123CMS

ПРОГРАММА ДЛЯ ЭВМ

Описание технической архитектуры программного обеспечения

Состав компонентов, технологический стек, модель данных, протоколы взаимодействия с устройствами воспроизведения и механизмы защиты информации

Наименование ПО	**Программа для ЭВМ «123CMS»**
Правообладатель	**Сарбашев Никита Борисович**
Версия документа	**1.2**
Год	**2026 г.**

Настоящий документ входит в комплект эксплуатационной документации на программу для ЭВМ «123CMS» и описывает продукт в том виде, в котором он реализован. Исключительное право на программу для ЭВМ «123CMS» принадлежит Сарбашеву Никите Борисовичу.

1. Общая схема

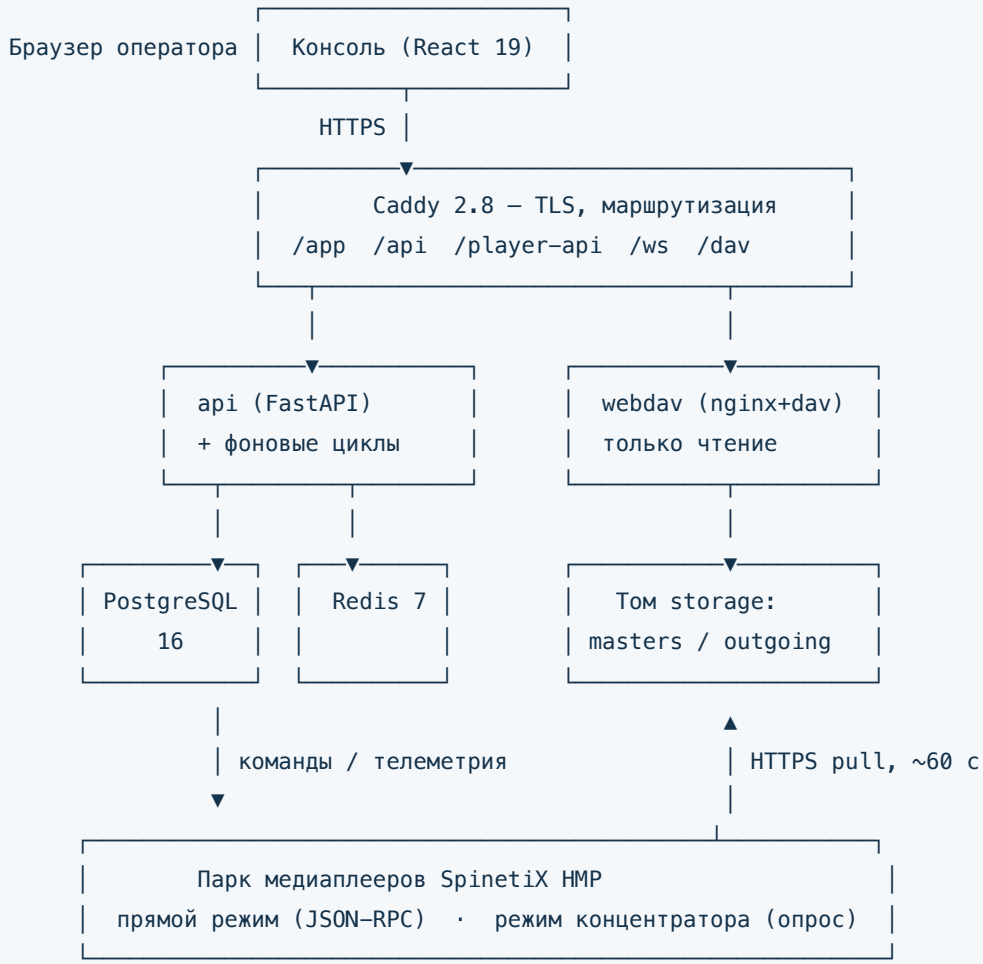
1.1. Принцип построения

Программа для ЭВМ «123CMS» построена как многоарендная серверная система с веб-интерфейсом. Весь программный комплекс поставляется в виде набора контейнеров и разворачивается единым стеком средствами Docker Compose как в облачном варианте, эксплуатируемом правообладателем, так и в варианте установки в контуре заказчика. Состав сервисов, схема их взаимодействия и прикладной код в обоих вариантах идентичны.

Комплекс состоит из следующих логических частей:

- **консоль оператора** — одностраничное веб-приложение на React и TypeScript, поставляемое в виде статических файлов;
- **прикладной сервер** — сервис на Python и FastAPI, обслуживающий программный интерфейс консоли, интерфейс устройств воспроизведения и подсистему аутентификации; является единственным источником прикладной логики;
- **фоновые обработчики** — набор циклов, выполняющих перекодирование медиаматериалов, публикацию подготовленных деревьев файлов, диспетчеризацию команд, сбор телеметрии и плановое обслуживание;
- **база данных PostgreSQL** — основное хранилище состояния;
- **Redis** — очереди, распределенные блокировки, кратковременный кеш и состояние подсистемы безопасности;
- **источник доставки контента** — сервер nginx с модулями WebDAV, отдающий только на чтение подготовленное дерево файлов;
- **обратный прокси-сервер Caddy** — единая точка входа с терминацией TLS.

1.2. Схема взаимодействия компонентов



Ключевое архитектурное решение: **контент никогда не отправляется на устройство принудительно**. Вся доставка построена на модели самостоятельного получения файлов устройством по протоколу HTTPS с фиксированным интервалом. Именно это позволяет разворачивать систему при нахождении устройств за средствами трансляции сетевых адресов, межсетевыми экранами и на мобильных каналах связи без проброса входящих портов и организации туннелей; тот же механизм одинаково обслуживает устройства в локальной сети.

2. Технологический стек

2.1. Серверная часть

Компонент	Технология	Версия
Язык программирования	Python	3.12 (диапазон `>=3.12,<3.13`)
Менеджер пакетов и рабочих пространств	uv	0.11.x
Веб-фреймворк	FastAPI	`>=0.115` (зафиксирована 0.139.0)
Сервер приложений ASGI	Uvicorn (сборка `standard`)	`>=0.30` (зафиксирована 0.50.0)

Компонент	Технология	Версия
Валидация и настройки	Pydantic, pydantic-settings	`>=2.7`, `>=2.4`
Объектно-реляционное отображение	SQLAlchemy 2 (асинхронный режим)	`>=2.0.30` (зафиксирована 2.0.51)
Миграции схемы	Alembic	`>=1.13`
Драйвер PostgreSQL	asyncpg	`>=0.29`
Клиент Redis	redis-py	`>=5.0`
Хеширование паролей	argon2-cffi	`>=23`
Одноразовые пароли (TOTP)	pyotp	`>=2.9`
Формирование QR-кодов	segno	`>=1.6`
Криптографические примитивы	cryptography	`>=43`
HTTP-клиент исходящих вызовов	httpx	`>=0.27`
Безопасный разбор XML	defusedxml	`>=0.7`
Прием многочастных загрузок	python-multipart	`>=0.0.9`
Push-уведомления	firebase-admin	`>=6.5`
Разбор PDF (модуль навигации)	pdfplumber	`>=0.11.10`
Обработка изображений (модуль навигации)	opencv-python-headless	`>=4.10`
Численные вычисления	numpy	`>=1.26`
Перекодирование медиа	FFmpeg	6 и выше

Серверная часть организована как рабочее пространство uv из двух пакетов: сервис прикладного интерфейса и библиотека доступа к данным (модели SQLAlchemy и миграции Alembic) с общим файлом фиксации версий зависимостей. Образ контейнера строится на базе `python:3.12-slim-bookworm` и обслуживает три роли, выбираемые командой запуска: прикладной сервер, одноразовое выполнение миграций и одноразовое первичное наполнение данных.

FFmpeg вызывается как отдельный внешний исполняемый файл (запуск дочернего процесса), а не связывается с приложением в виде библиотеки. Все остальные зависимости распространяются на условиях разрешительных лицензий (Apache-2.0, MIT, BSD, PSF).

2.2. Клиентская часть

Компонент	Технология	Версия
Язык программирования	TypeScript	~5.8
Библиотека представления	React, React DOM	^19.1.0
Сборщик	Vite	^6.3.0
Маршрутизация	react-router-dom	^7.6.0
Управление серверным состоянием	TanStack Query	^5.101.2
Интернационализация	i18next, react-i18next	^24.2.3, ^15.7.4

Компонент	Технология	Версия
Библиотека компонентов и дизайн-токенов	Consta	—
Картографический движок (модуль навигации)	maplibre-gl	^6.0.0
Формирование QR-кодов	qrcode-generator	^2.0.4
Модульное тестирование	Vitest, Testing Library	^3.2.6
Сквозное тестирование	Playwright	^1.61.1
Среда сборки	Node.js	22 (только на этапе сборки)

Консоль представляет собой одностраничное приложение, собираемое в набор статических файлов. В промышленной эксплуатации отдельный серверный компонент на Node.js отсутствует: собранные файлы отдаются напрямую обратным прокси-сервером. Каждый модуль консоли собирается в отдельный загружаемый по требованию фрагмент кода.

2.3. Инфраструктурные компоненты

Компонент	Технология	Роль
База данных	PostgreSQL 16	Основное хранилище состояния
Кеш, очереди, блокировки	Redis 7	Очереди уведомлений, поустройственные блокировки, кеш состояния парка, состояние подсистемы безопасности
Обратный прокси и TLS	Caddy 2.8	Единая точка входа
Источник доставки контента	nginx с модулями `dav` и `dav_ext` на базе Debian bookworm	Отдача подготовленного дерева файлов только на чтение
Перекодирование медиа	FFmpeg 6+	Формирование готовых для устройств версий материалов
Среда контейнеризации	Docker, Docker Compose	Упаковка и оркестрация стека

Штатные образы nginx поставляются без модулей WebDAV, необходимых для работы устройств воспроизведения (устройства выполняют запросы `OPTIONS` и обходят каталоги запросами `PROPFIND` перед получением файлов), поэтому источник доставки контента собирается на базе сборки nginx с расширенным набором модулей.

3. Состав развернутого стека

3.1. Сервисы

Сервис	Образ	Назначение	Опубликованные порты	Зависимости
`postgres`	`postgres:16`	Хранилище состояния	нет	—

Сервис	Образ	Назначение	Опубликованные порты	Зависимости
`redis`	`redis:7`	Очереди, блокировки, кеш, состояние безопасности. Запускается с политикой, исключающей вытеснение данных из памяти	нет	—
`migrate`	`123cms-api:latest`	Одноразовое приведение схемы базы данных к актуальному состоянию	нет	`postgres` (готов)
`api`	`123cms-api:latest`	Прикладной сервер и фоновые обработчики	нет (внутренний порт 8000)	`postgres`, `redis` (готовы), `migrate` (завершен успешно)
`webdav`	`123cms-webdav:bookworm`	Источник доставки контента	нет (внутренний порт 80)	`api`
`db-backup`	`postgres:16`	Ежесуточная зашифрованная резервная копия базы данных	нет	`postgres` (готов)
`caddy`	`caddy:2.8`	Обратный прокси, терминация TLS	**80, 443**	`api`, `webdav`

3.2. Тома данных

Том	Кем монтируется	Содержимое
`pgdata`	`postgres`	Файлы данных PostgreSQL
`redisdata`	`redis`	Персистентность Redis
`storage`	`api` (чтение и запись), `webdav` (только чтение)	Исходные загруженные файлы, перекодированные версии, подготовленные для каждого устройства деревья файлов, служебные файлы аутентификации WebDAV
`caddy-data`, `caddy-config`	`caddy`	Сертификаты и ключи TLS, рабочая конфигурация
`backup-data`	`db-backup`	Зашифрованные резервные копии базы данных

Все состояние комплекса хранится в перечисленных томах; файловые системы самих контейнеров являются одноразовыми.

3.3. Сетевая модель

- Порты в сеть хоста публикует только сервис обратного прокси — 80 и 443.
- База данных и Redis доступны исключительно внутри внутренней сети Docker Compose и не имеют опубликованных портов и выхода на сетевые интерфейсы хоста.
- Источник доставки контента также не публикует портов и доступен только через обратный прокси по пути `/dav`.
- Каждое входящее соединение — от браузеров операторов и от всех устройств воспроизведения, независимо от режима связи, — терминируется на обратном прокси на порту 443.

3.4. Маршрутизация обратного прокси

Путь	Назначение
`/api/*`, `/player-api/*`, `/ws/*`	Прикладной сервер
`/dav/*`	Источник доставки контента. Префикс передается источнику без изменений: он должен совпадать с запрошенным устройством адресом, иначе нарушается обход каталогов
`/app/*`	Статические файлы консоли. Ресурсы с хешем содержимого в имени кешируются как неизменяемые, шрифты — на 30 суток
`/platform`, `/platform/*`	Перенаправление в консоль платформы (маршрут внутри того же приложения)
`/`	Перенаправление на `/app/`

Обратный прокси устанавливает в каждом ответе заголовки `Strict-Transport-Security` и `X-Content-Type-Options: nosniff`, применяет сжатие и задает заголовок сетевого адреса клиента из фактического подключения, исключая его подмену клиентом.

В варианте установки в контуре заказчика имя хоста может быть внутренним; сертификат TLS выпускается встроенным удостоверяющим центром прокси-сервера, что позволяет системе работать полностью в изолированном контуре без обращения к внешним службам проверки. Поддерживается замена на сертификат собственного удостоверяющего центра организации.

4. Прикладной сервер и фоновые обработчики

4.1. Организация процесса

Прикладной сервер представляет собой единое приложение ASGI. Фоновая обработка вынесена не в отдельные контейнеры, а в асинхронные задачи, запускаемые в жизненном цикле приложения и выполняющиеся в том же цикле событий, что и обработка HTTP-запросов:

Цикл	Назначение
Обслуживание	Выборка и отправка команд из очереди, обработка истечения срока команд, разбор зависших отправок, опрос устройств в прямом режиме, пересчет состояния устройств, оценка оповещений, материализация расписания, удаление устаревших метрик
Перекодирование	Конвейер приема, анализа и перекодирования медиаматериалов
Уведомления	Служебный бот мониторинга состояния парка
Обращения	Обработка сообщений встроенного канала обращений в поддержку

Работоспособность каждого цикла отслеживается отдельным механизмом контроля активности и отражается в точке проверки работоспособности. Прекращение работы цикла обслуживания или перекодирования приводит к завершению процесса, после чего контейнер перезапускается средствами Docker; циклы уведомлений и обращений являются необязательными и их отсутствие не влияет на работу комплекса.

4.2. Пространства программного интерфейса

Префикс	Содержание	Аутентификация
`/api/v1/...`	Полный набор операций консоли: устройства, команды, состояние парка, метки, группы, площадки, пользователи, подписка, оповещения, контент, пресеты перекодирования, плейлисты, развертывание, мгновенный показ, показы, перехват, экстренные сценарии, аналитика, программатик-реклама, push-уведомления, правила разделения, настройки арендатора, матрица полномочий, группы видеостен, модуль навигации	Сессионная cookie и токен защиты от подделки межсайтовых запросов
`/api/platform/...`	Уровень оператора платформы: организации, лицензии, группы уведомлений, первичное развертывание	Отдельная сессия администратора платформы
`/player-api/...`	Интерфейс устройств: опрос концентратора, прием подтверждений показа, выгрузка снимков экрана	Токен, выданный конкретному устройству
`/ws/fleet`	Передача изменений состояния парка в консоль в реальном времени	Сессионная cookie
`/healthz`	Проверка работоспособности комплекса	Не требуется

Опубликовано машиночитаемое описание интерфейса в формате OpenAPI, охватывающее 125 адресов и 174 операции.

4.3. Конвейер перекодирования

Конвейер реализован как машина состояний над записями версий материалов, обслуживаемая фоновым циклом:

```
ожидание  →  анализ  →  перекодирование  →  готово
              ↓                ↓
          карантин          ошибка → (повтор) → карантин
```

Ключевые свойства:

- **Атомарный захват задания.** Переход задания в работу выполняется условным обновлением записи, что исключает двойной захват одного задания без применения распределенных блокировок.
- **Восстановление после сбоев.** В начале каждого цикла задания, оставшиеся в промежуточных состояниях дольше установленного времени (например, после аварийного завершения процесса), возвращаются в очередь; задания в состоянии ошибки повторяются до предельного числа попыток, после чего помещаются в карантин, а материал получает состояние «отклонен».
- **Ограничение параллелизма.** Одновременно выполняется не более заданного числа процессов кодирования; каждое кодирование работает в собственной сессии базы данных.
- **Ограничение по времени.** Анализ и кодирование ограничены по времени выполнения; при превышении дочерний процесс принудительно завершается.
- **Проверка параметров до вызова кодировщика.** Параметры пресета, задаваемые арендатором, попадают в командную строку кодировщика, поэтому проверяются по строгому перечню разрешенных значений до вызова; недопустимый пресет помещает версию в карантин, не обращаясь к кодировщику.

- **Отображение хода выполнения.** Процент выполнения публикуется в Redis по принципу наилучшего усилия и не является авторитетным: авторитетным источником состояния остается база данных, а сбой публикации не влияет на ход работы.

5. Модель данных

5.1. Общие свойства схемы

Схема реализована средствами SQLAlchemy 2 в асинхронном режиме и содержит 47 таблиц. Применяются первичные ключи типа UUID, генерируемые средствами PostgreSQL, нативные перечислимые типы и хранение всех отметок времени с часовым поясом в UTC. Изменения схемы поставляются в виде миграций Alembic; на момент составления документа в составе комплекса 52 миграции, применяемые последовательно.

5.2. Механизм многоарендности

Многоарендность реализована на уровне строк: каждая доменная таблица содержит столбец идентификатора арендатора с внешним ключом и каскадным удалением. Схема с отдельными базами данных или отдельными схемами для арендаторов не применяется.

Изоляция обеспечивается тремя независимыми механизмами:

1. **Составные внешние ключи.** Почти каждая доменная таблица несет ограничение уникальности по паре «идентификатор записи, идентификатор арендатора», а дочерние таблицы ссылаются на родительские через составной внешний ключ по паре полей, а не по одному идентификатору. Строка физически не может ссылаться на объект другого арендатора — это запрещено ограничением целостности базы данных. Каждая таблица дополнительно индексирована по идентификатору арендатора.
2. **Репозиторий с ограничением по арендатору.** Весь доступ к данным маршрутизируется через репозиторий, привязанный к арендатору текущей сессии.
3. **Автоматическая проверка исходного кода при сборке.** Статический анализатор проверяет, что запросы к таблицам арендатора выполняются с ограничением по арендатору, и приводит к отказу сборки при нарушении. Исключения — служебные обходы всего массива данных фоновыми циклами — перечислены в явном списке и записывают данные только через ограниченный репозиторий.

Обращения устройств воспроизведения определяют арендатора по собственной записи устройства (по токену или серийному номеру), а не по данным, переданным клиентом.

Намеренно не привязаны к арендатору три категории таблиц: таблица сессий (ограничивается транзитивно через учетную запись пользователя), связующие таблицы «многие ко многим» (обе стороны связи уже принадлежат арендатору) и глобальный карантинный список обращений от еще не закрепленных за арендатором устройств. Таблицы административного уровня платформы находятся над уровнем арендаторов и не имеют такой привязки по построению.

5.3. Вторая ось разграничения — зона

Поверх многоарендности действует вторая, более узкая ось: необязательная привязка записи к площадке для типов «группа», «метка», «материал», «плейлист» и «показ». Реализуется отдельным примесным классом с составным внешним ключом по паре «площадка, арендатор» и обеспечивается собственной автоматической проверкой исходного кода при сборке. Ось арендатора

при этом остается жесткой границей изоляции: зона лишь сужает видимость внутри одного арендатора и никогда не пересекает границы арендаторов.

5.4. Основные группы таблиц

Группа	Таблицы
Арендаторы и доступ	Арендаторы, пользователи, сессии, административные учетные записи платформы, сессии платформы, журнал аудита платформы
Уведомления	Группы уведомлений, расписания групп уведомлений
Обращения в поддержку	Сессии обращений, сообщения обращений
Парк устройств	Площадки, медиаплееры, группы видеостен, карантин необслуживаемых обращений, группы, связь групп и устройств, метки, связь меток и устройств
Коммерческий учет	Подписки, счета
Аудит	Журнал аудита
Взаимодействие с устройствами	Команды, состояние устройства, метрики устройства, события воспроизведения, оповещения, токены push-уведомлений
Контент	Материалы, связь материалов и меток, состав групп замены, правила разделения, пресеты перекодирования, версии материалов, шаблоны проекта устройства
Расписание	Плейлисты, позиции плейлистов, показы, состояние перехвата
Экстренные сценарии	Сценарии, активации сценариев
Программатик-реклама	Рекламные слоты, показы рекламы
Служебные	Состояние системы
Модуль навигации	Объекты, уровни, помещения, точки интереса, узлы и ребра графа проходов, версии схем, табло, привязки киосков

5.5. Значимые ограничения целостности

- **Неизменяемость журнала аудита.** Таблица журнала доступна только на добавление: роль базы данных, от имени которой работает прикладной сервер, не имеет для нее прав изменения и удаления, что дополнительно закреплено триггером.
- **Ровно одна исходная версия на материал.** Среди версий материала ровно одна является исходной (неизменяемая загрузка без привязки к пресету) и не более одной результирующей версии на каждый пресет; обеспечивается частичными и составными уникальными индексами.
- **Машина состояний устройства.** Переход состояния устройства проверяется на уровне базы данных относительно отметки времени подключения: зарегистрированное, но не подключенное устройство не имеет такой отметки, а каждое последующее состояние ее требует. Устройство может занимать лицензионное место только после подключения.
- **Не более одного активного оповещения** на пару «устройство, вид оповещения».

5.6. Хранение секретов в базе данных

Столбцы, содержащие секреты, следуют явному соглашению об именовании, различающему два способа хранения: обратимо зашифрованные значения (пароли управления устройствами, пароли WebDAV, секреты второго фактора аутентификации — шифруются на прикладном уровне

симметричным алгоритмом с ключом из конфигурации) и необратимые хеши, пригодные только для проверки. Каждое поле, содержащее секрет, исключено из отладочного представления объекта.

6. Взаимодействие с устройствами воспроизведения

6.1. Два независимых канала

Для каждого устройства используются два независимых канала: **канал контента**, всегда работающий по модели самостоятельного получения файлов устройством, и **канал команд и телеметрии**, характер которого зависит от того, может ли сервер обратиться к устройству напрямую.

6.2. Канал контента

Независимо от режима команд каждое устройство самостоятельно забирает свои файлы проекта и медиаматериалы по HTTPS с источника доставки контента, используя штатный механизм получения файлов встроенного программного обеспечения устройства, с интервалом около 60 секунд. Принудительная отправка файлов не производится никогда.

Разграничение доступа к контенту. При подключении устройства формируется случайный токен длиной 256 бит, из которого строится однострочный файл аутентификации базовой схемы, размещаемый в служебном дереве тома хранилища. Сервер nginx считывает файл, соответствующий конкретному устройству и арендатору, поэтому каталог одного устройства недоступен по учетным данным другого устройства и тем более другого арендатора. Открытое значение токена возвращается вызывающему ровно один раз — для внесения в настройки устройства; в базе данных оно хранится в зашифрованном виде, в файловой системе — только в виде необратимого хеша.

Публикация. Фоновый обработчик материализует текущее разрешенное расписание и назначения контента в каталог соответствующего устройства. Реализованы следующие свойства:

1. Формируется итоговый набор готовых версий, которые должны быть у устройства, и список расписания, по которому устройство их обнаруживает.
2. Файлы попадают в каталог устройства жесткими ссылками, а не копированием, — без дополнительного расхода дискового пространства; каждая запись атомарна, поэтому источник доставки никогда не отдает частично записанный файл.
3. Если требуемый набор файлов и список расписания идентичны предыдущей публикации, работа полностью пропускается и устройство не уведомляется — повторная публикация без изменений никогда не заставляет устройство перемонтировать контент посреди воспроизведения.
4. Этап удаления более не нужных файлов защищен предохранителем: если требуемый набор файлов не может быть полностью определен на диске, удаление откладывается и записывается в журнал, а список расписания все равно обновляется. Ошибка выше по цепочке не может оставить работающий экран без контента.
5. После фактической публикации устройству направляется побуждение выполнить получение файлов немедленно, не дожидаясь очередного интервала опроса.

6.3. Прямой режим

Устройство, доступное серверу по сети, управляется непосредственно:

Операция	Механизм
Команды	Вызовы JSON-RPC на устройстве

Операция	Механизм
Хранилище «ключ — значение»	Запись значения по соответствующему адресу интерфейса устройства
Телеметрия состояния	Чтение отчета устройства о собственном состоянии в формате XML
Оперативный снимок экрана	Получение текущего изображения с базовой аутентификацией

Клиент работает исключительно по HTTPS. При первом подключении самоподписанный сертификат устройства закрепляется по принципу доверия при первом контакте, отпечаток отображается оператору для сверки, а повторное закрепление выполняется явным действием.

Реализован полный перечень методов, предоставляемых устройством: перезапуск, выключение, выборочный сброс, чтение сведений об устройстве, чтение и запись конфигурации, разовое действие получения контента по требованию, запуск и проверка состояния обновления встроенного программного обеспечения, установка пароля управления, операции с хранилищем «ключ — значение», управление беспроводным соединением. Отдельных методов управления воспроизведением и получения снимка экрана устройство не предоставляет — соответствующие команды консоли строятся поверх перечисленных механизмов.

Ограничения обращений. Реализованы: распределенная блокировка на каждое устройство, сериализующая управляющие вызовы между экземплярами прикладного сервера; минимальный интервал обращений на пару «устройство, операция» (60 секунд для планового опроса, 10 секунд для снимка экрана по требованию, отдельный интервал для побуждения к получению файлов); общий для процесса семафор, ограничивающий число одновременных соединений с устройствами; ограничение времени каждого вызова и ограниченное число повторов с нарастающими задержками.

6.4. Режим концентратора

Устройство, недоступное серверу напрямую, периодически само обращается к серверу. Каждое обращение представляет собой уведомление одного из типов (готовность, сведения о состоянии, факт перезапуска, состояние получения файлов), а ответ сервера может нести не более одной команды. Устройство выполняет команду и сообщает результат при следующем обращении.

```
Устройство  ──обращение (готовность)──►  Сервер
Устройство  ◄──одна команда из очереди──  Сервер
            (устройство выполняет команду)
Устройство  ──обращение с результатом──►  Сервер
Устройство  ◄──следующая команда, если есть──  Сервер
```

Реализованные особенности:

- **Аутентификация по токену устройства.** Токен передается в заголовке авторизации (с резервным вариантом передачи параметром адреса) и проверяется сравнением, устойчивым ко временным атакам, с хранимым необратимым хешем. Арендатор определяется исключительно по записи устройства, соответствующей токену, и никогда — по содержимому запроса. Токен формируется один раз при регистрации устройства и не подлежит смене.
- **Подключение при первом обращении.** Первое аутентифицированное обращение выполняет процедуру подключения устройства.
- **Идемпотентный прием результатов.** Повторно переданный результат команды не создает дублирующих записей.
- **Пустой ответ при отсутствии команд.** При отсутствии команд возвращается действительно пустое тело ответа: передача любого тела без корректного метода приводит к неверному поведению

встроенного программного обеспечения устройства.

- **Приоритет экстренных команд** при выборе команды для вложения в ответ.

6.5. Очередь команд

Каждая команда записывается строкой в очередь на базе базы данных до отправки. Запись несет тип, параметры, состояние, ключ идемпотентности, счетчик попыток и их предельное число, срок действия, результат и инициатора. Каждый переход состояния фиксируется в неизменяемом журнале аудита.

Состояния: «в очереди», «отправлена», «выполнена», «ошибка», «истекла», «заменена». Срок жизни по умолчанию 24 часа и 10 минут для экстренных команд. Предусмотрены три попытки доставки с задержками 30 секунд, 2 минуты и 10 минут.

6.6. Подтверждение показа

Устройства самостоятельно передают на прикладной сервер события фактического воспроизведения с авторизацией по выданному конкретному устройству токену. Записи снабжены ключом устранения дубликатов, уникальным в пределах устройства. Эти события служат авторитетным источником для статистики воспроизведения и учитываются логикой определения состояния устройства: недавнее событие воспроизведения удерживает устройство в состоянии «предупреждение» вместо перехода в состояние «не в сети».

7. Механизмы защиты информации

7.1. Аутентификация

Механизм	Реализация
Хеширование паролей	argon2id, единый экземпляр хешера на процесс, контроль минимальной длины, поддержка автоматического перехеширования при изменении параметров
Второй фактор	Одноразовые пароли по алгоритму TOTP (RFC 6238), шаг 30 секунд, допуск расхождения часов в один шаг, блокировка повторного использования уже предъявленного шага времени
Подключение второго фактора	QR-код и текстовый ключ для ручного ввода при активации учетной записи по приглашению
Сессии	Серверная запись сессии, cookie сессии с признаками защиты от чтения сценариями, передачи только по защищенному соединению и ограничения межсайтовой передачи
Защита от подделки межсайтовых запросов	Отдельная cookie с токеном, передаваемым также в заголовке запроса (схема двойной отправки); проверяется для каждого изменяющего данные запроса
Срок действия сессии	Таймаут простоя и абсолютный срок жизни; сессия истекает даже при непрерывной активности
Блокировка учетной записи	Нарастающая задержка при повторных неуспешных попытках входа
Ограничение частоты запросов	Счетчик с фиксированным окном в Redis по паре «область, идентификатор»
Защита от перебора по множеству учетных записей	Отдельное ограничение числа неуспешных входов с одного сетевого адреса

Механизм	Реализация
Поведение при недоступности Redis	Отказ закрыто: операции аутентификации отклоняются, а не пропускаются без проверки

Долгоживущие ключи программного интерфейса, заменяющие полноценный вход пользователя, в системе отсутствуют.

Состояние подсистемы безопасности хранится в Redis, запускаемом с политикой, исключающей вытеснение данных из памяти при ее нехватке.

7.2. Авторизация

Каждый аутентифицированный запрос проверяется дважды: по роли вызывающего и, для ресурсов, привязанных к площадке, — по его области ответственности. Обе проверки выполняются централизованно, на стороне сервера, для каждого запроса, включая прямые обращения к программному интерфейсу. Маршрут, требующий полномочий, объявляет требуемое право явно, и проверка выполняется до какой-либо прикладной логики. Клиентской логики разграничения доступа, которой можно было бы доверять, не существует.

Отказ в доступе фиксируется в журнале аудита с указанием причины (недостаточные полномочия роли либо выход за пределы области ответственности).

7.3. Изоляция данных

Изоляция арендаторов реализована как структурное свойство схемы базы данных (составные внешние ключи), а не как соглашение, проверяемое при разработке. Дополнительно применяются ограничение доступа на уровне репозитория и автоматическая проверка исходного кода при сборке. Запрос, затрагивающий площадку или арендатора вне области ответственности вызывающего, отклоняется, а не отфильтровывается после выполнения.

7.4. Защита данных при хранении

Секреты, которые должны оставаться восстановимыми (пароли управления устройствами, пароли WebDAV, секреты второго фактора), хранятся в базе данных зашифрованными симметричным алгоритмом с ключом, считываемым из конфигурации окружения. Резервные копии базы данных шифруются алгоритмом AES-256 с отдельной парольной фразой; сценарии резервного копирования отказываются создавать незашифрованную копию, если парольная фраза не задана.

Разделены две роли базы данных: роль, от имени которой выполняются миграции и которая обладает правами изменения структуры, и роль времени выполнения с ограниченным набором прав, не включающим изменение и удаление записей журнала аудита.

7.5. Защита данных при передаче

Весь трафик — от браузеров операторов и от каждого устройства, независимо от режима связи, — терминируется по TLS на обратном прокси, единственном компоненте с опубликованными портами. Исходящие обращения к устройствам в прямом режиме выполняются только по HTTPS с закреплением сертификата устройства по принципу доверия при первом контакте и явным подтверждением повторного закрепления.

7.6. Аудит

Все действия, изменяющие состояние системы, фиксируются в неизменяемом журнале с указанием инициатора (пользователь, система или устройство), действия, объекта, категории и структурированных подробностей. Неизменяемость обеспечивается ограничением прав роли базы данных и триггером, то есть на уровне хранилища, а не только логикой приложения.

8. Резервное копирование, наблюдаемость и развертывание

8.1. Резервное копирование

Выделенный сервис создает резервную копию базы данных ежесуточно в заданный час средствами штатной утилиты выгрузки в сжатом формате с последующим шифрованием алгоритмом AES-256. Копии старше заданного числа суток удаляются автоматически. Дополнительно поставляется сценарий резервного копирования медиатеки с идентичной политикой шифрования и хранения и сценарий учебного восстановления, восстанавливающий последнюю копию в одноразовую базу данных, проверяющий наличие записей в ключевых таблицах и удаляющий ее; учебное восстановление никогда не затрагивает рабочую базу данных.

Дерево подготовленных для устройств файлов намеренно не входит в резервную копию: оно восстанавливается заново публикацией из исходных файлов после восстановления базы данных.

8.2. Наблюдаемость

- **Точка проверки работоспособности комплекса** возвращает состояние базы данных, Redis и каждого фонового цикла; общее состояние считается штатным только при штатном состоянии всех компонентов.
- **Проверки работоспособности на уровне контейнеров** реализованы для базы данных, Redis, прикладного сервера и источника доставки контента, а порядок запуска сервисов выстроен по зависимостям.
- **Постоянное соединение состояния парка** передает изменения состояния устройств в консоль в реальном времени.
- **Внешняя доставка оповещений** реализована веб-перехватчиком и служебным ботом мониторинга; оба механизма полностью неактивны, пока не настроены, и их сбои не влияют на работу основной логики.

8.3. Развертывание и обновление

Схема базы данных приводится к актуальному состоянию выделенной одноразовой задачей, объявленной зависимостью прикладного сервера, — приложение не применяет миграции самостоятельно при старте и предполагает, что схема уже актуальна. При обновлении миграции выполняются отдельным явным шагом до поднятия остальной части стека.

Миграции применяются только в направлении повышения версии; обратное выполнение миграций не поддерживается, и возврат к предыдущему выпуску выполняется восстановлением из резервной копии, снятой непосредственно перед обновлением.

Консоль поставляется в виде статических файлов, обновляемых заменой содержимого соответствующего каталога, а не пересборкой образа прикладного сервера. Кеш служебного сценария прогрессивного веб-приложения привязан к идентификатору сборки, поэтому каждое

развертывание полностью вытесняет предыдущий кеш оболочки приложения без ручного управления версиями.