Научная статья
УДК 004.738.52:004.434
https://doi.org/10.35266/1999-7604-2024-3-4

Краткое аннотирование новостей из области информационных технологий

Кирилл Владимирович Кузнецов[1], Светлана Александровна Лысенкова[2]✉
[1, 2]Сургутский государственный университет, Сургут, Россия
[1]kirill.kuznetsov.v@icloud.com, https://orcid.org/0009-0007-3401-1962
[2]lysenkova_sa@surgu.ru✉, https://orcid.org/0000-0003-1007-7610

Аннотация. В статье рассматривается концепция веб-платформы, созданной для сбора и аннотирования новостей в сфере информационных технологий. Платформа собирает данные из различных источников, предлагая пользователям удобный интерфейс для поиска и чтения новостных материалов. Одной из ключевых особенностей системы является возможность классификации статей, написание аннотаций и присвоение тегов при использовании больших языковых моделей (LLM), а также автоматического перевода статей с оригинального языка на другие.

Ключевые слова: веб-платформа, классификация информации, большие языковые модели, аннотирование новостей, сбор данных, контейнеризация

Для цитирования: Кузнецов К. В., Лысенкова С. А. Краткое аннотирование новостей из области информационных технологий // Вестник кибернетики. 2024. Т. 23, № 3. С. 31–39. https://doi.org/10.35266/1999-7604-2024-3-4.

Original article

Summary of news in information technology

Kirill V. Kuznetsov[1], Svetlana A. Lysenkova[2]✉
[1, 2]Surgut State University, Surgut, Russia
[1]kirill.kuznetsov.v@icloud.com, https://orcid.org/0009-0007-3401-1962
[2]lysenkova_sa@surgu.ru✉, https://orcid.org/0000-0003-1007-7610

Abstract. The article discusses the concept of a web platform designed for collecting and summarizing news in information technology. The platform gathers data from various sources, offering users a convenient interface for searching and reading news materials. The key features of the system are the ability to classify articles, make summaries, and assign tags using large language models (LLM), as well as the automatic translation of articles into other languages.

Keywords: web platform, information classification, large language models, news summary, data collection, OS-level virtualization

For citation: Kuznetsov K. V., Lysenkova S. A. Summary of news in information technology. *Proceedings in Cybernetics*. 2024;23(3):31–39. https://doi.org/10.35266/1999-7604-2024-3-4.

ВВЕДЕНИЕ

Ежедневно пользователи сталкиваются с тысячами новостных материалов, связанных с информационными технологиями. В таких условиях стремительного развития сложно оставаться в курсе последних новостей и тенденций.

Прогнозируется, что к 2025 г. объем данных, генерируемых, потребляемых, копируемых и хранимых, превысит 180 зеттабайт. В то же время в 2020 г. этот объем составил 64,2 зеттабайта. С 2012 по 2020 гг. процент полезной информации для анализа вырос с 22 до 37 %. Эти

данные включают информацию из различных областей: социальные сети, развлечения, мониторинг и многое другое [1–2].

Для решения проблем, связанных с увеличивающимся объемом информации, возникает необходимость в разработке платформы, способной автоматизировать процессы сбора, фильтрации и анализа данных. Она должна обеспечивать сбор новостей из различных источников, фильтрацию по тематике, написание аннотаций по каждой собранной статье, иметь возможность, основываясь на заголовке и контексте статьи, соотносить ее с одной из категорий сферы информационных технологий, определять теги, а также в случае, если язык оригинальной статьи отличается от русского языка, переводить заголовок и контекст.

МАТЕРИАЛЫ И МЕТОДЫ

Решение о выборе конкретной архитектуры для проектируемой платформы, принимаемое перед началом ее разработки, может зависеть от некоторых факторов.

В первую очередь платформа должна отвечать требованиям высокой скорости обработки большого количества источников. Это особенно важно, так как в современных условиях объемы данных растут экспоненциально [1, 2]. Не менее важным условием, которым система должна отвечать, является масштабируемость – архитектура должна позволять системе легко масштабироваться вместе с увеличением числа пользователей и количества источников новостных статей. Без надлежащей адаптивности система может быстро стать устаревшей. Надежность также играет важную роль – архитектура должна обеспечивать стабильную работу платформы в сценарии высоких нагрузок, при необходимости распределяя задачи между другими узлами, элементами системы.

Одним из основных требований для разрабатываемой платформы является возможность ее легкого развертывания и обеспечения наилучшей совместимости на различных операционных системах. В данном контексте оптимальным решением является применение технологии контейнеризации с использованием Docker и инструмента управления контейнерами Docker Compose.

Docker – это средство, позволяющее паковать всевозможные приложения прямо вместе со всеми включенными в них зависимостями в стандартизированные контейнеры. Эти самые контейнеры легко и просто могут быть перемещены между разными средами разработки и продакшна, обеспечивая консистентность и надежность выполнения приложений. Docker гарантирует изоляцию отличных процессов и ресурсов, что делает его мощным инструментом для создания, тестирования, а также развертывания всевозможных приложений в разнообразных сценариях [3].

Среди наиболее распространенных типов архитектуры программного обеспечения (ПО) выделяются следующие: монолитная, микросервисная, многоуровневая, сервис-ориентированная и клиент-серверная.

Монолитная архитектура предполагает организацию платформы в виде единого модуля, состоящего из тесно связанных между собой структурных компонентов. Этот метод является традиционным и обычно не требует значительных затрат для внедрения. Все программные компоненты монолитной системы взаимозависимы из-за использования встроенных механизмов обмена данными внутри системы. Модификация монолитной архитектуры возможна лишь частично и занимает много времени, поскольку даже небольшие изменения затрагивают большие области базы кода [4].

Идея микросервисной архитектуры основана на сервис-ориентированном подходе и предполагает создание независимых сервисов для обеспечения функциональности платформы. При таком подходе приложение разрабатывается как набор небольших сервисов; каждая служба является автономной и должна реализовывать одну бизнес-задачу в ограниченном контексте [5–8].

Многоуровневая архитектура, подобно клиент-серверной, делит платформу на уровни для управления данными и получения информации [5]. Архитектура данного типа может содержать два, три уровня или больше, в зависимости от потребностей проекта.

Сервис-ориентированная архитектура (SOA) – это метод разработки программного обеспечения, который использует программные компоненты, называемые сервисами, для создания бизнес-приложений. Каждый сервис предоставляет бизнес-возможности, сервисы также могут взаимодействовать друг с другом на разных платформах и языках. Разработчики применяют SOA для многократного использования сервисов в различных системах или объединения нескольких независимых сервисов для выполнения сложных задач [9].

Клиент-серверная архитектура ПО разделяет систему на два основных приложения, в которых клиент отправляет запросы на сервер, получая от него ответы [10]. Такой подход позволяет переложить задачу обработки данных на серверную часть.

При выборе архитектурного подхода для создания программного обеспечения важно учитывать особенности и ограничения каждого подхода. Рассмотрим, почему использование микросервисной архитектуры предпочтительно в проекте, где предусмотрено использование контейнеризации и Docker Compose, по сравнению с другими архитектурными стилями.

При рассмотрении клиент-серверной архитектуры возникают ограничения в масштабируемости и гибкости, однако, несмотря на это, она остается довольно хорошим выбором. При использовании клиент-серверного подхода проектирования ПО могут возникнуть проблемы с расширением серверной части при большом количестве запросов, что может привести к невозможности использования веб-платформы. Следует отметить, что некоторые из требований будет значительно сложнее реализовать при соблюдении данной архитектуры. При рассмотрении сценария одновременной обработки большого количества источников может возникнуть проблема с отображением аннотаций в реальном времени и в общей работе сервера.

Несмотря на свою простоту, монолитная архитектура имеет свои недостатки, которые могут затруднить ее применение в платформе. В таких системах все части приложения тесно взаимосвязаны, что усложняет их поддержку и масштабирование. Существует вероятность, что для внесения изменений придется значительно модифицировать участки кода программы. Один из весомых недостатков использования монолитной архитектуры заключается в ограниченной возможности гибкого управления каждого из компонентов; сбой в одном из компонентов повлияет на всю систему.

Многослойная структура платформы позволяет разделить ее на различные уровни для управления информацией и обработки данных, что способствует повышению гибкости и некоторой масштабируемости приложения. Однако такая структура может усложнить внесение изменений из-за своей монолитной природы. В рамках такого подхода к разработке программного обеспечения данные должны проходить через каждый слой даже в случаях, когда это не требуется.

Сервис-ориентированная архитектура (SOA) позволяет эффективно использовать сервисы и масштабировать их, однако для этого часто требуются сложные методы взаимодействия, такие как применение ESB (Enterprise Service Bus) – «сервисной шины предприятия» (связующего ПО, обеспечивающего централизованный и унифицированный событийно-ориентированный обмен сообщениями между различными информационными системами).

При выборе такого архитектурного подхода обычно придерживаются более унифицированного технологического стека для всех сервисов. Несмотря на то что такая архитектура ПО предусматривает разделение системы на отдельные сервисы, эти сервисы часто тесно связаны друг с другом.

Исследовав все вышеперечисленные подходы к созданию программного обеспечения, можно сделать вывод, что микросервисная архитектура является наиболее предпочтительной. Несмотря на некоторые недостатки (более длительное время разработки по сравнению с другими архитектурными решениями, необходимость проведения

комплексного тестирования для отладки всех компонентов), преимуществ у этого типа архитектуры больше. Микросервисная архитектура обеспечивает гибкость за счет использования небольших автономных сервисов, которые легко модифицировать или масштабировать при помощи контейнеризации. Развитие функциональности платформы через подход разделения на микросервисы позволяет создавать, внедрять и масштабировать каждый модуль индивидуально. Этот метод является оптимальным для проектов, где требуется быстрое развитие и способность оперативно реагировать на изменения рынка.

РЕЗУЛЬТАТЫ И ИХ ОБСУЖДЕНИЕ

Веб-платформа для обработки новостной информации объединяет технологии анализа текста с использованием больших языковых моделей (large language models, LLM) и информационных технологий с целью автоматизации сбора, анализа и представления актуальных новостей. Ниже представлена концептуальная модель, демонстрирующая ключевые компоненты и взаимосвязи между ними.

На диаграмме (рис. 1), представлена модель платформы. Модель включает несколько ключевых компонентов и модулей, которые взаимодействуют между собой для обеспечения работы системы.

На диаграмме представлены следующие ключевые компоненты:

1. Парсер:

Атрибуты: Ресурс

Методы: Запуск(), Собрать статьи(), Следующая страница(), Отправить на сервер()

Функции: Парсер отвечает за извлечение информации из источников. Он собирает ссылки на новостные статьи, переходя по страницам информационного ресурса, после чего отправляет их сервису. Сервис создает очередь ссылок на статьи для дальнейшего получения их контекста и отправки на сервер базы данных.

2. Источник:

Атрибуты: Название, Ссылка, Язык

Функции: Источник предоставляет информацию, из которой парсер будет извлекать статьи.

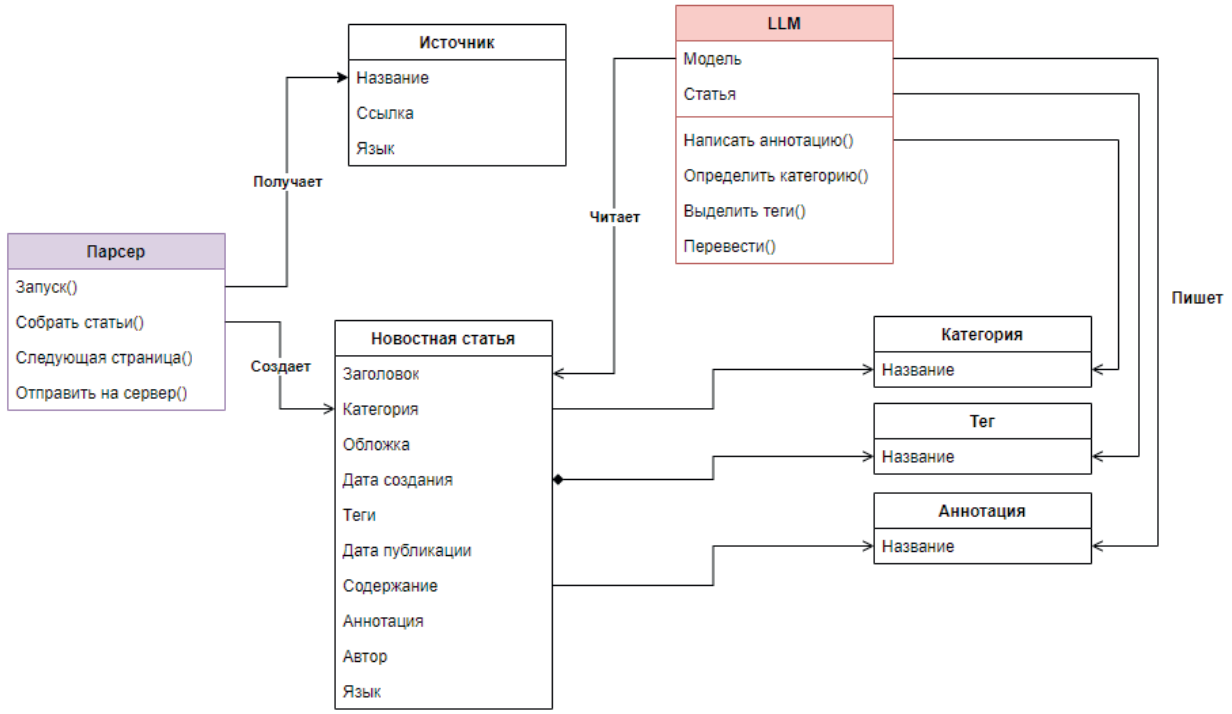

Рис. 1. Концептуальная диаграмма веб-платформы
Примечание: составлено авторами.

3. LLM:

Атрибуты: Модель, Статья

Методы: Написать аннотацию(), Определить категорию(), Выделить теги(), Перевести()

Функции: OpenChat обрабатывает статьи собранные парсером, анализирует их наполнение, пишет аннотации, определяет категории, присваивает теги и переводит.

4. Новостная статья:

Атрибуты: Заголовок, Категория, Обложка, Дата создания, Теги, Дата публикации, Содержание, Аннотация, Автор, Язык.

Функции: Этот объект содержит информацию о новостной статье, включая заголовок, дату создания, ссылку на источник, обложку, содержание, категорию, теги, аннотацию и язык.

5. Категория:

Атрибуты: Название

Функции: Включает в себя категорию статьи.

6. Тег:

Атрибуты: Название

Функции: Содержит теги, связанные с содержанием статьи.

7. Аннотация:

Атрибуты: Контекст

Функции: Хранит краткое описание содержания статьи.

Для более глубокого понимания архитектуры платформы и ее элементов, представим диаграмму, на которой отображены ключевые компоненты системы и их связи.

На рис. 2 представлены основные компоненты системы для анализа и классификации новостных статей, запущенные в контейнерах Docker. Каждый контейнер специализируется на выполнении своей конкретной задачи. Для некоторых групп контейнеров были настроены сетевые параметры для полной изоляции одних групп от других. Например, контейнер с работающим сервером базы данных (БД) находится в одной сети только с контейнером «golang-server», так что другие контейнеры, работающие с данными из БД, должны обращаться к этому контейнеру через соответствующие запросы к API.

Веб-интерфейс позволяет пользователям взаимодействовать с системой через браузер и HTTP-запросы. Клиентские запросы направляются на фронтенд, расположенный

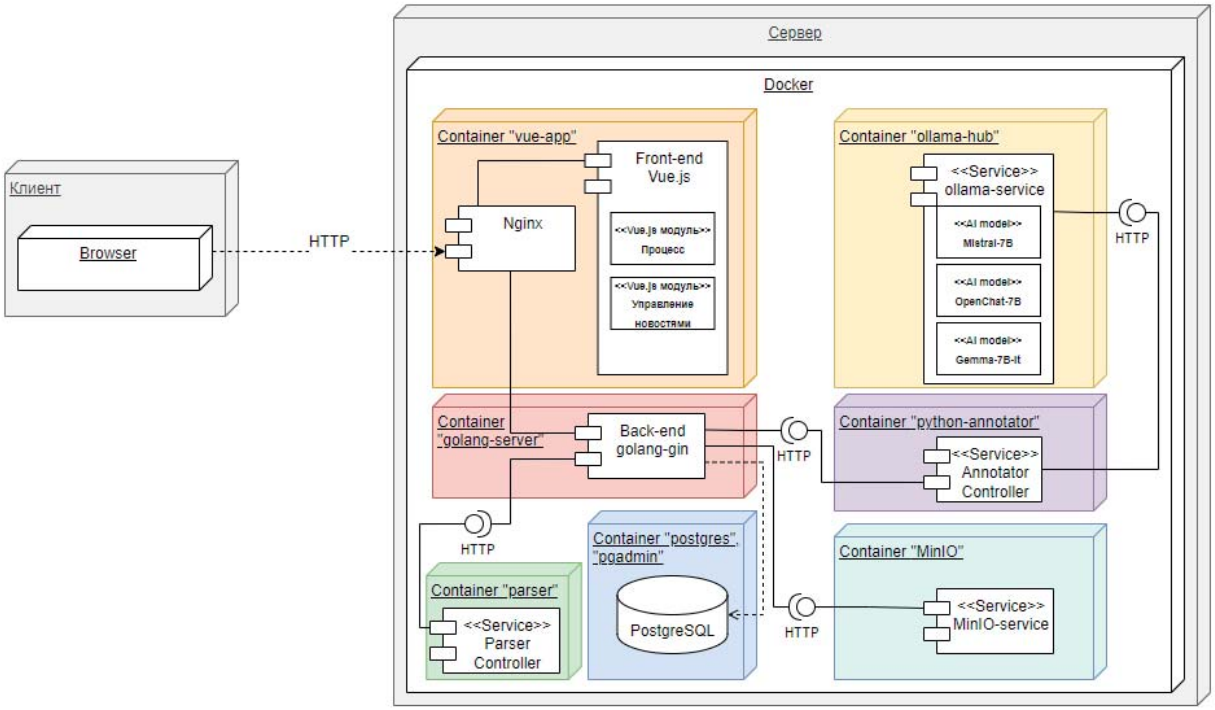

Рис. 2. Диаграмма компонентов
Примечание: составлено авторами.

в контейнере «vue-app». Nginx обрабатывает HTTP-запросы, а приложение на Vue.js предоставляет возможность просмотра новостных статей пользователю.

Используя контейнеризацию, можно упаковать каждый сервис со всеми его зависимостями в отдельное изолированное окружение для обеспечения единообразия среды как на этапе разработки, так и при развертывании и поддержке платформы. Кроме того, это позволяет избежать конфликтов между зависимостями и различными версиями программного обеспечения.

Рассмотрим классификацию новостных статей с использованием больших языковых моделей. Статья может быть отнесена к одной из следующих категорий: криптовалюта, конфиденциальность, безопасность и технологии. Эти категории хранятся в базе данных, в рис. 3. Значение «theme_id» используется как внешний ключ в соответствующем поле таблицы «articles».

	theme_id [PK] bigint	theme_name character varying (255)
1	1	technology
2	2	crypto
3	3	privacy
4	4	security

Рис. 3. Таблица «themes»
Примечание: составлено авторами.

После того как парсер соберет данные с веб-страницы, он проверяет все поля на заполненность, в случае успешной проверки, передаст их сервису. Сервис отправляет информацию на сервер для добавления в базу данных (табл. 1). После ответа от сервера, ссылка на статью удаляется из очереди сервиса.

Сервис аннотации запрашивает необработанные статьи с сервера каждые 4 часа. Сервер, взаимодействующий с базой данных, выполняет запрос (рис. 4) для поиска статей без аннотаций на определенных языках. Полученная информация о статьях (их исходном и отсутствующем языке аннотации) сортируется по дате публикации; затем сервер возвращает результат (рис. 5).

Сервис аннотаций отправляет запрос ollama-hub, фреймворк предназначен для запуска и управления большими языковыми моделями (LLM) на локальных вычислительных ресурсах, обеспечивающий возможность загрузки и развертывания выбранной LLM, а также доступ к ней через API, используя имя и параметры модели (рис. 6), запрос с заголовком статьи, содержанием и тематиками, к одной из которых можно отнести наполнение статьи.

После получения ответа требуется валидация результата нейронной сети (рис. 7). Для этого необходимо проверить наличие каждой тематики в ответе. В случае если совпадение было найдено, устанавливается тематика статьи, в противном случае, если совпадения не были найдены, то есть в ответе большой

Таблица 1

Некоторые данные из собранных новостных статей

Заголовок	Автор	Тематика	Язык	Дата публикации	Дата добавления
Spotify is once again hiking prices in the US	Paulius Grinkevičius	null	English	2024-06-03 13:35:00	2024-06-08 18:55:52.001527
Microsoft Edge introduces AI feature to improve…	Konstancija Gasaitytė	null	English	2024-05-22 11:54:00	2024-06-08 18:58:30.764772
Intel unveils its Lunar Lake chip for AI PCs	Paulius Grinkevičius	null	English	2024-06-04 08:56:00	2024-06-08 18:55:43.879135
Muslim Tinder exposes secrets, risks…	Vilius Petkauskas	null	English	2024-05-31 06:32:00	2024-06-08 18:56:36.649774
OneCoin fraudster gets ten years for laundering $400M	Damien Black	null	English	2024-01-26 09:40:00	2024-06-08 13:00:09.248526

Примечание: составлено авторами.

```
query := `
SELECT
    a.id AS article_id,
    ln.language_code AS native_language,
    l.language_code,
    l.language_name
FROM
    articles a
CROSS JOIN
    languages l
LEFT JOIN
    languages ln ON a.language_id = ln.language_id
LEFT JOIN
    annotations an ON a.id = an.article_id AND l.language_id = an.language_id
WHERE
    an.article_id IS NULL
ORDER BY a.post_date DESC;`
```

Рис. 4. Запрос нахождение статей без аннотации
Примечание: составлено авторами.

JSON
```
{
  "data": [
    {
      "article_id": "a484f6d1-110e-4569-997e-4e33ef3e93e8",
      "native_language": "en-US",
      "language_code": "en-US",
      "language_name": "English"
    },
    {
      "article_id": "a484f6d1-110e-4569-997e-4e33ef3e93e8",
      "native_language": "en-US",
      "language_code": "ru-RU",
      "language_name": "Russian"
    },
    {
      "article_id": "a484f6d1-110e-4569-997e-4e33ef3e93e8",
      "native_language": "en-US",
      "language_code": "fr-FR",
      "language_name": "French"
    },
    {
      "article_id": "a484f6d1-110e-4569-997e-4e33ef3e93e8",
      "native_language": "en-US",
      "language_code": "de-DE",
      "language_name": "German"
    },
    {
      "article_id": "b3b7c39b-90e5-4642-8c31-ec7d15d561bc",
      "native_language": "en-US",
      "language_code": "en-US",
      "language_name": "English"
    },
    {
      "article_id": "b476e0db-1136-41c5-972f-f42c5b9cd3b2",
      "native_language": "en-US",
      "language_code": "fr-FR",
      "language_name": "French"
    },
```

Рис. 5. Ответ сервера, список необработанных статей
Примечание: составлено авторами.

```
def __init__(self):
    super().__init__()
    self.model_name = "openchat"
    self.data = {
        "stream": False,  # Placeholder for stream option
        "options": {
            "temperature": 0.8,  # Placeholder for temperature option
            "top_p": 0.9
        }
    }
```

Рис. 6. Название модели и параметры большой языковой модели
Примечание: составлено авторами.

```
words = ["technology", "crypto", "privacy", "security"]
prompt = """
    Article:
    '''
    Title: %s
    Content: %s
    '''
    Classify the article to one of the topics:
    %s
    Use template to asnwer:
    Answer template: "Topic: topic 1"
    """
prompt = prompt % (article.title, article.body, ", ".join(words))
```

Рис. 7. Шаблон запроса для большой языковой модели
Примечание: составлено авторами.

языковой модели нет ни одной из указанной тематик, устанавливается значение «None». В дальнейшем перед отправкой на сервер для добавления в базу данных это позволит определить, корректно ли была обработана новостная статья, избегая некорректных записей.

Определение темы новостной статьи – один из обязательных этапов ее обработки (рис. 8). После загрузки статьи в систему необходимо выполнить ряд ключевых процедур, таких как аннотирование, перевод (в случае если язык статьи отличается от целевого языка, получаемого из ответа), категоризация и присвоение тегов (рис. 5). Эти действия позволяют группировать статьи по ряду параметров и повышают доступность для пользователей (табл. 2).

```
answer = self.generate(prompt=prompt, stream=stream, options=options)

answer = answer.message["content"].lower()

# Check if any of the words are present in the string
for word in words:
    if word in answer:
        article.theme_name = word
        break
else:
    article.theme_name = None
```

Рис. 8. Вызов метода генерации и валидация ответа нейронной сети
Примечание: составлено авторами.

ЗАКЛЮЧЕНИЕ

Применение микросервисной архитектуры кажется достаточно обоснованным и перспективным решением, поскольку это позволяет эффективно управлять различными компонентами системы, обеспечивая их изолированность и гибкость. Каждый микросервис может быть создан, внедрен и масштабирован индивидуально, что упрощает процесс разработки, обновления и поддержки всей платформы. На основе анализа популярных подходов к построению ПО, можно сделать вывод о целесообразности использования микросервисной архитектуры для разрабатываемой платформы.

Дальнейшее развитие веб-платформы может включать оптимизацию производительности и масштабируемости системы, параллельное выполнение процессов парсинга и аннотирования данных, а также использование кэширования для ускорения доступа к часто запрашиваемой информации. Также имеется возможность расширения числа источников новостных статей за счет добавления дополнительных информационных ресурсов, специализирующихся на области информационных технологий.

Таблица 2

Некоторые данные из аннотированных новостных статей

Заголовок	Автор	Тематика	Язык	Дата публикации	Дата добавления
Spotify is once again hiking prices in the US	Paulius Grinkevičius	technology	English	2024-06-03 13:35:00	2024-06-08 18:55:52.001527
Microsoft Edge introduces AI feature to improve…	Konstancija Gasaitytė	security	English	2024-05-22 11:54:00	2024-06-08 18:58:30.764772
Intel unveils its Lunar Lake chip for AI PCs	Paulius Grinkevičius	technology	English	2024-06-04 08:56:00	2024-06-08 18:55:43.879135
Muslim Tinder exposes secrets, risks…	Vilius Petkauskas	privacy	English	2024-05-31 06:32:00	2024-06-08 18:56:36.649774
OneCoin fraudster gets ten years for laundering $400M	Damien Black	crypto	English	2024-01-26 09:40:00	2024-04-29 13:00:09.248526

Примечание: составлено авторами.

Список источников

1. Big Data Statistics 2023: How Much Data is in The World? URL: https://firstsiteguide.com/big-data-stats (дата обращения: 25.06.2024).
2. Big data – statistics & facts. URL: https://www.statista.com/topics/1464/big-data (дата обращения: 25.06.2024).
3. Docker: как создавать образы контейнеров и развертывать приложения. URL: https://habr.com/ru/articles/776188 (дата обращения: 25.06.2024).
4. В чем разница между монолитной архитектурой и архитектурой микросервисов? URL: https://web.archive.org/web/20240622123247/https://aws.amazon.com/ru/compare/the-difference-between-monolithic-and-microservices-architecture (дата обращения: 25.06.2024).
5. 4 типа архитектуры программного обеспечения. URL: https://nuancesprog.ru/p/12019 (дата обращения: 26.06.2024).
6. Microservices architecture design. URL: https://learn.microsoft.com/en-us/azure/architecture/microservices (дата обращения: 25.06.2024).
7. Артамонов Ю. С., Востокин С. В. Разработка распределенных приложений сбора и анализа данных на базе микросервисной архитектуры // Известия Самарского научного центра Российской академии наук. 2016. Т. 18, № 4–4. С. 688–693.
8. Вальдивия Х. А., Лора-Гонсалес А., Лимон К. и др. Паттерны микросервисной архитектуры: многопрофильный обзор литературы // Труды Института системного программирования РАН. 2021. Т. 33, № 1. С. 81–96.
9. Что такое сервис-ориентированная архитектура (SOA)? URL: https://web.archive.org/web/20240622115612/https://aws.amazon.com/ru/what-is/service-oriented-architecture (дата обращения: 25.06.2024).
10. Щекочихин О. В., Черкасова Н. В. Анализ шаблонов проектирования информационных систем клиент-серверной архитектуры // Информационно-экономические аспекты стандартизации и технического регулирования. 2019. № 5. С. 26–29.

References

1. Big Data Statistics 2023: How Much Data is in The World? URL: https://firstsiteguide.com/big-data-stats (accessed: 25.06.2024).
2. Big data – statistics & facts. URL: https://www.statista.com/topics/1464/big-data (accessed: 25.06.2024).
3. Docker: kak sozdavat obrazy konteynerov i razvertyvat prilozheniya. URL: https://habr.com/ru/articles/776188 (accessed: 25.06.2024). (In Russ.).
4. V chem raznitsa mezhdu monolitnoy arkhitekturoy i arkhitekturoy mikroservisov? URL: https://web.archive.org/web/20240622123247/https://aws.amazon.com/ru/compare/the-difference-between-monolithic-and-microservices-architecture (accessed: 25.06.2024). (In Russ.).
5. 4 tipa arkhitektury programmnogo obespecheniya. URL: https://nuancesprog.ru/p/12019 (accessed: 26.06.2024). (In Russ.).
6. Microservices architecture design. URL: https://learn.microsoft.com/en-us/azure/architecture/microservices (accessed: 25.06.2024).
7. Artamonov Y. S., Vostokin S. V. Development of distributed applications for data collection and analysis on the basis of a microservice architecture. *Izvestia of Samara Scientific Center of the Russian Academy of Sciences*. 2016;18(4–4):688–693. (In Russ.).
8. Valdivia J. A., Lora-González A., Limón X. et al. Patterns Related to Microservice Architecture: A Multivocal Literature Review. *Trudy ISP RAN/Proc. ISP RAS*. 2021;33(1):81–96. (In Russ.).
9. Chto takoe servis-orientirovannaya arkhitektura (SOA)? URL: https://web.archive.org/web/20240622115612/https://aws.amazon.com/ru/what-is/service-oriented-architecture (accessed: 25.06.2024). (In Russ.).
10. Shchekochikhin O. V., Cherkasova N. V. Analysis of templates of designing information systems of client-server architecture. *Informatsionno-ekonomicheskiye aspekty standartizatsii i tekhnicheskogo regulirovaniya.* 2019;(5):26–29. (In Russ.).

Информация об авторах

К. В. Кузнецов – бакалавр.

С. А. Лысенкова – кандидат физико-математических наук, доцент.

About the authors

K. V. Kuznetsov – Bachelor's Degree Student.

S. A. Lysenkova – Candidate of Sciences (Physics and Mathematics), Associate Professor.