

**ПРОГРАММНЫЙ КОМПЛЕКС
«АВАНГАРД. РАБОЧЕЕ ПРОСТРАНСТВО»**

РУКОВОДСТВО СИСТЕМНОГО ПРОГРАММИСТА

Листов 104

Москва
2023

АННОТАЦИЯ

Документ содержит общие сведения о назначении и архитектуре Программного комплекса «Авангард. Рабочее пространство» (далее – Система), требования к аппаратному и программному обеспечению Системы, к квалификации персонала, сопровождающего Систему. Подробно описаны процессы жизненного цикла программного обеспечения на стадиях внедрения и сопровождения:

- первоначальная установка и настройка программного обеспечения Системы;
- поддержание работоспособности программного обеспечения Системы, в том числе восстановление Системы после сбоев;
- обновление программного обеспечения Системы.

Документ не содержит описания процесса вывода программного обеспечения Системы из эксплуатации.

СОДЕРЖАНИЕ

| | |
|---|----|
| 1. Общие сведения о системе..... | 6 |
| 1.1. Назначение Системы..... | 6 |
| 1.2. Состав Системы | 6 |
| 1.3. Требования к техническому и программному обеспечению | 7 |
| 1.4. Требования к квалификации персонала, обеспечивающего эксплуатацию Системы | 9 |
| 2. Установка и настройка программного обеспечения | 10 |
| 2.1. Предварительные шаги установки дистрибутива | 10 |
| 2.2. Установка и развертывание сервиса «Администрирование и контроль доступа (Хаб)» | 11 |
| 2.3. Установка и развертывание сервиса «Файловое хранилище»..... | 20 |
| 2.4. Установка и развертывание сервиса «Справочники»..... | 23 |
| 2.5. Установка и развертывание сервиса «Уведомления»..... | 27 |
| 2.6. Установка и развертывание сервиса «Распознавание речи» | 31 |
| 2.7. Установка и развертывание сервиса «Хранение и оборот документов» | 34 |
| 2.8. Установка и развертывание сервиса «Штатная расстановка» | 38 |
| 2.9. Установка и развертывание сервиса «Профиль пользователя» | 41 |
| 2.10. Установка и развертывание сервиса «Личные заметки» | 45 |
| 2.11. Установка и развертывание сервиса «Управление проектами и задачами» | 49 |
| 2.12. Установка и развертывание сервиса «Адресная книга»..... | 53 |
| 2.13. Установка и развертывание сервиса «Табель учета рабочего времени»..... | 57 |
| 2.14. Установка и развертывание сервиса «Коммуникации» | 60 |
| 2.15. Установка и развертывание сервиса «Интерактивные рабочие столы и аналитика» | 67 |
| 2.16. Установка и развертывание сервиса «Оболочка»..... | 70 |
| 3. Обновление программного обеспечения..... | 75 |
| 3.1. Обновление сервиса «Администрирование и контроль доступа (Хаб)» | 75 |
| 3.2. Обновление сервиса «Файловое хранилище» | 75 |
| 3.3. Обновление сервиса «Справочники» | 75 |
| 3.4. Обновление сервиса «Уведомления» | 76 |

| | | |
|-------|--|-----|
| 3.5. | Обновление сервиса «Распознавание речи» | 76 |
| 3.6. | Обновление сервиса «Хранение и оборот документов» | 76 |
| 3.7. | Обновление сервиса «Штатная расстановка» | 77 |
| 3.8. | Обновление сервиса «Профиль пользователя» | 77 |
| 3.9. | Обновление сервиса «Личные заметки» | 78 |
| 3.10. | Обновление сервиса «Управление проектами и задачами»..... | 78 |
| 3.11. | Обновление сервиса «Адресная книга» | 79 |
| 3.12. | Обновление сервиса «Табель учета рабочего времени» | 79 |
| 3.13. | Обновление сервиса «Коммуникации» | 79 |
| 3.14. | Обновление сервиса «Интерактивные рабочие столы и аналитика» 80 | |
| 3.15. | Обновление сервиса «Оболочка» | 80 |
| 4. | Проверка, восстановление и поддержание работоспособности программного обеспечения | 81 |
| 4.1. | Методы проверки работоспособности рабочих станций | 81 |
| 4.2. | Методы проверки работоспособности сервера | 81 |
| 4.3. | Методы проверки работоспособности базы данных | 82 |
| 4.4. | Методы восстановления работоспособности сервера | 84 |
| 4.5. | Методы восстановления работоспособности базы данных | 84 |
| 4.6. | Методы поддержания целостности базы данных | 85 |
| 4.7. | Методы поддержания безопасности базы данных | 86 |
| 4.8. | Методы диагностирования проблем обновления баз данных | 86 |
| 5. | Администрирование программного обеспечения | 88 |
| 5.1. | Доступ к сервису..... | 89 |
| 5.2. | Управление доступом..... | 90 |
| 5.3. | Константы..... | 102 |
| 6. | Аварийные ситуации | 104 |

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ

| Сокращенное наименование | Полное наименование |
|--------------------------|--|
| БД | База данных |
| ОС | Операционная система |
| Система | Программное обеспечение «Программный комплекс «Авангард. Рабочее пространство» |
| СУБД | Система управления базами данных |

1. ОБЩИЕ СВЕДЕНИЯ О СИСТЕМЕ

1.1. Назначение Системы

Программное обеспечение «Программный комплекс «Авангард. Рабочее пространство» предназначено для организации единого рабочего пространства организации, в том числе для обеспечения ведения проектной деятельности организации, с возможностью расширения набора автоматизируемых процессов благодаря использованию модульной архитектуры.

1.2. Состав Системы

Система представляет собой web-приложение и включает в свой состав следующий набор компонент:

- системные компоненты:
 - сервис «Администрирование и контроль доступа (Хаб)» (далее также – Хаб);
 - сервис «Файловое хранилище»;
 - сервис «Справочники»;
 - сервис «Уведомления»;
 - сервис «Оболочка»;
 - сервис «Распознавание речи».
- прикладные компоненты:
 - сервис «Хранение и оборот документов»;
 - сервис «Штатная расстановка»;
 - сервис «Профиль пользователя»;
 - сервис «Личные заметки»;
 - сервис «Управление проектами и задачами»;
 - сервис «Адресная книга»;
 - сервис «Табель учета рабочего времени»;
 - сервис «Коммуникации»;
 - сервис «Интерактивные рабочие столы и аналитика».

Прикладные компоненты состоят из:

- Базы данных;
- Сервера приложения;
- Клиентского приложения (фрагментарно загружается на машину пользователя).

В рамках Системы используются вспомогательные сторонние сервисы с открытым исходным кодом:

- Программный брокер сообщений (RabbitMQ);
- Платформа для обнаружения и анализа контента (Apache Tika).

1.3. Требования к техническому и программному обеспечению

1.3.1. Требования к программному обеспечению

Состав программного обеспечения пользовательской рабочей станции:

- браузер Google Chrome, Microsoft Edge, Яндекс Браузер последней или предпоследней версии;
- наличие актуальной версии установленных драйверов оборудования.

Требования к программному обеспечению тестового и продуктивного серверов:

- серверная операционная система семейства Linux;
- СУБД PostgresPro либо PostgreSQL версии не ниже 11;
- СУБД MongoDB 6 или MinIO;
- прокси Nginx не ниже 1.22;
- программная платформа .NET sdk 6.0.

1.3.2. Требования к техническому обеспечению

К аппаратному обеспечению серверной части, предназначенной для обеспечения функционирования Системы (не более 100 одновременных подключений пользователей), предъявляются следующие минимальные требования:

- Процессоры:
 - количество не менее 2;
 - архитектура процессора x86-64;
 - ядер не менее 8;
 - потоков не менее 16;
 - тактовая частота в режиме повышенной нагрузки не менее 3,3 ГГц;
 - кэш не менее 20 Мб;
 - поддержка памяти ECC.
- Оперативная память:
 - объем не менее 64 Гб;
 - тип DDR3/DDR4/DDR5 с функцией коррекции ошибок.

- Сетевой интерфейс:
 - не менее 1 порта 100 Мб/с.
- Дисковая подсистема:
 - аппаратный RAID;
 - интерфейс SAS не менее 6 Гб/сек;
 - HDD с буфером обмена не менее 128 Мб либо SSD.
- Коммуникационная среда должна обеспечивать информационное взаимодействие между компонентами Системы в соответствии с транспортным протоколом TCP/IP.

К аппаратному обеспечению рабочей станции пользователя, предназначенной для обеспечения доступа к функционалу Системы, предъявляются следующие требования:

- Процессоры:
 - количество не менее 1;
 - архитектура процессора x86-64;
 - ядер не менее 2;
 Допустимо использование следующих видов процессоров:
 - настольные процессоры Intel и AMD, вышедшие на рынок не ранее 2014 года;
 - мобильные процессоры Intel и AMD, вышедшие на рынок не ранее 2016 года, кроме линейки процессоров Intel Atom;
 - процессоры Apple линейки M1, M2 и более новые.
- Оперативная память:
 - объем не менее 4 Гб (рекомендуется 8 Гб);
 - тип DDR3/DDR4/DDR5.
- Сетевой интерфейс:
 - не менее 1 порта 100Мб/с
 - (доступ к сервисам системы со скоростью не ниже 8 Мбит/с (для быстрой загрузки приложения рекомендуется 25 Мбит/с и выше).
- Дисковая подсистема:
 - HDD с буфером обмена не менее 64 Мб либо SSD.
- Графический режим монитора:
 - 1366x768 и выше (рекомендуется 1920x1080).
- Клавиатура, мышь.

1.4. Требования к квалификации персонала, обеспечивающего эксплуатацию Системы

Персонал, выполняющий функции технического сопровождения Системы, должен обладать экспертными навыками:

- обеспечения функционирования серверов и рабочих станций в среде ОС семейства Linux;
- установки и настройки программного обеспечения, приведённого в п. 2.

2. УСТАНОВКА И НАСТРОЙКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

2.1. Предварительные шаги установки дистрибутива

2.1.1. Установка Postgresql

Установка СУБД на примере дистрибутива Postgresql-11 из базового репозитория AstraLinux 1.7:

Обновить информацию о пакетах и установить СУБД:

```
apt update  
apt install postgresql-11
```

2.1.2. Установка Dotnet CLI

Необходимо установить среду исполнения *dotnet*.

1) Скачать дистрибутив по ссылке:

<https://dotnet.microsoft.com/en-us/download/dotnet/thank-you/sdk-6.0.403-linux-x64-binaries>

2) Скопировать полученный архив в пустой каталог на сервер с помощью SCP-клиента (для windows можно использовать WinSCP)

3) Распаковать архив командой:

```
tar xvf dotnet-sdk-6.0.403-linux-x64.tar.gz
```

Если планируется, что web-сервер Kestrel будет использовать https в качестве ApplicationUrl приложений, то следует выполнить команду для установки сертификатов *dotnet*:

```
dotnet dev-certs https --clean  
dotnet dev-certs https -t
```

2.1.3. Установка Rabbit MQ

Установка проводится следующим образом:

```
apt update  
apt install rabbitmq-server
```

2.1.4. Установка nginx

1) Необходимо подключить Extended-репозиторий Astralinux 1.7. Для этого внесем изменения в файл */etc/apt/sources.list* командой:

```
nano /etc/apt/sources.list
```

Добавим текст с репозиториями:

```
# Основной репозиторий
```

```
deb https://dl.astralinux.ru/astra/stable/1.7_x86-64/repository-main/ 1.7_x86-64 main contrib non-free
```

```
# Оперативные обновления основного репозитория
deb https://dl.astralinux.ru/astra/stable/1.7_x86-64/repository-update/ 1.7_x86-64 main contrib non-free
```

```
# Базовый репозиторий
deb https://dl.astralinux.ru/astra/stable/1.7_x86-64/repository-base/ 1.7_x86-64 main contrib non-free
```

```
# Расширенный репозиторий
deb https://dl.astralinux.ru/astra/stable/1.7_x86-64/repository-extended/ 1.7_x86-64 main contrib non-free
```

- 2) Обновим информацию о пакетах и установим *nginx*:

```
apt update
apt install nginx
```

Подготовить ключ (*/usr/share/crt/private.key*) и сертификат (*/usr/share/crt/cert.crt*) который будут использоваться для SSL шифрования (*https*).

2.2. Установка и развертывание сервиса «Администрирование и контроль доступа (Хаб)»

2.2.1. Подготовка БД

- 1) Подключиться по *ssh* к серверу, на котором развернута СУБД;
- 2) Выполнить команду *psql*;
- 3) Выполнить команду: *su postgres*
- 4) Выполнить команду *create database app.hub*;
- 5) Выполнить команду *create database app.logs*.

2.2.2. Развёртывание БД логов

Выполнить скрипт на *app.logs*

```
create extension "uuid-ossf";
```

```
create schema logs;
```

```
create table logs.auth
```

```
(
```

```

        id      uuid not null,
        login   text,
        client  text,
        is_success boolean not null,
        remote_ip text,
        local_ip text,
        date_time timestamp with time zone default now()
    );

```

```

create index auth_date_created
on logs.auth (date_time);

```

```

create table logs.access
(
    id      uuid          default uuid_generate_v4() not null,
    client  text,
    resource text,
    login   text,
    claim   text,
    role_id uuid          not null,
    organization_id uuid  not null,
    is_success boolean    not null,
    description text,
    date_time timestamp with time zone default now()
);

```

```

create index access_login_indexed
on logs.access (login);

```

```

create index access_date_created
on logs.access (date_time);

```

```

create function logs.add_auth_event(data text) returns void
language plpgsql
as
$$
DECLARE
    _json_data json = data::json;
    _user_name text = _json_data ->> 'UserName';
    _client_id text = _json_data ->> 'ClientId';
    _timestamp timestamp = (_json_data ->> 'TimeStamp')::timestamp;

```

```

        _is_success boolean = (_json_data ->> 'IsSuccess')::boolean;
        _local_ip text = _json_data ->> 'LocalIpAddress';
        _remote_ip text = _json_data ->> 'RemoteIpAddress';
BEGIN

        insert into logs.auth(id, login, client, is_success, remote_ip, local_ip,
date_time)
        select uuid_generate_v4(), _user_name, _client_id, _is_success,
_remote_ip, _local_ip, _timestamp;
    end
    $$;

create function logs.add_access_event(data text) returns void
    language plpgsql
as
    $$
DECLARE
        _json_data json = data::json;
        _user_name text = _json_data ->> 'UserName';
        _claim text = _json_data ->> 'Claim';
        _role_id uuid = (_json_data ->> 'RoleId')::uuid;
        _org_id uuid = (_json_data ->> 'OrganizationId')::uuid;
        _is_success boolean = (_json_data ->> 'IsSuccess')::boolean;
        _timestamp timestamp = (_json_data ->> 'TimeStamp')::timestamp with
time zone;
        _description text = _json_data ->> 'Description';
        _client text = _json_data ->> 'Client';
        _resource text = _json_data ->> 'Resource';
BEGIN

        insert into logs.access(id, client, resource, login, claim, role_id,
organization_id, is_success, description, date_time)
        select uuid_generate_v4(), _client, _resource, _user_name, _claim,
_role_id, _org_id, _is_success, _description, _timestamp;
    end
    $$;

create function logs.get_auth_events(take integer DEFAULT 50, skip integer
DEFAULT 0, sort text DEFAULT 'date_time'::text, dir text DEFAULT 'desc'::text,
where_clause text DEFAULT NULL::text) returns SETOF refcursor
    language plpgsql

```

```

as
$$
DECLARE
    _sql1      text;
    _sql2      text;
    _refcursor  refcursor = 'data_cursor';
    _refcursor_total refcursor = 'total_cursor';
BEGIN
    _sql1 := 'SELECT login, client, is_success, date_time FROM logs.auth '
        || where_clause;

    _sql2 := 'SELECT COUNT(1) AS total FROM (' || _sql1 || ') as sub';

    _sql1 := _sql1 || ' ORDER BY ' || sort || ' ' || dir
        || ' LIMIT ' || take
        || ' OFFSET ' || skip;

    OPEN _refcursor FOR
        EXECUTE _sql1;
    RETURN NEXT _refcursor;

    OPEN _refcursor_total FOR
        EXECUTE _sql2;
    RETURN NEXT _refcursor_total;

END;

$$;

```

create function logs.get_access_events(take integer DEFAULT 50, skip integer DEFAULT 0, sort text DEFAULT 'date_time'::text, dir text DEFAULT 'desc'::text, where_clause text DEFAULT NULL::text) returns SETOF refcursor

```

language plpgsql
as
$$
DECLARE
    _sql1 text;
    _sql2 text;
    _data_cursor refcursor = 'data_cursor';
    _total_cursor refcursor = 'total_cursor';

```

BEGIN

```
_sql1 := 'SELECT client, resource, login, claim, role_id, organization_id,
is_success, description, date_time
FROM logs.access ' || where_clause;

_sql2 := 'SELECT COUNT(1) AS total FROM (' || _sql1 || ') as sub';

_sql1 := _sql1 || ' ORDER BY ' || sort || ' ' || dir
        || ' LIMIT ' || take
        || ' OFFSET ' || skip;

OPEN _data_cursor FOR
    EXECUTE _sql1;
RETURN NEXT _data_cursor;

OPEN _total_cursor FOR
    EXECUTE _sql2;
RETURN NEXT _total_cursor;
end;
$$;
```

2.2.3. Настройка бэкенд-приложения

- 1) Скопировать дистрибутив сервиса в директорию сервера (например, */usr/share/hosting/hub*);
- 2) Создать пользователя, под которым будет запускаться приложение. Пользователь должен иметь полный доступ к директории с приложением и права открывать сокеты;
- 3) Перейти в директорию дистрибутива */clients/default*;
- 4) Выполнить настройку *appsettings.json*.

```
{
  "ConnectionStrings": {
    "DefaultConnection": "Server=localhost; Database=app.hub; User
ID=postgres; Password=postgres",
    "LogsConnection": "Server=localhost; Port=5433; Database=app.logs; User
ID=postgres; Password=postgres"
  },
  "ApplicationUrls": [
    "https://localhost:1900"
```

```

],
"CorsOrigins": [
  "^http[s]?:\\\\x\\.x\\.x(:\\d{1,6})?$",
  "^http[s]?:\\\\localhost(:\\d{1,6})?$"
],
"HostedServices": {
  "Items": [
    {
      "Key": "OrganizationReplication",
      "Enabled": false,
      "Interval": "00:05:00"
    },
    {
      "key": "GridStateCaching",
      "Enabled": false
    },
    {
      "key": "PersonalFileImport",
      "Enabled": false,
      "Interval": "00:05:00"
    }
  ]
},
"Logging": {
  "IncludeScopes": false,
  "LogLevel": {
    "Default": "Warning",
    "System": "Warning",
    "Microsoft": "Warning"
  }
},
"Customization": {
  "GridState": {
    "FallbackOnNoCache": false
  }
},
"Navigation": {
  "Origins": {
    "Auth": "https://x.x.x.x:порт" //Внешний адрес сервиса «Авторизация»
  }
},

```



```

"Workspace": {
  "OrgMode": "multi",
  "RoleMode": "multi"
},
"EventBus": {
  "Enabled": true,
  "Broker": "app.workspace",
  "RetryCount": 10,
  "QueueName": "app.hub",
  "CommandQueueName": "app.hub",
  "ExchangeType": "direct",
  "BusAccess": {
    "Host": "x.x.x.x", /ip-адрес RabbitMQ
    "UserName": "user", //логин
    "Password": "password", //пароль
    "RetryCount": 10
  }
},
"Smtп": {
  "Host": "", //имя хоста почтового сервера
  "Port": 465, //порт
  "IsSsl": false, //использоваться ли ssl при подключении
  "Email": "", //адрес с которого отправлять сообщение
  "UserName": "", //логин подключения к почте
  "Password": "", /пароль подключения к почте
  "CheckCertificateRevocation": false //проверять ли ssl сертификат на
актуальность
},
"Authentication": {
  "Authority": "https://x.x.x.x:порт" , //Внешний адрес сервиса Хаб
  "ApiName": "identityServer", //идентификатор ресурса (для запросов на
API используя JWT из SPA-приложения в составе сервиса Хаб)
  "ApiSecret": "auth", //Секретное слово
  "ClaimType": "uri://schemas.quarta.su/permission-claim-type",
  "Password": { //Настройка политик безопасности (в отношении
новых паролей), уже созданным учеткам будет разрешено войти с их
действующими паролями
    "RequireDigit": false,
    "RequiredLength": 1,
    "RequireNonAlphanumeric": false,
    "RequireUppercase": false,

```

```

    "RequireLowercase": false,
    "RequiredUniqueChars": 1
  },
  "AccountBlocking": {
    "MaxLoginAttempts": 99    // Максимальное количество неудачных
попыток входа (правильный логин, неправильный пароль)
  },
  "Identity": { //Настройка SPA-приложения. Указанные настройки
заливаются в БД при запуске приложения
    "ClientId": "admin",
    "ClientName": "admin",
    "Host": "https://х.х.х.х:порт", //публичный хост сервиса
«Авторизация»
    //Адреса редиректов куда разрешено вернуться после удачной
аутентификации
    //Следует менять только Хост.
    "CallbackUrl": [
      "https://х.х.х.х:порт/login-callback",
      "https://х.х.х.х:порт/silent-callback"
    ],
    //Адреса на которые разрешено вернуться после разлогинивания
    "PostLogoutUrl": "https://х.х.х.х:порт/login",
    "Scope": "openid profile identityServer"
  }
},
"FileStorage": {
  "UseLocalStorage": true
}
}

```

2.2.4. Настройка systemd-сервиса

- 1) Создать systemd-файл (например, */etc/systemd/system/hub.service*);
- 2) Ввести настройки запуска сервиса *dotnet* приложением с указанием параметров. Входной точкой приложения является *Quarta.Auth.Web.dll*.
- 3) Выполнить команду *systemctl daemon-reload*, чтобы ввести сервис в действие.
- 4) Проверить состояние сервиса командой *systemctl status hub*;

Пример настроек *hub.service*

```

[Unit]
Description=[Hub] Авторизация
[Service]
WorkingDirectory=/usr/share/hosting/hub
ExecStart=/usr/bin/dotnet /usr/share/hosting/hub/Quarta.Auth.Web.dll
Restart=always
RestartSec=10
SyslogIdentifier=dotnet-bi
User=%ИМЯ СИСТЕМНОГО ПОЛЬЗОВАТЕЛЯ%
Environment=DOTNET_PRINT_TELEMETRY_MESSAGE=false
Environment=DOTNET_CLI_TELEMETRY_OPTOUT=false
Environment=ASPNETCORE_ENVIRONMENT=Production
[Install]
WantedBy=multi-user.target

```

2.2.5. Настройка nginx

- 1) Перейти в директорию */etc/nginx/conf.d* (или аналогичную директорию для размещения динамических модулей *nginx*);
- 2) Создать файл *app-hub-service.conf*;
- 3) Внести настройки *proxy_pass* между внешним адресом сервера и внутренним;
- 4) Проверить корректность настроек командой *nginx -t*;
- 5) Выполнить команду *nginx reload* если проверка прошла успешно;

Пример настроек *app-hub-service.conf*

```

server {
    listen 1900 ssl;

    location / {
        proxy_pass https://127.0.0.1:1900;
    }
}

```

Адрес *proxy_pass* должен соответствовать значению поля *ApplicationUrl* указанного в *appSettings.json*.

2.2.6. Настойка SPA-приложения

- 1) Перейти в директорию дистрибутива `wwwroot/app/assets/configurations/clients/default`;
- 2) Открыть файл `configuration.json`;
- 3) Внести настройки SPA-приложения.

Пример настроек *configuration.json*

```
{
  "serverUrl": "https://х.х.х.х:порт", //публичный адрес сервиса Хаб
  "authentication": {
    "authority": "https://х.х.х.х:порт", //публичный адрес сервиса Хаб
    "login_uri": "https://х.х.х.х:порт/login",
    "redirect_uri": "https://х.х.х.х:порт/login-callback",
    "silent_redirect_uri": "https://х.х.х.х:порт/silent-callback",
    "post_logout_redirect_uri": "https://х.х.х.х:порт/login",
    "client_id": "admin",
    "response_type": "id_token token",
    "scope": "openid profile identityServer"
  },
  "navigation": {
    "api": "https://х.х.х.х:порт/api/navigation", //публичный адрес сервиса
    "domain": "Auth"
  },
  "filestorage": {
    "fileStorageUrl": "https://х.х.х.х:порт/api/file-storage/", //публичный
    "maxAllowedFileSize": 1073741824
  }
}
```

Настройки секции `authentication` должны соответствовать значениям из конфига бэкенд-приложения *Authentication:Identity*.

2.3. Установка и развертывание сервиса «Файловое хранилище»

2.3.1. Подготовка БД

- 1) Скачать файлы дистрибутива с сайта производителя ПО MongoDB:
<https://www.mongodb.com/try/download/community>
 - a) `mongodb-org-server_6.0.3_amd64.deb`

в) mongodb-database-tools-debian10-x86_64-100.6.1.deb

Разместить скаченные файлы в директории /home/user/download

2) Установить MongoDB, выполнив команды:

```
dpkg -i /home/user/download/mongodb-org-server_6.0.3_amd64.deb
```

```
dpkg -i /home/user/download/mongodb-mongosh_1.6.1_amd64.deb
```

```
dpkg -i /home/user/download/mongodb-database-tools-debian10-x86_64-100.6.1.deb
```

Запустить службу MongoDB, выполнив команды:

```
systemctl start mongod
```

```
systemctl enable mongod
```

3) Убедиться, что MongoDB слушает свой порт, выполнив команду:

```
netstat -tunlp | grep -i mongo
```

Если MongoDB слушает локальный порт 127.0.0.1:27027, то необходимо исправить файл /etc/mongod.conf.

```
bindIp: 127.0.0.1 -> bindIp: 0.0.0.0
```

после чего перезапустить сервис: systemctl restart mongod

4) Создать базу и **collection** в ней (пока что пустой), выполнив команды:

```
mongosh
```

```
use filestorage
```

```
db.files.insertOne( { x: 1 } )
```

В результате возможен запуск на **localhost** без пароля.

2.3.2. Настройка бэкенд-приложения

1) Скопировать дистрибутив сервиса в директорию сервера (например, /usr/share/hosting/fs)

2) Создать пользователя, под которым будет запускаться приложение. Пользователь должен иметь полный доступ к директории с приложением и права открывать сокеты;

3) Выполнить настройку **appsettings.json**.

```
{
  "ApplicationUrls": [
    "https://localhost:5040"
  ],
  "MongoDB": {
    "ConnectionString": "mongodb://x.x.x.x.:27017", //адрес где размещена
СУБД «MongoDB»
    "Catalog": "production",
```

```

    "DefaultCollection": "files"
  },
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft": "Warning",
      "Microsoft.Hosting.Lifetime": "Information"
    }
  },
  "AllowedHosts": "*"
}

```

2.3.3. Настройка systemd-сервиса

- 1) Создать systemd-файл (например, */etc/systemd/system/fs.service*);
- 2) Ввести настройки запуска сервиса *dotnet* приложением с указанием параметров. Входной точкой приложения является *Quarta.FileStorage.NetCore.Web.dll*.
- 3) Выполнить команду *systemctl daemon-reload*, чтобы ввести сервис в действие.
- 4) Проверить состояние сервиса командой *systemctl status fs*.

Пример настроек *fs.service*

```

[Unit]
Description=[fs] Файловое хранилище
[Service]
WorkingDirectory=/usr/share/hosting/fs
ExecStart=/usr/bin/dotnet
/usr/share/hosting/fs/Quarta.FileStorage.NetCore.Web.dll
Restart=always
RestartSec=10
SyslogIdentifier=dotnet-bi
User=%ИМЯ СИСТЕМНОГО ПОЛЬЗОВАТЕЛЯ%
Environment=DOTNET_PRINT_TELEMETRY_MESSAGE=false
Environment=DOTNET_CLI_TELEMETRY_OPTOUT=false
Environment=ASPNETCORE_ENVIRONMENT=Production
[Install]
WantedBy=multi-user.target

```

2.3.4. Настройка nginx

- 1) Перейти в директорию */etc/nginx/conf.d* (или аналогичную директорию для размещения динамических модулей *nginx*);
- 2) Создать файл *app-fs-service.conf*;
- 3) Внести настройки *proxy_pass* между внешним адресом сервера и внутренним;
- 4) Проверить корректность настроек командой *nginx -t*;
- 5) Выполнить команду *nginx reload*, если проверка прошла успешно.

Пример настроек *app-fs-service.conf*

```
server {  
    listen    1040 ssl;  
    location / {  
        proxy_pass https://127.0.0.1:5040;  
    }  
}
```

Адрес *proxy_pass* должен соответствовать значению поля *ApplicationUrl* указанного в *appSettings.json*.

2.4. Установка и развертывание сервиса «Справочники»

2.4.1. Подготовка БД

- 1) Подключиться по *ssh* к серверу, на котором развернута СУБД;
- 2) Выполнить команду: *su postgres*.
- 3) Выполнить команду *psql*.
- 4) Выполнить команду *create database app.classifier*.

2.4.2. Настройка бэкенд-приложения

- 1) Скопировать дистрибутив сервиса в директорию сервера (например, */usr/share/hosting/classifier*).
- 2) Создать пользователя, под которым будет запускаться приложение. Пользователь должен иметь полный доступ к директории с приложением и права открывать сокеты;
- 3) Перейти в директорию дистрибутива */clients/default*;
- 4) Выполнить настройку *appsettings.json*.

```

{
  "ApplicationUrls": [ "https://localhost:1300" ],
  "ConnectionStrings": {
    "DefaultConnection": "Server=localhost; Database=app.classifier; User
ID=postgres; Password=postgres"
  },
  "AuthServer": {
    "Url": "https://х.х.х.х:порт", //хост приложение сервиса Хаб. Может
указываться внешний или внутренний адрес
    "ApiName": "", //идентификатор данного сервиса (режим «Ресурсы»
сервиса Хаб)
    "ApiSecret": "", //пароль данного сервиса (режим «Ресурсы» сервиса
Хаб, код авторизации)
    "ClaimType": "uri://schemas.quarta.su/permission-claim-type",
    "AccessEventEndpointName": "access_event_endpoint"
  },
  "EventBus": {
    "Enabled": true,
    "Broker": "app.workspace",
    "RetryCount": 10,
    "QueueName": "app.classifier",
    "CommandQueueName": "app.classifier",
    "ExchangeType": "direct",
    "BusAccess": {
      "Host": "х.х.х.х", //ip-адрес RabbitMQ
      "UserName": "user", //логин
      "Password": "password", //пароль
      "RetryCount": 10
    }
  },
  "Logging": {
    "LogLevel": {
      "Default": "Warning",
      "System": "Warning",
      "Microsoft": "Warning"
    }
  },
  "FileStorage": {
    "Url": http://х.х.х.х:порт//адрес файлового хранилища.
  },
  "CorsOrigins": [

```



```

    "^http[s]?:\\\\x\\.x\\.x\\.x(:\\d{1,6})?$",
    "^http[s]?:\\\\localhost(:\\d{1,6})?$"
]
}

```

2.4.3. Настройка systemd-сервиса

- 1) Создать systemd-файл (например, /etc/systemd/system/classifier.service);
- 2) Ввести настройки запуска сервиса **dotnet** приложением с указанием параметров. Входной точкой приложения является **Quarta.Classifier.WebApi.dll**;
- 3) Выполнить команду **systemctl daemon-reload**, чтобы ввести сервис в действие.
- 4) Проверить состояние сервиса командой **systemctl status classifier**;

Пример настроек **classifier.service**

```

[Unit]
Description=Справочники

[Service]
WorkingDirectory=/usr/share/hosting/classifier
ExecStart=/usr/bin/dotnet Quarta.Classifier.WebApi.dll -skip-env-config -
migrations
Restart=always
RestartSec=10
SyslogIdentifier=dotnet-classifier
User=%ИМЯ СИСТЕМНОГО ПОЛЬЗОВАТЕЛЯ%
Environment=DOTNET_PRINT_TELEMETRY_MESSAGE=false
Environment=DOTNET_CLI_TELEMETRY_OPTOUT=false
Environment=ASPNETCORE_ENVIRONMENT=Production

[Install]
WantedBy=multi-user.target

```

2.4.4. Настройка nginx

- 1) Перейти в директорию **/etc/nginx/conf.d** (или аналогичную директорию для размещения динамических модулей **nginx**);
- 2) Создать файл **app-classifier-service.conf**;

- 3) Внести настройки `proxy_pass` между внешним адресом сервера и внутренним;
- 4) Проверить корректность настроек командой ***nginx -t***;
- 5) Выполнить команду ***nginx reload*** если проверка прошла успешно.

Пример настроек ***app-classifier-service.conf***

```
server {
    listen 1300 ssl;

    location / {
        proxy_pass https://127.0.0.1:1300;
    }
}
```

Адрес `proxy_pass` должен соответствовать значению поля `ApplicationUrl` указанного в ***appSettings.json***.

2.4.5. Настойка SPA-приложения

- 1) Перейти в директорию дистрибутива `quarta-classifier-web-app/config/clients/default`;
- 2) Открыть файл `configuration.json`;
- 3) Внести настройки SPA-приложения.

Пример настроек ***configuration.json***

```
{
  "modules": {
    "classifier": {
      "serverUrl": "https://х.х.х.х:порт", //внешний хост-адрес сервиса
      «Справочники»
      "baseUrl": "/app/classifier", //базовый url приложения
      "fileStorage": {
        "fileStorageUrl": "https://х.х.х.х:порт/api/file-storage-proxy/",
        "maxAllowedFileSize": 1073741824
      }
    }
  },
  "authentication": {
```

```

    "authority": "https://х.х.х.х:порт", //внешний хост адрес бэкенд-
приложения сервиса Хаб
    "redirect_uri": "https://х.х.х.х:порт/callback", //внешний адрес
переадресации
    "post_logout_redirect_uri": "https://х.х.х.х:порт/login", //внешний адрес
страницы логина внутри приложения
    "silent_redirect_uri": "https://х.х.х.х:порт/silent-callback", //внешний
адрес автообновления токена
    "client_id": "", //идентификатор клиента (Режим «Клиенты» в сервисе
Хаб)
    "response_type": "id_token token",
    "scope": " openid profile identityServer classifier", //перечень
доступных ресурсов.

    "useLocalStorage": false,
    "automaticSilentRenew": true
  }
}

```

2.5. Установка и развертывание сервиса «Уведомления»

2.5.1. Подготовка БД

- 1) Подключиться по *ssh* к серверу, на котором развернута СУБД;
- 2) Выполнить команду: *su postgres*.
- 3) Выполнить команду *psql*.
- 4) Выполнить команду *create database app.notifications*.

2.5.2. Настройка бэкенд-приложения

- 1) Скопировать дистрибутив сервиса в директорию сервера (например, */usr/share/hosting/notifications*);
- 2) Создать пользователя, под которым будет запускаться приложение. Пользователь должен иметь полный доступ к директории с приложением и права открывать сокеты;
- 3) Перейти в директорию дистрибутива */clients/default*;
- 4) Выполнить настройку *appsettings.json*.

```

{
  "ApplicationUrls": [
    "https://localhost:1400" //адрес который займет веб-сервер Kestrel
  ],

```

```

    "AuthServer": {
        "Url": "https://х.х.х.х", //хост приложение сервиса Хаб. Может
указываться внешний или внутренний адрес
        "ApiName": "", //идентификатор данного сервиса (режим «Ресурсы»
сервиса Хаб)
        "ApiSecret": "", //пароль данного сервиса (режим «Ресурсы» сервиса
Хаб, код авторизации)
        "ClaimType": "uri://schemas.quarta.su/permission-claim-type",
        "AccessEventEndpointName": "access_event_endpoint"
    },
    "ConnectionStrings": {
        "DefaultConnection": "Server=localhost; Database=app.notifications; User
ID=postgres; Password=postgres"
    },
    "EventBus": {
        "Enabled": true,
        "Broker": "app.workspace",
        "RetryCount": 10,
        "QueueName": "app.notifications",
        "CommandQueueName": "app.notifications",
        "ExchangeType": "direct",
        "BusAccess": {
            "Host": "х.х.х.х", //ИП адрес RabbitMQ
            "UserName": "user", //ЛОГИН
            "Password": "password", //пароль
            "RetryCount": 10
        }
    },
    "CorsOrigins": [
        "^http[s]?://\\x\\.x\\.x(\\d{1,6})?$",
        "^http[s]?://\\localhost(\\d{1,6})?$"
    ],
    "Logging": {
        "LogLevel": {
            "Default": "Warning",
            "System": "Warning",
            "Microsoft": "Warning"
        }
    },
    "Smtp": {
        "Host": "", //имя хоста почтового сервера

```

```

    "Port": 465, //порт
    "IsSsl": false, //использоваться ли ssl при подключении
    "Email": "", //адрес, с которого отправлять сообщение
    "UserName": "", //логин подключения к почте
    "Password": "", /пароль подключения к почте
    "CheckCertificateRevocation": false //проверять ли ssl сертификат на
актуальность
  },
  "RabbitUserId": "E34D0E27-4176-469F-A5D7-BC34BC922510"
}

```

2.5.3. Настройка systemd-сервиса

- 1) Создать systemd-файл (например, /etc/systemd/system/notification.service);
- 2) Ввести настройки запуска сервиса **dotnet** приложением с указанием параметров. Входной точкой приложения является **Quarta.Notification.WebApi.dll**;
- 3) Выполнить команду **systemctl daemon-reload**, чтобы ввести сервис в действие;
- 4) Проверить состояние сервиса командой **systemctl status notifications**.

Пример настроек **notification.service**

```

[Unit]
Description=Уведомления

[Service]
WorkingDirectory=/usr/share/hosting/notifications
ExecStart=/usr/bin/dotnet Quarta.Notification.WebApi.dll -skip-env-config -
migrations
Restart=always
RestartSec=10
SyslogIdentifier=dotnet-notification
User=%ИМЯ СИСТЕМНОГО ПОЛЬЗОВАТЕЛЯ%
Environment=DOTNET_PRINT_TELEMETRY_MESSAGE=false
Environment=DOTNET_CLI_TELEMETRY_OPTOUT=false
Environment=ASPNETCORE_ENVIRONMENT=Production

[Install]
WantedBy=multi-user.target

```

2.5.4. Настройка nginx

- 1) Перейти в директорию */etc/nginx/conf.d* (или аналогичную директорию для размещения динамических модулей *nginx*);
- 2) Создать файл *app-notification-service.conf*;
- 3) Внести настройки *proxy_pass* между внешним адресом сервера и внутренним;
- 4) Проверить корректность настроек командой *nginx -t*;
- 5) Выполнить команду *nginx reload* если проверка прошла успешно.

Пример настроек *app-notification-service.conf*

```
server {  
    listen 1400 ssl;  
  
    location / {  
        proxy_pass https://127.0.0.1:1400;  
    }  
}
```

Адрес *proxy_pass* должен соответствовать значению поля *ApplicationUrl* указанного в *appSettings.json*.

2.5.5. Настройка SPA-приложения

- 1) Перейти в директорию дистрибутива *quarta-notification-web-app/config/clients/default*;
- 2) Открыть файл *configuration.json*;
- 3) Внести настройки SPA-приложения.

Пример настроек *configuration.json*

```
{  
    "modules": {  
        "nt": {  
            "serverUrl": "https://х.х.х.х:порт", //внешний адрес бэкенд-приложения  
            «Уведомления»  
            "baseUrl": "/app/notification" //базовый url приложения  
        }  
    },  
    "authentication": {
```

```

    "authority": "https://х.х.х.х:порт", //внешний хост адрес бэкенд-
приложения сервиса Хаб
    "redirect_uri": "https://х.х.х.х:порт/callback", //внешний адрес
переадресации
    "post_logout_redirect_uri": "https://х.х.х.х:порт/login", //внешний адрес
страницы логина внутри приложения
    "silent_redirect_uri": "https://х.х.х.х:порт/silent-callback", //внешний
адрес автообновления токена
    "client_id": "", //идентификатор клиента (Режим «Клиенты» в сервисе
Хаб)
    "response_type": "id_token token",
    "scope": " openid profile identityServer notification", //перечень
доступных ресурсов.
    "useLocalStorage": false,
    "automaticSilentRenew": true
  }
}

```

2.6. Установка и развертывание сервиса «Распознавание речи»

2.6.1. Настройка бэкенд-приложения

- 1) Установить docker и docker-compose, выполнив команды:
apt install docker.io
apt install docker-compose
- 2) Скопировать дистрибутив сервиса в директорию сервера (например, */usr/share/hosting/speech2textrecognizer*)
- 3) Создать пользователя, под которым будет запускаться приложение. Пользователь должен иметь полный доступ к директории с приложением и права открывать сокеты;
- 4) В директории */usr/share/hosting/speech2textrecognizer/src* выполнить сборку docker образа, запустив команду `docker build -t audio2text`;
- 5) В директории */usr/share/hosting/speech2textrecognizer/* создать файл `docker-compose.yaml` с содержанием

```

version: '3'

services:
  api:
    container_name: "audio2text-recognizer-api"
    image: audio2text
    ports:
      - "8000:8000"

```

```
env_file:
- ./src/.env
volumes:
- ./models:/models
```

- 6) В директории */usr/share/hosting/speech2textrecognizer/models* разместить языковые модели для распознавания текста, модели можно получить по ссылке <https://alphacephei.com/vosk/models>;
- 7) Выполнить настройку окружения файл */usr/share/hosting/speech2textrecognizer/src/.env*.

```
LANG_MODEL=vosk-model-ru-0.22 (имя скаченной модели из п3)
RECOGNIZER=vosk
```

2.6.2. Настройка systemd-сервиса

- 1) Создать systemd-файл (например, */etc/systemd/system/speech2text.service*);
- 2) Ввести настройки запуска сервиса *python* в виде *docker* образа;
- 3) Выполнить команду *systemctl daemon-reload*, чтобы ввести сервис в действие;
- 4) Проверить состояние сервиса командой *systemctl status speech2text*.

Пример настроек *speech2text.service*

```
[Unit]
Description=Распознавание аудио
Requires=docker.service
After=docker.service
StartLimitIntervalSec=60

[Service]
WorkingDirectory=/usr/share/hosting/speech2textrecognizer
ExecStart=docker-compose up
ExecStop=docker-compose down
1) TimeoutStartSec=0
Restart=on-failure RestartSec=10
User=%ИМЯ СИСТЕМНОГО ПОЛЬЗОВАТЕЛЯ%

[Install]
WantedBy=multi-user.target
```


Пример настроек *configuration.json*

```
{
  "modules": {
    "pf": {
      "serverUrl": "https://localhost:5180",
      "baseUrl": "/app/pf",
      "fileStorage": {
        "fileStorageUrl": "https://localhost:5180/api/file-storage-proxy/",
        "maxAllowedFileSize": 1073741824
      }
    },
    "pm": {
      "serverUrl": "https://х.х.х.х:порт", //внешний адрес сервиса «Ведение
проектов»
      "remoteEntry": "https://х.х.х.х:порт/remoteEntry.js"
    },
    "ds": {
      "serverUrl": "https://х.х.х.х:порт", //внешний адрес сервиса «Хранение
и оборот документов»
      "remoteEntry": "https://х.х.х.х:порт/remoteEntry.js"
    },
    "pb": {
      "serverUrl": "https://х.х.х.х:порт" //внешний адрес сервиса
«Телефонный справочник»
    }
  },
  "authentication": {
    "authority": "https://х.х.х.х:порт", //внешний хост адрес бэкенд-
приложения «Хаб»
    "redirect_uri": "https://х.х.х.х:порт/callback", //внешний адрес
перееадресации
    "post_logout_redirect_uri": "https://х.х.х.х:порт/login", //внешний адрес
страницы логина внутри приложения
    "silent_redirect_uri": "https://х.х.х.х:порт/silent-callback", //внешний
адрес автообновления токена
    "client_id": "", //идентификатор клиента (Режим «Клиенты» в Хабе)
    "response_type": "id_token token",
    "scope": " openid profile identityServer pf", //перечень доступных
ресурсов.
    "useLocalStorage": false,
```

```

        "automaticSilentRenew": true
    }
}

```

2.7. Установка и развертывание сервиса «Хранение и оборот документов»

2.7.1. Подготовка БД

- 1) Подключиться по ssh к серверу, на котором развернута СУБД;
- 2) Выполнить команду: *su postgres*.
- 3) Выполнить команду *psql*;
- 4) Выполнить команду *create database app.ds*;

2.7.2. Настройка бэкенд-приложения

- 1) Скопировать дистрибутив сервиса в директорию сервера (например, */usr/share/hosting/ds*);
- 2) Создать пользователя, под которым будет запускаться приложение. Пользователь должен иметь полный доступ к директории с приложением и права открывать сокеты;
- 3) Перейти в директорию дистрибутива */clients/default*;
- 4) Выполнить настройку *appsettings.json*.

```

{
  "ApplicationUrls": [ "https://localhost:1500" ],

  "ConnectionStrings": {
    "DefaultConnection": "Server=localhost; Database=app.ds; User
ID=postgres; Password=postgres"
  },
  "AuthServer": {
    "Url": "https://х.х.х.х:порт", //хост приложение сервиса Хаб. Может
указываться внешний или внутренний адрес
    "ApiName": "", //идентификатор данного сервиса (режим «Ресурсы»
сервиса Хаб)
    "ApiSecret": "", //пароль данного сервиса (режим «Ресурсы» сервиса
Хаб, код авторизации)
    "ClaimType": "uri://schemas.quarta.su/permission-claim-type",
    "AccessEventEndpointName": "access_event_endpoint"
  },
  "EventBus": {

```

```

"Enabled": true,
"Broker": "app.workspace",
"RetryCount": 10,
"QueueName": "app.ds",
"CommandQueueName": "app.ds",
"ExchangeType": "direct",
"BusAccess": {
  "Host": "x.x.x.x", //ip-адрес RabbitMQ
  "UserName": "user", //логин
  "Password": "password", //пароль
  "RetryCount": 10
},
"Logging": {
  "LogLevel": {
    "Default": "Warning",
    "System": "Warning",
    "Microsoft": "Warning"
  }
},
"OnlyofficeUrl": "https://x.x.x.x:порт", //адрес веб-сервера OnlyOffice
"FileStorage": {
  "Url": http://x.x.x.x:порт//адрес файлового хранилища.
},
"CorsOrigins": [
  "^http[s]?://\\x\\.x\\.x\\.x(:\\d{1,7})?$",
  "^http[s]?://\\localhost(:\\d{1,7})?$"
],
"ApacheTikaHost": "http://x.x.x.x:порт", //адрес апатч тика
"FullTextSearchUpdate": { //настройки полнотекстового поиска
  "IsWorkrOn": true,
  "RecordsToProcess": 50,
  "ThreadCount": 10
}
}

```

2.7.3. Настройка systemd-сервиса

- 1) Создать systemd-файл (например, */etc/systemd/system/ds.service*);
- 2) Ввести настройки запуска сервиса **dotnet** приложением с указанием параметров. Входной точкой приложения является **Quarta.DS.WebApi.dll**;

- 3) Выполнить команду *systemctl daemon-reload*, чтобы ввести сервис в действие.
- 4) Проверить состояние сервиса командой *systemctl status ds*.

Пример настроек *ds.service*

```
[Unit]
Description=Файловое хранилище

[Service]
WorkingDirectory=/usr/share/hosting/ds
ExecStart=/usr/bin/dotnet Quarta.DS.WebApi.dll -skip-env-config -migrations
Restart=always
RestartSec=10
SyslogIdentifier=dotnet-ds
User=%ИМЯ СИСТЕМНОГО ПОЛЬЗОВАТЕЛЯ%
Environment=DOTNET_PRINT_TELEMETRY_MESSAGE=false
Environment=DOTNET_CLI_TELEMETRY_OPTOUT=false
Environment=ASPNETCORE_ENVIRONMENT=Production

[Install]
WantedBy=multi-user.target
```

2.7.4. Настройка *nginx*

- 1) Перейти в директорию */etc/nginx/conf.d* (или аналогичную директорию для размещения динамических модулей *nginx*);
- 2) Создать файл *app-ds-service.conf*;
- 3) Внести настройки *proxy_pass* между внешним адресом сервера и внутренним;
- 4) Проверить корректность настроек командой *nginx -t*;
- 5) Выполнить команду *nginx reload* если проверка прошла успешно;

```
Пример настроек app-ds-service.conf
server {
    listen 1500 ssl;

    location / {
        proxy_pass https://127.0.0.1:1500;
    }
}
```

```
}
```

Адрес `проху_pass` должен соответствовать значению поля `ApplicationUrl` указанного в ***appSettings.json***.

2.7.5. Настойка SPA-приложения

- 1) Перейти в директорию дистрибутива `quarta-ds-web-app/config/clients/default`;
- 2) Открыть файл `configuration.json`;
- 3) Внести настройки SPA-приложения.

Пример настроек ***configuration.json***

```
{
  "modules": {
    "ds": {
      "serverUrl": "https://х.х.х.х:порт", //внешний адрес бэкенд-
приложения «Оборот и хранение документов»
      "baseUrl": "/app/ds", //базовый url приложения
      "fileStorage": {
        "fileStorageUrl": "https://х.х.х.х:порт/api/file-storage-proxy/",
        "maxAllowedFileSize": 1073741824
      },
      "options": { //настройки приложения
        "fileLibrary": {
          "tabs": ["allFiles"],
          "cardTabs": ["main", "approvals", "history"],
          "buttons": {
            "addFile": false,
            "addFolder": true
          }
        }
      },
      "classifier": {
        "serverUrl": https://х.х.х.х:порт//внешний адрес приложения
«Справочники»
      },
      "onlyoffice": {
```

```

        "serverUrl": "https://х.х.х.х:порт//внешний адрес сервиса Only
Office.
    },
    "authentication": {
        "authority": "https://х.х.х.х:порт", //внешний хост адрес бэкенд-
приложения сервиса Хаб
        "redirect_uri": "https://х.х.х.х:порт/callback", //внешний адрес
переадресации
        "post_logout_redirect_uri": "https://х.х.х.х:порт/login", //внешний адрес
страницы логина внутри приложения
        "silent_redirect_uri": "https://х.х.х.х:порт/silent-callback", //внешний
адрес автообновления токена
        "client_id": "", //идентификатор клиента (Режим «Клиенты» в сервисе
Хаб)
        "response_type": "id_token token",
        "scope": " openid profile identityServer ds classifier", //перечень
доступных ресурсов.
        "useLocalStorage": false,
        "automaticSilentRenew": true
    }
}

```

2.8. Установка и развертывание сервиса «Штатная расстановка»

2.8.1. Подготовка БД

- 1) Подключиться по *ssh* к серверу, на котором развернута СУБД;
- 2) Выполнить команду: *su postgres*.
- 3) Выполнить команду *psql*;
- 4) Выполнить команду *create database app.ss*.

2.8.2. Настройка бэкенд-приложения

- 1) Скопировать дистрибутив сервиса в директорию сервера (например, */usr/share/hosting/ss*);
- 2) Создать пользователя, под которым будет запускаться приложение. Пользователь должен иметь полный доступ к директории с приложением и права открывать сокеты;
- 3) Перейти в директорию дистрибутива */clients/default*;
- 4) Выполнить настройку *appsettings.json*.

```

{
  "ApplicationUrls": [
    http://localhost:1200 //адрес который займет веб-сервер Kestrel
  ],
  "AuthServer": {
    "Url": "https://х.х.х.х:порт", //хост приложение Хаб. Может указываться
    внешний или внутренний адрес
    "ApiName": "", //идентификатор данного сервиса (режим «Ресурсы»
    сервиса Хаб)
    "ApiSecret": "", //пароль данного сервиса (режим «Ресурсы» сервиса
    Хаб, код авторизации)
    "ClaimType": "uri://schemas.quarta.su/permission-claim-type",
    "AccessEventEndpointName": "access_event_endpoint"
  },
  "ConnectionStrings": {
    "DefaultConnection": "Server=localhost; Database=app.ss; User
    ID=postgres; Password=postgres"
  },
  "EventBus": {
    "Enabled": true,
    "Broker": "app.workspace",
    "RetryCount": 10,
    "QueueName": "app.ss",
    "CommandQueueName": "app.ss",
    "ExchangeType": "direct",
    "BusAccess": {
      "Host": "х.х.х.х", //ip-адрес RabbitMQ
      "UserName": "user", //логин
      "Password": "password", //пароль
      "RetryCount": 10
    }
  },
  "CorsOrigins": [
    "^http[s]?://\\|/x\\.x\\.x\\.x(:\\d{1,7})?$",
    "^http[s]?://\\|/localhost(:\\d{1,7})?$"
  ],
  "Logging": {
    "LogLevel": {
      "Default": "Warning",
      "System": "Warning",
      "Microsoft": "Warning"
    }
  }
}

```

```

    }
  },
  "FileStorage": {
    "Url": http://х.х.х.х:порт//адрес файлового хранилища.
  }
}

```

2.8.3. Настройка systemd-сервиса

- 1) Создать systemd-файл (например, */etc/systemd/system/ss.service*);
- 2) Ввести настройки запуска сервиса *dotnet* приложением с указанием параметров. Входной точкой приложения является *Quarta.SS.WebApi.dll*;
- 3) Выполнить команду *systemctl daemon-reload*, чтобы ввести сервис в действие.
- 4) Проверить состояние сервиса командой *systemctl status ss*.

Пример настроек *ss.service*

```

[Unit]
Description=Штатная расстановка

[Service]
WorkingDirectory=/usr/share/hosting/ss
ExecStart=/usr/bin/dotnet Quarta.SS.WebApi.dll -skip-env-config -migrations
Restart=always
RestartSec=10
SyslogIdentifier=dotnet-ss
User=%ИМЯ СИСТЕМНОГО ПОЛЬЗОВАТЕЛЯ%
Environment=DOTNET_PRINT_TELEMETRY_MESSAGE=false
Environment=DOTNET_CLI_TELEMETRY_OPTOUT=false
Environment=ASPNETCORE_ENVIRONMENT=Production

[Install]
WantedBy=multi-user.target

```

2.8.4. Настройка nginx

- 1) Перейти в директорию */etc/nginx/conf.d* (или аналогичную директорию для размещения динамических модулей *nginx*);
- 2) Создать файл *app-ss-service.conf*;

- 3) Внести настройки `proxy_pass` между внешним адресом сервера и внутренним;
- 4) Проверить корректность настроек командой ***nginx -t***;
- 5) Выполнить команду ***nginx reload*** если проверка прошла успешно.

Пример настроек ***app-ss-service.conf***

```
server {  
    listen 1200 ssl;  
  
    location / {  
        proxy_pass https://127.0.0.1:1200;  
    }  
}
```

Адрес `proxy_pass` должен соответствовать значению поля `ApplicationUrl` указанного в ***appSettings.json***.

2.9. Установка и развертывание сервиса «Профиль пользователя»

2.9.1. Подготовка БД

- 1) Подключиться по `ssh` к серверу, на котором развернута СУБД;
- 2) Выполнить команду: ***su postgres***.
- 3) Выполнить команду ***psql***;
- 4) Выполнить команду `create database app.profile`.

2.9.2. Настройка бэкенд-приложения

- 1) Скопировать дистрибутив сервиса в директорию сервера (например, ***/usr/share/hosting/profile***)
- 2) Создать пользователя, под которым будет запускаться приложение. Пользователь должен иметь полный доступ к директории с приложением и права открывать сокеты;
- 3) Перейти в директорию дистрибутива ***/clients/default***;
- 4) Выполнить настройку ***appsettings.json***.

```
{  
  "ApplicationUrls": [  
    "https://localhost:5180" //адрес который займет веб-сервер Kestrel  
  ],  
}
```



```
}  
}
```

2.9.3. Настройка systemd-сервиса

- 1) Создать systemd-файл (например, /etc/systemd/system/profile.service);
- 2) Ввести настройки запуска сервиса *dotnet* приложением с указанием параметров. Входной точкой приложения является *Quarta.Profile.WebApi.dll*;
- 3) Выполнить команду *systemctl daemon-reload*, чтобы ввести сервис в действие.
- 4) Проверить состояние сервиса командой *systemctl status profile*.

Пример настроек *profile.service*

```
[Unit]  
Description=Профиль пользователя  
  
[Service]  
WorkingDirectory=/usr/share/hosting/profile  
ExecStart=/usr/bin/dotnet Quarta.Profile.WebApi.dll -skip-env-config -  
migrations  
Restart=always  
RestartSec=10  
SyslogIdentifier=dotnet-profile  
User=%ИМЯ СИСТЕМНОГО ПОЛЬЗОВАТЕЛЯ%  
Environment=DOTNET_PRINT_TELEMETRY_MESSAGE=false  
Environment=DOTNET_CLI_TELEMETRY_OPTOUT=false  
Environment=ASPNETCORE_ENVIRONMENT=Production  
  
[Install]  
WantedBy=multi-user.target
```

2.9.4. Настройка nginx

- 1) Перейти в директорию */etc/nginx/conf.d* (или аналогичную директорию для размещения динамических модулей *nginx*);
- 2) Создать файл *app-profile-service.conf*;
- 3) Внести настройки *proxy_pass* между внешним адресом сервера и внутренним;
- 4) Проверить корректность настроек командой *nginx -t*;
- 5) Выполнить команду *nginx reload*, если проверка прошла успешно;

Пример настроек *app-profile-service.conf*

```
server {  
    listen 5180 ssl;  
  
    location / {  
        proxy_pass https://127.0.0.1:5180;  
    }  
}
```

Адрес `proxy_pass` должен соответствовать значению поля `ApplicationUrl` указанного в *appSettings.json*

2.9.5. Настойка SPA-приложения

- 1) Перейти в директорию дистрибутива `quarta-profile-web-app/config/clients/default`;
- 2) Открыть файл `configuration.json`;
- 3) Внести настройки SPA-приложения.

Пример настроек *configuration.json*

```
{  
    "modules": {  
        "pf": {  
            "serverUrl": "https://localhost:5180",  
            "baseUrl": "/app/pf",  
            "fileStorage": {  
                "fileStorageUrl": "https://localhost:5180/api/file-storage-proxy/",  
                "maxAllowedFileSize": 1073741824  
            }  
        },  
        "pm": {  
            "serverUrl": "https://х.х.х.х:порт", //внешний адрес сервиса «Ведение  
проектов»  
            "remoteEntry": "https://х.х.х.х:порт/remoteEntry.js"  
        },  
        "ds": {
```

```

        "serverUrl": "https://х.х.х.х:порт", //внешний адрес сервиса «Хранение
и оборот документов»
        "remoteEntry": "https://х.х.х.х:порт/remoteEntry.js"
    },
    "pb": {
        "serverUrl": "https://х.х.х.х:порт" //внешний адрес сервиса
«Телефонный справочник»
    }
}, "authentication": {
    "authority": "https://х.х.х.х:порт", //внешний хост адрес бэкенд-
приложения «Хаб»
    "redirect_uri": "https://х.х.х.х:порт/callback", //внешний адрес
переадресации
    "post_logout_redirect_uri": "https://х.х.х.х:порт/login", //внешний адрес
страницы логина внутри приложения
    "silent_redirect_uri": "https://х.х.х.х:порт/silent-callback", //внешний
адрес автообновления токена
    "client_id": "", //идентификатор клиента (Режим «Клиенты» в Хабе)
    "response_type": "id_token token",
    "scope": "openid profile identityServer pf", //перечень доступных
ресурсов.
    "useLocalStorage": false,
    "automaticSilentRenew": true
}
}

```

2.10. Установка и развертывание сервиса «Личные заметки»

2.10.1. Подготовка БД

- 1) Подключиться по ssh к серверу, на котором развернута СУБД;
- 2) Выполнить команду: *su postgres*.
- 3) Выполнить команду *psql*;
- 4) Выполнить команду *create database app.notes*;

2.10.2. Настройка бэкенд-приложения

- 1) Скопировать дистрибутив сервиса в директорию сервера (например, */usr/share/hosting/notes*);
- 2) Создать пользователя, под которым будет запускаться приложение. Пользователь должен иметь полный доступ к директории с приложением и права открывать сокеты;

- 3) Перейти в директорию дистрибутива */clients/default*;
- 4) Выполнить настройку *appsettings.json*.

```
{
  "ApplicationUrls": [
    "https://localhost:5140" //адрес который займет веб-сервер Kestrel
  ],
  "AuthServer": {
    "Url": "https://х.х.х.х:порт", //хост приложение Хаб. Может указываться
    //внешний или внутренний адрес
    "ApiName": "", //идентификатор данного сервиса (режим «Ресурсы»
    //Хаба)
    "ApiSecret": "", //пароль данного сервиса (режим «Ресурсы» Хаба, код
    //авторизации)
    "ClaimType": "uri://schemas.quarta.su/permission-claim-type",
    "AccessEventEndpointName": "access_event_endpoint"
  },
  "ConnectionStrings": {
    "DefaultConnection": "Server=localhost; Database=app.notes; User
    ID=postgres; Password=postgres"
  },
  "EventBus": {
    "Enabled": true,
    "Broker": "app.workspace",
    "RetryCount": 10,
    "QueueName": "app.notes",
    "CommandQueueName": "app.notes_cmd",
    "ExchangeType": "direct",
    "BusAccess": {
      "Host": "х.х.х.х", //ip-адрес RabbitMQ
      "UserName": "user", //логин
      "Password": "password", //пароль
      "RetryCount": 10
    }
  },
  "CorsOrigins": [
    "^http[s]?://\\/[x\\.x\\.x\\.x](\\:\\d{1,6})?$",
    "^http[s]?://\\/localhost(\\:\\d{1,6})?$"
  ],
  "Logging": {
    "LogLevel": {
```

```

        "Default": "Warning",
        "System": "Warning",
        "Microsoft": "Warning"
    }
},
"FileStorage": {
    "Url": https://х.х.х.х:порт – адрес Хранилища Файлов
}
}

```

2.10.3. Настройка systemd-сервиса

- 1) Создать systemd-файл (например, */etc/systemd/system/notes.service*);
- 2) Ввести настройки запуска сервиса *dotnet* приложением с указанием параметров. Входной точкой приложения является *Quarta.Note.WebApi.dll*;
- 3) Выполнить команду *systemctl daemon-reload*, чтобы ввести сервис в действие.
- 4) Проверить состояние сервиса командой *systemctl status notes*;

Пример настроек *notes.service*

```

[Unit]
Description=Заметки

[Service]
WorkingDirectory=/usr/share/hosting/notes
ExecStart=/usr/bin/dotnet Quarta.Note.WebApi.dll -skip-env-config -
migrations
Restart=always
RestartSec=10
SyslogIdentifier=dotnet-notes
User=%ИМЯ СИСТЕМНОГО ПОЛЬЗОВАТЕЛЯ%
Environment=DOTNET_PRINT_TELEMETRY_MESSAGE=false
Environment=DOTNET_CLI_TELEMETRY_OPTOUT=false
Environment=ASPNETCORE_ENVIRONMENT=Production

[Install]
WantedBy=multi-user.target

```

2.10.4. Настройка nginx

- 1) Перейти в директорию */etc/nginx/conf.d* (или аналогичную директорию для размещения динамических модулей *nginx*);
- 2) Создать файл *app-notes-service.conf*;
- 3) Внести настройки *proxy_pass* между внешним адресом сервера и внутренним;
- 4) Проверить корректность настроек командой *nginx -t*;
- 5) Выполнить команду *nginx reload*, если проверка прошла успешно;

Пример настроек *app-notes-service.conf*

```
server {  
    listen 5140 ssl;  
  
    location / {  
        proxy_pass https://127.0.0.1:5140;  
    }  
}
```

Адрес *proxy_pass* должен соответствовать значению поля *ApplicationUrl*, указанного в *appSettings.json*.

2.10.5. Настройка SPA-приложения

- 1) Перейти в директорию дистрибутива *quarta-notes-web-app/config/clients/default*;
- 2) Открыть файл *configuration.json*;
- 3) Внести настройки SPA-приложения;

Пример настроек *configuration.json*

```
{  
    "modules": {  
        "notes": {  
            "serverUrl": "https://localhost:5140",  
            "baseUrl": "/app/notes"  
        }  
    },  
    "authentication": {
```



```

    "authority": "https://х.х.х.х:порт", //внешний хост адрес бэкенд-
приложения «Хаб»
    "redirect_uri": "https://х.х.х.х:порт/callback", //внешний адрес
переадресации
    "post_logout_redirect_uri": "https://х.х.х.х:порт/login", //внешний адрес
страницы логина внутри приложения
    "silent_redirect_uri": "https://х.х.х.х:порт/silent-callback", //внешний
адрес автообновления токена
    "client_id": "", //идентификатор клиента (Режим «Клиенты» в Хабе)
    "response_type": "id_token token",
    "scope": "openid profile identityServer notes", //перечень доступных
ресурсов.
    "useLocalStorage": false,
    "automaticSilentRenew": true
  }
}

```

2.11. Установка и развертывание сервиса «Управление проектами и задачами»

2.11.1. Подготовка БД

- 1) Подключиться по ssh к серверу, на котором развернута СУБД;
- 2) Выполнить команду: *su postgres*.
- 3) Выполнить команду *psql*;
- 4) Выполнить команду *create database app.pm*.

2.11.2. Настройка бэкенд-приложения

- 1) Скопировать дистрибутив сервиса в директорию сервера (например, */usr/share/hosting/pm*);
- 2) Создать пользователя, под которым будет запускаться приложение. Пользователь должен иметь полный доступ к директории с приложением и права открывать сокеты;
- 3) Перейти в директорию дистрибутива */clients/default*;
- 4) Выполнить настройку *appsettings.json*.

```

{
  "ApplicationUrls": [
    "https://localhost:5140" //адрес который займет веб-сервер Kestrel
  ],
  "AuthServer": {

```



```
"Notification": {
  "enabled": true
}
}
```

2.11.3. Настройка systemd-сервиса

- 1) Создать systemd-файл (например, */etc/systemd/system/pm.service*);
- 2) Ввести настройки запуска сервиса *dotnet* приложением с указанием параметров. Входной точкой приложения является *Quarta.PM.WebApi.dll*.
- 3) Выполнить команду *systemctl daemon-reload*, чтобы ввести сервис в действие.
- 4) Проверить состояние сервиса командой *systemctl status pm*.

Пример настроек *pm.service*

```
[Unit]
Description=Управление проектами и задачами

[Service]
WorkingDirectory=/usr/share/hosting/pm
ExecStart=/usr/bin/dotnet Quarta.PM.WebApi.dll -skip-env-config -
migrations
Restart=always
RestartSec=10
SyslogIdentifier=dotnet-pm
User=%ИМЯ СИСТЕМНОГО ПОЛЬЗОВАТЕЛЯ%
Environment=DOTNET_PRINT_TELEMETRY_MESSAGE=false
Environment=DOTNET_CLI_TELEMETRY_OPTOUT=false
Environment=ASPNETCORE_ENVIRONMENT=Production

[Install]
WantedBy=multi-user.target
```

2.11.4. Настройка nginx

- 1) Перейти в директорию */etc/nginx/conf.d* (или аналогичную директорию для размещения динамических модулей *nginx*);
- 2) Создать файл *app-pm-service.conf*;

- 3) Внести настройки `proxy_pass` между внешним адресом сервера и внутренним;
- 4) Проверить корректность настроек командой ***nginx -t***;
- 5) Выполнить команду ***nginx reload***, если проверка прошла успешно;

Пример настроек ***app-pm-service.conf***

```
server {  
    listen 5110 ssl;  
  
    location / {  
        proxy_pass https://127.0.0.1:5110;  
    }  
}
```

Адрес `proxy_pass` должен соответствовать значению поля `ApplicationUrl` указанного в ***appSettings.json***.

2.11.5. Настойка SPA-приложения

- 1) Перейти в директорию дистрибутива `quarta-pm-web-app/config/clients/default`;
- 2) Открыть файл `configuration.json`;
- 3) Внести настройки SPA-приложения.

Пример настроек ***configuration.json***

```
{  
    "modules": {  
        "pm": {  
            "serverUrl": "https://localhost:5110",  
            "baseUrl": "/app/pm"  
        }  
    },  
    "authentication": {  
        "authority": "https://х.х.х.х:порт", //внешний хост адрес бэкенд-приложения “Хаб”  
        "redirect_uri": "https://х.х.х.х:порт/callback", //внешний адрес переадресации
```

```

        "post_logout_redirect_uri": "https://х.х.х.х:порт/login", //внешний адрес
страницы логина внутри приложения
        "silent_redirect_uri": "https://х.х.х.х:порт/silent-callback", //внешний
адрес автообновления токена
        "client_id": "", //идентификатор клиента (Режим «Клиенты» в Хабе)
        "response_type": "id_token token",
        "scope": " openid profile identityServer pm", //перечень доступных
ресурсов.
        "useLocalStorage": false,
        "automaticSilentRenew": true
    }
}

```

2.12. Установка и развертывание сервиса «Адресная книга»

2.12.1. Подготовка БД

- 1) Подключиться по ssh к серверу, на котором развернута СУБД;
- 2) Выполнить команду: *su postgres*.
- 3) Выполнить команду *psql*;
- 4) Выполнить команду *create database app.pb*.

2.12.2. Настройка бэкенд-приложения

- 1) Скопировать дистрибутив сервиса в директорию сервера (например, */usr/share/hosting/pb*);
- 2) Создать пользователя, под которым будет запускаться приложение. Пользователь должен иметь полный доступ к директории с приложением и права открывать сокеты;
- 3) Перейти в директорию дистрибутива */clients/default*;
- 4) Выполнить настройку *appsettings.json*.

```

{
  "ApplicationUrls": [
    "https://localhost:5140" //адрес который займет веб-сервер Kestrel
  ],
  "AuthServer": {
    "Url": "https://х.х.х.х:порт", //хост приложение Хаб. Может указываться
внешний или внутренний адрес
    "ApiName": "", //идентификатор данного сервиса (режим «Ресурсы»
Хаба)
  }
}

```

```

        "ApiSecret": "", //пароль данного сервиса (режим «Ресурсы» Хаба, код
авторизации)
        "ClaimType": "uri://schemas.quarta.su/permission-claim-type",
        "AccessEventEndpointName": "access_event_endpoint"
    },
    "ConnectionStrings": {
        "DefaultConnection": "Server=localhost; Database=app.pb; User
ID=postgres; Password=postgres"
    },
    "EventBus": {
        "Enabled": true,
        "Broker": "app.workspace",
        "RetryCount": 10,
        "QueueName": "app.pb",
        "CommandQueueName": "app.pb_cmd",
        "ExchangeType": "direct",
        "BusAccess": {
            "Host": "x.x.x.x", //ip-адрес RabbitMQ
            "UserName": "user", //ЛОГИН
            "Password": "password", //пароль
            "RetryCount": 10
        }
    },
    "CorsOrigins": [
        "^http[s]?://\\/[x\\.x\\.x\\.x](:\\d{1,6})?$",
        "^http[s]?://\\/localhost(:\\d{1,6})?$"
    ],
    "Logging": {
        "LogLevel": {
            "Default": "Warning",
            "System": "Warning",
            "Microsoft": "Warning"
        }
    },
    "FileStorage": {
        "Url": https://x.x.x.x:порт – адрес Хранилища Файлов
    }
}

```

2.12.3. Настройка systemd-сервиса

- 1) Создать systemd-файл (например, */etc/systemd/system/pb.service*);

- 2) Ввести настройки запуска сервиса *dotnet* приложением с указанием параметров. Входной точкой приложения является *Quarta.PB.WebApi.dll*;
- 3) Выполнить команду *systemctl daemon-reload*, чтобы ввести сервис в действие.
- 4) Проверить состояние сервиса командой *systemctl status pb*.

Пример настроек *pb.service*

```
[Unit]
Description=Адресная книга

[Service]
WorkingDirectory=/usr/share/hosting/pb
ExecStart=/usr/bin/dotnet Quarta.PB.WebApi.dll -skip-env-config -migrations
Restart=always
RestartSec=10
SyslogIdentifier=dotnet-pb
User=%ИМЯ СИСТЕМНОГО ПОЛЬЗОВАТЕЛЯ%
Environment=DOTNET_PRINT_TELEMETRY_MESSAGE=false
Environment=DOTNET_CLI_TELEMETRY_OPTOUT=false
Environment=ASPNETCORE_ENVIRONMENT=Production

[Install]
WantedBy=multi-user.target
```

2.12.4. Настройка nginx

- 1) Перейти в директорию */etc/nginx/conf.d* (или аналогичную директорию для размещения динамических модулей *nginx*);
- 2) Создать файл *app-pb-service.conf*;
- 3) Внести настройки *proxy_pass* между внешним адресом сервера и внутренним;
- 4) Проверить корректность настроек командой *nginx -t*;
- 5) Выполнить команду *nginx reload*, если проверка прошла успешно;

Пример настроек *app-pb-service.conf*

```
server {
    listen 5130 ssl;
```

```

    location / {
        proxy_pass https://127.0.0.1:5130;
    }
}

```

Адрес `proxy_pass` должен соответствовать значению поля `ApplicationUrl`, указанного в ***appSettings.json***.

2.12.5. Настойка SPA-приложения

- 1) Перейти в директорию дистрибутива `quarta-pb-web-app/config/clients/default`;
- 2) Открыть файл `configuration.json`;
- 3) Внести настройки SPA-приложения;

Пример настроек ***configuration.json***

```

{
  "modules": {
    "pb": {
      "serverUrl": "https://localhost:5130",
      "baseUrl": "/app/pb"
    }
  },
  "authentication": {
    "authority": "https://х.х.х.х:порт", //внешний хост адрес бэкенд-
приложения «Хаб»
    "redirect_uri": "https://х.х.х.х:порт/callback", //внешний адрес
перееадресации
    "post_logout_redirect_uri": "https://х.х.х.х:порт/login", //внешний адрес
страницы логина внутри приложения
    "silent_redirect_uri": "https://х.х.х.х:порт/silent-callback", //внешний
адрес автообновления токена
    "client_id": "", //идентификатор клиента (Режим «Клиенты» в Хабе)
    "response_type": "id_token token",
    "scope": " openid profile identityServer pb", //перечень доступных
ресурсов.
    "useLocalStorage": false,
    "automaticSilentRenew": true
  }
}

```


2.13. Установка и развертывание сервиса «Табель учета рабочего времени»

2.13.1. Подготовка БД

- 1) Подключиться по ssh к серверу, на котором развернута СУБД;
- 2) Выполнить команду: *su postgres*.
- 3) Выполнить команду *psql*;
- 4) Выполнить команду *create database app.timesheet*.

2.13.2. Настройка бэкенд-приложения

- 1) Скопировать дистрибутив сервиса в директорию сервера (например, */usr/share/hosting/timesheet*);
- 2) Создать пользователя, под которым будет запускаться приложение. Пользователь должен иметь полный доступ к директории с приложением и права открывать сокеты;
- 3) Перейти в директорию дистрибутива */clients/default*;
- 4) Выполнить настройку *appsettings.json*.

```
{
  "ApplicationUrls": [
    "https://localhost:5150" //адрес который займет веб-сервер Kestrel
  ],
  "AuthServer": {
    "Url": "https://х.х.х.х:порт", //хост приложение Хаб. Может указываться
    //внешний или внутренний адрес
    "ApiName": "", //идентификатор данного сервиса (режим «Ресурсы»
    //Хаба)
    "ApiSecret": "", //пароль данного сервиса (режим «Ресурсы» Хаба, код
    //авторизации)
    "ClaimType": "uri://schemas.quarta.su/permission-claim-type",
    "AccessEventEndpointName": "access_event_endpoint"
  },
  "ConnectionStrings": {
    "DefaultConnection": "Server=localhost; Database=app.timesheet; User
    ID=postgres; Password=postgres"
  },
  "EventBus": {
    "Enabled": true,
```

```

"Broker": "app.workspace",
"RetryCount": 10,
"QueueName": "app.timesheet",
"CommandQueueName": "app.timesheet_cmd",
"ExchangeType": "direct",
"BusAccess": {
  "Host": "x.x.x.x", //ip-адрес RabbitMQ
  "UserName": "user", //логин
  "Password": "password", //пароль
  "RetryCount": 10
},
},
"CorsOrigins": [
  "^http[s]?://\\|/x\\.x\\.x\\.x(:\\d{1,6})?$",
  "^http[s]?://\\|/localhost(:\\d{1,6})?$"
],
"Logging": {
  "LogLevel": {
    "Default": "Warning",
    "System": "Warning",
    "Microsoft": "Warning"
  }
},
"FileStorage": {
  "Url": "https://x.x.x.x:порт – адрес Хранилища Файлов"
}
}

```

2.13.3. Настройка systemd-сервиса

- 1) Создать systemd-файл (например, /etc/systemd/system/timesheet.service);
- 2) Ввести настройки запуска сервиса *dotnet* приложением с указанием параметров. Входной точкой приложения является *Quarta.Timesheet.WebApi.dll*;
- 3) Выполнить команду *systemctl daemon-reload*, чтобы ввести сервис в действие;
- 4) Проверить состояние сервиса командой *systemctl status timesheet*.

Пример настроек *timesheet.service*

[Unit]

Description=Табель учета рабочего времени

[Service]

WorkingDirectory=/usr/share/hosting/timesheet

ExecStart=/usr/bin/dotnet Quarta.Timesheet.WebApi.dll -skip-env-config -
migrations

Restart=always

RestartSec=10

SyslogIdentifier=dotnet-timesheet

User=%ИМЯ СИСТЕМНОГО ПОЛЬЗОВАТЕЛЯ%

Environment=DOTNET_PRINT_TELEMETRY_MESSAGE=false

Environment=DOTNET_CLI_TELEMETRY_OPTOUT=false

Environment=ASPNETCORE_ENVIRONMENT=Production

[Install]

WantedBy=multi-user.target

2.13.4. Настройка nginx

- 1) Перейти в директорию */etc/nginx/conf.d* (или аналогичную директорию для размещения динамических модулей *nginx*);
- 2) Создать файл *app-timesheet-service.conf*;
- 3) Внести настройки *proxy_pass* между внешним адресом сервера и внутренним;
- 4) Проверить корректность настроек командой *nginx -t*;
- 5) Выполнить команду *nginx reload*, если проверка прошла успешно.

Пример настроек *app-timesheet-service.conf*

```
server {  
    listen 5150 ssl;  
  
    location / {  
        proxy_pass https://127.0.0.1:5150;  
    }  
}
```

Адрес *proxy_pass* должен соответствовать значению поля *ApplicationUrl* указанного в *appSettings.json*.

2.13.5. Настойка SPA-приложения

- 1) Перейти в директорию дистрибутива quarta-timesheet-web-app/config/clients/default;
- 2) Открыть файл configuration.json;
- 3) Внести настройки SPA-приложения;

Пример настроек *configuration.json*

```
{
  "modules": {
    "timesheet": {
      "serverUrl": "https://localhost:5150",
      "baseUrl": "/app/timesheet",
      "fileStorage": {
        "fileStorageUrl": "https://localhost:5150/api/file-storage-proxy/",
        "maxAllowedFileSize": 1073741824
      }
    }
  },
  "authentication": {
    "authority": "https://х.х.х.х:порт", //внешний хост адрес бэкенд-
    приложения «Хаб»
    "redirect_uri": "https://х.х.х.х:порт/callback", //внешний адрес
    переадресации
    "post_logout_redirect_uri": "https://х.х.х.х:порт/login", //внешний адрес
    страницы логина внутри приложения
    "silent_redirect_uri": "https://х.х.х.х:порт/silent-callback", //внешний
    адрес автообновления токена
    "client_id": "", //идентификатор клиента (Режим «Клиенты» в Хабе)
    "response_type": "id_token token",
    "scope": " openid profile identityServer timesheet", //перечень
    доступных ресурсов.
    "useLocalStorage": false,
    "automaticSilentRenew": true
  }
}
```

2.14. Установка и развертывание сервиса «Коммуникации»

Для установки и развертывания сервиса необходимы:

- 1) python 3.10+;
- 2) docker.

2.14.1. Подготовка БД

- 1) Подключиться по ssh к серверу, на котором развернута СУБД;
- 2) Выполнить команду: *su postgres*.
- 3) Выполнить команду *psql*;
- 4) Выполнить команду *create database app.chats*.

2.14.2. Настройка бэкенд-приложения

Создать образ сервера:

- настроить *config.ini* в корне папки **server**;
- собрать образ *docker build -f server/Dockerfile -t chat-server-dev:latest*.

Необходимо подготовить *docker-compose.yml*

```
version: '3'

services:
  redis-dev:
    container_name: "redis-dev"
    image: redis:5
    restart: always
    networks:
      - back-dev

  server-dev:
    container_name: "server-dev"
    image: chat-server-dev
    restart: always
    networks:
      - back-dev
    ports:
      - "5190:443"
    command: "daphne -e ssl:443:privateKey=key.pem:certKey=crt.pem -b
0.0.0.0 -p 8001 server.asgi:application" # команда для запуска с ssl
    env_file: # env_file если необходимо
      - ./server_dev.env
    volumes: # здесь можно подключить свой config.ini
      - ./key/crt.pem:/code/crt.pem # подключение сертификата в контейнер
      - ./key/key.pem:/code/key.pem # подключение ключа в контейнер
      - ./media:/code/media/ # сохранение media
```

- ./models/:/code/stenography/models # передача моделей для распознавания

environment: # определить переменные виртуального окружения

-

QUARTA_HUB_OPENID_CONFIGURATION=https://x.x.x.x:порт/.well-known/openid-configuration #внешний хост адрес бэкенд-приложения “Хаб»

- QUARTA_HUB_SCOPE=identityServer openid sophie profile pb

- QUARTA_HUB_CLIENT_ID=chats

- QUARTA_HUB_CLIENT_SECRET=chats

- MEDIA_URL=/assets/messages/media/

worker-dev-1: # Рабочий для обработки фоновых задач

container_name: "worker-dev-1"

image: chat-server-dev

restart: always

networks:

- back-dev

command: "python -m celery -A server worker -l INFO -E -X recognize"

env_file:

- ./server_dev.env

volumes:

- ./media/:/code/media/

- ./models/:/code/stenography/models/

environment:

-

QUARTA_HUB_OPENID_CONFIGURATION=https://x.x.x.x:порт/.well-known/openid-configuration #внешний хост адрес бэкенд-приложения “Хаб»

- QUARTA_HUB_SCOPE=identityServer openid sophie profile pb

- QUARTA_HUB_CLIENT_ID=chats

- QUARTA_HUB_CLIENT_SECRET=chats

- MEDIA_URL=/assets/messages/media/

worker-dev-2:

container_name: "worker-dev-2"

image: chat-server-dev

restart: always

networks:

- back-dev

command: "python -m celery -A server worker -l INFO -E -Q recognize -c

1"

env_file:

- ./server_dev.env

volumes:

- ./media:/code/media/
- ./models:/code/stenography/models/

environment:

-

QUARTA_HUB_OPENID_CONFIGURATION=https://x.x.x.x:порт/.well-known/openid-configuration #внешний хост адрес бэкенд-приложения “Хаб»

- QUARTA_HUB_SCOPE=identityServer openid sophie profile pb
- QUARTA_HUB_CLIENT_ID=chats
- QUARTA_HUB_CLIENT_SECRET=chats
- MEDIA_URL=/assets/messages/media/

worker-dev-hub: # Рабочий для обработки шины RabbitMQ

container_name: "worker-dev-hub"

image: chat-server-dev

restart: always

networks:

- back-dev

command: "python manage.py handlerdatabus"

env_file:

- ./server_dev.env

volumes:

- ./media:/code/media/
- ./models:/code/stenography/models/

environment:

-

QUARTA_HUB_OPENID_CONFIGURATION=https://x.x.x.x:порт/.well-known/openid-configuration #внешний хост адрес бэкенд-приложения “Хаб»

- QUARTA_HUB_SCOPE=identityServer openid sophie profile pb
- QUARTA_HUB_CLIENT_ID=chats
- QUARTA_HUB_CLIENT_SECRET=chats
- MEDIA_URL=/assets/messages/media/

flower:

image: mher/flower

restart: always

environment:

- CELERY_BROKER_URL=redis://redis-dev:6379/0
- FLOWER_PORT=8888

ports:

- 8888:8888

networks:

- back-dev

networks:

back-dev:

driver: bridge

Запустить докер-контейнеры из директории с файлом `docker-compose.yml` командой:

`docker-compose up -d`

2.14.3. Настройка `nginx`

- 1) Перейти в директорию `/etc/nginx/conf.d` (или аналогичную директорию для размещения динамических модулей `nginx`);
- 2) Создать файл `app-message.conf`;
- 3) Внести настройки `proxy_pass` между внешним адресом сервера и внутренним;
- 4) Проверить корректность настроек командой `nginx -t`;
- 5) Выполнить команду `nginx reload`, если проверка прошла успешно.

Пример настроек `app-message.conf`

```
server {  
    listen 3190 ssl;  
    error_page 497 https://$host:3190$request_uri;  
    location /assets/messages/media {  
        expires 7d;  
        proxy_pass https:// х.х.х.х:порт; # ip-адрес и порт сервера  
        приложения;  
    }  
  
    location /ws/ {  
        proxy_http_version 1.1;  
        proxy_set_header Upgrade $http_upgrade;  
        proxy_set_header Connection "upgrade";  
        proxy_redirect off;
```



```

        proxy_pass https:// х.х.х.х:порт; # ip-адрес и порт сервера
приложения;
    }

    location / {
        add_header 'Access-Control-Allow-Origin' '*';
        root /home/midto/hosting/messages;
        index index.html;
        add_header Last-Modified $date_gmt;
        add_header Cache-Control 'no-store, no-cache, must-revalidate, proxy-
revalidate, max-age=0';
        try_files $uri $uri/ /index.html;
        if_modified_since off;
        etag off;
    }

    location ~* ^/(api|admin|speech_file) { # настройки для доступа к бэку
        # add_header 'Access-Control-Allow-Origin' '*';
        add_header 'Access-Control-Allow-Methods' 'GET, POST, OPTIONS';
        client_max_body_size 0;
        proxy_pass https:// х.х.х.х:порт; # ip-адрес и порт сервера
приложения;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header REMOTE_ADDR $remote_addr;
        proxy_set_header Host $host;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        if_modified_since off;
        etag off;
    }

    location /app/messages/storage/ { # настройки файлового хранилища
        expires 30d;
        add_header Pragma "public";
        add_header Cache-Control "public";
        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_set_header X-Forwarded-Host $server_name;
        proxy_redirect off;
        rewrite app/messages/storage/(.*) /$1 break;
    }

```

```

    проху_pass https:// х.х.х.х:порт; # адрес файлового хранилища;
  }
}

```

2.14.4. Установка и настройка openvidu

Требования к машине:

- Docker;
- Docker-compose;
- Порты:
 - 80 TCP или любой другой для HTTP соединения;
 - 443 TCP или любой другой для HTTPS соединения;
 - 3478 TCP+UDP используется сервером STUN/TURN для разрешения IP-адресов клиентов;
 - 40000 – 57000 TCP+UDP используется Kurento Media Server для установления медиа-соединений;
 - 57001 – 65535 TCP+UDP используется сервером TURN для установления ретранслируемых медиа-соединений.

Установка:

- 1) Для установки сервера желательное использовать */opt. cd /opt*;
- 2) `curl https://s3-eu-west-1.amazonaws.com/aws.openvidu.io/install_openvidu_latest.sh | bash` — скачивается последняя версия сервера и разворачиваются файлы в директории `opt/openvidu`;
- 3) Необходимо зайти в директорию ***cd openvidu***;
- 4) Настройки хранятся в файле *.env*. Основные (полное описание):
 - а. DOMAIN_OR_PUBLIC_IP — домен или IP, по которому будут подключаться к серверу;
 - б. OPENVIDU_SECRET — секретная фраза, необходима для авторизации подключения;
 - в. CERTIFICATE_TYPE — тип сертификата, `selfsigned` — самоподписанный генерирует сервер, `owncert` — сертификат пользователя, `letsencrypt` — генерация сертификата через `letsencrypt`;
 - г. HTTPS_PORT — порт для HTTPS;
 - д. HTTP_PORT — порт для HTTP;
 - е. OPENVIDU_RECORDING — включена ли запись конференций;
 - ж. OPENVIDU_RECORDING_PATH — путь для хранения записей;

- з. OPENVIDU_RECORDING_PUBLIC_ACCESS — доступ к записям через API необходим **true**;
 - и. OPENVIDU_WEBHOOK — вебхуки о различных событиях, необходим **true**;
 - к. OPENVIDU_WEBHOOK_ENDPOINT — сервер для вебхуков, пример: ***https://x.x.x.x:порт/webhook/***, где ***x.x.x.x*** — ip сервера приложения;
- 5) Для запуска `./openvidu start`.

2.15. Установка и развертывание сервиса «Интерактивные рабочие столы и аналитика»

2.15.1. Подготовка БД

- 1) Подключиться по *ssh* к серверу, на котором развернута СУБД;
- 2) Выполнить команду *su postgres*.
- 3) Выполнить команду *psql*;
- 4) Выполнить команду `create database app.bi`.

2.15.2. Настройка бэкенд-приложения

- 1) Скопировать дистрибутив сервиса в директорию сервера (например, */usr/share/hosting/bi*);
- 2) Создать пользователя, под которым будет запускаться приложение. Пользователь должен иметь полный доступ к директории с приложением и права открывать сокеты;
- 3) Перейти в директорию дистрибутива */clients/default*;
- 4) Выполнить настройку *appsettings.json*.

```
{
  "Logging": {
    "LogLevel": { //Уровни логирования
      "Default": "Warning",
    }
  },
  "ConnectionStrings": { //Строка подключения к БД
    "DefaultConnection": "Server=localhost; Database=app.bi; User
ID=postgres; Password=postgres"
  },
  "ApplicationUrls": [ //Хост занимаемый веб-сервером Kestrel
    "https://localhost:1100"
```

```

    ],
    "CorsOrigins": [ // Настройка CROSS-ORIGIN-RESOURCE-SHARING:
Перечень адресов, которым разрешено обращаться к API приложения.
        "^http[s]?://\\/[x\\.x\\.x](:\\d{1,6})?$",
        "^http[s]?://\\/localhost(:\\d{1,6})?$"
    ]
}

```

2.15.3. Настройка systemd-сервиса

- 1) Создать systemd-файл (например, */etc/systemd/system/bi.service*);
- 2) Ввести настройки запуска сервиса *dotnet* приложением с указанием параметров. Входной точкой приложения является *Quarta.Bi.WebApi.dll*;
- 3) Выполнить команду *systemctl daemon-reload*, чтобы ввести сервис в действие.
- 4) Проверить состояние сервиса командой *systemctl status bi*.

Пример настроек *bi.service*

```

[Unit]
Description=[BI] Интерактивные рабочие столы и аналитика
[Service]
WorkingDirectory=/usr/share/hosting/bi
ExecStart=/usr/bin/dotnet /usr/share/hosting/bi/Quarta.BI.WebApi.dll
Restart=always
RestartSec=10
SyslogIdentifier=dotnet-bi
User=%ИМЯ СИСТЕМНОГО ПОЛЬЗОВАТЕЛЯ%
Environment=DOTNET_PRINT_TELEMETRY_MESSAGE=false
Environment=DOTNET_CLI_TELEMETRY_OPTOUT=false
Environment=ASPNETCORE_ENVIRONMENT=Production
[Install]
WantedBy=multi-user.target

```

2.15.4. Настройка nginx

- 1) Перейти в директорию */etc/nginx/conf.d* (или аналогичную директорию для размещения динамических модулей *nginx*);
- 2) Создать файл *app-bi-service.conf*;
- 3) Внести настройки *proxy_pass* между внешним адресом сервера и внутренним;

- 4) Проверить корректность настроек командой *nginx -t*;
- 5) Выполнить команду *nginx reload*, если проверка прошла успешно.

Пример настроек *app-bi-service.conf*

```
server {
    listen 1100 ssl;

    location / {
        proxy_pass https://127.0.0.1:1100;
    }
}
```

Адрес `proxy_pass` должен соответствовать значению поля `ApplicationUrl` указанного в *appSettings.json*.

2.15.5. Настойка SPA-приложения

- 1) Перейти в директорию дистрибутива `quarta-bi-web-app/config/clients/default`;
- 2) Открыть файл `configuration.json`;
- 3) Внести настройки SPA-приложения.

Пример настроек *configuration.json*

```
{
  "modules": {
    "bi": {
      "serverUrl": "https://х.х.х.х:порт", //внешний адрес бэкенд-приложения
      «Интерактивные рабочие столы и аналитика»
      "baseUrl": "/app/bi" //базовый url приложения
    },
    "ds": {
      "serverUrl": "https://х.х.х.х:порт", //внешний адрес-бэкенд-
      приложения сервиса «Файловое хранилище»
      "remoteEntry": "https:// х.х.х.х:порт/remoteEntry.js" //адрес, из которого
      загружать приложение.
    },
  },
  "authentication": {
```

```

        "authority": "https://х.х.х.х:порт", //внешний хост адрес бэкенд-
приложения Хаб
        "redirect_uri": "https://х.х.х.х:порт/callback", //внешний адрес
переадресации
        "post_logout_redirect_uri": "https://х.х.х.х:порт/login", //внешний адрес
страницы логина внутри приложения
        "silent_redirect_uri": "https://х.х.х.х:порт/silent-callback", //внешний
адрес автообновления токена
        "client_id": "", //идентификатор клиента (Режим «Клиенты» в сервисе
Хаб)
        "response_type": "id_token token",
        "scope": "openid profile bi identityServer " //перечень доступных
ресурсов.
        "useLocalStorage": false,
        "automaticSilentRenew": true
    }
}

```

2.16. Установка и развертывание сервиса «Оболочка»

2.16.1. Развертывание дистрибутива

- 1) Скопировать дистрибутив сервиса в директорию сервера (например, */usr/share/hosting/ws*);
- 2) Создать пользователя, под которым будет запускаться приложение. Пользователь должен иметь полный доступ к директории с приложением;
- 3) Перейти в директорию дистрибутива */clients/default*.

2.16.2. Настройка nginx

- 1) Перейти в директорию */etc/nginx/conf.d* (или аналогичную директорию для размещения динамических модулей *nginx*);
- 2) Создать файл *app-ws-service.conf*;
- 3) Внести настройки *proxy_pass* между внешним адресом сервера и внутренним;
- 4) Проверить корректность настроек командой *nginx -t*;
- 5) Выполнить команду *nginx reload*, если проверка прошла успешно.

Пример настроек *app-ws-service.conf*

(указанные хосты для *proxy_pass* ссылаются на внутренние адреса сервисов (может также использоваться публичный адрес))

```

listen 80 ssl;
    error_page 497 https://$host:80$request_uri;
    root /usr/share/hosting/ws;
    location / {
        try_files $uri $uri/ /index.html;
    }
    location /assets/bi {
        proxy_pass https://localhost:5100;
    }
    location /assets/pm {
        proxy_pass https://localhost:5110;
    }
    location /assets/ds {
        proxy_pass https://localhost:5120;
    }
    location /assets/pb {
        proxy_pass https://localhost:5130;
    }
    location /assets/ts {
        proxy_pass https://localhost:5150;
    }
    location /assets/ss {
        proxy_pass https://localhost:5170;
    }
    location /assets/pf {
        proxy_pass https://localhost:5180;
    }
    location /assets/messages {
        proxy_pass https://localhost:5190;
    }
    location /app/messages/storage/ {
        rewrite app/messages/storage/(.*) /$1 break;
        proxy_pass https://x.x.x.x:1040;
    }
    location /assets/sn {
        proxy_pass https://localhost:3230;
    }
}

```

2.16.3. Настройка SPA-приложения

- 1) Перейти в директорию дистрибутива *quarta-ws-web-app/config/*;

- 2) Открыть файл `configuration.json`;
- 3) Внести настройки SPA-приложения.

Пример настроек ***configuration.json***

```
{
  "modules": {
    "bi": {
      "serverUrl": "https://x.x.x.x:3100",
      "path": "bi",
      "remoteEntry": "https://x.x.x.x:3100/remoteEntry.js",
      "exposedModule": "./FeaturesModule",
      "key": "FeaturesModule",
      "name": "BI"
    },
    "pm": {
      "serverUrl": "https://x.x.x.x:3110",
      "path": "pm",
      "remoteEntry": "https://x.x.x.x:3110/remoteEntry.js",
      "exposedModule": "./FeaturesModule",
      "key": "FeaturesModule",
      "name": "ProjectManagement"
    },
    "ds": {
      "serverUrl": "https://x.x.x.x:3120",
      "path": "ds",
      "remoteEntry": "https://x.x.x.x:3120/remoteEntry.js",
      "exposedModule": "./FeaturesModule",
      "key": "FeaturesModule",
      "name": "Documentation",
      "fileStorage": {
        "fileStorageUrl": "https://localhost:5120/api/file-storage-proxy/",
        "maxAllowedFileSize": 1073741824
      }
    },
    "pb": {
      "serverUrl": "https://x.x.x.x:3130",
      "path": "pb",
      "remoteEntry": "https://x.x.x.x:3130/remoteEntry.js",
      "exposedModule": "./FeaturesModule",
```



```

    "key": "FeaturesModule",
    "name": "Phonebook"
  },
  "notes": {
    "serverUrl": "https://x.x.x.x:3140",
    "path": "notes",
    "remoteEntry": "https://x.x.x.x:3140/remoteEntry.js",
    "exposedModule": "./FeaturesModule",
    "key": "FeaturesModule",
    "name": "Notes"
  },
  "timesheet": {
    "serverUrl": "https://x.x.x.x:3150",
    "path": "timesheet",
    "remoteEntry": "https://x.x.x.x:3150/remoteEntry.js",
    "exposedModule": "./FeaturesModule",
    "key": "FeaturesModule",
    "name": "Timesheet"
  },
  "nt": {
    "serverUrl": "https://localhost:5160",
    "path": "notifications",
    "remoteEntry": "http://localhost:3160/remoteEntry.js",
    "exposedModule": "./FeaturesModule",
    "key": "FeaturesModule",
    "name": "Notifications"
  },
  "messages": {
    "serverUrl": "https://x.x.x.x:3190",
    "path": "messages",
    "remoteEntry": "https://x.x.x.x:3190/remoteEntry.js",
    "exposedModule": "./FeaturesModule",
    "key": "FeaturesModule",
    "name": "messages"
  },
  "pf": {
    "serverUrl": "https://x.x.x.x:3180",
    "path": "pf",
    "remoteEntry": "https://x.x.x.x:3180/remoteEntry.js",
    "exposedModule": "./FeaturesModule",
    "key": "FeaturesModule",

```

```

        "name": "Profile"
    },
    "classifier": {
        "serverUrl": "https://x.x.x.x:3210"
    },
    "onlyoffice": {
        "serverUrl": "https://x.x.x.x"
    }
},
"authentication": {
    "authority": "https://x.x.x.x:порт", //внешний хост адрес бэкенд-
приложения «Хаб»
    "redirect_uri": "https://x.x.x.x:порт/callback", //внешний адрес
переадресации приложения
    "post_logout_redirect_uri": "https://x.x.x.x:порт/login", //внешний адрес
страницы логина внутри приложения
    "silent_redirect_uri": "https://x.x.x.x:порт/silent-callback", //внешний
адрес автообновления токена
    "client_id": "", //идентификатор клиента (Режим «Клиенты» в Хабе)
    "response_type": "id_token token",
    "scope": "openid identityServer ds ss bi notification classifier
integration", //перечень доступных ресурсов (указывается в «Хабе», режим
«Ресурсы»
    "useLocalStorage": false,
    "automaticSilentRenew": true
}
}

```

3. ОБНОВЛЕНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

3.1. Обновление сервиса «Администрирование и контроль доступа (Хаб)»

- 1) Распаковать дистрибутив с обновлением во временную папку.
- 2) Перенести существующие настройки (значения) из старого конфигурационного файла в новый.
 - Конфигурационный файл бэкэнда сервиса находится в директории ./Clients/Default/appsettings.json
 - Конфигурационный файл фронтэнда сервиса находится в директории
./wwwroot/app/assets/configurations/clients/default/configuration.js
on
- 3) Удалить директорию с сервисом, на место удаленной директории перенести директорию с настроенным обновлением.
- 4) Выдать разрешение на папку для пользователя, под которым настроена служба. Пользователь должен быть владельцем папки с правами на чтение / запись / выполнение.
- 5) Перезапустить службу.

3.2. Обновление сервиса «Файловое хранилище»

- 1) Распаковать дистрибутив с обновлением во временную папку.
- 2) Перенести существующие настройки (значения) из старого конфигурационного файла в новый.
 - Конфигурационный файл бэкэнда сервиса находится в директории ./appsettings.json
- 3) Удалить директорию с сервисом, на место удаленной директории перенести директорию с настроенным обновлением.
- 4) Выдать разрешение на папку для пользователя, под которым настроена служба. Пользователь должен быть владельцем папки с правами на чтение / запись / выполнение.
- 5) Перезапустить службу.

3.3. Обновление сервиса «Справочники»

- 1) Распаковать дистрибутив с обновлением во временную папку.
- 2) Перенести существующие настройки (значения) из старого конфигурационного файла в новый.

- Конфигурационный файл бэкэнда сервиса находится в директории `./Clients/Default/appsettings.json`
 - Конфигурационный файл фронтэнда сервиса находится в директории `./quarta-classifier-web-app/config/clients/default/configuration.json`
- 3) Удалить директорию с сервисом, на место удаленной директории перенести директорию с настроенным обновлением.
 - 4) Выдать разрешение на папку для пользователя, под которым настроена служба. Пользователь должен быть владельцем папки с правами на чтение / запись / выполнение.
 - 5) Перезапустить службу.

3.4. Обновление сервиса «Уведомления»

- 1) Распаковать дистрибутив с обновлением во временную папку.
- 2) Перенести существующие настройки (значения) из старого конфигурационного файла в новый.
 - Конфигурационный файл бэкэнда сервиса находится в директории `./Clients/Default/appsettings.json`
 - Конфигурационный файл фронтэнда сервиса находится в директории `./quarta-notification-web-app/config/clients/default/configuration.json`
- 3) Удалить директорию с сервисом, на место удаленной директории перенести директорию с настроенным обновлением.
- 4) Выдать разрешение на папку для пользователя, под которым настроена служба. Пользователь должен быть владельцем папки с правами на чтение / запись / выполнение.
- 5) Перезапустить службу.

3.5. Обновление сервиса «Распознавание речи»

- 1) Распаковать дистрибутив с обновлением во временную папку.
- 2) Провести сборку образа ***docker командой*** `docker build -t audio2text`.
- 3) Перезапустить службу `speech2text` командой:
`Systemctl restart speech2text`

3.6. Обновление сервиса «Хранение и оборот документов»

- 1) Распаковать дистрибутив с обновлением во временную папку.

2) Перенести существующие настройки (значения) из старого конфигурационного файла в новый.

- Конфигурационный файл бэкэнда сервиса находится в директории `./Clients/Default/appsettings.json`
- Конфигурационный файл фронтэнда сервиса находится в директории `./quarta-ds-web-app/config/clients/default/configuration.json`

3) Удалить директорию с сервисом, на место удаленной директории перенести директорию с настроенным обновлением.

4) Выдать разрешение на папку для пользователя, под которым настроена служба. Пользователь должен быть владельцем папки с правами на чтение / запись / выполнение.

5) Перезапустить службу.

3.7. Обновление сервиса «Штатная расстановка»

1) Распаковать дистрибутив с обновлением во временную папку.

2) Перенести существующие настройки (значения) из старого конфигурационного файла в новый.

- Конфигурационный файл бэкэнда сервиса находится в директории `./Clients/Default/appsettings.json`

3) Удалить директорию с сервисом, на место удаленной директории перенести директорию с настроенным обновлением.

4) Выдать разрешение на папку для пользователя, под которым настроена служба. Пользователь должен быть владельцем папки с правами на чтение / запись / выполнение.

5) Перезапустить службу.

3.8. Обновление сервиса «Профиль пользователя»

1) Распаковать дистрибутив с обновлением во временную папку.

2) Перенести существующие настройки (значения) из старого конфигурационного файла в новый.

- Конфигурационный файл бэкэнда сервиса находится в директории `./Clients/Default/appsettings.json`
- Конфигурационный файл фронтэнда сервиса находится в директории `./quarta-profile-web-app/config/clients/default/configuration.json`

3) Удалить директорию с сервисом, на место удаленной директории перенести директорию с настроенным обновлением.

4) Выдать разрешение на папку для пользователя, под которым настроена служба. Пользователь должен быть владельцем папки с правами на чтение / запись / выполнение.

5) Перезапустить службу.

3.9. Обновление сервиса «Личные заметки»

1) Распаковать дистрибутив с обновлением во временную папку.

2) Перенести существующие настройки (значения) из старого конфигурационного файла в новый.

- Конфигурационный файл бэкэнда сервиса находится в директории `./Clients/Default/appsettings.json`

- Конфигурационный файл фронтэнда сервиса находится в директории `./quarta-notes-web-app/config/clients/default/configuration.json`

3) Удалить директорию с сервисом, на место удаленной директории перенести директорию с настроенным обновлением.

4) Выдать разрешение на папку для пользователя, под которым настроена служба. Пользователь должен быть владельцем папки с правами на чтение / запись / выполнение.

5) Перезапустить службу.

3.10. Обновление сервиса «Управление проектами и задачами»

1) Распаковать дистрибутив с обновлением во временную папку.

2) Перенести существующие настройки (значения) из старого конфигурационного файла в новый.

- Конфигурационный файл бэкэнда сервиса находится в директории `./Clients/Default/appsettings.json`

- Конфигурационный файл фронтэнда сервиса находится в директории `./quarta-pm-web-app/config/clients/default/configuration.json`

3) Удалить директорию с сервисом, на место удаленной директории перенести директорию с настроенным обновлением.

4) Выдать разрешение на папку для пользователя, под которым настроена служба. Пользователь должен быть владельцем папки с правами на чтение / запись / выполнение.

- 5) Перезапустить службу.

3.11. Обновление сервиса «Адресная книга»

- 1) Распаковать дистрибутив с обновлением во временную папку.
- 2) Перенести существующие настройки (значения) из старого конфигурационного файла в новый.
 - Конфигурационный файл бэкэнда сервиса находится в директории `./Clients/Default/appsettings.json`
 - Конфигурационный файл фронтэнда сервиса находится в директории `./quarta-pb-web-app/config/clients/default/configuration.json`
- 3) Удалить директорию с сервисом, на место удаленной директории перенести директорию с настроенным обновлением.
- 4) Выдать разрешение на папку для пользователя, под которым настроена служба. Пользователь должен быть владельцем папки с правами на чтение / запись / выполнение.
- 5) Перезапустить службу.

3.12. Обновление сервиса «Табель учета рабочего времени»

- 1) Распаковать дистрибутив с обновлением во временную папку.
- 2) Перенести существующие настройки (значения) из старого конфигурационного файла в новый.
 - Конфигурационный файл бэкэнда сервиса находится в директории `./Clients/Default/appsettings.json`
 - Конфигурационный файл фронтэнда сервиса находится в директории `./wwwroot/config/clients/default/configuration.json`
- 3) Удалить директорию с сервисом, на место удаленной директории перенести директорию с настроенным обновлением.
- 4) Выдать разрешение на папку для пользователя, под которым настроена служба. Пользователь должен быть владельцем папки с правами на чтение / запись / выполнение.
- 5) Перезапустить службу.

3.13. Обновление сервиса «Коммуникации»

- 1) Распаковать дистрибутив с обновлением во временную папку.

2) Провести сборку образа ***docker build -f server/Dockerfile -t chat-server-dev:latest.***

3) Перезапустить сервис из директории с файлом docker-compose.yml командами:

`docker-compose down`

`docker-compose up -d`

3.14. Обновление сервиса «Интерактивные рабочие столы и аналитика»

1. Распаковать дистрибутив с обновлением во временную папку.
2. Перенести существующие настройки (значения) из старого конфигурационного файла в новый.
 - Конфигурационный файл бэкэнда сервиса находится в директории `./Clients/Default/appsettings.json`
 - Конфигурационный файл фронтэнда сервиса находится в директории `./quarta-bi-web-app/config/clients/default/configuration.json`
3. Удалить директорию с сервисом, на место удаленной директории перенести директорию с настроенным обновлением.
4. Выдать разрешение на папку для пользователя, под которым настроена служба. Пользователь должен быть владельцем папки с правами на чтение / запись / выполнение.
5. Перезапустить службу.

3.15. Обновление сервиса «Оболочка»

1. Распаковать дистрибутив с обновлением во временную папку.
2. Перенести существующие настройки (значения) из старого конфигурационного файла в новый.
 - Конфигурационный файл фронтэнда сервиса находится в директории `./config/configuration.json`
3. Удалить директорию с сервисом, на место удаленной директории перенести директорию с настроенным обновлением.
4. Выдать разрешение на папку для пользователя, под которым запущен публикующий этот сервис HTTP-сервер. Пользователь должен быть владельцем папки с правами на чтение / запись / выполнение.

4. ПРОВЕРКА, ВОССТАНОВЛЕНИЕ И ПОДДЕРЖАНИЕ РАБОТОСПОСОБНОСТИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

4.1. Методы проверки работоспособности рабочих станций

Чтобы проверить работоспособность рабочей станции (клиентской части Системы), требуется выполнить следующие действия:

- запустить рабочую станцию;
- запустить web-браузер;
- в адресной строке ввести адрес сервера Системы;
- в окне авторизации ввести логин и пароль (login: admin_user, пароль: admin), нажать на кнопку «Войти».

Система работоспособна, если в результате выполнения действий отображается главная страница.

4.2. Методы проверки работоспособности сервера

Чтобы проверить работоспособность сервера, требуется выполнить следующие действия:

- убедиться в том, что службы Postgres Pro, nginx, MongoDB и сервисов Системы работают.

Для этого убеждаемся, что их статус active, командами:

```
systemctl status nginx
```

```
● nginx.service - The nginx  
   Loaded: loaded (/usr/lib/  
   Active: active (running)
```

```
systemctl status postgrespro-ent-14
```

```
● postgrespro-ent-14.service  
   Loaded: loaded (/usr/lib/s  
   Active: active (running) s  
   Process: 100577 ExecStartPr
```

`systemctl status auth` (в зависимости от того, какое имя службе модуля хаба выдали);

```
● auth.service - [Authentication]  
   Loaded: loaded (/etc/systemd/s  
   Active: active (running) since  
   Main PID: 122667 (dotnet)
```

`systemctl status fs` (в зависимости от того, какое имя службе модуля файлового хранилища выдали);

```
● fs.service - [File Storage  
   Loaded: loaded (/etc/syst  
   Active: active (running)  
   Main PID: 1044 (dotnet)
```

`systemctl status mongod` (в зависимости от того, какое имя службе модуля файлового хранилища выдали).

```
● mongod.service - MongoDB De
   Loaded: loaded (/usr/lib/s
   Active: active (running) s
   Docs: https://docs.mongo
```

– убедиться в том, что Postgres Pro доступен. Для этого достаточно выполнить команду `psql` и в запустившемся интерактивном терминале Postgres Pro выполнить команду: `\conninfo`. При успешном соединении отобразится соответствующее сообщение, например:

```
postgres=# \conninfo
You are connected to database "postgres" as user "postgres" via socket in "/var/run/postgresql" at port "5432".
postgres=#
```

– убедиться в том, что Postgres Pro, nginx, MongoDB и Система открыли все настроенные порты:

`netstat -tnulp`

```
[root@DTS-KORUPCIA-APP ~]# netstat -tnulp
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 127.0.0.1:5000          0.0.0.0:*               LISTEN      51763/dotnet
tcp        0      0 127.0.0.1:5001          0.0.0.0:*               LISTEN      51763/dotnet
tcp        0      0 0.0.0.0:1040            0.0.0.0:*               LISTEN      15021/nginx: master
tcp        0      0 127.0.0.1:5040          0.0.0.0:*               LISTEN      1044/dotnet
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      1147/sshd
tcp        0      0 0.0.0.0:5432            0.0.0.0:*               LISTEN      100581/postgres
tcp        0      0 0.0.0.0:1080            0.0.0.0:*               LISTEN      15021/nginx: master
tcp        0      0 127.0.0.1:5080          0.0.0.0:*               LISTEN      1043/dotnet
tcp        0      0 127.0.0.1:17017         0.0.0.0:*               LISTEN      1347/mongod
tcp        0      0 127.0.0.1:25            0.0.0.0:*               LISTEN      1336/master
tcp        0      0 0.0.0.0:1050            0.0.0.0:*               LISTEN      15021/nginx: master
tcp        0      0 0.0.0.0:1090            0.0.0.0:*               LISTEN      15021/nginx: master
tcp        0      0 127.0.0.1:5090          0.0.0.0:*               LISTEN      1041/dotnet
```

4.3. Методы проверки работоспособности базы данных

В проверку работоспособности базы данных входит:

- проверка физической целостности базы данных;
- проверка сохранения введенных данных.

4.3.1. Проверка физической целостности базы данных

Проверка физической целостности базы данных проводится подсчётом контрольных сумм на файлах баз данных кластера.

Для этого следует предварительно выключить службу СУБД:

`systemctl stop postgrespro-ent-14`

Запустить проверку:

`pg_verify_checksums -D /var/lib/pgpro/ent-14/data/`

```
Проверка контрольных сумм завершена
Версия контрольных сумм данных: 1
Просканировано файлов: 6147
Просканировано блоков: 13774
Неверные контрольные суммы: 0
```

Проверить целостность базы данных mongodb:

- Подключиться к mongo и выполнить:

`use filestorage`

где filestorage наше имя базы данных.

- Открыть список коллекций:

`db.getCollectionInfos();`

- Далее для каждой коллекции выполнить проверку

`db.files.validate({full:true})`

В результате valid должен иметь значение true

```
db.files.validate({full:true})
{
  "ns" : "filestorage.files",
  "nInvalidDocuments" : 0,
  "nrecords" : 90,
  "nIndexes" : 1,
  "keysPerIndex" : {
    "_id_" : 90
  },
  "indexDetails" : {
    "_id_" : {
      "valid" : true
    }
  },
  "valid" : true,
  "repaired" : false,
  "warnings" : [ ],
  "errors" : [ ],
  "extraIndexEntries" : [ ],
  "missingIndexEntries" : [ ],
  "corruptRecords" : [ ],
  "ok" : 1
}
```

4.3.2. Проверка сохранения введенных данных

Для проверки сохранения данных, вносимых в базу данных Системы через браузер, нужно выполнить следующие действия:

- 1) запустить рабочую станцию;
- 2) открыть браузер;
- 3) в адресной строке ввести адрес сервера Системы;

4) в окне авторизации ввести логин и пароль, нажать на кнопку «Войти»;

5) на главной странице выбрать режим **Противодействие коррупции – Учет сообщений о гражданах, замещающих в ОГВ должности ГГС**;

6) нажать на кнопку добавления записи, в форме записи заполнить необходимые поля, прикрепить файл с помощью кнопки «Выбрать файл» и выполнить сохранение записи;

7) закрыть браузер;

8) повторно выполнить пп. 2-5;

9) в списке сообщений найти добавленную запись, открыть для просмотра, открыть прикрепленный файл кликом по гиперссылке;

10) закрыть форму просмотра записи, удалить запись;

Сохранение добавленной записи, в том числе прикрепленного файла, свидетельствует о работоспособности БД Системы.

4.4. Методы восстановления работоспособности сервера

Работоспособность сервера, в случае его отказа, восстанавливается специалистами из подразделения технической поддержки. Если технических неисправностей в оборудовании сервера не обнаружено и операционная система сервера работает без сбоев, то для восстановления его работоспособности рекомендуется переустановить серверную часть Системы.

4.5. Методы восстановления работоспособности базы данных

4.5.1. Методы восстановления работоспособности базы данных под управлением СУБД Postgres Pro 14

Для обеспечения возможности восстановления базы данных (например, поврежденной) администратор должен периодически выполнять её резервное копирование.

4.5.1.1. Резервное копирование базы данных

- 1) Подключиться под пользователем **postgres**:
su postgres

- 2) Создать резервную копию базы данных «database» (здесь подставить имя исходной базы данных) в файл по пути /tmp/database.bak:

```
pg_dump database > /tmp/database.bak
```

В результате процедуры по указанному пути будет создан файл резервной копии.

Рекомендация. Для гарантирования сохранности данных файл рекомендуется скопировать (средствами операционной системы) на внешнее устройство (CD-диск, флэш-карту или др.).

4.5.1.2. Восстановление базы данных

- 1) Подключиться под пользователем postgres:

```
su postgres
```

- 2) Восстановить базу «database» из файла бэкапа /tmp/database.bak:

```
psql database < /tmp/database.bak
```

4.5.2. Методы восстановления работоспособности базы данных под управлением СУБД MongoDB

4.5.2.1. Резервное копирование базы данных

Для резервного копирования на сервере необходимо запустить процедуру mongodump. В качестве параметров следует указать порт (port), на котором запущена служба, базу данных (db) и путь к директории (out):

```
mongodump --port=27017 --db filestorage --out=/tmp/mongo
```

В результате процедуры по указанному пути будет создана директория с резервной копией базы данных.

4.5.2.2. Восстановление базы данных

Для резервного копирования на сервере необходимо запустить процедуру mongorestore. В качестве параметров указываем порт (port) и директорию с бэкапом (dir).

```
mongorestore --port=27017 --dir=/tmp/mongo/
```

4.6. Методы поддержания целостности базы данных

Целостность базы данных обеспечивается встроенными средствами СУБД Postgres Pro 14 и СУБД MongoDB 6.

Кроме того, целостность баз данных Системы обеспечивается на этапе ввода данных посредством заданных разработчиком правил.

4.7. Методы поддержания безопасности базы данных

Безопасность базы данных обеспечивается встроенными средствами СУБД Postgres Pro 14, СУБД MongoDB 6 и операционными системами серверов.

4.8. Методы диагностирования проблем обновления баз данных

Обновление баз данных осуществляются через утилиту Quarta.Migrator.exe

В процессе прогона миграционных скриптов и обновления функций, триггеров и др. в случае возникновения ошибок утилита логирует проблемы.

4.8.1. Диагностирование проблем в работе сервисов

В процессе работы сервисы записывают ошибки в лог.

Для диагностирования проблем необходим SSH Клиент (например, Putty или аналог).

- 1) Подключиться к серверу, на котором осуществляется хостинг проблемного сервиса.

Для входа требуется:

- Хост (ip сервера);
- Порт;
- Логин;
- Пароль.

- 2) Получить идентификатор интересующего сервиса.

Для этого можно воспользоваться командой получения перечня докер-образов:

`docker ps -a`

Пример:

```
CONTAINER ID   IMAGE
5fa324f5c2f7   harbor.quarta.su/app-documentstorage/frontend:latest
a5998baf3ecd   harbor.quarta.su/app-documentstorage/app:latest
40ad2cab9706   harbor.quarta.su/app-integration-gosduma/app:latest
20305c5b0349   harbor.quarta.su/app-integration-gosduma/frontend:latest
ea3db8e85d49   harbor.quarta.su/app-notification/frontend:latest
32f44cc86e5d   harbor.quarta.su/app-notification/app:latest
da9c4c91f3f2   harbor.quarta.su/app-staffstructure/app:latest
4f5c1b57cc5f   harbor.quarta.su/app-staffstructure/frontend:latest
```

Например, необходимы логи сервиса «Хранение и оборот документов». Его имя: App-documentstorage, значит, идентификатор: a5998baf3ecd.

Если docker не используется, то логи доступны через команды вида `journalctl -u %наименование сервиса% -n 100 -f`

3) Получить логи сервиса с помощью команды команда:

`docker logs a5998baf3ecd`

```
warn: Microsoft.EntityFrameworkCore.Model.Validation(10622)
      Entity 'Person' has a global query filter defined and is the required end of a relationship with the entity 'PersonStructuralUnit'. This may lead to unexpected results when the required entity is filtered out. Either configure
      a query filters for both entities in the navigation. See https://go.microsoft.com/fwlink/?linkid=2131316 for more information.
Quarta.Migrator запускается...
Проверка соединения с базой... успешно.
Quarta.Migrator готов к работе.
Запуск миграции ED.
Миграция запущена.
Получение завершённых миграций... готово!
Получение миграционных скриптов...
Основной контекст: /Migrations/Migrations
Миграция 0 MigrationFile 20221128_001_kuznetsov_vv_alter_table_route_history.sql... добавлена... Выполнение операции прервано вследствие ошибки. 42703: столбец "document_executor_id" в таблице "document_route_history" не существует
crit: Microsoft.AspNetCore.Hosting.Diagnostics(6)
      Application startup exception
      System.Exception: Мигратор завершил работу с ошибкой.
      at Quarta.DS.Migrator.Builder.RunMigrations(IApplicationBuilder builder) in /app/Quarta.DS.Migrator/Builder.cs:line 67
      at Quarta.DS.WebApi.Startup.Configure(IApplicationBuilder app, IWebHostEnvironment env) in /app/Quarta.DS.WebApi/Startup.cs:line 110
      at System.RuntimeMethodHandle.InvokeMethod(Object target, Span`1 arguments, Signature sig, Boolean constructor, Boolean wrapExceptions)
      at System.Reflection.RuntimeMethodInfo.Invoke(Object obj, BindingFlags invokeAttr, Binder binder, Object[] parameters, CultureInfo culture)
      at Microsoft.AspNetCore.Hosting.ConfigureBuilder.Invoke(Object instance, IApplicationBuilder builder)
      at Microsoft.AspNetCore.Hosting.ConfigureBuilder.<>c__DisplayClass4.0.<Build>b__0(IApplicationBuilder builder)
      at Microsoft.AspNetCore.Hosting.GenericWebHostBuilder.<>c__DisplayClass15.0.<UseStartup>b__1(IApplicationBuilder app)
      at Microsoft.AspNetCore.Mvc.Filters.MiddlewareFilterBuilderStartupFilter.<>c__DisplayClass0.0.<Configure>b__0(MiddlewareFilterBuilder IMiddlewareFilterBuilder builder)
      at Microsoft.AspNetCore.Hosting.FilteringStartupFilter.<>c__DisplayClass0.0.<Configure>b__0(IApplicationBuilder app)
      at Microsoft.AspNetCore.Hosting.GenericWebHostService.StartAsync(CancellationToken cancellationToken)
      Unhandled exception. System.Exception: Мигратор завершил работу с ошибкой.
```

В логах можно отследить предупреждения и ошибки работы .Net-сервисов в зависимости от настройки секции Logging конфигурационного файла appsettings.json.

Настройка фиксации только ошибок:

```
"Logging": {
  "LogLevel": {
    "Default": "Error",
    "System": "Error" } },
```

Настройка логирования предупреждений:

```
"Logging": {
  "LogLevel": {
    "Default": "Warning",
    "System": "Warning" } },
```

Настройка логирования максимальной информации (не рекомендуется, т.к. лог будет перегружен):

```
"Logging": {
  "LogLevel": {
    "Default": "Info",
    "System": "Info" } },
```

5. АДМИНИСТРИРОВАНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Администрирование Системы обеспечивает:

- настройку и управление параметрами системы аутентификации пользователей, обеспечивающей доступ пользователей к ресурсам информационного портала;
- ведение списка пользователей, обеспечение регистрации пользователей информационного портала с предоставлением им ролей и прав доступа;
- ведение списка приложений и web-ресурсов;
- просмотр списка параметров работы Системы, настройка значений параметров для организаций группы компаний;
- ведение журналов событий аутентификации пользователей и проверок их прав доступа к информационным ресурсам, с возможностью формирования отчета об активности пользователей за период.

Задачи администрирования Системы выполняются в сервисе «Администрирование и контроль доступа (Хаб)».

Сервис Хаб предназначен для управления набором подключенных сервисов, управления правами доступа пользователей к Системе, авторизации и аутентификации пользователей, журналирования событий безопасности, а также настройки некоторых параметров функционирования Системы.

Сервис включает следующие режимы:

- Пользователи;
- Роли;
- Сотрудники;
- Константы;
- Клиенты;
- Ресурсы;
- События аутентификации;
- События безопасности;
- Активность пользователей.

5.1. Доступ к сервису

Для получения доступа к сервису Хаб:

4) в **адресной строке** браузера ввести адрес сервера. Адрес сервера имеет следующий вид – http://адрес_веб-сервера:порт (пример: <http://localhost:8080>);

5) в окне **Авторизации** ввести логин и пароль пользователя с правами администратора (*по умолчанию доступен пользователь «test» с паролем «1»*), нажать на кнопку **«Войти»**:

Рис. 5.1 – Окно авторизации

Для перехода в раздел «Администрирование» следует нажать кнопку профиля и в открывшемся меню выбрать **Администрирование**.

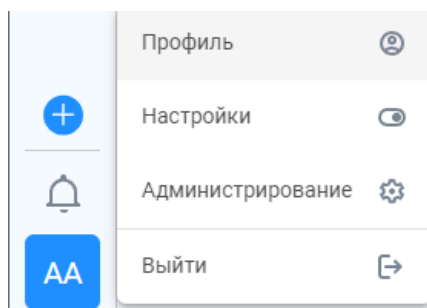


Рис. 5.2

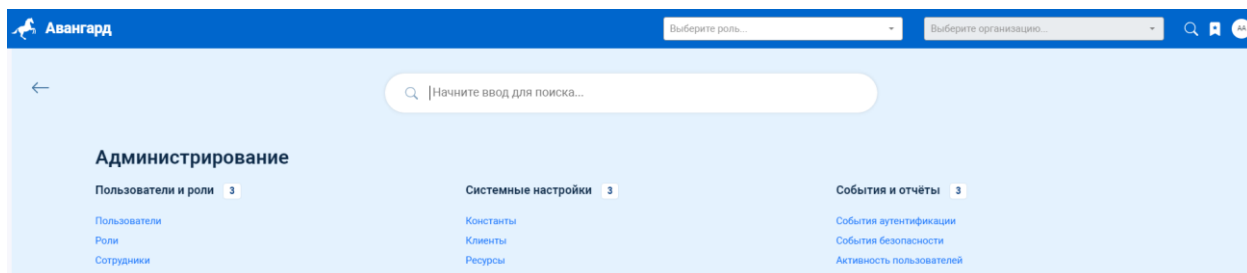


Рис. 5.3 – Главная страница сервиса администрирования

5.2. Управление доступом

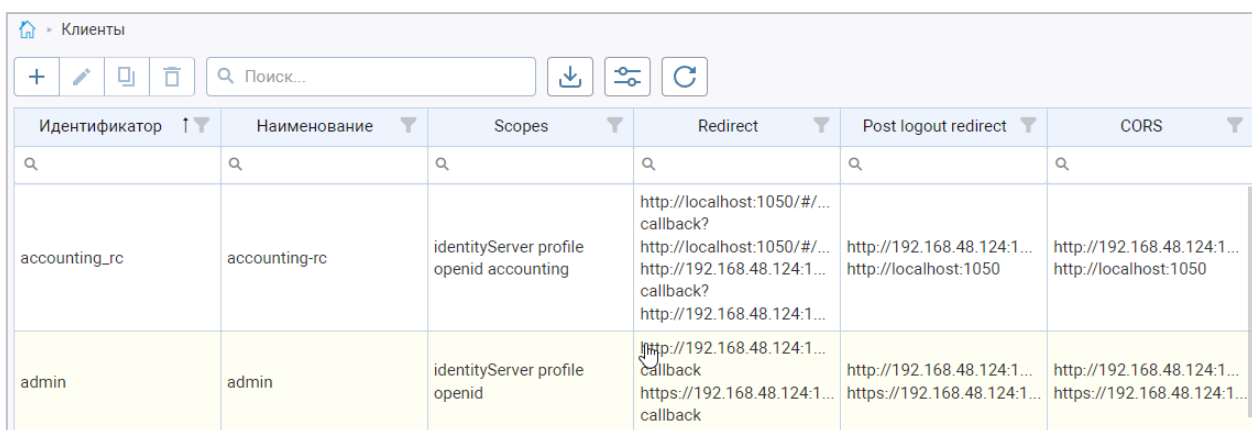
К задачам управления доступом относятся:

- Настройка фронтенд-приложений и бэкенд-сервисов системы (клиентов, ресурсов);
- Настройка прав доступа к режимам Системы (формирование ролей, списка пользователей и сотрудников).

При запуске Системы выполняется сначала верификация клиента, потом авторизация пользователя. В ходе работы пользователя клиент (приложение) взаимодействует с ресурсами и сервисами.

5.2.1. Клиенты

Выбрать подраздел «*Системные настройки*» → *Клиенты*. Откроется списочная форма Клиент-Приложений.



| Идентификатор | Наименование | Scopes | Redirect | Post logout redirect | CORS |
|---------------|---------------|---|--|---|---|
| accounting_rc | accounting-rc | identityServer profile openid accounting | http://localhost:1050/#/... callback? http://localhost:1050/#/... http://192.168.48.124:1... callback? http://192.168.48.124:1... | http://192.168.48.124:1... http://localhost:1050 | http://192.168.48.124:1... http://localhost:1050 |
| admin | admin | identityServer profile openid | http://192.168.48.124:1... callback https://192.168.48.124:1... callback | http://192.168.48.124:1... https://192.168.48.124:1... | http://192.168.48.124:1... https://192.168.48.124:1... |

Рис. 5.4 – Клиенты

Списочная форма Клиент-Приложений содержит следующий набор сведений о настройках приложений:

- 1) **Идентификатор** – уникальный строковый идентификатор приложения;
- 2) **Наименование** – Читабельное представление (описание) приложения;
- 3) **Scopes** – Перечень разрешений, которые может запрашивать приложение;
- 4) **Redirect** – Перечень разрешенных переадресаций пользователя после прохода аутентификации;

5) **Post Logout redirect** – Перечень разрешенных переадресаций пользователя после того, как пользователь выходит из системы;

6) **CORS** – Перечень хостов, с которых приложению разрешено осуществлять запросы на аутентификацию/авторизацию пользователя.

Создание нового клиента:

Нажать «+» над списочной формой клиентов. Будет открыта форма заполнения данных:

Управление клиентом admin

ID приложения *

admin

Наименование приложения *

admin

Доступные ресурсы (API, OpenId, Profile)

identityServer profile openid

Redirect Uri * Post Logout Redirect Uri * CORS origins

http://192.168.48.124:1098/login-callback
https://192.168.48.124:1098/login-callback

http://192.168.48.124:1098/login
https://192.168.48.124:1098/login

http://192.168.48.124:1098
https://192.168.48.124:1098

Время жизни токена (сек) *

86400

Тип токена

Reference

Сохранить Закрыть

Рис. 5.5 – Описание клиента

Поля повторяют ранее описанные поля для списочной формы. Для всех полей, где возможен ввод нескольких значений, их порядок не имеет значения. Из особенностей отдельных полей:

Доступные ресурсы – каждый клиент должен содержать обязательные параметры: *openid*, *profile*. Остальные значения должны браться из списка идентификаторов Ресурсов, заполненных в режиме «Ресурсы». Разделителем всех значений является символ «пробел».

Время жизни токена – заполняется исходя из требований к безопасности. Чем меньше интервал жизни токена, тем чаще осуществляется его автоматическое обновление, однако не рекомендуется делать его короче среднего времени сессии пользователя.

Тип токена – влияет на то, как будет передаваться токен авторизации из SPA-Приложения в Ресурс:

- 1) Reference: передаваться будет только обезличенный хэш-ключ, который требует подтверждения от сервиса Хаб;
- 2) JWT – самодостаточный токен.

Коды аутентификации – ключи для автоматизации отдельных аспектов процесса разработки, не предусмотрены для использования в реальных сценариях.

Требования к полям **Redirect Uri, Post Logout Redirect Uri** – разрешения адреса указываются в формате URL без указания возможных параметров или якорей: *<схема>: [//<хост>[:<порт>]]/[URL-путь]*.

Пример корректно написанных URL:

https://auth.gd-workspace.ru/callback

http://192.168.48.124:1090/login

https://gd-workspace.ru

Требования к полю CORS: разрешенный адрес должен содержать только элементы: *<схема>: [//<хост>[:<порт>]]* (т.е. без путей, параметров и якорей).

Пример:

https://gd-workspace.ru

http://auth-test.ru

Для сохранения нового клиента или редактирования имеющегося, нужно нажать кнопку «**Сохранить**». В случае успешного сохранения будет выполнена переадресация на списочную форму.

5.2.2. Ресурсы

Выбрать подраздел «**Системные настройки**» → «**Ресурсы**».

Откроется списочная форма Ресурсов.

| Идентификатор | Отображаемое название |
|----------------|-----------------------|
| bi | bi |
| classifier | classifier |
| ds | Document Storage API |
| identityServer | identityServer |
| integration | Сервис Интеграции |
| notification | notification |
| ss | Staff Structure |

Рис. 5.6 – Ресурсы

Списочная форма Ресурсов содержит следующий набор сведений:

Идентификатор — уникальный строковый идентификатор ресурса;

Отображаемое название — читабельное представление (описание) приложения.

5.2.2.1. Создание нового ресурса

Нажать «+» над списочной формой клиентов. Будет открыта форма заполнения данных:

Создание нового ресурса

Наименование *

Отображаемое наименование *

Адрес

http://localhost:5000

Сохранить Закрыть

Рис. 5.7 – Создание нового ресурса

Из особенностей полей:

Наименование — не должно содержать пробелов в названии.

Адрес — должен соответствовать формату **<схема>://<хост>[:<порт>]**.

Например: **https://gd-workspace.ru**.

В случае успешного сохранения будет выполнена переадресация на карточку Ресурса.

Пример:

The screenshot shows a web interface for editing a resource. The breadcrumb path is 'Ресурсы > Редактирование'. On the left, there is a sidebar with three items: 'Информация' (highlighted in blue), 'API', and 'Ключи аутентификации'. The main area is titled 'Управление ресурсом Сервис ОШС'. It contains three input fields: 'Наименование' with the value 'ss', 'Отображаемое наименование' with the value 'Сервис ОШС', and 'Адрес' with the value 'https://192.168.48.124:3170'. At the bottom of the form are two buttons: 'Сохранить' (highlighted in blue) and 'Закрыть'.

Рис. 5.8 – Описание ресурса

5.2.2.2. Раздел API

Если поле *Адрес* было заполнено корректно и Ресурс активен (запущен на сервере и имеет действующий ключ аутентификации), в данном режиме будет отражаться перечень его *URL*, которые могут быть использованы приложениями для обращения к данным в соответствии с их бизнес-логикой.

Кнопка «*Перезагрузить*» позволяет вручную запросить данный список у Ресурса. Полезен для случаев, когда спецификации Ресурса изменяются и эти изменения не были занесены в сервис Хаб во время запуска приложения.

| Ресурсы · Редактирование | | | |
|-----------------------------|---|----------|---|
| Информация | | | |
| API | | | |
| Полномочия | | | |
| Ключи аутентификации | | | |
| Управление API | | | |
| Адрес | | | |
| https://192.168.48.124:3170 | | | |
| Перезагрузить | | | |
| Поиск... | | | |
| | | | |
| Имя | URL | Включена | |
| active | /api/activity/active/{id Guid} | (Все) | ✓ |
| employee-list | /api/employee/list | | ✓ |
| get-employee | /api/employee/{id Guid} | | ✓ |
| get-position | /api/position/{id Guid} | | ✓ |
| get-temporarily-vacant | /api/temporarily-vacant/{id Guid} | | ✓ |
| get-unit | /api/unit/{id Guid} | | ✓ |
| person-fio-genitive | /api/integration-get/person-fio-genitive | | ✓ |
| person-staff-unit | /api/integration-get/person-staff-unit/{personId} | | ✓ |
| position-list | /api/position/list | | ✓ |
| select-structural-units | /api/integration-get/select-structural-units | | ✓ |
| structural-units | /api/integration-get/structural-units | | ✓ |
| structural-units-by-id | /api/integration-get/structural-units-by-id/{id} | | ✓ |
| temporarily-vacant-list | /api/temporarily-vacant/list | | ✓ |
| work-activity | /api/activity/{id Guid} | | ✓ |

Рис. 5.9 – Управление API

5.2.2.3. Раздел «Полномочия»

В данном разделе описывается справочник прав доступа, которые использует Ресурс при обращении к своему API. Эти данные обновляются при каждом запуске Ресурса на сервере.

Носит уведомительный характер и используется как источник данных при назначении Полномочий ролям в режиме «Роли».

| Ресурсы · Редактирование | | | |
|---------------------------------------|---|----------|---|
| Информация | | | |
| API | | | |
| Полномочия | | | |
| Ключи аутентификации | | | |
| Управление Полномочиями | | | |
| Поиск... | | | |
| | | | |
| Имя | Полномочие | Включено | |
| Управление справочником должностей | ss/classifier/position/manage | (Все) | ✓ |
| Управление справочником подразделений | ss/classifier/temporarily_vacant/manage | | ✓ |
| Управление справочником подразделений | ss/classifier/structural_unit/manage | | ✓ |
| Управление справочником сотрудников | ss/classifier/employee/manage | | ✓ |
| Чтение справочника должностей | ss/classifier/position/read | | ✓ |
| Чтение справочника подразделений | ss/classifier/temporarily_vacant/read | | ✓ |
| Чтение справочника подразделений | ss/classifier/structural_unit/read | | ✓ |

Рис. 5.10 – Управление Полномочиями

5.2.2.4. Раздел «Ключи аутентификации»

В данном разделе вносятся ключи, которые могут использоваться Ресурсом при следующих обращениях к сервису Хаб:

- 1) Верификация токена, полученного от Клиента;
- 2) Межсерверная передача данных (Полномочия);
- 3) Логирование и пр.

Ресурс может иметь больше одного действующего ключа, это позволяет выполнять постепенный rollout-ключей в случае кластерного развертывания Ресурса (полезно для сервисов с высокой нагрузкой).

Управление ключами доступа

Новый ключ ⓘ

Введите новый ключ

Сохранить

Удалить 🔍 Поиск...

| Хэш | Дата создания | Действителен до |
|---------------------------------------|---------------|-----------------|
| FKlvt19kwhRFAUUUwLkZr87dvUBa9m0r0H... | 29.11.2022 | |

Рис. 5.11 – Управление ключами доступа

Ключи хранятся в захешированном виде, что исключает возможность их чтения после записи.

В случае отсутствия хоть одного действующего ключа, Ресурс не сможет аутентифицироваться в сервисе Хаб. Это приведет к тому, что Ресурс не сможет проверить входящие токены и будет обязан отвергнуть запросы пользователей, как недоверенные.

После настройки Администратором Системы комбинации Клиента + Ресурса, пользователь сможет зайти в Приложение (ака Клиент) по адресу на котором он развернут:

Например: ***https://192.168.48.124:6200***.

В случае успешной настройки, пользователь будет переадресован на страницу ввода Логина/Пароля, а после — обратно в приложение.

В случае если настройка была выполнена с ошибкой, то возможны следующие сценарии:

- 1) Ошибка настройки Клиента — будет выведена ошибка при переадресации на страницу ввода логина при входе в систему;
- 2) Ошибка настройки Ресурса — будет выведена **Ошибка Аутентификации** при попытке пользователя запросить данные из Ресурса,

или ошибка **Отказано в Доступе**, если у пользователя недостаточно прав для выбранного действия.

5.2.3. Роли

При описании роли:

- определяется список доступных объектов и, для некоторых объектов, права доступа к объекту (управление или только чтение);
- могут быть ограничены права на уровне записи (разделение права для пользователей в разрезе динамически меняющихся данных).

В дереве присутствуют различные объекты: объекты структуры системы (подсистемы и модули, режимы), объекты интерфейса (разделы, пункты меню), объекты с общим назначением (приказы) и другие.

Чекбокс объекта имеет вид ☒, если роли предоставлен доступ к объекту и ко всем подчиненным объектам текущего объекта, вид ☐, если предоставлен доступ к некоторым подчиненным объектам.

Чтобы предоставить доступ к объекту, следует выполнить клик на чекбоксе. При предоставлении доступа к объекту автоматически предоставляется доступ к подчиненным объектам.

Роль может быть заблокирована / разблокирована (в форме списка ролей).

| Наименование | Активна | Назначается... |
|---------------|---------|----------------|
| Администратор | Да | (Все) |

Рис. 5.12 – Роли

Основная информация

Администратор

☒ Назначать новым пользователям

☒ Полный доступ по строкам (фильтрация по командам сохраняется)

Сохранить Закрыть

Рис. 5.13 – Роль. Основная информация

Полномочия

☐ Показать только выданные

☐ Свернуть все

Поиск

☒ Выбрать все

- ☒ Администрирование
- ☒ Document Storage API
- ☒ Staff Structure
- ☒ Сервис Интеграции

Сохранить Закрыть

Рис. 5.14 – Роль. Полномочия

🏠 > Пользователи > Управление пользователем "admin"

Основная информация

Роли

Смена пароля

Основная информация

Логин *
admin

Фамилия *
admin

Имя *
admin

Отчество
admin

Сотрудник
▼

Электронная почта *
Электронная почта

СНИЛС
СНИЛС

☐ Системная учетная запись? ⓘ

Сохранить Закреть

Рис. 5.17 – Пользователь. Основные сведения

🏠 > Пользователи > Управление пользователем "admin"

Основная информация

Роли

Смена пароля

Рабочие места

🔍 Начать поиск...

| | Назначенные роли |
|--------------------------------------|------------------|
| 🔍 | (Все) ▼ |
| ▼ Центральный сегмент | Администратор ✕ |
| ▼ Получатель бюджетных средств (ПБС) | Выбрать... |
| ▼ Администрация | Выбрать... |
| Администрация | Выбрать... |
| Кварта | Выбрать... |
| Помощники, Советники | Выбрать... |
| Центр разработки | Выбрать... |
| Экспертное управление | Выбрать... |
| ▼ Аппарат | Выбрать... |

Сохранить Закреть

Рис. 5.18 – Пользователь. Роли

🏠 > Пользователи > Управление пользователем "admin"

Основная информация

Роли

Смена пароля

Изменение пароля пользователя admin admin admin

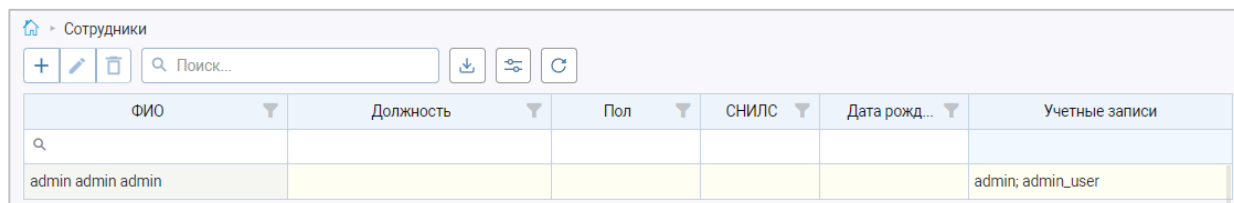
Введите новый пароль

Сохранить Закреть

Рис. 5.19 – Смена пароля

5.2.5. Сотрудники

Режим сотрудники используется для просмотра информации личных данных сотрудников.



| ФИО | Должность | Пол | СНИЛС | Дата рожд... | Учетные записи |
|-------------------|-----------|-----|-------|--------------|-------------------|
| admin admin admin | | | | | admin; admin_user |

Рис. 5.20 – Сотрудники

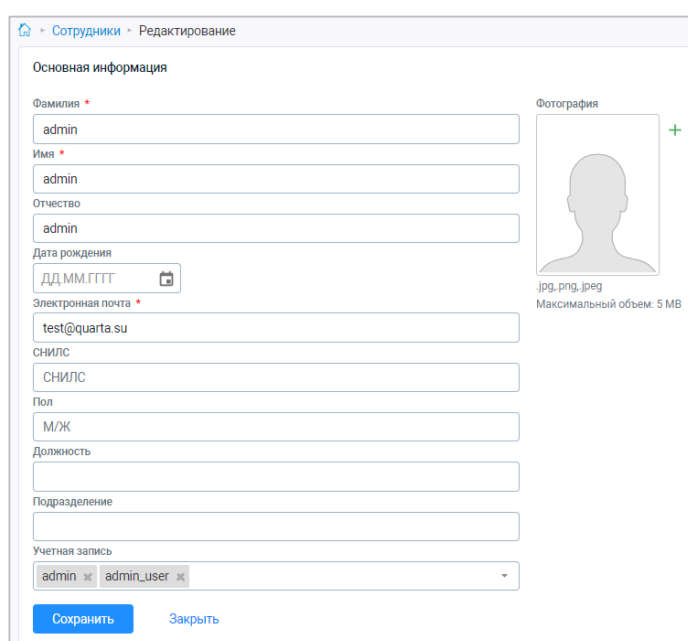
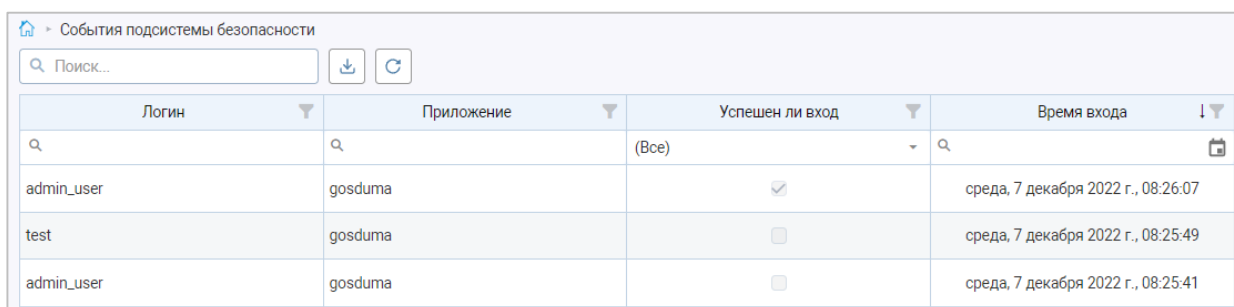


Рис. 5.21 – Сотрудники. Основная информация

5.2.6. События

Режимы просмотра событий используются для контроля подключения пользователей к системе и проверки прав доступа к объектам системы.

Справка об активности пользователей представляет собой xls-отчет с общим количеством входов каждого пользователя за заданный период.



| Логин | Приложение | Успешен ли вход | Время входа |
|------------|------------|-----------------|------------------------------------|
| admin_user | gosduma | ✓ | среда, 7 декабря 2022 г., 08:26:07 |
| test | gosduma | ✗ | среда, 7 декабря 2022 г., 08:25:49 |
| admin_user | gosduma | ✗ | среда, 7 декабря 2022 г., 08:25:41 |

Рис. 5.22 – События аутентификации

| События подсистемы безопасности | | | | | | | | | |
|---------------------------------|--------|--------|---|--------------------------------------|--------------------------------------|---------|---|--|------------------------------------|
| Поиск... | | | | | | | | | |
| Ло... | Пр... | Ре... | Код права | Иденти... | Иденти... | Успе... | Описание | | В... |
| Q | Q | Q | Q | Q | Q | (Все) | Q | | Q |
| admin... | ds | ds | ds/file_library/read | eae5f50f-2239-40c6-ba23-64ee9f729c37 | 6b600191-8099-4370-b7bf-7508956ed375 | ✓ | Вызвана проверка права (ds/file_library/read) " (Имя не найдено)" для пользователя admin_user. Пользователь работает под ролью "Администратор"(eae5f50f-2239-40c6-ba23-64ee9f729c37) в организации "Центральный сегмент"(6b600191-8099-4370-b7bf-7508956ed375). В доступе разрешено. | | среда, 7 декабря 2022 г., 11:26:09 |
| test | sophie | sophie | uri://schemas.quarta.su...registration/read | 6480aca8-4e2a-4834-80f3-f9beafa70481 | 2873b295-a717-4b8e-bc05-6c23f52a627d | ✓ | Вызвана проверка права (uri://schemas.quarta.su/staff/anticorruption/exter...registration/read) "Чтение сведений об адресах сайтов и (или) страниц сайтов" для пользователя test. Пользователь работает под ролью "Администратор ЯНАО"(6480aca8-4e2a-4834-80f3-f9beafa70481) в организации "Правительство Ямало-Ненецкого автономного округа"(2873b295-a717-4b8e-bc05-6c23f52a627d). В доступе разрешено. | | среда, 7 декабря 2022 г., 11:26:07 |

Рис. 5.23 – События безопасности

События подсистемы безопасности
Справка об активности пользователей

Дата начала периода
01.01.2023

Дата окончания периода
31.01.2023

Сформировать отчет

Рис. 5.24 – Настройка периода отчета

| Справка об активности пользователей | | |
|-------------------------------------|----------------|-------------------|
| с 01.01.2023 по 31.01.2023 | | |
| Субъект доступа | Учётная запись | Количество входов |
| admin a. a. | admin | 18 |
| Admin A. D. | Admin_user | 4 |
| Admin A. D. | admin_user | 635 |
| test t. t. | test | 146 |
| test t. t. | Test | 452 |

Рис. 5.25 – Пример отчета

5.3. Константы

Константы обеспечивают возможность настройки для организаций группы компаний различных значений параметров работы Системы.

Перечень параметров работы Системы определяется организацией-разработчиком. Пользователю доступно изменение набора доступных к выбору значений параметра и настройка значений параметра для организаций группы компаний.

| Константы | | | | |
|--|---|--------------|---------------------|------------------------|
| Код | Наименование | Тип значения | Тип выбора | Значения по умолчанию |
| HolidayCompositionTypeGeneralCountDays | Количество дней основного отпуска по умолчанию | Числовое | Одиночное | 28 |
| OptionalFiasAddressEnabled | Возможность ввода произвольного адреса в структуре ФИАС | Логическое | Одиночное | false; true |
| PersonalFileClassifierDetail | Типы должностей сотрудников, отбираемые в контролах выборки | Числовое | Множественный выбор | 01; 02; 03; 04; 05; 06 |
| PositionTypes | Перечисление доступных типов должностей | Строковое | Множественный выбор | 01; 02; 03; 04; 05; 06 |
| PositionTypesEdit | Перечисление доступных для редактирования типов должностей | Строковое | Множественный выбор | 01; 02; 03; 04; 05; 06 |

Рис. 5.26 – Список параметров

Основная информация

Все значения

Организации

Основная информация

Код *

HolidayCompositionTypeGeneralCountDays

Наименование *

Количество дней основного отпуска по умолчанию

Значения по умолчанию *

28

Сохранить
Закреть

Рис. 5.27 – Параметр. Основная информация

| Все значения | |
|--------------|-------------------------------------|
| Значение | Значение по умолчанию? |
| 28 | <input checked="" type="checkbox"/> |

Рис. 5.28 – Параметр. Список значений

Организации

Начать поиск...

| Разрешенные значения |
|---|
| <div> <div>Центральный сегмент</div> <div> <div>28</div> </div> </div> <div> <div>Получатель бюджетных средств (ПБС)</div> <div>Выбрать...</div> </div> <div> <div>Администрация</div> <div>Выбрать...</div> </div> <div> <div>Администрация</div> <div>Выбрать...</div> </div> <div> <div>Кварта</div> <div>Выбрать...</div> </div> <div> <div>Помощники, Советники</div> <div>Выбрать...</div> </div> <div> <div>Центр разработки</div> <div>Выбрать...</div> </div> <div> <div>Экспертное управление</div> <div>Выбрать...</div> </div> <div> <div>Аппарат</div> <div>Выбрать...</div> </div> <div> <div>Департамент информационных технологий и связи</div> <div> <div>28</div> </div> </div> <div> <div>Департамент культуры</div> <div>Выбрать...</div> </div> <div> <div>Управление делами</div> <div>Выбрать...</div> </div> <div> <div>Главное медицинское управление</div> <div>Выбрать...</div> </div> |

Сохранить
Закреть

Рис. 5.29 – Параметр. Настройка значений для организаций

6. АВАРИЙНЫЕ СИТУАЦИИ

Отсутствие доступа к Системе, нарушение функционирования Системы, нештатное поведение Системы может возникать по причине сбоев в функционировании аппаратного обеспечения и каналов передачи данных, прикладного и специального программного обеспечения, ошибок ввода данных авторизации и ошибок администрирования Системы, а также по причине несанкционированного вмешательства в данные Системы.

В зависимости от предполагаемой причины сбоя следует проверить:

- подключение аппаратных средств и работоспособность каналов передачи данных;
- работоспособность серверного программного обеспечения и БД Системы (пп. 4.2, 4.3);
- журнал событий и корректность распределения прав доступа пользователей (п. 5), отсутствие вредоносного ПО на сервере и рабочих станциях пользователей.

Восстановление работоспособности программного обеспечения и БД Системы описано в пп. 4.4, 4.5.