Описание архитектуры

1. Тип архитектуры

ПО имеет клиент-серверную модульную архитектуру. В минимальной конфигурации оно состоит из браузерного интерфейса, серверного веб-приложения и локальной базы данных. Необязательный шлюз моделей добавляется между веб-приложением и одним или несколькими серверами вывода.

Схема взаимодействия компонентов в минимальной конфигурации:

- Браузер пользователя взаимодействует с Обратным прокси по протоколу HTTPS.
- Обратный прокси передает запросы в Веб-приложение по протоколу HTTP на loopback-интерфейс.
- Веб-приложение сохраняет данные в локальной базе SQLite.
- Веб-приложение обращается к Серверу модели через OpenAI-совместимый HTTP API.

Схема взаимодействия компонентов в управляемой конфигурации:

- Веб-приложение обращается к Шлюзу моделей.
- Шлюз моделей управляет профилями запуска и передает запросы в Контейнерный движок вывода.
- Шлюз моделей реализует очередь запросов, переключение моделей и сбор телеметрии (с ускорителя или CPU).

2. Веб-приложение

2.1. Серверная часть
Серверная часть реализована на Python с использованием FastAPI и Uvicorn. Основные модули:

- app/config.py — загрузка настроек из переменных окружения;
- app/auth.py — пароли, сессии, CSRF и ограничение попыток входа;
- app/db.py — схема и операции SQLite;
- app/llm.py — клиент OpenAI-совместимого API и разбор потоковых ответов;
- app/main.py — HTTP API приложения, поток событий в браузер, административные методы;
- app/cli.py — консольное управление учетными записями.

2.2. Клиентская часть
Клиентская часть находится в каталоге static/ и состоит из HTML, CSS и JavaScript без внешних библиотек, CDN и этапа сборки. Она реализует страницу входа, список чатов, просмотр сообщений, отправку запроса, отображение потока и административную страницу.

3. Хранилище

SQLite работает в режиме WAL. Основные таблицы базы данных:

- users — учетные записи, роли и хеши паролей;

- sessions — HMAC-хеши токенов сессий и сроки действия;
- chats — принадлежность, название и выбранная модель;
- messages — сообщения, рассуждения и метрики ответа;
- events — события входа, изменения пользователей и ошибки.

Связи пользователей, чатов и сообщений защищены внешними ключами с каскадным удалением.

4. Интерфейс сервера моделей

Веб-приложение использует следующие методы:

- GET /v1/models;
- POST /v1/chat/completions с потоковым ответом SSE.

Поддерживаются поле reasoning_content, обычное поле content, теги <think> и итоговое поле usage, если сервер их передает. Для работы нужен сервер, совместимый именно с используемой частью интерфейса; формальная совместимость со всеми продуктами, использующими обозначение OpenAI API, не заявляется.

5. Шлюз моделей

Шлюз реализован на стандартной библиотеке Python и не требует Python-пакетов веб-приложения. Основные модули:

- gateway/httpserver.py — асинхронный HTTP/1.1-сервер;
- gateway/proxy.py — потоковая передача запроса и ответа;
- gateway/scheduler.py — конечный автомат, очереди и переключение моделей;
- gateway/backends/ — драйверы запуска серверов вывода;
- gateway/telemetry.py — сменные адаптеры телеметрии памяти;
- gateway/metrics.py — метрики Prometheus;
- gateway/config.py — настройки и загрузка профилей моделей;
- gateway/models.json — проверенный профиль AC700/tecovLLM.

Состояния планировщика: IDLE, READY, DRAINING, SWITCHING, FAILED.
Профиль модели задает идентификатор, драйвер запуска, контейнер, образ, устройства, тома, переменные окружения, команду, степень параллелизма, ожидаемый объем памяти, таймаут готовности и порт. Аппаратные пути находятся в профиле, а не в логике веб-приложения.

6. Адаптеры

6.1. Драйвер запуска
Реализован драйвер docker-vllm, формирующий команду docker run из профиля. Для тестов используется драйвер fake. Добавление другой среды вывода требует нового класса, реализующего общий интерфейс запуска, остановки, проверки процесса и готовности.

6.2. Телеметрия
Реализованы следующие адаптеры:

- teco-smi — для проверенной конфигурации AC700;
- fake — для автоматических тестов;
- none — для конфигураций без поддержанного аппаратного монитора.

Режим none означает отсутствие контроля физического освобождения памяти самим шлюзом и должен применяться только при наличии внешнего контроля ресурсов.

7. Границы безопасности

- Браузер взаимодействует только с веб-приложением.
- Сервер модели и административный API шлюза в проверенной конфигурации слушают loopback.
- Публичный обратный прокси закрывает внутренние пути /healthz, /metrics и /v1/*.
- Изменяющие запросы веб-приложения требуют CSRF-токен.
- Административные методы шлюза требуют Bearer-ключ, если он настроен; без ключей административный API шлюза отключен.
- Секреты передаются через файлы окружения и не должны храниться в Git.

8. Внешние связи

При штатной работе с локальной моделью ПО не требует соединения с иностранными серверами, удаленного управления или облачного API. Сетевой доступ может требоваться отдельно для первоначального получения зависимостей, контейнерных образов, моделей и TLS-сертификата. Эти материалы могут быть предварительно перенесены в закрытый контур.

Обновление ПО выполняется системным администратором. Механизма принудительного обновления из внешней сети в собственном коде нет.