РУКОВОДСТВО АДМИНИСТРАТОРА
ПО «Т-ПЛАТФОРМА»

Версия документа № 1

Правообладатель:
ООО «ИНСАЙТТЕХСОЛЮШИНС»
Генеральный директор:
Пашков С. А. _____

ВВЕДЕНИЕ

Настоящий документ содержит сведения, необходимые системному администратору для установки, запуска, эксплуатации, обслуживания, обновления и восстановления экземпляра программного обеспечения «Т-Платформа».

ПО «Т-Платформа» представляет собой веб-платформу для размещения, каталогизации, приобретения и потребления цифрового образовательного и медийного контента. Платформа включает пользовательский веб-интерфейс, административную панель, серверную часть, базу данных, кэш, интеграции с внешними сервисами оплаты, видеохостинга и электронной почты. Документ предназначен для системных администраторов, технических специалистов и специалистов сопровождения, выполняющих эксплуатацию экземпляра программного обеспечения в серверной среде.

В 1, 2, 3, 4, 5 и 6 пунктах описаны процессы необходимые для эксплуатации программного обеспечения для администратора

Документ состоит из 28 страниц и 40 иллюстраций.

НАЗНАЧЕНИЕ ДОКУМЕНТА

Настоящее руководство описывает:

- требования к серверной инфраструктуре;
- порядок подготовки окружения;
- порядок установки и запуска программного обеспечения;
- порядок остановки и перезапуска системы;
- состав основных компонентов системы;
- порядок настройки конфигурации;
- порядок резервного копирования и восстановления;
- порядок обновления программного обеспечения;
- порядок мониторинга состояния системы;
- действия при возникновении типовых ошибок;
- требования к безопасности и техническому обслуживанию.

Настоящий документ не является руководством конечного пользователя. Описание пользовательских сценариев, работы с каталогом, материалами, событиями и административной панелью приведено в документе «Руководство пользователя ПО Т-Платформа».

ТЕРМИНЫ И ОПРЕДЕЛЕНИЯ

ПО — программное обеспечение «Т-Платформа».

Экземпляр ПО — установленная и запущенная копия программного обеспечения на сервере или в облачной инфраструктуре.

Системный администратор — специалист, ответственный за установку, настройку, сопровождение, обновление и восстановление программного обеспечения.

Администратор платформы — пользователь с расширенными правами доступа в административной панели ПО.

Backend — серверная часть программного обеспечения, реализующая бизнес-логику и API.

Frontend — клиентская часть программного обеспечения, отображаемая пользователю в браузере.

База данных — хранилище данных платформы, включая пользователей, материалы, заказы, платежи и настройки.

Docker — средство контейнеризации, используемое для запуска компонентов ПО.

Docker Compose — инструмент для запуска нескольких связанных контейнеров.

Контейнер — изолированная среда выполнения отдельного компонента системы.

Резервная копия — копия данных, используемая для восстановления работоспособности системы при сбое.

1. ОБЩИЕ СВЕДЕНИЯ О ПРОГРАММНОМ ОБЕСПЕЧЕНИИ

ПО «Т-Платформа» предназначено для автоматизации процессов публикации, продажи, предоставления доступа и потребления цифрового контента.

Основные функциональные возможности платформы:

- размещение образовательных и медийных материалов;
- каталогизация цифрового контента;
- поиск и фильтрация материалов;
- проведение и публикация онлайн-событий;
- управление авторами;
- управление пользователями;
- управление категориями контента;
- обработка платежей;
- предоставление доступа к приобретённым материалам;
- защита цифрового контента;
- интеграция с внешними сервисами.

Платформа является веб-приложением и используется через браузер.

2. СОСТАВ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Экземпляр ПО «Т-Платформа» включает следующие основные компоненты:

1. **Frontend-приложение**

   Отвечает за отображение пользовательского интерфейса и административной панели в браузере.

2. **Backend-приложение**

   Отвечает за обработку запросов, бизнес-логику, авторизацию, работу с пользователями, материалами, заказами и интеграциями.

3. **База данных PostgreSQL**

   Используется для хранения данных платформы.

4. **Redis**

   Используется для кэширования и вспомогательных операций.

5. **Файловое хранилище / внешнее хранилище контента**

   Используется для хранения или предоставления доступа к цифровому контенту.

6. **Интеграции с внешними сервисами**

   Включают платёжные сервисы, видеохостинг, SMTP-серверы и иные внешние системы.

7. **Docker-конфигурация**

   Используется для контейнеризированного запуска компонентов системы.

3. ТРЕБОВАНИЯ К КВАЛИФИКАЦИИ АДМИНИСТРАТОРА

Для эксплуатации ПО «Т-Платформа» системный администратор должен обладать следующими навыками:

- базовые навыки администрирования Linux-серверов;
- знание командной строки Linux;
- понимание принципов работы веб-приложений;
- базовые знания Docker и Docker Compose;
- понимание принципов работы HTTP/HTTPS;
- базовые навыки работы с доменами и DNS;
- базовые навыки настройки SSL-сертификатов;
- понимание принципов резервного копирования;
- навыки анализа логов и диагностики ошибок.

4. ТРЕБОВАНИЯ К СЕРВЕРНОЙ СРЕДЕ

Для эксплуатации экземпляра ПО рекомендуется использовать сервер под управлением Linux.

Минимальные требования к серверу:

- операционная система: Linux;
- процессор: от 2 ядер;
- оперативная память: от 4 ГБ;
- дисковое пространство: от 40 ГБ;
- доступ в сеть Интернет;
- открытые сетевые порты для HTTP/HTTPS;
- установленный Docker;
- установленный Docker Compose.

Рекомендуемые требования:

- операционная система: Linux;
- процессор: от 4 ядер;
- оперативная память: от 8 ГБ;
- дисковое пространство: от 80 ГБ;
- доступ в сеть Интернет;
- открытые сетевые порты для HTTP/HTTPS;
- установленный Docker;
- установленный Docker Compose.

5. ТРЕБОВАНИЯ К ПРОГРАММНОМУ ОКРУЖЕНИЮ

Для работы экземпляра ПО необходимо наличие следующих компонентов:

- Linux-сервер;
- Docker;
- Docker Compose;
- PostgreSQL;
- Redis;
- веб-сервер (Nginx);
- доменное имя;
- SSL-сертификат;
- SMTP-сервер для отправки email-уведомлений;
- доступ к платёжным сервисам при использовании оплаты;
- доступ к сервису хранения или воспроизведения видео при использовании видеоконтента.

В случае контейнеризированного развёртывания PostgreSQL и Redis могут запускаться как отдельные Docker-контейнеры в составе общего проекта.

6. ПОДГОТОВКА СЕРВЕРА К УСТАНОВКЕ

Перед установкой ПО системному администратору необходимо выполнить следующие действия:

1. Подготовить сервер (VDS/VPS) под управлением Linux.
2. Обновить системные пакеты.
3. Установить Docker.
4. Установить Docker Compose.
5. Настроить доменное имя.
6. Настроить DNS-записи домена.
7. Настроить SSL-сертификат.
8. Проверить доступность необходимых портов.
9. Подготовить конфигурационные файлы приложения.
10. Получить необходимые ключи и параметры внешних сервисов.

Как установить актуальные версии Docker и Docker Compose

Перед началом установки новых приложений обновите списки пакетов в системе:

```
$ sudo apt update
```

Затем установите необходимые для дальнейшей работы зависимости:

```
$ sudo apt install ca-certificates gnupg curl
```

Данная команда устанавливает следующие компоненты:

- `ca-certificates` – набор корневых сертификатов доверенных центров сертификации, необходимый системам и приложениям для проверки подлинности HTTPS-соединений;
- `gnupg` – инструмент для работы с криптографическими ключами и подписями, используемый для проверки цифровой подписи репозитория и его пакетов;

- `curl` – утилита командной строки, используемая для загрузки данных по URL.

Установка Docker

Следующая команда устанавливает GPG-ключ, который будет использоваться пакетным менеджером `apt` для проверки подлинности пакетов из репозитория Docker. В команде используются различные ссылки на загрузку GPG-ключа. Для запуска команды на Ubuntu используйте запись:

```
$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /usr/share/keyrings/docker.gpg
```

В Debian команда выглядит следующим образом:

```
$ curl -fsSL https://download.docker.com/linux/debian/gpg | sudo gpg --dearmor -o /usr/share/keyrings/docker.gpg
```

Затем добавьте в систему APT-репозиторий Docker. В Ubuntu для этого запустите команду:

```
$ echo "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

В Debian команда выглядит так:

```
$ echo "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/docker.gpg] https://download.docker.com/linux/debian $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

Для того, чтобы пакетный менеджер `apt` узнал о новых пакетах и версиях, доступных из только что добавленного источника, обновите список пакетов через загрузку index-файлов с каждого подключённого репозитория:

```
$ sudo apt update
```

После чего установите Docker Engine с клиентом командной строки, базовым контейнерным движком `containerd` и плагином расширенной сборки образов Buildx:

```
$ sudo apt install docker-ce docker-ce-cli containerd.io docker-buildx-plugin
```

По окончании установки проверьте версию клиента Docker:

```
$ docker --version
```

В нашем примере версия Docker действительно является актуальной на момент написания статьи:

```
your-user@debian-server:~$ docker --version
Docker version 28.3.3, build 980b856
```

Установка Docker Compose

Клиент командной строки Docker CLI позволяет управлять Docker Engine и его контейнерами. Кроме того, он поддерживает расширения в виде плагинов. При запуске какого-либо расширения командой вида `docker <plugin>` Docker CLI ищет соответствующий исполняемый файл в определённых каталогах. Приоритет поиска в данном случае следующий:

- сначала – локальные пути пользователя:
  - `$DOCKER_CLI_PLUGIN_PATH` – если установлена переменная окружения, Docker в первую очередь проверяет именно её;
  - `~/.docker/cli-plugins/` – каталог плагинов для конкретного пользователя;
- затем – системные пути:
  - `/usr/local/lib/docker/cli-plugins/` – основной путь для плагинов, установленных вручную;
  - `/usr/lib/docker/cli-plugins/` – путь, куда плагины обычно ставятся пакетным менеджером.

Исходя из этого, логичным выглядит использование универсального решения. То есть создайте директорию, которая будет доступна для всех пользователей сервера и в которой будут храниться дополнительные плагины для Docker CLI, в том числе и Docker Compose:

```
$ sudo mkdir -p /usr/local/lib/docker/cli-plugins/
```

В данный каталог загрузите из GitHub бинарный файл последней версии Docker Compose:

```
$ sudo curl -SL https://github.com/docker/compose/releases/latest/download/docker-compose-$(uname -s)-$(uname -m) -o /usr/local/lib/docker/cli-plugins/docker-compose
```

Затем следующей командой сделайте загруженный файл исполняемым:

```
$ sudo chmod +x /usr/local/lib/docker/cli-plugins/docker-compose
```

После чего проверьте версию Docker Compose:

```
$ docker compose version
```

Сравните выведенный номер версии с номером релиза на GitHub. Это позволит убедиться в том, что у вас в системе установлен Docker Compose актуальной версии.

```
your-user@debian-server:~$ docker compose version
Docker Compose version v2.39.2
```

7. КОНФИГУРАЦИОННЫЕ ПАРАМЕТРЫ

Перед запуском ПО необходимо настроить переменные окружения и конфигурационные параметры.

К основным параметрам относятся:

- адрес базы данных;
- имя базы данных;
- пользователь базы данных;
- пароль базы данных;
- адрес Redis;
- секретный ключ приложения;
- параметры JWT-авторизации;
- настройки домена;
- настройки CORS;
- параметры SMTP-сервера;
- параметры платёжного шлюза;
- параметры интеграции с видео хостингом;
- параметры хранения файлов;
- режим запуска приложения.

Конфигурационные данные должны храниться в защищённом виде. **Не рекомендуется** размещать пароли, секретные ключи и токены доступа в публичных репозиториях или открытых каталогах. Конфигурационный файл хранится по пути:

```
/t-platoform/backend/config.py
```

8. УСТАНОВКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Установка ПО выполняется путём размещения файлов программного обеспечения на сервере и запуска контейнеров.

Общий порядок установки:

- Подключиться к серверу по SSH.
- Перейти в рабочий каталог.
- Разместить файлы программного обеспечения.
- Проверить наличие конфигурационных файлов.
- Проверить наличие файла docker-compose.yml.
- Указать необходимые переменные окружения в config.py.
- Выполнить сборку и запуск контейнеров.

Размещение файлов на сервере происходит путем перемещения их с локальной машины на сервер по протоколу SFTP. Для улучшения опыта разработки рекомендуется настроить CICD пайплайн.

9. ЗАПУСК И ОСТАНОВКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Запуск ПО осуществляется командой:

```
$ docker compose up –build -d
```

При запуске выполняется:

- запуск контейнера backend-приложения;
- запуск контейнера frontend-приложения;
- запуск контейнера базы данных;
- запуск контейнера Redis;
- запуск иных сервисов, предусмотренных конфигурацией;
- подключение компонентов к общей сети Docker.

После запуска необходимо проверить состояние контейнеров:

```
$ docker compose ps
```

Все основные сервисы должны находиться в состоянии `running` или аналогичном активном состоянии.

Остановка экземпляра ПО выполняется командой:

```
$ docker compose down
```

При выполнении команды останавливаются контейнеры, относящиеся к проекту.

Если необходимо остановить контейнеры без удаления связанных ресурсов, администратор должен учитывать параметры конкретной Docker Compose-конфигурации.

10. ПЕРЕЗАПУСК ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Для перезапуска всех сервисов используется команда:

```
$ docker compose restart
```

Для перезапуска отдельного сервиса используется команда:

```
$ docker compose restart <service_name>
```

Например:

```
$ docker compose restart frontend
```

или

```
$ docker compose restart frontend
```

Имена сервисов определяются в файле `docker-compose.yml`.

11. ПРОВЕРКА ДОСТУПНОСТИ СИСТЕМЫ

После запуска системный администратор должен проверить доступность основных интерфейсов.

Основные адреса:

Интерфейс	Назначение
`https://domain.com`	пользовательский интерфейс
`https://domain.com/admin`	административная панель
`https://domain.com/api`	API backend-приложения

Проверка пользовательского интерфейса выполняется через браузер.

Проверка API может выполняться через браузер, curl или иные инструменты диагностики:

```
$ curl –I https://domain.com
```

При корректной работе сервер должен возвращать успешный HTTP-ответ.

12. АДМИНИСТРАТИВНАЯ ПАНЕЛЬ

Административная панель доступна пользователям с соответствующими правами доступа.

Обычно административная панель располагается по адресу:

```
https://domain.com/admin
```

Через административную панель выполняются следующие действия:

- управление пользователями;
- управление авторами;
- управление образовательными программами и материалами;
- управление категориями;
- управление событиями;
- просмотр платежей и транзакций;
- настройка отдельных параметров платформы;
- проверка опубликованного контента.

Создание, изменение и удаление контента через административную панель относится к прикладному администрированию платформы и подробно описывается в руководстве пользователя.

Настоящее руководство описывает **техническую эксплуатацию экземпляра ПО**.

13. УПРАВЛЕНИЕ ПОЛЬЗОВАТЕЛЯМИ И РОЛЯМИ

В системе предусмотрено разграничение прав доступа.

Основные роли:

Роль	**Назначение**
Пользователь	просмотр, приобретение и потребление контента
Администратор платформы	управление пользователями, контентом, авторами, категориями, событиями и транзакциями
Системный администратор	техническое сопровождение экземпляра ПО на сервере

Системный администратор должен обеспечивать:

- защиту учётных записей администраторов;
- выдачу административных прав только уполномоченным лицам;
- своевременное отключение неиспользуемых учётных записей;
- контроль безопасности паролей;
- проверку корректности настроек доступа.

14. РЕЗЕРВНОЕ КОПИРОВАНИЕ

Для обеспечения сохранности данных необходимо регулярно выполнять резервное копирование.

Резервному копированию подлежат:

- база данных PostgreSQL;
- пользовательские файлы;
- загруженный контент;
- конфигурационные файлы;
- файл .env;
- файл docker-compose.yml;
- SSL-сертификаты при необходимости;
- иные данные, необходимые для восстановления экземпляра ПО.

Рекомендуемая периодичность резервного копирования:

- база данных — не реже одного раза в сутки;
- конфигурационные файлы — после каждого изменения;
- пользовательские файлы и контент — по расписанию, определяемому объёмом изменений;
- полный архив экземпляра — перед обновлением ПО.

Пример создания резервной копии базы данных PostgreSQL:

```
$ docker exec –t postgres pg_dump –U postgres t-platform > backup.sql
```

Если контейнер базы данных имеет другое имя, необходимо использовать фактическое имя контейнера.

Проверить список контейнеров можно командой:

```
$ docker ps
```

15. ВОССТАНОВЛЕНИЕ ИЗ РЕЗЕРВНОЙ КОПИИ

Восстановление системы может потребоваться при:

- сбое базы данных;
- повреждении данных;
- ошибочном удалении данных;
- сбое сервера;
- некорректном обновлении;
- переносе экземпляра ПО на новый сервер.

Общий порядок восстановления:

- Остановить программное обеспечение.
- Подготовить серверную среду.
- Восстановить конфигурационные файлы.
- Запустить базу данных.
- Восстановить базу данных из резервной копии.
- Восстановить пользовательские файлы и контент.
- Запустить все сервисы.
- Проверить доступность пользовательского интерфейса.
- Проверить доступность административной панели.
- Проверить корректность основных функций.

Пример восстановления базы данных:

```
$ cat backup.sql | docker exec -i postgres psql -U postgres -d tplatform
```

Перед восстановлением рекомендуется создать дополнительную копию текущего состояния данных, если это возможно.

16. ОБНОВЛЕНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ И ТИПОВЫЕ ОШИБКИ

Обновление программного обеспечения «Т-Платформа» выполняется системным администратором или иным уполномоченным техническим специалистом, ответственным за эксплуатацию экземпляра ПО.

Обновление может включать:

- замену файлов программного обеспечения;
- обновление backend-приложения;
- обновление frontend-приложения;
- обновление конфигурационных файлов;
- обновление структуры базы данных;
- обновление Docker-образов;
- перезапуск сервисов;
- проверку работоспособности пользовательского интерфейса и административной панели.

Перед началом обновления рекомендуется выполнить резервное копирование базы данных, пользовательских файлов и конфигурационных файлов. Это необходимо для возможности восстановления работоспособности системы в случае ошибки обновления.

16.1. Подготовка к обновлению

Перед обновлением системный администратор должен:

- Уведомить ответственных лиц о планируемых технических работах.
- Убедиться в наличии доступа к серверу.
- Проверить текущее состояние контейнеров.

- Проверить наличие свободного места на диске.
- Создать резервную копию базы данных.
- Создать резервную копию конфигурационных файлов.
- Сохранить текущую версию программного обеспечения.
- Проверить наличие новой версии файлов ПО.

Проверка состояния контейнеров выполняется командой:

```
docker compose ps
```

Проверка свободного места на диске:

```
df -h
```

Создание резервной копии базы данных PostgreSQL:

```
docker exec -t postgres pg_dump -U postgres tplatform > backup.sql
```

Если контейнер базы данных, пользователь базы данных или имя базы данных отличаются, необходимо использовать фактические значения, указанные в конфигурации экземпляра ПО.

16.2. Порядок обновления программного обеспечения

Обновление программного обеспечения выполняется в следующем порядке:

1. Перейти в рабочий каталог программного обеспечения:

```
cd /opt/t-platforma
```

2. Получить новую версию файлов программного обеспечения.

   Если обновление выполняется из репозитория:

```
git pull
```

Если обновление выполняется вручную, необходимо заменить файлы приложения на актуальную версию, сохранив конфигурационные файлы и пользовательские данные.

3. Выполнить сборку и запуск контейнеров:

```
docker compose up --build -d
```

4. Проверить состояние контейнеров:

```
docker compose ps
```

5. Проверить логи сервисов:

```
docker compose logs -f
```

6. Проверить доступность пользовательского интерфейса:

```
https://yourdomain.com
```

7. Проверить доступность административной панели:

```
https://yourdomain.com/admin
```

8. Проверить доступность API:

```
https://yourdomain.com/api
```

После обновления системный администратор должен убедиться, что основные функции платформы работают корректно: авторизация, просмотр контента, доступ к административной панели, работа с материалами, пользователями, платежами и внешними интеграциями.

16.3. Перезапуск программного обеспечения после обновления

Если после обновления требуется выполнить перезапуск всех сервисов, используется команда:

```
docker compose restart
```

Для перезапуска отдельного сервиса используется команда:

```
docker compose restart service_name
```

Например:

```
docker compose restart backend
```

или:

```
docker compose restart frontend
```

Имена сервисов определяются в файле `docker-compose.yml`.

16.4. Откат обновления

Если после обновления программное обеспечение работает некорректно, системный администратор должен выполнить откат к предыдущему рабочему состоянию.

Общий порядок отката:

1. Остановить программное обеспечение:

```
docker compose down
```

2. Вернуть предыдущую версию файлов программного обеспечения.
3. Восстановить конфигурационные файлы при необходимости.
4. Восстановить базу данных из резервной копии при необходимости:

```
cat backup.sql | docker exec -i postgres psql -U postgres -d tplatform
```

5. Запустить программное обеспечение:

```
docker compose up -d
```

6. Проверить состояние контейнеров:

```
docker compose ps
```

7. Проверить доступность пользовательского интерфейса и административной панели.

Перед восстановлением базы данных рекомендуется сохранить текущее состояние, если это возможно.

16.5. Типовые ошибки и способы их устранения

В процессе эксплуатации и обновления программного обеспечения могут возникать типовые ошибки. Ниже приведены наиболее распространённые ситуации и рекомендуемые действия системного администратора.

16.5.1. Сайт недоступен

Описание ошибки:

Пользовательский интерфейс не открывается в браузере.

Возможные причины:

- не запущены контейнеры;
- frontend-сервис завершился с ошибкой;
- backend-сервис недоступен;
- ошибка DNS-записей;
- истёк или некорректно настроен SSL-сертификат;
- закрыты сетевые порты;
- ошибка reverse proxy или веб-сервера.

Действия администратора:

Проверить состояние контейнеров:

```
docker compose ps
```

Проверить логи сервисов:

```
docker compose logs -f
```

Проверить доступность сайта:

```
curl -I https://yourdomain.com
```

Проверить DNS-записи домена и настройки SSL-сертификата.

16.5.2. Backend-приложение не запускается

Описание ошибки:

Контейнер backend находится в состоянии ошибки или постоянно перезапускается.

Возможные причины:

- неверные переменные окружения;
- отсутствует конфигурационный файл;
- недоступна база данных;
- недоступен Redis;
- ошибка в коде приложения;
- ошибка при сборке Docker-образа.

Действия администратора:

Проверить состояние контейнеров:

```
docker compose ps
```

Просмотреть логи backend-сервиса:

```
docker compose logs -f backend
```

Проверить наличие и корректность файла `.env`.

Проверить доступность базы данных и Redis.

16.5.3. Frontend-приложение не отображается корректно

Описание ошибки:

Страница открывается частично, отображается пустой экран или интерфейс работает некорректно.

Возможные причины:

- ошибка сборки frontend-приложения;
- неверный адрес backend API;
- кэш браузера содержит старую версию файлов;
- frontend-контейнер завершился с ошибкой;
- некорректные переменные окружения.

Действия администратора:

Проверить состояние frontend-сервиса:

```
docker compose ps
```

Просмотреть логи frontend-сервиса:

```
docker compose logs -f frontend
```

Проверить настройки адреса API.

Пересобрать контейнеры:

```
docker compose up --build -d
```

Очистить кэш браузера и повторить проверку.

16.5.4. Ошибка подключения к базе данных

Описание ошибки:

Приложение не может подключиться к PostgreSQL.

Возможные причины:

- контейнер PostgreSQL не запущен;
- неверный адрес базы данных;
- неверное имя базы данных;
- неверный пользователь или пароль;
- база данных недоступна по сети Docker;
- повреждение данных.

Действия администратора:

Проверить состояние контейнера базы данных:

```
docker compose ps
```

Просмотреть логи PostgreSQL:

```
docker compose logs -f postgres
```

Проверить переменные окружения:

```
POSTGRES_DB
POSTGRES_USER
POSTGRES_PASSWORD
POSTGRES_HOST
POSTGRES_PORT
```

При необходимости перезапустить контейнер базы данных:

```
docker compose restart postgres
```

16.5.5. Ошибка подключения к Redis

Описание ошибки:

Приложение сообщает об ошибке подключения к Redis или часть функций работает нестабильно.

Возможные причины:

- контейнер Redis не запущен;
- неверный адрес Redis;
- неверный порт Redis;

- ошибка сетевого взаимодействия между контейнерами.

Действия администратора:

Проверить состояние контейнера Redis:

```
docker compose ps
```

Просмотреть логи Redis:

```
docker compose logs -f redis
```

Перезапустить Redis:

```
docker compose restart redis
```

Проверить переменные окружения, отвечающие за подключение к Redis.

16.5.6. Не работает авторизация пользователей

Описание ошибки:

Пользователь не может войти в систему или после входа сразу выходит из аккаунта.

Возможные причины:

- ошибка backend-сервиса;
- неверные настройки JWT;
- неверный секретный ключ;
- ошибка CORS;
- некорректные настройки домена;
- проблема с cookies;
- ошибка подключения к базе данных.

Действия администратора:

Проверить логи backend:

```
docker compose logs -f backend
```

Проверить настройки домена и CORS.

Проверить переменные окружения, связанные с авторизацией:

```
JWT_SECRET
JWT_EXPIRES_IN
APP_DOMAIN
CORS_ORIGINS
```

Проверить доступность базы данных.

16.5.7. Не отправляются email-уведомления

Описание ошибки:

Пользователь не получает письма для восстановления пароля, регистрации или иных уведомлений.

Возможные причины:

- неверные SMTP-настройки;
- SMTP-сервер недоступен;
- неверный логин или пароль SMTP;
- почтовый сервис блокирует отправку;
- ошибка backend-сервиса;
- письма попадают в спам.

Действия администратора:

Проверить SMTP-параметры в конфигурации:

```
SMTP_HOST
SMTP_PORT
SMTP_USER
SMTP_PASSWORD
```

Проверить логи backend:

```
docker compose logs -f backend
```

Проверить доступность SMTP-сервера.

Проверить папку «Спам» у получателя.

16.5.8. Не проходят платежи

Описание ошибки:

Пользователь не может оплатить материал, платёж не создаётся или не подтверждается.

Возможные причины:

- неверные параметры платёжного провайдера;
- неверный секретный ключ;
- некорректный webhook-адрес;
- недоступен платёжный сервис;
- отсутствует HTTPS;
- ошибка backend-сервиса;
- ошибка обработки статуса платежа.

Действия администратора:

Проверить параметры платёжной интеграции.

Проверить логи backend:

```
docker compose logs -f backend
```

Проверить доступность webhook-адреса.

Проверить наличие действующего SSL-сертификата.

Проверить настройки платёжного сервиса в личном кабинете провайдера.