Инструкция для разворачивания front+back

Требования к инфраструктуре

Операционная система

ОС	Версия
Debian	`10`
Ubuntu	`20`
ALT Linux	`10`
Astra Linux	`1.7`
RedOS	`7.3.4`

Стороннее ПО

Компонент	Версия	Ресурс
Docker	`19.03.0`	https://docs.docker.com/engine/install/

docker-compose	`2.24.7`	https://docs.docker.com/compose/install/linux/
Nginx	`1.18.0`	https://docs.nginx.com/nginx/admin-guide/installing-nginx/installing-nginx-open-source/
PostgreSQL	`16-alpine`	https://hub.docker.com/_/postgres

Примечание: Версии ПО указаны для развертывания новых стендов, на существующих обновления не требуются.

Обязательным требованием для запуска является наличие на хостовой машине следующего ПО:

- Docker
- docker-compose

Рабочая конфигурация предполагает наличие двух веб-серверов Nginx:

- nginx-1 - веб-сервер установленный на хостовой ОС;
- nginx-2 - контейнер с веб-сервером nginx. используется для маршрутизации между микросервисами.

Веб-сервер nginx-1 может располагаться как на одной машине с сервисами, так и на отдельной. Основным требованием является наличие у веб-сервера доступа к порту хостовой машины, на которой запущен nginx-2 - маршрутизатор запроса на сервисы (по умолчанию используется 10000 порт).

Важно: Для каждого публичного кабинета, созданного на платформе, требуется наличие домена, спроксированного на стенд, а также SSL-сертификат.

Инсталляция стороннего ПО на сервер

Установка Nginx на хост

Установка Docker

Установка loki-docker-driver

Установка loki-docker-driver без доступа в интернет

```
sudo cp -r ./loki-docker-driver/* /var/lib/docker/plugins
sudo systemctl restart docker
docker plugin disable loki
docker plugin enable loki
```

Установка docker-compose

docker compose версии 2.x.x и старше устанавливается вместе с пакетом docker engine.

Распаковка контейнеров и загрузка образов

На сервере, создать в корне каталог */docker-containers*, куда распаковать содержимое архива *config.tar.gz*.

Распаковка конфигурации

```
sudo mkdir /docker-containers
sudo tar -xvf config.tar -C /docker-containers
```

Если с сервера нет доступа в интернет, для загрузки образов с Harbor, то поставка происходит через SFTP, для этого необходимо:

1. Загрузить архивы с SFTP на сервер, где будет размещен комплект поставки
2. Перейти в каталог с образами (каждый образ в архиве имеет вид <SERTVICE_NAME>.tar.gz)
3. Выполнить загрузку следующей командой:

Распаковка и загрузка образов

```
sudo docker load -i <SERVICE_NAME>.tar.gz                                    # Для загрузки одного образа
sudo ls -1 *.tar.gz | xargs --no-run-if-empty -L 1 docker load -i   # Для загрузки всех образов в каталоге
```

Проверить список загруженных образов можно командой:

Вывод списка образов

```
docker image ls
```

Перейти в каталог */docker-containers.* Структура каталога, из комплекта поставки, соответствует примеру ниже.

Все микросервисы приложений должны находиться в директории *./services*, все инфраструктурные микросервисы должны находиться в директории *./infra.*

Структура директорий

```
/docker-containers   # основной каталог, в который размещаются все каталоги с файлами docker-compose и настройками сервисов
├── ./services    # каталог с сервисами
|   ├── ./docker-compose.yml  #  файл содержащий описание архитектуры docker-контейнеров
|   ├── .env  # общий файл переменных среды
|   ├── ./action-api  #  каталог сервиса
|   |   ├── DefaultSetting
|   |   |   └── appsettings.DefaultSetting.json  #  файл настроек сервиса
|   |   └── .env  # файл переменных среды данного сервиса
|   |
|   ├── ./application-api
|   |   ├── DefaultSetting
|   |   |   └── appsettings.DefaultSetting.json
|   |   └── .env
|   └── ./...
|       ├── DefaultSetting
|       |   └── appsettings.DefaultSetting.json
|       └── .env
└── ./infra   #  каталог с инфраструктурными сервисами
    ├── docker-compose.yml
    └──  ./postgres   #  Каталог появится, после старта комоуза.
```

Настройки Nginx

Настройки хостового Nginx

Добавить проксирование с веб-сервера реверс-прокси, установленный непосредственно на хост с сервисами (с того, что должно проксировать на nginx в контейнер, который на 10000 порту) .

В примере ниже - если на хосте установлен Nginx, то конфигурация расположенная в */etc/nginx/sites-available* проксировала запросы на Nginx в контейнере).

Директиву *server_name* заменить на свой домен.

Пример конфигурации хостового Nginx

```
server {
  # публичные домены
  server_name
    client-lk-browser-dev.sado.ru # домен
    ...
    <some_name> # другие домены

  listen 80;

  location / {

     client_max_body_size 30m;

     proxy_set_header  Host $host;
     proxy_set_header  X-Forwarded-Proto $scheme;
     proxy_set_header  X-Real-IP $remote_addr;
     proxy_set_header  X-Forwarded-For $proxy_add_x_forwarded_for;

     # блок настроек необходимый для проксирования websocket запросов
     proxy_set_header Upgrade $http_upgrade;
     proxy_set_header Connection $http_connection;
     proxy_cache_bypass $http_upgrade;
     proxy_http_version 1.1;

     proxy_pass        http://127.0.0.1:10000/;

  }
#для блокировки swagger'a, интернал рестов, дебаг рестов

  if ($request_uri ~* (swagger|internal|debug)) {
    set $block_me_now A;
  }

  if ($remote_addr != xx.xx.xx.xx) {  # добавить разрешенный адрес
    set $block_me_now "${block_me_now}B";
  }

  if ($block_me_now = AB) {
    return 403;
    break;
  }

  location ~ .*(/internal/.*) {
     deny all;
  }

  location ~ .*(/debug/.*) {
    deny all;
  }

  location ~ .*(/swagger/.*) {
    deny all;
  }

  location ~ .*(/document-api/public/.*) {
    deny all;
  }
}
```

Настройка сим-линк

После создания необходимо конфигурации приведенной выше, необходимо перейти в директорию */etc/nginx/sites-enabled* и настроить сим-лик.

Для этого потребуется выполнить следующую команду:

Создание сим-линк

```
sudo ln -s ../sites-available/<имя файла с конфигурацией> <имя файла с конфигурацией>
```

Применение новой конфигурации Nginx

После добавления файла конфигурации в */etc/nginx/sites-available* и настройки сим-линк в */etc/nginx/sites-enabled* требуется проверить ее на синтаксические ошибки, если они отступают, выполнить перегрузку хвостового Nginx для их активации:

Применение конфигурации

```
sudo nginx -t                    # Проверка синтаксиса файла конфигурации
sudo nginx -s reload             # Перезапуск Nginx
```

Nginx в docker-compose

Отредактировать директиву *server_name* для каждого домена (типовой конфиг будет предоставлен) в *docker-compose/services /nginx/conf.d/default.conf* - этот файл смонтирован в контейнер с хоста через volume.

Пример конфигурации Nginx в контейнере

```
server {
    listen 80;

    client_max_body_size 1000m;

    server_name
    ~^[a-z]+?-[a-z]+?\.domain\.ru$
    .main-dev.sado.corp.artsofte.ru
    client-lk-browser-dev.sado.ru;

    location /sso/ {
        proxy_pass http://idsrv-api/;
    }

    location /workflow-api/ {
        proxy_pass http://workflow-api/;
    }

    location /profile-api/ {
        proxy_pass http://profile-api/;
    }

    location /client-document-api/ {
        proxy_pass http://client-document-api/;
    }

    location /action-api/ {
        proxy_pass http://action-api/;
    }

    location /application-api/ {
        proxy_pass http://application-api/;
    }

    location /orchestrator-api/ {
        proxy_pass http://orchestrator-api/;
    }

    location /catalog-api/ {
        proxy_pass http://catalog-api/;
    }
```

Настройка сервисов

Общие настройки окружения микросервисов находятся в файле */docker-containers/services/.env.*

Пример .env

```
# Реджестри
REGISTRY=artsofte

# Версии включаемых образов
NOCODE_VER=nocode-2.11.0

# Адрес сборщика логов
LOKI_URL=http://<loki-server>:3100/loki/api/v1/push

# Настройки окружения сервисов
IdentityServer__Authority=http://idsrv-api/
IdentityServer__UseRevocationJwtAccessToken=true

ApplicationServices__MonitoringService__Mock__Enable=false
ApplicationServices__ConfirmationService__Host=http://cs-api
ApplicationServices__NotificationService__Host=http://ns-api

HttpServiceHosts__ClientDocumentHost=http://client-document-api
HttpServiceHosts__FileServerHost=http://file-api
HttpServiceHosts__ExternalHost=http://external-api
HttpServiceHosts__GenerateDocumentHost=http://generate-document-api
ProfileApiHttpConnectionServiceSettings__ProfileApiHostAddress=http://profile-api
HttpServiceHosts__TemplateDocumentRenderHost=http://template-render-api

# Настройки подключения к реляционной БД
Dal__AutoMigration=true
Dal__ServerType=postgresql
Dal__DapperCoreConnectionSetting__Host=postgres
Dal__DapperCoreConnectionSetting__Port=5432
Dal__DapperCoreConnectionSetting__UserID=postgres
Dal__DapperCoreConnectionSetting__Password=rw_dev
Dal__DapperCoreConnectionSetting__Pooling=true
Dal__DapperCoreConnectionSetting__MaxPoolSize=30

Logger__OurLogLevel=Information
Logger__LogLevelFile=Information
Logger__LogLevelConsole=Information
Logger__SystemLogLevel=Warning
Logger__LoggingMiddlewareEnabled=true
Logger__UnwantedForLogPart[0]=swagger

TenantEnvironment__IsSearchOrchestrator=false

ConfirmOperation__Default__IsUseDbSettings=true
Debug__IsEnable=false
```

В каждом каталоге сервиса, присутствует файл настроек */docker-containers/services/<SERVICE_NAME>/.env*. В нём указаны переменные среды, уникальные для данного микросервиса. Например:

Пример файла настроек микросервиса

```
# Имя реляционной базы данных микросервиса
Dal__DapperCoreConnectionSetting__Database=document-search-api
```

В поставке, уже указаны реквизиты, нужные для подключения к БД в контейнере, на одном хосте с сервисами.

В случае, когда БД размещается на отдельном хосте, необходимо изменить эти реквизиты (хост, порт, логи, пароль, имя пользователя). Сервисы сами создают БД при первом старте.

Все микросервисы бекенда подключаются к PostgreSQL.

Примечание: База данных PostgreSQL, размещается в Docker-контейнере, только в варианте тестового развертывания. В продакшене, БД размещается и устанавливается на отдельном сервере.

В файле */docker-containers/services/docker-compose.yml* указан список контейнеров с конфигурациями запуска. В нём можно использовать переменные для тегов.

Пример указания переменной тега

```
provider-lk-browser:
  image: artsofte/sado-provider-lk-browser:${NOCODE_VER}
  networks:
    - network
  volumes:
    - "./provider-lk-browser/settings/:/usr/share/nginx/html/settings"
  restart: unless-stopped
```

В файле */docker-containers/services/.env* можно задать переменную тегов.

Пример указания тегов версий

```
NOCODE_VER=nocode-2.11.0
```

Настройка почты в микросервисе уведомлений

Важно: До старта docker-compose необходимо установить настройки SMTP/SMPP сервиса и реквезиты почты/телефона администратора продукта (провайдера).

В */docker-containers/services/ns-api/DefaultSetting/appsettings.DefaultSetting.json* прописываем настройки для подключения к почтовому серверу (Host, Port, User, Password, From).

Настройки ns-api

```
{
  "AvailableSendingTimeSettings": null,
  "Smtp": {
    "Enable": true,
    "Anonymous": false,
    "Host": "smtp.mail.ru",
    "Port": 587,
    "User": "sado@mail.ru",
    "Password": "",
    "From": "sado@mail.ru",
    "CheckCertificateRevocation": false,
    "SslMode": 1,
    "ContextualKey": ""
  },
  "Smpp": {
    "IsMocked": false,
    "Enable": false,
    "Host": "",
    "Password": "",
    "UserName": "",
    "FromName": "",
    "ContextualKey": ""
  },
  "ContextualSmtp": [],
  "ContextualSmpp": [],
  "WebMessengerConfig": {
    "Enable": true,
    "MockAllSms": true,
    "Webhook": "",
    "FromName": ""
  }
}
```

В */docker-containers/services/profile-api/.env* прописываем почту для отправки пароля учетной записи провайдера:

Настройки profile-api

```
ProviderCreator:Email=confirm@gmail.com
```

Настройка смс-шлюза

В файле */docker-containers/infra/kannel/kannel.conf* добавить реквизиты для подключения к смс точке (данную настройку, можно произвести позже).

Пример настроек kannel.conf

```
group = core
admin-port = 13000
smsbox-port = 13001
admin-password = bar

#### Настрйока для провайдера SMS, указаны реквизиты для sms-центр####
group = smsc
smsc = smpp
smsc-id = smsc
host = smpp2.smsc.ru  # заменить на свои
port = 3700  # заменить на свои
smsc-username = # логин для подключения к smpp точке
smsc-password = # пароль для подключения к smpp точке

system-type = "ISO"
interface-version = 34
source-addr-autodetect = yes
source-addr-ton = 5
source-addr-npi = 1
dest-addr-ton = 1
dest-addr-npi = 1
validityperiod = 1440
transceiver-mode = true
receive-port = 0
enquire-link-interval = 60
wait-ack-expire = 0
max-pending-submits = 300
throughput = 100
max-sms-octets = 140

group = smsbox
bearerbox-host = bearerbox
sendsms-port = 13016
global-sender = 13016

group = sendsms-user
username = "smsuser"   # логин для авторизации сервиса в smsbox
password = "fdgsdfg5464"  # пароль для авторизации сервиса на smsbox
default-smsc = smpp
max-messages = 10
concatenation = 1
```

При смене настроек подключения kannel.conf, нужно перезапустить контейнеры kannel и smsbox.

Перезапуск kannel и smsbox

```
docker-compose up -d --force-recreate kannel smsbox
```

Docker-образ kannel входит в комплекте поставки.

Настройка системы мониторинга

Систему мониторинга можно развернуть как на сервере с приложением, так и на отдельном.

На сервере, где он будет располагаться в корне создать каталог */docker-containers* (если его нет), куда распаковать содержимое архива *monitoring.tar.gz.*

Распаковка конфигурации

```
sudo mkdir /docker-containers
sudo tar -xvf monitoringi.tar.gz. -C /docker-containers
```

Полученная структура директорий:

Структура директорий

```
/docker-containers
└── ./monitoring
    ├── grafana/
    │         ├──data
    │   │     └──dashboards
    │   ├──datasources
    │   │       └──ds.yml
    │   └──dashboard.yml
    ├── loki/
    │   ├── prometheus/
    │   ├── config/
    │   │   └── prometheus.yml
    │   └── data/
    └── docker-compose.yml
```

Создать пользователей для сервисов и назначить им uid'ы:

Создание пользователей

```
useradd -r -s /bin/false grafana
useradd -r -s /bin/false loki
useradd -r -s /bin/false prometheus
usermod -u 10001 loki
groupmod -g 10001 loki
usermod -u 472 grafana
groupmod -g 472 grafana
```

Находясь в директории monitoring*i/* назначить владельцами loki-data/ и grafana-data/ созданных пользователей:

Назначение владельцев директорий

```
cd /docker-containers/monitoring
chown -R loki:loki loki/
chown -R grafana:grafana grafana/
chown -R nobody:nogroup prometheus/
```

В */docker-containers/monitoring/config/prometheus.yml* в блоке с параметром *job_name: node* и добавить все необходимые хосты с портом 9100 в блоке *static_configs:*

Подключение машин к prometheus

```
 - job_name: node
   static_configs:
     - targets: ['<хост1>:9100']
     - targets: ['<хост2>:9100']
     - targets: ['<хост3>:9100']
```

В директории monitoring/ выполнить запуск сервисов:

Запуск сервиса агрегации

```
cd /docker-containers/monitoring
docker-compose up -d
```

На сервере с приложением файле */docker-containers/services/.env указать* строку подключения к локи-серверу:

Указание строки подключения

```
LOKI_URL=http://<loki-server>:3100/loki/api/v1/push
```

Запуск

Создание сети

Создать виртуальную сеть в докере для работы сервисов (этот пункт опционально - сеть можно оставить дефолтную и убрать этот параметр из docker-compose.yml, либо указать свое название сети для компоуза и всех сервисов).

Создание сети

```
docker network create network
```

Запуск инфраструктурных микросервисов

Перейти в каталог */docker-compose/infra* и запустить инфраструктурные сервисы, необходимые для работы приложений.

Запуск инфраструктуры

```
cd /docker-containers/infra
docker-compose up -d
```

Увеличение максимального количества подключений PostgreSQL

В случае необходимости в настройках PostgreSQL возможно увеличить максимальное кол-во подключений.

Для этого:

Если PostgreSQL развернут процессом:

1. Открыть файл */var/lib/postgres/postgresql.conf*
2. Вместо *max_connections = 100*, указать *max_connections = 1000*
3. Перезапустить БД командой

   Перезапуск процесса PostgreSQL

   ```
   sudo systemctl restart postgresql
   ```

Если в контейнере:

1. Открыть файл */docker-containers/sado-infra/postgres/postgresql.conf*
2. Вместо *max_connections = 100*, указать *max_connections = 1000*

3. Перейти в директорию инфраструктуры и перезапустить БД

Перезапуск контейнера PostgreSQL

```
cd /docker-containers/infra
docker-compose up -d --force-recreate postgres
```

Запуск приложения

Запустить образы приложений из директории */docker-compose/services*

Запуск мискросервисов

```
cd /docker-compose/services
docker-compose up -d
```

Проверка

Проверяем запущенные контейнеры

Проверить, что контейнеры не в рестарте, имеют статус *Up* больше 20 секунд. Команда выполняется без привязки к каталогу:

Вывод информации о запущенных контейнерах

```
docker ps
```

Вывод при исправной работе

```
CONTAINER ID   IMAGE                                          COMMAND                  CREATED          STATUS
PORTS                                      NAMES
cd5b71d2377f   artsofte/sado-orchestrator-api:sado-dev        "dotnet Api.dll"         3 minutes ago    Up 3 minutes    80
/tcp, 443/tcp                              sado-dev_orchestrator-api_1
29a6d66fc09c   artsofte/sado-profile-api:sado-dev             "dotnet Api.dll"         5 minutes ago    Up 5 minutes    80
/tcp, 443/tcp                              sado-dev_profile-api_1
d7665dce2d5f   artsofte/sado-workflow-api:sado-dev            "dotnet Api.dll"         5 minutes ago    Up 5 minutes    80
/tcp, 443/tcp                              sado-dev_workflow-api_1
ccdc2e2a56cd   artsofte/sado-reserve-copy-api:sado-dev        "dotnet Api.dll"         10 minutes ago   Up 10 minutes
80/tcp, 443/tcp                            sado-dev_reserve-copy-api_1
6a4598c052cc   artsofte/sado-client-document-api:sado-dev     "dotnet Api.dll"         11 minutes ago   Up 11 minutes
80/tcp, 443/tcp                            sado-dev_client-document-
api_1
e942a3f11b43   artsofte/sado-async-operation-api:sado-dev     "dotnet Api.dll"         15 minutes ago   Up 15 minutes
80/tcp, 443/tcp                            sado-dev_async-operation-
api_1
3f01a2b748fb   artsofte/sado-action-api:sado-dev              "dotnet Api.dll"         15 minutes ago   Up 15 minutes   80
/tcp, 443/tcp
```

Вывод при наличии ошибок

```
CONTAINER ID   IMAGE                                              COMMAND              CREATED           STATUS
PORTS                                                                                                    NAMES
cd5b71d2377f   artsofte/sado-orchestrator-api:sado-dev                  "dotnet Api.dll"       3 minutes ago       Restarting (139) 18
seconds ago       80/tcp, 443/tcp                                                                             sado-dev_orchestrator-
api_1
29a6d66fc09c   artsofte/sado-profile-api:sado-dev                      "dotnet Api.dll"       5 minutes ago       Up 5
minutes                  80/tcp, 443/tcp                                                                      sado-dev_profile-
api_1
d7665dce2d5f   artsofte/sado-workflow-api:sado-dev                     "dotnet Api.dll"       5 minutes ago       Up 5
minutes                  80/tcp, 443/tcp                                                                      sado-
dev_workflow-api_1
ccdc2e2a56cd   artsofte/sado-reserve-copy-api:sado-dev                 "dotnet Api.dll"       10 minutes ago      Up 10
minutes                 80/tcp, 443/tcp                                                                       sado-dev_reserve-
copy-api_1
6a4598c052cc   artsofte/sado-client-document-api:sado-dev              "dotnet Api.dll"       11 minutes ago      Up 11
minutes                 80/tcp, 443/tcp                                                                       sado-dev_client-
document-api_1
e942a3f11b43   artsofte/sado-async-operation-api:sado-dev              "dotnet Api.dll"       15 minutes ago      Restarting (139)
57 seconds ago       80/tcp, 443/tcp                                                                          sado-dev_async-
operation-api_1
3f01a2b748fb   artsofte/sado-action-api:sado-dev                       "dotnet Api.dll"       15 minutes ago      Up 15
minutes                 80/tcp, 443/tcp
```

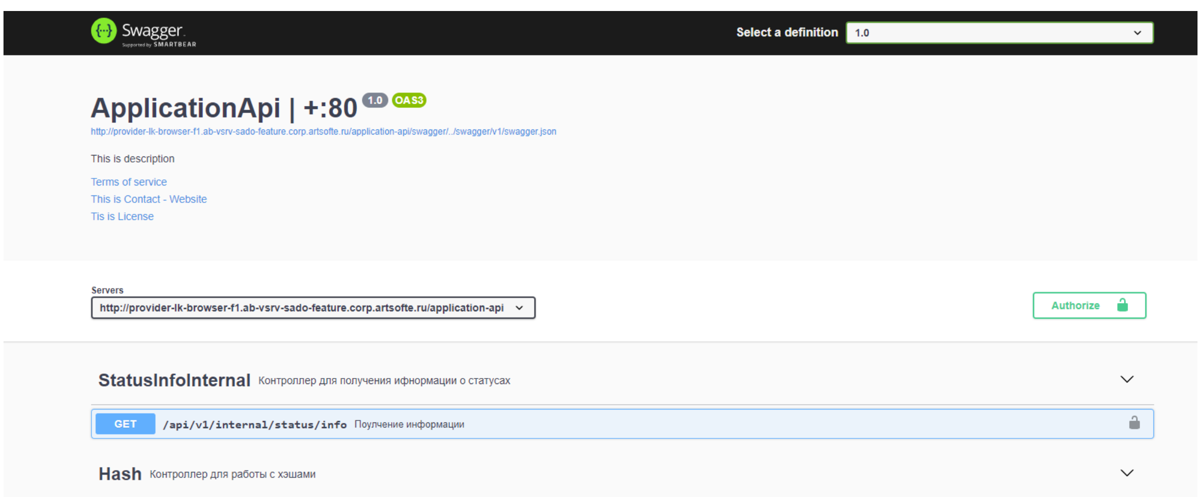
Далее войти в кабинет провайдера по дольмену указанному в *server_name* хостового Nginx, например, provider-lk-browser-dev.sado.ru.

Стартовая страница должна успешно загрузиться, без ошибок.

Доступность Swagger

Если Swagger включен на хостовом Nginx и в */docker-containers/services/.env* (SwaggerDocs:Enable=true), можно проверить доступность каждого из сервисов бекенда, выполнив http://provider-lk-browser-dev.sado.ru/action-api/swagger (вместо action-api, может быть любой другой сервис).

Должна загрузится страница следующего вида:



Скрипт опроса жизнеспособности сервисов

Для проверки жизнеспособности сервисов необходимо создать создать файл в *health_check.sh* директории */docker-containers/services/* и поместить в него код приведенный ниже.

health_check.sh

```
#!/bin/bash

# Алиасы провайдера и клиента
PROVIDER_ALIAS="provider"
CLIENT_ALIAS="client-client"

# Формирование списка сервисов и определение порта nginx
SERVICES=$(docker-compose config --services | sed 's/idsrv-api/sso/g; s/nginx//g; s/provider-lk-browser//g; s/client-lk-browser//g')
NGINX_PORT=$(docker-compose port nginx 80 | cut -d: -f2)
echo -e "\033[43m\nNGINX_PORT: ${NGINX_PORT}\033[0m\n"

# Вывод статуса каждого сервиса
echo -e "\033[43m\nПроверка сервисов:\033[0m\n"
for SERVICE in $SERVICES; do

  EXTERNAL_DOMAIN="localhost:${NGINX_PORT}/${SERVICE}/internal/health"
  PROVIDER_DOMAIN="http://${PROVIDER_ALIAS}.${EXTERNAL_DOMAIN}"
  CLIENT_DOMAIN="http://${CLIENT_ALIAS}.${EXTERNAL_DOMAIN}"

  response=$(curl -s -o - ${PROVIDER_DOMAIN})
  status=$(echo "$response" | grep -o '"status": *"[^"]*"' | awk -F ':"' '{print $2}' | tr -d '"' | head -n 1) # Парсинг статуса

  if [ "$status" == "" ]; then # Если сервис доступен только для клиента и статус придет пустой, отпраляем запрос еще раз
    response=$(curl -s -o - ${CLIENT_DOMAIN})
    status=$(echo "$response" | grep -o '"status": *"[^"]*"' | awk -F ':"' '{print $2}' | tr -d '"' | head -n 1)
  elif [ "$status" == "Healthy" ]; then
      echo -e "\033[32m$SERVICE ($status)\033[0m"
  else
      echo -e "\033[31m$SERVICE ($status)\033[0m"
      response_body=$(curl -s -o - ${PROVIDER_DOMAIN})
      echo -e "$response_body\n"
  fi
done

# Запрос разметки у фронта
echo -e "\n\033[43mHTML разметка кабинетов:\033[0m\n"

response=$(curl -s http://${PROVIDER_ALIAS}.localhost:${NGINX_PORT}/)
title=$(echo "$response" | grep -o '<title>[^<]*</title>' | sed -e 's/<title>//' -e 's/<\/title>//')

if [ "$title" = "Кабинет администратора" ]; then
  echo -e "\033[32mЗаголовок страницы: $title (ожидаемый)\033[0m"
else
  echo -e "\033[31mЗаголовок страницы: $title (не соответствует ожидаемому)\033[0m"
fi

response=$(curl -s http://${CLIENT_ALIAS}.localhost:${NGINX_PORT}/)
title=$(echo "$response" | grep -o '<title>[^<]*</title>' | sed -e 's/<title>//' -e 's/<\/title>//')

if [ "$title" = "Кабинет клиента" ]; then
  echo -e "\033[32mЗаголовок страницы: $title (ожидаемый)\033[0m"
else
  echo -e "\033[31mЗаголовок страницы: $title (не соответствует ожидаемому)\033[0m"
fi
```

Необходимо задать значения четырех переменных:

PROVIDER_ALIAS - алиас кабинета провайдера
CLIENT_ALIAS - алиас кабинета клиента

После этого сделать созданный файл исполняемым, выполнив следующую команду:

Выдача разрешения на выполнение

```
sudo chmod +x ./health_check.sh
```

Запуск проверки выполняется из директории со скриптом:

Запуск скрипта

```
sudo ./health_check.sh
```

Создание провайдера-администратора

После того, как система успешно запустилась, необходимо создать аккаунт провайдера-администратора. Для этого необходим доступ к internal-рестам.

Создание провайдера-администратора curl

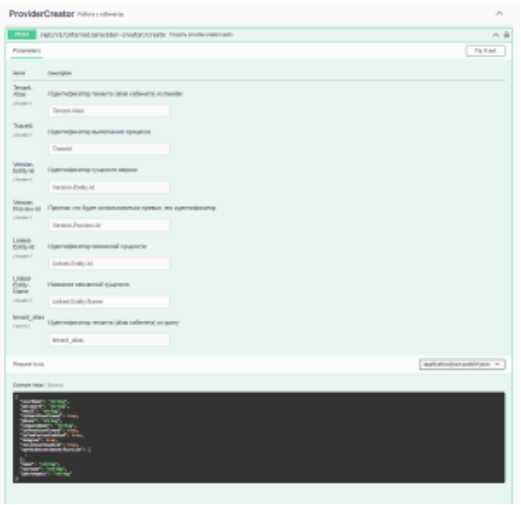
```
curl -X 'POST' \
  'http://{endpoint}/profile-api/api/v1/internal/provider-creator/create' \
  -H 'accept: */*' \
  -H 'Content-Type: application/json-patch+json' \
  -d '{
  "userName": "79000000000",
  "password": "k@l72798",
  "email": "sample@sample.ru",
  "isEmailConfirmed": true,
  "phone": "79000000000",
  "companyName": "COMPANY NAME",
  "isPhoneConfirmed": true,
  "isTwoFactorEnabled": true,
  "isAgree": true,
  "isLockoutEnabled": false,
  "authCabinetIdentifierList": [
    1
  ],
  "name": "Admin Name",
  "surname": "Admin Surname",
  "patronymic": "Admin Patronymic"
}'
```

Название	Тип	Описание	Ограничения
userName	string	Логин провайдера	
password	string	Пароль провайдера	Должен соответствовать правилам пароля. Их можно получить через **GET http://provider.{HOST}/sso/api/v1/public/setting-password/rules**
email	string	Почтовый адрес провайдера	Должен быть валидным почтовым адресом
isEmailConfirmed	bool	Является ли почтовый адрес подтвержденным	
phone	string	Номер телефона провайдера	Должен состоять только из чисел
companyName	string	Название компании	

isPhoneConfirmed	bool	Является ли номер телефона подтвержденным	
isTwoFactorEnabled	bool	Включена ли двухфакторная авторизация	
isAgree	bool	Приняты ли условия использования	
isLockoutEnabled	bool	Включена ли блокировка пользователя	
authCabinetIdentifierList	enum[]	Массив с идентификаторами, по которым необходимо осуществлять поиск пользователя	1 - поиск осуществляется по логину пользователя 2 - поиск осуществляется по телефону пользователя 3 - поиск осуществляется но почтовому адресу пользователя
name	string	Имя пользователя	
surname	string	Фамилия пользователя	
patronymic	string	Отчество пользователя	

Также создать провайдера-администратора можно через Swagger сервиса profile-api.

Необходимо перейти на http://provider.{HOST}/profile-api/swagger/index.html#/ProviderCreator/post_api_v1_internal_provider_creator_create и отправить запрос.



Пример запроса

```
{
  "userName": "79000000000",
  "password": "k@l72798",
  "email": "sample@sample.ru",
  "isEmailConfirmed": true,
  "phone": "79000000000",
  "companyName": "COMPANY NAME",
  "isPhoneConfirmed": true,
  "isTwoFactorEnabled": true,
  "isAgree": true,
  "isLockoutEnabled": false,
  "authCabinetIdentifierList": [
    1
  ],
  "name": "Admin Name",
  "surname": "Admin Surname",
  "patronymic": "Admin Patronymic"
}
```

Ответом запроса будет статус-код 200. Далее можно авторизоваться в кабинете провайдера (https://provider.{HOST}/account/login) под созданной учетной записью.

Смена настроек smtp/smpp в кабинете провайдера