ФГБОУ ВО «ПИМУ» Минздрава России

ИНСТРУКЦИЯ ПО УСТАНОВКЕ ЭКЗЕМПЛЯРА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

«Мультипараметрическая Интеллектуальная Система Анализа физиологических показателей (МИСА)»

(для проведения экспертной проверки)

Нижний Новгород,2026

Оглавление

1. ОБЩИЕ ПОЛОЖЕНИЯ

Настоящая инструкция разработана в соответствии с требованиями:

- Постановления Правительства Российской Федерации от 16.11.2015 № 1236 «Об утверждении Правил формирования и ведения единого реестра российских программ для электронных вычислительных машин и баз данных»;
- Методических рекомендаций по работе с Федеральной государственной информационной системой «Реестры программ для электронных вычислительных машин и баз данных» (раздел 2.1.4);
- ГОСТ 19.101-77 «Единая система программной документации. Виды программ и программных документов».

1.2. Инструкция содержит пошаговое описание действий по установке, настройке и развёртыванию программного обеспечения **«Мультипараметрическая Интеллектуальная Система Анализа физиологических показателей (МИСА)»** (далее – ПО) в тестовой среде для проведения экспертной проверки.

1.3. ПО предназначено для автоматизированного анализа вариабельности сердечного ритма и функционирует в режиме размещения на инфраструктуре правообладателя с предоставлением доступа внешним пользователям посредством защищённого интерфейса прикладного программирования.

1.4. Установка ПО осуществляется путём развёртывания контейнеризированного приложения на сервере под управлением операционной системы семейства Linux.

2. ТРЕБОВАНИЯ К АППАРАТНОМУ И ПРОГРАММНОМУ ОБЕСПЕЧЕНИЮ

2.1. Требования к аппаратному обеспечению сервера

Параметр	Минимальные	Рекомендуемые

	требования	**требования**
Процессор	x86_64, 2 ядра	x86_64, 4 ядра
Оперативная память	4 гигабайта	8 гигабайт
Дисковое пространство	20 гигабайт	50 гигабайт
Сетевой интерфейс	100 Мбит/с	1 Гбит/с

2.2. Требования к программному обеспечению сервера

• Linux (Debian 11+ / Ubuntu 22.04 LTS)

• Docker Engine 24.0+

• Docker Compose 2.20+

2.3. Сетевые требования

• Открытые порты: **80 (HTTP)** и/или **443 (HTTPS)** – для веб-интерфейса администратора и API

3. ПОДГОТОВКА СЕРВЕРА К УСТАНОВКЕ

3.1. Обновление пакетной базы операционной системы

sudo apt update && sudo apt upgrade –y

3.2. Установка системы контейнеризации Docker и Docker Compose

sudo apt install -y ca-certificates curl gnupg

sudo install -m 0755 -d /etc/apt/keyrings

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg

echo "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list

sudo apt update && sudo apt install -y docker-ce docker-ce-cli containerd.io docker compose-plugin

sudo usermod -aG docker $USER && newgrp docker

3.3. Проверка установки Docker

docker –version

docker compose version

3.4. Создание рабочего каталога для размещения ПО

sudo mkdir -p /opt/misa

sudo chown $USER:$USER /opt/misa

cd /opt/misa

4. ПОЛУЧЕНИЕ ДИСТРИБУТИВА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

4.1. Дистрибутив предоставляется в виде архива .zip Скопируйте архив на сервер и распакуйте:

Пример для архива с именем misa.zip

unzip misa.zip -d /opt/misa

4.2. В распакованном архиве содержится:

- app/ – исходный код
- alembic/, alembic.ini – миграции БД
- nginx/nginx.conf – конфигурация reverse proxy
- Dockerfile, docker compose.yml – контейнеризация
- .env.example
- requirements.txt – зависимости Python
- model.enc

5. НАСТРОЙКА ПЕРЕМЕННЫХ ОКРУЖЕНИЯ

5.1. Создание файла конфигурации переменных окружения:

cp .env.example .env

5.2. Редактирование файла .env:

nano .env

5.3. Пример содержимого файла .env для тестовой среды:

ini

```
# Database
POSTGRES_HOST=postgres
POSTGRES_PORT=5432
POSTGRES_USER=postgres
POSTGRES_PASSWORD=HRvTestPass001
POSTGRES_DB=hrvdb
DATABASE_URL=postgresql://postgres:HRvTestPass001@postgres:5432/hrvdb

# JWT
SECRET_KEY=testsecretkeyforexpertreview2026
ALGORITHM=HS256
ACCESS_TOKEN_EXPIRE_MINUTES=1440

# Model
MODEL_KEY=-0wZOyFvqpFcyQGHkfLlU7HO5-hg19h_-Szvt1DWRa8=
MODEL_PATH=model.enc

# Storage
ANALYSES_STORAGE_PATH=/app/storage/analyses

# Logging
LOG_LEVEL=INFO
LOG_JSON=false

# CORS
```

BACKEND_CORS_ORIGINS=["*"]

6. РАЗВЁРТЫВАНИЕ С ПОМОЩЬЮ DOCKER COMPOSE

6.1. Запуск контейнеров:

docker compose up –d

6.2. Проверка статуса контейнеров:

docker compose ps

Ожидаемый результат: все три сервиса (postgres, app, nginx) должны иметь статус "Up".

6.3. Применение миграций базы данных:

docker compose exec app alembic upgrade head

6.4. Инициализация начальных данных:

docker compose exec app python -c "from app.seed import run_seeders; run_seeders()"

6.5. Проверка логов приложения:

docker compose logs -f app

7. ПРОВЕРКА РАБОТОСПОСОБНОСТИ

7.1. Проверка доступности веб-интерфейса

Открыть в браузере адрес:

http://<IP-адрес-сервера>

7.2. Проверка точки здоровья (health check):

curl http://localhost/health

Ожидаемый ответ:

Json

{"status": "ok"}

7.3. Проверка доступности документации API:

Swagger UI: http://<IP-адрес-сервера>/docs

ReDoc: http://<IP-адрес-сервера>/redoc

8. УЧЁТНЫЕ ДАННЫЕ ДЛЯ ЭКСПЕРТНОЙ ПРОВЕРКИ

8.1. Учётная запись администратора (создаётся автоматически при инициализации):

Параметр	Значение
Адрес электронной почты	admin@example.com
Пароль	admin123
Роль	ADMIN

Внимание: После первой авторизации рекомендуется сменить пароль администратора.

8.2. Доступ к интерфейсу прикладного программирования:

Аутентификация осуществляется двумя способами:

Способ	Параметр	Пример
JWT-токен	Заголовок Authorization	Bearer <токен>
API-ключ	Заголовок X-API-Key	<ключ>

8.3. Получение JWT-токена:

```
curl -X POST http://localhost/api/v1/auth/login \
-H "Content-Type: application/json" \
-d '{"email":"admin@example.com","password":"admin123"}'
```

8.4. Пример запроса к API с токеном:

```
curl -X GET http://localhost/api/v1/auth/me \
-H "Authorization: Bearer <полученный_токен>"
```

9. ОСОБЕННОСТИ АРХИТЕКТУРЫ И ДОСТУПА

9.1. Модель развёртывания

ПО функционирует как saas решение. Доступ к функционалу осуществляется посредством:

- Веб-интерфейса администратора (для управления пользователями и настройки системы);

- http запросов к API приложения.

9.2. Отсутствие пользовательского интерфейса:

Графический пользовательский интерфейс предназначен для административных задач. Конечные пользователи взаимодействуют с системой через API.

9.3. Отсутствие средств активации:

ПО не использует технические средства активации, лицензионные ключи или механизмы привязки к оборудованию. Доступ контролируется посредством токенов аутентификации.

10. УСТРАНЕНИЕ ВОЗМОЖНЫХ ПРОБЛЕМ

10.1. Контейнеры не запускаются:

```
docker compose down
docker compose up -d
docker compose logs
```

10.2. Ошибка подключения к базе данных:

Проверить параметры в файле .env:

```
cat .env | grep POSTGRES
```

10.3. Ошибка миграций:

```
docker compose exec app alembic downgrade -1
docker compose exec app alembic upgrade head
```

10.4. Недостаточно дискового пространства:

```
docker system prune -a
df –h
```

11. КОНТАКТЫ ТЕХНИЧЕСКИХ СПЕЦИАЛИСТОВ

11.1. Для консультации по процессу развёртывания и настройки экземпляра ПО:

Канал	Контакт
Электронная почта	otl@pimunn.net

11.2. Режим работы службы поддержки:

Понедельник – Пятница: с 09:00 до 18:00 (по московскому времени)

Перерыв: с 13:00 до 14:00

Срок реакции на обращение: не более 24 часов в рабочее время

12. ПРИЛОЖЕНИЯ

Приложение А. Структура каталогов ПО

```
/opt/hrv-analyzer/
├── alembic/                    # Скрипты миграции базы данных
├── app/                        # Исходный код приложения
│   ├── api/                    # Эндпоинты API
│   ├── core/                   # Базовые настройки
│   ├── models/                 # Модели данных
│   ├── processing/             # Модули обработки данных ВСР
│   ├── services/               # Бизнес-логика
│   ├── static/                 # Статические файлы
│   ├── web/                    # Веб-интерфейс
│   └── main.py                 # Точка входа
├── nginx/                      # Конфигурация веб-сервера
├── docker-compose.yml          # Конфигурация контейнеров
├── Dockerfile                  # Образ приложения
├── .env                        # Переменные окружения
├── .env.example                # Пример переменных окружения
└── requirements.txt            # Зависимости Python
```

Приложение Б. Команды управления контейнерами

Команда	Описание
`docker compose up -d`	Запуск всех сервисов в фоновом режиме
`docker compose down`	Остановка и удаление контейнеров
`docker compose ps`	Просмотр статуса контейнеров
`docker compose logs -f`	Просмотр логов в реальном времени
`docker compose restart`	Перезапуск всех сервисов
`docker compose exec app <команда>`	Выполнение команды внутри контейнера приложения

Приложение В. Эндпоинты

Метод	Эндпоинт	Описание	Аутентификация
`GET`	`/`	Главная страница (веб-интерфейс)	Нет
`GET`	`/login`	Страница входа	Нет
`POST`	`/api/v1/auth/login`	Получение JWT-токена	Нет
`POST`	`/api/v1/auth/logout`	Выход из системы (инвалидация токена на клиенте)	JWT / API-ключ
`GET`	`/api/v1/auth/me`	Информация о текущем пользователе	JWT / API-ключ
`POST`	`/api/v1/analyses`	Загрузка файла для анализа (синхронно или асинхронно)	JWT / API-ключ
`GET`	`/api/v1/analyses`	Список анализов текущего пользователя	JWT / API-ключ
`GET`	`/api/v1/analyses/{analysis_id}`	Информация об анализе	JWT / API-ключ
`GET`	`/api/v1/analyses/{analysis_id}/result`	JSON-результат (только для `response_type=json`)	JWT / API-ключ
`GET`	`/api/v1/analyses/{analysis_id}/download`	Скачивание ZIP-архива с отчётом	JWT / API-ключ
`DELE`	`/api/v1/analyses/`	Удаление анализа	JWT / API-ключ

TE`	{analysis_id}`		
`GET`	`/api/v1/admin/users`	Список пользователей (только админ)	JWT / API-ключ
`POST`	`/api/v1/admin/users`	Создание пользователя (админ)	JWT / API-ключ
`GET`	`/api/v1/admin/users/{user_id}`	Данные пользователя (админ)	JWT / API-ключ
`PUT`	`/api/v1/admin/users/{user_id}`	Обновление пользователя (админ)	JWT / API-ключ
`POST`	`/api/v1/admin/users/{user_id}/api-key`	Генерация API-ключа (админ)	JWT / API-ключ
`PATCH`	`/api/v1/admin/users/{user_id}/block`	Блокировка/ разблокировка (админ)	JWT / API-ключ
`GET`	`/api/v1/admin/plans`	Список тарифных планов (админ)	JWT / API-ключ
`GET`	`/admin/users`	Веб-страница управления пользователями	JWT (требуется роль `ADMIN`)
`GET`	`/admin/users/new`	Веб-страница создания пользователя	JWT (требуется роль `ADMIN`)
`GET`	`/admin/users/{user_id}`	Веб-страница редактирования пользователя	JWT (требуется роль `ADMIN`)
`GET`	`/health`	Проверка работоспособности (health check)	Нет
`GET`	/api/v1/analyses/{analysis_id}/result	JSON-результат (только для response_type=json)	JWT / API-ключ