A Brief Review of Micro-frontends

Y.R. Prajwal[1], Jainil Viren Parekh[2] and Dr. Rajashree Shettar[3]
[1,2,3] Department of Computer Science, R.V. College of Engineering, Bengaluru, India
Email: [1]prajwalyr.cs17@rvce.edu.in, [2]jainilvparekhcs17@rvce.edu.in and [3]rajashreeshettar@rvce.edu.in

***Abstract*—** In this paper, a brief review of micro-frontends is presented. The paper discusses the idea behind the micro-frontends architecture and depicts that it is an extension of microservices to the frontend. Micro-frontends solve the problems caused by existing monolithic frontends. The paper discusses the various approaches to compose micro-frontends and gives a review on the existing systems that use micro-frontends. The benefits and drawbacks of micro-frontends are also presented. Micro-frontends have a great potential in developing frontend software if the core ideas are implemented effectively.

***Keywords*—** Micro-frontends, Microservices, Frontend, Web Application.

INTRODUCTION

Web applications are generally split into two parts: back-end, that provides APIs to perform server-side operations, and front-end, that provides a user-interface. *Micro-Frontends* is a term that was coined by ThoughtWorks Technology Radar in 2016 [1]. The concept of micro frontends is an extension of the micro services used in backend. Nowadays single page applications (SPA) that runs on top of a micro service architecture are very popular along with server-side rendering applications (SSR) and web pages formed by combining static HTML pages.

Although these frameworks are acceptable options to build feature-rich powerful web applications, they tend to be monolithic frontends thus increasing the size of client-side application.

This limits the scalability of development of the application by multiple teams. Micro-Frontends overcome this problem by decomposing the front-end into different features owned by independent teams such that each team has a separate area of business that the team is proficient in. Several industries like DAZN, Ikea, New Relic, SAP and others have adopted micro-frontends [2].

THE IDEA

In recent years, usage of microservices is very popular and has been researched in the field of information science. A traditional single web application system based on single architecture has

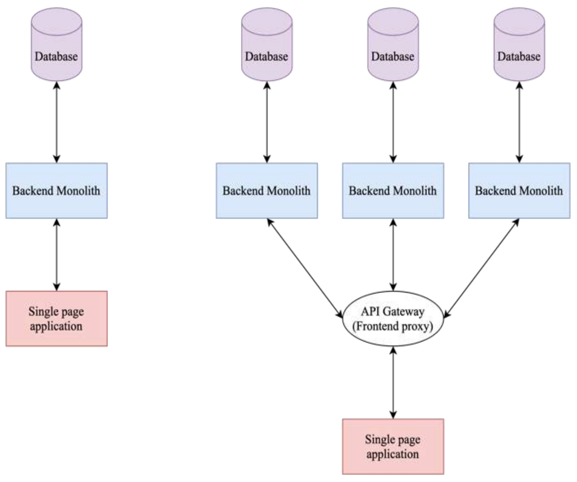

Fig.1 Comparison of traditional single page application with single service architecture(left) Microservices architecture (right)

various modules whose dependencies may not be clear. Furthermore, even a minor modification done to the project requires redeploying the entire project and since the project size will be significantly large this takes a long time.

Microservices architecture relies on a collection of loosely coupled smaller business modules known as services. The services communicate with each other through language independent APIs.

Each service can be developed, tested and deployed independently [3] thus allowing multiple features to be developed in parallel.

There is no one right way of developing software and engineers are constantly looking for new methods of designing software.

Frontend engineers have a few architectural options like SPA, SSR or combining static web pages. However, all these architectures cause the frontends to become monoliths.

This greatly rises the complexity of the project and adding new features might cause unwanted effects on other existing features.

The concept of microservices has many benefits and hence then same concept can be extended to frontends to solve this problem of monolithic frontends.

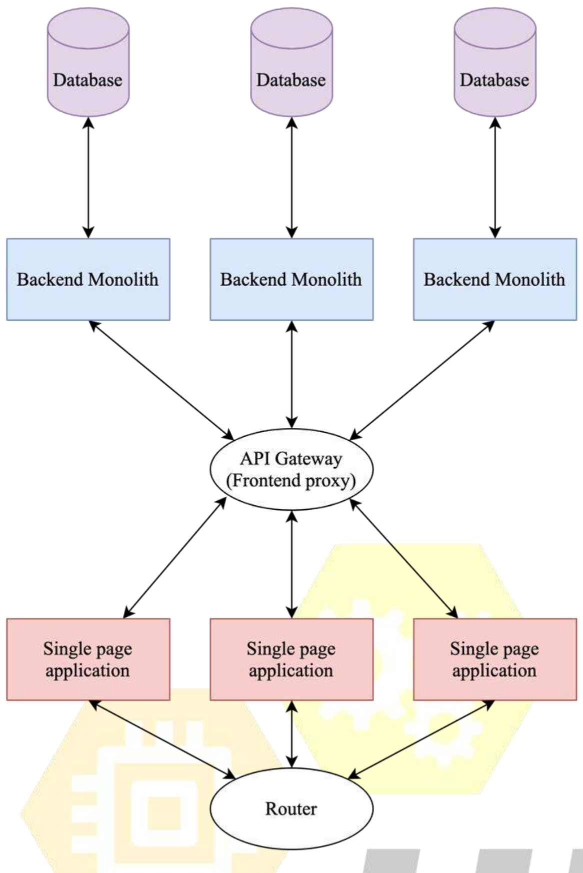

Fig. 2 Concept of microservices extended to form the micro-frontends architecture

The core ideas of micro-frontends are summarized as follows [4]:

- Technology agnostic: Every team must be able to choose and upgrade their technology stack without having to cooperate with other teams. Implementation details can be hidden by custom elements whilst delivering a neutral interface to others
- Code isolation: The teams should build independent apps which are self-contained without relying on global variables or shared states. Runtimes should not be shared across teams
- Naming conventions: In cases where isolation is not possible, agreeing on naming conventions for CSS, events, namespace and cookies helps in avoiding collision and resolving ownership.
- Use native browser features: Browser events are preferred over custom APIs for communication
- Resilient site: The feature developed should also be useful in the cases where JavaScript fails. Universal rendering and progressive enhancement can be used to improve the performance

COMPOSITION OF MICRO-FRONTENDS

There are two available options to identify micro-frontends: horizontal split and, vertical split [5]. With horizontal split, many smaller frontend applications are loaded onto the same webpage which requires many teams to collaborate their efforts as every team will be focusing on a part of the view.

In case of vertical split, every team is in charge for a business domain and Domain Driven Design (DDD) principle applies.

The composition of micro-frontends can have various approaches:

A. Composition with Ajax

The contents of different micro-frontends can be integrated to a single document by loading them using Ajax. A deeper Ajax integration is good for search engine compatibility, accessibility and performance.

However, CSS collisions are a possibility as the Ajax integration puts all the fragments into a single document. This can be avoided by agreeing on naming conventions or introducing team namespaces. The contents of multiple applications can be routed through a single frontend proxy that serves all the content via a unified domain.

B. Server-side composition

In server-side composition, the backend server will compose the view by fetching all the micro-frontends and assemble the final webpage. This helps in achieving good first-page load speeds which are not possible with client-side composition. This technique is helpful when building micro-frontends that are based on progressive enhancement principles.

A Content Delivery Network (CDN) can be used to serve cacheable pages. Server-side composition used by e-commerce companies like IKEA, Zolando and Amazon.

C. Client-side composition

Client-side rendering frameworks like React, AngularJS and Vue have become popular as they provide dynamic routing and a better user-interface. Micro-frontends are wrapped as web components to load on the browser.

Web components are a collection of various technologies including custom elements, HTML templates, Shadow DOM and HTML imports.

Web components allow the creation of technology agnostic frontends which can be loaded by the browser.

Shadow DOM reduces the risk of CSS collisions and also protects against global styles leaking in.

EXISTING SYSTEMS

Micro-frontends are becoming popular and the applications of micro-frontends in existing systems is mentioned below:

1. Micro-frontends along with microservices concept is applied to design a content management system. Mooa, micro-frontend framework for Angular is used. It supports having two or more Angular applications out of which one is used as a main project that loads all other applications [6].
2. Microservice (MS) architecture combined with micro-frontends is used to implement a prototype of web-based configurator for robot-based automation tools. The prototype clarifies that new functionalities like collaborative multiuser configuration are enabled. It also shows that it contributes to better development and simpler deployment using the principle of divide and conquer [7].
3. Micro-frontends also find application in educational management systems. Micro-frontends solutions combined with service-oriented architecture is applied to new generation of graduate information platform of East China Normal University. It is verified that the micro-frontends architecture can adjust to the future requirements of educational management information systems [8].
4. A modular architecture design for industrial Human Computer Interface (HMI) is based on the micro-frontends to develop web applications. It facilitates engineers to form an HMI from micro-frontends where each micro-frontend is integrated throughout, from user interface to its data acquisition [9].
5. Micro-frontends concept is used for the development of an education hub system. The system gathers online courses from different online course providers to serve as a single entry-point. It provides a search engine for users to locate a course that is most suitable to their requirements [10].
6. A Progressive Web Application (PWA) is designed based on micro-frontends and microservices for aiding the user in seamless attainment of geospatial data related to IoT [11].
7. A platform is designed to offer assistance in the operations involved in production of micro-frontends. It allows any producer to publish a micro-frontend to the outside world and any consumer to utilize them to satisfy his/her own requirements [12].

BENEFITS

A. Team Independence

Micro-frontends aim in isolating teams and are incredible tools for helping the teams work independently. Each team has full autonomy of a vertical slice of the application and can focus on specializing in that domain.

B. Code organization

Effective partitioning of the project into micro-frontends also helps in better code organization of the project. Every micro-frontend focuses on a smaller part of the application.

C. Release Independence

The micro-frontends architecture enables faster feature development by following the share nothing architecture. Parts of the project can be released independently.

D. Reduced surface testing and faster builds

Smaller frontends and releases imply that the surface of regression testing is reduced. This also implies that the build time is reduced. However, in practical applications this may not be achieved effectively [13].

E. Fault isolation

One of the important advantages of micro-frontends over monoliths is that in case of any error, the entire application need not be turned down. It is possible to detect in which module the error has occurred, and an appropriate fix can be made to that specific module.

F. Technology Agnostic

One of the main ideas of micro-frontends architecture is that it doesn’t depend on the underlying frameworks used to develop the micro-frontends. This adds the benefit of incorporating different technology stacks for different micro-frontends based on the existing skillset.

DRAWBACKS

A. Redundancy

The general aim in software development is to minimize code redundancy. However, as micro-frontends encompass multiple independent teams developing their stacks parallelly, it might introduce a lot of redundancy in JavaScript and CSS code. This unnecessarily increases the size of code.

B. Workflows that might cross boundaries

It will be difficult to create and design workflows that cross the boundaries between micro-frontends. The built-in navigation and routing cannot be utilized as the different micro-frontends might use different technology stacks and thus requiring custom navigation.

C. Communication

It is against the principles of micro-frontends to communicate between two micro-frontends. This might be problematic for existing features in the system.

D. Risk of Code divergence

The code might start to diverge to a significant extent while breaking down the project into different micro-frontends and end up with different projects altogether.

E. Performance issues

The first-page loading speed is significantly higher compared to other approaches. This latency might affect user interface. However, if the resources are cached the performance is better than other approaches. The performance analysis and comparison of micro-frontends with monolithic system are discussed in [14].

CONCLUSION

The micro-frontends concept solves the problems caused by frontend monoliths by extending the concept of microservices to frontends. Although there are a few drawbacks, micro-frontends have significant benefits and hence has great potential for frontend development. It is becoming popular and is adopted by many developers. The micro-frontends architecture is good for medium to large projects. Micro-frontends are not a good fit when the number of developers available is very small. The concept of micro-frontends can also be extended to native mobile applications but are proven to be beneficial for web applications. Micro-frontends are a solution for scaling development.

REFERENCES

[1] "Micro frontends," ThoughtWorks, November 2016. [Online]. Available: https://www.thoughtworks.com/radar/techniques/micro-frontends. [Accessed June 2021].

[2] M. Geers, Microfrontends in Action, Manning Publications, 2020.

[3] L. Chen, "Microservices: architecting for continuous delivery and DevOps," in *2018 IEEE International conference on software architecture (ICSA)*, 2018.

[4] G. Michael. [Online]. Available: https://micro-frontends.org/. [Accessed June 2021].

[5] S. Peltonen, L. Mezzalira and D. Taibi, "Motivations, benefits, and issues for adopting Micro-Frontends: A Multivocal Literature Review," *Information and Software Technology*, p. 106571, 2021.

[6] C. Yang, C. Liu and Z. Su, "Research and Application of Micro Frontends," *IOP Conference Series: Materials Science and Engineering*, vol. 490, p. 062082, 2019.

[7] E. Schäffer, A. Mayr, J. Fuchs, M. Sjarov, J. Vorndran and J. Franke, "Microservice-based architecture for engineering tools enabling a collaborative multi-user configuration of robot-based automation solutions," *Procedia CIRP*, vol. 86, pp. 86--91, 2019.

[8] D. Wang, D. Yang, H. Zhou, Y. Wang, D. Hong, Q. Dong and S. Song, "A Novel Application of Educational Management Information System based on Micro Frontends," *Procedia Computer Science*, vol. 176, pp. 1567-1576, 2020.

[9] M. Shakil and A. Zoitl, "Towards a Modular Architecture for Industrial HMIs," in *2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2020.

[10] A. Pavlenko, N. Askarbekuly, S. Megha and M. Mazzara, "Micro-frontends: application of microservices to web front-ends," *Journal of Internet Services and Information Security (JISIS)*, vol. 10, no. 2, pp. 59-66, 2020.

[11] M. Mena, A. Corral, L. Iribarne and J. Criado, "A Progressive Web Application Based on Microservices Combining Geospatial Data and the Internet of Things," *IEEE Access*, vol. 7, pp. 104577-104590, 2019.

[12] P. Y. Tilak, V. Yadav, S. D. Dharmendra and N. Bolloju, "A platform for enhancing application developer productivity using microservices and micro-frontends," in *2020 IEEE-HYDCON*, 2020.

[13] L. Steven, "Problems with Micro-frontends," March 2020. [Online]. Available: https://medium.com/swlh/problems-with-micro-frontends-8a8fc32a7d58. [Accessed June 2021].

[14] M. Kroiß, "From Backend to Frontend-Case study on adopting Micro Frontends from a Single Page ERP Application monolith," 2021.