

Развертывание и внедрение ПО Quans Store

Полная версия: инструкция по развертыванию,
жизненный цикл и техническая поддержка

Программное обеспечение	Quans Store (CMS для интернет-магазина)
Версия	1.0
Код документа	VP-QS-01.1 (сводная редакция)
Редакция	1.0 от 04.09.2026

Область документа

Документ описывает порядок развёртывания самостоятельного экземпляра ПО Quans Store на Linux-сервере с использованием Docker Compose, а также процессы, обеспечивающие поддержание жизненного цикла ПО: подготовку сервера и окружения, установку и первичную настройку, подключение внешних интеграций, проверку работоспособности, резервное копирование, обновление и откат, регистрацию и устранение неисправностей, приоритеты и порядок технической поддержки.

Развертывание выполняется на серверной инфраструктуре, расположенной на территории Российской Федерации. Все данные (включая базу данных, файлы приложения, логи и резервные копии) хранятся исключительно на серверах, находящихся в РФ. Техническая поддержка осуществляется на территории Российской Федерации специалистами ООО «КИБЕРИЯ».

Состав документа

Часть	Источник	Редакция источника
Часть I. Инструкция по развертыванию	VP-QS-01 «Инструкция по развертыванию ПО Quans Store»	1.0 от 25.08.2026
Часть II. Жизненный цикл и техническая поддержка	VP-QS-05 «Жизненный цикл и техническая поддержка ПО Quans Store»	1.0 от 19.08.2026

Содержание частей соответствует исходным документам без изменений. Сквозная нумерация страниц и общее содержание приведены для удобства работы со сводной редакцией. Требования к серверному окружению приведены в разделе 2 части I, состав установочного комплекта — в разделе 3 части I.

Содержание

ЧАСТЬ I. Инструкция по развертыванию (VP-QS-01)	3
Область документа	3
1. Назначение инструкции	3
2. Системные требования	3
3. Состав установочного комплекта	4
4. Подготовка операционной системы	5
5. Размещение файлов и каталогов	6
6. Подготовка переменных окружения (backend, frontend, ключи интеграций)	6
7. Ролевая модель доступа и права пользователей	7
8. Аналитика и журналирование действий	9
9. Сертификат доверия для внешних API	10
10. Сборка контейнерных образов	10
11. Инициализация базы данных	11
12. Создание администратора	12
13. Запуск сервисов	12
14. Настройка Nginx и TLS	12
15. Проверка после установки	14
16. Настройка резервного копирования	15
17. Обновление и откат	16
18. Удаление экземпляра	16
19. Типовые неисправности установки	17
ЧАСТЬ II. Жизненный цикл и техническая поддержка (VP-QS-05)	18
Область документа	18
1. Назначение	18
2. Модель жизненного цикла	18
3. Управление требованиями и изменениями	18
4. Обеспечение качества	19
5. Управление выпуском	19
6. Регистрация и устранение неисправностей	20
7. Приоритеты поддержки	20
8. Мониторинг и эксплуатационный контроль	21
9. Резервное копирование и восстановление	21
10. Управление информационной безопасностью	22
11. Совершенствование ПО	22
12. Персонал сопровождения	22
13. Документация и конфигурационный учёт	23
Классификация ПО. Правообладатель	24

Инструкция по развертыванию ПО Quans Store

Программное обеспечение	Quans Store (CMS для интернет-магазина)
Версия	1.0
Код документа	VP-QS-01
Редакция	1.0 от 25.08.2026

Область документа

Настоящий документ описывает порядок развёртывания самостоятельного экземпляра ПО Quans Store на Linux-сервере с использованием Docker Compose. ПО состоит из двух частей:

- **Backend** – API на базе Laravel (PHP), работающее с PostgreSQL, Redis, Meilisearch и очередями;
- **Frontend** – клиентская часть на Nuxt.js (Vue 3), проксируемая через Nginx.

Инструкция охватывает первоначальную установку, настройку окружения, запуск контейнеров, выдачу TLS-сертификатов, базовую проверку работоспособности, настройку резервного копирования, обновление и откат, а также типовые неисправности.

1. Назначение инструкции

Инструкция устанавливает порядок развертывания двух взаимосвязанных сервисов (backend и frontend) на одном Linux-сервере с использованием Docker Compose.

Команды рассчитаны на администратора, имеющего права `sudo` или доступ к `root`, а также доступ к DNS-записям для двух доменов (или поддоменов).

Развертывание выполняется на серверной инфраструктуре, расположенной на территории Российской Федерации. Все данные (включая базу данных, файлы приложения, логи и резервные копии) хранятся исключительно на серверах, находящихся в РФ.

2. Системные требования

Ниже приведены минимальные и рекомендуемые параметры оборудования. Для production-среды рекомендуется использовать ресурсы не ниже рекомендуемых.

Параметр	Минимальное значение	Рекомендуемое значение
Архитектура	x86-64	x86-64
Операционная система	Ubuntu 22.04.5 LTS и выше (24.04, 26.04)	Ubuntu 22.04.5 LTS и выше
Виртуализация	QEMU/KVM (возможна bare-metal)	QEMU/KVM
vCPU	2	4
RAM	4 GiB	8 GiB
Дисковое пространство	30 GB	50+ GB
Контейнерная среда	Docker Engine 29.7.1, Docker Compose v5.4.0 (плагин)	Docker Engine 29.7.1, Docker Compose v5.4.0
Прокси-сервер	Nginx 1.18 (устанавливается отдельно)	Nginx 1.18
TLS	Let's Encrypt (certbot) для автоматического выпуска	Let's Encrypt или сертификат российского УЦ
Сервер БД	PostgreSQL 15 (в контейнере)	PostgreSQL 15
Кеш / очереди	Redis 7 (в контейнере)	Redis 7
Поиск	Meilisearch (в контейнере)	Meilisearch
Входящие соединения	80/tcp, 443/tcp (HTTPS), SSH (22) только для администратора	80/tcp, 443/tcp, SSH (22)
Исходящие соединения	HTTPS к внешним API (СДЭК, ЮКасса, Яндекс, DaData и др.)	HTTPS к внешним API

3. Состав установочного комплекта

В комплект поставки входят два ZIP-архива:

- **quans-store-backend.zip** – исходный код backend (Laravel).

Содержит: composer.json, composer.lock, artisan, Makefile, README.md, package.json, vite.config.js, phpunit.xml, .env.example; docker-compose.deploy.yml (сервисы: fpm, pgsql, redis, meilisearch); docker-compose.yml (локальная разработка); deploy/build/fpm/Dockerfile, entryptpoint.sh, конфигурации PHP и внутреннего Nginx; стандартные папки Laravel.

- **quans-store-frontend.zip** – исходный код frontend (Nuxt.js).

Содержит: nuxt.config.ts, package.json, package-lock.json, eslint.config.js, sentry.client.config.ts, sentry.server.config.ts, .env.example, tsconfig.json; docker-compose.deploy.yml (сервис node); deploy/build/node/Dockerfile; папки app/, public/, server/, shared/.

Примечание: системные unit-файлы для автозапуска и скрипты резервного копирования в комплект не входят; они создаются администратором самостоятельно (см. раздел 16)

4. Подготовка операционной системы

1. Установить актуальные обновления безопасности ОС:

```
sudo apt update && sudo apt upgrade -y
```

2. Установить Docker Engine, Docker Compose плагин, Nginx, Certbot и утилиты:

```
sudo apt install -y ca-certificates curl
sudo install -m 0755 -d /etc/apt/keyrings
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o
/etc/apt/keyrings/docker.asc
sudo chmod a+r /etc/apt/keyrings/docker.asc
echo "deb [arch=$(dpkg --print-architecture) signed-
by=/etc/apt/keyrings/docker.asc] https://download.docker.com/linux/ubuntu $(.
/etc/os-release && echo "$VERSION_CODENAME") stable" | sudo tee
/etc/apt/sources.list.d/docker.list > /dev/null
sudo apt update
sudo apt install -y docker-ce docker-ce-cli containerd.io docker-buildx-
plugin docker-compose-plugin
sudo apt install -y nginx certbot python3-certbot-nginx unzip
```

Примечание: возможна установка Docker из локального зеркала или из репозитория дистрибутива ОС вместо официального сайта download.docker.com. При необходимости скорректируйте команды в соответствии с используемым источником.

3. Убедиться, что Docker и Nginx запущены и добавлены в автозагрузку:

```
sudo systemctl enable --now docker
sudo systemctl enable --now nginx
```

4. Установить часовой пояс Europe/Moscow и проверить синхронизацию времени:

```
sudo timedatectl set-timezone Europe/Moscow
timedatectl status
```

5. Создать две DNS-записи (или поддомена), указывающие на IP-адрес сервера:

- `api.<ваш-домен>` — для backend;
- `<ваш-домен>` или `app.<ваш-домен>` — для frontend.

6. Разрешить входящие соединения на порты 80 и 443 для всех, а SSH-доступ ограничить доверенными сетями. Порты контейнеров (3000, 8000, 5432, 6379, 7700) не должны быть опубликованы наружу; Nginx выступает обратным прокси.

5. Размещение файлов и каталогов

Создать директории для размещения кода:

```
sudo mkdir -p /var/www/html/quans-store-backend
sudo mkdir -p /var/www/html/quans-store-frontend
```

Распаковать архивы в соответствующие каталоги:

```
sudo unzip quans-store-backend.zip -d /var/www/html/quans-store-backend
sudo unzip quans-store-frontend.zip -d /var/www/html/quans-store-frontend
```

Установить владельца и права (для безопасности):

```
sudo chown -R $USER:$USER /var/www/html/quans-store-backend
sudo chown -R $USER:$USER /var/www/html/quans-store-frontend
```

6. Подготовка переменных окружения

Для каждой части приложения необходимо создать файл `.env` на основе `.env.example` с правами 600.

6.1. Backend (Laravel)

```
cd /var/www/html/quans-store-backend
sudo cp .env.example .env
sudo chmod 600 .env
```

Обязательные переменные (секретные значения генерируются для конкретного экземпляра):

Переменная	Назначение / пример без секрета
APP_NAME	Название магазина, например "Quans Store"
APP_ENV	production (или development для теста)
APP_KEY	Уникальный ключ Laravel (генерируется командой <code>php artisan key:generate</code> внутри контейнера FPM, см. раздел 9)
APP_DEBUG	false для production
APP_URL	<a href="https://api.<ваш-домен>">https://api.<ваш-домен>
APP_FRONTEND_URL	<a href="https://<ваш-домен-front>">https://<ваш-домен-front>
DB_CONNECTION	pgsql
DB_HOST	pgsql (имя сервиса в docker-compose)

DB_DATABASE	Имя базы данных
DB_USERNAME	Пользователь БД
DB_PASSWORD	Пароль БД
REDIS_HOST	redis
MAIL_*	Настройки SMTP для отправки писем
AWS_*	Настройки S3-хранилища (если используется)
MEILISEARCH_HOST	http://meilisearch:7700
MEILISEARCH_KEY	Ключ доступа к Meilisearch

6.2. Frontend (Nuxt.js)

```
cd /var/www/html/quans-store-frontend
sudo cp .env.example .env
sudo chmod 600 .env
```

Переменная	Назначение / пример без секрета
API_URL	<a href="https://api.<ваш-домен>/api">https://api.<ваш-домен>/api
PORT	Внутренний порт (обычно 3000)
ENVIRONMENT	production
YANDEX_API_KEY	Обязательный. Ключ для Яндекс.Карт (без него сборка не выполнится). Получить в кабинете разработчика Яндекса.
DADATA_API_KEY	Ключ для DaData (если используется)
SENTRY_DSN	(опционально)

7. Ролевая модель доступа и права пользователей

В системе используются два независимых контура доступа: административная панель для сотрудников и API магазина для покупателей.

7.1. Роли сотрудников

Роль	Доступ и основные операции
Администратор	Полный доступ к административным разделам, сотрудникам, настройкам, справочникам, журналу аудита, товарам, заказам и клиентам. Может создавать сотрудников и назначать им любую роль.
Контент-менеджер	Управление страницами, блоками страниц, настройками

	контента, публикациями, FAQ, footer/social-контентом, каталогом и товарами в разрешённых разделах: просмотр, создание, редактирование и удаление. Нет доступа к административным настройкам, dashboard, loyalty-разделам, размерам упаковки и товарным группам.
Мерчендайзер	Управление товарным ассортиментом и его атрибутами: товары, категории, бренды, теги, цвета, размеры, модели и товарные группы. Для товаров разрешены просмотр, создание, редактирование, удаление, импорт, экспорт и генерация feed; редактирование остатков доступно администратору и мерчендайзеру. Нет доступа к клиентам, dashboard, loyalty-разделам, URL redirect и административным настройкам
Оператор колл-центра	Доступ к клиентам и заказам, а также просмотр товаров. Импорт и удаление товаров запрещены. Фактически policy также разрешает создание, редактирование и удаление клиентов и изменение новых заказов

7.2. Назначение ролей и административная панель

Сотрудники входят в Filament-панель по адресу /admin через сессионную аутентификацию. Роль хранится в профиле сотрудника и выбирается администратором при создании или редактировании учетной записи. Управление сотрудниками доступно только администратору. Первоначальный администратор создается командой `php artisan employees:create-admin`.

Авторизация выполняется политиками Laravel для отдельных ресурсов и операций. Политики проверяют роль сотрудника и отдельно ограничивают операции просмотра, создания, редактирования, удаления, импорта, экспорта и другие специальные действия. Наличие учетной записи само по себе даёт технический вход в панель, но доступ к разделам ограничивается политиками.

7.3. Покупатель и API

Покупатель – отдельный тип пользователя API. Для API используются токены Sanctum. Поддерживаются анонимный покупатель, регистрация и вход по коду подтверждения, smart-login, обновление токена и выход из системы.

Профиль, адреса, получатели, заказы, checkout и операции программы лояльности доступны только аутентифицированному неанонимному покупателю. Анонимный покупатель может пользоваться открытой частью магазина, но при обращении к защищенным customer-разделам получает отказ в доступе.

7.4. Общие ограничения операций

- Заказы нельзя создавать или удалять из административной панели; изменение разрешено только для заказа со статусом new и только ролям администратора и оператора колл-центра.
- Журнал аудита доступен для просмотра только администратору; изменять или удалять записи аудита нельзя.
- Для ряда справочников удаление запрещено всем ролям, включая администратора.

Каждый административный ресурс подключается к отдельной policy. Базовые методы общей policy по умолчанию разрешают операции, однако для используемых ресурсов доступ ограничивается соответствующими policy.

8. Аналитика и журналирование действий

В процессе эксплуатации системы для диагностики и контроля собираются логи различных компонентов. Ниже приведено описание источников логов и способов доступа к ним.

8.1. Логи контейнеров (stdout)

Все контейнеры выводят логи в стандартный поток вывода (stdout). Просмотр логов выполняется командой:

```
docker logs <container_name>
```

Для просмотра в реальном времени добавьте флаг -f:

```
docker logs -f <container_name>
```

Основные контейнеры:

- quans-store-backend-fpm-1 – логи PHP-приложения (Laravel);
- quans-store-backend-pgsql-1 – логи PostgreSQL;
- quans-store-backend-redis-1 – логи Redis;
- quans-store-backend-meilisearch-1 – логи Meilisearch;
- quans-store-frontend-node-1 – логи фронтенд-приложения (Nuxt).

8.2. Логи Laravel (storage/logs)

Логи приложения сохраняются в каталоге storage/logs внутри контейнера. Этот каталог монтируется в хост-систему в рамках тома Docker. По умолчанию путь на хосте:

```
/var/lib/docker/volumes/.../data/storage/logs
```

Для просмотра можно использовать команды внутри контейнера:

```
docker exec -it quans-store-backend-fpm-1 tail -f /app/storage/logs/laravel.log
```

8.3. Логи Nginx

Nginx, установленный на хосте, сохраняет логи доступа и ошибок в стандартные каталоги:

- /var/log/nginx/access.log – логи запросов;
- /var/log/nginx/error.log – логи ошибок.

Просмотр:

```
sudo tail -f /var/log/nginx/access.log
sudo tail -f /var/log/nginx/error.log
```

8.4. Логи PostgreSQL

По умолчанию логи PostgreSQL выводятся в stdout контейнера, поэтому они доступны через docker logs. Для более детального логирования можно включить запись в файл внутри контейнера, но это требует дополнительной настройки.

Рекомендуется использовать docker logs для базовой диагностики.

8.5. Sentry (опционально)

Если в .env указан SENTRY_DSN, ошибки приложения автоматически отправляются в **self-hosted** инсталляцию Sentry, развернутую на российском сервере. Для просмотра ошибок необходимо войти в Sentry-дашборд, доступ к которому предоставляется администратором системы.

8.6. Рекомендации по мониторингу

- Регулярно проверяйте логи контейнеров на наличие критических ошибок.
- Настройте ротацию логов (например, через logrotate) для предотвращения переполнения диска.
- Используйте централизованные системы сбора логов (ELK, Graylog, Datadog) для production-сред.

9. Сертификат доверия для внешних API

Дополнительные корневые сертификаты не требуются, так как все внешние интеграции используют публичные HTTPS-сертификаты. Однако в условиях РФ могут возникать проблемы с TLS при обращении к некоторым банковским API – в этом случае необходимо установить соответствующий корневой сертификат Минцифры в систему и/или в контейнеры.

Если требуется установка, разместите сертификат в системе и смонтируйте его в контейнеры при необходимости (данная инструкция не описывает этот процесс, так как он зависит от конкретного сертификата).

10. Сборка контейнерных образов

10.1. Backend

```
cd /var/www/html/quans-store-backend
sudo docker compose -f docker-compose.deploy.yml build
```

Сборка создаст образы для сервисов: fpm (PHP-FPM), pgsql, redis, meilisearch. Рекомендуется пометить образ тегом с версией.

10.2. Frontend

```
cd /var/www/html/quans-store-frontend
sudo docker compose -f docker-compose.deploy.yml build
```

Сборка соберёт статические файлы приложения и образ для Node.js.

11. Инициализация базы данных

1. Запустить контейнеры в фоновом режиме (пока только для выполнения команд):

```
cd /var/www/html/quans-store-backend
sudo docker compose -f docker-compose.deploy.yml up -d
```

2. Сгенерировать ключ приложения APP_KEY внутри контейнера FPM:

```
sudo docker compose -f docker-compose.deploy.yml run --rm fpm php artisan
key:generate
```

Эта команда запишет сгенерированный ключ в файл `.env` автоматически.

3. Выполнить миграции и сиды внутри контейнера fpm:

```
sudo docker compose -f docker-compose.deploy.yml run --rm fpm php artisan
migrate:fresh --force
sudo docker compose -f docker-compose.deploy.yml run --rm fpm php artisan
db:seed
```

4. Очистить кеш конфигурации и маршрутов (обязательный шаг, чтобы избежать проблем с закэшированными данными из предыдущих конфигураций):

```
sudo docker compose -f docker-compose.deploy.yml run --rm fpm php artisan
optimize:clear
sudo docker compose -f docker-compose.deploy.yml run --rm fpm php artisan
config:cache
sudo docker compose -f docker-compose.deploy.yml run --rm fpm php artisan
route:cache
```

12. Создание администратора

Для создания учётной записи администратора выполните команду:

```
cd /var/www/html/quans-store-backend
sudo docker compose -f docker-compose.deploy.yml run --rm fpm php artisan
employees:create-admin
```

Следуйте интерактивным подсказкам: введите email, пароль и другие запрашиваемые данные. После завершения администратор будет создан и готов к входу в панель управления.

13. Запуск сервисов

13.1. Backend

```
cd /var/www/html/quans-store-backend
sudo docker compose -f docker-compose.deploy.yml up -d
sudo docker compose -f docker-compose.deploy.yml ps
```

Должны быть запущены контейнеры: fpm, pgsql, redis, meilisearch.

13.2. Frontend

```
cd /var/www/html/quans-store-frontend
sudo docker compose -f docker-compose.deploy.yml up -d
sudo docker compose -f docker-compose.deploy.yml ps
```

14. Настройка Nginx и TLS

14.1. Создание конфигураций для HTTP (порт 80)

Создать два конфигурационных файла Nginx – для frontend и backend – в /etc/nginx/sites-available/. На этом этапе конфигурации содержат только HTTP-блоки (без SSL), чтобы Certbot мог позже их модифицировать.

Для **backend (API)**: /etc/nginx/sites-available/api.yourdomain

```
server {
    server_name api.<ваш-домен>;
    client_max_body_size 30M;

    location / {
        proxy_pass http://127.0.0.1:8000;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto https;
    }

    location ~ /\. {
        deny all;
    }

    add_header X-Content-Type-Options nosniff;
    add_header X-Frame-Options SAMEORIGIN;
    add_header Referrer-Policy strict-origin-when-cross-origin;

    listen 80;
}
```

Для frontend: /etc/nginx/sites-available/app.yourdomain

```
server {
    server_name <ваш-домен-front>;
    client_max_body_size 30M;

    location / {
        proxy_pass http://127.0.0.1:3000;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto https;
    }

    location ~ /\. {
        deny all;
    }

    add_header X-Content-Type-Options nosniff;
    add_header X-Frame-Options SAMEORIGIN;
    add_header Referrer-Policy strict-origin-when-cross-origin;

    listen 80;
}
```

14.2. Активация конфигов и проверка

```
sudo ln -s /etc/nginx/sites-available/api.yourdomain /etc/nginx/sites-enabled/  
sudo ln -s /etc/nginx/sites-available/app.yourdomain /etc/nginx/sites-enabled/  
sudo nginx -t  
sudo systemctl reload nginx
```

14.3. Выдача TLS-сертификатов (Let's Encrypt)

Теперь, когда оба домена доступны по HTTP, запустите Certbot для автоматического получения и установки сертификатов:

```
sudo certbot --nginx
```

Следуйте инструкциям: введите email, согласитесь с условиями, выберите оба домена из списка. Certbot автоматически дополнит конфиги SSL-блоками, пропишет пути к сертификатам и настроит переадресацию с HTTP на HTTPS.

Проверка автоматического обновления:

```
sudo systemctl status certbot.timer # должен быть активен
```

Примечание: при необходимости возможно использование сертификатов российских удостоверяющих центров (например, УЦ Минцифры) вместо Let's Encrypt. В этом случае потребуется ручная настройка Nginx и размещение сертификатов.

15. Проверка после установки

```
cd /var/www/html/quans-store-backend  
sudo docker compose -f docker-compose.deploy.yml ps  
sudo docker compose -f docker-compose.deploy.yml run --rm fpm php artisan  
migrate:status  
  
cd /var/www/html/quans-store-frontend  
sudo docker compose -f docker-compose.deploy.yml ps  
  
# Проверка доступности по HTTPS  
curl -I https://<ваш-домен-front>/  
curl -I https://api.<ваш-домен>/admin/login  
  
# Проверка недоступности скрытых файлов  
curl -I https://api.<ваш-домен>/.env # должен вернуть 403 или 404
```

Ожидаемые результаты:

- оба контейнера (backend и frontend) имеют статус Up;
- страница входа в админку отвечает по HTTPS;
- команда migrate:status показывает все миграции как выполненные;

- скрытый файл .env недоступен;
- в логах контейнеров отсутствуют повторяющиеся исключения.

16. Настройка резервного копирования

Администратор должен самостоятельно создать скрипт резервного копирования и настроить его выполнение по расписанию (cron). Рекомендации:

- **Что бэкапить:** дампы базы данных PostgreSQL (через docker exec и pg_dump); файл .env (для восстановления конфигурации).
- **Способ:** выполнение команд через docker exec внутри контейнера pgsql.
- **Хранение:** загружать бэкапы в **российское S3-совместимое хранилище** (например, Яндекс Object Storage, Selectel, VK Cloud). Для загрузки используйте утилиту mc (MinIO Client) или s3cmd, настроенную на соответствующее хранилище.
- **Расписание:** ежедневно в 05:00 по МСК.
- **Срок хранения:** не менее 14 дней.

Примерный шаблон скрипта (адаптировать под свои параметры):

```
#!/bin/bash
# backup-quans.sh

BACKUP_DIR="/tmp/backup-quans"
mkdir -p "$BACKUP_DIR"
DATE=$(date +%Y%m%d_%H%M%S)

DB_NAME="<имя_БД>"
DB_USER="<пользователь>"
DB_PASS="<пароль>"

docker exec -i quans-store-backend-pgsql-1 pg_dump -U $DB_USER $DB_NAME >
"$BACKUP_DIR/quans_db_$DATE.sql"
gzip "$BACKUP_DIR/quans_db_$DATE.sql"

# Загрузка в российское S3-хранилище (пример с mc)
mc cp "$BACKUP_DIR/quans_db_$DATE.sql.gz" myminio/quans-backups/

# Удаление локальных копий старше 14 дней
find "$BACKUP_DIR" -name "*.sql.gz" -mtime +14 -delete
```

Настройка cron:

```
sudo crontab -e
# Добавить строку:
0 5 * * * /usr/local/bin/backup-quans.sh
```

17. Обновление и откат

Обновления предоставляются правообладателем (ООО «Киберия») в рамках договора технической поддержки в виде архивов с исходным кодом.

17.1. Обновление

1. Создать резервную копию базы данных и файлов (см. раздел 16) и проверить ее целостность.
2. Получить новый код (через CI/CD или распаковать новый архив).
3. Собрать новые образы с неизменяемым тегом (например, quans-backend:2.0):

```
cd /var/www/html/quans-store-backend
sudo docker compose -f docker-compose.deploy.yml build --no-cache
```

4. Выполнить миграции:

```
sudo docker compose -f docker-compose.deploy.yml run --rm fpm php artisan
migrate --force
```

5. Перезапустить Compose с новым образом:

```
sudo docker compose -f docker-compose.deploy.yml up -d
```

6. Выполнить smoke-тесты (проверка доступности, логи, см. раздел 15).

17.2. Откат

- Если новая версия работает некорректно, вернуть предыдущий тег образа и перезапустить контейнеры.
- Если миграции необратимо изменили данные, восстановить базу данных из резервной копии, созданной перед обновлением, согласно утвержденному плану отката.

18. Удаление экземпляра

Остановка контейнеров не удаляет данные. Перед окончательным удалением требуется:

1. Определить срок хранения обращений и других данных в соответствии с внутренним регламентом.
2. Сохранить или уничтожить резервные копии.
3. Отозвать ключи/токены доступа к внешним сервисам (если такие были настроены).
4. Остановить и удалить контейнеры:

```
cd /var/www/html/quans-store-backend
sudo docker compose -f docker-compose.deploy.yml down -v

cd /var/www/html/quans-store-frontend
sudo docker compose -f docker-compose.deploy.yml down -v
```

- Удалить каталоги проекта и данные томов только после письменного подтверждения владельца данных.

19. Типовые неисправности установки

Симптом	Проверка и действие
Контейнер не стартует (статус Exited)	Проверить логи контейнера: <code>docker logs <container_name></code> . Проверить наличие контейнера с БД (pgsql) и его состояние.
Ошибка подключения к БД (SQLSTATE[08006])	Проверить, запущен ли контейнер pgsql, правильность переменных DB_HOST, DB_DATABASE, DB_USERNAME, DB_PASSWORD в .env. Убедиться, что сеть Docker позволяет соединение между контейнерами.
Веб-интерфейс возвращает 502 Bad Gateway	Проверить, что контейнеры fpm и node запущены (<code>docker compose ps</code>). Убедиться, что Nginx проксирует на 127.0.0.1:8000 и 127.0.0.1:3000 соответственно, а также что эти порты опубликованы на localhost (в docker-compose должно быть <code>ports: - "127.0.0.1:8000:8000"</code> и аналогично для 3000). Проверить логи Nginx и контейнеров.
Ошибка сессии / циклическая переадресация при входе	Проверить APP_URL, X-Forwarded-Proto в конфиге Nginx (должен быть https), а также настройки SESSION_DRIVER и SESSION_SECURE_COOKIE (должен быть true для HTTPS). Убедиться, что каталог storage/ имеет правильные права (в контейнере).
Миграции не выполняются (ошибка прав)	Проверить, что файл database.sqlite (если используется SQLite) или подключение к PostgreSQL настроены корректно. Для PostgreSQL убедиться, что пользователь имеет права на создание схемы. Запускать миграции с флагом <code>--force</code> .
Ошибки при обращении к внешним API (СДЭК, ЮKassa)	Проверить корректность ключей и URL в .env. Убедиться, что сервер имеет доступ к интернету и исходящие HTTPS-соединения разрешены. Проверить логи контейнера fpm на наличие ошибок HTTP-запросов.
Проблемы с кешем или очередями (Redis)	Проверить, запущен ли контейнер redis. Выполнить <code>docker exec -it redis-контейнер redis-cli ping</code> – должен отвечать PONG. Проверить переменные REDIS_HOST,

Жизненный цикл и техническая поддержка ПО Quans Store

Программное обеспечение	Quans Store (CMS для интернет-магазина)
Версия	1.0
Код документа	VP-QS-05
Редакция	1.0 от 19.08.2026

Область документа

Программная система Quans Store (далее – ПО) рассматривается как самостоятельный модуль интернет-магазина, включающий backend-часть (API на Laravel) и frontend-часть (клиентское приложение на Nuxt.js). Иные программные модули, которые могут находиться в общей кодовой базе или инфраструктуре, в настоящем документе не описываются.

1. Назначение

Документ описывает процессы, обеспечивающие поддержание жизненного цикла ПО версии 1.0: планирование изменений, разработку, тестирование, выпуск, установку, эксплуатацию, устранение неисправностей, резервное копирование и совершенствование. Организационные роли могут совмещаться одним квалифицированным специалистом, если сохранены контроль и проверка изменений.

2. Модель жизненного цикла

Этап	Основной результат
Управление требованиями	Зафиксированная задача, критерии приемки и оценка влияния
Проектирование	Изменения модели данных, интерфейсов и интеграций
Разработка	Версионизируемый исходный текст и миграции
Проверка	Тесты, review и smoke-test
Выпуск	Неизменяемый тег образа, описание версии и план отката

Этап	Основной результат
------	--------------------

Развертывание	Миграции, запуск, health check и приемочная проверка
Эксплуатация	Мониторинг, журналы, резервные копии и управление доступом
Сопровождение	Исправления, обновления зависимостей и актуальная документация
Вывод из эксплуатации	Остановка, отзыв секретов и обработка данных по регламенту

3. Управление требованиями и изменениями

1. Зарегистрировать запрос на изменение с описанием цели, инициатора и ожидаемого результата.
2. Определить, затрагивает ли изменение пользовательский сценарий, данные, интеграции с внешними API (СДЭК, ЮKassa Яндекс, DaData и др.), безопасность или процедуру установки.
3. Сформулировать проверяемые критерии приемки и сценарий отката.
4. Реализовать изменение в отдельной ветви либо изолированном рабочем состоянии Git.
5. Провести code review и автоматические тесты; для изменения интерфейса выполнить ручную проверку.
6. Присвоить версии релиза неизменяемый тег образа и зафиксировать Git-коммит.
7. Обновить функциональную, установочную и эксплуатационную документацию одновременно с выпуском.

Совместимость данных. Любое изменение схемы оформляется миграцией. Перед применением миграций на рабочем экземпляре создаётся проверенная резервная копия PostgreSQL и определяется возможность отката.

4. Обеспечение качества

- модульные тесты бизнес-логики (корзина, заказы, оплата, бонусы);
- функциональные тесты API-эндпоинтов (авторизация, каталог, оформление заказа);
- проверка идемпотентности платежных операций и обработки webhook-уведомлений;
- проверка прав доступа и отображения административного интерфейса Filament;
- статический контроль отсутствия отладочных вызовов в прикладном коде;
- ручной сквозной smoke-test (регистрация → выбор товара → оформление → оплата);
- проверка миграций на копии базы данных и контроль целостности;
- проверка документации на соответствие фактическому поведению.

Примечание: текущая контрольная база включает более 100 автоматических тестов. Рост числа тестов не заменяет сквозную проверку с реальными интеграциями (СДЭК, ЮKassa, Яндекс.Карты и др.).

5. Управление выпуском

Артефакт релиза	Требование
Исходный текст	Git-коммит и архив поставки
Зависимости	composer.lock и package-lock.json без самопроизвольного обновления
Контейнерный образ	Уникальный тег; дополнительно фиксируется image ID
Миграции	Порядок применения и оценка обратимости
Описание версии	Новые функции, исправления, ограничения и известные проблемы
План отката	Предыдущий образ и совместимая резервная копия
Документация	Актуальные руководства и эксплуатационные инструкции

6. Регистрация и устранение неисправностей

6.1. Содержание обращения в поддержку

- дата и время с указанием временной зоны;
- адрес экземпляра и версия образа;
- роль пользователя и выполнявшееся действие;
- номер заказа или идентификатор сессии без избыточного раскрытия персональных данных;
- ожидаемый и фактический результат;
- текст ошибки, HTTP-код или идентификатор события;
- воспроизводимость и влияние на пользователей;
- фрагмент журнала с удалёнными токенами и персональными данными.

6.2. Процесс исправления

1. Зарегистрировать неисправность и присвоить приоритет.
2. Стабилизировать сервис: ограничить проблемный сценарий, перезапустить отдельный процесс либо переключиться на предыдущий образ.
3. Собрать диагностические сведения из журналов контейнеров (fpm, node, postgres, redis, horizon), Nginx и Sentry (**используется self-hosted инсталляция Sentry, развернутая на российском сервере**).
4. Воспроизвести ошибку в изолированном окружении или на копии обезличенных данных.
5. Разработать исправление и регрессионный тест.
6. Выполнить review, полный набор тестов и проверку критического сценария.
7. Развернуть исправление по процедуре выпуска и проконтролировать журналы.
8. Закрыть неисправность с описанием причины, изменения и профилактической меры.

7. Приоритеты поддержки

Техническая поддержка осуществляется на территории Российской Федерации

специалистами ООО «КИБЕРИЯ».

Приоритет	Критерий	Цель первичной реакции
P1 – критический	Недоступен весь экземпляр, потеря данных, массовая невозможность оформления заказов или проведения оплат	До 4 рабочих часов
P2 – высокий	Не работает значимая функция без приемлемого обходного пути (каталог, корзина, личный кабинет)	До 1 рабочего дня
P3 – обычный	Частичная ошибка с обходным путем или единичный сбой	До 3 рабочих дней
P4 – консультация	Вопрос, пожелание или улучшение без отказа функции	До 5 рабочих дней

Указанные интервалы являются обязательными для организации сопровождения и должны быть закреплены договором или внутренним регламентом. Срок устранения определяется после диагностики и не равен времени первичной реакции.

8. Мониторинг и эксплуатационный контроль

- состояние и health check контейнеров (fpm, node, pgsql, redis, meilisearch);
- доступность фронтенда и админки (/admin/ login) по HTTPS;
- работоспособность очередей (Laravel Horizon) и планировщика задач;
- статус индексации Meilisearch (Scout);
- ошибки в логах приложения, ошибки подключения к БД и внешним API;
- записи в таблице failed_jobs и статус обработки уведомлений;
- свободное место в /var/lib/ docker и каталогах проекта;
- выполнение cron-заданий (резервное копирование, очистка кеша);
- целостность PostgreSQL и наличие актуальной внешней копии резервной копии;
- мониторинг через Sentry (ошибки и производительность) – используется self-hosted инсталляция на российском сервере.

9. Резервное копирование и восстановление

Штатная копия создается ежедневно в 05:00 по МСК (см. раздел 14 Инструкции по установке). PostgreSQL дампируется через pg_dump, архивируется и загружается в **руссийское S3-совместимое хранилище** (например, Selectel Object Storage, Яндекс Object Storage или VK Cloud). Файлы .env и другие конфигурационные данные сохраняются отдельно.

Процедура восстановления:

1. Остановить контейнеры fpm и node либо обеспечить согласованное окно восстановления.

2. Сохранить текущую поврежденную базу и журналы для анализа.
3. Проверить целостность выбранной резервной копии (контрольные суммы).
4. Восстановить базу данных из дампа через psql.
5. При необходимости восстановить файлы из S3 (медиа, изображения товаров).
6. Запустить контейнеры, выполнить php artisan migrate:status и health checks.
7. Провести сквозной тест (регистрация, оформление заказа, оплата).

10. Управление информационной безопасностью

- своевременное обновление базового образа, ОС и зависимостей (Composer, npm);
- проверка уведомлений о уязвимостях и тестирование обновлений до выпуска;
- уникальные пароли и минимально необходимые роли для доступа к БД, Redis и серверу;
- отзыв ключей (APP_KEY, JWT_SECRET, API-ключей интеграций) при подозрении на компрометацию;
- запрет вывода секретов и полных персональных данных в журналы;
- ограничение SSH, применение firewall и защиты от перебора паролей;
- периодический аудит активных учётных записей администраторов и сотрудников;
- проверка срока TLS-сертификата и автоматического продления (certbot).

11. Совершенствование ПО

Предложения по развитию формируются из обратной связи пользователей и сотрудников, результатов эксплуатации, инцидентов и требований заказчиков. Каждое улучшение проходит тот же цикл: требование, оценка влияния, реализация, тестирование, выпуск и обновление документации.

Ближайшие функциональные направления, не входящие в версию 1.0:

- расширенная аналитика и дашборды для администраторов;
- интеграция с дополнительными платежными системами;
- мультивалютность и мультиязычность;
- мобильное приложение (iOS/Android);
- **расширенная интеграция с 1С и внешними системами учета** (дополнительные форматы обмена, расширенная номенклатура);
- расширенные возможности программы лояльности.

Примечание: базовая интеграция с 1С уже реализована в версии 1.0 (см. документ VP-QS-02 «Сведения о составе ПО»). В списке выше указаны направления развития, которые дополняют существующую функциональность.

12. Персонал сопровождения

Роль	Необходимые компетенции	Основные задачи
Администратор системы	Linux, Docker Compose, Nginx,	Развертывание, доступы,

	TLS, резервное копирование, базовая диагностика PostgreSQL и Redis	мониторинг, backup/restore и инфраструктурная безопасность
Разработчик	PHP 8.3, Laravel 11, Filament, PostgreSQL, Redis, Meilisearch, Nuxt.js, Vue 3, Git и автоматические тесты	Диагностика кода, исправления, миграции, развитие и выпуск образов
Специалист поддержки/тестирующий	Пользовательские сценарии интернет-магазина, администрирование Filament, регистрация дефектов, приемочное тестирование	Первичная поддержка, воспроизведение, проверка исправлений и актуализация инструкций
Владелец процесса	Бизнес-процессы интернет-торговли, полномочия организации	Управление каталогом, ценообразованием, акциями, сроками и приемка изменений

Для малой инсталляции роли администратора, разработчика и поддержки могут совмещаться, однако изменения в рабочем экземпляре должны быть проверены вторым лицом либо по формализованному чек-листу.

13. Документация и конфигурационный учёт

- версия документа указывается на титульном листе и в нижнем колонтитуле;
- изменение поведения сопровождается изменением соответствующего руководства;
- секретные значения не переносятся в документацию;
- команды и пути сверяются с фактическим Compose-релизом;
- устаревшая редакция архивируется вместе с релизом, но не используется как действующая;
- в карточке релиза фиксируются коммит, image ID, миграции и результаты тестирования.

Классификация ПО

Программное обеспечение «Quans Store» относится к следующим классам реестра отечественного ПО в соответствии с классификатором Министерства цифрового развития, связи и массовых коммуникаций Российской Федерации:

- 09.12 – Средства электронной коммерции;
- 09.11 – Средства управления содержимым (CMS), сайты и порталные решения.

Правообладатель

Исходный код ПО разработан сотрудниками ООО «КИБЕРИЯ» и является объектом авторского права.

Сведения о правообладателе

Наименование организации	ООО «КИБЕРИЯ»
Юридический адрес	656011, Россия, Алтайский край, г. Барнаул, ул. Ярных, дом 35, офис 10
ИНН	2221261043
КПП	222101001
ОГРН	1242200004419

Генеральный директор
ООО «КИБЕРИЯ»
М.П.



/Назмутдинов Р.Т./