ФГБОУ ВО «ПИМУ» Минздрава России

РУКОВОДСТВО АДМИНИСТРАТОРА

«Мультипараметрическая Интеллектуальная Система Анализа физиологических показателей (МИСА)»

Версия 1.0.0

ГОСТ Р 59795-2021

Нижний Новгород

2026

Оглавление

1. ОБЩИЕ СВЕДЕНИЯ

1.1. Назначение администратора

Администратор программного обеспечения «МИСА» обеспечивает установку, настройку, поддержание работоспособности и сопровождение системы. К функциональным обязанностям администратора относятся:

- первоначальное развёртывание ПО на серверном оборудовании;
- конфигурирование переменных окружения и параметров запуска;
- управление учётными записями пользователей, API-ключами и тарифными планами;
- ежедневный контроль работоспособности компонентов;
- выполнение резервного копирования и восстановления;
- обновление системы до новых версий;
- первичная диагностика и устранение неисправностей.

1.2. Требования к квалификации

- опыт администрирования Linux-систем (Ubuntu 22.04 LTS / Debian 11+);
- уверенное владение командной оболочкой bash ;
- знание принципов работы и команд Docker и Docker Compose;
- понимание основ TCP/IP, маршрутизации и межсетевого экрана (firewall);
- базовые навыки работы с базами данных PostgreSQL (pg_dump, psql).

1.3. Состав компонентов системы

Система построена по микросервисной архитектуре. Все компоненты выполняются в Docker-контейнерах и оркестрируются через Docker Compose.

Компонент	Технология	Назначение
Backend	Python 3.10+ / FastAPI	REST API, бизнес-логика, расчёт HRV-метрик, генерация отчётов
База данных	PostgreSQL 15+	Хранение пользователей, тарифов, метаданных анализов
Nginx	Nginx 1.29	Обратный прокси, раздача статики, HTTPS-терминация
Файловое хранилище	Docker volume	Хранение архивов с результатами анализов
ML-модель	scikit-learn, cryptography	Предсказание риска ПТСР (зашифрованный файл model.enc)

2. РАЗВЁРТЫВАНИЕ СИСТЕМЫ

2.1. Подготовка серверного окружения

Установите операционную систему **Ubuntu 22.04 LTS** в минимальной конфигурации. Обновите пакеты и установите вспомогательные утилиты:

sudo apt update && sudo apt upgrade -y

sudo apt install -y git curl make ca-certificates

Настройте firewall, открыв порты, необходимые для работы системы и удалённого администрирования:

sudo ufw allow 22/tcp

sudo ufw allow 80/tcp

sudo ufw allow 443/tcp

sudo ufw enable

Аппаратные требования:

Параметр	Минимальные требования	Рекомендуемые требования
Процессор	2 ядра, x86-64	4 ядра и более
Оперативная память	4 ГБ	8 ГБ и более
Дисковое пространство	20 ГБ (SSD)	50 ГБ и более (SSD)
Сетевой интерфейс	100 Мбит/с	1 Гбит/с

2.2. Установка Docker и Docker Compose

Установите Docker по официальной инструкции:

1. Обновите список пакетов и установите зависимости

sudo apt update

sudo apt install -y ca-certificates curl

2. Добавьте официальный GPG-ключ Docker

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

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

sudo chmod a+r /etc/apt/keyrings/docker.asc

3. Добавьте репозиторий Docker

echo "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc] https://download.docker.com/linux/ubuntu $(. /etc/os-release && echo "$VERSION_CODENAME") stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

4. Установите Docker Engine и плагин Docker Compose

sudo apt update

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

Добавьте текущего пользователя в группу docker:

sudo usermod -aG docker $USER

Проверьте установку:

docker --version && docker compose version

2.3. Получение дистрибутива

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

Пример для архива с именем hrv-analyzer.zip

unzip hrv-analyzer.zip -d /opt/misa

2.4. Настройка переменных окружения

Создайте файл .env из примера:

cp .env.example .env

Все необходимые переменные уже предопределены в файле .env.example.

Перед развёртыванием в производственном окружении необходимо заменить значения по умолчанию на собственные надёжные**.**

Ниже приведён полный список переменных с их назначением:

Переменная	**Описание**
POSTGRES_HOST	Имя хоста PostgreSQL (должно совпадать с именем сервиса в docker-compose)
POSTGRES_PORT	Порт PostgreSQL
POSTGRES_USER	Имя пользователя PostgreSQL

Переменная	Описание
POSTGRES_PASSWORD	Пароль PostgreSQL
POSTGRES_DB	Имя базы данных
DATABASE_URL	URL подключения к БД
SECRET_KEY	Секретный ключ для JWT (минимум 32 символа)
ALGORITHM	Алгоритм подписи JWT
ACCESS_TOKEN_EXPIRE_MINUTES	Время жизни токена в минутах
MODEL_KEY	Ключ для расшифровки model.enc
MODEL_PATH	Путь к файлу зашифрованной модели
ANALYSES_STORAGE_PATH	Путь для хранения результатов анализов
LOG_LEVEL	Уровень логирования (DEBUG/INFO/WARNING/ERROR)
LOG_JSON	Формат логов (JSON или текст)
BACKEND_CORS_ORIGINS	Разрешённые источники CORS

Рекомендации по замене:

Генерация надёжного SECRET_KEY (не менее 32 символов)

python -c "import secrets; print(secrets.token_hex(32))"

Генерация надёжного пароля для PostgreSQL

python -c "import secrets; print(secrets.token_urlsafe(20))"

Важно: Никогда не используйте значения по умолчанию в производственном окружении.

2.5. Сборка и запуск контейнеров

Запустите все сервисы в фоновом режиме:

docker compose up -d

Проверьте статус контейнеров:

docker compose ps

Проверьте работоспособность API:

curl http://localhost/health

Ожидаемый ответ: {"status":"ok"}

2.6. Загрузка ML-модели

Поместите предобученную модель прогнозирования риска ПТСР зашифрованный файл model.enc в корневую директорию проекта и укажите MODEL_KEY в .env. При старте приложения модель автоматически загрузится.

2.7. Инициализация базы данных

Перед первым запуском необходимо применить миграции и загрузить начальные данные (тарифные планы, учётная запись администратора):

1. Применить все миграции базы данных (создание таблиц)

docker compose exec app alembic upgrade head

2. Заполнить начальными данными (тарифные планы + учётная запись администратора)

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

2.8. Настройка автозапуска (systemd)

Для автоматического запуска системы после перезагрузки сервера создайте systemd-юнит:

sudo nano /etc/systemd/system/misa.service

Содержимое файла:

```
[Unit]
Description=MISA Docker Compose stack
Requires=docker.service
After=docker.service

[Service]
Type=oneshot
RemainAfterExit=yes
WorkingDirectory=/opt/misa
ExecStart=/usr/bin/docker compose up -d
ExecStop=/usr/bin/docker compose down
User=root

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

Активируйте сервис:

sudo systemctl daemon-reload

sudo systemctl enable misa.service

sudo systemctl start misa.service

Проверка статуса автозапуска:

sudo systemctl status misa.service

3. УПРАВЛЕНИЕ СИСТЕМОЙ

3.1. Команды управления контейнерами

Команда	Назначение
docker compose up -d	Запуск всех контейнеров в фоновом режиме
docker compose down	Штатная остановка и удаление контейнеров (тома сохраняются)
docker compose down -v	Остановка и удаление контейнеров вместе с томами (необратимо!)
docker compose ps	Просмотр статусов контейнеров
docker compose logs -f	Поток журналов всех сервисов
docker compose logs -f app	Журналы только сервера приложений
docker compose restart app	Перезапуск контейнера приложения
docker compose pull	Обновление Docker-образов из реестра
docker stats	Просмотр потребления ресурсов в реальном времени
docker compose exec app <команда>	выполнение произвольной команды внутри контейнера приложения.

3.2. Перезапуск отдельных сервисов

Для перезапуска одного сервиса без остановки остальных:

docker compose restart app

docker compose restart postgres

docker compose restart nginx

Перезапуск рекомендуется выполнять при изменении конфигурации сервиса или при нештатном поведении.

3.3. Обновление до новой версии

Типовая последовательность обновления:

1. Создать резервную копию (см. раздел 6).
2. Остановить систему:

docker compose down

3. Получить новый дистрибутив (архив), распаковать его в отдельный каталог.
4. Скопировать файлы .env и model.enc (если есть) из старой версии.
5. Пересобрать образы и запустить систему:

docker compose build

docker compose up –d

6. Применить миграции базы данных:

docker compose exec app alembic upgrade head

7. Проверить работоспособность:

curl http://localhost/health

docker compose ps

4. УПРАВЛЕНИЕ ПОЛЬЗОВАТЕЛЯМИ И ТАРИФАМИ

4.1. Веб-интерфейс администратора

Веб-интерфейс ориентирован на администратора. Конечные пользователи взаимодействуют с системой через REST API (описано в отдельном «Руководстве пользователя»).

Доступ к веб-интерфейсу осуществляется по адресу: http://<IP_сервера>/ (после входа в систему).

4.2. Учётная запись администратора по умолчанию

После инициализации начальных данных (п. 2.5) создаётся администратор:

- **Email:** admin@example.com

- **Пароль:** admin123

Важно: При первом входе смените пароль администратора через раздел редактирования пользователя.

4.3. Управление пользователями

Перейдите в админ-панель: нажмите ссылку **«Админ-панель»** в правом верхнем углу или откройте http://<IP>/admin/users.

Доступные действия:

- **Просмотр списка пользователей** – таблица с колонками: Email, Роль, Тарифный план, Использовано запросов, Статус (активен/заблокирован).
- **Создание пользователя** – кнопка «+ Создать пользователя». Поля: Email, Пароль, Роль (Обычный / Администратор), Тарифный план, Флаг «Активен».
- **Редактирование пользователя** – изменение email, пароля, роли, плана, статуса.
- **Блокировка / разблокировка** – заблокированный пользователь не может войти в систему и использовать API.

4.4. Выдача и отзыв API-ключей

API-ключи предназначены для автоматизированной загрузки файлов конечными пользователями через REST API.

1. В списке пользователей найдите нужного.
2. Нажмите кнопку **«API-ключ»**.
3. В появившемся окне будет отображён **новый API-ключ**. **Сохраните его сразу**, так как после закрытия окна ключ будет недоступен.
4. Передайте ключ пользователю защищённым каналом.

Примечание: При повторном нажатии «API-ключ» старый ключ аннулируется и генерируется новый.

4.5. Управление тарифными планами

Система поддерживает три типа тарифных планов:

Тип плана	Параметр ограничения	Описание
UNLIMITED	–	Без ограничений на количество анализов
LIMITED_COUNT	max_requests_total	Ограничение по суммарному числу запросов
LIMITED_INTERVAL	max_requests_per_interval + interval_days	Ограничение за период (дней)

Планы подписки предопределены в коде сидера (файл app/seed/seed_plans.py). Для добавления, изменения или удаления плана необходимо:

1. Отредактировать файл app/seed/seed_plans.py, изменив список plans_data.
2. Пересобрать образ приложения и запустить сидер:

docker compose build app

docker compose up -d

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

5. МОНИТОРИНГ И НАБЛЮДАЕМОСТЬ

5.1. Просмотр журналов (логов)

Журналы каждого контейнера выводятся в stdout/stderr и собираются Docker.

docker compose logs *# все сервисы, единовременно*

docker compose logs -f *# поток в реальном времени*

docker compose logs --tail 100 *# последние 100 строк*

docker compose logs app *# только бэкенд*

docker compose logs db *# только PostgreSQL*

Уровень детализации задаётся переменной окружения LOG_LEVEL (DEBUG / INFO / WARNING / ERROR).

5.2. Проверка работоспособности (health-check)

Проверка состояния всех компонентов:

curl http://localhost/health

Корректный ответ – HTTP 200 и JSON: {"status":"ok"}. При статусе HTTP 503 один или несколько компонентов недоступны – обратитесь к разделу 8 «Диагностика и устранение неисправностей».

5.3. Мониторинг ресурсов

- **Потребление ресурсов контейнерами:** docker stats
- **Занятое дисковое пространство:** df -h
- **Список запущенных контейнеров:** docker compose ps

При необходимости можно подключить дополнительные системы мониторинга (Prometheus, Grafana), но штатными средствами они не поставляются.

6. РЕЗЕРВНОЕ КОПИРОВАНИЕ И ВОССТАНОВЛЕНИЕ

6.1. Что подлежит резервному копированию

- файл .env с параметрами окружения и секретами;
- файл model.enc (если используется ML-модель);
- база данных PostgreSQL (пользователи, тарифы, метаданные анализов);
- Docker-том analyses_storage (сохранённые ZIP-архивы с результатами).

6.2. Создание резервной копии

Резервное копирование базы данных:

docker compose exec -T postgres pg_dump -U postgres hrvdb | gzip >

/backup/hrvdb_$(date +%Y%m%d_%H%M%S).sql.gz

Копирование конфигурации и модели:

cp /opt/misa/.env /backup/

cp /opt/misa/model.enc /backup/

Копирование Docker-тома с результатами:

#Определите фактическое имя тома (обычно <имя_папки_проекта>_analyses_storage)

docker volume ls | grep analyses_storage

 # Пример копирования (замените <project_name> на реальное)

docker run --rm -v <project_name>_analyses_storage:/src -v /backup:/dst alpine tar czf /dst/analyses_storage.tar.gz -C /src .

Рекомендуется хранить резервные копии не менее 30 дней на отдельном носителе и шифровать (например, AES-256).

6.3. Восстановление из резервной копии

1. Остановить систему:

cd /opt/misa

docker compose down

2. Восстановить файл .env и model.enc:

cp /backup/.env .

cp /backup/model.enc . # если есть

3. Восстановить базу данных:

   gunzip -c /backup/hrvdb_20260331.sql.gz | docker compose exec -T postgres psql -U postgres -d hrvdb

4. Восстановить Docker-том с результатами (при необходимости):

 # Используйте фактическое имя тома (см. п. 6.2)

docker volume rm <project_name>_analyses_storage

docker volume create <project_name>_analyses_storage

docker run --rm -v <project_name>_analyses_storage:/dst -v /backup:/src alpine tar xzf /src/analyses_storage.tar.gz -C /dst

5. Запустить систему и проверить работоспособность:

docker compose up -d

curl http://localhost/health

7. БЕЗОПАСНОСТЬ

7.1. Управление секретами

- Не размещайте файл .env под системой контроля версий – добавьте его в .gitignore.
- Сгенерируйте надёжный SECRET_KEY (см. п. 2.4).
- При компрометации SECRET_KEY – замените его в .env и перезапустите приложение. Все действующие JWT-токены станут недействительными.
- Для смены API-ключа пользователя используйте кнопку «API-ключ» в админ-панели (старый ключ аннулируется автоматически).

7.2. Настройка HTTPS (SSL/TLS)

Для защиты трафика необходимо настроить HTTPS с помощью Let’s Encrypt:

sudo apt install certbot python3-certbot-nginx

sudo certbot --nginx -d ваш_домен.ru

После получения сертификата обновите конфигурацию Nginx (файл nginx/nginx.conf). Пример блока server для HTTPS:

nginx

server {

listen 443 ssl;

server_name ваш_домен.ru;

```
    ssl_certificate /etc/letsencrypt/live/ваш_домен.ru/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/ваш_домен.ru/privkey.pem;
    # ... остальные настройки
}
server {
    listen 80;
    server_name ваш_домен.ru;
    return 301 https://$host$request_uri;
}
```

Проверьте автоматическое обновление сертификатов:

```
sudo certbot renew --dry-run
```

7.3. CORS и сетевая изоляция

В производственном окружении ограничьте CORS-источники в файле .env:

BACKEND_CORS_ORIGINS=["https://ваш_домен.ru"]

Также рекомендуется ограничить доступ к административным портам (80, 443, 22) с помощью firewall, разрешив только доверенные IP-адреса.

8. ДИАГНОСТИКА И УСТРАНЕНИЕ НЕИСПРАВНОСТЕЙ

Симптом	Возможная причина	Действия по устранению
Контейнер app не запускается, статус Exited	Ошибка в .env или недоступна БД	Просмотреть журнал: docker compose logs app; проверить DATABASE_URL; убедиться, что контейнер db запущен
HTTP 502 Bad Gateway	Приложение упало или не запустилось	docker compose restart app; проверить логи app
HTTP 500	Проблема соединения с	docker compose ps; проверить

Симптом	Возможная причина	Действия по устранению
на /health	БД	логи db; перезапустить БД: docker compose restart db
alembic upgrade завершается ошибкой	Несовместимость схемы или нет соединения с БД	Проверить DATABASE_URL; выполнить alembic current; восстановить БД из резервной копии
Логи содержат "Model loading failed"	Неверный MODEL_KEY или отсутствует model.enc	Проверить .env; убедиться в наличии файла; при отсутствии модели отключить её (не задавать MODEL_KEY)
Анализ завис в статусе processing	Процесс-обработчик завис	Перезапустить app: docker compose restart app; искать ERROR в логах с нужным analysis_id
Высокое потребление RAM	Утечка памяти / большой размер файлов	docker stats; проверить наличие неочищенных файлов в temp_files/; увеличить ресурсы сервера
Пользователь не может войти (401)	Истёкший или недействительный токен	Повторный логин; проверить ACCESS_TOKEN_EXPIRE_MINUTES в .env
Ошибка "rate limit exceeded"	Превышен лимит тарифного плана	Проверить requests_used_total через API; при необходимости сменить тарифный план в админ-панели
Нет ответа от API (таймаут)	Перегружен сервер или сетевые проблемы	Проверить docker compose ps; увеличить ресурсы; проверить

Симптом	Возможная причина	Действия по устранению
		firewall
Не удаётся подключиться к БД извне	PostgreSQL слушает только localhost	По умолчанию БД доступна только внутри Docker-сети. Для внешних подключений настройте проброс порта в docker-compose.yml (не рекомендуется для продакшн)

9. ТЕХНИЧЕСКАЯ ПОДДЕРЖКА

При возникновении вопросов и инцидентов:

- **Электронная почта:** otl@pimunn.net
- **Режим работы:** пн–пт, 09:00–18:00 по московскому времени

Адрес: 603000, Нижегородская область, г. Нижний Новгород, пл. Минина и Пожарского, д. 10/1

При обращении приложите:

- выгрузку журналов за период инцидента (docker compose logs --no-color > logs.txt);
- краткое описание обстоятельств и ваши действия

Регламентные сроки реагирования и устранения определены в документе «Описание процессов, обеспечивающих поддержание жизненного цикла программного обеспечения»..