https://doi.org/10.26160/2474-5901-2024-40-75-79

ПЕРСПЕКТИВЫ РАЗВИТИЯ АРХИТЕКТУРЫ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Андреев Р.А., Дулесов А.С.
Хакасский государственный университет им. Н.Ф. Катанова, Абакан, Россия

Ключевые слова: контейнеризация, масштабируемость, микросервисы, портабельность, развертывание, облачный сервис, отказоустойчивость, искусственный интеллект.
Аннотация. Рассматриваются перспективы развития архитектуры программного обеспечения, связанные с потребностями в изменении способов проектирования и разработки программных продуктов. Представлен краткий анализ текущих тенденций и совершенствующихся технологий в области разработки программного обеспечения, такие как микросервисная архитектура, контейнеризация, применение искусственного интеллекта и обновления программного обеспечения.

PROSPECTS FOR THE DEVELOPMENT OF SOFTWARE ARCHITECTURE

Andreev R.A., Dulesov A.S.
Katanov Khakass State University, Abakan, Russia

Keywords: containerization, scalability, microservices, portability, deployment, cloud service, fault tolerance, artificial intelligence.
Abstract. The prospects for the development of software architecture related to the need to change the ways of designing and developing software products are considered. A brief analysis of current trends and evolving technologies in software development, such as microservice architecture, containerization, artificial intelligence, and software updates, is presented.

Архитектура программного обеспечения является ключевым фактором в разработке современных приложений. С каждым годом появляются новые технологии и подходы, которые могут привнести новые перспективы в развитие программного обеспечения.

Одним из ключевых трендов в развитии архитектуры программного обеспечения является переход от монолитных систем к микросервисной архитектуре. Микросервисы представляют собой небольшие, независимые компоненты, которые могут разрабатываться, развертываться и масштабироваться независимо друг от друга. Этот подход позволяет улучшить гибкость и масштабируемость приложений, а также облегчить их развертывание и сопровождение.

В монолитной системе, все компоненты находятся в едином кодовой базе и взаимодействуют между собой напрямую. Однако, в микросервисной архитектуре, каждый сервис представляет собой отдельное приложение, что создает необходимость установления коммуникации и взаимодействия между сервисами.

Организация связей между сервисами может выполняться с помощью различных протоколов и инструментов, таких как HTTP, REST API, message

queues и т.д. Необходимо разработать четкую архитектурную модель, определить точки взаимодействия сервисов и протоколы коммуникации.

Управление зависимостями является еще одним важным аспектом микросервисной архитектуры. Каждый сервис имеет свои собственные зависимости – библиотеки, инструменты и другие модули, необходимые для его функционирования. Контролировать версии и обновления этих зависимостей в рамках такой архитектуры может быть сложно. Необходимо разработать стратегию управления зависимостями, которая позволит эффективно обновлять и внедрять изменения в каждом сервисе без привязки к другим зависимостям.

Однако необходимо иметь в виду, что переход к данной архитектуре требует дополнительных усилий в организации связей и управлении зависимостями между сервисами. Дополнительные усилия также требуются для обеспечения надежности и мониторинга микросервисной системы. Распределенная природа архитектуры означает, что возможны отказы и сбои в отдельных сервисах. Поэтому необходимо разработать механизмы отслеживания и регистрации состояния каждого сервиса, а также мониторинга и оповещения об ошибке [1].

В качестве примера можно предложить систему электронной коммерции, разработанную по принципу микросервисной архитектуры. В ней может быть несколько сервисов, таких как сервис аутентификации, сервис корзины покупок, сервис обработки платежей и сервис доставки. Для обеспечения взаимодействия между сервисами, следует определить протоколы коммуникации, а также установить и настроить соответствующие зависимости, такие как библиотеки для работы с базой данных или системой платежей. Также востребованы механизмы мониторинга для отслеживания состояния каждого сервиса и обнаружения возможных ошибок или сбоев.

Предлагаемые здесь дополнительные усилия необходимы для успешной реализации микросервисной архитектуры и обеспечения ее гибкости, масштабируемости и надежности.

Следующим направлением в развитии архитектуры программного обеспечения является использование контейнеров. Контейнеризация позволяет упаковать приложение и все его версии пакетов (т.е. модулей, библиотек) в изолированную среду, что обеспечит портабельность (Portability – переносимость программного обеспечения) приложения и упрощает его развертывание на различных платформах.

Технологии, такие как Docker – одна из популярных в контейнеризации, обеспечивают легкую установку и управление контейнерами. Docker предоставляет инструменты и платформу для создания и управления контейнерами, которые позволяют упаковать приложение и его зависимости в изолированную среду. Контейнеризация в Docker основывается на концепции контейнеров, которые представляют собой легковесные, автономные и изолированные экземпляры приложений, которые работают внутри общей операционной системы. Контейнеры изолированы друг от друга и от хост-

системы, что позволяет запускать их без конфликтов и непосредственно переносить между различными средами выполнения.

Использование Docker и контейнеризации в архитектуре программного обеспечения открывает новые возможности для разработчиков, делая процесс разработки и развертывания более удобным, быстрым и масштабируемым. Это помогает снизить риски и издержки, связанные с развертыванием и поддержкой приложений, и способствует созданию более надежных и гибких систем.

Еще одним направлением, которое представляет интерес, является архитектура с фокусом на кибербезопасность. Количество угроз растет, сдледовательно защита приложений от внешних атак становится все более важной задачей. Современная архитектура программного обеспечения должна включать в себя механизмы защиты данных, контроля доступа и обнаружения угроз. Это может включать в себя шифрование данных, применение протоколов безопасности и реализацию многоуровневой защиты. Снижая риски такого рода угроз, можно добиться значительно уменьшения ущерба, связанного с нарушением качества и потерей информации [2].

Следующим из наиболее перспективных направлений в развитии программного обеспечения является искусственный интеллект (ИИ). Применение алгоритмов машинного обучения и нейронных сетей предоставляет уникальные возможности для оптимизации и улучшения производительности различных приложений.

Алгоритмы машинного обучения позволяют системам обучаться на основе большого объема данных и выявлять скрытые закономерности. Это позволяет автоматически обнаруживать и исправлять ошибки в программном коде, что ускоряет разработку и обеспечивает более стабильную работу приложений. Кроме того, ИИ может помочь оптимизировать процессы в самом приложении, например, улучшая алгоритмы рекомендаций или оптимизируя работу с базами данных.

Применение искусственного интеллекта (ИИ) в архитектуре программного обеспечения направлено на предоставление персонализированного пользовательского опыта. При этом ИИ приложения могут «понимать» предпочтения пользователей, анализировать их поведение и предоставлять уникальные рекомендации или контент с учетом этих предпочтений. Тем самым можно добиться улучшение вовлеченности пользователей в создание более удобных и интересных продуктов.

Кроме того, ИИ позволит сократить затраты на тестирование и отладку программного обеспечения. Путем автоматизации процесса тестирования и самообучения моделей ИИ можно уменьшить количество ошибок и снизить риск возникновения проблем в работе приложения.

Применение ИИ в архитектуре программного обеспечения представляет значительный потенциал для оптимизации и улучшения производительности приложений, обеспечения персонализированного пользовательского опыта и сокращения затрат на разработку и отладку. Множество компаний и

организаций уже успешно используют ИИ в своих продуктах, и этот тренд, безусловно, будет только расти в будущем [3].

Кроме этого, одним из важных аспектов в развитии архитектуры программного обеспечения является учет требований к производительности, масштабируемости и отказоустойчивости.

Масштабируемость связана с расширением обработки все больших объемов данных необходимых пользователям. Например, горизонтальное масштабирование позволит распределять нагрузку на несколько серверов или узлов, обеспечивая более эффективную работу и увеличивая пропускную способность системы.

Отказоустойчивость связана с таким свойством архитектуры как надежность [4]. Здесь важно применят подходы, которые на основе статистики, теории вероятности и меры неопределенности информации. Они позволят на основе разработанных технологий и приложений обнаруживать сбои, резервировать ресурсы и обеспечивать непрерывную работу сетей даже при возникновении проблем.

Важным аспектом выбора архитектуры и технологий в разработке программного обеспечения является обеспечение согласованности и совместимости компонентов системы. Различные сервисы и модули приложения должны взаимодействовать между собой без конфликтов и зависимостей. Эффективность выбора будет зависеть от разработки качественных методов и алгоритмов, базирующихся на теориях вероятности, информации, когнитивизма и принятия решений.

Системный анализ требований к применению соответствующих архитектурных решений и технологий помогет создать эффективные и надежные приложения для их успешного функционировать в различных условиях.

Таким образом, развитие архитектуры программного обеспечения предоставляет широкий спектр возможностей для улучшения процессов разработки и проектирования приложений. Микросервисная архитектура, контейнеризация, кибербезопасность, искусственный интеллект и требования к производительности играют ключевую роль в формировании будущего архитектуры программного обеспечения. Учет данных решений позволит разработчикам обозначить перспективы и выбирать подходы, соответствующие требованиям и целям их проектов.

Список литературы

1. Рик Казман, Лен Басс, Пол Клементс, Архитектура программного обеспечения на практике. – 3-е изд. – Addison-Wesley Professional, 2012. – 576 с.
2. Ник Розански, Эоин Вудс, Архитектура программных систем. – Addison-Wesley Professional, 2005. – 317 с.
3. Роберт Мартин, Чистая архитектура. Искусство разработки программного обеспечения. – СПб.: Питер, 2022. – 352 с.
4. Жозеф Ингено. Руководство архитектора программного обеспечения. – Packt Publishing, 2018 – 594 с.

References

1. Rick Kazman, Len Bass, Paul Clements, Software Architecture in practice. – 3rd edition. – Addison-Wesley Professional, 2012. – 576 p.
2. Nick Rozanski, Eoin Woods, Architecture of software systems. – Addison-Wesley Professional, 2005, 317 p.
3. Robert Martin, Pure Architecture. The Art of Software Development. – SPb.: Piter, 2022. – 352 p.
4. Joseph Ingeno. Software Architect's Guide. – Packt Publishing, 2018. – 594 p.

Андреев Роман Андреевич – аспирант	**Andreev Roman Andreevich –** postgraduate student
Дулесов Александр Сергеевич – доктор технических наук, доцент, профессор кафедры Цифровых технологий и дизайна	**Dulesov Alexander Sergeevich –** doctor of technical sciences, associate professor, professor of the Department of digital technologies and design
grand-roman@yandex.ru	

Received 21.02.2024