ФГБОУ ВО «ПИМУ» Минздрава России

ИНСТРУКЦИЯ ПО ЭКСПЛУАТАЦИИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

«Мультипараметрическая Интеллектуальная Система Анализа физиологических показателей (МИСА)»

Версия 1.0.0

ГОСТ 19.505-79

Нижний Новгород

2026

Оглавление

1. НАЗНАЧЕНИЕ ПРОГРАММЫ

1.1. Функциональное назначение

Программа «Мультипараметрическая Интеллектуальная Система Анализа физиологических показателей (МИСА)» предназначена для автоматизированной обработки записей RR-интервалов (вариабельность сердечного ритма) и расчёта следующих параметров: временные показатели (SDNN, RMSSD, pNN50, TINN, AMo), спектральные показатели (LF, HF, LF/HF), индекс стресса CS, а также вероятностная оценка риска посттравматического стрессового расстройства (ПТСР) на основе математической модели. Программа реализована в виде REST API с веб-интерфейсом для администратора.

1.2. Эксплуатационное назначение

Программа эксплуатируется в следующих режимах:

- круглосуточный режим работы (24/7) с регламентными технологическими паузами;
- многопользовательский режим — одновременная работа до 25 пользовательских сессий (лимиты определяются тарифным планом);
- сервисный (административный) режим — выполнение операций администрирования (управление пользователями, тарифными планами, просмотр журналов, резервное копирование).

2. УСЛОВИЯ ВЫПОЛНЕНИЯ ПРОГРАММЫ

2.1. Требования к аппаратному обеспечению

Минимальная конфигурация серверного оборудования:

- процессор — 2 ядра x86_64;
- оперативная память — 4 ГБ;
- дисковое пространство — 20 ГБ SSD;
- сетевой интерфейс — 100 Мбит/с.

Рекомендуемая конфигурация:

- процессор — 4 ядра x86_64;
- оперативная память — 8 ГБ;
- дисковое пространство — 50 ГБ NVMe SSD;
- сетевой интерфейс — 1 Гбит/с.

2.2. Требования к программному обеспечению

Требования к серверу:

- операционная система: Ubuntu 22.04 LTS (рекомендуется), Debian 11+, а также отечественные ОС на базе Linux, включённые в реестр российского ПО;
- Docker Engine версии 24.0 и выше;
- Docker Compose версии 2.20 и выше;
- текстовый редактор для правки конфигурационных файлов (nano, vim).

Требования к рабочему месту администратора:

- современный веб-браузер с поддержкой HTML5 и JavaScript (Google Chrome, Mozilla Firefox, Yandex Browser актуальных версий);
- HTTP-клиент (curl, Postman, любой язык программирования) для тестирования API.

Требования к конечному пользователю (работа через API):

- любая среда, поддерживающая отправку HTTP-запросов (Python, Java, JavaScript, curl).

2.3. Требования к сетевой инфраструктуре

- пропускная способность канала Интернет — не менее 10 Мбит/с;
- открытые порты сервисов системы:

– 80/TCP (HTTP) и 443/TCP (HTTPS) — внешние, для доступа пользователей и администратора;

– 8000/TCP (FastAPI) — доступен только внутри Docker-сети (Nginx проксирует на него);– 5432/TCP (PostgreSQL) — доступен только внутри Docker-сети.

3. ВЫПОЛНЕНИЕ ПРОГРАММЫ

3.1. Загрузка программы

Под загрузкой программы понимается размещение дистрибутива на сервере и подготовка окружения к запуску. Загрузка выполняется однократно администратором при первоначальном развёртывании или при обновлении версии.

Последовательность действий по загрузке программы:

1. Получить дистрибутив в виде архива (например, misa-1.0.0.tar.gz или misa-1.0.0.zip) от разработчика.

2. Создать рабочий каталог:

sudo mkdir -p /opt/misa

sudo chown $USER:$USER /opt/misa

cd /opt/misa

3. Распаковать архив:

unzip /path/to/misa-1.0.0.zip -d /opt/misa

4. Создать файл конфигурации окружения на основе шаблона:

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

Переменная	Описание
	именем сервиса в docker-compose)
POSTGRES_PORT	Порт PostgreSQL
POSTGRES_USER	Имя пользователя PostgreSQL
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_ORI	Разрешённые источники CORS

Переменная	Описание
GINS	

5. При наличии зашифрованной ML-модели (model.enc) поместить её в корневой каталог.

6. Выполнить сборку Docker-образов:

docker compose build

3.2. Запуск программы
Команда штатного запуска:

docker compose up -d

Параметр -d (detached) обеспечивает запуск контейнеров в фоновом режиме.

Проверка успешного запуска:

docker compose ps

Все контейнеры (app, postgres , nginx) должны отображаться в состоянии Up.

Применение миграций базы данных (только при первом запуске или после обновления):

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()"

Учётная запись администратора по умолчанию

После выполнения сидеров создаётся администратор с следующими учётными данными:

- ***Email: admin@example.com***
- ***Пароль: admin123***

После старта становятся доступны следующие адреса:

Сервис	**Адрес**
Веб-интерфейс администратора	http://<IP_сервера>/
REST API (базовый URL)	http://<IP_сервера>/api/v1/
Swagger-документация API	http://<IP_сервера>/docs
ReDoc-документация	http://<IP_сервера>/redoc

3.3. Выполнение программы

В режиме штатного выполнения программа автоматически принимает HTTP-запросы от пользователей и администраторов через REST API и веб-интерфейс. Вмешательство администратора в штатном режиме не требуется.

Алгоритм обработки одного запроса (анализ файла):

1. Пользователь отправляет POST-запрос на /api/v1/analyses с файлом CSV/XLSX.

2. Backend аутентифицирует пользователя (JWT или API-ключ).

3. Проверяется соответствие формата файла и тарифные ограничения.

4. Файл сохраняется во временную директорию, в БД создаётся запись анализа.

5. Запускается асинхронная обработка (или синхронная, в зависимости от параметра sync).

6. Выполняется расчёт HRV-метрик, генерация отчётов (DOCX, XLSX, PNG).

7. Результат сохраняется в постоянное хранилище.

8. Пользователь получает результат (JSON или ссылку на скачивание ZIP-архива).

3.4. Завершение работы программы

Штатное завершение работы программы:

docker compose down

Команда останавливает все контейнеры системы и освобождает ресурсы. Данные пользователей и анализов сохраняются в Docker-томах и становятся доступны при последующем запуске.

Принудительное завершение (только при нештатной ситуации):

docker compose kill

Применяется в случае, если контейнеры не отвечают на штатное завершение. Может привести к потере данных активных сессий.

4. КОМАНДЫ УПРАВЛЕНИЯ И ОТВЕТЫ ПРОГРАММЫ

4.1. Команды управления серверной частью

Команды выполняются администратором из каталога проекта /opt/misa в командной оболочке операционной системы.

Команда	**Назначение**	**Ответ программы**
docker compose up -d	Запуск всех компонентов системы	Запускаются контейнеры; код возврата 0 при успехе
docker compose down	Штатная остановка всех компонентов	Контейнеры останавливаются и удаляются; тома сохраняются
docker compose ps	Просмотр статуса контейнеров	Таблица со статусами (Up, Exited)
docker compose logs -f	Просмотр журналов в реальном времени	Поток сообщений всех сервисов; завершение по Ctrl+C
docker compose restart app	Перезапуск только сервера приложений	Сервис перезапускается без остановки других
docker compose exec app	Открыть shell внутри контейнера приложения	Интерактивная оболочка
docker compose exec app alembic upgrade head	Применить миграции БД	Вывод информации о выполненных миграциях
docker stats	Просмотр потребления ресурсов	Таблица с CPU/MEM/I-O для всех контейнеров
curl http://localhost/health	Проверка работоспособности API	{"status":"ok"} или HTTP 503

4.2. Эндпоинты REST API

Серверная часть предоставляет программный интерфейс REST API. Полная интерактивная документация доступна по адресу http://<IP_сервера>/docs (Swagger UI).

Метод и путь	Назначение	Ответ программы
POST /api/v1/auth/login	Получение JWT-токена	JSON с access_token; HTTP 200 при успехе
GET /api/v1/auth/me	Информация о текущем пользователе	JSON с полями email, role; HTTP 200
POST /api/v1/analyses	Загрузка файла для анализа	При sync=true — ZIP-архив или JSON; при sync=false — analysis_id
GET /api/v1/analyses/{analysis_id}	Статус анализа	JSON со статусом (pending, processing, completed, failed)
GET /api/v1/analyses/{analysis_id}/result	JSON-результат (только для response_type=json)	{"ptsd_pred": 1, "ptsd_proba": 0.84}
GET /api/v1/analyses/{analysis_id}/download	Скачивание ZIP-архива с отчётами	Файл result.zip (DOCX, XLSX, PNG)
DELETE /api/v1/analyses/{analysis_id}	Удаление анализа	HTTP 204 (без тела)
GET /health	Проверка	{"status":"ok"}

Метод и путь	Назначение	Ответ программы
	работоспособности	
POST /api/v1/admin/users/{user_id}/api-key	Генерация нового API-ключа для пользователя (только для администратора)	JSON с полем api_key; HTTP 200

4.3. Элементы веб-интерфейса администратора

Веб-интерфейс доступен по адресу http://<IP_сервера>/ и предназначен только для администратора.

Элемент интерфейса	Назначение	Реакция программы
Страница /login	Вход в систему	Форма ввода email и пароля; после успешного входа — перенаправление на /
Главная страница /	Тестирование анализа (загрузка файла)	Выбор файла CSV/XLSX, кнопка «Анализировать», скачивание ZIP-архива с результатами
Кнопка «Админ-панель»	Переход к управлению пользователями	Открывается страница /admin/users
Таблица пользователей /admin/users	Просмотр, создание, редактирование пользователей	Список с колонками: Email, Роль, Тариф, Использовано запросов, Статус, Действия
Кнопка «Создать API-ключ»	Генерация API-ключа для	Отображение нового ключа; старый ключ аннулируется

Элемент интерфейса	Назначение	Реакция программы
	пользователя	
Кнопка «Заблокировать/Разблокировать»	Блокировка учётной записи	Пользователь теряет доступ к API

5. СООБЩЕНИЯ ОПЕРАТОРУ

5.1. Сообщения пользовательского интерфейса

При работе с веб-интерфейсом администратора программа выдаёт следующие сообщения:

Тип сообщения	Описание / условие появления	Действие администратора
Успешный вход	После ввода правильных email/пароля	Перенаправление на главную страницу
Ошибка входа	Неверный email или пароль	Повторить ввод; при утере пароля — восстановить через БД
Анализ завершён	После успешной обработки файла	Браузер автоматически скачивает ZIP-архив
Ошибка файла	Неверный формат, отсутствие столбцов, превышение лимитов	Исправить файл в соответствии с требованиями
Пользователь создан	Успешное создание учётной записи	Отображение в таблице пользователей

Тип сообщения	Описание / условие появления	Действие администратора
API-ключ сгенерирован	Ключ показан в диалоговом окне	**Скопировать и сохранить ключ**; после закрытия окна ключ недоступен

5.2. Системные сообщения и коды возврата

При выполнении команд управления и обращении к REST API система возвращает стандартные коды:

Код / сообщение	**Значение**	**Действие администратора**
Exit code 0	Команда выполнена успешно	Действия не требуются
Exit code != 0	Ошибка выполнения команды Docker	Просмотреть журналы: docker compose logs
HTTP 200 OK	Запрос к API выполнен успешно	Действия не требуются
HTTP 400 Bad Request	Неверный формат запроса (например, отсутствует файл)	Проверить тело запроса по схеме Swagger
HTTP 401	Отсутствует или	Проверить

Код / сообщение	**Значение**	**Действие администратора**
Unauthorized	неверный токен/API-ключ	заголовки Authorization или X-API-Key
HTTP 403 Forbidden	Недостаточно прав (пользователь заблокирован)	Разблокировать пользователя в админ-панели
HTTP 404 Not Found	Эндпоинт или анализ не существует	Проверить путь или analysis_id
HTTP 429 Too Many Requests	Превышен лимит запросов по тарифу	Дождаться окончания интервала или сменить тариф
HTTP 500 Internal Server Error	Внутренняя ошибка серверной части (включая ошибки структуры файла)	Просмотреть журналы: docker compose logs app
HTTP 503 Service Unavailable	Модель машинного обучения не загружена	Использовать response_type=zip или обратиться к администратору

Код / сообщение	Значение	Действие администратора
	(запрошен JSON)	

Важно: При получении HTTP-статуса 500 Internal Server Error следует в первую очередь проверить соответствие входного файла требованиям (наличие обязательных столбцов Сцена (этап), Время, rr, hr). Подробности ошибки записаны в журналы контейнера app.

5.3. Действия при ошибках

Типовая последовательность действий при возникновении ошибки:

1. Зафиксировать текст ошибки и обстоятельства (время, запрос, analysis_id).

2. Просмотреть журналы соответствующего сервиса:

docker compose logs -f app

docker compose logs -f postgres

docker compose logs -f nginx

Проверить состояние всех контейнеров:

docker compose ps

3. При необходимости перезапустить проблемный сервис:

docker compose restart app

4. Если ошибка не устраняется штатными средствами, направить запрос в службу технической поддержки по адресу otl@pimunn.net с приложением выгрузки журналов и описания обстоятельств.