Общество с ограниченной ответственностью «К-софт»
Юридический адрес: РФ, 302025, Орловская обл.,
г. Орёл, Московское ш, д. 137 к. 1, офис 82
ОГРН - 1215700003276 ИНН - 5751066082 КПП - 575101001
57.ksoft@gmail.com www.ksofteh.ru

ОБЩЕСТВО С ОГРАНИЧЕННОЙ ОТВЕТСТВЕННОСТЬЮ

«К-СОФТ»

302025, Орловская область, г. Орёл, Московское шоссе, д. 137, к. 1, офис 82,

тел/факс: (4862)-444-001, e-mail: 57.ksoft@gmail.com

ОКПО 72188653, ОГРН 1215700003276, ИНН/КПП 5751066082/575101001

Р.О.С.Дор.Контроль

Инструкция по установке

2026

Оглавление

1 Общие положения 3

1.1 Наименование системы 3

1.2 Условное обозначение системы 3

1.3 Функциональное назначение 3

2 Архитектура Системы 4

3 Программно-аппаратные требования 6

3.1 Аппаратные требования к серверной части 6

3.2 Программное обеспечение серверной части 6

3.3 Требования к рабочим местам 6

3.4 Программное обеспечение клиентской части 7

4 Установка программного обеспечения 7

4.1 Порядок установки 7

4.1.1 Серверное приложение 7

4.1.2 Сервис авторизации dm-auth 11

4.1.3 Клиентское приложение dm-client 12

1 Общие положения

1.1 Наименование системы

Р.О.С.Дор.Контроль.

1.2 Условное обозначение системы

Р.О.С.Дор.Контроль, Система.

1.3 Функциональное назначение

Р.О.С.Дор.Контроль – является централизованным информационно-аналитическим хранилищем и сервисом мониторинга для задач, связанных с обеспечением безопасности и контроля в сфере дорожного движения и городской инфраструктуры.

Основное функциональное назначение Системы заключается в консолидации узконаправленных тематических данных из разнородных внешних источников и предоставлении пользователям единого инструментария для их просмотра, анализа и мониторинга в реальном времени.

Для реализации своего назначения Система обеспечивает следующий набор ключевых функций:

- Административно-справочные функции:
  - Управление учётными записями и правами доступа пользователей.
  - Формирование и ведение нормативно-справочной информации (НСИ).
- Мониторинг транспортного комплекса и дорожной инфраструктуры:
  - Контроль местоположения, маршрутов и работы общественного транспорта и коммунальной техники.
  - Мониторинг состояния дорожного покрытия и дорожной ситуации.

- Контроль работы и состояния оборудования Интеллектуальных Транспортных Систем (ИТС).
- Мониторинг данных с метеостанций (погодные условия).

- Контроль сопутствующих служб и сервисов:
  - Контроль деятельности службы эвакуации транспортных средств.
  - Мониторинг занятости и работы парковочных пространств.
  - Контроль информационных носителей (мониторинг работы остановочных таблиц).
- Управление информационным наполнением и коммуникацией:
  - Мониторинг процессов публикации и актуальности контента: новостных лент, обращений граждан, опросов и анонсов событий.

2 Архитектура Системы

Система «Р.О.С.Дор.Контроль» реализована по принципу архитектуры «Единого окна» (Single Pane of Glass). Это означает, что её ключевая задача — агрегировать разнородные данные из множества автономных и технологически не связанных между собой внешних систем городской инфраструктуры и предоставлять их через единый, целостный веб-интерфейс. Для реализации этой концепции применена многоуровневая сервис-ориентированная архитектура: на первом уровне разнородные внешние источники данных подключаются через специализированные коннекторы интеграционного слоя, которые нормализуют и передают информацию в ядро системы — набор независимых сервисов бизнес-логики, а результаты их работы сохраняются в гибридном хранилище данных и визуализируются через единый веб-интерфейс (см. Рисунок 1). Такая архитектура чётко разделяет ответственность между компонентами, обеспечивая гибкость, масштабируемость и устойчивость.

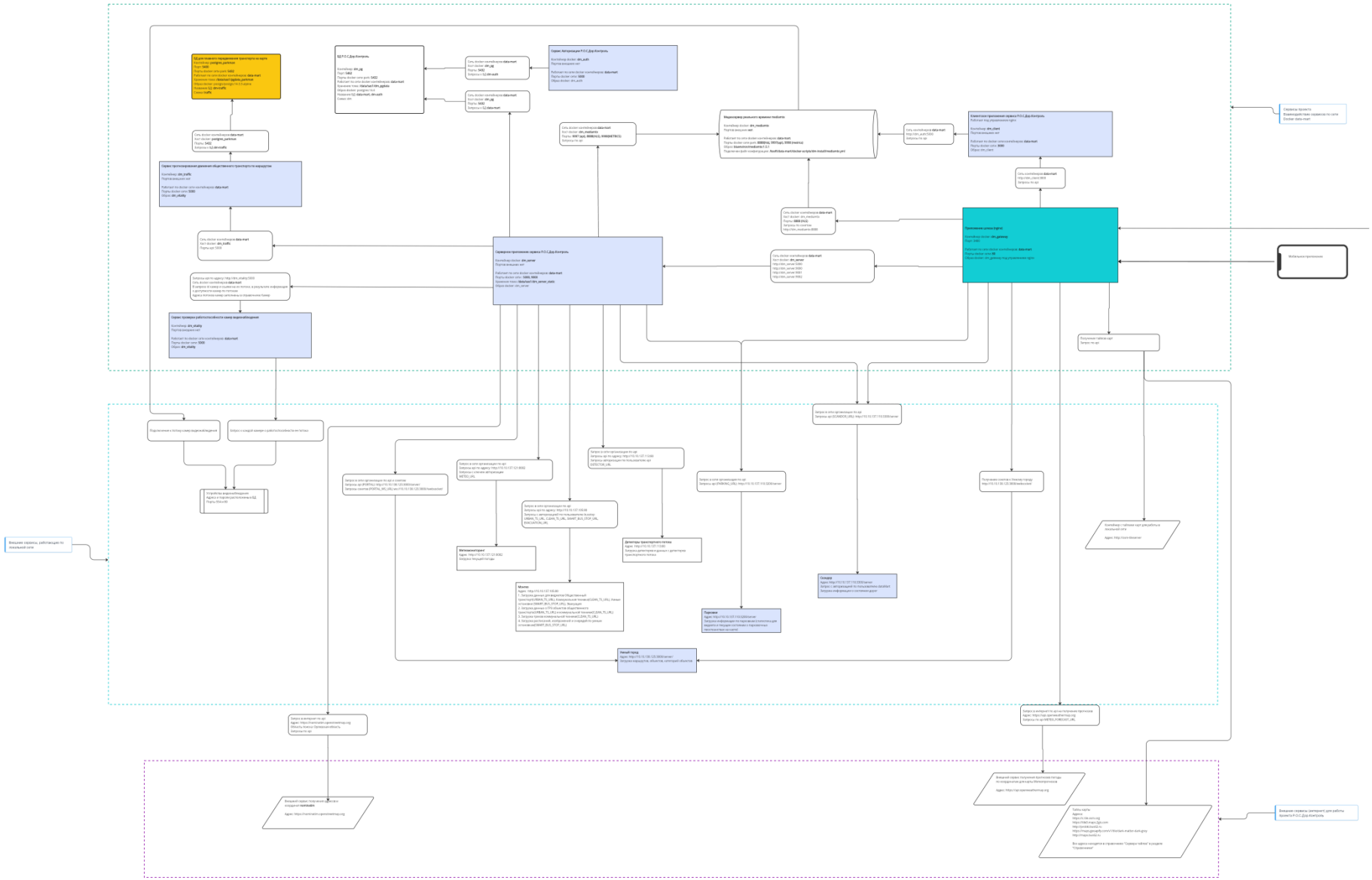


Рисунок 1 – Архитектура Системы

3 Программно-аппаратные требования

3.1 Аппаратные требования к серверной части

Таблица 1 – Рекомендуемый минимальный КТС сервиса.

Описание	Параметры
Процессор	не ниже Intel Xeon 8 Core, с тактовой частотой на менее 2.4 Ghz
Количество ядер процессора, шт.	10
RAM, ГБ	32
Ёмкость SSD-диска, ГБ	100
Ёмкость HDD-диска, Тб	500
Минимальная пропускная способность канала	30 Мбит/с
Некомендуемая пропускная способность канала	100 Мбит/с

3.2 Программное обеспечение серверной части

Для функционирования Системы требуется следующее программное обеспечение:

- в качестве основной операционной системе сервера – ОС семейства Linux;
- программное обеспечение для автоматизации развёртывания и управления приложениями в средах с поддержкой контейнеризации Docker.

3.3 Требования к рабочим местам

Рабочие места должны удовлетворять следующим требованиям:

СанПиН 2.2.1/2.1.1.1278-03 - Гигиенические требования к естественному, искусственному и совмещенному освещению жилых и общественных зданий.

СанПиН 2.2.2.1332-03 - Гигиенические требования к организации работы на копировально-множительной технике.

СанПиН 2.2.2/2.4.1340-03 - Гигиенические требования к персональным электронно-вычислительным машинам и организации работы.

3.4 Программное обеспечение клиентской части

На клиенте Системы не требуется предварительной установки каких-либо программных модулей. Работа с Системой осуществляется с помощью интернет-браузера. Для нормального функционирования клиентского АРМ рекомендуется использование рабочих станций со следующими характеристиками:

- центральный процессор с тактовой частотой от 2.4 Гц с двумя или более ядрами;
- объем оперативной памяти - 2Гб и более;

Рекомендовано использование Full HD монитора с разрешением экрана 1920x1080. На мониторах с меньшим разрешением отображение и функционал сайта могут отображаться некорректно.

Веб-браузер:

- Google Chrome 128+.

С целью обеспечения быстродействия системы, минимальная скорость подключения должна составлять не менее 2 мБит/сек.

4 Установка программного обеспечения

4.1 Порядок установки

Для работы сервисов Р.О.С.Дор.Контроль требуется сервер с установленным docker и docker-compose (ОС любая, где можно поставить docker и docker-compose).

Для запуска каждого сервиса нужно использовать команду

docker-compose up –d название_сервиса

4.1.1 Серверное приложение

Основной запуск сервера осуществляется с помощью команды

`npm start`

Команда запускает файл app.js в директории src

Запуск скриптов миграций для БД

```
"migrate:up": "npx sequelize-cli db:migrate"
"migrate:down": "npx sequelize-cli db:migrate:undo"
```

Переменные окружения

Для запуска сервера и корректной его работы требуется использовать переменные окружения:

```
// окружение
NODE_ENV=testing || production (или любое другие окружение из директории /src/config)

// приложение
APP_PORT = 5000 - по какому порту запущено приложение
WS_STREAMING_PORT = 9000 - включение отключение канала на MediaMTX сервере

// база данных
DB_HOST = "host" - хост БД
DB_PORT = "port" - порт БД
DB_NAME = "name" - имя схемы БД
DB_USER = "user" - пользователь БД
DB_PASSWORD = "pass" - пароль БД
DB_SYNC = true || false - синхронизация БД
DB_INIT = true || false - первоначальное заполнение БД

// для работы с контейнером mediamtx
MEDIAMTX_SERVICE_HOST=mediamtx
MEDIAMTX_SERVICE_API_PORT=8889
MEDIAMTX_SERVICE_HLS_PORT=8888
MEDIAMTX_SERVICE_METRICS_PORT=9998

// логирование
LOG_COLORIZE=true - делать подсветку или нет
LOG_LEVEL = "info" - уровень лога
Уровни логов:
error
warn
info
verbose
debug
```

```
silly

// подключение SWAGGER (для запуска production всегда false)
SWAGGER = true || false

// nominatim (запрос адресов)
NOMINATIM_URL = https://nominatim.openstreetmap.org

// метеомониторинг
METEO_URL = http://localhost:882
METEO_AUTH = login:pass - в base64, например VXNlcjpidXJzdHJveQ==

// портал Умный город
PORTAL = "https://smartcity.its57.ru/"

// загрузка маршрутов с умного города
ROUTE_LOAD = true
ROUTE_CRON = "0 0 * * *"

// загрузка объектов  с умного города
OBJECT_LOAD = false
OBJECT_CRON = "* * * * *"

// загрузка ГРЗ и наличия трека с монгео
OBJECT_REG_NUMBER_LOAD: false,
OBJECT_REG_NUMBER_CRON: "0 0 * * *",
OBJECT_TRACK_LOAD: false,
OBJECT_TRACK_CRON: "0 * * * *",

// Скандор
SCANDOR_URL = "http://localhost:3300/server"
SCANDOR_AUTH_URL = "http://localhost:3300/auth"
SCANDOR_LOGIN = "login"
SCANDOR_PASSWORD = "password"
// Сике
SNOW_SEEKER_URL = "http://localhost:3300/server"

// Парковки
PARKING_URL = "http://localhost:3200/server"
PARKING_LOAD = false,
PARKING_CRON = "0 8 * * *",

//Монгео
// общественный транспорт
URBAN_TS_URL = http://test.mongeo.ru
```

```
URBAN_TS_LOGIN = login
URBAN_TS_PASSWORD = password

// коммунальный транспорт
CLEAN_TS_URL = http://test.mongeo.ru
CLEAN_TS_LOGIN = login
CLEAN_TS_PASSWORD = password

// умные остановки
SMART_BUS_STOP_URL = http://cloud.its57.ru
SMART_BUS_STOP_LOGIN = login
SMART_BUS_STOP_PASSWORD = password
SMART_BUS_STOP_LOAD = false,
SMART_BUS_STOP_CRON = "0 6,18 * * *",
SMART_BUS_STOP_FORECAST_LOAD = false,
SMART_BUS_STOP_FORECAST_CRON = "*/1 * * * *",
SMART_BUS_STOP_QUEUE_LOAD = false,
SMART_BUS_STOP_QUEUE_CRON = "*/2 * * * *",
SMART_BUS_STOP_IMAGE_LOAD = false,
SMART_BUS_STOP_IMAGE_CRON = "*/3 * * * *",

// эвакуация
EVACUATION_URL=http://test.mongeo.ru
EVACUATION_LOGIN=login
EVACUATION_PASSWORD=password

// состояние камер
VITALITY_URL: http://localhost:5003
VITALITY_CHECK: "false",
VITALITY_CRON: "*/5 * * * *",

// детекторы
DETECTOR_ENABLE: false,
DETECTOR_URL: "http://host:port",
DETECTOR_CRON: "0 1 * * *",
DETECTOR_DATA_CRON: "*/5 * * * *",
DETECTOR_LOGIN: "login",
DETECTOR_PASSWORD: "pass",
DETECTOR_INTERVAL: 300, // получать данные по умолчанию за 5 минут

//наши детекторы
OUR_DETECTOR_ENABLE: false,
OUR_DETECTOR_URL: "http://localhost:3022",
OUR_DETECTOR_CRON: "0 1 * * *",
OUR_DETECTOR_DATA_CRON: "*/5 * * * *"
```

// сервис плавного передвижения
TRAFFIC_URL: "http://localhost:4998",
TRAFFIC_ROUTES_CRON: "0 0 * * *",
TRAFFIC_ROUTES_ENABLE: false,
TRAFFIC_WS_ENABLE: false,
```

Перед первым запуском sma-gis для БД выполнить команду

```
CREATE EXTENSION postgis;
```

Если обновляется контейнер сервера, то требуется обязательно перезапустить контейнер шлюза dm_gateway

4.1.2 Сервис авторизации dm-auth

Основной запуск сервера осуществляется с помощью команды:

`npm start`

Команда запускает файл app.js в директории src

Переменные окружения

Для запуска сервера и корректной его работы требуется использовать переменные окружения:

```dotenv
// окружение
NODE_ENV=testing || production (или любое другие окружение из директории /src/config)

// приложение
APP_PORT = 5000 - по какому порту запущено приложение

// начальный пользователь (будет создан, если нет пользователей в бд)
USER_LOGIN = admin - логин
USER_PASSWORD = admin - пароль

// база данных
DB_HOST = "host" - хост БД
DB_PORT = "port" - порт БД
```
```

DB_NAME = "name" - имя схемы БД
DB_USER = "user" - пользователь БД
DB_PASSWORD = "pass" - пароль БД
DB_SYNC = true || false - синхронизация БД
DB_INIT = true || false - первоначальное заполнение БД

// логирование
LOG_COLORIZE=true - делать подсветку или нет
LOG_LEVEL = "info" - уровень лога
Уровни логов:
error
warn
info
verbose
debug
silly

// подключение SWAGGER (для запуска production всегда false)
SWAGGER = true || false

// ключи токенов
JWT_ACCESS_SECRET=secret-access
JWT_REFRESH_SECRET=secret-refresh
JWT_EXPIRES_TIME=3600000
```

Если обновляется контейнер, то требуется обязательно перезапустить контейнер шлюза rd_gateway

### 4.1.3 Клиентское приложение dm-client

В каталоге проекта вы можете запустить:

`npm start`

Запускается приложение в режиме разработки.
Открыть [http://localhost:3000 ](http://localhost:3000), чтобы просмотреть его в вашем браузере.

Страница перезагрузится, когда вы внесете изменения.\
Вы также можете увидеть любые ошибки lint в консоли.

`npm run build`

Создает приложение для производства в папке `build`.\
```

Он правильно связывает React в рабочем режиме и оптимизирует сборку для достижения наилучшей производительности.

Сборка сокращена, а имена файлов содержат хэши.\
Ваше приложение готово к развертыванию!

См. раздел о [развертывании](https://facebook.github.io/create-react-app3/docs/deployment) для получения дополнительной
информации.

Дополнительно

Для запуска клиента и корректной его работы требуется использовать переменные окружения:

```
VITE_API_URL=адрес сервера(http://dm-server:5000/) | '/server/'
VITE_HOST=хост клиента
GENERATE_SOURCEMAP=false (нужно для работы карты)
VITE_PORT=порт клиента
VITE_AUTH_API_URL = адрес сервера(http://dm-server:5000/) | '/auth/'
VITE_SMART_CITY_API_WS=ws://localhost/websocket/
VITE_WS_STREAMING_URL=адрес веб-сокет сервера для контроля потоков с камер(ws://server:9000) | '/ws'
VITE_API_MEDIAMTX=http://mediamtx:8888 адрес для mediamtx
VITE_SCANDOR_API_URL= http://localhost/server/ для загрузки изображений со скандора
VITE_WS_TRAFFIC_URL = /traffic-ws/ сокеты для плавного передвижения общественного транспорта (ws://server:9002)
VITE_WS_DETECTORS_URL = "/detectors-ws/" детекторы транспортного потока по сокетам (ws://server:9001)
VITE_SNOW_SEEKER_URL = http://localhost/server/ для загрузки изображений с сикера
VITE_WS_SNOW_SEEKER_STREAMING_URL = /snow-seeker-ws/ адрес веб-сокета для контроля потоков с камер зимнего содержания
```

Если не указывать VITE_API_URL и VITE_API_WS, то будут относительные адреса /server/ и /websocket/