**Документация по приложению «Чат»**  
Это клиент-серверное чат-приложение, позволяющее пользователям обмениваться сообщениями в реальном времени, как в публичном, так и в приватном режиме. Приложение хранит историю сообщений в базе данных, поддерживает экспорт чата, уведомления и управление соединением.  
  
  
**1. Функциональные возможности приложения:**  
  
**1.1 Обмен сообщениями:**  
  
**Публичный чат:** Пользователи могут отправлять сообщения, которые сохраняются в базе данных и транслируются всем подключенным клиентам.  
  
**Личные сообщения:** Поддерживается отправка личных сообщений через команду /pm <username> <сообщение>. Такие сообщения сохраняются отдельно и отображаются с отличительной разметкой.  
  
**1.2 Управление пользователями:**  
  
**Отображение списка онлайн-пользователей:** Клиент получает обновления о количестве и именах подключённых пользователей, а по клику выводится детальный список.  
  
**Система команд:** В клиенте реализованы команды для управления (например, /who для списка пользователей, /clear для очистки экрана, /help для справки).  
  
**1.3 Работа с историей сообщений:**  
  
**Хранение в БД:** Все публичные сообщения сохраняются в таблице chat\_messages, а личные – в таблице private\_messages с использованием SQLAlchemy и базы данных SQLite.  
  
**Экспорт чата:** Пользователь может сохранить историю сообщений в текстовый файл через функцию экспорта.  
  
**1.4 Пользовательский интерфейс (UI):**  
  
**Графический интерфейс на Tkinter:** Клиент использует окно с полем для ввода сообщений, область для отображения чата и панель с элементами управления.  
  
**Цветовая разметка сообщений:** Сообщения выводятся с различными стилями и цветами в зависимости от их типа (системные, личные отправленные/полученные, и публичные сообщения).  
  
  
**1.5 Управление соединением:**  
  
**Автоматическое переподключение:** При потере связи клиент автоматически пытается восстановить соединение.  
  
**Ручное управление соединением:** Пользователь может отключать и восстанавливать соединение с сервером по своему усмотрению.  
  
**1.6 Дополнительные функции:**  
  
**Звуковые уведомления:** При получении новых сообщений (особенно личных) проигрывается звуковой сигнал (функция winsound, для операционной системы Windows).

**Логирование событий:** Сервер ведёт подробное логирование с использованием библиотеки colorlog для цветного и информативного вывода.

**2. Требования для сборки приложения в исполняемый файл (.exe)  
  
Окружение:**

Python 3.x (рекомендуется версия 3.6 и выше).

Операционная система: Windows (учитывая использование winsound и сборку в .exe).

**Необходимые библиотеки и пакеты:**

Tkinter: Для создания графического интерфейса (обычно входит в стандартную поставку Python).

SQLAlchemy: Для работы с базой данных SQLite.

socket, threading, datetime: Стандартные библиотеки Python для сетевого взаимодействия и многопоточности.

winsound: Для звуковых уведомлений (только для Windows).

colorlog: Для цветного логирования на стороне сервера.

**Инструмент для сборки:**

PyInstaller: Позволяет собрать Python-приложение в один исполняемый файл (.exe).

**3. Процесс сборки приложения  
  
3.1 Установите необходимые пакеты с помощью pip (В терминале):**pip install sqlalchemy colorlog pyinstaller

**3.2 Убедитесь что файлы проекта находятся в одной директории. Это необходимо для корректной работы сборщика.  
  
3.3 Откройте терминал, в директории с исходными файлами. И выполните 2 команды:**pyinstaller -w --onefile client.py

pyinstaller --onefile server.py

Для остальных файлов database, models. Собирать их в exe файл не нужно! Так как PyInstaller автоматически их подтягивает в server.py, так как он использует данные файлы.

**После завершения, создастся папка “dist” в которой будут готовые файлы .exe для запуска, без зависимостей!  
  
  
4. Внутренняя архитектура и механизмы работы приложения  
  
4.1 Общая архитектура  
  
Клиентская часть (client.py):**  
  
Реализована на Tkinter, предоставляет графический интерфейс.

Использует сокеты для подключения к серверу.

Обрабатывает ввод пользователя и команды (например, отправка сообщений, личные сообщения через /pm, команды для вывода справки, очистки чата и т.п.).

Работает в многопоточном режиме: основной поток отвечает за UI, а отдельный поток – за получение сообщений от сервера.

Применяет цветовую разметку сообщений для удобства восприятия (системные, личные отправленные, личные полученные, сообщения пользователя и публичные сообщения).

**4.2 Серверная часть (server.py):**Запускается как отдельный процесс и принимает подключения от клиентов через TCP-сокеты.  
  
Создает для каждого нового клиента отдельный поток для обработки его сообщений.  
  
**Хранит словари:**  
  
Клиентский сокет <> имя пользователя.  
  
Имя пользователя <> смещение часового пояса (для локализации времени).

Отправляет клиенту историю публичных сообщений из базы данных SQLite (таблица chat\_messages), используя SQLAlchemy.  
  
**Обрабатывает входящие сообщения:**  
  
Публичные сообщения: Сохраняются в БД, форматируются с учетом времени и рассылаются всем подключенным клиентам.  
  
Личные сообщения: Если сообщение начинается с команды /pm, сервер извлекает получателя, проверяет корректность команды, сохраняет сообщение в таблице private\_messages и отправляет его как отправителю, так и получателю с отличительной маркировкой.  
  
Производит рассылку служебных сообщений (например, при подключении/отключении пользователя) и обновляет информацию о текущем числе онлайн-пользователей.  
 **4.3 Детали коммуникации и обработки сообщений  
  
Начало сессии:**Клиент при подключении отправляет строку, содержащую имя пользователя и смещение часового пояса.  
  
Сервер регистрирует пользователя, сохраняет смещение и сразу отправляет клиенту историю сообщений. **Форматирование времени:**Сервер использует UTC и преобразует временные метки в локальное время клиента на основе переданного смещения. Это позволяет корректно отображать время отправки сообщений для пользователей из разных часовых поясов.

**База данных:**Модуль database.py и модель models.py отвечают за подключение к базе данных SQLite и определение таблиц:  
  
chat\_messages: хранит публичные сообщения (имя пользователя, текст сообщения, временная метка).  
  
private\_messages: хранит личные сообщения (отправитель, получатель, текст сообщения, временная метка). **Обработка команд на клиенте:**Если сообщение начинается с символа /, клиент сначала пытается обработать его локально:  
  
Команда /clear очищает чат.

Команда /who выводит список текущих онлайн-пользователей.

Команда /help выводит справку по доступным командам.

Команда /pm обрабатывается клиентом как личное сообщение и пересылается серверу для дальнейшей маршрутизации. **Управление соединением:**Клиент имеет возможность вручную отключать или восстанавливать соединение. При потере связи запускается процесс автоматического переподключения с интервалом (например, 5 секунд).  
  
Сервер при обнаружении разрыва соединения удаляет соответствующий сокет из списка и информирует остальных пользователей.  
 **Логирование:**Сервер выводит подробную информацию о событиях (подключения, ошибки, отправка сообщений) с использованием библиотеки colorlog, что упрощает мониторинг работы приложения.