Платформа учета и контроля строительных объектов “Периметр”

**[1. Общая информация 3](#_fztxco4do507)**

[1.1. Краткое описание платформы 3](#_szer7vkkj15z)

[1.2. Перечень решаемых задач 3](#_hd464xczwjcn)

[1.3. Функциональные возможности 3](#_xslbz07gz391)

**[2. Бэкенд 5](#_7n3bksm6qpf7)**

[2.1. Общее описание 5](#_7yn2fgr8mz7a)

[2.2. Основной сервер 5](#_ddgeg1kfc7nv)

[2.3. База данных 5](#_82znthdhwixr)

[2.4. Тайлер карт 5](#_ch8r0mqserw8)

[2.5. Хранилище файлов 5](#_emb7eknbtqux)

[2.6. CV и обработка изображений 6](#_cjwkcw2mxmuj)

**[3. Фронтенд 7](#_hvxdkyshu7fb)**

[3.1. Основной фреймворк 7](#_wjzyucch50z4)

**[4. Деплой 8](#_1q6k0fawn4xc)**

# Общая информация

## Краткое описание платформы

Платформа представляет собой базу строительных объектов и обладает функционалом для изменения данных объектов в процессе работ.

Внутри платформы используется ролевая система для разграничения доступа пользователей к функционалу платформы.

Дизайн платформы можно увидеть в презентации проекта: [It-Конторские - Задача 3.pptx](https://docs.google.com/presentation/d/1hybZqmmmmtK3QYRe0vadY2ffunc_XlSM/edit?usp=drive_link&ouid=105588688462785928280&rtpof=true&sd=true).

Скринкаст прототипа можно увидеть по ссылке: [Screencast.mp4](https://drive.google.com/file/d/1TSwPQo1J4ylLCJSHJTRewQ9PFnnd-1FR/view?usp=drive_link)  
Ссылки на Hithub:  
<https://github.com/iandrey77888/hackathon-backend>

<https://github.com/iandrey77888/hackathon-frontend>

## Перечень решаемых задач

Платформа решает следующие задачи:ввв

* Повышение качества донесения информации об объектах
* Удобство получения информации об объектах
* Контроль внесения важной информации об объекте только при нахождении пользователя на объекте (прием поставок, нарушения, замечания, инициация лабораторного контроля и т.п.)
* Упрощение процесса приема поставки путем автоматизации внесения данных о поставке в систему за счет машинного зрения (обработка фотографии накладной и выделение из нее необходимых данных для заполнения внутри системы)
* Прозрачность процесса выполнения работ на объекте путем ведения историй изменений в рамках объекта (хранение всех выписанных нарушений и замечаний, хранение всех результатов лабораторных контролей, ведение истории пополнения и расхода материалов, ведение истории выполнения каждого этапа работ в объемных единицах)

## Функциональные возможности

Функционал платформы доступный пользователю напрямую зависит от его роли внутри платформы (Служба Строительного Контроля (ССК), Подрядчик, Инспекционный Контрольный Орган (ИКО)).

Платформа обладает следующим функционалом:

* Просмотр сводной информации обо всех объектах на главном экране (доступно всем)
* Просмотр интерактивной карты объектов (доступно всем)
* Просмотр интерактивной карты нарушений и замечаний выписанных на объекте (доступно всем)
* Просмотр сводной информации об объектах на главном экране (доступно всем)
* Изменение ответственных за объект - ответственного со стороны подрядчика и ответственных со стороны инспекционных контрольных органов (доступно только ССК)
* Создание замечаний на объекте. При необходимости к замечанию можно прикрепить документ или фотографию (доступно только ССК)
* Принятие или отклонение исправленных подрядчиком замечаний (доступно только ССК)
* Создание нарушений на объекте. При необходимости к замечанию можно прикрепить документ или фотографию (доступно только ИКО)
* Принятие или отклонение исправленных подрядчиком нарушений (доступно только ИКО)
* Исправление нарушений и замечаний на объекте (доступно только Подрядчику)
* Просмотр графика работ на объекте (доступно всем)
* Просмотр детальной информации об этапе работ (доступно всем)
* Внесение изменений в график работ (доступно только ССК и Подрядчику)
* Принятие или отклонение предложенных изменений в график работ (доступно только ССК и Подрядчику)
* Закрытие рабочего дня с указанием прогресса выполненных работ и расхода материалов по каждому из активных этапов работ (доступно только Подрядчику)
* Прием поставок с использованием машинного зрения - подрядчик фотографирует накладную поставки в приложении и ожидает окончания обработки. По окончанию обработки - проверяет правильность заполненных полей, и либо принимает поставку, либо отклоняет ее с указанием причины. При необходимости к каждому материалу можно прикрепить дополнительные документы или фотографии. (доступно только Подрядчику)
* Просмотр склада объекта - материалов используемых на объекте и их количество на момент просмотра (доступно только Подрядчику и ИКО)
* Инициация лабораторного контроля на определенный материал из поставки (доступно только ИКО)
* Просмотр результатов лабораторного контроля на материалы (доступно только ИКО)

# Бэкенд

## Общее описание

Для реализации прототипа было принято решение использовать python с FastAPI как основу для быстрой реализации требуемого функционала. Также это позволяет в будущем произвести переход на более производительное решение путем замены текущего сервера на новый с совместимым API.

Бэкенд представляет собой набор сервисов для frontend приложения, обеспечивающих его данными для работы.

## Основной сервер

Основной backend-сервер представляет собой python приложение, написанное с использованием фреймворка FastAPI, предоставляющее frontend серверу REST API для получения и управления данными.

Для валидации JSON схем в запросах и ответах использован pydantic, для взаимодействия с базой данных был выбран ORM SQLAlchemy 2.

Параллельность обработки данных достигается за счет совместного использования Uvicorn в качестве ASGI сервера и Gunicorn в роли менеджера процессов для Uvicorn

## База данных

В качестве базы данных был выбран open-source PostgreSQL 17 с дополнением PostGis для хранения геоданных полигонов и точек.

Схема базы (приведена в папке docs) была разработана на основе требований заказчика.

## Тайлер карт

Система использует self-hosted тайлер карт [TileServer GL](https://github.com/maptiler/tileserver-gl/), распространяемый по открытой лицензии BSD 2.

Файлы используемых тайлов получены с [OpenStreetMap](https://www.openstreetmap.org/), имеющей лицензию [ODbL 1.0](https://opendatacommons.org/licenses/odbl/)

## Хранилище файлов

В качестве хранилища файлов было выбрано open-source s3-like хранилище [MinIO](https://github.com/minio/minio), распространяемое по лицензии GNU AGPLv3

## CV и обработка изображений

Для решения задач обработки изображений был выбран пайплайн из предобработки изображений с использованием [OpenCV](https://github.com/opencv/opencv) и [scikit-image](https://github.com/scikit-image/scikit-image), после чего изображение передается на локальный сервер [EasyOCR](https://github.com/JaidedAI/EasyOCR) , данные которого отправляются с запросом на формирование JSON ответа в LLM [Gemma3:4b](https://ollama.com/library/gemma3:4b) поднятую в [Ollama](https://ollama.com/).

Все элементы open-source и доступны для разворачивания в закрытом контуре. Благодаря тому, что для backend сервера CV сервер это просто внешний сервис, возможно в будущем заменить пайплайн или его элементы на более подходящие для решения задач

Функционал был проверен и частично реализован, но не был интегрирован в сервер в связи с нехваткой времени.

# Фронтенд

## Основной фреймворк

В связи с наличием требований по реализации как web, так и мобильных версий, было принято решение использовать React Native с фреймворком Expo для кросс-платформенной разработки. Expo предоставляет возможность компиляции нативных приложений под Android/IOS и отдельно web версии.

Единая кодовая база с возможностью разделения специфических участков также позволила упростить разработку для кросс-платформы

[не реализовано, планировалось]

Для работы в offline режиме планировалось выполнять кэширование данных об объекте от сервера и разрешить использовать их как временные референсные данные, чтобы приложение принимало решение о нахождении на объекте, позволяя создавать замечания и выполнять действия, сохраняя плановые пакеты в кэш после провала отпарвки. Когда приложение получало бы доступ в интернет, оно бы отправляло пакеты для валидации на сервере, после чего тот бы принимал решение о выполнении действия или отказе на основе данных геолокации.

# Деплой

Систему возможно развернуть в закрытом контуре благодаря open-source характеру ее элементов. Для этого необходимо выполнить шаги, описанные в [док на гите], после чего выполнить скрипты создания базы данных из файла docs/db.sql

На сервере, где будет поднят основной backend сервер необходимо установить переменные среды:

* DB\_URL - строка подключения к базе данных с указанием в ней логина и пароля в формате postgresql+psycopg2://%имя пользователя%:%пароль%@%адрес базы%:%порт базы%/%имя базы%
* MINIO\_URL - адрес развернутого MinIO
* MINIO\_ACCESS - access\_key пользователя в MinIO
* MINIO\_SECRET - secret\_key пользователя в MinIO

Для backend сервера потребуется собрать образ из Dockerfile (в корне репозитория), установив эти переменные  
Остальные сервисы можно поднять с помощью compose файла docker-compose.yml из корня проекта  
Для frontend - потребуется установить Android SDK и переменную среды ANDROID\_HOME, а также node.js версии v20.19.4.   
После чего перейти в корень фронтенд проекта, провести npm install, а затем npx expo run:andorid