Инструкция по установке экземпляра UX Rocket серверное решение

Оглавление

Введение

Программный продукт UX Rocket может быть поставлен заказчику в двух форматах: облачное решение и серверное решение. Подробную информацию вы можете получить по email support@excitekit.ru и телефону +7(495) 725-4376.

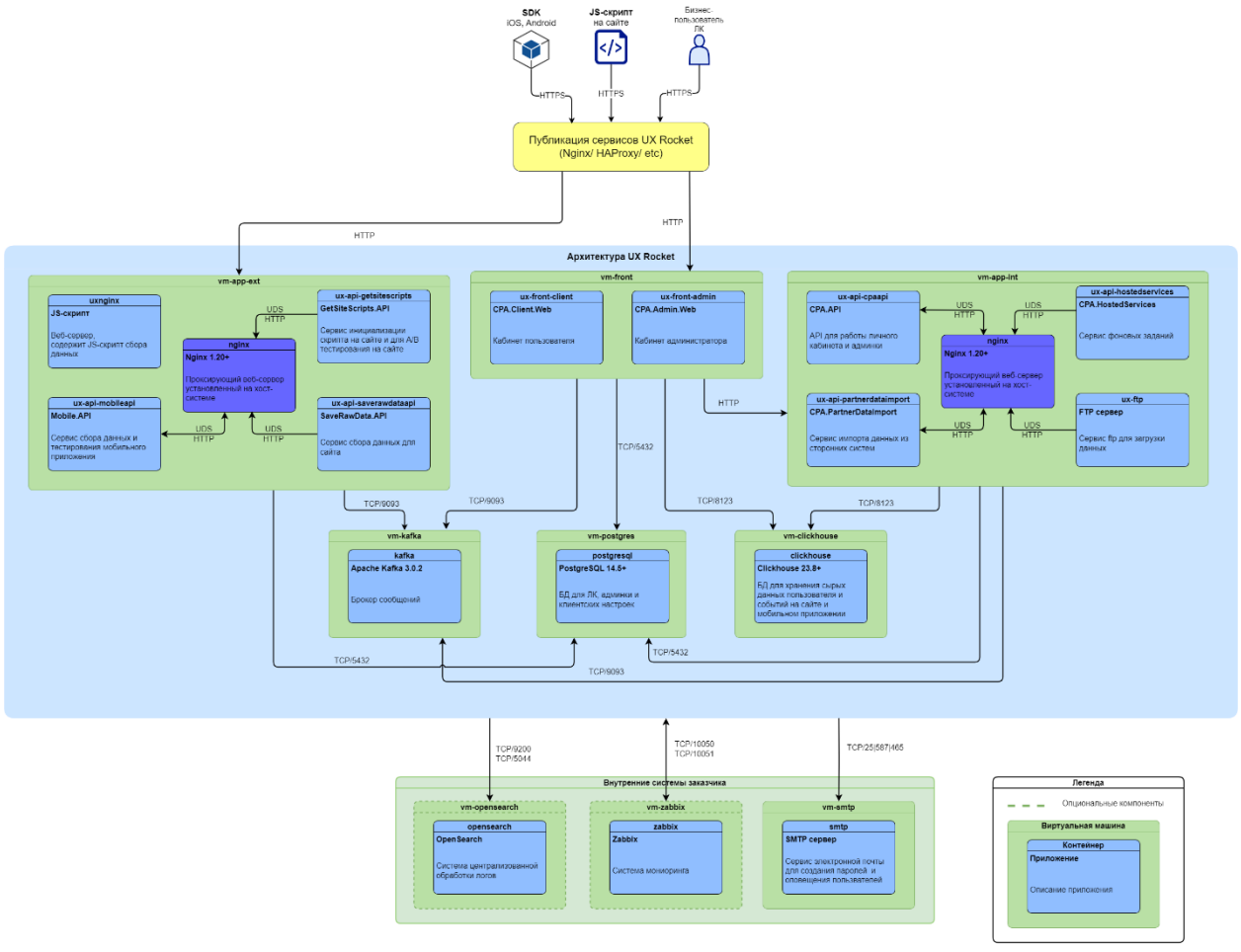
Данное руководство рассматривает вариант установки серверного решения UX Rocket. Серверное UX Rocket может быть развернуто как на физических серверах, так и полностью поддерживается работа в виртуальном окружении.

Требования к системе

Для установки программного продукта UX Rocket необходимы отдельные виртуальные машины для следующих компонент:

- External Backend Services (API)
- Internal Backend Services (API)
- Frontend Services (портал)
- Apache Kafka (брокер сообщений)
- Clickhouse (база данных)
- PostgreSQL (база данных)

Схема взаимодействия компонент для настройки сетевого доступа показана на рисунке ниже:



Минимальные системные требования к виртуальным машинам указаны в таблице ниже:

Назначение виртуальной машины	**Процессор, ядро (vCPU)**	**ОЗУ, ГБ (vRAM)**	**Объем SSD диска, ГБ**
External Backend Services (API)	8	8	60
Internal Backend Services (API)	4	8	60
Frontend Services (портал)	2	8	60
Apache Kafka (брокер сообщений)	4	12	75
Clickhouse (база данных)	4	10	100
PostgreSQL (база данных)	4	16	100

На виртуальных машинах необходимо установить одну из перечисленных ниже операционных систем: Astra Linux актуальной версии, Ubuntu 20.04 LTS и выше, Debian 11 и выше.

Оборудование пользователя должно соответствовать рекомендуемым требованиям для функционирования браузера, через который пользователь использует программное обеспечение. Для использования программного обеспечения производитель рекомендует пользователю использовать следующие браузеры:

- Google Chrome 87.0 и выше;
- Mozilla Firefox 84.0 и выше;
- Safari 14.0 и выше;
- Opera 72.0 и выше.

Сетевое взаимодействие компонентов

Для быстрого развертывания тестового стенда UX Rocket во всех конфигурационных файлах используется псевдо-домен верхнего уровня local.

Для примера будут использованы следующие значения:

Сервис	**IP-адрес**	**Hostname**
PostgreSQL	192.168.1.27	ux-astra-db.local
Kafka	192.168.1.23	ux-astra-kafka.local
Backend	192.168.1.17	ux-astra-app.local
Clickhouse	192.168.1.18	ux-astra-clickhouse.local
Frontend	192.168.1.21	
Backend EXT	192.168.1.16	

Файл .env внутри каждого сервиса файлах отвечает за сопоставления имен хостов и IP-адресов для контейнера.

Пример env-файлв:

```
# .env
# Настройки IP-адресов
IPappext=192.168.1.16
IPappint=192.168.1.17
IPclickhouse=192.168.1.18
IPfront=192.168.1.21
IPkafka=192.168.1.23
IPpostgresql=192.168.1.27
```

где hostname ux-astra-db.local сопоставляется с IP-адресом 192.168.1.27 для PostgreSQL.

Обязательно укажите свои значения IP-адресов в .env файлах для каждого из сервисов.

При необходимости Вы можете добавить и внести изменения в настройки host всем виртуальным машинам, на которых будут установлены сервисы, и на рабочем месте, с которого будет происходить тестирование системы.

Файл hosts в Linux

```
nano /etc/hosts
```

Файл hosts в Windows

```
c:\windows\system32\drivers\etc\hosts
```

Подробнее как найти и отредактировать файл hosts Вы можете посмотреть описание по ссылке https://1cloud.ru/help/dns/file-hosts

Установка стороннего программного обеспечения

Компоненты стороннего программного обеспечения рекомендуется развернуть с помощью Docker (Docker-compose). Docker – это программная платформа для быстрой разработки, тестирования и развертывания приложений. Docker значительно упрощает процесс настройки и управления приложением. Для установки стороннего ПО с помощью Docker на виртуальные машины нужно установить:

1. Docker Engine – Community + Docker Composec для запуска контейнеров.
2. git
3. SSH для подключения к серверу.

Перед началом установки Вам нужно получить логин и пароль от git-репозитория в службе технической поддержки ООО «ЭКСАЙТ КИТ» (раздел «Контактная информация).

Установки Docker и Docker-compose

Для Astra Linux

Обратитесь к руководству «Установка и администрирование Docker в Astra Linux» по ссылке https://wiki.astralinux.ru/pages/viewpage.action?pageId=158601444

Для Debian и Ubuntu

Обратитесь к руководству по установке Docker для вашей операционной системы по ссылке https://docs.docker.com/engine/install/

Проверить что docker установлен в системе можно следующими командами

```
docker --version

docker-compose --version
```

Установка Git

Проверить что git установлен в системе можно следующими командами

```
git --version
```

Если git не установлен в вашей ОС, то установите его

```
apt install git -y
```

Установка PostgreSQL

Перед началом работ убедитесь, что виртуальная машина соответствует общим требованиям.

1. Подключитесь к серверу, на котором будет развернута PostgreSQL по SSH.
2. Перейдите в режим суперпользователя (переключиться в root)
   ```
   sudo su -
   ```
3. Далее создайте каталог, в котором будут развернуты PostgreSQL сервисы, с помощью docker.
   ```
   mkdir -p /docker-app
   ```
4. Перейдите в созданный каталог
   ```
   cd /docker-app
   ```
5. Далее скачайте на сервер tgz-архив, который содержит все необходимые для работы предварительно настроенной базы данных PostgreSQL файлы (конфиги, docker-compose, db-файлы), из git-репозитория.
   ```
   git clone https://gitflic.uxrocket.ru/project/adminuser/ux-onpremise-postgresql.git
   ```
   Пример результата выполнения команды git clone
   ```
   Cloning into 'ux-onpremise-postgresql'...
   Username for 'https://gitflic.uxrocket.ru': ***@****.ru
   Password for 'https://***@****.ru@gitflic.uxrocket.ru':
   remote: Counting objects: 4, done
   remote: Finding sources: 100% (4/4)
   remote: Getting sizes: 100% (3/3)
   remote: Total 4 (delta 0), reused 4 (delta 0)

   Unpacking objects: 100% (4/4), 1.71 GiB | 12.72 MiB/s, done.
   ```
6. Распакуйте tgz-архив скачанный с помощью команды git clone
   ```
   cd /docker-app/
   root@ux-nt-postgresql:/docker-app/ux-onpremise-postgresql# more md5sum.log
   1dd4cb2f5978b6ce2a9a582dfba94326  ux-postgres.tgz
   root@ux-nt-postgresql:/docker-app/ux-onpremise-postgresql# md5sum ux-
   postgres.tgz
   1dd4cb2f5978b6ce2a9a582dfba94326  ux-postgres.tgz

   tar -xzvpf ux-postgres.tgz

   root@ux-nt-postgresql:/docker-app/ux-onpremise-postgresql# ls
   md5sum.log  ux-postgres  ux-postgres.tgz
   ```

```
root@ux-nt-postgresql:/docker-app/ux-onpremise-postgresql# mv ux-postgres ../
```

7. Пример структуры каталога /docker-app/ux-postgres после выполнения команды git clone и распаковки tgz-архив

```
root@ux-nt-postgresql:/docker-app/ux-postgres# tree -La 2
.
├── data
│   ├── base
│   ├── global
│   ├── pg_commit_ts
│   ├── pg_dynshmem
│   ├── pg_hba.conf
│   ├── pg_ident.conf
│   ├── pg_logical
│   ├── pg_multixact
│   ├── pg_notify
│   ├── pg_replslot
│   ├── pg_serial
│   ├── pg_snapshots
│   ├── pg_stat
│   ├── pg_stat_tmp
│   ├── pg_subtrans
│   ├── pg_tblspc
│   ├── pg_twophase
│   ├── PG_VERSION
│   ├── pg_wal
│   ├── pg_xact
│   ├── postgresql.auto.conf
│   ├── postgresql.conf
│   └── postmaster.opts
├── docker-compose.yml
├── .env
└── pg_hba.conf
```

8. Авторизуйтесь в docker registry

```
do
Результат:
docker login https://dockerhub.uxrocket.ru
Username: uxrocket
Password:
…
Login Succeeded
```

9. Отредактируйте файл .env (/docker-app/ux-postgres/.env), который отвечает за сопоставления имен хостов и IP-адресов для сервисов UX Rocket.

```
nano /docker-app/ux-postgres/.env

# .env
# Настройки IP-адресов
IPappext=192.168.1.16
IPappint=192.168.1.17
```

```
IPclickhouse=192.168.1.18
IPfront=192.168.1.21
IPkafka=192.168.1.23
IPpostgresql=192.168.1.27
```

10. Запустите контейнер с PostgreSQL

```
cd /docker-app/ux-postgres
docker-compose up -d

Результат:

Creating network "ux-postgres_default" with the default driver
Creating ux-postgres_postgres_1 ... done
root@ux-astra-db:/docker-app/ux-postgres#
```

11. Проверьте что контейнер с PostgreSQL запустился

```
sleep 10 && docker ps
```

Посмотреть реквизиты подключения к PostgreSQL вы можете в docker-compose.yml - /docker-app/ux-postgres/docker-compose.yml

Установка ClickHouse

Перед началом работ убедитесь, что виртуальная машина соответствует общим требованиям.

1. Подключитесь к серверу, на котором будет развернута Clickhouse по SSH.
2. Перейдите в режим суперпользователя (переключиться в root)

```
sudo su -
```

3. Далее создайте каталог, в котором будут развернуты Clickhouse сервисы, с помощью docker.

```
mkdir -p /docker-app
```

4. Перейдите в созданный каталог

```
cd /docker-app
```

5. Далее скачайте на сервер файлы (конфиги, docker-compose, db-файлы), необходимые для работы предварительно настроенной базы данных Clickhouse, из git-репозитория.

```
git clone https://gitflic.uxrocket.ru/project/adminuser/ux-onpremise-clickhouse.git
```

Результат выполнения команды git clone

```
Cloning into 'ux-onpremise-clickhouse'...
Username for 'https://gitflic.uxrocket.ru': ***@****.ru
Password for 'https://***@****.ru@gitflic.uxrocket.ru':
remote: Counting objects: 4, done
remote: Finding sources: 100% (4/4)
remote: Getting sizes: 100% (3/3)
remote: Total 4 (delta 0), reused 4 (delta 0)
Unpacking objects: 100% (4/4), 10.53 MiB | 14.70 MiB/s, done.
```

6. Распакуйте tgz-архив скачанный с помощью команды git clone

```
cd ux-onpremise-clickhouse
tar -xzvpf ux-clickhouse.tgz
```

7. Пример структуры каталога ux-clickhouse после выполнения команды git clone

```
root@ux-nt-clickhouse:/docker-app/ux-onpremise-clickhouse/ux-clickhouse# tree -La
2
.
├── conf
│   ├── config.d_listen.xml
│   ├── config.xml
│   └── users.xml
├── data
│   ├── access
│   ├── data
│   ├── dictionaries_lib
│   ├── flags
│   ├── format_schemas
│   ├── metadata
│   ├── metadata_dropped
│   ├── named_collections
│   ├── preprocessed_configs
│   ├── status
│   ├── store
│   ├── tmp
│   ├── user_defined
│   ├── user_files
│   ├── user_scripts
│   └── uuid
├── docker-compose.yml
├── .env
└── logs
    ├── clickhouse-server.err.log
    ├── clickhouse-server.log
    ├── errors.log
    └── trace.log│    ├── user_files
│   ├── user_scripts
│   └── uuid
├── docker-compose.yml
└── logs
    ├── clickhouse-server.err.log
    ├── clickhouse-server.log
    ├── errors.log
    └── trace.log
```

8. Переместите распакованный каталог ux-clickhouse в /docker-app/

```
cd /docker-app/ux-onpremise-clickhouse/
mv ux-clickhouse ../
```

9. Авторизуйтесь в docker registry

```
docker login https://dockerhub.uxrocket.ru
Результат:
docker login https://dockerhub.uxrocket.ru
Username: uxrocket
```

```
Password:
…
Login Succeeded
```

10. Отредактируйте файл .env (/docker-app/ux-postgres/.env), который отвечает за сопоставления имен хостов и IP-адресов для сервисов UX Rocket.

```
nano /docker-app/ux-clickhouse/.env

# .env
# Настройки IP-адресов
IPappext=192.168.1.16
IPappint=192.168.1.17
IPclickhouse=192.168.1.18
IPfront=192.168.1.21
IPkafka=192.168.1.23
IPpostgresql=192.168.1.27
```

11. Запустите контейнер с Clickhouse

```
cd /docker-app/ux-clickhouse
docker-compose up -d

Результат:

root@ux-nt-clickhouse:/docker-app/ux-clickhouse# docker-compose up -d
[+] Building 0.0s (0/0)
[+] Running 2/2
 ✓ Network ux-clickhouse_default              Created
0.1s
 ✓ Container ux-clickhouse-clickhouse-server-1  Started
```

12. Проверьте что контейнер с PostgreSQL запустился

```
sleep 10 && docker ps
```

Установка Apache Kafka

Перед началом работ убедитесь, что виртуальная машина соответствует общим требования.

1. Подключитесь к серверу, на котором будет развернута Kafka по SSH.
2. Перейдите в режим суперпользователя (переключиться в root)

```
sudo su -
```

3. Далее создайте каталог, в котором будут развернуты Kafka сервисы, с помощью docker.

```
mkdir -p /docker-app
```

4. Перейдите в созданный каталог

```
cd /docker-app
```

5. Далее скачайте на сервер файлы (конфиги, docker-compose), необходимые для работы предварительно настроенного брокера очередей Kafka, из git-репозитория.

```
git clone https://gitflic.uxrocket.ru/project/adminuser/ux-onpremise-kafka.git
```

Пример выполнения команды git clone

```
root@ux-nt-kafka:/docker-app# git clone https://***@****.ru/project/adminuser/ux-onpremise-kafka.git
Cloning into 'ux-onpremise-kafka'...
Username for 'https://gitflic.uxrocket.ru': ***@****.ru
Password for 'https://***@****.ru@gitflic.uxrocket.ru':
remote: Counting objects: 4, done
remote: Finding sources: 100% (4/4)
remote: Getting sizes: 100% (3/3)
remote: Total 4 (delta 0), reused 4 (delta 0)
Unpacking objects: 100% (4/4), 13.91 MiB | 11.77 MiB/s, done.
root@ux-nt-kafka:/docker-app#
```

6. Распакуйте tgz-архив скачанный с помощью команды git clone

```
root@ux-nt-kafka:/docker-app# cd ux-onpremise-kafka/
root@ux-nt-kafka:/docker-app/ux-onpremise-kafka# tar -xzvpf ux-kafka.tgz ./ux-kafka/
```

7. Переместите распакованный каталог ux-kafka в /docker-app/

```
root@ux-nt-kafka:/docker-app/ux-onpremise-kafka# mv ux-kafka ../
```

8. Пример структуры /docker-app/ux-kafka после выполнения команды git clone

```
/docker-app/ux-kafka
├── config
│   ├── connect-console-sink.properties
│   ├── connect-console-source.properties
│   ├── connect-distributed.properties
│   ├── connect-file-sink.properties
│   ├── connect-file-source.properties
│   ├── connect-log4j.properties
│   ├── connect-mirror-maker.properties
│   ├── connect-standalone.properties
│   ├── consumer.properties
│   ├── kraft
│   ├── log4j.properties
│   ├── producer.properties
│   ├── server.properties
│   ├── tools-log4j.properties
│   ├── trogdor.conf
│   └── zookeeper.properties
├── data
│   ├── hsperfdata_root
│   ├── kafka-logs
│   └── zookeeper
├── docker-compose.yml
└── .env
```

9. Отредактируйте файл .env (/docker-app/ux-kafka/.env), который отвечает за сопоставления имен хостов и IP-адресов для сервисов UX Rocket.

```
nano /docker-app/ux-kafka/.env

# .env
# Настройки IP-адресов
```

```
IPappext=192.168.1.16
IPappint=192.168.1.17
IPclickhouse=192.168.1.18
IPfront=192.168.1.21
IPkafka=192.168.1.23
IPpostgresql=192.168.1.27
```

10. Авторизуйтесь в docker registry

```
docker login https://dockerhub.uxrocket.ru
Результат:
docker login https://dockerhub.uxrocket.ru
Username: uxrocket
Password:
…
Login Succeeded
```

11. Запустите контейнер с Kafka

```
cd /docker-app/ux-kafka

root@ux-nt-kafka:/docker-app/ux-kafka# docker-compose up -d
[+] Building 0.0s (0/0)
[+] Running 3/3
 ✓ Network ux-kafka_default  Created
0.1s
 ✓ Container kafka0          Started
0.6s
 ✓ Container kafka-ui        Started
0.9s
root@ux-nt-kafka:/docker-app/ux-kafka#
```

13. Проверьте что контейнер с Kafka запустился

```
sleep 10 && docker ps
```

Установка ПО «UX Rocket»

Программное обеспечение UX Rocket надо развернуть с помощью Docker (Docker-compose). Установочные скрипты и docker-compose файлы хранятся в git репозитории.

Перед началом установки Вам нужно получить логин и пароль от git-репозитория в службе технической поддержки ООО «ЭКСАЙТ КИТ» (раздел «Контактная информация).

Предварительные требования к APP-серверу

Новые версии UX Rocket используют в базовом варианте сокет домена Unix (Unix domain socket, UDS), вместо TCP, так как этот вариант более эффективен.

Чтобы настроить обратный прокси-сервера (Nginx) на APP-сервере UXrocket (Docker-хост) для переадресации запросов на веб-сервер Kestrel внутрь контейнеров по UDS, нужно

1) Установить Nginx

Обратитесь к официальному руководству по установке Nginx по ссылке https://www.nginx.com/resources/wiki/start/topics/tutorials/install/

Для External Backend Services (API)

После установки Nginx, отредактировать файл **/etc/nginx/conf.d/default.conf**

```
more /etc/nginx/conf.d/ext.conf
# 13:23 20.09.2023
# saverawdataapi
server {
        listen 6000;
        server_name api.uxrocket-example.ru;
location  / {
        proxy_pass http://unix://docker-app/ux-backend/saverawdataapi/kestrel/api.sock;
        proxy_http_version 1.1;
        proxy_buffering off;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header content-type "application/json";
# Timeout
        proxy_connect_timeout 50s;
        proxy_send_timeout 50s;
        proxy_read_timeout 50s;
        send_timeout 50s;
    }
}
# getsitescriptsapi
server {
        listen 6001;
        server_name api.uxrocket-example.ru;
location  / {
#         include /etc/nginx/acl/uxrocket-dev-cors.conf;
        proxy_headers_hash_max_size 512;
        proxy_headers_hash_bucket_size 64;
        proxy_pass http://unix://docker-app/ux-backend/getsitescriptsapi/kestrel/api.sock;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection 'upgrade';
```

```
        proxy_set_header Host $host;
        proxy_cache_bypass $http_upgrade;
# Timeout
        proxy_connect_timeout 50s;
        proxy_send_timeout 50s;
        proxy_read_timeout 50s;
        send_timeout 50s;
    }
}

server {
        listen 6002;
        server_name api.uxrocket-example.ru;
location  / {
#         include /etc/nginx/acl/uxrocket-dev-cors.conf;
        proxy_headers_hash_max_size 512;
        proxy_headers_hash_bucket_size 64;
        proxy_pass http://unix://docker-app/ux-backend/mobileapi/kestrel/api.sock;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection 'upgrade';
        proxy_set_header Host $host;
        proxy_cache_bypass $http_upgrade;
# Timeout
        proxy_connect_timeout 50s;
        proxy_send_timeout 50s;
        proxy_read_timeout 50s;
        send_timeout 50s;
    }
}
```

nano /etc/nginx/nginx.conf

```
user  root;
```

Для Internal Backend Services (API)

После установки Nginx, отредактировать файл **/etc/nginx/conf.d/default.conf**

```
# cpaapi
server {
        listen 8000;
        server_name api.uxrocket-example.ru ux-nt-appint.excteam.ru;
        location  / {
                proxy_pass http://unix://docker-app/uxrocket-backend/cpaapi/kestrel/api.sock;
                proxy_http_version 1.1;

                proxy_buffering off;
                proxy_set_header X-Real-IP $remote_addr;
                proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
# Timeout
        proxy_connect_timeout 50s;
        proxy_send_timeout 50s;
        proxy_read_timeout 50s;
        send_timeout 50s;
        }
}
# partnerdataimport
server {
        listen 8500;
        server_name api.uxrocket-example.ru;
        location  / {
                proxy_headers_hash_max_size 512;
                proxy_headers_hash_bucket_size 64;
                proxy_pass                                      http://unix://docker-app/uxrocket-
backend/partnerdataimport/kestrel/api.sock;
                proxy_http_version 1.1;
                proxy_set_header Upgrade $http_upgrade;
                proxy_set_header Connection 'upgrade';
                proxy_set_header Host $host;
                proxy_cache_bypass $http_upgrade;
# Timeout
        proxy_connect_timeout 50s;
        proxy_send_timeout 50s;
        proxy_read_timeout 50s;
        send_timeout 50s;
        }
}

# ux-api-hostedservices
```

```
server {
        listen 9500;
        server_name api.uxrocket-example.ru;
        location  / {
                proxy_headers_hash_max_size 512;
                proxy_headers_hash_bucket_size 64;
                proxy_pass                      http://unix://docker-app/uxrocket-backend/ux-api-
hostedservices/kestrel/api.sock;
                proxy_http_version 1.1;
                proxy_set_header Upgrade $http_upgrade;
                proxy_set_header Connection 'upgrade';
                proxy_set_header Host $host;
                proxy_cache_bypass $http_upgrade;
# Timeout
        proxy_connect_timeout 50s;
        proxy_send_timeout 50s;
        proxy_read_timeout 50s;
        send_timeout 50s;
        }
}
```

nano /etc/nginx/nginx.conf

```
user  root
```

Установка External Backend Services (API)

Перед началом работ убедитесь, что виртуальная машина соответствует общим требования.

1. Подключитесь к серверу, на котором будет развернут Backend Services, по SSH.
2. Перейдите в режим суперпользователя (переключиться в root)

   ```
   sudo su -
   ```
3. Далее создайте каталог, в котором будут развернуты Backend сервисы, с помощью docker.

   ```
   mkdir -p /docker-app
   ```
4. Перейдите в созданный каталог

   ```
   cd /docker-app
   ```
5. Далее скачайте на сервер файлы (конфиги, docker-compose), необходимые для работы Backend Services UXRocket, из git-репозитория.

   ```
   git clone https://gitflic.uxrocket.ru/project/adminuser/ux-onpremise-appext.git
   ```

   Пример выполнения команды git clone

   ```
   root@ux-nt-appext:/docker-app# git clone
   https://gitflic.uxrocket.ru/project/adminuser/ux-onpremise-appext.git
   Cloning into 'ux-onpremise-appext'...
   Username for 'https://gitflic.uxrocket.ru': ***@****.ru
   Password for 'https://***@****.ru@gitflic.uxrocket.ru':
   remote: Counting objects: 4, done
   remote: Finding sources: 100% (4/4)
   ```

```
remote: Getting sizes: 100% (3/3)
remote: Total 4 (delta 0), reused 4 (delta 0)
Unpacking objects: 100% (4/4), 2.50 KiB | 2.50 MiB/s, done.
root@ux-nt-appext:/docker-app#
```

6. Распакуйте tgz-архив скачанный с помощью команды git clone

```
root@ux-nt-appext:/docker-app# cd ux-onpremise-appext/
root@ux-nt-appext:/docker-app/ux-onpremise-appext# tar -xzvpf ux-backend.tgz
```

7. Переместите распакованный каталог ux-backend в /docker-app/

```
root@ux-nt-kafka:/docker-app/ux-onpremise-appext # mv ux-backend ../
```

8. Пример структуры /docker-app/uxrocket-backend

```
/docker-app/ux-backend
├── docker-compose.yml
├── .env
├── getsitescriptsapi
│   ├── appsettings.Production.json
│   ├── getsitescriptsapi.log
│   └── kestrel
├── mobileapi
│   ├── appsettings.Production.json
│   ├── kestrel
│   └── mobileapi.log
└── saverawdataapi
    ├── appsettings.Production.json
    ├── kestrel
    └── saverawdataapi.log
```

9. Отредактируйте файл .env (/docker-app/ux-backend/.env), который отвечает за сопоставления имен хостов и IP-адресов для сервисов UX Rocket.

```
nano /docker-app/ux-backend/.env

# .env
# Настройки IP-адресов
IPappext=192.168.1.16
IPappint=192.168.1.17
IPclickhouse=192.168.1.18
IPfront=192.168.1.21
IPkafka=192.168.1.23
IPpostgresql=192.168.1.27
```

10. Авторизуйтесь в docker registry

```
docker login https://dockerhub.uxrocket.ru
Результат:
docker login https://dockerhub.uxrocket.ru
Username: uxrocket
Password:
…
Login Succeeded
```

11. Запустите контейнеры Backend Services UXRocket

```
cd /docker-app/uxrocket-backend
```

```
docker-compose up -d

root@ux-nt-appext:/docker-app/ux-backend# cd /docker-app/ux-backend
root@ux-nt-appext:/docker-app/ux-backend# docker-compose up -d
[+] Building 0.0s (0/0)
[+] Running 3/3
 ✓ Container ux-api-mobileapi          Started
0.3s
 ✓ Container ux-api-saverawdataapi     Started
0.3s
 ✓ Container ux-api-getsitescriptsapi  Started
```

12. Основные конфигурационные файлы настроек Backend Services UXRocket

```
/docker-app/uxrocket-backend/getsitescriptsapi/appsettings.Production.json
/docker-app/uxrocket-backend/mobileapi/appsettings.Production.json
/docker-app/uxrocket-backend/saverawdataapi/appsettings.Production.json
```

13. Проверьте что docker-контейнеры запустились

```
sleep 10 && docker ps
```

Установка Internal Backend Services (API)

Перед началом работ убедитесь, что виртуальная машина соответствует общим требования.

1. Подключитесь к серверу, на котором будет развернут Backend Services, по SSH.
2. Перейдите в режим суперпользователя (переключиться в root)

```
sudo su -
```

3. Далее создайте каталог, в котором будут развернуты Backend сервисы, с помощью docker.

```
mkdir -p /docker-app
```

4. Перейдите в созданный каталог

```
cd /docker-app
```

5. Далее скачайте на сервер файлы (конфиги, docker-compose), необходимые для работы Backend Services UXRocket, из git-репозитория.

```
git clone https://gitflic.uxrocket.ru/project/adminuser/ux-onpremise-appint.git
```

Пример выполнения команды git clone

```
root@ux-nt-appint:/docker-app# git clone
https://gitflic.uxrocket.ru/project/adminuser/ux-onpremise-appint.git
Cloning into 'ux-onpremise-appint'...
Username for 'https://gitflic.uxrocket.ru': ***@****.ru
Password for 'https://***@****.ru@gitflic.uxrocket.ru':
remote: Counting objects: 4, done
remote: Finding sources: 100% (4/4)
remote: Getting sizes: 100% (3/3)
remote: Total 4 (delta 0), reused 4 (delta 0)
Unpacking objects: 100% (4/4), 3.15 KiB | 3.15 MiB/s, done.
```

```
root@ux-nt-appint:/docker-app#
```

6. Распакуйте tgz-архив скачанный с помощью команды git clone

```
root@ux-nt-appext:/docker-app# cd ux-onpremise-appint/
root@ux-nt-appext:/docker-app/ux-onpremise-appint# tar -xzvpf ux-backend.tgz
```

7. Переместите распакованный каталог ux-backend в /docker-app/

```
root@ux-nt-kafka:/docker-app/ux-onpremise-appint # mv uxrocket-backend ../
```

8. Пример структуры /docker-app/uxrocket-backend

```
/docker-app/uxrocket-backend
├── cpaapi
│   ├── appsettings.Production.json
│   ├── cpaapi.log
│   └── kestrel
├── docker-compose.yml
├── .env
├── partnerdataimport
│   ├── appsettings.Production.json
│   ├── kestrel
│   └── partnerdataimport.log
├── ux-api-hostedservices
│   ├── appsettings.Production.json
│   ├── kestrel
│   └── ux-api-hostedservices.log
├── uxftp
│   ├── home
│   └── logs
└── uxnginx
    ├── access.log
    └── error.log
```

9. Отредактируйте файл .env (/docker-app/uxrocket-backend/.env), который отвечает за сопоставления имен хостов и IP-адресов для сервисов UX Rocket.

```
nano /docker-app/uxrocket-backend/.env

# .env
# Настройки IP-адресов
IPappext=192.168.1.16
IPappint=192.168.1.17
IPclickhouse=192.168.1.18
IPfront=192.168.1.21
IPkafka=192.168.1.23
IPpostgresql=192.168.1.27
```

10. Авторизуйтесь в docker registry

```
docker login https://dockerhub.uxrocket.ru
Результат:
docker login https://dockerhub.uxrocket.ru
Username: uxrocket
Password:
…
```

```
Login Succeeded
```

11. Запустите контейнеры Backend Services UXRocket

```
cd /docker-app/uxrocket-backend
docker-compose up -d

root@ux-nt-appext:/docker-app/ux-backend# cd /docker-app/ux-backend
root@ux-nt-appext:/docker-app/ux-backend# docker-compose up -d
[+] Building 0.0s (0/0)
[+] Running 3/3
 ✓ Container ux-api-mobileapi          Started
0.3s
 ✓ Container ux-api-saverawdataapi     Started
0.3s
 ✓ Container ux-api-getsitescriptsapi  Started
```

12. Основные конфигурационные файлы настроек Backend Services UXRocket

```
/uxrocket-backend/cpaapi/appsettings.Production.json
/uxrocket-backend/partnerdataimport/appsettings.Production.json
/uxrocket-backend/ux-api-hostedservices/appsettings.Production.json
```

13. Проверьте что docker-контейнеры запустились

```
sleep 10 && docker ps
```

Установка Frontend Services (портал)

Перед началом работ убедитесь, что виртуальная машина соответствует общим требования.

1. Подключитесь к серверу, на котором будет развернут Frontend Services, по SSH.
2. Перейдите в режим суперпользователя (переключиться в root)

```
sudo su -
```

3. Далее создайте каталог, в котором будут развернуты Frontend сервисы с помощью docker.

```
mkdir -p /docker-app
```

4. Перейдите в созданный каталог

```
cd /docker-app
```

5. Далее скачайте на сервер файлы (конфиги, docker-compose), необходимые для работы Frontend Services UXRocket, из git-репозитория.

```
git clone https://gitflic.uxrocket.ru/project/adminuser/ux-onpremise-frontend.git
```

Результат выполнения команды git clone

```
Cloning into ' ux-onpremise-frontend'...
Username for 'https://gitflic.uxrocket.ru': ***@****.ru
Password for 'https://***@****.ru@gitflic.uxrocket.ru':
```

6. Распакуйте tgz-архив скачанный с помощью команды git clone

```
/docker-app# cd ux-onpremise-frontend /
/docker-app/ux-onpremise-frontend # tar -xzvpf uxrocket-frontend.tgz
```

7. Переместите распакованный каталог uxrocket-frontend в /docker-app/

```
/docker-app/ux-onpremise-frontend # mv uxrocket-frontend ../
```

8. Пример структуры /docker-app/uxrocket-frontend после выполнения команды git clone

```
├── cpaadminweb
│   ├── appsettings.Staging.json
│   └── cpaadminweb.log
├── cpaclientweb
│   ├── appsettings.Staging.json
│   └── cpaclientweb.log
├── docker-compose.yml
├──.env
```

9. Авторизуйтесь в docker registry

```
docker login https://dockerhub.uxrocket.ru
Результат:
docker login https://dockerhub.uxrocket.ru
Username: uxrocket
Password:
…
Login Succeeded
```

14. Отредактируйте файл .env (/docker-app/uxrocket-backend/.env), который отвечает за сопоставления имен хостов и IP-адресов для сервисов UX Rocket.

```
nano /docker-app/uxrocket-backend/.env

# .env
# Настройки IP-адресов
IPappext=192.168.1.16
IPappint=192.168.1.17
IPclickhouse=192.168.1.18
IPfront=192.168.1.21
IPkafka=192.168.1.23
IPpostgresql=192.168.1.27
```

10. Запустите контейнеры Frontend Services UXRocket

```
cd /docker-app/uxrocket-frontend
docker-compose up -d
```

11. Основные конфигурационные файлы настроек Frontend Services UXRocket

```
/docker-app/uxrocket-frontend/cpaadminweb/appsettings.Production.json
/docker-app/uxrocket-frontend/cpaclientweb/appsettings.Production.json
```

14. Проверьте что docker-контейнеры запустились

```
sleep 10 && docker ps
```

Публикация сервиса UX Rocket

Общие требования к публикации

Для публикации UX Rocket рекомендуется использовать Nginx. В зависимости от решения, которое используется для публикации сервисов в корпоративной сети, функционал обратного прокси-сервера может быть совмещён со функционалом интернет-шлюза.

Рекомендуется использовать https при публикации сервисов.

После публикации External Backend Services (API) должно быть публично доступно из сети интернет. Доступ к Frontend Services может быть, как публичным, так и ограниченным корпоративной сетью заказчика. Остальные виртуальные машины и сервисы НЕ должны быть доступны из публичной сети интернет.

Для примера будут использованы следующие значения:

Внешний URL	Описание	Внутренний IP-адрес:Порт
lk-uxrocket.example.ru	кабинет пользователя	192.168.1.21:8080
admin-uxrocket.example.ru	кабинет администратора	192.168.1.21:7500
api-uxrocket.example.ru	Backend	192.168.1.16:8083

Обязательно замените значение для «Внешний URL» и «Внутренний IP» на ваши актуальные параметры.

Примеры конфигов Nginx для Backend и Frontend сервисов приведены ниже.

Пример конфигурации Nginx для Frontend Services

```
# redirect http to https
server {
	listen 80;
	server_name admin-uxrocket.example.ru;
	return 302 https://admin-uxrocket.example.ru$request_uri;
}

#https
server {
	listen 443 ssl http2;
	server_name admin-uxrocket.example.ru;
	client_max_body_size 20m;
#	 Let's Encrypt wildcard ssl certificate
	ssl_certificate /etc/letsencrypt/live/astra.uxrocket.ru/fullchain.pem;
	ssl_certificate_key /etc/letsencrypt/live/astra.uxrocket.ru/privkey.pem;
	include /etc/letsencrypt/options-ssl-nginx.conf; # managed by Certbot
	ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed by Certbot
	add_header X-Content-Type-Options nosniff;
```

```
        add_header X-XSS-Protection "1; mode=block";
        add_header X-Frame-Options SAMEORIGIN;
## Compression.
        gzip on;
        gzip_comp_level 5;
        gzip_min_length 10240;
        gzip_proxied expired no-cache no-store private auth;
        gzip_types text/plain text/css text/xml  application/xml;
        gzip_disable "msie6";
        location  / {
        proxy_pass http://192.168.1.21:7500;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection 'upgrade';
        proxy_set_header Host $host;
        proxy_cache_bypass $http_upgrade;
        proxy_set_header   X-Forwarded-For $proxy_add_x_forwarded_for;
        }
}

# redirect http to https
server {
        listen 80;
        server_name lk-uxrocket.example.ru;
        return 302 https://lk-uxrocket.example.ru$request_uri;
}

#https
server {
        listen 443 ssl http2;
        server_name lk-uxrocket.example.ru;
        client_max_body_size 20m;
#       Let's Encrypt wildcard ssl certificate
        ssl_certificate /etc/letsencrypt/live/astra.uxrocket.ru/fullchain.pem;
        ssl_certificate_key /etc/letsencrypt/live/astra.uxrocket.ru/privkey.pem;
        include /etc/letsencrypt/options-ssl-nginx.conf; # managed by Certbot
        ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed by Certbot
#
        location  / {
        proxy_pass http://192.168.1.21:8080;
        proxy_set_header   X-Forwarded-For $proxy_add_x_forwarded_for;
        }
}
```

Пример конфигурации Nginx для Backend Services

```
# redirect http to https
server {
        listen 80;
```

```
        server_name api-uxrocket.example.ru;
        return 301 https://api-uxrocket.example.ru$request_uri;
}

#https
server {
        listen 443 ssl http2;
        server_name api-uxrocket.example.ru;
        client_max_body_size 20m;
#        Let's Encrypt wildcard ssl certificate
        ssl_certificate /etc/letsencrypt/live/astra.uxrocket.ru/fullchain.pem;
        ssl_certificate_key /etc/letsencrypt/live/astra.uxrocket.ru/privkey.pem;
        include /etc/letsencrypt/options-ssl-nginx.conf; # managed by Certbot
        ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed by Certbot
#
        location  / {
        proxy_pass http://192.168.1.21:8083/;
        proxy_set_header   X-Forwarded-For $proxy_add_x_forwarded_for;
        # Proxy timeouts
        proxy_connect_timeout           60s;
        proxy_send_timeout              60s;
        proxy_read_timeout              60s;
        }
}
```

Результат установки и доступ

После выполнения всех выше указанных шагов в вашем контуре будет развернуто программное обеспечение UX Rocket. Для доступа нужны использовать следующие URL

https://lk-uxrocket.example.ru – кабинет пользователя UX Rocket;

https://admin-uxrocket.example.ru – кабинет администратора UX Rocket.

Обязательно изучите раздел «Публикация сервиса UX Rocket».

Вход в кабинет администратора с логином admin@uxrocket.ru с паролем admin.

В кабинете администратора Вам потребуется завести кабинет пользователя и назначить для него администратора (см. раздел «Настройка кабинета пользователя»).

Сводная таблица по развернутым сервисам

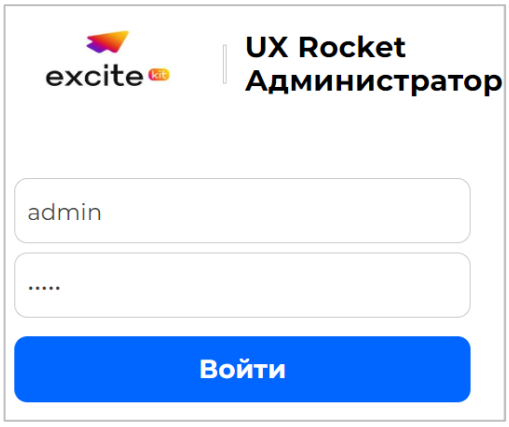
Виртуальная машина	**Название docker-контейнеров и используемые порты**	**Примечание**
PostgreSQL	postgre 5432	
Clickhouse	clickhouse-server 8123, 9000, 9009	
Apache Kafka	kafka0 2181, 9092, 9093 kafka-ui 8080	

Frontend Services	cpaclientweb 7000 cpaadminweb 7500 uxdemo 8082 Nginx 8080	На порт 8080 (http) приходят запросы, отправленные на lk-uxrocket.example.ru (см. Публикация Frontend Services)
Backend Services	Nginx 8083 uxnginx 80 getsitescriptsapi 6500 saverawdataapi 6000 mobileapi 9000 cpaapi 8000 partnerdataimport 8500 uxftp 21	На порт 8083 (http) приходят запросы, отправленные на api-uxrocket.example.ru (см. Публикация Backend Services)

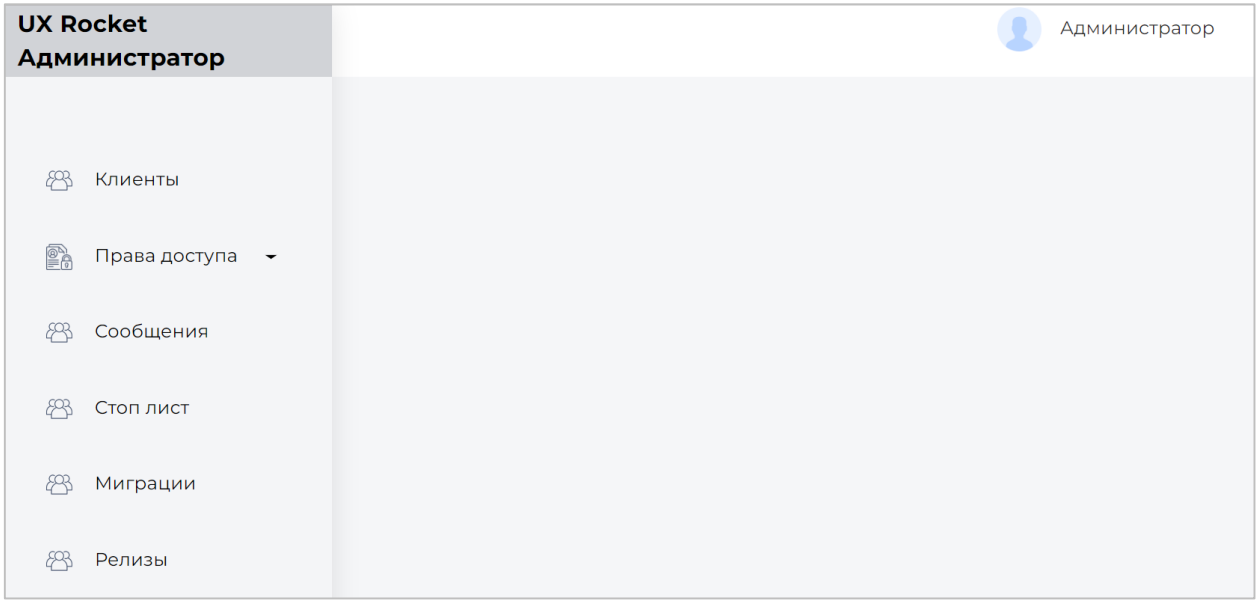
Настройка кабинета пользователя

После установки системы UX Rocket Вам надо создать личный кабинет пользователя (или несколько личных кабинетов). Для создания кабинета пользователя вы добавляете в кабинете администратора нового клиента, а система автоматически создаёт для этого клиента личный кабинет. Далее в кабинете пользователя можно заводить пользователей кабинета и назначать им права доступа.

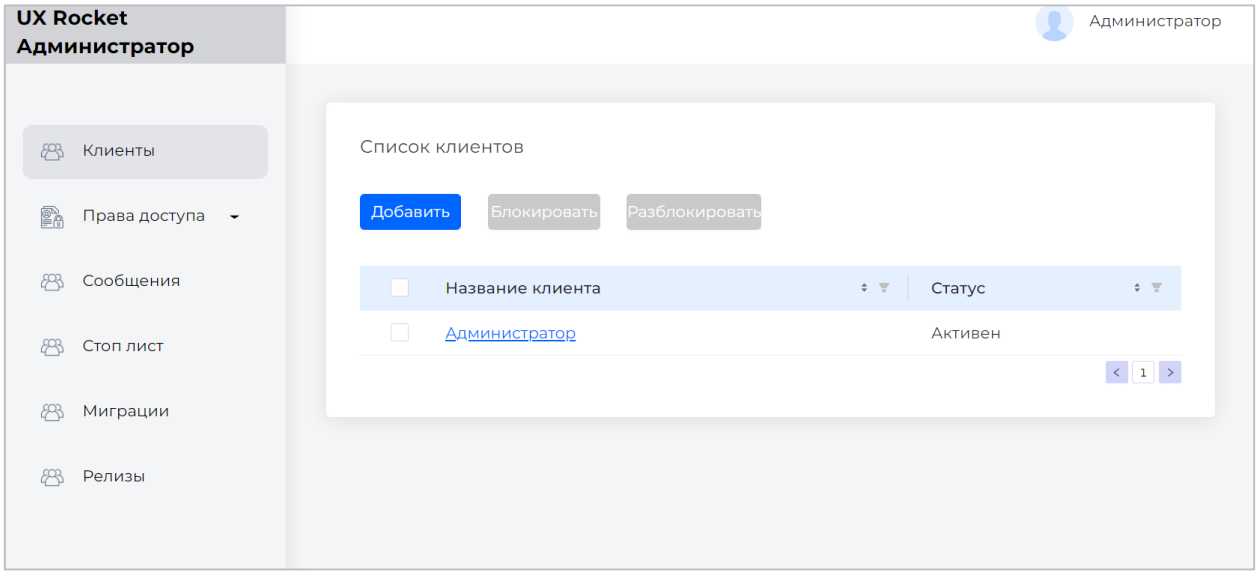
Входим в кабинет администратора по ссылке https://admin-uxrocket.example.ru, указанной в разделе «Результат установки и доступ», и попадаем на экран авторизации.



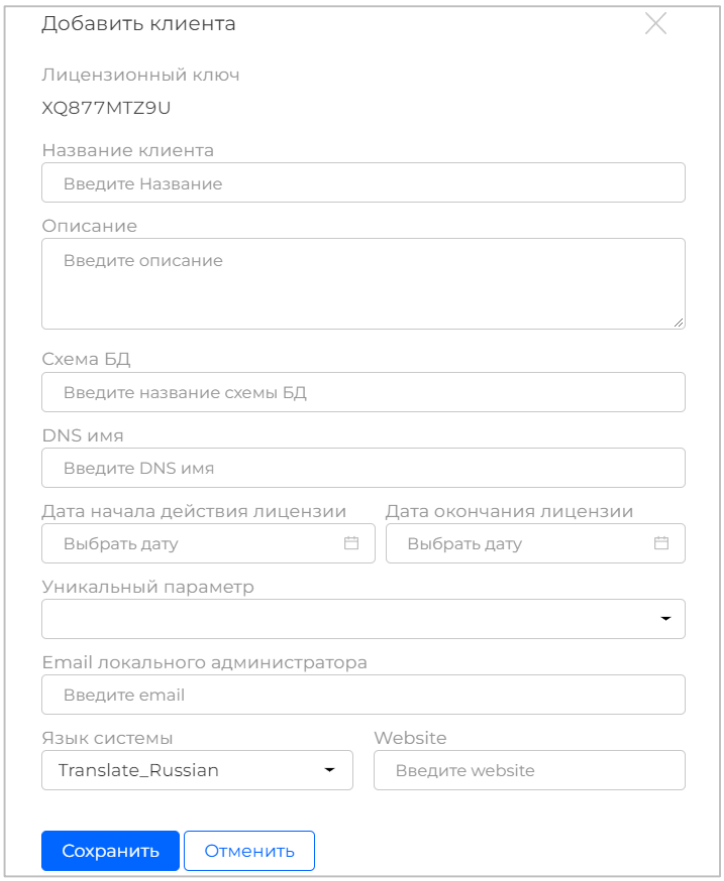
Вводим логин admin@uxrocket.ru с паролем admin и попадаем на главный экран кабинета администратора.



Выбираем пункт меню «Клиенты» и попадаем на экран «Список клиентов». По умолчанию в системе заведён клиент «Администратор», он нужен для работы кабинета администратора. **Запись «администратор» нельзя ни удалять, ни блокировать!**

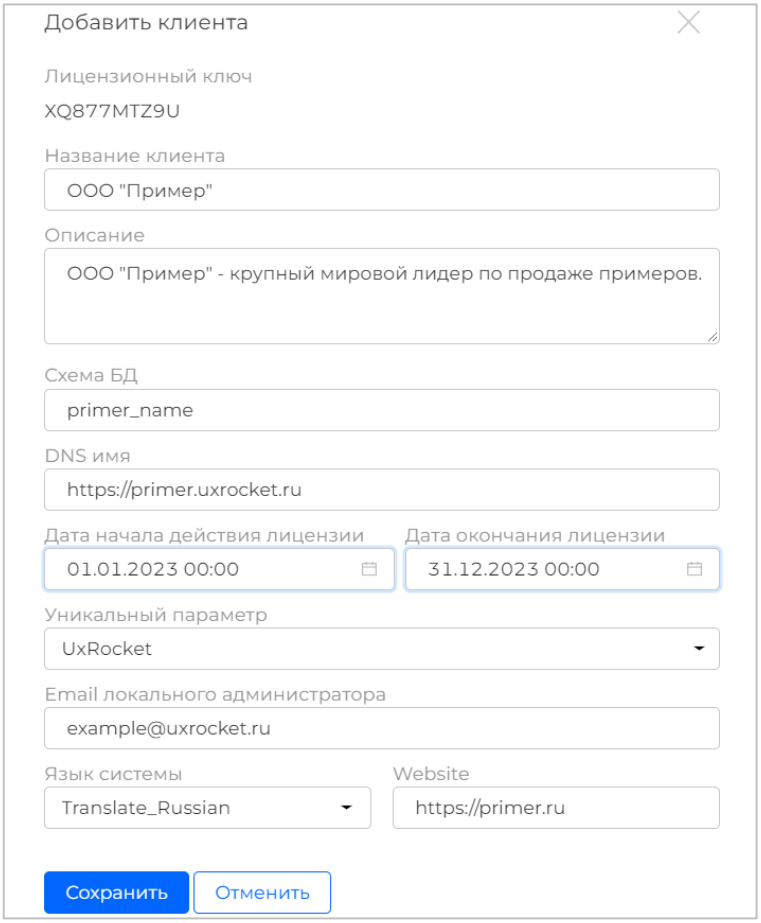


Для добавления нового клиента нажмите кнопку «Добавить». Система откроет окно для добавления нового кабинета клиента:

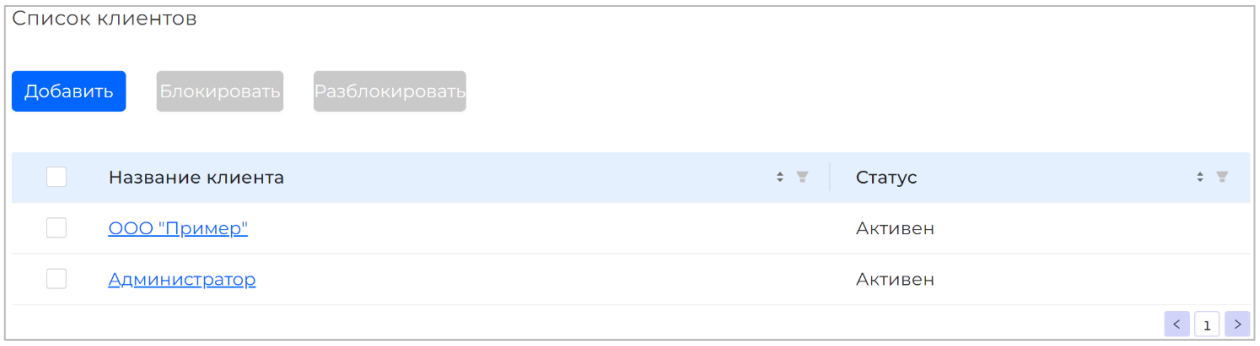


Все поля карточки клиента обязательны для заполнения. Поле «Схема БД» должна иметь буквенно-цифровой формат. Поле «DNS имя» должно иметь валидный https адрес в формате https://{name}.uxrocket.ru (поле нужно для облачной версии, поэтому укажите произвольный параметр name). Поле «Website» должно иметь валидный адрес.

Пример заполнения экрана показан на рисунке ниже.



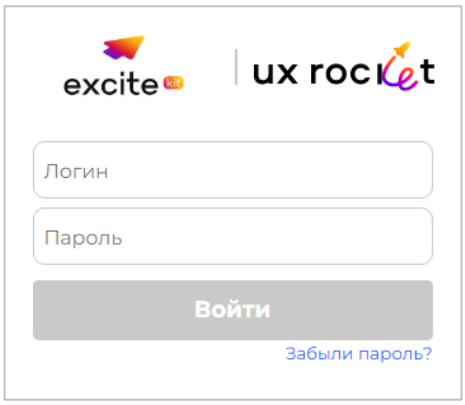
После нажатия кнопки «Сохранить» создаст нового клиента, новый кабинет клиента



и отправит письмо администратору кабинета на почту из поля «Email локального администратора». Письмо содержит ссылку для ввода пароля и входа в кабинет пользователя. Ссылка действительна 1 час.

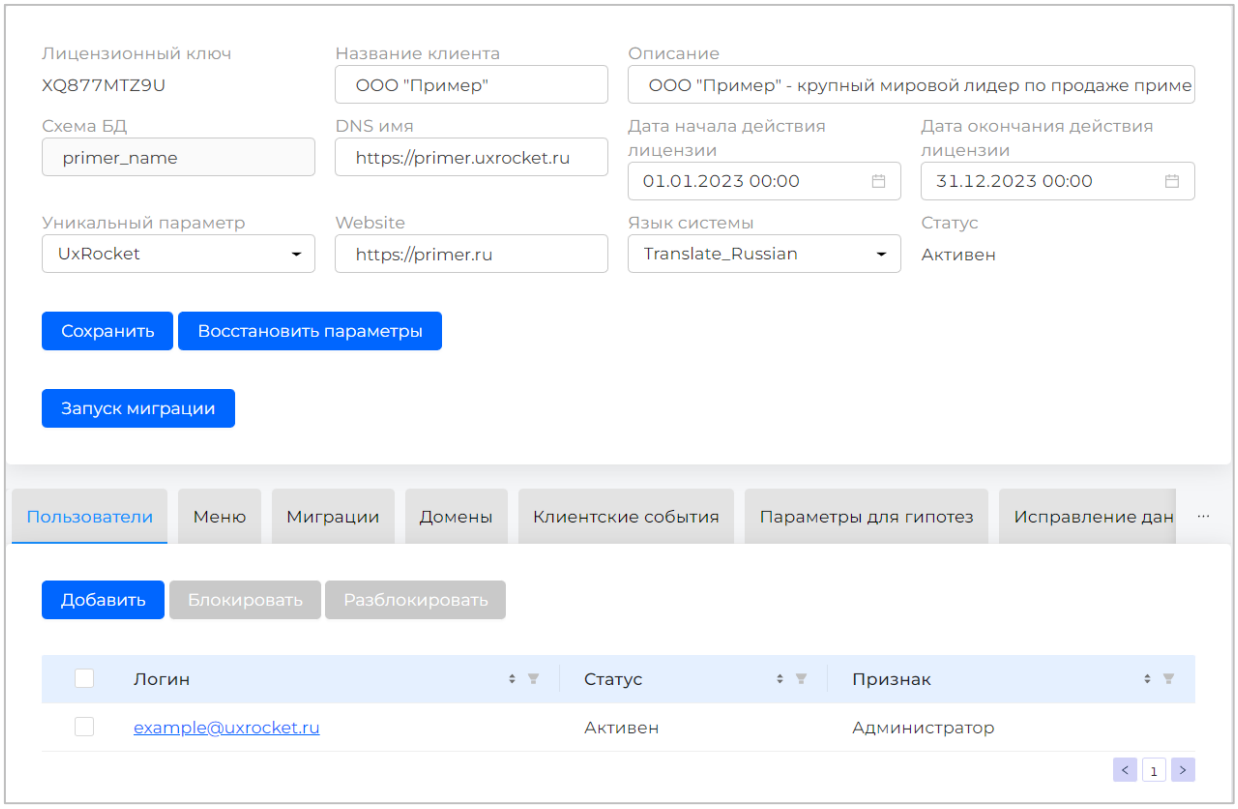
Если ссылка для стала неактивна, то для получения повторной ссылки перейдите в личный кабинет пользователя lk-uxrocket.example.ru и на экране авторизации

1) Введите в поле логин email администратора;
2) Нажмите ссылку «забыли пароль».

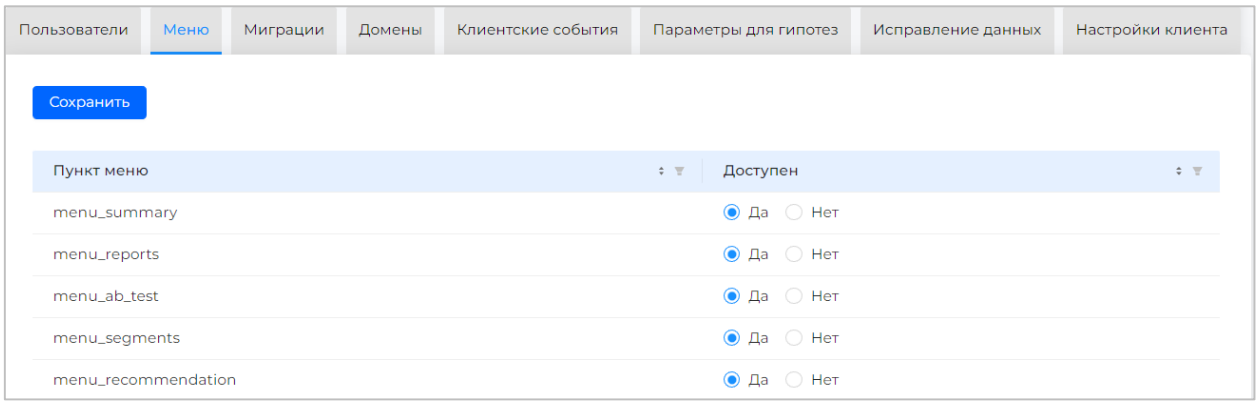


Система проверит почтовый адрес, указанный в поле «Логин», и, если он есть среди списка пользователей любого из кабинетов пользователя, то направит письмо со ссылкой для смены/заведения пароля.

В кабинете администратора после создания нового клиента надо настроить его параметры. Для этого в списке клиентов нажмите на название в клиента. Система откроет карточку клиента.



Перейдите на вкладку «Меню» и в столбце «Доступ» укажите пункты меню, которые надо сделать доступными в кабинете клиента.



Первоначальная настройка системы закончена. Вы можете зайти в кабинет пользователя и начать работу с системой UX Rocket. Документ «Руководство по работе с UX Rocket.docx» доступен на сайте производителя системы по ссылке https://uxrocket.ru/documentation .

Контактная информация

Связаться со специалистами службы технической поддержки ООО «ЭКСАЙТ КИТ» можно одним из следующих способов:

- **Сайт:** https://uxrocket.ru/
- **Телефон:** +7 (495) 725-43-76
- **Email:** support@excitekit.ru

Фактический адрес размещения службы поддержки: РФ, 109544, Москва г, Энтузиастов б-р, дом № 2, комната 47,48,49