""

ИНСТРУКЦИЯ ПО УСТАНОВКЕ И НАСТРОЙКЕ

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. Назначение документа

Настоящая Инструкция описывает процедуру первичного развертывания программы для ЭВМ «Автоматизированная система управления строительными проектами «Мегаполис»» (далее — «Программа») на сервере заказчика. Документ предназначен для администраторов инфраструктуры (системных администраторов).

По завершении развертывания Программа будет доступна пользователям через веб-интерфейс, а администратору — через модуль «Администрирование».

Примерная продолжительность установки: 15–60 минут в зависимости от операционной системы и наличия предварительной подготовки (уже установленный Docker сокращает время вдвое).

---

2. Требования к инфраструктуре

2.1. Минимальные требования к серверу

Параметр	Значение
Архитектура процессора	x86-64 (AMD64 / Intel 64)
Количество логических ядер	2
Тактовая частота	не ниже 2,0 ГГц
Оперативная память	4 ГБ
Свободное место на SSD	40 ГБ
Сетевое соединение	100 Мбит/с

2.2. Рекомендуемые требования к серверу

Параметр	Значение
Количество логических ядер	4
Оперативная память	8 ГБ и более
Свободное место на SSD	80 ГБ и более
Сетевое соединение	1 Гбит/с
Резервное хранилище	100 ГБ на отдельном физическом носителе

2.3. Поддерживаемые операционные системы

1. Astra Linux Common Edition 1.7.6 и выше (правообладатель — ООО «РусБИТех-Астра»);
2. РЕД ОС 7.3 и выше (правообладатель — ООО «РЕД СОФТ»);
3. Ubuntu Server 22.04 LTS, 24.04 LTS;
4. Иные операционные системы семейства Linux с современным ядром (6.0+) — работоспособность не гарантируется, но обычно функционирует.

Операционные системы семейств CentOS, Red Hat Enterprise Linux, SUSE Linux Enterprise **не поддерживаются** в соответствии с требованиями Постановления Правительства Российской Федерации № 1236.

2.4. Сетевые требования

Порт	Протокол	Направление	Назначение
80	TCP	Входящее	HTTP (для редиректа на HTTPS и для ACME-валидации Let's Encrypt)
443	TCP	Входящее	HTTPS (основной порт веб-интерфейса)
22	TCP	Входящее	SSH-администрирование (с ограничением по IP)
80/443	TCP	Исходящее	Репозитории пакетов, обновления, Docker Hub, Let's Encrypt

---

3. Подготовительный этап

3.1. Проверка прав доступа

Убедитесь, что у вас есть:

- Учетная запись администратора (root или учетная запись с правами sudo);
- SSH-доступ к серверу либо физическая консоль;
- Сведения о конфигурации сети (выделенный IP-адрес, DNS-имя, параметры межсетевого экрана).

3.2. Обновление операционной системы

Перед установкой рекомендуется обновить системные пакеты.

Для Astra Linux / Ubuntu:

```
sudo apt-get update
sudo apt-get upgrade -y
sudo apt-get autoremove -y
```

Для РЕД ОС:

```
sudo dnf update -y
sudo dnf autoremove -y
```

3.3. Настройка времени и часового пояса

Корректное время критически важно для работы JWT-токенов, журналирования и резервного копирования.

```
timedatectl set-timezone Asia/Krasnoyarsk   # при необходимости укажите ваш часовой пояс
timedatectl set-ntp true
timedatectl status
```

3.4. Создание технической учетной записи

Рекомендуется не запускать Docker от имени `root`. Создайте отдельную учетную запись для развертывания.

```
sudo useradd -m -s /bin/bash megapolis
sudo passwd megapolis
sudo usermod -aG sudo megapolis     # для Ubuntu / Astra
sudo usermod -aG wheel megapolis    # для РЕД ОС
```

Переход под созданную учетную запись:

```
sudo -i -u megapolis
```

---

4. Установка Docker и Docker Compose

4.1. Astra Linux Common Edition 1.7

1. Установите зависимости:

   ```
   sudo apt-get install -y ca-certificates curl gnupg lsb-release
   ```

2. Astra Linux поставляет свой пакет Docker в официальных репозиториях («cosmic»):

   ```
   sudo apt-get install -y docker.io docker-compose-plugin
   ```

3. Добавьте пользователя `megapolis` в группу `docker`:

   ```
   sudo usermod -aG docker megapolis
   ```

4. Выйдите и войдите заново, чтобы применились права группы.

5. Проверьте установку:

   ```
   docker --version
   docker compose version
   ```

   Ожидаемые результаты: Docker 20.10+ и Docker Compose v2.x.

4.2. РЕД ОС 7.3

1. Установите Docker из официальных репозиториев РЕД ОС:

   ```
   sudo dnf install -y docker-ce docker-ce-cli containerd.io docker-compose-plugin
   ```

   Либо используйте альтернативный способ (при отсутствии пакетов в репозиториях РЕД ОС) — установку через официальный скрипт Docker:

```
curl -fsSL https://get.docker.com -o get-docker.sh
sudo sh get-docker.sh
```

2. Активируйте и запустите Docker:

```
sudo systemctl enable --now docker
sudo systemctl status docker
```

3. Добавьте пользователя `megapolis` в группу `docker`:

```
sudo usermod -aG docker megapolis
```

4. Перелогиньтесь и проверьте:

```
docker --version
docker compose version
```

4.3. Ubuntu 22.04 / 24.04 LTS

Установка официальной последней версии Docker Engine по инструкции Docker Inc.:

1. Установка зависимостей:

```
sudo apt-get install -y ca-certificates curl gnupg
```

2. Добавление GPG-ключа Docker:

```
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
sudo chmod a+r /etc/apt/keyrings/docker.gpg
```

3. Добавление репозитория:

```
echo "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/
docker.gpg] \
    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. Установка пакетов:

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

5. Активация службы и группа пользователя:

```
sudo systemctl enable --now docker
sudo usermod -aG docker megapolis
```

6. Перелогиньтесь и проверьте:

```
docker run --rm hello-world
docker compose version
```

4.4. Установка Node.js для сборки клиентской части

Клиентская часть Программы собирается с помощью инструмента Vite на Node.js. Рекомендуется Node.js версии 18 LTS или 20 LTS.

На Ubuntu / Astra:

```
curl -fsSL https://deb.nodesource.com/setup_20.x | sudo -E bash -
sudo apt-get install -y nodejs
node --version     # ожидаемо v20.x
npm --version
```

На РЕД ОС:

```
sudo dnf module install -y nodejs:20/common
node --version
npm --version
```

---

5. Получение дистрибутива Программы

Правообладатель предоставляет дистрибутив Программы в виде:

- архива `.tar.gz` с полным набором файлов (рекомендуется);
- доступа к Git-репозиторию (при договоренности).

5.1. Получение архива и распаковка

Разместите архив в каталоге `/opt/` и распакуйте:

```
sudo mkdir -p /opt/megapolis
sudo chown megapolis:megapolis /opt/megapolis
cd /opt/megapolis
tar -xzf /tmp/megapolis-1.0.0-community.tar.gz --strip-components=1
```

После распаковки в `/opt/megapolis/` должны присутствовать каталоги `backend/`, `frontend/`, `database/`, `nginx/`, `scripts/` и файлы `docker-compose.yml`, `.env.example`, `README.md`.

5.2. Получение через Git (при договоренности)

```
cd /opt/
sudo git clone <URL_РЕПОЗИТОРИЯ_ПРАВООБЛАДАТЕЛЯ> megapolis
sudo chown -R megapolis:megapolis /opt/megapolis
cd /opt/megapolis
sudo -u megapolis git checkout v1.0.0-community     # переключиться на релизный
тег
```

---

6. Настройка переменных окружения

6.1. Создание файла `.env` из шаблона

```
cd /opt/megapolis
cp .env.example .env
chmod 600 .env
```

6.2. Генерация криптографических ключей

```
# Ключ JWT и HMAC (не менее 32 байт случайных данных = 64 hex-символа)
SECRET_KEY=$(openssl rand -hex 32)

# Пароль PostgreSQL (24 hex-символа, без проблемных спецсимволов)
DB_PASSWORD=$(openssl rand -hex 24)

# Запишем значения в .env
sed -i "s|^SECRET_KEY=.*|SECRET_KEY=${SECRET_KEY}|"                .env
sed -i "s|^POSTGRES_PASSWORD=.*|POSTGRES_PASSWORD=${DB_PASSWORD}|" .env

# Проверим
grep -E '^(SECRET_KEY|POSTGRES_PASSWORD)=' .env
```

6.3. Заполнение остальных переменных

Откройте `.env` в редакторе и заполните:

- `POSTGRES_USER` — произвольное имя пользователя базы, например `megapolis_prod`.
- `POSTGRES_DB` — имя базы, например `megapolis_prod`.
- `POSTGRES_SERVER` — оставьте значение `postgres` (имя сервиса в docker-compose).
- `CORS_ALLOW_ORIGINS` — полный URL веб-интерфейса, например `https://megapolis.example.ru`. Несколько источников разделяются запятой без пробелов.
- `ENVIRONMENT=production`.
- `DEBUG=false`.

- `APP_NAME=АСУ СП Мегаполис Community` (или ваше собственное наименование развертывания).
- Сроки и лимиты (`ACCESS_TOKEN_EXPIRE_MINUTES`, `REFRESH_TOKEN_EXPIRE_DAYS`, `MAX_FAILED_LOGIN_ATTEMPTS`, `ACCOUNT_LOCK_DURATION_MINUTES`) — использовать значения по умолчанию либо согласовать с политикой информационной безопасности заказчика.
- `NGINX_HTTP_PORT=80` и `NGINX_HTTPS_PORT=443` (для продуктивной эксплуатации).
- `POSTGRES_HOST_PORT` — удалить строку либо закомментировать `#` (порт PostgreSQL наружу не публикуется).

6.4. Проверка корректности файла `.env`

```
set -a
source .env
set +a

echo "SECRET_KEY длина: ${#SECRET_KEY} (минимум 64)"
echo "POSTGRES_PASSWORD длина: ${#POSTGRES_PASSWORD}"
echo "CORS_ALLOW_ORIGINS: ${CORS_ALLOW_ORIGINS}"
```

Длина `SECRET_KEY` должна быть не меньше 64 (длина hex-представления 32 байт).

---

7. Сборка клиентской части

Клиентская часть собирается один раз перед первым запуском. При последующих обновлениях сборка повторяется.

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

Скрипт `npm ci` устанавливает точные версии библиотек, указанные в `package-lock.json`. Скрипт `npm run build` выполняет проверку типов TypeScript и сборку оптимизированной статической версии в каталог `frontend/dist/`.

Ожидаемое время: 2–5 минут в зависимости от производительности сервера.

По завершении сборки в `frontend/dist/` будут файлы `index.html`, `assets/*.js`, `assets/*.css`, и другие статические ресурсы.

Каталог `frontend/dist/` монтируется в контейнер Nginx через том `- ./frontend/dist:/usr/share/nginx/html:ro`.

8. Первый запуск и проверка работоспособности

8.1. Запуск стека

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

Команда выполнит:

1. Скачивание образов `postgres:16-alpine`, `nginx:1.27-alpine`;
2. Сборку образа backend из `./backend/Dockerfile`;
3. Создание именованных томов `asu_demo_postgres_data`, `asu_demo_uploads`;
4. Создание изолированной сети `asu_demo_network`;
5. Запуск трех контейнеров (`postgres`, `backend`, `nginx`) в указанном порядке с учетом зависимостей.

При первом запуске PostgreSQL выполнит четыре SQL-скрипта инициализации из каталога `database/init-scripts/`:

- `01_schema.sql` — полная схема базы данных (схемы `auth`, `project`, `monitoring`, все таблицы, индексы, функции, триггеры);
- `02_seed_reference.sql` — справочные данные (муниципалитеты, государственные программы, виды работ, контрольные точки);
- `03_seed_demo_user.sql` — триггеры защиты демонстрационного пользователя и сам пользователь `demo`;
- `04_seed_demo_data.sql` — демонстрационные объекты капитального строительства, сметы, Ганнт-графики.

Для продуктивной эксплуатации скрипты `03_seed_demo_user.sql` и `04_seed_demo_data.sql` рекомендуется **удалить** перед первым запуском, чтобы не создавалась учетная запись `demo` с предустановленным паролем и демонстрационные данные. Вместо этого первичная учетная запись администратора создается отдельно (см. раздел 8.3).

8.2. Проверка работоспособности

Через 1–2 минуты после запуска проверьте состояние контейнеров:

```
docker compose ps
```

Все три контейнера должны быть в статусе `Up` и `healthy`.

Проверьте базовый эндпоинт:

```
curl -s http://localhost/api/auth/health
# Ожидаемый ответ: {"status":"ok"}
```

Откройте веб-интерфейс в браузере (с рабочей станции, имеющей доступ к серверу):

`http://<IP-АДРЕС-СЕРВЕРА>/`

Или, если установлены SSL-сертификаты:

`https://<ДОМЕН-СЕРВЕРА>/`

8.3. Создание первичной учетной записи администратора (продуктивная установка)

Если в `.env` не было создано демонстрационных данных, первичный администратор создается напрямую через SQL.

1. Подключитесь к PostgreSQL:

   ```
   docker exec -it asu_demo_postgres psql -U "${POSTGRES_USER}" -d "${POSTGRES_DB}"
   ```

2. Сгенерируйте bcrypt-хеш для временного пароля (вне контейнера):

   ```
   python3 -c "import bcrypt; print(bcrypt.hashpw(b'TemporaryPassword2026!', bcrypt.gensalt(12)).decode())"
   ```

   Выведется строка вида `$2b$12$ABC...` — скопируйте ее.

3. Вставьте запись в таблицу `auth.users` (замените `<HASH>` на полученное значение):

   ```
   INSERT INTO auth.users (username, full_name, password_hash, is_active, is_admin, default_module)
   VALUES ('admin', 'Администратор системы', '<HASH>', TRUE, TRUE, 'admin');
   ```

4. Назначьте администратору доступ ко всем модулям:

   ```
   INSERT INTO auth.user_modules (user_id, module_id, can_edit)
   SELECT u.user_id, m.module_id, TRUE
   FROM auth.users u
   CROSS JOIN auth.modules m
   WHERE u.username = 'admin' AND m.is_active = TRUE;
   ```

5. Проверьте:

   ```
   SELECT u.username, u.is_admin, COUNT(um.module_id) AS modules
   FROM auth.users u
   LEFT JOIN auth.user_modules um ON um.user_id = u.user_id
   WHERE u.username = 'admin'
   GROUP BY u.username, u.is_admin;
   ```

6. Выйдите из psql (`\q`).

7. Войдите в Программу: логин `admin`, пароль `TemporaryPassword2026!`.
8. **Немедленно смените пароль** через модуль «Администрирование» → учетная запись `admin` → «Сбросить пароль».

8.4. Первый вход на демонстрационном стенде

Если при первом запуске были применены SQL-скрипты с демонстрационными данными, используйте:

- Логин: `demo`
- Пароль: `demo`

Эта учетная запись имеет флаг «Администратор системы» и доступ ко всем модулям.

Внимание. Триггер в базе данных возвращает пароль учетной записи `demo` к эталону при любых попытках его изменить (см. руководство администратора, раздел 3.6). Для продуктивной эксплуатации триггер должен быть удален.

---

9. Настройка HTTPS

Для работы с публичного интернета настоятельно рекомендуется использовать HTTPS с сертификатом доверенного удостоверяющего центра.

9.1. Получение сертификата Let's Encrypt

1. Установите Certbot:

```
# Ubuntu / Astra
sudo apt-get install -y certbot
# РЕД ОС
sudo dnf install -y certbot
```

2. Перед получением сертификата временно остановите контейнер Nginx или остановите стек:

```
cd /opt/megapolis
docker compose down
```

3. Получите сертификат (замените `megapolis.example.ru` на ваш домен):

```
sudo certbot certonly --standalone -d megapolis.example.ru \
    --agree-tos -m admin@example.ru --non-interactive
```

   Сертификаты будут размещены в `/etc/letsencrypt/live/megapolis.example.ru/`.

4. Скопируйте сертификаты в каталог Nginx:

```
sudo mkdir -p /opt/megapolis/nginx/ssl
sudo cp /etc/letsencrypt/live/megapolis.example.ru/fullchain.pem /opt/
```

```
megapolis/nginx/ssl/
        sudo cp /etc/letsencrypt/live/megapolis.example.ru/privkey.pem /opt/
megapolis/nginx/ssl/
        sudo chown megapolis:megapolis /opt/megapolis/nginx/ssl/*.pem
        sudo chmod 640 /opt/megapolis/nginx/ssl/*.pem
```

5. Откройте файл `nginx/conf.d/default.conf` и активируйте блок для HTTPS, указав пути к сертификатам.
6. Запустите стек:

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

9.2. Автоматическое обновление сертификатов

Создайте скрипт обновления `/opt/megapolis/scripts/renew-ssl.sh`:

```
        #!/bin/bash
        set -euo pipefail

        CERT_DIR="/etc/letsencrypt/live/megapolis.example.ru"
        TARGET_DIR="/opt/megapolis/nginx/ssl"

        sudo certbot renew --quiet

        if [ -f "${CERT_DIR}/fullchain.pem" ]; then
            sudo cp "${CERT_DIR}/fullchain.pem" "${TARGET_DIR}/"
            sudo cp "${CERT_DIR}/privkey.pem" "${TARGET_DIR}/"
            sudo chown megapolis:megapolis "${TARGET_DIR}"/*.pem
            sudo chmod 640 "${TARGET_DIR}"/*.pem

            # Перезагрузить Nginx без перезапуска контейнера
            docker compose -f /opt/megapolis/docker-compose.yml exec nginx nginx -s
reload
        fi
```

Сделайте скрипт исполняемым и добавьте в crontab:

```
        chmod +x /opt/megapolis/scripts/renew-ssl.sh

        (sudo crontab -l 2>/dev/null; echo "0 3 * * 1 /opt/megapolis/scripts/renew-
ssl.sh") | sudo crontab -
```

Скрипт будет запускаться каждый понедельник в 03:00.

9.3. Самоподписанные сертификаты (для закрытых контуров)

Для внутренних развертываний, не имеющих доступа к Let's Encrypt, используйте сертификат корпоративного удостоверяющего центра или самоподписанный:

```
cd /opt/megapolis/nginx/ssl

openssl req -x509 -nodes -days 3650 -newkey rsa:4096 \
    -keyout privkey.pem \
    -out fullchain.pem \
    -subj "/C=RU/ST=Krasnoyarsky/L=Krasnoyarsk/O=Megapolis/
CN=megapolis.internal"

chmod 640 privkey.pem fullchain.pem
```

Самоподписанные сертификаты будут вызывать предупреждение в браузере — пользователям понадобится однократно добавить их в доверенные.

---

10. Конфигурация межсетевого экрана

10.1. UFW (Ubuntu / Astra)

```
sudo ufw default deny incoming
sudo ufw default allow outgoing
sudo ufw allow 22/tcp                  # SSH
sudo ufw allow 80/tcp                  # HTTP
sudo ufw allow 443/tcp                 # HTTPS
sudo ufw --force enable
sudo ufw status
```

Рекомендуется ограничить SSH-доступ IP-адресами администраторов:

```
sudo ufw delete allow 22/tcp
sudo ufw allow from <IP_АДМИНИСТРАТОРА> to any port 22
```

10.2. firewalld (РЕД ОС)

```
sudo systemctl enable --now firewalld
sudo firewall-cmd --permanent --add-service=ssh
sudo firewall-cmd --permanent --add-service=http
sudo firewall-cmd --permanent --add-service=https
sudo firewall-cmd --reload
sudo firewall-cmd --list-all
```

10.3. Защита от брут-форс атак на SSH

Установите fail2ban:

```
# Ubuntu / Astra
sudo apt-get install -y fail2ban
# РЕД ОС
sudo dnf install -y fail2ban

sudo systemctl enable --now fail2ban
```

Настройка параметров по умолчанию достаточна для базовой защиты. При необходимости — см. документацию fail2ban.

---

11. Настройка резервного копирования

В состав дистрибутива Программы уже включены production-готовые скрипты `app/scripts/backup-db.sh` (создание резервной копии) и `app/scripts/restore-db.sh` (восстановление из копии). Подробное описание возможностей — в руководстве администратора (`../admin-manual/ADMIN_MANUAL.md`, разделы 6.2 и 6.4).

Краткая последовательность настройки автоматического резервного копирования:

1. Убедитесь, что скрипты исполняемы (после распаковки дистрибутива права обычно сохраняются, но команду можно повторить без последствий):

   ```
   chmod +x /opt/megapolis/app/scripts/backup-db.sh /opt/megapolis/app/scripts/restore-db.sh
   ```

2. Запустите ручную резервную копию для проверки работоспособности:

   ```
   /opt/megapolis/app/scripts/backup-db.sh
   ls -la /var/backups/megapolis/
   ```

   В случае успеха в каталоге `/var/backups/megapolis/` появится файл вида `megapolis_YYYYMMDDTHHMMSSZ.dump.gz` с правами `0600`.

3. Создайте регулярную задачу `cron` (выполняется ежедневно в 03:00 локального времени):

   ```
   (crontab -l 2>/dev/null; echo "0 3 * * * /opt/megapolis/app/scripts/backup-db.sh >> /var/log/megapolis_backup.log 2>&1") | crontab -
   ```

4. Через сутки проверьте, что задача выполнилась корректно (по логу `/var/log/megapolis_backup.log` и появлению нового файла в каталоге резервных копий).

5. Ежемесячно — проверяйте возможность восстановления из резервной копии в тестовой среде (см. ADMIN_MANUAL § 6.3).

12. Контрольный список установки

По завершении установки проверьте каждый пункт:

- ☐ Сервер обновлен, часовой пояс настроен, NTP работает
- ☐ Создан непривилегированный пользователь для Docker (`megapolis`)
- ☐ Docker и Docker Compose установлены, в группе `docker` состоит пользователь `megapolis`
- ☐ Node.js 18 LTS или 20 LTS установлен
- ☐ Дистрибутив Программы распакован в `/opt/megapolis/`
- ☐ Файл `.env` создан, заполнен; длина `SECRET_KEY` ≥ 64 символов; `CORS_ALLOW_ORIGINS` указывает на реальный URL
- ☐ Клиентская часть собрана (`frontend/dist/` заполнен)
- ☐ Стек запущен: `docker compose ps` показывает все три контейнера в `Up (healthy)`
- ☐ Ответ `curl http://localhost/api/auth/health` — `{"status":"ok"}`
- ☐ Веб-интерфейс открывается в браузере
- ☐ Для продуктивной установки: демонстрационные данные (03_seed_demo_user.sql, 04_seed_demo_data.sql) удалены; создана первичная учетная запись администратора с сильным паролем
- ☐ Получен и настроен SSL-сертификат (или принято решение использовать самоподписанный/HTTP в закрытом контуре)
- ☐ Межсетевой экран настроен (открыты только порты 22, 80, 443)
- ☐ SSH-доступ ограничен IP-адресами администраторов, работает fail2ban
- ☐ Скрипт резервного копирования создан и запланирован в cron
- ☐ Первая ручная резервная копия успешна
- ☐ Установлен механизм мониторинга (внешний Uptime-сервис либо локальный скрипт с уведомлениями)

13. Приложение: устранение неполадок при установке

13.1. Ошибка «permission denied» при запуске Docker

Причина. Пользователь не входит в группу `docker`.

Решение:

```
sudo usermod -aG docker $(whoami)
# Выйдите из сессии и войдите заново
```

13.2. Ошибка «Cannot connect to the Docker daemon»

Причина. Служба Docker не запущена.

Решение:

```
sudo systemctl start docker
sudo systemctl enable docker
```

13.3. Контейнер PostgreSQL не запускается: «data directory ... has wrong ownership»

Причина. Том `asu_demo_postgres_data` имеет неверные права.

Решение: удалите том и создайте заново:

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

Внимание. Эта операция удаляет все данные! Применять только при первичной установке.

13.4. Backend не может подключиться к PostgreSQL

Причина. Несоответствие переменных окружения.

Решение:

1. Убедитесь, что в `.env` значения `POSTGRES_USER`, `POSTGRES_PASSWORD`, `POSTGRES_DB` совпадают с фактическими настройками базы.
2. Если база была создана с другими значениями — удалите том и переинициализируйте базу (см. 13.3).

13.5. Nginx возвращает 502 Bad Gateway при первом обращении

Причина. Backend еще не прогрелся (первый запуск, миграции).

Решение: подождите 30–60 секунд и обновите страницу. Если через 2 минуты все еще 502 — проверьте логи backend:

```
docker compose logs backend | tail -50
```

13.6. Ошибка при сборке клиентской части: «npm ERR! code ERESOLVE»

Причина. Конфликт версий зависимостей.

Решение:

```
cd /opt/megapolis/frontend
rm -rf node_modules package-lock.json
npm install
```

13.7. «certbot: SSL handshake failed» при получении сертификата Let's Encrypt

Причины и решения:

- Домен не резолвится в публичный IP сервера — проверить DNS.
- Порт 80 закрыт межсетевым экраном или занят другим приложением — проверить.
- DNS-провайдер использует CAA-записи, блокирующие Let's Encrypt — настроить CAA для Let's Encrypt либо использовать другой УЦ.

13.8. Программа работает, но пользователь не может войти

Причины и решения:

- `CORS_ALLOW_ORIGINS` не включает URL, с которого обращается пользователь — добавить корректный URL в `.env` и перезапустить backend.
- Используется HTTP вместо HTTPS, но cookie `refresh_token` установлен с флагом `Secure` — включить HTTPS либо временно отключить флаг `Secure` в `modules/auth/api/auth.py` (не рекомендуется для production).

Завершение установки

По завершении всех шагов Программа готова к эксплуатации. Дальнейшие действия — в руководстве пользователя и руководстве администратора.

Контакты правообладателя для консультаций по установке:

- Электронная почта: petr4820@yandex.ru
- Телефон: +7 (999) 448-48-20 (рабочие дни 9:00–18:00 UTC+7)

Просьба сообщать правообладателю об успешном развертывании для учета инсталляций и возможности уведомить о выходе обновлений безопасности.