NATIONAL UNIVERSITY OF SCIENCES AND TECHNOLOGY POLITEHNICA BUCUREŞTI

Doctoral School of Electronics, Telecommunications and Information Technology

Decision No. 94 **from** 05-10-2023

Ph.D. THESIS SUMMARY

Eng. Adrian PETCU

MIGRAREA CĂTRE WEB3.0: IMPACTUL SCALĂRII ȘI DESCENTRALIZĂRII ASUPRA APLICAȚIILOR SOFTWARE

MIGRATING TO WEB3.0: THE IMPACT OF SCALING AND DECENTRALISATION ON SOFTWARE APPLICATIONS

THESIS COMMITTEE

Prof. Dr. Eng. Mihai Alexandru CIUC Politehnica Univ. of Bucharest	President
Prof. Dr. Eng. Dan Alexandru STOICHESCU Politehnica Univ. of Bucharest	PhD Supervisor
Prof. Dr. Eng. Dan Marius DOBREA Technical Univ. of Iaşi	Referee
Prof. Dr. Eng. Cristian GRAVA Univ. of Oradea	Referee
Conf. Dr. Eng. Bogdan Cristian FLOREA Politehnica Univ. of Bucharest	Referee

BUCHAREST 2023

Table of contents

Chapter 1

Introduction

As technology increasingly integrates into our daily lives, it fundamentally transforms our perception, use, and dependence on digital applications and platforms. The way we use software applications has significantly evolved over time by emphasizing accessibility and scalability.

Within this dynamic process, there has been a shift from conventional platforms requiring local installation to web-based interfaces. This transition has evolved in a new era characterized by increased accessibility and widespread availability. The transition process is not limited to platform changes but rather includes a more comprehensive reevaluation of software functionalities and design principles. Moreover, the emergence of micro-front-ends has led to a departure from monolithic projects, giving priority to modular and scalable structures that meet the varied and constantly evolving requirements of modern consumers. This transition has led to the *centralization* of user information and the need for *trust* in the resilience and security of web platforms, as opposed to local data storage.

The aim of the present study is to investigate these changes, with a specific focus on migration, scalability, and *decentralization* of software systems.

1.1 Presentation of the Thesis Domain

The advent of the internet revolutionized the way we access and share information and opened up the world to a universe through which we can interact and influence projects, products, and social connections. Private companies have realized that the mode of distribution and control of software (under a license) can be easily achieved by exposing graphical interfaces in the form of web pages, as opposed to offering installable packages to users. However, this change in direction came with technical challenges and limitations. In the early days of the internet's rise in popularity, it was used by private companies to distribute installable packages to its users, thus solving the problem of software distribution and foregoing the need for hardware support. The next step in the

evolution of software applications led to a strategic reorientation toward web platforms and the replacement of installable packages on users' devices.

Creating web applications that fully mimic the behavior of installable ones proved to be a challenge as web browsers were poorly performant and had many limitations.

Although there are many benefits to migrating applications to the web domain, there can also be disadvantages. However, if the performance of the web application is comparable or even better than that of the locally installed application, we can consider that the benefits far outweigh the disadvantages. A primary research topic is the analysis of the performance of an application available in both versions, both as an installable on the hardware device and as an online version.

Cloud technology has revolutionized the development of online applications, allowing organizations to store and analyze data without managing physical servers, facilitating scalability and adaptability. Nonetheless, the transition to microservices architectures has highlighted a pain point: the monolithic frontend. This limited flexibility and speed of development, but the introduction of the micro-frontend concept has addressed this issue, allowing for the autonomous and scalable development of GUI components without interference.

1.1.1 Web3.0 and Decentralization

Looking ahead, the concept of Web3.0 and decentralization represents a new horizon in web application development. Web3.0 emphasizes decentralization, security, transparency, and user control over their data. The use of technologies such as blockchain and smart contracts will allow for the creation of safer and more transparent web applications. However, realizing this vision requires ongoing collaboration, research, and development [1] to address the technical, ethical, and social challenges that may arise. The potential of Web3.0 to reshape the internet in a way that better serves both the public and individual interest makes this exploration not just valuable but essential for the digital future of humanity.

1.2 Motivation

The evolution of software development has been characterized by a perpetual process of invention and readjustment. The landscape of software development has undergone significant transformations, starting from the era of natively installed applications, moving to web-based applications, and now advancing towards the frontier of decentralized applications utilizing Web3.0 technology. The aforementioned revolution has not only altered the way software is developed but also the way we understand and engage with it.

This thesis aims to provide an overview of the historical, current, and future aspects of software development, examining the transition from native installed applications to

web applications, analyzing the role of micro-frontends in addressing scalability issues, and exploring the potential of Web3.0 and decentralization.

1.3 Scope of the Research

Considering the evolutionary aspect of software applications, we can consider the historical analysis of their evolution over time to reach a radiography of current times and an analysis of future trends through the prism of decentralization.

Therefore, in the following chapters, we aim to fulfill the following objectives:

- Conducting a study on the evolution of applications from native environments to online platforms;
- Identifying potential issues and finding solutions for them;
- Researching and implementing popular solutions for scaling user interface graphics;
- Investigating the implications of decentralization as an evolutionary step in software applications. Practical applications and implementations;
- Drawing conclusions regarding the researched subject;

1.4 Thesis Structure

The thesis begins with an introductory chapter that presents the research field, motivation, and purpose of the current study.

Chapter 2 presents current trends and related works; the topics studied are cutting-edge, and their research in the academic world is rapidly growing. Alternative solutions and simple implementations thereof are presented and compared.

Chapter 3 describes the theoretical foundations of web applications and various methods of packaging them as well as popular software architecture systems. Last but not least, fundamental concepts underlying blockchain systems and decentralized applications in the Web3.0 world are presented. All these theoretical aspects are used as the basis for the research conducted in this thesis.

Chapter 4 introduces the concept of migrating natively installed applications to web applications, along with the challenges and advantages brought by this process. The central element of this section is the comparative performance analysis between two versions of the same application (Web and native installable) using well-established criteria. The chapter continues with the validation of the research methodology and the implementation of the proposed experiment. Finally, the results obtained are analyzed, and based on them, recommendations and conclusions are presented.

Chapter 5 analyzes in detail various popular micro-frontend solutions, assessing the advantages and disadvantages of each, and provides practical guidance for their implementation within modern organizations that desire to improve user experience in the online environment. The practical experiment consists of implementing two popular solutions and comparing their performance relative to the baseline implementation (monolith). The results obtained are centralized and analyzed, and based on them, conclusions, recommendations, and future perspectives are drafted.

Chapter 6 offers an in-depth analysis of the concepts, technologies, and architectures that underpin *Web3.0*, including smart contracts, cryptocurrencies, and peer-to-peer networks. Additionally, case studies and concrete examples are presented, illustrating how these technologies can be used to create safer, more transparent, and resilient applications that give more control to both individuals and communities.

Finally, **Chapter 7** presents the final conclusions, accompanied by a succinct presentation of the contributions presented in the thesis and some possible paths for future research.

Chapter 2

Current Trends and Related Work

2.1 Evolution of Applications

Over time, programming languages have undergone fundamental changes in terms of their capabilities, the ease of developing new programs[2], and collaborative work modes. With the rise of website popularity in the 2000s, dynamic programming languages such as Ruby and Javascript have gained momentum due to their focus on ease of development in the web space.

Since 2010, numerous programming languages have emerged. In the domain of application decentralization, the most notable are Rust and Solidity. Rust, developed by Mozilla in 2010, was designed as a programming language for safe and high-performance systems [3]. Solidity, created by the Ethereum consortium in 2014, serves as a contract-oriented language for writing smart contracts on the Ethereum blockchain. While Rust excels in building efficient and secure systems, Solidity plays a pivotal role in governing the behavior of decentralized applications through self-executing smart contracts.

2.1.1 Programming Languages and Collaborative Work

In addition to the evolution of programming language popularity over time, it must be mentioned that collaborative work has also undergone positive changes. The advent of version control systems has revolutionized the way software development can be distributed among multiple solution participants. Git is used in large development teams, where team members act somewhat independently and are spread over a large geographic area [4].

The choice of preferred programming language has been driven by the need for ease of writing code, increased productivity, and enhanced security. Collaborative work and open-source programs have contributed to the evolution of programming languages [5], transforming the way we write code and paving the way for the development of complex, innovative, and scalable solutions.

2.1.2 Application Distribution

The evolution of software solution distribution has seen a paradigm shift from acquiring the physical device on which it was stored to downloading the solution from the internet and, in modern days, accessing the software solution in the form of a web application.

2.1.3 Scaling of Web Graphic User Interfaces

Creating web applications with a diverse user base is an essential component of contemporary web development. Therefore, web applications must be built from the ground up to be scalable, to ensure they can handle a potential surge in users.

Graphic User Interfaces (GUIs) tend to become very difficult to maintain as numerous teams can concurrently develop new features that may affect the same piece of code. In chapter 5, we explore different types of scaling for user graphic interfaces.

2.2 Blockchain and Decentralization

The emergence of blockchain technology has substantially altered various sectors, from banking to healthcare. An efficient, decentralized method of data management and transaction processing, cryptographically secured by nature, transparent, and immutable, blockchain technology promises a disruptive response to many of the current challenges.

Blockchain, at its core, is a distributed ledger technology (DLT) that ensures records are both immutable and transparent. Unlike traditional centralized databases where control is held by a single entity, in a decentralized system, control is diffused among its participants.

2.2.1 Pre-Blockchain Decentralization

Peer-to-peer (P2P) systems represented a form of decentralization before the concept and widespread use of blockchain technology.

Establishing trust in P2P systems is crucial, and unlike centralized systems where trust is tied to a central entity, P2P requires a dynamic and decentralized security model. A decentralized security model is presented by the authors in [6].

2.2.2 Decentralization in the Blockchain Era

Blockchain aims for the decentralization of trust and consensus, providing immutable and transparent ledgers. This technology not only promises greater security but represents a potential threat to traditional centralized power systems. It introduces the potential for the decentralization of financial exchanges, contracts, and even government, supporting transactions that rely on cryptographic proof, without intermediaries. Leading

institutions, such as the United States and the European Union, are exploring monetary decentralization, with initiatives such as the Digital Euro [7] and Digital Dollar [8], although full adoption requires careful consideration of the risks. Beyond the economic sector, blockchain can bring benefits to the public and private spheres, from digital identities and secure voting to company benefits, although there are security concerns.

2.2.3 Decentralized Economy

Cryptocurrencies have paved the way for a new system of digital transfer and exchange, with the potential to digitize and decentralize economies. The implementation of digital currencies, such as the dollar and euro, based on blockchain technology, can revolutionize the economic landscape, enhancing efficiency, accessibility, and financial transparency. However, in this transition, it is essential to prudently address issues of privacy, security

2.2.4 Web3.0

Web3.0, or the decentralized web, has recently gained traction, emphasizing decentralization and granting users greater control over their data and identities. This vision stands in contrast to the Web2.0 model, which is dominated by a few large tech companies that control services and data. With the aid of blockchain technology, Web3.0 is moving towards a system where trust is decentralized and users have more control. Discussions are ongoing about how Web3.0 and Web2.0 might coexist and collaborate, yet they may also be in direct competition.

Chapter 3

Theoretical Fundamentals

3.1 Applications, Organization, and Distribution

An application is a material (software or hardware) designed to satisfy specific needs or to entertain end-users. Initially, the term "application" in the software world referred to code packages written in various programming languages (C++, Java, etc.) intended to be executed on a device by an operating system (Windows / Linux). Later, with the rise in internet popularity, applications migrated towards web interfaces.

3.1.1 Application Package Content

Applications generally consist of an executable file, libraries, and static resources. Depending on the technology, this can be represented either by a specific executable file (such as .exe, .sh, .dmg) or an HTML file. The process of installing an application involves unzipping a file in a certain location and configuring the necessary entries in the operating system for its proper functioning [9].

3.1.2 Types of Applications

Applications rely on operating systems to provide basic functionality and access to hardware. With the rapid expansion of the internet and its benefits, several types of applications have become available to end-users as the mode of application distribution has evolved from using hardware components for storage and distribution (CD, DVD, Floppy disk) to downloading them from public websites.

There are numerous types of media used for the implementation and distribution of an application: installable (native) applications, web applications, hybrid web applications, and progressive web applications.

3.2 Types of Application Architecture

In recent years, innovations in technologies and methodologies have allowed the development and implementation of software applications in a faster and more efficient way. One of the most significant trends in software development is the shift from applications with a monolithic architecture to those based on microservices.

3.2.1 Monolithic Architecture

In software development, the word "Monolith" refers to an application formed from a single project in which multiple components and services are combined and served under the same application infrastructure by a single platform. Such applications usually serve multiple domains of interest [10].

3.2.2 Microservices-based Architecture

Microservices-based architecture is a variant of Service-Oriented Architecture (SOA). Although SOA has been available since 1998, microservices were officially adopted in 2012 [11]. The central concept behind microservices consists of several small and autonomous services working together to serve the same purpose of a large application; a large and complex application is divided into smaller pieces [12], organized by subdomains, thus being easier to maintain.

3.2.3 Monoliths and Microservices; Comparative Analysis

Microservices are an obvious choice as they facilitate collaborative work and allow for the incremental updating of individual parts of the system. Moreover, if there is a specific demand for a certain domain of the application, only the services responsible for that domain are scaled to meet the consumer's needs [13].

3.3 Decentralization using Blockchain

Since the advent of blockchain technology, numerous practical applications have sparked general interest in the world of software development. Initially based primarily on cryptocurrencies, the evolution of the technology has been spectacular, and the data stored on the blockchain can be used to create complex applications with applicability in a variety of fields. The main advantages of blockchain technology are the traceability and immutability it offers.

3.3.1 Private Blockchains

Unlike public blockchains, private blockchain systems rely on strict and traceable control mechanisms.

3.3.2 Public Blockchains

While public blockchains provide total transparency of data and anyone can participate in the validation of transactions, the development of applications that use public infrastructures is vulnerable to attackers as the code is exposed.

3.3.3 Decentralized Storage (IPFS)

IPFS (InterPlanetary File System) is a distributed peer-to-peer file system that aims to connect all devices under the same file system. The most used data distribution system is still HTTP, but it shows deficiencies with a large volume of data. IPFS seeks to improve this system without jeopardizing the user experience.

3.3.4 Smart Contracts

Smart contracts, fundamental for applications on Ethereum, are digital versions of traditional contracts, transforming agreements into self-executing code on the blockchain. Unlike classic contracts, which require trust between parties for execution, smart contracts execute automatically upon the fulfillment of the terms, eliminating the need for mutual trust.

3.3.5 Consensus Mechanisms

The consensus mechanism is one of the essential components in the world of blockchain technology, ensuring its reliability. A consensus mechanism is essentially a complex procedure used by blockchain networks[14] to bring diverse and possibly untrustworthy parties to a consensus on the veracity of transactions. This not only helps verify the legitimacy of transactions but also acts as protection against double-spending. The most popular consensus mechanisms used at the moment are Proof of Work (PoW) and Proof of Stake (PoS).

3.3.6 Security and Attack Vectors

Security has become a top priority in the ever-changing ecosystem of blockchain technology. Blockchains are resistant to many common cyber threats due to their decentralized and inherently cryptographic nature, but they are not impervious to all known types of attacks.

3.3.7 Empowering Users and Data Governance

In a decentralized architecture, users have control over their personal data, differentiating from the traditional centralized model. In a public blockchain, data transparency and traceability are ensured.

3.3.8 Cost Models

To calculate the cost difference between a centralized and decentralized architecture, various aspects are considered, such as infrastructure costs, the complexity of application development, and the costs generated by information storage.

3.4 Electronic Wallets

Electronic wallets represent a paradigm shift in the field of digital finance as they allow users to store, transact, and manage their financial assets electronically.

3.4.1 Types of Electronic Wallets

The most popular solutions for maintaining and accessing an electronic wallet come in the form of a mobile app. However, there are several types of electronic wallets, ranging from hardware to software support, each bringing advantages and risks.

3.4.2 Security of Electronic Wallets

All electronic wallets are vulnerable because they depend on a phrase that the user remembers or notes down called a "mnemonic". If this phrase falls into the hands of a malicious user, the electronic wallet can be considered compromised, and all the assets stored at that address can be lost.

3.4.3 Interaction with Electronic Wallets

The communication between blockchain wallets and decentralized applications (dApps) stands out as a key advancement in the complex world of blockchain technology, paving the way for more fluid and simpler user interfaces. DApps run on peer-to-peer blockchain networks, as opposed to conventional programs that use centralized servers. A secure interface is necessary for users to interact with these dApps, whether to make a transaction, stake tokens or participate in other blockchain-related activities.

3.5 Web 3.0

Web3, also known as Web 3.0, is the idea of the next version of the World Wide Web, which focuses on the decentralization of data and a token-based economy [15]. Decentralization empowers the exchange of information between peers, eliminating middlemen and third parties that could control the data.

3.5.1 Web 3.0 Architecture

The Web 3.0 architecture revolutionizes the internet through decentralization, allowing users to control their personal data using blockchain technologies and smart contracts, and the connection to classic systems is made without considerable effort.

3.5.2 Web 3.0 Applications

There are numerous applications of Web 3.0 technology, most emphasizing the anonymity of participants and data transparency. Notable examples of technology use include DeFi (Decentralized Finance), NFTs, and DAOs (Decentralized Autonomous Organizations).

Chapter 4

Migrating from Natively Installed Applications to Web

In this chapter, I explore the evolution of applications, their content, and types of applications. Furthermore, I delve into the challenges posed by application migration and conclude with an analysis of the performance of a popular document editing application available both as an installable package and as a web application, measuring the time required to open documents and the RAM consumption.

4.1 User Experience and Technical Challenges

It has been demonstrated that the user experience (UX) during software usage has a significant influence on adoption and the manner in which the solution is used. As companies consider migrating their programs from installable packages to online applications, it is essential to consider their impact on the user experience (UX). There are several technical considerations involved in achieving a software migration, from choosing technologies to system security concerns.

4.1.1 User Experience

From the user experience perspective, the challenges of migrating to online applications, such as the dependency on internet connection and the learning curve for new interfaces, can be addressed by providing training and support resources that facilitate user adaptation.

4.1.2 Technical Requirements

The technical challenges of migrating applications to the web environment, such as browser limitations and interaction with hardware, can be approached with a strategy

that includes secure development, the use of secure cloud infrastructure, and performance optimization, thereby enhancing the user experience.

4.1.3 Advantages and Disadvantages of User-Installed Applications

User-installed applications generally require the user to own the hardware on which the application is installed. However, in the rapidly evolving world of the internet, this is a significant disadvantage. Some applications offer online synchronization of configurations and user content. However, to access this information [16], the user must install the same application on multiple internet-connected devices.

The advantages lie mainly in the capabilities of the application to function without an internet connection and its isolation from potential attackers, while the disadvantages are limited to restricted access to the application and the difficulty of interacting with the application for the first time.

4.2 Key Aspects in the Transition to a Decentralized Infrastructure

The transition from Web2 to Web3 involves a substantial move towards decentralization, and a business analysis underpinning the application in question must be taken into account before making the transition. Applications designed for private use within a company can benefit from a decentralized architecture, as the nature of blockchain transactions is based on immutability, thus, better resource tracking can be implemented.

When moving to a public blockchain and decentralized architecture, governance, security, interoperability, and confidentiality must be taken into consideration.

4.3 Comparative Analysis of the Performance of a Web Application Also Available in an Installable Format

For the purpose of this study, a popular document editing software solution was chosen because it is available both as an installable package and as a web application.

Devices used for comparison:

- **Device 1:** Apple Macbook Pro, 13 inches, CPU: Apple M2, Memory: 32GB
- **Device 2:** Apple Macbook Pro, 16 inches, CPU: Apple M1 Max, Memory: 64GB

Internet connectivity:

- 230 Mbps, Bucharest, Romania

Browser:

- Google Chrome, Version 111.0.5563.64 (arm64)

Performing multiple simulations for opening a file using both laptops and both solutions will provide valuable information about the performance difference between the online version and the user-installed version of the same application.

The time until the first rendering and the RAM consumption are essential factors to consider [17], as they provide insights into the user experience. The browser's cache system (memory saving) was not disabled, and the installed application was completely closed after each file opening.

The study was successfully conducted using two different types of files (small and large) - 390KB and 32MB. Comparing loading times and RAM consumption on both devices provides a good overview of the differences.

Users' reaction time and network request time were measured to see if the online version of the same application loads faster compared to the locally installed version.

The time spent opening files was measured from the beginning of the process (double-clicking on the executable file for the native application and clicking on the link for the web application) until the moment the document became interactive for 20 consecutive runs using the two devices mentioned.

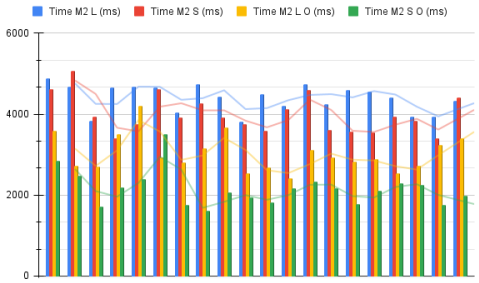
The calculation of the relative improvement RI was made using the formula of Equation 4.1, where N represents the new value and O the original value.

$$RI = \frac{N - O}{O} \tag{4.1}$$

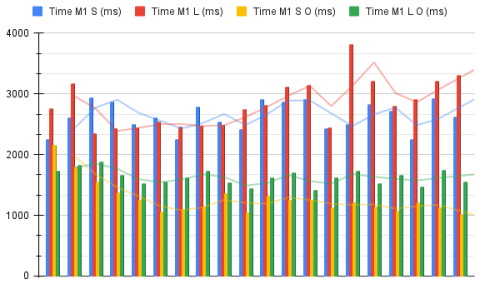
It was observed that using the installed application, the time spent opening a small file averages 3.994s, while opening the same file on the web version has an average of 2.160s on **Device 1**. Performing the same tests on a larger file, the average loading time for the native application was 4.396s compared to the web version, which had an average of 3.032s. This represents an improvement of 45.93%, respectively 31.03%, in the time spent on the online version compared to the installed application, as seen in Figure 4.1a. On average, the time decreased by approximately 38%.

RAM consumption on **Device 1** for the operation of opening the two files using the local application was observed to have an average of 351MB for small files and 367MB for large files, while for the web version, the averages were 227MB and 239MB, respectively, representing an average of 36% lower RAM consumption, as seen in Figure 4.2a.

Performing the same tests on **Device 2**, apart from the overall reduction in time spent compared to **Device 1** (expected, given the different specifications of the devices), it can be observed that opening the small file on the native application and on the web version lasted an average of 2.640s and, respectively, 1.282s. For the larger file, the results were 2.836s and 1.630s. An average decrease of 51.43%, respectively 42.5%, in

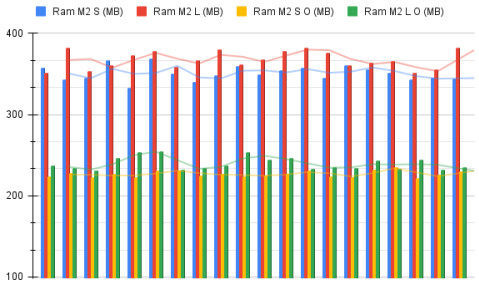


(a) Comparing times on Device 1

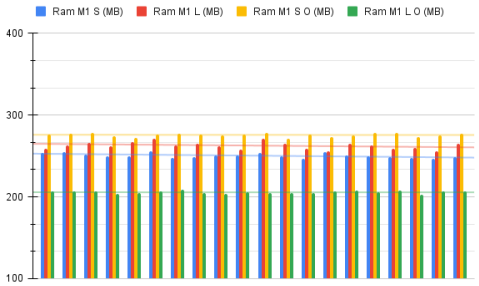


(b) Comparing times on Device 2

Fig. 4.1 Time to first interaction
S: Small file M1: Device 1
L: Large file M2: Device 2
O: Web app



(a) Comparing RAM consumption on Device 1



(b) Comparing RAM consumption on Device 2

Fig. 4.2 RAM Consumption
S: Small file M1: Device 1
L: Large file M2: Device 2
O: Web app

the time spent, was observed. In total, the time decreased by 47%. Figure 4.1b shows the evolution of the tests on device 2.

RAM consumption on **Device 2** for operations using the local application was observed to have an average of 250MB for small files and 262MB for large files, while for the web version, the averages were 275MB and 205MB, respectively, representing an average of 6.25% lower RAM consumption, as seen in Figure 4.2b.

The expected time for performing both operations, using the web application and the locally installed application can vary depending on network load, internet speed, and device load.

Calculating the results, we can assume that using the web version of the same application requires 42% less time until the application becomes interactive and the RAM consumption is 20% lower.

4.4 Conclusions

As software development and the internet evolve, it is necessary to keep pace with current trends. In the early stages of software development, interaction with systems was straightforward and primarily conducted through a command terminal. Recent trends are based on modern user interfaces with a simplified experience, as the interaction with everyday applications moves from desktop-installed applications to web applications.

Chapter 5

Scaling Web Application GUIs using Micro-Frontend

In this chapter, I analyze existing solutions for creating micro-front-end applications and highlight the obstacles and advantages they present.

In the first section, I research current architectural patterns along with their advantages and disadvantages. Then, I delve deeper into the micro-front-end architecture by investigating implementation models and techniques[18]. Finally, I explore the available methods for implementing micro-front-ends, each with advantages and disadvantages based on predefined criteria, and a practical comparison is made between the three chosen implementations.

5.1 Micro-Frontend

Micro-Frontend is a *Microservices* approach to front-end web technologies. The main goal of micro-front-ends is to divide the application into multiple application units based on pieces of functionality or screens representing a certain domain instead of creating a real *monolithic* front-end application[19].

5.1.1 Types of Composition

To build an application based on Micro-Frontends, there are several different options. For example, in a Micro-Frontend-based architecture, certain architectural decisions need to be made in advance[20], as these decisions will influence future implementation choices. Applications can be divided both by including multiple micro-frontends on the same page and by isolating micro-frontends to one per page.

5.1.2 Challenges of Using a Micro-Frontend Architecture

From the analysis conducted, only a limited number of studies have been created to find the best approach for adopting a micro-front-end architecture. Thus, there is no standard in force. The benefits of adopting a micro-front-end architecture include streamlined event coordination through a robust framework, backward compatibility ensured at updates, standardized communication between components, a Publisher/Subscriber model for centralized management of changes, optimal package size control through selective inclusion of dependencies, and style consistency across projects with the help of a unified styling library.

5.1.3 Benefits

Emphasizing the emerging attributes and benefits offered, besides the specific technical approach, the micro-front-end architecture brings many advantages to development teams.

5.1.4 Micro-Frontend Solutions

Several micro-front-end solutions have emerged in recent years, each with its benefits and drawbacks. There are multiple ready-to-use solutions available on the web, such as **SingleSPA** or **NX**, but also many solutions can be built from scratch.

5.2 Performance Analysis of Various Micro-Frontend Solutions

A detailed research methodology was employed to compare the benefits of different micro-frontend solutions. The methodology included an extensive analysis of previous research and publications related to the topic, as well as an examination of third-party vendors' documentation for existing micro-frontend solutions.

Comparing the advantages and disadvantages of each solution with respect to the identified criteria, I determined which solution would be most suitable for migrating an application to a micro-frontend architecture.

The implementation of a simple web application took place as part of this study to compare two micro-frontend solutions with the monolithic approach. For comparison, similar variations of the application were implemented using iframes and Module Federation. The turquoise and purple blocks represent micro-frontends for the iframe and Module federation implementations, respectively. The navigation bar and the top-right block belong to the central application used solely for loading the modules. Using the practical implementation, I obtained information and conclusions about the performance of each micro-frontend solution in meeting the chosen criteria.

By analyzing the implemented solutions with the inspection tools provided by browsers, I was able to determine and analyze both the time taken for the first paint of content and the time taken for resource loading for each type of application.

From the evaluation criteria listed in the research methodology, several findings can be deduced regarding the advantages and disadvantages of the various solutions analyzed.

We can calculate a relative improvement RI using Equation 5.1, where N represents the new value and O represents the initially observed value.

$$RI = \frac{N - O}{O} \tag{5.1}$$

Analyzing the information resulting from the implementation of a simple web application, as seen in Table 5.1 and Figure 5.1, we can observe that the solution using Module Federation is more performant than the solution implemented with iframes. Figure 5.1 is a radar chart where the values obtained from the analysis and presented in Table 5.1 are compared among the three implementations.

Module Federation shows a 55% reduction in the time required for the **first contentful paint** compared to iframes and a 29% increase in time compared to a monolithic application. However, considering that the first contentful paint depends on the size of the package, for larger applications, the time of the first contentful paint will increase compared to Module Federation, which relies on asynchronous loading, and modules are loaded progressively, depending on user needs.

The results show that the **bundle size** is reduced in Module Federation compared to the iframe solution. Duplication of resources in a solution using iframes or routing is often inevitable, thus increasing the bundle size. Analyzing the results obtained and listed in Table 5.1, we can observe a 60% decrease in the bundle size for Module Federation compared to the iframe implementation. However, Module Federation implementation shows a 22% increase in bundle size compared to the monolithic approach.

Regarding the **number of requests** and the total size of the resources loaded, Module Federation surpasses the iframe implementation but is less performant than the monolithic implementation. The iframe implementation also shows a significant increase in total size compared to the monolithic approach, mainly because the resources are not shared between different parts of the application. Module Federation shows a 23% decrease in the number of requests compared to the iframe implementation and a 100% increase in the number of requests compared to the monolithic solution.

Load time is calculated based on the initial loading of resources, and since Module Federation loads secondary modules asynchronously, there is a significant improvement in speed compared to both the monolith and iframes. The results show a 30% decrease when comparing the load time of Module Federation to iframes and a 42% decrease when compared to the monolithic approach.

Table 5.1 Comparison of resources utilized by each solution

Type/Criteria	**Monolith**	**Iframe**	**Module Federation**
First paint	418ms	1222ms	540ms
Number of requests	13	34	26
Downloaded resources' size	5.4MB	16.6MB	6.6MB
Load time	1.35s	1.12s	0.774s

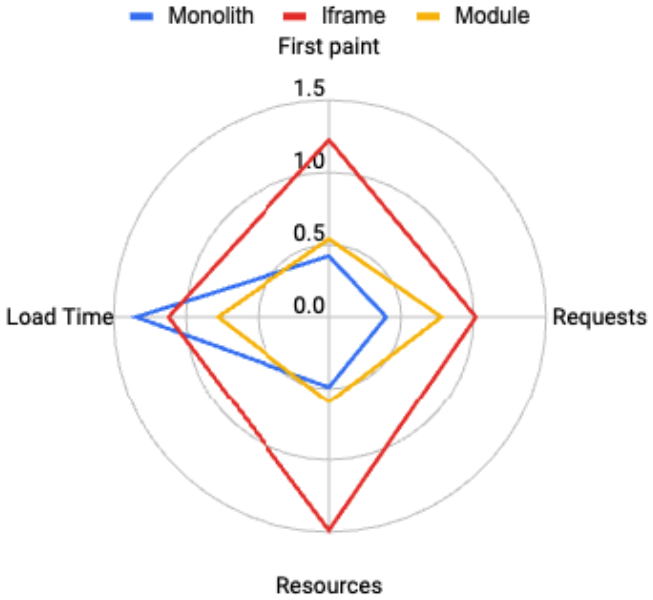


Fig. 5.1 Relative comparison of the implemented solutions

Although the results of the Module Federation solution show a lack of performance on some specific chapters for the studied implementation, we can assume that for larger applications, it will surpass the monolithic approach. Using decoupled code sources that work together to form an entire application provides teams with independence and scalability without reducing performance or degrading the user experience.

5.3 Conclusions

Based on the analysis conducted in this chapter, it can be concluded that micro-frontends offer a promising solution for overcoming the challenges posed by monolithic frontend applications. By dividing frontend applications into smaller, more manageable pieces, development teams can work more efficiently, resulting in shorter development cycles.

Chapter 6

Web3.0 and Decentralization; Practical Applications

6.1 Decentralized Authentication Using Web 3.0

This section addresses an innovative method of authentication based on decentralization and Ethereum blockchain technology, creating links between on-chain and off-chain resources. Current methods of authentication, the definitions of Web 3.0, and their applications are analyzed. The proposed technical stack and the advantages and disadvantages of Web 2.0 authentication compared to Web 3.0 are presented. Finally, a practical implementation of the proposed method is provided with a detailed guide, and the benefits of an anonymous authentication system are discussed. A study compares the authentication times of the proposed method with other similar mechanisms.

6.1.1 Secure Authentication Mechanisms

The most popular authentication mechanism is based on the user entering a username/e-mail and password combination to gain system access. However, if the password is compromised, the user could lose access to the system forever. There is a tendency to move away from the username and password combination in favor of using just a single-factor authentication method to shorten the time required to create an account on a certain platform. This study aims to explore the viability of completely anonymous authentication systems, thus bridging the operations carried out through the blockchain and outside the blockchain.

6.1.2 System Components

Unlike traditional authentication mechanisms, the proposed system requires an additional component, namely the electronic wallet through which the user's identity is validated.

Using an Ethereum electronic wallet together with a traditional web application can provide users with increased security and anonymity.

6.1.3 Authentication in the Web 2.0 Era

There are multiple modes of authentication in any web-based or user-installed applications, either based on user credentials or certificates relying on secondary validation of user identities. All Web 2.0 authentication methods have a centralized identity management system that stores the user's credentials and/or private information held by the user.

6.1.4 Authentication in the Web 3.0 Era

Web3 authentication [21] is the starting point for most decentralized applications and is based on the mechanisms incorporated into hardware or software electronic wallets, such as signing transactions and messages. My study is based on the message signing capability to verify that the user has access to the private key.

6.1.5 Implementation of the Authentication System Using Web 3.0

Proposed Authentication Flow

To connect an electronic wallet to a decentralized application, the user must confirm that the site can interact with the wallet, and the login process begins with obtaining a nonce from the backend, which is then digitally signed through the graphical interface using the wallet's address. After signing the nonce, the message is decrypted and validated by the backend, leading to the creation of a JWT that authenticates the user's identity for future sessions on the frontend.

Authentication Diagram

Figure 6.1 outlines the complete authentication flow using third-party electronic wallet software.

6.1.6 Comparative Performance Analysis of Authentication Methods

The proposed system was successfully implemented using Java and JavaScript, leveraging the power of the Spring and Angular frameworks. The library used to interact with the electronic wallet was Web3.js, as it benefits from large community support and frequent security updates.

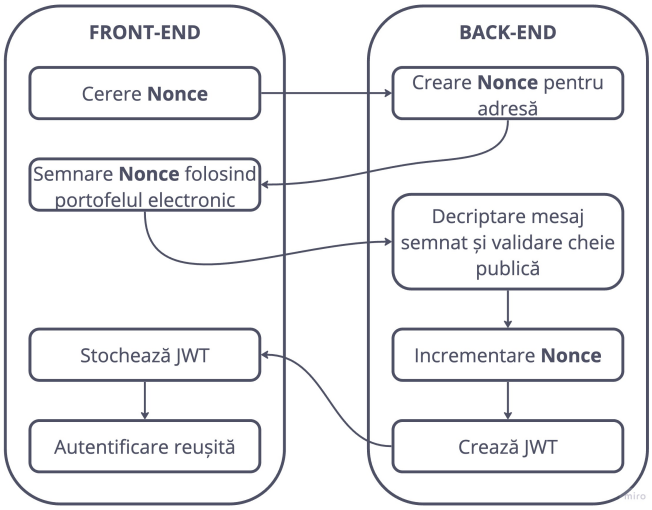


Fig. 6.1 End-to-end authentication flow.

To compare the proposed authentication mechanism with other relevant systems, a series of authentication operations was performed (Figure 6.2). A relevant authentication model used for comparison in this study is SMS authentication. Unlike traditional 2FA, SMS authentication requires only that the user confirm authentication by entering the code received via SMS, without needing to enter other authentication data. User reaction time and network request time were measured to see if the proposed mechanism has a faster end-to-end authentication time compared to similar solutions. After conducting a series of 50 consecutive authentication operations using 3 distinct methods (authentication through an electronic wallet, mobile electronic wallet authentication, and SMS authentication), it was observed that the average authentication time for an electronic wallet authentication operation is 2.698 seconds (Figure 6.2a). The average network request time is 375 ms and can vary depending on the network or server load. Similar to the virtual wallet authentication method, the request times for a mobile electronic wallet authentication operation average 355 ms. Using mobile electronic wallet authentication, which depends on an external server for communication, took an average of 7.294 seconds (Figure 6.2b). This represents a 170% increase compared to standard electronic wallet authentication (2.698 seconds), and the user's reaction time has the greatest impact on the total time spent (the user must scan a QR code and approve two distinct operations on the mobile phone).

Using a publicly available platform that implemented an authentication mechanism using SMS codes, a similar number of tests were performed, and the average time spent per operation was 13.799 seconds (Figure 6.2c). The network load for this experiment is significantly higher (3.782 seconds) due to the observed platform's architecture. User interaction times are longer because of the operations that must be performed (entering the phone number, waiting for the SMS, entering the SMS code, and waiting for validation). In the case of authentication using the SMS code (13.799s), an increase of 411%,

respectively 89% compared to the standard time for electronic wallet authentication (2.698s) and mobile electronic wallet authentication time (7.294s) was observed.

The waiting time for performing an end-to-end authentication can vary depending on the network load, user reaction times, and the implemented architecture. However, the proposed electronic wallet authentication setup appears to be faster compared to other observed solutions.

The results show that the proposed solution has a significantly reduced total authentication time compared to the traditional method. Performing an authentication using a browser extension software electronic wallet is five times faster compared to the observed SMS authentication mechanism.

6.1.7 Conclusions

Decentralized applications that rely on off-chain computations can only be accessed using an anonymous electronic wallet address. However, the identities of users holding certain accounts need to be validated through a KYC process for more sensitive financial operations.

6.2 A Practical Implementation of a Document Digital Document Signature System Using Blockchain Technology

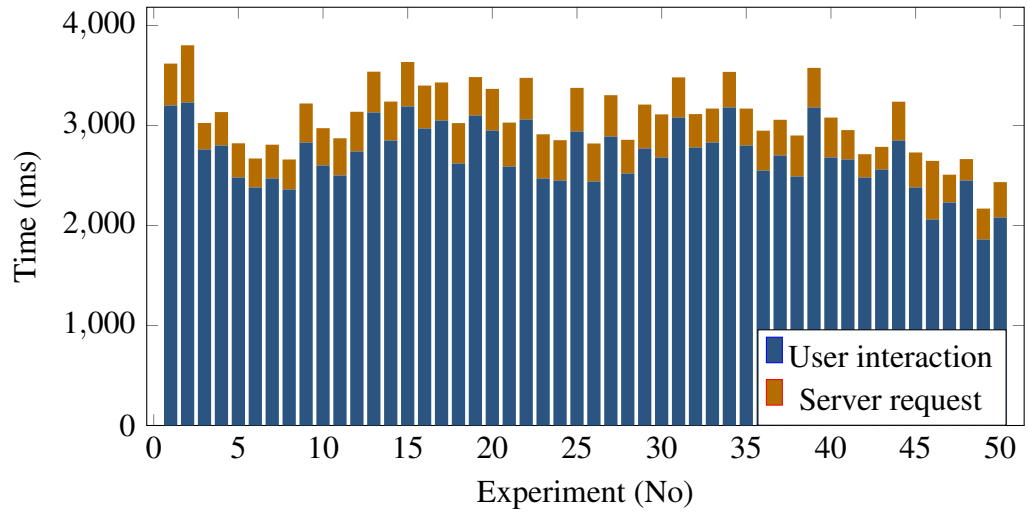
In the first part of this section, I will explore the types of document signature mechanisms, their history, availability, and their acceptance by the government. Then, I will continue by investigating types of checksum and hash techniques, followed by the section where I will describe a practical implementation proposal of our system, along with its advantages and disadvantages.

6.2.1 Platforms for Digital Signatures

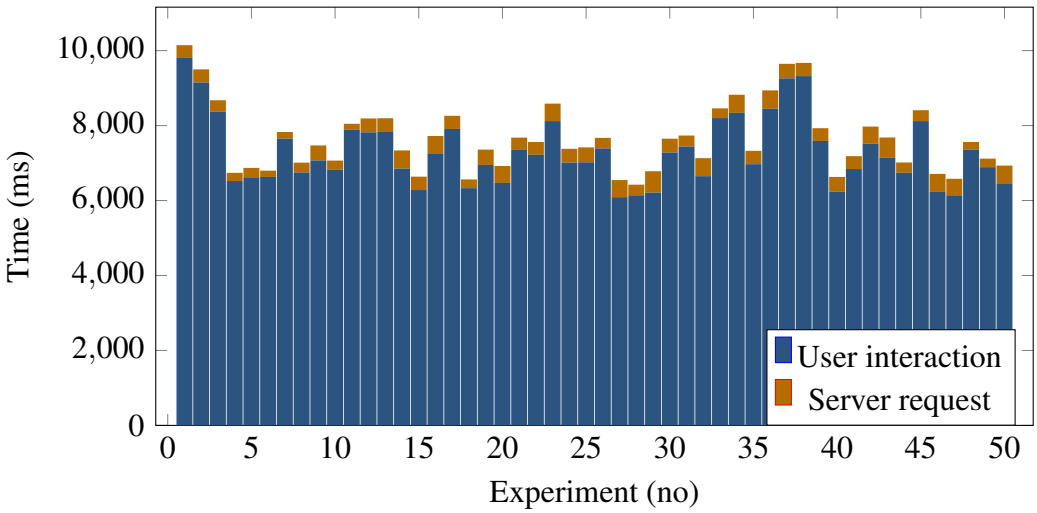
Our study aims to explore the feasibility of blockchain-based document signature systems using hash mechanisms and decentralization. Additionally, the proposed solution's implementation does not require modifying the original document by attaching signatures, unlike traditional document signature mechanisms."

6.2.2 Types of Document Signing

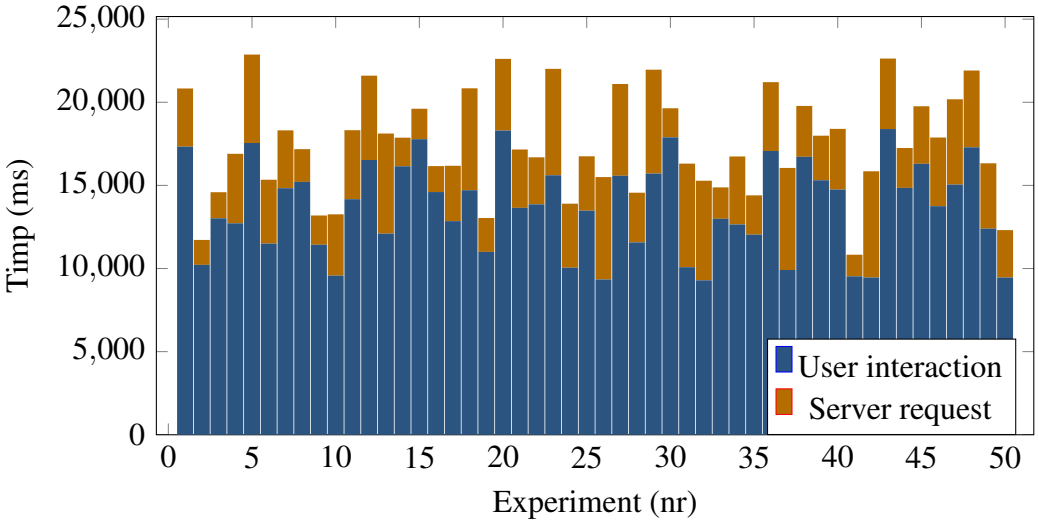
There are various types of content authentication for documents, ranging from physical presence to digital interaction with documents.



(a)



(b)



(c)

Fig. 6.2 Time spent for authentication. (**a**) Authentication with digital wallet; (**b**) Authentication with mobile digital wallet; (**c**) Authentication with SMS.

Digital certificates are issued by a trusted third party, such as a Certificate Authority (CA), and are used to verify the signer's identity [22]. When a document is signed using a digital certificate, the signature is encrypted using the private key associated with the certificate. This creates a secure record, evident against tampering attempts, of the document's content at the moment of signing [23]. The recipient of a digitally signed document can verify its authenticity and integrity using the public key from the certificate, while biometric and handwritten signatures provide personal authentication methods through unique physical features or manual markings to validate the agreement with the document's content.

6.2.3 Digital Signatures

- Cryptographic Hash: This method involves generating a unique hash value for a document using a cryptographic algorithm, such as SHA-256 [24]. The hash value is then encrypted using the signer's private key, creating a digital signature. The signature can be verified by decrypting it using the signer's public key and comparing the result with the original hash value.

- Digital Certificates: As mentioned earlier, digital certificates can be used to verify the signer's identity and the authenticity of the signature. This is done by verifying the certificate against a list of trusted CAs and ensuring that it has not been revoked.

6.2.4 Hashes and Checksums

Hashing is similar to a checksum, but it is generally used for security, not error detection. In devices that calculate the hash, an input or 'message' is introduced, and a fixed-size string of characters is returned, known as a hash value or message digest. Given the same inputs, we should always receive the same hash value. The collision rate of a hash function measures how likely it is for two different inputs to produce the same hash output.

Any hash function can experience a collision, but the probability of this happening with Keccak256 is extremely low. For example, the likelihood of a collision with Keccak256 is approximately 2^{-128}, which is practically zero for all practical purposes [25].

6.2.5 Implementation of the Proposed System

As mentioned earlier, the blockchain is a distributed digital ledger technology that allows for secure and transparent data storage on a network of computers, ensuring that all past data cannot be altered. Therefore, we can assume that using blockchain to read the list of signatories for each document hash is a valid alternative to modifying the documents themselves to attach signatures.

Components of the Document Signing System Using Blockchain

The proposed system consists of several parts that work together to realize decentralized document-signing processes. Of course, the validation of each individual's identity must be done manually to ensure that the concerned person is signing the document.

Document Hashing The document hashing component in the proposed system will be responsible for generating hashes from the input documents. These documents can be in any format, from .txt to .docx, .doc, .pdf [26]. There are several libraries available on the internet that will facilitate the faster implementation of any hashing algorithm.

```
1 import sha3
2
3 k = sha3.keccak_256()
4 k.update(file.read())
5 h3 = k.hexdigest()
6
7 print(h3)
```

Listing 6.1 Generating the hash using Python

Calculating the hash of a specific file can be achieved either on the application's back-end or front-end. For a better user experience and to reduce the computational power required for generating the hash, we can simply use a front-end library.

```
1 const keccak256 = require('keccak256')
2
3 console.log(keccak256('file contents').toString('hex'))
```

Listing 6.2 Generating hash keccak256 using JavaScript

Blockchain Since smart contracts and signature data are public, multiple applications can be created based on the basic system [27]. For instance, because the Ethereum blockchain can emit events for specific operations, applications can listen to these events and react accordingly to display off-chain data. Clauses can be added so that whenever all necessary parties have signed a certain document, specific operations on the blockchain can be executed.

Web Application The web application is designed to be the central interface and the main point of interaction with the document signature workflow. Interaction with the nodes of the blockchain network to retrieve information about the network's state, including transactions, blocks, and other data, is done through the Remote Procedure Call (RPC) protocol.

Component Diagram

As shown in Figure 6.3, numerous components interact to form the complete system. Since the system is public, as many secondary applications as possible can be created that will eventually interact with the basic system. A use case would be the conditioning of signing a certain document "hash" for a smart contract to be executed.

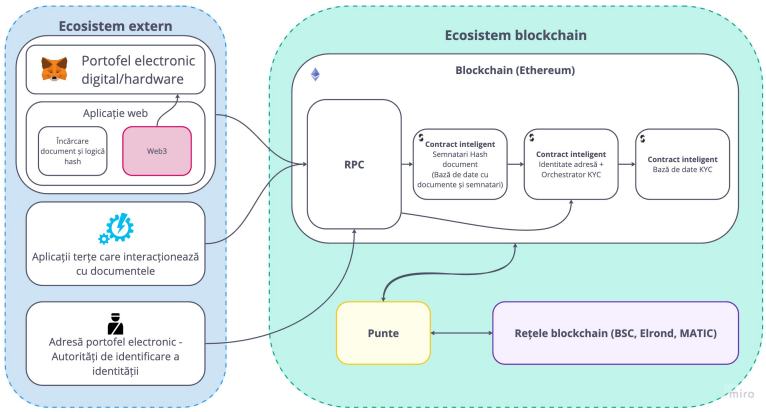


Fig. 6.3 High-level component diagram

Document Signing and Validation Flow through Blockchain

Similar to available digital signature processes, to sign our documents we first need to compute a hash of the targeted document and store it on the blockchain along with the signer's digital wallet address. If the targeted document is modified in any way, its hash will change and, therefore, the signature will not be able to be found.

Figure 6.4 presents a simplified flow of creating and validating document signatures. Verification can be done by manually calculating the hash of the targeted document and checking it by querying the blockchain or by using a graphical user interface that calculates the hash and displays all the signatures that have been applied to it, along with the person responsible for the signature.

In Figure 6.5, an example of the process of signing a document using the proposed solution above can be seen. The user logs in based on the digital wallet and uploads a document for signing. The application calculates the document's hash using the keccak256 algorithm and sends this hash to the blockchain.

The smart contract component installed in the blockchain for this application will take this hash and store it in the blockchain along with the user's digital wallet address authenticated in the application after their confirmation.

The blockchain thus provides a distributed environment for storing information at an exponentially reduced cost compared to traditional databases. In Figure 6.5, the cost of writing information to the blockchain can be seen: approximately 0.00014913 ETH, equivalent to about 0.27 USD, a cost that is paid by the end-user and not by

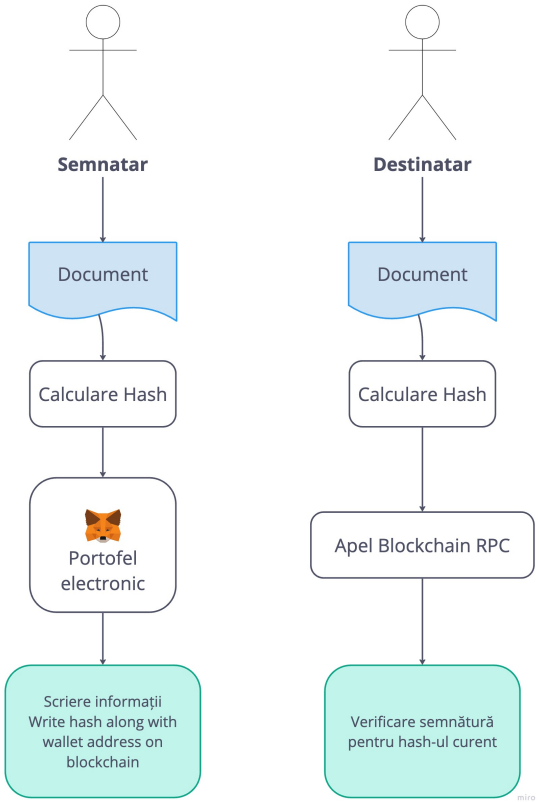


Fig. 6.4 Complete digital document signing flow

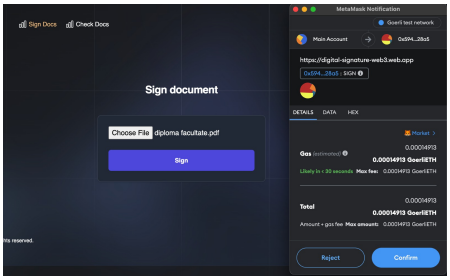


Fig. 6.5 Graphic example of document signing flow

the application that provides the electronic document signing service. Any subsequent operation of reading from the blockchain is free.

In Figure 6.6, an example of a process for verifying the signatories of a document can be seen. The user logs in based on the digital wallet and uploads a document for verification. The application calculates the document's hash using the keccak256 algorithm and sends this hash to the blockchain.

6.2.6 Conclusions

Although the implementation of the proposed system has great potential, there are several parts of the system that need to be addressed before it can become fully functional. Governments need to issue blockchain-linked identities that can be correlated with

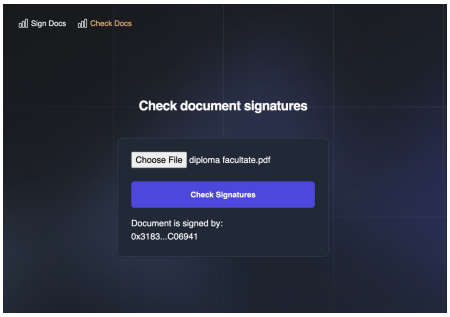


Fig. 6.6 Graphic example of verifying signatures applied to a document

specific digital wallet addresses. Without official recognition of software digital wallets, the signatures would have no value. Government legislation regarding digital signatures must be implemented to support blockchain-related mechanisms, such as those proposed.

Chapter 7

Conclusions

The research in this paper was dedicated to the study of the evolution of applications through the lens of their transition to the online environment, exploring aspects such as the scaling of graphical interfaces and, with a future perspective, exploring the consequences of the transition to a decentralized environment. Besides technological advances, the transition from natively installed software to sophisticated web platforms reflects transformations in societal norms and expectations and the development of the internet as an omnipresent tool. The ultimate goal was to provide a thorough examination of these changes, highlighting the significance of each stage in this evolutionary process.

7.1 General Objectives and Results

The transition from natively installable applications to web-based applications was perhaps an early sign of the scalable and interconnected potential of a digital universe offered by the internet. Beyond moving technical solutions onto a different platform, this transition represents a change in mindset and paradigm in creating software solutions. Web applications, unlike natively installable ones, have emphasized omnipresence and availability, thus transcending geographic boundaries, hardware limitations, and pointing to a future where updates in appearance, functionality, and performance can happen daily.

Many tech giants have already made the step towards web-only applications out of the desire to reach as many users as possible and have better control over the distribution of the solution. Comparing such an application, available in both installable and web formats, we observed that although there is an improvement in the performance of web applications compared to native applications, the transition to such an infrastructure requires ideological changes on the part of companies.

With the transition to the online environment, the need for scaling and performance improvement led not only to architectural changes for optimization and server resilience but also to the need for **scaling graphical interfaces using micro-frontends**. As

applications grew, so did their digital footprint, making them difficult to maintain and expand. The concept of micro-frontends emerged as a solution to this problem, thus allowing teams to break down monolithic GUI applications into small, independent applications, thereby accelerating development times and offering a consistent browsing experience across the entire platform.

Although the technologies that allow scaling graphical interfaces are still developing, there are stable solutions that allow separating GUI applications into micro-frontends. A comparative performance study was conducted, and the most efficient solution was module federation. Although this solution does not present obvious performance disadvantages, the advantages of the solution can be observed as the applications increase in size.

However, without a thorough examination of decentralization, especially through the prism of **blockchain and web3 environments**, the story of web applications would be incomplete. Decentralization is a philosophical as well as a technological shift. Its fundamental purpose is to change the paradigm of centralized control and singular authority, whether in the banking domain, data storage, or even application hosting. In addition to technological advantages such as fault tolerance and security, decentralization has positive effects, such as better user privacy and more equitable access to resources. The promise of a decentralized internet, where individuals truly own their data and digital assets, is reinforced by the growth of web3 environments.

It is important to consider the broader ramifications of changes as we analyze these massive evolutions. Over time, web applications have evolved from simple tools to extensions of our social fabric that affect how we interact, communicate, work, play, and even think. Their evolution reveals much about our own development as a digital society, about challenges, aspirations, and the continuous search for innovation.

As in any evolutionary journey, there are many risks and challenges along the way. Questions about data privacy, the role of centralized tech giants, the impact of emerging technologies like blockchain on the environment, and the digital divide will continue to be at the center of attention as online applications develop.

In this paper, we studied several practical applications of the decentralized environment using web3, and the results show that their performance can exceed that of classical systems. This fact, coupled with the fact that data is secure and transparent, reinforces the belief that web3 applications have a well-established place in the future evolutionary path of applications.

7.2 Original Contributions

The major contributions of the author in this thesis (methodologies and concepts) are summarized and presented in the following paragraphs, divided by the chapter number. Each contribution will contain the following information:

- a brief description of the contribution
  - additional information regarding the practical applications of the contributions in both the academic world and within private companies;
  - information regarding the uniqueness of the contribution;

In chapter 4

- Evaluation of the trends in migrating applications from the native installed environment to the online environment. Development of a methodology for comparing the performance of applications existing both in the installed and online environments.
  - The results obtained can be used for strategic planning of software development: Understanding the trends in application migration from the native to the online environment can help organizations decide how and where to invest in software development;
  - To my knowledge, I have not found similarities between this study and other studies in the literature;

In chapter 5

- Evaluation of options related to scaling Graphical User Interface (GUI) applications. Recommendations on scaling approach and a practical performance comparison of identified solutions;
  - Research and implementation of popular solutions for scaling micro-frontend applications, performance comparison between existing scaling solutions and the classic monolithic version of the same application;
  - Highlighting how different scaling approaches affect the performance of GUI applications can help software developers make more informed decisions about their design and implementation;
  - To my knowledge, I have not found similarities between this study and other studies in the literature;

In chapter 6

- Development of a practical decentralized authentication application using Web 3.0 and blockchain, which is very useful by being implemented in numerous practical applications in the field of distributed applications;
  - Implementation of an authentication solution using Web3.0 using JavaScript, Java technologies, and a wide range of electronic wallets;
  - The proposed solution can be used by organizations that rely on participant anonymity;
  - The proposed solution is popular among decentralized applications, but, to my knowledge, a similar study comparing performance does not exist in the literature;
- Development of an innovative method of digitally signing documents without altering them using electronic wallets and storing the signatures in Blockchain;
  - The study and creation of an architecture of a system that allows the signing of electronic documents using tools provided by the Blockchain ecosystem;
  - The proposed system can replace traditional systems of signing digital documents and can lead to the creation of decentralized applications that are secured by the immutable nature of Blockchain technology;
  - To my knowledge, I have not found similar approaches to document signing in the literature;

7.3 List of Original Publications

1. **Petcu, Adrian**, Bogdan Pahontu, Madalin Frunzete, and Dan Alexandru Stoichescu. A secure and decentralized authentication mechanism based on web 3.0 and ethereum blockchain technology. *Applied Sciences*, Vol 13(Nr 4):Pag 2231, 2023. doi: 10.3390/app13042231. URL https://doi.org/10.3390/app13042231, Q2, impact factor 2.9, eISSN: 2076-3417, WOS:000938088300001 *cited by **8** papers at **01.09.2023***

2. **Petcu, Adrian**, Madalin Frunzete, and Dan Alexandru Stoichescu. Benefits, challenges, and performance analysis of a scalable web architecture based on micro-frontends. *UPB Scientific Bulletin*, Vol 85(Nr 3):Pag 319, 2023, ISSN: 2286-3540, WOS:001052259100025

3. **Petcu, Adrian**, Madalin Frunzete, and Dan Alexandru Stoichescu. Evolution of applications: From natively installed to web and decentralized. In *International Conference on Computational Science and Its Applications*, pages 253–270. Springer, 2023, 3-6 July 2023, Athens

4. **Petcu, Adrian**, Madalin Frunzete, and Dan Alexandru Stoichescu. A practical implementation of a digital document signature system using blockchain. In *2023 13th International Symposium on Advanced Topics in Electrical Engineering (ATEE)*, pages 1–6. IEEE, 2023, 23-25 March 2023, Bucharest *cited by **one** paper at **01.09.2023***

5. Bogdan-Ionut Pahontu, **Petcu, Adrian**, Alexandru Predescu, Diana Andreea Arsene, and Mariana Mocanu. A blockchain approach for migrating a cyber-physical water monitoring solution to a decentralized architecture. *Water*, Vol 15(Nr 16):Pag 2874, Aug 2023. ISSN 2073-4441. doi: 10.3390/w15162874. URL http://dx.doi.org/10.3390/w15162874, Q2, impact factor 3.5, eISSN: 2073-4441 WOS:001056031300001

Research Papers

1. Adrian Petcu. Contribuţii la migrarea aplicațiilor din mediul standalone in online. In *Technical Report no 1*. University POLITEHNICA of Bucharest, 2018

2. Adrian Petcu. Evolutia aplicațiilor web: De la nativ la hibrid. In *Technical Report no 2*. University POLITEHNICA of Bucharest, 2018

3. Adrian Petcu. Micro front-ends: Micro aplicatii bazate pe tehnologii front-end. In *Technical Report no 3*. University POLITEHNICA of Bucharest, 2019

4. Adrian Petcu. Aplicatii web progresive. alternativa viabila a aplicatiilor native. In *Technical Report no 4*. University POLITEHNICA of Bucharest, 2019

5. Adrian Petcu. Web assembly avantaje si provocari. In *Technical Report no 5*. University POLITEHNICA of Bucharest, 2020

7.4 Future Work

The development of web applications will have an impact across a wide range of industries, from private business to social interactions, from governance to entertainment. The topics addressed in this thesis can be extended by developing the following subjects.

- Analysis of the long-term impact of transferring business costs to the users of the system in the context of **Web3.0** applications;

- Regulation: As decentralized web applications become more integrated into our societies, an increase in legislative frameworks to regulate their operation is expected. This will range from data privacy laws antitrust considerations, to rules ensuring equitable access;

- The impact of the introduction of digital currencies on businesses. The potential for changing cost models and taxation;
- The integration of digital identities into existing systems and the development of decentralized autonomous organizations;
- Asset Digitization: On **Web3.0** platforms, everything from real estate to intellectual property can be tokenized in digital form. This could democratize investment opportunities, allowing fractional ownership of assets that were previously accessible only to a select few through complex notarial procedures;

References

[1] Fitri Wulandari. What is Web3 | Definition and Meaning. https://capital.com/web3-definition, 2023. [Online; accesat 2023-07-02].

[2] M. Metcalf. Why Fortran? *ACM SIGPLAN Fortran Forum*, 2(1):13–14, 3 1983.

[3] Wikipedia. Timeline of programming languages — Wikipedia, the free encyclopedia. http://en.wikipedia.org/w/index.php?title=Timeline%20of%20programming%20languages&oldid=1164870771, 2023. [Online; Accesat 31-07-2023].

[4] Wikipedia. Git — Wikipedia, the free encyclopedia. http://ro.wikipedia.org/w/index.php?title=Git&oldid=15762410, 2023. [Online; Accesat 31-July-2023].

[5] Jackson Reeves. Git Best Practices for Team Collaboration — dev.to. https://dev.to/jtreeves/git-best-practices-for-team-collaboration-3bf0, 2022. [Accesat 31-07-2023].

[6] C. Selvaraj and S. Anand. Peer profile based trust model for P2P systems using genetic algorithm. *Peer-to-Peer Networking and Applications*, 5(1):92–103, oct 1 2011.

[7] E. C. Bank. Digital euro. https://www.ecb.europa.eu/paym/digital_euro/html/index.en.html, nov 8 2022. [Online; accesat 2023-08-02].

[8] S. Licata. Digital Dollar. https://digitaldollarproject.org/, jun 23 2023. [Online; accesat 2023-08-02].

[9] Xiong Zhang, Wei T Yue, and Wendy Hui. Software piracy and bundling in the cloud-based software era. *Information Technology & People*, 32(4):1085–1122, 2019.

[10] Anfel Selmadji, Abdelhak-Djamel Seriai, Hinde Lilia Bouziane, Rahina Oumarou Mahamane, Pascal Zaragoza, and Christophe Dony. From monolithic architecture style to microservice one based on a semi-automatic approach. In *2020 IEEE International Conference on Software Architecture (ICSA)*, pages 157–168. IEEE, 2020.

[11] Francisco Ponce, Gastón Márquez, and Hernán Astudillo. Migrating from monolithic architecture to microservices: A rapid review. In *2019 38th International Conference of the Chilean Computer Science Society (SCCC)*, pages 1–7. IEEE, 2019.

[12] Nicola Dragoni, Saverio Giallorenzo, Alberto L Lafuente, Manuel Mazzara, Fabrizio Montesi, Ruslan Mustafin, Lara Safina, and Gianluigi Zavattaro. Microservices: yesterday, today, and tomorrow. *Communications of the ACM*, 60(6):36–44, 2017.

[13] Calin CONSTANTINOV, Lucian IORDACHE, Adrian GEORGESCU, Paul-Stefan POPESCU, and Mihai MOCANU. Performing social data analysis with neo4j: Workforce trends & corporate information leakage. In *2018 22nd International Conference on System Theory, Control and Computing (ICSTCC)*, pages 403–406, 2018. doi: 10.1109/ICSTCC.2018.8540645.

[14] Shubhani Aggarwal and Neeraj Kumar. Chapter eleven - cryptographic consensus mechanisms - ntroduction to blockchain. In Shubhani Aggarwal, Neeraj Kumar, and Pethuru Raj, editors, *The Blockchain Technology for Secure and Smart Applications across Industry Verticals*, volume 121 of *Advances in Computers*, pages 211–226. Elsevier, 2021. doi: https://doi.org/10.1016/bs.adcom.2020.08.011. URL https://www.sciencedirect.com/science/article/pii/S0065245820300668.

[15] Ahto Buldas, Dirk Draheim, Mike Gault, Risto Laanoja, Takehiko Nagumo, Märt Saarepera, Syed Attique Shah, Joosep Simm, Jamie Steiner, Tanel Tammet, and Ahto Truu. An ultra-scalable blockchain platform for universal asset tokenization: Design and implementation. *IEEE Access*, 10:77284–77322, 2022. doi: 10.1109/ACCESS.2022.3192837.

[16] Mihai Liviu Despa. Comparative study on software development methodologies. *Database Systems Journal*, 5(3):37–56, 2014.

[17] Samuel Kounev, Klaus-Dieter Lange, and Jóakim von Kistowski. *Systems benchmarking: for scientists and engineers*, volume 1. Springer, 2020.

[18] A. Patwardhan. How to scale Frontend apps using Micro Frontends | WealthDesk. https://wealthdesk.in/blog/scaling-frontend-apps-using-micro-frontends/, may 6 2022. [Online; accesat 2023-06-21].

[19] Andrey Pavlenko, Nursultan Askarbekuly, Swati Megha, and Manuel Mazzara. Micro-frontends: application of microservices to web front-ends. *J. Internet Serv. Inf. Secur.*, 10(2):49–66, 2020.

[20] Luca Mezzalira. *Building Micro-Frontends*. " O'Reilly Media, Inc.", 2021.

[21] Zhuotao Liu, Yangxi Xiang, Jian Shi, Peng Gao, Haoyu Wang, Xusheng Xiao, Bihan Wen, Qi Li, and Yih-Chun Hu. Make web3.0 connected. *IEEE Transactions on Dependable and Secure Computing*, 19(5):2965–2981, 2022. doi: 10.1109/TDSC.2021.3079315.

[22] Zhen Qin, Chen Yuan, Yilei Wang, and Hu Xiong. On the security of two identity-based signature schemes based on pairings. *Information Processing Letters*, 116(6): 416–418, 2016. ISSN 0020-0190. doi: https://doi.org/10.1016/j.ipl.2016.02.003. URL https://www.sciencedirect.com/science/article/pii/S0020019016300096.

[23] Qiuxia Zhang, Zhan Li, and Chao Song. The improvement of digital signature algorithm based on elliptic curve cryptography. In *2011 2nd International Conference on Artificial Intelligence, Management Science and Electronic Commerce (AIMSEC)*, pages 1689–1691, 2011. doi: 10.1109/AIMSEC.2011.6010590.

[24] Matthieu Finiasz and Nicolas Sendrier. Security bounds for the design of code-based cryptosystems. In *Advances in Cryptology–ASIACRYPT 2009: 15th International Conference on the Theory and Application of Cryptology and Information Security, Tokyo, Japan, December 6-10, 2009. Proceedings 15*, pages 88–105. Springer, 2009.

[25] Shahram Bakhtiari, Reihaneh Safavi-Naini, Josef Pieprzyk, et al. Cryptographic hash functions: A survey. Technical report, Citeseer, 1995.

[26] J. Zhang, Y. Wang, and X. Liu. Blockchain-based digital signature: A survey. *IEEE Communications Surveys and Tutorials*, 2020.

[27] Weidong Fang, Wei Chen, Wuxiong Zhang, Jun Pei, Weiwei Gao, and Guohui Wang. Digital signature scheme for information non-repudiation in blockchain: a state of the art review. *EURASIP Journal on Wireless Communications and Networking*, 2020(1):1–15, 2020.

[28] **Petcu, Adrian**, Bogdan Pahontu, Madalin Frunzete, and Dan Alexandru Stoichescu. A secure and decentralized authentication mechanism based on web 3.0 and ethereum blockchain technology. *Applied Sciences*, Vol 13(Nr 4):Pag 2231, 2023. doi: 10.3390/app13042231. URL https://doi.org/10.3390/app13042231.

[29] **Petcu, Adrian**, Madalin Frunzete, and Dan Alexandru Stoichescu. Benefits, challenges, and performance analysis of a scalable web architecture based on micro-frontends. *UPB Scientific Bulletin*, Vol 85(Nr 3):Pag 319, 2023.

[30] **Petcu, Adrian**, Madalin Frunzete, and Dan Alexandru Stoichescu. Evolution of applications: From natively installed to web and decentralized. In *International Conference on Computational Science and Its Applications*, pages 253–270. Springer, 2023.

[31] **Petcu, Adrian**, Madalin Frunzete, and Dan Alexandru Stoichescu. A practical implementation of a digital document signature system using blockchain. In *2023 13th International Symposium on Advanced Topics in Electrical Engineering (ATEE)*, pages 1–6. IEEE, 2023.

[32] Bogdan-Ionut Pahontu, **Petcu, Adrian**, Alexandru Predescu, Diana Andreea Arsene, and Mariana Mocanu. A blockchain approach for migrating a cyber-physical water monitoring solution to a decentralized architecture. *Water*, Vol 15(Nr 16):Pag 2874, Aug 2023. ISSN 2073-4441. doi: 10.3390/w15162874. URL http://dx.doi.org/10.3390/w15162874.

[33] Adrian Petcu. Contribuţii la migrarea aplicațiilor din mediul standalone in online. In *Technical Report no 1*. University POLITEHNICA of Bucharest, 2018.

[34] Adrian Petcu. Evolutia aplicațiilor web: De la nativ la hibrid. In *Technical Report no 2*. University POLITEHNICA of Bucharest, 2018.

[35] Adrian Petcu. Micro front-ends: Micro aplicatii bazate pe tehnologii front-end. In *Technical Report no 3*. University POLITEHNICA of Bucharest, 2019.

[36] Adrian Petcu. Aplicatii web progresive. alternativa viabila a aplicatiilor native. In *Technical Report no 4*. University POLITEHNICA of Bucharest, 2019.

[37] Adrian Petcu. Web assembly avantaje si provocari. In *Technical Report no 5*. University POLITEHNICA of Bucharest, 2020.