ОБЩЕСТВО С ОГРАНИЧЕННОЙ ОТВЕТСТВЕННОСТЬЮ
«РТ КИС»
123290, Российская Федерация, г. Москва, ул. 2-я Магистральная, д. 8А, стр. 2
ОГРН 1207700233806, ИНН 7714461666, КПП 771401001

УТВЕРЖДАЮ
Генеральный директор
ООО «РТ КИС»

_____ С.В. Пчелинцева
«____» _____ 20__г.

ЕДИНАЯ ЦИФРОВАЯ ПЛАТФОРМА СКОРОЙ МЕДИЦИНСКОЙ ПОМОЩИ

Шифр: ЕЦП СМП

Инструкция по установке экземпляра программного обеспечения

РНПЦ.466451.001И2

Листов 37

Москва, 2024 г.

АННОТАЦИЯ

В документе приведено описание действий по установке и настройке ПО (требования к системе, установка и т.д.), указания по разворачиванию экземпляра на сервере.

Оформление и содержание документа выполнено в соответствии с документами «Методические рекомендации по работе с Федеральной государственной информационной системой «Реестры программ для электронных вычислительных машин и баз данных» (ФГИС Реестры ПО).

СОДЕРЖАНИЕ

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ

В настоящем документе применяют следующие термины и сокращения, которые представлены в таблице 1.

Таблица 1 — Перечень терминов и сокращений

Термин/ Сокращение	**Расшифровка**
ЕЦП СМП	Единая цифровая платформа скорой медицинской помощи
ИС	Информационная система
ОС	Операционная система
ПО	Программное обеспечение
ПТК	Программно-технический комплекс
ПЭВМ	Персональная электронно-вычислительная машина
СМП	Скорая медицинская помощь
ЭВМ	Электронно-вычислительная машина

1 ВВЕДЕНИЕ

1.1 Назначение программы

ЕЦП СМП предназначена для автоматизации процессов сбора, обработки и хранения информации при осуществлении приема вызовов, мониторинга и управления выездными бригадами станций (отделений) скорой и неотложной медицинской помощи, диспетчерского управления транспортными средствами служб скорой и неотложной медицинской помощи медицинских государственных и коммерческих медицинских организаций субъектов РФ.

1.2 Функции программы

ЕЦП СМП состоит из функциональных модулей, логически соответствующих автоматизируемым направлениям деятельности СМП, а также характеру выполняемых задач. Состав и назначение функциональных модулей представлены в таблице 2. Функциональные модули реализованы в составе серверных компонентов ЕЦП СМП.

Таблице 2 — Функциональные модули ЕЦП СМП и их функции

№ п/п	Функциональный (сервисный) модуль	Основные функции
1	Прием вызова	– Прием и регистрация вызова; – Ведение реестра вызовов; – Мониторинг нормативов обслуживания вызова
2	Координация вызовов	– Управление распределением вызовов на бригады СМП; – Отображение информации о вызовах и бригадах
3	Ведение документации по вызову	– Создание электронной карты вызова (обогащение первичной информации о вызове сведениями о пациенте, результате выезда и оказанной помощи); – Ведение реестра электронных карт вызова

№ п/п	Функциональный (сервисный) модуль	Основные функции
4	Управление бригадами	– Создание карточки бригады СПМ; – Ведение реестра карточек бригад СМП; – Планирование бригад СМП; – Управление составом и статусом бригады СМП
5	Кадровый учет	– Создание карточки сотрудника; – Ведение реестра сотрудников; – Ведение планового расписания сотрудника; – Учет выданного сотруднику оборудования
6	Учет транспортных услуг	– Создание карточки транспортного средства; – Ведение реестра транспортных средств; – Управление данными о комплектации транспорта оборудованием; – Управление статусами транспортного средства
7	Учет оборудования	– Создание карточки оборудования; – Ведение реестра оборудования; – Управление статусами оборудования
8	Оперативный мониторинг	– Отображение на карте в реальном времени информации о местах дислокации медицинских организациях, местах вызова, бригадах СМП; – Отображение в табличном виде информации о обслуживаемых вызовах, статусах бригад, отклонениях от нормативного времени обслуживания вызова
9	Отчетность	– Формирование регламентированных отчетов

№ п/п	Функциональный (сервисный) модуль	Основные функции
10	Реестр услуг застрахованных лиц	– Создание карточки застрахованного лица; – Ведение реестра услуг
11	Формирование расчетной информации	– Формирование отчетной информации об оказанных пациенту услугах
12	Картотека пациентов	– Предоставление карточки застрахованного по программе дополнительного медицинского страхования; – Ведение реестра застрахованных лиц
13	Администрирование	– Ведение справочников и классификаторов; – Настройка ЕЦП СМП
14	Авторизация и аутентификация	– Проверка пользователей и доступных им прав; – Контроль прав доступа пользователя к объекту
15	Картография	– Сервисный модуль, обеспечивающий: – Отображение электронной карты, подвижных и стационарных, объектов, геозон; – Прямое и обратное геокодирование; – Построение маршрутов транспорта

1.3 Сведения о технических и программных средствах, обеспечивающих выполнение программы

Стабильная работа программы обеспечивается на серверном программно- техническом комплексе (ПТК).

Тип оборудования серверного ПТК, его количественные и качественные характеристики выбираются исходя из количества подключенных ИС и ПО. Минимальный ПТК должен обладать следующими характеристиками, представленные в таблице 3

Таблица 3 — Технические характеристики ПТК

№	Тип оборудования	Краткие характеристики сервера	Количество, единиц
1	Сервер приложений	CPU – 12 ядер RAM – 24 ГБ HDD – 500 ГБ	1
2	Сервер баз данных	CPU – 24 ядер RAM – 32 ГБ HDD – 500 ГБ	2
3	Сервер картографии	CPU – 12 ядер RAM – 12 ГБ HDD – 1 ТБ	1

Программа функционирует на серверном оборудовании под управлением операционной системы Ubuntu Server. Ubuntu Server — свободно распространяемая по лицензии GNU GPL ОС для управления web-серверами, основанная на дистрибутиве Debian GNU\Linux.

Программное обеспечение так же возможно установить на операционные системы:

- Astra Linux;
- Альт Линукс;
- Ред ОС.

Также должно быть предусмотрено необходимое количество АРМ, предназначенных для автоматизации деятельности должностных лиц с характеристиками не хуже следующих:

– ПЭВМ (с процессором Intel Core 2 Duo с тактовой частотой не менее 3,5 ГГц, не менее 4 Гбайт оперативной памяти, типом памяти DDR4 SDRAM, жестким диском объемом не менее 500 Гб, монитором с диагональю не менее, 21”, разрешением не менее 1920×1080, сетевым адаптером для подключения к ЛВС 10/100/1000 Мбит/с);

– многофункциональное устройство печати;

– программное обеспечение общесистемное и прикладное:

- ОС: Windows 10 x64 и выше/Linux;
- web-браузер Google Chrome (версия 95)/Mozilla Firefox (версия 90);
- пакет офисных программ MS Office или OpenOffice.

Для расширения возможностей ОС на технические средства устанавливаются программные средства, перечисленные в таблице 4

Таблица 4 — Программные средства

№ п/п	ПО, расширяющее возможности ОС	Назначение	Лицензия
1	curl	Кроссплатформенная служебная программа командной строки, позволяющая взаимодействовать с множеством различных серверов по множеству различных протоколов с синтаксисом URL https://curl.se/	Свободное программное обеспечение Лицензия: MIT/X derivate license
2	nginx	Веб-сервер и почтовый прокси-сервер, работающий на Unix-подобных операционных системах https://nginx.org/	Свободное программное обеспечение Лицензия: 2-пунктная лицензия BSD

№ п/п	ПО, расширяющее возможности ОС	Назначение	Лицензия
3	openssl	Криптографический пакет с открытым исходным кодом OpenSSL для работы с SSL/TLS https://www.openssl.org	Свободное программное обеспечение Лицензия: Apache License 2.0
4	php	Язык программирования применяется для создания динамических web-сайтов https://php.net/	Свободное программное обеспечение Лицензия: PHP License
5	JasperReports	Java-библиотека для создания отчётов. http://community.jaspersoft.com /project/jasp erreports-library	Свободное программное обеспечение Лицензия: GNU Lesser General Public License
6	postgresql	Свободная объектно-реляционная система управления базами данных (СУБД) https://www.postgresql.org/	Свободное программное обеспечение Лицензия: PostgreSQL License (free and open-source)

№ п/п	ПО, расширяющее возможности ОС	Назначение	Лицензия
7	gzip	Утилита сжатия и восстановления (декомпрессии) файлов, использующая алгоритм Deflate. Применяется в основном в UNIX-системах, в ряде которых является стандартом де-факто для сжатия данных. http://www.gnu.org/software/gzip/	Свободное программное обеспечение Лицензия: GNU GPL
8	pg_dump	Утилита резервного копирования из состава postgresql	Свободное программное обеспечение Лицензия: PostgreSQL License (free and open-source)

2 СТРУКТУРА ПРОГРАММЫ

В разделе «Структура программы» приведены сведения о структуре программы, ее составных частях, о связях между составными частями и о связях с другими программами.

2.1 Сведения о структуре программы

Программа ЕЦП СМП строится по принципам сервисно-ориентированной архитектуры, которые предполагают создание прикладного приложения из сервисов, поддерживающих независимое развертывание. Как правило, каждый сервис ограничен только своей выполняемой функцией и взаимодействует с другими сервисами посредством программных интерфейсов.

Программа ЕЦП СМП состоит из следующих сервисов, представленных в таблице 5.

Таблица 5 —Сервисы ЕЦП СМП и их функции

№ п/п	Сервис	Назначение сервиса
1	Frontend	Пользовательский интерфейс использования системы, источник запросов пользователей системы
2	Login-Data-Service	Сервис аккумулирует в себе процессы аутентификации, авторизации и управляет пользовательскими настройками системы. Является proxy-сервисом для всех входящих пользовательских запросов. Сопровождается отдельной базой данных
3	Data Warehouse	Хранилище больших объемов данных в специальном формате, отчетность. Сопровождается отдельной базой данных
4	Бизнес-ядро	Набор модулей, обеспечивающих работу СМП

Компонентная схема программы ЕЦП СМП приведена на рисунке 1.

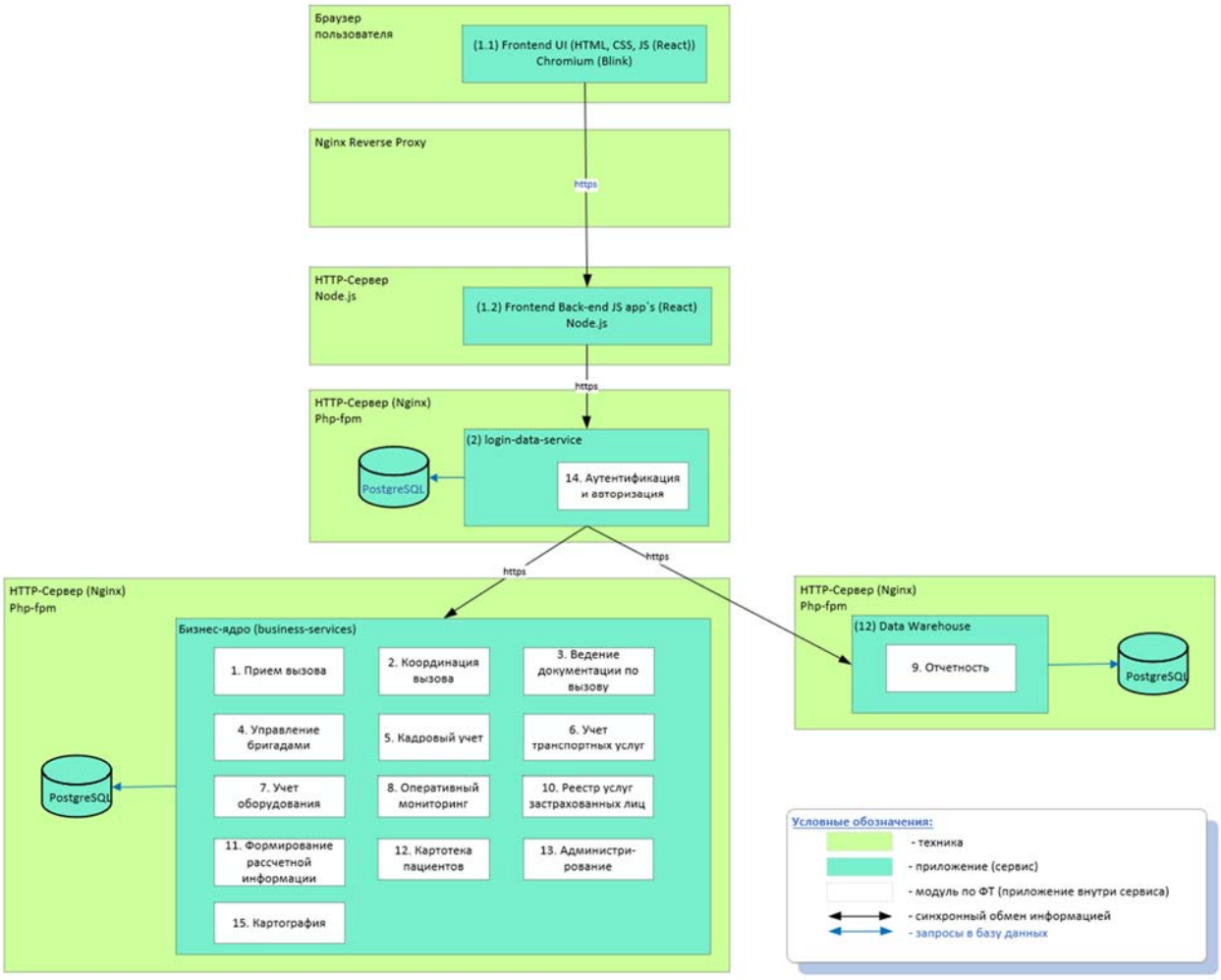


Рисунок 1 — Компонентная схема программы ЕЦП СМП

2.2 Сведения о связях между составными частями программы

Для обеспечения внутренней информационной безопасности в сети реализован метод разделения вычислительной сети на отдельные сегменты (зоны) в соответствии с уровнем доверия к работающим в каждом сегменте пользователям и сетевым сервисам. Каждая зона характеризуется наличием специфичных для нее сервисов, а также определенными правилами обмена сетевым трафиком с другими зонами. В вычислительной сети выделяются следующие зоны (перечислены в порядке возрастания доверия):

- Internet — внешняя незащищенная сеть. Из нее приходят запросы в приложение;
- DMZ (демилитаризованная зона) — предназначена для размещения клиентской части приложения, к которому настроен доступ через Интернет;

– Inside (внутренняя сеть) — наиболее защищенный сегмент сети. Предназначен для размещения серверной части приложения.

Размещение базовых программных компонентов осуществляется в сегментах Inside и DMZ.

Запрос приходит от пользователя в сервис frontend из Интернета через систему балансировки нагрузки. Затем сервис frontend передает запрос через прокси клиентской части в сервис аутентификации и авторизации, откуда маршрутизируется на остальные сервисы.

3 УСТАНОВКА И НАСТРОЙКА ПРОГРАММЫ

3.1 Описание состава и содержания исходного кода

Описание состава и содержания исходного кода ЕЦП СМП поставляется на машинном носителе информации.

3.2 Общие сведения по установке и настройке программы

Для обеспечения требуемого уровня сопровождаемости Системы разработка программного обеспечения ведется в соответствии с практиками «непрерывная интеграция/непрерывное развертывание» (CI/CD, Continuous Integration/Continuous Delivery).

Разработчики ведут создание ЕЦП СМП с использованием инструмента GitLab, который имеет встроенную систему контроля версий и позволяет выполнять совместную разработку силами нескольких команд, применять обновления кода, выполнять сборку и при необходимости откатывать изменения. Сервер GitLab развернут на собственном оборудовании исполнителя работ.

Сборка релизов происходит также на стороне разработчика в среде непрерывной интеграции и доставки GitLab CI/CD в соответствии со скриптом (сценарием) сборки приложения. Также на технических средствах разработчика развернута тестовая среда (так называемый, контур разработки и тестирования), в которой происходит развертывание разработанных компонентов программного обеспечения их автономное и совместное тестирование.

Для доставки приложения в промышленную среду разработчики в заархивированном виде передают их заказчику для развертывания и настройки. В качестве опытной среды используется контур тестирования и разработки. Предполагается развертывание ПО в составе общей сборки с компонентами прикладной инфраструктуры и технологическими компонентами.

3.3 Предварительная настройка

Перед началом работ по системному администрированию должно быть настроено удаленное администрирование серверов средствами ОС.

3.4 Запуск смежных систем

Перед началом работ по системному администрированию серверов запуск смежных систем не требуется.

3.5 Подготовка к развертыванию и установка необходимых программных пакетов

Для установки и настройки СПО ЕЦП СМП на сервере пользователю с административными правами необходимо выполнить следующие действия:

1) авторизоваться на сервере — в данном руководстве используется имя пользователя “CIS” и IP-адрес сервера “172.16.0.17”:

```
$ ssh cis@172.16.0.17
```

Для всех дальнейших команд в инструкции, необходимо находиться или произвести вход в режим суперпользователя root:

```
sudo -s
```

2) выполнить добавление источников необходимых программных пакетов и их установку в системе:

```
apt install dirmngr ca-certificates software-properties-common
apt-transport-https lsb-release curl -y
```

```
add-apt-repository ppa:ondrej/php
```

```
curl -fSsL https://www.postgresql.org/media/keys/ACCC4CF8.asc | gpg --dearmor |
sudo tee /usr/share/keyrings/postgresql.gpg > /dev/null
```

```
echo deb [arch=amd64,arm64,ppc64el signed-
by=/usr/share/keyrings/postgresql.gpg] http://apt.postgresql.org/pub/repos/apt/
$(lsb_release -cs)-pgdg main | sudo tee /etc/apt/sources.list.d/postgresql.list
```

```
apt update
```

```
apt install nginx openssl openvpn php8.1 php8.1-curl php8.1-fpm php8.1-intl
php8.1-mbstring php8.1-mysql php8.1-pgsql php8.1-xml php8.1-zip composer
supervisor openjdk-17-jre-headless postgresql-client-15 postgresql-15 npm -y
```

```
curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.7/install.sh | bash
```

```
nvm install 18
```

```
source ~/.bashrc
```

```
npm install --global yarn
```

3) загрузить дамп БД для сервиса grant/pre_production — csmp:

```
ДАМП БД CSMP
```

4) загрузить сжатые архивы проектов/сервисов:

```
ПРОЕКТЫ ГРАНТ
```

Наличие директории для каталогов всех проектов и дампов. Например, **/data/**.

– перейти в директорию с архивами и разархивировать проекты/сервисы в директорию **/data/** командой:

```
tar -xzvf название_проекта.tar.gz -C /data/
```

или

```
for project in {pre_production,login-data-service,dwh,front}; do tar -xzvf
$project.tar.gz -C /data/; done
```

– аналогичным образом разархивировать дампы БД:

```
tar -xzvf 2024-05-20.tar.gz -C выбранный_вами_путь
```

– установить права для всех подкаталогов и файлов корневой директории проектов:

```
find . -type d -exec chmod 775 {} \;
find . -type f -exec chmod 664 {} \;
```

5) задать настройки доступа для сервиса PostgreSQL в **/etc/postgresql/15/main/pg_hba.conf**.

6) для пользователей postgres, all и replication_user по образцу, учитывая свою локальную адресацию, не меняя ее (ip-адреса типа 172.16.10.17) предоставить права доступа:

```
# Database administrative login by Unix domain socket
 local   all             postgres                                trust
# TYPE  DATABASE        USER            ADDRESS                 METHOD
# "local" is for Unix domain socket connections only
 local   all             all                                     trust
# IPv4 local connections:
 host    all             all             127.0.0.1/32            trust
 host    all             all             172.16.10.17/32          trust
 host    all             all             172.16.10.9/24           md5
# IPv6 local connections:
 host    all             all             ::1/128                 trust
# Allow replication connections from localhost, by a user with the replication
privilege.
 local   replication     all                                      trust
 host    replication     all             127.0.0.1/32             trust
 host    replication     all             ::1/128                 trust
# Логическая репликация с dwh
 host    replication     replication_user    0.0.0.0/0     md5
```

7) загрузить файл по ссылке и скопировать содержимое файла в файл **/etc/postgresql/15/main/postgresql.conf**:

```
postgresql.conf
```

8) изменить значение в переменной listen_addresses — ip-адрес 172.16.10.17 на ip-адрес своего сервера:

```
sed -i s/172.16.10.17/$(echo `hostname -I`)/
/etc/postgresql/15/main/postgresql.conf
```

9) перезапустить сервис PostgreSQL:

```
service postgresql restart
```

10) загрузить файл по ссылке и скопировать содержимое файла в **/etc/php/8.1/fpm/pool.d/www.conf**:

```
www.conf
```

– установить значение memory_limit в **/etc/php/8.1/fpm/php.ini**:

```
memory_limit = 1024M
```

11) перезапустить сервис php8.1:

```
service php8.1-fpm restart
```

3.6 База данных csmp & центральный сервис grant/pre_production

Внимание! Вводите команды для баз данных от имени пользователя, обладающего:

- правами на создание:
  - ролей
  - баз данных
- имеющего одноименную БД

1) произвести вход в режим командной строки сервиса postgres СУБД PostgreSQL от подходящего под требования выше пользователя:

```
psql -U postgres
```

2) создать пользователей #ваш_юзер1, #ваш_юзер2 и #ваш_юзер3 с паролем #ваш_пароль в системе Postgres:

```
create user #ваш_юзер1  with password '#ваш_пароль1';
create user #ваш_юзер2  with password '#ваш_пароль2';
create user #ваш_юзер3 ;
```

3) создать базу данных имя_бд:

```
create database имя_бд;
```

4) предоставить полный доступ к БД «имя_бд» для пользователей #ваш_юзер1 и #ваш_юзер2:

```
grant all on database имя_бд to #ваш_юзер1, #ваш_юзер2 ;
```

5) следует выйти из режима командной строки системы Postgres:

```
\q;
```

6) применить дамп для БД имя_бд по примеру ниже:

```
psql csmp < путь_к_директории/2024-05-20.sql или sudo -u postgres имя_бд <
путь_к_директории/2024-05-20.sql
```

Внимание! Необходимо выбрать название для веб-сервера (например, example.ru)

3.7 Nginx

1) Убедиться в отключенном состоянии других веб/прокси-серверов:

```
service apache2 stop
```

2) убедиться во включенном состоянии nginx:

```
service nginx start
service nginx status
```

3) удалить файл настроек по умолчанию для nginx:

```
rm /etc/nginx/sites-enabled/default
```

4) загрузить файл по ссылке с ***наименованием веб-сервера*** и поместить в директорию **/etc/nginx/sites-available/**:

```
example.ru
```

5) воспользоваться примером применения настроек nginx для всех бэкенд-сервисов:

```
ln -s /etc/nginx/sites-{available,enabled}/example.ru
nginx -t
service nginx reload
```

Внимание! Предупреждение, касающиеся всех сервисов: базовое имя конфигурационного файла должно соответствовать *server_name* в конфигурационном файле nginx:

```
server_name example.ru;
```

Для всех backend сервисов: pre_production, login-data-service, dwh.

Пример абсолютного пути к конфигурационному файлу:

```
/data/название-back-end-сервиса/config/example.ru.inc
```

1) перейти в директорию проекта/сервиса pre_production:

```
cd /data/pre_production/config/
```

2) создать файл example.ru.inc: и заполнить содержимым ниже:

```
<?php

$configuration = [
    'name'                 => 'example.ru',
    'replication_platform' => 'none',
    'db'                   => [
            'dbName'   => 'имя_бд_lds',
            'host'     => '127.0.0.1',
            'port'     => 5432,
            'user'     => '#ваш_юзер_lds',
            'password' => '#ваш_пароль_lds'
            ],
    'migration' => [
        'db'                => [
            'development' => [
                'type'      => 'pgsql',
                'host'      => '127.0.0.1',
                'port'      => 5432,
                'database'  => 'имя_бд_lds',
                'user'      => '#ваш_юзер_lds',
                'password'  => '#ваш_пароль_lds',
                'directory' => 'back',
            ],
        ],
        'schema_info'       => 'schema_info',
        'schema_migrations' => 'schema_migrations',
        'migrations_dir'    => ['default' => PATH_ROOT . '/migrations'],
        'db_dir'            => PATH_ROOT . '/db',
        'log_dir'           => PATH_ROOT . '/logs',
        'base_dir'          => PATH_ROOT . '/vendor/rt-cis/migration',
```

```
    ],
    'DEBUG_ENABLE' => true
];
```

3) предоставить полные права на таблицу schema_migrations пользователю #ваш_юзер1;

```
psql -U postgres -d имя_бд
GRANT ALL ON TABLE schema_migrations TO #ваш_юзер1;
\q
```

4) указать свой ip-адрес и произвести минимальную проверку работы сервиса:

```
curl -i -H "Accept: application/json" -H "Content-Type: application/json" -X
GET http://$(echo `hostname -I`):80/staff/list
```

3.8 База данных lds_db & сервис авторизации login-data-service

1) Использовать sql -U postgres и создать пользователя и базу данных для сервиса авторизации:

```
create user #ваш_юзер_lds with password '#ваш_пароль_lds';
create database имя_бд_lds;
grant all on database имя_бд_lds to #ваш_юзер_lds;
\c имя_бд_lds
GRANT CREATE ON SCHEMA public TO #ваш_юзер_lds;
\q
```

2) перейти в директорию проекта:

```
cd /data/login-data-service/config
```

3) создать файл example.ru.inc и заполнить содержимым ниже:

```
<?php

$configuration = [
    'name'                 => 'example.ru',
    'replication_platform' => 'none',
    'db'                   => [
            'dbName'   => 'имя_бд_lds',
            'host'     => '127.0.0.1',
            'port'     => 5432,
            'user'     => '#ваш_юзер_lds',
            'password' => '#ваш_пароль_lds'
            ],
```

```
    'migration' => [
        'db'                => [
            'development' => [
                'type'      => 'pgsql',
                'host'      => '127.0.0.1',
                'port'      => 5432,
                'database'  => 'имя_бд_lds',
                'user'      => '#ваш_юзер_lds',
                'password'  => '#ваш_пароль_lds',
                'directory' => 'back',
            ],
        ],
        'schema_info'       => 'schema_info',
        'schema_migrations' => 'schema_migrations',
        'migrations_dir'    => ['default' => PATH_ROOT . '/migrations'],
        'db_dir'            => PATH_ROOT . '/db',
        'log_dir'           => PATH_ROOT . '/logs',
        'base_dir'          => PATH_ROOT . '/vendor/rt-cis/migration',
    ],
    'DEBUG_ENABLE' => true
];
```

4) создать файл proxy_settings.php по образцу и заполнить содержимым ниже, но необходимо указать свой ip-адрес для стенда:

```
<?php

use App\Controller\ProxyDwhController;
use App\Controller\ProxyGrantController;

return [
//  'current_stage'    => 'locale',
  'current_stage'    => 'prod_stand',

  'proxy_url_prefix' => [
    'grant' => '/proxy/grant',
    'dwh'   => '/proxy/dwh',
  ],

  'proxy_handlers' => [
    'grant' => ProxyGrantController::class,
    'dwh'   => ProxyDwhController::class,
  ],

  'service'          => [
    'pre_production' => [
      'locale'     => [
        'url'        => 'nginx-1:80',
        'auth_token' => '',
      ],
```

```
      'prod_stand' => [
        'url'        => 'http://172.16.10.17:80',
        'auth_token' => 'Basic Z3JhbnQ6cnRjaXNfZ3JhbnQ=',
      ],
    ],
    'dwh'   => [
      'locale'     => [
        'url'        => 'nginx-1:82',
        'auth_token' => '',
      ],
      'prod_stand' => [
        'url'        => 'http://172.16.10.17:82',
        'auth_token' => 'Basic Z3JhbnQ6cnRjaXNfZ3JhbnQ=',
      ],
    ],
  ],
  'middleware' => [
    'default_proxy_object' => ['auth:access', 'permission:can']
  ]
];
```

```
sed -i s/172.16.10.17/$(echo `hostname -I`)/g /data/login-data-
service/config/proxy_settings.php
```

5) создать файл pzi_settings.php и заполнить содержимым ниже.

Содержимое файла конфигурации pzi_settings.php:

```
<?php
 return [
 ////////////////////////////////////////////////////////////////
 // SL2 = Уровень защиты 2. SL3 = Уровень защиты 3.
 ////////////////////////////////////////////////////////////////
 //    'security_level' => 'SL3',
     'security_level' => 'SL2',
 ////////////////////////////////////////////////////////////////
 // IDS = система находится в режиме обнаружения вторжений. В этом случае если
в систему приходит запрос с
 // неизвестным endpoint, система его ПРОПУСКАЕТ, но записывает информацию об
обнаружении вторжения для администратора;
 // IPS = система находится в режиме предотвращения вторжений. В этом случае
если в систему приходит запрос с
 // неизвестным endpoint, система его НЕ ПРОПУСКАЕТ, и записывает информацию об
обнаружении вторжения для администратора.
 ////////////////////////////////////////////////////////////////
     'intrusion_system' => 'IDS',
 //    'intrusion_system' => 'IPS',
 ////////////////////////////////////////////////////////////////
```

```
 // Срок в течении которого запрещено повторное использование идентификатора
пользователя
 // По умолчанию 365 дней
 //
 //
 // * Возможно будет использоваться в качестве указателя - в течение какого
времени данные должны храниться в БД *
 // * Прежде чем будут перенесены в таблицу истории
*
 //
 ////////////////////////////////////////////////////////////////
     'reuse_uuid_period' => 365,
 ////////////////////////////////////////////////////////////////
 // Минимальная сложность пароля с определяемыми оператором требованиями к
регистру,
 // количеству символов, сочетанию букв верхнего и нижнего регистра, цифр и
специальных символов:
 //
 // для УЗ2 по умолчанию алфавит не менее 70 символов (проверить и подобрать
алфавит): a..zA..Z0..9!@#$%^&()+"№;:?,./| // для УЗ3 по умолчанию алфавит не
менее 60 символов (проверить и подобрать алфавит): a..zA..Z0..9!@#$%^ // //
 // * Предполагается использование как настройки, по этому тут будет храниться
регулярное выражение.             *
 //
 ////////////////////////////////////////////////////////////////
     'password_complexity' => [
         'SL2' => '/^[a-zA-Z0-9!@#$%^&()+"№;:?,.\/|]+$/',
         'SL3' => '/^[a-zA-Z0-9!@#$%^]+$/',
     ],
 ////////////////////////////////////////////////////////////////
 // Минимального количества измененных символов при создании новых паролей:
 // для УЗ2 от 3 до 8 попыток, по умолчанию - 8;
 // для УЗ3 от 3 до 10 попыток, по умолчанию - 10.
 //
 //
 // * По причине того, что мы не храним пароли в БД, мы не сможем реализовать
это требование.                    *
 //
 ////////////////////////////////////////////////////////////////
     'min_changed_characters' => [
         'SL2' => [3, 8],
         'SL3' => [3, 10],
     ],
 ////////////////////////////////////////////////////////////////
 // Блокировка программно-технического средства или учетной записи пользователя
 // в случае достижения установленного максимального количества неуспешных
попыток аутентификации:
 //
 //
 // для УЗ2 от 10 до 30 минут, по умолчанию - 10;
 // для УЗ3 от 5 до 30 минут, по умолчанию - 5.
```

```
////////////////////////////////////////////////////////////////
    'session_lock_duration' => [
        'SL2' => 10,
        'SL3' => 5,
    ],
////////////////////////////////////////////////////////////////
// Количество попыток для аунтификации
//
//
// для УЗ2 попыток - 10;
// для УЗ3 попыток - 5;
////////////////////////////////////////////////////////////////
    'maximum_attempt' => [
        'SL2' => 10,
        'SL3' => 5,
    ],
////////////////////////////////////////////////////////////////
// Смена паролей:
//
//
// для УЗ2 не более чем через 90 дней, по умолчанию - 90;
// для УЗ3 не более чем через 120 дней, по умолчанию - 120.
////////////////////////////////////////////////////////////////
    'password_change_interval' => [
        'SL2' => 90,
        'SL3' => 120,
    ],
////////////////////////////////////////////////////////////////
// Длина пароля пользователя:
//
//
// для УЗ2 не менее чем 8 символов, по умолчанию - 8;
// для УЗ3 не менее чем 8 символов, по умолчанию - 8;
////////////////////////////////////////////////////////////////
    'min_user_password_length' => [
        'SL2' => 8,
        'SL3' => 8,
    ],
////////////////////////////////////////////////////////////////
// Длина пароля администратора:
//
//
// для УЗ2 не менее чем 12 символов, по умолчанию - 12;
// для УЗ3 не менее чем 12 символов, по умолчанию - 12;
////////////////////////////////////////////////////////////////
    'min_admin_password_length' => [
        'SL2' => 12,
        'SL3' => 12,
    ],
////////////////////////////////////////////////////////////////
// Минимальное время действия пароля:
```

```
 //
 //
 // по умолчанию не менее 30 дней.
 ////////////////////////////////////////////////////////////////
     'min_password_expiry' => 30,
 ////////////////////////////////////////////////////////////////
 // Запрет на использование пользователями определенного оператором числа
последних использованных паролей
 // при создании новых паролей:
 //
 //
 // по умолчанию - 5 последних паролей;
 ////////////////////////////////////////////////////////////////
     'number_of_previous_passwords_disallowed_to_repeat' => 5,
 ////////////////////////////////////////////////////////////////
 // В информационной системе обеспечивается блокирование сеанса доступа
пользователя
 // после времени бездействия (неактивности) пользователя:
 //
 //
 // для УЗ2 до 15 минут, по умолчанию - 15;
 // для УЗ3 - не устанавливается.
 ////////////////////////////////////////////////////////////////
     'session_timeout' => [
         'SL2' => 15,
         'SL3' => null,
     ],
 ////////////////////////////////////////////////////////////////
 // Срок через который идентификатор пользователя должен быть автоматически
заблокирован, если он не использовал систему:
 //
 //
 // для УЗ2 до 90 дней, по умолчанию - 90 дней;
 // для УЗ3 - не устанавливается.
 ////////////////////////////////////////////////////////////////
     'auto_block_period' => [
         'SL2' => 90,
         'SL3' => null,
     ],
 ////////////////////////////////////////////////////////////////
 // Количество циклов неудачных входов, после которых идентификатор приложения
должен быть автоматически заблокирован:
 //
 //
 // для УЗ2 - 2 цикла;
 // для УЗ3 - 3 цикла.
 ////////////////////////////////////////////////////////////////
     'cycle_before_locking' => [
         'SL2' => 2,
         'SL3' => 3,
     ],
```

```
];
```

Внимание! Обращайте внимание на вывод команд запуска скриптов!

Шаблон возможных флагов/опций при запуске PHP-скриптов для разных backend сервисов могут отличаться:

```
--db_config/--config = server_name
```

– перейти в корневую директорию проекта/сервиса:

```
cd /data/login-data-service/
```

– применить миграции:

```
php cli/db_migration.php db:migrate --config=example.ru
```

– установить пакет redoc-cli:

```
npm install -g redoc-cli
```

– запустить скрипт создания суперпользователя:

```
php ./cli/pzi_root_init.php --config=example.ru --last_name=Рутов --
first_name=Рут --father_name=Рутович --password=tSwG48y1rdsmDhFmaY5PxNH8v9farf
```

– запустить скрипт импорта данных из grant:

```
php cli/import_from_grant.php --config=example.ru
```

– запустить скрипт импорта данных URI:

```
php ./cli/import_uri.php --config=example.ru
```

– запустить скрипт, подготавливающий тестовые аккаунты:

```
php cli/prepare_test_accounts.php --config=example.ru
```

– применить заполнение БД справочной информацией:

```
php cli/seeder.php --config=example.ru
```

– запустить скрипт генерации документации:

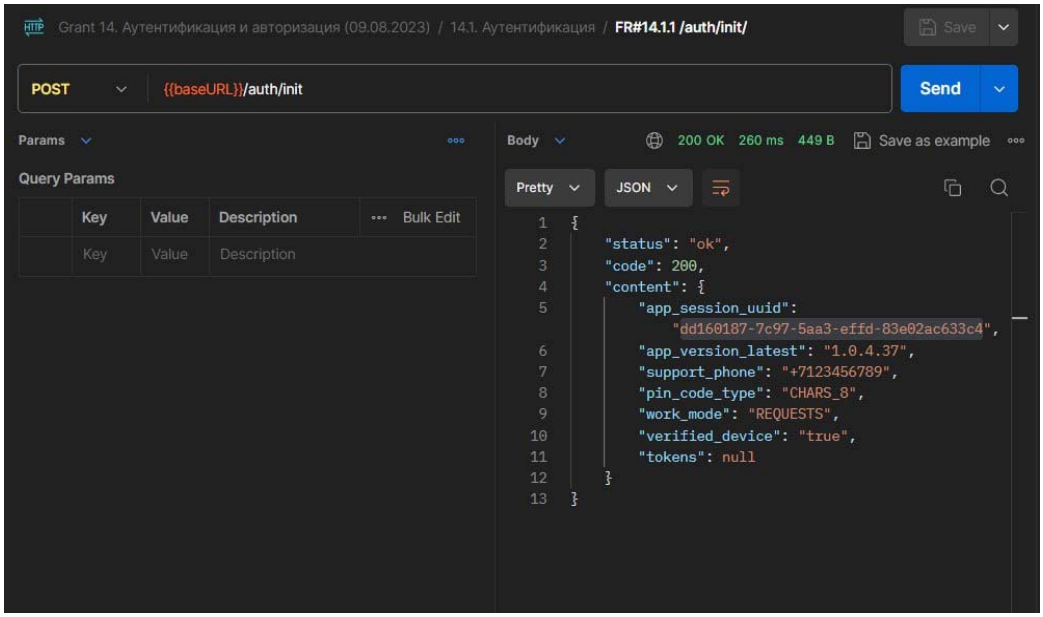
```
php ./cli/generate_swagger_doc.php --config=example.ru
```

– проверить работу сервиса (необязательно, если затруднительно):

Инициализация:

```
curl --location "http://$(echo `hostname -I`):81/auth/init" \
 --header 'Content-Type: application/json' \
 --header 'Authorization: Basic Z3JhbnQ6cnRjaXNfZ3JhbnQ=' \
 --data '{
     "app_uniq_id": "b0954d2d-9541-4ca0-8bc0-af776a87e3cf",
     "app_session_init_datetime": "20231010235959",
     "app_version": "1.0.4.37",
     "app_token": "123",
     "platform": "android",
     "model": "Postman",
     "refresh_token":
"eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJjb2RlX3R5cGUiOiJWS0JaUEdCVFE4T1YiLCJpY
XQiOjE3MTQ5OTYwNDIsImV4cCI6MTcxNzU4ODA0Mn0.x9pWUed2D9g58G4_aJOZ5uBRMfVxLdmowcQ_
peO0dKM"
 }'
```

– получить «app_session_uuid»:



– использовать полученный app_session_uuid в следующем запросе:

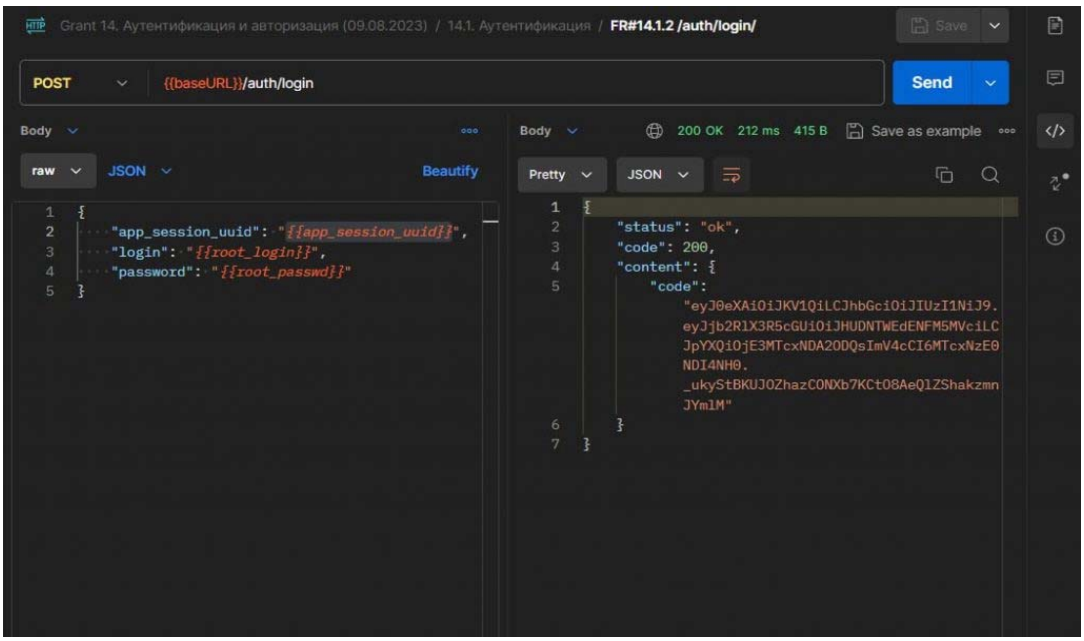
Запросы curl отредактированы в текстовом редакторе Sublime Text.

Вход в систему:

```
curl --location "http://$(echo `hostname -I`):81/auth/login" \
 --header 'Content-Type: application/json' \
 --header 'Authorization: Basic Z3JhbnQ6cnRjaXNfZ3JhbnQ=' \
 --data '{
     "app_session_uuid": "ВАШ APP_SESSION_UUID",
     "login": "adm_root",
```

```
    "password": "tSwG48y1rdsmDhFmaY5PxNH8v9farf"
}'
```

– получить код:

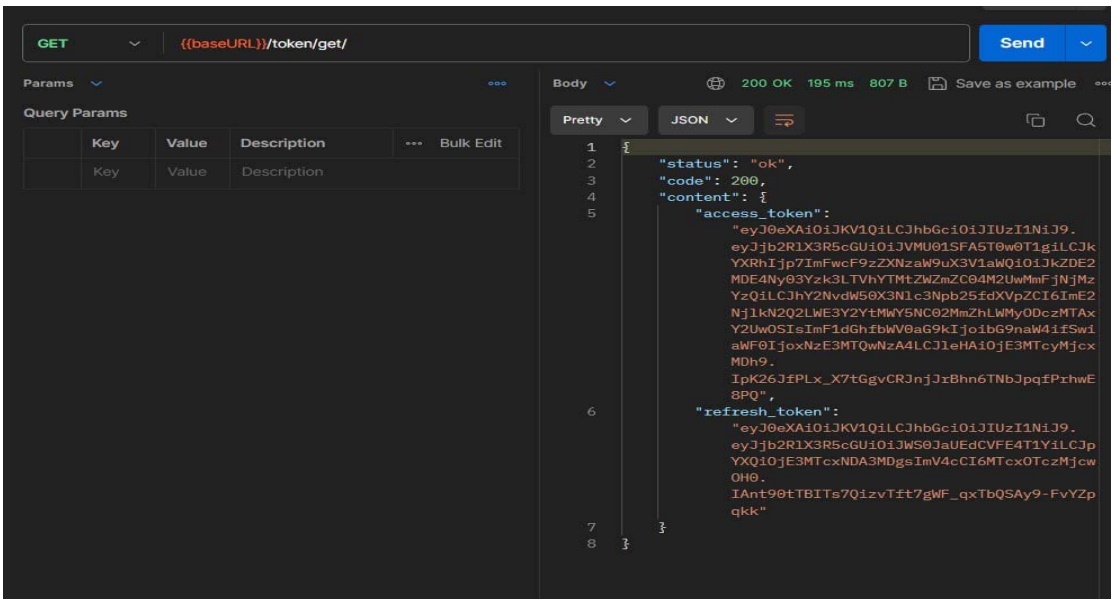


– использовать в следующем запросе:

Получение token (ключа):

```
curl --location --request GET "http://$(echo `hostname -I`):81/token/get/" \
 --header 'app_session_uuid: ВАШ APP_SESSION_UUID' \
 --header 'code: ВАШ CODE' \
 --header 'Content-Type: application/json' \
 --header 'Authorization: Basic Z3JhbnQ6cnRjaXNfZ3JhbnQ=' \
 --data '{
    "app_session_uuid": "ВАШ APP_SESSION_UUID",
    "code": "ВАШ CODE"
 }'
```

– получить access_token:



Пример использования сервиса авторизации в качестве proxy:

```
curl --location "http://$(echo `hostname -I`):81/proxy/grant/employee/list" \
 --header 'access_token: ВАШ ACCESS_TOKEN' \
 --header 'Content-Type: application/json' \
 --header 'Authorization: Basic Z3JhbnQ6cnRjaXNfZ3JhbnQ=' \
 --data '{
     "entity_condition": {

}, "search": "E1A39B384294919058AD6D82"

}'
```

3.9 Репликация базы данных имя_бд & сервис отчетов dwh

1) Перейти в конфигурационную директорию проекта:

```
cd /data/dwh/config/
```

– создать файл example.ru.inc и заполнить содержимым ниже:

```
<?php

$configuration = [
    'name'                 => 'local.com',
    'replication_platform' => 'none',
    'db'                   => [

            'dbName'   => 'dwh_имя_бд',
            'host'     => '127.0.0.1',
            'port'     => 5432,
            'user'     => '#ваш_юзер1',
            'password' => '#ваш_пароль1'
```

```
    ],
    'migration'            => [
        'db'                => [
            'development' => [
                'type'      => 'pgsql',
                'host'      => '127.0.0.1',
                'port'      => 5432,
                'database'  => 'dwh_имя_бд',
                'user'      => '#ваш_юзер1',
                'password'  => '#ваш_пароль1',
                'directory' => 'back',
            ],
        ],
        'schema_info'       => 'schema_info',
        'schema_migrations' => 'schema_migrations',
        'migrations_dir'    => ['default' => PATH_ROOT . '/migrations'],
        'db_dir'            => PATH_ROOT . '/db',
        'log_dir'           => PATH_ROOT . '/logs',
        'base_dir'          => PATH_ROOT . '/vendor/rt-cis/migration',
    ],
    'DEBUG_ENABLE' => true
];;
```

2) использовать клиента psql -U postgres и создать пользователя с правами доступа:

```
CREATE USER replication_user WITH REPLICATION PASSWORD 'replication_pass';
GRANT CONNECT ON DATABASE имя_бд TO replication_user; GRANT pg_read_all_data TO
replication_user;
```

3) создать publication и replication_slot на оригинальной БД имя_бд:

```
\c имя_бд
CREATE PUBLICATION имя_бд_publication FOR TABLES IN SCHEMA eav, plain;
SELECT pg_create_logical_replication_slot('имя_бд_slot', 'pgoutput');
```

4) проверить replication_slot:

```
select * from pg_catalog.pg_replication_slots;
\q
```

5) снять дамп схем с БД имя_бд:

```
sudo -u postgres pg_dump --schema-only имя_бд >
путь_к_директории/базовое_имя_файла.sql
psql -U postgres
```

6) создать БД dwh_имя_бд:

```
create database dwh_имя_бд;
grant all on database dwh_имя_бд to #ваш_юзер1, #ваш_юзер2, replication_user;
\q
```

7) применить дамп схемы к dwh_имя_бд:

```
sudo -u postgres psql -d dwh_имя_бд < путь_к_директории/базовое_имя_файла.sql
```

8) на стороне реплицирующей БД (dwh_csmp) создать подписку:

```
CREATE SUBSCRIPTION csmp_subscription CONNECTION 'host=localhost port=5432
dbname=имя_бд user=replication_user password=replication_pass' PUBLICATION
имя_бд_publication WITH (create_slot = false, slot_name='имя_бд_slot');
\q
```

9) применить миграции и заполнить справочники:

```
cd /data/dwh
php cli/db_migration.php db:migrate VERSION=-1000 --config=example.ru
php cli/db_migration.php db:migrate --config=example.ru
php ./cli/Seeder.php --config=example.ru --type=dictionary
php ./cli/Seeder.php --config=example.ru --type=custom
```

Убедиться в наличии директорий и файла: QueueWorker/output.log в **/data/dwh/logs/**, если нет, то создать директорию и пустой файл:

Проверить работоспособность репликации:

```
psql -U postgres -d dwh_имя_бд
select * from eav.dictionary_entity_value;
SELECT * FROM pg_stat_replication;
```

3.10 Frontend сервис

1) Перейти в директорию проекта:

```
cd /data/front/
```

2) создать конфигурационный файл .env.production с содержимым:

```
VITE_PUBLIC_PATH='/'
VITE_API_BASE_URL='/api'
VITE_API_AUTH='http://172.16.10.17:81' # 'http://nginx-1:81' # for run in
docker
VITE_API_AUTH_BASE_URL='/pzi'
VITE_API_URL='http://172.16.10.17:80' # 'http://nginx-1:80' # for run in docker
VITE_API_DWH_BASE_URL='/dwh'
VITE_API_DWH='http://172.16.10.17:82'
```

```
# For Dev
VITE_DEV_API_PORT='4000'
VITE_OFF_AUTH_PROXY=false
VITE_MOCK_BROWSER=false
VITE_IGNORE_CASL=false
NODE_ENV='production'
```

```
sed -i s/172.16.10.17/$(echo `hostname -I`)/g /data/front/.env.production
```

3) установить пакеты:

```
yarn
```

4) собрать проект:

```
yarn build
```

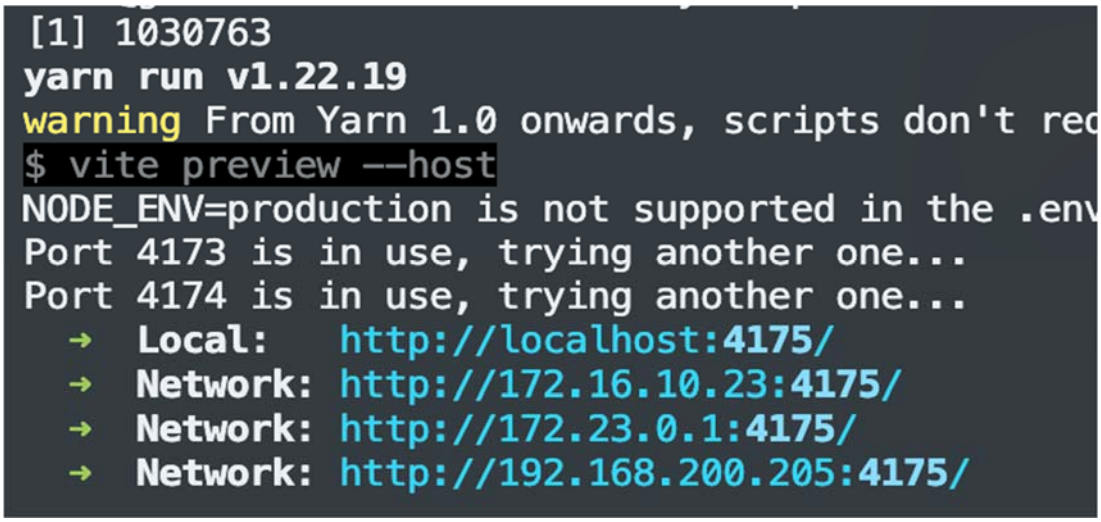
5) узнать свой ip-адрес:

```
hostname -I
```

6) запустить процесс frontend сервиса в фоне:

```
yarn preview -- --host &
```

7) найти в выводе свой ip-адрес и выделенный порт по примеру:



8) выйти из фонового режима:

```
Ctrl + A + D
```

3.11 Создание и запуск фоновых служб системы в supervisor

1) Перейти в директорию/Создать **/etc/supervisor/conf.d/**:

```
cd /etc/supervisor/conf.d/
```

2) создать файл pre_production.conf с содержимым:

```
[program:pre_production_queue]
command=bash -c 'while true; do php /data/pre_production/cli/console process-
queue --db_config=example.ru; sleep 20; done'
user=www-data
```

3) создать файл dwh.conf с содержимым:

```
[program:dwh_report]
directory=/data/dwh/
command=bash -c 'while true; do /usr/bin/php cli/ReportsWorker.php --
config=example.ru >> logs/QueueWorker/output.log 2>&1; sleep 5; done'
user=root
process_name=%(program_name)s_%(process_num)02d
```

4) создать файл telematics.conf с содержимым:

```
[program:telematics]
directory=/data/pre_production/
command=bash -c 'while true; do /usr/bin/php cli/console telematic:update-
coordinates --db_config=example.ru; sleep 15; done'
user=root
process_name=%(program_name)s_%(process_num)02d
```

5) перезапустить службу supervisor:

```
service supervisor restart
```

ЕЦП СМП установлена и полностью готова к работе в штатном режиме.

4 ПРОВЕРКА ПРОГРАММЫ

Для проверки программы необходимо выполнить следующие действия:

– Открыть в браузере новую страницу и в поле адрес ввести: http://172.16.0.17. Открывается форма авторизации программы (рисунок 2).

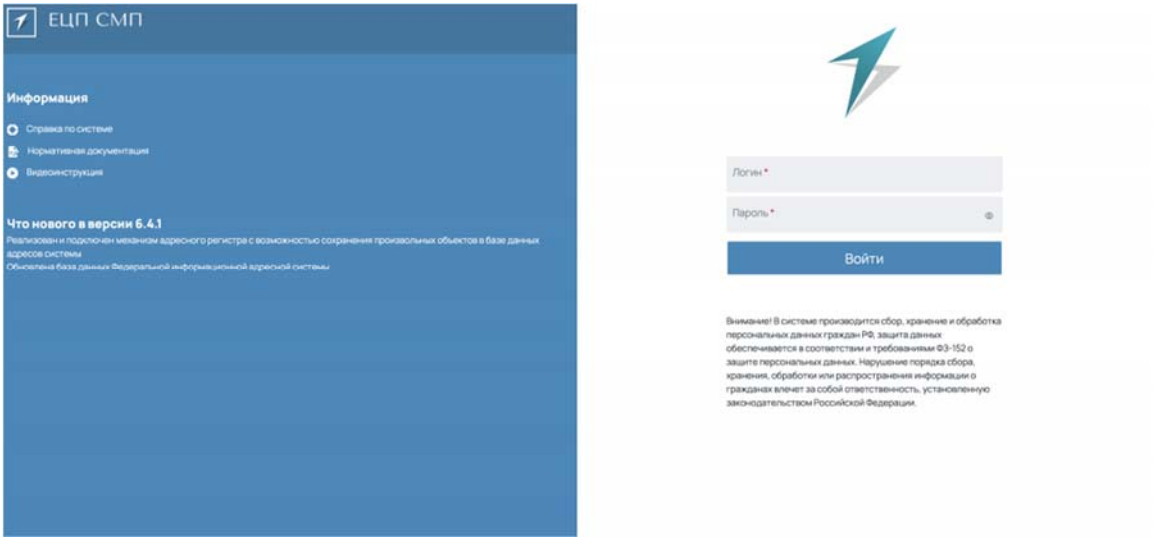


Рисунок 2 — Окно авторизации ЕЦП СМП

– ввести логин и пароль администратора:

Логин: adm_root;

Пароль: tSwG48y1rdsmDhFmaY5PxNH8v9farf.

– нажать кнопку <Войти> — откроется главное меню ЕЦП СМП (рисунок 3) с полем учетной записи пользователя в панели главного меню.

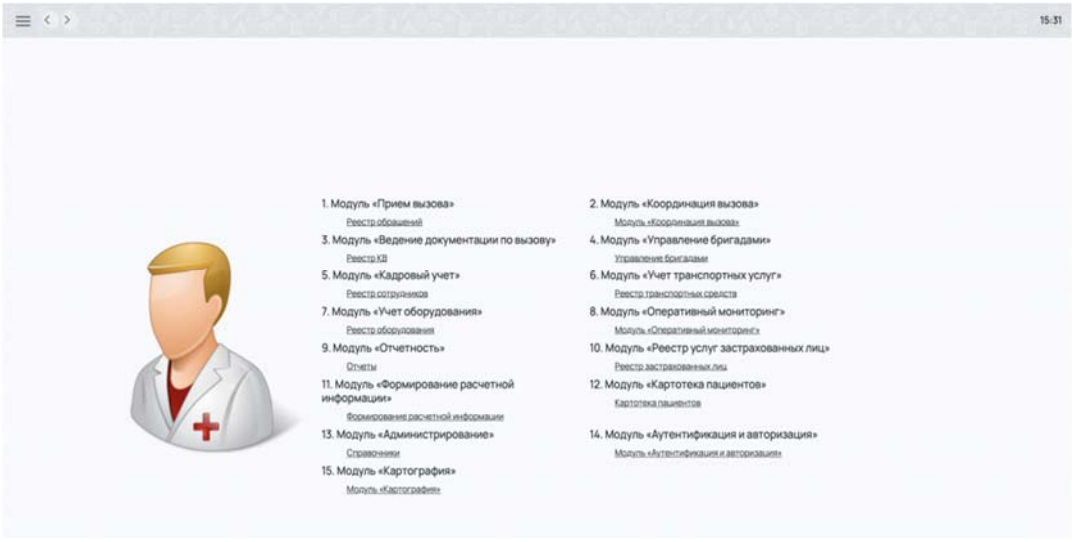


Рисунок 3 – Главное меню ЕЦП СМП

5 СООБЩЕНИЯ АДМИНИСТРАТОРУ

Сообщения администратору подразделяются на:

- диагностические сообщения;
- сообщения об ошибках.

Все сообщения протоколируются в журналах логов, расположенных в папке **/data/www/logs/**.

5.1 Сообщения об ошибках

Сообщения об ошибках сервера, выдаваемые системой программисту, приведены в таблице 6.

Таблица 6 — Сообщения об ошибках сервера

№ п/п	Ошибка	Условия выдачи сообщения
1	Ошибки взаимодействия с БД	
1.1	ОШИБКА: Unable to complete network request to host «127.0.0.1». Failed to establish a connection. Подключение не установлено, т.к. конечный компьютер отверг запрос на подключение	Сообщение отображается при невозможности соединения с БД
2	Ошибки конфигурационного файла	
2.1	ОШИБКА: Ошибка сокета 10049 при выполнении bind. Требуемый адрес для своего контекста неверен	Неверный IP адрес или порт в конфигурационном файле

5.2 Протоколирование ошибок и предупреждений

Все ошибки записываются в лог с началом записи «ОШИБКА».

Все предупреждения записываются в лог с началом записи «ПРЕДУПРЕЖДЕНИЕ».