МБорд РУКОВОДСТВО АДМИНИСТРАТОРА

Общие положения

Настоящее руководство описывает порядок администрирования программного обеспечения «МБорд» — веб-платформы мониторинга промышленного оборудования, разработанной ООО «Метрикор».

Документ предназначен для системных администраторов и инженеров, ответственных за настройку, сопровождение и управление системой «МБорд» в рамках промышленной эксплуатации.

Область применения: администрирование через веб-интерфейс системы и конфигурирование серверной части.

Требования к квалификации администратора:

- Понимание принципов веб-приложений (HTTP, REST API, JWT)
- Опыт работы с Docker и Docker Compose
- Базовые знания PostgreSQL/TimescaleDB
- Понимание принципов мониторинга промышленного оборудования

Версия системы: 2.3.0+

---

1. Управление пользователями

1.1 Создание пользователя через веб-интерфейс

Для создания нового пользователя:

1. Перейдите в раздел **«Пользователи»** в боковом меню.
2. Нажмите кнопку **«Добавить пользователя»**.
3. Заполните форму создания:

Поле	Обязательное	Описание
Имя	Да	Полное имя пользователя (2–100 символов)
Email	Да	Уникальный адрес электронной почты
Пароль	Да	Минимум 8 символов, заглавная буква, цифра, спецсимвол
Роль	Да	Одна из: admin, engineer, operator, viewer

1. Нажмите **«Создать»**.

При успешном создании пользователь получает возможность авторизации с указанными учётными данными.

1.2 Роли пользователей

Система поддерживает 4 роли с разграничением прав доступа:

- **admin** — полный доступ ко всем функциям системы, управление пользователями и конфигурацией
- **engineer** — управление оборудованием, контроллерами и пользователями (кроме admin)
- **operator** — оперативный мониторинг, подтверждение и разрешение алертов
- **viewer** — просмотр дашборда и алертов без возможности модификации

1.3 Деактивация пользователя

Для деактивации учётной записи:

1. Откройте карточку пользователя в разделе **«Пользователи»**.
2. Нажмите кнопку **«Деактивировать»**.
3. Подтвердите действие в диалоговом окне.

Деактивированный пользователь:

- Не может авторизоваться в системе
- Его активные сессии аннулируются немедленно
- Данные о пользователе сохраняются в системе (для аудита)
- Может быть повторно активирован администратором

Важно: невозможно деактивировать последнюю учётную запись с ролью admin.

1.4 Seed accounts (начальные учётные записи)

При первом запуске системы или при необходимости гарантированного наличия администраторских учётных записей используется механизм seed accounts.

Начальные учётные записи задаются через переменную окружения `SEED_ACCOUNTS_JSON`:

```
[
  {
    "email": "admin@metrikor.ru",
    "password": "SecurePassword123!",
    "full_name": "Администратор",
    "role": "admin"
  },
  {
    "email": "engineer@metrikor.ru",
    "password": "EngineerPass456!",
    "full_name": "Инженер",
    "role": "engineer"
  }
]
```

Поведение seed accounts:

- Создаются автоматически при запуске приложения
- Если пользователь с указанным email уже существует — запись не перезаписывается
- Пароли хранятся в bcrypt-хешированном виде
- Рекомендуется сменить пароли seed accounts после первого входа

---

2. Ролевая модель и матрица прав

2.1 Матрица доступа к разделам

Раздел / Функция	admin	engineer	operator	viewer
Дашборд — просмотр	✅	✅	✅	✅
Дашборд — переключение режимов	✅	✅	✅	✅
Алерты — просмотр	✅	✅	✅	✅
Алерты — подтверждение	✅	✅	✅	❌
Алерты — разрешение	✅	✅	❌	❌
Алерты — удаление	✅	❌	❌	❌
Контроллеры — просмотр	✅	✅	✅	❌
Контроллеры — управление	✅	✅	❌	❌
Пользователи — просмотр	✅	✅	❌	❌
Пользователи — создание	✅	✅*	❌	❌
Пользователи — деактивация	✅	❌	❌	❌
Настройка алертов (пороги)	✅	✅	❌	❌
Экспорт данных (Excel)	✅	✅	✅	✅
API документация	✅	✅	❌	❌

** engineer может создавать только пользователей с ролями operator и viewer*

2.2 Принципы разграничения

- Доступ проверяется на уровне API (backend) — обход через прямые запросы невозможен
- JWT-токен содержит информацию о роли пользователя
- При изменении роли требуется повторная авторизация для обновления токена
- Все действия логируются с привязкой к пользователю

---

3. Управление контроллерами

3.1 Добавление контроллера

Для добавления нового контроллера в систему:

1. Перейдите в раздел **«Контроллеры»**.
2. Нажмите **«Добавить контроллер»**.
3. Заполните форму:

Поле	Обязательное	Описание
Название	Да	Уникальное имя контроллера
Модель	Нет	Модель оборудования (например, Segnetics SMH5)
Серийный номер	Нет	Серийный номер устройства
Описание	Нет	Дополнительная информация

1. Нажмите **«Создать»**.

После создания контроллер появляется в списке с состоянием «Оффлайн» до момента получения первых данных.

3.2 Активация и деактивация

Деактивация контроллера:

- Прекращает приём данных от контроллера
- Датчики контроллера переходят в состояние «Оффлайн»
- Исторические данные сохраняются
- Алерты по датчикам деактивированного контроллера не генерируются

Активация контроллера:

- Возобновляет приём данных
- Датчики переходят в состояние «Онлайн» при получении новых данных
- Восстанавливается генерация алертов

3.3 Механизм поступления данных

Контроллеры Segnetics записывают показания в CSV-файлы. Компонент FileWatcher отслеживает появление новых файлов и загружает данные в TimescaleDB:

```
Контроллер → CSV файл → FileWatcher → Backend API → TimescaleDB
```

Формат CSV-файла:

- Разделитель: точка с запятой (`;`)
- Кодировка: UTF-8
- Первая строка: заголовки (timestamp, sensor_name, value)
- Временная метка: ISO 8601

---

4. Настройка алертов

4.1 Пороговые правила

Система алертов основана на пороговых правилах. Для каждого датчика задаются четыре пороговых значения:

Параметр	Описание	Результат при нарушении
min_critical	Нижняя критическая граница	Алерт уровня critical
min_normal	Нижняя граница нормы	Алерт уровня warning
max_normal	Верхняя граница нормы	Алерт уровня warning
max_critical	Верхняя критическая граница	Алерт уровня critical

Логика срабатывания:

- Значение < min_critical → critical alert
- min_critical ≤ Значение < min_normal → warning alert
- min_normal ≤ Значение ≤ max_normal → норма (нет алерта)
- max_normal < Значение ≤ max_critical → warning alert
- Значение > max_critical → critical alert

4.2 Настройка порогов для датчика

1. Перейдите на детальную страницу датчика.
2. Нажмите **«Настройка порогов»** (доступно для admin и engineer).
3. Заполните пороговые значения:

```
 min_critical: -10.0
min_normal:    5.0
max_normal:   80.0
max_critical: 95.0
```

1. Нажмите **«Сохранить»**.

Рекомендации по настройке порогов:

- Изучите паспорт оборудования для определения допустимых диапазонов
- Установите нормальные значения с запасом от критических (не менее 10%)
- Учитывайте сезонные колебания параметров
- После настройки проведите тестирование на исторических данных

4.3 Массовая настройка

Для однотипного оборудования доступна массовая настройка порогов:

1. Выберите несколько датчиков одного типа.
2. Нажмите **«Массовая настройка порогов»**.
3. Задайте значения — они применятся ко всем выбранным датчикам.

---

5. Мониторинг системы

5.1 Health endpoint

Для проверки состояния системы доступен эндпоинт:

```
GET /api/v1/health
```

Ответ при нормальной работе:

```
{
  "status": "healthy",
  "database": "connected",
  "redis": "connected",
  "version": "2.3.0",
  "uptime_seconds": 86400
}
```

Ответ при наличии проблем:

```
{
  "status": "degraded",
  "database": "connected",
  "redis": "disconnected",
  "version": "2.3.0",
  "uptime_seconds": 86400
}
```

Рекомендуется настроить внешний мониторинг (Prometheus, Zabbix, Grafana) для периодического опроса health endpoint.

5.2 API документация

Интерактивная документация API доступна по адресу:

```
https://<domain>/api/v1/docs
```

Документация сгенерирована автоматически на основе FastAPI/OpenAPI и содержит:

- Полный список эндпоинтов
- Схемы запросов и ответов
- Возможность выполнения тестовых запросов
- Описание моделей данных

Доступ к документации ограничен ролями admin и engineer.

5.3 Логирование

Система ведёт структурированные логи (JSON format):

Компонент	Расположение	Описание
Backend	stdout (Docker)	API запросы, ошибки, бизнес-логика
FileWatcher	stdout (Docker)	Обработка CSV файлов, ошибки парсинга
Nginx	/var/log/nginx/	HTTP запросы, ошибки проксирования
PostgreSQL	stdout (Docker)	SQL запросы (slow log > 1s)

Для просмотра логов Docker-контейнеров:

```
docker compose logs -f backend
docker compose logs -f file-watcher
docker compose logs --tail=100 timescaledb
```

---

6. Конфигурация окружения

6.1 Переменные окружения

Система конфигурируется через переменные окружения, задаваемые в файле `.env` или через Docker Compose:

Переменная	Обязательная	Описание	Пример
`DATABASE_URL`	Да	Строка подключения к TimescaleDB	`postgresql+asyncpg://user:pass@timescaledb:5432/mboard`
`REDIS_URL`	Да	Строка подключения к Redis	`redis://redis:6379/0`
`SECRET_KEY`	Да	Секретный ключ для подписи JWT	Случайная строка 32+ символов
`SEED_ACCOUNTS_JSON`	Нет	JSON-массив начальных учётных записей	См. раздел 1.4
`LOG_LEVEL`	Нет	Уровень логирования	`INFO` (по умолчанию)
`CORS_ORIGINS`	Нет	Разрешённые CORS-домены	`https://demo.metrikor.ru`
`ACCESS_TOKEN_EXPIRE_MINUTES`	Нет	Время жизни JWT-токена (минуты)	`1440` (24 часа)
`POSTGRES_USER`	Да	Пользователь PostgreSQL	`mboard`
`POSTGRES_PASSWORD`	Да	Пароль PostgreSQL	Случайная строка
`POSTGRES_DB`	Да	Имя базы данных	`mboard`
`REDIS_MAXMEMORY`	Нет	Лимит памяти Redis	`200mb`
`FILE_WATCHER_PATH`	Нет	Директория для мониторинга CSV	`/data/csv`
`VITE_API_URL`	Да	URL API для фронтенда	`https://demo.metrikor.ru/api/v1`

6.2 Генерация SECRET_KEY

Для генерации безопасного секретного ключа:

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

Или:

```
openssl rand -base64 32
```

Важно: при смене SECRET_KEY все текущие сессии пользователей будут аннулированы (JWT-токены станут невалидными).

6.3 Развёртывание (обзор)

Система состоит из 4 Docker-контейнеров:

Контейнер	Образ	Порт	Назначение
timescaledb	timescale/timescaledb:latest-pg16	5432	База данных с поддержкой временных рядов
redis	redis:7	6379	Кэширование и брокер сообщений
backend	mboard-backend	8000	API сервер (FastAPI + Uvicorn)
frontend	mboard-frontend	80	Веб-интерфейс (Nginx + React SPA)

Развёртывание выполняется через Docker Compose:

```
docker compose up -d
```

6.4 Резервное копирование

Рекомендуемый порядок резервного копирования:

База данных (ежедневно):

```
docker compose exec timescaledb pg_dump -U mboard mboard > backup_$(date +%Y%m%d).sql
```

Конфигурация:

- Файл `.env` с переменными окружения
- Файл `docker-compose.yml`
- Директория с CSV-файлами (при необходимости)

Восстановление из резервной копии:

```
docker compose exec -T timescaledb psql -U mboard mboard < backup_20260420.sql
```

6.5 Обновление системы

Порядок обновления:

1. Создайте резервную копию базы данных.
2. Остановите контейнеры: `docker compose down`
3. Обновите образы: `docker compose pull`
4. Запустите контейнеры: `docker compose up -d`
5. Проверьте health endpoint: `curl http://localhost:8000/api/v1/health`
6. Проверьте логи на наличие ошибок: `docker compose logs --tail=50`

Миграции базы данных применяются автоматически при запуске backend-контейнера (Alembic).

Приложение А. Устранение типичных проблем

Проблема	Возможная причина	Решение
Не удаётся авторизоваться	Неверный пароль или деактивированный аккаунт	Проверить аккаунт в БД, сбросить пароль через seed
Датчики показывают «Оффлайн»	Нет новых CSV-файлов	Проверить FileWatcher и доступ к директории CSV
Высокое потребление памяти	Большой объём данных в Redis	Проверить REDIS_MAXMEMORY, очистить кэш
Медленные запросы	Отсутствие индексов или сжатия	Проверить slow log PostgreSQL, настроить compression
502 Bad Gateway	Backend не запущен	Проверить `docker compose ps`, логи backend
Не генерируются алерты	Не настроены пороги	Настроить пороговые значения для датчиков

Приложение Б. Контакты

Параметр	Значение
Организация	ООО «Метрикор»
Email поддержки	info@metrikor.ru
Демо-стенд	https://demo.metrikor.ru