Описание технической архитектуры программного обеспечения

Наименование ПО: Программный комплекс распознавания и оценки физической активности «AI Цифровая физкультура».

Правообладатель: ООО «ЭЙАЙ СОЛЮШНС», Сахалинская область, М.О. Южно-Курильский, Тер. ТОР Курилы; ОГРН 1256500006388 от 26.12.2025; ИНН/КПП 6500027812/650001001.

Документ описывает совокупность важнейших решений об организации программной системы: выбор структурных элементов и их интерфейсов, соединение элементов структуры, архитектурный стиль.

---

1. Архитектурный стиль

ПО построено как **сервис-ориентированная контейнеризованная система** (микросервисная архитектура) со следующими определяющими принципами:

1. **Разделение ответственности «тонкий клиент — тяжёлый инференс».** Клиент (браузер или киоск-терминал) отвечает за захват видео, интерфейс, бизнес-логику подсчёта повторений и отчётность. Ресурсоёмкий инференс нейросетей вынесен в отдельный GPU-сервис.
2. **Событийно-потоковое взаимодействие.** Кадры видео и результаты распознавания передаются по постоянному соединению WebSocket, а не запрос-ответ HTTP, что обеспечивает работу в реальном времени (обработка порядка 30 кадров в секунду).
3. **Изоляция и независимое развёртывание.** Каждый структурный элемент — отдельный Docker-контейнер с собственным жизненным циклом; связность — только через объявленные сетевые интерфейсы.
4. **Единая точка входа (Reverse Proxy).** Внешний доступ ко всем элементам системы проходит через обратный прокси-сервер, маршрутизирующий запросы по префиксу пути.
5. **Отказоустойчивость и деградация.** Отсутствие GPU переводит инференс на CPU; отсутствие специализированной модели мяча переключает систему на резервный детектор; недоступность языковой модели отключает только текстовые рекомендации, не нарушая основную функциональность.
6. **Локальность обработки данных.** Инференс выполняется на площадке заказчика, видеопоток не покидает контур установки.

---

2. Структурные элементы системы

Элемент (сервис)	Роль	Технологии	Внутренний порт
`nginx`	Обратный прокси, единая точка входа, TLS-терминация	nginx	80 / 443

Элемент (сервис)	Роль	Технологии	Внутренний порт
`gesture_web_demo`	Клиентское веб-приложение упражнений (SPA), статика	JavaScript, HTML/CSS, nginx	80 (внешн. 4575)
`rtmpose_server`	**Ядро ИИ**: инференс нейросетей позы и мяча, мультиобъектный трекинг	Python, FastAPI, ONNX Runtime, CUDA	10290
`gesture_ws_server`	Сервер прикладной логики: приём кадров, агрегация событий, выгрузка записей, интеграция с языковой моделью	Python, WebSocket	10282
`admin_api`	Административный API: объекты, пользователи, отчёты, авторизация	Python, FastAPI, JWT	8000
`postgres`	Хранилище данных (объекты, занятия, история результатов)	PostgreSQL 16	5432
`go2rtc`	Медиашлюз RTSP → WebRTC для IP-камер	go2rtc	1984 / 8554 / 8555
`camera_configurator`	Мастер настройки камер с ИИ-ассистентом	React	80 (внешн. 2105)
`frog_pond`, `games`, `arcade`, `runner`, `pond3d`, `godot_frog`	Детские игровые модули с управлением движениями тела	React, Pixi.js, Three.js, Godot	80 (внешн. 4577–4582)
`ptz_tracker`	Автосопровождение участника поворотной PTZ-камерой (опциональный профиль)	Python	8090

Компоненты ИИ, исполняемые внутри `rtmpose_server`: **YOLOX-nano** (детекция людей и мяча), **RTMPose** (17 ключевых точек скелета), **TrackNetV3 + InpaintNet** (детекция и восстановление траектории мяча), **OC-SORT** (фильтр Калмана + метрика OKS — сопоставление скелетов между кадрами). Клиентский инференс — **MediaPipe PoseLandmarker** (33 точки) в игровых модулях.

---

3. Интерфейсы структурных элементов

3.1. Внешние интерфейсы (граница системы)

Интерфейс	Протокол	Назначение
Веб-интерфейс пользователя	HTTPS	Работа занимающегося и преподавателя
Захват видео с локальной камеры	WebRTC / `getUserMedia`	Источник видеопотока в браузере
Подключение IP-камер	RTSP → WebRTC (через `go2rtc`)	Источник видеопотока со стационарных камер
Административный API	HTTPS/REST + JWT	Управление объектами и отчётами

3.2. Внутренние интерфейсы (между сервисами)

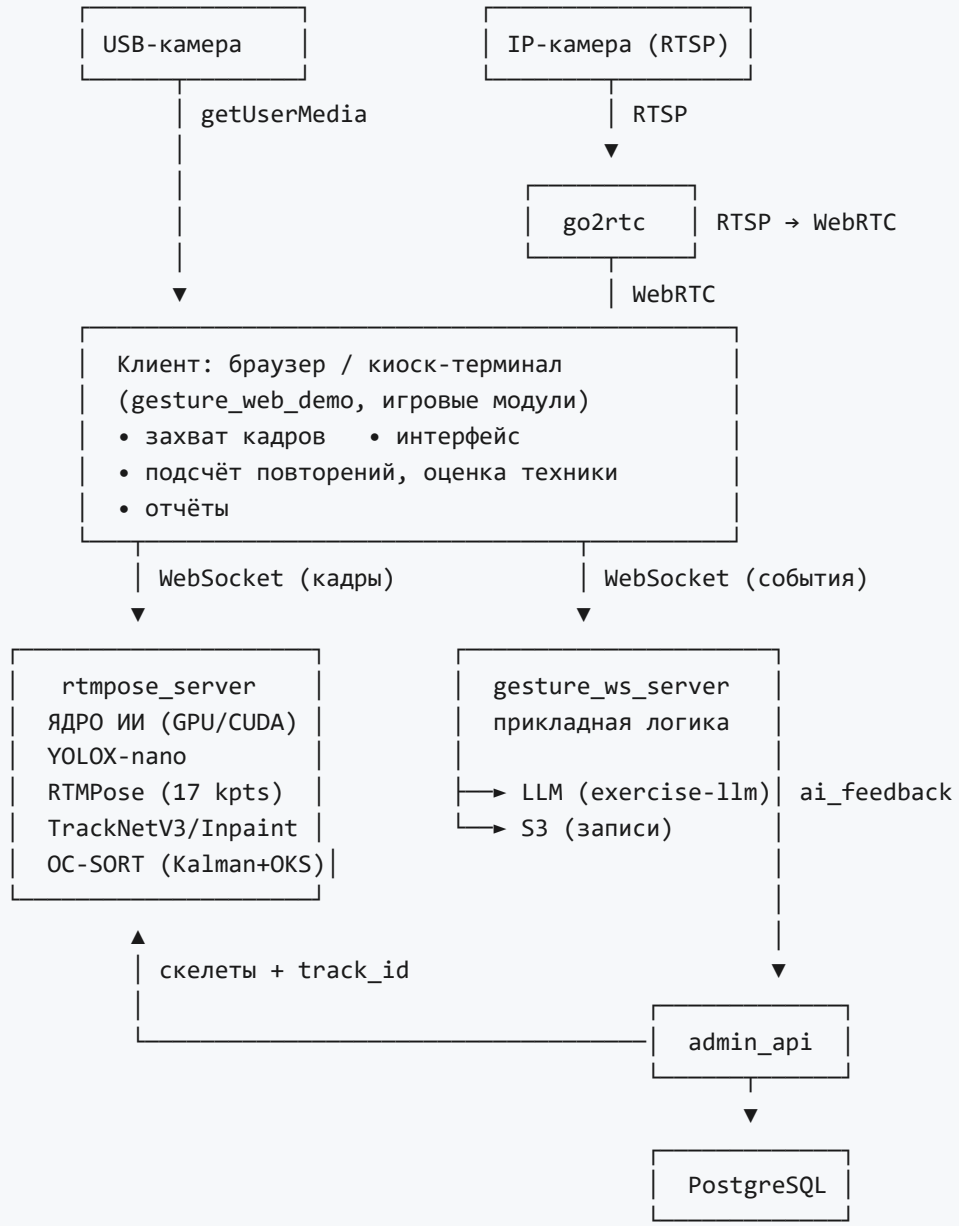
От → К	Протокол	Содержание обмена
Клиент → `rtmpose_server`	**WebSocket** `/ws`	Кадры (JPEG/бинарно) → массив персон: `track_id`, 17 точек `(x, y, score)`
Клиент → `gesture_ws_server`	**WebSocket** `/ws` (до 8 МБ на сообщение)	Кадры и события занятия, запись видео
`gesture_ws_server` → LLM	HTTP, OpenAI-совместимый REST (`/v1`)	Запрос текстовой рекомендации по технике → `ai_feedback`
`gesture_ws_server` → S3	HTTPS (S3 API)	Выгрузка видеозаписей занятий
`admin_api` → `postgres`	TCP, протокол PostgreSQL	Персистентность данных
`nginx` → все сервисы	HTTP / WS (проксирование)	Маршрутизация по префиксу пути
Клиент → `go2rtc` → IP-камера	WebRTC ↔ RTSP	Доставка видеопотока в браузер

Сервисные (диагностические) интерфейсы ИИ-ядра: `GET /health` — состояние и устройство инференса (`cuda` / `cpu`); `GET /debug/detect` — результаты детекции и оценки достоверности поз; `GET /exercise/log` — журнал распознанных упражнений.

---

4. Соединение элементов структуры

4.1. Схема потоков данных



Внешний доступ ко всем сервисам — через обратный прокси nginx (80/443, TLS).

4.2. Сетевая организация

- Все сервисы объединены во внутреннюю сеть Docker-проекта (`default`) и недоступны извне напрямую.
- Наружу опубликован только `nginx` (порты 80/443), выполняющий маршрутизацию.
- Сервис `go2rtc` работает в режиме `network_mode: host` — это необходимо для доступа к IP-камере, размещённой в отдельном физическом сетевом сегменте.
- Дополнительная сеть `ais5_net` используется для межстекового взаимодействия (передача событий во внешнюю аналитическую платформу); подключён к ней только `gesture_ws_server` .

4.3. Управление состоянием и данными

- **Постоянное хранение:** PostgreSQL (том `pgdata`), файлы загрузок (том `admin_uploads`).
- **Веса нейросетевых моделей** (ONNX) включены в состав образа `rtmpose_server` (каталог `models_cache`); доступ в сеть Интернет в процессе работы не требуется.
- **Конфигурация** задаётся переменными окружения (`.env`), в том числе устройство инференса `RTMPOSE_DEVICE`, режим точности `RTMPOSE_MODE`, пороги детекции `RTMPOSE_DET_SCORE` и достоверности позы `RTMPOSE_MIN_POSE_SCORE`.
- **Отказоустойчивость:** все сервисы запускаются с политикой `restart: unless-stopped`; для `postgres` определена проверка готовности (healthcheck), от которой зависит запуск `admin_api`.

---

5. Уровни (слои) архитектуры

Слой	Содержание
Слой представления	Веб-клиент упражнений, игровые модули, административная панель, мастер настройки камер
Слой прикладной логики	Конечные автоматы фаз упражнений, счётчики повторений, оценка техники, проверка нормативов ГТО, ведение счёта матча
Слой искусственного интеллекта	Инференс нейросетей (детекция, оценка позы, детекция мяча), мультиобъектный трекинг, генерация текстовых рекомендаций языковой моделью
Слой доступа к данным	`admin_api`, объектная модель, миграции
Слой хранения	PostgreSQL, тома Docker, объектное хранилище S3 (записи занятий)
Слой инфраструктуры	Docker Compose, обратный прокси nginx, медиашлюз go2rtc, среда исполнения ONNX Runtime + CUDA

---

6. Ключевые архитектурные решения и их обоснование

1. **Вынесение инференса в отдельный GPU-сервис.** Обеспечивает работу в реальном времени (~30 мс на кадр на NVIDIA RTX 3050 против ~400 мс на CPU) и позволяет обслуживать несколько клиентских терминалов одним сервером.
2. **Связка bottom-up детекции и top-down оценки позы (YOLOX → RTMPose).** Обеспечивает одновременное распознавание нескольких участников, что недостижимо для односкелетных детекторов.
3. **Трекинг на основе фильтра Калмана с метрикой OKS.** Сохраняет индивидуальный счёт за участником при кратковременной потере детекции и при пересечении траекторий.
4. **Специализированная модель TrackNetV3 для мяча.** Кадровые детекторы не находят быстролетящий смазанный мяч; модель анализирует последовательность из 8 кадров.
5. **Подсчёт повторений на стороне клиента.** Снижает нагрузку на сервер и сохраняет отзывчивость интерфейса при кратковременных сетевых задержках.
6. **Контейнеризация и единый `docker compose`.** Обеспечивает воспроизводимость развёртывания и совместимость с российскими ОС (Astra Linux, РЕД ОС).

7. Требования, обеспечиваемые архитектурой

- **Производительность:** обработка видеопотока в реальном времени, до 3 участников одновременно.
- **Масштабируемость:** горизонтальное добавление терминалов к одному ИИ-серверу; вертикальное — заменой GPU.
- **Безопасность:** внешний доступ только через обратный прокси с TLS; авторизация административного API по JWT; сервисы БД и ИИ-ядра не публикуются наружу.
- **Импортонезависимость:** развёртывание на Linux (в т.ч. отечественные ОС), локальный инференс, отсутствие обязательных обращений к иностранным облачным сервисам, отсутствие проприетарных иностранных SDK.

Руководитель ООО «ЭЙАЙ СОЛЮШНС»

‹должность› ____ / ‹Фамилия И.О.› /

«_**»** _ ***20*** г. М.П.