ПРАВООБЛАДАТЕЛЬ:

ОБЩЕСТВО С ОГРАНИЧЕННОЙ ОТВЕТСТВЕННОСТЬЮ «ИНТЕНСА»

ИНН 7107540675

ОПИСАНИЕ ПРОЦЕССОВ, ОБЕСПЕЧИВАЮЩИХ ПОДДЕРЖАНИЕ ЖИЗНЕННОГО ЦИКЛА ПО «INTENSA E-COMMERCE PLATFORM» (альтернативное название «INTENSA SHOP»)

Электронный документ

Листов 35

г.Тула

2025

1. ОБЩИЕ СВЕДЕНИЯ И СООТВЕТСТВИЕ НОРМАТИВНЫМ ТРЕБОВАНИЯМ И СТАНДАРТАМ

1.1. Документ содержит описание процессов жизненного цикла ПО «Intensa Shop».

1.2. Применяемые нормативные документы
Процессы разработки и сопровождения ПО «INTENSA SHOP» соответствуют следующим нормативным документам и методическим рекомендациям:

А) Российское законодательство:
- Федеральный закон от 27 июля 2006 г. № 149-ФЗ "Об информации, информационных технологиях и о защите информации"
- Федеральный закон от 27 июля 2006 г. № 152-ФЗ "О персональных данных"
- Приказ Минпромторга России от 19.11.2020 № 1047 "О ведении Реестра отечественного программного обеспечения"
- Постановление Правительства РФ №1236 от 16 ноября 2015 года "О составе сведений, включаемых в единый реестр российского программного обеспечения"

Б) ГОСТ стандарты (с указанием, где они применяются):
- ГОСТ Р 54735-2011 "Качество программных средств. Классификация дефектов"
  → Применяется: при классификации дефектов и приоритизации задач
- ГОСТ Р ИСО/МЭК 12207-2010 "Процессы и организация разработки, поставки, использования, поддержки, снятия с эксплуатации и утилизации программного обеспечения"
  → Применяется: общая методология управления процессами ПО
- ГОСТ Р ИСО/МЭК 27001-2021 "Информационные технологии. Методы и средства обеспечения безопасности. Управление информационной безопасностью"
  → Применяется: при управлении информационной безопасностью
- ГОСТ Р ИСО/МЭК 27002-2012 "Информационные технологии. Методы и средства обеспечения безопасности. Практические правила по управлению информационной безопасностью"
  → Применяется: при разработке политик информационной безопасности
- ГОСТ Р 51622-2000 "Межсетевые экраны. Сборка, испытание и аттестация"
  → Применяется: при настройке брандмауэров и сетевой безопасности
- ГОСТ Р 34.10-2012 "Информационная технология. Криптографическая защита информации. Процессы формирования и проверки электронной подписи"
  → Применяется: при подписи кода и артефактов
- ГОСТ Р 34.11-2012 "Информационная технология. Криптографическая защита информации. Функция хеширования"
  → Применяется: при хешировании данных и кодов

└─ ГОСТ Р 34.201-2020 "Защита от несанкционированного доступа к защищаемой информации. Сертификационное испытание программного обеспечения"
→ Применяется: при сертификации ПО в Реестре

В) Методические рекомендации:
├─ Методические рекомендации АНО ЦКИТ по включению ПО в Реестр
├─ Методические рекомендации Минцифры по разработке и сопровождению ПО
└─ Методические рекомендации по защите персональных данных

Г) Международные стандарты:
├─ ISO/IEC 12207:2017 "Software and systems engineering - Software life cycle processes"
├─ ISO/IEC 27001:2022 "Information security management"
├─ ISO/IEC 25010:2023 "Software quality models"
└─ SWEBOK v3.0 "Software Engineering Body of Knowledge"

2.2. Соответствие процессов ГОСТ Р ИСО/МЭК 12207-2010

Процессы ПО соответствуют модели ГОСТ Р ИСО/МЭК 12207-2010:

ПРОЦЕССЫ ЖИЗНЕННОГО ЦИКЛА

ОСНОВНЫЕ ПРОЦЕССЫ (Primary Life Cycle Processes):
├─ Процесс приобретения (Acquisition)
│ └─ Раздел 2 "Разработка и внедрение"
│
├─ Процесс поставки (Supply)
│ └─ Раздел 4 "Технические средства активации и распространения"
│
├─ Процесс разработки (Development)
│ └─ Раздел 2 "Разработка и внедрение" (архитектура, кодирование)
│
├─ Процесс функционирования (Operation)
│ └─ Раздел 6 "Сопровождение"
│
└─ Процесс обслуживания (Maintenance)
└─ Раздел 6 "Сопровождение" + Раздел 3 "Управление версиями"

ВСПОМОГАТЕЛЬНЫЕ ПРОЦЕССЫ (Supporting Life Cycle Processes):
├─ Процесс документирования (Documentation)
│ └─ Раздел 8 "Документирование"
│
├─ Процесс управления конфигурацией (Configuration Management)
│ └─ Раздел 3 "Управление версиями и релизами"
│

├─ Процесс гарантии качества (Quality Assurance)
│ └─ Раздел 4 "Тестирование" + метрики качества
│
├─ Процесс верификации (Verification)
│ └─ Раздел 4 "Тестирование"
│
├─ Процесс валидации (Validation)
│ └─ Раздел 4 "Тестирование"
│
├─ Процесс управления рисками (Risk Management)
│ └─ Раздел 9 "Управление рисками"
│
└─ Процесс управления конфликтами (Conflict Resolution)
   └─ Раздел 7 "Порядок обработки обращений"

ОРГАНИЗАЦИОННЫЕ ПРОЦЕССЫ (Organizational Life Cycle Processes):
├─ Процесс управления организацией
│ └─ Раздел 8 "Информация о персонале"
│
├─ Процесс совершенствования
│ └─ Раздел 6 "Сопровождение" + планирование развития
│
└─ Процесс подготовки кадров
   └─ Раздел 8 "Информация о персонале"

1.3. Соответствие информационной безопасности

Процессы разработки и сопровождения включают требования информационной безопасности в соответствии с ГОСТ Р ИСО/МЭК 27001-2021:

├─ Политика информационной безопасности: документирована и доступна (см. раздел 9)
├─ Управление доступом: четкие уровни доступа (раздел 8)
├─ Управление активами: инвентаризация всех компонентов (раздел 8)
├─ Контроль доступа: аутентификация и авторизация (раздел 2)
├─ Криптография: использование ГОСТ Р 34.10-2012 (раздел 4 "Активация")
├─ Мониторинг безопасности: логирование и аудит (раздел 5 "CI/CD")
├─ Управление инцидентами: процедура обработки обращений (раздел 7)
└─ Непрерывность деятельности: резервное копирование (раздел 8)

2. УПРАВЛЕНИЕ ТРЕБОВАНИЯМИ (REQUIREMENTS MANAGEMENT). РАЗРАБОТКА И ВНЕДРЕНИЕ.

2.1. Управление требованиями

2.1.1. Процесс сбора требований

Требования к ПО собираются на следующих этапах:

A) Начальная фаза (при разработке новой версии)
├─ Интервью со Stakeholders (заказчик, end-users, бизнес)
├─ Анализ рынка и конкурентов
├─ Изучение обратной связи от текущих пользователей
├─ Анализ обращений в техподдержку
└─ Документирование всех требований в системе управления задачами

B) Текущая фаза (в процессе разработки)
├─ Ежемесячные встречи с заказчиком (Sprint Planning)
├─ Еженедельные синхронизации с командой разработки
├─ Анализ обратной связи пользователей
└─ Приоритизация задач на основе Impact-Effort матрицы

C) Финальная фаза (перед релизом)
├─ Верификация выполнения всех требований
├─ Валидация с заказчиком
├─ Документирование финальных изменений
└─ Создание Release Notes

2.1.2. Классификация требований

Все требования классифицируются по типам:

Функциональные требования (Functional Requirements):
├─ Описание функций ПО (что система должна делать)
├─ Примеры: "Добавление товара в корзину", "Оформление заказа"
├─ Трейсируемость: прямое отображение на тест-кейсы
└─ Статус: выполнено / в разработке / не выполнено

Нефункциональные требования (Non-Functional Requirements):
├─ Производительность (response time <500ms, throughput >100 RPS)
├─ Безопасность (HTTPS/TLS 1.2+, bcrypt for passwords)
├─ Масштабируемость (горизонтальное масштабирование)
├─ Надежность (99.5% uptime SLA)
├─ Пригодность к эксплуатации (документация, поддержка)
└─ Статус: тестируется / выполнено / не выполнено

Требования безопасности (Security Requirements):
├─ Шифрование данных в покое (AES-256)
├─ Шифрование данных в пути (TLS 1.2+)
├─ Управление доступом (RBAC, 2FA)
├─ Защита от типовых атак (OWASP Top 10)
├─ Логирование и аудит (6 месяцев хранения)

└─ Защита персональных данных (ФЗ-152)

Требования соответствия (Compliance Requirements):
├─ ГОСТ Р 34.10-2012 (цифровая подпись)
├─ ГОСТ Р ИСО/МЭК 27001 (информационная безопасность)
├─ ФЗ-152 "О персональных данных"
└─ Реестр ПО Минцифры (документирование)

2.1.3. Управление трейсируемостью требований (Requirements Traceability)

Каждое требование имеет уникальный ID и отслеживается:

Requirements → Design → Code → Tests → Deployment

Пример:
REQ-001: "Система должна поддерживать оплату через Paykeeper"
  ├─ Design: раздел в архитектуре "Модуль обработки платежей"
  ├─ Code: /src/Services/PaymentProcessor.php
  ├─ Tests: tests/Feature/PaymentTest.php
  ├─ PR: #1234 в GitLab
  ├─ Release: v1.2.0
  └─ Status: IMPLEMENTED

Инструмент: система управления задачами (Jira / GitLab Issues / собственная)

2.1.4. Контроль изменений требований (Change Control)

При необходимости изменить требование:

1. Создается Change Request (CR) с описанием
2. Оценивается impact на архитектуру, сроки, стоимость
3. Согласуется с заказчиком
4. Документируется в версионированном документе
5. Добавляется в backlog / roadmap
6. Выполняется на соответствующем спринте

Матрица влияния изменений:
├─ Архитектура: требуется ли передизайн компонентов?
├─ База данных: требуются ли миграции?
├─ API: совместимо ли с существующими клиентами?
├─ Тесты: требуются ли новые тест-кейсы?
├─ Документация: какие разделы нужно обновить?
└─ Сроки: насколько изменится timeline?

2.1.5. Метрики управления требованиями

Для отслеживания качества процесса управления требованиями:

├─ Полнота требований: 100% требований специфицировано до начала разработки
├─ Трейсируемость: каждое требование привязано к коду и тестам
├─ Валидность: требования соответствуют зафиксированным потребностям заказчика
├─ Стабильность: <10% требований изменяется на спринт
├─ Покрытие тестами: 100% функциональных требований покрыто тестами
├─ Прозрачность: заказчик имеет доступ к статусу всех требований
└─ Соответствие архитектуре: требования соответствуют техническим возможностям

2.1.6. Поддержание жизненного цикла программного обеспечения «Intensa Shop» осуществляется за счет развития и сопровождения программного обеспечения (далее – ПО).

Развитие ПО предполагает, в том числе предоставление со стороны Заказчика технических и функциональных требований к ПО, а со стороны Исполнителя выполнение работ по аналитике, разработке, тестированию и внедрению в зависимости от заключённых договоров и соглашений.

ПО развертывается и используется Заказчиком по модели SaaS, а также может быть установлено в инфраструктуре Заказчика.

2.2. Архитектурные принципы

ПО «Intensa Shop» построено на следующих архитектурных принципах:

1. Модульная архитектура
   - Каждый модуль представляет отдельную функциональную область
   - Четкие границы между модулями
   - Минимизация зависимостей между модулями

2. Слоистая архитектура (Clean Architecture)
   Каждый модуль разделен на три слоя:

   Domain (Доменный слой):
   - Entities - сущности предметной области
   - Value Objects - объекты-значения
   - Domain Services - доменные сервисы
   - Repository Interfaces - интерфейсы репозиториев
   - Domain Events - доменные события

Application (Прикладной слой):
- Commands & Handlers - команды и их обработчики
- Queries & Handlers - запросы и их обработчики
- DTOs - объекты передачи данных
- Application Services - сервисы приложения

Infrastructure (Инфраструктурный слой):
- Repository Implementations - реализации репозиториев
- External Service Adapters - адаптеры внешних сервисов
- Database Models - модели базы данных

3. SOLID принципы объектно-ориентированного программирования
- Single Responsibility Principle
- Open/Closed Principle
- Liskov Substitution Principle
- Interface Segregation Principle
- Dependency Inversion Principle

Технологический стек

Основные технологии:
- PHP 8.4 - язык программирования
- Laravel 11.x - веб-фреймворк
- PostgreSQL 17 - система управления базами данных
- Redis 7.x - система кэширования и очередей
- Docker - контейнеризация приложений

Дополнительные инструменты:
- Composer - управление зависимостями PHP
- PHPUnit - фреймворк тестирования
- Laravel Telescope - инструмент отладки и мониторинга

Модульная структура

Проект организован в виде следующих модулей:

- Cart - модуль корзины покупок
- Exchange - модуль обмена данными с 1С
- Filament - модуль административной панели
- Form - модуль форм обратной связи

- Mindbox - модуль интеграции с CDP системой
- Order - модуль управления заказами
- Post - модуль публикаций и новостей
- Products - модуль каталога товаров
- User - модуль управления пользователями
- Shared - общие компоненты системы

3. УПРАВЛЕНИЕ ВЕРСИЯМИ И РЕЛИЗАМИ

Система контроля версий

Для управления версиями исходного кода используется система контроля версий Git.

Организация репозитория:
- Основная ветка: main (продакшн-версия)
- Ветка разработки: develop (интеграция изменений)
- Функциональные ветки: feature/* (разработка новых функций)
- Ветки исправлений: fix/* (исправление ошибок)
- Ветки релизов: release/* (подготовка релизов)

Семантическое версионирование

ПО использует семантическое версионирование (SemVer) формата MAJOR.MINOR.PATCH:

MAJOR - увеличивается при внесении несовместимых изменений в API
MINOR - увеличивается при добавлении новой функциональности с обратной совместимостью
PATCH - увеличивается при исправлении ошибок с обратной совместимостью

Этапы версионирования:

1. Активная разработка (0.x.x):
   - Версии 0.0.x - начальная разработка
   - Версии 0.1.x до 0.9.x - активная разработка функционала
   - Breaking changes разрешены в MINOR версии

2. Стабильная разработка (1.x.x+):
   - Версия 1.0.0 - стабильный релиз, готовый к продакшену
   - Строгое следование SemVer
   - Breaking changes только в MAJOR версии

Влияние типов изменений на версии:

Тип коммита	Версия 0.x.x	Версия 1.x.x+
fix:	0.1.1 → 0.1.2	1.1.1 → 1.1.2
feat:	0.1.1 → 0.2.0	1.1.1 → 1.2.0
feat!:	0.1.1 → 0.2.0	1.1.1 → 2.0.0

Процесс создания релизов

Для автоматизации процесса релиза используется пакет intensa/release-tools.

Основные команды:
vendor/bin/release.sh 0.1.5 # Полный релиз
vendor/bin/release.sh 0.1.5 prepare # Только подготовка
vendor/bin/release.sh 0.1.5 publish # Только публикация
vendor/bin/generate-changelog.sh # Генерация changelog

Конвенциональные коммиты:
Все коммиты следуют формату: <type>[scope]: [TASK-ID] <description>

Где:
- type - тип изменения (feat, fix, docs, style, refactor, test, chore)
- scope - область изменения (необязательно)
- TASK-ID - идентификатор задачи
- description - краткое описание изменения

Примеры:
feat(auth): IEP-1234 добавлена поддержка OAuth2
fix(validation): IEP-1235 исправлена валидация email
docs: обновлена документация API

4. ТЕСТИРОВАНИЕ

4.1. Стратегия тестирования

ПО использует пирамиду тестирования, включающую:

1. Модульные тесты (Unit Tests)
   - Тестирование отдельных классов и методов

- Изоляция через моки и стабы
- Быстрое выполнение (<1 секунды на тест)
- Высокое покрытие кода (стремление к 80%+)

2. Интеграционные тесты (Integration Tests)
- Тестирование взаимодействия компонентов
- Проверка работы с базой данных
- Тестирование внешних интеграций

3. Функциональные тесты (Functional Tests)
- Тестирование API-эндпоинтов
- Проверка полного цикла запрос-ответ
- Валидация форматов данных

4. Приемочные тесты (Acceptance Tests)
- Тестирование с точки зрения пользователя
- Проверка соответствия требованиям

4.2. Инструменты тестирования

Основной фреймворк: PHPUnit

Структура тестов:

```
tests/
├── Unit/             # Модульные тесты
├── Integration/      # Интеграционные тесты
└── Feature/          # Функциональные тесты
```

Команды запуска:

```
php artisan test                  # Запуск всех тестов
php artisan test --testsuite=Unit   # Только модульные тесты
php artisan test --coverage         # С покрытием кода
```

Дополнительные инструменты:
- Model Factories - генерация тестовых данных
- Database Seeders - заполнение тестовой БД
- Mockery - создание моков для изоляции тестов

Критерии качества

Обязательные метрики:

- Покрытие кода тестами: минимум 80% для доменной логики
- Цикломатическая сложность: максимум 10 для методов
- Дублирование кода: максимум 5%

Автоматические проверки при каждом коммите:
- Запуск линтеров (PHP CS Fixer)
- Запуск статического анализа (PHPStan)
- Запуск всех тестов
- Проверка покрытия кода

4.3. КРИТЕРИИ ПРИЕМКИ (ACCEPTANCE CRITERIA)

Каждая задача (story / feature) имеет четко определенные критерии приемки:

Пример:
Feature: "Добавление товара в корзину"

Критерии приемки:
✓ Пользователь видит кнопку "Добавить в корзину"
✓ При нажатии количество в корзине увеличивается
✓ При повторном нажатии на тот же товар количество увеличивается
✓ Цена товара считается корректно (с учетом скидок)
✓ Корзина сохраняется при перезагрузке страницы
✓ API возвращает 200 и обновленные данные корзины
✓ Error handling: 400 если quantity невалидна
✓ Performance: добавление товара происходит <500ms

Критерии приемки проверяются:
├─ В процессе разработки (developer self-test)
├─ При code review (peer review)
├─ При testing (QA team)
└─ При демонстрации заказчику (customer acceptance)

5. НЕПРЕРЫВНАЯ ИНТЕГРАЦИЯ И ДОСТАВКА (CI/CD)

Этапы CI Pipeline

Этапы непрерывной интеграции при каждом push в репозиторий:

1. Checkout кода
   Получение последней версии из репозитория

2. Установка зависимостей
   composer install --no-dev --optimize-autoloader
   npm install --production

3. Статический анализ
   - PHP CS Fixer - проверка стиля кода

4. Запуск тестов
   php artisan test --parallel
   php artisan test --coverage --min=80

5. Сборка артефактов
   - Компиляция ассетов
   - Оптимизация autoloader
   - Создание архива для развертывания

Этапы CD Pipeline

Development среда (автоматически):
1. Развертывание на dev-сервер
2. Запуск миграций базы данных
3. Очистка кеша приложения
4. Запуск smoke-тестов

Staging среда (автоматически после прохождения CI):
1. Развертывание на staging-сервер
2. Запуск миграций базы данных
3. Запуск интеграционных тестов
4. Уведомление команды

Production среда (с ручным подтверждением):
1. Создание резервной копии
2. Переключение в режим обслуживания
3. Развертывание новой версии
4. Запуск миграций базы данных
5. Очистка кеша приложения
6. Smoke-тесты
7. Выход из режима обслуживания
8. Мониторинг метрик

6. СОПРОВОЖДЕНИЕ

Сопровождение ПО со стороны исполнителя включает в себя: регистрацию и анализ обращений по возникающим функциональным или техническим ошибкам, консультации по функциям и вопросам эксплуатации, предоставление рекомендаций по оптимизации в зависимости от уровня сервисного обслуживания.

Сопровождение ПО необходимо для обеспечения:
- отсутствия простоя в работе Заказчика по причине невозможности функционирования ПО (аварийная ситуация, ошибки в работе ПО, ошибки в работе пользователей Заказчика и т.п.);
- обеспечения гарантий корректного функционирования ПО и дальнейшего развития её функционала.

Обозначенные цели должны быть достигнуты путем:
- консультирования пользователей и администраторов ПО по вопросам эксплуатации через электронную почту, телефонную связь или письменному запросу Заказчика;
- обеспечение Заказчика новыми версиями ПО по мере их выхода;
- обеспечение Заказчика изменениями и дополнениями к эксплуатационной и пользовательской документации;
- устранение ошибок в случае выявления их при работе с ПО.

Организация поддержки

Система поддержки организована в три уровня:

1-я линия (L1) - Пользовательская поддержка:
- Прием и классификация обращений
- Решение типовых проблем
- Консультации пользователей
- Время реакции: 4 часа

2-я линия (L2) - Техническая поддержка:
- Решение технических проблем
- Анализ логов и ошибок
- Подготовка патчей
- Время реакции: 8 часов

3-я линия (L3) - Экспертная поддержка:
- Сложные технические проблемы
- Оптимизация производительности
- Архитектурные изменения
- Время реакции: 24 часа

Классификация по приоритету:

Critical (P1):
- Полная недоступность системы
- Критические ошибки безопасности
- SLA: решение в течение 4 часов

High (P2):
- Частичная недоступность функционала
- Серьезные ошибки в бизнес-логике
- SLA: решение в течение 24 часов

Medium (P3):
- Некритичные ошибки
- Неудобства в использовании
- SLA: решение в течение 5 рабочих дней

Low (P4):
- Косметические проблемы
- Запросы на улучшения
- SLA: решение в течение 2 недель

Обновление и патчи

Типы обновлений:

1. Patch обновления (x.x.N):
   - Исправления критических ошибок
   - Обновления безопасности
   - Рекомендуется устанавливать немедленно

2. Minor обновления (x.N.x):
   - Новая функциональность
   - Некритичные исправления
   - Планируются регулярно

3. Major обновления (N.x.x):
   - Значительные изменения
   - Несовместимые изменения API
   - Требуют тщательной подготовки

График плановых обновлений:

- Minor релизы: ежемесячно (первая неделя месяца)
- Patch релизы: по мере необходимости
- Major релизы: 1-2 раза в год

Технические окна для обновлений:
- Плановые: воскресенье 03:00-06:00
- Экстренные: по согласованию

Информация о технической поддержке:
Техническая поддержка осуществляется правообладателем.
Время работы технической поддержки: Пн-Пт 9:00-18:00
Контактные данные технической поддержки: support@intensa-shop.ru

7. ПЕРЕЧЕНЬ ОКАЗЫВАЕМЫХ УСЛУГ В РАМКАХ СОПРОВОЖДЕНИЯ ПО «INTENSA SHOP»

Ниже перечислены возможные варианты и составы предоставляемых услуг в рамках договора сопровождения. Состав и наличие определенных услуг может измениться в зависимости от согласованных условий договора.

Консультационные услуги:

- сбор дополнительных данных, необходимых для анализа несоответствий в работе ПО;
- локализация и выяснение причин возникновения инцидентов;
- рекомендации по возможным способам компенсации функциональности ПО, затронутых инцидентом;
- оказание консультаций по вопросам архитектуры и устройства ПО (за исключением программного кода - исходного текста, объектного кода);
- консультации по интеграции с внешними системами (1С, платежные системы, CDP);
- консультации по оптимизации производительности ПО;
- прием обращений с предложениями о доработке - расширении функциональных возможностей ПО (доработка ПО в рамках Договора сопровождения Исполнителем не производится).

Стандартные услуги:

- устранение инцидентов, возникших вследствие того, что ПО функционирует не в соответствии с документацией, включая выпуск:
  - оперативных обновлений ПО;
  - внесение изменений в документацию;
  - разработка скриптов корректировки данных ПО, затронутых инцидентом;
- внесение изменений в пользовательскую и эксплуатационную документацию;

- профилактические работы и техническое обслуживание ПО, включая:
  - анализ лог-файлов, содержащих данные об ошибках и предупреждениях ПО;
  - анализ состояния базы данных в части избыточности служебной системной информации;
  - мониторинг производительности и доступности ПО;
- управление процессом оказания стандартных и консультационных услуг, предусмотренных договором сопровождения;
- предоставление новых версий ПО по мере их выпуска;
- обновление API документации;
- резервное копирование данных (при развертывании в инфраструктуре Заказчика).

Дополнительные услуги:

- оказание услуг специалистами Исполнителя на территории Заказчика в целях оказания услуг по техническому сопровождению ПО (выезд), не входящих в состав стандартных услуг (в случае развертывания ПО в инфраструктуре Заказчика);
- устранение инцидентов, возникших вследствие причин, не связанных с некорректным (несоответствующим документации) функционированием ПО;
- инструктаж пользователей ПО;
- выполнение сервисных работ по эксплуатации ПО, в том числе:
  - участие в настройке ПО на тестовых стендах Заказчика;
  - поддержка тестового стенда на территории Заказчика;
  - установка патча и участие в приемо-сдаточных испытаниях;
  - участие в разработке и поддержании в актуальном состоянии регламентного обеспечения процессов сопровождения ПО;
  - настройка интеграций с внешними системами;
  - оптимизация производительности ПО;
- дополнение документации ПО по запросу Заказчика;
- предоставление оперативных обновлений;
- консультационные услуги и стандартные услуги, оказываемые в расширенное время оказания услуг;
- консультационная поддержка при разборе конфликтных ситуаций с клиентами Заказчика, а также при решении спорных вопросов внутри организационной структуры Заказчика, связанных с использованием ПО, за исключением ситуаций, возникших по вине Исполнителя;
- консультационная поддержка при разборе внештатных ситуаций в качестве экспертов по функционированию ПО за исключением ситуаций, возникших по вине Исполнителя;
- обучение персонала Заказчика работе с ПО;
- проведение аудита безопасности ПО;
- помощь в миграции данных между версиями ПО.

8. ПОРЯДОК РЕГИСТРАЦИИ И ОБРАБОТКИ ОБРАЩЕНИЙ. ИНФОРМАЦИЯ О ПЕРСОНАЛЕ. ДОКУМЕНТИРОВАНИЕ.

8.1. Обработка обращений.
Исполнитель принимает обращения по электронной почте: support@intensa.ru
Телефон для связи +7 (495) 255-21-06

Стандартное время приема и регистрации обращений: с 9-00 до 18-00 в рабочие дни.

Под обращением подразумевается сообщение об обнаружении несоответствия, информация о необходимой консультации и/или предложения о доработке ПО.

Работа по обращению выполняется в следующем порядке:

1. При обнаружении несоответствия Заказчик направляет Исполнителю по электронной почте обращение в свободной или согласованной в рамках договора форме. Форма заполняется максимально полно, при отсутствии у Заказчика информации по некоторым пунктам может быть принята Исполнителем.

Рекомендуемая структура обращения:
- Описание проблемы
- Шаги воспроизведения
- Ожидаемое поведение
- Фактическое поведение
- Версия ПО
- Окружение (браузер, ОС)
- Скриншоты/логи (при наличии)

2. В целях повышения оперативности обработки обращения для дополнительных вопросов/уточнений Заказчика может быть использован телефон (или иные средства) для связи с Исполнителем.

3. Сроки оказания услуг начинаются в момент получения Исполнителем обращения. Если обращение было направлено вне стандартного времени оказания услуг, время начала его обработки совпадает с началом исчисления следующего стандартного времени оказания услуг.

4. Исполнитель осуществляет обработку обращения (регистрацию, проверку и анализ информации, предоставление обходного или промышленного решения, оказание консультации) в соответствии с приоритетом, указанным Заказчиком в соответствии с регламентным временем, согласованным в рамках договора сопровождения.

5. По результатам работы (устранения инцидента) Исполнитель передает Заказчику обходное решение или патч с оперативным обновлением ПО, устраняющим ошибку. В случае, если обходное решение предоставить невозможно, то Исполнитель по согласованию с Заказчиком осуществляет только устранение ошибки или консультирование.

6. В случае, если предполагаемое в обращении несоответствие не подтверждается, Исполнитель предоставляет Заказчику ссылку на положение документации по ПО, подтверждающее факт функционирования ПО согласно объявленному в документации, либо сообщает об отсутствии соответствующих документов, регламентирующих указанное функционирование. При этом Исполнитель предоставляет общие рекомендации по дальнейшей работе с ситуацией, указанной в обращении.

7. После решения проблемы Исполнитель уведомляет Заказчика о закрытии обращения с указанием выполненных работ и рекомендациями по предотвращению подобных ситуаций в будущем.

8.2 Информация о персонале.

1) Пользователи

Пользователи ПО должны обладать навыками работы с персональным компьютером на уровне пользователя и веб-браузером.

Для работы с ПО пользователю необходимо изучить свои должностные инструкции и руководства по функционированию ПО и при необходимости пройти курс обучения по работе с ПО.

Уровни доступа пользователей:
- Администратор - полный доступ к административной панели
- Менеджер - управление заказами и клиентами
- Контент-менеджер - управление контентом и публикациями
- Клиент - доступ к личному кабинету и оформлению заказов

2) Команда сопровождения

Состав команды сопровождения регламентируется согласованными в рамках договора сопровождения условиями. Рекомендуемый состав команды сопровождения со стороны Исполнителя:

- Руководитель группы технической поддержки
- Backend разработчик (PHP/Laravel)
- Frontend разработчик (JavaScript/Vue.js/Nuxt.js)
- DevOps инженер
- Инженер по тестированию
- Специалист технической поддержки (L1)

Квалификационные требования:

- Опыт работы с технологиями: PHP 8.4+, Laravel 11.x, PostgreSQL, Redis
- Знание архитектурных паттернов: DDD, Clean Architecture
- Опыт работы с системами контроля версий (Git)
- Навыки отладки и профилирования приложений

Команда разработки и внедрения

Состав команды разработки и внедрения регламентируется согласованными в рамках договора внедрения условиями, сроками и объемом работ. Рекомендуемый состав команды со стороны Исполнителя:

- Руководитель проекта
- Системный аналитик/архитектор
- Backend разработчики (PHP/Laravel) - 2-3 специалиста
- Frontend разработчики (JavaScript/Vue.js/Nuxt.js) - 2 специалиста
- DevOps инженер
- Инженеры по тестированию - 2 специалиста
- Технический писатель

Квалификационные требования:
- Опыт коммерческой разработки от 3 лет
- Глубокие знания технологического стека проекта
- Опыт проектирования и разработки высоконагруженных систем
- Знание принципов Domain-Driven Design и Clean Architecture
- Опыт работы с CI/CD и автоматизацией развертывания
- Навыки написания технической документации

Инфраструктура разработки:

Серверы разработки и тестирования:
- Development сервер (dev.intensa-shop.ru)
- Staging сервер (staging.intensa-shop.ru)
- Production сервер (intensa-shop.ru)

Системы управления:
- Система контроля версий: Git
- Система управления задачами: внутренняя система трекинга
- Система документации: внутренний портал документации
- Система мониторинга: внутренняя система мониторинга и алертинга

Резервное копирование:

Все данные регулярно резервируются в соответствии со следующим графиком:
- База данных: ежедневные полные копии, инкрементальные каждые 6 часов
- Файлы приложения: еженедельные полные копии
- Пользовательские данные: ежедневные копии

Резервные копии хранятся в географически распределенных дата-центрах для обеспечения отказоустойчивости.

8.3. ДОКУМЕНТИРОВАНИЕ

8.3.1. Виды документации

Документация по ПО «INTENSA SHOP» включает следующие виды:

А) ТЕХНИЧЕСКАЯ ДОКУМЕНТАЦИЯ:

1. Архитектурная документация
   ├─ Описание технической архитектуры (раздел 4.1-4.6)
   ├─ Диаграммы компонентов (4.1-4.5)
   ├─ Entity-Relationship диаграммы (5.2)
   ├─ API диаграммы взаимодействия
   ├─ Развертывание диаграммы (инфраструктура)
   └─ Обновляется: при изменении архитектуры

2. API Документация
   ├─ Swagger/OpenAPI спецификация
   ├─ Примеры запросов и ответов
   ├─ Коды ошибок (400, 401, 403, 404, 422, 500)
   ├─ Rate limiting информация
   ├─ Аутентификация и авторизация
   ├─ Версионирование API (v1, v2)
   └─ Инструмент: Swagger UI / ReDoc

3. База данных документация
   ├─ Entity-Relationship диаграмма
   ├─ Описание каждой таблицы
   ├─ Описание колонок (тип, ограничения, индексы)
   ├─ Описание foreign keys и relationships
   ├─ Описание хранимых процедур (если есть)
   └─ Обновляется: при каждой миграции БД

4. Инфраструктурная документация
   ├─ Описание серверной архитектуры
   ├─ Конфигурация Nginx / Kubernetes
   ├─ Переменные окружения (.env примеры)
   ├─ Требования к хостингу (CPU, RAM, disk, network)
   ├─ Процедуры развертывания (development, staging, production)
   └─ Обновляется: при изменении инфраструктуры

5. Code documentation
   ├─ PHPDoc комментарии в коде (для каждого класса и метода)
   ├─ README файлы в папках модулей
   ├─ CONTRIBUTING.md для новых разработчиков
   ├─ CODE_STYLE.md для стиля кода
   └─ Обновляется: вместе с кодом (обязательно)

B) ПОЛЬЗОВАТЕЛЬСКАЯ ДОКУМЕНТАЦИЯ:

1. Руководство пользователя (User Guide)
   ├─ Начало работы (Getting Started)
   ├─ Описание основных функций
   ├─ Пошаговые инструкции (tutorials)
   ├─ Скриншоты и видео
   ├─ FAQ (часто задаваемые вопросы)
   └─ Языки: Русский + Английский

2. Руководство администратора (Admin Guide)
   ├─ Управление пользователями и ролями
   ├─ Управление товарами и категориями
   ├─ Управление заказами и платежами
   ├─ Интеграция с внешними системами
   ├─ Резервное копирование и восстановление
   └─ Мониторинг и логирование

C) ПРОЦЕССНАЯ ДОКУМЕНТАЦИЯ:

1. Документация процессов разработки
   ├─ Жизненный цикл ПО (это документ)
   ├─ Процесс разработки новой функции
   ├─ Процесс исправления ошибок
   ├─ Процесс создания релиза
   └─ Процесс развертывания

2. Документация процессов сопровождения

├─ Процесс обработки обращений (раздел 7)
├─ Процесс классификации инцидентов
├─ Процесс выпуска патчей
└─ Процесс SLA мониторинга

D) НОРМАТИВНАЯ ДОКУМЕНТАЦИЯ:

1. Лицензии и соглашения
├
├─ NDA (Non-Disclosure Agreement)
├─ Соглашение об обработке персональных данных
└─ Terms of Service

2. Политики
├─ Политика конфиденциальности
├─ Политика безопасности (раздел 9)
├─ Политика использования cookies
└─ Политика возврата и отмены

8.3.2. Требования к документации

Все документы должны соответствовать следующим требованиям:

A) Содержание
├─ Точность: все информация актуальна и проверена
├─ Полнота: охватывают все аспекты функциональности
├─ Ясность: написаны простым и понятным языком
├─ Структурированность: логическая организация с оглавлением
├─ Примеры: содержат практические примеры использования
└─ Обновление: актуализируются с каждым релизом

B) Форматы
├─ Текстовые: Markdown (GitLab)
├─ PDF: для официальной документации
├─ HTML: для веб-сайта и в Swagger
├─ Видео: для обучающих материалов (Rutub)
└─ Интерактивные: живые примеры в API документации

C) Языки
├─ Русский: обязательно для всех документов
├─ Английский: обязательно для технической документации
└─ Другие: по запросу заказчика

D) Версионирование
├─ Все документы хранятся в Git репозитории
├─ Версия документа совпадает с версией ПО
├─ История изменений отслеживается через Git commits
├─ Changelog документируется для каждой версии

8.3.3. Процесс обновления документации

Когда должна обновляться документация:

1. При разработке новой функции
   ├─ Документация пишется параллельно с кодом
   ├─ API документация обновляется в Swagger
   ├─ Пользовательская документация обновляется
   └─ Требование: документация перед code review

2. При исправлении ошибки
   ├─ Если ошибка в документации → обновить
   ├─ Если нужно уточнить документацию → обновить
   └─

3. При изменении API
   ├─ Swagger спецификация обновляется первой
   ├─ Примеры запросов обновляются
   ├─ Старая версия API документируется как deprecated
   └─ Migration guide пишется для пользователей

4. При мажорном релизе
   ├─ Пересмотр всей документации
   ├─ Обновление скриншотов (если изменен UI)
   ├─ Пересоздание видеоуроков
   └─ Обновление FAQ

8.3.4. Инструменты документирования

Используемые инструменты:

├─ Git + Markdown: для технической документации
├─ Swagger/OpenAPI: для API документации
├─ PlantUML: для диаграмм (архитектура, БД)
├─ Wiki: внутренний wiki для команды
├─ Video: OBS Studio для видеоуроков

8.3.5. Ответственность за документацию

├─ Архитектурная документация: Lead разработчик / Архитектор
├─ API документация: Backend разработчик + Technical writer
├─ Пользовательская документация: Product Manager + Technical writer
├─ Инфраструктурная документация: DevOps инженер
├─ Code documentation: Каждый разработчик (за свой код)
└─ Процессная документация: Руководитель проекта / Team lead

9. ПОЛИТИКА ИНФОРМАЦИОННОЙ БЕЗОПАСНОСТИ В ПРОЦЕССАХ РАЗРАБОТКИ

9.1. Принципы информационной безопасности

При разработке и сопровождении ПО «INTENSA SHOP» применяются следующие принципы информационной безопасности (в соответствии с ГОСТ Р ИСО/МЭК 27001):

Принцип 1: Confidentiality (Конфиденциальность)
├─ Исходный код хранится в защищенном репозитории (GitLab)
├─ Доступ ограничен только авторизованным лицам
├─ Используется 2FA (двухфакторная аутентификация)
├─ Все чувствительные данные (API ключи) хранятся в защищенном vault
└─ Логирование доступа: все операции записываются в аудит-логи

Принцип 2: Integrity (Целостность)
├─ Все коммиты подписаны (GPG署名)
├─ Требуется code review перед merge
├─ Каждая версия ПО имеет криптографический хеш (SHA-256)
├─ Используется ГОСТ Р 34.10-2012 для подписей артефактов
└─ Проверка целостности при развертывании

Принцип 3: Availability (Доступность)
├─ SLA: 99.5% uptime (максимум 5 часов простоя в месяц)
├─ Мониторинг 24/7 (Zabbix)
├─ Резервные копии ежедневно (+ инкрементальные каждые 6 часов)
├─ Failover <30 сек при отказе компонента
└─ Процедура Disaster Recovery протестирована квартально

Принцип 4: Authentication (Аутентификация)
├─ Для разработчиков: 2FA в GitLab (TOTP или U2F)
├─ Для On-Premise: лицензионный ключ с цифровой подписью
├─ Для API: Bearer токены (JWT с подписью RS256)
└─ Для сервис-аккаунтов: автоматическая ротация ключей (каждый месяц)

Принцип 5: Authorization (Авторизация)
├─ Role-Based Access Control (RBAC) для всех уровней
├─ Разработчик: доступ только в свою функциональную область (code review в других)
├─ DevOps: доступ в staging + ограниченный доступ в production
├─ Admin: полный доступ (с логированием)
├─ Audit: просмотр логов действий без возможности изменения
└─ Периодическая ревизация прав доступа (раз в квартал)

9.2. Безопасность исходного кода

Требования при работе с исходным кодом:

A) Хранилище (Git Repository)
├─ Место: GitLab (на территории РФ)
├─ Защита: HTTPS + SSH ключи (минимум 4096 бит)
├─ Ветки: protect main и develop (требуется 2 approval для merge)
├─ История: нельзя переписывать историю (force push запрещен)
├─ Резервные копии: ежедневно на S3 (отдельный регион)
└─ Доступ: логирование всех pull, fetch, push операций

B) Управление секретами (Secrets Management)
├─ API ключи: хранятся в HashiCorp Vault, не в коде
├─ Пароли БД: никогда не коммитятся, используются переменные окружения
├─ SSH ключи: защищены passphrase, не коммитятся
├─ Сканирование: перед каждым commit проверяется наличие секретов
├─ Инструмент: GitLeaks + TruffleHog (автоматическое сканирование)
└─ Инцидент: если secret обнаружен, немедленно ротация

C) Безопасность зависимостей (Dependency Security)
├─ Инвентаризация: все зависимости в composer.json и package.json
├─ Сканирование: еженедельно проверяются CVE (Common Vulnerabilities)
├─ Инструменты: Dependabot + Snyk + OWASP Dependency-Check
├─ Ограничение версий: используются точные версии (не ^x.y.z)
├─ Updates: критичные обновления устанавливаются немедленно
├─ Процедура:
│ ├─ Обнаружение CVE в зависимости
│ ├─ Оценка риска (CVSS score)
│ ├─ Обновление версии (если доступно)
│ ├─ Тестирование (полная регрессия)
│ └─ Развертывание на production
└─ SBOM (Software Bill of Materials): публикуется для каждого релиза

9.3. Безопасность тестирования

Требования при тестировании ПО:

A) Тестовые данные
├─ Использование: только сгенерированные (fake) данные, реальные запрещены
├─ Персональные данные: используются плацебо-значения (john.doe@example.com)
├─ Финансовые данные: используются тестовые номера карт (4111 1111 1111 1111)
├─ Регулярная очистка: тестовые данные удаляются после каждого спринта
└─ Изоляция: тестовые БД отделены от production БД

B) Тестирование безопасности
├─ OWASP ZAP: сканирование веб-приложения ежемесячно
├─ Penetration Testing: внешний пентест ежеквартально
├─ SAST (Static Application Security Testing): при каждом push (PHPStan + Psalm)
├─ DAST (Dynamic Application Security Testing): в staging перед production
├─ Dependency scanning: еженедельно (Snyk / Dependabot)
├─ Результаты: критичные уязвимости блокируют merge в main
└─ Мониторинг: Sentry для отслеживания ошибок на production

C) Тестовые окружения
├─ Production: реальные данные, максимальная безопасность
├─ Staging: копия production (но данные анонимизированы)
├─ Development: локальное окружение разработчика
├─ CI: изолированное окружение для автоматических тестов
└─ Доступ: только авторизованные лица, все логируется

9.4. Безопасность развертывания (Deployment Security)

При развертывании ПО на production:

A) До развертывания
├─ Code review: как минимум 2 одобрения
├─ Тесты: все тесты должны пройти (100% в critical path)
├─ Security scan: OWASP ZAP + PHPStan должны пройти без критичных
├─ Резервная копия: БД бэкап перед development
└─ Backup plan: процедура rollback готова

B) Во время развертывания
├─ Режим обслуживания: приложение переходит в maintenance mode
├─ Миграции: выполняются с откатом при ошибке
├─ Smoke tests: критичные функции проверяются вручную
├─ Мониторинг: логирование всех ошибок в реальном времени

└─ Окно развертывания: обычно в ночное время (02:00-05:00 UTC)

C) После развертывания
├─ Проверка метрик: CPU, memory, disk, network в норме?
├─ Проверка логов: нет критичных ошибок?
├─ Проверка функциональности: основные сценарии работают?
├─ Уведомление: team notified of successful deployment
└─ Мониторинг: усиленное наблюдение первые 2 часа

A) Процесс rollback
├─ Условие: если критичная ошибка обнаружена в течение 1 часа
├─ Процедура:
│ ├─ 1. Немедленное уведомление team
│ ├─ 2. Переключение на предыдущий version (Docker image rollback)
│ ├─ 3. Проверка стабильности
│ └─ 4. Post-mortem meeting
├─ Время: <5 минут на выполнение rollback
└─ Документирование: в инцидент-репорт

9.5. Безопасность персонала

Требования к сотрудникам:

A) При найме
├─ Проверка квалификации: проверяются скиллы и опыт
├─ Проверка анкеты: анкета о конфликтах интересов
├─ NDA подписание: обязательное для всех сотрудников
├─ Обучение безопасности: вводный курс по политике ИБ
└─ Получение доступа: 2FA, SSH ключ, аккаунт в системах

B) Во время работы
├─ Ежегодное обучение: курс по информационной безопасности
├─ Ревизия доступов: раз в квартал проверяется наличие необходимых прав
├─ Отклонение от стандартов: обсуждение и исправление
├─ Инциденты: немедленное уведомление руководства
└─ Аудит: случайная проверка действий сотрудников (sampling audit)

C) При увольнении
├─ Отзыв доступа: немедленное отключение всех аккаунтов
├─ Возврат оборудования: ноутбук, SSH ключи и т.д.
├─ Архивирование: документы сотрудника архивируются
├─ Знания: передача проектов на других сотрудников
└─ Follow-up: проверка удаления доступа через месяц

9.6. Мониторинг и инцидент-менеджмент

A) Мониторинг безопасности
├─ Алерты: Alertmanager для уведомления при аномалиях
├─ Анализ: SIEM система для корреляции событий
├─ Реагирование: автоматические ответы на типовые инциденты
└─ Отчетность: ежемесячный отчет по инцидентам и трендам

B) Процесс обработки инцидента безопасности
├─ 1. DETECTION: обнаружение инцидента (автомат или вручную)
├─ 2. ANALYSIS: анализ масштаба и impact
├─ 3. CONTAINMENT: остановка распространения инцидента
├─ 4. ERADICATION: удаление причины инцидента
├─ 5. RECOVERY: восстановление нормальной работы
├─ 6. LESSONS LEARNED: анализ и улучшения
└─ Время SLA:
  ├─ Critical: анализ в течение 1 часа
  ├─ High: анализ в течение 4 часов
  └─ Medium/Low: анализ в течение 24 часов

C) Уведомление stakeholders
├─ Пользователей: если затронуты их данные
├─ Заказчика: если затронута его инфраструктура
├─ Регуляторов: если требуется по закону (ФЗ-152)
├─ Сроки: в течение 72 часов (в соответствии с ФЗ-152)
└─ Документирование: все уведомления логируются

9.7. Соответствие требованиям защиты персональных данных (ФЗ-152)

При работе с персональными данными пользователей:

A) Основные требования
├─ Согласие: получено согласие на обработку ПД
├─ Назначение: обработка только для указанных целей
├─ Принцип минимизации: собираются только необходимые данные
├─ Хранение: только на территории РФ (ОЧЕНЬ ВАЖНО!)
├─ Время хранения: ограничено (удаление после срока действия)
└─ Безопасность: использование шифрования и других мер

B) Права субъектов ПД
├─ Право на доступ: пользователь может скачать свои данные
├─ Право на исправление: пользователь может обновить свои данные

├─ Право на удаление: пользователь может запросить удаление
├─ Право на ограничение: пользователь может ограничить обработку
├─ Право на передачу: пользователь может получить данные в стандартном формате
└─ Процесс: /api/v1/users/export-data, /api/v1/users/delete-account

C) Документирование
├─ Политика конфиденциальности: опубликована на сайте
├─ DPIA (Data Protection Impact Assessment): проведена для критичных обработок
├─ Реестр обработок: все обработки ПД задокументированы
└─ Соглашение об обработке: заключено с провайдерами облака

9.8. Аудит и соответствие

A) Внутренние аудиты
├─ Частота: ежеквартально
├─ Проверяемые области: доступы, логи, политики
├─ Инструменты: скрипты автоматизированной проверки
└─ Отчет: предоставляется руководству

B) Внешние аудиты
├─ Сертификация: ISO/IEC 27001 (раз в 3 года)
├─ Пентест: внешний penetration testing ежеквартально
├─ Соответствие: проверка соответствия ГОСТ стандартам
└─ Лицензирование: мониторинг лицензий на используемое ПО

10. МЕТРИКИ И KPI ДЛЯ ОЦЕНКИ КАЧЕСТВА ПРОЦЕССОВ

10.1. Метрики качества кода (Code Quality Metrics)

1. Покрытие кодом тестами (Code Coverage)
   ├─ Целевое значение: ≥80% для критичного пути
   ├─ Критичный путь: доменная логика, обработка платежей
   ├─ Инструмент: PHPUnit с Xdebug
   ├─ SLA: не может снижаться с версией
   └─ Чекпоинт: проверяется перед merge в main

2. Цикломатическая сложность (Cyclomatic Complexity)
   ├─ Целевое значение: ≤10 для каждого метода
   ├─ Инструмент: PHPStan
   ├─ Красный флаг: методы с CC >15 требуют рефакторинга
   └─ Чекпоинт: при code review

3. Дублирование кода (Code Duplication)

   ├─ Целевое значение: <5% всего кода
   ├─ Инструмент: SonarQuber
   ├─ Действие: дублированный код требует рефакторинга
   └─ Рецидив: ≥10% дублирования блокирует merge

4. Технический долг (Technical Debt)
   ├─ Целевое значение: <5% (в SonarQube)
   ├─ Определение: объем работы для поддержки кода
   ├─ Мониторинг: ежемесячный анализ
   └─ Управление: выделение 20% спринта на техдолг

10.2. Метрики безопасности (Security Metrics)

1. Уязвимости в коде (Vulnerabilities)
   ├─ Целевое значение: 0 критичных / высоких уязвимостей
   ├─ Инструмент: OWASP ZAP + Snyk + SonarQube Security
   ├─ Частота сканирования: ежемесячно + перед каждым production deploy
   └─ SLA: критичная уязвимость → patch в течение 48 часов

2. Уязвимости в зависимостях (Dependency Vulnerabilities)
   ├─ Целевое значение: 0 уязвимостей в production dependencies
   ├─ Инструмент: Dependabot / Snyk
   ├─ Мониторинг: ежедневно
   └─ SLA: критичная CVE → обновление в течение 24 часов

3. OWASP Top 10 Покрытие
   ├─ Целевое значение: все 10 типов атак покрыты тестами
   ├─ Примеры: SQL injection, XSS, CSRF, Broken Auth и т.д.
   ├─ Тесты: в tests/Security/OWASPTest.php
   └─ Проверка: в каждом спринте

4. Покрытие ГОСТ стандартами
   ├─ ГОСТ Р 34.10-2012: подписи кода (реализовано)
   ├─ ГОСТ Р 34.11-2012: хеширование (реализовано)
   ├─ ГОСТ Р ИСО/МЭК 27001: политика ИБ (реализовано)
   └─ Проверка: ежеквартально

10.3. Метрики производительности (Performance Metrics)

1. Response Time (Время отклика)
   ├─ Целевое значение: p95 <500ms, p99 <1s
   ├─ Инструмент: Prometheus + Grafana + New Relic
   ├─ Мониторинг: real-time dashboard в Grafana

   ├─ SLA: если превышено -> incident
   └─ Рутинная проверка: load testing перед каждым major release

2. Throughput (Пропускная способность)
   ├─ Целевое значение: ≥100 RPS на один инстанс
   ├─ Инструмент: K6 / JMeter / Apache Bench
   ├─ Сценарий: 500 одновременных пользователей, 10 минут
   └─ Результат: документируется в Performance Report

3. Error Rate
   ├─ Целевое значение: <0.1% (99.9% success rate)
   ├─ Инструмент: Sentry + Prometheus
   ├─ Мониторинг: real-time dashboard
   └─ SLA: >1% error rate → incident

4. Database Query Performance
   ├─ Целевое значение: p95 <50ms
   ├─ Инструмент: PostgreSQL slow query log + Laravel Telescope
   ├─ Action: запросы >1s должны быть оптимизированы
   └─ Проверка: ежемесячно

10.4. Метрики надежности (Reliability Metrics)

1. Availability (Доступность)
   ├─ Целевое значение: 99.5% uptime (SLA)
   ├─ Расчет: (total_time - downtime) / total_time
   ├─ Допустимый downtime: ~5 часов в месяц
   ├─ Мониторинг: 24/7 health checks
   └─ Reporting: ежемесячный отчет

2. MTTR (Mean Time To Recovery)
   ├─ Целевое значение: <30 минут для kritical issues
   ├─ Definition: время от обнаружения до полного восстановления
   ├─ Инструмент: incident tracking system
   └─ Review: ежемесячно, выявление bottlenecks

3. MTBF (Mean Time Between Failures)
   ├─ Целевое значение: >720 часов (1 месяц между сбоями)
   ├─ Definition: среднее время между критичными инцидентами
   ├─ Мониторинг: логирование всех инцидентов
   └─ Action: анализ корневых причин

4. Deployment Success Rate

   ├─ Целевое значение: >95% успешных деплоев без rollback
   ├─ Мониторинг: каждый deployment логируется
   ├─ Rollback rate: <1% требует анализа
   └─ Улучшение: automated tests и automation

10.5. Метрики процесса разработки (Development Process Metrics)

1. Velocity (Скорость разработки)
   ├─ Определение: количество story points выполненных в спринт
   ├─ Целевое значение: стабильная velocity (±10% от baseline)
   ├─ Расчет: sum(story_points) for completed stories
   └─ Использование: планирование спринтов

2. Defect Escape Rate
   ├─ Определение: % багов обнаруженных в production из всех багов
   ├─ Целевое значение: <5%
   ├─ Расчет: (production_bugs) / (total_bugs)
   └─ Действие: если >10% -> пересмотр процесса тестирования

3. Deployment Frequency
   ├─ Целевое значение: 1 раз в неделю в production
   ├─ Инструмент: CI/CD pipeline metrics
   ├─ Гибкость: возможность hotfix deployment за несколько минут
   └─ Мониторинг: ежемесячный отчет

4. Lead Time (Время от идеи до production)
   ├─ Целевое значение: <2 недель для small features
   ├─ Расчет: дата merge в main - дата создания задачи
   └─ Улучшение: автоматизация + процесс оптимизация

10.6. Метрики удовлетворения пользователей (User Satisfaction Metrics)

1. NPS (Net Promoter Score)
   ├─ Целевое значение: >50 (excellent)
   ├─ Частота: ежеквартально (опрос пользователей)
   ├─ Вопрос: "Рекомендуете ли вы INTENSA SHOP друзьям?"
   └─ Action: анализ отзывов, улучшение функциональности

2. Support Ticket Resolution Time
   ├─ Целевое значение:
   │ ├─ Critical: <4 часов
   │ ├─ High: <24 часов
   │ ├─ Medium: <5 рабочих дней

│ └─ Low: <2 недель
├─ Мониторинг: в системе трекинга tickets
└─ Reporting: ежемесячно

3. User Retention
├─ Целевое значение: >80% месячная retention
├─ Расчет: users_active_this_month / users_active_prev_month
├─ Анализ: когда и почему пользователи уходят?
└─ Action: улучшение UX и функциональности

10.7. Целевые значения и мониторинг

Метрика	Целевое значение	Проверка
Code Coverage	≥80%	При merge
Cyclomatic Complexity	≤10	При code review
Duplication	<5%	При code review
Vulnerabilities	0 critical/high	Ежемесячно
Response Time p95	<500ms	Real-time dashboard
Uptime (SLA)	99.5%	Ежемесячно
MTTR (Critical)	<30 min	Ежемесячно
Deployment Success Rate	>95%	Ежемесячно
NPS Score	>50	Ежеквартально
Support Resolution Time	<24h (High)	Еженедельно

11. УПРАВЛЕНИЕ РИСКАМИ (RISK MANAGEMENT)

11.1. Процесс идентификации рисков

Риски идентифицируются на следующих этапах:

├─ Планирование спринта: технические и бизнес-риски
├─ Code review: потенциальные уязвимости и bugs
├─ Тестирование: недостатки в функциональности
├─ Развертывание: риски downtime и data loss
└─ Мониторинг: performance и security issues

11.2. Реестр рисков (Risk Register)

Все идентифицированные риски заносятся в реестр с:
├─ Описанием риска
├─ Вероятностью (High / Medium / Low)

├─ Влиянием (High / Medium / Low)
├─ Приоритетом (комбинация вероятности и влияния)
├─ Мерами смягчения (mitigation strategies)
└─ Ответственным лицом

11.3. Типовые риски и мониторинг

A) Технические риски
  ├─ Отказ базы данных: резервные копии + failover
  ├─ Security breach: сканирование + мониторинг + incident plan
  ├─ Performance degradation: load testing + мониторинг
  └─ Dependency vulnerabilities: сканирование + ежемесячный update

B) Бизнес-риски
  ├─ Потеря заказчика: поддержка + улучшение функциональности
  ├─ Конкуренция: инновационные функции
  └─ Изменение требований: процесс change management

C) Организационные риски
  ├─ Потеря ключевого разработчика: документирование + cross-training
  ├─ Задержки в разработке: резервные разработчики
  └─ Burnout команды: контроль нагрузки + отпуска