**Техническое задание**

**Консольный клиент Telegram (TG\_CLI)**

**1. Общие сведения**

**1.1. Наименование системы**

Консольный клиент Telegram (TG\_CLI)

**1.2. Назначение и цели создания системы**

Создание консольного клиента для быстрой проверки новых сообщений в чатах и каналах Telegram без перехода в GUI-клиент и с минимальным отвлечением от основной работы пользователя.

**2. Требования к системе**

**2.1. Функциональные требования**

1. Авторизация пользователя в Telegram (однократно)
2. Отображение списка диалогов с возможностью:
   * Показа только непрочитанных диалогов
   * Показа всех диалогов
   * Переключения между этими режимами
3. Отображение истории сообщений для выбранного диалога
4. Ответ на конкретные сообщения с поддержкой цитирования
5. Поддержка медиа-контента с возможностью:
   * Отображения информации о медиа
   * Сохранения медиа-файлов на компьютер
   * Открытия сохраненных файлов во внешних программах
6. Кеширование данных для уменьшения количества запросов к серверам Telegram
7. Асинхронная загрузка данных без блокировки интерфейса
8. Работа параллельно с обычным клиентом Telegram на других устройствах

**2.2. Нефункциональные требования**

1. Интерфейс на русском языке
2. Минимальное использование ресурсов системы
3. Работа в терминале без графического интерфейса
4. Корректное отображение и работа с Unicode (эмодзи, кириллица и т.д.)
5. Поддержка работы в Windows, Linux и macOS
6. Отзывчивый интерфейс без "зависаний" при работе с сетью

**3. Технологический стек**

1. Язык программирования: Python 3.7+
2. Библиотеки:
   * Telethon - для работы с Telegram API
   * curses/windows-curses - для создания консольного интерфейса
   * asyncio - для асинхронной работы

**4. Архитектура системы**

**4.1. Структура проекта**

TG\_Console\_client/

├── app.py # Главный модуль запуска приложения

├── cli.py # Консольный интерфейс

├── config.py # Конфигурация приложения

├── state.py # Управление состоянием приложения

├── api/

│ └── client.py # Клиент для взаимодействия с API Telegram

├── input/

│ └── keys.py # Обработка клавиатурного ввода

├── services/

│ ├── cache.py # Менеджер кеширования данных

│ └── media.py # Обработчик медиа-контента

├── utils/

│ └── async\_loader.py # Асинхронный загрузчик данных

└── views/

├── chat.py # Отображение чата

└── dialogs.py # Отображение списка диалогов

**4.2. Компоненты системы**

1. **API-клиент** - обеспечивает взаимодействие с Telegram API
2. **CLI-интерфейс** - отвечает за отображение информации и взаимодействие с пользователем
3. **Обработчик ввода** - обрабатывает действия пользователя через клавиатуру
4. **Менеджер состояний** - управляет переключением между экранами и режимами
5. **Кеш-менеджер** - отвечает за кеширование данных
6. **Медиа-обработчик** - работает с медиа-контентом сообщений
7. **Асинхронный загрузчик** - обеспечивает неблокирующую загрузку данных

**5. Управление проектом и его параметры**

**5.1. Горячие клавиши**

1. **Глобальные клавиши**:
   * Ctrl+R - Переподключиться к серверу
   * Ctrl+C - Корректное завершение работы
   * Ctrl+L - Очистить экран и перерисовать интерфейс
2. **В режиме списка диалогов**:
   * ↑/↓ - Навигация по списку
   * → - Открыть выбранный диалог
   * Tab - Переключить режим отображения (все/только непрочитанные)
   * Q - Выход из программы
3. **В режиме чата**:
   * ↑/↓ - Навигация по сообщениям
   * ← - Вернуться к списку диалогов
   * R - Ответить на выбранное сообщение
   * S - Сохранить медиа из сообщения (если есть)
   * Q - Вернуться к списку диалогов
4. **В режиме ответа**:
   * Enter - Отправить сообщение
   * Esc - Отменить ответ

**5.2. Параметры конфигурации**

1. API\_ID и API\_HASH - учетные данные для Telegram API
2. SESSION\_NAME - имя файла сессии
3. MESSAGES\_LIMIT - количество загружаемых сообщений
4. DIALOGS\_LIMIT - количество загружаемых диалогов
5. MAX\_CACHE\_AGE - максимальное время жизни кеша в минутах
6. DOWNLOAD\_DIR - директория для загрузки медиа-файлов

**Алгоритм работы**

**1. Запуск приложения**

1. Запуск главного модуля app.py
2. Инициализация компонентов:
   * Загрузка конфигурации (config.py)
   * Инициализация API-клиента (api/client.py)
   * Инициализация менеджера состояний (state.py)
   * Инициализация CLI-интерфейса (cli.py)
   * Инициализация кеш-менеджера (services/cache.py)
   * Инициализация обработчика медиа (services/media.py)
   * Инициализация асинхронного загрузчика (utils/async\_loader.py)
3. Проверка авторизации:
   * Если сессия существует, используется она
   * Если сессии нет, запрашивается авторизация:
     + Запрос номера телефона
     + Отправка кода авторизации
     + Ввод кода и завершение авторизации

**2. Основной цикл работы**

1. Инициализация консольного интерфейса curses
2. Установка начального состояния (список диалогов)
3. Запуск асинхронного цикла обработки событий:
   * Обработка пользовательского ввода
   * Обновление данных
   * Отрисовка интерфейса
   * Обработка асинхронных задач

**3. Отображение списка диалогов**

1. Загрузка списка диалогов:
   * Проверка кеша
   * Если кеш актуален, использование данных из кеша
   * Если кеш устарел, асинхронная загрузка свежих данных
   * Сохранение в кеш
2. Отображение списка:
   * Если выбран режим "Только непрочитанные", фильтрация диалогов
   * Отрисовка заголовка и строки статуса
   * Отрисовка диалогов с индикацией непрочитанных сообщений
   * Выделение выбранного диалога
   * Отрисовка индикатора прокрутки
3. Обработка действий:
   * При нажатии Tab - переключение режима отображения
   * При нажатии ↑/↓ - навигация по списку
   * При нажатии → - открытие выбранного диалога
   * При нажатии Q - завершение работы

**4. Отображение чата**

1. Загрузка сообщений для выбранного диалога:
   * Проверка кеша
   * Если кеш актуален, использование данных из кеша
   * Если кеш устарел, асинхронная загрузка свежих данных
   * Сохранение в кеш
2. Отображение сообщений:
   * Отрисовка заголовка с именем диалога
   * Отрисовка сообщений с информацией о отправителе и времени
   * Отображение медиа-информации для сообщений с медиа
   * Выделение выбранного сообщения
   * Отрисовка индикаторов прокрутки
3. Обработка действий:
   * При нажатии ↑/↓ - навигация по сообщениям
   * При нажатии ← - возврат к списку диалогов
   * При нажатии R - переход в режим ответа на выбранное сообщение
   * При нажатии S - сохранение медиа из выбранного сообщения
   * При нажатии Q - возврат к списку диалогов

**5. Ответ на сообщение**

1. Переход в режим ответа:
   * Выделение выбранного сообщения
   * Отображение строки ввода с подсказкой "reply>"
   * Включение отображения курсора
   * Включение режима ввода текста
2. Обработка ввода:
   * Ввод текста ответа
   * При нажатии Enter - отправка сообщения
   * При нажатии Esc - отмена ответа и возврат в режим чата
3. При отправке ответа:
   * Асинхронная отправка сообщения с цитированием
   * Отметка сообщения как прочитанного
   * Обновление списка сообщений
   * Возврат в режим чата

**6. Работа с медиа**

1. Отображение медиа-информации:
   * Распознавание типа медиа (фото, видео, документ и т.д.)
   * Показ информации о размере, разрешении, длительности и т.д.
2. Сохранение медиа:
   * При нажатии S на сообщении с медиа - асинхронная загрузка
   * Генерация имени файла в зависимости от типа медиа
   * Сохранение в указанную директорию
   * Отображение пути к сохраненному файлу
3. Открытие медиа:
   * После сохранения - попытка открытия файла во внешней программе
   * Использование системных инструментов для открытия (xdg-open, open, startfile)

**7. Завершение работы**

1. При нажатии Ctrl+C или Q в режиме диалогов:
   * Отмена всех активных асинхронных задач
   * Сохранение кеша на диск
   * Корректное отключение от Telegram API
   * Восстановление нормального состояния терминала
   * Завершение работы программы

**Дополнительные алгоритмы**

**Кеширование данных**

1. При запросе данных:
   * Проверка наличия данных в кеше
   * Проверка времени последнего обновления кеша
   * Если кеш актуален, возврат данных из кеша
   * Если кеш устарел, загрузка свежих данных
2. При обновлении данных:
   * Сохранение данных в кеш
   * Обновление времени последнего обновления
3. При завершении работы:
   * Сохранение кеша на диск для использования при следующем запуске

**Асинхронная загрузка данных**

1. Создание асинхронной задачи:
   * Запуск корутины в фоновом режиме
   * Регистрация коллбэков для успешного завершения и ошибок
2. Обработка завершения задачи:
   * В случае успеха - вызов коллбэка с передачей результата
   * В случае ошибки - вызов коллбэка обработки ошибок
3. В основном цикле:
   * Проверка наличия завершенных задач
   * Выполнение их коллбэков
   * Отображение индикатора загрузки, пока есть активные задачи