ВВГУ

ФГБОУ ВО «Владивостокский государственный университет »

XXV

Материалы Международной научно-практической конференции студентов, аспирантов и молодых ученых

# ИНТЕЛЛЕКТУАЛЬНЫЙ ПОТЕНЦИАЛ ВУЗОВ – НА РАЗВИТИЕ ДАЛЬНЕВОСТОЧНОГО РЕГИОНА РОССИИ И СТРАН АТР

4–7 апреля 2023 г.

В четырех томах

Том 4

ISBN 978-5-9736-0711-1 (Т. 4)

9 785973 607111

Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение
высшего образования «Владивостокский государственный университет»

# ИНТЕЛЛЕКТУАЛЬНЫЙ ПОТЕНЦИАЛ ВУЗОВ – НА РАЗВИТИЕ ДАЛЬНЕВОСТОЧНОГО РЕГИОНА РОССИИ И СТРАН АТР

Материалы XXV международной научно-практической
конференции студентов, аспирантов и молодых ученых
4–7 апреля 2023 г.

Том 4

Под общей редакцией д-ра экон. наук Т.В. Терентьевой

Электронное научное издание

Владивосток
Издательство ВВГУ
2023

УДК 378.4
ББК 74.584(255)я431
И73

И73 **Интеллектуальный потенциал вузов – на развитие Дальневосточного региона России и стран АТР :** материалы XXV международной науч.-практ. конф. студентов, аспирантов и молодых ученых (г. Владивосток, 4–7 апреля 2023 г.) : в 4 т. Т. 4 / под общ. ред. д-ра экон. наук Т.В. Терентьевой ; Владивостокский государствен- ный университет ; Электрон. текст. дан. (1 файл: 12,0 МБ). – Владивосток: Изд-во ВВГУ, 2023. – 1 электрон., опт. диск (CD-ROM). – Систем. требо- вания: Intel Pentium (или аналогичный процессор других производителей), 500 МГц; 512 Мб оперативной памяти; видеокарта SVGA, 1280×1024 High Color (32 bit); 5 Мб свободного дискового пространства; операц. система Windows XP и выше; Acrobat Reader, Foxit Reader либо любой другой их аналог.

ISBN 978-5-9736-0711-1 DOI: https://doi.org//10.24666/0710-1

Включены материалы XXV международной научно-практической конференции студентов, аспирантов и молодых ученых «Интеллектуальный потенциал вузов – на развитие Дальневосточного региона России и стран Азиатско-Тихоокеанского региона», состоявшейся во Владивостокском государственном университете (г. Владивосток, 4–7 апреля 2023 г.).

Том 4 включает в себя следующие секции:

- МАТЕМАТИЧЕСКОЕ МОДЕЛИРОВАНИЕ.
- ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ.
- ИНФОРМАТИЗАЦИЯ НА ПРЕДПРИЯТИЯХ.
- ЭЛЕКТРОННЫЕ ТЕХНОЛОГИИ.
- ОКНО В ЦИФРОВОЙ МИРЪ.
- КАЧЕСТВО УСЛУГ И ТЕХНОЛОГИЙ.
- ИННОВАТИКА НА ТРАНСПОРТЕ.
- АКТУАЛЬНЫЕ ВОПРОСЫ БЕЗОПАСНОСТИ
- ЭКОЛОГИЯ И ОХРАНА ОКРУЖАЮЩЕЙ СРЕДЫ.
- НАУЧНЫЙ СТАРТ.
- СЕКЦИЯ АСПИРАНТОВ.

УДК 378.4
ББК 74.584(255)я431

---

Электронное учебное издание

Минимальные системные требования:

Компьютер: Pentium 3 и выше, 500 МГц; 512 Мб на жестком диске; видеокарта SVGA, 1280×1024 High Color (32 bit); привод CD-ROM. Операционная система: Windows XP/7/8.

Программное обеспечение: Internet Explorer 8 и выше или другой браузер; Acrobat Reader, Foxit Reader либо любой другой их аналог.

ISBN 978-5-9736-0711-1



Под общей редакцией д-ра экон. наук Т.В. Терентьевой

Компьютерная верстка М. А. Портновой

690014, г. Владивосток, ул. Гоголя, 41

Тел./факс: (423)240-40-54

Объем 12,0МБ. Усл.-печ. л. 42,73

Подписано к использованию 10 октября 2023 г.

Тираж 300 (I –25) экз.

УДК 004

# МИКРОФРОНТЕНДЫ: МИКРОСЕРВИСЫ В МИРЕ ФРОНТЕНДА

**С.А. Кулеш**
бакалавр
**О.Б. Богданова**
ст. преподаватель

*Владивостокский государственный университет*
*Владивосток. Россия*

*В разработке каждого приложения ключевую роль всегда играет его архитектура, так как от ее выбора зависит дальнейшая судьба проекта. Чем более верный выбор будет сделан изначально, тем меньше проблем будет выявлено потом. В данной статье рассматривается архитектура микрофронтендов для web-сайтов, их преимущества и недостатки, а также об их отличии от микросервисов.*

***Ключевые слова:*** *архитектура, микрофронтенд, монолит, микросервис.*

# MICROFRONTENDS: MICROSERVICES IN THE FRONTEND WORLD

*In the development of each application, its architecture always plays a key role, since the further fate of the project depends on its choice. The more correct choice is made initially, the less problems will be revealed later. This article discusses the architecture of microfrontends for websites, their advantages and disadvantages, as well as their difference from microservices.*

***Keywords:*** *architecture, microfrontend, monolith, microservice.*

Немного об необходимости архитектуры. При разработке крупных приложений всегда особое внимание уделяется его будущей архитектуре, так как если не задумываться об этом вопросе сразу, то в будущем возникают проблемы при его поддержке.

Главной целью конечно же является снижение сложности проекта. Когда заходит речь о построении архитектуры приложения, организации его структуры, то под этим, обычно, понимают декомпозицию программы на функциональные модули, сервисы, слои, подпрограммы, а также организацию из взаимодействия друг с другом и внешними модулями.

Стоит отметить, что чем более независимые модули получатся при разработке, тем более безопасней будет их поддержка после, так как станет возможно заботиться о каждой из частей отдельно, не заботясь о картине в целом.

При таком подходе приложения превращается в некий конструктор, где каждая из частей общается между собой по строго определенным и простым правилам. Это позволяет регулировать ее сложность, а также получить следующие преимущества:

– масштабируемость (Scalability) – возможность расширять систему и увеличивать ее производительность, за счет добавления новых модулей;

– ремонтопригодность (Maintainability) – изменение одного модуля не требует изменения других модулей;

– заменимость модулей (Swappability) – модуль легко заменить на другой

– возможность тестирования (Unit Testing) – модуль можно отсоединить от всех остальных и протестировать / починить;

– переиспользование (Reusability) – модуль может быть переиспользован в других программах и другом окружении;

– сопровождаемость (Maintenance) – разбитую на модули программу легче понимать и сопровождать.

Проблема модулей. При декомпозиции модулей, связями обрастают как модули между собой, так и подмодули внутри каждого из них. Соответственно, чем больше модулей – тем больше связей в приложении. И если не задумываться о них сразу при построении архитектуры, то они становятся хаотичными и неявными, что сильно усложняет поддержку проекта и добавление нового функционала в него, а также перечеркивает преимущества, описанные выше.

Из чего можно сделать вывод, что для хорошей архитектуры модули должны обладать:

– сильной связанностью – внутренние компоненты модуля направлены на решение одной общей конкретной и четкой задачи;

– слабой зацепленностью – это когда модули между собой наименее зависимы и связи между ними прозрачные.

Микросервисная архитектура в сравнении с монолитной. Термин «Microservice Architecture» получил распространение в последние несколько лет как описание способа дизайна приложений в виде набора независимо развертываемых сервисов, построенных вокруг бизнес-потребностей, насколько это возможно небольших, слабо связанных и легко изменяемых модулей. Сами по себе эти сервисы могут быть написаны на разных языках и использовать разные технологии. [1]

Обычно для входа в эти сервисы используется некая единая точка входа, которая играет роль шлюза и зачастую представляет из себя API, через которую будут вызываться необходимые микросервисы. Применение такого подхода обеспечивает централизованное управление микросервисами и обеспечивает независимость клиентов от протоколов: REST, GRPC и т.д.

Для понимания преимуществ сервисов, лучше всего сравнить его с монолитом: приложением, построенном как единое целое. То есть в одном проекте будут находиться как front-приложение, back-сервер, а также база данных.

Монолитные приложения успешны в основном лишь на ранних этапах проекта, но чем дальше, тем больше вреда они несут.

В процессе жизни приложения обычно становится крайне трудно сохранять качественную модульную структуру, из-за чего появляется влияние частей друг на друга, что влечет за собой ошибки в тех местах, которые даже не изменялись. Следовательно, масштабировать приходится все приложение целиком, даже если это требуется только для одного модуля этого приложения из-за сильной связности модулей.

Также сказывается и на продолжительности тестирования, т.к. приходится охватывать сразу все приложение по той же причине.

Крупным минусом является также то, что зачастую для такого монолита приходится использовать одну основную выбранную технологию как ядро приложения, что ухудшает гибкость приложения и приводит к так называемому легаси-коду.

Все усугубляется если разработкой занимается несколько команд, т.к. чем больше людей в 1 проекте – тем больше конфликтов при разработке возникает, а также каждая команда бу- дет связана в выборе технологий, т.к. от ее выбора зависят и другие, что только замедляет темп разработки.

Эти неудобства привели к архитектурному стилю микросервисов.

Помимо возможности независимого развертывания и легкого масштабирования сервисы получают четкие физические границы. Благодаря этому, каждый из сервисов может разрабатываться с различными технологиями, которые никак не будут влиять друг на друга.

Помимо физических границ, разделяется также и зоны ответственности, что позволяет более четко структурировать код и всегда знать, где конкретно произошла ошибка. А также делает приложения более отказоустойчивым, т.к. при отказе одного из сервисов, остальные продолжат свою работу.

К сожалению, микросервисная архитектура как таковая не применима к среде front-end приложений, так как они по своей сути всегда остаются довольно взаимозависимыми, но никто не мешает взять основные концепции из микросервисов и адаптировать их под свои нужды, что собственно и стало продвигаться среди разработчиков с недавних пор.

Микрофронтенды. По своей природе все frontend-приложения монолитны – кроме приложений, реализующих микрофронтенды, из-за того, что UI не совсем похож на сервисы – это интерфейс между конечным пользователем и продуктом, он должен быть продуманным и системным, т.к. отвечает за взаимодействие с пользователем [2].

Решение данной проблемы довольно простое, так как аналогичные принципы уже давно существуют в среде backend. Необходимо руководствоваться основополагающей идеей – разделение монолита на небольшие UI-фрагменты.

Иначе говоря, микрофронтенды – это техника разработки, которая позволяет разработчикам делать независимые релизы UI-компонентов для разных приложений, сохраняя независимость каждой из них.

Отличным плюсом является также то, что теперь каждому микросервису будет соответствовать свой четко определенный микрофронтенд, что позволит создавать полностью независимые сервисы друг от друга.

Подход в разделении frontend и backend частей приложения также является неоднозначным и существует несколько решений [3].

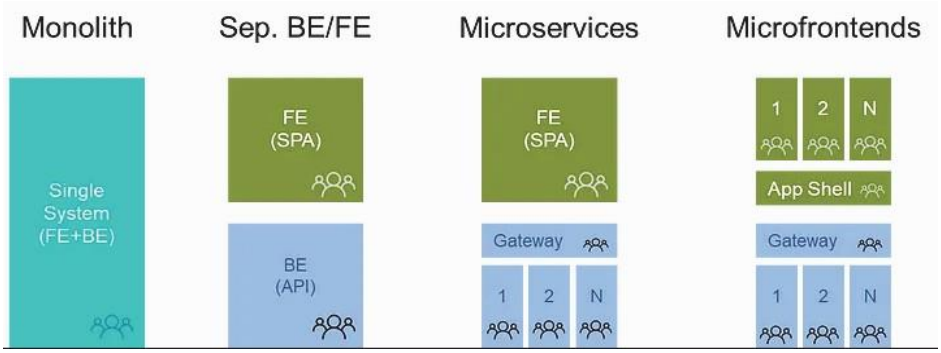


Рис. 1. Примеры решений для разделения forntend и backend

Микрофронтенды на практике. Основная идея микрофронтендов полностью соответствует идеи микросервисов – ядро приложения может состоять из независимых частей. Примером могут служить панель администратора, новостная лента и наконец сам интернет-магазин.

При таком подходе за каждый сервис сможет отвечать независимая команда, которая будет поддерживать как его backend, так и frontend составляющие, что сократит время на разработку новых возможностей и поддержку кода в разы.

Как отмечалось выше, frontend это в первую очередь UI составляющая сайта с некоторой бизнес-логикой, и по своей сути не может быть полностью независимой для удобства конечного пользователя. Поэтому в каждом из сервисов будут использоваться общие пакеты – независимые компоненты или модули сайта, которые переиспользуются в разных сервисах, например, UI-KIT или общую конфигурацию.

Обычно в качестве основной технологии для реализации микрофронтендов отвечает frontend фреймворк. Примером могут служить самые популярные из вариантов: React, Angular, или же Vue, как в моем случае.

Для размещения микрофронтендов в основном используется некий монорепозиторий, в котором находится весь наш каркас приложения. Зачастую он и является точкой входа в приложение, а также отвечает за роутинг в нем.

Альтернативой может служить менеджер пакетов, в котором и будут располагаться наши пакеты и сервисы, а после чего подключаться там, где необходимо.

Первый вариант удобен тем, что нет необходимости следить за версионностью пакетов, нет необходимости плодить целую кучу репозиториев для каждого из них, но взамен мы по- лучаем нагромождение кода. Идеальный подход в данном случае, это грамотное совмещение обоих из них.

Так же в дополнение можно отметить, что мы также используем хранение пакетов не в менеджере пакетов, а напрямую на серверах статики, откуда через API можем получить актуальную версию необходимого нам пакета. Такой подход исчерпывает недостатки двух предыдущих, но появляется новый – лишь прод-версия сервиса, что повышает ответственность разработчиков и заставляет быть крайне аккуратным при изменении кода.

Неплохим вариантов будет использования SPA, о котором уже упоминалось выше. По своей сути это монолит, выделенный из более крупного монолита. Он полностью или частично независим от первоначального, что позволяет разграничить зоны ответственности. В остальном подход крайне похож на монорепозиторий.

Но стоит отметить, что подход микрофронтендов не всегда оправдывает себя. В основном он используется тогда, когда монолитная архитектура исчерпала себя и стала приносить больше вреда чем пользы. То есть если необходимо создать MVP-приложение, то не стоит заморачиваться с микрофронтендами, т.к. это займет больше времени, чем написание приложения как монолита. Необходимо грамотно оценивать, когда какой подход необходим.

Некоторые критерии для выбора микрофронтедов:

– работает большая команда разработчиков (около 10 человек) или есть несколько команд разработки

– наличие большого монолитного проекта

– сборка, тесты, деплой занимают крайне много времени.

– внедрение новых технологий становится крайне затруднительным.

Также можно выделить несколько недостатков данного подхода:

– гораздо более сложная инфраструктура, чем у монолита.

– размытие зон ответственности (должен ли я править UI-KIT? Должен ли я вносить изменения в чужой сервис и т.д.)

---

1. Информация о Микросервисах . – Текст: электронный. – URL: https://habr.com/ru/articles/249183/

2. Статья «Микрофронтенды: о чем это мы». – Текст: электронный. – URL: https://habr.com/ru/companies/raiffeisenbank/articles/459540/

3. Микрофронтенды: микросервисы для фронтенда. – Текст: электронный. – URL: https://habr.com/ru/ companies/first/articles/697994/

Научное издание

# ИНТЕЛЛЕКТУАЛЬНЫЙ ПОТЕНЦИАЛ ВУЗОВ – НА РАЗВИТИЕ ДАЛЬНЕВОСТОЧНОГО РЕГИОНА РОССИИ И СТРАН АТР

Материалы XXV международной научно-практической конференции студентов, аспирантов и молодых ученых 4–7 апреля 2023 г.

Том 4

Под общей редакцией д-ра экон. наук Т.В. Терентьевой

Электронное научное издание

Компьютерная верстка М.А. Портновой

Подписано к использованию: 10.10.2023. Формат 60×84/8
Уч.-изд. л. 38,82. Усл.-печ. л. 42,73.
Тираж 500 экз. (I–50). Заказ № 11-23

Издательство Владивостокского государственного университета
690014, Владивосток, ул. Гоголя, 41
Отпечатано в ресурсном информационно-методическом центре ВВГУ
690014, Владивосток, ул. Гоголя, 41