""

РУКОВОДСТВО АДМИНИСТРАТОРА

программы для ЭВМ «Автоматизированная система управления строительными проектами «Мегаполис»», версия 1.0.0

Правообладатель: Индивидуальный предприниматель Костылев Петр Николаевич (ИНН 245722035517, ОГРНИП 325246800083856) **Редакция документа:** 1.1 **Дата утверждения:** 19.06.2026

Оглавление

1. Введение
2. Архитектура развертывания
3. Управление учетными записями пользователей
4. Управление правами доступа (RBAC)
5. Аудит действий администратора
6. Резервное копирование и восстановление данных
7. Мониторинг работы Программы
8. Обновление Программы до новых версий
9. Управление информационной безопасностью
10. Типовые проблемы и устранение
11. Взаимодействие с правообладателем
12. Приложения

---

1. Введение

1.1. Назначение документа

Руководство администратора предназначено для лиц, которым на стороне заказчика поручено администрирование Программы. Руководство охватывает:

- настройку учетных записей и прав доступа;
- резервное копирование и восстановление данных;
- мониторинг работоспособности;
- обновление до новых версий;
- информационную безопасность.

1.2. Предварительные требования к администратору

Администратор должен обладать следующими знаниями и навыками:

- базовые навыки работы с операционной системой Linux (Astra Linux 1.7, РЕД ОС 7.3 или Ubuntu 22.04/24.04);
- понимание основ контейнеризации Docker и оркестрации Docker Compose;
- начальные знания о работе PostgreSQL;
- понимание принципов сетевой безопасности (TLS/HTTPS, firewall);
- владение русским языком на уровне, необходимом для чтения документации.

Специфические знания по программной разработке не требуются — администрирование Программы не предполагает модификации исходного кода.

1.3. Учетная запись администратора

На этапе установки создается учетная запись с правами администратора. Администратор может создавать другие учетные записи, в том числе с правами администратора. Для работы с Программой администратор использует модуль «Администрирование» в веб-интерфейсе.

Для процедур, не отраженных в веб-интерфейсе (резервное копирование, обновление, диагностика), используется интерфейс командной строки операционной системы сервера.

---

2. Архитектура развертывания

2.1. Состав компонентов

Программа поставляется в виде стека Docker Compose, содержащего три основных контейнера и два внешних компонента:

Компонент	Docker-образ / исполняемый файл	Назначение
PostgreSQL 16	`postgres:16-alpine`	Хранение прикладных данных
Серверная часть	Собирается из `./backend/Dockerfile` (образ Python 3.12-slim + приложение)	Серверная часть, программный интерфейс REST, WebSocket, фоновые задания
Nginx	`nginx:1.27-alpine`	Обратный прокси, отдача статических файлов клиентской части, завершение TLS
Клиентская часть (frontend)	Собирается отдельно (команда `npm run build`), результат монтируется в Nginx	Веб-интерфейс в браузере пользователя
Резервное хранилище	Каталог `/var/backups/megapolis/` (рекомендация)	Резервные копии базы данных и загруженных файлов

Все компоненты размещены в изолированной docker-сети (`asu_demo_network` по умолчанию).

2.2. Структура каталога развертывания

```
/opt/megapolis/
├── backend/              # Исходный код серверной части (не требует изменений администратором)
├── frontend/
│   ├── src/              # Исходный код клиентской части
│   └── dist/             # Собранные статические файлы, монтируются в Nginx
├── database/
│   └── init-scripts/     # SQL-скрипты начальной инициализации и миграций
├── nginx/
│   ├── nginx.conf
│   ├── conf.d/
│   │   └── default.conf
│   └── ssl/              # TLS-сертификаты (Let's Encrypt или внутренние)
├── scripts/              # Скрипты обслуживания (backup, restore, build)
├── docker-compose.yml    # Конфигурация стека
├── .env                  # Переменные окружения (конфиденциально)
└── .env.example          # Шаблон переменных окружения
```

2.3. Сетевые порты

Порт	Назначение	Открывается наружу
80	Входящий HTTP (Nginx); автоматический редирект на HTTPS	Да (интернет)
443	Входящий HTTPS (Nginx)	Да (интернет)
5432 (внутри контейнера)	PostgreSQL	Только в пределах docker-сети
8000 (внутри контейнера)	Серверная часть	Только в пределах docker-сети

Для отладки можно опубликовать порт `127.0.0.1:5433 → postgres:5432` (указано в `docker-compose.yml`). В продуктивной эксплуатации рекомендуется закомментировать эту строку.

2.4. Переменные окружения (`.env`)

Переменная	Пример значения	Назначение
`POSTGRES_SERVER`	`postgres`	Имя хоста PostgreSQL (= имя сервиса в docker-compose)
`POSTGRES_PORT`	`5432`	Порт PostgreSQL
`POSTGRES_USER`	`asu_demo`	Имя пользователя PostgreSQL (рекомендация: не использовать `postgres`)
`POSTGRES_PASSWORD`	криптостойкая строка из `openssl rand -hex 24`	Пароль пользователя PostgreSQL
`POSTGRES_DB`	`asu_demo`	Имя базы данных
`SECRET_KEY`	криптостойкая строка из `openssl rand -hex 32` (не менее 64 hex-символов)	Криптографический ключ для JWT и HMAC-подписей refresh-токенов
`ACCESS_TOKEN_EXPIRE_MINUTES`	`600`	Срок действия access-токена (по умолчанию 10 часов)
`REFRESH_TOKEN_EXPIRE_DAYS`	`7`	Срок действия refresh-токена
`MAX_FAILED_LOGIN_ATTEMPTS`	`10`	Количество неудачных попыток до блокировки
`ACCOUNT_LOCK_DURATION_MINUTES`	`15`	Длительность блокировки учетной записи
`ENVIRONMENT`	`production`	Режим работы: `production` (строгая безопасность) или `development`
`DEBUG`	`false`	Включение отладочной документации API (`/docs`, `/redoc`) — в продуктивной эксплуатации держать `false`
`APP_NAME`	`АСУ СП Мегаполис Community`	Название приложения, отображаемое в заголовке
`CORS_ALLOW_ORIGINS`	`https://megapolis.example.ru`	Список разрешенных источников для CORS, разделенных запятой
`CORS_ALLOW_CREDENTIALS`	`true`	Разрешение передачи cookies с запросами

Переменная	Пример значения	Назначение
`RATE_LIMIT_DEFAULT`	`600/hour`	Общий лимит запросов с одного IP
`RATE_LIMIT_LOGIN`	`15/15minutes`	Лимит попыток входа
`RATE_LIMIT_REFRESH`	`30/15minutes`	Лимит попыток обновления токена
`NGINX_HTTP_PORT`	`8080`	Публикуемый HTTP-порт на хосте (опционально)
`NGINX_HTTPS_PORT`	`8443`	Публикуемый HTTPS-порт на хосте (опционально)
`POSTGRES_HOST_PORT`	`5433`	Публикуемый порт PostgreSQL (только для отладки)

Важно. Файл `.env` содержит критически важные секреты и должен быть защищен:

- права на файл `600` (только для пользователя-владельца);
- файл не должен попадать в системы контроля версий;
- при передаче между сотрудниками — использовать защищенные каналы.

---

3. Управление учетными записями пользователей

3.1. Создание пользователя

1. Войдите в Программу под учетной записью с правами администратора.
2. Перейдите в модуль «Администрирование» → раздел «Пользователи».
3. Нажмите кнопку **Создать пользователя**.
4. Заполните форму:
    - «ФИО» (обязательное, формат «Фамилия Имя Отчество» на русском языке);
    - «Логин» (не обязательно — генерируется автоматически из ФИО; можно задать вручную);
    - «Должность» (произвольный текст);
    - «Рабочий телефон» (произвольный текст);
    - «Электронная почта» (не обязательно, но рекомендуется);
    - Чек-бокс «Администратор системы» (назначать только при необходимости);
    - Список модулей, к которым предоставляется доступ:
        - «Мониторинг и аналитика» (employee_da);
        - «Сметы» (estimator_os);
        - «Инженер» (engineer_oks).

- Для каждого модуля — выбор «полный доступ» (редактирование) или «только просмотр».
- «Модуль по умолчанию» — модуль, который открывается при входе пользователя.

5. Нажмите **Сохранить**.
6. Программа сгенерирует пароль и отобразит его **один раз** в модальном окне:
   - для обычных пользователей — 9-символьный пароль (4 строчные буквы + 4 цифры + 1 спецсимвол);
   - для администраторов — 24-символьный пароль со смешанным регистром и спецсимволами.
7. **Обязательно запишите или скопируйте пароль** до закрытия окна — после закрытия он не может быть восстановлен.
8. Передайте пароль пользователю через защищенный канал (корпоративный мессенджер, запечатанный конверт).

3.2. Редактирование пользователя

1. В разделе «Пользователи» найдите учетную запись через поле поиска либо фильтры.
2. Нажмите на строку пользователя либо кнопку «Редактировать».
3. Измените нужные поля.
4. Нажмите **Сохранить**.

Нельзя изменить логин после создания учетной записи (это защита от случайного нарушения прав доступа).

3.3. Сброс пароля пользователя

1. В разделе «Пользователи» выберите учетную запись.
2. Нажмите **Сбросить пароль**.
3. Подтвердите действие.
4. Программа сгенерирует новый пароль и покажет его один раз.
5. **Одновременно** будут завершены все активные сессии этого пользователя (пользователь будет принудительно выведен из системы и должен войти с новым паролем).
6. Передайте новый пароль пользователю.

3.4. Деактивация учетной записи

Применяется при увольнении сотрудника, переводе на другую должность, длительном отсутствии.

1. В разделе «Пользователи» выберите учетную запись.
2. Нажмите **Деактивировать**.
3. Подтвердите действие.

4. Программа:
   - пометит учетную запись как неактивную (`is_active = FALSE`);
   - немедленно завершит все активные сессии пользователя.
5. Пользователь более не сможет войти в систему. При необходимости можно позднее активировать учетную запись тем же способом (кнопка **Активировать**).

Нельзя деактивировать собственную учетную запись (защита от случайной потери доступа).

3.5. Разблокировка учетной записи

Если пользователь заблокирован после 10 неудачных попыток входа:

1. В разделе «Пользователи» найдите заблокированную учетную запись (обычно в таблице отображается индикатор «Заблокирована до …»).
2. Нажмите **Разблокировать**.
3. Счетчик неудачных попыток будет сброшен, пользователь сможет войти с текущим паролем.

Если пользователь забыл пароль и заблокирован — сбросьте пароль (пункт 3.3), разблокировка произойдет автоматически.

3.6. Защита учетной записи `demo` на демонстрационном стенде

На демонстрационном стенде Программы используется учетная запись `demo / demo` с флагом «Администратор системы». Для обеспечения доступности этой учетной записи в базе данных установлены триггеры:

- любые попытки изменить пароль, имя пользователя, флаг активности, флаг «Администратор системы» или установить блокировку — отменяются;
- учетная запись всегда сохраняет пароль `demo`, флаги `is_active = TRUE` и `is_admin = TRUE`;
- счетчик неудачных попыток обнуляется автоматически.

Триггеры находятся в SQL-скрипте `database/init-scripts/03_seed_demo_user.sql` (объект `auth.protect_demo_user`).

В продуктивных развертываниях эти триггеры должны быть удалены (либо не устанавливаться вовсе). В продуктивной среде используются обычные учетные записи с реальными паролями.

---

4. Управление правами доступа (RBAC)

4.1. Модель прав доступа

Программа реализует ролевую модель доступа (Role-Based Access Control, RBAC) через связку «пользователь — модуль — право редактирования»:

- **Учетная запись пользователя** (таблица `auth.users`) — идентифицирует пользователя.
- **Модуль** (таблица `auth.modules`) — четыре пользовательских модуля + модуль администрирования.

- **Связь «пользователь — модуль»** (таблица `auth.user_modules`) — определяет, к каким модулям у пользователя есть доступ, и имеет ли он право редактировать (`can_edit`).
- **Флаг «Администратор системы»** (`users.is_admin = TRUE`) — отдельно; предоставляет полный доступ к модулю «Администрирование» и автоматически расширяет права в остальных модулях.

4.2. Типовые ролевые шаблоны

Роль	Доступные модули	Права
Инспектор ООиАД (просмотр)	Мониторинг и аналитика, Инженер	Только просмотр
Специалист ООиАД	Мониторинг и аналитика, Инженер	Полный доступ
Сметчик	Сметы	Полный доступ
Инженер ОКС	Инженер, Сметы	Полный доступ
Руководитель	Мониторинг и аналитика, Инженер, Сметы	Полный доступ во всех модулях
Администратор системы	Все модули + Администрирование	Полный доступ; флаг «Администратор системы»

4.3. Рекомендации по распределению прав

1. **Принцип минимально необходимых прав.** Каждой учетной записи присваиваются только те модули и уровни доступа, которые необходимы для выполнения ее рабочих обязанностей.
2. **Минимум администраторов.** Флаг «Администратор системы» присваивается ограниченному кругу лиц (1–2 человека на 50 пользователей; меньше при централизованном администрировании).
3. **Разделение обязанностей.** Пользователь, отвечающий за ведение данных (например, специалист ООиАД), не должен одновременно иметь административных прав.
4. **Регулярный пересмотр.** Раз в полугодие — проверка перечня учетных записей на предмет актуальности (увольнения, перевод на другие должности).

---

5. Аудит действий администратора

5.1. Состав журнала

Все административные действия фиксируются в таблице `auth.admin_audit_log`:

Поле	Содержание
`log_id`	Уникальный идентификатор записи
`actor_user_id`	Идентификатор администратора, выполнившего действие
`action`	Тип действия (`user.create`, `user.update`, `user.deactivate`, `user.activate`, `user.unlock`, `user.password_reset`, `user.modules_update`)
`target_type`	Тип объекта действия (`user`)
`target_id`	Идентификатор объекта действия (UUID учетной записи)
`details`	Подробности изменения в формате JSON (измененные поля, их новые значения)
`ip_address`	IP-адрес, с которого выполнено действие
`user_agent`	User-Agent браузера администратора
`created_at`	Дата и время действия (с часовым поясом)

5.2. Доступ к журналу

Пользовательский интерфейс просмотра журнала в Программе отсутствует (сокращенная поверхность атаки). Для получения данных журнала используется SQL-запрос к базе данных.

Пример запроса. Отчет о всех действиях администратора за последние 30 дней:

```sql
SELECT
    a.created_at AT TIME ZONE 'Asia/Krasnoyarsk' AS "дата и время",
    u.username AS "администратор",
    u.full_name AS "ФИО",
    a.action AS "действие",
    a.target_id AS "идентификатор объекта",
    a.ip_address AS "IP-адрес",
    a.details AS "подробности"
FROM auth.admin_audit_log a
LEFT JOIN auth.users u ON u.user_id = a.actor_user_id
WHERE a.created_at >= NOW() - INTERVAL '30 days'
ORDER BY a.created_at DESC;
```

Выполняется через:

1. Подключение к контейнеру PostgreSQL:

```
docker exec -it asu_demo_postgres psql -U <POSTGRES_USER> -d <POSTGRES_DB>
```

2. Выполнение SQL-запроса.

3. Экспорт результата в Excel/CSV через стандартные средства `psql` (`\copy ... TO 'audit.csv' CSV HEADER`).

5.3. Срок хранения журнала

Записи журнала хранятся 90 календарных дней. По истечении срока они автоматически удаляются при старте backend (функция `auth.cleanup_old_audit_logs(90)`).

При необходимости более длительного хранения (например, для комплаенс-требований ФСТЭК, ФСБ по объектам КИИ) — администратор инфраструктуры настраивает отдельную процедуру архивирования журнала во внешнюю SIEM-систему.

---

6. Резервное копирование и восстановление данных

6.1. Стратегия резервного копирования

Рекомендуемая стратегия.

- Ежедневно — полный дамп базы данных PostgreSQL (сжатый pg_dump) и архивирование загруженных файлов.
- Ежемесячно — отдельное хранение полного резервного набора на независимом физическом носителе или в защищенном удаленном хранилище.
- Перед обновлением до новой версии — внеплановое резервное копирование.

Рекомендуется хранить копии не менее:

- 7 последних ежедневных;
- 4 последних еженедельных;
- 12 последних ежемесячных;
- неограниченное количество копий перед обновлениями (до успешного запуска новой версии).

6.2. Процедура создания резервной копии

В состав дистрибутива Программы включен готовый production-скрипт `app/scripts/backup-db.sh`. Скрипт реализует следующие возможности:

- безопасное чтение реквизитов подключения из `app/.env` (с проверкой наличия и непустоты обязательных переменных);
- проверку, что контейнер PostgreSQL запущен;
- атомарное создание дампа через временный файл `*.partial` с последующим переименованием — это исключает ситуацию, когда параллельный процесс восстановления подхватит «недоделанный» дамп;
- сжатие дампа в формате `pg_dump -F c | gzip -9` (кастомный формат, пригодный для `pg_restore`);
- проставление прав 0600 на готовый дамп;

- автоматическую ротацию: удаление дампов старше `BACKUP_RETENTION_DAYS` дней (по умолчанию 30);
- структурированный вывод с UTC-timestamp, пригодный для отправки в систему централизованного логирования;
- 5 кодов возврата (0 — успех, 1 — отсутствуют предусловия, 2 — нет docker/gzip, 3 — ошибка дампа, 4 — ошибка ротации) для надежного контроля в задачах cron.

Запуск скрипта вручную:

```
cd /opt/megapolis
./app/scripts/backup-db.sh                    # копия в /var/backups/megapoli
./app/scripts/backup-db.sh /custom/path       # копия в указанный каталог
BACKUP_RETENTION_DAYS=60 ./app/scripts/backup-db.sh   # переопределить retent
```

Регулярный запуск по расписанию (crontab):

```
# Ежедневно в 03:00 по местному времени
0 3 * * * /opt/megapolis/app/scripts/backup-db.sh >> /var/log/
megapolis_backup.log 2>&1
```

Перед первым запуском убедитесь, что:

1. У пользователя, от имени которого запускается cron, есть право вызывать `docker exec` (как правило — членство в группе `docker`).
2. Каталог `/var/backups/megapolis/` существует и доступен для записи. При отсутствии скрипт его создаст.

6.3. Проверка целостности резервных копий

Рекомендуется ежемесячно проверять возможность восстановления. Для этого:

1. Скопируйте последнюю резервную копию на тестовый сервер.
2. Восстановите ее в отдельной базе данных (см. 6.4).
3. Выполните несколько SQL-запросов к таблицам `project.projects`, `monitoring.project_monitoring`, `auth.users` — убедитесь, что данные читаются корректно.
4. Удалите тестовую базу после проверки.

6.4. Процедура восстановления данных

Внимание. Восстановление приведет к потере всех данных, внесенных в базу после создания резервной копии. Перед восстановлением рекомендуется выполнить внеплановую резервную копию текущего состояния.

В состав дистрибутива Программы включен готовый скрипт `app/scripts/restore-db.sh`. Скрипт:

- запрашивает у администратора явное подтверждение строкой «`I CONFIRM RESTORE`» (либо принимает переменную окружения `FORCE_RESTORE=yes` для неинтерактивного режима в задачах автоматизации);
- пересоздает базу данных через суперпользователя `postgres` контейнера (DROP + CREATE);
- распаковывает указанный дамп и применяет его через `pg_restore --exit-on-error`;
- возвращает корректные коды выхода (0 — успех; 1 — нет предусловий; 2 — нет docker/gunzip; 3 — отказ пользователя; 4 — ошибка восстановления).

Последовательность действий администратора:

1. Остановите backend (соединения с базой помешают пересоздать ее):

```
cd /opt/megapolis
docker compose stop backend
```

2. Запустите скрипт восстановления, указав путь к дампу:

```
./app/scripts/restore-db.sh /var/backups/megapolis/
megapolis_YYYYMMDDTHHMMSSZ.dump.gz
```

3. Подтвердите действие, введя ровно `I CONFIRM RESTORE` (либо запустите неинтерактивно: `FORCE_RESTORE=yes ./app/scripts/restore-db.sh ...`).
4. Запустите backend:

```
docker compose up -d backend
```

5. Проверьте работоспособность:

```
curl http://localhost:${NGINX_HTTP_PORT:-8080}/api/auth/health
```

6. Запустите backend:

```
docker compose up -d backend
```

7. Проверьте работоспособность:

```
curl http://localhost:${NGINX_HTTP_PORT:-8080}/api/auth/health
```

6.5. Загруженные пользователями файлы

Пользовательские файлы (например, XML-файлы сметной документации при загрузке) сохраняются в именованный том Docker `asu_demo_uploads`. Резервное копирование тома:

```
docker run --rm \
    -v asu_demo_uploads:/uploads \
```

```
    -v /var/backups/megapolis:/backup \
    alpine tar -czf /backup/uploads_${TIMESTAMP}.tar.gz -C /uploads .
```

Восстановление:

```
docker run --rm \
    -v asu_demo_uploads:/uploads \
    -v /var/backups/megapolis:/backup \
    alpine tar -xzf /backup/uploads_YYYYMMDDTHHMMSSZ.tar.gz -C /uploads
```

---

7. Мониторинг работы Программы

7.1. Проверка работоспособности (healthcheck)

Каждые 30 секунд Docker автоматически проверяет состояние backend через HTTP-запрос к `/api/auth/health`. Ответ `{"status":"ok"}` означает нормальную работу.

Ручная проверка:

```
curl -s http://localhost:${NGINX_HTTP_PORT:-8080}/api/auth/health
```

Ожидаемый ответ:

```
{"status":"ok"}
```

7.2. Проверка состояния контейнеров

```
cd /opt/megapolis
docker compose ps
```

В колонке «STATUS» должны отображаться значения «Up» и «healthy» для всех трех контейнеров (`postgres`, `backend`, `nginx`).

7.3. Просмотр логов

Логи backend (основные события в формате JSON):

```
docker compose logs -f --tail=200 backend
```

Логи Nginx:

```
docker compose logs -f --tail=200 nginx
```

Логи PostgreSQL:

```
docker compose logs -f --tail=200 postgres
```

Поиск конкретного события в логах:

```
docker compose logs backend | grep '"level": "ERROR"'
```

7.4. Ежедневный снимок состояния рисков

Backend запускает планировщик (APScheduler), который в 16:55 UTC (23:55 по времени Красноярска) выполняет задачу «daily_risk_snapshot». Задача создает снимок цветовых индикаторов рисков по всем объектам на текущий день в таблице `monitoring.daily_risk_snapshots`.

Проверка выполнения:

```
SELECT
    snapshot_date,
    COUNT(*) AS "количество объектов",
    COUNT(*) FILTER (WHERE worst_risk_color = 'red') AS "красных",
    COUNT(*) FILTER (WHERE worst_risk_color IN ('yellow', 'yellow_plus')) AS
"желтых",
    COUNT(*) FILTER (WHERE worst_risk_color = 'green') AS "зеленых"
FROM monitoring.daily_risk_snapshots
WHERE snapshot_date >= CURRENT_DATE - 7
GROUP BY snapshot_date
ORDER BY snapshot_date DESC;
```

Если за последние дни снимков нет — проверьте логи backend на предмет ошибок APScheduler.

7.5. Метрики, рекомендуемые к мониторингу

Метрика	Норма	Действия при отклонении
Доступность эндпоинта `/api/auth/health`	100 %	Перезапустить backend: `docker compose restart backend`
Время отклика эндпоинта входа `/api/auth/login`	< 500 мс	Проверить нагрузку на PostgreSQL, размер таблицы `auth.sessions`
Размер таблицы `monitoring.daily_risk_snapshots`	Растет на 10 строк/день для 10 объектов	При отсутствии роста — проверить APScheduler
Количество активных сессий	Несколько десятков	При тысячах — проверить, нет ли утечки
Дисковое пространство тома `asu_demo_postgres_data`	< 80 %	Увеличить диск либо выполнить `VACUUM FULL`

7.6. Очистка устаревших данных

Программа автоматически выполняет следующие очистки при старте backend:

- устаревшие сессии (`auth.cleanup_expired_sessions(30)`) — старше 30 дней;
- устаревшие записи журнала аудита (`auth.cleanup_old_audit_logs(90)`) — старше 90 дней.

Для ручного запуска очистки:

```
SELECT auth.cleanup_expired_sessions(30);
SELECT auth.cleanup_old_audit_logs(90);
```

---

8. Обновление Программы до новых версий

8.1. Процедура планового обновления

1. **Прочитайте примечания к релизу (Release Notes)**, опубликованные правообладателем. Особое внимание — к изменениям схемы базы данных и конфигурации.
2. **Создайте внеплановую резервную копию** (см. раздел 6).
3. **Остановите стек:**

   ```
   cd /opt/megapolis
   docker compose stop
   ```

4. **Скачайте новую версию.** Правообладатель предоставляет архив с новой версией (либо Git-репозиторий с тегом соответствующей версии). Распакуйте новую версию в отдельный каталог, перенесите файл `.env` из предыдущей версии.
5. **Пересоберите образы backend:**

   ```
   docker compose build --no-cache backend
   ```

6. **Пересоберите клиентскую часть:**

   ```
   cd /opt/megapolis/frontend
   npm ci
   npm run build
   ```

7. **Запустите стек:**

   ```
   cd /opt/megapolis
   docker compose up -d
   ```

8. **Проверьте работоспособность:**
   - ответ эндпоинта `/api/auth/health`;
   - открытие пользовательского интерфейса в браузере;
   - вход администратора;

◦ выполнение типовой операции (например, загрузка списка объектов).

8.2. Миграции базы данных

При запуске нового backend автоматически выполняется модуль `migration_runner`, который:

1. Проверяет наличие таблицы `public.schema_migrations` (создает при отсутствии);
2. Сравнивает версии миграций, примененных к базе, с перечнем файлов в `database/init-scripts/`;
3. Применяет отсутствующие миграции в порядке номеров;
4. Каждая миграция выполняется в отдельной транзакции с журналированием хеша файла.

Важно. При наличии в новом релизе SQL-скриптов с номерами, превышающими номера предыдущих миграций, backend при запуске применяет их автоматически. Если миграция завершилась с ошибкой, backend не запускается (режим fail-fast) — в этом случае проверьте логи, исправьте проблему (обычно это ручное восстановление целостности данных) и перезапустите backend.

8.3. Откат (rollback) при неудачном обновлении

1. Остановите стек: `docker compose stop`.
2. Восстановите базу данных из резервной копии (см. 6.4).
3. Откатите каталог с файлами на предыдущую версию.
4. Запустите стек: `docker compose up -d`.

---

9. Управление информационной безопасностью

9.1. Защита секретов

- Файл `.env` имеет права `600`, владелец — пользователь, от имени которого запускается Docker.
- Значение `SECRET_KEY` генерируется криптостойким генератором один раз при установке. **Изменение `SECRET_KEY` после ввода Программы в эксплуатацию приведет к инвалидации всех существующих сессий и refresh-токенов.**
- `POSTGRES_PASSWORD` и `SECRET_KEY` не должны попадать в лог-файлы, скриншоты, переписку.

9.2. HTTPS и сертификаты

Рекомендуется использовать HTTPS с сертификатом, выданным доверенным удостоверяющим центром.

- Для общедоступных развертываний — Let's Encrypt (через Certbot);
- Для развертываний в закрытом корпоративном контуре — внутренний корпоративный УЦ;
- Для контура КИИ — сертификаты, сформированные сертифицированными ФСБ средствами криптозащиты.

Сертификаты монтируются в контейнер Nginx через том `./nginx/ssl:/etc/nginx/ssl:ro`. Обновление сертификатов (при использовании Let's Encrypt) — автоматическое через Certbot (настраивается администратором инфраструктуры).

9.3. Защита сети

- Фильтрация входящего трафика на межсетевом экране: допускаются порты 80 и 443, все остальные — закрыты.
- Ограничение административного доступа по SSH по IP-адресам (whitelist).
- Использование fail2ban либо аналогичных средств защиты от брут-форс атак на SSH.
- Внутренние сетевые сервисы (PostgreSQL порт 5432) не публикуются наружу.

9.4. Регулярные действия администратора

Задача	**Периодичность**
Проверка состояния контейнеров	Ежедневно
Ревизия журнала аудита действий администратора	Еженедельно
Проверка успешности резервного копирования	Ежедневно (автоматическое уведомление при ошибке)
Проверка возможности восстановления из резервной копии	Ежемесячно
Обновление операционной системы сервера	По мере выхода обновлений безопасности
Ревизия учетных записей (актуальность)	Раз в полугодие
Ротация административных паролей	Раз в полугодие
Проверка сертификатов HTTPS на срок действия	Ежемесячно
Обновление Программы до новых версий	По мере выхода релизов правообладателя

9.5. Инциденты информационной безопасности

В случае подозрения на инцидент (несанкционированный доступ, утечка данных, атака):

1. Немедленно изолируйте сервер (отключение сетевых соединений, кроме административного подключения SSH).
2. Сделайте внеплановое резервное копирование (для последующего расследования).
3. Извлеките логи (backend, Nginx, PostgreSQL, ОС) в защищенное хранилище.
4. Сообщите руководству организации заказчика.
5. При значительных инцидентах — обратитесь в соответствующие органы (ФСТЭК, ФСБ, МВД — в зависимости от характера инцидента).
6. Уведомите правообладателя по адресу petr4820@yandex.ru.

10. Типовые проблемы и устранение

10.1. Backend не стартует

Симптомы. Контейнер `asu_demo_backend` находится в состоянии «restarting» либо «exited».

Диагностика:

```
docker compose logs backend | tail -100
```

Вероятные причины и решения:

Сообщение в логе	Причина	Решение
`ValueError: CORS_ALLOW_ORIGINS must be explicitly set`	Не задана переменная `CORS_ALLOW_ORIGINS` в `.env`	Задайте явный список источников
`sqlalchemy.exc.OperationalError: connection refused`	PostgreSQL недоступен	Проверьте состояние контейнера `postgres`; убедитесь, что контейнер `postgres` стартует до `backend` (настройка `depends_on`)
`Database migration failed: ...`	Ошибка в SQL-скрипте миграции	Изучите текст ошибки; при необходимости обратитесь в техническую поддержку
`SECRET_KEY must be at least 32 characters`	Слишком короткий `SECRET_KEY`	Сгенерируйте новый длинный ключ

10.2. PostgreSQL не стартует

Симптомы. Контейнер `asu_demo_postgres` не переходит в состояние «healthy».

Вероятные причины и решения:

Сообщение	Причина	Решение
`FATAL: password authentication failed`	Несоответствие между файлом `.env` и уже установленной базой	Проверьте, что `POSTGRES_USER` и `POSTGRES_PASSWORD` соответствуют значениям, заданным при первом запуске базы; либо удалите том `asu_demo_postgres_data` для полной переинициализации
`could not resize shared memory segment`	Недостаточно оперативной памяти	Увеличьте выделенную память для Docker (минимум 2 ГБ)
`database is being accessed by other users`	Другие соединения блокируют восстановление	Остановите все использующие базу приложения перед восстановлением

10.3. Nginx возвращает 502 Bad Gateway

Причина. Backend недоступен.

Решение:

1. Проверьте состояние backend: `docker compose ps`.
2. Проверьте логи backend: `docker compose logs backend`.
3. При необходимости перезапустите: `docker compose restart backend`.

10.4. Пользователь не видит «свои» данные

Симптом. Пользователь видит пустую таблицу или не имеет доступа к модулю.

Решение:

1. Проверьте в модуле «Администрирование», что у пользователя выбраны нужные модули и установлен правильный флаг «полный доступ» / «только просмотр».
2. Попросите пользователя выйти и войти заново (обновить access-токен).
3. Проверьте логи backend на предмет ошибок авторизации.

10.5. Медленная работа таблицы объектов

Причины и решения:

Признак	Решение
10+ тысяч объектов в таблице	Использовать фильтры перед прокруткой; настроить серверную пагинацию (в следующих релизах)
Медленные SQL-запросы	Выполнить `VACUUM ANALYZE` на PostgreSQL; проверить наличие необходимых индексов
Перегруженный сервер	Увеличить ресурсы (CPU/RAM); использовать горизонтальное масштабирование

10.6. Проблемы с кодировкой при экспорте

Причина. Пользователь открывает экспортированный файл в программе, не поддерживающей UTF-8.

Решение:

- Для Excel-файлов (.xlsx) и Word-файлов (.docx) — использовать Microsoft Office 2013+, Яндекс 360, LibreOffice 6+. Более ранние версии могут отображать кириллицу с искажениями.
- Русскоязычные имена файлов корректно обрабатываются современными браузерами (Chrome, Firefox, Edge, Яндекс.Браузер). При использовании устаревших браузеров возможны проблемы — обновите браузер.

---

11. Взаимодействие с правообладателем

11.1. Регулярные каналы связи

- **Техническая поддержка:** petr4820@yandex.ru — основной канал для всех обращений.
- **Телефон:** +7 (999) 448-48-20 (рабочие дни 9:00–18:00 UTC+7).
- **Официальный сайт:** https://asumegapolis.ru.

11.2. Формат запросов в техническую поддержку

При обращении рекомендуется указывать:

1. Версию Программы (установленную в поле «О программе» либо в файле VERSION в каталоге установки);
2. Операционную систему сервера и ее версию;
3. Описание проблемы с пошаговым воспроизведением;
4. Сообщения об ошибках из логов (фрагмент релевантных строк);
5. Контактные данные для обратной связи.

11.3. Отчетность о продуктивной эксплуатации

Правообладатель может запросить раз в год обобщенные сведения о работе Программы на стороне заказчика:

- количество активных пользователей;
- количество объектов в учете;
- общий объем данных;
- отсутствие критических инцидентов.

Указанные сведения не содержат персональных данных пользователей и носят статистический характер. Предоставление сведений — добровольное.

12. Приложения

12.1. Справочная информация

- Руководство пользователя — `../user-manual/USER_MANUAL.md`;
- Инструкция по установке и настройке — `../install-manual/INSTALL_MANUAL.md`;
- Описание функциональных характеристик — `../func-description/FUNCTIONAL_DESCRIPTION.md`;
- Процессы жизненного цикла — `../lifecycle/LIFECYCLE.md`.

12.2. Нормативные ссылки

- Федеральный закон от 27.07.2006 № 152-ФЗ «О персональных данных»;
- Федеральный закон от 27.07.2006 № 149-ФЗ «Об информации, информационных технологиях и о защите информации»;
- Федеральный закон от 26.07.2017 № 187-ФЗ «О безопасности критической информационной инфраструктуры Российской Федерации»;
- Приказ ФСТЭК России от 25.12.2017 № 239 «Об утверждении Требований по обеспечению безопасности значимых объектов критической информационной инфраструктуры Российской Федерации»;
- Методические рекомендации Минцифры России по установке обновлений программного обеспечения на объектах критической информационной инфраструктуры.

Настоящее Руководство администратора является обязательным к применению при эксплуатации Программы. В случае противоречий между настоящим Руководством и инструкциями правообладателя в рамках технической поддержки следует руководствоваться более поздними указаниями правообладателя.