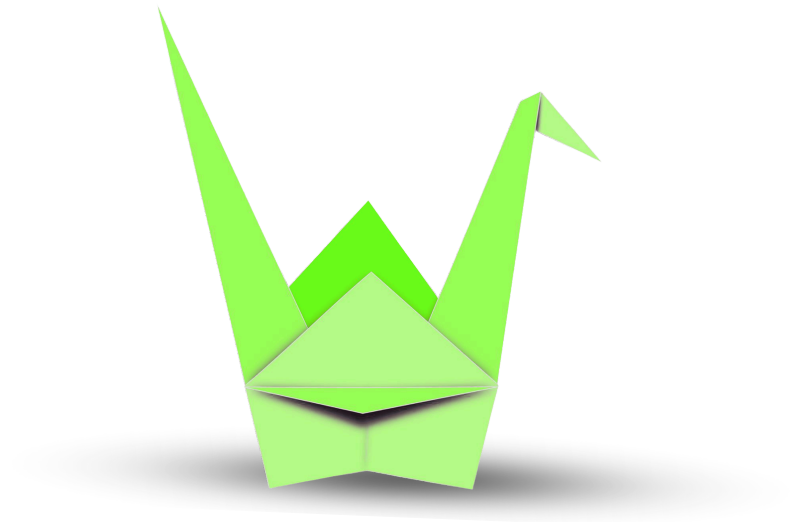
Технические требования для запуска проекта Diamond OA

Системная архитектура для автоматизированных издательских сред, ориентированных на метаданные



zeba.academy

Zeba Academy

Впервые опубликовано: Zeba Academy и Zeba Books.

Год издания: 2026

Серия: Zeba Academy Blueprints -- Технические директивы суверенных систем (Sovereign Systems Technical Directives)

Цель серии: Этот цикл директив разработан для борьбы с «энштификацией» и избыточной перегруженностью современного ПО. Наша цель — вернуть суверенный контроль над нашими системами, сократив разрыв между глубокой академической теорией и критически важной промышленной реализацией. Мы убеждены, что программное обеспечение должно быть быстрым, долговечным и, прежде всего, понятным для его владельца и пользователя.

Главный архитектор: Суфян бин Узайр, сертифицированный Google Cloud Professional DevOps-инженер.

Основной стек: Linux, Rust, Zig, C++, Flutter и PHP.

Email: hello@zeba.academy

Технические требования для запуска проекта Diamond OA

Системная архитектура для автоматизированных издательских сред, ориентированных на метаданные

Определение системы и операционные ограничения

Система Diamond OA должна функционировать как полностью автоматизированная, нейтральная с точки зрения затрат издательская инфраструктура с минимальным человеческим вмешательством. Она требует детерминированных рабочих процессов и структурированной обработки данных для обеспечения согласованности и масштабируемости. Система также должна поддерживать высокий уровень интероперабельности с внешними научными сервисами при низких накладных расходах на инфраструктуру.

Проектирование детерминированного издательского движка с нулевой доходностью

Издательская система Diamond Open Access (OA) должна быть спроектирована как **вычислительная инфраструктура с нулевой выручкой,**[1] где операционная эффективность целиком проистекает из детерминированного исполнения программного обеспечения, а не из финансовых потоков или ручного редакторского труда. В отличие от традиционных издательских моделей, полагающихся на человекоемкие процессы и механизмы монетизации

[1] Diamond Open Access (UNESCO Open Science Framework) - https://www.unesco.org/en/diamond-open-access - Дата обращения: 9 апреля 2026 г.

(подписки или APC - сборы за публикацию), эта система функционирует как **вычислительный конвейер с замкнутым циклом.**

В своей основе платформа должна моделироваться как ориентированный ациклический граф (DAG) состояний трансформации, где каждая рукопись проходит через дискретные, четко определенные этапы обработки. Эти этапы должны быть строго идемпотентными: повторное выполнение любого этапа должно давать идентичный результат, не создавая побочных эффектов или несоответствий.

Для поддержания целостности системы должны соблюдаться следующие ограничения:

- **Детерминированная модель исполнения:** все преобразования должны давать согласованные результаты при идентичных входных данных, исключая недетерминированное поведение.
- **Валидация на основе схем:** каждый объект данных, особенно метаданные, должен соответствовать предопределенным схемам (например, XML-схемам, JSON-схемам).
- **Событийно-ориентированная оркестрация:** переходы в рабочем процессе должны инициироваться системными событиями, а не ручным вмешательством.

Это превращает издательский процесс в **контролируемую среду исполнения,** где рукописи ведут себя как вычислительные объекты, а не как продукты редакторского творчества. Результатом является система, которая поддается воспроизведению, аудиту и масштабированию без пропорционального роста операционных издержек.

Архитектура инфраструктуры и топология развертывания

Инфраструктура должна обеспечивать масштабируемость и отказоустойчивость операций при четком разделении уровней приложения и хранения данных. Она должна поддерживать автоматизированное развертывание и надежную доступность системы.

Создание облачной среды выполнения с контейнерной оркестрацией

Инфраструктура должна быть реализована как облачная распределенная система (*cloud-native*), использующая контейнеризацию для обеспечения изоляции, переносимости и масштабируемости. Архитектура должна обеспечивать строгое разделение между вычислительными уровнями без сохранения состояния (*stateless compute*) и уровнями персистентного хранения данных (*stateful persistence*), что позволяет независимо масштабировать их и изолировать сбои.

Система состоит из следующих основных уровней:

- **Вычислительный уровень (Compute Layer):** Сервисы без сохранения состояния, развернутые в виде контейнеров и оркеструемые с помощью Kubernetes.[2] Эти сервисы отвечают за прием, трансформацию, валидацию и рендеринг данных.
- **Уровень постоянного хранения (Persistence Layer):** Реляционная СУБД (например, PostgreSQL), хранящая структурированные

[2] Kubernetes Documentation for Container Orchestration - https://kubernetes.io/docs/home/ - Дата обращения: 9 апреля 2026 г.

метаданные, состояния рабочих процессов и журналы аудита. Должны быть включены механизмы репликации и аварийного переключения.

- **Уровень объектного хранилища (Object Storage Layer):** S3-совместимое хранилище для неизменяемых артефактов, таких как рукописи, XML-файлы и результаты рендеринга.
- **Уровень доступа (Ingress Layer):** Системы обратного прокси (например, NGINX или Envoy), управляющие терминацией TLS, маршрутизацией и фильтрацией запросов.

Предоставление инфраструктуры должно следовать декларативной парадигме, реализуемой с помощью таких инструментов, как Terraform. Это гарантирует, что конфигурация инфраструктуры версионируется, воспроизводима и не зависит от конкретной среды.

Операционная устойчивость достигается за счет:

- **Проб состояния (Health probes):** Проверки работоспособности и готовности для мониторинга сервисов.
- **Автоматических механизмов отказоустойчивости:** Для систем баз данных и хранилищ.
- **Стратегий резервного копирования на основе снимков:** С периодической валидацией данных.

Вопросы безопасности включают:

- Сквозное шифрование (TLS).
- Аутентификацию на основе токенов (JWT или OAuth2).
- Управление доступом на основе ролей (RBAC).

Такая архитектура гарантирует, что система способна выдерживать **частичные сбои,** сохраняя доступность и согласованность данных без каскадных нарушений в работе.

Движок редакционных процессов и обработка на основе состояний

Редакционная система должна управлять жизненным циклом рукописей через структурированные **процессы на базе состояний**, что гарантирует целостность данных и полную прозрачность истории изменений. Она должна поддерживать контролируемые переходы, действия на основе ролей и автоматизированную обработку событий на всех этапах жизненного цикла публикации.

Моделирование жизненного цикла рукописи как конечного автомата (FSM)

Редакционный рабочий процесс должен быть формализован как **конечный автомат (FSM)**, где каждая рукопись рассматривается как объект с состоянием,[3] переходящий через предопределенный жизненный цикл. Такой подход устраняет двусмысленность и обеспечивает строгий контроль над сменой состояний.

Типичные состояния включают:

- Подача (Submission)
- Первичная валидация (Initial validation)
- Рецензирование (Peer review)

[3] Workflow Management System (WfMS): Concepts and Architecture - https://en.wikipedia.org/wiki/Workflow_management_system - Дата обращения: 9 апреля 2026 г.

- Доработка (Revision)
- Принятие (Acceptance)
- Производство (Production)
- Публикация (Publication)

Каждый переход между состояниями регулируется:

- **Правилами контроля доступа:** гарантируют, что только авторизованные роли могут инициировать переходы.
- **Ограничениями валидации:** проверяют полноту метаданных и целостность файлов.
- **Триггерами событий:** генерируются действиями пользователя или автоматизированными процессами.

Ключевые архитектурные особенности включают:

- **Неизменяемое версионирование (Immutable versioning):** каждая ревизия сохраняется как новый объект, что позволяет сохранять историю состояний.
- **Событийно-ориентированное выполнение:** очереди сообщений (например, Kafka, RabbitMQ) запускают асинхронные процессы.
- **Всеобъемлющее ведение журналов аудита (Audit logging):** каждый переход фиксируется с указанием метки времени и личности исполнителя.

Данные структурированы в двух основных системах хранения:

1. **Реляционные базы данных** - для состояний рабочих процессов и метаданных.
2. **Объектные хранилища** - для файлов рукописей и их версий.

Автоматизация дополнительно усиливается за счет:

- **Асинхронного выполнения задач** для последующей обработки (downstream processing).
- **Запланированных заданий** (*cron jobs*) для напоминаний, контроля дедлайнов и обслуживания системы.

Это превращает редакционный процесс в **программируемый механизм исполнения,** обеспечивающий прозрачность, воспроизводимость и масштабируемость.

XML-ориентированное издательство и каноническое моделирование данных

Система должна использовать структурированные машиночитаемые данные для обеспечения согласованности и интероперабельности всех выходных форматов и интеграций.

Использование JATS XML в качестве канонического слоя данных

Система должна придерживаться парадигмы «XML-first», где стандарт JATS XML служит каноническим представлением всего научного контента. Это устраняет избыточность и гарантирует структурную целостность всех производных форматов.[4]

Конвейер обработки выполняет структурированные преобразования:

1. Парсинг: Входные документы (DOCX, LaTeX) анализируются системой.

[4] JATS: Journal Article Tag Suite (NISO Standard Documentation) - https://jats.nlm.nih.gov/publishing/ - Дата обращения: 9 апреля 2026 г.

2. Нормализация: Контент очищается от лишнего форматирования и приводится к единому стандарту.
3. Конвертация: Преобразование в JATS XML с использованием таких инструментов, как Pandoc.
4. Валидация: Проверка XML-файла на соответствие определениям схем.

Схема XML кодирует:

- Структуру документа: разделы, рисунки, таблицы.
- Метаданные участников: авторы, аффилиации, идентификаторы ORCID.
- Библиографические ссылки: с использованием постоянных идентификаторов (DOI и др.).

XML-документ хранится как **единый источник истины**, в то время как все производные форматы генерируются динамически через конвейеры трансформации.

Уровни рендеринга включают:

- **XSLT-преобразования** для генерации HTML-версий.
- **Движки верстки** (на базе LaTeX или CSS) для вывода в формате PDF.

Данная модель обеспечивает нормализацию данных, интероперабельность и долгосрочную сохранность контента, так как XML является машиночитаемым и платформо независимым форматом.

Инфраструктура постоянных идентификаторов и API-интеграция

Система должна интегрировать постоянные идентификаторы через автоматизированные воркфлоу API для обеспечения надежного связывания и интероперабельности между научными системами.[5] Этот уровень должен поддерживать согласованный обмен метаданными и их валидацию внешними сервисами.

Внедрение DOI и систем идентификации как распределенных сервисов

Постоянные идентификаторы критически важны для интероперабельности и должны быть реализованы как **распределенные сервисы, управляемые через API.** Каждой статье присваивается DOI, который регистрируется через внешние сервисы, такие как Crossref.

Рабочий процесс регистрации DOI включает:

- Детерминированную генерацию идентификаторов.
- Упаковку метаданных в структурированные форматы (XML/JSON).
- Отправку через API и валидацию ответов.

Для обеспечения надежности система должна реализовывать:

- **Механизмы повторных попыток (retry mechanisms)** для неудачных вызовов API.

[5] Crossref REST API and Metadata Deposit Guide - https://www.crossref.org/documentation/retrieve-metadata/rest-api/ - Дата обращения: 9 апреля 2026 г.

- **Всеобъемлющее логирование** циклов «запрос-ответ».
- **Мониторинг конечных точек разрешения (resolution endpoints)**.

Дополнительные интеграции включают:

- ORCID для однозначной идентификации авторов.
- Институциональные идентификаторы (например, ROR) для отслеживания аффилиаций.

Эти идентификаторы формируют **распределенный граф метаданных,** обеспечивая:

- Межплатформенное связывание.
- Отслеживание цитирований.
- Семантическое обогащение.

Уровень интеграции должен быть устойчивым, **поддерживать асинхронность** и корректно обрабатывать сетевые сбои

Инженерное проектирование протоколов раскрытия метаданных и обеспечения обнаруживаемости

Система должна предоставлять структурированные и соответствующие стандартам метаданные для обеспечения бесшовной интеграции с сервисами индексирования и поиска. Необходимо гарантировать единообразие форматов, эффективность извлечения данных и совместимость с протоколами сбора.

Обеспечение машинного сбора данных и индексирования

Обнаруживаемость (*discoverability*) достигается через стандартизированные протоколы раскрытия метаданных, что позволяет внешним системам эффективно собирать и индексировать контент.

Система должна реализовывать **эндпоинты OAI-PMH**, поддерживающие:

- Извлечение XML-записей с уникальными идентификаторами.
- Инкрементальный сбор данных на основе временных меток.
- Строгую валидацию схем.

На уровне представления метаданные должны быть внедрены с использованием:

- **Разметки Schema.org** для оптимизации в поисковых системах (SEO).
- Динамически генерируемых карт сайта (*sitemaps*).
- Канонических URL для однозначной идентификации ресурсов.

Стратегии оптимизации производительности включают:

- API-эндпоинты с низкой задержкой.
- Механизмы кэширования для снижения нагрузки на базу данных.
- Эффективное выполнение запросов для работы с большими наборами данных.

Это гарантирует масштабируемую обнаруживаемость публикаций на машинном уровне, исключая необходимость ручного индексирования или иного вмешательства.[6]

Производственный конвейер и детерминированная система рендеринга

Система должна реализовывать автоматизированный и воспроизводимый воркфлоу, который преобразует структурированный контент в стандартизированные выходные форматы. Это должно обеспечивать согласованность, точность и эффективность на всех этапах рендеринга.

Автоматизация многоформатной трансформации из канонического XML

Производственный конвейер функционирует как детерминированная система рендеринга, в которой валидированный XML автоматически преобразуется в готовые к публикации форматы сразу после принятия рукописи.

Последовательность трансформации включает:

- Нормализацию контента и техническое редактирование.
- Валидацию и обогащение XML.
- Рендеринг в форматы HTML и PDF.

Используемые технологии:

- Pandoc для конвертации форматов.

[6] Open Archives Initiative Protocol for Metadata Harvesting (OAI-PMH) Specification - https://www.openarchives.org/OAI/openarchivesprotocol.html - Дата обращения: 9 апреля 2026 г.

- LaTeX для высококачественной верстки.
- XSLT для структурированных преобразований.

Автоматизация реализуется через конвейеры **CI/CD**, что гарантирует:

- Автоматический запуск задач по трансформации.
- Контроль версий выходных артефактов.
- Обновление развертывания без простоев (*zero-downtime deployment*).

Вопросы производительности включают:

- Эффективную загрузку ресурсов.
- Стратегии кэширования выходных данных.
- Сжатие файлов для дистрибуции.

Система гарантирует детерминированный рендеринг: идентичные входные XML-данные всегда производят идентичные выходные файлы,[7] что подтверждает принцип воспроизводимости.

Долгосрочное хранение, наблюдаемость и проектирование жизненного цикла

Долгосрочная устойчивость требует наличия надежных механизмов обеспечения сохранности и инфраструктуры непрерывного мониторинга (наблюдаемости).

Стратегии обеспечения сохранности включают:

- Распределенную репликацию между узлами хранения.

[7] Pandoc User Guide: Universal Document Conversion and Workflow Automation - https://pandoc.org/MANUAL.html - Дата обращения: 9 апреля 2026 г.

- Использование долговечных форматов (XML, PDF/A).
- Валидацию контрольных сумм для верификации целостности данных.

Стек инструментов наблюдаемости должен включать:

- Сбор метрик (например, Prometheus).
- Централизованное логирование (например, стек ELK).
- Системы оповещения для обнаружения аномалий.

Инструменты аналитики, соответствующие требованиям конфиденциальности (например, Matomo), обеспечивают понимание моделей использования без компрометации пользовательских данных.[8]

Операции по техническому обслуживанию должны быть полностью автоматизированы:

- Регулярное резервное копирование и тестирование восстановления.
- Установка патчей безопасности и обновление зависимостей.
- Индексация и оптимизация базы данных.

Воспроизводимость инфраструктуры имеет критическое значение. Вся система должна быть реконструируемой на основе **декларативных конфигураций,** что позволяет быстро восстанавливаться после катастрофических сбоев.

[8] Providing Observability and Transparency in Distributed Digital Preservation Systems - https://www.digipres.org/publications/ipres/ipres-2024/papers/providing-observability-and-transparency-in-a-distributed-digita - Дата обращения: 9 апреля 2026 г.

Заключительная техническая перспектива

Система публикаций Diamond Open Access - это, прежде всего, вызов в области проектирования распределенных систем. Она требует тесной интеграции между оркестрацией рабочих процессов, моделированием метаданных и автоматизацией инфраструктуры.

Благодаря внедрению:

- XML-ориентированных канонических моделей данных,
- Контейнеры Зированной облачной инфраструктуры,
- API-управляемой интеграции идентификаторов,
- Детерминированных производственных конвейеров,

система эволюционирует в самодостаточную автономную издательскую платформу.

Такая архитектура устраняет зависимость от ручных процессов, обеспечивает глобальную интероперабельность и соответствует современным стандартам научной коммуникации. В конечном счете, это представляет собой переход от редакционных рабочих процессов к вычислительным конвейерам, что делает академические публикации масштабируемыми, воспроизводимыми и отказоустойчивыми.

Zeba Academy - это специализированная инициатива в области технических исследований и обучения, основанная на принципах суверенной системной инженерии. Проект, созданный Суфяном бин Узайром - автором, университетским преподавателем и сертифицированным Google Cloud DevOps-инженером, служит мостом между академической теорией и реализацией критически важных систем.

Мы отвергаем «энштификацию» современного ПО. Наша основная миссия - продвижение **архитектуры без балласта (Anti-Bloat Architecture)** через освоение следующих направлений:

- **Системные языки.** Разработка на Rust, Zig и C++ высокопроизводительных фундаментов с упором на безопасность памяти и детерминизм исполнения.
- **SRE и DevOps.** Автоматизация профессионального уровня на базе Google Cloud, Terraform и неизменяемой инфраструктуры (Immutable Infrastructure). Мы ликвидируем операционную хрупкость и ручной труд (toil).
- **Высокопроизводительные интерфейсы**. Проектирование на Flutter кроссплатформенных систем с нативным откликом. Без компромиссов и задержек, присущих стандартным веб-оболочкам.
- **Регенерация веб-издательства.** Возврат WordPress и PHP в строй через радикальную очистку от «шлака». Объектное кэширование в Redis и Unix-сокеты превращают стандартные платформы в скоростные геостабильные движки.
- **Модернизация Legacy-систем**. Перенос классических вычислительных задач и старых кодовых баз на C в современные парадигмы безопасной работы с памятью и актуальные системы сборки.

Zeba Academy не просто учит писать код - мы проектируем надежность. Объединяя аналитическую строгость исторических исследований с точностью сертифицированной облачной инженерии Google, мы вооружаем наших «оперативников» директивами, необходимыми для создания систем, которые будут безопасными, быстрыми и долговечными.

Вебсайт: - https://zeba.academy

زیبا

Zeba Academy

zeba.academy