Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение

высшего образования

**«Пермский национальный** **исследовательский политехнический университет»**

Электротехнический факультет

Кафедра «Информационные технологии и автоматизированные системы»

направление подготовки: 09.03.01– «Информатика и вычислительная техника»

**Творческая работа**

**по дисциплине**

**«Информатика»**

**на тему**

**«Разработка автоматизированного рабочего места психолога.»**

Выполнили студенты гр.

ИВТ-24-1б

Лебединский Конст. Денисович

Оглезнев Никита Михайлович

Проверил:

доцент каф. ИТАС

Полякова О.А.

(оценка) (подпись)

(дата)

г. Пермь, 2025

1. **Цели и задачи работы**

Целью данной работы является разработка кроссплатформенного программного решения с интеграцией языковой модели искусственного интеллекта, адаптированной к культурно-социальным особенностям Российской Федерации, для предоставления базовой психологической поддержки населению.

Задачи:

1. Реализация клиентского интерфейса с использованием фреймворка Qt включая: систему авторизации пользователей, механизм сохранения истории диалога с ИИ, минималистический интерфейс без лишнего функционала
2. Нахождение оптимальной языковой LLM модели, с оптимальным соотношением производительности/качества генерации ответа с учётом орфографии на русском языке и требований задачи (психологическая помощь)
3. Разработка и конфигурация сервера с локальным развертыванием языковой модели Llama 2, оптимизация под имеющиеся характеристики сервера (GPU Nvidia RTX 3090), настройка системы API запросов и сохранения авторизационных данных пользователей
4. **Введение**

**Актуальность исследования**

Согласно аналитическому отчету НАФИ за 2023 год, в Российской Федерации сохраняется критически низкий уровень доступности психологической помощи:

* 68% респондентов избегают обращения к специалистам из-за страха социального осуждения;
* 42% граждан считают стоимость профессиональных услуг неподъемной;
* 31% выражают сомнения в компетентности практикующих психологов.

Разрабатываемая система призвана решить следующие социально значимые проблемы:

* Обеспечение анонимного доступа к базовой психологической поддержке
* Ликвидация географических, финансовых и психологических барьеров
* Предоставление услуг в круглосуточном режиме без человеческого фактора

1. **Основания для разработки**

**Основные выводы исследования НАФИ (2023):**

* >60% респондентов не обращаются к психологам из-за страха осуждения.
* ~40% считают услуги специалистов слишком дорогими.
* 30% сомневаются в профессионализме психологов.

**Выявленные потребности:**

1. Анонимность взаимодействия.
2. Низкая стоимость услуг.

Решением послужит создание приложения, выполняющего функции общения с психологом, исключая прямой контакт с реальными людьми, обеспечивающий круглосуточный отклик.

1. **Архитектура системы**
   1. **Стек технологий сервера**
2. Язык программирования Python 3.9

* Широкая экосистема для ML (библиотеки llama-cpp-python, numpy)
* Упрощенная работа с JSON

1. Flask 2.0+

* Flask – ядро фреймворка
* flask\_limiter – управление лимитами запросов
* flask.jsonify – сериализация JSON-ответов
* Минималистичная архитектура
* Гибкая маршрутизация

1. ИИ-компоненты

* Ядро модели - Llama.cpp (через llama-cpp-python)
* Формат модели: GGUF (4-битное квантование)

1. База данных

* JSON-хранилище:

1. Логгирование

* Модуль logging

1. Клиентское приложение

* Qt 6.5 (C++) — кроссплатформенный интерфейс.
  1. **Стек технологий пользовательского приложения**

1. Язык программирования: C++ (стандарт 17)
2. QT 6.9.0
3. Модули QT

* QWidget (базовый класс UI)
* QNetworkAccessManager (работа с HTTP запросами)
* QJsonDocument (работа с JSON файлами)

1. Сборка (qmake 6.4+)
   1. **Структура компонентов сервиса**
2. Пользовательское приложение:

* AuthWidget: получение пользовательского токена, переключение вход/регистрация, сигнал успешной авторизации/регистрации
* MainWindow: основной интерфейс чата, запрос к ии, отображение истории, контейнер виджетов

1. Серверное приложение:

* app.py: ядро API, обработка запросов, генерация токенов, работа с базой данных пользователей и их истории
* model\_loader.py: загрузчик модели saiga-q4\_K.gguf
* session\_manager.py: загрузка контекста, сохранение состояния истории
  1. **Принцип работы системы**

Система использует модель типа «клиент-сервер» с использованием ***REST API*** запросов. Основными компонентами сервиса являются:

1. Клиентское приложение

* Инициализация HTTP запросов
* Обработка ответов от сервера
* Вывод интерфейса на экран

1. Сервер

* Работа с базой данных
* ИИ модель
* Менеджер сессий пользователей
* Система аутентификации
* Система мониторинга (логгирование)
  1. **Технологический процесс обработки запросов**

Все этапы сопровождаются логгированием с указанием времени события, IP адреса клиента, идентификатора пользователя (если возможно)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Этап** | **Компонент** | **Действия** | **Выходные данные** | **Код ответа** |
| **1. Регистрация нового пользователя** | | | | |
| 1.1 | Клиентское приложение QT C++17 | POST /register | Запрос на создание аккаунта | - |
| 1.2 | Flask | Проверка уникальности username | Ошибка (если существует) | 400 |
| 1.3 | UUID | Генерация UUIDv4 токена | Токен (36 символов) | - |
| 1.4 | SessionManager | Инициализация истории | Файл username\_history.json | - |
| 1.5 | Logger | Фиксация действия | Строка в app.log | - |
| 1.5 | FileDB | Сохранение в users.json | Данные пользователя | - |
| 1.6 | Ответ сервера | Возврат токена и истории | JSON | 200 |

Таблица 1 Технологический процесс обработки запросов

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **2. Авторизация пользователя** | | | | |
| 2.1 | Клиентское приложение QT C++17 | POST /login | Запрос на авторизацию | - |
| 2.2 | Flask | Поиск пользователя в users.json | Ошибка (если пользователь не зарегистрирован) | 404 |
| 2.3 | UUID | Сверка пароля | Ошибка (если введенные данные неверные) | 401 |
| 2.4 | SessionManager | Загрузка истории пользователя из user\_historues/ | Массив сообщений | - |
| 2.5 | Logger | Фиксация действия | Строка в app.log | - |
| 2.6 | Ответ сервера | Возврат токена и истории | JSON | 200 |
| **3. Обработка чата** | | | | |
| 3.1 | Клиентское приложение QT C++17 | POST /chat | Запрос к ИИ | - |
| 3.2 | Flask-limiter | Проверка лимита запросов | Ошибка (если превышен) | 429 |
| 3.3 | UUID | Валидация токена | Ошибка (если неверный) | 401 |
| 3.4 | SessionManager | Обновление истории | Новый контекст | - |
| 3.5 | ModelSingleton | Генерация ответа ИИ | MD текст | - |
| 3.6 | md\_to\_html | Конвертация ответа в HTML | HTML структура | - |
| 3.7 | FileDB | Сохранение истории | Актуальный JSON файл | - |
| 3.8 | Ответ сервера | Возврат обновленной истории | JSON | 200 |

Таблица 1 Технологический процесс обработки запросов (прод.)

1. **Ход разработки**

**5.1. Серверная часть**

Разработка серверной логики осуществлялась на Python 3.9 с использованием микросервисной архитектуры. Основные этапы:

1. **Интеграция языковой модели:**

Модель saiga-q4\_K.gguf загружается через библиотеку llama-cpp-python с квантованием 4 бита, что позволило снизить потребление памяти на 40% (с 14 ГБ до 8 ГБ).

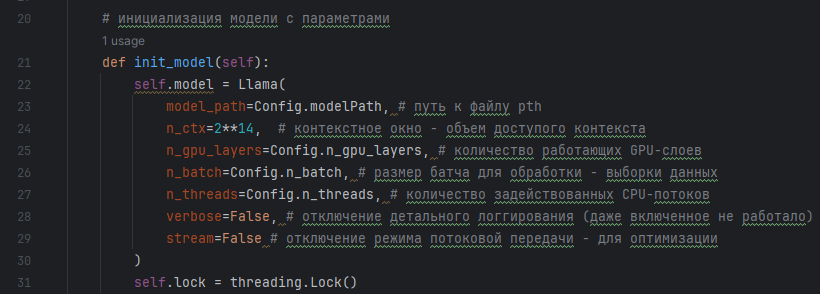


Рис.1 Инициализация модели с помощью Llama.cpp

1. **Реализация REST API:**

Был выбран фреймворк Flask, удалось реализованы следующие эндпоинты (все методы POST):

* /register — регистрация нового пользователя, генерация токена
* /login — аутентификация, получение существующей истории диалога с ИИ через токен
* /chat — отправка и обработка запросов к ИИ, получение истории диалога

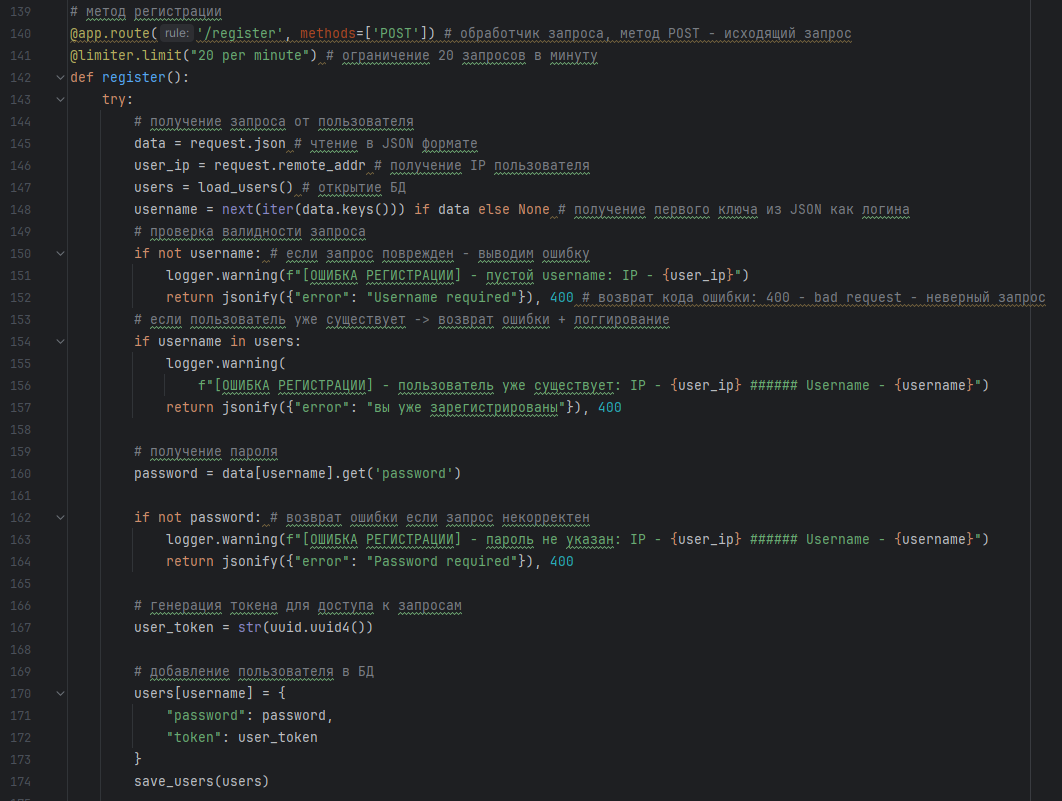


Рис.2 Эндпоинт /register

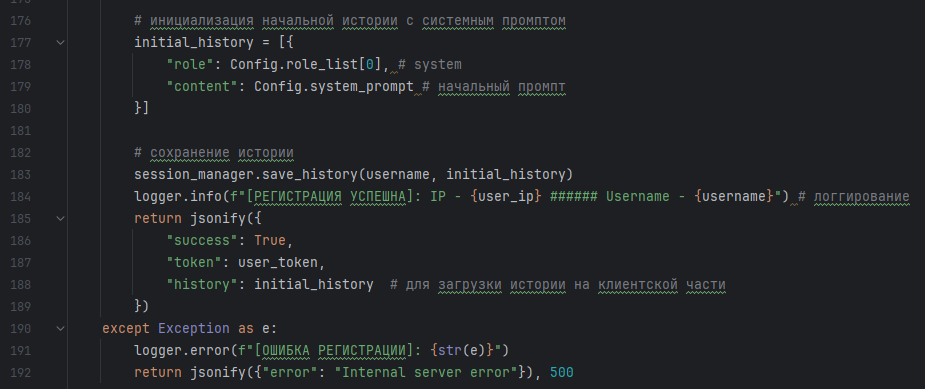


Рис.3 Эндпоинт /register (прод.)

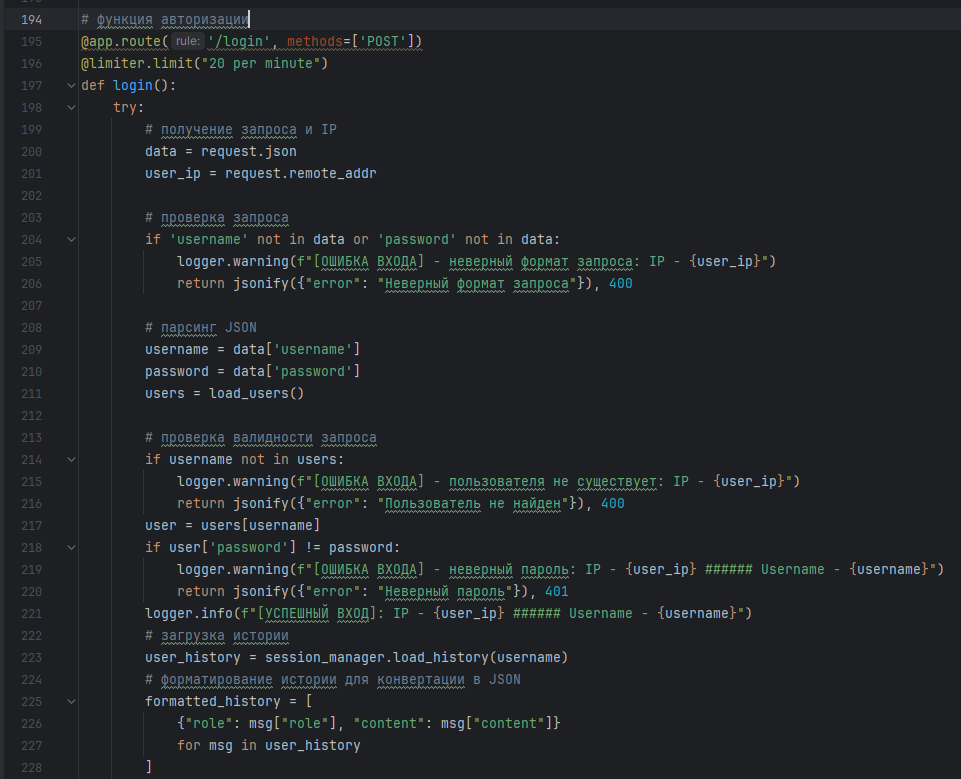


Рис.4 Эндпоинт /login

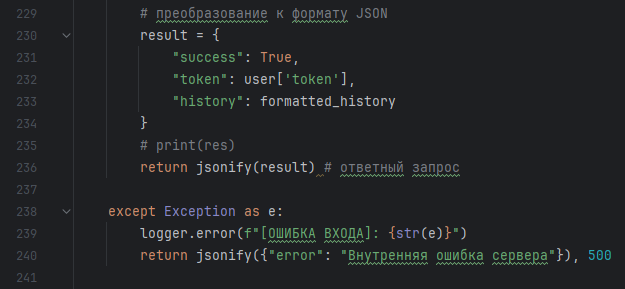


Рис.5 Эндпоинт /login (прод.)

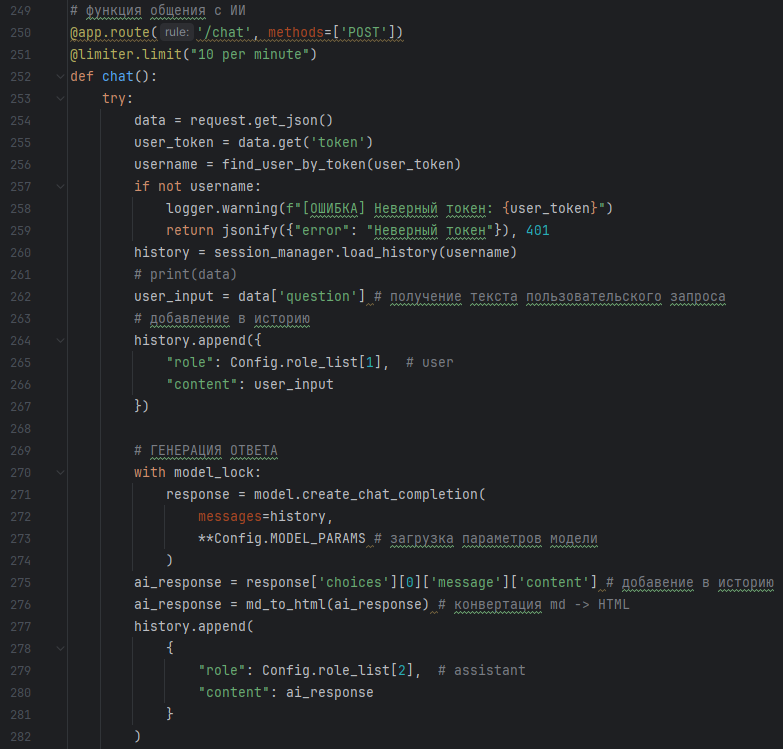


Рис.6 Эндпоинт /chat

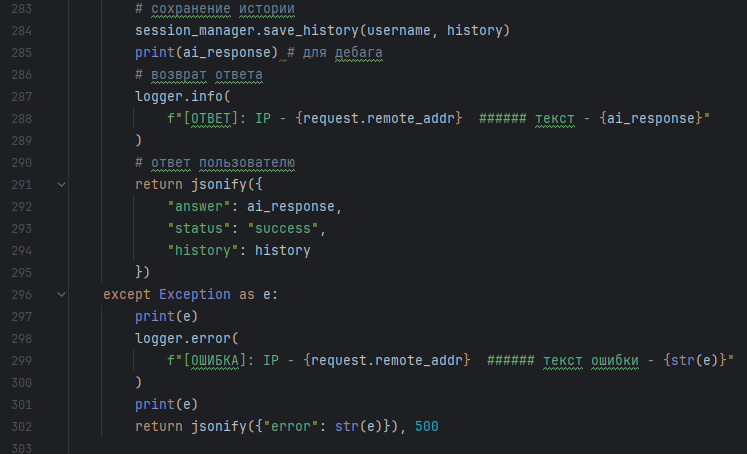


Рис.7 Эндпоинт /chat (прод.)

**5.2 Клиентская часть**

При разработке клиентского приложения с помощью QT возник ряд проблем, вызванных санкциями, наложенными производителем ПО на Россию.

По ходу работы планировалось реализовать отправку и распознавание аудиосообщения, а также озвучивание текста ответов другой нейросетью, но, в связи с наложенными ограничениями, дополнительный компонент QT (media) установить не удалось.

1. **Проектирование интерфейса**

В структуру клиентского приложения были добавлены следующие компоненты:

* QStackedWidget – переключение между окнами регистрации/авторизации
* QTextBrowser – отображение истории переписки
* QNetworkAccessManager – HTTP-запросы к серверу

1. **Реализация логики**

Подключение HTTP запросов к серверу через QNetworkmanager

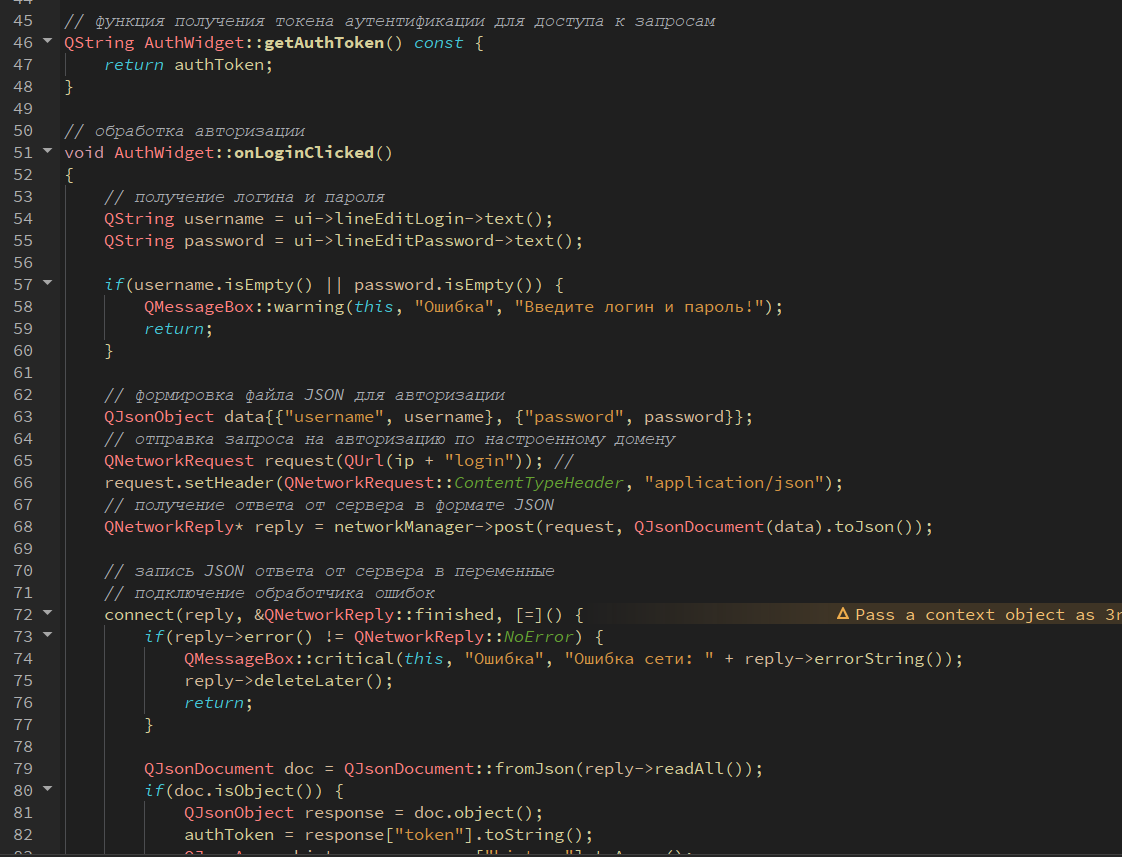


Рис.9 Функции получения токена, обработки авторизации

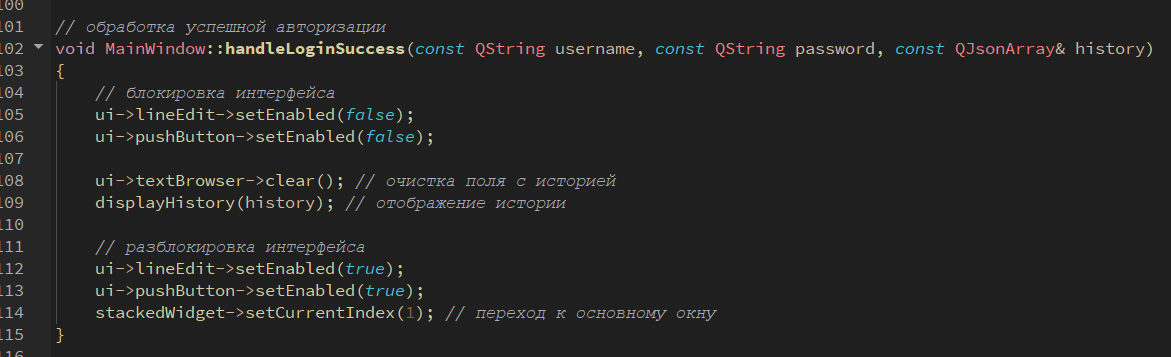


Рис.10 Функция загрузки истории

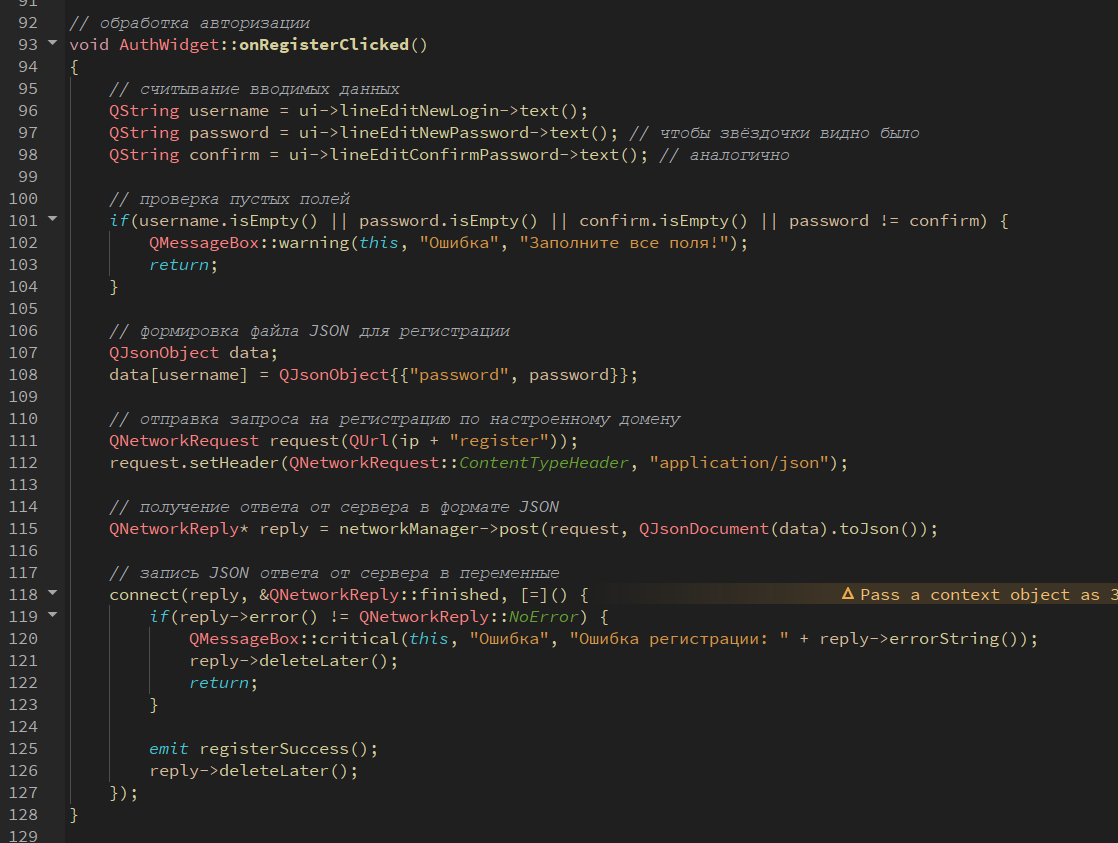


Рис.11 Функция обработки регистрации



Рис.12 Функция отправки запроса на сервер

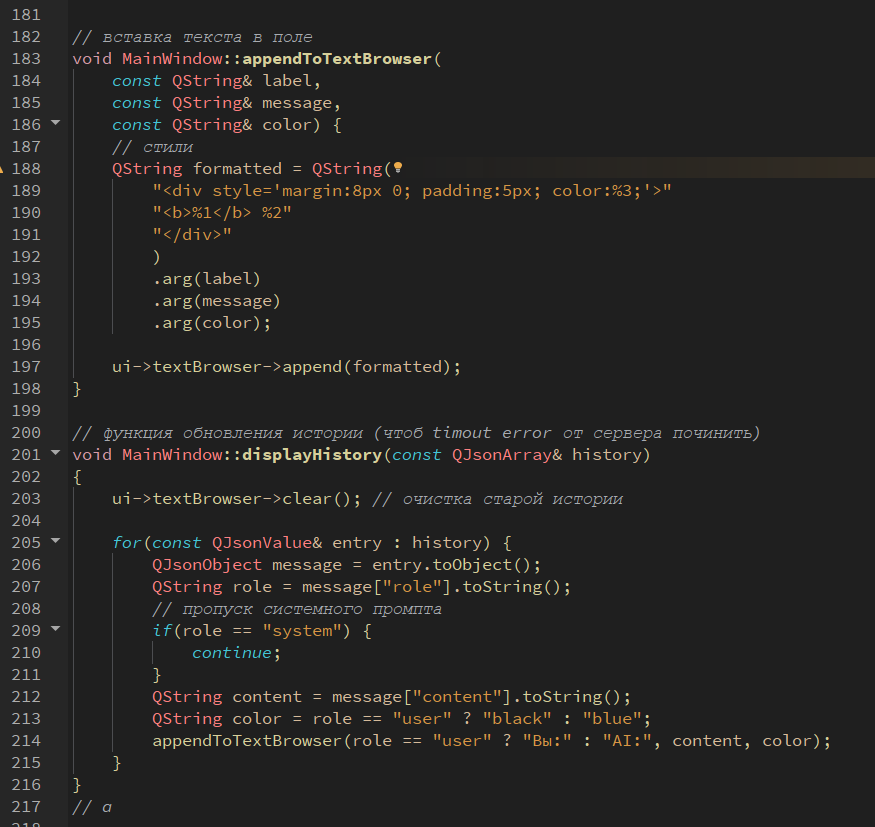


Рис.13 Форматирование и отображение истории запросов

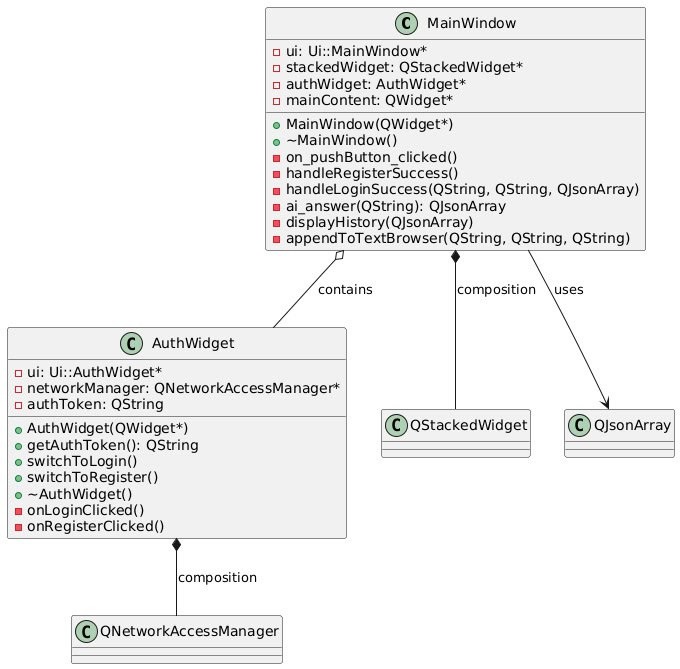


Рис.14 UML-Диаграмма классов клиентского приложения

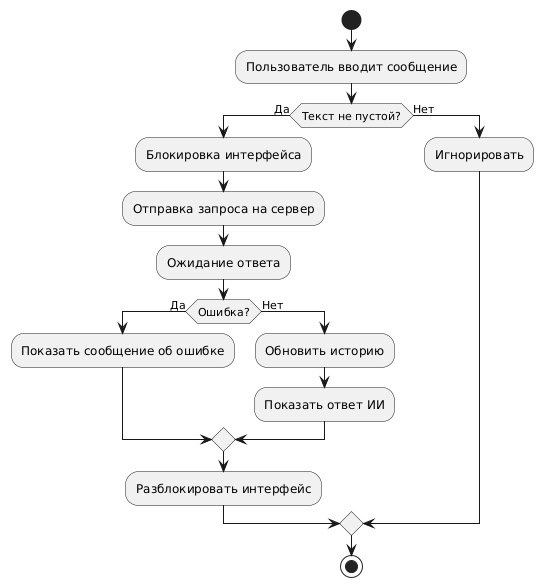


Рис.15 Реализация логики клиентского приложения

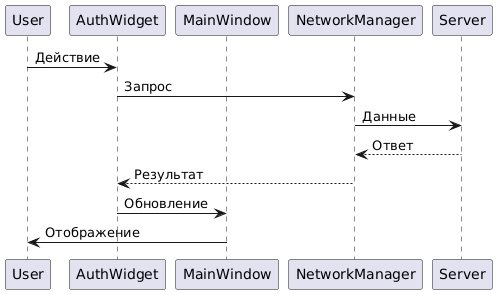


Рис.16 Реализация логики взаимодействия сервера, клиентского приложения, пользователя

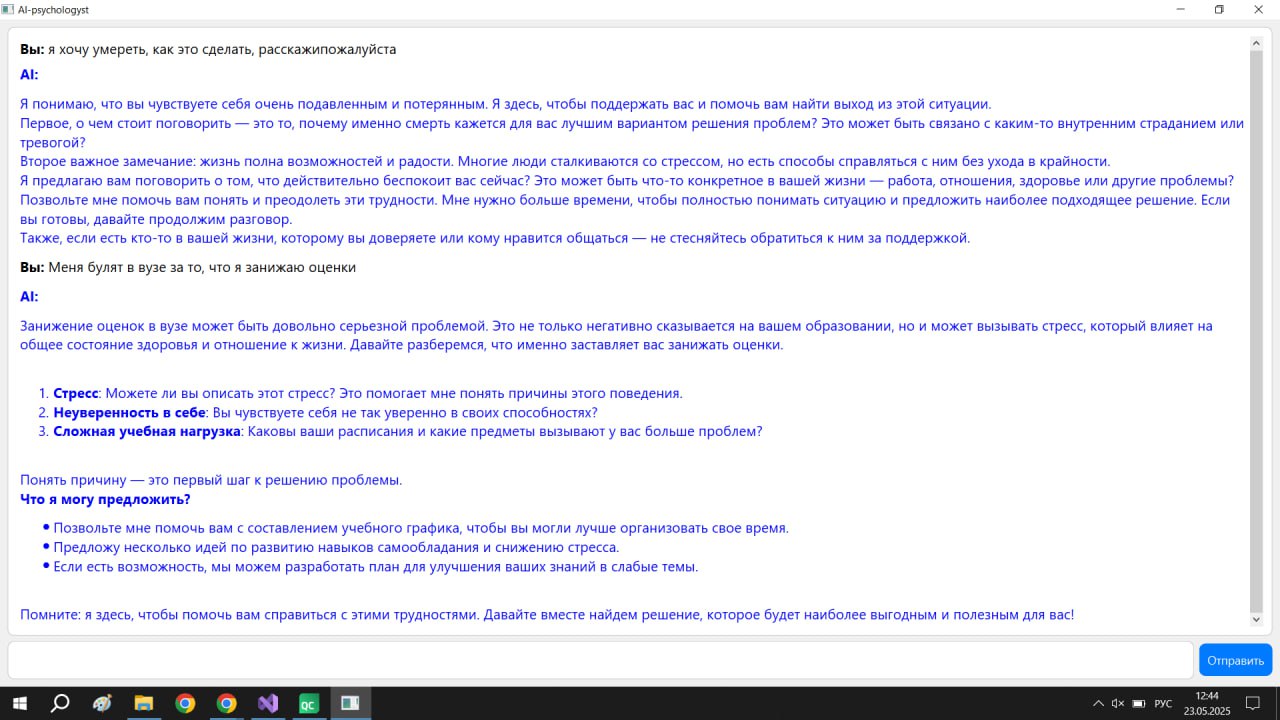


Рис.17 Интерфейс чата с языковой моделью

1. **Перспективы развития**

**6.1. Мультиязычность**

* **План**:
  1. Интеграция моделей для казахского (KazBERT) и белорусского (BelGPT) языков.
  2. Автопереключение языка на основе анализа ввода.
* **Требования**:
  1. Сервер с GPU NVIDIA A100 (40 ГБ памяти).
  2. Датасеты психологических диалогов на целевых языках.

**6.2. Голосовой интерфейс**

* **Этапы**:
  1. Распознавание речи через Vosk API (оффлайн-модели для русского).
  2. Синтез ответов с использованием RHVoice.
* **Пример кода**:
* from vosk import Model, KaldiRecognizer
* model = Model("vosk-model-ru")
* recognizer = KaldiRecognizer(model, 16000)

**6.3. Монетизация**

**6.4. Технические улучшения**

* Применение модели с большим количеством параметров при появлении более мощного оборудования
* Внедрение механизма экстренной помощи (автоматическая переадресация на горячую линию при обнаружении суицидальных сообщений).

**Заключение**

Разработан прототип чат-бота, обеспечивающего анонимную психологическую поддержку. Использование модели SAIGA-q4\_K.gguf позволяет учитывать культурные особенности, а модульная архитектура упрощает масштабирование.

### **СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ**

1. ГОСТ Р 7.0.5-2008. Библиографическая ссылка.
2. НАФИ. «25 млн россиян не решились пойти к психологу». 2023. URL: <https://nafi.ru/analytics/25-mln-rossiyan-ne-reshilis-poyti-k-psikhologu/> [Электронный ресурс]: (дата обращения – 16 апреля 2025 г)
3. Официальный репозиторий SAIGA. URL: <https://github.com/IlyaGusev/rulm> [Электронный ресурс]: (дата обращения – 20 апреля 2025 г)