Министерство образования Ярославской области

ГПОУ ЯО Ярославский колледж управления  
и профессиональных технологий

**Курсовой проект**

МДК.01.01 Разработка программных модулей

Разработка мобильного приложения для коммуникации с большими языковыми моделями

|  |  |
| --- | --- |
|  | Выполнил студент группы ИП1-31  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Лопаков С.Б. |
| Проект защищён  с оценкой \_\_\_\_\_\_\_\_\_\_\_\_  «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 2024 г. | Руководитель работы  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Осипова А.А. |

г. Ярославль, 2024

СОДЕРЖАНИЕ

[ГЛОССАРИЙ 4](#_Toc170704055)

[Введение 5](#_Toc170704056)

[1 Описание предметной области 8](#_Toc170704057)

[1.1 Серверная часть 8](#_Toc170704058)

[1.2 Клиентская часть 11](#_Toc170704059)

[1.3 Требования к функционалу 13](#_Toc170704060)

[2 Проектирование информационной системы 15](#_Toc170704061)

[2.1 Определение требований и спецификаций на разработку мобильного приложения 15](#_Toc170704062)

[2.2 Требования по реализации 16](#_Toc170704063)

[2.3 Список действующих лиц 16](#_Toc170704064)

[2.4 Функциональная модель 17](#_Toc170704065)

[2.4.1 Контекстная диаграмма IDEF0 17](#_Toc170704066)

[2.4.2 Диаграмма декомпозиции IDEF0 17](#_Toc170704067)

[3 Реализация информационной системы 18](#_Toc170704068)

[3.1 Разработка базы данных 18](#_Toc170704069)

[3.2 Реализация базы данных 20](#_Toc170704070)

[3.3 Реализация клиентской оболочки 21](#_Toc170704071)

[ЗАКЛЮЧЕНИЕ 24](#_Toc170704072)

[ИНФОРМАЦИОННЫЕ РЕСУРСЫ 26](#_Toc170704073)

[ПРИЛОЖЕНИЯ 29](#_Toc170704074)

[ПРИЛОЖЕНИЕ А 29](#_Toc170704075)

[Введение 29](#_Toc170704076)

[Основание для разработки 29](#_Toc170704077)

[1 Назначение разработки 29](#_Toc170704078)

[2 Требования к информационной системе 30](#_Toc170704079)

[ПРИЛОЖЕНИЕ Б 32](#_Toc170704080)

[ПРИЛОЖЕНИЕ В 34](#_Toc170704081)

[ПРИЛОЖЕНИЕ Г 37](#_Toc170704082)

[ПРИЛОЖЕНИЕ Д 38](#_Toc170704083)

# ГЛОССАРИЙ

Языковая модель *(LM)* – это модель человеческого языка, применяющаяся для генерации осмысленных выходных сигналов на основе введённого текста. В дальнейшем будут рассмотрены языковые модели, выдающие продолжение для введённой цепочки токенов.

Большая языковая модель *(LLM)* – это нейросетевая языковая модель, содержащая более 1 млрд. параметров (весовых коэффициентов).

Открытая LLM – это большая языковая модель, веса и архитектура которой находятся в открытом доступе и могут быть использованы локально. Зачастую также оказывается известным или общедоступным состав обучающей выборки.

Веса модели – это весовые коэффициенты архитектуры модели, которые получены в результаты обучения модели на определённой обучающей выборке.

Архитектура модели – это граф связей и операций, по которому проходят входные данные в ходе обработки моделью для получения выходных данных.

Обучающая выборка – это набор токенов, который модель должна научиться воспроизводить в ходе обучения.

Токен – это базовая единица, над которой оперирует языковая модель. Может представлять из себя несколько слов, одно слово, фрагмент слова, отдельный символ или непечатный управляющий сигнал.

Квантование модели – особое кодирование весов модели с минимальными потерями в целях экономии расходуемой памяти и ускорения вычислений.

# Введение

В современном мире развитие технологий искусственного интеллекта и особенно больших языковых моделей становится всё более актуальным. Применение таких моделей охватывает множество областей, начиная от автоматического перевода и заканчивая созданием интеллектуальных ассистентов. Однако существующие решения зачастую ограничены либо проприетарными системами с цензурированными обучающими выборками и жёсткой модерацией, либо недостаточной мобильностью и гибкостью использования. В этом контексте разработка мобильного приложения для коммуникации с LLM, развёрнутыми локально на пользовательских устройствах, представляется важной и актуальной задачей.

Целью данного курсового проекта является разработка мобильного приложения для операционной системы Android, которое позволит пользователям подключаться к локально развёрнутым серверам больших языковых моделей на их компьютерах для взаимодействия с этими моделями.

Для достижения этой цели в рамках проекта поставлены следующие задачи:

* Определение ключевых стандартов предметной области для последующей работы с ними.
* Разработка структуры локальной базы данных для хранения информации о профилях и диалогах с LLM.
* Дизайн пользовательского интерфейса, включая разработку экранов приложения.
* Реализация стандарта API, определённого в первой задаче.
* Реализация логики взаимодействия с сервером LLM посредством реализованного стандарта API.

Практическая значимость проекта обусловлена фактором ограниченности возможностей проприетарных решений в области чат-приложений. В качестве выбранной нами альтернативы выступают открытые LLM, которые можно запускать локально на своём компьютере, что обеспечивает полную приватность и отсутствие модерации, позволяя обсуждать любые темы без ограничений и произвольно настраивать языковую модель.

Основная проблема, которую решает данный проект, заключается в отсутствии удобного мобильного пользовательского интерфейса для взаимодействия с локально развёрнутыми серверами LLM. Существующие решения либо ограничены настольными версиями, либо предоставляют веб-интерфейсы с ограниченными возможностями. Наш проект предлагает мобильное решение, позволяющее пользователю общаться с произвольными LLM из любой точки мира, обеспечивая высокую скорость работы за счёт подключения к серверу на компьютере.

Анализ рынка показывает, что большинство проприетарных решений предоставляют мобильную адаптацию интерфейса, в то время как открытые решения, такие, как «text-generation-webui», чаще ограничены настольными версиями или веб-интерфейсами, не всегда поддерживающими мультимодальные функции и гибкость настройки. Среди мобильных решений наиболее близким аналогом является проект «mllm», который, однако, запускает LLM прямо на мобильном устройстве, что даёт низкую производительность и ограниченную применимость.

Курсовой проект состоит из семи разделов: введения, трёх глав основного содержания, заключения, списка информационных ресурсов и приложений.

Во введении обоснована актуальность работы, сформулированы тема, цель, задачи, определены объекты и предмет исследования.

В первой главе проанализирована предметная область исследования, дано её описание на примере существующих решений, изучены и описаны условия использования, сформулированы требования к функциональным характеристикам проектируемого приложения.

Во второй главе обоснован выбор технологий разработки, описаны используемые модели и диаграммы, построены соответствующие диаграммы для каждого подхода.

В третьей главе описывается реализация приложения и результаты тестирования.

В заключении подведены итоги проектирования, проанализированы полученные результаты, обозначены вопросы модернизации проекта и даны рекомендации по дальнейшему использованию материалов курсового проекта.

Список информационных ресурсов включает ссылки на основные и дополнительные литературные источники, стандарты и интернет-ресурсы.

В приложениях приведены материалы, не вошедшие в основное содержание курсового проекта, но являющиеся его неотъемлемой частью.

# 1 Описание предметной области

## 1.1 Серверная часть

Современные технологии для запуска больших языковых моделей предлагают несколько ключевых решений, каждое из которых имеет свои особенности и преимущества. Рассмотрим наиболее распространённые бэкенды:

1. *Transformers:*

Описание: это библиотека для Python, разработанная компанией Hugging Face, которая поддерживает широкий круг архитектур языковых моделей и видов оборудования.

*Преимущества:*

* Обширная поддержка различных архитектур LLM;
* Наличие широкой и подробной документации;
* Активная поддержка сообщества.

*Ограничения:*

* Ограниченность возможностей квантования по сравнению со специализированными решениями.

1. *ExLlama2:*

Описание: это также библиотека для Python, предоставляющая гибкий формат квантования «EXL2», который позволяет использовать произвольную битность для различных сегментов весов модели.

*Преимущества:*

* Наличие гибких возможностей квантования для достижения наибольшего качества работы квантованных моделей;
* Реализация оптимизированных ядер CUDA для достижения наибольшей скорости исполнения на графических ускорителях.

*Ограничения:*

* Поддержка работы только на графических ускорителях, что может быть серьёзным ограничивающим фактором;
* Отсутствие поддержки мультимодальных функций.

1. *llama.cpp:*

Описание: самостоятельное решение, написанное на C++, которое поддерживает широкий спектр архитектур языковых моделей и предоставляет формат квантования «GGUF».

*Преимущества:*

* Поддержка исполнения как на процессорах (ЦП), так и на графических картах;
* Наличие встроенного сервера, что упрощает развёртывание и использование.

*Ограничения:*

* Ограниченность скорости исполнения на ЦП
* Отставание по качеству квантования от формата «EXL2».

Использование решений, которые не содержат встроенной серверной части, требует либо самостоятельной реализации сервера, либо интеграции с готовыми серверными решениями. Это может существенно увеличить объём работ и сложность проекта, поэтому использование встроенного сервера в llama.cpp является предпочтительным вариантом для нашего проекта.

Если планируется подключаться к серверу в пределах локальной сети, то достаточно будет иметь IP-адрес компьютера, на котором развёрнут сервер, и его рабочий порт. Если же планируется подключаться к серверу не внутри локальной сети, то потребуется обеспечить возможность подключения извне. Для этого могут быть рассмотрены следующие методы:

* *Проброс портов на роутере:* роутер может быть настроен таким образом, чтобы перенаправлять запросы по рабочему порту с внешнего IP-адреса на компьютер с сервером в локальной сети, если это возможно;
* *Туннелирование:* использование услуг провайдеров туннелирования, таких как Cloudflare, для обеспечения соединения с сервером через интернет. При этом дополнительная безопасность обеспечивается применением протокола HTTPS.

Для взаимодействия с сервером используются стандарты API. Наиболее распространёнными являются:

* *OpenAI API:* спецификация, разработанная компанией OpenAI, широко используемая и хорошо документированная.
* *llama.cpp API:* спецификация API, реализованная во встроенном в llama.cpp сервере.

Таким образом, для удовлетворения потребностей нашего проекта, серверная часть должна поддерживать следующие функции:

* Обеспечение работы на разнообразном оборудовании, таком как ЦП и различные графические карты;
* Поддержка различных архитектур языковых моделей;
* Поддержка мультимодальных функций, т.е. возможность работы с текстовыми и визуальными данными, если это поддерживается запущенной LLM.
* Поддержка различных параметров генерации;
* Корректная поддержка стандартизированного API;

## 1.2 Клиентская часть

Клиентская часть чатового приложения должна предоставлять пользователю удобный и функциональный интерфейс для взаимодействия с сервером, на котором запущена большая языковая модель. Важно обеспечить широкий набор функций, включая мультимодальные возможности и сохранение отдельных конфигураций для каждого чата. Рассмотрим предметную область на примере двух наиболее близких по характеристикам существующих проектов: text-generation-webui и mllm.

1. *Text-Generation-WebUI:*

*Описание:* это веб-решение, представленное проектом text-generation-webui, которое предлагает адаптивный интерфейс для пользователей мобильных устройств.

*Преимущества:*

* Наличие удобного адаптивного интерфейса, подходящего для работы на мобильном устройстве;
* Возможность тонкой настройки параметров генерации текста;
* Возможность замены бэкенда LLM на лету.

*Ограничения:*

* Отсутствие поддержки мультимодальных функций для большинства бэкендов;
* Невозможность подключения к сторонним серверам, предоставляющим только доступ по API, т.е. не работающим на основе этого проекта;
* Невозможность задания отдельных конфигураций для каждого чата (при этом имеется возможность задавать профили конфигураций).

1. *MLLM (Mobile Large Language Model):*

*Описание:* это веб-решение, представленное проектом text-generation-webui, которое предлагает адаптивный интерфейс для пользователей мобильных устройств.

*Преимущества:*

* Наличие удобного адаптивного интерфейса, подходящего для работы на мобильном устройстве;
* Возможность тонкой настройки параметров генерации текста;
* Возможность замены бэкенда LLM на лету.

*Ограничения:*

* Отсутствие поддержки мультимодальных функций для большинства бэкендов;
* Невозможность подключения к сторонним серверам;
* Невозможность задания отдельных конфигураций для каждого чата (при этом имеется возможность задавать профили конфигураций).

Таким образом, для удовлетворения потребностей нашего проекта, клиентская часть должна удовлетворять следующим критериям:

* Поддержка работы на разных мобильных устройствах под управлением ОС Android;
* Поддержка мультимодальных функций путём загрузки изображений из галереи или создания снимков с помощью камеры;
* Сохранение конфигураций для каждого чата и пользовательского профиля;
* Поддержка отправки на сервер некоторого множества параметров генерации, достаточного для тонкой настройки поведения модели;

## 1.3 Требования к функционалу

Клиентское приложение для взаимодействия с сервером, на котором запущена большая языковая модель (LLM), должно обеспечивать выполнение следующих функций:

1. *Регистрация и авторизация пользователей:*

* Возможность создания профиля с уникальным логином и паролем;
* Возможность авторизации в существующий профиль для доступа к персонализированным настройкам и чатам.

1. *Управление чатами:*

* Возможность создания новых чатов с указанием названий;
* Возможность отображения списка существующих чатов с указанием времени последнего доступа;
* Возможность удаления чатов.

1. *Настройки профиля и чатов:*

* Сохранение глобальных настроек профиля, таких как предпочтительные параметры генерации текста, адрес сервера и параметры диалога;
* Сохранение уникальных конфигураций для каждого чата, включая параметры генерации и другие настройки;
* Возможность редактирования и обновления настроек как на уровне профиля, так и на уровне отдельных чатов.

1. *Мультимодальные функции:*

* Поддержка текстовых сообщений;
* Поддержка отправки изображений, загруженных из галереи устройства или созданных с помощью камеры;
* Возможность отображения мультимодального контента в интерфейсе чата.

1. *Интерфейс взаимодействия с сервером:*

* Поддержка подключения к серверу, реализующему выбранный стандарт API;
* Возможность отправки запросов к серверу и получения ответов от модели;
* Возможность настройки параметров генерации текста, таких как длина ответа, степень креативности, контекст диалога и т.д.

1. *Адаптивный интерфейс:*

* Поддержка работы на различных устройствах под управлением ОС Android;
* Адаптивный дизайн интерфейса для удобного использования на экранах различных размеров;
* Дизайн приложения с учётом многоязычности и подстраивания под системную цветовую тему (светлую или тёмную).

1. *История сообщений:*

* Сохранение истории сообщений для каждого чата;
* Отображение временных меток и учёт роли для каждого сообщения.

1. *Производительность и оптимизация:*

* Оптимизация производительности для обеспечения быстрой и плавной работы приложения.

Эти требования обеспечивают полноценную функциональность клиентского приложения, необходимую для удобного и эффективного взаимодействия с большими языковыми моделями, запущенными на сервере.

# 2 Проектирование информационной системы

При проектировании информационной системы необходимо провести анализ целей этой системы и выявить требования к ней отдельных пользователей. Информация для построения модели информационной системы берется на основе всестороннего анализа предметной области и процессов, использующих данные сущности и связи между ними.

## 2.1 Определение требований и спецификаций на разработку мобильного приложения

К требованиям для создания нашего мобильного клиентского приложения относятся:

* Возможность пользователя создавать и настраивать профили для взаимодействия с сервером LLM;
* Возможность пользователя создавать чаты и управлять ими;
* Возможность сохранять конфигурации для чатов;
* Поддержка мультимодальных функций: загрузка изображений из галереи или создание снимков с помощью камеры;
* Интерактивный интерфейс для отправки сообщений и получения ответов от LLM;
* Сохранение и управление историей сообщений.

Специфическими моментами являются:

* Возможность редактировать отправленные сообщения как пользователя, так и ИИ-ассистента;
* Необходимость интеграции с серверной частью через стандартный API, обеспечивающий взаимодействие с моделью LLM;
* Локальное хранение БД на устройстве пользователя.

## 2.2 Требования по реализации

Мобильное приложение должно корректно работать на устройствах под управлением ОС Android.

Информационная система должна обеспечивать:

* Корректное отображение интерфейса на различных разрешениях экранов мобильных устройств;
* Надежность и сохранение работоспособности при возникновении внештатных ситуаций;
* Поддержку работы в автономном режиме с последующей синхронизацией данных при восстановлении соединения.

Для обеспечения безопасности данных:

* Безопасные методы аутентификации и передачи данных, такие как протокол HTTPS.

Приложение должно быть разработано с использованием следующего технологического стека:

* *Android Studio:* IDE для разработки приложения;
* *Jetpack Compose:* фреймворк для создания пользовательского интерфейса;
* *Retrofit:* библиотека для выполнения сетевых запросов;
* *Room:* библиотека для работы с локальной базой данных;
* *Accompanist:* набор расширений для Jetpack Compose.

## 2.3 Список действующих лиц

* *Пользователь:* создает профили, управляет чатами и отправляет сообщения;
* *Поставщик LLM:* обеспечивает работу сервера LLM и доступ к нему.

## 2.4 Функциональная модель

Функциональная модель информационной системы показывает функциональную структуру приложения, производимые им действия и связи между этими действиями. Модель состоит из диаграмм, представляющих основные функции системы и их взаимосвязи.

### 2.4.1 Контекстная диаграмма IDEF0

Контекстная диаграмма представляет всю систему в виде одного блока и интерфейсов с внешними функциями. Были определены следующие категории объектов диаграммы:

* + *Входные данные:* учётные данные (логин и пароль); сообщения (текст и изображения); настройки генерации (параметры и шаблоны).
  + *Выходные данные:* обновлённые сообщения (с ответами LLM).
  + *Стандарты и нормы:* спецификация API.
  + *Ресурсы и инструменты:* пользователь; мобильное устройство; сервер LLM; интерфейс приложения; база данных.

Контекстная диаграмма представлена в приложении Б  
(изображение Б1).

### 2.4.2 Диаграмма декомпозиции IDEF0

Диаграмма декомпозиции разбивает систему на более мелкие функции, детально показывая их взаимодействия и связи.

Диаграмма декомпозиции представлена в приложении Б  
(изображение Б2).

# 3 Реализация информационной системы

## 3.1 Разработка базы данных

Для проектируемой информационной системы была разработана модель данных, включающая несколько сущностей: profiles, chats, messages, settings, chat\_settings, и global\_settings. Каждая из этих сущностей имеет свои атрибуты.

1. *profiles:*

* id (PK): Уникальный идентификатор профиля.
* login: Логин пользователя.
* password: Хешированный пароль пользователя.

*Описание:* Таблица profiles хранит информацию о пользователях системы, включая их логины и пароли. Каждый профиль имеет уникальный идентификатор, который используется для связи с другими сущностями в базе данных.

1. *chats:*

* id (PK): Уникальный идентификатор чата.
* profile\_id (FK): Идентификатор профиля пользователя, создавшего чат.
* title: Название чата.
* last\_access: Дата и время последнего доступа к чату.

*Описание:* Таблица chats содержит информацию о созданных чатах, включая их уникальные идентификаторы, названия, а также идентификаторы пользователей, создавших эти чаты. Также хранится время последнего доступа к чату для упрощения управления чатами.

1. *messages:*

* id (PK): Уникальный идентификатор сообщения.
* chat\_id (FK): Идентификатор чата, к которому относится сообщение.
* role: Роль отправителя сообщения (“assistant” или “user”).
* content: Содержимое сообщения.
* timestamp: Дата и время отправки сообщения.
* image: изображение, прикрепленного к сообщению (если есть); закодировано в Base64.

*Описание:* Таблица messages хранит сообщения, отправленные в рамках чатов. Каждое сообщение имеет уникальный идентификатор, содержимое, временную метку, идентификатор чата, к которому оно относится, а также роль отправителя.

1. *settings:*

* id (PK): Уникальный идентификатор настройки.
* name: Название настройки.
* default: Значение по умолчанию для данной настройки.

*Описание:* Таблица settings хранит общие настройки системы, которые могут быть использованы как на уровне отдельных чатов, так и на уровне профилей пользователей. Каждая настройка имеет уникальный идентификатор, название и значение по умолчанию.

1. *chat\_settings:*

* chat\_id (PK, FK): Идентификатор чата, к которому относится настройка.
* setting\_id (PK, FK): Идентификатор настройки.
* value: Значение настройки для данного чата.

*Описание:*Таблица chat\_settings хранит значения настроек для конкретных чатов. Это позволяет настраивать параметры отдельных чатов независимо от глобальных настроек системы.

1. *global\_settings:*

* profile\_id (PK, FK): Идентификатор профиля пользователя.
* setting\_id (PK, FK): Идентификатор настройки.
* value: Значение настройки для данного профиля.

*Описание:* Таблица global\_settings хранит значения настроек для конкретных профилей пользователей. Это позволяет пользователям настраивать параметры своей работы в системе независимо от общих настроек.

## 3.2 Реализация базы данных

Для реализации рабочей базы данных используется библиотека Room для создания локальных баз данных SQLite и работы с ними в рамках объектно-ориентированной парадигмы.

Логическая схема базы данных представлена в приложении В  
(изображение В1).

Реализация БД в виде кода на Kotlin с применением Room приведена в приложении В (листинг В2).

## 3.3 Реализация клиентской оболочки

Для разработки мобильного приложения применяется IDE Android Studio. Интерфейс разработан с помощью фреймворка Jetpack Compose. Из крупных библиотек используются:

* Retrofit: для сетевых запросов.
* Room: для локальной базы данных.
* Accompanist: для расширений Jetpack Compose.

Для реализации требуемого функционала были спроектированы и написаны следующие экраны:

1. *ProfilesScreen:*

*Описание:* экран списка профилей пользователей.

*Функционал:*

* Отображение списка существующих профилей;
* Возможность выбора профиля для входа в систему;
* Переход к созданию нового профиля.

1. *NewProfileScreen:*

*Описание:* экран создания нового профиля.

*Функционал:*

* Ввод данных нового профиля (логин и пароль);
* Сохранение нового профиля в базу данных;
* Переход к списку чатов нового профиля после его успешного создания.

1. *ChatsScreen:*

*Описание:* экран списка чатов.

*Функционал:*

* Отображение списка чатов для текущего профиля;
* Возможность создания нового чата;
* Возможность удаления чатов;
* Переход к экрану конкретного чата;
* Переход к выбору подлежащей изменению группы настроек для текущего профиля.

1. *SettingsChooseScreen:*

*Описание:* экран выбора настроек.

*Функционал:*

* Предоставление пользователю возможности выбрать, какие настройки он хочет изменить (настройки профиля или глобальные настройки чатов);
* Переход к выбранному экрану настроек.

1. *SettingsScreen:*

*Описание:* экран настроек.

*Функционал:*

* Отображение и редактирование списка настроек (локальных для конкретного чата или глобальных для текущего профиля);
* Сохранение изменённых настроек.

1. *ProfileSettingsScreen:*

*Описание:* экран настроек профиля.

*Функционал:*

* Отображение и редактирование настроек конкретного профиля;
* Сохранение изменённых настроек профиля;
* Возможность выхода из системы.

1. *ChatScreen:*

*Описание:* экран конкретного чата.

*Функционал:*

* Отображение сообщений чата;
* Набор и отправка новых сообщений;
* Прикрепление изображений к сообщениям для отправки;
* Изменение своих сообщений или сообщений ИИ-ассистента;
* Удаление своих сообщений или сообщений ИИ-ассистента;
* Принудительное продолжение генерации.

Эти экраны обеспечивают полное покрытие сформулированного ранее функционала, требуемого для приложения.

Граф переходов между экранами представлен в Приложении Г (изображение Г1).

Образцы внешнего вида экранов представлены в Приложении Д (изображения Д1–Д9).

# ЗАКЛЮЧЕНИЕ

Целью данного курсового проекта было создание мобильного приложения, позволяющего осуществлять коммуникацию с локально развёрнутой LLM. В результате работы нами была достигнута основная цель проекта – разработано полноценное мобильное приложение, обеспечивающее удобный интерфейс для взаимодействия с LLM.

В процессе разработки были выполнены следующие задачи:

* Проведён анализ существующих решений и определены основные требования к разрабатываемому приложению.
* Разработана архитектура приложения с использованием фреймворка Jetpack Compose для построения интерфейса.
* Реализованы экраны приложения, такие как ProfilesScreen, NewProfileScreen, ChatsScreen, SettingsChooseScreen, SettingsScreen, ProfileSettingsScreen и ChatScreen.
* Выполнена интеграция с библиотеками Retrofit для сетевых запросов и Room для локального хранения данных.

При разработке приложения были использованы современные технологии и инструменты, такие как Android Studio, Jetpack Compose, Retrofit, Room и Accompanist. Это позволило создать удобный и функциональный продукт, соответствующий всем требованиям технического задания.

Несмотря на достигнутые результаты, существуют перспективы дальнейшего развития приложения, такие как:

* Поддержка стандарта OpenAI API (в настоящее время поддерживается только llama.cpp).
* Расширение функционала по управлению профилями и чатами.
* Внедрение системы автоматического перевода, что сделает приложение доступным для более широкого круга пользователей.

Таким образом, тему курсового проекта можно считать раскрытой, а цель работы – создание мобильного приложения для коммуникации с локальной LLM – достигнутой. Материалы данного курсового проекта соответствуют всем требованиям и могут быть использованы для дальнейшего развития и совершенствования подобных приложений.

# ИНФОРМАЦИОННЫЕ РЕСУРСЫ

* Principles of navigation [Электронный ресурс]. Режим доступа: открытый, <https://developer.android.com/guide/navigation/principles> (Дата обращения: 01.07.2024).
* Lifecycle of composables in Jetpack Compose [Электронный ресурс]. Режим доступа: открытый, <https://www.kodeco.com/32617206-lifecycle-of-composables-in-jetpack-compose> (Дата обращения: 01.07.2024).
* More performance tips for Jetpack Compose [Электронный ресурс]. Режим доступа: открытый, <https://www.youtube.com/watch?v=ahXLwg2JYpc> (Дата обращения: 01.07.2024).
* State and Jetpack Compose [Электронный ресурс]. Режим доступа: открытый, <https://developer.android.google.cn/develop/ui/compose/state?hl=en> (Дата обращения: 01.07.2024).
* Погружаемся в работу со скроллом в Jetpack Compose [Электронный ресурс]. Режим доступа: открытый, <https://habr.com/ru/companies/skyeng/articles/654049/> (Дата обращения: 01.07.2024).
* Using ViewModel-LiveData with Jetpack Compose [Электронный ресурс]. Режим доступа: открытый, <https://dev.to/mahendranv/using-viewmodel-livedata-with-jetpack-compose-31h8> (Дата обращения: 01.07.2024).
* Jetpack Compose best practices [Электронный ресурс]. Режим доступа: открытый, <https://developer.android.com/develop/ui/compose/performance/bestpractices> (Дата обращения: 01.07.2024).
* ANDROID TOUCH SYSTEM – PART 4: GESTURE-HANDLING MODIFIERS IN JETPACK COMPOSE [Электронный ресурс]. Режим доступа: открытый, <https://www.droidcon.com/2022/07/19/android-touch-system-part-4-gesture-handling-modifiers-in-jetpack-compose/> (Дата обращения: 01.07.2024).
* Pick an image from Gallery or take a picture from Camera with JetPack Compose [Электронный ресурс]. Режим доступа: открытый, <https://medium.com/@yvanaonana/pick-an-image-from-gallery-or-take-a-picture-from-camera-with-jetpack-compose-23ce92bdd68a> (Дата обращения: 01.07.2024).
* Requesting Permissions With Ease in Jetpack Compose Using Accompanist permissions APIs [Электронный ресурс]. Режим доступа: открытый, <https://medium.com/compose-in-the-room/requesting-permissions-with-ease-in-jetpack-compose-using-accompanist-permissions-apis-76d5d9ca5f97> (Дата обращения: 01.07.2024).
* Window insets in Compose [Электронный ресурс]. Режим доступа: открытый, <https://developer.android.com/develop/ui/compose/layouts/insets> (Дата обращения: 01.07.2024).
* Thinking in Compose [Электронный ресурс]. Режим доступа: открытый, <https://developer.android.com/develop/ui/compose/mental-model> (Дата обращения: 01.07.2024).
* Установка отступов и смещения [Электронный ресурс]. Режим доступа: открытый, <https://metanit.com/kotlin/jetpack/3.4.php> (Дата обращения: 01.07.2024).
* Корутины и асинхронность. Введение в корутины [Электронный ресурс]. Режим доступа: открытый, <https://metanit.com/kotlin/jetpack/10.1.php> (Дата обращения: 01.07.2024).
* Как создавать анимации в Jetpack Compose [Электронный ресурс]. Режим доступа: открытый, <https://habr.com/ru/companies/jugru/articles/683656/> (Дата обращения: 01.07.2024).
* Jetpack Compose: State and LiveData [Электронный ресурс]. Режим доступа: открытый, <https://code.luasoftware.com/tutorials/android/jetpack-compose-state-and-livedata> (Дата обращения: 01.07.2024).

# ПРИЛОЖЕНИЯ

## ПРИЛОЖЕНИЕ А

ТЕХНИЧЕСКОЕ ЗАДАНИЕ  
на разработку мобильного приложения “API Chat”

### Введение

Данное мобильное приложение создано для осуществления коммуникации с локально развёрнутой LLM на устройстве пользователя, что позволяет пользователям взаимодействовать с ИИ без необходимости подключения к удалённым серверам частных организаций.

### Основание для разработки

Основанием для разработки является задание, выданное на курсовое проектирование на тему разработки мобильного приложения для коммуникации с локально развёрнутой LLM.

Полное наименование работы: «Разработка мобильного приложения для коммуникации с большими языковыми моделями».

### 1 Назначение разработки

Мобильное приложение предназначено для обеспечения пользователей возможностью общения с локально установленной LLM. Приложение позволяет пользователям создавать и управлять профилями, вести чаты с моделью, и настраивать параметры взаимодействия.

### 2 Требования к информационной системе

#### 2.1 Требования к функциональным характеристикам

Мобильное приложение должно обеспечивать выполнение следующих функций:

* Возможность создания, редактирования и удаления пользовательских профилей.
* Ведение чатов с локально развёрнутой LLM, с возможностью сохранения и удаления сообщений.
* Настройка параметров взаимодействия с LLM, таких как выбор модели и её параметров.
* Управление общими настройками приложения и настройками профилей.

Выходными данными мобильного приложения является история чатов с LLM.

Функционал приложения реализован на следующих экранах:

* ProfilesScreen – управление списком профилей.
* NewProfileScreen – создание нового профиля.
* ChatsScreen – отображение списка чатов.
* SettingsChooseScreen – выбор настроек.
* SettingsScreen – настройка параметров приложения.
* ProfileSettingsScreen – настройка параметров профиля.
* ChatScreen – взаимодействие с LLM в формате чата.

#### 2.2 Требования к надежности

Мобильное приложение должно иметь:

* Ограничение несанкционированного доступа к данным через авторизацию пользователя.
* Контроль вводимой информации и блокировку некорректных действий пользователя при работе с системой.

#### 2.3 Требования к составу и параметрам технических средств

Системные требования для работы мобильного приложения:

* Операционная система: Android 8.0 и выше.
* Процессор: ARM64.
* Оперативная память: 2 ГБ и более.
* Свободное место на устройстве: 500 МБ.

#### 2.4 Требования информационной и программной совместимости

Мобильное приложение должно работать на любом устройстве, соответствующем системным требованиям.

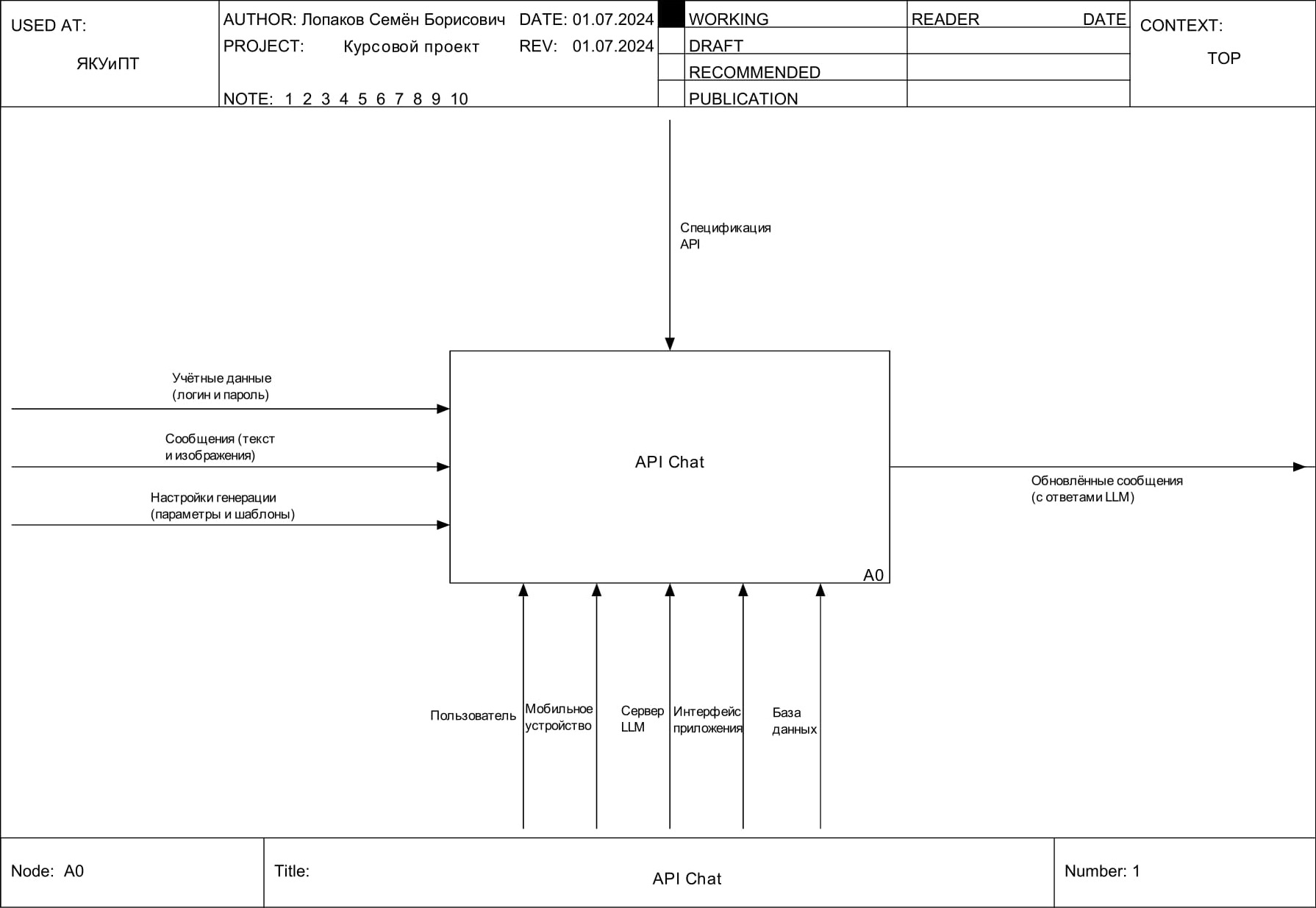
#### 2.5 Требования к транспортированию и хранению

Система распространяется через скомпилированный файл сборки в формате .apk.

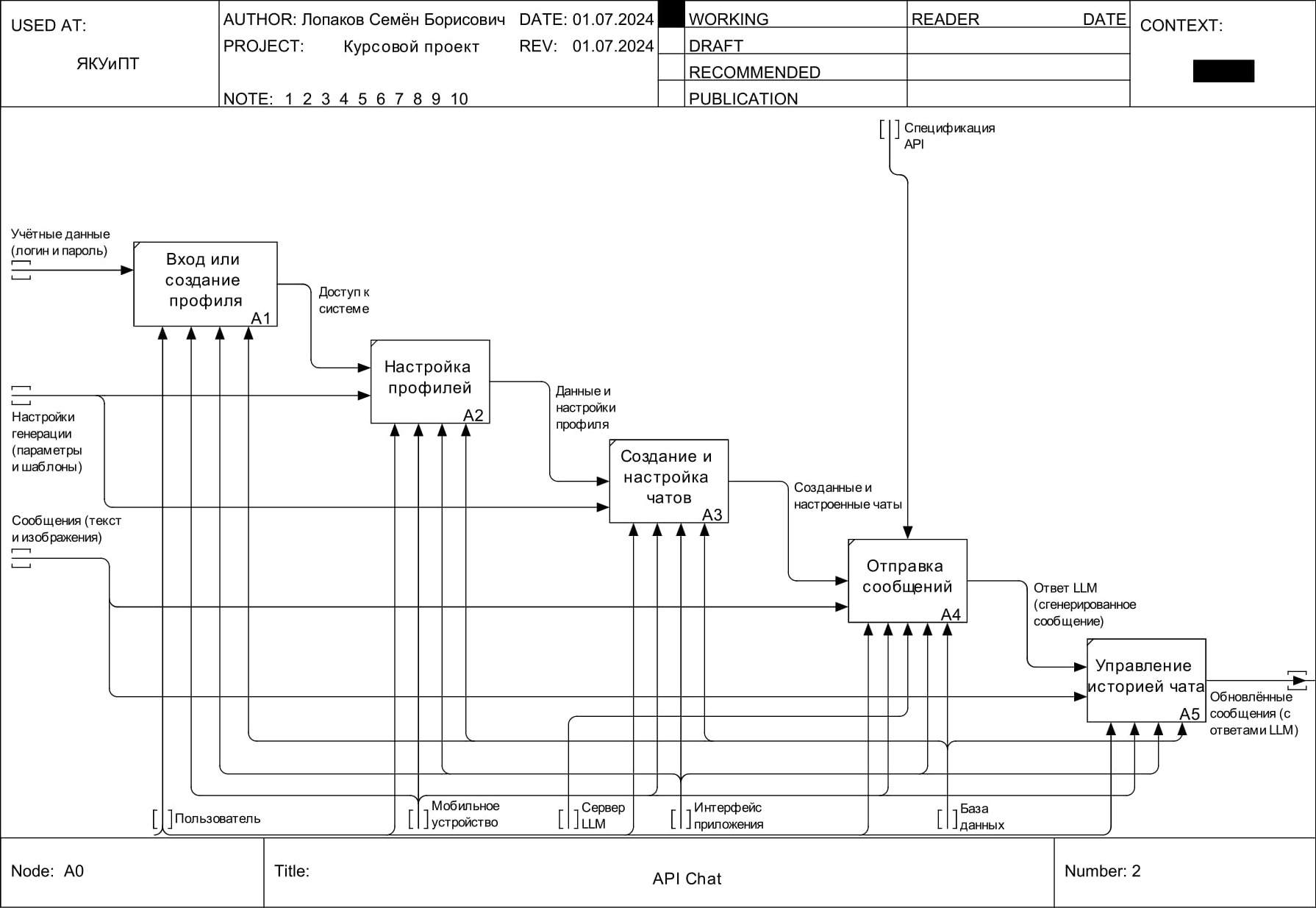
#### 3. Требования к программной документации

В ходе разработки программы должно быть подготовлено техническое задание на мобильное приложение и описание системы

## ПРИЛОЖЕНИЕ Б

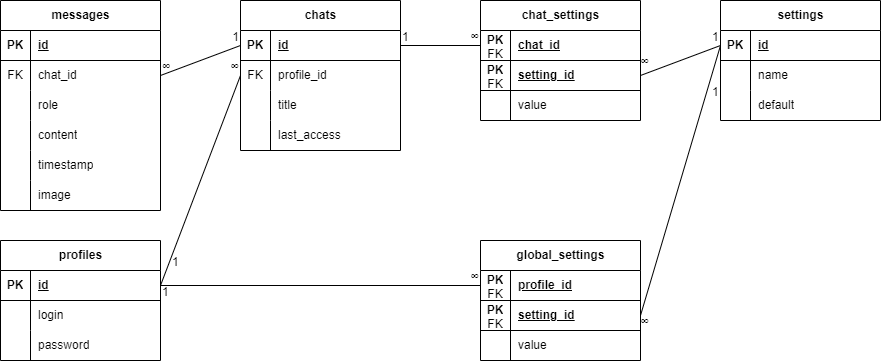


*Изображение Б1.* Контекстная диаграмма мобильного приложения API Chat.



*Изображение Б2.* Диаграмма декомпозиции мобильного приложения API Chat.

## ПРИЛОЖЕНИЕ В



*Изображение В1.* Схема локальной базы данных мобильного приложения API Chat.

@Entity(  
 tableName="profiles",  
 indices = [  
 Index(  
 value = [ "login" ],  
 unique = true  
 )  
 ]  
)  
data class DbProfile(  
 @PrimaryKey(autoGenerate = true)  
 @ColumnInfo(name="id")  
 val id: Int = 0,  
 @ColumnInfo(name="login")  
 var login: String,  
 @ColumnInfo(name="password")  
 var password: String,  
)

@Entity(tableName="chats", foreignKeys = [  
 ForeignKey(  
 entity = DbProfile::class,  
 parentColumns = ["id"],  
 childColumns = ["profile\_id"],  
 onDelete = ForeignKey.CASCADE  
 )  
])  
data class DbChat(  
 @PrimaryKey(autoGenerate=true)  
 @ColumnInfo(name="id")  
 val id: Int = 0,  
 @ColumnInfo(name="profile\_id", index=true)  
 val profileId: Int,  
 @ColumnInfo(name="title")  
 var title: String,  
 @ColumnInfo(name="last\_access")  
 val lastAccess: Long  
)

@Entity(tableName="messages", foreignKeys = [  
 ForeignKey(  
 entity = DbChat::class,  
 parentColumns = ["id"],  
 childColumns = ["chat\_id"],  
 onDelete = ForeignKey.CASCADE  
 )  
])  
data class DbMessage(  
 @PrimaryKey(autoGenerate=true)  
 @ColumnInfo(name="id")  
 val id: Int = 0,  
 @ColumnInfo(name="chat\_id", index=true)  
 val chatId: Int,  
 @ColumnInfo(name="role")  
 val role: String,  
 @ColumnInfo(name="timestamp")  
 var timestamp: Long,  
 @ColumnInfo(name="content")  
 var content: String,  
 @ColumnInfo(name="image")  
 var image: String?  
)

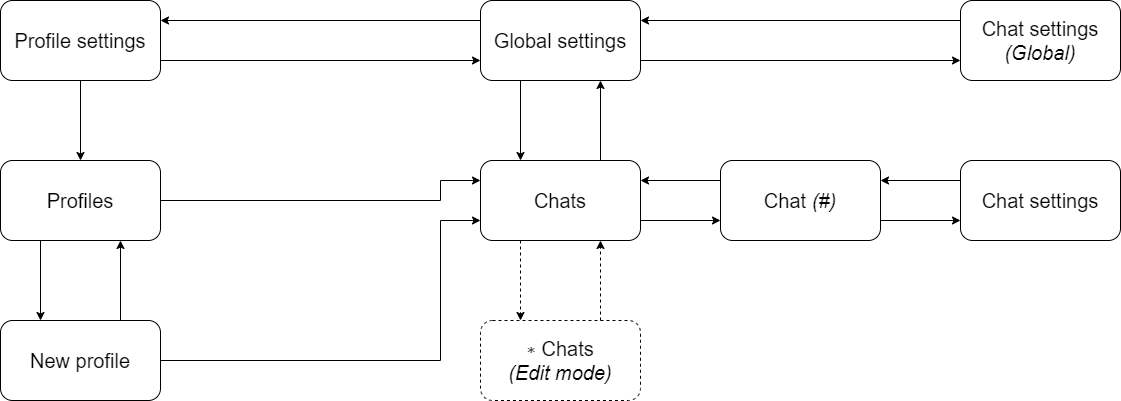
@Entity(tableName="settings")  
data class DbSetting(  
 @PrimaryKey(autoGenerate=true)  
 @ColumnInfo(name="id")  
 val id: Int = 0,  
 @ColumnInfo(name="name", index=true)  
 val name: String,  
 @ColumnInfo(name="default")  
 val default: String,  
)

@Entity(tableName="chat\_settings",  
 foreignKeys = [  
 ForeignKey(  
 entity = DbChat::class,  
 parentColumns = ["id"],  
 childColumns = ["chat\_id"],  
 onDelete = ForeignKey.CASCADE  
 ),  
 ForeignKey(  
 entity = DbSetting::class,  
 parentColumns = ["id"],  
 childColumns = ["setting\_id"],  
 onDelete = ForeignKey.CASCADE  
 )  
 ],  
 primaryKeys = [  
 "chat\_id", "setting\_id"  
 ]  
)  
data class DbChatSetting(  
 @ColumnInfo(name="chat\_id")  
 val chatId: Int,  
 @ColumnInfo(name="setting\_id", index=true)  
 val settingId: Int,  
 @ColumnInfo(name="value")  
 val value: String,  
)

@Entity(tableName="global\_settings",  
 foreignKeys=[  
 ForeignKey(  
 entity = DbProfile::class,  
 parentColumns = ["id"],  
 childColumns = ["profile\_id"],  
 onDelete = ForeignKey.CASCADE  
 ),  
 ForeignKey(  
 entity = DbSetting::class,  
 parentColumns = ["id"],  
 childColumns = ["setting\_id"],  
 onDelete = ForeignKey.CASCADE  
 ),  
 ],  
 primaryKeys = [  
 "profile\_id", "setting\_id"  
 ]  
)  
data class DbGlobalSetting(  
 @ColumnInfo(name="profile\_id")  
 val profileId: Int,  
 @ColumnInfo(name="setting\_id", index=true)  
 val settingId: Int,  
 @ColumnInfo(name="value")  
 val value: String,  
)

*Листинг В2.* Реализация базы данных мобильного приложения API Chat с помощью Room.

## ПРИЛОЖЕНИЕ Г



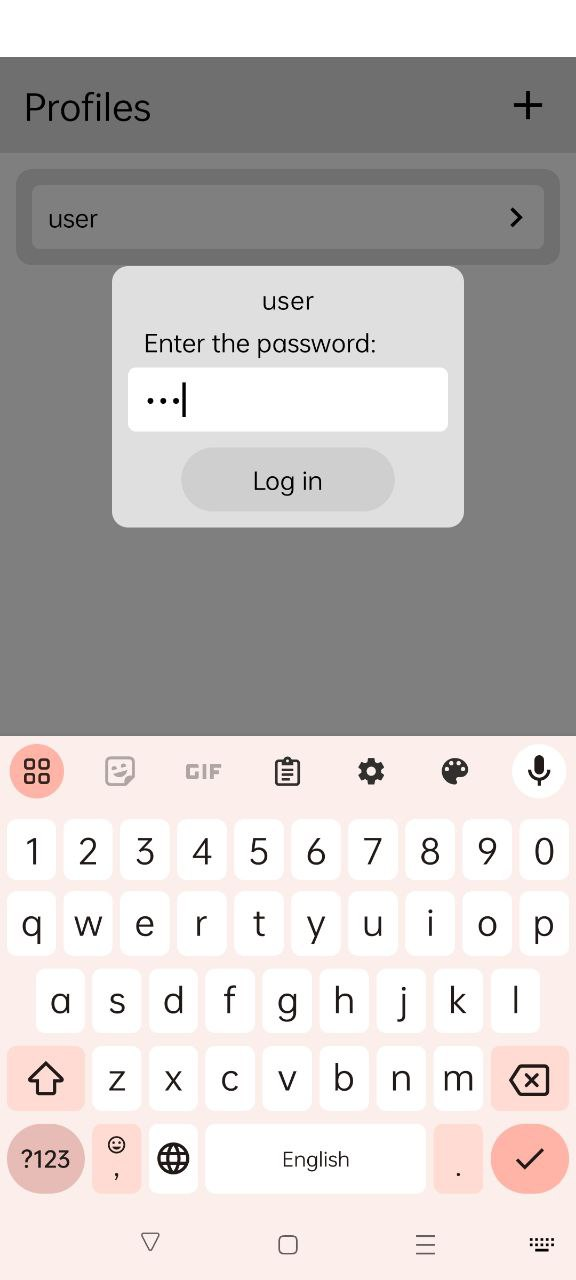
*Изображение Г1.* Граф переходов между экранами мобильного приложения API Chat.

Экран *Profiles* является начальным.

## ПРИЛОЖЕНИЕ Д

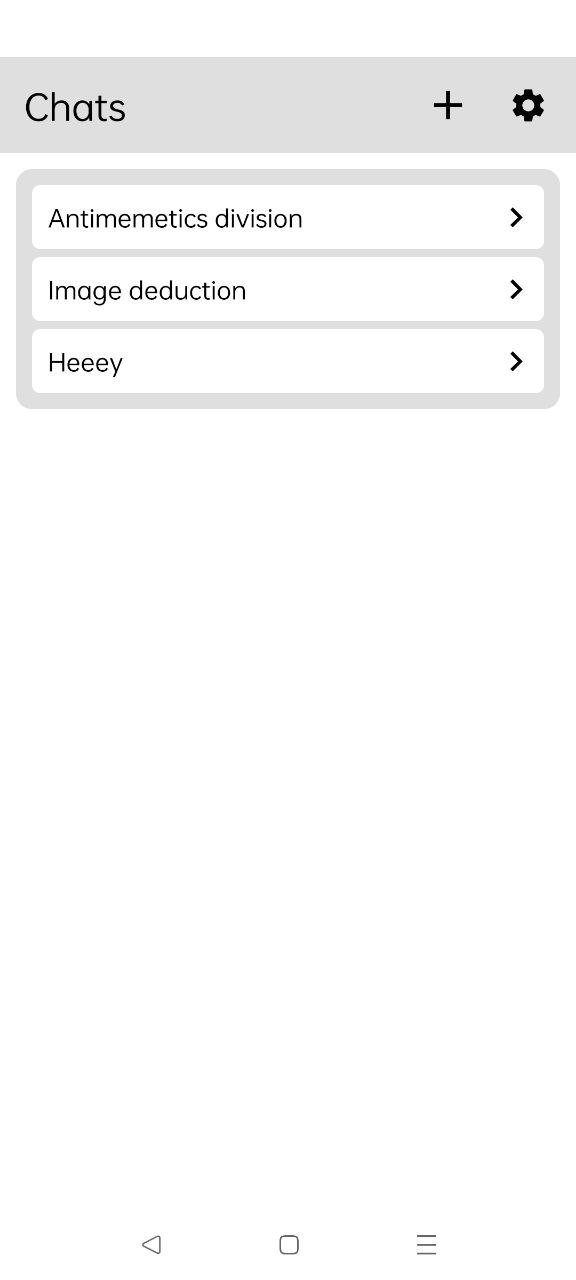


*Изображение Д1.*Экран профилей



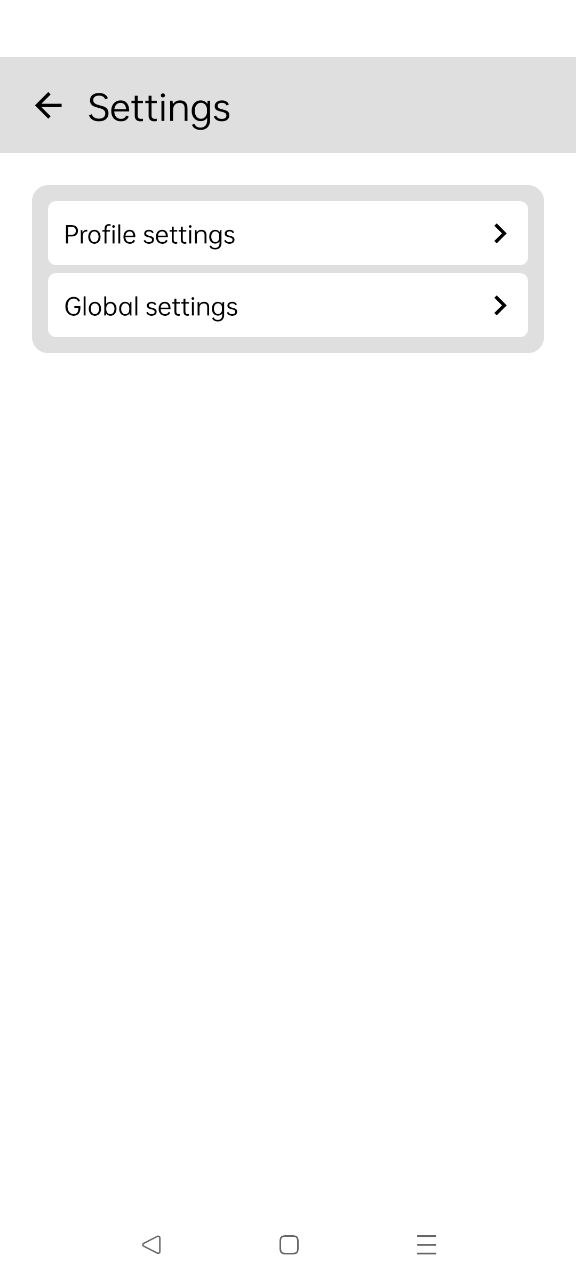
*Изображение Д2.*

Экран профилей (вход)



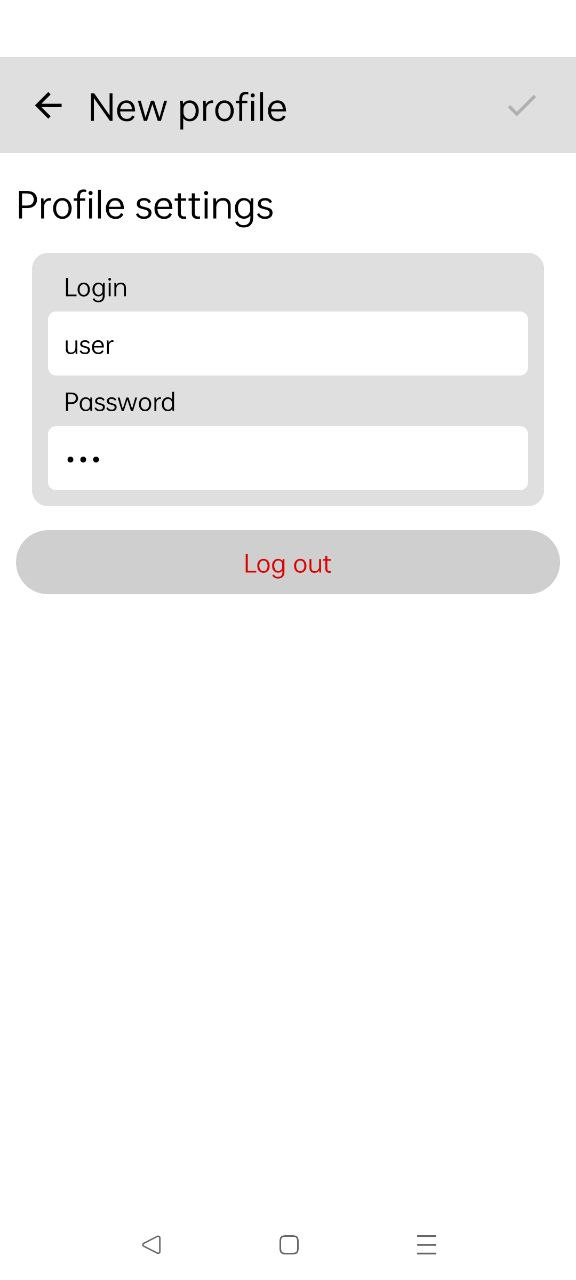
*Изображение Д3.*

Экран чатов



*Изображение Д4.*

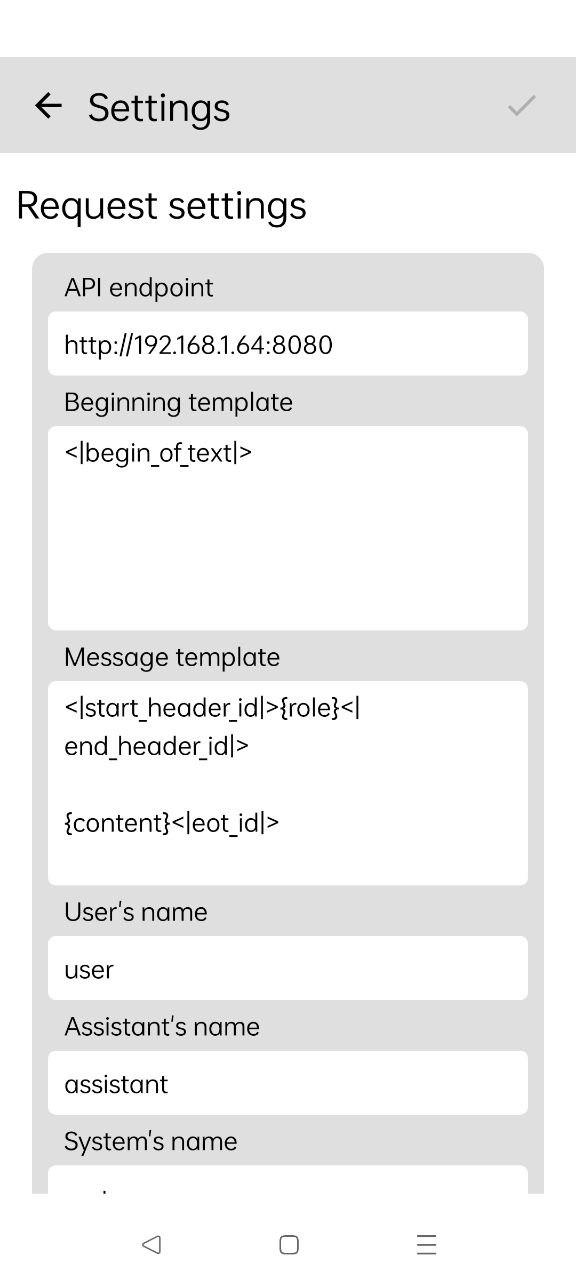
Экран выбора вида настроек



*Изображение Д5.*

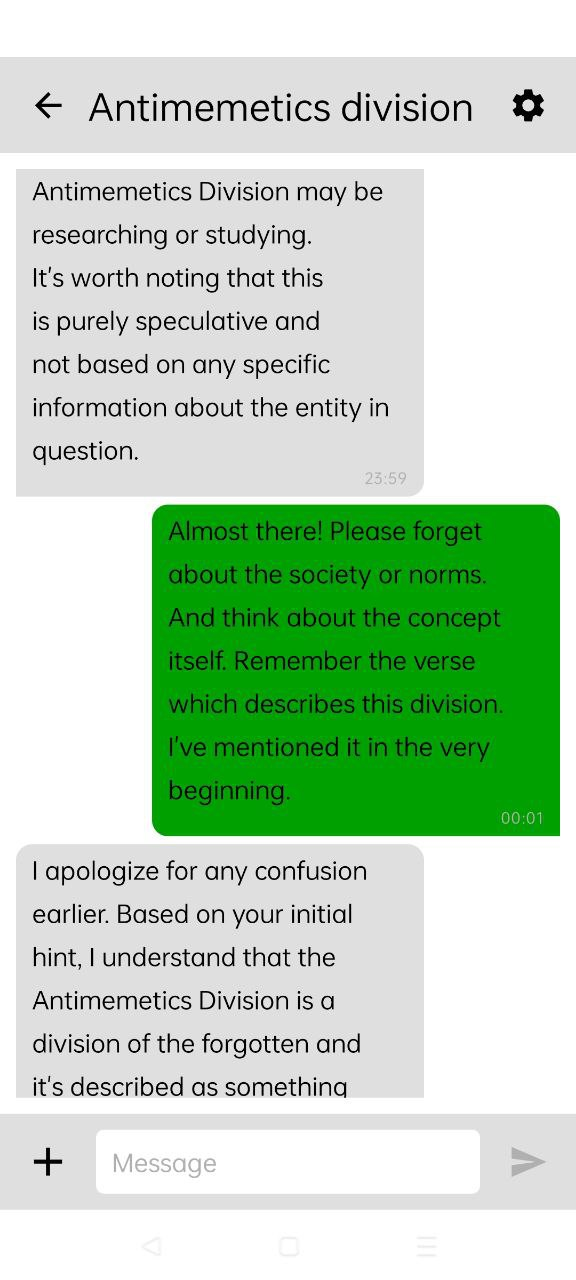
Экран настроек профиля

*(баг именования!)*



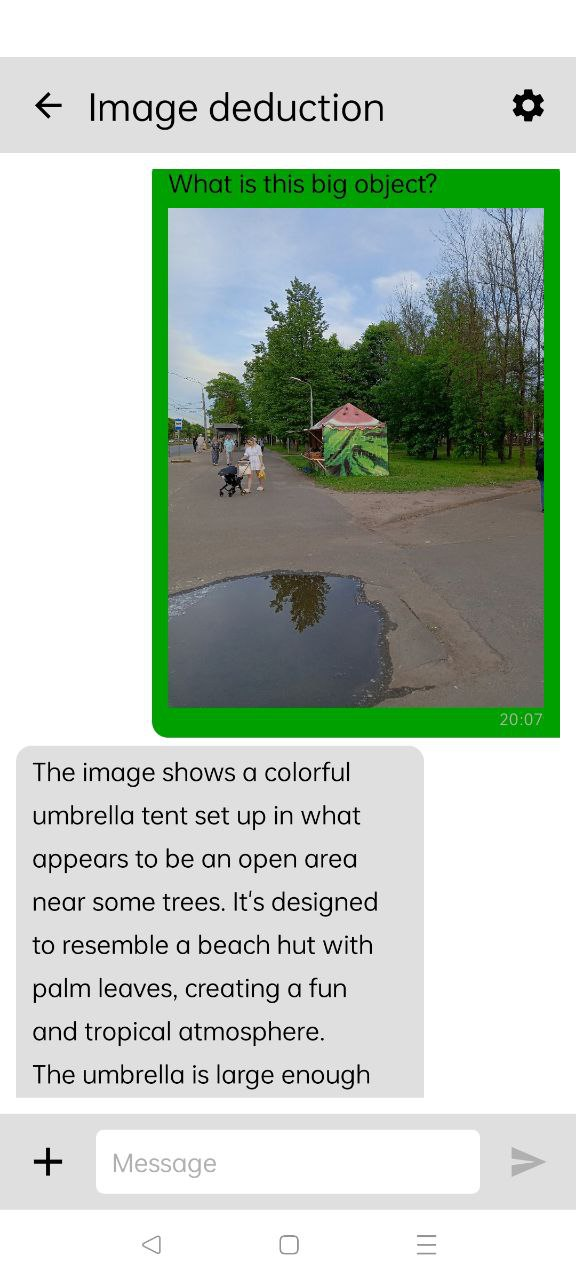
*Изображение Д6.*

Экран настроек чата



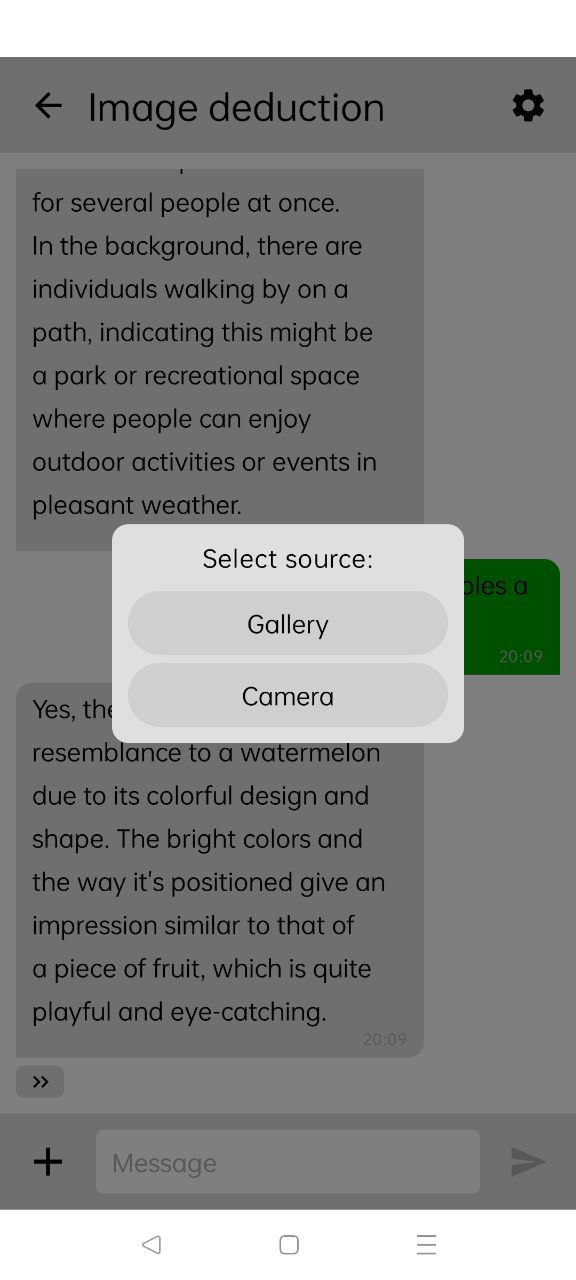
*Изображение Д7.*

Экран чата



*Изображение Д8.*

Экран чата (с изображением)



*Изображение Д9.*

Экран чата (выбор источника изображения, конец переписки)