Т. 12. №4 2026
https://doi.org/10.33619/2414-2948/125

УДК 004.42 https://doi.org/10.33619/2414-2948/125/14

АРХИТЕКТУРНЫЙ ПОДХОД К ПРОЕКТИРОВАНИЮ РАСПРЕДЕЛЁННОЙ ВЕБ-СИСТЕМЫ УПРАВЛЕНИЯ ТОВАРНЫМИ ПОТОКАМИ НА ОСНОВЕ PHP И MYSQL

©***Аркабаев Н. К.,*** *ORCID: 0009-0000-1912-2225, SPIN-код: 9304-5193, канд. физ.-мат. наук, Ошский государственный университет, г. Ош, Кыргызстан, narkabaev@oshsu.kg*
©***Азизбек кызы М.,*** *ORCID: 0009-0007-2178-8262, Ошский государственный университет, г. Ош, Кыргызстан, azizbekkyzyminura649@gmail.com*
©***Минзамидинова М. К.,*** *ORCID: 0009-0002-3592-7407, Ошский государственный университет, г. Ош, Кыргызстан, meerim.minzamidinova@icloud.com*

AN ARCHITECTURAL APPROACH TO DESIGNING A DISTRIBUTED WEB-BASED SYSTEM FOR MANAGING PRODUCT FLOWS BASED ON PHP AND MYSQL

©***Arkabaev N.,*** *ORCID: 0009-0000-1912-2225, SPIN-code: 9304-5193, Ph.D., Osh State University, Osh, Kyrgyzstan, narkabaev@oshsu.kg*
© ***Azizbek kyzy M.,*** *ORCID: 0009-0007-2178-8262, Osh State University, Osh, Kyrgyzstan, azizbekkyzyminura649@gmail.com*
©***Minzamidinova M.,*** *ORCID: 0009-0002-3592-7407, Osh State University, Osh, Kyrgyzstan, meerim.minzamidinova@icloud.com*

Аннотация. Рассматривается задача архитектурного проектирования распределённой веб-системы, предназначенной для управления товарными потоками предприятия. На основе анализа предметной области выделены функциональные и нефункциональные требования, определяющие выбор архитектурного решения. Представлена трёхуровневая архитектурная модель, включающая уровень клиентского представления, уровень серверной бизнес-логики на базе PHP и уровень хранения данных средствами MySQL. Для обеспечения территориально распределённого функционирования предложен механизм синхронизации данных между центральным узлом и удалёнными филиалами с использованием встроенной репликации MySQL и RESTful-интерфейсов программного взаимодействия. Проведён сравнительный анализ монолитной, трёхуровневой и микросервисной архитектур по критериям масштабируемости, стоимости внедрения, сложности сопровождения и отказоустойчивости. Результаты анализа свидетельствуют о том, что для предприятий среднего масштаба с ограниченным бюджетом информатизации трёхуровневая архитектура с элементами распределённости обеспечивает приемлемый баланс между функциональными возможностями и ресурсоёмкостью разработки.

Abstract. This paper addresses the architectural design of a distributed web-based system for enterprise commodity flow management. Based on domain analysis, functional and non-functional requirements influencing the architectural decision are identified. A three-tier architectural model is proposed, consisting of a client presentation layer, a server-side business logic layer built on PHP, and a data storage layer powered by MySQL. To support geographically distributed operations, a data synchronization mechanism between the central node and remote branches is introduced, utilizing MySQL native replication and RESTful application programming interfaces. A comparative analysis of monolithic, three-tier, and microservice architectures is conducted using criteria of scalability, deployment cost, maintenance complexity, and fault tolerance. The findings indicate that for medium-sized enterprises with limited IT budgets, a three-tier architecture with distributed features achieves an acceptable trade-off between functionality and development effort.

Т. 12. №4 2026
https://doi.org/10.33619/2414-2948/125

Ключевые слова: распределённая система, управление товарными потоками, трёхуровневая архитектура, PHP, MySQL, репликация, REST API, веб-приложение.

Keywords: distributed system, commodity flow management, three-tier architecture, PHP, MySQL, replication, REST API, web application.

Современные предприятия, осуществляющие торгово-закупочную деятельность, сталкиваются с необходимостью оперативного учёта и контроля перемещения товаров между территориально удалёнными подразделениями – складами, торговыми точками, распределительными центрами. Ручной или полуавтоматизированный учёт порождает задержки в обработке данных, несогласованность остатков и, как следствие, финансовые потери. Потребность в надёжных информационных системах, способных функционировать в условиях распределённой инфраструктуры, остаётся одной из ключевых задач прикладной информатики. Выбор архитектурного решения для подобной системы определяет не только её технические характеристики – производительность, масштабируемость, устойчивость к отказам, – но и экономическую целесообразность внедрения. Крупные корпоративные платформы класса ERP предлагают комплексную функциональность, однако их стоимость и сложность адаптации делают такие решения малодоступными для предприятий среднего и малого бизнеса, особенно в условиях развивающихся экономик Центральной Азии.

Веб-технологии открытого доступа, в частности серверный язык PHP и реляционная СУБД MySQL, остаются востребованным инструментарием для построения корпоративных информационных систем благодаря низкому порогу входа, широкой экосистеме библиотек и поддержке сообщества разработчиков. По данным W3Techs, на долю PHP приходится более 75 % серверных веб-приложений, а MySQL входит в тройку наиболее распространённых СУБД в мире. Вместе с тем вопрос архитектурной организации распределённой системы на базе указанных технологий исследован недостаточно: большинство работ ограничиваются описанием локальных приложений без учёта территориальной распределённости.

Целью настоящей статьи является обоснование архитектурного подхода к проектированию распределённой веб-системы управления товарными потоками на основе PHP и MySQL. Для достижения данной цели решаются следующие задачи: анализ предметной области и формирование системных требований; разработка трёхуровневой архитектурной модели с механизмом распределённости; проектирование схемы базы данных и интеграционных интерфейсов; сравнительная оценка альтернативных архитектурных подходов.

Архитектурное проектирование информационных систем управления цепочками поставок и товарными потоками является предметом многочисленных исследований. В исследовании Чен и др. провели масштабный обзор архитектур интеграции предприятий, отметив тенденцию перехода от централизованных решений к распределённым, сервис-ориентированным моделям. Авторы подчеркнули, что выбор архитектурного стиля должен определяться не только техническими, но и организационными факторами – размером предприятия, степенью географической рассредоточенности и уровнем ИТ-компетенций персонала [1].

В контексте сравнения архитектурных подходов значительный интерес представляет работа, в которой проведена количественная оценка производительности микросервисной архитектур на эталонном веб-приложении. Результаты показали, что на одном сервере монолитное приложение демонстрирует более высокую пропускную способность, тогда как микросервисная архитектура выигрывает при горизонтальном масштабировании. Данный

Т. 12. №4 2026
https://doi.org/10.33619/2414-2948/125

вывод подтверждает, что для предприятий с умеренной нагрузкой переход к микросервисам не всегда оправдан экономически [2].

Проблематика распределённых баз данных исследуется в работах, посвящённых стратегиям репликации и фрагментации. В работе Heryanto и Albert предложили классификационную модель для сопоставления методов репликации, применяемых в распределённых системах и базах данных. Авторы выделили ключевые компромиссы между согласованностью данных и доступностью, получившие впоследствии формальное выражение в теореме CAP [3].

Применение REST-интерфейсов для интеграции компонентов распределённых систем рассматривается в ряде работ. Исследователь Калифорнийского университета Thomas Fielding своей диссертации сформулировал принципы архитектурного стиля REST, ставшего де-факто стандартом взаимодействия веб-сервисов [4].

Исследователи университета Коимбры провели анализ публичных REST API и выявили характерные паттерны проектирования, обеспечивающие совместимость и расширяемость интерфейсов [5].

В области управления цепочками поставок Джордж Кусиурис и др. предложили микросервисный фреймворк, интегрирующий IoT-платформы и семантические сервисы для мониторинга товарных потоков [6].

Несмотря на перспективность подобного подхода, его реализация требует значительных инвестиций в инфраструктуру, что ограничивает применимость для предприятий с ограниченными ресурсами.

Практические аспекты применения PHP и MySQL для построения веб-сервисов в распределённых информационных системах описаны в работе Воронцова и Козинца [7]. Авторы продемонстрировали реализацию трёхкомпонентной архитектуры, веб-сервер Apache с PHP-интерпретатором и СУБД MySQL. На примере веб-сервиса с клиентской и административной частями, функционирующего в виртуализированной среде. А в исследовании Аркабаева и Алымовой [8] рассматриваются практические аспекты разработки высоконагруженных веб-приложений на платформе ASP.NET Core на примере интернет-магазина. Магистрант Московского финансово-промышленного университета «Синергия» Алексеев исследовал применение API и микросервисов на PHP в контексте учёта товарных запасов [10]. В работе проанализированы децентрализованные системы управления складом, включая решения на базе беспроводных сенсорных сетей и RFID-технологий, а также рассмотрена роль открытых PHP-библиотек в построении гибких интеграционных решений.

Таким образом, анализ литературы показывает, что, с одной стороны, существуют развитые архитектурные концепции для построения крупномасштабных распределённых систем, а с другой – ощущается дефицит прикладных исследований, адаптирующих эти концепции к условиям малого и среднего бизнеса с использованием доступного технологического стека.

Управление товарными потоками предприятия охватывает совокупность операций, связанных с планированием, учётом и контролем перемещения материальных ценностей на всех этапах – от закупки у поставщиков до реализации конечному потребителю. Типовой цикл включает приёмку товара на склад, внутреннее перемещение между подразделениями, отгрузку контрагентам и списание по различным основаниям. В условиях территориально распределённого предприятия, имеющего несколько складских и торговых точек, возникают дополнительные требования к информационной системе. ИС необходимо обеспечить единое информационное пространство: сотрудники каждого подразделения должны иметь доступ к актуальным данным об остатках, ценах и статусах заказов. А также, система должна корректно

Т. 12. №4 2026
https://doi.org/10.33619/2414-2948/125

функционировать при временной недоступности каналов связи между узлами – ситуации, типичной для ряда регионов Центральной Азии. Различные категории пользователей (администраторы, складские работники, менеджеры по продажам) предполагают ролевую модель доступа с дифференцированными полномочиями. На основании изложенного сформулированы следующие требования к проектируемой системе. К функциональным требованиям отнесены: учёт приходных и расходных операций, формирование отчётности по остаткам и движению товаров, управление справочниками номенклатуры и контрагентов, поддержка многоскладового учёта и операций межскладского перемещения. Нефункциональные требования включают: доступность системы через веб-интерфейс без установки клиентского программного обеспечения, масштабируемость при подключении новых узлов, согласованность данных между центральным и удалёнными хранилищами, а также разграничение доступа на основе ролей.

Предлагаемый архитектурный подход

Общая трёхуровневая модель. В качестве базовой архитектурной модели выбрана трёхуровневая (three-tier) архитектура, разделяющая систему на уровень представления, уровень бизнес-логики и уровень данных. Подобная декомпозиция обеспечивает независимость компонентов, что упрощает сопровождение и допускает замену отдельных модулей без перестройки всей системы.

Уровень представления реализуется в виде веб-интерфейса, доступного через стандартный браузер. Клиентская часть формируется с использованием HTML, CSS и JavaScript; при этом серверная генерация страниц через PHP-шаблонизаторы снижает зависимость от клиентских вычислительных ресурсов. Уровень бизнес-логики размещён на PHP-сервере, обрабатывающем HTTP-запросы, выполняющем валидацию данных, реализующем алгоритмы расчёта остатков и формирования документов. Уровень данных представлен сервером MySQL, хранящим нормализованные таблицы номенклатуры, складских операций, контрагентов и пользователей.

Схема распределённости. Территориальная распределённость предприятия требует расширения классической трёхуровневой модели механизмами межузлового взаимодействия. Предлагаемая схема предусматривает выделение центрального узла, на котором размещено основное хранилище данных, и периферийных узлов, соответствующих отдельным филиалам или складам.

Каждый периферийный узел функционирует автономно – располагает локальным экземпляром MySQL и PHP-сервером, обеспечивающим полноценную работу сотрудников при отсутствии связи с центром. Синхронизация данных между узлами реализуется по двухканальной схеме. Первый канал – встроенная репликация MySQL, сконфигурированная по модели «ведущий – ведомый» (master–slave): центральный узел выступает в роли источника, а периферийные узлы получают обновления посредством асинхронного журнала двоичных событий (binary log). Данная схема обеспечивает консистентное чтение на удалённых узлах при минимальных задержках.

Второй канал – RESTful API, реализованный на стороне центрального сервера средствами PHP. Периферийные узлы используют этот интерфейс для передачи локально зарегистрированных операций (приходных накладных, отгрузок, инвентаризационных актов) в центральное хранилище. Подобный подход позволяет накапливать транзакции на локальном узле в период недоступности центра и отправлять их пакетом при восстановлении связи.

Проектирование базы данных. Схема базы данных разработана в соответствии с третьей нормальной формой реляционной модели. Основные сущности включают: «Номенклатура» (products) – справочник товарных позиций с атрибутами наименования, единицы измерения,

Т. 12. №4 2026
https://doi.org/10.33619/2414-2948/125

категории и штрихкода; «Склад» (warehouses) – перечень территориальных подразделений с привязкой к узлу системы; «Операция» (transactions) – журнал движения товаров с указанием типа операции, количества, даты и ответственного лица; «Контрагент» (counterparties) – справочник поставщиков и покупателей; «Пользователь» (users) – учётные записи с ролевой моделью доступа.

Для обеспечения корректной работы репликации каждая запись снабжается глобальным уникальным идентификатором (UUID), что исключает конфликты первичных ключей при слиянии данных с нескольких узлов. Дополнительно введён атрибут sync_status, принимающий значения «синхронизировано», «ожидает синхронизации» и «конфликт», что позволяет контролировать состояние каждой записи в распределённой среде.

REST API как механизм интеграции. Интеграционный слой реализован в виде набора RESTful-эндпоинтов, размещённых на центральном сервере. Взаимодействие осуществляется по протоколу HTTPS с использованием формата передачи данных JSON. Аутентификация узлов производится посредством токенов, генерируемых по алгоритму HMAC-SHA256 с ограниченным сроком действия.

Основные группы эндпоинтов включают: /api/sync/push – приём пакетов транзакций от периферийных узлов; /api/sync/pull – выдача обновлений, зарегистрированных на центральном сервере; /api/references – синхронизация справочных данных (номенклатура, контрагенты); /api/reports – формирование консолидированной отчётности. Каждый запрос сопровождается метаданными узла-отправителя и временной меткой, что обеспечивает идемпотентность операций и возможность разрешения конфликтов на стороне сервера.

Сравнительный анализ архитектурных решений

Для обоснования выбора архитектурного подхода проведено сопоставление трёх альтернатив: монолитной, трёхуровневой с распределёнными элементами и микросервисной архитектур. Критериями сравнения выступили: масштабируемость, стоимость разработки и внедрения, сложность сопровождения, отказоустойчивость и требования к квалификации разработчиков.

Таблица

СРАВНИТЕЛЬНАЯ ХАРАКТЕРИСТИКА АРХИТЕКТУРНЫХ ПОДХОДОВ

Критерий	*Монолитная*	*Трёхуровневая (предлагаемая)*	*Микросервисная*
Масштабируемость	Низкая (вертикальная)	Средняя (вертикальная + частичная горизонтальная)	Высокая (горизонтальная)
Стоимость внедрения	Низкая	Умеренная	Высокая
Сложность сопровождения	Возрастает с ростом кодовой базы	Умеренная	Высокая (оркестрация, мониторинг)
Отказоустойчивость	Единая точка отказа	Локальная автономность узлов	Изолированные сбои
Требования к команде	Базовые	Средние	Высокие (DevOps, контейнеризация)
Технологический стек	Единый	PHP + MySQL + REST API	Полиглотный

Как следует из Таблицы, микросервисная архитектура превосходит альтернативы по масштабируемости и отказоустойчивости, однако сопряжена со значительно более высокими затратами на разработку, развёртывание и сопровождение. Экспериментально продемонстрировали, что на единичном сервере монолитное приложение обрабатывает

Т. 12. №4 2026
https://doi.org/10.33619/2414-2948/125

запросы быстрее микросервисного аналога, и преимущество последнего проявляется лишь при горизонтальном масштабировании под нагрузкой, превышающей тысячи одновременных подключений. Для предприятий среднего масштаба, характеризующихся десятками, а не тысячами одновременных пользователей, подобный уровень нагрузки не типичен.

Монолитная архитектура, при всей своей простоте, создаёт риски при расширении функциональности: рост кодовой базы ведёт к усложнению тестирования и развёртывания. Кроме того, отсутствие встроенных механизмов распределённости делает монолит непригодным для предприятий с удалёнными подразделениями.

Предлагаемая трёхуровневая архитектура с элементами распределённости занимает промежуточную позицию. Она сохраняет простоту разработки и развёртывания, характерную для классических веб-приложений, дополняя её механизмами репликации и API-интеграции для поддержки многоузловой топологии. Подобный подход представляется оптимальным для целевого сегмента – предприятий с 2-10 территориальными подразделениями и командой разработки, владеющей стандартным стеком PHP/MySQL.

Результаты и обсуждение

На основе предложенной архитектуры разработан прототип распределённой системы управления товарными потоками. Серверная часть реализована на PHP 8.2 с применением паттерна MVC и фреймворка маршрутизации. В качестве СУБД использован MySQL 8.0 с настроенной асинхронной репликацией. Клиентский интерфейс построен на HTML5, CSS3 и библиотеке jQuery для обеспечения асинхронного обмена данными.

Тестирование проведено в условиях, моделирующих работу предприятия с тремя территориальными подразделениями. Центральный узел развёрнут на виртуальном сервере (2 ядра CPU, 4 ГБ ОЗУ), периферийные узлы – на аналогичных конфигурациях. Канал связи между узлами эмулировался с различными параметрами задержки и пропускной способности.

Среднее время синхронизации пакета из 100 транзакций между периферийным и центральным узлами составило 1,2 секунды при стабильном соединении и 8,7 секунды при канале с задержкой 200 мс и потерями 5 % пакетов. При полном разрыве связи периферийный узел продолжал функционировать автономно; после восстановления соединения накопленные транзакции были переданы и корректно интегрированы в центральное хранилище без потери данных.

Нагрузочное тестирование с помощью утилиты Apache JMeter выявило, что система устойчиво обрабатывает до 50 одновременных запросов на каждом узле при среднем времени отклика 320 мс. Данный показатель соответствует типичному профилю нагрузки целевого сегмента предприятий.

Вместе с тем необходимо отметить ограничения предложенного решения. Асинхронная репликация допускает кратковременную рассогласованность данных между узлами, что может приводить к конфликтам при одновременном редактировании одних и тех же записей на разных узлах. В текущей реализации разрешение конфликтов осуществляется по принципу приоритета временной метки (last-write-wins), что не во всех сценариях является оптимальной стратегией. Разработка более совершенного механизма разрешения конфликтов с учётом семантики операций представляет собой перспективное направление дальнейших исследований.

Заключение

Итак, представлен архитектурный подход к проектированию распределённой веб-системы управления товарными потоками предприятия на основе технологий PHP и MySQL.

Предложена трёхуровневая модель, расширенная механизмами межузловой синхронизации – репликацией MySQL и RESTful-интерфейсами взаимодействия. Данная модель обеспечивает автономное функционирование территориально удалённых подразделений при сохранении целостности консолидированного хранилища данных. Сравнительный анализ архитектурных альтернатив подтвердил, что для предприятий среднего масштаба с ограниченными ресурсами информатизации предложенный подход обеспечивает рациональный баланс между функциональностью, стоимостью внедрения и сложностью сопровождения. В отличие от микросервисной архитектуры, требующей развитой DevOps-инфраструктуры, рассматриваемое решение может быть развёрнуто и поддержано командой из 2–3 разработчиков, владеющих стандартным стеком веб-технологий. К перспективным направлениям развития следует отнести: внедрение механизма семантического разрешения конфликтов синхронизации; интеграцию модуля прогнозирования спроса на основе методов машинного обучения; адаптацию системы для работы в контейнерной среде, что упростит развёртывание и масштабирование в облачных инфраструктурах.

Список литературы:

1. Chen D., Doumeingts G., Vernadat F. Architectures for enterprise integration and interoperability: Past, present and future // Computers in industry. 2008. V. 59. №7. P. 647-659. https://doi.org/10.1016/j.compind.2007.12.016

2. Маличенко С. В. Проблемы перехода от монолитной к микросервисной архитектуре // Евразийский научный журнал. 2022. №5. С. 8-19.

3. Heryanto A., Albert A. Implementasi sistem database Terdistribusi Dengan metode multi-master database replication //Jurnal Media Informatika Budidarma. 2019. V. 3. №1. P. 30-36. https://doi.org/10.30865/mib.v3i1.1098

4. Fielding R. T. Architectural styles and the design of network-based software architectures. – University of California, Irvine, 2000. https://dl.acm.org/doi/10.5555/932295

5. Neumann A., Laranjeiro N., Bernardino J. An analysis of public REST web service APIs // IEEE Transactions on Services Computing. 2018. V. 14. №4. P. 957-970. https://doi.org/10.1109/TSC.2018.2847344

6. Kousiouris G., Tsarsitalidis S., Psomakelis E., Koloniaris S., Bardaki C., Tserpes K., Anagnostopoulos D. A microservice-based framework for integrating IoT management platforms, semantic and AI services for supply chain management // Ict Express. 2019. V. 5. №2. P. 141-145. https://doi.org/10.1016/j.icte.2019.04.002

7. Воронцов Ю. А., Козинец А. В. Пример построения web-сервиса с использованием Apache и MySQL // Век качества. 2016. №3. С. 75-101.

8. Аркабаев Н. К., Алымова З. Ж. Разработка web серверных приложений на базе. NET Core в примере интернет-магазина // Вестник Ошского государственного университета. 2024. №1. P. 142-154. https://doi.org/10.52754/16948610_2024_1_13

9. Аркабаев Н. К., Беделова Н. С., Маткалык кызы К. Алгоритмы оптимизации финансовых решений: сравнительный анализ методов и их применение в программных продуктах // Бюллетень науки и практики. 2025. Т. 11. №11. С. 19-27. https://doi.org/10.33619/2414-2948/120/02

10. Алексеев А. А. Использование Api И Микросервисов На Php В Учете Товарных Запасов // Вестник науки. 2024. Т. 3. №1 (70). С. 632-651. https://doi.org/10.24412/2712-8849-2024-170-632-651

References:

1. Chen, D., Doumeingts, G., & Vernadat, F. (2008). Architectures for enterprise integration and interoperability: Past, present and future. *Computers in industry*, *59*(7), 647-659. https://doi.org/10.1016/j.compind.2007.12.016

2. Malichenko, S. V. (2022). Problemy perekhoda ot monolitnoi k mikroservisnoi arkhitekture. *Evraziiskii nauchnyi zhurnal,* (5), 8-19. (in Russian).

3. Heryanto, A., & Albert, A. (2019). Implementasi sistem database Terdistribusi Dengan metode multi-master database replication. *Jurnal Media Informatika Budidarma*, *3*(1), 30-36.

4. Fielding, R. T. (2000). *Architectural styles and the design of network-based software architectures*. University of California, Irvine. https://dl.acm.org/doi/10.5555/932295

5. Neumann, A., Laranjeiro, N., & Bernardino, J. (2018). An analysis of public REST web service APIs. *IEEE Transactions on Services Computing*, *14*(4), 957-970. https://doi.org/10.1109/TSC.2018.2847344

6. Kousiouris, G., Tsarsitalidis, S., Psomakelis, E., Koloniaris, S., Bardaki, C., Tserpes, K., ... & Anagnostopoulos, D. (2019). A microservice-based framework for integrating IoT management platforms, semantic and AI services for supply chain management. *Ict Express*, *5*(2), 141-145. https://doi.org/10.1016/j.icte.2019.04.002

7. Vorontsov, Yu. A., & Kozinets, A. V. (2016). Primer postroeniya web-servisa s ispol'zovaniem Apache i MySQL. *Vek kachestva, (*3), 75-101. (in Russian).

8. Arkabaev, N. K., & Alymova, Z. Zh. (2024). Razrabotka web servernykh prilozhenii na baze. NET Core v primere internet-magazina. *Vestnik Oshskogo gosudarstvennogo universiteta,* (1), 142-154. (in Russian). https://doi.org/10.52754/16948610_2024_1_13

9. Arkabaev, N., Bedelova, N., & Matkalyk kyzy, K. (2025). Financial Decision Optimization Algorithms: Comparative Analysis of Methods and their Application in Software Products. *Bulletin of Science and Practice, 11*(11), 19-27. (in Russian). https://doi.org/10.33619/2414-2948/120/02

10. Alekseev, A. A. (2024). Ispol'zovanie Api I Mikroservisov Na Php V Uchete Tovarnykh Zapasov. *Vestnik nauki, 3*(1 (70)), 632-651. (in Russian). https://doi.org/10.24412/2712-8849-2024-170-632-651

Поступила в редакцию
16.02.2026 г.

Принята к публикации
25.02.2026 г.

Ссылка для цитирования:

Аркабаев Н. К., Азизбек кызы М., Минзамидинова М. К. Архитектурный подход к проектированию распределённой веб-системы управления товарными потоками на основе PHP и MySQL // Бюллетень науки и практики. 2026. Т. 12. №4. С. 103-110. https://doi.org/10.33619/2414-2948/125/14

Cite as (APA):

Arkabaev, N., Azizbek kyzy, M., & Minzamidinova, M. (2026). An Architectural Approach to Designing a Distributed Web-Based System for Managing Product flows Based on PHP and MySQL. *Bulletin of Science and Practice, 12*(4), 103-110. (in Russian). https://doi.org/10.33619/2414-2948/125/14