Digitization of Swedish Caribbean Colonial History

Georgios Davakos
Rewa Abou raas

Institutionen för informationsteknologi
Department of Information Technology
Abstract

Digitization of Swedish Caribbean Colonial History

Rewa Abou raas, Georgios Davakos

Digitization of Swedish Caribbean Colonial history is part of a research project that works on digitizing historical documents and artifacts from the Swedish colonial timeline of 1784-1878. This thesis aims to improve the accessibility of the digitized artifacts, improve the presentation of the artifacts, and provide more information about the digitization project. This can be achieved by creating a web application where users can search for the different historical artifacts as well as find more information about the ongoing digitization. The thesis can be divided into two parts, a frontend that focuses on building presentation layouts in the form of static websites and studying the effects of usability principles on the design of the application by evaluating it. The second part is the backend that mainly looks to compare the use of monolithic architecture and microservices architecture, this is achieved by either bundling up the different components of the backend into a single system or by dividing them into multiple smaller services capable of communicating with each other through HTTP requests.

The result of the thesis is a functional web application that follows Jakob Nielsen’s design heuristics for the user interface. It was concluded that the web application was perceived as more user friendly when it followed Jakob Nielsen’s design heuristics and any shortage of usage could be directly linked to the heuristics and improved by following them more precisely. When it comes to the backend, the monolithic architecture proved to be better for demonstrating the web application.
## Contents

1 Introduction ........................................... 5  
   1.1 Purpose and Goals ..................................... 5  
   1.2 Methodology ........................................ 6  

2 Background ........................................... 6  
   2.1 General Background .................................... 6  
   2.2 Related Work .......................................... 6  

3 Theory ................................................ 7  
   3.1 Frontend ............................................... 7  
      3.1.1 Websites builder .................................... 7  
      3.1.2 Prototype ........................................ 7  
      3.1.3 Design ........................................... 8  
      3.1.4 Evaluation ........................................ 9  
   3.2 Backend ............................................... 10  
      3.2.1 Database .......................................... 10  
      3.2.2 Monolithic Architecture ............................ 11  
      3.2.3 Microservice Architecture .......................... 11  
      3.2.4 Database Search Engine ............................. 11  
      3.2.5 Set Theory ....................................... 12  

4 Method ................................................. 12  
   4.1 Building the Frontend .................................. 12  
      4.1.1 Prototype ......................................... 12  
      4.1.2 Implementation .................................... 13  
      4.1.3 Evaluation ........................................ 14  
   4.2 Building the Backend .................................. 14  
      4.2.1 Designing a Microservice Architecture ............... 14  
      4.2.2 Building the Database ................................ 14  
      4.2.3 From Filemaker Pro to SQL .......................... 15  
      4.2.4 Building the Database Search Engine ................. 15  

5 Result ............................................... 15  
   5.1 Web Application ....................................... 15  
   5.2 Evaluations results ..................................... 18  
   5.3 The Database .......................................... 19  
   5.4 The Microservice Architecture .......................... 20  
   5.5 Converting Filemaker Pro database to SQL ................. 20  
   5.6 The Database Search Engine ............................. 20  
   5.7 Login Service ......................................... 21
6 Discussion
6.1 Website builder .......................................................... 22
6.2 Prototype ................................................................. 23
6.3 Evaluation ................................................................. 23
6.4 Microservice or Monolithic Architecture ....................... 24
6.5 The SQL Database ..................................................... 25
6.6 The Shortcomings of the Database Search Engine ............. 25
6.7 Sorting the Data in the Search Page ............................... 25
6.8 The Login Service ..................................................... 26

7 Conclusion ............................................................... 26

8 Future Work ............................................................. 26
8.1 Frontend ................................................................. 26
8.2 Backend ................................................................. 27

References ............................................................... 28
1 Introduction

In 1784, a trade was made between France and Sweden. Sweden was given St. Barthélemy while France gains trading rights in the Swedish port of Gothenburg. At the time, 281 slaves lived on the island [1].

This period of the history of Sweden is not well known for several reasons. Fredrik Thomasson, a researcher at Uppsala University who has researched for almost ten years into Sweden’s colonial history [2], made an interview with the local news [3]. In his interview he mentioned several reasons as to why this part of Swedish history isn’t commonly found in Swedish history books, some of those reasons are:

- From an economical perspective, the colony was no great success and has not garnered popular attention.
- Sweden’s colonial past has been difficult to reconcile with its modern self-image as a philanthropist and progressive state.
- No extensive research, until very recently, has been carried out on this topic.

A Ph.D. thesis by Victor Wilson states that around 10,000 African captives were transported on Swedish ships [4]. The impact of the Swedish slave trade on today’s society, particularly its connections with contemporary anti-black racism in Sweden, has been a subject of debate. That is why it is important to know more about this case so that Sweden can identify the impact of this history on the country [5].

1.1 Purpose and Goals

The purpose of this paper is to build a web application where the digitised documents from this timeline can be accessed. The application is required to have a feature that is allowing the user to search through these documents. Finally, the application must also be capable of converting a Filemaker Pro database to an SQL database.

The goals are:

1. To design and evaluate a web application and study the effect of usability principles on the design.

2. To compare a monolithic architecture to a microservice architecture.
1.2 Methodology

This project is divided into two parts:

1. Backend development.
2. Frontend development.

The work on each part had to be performed in parallel, each group member was assigned to work on one part. Once each part was completed the members of the team had to work together to merge the frontend with the backend.

2 Background

2.1 General Background

In the year 1784, Sweden was under negotiations with France to form a military alliance resulting in Sweden taking possession of the Caribbean island of Saint Barthélemy and in exchange France received trade rights in Gothenburg [1]. The possession of the island was an attempt by Sweden to expand its overseas trade by means of a Caribbean colony. Saint Barthélemy was under Swedish occupation for nearly one century before it was handed over back to France [6].

Most of the historical archives haven’t been available to the public but a successful digitization project from 2011-2016 has opened up the research to the ”Swedish governmental archives” (300 000 pages) from within the ”French colonial archives” [6]. This has led to a research project by the name ”Swedish Caribbean Colonialism 1784-1878. Integrating, classifying, publishing, and investigating dispersed Swedish colonial archives”, with the shorter name for this project being ”SweCarCol”. The project is part of the research program DIGARV that is based on a government assignment to promote so-called ”data-driven research” [7]. This implies that research is conducted on digital materials that as of now are mostly in physical form.

2.2 Related Work

An ongoing project is Alvin, a platform built and maintained by Uppsala University Library in an attempt to preserve Swedish historical artefacts and documents in digital form. The goal of the project has been to unify many different library systems that
were used in the past into a single platform that was accessible by the public [8,9]. This project proved successful and launched on the 28th of November, 2014, with over 1500 records.

3 Theory

3.1 Frontend

Frontend refers to the part that is visible to the user, not necessarily everything in detail but the result of it. It can refer to a web application, a mobile application, or even a desktop application. In this paper, the frontend refers to the web application.

Building the structure of a web application is usually done using HTML: HyperText Markup Language, HTML5 is used in this project, and styling it is done with CSS. Making the web application functional and specifying the action of its content was done using JavaScript programming language.

3.1.1 Websites builder

Many software companies offer cloud-based web development services. This service is quite popular nowadays because of its simplicity. A business owner can simply design a web application by easily placing a page header, text area, and more basic options with visual help without any programming knowledge [10]. This service also provides cheap publishing options. Shopify¹ and Webflow² are two examples of popular website builders.

3.1.2 Prototype

A prototype is a hypothesized design of the required application. Prototypes are done to confirm understanding of the requirements and enable early discussions and feedback [11].

A good prototype can be interacted with to visual functionality, this doesn’t necessarily mean that it should be done digitally. In a matter of fact, there are many ways to do a prototype. Different types of prototypes can be classified according to their speed,

¹https://www.shopify.com/
²https://webflow.com/
fidelity, or purpose. Wireframes, Post-it prototype, paper prototype, wizard of oz, and mock-ups are examples of a prototype [11].

Prototypes can also be divided into two types, interactive and static prototypes. Interactive prototypes should include the response of each action before the test. On the other hand, in static prototypes, the interaction is fake [12].

Wizard of Oz and paper prototype are examples of static prototypes, it requires the designer to act as the computer and move the prototype view according to an action done by a user. While both of them are statics, wizard of Oz requires software equipment while paper prototypes can be done on paper without any implementation.

3.1.3 Design

**UX**  Stands for user experience, and indeed it focuses on how the user feels while interacting with the interface of a company or a project. An exemplary user experience must ensure that the user can use the system without problems and enjoy using it [13].

**UI**  Stands for User Interface, and it focuses more on the appearance of the product, a good UI design must be:

Accessible, Usable, Socially, and economically acceptable [14]

**Accessibility:**  The product should be accessible by as many users as possible, including people with disabilities. The aspects that are covered by this paper are:

- Physical: The web application should be accessible from as many web devices as possible, some methods were used to make the website readable if this option is available.
- Conceptual: Complicated instruction should be avoided.
- Cultural: The project design must avoid any metaphors that can be sensitive.

**Usability:**  The ISO standard defines usability as:
"The extent to which a system, product or service can be used by specified users to achieve specified goals with effectiveness, efficiency, and satisfaction in a specified context of use." [15]

The issues in this topic can be solved from two different points of view. First, the web application should be functional, this means that it should include all the functions required by the project owner. Developers may also be ready for any changes and keep the design flexible so that the addition or removal of any function should not cause the interface to crash or become faulty. Another aspect is the usability of a web application, this aspect, academically, is studied on the human-computer interaction level as well as system design with a user perspective.

Jakob Nielsen is the author of the usability checklist 10 Usability Heuristics. Those rules are for general use and are not specific instruction, that is why they are called "heuristics" [16]. Jakob Nielsen’s design heuristics are followed in this project to deliver a usable product. More about design heuristics can be found in method, section 4

Acceptability: Product acceptance by Nielsen is:

"Usability together with utility is considered to influence the usefulness of the product. Usefulness is one of the attributes affecting acceptability [17]."

Acceptability can not be evaluated in the same way as usability because acceptability varies according to the circumstances of using the product. Yet, developers may follow some features to build an acceptable product.

Political: The product must not break the rule of the (company/government etc.) Cultural and social: When interacting with a human, it is important to know your audience, their culture, and tradition and avoid things that may disturber the community. Usefulness and economic.

3.1.4 Evaluation

Formative evaluation can be done along with the implementation process to evaluate individual functionality and usability. At the end of the project, summative evaluation is important to evaluate the final product by making tests on it as some users try it.

The evaluation process can be done by developers and is commonly known as the expert evaluation or by other people.
The test aims to evaluate the user’s overall experience using the website as well as focusing on the details.

- During the evaluation, testers focus mostly on how the user feels while using the product, many things help so that users feel safe while using the product and it is everything from colors to construction. and how easy it is to navigate through the website and reach their goal without.

- The evaluation must let the user find their way through the website and observe how easily the user can perform a task without guiding them through the process.

- Evaluations can reveal some important points that a developer may have forgotten about completely.

In short, usability evaluation allows the developer to see what the product looks like from the perspective of others and identify problems developers have not discovered before.

### 3.2 Backend

While the frontend is the part of a website or software application that the user directly interacts with, the backend is the part the user doesn’t interact with. Developers will sometimes refer to the backend as the "data access layer" [18], as the name suggests, the backend facilitates indirect communication between the user and a database. A practical example of this can be found on a website such as facebook\(^3\) where the contents of users’ profiles are stored inside a database. Whenever a user attempts to watch someone’s profile the frontend makes a request to the backend which in turn gathers the profile information found inside the database and hands it over to the frontend who then displays it to the user.

Before building the backend it is important to consider different architectures, each suited more than the other dependent on the demands. In this paper, only the architectures that were found to be relevant are being explored.

#### 3.2.1 Database

The database is an organised structure of data [19]. This data is managed by a *database management system* that can create, edit, read or delete data in the database. There are two types of database management systems, SQL and NoSQL. This paper will only focus on SQL.

\(^3\)facebook.com
What is SQL
Structured Query Language or more commonly known, SQL, is a collection of programming languages that are used to create, edit, read or delete data from a database. What all of the SQL-languages have in common is that they are used to form so called relation databases [20]. It should be noted that SQL-languages share a majority of the syntax. Each SQL-language is paired with a database management system, e.g. MySQL or PostgreSQL. When referring to a database management system it is common to just refer to the SQL-language that the system is using.

3.2.2 Monolithic Architecture

A monolithic architecture is typically a single program where the components of the program are interconnected and tightly-coupled. What this means is that the entire codebase of the program is compiled into a single file. This type of architecture is often-times chosen because of how little time can be spent planning while more time can be spent writing the code for the application. Deployment and distribution of the application are also quite simple since you are only responsible for a single file [21].

3.2.3 Microservice Architecture

Contrast to a monolithic architecture, a microservice architecture is based on the idea that an application can be divided into small modules, each responsible for a single task, e.g., a login module or an account creation module. In this type of architecture, the modules are referred to as services [22]. The microservice architecture forms a network of services that can communicate with each other by using HTTP requests which means that each service is not required to be running on the same device.

3.2.4 Database Search Engine

A database search engine is an application that allows the user to search for information stored in a database by using a text based query [23]. Full-text search and the search of specific words are examples of text based queries.

One implementation of a database search engine is to create an application that interacts with a database management system in order to retrieve all entries in the database so they can be indexed by the database search engine application. Indexing the data eliminates the process where the application has to retrieve all the data from the database and then attempt to match it with the search query. Instead, by indexing the data it is possible to decrease the time it takes to perform a search. The downside to this approach
is that every time the data in the database is altered, either by introducing new data, removing existing data or editing the existing data, then the application will need to index the database once again. It should be noted that indexing strategies are outside the scope of this paper.

3.2.5 Set Theory

Set theory is a branch in mathematics that studies well-defined collections, called sets [24]. What makes a set well-defined is the trait that is chosen to define the set, e.g. people who fall in the age group 35-40 years old. An important property of sets is that an element or object inside a set can only be found once. While this implies that every element in the set is unique it doesn’t prevent an element to belong to one or more sets, e.g. a person who is 35 years old belongs both in age groups 35-40 and 30-35, while someone aged 34 only belongs to age group 30-35. A more academic way could summarise this to, "when two well-defined sets intersect each other, the elements found in the intersection exist in both sets”.

4 Method

4.1 Building the Frontend

building the frontend went through three phases, the prototype phase, the implementation phase, and the evaluation phase.

4.1.1 Prototype

The prototype of this project is a paper prototype because it is quick and cheap to make which means it is easy to change some details. It also contains all the main features.

The prototype was made according to the requirements and included multiple pages, to visualise the interaction between pages and the functionality of features an interactive video was made. The designing process also took into consideration UX and UI methodologies.

One of the features was a header, the header is a navigation tool that is shown on all pages and navigates the user in two ways, it gives the user a first look at the page content and helps them go to a specific page.
The functionality of the page is mostly on the Search page, that is why the search process needed to be represented accurately in the prototype.

### 4.1.2 Implementation

The web application started with multiple pages, one for each subject. The following 10 Nielsen’s and Molich’s heuristics for User Interface Design were followed to ensure the best usability [16].

1. **Visibility of system status:** SweCarCol’s biggest functionality is to search for results in the database and upload files to the database. System status should be visible to the user in case they wait for the results.

2. **Match between system and real world:** This can be achieved by using symbols and metaphors that are as near to the real life as possible.

3. **User control and freedom.**

4. **Consistency and standards:** By using physical consistency between all pages, and conceptual consistency in using words, the web application will give the user a sense of familiarity.

5. **Error prevention:** is used in places where errors can cause harm to the web application or user.

6. **Recognition rather than recall:** This helps the user to use the pages and functions easily and without having to remember small details.

7. **Flexibility and efficiency of use:** The web application users may vary. The implementation process must take into consideration that users’ needs are not always so advanced and an easier solution must be provided.

8. **Aesthetic and minimalist design:** More is not always better, in order to make it easy for the user to access information, the web application should follow simple headlines and design.

9. **Help users recognise, diagnose, and recover from errors**

10. **Help and documentation:** In case the system need explanation, it is necessary to provide simple documentation so that users understand how to use the system.
4.1.3 Evaluation

An expert formative evaluation was regularly done to measure the usability and functionality during the development process, the usability of the product is measured by comparing the product with Nielsens’s and Molich’s design heuristic [25]. Furthermore, a participant summative evaluation took a place at the end of the project. People that represent the end-user tried to use the web application [26].

The tools used in the evaluation process were: three participants and five different scenarios. According to Nielsen, this process can find up to 50 percent of the usability problems that can be found [27].

Participants were given the scenario and asked to think aloud while testing the application and performing the tasks. This process gives important information about the feeling of the user as well as how fast they can perform a task and how easy it is to know what they should do next [28].

4.2 Building the Backend

The backend had to be divided into two parts, a design phase for the microservice architecture followed by an implementation phase.

4.2.1 Designing a Microservice Architecture

The microservice architecture was built with the goal of making each component as loosely coupled as possible [29]. Ideally the frontend is the only one communicating with each service. With these requirements out of the way the only thing that is left is to figure out all the necessary services for this application, draw the entire architecture and finally to start working on each service.

Even though a microservice architecture makes it possible to use different programming languages, each best suited for the implemented service, this paper doesn’t take advantage of that and instead has all the services built with java.

4.2.2 Building the Database

An SQL-database was built, while an already existing Filemaker Pro database was used as a reference. The number of tables and columns as well as column names were copied
directly from the Filemaker Pro database and inserted into the SQL database. The data (rows) inside the Filemaker Pro database were not copied and pasted into the SQL database.

4.2.3 From Filemaker Pro to SQL

The web application needs to be capable of converting a Filemaker Pro database into an SQL database. This is achieved by converting the existing Filemaker Pro database into a CSV-file and then have it be uploaded into the SQL database through the help of the web application.

4.2.4 Building the Database Search Engine

Once the SQL database that is meant to host all the data from the Filemaker Pro database has been designed, with all the tables and column name, then it is possible to proceed with the database search engine. The database search engine service is implemented using the "hibernate search" library [30]. During the implementation the columns that should be searchable have to be identified. Once that is done then it’s possible to implement the search endpoints.

5 Result

5.1 Web Application

The web application pages came as a result of several design decisions and Nielsen’s and Molich’s heuristics for User Interface Design. The following section is going to represent the website with the used methods, how, and where each method was used.

Navigation

Navigation is important to the accessibility of the page through viewing the parts that are not showing on an individual page, see figure 1.

Figure 1: The navigation bar

It is also important to highlight the part where the user is, so the user at all times knows
which page they are on, and this process increases the usability. Hierarchical drop-down menus are also avoided for the same reason since hierarchical drop-down menus often become problematic for the user when they become long. However, a drop-down menu is used in the responsive version of the page. see figure 2.

This menu is only shown when the user clicks on the menu logo. Menu logo is universal and easy to recognize, a new logo may confuse the user and are not necessary on this web application. this is done so that the design follows the rule of *Match between system and the real world:* Symbols in this web application match the common symbols, such a symbol is used the show the menu when the web application is used from a smaller device. another example is in the sort by menu, users can sort results Ascending/Descending. An arrow is shown to represent the current method used.

![Figure 2: The responsive navigation bar](image)

**Body**

*Consistency and standards:* Same colors, sections, and placement of elements were used in all pages to maintain a physical consistency. When it comes to conceptual consistency, it can be most recognized in searching in advanced search, categories for example have the same names as in the result.

*Aesthetic and minimalist design:* pretty simple design is used to make using the page easier for the user.

**Search page**

*Visibility of system status:* search functionality gives results quite fast and no need for system status to be shown in this case, however uploading files is more complicated since there is no current method that gives information about when this process is done, it is hard to give the user an accurate system status. A message can be shown for a few seconds to inform the user that the file is uploading.

*Error prevention:* searching for results that don’t exist or trying to login to an account that does not exist results in an error message and stopping the process in the backend,
this helps the user to understand what is going wrong as well as protecting the website.

Recognition rather than recall: Result card contains important information, sometimes it is hard to recognize what each word refers to and that is why the type of information is shown.

Flexibility and efficiency of use: searching for results can be done using a free word search without any advanced setting, or with an advanced search option, not all options must be filled. Instead, users can choose how advanced the search is by simply filling the desired searching option bar.

Help and documentation: Choosing the right category is challenging for new users who do not know what are the available category. That is why a list with categories name was implemented.

Figure 3: The Advanced search options
5.2 Evaluations results

The evaluation process was informative, three participants tested the web application and gave feedback on many points, those points were even connected to the following points of Nielsen’s usability heuristics for user interface design:

- Visibility of system status: When a search is done, the search word disappeared from the search field, this made the user unsure whether the search was done or not. The results came quite fast so that was not a big problem when there were results. The problem was more obvious when there were no results.
• Match between system and the real world: The word "Documentation" is used in navigation to refer to a piece of information about the project. Feedback about this word shows that it made the user confused about the content of this section, some even thought that it refers to a documentation of the database. Another aspect is the category value, participants were not exactly sure how the result will look like, this result happened mainly when a new user tries to use the advanced search, it can be solved if the user started with a simpler search process.

• Consistency and standards: An important factor for inconsistency is language consistency, a participant highlighted the difference in the language between the results of the search and the language used in the web application. This is a result of the different languages used in the database and the required language of the web application.

• Flexibility and efficiency of use: Admin functionality is currently limited to uploading files. However, a flexible system can even provide more functionality that can be used in the future development of the application. Important functionality for the admin can be to control the uploaded file and be able to see the date of the upload process through a search function. The last part can also be useful for the participant in case they would like to know if there are any new documents in the application.

• Help and documentation: The existence of a category name makes it easier for the user to understand what kind of documents can be shown in results.

Navigating between different parts and pages was done without any problems, users could achieve their goal in a matter of seconds when it comes to changing the view. Searching for results took up to three minutes, that is because the advanced search process includes many options.

Participants in all five scenarios were able to achieve all required tasks successfully without instruction from our side, except when it comes to uploading a file which is not possible for a participant to do remotely.

5.3 The Database

The database was built using MySQL. The Filemaker Pro database had a single table which was copied over to the MySQL database. Due to privacy reasons the column names, the number of columns, and data types are not included in this paper.
5.4 The Microservice Architecture

The microservice architecture consisted of 3 services, the search engine service, the upload service and a login service. The frontend and the discovery server are not considered to be services.

Figure 6: The microservice architecture

5.5 Converting Filemaker Pro database to SQL

The "Upload Service" is converting the Filemaker Pro database to an SQL database. To be more precise, the service requires the user to upload a CSV-file that represents the Filemaker Pro database, then it proceeds to verify that the uploaded file is a CSV-file. If the file is a CSV-file then the service proceeds to add the data into the SQL database.

5.6 The Database Search Engine

The database search engine is the "Search Engine Service" found in figure 6. This service has three endpoints, simple search, advanced search and sorted search.

Simple search
The simple full-text search is able to take an entire sentence as input and proceed to match it with data inside the SQL database.
Advanced search
The advanced search extends the simple search by also taking in arguments for specific columns found in the SQL database, e.g. search for the word "religion" in the "category" column. When implementing advanced search it was to be defined as a search query with multiple search options, e.g. a free text search, selecting a category and a year. Set theory can be applied by treating an individual search (e.g. free text search) as a set. By doing so, it would be possible to model each option in the advanced search as an individual search. The search result from each search option found from an advanced search would be a set where the elements of interest are in the intersection between all the sets as displayed in figure 7.

![Diagram of set theory applied to advanced search]

Figure 7: A diagram where each search is represented by a circle.

Sorted search
Finally the sorted search extends the advanced search with two additional inputs, one for a column and the other for the sorting order, either ascended or descended. The only difference to advanced search is that the resulting data is sorted by the column the user specified.

5.7 Login Service

A simple "Login Service" was built for the purpose of simulating the use of login so someone could upload a CSV-file. The service has 2 endpoints, an authentication and a validation endpoint.

When a user wants to upload the Filemaker Pro database the user has to go to the login page and provide their login credentials. The authentication endpoint is used to verify that the credentials are correct and if the login is successful then a JSON web token is provided to the user who just logged in [31]. The second endpoint is for validating the JSON web token when the user tries to access pages were they need authorisation.
6 Discussion

6.1 Website builder

Even though website builder is a fast tool and easier than developing a web application using HTML, CSS, and JavaScript. Here are some reasons why websites builder should be avoided:

- Designing a web application is not only about how it will look at the end. A big part of any project is the code behind it, and unfortunately, the code behind the nice web application developed by website builders is often hard to read through or change.

- Furthermore, web applications that are built by websites builder load often slowly because they often use extra sections and put sections inside of another which make the page slower, and their functionality may not work in some browsers.

- **Bad SEO:** SEO stands for search engine optimisation, it is the procedure that makes a web application show higher in a search engine like Google. A web application that is built with a website builder often has bad SEO. one of the reasons for that is: **HTML Errors**, not using a valid HTML language and tags leads to lower SEO. Websites builders may choose to have a nice-looking page over a valid one.

- **No access to the code:** Most websites builder do not provide the user with the full code behind the web application. This is important because web applications change and grow over time and websites builder may not always be able to keep up with the progress.

- **Website builder terms:** as a result of the previous point, the web application can only be published by the same software company and this makes the owner forced to agree on the terms and conditions that this company has to offer.

In addition to HTML features, Thanks to HTML5 users can access web applications without being connected to the Internet. That is why HTML5 was used to develop the SweCarCol web application.
6.2 Prototype

Mattias Arvola [11] considers the paper prototype to be popular because they make it easy to see how different sides are connected and because everything was inspired by paper and ink and it is nice to go back in time.

Interactive designs offer higher fidelity because they include are done with software that gives a closer look to the reality and because their functionality works while paper prototypes functionality is fake.

Nevertheless, creating interactive prototypes is time-consuming while static low fidelity prototypes can be done much faster, which gives the designer more time to work on the design. Designers feel less committed to low-fidelity prototypes and are more flexible about changing the design [12].

6.3 Evaluation

Regular UI evaluation process is always better, they give early feedback and more ideas. Yet, such an evaluation is time-consuming, a trade between evaluation and development must be done. The implementation process was more important in this project because the web application is not the final product, it will be developed further before it comes to publish stage. Nevertheless, UX expert evaluations for functionality were regularly done to ensure the functionality of the connection between backend and frontend. In addition to a final UI/UX evaluation.

According to Nielsen, No more than 5 people are needed to effectively evaluate a product [32].

![Figure 8: Why You Only Need to Test with 5 Users](image)
Three users were contacted in this project evaluation process instead of five to maintain some privacy about the project and according to figure 8, 50 percent of the problem were found. Feedback points gave a new perspective of the application and suggested new features.

**6.4 Microservice or Monolithic Architecture**

The decision to use a microservice architecture was made on the basis that the web application had discrete features, an upload, a search and a login feature. This meant that each feature could be modelled as a service. By doing so it would be possible (if needed) to assign work for each service to different individuals, they could then work independently of each other and in parallel.

Some of the other benefits of a microservice architecture are that new developers would not have to learn the entire codebase of the application. Instead those developers can focus on learning the codebase of the service or services they have been assigned without having knowledge of the inner workings of the entire application. While in the context of a web-application, if one or more services stop working, the website can still be up and running as long as it isn’t one of the services that stopped working.

Unfortunately what wasn’t taken into account was that for prototyping purposes this solution was quite cumbersome to work with as multiple Java Virtual Machines had to be running at the same time. This would slow down the computer that was running the web application and if the memory of that computer wasn’t managed properly the computer would shut down because the memory would be full.

By using a monolithic architecture it would decrease the time it would take to design the backend and the implementation of the backend could have started earlier. Additionally the web application would require less memory since only one instance of the Java Virtual Machine would be running which in turn would minimise the risk of the computer that was running the web application from crashing.

Some of the drawbacks of using a monolithic architecture are that if the codebase grows too big, perhaps because the client is demanding more features, it could become quite difficult to introduce new developers into the project. The difficulty stems from the fact that the new developer will require time to familiarise themselves with the codebase which would be bigger compared to the microservices codebase the developer would be in charge of. This could in turn lead to a decline in the time spent developing the application. Other potential drawbacks are that modularity could break down over time and as a result, the quality of the code contributed into the project would decrease as it is dependent on the understanding of the codebase.
Ensuring that the application is scalable is important for this project as there are potential for more features to be added in the future. While both solutions do have their pros and cons as mentioned earlier, the loosely coupled nature of a microservice architecture would appear to be the more suitable architecture.

6.5 The SQL Database

When working on the SQL database there was a focus on ensuring that the solution was independent of the SQL-language that was used. This is why when reading through sections 4.2.3, 5.5 and 5.6 the database is referred to as an SQL database and not by the specific database management system that is being used. The approach ended up being successful and the final result can be replicated with other database management systems using SQL.

6.6 The Shortcomings of the Database Search Engine

While the database search engine was able to search through the entire database to find data, it was unclear as to how the database search engine could be tested in order to evaluate the accuracy of a search. This would have been helpful for evaluating the database search engine and for making improvements.

It was also identified that the lack of understanding of hibernate search [30], the library that was being used to implement the database search engine, lead to some features not being implemented. For instance in the current build it is not possible to search for dates. This is because a different date data type is required by the hibernate search library compared to what is being used at the moment. Unfortunately due to time constraints no efforts were made to change the data type as it would require the search engine service to be refactored. If work on the database search engine were to continue, ensuring that dates are searchable should be a priority.

6.7 Sorting the Data in the Search Page

One of the features that were implemented in the search engine service was sorted search, it performed a similar job to advanced search but the data it returned would be sorted. A sorted search was never meant to be a search function but instead, it was meant to sort the data that was displayed on the search page. When using the web application, the only way a user can access sorted search is if they first perform either a simple or an advanced search. After doing so, it is possible to perform a sorted search by selecting one of the options to sort the data by. This solution is redundant since the sorting requires
performing a search in the database to retrieve the data that was already visible to the user and only to be sorted after that. Instead, the sorting should be performed by the frontend. A potential way to perform the sorting from the frontend is by having all the data that is displayed on the search page to be stored in a variable, javascript functions can then be used with the variable as input to sort the data. This solution would eliminate the need to make an HTTP request to the backend which has the potential to improve the time it takes to sort the data.

6.8 The Login Service

The login service was the only service in the microservice architecture that ended up being outside the scope of this paper. It was made only to demonstrate the behaviour of the final product. Since this service was outside the scope of this paper, the implementation was quite basic and leaves a lot to be desired. As a result very little time was put into security which lead to the JSON web token being stored in a cookie, this isn’t safe as it is vulnerable to Cross-Site Request Forgeries [33]. For demonstration this service does the job it is intended for but for real world use it would need more work.

7 Conclusion

Building a web application according to the usability principles found in section 4.1.2, resulted in a clear correlation between the application and the principles. By evaluating the web application it was observed that when the design of the web application deviated from the usability principles the application became difficult to use. While when the usability principles were followed, the user experience was improved.

As for the backend, it was concluded that for demonstration purposes a monolithic architecture should be used due to the low risk of crashing from the memory being full. While the modularity and loose coupling that is offered by the microservice architecture could not be overlooked.

8 Future Work

8.1 Frontend

By using the results from the evaluation of the web application, the application can be improved by:
• Making it more clear to the user that they have successfully performed a search.

• Changing the name of some of the items on the navigation bar to minimise the risk of confusion.

• Being more consistent with the language that is used across the entire website. To add to this point, perhaps implement a feature that changes the language for the entire website.

• Creating a more clear administration environment.

8.2 Backend

As discussed in section 6.6 there is a need to develop a method to test the accuracy of the database search engine.
References


