Ай-ТИ Фреш

ТЕХНИЧЕСКИЙ РАЗБОР

Легаси-код: как обновить систему, не остановив бизнес

Риски устаревших систем, типовые сценарии отказов и безопасная стратегия поэтапного рефакторин...

Июль 2026

itfresh.ru · ИТ-аутсорсинг для юридических лиц

О ЧЁМ ЭТОТ РАЗБОР

Суть проблемы

Почти в каждой компании есть система, на которой держится бизнес: самописная учётка, старый сайт с заказами, база на снятой с поддержки платформе. Автор кода уволился, документации нет, любая правка ломает соседнее. Обновлять страшно, переписывать дорого, а оставить как есть — копить уязвимости и однажды встать вместе с продажами.

Почему это важно бизнесу

- Простой ключевой системы останавливает продажи и обслуживание клиентов — потери растут с каждым часом
- Уязвимости в устаревшем ПО — прямой путь к взлому, утечке персональных данных и штрафам по 152-ФЗ
- Поддержка легаси съедает бюджет: без тестов и CI/CD каждая правка требует ручной регрессии и согласований, цикл изменений растягивается, а разви…
- Уход единственного «носителя знаний» превращает систему в чёрный ящик, который никто не рискует трогать
- Импортозамещение и прекращение поддержки западного ПО делают модернизацию срочной, а не «когда-нибудь»

ЧТО МЫ ЗАКЛАДЫВАЕМ

Ключевые параметры реализации

31.12.2025

окончание поддержки безопасности PHP 8.1: версии ≤ 8.1 больше не получают патчи

php.net, Supported Versions, 2026

8.2 → 31.12…

PHP 8.2 — только security-fix до конца 2026; актуальные ветки 8.3, 8.4, 8.5

php.net, Supported Versions, 2026

4 года

цикл поддержки ветки PHP: 2 года активной + 2 года только безопасность

php.net, Release Cycle Update, 2024

≤ 10

порог цикломатической сложности метода в аудите; выше — кандидат на рефакторинг

Наш регламент статического анализа кода

70%+

целевое покрытие характеризационными тестами критичной бизнес-логики

Наш стандарт безопасного рефакторинга

1 раб. день

целевое время восстановления ключевой системы из бэкапа (RTO)

Наш регламент резервного копирования и DR

РЕАЛИЗАЦИЯ 1 ИЗ 3

Миграция «одним рывком» без параллельного контура и отката

Что настраиваем

Сценарий из практики: перенос ядра учётной системы за одни выходные, старую платформу гасят сразу

Как мы это делаем

1. Пятница: данные из легаси-СУБД переносятся в новую систему единой пакетной выгрузкой, исходную платформу отключают в ту же ночь
2. Понедельник: часть пользователей не может войти — не смигрировали права и активные сессии, кэш прав отдаёт записи чужих учёток
3. Отката нет: исходную БД уже вывели из эксплуатации, восстановление из бэкапа и ручная сверка занимают недели
4. По итогам разбора: нагрузочное и негативное тестирование новой системы не проводилось, критичные сценарии входа и прав не проверяли

РЕЗУЛЬТАТ

Недели частичного простоя, ручная сверка расхождений в данных, потеря доверия пользователей. Стоимость аварийного восстановления многократно превысила экономию от быстрой миграции.

КЛЮЧЕВОЙ НЮАНС

Замена работающей системы «большим взрывом» без нагрузочного тестирования и отработанного отката — худший сценарий. Поэтапная миграция за фасадом дольше, но оставляет путь назад на каждом шаге.

РЕАЛИЗАЦИЯ 2 ИЗ 3

Легаси-планировщик не выдержал пиковой нагрузки

Что настраиваем

Сценарий из практики: система планирования ресурсов родом из 1990-х, рассчитанная на типовой поток заявок

Как мы это делаем

1. Внешний форс-мажор вызывает лавину перестановок — объём изменений на порядок выше расчётного
2. Однопоточный алгоритм пересчёта не масштабируется: очередь задач растёт быстрее, чем обрабатывается, состояние ресурсов рассинхронизируется
3. Система входит в каскадный сбой; ручное восстановление занимает дни, пока системы на современном стеке держат ту же нагрузку
4. Разбор показывает: пиковый профиль нагрузки под систему никогда не тестировался

РЕЗУЛЬТАТ

Многодневный отказ ключевой функции в самый пиковый период, ручной разбор состояния, прямые потери и репутационный ущерб. Накопленный техдолг превратился в убытки за считаные дни.

КЛЮЧЕВОЙ НЮАНС

Легаси «работает», пока нагрузка типовая. Пиковый сценарий, под который систему никто не проверял, превращает техдолг в прямые убытки. Нагрузочное тестирование по реальному пиковому профилю обязательно.

РЕАЛИЗАЦИЯ 3 ИЗ 3

Мёртвый код и ручной деплой: отказ за считаные минуты

Что настраиваем

Сценарий из практики: ручной выкат обновления на кластер из восьми серверов

Как мы это делаем

1. Обновление раскатывают по узлам вручную; один сервер пропускают — на нём остаётся прошлая версия модуля
2. На пропущенном узле оживает давно неиспользуемый код: его флаг активации в новой версии переиспользован под другой смысл
3. За десятки минут узел с рассинхронизированной логикой генерирует лавину ошибочных операций
4. Отката и предполётных проверок нет — обнаружить расхождение версий и остановить поток за минуты невозможно

РЕЗУЛЬТАТ

Отказ с убытком, превысившим собственный капитал, за считаные десятки минут. Ни автоматического выката, ни контроля версий на узлах, ни быстрого отката — расхождение вскрылось слишком поздно.

КЛЮЧЕВОЙ НЮАНС

Неудалённый мёртвый код плюс ручной деплой без проверок — мина замедленного действия. Автоматизация выката с контролем версий на всех узлах и чистка легаси — не перфекционизм, а страховка капитала.

ЧТО МЫ ВЫУЧИЛИ НА ПРАКТИКЕ

Подводные камни

- ✕ **Переписать всё с нуля**
  Big-bang-замена почти всегда срывает сроки: старая система деградирует, новая не готова, бизнес платит за обе одновременно.
- ✕ **Рефакторинг без тестов**
  Без фиксации текущего поведения (характеризационных тестов) каждое изменение — лотерея с регрессиями на боевом сервере.
- ✕ **Ручной деплой на прод**
  FTP и «накатывание руками» — человеческий фактор без возможности отката: рассинхрон версий между серверами приводит к отказу за минуты, а откатиться…
- ✕ **«Работает — не трогаем»**
  Откладывание до отказа: уязвимости копятся, стоимость модернизации растёт, а замена в итоге происходит в аварийном режиме.
- ✕ **Всё держится на одном человеке**
  Уход единственного знающего разработчика превращает систему в чёрный ящик — цена и риск любых правок вырастают кратно.
- ✕ **Рефакторить всё подряд**
  Без матрицы «бизнес-ценность × риск × частота изменений» бюджет заканчивается раньше, чем появляется видимый результат.
- ✕ **Модернизация без спонсора сверху**
  Без владельца бюджета и полномочий проект глохнет на середине: финансирование урезают на первом же квартальном пересмотре, а команда возвращается к «…
- ✕ **Новый код без новой культуры**
  Без code review, CI/CD и обучения команды переписанная система накапливает тот же технический долг за год-два.

УРОВНИ РЕАЛИЗАЦИИ

Как правильно

МИНИМУМ

- Инвентаризация: какие самописные системы критичны и на каком ПО работают
- Проверенные резервные копии кода и баз данных с тестом восстановления
- Изолировать легаси от интернета, закрыть известные уязвимости

НОРМАЛЬНО

- Аудит кода: метрики сложности, дубликаты, статический анализ уязвимостей
- Матрица приоритизации модулей: ценность × риск × частота изменений
- Характеризационные тесты на ключевую бизнес-логику до любых правок
- CI/CD вместо ручного деплоя: автоматическая сборка, тесты, откат

ХОРОШО

- Паттерн Strangler Fig: поэтапная замена модулей за reverse proxy
- Покрытие тестами 70%+ и блокировка релиза при расхождении поведения
- План перехода на поддерживаемый стек с учётом импортозамещения
- Передача знаний: код-ревью, парная работа, документация архитектуры

ПРОВЕРЬТЕ У СЕБЯ

Чек-лист самопроверки

- [ ] Знаете ли вы, какие самописные системы напрямую влияют на выручку вашей компании?
- [ ] Есть ли у вас системы на ПО, снятом с поддержки: PHP ≤ 8.1 (EOL 31.12.2025), Windows Server 2012/2012 R2, устар...
- [ ] Есть ли актуальная документация и минимум два человека, понимающие код ключевой системы?
- [ ] Покрыта ли критичная бизнес-логика автоматизированными тестами?
- [ ] Автоматизирован ли деплой на боевой сервер и предусмотрен ли быстрый откат?
- [ ] Сможете ли вы восстановить ключевую систему из резервной копии за один рабочий день?
- [ ] Знаете ли вы, сколько часов в месяц команда тратит на «тушение пожаров» в старых системах?
- [ ] Проверяли ли вы систему под пиковой нагрузкой (сезон, распродажа, отчётный период)?
- [ ] Есть ли утверждённый план поэтапной модернизации с бюджетом и приоритетами?

Если хотя бы на два вопроса ответ «нет» или «не знаю» — тема требует внимания.

ЧТО МЫ ДЕЛАЕМ

Как поможет ITFresh

ITFresh — ИТ-аутсорсинг для юридических лиц до 50 рабочих мест в Москве и области. 15+ лет практики, собственная инфраструктура в дата-центре МТС (8 серверов Dell Xeon Platinum).

- Аудит легаси-систем: версии ПО, уязвимости, метрики кода, карта рисков с оценкой для бизнеса
- План поэтапной модернизации: приоритизация модулей, выбор стека, оценка бюджета и сроков
- Внедрение резервного копирования, тестового контура и CI/CD без остановки работы компании
- Сопровождение на время миграции: мониторинг, обновления, поддержка старой и новой систем

15+
лет в ИТ-поддержке

50
рабочих мест — наш профиль

МТС
дата-центр, Москва

КОНТАКТЫ

Обсудить вашу задачу

Сайт	**itfresh.ru**
Телефон	**+7 903 729-62-41**
Telegram	**@ITfresh_Boss**

Бесплатно посмотрим вашу инфраструктуру по этому чек-листу и скажем, где тонко — без обязательств.

itfresh.ru

Техническая база

01 PHP: Supported Versions — жизненный цикл и EOL веток (php.net — 2026)

02 PHP: Release Cycle Update (2 года активной + 2 года безопасности) (wiki.php.net — 2024)

03 Azure Architecture Center: Strangler Fig pattern (learn.microsoft.com — 2024)

04 Microsoft Product Lifecycle: Windows Server (даты EOL) (learn.microsoft.com — 2026)

05 GitLab Docs: CI/CD пайплайны, сборка и откат (docs.gitlab.com — 2025)

06 nginx documentation: reverse proxy (ngx_http_proxy_module) (nginx.org — 2025)

07 ФЗ-152 «О персональных данных» (действующая редакция) (pravo.gov.ru — 2024)

08 Единый реестр российского ПО (Минцифры) (reestr.digital.gov.ru — 2026)

09 Наш шаблон: аудит легаси — карта рисков и метрики кода (itfresh.ru — 2025)

10 Наш регламент: резервное копирование и тест восстановления (DR) (itfresh.ru — 2025)

Основано на официальной документации продуктов и нашей практике внедрения.