Hippo Digital

Management Consultant

Alexander Adu-Sarkodie: 18/07/2023

Documentation and Solution Design

Life cycle software implementation with Microfrontends

https://github.com/kukuu/hippy/tree/main

Problem statement

I was super inspired as Head of Development to conduct a Prototype project to find a solution to optimise sections of an existing eCommerce application for a client we had initially built an online eCommerce trading platform and make it lean, more efficient and ensure core parts of the application become re-usable as the Client was expanding in creating further categories of the platform and wanted to leverage existing resources and re-use.
My research into various surveys and reports in the landscape of emerging technologies indicated a growing trend towards adopting Microfrontends as a preferred architectural approach. These reports highlight benefits such as improved development speed, scalability, modular architecture, and team autonomy. They also provide insights into real-world implementations, challenges, and best practices.

What is a Microfrontend?

A Microfrontend is an Architectural approach in web development where a Web application’s user interface is divided into smaller independent, and self-contained modules, each responsible for a specific feature or functionality.

These Microfrontends can be developed, tested, and deployed independently allowing teams to work on different parts of the application simultaneously.

When combined, they create a cohesive and scalable user interface that enhances code reusability and maintainability.

The Solution

1. Evidence
2. Architecture - https://www.figma.com/file/rRULXARl2cYkMxCbSLah1X/Microfrontends?type=whiteboard&node-id=0-1&t=GVPZQp8mP7xGEj8T-0
3. Design choices
4. Technology choices
5. Timescales
6. Deployment
7. Budget

How the solution was delivered

1. Size of squad
2. Testing
3. Deployment decisions
4. Tooling
5. Development methods

I. Challenges/setbacks and how this was overcome

Summary

The main challenge of Microfrontends is creating a fast and responsive client. We must never lose sight of the fact that the frontend lives in an environment with limited memory, CPU, and network, or we risk ending up with a slow UI.

A snappy UI is vital for the success of the product. Our initial research during user workshops and brainstorming revealed that a site that loads in 1 second has a conversion rate three times higher than a site that loads in 5 seconds. Every second the user has to wait, money is thrown out of the window.

Being a complex system, Microfrontends are not meant for a greenfield project. In a traditional way, projects get built, dockerised, tested and then deployed. So long as the servers are up and working you could be safe. With MFE you have a hostpage which will dynamically load different sections from different teams into the Hostpage. Teams could mess up and take down the Hostpage.

Worth knowing that JavaScript is a single-threaded execution model. It is not particularly sandboxed at all so very likely runtime environmental issues will occur that is not expected. Different data formats etc.
While developing, we never lost sight of the fact that the frontend lives in an environment with limited memory, CPU, and network, or we risked ending up with a slow UI. Heavy reliance on JavaScript to render the page negatively affects accessibility.

II. Challenges/setbacks and how this was overcome

Strategies and best practices

- Communication
- Version and deployment
- Cross-Origin Resource Sharing (CORS)
- Shared code and dependencies
- Consistency and styling
- Testing and Quality Assurance
- Performance and Load Times
- Monitoring and Debugging
- Team Collaboration

What was implemented and delivered

Compelling design solution factors

- Independent Development and Deployment
- Technology Agnosticism
- Scalability and Performance
- Modular Architecture and Reusability
- Team Autonomy and Productivity
- User Experience Composition
- Simplified Maintenance and Troubleshooting

What was implemented and delivered

Resources

- Architecture - https://www.figma.com/file/rRULXARl2cYkMxCbSLah1X/Microfrontends?type=whiteboard&node-id=0-1&t=GVPZQp8mP7xGEj8T-0
- Source code (Git Repository) - https://github.com/kukuu/microfrontends/tree/main/single-SPA-module-federation
- Snapshot of application - https://github.com/kukuu/microfrontends/blob/main/microfrontends-ecommerce-app.png

Conclusion

The project was delivered on time as agreed by the milestone definition and roadmap with our stakeholders. Lined-up initiatives and project execution timeframes for review and releases were met as set up and validated by the SLA, SoW and contract delivery. The success of the Prototype continued to serve as a business model to enter into more delivery and aftercare contract services.

Analytics gathered showed that conversion rates, click-through rates, first painting, loading times and Average Order Value (AOV) increased by 30%. Cart Abandonment Rate, Bounce Rates dropped drastically as pages loaded faster from 60% to 40%. We aimed for a Page Load Time of 2 seconds which was achieved. This was crucial in providing a good user experience and reducing Bounce Rates which had led to lower conversion rates.

There was a greater improvement in Mobile responsiveness targeting new devices. Increased ratios of Customer Lifetime Value (CLV) and Repeat Purchase Rates were recorded as a result of increased efficiency.

Microfrontend teams focus on tactical delivery that is more customer-centric. Tweaking interfaces to get better conversion, look and engage better with customers.

Being a complex system, it is worth noting that Microfrontends are not meant for Greenfield projects. Teams could mess up and take down the hostpage that aggregates the individual Microfrontends. JavaScript is a single-threaded execution model. It is not particularly sandboxed at all so very likely environmental issues could occur that are not expected.