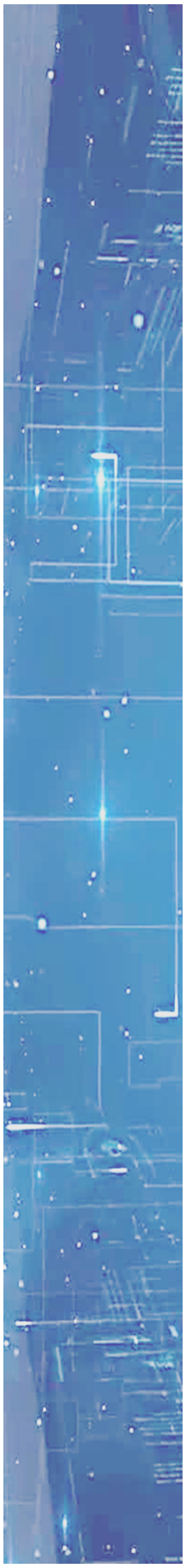
Навигатор BI

Версия 81

05.10.2026



Руководство по установке ИИ-помощника (версия 03.140.01)

Москва
2026

Навигатор BI: Руководство по установке ИИ-помощника (версия 03.140.01)

М.: ПАО «Сбербанк», 2026

Программное обеспечение и настоящий документ не могут быть скопированы, размножены, использованы по частям для составления других текстов, переведены на другие языки, если это не оговорено в письменной форме в договоре на поставку программного обеспечения.

Программное обеспечение, описанное в настоящем Руководстве, поставляется по лицензионному соглашению и может использоваться или копироваться только в соответствии с условиями этого соглашения.

Разработчиком и генеральным распространителем программного продукта «Навигатор BI» является ПАО «Сбербанк».

Адрес: Россия, Москва, 117997, ул. Вавилова, д. 19

navplatform1@sberbank.ru

https://sberanalytics.ru/products/ksb/navigator

© ПАО «Сбербанк» 2026

Содержание

1 Сравнение функционала с LLM и без 4

2 Настройка версии с LLM 5

2.1 Установка OpenSearch 5

2.2 Установка Postgres 6

2.3 Python приложения: 7

2.4 Java приложения 9

2.4.1 Настройка 9

2.4.2 Запуск 10

2.5 Настройка через APM Навигатор 12

2.6 Настройка Portal Server 12

3 Настройка версии без LLM 14

3.1 Установка OpenSearch: 14

3.2 Python приложения: 14

3.3 Java приложения 15

3.3.1 Настройка 15

3.3.2 Запуск 16

3.4 Настройка в APM Навигатор 18

3.5 Настройка автоматической разметки дешбордов 19

3.6 Настройка Portal Server 19

4 FAQ по целевой архитектуре 21

4.1 Как проверить работоспособность приложений ? 21

4.2 Как настраивается sslType? 21

1 Сравнение функционала с LLM и без

ИИ-агент/Навык	Описание	Версия без LLM	Версия с LLM*
GenBI	Генерация визуала в виде виджетов из сырых данных по пользовательскому запросу, которые можно добавить себе на обзорный экран	-	+
Q&A по данным	Анализа над данными с выявлением трендов, зависимостей, отклонений и т.д. Ответ представляется в виде текстовой аналитики с применением форматирования текста	-	+
Поиск по дэшбордам	Находит размеченные дэшборды в системе и предлагает их для быстрого перехода	-	+
Поиск по документам	Помогает найти нужные документы и ответить на вопросы, анализируя их содержимое	+	+
Поиск по показателям	Поиск виджетов среди размеченных дэшбордов с опцией добавления на обзорный экран	-	+
Супервизор	Интеллектуальный агент, который управляет, координирует или контролирует поведение других агентов в мультиагентной системе.	-	+
Системные команды	Позволяют управлять интерфейсом Навигатора	-	+
Доуточнения	Фильтрация данных на дэшборде по запросу	-	+
Описание экранов	Позволяет описывать экраны с помощью LLM на основе их метаданных для вывода в визуале и улучшения ИИ-поиска	-	+

*Рекомендуемая LLM - Gigachat 2 Max

Минимальные системные требования для сервера - 8 CPU, 32 RAM (поиск по дэшбордам может работать на 4 CPU и 16 RAM)

Для ускорения обучения моделей поиска предлагается GPU с минимальной конфигурацией - 24gb ОП (4090)

Для доступа к функциям речевых технологий (распознавание и синтез речи) используется отдельное приложение - SaluteSpeech

Версия Python не меньше 3.12

2 Настройка версии с LLM

2.1 Установка OpenSearch

Docker:

1. Настроить config.toml в репозитории, обязательно поставив свои значения (**связка "admin" и "password" указана в качестве примера и не сработает, так как есть проверка на распространенные логины и пароли**) логина-пароля и хоста (если запускается на разных машинах).

```
[connections]
[connections.POSTGRES-DB]
dialect="postgresql+asyncpg"
host="postgres"
port=5432
database="database"
username="admin"
password="password"

[connections.PVS-DB]
username="admin"
password="password"
host="opensearch"
port=9200
index="box-navigator"
```

2. В .env необходимо удостоверится, что значения OPENSEARCH_* должны совпадать со значениями с [connections.PVS-DB] в config.toml

3. Запустить контейнеры миграции и дождаться их выполнения.

```
docker compose run --rm supervisor-migrations
docker compose run --rm genbi-migrations
docker compose run --rm search-tool-migrations
```

On premise:

1. Ссылка на документацию по установке https://docs.opensearch.org/latest/install-and-configure/install-opensearch/index/
Упрощенно для Debian-подобных систем:

```
sudo apt install opensearch
```

Или для RHEL/CentOS

```
sudo yum install opensearch
# или для dnf:
sudo dnf install opensearch
```

По умолчанию OpenSearch запускается по порту 9200.

Можно потестировать работоспособность по ссылке:

```
curl -X GET "localhost:9200/"
```

2. Настроить configt.toml в репозитории, поставив свои значения (**связка "admin" и "password" указана в качестве примера и не сработает, так как есть проверка на распространенные логины и пароли**) логина-пароля и хоста [connections.PVS-DB] (если запускается на разных машинах).

```
[connections]
[connections.POSTGRES-DB]
dialect="postgresql+asyncpg"
host="127.0.0.1"
port=5432
database="database"
username="admin"
password="password"

[connections.PVS-DB]
username="admin"
password="password"
host="127.0.0.1"
port=9200
index="box-navigator"
```

3. Создать необходимые индексы для работы поиска экранов, показателей и GenBI. Для этого необходимо выполнить:

```
cd search-tool && DB_SECRETS=../config.toml ./venv/bin/python -m
migrations.opensearch.create_indices
cd genbi && DB_SECRETS=../config.toml ./venv/bin/python -m
migrations.opensearch.create_indices
```

2.2 Установка Postgres

1. Ссылка на документацию по установке https://www.postgresql.org/download/
2. Настроить config.toml в репозитории, поставив свои значения (**связка "admin" и "password" указана в качестве примера и не сработает, так как есть проверка на распространенные логины и пароли**) логина-пароля и хоста [connections.POSTGRES-DB](если запускается на разных машинах).

```
[connections]
[connections.POSTGRES-DB]
dialect="postgresql+asyncpg"
host="127.0.0.1"
port=5432
database="database"
username="admin"
password="password"

[connections.PVS-DB]
username="admin"
password="password"
```

```
host="127.0.0.1"
port=9200
index="box-navigator"
```

3. В .env необходимо удостоверится, что значения POSTGRES_* должны совпадать со значениями с [connections.POSTGRES-DB] в config.toml
4. Выполнить миграцию для postgres:

```
cd supervisor && ./venv/bin/pip install alembic
DB_SECRETS=../config.toml ./venv/bin/alembic -c migrations/alembic.ini upgrade
head
```

2.3 Python приложения:

Docker:

1. Python приложения поднимаются как docker контейнеры. См. env-файл для изменении портов по умолчанию для python - сервисов.

Поднимаются в составе команды:

```
docker compose up -d
```

2. Настроить переменные окружения. См. пункт 8 в разделе "On premise"

On premise:

1. Создать пользователя **voice_app:**

```
useradd voice_app:voice_app
```

2. Распаковать архив с дистрибутивом в `/home/voice_app/`
3. Изменить права на папок с моделями:

```
sudo chown -R voice_app:voice_app /home/voice_app
        sudo chmod -R a+x /home/voice_app/
```

4. Проверить наличие Python 3.12, при необходимости – установить;
5. Последний дистрибутив приходит с поставленными Python environments, то есть в каждом из сервисов есть папка venv с уже установленными пакетами Python. В таком случае, нужно запустить

```
sudo ./install.sh
```

и создадутся службы сервисов. Пункты 6-8 данного руководства для случая, когда этих environments - еще нет

6. Под пользователем voice_app установить **python3 pip**;
7. Под пользователем voice_app установить **PyTorch**:

```
pip install torch
pip install pydantic-settings
```

8. Под пользователем voice_app запустить установку зависимостей в каждой папке, где model_1 - название папки/сервиса:

```
su voice_app
        source venv/bin/activate
        cd /home/voice_app/model_1
        pip install -r requirements.txt
        source deactivate
        source model_2_env/bin/activate
        cd /home/voice_app/model_2
        pip install -r requirements.txt
        ...
```

9. В файле .env (убрать .example) необходимо прописать какую LLM планируется использовать - GigaChat или OpenAI-совместимую LLM

```
LLM_PROVIDER = "openai"
OPENAI_API_KEY = "токен к API"
OPENAI_BASE_URL = "ссылка до API"
OPENAI_MODEL = "название модели"
```

В случае же использования GigaChat есть возможность ходить по токену, для этого нужно кроме первых двух переменных окружения указать еще и GIGACHAT_CREDENTIALS:

Host для LLM

```
LLM_PROVIDER = "gigachat"
GIGACHAT-HOST = "https://gigachat:5050"
GIGACHAT_CREDENTIALS = "токен к API"
GIGACHAT_SCOPE = "GIGACHAT_API_CORP"
```

GIGACHAT_SCOPE зависит от типа ключа (PERS/B2B/CORP):
GIGACHAT_API_PERS — доступ для физических лиц.
GIGACHAT_API_B2B — доступ для ИП и юридических лиц по платным пакетам.
GIGACHAT_API_CORP — доступ для ИП и юридических лиц по схеме pay-as-you-go.

Дополнительно можно задать статичные заголовки, которые будут добавляться к каждому запросу в LLM.

```
LLM_STATIC_HEADERS="x-llm-token=123,x-other-header=abc"
```

Формат - пары key=value через запятую.

10. Отредактировать файлы `/home/voice_app/.env`, изменив параметры **–host** и **–port** на необходимые. (для моделей: genbi, supervisor, multiagent, search-tool)

11. Запустить все модели по команде из пункта 5, либо самостоятельно создать службы для запуска скриптов (из-под пользователя voice_app):

```
su voice_app
        source *PATH_TO_MODEL1_env*/bin/activate
        python *PATH_TO_MODEL1*/app.py
        source deactivate
        ...
```

При отсутствии доступа к интернету:

1. Удостовериться что директория hf_cache находится внутри nvg-vectorizer.
2. Иногда, в ином случае, директория hf_cache может находится в корне проекта. Тогда выполнить:

```
mv /home/voice_app/hf_cache /home/voice_app/nvg-vectorizer/hf_cache
```

3. При запуске nvg-vectorizer:

```
cd nvg-vectorizer && source ./OFFLINE_ENV.sh
```

2.4 Java приложения

2.4.1 Настройка

AI Worker Manager

Создайте папку с конфигурацией для **AI Worker Manager**, после чего скопируйте туда **всю** конфигурацию из дистрибутива (конфиги для воркера находятся по пути Docs/config).

1. Настройте application.yml согласно примеру (необходимо переименовать файл 'application__for-Ai-Worker-Manager.yml' в 'application.yml')

2. Настройте worker-config.yml

3. Настройте core-props.yml

4. Настройте (опционально) blacklist.yml.

   Если в ходе развертывания вы столкнулись с тем, что приложение пытается обработать или изменить защищенные свойства, то можно внести их в черный список.

   Важно! По умолчанию некоторые свойства уже заданы в черном списке (см. в примере), однако если вы создадите свой файл `blacklist.yml`, то в чёрный список попадут **только** те свойства, которые указаны в этом файле. Старые (по умолчанию) свойства учитываться не будут. Чтобы добавить файл нужно указать его при запуске в **spring.config.import**

AI Master

Создайте папку с конфигурацией для **AI Master**, после чего скопируйте туда **всю** конфигурацию из дистрибутива (конфиги для мастера находятся по пути Docs/config).

1. Настройте application.yml согласно примеру (необходимо переименовать файл 'application__for-Ai-Master.yml' в 'application.yml')

2. Настройте master-config.yml

3. Настройте task-config.yml

   Обратите внимание, что пример файла в будущих релизах может поменяться.

4. Настройте core-props.yml

5. Настройте (опционально) blacklist.yml

Если в ходе развертывания вы столкнулись с тем, что приложение пытается обработать или изменить защищенные свойства, то можно внести их в черный список.

Важно, по умолчанию некоторые свойства уже заданы в черном списке (см. в примере), одна если вы создадите свой файл `blacklist.yml`, то в чёрный список попадут **только** те свойства, которые указаны в этом файле. Старые (по умолчанию) свойства учитываться не будут. Чтобы добавить файл нужно указать его при запуске в **spring.config.import**

2.4.2 Запуск

Обратите внимание, что в этой инструкции рассмотрен **базовый** вариант инфраструктуры с одним мастером и воркером – это **минимальная единица** развертывания. Однако, приложение умеет работать в режиме мульти-инстанса с масштабируемым числом экземпляров приложения, распределяя нагрузку между друг другом.

AI Master

Поля в {{ }} необходимо заполнить самостоятельно в зависимости от вашего окружения и потребностей.

Пример systemd-сервиса:

```
[Unit]
Description=Navigator master service
After=network.target

[Service]
User={{ service_user }}
WorkingDirectory={{ master_app_dir }}
ExecStart={{ java_home }}/bin/java {{ java_opts_common }}
-DKEY_FOLDER={{ ssl_folder }} -Dnavigator_apps_log_dir={{ logs_folder }}
-DINSTANCE_ID={{ app_name }} -Dserver.port={{ port_master }}
-Dencryptor.key=navigator -jar {{ master_app_dir }}/current.jar --
spring.config.import=file:{{ master_app_dir }}/config/config-sources.yml,file:
{{ master_app_dir }}/config/core-props.yml
SuccessExitStatus=143
LimitNOFILE=65536

Restart=always
RestartSec=10s
StartLimitInterval=300s
StartLimitBurst=5

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

Обратите внимание, что при запуске приложения обязательно должны быть указаны -Dencryptor.key, --spring.config.import с config-sources.yml и core-props.yml

Имя поля	Обозначение
{{ service_user }}	под каким пользователем запустить приложение
{{ java_home }}	путь расположения JVM

Имя поля	Обозначение
{{ java_opts_common }}	опционально, java параметры для запуска приложения, например `-Xms512m -Xmx1024m`
{{ port_master }}	на каком порту должен запуститься мастер
{{ master_app_dir }}	директория с исполняемым приложением мастера и конфигурацией
{{ app_name }}	уникальное имя сервиса
{{ logs_folder }}	путь для папки с логами
{{ ssl_folder }}	путь до папки с сертификатами

AI Worker Manager

Поля в {{ }} необходимо заполнить самостоятельно в зависимости от вашего окружения и потребностей.

Пример systemd-сервиса:

```
[Unit]
Description=Navigator worker-manager service
After=network.target

[Service]
User={{ service_user }}
WorkingDirectory={{ worker_app_dir }}
ExecStart={{ java_home }}/bin/java {{ java_opts_common }}
-DKEY_FOLDER={{ ssl_folder }} -Dnavigator_apps_log_dir={{ logs_folder }}
-DINSTANCE_ID={{ app_name }} -Dserver.port={{ port_worker }}
-Dencryptor.key=navigator -jar {{ worker_app_dir }}/current.jar --
spring.config.import=file:{{ worker_app_dir }}/config/application.yml
SuccessExitStatus=143
LimitNOFILE=65536

Restart=always
RestartSec=10s
StartLimitInterval=300s
StartLimitBurst=5

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

Обратите внимание, что при запуске приложения обязательно должны быть указаны -Dencryptor.key, --spring.config.import с application.yml

Имя поля	Обозначение
{{ service_user }}	под каким пользователем запустить приложение

Имя поля	Обозначение
{{ java_home }}	путь расположения JVM
{{ java_opts_common }}	опционально, java параметры для запуска приложения, например `-Xms512m -Xmx1024m`
{{ port_worker }}	на каком порту должен запуститься воркер
{{ worker_app_dir }}	директория с исполняемым приложением мастера и конфигурацией
{{ app_name }}	уникальное имя сервиса
{{ logs_folder }}	путь для папки с логами
{{ ssl_folder }}	путь до папки с сертификатами

2.5 Настройка через АРМ Навигатор

1. В интерфейсе Навигатора перейти в **АРМ:Администрирование – Система – Настройки – Опции реализации**
2. Установить значение параметра `enable-voice-assistant` в **true**, если такой параметр отсутствует – создать его.
3. Перейти на обзорный экран, обновить страницу и проверить результат установки (в правом нижнем углу на обзорном экране должна появится кнопка ИИ-помощника).

2.6 Настройка Portal Server

1. В файле configSources.yml (или navigator-master-config) прописать путь до конфигурационного файла, добавив новый блок для ИИ (AiModuleConfig.yml). Важно чтобы файлы находились в одной папке

блок ИИ

```
ai-module:
  nameSpace: portal-server
  type: yaml
  priority: 5
  path: AiModuleConfig.yml
  poll-period: 60s
```

2. Скопировать конфигурационный файл `AiModuleConfig.yml` на сервер приложений Навигатора в папку, где располагается файл configSources.yml `%путь_к_WildFly%/standalone/configuration/app_props/`, прописать необходимые параметры:

- В параметрах `previewModelUrl` и `targetModelUrl` в случае использования одной среды можно закомментировать один из блоков (например preview) , в случае двух и более сред – Ip-адреса серверов для Preview и Target. Эти адреса можно посмотреть в `%путь_к_WildFly%/standalone/configuration/standalone.xml` в секции `<system-properties>`

3. В случае использования **https** – необходимо раскомментировать в каждой секции конфигурационного файла строки `certProps` с параметрами сертификатов
4. Перезапустить WildFly

3 Настройка версии без LLM

3.1 Установка OpenSearch:

1. Ссылка на документацию по установке https://docs.opensearch.org/latest/install-and-configure/install-opensearch/index/
Упрощенно для Debian-подобных систем:

```
sudo apt install opensearch
```

Или для RHEL/CentOS

```
sudo yum install opensearch
# или для dnf:
sudo dnf install opensearch
```

По умолчанию OpenSearch запускается по порту 9200.

Можно потестировать работоспособность по ссылке:

```
curl -X GET "localhost:9200/"
```

2. Настроить config.toml в репозитории, поставив свои значения (**связка "admin" и "password" указана в качестве примера и не сработает, так как есть проверка на распространенные логины и пароли**) логина-пароля и хоста (если запускается на разных машинах).

```
[connections]
[connections.PVS-DB]
username="admin"
password="password"
host="127.0.0.1"
port=9200
index="box-navigator"
```

3. Создать необходимые индексы для работы поиска экранов, показателей.

```
cd search-tool && DB_SECRETS=../config.toml ./venv/bin/python -m
migrations.opensearch.create_indices
```

3.2 Python приложения:

1. Создать пользователя **voice_app:**

   ```
   useradd voice_app:voice_app
   ```

2. Распаковать архив с дистрибутивом в `/home/voice_app/`

3. Изменить права на папок с моделями `nvg-tool-search, nvg-query-clf, nvg-vectorizer`:

```
sudo chown -R voice_app:voice_app /home/voice_app
        sudo chmod -R a+x /home/voice_app/
```

4. Проверить наличие Python 3, при необходимости – установить;
5. Под пользователем voice_app установить **python3 pip**;
6. Под пользователем voice_app установить **PyTorch**:

```
pip install torch
```

7. Под пользователем voice_app запустить установку зависимостей в каждой папке:

```
su voice_app
        source model_1_env/bin/activate
        cd /home/voice_app/nvg-tool-search
        pip install -r requirements.txt
        source deactivate
        source model_2_env/bin/activate
        cd /home/voice_app/nvg-query-clf
        pip install -r requirements.txt
        ...
```

8. В файле app_fastAPI.py заменить в строке uvicorn.run(app, host="127.0.0.1", port=5000) host и port на необходимый
9. Следует учитывать, что порты для всех моделей должны быть разные
10. Запустить все модели, либо создать службы для запуска скриптов (из-под пользователя voice_app).

```
su voice_app
        source *PATH_TO_MODEL1_env*/bin/activate
        python *PATH_TO_MODEL1*/app.py
        source deactivate
        ...
```

3.3 Java приложения

3.3.1 Настройка

AI Worker Manager

Создайте папку с конфигурацией для **AI Worker Manager**, после чего скопируйте туда **всю** конфигурацию из дистрибутива (конфиги для воркера находятся по пути Docs/config).

1. Настройте application.yml согласно примеру (необходимо переименовать файл 'application__for-Ai-Worker-Manager.yml' в 'application.yml')
2. Настройте worker-config.yml
3. Настройте core-props.yml
4. Настройте (опционально) blacklist.yml.

Если в ходе развертывания вы столкнулись с тем, что приложение пытается обработать или изменить защищенные свойства, то можно внести их в черный список.

Важно! По умолчанию некоторые свойства уже заданы в черном списке (см. в примере), однако если вы создадите свой файл `blacklist.yml`, то в чёрный список попадут **только** те свойства, которые указаны в этом файле. Старые (по умолчанию) свойства учитываться не будут. Чтобы добавить файл нужно указать его при запуске в **spring.config.import**

AI Master

Создайте папку с конфигурацией для **AI Master**, после чего скопируйте туда **всю** конфигурацию из дистрибутива (конфиги для мастера находятся по пути Docs/config).

1. Настройте application.yml согласно примеру (необходимо переименовать файл 'application__for-Ai-Master.yml' в 'application.yml')
2. Настройте master-config.yml
3. Настройте task-config.yml

Обратите внимание, что пример файла в будущих релизах может поменяться.

4. Настройте core-props.yml (значение для USE_AGENT_SUPERVISOR должно быть false)
5. Настройте (опционально) blacklist.yml

Если в ходе развертывания вы столкнулись с тем, что приложение пытается обработать или изменить защищенные свойства, то можно внести их в черный список.

> **Важно,** по умолчанию некоторые свойства уже заданы в черном списке (см. в примере), одна если вы создадите свой файл `blacklist.yml`, то в чёрный список попадут **только** те свойства, которые указаны в этом файле. Старые (по умолчанию) свойства учитываться не будут. Чтобы добавить файл нужно указать его при запуске в **spring.config.import**

3.3.2 Запуск

Обратите внимание, что в этой инструкции рассмотрен **базовый** вариант инфраструктуры с одним мастером и воркером – это **минимальная единица** развертывания. Однако, приложение умеет работать в режиме мульти-инстанса с масштабируемым числом экземпляров приложения, распределяя нагрузку между друг другом.

AI Master

Поля в {{ }} необходимо заполнить самостоятельно в зависимости от вашего окружения и потребностей.

Пример сервиса:

```
[Unit]
Description=Navigator master service
After=network.target

[Service]
User={{ service_user }}
```

```
WorkingDirectory={{ master_app_dir }}
ExecStart={{ java_home }}/bin/java {{ java_opts_common }}
-DKEY_FOLDER={{ ssl_folder }} -Dnavigator_apps_log_dir={{ logs_folder }}
-DINSTANCE_ID={{ app_name }} -Dserver.port={{ port_master }}
-Dencryptor.key=navigator -jar {{ master_app_dir }}/current.jar --
spring.config.import=file:{{ master_app_dir }}/config/config-sources.yml,file:
{{ master_app_dir }}/config/core-props.yml
SuccessExitStatus=143
LimitNOFILE=65536

Restart=always
RestartSec=10s
StartLimitInterval=300s
StartLimitBurst=5

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

Обратите внимание, что при запуске приложения обязательно должны быть указаны -Dencryptor.key, --spring.config.import с config-sources.yml и core-props.yml

Имя поля	Обозначение
{{ service_user }}	под каким пользователем запустить приложение
{{ java_home }}	путь расположения JVM
{{ java_opts_common }}	опционально, java параметры для запуска приложения, например `-Xms512m -Xmx1024m`
{{ port_master }}	на каком порту должен запуститься мастер
{{ master_app_dir }}	директория с исполняемым приложением мастера и конфигурацией
{{ app_name }}	уникальное имя сервиса
{{ logs_folder }}	путь для папки с логами
{{ ssl_folder }}	путь до папки с сертификатами

AI Worker Manager

Поля в {{ }} необходимо заполнить самостоятельно в зависимости от вашего окружения и потребностей.

Пример сервиса:

```
[Unit]
Description=Navigator worker-manager service
After=network.target

[Service]
```

```
User={{ service_user }}
WorkingDirectory={{ worker_app_dir }}
ExecStart={{ java_home }}/bin/java {{ java_opts_common }}
-DKEY_FOLDER={{ ssl_folder }} -Dnavigator_apps_log_dir={{ logs_folder }}
-DINSTANCE_ID={{ app_name }} -Dserver.port={{ port_worker }}
-Dencryptor.key=navigator -jar {{ worker_app_dir }}/current.jar --
spring.config.import=file:{{ worker_app_dir }}/config/application.yml
SuccessExitStatus=143
LimitNOFILE=65536

Restart=always
RestartSec=10s
StartLimitInterval=300s
StartLimitBurst=5

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

Обратите внимание, что при запуске приложения обязательно должны быть указаны -Dencryptor.key, --spring.config.import с application.yml

Имя поля	Обозначение
{{ service_user }}	под каким пользователем запустить приложение
{{ java_home }}	путь расположения JVM
{{ java_opts_common }}	опционально, java параметры для запуска приложения, например `-Xms512m -Xmx1024m`
{{ port_worker }}	на каком порту должен запуститься воркер
{{ worker_app_dir }}	директория с исполняемым приложением мастера и конфигурацией
{{ app_name }}	уникальное имя сервиса
{{ logs_folder }}	путь для папки с логами
{{ ssl_folder }}	путь до папки с сертификатами

3.4 Настройка в АРМ Навигатор

1. В интерфейсе Навигатора перейти в **АРМ:Администрирование – Система – Настройки – Опции реализации**
2. Установить значение параметра `enable-voice-assistant` в true, если такой параметр отсутствует – создать его.
3. Перейти на обзорный экран, обновить страницу и проверить результат установки (в правом нижнем углу на обзорном экране должна появится кнопка ИИ-помощника).

3.5 Настройка автоматической разметки дешбордов

Для работы скрипта по автоматической разметке дешбордов в ИИ выполните следующие действия:

1. В разделе Данные - Справочники выберите системные. Создайте системный справочник, назовите его "Атрибут "Статус"".
   Этот справочник будет определять статус Эксплуатации дешборда. Пример

   ```
   select sname, svalue, nord
   from (values ('Тест', 'Тест', 0),
                ('MVP', 'MVP', 1),
                ('ПРОМ', 'ПРОМ', 2)
         )t(sname, svalue, nord)
   ```

   Где статусы эксплуатации = Тест, MVP, ПРОМ
2. Создайте дополнительный атрибут дешборда, назовите его "Статус эксплуатации", тип данных = "Справочник"
3. Выберите справочник, созданный в п.1
4. Включите параметры "Обязательный атрибут" и "Хранить значения"
5. При разработке дешборда меняйте статусы эксплуатации, либо установите статус НЕ равный "Тест" с nord = 0, для того чтобы при создании устанавливался он и дешборд попадал в ИИ
6. Скрипт разметки смотрит на этот атрибут, если он НЕ равен "Тест" , тогда дешборд, его экраны и виджеты автоматически попадают в очередь на обучение. Контролы(фильтры) автоматически не размечаются.
7. Если пользователь через АРМ убирает с объекта галочку "Доступен в ИИ помощнике" такое состояние сохраняется, данный объект больше не попадет в автоматическую разметку.
8. Скрипт активен при условии запущенного крона на БД
9. Скрипт отрабатывает каждые 5 минут

3.6 Настройка Portal Server

1. В файле configSources.yml (или navigator-master-config) прописать путь до конфигурационного файла, добавив новый блок для ИИ (AiModuleConfig.yml). Важно чтобы файлы находились в одной папке

   1. **блок ИИ**

      ```
      ai-module:
        nameSpace: portal-server
        type: yaml
        priority: 5
        path: AiModuleConfig.yml
        poll-period: 60s
      ```

2. Скопировать конфигурационный файл `AiModuleConfig.yml` на сервер приложений Навигатора в папку, где располагается файл configSources.yml `%путь_к_WildFly%/standalone/configuration/app_props/` , прописать необходимые параметры:

   - В параметрах `previewModelUrl` и `targetModelUrl` в случае использования одной среды можно закомментировать один из блоков (например preview) , в случае двух и более сред – Ip-адреса серверов для Preview и Target. Эти адреса можно посмотреть в `%путь_к_WildFly%/standalone/configuration/standalone.xml` в секции `<system-properties>`

3. В случае использования **https** – необходимо раскомментировать в каждой секции конфигурационного файла строки `certProps` с параметрами сертификатов.
4. Перезапустить WildFly;

4 FAQ по целевой архитектуре

4.1 Как проверить работоспособность приложений ?

Необходимо сделать запрос GET для https://host.ru/admin/health AI Master и https://host.ru/health/check AI Worker Manager. Если у вас включена аутентификация по сертификату, то предварительно внесите ваш сертификат в security-config.yml

4.2 Как настраивается sslType?

sslType может принимать несколько значений:

CERT_KEY - связка сертификат + ключ

sslType

```
sslType: CERT_KEY
certProps:
    cert: test.pem
    key: test.key
```

CERT_KEY_CA - связка сертификат + ключ + корневая цепочка

sslType

```
sslType: CERT_KEY
certProps:
    cert: test.pem
    key: test.key
    ca: test.pem
```

CERT_KEY_PASSWORD - сертификат + пароль от него

sslType

```
sslType: CERT_KEY_PASSWORD
certProps:
    cert: test.p12
    password: testpassword # должен быть зашифрован с помощью encryptor.key
```

SECMAN - получение сертификата с указанным CN из секмана, работает только при указании секции secman в core-props.yml

sslType

```
sslType: SECMAN
certProps:
    cert: CI123456-DEV-AI-WORKER.pem # может еще принимать значение CI123456-DEV-AI-WORKER.p12
```

IGNORE_SSL - интеграция без сертификатов

sslType

```
sslType: IGNORE_SSL
# секции certProps нет
```