Столыпинский вестник

Научная статья
Original article
УДК 004.4

РОЛЬ АРХИТЕКТУРНОГО ПОДХОДА ПРИ РАЗРАБОТКЕ ПРОГРАММНЫХ ПРИЛОЖЕНИЙ

THE ROLE OF THE ARCHITECTURAL APPROACH IN THE DEVELOPMENT OF SOFTWARE APPLICATIONS

Егорян В.В., Студент бакалавриата 3 курс, МИРЭА-Российский технологический университет (РТУ МИРЭА), 119454, Россия, г. Москва, проспект Вернадского, 78, Институт информационных технологий, Россия, г. Москва

Калугин А.В., Студент бакалавриата 3 курс, МИРЭА-Российский технологический университет (РТУ МИРЭА), 119454, Россия, г. Москва, проспект Вернадского, 78, Институт информационных технологий, Россия, г. Москва

Egoryan V.V., Undergraduate student 3rd year, MIREA-Russian Technological University (RTU MIREA), 119454, Russia, Moscow, Vernadsky Avenue, 78, Institute of Information Technology, Russia, Moscow

Kalugin A.V., Undergraduate student 3rd year, MIREA-Russian Technological University (RTU MIREA), 119454, Russia, Moscow, Vernadsky Avenue, 78, Institute of Information Technology, Russia, Moscow

Аннотация: Термин архитектуры приложений является новым относительно сферы разработки программного обеспечения. При разработке

программных приложений разработчики сталкиваются с трудностями организации работы элементов систем и модулей приложения. Архитектура приложений и данных значительно облегчает процесс разработки и делит его на разграниченные комплексы мероприятий.

Abstract: The term application architecture is new to the field of software development. While developing software applications, developers face difficulties in organizing the work of system elements and application modules. The architecture of applications and data greatly facilitates the development process and divides it into delimited sets of activities.

Ключевые слова: Архитектура, приложения, данные.

Keywords: Architecture, application, data.

Настоящая статья посвящена теме роли архитектурного подхода при разработке приложений и данных.

Понятие архитектуры подразумевает наличие шаблонов и методик, которые гарантируют устойчивое и бесперебойное функционирование приложения, а также обеспечивает структурное хранение данных, исключая риск возникновения утечек данных и несанкционированную обработку.

Целью исследования является выяснение роли архитектурного подхода при разработке приложений.

Объектом исследования является архитектурный подход разработки приложений.

Предмет исследования

Роль архитектурного подхода при разработке программного приложения.

Практическая значимость исследования

Использование различных шаблонов архитектур позволяет облегчить разработку, разделить обязанности в команде разработчиков, оптимизировать пути взаимодействия между разработчиками приложения.

Определение архитектуры разработки приложений и данных

Архитектура программного обеспечения – совокупность стратегически важных решений для организации системы. Это набор различных методов, шаблонов, моделей и парадигм, которые обеспечивают разработку структурированного и гибкого к изменениям приложения. Это также набор стандартов, правил и политик для определения вида данных, способа их хранения, размещения и интеграции. Цели существования архитектуры приложений и данных – уменьшение человеческих трудозатрат на создание и сопровождение системы, упрощение развертывания, поддержка жизненного цикла информационной системы. Хорошо спроектированная архитектура обеспечивает системе легкость в освоении.

Архитектура приложений и данных за все время существования выделяет несколько основных видов: монолитную, клиент-серверную, трехуровневую, распределенную, сервисную.

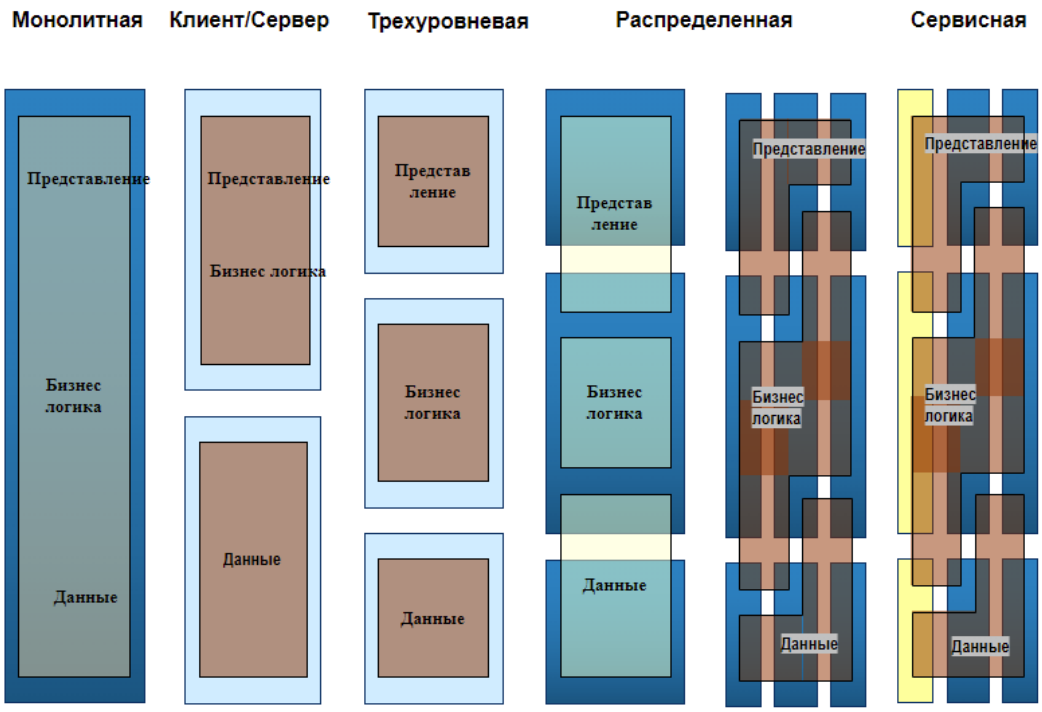

Рисунок 1 – Основные виды архитектуры приложений и данных

Монолитная архитектура

Монолитное приложение представляет собой приложение, доставляемое через единое развертывание. Все элементы будут управляться внутри одного и того же модуля.

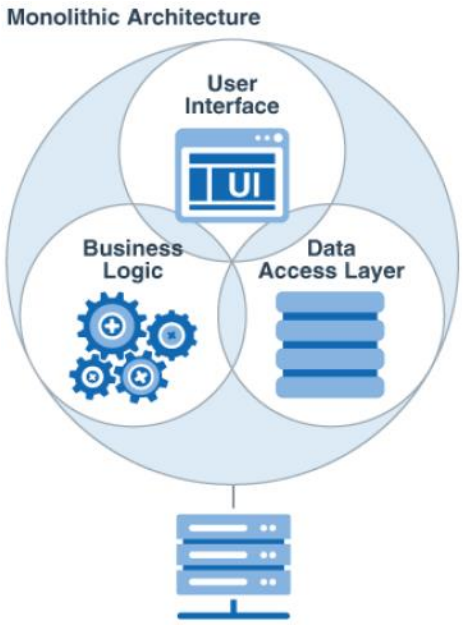

Рисунок 2 – Пример монолитной архитектуры приложений и данных

Из преимуществ монолитной архитектуры можно выделить легкость реализации самого приложения и легкость реализации бизнес-логики [4].

Также монолитная архитектура значительно упрощает сквозное тестирование, при котором происходит эмуляция опыта использования приложения пользователем. Монолитный подход обеспечивает простоту развертывания и легкость масштабируемости, подход также крайне прост в эксплуатации.

Из недостатков можно выделить отсутствие изоляции между модулями, таким образом ошибка в одном из модулей может вызвать замедление или ошибку во всем приложении. При большем масштабе приложения, добавление новых модулей, функций или работа с отдельно взятым модулем может вызвать затруднения, становится невозможно масштабировать отдельные части системы.

Двухуровневая клиент-серверная архитектура

Распределенная архитектура включает в себя виды клиент-серверной архитектуры. Клиент-серверная архитектура приложений и данных обеспечивает разделение отправки запросов и распределение процессов на разные потоки, которые функционируют независимо друг от друга.

Характеристики клиент-серверной архитектуры приложений и данных [2]:

- масштабируемость;
- адаптивность к расширению;
- прозрачность;
- инкапсуляция услуг;
- асимметричность протоколов;
- целостность;
- независимость от платформы.

Клиент-серверные архитектуры бывают нескольких типов: одноуровневыми, двухуровневыми, трехуровневыми и многоуровневыми. Пример клиент-серверной архитектуры представлен на Рисунке 3.

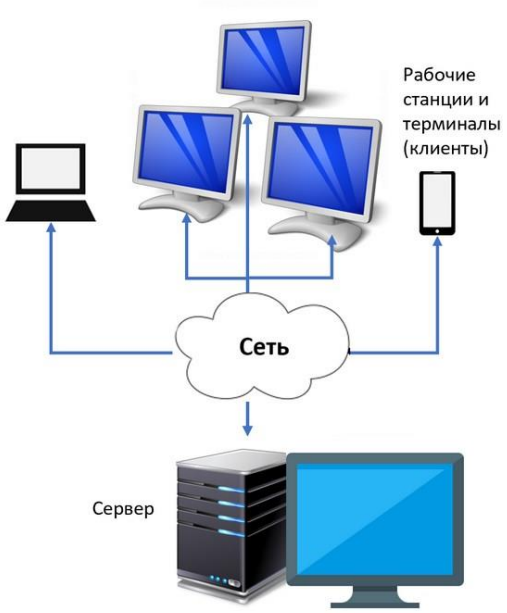

Рисунок 3 – Пример клиент-серверной архитектуры

Одноуровневая клиент-серверная архитектура

Одноуровневая клиент-серверная архитектура подразумевает наличие сервера базы данных, на котором находятся необходимые данные и рабочих станций, на которых находится программное обеспечение. Из преимуществ одноуровневой архитектуры можно выделить надежность, однако данная архитектура может вызвать проблему синхронизации данных на отдельных машинах. Пример одноуровневой клиент-серверной архитектуры представлен на Рисунке 4.

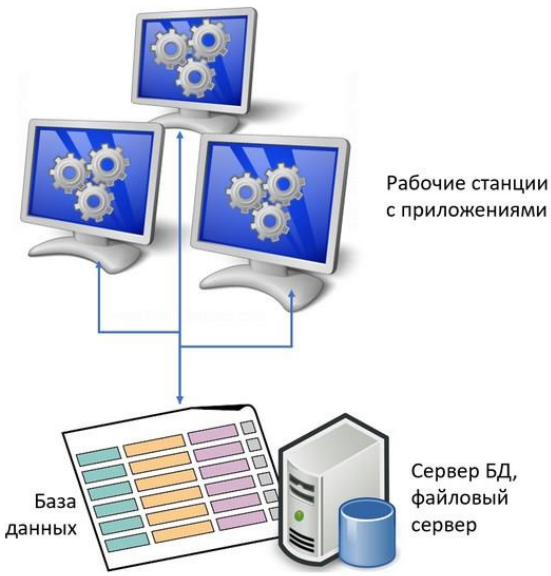

Рисунок 4 – Пример одноуровневой клиент-серверной архитектуры

Двухуровневая клиент-серверная архитектура

Двухуровневая клиент-серверная архитектура подразумевает наличие сервера, на котором находится программное обеспечение и рабочих станций, которым предоставлен интерфейс, работа с данными производится непосредственно на самом сервере. Двухуровневую клиент-серверную архитектуру можно разделить на два вида: толстый клиент-тонкий сервер и тонкий клиент-толстый сервер. Толстый клиент подразумевает, что приложение обрабатывает данные на рабочей станции, а тонкий клиент подразумевает хранение этих данных на сервере. Тонкий клиент, толстый сервер обеспечивает обработку и хранение данных на сервере, эта модель является практическим представлением облачных вычислений, которые на

сегодняшний день набирают большую популярность в разработке ИТ-продуктов. Пример двухуровневой клиент-серверной архитектуры представлен на Рисунке 5.

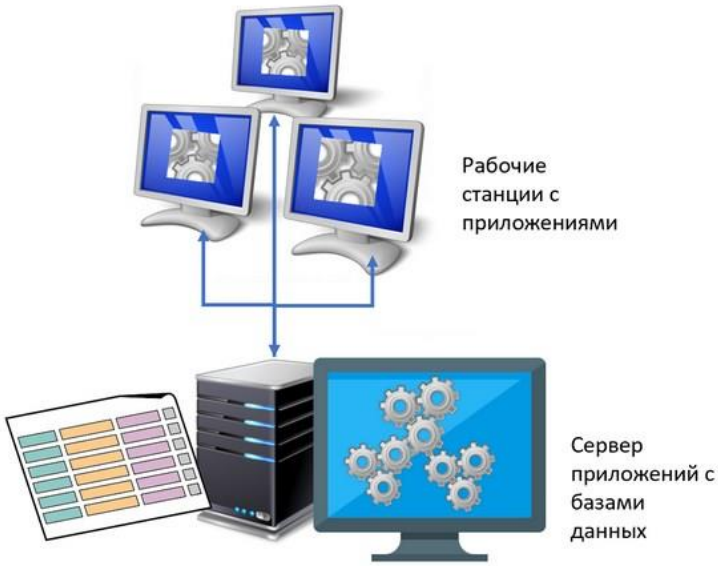

Рисунок 5 – Пример двухуровневой клиент-серверной архитектуры приложений и данных

Из преимуществ двухуровневой архитектуры приложений можно выделить легкость конфигурации и модификации приложения, легкость эксплуатации системы, адаптивность к масштабированию и высокую производительность.

Из недостатков выделяются низкая производительность при большой нагрузке обращений к серверу и низкий уровень безопасности, поскольку все данные находятся на центральном сервере.

Трехуровневая клиент-серверная архитектура подразумевает разделение сервера на два уровня: сервер, на котором находится приложение и сервер, на котором находятся необходимые данные и база данных. Все обращения рабочих станций поступают в промежуточное программное обеспечение, сервер запрашивает данные с серверов и обрабатывает их на промежуточном сервере. Пример трехуровневой архитектуры приведен на Рисунке 6.

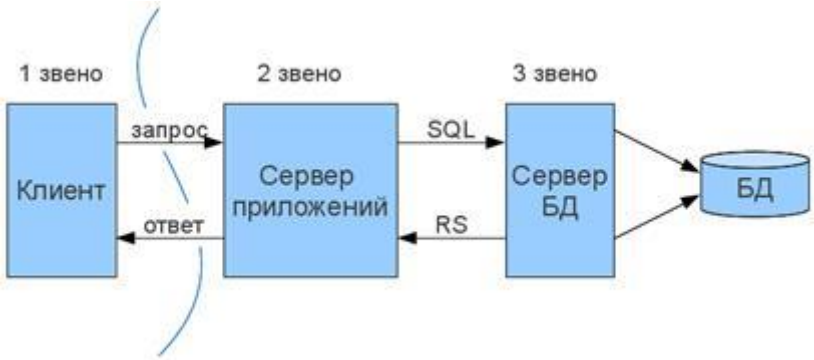

Рисунок 6 – Пример трехуровневой клиент-серверной архитектуры приложений и данных

Из преимуществ трехуровневой архитектуры можно выделить целостность данных, защищенность базы данных и более высокую безопасность в сравнении с двухуровневой.

Недостатком является более сложная структура коммуникации между клиентом и сервером.

Многоуровневая клиент-серверная архитектура

Многоуровневая клиент-серверная архитектура приложений и данных похожа на трехуровневую, она предусматривает большее разветвление серверов под отдельные нужды. Пример многоуровневой архитектуры приведен на Рисунке 7.

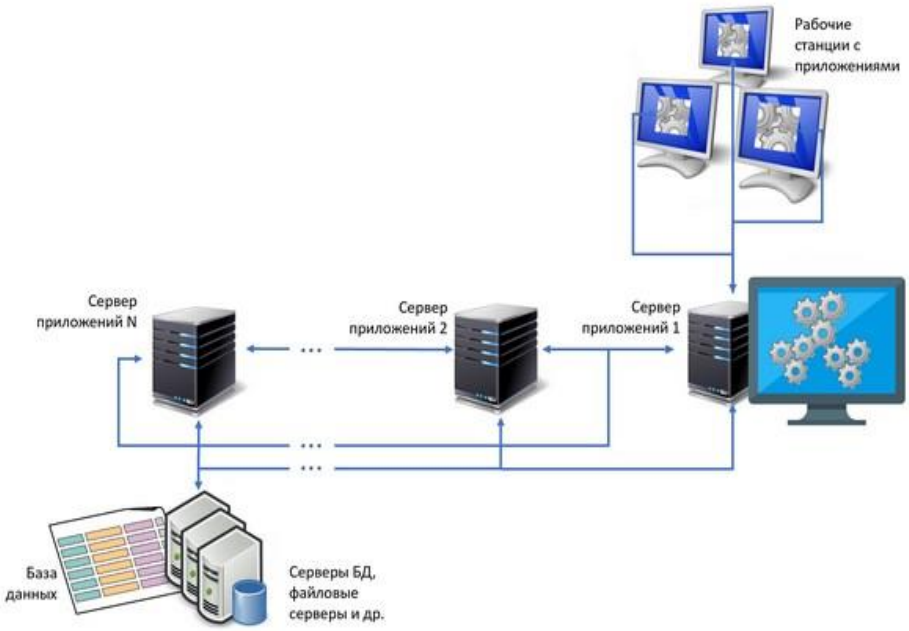

Рисунок 7 – Пример многоуровневой клиент-серверной архитектуры приложений и данных

Преимущества многоуровневой архитектуры идентичны преимуществам трехуровневой, а также можно выделить, что данная архитектура адаптивна к гибкому предоставлению услуг и совместной работе элементов и модулей приложения. Из недостатков можно выделить многокомпонентность и сложность архитектуры.

Распределенная архитектура

Распределенная архитектура приложений и данных подразумевает наличие клиентского приложения, корпоративных компонентов и серверов баз данных и наличие уровня представления. В рамках распределенной архитектуры обособлен интерфейс для представления пользователю, также свойственно использование виртуальных серверов, логическое распределение областей и модулей, использование большего количества серверов. Пример распределенной архитектуры представлен на рисунке 8.

Рисунок 8 – Пример распределенной архитектуры приложений и данных

Из преимуществ распределенной архитектуры приложений и данных можно выделить, повышенную надежность, эффективное использование ресурсов, гибкость разработки.

Микросервисная архитектура

Микросервисная архитектура – вид сервисно-ориентированной архитектуры приложений и данных, которая направлена на создание отдельных независимых компонентов, которые в совокупности составляют приложение. Этот подход ориентирован на возможности и бизнес-приоритеты. Каждый модуль функционирует в потоке не пересекаясь с другими процессами. Пример микросервисной архитектуры представлен на рисунке 9.

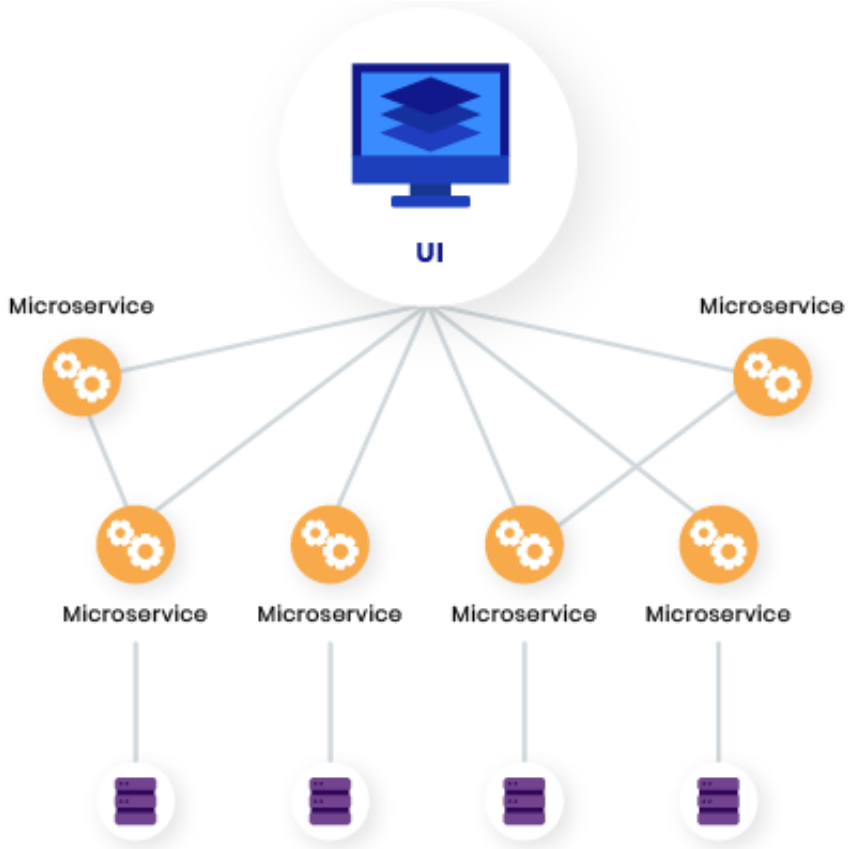

Рисунок 9 – Пример микросервисной архитектуры приложений и данных

Из преимуществ микросервисной архитектуры можно выделить легкость разработки, тестирования, развертки, повышенную гибкость и адаптивность к горизонтальному масштабированию.

Из недостатков можно выделить ограничение вертикального масштабирования, низкую безопасность и необходимость вести разработку на разных языках программирования.

Можно сделать вывод, что архитектурный подход разработки приложений и данных крайне важен для разработки программного приложения. Архитектурный подход играет ключевую роль для обеспечения качественной разработки устойчивого программного обеспечения.

Список используемых источников

1. Ю.П. Сурмин. Теория систем и системный анализ: учебное пособие / Ю.П. Сурмин. - Киев : МАУП, 2003. - 368 с. - (Бакалавр. Академический курс). - ISBN 966-608-290-X. - URL: https://studfile.net/preview/996456/ (дата обращения: 15.05.2022) - Режим доступа: Файловый архив студентов StudFiles.net. - Текст: электронный.
2. ITELON. Архитектура «Клиент-Сервер» : [сайт] - URL: https://itelon.ru/blog/arkhitektura-klient-server/ (дата обращения: 22.05.2022) - Текст: электронный.
3. martinFowler.com Bounded Context : [сайт] - URL: https://martinfowler.com/bliki/BoundedContext.html (дата обращения: 22.05.2022) - Текст: электронный.
4. Хабр. Лучшая архитектура для MVP: монолит, SOA, микросервисы. : [сайт] - URL: https://habr.com/ru/company/otus/blog/477930/ (дата обращения: 22.05.2022) - Текст: электронный.
5. Роберт М. Чистая архитектура. Искусство разработки программного обеспечения: книга / М. Роберт. - Санкт-Петербург : Питер, 2018. - 352 с. - ISBN 978-5-4461-0772-8. - Текст: непосредственный.

List of sources used

1. Yu.P. Surmin. Theory of systems and system analysis: a textbook / Yu.P. Surmin. - Kiev : IAPM, 2003. - 368 p. - (Bachelor. Academic course). - ISBN 966-608-290-X. - URL: https://studfile.net/preview/996456 / (accessed: 05/15/2022) - Access mode: Students' File Archive StudFiles.net . - Text: electronic.

2. ITELON. Client-Server Architecture : [site] - URL: https://itelon.ru/blog/arkhitektura-klient-server / (accessed: 05/22/2022) - Text: electronic.
3. martinFowler.com Bound Context : [site] - URL: https://martinfowler.com/bliki/BoundedContext.html (accessed: 05/22/2022) - Text: electronic.
4. Habr. The best architecture for MVP: monolith, SOA, microservices. : [website] - URL: https://habr.com/ru/company/otus/blog/477930 / (accessed: 05/22/2022) - Text: electronic.
5. Robert M. Pure Architecture. The Art of software development: a book / M. Robert. - St. Petersburg : Peter, 2018. - 352 p. - ISBN 978-5-4461-0772-8. - Text: direct.

Для цитирования: Егорян В.В., Калугин А.В., РОЛЬ АРХИТЕКТУРНОГО ПОДХОДА ПРИ РАЗРАБОТКЕ ПРОГРАММНЫХ ПРИЛОЖЕНИЙ// Научный сетевой журнал «Столыпинский вестник» №9/2022