

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, pgsql, 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»



/Назмутдинов Р.Т./