

Evolution of Applications: From Natively Installed to Web and Decentralized

Adrian Petcu(✉), Madalin Frunzete, and Dan Alexandru Stoichescu

Faculty of Electronics, Telecommunications, and Information Technology, University Politehnica of Bucharest, Bucharest, Romania
adrian.petcu@stud.etti.upb.ro
https://sdetti.upb.ro/

Abstract. Applications evolve from year to year either to fit consumer needs or to increase productivity and simplify work for developers. With the rapid development of web technologies, there has been a significant shift towards web applications as a means to distribute software instead of traditional, installable packages. This paper analyses the evolution of applications along with challenges and strategies to overcome them. More specifically, we discuss the trend of applications being shifted to online versions rather than native-oriented ones, which must be installed on the users' devices. Comparisons for performance and speed of development have been studied, considering current limitations and advantages/disadvantages. Ultimately, we conclude that while transitioning to web applications requires careful planning and execution, the benefits of this shift make it a worthwhile endeavor for many organizations.

Keywords: evolution · application · native · native · pwa · web3

1 Introduction

Throughout the past few decades, software distribution has seen significant evolution. Formerly, software was distributed via installable packages, requiring manual installation on each device, whereas, currently, software applications take the form of websites. The introduction of web technologies has led to a shift toward web applications accessed via a web browser and do not require installation. This change offers various benefits, including more straightforward accessibility, enhanced scalability, and decreased maintenance expenses. In addition, consumers' ability to use web applications from any device with an internet connection has increased their popularity.

As a result, more and more businesses are considering migrating their programs from installable packages to online applications. Nevertheless, this change presents several obstacles, including technological needs, user experience, and cultural and organizational constraints. This study examines these obstacles and proposes solutions for migrating from installable packages to web applications successfully. By identifying these obstacles and options, companies can make

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2023
O. Gervasi et al. (Eds.): ICCSA 2023 Workshops, LNCS 14104, pp. 253–270, 2023.
https://doi.org/10.1007/978-3-031-37105-9_18

educated decisions regarding whether or not to migrate their applications and how to do so successfully.

Transitioning applications from native platforms to the web poses several user experience and technical challenges. This transition can be further explored in the area of achieving decentralization using blockchain. In the first section, we explore the evolution of applications, their contents, and the types of applications. Further on, we explore challenges posed by migrating applications, and we conclude with a performance analysis 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 RAM consumption.

2 Applications

An application is any material (software or hardware) designed for end-users to use in order to fulfill specific needs or for entertainment. Initially, the term "application" in the software world referred to bundles of code written in various programming languages (C++, Java, etc.) meant to be executed on a device containing an operating system (Windows/Linux).

Operating systems are system software designed to ensure proper functionality of a hardware device (personal computer, laptop, mobile phone) and provide common services for applications. For example, an application designed to interact with hardware peripherals (mouse, keyboard) would communicate with the operating system and not with the device directly.

Applications usually require "installation" on the operating system as a step before being executed and used. As a user, you would have to download the bundle (executable), install it (or unpackage it), and execute the entry point of the application (an executable file ".exe" or ".sh"), then the entry point would manage the dependencies and static assets.

Users would have to verify the source of the application in order to protect themselves from malicious software design to gain control of their operating system and personal data.

2.1 Application Package Contents

Application bundles usually follow the same pattern - an entry point, libraries, and static assets. The entry point of an application, depending on the technology, is an executable file or a simple HTML (Hyper-Text-Markup-Language) file.

The installation step of an application consists of unpacking a file in a user-predefined location along with adding entries in the operating system for the proper functioning of the application (user settings and low-level configurations) in the registry.

When installing Windows OS-based applications, low-level configurations are stored in the Windows Registry, which is a hierarchical database designed to store information, settings, and options for applications and hardware. For example, when a program is installed, a new subkey containing settings such as a

program's location, its version, and how to start the program are all added to the Windows Registry.

For Linux-based applications, the installation step is simplified as there is no central "registry" of configurations for the applications. Unpacking the application bundle contents into a predefined user folder will suffice as the configuration files and settings are stored in the same folder as the application.

As presented in Fig. 1 applications usually have a single entry point and may contain multiple layers in which the information is organized. File types may vary between libraries, binary executables, configurations, and internal storage.

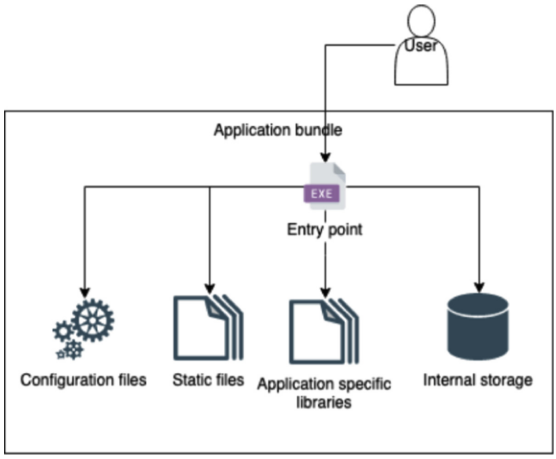


Fig. 1. Application bundle contents

2.2 Types of Applications

As stated in the previous part, applications rely on operating systems to provide core functionality and access to hardware. With the rapid expansion of the internet and its benefits, several types of applications have become available for end-users.

Native, User-Installed Applications. These applications require a physical device where the user manually performs the installation step, whether by downloading an executable and running it or by using a third-party application designed to download and install applications (e.g., Windows Store, Google Play, Apple AppStore) [7].

User-installed applications require regular updates to protect the user from potential security breaches and bring new features and functionalities.

Depending on the nature of the application, internet connectivity is not required. The user can benefit from the feature in an isolated environment with

fast response [9] and interaction as there are no other dependencies. Hardware resources and storage would become problematic if the desired application performed multiple tasks demanding more storage and computational memory. For example, from one version to another, the user might encounter a situation requiring upgrading his hardware/operating system to continue using the application.

Web-Based Applications. Web-based applications mimic the behavior and interaction of a typical native application but with limited or no access to physical hardware.



Fig. 2. Website from 1996

Due to browser limitations, web-based applications were initially trivial and provided a not-so-friendly user experience as the user would have to wait between screens [3] as seen in Fig. 2. Compared with native applications, where the end-user would get instant feedback, this was considered a disadvantage. With the evolution of browsers, applications did not require the user to wait for a full-page refresh to continue to the next screen. This provided a bridge between native applications and web-based applications in which the users would benefit from faster feedback.

The main difference between a web-based and a user-installed application is the installation step. The end-user must only access a web page to start using the application, skipping the installation step (procuring the installable package, running it, configuring the product, etc.). Resources and storage would no longer be a problem as the web server would handle the computational and storage load

and only static resources and plain text would be sent to the user. The evolution of browsers paved the way for hardware interaction, such that users could access and interact with physical hardware like cameras, file systems, etc.

Web applications can be accessed anywhere and from any device as long as it has a browser installed. The information is saved in real-time [12], and hand-off can be made quickly between devices to continue working on the task at hand from a different device, thus, increasing mobility. Even though web applications provide easy access to the user, they are costly to the entity hosting them. The computational and hardware resources which initially were the responsibility of the end-user now lie in the responsibility of the server hosting the application, which means that the costs of maintaining and hosting the applications would be transferred to the application owner.

Unlike regular personal computers, application servers can scale when the load increases to keep the response time in parameters for an increase in user number and even for surges like multiple users accessing the web app after clicking an advert. Server scaling can be either vertical, which means increasing the resources of a particular server (RAM, CPU, storage), or horizontal, which means creating more virtual instances of the same server to handle the temporary increase in requests as seen in Fig. 3.

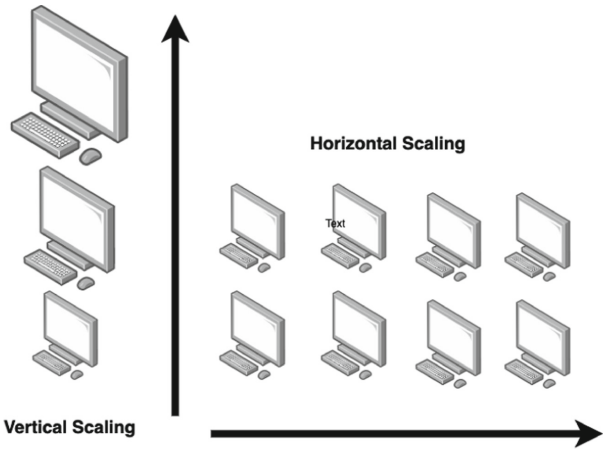


Fig. 3. Horizontal vs vertical scaling

Hybrid Web Applications. These applications combine the benefits of web-based and native applications, providing offline availability and one source code for all platforms, making the development of applications more accessible. Hybrid web applications are written primarily on web-oriented programming languages (HTML, CSS, JS) and are executed within a browser extended to communicate with the native elements of the specific device [5].

For example, communication with the camera or surfing through the phone agenda can be achieved using the cross-native library already developed by third-party vendors, thus enabling interaction with the native hardware from the web application.

The main advantage of hybrid application development has mostly been based on the philosophy of write-once, run everywhere where companies would try to create a cross-platform application using a single code base [13]. Hybrid applications can be distributed through app stores like native applications and are typically developed using frameworks like Apache Cordova, PhoneGap, and Ionic/Capacitor.

Developing hybrid mobile applications becomes more and more accessible as developers oriented on web development can be assigned to mobile development projects in no time as the initial setup is fast and code is written in widely known frameworks like (Angular, React, and Vue).

Table 1. Advantages of hybrid web and native applications

Application	Advantages	Disadvantages
Web Hybrid	– Uses Web technologies – Write once deploy anywhere – Large developer community – Offline support	– Lower performance – Limited access to native features – Dependance on 3rd party frameworks – Limited offline access
Native	– High Performance – Access to native functionality – Improved user experience – Offline capabilities	– Platform-specific development – App distribution – Limited cross-platform capabilities – Higher development costs

As we can observe from Table 1, there are some disadvantages of developing hybrid web applications, generated mainly by the smoothness of the transitions in the UI and the general consistency [6] with the native elements like menus, dropdowns, and selectors but also in performance and native functionality access.

To close the gap between hybrid web and native development, there are some frameworks that allow developers to write code only once and build/bundle the code for multiple devices. The general layout of these applications is written in a meta-syntax which is converted by tools in native code, and, by selecting the desired OS output [17], the framework will generate the application bundle for the specific OS (Windows, Android, iOS).

Progressive Web Applications (PWA). Developing multiple platforms has become a popular alternative for companies without resources or wanting to hire specialized mobile developers for each target platform. Until recently, the web platform lagged behind mobile-centric innovation [1] in its ability to compete with native applications or multiple platforms. Instead, a new set of standards backed by the Google Web Fundamentals group is trying to bridge this gap by introducing features such as offline support, background synchronization, and

installation on the web home screen [19] (very similar to how native applications behave).

Progressive Web Applications (PWA) refers to a term created by Russel and Bermanan (2015) in a blog post that covers initial implementation ideas. PWAs are defined by a set of concepts and keywords, including progressive, responsive, independent of connectivity, application-like, fresh, secure, discoverable, reusable, installable, and connectable (Addy Osmani, 2015).

These are PWA's contributions to unifying the mobile experience, where web applications can be installed and distributed without application markets, work without Internet connectivity, receive push notifications [4] and look like native applications.

PWAs would be a better alternative to both native applications and hybrid web applications. For example, both would require installing and searching with an internet search engine that would not find application-specific content meaning that application owners [10] would lose traffic as install conversion would be low, plus the applications with content could not be found via a search engine.

The core of PWA's functionality is assured by the Service Worker API, which allows developers to use a background script that acts as a network and device proxy.

The advantages of using progressive web applications are as follows:

1. Offline functionality
2. Great performance (seamless transitions and page loads)
3. No installation and updates are required (installing is done by saving the link to the homepage, which then acts like an app)
4. Platform-specific features (push notifications)
5. Low data consumption
6. App store independent
7. Discoverable (appear in search engines)
8. Linkable (installing an app is just navigating to a specific link)
9. Responsive (adapt to the browser size)

2.3 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 fast-evolving world of the internet, this is a significant disadvantage. Some applications provide online synchronization of users' configurations and contents. However, to access that information [2], the user must install the same application on multiple devices connected to the internet.

There is also the problem of version compatibility. Often users need to keep the application's version installed on multiple devices [14] in sync to benefit from internet synchronization. This provides discomfort for the end-user as he needs to check and install the latest version of the application regularly.

The information stored on local devices is often lost in case of a hardware malfunction or natural disasters. This results in a very low resilience, and for

Table 2. Advantages and disadvantages of user-installed applications

Advantages	Disadvantages
– Application isolation for highly sensitive documents – No need for internet connection and synchronization	– Low level of resilience – Version compatibility – Regular application updates need to be installed – Access to user documents is limited by hardware – Cannot be accessed unless the application is installed

the user to protect himself against such events, he might be needed to perform regular backups [8] of the data and configuration (if the application provides such functionality).

From the data security point of view, applications installed on personal hardware are more secure as the user can decide whether to connect the device to the internet. The user can interact with the application in isolation on a local machine if the information is highly sensitive.

As presented in the Table 2 there are several advantages and disadvantages to using a user-installed application compared to web applications.

3 User Experience and Technical Challenges

It has been shown that the user experience (UX) of software has a substantial influence on adoption and usage. As enterprises contemplate migrating their programs from installable packages to online applications, it is crucial to consider their impact on user experience (UX). There are multiple technical considerations involved in achieving a migration of software from the choice of technologies to security. This chapter presents an overview of the most significant user experience (UX) difficulties posed by the move to online applications as well as the technical difficulties associated with techniques for overcoming them.

3.1 User Experience

Challenges. The transition from installable packages to online applications presents a number of UX difficulties. Performance poses a significant concern. Web applications may be slower than locally installed applications due to factors such as network latency, server load, and browser compatibility issues. This can lead to resistance from users as locally installed applications might be faster and more responsive to user commands.

Online applications usually require an internet connection to operate, limiting their use in regions with poor access. In addition, web applications may not offer the same functionality or capabilities as locally installed software, resulting in user frustration and resistance. While switching to online applications, users may be required to learn new interfaces or procedures, which can be complex and time-consuming.

Strategy. Numerous techniques can be used to overcome the UX difficulties posed by the migration to online applications. One strategy is performance optimization. This involves approaches for improving the performance of online applications, including server-side rendering, caching, and minification. Providing offline access to web applications is another method [20]. This may be accomplished with service workers and local data caching, as presented previously using PWA.

Feature parity is another essential factor to consider as users might get frustrated if features are missing from an application that is intended to replace a legacy solution. Using native browser APIs and frameworks is one method for guaranteeing that online applications have the same capabilities and functionality as locally installed applications.

Users can adjust to new interfaces or processes with the aid of user training and assistance. Techniques for offering training materials, tutorials, and support resources can simplify the shift to online applications.

Device Hand-Off Between Applications. Working with installed applications might not be very convenient if the user is always on the move and must shift from one device to another in order to continue work.

The term hand-off or handover originally came from the ability of cellular networks to transfer an ongoing call or data session from one channel connected to the core network to another channel for the phone user to have a seamless experience during a call and in transit.

At the time of writing this article, application hand-off allows users with devices with the same core operating system to instantly shift from one device to another for basic actions and applications (writing emails, surfing the web). Operating systems provide this functionality for core applications as it requires elevated access to the operating system kernel. Application developers are provided with access and documentation to implement this feature, but a limited number of applications do so.

3.2 Technical Requirements

Challenges. Transitioning from installable packages to online applications presents several technological difficulties. Data security is a significant challenge. Cross-site scripting (XSS) and SQL injection may make online applications more susceptible to attack than locally installed programs. This might result in data breaches and other security problems, which can be mitigated with careful consideration of the differences between online environments and local environments.

Another area for improvement is data storage. Online applications may require a different approach to data storage than locally installed programs, posing a challenge for enterprises with an established data architecture. In addition, online applications require additional resources than locally installed programs, which can be costly and require the purchase of expensive hardware or cloud infrastructure.

Access to specific machine data is also an important factor to consider, as some factory applications might require interaction with machinery in order to extract data and act according to it.

Limitations of web browsers for more complex actions performed by specialized applications is another factor to consider when migrating applications to a web environment. Connectivity to computer ports to interact with hardware might also present a challenge.

Strategy. Various approaches might be utilized to address the technical difficulties associated with the migration to online applications. Using secure development methods, such as regular code reviews, penetration testing, and encryption is one strategy, similar to the ways of work when developing a native application. Using secure hosting and cloud infrastructure [18] is another method. This usually involves utilizing suppliers with robust security protocols and updating software and systems regularly.

Data splitting and encryption can be employed to handle data storage. Using techniques like server-side rendering and caching, it is possible to optimize performance and improve user experience. Lastly, resource demands can be met via strategies such as load balancing and auto-scaling.

4 Web3 and Decentralization

4.1 Web3

As previously researched, Web3 is based on blockchain technology [16], a decentralized and immutable ledger that enables secure and transparent transactions without the need for intermediaries. Web3 uses blockchain technology to provide decentralized data storage, identity management, and user-to-user transactions.

As opposed to Web2, also known as the "Social Web" which is the current stage of the internet which is rather dominated by social media, mobile devices and cloud computing, web3 has a strong focus on decentralization and open web.

The most popular stack for developing decentralized applications relies on the Ethereum Virtual Machine (EVM) blockchain platform which benefits the support of numerous APIs and libraries that help application developers achieve their goals in a shorter amount of time.

Ethereum is a public blockchain that allows decentralized applications to be deployed and run inside it [15]. Smart contracts are the most significant enablers for transitioning to a decentralized application as they allow developers to write code directly in the blockchain. They execute automatically and autonomously when specific predetermined conditions are met. In the Ethereum context, smart contracts are written in a high-level programming language named Solidity and are compiled into bytecode that can be executed on the EVM.

As seen in Fig. 4 Web3 heavily relies on peer-to-peer interaction and decentralization of data, thus, reducing the risk of having a single source of truth or failure.

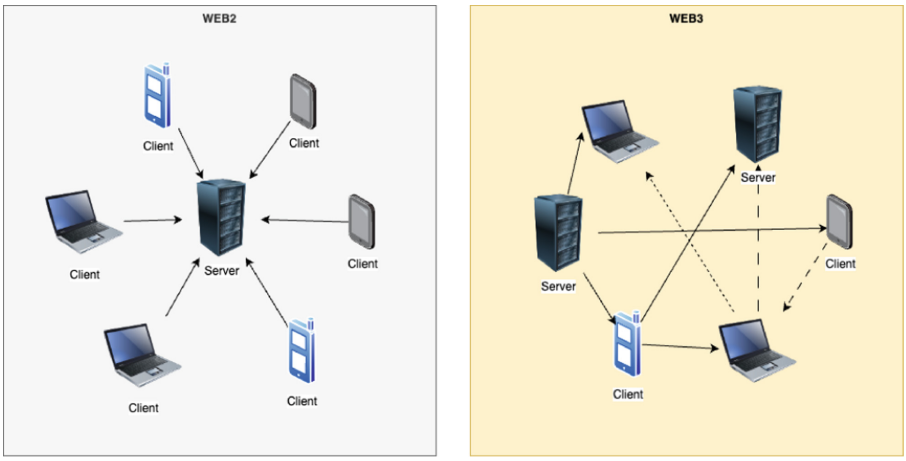


Fig. 4. Web2 vs Web3

4.2 Key Points into Transitioning to a Decentralized Infrastructure

The transition from Web2 to Web3 involves a substantial move toward decentralization and an analysis of the core business of the application in scope must be taken into consideration before proceeding with the transition. Applications designed for private use inside a company may benefit from a decentralized architecture since the nature of blockchain transactions relies on immutability, thus, better tracking of resources can be put in place.

When migrating to a public blockchain and a decentralized architecture, governance, security, interoperability, and privacy must be considered.

From a governance perspective, the nature of Web3 implies that there should not be any controlling authority; instead, decisions should be made by consensus among network participants. Transparency of the governance is important and it should be inclusive and open to all network participants.

Since using a public blockchain involves that data is public, a proper audit of the smart contracts should be put in place to avoid any attacks. Secure communication between the application and the blockchain should be considered.

Applications designed to run in a decentralized environment must consider that Web3 networks are interoperable, meaning they can communicate and share data with other networks. The adoption of open standards and protocols should be considered to ensure that the application operates as designed and can be seamlessly integrated with other networks.

Web3 networks must protect the privacy of network participants while also ensuring transparency and accountability. Scenarios in which the full identity of a user is not known are often and might require changing the business logic of the intended application to allow anonymous participants.

5 Research

Performance is the most important aspect of web applications compared to installable packages. Having a performant application can also be translated into a better user experience and better productivity. This study aims to compare the startup time, execution time, and memory usage of an application being available both under an installable package form as well a web application. Comparing bundle sizes between different versions of the same product might provide insights into the evolution of storage required to run a specific application.

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

Devices used for comparison:

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

Internet connectivity:

- 230 Mbps, Bucharest, Romania

Browser:

- Google Chrome, Version 111.0.5563.64 (arm64)

Running multiple simulations of opening a file using both laptops and both solutions will provide valuable insights into the performance difference between the online version and the user-installed version of the same application.

Time to first paint and RAM consumption are essential factors to consider [11] as they provide information regarding the user experience. Browser resource caching has not been deactivated, and the installed application has been completely closed after each file opens.

6 Results

The study has been carried out successfully using two different types of files (small and large) - 390kb and 32MB. Comparing the loading times and RAM consumption on both devices offers a good overview of the differences.

User reaction times and network request times have been measured in order to observe whether the online version of the same application loads faster compared to the locally-installed version.

The time spent opening the files has been measured from the start of the process (double-click the executable file for the natively installed application and click the link for the web application) to the time when the document becomes first interactive for some 20 consecutive runs using the two mentioned devices.

Calculation of relative improvement has been achieved using Eq. 1 formula, where N represents the new value and O represents the original value.

$$RI = \frac{N - O}{O} \tag{1}$$

It has been observed that using the installed application, the time spent opening a small file is an average of 3.994 s, while opening the same file on the web version has an average of 2.160 s on **Device 1**. Running the same tests on a larger file, the average load time for the native app was 4.396 s compared to the web version, which had an average of 3.032 s. This represents a 45.93% respectively 31.03% improvement in time spent on the online version compared to the installed application as seen in Fig. 5a. Overall average time decrease has been roughly 38%.

RAM consumption on **Device 1** for the operation of opening the two files using the local application has been observed to have an average of 351MB and 367MB for the small and large files, while for the web version, averages were 227MB and 239MB respectively which represents an average of 36% less RAM consumption as seen in Fig. 6a.

Running the same tests on **Device 2** apart from the overall decrease in time spent compared to **Device 1** (which is expected since the specifications of the devices are different), we have observed that opening the small file on the native application and on the web version took an average of 2.640 s and 1.282 s respectively. For the larger file, the results were 2.836 s and 1.630 s. An average 51.43% respectively 42.5% decrease in time spent has been observed. Overall the time decreased by 47%. Figure 5b shows the evolution of tests on device 2.

RAM consumption on **Device 2** for the operations using the local application has been observed to have an average of 250 MB and 262 MB for the small and large files, while for the web version, averages were 275 MB and 205MB respectively which represents an average of 6.25% less RAM consumption as seen in Fig. 6b.

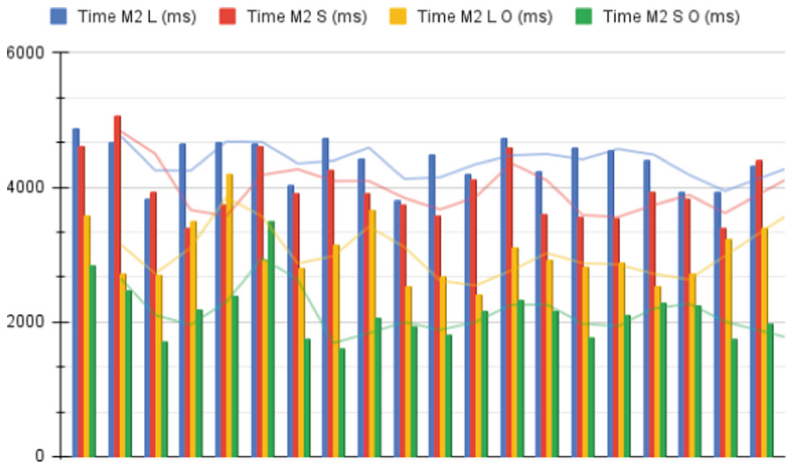
The expected time for performing both operations, using the web application and the locally installed application, may vary based on network load, internet speed, and device load.

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

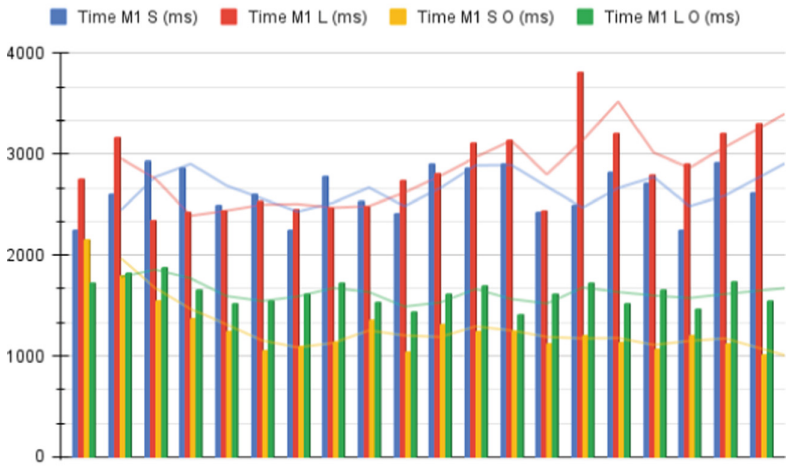
7 Discussion

Migrating applications to a web version might bring many benefits to both private and public companies. As presented in this paper, many of the advantages outweigh the disadvantages as internet connectivity is relatively common and internet speed improves from year to year.

Limitations in terms of hardware interaction can be overcome with the rapid development and evolution of browsers. Security should always be a priority when migrating to an online environment since internet connectivity might expose potential vulnerabilities inside organizations.



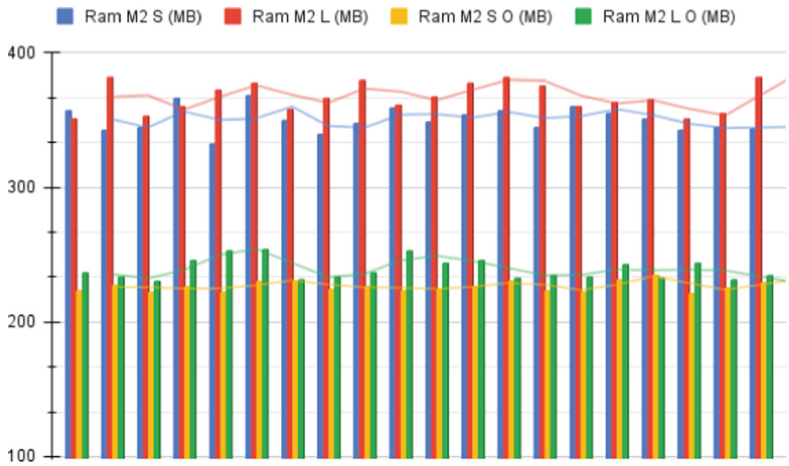
(a) Device 1 time comparison



(b) Device 2 time comparison

Fig. 5. Time spent till interactive. S: Small File, L: Large file, O: Web application, M1: Device 1, M2: Device 2

Removing the need to install an application and quickly accessing it using a link provides faster onboarding to both new customers and employees. Results show that the studied application performs better on the web version, most likely due to caching and better optimization. Nevertheless, this cannot be generalized for all applications, especially those requiring lots of computational and GPU power.



(a) Device 1 RAM comparison



(b) Device 2 RAM comparison

Fig. 6. RAM Consumption. S: Small File, L: Large file, O: Web application, M1: Device 1, M2: Device 2

Migrating applications to an online version might bring more benefits than advantages, but a thorough assessment needs to be made before deciding to migrate applications (e.g., Specialized applications for factories and 3d graphic design).

These challenges are an intriguing expansion for this study, and the authors want to broaden the results of the current work by expanding the research to include a more in-depth review of the migration implications.

7.1 Current Trends

Many companies choose to migrate their products online due to the fact that they can reach more clients with a simple registration and payment step rather than downloading and installing the product. This provides flexibility and ease of user access in a modern, user-friendly GUI in the web browser and offers better conversion rates.

An increasing number of companies choose to offer web-based applications nowadays rather than native ones, as developing and maintaining a native application can be cumbersome, especially from the manpower point of view. However, more developers specialize in web development rather than native.

Updates to the application functionality are done on the fly without the user's manual input, and subscription-based products can be easily implemented without the risk of the product being hacked (access is granted via username and password rather than through a CD-key).

The operating system is no longer considered an issue, as all user interaction is done using the browser. However, some applications may require more system resources to perform complex tasks.

Even though implementing PWA standards is straightforward and brings benefits by default, such applications are not marketed as such due to the lack of popularity.

8 Conclusions

As software development and the internet evolve, so does the need to keep up with the current trends. In the early times of software, interaction with the systems was trivial and mainly achieved through a command terminal. Recent trends rely on modern user interfaces with a simplified user experience as interaction with day-to-day applications shifts from desktop-installed to web applications.

In this paper, we have provided an overview of the previous, current, and future state of the development of applications, both native and web. We have studied the current state of development and preferred approaches. In addition, studied the trends and directions for the software development of applications.

Furthermore, the decentralization of the application and its implication has been studied and key points on performing this step have been taken into account and documented.

Shifting an application from desktop-installed to web poses many challenges, which have been outlined in the presented sections. Directions to overcome those challenges have been considered for every specific scenario.

A study has been carried out to compare the time spent on performing a similar task on both a desktop-installed and a web version of the same piece of software. The results show that the web application loaded faster and consumed fewer resources than the desktop version.

References

1. Biørn-Hansen, A., Majchrzak, T.A., Grønli, T.-M.: Progressive web apps for the unified development of mobile applications. In: Majchrzak, T.A., Traverso, P., Krempels, K.-H., Monfort, V. (eds.) WEBIST 2017. LNBIP, vol. 322, pp. 64–86. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-93527-0_4
2. Despa, M.L.: Comparative study on software development methodologies. Database Syst. J. **5**(3), 37–56 (2014)
3. Dingli, A., Seychell, D.: The New Digital Natives. JB Metzler, Stuttgart (2015)
4. Fortunato, D., Bernardino, J.: Progressive web apps: an alternative to the native mobile apps. In: 2018 13th Iberian Conference on Information Systems and Technologies (CISTI), pp. 1–6. IEEE (2018)
5. Georgiev, M., Jana, S., Shmatikov, V.: Breaking and fixing origin-based access control in hybrid web/mobile application frameworks. In: NDSS symposium, vol. 2014, p. 1. NIH Public Access (2014)
6. Gonsalves, M.: Evaluating the mobile development frameworks Apache Cordova and Flutter and their impact on the development process and application characteristics. California State University, Chico (2019)
7. Ickin, S., Petersen, K., Gonzalez-Huerta, J.: Why do users install and delete apps? A survey study. In: Ojala, A., Holmström Olsson, H., Werder, K. (eds.) ICSOB 2017. LNBIP, vol. 304, pp. 186–191. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-69191-6_13
8. Jazayeri, M.: Some trends in web application development. In: Future of Software Engineering (FOSE 2007), pp. 199–213. IEEE (2007)
9. Jobe, W.: Native apps vs. mobile web apps. Int. J. Interact. Mob. Technol. **7**(4) (2013)
10. Kotane, I., Znotina, D., Hushko, S.: Assessment of trends in the application of digital marketing. Sci. J. Polonia Univ. **33**(2), 28–35 (2019)
11. Kounev, S., Lange, K.D., von Kistowski, J.: Systems Benchmarking: For Scientists and Engineers, vol. 1. Springer, Heidelberg (2020)
12. Lim, S.H.: Experimental comparison of hybrid and native applications for mobile systems. Int. J. Multimed. Ubiquit. Eng. **10**(3), 1–12 (2015)
13. Malavolta, I., Ruberto, S., Soru, T., Terragni, V.: Hybrid mobile apps in the Google play store: an exploratory investigation. In: 2015 2nd ACM International Conference on Mobile Software Engineering and Systems, pp. 56–59. IEEE (2015)
14. Montelius, A.: An exploratory study of micro frontends. Master's thesis, Linköping University, Software and Systems (2021)
15. Panda, S.K., Satapathy, S.C.: An investigation into smart contract deployment on Ethereum platform using Web3.js and solidity using blockchain. In: Bhateja, V., Satapathy, S.C., Travieso-González, C.M., Aradhya, V.N.M. (eds.) Data Engineering and Intelligent Computing. AISC, vol. 1407, pp. 549–561. Springer, Singapore (2021). https://doi.org/10.1007/978-981-16-0171-2_52
16. Petcu, A., Pahontu, B., Frunzete, M., Stoichescu, D.A.: A secure and decentralized authentication mechanism based on Web 3.0 and Ethereum blockchain technology. Appl. Sci. **13**(4), 2231 (2023)
17. Qu, C., Calheiros, R.N., Buyya, R.: Auto-scaling web applications in clouds: a taxonomy and survey. ACM Comput. Surv. (CSUR) **51**(4), 1–33 (2018)
18. Selvarajah, K., Craven, M.P., Massey, A., Crowe, J., Vedhara, K., Raine-Fenning, N.: Native apps versus web apps: which is best for healthcare applications? In: Kurosu, M. (ed.) HCI 2013. LNCS, vol. 8005, pp. 189–196. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-39262-7_22

19. Tandel, S., Jamadar, A.: Impact of progressive web apps on web app development. Int. J. Innov. Res. Sci. Eng. Technol. **7**(9), 9439–9444 (2018)
20. Torchiano, M., Ricca, F., Marchetto, A.: Are web applications more defect-prone than desktop applications? Int. J. Softw. Tools Technol. Transfer **13**, 151–166 (2011)