

**IPL**

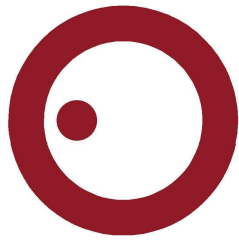
**escola superior de tecnologia e gestão**  
instituto politécnico de leiria

Instituto Politécnico de Leiria  
Escola Superior de Tecnologia e Gestão  
Departamento de Engenharia Informática  
Mestrado em Eng.<sup>3</sup> Informática – Computação Móvel

## FUNDAMENTALS OF DESIGN SYSTEMS

MAROUEN ABDI





**IPL**

**escola superior de tecnologia e gestão**  
instituto politécnico de leiria

Instituto Politécnico de Leiria  
Escola Superior de Tecnologia e Gestão  
Departamento de Engenharia Informática  
Mestrado em Eng.<sup>a</sup> Informática – Computação Móvel

## FUNDAMENTALS OF DESIGN SYSTEMS

MAROUEN ABDI

Número: 2182710

Estágio realizado sob orientação do Professor Doutor Silvio Priem Mendes ([smendes@ipleiria.pt](mailto:smendes@ipleiria.pt)).

Leiria, Novembro de 2020



## ACKNOWLEDGEMENTS

---

I would first like to thank my thesis advisor, Dr. Silvio Mendes of the Escola Superior de Tecnologia e Gestão de Leiria, at Politécnico de Leiria. The door to Dr. Mendes's office was always open whenever I ran into a trouble spot or had a question about my research or writing. He consistently allowed this paper to be my own work but steered me in the right direction whenever he thought I needed it.

It is my radiant sentiment to place on record my best regards, deepest sense of gratitude to Mr. Nelson Rodrigues CEO of Mediaweb Creations, Ms. Ines Neves, Director Manager, and all the team of Mediaweb Creations for their careful and precious guidance which were extremely valuable for my study both theoretically and practically.

Finally, I must express my very profound gratitude to my parents and to my sister for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them.



## RESUMO

---

Este projeto de tese consiste no mundo dos sistemas de design, os benefícios que trazem para a organização, equipas, utilizadores e o negócio. Vamos mergulhar neste novo mundo de sistemas de design, para entender o que são, como podemos estabelecê-los e quais são as regras, fundamentos e diretrizes que devemos seguir. Passando por mais detalhes, como criação de cores, tipografia, espaços e inventários de ícones. Não só isso, vamos também percorrer os padrões de IU que são muito importantes para a estrutura de um sistema de design e isso será seguido pelo processo de criação do Rocket Design System.

Ter um sistema de design terá um bom impacto no negócio, uma boa comunicação da equipa, agilizará o trabalho e manterá a organização atualizada com as tecnologias e as mudanças do mercado. Foi por isso que a Mediaweb Creations se concentrou na criação do "Rocket Design System", no qual tive a oportunidade de trabalhar durante o estágio em que estive com eles.

A Mediaweb Creations é uma empresa especializada no desenvolvimento web. Fundada em 2006, a Mediaweb está atualmente presente em Leiria e Lisboa, Portugal. A sua missão é ajudar os clientes a terem melhores experiências de utilizador, provando diversos serviços de UX / UI Design, Frontend e Web Development.

Os benefícios de ter um sistema de design em uma organização é principalmente no desenvolvimento web, o que nos faz ter uma única fonte de verdade e fará com que o processo demore menos, seduzindo os clientes com um produto atraente, fortalecendo as equipas e permitindo que a informação flua mais longe.





## ABSTRACT

---

This thesis project consists of the world of design systems, the benefits that bring to the organization, teams, users, and the business. We will dive into this new world of design systems, to understand what are they, how we can establish them, and what are the rules, foundations, and guidelines that we should stick to. Going through more details, such as creating colors, typography, spaces, and icon inventories. Not only that, but we will go through UI patterns which are very important for the structure of a design system and this will be followed by the process of creating Rocket Design System.

Having a design system will have a good impact on the business, good team communications, make the work fast, and keep the organization up to date with technologies and market changes. That was the reason why Mediaweb Creations focused on creating "Rocket Design System" which I had the opportunity to work on during the internship I had with them.

Mediaweb Creations is a company specialized in web development. Founded in 2006, Mediaweb is present in Leiria and Lisbon, Portugal. Their mission is to help the clients having better user experiences by providing UX/UI Design, Front-End, and Web Development several services.

The benefits of having a design system in an organization. Especially a web development related one, having a single source of truth will make the process take less time, seducing clients with an attractive product, making the teams stronger, and letting the information flow goes farther.



# ÍNDICE

---

Acknowledgements	i
Resumo	iii
Abstract	v
Índice	vii
Lista de Figuras	ix
Lista de Tabelas	xiii
1 INTRODUCTION	1
1.1 Introducing Design System . . . . .	1
1.2 Rise of Design System . . . . .	2
1.3 The Rise of The Web World . . . . .	3
1.4 The Impact of Technology on Industry . . . . .	5
1.5 The Agile Transition . . . . .	5
1.6 Benefits of Design Systems . . . . .	6
1.7 Measure Design System as Impact . . . . .	12
2 STATE OF THE ART	15
2.1 UX Methodology . . . . .	15
2.2 Atomic Design Methodology . . . . .	17
2.3 Foundations . . . . .	25
2.4 Creating The Inventory . . . . .	33
2.5 UI Patterns . . . . .	38
2.6 UI Patterns in Design Systems . . . . .	43
3 COMPETITOR ANALYSIS	49
3.1 IBM Carbon Design System . . . . .	49
3.2 Google Material Design System . . . . .	54
3.3 Microsoft Fluent Design System . . . . .	58
3.4 Foundations Comparison . . . . .	62
3.5 Overall Comparison . . . . .	63
4 ROCKET DESIGN SYSTEM DEVELOPMENT	65

## ÍNDICE

4.1	Work Environment . . . . .	65
4.2	Creating Inventory . . . . .	67
4.3	Documentation . . . . .	69
4.4	Creating Components . . . . .	71
4.5	Creating Screen Templates . . . . .	72
5	CONCLUSION AND FUTURE WORK	77
	BIBLIOGRAFIA	79
	DECLARAÇÃO	81

## LISTA DE FIGURAS

---

Figura 1	UXPin State of the Enterprise UX Survey . . . . .	2
Figura 2	Difference between 1990's Yahoo and 2019's Yahoo . . . . .	3
Figura 3	Page-based design vs Component-based design . . . . .	5
Figura 4	Visualization of web mobile application and design system by Cristiano Rastelli . . . . .	9
Figura 5	Design system saves money by uxdesign . . . . .	10
Figura 6	Internal Sketch Design System Coverage Tool by Keap . . .	14
Figura 7	An example of a chemical equation showing hydrogen and oxygen atoms combining together to form a water molecule	15
Figura 8	The periodic table of chemical elements . . . . .	16
Figura 9	The periodic table of HTML elements . . . . .	17
Figura 10	Atomic Design elements: Atoms . . . . .	18
Figura 11	Atoms include basic HTML tags (inputs, labels, and buttons)	19
Figura 12	Atomic Design elements: Molecules . . . . .	19
Figura 13	A search form molecule is composed inputs, labels, and buttons	20
Figura 14	Atomic Design elements: Organisms . . . . .	20
Figura 15	Header organism is composed of a search form molecule, logo atom, and primary navigation molecule. . . . .	21
Figura 16	Atomic Design elements: Templates . . . . .	22
Figura 17	Atomic Design elements: Templates . . . . .	22
Figura 18	Atomic Design elements: Pages . . . . .	23
Figura 19	The page stage replaces placeholder content with real repre- sentative content to bring the design system to life . . . . .	24
Figura 20	Atomic Design Methodology for Streamloan . . . . .	24
Figura 21	Beagle – Responsive Admin Template based on Bootstrap .	27
Figura 22	Contrast Grid [25] . . . . .	29
Figura 23	IBM Carbon Design system colors inventory . . . . .	34
Figura 24	Material Design Design system Typography inventory . . . .	35
Figura 25	Airbnb Design system Icons inventory . . . . .	37
Figura 26	IBM Carbon Design system spacing inventory . . . . .	38
Figura 27	Bootstrap's thumbnails component (default example) . . . .	39
Figura 28	Screenshot of Slackbot from Slack . . . . .	40

Figura 29	Screenshot of drag and drop option from Invision . . . . .	41
Figura 30	Screenshot from Pinterest . . . . .	42
Figura 31	Screenshot from Monash University's editorial style guide . . . . .	43
Figura 32	Screenshot of one of Instagram's brand guidelines . . . . .	44
Figura 33	Screenshot from the goals of Material design . . . . .	45
Figura 34	Screenshot of Lightning Design System Design Tokens . . . . .	46
Figura 35	Screenshot of Walmart's Display Price component with deprecation warning . . . . .	48
Figura 36	IBM Carbon Design System . . . . .	49
Figura 37	Carbon Design System - Color Inventory . . . . .	50
Figura 38	Carbon Design System - Typography Inventory . . . . .	51
Figura 39	Carbon Design System - Icons Inventory . . . . .	51
Figura 40	Carbon Design System - Spaces Inventory . . . . .	52
Figura 41	Carbon Design System - Storybook . . . . .	53
Figura 42	Carbon Design System - Documentation . . . . .	53
Figura 43	Google Material Design System . . . . .	54
Figura 44	Material Design System - Color inventory . . . . .	55
Figura 45	Material Design System - Typography inventory . . . . .	56
Figura 46	Material Design System - Icons inventory . . . . .	56
Figura 47	Material Design System - Spaces inventory . . . . .	57
Figura 48	Material Design System - Button Component . . . . .	57
Figura 49	Carbon Design System - Documentation example . . . . .	58
Figura 50	Microsoft Fluent Design System . . . . .	59
Figura 51	Microsoft Fluent Design System - Color Inventory . . . . .	60
Figura 52	Microsoft Fluent Design System - Typography Inventory . . . . .	60
Figura 53	Microsoft Fluent Design System - Spaces Inventory . . . . .	61
Figura 54	Microsoft Fluent Design System - Documentation . . . . .	62
Figura 55	Foundation comparison of Carbon, Material and Fluent Design Systems . . . . .	63
Figura 56	Overall comparison of Carbon, Material and Fluent Design Systems . . . . .	63
Figura 57	Screenshot of the CSS file . . . . .	68
Figura 58	Screenshot of the folder of the new structure . . . . .	68
Figura 59	Screenshot of the script project shared as open source on Mediaweb github account . . . . .	69
Figura 60	Screenshot of the documentation of the Accordion Component . . . . .	70
Figura 61	Screenshot of the canvas of the Accordion Component . . . . .	71
Figura 62	Screenshot of the Button Component . . . . .	72

Figura 63	Screenshot of the Button Group Component . . . . .	72
Figura 64	Screenshot of the Header Component . . . . .	72
Figura 65	Screenshot of the Three columns page . . . . .	73
Figura 66	Screenshot of the Master Detail page . . . . .	73
Figura 67	Screenshot of the Admin Dashboard page . . . . .	74
Figura 68	Comparing Rocket Desing System with competitors . . . . .	74







## INTRODUCTION

---

### 1.1 INTRODUCING DESIGN SYSTEM

Before starting the process of creating Rocket Design system, we needed to go back through the history of design systems to understand them more and get a better introduction to it.

In 1968, Douglas McIlroy the American mathematician presented in the software engineering conference of Nato a component-based development to solve the dilemma. Because it provided a solution to speed up the programming's potential by creating reusable code which will provide efficiency and ease to scale. This idea lowered the effort and boosted the speed of the development allowing software to better use the power of modern computers. Now, After more than 50 years, a similar challenge is faced but this time in design. A huge struggle to scale design with the applications it supports because the design is still bespoke tailoring solutions for individual problems.

A design system is the center of shared parts and procedures, such as components, patterns, and guidelines to build reliable and robust products. It's the ecosystem where the design procedure happens and the fruit of the design thinking reaches its highest level. The term design system itself envelops all of the design, code, and content assets, they are tailored to organizational necessities. Also, design systems show the culture, team values, and visual language of a company.

Design systems empower teams to manufacture better products quicker by making design reusable. Reusability makes scale conceivable. This is the heart and essential value of design systems. A design system is an assortment of reusable components, guided by clear guidelines, that can be amassed together to build any number of applications. The goal of a design system is to build a solution that will help the team to scale a product successfully. With a well defined and clear system, both teams of designers and developers will focus their efforts on giving what the user needs, instead of creating elements and inventing solutions that may be done before.[13]

## 1.2 RISE OF DESIGN SYSTEM

Design systems are an intriguing issue at this moment. Everybody discusses them and needs to construct them. It is apparent that the product has eaten the world, and design has a significant function in this. Without a doubt, design systems are truly necessary assistance for scaling design and advancement groups. Managers and decision-makers started to be convinced of the value a design system can bring to their organization. They are already aware of how it can make their life much simpler and work easier. Because they are already sure that having established guidelines and usability patterns will empower the communication between the team members and the simplicity of the user interface.

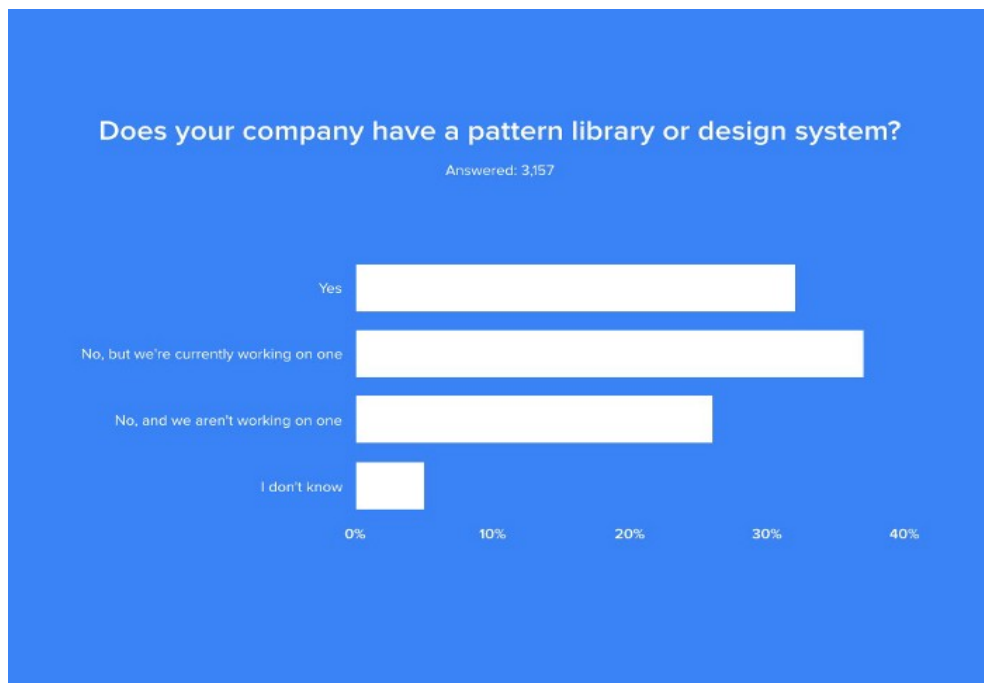


Figura 1: UXPin State of the Enterprise UX Survey

UXPin made a series of posts to cover their journey while building their design system presented in Figure 1 [5]. The journey started with interviewing more than 40 designers, developers, and product leaders from different companies and they end up running a large-scale survey on the state of enterprise design. The survey had 3.157 respondents and showed that most enterprises either have a design system in use or are currently working on building one. Big names came on the list such as Salesforce, IBM, Airbnb, Microsoft, and Atlassian lead the group with mature usage of living design systems.[5] [18]

We can say that design systems are the crossing point between art, technology, and industry. The rise of the computer and internet and technologies, more precisely the web development (the improvement of CSS and JavaScript), the influence that technology brought to industries and businesses in general, and the change from Waterfall to Agile methodologies. These factors were very critical to the rise of design systems.

### 1.3 THE RISE OF THE WEB WORLD

The beginning of the internet was dull for design. Websites were minimal more than glorified Word documents. In these prior times, the web was constructed using just HTML. CSS didn't exist yet, and it was surrendered over to singular browsers to decide how a webpage would be shown. An early webpage design lacking CSS styling can be seen in Figure 2. of the search engine, Yahoo compared with a newer version.



Figure 2: Difference between 1990's Yahoo and 2019's Yahoo

this all started to change during the 1990s, as the number of individuals having access to personal computers and the internet was increasing. At last, this implied the web developed in prominence as the spot for electronic publishing. Web creators became progressively baffled by the limitations of HMTL regarding styling. Browsers, perceiving the opportunity this presented, started investigating approaches to give the creators more control on their pages. The first proposition for CSS was presented in 1994 and prodded a ton of conversations. Surprisingly, a lot of discussions spun around who should control how a page would be seen. On the one side, many felt

that creators ought to decide the last view, guaranteeing that it coordinated their aim and vision. On the opposite side, some accepted that clients ought to be able to choose how they might want to see a page. These discussions murmured the production of the World Wide Web Consortium (W3C), to help work out norms and best practices for the web and eventually, HTML and CSS. This was an important development in the world of the Internet. Without the W3C to unify and make recommendations, every browser might uphold an alternate HTML particular. [12]

As the web developed, so did the need to have a better user experience. Designers started to play with this new medium, applying the fundamentals of layout and grid techniques utilized on paper. Indeed, even with the selection and spread of CSS, designing on the Web in these early days was difficult and restricted. Web layouts were being created utilizing tables, and control was absent while modifying layouts using HTML and CSS. It is a similar idea to making a table in a Microsoft Office Word Document. Yet, a few issues emerge, for instance, to implement a convoluted design interface using just pictures embedded inside table data cells. Table cells in a similar line can't be distinctive width. Web designers needed to get exceptionally innovative, by using confounded workarounds of sliced Photoshop pictures, tables within tables, and concealed content among an assortment of different hacks.

By 2002 the idea of "Table-less design" was starting to pick up energy. Instead of depending on tables for situating the components of pages, CSS had progressed enough to carry out the responsibility of doing the job more proficiently. This freed designers up to try different things with an assortment of imaginative and progressively complex layouts. Up until this time, numerous websites were assembled page by page. Components, for example, the header, footer, and menu were copied and pasted over each page. Change something on one page, and you need to go to each other page to roll out similar improvements. As the multifaceted nature and size of websites increased, performance, scalability, and maintainability turned out to be genuine concerns.

JavaScript additionally assisted with changing how we approach building websites. It permitted designers to build components as opposed to implement whole pages. Components resemble Lego blocks that can be moved around to rethink practically any interface as in Figure 3 [21]. Rethinking pages and even entire sites turned out to be quicker and simpler. The pace of improvement picked up rapidly, permitting organizations to advance and deliver quicker than at any other time.

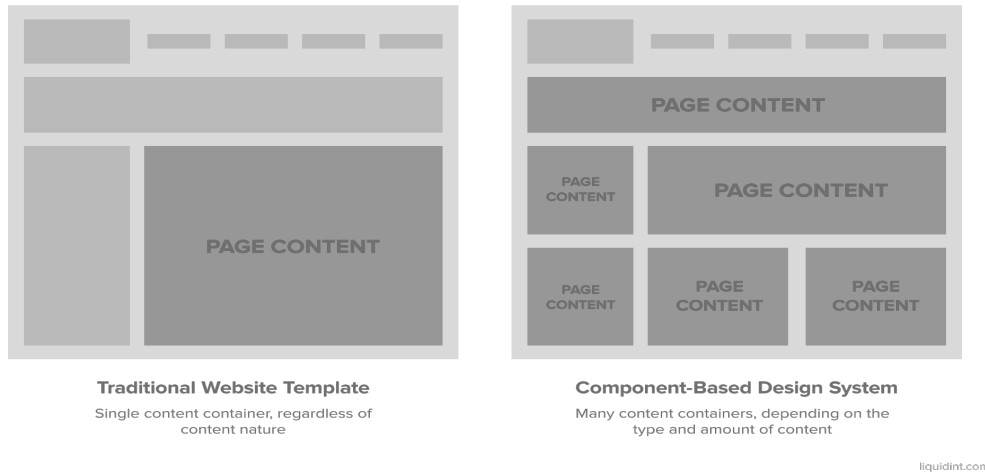


Figura 3: Page-based design vs Component-based design

1.4 THE IMPACT OF TECHNOLOGY ON INDUSTRY

Internet developed quickly and the number of persons having their PCs for personal purposes rose. As indicated by a report from the U.S Department of Labor, Personal computer's possession in the U.S rose from 15% to 35% between the years 1990 and 1997. Personal computers are not a luxury product anymore, they were a need. Organizations quickly perceived the potential of this new market. Numerous organizations were established during what is known as the dot-com boom. This expansion in demand for technology caused a whirlwind of fervor and a need to keep moving to build greater and better software products as fast as expected.

Besides the birth of a new market with huge potential, the technology had a great impact on the organization itself. Business innovation enabled companies to improve their communication processes. Emails, messaging, websites, and applications, for instance, communication with buyers is much easier now. Using a few sorts of data technology communication specialized techniques empowers organizations to immerse the financial market with their message. Organizations may likewise get more buyer criticism through these electronic specialized strategies.

1.5 THE AGILE TRANSITION

The conventional waterfall technique for programming development was based on a weighty cycle and long lead times. it could take a long time to get from

thoughts to distribution. Once secured in a concept, organizations using the waterfall cycle couldn't undoubtedly change course on the off chance that they found that something just wouldn't work. This was because a waterfall cycle moved toward item improvement in a consecutive design. Necessities would be passed on to designers, and designs would be passed on to engineers, pretty much ruling out coordinated effort among groups. Without this joint effort set up, it was incredibly hard to move bearings when issues emerged later all the while.

During the year 2001, with the lack of progress leaders tried to find a better way to develop products. This leads to the birth of an Agile, fast-delivery approach to building software. it allows organizations to convey their products rapidly, with full knowledge that they might not have hit the nail on the head the first run through. They could then accumulate client criticism and improve in quick progression. Agile spotlights on the building and using the product and not the cycle of tools it takes to make that product.

## 1.6 BENEFITS OF DESIGN SYSTEMS

Each company has its uniqueness, both in its contributions and its difficulties. Imparting the benefit and advantage of a design system can acquire a relatable way is important. Approach it as you would a problem issue. The initial step is to comprehend the organization, its objectives, and challenges, just as individual jobs and the parts they play. Few essential questions need to have an answer. What are our goals as a company? How can we make the user experience and broader organization's goals even? Who will benefit from such a system? Who will benefit the most?

To be able to answer the previous questions, we should examine the value of a design system in light of who will benefit and how. We will have three targets: The employees of the company, users of the system, and the company itself. As a start, we will focus on the first target which is the workers of the organization. As a manager or a CEO, life will be perfect if the employee's life is comfortable which will make benefits the other target audiences.

Based on a survey made by **Sarah Vesselov and Taurie Davis in Building Design Systems** [2] on 82 people with questions about what's frustrating them most in their work. We used this data to explain more the benefits of design systems on each role.

## Designers

Most of designers are put with critical thinking and elevating the perception of design at the head of their needs list. A lot of them felt baffled and kept down by obscure requirements, an excessive number of meetings, and the pressure to convey solutions rapidly without legitimate disclosure.

- A lot of meetings
- Out-dated documentation
- Lack of understanding the value of design
- Vague requirements
- Pressure for quick delivery
- Lack of support from product owners

Design systems can help lighten this dissatisfaction while empowering designers to move quicker and be more profitable. A setup system gives the design group admittance to styles, shared components, and implementation guidelines. This single source of truth causes the group to work freely with each other while guaranteeing consistency. Common design documents permit designers to rapidly and precisely set up high-fidelity designs.

Design systems also offer designers the chance to venture outside of their conventional tool-set and make models in the browsers. Doing so empowers them to perceive what will be underway rather than the ideal world inside the design tools. Some design systems make components accessible in the browser, permitting designers to rapidly and effectively change attributes and behavior with no coding skills needed.



## **Engineers**

The engineers offered shockingly comparable responses as the designers. Such a large number of meetings, lacking prerequisites, and missing design resources were among their important grumbles. Similarly, with designers, there was an individual need to take part in satisfying work that used their creativity to take care of everyday issues.

- A lot of meetings
- Documentation maintenance
- Muddled dev environements
- Lack transparency between teams
- Lack of discovery
- Missing assets

The benefits that design systems can bring for engineers, clarifying solutions, making assets available for reading, give engineers the tools to create content with autonomy. The design-to-engineer handoff is an extra source of dissatisfaction for some. Designers regularly need to make definite specs, for example, red-lined designs, for each design gave off to an engineer. Done physically, this is a time-consuming cycle. With a built-up system, design specs stay predictable and can be consequently recorded and produced. Engineers can utilize these rules as a source of perspective during the execution stage, eliminating the requirement for a to and fro with designers. With well-detailed documentation, engineers will have the option to duplicate the code for the components they require and proceed onward.

## **Design system as a language**

A decent beginning stage is to emphasize design systems as a language. Every language is one of a kind to the organization and the product. Documenting this language and setting up a reasonable vision diminishes the probability of miscommunication and makes it simpler for workers over the company to add better user experiences. The design, product, development, sales, and marketing departments all remain to benefits.

Making this language open to all the teams expands consistency and engages all groups to be important for improving the user experience. There is no requirement

for different teams to exceed limits or inadvertently work against UX if the design language is accessible to all.

Making this language accessible to all teams inside the organization, will remarkably increase the speed and the efficiency of the work. Cristiano Rastelli made an outstanding article talking about how his team measured the effect of their Design System Cosmos at Badoo. By looking after their Git logs, he was able to show us with numbers how it's clear and obvious that the Design System has a great impact on the work speed.

In the figure below we can see that the amount and frequency had faced a big change after the Cosmos Design System was brought to life as can be see in figure 4 [22].

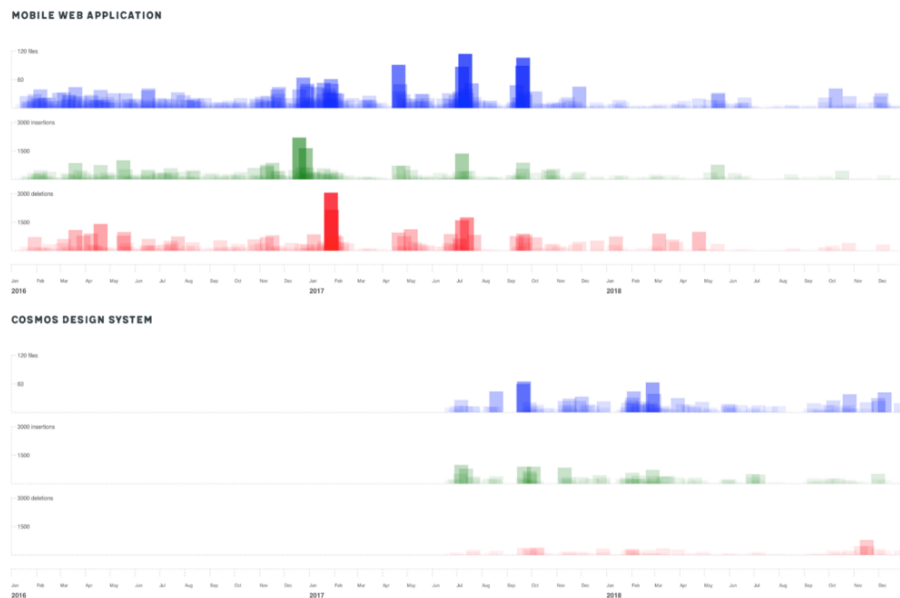


Figura 4: Visualization of web mobile application and design system by Cristiano Rastelli

### Organizations level

Numerous organizations put additional time and exertion into developing features than they do into growing great user interfaces. As designers, it tends to be baffling to take a load off at the table however feel no force while in that seat. Strain to push out outcomes can cause you to feel like you are a specialist as opposed to an accomplice. Huge numbers of the most successful organizations are design-driven or have a profound appreciation for the value a great design can bring. An investigation distributed by McKinsey and Company in 2018 found a solid connection between's

the way fit an organization is at design and how well it performs monetarily. Apple, Airbnb, Slack, and IBM strike a chord rapidly. These organizations have put time and assets into the design, with design administration existing the C-suite.[9]

Having a Design System will help the company saving money, As a product develops, there turns out to be more information and on-boarding expected to comprehend the complexities of the system. It requires some investment and cash to deliver products to advertise without a system set up as can be see in figure 5 [23].

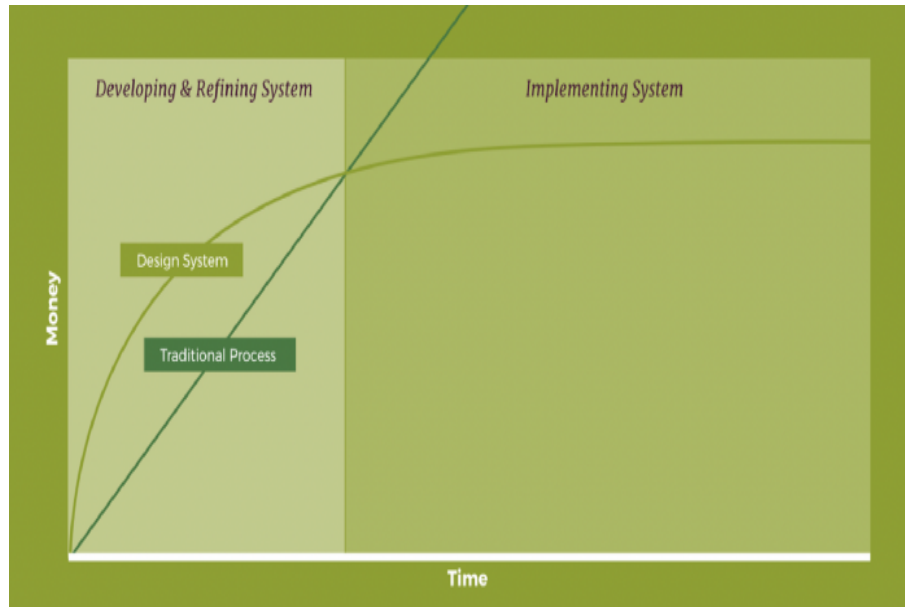


Figura 5: Design system saves money by uxdesign

With a design system set up, product teams can construct products all the more proficiently at scale, this will allow more focus on adjusting and making more enjoyable moments for the clients.

The savings cost can be calculated with a simple formula:

$$H * 52 * R * T = S$$

- H : Hours per week
- 52 : weeks of the year
- R : Hourly rate
- T : Team size
- S : Year savings

For instance, if a designer and a developer spend 2.5 hours on a new component creation from scratch for 52 weeks which are the workweeks in a year. The hourly rate will be respectively 70\$ and 85\$ per hour. The company would save about :

$$2.5 * 52 * 70 * 1 = 9,100\$$$

$$2.5 * 52 * 85 * 1 = 11,050\$$$

And if we consider in a small organization we will have 2 designer and 7 developers.

$$2.5 * 52 * 70 * 2 = 18,200\$$$

$$2.5 * 52 * 85 * 7 = 77,350\$$$

Annual savings = 95,550\$ saved in a year.

## Users level

The essential beneficiary of a well-thought-out design system is the end-user. Enchanting users is the thing that will frequently decide the accomplishment of a product. The general usability of a product, combined with the advantages it offers a user, is basic. All the advantages listed in the other sections down to profit the user:

- Cheerful and more productive employees have more opportunities to take a shot at complex flows and user experience arrangements.
- Diminished process duration permits improved experiences to make it into the product quicker, expanding feedback, and further cycle.
- Sales and promoting can fittingly address purchaser concerns and inquiries before are secured.

A design system will build trust and consistency, the end-user expects a cleaned and predictable experience when using a product. A Design System can help refine the nature of our product by making the experience more steady, unsurprising, and available. This happens when design and improvement teams chip away at building a solitary source of truth. For instance, if we want to think about a sample objective for a flight finder product, it will be improving consistency and predictability. The key results will be increasing both the NPS of the mobile flight product and the design system coverage for flights product.[10]

### 1.7 MEASURE DESIGN SYSTEM AS IMPACT

It's truly difficult to measure people's joy or the effect on how people work and think in term of user interfaces or the groups' productivity and speed in conveying updates and a new feature to a product and more the impact of an adjustment in a complex system such as a big organization. It's a very common problem in the Design systems to include some meaningful metrics because most managers believe in a myth that has been around seen decades.

*If you can't measure it you can't manage it*

**Peter Drucker**

Measuring the impact of Design Systems can be broken down into two different categories: subjective and objective.

### **Subjective Measurement**

This may come through reviewing end-clients or internal groups to hear a heartbeat on how individuals' point of view may change. For end-clients, this may come as an NPS review in would like to see higher evaluations, positive input, and ex- and trust of the product. End-users may see an improved encounter from better availability or consistency in the experience. For internal teams, this can be pre- sented as an internal survey that focuses on the improvements that can be made in the workflow, some question can come to mind like :

- How frequently do you end up utilizing the Design System?
- What is the thing you would improve about the Design System?
- How does the Design System improve your work process on a 5 stars scale?

### **Objective Measurement**

We can measure the impact of a design system in a very objective way by asking the teams a few questions :

- Does our Design System have a component library?
- Are we providing robust documentation?
- Does the Design System have a process?

The estimation of the impact can likewise be followed by specific metrics. For the end-client, it may be seeing page load times and accessibility. Preferably, when working out a design system, the team is attempting to bring together code, address accessibility issues, and so forth. Maybe we can also take a gander at the number of lines of code that were eliminated or moved to utilize design tokens.[11]

### Sketch Design System Coverage

A breakdown of how the design system is used in sketch files

Overall coverage across all tracked files

**3.72%** 23095 of 620727 layers are Design System Symbols      **46.8%** The Internal Forms project has the most coverage      **0%** The Partner Edition project has the least coverage

By project

Project	Last updated ↓	File count	Layers	Layers w/ DS	Design System Coverage
Automation	3/25/2019	9	126,004	2,229	1.8%
Buy Now - CAM	3/25/2019	1	609	135	22.2%
Hackathon and Other Misc	3/25/2019	7	4,072	417	10.2%
Infusionsoft by Keap	3/25/2019	24	185,271	9,900	5.3%
Internal Forms	3/25/2019	1	1,641	768	46.8%
Keap	3/25/2019	16	177,161	8,804	5%
Keap Mobile Apps	3/25/2019	2	42,527	356	0.8%
Partner Edition	3/25/2019	1	0	0	0%
Phase II	3/25/2019	2	83,442	486	0.6%

Figura 6: Internal Sketch Design System Coverage Tool by Keap

We can likewise measure the impact of the product by estimating the selection of the design system. Sean Rice and Jordan Reed at Keap needed an approach to comprehend which projects required cleanup work, as can be see in figure 6 [23], which undertakings may require new components, and an overall depiction of design adoption. They made an inner device to monitor design system utilization in Sketch by working off of node-sketch.

STATE OF THE ART

---

## 2.1 UX METHODOLOGY

The idea of User Experience has been around sooner than the abbreviation UX itself. At the point when we lead user research through methods like surveys or interviews, it is a cycle of user-focused design intending to improve the experience of clients for products and services.

The search for a methodology to craft something innovative sometimes lead to taking a look somewhere else. Taking a look at other areas and industries can sometimes give inspiration that leads to huge discoveries. Given this incredibly perplexing world, we have made, it appears to be just common that different fields would have handled the same issues and we could gain from and appropriate. Notably, heaps of different fields, for example, industrial design and architecture have created keen particular frameworks for assembling colossally complex objects like planes, boats, and skyscrapers. For our technology we are creeping back to smaller details of this complex world that we have, the natural world, bringing back memories from our high school life precisely chemistry classes.[3]

Chemical reactions are spoken to by chemical equations, as in Figure 7 [3] which frequently show how atomic components consolidate together to shape molecules. In the model underneath we perceive how hydrogen and oxygen consolidate together to frame water molecules.

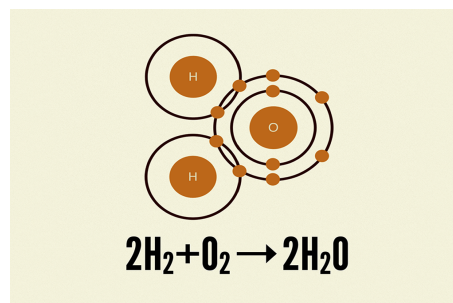


Figura 7: An example of a chemical equation showing hydrogen and oxygen atoms combining together to form a water molecule



As a natural behavior, atomic elements join together to frame molecules. These molecules can consolidate further to frame generally complex organisms. To clarify somewhat further :

- **Atoms:** are the essential structure blocks of all issues. Every chemical component has particular properties, and they can't be broken down without losing their importance. They are made out of many littler pieces like protons, electrons and neutrons however atoms are the smallest functional unit.
- **Molecules:** are gatherings of at least two atoms held together by chemical bonds. These blends of atoms take on their unique properties, and become more substantial and operational than atoms.
- **Organisms:** are gatherings of molecules working all together. These generally intricate structures can go from single-celled organisms all path up to unbelievably modern living like the human being.

So to conclude, atoms join together to shape molecules, which further combine to frame organisms. This atomic hypothesis implies that all issues in the realized universe can be broken down into a limited set of atomic elements presented in the figure 8[3].

**Periodic Table of the Elements**

1 IA H Hydrogen 1.008	2 IIA He Helium 4.003																	18 VIIIA Ar Argon 39.948																	
3 IIA Li Lithium 6.941	4 IIIA Be Beryllium 9.012											5 IIIA B Boron 10.811	6 IVA C Carbon 12.011	7 VA N Nitrogen 14.007	8 VIA O Oxygen 15.999	9 VIIA F Fluorine 18.998	10 VIIIA Ne Neon 20.180																		
11 IA Na Sodium 22.990	12 IIA Mg Magnesium 24.305	13 IIIA Al Aluminum 26.982	14 IIIA Si Silicon 28.086	15 VA P Phosphorus 30.974	16 VIA S Sulfur 32.06	17 VIIA Cl Chlorine 35.453	18 VIIIA Ar Argon 39.948											19 IA K Potassium 39.098	20 IIA Ca Calcium 40.078	21 IIIB Sc Scandium 44.956	22 IIIB Ti Titanium 47.867	23 IIIB V Vanadium 50.942	24 IIIB Cr Chromium 51.996	25 IIIB Mn Manganese 54.938	26 IIIB Fe Iron 55.845	27 IIIB Co Cobalt 58.933	28 IIIB Ni Nickel 58.693	29 IIIB Cu Copper 63.546	30 IIIB Zn Zinc 65.38	31 IIIB Ga Gallium 69.723	32 IIIB Ge Germanium 72.631	33 IIIB As Arsenic 74.922	34 IIIB Se Selenium 78.971	35 IIIB Br Bromine 79.904	36 IIIB Kr Krypton 84.798
37 IA Rb Rubidium 84.468	38 IIA Sr Strontium 87.62	39 IIIB Y Yttrium 88.906	40 IIIB Zr Zirconium 91.224	41 IIIB Nb Niobium 92.906	42 IIIB Mo Molybdenum 95.95	43 IIIB Tc Technetium 98.907	44 IIIB Ru Ruthenium 101.07	45 IIIB Rh Rhodium 102.906	46 IIIB Pd Palladium 106.42	47 IIIB Ag Silver 107.868	48 IIIB Cd Cadmium 112.414	49 IIIB In Indium 114.818	50 IIIB Sn Tin 118.711	51 IIIB Sb Antimony 121.760	52 IIIB Te Tellurium 127.6	53 IIIB I Iodine 126.904	54 IIIB Xe Xenon 131.29																		
55 IA Cs Cesium 132.905	56 IIA Ba Barium 137.328	57-71 Lanthanide Series	72 IIIB Hf Hafnium 178.49	73 IIIB Ta Tantalum 180.948	74 IIIB W Tungsten 183.84	75 IIIB Re Rhenium 186.207	76 IIIB Os Osmium 190.23	77 IIIB Ir Iridium 192.22	78 IIIB Pt Platinum 195.085	79 IIIB Au Gold 196.967	80 IIIB Hg Mercury 200.59	81 IIIB Tl Thallium 204.383	82 IIIB Pb Lead 207.2	83 IIIB Bi Bismuth 208.980	84 IIIB Po Polonium [209]	85 IIIB At Astatine [210]	86 IIIB Rn Radon 222.018																		
87 IA Fr Francium 223.021	88 IIA Ra Radium 226.025	89-103 Actinide Series	104 IIIB Rf Rutherfordium [261]	105 IIIB Db Dubnium [262]	106 IIIB Sg Seaborgium [263]	107 IIIB Bh Bohrium [264]	108 IIIB Hs Hassium [265]	109 IIIB Mt Meitnerium [266]	110 IIIB Ds Darmstadtium [268]	111 IIIB Rg Roentgenium [269]	112 IIIB Cn Copernicium [277]	113 IIIB Nh Nihonium [278]	114 IIIB Fl Flerovium [277]	115 IIIB Uup Ununpentium [278]	116 IIIB Lv Livermorium [276]	117 IIIB Uus Ununseptium [276]	118 IIIB Uuo Ununoctium [276]																		
		57 Lanthanide Series	58 La Lanthanum 138.905	59 Ce Cerium 140.116	60 Pr Praseodymium 140.908	61 Nd Neodymium 144.242	62 Pm Promethium 144.913	63 Sm Samarium 150.36	64 Eu Europium 151.964	65 Gd Gadolinium 157.25	66 Tb Terbium 158.925	67 Dy Dysprosium 162.500	68 Ho Holmium 164.930	69 Er Erbium 167.259	70 Tm Thulium 168.934	71 Lu Lutetium 173.055																			
		Actinide Series	89 Ac Actinium 227.028	90 Th Thorium 232.038	91 Pa Protactinium 231.036	92 U Uranium 238.029	93 Np Neptunium 237.048	94 Pu Plutonium 244.064	95 Am Americium 243.061	96 Cm Curium 247.070	97 Bk Berkelium 247.070	98 Cf Californium 251.080	99 Es Einsteinium [254]	100 Fm Fermium 257.085	101 Md Mendelevium 258.1	102 No Nobelium 259.101	103 Lr Lawrencium [260]																		

Figura 8: The periodic table of chemical elements

## 2.2 ATOMIC DESIGN METHODOLOGY

After talking about chemistry and the atomic theory in the last section, it's time to explain more what's the point of mentioning all that.

We discussed how all issues known to mankind can be separated into a finite set of atomic elements. As it occurs, our interfaces can be broken into a comparative set of elements. Josh Duck an Australian engineering manager has perfectly verbalized how the entirety of our sites and applications are made out of similar HTML elements in his Periodic Table Of HTML Elements presented in figure 9[3].

The image shows a 'Periodic Table of the Elements' where each element is an HTML tag, color-coded by function. The title is 'Periodic Table of the Elements'. The elements are arranged in a grid that tapers to the right, similar to the periodic table of chemistry. A legend at the bottom identifies the categories:

- Root element (green)
- Metadata and scripting (blue)
- Embedding content (purple)
- Text-level semantics (yellow)
- Grouping content (orange)
- Forms (light green)
- Document sections (light blue)
- Tabular data (light orange)
- Interactive elements (grey)

Periodic Table of the Elements																								
html																	col	table						
head	span																div	fieldset	form	body	h1	section	colgroup	tr
title	a																pre	meter	select	aside	h2	header	caption	td
meta	rt	dfn	em	i	small	ins	s	br	p	blockquote	legend	optgroup	address	h3	nav	menu	th							
base	rp	abbr	time	b	strong	del	kbd	hr	ol	dl	label	option	datalist	h4	article	command	tbody							
link	noscript	q	var	sub	mark	bdi	wbr	figcaption	ul	dt	input	output	keygen	h5	footer	summary	thead							
style	script	cite	samp	sup	ruby	bdo	code	figure	li	dd	textarea	button	progress	h6	hgroup	details	tfoot							
<div style="display: flex; justify-content: space-around; margin-top: 10px;"> <span>img</span> <span>area</span> <span>map</span> <span>embed</span> <span>object</span> <span>param</span> <span>source</span> <span>iframe</span> <span>canvas</span> <span>track*</span> <span>audio</span> <span>video</span> <span>device*</span> </div>																								

Figura 9: The periodic table of HTML elements

Since we started with a similar approach by having a finite set of building blocks, we can through the same process that has to do with the natural world to structure and create user interfaces. And this is when the Atomic Design enters.

Atomic design is a methodology made out of five distinct stages cooperating to make interface design systems in a more conscious and various leveled way. The five phases are :

- **Atoms**
- **Molecules**
- **Organisms**
- **Templates**
- **Pages**

Atomic Design is not a linear cycle, yet rather a mental model to assist us with thinking about our UIs as both a strong entire and an assortment of parts simultaneously. Every one of the five phases assumes a key part in the chain of command of our interface structure design. We should jump into each phase in more fine-grained detail.

### *Atoms*

On the off chance that atoms are the fundamental building blocks of the issue, at that point, the atoms of our interfaces fill in as the basic building blocks that contain all our user interfaces. These atoms incorporate essential HTML elements like form labels, inputs, buttons, and others that can't be separated any further consistently to be useful.

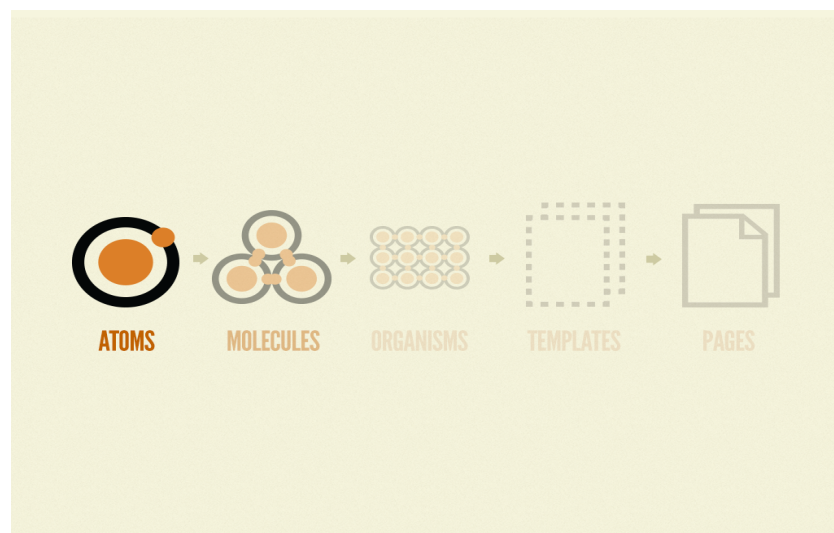


Figura 10: Atomic Design elements: Atoms

Every atom in the natural world possesses its unique properties. Hydrogen atoms have only one electron and the Helium atom contains two. Each interface atom has its own and unique properties too, such as dimensions, font size, background color... With these properties, we can control how the element will be applied in the user interface system.

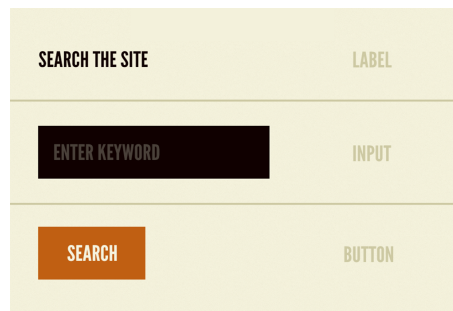


Figura 11: Atoms include basic HTML tags (inputs, labels, and buttons)

With regards to a pattern library, atoms exhibit all the base styles initially, which can be a useful reference to hold returning to as we create and keep up our design system. In any case, similar to atoms in the natural world, interface atoms don't exist in a vacuum and just truly come to life with the application.

### *Molecules*

In science, molecules are gatherings of atoms reinforced together that take on distinct new properties. For example, water molecules and hydrogen peroxide molecules have their one of a kind properties and carry on unexpectedly, even though they're comprised of similar atomic elements (hydrogen and oxygen).

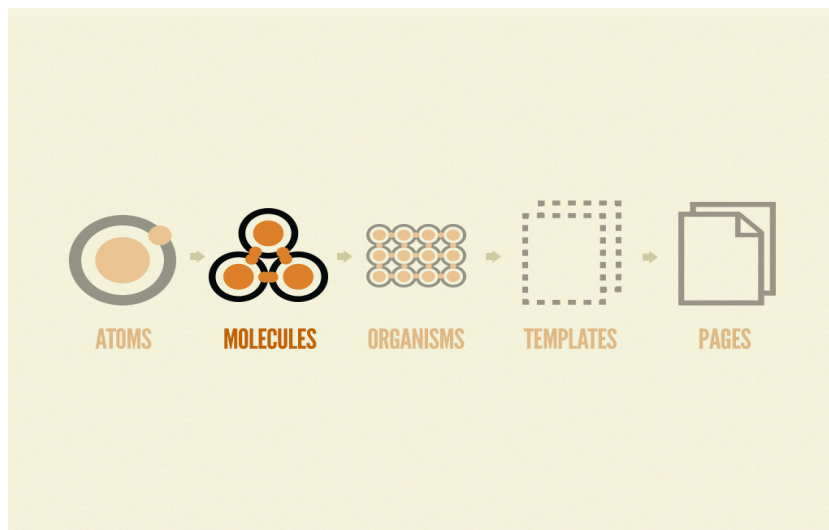


Figura 12: Atomic Design elements: Molecules

In interfaces, molecules are generally simple gatherings of user interface elements working altogether. For instance, a form label, search input, and button can combine

to make a search form molecule. At the point when joined, these abstract atoms out of nowhere have a purpose.

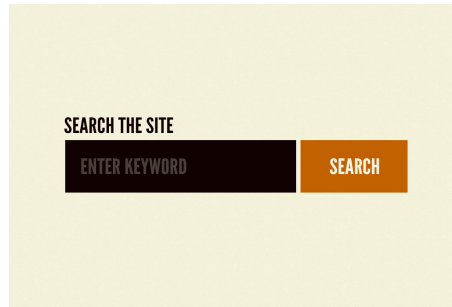


Figura 13: A search form molecule is composed inputs, labels, and buttons

The label atom presently characterizes the input. Tapping the button submits the form. The outcome is a straightforward, versatile, reusable segment that can be dropped in anyplace search feature is required.

### *Organisms*

Organisms are moderately perplexing user interface components made out of gatherings of molecules and additionally atoms as well as different organisms. These organisms form distinct areas of an interface.

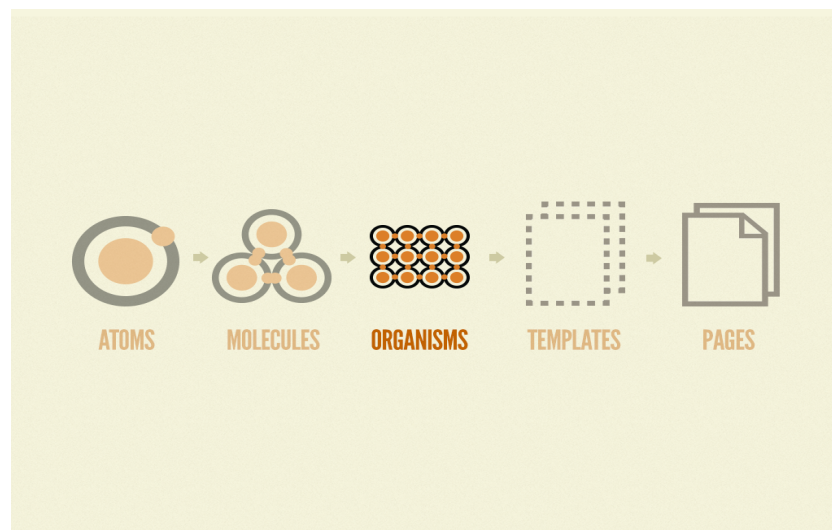


Figura 14: Atomic Design elements: Organisms

By going back to the search form molecule, it is usually found in the header of a lot of web applications. So, if we put these molecules in the context of an app's header, we will be talking about header organisms as presented in the figure below.

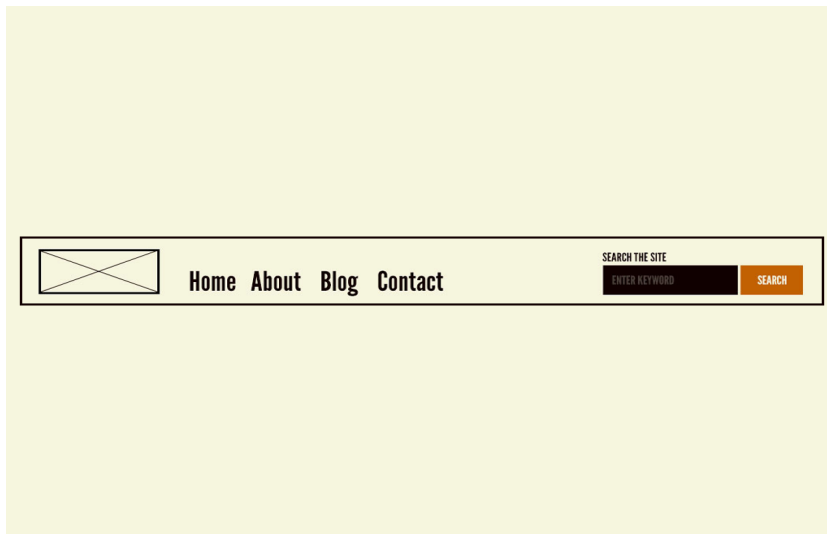


Figura 15: Header organism is composed of a search form molecule, logo atom, and primary navigation molecule.

The header forms an independent area of an interface, although it contains a few little blocks of interface with their novel properties and functionality. Organisms can comprise of comparable or diverse atom types. A header organism may comprise of divergent elements, for example, a logo picture, primary navigation list, and search form. These kinds of organisms are present on pretty much every available website.

*Templates*

Now, it's an ideal opportunity to bid farewell to our chemistry analogy. The language of atoms conveys with it a helpful hierarchy for us to purposely build the parts of our design system.

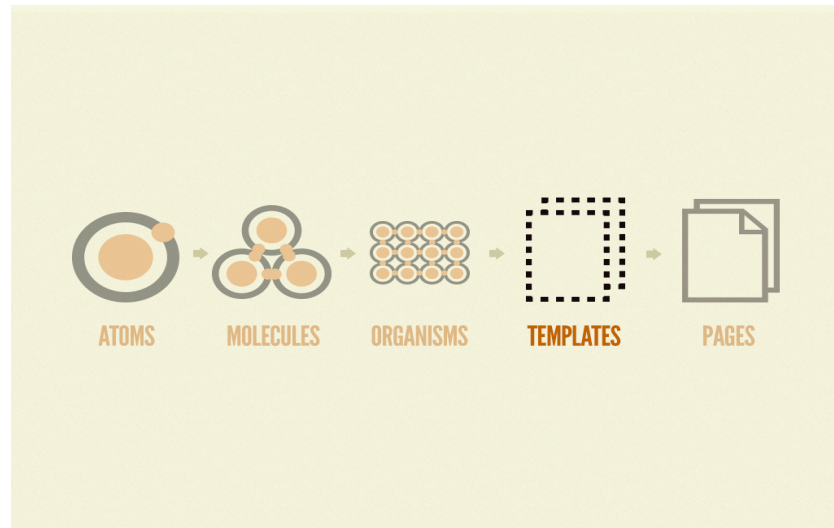


Figura 16: Atomic Design elements: Templates

Formats are page-level items that place components into a design and well-speak the design's hidden content structure. To expand on past models, we can take the header organism and apply it to a landing page format.

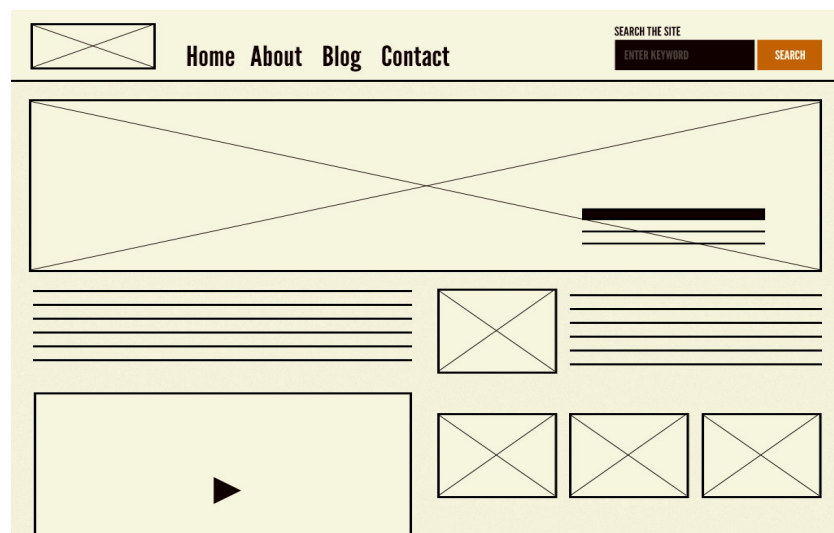


Figura 17: Atomic Design elements: Templates

The homepage template in the figure above shows all the important page components working together, which gives a setting to these moderately abstract molecules and organisms. While making a powerful design system, it's basic to exhibit what components look like and function together with regards to a format to demonstrate the parts signify a well-working whole.

Another significant attribute of templates is that they center around the page's fundamental content structure as opposed to the page's final content. Design systems must record for the dynamic idea of the content, so it's extremely useful to verbalize significant properties of components like picture sizes and character lengths for headings and text sections.

### *Pages*

Pages are explicit instances of templates that show what a user interface resembles with real content set up. Expanding on our past model, we can take the landing page template and pour some text, pictures, and media into the template to show real content in real life.

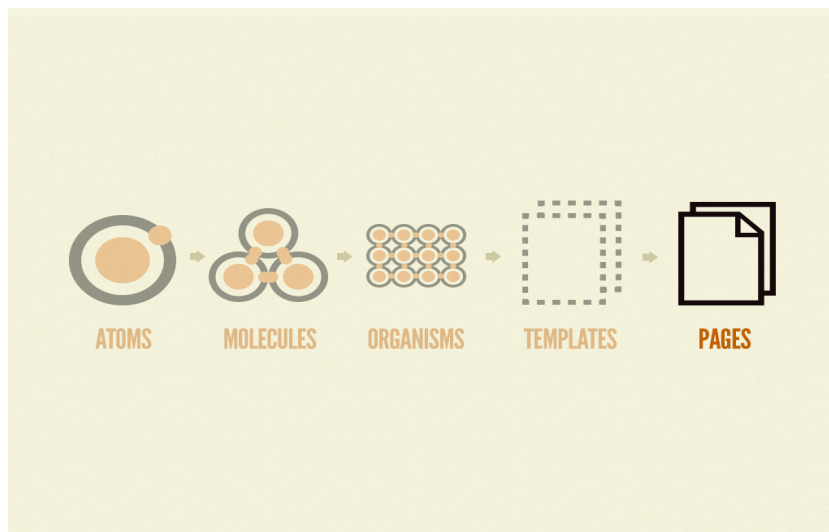


Figura 18: Atomic Design elements: Pages

The page stage is the most solid phase of atomic design, and it's significant for some clear reasons. All things considered, this is the thing that clients will see and interact with when they visit our product. This is the thing that our partners will close down. Also, this is the place we see each one of those components meeting up to form a delightful and useful UI.



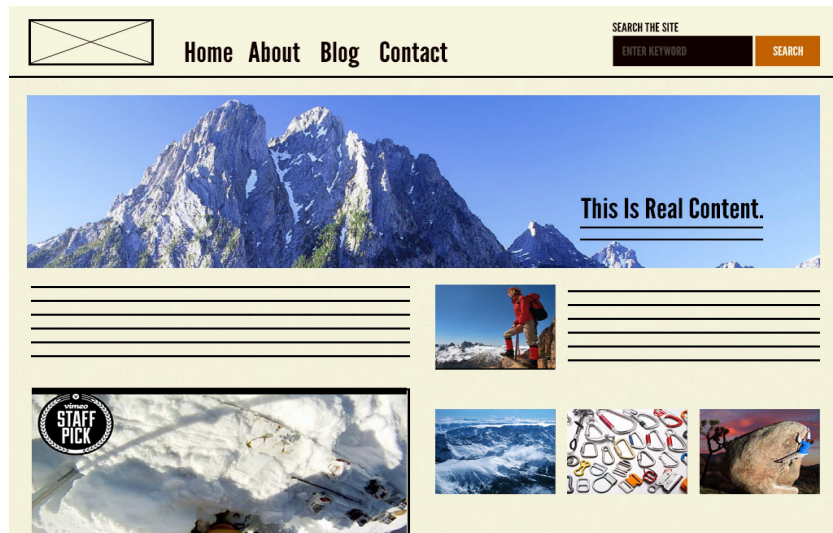


Figura 19: The page stage replaces placeholder content with real representative content to bring the design system to life

Besides demonstrating the last interface as our users will see it, pages are basic for testing the adequacy of the effective design system. It is at the page stage that we're ready to investigate how each one of those examples holds up when real content is applied to the design framework. To be sure that everything works as it should. If the answer is no, we can circle back and adjust our molecules, organisms, and templates.

And to conclude, the figure 20 is a great illustration of how Atomic Design was applied to Streamloan application.

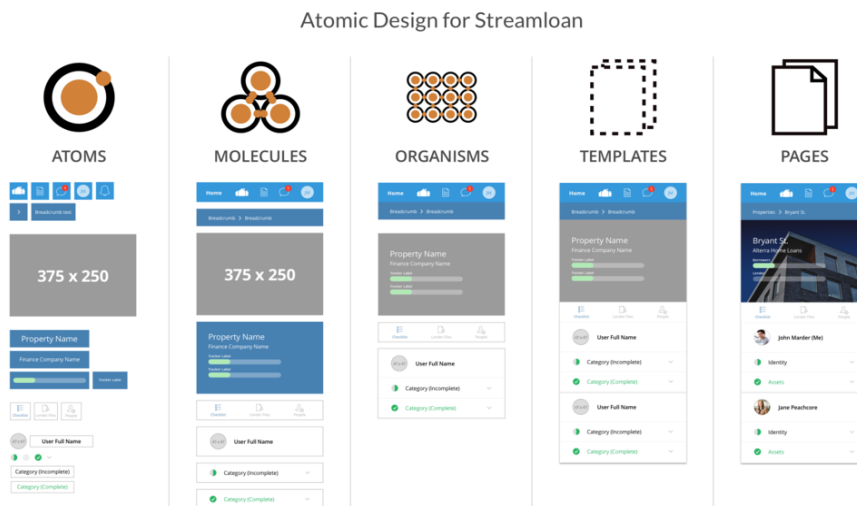


Figura 20: Atomic Design Methodology for Streamloan

## 2.3 FOUNDATIONS

Current design systems are the consequence of numerous long periods of advancement in the manner we compose front-end code. A few years ago most websites were worked with single-use, wasteful, delicate, and conflicting front-end codebases. Through hard-won experience and numerous long periods of joint effort and emphasis, front-end developers have built up more modern practices to compose and sort out the code. Presently, there is a blast of front-end structures and tools to support we write better, more viable HTML, CSS, and JavaScript.

Despite the innovations and tools behind them, an effective design system must follow certain core values. Consistency, how components are built, and overseen follows a predictable pattern. Self-contained, it should be treated as an independent dependency. Reusability, building component that can be used later in different contexts. Accessibility is important, application built with the design system will be used by as many people as possible. And the design system should be robust, ready to be applied in any product or platform with good performance and fewer bugs.[4]

### *Consistency*

The most significant task when beginning is to characterize the standards of our system, record them, and guarantee that everybody tails them. At the point when we have unmistakably documented code guidelines and best practices set up, designers and engineers can undoubtedly utilize and contribute to the design system.

### **Code Style Guides**

Code style guides give the grammar rules of syntax and semantics for the code. Code grammar is the arrangement of rules for organizing and designing the code (for example the curly braces always go on a new line). Code semantics give the standards to making the code reasonable (for example order CSS property assertions). Be that as it may, avoid being bogged down fighting over tabs and spaces. The most significant thing is to wind up with reliably composed code, not to accomplish hypothetical perfection.

## Automating Code Style

To enforce our code norms and accomplish consistency in our system, contributors must write code that follows the rules through linting and tooling. Linting is a process to analyze code and raise errors and warnings when the code doesn't follow the syntax rules or is broken, buggy, or malformed. There are a few different tools available on the market such as CSSLint and StyleLint for CSS and JSHint and ESLint for JavaScript. These tools can be run manually during the development process or launched automatically as an automated pre-commit hook before the code is checked into source control.

### *Self-Contained*

Our design system should be in a source control repository independent of our main codebase. Although it will require more work to get up and running, a separate repository brings many long-term benefits:

- Having versioned releases of our code can be needed later on
- Ability to share code across multiple teams and products
- Ease to build components isolated which will allow their usability
- Providing a good infrastructure for a powerful front-end testing architecture
- Forming a foundation for a living style guide website

An independent design system archive functions as a single source of truth. There is just one spot where components are characterized, which at that point gets shared into other code bases as a discrete dependency. Since all uses point back to a sanctioned implementation, changes in a solitary spot propagate through the whole system. In a perfect world, the entirety of the code for every part inside the system is co-founded: CSS, JavaScript, HTML formats, and documentation all live in a similar spot, conceivably the same directory. The closer the pieces are to one another, the simpler it is to follow and oversee dependencies between pieces of code, and the simpler it will be to refresh and keep up.

## Reusability

Successful design systems are exceptionally reusable. Bootstrap, the most-utilized front-end library ever, powers a large number of sites since it was architected in light of usability. Building components to be reused in various settings is imperatively significant, yet difficult to progress well-made components excessively engaged for a single use case or excessively inflexible, and users will wind up making their patterns as we can see in figure 21 [24] an example of Admin Template based on Bootstrap.

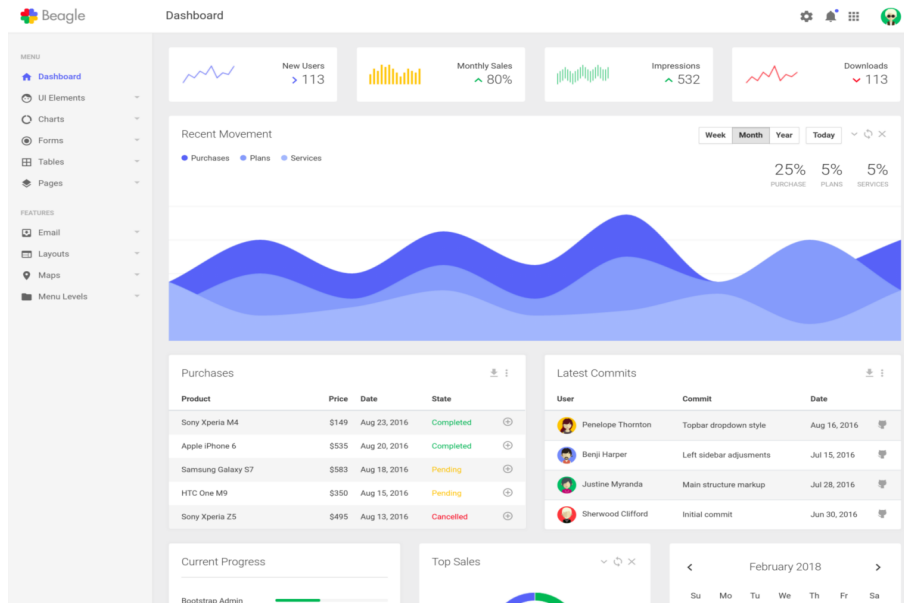


Figura 21: Beagle – Responsive Admin Template based on Bootstrap

To be reusable and adaptable, patterns should be particular, composable, nonexclusive, and adaptable. Composable components can be putten together to create new patterns, they should handle different use cases. Flexibility is important too for components so that be extended to work in different contexts. A fundamental best practice in the software engineering world is Don't Repeat our- self known as DRY. When two different blocks of code are doing the same job, the possibility of bugs is multiplied by two and the will increase as well the amount spent maintaining functionality. The main purpose of a good design system is to DRY up our development and reduce duplicating blocks of code by making them reusable.

*Accessibility*

For a long time, accessibility, or a11y, has been misconstrued as building products for a little group of clients of assistive innovation and far too often dismissed as excessively complex, also tedious, or "not our clients." Accessibility, however, isn't only for a small little group, yet for an expected 15% of individuals overall the planet. In the U.S only there are 56.7 million people with a wide range of lasting or impermanent visual, hear-able, and psychological impairments.

*The Web is fundamentally designed to work for all people, whatever their hardware, software, language, culture, location, or physical or mental ability. When the Web meets this goal, it is accessible to people with a diverse range of hearing, movement, sight, and cognitive ability.*

**WEB ACCESSIBILITY INITIATIVE (WAI)****W3C**

Fortunately, these attitudes are changing and our industry is grasping a more comprehensive meaning of accessibility. Making our site open to users with inabilities improves the experience for every individual who visits our site. If that isn't sufficient inspiration, improving our site's accessibility can help improve SEO, and it's getting progressively more significant from a legitimate stance to evade expensive lawsuits.

So, enforcing accessibility or a11y with our design system is a must. To make sure that our organization creates accessible features and apps, enforcing accessibility best practices is a need in the code blocks of the design system. And to be sure that is the case, we need to follow some practices :

- Test the color usage compared to the color contrast guidelines (as presented in the [Figure 22](#))
- Manufacture components to be keyboard and screen reader accessible by default. The eBay Accessibility MIND design library is an astounding, careful asset to help the direct improvement of available components and best practices. Urging contributors to work as indicated by these rules and test their code using keyboard-only navigation and assistive innovation gadgets like screen readers.
- Remember for our documentation code principles and rules for common a11y best practices, for example, using bigger, legible text sizes, continually partner

a form field with a label, and appropriately adding alt text attributes to pictures, to give some examples. Salesforce’s Lightning design system and Shopify’s Polaris are incredible examples of accessibility guidelines in practice.

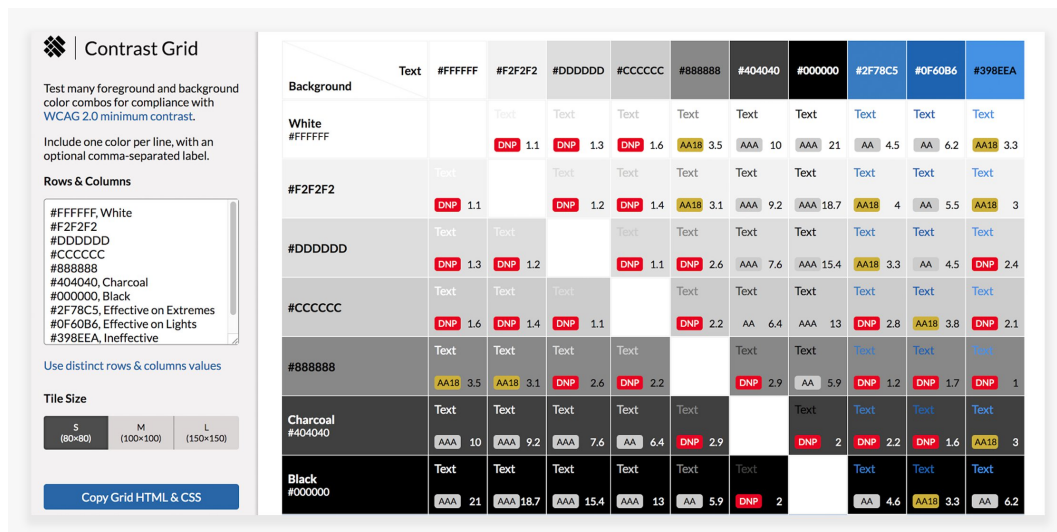


Figura 22: Contrast Grid [25]

These accessible practices improve ease of use for everybody by making it simpler to see, collaborate with, and explore a site, improving structure fulfillment rates and decreasing user mistakes. Use the intensity of our single wellspring of truth to make an establishment for accessibility, hence diminishing a portion of the weights from product teams. It’s a lot simpler to work in accessibility from the beginning than to jolt it on after a component has been designed and built.

### *Robust*

A robust design system has a solid establishment of tests behind it. Testing gives confidence in our code, which encourages adoption. Users will realize that they can redesign or change their user interfaces without it breaking in unforeseen ways. Also, our design system is remarkably situated to frame an establishment for powerfully testing our front-end code.

## Test our Design System

Staying up with the latest tests for pages, applications, and features especially on a fast-changing site or one with hefty experimentation requires a ton of work. We can limit the extent of our tests and increase more elevated levels of confidence in our site-wide front-end code by intensely testing our design system components. we as of now need to create a code model for the various conditions of every segment for our documentation use those as our test fixtures.

## Type of Tests

Regarding tests, the improvement of the software quality and the need to ensure that our product is good to go. Four types of test are being used by testing engineers for ensuring stability in their software:

- **Unit testing** These tests confirm that little units of code act like expected. Unit tests execute functions with a lot of predefined inputs, at that point confirm that they return the anticipated output. Some well-known frameworks use Mocha, Jasmine, and Jest.
- **Functional testing** In functional tests, fixtures of the code is run in a virtual "headless" browser, then tried by performing simulated user actions, and checking the new condition of the browser for few expected results. Useful testing structures incorporate Nightwatch and Protractor.
- **Visual regression testing** These tests help get unintended visual changes to components styles. The test system takes screen captures of our fixtures both when the changes, at that point looks at them using a calculation to recognize visual contrasts. There are open-source frameworks available like Wraith, Gemini, and BackstopJS.
- **Automated accessibility testing** Leverage tooling to guarantee that our components are accessible. A few alternatives for running automated a11y reviews are Paypal's AATT and a11y by Addy Osmani, and Ax by Deque Systems.

### *Documentation*

In design systems, many guidelines must be documented. These guidelines, as well as live code examples, will be the main documentation for the design system.

Providing clean and well-defined rules on how each component should be used, required properties, and styles option. The user should be able to get answers for a different question, what, when, where, and why in each component that is documented. Including all implementation rules are important to guide users in using the components properly, this means adding the class name, properties, and code snippets with a valid example.

If a live code example is provided this will have good effects on the usage guidelines and the technical rules they will be tied together to have a visual representation of the look and the functioning of the component. As we implement our system iteratively, it is gainful to get out the state of every component. Making a page that shows every component ready to be used. Tag components as new on the off chance that they have been recently included component gives the team an overview of the advancement being made to our system.

Writing documentation is a challenge, and staying up with the latest version is considerably harder. By following the rules, the created structure will use a similar component for both the design system and our product. At the point when a change is made to a component, the impact will be seen in the two spots. this stays up with the latest without having to manually make changes. Change the code in one spot, and our design system will refresh alongside our product.

Adding searchability to our documentation is very important, trying to find the documentation of a specific component can be a hard and time-consuming task. It will be easier for developers to have the search feature available to look for the exact feature that needs in the documentation.[7]



## **Keep the library code and documentation code closer**

Earlier, we examined storing our design system code in a different repository that works as our single source of truth. At the point when documentation and code are together, almost certainly, we'll make sure to make the necessary updates in the documentation when a component changes. Consider adding a pre-commit hook to our design system repository to alert contributors when they push their code changes and don't contain updated documentation.

## **Automate documentation**

Starting the documentation can be easy and simple, by beginning with simple and human-readable files written in markdown in the same location as each component. Github is already configured to display any file named README.md when we're viewing a repository's content on the web. When Creating a web site just for documentation, automation can be very important to simplify things. Instead of making a new code base for a separate project, we can use a tool that generates the documentation automatically and we will be reducing the amount of time that will be spent on writing and maintaining the code.

## 2.4 CREATING THE INVENTORY

To start the way toward scaling our design operations, we should comprehend the present status of our design and the improvement of the ecosystem. Making a total inventory requires significant investment. we should choose just scarcely any exercises to get the team ready and proceed at the point when we'll have the full team focused on the assignment [2].

### *Colors Inventory*

One of the important parts of the design is the arrangement of colors used over the system's portfolio as we can see in figure 23. Starting the design system creation with sprints dedicated to bringing together the color palette. Colors influence all the pieces of the system, so we need to arrange them first. Few rules should be followed for the creation of the color inventory:

- **Identifying the primary colors:** To identify base colors, usually it's the colors associated with the logo of the brand and the ones that appear more in the UI.
- **Listing all color variables used in CSS files:** All color variables should be listed so the developer will not waste time on finding the color code while creating components.
- **Organizing the colors:** Organizing the colors per category such as hues, shades, tones, tints, or just the similarity.
- **Few in one:** Counting the number of different colors and taking notes of anomalies (For instance, the different shades of grey).
- **Naming convention:** Deciding which approach to follow for naming the colors. Actual names (e.g. #b9b9b9 – silver), abstract names (e.g. #b9b9b9 – pigeon), or functional names (e.g. #b9b9b9 – silver-base).
- **Building accent palette:** Building the palette of secondary colors (e.g. lighter or darker green). It can be done by making arbitrary decisions or a functional approach that is less time-consuming and easier (for example, 20% darker silver will be @silver-darker-20 #CDCDCD).
- **Testing the palette:** Testing how the new palette is affecting the UIs is critical. For better feedback, inviting product designers to test the interface

and make suggestions for changes in the palette is important to improve the inventory and let them be part of the process.

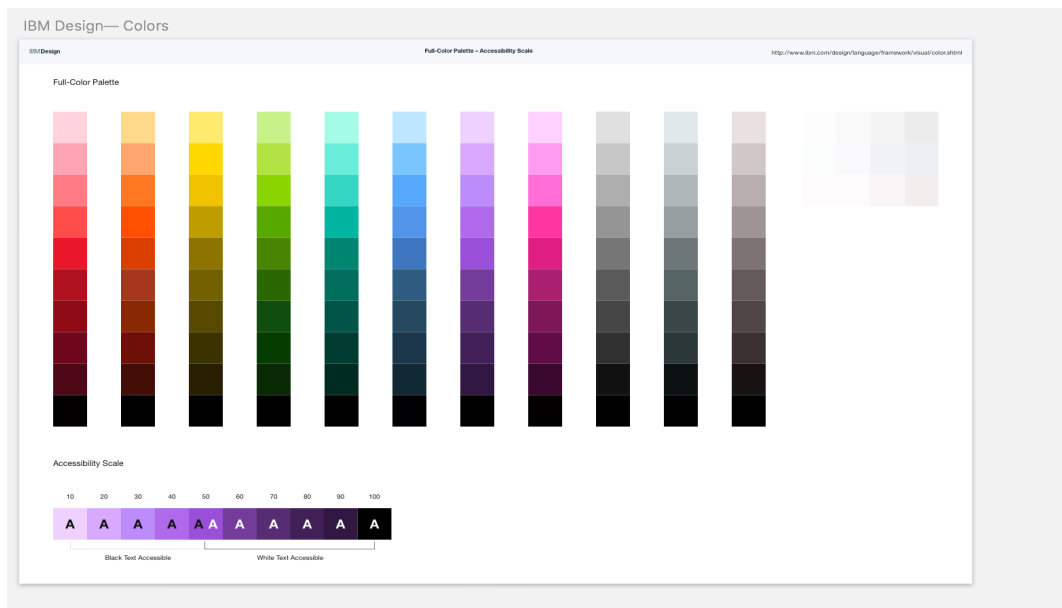


Figura 23: IBM Carbon Design system colors inventory

After testing and getting feedback about the color palette, finalizing the palette, and communicated it to the organization by adding to the design systems. It's important to document it in the documentation of the design system as well, especially if CSS preprocessors were used for the variables. Then this new palette should be delivered to the designers and their design tools such as UXPin, Sketch, etc [5][6].

### *Typography Inventory*

In a perplexing project, typography can rapidly get out of hand. The absence of reliable typographic scale used over the project makes the data engineering tangled and difficult to comprehend for clients, and it also builds the expense of the product maintenance because of the fragmentation of code. By inventorying text styles, we can see where styles become inconsistent a good representation from Material Design is in figure 24.

- **Building a consistent type scale:** Few approaches are available to build a typographic scale, by optimizing the scale to use existing styles or by building a harmonious scale using the golden ratio or major second.

- **Testing the new scale:** Comparing the new typographic scale with the text styles in the inventory, we should either match, replace, or merge the existing ones.
- **Implementing the new scale in CSS:** We should consider using preprocessors and mixins on a test server.
- **Testing the interface:** Checking how the interface is behaving with the new scale. Is everything readable? The styles are covered? Does it reinforce the right information architecture?

Scale Category	Typeface	Weight	Size	Case	Letter spacing
H1	Roboto	Light	96	Sentence	-1.5
H2	Roboto	Light	60	Sentence	-0.5
H3	Roboto	Regular	48	Sentence	0
H4	Roboto	Regular	34	Sentence	0.25
H5	Roboto	Regular	24	Sentence	0
H6	Roboto	Medium	20	Sentence	0.15
Subtitle 1	Roboto	Regular	16	Sentence	0.15
Subtitle 2	Roboto	Medium	14	Sentence	0.1
Body 1	Roboto	Regular	16	Sentence	0.5
Body 2	Roboto	Regular	14	Sentence	0.25
BUTTON	Roboto	Medium	14	All caps	1.25
Caption	Roboto	Regular	12	Sentence	0.4
OVERLINE	Roboto	Regular	10	All caps	1.5

Figura 24: Material Design Design system Typography inventory

After getting feedback, testing, and iterating the scale. Finalizing the typographic scale, making part of the design system documentation especially the name mixins if they are were used, and communicate it with the rest of the teams is most is the last step for the typography inventory creation.

### *Icons Inventory*

Icons give the essential setting to user experience and accelerate the acknowledgment of key activities in a large portion of the interfaces. Conflicting usage of icons can prompt outrageous confusion of users and increment the maintenance cost. By making an inventory of icons, will enable the team to comprehend the torment of not having a design system set up.[8]

- **Choosing icons from the UIs that will be part of the system.**
- **Choosing the method of icon management:** Discussing with the team the best way to manage and add icons in the design system.
- **Implementing on a test server:** Implementing the icons on a test server is the ideal way in case of technology change which is very common for most organizations to follow up the latest ones on the markets.
- **Deciding how to use the icons:** There few ways to use icons like Inlining SVGs, SVGs as URLs, icon fonts. It's important to decide with the development team which method will be followed to avoid confusion.
- **Marking inconsistencies:** Paying attention to small details like having two icons from different families in the same user interface or the same icons assigned to different actions.

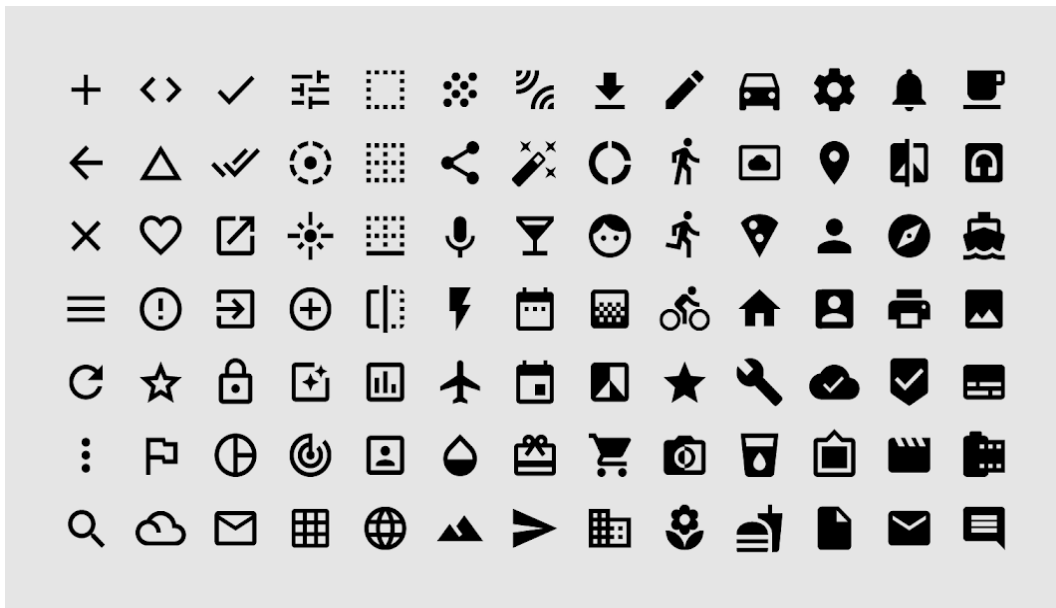


Figura 25: Airbnb Design system Icons inventory

Icons are a significant aspect of the visual language and ought to turn out to be important for the system. Settle the library, pick the usage strategy, and ensure that all colleagues have simple admittance to all icons.

*Spaces Inventory*

Space is the key element of any UI and designers and engineers need to oversee it effectively. Adding to the inventory various types of grids utilized over the products and maybe likewise jump profound into paddings in the containers to see any irregularities is very important and will facilitate the process of user interface building. Make a detailed list and documentation of the grid systems used in the design system is a must because it's the first thing the developers will look for while building a single page.

Token	rem	px	Example
<code>layout-01</code>	1	16	
<code>layout-02</code>	1.5	24	
<code>layout-03</code>	2	32	
<code>layout-04</code>	3	48	
<code>layout-05</code>	4	64	
<code>layout-06</code>	6	96	
<code>layout-07</code>	10	160	

Figura 26: IBM Carbon Design system spacing inventory

## 2.5 UI PATTERNS

If we dive in more for UI patterns user interface patterns, we find out that they are in the digital circle of sites, applications, and native mobile applications. They give a language to talk about interactive design. They offer function, interaction, and a purpose.

UI patterns document reusable pieces of an interface that have a purpose. To understand more UI patterns, we can see in the figure below, and an example from Bootstrap about a thumbnails component [1].

## Default example

By default, Bootstrap's thumbnails are designed to showcase linked images with minimal required markup.

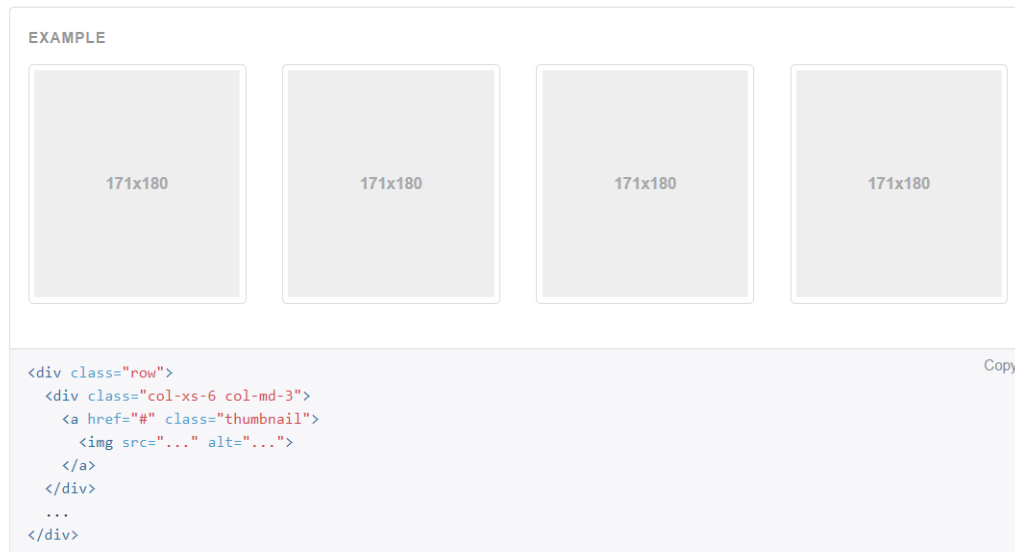


Figura 27: Bootstrap's thumbnails component (default example)

As we can see in figure 27 the thumbnail pattern or component as the developer's preferred term provides small image previews in a collection where each picture has a link to resources with a bigger size (High-resolution image). If the component is a preview of an item or a product, the link will take the user to a detail page. It can be a small preview of a video, it will redirect us to the appropriate player to watch it (e.g. wetube home page). And every pattern should have key features, for our example, the thumbnail will have :

- Small images
- Linked resources
- Collection to represent

UI patterns are more abstract than visual style. While patterns can regularly be recognized by visual likeness, these components exhibit it's not generally so natural: a pattern depicts behavior, which can be separated from effectively recognizable visual presentation. we can, for instance, apply a solid, sensational visual style or an inconspicuous, quieted flavor to a thumbnail collection.



*UI Patterns benefits*

Knowing the patterns and understanding the choices that went into them let us exploit the mounting intelligence of entire generations and businesses that invested time in these patterns, without rehashing an already solved problem. The little, reusable UI solutions found in these patterns would then be able to be formed together to make stronger and instinctive experiences.[16]

**Design efficiently**

Understanding patterns can assist we to design efficiently by rapidly perceiving the most ideal tool for the task, understanding the estimation of a few solutions, and solving the biggest number of issues at once. We can take the example of autocomplete search input, it helps users navigating the content of a website and recognizing what they are searching for without knowing the correct term or with wrong spelling. This allows them to select a result after typing few characters without wasting time on finding the exact name, learning this autocomplete component will allow us to decide when we need to use it and it will be the same with all UI patterns.

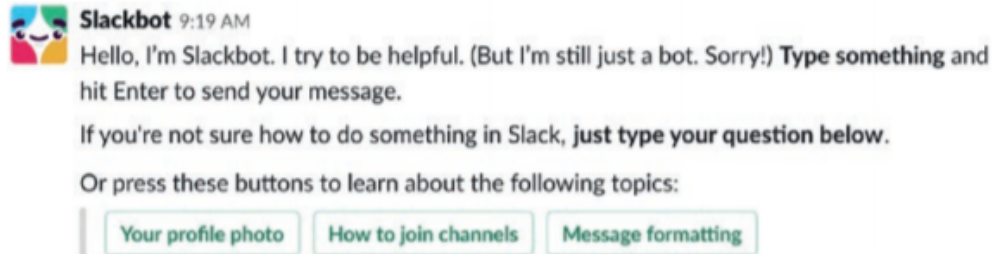


Figura 28: Screenshot of Slackbot from Slack

**Consistency**

Utilizing familiar patterns lets we encourage consistency. With the familiar thought of drag and droplet, we straightforwardly control an item by dragging and dropping it. A typical use of drag and drop is to transfer a picture by dragging it from our PC's file system to a target drop area in the interface. Most to-do applications let we drag and drop to-do things to reorder them or move them. The more inescapable drag and drop interfaces become over the Web, the almost certain users will understand the most effective method to collaborate with them.

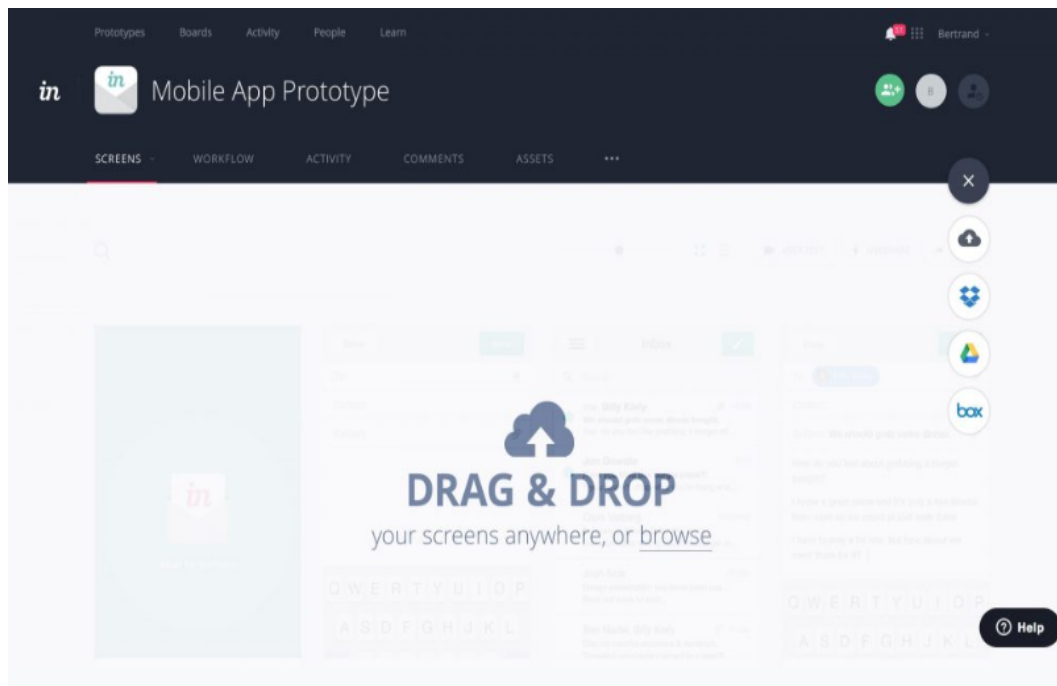


Figura 29: Screenshot of drag and drop option from Invision

## Reusability

By their common nature, patterns let us reuse design solutions both visually and inside code. Visual reiteration lets us incorporate consistency with our interfaces, making a learnable encounter for our users. Reusability in code additionally spares time, letting us refactor and improve existing features as opposed to rebuilding new ones. A pattern joins many design choices to tackle an issue, while programming encodes those decisions, patterns guarantee we Don't Repeat ourself (DRY) and making each design choice just a single time.

### *UI Patterns properties*

The UI pattern recipe is composed of three main ingredients :

- **Solution:** what the component do?
- **Problem:** why we need this pattern? What the problem that the user is facing?
- **Context:** when to use the pattern?

Based on the thumbnails component we can understand more these ingredients one by one. The solution is describing the component propose a collection of small picture preview linked to large-sized resources. The problem is that the user is having a problem navigating a huge collection of content and just selecting what he wants. The context is the user needs a preview of the picture before selecting or downloading a bigger file or watching a long video. To understand more, we can see in the figure 30 an example from the Pinterest home page that uses thumbnails to display the collection of pictures they are offering.

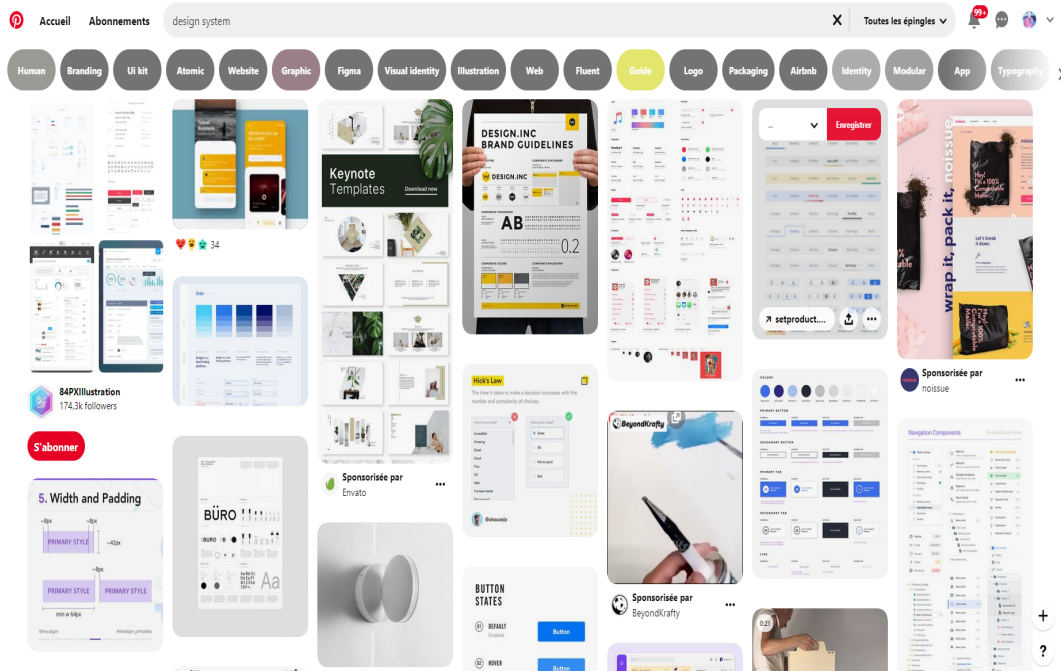


Figura 30: Screenshot from Pinterest

On the off chance that Pinterest consistently stacked high-resolution pictures at their full size rather than thumbnails, that would slow down the user experience. Pinterest needs to introduce thumbnails to make browsing smoother and help their users to discover more ideas. In an enormous collection like that, realizing what picture will show up next or if it's something the user needs to find in detail. By using thumbnails, we can rapidly peruse a bigger arrangement of choices before focusing on a particular item.

## 2.6 UI PATTERNS IN DESIGN SYSTEMS

To understand what are we going through in this section, as a start we should know what is a pattern library. A pattern library is an assortment of patterns, used to convey and improve design choices. This incorporates reusable solutions for issues centered around interaction and UX components.

As a digital professional, we may be comfortable with other design and code assets in the world related to patterns: style guides, style manuals, brand manuals, identity guidelines, front-end style guides, lawets, etc. Much of the time, they're complementary ideas that function admirably together. In other cases, we will find these together under the name "Pattern library" or "Design System". To get more comfortable with the concept and be able to answer the question when and how to use them? We should understand the differences between them.

### Style guides

Style guides, style manuals, center around the written word to set norms about communication styles to guarantee consistency in tone, selection of words, punctuation, grammar, and other language rules.



Figura 31: Screenshot from Monash University's editorial style guide

## Brand guides

Brand guides or visual style guides lean toward visual matters of a brand's identity, including logos and icons, color palettes, and typography. They are some of the time mixed with editorial style guides. One key distinction among editorial and brand style guides is that marking rules can be utilized by outer parties.



Figura 32: Screenshot of one of Instagram's brand guidelines

## Design guidelines

Design guidelines describe the visual language as particular from brand or visual style guides ordinarily address conceptual topics. For instance, Material Design characterizes a predictable metaphor to use throughout designs in the Material Design style.



Figura 33: Screenshot from the goals of Material design

## Front-end style guides

Front-end style guides are somewhat similar to patterns in real life inside an organization, transported with code snippets, design resources, and whatever else important to finish everyday design and development errands influencing the front-end of a product.

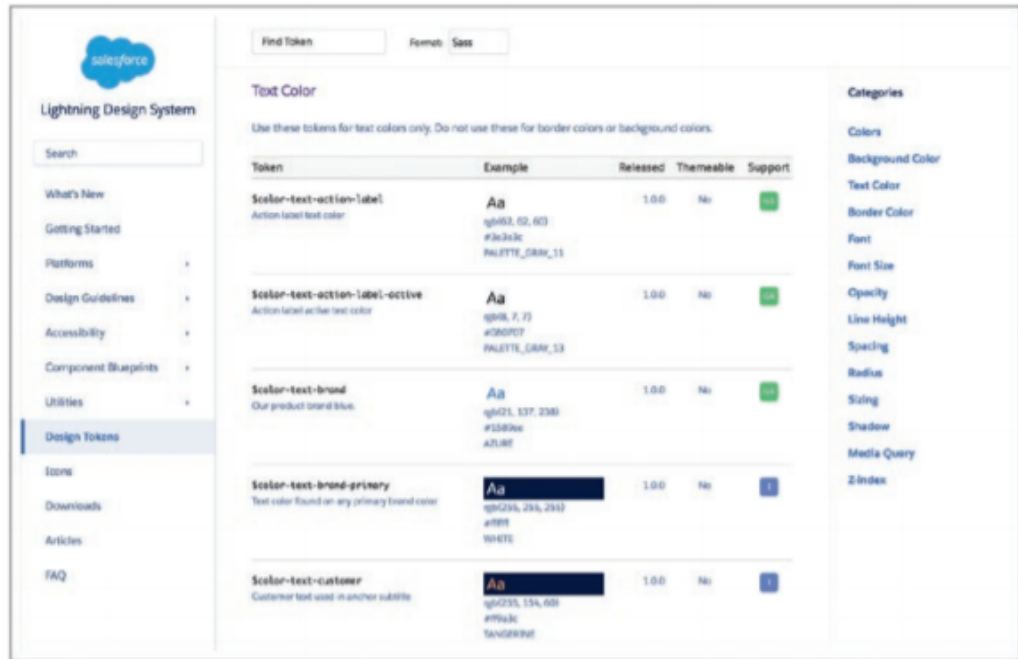


Figura 34: Screenshot of Lightning Design System Design Tokens

## Living style guides

Living style guides allude to guides that are in a state of harmony with the production environment. For instance, if we change a component in a living style guide and it will change in production over our whole site. They are made to give enough space for design thinking sharing about some elements such as typography decisions and keeping the guidelines on the same level with its actual execution.

## Code style guides

Code style guides or code standards frequently center around the code arranging and naming conventions of a product engineering team, for example, regardless of whether they use tabs or spaces to indent code and how they name strategies. Code style guides are frequently very separated from design matters. Accordingly, they're frequently put away independently from design-oriented style guides and design system resources, housed in a codebase README page or a code storehouse's wiki.

### *Documentation*

Practically speaking, instead of making pattern libraries and documenting patterns from scratch, organizations make design systems that encode patterns with a particular visual design language in component libraries with documentation. Some of the time they connect the tools and assets that address the broader pattern however once in a while, they do describe an abstract pattern, they describe a particular execution of a pattern in their particular domain. With regards to creating and documenting new patterns in a design system, we should focus the efforts on that one most essential one in the business, or the patterns that cause the most dispute in the organization. Introducing the component, mentioning the best practices, listing the properties, and how the user can customize based on his needs. For example, when and how to use a link, the props for a link component will be the URL of the link and how it will be showed (same page or new tab) and the user can customize the style related to the component (Changing the color of the link and the action when the mouse is over it).

### *Assembly*

When we have an unfinished version of all patterns, we can chip away at presenting them. A viable design system is educated by diverse contributors. Imagining a designer's inclination about developers using a design system when it is terrible and untouched by the designers that would have the skills to ensure its usability. Likewise, developers feeling about being compelled to rely upon a UI Kit that is constantly out of sync with production styles where some elements are outdated and some aren't fit to be turned out yet. As such, all parties contributing and using the design system must have the freedom to use it and add to it.

Some options that are available to ensure the patterns storage :

- **On cloud:** Having a folder on the cloud with the patterns collection is easily accessed but it might lack versioning.
- **Code repository:** The most preferred option by developers, this option will ensure versioning and full control with code security for updates.
- **Internal website:** a normal website with commenting on the features.



### Versioning

The designs, code, and basic UI patterns will change after some time. It's not always possible to reveal each pertinent change at the same time, such as moving a site starting with one front-end framework to a more powerful or newer one. As such, the design system may have numerous versions: one using the old framework for most patterns and one using a newer one for upgraded interactions. Like any software bundle, we could use semantic versioning to denote patches, minor changes, and significant breaking changes in the design system.

For instance, as we can see in the figure 35, the style guide of Walmart is suggesting the "Display Price" component which has been deprecated in favor of the more generic "Price" component.

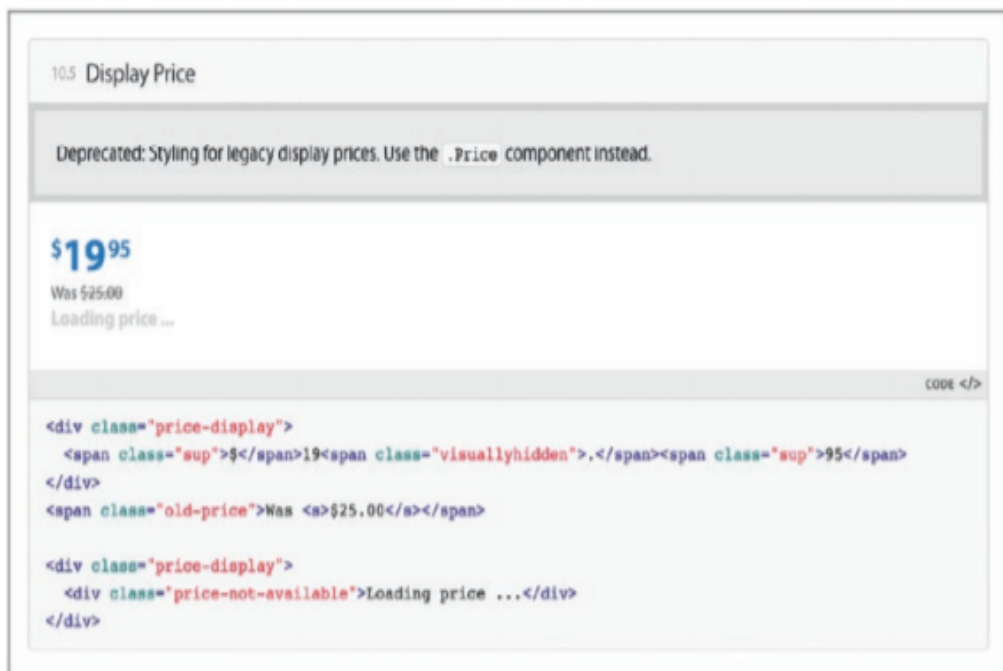


Figura 35: Screenshot of Walmart's Display Price component with deprecation warning

## COMPETITOR ANALYSIS

---

### INTRODUCTION

The impact of a design system on the company is very important on many levels this is why there is plenty of successful design systems available on the market. In this chapter, we will focus on comparing just three systems made by big companies such as Microsoft, IBM, and Google.

#### 3.1 IBM CARBON DESIGN SYSTEM

Even though IBM was once in the personal computing game with Apple and Microsoft, they have since moved concentration to enormous venture IT needs. They offer everything from business counseling to programming advancement administrations to IT facilitating and the board, to programming items to equipment (workers, centralized servers, stockpiling), and in any event, financing. The main features of the Carbon design system are Carbon tools and rich resources including design files for Sketch, Axure, and Adobe XD as well as resources for engineers.

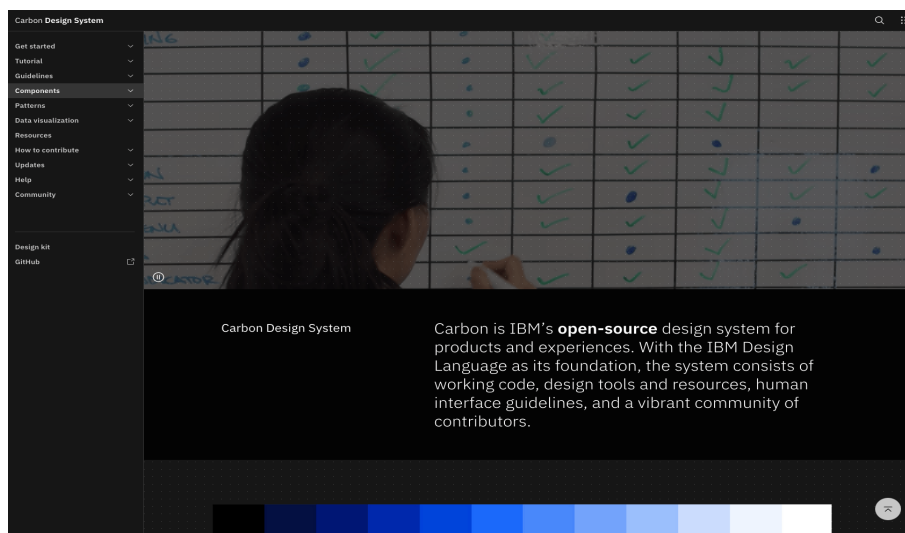


Figura 36: IBM Carbon Design System

Their philosophy that they have faith in progress—that by applying knowledge, reason, and science they can improve business, society, and the human condition. Given their scale and degree, they accept that great design isn't only a prerequisite, it's a more profound obligation to the individuals they serve and the connections they assemble.

### *Styles Inventories*

In this section, we will present what Carbon Design System offers in its color, typography, icons, and space inventories.

## **Color Inventory**

Color inventory defines the theme's colors and is useful for several parts of the system such as components and background.

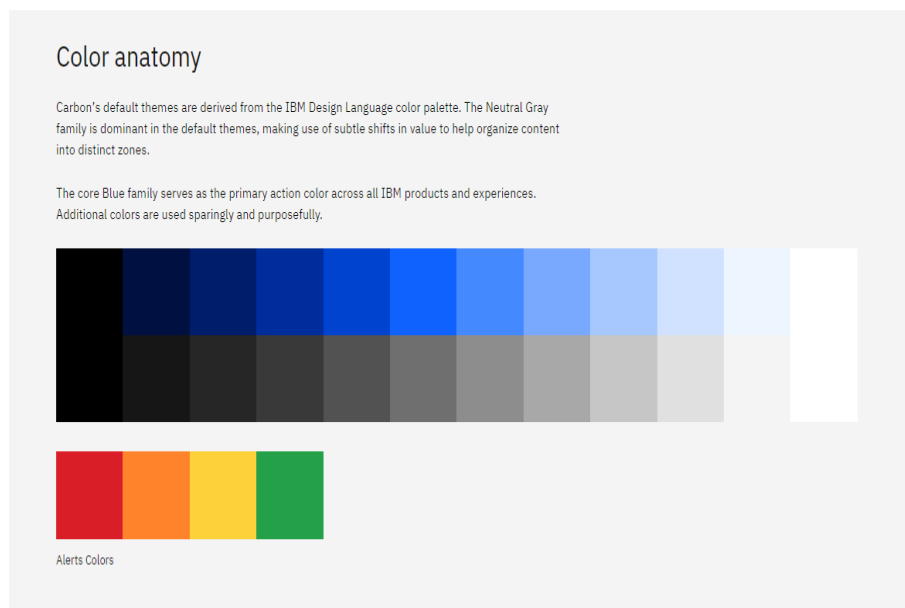


Figura 37: Carbon Design System - Color Inventory

## **Typography Inventory**

The typography inventory defines the fonts that are being used in the design system.

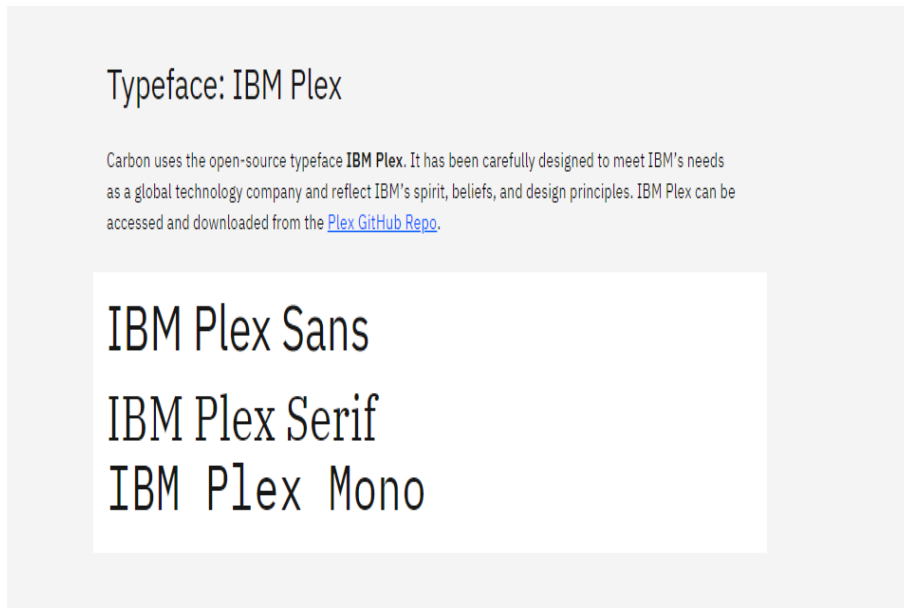


Figura 38: Carbon Design System - Typography Inventory

## Icons Inventory

The icons inventory or library is a set of icons that can be useful for developing some components such as Alerts and Buttons, they can be designed and created by the designer teams or by using an available library on the market.

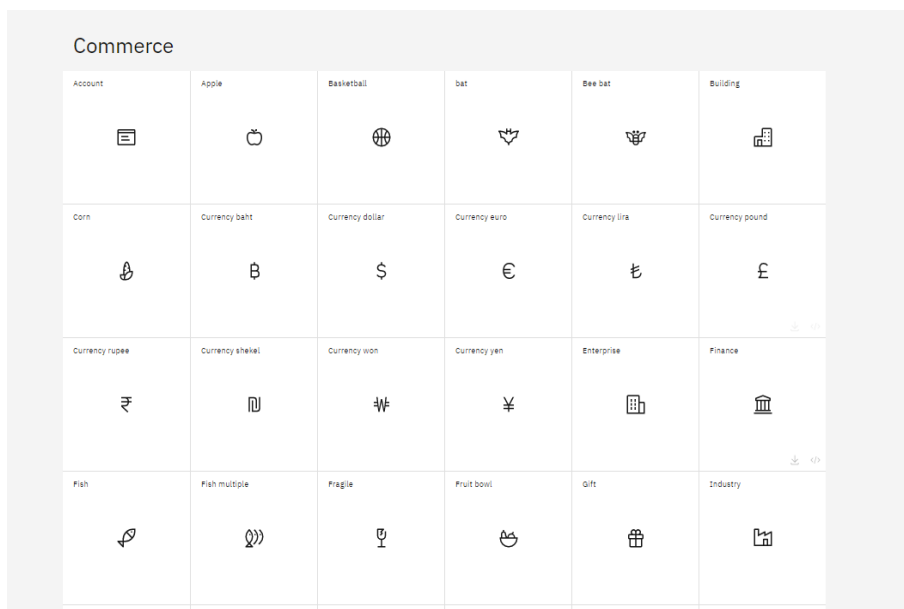


Figura 39: Carbon Design System - Icons Inventory

## Spaces Inventory

Defining spaces is very important for the grid and the organization of screen templates, they simplify the page creation and make the process faster.

**Spacing scale**

Use the spacing scale when building individual components. It includes small increments needed to create appropriate spatial relationships for detail-level designs. This scale is applied and used within all Carbon components.










Token	rem	px	Example
\$spacing-01	0.125	2	
\$spacing-02	0.25	4	
\$spacing-03	0.5	8	
\$spacing-04	0.75	12	
\$spacing-05	1	16	
\$spacing-06	1.5	24	
\$spacing-07	2	32	
\$spacing-08	2.5	40	
\$spacing-09	3	48	

Figura 40: Carbon Design System - Spaces Inventory

### *UI Patterns*

In this subsection, we will present the components library that Carbon Design has to offer. The system has 37 components with Storybook integrated where the user can have a live preview, test it with different values, and check code snippets.

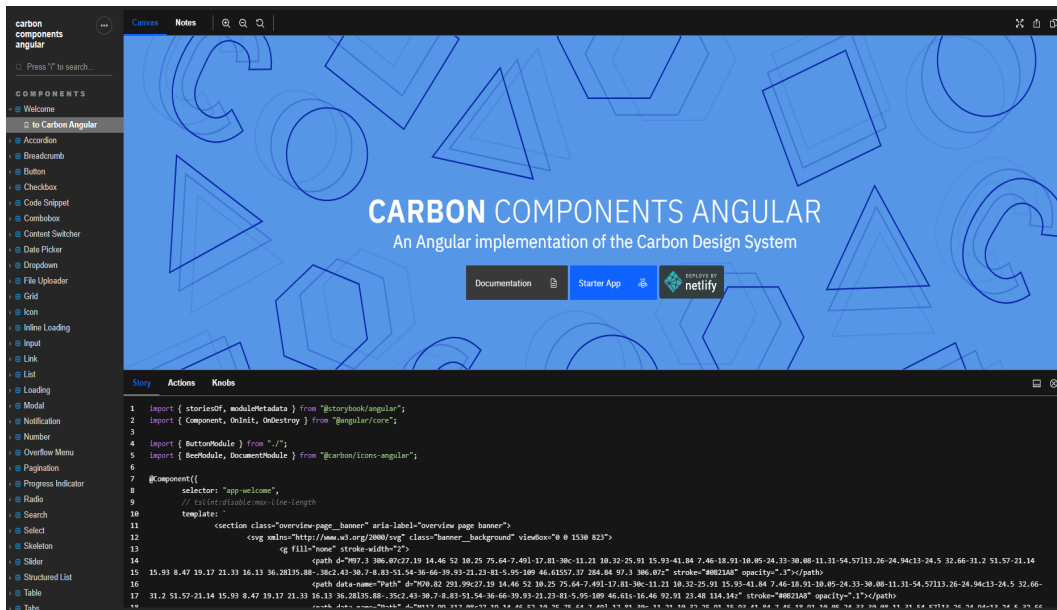


Figura 41: Carbon Design System - Storybook

### Documentation

Documentation is a very important part of a design system. A well-documented system will make the development process easier and faster and will make the development team more productive. The documentation is not implemented in the storybook but on the website of Carbon.

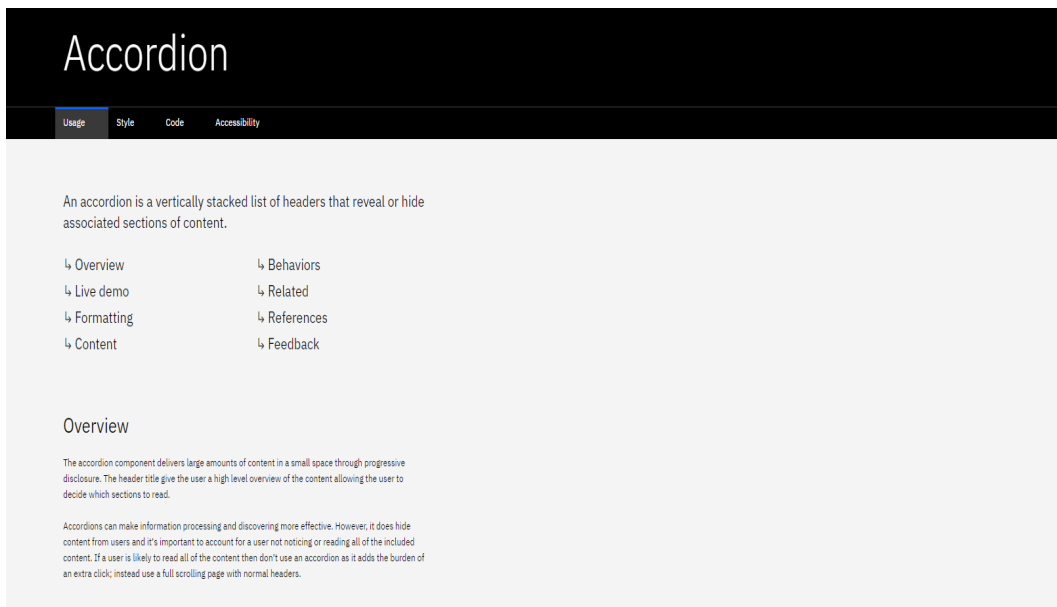


Figura 42: Carbon Design System - Documentation

The documentation provided by Carbon Design System is very rich and covers design and development. We can find when we should use the component, a live demo, alignment, placement, parameters, and different behaviors of the UI patterns.

### 3.2 GOOGLE MATERIAL DESIGN SYSTEM

The Google we know and use is a behemoth of an American innovation company that focuses on providing internet-related services and products such as web-based publicizing technologies, the famous search engine, cloud computing, for both mobile and desktop use and they are in hardware as well. It is viewed as one of the Big Four innovative organizations, with Amazon, Apple, and Microsoft. Regarding design systems, Google has Material Design to offer.

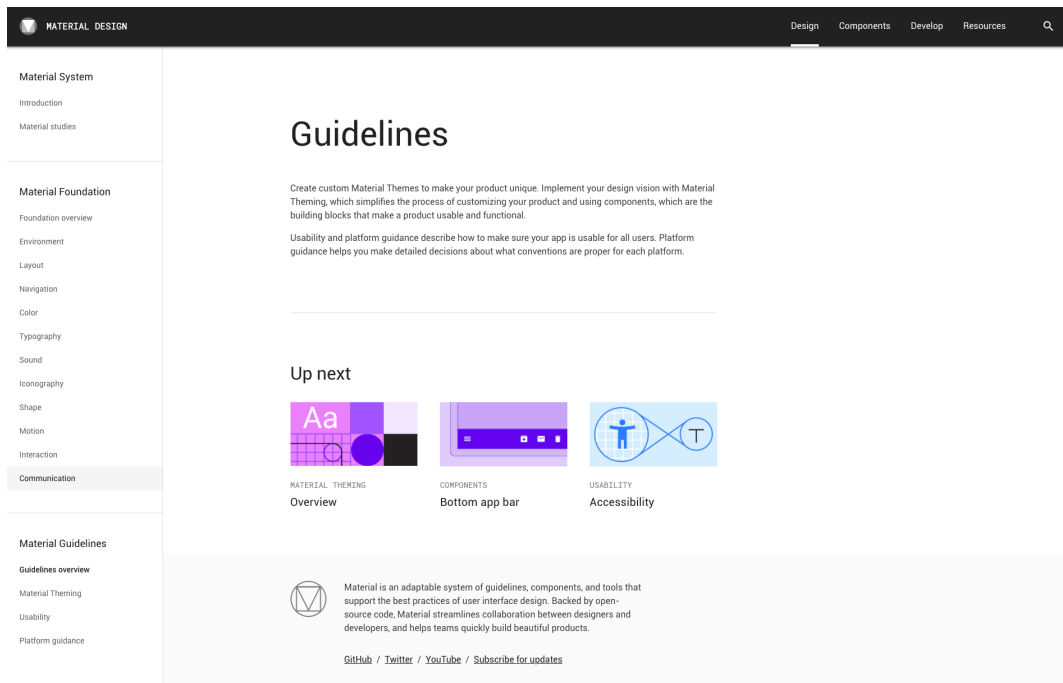


Figura 43: Google Material Design System

Their philosophy was pretty clear, regarding design, Google’s openly shared Material Design system prepared for some to follow. Absolute attention to detail arranged and cataloged components in a manner the design world has never so briefly observed. They committed the errors so users will not need to and provided requests and importance to the atomic design rules that all design systems are based on today.

## *Styles Inventories*

In this section, we will present what Material Design System offer in their color, typography, icons, and spaces inventories.

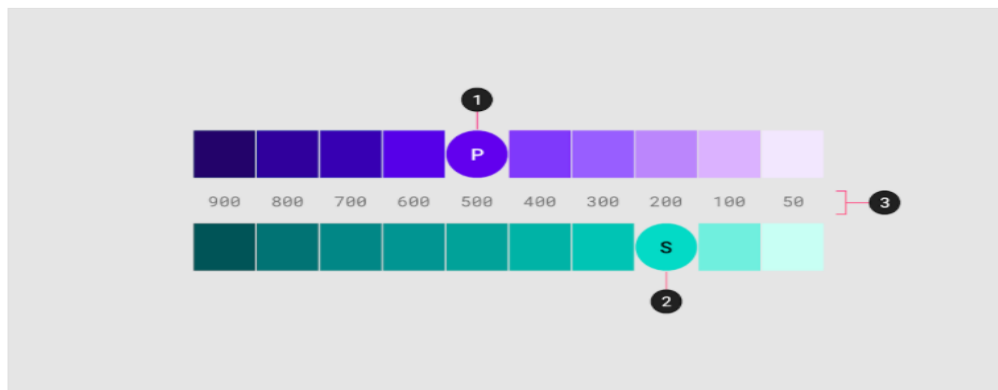
### **Color Inventory**

Color inventory defines the theme's colors and is useful for several parts of the system such as components and background.

#### **Colors and theming**

Color themes are designed to be harmonious, ensure accessible text, and distinguish UI elements and surfaces from one another.

The [Material Design palette tool](#) or 2014 Material Design palettes are available to help you select colors.



A sample primary and secondary palette

Figura 44: Material Design System - Color inventory

### **Typography Inventory**

The typography inventory defines the fonts that are being used in the design system.



### Example type scale

This example type scale uses the Roboto typeface for all headlines, subtitles, body, and captions, creating a cohesive typography experience. Hierarchy is communicated through differences in font weight (Light, Medium, Regular), size, letter spacing, and case.

Scale Category	Typeface	Weight	Size	Case	Letter spacing
H1	Roboto	Light	96	Sentence	-1.5
H2	Roboto	Light	60	Sentence	-0.5
H3	Roboto	Regular	48	Sentence	0
H4	Roboto	Regular	34	Sentence	0.25
H5	Roboto	Regular	24	Sentence	0

Figura 45: Material Design System - Typography inventory

### Icons Inventory

The icons inventory or library is a set of icons that can be useful for developing some components such as Alerts and Buttons, they can be designed and created by the designer teams or by using an available library on the market.

#### Design approach

The tactile and physical quality of Material is reflected in the design of Material icons. Each icon is cut, folded, and lit as paper would be, but represented by simple graphic elements. The quality of Material is sturdy, with clean folds and crisp edges. Surfaces interact with light through subtle highlights and consistent shadows.



Physical prototype



Lighting study

Figura 46: Material Design System - Icons inventory

### Spaces Inventory

Defining spaces is very important for the grid and the organization of screen templates, they simplify the page creation and make the process faster.

### Responsive layout grid

The Material Design responsive layout grid is an overarching guide to the placement of components and elements. The responsive layout grid adapts to screen sizes and orientation, ensuring consistency across layouts.

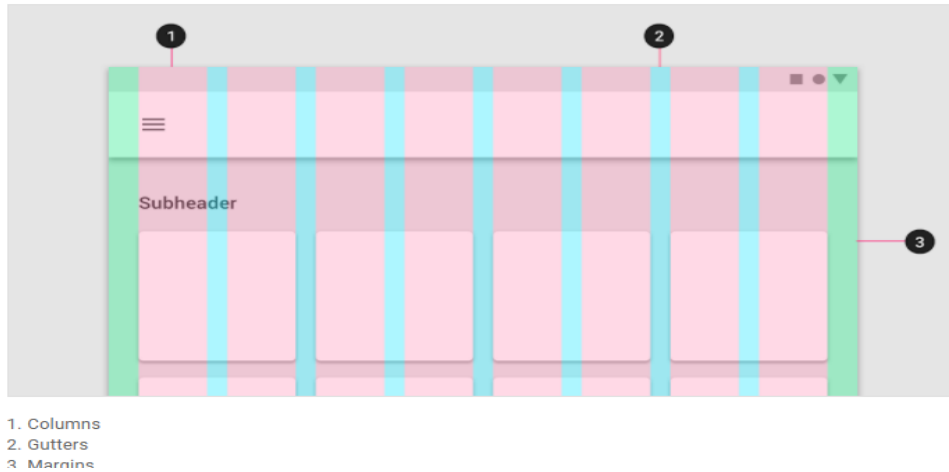


Figura 47: Material Design System - Spaces inventory

### UI Patterns

In this subsection, we will present the components library that Material Design has to offer. Material Design system has 28 components and they are not implementing Storybook to present and document the components but it's on the official website where the user can try a few different behaviors and configuration.

## Buttons

Buttons allow users to take actions, and make choices, with a single tap.

### Interactive demo

This demo lets you preview the button component, its variations, and configuration options. Each tab displays a different type of button.

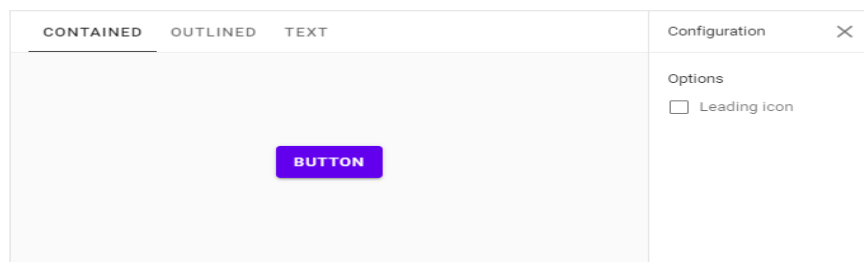


Figura 48: Material Design System - Button Component

*Documentation*

Documentation is a very important part of a design system. A well-documented system will make the development process easier and faster and will make the development team more productive. Material Design offers a live preview, hierarchy, and placement, theming, and different behaviour that UI patterns offer.

## Usage

Buttons communicate actions that users can take. They are typically placed throughout your UI, in places like:

- Dialogs
- Modal windows
- Forms
- Cards
- Toolbars

Figura 49: Carbon Design System - Documentation example

### 3.3 MICROSOFT FLUENT DESIGN SYSTEM

Microsoft is known for its ubiquitous computer operating system Windows, computer software such as Office, and the famous and first browser of all times Internet Explorer known as Microsoft Edge since 2015. Microsoft released in 2017 a design system that they called Fluent Design System.

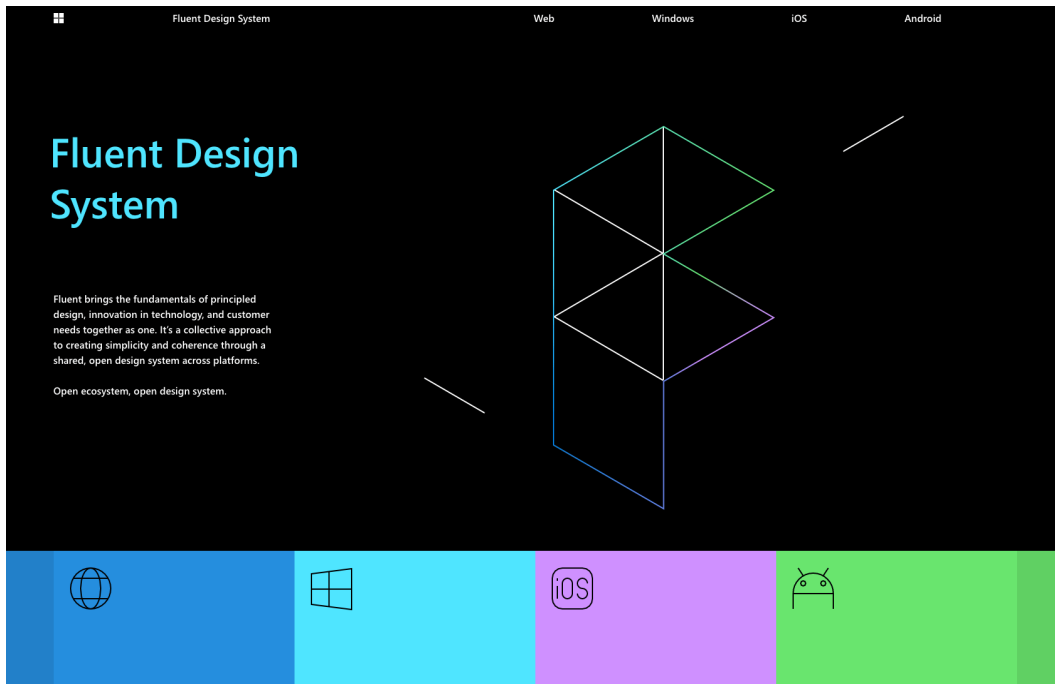


Figura 50: Microsoft Fluent Design System

Microsoft's Inclusive Design philosophy is resulting from digital environments. It empowers and draws on the full scope of human assorted variety. Above all, this implies including and gaining from people with a scope of viewpoints. They believe that avoidance happens when we tackle issues utilizing our own biases. As Microsoft designers, they search out those exclusions and use them as chances to make new thoughts and comprehensive designs.

### *Styles Inventories*

In this section, we will present what Fluent Design System offers in their color, typography, icons, and spaces inventories.

### **Color Inventory**

Color inventory defines the theme's colors and is useful for several parts of the system such as components and background.

## Color palettes

### Theme colors

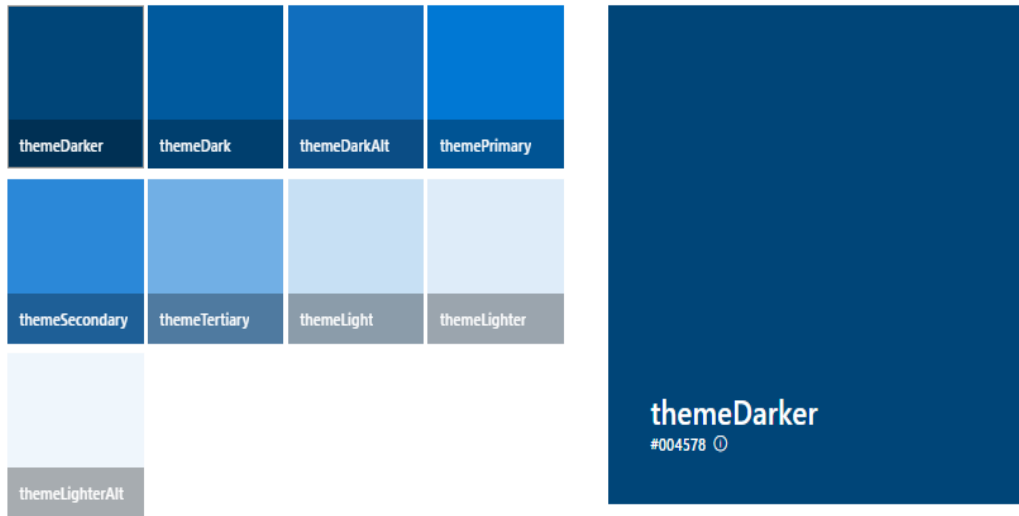


Figura 51: Microsoft Fluent Design System - Color Inventory

## Typography Inventory

The typography inventory defines the fonts that are being used in the design system.

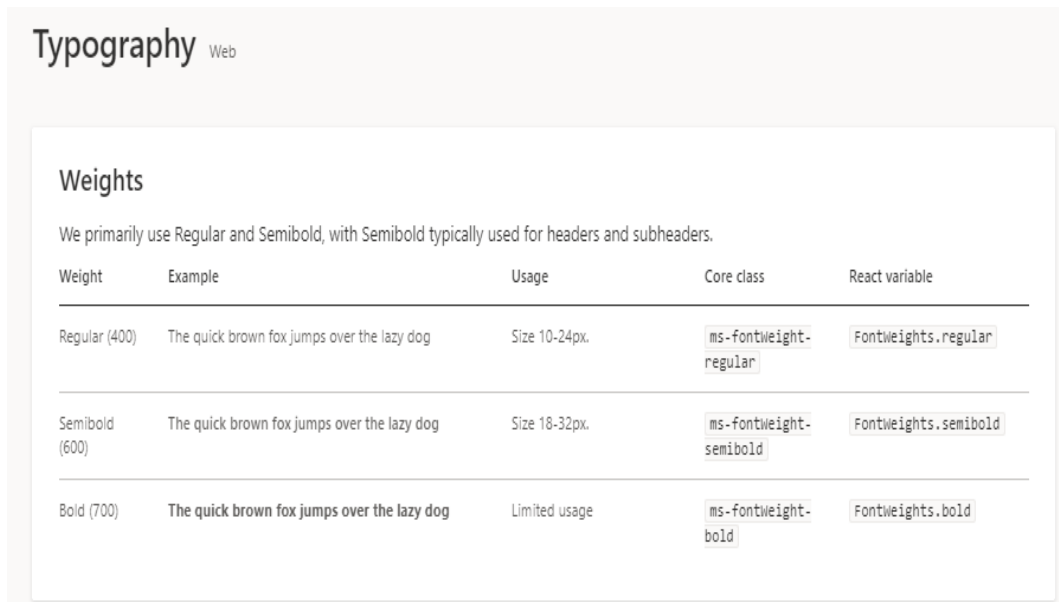


Figura 52: Microsoft Fluent Design System - Typography Inventory

**Icons Inventory** The icons inventory or library is a set of icons that can be useful for developing some components such as Alerts and Buttons, they can be designed and created by the designer teams or by using an available library on the market.

### Spaces Inventory

Defining spaces is very important for the grid and the organization of screen templates, they simplify the page creation and make the process faster.

#### Grid

Fabric Core comes with a mobile-first, 12-column, responsive grid that you can use to create flexible layouts for a variety of screen sizes and device types.

Note that this grid is only available via Fabric Core CSS. If you're not using Fabric Core, Fluent UI React's [Stack](#) can cover some of the same use cases, or you can use [CSS grid](#).

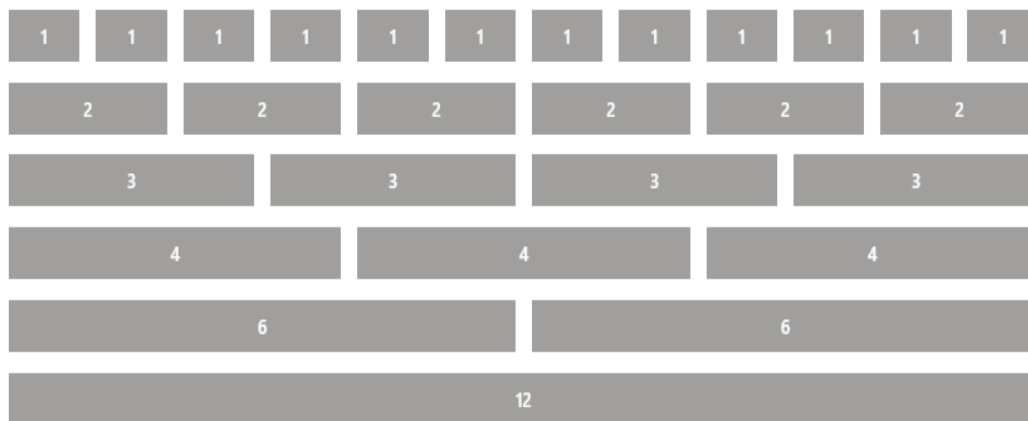


Figura 53: Microsoft Fluent Design System - Spaces Inventory

#### *UI Patterns*

In this subsection, we will present the components library that Fluent Design has to offer. Microsoft is providing 56 reusable components in their design system.

#### *Documentation*

Documentation is very important part of a design system. A well-documented system will make the development process easier and faster and will make the

development team more productive. Fluent UI is offering a very rich documentation with usage, best practices, and implementation of the components.

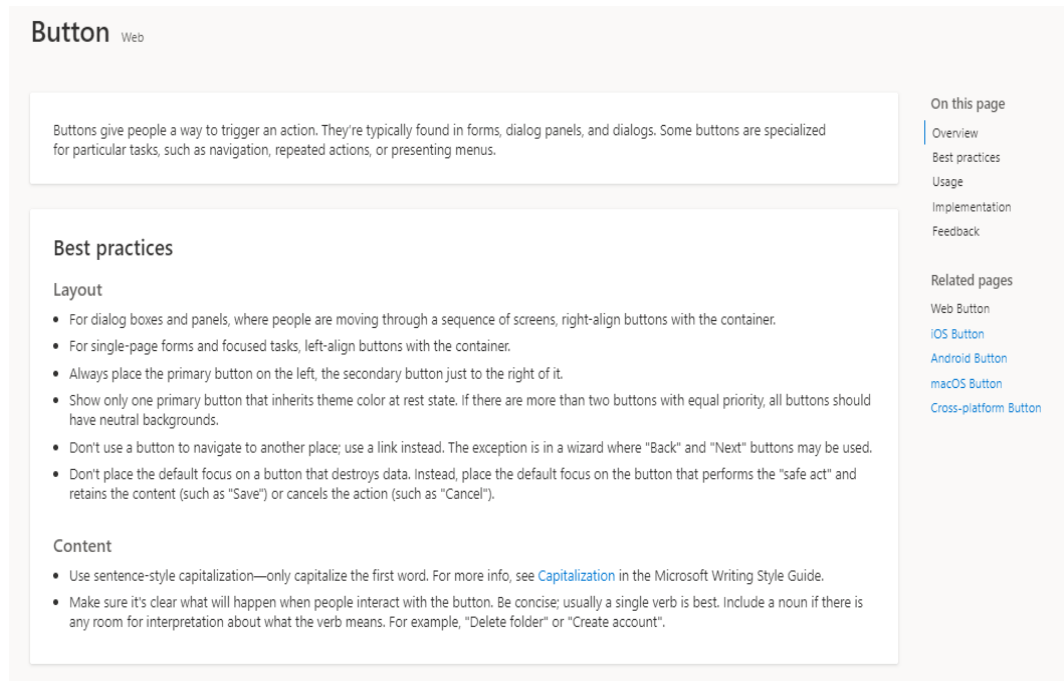


Figura 54: Microsoft Fluent Design System - Documentation

### 3.4 FOUNDATIONS COMPARISON

In this section, we will compare the three systems following certain core values that approve their effectiveness as we can see in the figure 55

Values	Carbon Design System	Material Design System	Fluent Design System
<b>Self-contained</b>	Version 10 704 releases	Version 8.0.0 612 releases	Version 7.153.2 9232 releases
<b>Reusability</b>	Reusable components	Reusable components	Reusable components
<b>Accessibility</b>	Blind users Low vision users Color-Blind users Hard-of-hearing users Physical disabilities Cognitive disabilities	Screen Reader Directional Controller Clear Visual Hierarchy	Not mentionned on the website.
<b>Robust</b>	Tests available	Test available	Tests available

Figura 55: Foundation comparison of Carbon, Material and Fluent Design Systems

### 3.5 OVERALL COMPARISON

We conclude this chapter with an overall comparison. We will present a comparison in figure 56 what these three design systems have in terms of features and advantages.

Attributes	Carbon Design System	Material Design System	Fluent Design System
<b>Cost</b>	Free	Free	Free
<b>Platforms</b>	Web - Angular - React - VueJS	Web - Android - iOS Flutter - cross-platform	Web - Windows - iOS Android - MacOS cross-platform
<b>Documentation</b>	Live demo When to use Alignment Placement Possible behaviours	Live preview Hierarchy Placement Theming Possible behaviours	Usage Best Practices Implementation
<b>UI usability</b>	YES	YES	YES

Figura 56: Overall comparison of Carbon, Material and Fluent Design Systems





## ROCKET DESIGN SYSTEM DEVELOPMENT

---

In this chapter, we will be discussing how Rocket Design System was implemented during my internship. What are the steps that the team followed to develop the product and comparing the available design systems on the market?

### 4.1 WORK ENVIRONMENT

In this section, we will discuss the work environment. By presenting both software and hardware tool used to build Rocket Design System.

#### *Hardware Environment*

In this subsection, we will present the hardware specification of the machine used for the development process.

- **CPU:** Intel(R) Core (TM) i5-7300 CPU@2.5GHZ
- **RAM:** 16GB
- **SSD:** 256GB
- **OS:** Windows 10

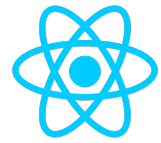
#### *Software Environment*

The rise of web development, gave us the opportunity to have several frameworks and technologies. In this subsection, we will be presenting the technologies used to elaborate our design system.

## *Frameworks*

### *ReactJS*

ReactJS is an open-source Javascript library for front-end web development. To build user interfaces and components. It is maintained by Facebook since 2013. This framework can be used for the development of single-page and mobile applications. It is only taking care of rendering data to the DOM but there are few libraries available for other purposes such as Redux for state management and React Router for routing.



### *Storybook*

A storybook is an open-source tool that helps in the process of developing UI components by offering few features to create them in isolation and create documentation and live preview with the possibility for the users to interact.



## *Programming Languages*

### *HTML*

Hypertext Markup Language or more known as HTML is the standard markup language for documents destined to be displayed in web browsers. It can be joint with other technologies such as CSS and JavaScript to add more styles and interactions that can be limitations for the markup language.



### *CSS*

Cascading Style Sheets known as CSS is a style sheet language used to describe how a Markup language document is presented on a web browser. It is designed to separate the presentation and the content by including layout, colors, and fonts.



*SASS*

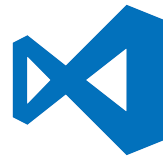
Syntactically awesome style sheets also known as SASS is a preprocessor scripting language that is compiled to Cascading Style sheets (CSS). Sassscript is the scripting language itself.

*Javascript*

JavaScript often abbreviated as JS, is a programming language that conforms to the ECMAScript specification. It is high-level and often just-in-time compiled.

*Softwares**Visual Studio Code*

Visual Studio Code is a source code editor developed by Microsoft for Windows, Linux, and Mac operating systems. It has more features than just editing code such as debugging, syntax check, code completion, snippets, code refactoring, and embedded Git.

*Google Chrome*

Google chrome is a multi-platform web browser developed by Google. It's supported on Android, Windows, Linux, macOS, Chrome OS, and iOS operating systems.



## 4.2 CREATING INVENTORY

Creating an inventory is a mandatory step in the design system creation. This includes color, typography, spaces, and icons. Since the company is an official partner for OutSystems low code platform, they got their inspiration from the styles of this last. The company's goal was to create a design system on two different frameworks to attract clients by having more options to offer. The company had the styles inventory stored in a big CSS

file, presented in figure 57, inspired by OutSystems with more content and rules defined by the designer's team.

```

8541 }
8542
8543 .fade-enter.fade-enter-active .content {
8544   opacity: 1;
8545 }
8546
8547
8548 .fade-leave.fade-leave-active .header {
8549   transition: none;
8550   opacity: 0;
8551   transform: translateY(-200px);
8552 }
8553
8554 .screen-container.fade-leave {
8555   position: fixed;
8556   top: 0;
8557   width: 100%;
8558   z-index: -1;
8559 }
8560
8561 /* Bottom and Top Transitions */
8562 .slide-from-bottom-leave.slide-from-bottom-leave-active {
8563   -webkit-transform: translateY(-30%) translateZ(0);
8564   transform: translateY(-30vh) translateZ(0);
8565   opacity: 0;
8566 }
8567
8568 .slide-from-top-leave.slide-from-top-leave-active {
8569   -webkit-transform: translateY(30%) translateZ(0);
8570   transform: translateY(30vh) translateZ(0);
8571   opacity: 0;
8572 }
8573

```

Figura 57: Screenshot of the CSS file

This approach was not very helpful specially to customize or update any part of the inventory because developers need to spend some time finding what they need and the risk to duplicate classes was high. So, an improvement was needed to have a better solution. The idea was to divide the file into parts and create a script to reunite these parts and generate a new version of the CSS file.



Figura 58: Screenshot of the folder of the new structure

Inspired by the SASS 7-1 pattern, this big file was divided into seven folders and the main file as illustrated in figure 58. These folders have several files that have rules and classes converted from CSS to SASS and the main file imports all these files for a Gulp script implemented to generate a CSS file that the teams will use for the styles inventory.

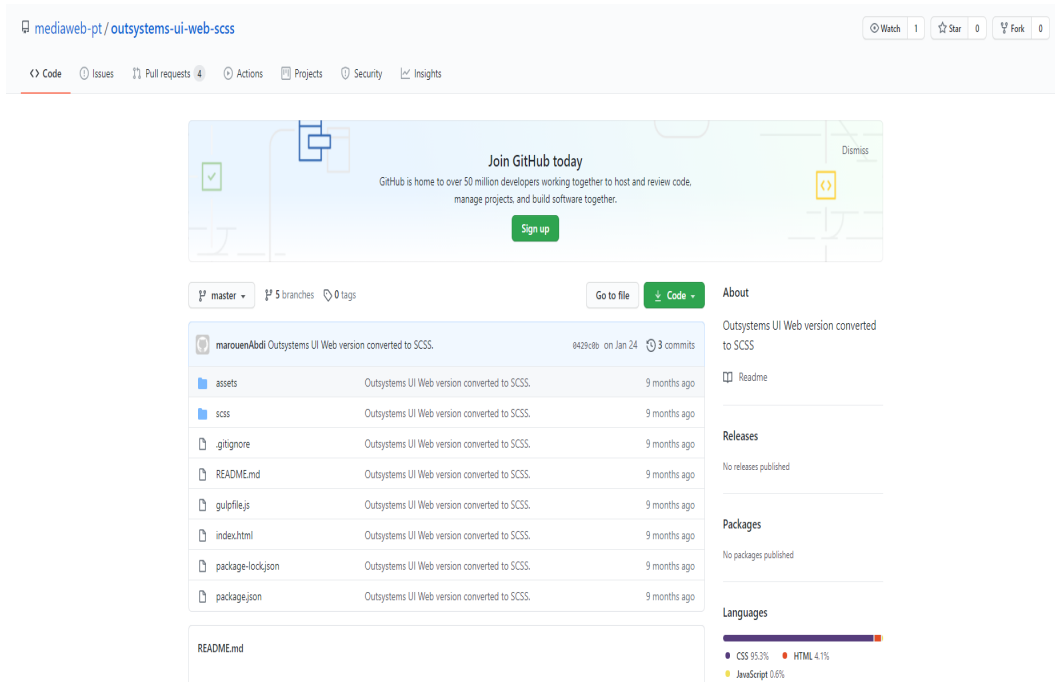


Figura 59: Screenshot of the script project shared as open source on Mediaweb github account

By following this approach, updating or changing anything in the styles inventory will be easier and less time-consuming. After making the updates we just need to launch the script and a new CSS file with the updates will be generated.

### 4.3 DOCUMENTATION

After the inventory was set, the team went through a design sprint for a week to define a few points :

- Onboarding and defining the challenge.
- Defining the UI Patterns that will be in the system.
- Defining documentation requirements.
- Creating documentation.

During this design sprint, the team was able to discuss and choose the library of the UI Patterns that will be used in Rocket Design System, researched and compared the content of other design systems on the market, and started creating the documentation of the UI patterns and

the inventories. For the implementation of the documentation, Storybook was the tool to make it is open-source, and being isolated from the main project made it the best option to have while doing the research phase.

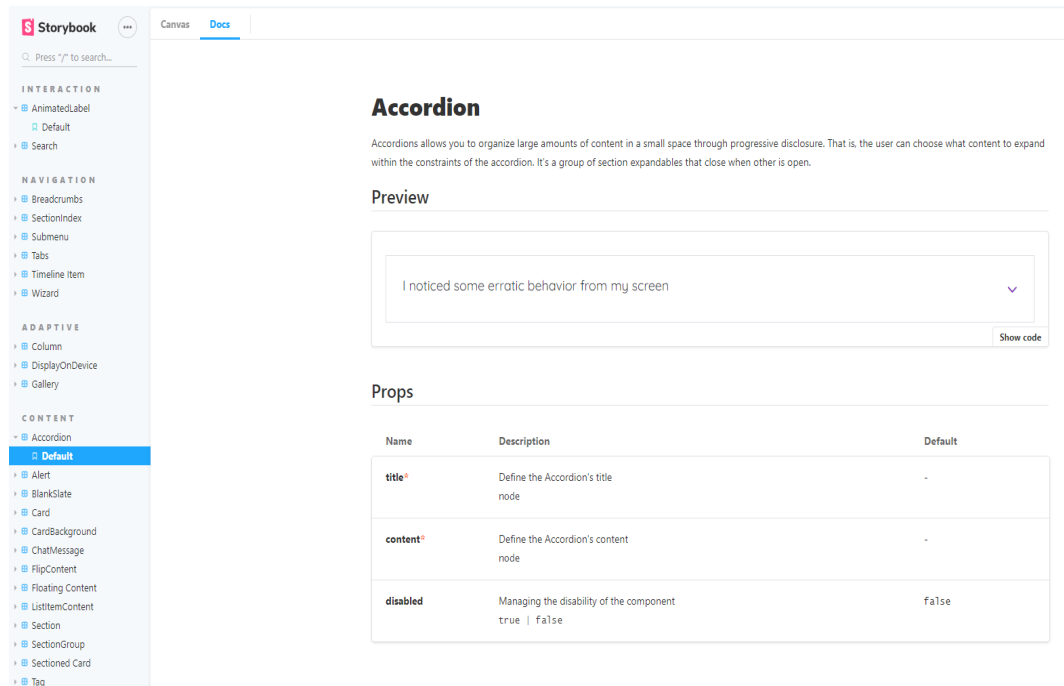


Figura 60: Screenshot of the documentation of the Accordion Component

And not only that, the Storybook offers a preview option with the possibility for the users to interact with the component and make few tests by changing the values of their properties as we can see in figure 61.

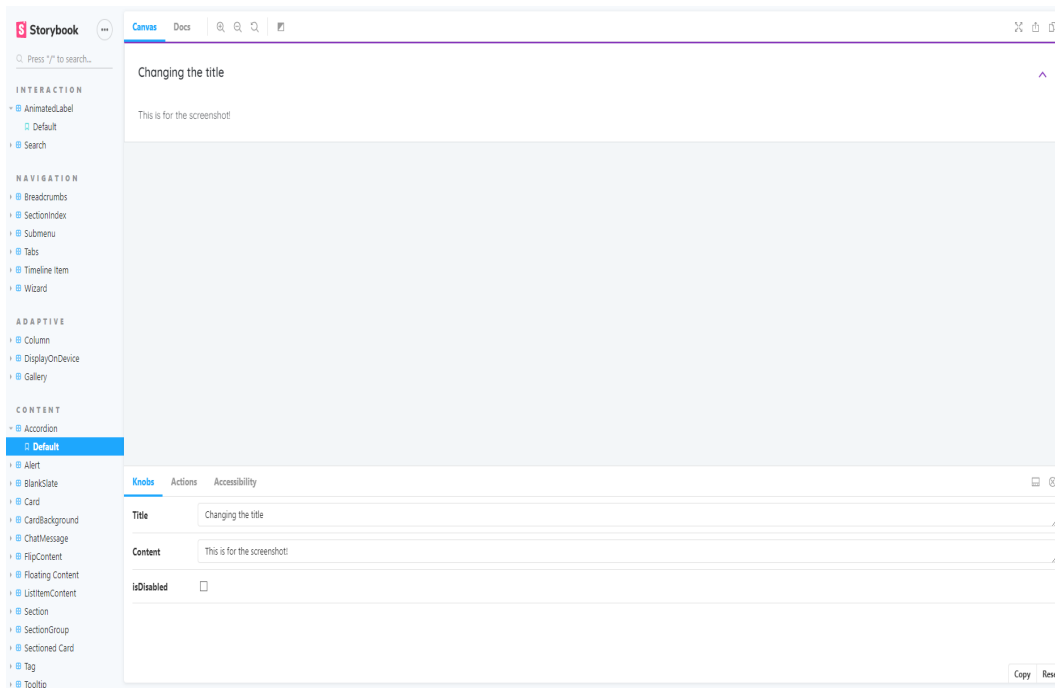


Figura 61: Screenshot of the canvas of the Accordion Component

#### 4.4 CREATING COMPONENTS

Building the components was based on the Atomic Design methodology approach. Inspired by chemistry, the work was divided into small pieces. Starting with basic components (atoms) such as buttons, inputs, links, etc. Then to more complex components (molecules and organisms) such as button group, header, etc. Documenting the component was work done at the same time as building it, this was the advantage of using the Storybook that we were able to have a live preview of the component, test it and add documentation to it. In the next figures, we can see different examples of UI Patterns built.

The figure 62 is a screenshot of the button component which is a basic component representing the Atom example from our methodology.



## Button

**Buttons** are best used to communicate actions that users can take - to commit a change or complete steps in a task. An example of their usage is confirming the deletion of a file in a confirmation dialog.

### Primary

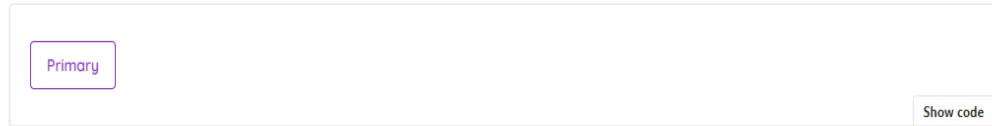


Figura 62: Screenshot of the Button Component

The figure 63 is a screenshot of the button group component which is a more complex component representing the Molecule example from our methodology.

## ButtonGroup

Represent a choice of available options that are displayed.

### Preview

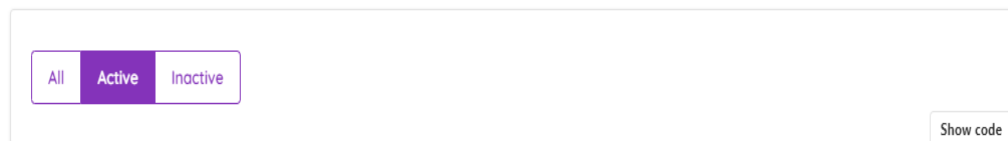


Figura 63: Screenshot of the Button Group Component

The figure 64 is a screenshot of the header component which is a complex component representing the Organism example from our methodology.

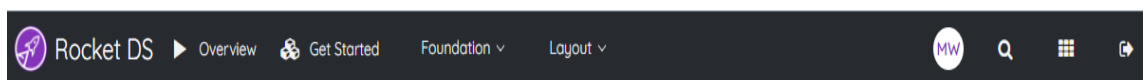


Figura 64: Screenshot of the Header Component

## 4.5 CREATING SCREEN TEMPLATES

In this section, we will present a few figures representing the screen templates. Which are pages where the components functioning together to give them more context to behave together and have a purpose.

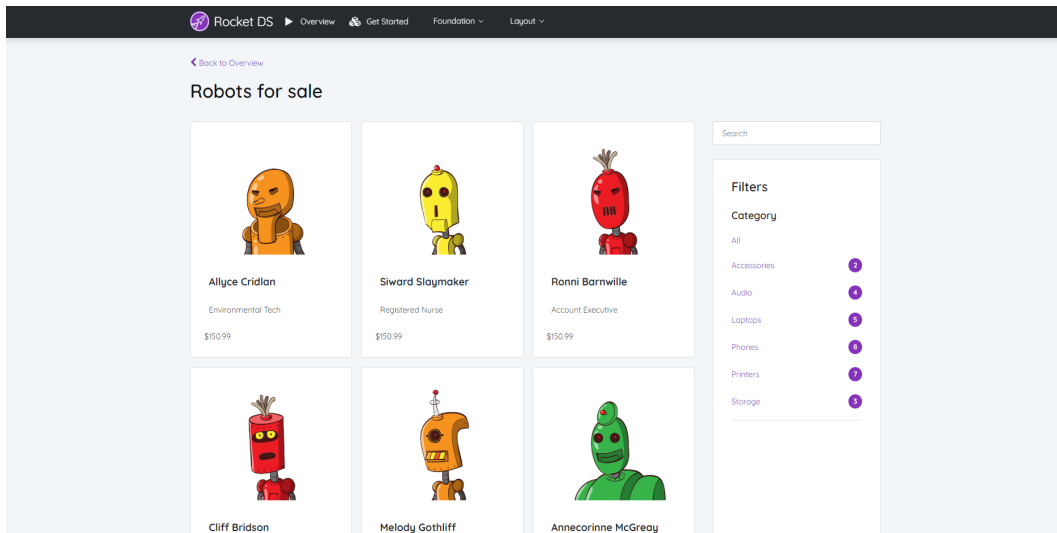


Figura 65: Screenshot of the Three columns page

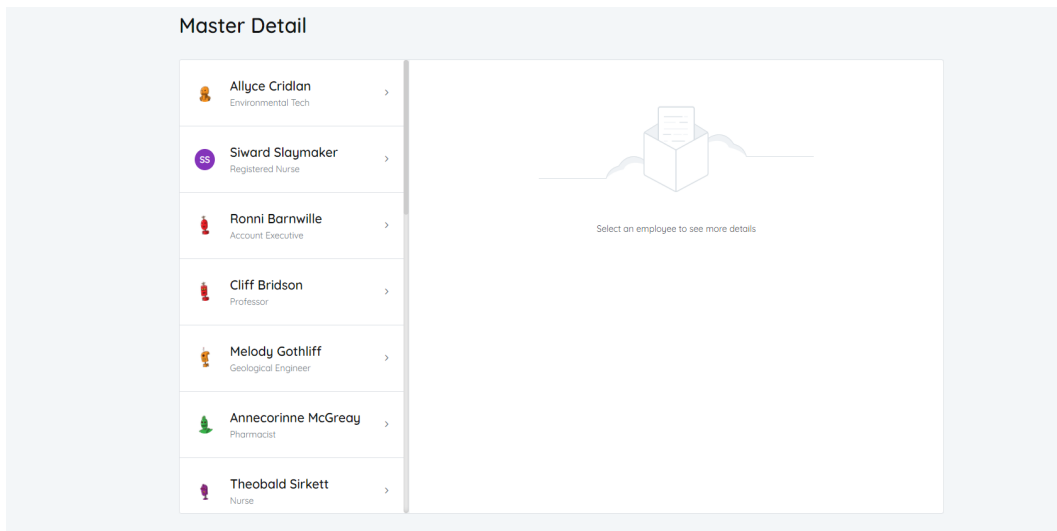


Figura 66: Screenshot of the Master Detail page

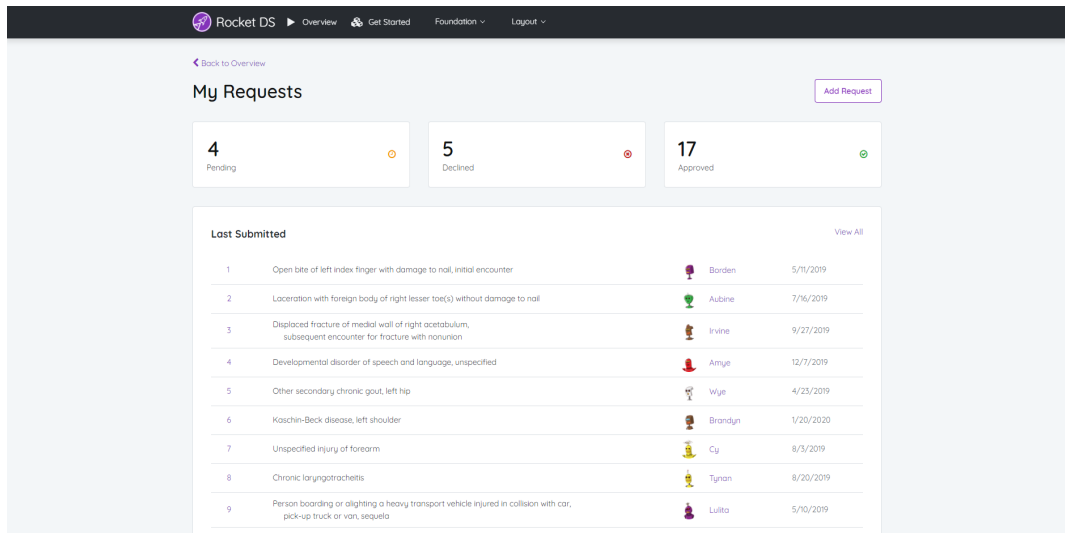


Figura 67: Screenshot of the Admin Dashboard page

COMPARING TO COMPETITORS

In this section, we will compare our system with others available on the market such as Carbon Design System from IBM and Material Design System from Google based on the features that they are offering in the figure 68.

	Rocket Design System	Carbon Design System	Material Design
Available in different frameworks	NO	Web-Angular-React-Vue Vanila	Web-React
OutSystem version	YES	NO	NO
Storybook	YES	YES	NO
Screen Templates	YES	YES	YES
Best Practices in documentation	NO	YES	NO

Figura 68: Comparing Rocket Desing System with competitors

## VALUE ADDED

Employing Rocket Design System will have a huge impact from an enterprise perspective. We can identify the added value of the system to the company in a few points:

- Ability to deliver the best quality of code and best practices.
- Upgrading will be easier since the product is based on modularity.
- Adding a new product to the portfolio will have a good impact on the interest of the client.
- The research phase was beneficial for all the teams regarding their hard and soft skills.
- Training will be easier, faster, and made by the current teams and will not cost the company money and time for recruits.
- Making the development process faster and become more competitive with short time deliveries.



## CONCLUSION AND FUTURE WORK

---

This research aimed to identify the importance of a design system in an organization. Based on several types of research, it can be concluded that design systems have a huge impact on the organization and not only that, their teams, the users, and the business industry as well.

Inspired by other areas like chemistry we were able to go through the journey of creating our design system from scratch. Basing on other experiences and researches, we could define various rules and guidelines to follow so we can learn from the past to plan a better future. Improvement is something the human being is facing since we came to life, especially in the technology area, inventions, ideas, and new technologies are being born every day which will make us question our- selves. And now what's next? As future work, we can think about two majors things for our design system. How to maintain it and keep it updated and stable? How can we improve it?

After going through this journey with researches, I was able to understand more about the other teams that my organization is made of. As a developer, I had the opportunity to dive into the designer world. Understanding their struggles and how their journey looks like will help me to become more flexible and more understanding. After understanding the part of the designer in the process, the communication will be way easier and this will have a positive reflection on the work.

I had also the opportunity to improve my skills in web development, the area that always fascinated me and the one that I'm starting my career in. I had the chance to master innovative tools and technologies especially the ReactJS framework and Storybook. And my knowledge in programming languages has improved too, especially in JavaScript and SASS, I was able to have more challenges that allowed me to have more advanced skills using these languages.

Like any product, Informatic systems and software need to be updated. Improving your product is becoming a mandatory task to add to our to-do lists. As a future work for Rocket Design System we can talk about two future updates to work on which are:

- **Creating version in other frameworks** Having the system available on few frameworks will attract more clients and improve the company's reputation because this is proof that the teams can make any challenge and can master different frameworks and technologies to deliver the same product and give more option to future users depending on their needs.
- **Adding Best Practices documentation** Improving documentation is very important and adding best practices will prove that the teams are making a huge effort to deliver a great user experience for their clients. This will reduce the maintenance and the assistance processes later on with the clients since they can have all the answers to their questions from the documentation.

## BIBLIOGRAFIA

---

- [1] Diana MacDonald, *Practical UI Patterns for Design System Fast-Track Interaction Design for a Seamless User Experience*. Victoria, VIC, Australia, 2019.
- [2] Sarrah Vesselov, Taurie Davis, *Building Design Systems*. Dade City, FL, USA, 2019.
- [3] Brad Frost, *Atomic Design*, Pittsburgh, Pennsylvania, USA, 2016.
- [4] Marco Suarez, Jina Anne, Katie Saylor-Mille, Diana Mounte, Roy Stanfield, *Design Systems Handbook*, New York, NY, USA, 2017.
- [5] Marcin Treder, CEO and Head of Design Operations at UXPin, *The Actionable Guide to Starting Your Design System The 100-Point Process Checklist*, New York, NY, USA, 2017.
- [6] Marcin Treder, Jerry Cao, Vince Ho, *Enterprise UX Industry Report 2017 to 2018*, New York, NY, USA, 2017.
- [7] Ben Goldman, *DSM Field Guide*, New York, NY, USA, 2018.
- [8] Elisa Pyrohonen, *Hack the design system. Revolutionize the way your organization scales design*, New York, NY, USA, 2019.
- [9] Robert J. Moore, Eric Young Liu, Saurabh Mishra, Guang-Jie Ren *Design Systems for Conversational UX*, San Jose, CA, USA.
- [10] Churchill Elizabeth, *Scaling UX with Design Systems*, ACM, NY, USA, 2019.
- [11] Moore, Robert J. and Raphael Arar, *Conversational UX Design: A Practitioner's Guide to the Natural Conversation Framework*, ACM, New York, USA, 2019.
- [12] Alla Kholmatova, *Design Systems A practical guide to creating design languages for digital products*, Freiburg, Germany, 2017.
- [13] Donella H. Meadows, *Thinking in Systems*, London, UK, 2009.
- [14] Andrew Couldwell, *Laying the Foundations*, New York, USA, 2019
- [15] Christopher Alexander, Sara Ishikawa, Murray Silverstein, *A Pattern Language*, New York, Oxford University Press 1977.



- [16] Micah Godbolt, *Frontend Architecture for Design Systems*, Newton, Massachusetts, USA, 2016.
- [17] Leah Buley, *The User Experience Team of One: A Research and Design Survival Guide*, NY, USA, 2013.
- [18] Marcin Treder, CEO and Head of Design Operations at UXPin, *The Actionable Guide to Starting Your Design System The 100-Point Process Checklist*, New York, NY, USA, 2017.
- [19] Adam Silver, *Form Design Systems*, Freiburg, Germany, 2018.
- [20] Jeff Gothelf, *Lean UX: Designing Great Products With Agile Teams*, Newton, Massachusetts, USA, 2016.
- [21] Liquid Alumni, *Save Time and Create a Better User Experience by Using Design Systems*, 2016. [Online]. Available: <https://www.liquidint.com/blog/design/save-time-and-create-a-better-user-experience-by-using-design-systems>.
- [22] Cristiano Rastelli, *Measuring the Impact of a Design System*, 2019. [Online]. Available: <https://didoo.medium.com/measuring-the-impact-of-a-design-system-7f925af090f7>.
- [23] Ryan Lum, *How to: measure a Design System's impact* , 2019. [Online]. Available: <https://uxdesign.cc/how-to-measure-design-system-impact-guide-f1f9f0c3704f>.
- [24] Bootstrap, *Beagle – Responsive Admin Template* [Online]. Available: <https://themes.getbootstrap.com/product/beagle-responsive-admin-template>.
- [25] Eightshapes, *Contrast Grid*, 2017. [Online]. Available: <https://contrast-grid.eightshapes.com>.

## DECLARAÇÃO

---

Declaro, sob compromisso de honra, que o trabalho apresentado nesta dissertação, com o título “*Fundamentals of Design Systems*”, é original e foi realizado por Marouen Abdi (2182710) sob orientação de Professor Doutor Silvio Priem Mendes ([smendes@ipleiria.pt](mailto:smendes@ipleiria.pt)).

*Leiria, Novembro de 2020*

---

Marouen Abdi