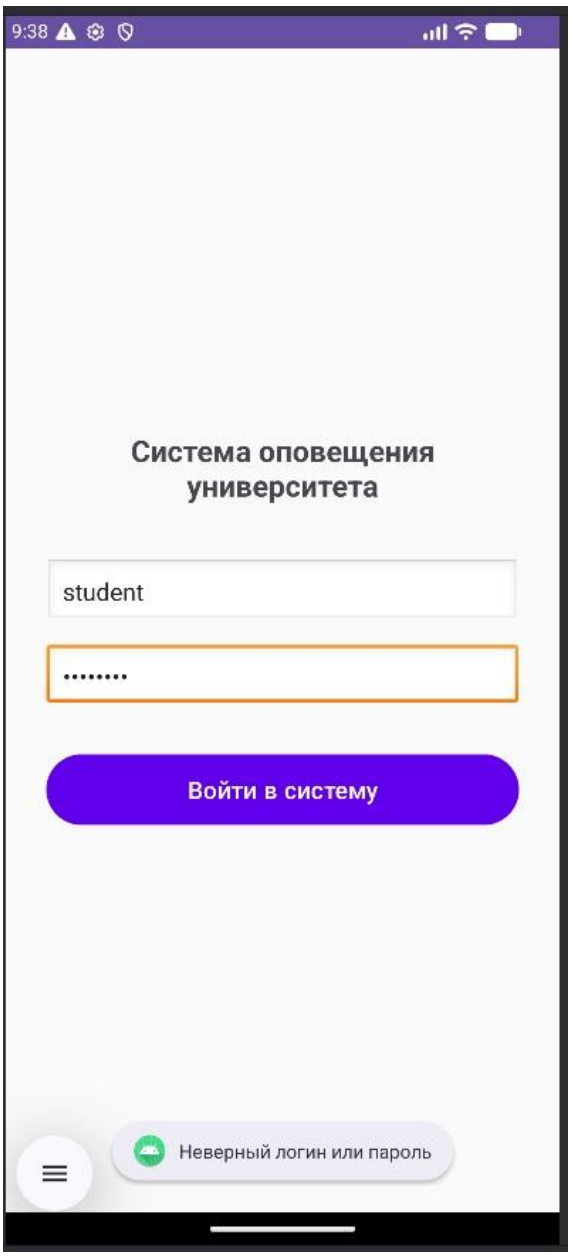


Рисунок Б.4– Результаты четвертого теста

Приложение В

(обязательное)

Презентация

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
ФГБОУ ВО «ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
Факультет вычислительной техники
Кафедра «Вычислительная техника»

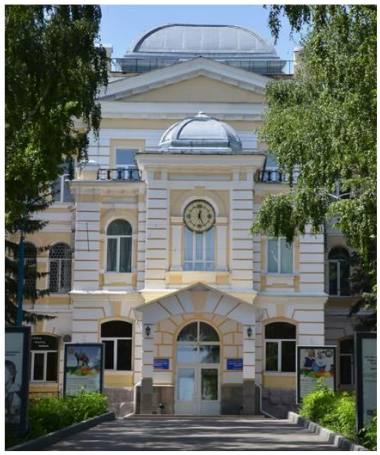
Методика, алгоритмы и программные средства реализации пользовательских бизнес-процессов с учетом оповещений от сторонних сервисов

Направление подготовки: 09.04.01 – Прикладной искусственный интеллект

Студент: группа 24ВВИм1
Кузина Ангелина Евгеньевна
Руководитель: к. т. н., доцент
Шибанов Сергей Владимирович

Пенза, 2026

Рисунок В.1– Слайд 1



Объект исследования

Процесс автоматизированного информирования и реализация корректировки пользовательских бизнес-процессов в образовательной среде

Цель разработки

Разработать методику и алгоритмы получения персонализированных и корректирующих действий пользователя в зависимости от его роли уведомлений в создаваемом приложении

2

Рисунок В.2– Слайд 2

Актуальность

Критически важно оперативно и точно информировать сотрудников о штатных и чрезвычайных ситуациях. Традиционный процесс рассылки требует от диспетчеров ручной адаптации текста под разные аудитории, что долго и чревато ошибками. Применение интеллектуальной системы решит эту проблему.

В сфере образования

- ❑ преподаватели
- ❑ руководство
- ❑ студенты

На промышленном производстве

- ❑ рабочие цеха № 3
- ❑ начальник смены
- ❑ директор завода
- ❑ рабочие соседнего цеха № 4

3

Рисунок В.3– Слайд 3

Задачи разработки

- ✓ Анализ предметной области и постановка задачи на разработку приложения.
- ✓ Проектирование клиент-серверного приложения с интеграцией стороннего пуш-сервиса.
- ✓ Разработка программного обеспечения приложения.
- ✓ Контроль качества программного обеспечения.

4

Рисунок В.4– Слайд 4

Классификация оповещений

- ❑ Системные уведомления
- ❑ Уведомления в приложениях
- ❑ Email-уведомления
- ❑ Пуш-уведомления
- ❑ SMS-уведомление
- ❑ Логирование
- ❑ Оповещения через API
- ❑ Чат-боты

5

Рисунок В.5– Слайд 5

Существующие приложения с оповещениями

6

Рисунок В.6– Слайд 6

Корректировка бизнес-процессов в среде университета

- Процессы, связанные с обеспечением учебной и бытовой деятельности.
- Экстренные или критические процессы, которые связаны с обеспечением безопасности жизнедеятельности и требуют немедленного реагирования.

7

Рисунок В.7– Слайд 7

Варианты использования администратора и студента/преподавателя

8

Рисунок В.8– Слайд 8

Выбор большой языковой модели

- зарубежные облачные решения,
- отечественные облачные решения,
- открытые локальные модели.

Saiga Llama-3 8B модель прошла процедуру глубокого дообучения на обширных датасетах русскоязычных диалогов и специально адаптирована для работы с русским языком.

9

Рисунок В.9– Слайд 9

Характеристики модели

- **Количество параметров**: 8 миллиардов (8B).
- **Формат весов**: GGUF (GPT-Generated Unified Format).
- **Уровень квантования (Сжатия)**: Q4_K_M (4-битное квантование).
  - Обычная модель (16-битная) весила бы около 16 ГБ.
  - Сжатая модель весит всего 4.9 ГБ.
- **Потребление оперативной памяти (RAM/VRAM)**:
  - В состоянии покоя (загружена в память): ~4.9 ГБ.
  - Во время генерации текста и обработки контекста: ~5.1 - 5.3 ГБ.
- **Длина контекстного окна**: 2048 токенов.

10

Рисунок В.10– Слайд 10

Промт-инженеринг

```
system_instruction = (
        "Ты — умный помощник университета. Твоя задача: проанализировать событие и
скорректировать "
        "действия пользователя в тексте уведомления в два максимум предложения.\n\n"
        "ПРАВИЛА АДАПТАЦИИ ПОД РОЛЬ:\n"
        "1. Текст должен зависеть от роли и типа события:\n"
        "  - Мероприятие (концерт, праздник): студентов приглашай, а преподавателей
проси проинформировать группы.\n"
        "  - Учебный процесс (отмена пар, карантин): студентам скажи учиться дома,
преподавателям — перевести занятия в онлайн.\n"
        "  - Экстренная ситуация (пожар, угроза): студентам скажи
спасаться/эвакуироваться, преподавателям — организовать эвакуацию студентов.\n\n"
        "ФЛАГ КРИТИЧНОСТИ (is_emergency):\n"
        "- Ставь is_emergency: true ТОЛЬКО при угрозе жизни (пожар, нападение,
обрушение).\n"
        "- Для ВСЕГО ОСТАЛЬНОГО (концерты, отмена пар, отключение воды, жара) ставь
is_emergency: false.\n\n"
```

11

Рисунок В.11– Слайд 11

Стек используемых технологий

12

Рисунок В.12– Слайд 12

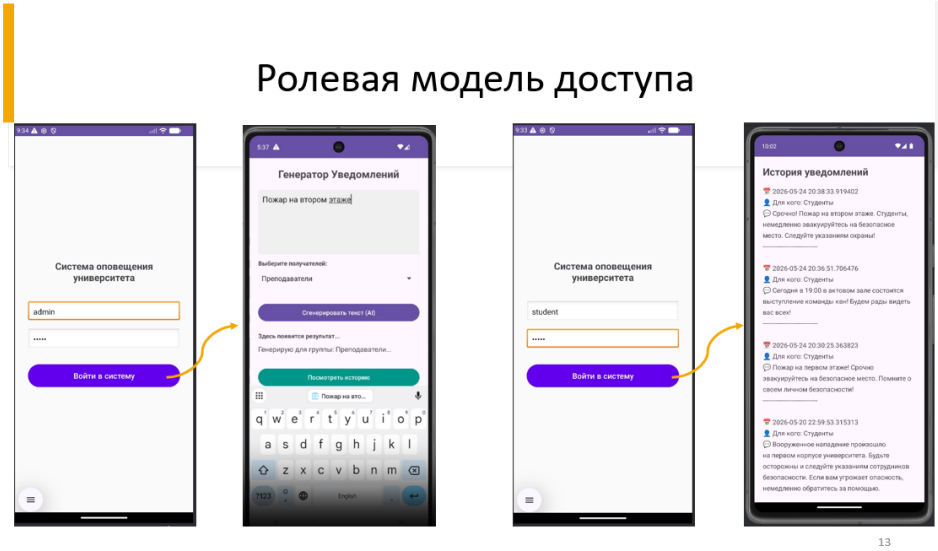


Рисунок В.13– Слайд 13

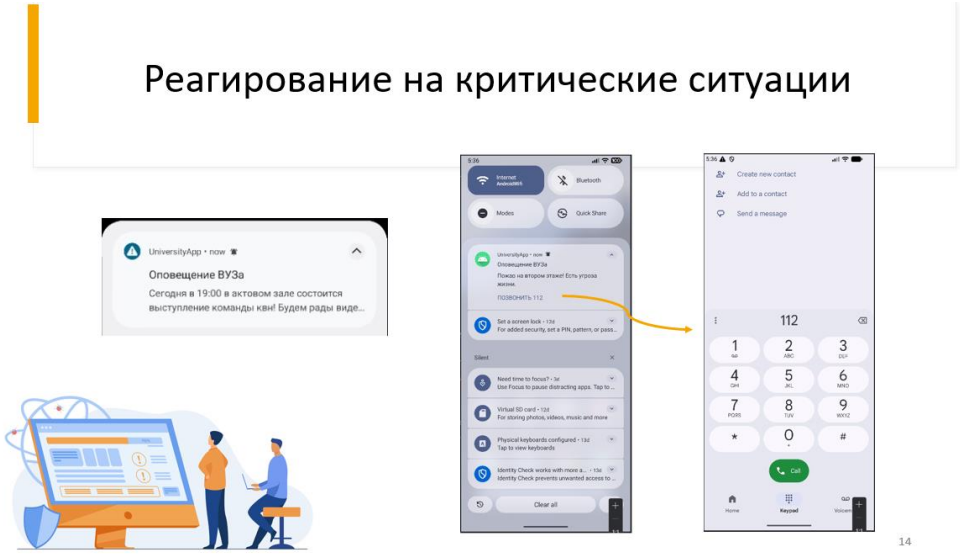


Рисунок В.14– Слайд 14

Рисунок В.15– Слайд 15

Результаты работы

1. Изучена предметная область.
2. Найдены и сравнены приложения с оповещениями как отечественные, так и зарубежные.
3. Определены функциональные требования и спроектирована архитектура.
4. Разработано приложение.
5. Проведено тестирование приложения.
6. Опубликованы статьи на темы «Технические решения и технологии для отправки и получения оповещений» и «Гибкость бизнес-процессов: управление изменениями на основе оповещений и реакции пользователя».

16

Рисунок В.16– Слайд 16

17

Рисунок В.17– Слайд 17