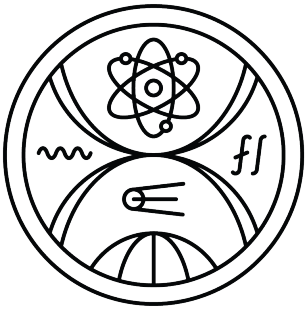
COMENIUS UNIVERSITY IN BRATISLAVA
FACULTY OF MATHEMATICS PHYSICS AND INFORMATICS

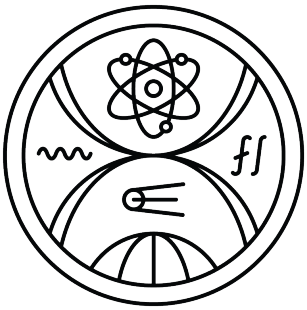


ANALYSIS, DESIGN AND IMPLEMENTATION OF MICRO-FRONTEND ARCHITECTURE

Diploma thesis

2024 Bc. Pavol Repiský

COMENIUS UNIVERSITY IN BRATISLAVA
FACULTY OF MATHEMATICS PHYSICS AND INFORMATICS



ANALYSIS, DESIGN AND IMPLEMENTATION OF MICRO-FRONTEND ARCHITECTURE

Diploma thesis

Study program:	Applied Computer Science
Branch of study:	Computer Science
Department:	Department of Computer Science
Supervisor:	RNDr. Ľubor Šešera, PhD.
Consultant:	Ing. Juraj Marák

Bratislava, 2024 Bc. Pavol Repiský

37862688

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Bc. Pavol Repiský
Študijný program: aplikovaná informatika (Jednoodborové štúdium, magisterský II. st., denná forma)
Študijný odbor: informatika
Typ záverečnej práce: diplomová
Jazyk záverečnej práce: anglický
Sekundárny jazyk: slovenský

Názov: Analysis, Design and Implementation of Micro-frontend Architecture
Analýza, návrh a implementácia mikrofrontendovej architektúry

Anotácia: Mikrofrontendy predstavujú ďalší logický krok vo vývoji architektúry webových aplikácií. Tento prístup si však vyžaduje zvýšenie zložitosti architektúry a vývoja projektu. Problémy ako smerovanie, opätovná použiteľnosť, poskytovanie statických aktív, organizácia úložiska a ďalšie sú stále predmetom značnej diskusie a komunita ešte musí nájsť riešenia, ktoré dokážu efektívne spustiť projekt a riadiť výslednú zložitosť. Aj keď boli navrhnuté a diskutované niektoré prístupy, existuje veľké množstvo poznatkov a potenciálu na objavenie nových prístupov.

Cieľ: Preskúmajte existujúcu literatúru o prístupoch k návrhu a vývoju webových aplikácií pomocou mikro-frontend architektúry.
Porovnajte existujúce prístupy z hľadiska opätovnej použiteľnosti, rozšíriteľnosti, zdieľania zdrojov a správy stavu aplikácií.
Identifikujte prístupy, ktoré sú najvhodnejšie pre vývoj podnikových aplikácií, potom navrhnite a implementujte prototypovú mikrofrontendovú aplikáciu pomocou jedného vybraného prístupu.

Literatúra:
https://www.researchgate.net/publication/351282486_Micro-frontends_application_of_microservices_to_web_front-ends
https://www.angulararchitects.io/blog/micro-apps-with-web-components-using-angular-elements/
https://www.diva-portal.org/smash/record.jsf?pid=diva2%3A1570726&dswid=5530
https://www.diva-portal.org/smash/record.jsf?pid=diva2%3A1778834&dswid=-4588
https://www.scientificbulletin.upb.ro/rev_docs_arhiva/reze1d_965048.pdf

Vedúci: RNDr. Ľubor Šešera, PhD.
Konzultant: Ing. Juraj Marák
Katedra: FMFI.KAI - Katedra aplikovanej informatiky
Vedúci katedry: doc. RNDr. Tatiana Jajcayová, PhD.

Spôsob sprístupnenia elektronickej verzie práce:
bez obmedzenia

Comenius University Bratislava
Faculty of Mathematics, Physics and Informatics

THESIS ASSIGNMENT

Name and Surname:	Bc. Pavol Repiský
Study programme:	Applied Computer Science (Single degree study, master II. deg., full time form)
Field of Study:	Computer Science
Type of Thesis:	Diploma Thesis
Language of Thesis:	English
Secondary language:	Slovak

Title: Analysis, Design and Implementation of Micro-frontend Architecture

Annotation: Micro-frontends represents the next logical step in the development of a web-application architecture. However, this approach necessitates an increase in the complexity of the project architecture and development. Issues such as routing, reusability, static asset serving, repository organization, and more are still the subject of considerable discussion, and the community has yet to find any solutions that can effectively bootstrap a project and manage the resulting complexity. While there have been some approaches proposed and discussed, there is a great deal of knowledge and potential for new approaches to be discovered.

Aim: Review existing literature about approaches to design and development of web applications using micro-frontend architecture.
Compare existing approaches from aspects of reusability, extendibility, resource sharing and application state management.
Identify approaches best suited for enterprise application development, then design and implement a prototypical micro-frontend application using one selected approach.

Literature:
https://www.researchgate.net/publication/351282486_Micro-frontends_application_of_microservices_to_web_front-ends
https://www.angulararchitects.io/blog/micro-apps-with-web-components-using-angular-elements/
https://www.diva-portal.org/smash/record.jsf?pid=diva2%3A1570726&dswid=5530
https://www.diva-portal.org/smash/record.jsf?pid=diva2%3A1778834&dswid=-4588
https://www.scientificbulletin.upb.ro/rev_docs_arhiva/reze1d_965048.pdf

Supervisor:	RNDr. Ľubor Šešera, PhD.
Consultant:	Ing. Juraj Marák
Department:	FMFI.KAI - Department of Applied Informatics
Head of department:	doc. RNDr. Tatiana Jajcayová, PhD.

Assigned: 05.10.2023

Approved: 05.10.2023 prof. RNDr. Roman Ďurikovič, PhD.
Guarantor of Study Programme

Acknowledgement

Tu môžete poďakovať školiteľovi, prípadne ďalším osobám, ktoré vám s prácou nejako pomohli, poradili, poskytli dáta a podobne.

Abstrakt

Slovenský abstrakt v rozsahu 100-500 slov, jeden odstavec. Abstrakt stručne sumarizuje výsledky práce. Mal by byť pochopiteľný pre bežného informatika. Nemal by teda využívať skratky, termíny alebo označenie zavedené v práci, okrem tých, ktoré sú všeobecne známe.

Kľúčové slová: jedno, druhé, tretie (prípadne štvrté, piate)

Abstract

Abstract in the English language (translation of the abstract in the Slovak language).

Keywords:

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background and Motivation

Provide an overview of the background that led to the initiation of the research, addressing the historical context and the driving forces that underscore the significance of the study

1.2 Objectives and Scope

Outline the specific objectives that the study aims to achieve, the scope of the investigation.

1.3 Scope and Limitations

Acknowledging the constraints and boundaries of the study, transparently discuss the limitations and delimitations inherent in the research

Chapter 2

Literature Review

2.1 A Brief History of Web App Architecture

Mention following technologies:

- Static Web Pages
- CGI-Bin
- Client-Server Architecture
- Model-View-Controller
- Service-Oriented Architecture
- Single-Page Applications

2.2 Modern Architecture Patterns

2.2.1 Progressive Web app

2.2.2 Serverless Architecture

2.2.3 Microservices

2.2.4 Micro-Frontends

Chapter 3

Theoretical Framework

The chapter will encompass the essential theoretical information necessary for comprehending the thesis.

Chapter 4

Analysis

Provide a description outlining the purpose and content of this chapter, detailing the topics being discussed herein.

4.1 Micro-Frontends in a Nutshell

- Define microfrontends and highlight its significance in addressing the limitations of traditional monolithic frontend development.
- Discuss the essential features and characteristics of microfrontends, such as modularity, independent deployment, and technology agnosticism...

4.2 Composition Approaches

When considering the architecture of a micro-frontends application, there are several options to choose from. Some are as straightforward as employing a link, while others could fill a whole book. However, there is not a one-size-fits-all solution, it is all about finding the optimal balance for each project. This section presents the five primary approaches, outlining their advantages, disadvantages, and use cases, ranging from the most straightforward to more complex ones.

4.2.1 Composition via links

This architecture composes different pages and features via hyperlinks. Each feature consists of a standalone page interconnected through hyperlinks. URLs serve as the contractual interface between teams, with each team specifying their URL patterns, typically in JSON format, and updating links as necessary. A common scenario involves a standalone cart checkout application being linked from various applications. [4][6]
[Code sample here]

Advantages

The composition via links approach to microfrontends ensures their complete independence from one another, eliminating direct communication between them. Instead, they interact solely through hyperlinks. This design choice provides a significant advantage: any errors occurring within one microfrontend are isolated, preventing them from affecting others in the system. Additionally, this approach simplifies project setup, requiring no additional tools or mechanisms beyond sharing URL patterns. [4][6]

Disadvantages

One major drawback of this approach is the lack of an option to seamlessly combine different microfrontends into a unified view. Users are required to navigate between them by clicking on respective hyperlinks, which can lead to a fragmented user experience. Additionally, common elements such as headers must be individually implemented and maintained in each microfrontend, resulting in duplication of effort and potential inconsistencies across the application. [4][6]

4.2.2 Composition via iframes

Iframes are a well-established technique in web development. Essentially, an iframe is an HTML element that represents a nested browsing context, allowing the embedding of another HTML page into the current one. Compared to composition via links, the transition to iframes requires minimal changes—all that's needed is to include an iframe tag and specify the URL. Thus, teams only need to share URL patterns with each other. [2]
[Code sample here]

Advantages

Iframes offer the same loose coupling and robustness as link composition. They are straightforward to set up and provide strong isolation against scripts and styles leaking out. Additionally, they are universally supported by all browsers by default. [3][4][5]

Disadvantages

However, the main advantage of iframes also presents a significant disadvantage. It is impossible to share common dependencies across different iframes, leading to larger file sizes and longer download times. Communication between iframes is restricted to the iframe API, which is cumbersome and inflexible. These limitations make tasks such as routing, managing history, and implementing deep-linking quite challenging.

Another drawback is that the outer document must precisely know the height of the iframe to prevent scrollbars and whitespace, which can be particularly tricky in responsive designs. While there are JavaScript libraries available to address this, iframes remain performance-heavy. Using numerous iframes on the same page can significantly degrade page speed. Furthermore, iframes are detrimental to accessibility and search engine optimization (SEO), as each iframe is considered a separate page by search engines. [4][3][5]

Suitability

Despite the numerous disadvantages associated with iframes, they can be a suitable choice in specific scenarios. For instance, consider the Spotify desktop application, which incorporates iframes for certain parts of its functionality. In this context, concerns about responsiveness, SEO, and accessibility are not much of an issue.

In conclusion, iframes are not ideal for websites where performance, SEO, or accessibility are crucial factors. However, they can be a suitable option for internal tools due to their simple setup. [4]

4.2.3 Composition via Ajax

Asynchronous JavaScript and XML (AJAX) is a technique that enables fetching content from a server through asynchronous HTTP requests. It leverages the retrieved content to update specific parts of a webpage without requiring a full page reload (AjaxDocs).[1]

When considering microfrontends, the transition from traditional approaches isn't overly complex. Each microfrontend must be exposed as a fragment endpoint and subsequently fetched via an Ajax call, then dynamically inserted into an existing element within the document.

However, this approach introduces unique challenges. Notably, CSS conflicts may arise, particularly if two microfrontends utilize the same class names, leading to undesired overrides. To mitigate such conflicts, it's imperative to namespace all CSS selectors. Various tools, such as SASS, CSS Modules, or PostCSS, facilitate this process automatically.

Additionally, JavaScript isolation is crucial. A commonly employed technique involves encapsulating scripts within Immediately Invoked Function Expressions (IIFE), thereby confining the code's scope to the anonymous function. In instances where

global variables are necessary, they can be declared explicitly or managed through namespacing. [4]

Advantages

With all microfrontends integrated into the same Document Object Model (DOM), they are no longer treated as separate pages, as was the case with iframes. Consequently, concerns such as setting precise heights become obsolete. Furthermore, SEO (Search Engine Optimization) and accessibility are no longer issues since they operate within the context of the final assembled page.

This approach also facilitates the implementation of fallback mechanisms in case JavaScript fails, such as providing a link to a standalone page. Moreover, it supports flexible error handling strategies in the event of fetch failures. [4]

Disadvantages

Asynchronous loading, while powerful, can introduce delays in content rendering, resulting in parts of the page appearing later, thereby potentially degrading the user experience. Additionally, a notable concern within this architecture is the lack of isolation between microfrontends, which is a crucial aspect of microfrontends.

Interactions that trigger AJAX calls may suffer from latency issues, particularly in scenarios with poor network connectivity. Furthermore, this approach do not inherently provide lifecycle methods, unless implemeted manually. [4]

Suitability

This approach is well-established, robust, and easy to implement. It's particularly well-suited for websites where the markup is primarily generated on the server side. However, for pages that demand extensive interactivity or rely heavily on managing local state, client-side approaches may prove to be more suitable alternatives.

4.2.4 Server-side composition

Server-side composition is typically orchestrated by a service positioned between the browser and application server. This service is responsible for assembling the view by aggregating various microfrontends and constructing the final page before delivering it to the browser. There are two primary approaches to accomplish this:

Server-side Includes (SSI): In this approach, a server-side view template is created, incorporating SSI include directives. These directives specify URLs from which content should be fetched and included in the final page. The web server replaces these directives with the actual content from the referenced URLs before sending the page to the client.

Edge-Side Includes (ESI): ESI is a specification that standardizes the process of assembling markup. Similar to SSI, ESI involves including directives within the server-side view template. However, ESI typically requires the involvement of content delivery network providers like Akamai or proxy servers such as Varnish. The web server replaces ESI directives with content from the referenced URLs before transmitting the page to the client.

Other Approaches: Various libraries and frameworks exist, which simplifies the server-side composition. Two notable examples include:

- Tailor: A Node.js library developed by Zalando
- Podium: Built upon Tailor by Finn.no, Podium extends its capabilities and provides additional features. [4]

Advantages

One of the most significant advantages is the excellent first load performance, as the page is pre-assembled on the server, thereby minimizing stress on the user's device. This approach strongly supports progressive enhancement, allowing for additional interactivity to be seamlessly incorporated via client-side JavaScript. Moreover, this technique has been in existence for a considerable period, benefiting from extensive testing, documentation, and refinement. Consequently, it is recognized for its reliability, ease of maintenance, and positive impact on search engine ranking. [4]

Disadvantages

For larger server-rendered pages, browsers may spend considerable time downloading markup instead of prioritizing necessary assets. Moreover, technical isolation is lacking, necessitating the use of prefixes and namespacing to prevent conflicts. Pure server-side solution may notbe optimal for highly interactive applications, requiring combination with other techniques. [4]

Suitability

This approach is ideal for pages prioritizing performance and search engine ranking, as it excels in reliability and functionality even in the absence of JavaScript. However, it may not be optimal for pages requiring instantaneous responses to user input, as it may not offer the level of interactivity needed in such scenarios. [4]

4.2.5 Composition via Web Components

...

4.3 Strengths

List all the benefits of microfrontends.

4.4 Drawbacks

List all the negatives of microfrontends.

4.5 Success stories

Mention some notable companies such as Zalando, Upwork, and Dazn that have adopted microfrontends, and discuss their experiences with this approach.

Chapter 5

Design

5.1 Introduction

Provide a brief introduction to the app, its purpose, target audience, and any relevant background information.

5.2 System Architecture

Provide a comprehensive overview of the overall architecture of the microfrontends system.

5.2.1 Functional Requirements

5.2.2 Nonfunctional Requirements

5.3 Tech Stack

- List all tools, frameworks and libraries which will be utilized.
- Explain what they do and reason for their selection.

5.4 Components

- Specify the micro-frontends the app will be divided into.
- Describe all the components the app will consist of.
- Develop a comprehensive component diagram.
- Elaborate on the implementation of communication protocols.

5.5 Graphical User Interface

- Sketch wireframes depicting the GUI of the app.
- Sketch wireframes depicting the GUI of the app.

Chapter 6

Implementation

Provide a description outlining the purpose and content of this chapter, detailing the topics being discussed herein.

6.1 Overview

- Describe the application implementation process.
- Highlight key considerations and the overall approach to implementing micro-frontends.

6.2 Styling & Sharing

- Describe the challenge posed by CSS in a micro-fronted architecture, where styles are global, inherit, and cascade without the support of a module system, namespacing, or encapsulation.
- Highlight the necessity of ensuring that each micro frontend doesn't conflict with others regarding CSS properties.
- Explain the approach taken to address these challenges, emphasizing the need for consistency in the graphical user interface (GUI) across all micro frontends.
- Discuss the implementation of a shared UI component library as a solution to promote consistency and streamline development efforts.
- Describe how was static assets sharing managent.

6.3 Routing

- How was the routing issue resolved

- Which technologies are utilized for both internal and external routing

6.4 Cross-application communication

- Discuss when micro-frontends must communicate with each other.
- Explain the techniques utilized for communication.
- Discuss how was tight coupling avoided.

6.5 Versoning & Infrastructure

- Discuss the type of repository employed (Mono-repo/Multi-repos) and reasons behind its selection.
- Enumerate all automated workflows which were utilized.
- Highlight any additional tools employed
- Describe the deployment process

6.6 Testing

- Detail the types of tests utilized.
- Explain the testing process, including any automation implemented.
- List the testing tools utilized.

Chapter 7

Conclusion

Provide a description outlining the purpose and content of this chapter, detailing the topics being discussed herein.

7.1 Implementation Results

- Presents the findings and outcomes from the implementation and analysis of microfrontend architectures.
- Cover key metrics, performance indicators, and notable observations to support the thesis's conclusions.

7.2 Practical Implications

Discuss the circumstances under which microfrontends are suitable and when they may not be the optimal solution.

7.3 Recommendations for Future Work

Outline potential areas for future research and improvement in microfrontend architectures.

[?]

Bibliography

[1] Ajax. *MDN Web Docs*.

[2] <iframe>: The inline frame element. *MDN Web Docs*.

[3] Swati Megha Andrei Pavlenko, Nursultan Askarbekuly and Manuel Mazzara. Micro-frontends: application of microservices to web front-ends. 05 2020.

[4] Michael Geers. *Micro Frontends in Action*. Manning, 2020.

[5] Cam Jackson. Micro frontends. 06 2019.

[6] okmttdhr. Micro frontends patterns. *dev.to*, 2020.