UNIVERSIDADE DE LISBOA
FACULDADE DE CIÊNCIAS
DEPARTAMENTO DE INFORMÁTICA

GITUI: A COMMUNITY-BASED UI ADAPTATION REPOSITORY

Ricardo Alexandre Rosado Costa

Mestrado em Engenharia Informática
Especialização em Sistemas de Informação

Trabalho de Projeto orientado por:
Prof. Doutor Tiago João Vieira Guerreiro

2022

Agradecimentos

Este trabalho marca o final de um percurso bastante importante, que incluiu vários desafios que apenas puderam ser superados devido às pessoas que me apoiaram nas situações mais difíceis.

Por este motivo quero agradecer aos meus pais e à minha namorada por sempre me terem motivado e apoiado ao longo deste percurso académico e pessoal, e também à minha restante família e amigos pelas palavras de encorajamento.

Obrigado também à equipa do DI-LaSIGE e ao grupo Tech & People Lab por toda a ajuda e também pelos novos desafios que, embora exigentes, contribuíram para o meu desenvolvimento profissional e também pessoal.

Aproveito também para agradecer a todos aqueles que participaram nos estudos efetuados ao longo deste trabalho, tornando possível a concretização deste trabalho.

Obrigado por tudo!

Aos meus pais.

Resumo

Diversos elementos das nossas vidas foram democratizados (e.g., as redes sociais democratizaram o acesso e partilha de informação; as impressoras 3D permitiram a qualquer indivíduo criar e partilhar modelos de novas interfaces físicas).

No entanto, existe um elemento cujo os utilizadores não detêm qualquer controlo: a Interface Utilizador. Atualmente, os designers realizam estudos para perceber qual a interface que satisfaz a maioria dos utilizadores. No entanto, é impossível criar uma solução que satisfaça uma comunidade inteira, uma vez que cada utilizador apresenta necessidades e preferências distintas.

Face ao exposto, a presente dissertação procura resolver este problema permitindo a todos os utilizadores alterar e partilhar interfaces web adaptadas com a comunidade, transferindo o controlo da adaptação de interfaces web dos *developers* para os próprios utilizadores. De modo a alcançar este objetivo, realizámos uma análise da literatura existente acerca desta temática de maneira a definir a melhor abordagem a seguir para solucionar o problema previamente descrito. Tendo por base a informação recolhida através do estado da arte, desenvolvemos uma plataforma que permite aos utilizadores criar e guardar alterações nas interfaces. Esta plataforma engloba também a possibilidade de procurar e partilhar estas mesmas interfaces customizadas, oferecendo assim, a cada utilizador, a possibilidade de importar adaptações já existentes no repositório, e consequentemente substituir qualquer interface web por uma versão adaptada da mesma que satisfaça todas as suas necessidades e interesses específicos.

Para além destes benefícios que a nossa plataforma oferece à comunidade, os proprietários de websites também poderão retirar proveito de forma indireta ao terem ter acesso a informação acerca das alterações mais comuns/recorrentes efetuadas por parte dos utilizadores, que poderão ser úteis no desenvolvimento das interfaces dos seus websites, tornando estes mais robustos e inclusivos.

A plataforma é composta por dois componentes diferentes: a ferramenta de edição e um repositório web. A ferramenta de edição, desenvolvida como uma extensão para o Google Chrome, possibilita aos utilizadores customizar a interface de qualquer página web através de diferentes operações (e.g., mover e ocultar elementos, alterar a cor e o tamanho do texto, aumentar ou diminuir margens) que estarão disponíveis através de um clique e, para utilizadores mais experientes, injeção de código JavaScript e/ou CSS (*Cas-*

cading Style Sheets). O repositório web, implementado como uma aplicação web, é responsável pela partilha das adaptações criadas através da ferramenta de edição, e também pela interação entre os utilizadores e a comunidade.

De maneira a tirar partido da comunidade, caso o repositório não possua uma adaptação que satisfaça totalmente as necessidades do utilizador, a nossa plataforma confere concomitantemente a possibilidade de importar uma adaptação do repositório e otimizá-la utilizando a ferramenta de edição, de maneira a adicionar apenas os aspetos que estavam em falta na adaptação original, mantendo o conteúdo desenvolvido pelo criador da adaptação.

O repositório web, para além de ser responsável pela partilha das interfaces criadas pelos utilizadores, também permite a comunicação entre estes. Através do nosso sistema de pedidos, os utilizadores podem fazer solicitações de customização para um website em específico, demonstrando o que pretendem através de uma descrição e/ou uma imagem providenciadas. Estes pedidos serão posteriormente publicados no nosso repositório web, com o intuito de serem vistos e analisados por outros utilizadores que poderão adotar a iniciativa de responder submetendo uma adaptação que satisfaça o pedido.

Transversalmente, os utilizadores podem também avaliar e comentar as adaptações presentes no repositório web de maneira a valorizar ou criticar o trabalho da comunidade. Com isto pretende-se que os utilizadores que respondem aos pedidos da comunidade sejam valorizados pelo seu esforço e, consequentemente, se sintam motivados a continuar a interagir com a comunidade, visto que as avaliações recebidas e outras estatísticas estarão presentes na sua página de perfil, que está visível para a comunidade.

Para avaliar a usabilidade e aceitação da plataforma, foram efetuados dois estudos qualitativos com recurso a entrevistas. O primeiro estudo centrou-se na componente de edição de interfaces web, tendo como objetivos avaliar o processo de edição da ferramenta, ou seja, compreender a experiência de cada participante e a sua opinião relativamente às diferentes funcionalidades da ferramenta, bem como identificar o grau de interesse por parte dos utilizadores em utilizar a nossa plataforma.

Este estudo foi realizado por nove participantes e consistiu em três fases: uma primeira entrevista onde é solicitado aos participantes que completem um conjunto de exercícios; um segundo período onde estes devem utilizar a ferramenta em casa sozinhos, e uma entrevista final onde os participantes devem relatar a sua experiência.

O segundo estudo, centrado na aplicação web, focou-se em avaliar a usabilidade da aplicação web, no sentido de detetar alguns aspetos que possam ser melhorados (e.g., a intuitividade da aplicação e a fluência das suas diversas funcionalidades).

Este segundo estudo consistiu numa entrevista onde os participantes completaram uma série de exercícios com o propósito de os guiar pelas diversas funcionalidades da aplicação web, de forma a entender a sua opinião relativa à mesma.

Tendo em consideração os resultados obtidos em ambos os estudos, tornou-se possível concluir que o nível de aceitação da nossa plataforma é consideravelmente elevado. No

que toca à usabilidade da plataforma, os utilizadores foram capazes de concluir as tarefas que lhes foram atribuídas durante a participação nos estudos e demonstraram facilidade em utilizar a plataforma, sendo que o processo de habituação foi relativamente fácil para a maioria dos participantes, que utilizaram a nossa plataforma em casa de forma autónoma durante duas semanas após a primeira entrevista.

Para além disso, os participantes partilharam sugestões que, para eles, poderiam contribuir para aprimorar a plataforma, o que é uma mais valia para o desenvolvimento futuro.

Palavras-chave: Interfaces de utilizador, Acessibilidade, Contribuição colaborativa, Adaptação, Aplicação Web

Abstract

Several aspects of our lives have been democratized (e.g., social networks democratized access and delivery of information, and 3D printers enabled anyone to create and share models for new physical interfaces).

Conversely, one crucial aspect remains out of bounds to individual agency: The User Interface. Today, designers perform studies in order to understand the interface that satisfies the majority of the users. Unfortunately, it is impossible to create a solution that satisfies the whole community, as users have different necessities and preferences.

This thesis aims to solve this issue, allowing users of web interfaces to tweak and share adaptations with other users, shifting the agency of UI adaptation from developers to users. To achieve this, a prototype that allows users to create and save changes to web interfaces was developed. This prototype connects to a platform that allows other users of the community to search and share customized web interfaces.

Using this platform, users can replace any UI with an adapted version that satisfies their necessities.

Our platform aims to benefit not only the community, giving the users access to a collection of UI adaptations where they could browse for the ones that match their necessities, but also the website owners, informing them of what could be improved.

This platform benefits the community since it provides the users with a variety of UI adaptations that should match all their necessities; but it also benefits website owners, reducing their burden to provide accessible web content, since the community can create their accessible version of any web page.

Keywords: User Interfaces, Accessibility, Crowdsourcing, Adaptation, Web Application

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Internet access is fundamental to exercising one's human rights[4], as it is tied to a set of human capabilities that are considered fundamental to a life worth living. However, overly complex interfaces, lack of flexibility, and the inability to transform content presentation all prevent the effective use of the Internet.

Nowadays, User Interfaces (UIs) follow a "one size fits all" approach. This means that website owners develop their interfaces targeting a average end-user, which might not be a problem for most of the population, but users who are less tech-savvy or have some disability (e.g., visual impairments, movement disorders, etc.) may have some issues using an interface without some kind of assistance.

1.1 Motivation

Unlike several aspects of our lives, User Interfaces remains out of bounds to individual agency. Today, we are still restricted to the interfaces created by the developers/designers which are developed following the accessibility guidelines. However, there is no solution that satisfies all users' interests/necessities.

A proposed approach is to address this problem by developing a solution that automatically adapts an interface based on user modelling (e.g., [10, 7, 14]). However, this approach can sometimes be confusing for the users because, as the user model changes, the interface presented to them also changes, resulting in a different interface being presented each time a user opens a web page.

For this reason, a system that enables users to interact with the community, giving them the option to search and share adaptations based on their preferences might be more beneficial and less confusing.

With the existence of a platform that allows searching and sharing customized interfaces, users are provided with a variety of interface adaptations of any website (created by other users of the community), so that they can select the version that better suits their needs. This implies an instant delivery of a new version of the interface, that otherwise would take a long time to be made available. However, some articles that already implemented similar platforms (e.g., [11]) have faced a "cold-start problem", meaning that in a starting phase, the platform is "empty" because no users have already shared any adaptations.

The "cold-start problem" makes it necessary for any user to be able to create their own changes to the interface, even with no programming skills. Nebeling et al. [12] and Nebeling and Norrie [11] developed a visual editing tool that, with simple operations, provided very promising results.

In short, in order to promote equality in the Internet, we developed an approach that aims to shift the agency of UI adaptation from developers to users. Instead of looking for approaches that automatically adapts interfaces based on user modelling, in this thesis i present a platform that allows users to interact with the community, sharing adaptations that might be useful for others, giving them the choice to be presented with an interface that better fits their needs.

1.2 Objectives

The objective of this thesis is to allow users of interfaces to tweak and share interface adaptations with other users, shifting the agency of UI adaptation from developers to users.

To achieve this, we developed a platform that allows users to create customized interfaces that can be shared with the community.

To evaluate our approach, we want to address the following questions:

1. How do people respond to a UI personalization tool?
2. Do users have the skills and interest in personalizing an interface for themselves? Do they prefer to ask for help?
3. Can expert users personalize for others? What motivates them (e.g., gamification, gratitude, ...)? What factors do they consider?

1.3 Contributions

The main contributions of this dissertation are the following:

- A review of the state of the art on UI personalization and crowsourced UIs.
- A personalization tool that allows people to edit web pages and make those changes persistent.
- A platform for UI sharing that allows users to create editions and share them with others, democratizing UI design.
- Two studies that, among other objectives, evaluate the usability and acceptance of our platform, identifying its strengths and points of improvement.

1.4 Document's Structure

This document is structured in 6 chapters:

- **Chapter 2 - Related Work:** This chapter presents the research made in order to understand the state of the art relative to the area of this dissertation and the solutions that address it.
- **Chapter 3 - System Design:** Presents use case scenarios, the proposed approach and functional and non-functional requirements.
- **Chapter 4 - Implementation:** Presents the system architecture, the technologies and data structure used for its development, and a description of its functionalities.
- **Chapter 5 - User Studies:** Presents the studies performed to evaluate our solution, as well as a discussion regarding the results obtained.
- **Chapter 6 - Conclusion:** This chapter presents the conclusions drawn from this dissertation, the limitations of our solution, and the future work.

Chapter 2

Related Work

2.1 User Interface Adaptation

According to Karger and Quan [9], interfaces that support customization and are able to adapt to individuals' specific use patterns have the potential to be more beneficial than ones designed to be "one size fits all".

The customization process can either be automated or user-driver. A system with an automated customization process use information gathered about the users to automatically adapt any interface to their necessities. On the other hand, systems with user-driven UI customization rely on the users to create their own interfaces using a provided toolkit.

In this chapter we analyze these two approaches of UI customization, and the importance of the user model in each one.

2.1.1 Automatically generated UI adaptations

Approaches that automatically generate UI adaptations tend to use a flexible user model. The user model is created using different methods to gather information about the user, such as questionnaires to gather information about the user's interests/capabilities. However, in order to make the user model flexible, it needs to be able to adapt based on each interaction the user makes with a web page.

A good example of this type of approach is UIFlex[14], a web-based tool that automatically adapts interfaces based on the interaction profile of each user. This means that each time the user interacts with the interface, its profile gets updated.

Based on this profile, the tool will then change elements of the user interface (e.g., changing color maps, highlighting the navigation focus, activating video subtitles automatically) the next time the user visits a web page.

However, flexible systems are generally less stable for the user. This means that, in some situations, users will be presented with a different interface each time they visit the same web page, due to the fact that the user model keeps changing each time they interact

with a web page, which might be a huge downside since it may confuse the users.

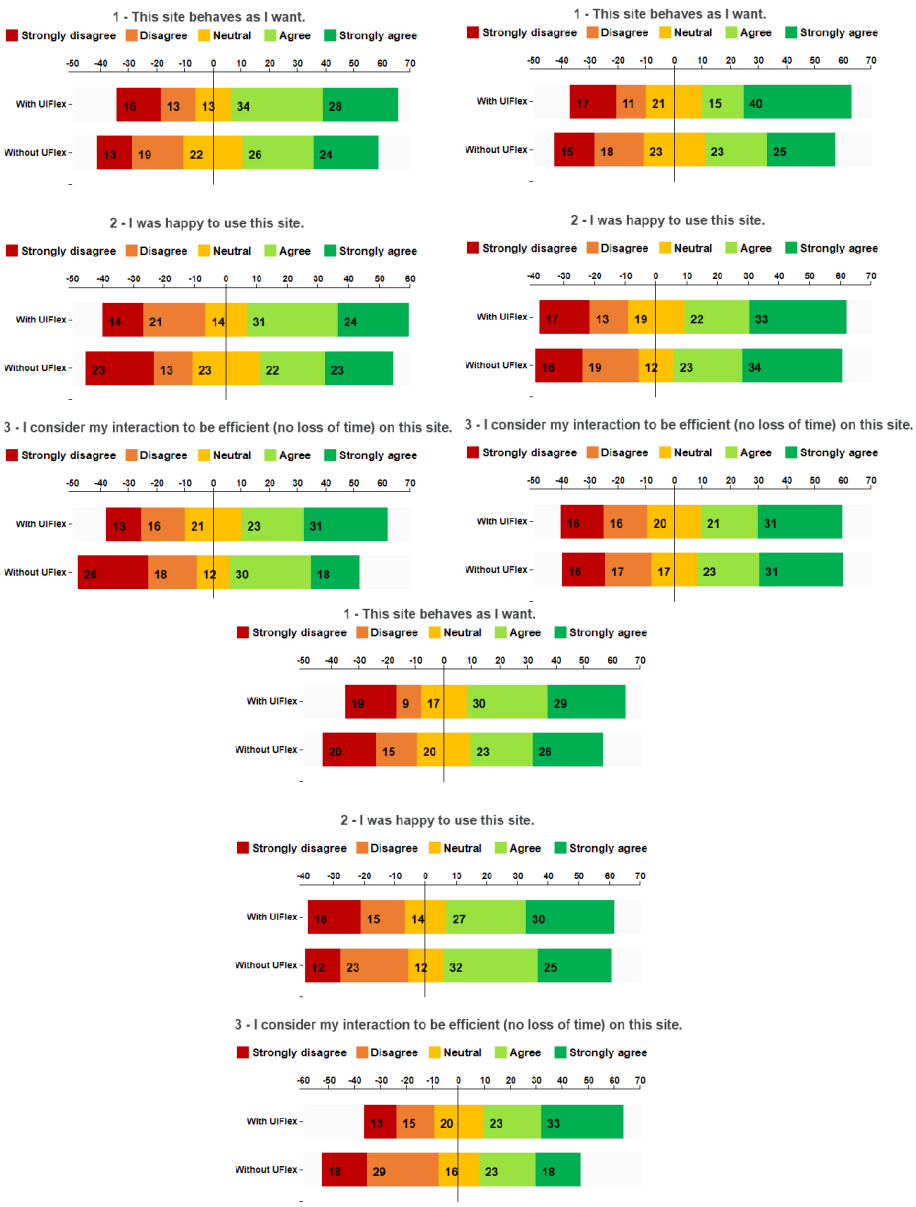


Figure 2.1: UIFlex[14] results on 3 different websites.

The results of UIFlex detected some usability improvements when the tool is used. As it is shown in figure 2.1, when UIFlex is being used, the websites have a better behaviour, users have a better experience using the websites, and the time lost from the user's interaction with the website is reduced.

The authors stated that, if a website has not been developed following the accessibility guidelines of the W3C[1], some customization may not work properly. However, during the

[1]https://www.w3.org/TR/wcag-3.0/

evaluating done by users, they could understand that the users should be able to customize the interface themselves according to their preference and/or needs.

A similar approach is SUPPLE[7], a system that automatically generates interfaces but, in addition to user behavior, it also takes into account device characteristics and the available widgets. The optimization process is based on two types of cost functions:

- The first one (preference-based personalization) is based on a user's subjective preferences. It takes into account different choices of parameters (e.g., minimum target size, minimum visual cue size), resulting in different styles being applied on the interfaces generated.
- The second (ability-based personalization) reflects the expected time a person would need to perform a typical set of tasks with a particular user interface, capturing its motor abilities.

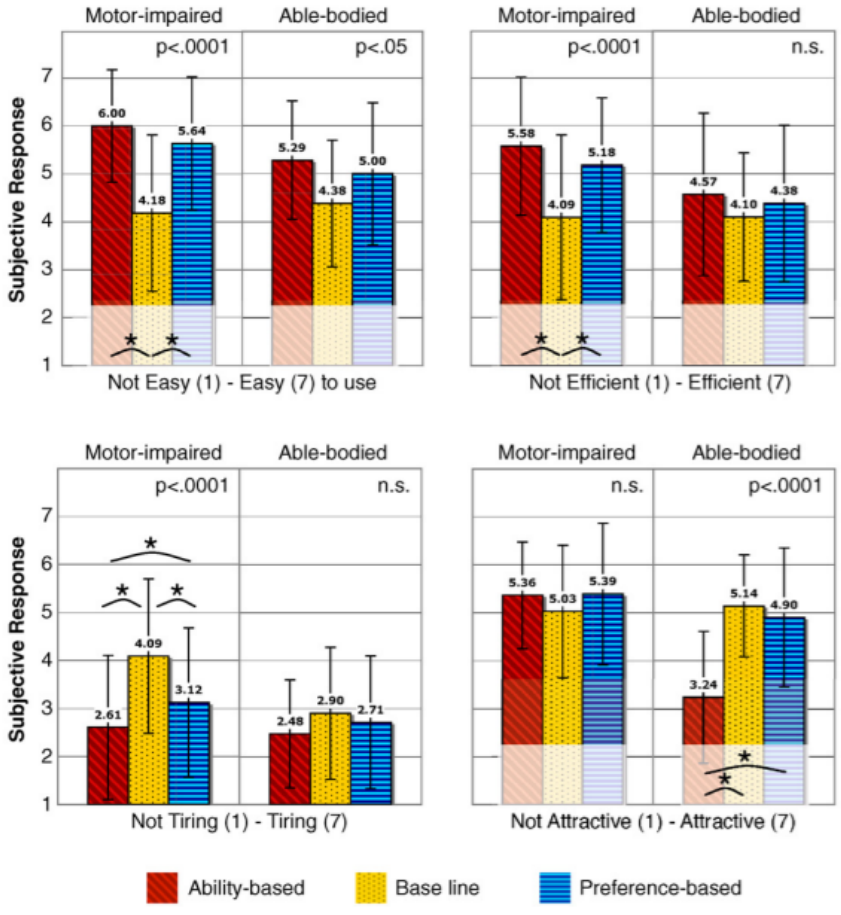


Figure 2.2: SUPPLE[7] - Results

The results were slightly more promising than those provided by UIFlex[14]. Figure 2.2 shows that either type of personalization will result in an interface that is easier, more

efficient and less tiring than the original version.

Unlike UIFlex, SUPPLE was not developed to improve existing design processes, its goal is to provide motor impaired people with interfaces that are easier for them to interact with when compared to the manually designed default interfaces.

Kumar et al. [10] presented Bricolage, an approach that, unlike the ones presented in the previous articles, does not rely on user modelling/behavior.

Bricolage is an algorithm for transferring design and content between Web pages. It allows matching visually and semantically similar elements in pages to create coherent mappings between them. These mappings can then be used to automatically transfer the content from one page into the style and layout of the other.

Figure 2.3: Example of Bricolage[10]'s mapping algorithm.

Bricolage uses structured prediction [5] to learn how to transfer content between

pages. It trains a corpus of human-generated mappings, collected using a Web-based crowdsourcing interface that was seeded with 50 popular Web pages that were decomposed into a visual hierarchy by a novel, constraint-based page segmentation algorithm, Bento.

Bento is a page segmentation algorithm that uses the page's DOM tree as a starting point to segment each page into a hierarchy of salient regions that can be extracted and manipulated.

This approach does not have the ability to adapt to the individual needs of each user. However, it can provide him with a more friendly/simpler version of the original interface.

One thing to note is that there is no guarantee that every web page can fit into the layout and style of one of the web pages in the collection.

These approaches are proven to be beneficial and efficient, meaning that with a reduced amount of effort, users are able to obtain an interface that is automatically adapted to satisfy their interests/necessities. However, using these approaches, the user is does not have much control on the customization process.

For this reason, we also considered to explore user-driven UI adaptation methods, which are more time consuming and require some effort from the users, but gives them full control of the customization process.

2.1.2 User-Driven UI adaptations

When it comes to User-Driven UI adaptations, there is no need for a flexible user-model.

Nebeling et al. [12] and Nebeling and Norrie [11] implement this type of adaptation based on a toolkit that relies on 7 types of adaptation operations to make effective use of both small and large-screen settings. Each operation is responsible for a change in the interface, and the new interface is the result of all the operations applied to the original interface (see figure 2.4).

The changes made by the users are then stored in a server in the form of a new layout template and can be then automatically downloaded and applied in their subsequent visits.

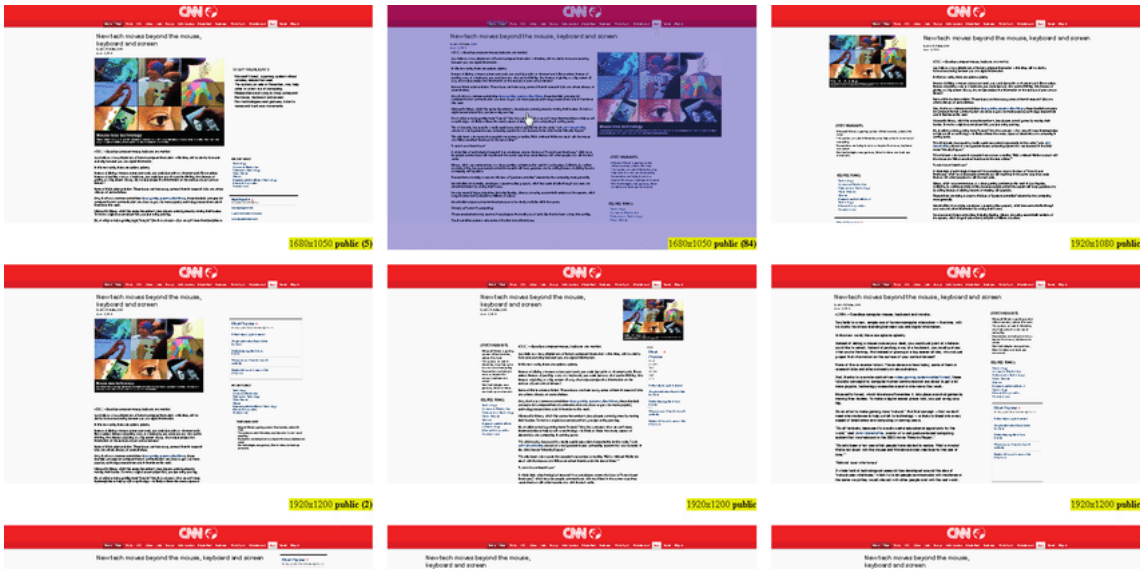


Figure 2.4: Example of adaptations created by CrowdAdapt's [12] toolkit.

The toolkits developed by Nebeling et al. [12] and Nebeling and Norrie [11] are very similar. The difference is in the structure of the modifications.

In Nebeling and Norrie [11]'s solution, all adaptations are essentially modifications of the CSS that can additionally be downloaded for the original website, making this process very lightweight, since no additional versions of the HTML document must be maintained, and also website generation is not required. Since CSS definitions can be cascaded, the layout of elements can be adjusted in multiple steps by building from previous definitions. This is important for the users because it means that the adaptations do not have to be defined from scratch, but can be based on other users' contributions, requiring less effort of the end-user. However, there are limitations to what the toolkit can do with only CSS modifications.

Crowdadapt [12], on the other hand, manipulates both CSS and/or HTML DOM, making it a more powerful toolkit, but the process of importing the adaptations is also heavier because it implies a new website generation.

The operations used by Nebeling et al. [12] and Nebeling and Norrie [11] are the following:

- **Move:** Repositions web page elements in the document via drag-and-drop.
- **Resize:** Scales elements horizontally and/or vertically by dragging the edges or corners as known from common window managers.
- **Spacer:** Increases or decreases the space around an element.
- **Hide:** Toggles the visibility of an element.
- **Collapse:** Replaces an element with a placeholder link to later unfold the content.

- **Font-size:** Increases or decreases the text height.
- **Multi-column:** Controls the number of columns used for content layout.

Akiki *et at.* [1] also presented a system that allows users to manipulate UIs by minimizing its feature-set. This system is based on a previous one developed by the same authors called Role-Based User Interface Simplification (RBUIS)[2]. RBUIS minimizes the feature-set of a UI by assigning roles to tasks in task models, achieving a multi-layer user interface design [15].

In RBUIS, enterprise administrators are assigned roles to create adaptations, and end-users are given the ability to provide feedback on these adaptations.

The new version of RBUIS allows end-users to perform adaptations through a web-based feature-set editing tool, complementing the former system from the following perspectives:

- Using a simple tool (figure 2.5), end-users can adapt the feature-set without attaching the adaptations to user roles. Afterward administrators could attach the UI adapted by the crowd to one or more enterprise roles.
- The proposed technique is potentially helpful with non-role-based enterprise tools (e.g., word processors, spreadsheet managers, etc.) where the user could apply one of the crowd-adapted user interfaces based on a given context.

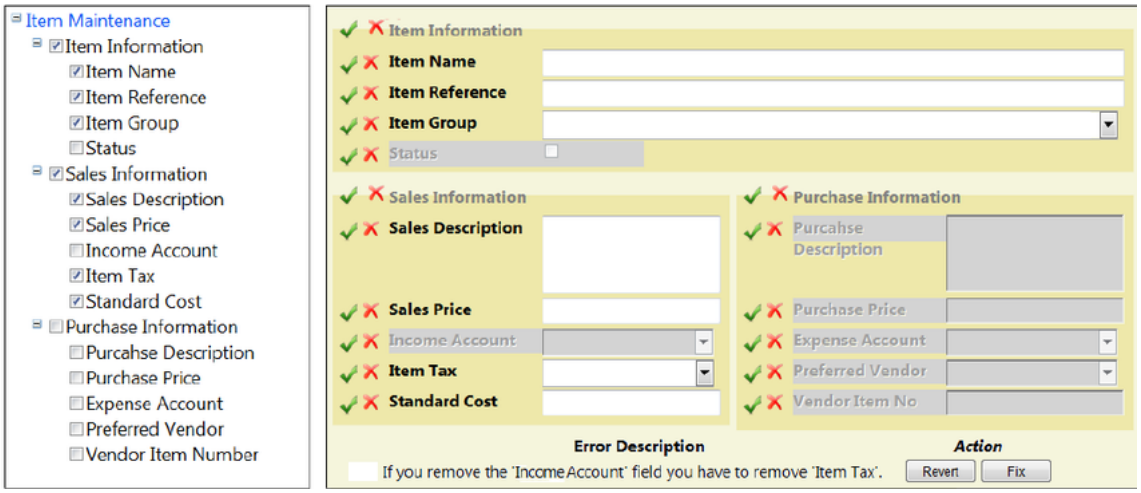


Figure 2.5: Feature-set editing tool developed by Akiki et al. [1]

The feature-set editing tool can be used by both experts and non-experts. The user can choose what features to include/exclude, and it executes an automatic verification to check whether the adaptation creates any conflicts (as it can be seen in the bottom section of the image above).

Resutls from a study showed encouraging results in terms of perceived usability, measured efficiency, and effectiveness. However, it does not support the adaptation of concrete UI widget properties (size, location, etc.), which might be a very important feature for most end-users.

2.2 Collaborative/Crowdsourced User Interfaces

This section describes different approaches used to create a system that takes advantage of the community to present users with an interface that suits all their needs.

The approach developed by Takagi et al. [16] achieves this by creating a system where users can submit a request addressing the problems that they want to see improved in an interface. Other users (called supporters by the authors) can then volunteer to renovate the UI based on these requests.

The system is divided into three major components: scripts for end-users, an authoring tool for supporters, and a server-side repository with services, as it is shown in figure 2.6.

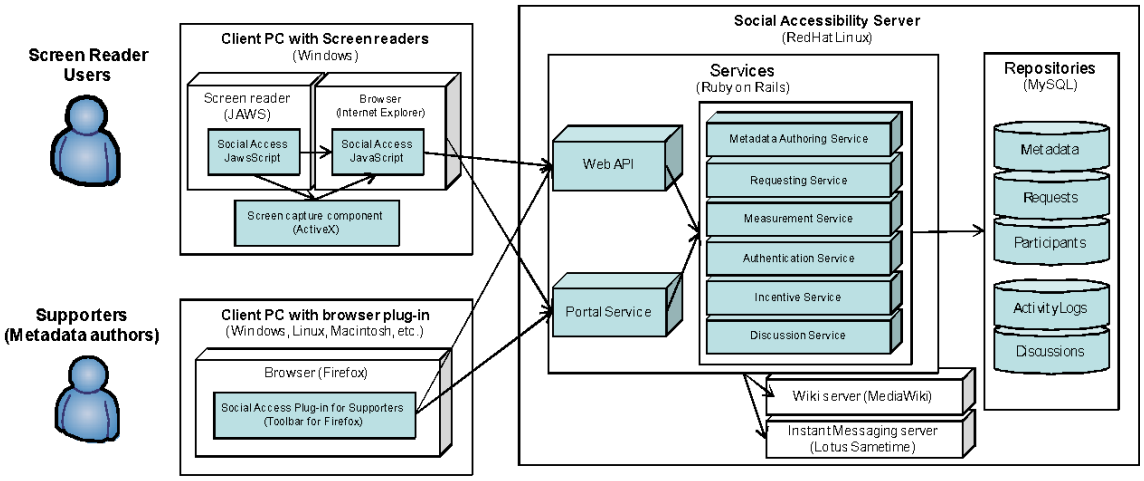


Figure 5. Architecture of the Pilot Social Accessibility System

Figure 2.6: Architecture of the Pilot Social Accessibility System, developed by Takagi et al. [16].

Using the authoring tool, supporters are able to add metadata to HTML elements in a web page. The new version of the web page (including the metadata created by the supporters) is then stored in the repository. To access this data, the end-users should connect to the repository using the browser and import the data that they find useful. After importing the new version of the web page, all of the metadata is automatically added to the page when the user visits the page again.

This approach relies heavily on an active community, since every request has to be addressed individually by a supporter, which can be a downside. If the number of active

supporters is much lower than the rate that requests are submitted, each request can take a long time to be solved.

The authors implemented an incentive mechanism that gives points to both end-users and supporters, which can be useful to motivate them to be active in the social computing system.

The adaptation system developed by Akiki et al. [1], which was already described in the section 2.1.2, applied a different process for crowdsourcing the adaptation of enterprise application UIs.

Figure 2.7 describes the process implemented in this system. Both users and experts (internal enterprise staff members) are able to customize and verify a UI's feature set. Before the new UI is made available to the community, the administrator needs to check, integrate and publish it internally. After it is published, all users gain access to the adapted UI and are now able to use it and rate it. Finally, the administrator shares the internal UIs and rating with the external community.

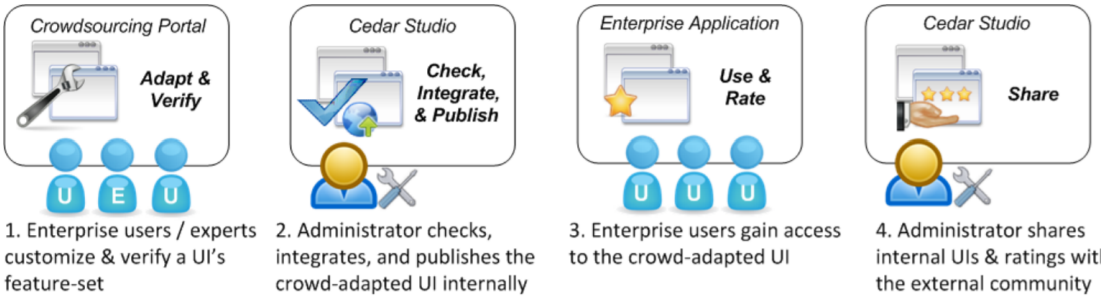


Figure 2.7: Process for Crowdsourcing Enterprise Application User Interface Adaptations used in Akiki et al. [1]'s approach.

Using this approach, both users and experts can customize and verify a UI's feature set, but only an administrator can integrate and publish the crowd-adapted UI, so there is always a need for an internal staff member to participate in the adaptation process.

Both the approaches of [16] and [1] imply a role separation. The first implies a separation between end-users and supporters, while the subsequent implies a separation between end-users and administrators.

This type of separation limits the power given to each user. In Takagi et al. [16]'s approach users can only submit requests and wait for supporters to solve them. Akiki et al. [1]'s approach allows users to customize and verify the feature set of a UI, however only an administrator can integrate and publish these changes.

The solution developed by Nebeling and Norrie [11] provides an interesting approach for crowdsourcing UIs that also solve this role separation problem.

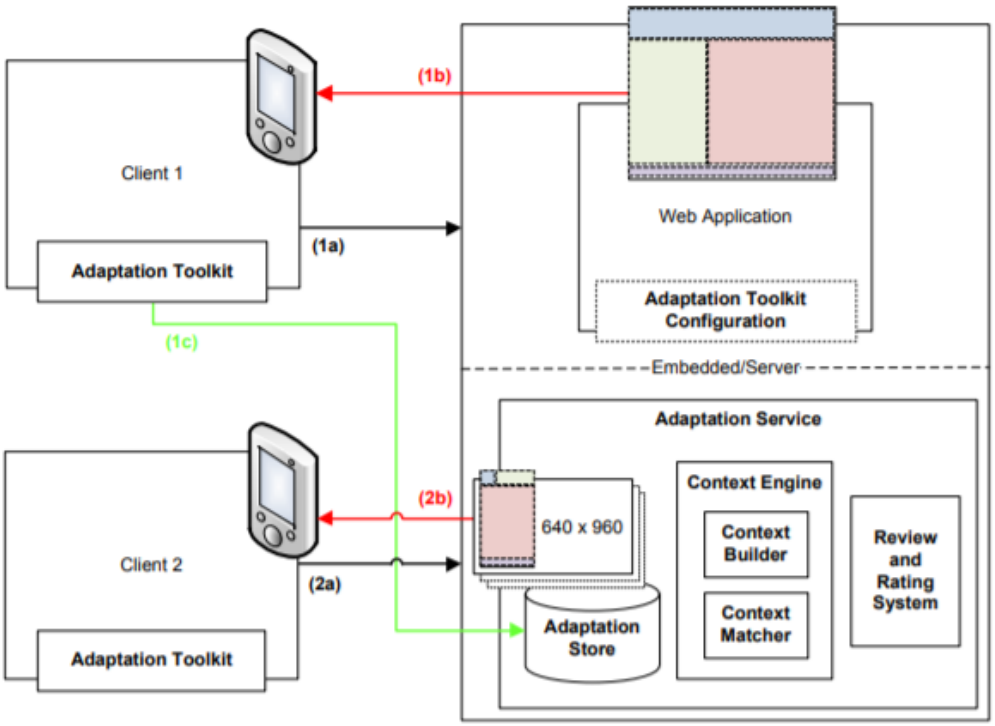


Figure 2.8: Nebeling and Norrie [11]'s conceptual model of crowdsourcing architecture also showing the definition and deployment processes carried out by one client to the benefit of others

Figure 2.8 shows the client/server architecture behind a tyipical web application as well as processes (1) and (2) initiated by two separate clients. Both users have the adaptation toolkit (mentioned in the section 2.1.2) installed and running on the browser.

The adaptation service, located on the server-side, comprises the following three components:

1. The adaptation store, which is an interface to an underlying database designed for the storage and retrieval of website adaptations.

2. The context engine, which functions as a recommender system to rank and select the best-matching adaptations.

3. The review and rating system for users to take influence on the recommendations made by the system.

The image also provides an example of how the architecture enables crowdsourced web adaptation, showing two user using a mobile phone for accessing an example site that has initially been created with a standard resolution of 1024x768 in mind.

The first user (Client 1) starts by opens the web application, receiving the original interface in its default layout. However, the original interface of the web application is too large to fit into the user's mobile phone. Using Nebeling and Norrie [11]'s toolkit,

the user was able to adapt the interface to fit the device resolution of 640x960. The user's toolkit automatically synchronizes with the adaptation store where it maintains a record with the defined adaptations for the each user, allowing the user to be presented with the adapted version of the interface in the subsequent visits.

When the second user (Client 2) access the web application using the toolkit, recommended adaptations will be downloaded and applied, so that the end-user is automatically presented with the adapted mobile version that was created and shared by the first user.

A platform like this, that is more open towards the community, might bring more benefits for its, discarding the need to submit a request every time a user wants to improve something in the UI. Thus, every user can open the platform and choose what version of the interface they like the most.

Also, since every user can share their adaptations, the adaptation store will provide a vast variety of adaptations, presenting a solution for a large amount problems. However, even if there is no perfect solution for specific user, he/she can import the version that attempts to solve most of his/her needs, and optimize it using the adaptation toolkit.

Social Cheatsheet[17] complements the approach of Nebeling and Norrie [11]. Social CheatSheet is a novel platform that overlays relevant community-curated instructions and multi-step tutorials atop any web application and offers an easy curation interface for adding and editing content.

Instead of focusing on customizing the interface, Social Cheatsheet's approach allows users to create task-focused curated instructions and multi-step tutorials using a combination of screenshots and snippets of web-based help resources. It also allows users to use and edit other users' curated help on any web application, bypassing the potential barrier of relying on application owners to integrate the community help system.

Sometimes adapting the UI's layout and style might not be enough for users to have the best experience when using a website. Vermette et al. [17]'s approach can provide instructions and tutorials to the users, further optimizing their experience with any website, especially when combined with UI layout and style adaptations.

According to the results obtained by the authors (the evaluation was a short-term small-scale deployment), most of the users (11/15) found the platform easy to use. Users who would otherwise use external tools and applications for saving and sending help to others said that, with Social CheatSheet, there is no longer the need to switch between other applications. In addition, the results showed that Social CheatSheet was useful in finding relevant curated content when compared to the traditional forms of help, such as documentation created by application owners.

Sharing new versions of an interface is a key feature for the purpose of this thesis.

However, some users might want to solely import small changes that only affects a small group of elements, instead of a whole new version of the interface. The platform developed by Garbett et al. [8] allows users to propose and promote ideas in response to community needs, and collaboratively design the concept through a series of customizable features.

Throughout this system, users can discuss their ideas for the overall design and specific design tasks.

These campaigns are divided into three phases:

- **Support Phase:** During the support phase, users are encouraged to create and submit their ideas. Also, there are discussion forums where users can discuss and vote on other users' comments.

- **Design Phase:** The design phase consists of two components; an interface showing the existing contributions, and a submission interface where users can still contribute their ideas. In this phase, users can vote up and down on the existing ideas.

- **Build Phase:** This is the phase where the app is developed and submitted to the app store. During this process, supporters are presented with a launch status indicator that provides feedback on the current status of the movement (building the app, submitting to the app store, available to download, etc.).

For the purpose of this thesis, the design phase is relevant to analyse. Although this platform is designed for users to collaborate on ideas that are still being developed, if implemented on an web page that already exists, it can highlight some aspects of the interface that can be improved. Looking at this list of suggestions presented by the community, other users can then address them and share their solutions for other users to add to their interfaces.

2.3 Summary

Looking at the information gathered in this research, we can identify two distinct types of UI adaptation (Automatic vs User-Driven), each one with its benefits and downsides.

Automatically generated UI adaptations does not require any work from the user in the process of adaptation, and the adapted version of the UI is built and delivered in seconds. For these type of approaches, the user model is very important since it needs to be flexible, taking into account each interaction the user makes with the interface, in order to adapt the UI according to his/her behavior. However, since the user model is constantly changing, when opening the same page many times, users can be presented with a different UI each time, confusing them. Also, as it is mentioned in the Proença et al. [14] solution, end-users felt the need to further adapt the UI provided by the system.

Bricolage [10], that allows transferring content from one page to another, doesn't rely on user modelling. However, implementing a similar process is not only expensive in both time and resources, but also it does not ensures that every web page can fit into the style and layout of one of the web pages in the collection.

On the other side, User-Driven UI adaptations take more time to be developed as the work needs to be done all by the user, but the end-result is guaranteed to be optimized for each user.

The concern with this type of adaptation consists in the process of website maintenance. In order to save the adaptations created by the user, there needs to be a way to identify the DOM elements that were customized (e.g., full path from root, element identifiers, etc.). When a new version of the website is launched into production, the DOM structure of the web page might change. Due to these changes, the platform might not be able to fetch the desired DOM element, resulting in some adaptations not being applied.

Despite this downside, User-Driven UI adaptations might be the better approach, as it ensures that the final adaptation is fully optimized for each user, and also, some solution can be developed to make these adaptations more resilient to website maintenance.

The articles [12, 11] provided a very interesting toolkit for users to customize their interfaces without needing any programming skills. Through simple operations, users can create changes to their interface, resulting in an optimized version of the interface for them.

Regarding the social aspect of this project (the web platform), the big question relies in the existence of role separation. With the existence of role separation, it is expected that the adaptations shared with the community are more reliable, since they were developed by experienced users in response to a request (e.g., [16]), or checked by administrators (e.g., [1]).

However, there are a lot of limitations. A higher number of requests will result in a longer response time. The same goes for the second case; if administrators have to check a large amount of adaptations before choosing which ones to integrate and publish, the same effect will occur.

For this reason, a platform without role separation might be a better approach for our system, enabling every user to share their adaptations with the community (similar to Nebeling and Norrie [11]'s approach).

The downside of a platform without role separation is the existence of adaptations that might be worthless for most of the community, and each user wants to be presented with the best options available. To solve this problem, there is a need to implement a ranking system in order to place the most popular/most used adaptations on top of the others, making it easy for each user to choose the best options to integrate.

Currently, there are already other tools that implement this type of approach. Tampermonkey[2] allows users to search and create scripts that can be injected into a web page. However, users that are not tech-savvy are not able to create their own scripts. On the other hand, Tampermonkey has a large community, giving users that are not tech-savvy the possibility to search for a variety of already existing solutions that might appeal to them.

[2]https://www.tampermonkey.net/

Chapter 3

System Design

The research presented in chapter 2 allowed us to have a better understanding of the state of the art regarding the theme of this thesis. From it, we were able to decide what is the best approach for our platform.

This chapter describes the scenarios that our platform intends to address, the proposed approach, and the functional and non-functional requirements.

3.1 Scenarios of the Problem

The following scenarios allow us to describe how the platform help address each of these problems. Each scenario represents a different type of situation with a different type of user.

Section 3.3 then describes how these scenarios are addressed using our platform.

3.1.1 Scenario 1: UI Optimization - Non-tech savvy users

Robert is a 35 year old football enthusiast. Since he has some knowledge about football, he likes to bet on games to try and make some extra money. To help him, he regularly uses a sports website that provides statistics of the teams and their previous games.

The website that he uses is known for providing the most reliable statistics, giving information about all that happened in the past. However, this is a downside for Robert too, since it requires more time to gather all the information that he needs.

Robert only likes to make safe bets, so he only bets in the teams that he has more knowledge, which makes most of the information provided by the website useless to him. Due to this, Robert would like a way to navigate faster in this website (e.g., removing the information about other sports, teams and statistics that he does not take into account, etc.), in other to save him some time gathering the information to place a bet. After all, timing is a key factor to make the most money on a bet, since the odds can change every minute.

3.1.2 Scenario 2: UI Optimization - Disabled Users

Lana is a 30 year old worker of a logistics company. She works at a warehouse, arranging the products from suppliers, in order for them to be delivered to retail stores.

Unfortunately, she got injured during a work accident, and now she can not move. She went to the doctor and was informed that her condition is temporary, but she had put a few weeks of sick leave.

With the amount of time that she now spends at home, she started to spend more time using the computer to keep in touch with her friends. However, her condition makes it a bit difficult to use web applications that require a lot of interactivity, and she can not spend a large period of time in front of the computer, because she needs a lot of rest to recover.

It was obvious for Lana that she needed a way to improve the interaction with the websites that she visits, in order to reduce the amount of effort needed to navigate them and thus reducing the amount of time spent in front of the computer. Howver, she can't do it herself.

3.1.3 Scenario 3: UI Optimization - Specific Necessities

José was a secretary for a large insurance company until two years ago when he retired.

With the money that he gathered over the years, he started many side jobs, including selling cars and renting houses to other people. He uses a website to advertise the cars that he is selling and the houses that he wants to rent.

Due to his age, José has some special necessities. His vision is not the same as it once was and his lack of body strength causes him to tremble a bit. This affects his use of the computer, which he needs to use to keep his side hustles running.

Even though he has no advanced knowledge regarding computers, his work as a secretary made him familiar with computers and the internet, so he started searching for websites that could improve the interface of the website according to his needs.

Unfortunately, in any of the websites that he visited, José didn't find any solution that addressed all of his requirements.

3.2 Proposed Approach

While defining our approach, taking into account our main objectives and the research performed in the chapter 2, we decided to develop a platform comprised by two components:

1. An **editing tool**, responsible for the customization of the web pages. Any user should be able to customize any web page according to his needs. This means that the editing tool should provide functionalities for all kinds of users (i.e, experts and non experts).

2. A **web repository**. The web repository is responsible for displaying all the available adaptations created by other users. Any user can open the web repository and download an adaptation that will be applied when he visits the desired website.
   The web repository can also be used to create requests. If a user does not find an adaptation that satisfied all his needs, he can create a request that will be visible to the community, describing what he wishes. Any user should then be able to address this request, creating the adaptation using the editing tool, and sending it as a reply to the user who sent the request.

Our platform offers its users the possibility to use any adaptation available in the web repository as a starting point, and further adapt it to ensure that the end result fully satisfies him. This way, if the user does not find an adaptation that satisfies all his needs, he can import an adaptation that satisfies some of his needs and further adapt it.

Finally, for security reasons, our editing tool should provide the option to create the adaptations as public or private, as some users might not be comfortable with sharing their adaptations. This means that only the public adaptations will become available in the web repository, and the private will be available only to their creators.

3.3 Interaction Scenarios

This section describes how our platform is able to help the users in the scenarios that were previously presented.

Although the focus of this section is these specific scenarios,it is important to mention that our platform should be helpful for a much more extensive group of people.

3.3.1 Scenario 1: UI Optimization - Non-tech savvy users

During the playoffs of the Champions League, Robert placed a good amount of bets and won a nice amount of money. However, he noticed that he could've made much more money with the same bets if the timing was better.

To make sure that this won't happen again, he starts looking for ways to improve his navigation in the statistics website.

During his search, he is presented with our platform. He then starts using the editing tool to remove the unnecessary information, highlighting the most relevant statistics, and

creating shortcuts to the pages of the website that he most regularly visits.

These adjustments allowed Robert to quickly navigate to the pages of the teams and leagues that he is interested in, and the statistics that he thinks that are useless are removed from the screen, improving a lot the time required to navigate the website and the time required to look for the statistics that are important to him.

3.3.2 Scenario 2: UI Optimization - Disabled Users

A few days had passed by and Lana still had difficulties using the computer for long periods of time. She shared her frustration with her friends in social media.

Suzy, one of her friends, introduced her to our platform. Lana told Suzy that the platform seems interesting and would like to try and create some adaptations that improve her experience while using the computer.

Suzy had already started learning how to use the platform, and she knew that due to Lana's situation, it would be hard for her to create her own adaptations, so she offered to help her. She asks Lana what websites she uses regularly and what she wants to improve, and then created some adaptations that might be helpful for Lana.

Lana then used our platform's web repository to search for the adaptations created by Suzy and install them. After installing the adaptations, her experience has significantly improved, so she thanked Suzy for her work and ensured her that when she gets better she will develop some adaptations to share with her as well.

3.3.3 Scenario 3: UI Optimization - Specific Necessities

The days had gone by and José was still struggling to find a solution that satisfied him.

He then finds our platform and starts looking on the web repository. He notices that, like the other websites that he visited, our web repository doesn't have an adaptation that satisfies all his needs.

However, he notices that our web repository also provides the option to create requests for the community do address. Taking advantage of this functionality, José creates a request describing everything that he needs to keep using the website.

A few days had gone by and José comes back to our web repository. He notices that someone already replied to his request. He checks the information about the user who replied to him and notices that the he is very active on the platform and has good feedback on the adaptations that he created, so he decides to download the adaptation.

After trying the new version of the website, José is now able to keep up with his side jobs, which was impossible with the original version due to his vision and motor impairments.

3.4 Requirements

This section describes the system's functional and non-functional requirements.

3.4.1 Functional Requirements

- **Freedom/Control:**
  - Users should be able to import multiple adaptations for one website, and switch between them at any time.
  - Users should be able to import any adaptation from the web application, and further adapt it using the editing tool, using the initial adaptation as a starting point.
- **Continuity:** Since any website can modify its structure, some adaptations may become obsolete over time. The approach must take into account any change that affects the implementation of the adaptation, and act upon it.
- **User friendly:** Every user should be able to use, create and share changes to any web page. This means that a user with no programming experience should be able to create and share its own changes.
- **Community oriented:** Users should be able to communicate with the community, share ideas or report any existing problem.
- **Discoverability:** When searching for available adaptations in the web repository, users should be presented with the options that are most relevant.
- **Transparency:** Previously imported adaptations should be automatically applied when the user opens a web page.

3.4.2 Non-functional Requirements:

- **Privacy:** Some users might not want to share their adaptations with the community. Should the user decide to store an adaptation as private, other users should not have access to it.
- **Usability:** Our platform should be accessible to all kinds of users. Although the editing tool allows advanced users to create their Javascript code and CSS rules, it

is also important to allow users to create changes to the interface without requiring any programming knowledge.

- **Documentation:** This platform (especially the editing tool) needs to be user-friendly, in order to allow all types of users to create and share their adaptations. Providing user guides on how to use our tools is a key component to achieving this.
- **Scalability:** Since all adaptations (both private and public) need to be stored in a repository, it is inevitable that the size of the repository will grow exponentially with time. That being said, it is important to choose an approach that is able to store large amounts of data.

3.5 Summary

Taking into consideration the research performed in the previous chapter and the type of scenarios that we want to address, we were able to design our system and define its functional and noon-functional requirements.

Our system consists in a platform that is comprised by two components: an editing tool that allows users to create and save new versions of any UI, and a web repository where users can share their adaptations with the community.

In order to enable every user to user our platform, we should make it as user friendly as possible. For this reason, the system should include features that are easy to implement for non-tech-savvy users, as well as features that reduce the amount of work required to achieve the version of an interface (e.g., using an already existing adaptation as a starting point to further customization).

Chapter 4

Implementation

This chapter describes our platform GitUI, presenting the system overview, the data used in the system, and the platforms and tools used to its development.

4.1 System Overview

The GitUI platform is comprised of two components:

1. The **editing tool:** The editing tool is implemented as a chrome extension and is responsible for the creation and implementation of adaptations. An adaptation consists of a set of operations that, when applied to the target website, result in a new version of the website that is fully adapted to the user. The editing tool should be accessible to all types of users (i.e., from beginners to advanced users).

2. The **web repository:** The second component is a web application that acts as a web repository, displaying all the adaptations created by the community, in order for other users to choose and download the ones that they find useful.

For this platform to work, the two components need to be able to communicate between themselves. The editing tool needs to be able to store the adaptations created, not only to implement them when the user visits the same web page again but also to make them available in the web repository.

Also, the editing tool should be able to receive the adaptations that the user downloaded from the web repository.

This highlights the need for a data storage platform. In the section 4.2.1 we present the considered options and the pros and cons of each one.

After a deep analysis of the system requirements, we have developed the use case diagram for both components of our platform. These diagrams allow us to have a perception of the functionalities that both components provide, as well as the interaction between the user and the system.

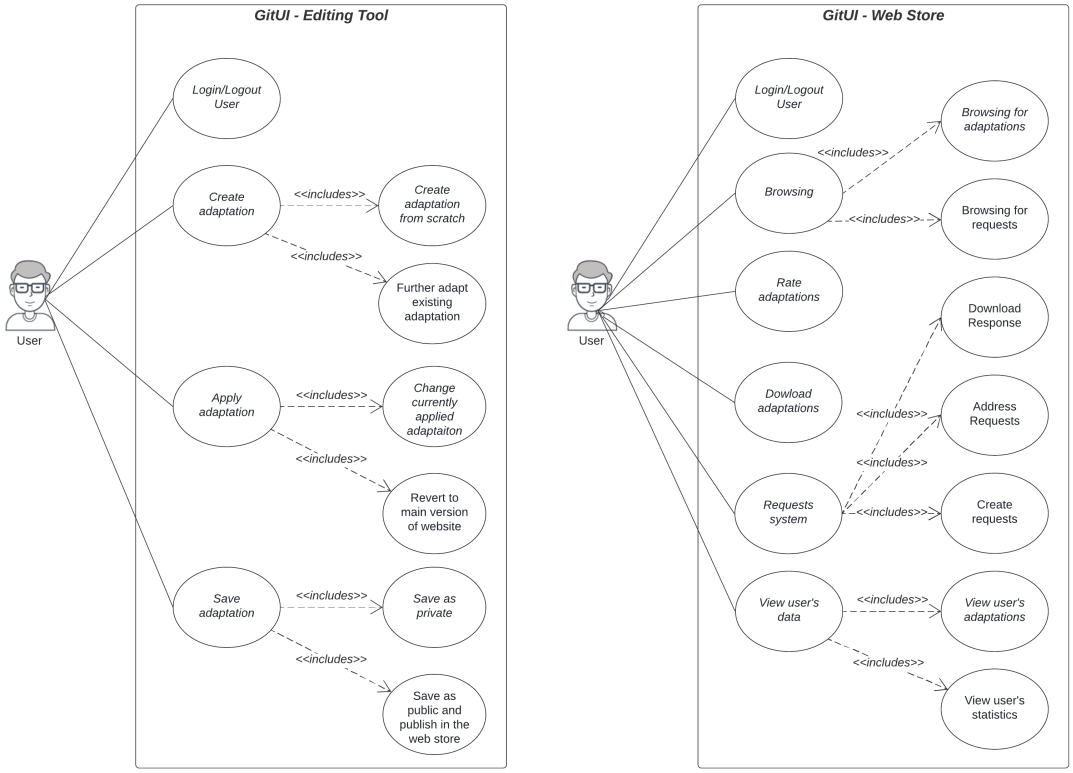


Figure 4.1: GitUI's use case diagram

4.2 Data storage and sharing

This section describes how we decided to structure our data.

Our data is saved as JSON using Firebase's Cloud Firestore[1]. We have identified three main collections that need to be stored (as it is shown in the table 4.1):

- **Users:** Regarding the users, we use an authentication system (described in the section 4.2.1) to authenticate each user. However, there are some data related to the user that is crucial to make our platform work. Each document contains information about the user's adaptations, requests and requests that he is currently addressing. Also, some information about the user (e.g., username, email, etc.) is stored to be displayed on the user's page in the web repository.

- **Adaptations:** The adaptations are the main data of our platform. It's the data that our system uses to apply the customizations created by the users to a website. An adaptation consists mainly of a set of operations, a script and a CSS stylesheet that are injected into the target web page when the user visits it. The combination of these three types of customization will result in a new version of the target website.

[1] https://firebase.google.com/docs/firestore

There are also other fields that are useful for the web repository (e.g., tags can be added to the adaptation to facilitate browsing in the web repository; the rating and number of downloads can be useful for gamification purposes, etc.). A deeper explanation of these fields is provided in the sections 4.3 and 4.4.

- **Requests:** Each request has two main fields, the target website and the body of the request. When creating a request, the user should provide a detailed description of what he wishes and, if needed, there is also the option to provide an image with some visual explanation.

User	
UID	Integer
Username	String
Email	String
PhotoURL	String
Requests	Array
Requests Received	Array
Requests Rated	Array
Adaptations	Array

Adaptation	
Owner	String
Operations	Array
Script	String
Stylesheet	String
Title	String
Description	String
Private	Boolean
Tags	Array
Creation Date	Timestamp
Last Updated	Timestamp
Rating	Float
Comments	Array
Downloads	Integer
Website	String
Image	String

Request	
Owner	String
Website	String
Response	String
Image	String
Submission Date	Timestamp
Text	String
Verified	Boolean
Adressed	Boolean
Adressed by	String

Table 4.1: GitUI's Data structure

Looking at our data model (described in the table 4.1), there are no complex data types that need to be stored. Only the images are stored separately, using Firebase's Cloud Storage[2].

The adaptation model contains a field that is particularly important to the social aspect of our platform: the *Private* field. The *Private* field is a Boolean that indicates whether an adaptation should be available in the web repository for other users or only to the creator.

[2]https://firebase.google.com/docs/storage

Some users might not want to share their adaptations with the community. For this reason, if an adaptation is set to private, only its creator can view it in the web repository.

That being said, we need to store three different types of data:

- **Main data:** The main data of our platform consists of the three collections mentioned above (Users, Adaptations, and Requests). This data is stored as JSON.

- **Images:** As previously mentioned, our platform will store images separately from the rest of the data.

- **Authentication data:** Although we have a collection for the user data, the authentication system uses a different procedure to implement all its functionalities. Some of its data is still used in our database (i.e., the user UID, Username, Email and PhotoURL), but this requires to have a separate set of data for the authentication system.

The next section will discuss the different methods and platforms used to store these different types of data.

4.2.1 Storage Platform

The initial idea of this project was to use a GitHub repository to store our data and communicate with it using GitHub's REST API. However, after analysing our data structure and system requirements, we came to the conclusion that using a GitHub repository to store and manage our data would require a much more complex approach, whereas Firebase offers different components (e.g., Cloud Firestore, Cloud Storage, Authentication system) that facilitate the development of our system.

Firebase is an app development platform that offers many features that are very helpful to our platform, such as two different types of databases to store our data, a storage system to store our images and an integrated authentication system that allows authentication using external providers (e.g., Google, Facebook, etc.).

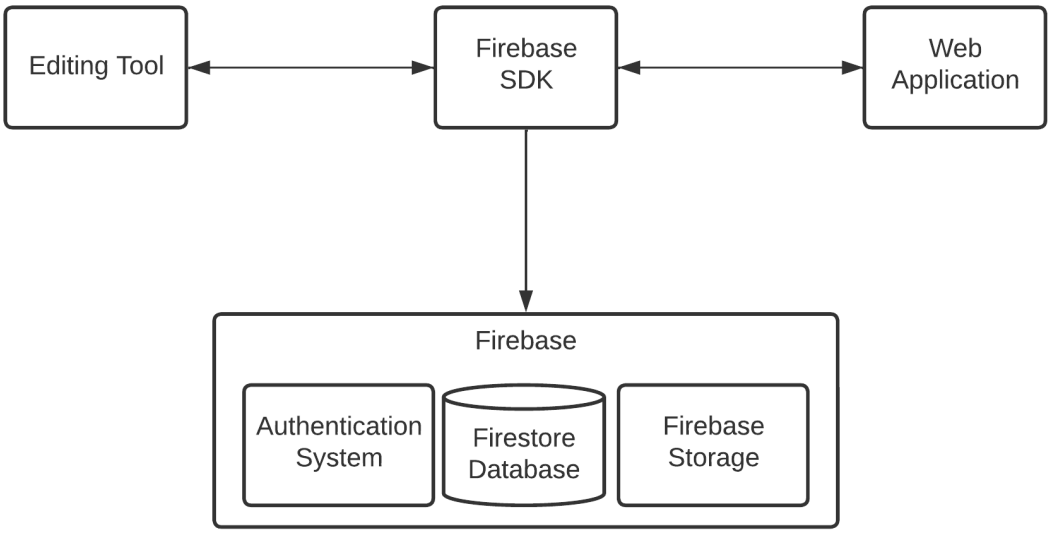


Figure 4.2: Platform's Structure.

Figure 4.2 describes the structure of our platform using Firebase. The Firebase SDK is responsible for the communication between Firebase and both our components, discarding the need of an API, and it provides a lot of features that facilitate the communication process.

4.2.1.1 Realtime Database vs Cloud Firestore

Firebase offers two types of databases: Realtime Database and Cloud Firestore. Both have their advantages and disadvantages, so it is important to consider which option to use in our platform.

According to Firebase official documentation, Realtime Database stores the data as a big JSON tree, while Cloud Firestore stores it as a collection of documents.

Cloud Firestore provides the ability to store complex and/or hierarchical data using subcollections within documents. This can be useful regarding the version control of adaptations.

Regarding reliability and performance, Realtime Database has an extremely low-latency, making it an ideal option for frequent state-syncing. However it's a regional solution, making it limited to zonal availability within a region. On the other hand, Cloud Firestore stores the data across multiple data centers in distinct regions, ensuring global scalability and strong reliability.

Both options provide security when reads and writes are executed through the SDK. Cloud Firestore also provides the ability to create rules that constrain queries. This means that if the query's results might contain data the user doesn't have access to, the entire query fails.

The final aspect to take in consideration is the pricing of both options. Realtime Database charges only for bandwidth and storage, but at a higher rate (5€ for each GB/month). Cloud Firestore charges primarily on operations performed in your database (read, write, delete) and, at a lower rate, bandwidth and storage.

4.2.1.2 Version Control

As mentioned above, Cloud Firestore provides the ability to store hierarchical data using subcollections within documents. Taking advantage of this functionality, it's possible to synchronize the state of all different versions of a parent adaptation.

By comparing the dates on which the main adaptation and those that derive from it was last updated, it is possible to detect when an adaptation is out of date. The user can then be notified that a new version of his adaptation is available for download.

4.3 GitUI Editing Tool

As previously mentioned, the editing tool is responsible for creating and applying adaptations. In order to do this, it needs to have full access to the contents of the website. For this reason, we decided to create the editing tool as a chrome extension.

Extensions are built mainly using JavaScript, HTML and CSS, and they provide different components that allows users to interact with any website, such as content scripts, background pages, and UI elements (implemented as a popup in our platform).

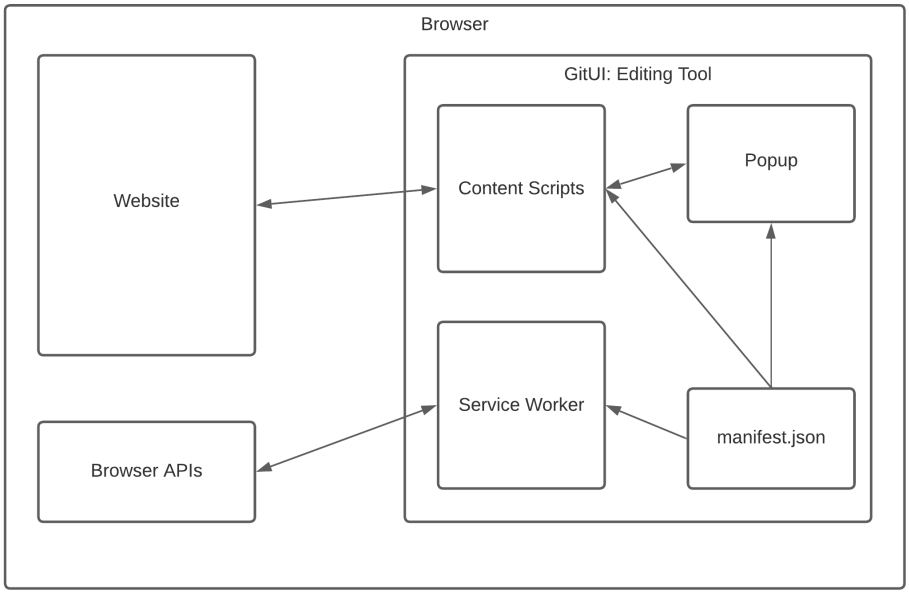


Figure 4.3: GitUI Editing tool architecture.

Looking at the extension's architecture, we can see the roles of the different components used in its development:

- **Manifest:** The manifest is the core of an extension. It's a JSON-formatted file that contains all the information about the extension. All the content scripts, service workers, permissions, etc. should be declared in this file.

- **Service Worker:** Service workers are the extension's event handlers. It listens for browser events and acts upon them as instructed. The service worker has access to all browser APIs, as long as they are declared in the manifest.

- **Content Scripts:** Content Scripts are responsible for the communication between the extension and the website. They allow logic to be inserted into a website in order to read/modify its contents. This is the component that is responsible for the customization process of the website.

- **Popup:** The popup is responsible for the interaction between the user and the website. The popup contains the set of instructions that the user can send to the content script, indicating which task should be performed next. Users use functionalities provided by the popup to create their adaptations, as well as to change the adaptation that is currently being applied.

4.3.1 Browser APIs

Browser APIs, declared as "permissions" in the *manifest.json* file, are extension-specific APIs that are used in the interaction between the extension and the browser. Our platform makes use of four of these APIs:

- **Scripting:** Used to execute scripts in different websites.

- **Storage:** Provides access to the different types of storage provided by the browser. Our platform uses local storage to store its persistent data, as it is the type of storage that allows storing the largest amount of data.

- **Tabs:** Used to interact with the different browser tabs. This is the API that we use to access and interact with the contents of web pages.

- **Identity:** This API is used to get the OAuth2 access token of the current user logged in to the browser.

4.3.2 Firebase Integration

Starting from January 17, 2022, the Chrome web repository stopped accepting new extensions that use the manifest version 2 (MV2)[3]. This required us to develop the editing tool using manifest version 3 (MV3), which provides a number of new features and functional changes.

There was one functional change that had a significant impact on the development of our extension.

"Remotely hosted code is no longer allowed; an extension can only execute JavaScript that is included within its package."[4]

This constrain means that we can not import Javascript libraries (e.g., Firebase) using a Content Delivery Network (CDN).

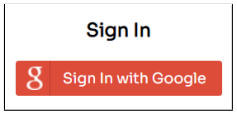
To work around this constraint, we decided to use Webpack, a module bundler, to build a dependency graph of our entries (i.e., the libraries that our extension uses), combining them into a bundle of static assets that are available for our extension to use. Using this method, we can install our extension's libraries using *npm* (Node Package Manager) and declare them as entries for Webpack.

During the extension's building process, Webpack then converts these entries into a bundle of static files that are available for our extension to use, avoiding the need for CDNs to import our extension's dependencies.

4.3.3 User Interaction

The user interaction is performed using the extension's popup. Using the popup, the user can start creating an adaptation or change the currently applied adaptation.

[3] https://developer.chrome.com/blog/mv2-transition/
[4] https://developer.chrome.com/docs/extensions/mv3/intro/mv3-overview/



(a) Logged out screen.

(b) Logged in screen.

Figure 4.4: Editing tool - Popup.

To change the currently applied adaptation, the user simply needs to change the selected option of the *select* input above the Move button. If the selected option is empty (as it is in the figure above), it means that the user is using the original version of the website.

As it can be seen in the figure 4.4, our editing tool provides 9 different operations that our user can use to customize the website (similar to those implemented in [11] and [12]). Each operation requires a certain number of steps to implement, as it is described below:

- **Move:** Changes the parent of an element, changing its position on the page.
  Steps:
  1. Click on the button “Move” in the popup.
  2. Click on the DOM element you want to move (the selected element will be highlighted in green, as it is in figure 4.5).
  3. Click on the parent you want to insert the element in (The new parent will be highlighted in blue, as it is in figure 4.5). The element will be inserted as the last child of the new parent.
  4. Use the “Up Arrow” and “Down Arrow” keys to move the element up and down respectively in the parent’s child list.
  5. Click the “Enter” key to complete the operation or the “Escape” key to cancel the operation.
- **Reorder:** Reorders an element, changing its index in the parent’s child list.
  Steps:

1. Click on the button “Reorder”.
2. Click on the element you want to reorder (the selected element will be highlighted in green, and its parent will be highlighted in blue).
3. If the clicked element is contained inside the element you want to reorder but it is not t the desired element, use the “Left Arrow” key to switch to the element’s parent.
4. After reaching the desired element, use the “Up Arrow” and “Down Arrow” keys to move the element up and down respectively in the parent’s child list.
5. Click the “Enter” key to complete the operation or the “Escape” key to cancel the operation.

- **Resize:** Changes the width and/or height of an element.
  Steps:

  1. Click on the button “Resize”.
  2. Click on the element you want to resize (the selected element will be highlighted in green).
  3. Move the cursor horizontally and/or vertically to change the element’s width and height respectively.
  4. Click again to complete the operation.

- **Spacer** Increases or reduces the margin of an element.
  Steps:

  1. Select the type of margin you want to increase/reduce.
  2. Click on the element you want to adjust (the selected element will be highlighted in green).
  3. Move the cursor to resize the previously selected margin (vertically if the selected margin is “Top” or “Bottom”,
  4. horizontally if the selected margin is “Left” or “Right”).
  5. Click again to complete the operation.

- **Hide:** Hides an element from the page.
  Steps:

  1. Click on the button “Hide”.
  2. Click on the element you want to hide (the selected element will be highlighted in green).

- **Shortcut:** Creates a button that, when clicked, redirects the user to another page based on a URL provided by the user.
  Steps:

  1. Insert the text that will be displayed on the button.
  2. Insert the destination URL. The URL should follow the "HTTPS" or "HTTP" scheme.
  3. Click on the button "Shortcut".
  4. Move the cursor to the place where you wish to insert the button.
  5. Click to complete the operation.

- **Font-Size:** Changes the font size of an element.
  Steps:

  1. Select the desired font size (in pixels).
  2. Click the button "Font-Size".
  3. Click on the element you want to adapt (the selected element will be highlighted in green).

- **Change color:** Changes the text color of an element.
  Steps:

  1. Select the desired color using the color picker.
  2. Click the button "Change color".
  3. Click on the element you want to adapt (the selected element will be highlighted in green).

- **Inspect Element:** Provides information about an element (e.g., its tag, id, class list and parents).
  Steps:

  1. Click on the button "Inspect Element".
  2. Click on the element you want to inspect (the selected element will be highlighted in green).

We aim to enable every to use any of these operations, regardless of their experience with computers. However, if there is any issue with any of the operations or other functionality, in the top right of the popup (see figure 4.4b) there is a button that the user can click to open a modal displaying useful information of all the operations and explaining the different functionalities of the editing tool.

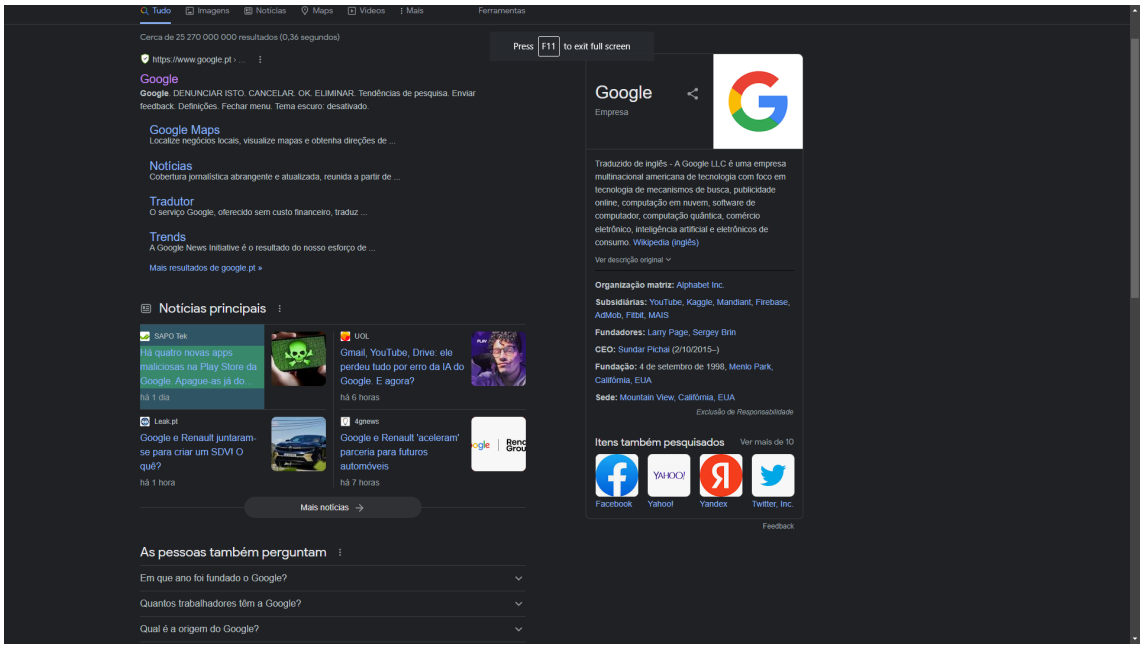


Figure 4.5: Editing Tool - Highlighted element.

For more advanced users, besides these operations, we also provide the possibility to inject their own JavaScript code and/or CSS stylesheet through the modal displayed in the figure 4.6. Our operations should be enough to create a better and more accessible version of a website, but if a user wants to include new functionalities or adapt a website in ways that are too complex using only those 9 operations, this should help him with that process.

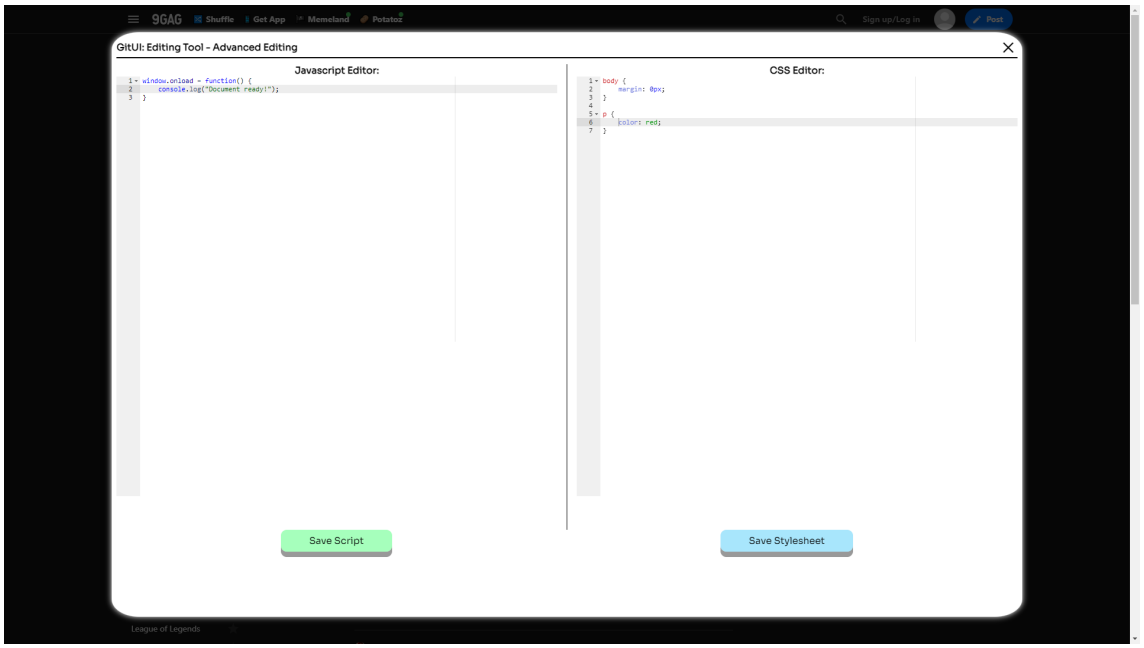


Figure 4.6: Editing Tool - Code editor.

Even though not all users will take advantage of this functionality, due to the social aspect of our platform, advanced users can create more complex and robust adaptations, and then share them in our web repository, for all other users to use, including those with no computer experience.

After the user is satisfied with the adaptation, he can save it using the button at the bottom of the popup (see figure 4.4b). The modal represented in the figure below will be displayed, for the user to insert the remaining data that will be attached to the adaptation.

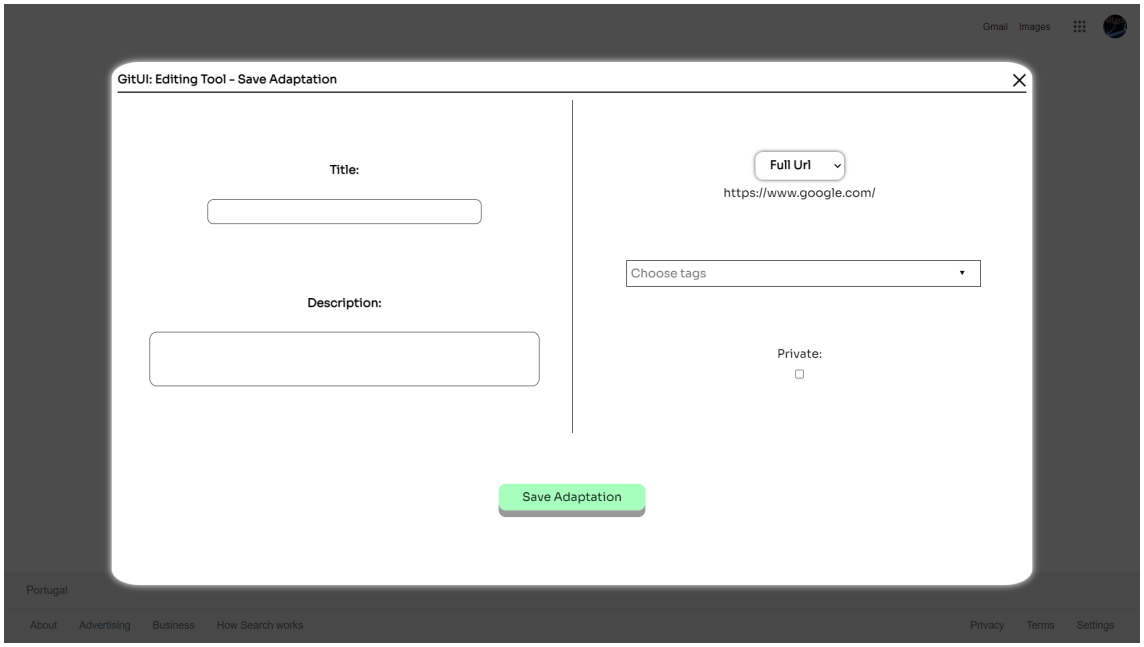


Figure 4.7: Editing Tool - Saving a new Adaptation.

The user should insert a title and a description for the adaptation and, on the right side of the modal, there are some options that need to be selected.

The first one is the URL to which the adaptation should be applied. Any adaptation has two options for this field: **Full Url** and **Base Url**. The first option applies this adaptation only to the current page, while the second applied the adaptation to the root page of the website and all its routes. The second option is useful for websites that share the same template for most of their routes, allowing users to use the same adaptation for the whole website.

The second field is a multi-select for the tags that will be associated with the adaptation. The user can choose one or more tags to add to the adaptation (e.g., Visual Impairment, Color Blindness, etc.). This field has no impact on the adaptation itself, but it is useful for the web repository, as it makes the adaptation easier to find when browsing adaptations filtered by tags.

Lastly, there is the *private* field, previously mentioned in section 4.2. This is the

Boolean field that will decide whether the adaptation will be made available in the web repository, or only to its creator.

If the user is further developing an already existing adaptation, the modal will have a slightly different appearance (see figure 4.8).

Figure 4.8: Editing Tool - Updating an already existing Adaptation.

Now the user has two options: save the new version as a whole new adaptation, or update the existing one. This is where the sharing process of our platform takes place. Any user is free to further adapt an adaptation that belongs to another user, resulting in a version of the adaptation that is optimized for a specific user. This means that a user is able to work on top of another user's work.

That being said, the update button has two options:

1. If the user updating the adaptation is its creator, a simple update will be performed. The data of the adaptation will be updated, as well as the date of the last update.

2. If the user is updating an adaptation that belongs to another user, a new version of that adaptation will be stored as a child of the main adaptation, taking advantage of the hierarchical storage system of Firestore's database. This allows all the adaptations that derive from another adaptation to be notified when the main adaptation is updated.

4.4 GitUI Web Application

The main purpose of our web application is to act as a web repository, receiving the data created in the editing tool, and displaying it for the whole community, for other users to download the adaptations that they find useful.

The web application is built using *Node.js* and *React* for the front-end. React is a front-end library that takes advantage of JSX (Javascript XML), a syntax extension that facilitates the creation of components, resulting in a simpler and cleaner source code.

When a DOM update is necessary, instead of updating the whole DOM, React compares it to a virtual DOM, in order to identify the necessary changes to bring the DOM to the desired state, and update only the necessary elements[5].

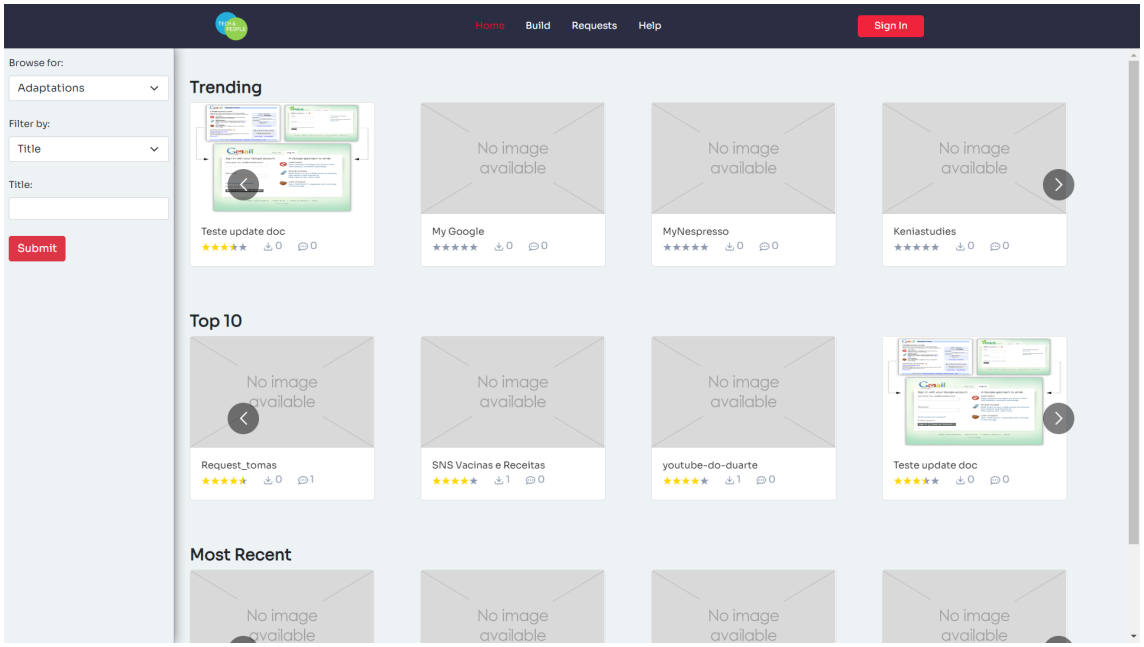


Figure 4.9: Web Application - Home page.

Figure 4.9 is a screenshot of the home page of our web application, where we can identify its five different routes:

- **Home:** The main page of our application. Provides users with the Trending, Top 10 and Most Recent adaptations, sorted by rating. Also allows browsing for both adaptations and requests, using the filters on the sidebar on the left side of the screen.

- **Build:** Contains a brief explanation of the editing tool functionalities, and provides a link for users to download it.

[5]https://reactjs.org/docs/rendering-elements.html

- **Requests:** The requests page allows users to search for other users' requests. If the user is logged in, three additional tabs are available, for the user to see the requests that he created, the ones that he is currently addressing, and the last tab can be used to create a new request.
- **Help:** Provides the user with a guide on how to use our platform.
- **Sign In/Profile:** If no user is logged in, redirects the user to the login page. Otherwise, displays the user's profile page.

4.4.1 User Profile

Our platform uses Firebase Authentication to allow users to log in using their Google Account.

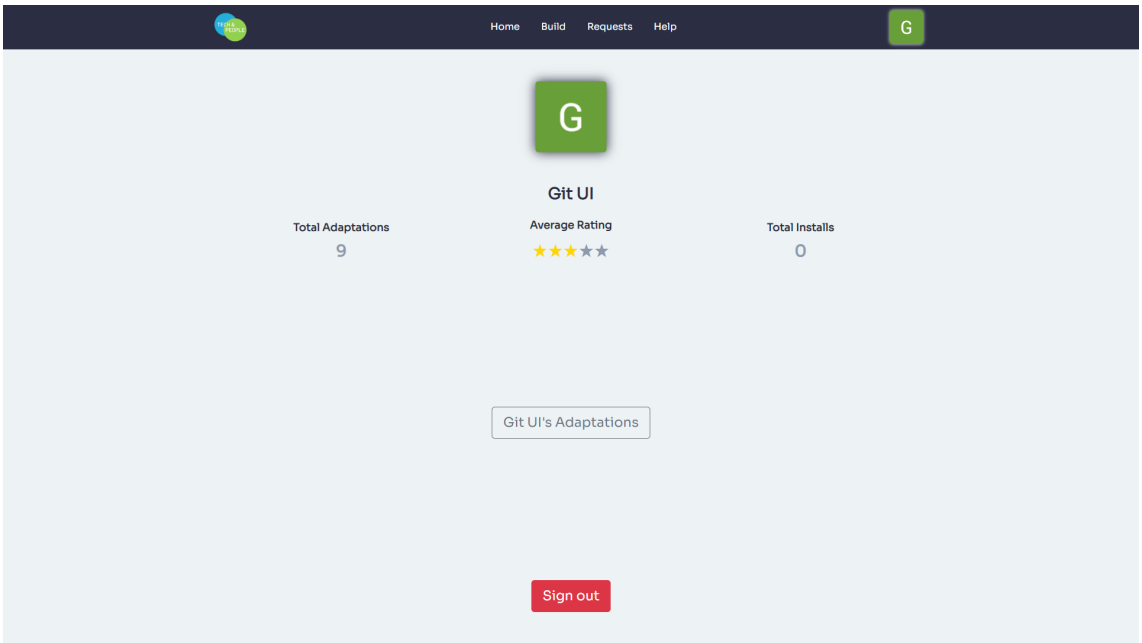


Figure 4.10: Web Application - Profile page.

Once logged in, the user is able to access the profile page (present in figure 4.10), where he can see his information. The number of adaptations, average rating and total installs of each user is visible to the whole community.

These statistics act as a gamification system for our platform since users might be more interested in the adaptations created by a user that is acknowledged by the community than a user with a poor rating that has barely created any adaptations.

This page also provides access to the adaptations created by the user. If the user is logged in and navigates to his own adaptations page, he is able to see all of his adaptations,

including the private ones. However, if the user navigates to another user's adaptations page only the public ones will be available.

4.4.2 Adaptations

The adaptation's page displays its information to the community. Once the adaptation is available in the web repository, the user can choose to add an image to it by clicking on the top right of the default image (see figure 4.11).

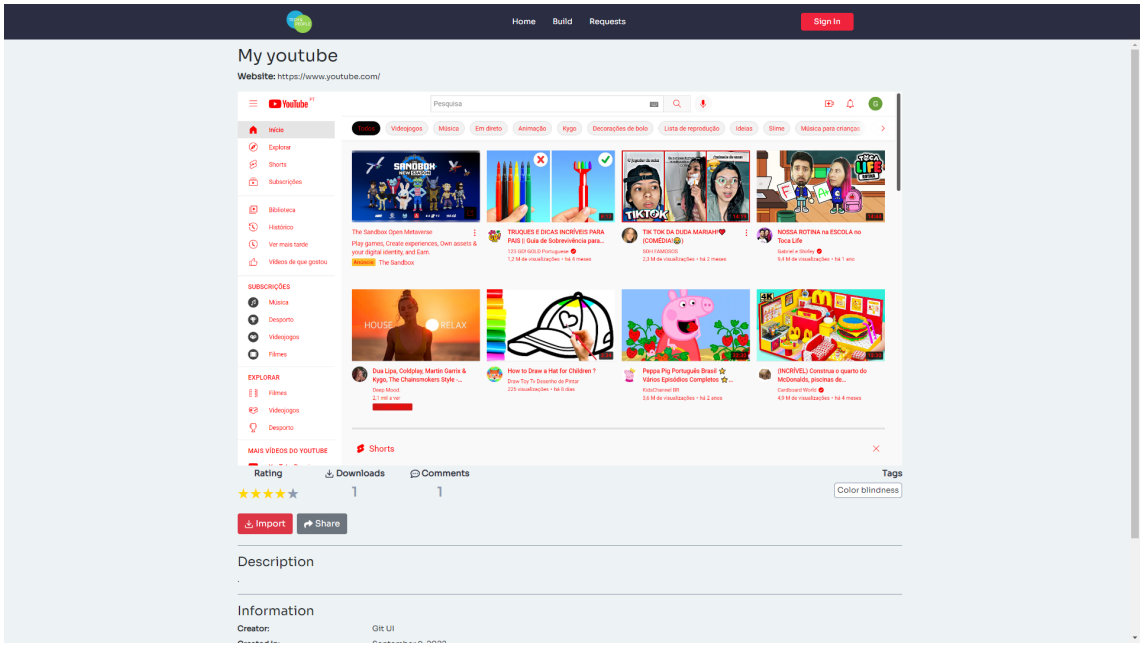


Figure 4.11: Web Application - Adaptation page (Top).

Any user can give feedback on an adaptation in two ways: rating an adaptation by clicking in the stars below the adaptation's image, and discussing the adaptation in the comment section, at the bottom of the page (see figure 4.12).

To import the adaptation, the user simply needs to click the red button that says "Import". If the user is not logged in, the button will redirect him to the login page first.

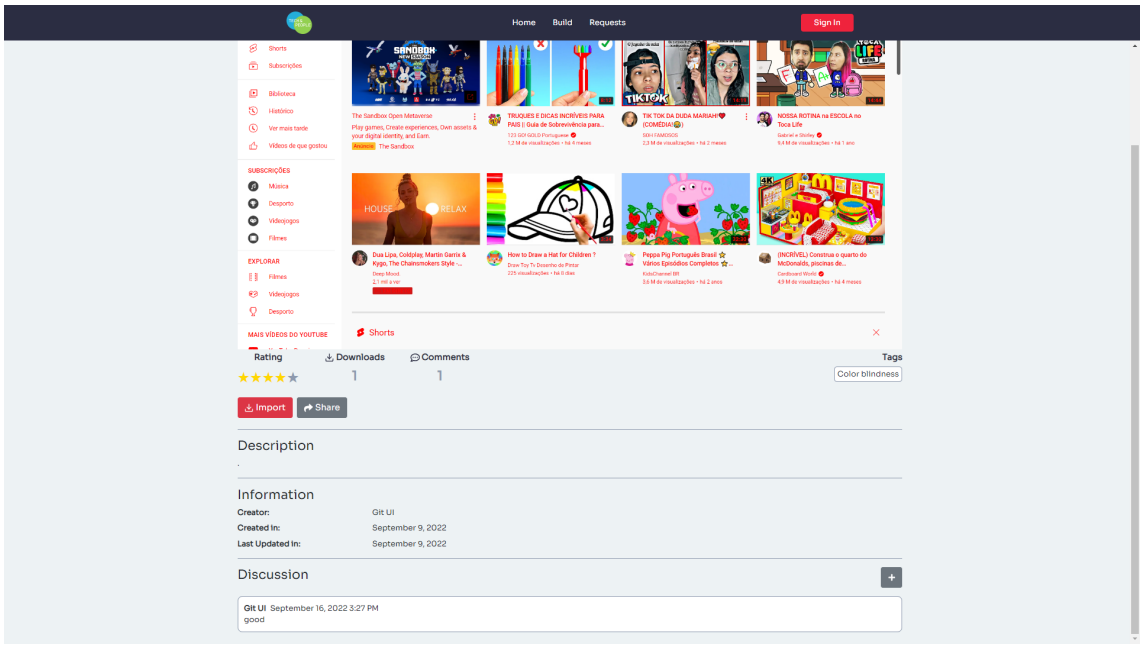


Figure 4.12: Web Application - Adaptation page (Bottom).

There is one additional feature that might be interesting for users who want to share their adaptations with a private group of people. As it can be seen in figure 4.11, there is a share button that, when clicked, copies the URL of the adaptation's page to the clipboard. This allows any user to share their adaptations (including the private ones) adaptations with anyone that gets access to this URL.

4.4.3 Requests

The request system has a unique process to ensure that any user can reply to any request, and that the user who submitted the request is satisfied with the response.

When a user finds a request that he wishes to address, he can click the "Address this request" button (see figure 4.13). The requester can then see which user is currently addressing his request and the date that he started addressing it.

The user replying to the request can use the request's explanation and the image (if provided) to guide him in the development of the response. However, if this is not enough, the request's page has a comment section where users can discuss what is intended.

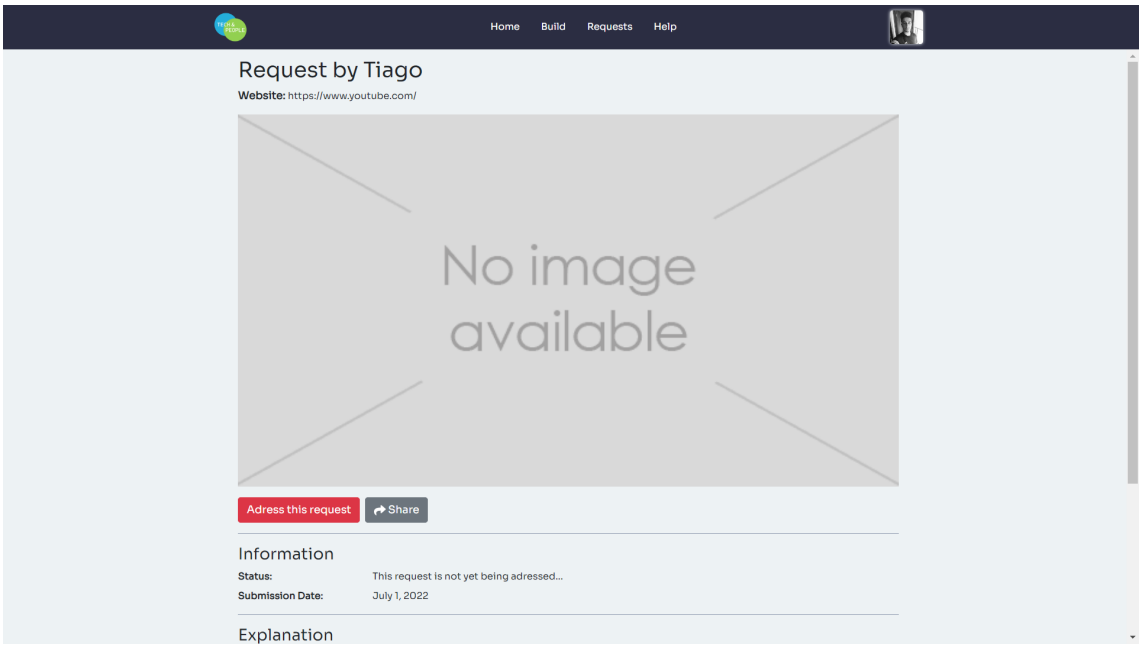


Figure 4.13: Web Application - Request page.

If the creator of the request isn't happy with the response or, for some other reason, wants a different user replying to his request, he can choose to remove the user who is currently replying to the request, making it available for other users to address.

On the other hand, if a response is submitted and the request's creator is happy with it, he can confirm the response and the request will be completed.

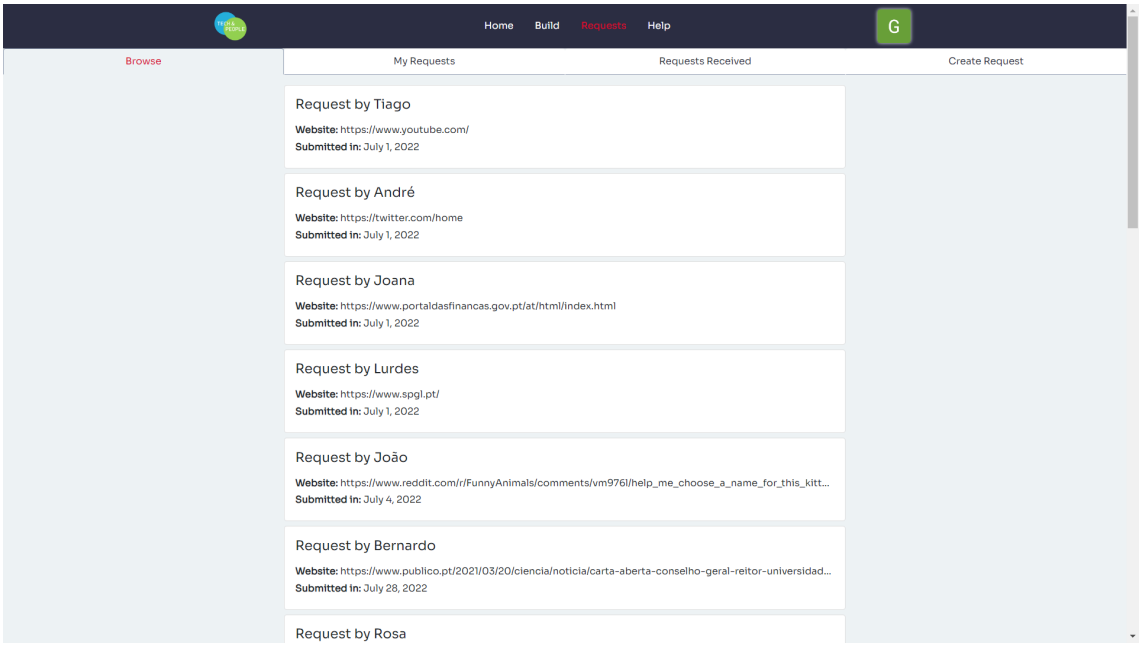


Figure 4.14: Web Application - Requests page.

The requests page (figure 4.14)has four different tabs:

- **Browse tab:** Provides the user with a list of requests available to be addressed. This is the only tab available if the user is not logged in.
- **My requests:** Contains a list of the requests created by the user.
- **Requests Received:** Contains a list of the requests that the user is currently addressing.
- **Create Request:** Allows the user to create a new request, providing its information using the form in the figure 4.15.

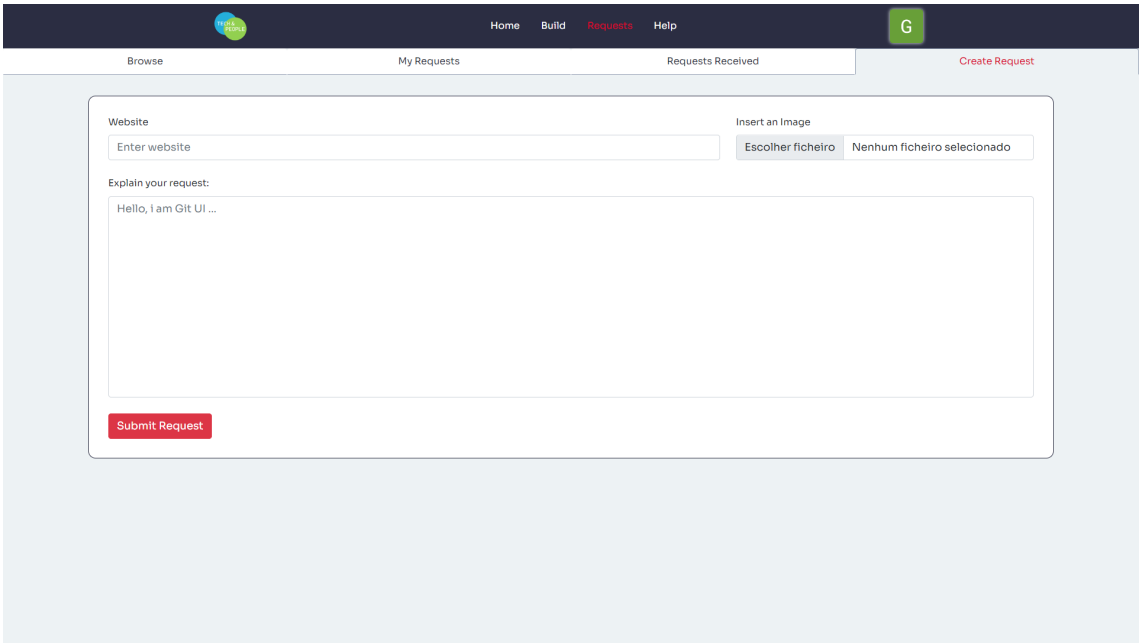


Figure 4.15: Web Application - Creating a request.

When creating a request, the user should provide the target website as well as a detailed explanation of what is intended. Since some requests can be harder to explain using only text, the user can choose to add an image with a visual explanation of what he wants.

Once created, the request will then be available in the browsing tab, and other users can reply to it using the process previously described.

4.5 Summary

Our platform consists in two components: the editing tool and the web repository. The two components are connected to Firebase, which is responsible for the authentication and data storage.

The editing tool is implemented as a chrome extension allows users to customize the UI using a set of operations that are accessible through a popup. For expert users, there is

also the possibility to inject JavaScript and/or CSS code. The adaptations created in the editing tool are persistent, meaning that users will be presented with the new version of the UI each time that they visit the same website again (unless they change the current UI to another adaptation or to the original version).

The web repository, implemented as a web application, allows users to download adaptations that were shared by the community. If there is some feature missing in the downloaded adaptation, users are able to further customize it using the editing tool to achieve the desired version.

If there is no adaptation available in the web repository that satisfies the necessities/interests of a user, the web repository also allows users to create personalization requests that can be addressed by the community.

Chapter 5

User Studies

We have conducted two studies that, among other objectives, aim to evaluate the usability of our platform.

5.1 UI Personalization

The first study was conducted with the intention of evaluating the acceptance and usefulness of our platform's editing tool and informing the design of the web repository. During the course of this study, the participants were unaware of the existence of the second component of our platform, the web application.

This section will present the feedback provided by the users regarding our editing tool. However, we also took the opportunity to explore the opportunities for community-based personalization of UIs, which is more deeply explained in the article [3], where the following research questions are addressed:

- How do people respond to a UI personalization tool?
- Do users have the skills and interest in personalizing an interface for themselves? Do they prefer to ask for help? Are they efficient in asking for help?
- Can UI creators personalize for others? What motivates them (e.g., gamification, gratitude, etc.)? What factors do they consider?
- How does the type of personalization request affect people's responses?

5.1.1 Apparatus

This study uses an adjusted version of our platform's editing tool. This version of the editing tool allows users to create and manage their requests inside of the browser extension, discarding the need to access the web application to use the request system.

The editing tool is now separated into two different sections: the edit menu and the requests menu.

(a) Edit Menu.

(b) Request Menu.

Figure 5.1: GitUI: Editing Tool - Study version.

The edit menu (figure 5.1a) works similarly to what is described on section 4.3. However, this new version has a request system implemented inside the editing tool.

The request system used in this study is a bit different from the one implemented in our web application. Using the popup, the user has access to the requests menu (figure 5.1b), allowing him to create and manage his requests using only the editing tool.

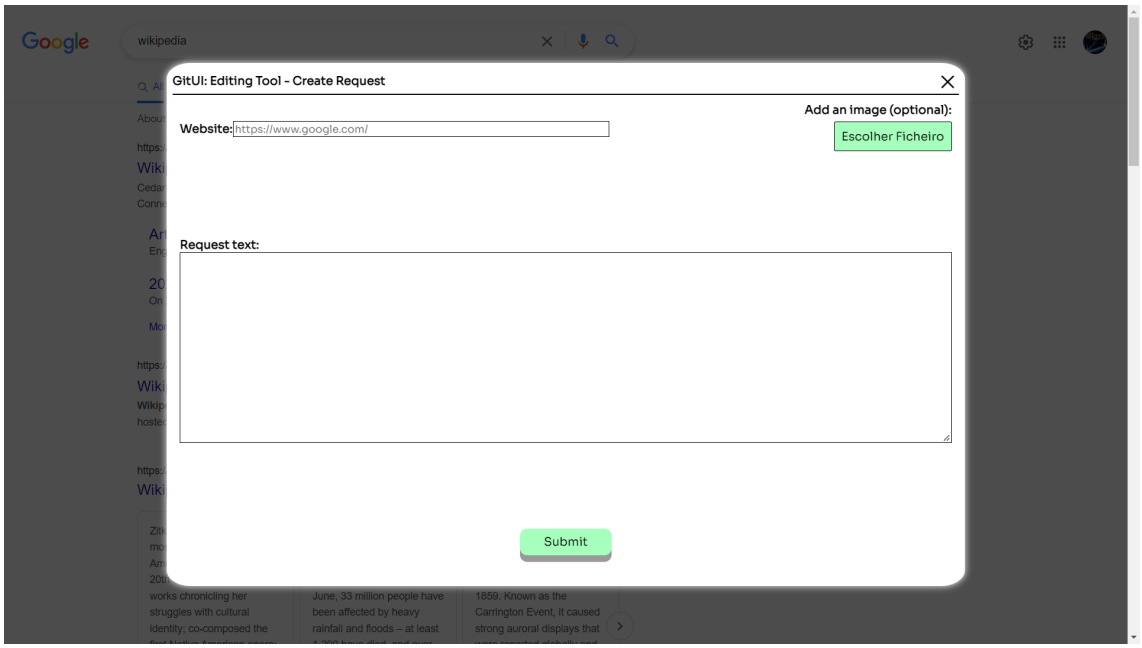


Figure 5.2: GitUI: Editing Tool - Creating a request.

To create a request, the user clicks the first button of the requests menu, opening the

modal in the image above. The user should then insert the target website, a text explanation containing the details of the request and, optionally, an image with a visual explanation of what is intended.

The request will then be available, and the user is able to see its status and, once available, download its reply by clicking on the second button of the requests menu, which presents a list of all the requests created by the user.

To reply to a request, the user should click the third and last button of the requests menu, opening a list of all the requests that he has received. After reading the request explanation, the user should navigate to the target website and start developing the reply using the edit menu of the tool. After creating the reply, the user can open the request again using the list of requests received and is now able to select it as a response to the request.

Once the reply is submitted, the researchers will look at the reply and, if accepted, the request will be marked as verified, making it available for the creator to download the reply.

5.1.2 Participants

The study was performed with people with different technical and demographic profiles. The participants are separated between two groups: advanced users (developers/designers) and casual users.

Inclusion criteria:

- Being interested in participating in the study.
- Signing an informed consent form.
- Using the Internet for more than four hours a week.

Exclusion criteria:

- Inability to participate or inability to sign an informed consent.

ID	Age	Gender	BI	Adv. Computer Skills
P1	25	M	8	Yes
P2	33	F	12	Yes
P3	27	M	6	No
P4	35	F	9	No
P5	59	F	9	No
P6	31	M	12	Yes
P7	28	M	10	Yes
P8	28	M	12	Yes
P9	25	M	9	Yes
P10	23	F	1	No

Table 5.1: Study 1: Participants profile, where BI is the daily hours spent using the Internet.

In total, 10 participants were recruited. From this list (see table 5.1), only 9 participants concluded the study (P1-P9).

Before the beginning of the study we conducted a pilot study with two participants (P1 and P2) in which the only difference was the study duration (1 week), in order to ensure that our tool was working as intended. We decide to include P1 and P2 in the results of our study as they still provided valuable input.

5.1.3 Procedure

Before the start of the study, people who were interested in the study filled out an online questionnaire, which includes a study description, informed consent and questions. This questionnaire was used to collect the participants' demographic data, allowing the research team to select those who fit the inclusion and exclusion criteria to participate in our study.

After all participants are chosen and both groups are built, the study is ready to start. The study is divided into three different phases:

1. **Think-aloud Personalization:** The first phase starts with the researcher sharing the details of the project with the participant and collecting an informed consent signed by the participant.
   The participant then participates in a recorded interview where he/she will be asked to personalize a website in order to make it look as similar as possible to an example given to him by the researcher (example provided in figures 5.3 and 5.4).

Figure 5.3: UI Personalization study: Original interface.

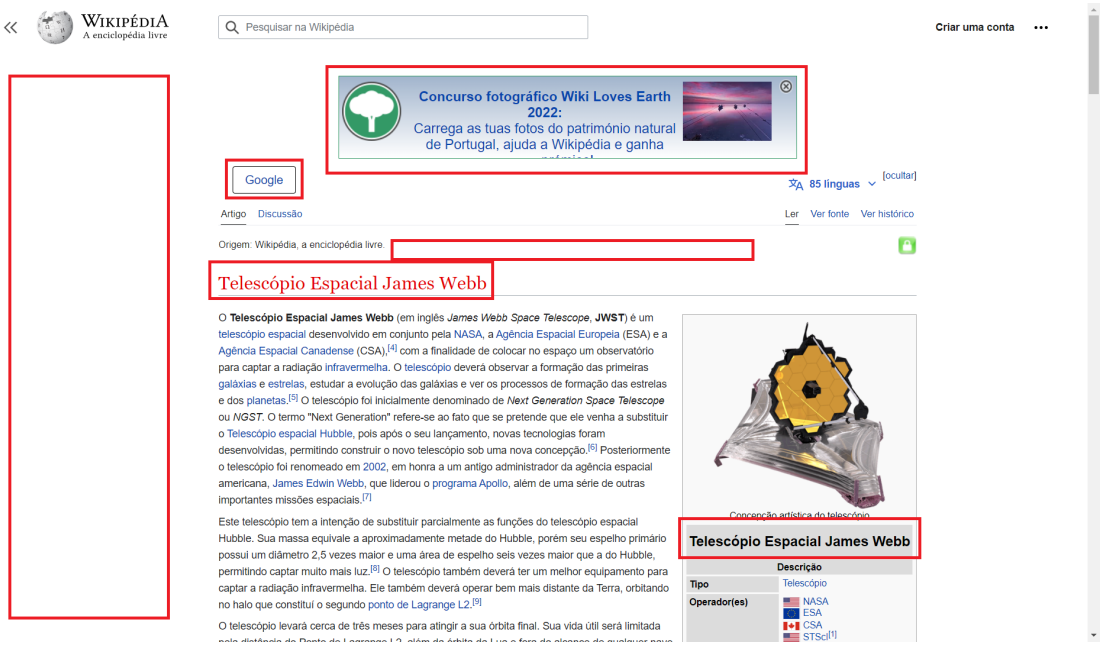


Figure 5.4: UI Personalization study: Desired interface.

During the interview, participants are encouraged to share their thoughts and perspectives about the system and future opportunities they might see.

After completing this exercise, the participants will also be taught how to create and answer personalization requests, as this will be important for the next phase. After the interview, the participant will be asked to complete the SUS[1] (System Usabil-

[1]https://digital.gov/2014/08/29/system-usability-scale-improving-products-since-1986/

ity Scale) questionnaire, a short version of the IPIP-NEO (5 five-factor personality inventory)[6], and a computer self-efficacy scale.

2. **Personalization at Home:** The second phase of this study lasts for two weeks. During this time, each participant will be asked to use our tool as needed and their usage will be seen as an acceptance indicator. The minimum requirement is to use our tool on a minimum of three different websites, personalizing it or creating a request to ask for help.
   The participants will also receive requests made by other participants every two days. These requests can come from other participants or, if not enough requests are created, the researchers will send fictional requests with different levels of complexity, types of websites and user characteristics (age, profession, etc.).

3. **Final interview:** After the two weeks of the previous phase a second interview was conducted, in order to perceive each participant's experience with the tool.

5.1.4 Analysis

To analyze the outcome of the study, two researchers gathered to identify and discuss the themes and sub-themes that will be addressed, resulting in the table below.

User Experience and Usability
• Guidance while customizing • Lack of feedback • Workflow Alternatives • Help • Challenges
Customization for Self
• Goals: Create simpler UIs • Help with the creative process • Motivation • Operations and Code injection • Challenges • UI creators vs Regular Users
Requesting Customizations
• Motivation • Target Users • Self-Confidence • Sense of control • Altruism • Amount of Work
Customization for Others
• Motivation • Impact of empathy • Long Term User Engagement • Planning • Impact of deadlines • Community vs Individual Requests • Impact on Customizing for Self

Table 5.2: Study 1: Themes and sub-themes identified.

The next chapter will present the results obtained regarding the themes and sub-themes identified, however, we only present the results that are relevant to the evaluation of our platform. Some topics are only relevant to answer the research questions mentioned in section 5.1. These topics are better explained in the article [3].

5.1.5 Results

The participants successfully completed the tasks that were given to them in the first interview and quickly understood how most of the operations worked, except the "Move" and "Reorder" operations, which they took a little longer to understand.

While personalizing at home, the participants were able to use most of the operations of our tool confidently, except the "Move" and "Reorder", which have a slightly more complex workflow.

Our System Usability Scale results, which were measured using the same approach as Pekpazar et al. [13], rated our platform eighty-two out of one hundred (82/100).

5.1.5.1 User Experience and Usability

Guidance while customizing and Lack of feedback

These two sub-themes are strongly connected. The major concern the users had while using our tool was knowing their "next step". The tool includes a menu that provides a guide on how to perform each operation, however, the participants felt that they needed another mechanism to help them during the creative process. Some users said that including a section with the different possible functionalities would be useful (e.g., while performing a "Reorder" clicking on the up and down arrows will change the element's index).

Another requested improvement was the feedback on the current state of the operation. When the user starts an operation, he should choose the target element (which is highlighted in green). Some participants, especially those unfamiliar with the DOM structure, said they felt confused and didn't know what to do while performing certain operations. They said that some sort of feedback (e.g., adding a message saying "Select the desired element" while the user is selecting the target element) would be helpful in guiding them during the creative process.

Workflow Alternatives

Some participants suggested that having different workflow alternatives would improve the editing tool's usability.

To customize a web page, the user should open the popup, choose the desired operation and then choose the desired element to apply the selected operation. However, some participants suggested that they would prefer first to select the desired element and then choose the operation to apply to it.

Having different alternatives for the tool's workflow would make it more accessible for all types of users.

Challenges

There are some factors that have a big impact on the tool's usability. The biggest one is the website's DOM structure.

The DOM structure is key to determining the complexity of an adaptation. A simple task might be harder to implement on websites with a complex DOM structure.

As an example, some participants were asked to hide the thumbnails of YouTube's videos, which should require only one "Hide" operation for each video. However, YouTube's DOM structure has some overlays over the thumbnail that are used to handle click events. For this reason, more than one "Hide" operation is necessary for each video's thumbnail since each operation will hide an overlay and the thumbnail will only be accessible after all the overlays are hidden.

5.1.5.2 Customization for Self

The customization for self was mostly limited to the minimum required (3 adaptations per week). The following sub-themes will discuss the reasons and the experience of the participants regarding this theme.

Main goal

The main goal of most participants, when creating an adaptation, was to simplify the web page. The most used operations were the "Hide" and "Shortcut" to remove irrelevant information and improve the website's navigation.

Additionally, the "Change color" operation was frequently used to create new themes for web pages.

Help with the creative process

Some participants stated that some visual explanation while implementing an operation would be beneficial.

Including visual information to guide the user in the process of implementing an operation is something that was requested, as the participants mentioned that some operations were complex to implement, and opening the help menu every time that they start an operation wasn't feasible.

Operations and Code injection

As mentioned above, "Hide", "Shortcut" and "Change color" were the most used operations. The "Move" and "Reorder" have a different implementation process that was challenging for the participants to understand, which reflected a lot in their usage.

For this reason, some participants stated that, for more complex adaptations, there is still a need to inject Javascript or CSS into the web page.

Challenges

One of the participant's main challenges was the habituation. Most participants were able to complete the challenges presented to them in the first interview relatively easily. However, they stated that after a few days, during the second phase where the participants have to personalize at home, they forgot how some of the operations work (especially the "Move" and "Reorder" operations).

Another challenge was the amount of work required to create a more complex adaptation. The participants complained that each operation requires a considerable amount of effort to implement. This includes opening the popup, inputting the necessary data (e.g., choosing the desired font size, selecting the target margin, etc.) and clicking on the button to start the operation. Having to do this for every operation was a downside for users that wanted to create larger and more complex adaptations.

UI Creators vs Regular Users

In general, both types of users prefer to use the operations over the code injection. Even UI creators only resorted to code injection (mostly CSS) in situations where the task was too complex to implement using only operations.

Only one participant said that he would only need the CSS injector to create the adaptations.

5.1.5.3 Requesting Customizations

Not many users requested customizations. In fact, only one user created a request, and she stated that it was only to test the functionality. The sub-themes presented below present the reasons that lead to this outcome, according to the participants.

Self-confidence and Sense of control

One of the reasons not many participants requested customizations has to do with their capacities. The participants said that they wouldn't need the help of other people to customize a website because they were capable of doing it themselves.

Also, by customizing for themselves, they had full control of the outcome, ensuring that the result totally corresponds to what they want.

Amount of Work

Another reason for the lack of requests was the amount of work required to submit a request. According to the participants, unless the desired adaptation is very complex, the amount of work required to create it and the amount of work required to describe it are very similar, which was a discouraging factor to them.

5.1.5.4 Customization for Others

The participants had a positive experience responding to the requests. Most of them said that this process was the main reason that would make them keep our tool installed, as it allows them to help other people.

Impact of empathy

The feeling of helping others was the main motivation for the participants to respond to the requests sent to them. According to the participants, responding to the requests doesn't take much time and they felt good while doing it.

When it comes to the requester profile, although the participants liked to have that information (P10 mentioned that makes the request more personal), it didn't have much impact on whether the participant would respond to their request or not, but it helped them to choose which requests to prioritize. The participants would prioritize requests the most impactful requests (i.e., requests that would help people with some disability).

Long Term User Engagement

Although the participants enjoyed responding to requests, they felt the need for a mechanism to keep them engaged for long periods of time. Two different aspects were mentioned that would help keep users engaged: **Gamification** and **Feedback**.

A gamification system was highly suggested by the participants. Having a gamification system would help motivate the users in two factors: the challenge and emotional reward. With a star classification system, users would be recognized for their responses, motivating them to customize more and better. This type of system would also create a more competitive engagement on the requests, where users would aim to have the most and better-classified responses.

Having feedback on the responses was also highly asked. After replying to a request, the participants mentioned that they would like to be informed of the requester's opinion of their response, in order to keep them motivated.

Some users mentioned that a communication system would be beneficial for both the requester and the replier, not only to provide feedback but also to ensure that the response fully addresses all the requester's necessities.

Impact on customizing for self

Some participants had a hard time understanding the capabilities of our tool and didn't know where to start the customization process, so they used the requests as an inspiration for the adaptations that they would create for themselves.

Analysing the requests that they received gave the participants a more critical view of the websites and helped them explore functionalities of the tool that they wouldn't use otherwise.

5.2 Web Application

The second study aimed to explore and evaluate the functionalities of our web repository.

The objectives of this study are the following:

- Detect possible improvements on the website's usability.
- Obtain feedback on the different functionalities of the website.
- Obtain suggestions for functionalities that are missing on the website, or functionalities that might be interesting to explore.

5.2.1 Apparatus

This study uses our whole platform (as it is described in the chapter 4), but the main focus is on the web application.

Since this study already includes both components of our platform, we decided to use the original version of our platform, where the request system is implemented in the web repository (described in 4.4.3) and not on the editing tool.

5.2.2 Participants

Since the intention is to obtain feedback on our web application, we chose to invite a group of four UI creators to participate in this study since they have the best profile for detecting any aspect that can be improved.

Three out of the four participants had already participated in the previous study, so they were already familiar with the editing component of our platform.

Inclusion criteria:

- Being interested in participating in the study.
- Signing an informed consent form.
- Possessing advanced computer skills.

Exclusion criteria:

- Inability to participate or inability to sign an informed consent.

5.2.3 Procedure

Before starting, the participants are presented with the study description and informed consent.

After this, the participant participates in a recorded interview where he will be asked to complete a set of predefined tasks. Since we will be recording the voice and the screen during the interview, the participant will be encouraged to talk openly about his experience while he performs the requested tasks. The tasks completed during the interview are designed to guide the participant while using the web application, exploring all the different functionalities:

1. Login to the web application and download the editing tool.

2. Import an adaptation and start using it. Rate it and add a comment.

3. Customize an adaptation using another adaptation as a starting point.

4. Add an image to one of the user's adaptations and view the user statistics.

5. Search for an adaptation with a specific title.

6. Create a request.

7. Reply to a request.

After the interview, the participant will be asked to fill out the SUS (System Usability Scale) questionnaire.

5.2.4 Analysis

To analyse this study, we listened to the recordings of the interview and identified the themes that are relevant to address.

That being said, we have identified the following three themes: "Intuition", "Requests workflow" and "Privacy concerns".

5.2.5 Results

The results of this study were very promising. In general, the participants found the web repository simple and easy to use. However, there is always space for improvement, and the participants made some interesting suggestions.

5.2.5.1 Intuition

All of the participants mentioned some aspects that needed improvement in order to make the website more intuitive. Although this doesn't affect any of the website's functionalities, it might have some impact on its usage.

Among others, these were the main suggestions:

- The link to download our editing tool is located in the tab "Build" (see figure 4.9). Since this tab's only purpose is to download the editing tool extension, the participants suggested that it should be renamed to "Download".
- To match the initial color scheme, some buttons were colored red. The participants suggested that this might result in some confusion for the user because usually, the color red might suggest a negative action (e.g., delete, reset, etc.).
- The link to the user's adaptations page is located inside the user's profile page (see figure 4.10). One of the participants suggested that it would be better to have a tab that redirects the user directly to this page, instead of having to navigate to the profile page first.
- When asked to change the image of an adaptation (see figure **??**), some participants mentioned that it would be helpful to add some text to the button used to change the image of an adaptation (located in the top-right corner of the adaptation's image) since some users might not understand the purpose of the button simply by looking at the icon.

5.2.5.2 Requests workflow

Since three out of the four participants had already participated in the first study, they were familiar with our request system. However, the workflow of this system is a bit different in the final version of our platform (see section 4.4.3).

One of the participants said that the new workflow for the requests system should take more advantage of the community. Instead of one user addressing a request at a time, multiple users should be able to address the same request at the same time, resulting in a list of responses being submitted for one request. The requester would then choose the response that satisfies him the most, and verify the request, declaring it as complete.

This approach would reduce the time required to complete the request since many responses are being submitted at the same time.

5.2.5.3 Privacy concerns

As expected, data privacy is a major concern for most users. During the interview, the privacy of the created adaptations was one of the aspects that the participants were most

interested in.

Since our web repository allows using public adaptations as a baseline for further customization, the participants were concerned that the privacy aspect of the adaptations might be compromised. One of the participants asked an interesting question:

"If a user replies to a request with a private adaptation and the requester downloads the response, the private response is now visible to the requester. If the requester then further customizes the response, sets the adaptation to public (see figure 4.8) and saves it as a new adaptation, wouldn't that compromise the privacy of the original response?"

In fact, this scenario would compromise the privacy of the adaptation submitted as a response, since its new version would be available in the web repository, exposing the contents of the original private adaptation that was used as a baseline.

5.3 Discussion

The feedback obtained from both studies gave us a better understanding of our platform's acceptance, and identified several aspects that can be improved in the future.

Our platform covers two different areas (similar to the ones identified in section 2) that will be discussed separately, in order to identify the positive and negative aspects of both areas.

5.3.1 UI Personalization

Looking at the customization process, we have identified several aspects that can be improved.

The main concern is overcoming the limitations imposed by the website's DOM structure. Complex DOMs are prone to generate unexpected behaviours during the customization process. Youtube's website is a good example, as the video thumbnails possess several invisible containers on top of them to trigger events. Since the element containing the thumbnail is behind all these containers, it is impossible to apply any operation on it because the user can't click on it to select it as the target for the operation, leaving the code injection as the only option to customize elements in these types of situations, which not all users are able to implement.

The customization process is another aspect that needs to be improved. While some participants mentioned that they had a hard time remembering how to implement some of the operations alone, others suggested different workflows to implement the different operations, implying that the current one wasn't very friendly. To improve the user's experience, we could add visual explanations that would be visible during the customization process, as well as allow the user to choose between different workflows for applying the

different operations.

In a production environment, our editing tool would need some improvements in order to become more user-friendly. With the platform's current state most participants limited their customizations to removing unnecessary information and creating themes, but with a more friendly workflow, users won't be so reticent when it comes to exploring the different functionalities of our editing tool.

5.3.2 Collaborative/Crowdsourced User Interfaces

The social aspect of our platform has received very positive feedback.

The participants were highly motivated to respond to the requests sent to them, not only because it made them feel good about themselves, but also because they can get inspiration and a more critical view of the adaptations that they create for themselves.

During the first study, the participants suggested the existence of a gamification and feedback system, as it would inform them of the requester's opinion regarding their work, and keep them motivated in the long term. This type of system is already present in our web repository, where the users are able to rate the adaptations and see the number of people who installed them. There is also a discussion section for both the adaptations and the requests, where users can comment to provide feedback.

Data privacy is also an important aspect to take into account in a social platform like ours. Private data should always stay private, and during the second study, the participants detected situations where data privacy would be compromised. This is a topic that should be addressed with urgency since users will lose their trust in our platform if their private data gets exposed.

Taking into account the feedback provided, we can conclude that the social aspect of our platform is a success. The users enjoy helping other users in need, in fact, some might only use our platform to help the community, as mentioned by one of the participants of the first study.

Chapter 6

Conclusions

This dissertation is a first step to shift the agency of UI adaptation from developers to users. To achieve this, we developed a platform that allows users to create customized interfaces and share them with the community.

Our platform is divided into two components: the Editing tool and the Web Repository. The former is responsible for the creation and sharing of UI adaptations, while the latter is responsible for displaying the shared adaptations, allowing the other users to download, use them and further adapt them.

To evaluate our platform, we performed two studies. The first study not only evaluates the usability of our editing tool but also tries to understand how the users would respond to this type of platform (more deeply explained in the article[3]). The second study evaluated the usability of our web repository.

Taking into account the feedback obtained from both studies, we can conclude that our platform is highly accepted, although there are some aspects that need to be taken into account in order to maximize its potential.

Most participants prefer to customize for others rather than for themselves, which is an optimal behaviour for our platform because the focus should be on helping users with special necessities that are not able to customize for themselves, relying either on the adaptations that are available in our web repository or on our request system.

An aspect that is crucial to keep users engaged in the long term is the gamification system. This type of system was highly requested during the first study and, during the second study (where this system was already present), the feedback was very positive. Since each adaptation has statistics and a rating attached to it, the users will be motivated to create the best adaptations possible, in order to achieve the highest rating possible.

6.1 Summary of Contributions

This project has made the following contributions:

1. An editing tool implemented as a *Chrome* extension that allows users to create and save changes to the interfaces.

2. A web application that acts as a web repository for searching and sharing customized interfaces.

3. Two studies that evaluate the acceptance and benefits of UI customization for self and for others.

This platform was subject to two different studies. Each study was focused on each component of the platform, with the purpose of evaluating the tool's usability and acceptance.

The first study, which is focused on the editing tool, also resulted in the collaboration in the development of the article [3] that will be available later this year.

6.2 Limitations

There are some limitations to our platform, especially regarding the editing tool's user interaction.

When performing an operation, the website's DOM is a very important factor. More complex DOMs are harder to customize. This limitation can be perceived, for instance, while performing a "Move" operation. After selecting the element that will be moved, choosing the right parent to place the element in its new location can be very hard if the web page has a lot of elements and a complex structure.

The fact that the editing tool is implemented as a browser extension requires the user interaction to be performed via a popup. For complex adaptations that use large amounts of operations, a large amount of clicks is required, since each time the user wants to start an operation, he needs to open the popup and click on the desired button to start the operation.

On the other hand, the web application does not have any considerable limitations. Its main purpose is to display the data created in the editing tool, which was relatively simple to implement due to the combination of *React's* reusable components and Firebase's integrated SDK.

6.3 Future Work

The future work for this project includes fixing the usability problems regarding the editing tool's user interaction, but there are some other topics that would be interesting to explore in order to improve our platform. Among others, these are the main topic that would be worth exploring in the future.

6.3.1 Solving usability problems

- Some operations require some input to be implemented (e.g., before starting the operation "Font-size" the user needs to select the desired font size in pixels). In order to reduce the amount of clicks and time needed, the editing tool could save the previous values for these types of inputs, so that the user does not need to provide them again if he wants to implement the operation again with the same values.
- Add visual cues (e.g., tutorials, controls, etc.) for each operation.
- Create new controls for selecting the desired DOM element, overcoming the limitations imposed by complex DOMs.
- Improve the intuition issues in the web repository to match the feedback obtained from the second study (e.g., changing the colors of the buttons, renaming the "Build" page to "Download", etc.).

6.3.2 Possible Improvements

- Create different workflows for the editing tool, giving the user the choice to use the best fit for him.
- When a website updates its DOM structure, some changes in the adaptations may result in an error. Currently, the user is alerted of this error, but in the future, our platform might be able to solve this type of errors by itself.
- Currently the editing tool is limited to the browser *Google Chrome*, which is a limitation for users who use other browsers. Developing the extension for other browsers would extend the number of users who are able to join our community.
- Implementing an option to select a whole group of DOM elements as target for an operation would reduce the number of clicks required to implement the same operation in all elements.

Bibliography

[1] P. Akiki, A. Bandara, and Y. Yu. Crowdsourcing user interface adaptations for minimizing the bloat in enterprise applications. In *Proceedings of the 5th ACM SIGCHI Symposium on Engineering Interactive Computing Systems*, EICS '13, page 121–126, New York, NY, USA, 2013. Association for Computing Machinery. ISBN 9781450321389. doi: 10.1145/2494603.2480319. URL https://doi.org/10.1145/2494603.2480319.

[2] P. A. Akiki, A. K. Bandara, and Y. Yu. Rbuis: Simplifying enterprise application user interfaces through engineering role-based adaptive behavior. In *Proceedings of the 5th ACM SIGCHI Symposium on Engineering Interactive Computing Systems*, EICS '13, page 3–12, New York, NY, USA, 2013. Association for Computing Machinery. ISBN 9781450321389. doi: 10.1145/2494603.2480297. URL https://doi.org/10.1145/2494603.2480297.

[3] S. Alves, R. Costa, K. Montague, and T. Guerreiro. Investigating citizen-led personalization of user interfaces: For self and others. 2022.

[4] P. Blanck. The struggle for web equality by persons with cognitive disabilities. *Behavioral sciences & the law*, 32, 03 2014. doi: 10.1002/bsl.2101.

[5] M. Collins. Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP 2002)*, pages 1–8. Association for Computational Linguistics, July 2002. doi: 10.3115/1118693.1118694. URL https://aclanthology.org/W02-1001.

[6] D. R. Compeau and C. A. Higgins. Computer self-efficacy: Development of a measure and initial test. *MIS Quarterly*, 19(2):189–211, 1995. ISSN 02767783. URL http://www.jstor.org/stable/249688.

[7] K. Z. Gajos, D. S. Weld, and J. O. Wobbrock. Automatically generating personalized user interfaces with supple. *Artificial Intelligence*, 174(12):910–950, 2010. ISSN 0004-3702. doi: https://doi.org/10.1016/j.artint.2010.05.

005. URL https://www.sciencedirect.com/science/article/pii/S0004370210000822.

[8] A. Garbett, R. Comber, E. Jenkins, and P. Olivier. *App Movement: A Platform for Community Commissioning of Mobile Applications*, page 26–37. Association for Computing Machinery, New York, NY, USA, 2016. ISBN 9781450333627. URL https://doi.org/10.1145/2858036.2858094.

[9] D. Karger and D. Quan. Prerequisites for a personalizable user interface. 02 2004.

[10] R. Kumar, J. O. Talton, S. Ahmad, and S. R. Klemmer. Bricolage: Example-based retargeting for web design. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, page 2197–2206, New York, NY, USA, 2011. Association for Computing Machinery. ISBN 9781450302289. doi: 10.1145/1978942.1979262. URL https://doi.org/10.1145/1978942.1979262.

[11] M. Nebeling and M. C. Norrie. Tools and architectural support for crowdsourced adaptation of web interfaces. In *Proceedings of the 11th International Conference on Web Engineering*, ICWE'11, page 243–257, Berlin, Heidelberg, 2011. Springer-Verlag. ISBN 9783642222320.

[12] M. Nebeling, M. Speicher, and M. C. Norrie. Crowdadapt: Enabling crowdsourced web page adaptation for individual viewing conditions and preferences. In *Proceedings of the 5th ACM SIGCHI Symposium on Engineering Interactive Computing Systems*, EICS '13, page 23–32, New York, NY, USA, 2013. Association for Computing Machinery. ISBN 9781450321389. doi: 10.1145/2494603.2480304. URL https://doi.org/10.1145/2494603.2480304.

[13] A. Pekpazar, R. Öztürk, and C. Altin Gumussoy. *Usability Measurement of Mobile Applications with System Usability Scale (SUS): Selected Papers from the Global Joint Conference on Industrial Engineering and Its Application Areas, GJCIE 2018, June 21–22, 2018, Nevsehir, Turkey*, pages 389–400. 01 2019. ISBN 978-3-030-03316-3. doi: 10.1007/978-3-030-03317-0_32.

[14] M. d. Q. Proença, V. G. Motti, K. R. d. H. Rodrigues, and V. P. d. A. Neris. Coping with diversity - a system for end-users to customize web user interfaces. *Proc. ACM Hum.-Comput. Interact.*, 5(EICS), May 2021. doi: 10.1145/3457151. URL https://doi.org/10.1145/3457151.

[15] B. Shneiderman. Promoting universal usability with multi-layer interface design. *SIGCAPH Comput. Phys. Handicap.*, (73–74):1–8, June 2002. ISSN 0163-5727.

doi: 10.1145/960201.957206. URL https://doi.org/10.1145/960201.957206.

[16] H. Takagi, S. Kawanaka, M. Kobayashi, T. Itoh, and C. Asakawa. Social accessibility: Achieving accessibility through collaborative metadata authoring. Assets '08, page 193–200, New York, NY, USA, 2008. Association for Computing Machinery. ISBN 9781595939760. doi: 10.1145/1414471.1414507. URL https://doi.org/10.1145/1414471.1414507.

[17] L. Vermette, S. Dembla, A. Y. Wang, J. McGrenere, and P. K. Chilana. Social cheatsheet: An interactive community-curated information overlay for web applications. *Proc. ACM Hum.-Comput. Interact.*, 1(CSCW), Dec. 2017. doi: 10.1145/3134737. URL https://doi.org/10.1145/3134737.