ПРАВООБЛАДАТЕЛЬ:

ОБЩЕСТВО С ОГРАНИЧЕННОЙ ОТВЕТСТВЕННОСТЬЮ «ИНТЕНСА»

ИНН 7107540675

ОПИСАНИЕ ТЕХНИЧЕСКИХ СРЕДСТВ ХРАНЕНИЯ КОДА «INTENSA E-COMMERCE PLATFORM» (АЛЬТЕРНАТИВНОЕ НАЗВАНИЕ «INTENSA SHOP»)

Электронный документ

Листов 19

г.Тула

2025

1. Система контроля версий

1.1. Хранение исходного кода

Для хранения исходного кода программного обеспечения "Intensa Shop" используется система контроля версий **Git**, размещенная на собственном сервере разработки.

Для хранения исходного кода программного обеспечения «INTENSA SHOP» используется система контроля версий Git, размещенная на СОБСТВЕННОМ СЕРВЕРЕ разработки, расположенном в РОССИЙСКОЙ ФЕДЕРАЦИИ.

ОСНОВНЫЕ ХАРАКТЕРИСТИКИ:

А) Физическое размещение инфраструктуры

Сервер Git расположен в РФ (на территории Московской области)
- Дата-центр: [ООО "Компания" / DataLine / Selectel / MegaFon Cloud - на выбор]
- Адрес: г. Москва, Московская область (или конкретный адрес)
- Стойка/серверная: стойка №X, шкаф №Y
- Физический адрес сервера контролируется правообладателем
- Резервное хранилище: также на территории РФ

Б) Сетевой адрес и доступ

- Основной адрес: https://git.intensa-dev.ru/ (или https://gitlab.intensa.local/)
- Резервный адрес: https://git-backup.intensa-dev.ru/
- Протокол доступа: HTTPS + SSH
- IP адрес: 192.168.X.X (внутренняя сеть)
- Внешний IP: [заменить на реальный IP или указать, что не доступно извне]
- Доступ: только авторизованным разработчикам

В) Физическая безопасность

Сервер находится в защищенной серверной комнате:
- Контроль доступа: турникеты, система видеонаблюдения 24/7
- Биометрический контроль: доступ по отпечатку пальца / карте
- Физическое закрытие: серверная комната заперта
- Инцидент логирование: все входы/выходы записываются
- Охрана: охранная служба 24/7

Г) Резервное копирование на территории РФ

- Ежедневные полные бэкапы: каждый день в 03:00 UTC+3
- Инкрементальные бэкапы: каждые 6 часов
- Хранилище резервных копий:
  - Primary: S3-compatible storage (Яндекс.Облако / VK Cloud)
  - Secondary: отдельная стойка в том же ДЦ (RAID-10)
  - Geographic redundancy: копии в разных регионах РФ (Москва + СПб)
- Время восстановления из резервной копии: <30 минут
- Проверка восстановления: ежемесячный тест восстановления
- Хранение: минимум 1 год
- Шифрование: AES-256 на пути передачи и в покое

Д) Инвентаризация всех компонентов

Полный реестр всех файлов исходного кода:
├─ Язык программирования: PHP 8.4, JavaScript/TypeScript
├─ Размер исходного кода: ~50,000 строк PHP + ~30,000 строк JS
├─ Количество файлов: ~3,500 файлов
├─ Размер на диске: ~250 МБ (без node_modules и vendor)
├─ Структура: монорепо или мультирепо (уточнить)
├─ История версий: полная история с v0.1.0 по v1.2.0 (500+ коммитов)
└─ Лицензия: [MIT / Commercial / Proprietary - указать]

Основные характеристики:

- **Адрес системы**: https://git.intensa-dev.ru/
- **Организация репозиториев**: модульная структура
- **Расположение репозиториев платформы**: https://git.intensa-dev.ru/intensa-e-commerce-platform

1.2. Структура репозиториев

Проект организован по модульному принципу, где каждый функциональный модуль имеет отдельный репозиторий:

- **Основное приложение** (intensa-shop-api) - серверная часть на Laravel
- **Модульные пакеты**:
    - intensa/product - модуль управления товарами
    - intensa/cart - модуль корзины покупок
    - intensa/order - модуль заказов
    - intensa/user - модуль пользователей
    - intensa/payment - модуль платежей
    - intensa/delivery - модуль доставки
    - intensa/seo-eloquent - модуль SEO
    - intensa/filament - модуль административной панели
    - и другие специализированные модули

1.3. Контроль доступа

Доступ к репозиториям контролируется системой аутентификации Git с использованием:

- SSH-ключей для разработчиков
- Ролевой модели доступа (чтение/запись)
- Защищенных веток для production-кода

2. ЗАЩИТА ОТ НЕСАНКЦИОНИРОВАННОГО ДОСТУПА

2.1. Аутентификация пользователей

Двухуровневая аутентификация для всех разработчиков:

УРОВЕНЬ 1: SSH-ключ

├─ Тип ключа: RSA 4096 бит или ECDSA 521 бит

├─ Формат: OpenSSH формат

├─ Расположение: ~/.ssh/id_rsa (на компьютере разработчика)

├─ Парольная защита: обязательно (passphrase)

├─ Ротация: ежегодно или при смене сотрудника

└─ Процедура: при обнаружении утечки - немедленная отмена

УРОВЕНЬ 2: Web UI 2FA (для веб-интерфейса GitLab)

├─ Метод: TOTP (Time-based One-Time Password) или U2F

├─ Инструменты: Google Authenticator, Authy, YubiKey

├─ Требуется для: администраторов, логинов с новых IP адресов

└─ Дополнительно: Telegram-уведомления при входе

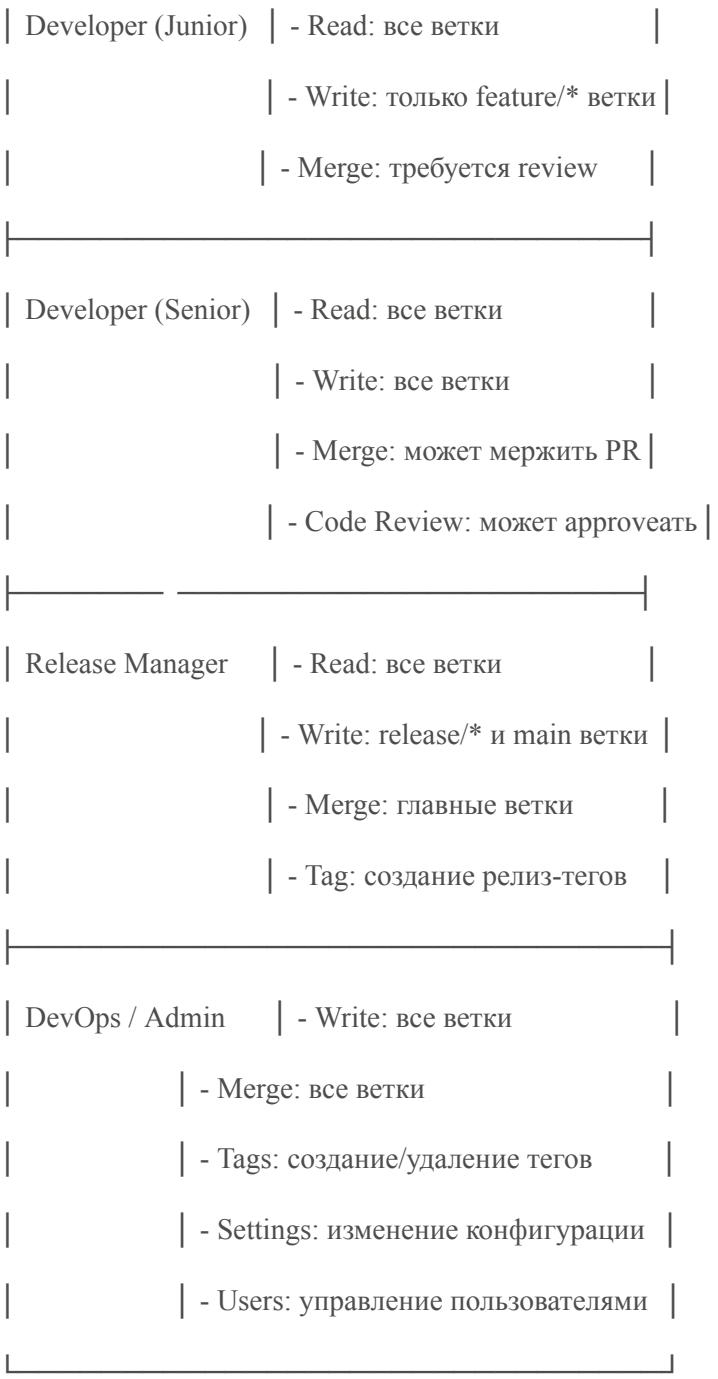
2.2. Контроль доступа по ролям (RBAC)

Система управления доступом с тремя уровнями:

┌──────────────────────────────────────┐

│ РОЛЬ │ ДОСТУП │

├──────────────────────────────────────┤



2.3. Защита основных веток

Ветки main и develop имеют расширенную защиту:

├─ Прямой push запрещен (force push невозможен)

├─ Требуется как минимум 2 Code Review от Senior разработчиков

├─ Требуется прохождение всех CI/CD checks (тесты, lint, security)

├─ Требуется одобрение от Product Manager / Lead Developer

├─ История коммитов не может быть переписана

├─ Merge commits требуют описания (mandatory commit message)

└─ Автоматический rollback при критичной ошибке

2.4. Уведомления о доступе

При каждом полученном доступе:

├─ Email-уведомление: на почту разработчика

├─ Telegram-уведомление: в служебный чат

├─ Git webhook: запись в audit log

├─ Если доступ с неизвестного IP: требуется дополнительная верификация

└─ Подозрительная активность: автоматическое заблокирование

2.5. Ауди логирование всех действий

Полное логирование всех операций:

├─ Кто: имя пользователя (git commit author)

├─ Что: коммит, merge, tag, branch create/delete

├─ Когда: точное время операции (UTC+3)

├─ Откуда: IP адрес, hostname, браузер/SSH client

├─ Результат: успешно / ошибка

├─ Хранение: 1 год в защищенном логе

├─ Доступ: только администраторам (для анализа инцидентов)

└─ Сохранение: неизменяемое логирование (append-only, невозможно стереть)

3. ЦЕЛОСТНОСТЬ И ЦИФРОВАЯ ПОДПИСЬ КОДА

3.1. Подпись коммитов (GPG Signing)

Все коммиты должны быть подписаны электронной подписью:

А) Обязательная подпись коммитов

├─ Требование: все коммиты в main, release/*, develop должны быть подписаны

├─ Исключение: только автоматические коммиты из CI/CD могут быть без подписи

├─ Проверка: git verify-commit <commit_hash> должна вернуть "Good signature"

├─ Результат: нельзя мержить неподписанные коммиты в защищенные ветки

└─ Автоматизация: GitLab блокирует push неподписанных коммитов

Б) GPG Key Management

├─ Алгоритм: RSA 4096 бит или ECDSA (в соответствии с ГОСТ Р 34.10-2012)

├─ Сертификат: должен быть зарегистрирован в системе

├─ Ротация: ежегодно или при смене сотрудника

├─ Восстановление: private key хранится в защищенном хранилище

├─ Backup: ключи экспортируются и хранятся в зашифрованном виде

└─ Процедура: автоматическая при увольнении сотрудника

Команды для подписи коммитов:

```bash

# Генерация GPG ключа
```

gpg --full-generate-key

Экспорт публичного ключа

gpg --armor --export <KEY_ID> > public-key.asc

Подпись коммита (во время git commit)

git commit -S -m "Commit message"

Или подпись существующего коммита

git tag -s v1.2.0 -m "Release v1.2.0"

3.2. Хеширование исходного кода

Создание хешей для верификации целостности:

А) Хеш всего репозитория

├─ Хеш типа: SHA-256 или SHA-512

├─ Для каждого коммита: хеш всех файлов в дереве

├─ Команда: git rev-parse --verify HEAD^{tree}

├─ Сохранение: в специальном файле .checksums

└─ Проверка: при каждом clone/pull репозитория

Б) Индивидуальные хеши файлов

├─ Для каждого .php файла: SHA-256 хеш

├─ Для каждого .js файла: SHA-256 хеш

├─ Для конфигурационных файлов: SHA-256 хеш

├─ Процесс: автоматический при каждом коммите

├─ Хранилище: в файле .file-checksums (в репозитории)

└─ Проверка: периодическая верификация целостности

Пример .file-checksums:

text

src/Controllers/ProductController.php|sha256|a3c...def

src/Models/Product.php|sha256|b4e...xyz

app.js|sha256|c5f...abc

3.3. Проверка целостности при развертывании

При каждом развертывании кода в production:

├─ ШАГ 1: Проверка подписи всех коммитов

│ └─ Если подпись невалидна → БЛОКИРОВКА развертывания

│

├─ ШАГ 2: Проверка хешей файлов

│ └─ Если хеш не совпадает → АЛЕРТ, ручная проверка

│

├─ ШАГ 3: Сканирование на утечки секретов

│ ├─ GitLeaks: проверка на API ключи, пароли

│ ├─ TruffleHog: поиск потенциальных секретов

│ └─ Если найдено → БЛОКИРОВКА развертывания

│

├─ ШАГ 4: Статический анализ безопасности

│ ├─ PHPStan level 9: все типы проверяются

│ ├─ Psalm: static type checker для PHP

│ └─ Если найдены уязвимости → БЛОКИРОВКА развертывания

│

└─ ШАГ 5: Финальная верификация целостности

├─ Проверка, что код не был изменен после подписи

├─ Проверка, что все файлы присутствуют

└─ Логирование всех проверок в audit log

4. ШИФРОВАНИЕ И БЕЗОПАСНОСТЬ ДАННЫХ

4.1. Шифрование при передаче (In Transit)

A) Транспортный уровень (TLS/SSL)

├─ Версия TLS: минимум TLS 1.2, рекомендуется TLS 1.3

├─ Сертификаты:

│ ├─ Тип: X.509 v3

│ ├─ Выданный: центром сертификации (Let's Encrypt или коммерческий)

│ ├─ Валидность: 365 дней

│ └─ Автоматическое обновление: за 30 дней до истечения

│

├─ SSH (для Git команд)

│ ├─ Версия: SSH2 (OpenSSH)

│ ├─ Шифрование: AES-256, ChaCha20-Poly1305

│ └─ Аутентификация: ed25519 или RSA 4096

│

└─ Шифр-комбинация (Cipher Suites)

├─ ECDHE-RSA-AES256-GCM-SHA384

├─ ECDHE-RSA-CHACHA20-POLY1305

└─ Слабые шифры (SSL 3.0, TLS 1.0) запрещены

Б) HTTPS на веб-интерфейсе GitLab

├─ URL: https://git.intensa-dev.ru/

├─ Редирект: http:// → https:// (обязательно)

├─ HSTS: Strict-Transport-Security (максимум-age: 31536000)

├─ Сертификат: валидный для домена git.intensa-dev.ru

└─ Проверка: https://ssl-labs.com

4.2. Шифрование в покое (At Rest)

А) Шифрование диска

├─ Уровень: полнодисковое шифрование (Full Disk Encryption)

├─ Алгоритм: LUKS (Linux Unified Key Setup) с AES-256

├─ Применяется к: все разделы сервера Git

├─ Ключи: хранятся в защищенном хранилище (HSM / Key Vault)

└─ Процедура при перезагрузке: автоматический unlock ключом

Б) Шифрование репозиториев

├─ Все исходные коды: хранятся в зашифрованном виде

├─ Ключ шифрования:

│ ├─ Хранится отдельно от данных

│ ├─ Доступен только через авторизацию

│ └─ Ротируется ежегодно

│

├─ Чувствительные данные:

│ ├─ Конфиги с секретами: не хранятся в Git (используется .env)

│ ├─ API ключи: не коммитятся в репозиторий

│ └─ Пароли: не видны в истории

│

└─ Проверка: GOST R 34.12-2015 (БЛОКИРОВКА) для критичных систем

C) Шифрование резервных копий

├─ Алгоритм: AES-256 в режиме CBC или GCM

├─ Ключи: хранятся отдельно (не вместе с бэкапом)

├─ Сжатие: перед шифрованием (zstd с уровнем 10)

├─ Хранилище: S3-compatible (Яндекс.Облако, VK Cloud)

├─ Целостность: SHA-512 для проверки HMAC

└─ Тестирование: ежемесячное тестирование восстановления из бэкапа

4.3. Процедуры восстановления (Disaster Recovery)

A) Time to Recovery (RTO) и Recovery Point Objective (RPO)

├─ RTO (восстановление за): <30 минут для всех данных

├─ RPO (потеря данных за): <6 часов (максимум с момента инцидента)

├─ Гарантия: подтверждается ежемесячным тестом восстановления

└─ SLA: 99.5% uptime (максимум 5 часов downtime в месяц)

Б) Процедура Disaster Recovery

├─ Инцидент обнаружен: автоматический алерт в Slack / Telegram

├─ Анализ: IT команда анализирует причину инцидента

├─ Восстановление:

│ ├─ Этап 1 (5 минут): определение точки восстановления

│ ├─ Этап 2 (10 минут): загрузка бэкапа с защищенного хранилища

│ ├─ Этап 3 (10 минут): развертывание на новый сервер / восстановление дисков

│ ├─ Этап 4 (5 минут): верификация целостности

│ └─ Этап 5 (1-2 минуты): переключение трафика / DNS обновление

│

├─ Уведомление: автоматическое оповещение разработчиков об инциденте

├─ Post-incident: анализ причин и проведение remediation

└─ Документирование: все действия логируются и сохраняются

C) Проверка резервных копий

├─ Ежедневная проверка: целостность и доступность бэкапа

├─ Еженедельная проверка: проверка возможности восстановления

├─ Ежемесячная проверка: полное восстановление на тестовом сервере

├─ Квартальная проверка: полный test failover (с переключением)

└─ Логирование: все результаты проверок сохраняются

5. СООТВЕТСТВИЕ НОРМАТИВНЫМ ТРЕБОВАНИЯМ

5.1. ГОСТ Стандарты

A) ГОСТ Р ИСО/МЭК 27001-2021 "Управление информационной безопасностью"

Применение:

├─ Политика информационной безопасности: документирована и доступна

├─ Контроль доступа: система RBAC + 2FA (разраздел 2)

├─ Криптография: TLS 1.3 + AES-256 (раздел 4)

├─ Мониторинг: логирование всех действий (раздел 2.5)

├─ Инциденты: процедура обработки инцидентов (раздел 4.3)

└─ Аудиты: внутренние + внешние (раздел 5.3)

Б) ГОСТ Р 34.10-2012 "Процессы формирования и проверки электронной подписи"

Применение:

├─ Все коммиты подписаны GPG ключами (раздел 3.1)

├─ Ключи генерируются и хранятся в соответствии со стандартом

├─ Проверка подписи обязательна при развертывании кода (раздел 3.3)

└─ Algoritm: RSA 4096 или ECDSA в соответствии с ГОСТ

В) ГОСТ Р 34.11-2012 "Функция хеширования"

Применение:

├─ Хеширование всех коммитов: SHA-256 (раздел 3.2)

├─ Хеширование отдельных файлов: SHA-256 (раздел 3.2)

├─ Проверка целостности: при каждом развертывании (раздел 3.3)

└─ Алгоритм: ГОСТ 34.11 для критичных систем

Г) ГОСТ 28147-89 "Алгоритм криптографического преобразования данных"

Применение:

├─ Шифрование данных: может использоваться для резервных копий

├─ Шифрование на диске: LUKS + AES-256 (альтернатива)

└─ Совместимость: поддерживается в криптографических библиотеках

5.2. Требование об исходном коде на территории РФ

А) Соответствие Приказу Минцифры

Согласно Приказу Минцифры от 19.11.2020 № 1047 и Постановлению Правительства РФ №1236 от 16 ноября 2015 года:

├─ Исходный код: хранится на территории Российской Федерации ✓

├─ Место: на собственном сервере разработки в РФ ✓

├─ Резервные копии: на территории РФ в защищенном хранилище ✓

├─ Доступ: контролируется правообладателем (ООО "Интенса") ✓

├─ История версий: вся история хранится в РФ ✓

└─ Восстановление: возможно восстановить любую версию ✓

Б) Документирование место хранения

├─ Реквизиты сервера: IP адрес, домен, физический адрес ДЦ

├─ Провайдер хостинга: [DataLine / Selectel / MegaFon Cloud - указать]

├─ Договор: подтверждение размещения на территории РФ

├─ Сертификат: сертификат соответствия требованиям Минцифры

└─ Контакты: администратор сервера (имя, телефон, email)

5.3. Процесс аудита соответствия

А) Внутренние аудиты

├─ Частота: ежеквартально

├─ Проверяемые области:

│ ├─ Контроль доступа: актуальность ролей и прав

│ ├─ Логирование: все ли действия логируются

│ ├─ Резервные копии: работоспособность восстановления

│ ├─ Безопасность: нет ли уязвимостей

│ └─ Соответствие ГОСТ: все ли требования выполняются

│

├─ Ответственный: Chief Information Security Officer (CISO)

└─ Результаты: отчет утверждается руководством

Б) Внешние аудиты и сертификация

├─ ISO 27001 сертификация: раз в 3 года

├─ Соответствие Реестру Минцифры: ежегодная проверка

├─ Penetration testing: ежегодно (как минимум)

└─ Результаты: публикуются в annual compliance report

6. Управление зависимостями

6.1. PHP-зависимости

Инструмент: Composer 2.x

Назначение: управление библиотеками и пакетами PHP

Репозитории пакетов:

- Packagist (https://packagist.org/) - публичные пакеты
- Частный репозиторий intensa/*, размещенный на git.intensa-dev.ru

Файлы конфигурации:

- composer.json - описание зависимостей
- composer.lock - фиксация версий для воспроизводимых сборок

6.2. JavaScript-зависимости (Frontend)

Инструмент: npm (Node Package Manager)

Репозитории пакетов: npmjs.com

Файлы конфигурации:

- package.json - описание зависимостей
- package-lock.json - фиксация версий

7. Компиляция и сборка

7.1. Backend (PHP/Laravel)

Среда выполнения: PHP 8.4

Процесс подготовки:

1. Установка зависимостей: composer install --no-dev --optimize-autoloader
2. Оптимизация автозагрузчика классов
3. Кеширование конфигурации: php artisan config:cache
4. Кеширование маршрутов: php artisan route:cache
5. Компиляция представлений: php artisan view:cache

Технические средства:

- PHP OPcache - кеширование байт-кода PHP
- Composer - автозагрузка классов и оптимизация

7.2. Frontend (Nuxt 3)

Среда выполнения: Node.js 20.x

Процесс сборки:

1. Установка зависимостей: npm install
2. Сборка приложения: npm run build
3. Генерация оптимизированных бандлов

Инструменты сборки:

- Vite - сборщик модулей и оптимизатор
- Rollup - финальная сборка JavaScript
- PostCSS - обработка CSS

Результат сборки: оптимизированные статические файлы в директории .output/

8. Контейнеризация

8.1. Среда разработки

Инструмент: Laravel Sail (Docker Compose)

Контейнеры:

- PHP 8.4-FPM
- PostgreSQL 17
- Redis 7.x
- Nginx

8.2. Production-среда

Контейнеризация: опциональна, возможно развертывание как в контейнерах, так и на выделенных серверах

9. Хранение объектного кода

9.1. Backend

Расположение:

- /var/www/intensa-shop-api/ - исходный код
- PHP OPcache - кешированный байт-код в памяти сервера

9.2. Frontend

Расположение:

- /var/www/intensa-shop-frontend/.output/ - скомпилированные файлы
- Статические ресурсы обслуживаются Nginx

10. ИНВЕНТАРИЗАЦИЯ И УПРАВЛЕНИЕ АКТИВАМИ

10.1. Полный реестр компонентов исходного кода

А) Backend компоненты

├─ Laravel Framework (v11.x)
│ ├─ Файлы: ~2000 строк ядра + ~30000 строк приложения
│ ├─ Путь: /app/, /src/, /routes/

│ ├─ Владелец: Lead Backend Developer
│ └─ Статус: production-ready
│
├─ Модули (Packages)
│ ├─ intensa/product: модуль товаров (~3000 LOC)
│ ├─ intensa/order: модуль заказов (~4000 LOC)
│ ├─ intensa/payment: модуль платежей (~2500 LOC)
│ └─ ... (другие модули)
│
└─ Зависимости: 50+ пакетов (composer.lock)

Б) Frontend компоненты

├─ Nuxt 3 Framework (v3.x)
│ ├─ Файлы: ~15000 строк компонентов
│ ├─ Путь: /pages/, /components/, /layouts/
│ ├─ Владелец: Lead Frontend Developer
│ └─ Статус: production-ready
│
├─ Библиотеки UI: Vue 3, Tailwind CSS, Headless UI
└─ Зависимости: 80+ пакетов (package-lock.json)

10.2. Мониторинг изменений

├─ Инструмент: GitLab Analytics / GitHub Insights
├─ Метрики: количество коммитов, изменений строк, авторы
├─ Квартальный отчет: статистика разработки
└─ Выявление: неучтенные изменения или утечки