**Процессы поддержки жизненного цикла ПО, устранение неисправностей и совершенствование. Персонал и роли**

**1. Назначение документа**

Документ описывает процессы, обеспечивающие полный жизненный цикл системы мониторинга транспортных потоков (далее — Система): планирование, разработку, тестирование, выпуск, эксплуатацию, сопровождение, устранение неисправностей, улучшения и вывод из эксплуатации. Также определяются роли, ответственность, требования к персоналу и регламенты взаимодействия.

**2. Нормативные ссылки и термины**

* ITIL v4 (инциденты/проблемы/изменения/релизы/знания).
* ГОСТ 34/19 (при необходимости).
* Внутренние регламенты ИБ и эксплуатации.
* Термины: **Инцидент** (нарушение или угроза доступности/функционала), **Проблема** (первопричина), **Запрос на изменение (RFC)**, **Релиз**, **Хотфикс**, **SLA/SLO**.

**3. Жизненный цикл ПО (сквозной процесс)**

1. **Планирование**: формирование требований (backlog), оценка, приоритезация.
2. **Аналитика и проектирование**: уточнение бизнес-правил, дизайн API/БД/интерфейсов.
3. **Разработка**: ветвление в GitLab, код-ревью, статический анализ, юнит-тесты.
4. **Тестирование**: функциональное/регресc/нагрузочное/безопасность на staging.
5. **Релиз/деплой**: через CI/CD, миграции БД, проверка отката.
6. **Эксплуатация**: мониторинг, резервное копирование, аудит.
7. **Сопровождение**: инциденты, проблемы, запросы пользователей, улучшения.
8. **Совершенствование**: ретроспективы, пост-мортемы, KPI DevOps (Lead Time, DF, MTTR).
9. **Вывод из эксплуатации**: архивирование данных, выключение интеграций, отзыв доступов.

Ответственность по стадиям фиксируется в RACI-матрицах (см. раздел 11).

**4. Управление требованиями и изменениями**

* Источник требований: Заказчик, продукт-оунер, эксплуатация, аудит.
* Артефакты: **Epic/Feature/User Story** в Jira, описания, макеты, критерии приёмки.
* Оценка: планирование спринта, ч/часы.
* **RFC (Request For Change)**: карточка изменения с оценкой рисков, планом релиза/отката, влиянием на ИБ и производительность.
* Классы изменений: стандартные (шаблонные), нормальные (CAB-утверждение), срочные (ECAB/аварийные).

**5. Ветвление, код-ревью, качество**

* Стратегия: **main** (production)
* Требования к merge request: прохождение пайплайна (build/test/deploy).
* Secrets: хранение вне кода (CI/CD variables, Vault), ревью .env.example.

**6. Сборка, тестирование и релизы (CI/CD)**

* GitLab CI: стадии build → test → deploy.
* Стенды: dev (локально через Sail), staging (полная проверка), production.
* Версионирование: **SemVer** MAJOR.MINOR.PATCH.
* Окна релизов: плановые (например, еженедельно, вторник 18:00–20:00), аварийные — по решению ECAB.
* Процедуры:
  + php artisan migrate --force / проверка migrate:rollback.
  + Тёплый старт воркеров очередей, прогрев кэшей config:cache, route:cache.
  + Smoke-тесты после деплоя, контрольные чек-листы.

**7. Эксплуатация: мониторинг, резервное копирование, безопасность**

**Мониторинг/логирование**

* Технические метрики (CPU/RAM/IO, пулы PHP-FPM), статус воркеров очередей.
* Логи приложения, web-сервера, БД;

**Резервное копирование**

* БД: ежедневные полные + часовые инкрементальные, хранение 60 дней, off-site копии.
* Медиа: snapshot/репликация/object storage.
* Ежемесячные тестовые восстановление на staging.

**Безопасность**

* HTTPS, актуальные TLS-настройки.
* Обновления зависимостей (ежемесячно/по CVE ASAP).
* RBAC, принцип наименьших прав.
* Аудит действий (Spatie ActivityLog), ротация ключей/токенов.

**8. Управление инцидентами, проблемами и сервисными запросами**

**Каналы фиксации:** GitLab Issues (или Service Desk), e-mail, горячая линия/Telegram.  
**Классификация инцидентов:**

| **Класс** | **Пример** | **Цель восстановления (SLA)** | **Первичное реагирование** |
| --- | --- | --- | --- |
| P1 (Критический) | Полный простой прод. | ≤ 2 ч | ≤ 15 мин (24×7) |
| P2 (Высокий) | Критичный модуль/документооборот | ≤ 8 ч | ≤ 30 мин (раб. время) |
| P3 (Средний) | Ошибка с обходным решением | ≤ 3 раб. дня | ≤ 4 ч |
| P4 (Низкий) | Минорный дефект/UI | Планово | ≤ 1 раб. день |

**Процесс:**

1. Регистрация и приоритезация.
2. Диагностика, временное решение (workaround) при необходимости.
3. Восстановление сервиса и подтверждение с Заказчиком.
4. Пост-мортем (для P1/P2) с планом предотвращения повторов.

**Проблемы (Problem Management):** анализ первопричин (RCA), задачи на исправление/архитектурные изменения, контроль закрытия.

**Сервисные запросы:** доступы, отчёты, выгрузки — по каталогу услуг с целевыми сроками.

**9. Управление знаниями и документацией**

* Репозиторий знаний: GitLab, Confluence.
* Шаблоны: чек-листы релиза, инструкции деплоя/отката, FAQ по ролям.
* Обновление: с каждым релизом — запись изменений (CHANGELOG), инструкция «Что нового».
* Стандарты оформления: версии документов, владельцы разделов.

**10. Улучшения и технический долг**

* Канал «Предложения по улучшениям» (Product backlog).
* Ретроспектива спринта: сбор метрик (Lead Time, Deployment Frequency, MTTR, Change Failure Rate).
* Квартальный техдолг-спринт: рефакторинг, обновление стеков, оптимизация запросов/индексов, улучшение DX.

**11. Персонал, роли и ответственность (RACI)**

**Ключевые роли:**

* **Product Owner (Заказчик/бизнес-владелец):** приоритезация, принятие релизов.
* **Технический лидер/Архитектор:** решения по архитектуре, code standards, сложные инциденты (L3).
* **Backend разработчик (Laravel):** фичи/исправления, тесты, миграции.
* **Frontend разработчик (Vue):** SPA/UX, сборки, интеграция API.
* **DevOps/SRE:** CI/CD, мониторинг, бэкапы, инфраструктура, DR-план.
* **QA инженер:** план/сценарии тестирования, тест-артефакты, отчёт о качестве.
* **Системный администратор:** ОС/сети/сертификаты, доступы, резервное копирование.
* **Поддержка L1/L2:** приём обращений, первичная диагностика, эскалация.

**Минимальный состав (для стабильной эксплуатации):**

* 1 Tech Lead (совмещает Senior Backend),
* 1 Backend, 1 Frontend,
* 1 QA,
* 1 DevOps/SRE (частичная занятость допустима),
* 1 L1/L2 (или оператор Service Desk).

**RACI (пример, укороченный):**

| **Процесс/задача** | **PO** | **TL/Arch** | **Dev** | **QA** | **DevOps** | **L1/L2** |
| --- | --- | --- | --- | --- | --- | --- |
| Приоритизация backlog | A | C | I | I | I | I |
| Реализация фичи | C | A | R | C | I | I |
| Релиз в prod | I | A | R | C | R | I |
| Инцидент P1 | I | A | R | I | R | R |
| RCA/Проблема | I | A | R | C | C | I |
| Бэкапы/DR-тест | I | C | I | I | A/R | I |

(A — Accountable, R — Responsible, C — Consulted, I — Informed)

**12. Доступы и управление учётными записями**

* Процесс Joiner/Mover/Leaver: заявки на доступ, согласование, аудит.
* Роли в системе соответствуют RBAC; минимум прав для каждой роли.
* Обязательная 2FA для административных доступов (GitLab/серверы).
* Регулярная ревизия доступов (ежеквартально).

**13. Управление конфигурациями и окружениями**

* Конфигурационные единицы: версии приложений, Docker-образы, .env параметры, схемы БД, планы бэкапов.
* CMDB (упрощённо): таблица в Wiki/Issues для связи компонент и ответственных.
* Окружения: dev (локально, Sail), staging (полное зеркало), prod.
* Все изменения конфигураций — через RFC, фиксируются в репозитории/CI variables.

**14. План непрерывности и аварийное восстановление (DR)**

* RPO/RTO (⟨например, RPO ≤ 1 час, RTO ≤ 4 часа для prod⟩).
* Сценарии: отказ БД, потеря узла web, повреждение медиа-хранилища.
* План действий: переключение на реплику/восстановление из бэкапа, redeploy образов, проверка целостности.
* Ежеквартальные учения.

**15. Коммуникации и отчётность**

* Каналы: корпоративный email, мессенджер (Telegram/Slack), доска задач (GitLab/Jira).
* Регулярные отчёты: еженедельный статус проекта, ежемесячный отчёт эксплуатации (SLA, инциденты, изменения, бэкапы).
* Пост-мортемы для P1/P2 публикуются в базе знаний.

**16. Вывод из эксплуатации**

* План деактивации интеграций и уведомлений.
* Архивирование БД и медиа в соответствии с политикой хранения.
* Отзыв сертификатов/доступов, удаление секретов.
* Итоговый отчёт и закрытие документации.