ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ «АВТОТРЕК»

(ПО «Автотрек»)

Руководство системного программиста

Листов 31

Москва, 2025

Содержание

1 Общие сведения о Программе 3
1.1 Наименование и обозначение программы 3
1.2 Цель создания 3
1.3 Назначение программы 3
1.4 Технические и программные средства, обеспечивающие функционирование ПО «Автотрек» 4
1.4.1 Программное обеспечение ПО «Автотрек» 4
1.4.2 Комплекс технических средств ПО «Автотрек» 5
1.5 Уровень квалификации администратора программы 6
1.6 Перечень эксплуатационной документации, с которым необходимо ознакомиться системному программисту 7
2 Структура программы 8
2.1 Описание архитектуры ПО «Автотрек» 8
2.2 Компоненты программы 8
2.2.1 Компонент Администрирование 9
2.2.2 Компонент ведения справочников 9
2.2.3 Компонент ведения маршрутов 9
2.2.4 Компонент мониторинга и контроля 9
2.2.5 Компонент ведения обслуживания и ремонтов 10
2.2.6 Компонент управления вспомогательной техникой 10
2.2.7 Компонент планирования работы персонала 10
2.2.8 Компонент сопровождения выхода на линию 10
3 Установка Программы 12
3.1 Характеристики серверного оборудования ПО «Автотрек» 12
3.2 Сборка ПО «Автотрек» 12
3.2.1 Компоненты программы 13
3.3 Команды для администрирования программы 24
4 Запуск и завершение работы программы 26
5 Проверка программы 27
6 Настройка программы 28
7 Сообщения системному программисту 29
8 Действия при переходе ПО «Автотрек» в аварийный режим функционирования 30
Перечень сокращений 31

1 Общие сведения о Программе

1.1 Наименование и обозначение программы

Полное наименование программы: Программное обеспечение «Автотрек».

Сокращенное наименование программы: ПО «Автотрек» (далее — ПО, Программа).

1.2 Цель создания

Цель создания ПО «Автотрек» — автоматизация бизнес-процессов управления подвижным составом и планирования рабочего времени водителей для организаций с парком от 50 машин.

1.3 Назначение программы

Программа предназначена для автоматизации бизнес-процессов:

- мониторинга парка ТС компании в режиме реального времени, а также для просмотра исторических данных местоположений конкретного транспортного средства и сотрудника, основных характеристик, замеряемых бортовыми комплексами оборудования (для ТС) и абонентскими телематическими терминалами;
- формирования маршрутов следования ТС, а также мониторинга факта следования по маршруту;
- диспетчеризации работы подвижного состава компании с ведением путевых листов и план-графиков работы водителей;
- ведения в программе информации о проводимых ремонтах ТС;
- общения диспетчеров в чате;
- администрирования работы сотрудников функционального подразделения компании;
- контроля расхода топлива.

1.4 Технические и программные средства, обеспечивающие функционирование ПО «Автотрек»

1.4.1 Программное обеспечение ПО «Автотрек»

Программное обеспечение серверной части ПО «Автотрек» реализовано с использованием следующего системного и специального ПО:

- операционная система — Ubuntu, Debian GNU/Linux;
- сервер СУБД — PostgreSQL;
- Apache Tomcat;
- ClickHouse;
- Grafana;
- Docker;
- Docker-compose;
- Elasticsearch;
- Gitlab runner;
- Kibana;
- Kubernetes;
- Lighttpd;
- MapServer;
- MinIO;
- MongoDB;
- Monit;
- OSRM;
- Pgbouncer;
- Prometeus;
- RabbitMQ;
- Redis;
- RMQ Gate;
- Ruby.

1.4.2 Комплекс технических средств ПО «Автотрек»

В состав комплекса технических средств ПО «Автотрек» входит:

– серверное оборудование, необходимое для функционирования программы.

1.4.2.1 Серверная часть ПО «Автотрек»

Минимальные и рекомендуемые значения показателей к техническим характеристикам серверного оборудования представлены в таблице 1.

Таблица 1 — Требования к техническим характеристикам серверного оборудования

Показатель	Минимальное значение	Рекомендуемое значение
Процессор		
Количество ядер	20 шт.	40 шт. (из них 40 физических)
Тактовая частота	2600 МГц	3600 МГц
Оперативная ECC-память		
Объем	64 Гб	128 Гб
Частота	2400 МГц	2400 МГц
Дисковая подсистема		
SSD	2 Тб	4 Тб
HDD	512 Гб	1 Тб
Сетевой интерфейс		
Скорость подключения	1 Гбит/с	1 Гбит/с
Количество	2 шт.	2 шт.

1.4.2.2 Аппаратное обеспечение стационарных рабочих мест ПО «Автотрек»

Конфигурация устройств, используемых в качестве стационарных рабочих мест для пользователей ПО «Автотрек», должна соответствовать требованиям, представленным в таблице 2.

Таблица 2 — Конфигурация ПК

Показатель	Минимальное значение	Рекомендуемое значение
Процессор		
Количество ядер	4 (не менее двух физических) шт.	4 шт. физических
Тактовая частота	2400 МГц	3200 МГц
Оперативная память		
Объем	8 Гб	16 Гб
Частота	1333 МГц	2400 МГц
Сетевой интерфейс		
Скорость подключения	100 Мбит/с	1 Гбит/с
Экран		
Диагональ	15 дюймов	19 дюймов
Разрешение по горизонтали	1280 пикселей	1440 пикселей
Форм-фактор	4:3	16:9

1.5 Уровень квалификации администратора программы

Системное администрирование ПО «Автотрек» включает в себя следующие функции:

– обеспечение управления настройками;
– обеспечение управления правами доступа к функциям и информации;
– журналирование действий пользователей и автоматических операций.

Для выполнения своих обязанностей системный программист должен обладать следующими знаниями и умениями:

- обладать обширными знаниями предметной области;
- обладать высоким уровнем квалификации и практическим опытом выполнения работ по установке, настройке и администрированию программных средств, применяемых в ПО «Автотрек»;
- иметь профессиональные знания и практический опыт в области системного администрирования;
- обладать высоким уровнем квалификации и практическим опытом выполнения работ по установке, настройке и администрированию СУБД, используемых в ПО «Автотрек».
- обладать общими знаниями серверных операционных систем;
- обладать знаниями в области информационной безопасности.

1.6 Перечень эксплуатационной документации, с которым необходимо ознакомиться системному программисту

Перед началом работы системному программисту следует ознакомиться с данным руководством.

2 Структура программы

2.1 Описание архитектуры ПО «Автотрек»

ПО «Автотрек» реализована с использованием трехуровневой архитектуры, и состоит из следующих функциональных компонентов:

– уровень клиентских АРМ пользователей (тонкий клиент/блок «Пользовательский интерфейс»);
– уровень сервера приложений (сервер аналитической обработки/блок «Бизнес-логика»);
– уровень базы данных (сервер БД/блок «Управление данными»).

Доступ пользователей к функциям программы реализован с использованием технологии «тонкого клиента» (через web-браузер).

С точки зрения программного обеспечения роль клиента выполняет любой веб-браузер. Для запуска программы достаточно набрать в адресной строке браузера адрес сервера, после чего будет установлена связь с сервером приложений и загружено окно входа в программу.

2.2 Компоненты программы

В состав ПО «Автотрек» входят следующие программные компоненты:

– компонент Администрирование;
– компонент ведения справочников;
– компонент ведения маршрутов;
– компонент мониторинга и контроля;
– компонент ведения обслуживания и ремонтов;
– компонент управления вспомогательной техникой;
– компонент планирования работы персонала;
– компонент сопровождения выхода на линию.

А также функциональность оповещений и обмена сообщениями между пользователями программного обеспечения.

Подробно функции программных компонентов и их выполнения описаны в документе «Руководство оператора».

2.2.1 Компонент Администрирование

Компонент Администрирования предназначен для управления настройками Программы и правами доступа к функциям и ее данным.

Компонент обеспечивает:

- управление настройками Программы;
- управление правами доступа к функциям и данным в Программе;
- журналирование действий пользователей и автоматических операций в Программе.

2.2.2 Компонент ведения справочников

Компонент ведения справочников предназначен для автоматизации процесса планирования работы и эксплуатации ТС и обеспечивает формирование и ведение соответствующих реестров и справочников.

2.2.3 Компонент ведения маршрутов

Компонент ведения маршрутов предназначен для автоматизации процесса планирования работы и эксплуатации ТС и обеспечивает формирование маршрутов движения (в зависимости от типа объекта).

2.2.4 Компонент мониторинга и контроля

Компонент мониторинга и контроля предназначен для предоставления пользователям доступа к функциям мониторинга и контроля работы.

Компонент обеспечивает:

- картографическое обеспечение Программы;
- назначение ТС, водителей и сотрудников;

– визуализацию пространственных данных (в т. ч. текущего местонахождения, истории передвижения, местонахождения различных гео-объектов и мест) и сопутствующую справочную информацию;
– оперативное управления (изменений заданий (маршрутов), статусов, отправка сообщений);
– контроль соблюдения правил эксплуатации и технического состояния ТС;
– учет и контроль параметров работы ТС и персонала, контроль выполнения заданий.

2.2.5 Компонент ведения обслуживания и ремонтов

Компонент ведения обслуживания и ремонтов предназначен для автоматизации процесса ведения ремонтных работ с ТС и обеспечивает формирование печатных форм для отправки ТС на ремонт.

2.2.6 Компонент управления вспомогательной техникой

Компонент управления вспомогательной техникой предназначен для автоматизации процесса планирования работы вспомогательной техники и обеспечивает формирование печатных форм рапортов для отчетности о проведенной работе.

2.2.7 Компонент планирования работы персонала

Компонент планирования работы персонала предназначен для автоматизации процесса планирования работы и эксплуатации ТС и обеспечивает формирование заданий персоналу.

2.2.8 Компонент сопровождения выхода на линию

Компонент сопровождения выхода на линию предназначен для повышения эффективности работы водителей за счет своевременного контроля и анализа их

действий на маршруте, а также сокращения временных затрат на диспетчерское управление.

Компонент обеспечивает:

– возможность мониторинга, контроля и расчета эффективности работы водителей ТС;
– автоматический расчет статуса водителя в режиме реального времени;
– контроль над работой диспетчеров;
– создание служебных записок для фиксации нарушений водителей.

3 Установка Программы

3.1 Характеристики серверного оборудования ПО «Автотрек»

ПО «Автотрек» устанавливается разработчиком.

Системному программисту обеспечивается доступ на серверы и предусмотрена возможность самостоятельного подключения удаленных сервисов.

Системному программисту необходимо контролировать работоспособность портов, которые первоначально были установлены разработчиком для обеспечения их доступности и осуществления своевременной поддержки.

3.2 Сборка ПО «Автотрек»

Сборку ПО серверных компонентов рекомендуется выполнять с использованием Docker. Исходный код серверных компонентов содержит Dockerfile, необходимый для создания образов.

Приложения сторонних разработчиков, необходимые для работы Программы:

- PostgreSQL версии 15 и выше с установленными расширениями:
  - PostGIS версии 3.3.2 и выше;
  - uuid-ossp версии 1.1 и выше;
  - PG Partman версии 4.7.2 и выше (рекомендовано, но необязательно);
- S3 хранилище, например, Minio;
- RabbitMQ версии 4 и выше;
- Redis версии 7.0 и выше;
- сервер Rocket Chat версии 7.0 и выше:
  - Rocket Chat требует нереляционную базу данных MongoDB версии 7.0 и выше.
- OSRM сервер с загруженным графом для Московской области;
- Front-web сервер: рекомендуется Nginx версии 1.27 и выше.

3.2.1 Компоненты программы

3.2.1.1 Frontend

Frontend — это клиентская часть для работы через web-browser.

После сборки проекта необходимо настроить Front-web-server для доступа к получившимся статическим файлам. Пример конфигурации содержится в самом проекте в файле `docker/nginx.conf`. Также в проекте есть Dockerfile, создающий контейнер с легковесным web-сервером, позволяющим обеспечить доступность статичных файлов (шаблонов страниц, JS-скриптов, изображений, CSS-стилей и прочего) по HTTP. Данная конфигурация направлена на простоту и скорость взаимодействия компонентов программы. Связанные с безопасностью настройки (такие, как SSL-шифрование) рекомендуется выполнять на Front-web сервере.

Front-web должен проксировать запросы от пользователей на статику данного компонента.

Пример конфигурации Nginx в качестве Front-web сервера:

```
upstream web {
  server 127.0.0.1:8080; # адрес контейнера со статикой
}

  location ~ ^/next(/.*|$) {
    proxy_pass http://web/codd$1;
  }
```

Для корректной работы Frontend необходимо, чтобы Front-web сервер проксировал запросы к статике через алиас next.

3.2.1.2 User Management

User Management сервис, обеспечивающих сквозную авторизацию и аутентификацию пользователей.

Для корректной работы приложения требуется .NET Runtime версии 8.x. Для сборки исполняемого проекта из исходного кода — .NET SDK версии 8.x. По результатам сборки проекта создаётся исполняемый бинарный файл.

В состав исходного кода также входит Dockerfile для создания образа с сервером приложения.

Для конфигурации сервиса необходимо определить следующие параметры среды:

- `ConnectionStrings__ApplicationContext` — параметры для доступа к Postgresql-серверу;
- `ConnectionStrings__Redis` — параметры для доступа к Redis;
- `Kestrel__Endpoints__gRPC__Url` — адрес для обслуживания запросов gRPC;
- `Kestrel__Endpoints__Http__Url` — адрес для обслуживания HTTP запросов;
- `TokenManager__PrivateKey` — секретная случайная строка для обеспечения шифрования временной стойкости для токенов доступа пользователей (ключи периодически меняются, гарантированная стойкость не требуется).

Пример конфигурации:

```
ASPNETCORE_ENVIRONMENT=Production
ConnectionStrings__ApplicationContext=Host=127.0.0.1;Port=5432;Database=mps_
production;Username=user;Password=password;Timeout=5;CommandTimeout=10
ConnectionStrings__Redis=127.0.0.1:6379
Kestrel__Endpoints__gRPC__Url=http://0.0.0.0:50051
Kestrel__Endpoints__Http__Url=http://0.0.0.0:9200

TokenManager__PrivateKey=GWYy3ddNjLOVowuNR0feKvtJTr8VvKNf2dyjwU9gCdWKx5SirRI
1XfHGKwR0L7NUTCVNN2gghlbktzPsIFkhKvAkg52RzAqgLrofB9Cn7nldWSerFyvRzZNxz0LsbEO
v
```

Для обслуживания запросов пользователей рекомендуется проксировать запросы средствами Front-web сервера. Для взаимодействия между серверными

компонентами программы (кроме Frontend) рекомендуется использовать прямое подключение.

Пример конфигурации Nginx в качестве Front-web сервера:

```
upstream user_management {
  server 127.0.0.1:9200; # адрес контейнера с сервером приложения
}

  location ~ ^/user-management/(.+) {
    proxy_pass http://user_management/$1$is_args$args;
  }
```

Для корректной работы Frontend необходимо, чтобы Front-web сервер проксировал запросы к сервису через алиас user-management.

3.2.1.3 Telemetry receiver

Telemetry receiver — сервис приёма телеметрии.

Для корректной работы приложения требуется интерпретатор Ruby версии 3.4.1+ и Ruby gems. В состав исходного кода также входить Dockerfile, позволяющий создать контейнер с рабочим сервером сервиса приема телеметрии.

Для запуска проекта необходим выполнить команду:

```
bundle exec puma -p [Номер порта для приёма телеметрии]
```

Для конфигурации сервиса необходимо определить следующие параметры среды:

- `SETTINGS__RABBITMQ__EXCHANGE_NAME` — название обменника сообщениями для передачи полученных пакетов телеметрии;
- `SETTINGS__RABBITMQ__ROUTING_KEY` — ключ маршрутизации для передачи полученных пакетов телеметрии;

– SETTINGS__RABBITMQ__CURRENT__HOST — ip адрес сервера RabbitMQ;
– SETTINGS__RABBITMQ__CURRENT__USER — имя пользователя для обращений к RabbitMQ;
– SETTINGS__RABBITMQ__CURRENT__PASSWORD — пароль для обращений к RabbitMQ;
– SETTINGS__RABBITMQ__CURRENT__VHOST — название внутреннего виртуального хоста RabbitMQ.

Пример конфигурации:

```
SETTINGS__RABBITMQ__EXCHANGE_NAME=mps.core.telemetry
SETTINGS__RABBITMQ__ROUTING_KEY=telemetry.data
SETTINGS__RABBITMQ__CURRENT__HOST=127.0.0.1
SETTINGS__RABBITMQ__CURRENT__USER=user
SETTINGS__RABBITMQ__CURRENT__PASSWORD=password
SETTINGS__RABBITMQ__CURRENT__VHOST=mps.production
```

Для корректного взаимодействия сервисов важно чтобы название очереди соответствовало шаблону [prefix]core.telemetry где [prefix] соответствует параметру SETTINGS__RABBITMQ__CURRENT__PREFIX основного backend сервера (см. далее пп. 3.2.1.4 Core service) а параметр SETTINGS__RABBITMQ__ROUTING_KEY должен быть telemetry.data.

Для приема пакетов телеметрии необходимо проксировать запросы средствами Front-web сервера.

Пример конфигурации Nginx в качестве Front-web сервера:

```
upstream telemetry-receiver {
  server 127.0.0.1:8081; # адрес контейнера с приемщиком телеметрии
}

  location ~ ^/telemetry-receiver {
    proxy_pass http://telemetry-receiver;
  }
```

3.2.1.4 Core service

Core service — основной сервис бизнес-логики.

Для корректной работы приложения требуется интерпретатор Ruby версии 3.4.1+ и Ruby gems. В состав исходного кода также входить Dockerfile, позволяющий создать контейнер с рабочим сервером сервиса приема телеметрии.

Для запуска проекта необходим запускать три инстанса приложения:

- сервер приложения `bundle exec puma -p [Номер порта для обработки запросов пользователей]`;
- сервер выполнения отложенных и регулярных команд `bundle exec sidekiq`;
- сервер асинхронного выполнения команд и обработки внутренних событий `bundle exec rake sneakers:run`.

Все три инстанса используют одинаковые настройки.

Для конфигурации сервиса необходимо определить следующие параметры среды:

- `SETTINGS__SYSTEM__ENABLE_REDIS_CACHE_STORE` — если задан, то для кеширования результатов некоторых часто повторяемых запросов будет использовано in-memory хранилище.
- `SETTINGS__RAILS__SECRET_KEY_BASE` — секретный ключ для шифрования некоторых данных на стороне сервера.
- настройки взаимодействия с RabbitMQ. Количество предзагружаемых из очереди сообщений, количество разделяемых тредов, общее количество тредов и количество копий процесса:
  - `SETTINGS__SNEAKERS__PREFETCH`.
  - `SETTINGS__SNEAKERS__SHARE_THREADS`.
  - `SETTINGS__SNEAKERS__THREADS`.
  - `SETTINGS__SNEAKERS__WORKERS`.
- Параметры для доступа к FTP-серверу оператора электронных путевых листов:
  - `SETTINGS__EASSIGNMENT_FTP__HOST`.

- SETTINGS__EASSIGNMENT_FTP__USER.
- SETTINGS__EASSIGNMENT_FTP__PASSWORD.

- Параметры доступа к S3 сервису для хранения загружаемых файлов:
  - SETTINGS__MINIO__ENDPOINT.
  - SETTINGS__MINIO__SECRET_ACCESS_KEY.
  - SETTINGS__MINIO__ACCESS_KEY_ID.
  - SETTINGS__MINIO__BUCKET.
- Параметры URL сервиса, по которому доступен Сервис Управления Пользователями и секретный (системный) токен:
  - SETTINGS__USER_MANAGEMENT__URL.
  - SETTINGS__USER_MANAGEMENT__TOKEN.
- Параметры доступа к базе данных:
  - SETTINGS__POSTGRESQL__ADAPTER.
  - SETTINGS__POSTGRESQL__ENCODING.
  - SETTINGS__POSTGRESQL__HOST.
  - SETTINGS__POSTGRESQL__PORT.
  - SETTINGS__POSTGRESQL__USER.
  - SETTINGS__POSTGRESQL__PASSWORD.
  - SETTINGS__POSTGRESQL__DATABASE.
  - SETTINGS__POSTGRESQL__POOL.
  - SETTINGS__POSTGRESQL__TIMEOUT.
- Параметры доступа к серверу RabbitMQ (также см. описание сервиса приёма телеметрии в пп. 3.2.1.3 Telemetry receiver):
  - SETTINGS__RABBITMQ__CURRENT__HOST.
  - SETTINGS__RABBITMQ__CURRENT__USER.
  - SETTINGS__RABBITMQ__CURRENT__PASSWORD.
  - SETTINGS__RABBITMQ__CURRENT__VHOST.
  - SETTINGS__RABBITMQ__CURRENT__EXCHANGES__TELEMETRY.
- Параметры доступа к Redis:
  - SETTINGS__REDIS__HOST.

    - SETTINGS__REDIS__PORT.
    - SETTINGS__REDIS__DB.
- Параметры доступа к серверу Rocket.Chat для автоматического создания учётных записей пользователей и бесшовной аутентификации:
    - SETTINGS__ROCKETCHAT__HOST.
    - SETTINGS__ROCKETCHAT__PORT.
    - SETTINGS__ROCKETCHAT__USERNAME.
    - SETTINGS__ROCKETCHAT__PASSWORD.

Для обслуживания запросов пользователей рекомендуется проксировать запросы средствами Front-web сервера.

Пример конфигурации Nginx в качестве Front-web сервера:

```
upstream api {
  server 127.0.0.1:3000;
}

  # API
  location ~ ^/(api|rails|photos|esmo|print|directories|mayor|rnis|routing-
service|stream) {
    proxy_pass http://api;
  }

  # WebSocket
  location ~ ^/socket {
    proxy_pass_request_headers on;
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection 'upgrade';
    proxy_set_header Host $host;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forward-For $proxy_add_x_forwarded_for;
    proxy_set_header X-Forward-Proto http;
    proxy_set_header X-Nginx-Proxy true;

    proxy_pass http://api;
    proxy_http_version 1.1;
    proxy_read_timeout 180s;
  }
```

Для корректной работы Frontend необходимо, чтобы Front-web сервер проксировал запросы к сервису через алиасы:

- api;
- rails;
- photos;
- esmo;
- print;
- directories;
- routing-service;
- stream;
- socket.

Пример конфигурации развёрнутого приложения на базе Docker Compose:

```
x-service: &service
  init: true
  restart: always
  env_file: .env

x-postgres: &postgres
  image: gitlab/mps/postgres:15

x-core: &core
  <<: *service
  image: gitlab/mps/core:latest

services:
  postgresql15:
    <<: *postgres
    restart: always
    container_name: postgres
    shm_size: 1g
    environment:
      POSTGRES_USER: mps
      POSTGRES_PASSWORD: 123456789
      POSTGRES_DB: mps
    volumes:
      - ./volumes/postgresql15/data:/var/lib/postgresql/data
```

```
  expose:
    - 5432

rabbitmq:
  image: rabbitmq:4.0-management
  restart: always
  environment:
    RABBITMQ_DEFAULT_USER: mps
    RABBITMQ_DEFAULT_VHOST: mps.production
    RABBITMQ_DEFAULT_PASS: 123456789
  volumes:
    - ./volumes/rabbitmq/data:/var/lib/rabbitmq
  expose:
    - 4369
    - 5671
    - 5672
    - 15671 # Management Port
    - 15672 # Management Port
    - 15674 # Port of Stomp Plugin
    - 25672



minio:
  image: quay.io/minio/minio:RELEASE.2025-01-20T14-49-07Z
  command: server --address ":9000" --console-address ":9001" /data
  restart: always
  expose:
    - 9000
    - 9001
  volumes:
    - ./volumes/minio/data:/data
  environment:
    MINIO_ROOT_USER: mps
    MINIO_ROOT_PASSWORD: 123456789

redis:
  image: redis:7
  restart: always
  volumes:
    - ./volumes/redis/data:/data
  expose:
    - 6379
```

```
 nginx:
   image: nginx:1.27
   restart: always
   volumes:
     - ./volumes/nginx/nginx.conf:/etc/nginx/nginx.conf:ro
     - ./volumes/nginx/conf.d:/etc/nginx/conf.d:ro
   ports:
     - 80:80

web:
   image: gitlab/mps/web:latest
   restart: always
   expose:
     - 80

 user-management:
   <<: *service
   image: gitlab/mps/madi-operational-management/usermanagement:candidate
   expose:
     - 9200
     - 50051

 core:
   <<: *core
   command: bundle exec puma

 core-sneakers:
   <<: *core
   command: bundle exec rake sneakers:run

 core-cron:
   <<: *core
   command: bundle exec sidekiq

 parkon-telemetry-receiver:
   <<: *service
   image: gitlab/mps/telemetry-receiver:latest
   command: bundle exec puma -p 80
   expose:
  - 80
```

И соответствующая конфигурация Nginx:

```
upstream web {
  server web:80;
}

upstream user_management {
  server user-management:9200;
}

upstream telemetry-receiver {
  server parkon-telemetry-receiver:80;
}

upstream api {
  server core:3000;
}

server {
  listen ssl default_server;
  server_name 53.64.98.213;
  large_client_header_buffers 100 5120k;
  client_max_body_size 100M;

  # Default proxy settings
  # ------------------------------------------------------------------------
--

  proxy_set_header Host $http_host;
  proxy_set_header X-Real-IP $remote_addr;
  proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
  proxy_set_header X-Forwarded-Proto $scheme;
  proxy_redirect off;

  location ~ ^/next(/.*|$) {
    proxy_pass http://web/codd$1;
  }

  location ~ ^/user-management/(.+) {
    proxy_pass http://user_management/$1$is_args$args;
  }

  location ~ ^/api/v2/telemetry/data {
```

```
    if ($request_method = POST) {
      proxy_pass http://telemetry-receiver;
    }

    if ($request_method = GET) {
      proxy_pass http://api;
    }
  }

  # API
  location ~ ^/(api|rails|photos|esmo|print|directories|mayor|rnis|routing-
service|stream) {
    proxy_pass http://api;
  }

  # WebSocket
  location ~ ^/socket {
    proxy_pass_request_headers on;
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection 'upgrade';
    proxy_set_header Host $host;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forward-For $proxy_add_x_forwarded_for;
    proxy_set_header X-Forward-Proto http;
    proxy_set_header X-Nginx-Proxy true;

    proxy_pass http://api;
    proxy_http_version 1.1;
    proxy_read_timeout 180s;
  }

}
```

3.3 Команды для администрирования программы

По умолчанию, большинство команд выполняется с использованием настроек сред development, предназначенной для разработки. Чтобы выполнять команды в рабочей среде требуется установить переменную окружения `RAILS_ENV=production`.

Список всех возможных команд с описанием можно получить, использовав команду `rake -T`.

При администрировании полезными могут оказаться следующие команды:

– rake db:migrate:status отображает состояние миграций БД;
– rake db:rollback откатывает последнее изменение схемы БД;
– rake db:migrate применить ожидающие миграции БД.

Для поиска неисправностей может быть полезно интерактивное взаимодействие с приложением. Для этого предусмотрена команда rails console которая запускает интерактивную консоль с загруженной средой.

Все эти команды должны быть выполнены в корне проекта.

Текущее развёртывание выполнено при помощи утилиты Docker Compose, в связи с этим, выполнять команды нужно внутри соответствующего контейнера:

```
sudo docker-compose exec НАЗВАНИЕ_СЕРВИСА КОМАНДА
```

например: `sudo docker-compose exec core_cron rake db:migrate:status`

Также иногда удобнее просматривать сырые логи в приложении. Для этого подходит команда:

```
sudo docker-compose logs [—
tail=КОЛИЧЕСТВО_СТРОК_ЛОГОВ_НАЧИНАЯ_С_САМЫХ_НОВЫХ] [-f] НАЗВАНИЕ_СЕРВИСА
```

4 Запуск и завершение работы программы

Запуск ПО «Автотрек» осуществляется с помощью интернет-браузера по адресу https://autotrek.traklight.ru:9443/next.

Для завершения Программы необходимо нажать на кнопку 

в верхнем правом углу или закрыть страницу интернет-браузера.

5 Проверка программы

Специальные проверки методом прогона Программы для определения ее работоспособности не предусмотрены.

Общее заключение о работоспособности Программы может быть сформировано в процессе работы с её компонентами.

6 Настройка программы

ПО «Автотрек» не требует настройки со стороны системного программиста.

Для работы пользователей не требуется дополнительных настроек.

Работа пользователя с программой осуществляется через веб-браузер. Для входа в программу пользователю необходимо только предоставить IP-адрес, и ввести логин и пароль в соответствующие поля (устанавливается для каждого пользователя администратором самостоятельно).

Администрирование пользователей на уровне управления доступом осуществляется через веб-интерфейс и выполняется пользователем с ролью администратор.

7 Сообщения системному программисту

В процессе работы Программы выдача сообщений системному программисту не предусмотрена.

8 Действия при переходе ПО «Автотрек» в аварийный режим функционирования

Причинами нарушения непрерывного режима функционирования Программы и перехода из штатного в аварийный режим функционирования могут являться:

– отключение электроэнергии;
– недоступность каналов передачи данных (авария).

Действия администратора в аварийном режиме включают:

– диагностирование инцидентов или проблем, связанных со сбоями или нештатными ситуациями в работе ПО «Автотрек»;
– восстановление при необходимости программно-аппаратной конфигурации ПО «Автотрек» (сетевого и серверного оборудования);
– восстановление информации при ее утере средствами резервного копирования и восстановления;
– расследование причин нештатной ситуации и определение причин инцидента или проблемы.

Реагирование на аварийные ситуации включает оповещение обслуживающего персонала, принятие контрмер, необходимое восстановление информации, выработку и проведение профилактических мероприятий.

После проведения первичной диагностики и определения причин нештатной ситуаций нештатный режим переходит в сервисный режим.

При переходе ПО «Автотрек» в сервисный режим время отклика на пользовательские запросы увеличивается и может наблюдаться небольшое снижение общей производительности.

Перечень сокращений

Сокращение	Расшифровка
АРМ	Автоматизированное рабочее место
БД	База данных
ПО	Программное обеспечение
СУБД	Система управления базами данных
ТС	Транспортное средство