Лабораторная информационная медицинская система.

Инструкция по установке.

УТВЕРЖДАЮ

Генеральный директор

ООО «ИНФОРМАЦИОННЫЕ СИСТЕМЫ В ЗДРАВООХРАНЕНИИ»

_____/*М.С. Баканов*/

«___»_____г.

2022, Москва

Содержание

1 Введение

Данный документ содержит порядок установки и настройки лабораторной информационной медицинской системы (далее по тексту - «системы»).

1.1 Назначение и функции комплекса.

Комплекс предназначен для автоматизации приёма, единого учета и обработки заявок на лабораторно-диагностические исследования, проведения и контроля исследований.

1.2 Компоненты комплекса.

- система регистрации заказов (далее – СРЗ) - реализация возможностей приема биоматериала и регистрации заказов на лабораторные исследования непосредственно в лаборатории;
- система для лабораторных исследований (далее – СЛИ) - реализация возможности исследований состава получаемых для биоматериалов, смывов, проб почвы;
- система доставки результатов (далее – СДР) - реализация возможности доставки заказчикам исследований результатов этих исследований в требуемой форме заказчику исследования;
- система контроля качества лабораторных исследований (далее – СККЛИ) - реализация возможности оценки качества получаемых результатов лабораторных исследований и технологических процессов лаборатории;
- система отчетности (далее – СО) - реализация возможности получения оперативной статистической и аналитической отчетности по функционированию всех систем Комплекса и технологическим процессам лаборатории.

2 Требования для установки комплекса

Для установки и настройки системы существуют обязательные минимально необходимые требования.

2.1 Требования к квалификации специалиста

Настройку и установку всех компонентов и подсистем, входящих в систему должен выполнять технический специалист, имеющий соответствующую квалификацию по установке серверных систем.

Технический специалист должен отвечать следующим квалификационным требованиям:

- иметь опыт администрирования Windows, Linux, Oracle;
- владеть знаниями в области сетевых технологий;
- иметь знания в области Java-технологий;
- владеть методами системного администрирования.

2.2 Требования к аппаратному обеспечению

Сервер приложений системы для лабораторных исследований:

- Центральный процессор Intel® Xeon® E5-2697A v4 с частотой 2,6 ГГц ;
- Оперативная память 64 Гб 2,4 ГГц;
- HDD 250 Гб;

Сервер подсистемы интеграции:

- Центральный процессор Intel® Xeon® E5-2670 с частотой 2,6 ГГц ;
- Оперативная память 64 Гб 2,4 ГГц;
- HDD 250 Гб;

Инструментальный сервер:

- Центральный процессор Intel® Xeon® E5-2670 с частотой 2,6 ГГц ;
- Оперативная память 32 Гб 2,4 ГГц;
- HDD 500 Гб;

Сервер базы данных комплекса Master:

- Центральный процессор Intel® Xeon® E5-2697A v4 с частотой 2,6 ГГц ;
- Оперативная память 64 Гб 2,4 ГГц;
- HDD 2000 Гб;

Сервер базы данных комплекса Slave:

- Центральный процессор Intel® Xeon® E5-2697A v4 с частотой 2,6 ГГц;
- Оперативная память 64 Гб 2,4 ГГц;
- HDD 2000 Гб;

Сервер базы данных комплекса Slave:

- Центральный процессор Intel® Xeon® E5-2697Av4 с частотой 2,6 ГГц;
- Оперативная память 64 Гб 2,4 ГГц;
- HDD 500 Гб;

АРМ пользователя:

- Центральный процессор Intel с частотой 2,0 ГГц
- Оперативная память 4 Гб
- HDD 20 Гб
- При использовании Linux необходим перечень принтеров для непосредственной настройки драйвера печати.

2.3 Необходимое ПО

Следующее ПО необходимо установить перед основной установкой компонентов системы:

- на уровне сервера приложений: Linux Ubuntu версии 18.04 или выше;
- на уровне сервера web и почтового прокси-сервера: Nginx версий 1.18.0 или выше;
- PostgreSQL
- MongoDB;
- на уровне поисковых систем: ElasticSearch;

2.4 Требования к каналам связи

Способность систем к обмену данными должна учитывать возможность использования каналов со скоростью передачи не выше 100 Мбит/с.

2.5 Комплектность инсталляционного пакета

Инсталляционный пакет состоит из:

- Установка и конфигурация MongoDB;

- Установка и конфигурация RabbitMQ;

- Запуск и конфигурация инструментальной службы;

- Запуск и конфигурация «Процессора аудита»

3 Настройка системы

3.1 Просмотр версии ОС

Информация о выпуске:

lsb_release -dc

Разрядность ОС:

uname –m

3.2 Установка и конфигурация MongoDB

Руководство по установке расположено по адресу:

https://docs.mongodb.com/manual/tutorial/install-mongodb-on-ubuntu/

3.2.1 Установка MongoDb с помощью скрипта

Скрипт установки позволяет установить MongoDb на Ubuntu версий "16.04", "18.04" и "20.04".
Для установки БД на другую платформу обратитесь к официальной документации.

Перейдите в каталог расположения скрипта установки **"mongo-install.sh"**, *<scriptFolder>* - путь до директории с файлом скрипта

cd <scriptFolder>

Выполните запуск скрипта

sudo sh ./mongo-install.sh

Код скрипта "mongo-install.sh"

```
#!/bin/sh
#run script command 'sudo sh ./mongo-install.sh'
# Check 'gnupg'  install
 dpkg -s "gnupg" &> /dev/null
 if [ $? -ne 0 ]
    then
    echo "Installing 'gnupg' ..."
    sudo apt-get install -y gnupg
 fi


#Import the public key used by the package management system
wget -qO - https://www.mongodb.org/static/pgp/server-4.4.asc | sudo apt-key add -
Version=$(lsb_release -sr)
#Create a list file MongoDB for specific version Ubuntu
case $Version in
"16.04")
        echo "deb [ arch=amd64,arm64 ] https://repo.mongodb.org/apt/ubuntu xenial/mongodb-org/4.4 multiverse" | sudo tee /etc/apt/sources.list.d/mongodb-org-4.4.list
        ;;
"18.04")
        echo "deb [ arch=amd64,arm64 ] https://repo.mongodb.org/apt/ubuntu bionic/mongodb-org/4.4 multiverse" | sudo tee /etc/apt/sources.list.d/mongodb-org-4.4.list
```

```
        ;;
"20.04")
        echo "deb [ arch=amd64,arm64 ] https://repo.mongodb.org/apt/ubuntu focal/mongodb-org/4.4 multiverse" | sudo tee /etc/apt/sources.list.d/mongodb-org-4.4.list
        ;;
*)
        echo "We can't install mongodb automatically for this version Ubuntu. Please go to the official site MongoDb"
        exit 1
        ;;
esac
# Install MongoDb
sudo apt-get update -y
sudo apt-get install -y mongodb-org
# Prevent unintended upgrades
echo "mongodb-org-server hold" | sudo dpkg --set-selections
echo "mongodb-org-shell hold" | sudo dpkg --set-selections
echo "mongodb-org-mongos hold" | sudo dpkg --set-selections
echo "mongodb-org-tools hold" | sudo dpkg --set-selections
#Register service
sudo systemctl daemon-reload
#Enable start after reboot
sudo systemctl enable mongod
# Start MongoDb
echo "Starting MongoDb.."
sudo systemctl restart mongod
```

3.2.2 Настройка и инициализация базы данных MongoDb

Для корректной работы системы необходимо предварительно создать следующие базы данных и коллекции.

Для выполнения управляющих комманд базы *Mongo* откройте оболочку *Mongo Shell.* Для этого откройте терминал и выполните команду:

```
mongo
```

Создайте базу данных *"AuditDb".* Для этого выполните команду в оболочке *Mongo Shell:*

```
use AuditDb
```

В базе "AuditDb" создайте capped коллекции "DeviceLogEvents", "DeviceMessages", "DeviceMetrics", выполнив последовательно команды и указав параметр <size> - максимальный размер коллекции в байтах. Например 10000000

db.createCollection("DeviceLogEvents", {capped: true, size: <size>})

db.createCollection("DeviceMessages", {capped: true, size: <size>})

db.createCollection("DeviceMetrics", {capped: true, size: <size>})

1. Создайте индексы по коллекциям, выполнив последовательно команды:

db.DeviceLogEvents.createIndex({ "deviceid": -1, "time.Ticks":-1, "message":-1}, {unique: false})

db.DeviceMessages.createIndex({ "deviceid": -1, "time":-1, "text":-1 }, {unique: false})

db.DeviceMetrics.createIndex({ "time":-1, "deviceid": -1}, { unique: false})

3.3 Установка RabbitMQ

Для установки RabbitMQ с помощью скрипта выполните следующие команды

Перейдите в каталог расположения скрипта установки **"rmq-install.sh"**, <*scriptFolder*> - путь до директории с файлом скрипта

cd <scriptFolder>

Выполните запуск скрипта

sudo sh ./rmq-install.sh

Код скрипта **"rmq-install.sh"**

```
#!/bin/sh
#run script command 'sudo sh ./rmq-install.sh'
sudo apt-get update -y
## Install prerequisites
sudo apt-get install curl gnupg -y
## Install RabbitMQ signing key
curl -fsSL https://github.com/rabbitmq/signing-keys/releases/download/2.0/rabbitmq-release-signing-key.asc | sudo apt-key add -
## Install apt HTTPS transport
sudo apt-get install apt-transport-https
## Add Bintray repositories that provision latest RabbitMQ and Erlang 23.x releases
sudo tee /etc/apt/sources.list.d/bintray.rabbitmq.list <<EOF
## Installs the latest Erlang 23.x release.
```

Change component to "erlang-22.x" to install the latest 22.x version.

"bionic" as distribution name should work for any later Ubuntu or Debian release.

See the release to distribution mapping table in RabbitMQ doc guides to learn more.

deb https://dl.bintray.com/rabbitmq-erlang/debian bionic erlang

Installs latest RabbitMQ release

deb https://dl.bintray.com/rabbitmq/debian bionic main

EOF

Update package indices

sudo apt-get update -y

Install rabbitmq-server and its dependencies

sudo apt-get install rabbitmq-server -y --fix-missing

 # enable managment plagin

rabbitmq-plugins enable rabbitmq_management

 # enable web stomp plugin

rabbitmq-plugins enable rabbitmq_web_stomp

echo "Starting server.."

service rabbitmq-server start

3.3.1 Конфигурация RabbitMQ

Необходимо создать файл конфигурации definitions.json.

{

"rabbit_version": "3.8.8", //TODO указать версию RMQ

"users": [],

"vhosts": [],

"permissions": [],

"parameters": [],

"policies": [],

"queues": [],

"exchanges": [],

"bindings": []

}

Далее надо добавить секцию **queues** и определить очереди сообщений, указать **<serverId>** - **Идентификатор инструментальной службы** (уникальный идентификатор ИС в рамках модели ЛИС):
Пример, для сервера с идентификатором "3" название очереди конфигурации примет вид ***"ConfigQueueServer_<serverId>"*** → ***"ConfigQueueServer_3"***

"queues": [{

```
        //очередь результатов с устройств
        "name": "results_queue",
        "vhost": "/",
        "durable": true,
       "auto_delete": false,
        "arguments": {
            "x-queue-type": "classic"
        }
    },    {
        //очередь результатов контроля качества с устройств
        "name": "quality_results_queue",
        "vhost": "/",
        "durable": true,
        "auto_delete": false,
        "arguments": {
            "x-queue-type": "classic"
        }
    },
    {
        //очередь аудита сообщений отправляемых устройством
        "name": "device_audit_queue",
        "vhost": "/",
        "durable": true,
       "auto_delete": false,
        "arguments": {
            "x-queue-type": "classic"
    }
    },
    {
        //очередь аудита сообщений отправляемых в терминал к устройству
        "name": "to_device_audit_queue",
        "vhost": "/",
        "durable": true,
        "auto_delete": false,
        "arguments": {
```

```
            "x-queue-type": "classic"
        }
    },
    {
        //очередь задач инструментальной службы
        "name": "TasksQueueServer_<serverId>", //<serverId> - идентификатор сервера в БД ЛИС
        "vhost": "/",
        "durable": true,
        "auto_delete": false,
        "arguments": {
            "x-queue-type": "classic"
        }
    },
    {
        //очередь подверждения результатов инструментальной службы
        "name": "ApproveDeviceResults",
        "vhost": "/",
        "durable": true,
        "auto_delete": false,
        "arguments": {
            "x-queue-type": "classic"
        }
    },
    {
        //очередь сбора метрик с устройств
        "name": "DeviceMetricsQueue",
        "vhost": "/",
        "durable": true,
        "auto_delete": false,
        "arguments": {
            "x-queue-type": "classic"
        }
    },
    {
```

```
        //очередь лога событий с устройств
        "name": "DeviceEventsQueue",
        "vhost": "/",
        "durable": true,
        "auto_delete": false,
        "arguments": {
            "x-queue-type": "classic"
        }
    },
    {
        //Очередь конфигурации сервера в БД ЛИС
        "name": "ConfigQueueServer_<serverId>", //<serverId> - идентификатор сервера в БД ЛИС
        "vhost": "/",
        "durable": true,
        "auto_delete": false,
        "arguments": {
            "x-queue-type": "classic"
        }
    }
    ]
```

1. Добавить секцию **exchanges** и определить конфигурацию обменников:

```
"exchanges": [
    {
        //Обменник аудита сообщений от устройства к драйверу
        "name": "dev_messages_exchange",
        "vhost": "/",
        "type": "topic",
        "durable": true,
        "auto_delete": false,
        "internal": false,
        "arguments": {}
    },
    {
        //обменник задач для драйверов устройств
```

```
        "name": "driver_tasks_exchange",
        "vhost": "/",
        "type": "topic",
        "durable": true,
        "auto_delete": false,
        "internal": false,
        "arguments": {}
    },
    {
        //обменник результатов
        "name": "LisResults",
        "vhost": "/",
        "type": "topic",
        "durable": true,
        "auto_delete": false,
        "internal": false,
        "arguments": {}
    },
    {
        //обменник сообщений (команд) терминала для устройства
        "name": "to_device_messages_exchange",
        "vhost": "/",
        "type": "topic",
        "durable": true,
        "auto_delete": false,
        "internal": false,
        "arguments": {}
    },
    {
        //обменник метрик устройств
        "name": "DeviceMetricsExchange",
        "vhost": "/",
        "type": "topic",
        "durable": true,
        "auto_delete": false,
```

```
        "internal": false,
        "arguments": {}
    },
    {
        //обменник лога событий устройств
        "name": "DeviceEventsExchange",
        "vhost": "/",
        "type": "topic",
        "durable": true,
        "auto_delete": false,
        "internal": false,
        "arguments": {}
    },
    {
        //обменник конфигурации инструментального сервера
        "name": "instrumental_config_exchange",
        "vhost": "/",
        "type": "topic",
        "durable": true,
        "auto_delete": false,
        "internal": false,
         "arguments": {}
    }
]
```

Далее, необходимо добавить секцию **bindings** и определить конфигурацию привязок, указать **<serverId>** - **Идентификатор инструментальной службы** (уникальный идентификатор ИС в рамках модели ЛИС):

```
"bindings": [
    {
        //привязка очереди результатов
        "source": "LisResults",
        "vhost": "/",
        "destination": "results_queue",
        "destination_type": "queue",
        "routing_key": "sample.#",
        "arguments": {}
```

```
},
{
    //привязка очереди результатов контроля качества
    "source": "LisResults",
    "vhost": "/",
    "destination": "quality_results_queue",
    "destination_type": "queue",
    "routing_key": "qc.#",
    "arguments": {}
},
{
    //привязка очереди аудита сообщений от устройств
    "source": "dev_messages_exchange",
    "vhost": "/",
    "destination": "device_audit_queue",
    "destination_type": "queue",
    "routing_key": "from_device.#",
    "arguments": {}
},
{
    //привязка очереди команд терминала к устройству
    "source": "to_device_messages_exchange",
    "vhost": "/",
    "destination": "to_device_audit_queue",
    "destination_type": "queue",
    "routing_key": "to_device.#",
    "arguments": {}
},
{
    //привязка очереди задач для устройств
    "source": "driver_tasks_exchange",
    "vhost": "/",
    "destination": "TasksQueueServer_<serverId>",
    "destination_type": "queue",
    "routing_key": "#.to_server_<serverId>",
    "arguments": {}
},
{
    //привязка очереди подтверждения результатов
    "source": "LisResults",
    "vhost": "/",
    "destination": "ApproveDeviceResults",
    "destination_type": "queue",
    "routing_key": "sample.#",
```

```
        "arguments": {}
    },
    {
        //привязка очереди метрик устройства
        "source": "DeviceMetricsExchange",
        "vhost": "/",
        "destination": "DeviceMetricsQueue",
        "destination_type": "queue",
        "routing_key": "",
        "arguments": {}
    },
    {
        //привязка очереди событий устройства
        "source": "DeviceEventsExchange",
        "vhost": "/",
        "destination": "DeviceEventsQueue",
        "destination_type": "queue",
        "routing_key": "",
        "arguments": {}
    },
    {
        //привязка очереди конфигурации инструментального сервера
        "source": "instrumental_config_exchange",
        "vhost": "/",
        "destination": "ConfigQueueServer_<serverId>",
        "destination_type": "queue",
        "routing_key": "#.server_<serverId>",
        "arguments": {}
    }
]
```

Потом необходимо загрузить файл конфигурации на сервер RabbitMQ с помощью веб-интерфейса (например, http://az-rm01:15672/#/) и на главной вкладке (Overview) перейти к пункту импорта настроек, выбрать файл настроек (см. Рис.1, п.1) и выполнить команду "Upload broker definitions" (см. Рис.1, п.2).

Рисунок 1 - Загрузка файла конфигурации на сервер RabbitMQ.

3.4 Запуск и конфигурация Инструментальной службы

Предусловия: на хост с Linux должен быть установлен dotnet core sdk 3.1 или выше (подробности по установке *https://docs.microsoft.com/ru-ru/dotnet/core/install/linux*).

3.4.1 Конфигурация инструментального сервера

Создать директорию для файлового репозитория драйверов и скопировать все сборки драйверов, сжатые в zip-архив. Маска названия пакета драйвера *{Имя сборки}.{Версия сборки}.zip.*

Скопировать сборку **Invitro.Ethereum.Devices.DriverHost** в произвольную директорию.

В настройках (**appsettings.json)** сборки **Invitro.Ethereum.Devices.DriverHost** указать директорию файлового репозитория пакетов драйверов (*DriverRepositoryFolderPath*) и директорию для распаковки пакетов драйверов (*DriverFolderPath*).
<repositoryFolder> - путь к директории репозитория драйверов,
<driversFolder> - путь к директории распаковки драйверов

```
"DriverRepository": {
  "DriverRepositoryFolderPath": "<repositoryFolder>",
  "DriverFolderPath": "<driversFolder>"
      }
```

Затем необходимо скопировать сборку инструментальной службы **Invitro.Ethereum.InstrumentalApi** в произвольную директорию.

В настройках (**appsettings.json)** сборки **Invitro.Ethereum.InstrumentalApi** указать путь до сборки (*Path*) из *п.3 (Invitro.Ethereum.Devices.DriverHost.dll)* и опционально путь до директории с конфигурациями драйверов (*DriverConfigDirectory*), если планируется использовать конфигурации для драйверов из файла, в секции *DriverHostConfiguration*.

<driverhostFolder> - путь к директории хоста драйвера
<driverConfigFolder> - путь сохранения конфигураций запускаемых устройств (если путь не задан конфигурация передается через командную строку хосту драйвера при запуске)

```
"DriverHostConfiguration": {

    "Path": "<driverhostFolder>/Invitro.Ethereum.Devices.DriverHost.dll",

    "DriverConfigDirectory": "<driverConfigFolder>"

  }
```

В настройках (appsettings.json) сборки Invitro.Ethereum.InstrumentalApi необходимо указать конфигурации очередей RabbitMQ в секции QueuesConfiguration.
TaskQueue - очередь задач из сервиса управления заданиями.

ResultQueue - очередь результатов с приборов.

OutputDeviceMessageQueue - очередь для прослушивания сообщения с устройства в режиме терминала.

InputDeviceMessageQueue - очередь для управления устройством в режиме терминала.

ServerConfigQueue - очередь конфигурации инструментального сервера.

ApproveResultsQueue - очередь подтверждения результатов

DeviceEventsQueue - очередь лога событий устройства.

DeviceMetricsQueue - очередь метрик устройства.

Аббревиатуру **"<serverId>"** необходимо заменить на **Идентификатор инструментальной службы** (уникальный идентификатор ИС в рамках модели ЛИС)

```
"QueuesConfiguration": {
  "TaskQueue": {
    "Host": "rabbit",
    "VHost": "/",
    "User": "guest",
    "Password": "guest",
    "App": "Invitro.Ethereum.InstrumentalApi",
    "Queue": "TasksQueueServer_<serverId>",
    "Channels": 10
  },
  "ResultQueue": {
    "Host": "rabbit",
    "VirtualHost": "/",
    "User": "guest",
    "Password": "guest",
    "ExchangeName": "LisResults",
    "QcRoutingKey": "qc.device_{0}",
    "SampleRoutingKey": "sample.device_{0}"
  },
  "ApproveResultsQueue": {
```

```
  "Host": "localhost",
  "VHost": "/",
  "User": "guest",
  "Password": "guest",
  "App": "Invitro.Ethereum.InstrumentalApi",
  "Queue": "ApproveDeviceResults",
  "Channels": 10
},
"ServerConfigQueue": {
  "Host": "rabbit",
  "VHost": "/",
  "User": "guest",
  "Password": "guest",
  "App": "Invitro.Ethereum.InstrumentalApi",
  "Queue": "ConfigQueueServer_<serverId>",
  "Channels": 1
},
"OutputDeviceMessageQueue": {
  "Host": "rabbit",
  "VirtualHost": "/",
  "User": "guest",
  "Password": "guest",
  "ExchangeName": "dev_messages_exchange",
  "RoutingKey": "from_device.device_{0}",
  "WssUrl": "wss://ethereum.invitro.ru/rmq"
},
"InputDeviceMessageQueue": {
```

```
    "Host": "rabbit",
    "VirtualHost": "/",
    "User": "guest",
    "Password": "guest",
    "ExchangeName": "to_device_messages_exchange",
    "WssUrl": "wss://ethereum.invitro.ru/rmq"
  },
  "DeviceEventsQueue": {
    "Host": "rabbit",
    "VirtualHost": "/",
    "User": "guest",
    "Password": "guest",
    "ExchangeName": "DeviceEventsExchange",
    "ExchangeType": "topic",
    "Port": 5672
  },
  "DeviceMetricsQueue": {
    "Host": "rabbit",
    "VirtualHost": "/",
    "User": "guest",
    "Password": "guest",
    "ExchangeName": "DeviceMetricsExchange"
  }
}
```

В настройках (**appsettings.json)** сборки **Invitro.Ethereum.InstrumentalApi** в секции *InstrumentalServer* указать:
InsturmentalServerId - уникальный идентификатор ИС в рамках модели ЛИС (<serverId>)
CheckDevicesTimeoutSec - интервал (в секундах) переодической проверки

состояния запущенных драйверов
ApplyConfigDelaySec - задержка в секундах между получением новой конфигурации и ее применением

```
"InstrumentalServer": {
      "InstrumentalServerId": <serverId>,
      "CheckDevicesTimeoutSec": 30,
      "ApplyConfigDelaySec": 3
        }
```

В настройках (**appsettings.json)** сборки **Invitro.Ethereum.InstrumentalApi** указать настройки интервала сбора метрик с устройства (в секундах), в секции *DeviceMetrics.*

```
"DeviceMetrics": {
  "CollectIntervalSec": 30
      }
```

В настройках (**appsettings.json)** сборки **Invitro.Ethereum.InstrumentalApi** указать конфигурацию подключения к базе данных **MongoDB** в секции ***ConnectionStrings***.

MongoDbConnection - подключение к БД настроек инструментального сервера.

AuditDbConnection - подключение к БД процессора аудита

```
"ConnectionStrings": {
  "MongoDbConnection": "mongodb://az-rm01:27017/InstrumentalDb",
  "AuditDbConnection": "mongodb://az-rm01:27017/AuditDb"
      }
```

3.4.2 Регистрация инструментальной службы с помощью файла скрипта

Перейдите в каталог расположения сборки инструментально сервера, <*serverFolder*> - путь до директории со сборками инструментального сервера

```
cd <serverFolder>
```

Убедитесь, что в директории сборки инструментального сервера присутствуют файлы **Invitro.Ethereum.Instrumental.service** и **srvinstall.sh** используя команду:

```
ls -la
```

Запустите скрипт на выполнение передав параметр <username> - имя пользователя от имени которого будет зарегистрирована служба

```
sudo sh ./srvinstall.sh "<username>"
```

Содержание файла **Invitro.Ethereum.Instrumental.service**
Параметры <targetdir> и <username> заполняются автоматически при выполнении скрипта

```
[Unit]
```

Description=Invitro.Ethereum.InstrumentalApi

[Service]

WorkingDirectory=<targetdir>

ExecStart=/usr/bin/dotnet <targetdir>/Invitro.Ethereum.InstrumentalApi.dll

SyslogIdentifier=Invitro.Ethereum.InstrumentalApi

User=<username>

 # ensure the service restarts after crashing

Restart=always

 # amount of time to wait before restarting the service

RestartSec=5

KillSignal=SIGINT

Environment=ASPNETCORE_ENVIRONMENT=Production

Environment=DOTNET_PRINT_TELEMETRY_MESSAGE=false

 # This environment variable is necessary when dotnet isn't loaded for the specified user.

 # To figure out this value, run 'env | grep DOTNET_ROOT' when dotnet has been loaded into your shell.

Environment=DOTNET_ROOT=/opt/rh/rh-dotnet31/root/usr/lib64/dotnet

[Install]

WantedBy=multi-user.target

Код скрипта **srvinstall.sh**

#!/bin/bash

#run script command 'sudo sh ./srvinstall.sh "<username>"'

user=$1

servicename="Invitro.Ethereum.Instrumental"

filename="$servicename.service"

servicesdir="/etc/systemd/system/"

installfile="$servicesdir$filename"

if [-z "$user"]

then

echo "Empty argument '<username>'"

echo "Use command: 'sudo sh ./srvinstall.sh <username>'"

exit 1

fi

echo "Start install $servicename"

echo "Copy installfile to $servicesdir.."

cp ./$filename $servicesdir

echo "Modifying install file.."

sed -i 's?<targetdir>?`pwd`'?' $installfile

sed -i "s/<username>/$user/" $installfile

echo "Register service.."

systemctl daemon-reload

echo "Starting service.."

systemctl start $servicename

echo "Add service to autoload.."

systemctl enable $filename

echo "Install $servicename complete"

echo "Use command 'service $servicename status' for check service state"

3.4.3 Регистрация инструментальной службы в ручном режиме

1. Настроить файл **Invitro.Ethereum.Instrumental.service** для запуска сборки **Invitro.Ethereum.InstrumentalApi**, как службы Linux.
   Необходимо задать параметры:
   <targetdir> - директория расположения сборки инструментального сервера
   <username> - имя пользователя системы

[Unit]
Description=Invitro.Ethereum.InstrumentalApi

[Service]
WorkingDirectory=<targetdir>

ExecStart=/usr/bin/dotnet <targetdir>/Invitro.Ethereum.InstrumentalApi.dll
SyslogIdentifier=Invitro.Ethereum.InstrumentalApi

User=<username>

 # ensure the service restarts after crashing
Restart=always
 # amount of time to wait before restarting the service
RestartSec=5

KillSignal=SIGINT
Environment=ASPNETCORE_ENVIRONMENT=Production
Environment=DOTNET_PRINT_TELEMETRY_MESSAGE=false

 # This environment variable is necessary when dotnet isn't loaded for the specified user.
 # To figure out this value, run 'env | grep DOTNET_ROOT' when dotnet has been loaded into your shell.
Environment=DOTNET_ROOT=/opt/rh/rh-dotnet31/root/usr/lib64/dotnet

[Install]
WantedBy=multi-user.target

2. Выполнить команду из директории файлом **Invitro.Ethereum.Instrumental.service** для запуска из systemd:

   sudo cp Invitro.Ethereum.Instrumental.service /etc/systemd/system/Invitro.Ethereum.Instrumental.service

3. Перезагрузить настройки systemd командой:

   sudo systemctl daemon-reload

4. Запустить службу командой:

   sudo systemctl start Invitro.Ethereum.Instrumental

5. Добавить службу в автозапуск:

   systemctl enable Invitro.Ethereum.Instrumental.service

3.5 Запуск и конфигурация "Процессора аудита"

3.5.1 Конфигурация процессора аудита

Скопировать сборку процессора аудита **Invitro.Ethereum.AuditProcessor** в произвольную директорию.

В настройках (**appsettings.json)** сборки **Invitro.Ethereum.AuditProcessor** указать конфигурацию подписчиков, секция *SubscribersConfiguration:*

UpdateDatabaseDelaySec - интервал в течении которого накапливаются данные перед добавлением в БД
QueueConnection - настройки подключения к очереди
DeviceMetrics - конфигурация очереди метрик устройств
FromDeviceMessages - очередь сообщений от устройства к драйверу
ToDeviceMessages - очередь сообщений (команд) терминала к устройству
DeviceEvents - очередь лога событий устройств

```
"SubscribersConfiguration": {
  "DeviceMetrics": {
    "UpdateDatabaseDelaySec": 5,
    "QueueConnection": {
      "Host": "localhost",
      "VHost": "/",
      "User": "guest",
      "Password": "guest",
      "App": "Invitro.Ethereum.AuditProcessor",
      "Queue": "DeviceMetricsQueue",
      "Channels": 10
    }
  },
  "FromDeviceMessages": {
    "UpdateDatabaseDelaySec": 5,
    "QueueConnection": {
      "Host": "localhost",
      "VHost": "/",
      "User": "guest",
      "Password": "guest",
      "App": "Invitro.Ethereum.AuditProcessor",
      "Queue": "device_audit_queue",
      "Channels": 10
    }
  },
  "ToDeviceMessages": {
    "UpdateDatabaseDelaySec": 5,
    "QueueConnection": {
      "Host": "localhost",
      "VHost": "/",
      "User": "guest",
      "Password": "guest",
      "App": "Invitro.Ethereum.AuditProcessor",
      "Queue": "to_device_audit_queue",
      "Channels": 10
    }
  },
  "DeviceEvents": {
    "UpdateDatabaseDelaySec": 5,
    "QueueConnection": {
      "Host": "localhost",
      "VHost": "/",
```

```
    "User": "guest",
    "Password": "guest",
    "App": "Invitro.Ethereum.AuditProcessor",
    "Queue": "DeviceEventsQueue",
    "Channels": 10
  }
}
    }
```

В настройках (**appsettings.json)** сборки **Invitro.Ethereum.AuditProcessor** указать конфигурацию подключения к базе данных MongoDB в секции *ConnectionStrings*.
MongoDbConnection - подключение к БД процессора аудита

```
"ConnectionStrings": {
  "MongoDbConnection": "mongodb://az-rm01:27017/AuditDb"
    }
```

3.5.2 Регистрация процессора аудита как службы

Перейдите в каталог расположения сборки процессора аудита, *<processorFolder>* - путь до директории со сборками процессора аудита

```
cd <processorFolder>
```

Убедитесь, что в директории сборки процессора аудита присутствуют файлы **Invitro.Ethereum.AuditProcessor.service** и **srvinstall.sh** используя команду:

```
ls -la
```

Запустите скрипт на выполнение передав параметр <username> - имя пользователя от имени которого будет зарегистрирована служба

```
sudo sh ./srvinstall.sh "<username>"
```

Содержание файла **Invitro.Ethereum.AuditProcessor.service**
Параметры <targetdir> и <username> заполняются автоматически при выполнении скрипта

```
[Unit]

Description=Invitro.Ethereum.AuditProcessor

[Service]

WorkingDirectory=<targetdir>

ExecStart=/usr/bin/dotnet <targetdir>/Invitro.Ethereum.AuditProcessor.dll -url "http://0.0.0.0:8085"

SyslogIdentifier=Invitro.Ethereum.AuditProcessor
```

User=<username>

 # ensure the service restarts after crashing

Restart=always

 # amount of time to wait before restarting the service

RestartSec=5

KillSignal=SIGINT

Environment=ASPNETCORE_ENVIRONMENT=Production

Environment=DOTNET_PRINT_TELEMETRY_MESSAGE=false

 # This environment variable is necessary when dotnet isn't loaded for the specified user.

 # To figure out this value, run 'env | grep DOTNET_ROOT' when dotnet has been loaded into your shell.

Environment=DOTNET_ROOT=/opt/rh/rh-dotnet31/root/usr/lib64/dotnet

[Install]

WantedBy=multi-user.target

Код скрипта **srvinstall.sh**

#!/bin/bash

#run script command 'sudo sh ./srvinstall.sh "<username>"'

user=$1

servicename="Invitro.Ethereum.AuditProcessor"

filename="$servicename.service"

servicesdir="/etc/systemd/system/"

installfile="$servicesdir$filename"

if [-z "$user"]

then

echo "Empty argument '<username>'"

```
 echo "Use command: 'sudo sh ./srvinstall.sh <username>'"
 exit 1
fi
echo "Start install $servicename"
echo "Copy installfile to $servicesdir.."
cp ./$filename $servicesdir
echo "Modifying install file.."
sed -i 's?<targetdir>?'`pwd`'?' $installfile
sed -i "s/<username>/$user/"  $installfile
echo "Register service.."
systemctl daemon-reload
echo "Starting service.."
systemctl start $servicename
echo "Add service to autoload.."
systemctl enable $filename
echo "Install $servicename complete"
echo "Use command 'service $servicename status' for check service state"
```

3.6 Создание ролей

После первичной установки системы специалист с административной ролью создает следующие роли: Администратор правил, Администратор данных и Администратор (инженер).

На любом ПК в текущей сети перейти, используя веб-браузер на сайт https://test-ethereum.invitro.ru и войти с правами роли Техник.

Перейти в модуль «Безопасность».

В открывшемся модуле «Безопасность» щелкните по кнопке раскрытия списка «Роли» (см. Рис. 3, п.7) и щелкните по пункту «Роли» в открывшемся списке (см. Рис. 3, п.8). Далее щелкните по кнопке «Создать» (см. Рис. 3, п.2).

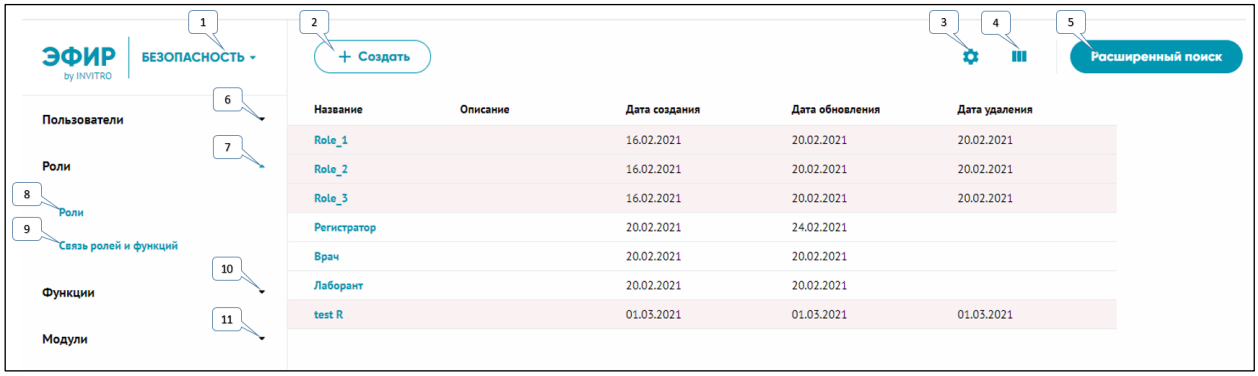

Рисунок 3 – Модуль «Безопасность» (1 – кнопка раскрытия списка модулей, 2 – кнопка «Создать», 3 – кнопка общей настройки отображения данных, 4 – кнопка выбора данных для отображения, 5 – кнопка «Расширенный поиск», 6 – кнопка раскрытия списка «Пользователи», 7 – кнопка раскрытия списка «Роли», 8 - пункт списка «Роли», 9 – пункт списка «Связь ролей и функций», 10 – кнопка раскрытия списка «Функции», 11 – кнопка раскрытия списка «Модули»).

В открывшемся окне создания роли (см. Рис. 4) заполните поле «Название» (см. Рис. 4, п.1) и поле «Описание» (см. Рис. 4, п.2). Щелкните по кнопке «Добавить» в блоке «Пользователи (см. Рис. 4, п.3), выберите из списка пользователя, которому необходимо добавить текущую роль.

Для выбора пользователя отметьте его в списке флажком ☑ и щелкните по кнопке «Сохранить» или по кнопке «Отмена» для отмены добавления пользователя. Аналогично добавьте необходимые для текущей роли функции. Щелкните по кнопке «Сохранить» (см. Рис. 4, п.3).

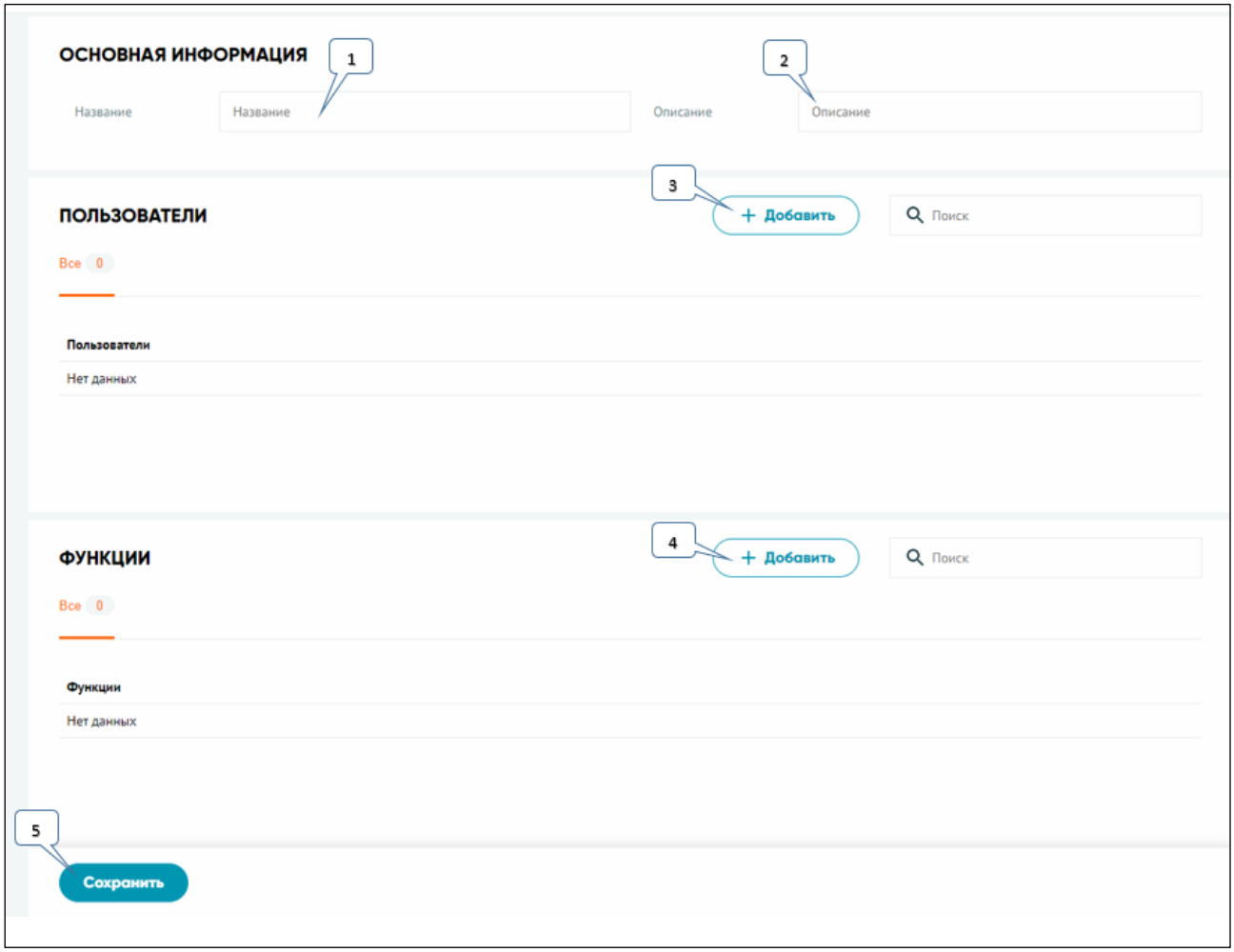

Рисунок 4 – Окно создания роли (1 – поле «Название», 2 – поле «Описание», 3 – кнопка «Добавить» блока «Пользователи», 4 – кнопка «Добавить» блока «Функции», 5 – кнопка «Сохранить»)

3.7 Установка сервиса печати

При необходимости дополнительно можно установить сервис печати.

Информацию по установке дополнительного сервиса можно получить обратившись в техническую поддержку.

4 Список сокращений

Сокращение	Описание
СРЗ	система регистрации заказов
СЛИ	система для лабораторных исследований
СДР	система доставки результатов
СККЛИ	система контроля качества лабораторных исследований
СО	система отчетности
ПО	программное обеспечение
ОС	операционная система
БД	база данных
ПК	Персональный компьютер