Платформа для автоматизации технологических процессов и управления производством

"WISECON"

РУКОВОДСТВО ПО РАЗВЕРТЫВАНИЮ И АДМИНИСТРИРОВАНИЮ ПО WisePredictiveAnalytics, WiseCPM, WiseAlarmManager, WiseMPA, Интегрированное управление турбокомпрессорным оборудованием

версия для ОС Astra Linux

2026

Содержание

СПИСОК СОКРАЩЕНИЙ

БД	База данных
БДРВ	База данных реального времени
ОС	Операционная система
РСУ	Распределенная система управления
СВТ	Средства вычислительной техники
СПУ	Система процедурного управления
СУБД	Система управления базами данных

1. Введение

В настоящем документе приводятся требования к техническому и системному программному обеспечению, необходимому для развертывания модулей "WISECON" WisePredictiveAnalytics, WiseCPM, WiseAlarmManager, WiseMPA и Интегрированное управление турбокомпрессорным оборудованием, а также порядок установки компонентов платформы на СВТ с операционной системой Astra Linux (далее – ОС).

2. Общие сведения

2.1 Архитектура развертывания

Платформа поддерживает следующие архитектуры развертывания:

- **Единый сервер системы.** Все компоненты системы устанавливаются на один сервер системы с графическим интерфейсом операционной системы.
- **Сервер системы и инженерная станция.** Все серверные компоненты устанавливаются на один сервер. Клиентские компоненты устанавливаются на инженерную станцию с графическим интерфейсом операционной системы.
- **Веб-сервер, сервер баз данных и инженерная станция.** Все серверные компоненты устанавливаются на сервер баз данных. Веб-приложение (WisePredictiveAnalytics *InfraPortal)* устанавливается на веб-сервер. Клиентские компоненты устанавливаются на инженерную станцию с графическим интерфейсом операционной системы.
- **Веб-сервер, сервер баз данных, сервер приложений и инженерная станция.** Все серверные компоненты устанавливаются на сервер приложений. Компоненты баз данных устанавливаются на сервер баз данных. Веб-приложение (WisePredictiveAnalytics *InfraPortal)* устанавливается на

веб-сервер. Клиентские компоненты устанавливаются на инженерную станцию с графическим интерфейсом операционной системы.

2.2 Условия, необходимые для выполнения компонентов платформы

Требования к техническому обеспечению СВТ единого сервера, веб-сервера, сервера баз данных, сервера приложений:

- Центральный процессор – частота 2800 МГц и выше;
- Оперативная память – 16 ГБ или больше;
- Жесткий или SSD диск – емкость 300 ГБ или больше;
- Сетевое оборудование – сетевая карта 10/100/1000 Base-T.

Требования к техническому обеспечению СВТ единого сервера, веб-сервера, сервера баз данных, сервера приложений:

- Центральный процессор – частота 2800 МГц и выше;
- Оперативная память – 8 ГБ или больше;
- Жесткий или SSD диск – емкость 100 ГБ или больше;
- Сетевое оборудование – сетевая карта 10/100/1000 Base-T.

Требования к системному программному обеспечению веб-сервера, сервера баз данных, сервера приложений:

- Операционная система Astra Linux SE 1.7 или старше (Desktop/Server), 64 бит;

Требования к системному программному единого сервера, инженерной станции приложений:

- Операционная система Astra Linux SE 1.7 или старше (Desktop) с графическим интерфейсом, 64 бит.

2.2.1 Требования к составу периферийных устройств

Основные требования к составу периферийных устройств не предъявляются.

2.2.2 Требования к параметрам периферийных устройств

Основные требования к параметрам периферийных устройств не предъявляются.

3. Порядок развертывания компонентов платформы

3.1 Добавление доступа к репозиториям

Установка СУБД PostgreSQL требует доступа к репозиториям Astra Linux 1.7. Для настройки репозиториев следует выполнить следующее:

1. Открыть терминал Astra Linux, выполнить команду открытия в текстовом редакторе файла репозиториев:

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

2. Добавить в него список репозиториев:

   при наличии доступа в Интернет вставить в файл следующие строки:

```
# Основной репозиторий
deb
https://dl.astralinux.ru/astra/stable/1.7_x86-64/repository-main/
1.7_x86-64 main contrib non-free
# Оперативные обновления основного репозитория
deb
https://dl.astralinux.ru/astra/stable/1.7_x86-64/repository-update/
1.7_x86-64 main contrib non-free
# Базовый репозиторий
deb
https://dl.astralinux.ru/astra/stable/1.7_x86-64/repository-base/
1.7_x86-64 main contrib non-free
# Расширенный репозиторий
deb
https://dl.astralinux.ru/astra/stable/1.7_x86-64/repository-extende
d/ 1.7_x86-64 main contrib non-free
# Расширенный репозиторий (компонент astra-ce)
deb
https://dl.astralinux.ru/astra/stable/1.7_x86-64/repository-extende
d/ 1.7_x86-64 astra-ce
# Последнее срочное обновление (если доступно):
```

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

при отсутствии доступа к Интернету вставить в файл строки подключения к локальным репозиториям – при наличии доступа к ftp-серверу dln4-lnxupd.srv.lukoil.com вставить такие строки:

```
deb
ftp://dln4-lnxupd.srv.lukoil.com/Astra/dl.astralinux.ru/astra/froze
n/1.7_x86-64/1.7.8/repository-base/ 1.7_x86-64 main contrib
non-free
deb
ftp://dln4-lnxupd.srv.lukoil.com/Astra/dl.astralinux.ru/astra/froze
n/1.7_x86-64/1.7.8/repository-extended/ 1.7_x86-64 main contrib
non-free
deb
ftp://dln4-lnxupd.srv.lukoil.com/Astra/dl.astralinux.ru/astra/froze
n/1.7_x86-64/1.7.8/repository-main/ 1.7_x86-64 main contrib
non-free
deb
ftp://dln4-lnxupd.srv.lukoil.com/Astra/dl.astralinux.ru/astra/froze
n/1.7_x86-64/1.7.8/repository-update/ 1.7_x86-64 main contrib
non-free
```

3. Сохранить и закрыть файл (в программе nano комбинация клавиш Ctrl-O, Ctrl-X).
4. Обновить информацию о пакетах установки, доступных из репозиториев:

```
sudo apt update
```

3.2 Установка СУБД PostgreSQL

Для установки на ОС Astra Linux Special Edition версии 1.7 следует выбирать СУБД PostgreSQL версий 14-14.5, на ОС Astra Linux Special Edition версии 1.8 – СУБД PostgreSQL версии 15 (пакеты для установки входят в

состав соответствующей версии ОС). Также допускается использование СУБД Postgres Pro при условии совместимости версии с ОС.

Порядок установки:

1. Открыть терминал и проверить версию PostgreSQL, доступную из пакета репозитория:

```
apt search --names-only '^postgresql$'
```

Вывод терминала должен содержать строку с указанием версии, доступной для установки:

```
Sorting... Done
Full Text Search... Done
postgresql/stable 14+245+ci11 all
  object-relational SQL database (supported version):
```

2. Установить СУБД PostgreSQL:

```
sudo apt install -y postgresql postgresql-contrib
```

3. Проверить статус службы postgresql.service – должен быть «active»:

```
sudo systemctl status postgresql.service
```

4. Проверить наличие созданного кластера:

```
sudo -u postgres psql
\l
```

Вывод команды «\l» должен показать наличие 3 баз данных: postgres, template0, template1.

5. Задать пароль пользователю postgres:

```
ALTER USER postgres WITH PASSWORD 'postgres';
```

После просмотра информации о созданных базах и задания пароля пользователя postgres ввести команду «\q» для выхода из утилиты psql.

6. В случае необходимости удаленного сетевого доступа к базам данных необходимо настроить доступ к БД:
    a) Перейти в директорию с файлами конфигурации БД:

```
cd /etc/postgresql/14/main/
```

    b) Открыть файл postgresql.conf в текстовом редакторе:

```
sudo nano postgresql.conf
```

c) Найти следующую строку – для поиска в программе nano можно использовать комбинацию клавиш Ctrl-W:

```
#listen_addresses = 'localhost'
```

d) Исправить ее таким образом:

```
listen_addresses = '*'
```

e) Сохранить и закрыть файл postgresql.conf (в программе nano комбинация клавиш Ctrl-O, Ctrl-X).

f) Открыть файл pg_hba.conf в текстовом редакторе:

```
sudo nano pg_hba.conf
```

g) В конце файла найти следующие строки:

```
# IPv4 local connections:
host    all     all     127.0.0.1/32     scram-sha-256
```

h) Исправить вторую строку таким образом:

```
host    all     all     all     scram-sha-256
```

i) Сохранить и закрыть файл pg_hba.conf (в программе nano комбинация клавиш Ctrl-O, Ctrl-X).

j) Перезапустить службу PostgreSQL:

```
sudo systemctl restart postgresql
```

3.3 Установка среды выполнения .NET

Репозитории Astra Linux Special Edition версии 1.7 содержат пакеты .NET версии 6.0, тогда как для запуска компонентов платформы необходима версия .NET 8.0. В связи с этим установка данной версии выполняется вручную из поставляемых deb-пакетов. В репозиториях ОС Astra Linux SE 1.8 версия .NET 8.0 имеется в наличии и должна устанавливаться напрямую из репозитория.

Порядок установки из архива (для Astra Linux SE 1.7):

1. Открыть терминал, перейти в каталог, где расположен файл aspnetcore-runtime-8.0.22-linux-x64.tar.gz, входящий в установочный пакет платформы.
2. Скопировать файл в директорию /opt:

```
sudo cp aspnetcore-runtime-8.0.22-linux-x64.tar.gz /opt
```

3. Выполнить команды для распаковки архива:

```
DOTNET_ROOT=/opt/dotnet
DOTNET_FILE=/opt/aspnetcore-runtime-8.0.22-linux-x64.tar.gz
mkdir -p "$DOTNET_ROOT"
tar zxf "$DOTNET_FILE" -C "$DOTNET_ROOT"
```

4. Для задания переменных окружения на уровне ОС создать файл скрипта с одновременным открытием в текстовом редакторе nano:

```
sudo nano /etc/profile.d/dotnet.sh
```

5. Добавить в файл следующее содержание:

```
export DOTNET_ROOT=/opt/dotnet
export PATH=$PATH:$DOTNET_ROOT:$DOTNET_ROOT/tools
```

6. Сохранить и закрыть файл dotnet.sh (в программе nano комбинация клавиш Ctrl-O, Ctrl-X).
7. Если файл архива больше не нужен, удалить его:

```
rm -f "$DOTNET_FILE"
```

8. Перезагрузить компьютер.

Установка .NET из репозитория для ОС Astra Linux SE 1.8 выполняется следующими командами:

```
sudo apt-get update
sudo apt-get install -y aspnetcore-runtime-8.0
```

3.4 Установка СУБД ClickHouse

СУБД ClickHouse отсутствует в официальных репозиториях Astra Linux, поэтому далее рассмотрена установка из deb-пакетов:

1. Открыть терминал, перейти в каталог, где расположены файлы deb-пакетов, входящие в установочный пакет платформы. Для установки необходимы следующие файлы:
   - clickhouse-common-static_25.10.2.65_amd64.deb
   - clickhouse-server_25.10.2.65_amd64.deb
   - clickhouse-client_25.10.2.65_amd64.deb

Данные файлы обеспечивают установку ClickHouse версии 25.10.2.65.

2. Последовательно запустить установку каждого файла:

```
sudo dpkg -i clickhouse-common-static_25.10.2.65_amd64.deb
sudo dpkg -i clickhouse-server_25.10.2.65_amd64.deb
sudo dpkg -i clickhouse-client_25.10.2.65_amd64.deb
```

В процессе установки сервера будет предложено задать пароль пользователя default – рекомендуется задать пароль «default», так как в конфигурации платформы он используется по умолчанию.

3. Запустить сервер ClickHouse:

```
sudo systemctl start clickhouse-server
```

4. Проверить возможность подключения к ClickHouse из терминала:

```
clickhouse-client --password default
```

Для проверки ввести команду вывода всех имеющихся баз данных:

```
SHOW DATABASES
```

В выводе терминала должны быть показаны 4 базы данных, созданные по умолчанию:

```
   ┌─name───────────────┐
1. │ INFORMATION_SCHEMA │
2. │ default            │
3. │ information_schema │
4. │ system             │
   └────────────────────┘
```

5. Для выхода из консольного клиента ClickHouse ввести команду «exit» или нажать сочетание клавиш Ctrl-D.

3.5 Установка Python

Для работы backend-части InfraPortal требуется среда языка программирования Python версии выше 3.10. В репозиториях ОС Astra Linux 1.7 доступна версия 3.7, которая является устаревшей. В настоящем разделе приведен процесс ручной установки Python версии 3.12 из установочного пакета:

1. Открыть терминал, перейти в каталог, где расположен файл python3.12.tar.gz. Скопировать его в каталог /opt:

```
sudo cp python3.12.tar.gz /opt
```

2. Перейти в каталог /opt, распаковать данный архив, затем удалить копию:

```
cd /opt
sudo tar zxvf python3.12.tar.gz
sudo rm -f python3.12.tar.gz
```

   В результате будет создан каталог /opt/python3.12 со всеми необходимыми файлами Python версии 3.12.

3. Проверить наличие уже установленных версий Python в каталоге /usr/bin:

```
ls -l /usr/bin/ | grep python
```

   В выводе терминала должны присутствовать файлы python2.7 и python3.7.

4. Создать переключатель версий Python для выбора версии 3.12, для чего последовательно выполнить команды:

```
sudo update-alternatives --install /usr/bin/python python
/usr/bin/python2.7 1
sudo update-alternatives --install /usr/bin/python python
/usr/bin/python3.7 2
sudo update-alternatives --install /usr/bin/python python
/opt/python3.12/bin/python3.12 3
```

   Важно не забыть цифру в конце команды, так как она в случае необходимости позволяет переключать текущую версию Python.

5. Для задания переменной окружения на уровне ОС создать файл скрипта с одновременным открытием в текстовом редакторе nano:

```
sudo nano /etc/profile.d/python.sh
```

6. Добавить в файл следующее содержание:

```
export PATH=$PATH:/opt/python3.12/bin/python
```

7. Сохранить и закрыть файл python.sh (в программе nano комбинация клавиш Ctrl-O, Ctrl-X).

8. Перезагрузить компьютер.

9. Открыть терминал и проверить актуальную версию Python, введя следующую команду:

```
python --version
```

В выводе терминала должно отобразиться следующее:

```
Python 3.12.12
```

3.6 Установка Node.js

Для работы frontend-части WisePredictiveAnalytics (InfraPortal) требуется установка Node.js версии выше 20. В репозиториях ОС Astra Linux 1.7 доступна версия 10.24.0, которая является устаревшей. Также следует отметить, что для дальнейшей успешной сборки frontend-части InfraPortal необходимо наличие на компьютере не менее 4 ГБ незанятой оперативной памяти, иначе сборка проекта Node.js может стать невозможной. В настоящем разделе приведен процесс ручной установки Node.js версии 25.2.1 из установочного пакета:

1. Открыть терминал, перейти в каталог, где расположен файл node-v25.2.1-linux-x64.tar.gz. Скопировать его в каталог /opt:

```
sudo cp node-v25.2.1-linux-x64.tar.gz /opt
```

2. Перейти в каталог /opt, распаковать данный архив, затем удалить копию:

```
cd /opt
sudo tar zxvf node-v25.2.1-linux-x64.tar.gz
sudo rm -f node-v25.2.1-linux-x64.tar.gz
```

В результате будет создан каталог /opt/node со всеми необходимыми Node.js версии 25.2.1.

3. Для задания переменной окружения на уровне ОС создать файл скрипта с одновременным открытием в текстовом редакторе nano:

```
sudo nano /etc/profile.d/node.sh
```

4. Добавить в файл следующее содержание:

```
export PATH=$PATH:/opt/node/bin
```

5. Сохранить и закрыть файл node.sh (в программе nano комбинация клавиш Ctrl-O, Ctrl-X).
6. Создать ссылки типа simlink для файлов каталога /opt/node/bin:

```
ln -s /opt/node/bin/node /usr/local/bin/node
ln -s /opt/node/bin/npm /usr/local/bin/npm
ln -s /opt/node/bin/npx /usr/local/bin/npx
```

7. Для использования утилиты npm открыть для изменения файл /opt/node/bin/npm:

```
sudo nano /opt/node/bin/npm
```

8. Изменить содержимое файла npm следующим образом:

```
#!/usr/bin/env node
require('/opt/node/lib/node_modules/npm/lib/cli.js')(process)
```

9. Сохранить и закрыть файл npm (в программе nano комбинация клавиш Ctrl-O, Ctrl-X).
10. Для использования утилиты npx открыть для изменения файл /opt/node/bin/npx:

```
sudo nano /opt/node/bin/npx
```

11. Найти в файле npx следующие строки и изменить их следующим образом:
    - исходная строка (в файле npx строка № 3):

      ```
      const cli = require('../lib/cli.js')
      ```

      строка после изменения:

      ```
      const cli =
      require('/opt/node/lib/node_modules/npm/lib/cli.js')
      ```

    - исходная строка (в файле npx строка № 6):

      ```
      process.argv[1] = require.resolve('./npm-cli.js')
      ```

      строка после изменения:

      ```
      process.argv[1] =
      require.resolve('/opt/node/lib/node_modules/npm/bin/npm-cli.js
      ')
      ```

    - исходная строка (в файле npx строка № 27):

      ```
      const { definitions, shorthands } =
      require('@npmcli/config/lib/definitions')
      ```

строка после изменения:

```
const { definitions, shorthands } =
require('/opt/node/lib/node_modules/npm/node_modules/@npmcli/c
onfig/lib/definitions')
```

12. Сохранить и закрыть файл npm (в программе nano комбинация клавиш Ctrl-O, Ctrl-X).
13. Перезагрузить компьютер.
14. Открыть терминал и проверить актуальную версию Node.js, введя следующую команду:

    ```
    node
    ```

    В выводе терминала должно отобразиться следующее:

    ```
    Welcome to Node.js v25.2.1
    ```

15. Проверить актуальную версию npm, введя следующую команду:

    ```
    npm –version
    ```

    В выводе терминала должно отобразиться следующее:

    ```
    11.6.2
    ```

16. Проверить актуальную версию npx, введя следующую команду:

    ```
    npx –version
    ```

    В выводе терминала должно отобразиться следующее:

    ```
    11.6.2
    ```

4. Установка сервисов на сервер приложений

4.1 Установка сервиса InfoService

Для установки InfoService необходимо скопировать архив InfoService.tar.gz на машину с ОС Astra Linux в нужный каталог, затем распаковать архив следующей командой в терминале:

```
tar zxvf InfoService.tar.gz
```

По умолчанию InfoService имеет базовые настройки, предусматривающие локальное размещение всех сервисов платформы.

В случае, если платформа предусматривает распределенное развертывание, выполните следующие настройки:

1. Перейдите в каталог сервиса и откройте файл appsettings.json в текстовом редакторе (например, Kate).
2. Параметр “Serilog” (Рисунок 4.1.1) отвечает за ведение логов. При необходимости можете изменить периодичность создания записей, поменяв параметр “rollingInterval”. По умолчанию ему присвоено значение “Month” (каждый месяц). Подробнее смотрите документацию по Serilog.
3. В поле “Url” при необходимости измените данные, которые будут использоваться другими сервисами при обращении к InfoService (Рисунок 4.1.1).

```
"AppSettings": {
  "Name": "InfoService",
  "Version": "Dev"
},
"Serilog": {
  "MinimumLevel": {
    "Default": "Information",
    "Override": {
      "Microsoft": "Warning",
      "Microsoft.Hosting.Lifetime": "Information"
    }
  },
  "WriteTo": [
    {
      "Name": "File",
      "Args": {
        "path": "./logs/log-.txt",
        "rollingInterval": "Month"
      }
    },
    {
      "Name": "Console"
    }
  ]
},
"AllowedHosts": "*",
"Kestrel": {
  "Endpoints": {
    "gRPC": {
      "Url": "http://localhost:5060",
      "Protocols": "Http2"
    }
  }
},
```

Рисунок 4.1.1 – Настройка appseting.json для InfoService

4. В настроечных параметрах ConfigurationDbConfig, ProjectDbConfig, MessageLogDbConfig, HistoryEventDbConfig измените данные для подключения к базам данных PostgreSQL (Рисунок 4.1.2, для примера использован IP-адрес 172.30.205.26):
    - ConfigurationDbConfig – база конфигурации платформы;
    - ProjectDbConfig – база проекта;
    - MessageLogDbConfig – база журнала сообщений;

- HistoryEventDbConfig – база сохранения событий.

Поменяйте, при необходимости, имя пользователя и пароль для каждой из баз данных.

```
"ConfigurationDbConfig": {
  "Prefix": "PA_Config",
  "DataBase": "PA_Config",
  "Host": "172.30.205.26",
  "Port": "5432",
  "Username": "postgres",
  "Password": "postgres",
  "Pooling": "true"
},
"ProjectDbConfig": {
  "Prefix": "PA_Project",
  "DataBase": "PA_Project",
  "Host": "172.30.205.26",
  "Port": "5432",
  "Username": "postgres",
  "Password": "postgres",
  "Pooling": "true"
},
"MessageLogDbConfig": {
  "Prefix": "PA_Log",
  "DataBase": "PA_Log",
  "Host": "172.30.205.26",
  "Port": "5432",
  "Username": "postgres",
  "Password": "postgres",
  "Pooling": "true"
},
"HistoryEventDbConfig": {
  "Prefix": "PA_History",
  "DataBase": "PA_History",
  "Host": "172.30.205.26",
  "Port": "5432",
  "Username": "postgres",
  "Password": "postgres",
  "Pooling": "true"
},
```

Рисунок 4.1.2 – Настройка параметров базы данных в файле appsettings.json для сервиса InfoService

5. Сохраните файл и выйдете из текстового редактора.

4.1.1 Запуск сервиса InfoService

Для запуска InfoService необходимо наличие ранее установленной среды выполнения .NET (см. раздел 3.6). Запуск может осуществляться вручную в терминале Astra Linux, в виде daemon-сервиса, а также из ярлыка на рабочем столе с заранее подготовленным скриптом. Ниже рассмотрены все варианты.

Ручной запуск необходим, чтобы убедиться в правильной работе InfoService перед настройкой daemon-сервиса либо перед созданием ярлыка со скриптом. Для запуска в ручном режиме следует выполнить следующие действия:

1. Открыть терминал Astra Linux, перейти в директорию с InfoService, убедиться в возможности запуска сервиса от различных пользователей:

   ```
   ls -l | grep InfoService$
   ```

   В выводе терминала содержится информация о возможности запуска файла – последняя буква «x» в строке «-rwxr-xr-x» указывает на то, что файл может запускаться различными пользователями. Если вывод отличается, необходимо дать пользователям права на запуск файла:

   ```
   sudo chmod a+x InfoService
   ```

2. Выполнить команды создания переменных окружения, если после установки среды выполнения .NET 8.0 терминал был закрыт:

   ```
   export DOTNET_ROOT=~/.dotnet
   export PATH=$PATH:$DOTNET_ROOT:$DOTNET_ROOT/tools
   ```

3. Запустить InfoService:

   ```
   ./InfoService
   ```

4. Вывод терминала должен содержать подобную информацию (localhost будет заменен на IP-адрес, указанный Вами в файле appsettings.json):

Рисунок 4.1.1 – Настройка параметров для подключения к сервису авторизации

В случае необходимости остановки InfoService нажать Ctrl-C.

Создание daemon-сервиса для InfoService выполняется в следующем порядке:

1. Создать файл для запуска daemon-сервиса:

```
sudo nano /etc/systemd/system/infoservice.service
```

2. Внести в файл следующее содержание, при этом вместо «WisePredictiveAnalytics» указать директорию, в которой размещен ранее разархивированный каталог InfoService:

```
[Unit]
Description=InfoService
```

```
[Service]
Type=simple
Environment=DOTNET_ROOT=/opt/dotnet
Environment=PATH=$PATH:$DOTNET_ROOT:$DOTNET_ROOT/tools
ExecStart=sh -c "(cd /WisePredictiveAnalytics/InfoService &&
./InfoService)"
Restart=on-failure
RestartSec=10

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

3. Сохранить и закрыть файл InfoService (в программе nano комбинация клавиш Ctrl-O, Ctrl-X).
4. Добавить daemon-сервис в автозагрузку:
```
sudo systemctl enable infoservice
```
5. Запустить сервис:
```
sudo systemctl daemon-reload
sudo systemctl start infoservice
```
6. Проверить состояние сервиса:
```
sudo systemctl status infoservice
```
Вывод терминала должен содержать информацию о статусе сервиса (Loaded: loaded, Active: active (running), а также те же сообщения, как при ручном запуске).

Для запуска InfoService из ярлыка на рабочем столе необходимо выполнить следующие действия:

1. Если InfoService ранее был запущен как daemon-сервис, необходимо его остановить:
```
sudo systemctl stop infoservice
```
2. В терминале перейти в директорию, где будет расположен скрипт – рекомендуется каталог службы InfoService. Создать файл скрипта с одновременным открытием в текстовом редакторе:

```
nano InfoService.sh
```

3. Внести в файл следующее содержание, при этом вместо «WisePredictiveAnalytics» указать директорию, в которой размещен ранее разархивированный каталог InfoService:

```
#!/bin/bash
export DOTNET_ROOT=/opt/dotnet
export PATH=$PATH:$DOTNET_ROOT:$DOTNET_ROOT/tools
cd ~/WisePredictiveAnalytics/InfoService && ./InfoService
```

4. Сохранить и закрыть файл InfoService.sh (в программе nano комбинация клавиш Ctrl-O, Ctrl-X).
5. Выдать всем пользователям права на запуск файла скрипта InfoService.sh:

```
sudo chmod 777 InfoService.sh
```

6. В графическом интерфейсе Astra Linux открыть каталог, содержащий файла скрипта InfoService.sh, нажать на нем правой кнопкой мыши и в контекстном меню выбрать «Отправить» – «Рабочий стол (создать ярлык)», либо «Send» – «Desktop (create shortcut)» в английской версии Astra Linux. На рабочем столе двойным нажатием мыши по файлу ярлыка запустить InfoService – после нажатия откроется терминал с той же информацией, что при ручном запуске. Для остановки InfoService достаточно закрыть данный терминал.

4.2 Установка сервиса DataCollector

Для установки DataCollector необходимо скопировать архив DataCollector.tar.gz на машину с ОС Astra Linux в нужный каталог, затем распаковать архив следующей командой в терминале:

```
tar zxvf DataCollector.tar.gz
```

По умолчанию DataCollector имеет базовые настройки, предусматривающие локальное размещение всех сервисов платформы.

В случае, если платформа предусматривает распределенное развертывание, выполните следующие настройки:

1. Перейдите в каталог сервиса и откройте файл appsettings.json в текстовом редакторе (например, Kate).
2. При необходимости измените параметр “Url”, указав в нем IP-адрес сервера, на котором расположен сервис DataCollector (по умолчанию localhost). В этой же строке укажите порт (по умолчанию 5065). Остальные параметры рекомендуется оставить без изменений (Рисунок 4.2.1).

```
"Logging": {
  "LogLevel": {
    "Default": "Information",
    "Microsoft.AspNetCore": "Warning"
  }
},
"AllowedHosts": "*",
"Kestrel": {
  "Endpoints": {
    "gRPC": {
      "Url": "http://localhost:5065",
      "Protocols": "Http2"
    }
  }
},
"KeyCloakInfo": {
  "UseAuthorizationService": false,
  "ClientId" : "DataCollector",
  "ClientSecret": "",
  "CheckIncomingTokens": false
}
```

Рисунок 4.2.1 – Файл appsettings.json

3. Сохраните файл и выйдете из текстового редактора.

4.2.1 Запуск сервиса DataCollector

Для запуска DataCollector необходимо наличие ранее установленной среды выполнения .NET 8.0 (см. раздел 3.6). Запуск может осуществляться вручную в терминале Astra Linux, в виде daemon-сервиса, а также из ярлыка на рабочем столе с заранее подготовленным скриптом. Все варианты запуска DataCollector аналогичны вариантам запуска InfoService (см. раздел 4.1.1).

После запуска DataCollector в терминале Astra Linux отобразится подобное сообщение (localhost будет заменен на IP-адрес, указанный Вами в файле appsettings.json):

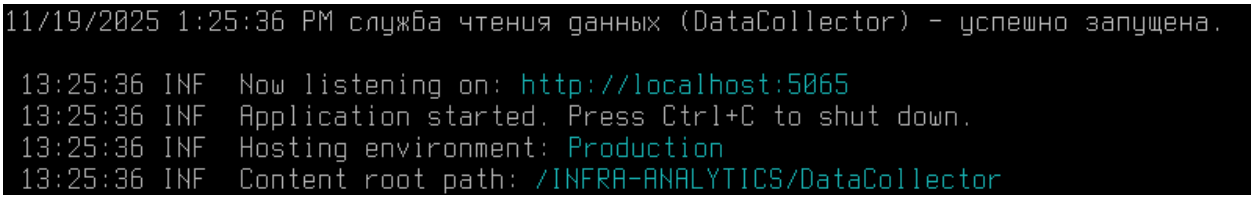

Рисунок 4.2.2 – Информация в терминале после запуска сервиса DataCollector

Настроечный файл daemon-сервиса для службы DataCollector должен иметь следующее содержание (вместо «WisePredictiveAnalytics» указать директорию, в которой размещен ранее разархивированный каталог DataCollector):

```
[Unit]
Description=DataCollector

[Service]
Type=simple
Environment=DOTNET_ROOT=/opt/dotnet
Environment=PATH=$PATH:$DOTNET_ROOT:$DOTNET_ROOT/tools
ExecStart=sh -c "(cd / WisePredictiveAnalytics /DataCollector &&
./DataCollector)"
Restart=on-failure
RestartSec=10

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

В остальном настройка daemon-сервиса для службы DataCollector выполняется так же, как для InfoService.

4.3 Установка сервиса DataProcessing

Для установки DataProcessing необходимо скопировать архив DataProcessing.tar.gz на машину с ОС Astra Linux в нужный каталог, затем распаковать архив следующей командой в терминале:

```
tar zxvf DataProcessing.tar.gz
```

По умолчанию DataProcessing имеет базовые настройки, предусматривающие локальное размещение всех сервисов платформы.

В случае, если платформа предусматривает распределенное развертывание, выполните следующие настройки:

1. Перейдите в каталог сервиса и откройте файл appsettings.json в текстовом редакторе (например, Kate).
2. При необходимости измените параметр “Url”, указав в нем IP-адрес сервера, на котором расположен настраиваемый сервис DataProcessing (по умолчанию localhost). В этой же строке укажите порт (по умолчанию 5064). Остальные параметры рекомендуется оставить без изменений (Рисунок 4.3.1).

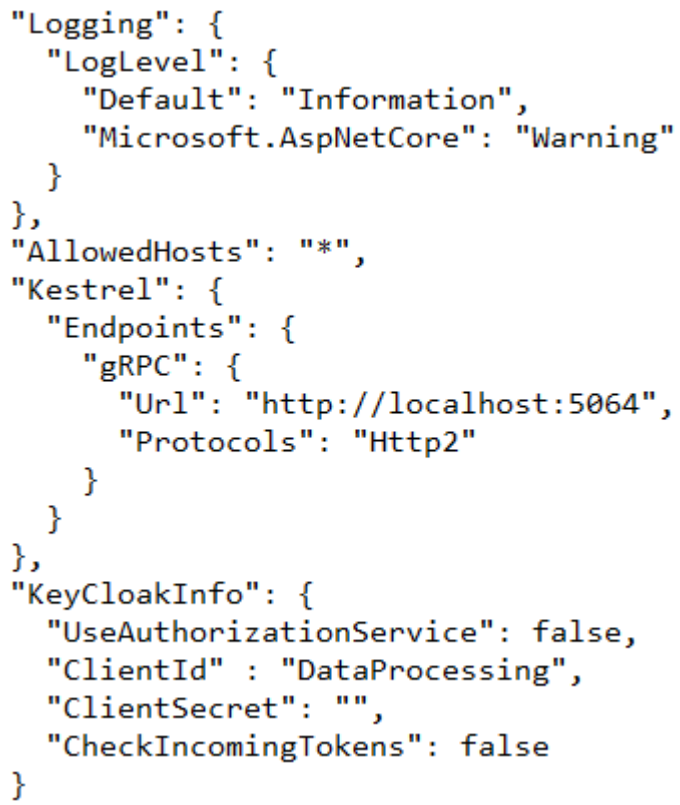

Рисунок 4.3.1 – Файл appsettings.json

3. Сохраните файл и выйдете из текстового редактора.

4.3.1 Запуск сервиса DataProcessing

Для запуска DataProcessing необходимо наличие ранее установленной среды выполнения .NET 8.0 (см. раздел 3.6). Запуск может осуществляться вручную в терминале Astra Linux, в виде daemon-сервиса, а также из ярлыка на рабочем столе с заранее подготовленным скриптом. Все варианты запуска DataProcessing аналогичны вариантам запуска InfoService (см. раздел 4.1.1).

После запуска DataProcessing в терминале Astra Linux отобразится подобное сообщение (localhost будет заменен на IP-адрес, указанный Вами в файле appsettings.json):

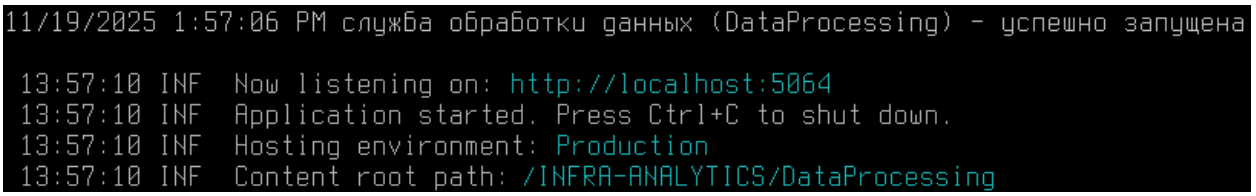

Рисунок 4.3.2 – Информация в терминале после запуска сервиса DataProcessing

Настроечный файл daemon-сервиса для службы DataProcessing должен иметь следующее содержание (вместо «WisePredictiveAnalytics» указать директорию, в которой размещен ранее разархивированный каталог DataProcessing):

```
[Unit]
Description=DataProcessing


[Service]
Type=simple
Environment=DOTNET_ROOT=/opt/dotnet
Environment=PATH=$PATH:$DOTNET_ROOT:$DOTNET_ROOT/tools
```

```
ExecStart=sh -c "(cd /WisePredictiveAnalytics /DataProcessing &&
./DataProcessing)"
Restart=on-failure
RestartSec=10

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

В остальном настройка daemon-сервиса для службы DataProcessing выполняется так же, как для InfoService.

4.4 Установка сервиса DataServer

Для установки DataServer необходимо скопировать архив DataServer.tar.gz на машину с ОС Astra Linux в нужный каталог, затем распаковать архив следующей командой в терминале:

```
tar zxvf DataServer.tar.gz
```

По умолчанию DataServer имеет базовые настройки, предусматривающие локальное размещение всех сервисов платформы.

В случае, если платформа предусматривает распределенное развертывание, выполните следующие настройки:

1. Перейдите в каталог сервиса и откройте файл appsettings.json в текстовом редакторе (например, Kate).
2. При необходимости измените параметр “Url”, указав в нем IP-адрес сервера, на котором расположен настраиваемый сервис DataServer (по умолчанию localhost). В этой же строке укажите порт (по умолчанию 5067). Остальные параметры рекомендуется оставить без изменений (Рисунок 4.4.1).

```
"Logging": {
  "LogLevel": {
    "Default": "Information",
    "Microsoft.AspNetCore": "Warning"
  }
},
"AllowedHosts": "*",
"Kestrel": {
  "Endpoints": {
    "Http":{
      "Url":"http://localhost:5100"
    },
    "gRPC": {
      "Url": "http://localhost:5067",
      "Protocols": "Http2"
    }
  }
},
"KeyCloakInfo": {
  "UseAuthorizationService": false,
  "ClientId" : "DataServer",
  "ClientSecret": "",
  "CheckIncomingTokens": false
}
```


Рисунок 4.4.1 – Файл appsettings.json

3. Сохраните файл и выйдете из текстового редактора.

4.4.1 Запуск сервиса DataServer

Для запуска DataServer необходимо наличие ранее установленной среды выполнения .NET 8.0 (см. раздел 3.6). Запуск может осуществляться вручную в терминале Astra Linux, в виде daemon-сервиса, а также из ярлыка на рабочем столе с заранее подготовленным скриптом. Все варианты запуска DataServer аналогичны вариантам запуска InfoService (см. раздел 4.1.1).

После запуска DataServer в терминале Astra Linux отобразится подобное сообщение (localhost будет заменен на IP-адрес, указанный Вами в файле appsettings.json):

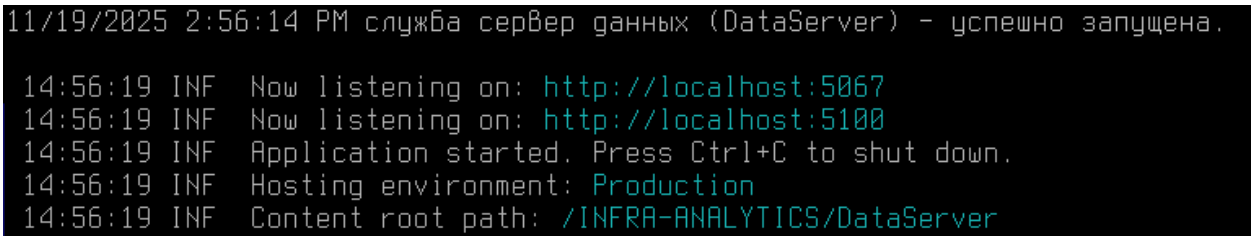

Рисунок 4.4.2 – Информация в терминале после запуска сервиса DataServer

Настроечный файл daemon-сервиса для службы DataServer должен иметь следующее содержание (вместо «WisePredictiveAnalytics» указать директорию, в которой размещен ранее разархивированный каталог DataServer):

```
[Unit]
Description=DataServer

[Service]
Type=simple
Environment=DOTNET_ROOT=/opt/dotnet
Environment=PATH=$PATH:$DOTNET_ROOT:$DOTNET_ROOT/tools
ExecStart=sh -c "(cd /WisePredictiveAnalytics/DataServer &&
./DataServer)"
Restart=on-success
RestartSec=10

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

В остальном настройка daemon-сервиса для службы DataServer выполняется так же, как для InfoService.

4.5 Установка InfraPortal

Для установки InfraPortal необходимо заранее установить Python версии не ниже 3.10 (см. раздел 3.8) и Node.js версии не ниже 20 (см. раздел 3.9).

Установка InfraPortal включает отдельную установку backend-части и frontend-части, которые рассмотрены в настоящем разделе.

4.5.1 Установка и запуск backend

Для установки backend-части InfraPortal необходимо выполнить следующие действия:

1. Распаковать архив backend.tar.gz из установочного пакета в нужный каталог – в графическом интерфейсе ОС Astra Linux нажать правой кнопкой мыши на файл, выбрать «Extract», затем «Extract archive here». Предлагается создать новый каталог InfraPortal в директории с другими компонентами платформы (InfoService, DataCollector, Configurator и т.д.) и разместить в ней каталоги с backend и frontend.
2. Открыть терминал, перейти в распакованный каталог backend, затем запустить команду установки необходимых пакетов Python (перечень пакетов приведен в файле requirements.txt, сами пакеты находятся в каталоге python-packages):

```
python -m pip install --no-index --find-links python-packages -r
requirements.txt
```

3. Проверить возможность запуска backend следующей командой:

```
python -m uvicorn --reload webServerApp.asgi:app --host localhost
--port 8000
```

Вывод терминала должен содержать подобную информацию (вместо WisePredictiveAnalytics/InfraPortal будет показан каталог, куда Вы распаковали архив backend.tar.gz):

```
INFO:     Will watch for changes in these directories:
['/WisePredictiveAnalytics/InfraPortal/backend']
INFO:     Uvicorn running on http://localhost:8000 (Press CTRL+C to
quit)
INFO:     Started reloader process [3884] using StatReload
INFO:     Started server process [3886]
```

```
INFO:     Waiting for application startup.
INFO:     ASGI 'lifespan' protocol appears unsupported.
INFO:     Application startup complete.
```

1. Создать каталог для логов backend и frontend, при этом вместо «WisePredictiveAnalytics/InfraPortal» указать директорию, в которой размещен ранее разархивированный каталог backend:

```
mkdir /WisePredictiveAnalytics/InfraPortal/logs
chmod -R 755 /WisePredictiveAnalytics/InfraPortal/logs
```

2. Создать файл для запуска daemon-сервиса:

```
sudo nano /etc/systemd/system/backend.service
```

3. Внести в файл следующее содержание, при этом вместо «WisePredictiveAnalytics/InfraPortal» указать директорию, в которой размещен ранее разархивированный каталог backend:

```
[Unit]
Description=Backend on Django
After=network.target

[Service]
Type=simple
User=root
Group=root
WorkingDirectory=/WisePredictiveAnalytics/InfraPortal/backend
ExecStartPre=/bin/sleep 30
ExecStart=python -m uvicorn --reload webServerApp.asgi:app --host
localhost --port 8000
StandardOutput=append:/
WisePredictiveAnalytics/InfraPortal/logs/back.log
StandardError=append:/
WisePredictiveAnalytics/InfraPortal/logs/back.log
Restart=on-failure
RestartSec=10

[Install]
```

```
WantedBy=multi-user.target
```

4. Сохранить и закрыть файл backend.service (в программе nano комбинация клавиш Ctrl-O, Ctrl-X).
5. Добавить daemon-сервис в автозагрузку:

```
sudo systemctl enable backend
```

6. Запустить сервис:

```
sudo systemctl daemon-reload
sudo systemctl start backend
```

7. Проверить состояние сервиса:

```
sudo systemctl status backend
```

Вывод терминала должен содержать информацию о статусе сервиса (Loaded: loaded, Active: active (running), а также те же сообщение «Started Backend on Django»).

8. Для ограничения роста файла логов необходимо настроить периодическую очистку логов с помощью службы logrotate. Для этого необходимо выполнить следующие действия (вместо «WisePredictiveAnalytics/InfraPortal» указывать директорию, в которой размещен ранее разархивированный каталог backend):

23.1 Создать с последующим открытием в текстовом редакторе файл для logrotate backend-части:

```
nano
/WisePredictiveAnalytics/InfraPortal/logs/logrotate-back.conf
```

23.2 Внести в файл следующее содержание:

```
/WisePredictiveAnalytics/InfraPortal/logs/back.log {
    hourly
    missingok
    rotate 24
    compress
    create
}
```

23.3 Запустить logrotate backend-части:

```
logrotate
/WisePredictiveAnalytics/InfraPortal/logs/logrotate-back.conf
--state
/WisePredictiveAnalytics/InfraPortal/logs/logrotate-state
–verbose
```

23.4 Проверить корректность работы logrotate:

```
cat /WisePredictiveAnalytics/InfraPortal/logs/logrotate-state
```

Вывод терминала должен содержать подобную информацию:

```
logrotate state -- version 2
"/WisePredictiveAnalytics/InfraPortal/logs/back.log"
2025-11-25-11:0:0
```

4.5.2 Установка и запуск frontend

Для установки frontend-части InfraPortal необходимо выполнить следующие действия:

1. Распаковать архив frontend.tar.gz из установочного пакета в нужный каталог – в графическом интерфейсе ОС Astra Linux нажать правой кнопкой мыши на файл, выбрать «Extract», затем «Extract archive here». Предлагается разместить frontend каталог в том же каталоге, где размещен backend.
2. Открыть терминал и перейти в распакованный каталог frontend. Запустить сборку проекта:

   ```
   npm run build
   ```

   Дождаться окончания сборки проекта. При успешном результате в терминале отобразится следующая информация:

   ```
   ✓ Compiled successfully in 54s
   ✓ Collecting page data
   ✓ Generating static pages (61/61)
   ✓ Collecting build traces
   ✓ Finalizing page optimization
   ```

3. Для запуска frontend-части InfraPortal на порту 80 требуется зайти в терминал с root-правами, перейти в распакованный каталог frontend (вместо «WisePredictiveAnalytics/InfraPortal» указать директорию, в которой размещен каталог frontend) и запустить frontend:

```
sudo -i
cd /WisePredictiveAnalytics/InfraPortal/frontend
npm start -- --port 80
```

Вывод терминала должен содержать подобную информацию (вместо WisePredictiveAnalytics/InfraPortal будет показан каталог, куда Вы распаковали архив frontend.tar.gz):

```
> frontend@0.1.0 start
> NEXT_PUBLIC_TEST=true next start --port 80

   ▲ Next.js 15.4.2
   - Local:        http://localhost:80
   - Network:      http://192.168.150.208:80

 ✓ Starting...
 ⚠ "next start" does not work with "output: standalone"
configuration. Use "node .next/standalone/server.js" instead.
 ✓ Ready in 466ms
(node:10902) Warning: `--localstorage-file` was provided without a
valid path
(Use `node --trace-warnings ...` to show where the warning was
created)
```

9. Для остановки frontend и возврата к вводу команд нажать сочетание клавиш Ctrl-C. Также можно вернуться к работе от текущего пользователя, нажав сочетание клавиш Ctrl-D.
10. Создать файл для запуска daemon-сервиса:

```
sudo nano /etc/systemd/system/frontend.service
```

11. Внести в файл следующее содержание, при этом вместо «WisePredictiveAnalytics/InfraPortal» указать директорию, в которой размещен ранее разархивированный каталог frontend:

```
[Unit]
Description=Frontend on Next.js
After=network.target

[Service]
EnvironmentFile=/WisePredictiveAnalytics/InfraPortal/frontend/.env.
production
Type=simple
User=root
Group=root
WorkingDirectory=/WisePredictiveAnalytics/InfraPortal/frontend
ExecStart=npm start -- --port 80
StandardOutput=append:/
WisePredictiveAnalytics/InfraPortal/logs/front.log
StandardError=append:/
WisePredictiveAnalytics/InfraPortal/logs/front.log
Restart=on-failure
RestartSec=10

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

12. Сохранить и закрыть файл frontend.service (в программе nano комбинация клавиш Ctrl-O, Ctrl-X).
13. Добавить daemon-сервис в автозагрузку:

```
sudo systemctl enable frontend
```

14. Запустить сервис:

```
sudo systemctl daemon-reload
sudo systemctl start frontend
```

15. Проверить состояние сервиса:

```
sudo systemctl status frontend
```

Вывод терминала должен содержать информацию о статусе сервиса (Loaded: loaded, Active: active (running), а также те же сообщение «Started Frontend on Next.js»).

16. Для ограничения роста файла логов необходимо настроить периодическую очистку логов с помощью службы logrotate. Для этого необходимо выполнить следующие действия (вместо «WisePredictiveAnalytics/InfraPortal» указывать директорию, в которой размещен ранее разархивированный каталог frontend):

    23.1 Создать с последующим открытием в текстовом редакторе файл для logrotate frontend-части:

    ```
    nano
    /WisePredictiveAnalytics/InfraPortal/logs/logrotate-front.con
    f
    ```

    23.2 Внести в файл следующее содержание:

    ```
    /WisePredictiveAnalytics/InfraPortal/logs/front.log {
        hourly
        missingok
        rotate 24
        compress
        create
    }
    ```

    23.3 Запустить logrotate frontend-части:

    ```
    logrotate
    /WisePredictiveAnalytics/InfraPortal/logs/logrotate-front.con
    f --state
    /WisePredictiveAnalytics/InfraPortal/logs/logrotate-state
    –verbose
    ```

    23.4 Проверить корректность работы logrotate:

    ```
    cat /WisePredictiveAnalytics/InfraPortal/logs/logrotate-state
    ```

    Вывод терминала должен содержать подобную информацию:

    ```
    logrotate state -- version 2
    ```

```
"/WisePredictiveAnalytics/InfraPortal/logs/front.log"
2025-11-25-11:0:0
"/WisePredictiveAnalytics/InfraPortal/logs/back.log"
2025-11-25-11:0:0
```

5. Требования к персоналу

Для администрирования программного обеспечения персонал должен обладать следующими знаниями и навыками:

навыками системного администрирования операционной системы Astra Linux версий 1.7, 1.8;

навыками системного администрирования баз данных;

знаниями и навыками администрирования АСУ ТП (при использовании промышленных протоколов).

6. Характеристика программы

6.1 Описание основных характеристик программы

Программное обеспечение позволяет:

- Осуществлять сбор и долговременное хранение данных технологических процессов;
- Осуществлять анализ данных и событий технологического оборудования и режимов его работы;
- Осуществлять оптимизацию параметров и режимов технологических процессов;
- Осуществлять визуализацию технологических процессов.

6.2 Режим работы программы

Работа сервисов (служб) в операционной системе Astra Linux осуществляется в форме фоновых процессов или сервисов.

Работа программ *Configurator* и *WisePredictiveAnalytics Workstation* осуществляется в многооконном режиме работы в операционной системе Astra Linux с графическим интерфейсом.

Работа веб-приложения *InfraPortal* при развертывании в Astra Linux может осуществляться как в dev-режиме, так и в prod-режиме в форме фоновых процессов или сервисов.

6.3 Средства контроля правильности выполнения программы

Контроль правильности выполнения программного обеспечения выполняется встроенными средствами самого программного обеспечения.

6.4 Самовосстанавливаемость программы

Саммовосстанавлиемость программного обеспечения выполняется встроенными средствами самого программного обеспечения.

7. Обращение к программе Configurator

7.1 Загрузка и запуск программы Configurator

Запуск программы конфигурирования платформы Configurator имеет следующие особенности:

1. Перед запуском программы следует запустить следующие службы в указанной последовательности:
   - InfoService
   - DataCollector (при использовании коллекторов)
   - DataProcessing (при использовании обработки данных)
   - DataServer (при использовании витрины данных)
   - Portal_backend (при использовании веб-приложения)
   - Portal_frontend (при использовании веб-приложения)

   Если запуск служб был сконфигурирован в виде daemon-сервисов, то при загрузке компьютера они запускаются автоматически. Статус каждой службы можно проверить в терминале с помощью команд:

```
sudo systemctl status infoservice
sudo systemctl status datacollector
sudo systemctl status dataprocessing
sudo systemctl status dataserver
sudo systemctl status backend
sudo systemctl status frontend
```

   Для ручного запуска daemon-сервисов следует выполнить в терминале следующие команды:

```
sudo systemctl start infoservice
sudo systemctl start datacollector
sudo systemctl start dataprocessing
sudo systemctl start dataserver
```

```
sudo systemctl start backend
sudo systemctl start frontend
```

Также возможен запуск служб из ярлыков рабочего стола с заранее сконфигурированными скриптами, что подробно описано в разделе 4.1.1 настоящего документа, при этом каждая служба будет открыта в отдельном терминале.

2. Запуск программы *Configurator* двойным кликом мыши по исполняемому файлу *Configurator*, который расположен в каталоге *Configurator*.
3. При запуске программы должно открыться окно входа (Рисунок 7.1.1).
4. Вход осуществляется с учетными данными пользователя, созданного при установке. При использовании режима разработчика используются следующие учетные данные:
   - логин: admin
   - пароль: admin

5. В случае успешной загрузки программы должно открыться основное окно программы (Рисунок 7.1.2).
6. Дальнейшая работа с программой описана в документе «Руководство пользователя WisePredictiveAnalytics Configurator».

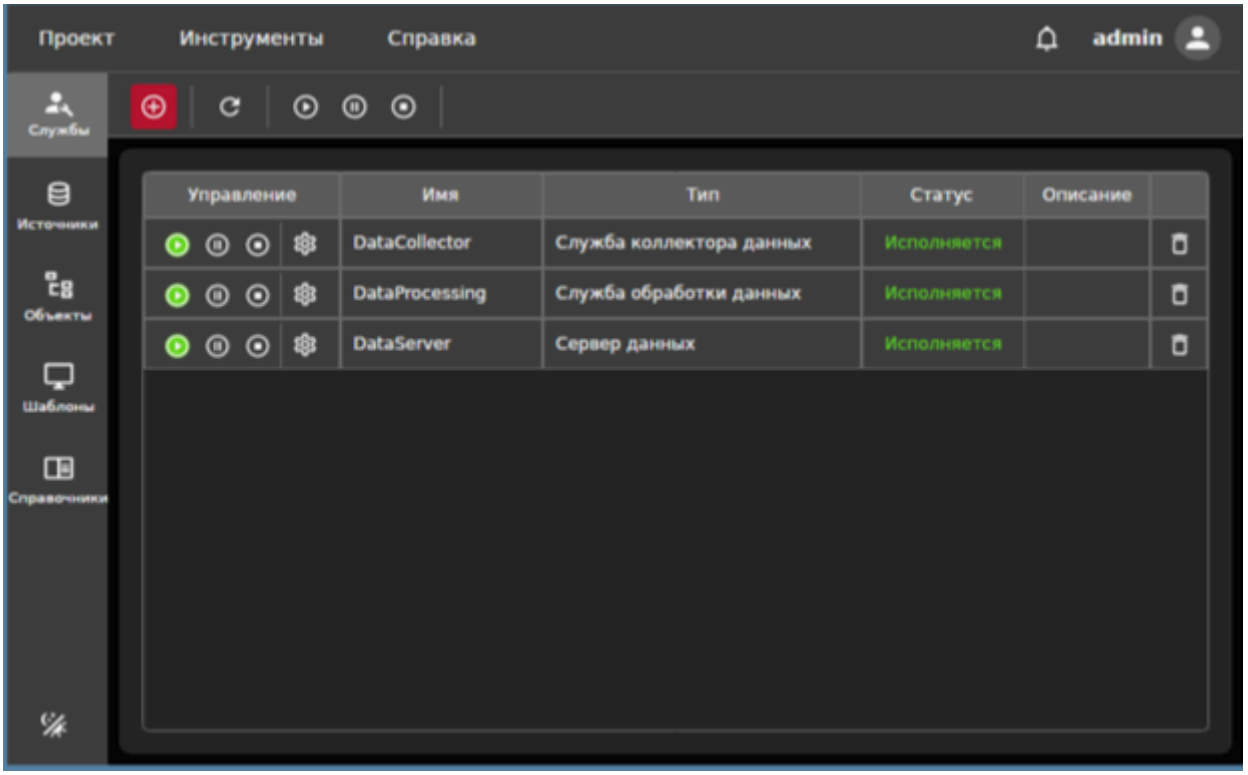

Рисунок 7.1.2 – Основное окно программы Configurator

8. Входные и выходные данные

8.1 Организация используемой входной информации

Входная информация может быть представлена в виде:

- Получение данных от АСУТП с использованием промышленных протоколов;
- Получение данных из файлов «CSV»;
- Получение данных от внешних информационных систем по API.

8.2 Организация используемой выходной информации

Выходная информация может быть реализована в виде:

- Запись данных в источники с использованием, специализированных протоколов источников данных;
- Публикация данных с использованием промышленных протоколов и API;
- Мониторинг системных сообщений и метрик с использованием Syslog и Prometheus;
- Визуализация данных в настольном и веб-приложении.