DOI: 10.58168/MoInSyTe2024_127-132
УДК 004.9

ПРЕИМУЩЕСТВА И НЕДОСТАТКИ ПРИЛОЖЕНИЙ С МИКРОФРОНТЕНД-АРХИТЕКТУРОЙ

О.В. Оксюта[1], Д.С. Арапов[1], А.С. Грошев[1]

[1]ФГБОУ ВО «Воронежский государственный лесотехнический университет имени Г.Ф. Морозова»

Аннотация. В статье приводятся результаты анализа существующих подходов к построению модульной архитектуры на основе микрофронтенда. Рассмотрены современные фреймворки и библиотеки, позволяющие реализовать данную архитектуру. Проанализированы преимущества и недостатки подобного подхода.

Ключевые слова: микрофронтенд, архитектура, фронтенд, клиент, приложение, микроприложение.

ADVANTAGES OF APPLICATIONS WITH MICRO-FRONTEND ARCHITECTURE

O.V. Oksyuta[1], D.S. Arapov[1], A.S. Groshev[1]

[1]Voronezh State University of Forestry and Technologies named after G.F. Morozov

Abstract. The article presents the results of analyzing the existing approaches to building a modular architecture based on microfrontend. The paper considers modern frameworks and bibliotecs that allow to realize this architecture. It analyzes the advantages and disadvantages of this approach.

Keywords: microfrontend, architecture, frontend, client, application, microapplication.

Введение

Микрофронтенды – это архитектурный подход к разработке клиентской части приложения, который становится все более популярным. В командах, работающих над большими проектами, активно внедряется этот подход, крупные компоненты конвертируются в самостоятельные микро-приложения. Подобная архитектура обладает неоспоримыми достоинствами, благодаря которым с ней точно стоит познакомиться поближе.

За последние года подход с разбиением программного обеспечения на мелкие и удобные для управления части стал очень распространенным. Идея такого подхода заключается в том, чтобы иметь множество сервисов, которые можно разрабатывать, тестировать и развертывать независимо друг от друга.

Это то, что представляет собой микросервисная архитектура, применяемая на стороне бэкенда. Применив этот же подход на стороне клиента, мы получим микрофронтенд-архитектуру. В одной фразе ее можно описать так: архитектурный стиль, в котором независимо поставляемые фронтенд-приложения объединяются в единое целое.

Среди известных компаний, которые уже активно применяют этот подход, можно назвать Microsoft, Spotify, Leroy Merlin. Но их намного больше. И ожидается, что в ближайшем будущем их количество будет только расти.

Недостатки монолитной архитектуры

Казалось бы, зачем разделять приложение, и усложнять общую архитектуру? Но с ростом кодовой базы фронтенд-приложений становится крайне тяжело продолжать поддерживать и улучшать приложение.

Фронтенд-разработчики столкнулись с той же проблемой, которую решают бекенд-разработчики, разрабатывая серверную часть приложения. Одно из решений, к которому они пришли, получило название “микросервисная архитектура”. Этот подход прочно закрепился среди разработчиков серверной части веб-приложений.

Микрофронтенды – это не что иное, как попытка применить микросервисную архитектуру на клиентской стороне, и попытка весьма успешная. Ведь монолитная архитектура имеет существенные недостатки:

- Невозможность полноценно разделить ответственность за отдельные части приложения;
- Необходимость собирать и развертывать все приложение целиком, даже при небольших изменениях в отдельном компоненте;
- Сильная зависимость от используемой технологии, например, фреймворка Angular.
- Как следствие, сложности при переходе на другую технологию.

Устоявшиеся подходы в микрофронтендах

Чтобы разработать микрофронтенд-приложение, необходимо использование дополнительных инструментов.

Начать следует с выбора оркестратора для микроприложений. Популярным вариантом является single-spa. Количество скачиваний этого фреймворка выросло за период 2021 г. – 2024 г. в 5 раз: с 39 тыс. до 195 тыс. скачиваний в неделю. [1]

Single-spa – это мета-фреймворк для создания приложений, поддерживающий загрузку кода приложений “по требованию”, поддержку любых фреймворков, а также множество примеров и вспомогательных библиотек, позволяющих быстро подключить микроприложение, написанное на React, Angular, Svelte, Vue.JS и т. д. [2]

Альтернативным вариантном является техника Module Federation. Это тип JavaScript-архитектуры, поддерживаемый, начиная с Webpack v5. Он тоже позволяет построить микрофронтенд приложение, при этом появляется возможность подключать зависимости между отдельными приложениями. Кроме того, их можно использовать совместно в single-spa. Однако мы рассмотрим использование именно single-spa, поскольку Module Federation относительно новая технология и подходы в ней активно меняются.

Следующим этапом является создание начальной точки приложения. Это будет один HTML-файл, подключающий код single-spa и определяющий URL, по которым будут загружаться микроприложения. Для упрощения этого процесса обычно используют Import-Maps. Это не что иное, как словарь соответствий названий пакетов и ссылок, по которым их нужно загрузить. Для внедрения такого подхода часто применяют SystemJS, который является загрузчиком модулей, обеспечивающий поддержку старых браузеров вплоть до IE11, но при этом разрешающий использование возможностей современного EcmaScript, а также Import-Maps [3].

Остается написать микрофронтенд-приложения и подключить их на страницу. Рекомендуется использовать single-spa parsels. Это независимые от используемого фреймворка компоненты, которые могут быть смонтированы в DOM в самом микроприложении. [4]

Преимущества микрофронтенд-архитектуры

В наше время фронтенд – это не просто верстка и клики на кнопки. Фронтенд-программы имеют тенденцию быстро расти и усложняться, а при использовании монолитной архитектуры все становится гораздо труднее поддерживать.

Микрофронтенды могут помочь нам с этой проблемой. Они дают возможность достичь менее сложной и громоздкой архитектуры в отдельных частях большого проекта.

Преимущества становятся явно видны при наличии нескольких команд. Каждое независимое приложение может быть реализовано разными командами и даже с использованием разных технологий. Это обеспечивает масштабируемость, гибкость и адаптивность, аналогично микросервисам на бэкенде.

Более того, такой подход позволяет смешивать на одной веб-странице компоненты, разработанные с помощью различных фреймворков. А еще открывается поле для экспериментов с новыми технологиями. Микрофронтенды уменьшают строгость привязки к конкретной технологии.

Команда всегда может принять решение о выборе нового технологического стека без необходимости переводить то, что было разработано ранее. Кроме того, каждый фрагмент, из которого состоит архитектура микрофронтенда, несомненно, меньше, чем монолит фронтенда, и перевод его на новую технологию займет меньше времени. Поэтому появляется возможность пробовать что-то новое, с целью подобрать лучшую технологию для решения задачи.

Еще одно преимущество – это упрощение обслуживания. Создание небольших микроприложений быстрее и проще по сравнению с большим монолитным программным обеспечением. Проще вносить изменения, исследовать кодовую базу. Кроме того, процесс развертывания тоже улучшается. Фактически, когда команда заканчивает работу над функцией, она может быстро поднять новую версию микро-приложения, без необходимости затрагивать другие. Так как приложение по размеру меньше, то и собирается оно быстрее.

Недостатки микрофронтенд-архитектуры

Конечно, у каждого подхода найдутся свои минусы. В случае микрофронтенд-архитектуры это, как минимум, необходимость настройки общей архитектуры, а также обучения разработчиков работе с ней. На это может потребоваться достаточно много времени с учетом нестандартности подхода.

Кроме того, микрофронтенд приложения требуют тщательной настройки связи между различными микроприложениями, что может являться непростой задачей. Необходимо тщательно спланировать обмен данными и событиями между компонентами, что может усложнить архитектуру системы.

Не стоит забывать и о том, что введение дополнительных слоёв абстракции и загрузка нескольких фреймворков или даже разных версий одного фреймворка

могут повлиять на производительность приложения. Также возможно увеличение времени загрузки из-за того, что браузеру может потребоваться загрузить больше кода.

С учетом всего этого становится очевидно, что такой подход подойдет не каждом проекту. Нет смысла внедрять микрофронтенды в персональный блог, информационный веб-сайт или простой интернет-магазин. Однако, использоавние микрофронтенд-архитектуры может быть обосновано при работе в крупных распределенных командах над большими проектами.

Выводы

Микрофронтенды становятся следующим шагом в разработке клиентской части приложений. Несмотря на все нюансы, это естественное развитие фронтенда. Когда приложение становится огромным, возникает необходимость разбить его на небольшие части.

Хотя эта архитектура пока еще пока еще менее популярна, чем микросервисная архитектура, используемая при разработке бэкенда, все больше компаний начинают применять ее в своих приложениях. Именно поэтому именно сейчас стоит познакомиться с ней, чтобы иметь возможность наблюдать за развитием технологии будущего.

Но стоит помнить и о том, что данная архитектура имеет свои особенности и недостатки. При использовании подобных технологий в реальных проектах нужно четко понимать, какие проблемы решит ее применение, и не добавит ли дополнительных.

Список литературы

1. NPM Stats. URL: https://npm-stat.com/charts.html?package=single-spa&from=2021-01-01&to=2024-03-27

2. Michael Geers. Micro Frontends in Action // Manning Publications Co., – 2020. – С. 134-140.

3. SystemJS. – URL: https://github.com/systemjs/systemjs#systemjs

4. Parsels Overview. – URL: https://single-spa.js.org/docs/parcels-overview

5. Полуэктов А.В., Макаренко Ф.В., Ягодкин А.С. Использование сторонних библиотек при написании программ для обработки статистических данных // Моделирование систем и процессов. – 2022. – Т. 15, № 2. – С. 33-41.

References

1. NPM Stats. URL: https://npm-stat.com/charts.html?package=single-spa&from=2021-01-01&to=2024-03-27.

2. Michael Geers. Micro Frontends in Action // Manning Publications Co., - 2020. - P. 134-140.

3. SystemJS. URL: https://github.com/systemjs/systemjs#systemjs.

4. Parsels Overview. URL: https://single-spa.js.org/docs/parcels-overview.

5. Poluektov A.V., Makarenko F.V., Yagodkin A.S. The use of third-party libraries when writing programs for processing statistical data // Modeling of systems and processes. - 2022. – Vol. 15, No. 2. – pp. 33-41.