SOFTWARE ARCHITECTURE DESCRIPTION

Information item: **Software Architecture Description**. Content and viewpoints are informed by **ISO/IEC/IEEE 42010** (architecture description) and **ISO/IEC/IEEE 15289** (life-cycle information items). User/operator documentation is published separately with reference to **ISO/IEC 26514**. This wording does not claim certification or verified conformity. Print layout targets A4 for reproducible distribution.

на программное обеспечение «CoronerChat»

Версия изделия: 2.7.10 (`package.json` / GitHub Latest)

Аннотация

К программе (изделию)

CoronerChat 2.7.10 — прикладное программное обеспечение для стримеров и модераторов: объединение чатов Twitch, VK Video Live, Kick, YouTube Live, Rutube, DonationAlerts и MemeAlerts в единую ленту с нормализацией формата сообщений, буфером и поиском; отображение во встроенном веб-интерфейсе и в режиме оверлея OBS; авторизация OAuth 2.0 (где применимо); каталоги эмодзи/бейджей (в т.ч. 7TV); OBS WebSocket; UX-пакет эфира (пресеты, алерты, goal/vote); локальная статистика сессий и достижения; локализация UI (ru/en/ro/de/fi); автообновление с GitHub Latest (`dorimeryt-alt/CoronerChat`); релей «now playing»; desktop Electron и опциональный zapret. Поставка: NSIS Setup / portable, каталог данных `CoronerChat-data`.

К настоящему документу

Настоящее техническое описание содержит сведения об архитектуре ПО «CoronerChat»: структурную модель в нотации IDEF0, контекстные и декомпозированные диаграммы потоков данных (DFD), описание взаимодействия с каждой внешней платформой и смежными HTTP-сервисами, перечень хранилищ данных, диаграммы процессов (BPMN) и последовательностей (UML), состав программных модулей, группы маршрутов HTTP и типы сообщений WebSocket, нефункциональные требования и словарь данных; в разделе 1 приведён **перечень реализованных функций изделия** (п. 1.5). Диаграммы, выполняемые средствами Mermaid в среде браузера, приведены для наглядности и сопровождения кода. Документ предназначен для разработчиков, системных аналитиков и лиц, ответственных за сопровождение комплекта программной документации. Эксплуатационная документация для оператора изложена в `docs/coronerchat-operation-print.html` и на сайте (`docs/guide.html`) в соответствии с ISO/IEC 26514.

Ключевые слова: Software Architecture Description; ISO/IEC/IEEE 42010; CoronerChat 2.7.10; IDEF0; DFD; BPMN; UML; WebSocket; OAuth 2.0; Twitch; VK; Kick; YouTube; Rutube; DonationAlerts; MemeAlerts; i18n; auto-update; Electron; Node.js.

Нормативные и справочные ссылки

- ISO/IEC/IEEE 42010 — Systems and software engineering — Architecture description.
- ISO/IEC/IEEE 15289 — Content of life-cycle information items (software documentation).
- ISO/IEC 26514 — Requirements for designers and developers of user documentation.
- ISO/IEC 12207 — Software life cycle processes (supply, operation, maintenance).
- ISO/IEC 25010 — Systems and software Quality Requirements and Evaluation (SQuaRE).

- ISO/IEC 27001 / 27002 — Information security management (controls mapping; see Security statement).
- ECMA-262 (JavaScript), WHATWG HTML Living Standard, WebSocket Protocol RFC 6455.
- Спецификации внешних систем: Twitch IRC / EventSub / Helix, VK Video Live API, Kick Chat, DonationAlerts, MemeAlerts, YouTube Data API / Google OAuth 2.0, 7TV (по используемым в коде конечным точкам).
- Исходный код изделия: каталоги `src/server/`, `src/providers/`, `src/electron/`, `package.json`.

Термины, определения и сокращения

Термин / сокращ.	Определение
ПО, изделие	Программное обеспечение CoronerChat — сервер чата и UI (веб и Electron).
DFD	Data Flow Diagram — диаграмма потоков данных.
ICOM	В IDEF0: Input, Control, Output, Mechanism — входы, управление, выходы, механизмы.
OAuth 2.0	Протокол делегирования доступа; используется для Twitch, VK, Kick, DonationAlerts, YouTube (Google).
WS	WebSocket — двунаправленный канал между браузером и `ChatAppServer` по пути `/ws`.
A1–A6	Декомпозиция контекстной функции A-0 по IDEF0 в настоящем документе.
Helix	REST API Twitch.
EventSub	Подсистема подписки на события Twitch (в т.ч. баллы канала, Hype Train).
DA	DonationAlerts.
MA	MemeAlerts.

CoronerChat — Software Architecture Description (v2.7.10)

Содержание

1. Общие положения

1.1. Назначение изделия

Единый клиент для приёма, нормализации, буферизации и отображения чата с нескольких платформ (Twitch, VK Video Live, Kick, YouTube Live, Rutube, DonationAlerts, MemeAlerts), а также: OAuth, прокси ассетов и каталогов эмодзи (7TV), OBS WebSocket, UX-пакет эфира, локальная статистика и достижения, локализация UI, автообновление с GitHub Latest, релей now playing, опционально zapret на desktop, оверлей и планшетный режим через общий HTTP+WS сервер.

1.2. Назначение документа

Формализация архитектуры для сопровождения, обучения и согласования изменений; договорённость о границах системы и внешних интерфейсах.

1.3. Область применения документа

Разработка, сопровождение и выпуск CoronerChat. Эксплуатация оператором — по `docs/coronerchat-operation-print.html` и `docs/guide.html` (ISO/IEC 26514).

1.4. Границы системы

В составе ПО: процесс Node.js (`ChatAppServer`), встроенный web-ui, провайдеры в `src/providers/`, при desktop — оболочка Electron.

Вне ПО: внешние платформы (Twitch, VK Video Live, Kick, DonationAlerts, MemeAlerts, YouTube, Rutube, 7TV/CDN), OBS Studio, GitHub Releases (канал обновлений), ОС, браузер пользователя при внешней авторизации, сеть и политики DPI (zapret).

1.5. Перечень реализованных функций изделия

Ниже приведена **характеристика реализованного функционала** текущей версии CoronerChat: перечень не претендует на исчерпание всех пунктов UI, а фиксирует основные возможности, подтверждаемые кодом сервера и встроенного клиента.

1.5.1. Режимы поставки и запуска

- режим **web**: запуск ядра Node.js, встроенный web-ui по HTTP, WebSocket `/ws`;
- режим **desktop**: оболочка Electron + Chromium, доступ к локальному серверу ядра;
- сборки под Windows: переносимый исполняемый файл, распакованный каталог, установщик NSIS с сохранением пользовательских данных при обновлении (политика инсталлятора).

1.5.2. Приём и объединение чатов нескольких платформ

- **Twitch**: чтение чата по IRC (включая сценарии анонимного просмотра и чтения от имени авторизованного пользователя); параллельно — события **баллов канала** и **Hype Train** через EventSub; при ограничениях EventSub — опрос наград через Helix (poller); реконнект и диагностика транспорта в `state.twitchTransport`;
- **VK Video Live**: приём сообщений по WebSocket API; каталог и прокси бинарных ресурсов;
- **Kick**: приём сообщений чата по WebSocket при заданном канале и токене OAuth;
- **DonationAlerts**: получение событий донатов по HTTP long-poll;
- **MemeAlerts**: подключение к шлюзу по WebSocket с нормализацией JWT из OBS-ссылки; управление connect/disconnect из UI;

- **YouTube Live**: получение сообщений живого чата через YouTube Data API v3 (ключ API или OAuth Google) либо browser-chat режим;
- **Rutube**: чтение чата эфира через публичный poll API; отправка — cookie-сессия (встроенное окно входа desktop); маршруты `/api/settings/rutube`, `/api/auth/rutube/*`;

1.5.3. Нормализация, буфер и доставка в интерфейс

- единая точка слияния потоков в `ChatAppServer`: нормализация сообщений, дедупликация (в т.ч. redemption и hype), ограниченный буфер и история;
- рассылка клиентам по WebSocket с типами `snapshot`, `state`, `chat.message`, `chat.moderation`, `chat.annotation` и др.;
- встроенный **web-ui**: отображение объединённого чата, настроек, оверлея и планшетного режима, темы и масштабы отображения (см. `web-ui.js`).

1.5.4. OAuth 2.0 и локальное хранение учётных данных

- потоки авторизации для **Twitch**, **VK** (включая сценарии с браузером и implicit), **Kick**, **DonationAlerts**, **YouTube (Google)** через маршруты `/api/auth/...`;
- сохранение токенов и параметров сессии в каталоге данных приложения (`*-auth.json`, `app-state.json` и др.).

1.5.5. Действия в экосистеме Twitch (при наличии прав доступа токена)

- отправка сообщений в чат через Helix `Send Chat Message`;
- модерация: удаление сообщений, таймаут, бан — при наличии соответствующих прав и scopes;
- изменение **названия** и **категории** трансляции для собственного канала;
- поддержка ряда slash-команд из поля ввода (в т.ч. `/raid`, `/announce`, `/shoutout`, настройки скорости чата, shield mode и др. в объёме, реализованном сервером и API).

1.5.6. Прокси HTTP, каталоги эмодзи и бейджей

- кэшируемые каталоги и выдача ресурсов **7TV**, **Twitch** (бейджи, эмодзи, клипы и вспомогательные запросы), **VK** (каталог, asset) — для снижения проблем CORS и единообразного Referrer;
- prefetch и health-check для 7TV по маршрутам API изделия.

1.5.7. Интеграции в desktop-сборке

- телеметрия **OBS Studio** через WebSocket плагина;
- релей **now playing** от внешнего источника метаданных трека;
- опциональный запуск внешнего компонента **zapret** для обхода ограничений DPI (конфигурируемый сценарий).

1.5.8. Диагностика, журналирование и устойчивость

- расширенный `runtime.log` (HTTP, WebSocket, чат-пайплайн, авторизация, транспорты, исключения); маскирование секретов в логах;
- режим трассировки сессии (переменная окружения), метрики нагрузки и памяти в `state.debug`;
- ограничение размера буфера сообщений, единственный экземпляр desktop-приложения (single instance lock).

1.5.9. Эфир, статистика, локализация и обновления (2.7.x)

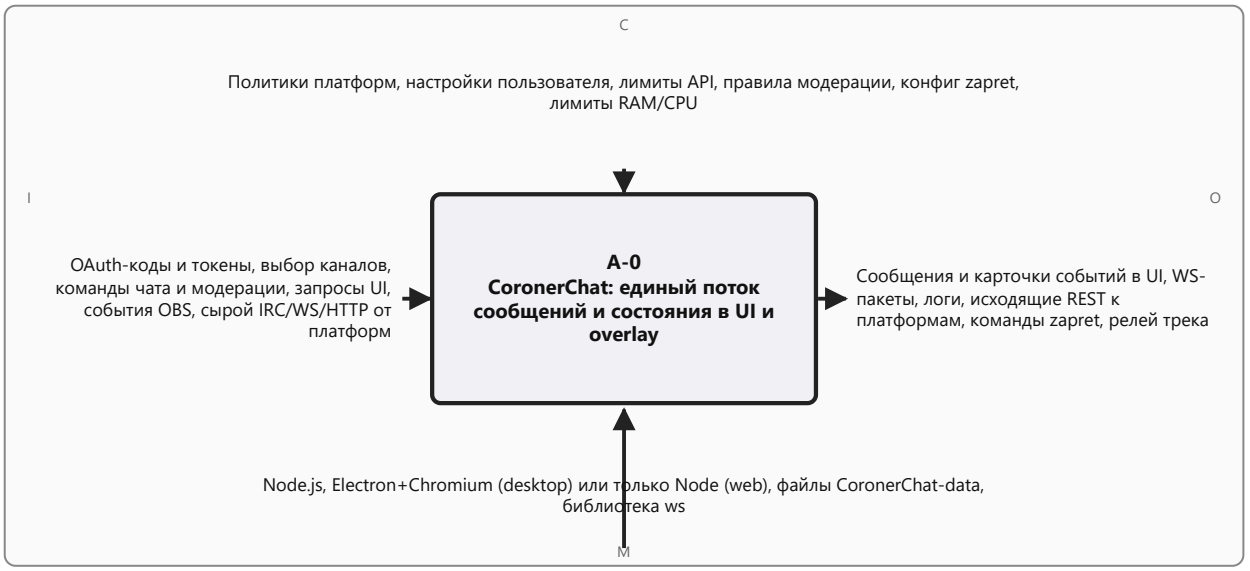
- **UX-пакет «Эфир»**: чеклист готовности, пресеты TW/Мульти/Полн., фокус-режим, favorites, keywords/цензор, soft-confirm, resend, song request, overlays `?alerts=1` / `?goal=1` / `?vote=1` / `?recap=1`;
- **Статистика и достижения**: локальная история эфиров, категории/игры, топы чата и донатов, полка достижений в приложении (не в OBS overlay);
- **i18n**: словари `src/server/i18n/locales/{ru,en,ro,de,fi}.json`, персист `uiLocale`, `POST /api/settings/locale`; applyI18nDom без сброса вложенных контролов (2.7.5);
- **Автообновление**: канал `dorimeryt-alt/CoronerChat`; `GET …/releases/latest`; установка Setup; skip-версия `POST /api/update/dismiss`; install `POST /api/update/install`; отсев legacy-тегов 2.25–2.35.

2. Структурная модель (IDEF0)

Модель приведена в нотации IDEF0 (ICOM). Диаграммы выполнены в виде векторной графики SVG.

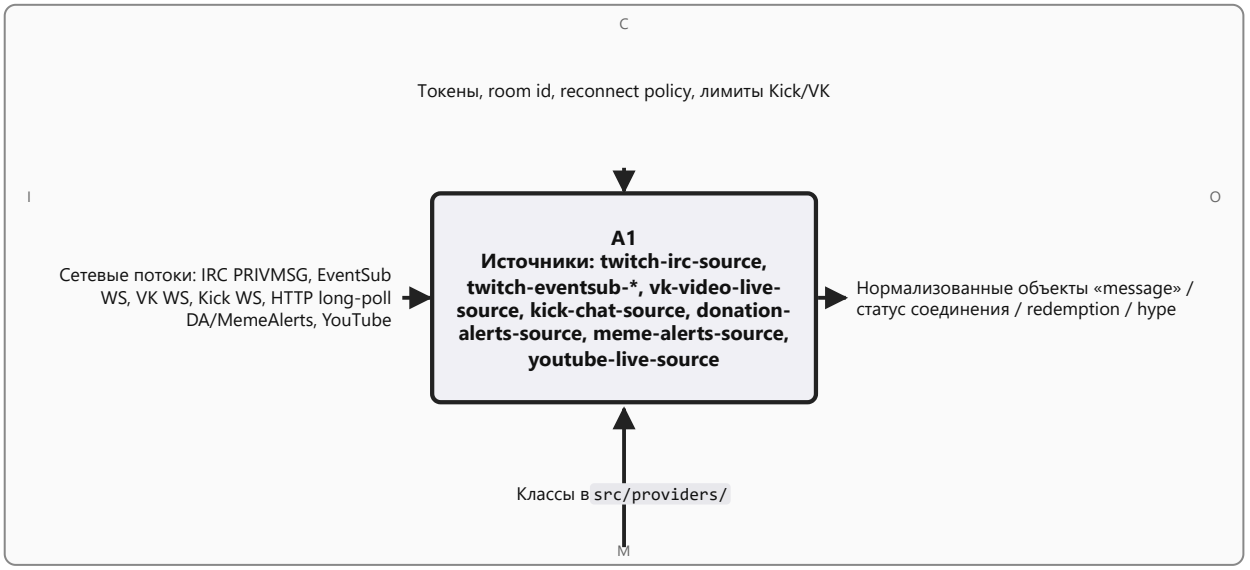
2.1. Контекст деятельности A-0

A-0. Агрегировать и показать мультиплатформенный чат и состояние стрима

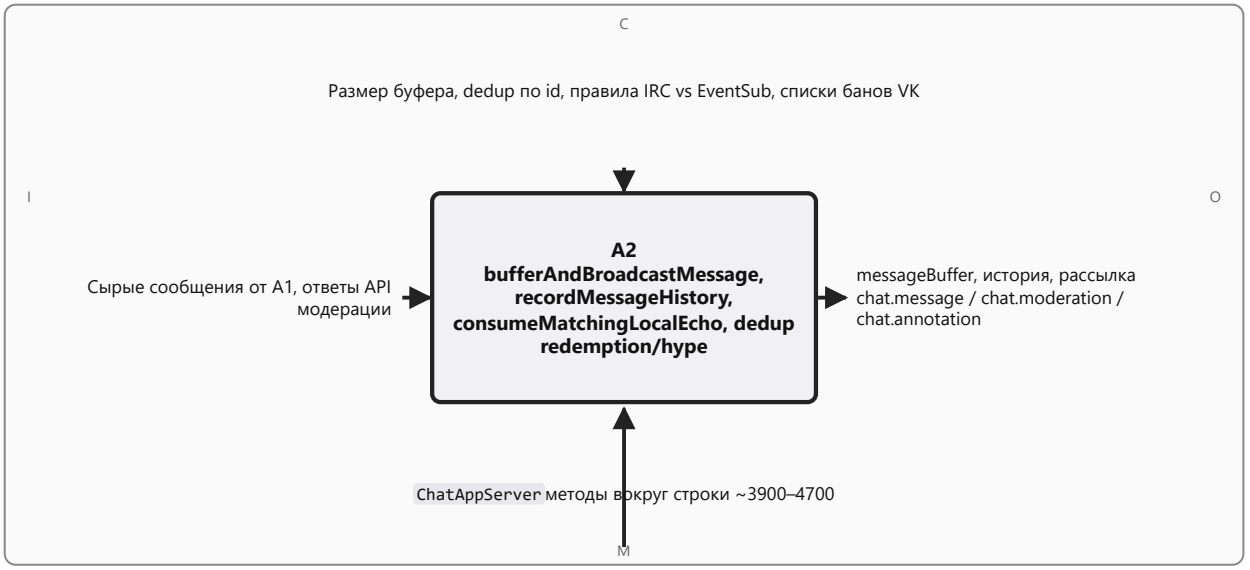


2.2. Декомпозиция процессов A1–A6

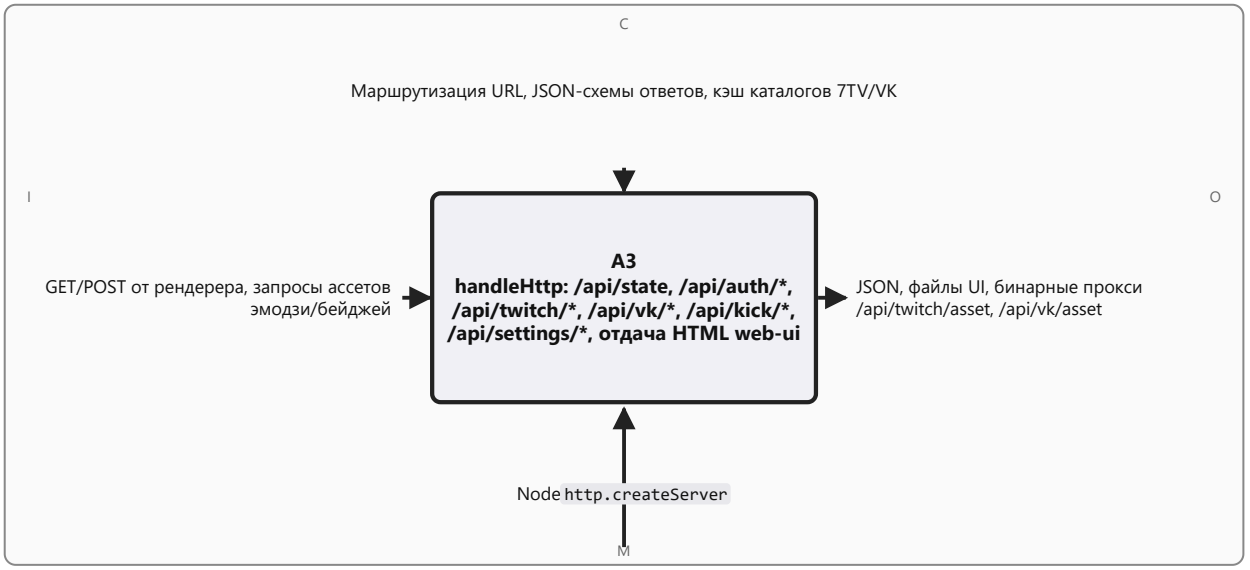
A1. Приём событий от внешних чатов



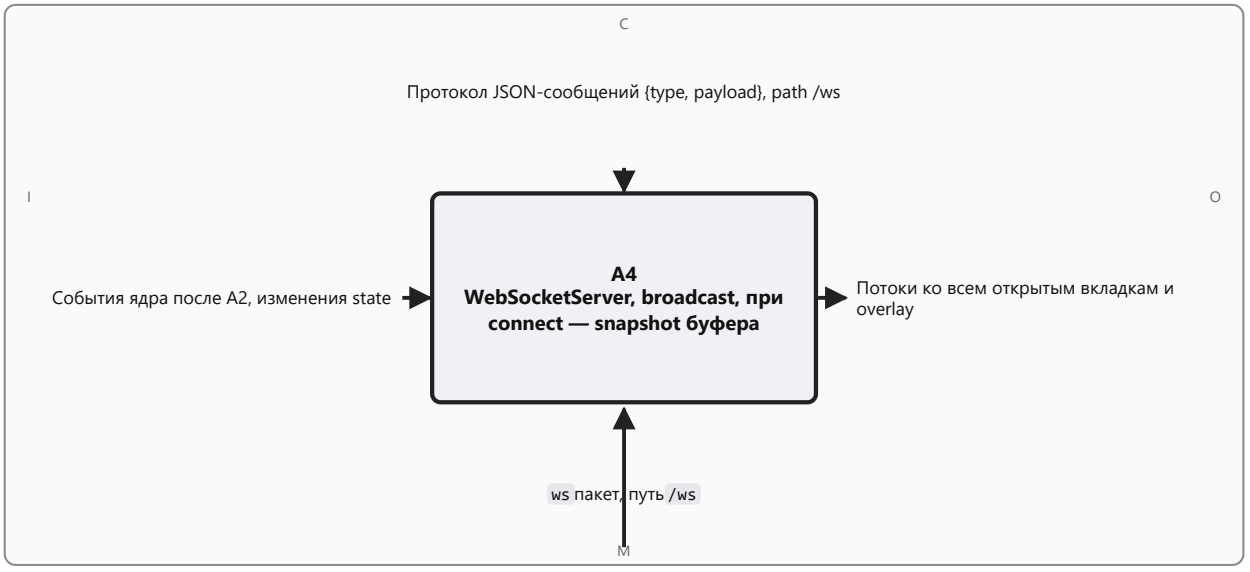
A2. Нормализация, дедупликация, буфер, модерация



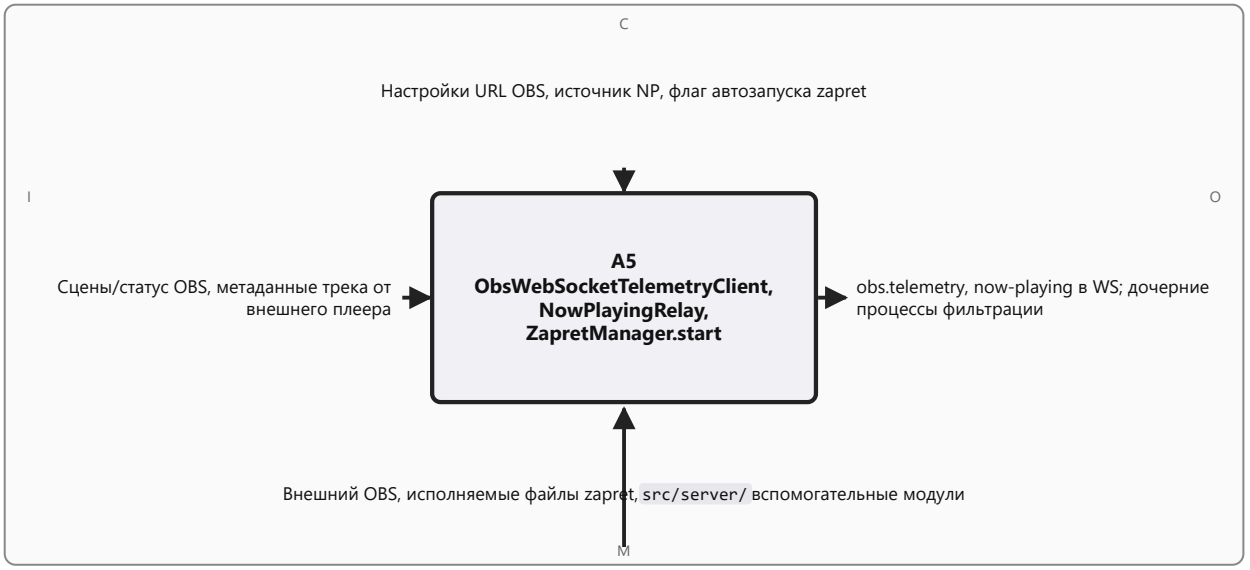
A3. HTTP: UI, REST, статика, прокси ассетов



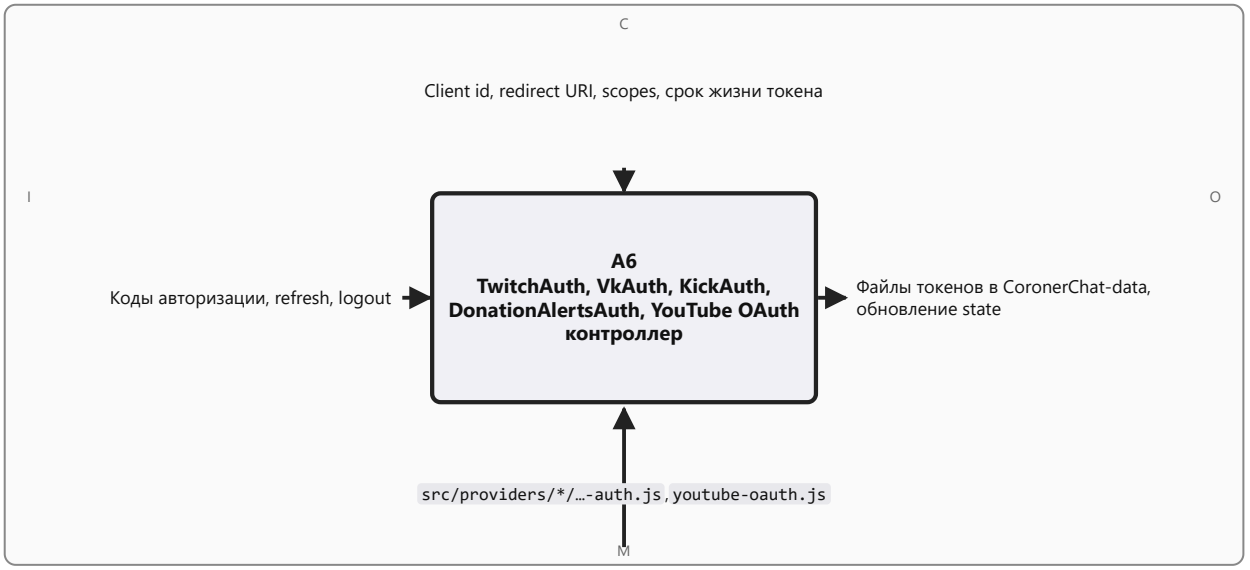
A4. WebSocket: снимок, поток сообщений, отладка



A5. OBS, now playing, zapret (desktop)



A6. Управление OAuth и сохранение сессий

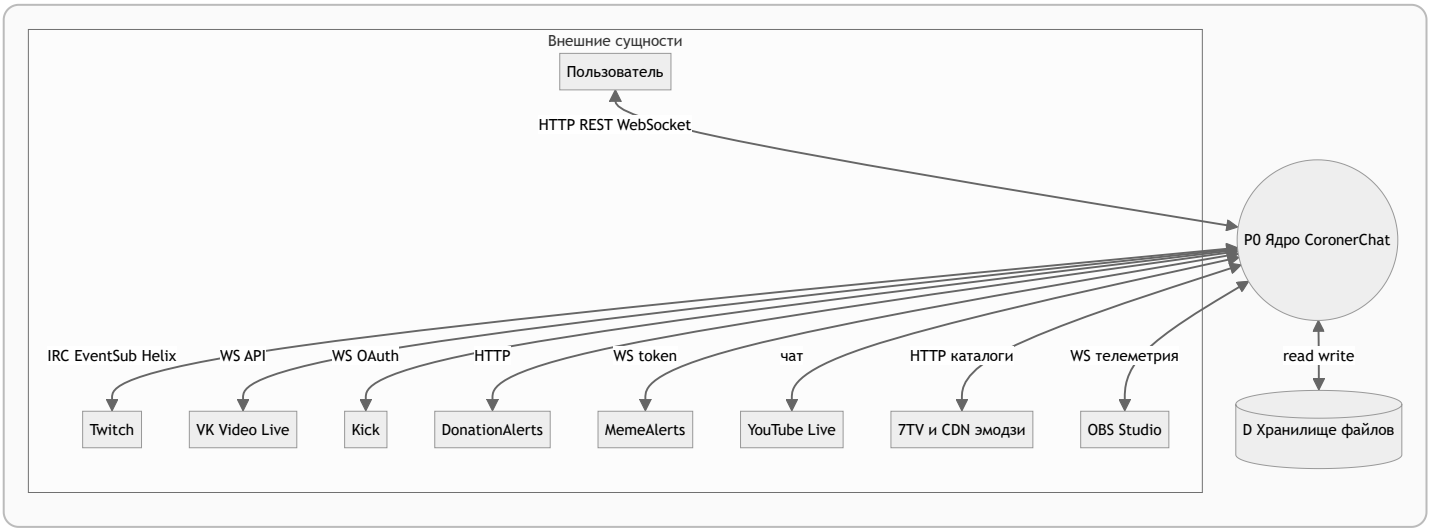


3. Состав программных модулей и соответствие процессам кодовой базе

Процесс	Главные артефакты	Назначение кратко
A1	`twitch-irc-source.js`, `twitch-eventsub-channel-points.js`, `vk-video-live-source.js`, `kick-chat-source.js`, …	Подключение к внешним транспортам, реконнект, парсинг в события
A2	`chat-app-server.js` (bufferAndBroadcastMessage, handleChannelPointsRedemption, handleHypeTrainEvent)	Единая точка входа сообщений в UI-поток
A3	`chat-app-server.js` handleHttp	REST и отдача встроенного web-ui
A4	`chat-app-server.js` wss.on("connection")	Подписка клиентов, snapshot
A5	ObsWebSocketTelemetryClient, NowPlayingRelay, ZapretManager	Внешние интеграции desktop
A6	`*-auth.js`, сохранение в каталог данных	Идентичность пользователя на платформах
UI	`web-ui.js` (встраивается сервером)	Рендер чата, настройки, overlay, WebSocket-клиент на `/ws`

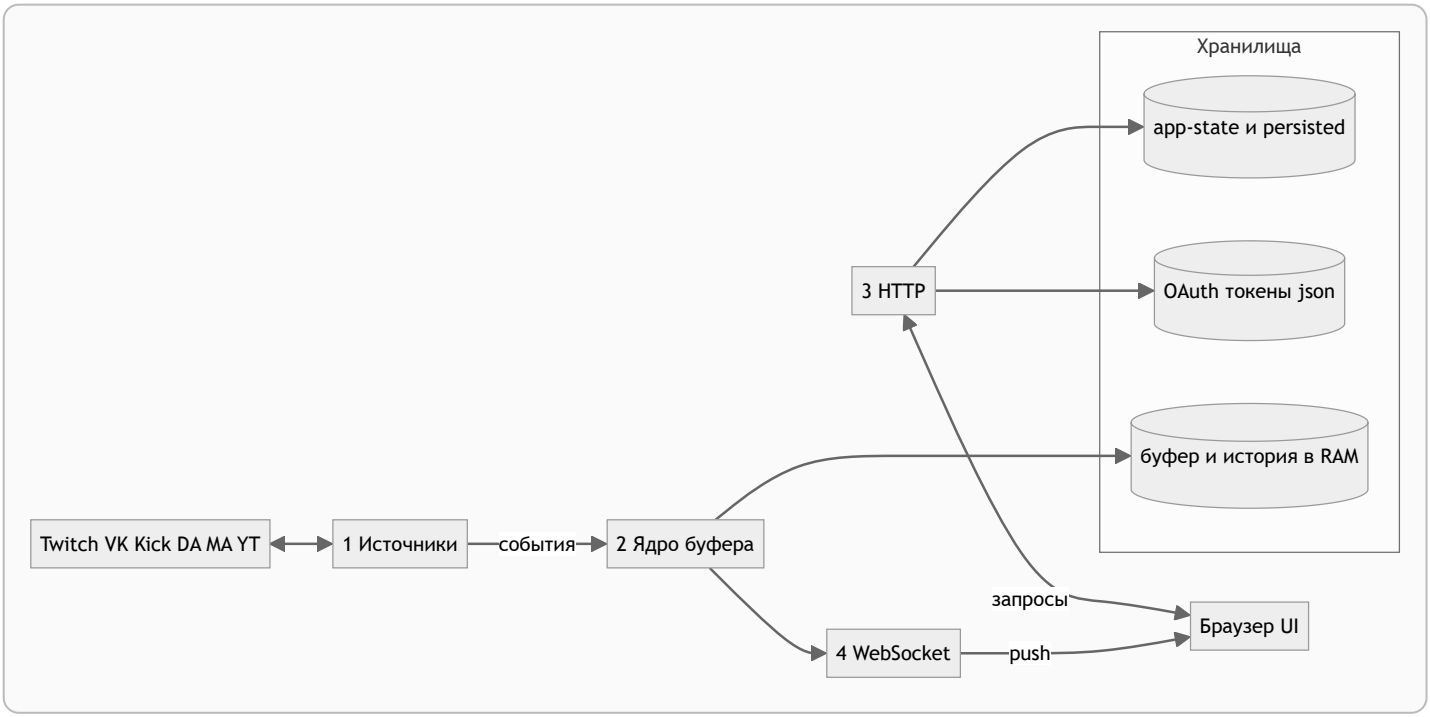
4. Контекстная диаграмма потоков данных (DFD, уровень 0)

Рисунок 4 — Контекстная DFD: изделие «CoronerChat» и внешние сущности



5. Диаграмма декомпозиции первого уровня (DFD-1)

Рисунок 5 — DFD первого уровня: источники, ядро, HTTP, WebSocket, хранилища



6. Описание взаимодействия с внешними платформами (DFD, уровень 2)

Все перечисленные ниже интеграции сходятся в единый процесс A2 — класс `ChatAppServer` (`src/server/chat-app-server.js`): нормализация сообщений, дедупликация, `messageBuffer`, рассылка клиентам по WebSocket. Отличия заключаются во внешнем транспорте, правилах аутентификации и составе типов событий.

6.1. Twitch

Назначение. Приём чата по IRC; дополнительно — события баллов канала и Hype Train через EventSub WebSocket и при необходимости нормализация наград через Helix REST (poller); доступ к API Twitch по OAuth.

Компоненты ПО.

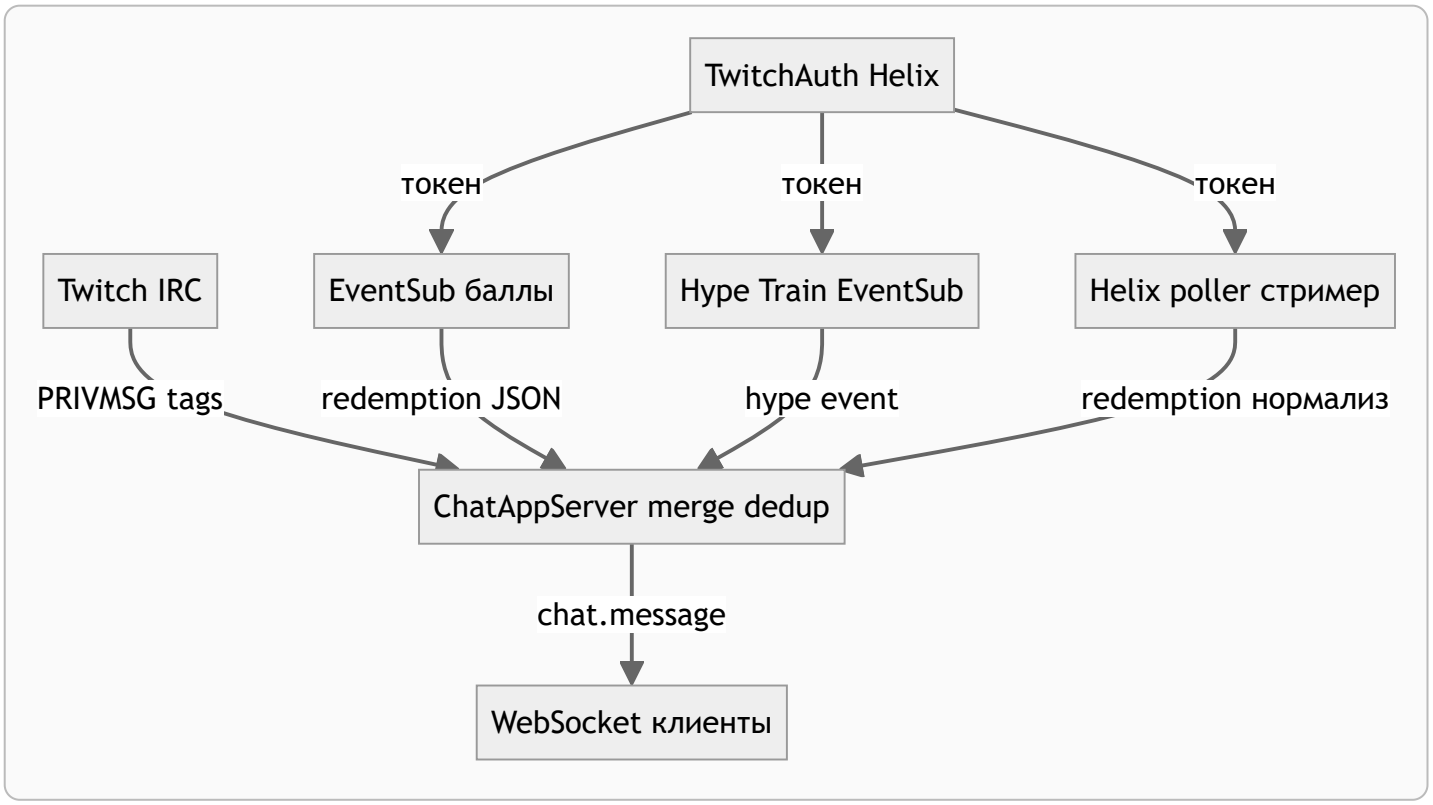
Компонент	**Файл (каталог `src/providers/twitch/`)**	**Роль**
TwitchIrcSource	`twitch-irc-source.js`	Подключение к IRC, PRIVMSG, теги, реконнект.
TwitchChannelPointsEventSub, TwitchHypeTrainEventSub, TwitchChannelPointsPoller	`twitch-eventsub-channel-points.js`	EventSub уведомления; поллер Helix при отсутствии EventSub для баллов.
TwitchAuth (Helix)	`twitch-auth.js`	OAuth, хранение токена, вызовы Helix.

Внешние интерфейсы. IRC (Twitch); HTTPS EventSub; HTTPS Helix (`api.twitch.tv`).

HTTP API изделия (фрагмент). `/api/auth/twitch/*`; `/api/twitch/badges`, `/api/twitch/emotes`, `/api/twitch/asset`, `/api/twitch/chatters`, `/api/twitch/clip` (см. реализацию `handleHttp`).

Состояние в `state`. `twitchTransport` (IRC, channelPoints, hypeTrain, сетевые подсказки).

Рисунок 6.1 — DFD-2: Twitch, несколько параллельных входов в слияние A2



6.2. VK Video Live

Назначение. Приём сообщений и служебных событий трансляции VK Video Live по WebSocket API.

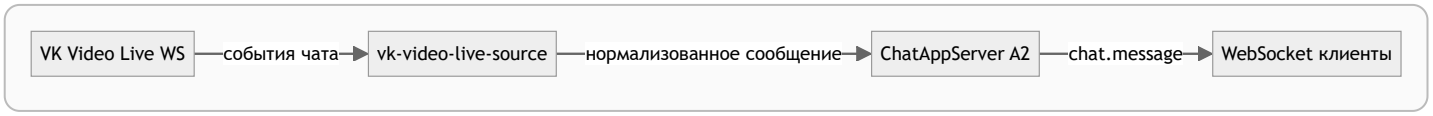
Компонент	Файл	Роль
Источник VK	`src/providers/vk/vk-video-live-source.js`	WS-клиент, парсинг событий в унифицированные сообщения.
VkAuth	`src/providers/vk/vk-auth.js`	OAuth / implicit, запись `vk-auth.json`.
Вспомогательные HTTPS-вызовы	`src/providers/vk/vk-https-utils.js`	Общие утилиты запросов к API VK.

Внешние интерфейсы. WebSocket и HTTPS API VK Video Live (см. актуальную документацию VK).

HTTP API изделия. `/api/auth/vk/*` (в т.ч. `start`, `poll`, `logout`, `callback`, `implicit-token`); `GET /api/vk/catalog`, `GET /api/vk/asset` (каталог и прокси бинарных ресурсов).

Состояние. `vkTransport` (реконнект, время последнего подключения, ошибки сети).

Рисунок 6.2 — DFD-2: VK Video Live → ядро



6.3. Kick

Назначение. Приём сообщений чата Kick по WebSocket; отправка сообщений при наличии прав — через REST API Kick.

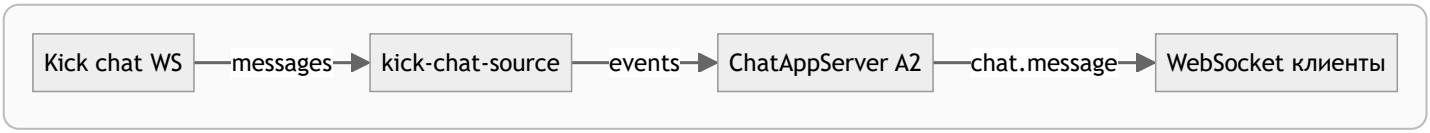
Компонент	Файл	Роль
KickChatSource	`src/providers/kick/kick-chat-source.js`	WS чата, интеграция с ядром.
KickAuthController	`src/providers/kick/kick-auth.js`	OAuth Kick, `kick-auth.json`.
Kick HTTP	`kick-http-public.js`, `kick-api.js`	Публичные данные канала, постинг сообщений.

Внешние интерфейсы. WebSocket чата Kick; HTTPS OAuth и API Kick.

HTTP API изделия. `/api/auth/kick/*`; `POST /api/settings/kick` (сохранение канала и перезапуск транспорта при необходимости).

Состояние. `state.kick`, `state.kickAuth` (снимок авторизации в ответах API).

Рисунок 6.3 — DFD-2: Kick → ядро



6.4. DonationAlerts

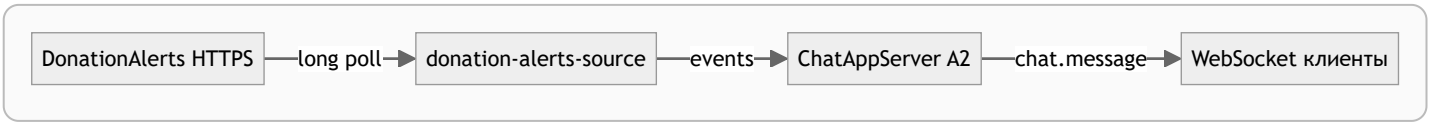
Назначение. Получение оповещений о донатах и связанных событиях через HTTP long-polling.

Компонент	Файл	Роль
DonationAlertsSource	`src/providers/donation-alerts/donation-alerts-source.js`	Long-poll цикл, преобразование в сообщения чата/алертов.
DonationAlertsAuth	`src/providers/donation-alerts/donation-alerts-auth.js`	OAuth, `donation-alerts-auth.json`.

Внешние интерфейсы. HTTPS API DonationAlerts (документация сервиса).

HTTP API изделия. `/api/auth/donation-alerts/*` (config, start, poll, logout, callback).

Рисунок 6.4 — DFD-2: DonationAlerts → ядро



6.5. MemeAlerts

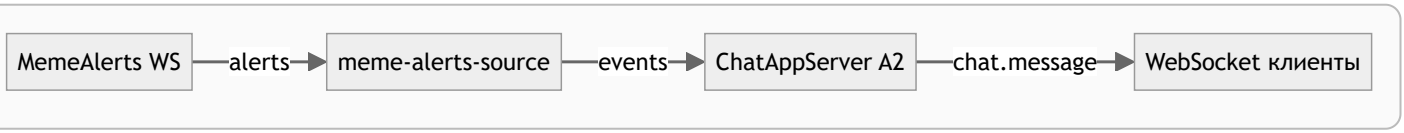
Назначение. Приём алертов MemeAlerts в реальном времени по Socket.IO с использованием JWT (`obsToken`) из «Ссылки для OBS».

Компонент	Файл	Роль
MemeAlertsSource	`src/providers/meme-alerts/meme-alerts-source.js`	Подключение к шлюзу MA, события в ядро.
Утилиты токена	`src/providers/meme-alerts/meme-alerts-token.js`	Разбор URL OBS / JWT `obsToken`, нормализация сохранённой строки.

Внешние интерфейсы. WebSocket MemeAlerts; пользователь вводит URL или JWT в настройках (персистится в `app-state`).

HTTP API изделия. `POST /api/meme-alerts/connect`, `POST /api/meme-alerts/disconnect` — управление сессией из UI.

Рисунок 6.5 — DFD-2: MemeAlerts → ядро



6.6. YouTube Live

Назначение. Получение сообщений живого чата трансляции YouTube через YouTube Data API v3 с авторизацией по API key или по OAuth 2.0 (Google).

Компонент	Файл	Роль
YouTubeLiveSource	`src/providers/youtube/youtube-live-source.js`	Опрос liveChatMessages, реконнект, дедуп по id.
YouTube OAuth	`src/providers/youtube/youtube-oauth.js`	Хранение `youtube-oauth.json`, обмен кодов.

Внешние интерфейсы. HTTPS `www.googleapis.com` (YouTube Data API).

HTTP API изделия. `/api/auth/youtube/google`, `/api/auth/youtube/logout`; настройки канала и ключа — через общие маршруты настроек и `state.youtube`.

Состояние. `state.youtube` (канал, ключ API, признаки OAuth).

Рисунок 6.6 — DFD-2: YouTube Live → ядро

6.6a. Rutube

Назначение. Чтение чата эфира Rutube через публичный poll API; отправка сообщений — через cookie-сессию (встроенное окно входа desktop).

Компонент	Файл	Роль
RutubeChatSource	`src/providers/rutube/rutube-chat-source.js`	Poll чата эфира, нормализация в единый формат.
Rutube HTTP	`src/providers/rutube/rutube-http.js`	HTTP-запросы к публичным API Rutube.
Rutube browser session	`src/providers/rutube/rutube-chat-browser-source.js`	Cookie-сессия для отправки (desktop).

Внешние интерфейсы. HTTPS публичный API Rutube (poll); cookie-сессия страницы эфира для send.

HTTP API изделия. `POST /api/settings/rutube`; `POST /api/auth/rutube/browser`, `POST /api/auth/rutube/refresh`.

Состояние. `state.rutube` (URL эфира, статус подключения, признак cookie-сессии).

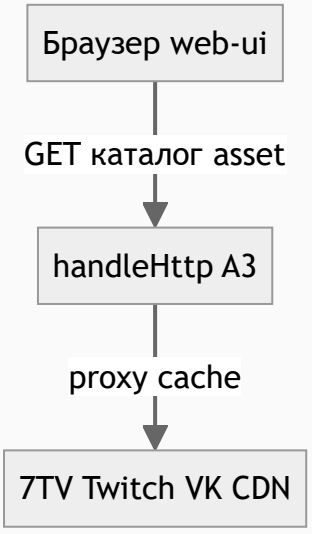
Рисунок 6.6а — DFD-2: Rutube → ядро

6.7. Смежные сервисы отображения (7TV, каталоги Twitch/VK, прокси)

Назначение. Не являются источниками текста чата, но обеспечивают каталоги эмодзи/бейджей и доставку бинарных ресурсов без CORS в браузере.

Направление	Примеры маршрутов изделия	Примечание
7TV	`/api/7tv/catalog`, `/api/7tv/health`, `/api/7tv/asset`, prefetch	Кэш на диске, см. `7tv-cache`.
Twitch	`/api/twitch/badges`, `/api/twitch/emotes`, `/api/twitch/asset`	Совместно с чатом Twitch.
VK	`/api/vk/catalog`, `/api/vk/asset`	Совместно с VK Video Live.

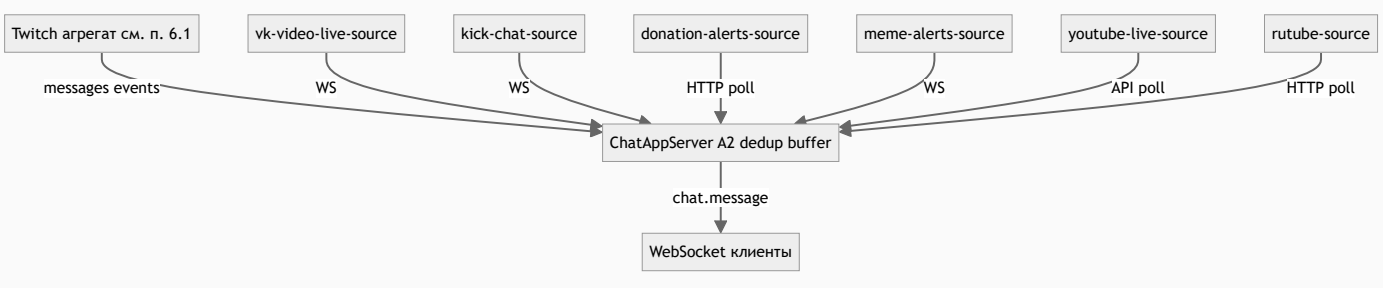
Рисунок 6.7 — Вспомогательные HTTP-потоки к CDN/каталогам (логическая связь с A3)



6.8. Сводная схема источников сообщений

На рисунке 6.8 показаны все основные источники, формирующие единый поток `chat.message` после обработки в A2 (детализация ветвления Twitch — на рисунке 6.1).

Рисунок 6.8 — Сводная DFD-2: платформы чата → ChatAppServer → клиенты



7. Хранилища данных

Таблица 7 — Персистентные и оперативные хранилища

Узел DFD	Типичный путь	Содержимое
D app-state	`CoronerChat-data/app-state.json` (рядом с exe или из `getCoronerChatDataDirectory`)	Каналы, настройки UI, флаги платформ, ревизии
D twitch-auth	`twitch-auth.json` и др. в том же каталоге	OAuth Twitch, scopes, user id
D vk-auth	`vk-auth.json`	Токены VK Video Live, сессия OAuth/implicit
D kick-auth	`kick-auth.json`	OAuth Kick, refresh
D donation-alerts-auth	`donation-alerts-auth.json`	Токен DonationAlerts для long-poll
D youtube-oauth	`youtube-oauth.json`	OAuth Google для чата YouTube, если используется
D кэши	`7tv-cache`, `twitch-badges-cache`, …	Каталоги эмодзи/бейджей, манифесты
D логи	`runtime.log`	Диагностика сервера
D RAM	`messageBuffer`, `messageHistory` в процессе	Окно последних сообщений для snapshot и поиска

8. Описание процессов (BPMN)

8.1. Сводная таблица потоков OAuth и настройки доступа по платформам

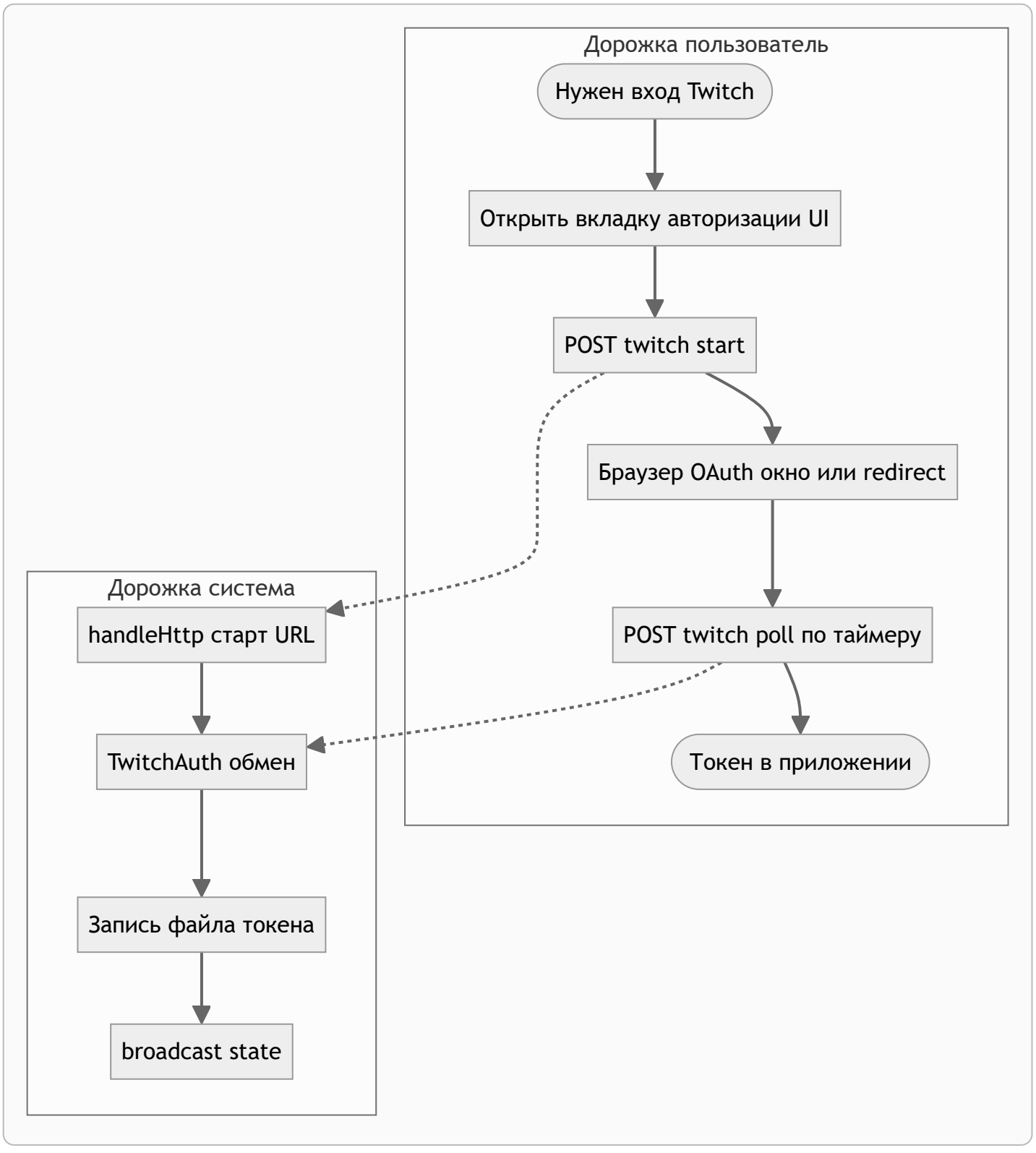
Таблица 8.1 — Соответствие платформ маршрутам `handleHttp` (фрагмент)

Платформа	Хвост пути после `/api/auth/`	Файл контроллера	Персистенция токена
Twitch	`twitch/client-id`, `start`, `poll`, `logout`	`twitch-auth.js`	`twitch-auth.json`
VK Video Live	`vk/config`, `start`, `poll`, `logout`, `browser`, `callback`, `implicit-token`, `start-implicit`	`vk-auth.js`	`vk-auth.json`
Kick	`kick/config`, `start`, `poll`, `logout`, `callback`	`kick-auth.js`	`kick-auth.json`
DonationAlerts	`donation-alerts/config`, `start`, `poll`, `logout`, `callback`	`donation-alerts-auth.js`	`donation-alerts-auth.json`
YouTube (Google)	`youtube/google`, `youtube/logout`	`youtube-oauth.js`	`youtube-oauth.json`
MemeAlerts	OAuth внешнего сервиса не используется	—	JWT в настройках приложения (`app-state`)

8.2. OAuth Twitch — полная цепочка (пример)

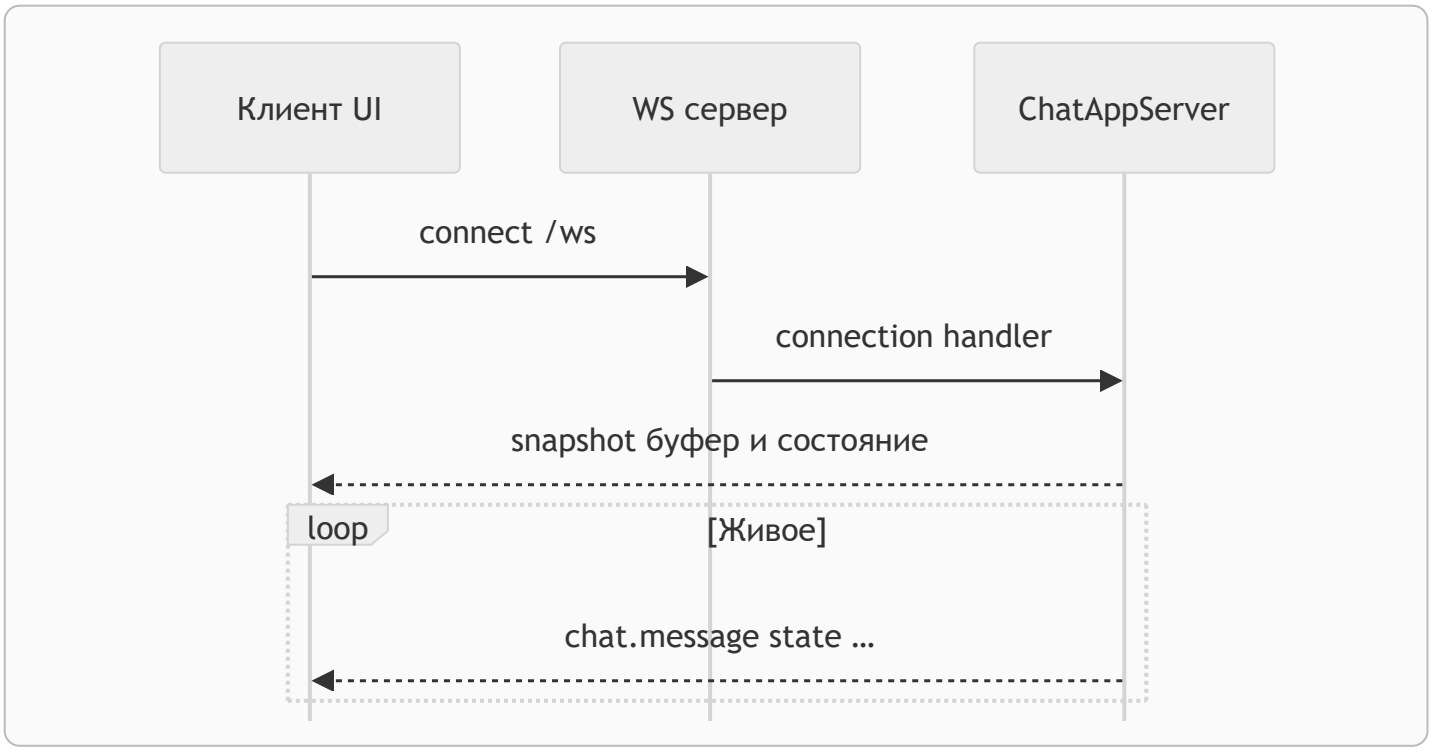
Наиболее разветвлённый сценарий UI↔сервер; для остальных платформ сохраняется логика «start → внешний браузер или окно → poll → запись json → broadcast state».

Рисунок 8.2 — BPMN: авторизация Twitch



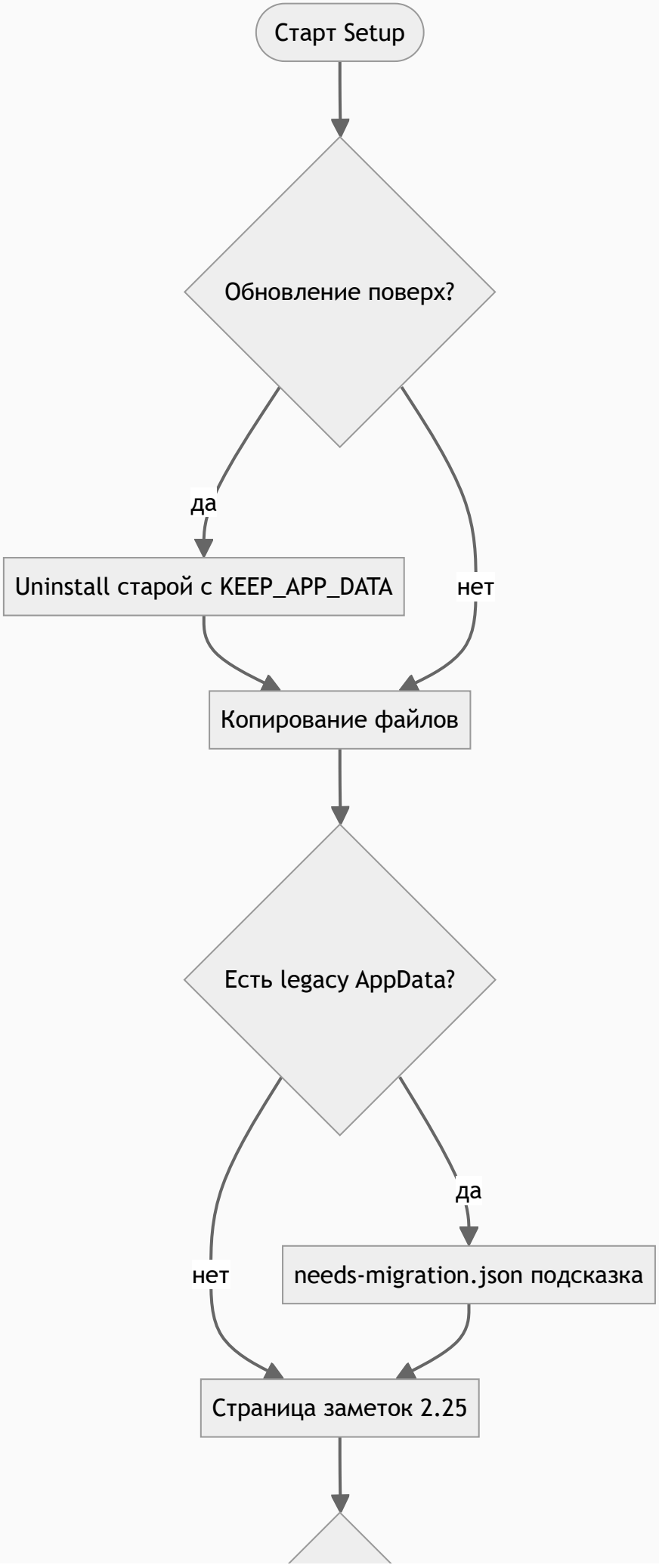
8.3. Первое подключение WebSocket клиента

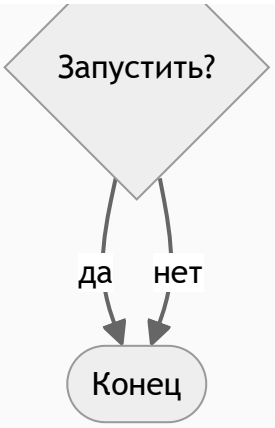
Рисунок 8.3 — BPMN (UML sequence): подключение к `/ws`



8.4. Установка NSIS (ветвления сценария)

Рисунок 8.4 — BPMN: установщик Windows

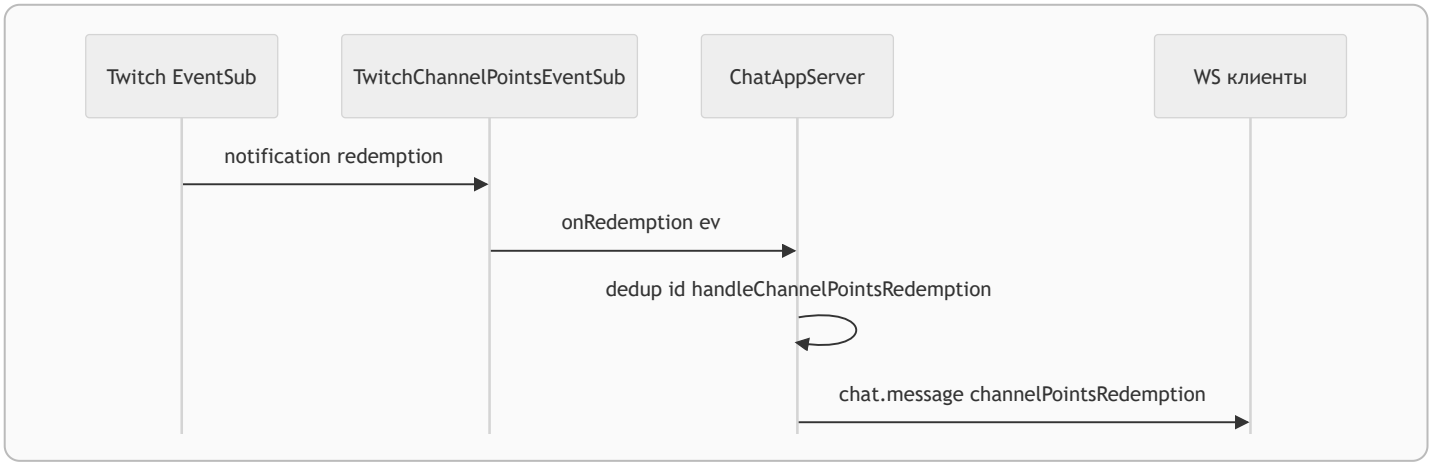

Старт Setup
Обновление поверх?
да
нет
Uninstall старой с KEEP_APP_DATA
Копирование файлов
Есть legacy AppData?
да
нет
needs-migration.json подсказка
Страница заметок 2.25



9. Диаграммы последовательности взаимодействия (UML)

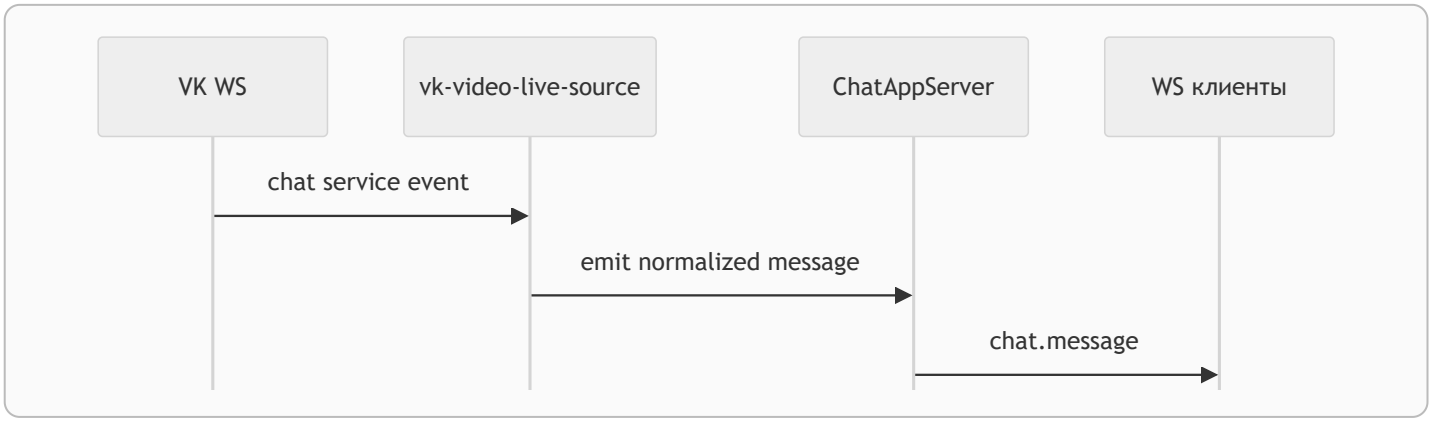
9.1. Twitch — награда баллов канала (EventSub)

Рисунок 9.1 — UML sequence: EventSub redemption → UI



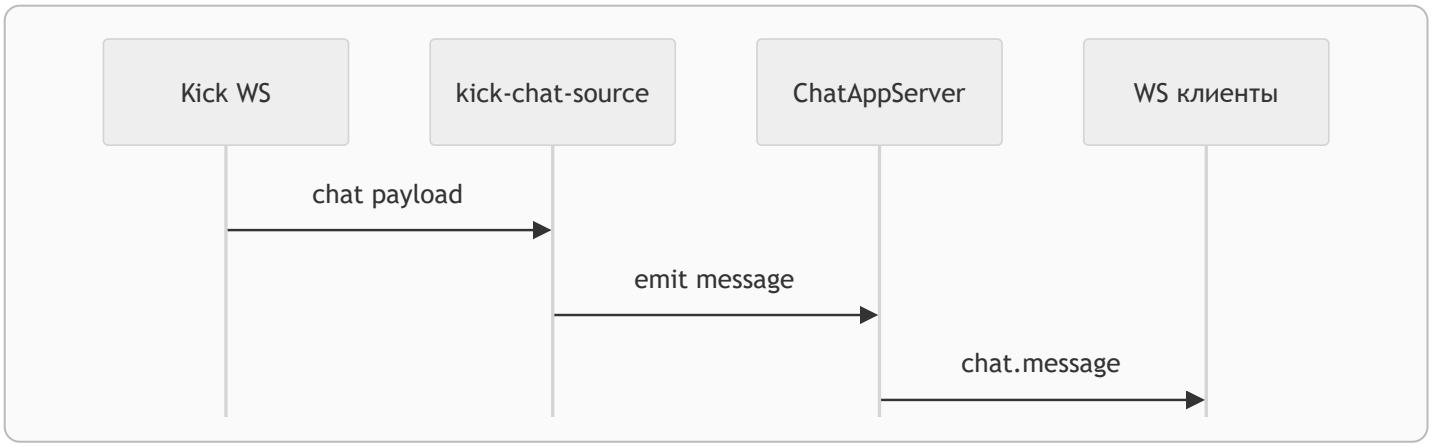
9.2. VK Video Live — сообщение чата

Рисунок 9.2 — UML sequence: упрощённый поток



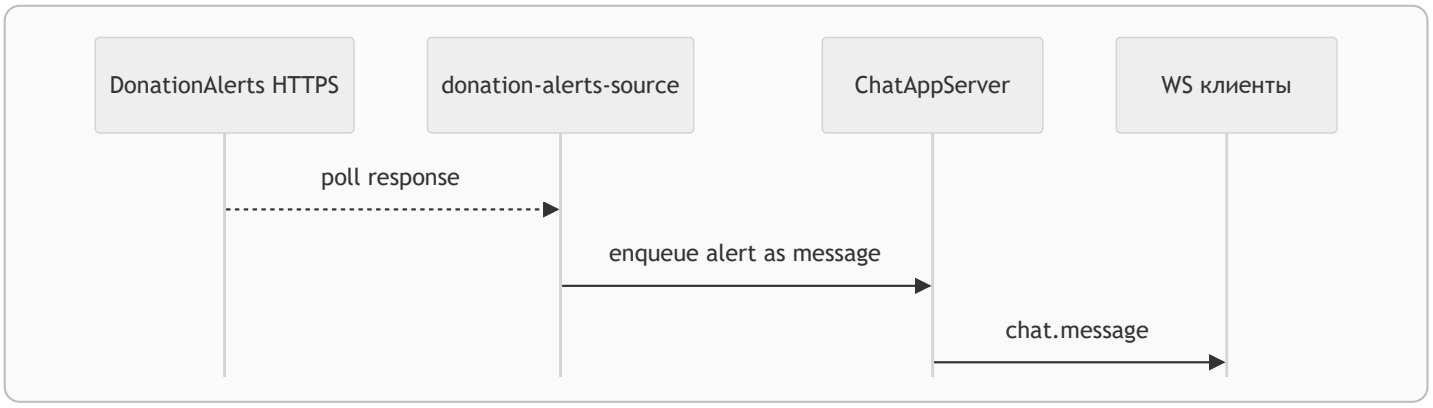
9.3. Kick — сообщение чата

Рисунок 9.3 — UML sequence: упрощённый поток



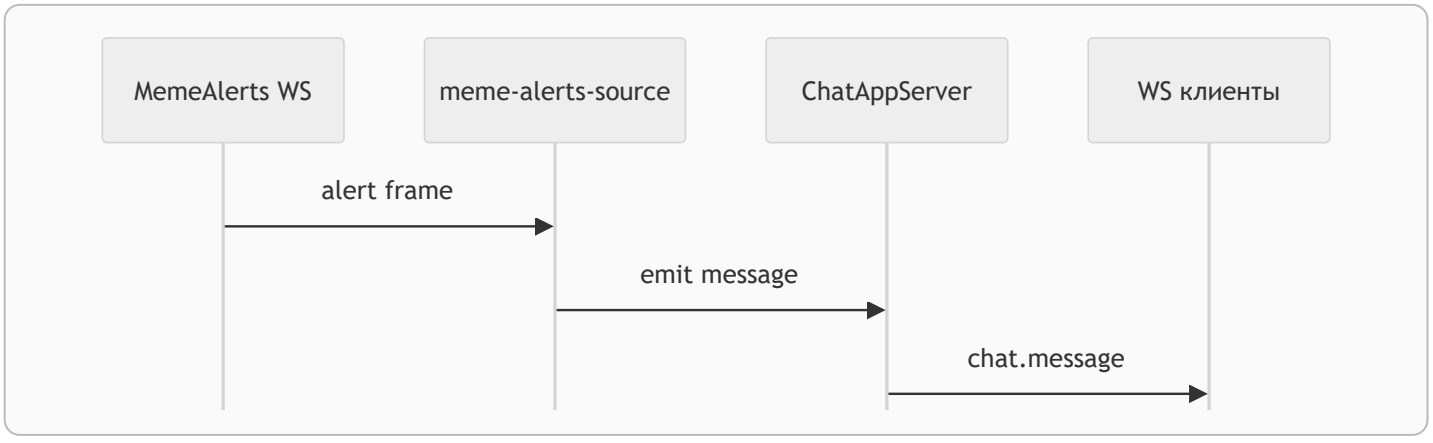
9.4. DonationAlerts — событие long-poll

Рисунок 9.4 — UML sequence: упрощённый поток



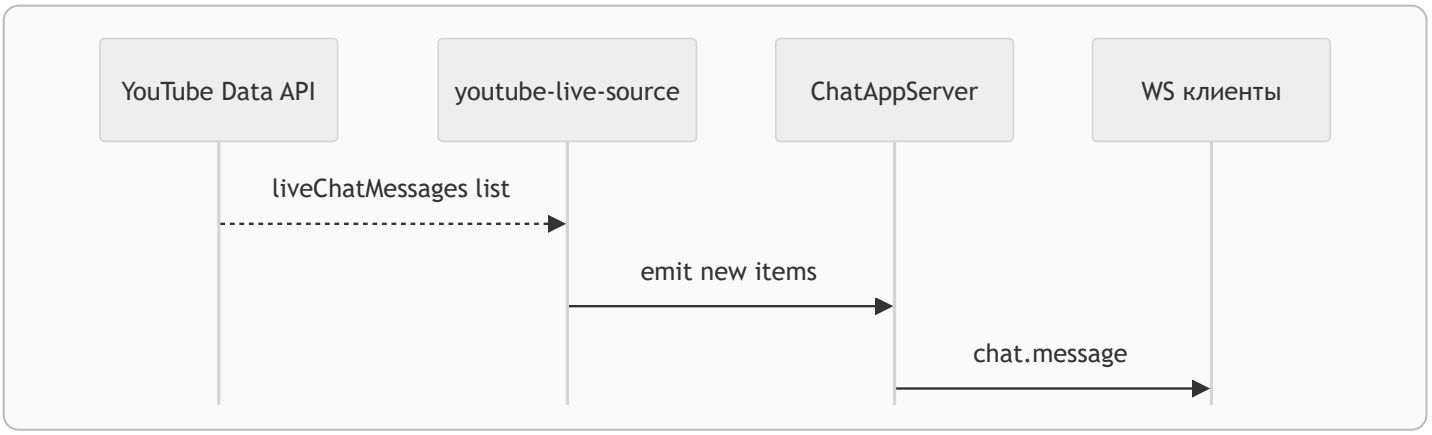
9.5. MemeAlerts — событие WebSocket

Рисунок 9.5 — UML sequence: упрощённый поток

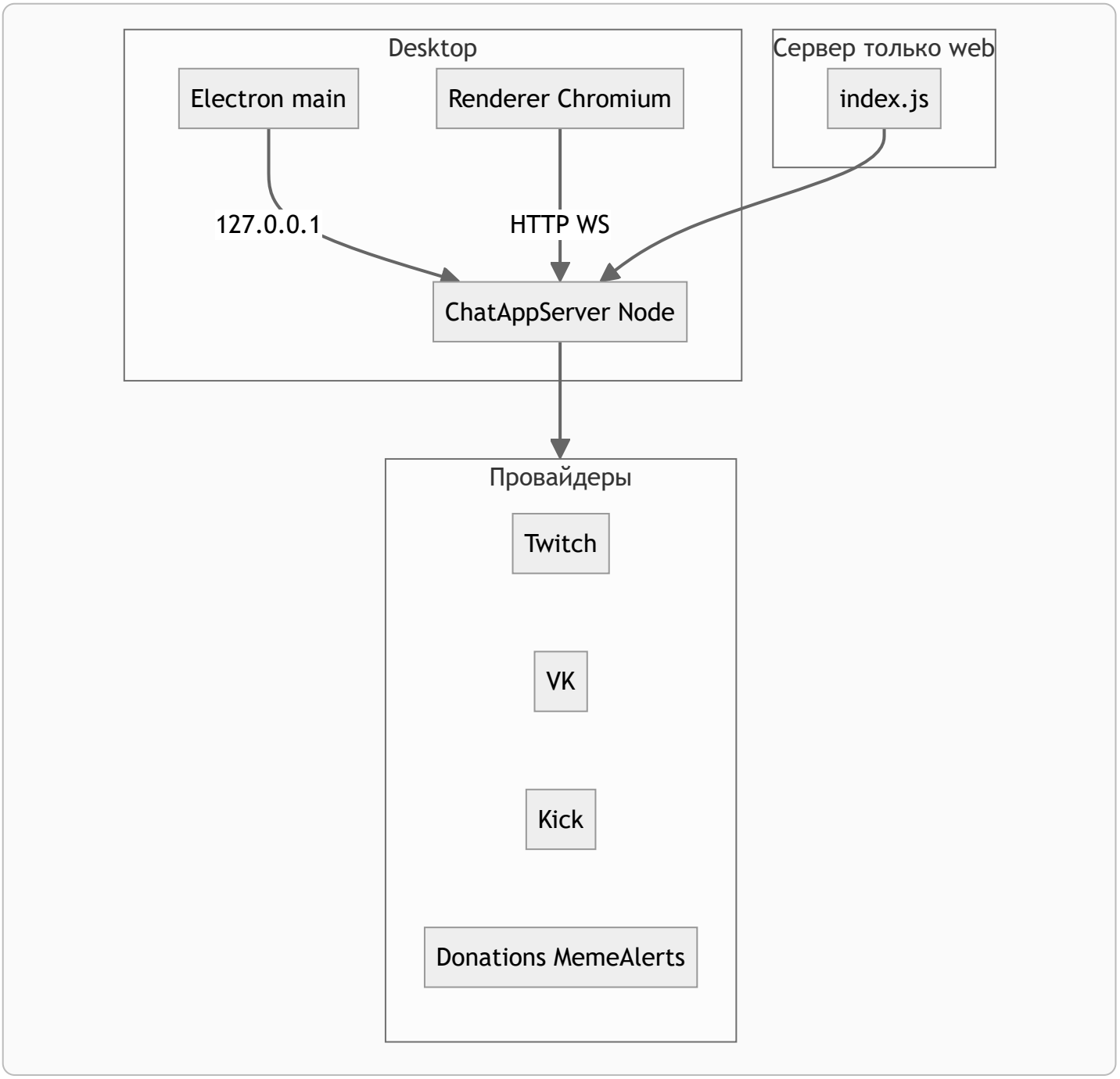


9.6. YouTube Live — сообщение чата (опрос API)

Рисунок 9.6 — UML sequence: упрощённый поток



10. Компоненты и развёртывание



На desktop при занятом порту Electron может подключиться к уже запущенному экземпляру сервера (`startOrReuseDesktopServer`) — два окна, один процесс Node с бэкендом.

11. События по WebSocket (типы `type`)

Таблица 11 — Перечень типов сообщений WebSocket

type	Назначение
`snapshot`	Первичная выдача при подключении: буфер сообщений и актуальное состояние
`state`	Полный или частичный снимок `state`: каналы, транспорты twitch/vk, OAuth, revision UI
`chat.message`	Одно сообщение чата или системное оформленное событие
`chat.moderation`	Удаление, скрытие, таймаут и др. модерация
`chat.annotation`	Доп. метки к сообщению
`chat.clear-local`	Сброс локального отображения
`eventBus.append`	Центр событий: заголовок и деталь
`obs.telemetry`	Снимок с OBS WebSocket
`now-playing`	Трек now playing relay
`debug`	Отладочное состояние при включённой диагностике

12. Группы HTTP API (фрагмент, актуально для 2.7.10)

Таблица 12 — Группы маршрутов `handleHttp` / `handleApiRequest`

Группа	Примеры путей	Роль
Состояние	`GET /api/state`, `GET /api/qr`	Снимок для UI, QR
OAuth / сессии	`/api/auth/twitch/*`, `vk/*`, `kick/*`, `donation-alerts/*`, `youtube/*`, `rutube/*`	Старт, poll, logout, callback, browser login
Донаты MA	`POST /api/meme-alerts/connect|disconnect`	JWT/OBS-ссылка MemeAlerts
Каталоги	`/api/7tv/*`, `/api/twitch/badges|emotes|asset`, `/api/vk/catalog|asset`	Кэш и прокси
Настройки	`POST /api/settings/ui|locale|obs|youtube|kick|rutube|donation-filters|update|…`	Персистенция, в т.ч. `uiLocale`
Обновления	`POST /api/update/check|install|dismiss`; `GET/POST /api/settings/update`	GitHub Latest, Setup, skip-версия
OBS / deck	`/api/obs/*`, `/api/deck/*`	WebSocket OBS; LAN gated токеном
Модерация и шоу	`/api/moderation/*`, poll/prediction/raid/clip/stream	Helix и локальные действия

13. Нефункциональные требования

- **Один экземпляр desktop:** `app.requestSingleInstanceLock` в Electron; при втором запуске — сообщение пользователю.
- **Память и CPU:** ограничение буфера сообщений, метрики в state, опциональное давление по памяти.
- **Безопасность:** contextIsolation в Electron, токены на диске в каталоге данных, HTTPS в продакшене по конфигу хоста; HTTP-сервер по умолчанию привязан к `127.0.0.1`; чувствительные маршруты (debug, экспорт/импорт настроек, OBS, `/api/deck/*`) с недоверенных адресов сети требуют заголовок `X-CoronerChat-Deck-Token`, сверяемый с `CORONERCHAT_DECK_TOKEN` (см. руководство оператора, §5.14, 8.6).
- **Zapret:** только desktop, отдельный процесс/профили под протоколы DPI.

14. Словарь данных (потоки и поля)

- **Сообщение чата** (в `chat.message`): `id`, `source`, `channel`, `author`, `text`, `timestamp`, `raw` (IRC tags / redemption / hype payload), флаги `channelPointsRedemption`, `deleted`, `pinned` и др.
- **state.twitchTransport**: статусы IRC, channelPoints (EventSub / poller / irc-only), hypeTrain, ошибки detail для диагностики.
- **state.vkTransport**: реконнект и сеть VK WS (аналогично по смыслу полям «последний разрыв», ETA и т.д., без EventSub).
- **state.kick**: выбранный канал и поля состояния клиента Kick (OAuth в `state.kickAuth` / ответах `/api/auth/kick/*`).
- **state.youtube**: канал, API key, флаги OAuth для YouTube Live.
- **state.rutube** / auth cookie session: URL эфира, статус чтения/отправки.
- **state.memeAlerts**, **persistedState.memeAlertsToken**: JWT/URL MemeAlerts, `tokenConfigured`, минимальная сумма фильтра.
- **persistedState.uiLocale**, **dismissedUpdateVersions**: язык UI и пропущенные версии апдейта.
- **persistedState.donationAlertsMinAmount** (и связанные поля DA): фильтрация сумм донатов на стороне клиента настроек.
- **Redemption EventSub**: поля `id`, `reward`, `user_input`, плоские `user_id` / `user_login` или вложенный `user` после poller.
- **Hype Train EventSub**: фазы begin / progress / end, в сообщении `raw.hypeTrain` с `level`, `goal`, `total`.
- **DonationAlerts / MemeAlerts**: события оформляются как сообщения с источником `source`, отражающим платформу; детали в `raw` по схеме соответствующего провайдера.

14a. Обновления изделия и локализация UI (сводка)

Канал обновлений (2.7.5). Репозиторий GitHub `dorimeryt-alt/CoronerChat` зашит в приложение. Проверка при запуске: `GET /repos/…/releases/latest` (релиз Latest), с отсевом legacy-тегов линии 2.25–2.35. API: `POST /api/update/check`, `/api/update/install` (скачивание Setup + NSIS), `/api/update/dismiss` (skip-версия в `dismissedUpdateVersions`); «напомнить позже» — только на сессию. Каталог `CoronerChat-data` при обновлении сохраняется.

Локализация (2.7.5). Языки `ru`, `en`, `ro`, `de`, `fi` — словари `src/server/i18n/locales`, персист `uiLocale`, `POST /api/settings/locale`. Клиентский `applyI18nDom` не затирает узлы с вложенными контролами (фикс смены языка в 2.7.5). См. `docs/updates-and-releases.md`, `docs/i18n.md`, `docs/security.html`.

15. Заключение

Документ описывает архитектуру CoronerChat **2.7.10** (IDEF0, DFD, BPMN, UML); п. 1.5 фиксирует реализованный функционал, включая Rutube, UX-пакет, статистику/достижения, i18n и автообновление. Information item: Software Architecture Description (ISO/IEC/IEEE 42010, 15289). Эксплуатация — `docs/coronerchat-operation-print.html`,

`docs/guide.html` (ISO/IEC 26514). При изменении кода синхронно обновляйте маршруты, имена модулей и диаграммы.

Версия изделия: **2.7.10** (`package.json`). Печать: A4. Публикация: GitHub Pages `docs/`.