# 

# 

# 

# 

7

# C:\Users\user\Downloads\lidar_bg_light.png

[1. Введение 4](#_Toc210294869)

[2. Краткий обзор решения 4](#_Toc210294870)

[3. Технологический стек 4](#_Toc210294871)

[4. Архитектура 5](#_Toc210294875)

[5. Жизненный цикл данных 6](#_Toc210294880)

[6. Функциональные возможности 6](#_Toc210294881)

[7. Руководство по запуску 7](#_Toc210294882)

[8. Работа с интерфейсом 8](#_Toc210294889)

[9. API и интеграция 9](#_Toc210294894)

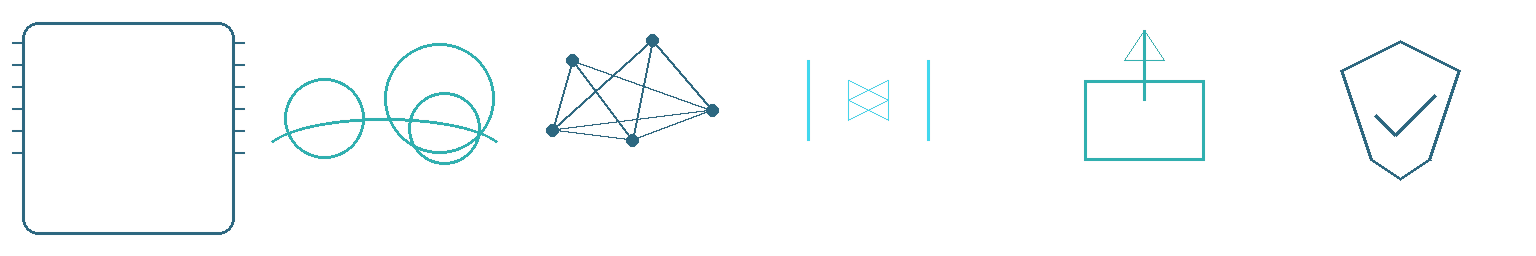
[10. Конфигурация и окружение 10](#_Toc210294901)

[11. Безопасность и эксплуатационные ограничения 10](#_Toc210294902)

[12. Мониторинг и сопровождение 11](#_Toc210294903)

[13. План развития и риски 11](#_Toc210294904)

[14. Приложения 11](#_Toc210294905)





# 1. Введение

Документ фиксирует текущее состояние платформы обработки облаков точек (далее — Платформа). Решение позволяет инженерным и аналитическим командам загружать, очищать и готовить LiDAR-данные.

Материал структурирован по техническим и эксплуатационным аспектам: архитектура, стек, сценарии использования, запуск, интеграция.

Версия 1.1 содержит расширенный раздел по API, уточнённые инструкции запуска и описание жизненного цикла данных.

# 

# C:\Users\user\Downloads\lidar_bg_light.png

# 2. Краткий обзор решения

Платформа покрывает полный цикл: импорт PCD-файлов, визуализация, ручное и автоматизированное редактирование, экспорт очищенных результатов.

* Backend на FastAPI управляет REST API, базой данных, файловым хранилищем и ML-пайплайном.
* Frontend на Vite + React + three.js реализует визуализацию облака и пользовательские сценарии.
* ML-компонент на PyTorch и Open3D выполняет предпросмотр и применение масок очистки.

Метаданные хранятся в SQLite (файл data/app.db), а файлы данных и экспортов — в каталоге data/{dataset\_id}. REST API одинаково доступен интерфейсу и внешним клиентам.

# 



# 3. Технологический стек

Технологический стек разделён на серверную часть, клиентскую часть и инфраструктурные компоненты. Ниже указаны ключевые технологии.

## 3.1 Backend

* Python 3.11, FastAPI, Uvicorn.
* SQLAlchemy 2.x и Pydantic v2 для описания моделей и схем ответов.
* Файловый слой в storage.py обеспечивает чтение и запись PCD и бинарных тайлов.
* ML-модуль использует PyTorch, Open3D, NumPy.

## 3.2 Frontend

* Vite, React 18, TypeScript, React Query.
* Three.js и пользовательские утилиты для работы с точками, кистью и статистикой.
* Ключевые компоненты: DatasetDetail, PointCloudEditor, AutoCleanPanel, RenderStats.

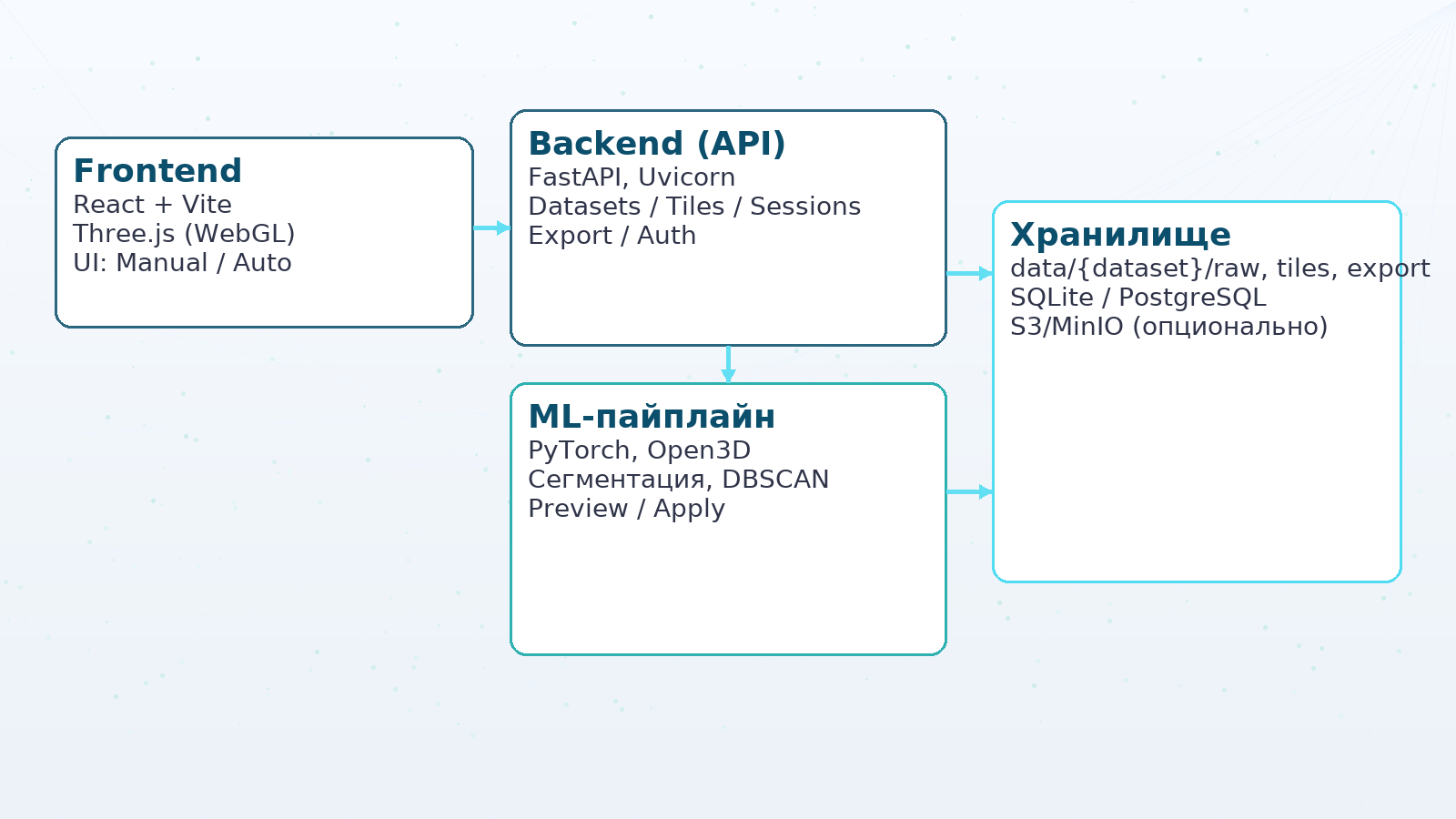
## 3.3 Инфраструктура и инструменты

* Хранение файлов: локальный каталог data/. Возможно подключение S3/MinIO.
* Логи: стандартный вывод Uvicorn и dev-сервера Vite, при необходимости подключается централизованный сбор.
* Контейнеризация: заготовки Dockerfile и docker-compose в каталоге infra/.
* Тестирование: pytest для backend и Vitest/React Testing Library для фронтенда (по мере расширения).

# C:\Users\user\Downloads\lidar_bg_light.png

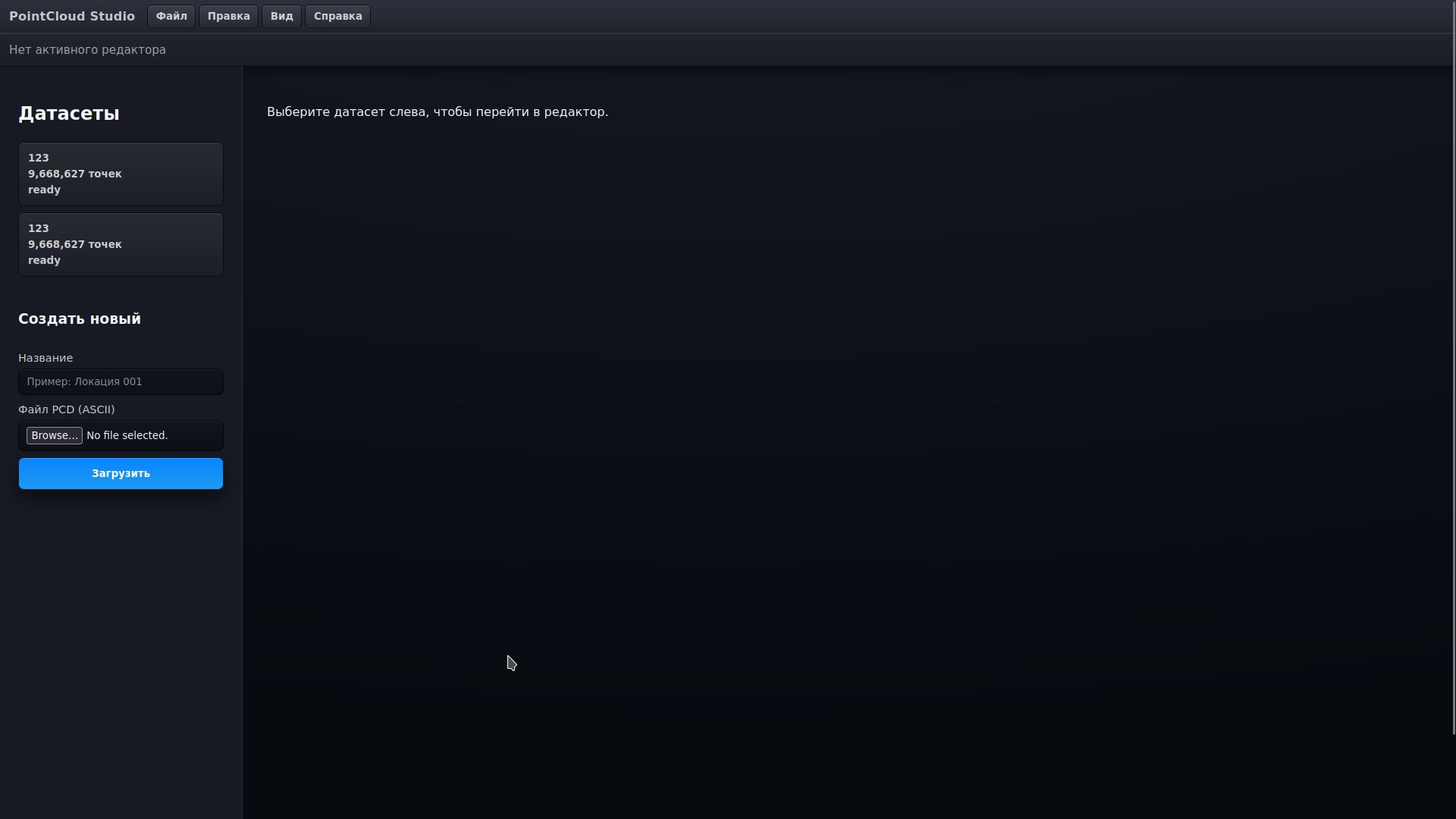
# 4. Архитектура

Архитектура построена вокруг REST API, WebGL-интерфейса и ML-пайплайна, работающих с общим файловым хранилищем

. 

## 4.1 Общий поток

Пользователь загружает PCD через UI или запрос /datasets/upload. Backend сохраняет файл, создаёт метаданные и разбивает облако на тайлы. Frontend по REST API получает информацию, отображает облако и предоставляет инструменты очистки.

ML-сервис по запросу формирует предпросмотр удаления объектов и возвращает статистику. Пользователь решает, применять ли маску.

## C:\Users\user\Downloads\lidar_bg_light.png

## 4.2 Backend

* Модуль datasets — создание датасетов, загрузка файлов, получение метаданных.
* Модуль tiles — выдача списка тайлов и скачивание конкретного тайла по координатам.
* Модуль sessions — управление сессиями редактирования с версионностью операций.
* Модуль export — формирование конечного processed\_points.pcd и возврат статистики.
* Модуль ml — предпросмотр и применение масок очистки.

## 4.3 Frontend

DatasetDetail агрегирует данные датасета и сессий. PointCloudEditor управляет WebGL-сценой и инструментами. AutoCleanPanel взаимодействует с ML-API.

Компонент RenderStats отображает ключевые метрики: количество загруженных точек, тайлов и среднее время кадра.

## 

## 4.4 ML-пайплайн

Pipeline в app/backend/ml/pipeline.py сегментирует грунт, выполняет кластеризацию DBSCAN, задаёт эвристические метки и при необходимости классифицирует кластеры нейросетью.Предпросмотр хранится в памяти процесса: mask, labels, stats. Применение маски создаёт новый файл с постфиксом\_clean.

# C:\Users\user\Downloads\lidar_bg_light.png

# 5. Жизненный цикл данных

* Импорт: PCD сохраняется в data/{dataset\_id}/raw, создаётся запись в таблице datasets.
* Разбиение: формируются бинарные тайлы и индексы, записываются в таблицу tiles.
* Редактирование: операции хранятся в таблицах sessions и operations, что позволяет отслеживать историю.
* Предпросмотр: ML создаёт маску, статистику по классам и возвращает идентификатор preview\_id.
* Применение: маска превращается в новый файл, путь сохраняется в UI и может быть использован для экспорта.
* Экспорт: backend комбинирует ручные и автоматические изменения и формирует processed\_points.pcd.

# C:\Users\user\Downloads\lidar_fig_data_lifecycle_light.png



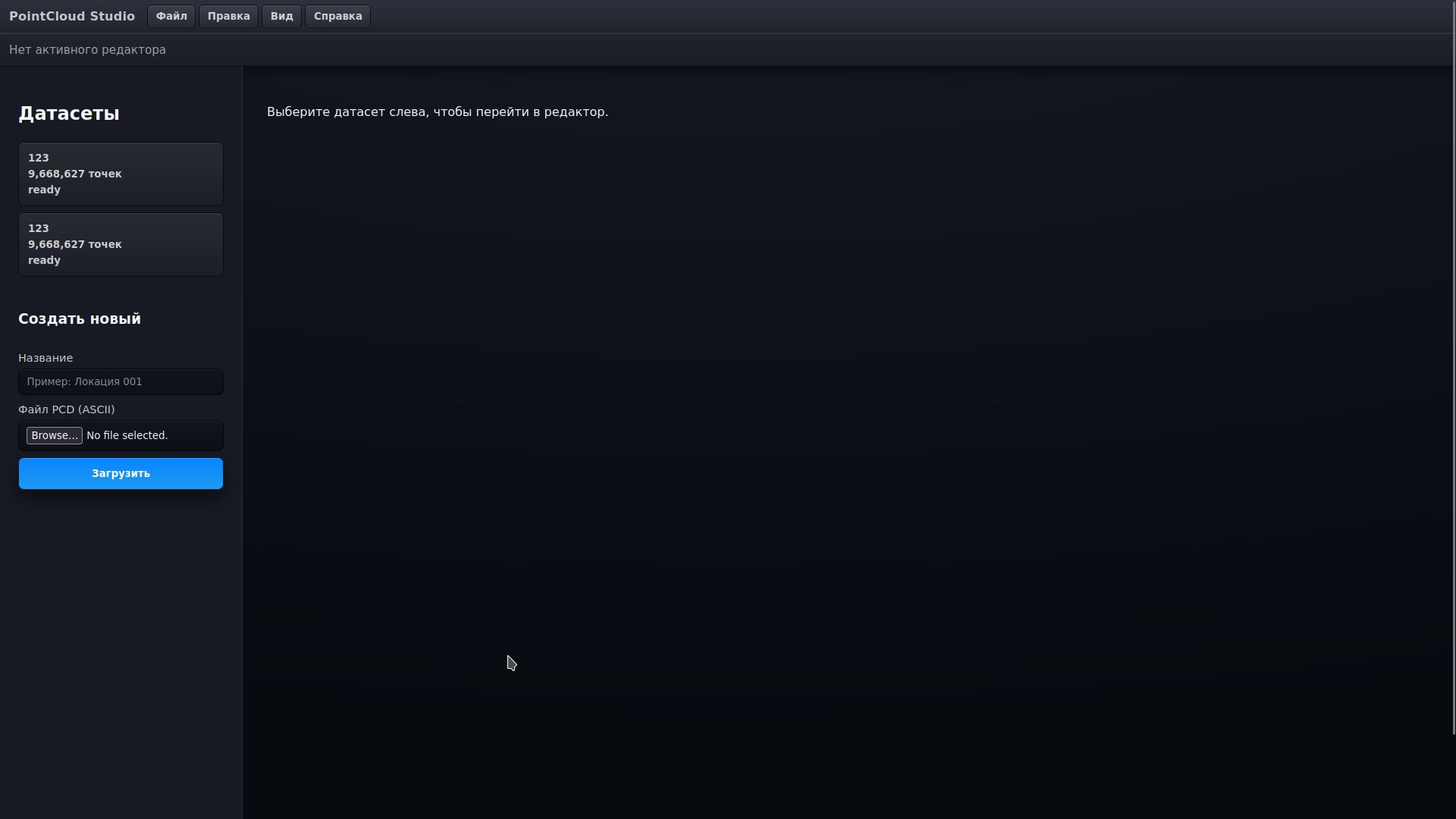
# 6. Функциональные возможности

* Управление датасетами, просмотр метаданных, фильтрация по статусу.
* Интерактивная визуализация: вращение, масштабирование, адаптивный порог выбора точек.
* Кисть удаления и восстановления, настройка радиуса, размера точек, шага по пикселям.
* Создание и переключение сессий редактирования, предотвращение конфликтов по baseVersion.
* Auto Clean: предпросмотр, выбор классов для удаления, применение маски.
* Экспорт: формирование обработанного PCD и предоставление статистики по точкам.
* Индикация загрузки тайлов и метрик визуализации.
* Возможность интеграции через REST API для автоматизации загрузок и выгрузок.

# 



# 7. Руководство по запуску

Руководство охватывает локальный и контейнерный запуск, а также требования к окружению и диагностику. 

## 7.1 Системные требования

* ОС: Windows 10/11, macOS, Linux (Ubuntu 22.04+).
* Python 3.11 и Node.js 18 или выше. npm 9+.
* Оперативная память: 16 ГБ и более для облаков свыше 10 млн точек.
* GPU с поддержкой CUDA рекомендуется для ускорения ML-пайплайна.

## 



## 7.2 Backend (локальный запуск)

Команды для запуска backend:

* python -m venv .venv
* source .venv/bin/activate # Windows: .venv\Scripts\activate
* pip install -r app/backend/requirements.txt
* uvicorn app.backend.main:app --reload --port 8000

API будет доступно по адресу http://localhost:8000. База данных создаётся в data/app.db.

## 7.3 Frontend (локальный запуск)

Команды для запуска frontend:

* cd app/frontend
* npm install
* npm run dev

Интерфейс доступен на http://localhost:5173. Запросы /api проксируются на backend. Для внешнего backend используйте VITE\_API\_BASE.

## 7.5 ML-пайплайн

* Чекпоинты нейросетей хранятся локально; путь передаётся в параметре checkpoint запроса /ml/preview.
* Для работы на GPU требуется установленная CUDA и версия PyTorch с поддержкой CUDA.
* При запуске на CPU обработка занимает больше времени (в среднем в 3–5 раз).

# 



# 8. Работа с интерфейсом

Интерфейс делится на 3D-сцену, панель статистики и боковую панель режимов.

## 8.1 Основные области

* 3D-сцена — визуализация облака с режимом обзора Orbit.
* Панель статистики — количество загруженных точек, тайлов, показатели производительности.
* Боковая панель — переключение режимов Manual/Auto, управление сессиями, кнопки экспорта.

## 8.2 Режим Manual

* Инструменты: delete, restore, inspect.
* Регулировки: радиус кисти, размер точек, шаг по пикселям.
* Управление: левая кнопка мыши — применение инструмента, правая — вращение камеры, колесо — масштаб.
* Сброс и применение выделения доступны из UI и вызывают функции resetSelection и applySelection.

## 8.3 Режим Auto

* Настраиваемые параметры предпросмотра: voxel\_size, eps, min\_points, тип модели, путь к checkpoint.
* Интерфейс показывает статистику по классам и позволяет выбрать классы для удаления.
* После применения создаётся файл с постфиксом \_clean, который можно просмотреть и экспортировать.

## 8.4 Советы

* Shift + колесо — ускоренное масштабирование.
* Ctrl + колесо — изменение радиуса кисти (в браузерах Chrome и Edge).



# 9. API и интеграция

REST API сгруппировано по доменным областям. Базовый путь по умолчанию /api. 

## 9.1 Ресурс datasets

* GET /datasets/ — список датасетов с сортировкой по дате создания.
* POST /datasets/ — создание записи без файла для преднастроенных ETL сценариев.
* POST /datasets/upload — загрузка PCD с автоматическим построением тайлов. Допускает файлы до 10 ГиБ.
* GET /datasets/{dataset\_id} — получение полных метаданных датасета.

## C:\Users\user\Downloads\lidar_bg_light.png

## 9.2 Ресурс tiles

* GET /datasets/{dataset\_id}/tiles/ — перечень тайлов с координатами и количеством точек.
* GET /datasets/{dataset\_id}/tiles/{z}/{x}/{y} — скачивание бинарного тайла.

## 

## 9.3 Ресурс sessions

* GET /datasets/{dataset\_id}/sessions/ — список сессий редактирования.
* POST /datasets/{dataset\_id}/sessions/ — создание новой сессии.
* PATCH /datasets/{dataset\_id}/sessions/{session\_id}/ops — добавление операций с проверкой baseVersion.
* GET /datasets/{dataset\_id}/sessions/{session\_id} — просмотр конкретной сессии.

## 9.4 Ресурс export

* POST /datasets/{dataset\_id}/export/ — инициировать экспорт и получить информацию о задаче.
* GET /datasets/{dataset\_id}/export/ — скачать processed\_points.pcd. 404, если файл ещё не создан.

## 9.5 Ресурс ml

* POST /ml/preview — сформировать предпросмотр, вернуть preview\_id и статистику.
* POST /ml/apply — применить маску. Параметр classes\_to\_remove позволяет ограничить список классов.
* Ошибки: 404 при отсутствии предпросмотра, 400 при некорректных параметрах.



## 9.6 Интеграционные сценарии

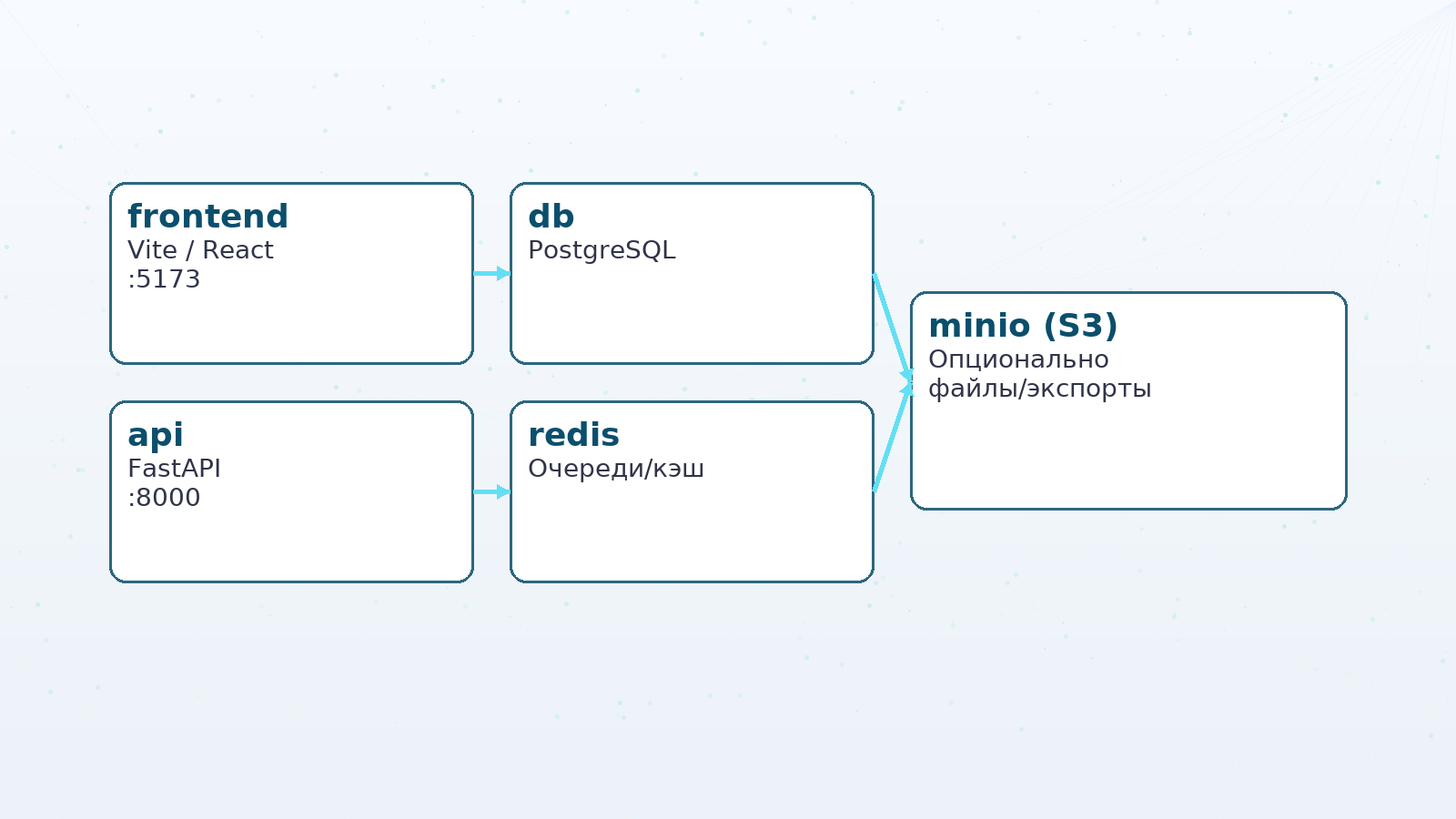
* Рекомендуется размещать backend за API Gateway и добавлять авторизацию (например, JWT или OAuth).
* Экспорт и предпросмотр можно вызывать из сторонних скриптов для построения конвейеров обработки.
* Для больших массивов данных планируется поддержка пакетного экспорта и асинхронных очередей.

# 



# 10. Конфигурация и окружение

* Переменные backend: APP\_NAME, DATABASE\_URL (по умолчанию sqlite:///data/app.db), LOG\_LEVEL.
* Переменные frontend: VITE\_API\_BASE (адрес API), VITE\_ENABLE\_ML (флаг отображения панели Auto Clean).
* Структура data/: app.db, каталоги raw, tiles, export для каждого датасета





11. Безопасность и эксплуатационные ограничения

* Авторизация не реализована. Перед продуктивным запуском требуется SSO, JWT или иная схема аутентификации.
* Лимиты: загрузка до 10 ГиБ за один файл, уверенная работа UI до 20 млн точек при 30 FPS на средней видеокарте.
* Файлы не шифруются. Необходимо ограничить доступ к серверу и каталогу data/.
* Логи содержат пути к файлам. Настроить политику хранения и очистки логов.
* Следить за объёмом диска, так как экспорт создаёт дополнительные копии облаков.

# 



# 12. Мониторинг и сопровождение

* Метрики: активные датасеты, количество сессий, время обработки предпросмотра, FPS в интерфейсе.
* Интеграция с системой логирования (ELK, Loki) рекомендуется для продуктивной среды.
* Периодически выполнять тестовые загрузки и экспорты для провеи зависимостей.
* CI/CD: автоматические проверки зависимостей, линтеров и unit-тестов.

# 



# 13. План развития и риски

* Запланированные улучшения:
* Введение ролей и авторизации с разграничением права редактирования.
* Фоновые воркеры для тяжёлых задач (Celery + Redis).
* Интеграция с объектным хранилищем S3/MinIO для крупных проектов.
* Новые модели и эвристики (включая segment-anything).
* Риски:
* Нехватка GPU при массовом запуске автоочистки.
* Рост объёма диска из-за множества экспортов.
* Отсутствие авторизации в базовой поставке.
* Необходимость регулярного обновления библиотек для предотвращения уязвимостей.

# 



# 14. Приложения

Приложения содержат структуру репозитория, форматы данных и словарь терминов.

## 14.1 Структура репозитория

* app/backend/ — серверная часть (FastAPI, ML).
* app/frontend/ — интерфейс на React.
* configs/ — дополнительные конфигурации и шаблоны.
* data/ — локальное файловое хранилище (не хранится в VCS).
* docs/ — документация и вспомогательные материалы.

## 14.2 Форматы данных

* Поддерживаемые входные форматы: PCD ASCII, Binary, Binary compressed.
* Экспортируемый формат: PCD ASCII для совместимости с инструментами обработки.
* Тайлы: бинарный формат .bin с массивом точек и индексами в исходном облаке.
* Датасет — загруженное облако точек с метаданными.
* Сессия — набор операций редактирования с версионностью.
* Тайл — часть облака для ленивой подгрузки и рендеринга.
* Предпросмотр — промежуточная маска очищаемых точек, созданная ML-пайплайном.