Цифровая платформа
«Управление в пространстве» (ЦП УВП)

версия 0.38

РУКОВОДСТВО ПО ИНСТАЛЛЯЦИИ И РАЗВЕРТЫВАНИЮ

Содержание

1 Перечень терминов, сокращений и обозначений 3
2 Назначение и основные возможности 4
2.1 Ключевые особенности 4
2.2 Возможности платформы 4
3 Установка платформы 5
3.1 Системные требования 5
3.2 SSL-сертификаты 6
3.3 Завершение установки 8
3.4 Установка с использованием docker compose 8
3.5 Переменные окружения 13
3.6 Песочница и удаленный запуск расчетов 13
3.7 Прокси-сервер для клиентского приложения 16

1 Перечень терминов, сокращений и обозначений

Сокращение	Определение
БД	База данных
Платформа, ЦП УВП	Цифровая платформа «Управление в пространстве»
API	Application Programming Interface – программный интерфейс приложения
LDAP	англ. Lightweight Directory Access Protocol – протокол доступа к службе каталогов
SSL	англ. Secure Sockets Layer – протокол безопасности, создающий зашифрованное соединение между веб-сервером и веб-браузером
SSL-сертификат	Цифровой сертификат, удостоверяющий подлинность веб-сайта и позволяющий использовать зашифрованное соединение

2 Назначение и основные возможности

Цифровая платформа «Управление в пространстве» – платформа, автоматизирующая процесс загрузки, анализа и интерпретации данных на основе методов математического моделирования.

Российская аналитическая low-code BI-платформа «Управление в пространстве» со встроенным искусственным интеллектом и геоаналитикой, которые используются для поддержки принятия управленческих решений. Используется программное обеспечение с открытым исходным кодом (не используются права третьих лиц, одним из основных преимуществ является свобода использования, модифицирования программы), полностью собственная разработка компании БИТ.

Включена в Единый реестр российских программ для ЭВМ и БД под рег. номером 7516. Может быть использована для импортозамещения таких продуктов как Power BI, Tableau, Qlik Sense, SPSS Statistics, ArcGIS Business Analyst.

BI-платформа «Управление в пространстве» в сравнении с другими BI-платформами на российском рынке имеет расширенные функциональные возможности в части решения задач 3D-картографического мониторинга, построения цифровых двойников, моделей и расчетов в пространстве, динамического моделирования, сценарного и целевого прогнозирования и формирования рекомендаций, low-code и визуального конструирования аналитических решений.

2.1 Ключевые особенности

- Платформа реализует комплексную аналитическую поддержку принятия решений с использованием технологий продвинутой аналитики на основе математического моделирования, геоаналитики, искусственного интеллекта и больших данных.
- Платформа обеспечивает визуальное конструирование аналитических приложений.

2.2 Возможности платформы

- Базовая аналитика:
    - сбор, загрузка и преобразование данных;
    - визуализация данных с использованием электронной картографии, таблиц и диаграмм, аналитических панелей.
- Продвинутая аналитика:
    - геоаналитика (продвинутая обработка пространственных данных, 3D-картография, геоинформационный мониторинг, пространственный анализ и пространственное моделирование);
    - математическое моделирование с использованием математической статистики и методов машинного обучения (ML);
    - сценарное многовариантное прогнозирование, оптимизационное и целевое управление;
    - предписывающая аналитика на основе генерации рекомендаций на естественном языке с использованием продукционного подхода.

3 Установка платформы

3.1 Системные требования

3.1.1 Аппаратная часть

3.1.1.1 Сервер

Важно! Конфигурации могут меняться в зависимости от предполагаемой инфраструктуры.

- Процессор: 8 ядер, 2.30 ГГц и выше;
- Оперативная память: 8 ГБ и выше;
- Жесткий диск: от 20 Гб. *Размер определяется объемом базы данных*;
- Сетевой интерфейс: 1 Гбит/сек.

Для каждой установки необходимо определять объемы оперативной памяти / ядер / жесткого диска исходя из количества компонент, которые будут установлены на север, а также количества пользователей.

Указанные характеристики подходят для установки приложения и кэширующей БД для **25 пользователей**. Если необходимо дополнительно разместить на этом же сервере:

- Сервис расчетов R: ОЗУ + 2 Гб (определяется количеством расчетов), жесткий диск + 5 Гб;
- Сервис подготовки геоданных Martin: ОЗУ + 4 Гб, жесткий диск + 5 Гб (определяется объемами картографических данных);
- БД Postgres: ОЗУ +4 Гб, жесткий диск +10 Гб (определяется объемами данных).

3.1.1.2 Рабочая станция

- Процессор: 4 ядра, 2.40 ГГц и выше;
- Оперативная память: 4 ГБ и выше;
- Сетевой интерфейс: от 20 Мбит/сек.

Дополнительно:

- Монитор 24" и более (разрешение 2560x1440) для оптимального отображения визуализаторов;
- Дискретная видеокарта для работы с большим количеством картографических визуализаторов.

3.1.2 Программная часть

3.1.2.1 Сервер

- Операционная система linux (Ubuntu, AstraLinux – debian 7+). Рекомендуется Ubuntu 20 Server LTS. В случае Astra-Linux рекомендуется версия 1.7+;
- Операционная система linux (CentOS, Oracle Linux – redhat 7.9+). Рекомендуется CentOS 7;
- БД приложения: PostgresSQL 15 + PostGIS 3.2;
- Кэширующая БД: Redis 6.0.7;
- docker + docker-compose – в случае работы в контейнерах;
- python 3.10, nginx 1.23.1 – в случае работы без контейнеров.

Опционально:

- Сервис подготовки геоданных Martin (https://github.com/maplibre/martin);
- Сервис расчетов R (https://www.r-project.org/, https://hub.docker.com/_/r-base).

3.1.2.2 Рабочая станция

- Веб-браузер (лучше использовать последнюю доступную версию):
  - Google Chrome 90+;
  - Mozilla Firefox 90+;
  - Edge 101+;
  - Яндекс.Браузер 21.6.3.

3.2 SSL-сертификаты

Для того чтобы платформа могла работать с использованием HTTPS-соединения, необходимо в прокси-сервере, на котором размещено клиентское приложение (frontend), добавить сертификаты. Для этого необходимо выписать SSL-сертификаты на сервер, на котором будет размещаться приложение.

3.2.1 Получение сертификатов

Получение сертификатов в центре сертификации остается вне этого руководства. Для получения самоподписанного сертификата для сервера можно использовать следующий скрипт команд (для сервера Ubuntu):

generate.sh

```
#!/bin/sh

if [ "$#" -ne 1 ]
then
  echo "Must supply a domain"
  return 1
fi

DOMAIN=$1

openssl genrsa -des3 -out ca.key 2048
openssl req -x509 -new -nodes -key ca.key -sha256 -days 1825 -out ca.pem

openssl genrsa -out $DOMAIN.key 2048
openssl req -new -key $DOMAIN.key -out $DOMAIN.csr

cat > $DOMAIN.ext << EOF
authorityKeyIdentifier=keyid,issuer
basicConstraints=CA:FALSE
keyUsage = digitalSignature, nonRepudiation, keyEncipherment, dataEncipherment
subjectAltName = @alt_names
[alt_names]
DNS.1 = $DOMAIN
EOF

openssl x509 -req -in $DOMAIN.csr -CA ca.pem -CAkey ca.key -CAcreateserial \
-out $DOMAIN.crt -days 825 -sha256 -extfile $DOMAIN.ext
```

Для вызова скрипта необходимо указать имя домена, для которого будет выпущен сертификат:

```
$ . generate.sh domain.name
```

Во время работы скрипта необходимо указать некоторые параметры для выпуска сертификатов.

После этого необходимо забрать полученные файлы:

```
domain.name.crt
domain.name.key
```

3.2.2 Запуск приложения в docker или docker-compose с использованием SSL-сертификатов

Для работы приложения необходимо удостовериться, что версия платформы была собрана с возможностью подключения SSL-сертификатов. Для этого необходимо проверить в контейнере с клиентским приложением (frontend) конфигурационный файл и убедиться, что в нем указаны сертификаты для работы по https.

Заходим в контейнер:

```
$ docker exec -it vismind-frontend /bin/sh
```

Проверяем содержимое файла конфигурации **nginx**:

```
$ cat /etc/nginx/conf.d/default.conf | grep "443 http2 ssl"
```

Если команда возвращает строку вида:

```
listen       443 http2 ssl;
```

значит, клиент настроен, и можно переходить к шагу замены сертификатов.

Если ответ пустой или строка начинается с символа # – необходимо обратиться к разработчику для получения дистрибутива с активированной функцией SSL-соединения.

3.2.2.1 Замена сертификатов

Если приложение настроено на работу с SSL, необходимо загрузить корректные сертификаты, выпущенные для сервера:

```
domain.name.crt
domain.name.key
```

Для этого необходимо в файле **docker-compose.yml** передать внутрь контейнера корректные файлы и пробросить наружу 443 порт. Проброс портов предназначен для предоставления внешним пользователям доступа к некоторым внутренним службам, используя единственный внешний IP-адрес.

Пример конфигурации:

```
  vismind-frontend:
    container_name: vismind-frontend
    image: vismind-frontend:latest
    ports:
      - 8002:443
    volumes:
      - ./files/static:/usr/share/nginx/media/static
   - ./domain.name.crt:/etc/ssl/certs/domain.crt:ro;
   - ./domain.name.key:/etc/ssl/private/domain.key:ro;
    depends_on:
      - vismind-backend
```

```
networks:
  - vismind-nt
restart: unless-stopped
```

После чего необходимо перезапустить приложение. После перезапуска приложение будет доступно на 443 порту по протоколу HTTPS.

3.3 Завершение установки

После завершения установки необходимо проверить, что приложение запустилось по выбранному адресу:

- http://localhost:8001 – если не изменялись порт и не используется SSL;
- https://locahost:8002 – если во время установки были добавлены SSL-сертификаты.

При использовании «чистой» установки по умолчанию настроена учетная запись:

```
login: admin
password: admin
```

После входа в приложение можно изменить пароль для Администратора и создать новых пользователей.

3.4 Установка с использованием docker compose

3.4.1 Поставка

Платформа поставляется в виде tar-архива, который содержит все необходимые компоненты для начала работы:

1. Docker-образы платформы.
2. Бэкап БД репозитория.
3. Файл лицензии.
4. Скрипты установки и запуска платформы.

Важно! В данной инструкции не рассматривается установка сервиса подготовки геоданных сервера **Martin** и его прокси-сервера **nginx proxy**, а также установка сервиса расчетов **R**.

3.4.2 Системные требования

Перед началом работы необходимо установить на сервер:

1. docker – Инструкция по установке Docker.
2. docker compose – с июля 2023 считается устаревшим и является частью Docker. Если версия Docker установлена без него – можно установить по старому – Инструкция по установке docker compose

На сервере должен быть доступ в Интернет для загрузки образов PostgreSQL и Redis. Если доступа до docker hub на сервере нет – необходимо загрузить эти образы самостоятельно.

1. PostgreSQL – **postgis/postgis:15-3.3-alpine**
2. Redis – **redis:6.0.7-alpine**

3.4.3 Установка

Установка платформы использует Docker для запуска необходимых компонентов в контейнерах с помощью **docker compose**. Сам процесс установки автоматический и запускает платформу с использованием внутренней БД PostgreSQL 15.

Для установки необходимо выполнить команды:

```
$ tar -xvzf uvp_fortum.tar.gz
$ cd fortum-customer
$ sudo chmod +x install.sh
$ . install.sh
```

По окончанию установки платформа будет доступна по адресу http://localhost:8001

3.4.4 Запуск и остановка

Для запуска приложения выполнить скрипт:

```
$ . start_application.sh
```

Для остановки приложения выполнить скрипт:

```
$ . stop_application.sh
```

3.4.5 Начало работы

Для начала работы необходимо открыть страницу авторизации по адресу http://localhost:8001.

Логин по умолчанию: `admin`

Пароль по умолчанию: `admin`

3.4.6 Конфигурация приложения

По умолчанию платформа запускается с использованием соединения с базой данных репозитория, которая создалась во время установки. Она расположена в контейнере `vismind-postgresql`.

Вся внутренняя конфигурация серверной части платформы может быть изменена с помощью переменных окружения. Количество доступных переменных больше, чем представлено ниже. Ниже представлены переменные, нужные для настройки подключения к базе данных приложения и настройки аутентификации.

Переменные окружения задаются для контейнера **vismind-backend** в файле ./distr/docker-compose.yaml:

```
...
  vismind-backend:
    container_name: vismind-backend
    image: vismind-backend:0.36-fortum
    environment:
      # repository
      - VISMIND_REPOSITORY_HOST=vismind-postgresql
      - VISMIND_REPOSITORY_PORT=5432
      - VISMIND_REPOSITORY_NAME=repo_vismind
      - VISMIND_REPOSITORY_USER_LOGIN=
      - VISMIND_REPOSITORY_USER_PASSWORD=
      # redis
      - VISMIND_REDIS_HOST=vismind-redis
      - VISMIND_REDIS_PORT=6379
      - VISMIND_REDIS_DB=1
      - VISMIND_REDIS_PASSWORD=
      # LDAP
      - VISMIND_LDAP_ALLOW_LOGIN=1
      - VISMIND_LDAP_HOST=
      - VISMIND_LDAP_PORT=389
```

```
      - VISMIND_LDAP_USE_SSL=0
      - VISMIND_LDAP_GET_INFO=ALL
      - VISMIND_LDAP_CLIENT_STRATEGY=SYNC
      - VISMIND_LDAP_AUTHENTICATION=NTLM
      - VISMIND_LDAP_USER_HOST=int.bittechno.ru
      - VISMIND_LDAP_DOMAIN_COMPONENT=
      - VISMIND_LDAP_USERS_PARTITION=
      - VISMIND_LDAP_USERNAME=sAMAccountName
      - VISMIND_LDAP_GROUPS=memberOf
      - VISMIND_LDAP_FIRST_NAME=givenName
      - VISMIND_LDAP_LAST_NAME=sn
      -                                              VISMIND_LDAP_EMAIL=mail
      -                               VISMIND_LDAP_DEFAULT_GROUP=Администраторы
      -                               VISMIND_AUTH_DEFAULT_GROUP=Администраторы
      -                                      VISMIND_LDAP_SYNC_GROUPS_WITH_AD=0
      # kerberos
      - VISMIND_KERBEROS_ALLOW_LOGIN=1
      - VISMIND_KERBEROS_SYSTEM_USER_LOGIN=
      - VISMIND_KERBEROS_SYSTEM_USER_PASSWORD=
    volumes:
      - ./files:/app/files
      - ./kerberos_certs/krb5.conf:/etc/krb5.conf:ro
      - ./kerberos_certs/krb5.keytab:/etc/krb5.keytab:ro
    depends_on:
      - vismind-redis
      - vismind-postgresql
    networks:
      - vismind-nt
    restart: unless-stopped
...
```

Если необходимо использовать внешнюю БД Redis или PostgreSQL – необходимо указать подключение к ним в переменных окружения.

Авторизация с помощью доменных учетных записей и сквозная авторизация с помощью доменных учетных записей по умолчанию включены на клиенте, но не будут работать, пока не будут включены в конфигурации серверной части. Для включения необходимо заменить конфигурацию в файле ./distr/docker-compose.yaml и перезагрузить приложение:

```
$ docker compose -f ./distr/docker-compose.yaml down && docker compose -f ./distr/docker-compose.yaml up -d
```

3.4.6.1 Описание параметров конфигурации

Все переменные окружения, задающие конфигурацию платформы, начинаются с префикса `VISMIND_`. Затем следует имя переменной из конфигурационного файла приложения. Если значение конфигурации не задать переменной окружения – будут использоваться значения по умолчанию.

Значения по умолчанию не заданы для соединений с внешними службами (Postgres, Redis, Ldap).

3.4.6.1.1 PostgreSQL 15 repository

Определяет параметры подключения к БД репозитория приложения.

```
VISMIND_REPOSITORY_HOST           # имя сервера PostgreSQL
VISMIND_REPOSITORY_PORT           # порт, по умолчанию 5432
VISMIND_REPOSITORY_NAME           # имя БД репозитория платформы
```

```
VISMIND_REPOSITORY_USER_LOGIN    # логин пользователя для доступа к репозиторию
VISMIND_REPOSITORY_USER_PASSWORD    # пароль пользователя для доступа к репозиторию
```

База данных **не создается в процессе установки**. Ее развертывание происходит из файла **empty_repository.backup**. При необходимости переноса на другой сервер можно свернуть БД из контейнера **vismind-postgresql** и перенести ее на другой сервер. После этого можно будет изменять параметры соединения и перезапускать приложение.

3.4.6.1.2 Redis in-memory database

Определяет параметры подключения к Redis для хранения состояний и in-memory расчетов приложения.

```
VISMIND_REDIS_HOST      # имя сервера Redis
VISMIND_REDIS_PORT      # порт, по умолчанию 6379
VISMIND_REDIS_DB        # номер бд в которой будут хранится данные
VISMIND_REDIS_PASSWORD  # пароль для доступа к Redis
```

БД Redis не хранит свое состояние, поэтому все временные данные, хранящиеся в ней, будут удалены при остановке контейнера.

3.4.6.1.3 LDAP Active Directory

Параметры подключения к серверам Active Directory для авторизации с использованием доменных учетных записей:

```
VISMIND_LDAP_ALLOW_LOGIN              # Использовать LDAP
VISMIND_LDAP_HOST                       # Адрес сервера LDAP. Можно указывать множество хостов через разделитель <;> (точка с запятой)
VISMIND_LDAP_PORT                     # Порт сервера LDAP
VISMIND_LDAP_USE_SSL                  # Использовать ssl
VISMIND_LDAP_GET_INFO                 # Чтение информации о сервере,
                                      # возможные значения:
                                      #   NONE
                                      #   DSA
                                      #   SCHEMA
                                      #   ALL

VISMIND_LDAP_CLIENT_STRATEGY          # Стратегии соединения,
                                      # возможные значения:
                                      #   SYNC
                                      #   ASYNC
                                      #   LDIF
                                      #   RESTARTABLE
                                      #   REUSABLE
                                      #   SAFE_SYNC
                                      #   SAFE_RESTARTABLE

VISMIND_LDAP_AUTHENTICATION           # Аутентификация, возможные значения:
                                      #   ANONYMOUS
                                      #   SIMPLE
                                      #   SASL
                                      #   NTLM

VISMIND_LDAP_USER_HOST                # Наименование домена (обязательный)
VISMIND_LDAP_DOMAIN_COMPONENT         # Домен, в котором записан пользователь. Пример: dc=int,dc=bittechno,dc=ru
```

```
VISMIND_LDAP_USERS_PARTITION              # Адрес до раздела с пользователями.
Пример: ou=Service Accounts,ou=Enterprise
                                          # (если оставить пустой, то поиск будет
происходить по всему домену)

                                          # Атрибуты LDAP для использования при
создании пользователя в платформе
VISMIND_LDAP_USERNAME                     # Логин пользователя
VISMIND_LDAP_GROUPS                          # Группы, которые надо проверить для
добавления в группы
VISMIND_LDAP_FIRST_NAME                   # Имя пользователя
VISMIND_LDAP_LAST_NAME                    # Фамилия пользователя
VISMIND_LDAP_EMAIL                                          # Почта пользователя
VISMIND_LDAP_DEFAULT_GROUP=Администраторы    # группа, в которую автоматически
попадет                 новый                  пользователь                 LDAP
VISMIND_AUTH_DEFAULT_GROUP=Администраторы    # группа, в которую автоматически
попадает          пользователь,          созданный          администратором
VISMIND_LDAP_SYNC_GROUPS_WITH_AD=0              # признак синхронизации групп
пользователя с теми группами, которые у него есть в LDAP
```

При использовании LDAP пользователь будет являться **внешним** по отношению к платформе. Внешнему пользователю нельзя изменить атрибуты, которые получаются при его подключении из LDAP. При первом подключении такого пользователя в платформе будет создана учетная запись, которая будет заполнена данными из LDAP. При настройках по умолчанию, пользователь будет добавлен в группу Администраторы.

3.4.6.2 Некоторые комментарии по kerberos-аутентификации

При входе в платформу, если последней была использована kerberos-аутентификация, пользователю будет открыта страница kerberos-аутентификации. При этом сама аутентификация происходит через некоторое время по окончанию загрузки всех ресурсов на странице. Если необходимо сменить тип авторизации, необходимо выйти из приложения. Для этого надо либо нажать «Выйти» в меню, либо перейти на страницу /logout.

В режиме инкогнито сквозная авторизация не работает, как и в случае если учетная запись пользователя, под которой осуществлен вход, не является доменной. В этом случае пользователю будет предложено ввести логин и пароль средствами браузера. В этом окне необходимо указывать логин и пароль с именем домена, например: MYDOMAIN\Username.

3.4.7 Установка клиента Oracle Database

Для обеспечения работы с базой данных Oracle необходимо на сервере, где будет развернуто приложение, провести установку клиента:

1. Скачать последнюю версию клиента Oracle с сайта: https://www.oracle.com/ru/database/technologies/instant-client/linux-x86-64-downloads.html
2. Распаковать zip-архив в папку /opt на сервере, где будет запущен контейнер **vismind-backend**.
3. Перейти в папку и добавить символьную ссылку: `ln -s . lib`
4. Смонтировать папку клиента при запуске контейнера, добавив опцию для **docker start**:

   `--volume=/opt/instantclient_21_1:/opt/oracle`

3.5 Переменные окружения

Приложение при старте проверяет переменные окружения. Для того чтобы переменная из конфигурационного файла была считана из переменных окружения, она должна начинаться на префикс VISMIND_ в переменных окружения.

Важно! Проверяются все переменные из конфигурационного файла в переменных окружения.

Например: считать название приложения.

Название приложения в конфигурационном файле находится в переменной **app_name**. Тогда к **app_name** добавим префикс VISMIND – получится параметр **VISMIND_APP_NAME**. Этот параметр будет искаться в переменных окружения.

И в переменную **app_name** присвоится значение:

- или из переменной окружения **VISMIND_APP_NAME**, если она будет найдена;
- или из **app_name**, если переменная окружения не найдена.

3.6 Песочница и удаленный запуск расчетов

Код из блока Python можно запустить на выполнение удаленно (Рисунок 1). При этом код будет выполняться внутри изолированного контейнера, что обеспечит дополнительный уровень безопасности.

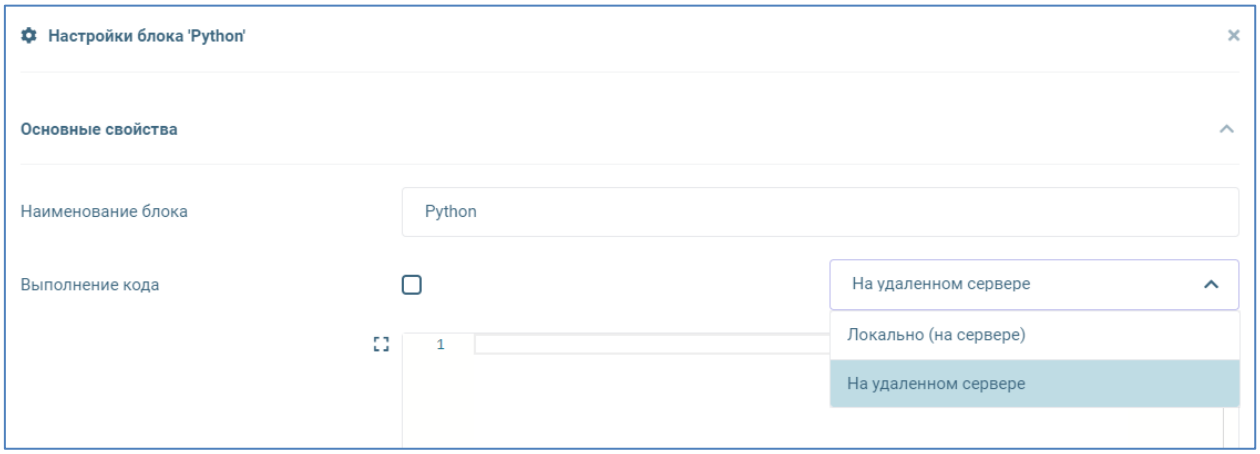

Рисунок 1 – Пример настройки блока Python

Это накладывает определенные ограничения на сам код – нельзя использовать методы библиотеки ядра, так как ядро системы отсутствует при выполнении в удаленном контейнере (методы VmResource).

3.6.1 Сборка docker-образа для запуска кода в удаленном окружении

Библиотеки, которые должны быть доступны внутри контейнера (подключаемые к коду), должны быть добавлены на этапе сборки образа.

1. Собрать образ приложения из репозитория **vismind-sandbox**.

   ```
   docker build -f .\Dockerfile -t vmbackend:sandbox .
   ```

2. Загрузить в docker контейнер. Имя и тег по умолчанию: **vmbackend:sandbox**.

   ```
   docker save -o D:\\vmbackend.sandbox.tar  vmbackend:sandbox
   ```

Результатом будет файл **vmbackend.sandbox.tar**, который можно включить в инсталлятор приложения.

3.6.2 Загрузка образа в локальный docker репозиторий сервера

На сервер, на котором будет выполняться код в изолированном контейнере, необходимо загрузить образ:

```
docker load --input D:\\vmbackend.sandbox.tar
```

3.6.3 Настройка сервера для доступа к docker

После этого необходимо разрешить соединение и управление **docker** по сети. Приложение будет соединяться с сервером, запускать образ с кодом, который создал пользователь, и после получения результата, удалять за собой отработавший контейнер.

Необходимо создать файл:

```
/etc/systemd/system/docker.service.d/startup_options.conf
```

Без сертификатов безопасности (TCP-порт 2375):

```
[Service]

ExecStart=

ExecStart=/usr/bin/dockerd -H fd:// -H tcp://0.0.0.0:2375 --
containerd=/run/containerd/containerd.sock
```

Перезапустить сервис:

```
sudo systemctl daemon-reload

sudo systemctl restart docker
```

Добавить разрешение в firewall:

```
sudo ufw allow 2375/tcp
```

После этого сервис доступен по сети на 2375 TCP-порту.

С использованием сертификатов безопасности (TCP-порт 2376):

Добавить файл расширения сервиса **systemd** (основной файл с описанием юнита /etc/systemd/system/multi-user.target.wants/docker.service) для изменения параметров старта демона:

```
/etc/systemd/system/docker.service.d/startup_options.conf

[Service]

ExecStart=

ExecStart=/usr/bin/dockerd -H fd:// -H tcp://0.0.0.0:2376 --
containerd=/run/containerd/containerd.sock --tlsverify --
tlscacert=/etc/docker/ssl/ca.pem --tlscert=/etc/docker/ssl/cert.pem --
tlskey=/etc/docker/ssl/key.pem
```

Первой пустой строкой сбрасывается значение параметров старта для сервиса **docker**. Во второй необходимо указать:

- `H fd://` – означает, что параметрами прослушивания сокета управляет **systemd** в файле /etc/systemd/system/sockets.target.wants/docker.socket (по умолчанию подключение к dockerd осуществляется через unix-сокет /var/run/docker.sock, изменять можно в юните сокета или параметром `-H unix:///var/run/docker-bootstrap.sock`);
- `H tcp://0.0.0.0:2376` – указываем, что демон **dockerd** (бэкенд часть платформы контейнеризации, не путать с клиентской частью **docker**) запустит серверный TCP-порт 2376, слушающий входящие соединения на всех доступных IP-адресах сетевых

интерфейсов (для запуска сокета только на конкретном IP-адресе укажите его вместо 0.0.0.0);

- `--tls*` – параметры, включающие защиту TLS-сертификатами сетевого подключения к сервису.

С TLS-сертификатами безопасности TCP-порт 2376. Разрешить доступ к порту в файрволе:

```
sudo ufw allow 2376/tcp
```

Обновите конфигурацию демонов **systemd** и перезагрузить сервис **docker**:

```
sudo systemctl daemon-reload
sudo systemctl restart docker
```

Посмотреть статус запущенного сервиса можно командой:

```
systemctl status docker
```

Основной процесс /usr/bin/dockerd должен быть запущен с новыми параметрами.

Проверим, что подключение без сертификата будет отклонено:

```
docker -H tcp://127.0.0.1:2376 info
```

Вывод ошибки:

```
Get http://127.0.0.1:2376/v1.21/containers/json: malformed HTTP response "\x15\x03\x01\x00\x02\x02".
* Are you trying to connect to a TLS-enabled daemon without TLS?
```

3.6.4 Настройка образа в приложении

Для настройки образа в приложении необходимо:

1. Указать название образа контейнера:

   ```
   docker_for_python_iso_name = "vmbackend:sandbox"
   ```

2. Указать URL docker-контейнера:

   ```
   docker_for_python_base_url = "tcp://dev.int.bittechno.ru:2376"
   ```

3. Добавить сертификаты docker-контейнера:

   - Путь к сертификату клиента:

     ```
     docker_for_python_client_cert = "./conf/docker/cert.pem"
     ```

   - Путь к ключу клиента:

     ```
     docker_for_python_client_key = "./conf/docker/key.pem"
     ```

   - Путь к центру сертификации:

     ```
     docker_for_python_ca_cert = "./conf/docker/ca.pem"
     ```

   - Установить флаг, указывающий необходимость удаления контейнера после выполнения:

     ```
     docker_for_python_is_remove = 1
     ```

     - Этот параметр нельзя отставлять надолго включенным, он нужен для отладки контейнера, в случае если код по какой-то причине не выполняется и приложению не отправляется корректная ошибка.
     - Если его включить, каждый запуск будет создавать контейнер, но **не будет его удалять** по завершению работы, что приведет к большому количеству контейнеров, которые останутся завершенными в docker сервера.

Если образ не установлен или настройки приложения заданы некорректно – при попытке запуска будет возвращена системная ошибка, которую вернет сам сервер. Это необходимо для корректной отладки соединения, в случае если настройки некорректные.

3.7 Прокси-сервер для клиентского приложения

Для корректной работы пользовательского приложения в docker-контейнере и без него нужен прокси-сервер, на котором будут выполняться роли:

1. Хостинг приложения по пути '/'.
2. Проксирование запросов к бэкенду по путям:
   - /api
   - /socket.io
   - /metrics
3. Подготовка и отправка статики по пути /static.
4. *Опционально:* предоставление доступа к эндпоинту /metrics для мониторинга приложения с использованием Prometheus.
5. *Опционально:* предоставление эндпоинта для проксирования запросов и организации кэширования ответов тайлового сервера Martin.

Таким образом, прокси-сервер выполняет роль диспетчера всех запросов приложения и является критически важным элементом при развертывании приложения.

В качестве прокси-сервера используется **nginx** (на 03 Feb 2023 используется версия 1.23.3).

3.7.1 Базовая конфигурация

nginx.conf

```
# Определяем пользователя
user  nginx;

# Определяем количество рабочих процессов автоматически
# Параметр auto поддерживается только начиная с версий 1.3.8 и 1.2.5.
worker_processes auto;

# Данная директива указывает сколько файловых дескрипторов будет использовать
Nginx. На каждое соединение надо
# выделять по два дескриптора, даже для статических файлов (картинки/JS/CSS):
один для соединения с клиентом,
# а второй — для открытия статического файла. Таким образом, значение
worker_rlimit_nofile должно быть не меньше
# удвоенного значению Max Clients.
# В системе это значение можно установить из командной строки ulimit -n 200000
или используя
# /etc/security/limits.conf.
# Проверить установленное значение в системе
# hard limit `ulimit -Hn` и soft limit `ulimit -Sn`
worker_rlimit_nofile 20000;

# !! для установки без контейнера надо вывод ошибок перенаправить в файлы, а не
STDOUT !!
# pid        /var/log/nginx/nginx.pid;
```

```
# задаем общие параметры логирования
# error_log  /var/log/nginx/error.log warn;
error_log  /dev/stdout info;
# логируем только критические ошибки
# error_log /var/log/nginx/error.log crit
# Если нужно полностью отключить ошибки (использовать с пониманием зачем это нужно)
# error_log /dev/null crit;

events {
   # увеличение по сравнению со стандартным 1024, для продуктивного контура, можно увеличивать больше в зависимости от нагрузки и ресурсов
   # worker_processes * worker_connections = max_clients (кол-во одновременных соединений)
   worker_connections  2048;

   # предпочтительнее для Linux
   use epoll;

   # Для того, чтобы Nginx пытался принять максимальное количество подключений, необходимо включить
   # директиву multi_accept. Однако при слишком маленьком значении worker_connections, их лимит может быть
   # быстро исчерпан.
   multi_accept on;
}

http {

    include       /etc/nginx/mime.types;
    default_type  application/octet-stream;

    log_format  main  '$remote_addr - $remote_user [$time_local] "$request" '
                      '$status $body_bytes_sent "$http_referer" '
                      '"$http_user_agent" "$http_x_forwarded_for"';

    # Куда писать лог доступа и уровень логирования
    # добавляем  buffer=16k для оптимизации логирования
    # access_log  /var/log/nginx/access.log  main buffer=16k;
    access_log /dev/stdout main;
    # либо отключаем лог - так будет работать намного быстрее
    # access_log off;

    # Для нормального ответа 304 Not Modified;
    if_modified_since before;

    # Включаем поддержку WebP
    map $http_accept $webp_ext {
        default "";
        "~*webp" ".webp";
    }

    ##
    # Basic Settings
    ##
```

```
    # Используется, если количество имен серверов большое
    #server_names_hash_max_size 1200;
    #server_names_hash_bucket_size 64;


    ### Обработка запросов ###

    # Метод отправки данных sendfile более эффективен, чем стандартный метод
read+write
    sendfile on;
    # Будет отправлять заголовки и и начало файла в одном пакете
    tcp_nopush on;
    tcp_nodelay on;

    ### Адрес резолвера DNS ###
    # resolver 10.0.0.1 valid=30s;

    ### Информация о файлах ###

    # Максимальное количество файлов, информация о которых будет содержаться в
кеше
    open_file_cache max=200000 inactive=20s;
    # Через какое время информация будет удалена из кеша
    open_file_cache_valid 30s;
    # Кеширование информации о тех файлах, которые были использованы хотя бы 2
раза
    open_file_cache_min_uses 2;
    # Кеширование информации об отсутствующих файлах
    open_file_cache_errors on;


    # Удаляем информацию об nginx в headers
    server_tokens off;
    # Будет ждать 30 секунд перед закрытием keepalive соединения
    keepalive_timeout 30s;
    ## Максимальное количество keepalive запросов от одного клиента
    keepalive_requests 100;

    # Разрешает или запрещает сброс соединений по таймауту
    reset_timedout_connection on;
    # Будет ждать 30 секунд тело запроса от клиента, после чего сбросит
соединение
    client_body_timeout 30s;
    # В этом случае сервер не будет принимать запросы размером более 200Мб
    client_max_body_size 200m;
    # Если клиент прекратит чтение ответа, Nginx подождет 30 секунд и сбросит
соединение
    send_timeout 30s;

    # Proxy #
    # Задаёт таймаут для установления соединения с проксированным сервером.
    # Необходимо иметь в виду, что этот таймаут обычно не может превышать 75
секунд.
    proxy_connect_timeout 30s;
```

```
    # Задаёт таймаут при передаче запроса проксированному серверу.
    # Таймаут устанавливается не на всю передачу запроса, а только между двумя операциями записи.
    # Если по истечении этого времени проксируемый сервер не примет новых данных, соединение закрывается.
    proxy_send_timeout 30s;
    # Задаёт таймаут при чтении ответа проксированного сервера.
    # Таймаут устанавливается не на всю передачу ответа, а только между двумя операциями чтения.
    # Если по истечении этого времени проксируемый сервер ничего не передаст, соединение закрывается.
    proxy_read_timeout 30s;

    # Размер буфера используемого для проксированых запросов
    # proxy_buffers 32 4k;

    ##
    # Кэш для геосервера
    ##
    # proxy_cache_path    /var/cache/nginx/
    #                     levels=1:2
    #                     max_size=10g
    #                     inactive=60m
    #                     use_temp_path=off
    #                     keys_zone=tiles_cache:10m;

    ##
    # Gzip Settings
    ##

    # Включаем сжатие gzip
    gzip on;
    # Для IE6 отключить
    gzip_disable "msie6";
    # Добавляет Vary: Accept-Encoding в Headers
    gzip_vary on;
    # Сжатие для всех проксированных запросов (для работы NGINX+Apache)
    gzip_proxied any;
    # Устанавливает степень сжатия ответа методом gzip. Допустимые значения находятся в диапазоне от 1 до 9
    gzip_comp_level 6;
    # Задаёт число и размер буферов, в которые будет сжиматься ответ
    gzip_buffers 16 8k;
    # Устанавливает минимальную HTTP-версию запроса, необходимую для сжатия ответа. Значение по умолчанию
    gzip_http_version 1.1;
    # MIME-типы файлов в дополнение к text/html, которые нужно сжимать
    gzip_types text/plain text/css application/json application/x-javascript text/xml application/xml application/xml+rss text/javascript application/javascript image/svg+xml;
    # Минимальная длина файла, которую нужно сжимать
    gzip_min_length 10;

  # Подключаем конфиги конкретных сайтов
  include /etc/nginx/conf.d/*.conf;
```

```
}
```

Для самого приложения в контейнере или в **nginx** настраивается конфигурация **default.conf**, которая располагается в каталоге /etc/nginx/conf.d/.

Если установка администратором произведена в другое место (при установке не в контейнере) – необходимо не забыть заменить в файле **nginx.conf** путь до конфигураций сайтов.

default.conf

```
upstream app_server {
    # указываем правильный адрес
    server vismind-backend:35115 max_fails=5 fail_timeout=5s;
    # ребуется для оптимизации keep alive соединений
    keepalive 50;
}


server { # подключение к бэкенду 80 и 443 порты

    listen       80;
    # listen       443 http2 ssl;
    # ssl_certificate     /etc/ssl/certs/ssl.crt;
    # ssl_certificate_key /etc/ssl/private/ssl.key;

    # устанавливать если меняется на доменное имя
    # server_name  my_server_name;

    # Основное приложений
    location / {
        root   /usr/share/nginx/html;
        try_files $uri$args $uri$args/ $uri/ /index.html;
    }

    # Папка статики
    location /static/ {
        root /usr/share/nginx/media;
        add_header Access-Control-Allow-Origin *;
        # expires 1d; # VISMIND-6615
    }

    # Обратный прокси для бэкенда
    location /api/ {
        proxy_http_version 1.1;
        proxy_set_header   "Connection" "";
        proxy_set_header Host $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_connect_timeout   600;
        proxy_send_timeout      600;
        proxy_read_timeout      600;
        proxy_buffers           32 4k;

        proxy_redirect $scheme://$host/api/v1/ $scheme://$http_host/api/;
```

```
        # с бэкендом работа идет по http
        proxy_pass http://app_server/api/v1/;
    }

    # Проброс сокетов
    location /socket.io {
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "Upgrade";
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_pass http://app_server;
        tcp_nodelay on;
        keepalive_timeout 600;
    }

    # Ендпоинт для получения метрик приложения
    location /metrics {
        proxy_set_header Host $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_pass http://app_server/metrics;
    }

    # Геосервер с кэшированием запросов
    location ~ /tilecache/(?<fwd_path>.*) {
        proxy_pass
http://dev.int.bittechno.ru:3000/static/$fwd_path$is_args$args;
    }

    # Геосервер без кэширования, по сути прокси
    location ~ /tileserver/(?<fwd_path>.*) {
        proxy_pass
http://dev.int.bittechno.ru:3000/dynamic/$fwd_path$is_args$args;
    }

    # 404 ошбка
    #error_page  404              /404.html;

    # перенаправление при ошибках 50x на /50x.html2
    #error_page   500 502 503 504  /50x.html;
    #location = /50x.html {
    #    root   /usr/share/nginx/html;
    #}
}
```

3.7.2 Прокси-сервер для относительных адресов геосервера

Дополнительная роль, которую может выполнять **nginx** – перенаправление запросов на геосервер в случае использования относительных адресов. Это необходимо для того, чтобы не переписывать адреса при переносе задач между разными окружениями.

Для этого в конфигурации задается параметр **location** для кэшируемых и динамических тайлов. Геосервер должен иметь свой прокси-сервер с настроенным кэшированием по каталогам /static и /dynamic.

Пример location для reverse proxy

```
# Геосервер с кэшированием запросов
    location ~ /tilecache/(?<fwd_path>.*) {
        proxy_pass
http://martin_address/static/$fwd_path$is_args$args;
    }

    # Геосервер без кэширования, по сути прокси
    location ~ /tileserver/(?<fwd_path>.*) {
        proxy_pass
http://martin_address:3000/dynamic/$fwd_path$is_args$args;
    }
```

При использовании проброса запросов на геосервер они направляются на прокси-сервер геосервера, который отвечает за кэширование и их перенаправления на сам геосервер (Рисунок 2).

Важно! При использовании этого варианта проксирования запросов нельзя изменять заголовки запросов, так как ответ геосервера содержит адрес, по которому надо получать тайлы, а этот адрес формируется из заголовков запросов пришедших на геосервер.

3.7.2.1 Схема работы прокси-сервера и геосервера

Схема работы прокси-сервера и геосервера приведена на рисунке (Рисунок 2).

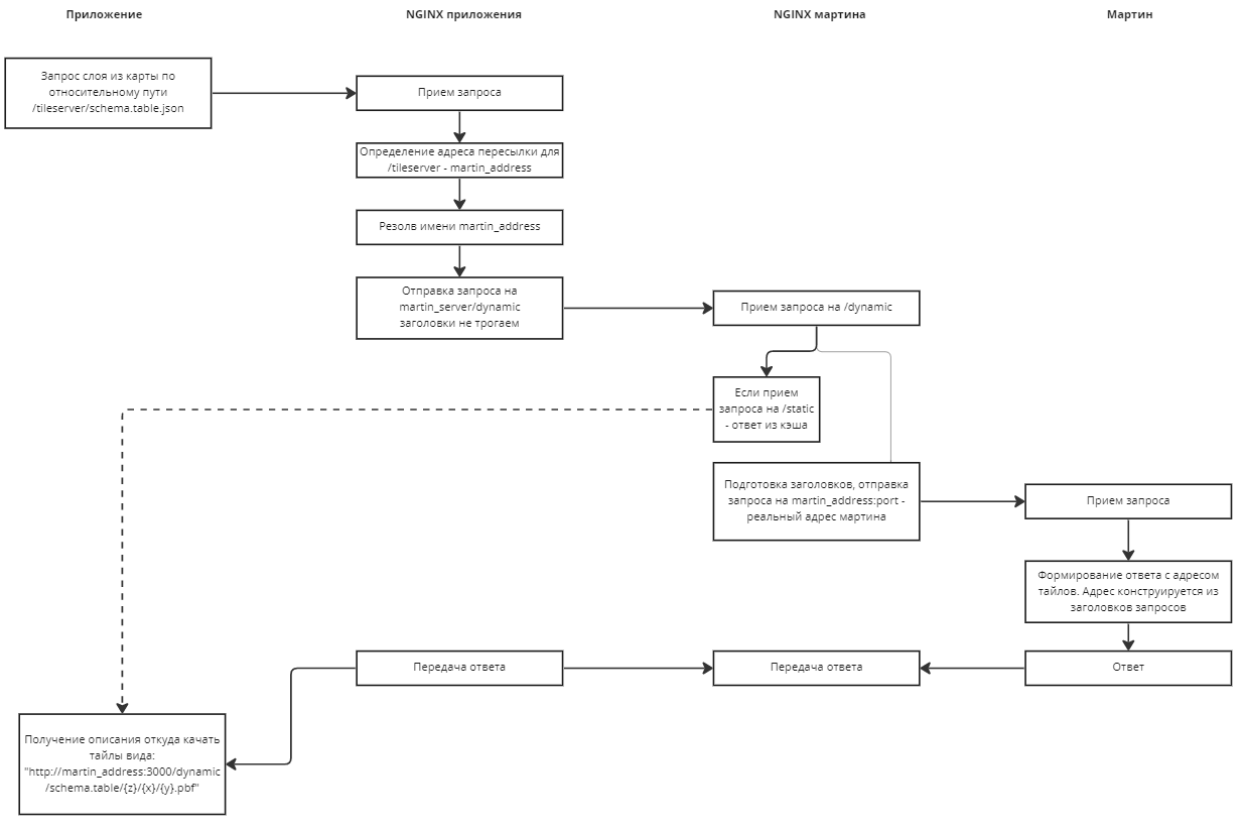

Рисунок 2 – Схема работы прокси-сервера и геосервера