# Описание технической архитектуры программного обеспечения

**1. Назначение и контекст**

Система обеспечивает сквозной цифровой процесс планирования, выполнения и подтверждения замеров интенсивности движения на ПУИДД (пунктах учёта), включая документооборот (планы-графики, акты, отчёты, фотоотчёты), хранение первичных данных и предоставление отчётности.

Основные пользователи: Администратор, Менеджер групп/подгрупп, Учётчик, Представители Заказчика.

**2. Технологический стек**

**Backend**

* Язык/фреймворк: PHP 8.3, Laravel 11
* Аутентификация SPA: Laravel Sanctum
* RBAC: spatie/laravel-permission
* Аудит действий: spatie/laravel-activitylog
* Очереди: Laravel Queue (Redis)
* Хранилище файлов: Laravel Filesystem (локально/S3-совместимое)
* Отчётность: генерация PDF (jsPDF на фронте, опционально dompdf на бэке), Excel (maatwebsite/excel — опционально)

**Frontend**

* Vue 3, Vite, Pinia, Vue Router
* Bootstrap 5 + Bootstrap Icons
* Экспорт PDF: jsPDF + jspdf-autotable

**Инфраструктура**

* Контейнеризация: Docker / Laravel Sail
* БД: Postgres 17 (или MariaDB)
* Кэш/очереди: Redis 7
* Веб-сервер (prod): Nginx + PHP-FPM

**3. Роли и права (RBAC)**

* **Администратор**: управление пользователями, ролями, справочниками, правами.
* **Менеджер группы/подгруппы**: назначение исполнителей, контроль выполнения, подписание планов.
* **Учётчик**: загрузка замеров и фото «до/после», формирование форм учёта.
* **Заказчик**: просмотр, согласование документов, выгрузка отчётов.

Права реализованы через spatie/permission и проверяются как на API (middleware), так и на фронтенде (UI-гварды).

**4. Высокоуровневая архитектура**

REST API (Laravel)  
Sanctum/RBAC

Frontend (SPA)  
Vue3/Vite/Pinia/Bootstrap

Postgres  
Core domain entities

File Storage  
(photos, docs, exports)

Async jobs

Queue Workers (Redis)

-генерация PDF/Excel

-ресайз фото/превью

-импорт/экспорт, нотификации

**5. Модули Backend (Laravel)**

1. **Аутентификация и сессии**
   * Sanctum для SPA-аутентификации (cookie-based), CSRF-защита.
   * Ограничение по CORS и доменам.
2. **Пользователи, роли, разрешения**
   * CRUD пользователей, привязка к ролям/правам.
   * Логи действий (активация/блокировка, выдача прав).
3. **Справочники**
   * Регионы, районы, дороги/участки, ПУИДД (координаты, протяжённость, частота мониторинга).
   * Менеджеры групп/подгрупп, назначение районов.
4. **Заявки и календарные планы**
   * CRUD заявок (периоды, ответственные, набор ПУИДД).
   * Создание календарного плана-графика.
   * Подписание периодов (подписи «месяц/год/заявка»).
5. **Планы точек (Point Plans)**
   * Слоты «дата/время» для конкретного ПУИДД.
   * Статусы: запланировано → выполнено → согласовано.
   * Сводка по прогрессу (planned/completed/overdue/remaining).
6. **Замеры и статистика**
   * Приём часовых рядов по категориям ТС.
   * Автозаполнение по данным предыдущего года (бизнес-правила).
   * Валидация и аудит изменений.
7. **Медиафайлы (фото ДО/ПОСЛЕ, документы)**
   * Хранение файлов (оригинал + превью).
   * Привязка к плану, типизация (before/after/doc), безопасные URL.
8. **Отчётность и документы**
   * План-график (PDF/Excel), КС-2, фотоотчёты, карты ПУИДД, отчёты интенсивности.
   * Шаблонизация, автоподстановка ФИО/должностей.
9. **Уведомления и согласование**
   * Email/внутренние уведомления (очереди).
   * Статусы согласования заявок/планов.
10. **Журналирование**

* activitylog: кто/что/когда изменил; привязка к сущностям.

**6. Модули Frontend (Vue SPA)**

* **AppShell**: каркас приложения, авторизация, глобальные перехватчики ошибок.
* **Pinia-stores**: Auth, справочники, планы, заявки, отчётность.
* **Маршрутизация** (Vue Router): экраны профиля, пользователи, заявки, планы, ПУИДД, назначения, отчёты, фотоотчёты, документы.
* **Компоненты**: таблицы с пагинацией/фильтрами, модальные формы, drag-and-drop (назначение менеджеров), экспорты PDF/Excel.
* **UI**: Bootstrap 5, Bootstrap Icons.
* **Экспорт PDF**: jsPDF + autotable (план-графики, формы учёта).

**7. API-контракт**

* **Стиль**: REST/JSON, версии через префикс /api.
* **Пагинация**: page, per\_page; meta {current\_page, last\_page, total}.
* **Фильтры**: request\_id, district\_id, subgroup\_manager\_id, month, диапазоны дат.
* **Ответы об ошибках**: HTTP-коды + {message, errors?}.
* **Авторизация**: Sanctum cookie + CSRF токен; все state-изменяющие операции за защищённым middleware.

**8. Данные (основные сущности)**

* users, roles, permissions, model\_has\_roles, role\_has\_permissions
* regions, districts (ссылки на ответственных)
* roads/segments (опционально), points (ПУИДД: координаты, частота, протяжённость, номер записи)
* requests (заявки), schedules (планы-графики),
* point\_plans (слоты «день/время/ПУИДД/заявка/м-джер»), signatures (подписанные {request\_id, year, month})
* measurements (часовые ряды по category\_key), files (before/after/doc, url, thumb\_url)
* activities (журнал аудита)

Индексы по FK, датам и частым фильтрам (request\_id, district\_id, month, subgroup\_manager\_id).

**9. Очереди и фоновые задачи**

* Генерация превью изображений, видео-кадров (если применимо).
* Формирование отчётов (PDF/Excel) по запросу.
* Импорт/экспорт больших наборов (chunked).
* Отправка уведомлений.
* Планировщик (Scheduler) для периодических задач: очистка временных файлов, ротация, пересборка кэшей.

Брокер: Redis. Воркер запускается командой:

./vendor/bin/sail artisan queue:work --timeout=300

**10. Хранение файлов**

* Файлы загружаются через API, сохраняются в storage/app/public (dev) или S3-совместимое хранилище (prod).
* Доступ — по защищённым URL (подписанные/временные), директории на основе сущности и даты.
* Превью (thumbnails) для изображений, ограничения на размер/формат.

**11. Безопасность**

* HTTPS везде (валидный TLS-сертификат; HSTS).
* Sanctum (cookie, SameSite=Lax/Strict), CSRF, CORS-белый список.
* RBAC на бэке + условный рендер на фронте.
* Лимитирование запросов (rate limit) на авторизацию/загрузки.
* Очистка EXIF/метаданных при загрузке (опционально).
* Регулярные обновления зависимостей (Dependabot/ renovate — опционально).

**12. Окружения и развёртывание**

**Dev (локально)**

* Запуск одной командой: ./vendor/bin/sail up
* Очереди: ./vendor/bin/sail artisan queue:work --timeout=300
* Frontend dev: npm run dev -- --host 127.0.0.1

**Prod**

* Контейнеры: Nginx, PHP-FPM (app), MySQL, Redis, Queue worker, Scheduler.
* Конфиги через .env (секреты из CI/CD/хранилища секретов).
* Миграции: php artisan migrate --force в пайплайне.
* Логи в файловую систему + системный сборщик (опц.).

**13. CI/CD (GitLab)**

* Стейджи: install → build → test → package → deploy.
* Сборка фронта (Vite) → публикация dist/ как артефакт → копирование в веб-образ.
* Backend: composer install --no-dev --optimize-autoloader, кэширование конфигов/роутов/видов.
* Публикация Docker-образов в GitLab Registry.
* Деплой на staging/production с миграциями и прогревом кэшей.

**14. Производительность и масштабирование**

* Индексация часто используемых полей, eager loading вместо N+1.
* Пагинация на всех списках, серверные фильтры.
* Кэширование справочников (Redis).
* Отделение воркеров/очередей от веб-узла.
* Горизонтальное масштабирование фронта и веб-части (статический фронт, sticky-less).
* Object Storage для медиа (S3) при росте объёмов.

**15. Наблюдаемость и резервирование**

* Логи приложения (storage/logs) + ротация.
* Метрики/трейсинг (опционально: Sentry, Prometheus/Grafana).
* Бэкапы БД (ежедневно, хранение 60 дней), бэкапы хранилища файлов по расписанию.
* Тестовые восстановления на стенде — ежемесячно.

**16. Интеграции**

* REST API для обмена с внешними ГИС/ERP/бухгалтерией (аутентификация по токену/клиентским сертификатам).
* Импорт точек/замеров из Excel/CSV по каталогу шаблонов.
* Веб-хуки/почтовые шлюзы (опционально) для уведомлений.

**17. Ограничения и допущения**

* Работа в современных браузерах (Chrome/Firefox/Edge/Safari, последняя и предпоследняя версии).
* Отсутствие офлайн-режима; требуется стабильное сетевое соединение.
* Обработку тяжёлых файлов нужно выносить в очереди (не в синхронные запросы).
* Ограничение на размер загружаемых медиа задаётся на уровне Nginx/PHP и приложения.

**Приложение A. Карта соответствия разделам интерфейса (по маршрутам)**

* Пользователи/Профиль/Роли-права: /users, /profile, /roles-permissions
* ПУИДД и реестр: /point/list, /points-registry, /district-card
* Заявки и планы: /request/\*, /schedule/list, /schedule/schedule-plan/:id, /point-plan/\*
* Назначения: /assignment (drag-and-drop карточки)
* АРМ менеджера: /manager (выполнение планов, загрузка фото, формы учёта, автозаполнение по прошлому году)
* Отчёты/документы: /report-intens, /act-ks2, /photo-report, /info-letter
* Сервисные разделы: /logs, /notifications, /gen-calendar, /map-print-live