Alma Mater Studiorum
Università di Bologna
Campus di Cesena

DIPARTIMENTO DI INFORMATICA – SCIENZA E INGEGNERIA
Corso di Laurea Magistrale in Ingegneria e Scienze Informatiche

Web Component per un Design System: un caso di studio aziendale

Elaborato in:
APPLICAZIONI E SERVIZI WEB

Relatore:
Prof.ssa
Silvia Mirri

Presentata da:
Andrea Pruccoli

Sessione II
Anno Accademico 2021-2022

Ai miei familiari.

Ai miei amici.

Alla mia morosa.

Introduzione

I contenuti e i servizi offerti dal Web hanno subito negli anni una costante e continua evoluzione, dovuti soprattutto alla maggiore disponibilità di dispositivi in grado di navigarlo, partendo dai desktop fino ad arrivare agli smartwatch.

Oggi i requisiti e le aspettative che gli utenti hanno nei confronti delle applicazioni Web sono sempre maggiori, desiderano un accesso ai contenuti sempre più rapido, interfacce semplici e facili da usare, oltre che reattive, e che tali contenuti siano accessibili da una vasta gamma di dispositivi che presentino sempre più funzionalità.

Le aziende devono essere sempre pronte a rispondere a queste esigenze e a fornire agli utenti finali che usano i loro prodotti la miglior esperienza possibile, per questo dovrebbero rimanere aggiornati sulle tecnologie disponibili per la creazione di applicazioni Web che possano essere usufruite su più dispositivi.

Questo è ancora più importante quando un'azienda possiede più prodotti, sviluppati da team diversi che usano tecnologie differenti tra loro. Per alcune aziende è importante che i propri prodotti, sebbene trattino tematiche differenti, si presentino con interfacce che rimandino al proprio marchio, non solamente per via della presenza del loro logo o del loro nome nell'header o nel footer del sito, quanto più nei componenti utilizzati per creare le interfacce.

Succede così che i vari team devono, ognuno, progettare e sviluppare i componenti nella propria tecnologia, in modo che abbiano le stesse funzio-

nalità, stesso stile e stesso comportamento in ogni situazione. Il più delle volte, questo è difficile da realizzare e, nel lungo periodo, anche costoso da mantenere. Riuscire a centralizzare lo sviluppo di questi elementi in un unico punto aiuta l'azienda a mantenere bassi i costi di manutenzione e a rendere omogenea l'esperienza degli utenti tra i vari prodotti.

L'obiettivo del lavoro svolto nell'ambito di questo progetto di tesi è illustrare le potenzialità e l'utilità fornite dall'introduzione di una suite di componenti personalizzati, seguendo lo standard dei Web Component, all'interno dei prodotti forniti da una grande impresa, quale che è il Gruppo Maggioli. L'analisi non vuole studiare la qualità delle interfacce e dell'esperienza utente dal punto di vista di chi usufruisce dei prodotti, quanto più dell'esperienza di chi utilizza tali componenti all'interno dei propri progetti per creare l'interfaccia utente da presentare poi agli utenti finali.

Vengono considerati sia aspetti finalizzati agli obiettivi (aspetti di qualità pragmatica), come apprendibilità, efficienza e controllabilità, sia aspetti non finalizzati agli obiettivi (aspetti di qualità edonica), come la stimolazione e l'originalità, sia l'impressione generale data dall'attrattività, sulla base dei medesimi principi e caratteristiche dell'usabilità delle interfacce-utente noti in letteratura.

La tesi è strutturata come segue:

- Nel primo capitolo si descrive lo stato dell'arte attuale che riguardano lo sviluppo dei Web Component, ripercorrendo l'evoluzione storica che ha portato il Web sino al suo stato attuale e introducendo i concetti chiave che caratterizzano lo sviluppo Web odierno, elencando anche quelle che sono le più moderne e utilizzate librerie o framework disponibili sul mercato.

  Si passa poi all'introduzione del concetto di Web Component, spiegando le motivazioni che hanno portato alla nascita di questa tecnologia e dei benefici che può portare.

Il capitolo prosegue con la descrizione delle tecnologie che compongono lo standard, fornendo quando possibile anche esempi di codice, concludendo con un elenco delle librerie e framework attualmente disponibili per lo sviluppo dei Web Component e del supporto dello standard da loro offerto.

- Nel secondo capitolo si descrive il contesto aziendale del Gruppo Maggioli, facendo un piccolo excursus storico dell'azienda santarcangiolese per descrivere poi brevemente i principali prodotti che vengono interessati dal lavoro presentato, per concludere con le motivazioni aziendali che hanno portato all'avvio e allo sviluppo del progetto di tesi.

- Nel terzo capitolo si descrivono le azioni e le scelte prese dall'azienda per realizzare una suite di Web Component all'interno del già esistente progetto del *Design System*. Verrà da prima descritto il processo di individuazione dei componenti da generalizzare e successivamente della tecnologia scelta per soddisfare le necessità aziendali, descrivendo aspetti come l'organizzazione e il versionamento del codice, modalità di testing e visualizzazione dei componenti sviluppati in apposito ambiente, con particolare focus sulla modalità di pubblicazione e rilascio dei componenti e sul modello di contribuzione adottato.

- Nel quarto capitolo viene presentata un'analisi qualitativa del progetto sulla base dei dati raccolti da un questionario somministrato al personale Maggioli che ha per primo integrato la suite di componenti all'interno dei propri progetti, accompagnata dalla descrizione della metodologia utilizzata per creare ed analizzare il questionario.

  Il capitolo termina con una breve retrospettiva su quello che è stato fin'ora lo sviluppo dei Web Component all'interno del *Design System* aziendale, riportando le principali problematiche riscontrate durante il percorso e come queste siano state risolte.

Indice

Elenco delle figure

List of Listings

Capitolo 1

Web Components, stato dell'arte

In questo primo capitolo saranno presentati i principali concetti della tecnologia su cui si basa il lavoro di tesi, i Web Components, dopo aver riportato un excursus storico sullo sviluppo del Web e delle tecnologie attualmente utilizzate per sviluppare la parte client delle applicazioni Web.

1.1 Introduzione

Il Web, conosciuto anche come World Wide Web o WWW, fu proposto il 12 Marzo 1989 e reso pubblico il 6 Agosto 1991 da Tim Barners-Lee, informatico inglese presso il CERN di Ginevra [5].

L'idea iniziale di Tim Barners-Lee era quella di trasformare Internet in uno strumento di pubblicazione per la condivisione e la disseminazione di dati scientifici e, più genericamente, d'informazioni, che col tempo è divenuto essenziale ed indispensabile per moltissime persone ed organizzazioni. La sua evoluzione, nel corso degli anni, ha riunito diverse discipline come media, scienza dell'informazione e tecnologie di informazione e comunicazione, facilitando in questo modo la creazione, manutenzione, condivisione e uso di diverse tipologie di informazioni da ogni sorgente, in ogni momento, usando

e sfruttando una vasta varietà di dispositivi come computer desktop, computer portatili, computer palmari e telefonini -fino ad arrivare agli odierni smartphone/tablet [6].

Nel corso della sua evoluzione, il Web è stato classificato con le seguenti etichette [5]:

Web 1.0 definito anche come `read-only Web` (o statico) dallo stesso Tim Barners-Lee, se ne fa risalire l'implementazione nel 1989 e durò fino al 2005. Statico poiché chi usufruiva del servizio era in grado di condividere le proprie informazioni e documenti con altri utenti, ma a parte cliccare sui link presenti in pagina l'interazione con il sito Web era nulla. A quel tempo il World Wide Web era solamente uno spazio in cui gli elementi di interesse, definiti come risorse, erano identitifcati globalmente tramite Uniform Resource Identifier -URL.

Web 2.0 definito da Dale Dougherty nel 2004 come un `read-write Web`, è la seconda generazione del Web. Tramite l'introduzione di nuove e più evolute interfacce utente e di strutture integrate, gli utenti non solo sono in grado di usufruire passivamente dei contenuti presenti nei vari siti Web (che stanno esponenzialmente crescendo), ma sono anche in grado di generare e di modificare il contenuto presente, arricchendo e portando il loro contributo nella base di contenuti che è il Web. Nascono così nuovi siti che permettono agli utenti di condividere contenuti e collaborare in nuovi modi, alcuni esempi sono i social network come MySpace, siti di condivisione di contenuti multimediali come YouTube e siti di divulgazione informativa collaborativa come Wikipedia.

Con l'introduzione di nuove tecnologie come AJAX (Asynchronous JavaScript and XML), Ruby, blog wiki e tanto altro, il Web, dando la possibilità agli utenti di diventare autori del materiale presente sulla rete e non solo semplici spettatori, diviene velocemente un luogo dinamico e altamente interattivo. Un'altra caratteristica molto importante di questa generazione è la proliferazione di siti Web che fanno uso di

API (Application Programming Interfaces), fornendo un metodo più semplice e rapido agli sviluppatori per recuperare informazioni attraverso i server Web e poter generare a loro volta applicazioni basate sui dati così recuperati.

Web 3.0 conosciuto anche come `semantic Web`[1] o `executable Web`, ha come idea di base la definizione di dati strutturati collegati tra di loro, in modo da rendere più efficace la ricerca, automazione, integrazione e riuso attraverso diverse applicazioni. Il `Web 3.0` è stato pensato come un Web dove il concetto di sito/pagina Web sfuma, dove i dati non sono posseduti ma piuttosto condivisi, trasformando quello che è il Web adesso da un "semplice" contenitore di documenti in un `database`, fornendo in questo modo l'accesso ai contenuti da parte di diverse applicazioni e non solo dai browser.

Altra particolarità del **Web 3.0** è quello di rendere i dati presenti su Internet non solo comprensibili dall'uomo, ma anche comprensibili dalle macchine tramite l'utilizzo di tecnologie come **RDF** (Resource Description Framework) e **OWL** (Web Ontology Language).

1.2 Le applicazioni Web

Lo sviluppo di Web front-end è un'area lavorativa con un passo di crescita elevato e rapido, molte sono le librerie e i framework che ogni anno vengono constantemente creati e mantenuti, e ad oggi JavaScript è il linguaggio di programmazione per la creazione di applicazioni Web più usato dagli sviluppatori. Framework e librerie diversi hanno scopi e finalità diversi, alcuni sono usati per dare uno stile strutturato al proprio progetto in maniera più efficente, altri sono utilizzati per aiutare gli sviluppatori a strutturare il modo in cui scrivono codice e suddividono logicamente i file [7].

[1] `https://en.wikipedia.org/wiki/Semantic_Web`

Per progetti piccoli o di piccola-media grandezza potrebbe non essere necessario dover ricorrere all'uso di framework o librerie, ma si rivelano essere molto utili quando sono presenti diversi requisiti utente e un'architettura complessa già pianificata.

Di seguito alcuni concetti principali riguardanti lo sviluppo di applicazioni Web tramite JavaScript:

Architettura a componenti uno dei concetti più importanti degli ultimi anni, l'uso di componenti fornisce una maniera per poter organizzare un'applicazione, ogni framework e libreria gestisce in maniera diversa la logica, l'HTML e la gestione dello stile dei propri componenti.

Document Object Model (DOM) è la rappresentazione dei dati degli oggetti che compongono la struttura e il contenuto di un documento nel Web. Rappresenta il documento come nodi e oggetti in modo tale che i linguaggi di programmazione possano connettersi ed effettuare operazioni sulla pagina, fungendo come un'interfaccia, fornendo la capacità ai programmi di modificare la struttura, lo stile e il contenuto dei documenti.

Single Page Application (SPA) sono applicazioni Web dove solo alcune parti della pagina vengono renderizzate a seguito della ricezione dell'input da parte dell'utente, e non l'intera pagina (previa nuova richiesta HTTP al server), ciò comporta transizioni più rapide ed un'esperienza utente più piacevole.

Gestione dello stato un'applicazione che permette agli utenti di mostrare, aggiungere e rimuovere contenuti necessita di tenere traccia e aggiornare i dati sottostanti, conosciuti come `stato`. Lo stato può ad esempio essere un concetto locale a livello di singolo componente oppure globale in modo da essere condiviso da più componenti e pagine.

Reattività ogni volta che lo stato o i dati vengono modificati e aggiornati, l'interfaccia grafica necessita di aggiornarsi di conseguenza per poter

rappresentare graficamente i nuovi dati. La reattività dell'interfaccia grafica è gestita in modo diverso da ogni framework.

Riusabilità la riusabilità del codice porta diversi vantaggi: permette di scrivere meno codice, ne migliora la leggibilità, migliora la strutturazione del codice e riduce il costo della definizione e manutenzione dei test. La riusabilità del codice si può perseguire estraendo la definizione di classi e metodi quando se ne presenta la possibilità, risultando in minori errori, inefficienza e migliore manutenibilità del codice, in quanto future modifiche dovranno essere apportate in un unico posto per poi vedersi riversate ovunque quelle classi o metodi vengano usati.

Instradamento ci sono diversi modi per poter navigare un'applicazione Web. L'instradamento lato server permette al browser di comunicare con un server e recuperare il nuovo contenuto da mostrare, risultando nella modifica dell'URL nella barra degli indirizzi, salvando contestualmente la storia della navigazione per poter fornire le funzionalità di indietro e avanti.

Le SPA, tuttavia, caricano un unico file HTML e aggiornano continuamente il DOM senza la necessità di dover navigare un URL diverso. In questo caso si parla di instradamento lato client, e le SPA sono in grado di gestire molte viste senza mai dover modificare l'indirizzo Web (anche se viene comunemente fatto per poter sfruttare maggiori funzionalità).

Sebbene si possa scrivere codice JavaScript senza l'aiuto di alcun framework, molti sono i framework che vengono utilizzati dai sviluppatori di tutto il mondo come strumenti per aiutarli nella gestione della struttura del codice, delle prestazioni e della riusabilità. Tra quelli presenti in letteratura, i framework più utilizzati per lo sviluppo di applicazioni Web front-end, secondo un recente questionario della piattaforma *State of Js* compilato da più di 16000 sviluppatori di diverse nazioni [8], adottano principi delle single page application, e sono (in ordine di uso secondo il questionario):

React la prima versione fu rilasciata nel 2013 ed è di proprietà di Meta (precedentemente Facebook), si tratta di una libreria open-source mantenuta sia dal team di sviluppo di Meta che dalle aziende e gli sviluppatori che ne formano la community. Nel 2021 è stato il framework più usato per la costruzione di interfacce utente ed il terzo framework che più ha soddisfatto gli sviluppatori che lo hanno utilizzato.

Nella libreria sono inclusi la gestione dello stato e la sua renderizzazione nel DOM, mentre manca nel pacchetto core la gestione dell'instradamento. Feature principale di React è la composizione di componenti e il fatto che componenti scritti da sviluppatori diversi dovrebbero essere in grado di lavorare bene insieme. Altri principi di design promossi da React sono la common abstraction (ovvero limitare l'inserimento di feature a meno che non riguardino la maggior parte degli elementi del pacchetti core per tenere basso il peso della libreria), la stabilità, l'interoperabilità e una buona esperienza di sviluppo.

Angular iniziato come progetto secondario di un dipende di Google, nasce come AngularJS nel 2010 per poi venir riscritto completamente in TypeScript e cambiare nome semplicemente in Angular verso il 2015, è il secondo framework più utilizzato ma solamente il nono framework più apprezzato nel 2021, vedendo quindi continuare la sua discesa nell'apprezzamento da parte degli sviluppatori.

Principio cardine del framework è quello di applicare il principio di singola responsabilità per tutti i componenti e i servizi all'interno di un progetto. Propone una propria style guide da seguire con la quale poter applicare il più possibile i principi di design da questa proposti.

Vue.js libreria open-source creata per essere facilmente adottabile all'interno di ogni progetto e capace di scalare tra l'essere una semplice libreria ed un vero e proprio framework, a seconda delle necessità e dei casi d'uso. Rilasciato nel 2013, è ad oggi mantenuto dal suo creatore e dai

membri del core team. Secondo il questionario è il terzo framework più utilizzato e il quarto per gradimento da parte della community.

Svelte il più recente dei framework qui descritti, viene rilasciato inizialmente nel 2016 scritto in JavaScript per essere poi riscritto completamente in TypeScript nel 2019. Particolarità di Svelte è la conversione del codice di un'applicazione in JavaScript durante la fase di compilazione anziché interpretarne il codice in fase di esecuzione. Nel questionario di riferimento del 2021, si piazza al quarto posto come utilizzo e secondo per gradimento, andando a scavalcare React, Angular e Vue.

Di seguito, la figura 1.1 e figura 1.2 mostrano le classifiche dei più famosi ed utilizzati framework (tra cui i 4 sopra citati) ordinati sia per utilizzo che per gradimento da parte degli sviluppatori (fonte: questionario online somministrato da State Of Js[2]).

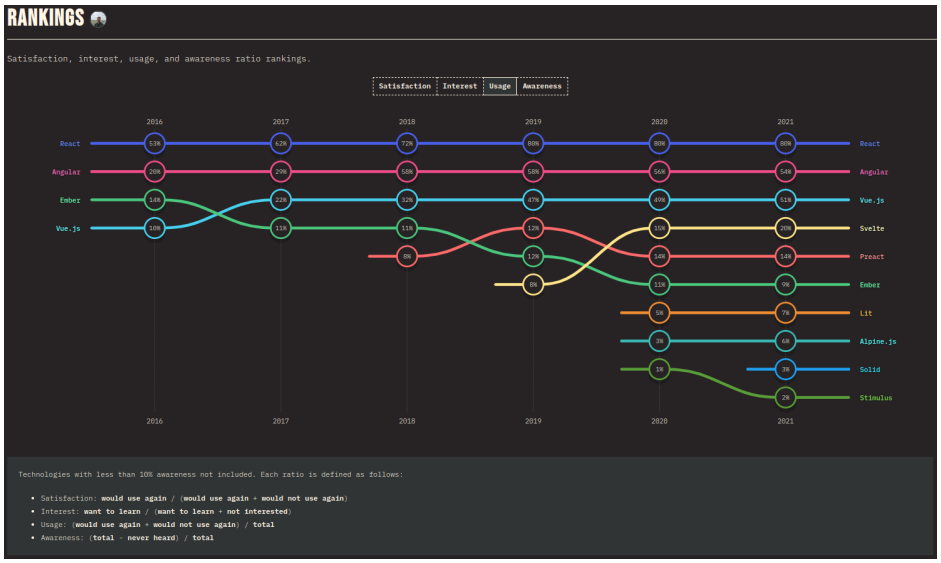


Figura 1.1: Classifica dei framework secondo la scala di utilizzo da parte degli sviluppatori [8]

[2] https://2021.stateofjs.com/en-US/libraries/front-end-frameworks

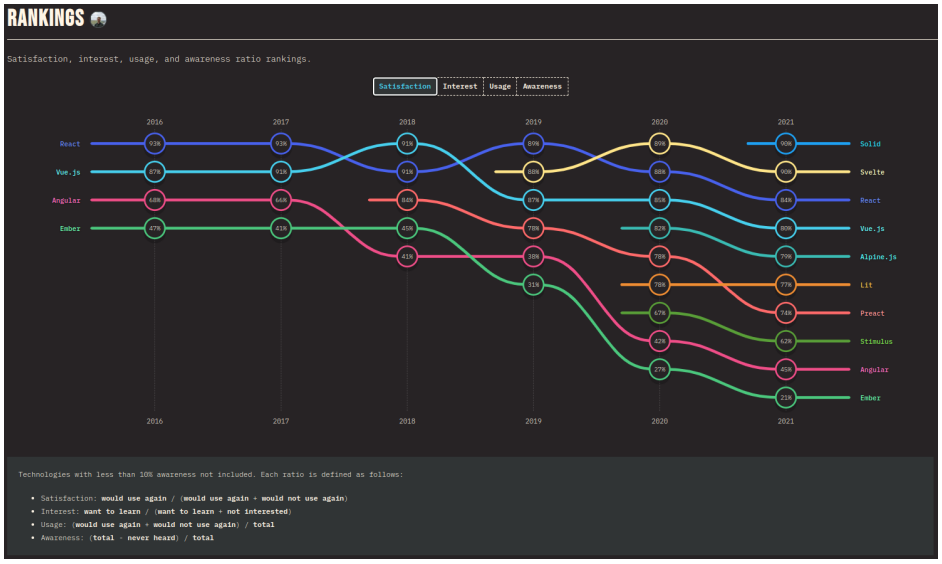


Figura 1.2: Classifica dei framework secondo la scala di gradimento da parte degli sviluppatori [8]

1.3 I Web Component

Con il termine *"Web Component"* non ci riferiamo ad una singola tecnologia o ad una singola libreria, il termine viene utilizzato per fare riferimento ad una collezione di tecnologie che permettono agli sviluppatori di descrivere efficacemente le implementazioni di elementi HTML che già esistono [9].

Provando a vederlo da un altro punto di vista, come possiamo implementare l'elemento `<p>` come combinazione di altri elementi HTML esistenti? Plausibilmente quel tag non è altro che una definizione di determinate regole CSS, quindi si potrebbe implementare utilizzando un tag `<div>` a cui viene applicato qualche stile inline, oppure definire una classe `p` nella quale vengono definite le regole per rappresentare un paragrafo.

E l'elemento `<select>` con i relativi tag `<option>` che ne costituiscono la lista di elementi? L'implementazione in questo caso inizia a complicarsi, con la probabilità di diventare un intruglio disordinato di stili, JavaScript e

HTML che rischiano di scontrarsi con stili, JavaScript ed altri elementi già presenti in pagina. L'implementazione risulterebbe essere ancora fattibile, ma con grandi rischi e problematiche se non ben organizzato.

Infine, come poter implementare un tag `<video>`? Inizia a diventare sempre più complicato e rischioso, e questi sono le tipologie di problematiche che i Web Component cercano di risolvere, fornendo agli sviluppatori un metodo comune per poter incapsulare, proteggere e impacchettare questi concetti.

1.3.1 Perché i Web Component?

Il Web è stato, ed è tutt'ora, in uno stato di transizione. Inizialmente progettato per visualizzare documenti, si è lentamente mutato in una piattaforma applicativa, trasformando radicalmente il panorama dello sviluppo software. Mai fu più semplice rilasciare una nuova versione di un'applicazione: uno sviluppatore effettua il push di modifiche al codice o di nuove risorse ad un server remoto, tramite azioni manuali o automatizzate queste modifiche vengono rilasciate e l'utente finale tramite il ricaricamento della pagina si ritrova ad utilizzare la nuova versione dell'applicazione Web.

Sfortunatamente i browser non hanno mantenuto lo stesso passo, questo ha forzato gli sviluppatori a trovare nuove ingegnose soluzioni per colmare il divario.

Negli ultimi anni sono nate librerie e framework JavaScript (come quelle descritte nel capitolo 1.2), progettate con l'intento di fornire una struttura solida e standard alle applicazioni Web. Queste librerie non seguono rigorosamente un approccio MVC ma approcci che discendono da questo, come MVVM e MVP -prendendo il nome generico di architetture MV*.

Caricatori di moduli come Require.js[3] hanno aiutato molto a consolidare tale struttura solida fornendo quello che è il concetto di `import` comune in

[3] `https://requirejs.org/`

altre piattaforme (come Java). La presenza di questi strumenti ha permesso agli sviluppatori di pensare più modularmente. La nascita e crescita di jQuery[4] e di librerie di widget UI ha aiutato la normalizzazione dello sviluppo Web, scrivere codice JavaScript che permettesse di interagire con il DOM attraverso la flora di browser disponibili era un incubo, poiché gli unici elementi UI nativi supportati dai browser erano delle semplici form.

Nonostante l'arrivo di HTML5 e di nuove API, il Web rimane ancora abbastanza limitato rispetto alle altre piattaforme applicative. Per esempio, il Web non è estensibile, nel senso che non si possono creare nuove tipologie di elemento o estendere quelli già esistenti, e non esistono import o metodi per incapsulare i componenti.

Tutto ciò però cambia tra il 2010 e il 2012, quando in Google iniziano i lavori per poter redigere uno standard per l'ampliamento del DOM, quello che oggi conosciamo come "Web Component". Le specifiche riguardano un insieme di API per la piattaforma Web che hanno l'obiettivo di permettere agli sviluppatori di estendere i tag HTML presenti nei browser e rendere i componenti così creati utilizzabili su ogni browser di ogni dispositivo, con qualsiasi libreria o framework JavaScript che utilizza HTML [10].

Nonostante se ne faccia risalire la nascita presso gli uffici di Google, bisogna menzionare che il concetto di estensione del DOM era nato già nel 1998 presso Microsoft sotto il nome di `HTML Components`, progetto abbandonato nel 2011 per l'elevata complicazione dell'implementazione e per l'utilizzo limitato al solo browser Internet Explorer.

Un altro tentativo di estensione del Web è avvenuto nel 2001 da parte di Mozilla, che però vide la stessa fine del progetto di Microsoft nel 2012, abbandonato causa eccessiva complicazione implementativa e uso limitato ai soli prodotti della compagnia [11].

Spesso il termine "Web Component" viene erroneamente utilizzato per riferirsi ad "HTML5", ma i due termini hanno significati diversi per persone

[4] `https://jquery.com/`

diverse. HTML5 fa riferimento ad una nuova tipologia di documento con nuovi elementi, CSS3, e le nuove API JavaScript, che tutte insieme ridefiniscono lo sviluppo Web. Lo stesso vale per i Web Component, il termine è utilizzato per riferirsi ad una nuova collezione di funzionalità che, quando utilizzate insieme, permettono agli sviluppatori di creare componenti riutilizzabili.

Immaginiamo un mondo in cui la piattaforma Web sia nativamente espandibile, dove ogni elemento possa essere esteso, e nuovi elementi possano essere definiti per creare con facilità ricche interfacce utenti e dove possano essere facilmente importabili tramite una metodologia standard, includendo tutte le risorse come JavaScript, CSS, e immagini.

Inoltre immaginiamo che questo sistema sia stato ottimizzato per deduplicare le richieste di uno stesso import e che blocchi di markup possano essere caricati e marcati come inerti in modo da non avere impatto sulle performance. Oppure un'intera e reale applicazione. Queste sono le promesse dei Web Component.

Immaginiamo di poter creare un componente `dialog` semplicemente importando la risorsa che la definisce e tramite l'utilizzo di appropriato markup:

```
1 <head>
2   <link rel="import" href="/imports/dialog/index.html" />
3 </head>
4 <body>
5   <dialog-component title="Yoda">
6     Do or do not. <br />
7     There is no try.
8   </dialog-component>
9 </body>
```

Listing 1: Import funzionalità di dialog come Web Component direttamente da file

Sarebbe un bel miglioramento in termini di leggibilità rispetto a:

```
<head>
  <link
    rel="stylesheet"
    href="//code.jquery.com/ui/1.11.0/themes/smoothness/jquery-ui.css"
  />
  <script src="//code.jquery.com/jquery-1.10.2.js"></script>
  <script src="//code.jquery.com/ui/1.11.0/jquery-ui.js"></script>
  <script>
    $(function () {
        $("#dialog").dialog();
    })
  </script>
</head>
<body>
  <div id="dialog" title="Yoda">
    Do or do not. <br />
    There is no try.
  </div>
</body>
```

Listing 2: Import funzionalità di dialog con riferimenti ai file jQuery

Con ciò non si vuole denigrare l'utilità che librerie di widget come quella nel listato 2 siano inutili o non abbiano nulla da offrire, poiché ad oggi ancora non tutti i browser supportano completamente i Web Component[5]. Inoltre i Web Component da per loro non sono un rimedio universale per la creazione di ricche interfacce grafiche, il codice del componente `dialog` nel listato 1 rimane ancora lo stesso di quello nel listato 2, ma nel primo caso il componente viene confezionato in un'interfaccia semplice, nativa e standardizzata con un metodo per includere le proprie dipendenze.

Quanto detto potrebbe non sembrare un gran miglioramento, ma l'abilità di estendere, astrarre, importare ed incapsulare in maniera nativa rende lo sviluppo di applicazioni Web molto più semplice. Permette agli sviluppatori di creare elementi riusabili con cicli di vita del componente standard

[5] `https://caniuse.com/?search=web%20components`

ed importabili all'interno di ogni applicazione Web, indipendentemente dalla libreria o framework utilizzata per svilupparla. Questa uniformità crea un contratto implicito tra gli elementi e l'applicazione, rendendo la creazione e la gestione di interfacce un processo più regolare.

Benefici

I benefici dati dall'uso dei Web Component possono essere individuati nei seguenti punti [12]:

- *Natura nativa*: essendo uno standard HTML e supportato dai browser moderni, per poter sviluppare i Web Component non è necessario dover fare uso di librerie di terze parti e quindi per utilizzarle non è necessario includere altre librerie al progetto. Per esempio, un componente scritto in React può essere utilizzato solo all'interno di un ambiente sviluppato in React.

- *Facili da usare e da condividere*: la natura nativa ne permette facilmente il riuso e la condivisione in diversi progetti, con diversi ecosistemi e stack tecnologici utilizzati.

- *Semplicità d'uso*: molto utili in casi specifici, come ad esempio quando il progetto richiede un alto livello di sicurezza dove l'uso di librerie di terze parti è fortemente limitato e viene favorito l'uso di una tecnologia nativa, oltre ad assicurarsi il controllo completo del contenuto delle librerie importate. Librerie di grandi dimensioni di oggi, come possono essere React e Angular, possiedono una codebase abbastanza ampia e questo rende difficile il lavoro di controllo sui bug e le vulnerabilità che possono emergere di versione in versione.

- *Soluzione senza dipendenze complesse*: l'uso di specifiche di HTML da un vantaggio sui framework/librerie più popolari poiché è possibile utilizzare uno specifico Web Component senza dover importare all'interno

del progetto le dipendenze del framework. Un widget creato usando Angular potrà essere utilizzato solo in un progetto che fa uso di Angular, mentre il widget creato coi Web Component può essere direttamente inserito ed utilizzato senza nessun'altra dipendenza.

- *Codice leggibile*: altro grande vantaggio dell'uso dei Web Component è che se noi definiamo un componente col nome di `<my-web-component>` e lo utilizziamo all'interno della costruzione di una pagina Web, analizzando il codice della pagina una volta arrivati sul browser quel tag sarà presente tra i nodi del DOM, ciò semplifica le operazioni di debug. Altre librerie al contrario potrebbero non riflettere sul DOM il nome dei componenti che vengono definiti in fase di sviluppo, quindi il componente `<Header/>` utilizzato per la definizione di un header potrebbe essere tradotto in codice nel DOM come una serie di tag `<div>`, complicando potenzialmente le operazioni di debug sopratutto da parte di chi quell'interfaccia non l'ha creata.

Potenziali problematiche

D'altro canto, i Web Component non sono un'implementazione perfetta, esistono quindi alcune problematiche che i programmatori devono affrontare in determinati casi di sviluppo:

- *Mancanza di informazione*: essendo una tecnologia piuttosto nuova, almeno per quanto riguarda la sua adozione da parte della comunità, le informazioni sono abbastanza carenti per quanto riguarda problematiche precise. Lo sviluppatore deve impiegare diverso tempo per comprendere perché una determinata funzionalità non funziona, con bassa probabilità di trovare risposte online su forum famosi come StackOverflow.

- *Velocità di inizializzazione*: poiché i Web Component vengono registrati tramite JavaScript, il processo di caricamento, processazione,

registrazione e renderizzazione potrebbe impiegarci diversi millisecondi, questo potrebbe causare la visualizzazione del componente senza stile - il fenomeno viene definito come `flash of unstyled content` o `FOUC`. Per questo motivo l'uso dei Web Component non è consigliato per applicazioni progettate per essere visualizzate una sola volta, dove la velocità di caricamento e l'abilità di catturare l'attenzione dell'utente sono fondamentali. D'altra parte, per tutte le altre applicazioni dove si fa uso della cache del browser e l'utente torna a visualizzare quella pagina i Web Component sono ancora consigliati.

- *SEO*: facendo uso dei Web Component all'interno delle proprie applicazioni, uno dei dubbi è se l'uso di tali componenti possa influire negativamente sul SEO [13]. In una recente intervista [14] in cui gli veniva chiesto se ci fossero problemi relativi a ciò, John Mueller ha risposto che i Web Component, essendo implementati usando diverse forme di JavaScript, e che tali forme vengono processate correttamente durante le ricerche Google, sono complemtamente supportati, seppur concludendo che è sempre meglio controllare che non influiscano negativamente tramite gli strumenti messi a disposizione.

- *Integrazione con librerie di terze parti*: quando si importano librerie esterne che contengono degli stili si potrebbero incontrare dei problemi, in tal caso dovrebbe essere necessario scendere a compromessi per poter utilizzare tali librerie nei propri Web Component.

1.3.2 Composizione

I Web Component si basano principalmente sull'utilizzo di tre tecnologie fondamentali [15]:

- **Custom element**: un set di API JavaScript che permettono di definire elementi personalizzati (tag HTML non presenti di default come `<div>`), il loro comportamento e registrarli nel `CustomElementRegistry`, con la possibilità di utilizzarli all'interno della propria interfaccia.

- **Shadow DOM**: un set di API JavaScript per collegare e incapsulare un albero DOM `in ombra`, chiamato `shadow DOM`, all'albero del DOM principale di una pagina, chiamato `light DOM`. Lo `shadow DOM` viene renderizzato in modo separato dal DOM principale, in questo modo se ne possono mantenere separate le funzionalità con anche il vantaggio di poter definire lo stile del componente in maniera totalmente autonoma rispetto allo stile globale.

- **HTML template**: tramite l'utilizzo dei tag `<template>` e `<slot>` permette di definire una struttura HTML che non viene renderizzata immediatamente, ma che può essere utilizzata più volte come base per la struttura di un componente personalizzato.

- **HTML Module**: merita una menzione anche il successore degli `HTML Import`[6], al momento ancora un proposal[7] e in fase di sviluppo, la loro introduzione consentirà di includere e riutilizzare template HTML come moduli, similmente a come è già possibile fare per JavaScript con i JavaScript Module[8].

CustomElementRegistry

Il `CustomElementRegistry`[9] è un'interfaccia che permmette di registrare ed interrogare i custom element all'interno dell'istanza del browser dove questi vengono utilizzati. La funzionalità è accessibile tramite l'oggetto `window.customElements`, da cui è possibile richiamare le seguenti funzioni [16]:

- *define*: definisce un nuovo custom element, che può essere di tipo *customized* se eredita/estende un tag HTML esistente, o *autonomous* se

[6] `https://www.w3.org/TR/html-imports/`

[7] `https://github.com/WICG/webcomponents/blob/gh-pages/proposals/html-modules-explainer.md`

[8] `https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Modules`

[9] `https://developer.mozilla.org/en-US/docs/Web/API/CustomElementRegistry`

invece non eredita da un tag HTML esistente. Prevedue due diverse sintassi, `define(name, constructor, options)` nel primo caso e `define(name, constructor)` nel secondo caso, dove `name` fa riferimento al tag HTML da utilizzare per richiamare il custom element, `constructor` la funzione costruttore dell'elemento o il riferimento alla classe, `options` parametro opzionale che controlla come l'elemento viene definito, il cui unico campo supportato ad oggi è `extends` il quale specifica da quale elemento HTML esistente il componente estende.

- *whenDefined*: dato in input il tag di un custom element, ritorna una `Promise` che risolve quando tale componente viene definito all'interno del browser. Utile nei casi in cui la definizione del componente impieghi più tempo del caricamento della pagina oppure se questa è volutamente posticipata, in questo modo è possibile conoscere l'esatto momento in cui quel componente sarà disponibile ed agire di conseguenza. La sintassi è `whenDefined(name)`.

- *get*: a partire dal tag di un custom element, ritorna il costruttore utilizzato per la sua definizione. La sintassi è `get(name)` e tale funzionalità è segnata come sperimentale.

Form-associated custom element

Quando si decide di incapsulare un custom element all'interno di uno shadow DOM, si ottiene il vantaggio di isolare lo stile del componente da quello dell'applicazione in cui viene incluso, ma si perde la naturale interazione che i componenti di input HTML hanno quando sono figli di un componente `<form>`.

Per ovviare a questa problematica, è possibile aggiungere, all'interno della classe del componente personalizzato che si vuole sviluppare, una proprietà statica chiamata `formAssociated` valorizzata a `true`. Così facendo sarà possibile sfruttare le funzionalità messe a disposizione dall'interfaccia `ElementInternals` per poter correttamente far comunicare il custom element

all'interno dello shadow DOM con un eventuale elemento `<form>` presente nel light DOM [17].

```
class MyCheckbox extends HTMLElement {
  static formAssociated = true;
  static observedAttributes = ['checked'];

  constructor() {
    super();
    this._internals = this.attachInternals();
    this.addEventListener('click', this._onClick.bind(this));
  }

  get form() { return this._internals.form; }
  get name() { return this.getAttribute('name'); }
  get type() { return this.localName; }

  get checked() { return this.hasAttribute('checked'); }
  set checked(flag) { this.toggleAttribute('checked', Boolean(flag)); }

  attributeChangedCallback(name, oldValue, newValue) {
    // name will always be "checked" due to observedAttributes
    this._internals.setFormValue(this.checked ? 'on' : null);
  }

  _onClick(event) {
    this.checked = !this.checked;
  }
}
customElements.define('my-checkbox', MyCheckbox);
```

Listing 3: Definzione di un custom element con proprietà statica `formAssociated` per interoperare nativamente con i `form`

Definendo il custom element come nel listato 3, è possibile utilizzarlo come un comune elemento di input dei form, come mostrato nel listato che segue:

```
<form action="..." method="...">
  <label>
    <my-checkbox name="agreed"></my-checkbox>
    I read the agreement.
  </label>
  <input type="submit">
</form>
```

Listing 4: Uso del componente definito nel listato 3

Cicli di vita

Dal momento che un custom element è creato sino al momento della sua distruzione possono avvenire diverse situazioni: l'elemento viene inserito nel DOM, viene aggiornato a seguito di un'interazione con la UI o viene rimosso dal DOM.

Gli scenari appena citati rappresentano i cicli di vita di un elemento, sfruttabili dagli sviluppatori tramite determinate funzioni chiamate `Custom Element Reactions` [18]. Queste funzioni vengono vengono richiamate con particolare cura, in ordine, per prevenire che il codice dell'utente venga eseguito durante un processo delicato. La loro invocazione viene volutamente ritardata sino a quando è sicuro farlo. Inoltre, per assicurarsi che siano invocate nello stesso ordine con il quale sono state scatenate, ogni custom element ha una coda di custom element reaction dedicata [19] [20] [21].

I cicli di vita disponibili sono:

- *constructor()*: la presenza del costruttore in un custom element è mandatoria, viene richiamato quando un elemento precedentemente creato diventa definito tramite il metodo `customElement.define()`.

- *connectedCallback()*: invocata quando un elemento è aggiunto al DOM. Da quel momento si è certi che l'elemento è disponibile nel DOM e si possono impostare con tranquillità attributi, recuperare risorse, eseguire codice di setup o renderizzare template. Importante ricordare che

questa funzione può essere richiamata anche più volte all'interno del ciclo di vita dell'elemento.

- *disconnectedCallback()*: in contrapposizione alla precedente, questa funzione viene invocata quando un elemento viene rimosso dal DOM, il momento migliore per eseguire codice di pulizia e per liberare risorse in modo da evitare perdita di memoria. Come `connectedCallback()`, anche questa funzione può essere richiamata più volte durante il periodo di vita del componente, bisogna quindi prestare attenzione al codice che viene inserito all'interno.

- *attributeChangedCallback(name, oldValue, newValue)*: la sua esecuzione viene scatenata quando gli attributi dell'elemento vengono aggiunti, rimossi, aggiornati o sostituiti o quando l'istanza del componente viene aggiornata. Per una questione di ottimizzazione delle performance, non tutti gli attributi vengono *osservati*, ma solo quelli che vengono ritornati all'interno dell'array del metodo statico `static get observedAttributes`. I parametri della funzione rappresentano, rispettivamente, il nome dell'attributo, il valore che possedeva precedentemente e il valore che possiede attualmente.

- *adoptedCallback()*: funzione invocata solo in casi specifici in cui sil custom element venga adottato all'interno di un nuovo documento, solitamente per via di un elemento `<iframe/>`. Nel momento dell'adozione il custom element non viene distrutto e ricreato, quindi le altre funzioni come `constructor`, `connectedCallback` e `disconnectedCallabck` non vengono invocate.

- *formAssociatedCallback(form)*: richiamato quando il browser associa o dissocia il componente con un elemento di un form.

- *formDisabledCallback(disabled)*: invocato dopo che lo stato `disabled` dell'elemento cambia, sia perché l'attributo dell'elemento viene aggiunto o rimosso, o perché viene modificato lo stato `disabled` di un elemen-

to padre del componente. Il parametro della funzione indica il nuovo valore per lo stato `disabled` dell'elemento.

- *formResetCallback()*: questa funzione viene chiamata quando il form padre dell'elemento viene resettato.

- *formStateRestoreCallback(state, mode)*: richiamato quando il browser ripristina lo stato dell'elemento (per via di una navigazione, ad esempio), oppure quando la funzionalità di autocompilazione del browser imposta un valore nell'elemento. Nel primo caso il parametro `mode` avrà il valore `restore`, mentre nel secondo caso il valore sarà `autocomplete`.

```
class MyComponent extends HTMLElement {
    static get observedAttributes() { return ['first', 'second']; }

    // Il costruttore con l'istruzione super() è obbligatoria
    constructor() {
        super();
        /*code here*/
    }

    connectedCallback() { /*code here*/ }

    attributeChangedCallback(name, oldValue, newValue) { /*code here*/ }

    adoptedCallback() { /*code here*/ }

    disconnectedCallback() { /*code here*/ }

    formAssociatedCallback(form) { /*code here*/ }

    formDisabledCallback(disabled) { /*code here*/ }

    formResetCallback() { /*code here*/ }

    formStateRestoreCallback(state, mode) { /*code here*/ }
}

customElements.define('my-component', MyComponent);
```

Listing 5: Esempio dei cicli di vita di un custom element

Approccio

Le API rese disponibili permettono la creazione, definizione, registrazione e uso di un Web Component all'interno del DOM secondo il seguente approccio:

1. Creare una **classe** usando la sintassi `ECMAScript 2015`[10] dove vengono specificate le funzionalità del componente

2. Registrare la classe come un custom element tramite l'utilizzo dell'API `CustomElementRegistry.define(...args)`

3. Se richiesto, collegare all'elemento uno shadow DOM tramite l'API `Element.attachShadow(...args)` ed aggiungere elementi figli o stare in ascolto di particolari eventi del DOM usando i normali metodi che sono disponibili anche nel light DOM

4. Se richiesto, definire un template personalizzato da attaccare allo shadow DOM tramite i tag `<template>` e `<slot>`

5. Utilizzare il componente così definito all'interno della propria pagina HTML.

Di seguito un esempio di codice tratto da un repository ufficiale di Mozilla[11].

[10] `https://262.ecma-international.org/6.0/`

[11] `https://github.com/mdn/web-components-examples/tree/main/simple-template`

```
class MyParagraph extends HTMLElement {
    constructor() {
        super();

        const template = document.getElementById('my-paragraph');
        const templateContent = template.content;

        this.attachShadow({mode: 'open'}).appendChild(
         templateContent.cloneNode(true)
        );
    }
}

customElements.define('my-paragraph', MyParagraph);
```

Listing 6: Definizione classe Web Component e relativa registrazione nel CustomRegistry

Nel listato 6 di qui sopra è definita la classe `MyParagraph` che contiene un semplice costruttore, dove vengono eseguite le operazioni minime di recupero di una copia del template (in questo caso è stato definito esternamente) e il suo inserimento all'interno dello shadow DOM del componente, in modo che eventuale stile proveniente dalla pagina non possa influire sullo stile interno.

```html
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>Simple template</title>
  <script src="main.js" defer></script>
</head>
<body>
  <h1>Simple template</h1>

  <template id="my-paragraph">
    <style>
      p {
        color: white;
        background-color: #666;
        padding: 5px;
      }
    </style>
    <p><slot name="my-text">My default text</slot></p>
  </template>

  <my-paragraph>
    <span slot="my-text">Let's have some different text!</span>
  </my-paragraph>

  <my-paragraph>
    <ul slot="my-text">
      <li>Let's have some different text!</li>
      <li>In a list!</li>
    </ul>
  </my-paragraph>

</body>
</html>
```

Listing 7: Uso del componente personalizzato all'interno di una pagina HTML

Il listato 7 mostra come un Web Component possa essere importato (riga 6) ed utilizzato (riga 22-31).

Questo appena descritto è l'approccio tradizionale fornito dalle specifiche ufficiali. Ad oggi l'implementazione, registrazione ed utilizzo dei Web Component sono resi più semplici grazie alla nascita di varie librerie e supporto da parte di alcuni framework.

1.4 Librerie e supporto

à In questi ultimi anni l'interesse nei confronti dei Web Component è aumentato costantemente, portando alla creazione di varie librerie dedicate alla creazione e gestione di intere suite di questa tipologia di componenti, ma anche all'adozione di questo approccio da parte di famosi framework e librerie di creazione di applicazioni Web e interfacce grafiche già presenti sul mercato.

Le principali librerie dedicate alla creazione di Web Component sono `Lit`, `Stencil` e `FAST`, mentre i framework e librerie nate per la creazione di applicazioni Web e interfacce grafiche che hanno aggiunto il supporto alla creazione di Web Component all'interno dei loro pacchetti sono `Angular`, `Vue` e `Svelte`.

React possiede una pagina dedicata[12] nella sua documentazione dove spiega che la libreria e i Web Component sono costruiti per risolvere problemi differenti, ma complementari, e che come sviluppatore ognuno è libero di usare i Web Component nei propri progetti React oppure di usare sintassi React nella definizione di Web Component.

[12] `https://it.reactjs.org/docs/web-components.html`

Lit

Attualmente la più popolare libreria per la creazione di Web Component con oltre 12000 stelle su GitHub[13] e più di 320000 download settimanali[14], ha superato nell'ultimo anno la popolarità di `Stencil`. Precedentemente conosciuta come `LitElement`[15], la quale a sua volta fu l'evoluzione del `Polymer Project`[16], la libreria è di proprietà di Google.

Come riportato nella loro documentazione [22], lo scopo della libreria è quello di creare Web Component in maniera veloce, leggera e semplice, qualità di cui si fa cavallo di battaglia, grazie soprattutto al fatto che ogni componente di Lit in sé segue gli standard di tale tecnologia.

Lit si proclama come una perfetta soluzione per lo sviluppo di componenti e design system condivisibili, per il progressivo miglioramento di siti HTML o per la costruzione completa di applicazioni Web ricche di feature e altamente interattive. Durante la creazione di un'applicazione Web Lit non si fa mancare la possibilità di poter usare dei Web Component "vanilla" o creati tramite altri strumenti.

Il sito `Custom Elements Everywhere`[17], un progetto nato dalla community con l'obiettivo di testare l'interoperabilità dei Web Component - nello specifico i custom element - con i principali framework e librerie fronted tramite un determinato numero di test per identificare eventuali problematiche ed evidenziare eventuali potenziali fix in fase di implementazione, assegna a Lit un punteggio di 100% di compatibilità.

[13] `https://github.com/lit/lit/`
[14] `https://www.npmjs.com/package/lit`
[15] `https://lit.dev/docs/v1/`
[16] `https://polymer-library.polymer-project.org/3.0/docs/devguide/feature-overview`
[17] `https://custom-elements-everywhere.com/`

Figura 1.3: Supporto di Lit secondo i test eseguiti da `Custom Elements Everywhere` [1]

Stencil

Con più di 10000 stelle su GitHub[18] e più di 340000 download settimanali da NPM[19], figlio del progetto `Ionic`[20], Stencil nasce con l'obiettivo di fornire agli sviluppatori i giusti mezzi per creare potenti e ricche librerie di componenti e design system [23]. A differenza di `Lit` però i componenti scritti tramite questa libreria non sono direttamente utilizzabili come Web Component, Stencil utilizza TypeScript, JSX e CSS per creare componenti che verranno poi compilati in Web Component, fornendo API extra che rendono l'esperienza di sviluppo più semplice.

Stencil genera componenti che possono essere utilizzati direttamente all'interno dei più popolari framework, ma fornisce anche la possibilità di generare componenti nativi dei principali framework esistenti (per ora `Angular`, `Vue` e `React`) tramite l'utilizzo della funzionalità `Output Target`[21].

Così come per Lit, anche per Stencil `Custom Elements Everywhere` assegna un punteggio di 100% di compatibilità con la tecnologia Web Component.

[18] `https://github.com/ionic-team/stencil`
[19] `https://www.npmjs.com/package/@stencil/core`
[20] `https://ionicframework.com/`
[21] `https://stenciljs.com/docs/output-targets`

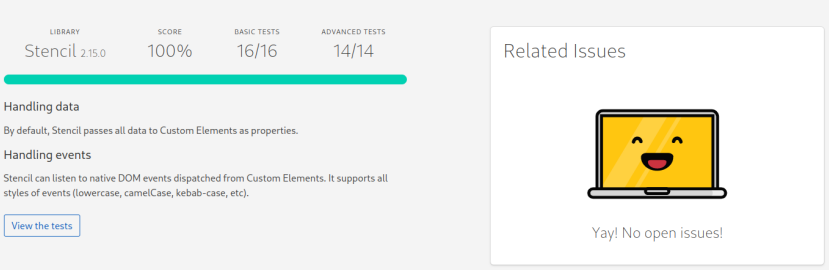


Figura 1.4: Supporto di Stencil secondo i test eseguiti da `Custom Elements Everywhere` [1]

FAST

Di proprietà di Microsoft, con più di 7000 stelle su GitHub[22] e più di 5000 download settimanali[23], FAST è una collezione di tecnologie create sulla base dei Web Component e standard Web moderni, progettato per aiutare gli sviluppatori ad affrontare in maniera efficace alcune delle più comuni sfide nella progettazione e sviluppo di siti ed applicazioni Web [24].

Essendo costruito direttamente sugli standard Web Component di W3C, FAST non ha un proprio modello di componente, permettendo ai componenti generati da questa libreria di funzionare esattamente come gli elementi HTML nativi. Non c'è necessità di utilizzare un particolare framework per usare i componenti creati con FAST, ma questi ultimi possono essere usati in combinazione con qualsiasi libreria e framework a scelta.

A differenza delle due librerie precedenti, su `Custom Elements Everywhere` non sono presenti test che possano dare un riscontro del supporto che FAST offre, ma dalle premesse si può immaginare che sia simile a quello di `Lit` e `Stencil`.

[22] `https://github.com/microsoft/fast/`
[23] `https://www.npmjs.com/package/@microsoft/fast-components`

Angular

Tramite il pacchetto `@angular/elements`[24], il framework mette a disposizione l'API `createCustomElement()` che funge da ponte per l'interfaccia del componente e le funzionalità di change detection di Angular e quelle pre-esistenti nelle API del DOM, mettendo a disposizione l'infrastruttura del framework al browser.

La conversione avviene passando all'API `createCustomElement` il riferimento al componente Angular da convertire (assieme alle sue dipendenze) e il riferimento all'oggetto `Injector`. Il risultato di questa funzione sarà una classe configurata per generare un'istanza auto-caricabile del componente e potrà essere utilizzata per registrare il custom element tramite il `CustomElementRegistry`. Da quel momento sarà possibile utilizzare l'elemento registrato come un qualsiasi elemento HTML.

Il processo di conversione analizza il componente Angular, mappando le proprietà di input in attributi per il custom element, mentre gli eventi generati dalle proprietà di output non saranno più gestiti tramite il costrutto `EventEmitter` ma come costrutti dell'HTML `Custom Event`.

`Custom Elements Everywhere`[25] assegna ad Angular un punteggio di 91% di compatibilità.

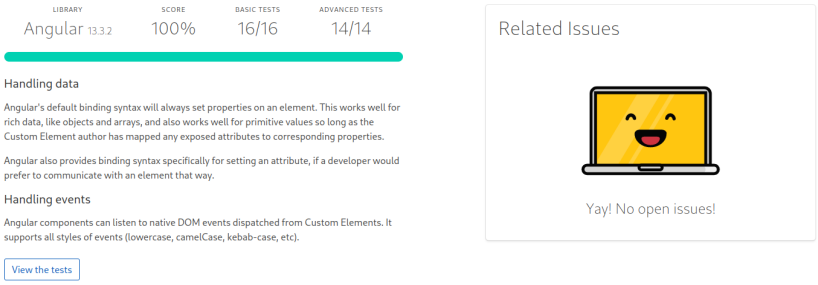


Figura 1.5: Supporto di Angular secondo i test eseguiti da `Custom Elements Everywhere` [1]

[24] `https://angular.io/guide/elements#using-custom-elements`
[25] `https://custom-elements-everywhere.com/`

Vue

Vue supporta la creazione di custom element usando esattamente le stesse API dei componenti Vue tramite l'utilizzo del metodo `defineCustomElement()`, il cui oggetto di ritorno è il costruttore che estende `HTMLElement`[26].

Anche in questo framework, il custom element generato con Vue conterrà internamente al proprio shadow root un'istanza del componente Vue quando `connectedCallabck` viene invocata per la prima volta, mentre all'invocazione della `disconnectedCallback` l'istanza verrà rimossa solo se il componente viene rimosso dal documento HTML.

La gestione delle proprietà di Vue è simile alle proprietà di input di Angular, quindi anche queste verranno mappate in attributi del custom element, e gli eventi verranno gestiti come `Custom Event`.

`Custom Elements Everywhere`[27] assegna a Vue un punteggio di 91% di compatibilità. Infatti Vue 3, come si può evincere dalla figura 1.6, non supporta lo stare in ascolto di `Custom Event` i cui nomi iniziano con una lettera maiuscola[28].

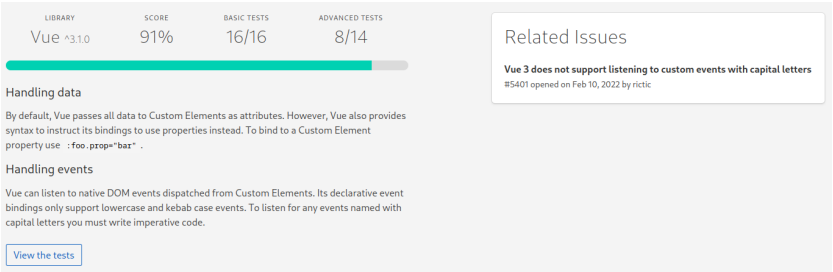


Figura 1.6: Supporto di Vue secondo i test eseguiti da `Custom Elements Everywhere` [1]

[26] `https://vuejs.org/guide/extras/web-components.html#building-custom-elements-with-vue`

[27] `https://custom-elements-everywhere.com/`

[28] `https://github.com/vuejs/core/issues/5401`

Svelte

Svelte offre la possibilità di convertire i propri componenti in custom element attraverso l'opzione del compilatore `customElement: true` e la dichiarazione del tag da utilizzare nell'HTML per richiamare l'uso di quel componente tramite l'inserimento dell'elemento `<svelte:options tag="my-element" />` nel template[29].

`Custom Elements Everywhere`[30] assegna a Svelte un punteggio di 100% di compatibilità.

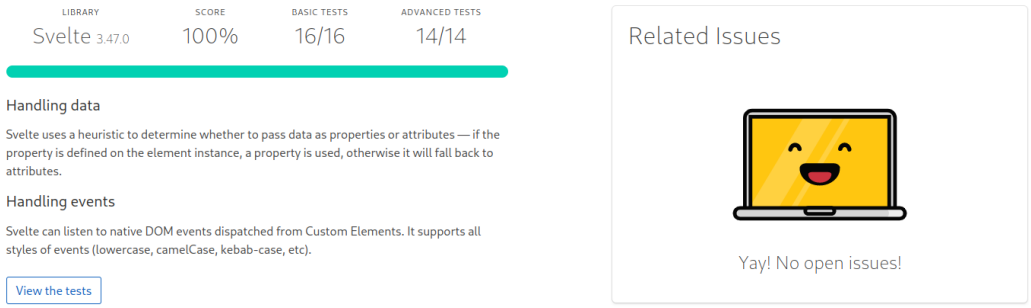


Figura 1.7: Supporto di Svelte secondo i test eseguiti da `Custom Elements Everywhere` [1]

React

`Custom Elements Everywhere`[31] assegna a React un punteggio di 71% di compatibilità e a React Experimental un punteggio di 100% di compatibilità. Questo perché i lavori per supportare completamente i Web Component in React sono per ora presenti solo su una branch che il team non ha ancora deciso di includere nel ramo principale, una issue[32] su GitHub per discutere

[29] `https://svelte.dev/docs#run-time-custom-element-api`
[30] `https://custom-elements-everywhere.com/`
[31] `https://custom-elements-everywhere.com/`
[32] `https://github.com/facebook/react/issues/11347#issuecomment-988970952`

sulle tempistiche e altre funzionalità da aggiungere è stata aperta ad Ottobre 2017.

Le principali problematiche presenti nel pacchetto core riguardano la gestione delle `Prop` e la gestione degli eventi.

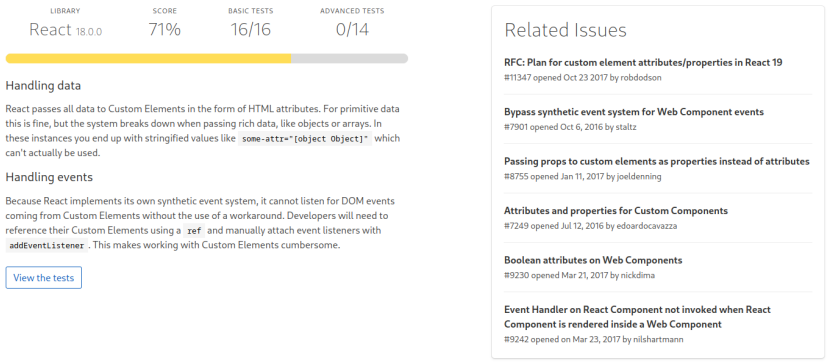


Figura 1.8: Supporto di React secondo i test eseguiti da `Custom Elements Everywhere` [1]

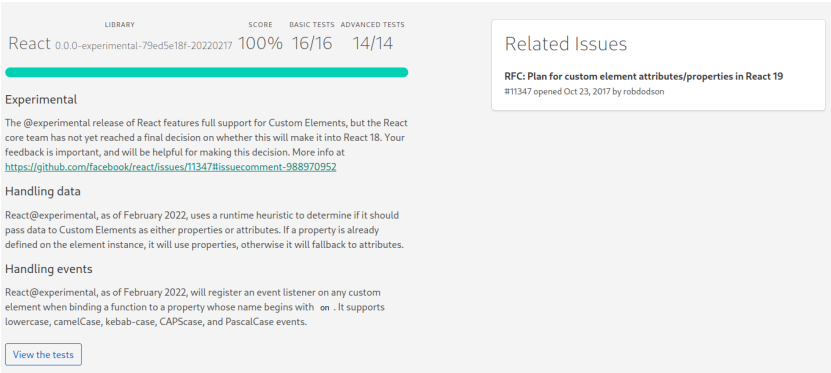


Figura 1.9: Supporto di React Experimental secondo i test eseguiti da `Custom Elements Everywhere` [1]

Capitolo 2

Il caso di studio

Al giorno d'oggi le aziende non possono tralasciare la cura delle proprie interfacce utente. Con l'avvento degli smartphone e dei tablet che sempre più vengono utilizzati per la navigazione Web, sono aumentate le aspettative dell'utente e, di conseguenza, la necessità di produrre applicazioni Web che soddisfino tali aspettative.

È anche diventato più semplice realizzare nuove applicazioni e ciò ha aumentato le dimensioni dell'offerta di mercato, dalla quale è importante riuscire a distinguersi.

In questo capitolo verrà riportato il contesto del caso di studio che avrà luogo all'interno del Gruppo Maggioli, organizzazione italiana di rilievo soprattutto nel mercato pubblico che ha visto un'importante crescita in questi ultimi anni.

2.1 Gruppo Maggioli

2.1.1 La storia

La storia della famiglia Maggioli come impresa inizia nel 1905, in un laboratorio a Santarcangelo di Romagna, con la creazione di prodotti artigianali per i privati locali, mentre 25 anni dopo, nel 1930, tali prodotti vengono creati e venduti anche per le scuole pubbliche del territorio.

L'attività di tipografia e di creazione di modulistica per gli Enti Locali, tutt'ora attivi, viene avviata nel 1950.

Nel 1972, a poco più di due decenni, la famiglia intraprende un nuovo cammino nell'editoria con l'obiettivo di pubblicare prodotti di livello professionale per la Pubblica Amministrazione e i liberi professionisti. Nasce così CEM Casa Editrice Maggioli, che nel 1978 cambia nome nell'odierna Maggioli Editore.

Nel 1982 nasce il marchio Conespo, oggi Maggioli Formazione, con lo scopo di elargire corsi per amministratori, dirigenti e funzionari della Pubblica Amministrazione.

Nel 1988 la società decide di avviare un nuovo processo di business, lo sviluppo tecnologico che investe l'intera società accompagna la crescita e l'affermazione di Maggioli Informatica con l'offerta di software, servizi e progetti che anticipano il processo di Digital Transformation per la Pubblica Amministrazione e le Aziende che segnerà gli anni successivi [25].

La forte crescita in termini finanziari e di volume del Gruppo Maggioli ha aumentato il fatturato e gli utili, permettendo di allargare i possibili orizzonti a disposizione dei manager. Nel corso degli ultimi anni il Gruppo ha infatti acquisito ed integrato all'interno dell'area informatica molte aziende più piccole, sia sul territorio nazionale che estero, con l'obiettivo di partecipare a bandi di progetti europei innovativi, diversificare le attività di business e per ampliare le aree geografiche di mercato raggiungibili.

Le ultime operazioni di rilievo in ambito nazionale riguardano l'acquisizione dell'azienda GLOBO, dotata di consolidata esperienza nel campo dei Sistemi Informativi Geografici, Injenia, che si prodiga nell'aiutare le aziende nella loro trasformazione digitale, Deepcyber, specializzata in cyber security, intelligenza artificiale e antifrode, concludendo con Synapsys, focalizzata su progetti di Software e System Integration per la Pubblica Amministrazione con focus sul settore della Sanità [25].

In ambito internazionale, dal 2016 Maggioli allarga ufficialmente i propri

confini estendendosi in America Latina con Maggioli Latam S.A.S a Bogotà in Colombia, in Spagna con Infaplic SL e Atm Dos SL, le quali verranno successivamente fuse con Galileo Sa, presente a Tenerife nelle Isole Canarie, per formare ATM Gruppo Maggioli [26, 27]

Figura 2.1: Mappa delle sedi del Gruppo Maggioli [28].

Ad oggi il Gruppo Maggioli conta 140000 clienti, 70 tra sedi e filiali (considerando territorio nazionale e internazionale, posizione mostrata nell'immagine 2.1) e più di 2600 persone in continua crescita, di cui circa 250 tecnici informatici [25] [29].

Il bilancio consolidato del 2020 raggiunge i 186 milioni di euro di ricavi, mentre nel 2021 chiude con un bilancio consolidato che raggiunge i 244 milioni di euro [29], superando di gran lunga la previsione di 210 milioni di euro, questo grazie anche alla pronta risposta dell'azienda nel periodo della pandemia di rendere disponibile il remote working ad oltre l'88% dei dipendenti, operazione divenuta possibile grazie al grande supporto tecnico del team IT. Quella del remote working è una soluzione che l'azienda ha mantenuto, per il momento, anche oltre la fine dell'emergenza sanitaria, venendo incontro a quelle che sono le esigenze e le richieste che i dipendenti e il mercato chiede oggi.

La sede principale del ramo informatico si trova a Santarcangelo di Romagna (RN) di fronte alla sede legale.

2.1.2 Struttura aziendale

Nonostante siano passati più di 100 anni dalla sua nascita, Maggioli rimane ad oggi ancora un'impresa fortemente a conduzione familiare. Con riferimento all'organigramma aziendale aggiornato a Luglio 2022, la struttura principale vede al vertice Manlio Maggioli nelle vesti di Presidente Onorario. All'interno del Consiglio di Amministrazione, insieme al capofamiglia Manlio, sono presenti anche i tre figli Paolo, Amalia e Cristina, assieme ad una figura legale e ad un esperto responsabile dei processi di business e della Sicurezza e Ambiente.

Paolo Maggioli ricopre i ruoli di Presidente e Amministratore Delegato, la sorella Cristina è Responsabile delle Risorse Umane mentre Amalia è responsabile delle aree Commerciale, Marketing ed Estero le quali prevedono le sotto-aree Rete Commerciale per la Pubblica Amministrazione, Supporto Commerciale, Marketing e Comunicazione Esterna.

In aggiunta, i tre fratelli ricoprono anche il ruolo di Presidente o Vice Presidente in un terzo delle società partecipate. È evidente come la famiglia Maggioli abbia ancora in mano le redini dell'intero Gruppo nonostante le recenti acquisizioni che hanno visto aumentare di molto il numero di sedi, di fatturato e di dipendenti.

2.2 Prodotti

Seguirà ora una breve presentazione dei principali prodotti di Maggioli dove il lavoro di tesi è stato integrato.

Una delle strategie di Maggioli Editore per innovare l'offerta editoriale consiste nell'arricchire la produzione cartacea con contenuti totalmente digitali accessibili in modalità multicanale.

Questa scelta è pilotata principalmente da due obiettivi. Innanzitutto si vuole rendere interattiva la consultazione dell'opera (libro o rivista) consentendo tipiche funzionalità come la ricerca nei testi, i suggerimenti di contenuti correlati, l'integrazione di audio-video esplicativi o di approfondimento. In secondo luogo questa opportunità riduce l'obsolescenza del volume cartaceo poiché, oltre ad abilitare la consultazione del volume in modalità digitale, consente di distribuire continuamente nuovi contenuti che aiutano l'Editore a rendere sempre aggiornati i volumi nel tempo. Si pensi, ad esempio, quanto questo sia importante per tenere allineati i riferimenti normativi per i volumi dell'area legale, commerciale e tecnica e per la Pubblica Amministrazione. Per ottenere questi risultati, l'offerta tipicamente "cartacea" viene integrata e, a volte, completamente sostituita da piattaforme web e mobile di consultazione digitale. Dal punto di vista della *User Experience* (UX), della costruzione della *brand identity* e dell'ottimizzazione dei tempi di sviluppo delle nuove piattaforme, le principali scelte tecniche hanno riguardato la creazione e l'adozione di un *Design System* condiviso da tutte le applicazioni e il consolidamento di tutta la produzione editoriale in una banca dati documentale nativamente cloud che abilita l'accesso ai documenti da qualsivoglia canale tramite interfacce dedicate (API).

2.2.1 Periodici Maggioli

Periodicimaggioli.it è la piattaforma dedicata alle riviste edite da Maggioli. Suddivisa in nove aree tematiche (Amministrazione e PA Digitale, Appalti e Contratti, Polizia e Attività Economiche, Edilizia Urbanistica e Ambiente, Personale e organizzazione, Sanità e Sociale, Scuola, Servizi demografici, Bilancio Contabilità e Tributi), conta 16 testate e contiene, ad oggi, oltre 2000 fascicoli. I clienti tipici delle riviste sono amministratori e dipendenti pubblici e professionisti che lavorano a vario titolo con la PA.

Ogni fascicolo contiene in media una ventina di contributi specializzati scritti da professionisti che collaborano come autori in qualità di esperti dell'area tematica di riferimento. Le redazioni interne, in accordo con gli autori, si occupano di produrre contenuti di approfondimento da distribuire in modalità esclusivamente digitale con i fascicoli (ad esempio raccolte di normativa, esempi pratici, moduli precompilati, interviste etc.).

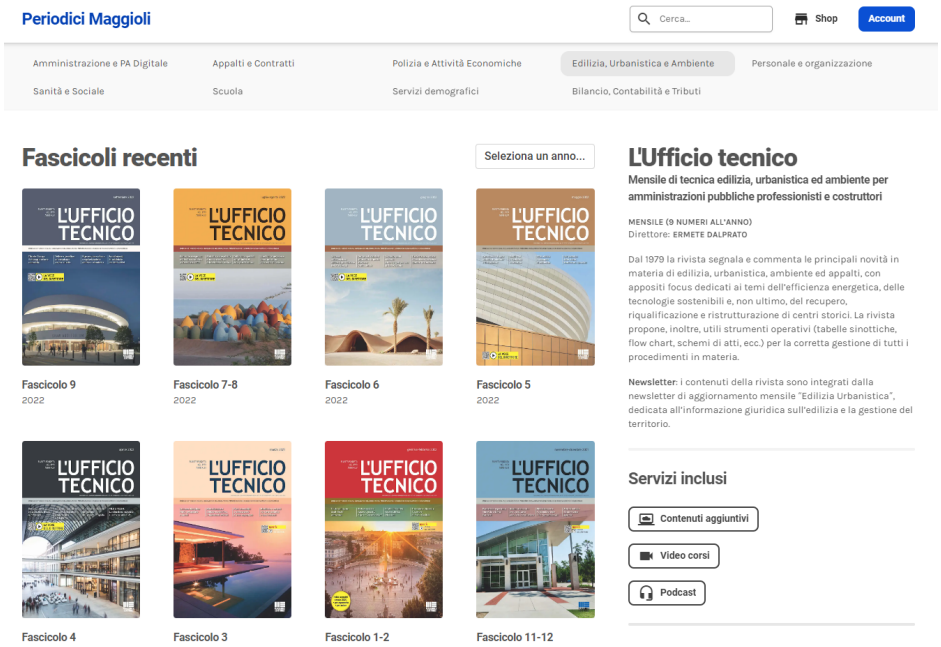


Figura 2.2: Sito di Periodicimaggioli.it [2]

La piattaforma consente agli abbonati l'accesso alla versione digitale della rivista che riproduce fedelmente in formato HTML la veste grafica della versione cartacea. Nella consultazione online è presente un indice navigabile dei contenuti, la ricerca *full text* nell'intero fascicolo e la possibilità da parte del cliente di organizzare una propria sezione dei preferiti con una granularità che può arrivare al singolo articolo della rivista. Ogni fascicolo può essere arricchito con contenuti aggiuntivi direttamente consultabili dalla piattaforma tramite appositi *widget*.

I contenuti aggiuntivi sono essenzialmente:

- fascicoli di materiale di approfondimento;
- video-lezioni a tema realizzati dalla redazione di Maggioli Formazione;
- *podcast* registrati con i direttori delle riviste o altre personalità di rilievo;
- una sezione di Q&A con cui ogni cliente ha la possibilità di rivolgere un certo numero dei quesiti complessi ai quali gli autori daranno una risposta.

La nuova piattaforma rilasciata a maggio 2022, sviluppando usando tecnologie moderne come Spring Framework[1] per il backend e NextJS[2] per il frontend, ha sostituito una precedente versione, sviluppata usando il CRM WordPress[3], arricchendone le funzionalità e adottando i princìpi e le componenti del *Design System* aziendale.

2.2.2 Synbee

Synbee è un progetto in corso che si propone di realizzare un modello innovativo di distribuzione dei contenuti editoriali pubblicati da Maggioli (articoli su siti web tematici, libri, riviste, webinar formativi). Il principale deliverable di Synbee sarà una piattaforma *cloud* rivolta ai professionisti (sia studi che *freelance*) per la gestione del proprio lavoro quotidiano con un'attenzione particolare al *team working*. Come strumento di gestione di "pratiche" offre i tipici strumenti come calendario, organizzazione e storage della documentazione, organizzazione di attività, template di pratiche, *editing* collaborativo di documenti direttamente in piattaforma e integrazioni con strumenti di uso comune come mail e PEC, fatturazione elettronica, *instant messaging* etc. L'applicazione consente di "configurare" ogni singola pratica

[1] `https://spring.io/`
[2] `https://nextjs.org/`
[3] `https://wordpress.com/it/`

coinvolgendo attivamente nella gestione dei task uno o più professionisti di uno studio, consulenti esterni e il cliente stesso.

Synbee è rivolta ai professionisti di qualunque area. Quindi, oltre alle funzionalità comuni citate in precedenza, ogni cliente può specializzare il proprio *workspace* adottando moduli "pluggabili" specifici nell'ottica di una *Composable Application*. Ad esempio un viewer di file *cad* per ingegneri e architetti, l'integrazione con il processo telematico per gli avvocati, tool di analisi di bilancio per i commercialisti etc..

La piattaforma si propone anche di aiutare i professionisti suggerendo dei contenuti della produzione editoriale di Maggioli attinenti i temi e gli argomenti della singola pratica. I contenuti possono essere ad accesso libero o acquistabili in app e sono principalmente post di siti web tematici, volumi, articoli di riviste, webinar e/o video-lezioni.

Figura 2.3: Estratto dal sito di Synbee ancora in sviluppo

Il contesto della pratica viene dedotto applicando algoritmi di *Machine Learning* alle attività e ai documenti che compongono la pratica stessa. Altri algoritmi eseguono il *matching* tra il contesto della pratica e i contenuti da suggerire. Tutti i contenuti editoriali sono analizzati con algoritmi di *Natural Language Processing* (NLP) per ricavare entità rilevanti (persone, luoghi, enti o aziende, riferimenti normativi) e i "concetti" principali del contenuto stesso. Dai contenuti audio/video vengono preliminarmente estratte le

trascrizioni tramite servizi cognitivi che eseguono lo *Speech to Text*. I risultati del *matching* vengono presentati all'interno della piattaforma con una granularità molto fine con l'obiettivo di agevolare una nuova modalità di formazione continua fatta di brevi contenuti strettamente legati al contesto della pratica.

La prima release di Synbee è prevista entro il 2022, sviluppata anch'essa con tecnologie moderne quali Spring Framework[4] per il backend e Angular[5] per il frontend. Anche la piattaforma Synbee, analogamente a quella di Periodicimaggioli.it, adotta i princìpi e le componenti del *Design System*.

2.2.3 Maggioli Design System

Nel 2020 l'azienda Maggioli ha iniziato i lavori per la progettazione e sviluppo di un design system aziendale, conosciuto come `Maggioli Design System`[6], a cui ci si riferisce col nome di `Magma`. Il progetto, in continua mutazione ed evoluzione, nasce con l'intento di migliorare il *business value* dei prodotti e dei processi aziendali tramite l'adozione di una meta style guide in grado di allineare UI e UX di tutti i prodotti aziendali [30].

2.3 Avvio del progetto

Ed è all'interno del progetto *Maggioli Design System* che nasce la necessità da parte dell'azienda, nella primavera del 2021, di possedere una suite di componenti scritti secondo lo standard attuale da poter poi utilizzare nei vari prodotti di proprietà dell'azienda, siano essi sviluppati con framework frontend come Angular o React, sia che si tratti di piattaforme WordPress.

L'idea è che tale suite non sia un progetto chiuso e sviluppato interamente dai membri del core team, ma che sia un prodotto aperto alla contribuzione da parte dei dipendenti e collaboratori di Maggioli, che sia per la creazione di

[4] `https://spring.io/`
[5] `https://angular.io/`
[6] `https://design-system.maggiolicloud.it/`

nuovi componenti, risoluzione di bug o modifiche strutturali all'architettura utilizzata, con l'intento di creare una community attorno a tale progetto.

Le modalità di distribuzione e di contribuzione di questa suite verranno trattate più in dettaglio nelle sezioni 3.4 e 3.5 del prossimo capitolo.

Capitolo 3

La suite di componenti

In questo capitolo verranno descritte le azioni e scelte intraprese dall'azienda in merito alle tecnologie e metodologie adottate per la realizzazione di una suite di componenti per il progetto già in essere *Design System*.

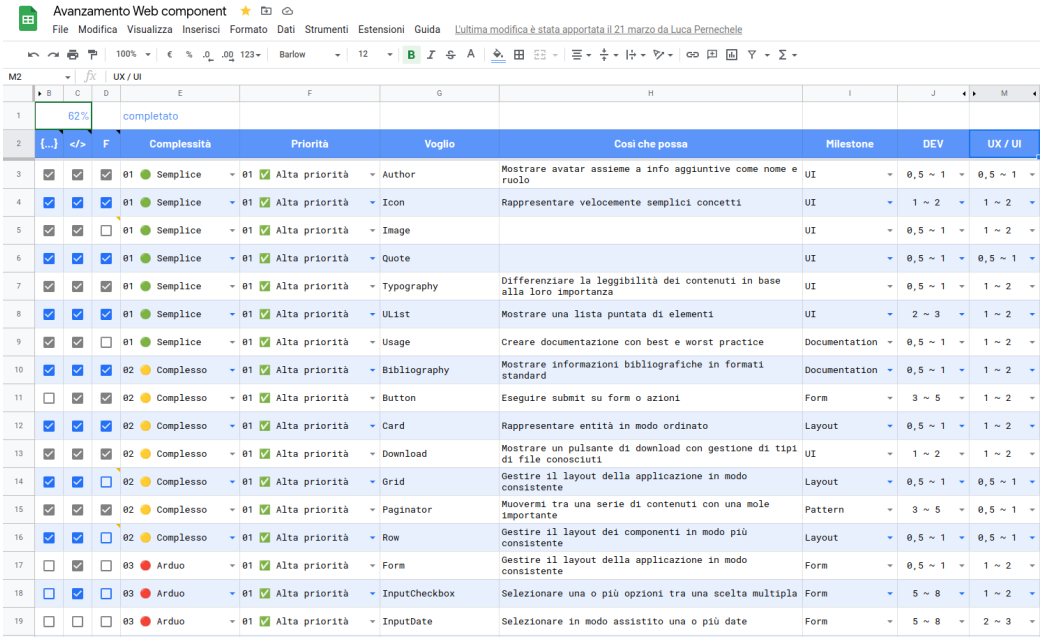
Particolare focus verrà assegnato al modello di contribuzione (sezione 3.5), considerata una delle tematiche più importanti all'interno del progetto, seguita dalla modalità di rilascio adottata (sezione 3.4) e dall'adozione di processi di automazione (sezione 3.6) per lo sviluppo e rilascio dei componenti.

3.1 Individuazione dei componenti

Uno dei primi passi operati dal team è stato individuare una lista di potenziali componenti, considerando validi inizialmente solo quelli di uso più comune e non prettamente inerenti allo specifico dominio applicativo dei vari progetti individuati (quelli descritti nel capitolo 2), integrando poi tale lista con componenti meno comuni ma potenzialmente utilizzabili anche da altri prodotti non inizialmente inclusi tra quelli in cui questa suite di componenti verrà inserita.

Effettuata la cernita dei componenti, gli stessi sono stati riportati su un foglio di calcolo (come si può vedere in figura 3.1) utilizzato in prima battuta per decorarli con informazioni aggiuntive quali la priorità e la complessità di

realizzazione, area di competenza del componente, descrizione del risultato finale che se ne vuole ottenere, stato completamento del componente (suddiviso tra realizzazione del mockup, scrittura codice di logica applicativa e codice di markup HTML) e stima sulle tempistiche per la realizzazione del componente differenziati per quanto riguarda lo sviluppo della logica applicativa e dell'interfaccia grafica / esperienza utente.



Avanzamento Web component

File Modifica Visualizza Inserisci Formato Dati Strumenti Estensioni Guida — L'ultima modifica è stata apportata il 21 marzo da Luca Pernechele

M2 — UX / UI

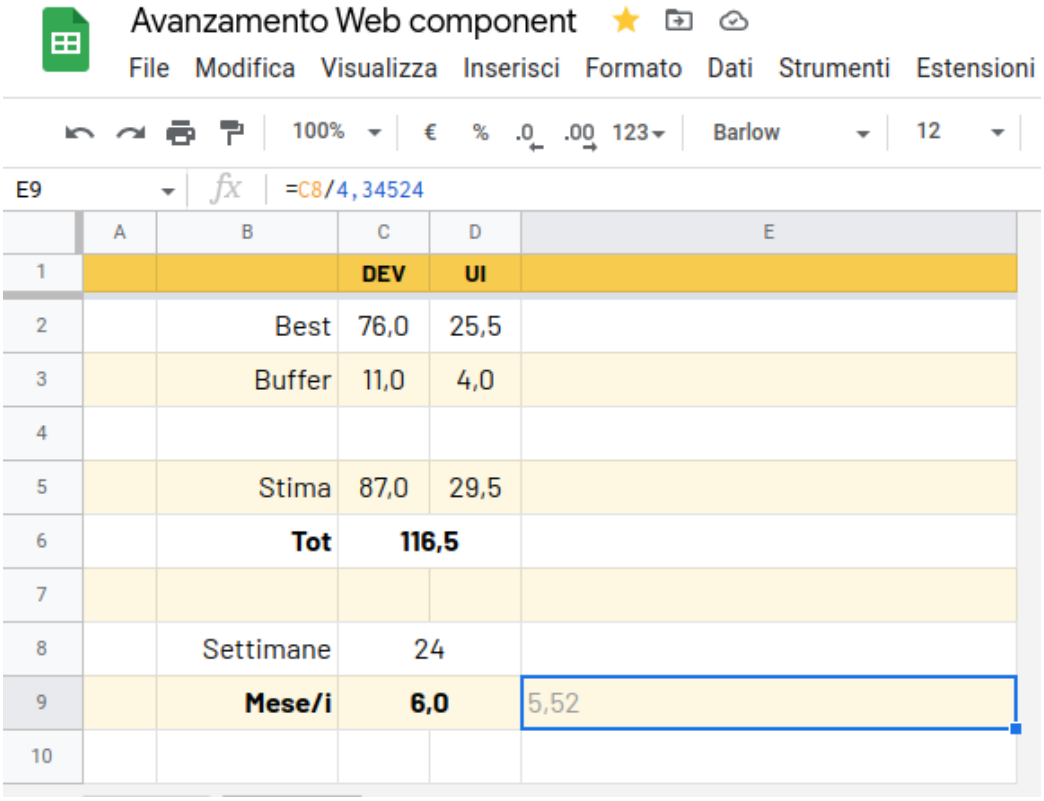
62% completato

{...}	</>	F	Complessità	Priorità	Voglio	Così che possa	Milestone	DEV	UX / UI
☑	☑	☑	01 Semplice	01 Alta priorità	Author	Mostrare avatar assieme a info aggiuntive come nome e ruolo	UI	0,5 ~ 1	0,5 ~ 1
☑	☑	☑	01 Semplice	01 Alta priorità	Icon	Rappresentare velocemente semplici concetti	UI	1 ~ 2	1 ~ 2
☑	☑	☐	01 Semplice	01 Alta priorità	Image		UI	0,5 ~ 1	1 ~ 2
☑	☑	☑	01 Semplice	01 Alta priorità	Quote		UI	0,5 ~ 1	0,5 ~ 1
☑	☑	☑	01 Semplice	01 Alta priorità	Typography	Differenziare la leggibilità dei contenuti in base alla loro importanza	UI	0,5 ~ 1	1 ~ 2
☑	☑	☑	01 Semplice	01 Alta priorità	UList	Mostrare una lista puntata di elementi	UI	2 ~ 3	1 ~ 2
☑	☑	☐	01 Semplice	01 Alta priorità	Usage	Creare documentazione con best e worst practice	Documentation	0,5 ~ 1	1 ~ 2
☑	☑	☑	02 Complesso	01 Alta priorità	Bibliography	Mostrare informazioni bibliografiche in formati standard	Documentation	0,5 ~ 1	1 ~ 2
☐	☑	☑	02 Complesso	01 Alta priorità	Button	Eseguire submit su form o azioni	Form	3 ~ 5	1 ~ 2
☑	☑	☑	02 Complesso	01 Alta priorità	Card	Rappresentare entità in modo ordinato	Layout	0,5 ~ 1	1 ~ 2
☑	☑	☑	02 Complesso	01 Alta priorità	Download	Mostrare un pulsante di download con gestione di tipi di file conosciuti	UI	1 ~ 2	1 ~ 2
☑	☑	☐	02 Complesso	01 Alta priorità	Grid	Gestire il layout della applicazione in modo consistente	Layout	0,5 ~ 1	0,5 ~ 1
☑	☑	☑	02 Complesso	01 Alta priorità	Paginator	Muovermi tra una serie di contenuti con una mole importante	Pattern	3 ~ 5	0,5 ~ 1
☑	☑	☐	02 Complesso	01 Alta priorità	Row	Gestire il layout dei componenti in modo più consistente	Layout	0,5 ~ 1	0,5 ~ 1
☐	☑	☐	03 Arduo	01 Alta priorità	Form	Gestire il layout della applicazione in modo consistente	Form	0,5 ~ 1	1 ~ 2
☐	☑	☐	03 Arduo	01 Alta priorità	InputCheckbox	Selezionare una o più opzioni tra una scelta multipla	Form	5 ~ 8	1 ~ 2
☐	☐	☐	03 Arduo	01 Alta priorità	InputDate	Selezionare in modo assistito una o più date	Form	5 ~ 8	2 ~ 3

Figura 3.1: Alcuni dei Web component individuati con le relative informazioni nel foglio di calcolo

Tali stime sono state poi aggregate all'interno di un secondo foglio all'interno del documento (come si può vedere in figura 3.2) e utilizzate sia come riferimento del tempo necessario allo sviluppo per il team stesso, sia come informazione da condividere con gli organi decisionali. A seguito di un confronto con quest'ultimi, sono state riviste le priorità assegnate ai componenti, cercando di includere nel primo lotto di componenti che saranno rilasciati il più piccolo insieme che possa rappresentare al meglio la poten-

zialità della suite proposta e dei vantaggi che l'adozione di uno standard a livello aziendale può portare.



	A	B	C	D	E
1			DEV	UI	
2		Best	76,0	25,5	
3		Buffer	11,0	4,0	
4					
5		Stima	87,0	29,5	
6		**Tot**	**116,5**		
7					
8		Settimane	24		
9		**Mese/i**	**6,0**		5,52
10					

Figura 3.2: Stime aggregate delle tempistiche della realizzazione della suite di Web component

Le stime presenti nel documento, misurate in giorni uomo e frutto di un analisi preliminare effettuato da i membri del core team, sono state suddivise in cinque elementi diversi per rappresentare al meglio l'impegno necessario da parte del team per poter adeguatamente sviluppare e testare le funzionalità necessarie:

- **0,5 ∼ 1**: riservato principalmente a componenti grafici e che non presentano o presentano una logica applicativa quasi nulla

- **1 ∼ 2**: per componenti che presentano una minima logica applicativa (come può essere la renderizzazione condizionale di alcune parti in base alle proprietà fornite)

- **2 ∼ 3** e **3 ∼ 5**: per quei componenti che presentano anche azioni particolari che possono essere effettuate o che operano in combinazione con altri componenti

- **5 ∼ 8**: principalmente riservato ai componenti di interazione con i form

3.2 Tecnologia adottata

Sin da subito, poiché era già nei piani di sviluppo del design system, l'attenzione è stata rivolta sulla tecnologia dei `Web Component`, in particolare l'argomento è stato sollevato all'interno del progetto Synbee (sezione 2.2.2).

Qui nasce la necessità di fornire, ad aziende terze, degli strumenti che permetterso la creazione di componenti grafici da poter integrare all'interno del prodotto Synbee, sviluppato per la parte client con il framework `Angular`. Diversi sono stati i momenti di discussione che hanno coinvolto i responsabili e le figure tecniche che curano il prodotto e le controparti del team di Ricerca e Sviluppo in merito a quale potesse essere la soluzione migliore da adottare, e durante queste discussioni uno dei punti sul quale si tornava più spesso di tutti era lasciare la libertà di poter generare questi componenti in maniera agnostica dal framework o libreria che Synbee utilizzasse per sviluppare l'interfaccia utente.

Data la premessa, i responsabili hanno così decretato che i tempi fossero abbastanza maturi in azienda per poter dedicare parte delle proprie risorse allo studio e alla implementazione di una serie di componenti personalizzati che potessero essere utilizzati all'interno dei vari prodotti di Maggioli, basati sul già presente design system.

A seguito degli studi e le ricerche condotte sulla libreria o framework da utilizzare per lo sviluppo dei componenti, la scelta è ricaduta sulla libreria `Stencil`.

La decisione è stata presa tenendo in considerazione il livello di maturità della libreria, l'utilizzo da parte della community e la presenza di un'adeguata documentazione. In questo, Stencil è sembrata la scelta migliore rispetto alle alternative al tempo trovate, quali:

- *LitElement*
- *Angular Elements*
- *SkateJS*[1]

`SkateJS` presentava una scarsa documentazione e un palese stato di abbandono da parte del team di sviluppo (ultima versione ufficiale rilasciata nel 2017 [2] e ultima commit sul branch principale nel 2019 [3]).

`Angular Elements` e `LitElement` invece erano sembrate meno mature di `Stencil`, con meno esempi disponibili e meno guide.

Oltre a ciò, c'è un altro motivo più importante di tutti questi, già citato nella sezione 1.4, ed è la funzionalità degli **Output Targets**. Grazie a tale funzionalità, infatti, oltre a poter compilare le classi scritte tramite la libreria per generare custom element scritti con gli standard HTML (e quindi utilizzabili su qualsiasi progetto che sfrutti HTML), è possibile generare i componenti tradotti nel linguaggio del framework target. I framework disponibili con questa funzionalità sono `Angular`, `React` e `Vue`.

Questa possibilità è stata considerata di alto interesse, dato l'utilizzo in azienda dei primi due framework citati per la creazione di applicazioni Web. Ed è principalmente per questo motivo che si è deciso di proseguire con l'adozione della libreria `Stencil`, sebbene ad oggi tale funzionalità non sia stata ancora sfruttata a pieno.

[1] `https://skatejs.netlify.app/`

[2] `https://github.com/skatejs/skatejs/releases`

[3] `https://github.com/skatejs/skatejs/commits/master`

3.2.1 Organizzazione del codice

Poiché non nasce come opera a sé stante ma piuttosto all'interno di un progetto già esistente, come già descritto nella sezione 2.3, il repository del progetto di realizzazione della suite di componenti tramite la libreria *Stencil* è stato inserito all'interno del più ampio repository, gestito come *monorepo*, del *Design System* aziendale.

La gestione di progetti multipli all'interno di un unico repository è stato affidato al tool *Nx*[4] fornito da Nrwl, strumento molto utile in questo contesto che aiuta ad adottare una gestione strutturata del codice condiviso, consente di impostare delle restrizioni di accesso ed aiuta ad ottimizzare i tempi in fase di sviluppo, compilando solo le parti interessate dalle modifiche apportate ed evitando quindi di ettuare operazioni non necessarie [31].

3.2.2 Versionamento

Lo strumento utilizzato per la gestione del controllo di versione (o Version Control System) del monorepo è Git[5], divenuto ormai uno standard all'interno del team di Ricerca e Sviluppo dopo la transizione da SVN[6], mentre il codice viene salvato all'interno dell'area dell'organizzazione sulla piattaforma pubblica GitLab[7].

Per tenere traccia delle problematiche presenti in tutta la fase di vita del progetto, dei bug emersi durante il suo sviluppo, delle attività pianificate e quelle da pianificare, delle milestone prefissate ma soprattuto per avere una panoramica dei progressi raggiunti viene utilizzata l'area dedicate alle *issue* presenti sulla piattaforma GitLab, organizzate tramite l'utilizzo di *labels*[8] semplici e relative ad un specifico ambito, denominate *scoped*[9].

[4] `https://nx.dev/`
[5] `https://git-scm.com/`
[6] `https://subversion.apache.org/`
[7] `https://about.gitlab.com/`
[8] `https://docs.gitlab.com/ee/user/project/labels.html`
[9] `https://docs.gitlab.com/ee/user/project/labels.html#scoped-labels`

Queste etichette sono gestite a livello di singolo progetto e sono state suddivise nelle seguenti macro-categorie:

- *Prodotto*: per distinguere per quale prodotto di Maggioli è stata aperta la issue o merge request
- *Sotto-progetto*: su quale progetto del *Design System* impatta la issue o merge request
- *Priorità*: individua il livello di attenzione da fornire, se per un difetto riscontrato, se si tratta di un miglioramento o se è stato richiesto espressamente da un singolo prodotto
- *Ambito*: utile a distinguere la tipologia di processo a cui fa riferimento
- *Stato*: indica lo stato della lavorazione

Le macro-categorie *Sotto-progetto* e *Ambito* sono state inoltre utilizzate per definire uno standard nella scrittura dei messaggi di commit durante lo sviluppo del progetto, raggiunto grazie all'introduzione dello strumento *commitlint* [32].

La decisione di utilizzare uno strumento del genere nasce dall'esigenza di imporre un determinato rigore al modo in cui vengono generati i messaggi di commit, poiché in un progetto che prevede la contribuzione da parte di più individui, plausibilmente provenienti da team diversi e contesti diversi (ma anche aziende diverse, come nel caso di Maggioli), questi ultimi potrebbero non avere una cultura comune sull'argomento.

3.2.3 Mockup

Passo fondamentale nella realizzazione di componenti grafici, che sono pensati per avere un'interazione con l'utente finale, è la progettazione del loro design e la realizzazione di un prototipo tramite strumenti di mockup.

I mockup hanno il compito di rappresentare come sarà il look & feel dei componenti, dando una concreta idea agli sviluppatori e agli organi di decisione del risultato che si vuole ottenere. Tramite l'utilizzo di questo strumento è inoltre possibile generare diverse varianti grafiche in pochi passi, operazione che potrebbe invece richiedere molto tempo a livello implementativo.

Alcuni strumenti di mockup inoltre non solo rendono possibile la renderizzazione della componente grafica, ma permettono persino di implementare un collegamento tra le diverse viste di uno stesso componente (o tra più componenti), con la finalità di rappresentare anche le funzionalità che il componente dovrebbe avere. Questo, unito con un confronto con il cliente (che può essere l'utente finale o altri team di sviluppo), permette di minimizzare il lavoro necessario in fase si sviluppo del componente.

Anche la revisione del lavorato viene agevolata, dando possibilità ai designer di modificare velocemente i diversi elementi che compongono l'interfaccia per raggiungere una decisione comune quando si devono mettere d'accordo diverse persone [33].

Tra i vari strumenti per la creazione di mockup presenti, per la progettazione del design della suite di Web Component, Maggioli si affida alla piattaforma *Figma*[10], che già utilizza per la definizione delle interfacce grafiche di altri prodotti e particolarmente indicato per la realizzazione di design system.

Feature particolarmente interessati di questa piattaforma sono la possibilità di dividere la lavorazione in diversi progetti, mantenendo ben separato lo scope di ogni lavorato, condividere i componenti realizzati tra i vari progetti e conseguentemente essere in grado di utilizzare un unico stile per tutti i prototipi, uniformando il lavoro svolto da team diversi, infine la possibilità di ottenere snippet di codice autogenerato per CSS, iOS e Android [34].

Figma presenta inoltre una vasta community di sviluppatori da tutto il mondo che condividono template e plugins importabili all'interno del proprio

[10] `https://www.figma.com/`

progetto[11].

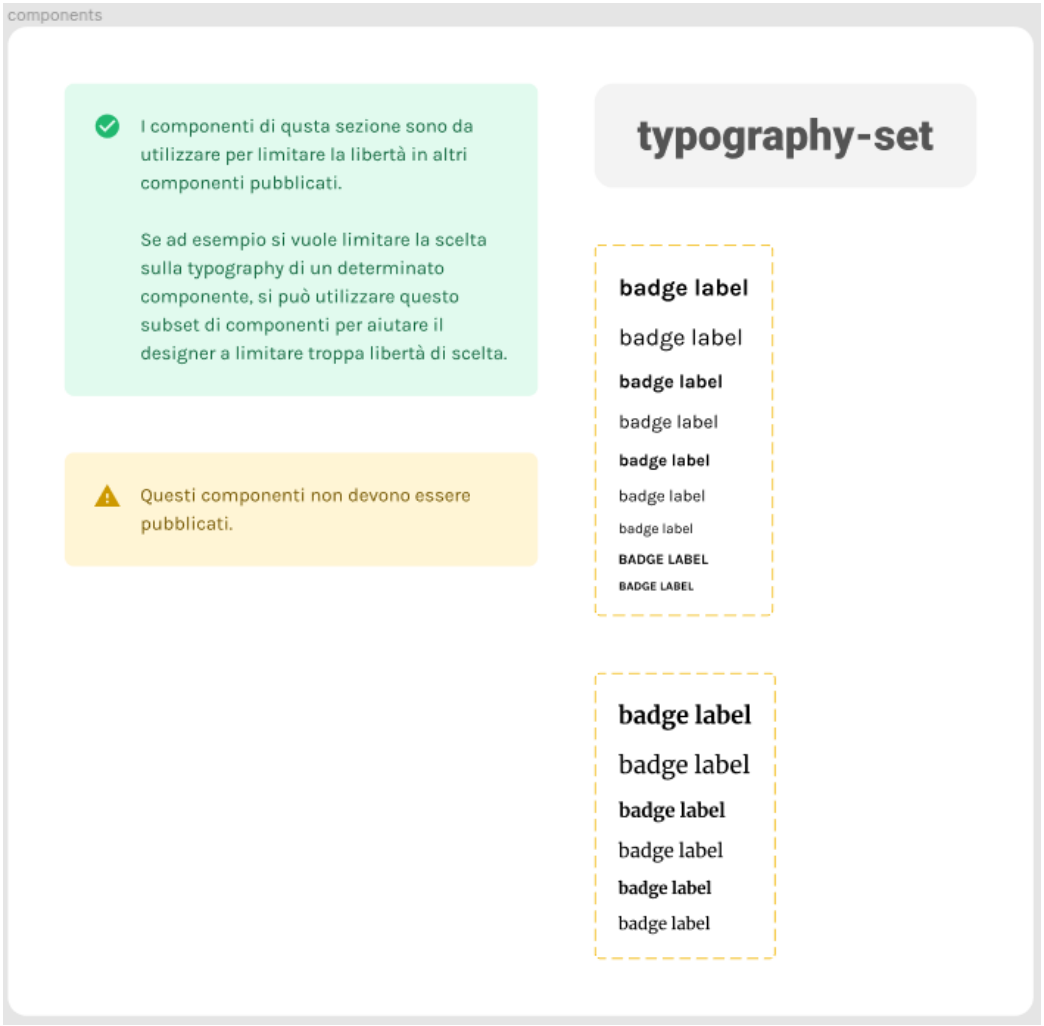


Figura 3.3: Esempio di elementi riutilizzabili nei vari progetti su Figma, in figura la definizione delle varie varianti di tipografia possibili

[11] `https://www.figma.com/community/plugins`

Figura 3.4: Esempio di design di un Web component su Figma, completo delle varianti possibili

3.2.4 Vetrina dei componenti

Sebbene tramite Figma sia possibile replicare le funzionalità che un componente grafico deve implementare, rimane un'esperienza totalmente diversa dal testare il componente quando questo è in fase di sviluppo o viene effettivamente sviluppato.

Risulta altresì utile avere a disposizione un'ambiente pubblico dove questi componenti non solo sia possibile provarli direttamente su un browser, ma sia possibile anche condividerli con gli sviluppatori che devono utilizzarli all'interno dei propri prodotti, senza la necessità che questi ultimi siano dei

contributor all'interno del progetto *Design System* o che debbano effettuare il download della codebase, effettuare la compilazione dei vari sotto-progetti ed infine avviare l'ambiente di visualizzazione dei componenti.

Date queste necessità, in azienda si è fatto largo uso di Storybook[12], strumento open-source per lo sviluppo di interfacce grafiche che rende lo sviluppo più rapido e semplice tramite la possibilità di isolare i componenti. Tramite Storybook è possibile sviluppare intere interfacce grafiche senza la necessità di impostare un complesso stack di sviluppo, forzare la presenza di dati in un database o navigare l'intera applicazione che si sta sviluppando, permettendo allo stesso tempo di associare la dovuta documentazione grazie all'introduzione nella codebase delle *story* [35].

Le story sono gli elementi fondanti di Storybook, permettono di catturare e visualizzare l'aspetto del componente nei vari stati possibili grazie alla definizione di vari tipi di *argomenti* e tramite l'utilizzo di diversi *template*. Non vi è un vincolo sul numero di storie che un componente può avere, sta allo sviluppatore riuscire a rappresentare ogni stato possibile tramite le combinazioni di template e argomenti.

[12] `https://storybook.js.org/`

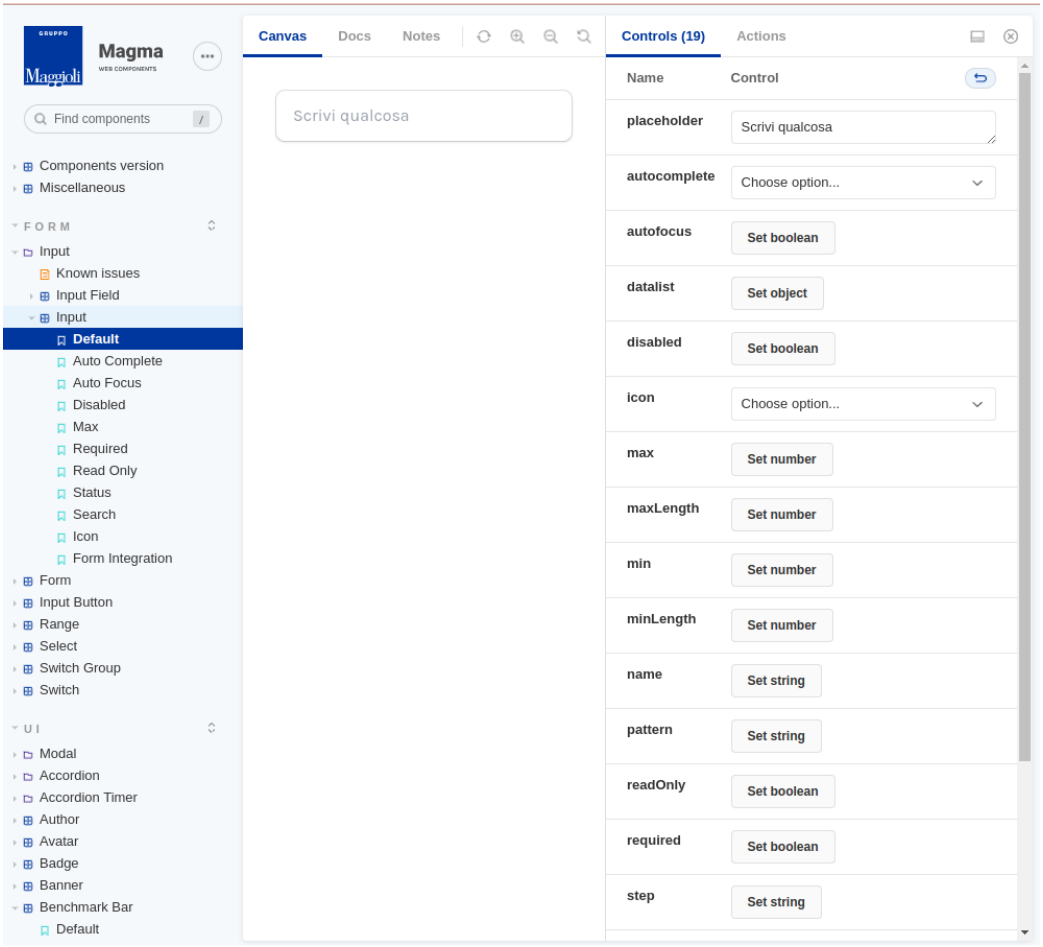


Figura 3.5: Visualizzazione del componente `mds-input` sull'istanza Storybook pubblica del *Design System* [3]

3.3 Testing

Per essere sicuri che i componenti Stencil sviluppati funzionino nella maniera che ci si aspetta, la libreria fornisce supporto per l'esecuzione di test *unitari* e *end-to-end*.

Esistono diverse filosofie su come il test di componenti debba essere fatto e come differenziare i casi in cui un test sia di tipo unitario e quando di tipo

end-to-end. In merito a tale argomento, Stencil propone una sua interpretazione delle due tipologie, in modo che il fardello non ricada sulle spalle degli sviluppatori, permettendo a questi ultimi di concentrarsi sullo sviluppo [23]:

- **Unit test**: si focalizza sul testare i metodi di un componente in ambiente isolato. Quando ad un metodo viene passato un argomento `X`, dovrebbe ritornare come risultato `Y`.

- **End-to-end test**: si focalizza su come il componente viene renderizzato nel DOM e come i singoli componenti interagiscono tra loro. Per esempio, quando il componente `my-component` possiede l'attributo `X`, il componente figlio renderizza il testo `Y` e il padre si aspetta di ricevere l'evento `Z`.

Entrambe le tipologie di test si basano sull'utilizzo dello strumento JavaScript *Jest*[13], mentre per i soli test end-to-end è possibile utilizzare come strumento *Puppeteer*[14] per eseguire i test dei componenti all'interno di un browser per fornire risultati più realistici.

Stencil permette agli sviluppatori di personalizzare la configurazione degli strumenti sopracitati, come ad esempio la cartella dove ricercare i file da testare. Permette inoltre di fare uso di *Visual Studio Code*[15] come strumento di debugger piuttosto che lanciare il comando direttamente da terminale.

In generale i test di regressione sono utilizzati per validare che le modifiche che sono state introdotte all'interno del codice durante lo sviluppo non abbiano un impatto inaspettato sul sistema.

Ma non è solo importante che il sistema continui a funzionare esattamente come prima (più le modifiche introdotte, ovviamente), ma è anche importante ciò che l'utente finale vede e con il quale interagisce. I test di regressione visuali hanno lo stesso scopo dei test di regressione classici, ma sono più

[13] `https://jestjs.io/`
[14] `https://pptr.dev/`
[15] `https://code.visualstudio.com/`

focalizzati sull'interfaccia presentata all'utente finale piuttosto che a quanto funzionale sia il sistema [36].

Per questo il team di sviluppo della suite di Web component ha trovato utile l'introduzione di uno strumento per effettuare test di regressione visuale come *Loki*[16], i cui obiettivi sono di fornire un'esperienza di setup facile, ridurre al minimo il costo di manutenzione aggiuntivo introdotto, effettuare test riproducibili e indipendenti dal sistema operativo, oltre a supportare tutte le piattaforme che Storybook supporta [37].

Tramite i comandi che la libreria mette a disposizione, è possibile creare dei *snapshot* delle viste dei vari componenti che verranno utilizzati per comparare lo stato dei componenti dopo una modifica. Se Loki riscontra delle regressioni il test fallisce e nuovi *snapshot* rappresentati le discrepanze tra il nuovo stato e quello precedente vengono salvati in un'apposita cartella, consultabile dal team di sviluppo che, a seguito di una review, dovrà decidere se accettare le modifiche visuali presenti (poiché pianificate) [38].

3.4 Modalità di distribuzione e rilascio

Per progetti di questo tipo, progetti *open* dove si sviluppano componenti la cui intenzione è quella di renderli disponibili a terzi, uno dei metodi più comuni esistenti per la condivisione del lavorato è la pubblicazione su una piattaforma di gestione di pacchetti.

Nel caso di Maggioli, i componenti realizzati vengono distribuiti sul registro pubblico reso disponibile da *npm* (Node Package Manager[17]), sotto l'organizzazione `maggioli-design-system`[18] gestita direttamente dai membri del core team responsabili dello sviluppo della suite.

Per poter essere utilizzati, tali componenti devono essere installati all'interno del proprio progetto tramite una delle seguenti modalità:

[16] `https://loki.js.org/`

[17] `https://docs.npmjs.com/about-npm`

[18] `https://www.npmjs.com/search?q=maggioli-design-system`

- *Installazione tramite gestore di pacchetti*: se si vuole installare un componente all'interno di un progetto *Node*, si può ricorrere all'utilizzo di un *package manager* come il sopracitato *npm* o ad altri package manager presenti in letteratura come *yarn*[19], *pnpm*[20] etc. Prendendo come esempio uno dei componenti sviluppati già rilasciati, il componente `MdsIcon`[21], l'installazione tramite terminale avviene tramite il comando `npm install @maggioli-design-system/mds-icon`.

- *Tramite supporto di una CDN*: come alternativa all'installazione tramite package manager, si può ricorrere all'utilizzo di una CDN[22], una rete distribuita per la consegna di contenuti come può essere *jsDeliver*[23]. Riprendendo l'esempio al punto precedente, per includere il componente `MdsIcon` all'interno del proprio progetto tramite link alla CDN basta includere, all'interno del proprio HTML, il seguente codice:

```
1  <body>
2          <script type="module">
3                  import { defineCustomElements } from
                   ↪  "https://cdn.jsdelivr.net/npm/@maggioli-design-system
                   ↪  /mds-icon@2.1.0/dist/esm/loader.js"
4                  defineCustomElements()
5          </script>
6
7          <h1>MdsIcon JsDeliver</h1>
8
9          <mds-icon name="https://clayto.com/icons/font-awesome/solid/carro
           ↪  t.svg"></mds-icon>
10 </body>
```

Listing 8: Import Web Component tramite CDN

[19] https://yarnpkg.com/
[20] https://pnpm.io/it/
[21] https://www.npmjs.com/package/@maggioli-design-system/mds-icon
[22] https://it.wikipedia.org/wiki/Content_Delivery_Network
[23] https://www.jsdelivr.com/

3.4.1 Pacchettizzazione dei componenti Stencil

Stencil permette di effettuare il bundle dei componenti sviluppati in due modalità differenti, in base a quale e quanti output target sono stati definiti nel suo file di configurazione [23]:

- *dist*: modalità di bundle classica, i componenti vengono generati come una libreria riutilizzabile che può autocaricarsi in maniera lazy. Il bundle così generato è treeshakable, ovvero solo i componenti effettivamente importati nel progetto finiranno nel codice del compilato. Questo approccio è consigliato se si intende supportare versioni meno aggiornate di Browser (come IE11).

- *dist-custom-element*: modalità di bundle per eseguire l'operazione di tree shaking in maniera ottimizzata. L'operazione di tree shaking è resa possibile dall'utilizzo, da parte di Stencil, dei ES Modules[24]. Questo approccio viene consigliato solo per utilizzo sui browser moderni che supportano già le funzionalità di Custom Elements, Shadow DOM e variabili CSS.

Il core team di sviluppo però non si è trovato soddisfatto con le modalità offerte da Stencil, non ha ritenuto performante imporre agli utilizzatori della suite di Web Component di dover scaricare tutti quanti i componenti quando alla fine ne utilizzerebbero solo una minima parte, per questo motivo risorse e particolare attenzione sono state destinate al trovare una soluzione che rispettasse i desideri del team.

Dopo diversi confronti avvenuti internamente al team, si è deciso di adottare un approccio poco ortodosso per quanto riguarda la pubblicazione dei componenti su *npm*. Tramite l'utilizzo di uno script Node personalizzato, quando giunge il momento di pubblicare la nuova versione (o la prima versione) di un componente della suite, questo viene *isolato* dagli altri all'interno di una cartella temporanea che non viene versionata, dove viene replicata la

[24] `https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Modules`

struttura di Stencil, facendolo diventare a tutti gli effetti un progetto Stencil dove è presente un solo componente.

Per supportare l'operazione di *isolamento* del componente, all'interno del repository Stencil sono presenti dei template dei file necessari per l'impostazione della struttura del codice, tra cui il file di configurazione di Stencil `stencil.config.ts`, il file necessario per la pubblicazione del pacchetto `package.json` e il file necessario per automatizzare i processi di CI/CD `.gilab-ci.yml` (che verrà approfondito nella sezione 3.6).

Una volta isolato, il componente viene compilato, vengono lanciati i test unitari e quelli end-to-end, viene fatto un test per assicurarsi che la versione del componente che si sta tentando di pubblicare non esista già. Se i passaggi precedenti sono andati a buon fine, il componente viene infine pubblicato tramite il package manager *npm*.

In questo modo, quando un componente verrà installato all'interno di un progetto, verrà installato solo il codice necessario per far funzionare correttamente quel singolo componente, più i componenti che sono stati inseriti all'interno della proprietà `dependencies` del file `package.json`.

Un altro vantaggio importante che porta con se tale approccio è quello di poter tenere separate le varie versioni dei componenti, senza dover aggiornare l'intera libreria ogni volta che viene apportata una modifica ad uno degli n componenti presenti. Nelle fasi iniziali del progetto si è tradotto in più lavoro di manutenzione, soprattutto per via delle interdipendenze che esistono tra i vari componenti, ma una volta stabilizzato lo sviluppo, grazie anche all'introduzione di strumenti e tecniche di automazione (sezione 3.6), l'esperienza di sviluppo e di pubblicazione si sono semplificate radicalmente.

3.5 Modello di contribuzione

Dato l'intento da parte dell'azienda di rendere il progetto di realizzazione dei Web Component come frutto di lavoro congiunto da diversi team interni, ma riservandosi la possibilità di far contribuire anche team esterni, si è reso

necessario effettuare delle modifiche al modello di contribuzione attuale utilizzato per il design system aziendale, al fine di ridurre i problemi causati da contribuzioni che arrivano da team variegati con regole di background diverse gli uni dagli altri.

Nello specifico si è deciso di passare da un modello dove lo sviluppo e la manutenzione del progetto sono affidate interamente al core team, ad un modello aperto alla community dove entra in gioco una nuova figura che ha la possibilità e capacità di apportare del valore aggiunto al progetto, sia esso relativo alla realizzazione di nuovi componenti o risoluzione di bug esistenti [39] [40].

Le figure presenti saranno quindi:

- **Core team member**: membro responsabile del progetto Design System, non ha più la sola responsabilità di sviluppare codice o progettare il design dei componenti, ma ha anche l'obiettivo di aiutare e rendere partecipe chi vuole contribuire al progetto.

- **Contributor member**: colui che contribuisce in modo attivo al progetto, proponendo iniziative, modifiche al codice o scrivendo articoli per il sito della documentazione del Design System (ancora in fase di sviluppo durante la stesura di questo documento).

Questo cambio nel modello di contribuzione, il cui nuovo flusso viene rappresentato in figura 3.6, serve a supportare tutte le persone che vogliono contribuire e proteggerle dalle insidie di sviluppo che si presentano in progetto così strutturati, in quest'ottica i membri del core team hanno sono chiamati ad impegnarsi per aiutare i contributor a renderli parte integrante del progetto.

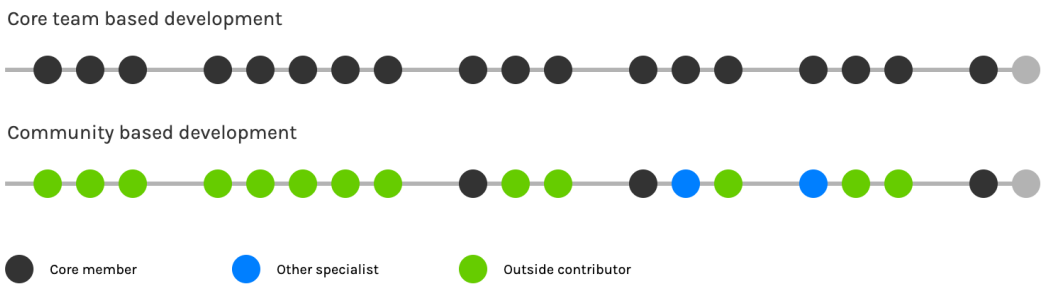


Figura 3.6: Il cambio del modello di contribuzione ha lo scopo di slegare lo sviluppo dei core member dal flusso, dando a tutti la possibilità di contribuire

3.5.1 Branching del repository

Per poter efficacemente perseverare il desiderio dell'azienda di abbracciare una cultura simile a quella dell'*open source*, il primo passo resosi necessario è stato ristrutturare il repository in modo che chiunque possa contribuire senza compromettere involontariamente il lavoro fatto in precedenza da qualcun altro. La nuova struttura adottata è la seguente:

- Il branch principale è il branch **main**, quello da cui partono le pubblicazioni dei vari pacchetti, le modifiche a questo branch devono essere apportare solo tramite apertura di una *merge request* provenienti dal branch **dev**
- Il branch di sviluppo è il branch **dev**, dove la push diretta di commit è consentita solo ai membri del *core team* mentre i *contributor* possono apportare le modifiche tramite apertura di una *merge request* da una **feature-branch**
- Per ogni contribuzione diversa, sia essa l'introduzione di una nuova feature o la risoluzione di un bug, è richiesta l'apertura di una **feature-branch** specifica

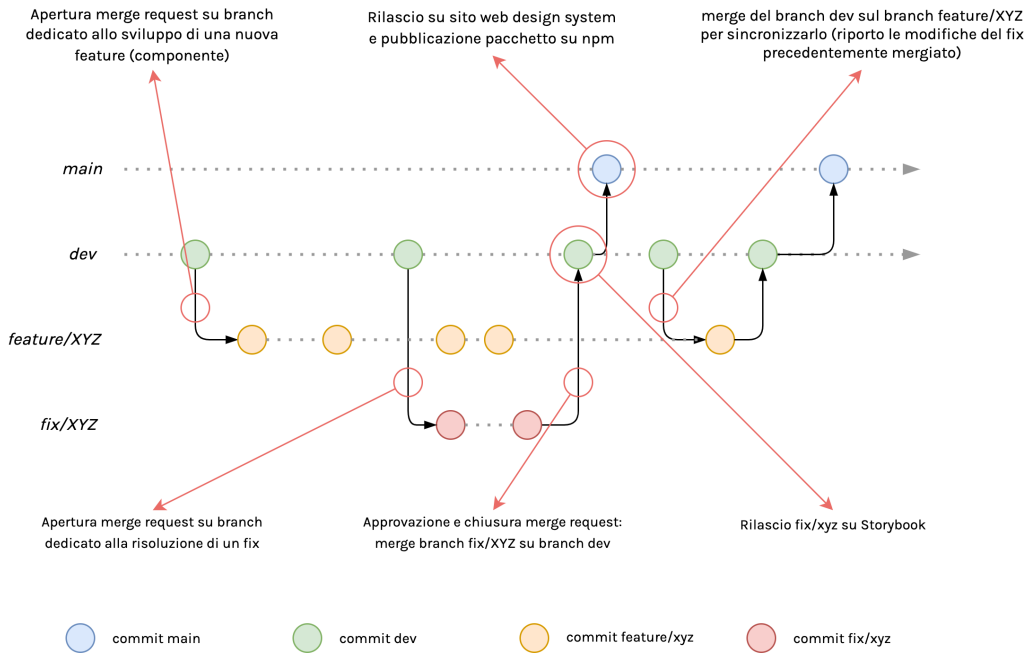


Figura 3.7: Rappresentazione del flusso di contribuzione tramite le branch del repository

3.5.2 Flusso di contribuzione

Successivamente alla ristrutturazione delle branch del repository, un altro passo di grande importanza per l'apertura alla contribuzione da parte della community è stato quello di redigere un chiaro flusso di contribuzione, che tutti i partecipanti al progetto sono tenuti a seguire, per poter più facilmente gestire gli interventi di sviluppo apportati al Design System tramite l'adozione di linee guida e standard qualitativi [41].

Verranno ora elencati i vari tipi di contribuzione che il core team ha individuato come scenari possibili e gli step (e i relativi sotto passaggi) richiesti al fine di generare una corretta richiesta di contribuzione.

Non tutti gli step che verranno elencati saranno necessari per ogni tipologia di contribuzione.

Tipologie di contribuzione

- **Bug Fix**: il tipo di intervento meno impattante, si intende la correzione di un difetto legato al funzionamento inatteso di qualche componente o pacchetto node pubblicato.

- **Small enhancement**: consiste nell'aggiungere una piccola funzionalità ad un componente, come l'aggiunta di un colore ad una label o qualunque cosa che non impatti sull'architettura di un componente.

- **Large enhancement**: può riguardare l'aggiunta di una qualche funzionalità che non si limita ad estendere una funzionalità già presente, ma richiede un'integrazione più profonda o un'operazione di refactoring di un componente specifico.

- **New feature**: relativo alla creazione di un nuovo componente o di altro elemento che aggiunge funzionalità al Design System.

- **New feature across components**: quando introduce una nuova funzionalità che ha un impatto a livello architettonico su più componenti, come potrebbe essere l'aggiunta di un nuovo attributo ad una determinata categoria di componenti (*e.g.* un nuovo attributo ai componenti form).

- **Major initiative**: relativamente all'implementazione o al cambio di architettura di altre parti del Design System, come può essere il cambio di stil edi tutti i componenti o la modifica dei design token.

Step

- **Propose**: passaggio iniziale, dove si propongono le ragione dell'intervento che si vuole apportare. La proposta serve come strumento per gestire l'intervento e capire se va mantenuto nel flusso, scartato o indirizzato diversamente.

1. *Fill a proposal template*: compilazione di un form dedicato alla proposta tramite lo strumento messo a disposizione dal core team. Deve rispondere alle domande *Che cosa? Come mai? Per chi/quanti? Entro quando?*

2. *Check across current experiences*: prima di procedere alla pubblicazione della proposta è necessario verificare che non sia già presenti ticket legati allo stesso tipo di intervento.

3. *Draft requirements*: preparazione di una bozza sui requisiti necessari a poter sviluppare la lavorazione. Per semplici componenti dell'interfaccia utente possono essere sufficienti elenchi di *Deve*, *Dovrebbe*, *Potrebbe* etc. La proposta di un layout complesso può richiedere pagine e includere valori proposti e demo di codice. Questi dettagli portano alla definizione delle priorità delle funzionalità durante le revisioni e le presentazioni.

4. *Review by system team / product owner*: una volta avviata la proposta, questa viene verificata dal core team ed eventualmente dal product owner, adatto a verificare più da vicino il motivo della proposta, se pensata per un prodotto specifico.

5. *Present to system team / community*: superata la fase di review, la proposta viene pubblicata all'interno della community dove tutti possono controllarla pubblicamente. Qualsiasi comportamento inappropriato da parte dei membri della community sarà tempestivamente gestito da uno dei membri del core team.

6. *Finalize scope*: fase di definizione dell'intervento, dove si deve decidere come gestire la lavorazione e classificandolo secondo una delle tipologie sopra descritte 3.5.2.

7. *Plan subsequent tasks*: pianificare le attività successive facendo uso degli strumenti messi a disposizione (come visto nella sezione 3.2.2)

- **Design**: se previsto dalla lavorazione, è necessario realizzare alcuni elementi di design per illustrare agli altri membri come sarà la lavorazione a livello di UI e UX. In questa macro-fase il contributor può chiedere aiuto al designer del core team se non si sente confidente nella realizzazione.

    1. *Setup from file template*: creare un file su Figma (sezione 3.2.3) e gestirne la presentazione con stili e componenti già serviti dal Design System su Figma: asset come Color palette, Typography, Identity e Web Component.
    2. *Integrate existing assets*: se la lavorazione estende qualcosa di già fatto in precedenza, estendere ciò che è stato fatto con nuove viste che comprendono le nuove lavorazioni.
    3. *Design features*: le funzionalità devono essere rappresentate in modo da dare un punto di riferimento e abilitare i programmatori allo sviluppo.
    4. *Align style with design tokens*: il Design System condivide su Figma gli asset necessari per mantenere il più possibile allineati i componenti in sviluppo con i contributi di Figma.
    5. *Annotate design*: se necessario, aggiungere note che possano aiutare chi visualizza i componenti o le modifiche a capire di cosa si tratta e come funzionano.
    6. *Test accessibility*: controllare che le modifiche fatte passino i test di accessibilità in modo da creare materiale che rispetti gli standard definiti dal Design System.
    7. *Test usability*: come per i test di accessibilità, controllare che i pattern di user experience utilizzati siano aggiornati con gli standard attuali.
    8. *Present to assigned developer*: per assicurarsi che il lavoro sia corretto anche da un punto di vista funzionale, è necessario pre-

sentare il design fatto da uno sviluppatore del core team o altro contributor, affinché sia validato per essere lavorabile per il codice.

9. *Present to system team*: se non già presente nella fase precedente, un membro del core team dovrebbe avere la supervisione del lavoro svolto fino a questo punto.
10. *Present to design community*: l'ideale è che il lavoro svolto sia visibile anche dalla micro community interna del design system, qualsiasi occhio che può portare un miglioramento al lavoro è ben accetto.
11. *Review by system developer*: conclusa la fase di presentazione, si passa alla fase di review da parte degli sviluppatori del core team.
12. *Review by content strategist*: tra i reviewer è molto utile la presenza di un content strategist che possa valutare che il design sia conforme anche da questo punto di vista.
13. *Review by accessibility expert*: dal momento che il design deve rispettare gli standard qualitativi in termini di accessibilità, è richiesta la review anche da parte di un esperto dell'argomento.
14. *Finalize and deliver*: un designer garantisce che la grafica e le annotazioni siano complete, i token integrati e i dettagli espansi prima del trasferimento dell'incarico allo sviluppatore incaricato.

- **Code**: rappresenta la parte di sviluppo del codice e i flussi legati a questa parte del processo.

1. *Create code/repo branch*: il primo passo che è necessario fare è creare un branch legato al tipo di lavorazione che viene fatta.
2. *Generate files*: preparare la struttura dei file necessari ad introdurre la funzionalità, fix o il cambiamento che si vuole apportare.
3. *Source reusable code*: prima di effettuare una commit e una push sul branch remoto, il codice scritto dovrebbe arrivare fino ad un punto tale che sia funzionante ed eseguibile.

4. *Draft API*: preparare una API che rispetti gli standard definiti dal Deisgn System.
5. *Code HTML, CSS, JS, etc.*: scrivere il codice HTML, CSS, JavaScript o Typescript e il resto dei formati necessari. La sezione legata ai Web Component ha un comando che ne assiste la creazione.
6. *Create visual testing page view*: viene richiesto di aggiungere una vista che consenta di effettuare visual testing del lavoro svolto. Attualmente il Design System permette di aggiungere viste su Storybook per quanto riguarda la lavorazione dei Web Component.
7. *Document API*: è necessario decorare il codice della lavorazione con della documentazione che spieghi il comportamento delle API che sono state implementate.
8. *Update dependents*: se necessario, aggiornare le dipendenze del progetto nel file `package.json` dei singoli componenti coinvolti nella lavorazione.
9. *Compose, run unit tests*: per ogni componente viene richiesta, qualora sia possibile, la creazione di test unitari.
10. *Test accessibility (manual)*: aggiungere test di accessibilità manuali.
11. *Test accessibility (automated)*: aggiungere test di accessibilità automatici.
12. *Test browsers and devices*: controllare che il codice scritto rispetti il comportamento previsto sui browser principali (Chrome, Firefox e Safari).
13. *Test visual regression*: aggiungere test di regressione visuale.
14. *Test functionality*: aggiungere test funzionali in modo da verificare che le API create siano effettivamente funzionanti in fase di runtime.

15. *Present to assigned designer*: una volta che il componente è funzionante, un designer del core team o un designer assegnato in precedenza dovrebbe verificare che il design sia consistente con lo step **Design** descritto in precedenza, o che sia costruito con una serie di compromessi ragionevoli.

16. *Present to system team*: se non già presente nello step precedente, uno o più membri del core team dovrebbero verificare che il lavoro svolto sia conforme agli standard definiti dal design system.

17. *Present to developer community*: il lavoro dovrebbe essere esposto alla micro community interna aziendale.

18. *Review by system designer*: un designer del core team è assegnato per fare la review del lavoro svolto e confermare che tutto sia pronto dal punto di vista della user interface.

19. *Review by system developer*: uno sviluppatore del core team è assegnato per fare la review del lavoro svolto e confermare che tutto sia pronto dal punto di vista del coding standard e sulla qualità generale del codice.

20. *Review by accessibility expert*: un esperto di accessibilità è assegnato per fare la review dell'accessibilità del componente.

21. *Write changelog*: presentare un changelog nel quale sono elencate le modifiche effettuate.

22. *Finalize and merge code*: una volta che il lavoro è finalizzato, aprire una *merge request* dal branch di lavoro della funzionalità verso il branch **dev**, in modo che il codice sia aggiunto alla codebase principale.

- **Doc**: questo è lo step finale per quanto riguarda lo sviluppo, dove si inseriscono le informazioni che vanno rese pubbliche a tutti gli utilizzatori del Design System.

1. *Setup file from template*: scrivere la documentazione relativa al lavoro svolto seguendo un template già esistente.
2. *Source reusable doc, images*: fornire documentazione riutilizzabile, eventualmente con immagini.
3. *Compose intro*: scrivere una introduzione al componente e alla sua funzione.
4. *Extend previous documentation*: se necessario, ampliare la documentazione già esistente di un componente relativamente al lavoro fatto.
5. *Compose examples*: aggiungere esempi di utilizzo che aiutino a capire come sfruttare le funzionalità del componente.
6. *Compose design guidelines*: descrivere come il risultato raggiunto può essere sfruttato a livello di design.
7. *Compose accessibility guidelines*: aggiungere eventuali note sul cosa fare e cosa non fare per evitare problemi di accessibilità.
8. *Compose content guidelines*: descrivere come scrivere contenuti all'interno dell'elemento.
9. *Update design files*: aggiornare i file su Figma ai quali sono state apportate le modifiche o le nuove funzionalità.
10. *Produce static images*: preparare immagini statiche che rappresentino il componente e le sue funzionalità principali.
11. *Review by system developer*: uno sviluppatore del core team è assegnato per eseguire la review della documentazione.
12. *Review by system designer*: un designer del core team è assegnato per eseguire la review della documentazione.
13. *Review by accessibility specialist*: uno specialista dell'accessibilità è assegnato per eseguire la review della documentazione.
14. *Review by system editor*: un redattore del sito della documentazione è assegnato per eseguire la review della documentazione.

15. *Finalize content*: finalizzare la documentazione scritta e assicurarsi che sia tutto a posto.
16. *Package for publishing*: pubblicare il materiale avviandone il processo di rilascio.

- **Release**: fase di rilascio del lavoro svolto (sezione 3.4).

3.6 Automazione

Con il termine *DevOps* ci si riferisce ad una metodologia di sviluppo software che punta alla comunicazione, collaborazione e integrazione tra gli sviluppatori e gli addetti alle *operations*[25], aiutando le aziende nella gestione dei rilasci e standardizzando gli ambienti di sviluppo.

Alla base di tale metodologia ci sono le pratiche conosciute come **CI/CD**:

- *Continous Integration (CI)*: integrazione del lavoro svolto dagli sviluppatori in maniera frequente, scoraggiandone la persistenza sulla sola macchina locale dove le modifiche vengono eseguite, seguita da verifica automatica dell'integrità per rilevare l'insorgere di errori il più rapidamente possibile.
- *Continous Delivery/Deployment (CD)*: successiva alla fase di CI, si intendono tutte le operazioni necessarie per rilasciare nuove versioni del software in modo semplice e ripetibile.

Il prefisso *Continous* sta ad indicare che tali operazioni vengono svolte in maniera automatica tramite l'utilizzo di specifici tool allo scatenarsi di uno specifico evento, il più delle volte la push di una commit sulla piattaforma di versionamento del codice.

Maggioli da anni ha abbracciato la cultura *DevOps*, introducendo pratiche di CI/CD all'interno dei suoi prodotti sviluppati all'interno del reparto

[25] `https://it.wikipedia.org/wiki/Operations`

Ricerca&Sviluppo, per poi allargare tale integrazione anche agli altri prodotti e software sviluppati nel gruppo. Ad oggi sono ormai pochi i prodotti che non adottano una qualche metodologia DevOps.

Anche il progetto *Design System* adotta tali pratiche per quanto riguarda la verifica di integrità delle modifiche apportate con ogni nuova commit che viene sincronizzata con il server e per il rilascio di nuove versioni dei sotto-progetti che lo compongono, incluso anche il deploy della nuova versione dello Storybook dei componenti sviluppati con Stencil.

Particolare cura è stata concessa al sotto-progetto dei Web Component dove, come già citato nella sezione 3.4.1, è stato adottato un metodo poco ortodosso per poter pubblicare ogni componente della suite come pacchetto singolo.

Per automatizzare i passaggi descritti in tale sezione, il cui compito è stato affidato allo strumento *pipeline* messo a disposizione da GitLab [42], è stato predisposto un template del file `.giltab-ci.yml` come mostrato nel listato 9, dove vengono definiti tutti gli step che il processo di automazione dovrà eseguire per ogni componente.

Questo template viene utilizzato come base per poter generare lo specifico file relativo ad ogni singolo componente, processo reso più semplice per lo sviluppatore grazie all'introduzione di uno script all'interno del file *package.json* del progetto Stencil, che crea una copia di tale file per ogni componente presente nella cartella `src/components`, andando a sostituire il nome del componente con il placeholder {{ `componentName` }}.

```
.base-{{ componentName }}:
  variables:
    COMPONENT: mds-{{ componentName }}

# ISOLATE
{{ componentName }}-isolate:
  extends: [.base-isolate, .base-{{ componentName }}]
  dependencies: [base-stencil-build]

# TEST
{{ componentName }}-publish-test:
  extends: [.base-stencil-publish-test, .base-{{ componentName }}]
  dependencies: [base-stencil-build, {{ componentName }}-isolate]

# PUBLISH
{{ componentName }}-publish:
  extends: [.base-stencil-publish, .base-{{ componentName }}]
  dependencies: [base-stencil-build, {{ componentName }}-isolate]
  needs: [base-stencil-build, {{ componentName }}-isolate, {{
  ↪  componentName }}-publish-test]

# INSTALL TEST
{{ componentName }}-install-test:
  extends: [.base-stencil-install-test, .base-{{ componentName }}]
  dependencies: [base-stencil-build, {{ componentName }}-isolate]
  needs: [base-stencil-build, {{ componentName }}-isolate, {{
  ↪  componentName }}-publish]
```

Listing 9: Template del file *.gitlab-ci.yml* generato per ogni componente

Tali operazioni vengono eseguite ogni qual volta viene fatto il push del lavoro su qualsiasi branch o merge request, ma limitati ai soli componenti che sono stati modificati dalle commit interessate dalla push.

È da notare l'utilizzo congiunto delle proprietà `dependencies` e `needs` nella definizione degli step, la prima per indicare da quali job precedenti il job attuale deve recuperare gli artefatti da utilizzare, la seconda per indicare

quali job precedenti devono essere completati per permettere l'esecuzione del job attuale. La combinazione di queste due proprietà evita che il fallimento delle operazioni relative ad uno o più componenti si ripercuotano su tutta la pipeline, che invece può continuare l'esecuzione dei job per i componenti che non hanno riscontrato problemi, procedendo fino anche alla pubblicazione di quest'ultimi su *npm*.

Capitolo 4

Analisi qualitativa del progetto

In questo capitolo viene descritta la metodologia adottata nel somministrare il questionario agli sviluppatori che hanno avuto modo di integrare il lavoro svolto per la creazione di una suite di Web Component nel design system aziendale all'interno dei loro progetti, analizzando poi i risultati ottenuti.

Il caso di studio termina con la retrospettiva rispetto alla realizzazione del progetto, accompagnata dalle principali problematiche incontrate durante lo sviluppo e le considerazioni finali.

4.1 Questionario

Per ottenere un feedback da parte dei colleghi e degli altri reparti sull'utilizzo della suite di componenti all'interno dei propri progetti, alla fine della prima fase di sviluppo e dopo aver rilasciato la prima versione ufficiale della suite, è stato deciso di somministrare un questionario composto da un ristretto numero di domande.

Lo scopo è quello di valutare la buona riuscita del progetto in base alle emozioni e sensazioni, oltre che l'effettiva utilità, che provano gli sviluppatori nell'utilizzare i componenti messi a disposizione piuttosto che crearne di propri in base al framework o piattaforma di utilizzo.

In questa sezione verrà presentata la tipologia di questionario somministrato e i risultati ottenuti.

4.1.1 Metodologia

La tipologia di questionario utilizzata è lo **User Experience Questionnaire**[1] (`UEQ`), che ha lo scopo di misurare l'esperienza percepita dagli utenti di prodotti interattivi.

La prima versione risale al 2005 creata da Andreas Hinderks, Martin Schrepp e Jörg Thomaschewski, tutti e tre con competenze in ambito di user experience, progettazione di interfacce grafiche e human computer interaction. Per la sua creazione è stato adottato un approccio analitico dei dati per poter assicurare una rilevanza pratica delle aree individuate a partire da un grande numero di elementi [43].

Composto da 6 distinte scale, ognuna delle quali descrive un distinto aspetto qualitativo di un prodotto, l'attuale versione dell'UEQ conta 26 elementi che hanno la forma di un differenziale semantico[2], quindi ogni elemento è rappresentato da due termini con significato opposto. L'ordine dei termini è randomizzato per elemento, questo sta a significare che metà degli elementi di un'area iniziano con il termine positivo e l'altra metà degli elementi inizia con il termine negativo. Viene utilizzata una scala di valutazione a 7 posizioni.

Un esempio di elemento è il seguente:

`attraente` ◯ ◯ ◯ ◯ ◯ ◯ ◯ `non attraente`

Gli elementi sono dimensionati in una scala da -3 a +3, dove -3 indica la risposta maggiormente negativa, 0 una risposta neutrale e +3 la risposta maggiormente positiva.

La consistenza e la validità delle scale utilizzate è stata investigata in 11 test di usabilità con 144 partecipanti dal vivo e con oltre 700 partecipanti

[1] `https://www.ueq-online.org/`

[2] `https://it.wikipedia.org/wiki/Differenziale_semantico`

da un questionario online. I risultati di questi studi hanno dimostrato una consistenza sufficiente e una buona validità delle scale costruite [44].

Composizione

Come scritto poco sopra, i 26 elementi che compongono il questionario sono suddivisi in 6 diverse scale [43]:

- *Attrattività*: Impressioni generali del prodotto. Gli utenti gradiscono o meno il prodotto?
- *Apprendibilità*: È facile familiarizzare col prodotto? È facile imparare ad usare il prodotto?
- *Efficienza*: Gli utenti riescono a portare a termine i loro compiti senza sforzi inutili?
- *Controllabilità*: L'utente sente di avere il controllo dell'interazione?
- *Stimolazione*: L'utente prova motivazione ed emozionante durante l'uso del prodotto?
- *Originalità*: Il prodotto è innovativo e creativo? Riesce a catturare l'interesse dell'utente?

L'*Attrattività* è una dimensione di pura valenza ed è l'unica scala che è composta da 6 elementi, mentre tutte le altre scale sono composte da 4 elementi ciascuno, come mostrato in figura 4.1. *Apprendibilità*, *Efficienza* e *Controllabilità* sono aspetti di qualità pragmatici, mentre *Stimolazione* e *Originalità* sono aspetti di qualità edoniche.

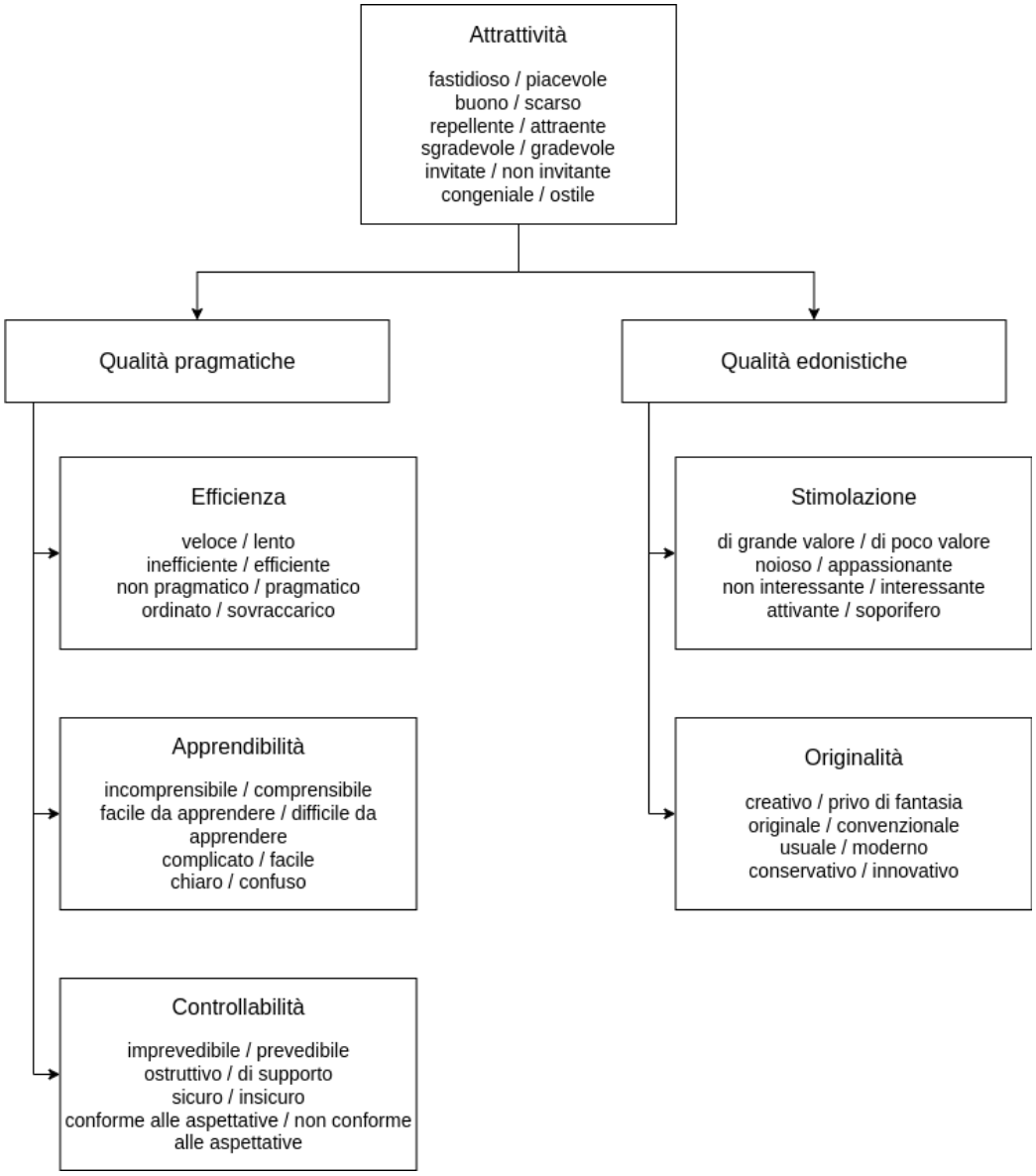


Figura 4.1: Le 6 scale del questionario UEQ

Scenari d'utilizzo

Tipici scenari in cui si può utilizzare questa tipologia di questionario sono [43]:

1. *Confrontare l'esperienza utente di due prodotti*: i prodotti in questione potrebbero riguardare una versione attualmente utilizzata e una nuova versione ridisegnata, oppure potrebbero essere prodotti di competitor diretti nel mercato in cui opera l'azienda. La comparazione dei prodotti può essere banalmente fatta sulla misura dei valori ottenuti dai questionari.

2. *Controllare che un prodotto abbia una buona esperienza utente*: il prodotto soddisfa le aspettative generali che riguardano la user experience? La risposta a questa domanda a volte è immediatamente chiara osservando il valore medio delle 6 scale, ma per avere un quadro migliore della qualità è anche necessario confrontare i valori ottenuti con i risultati di altri prodotti già affermati. Per questo UEQ offre un data set di oltre 400 valutazioni di prodotti (valutati con il questionario UEQ) aggiornato ogni anno con i quali confrontare i risultati ottenuti sul proprio prodotto.

3. *Come parte di un test di usabilità*: i dati raccolti vengono utilizzati per avere un feedback sulle impressioni che i partecipanti al questionario hanno in merito all'esperienza utente offerta dal prodotto in esame. Il momento migliore per somministrarlo è subito dopo che sono stati completati i compiti del test, in modo che il giudizio del partecipante non sia contaminato dal confronto con chi ha condotto il test.

4. *Determinare aree di miglioramento*: il questionario non può di certo individuare quali siano gli aspetti puntuali da modificare per migliorare la user experience del prodotto, ma può dare un'idea di quale che siano le aree dove i miglioramenti avranno un maggior impatto.

Un aspetto che non è possibile ottenere data la natura del questionario è un valore di KPI per la user experience, valore molto amato dai manager poiché riduce la complessità dei risultati ad un singolo valore numerico, dando l'impressione che le cose siano controllabili e migliorabili misurando e ottimizzando questo singolo valore.

Per poter ottenere un valore simile dal questionario UEQ è necessario aggiungere altre 6 domande (dopo i 26 elementi originali) in merito alle diverse scale che compongono il questionario, valutate tramite una scala di Likert a 7 valori che va da *Per niente importante (1)* a *Molto importante (7)*.

Strumenti di analisi

UEQ mette a disposizione degli utilizzatori del questionario anche uno strumento per facilitarne l'analisi dei risultati, un file Excel disponibile sempre dal loro sito [43].

Il documento Excel è composto dai seguenti fogli di lavoro:

- *Data*: unico foglio che necessita di input, qui vanno inserite le risposte ai questionari, una riga per ogni partecipante, nella scala di valori da 1 a 7 così come sono stati raccolti.

- *DT*: i dati presenti nel foglio precedente vengono trasformati dalla scala da 1 a 7 in una scala da -3 a +3, trasformazione necessaria per eseguire le dovute valutazioni nei fogli successivi.

- *Results*: vengono calcolati i valori medi per ogni scala, oltre che alla deviazione media e alla deviazione standard per ogni elemento. Già in questo foglio è possibile effettuare una prima valutazione dell'esperienza utente del prodotto.

- *Confidence_Intervals*: vengono calcolati gli intervalli di confidenza per le medie delle scale e le medie di ogni singolo elementi.

- *Answers_Distributions*: mostra la distribuzione delle risposte per ogni elemento. Può essere utile per notare aspetti del prodotto che sono particolarmente apprezzati (in caso di risposte positive) o meno (in caso di risposte negative).

- *Scale_Consistency*: vengono calcolati l'alpha di Cronbach[3] e il coefficiente Lambda2 di Guttman[4], utilizzati per stimare l'affidabilità di una scala.

- *Benchmark*: confronta i risultati ottenuti sul prodotto in esame con un data set di valori di altri prodotti per avere una stima più significativa dei risultati.

- *Inconsistencies*: qui vengono individuate le risposte `inconsistenti`, ovvero risposte date casualmente o parzialmente da qualche partecipante, che possono essere poi rimosse dal foglio dei dati per ottenere un'analisi meno contaminata.

- *Sample_Size*: aiuta a capire quante risposte bisognerebbe ottenere per avere un risultato più significativo dei dati.

- *KPI_Calculation*: permette di calcolare il valore di KPI in base alle 6 domande addizionali aggiunte al questionario.

Ad oggi il questionario, diventato molto famoso, è disponibile in più di 30 lingue diverse, grazie al lavoro da parte della community di internet.

4.1.2 Dati raccolti

In questa sezione si analizzeranno le risposte al questionario facendo uso degli strumenti messi a disposizione dall'UEQ, come già descritto nella sezione 4.1.1.

Risultati

In questa prima sezione vengono calcolati i valori medi e la varianza sia per ogni elemento (figura 4.2) del questionario che per ogni scala (figura 4.4),

[3] `https://it.wikipedia.org/wiki/Alpha_di_Cronbach`

[4] `https://www.statisticshowto.com/guttmans-lambda-2/`

inoltre per ogni elemento viene calcolata anche la deviazione standard (figura 4.2).

Grazie alla rappresentazione grafica in figura 4.3, possiamo notare come la quasi totalità degli elementi abbia un *valore medio* positivo, indicando sin da subito un buon riscontro del prodotto da parte della popolazione che ha compilato il questionario. Ciò è ulteriormente convalidato da una grande presenza di valori bassi per quanto riguarda la *deviazione standard*, indicando che le risposte ottenute per ogni elemento sono per la maggior parte omogenee.

Item	Mean	Variance	Std. Dev.	No.	Left	Right	Scale
1	1,0	0,5	0,7	5	incresioso	gradito	Attrattività
2	1,6	0,3	0,5	5	incomprensibile	comprensibile	Apprendibilità
3	1,6	0,3	0,5	5	creativo	privo di fantasia	Originalità
4	1,2	0,7	0,8	5	facile da capire	difficile da capire	Apprendibilità
5	2,2	0,2	0,4	5	di grande valore	di poco valore	Stimolazione
6	1,4	0,3	0,5	5	noioso	appassionante	Stimolazione
7	1,4	0,3	0,5	5	non interessante	interessante	Stimolazione
8	-0,4	1,3	1,1	5	imprevedibile	prevedibile	Controllabilità
9	1,2	1,7	1,3	5	veloce	lento	Efficienza
10	1,4	0,3	0,5	5	originale	convenzionale	Originalità
11	1,4	0,3	0,5	5	ostruttiva	di supporto	Controllabilità
12	1,8	0,2	0,4	5	bene	male	Attrattività
13	0,8	0,7	0,8	5	complicato	facile	Apprendibilità
14	1,8	0,7	0,8	5	repellente	attraente	Attrattività
15	1,6	0,3	0,5	5	usuale	moderno	Originalità
16	1,6	0,3	0,5	5	sgradevole	piacevole	Attrattività
17	0,8	1,7	1,3	5	sicuro	insicuro	Controllabilità
18	1,2	0,7	0,8	5	attivante	soporifero	Stimolazione
19	2,0	0,5	0,7	5	aspettativo	non aspettativo	Controllabilità
20	2,4	0,3	0,5	5	inefficiente	efficiente	Efficienza
21	1,6	0,3	0,5	5	chiaro	confuso	Apprendibilità
22	1,4	1,3	1,1	5	non pragmatico	pragmatico	Efficienza
23	1,8	0,7	0,8	5	ordinato	sovraccarico	Efficienza
24	1,4	0,3	0,5	5	attrattivo	non attrattivo	Attrattività
25	0,6	1,3	1,1	5	simpatico	antipatico	Attrattività
26	1,4	2,3	1,5	5	conservativo	innovativo	Originalità

Figura 4.2: Valore medio e varianza per ogni elemento

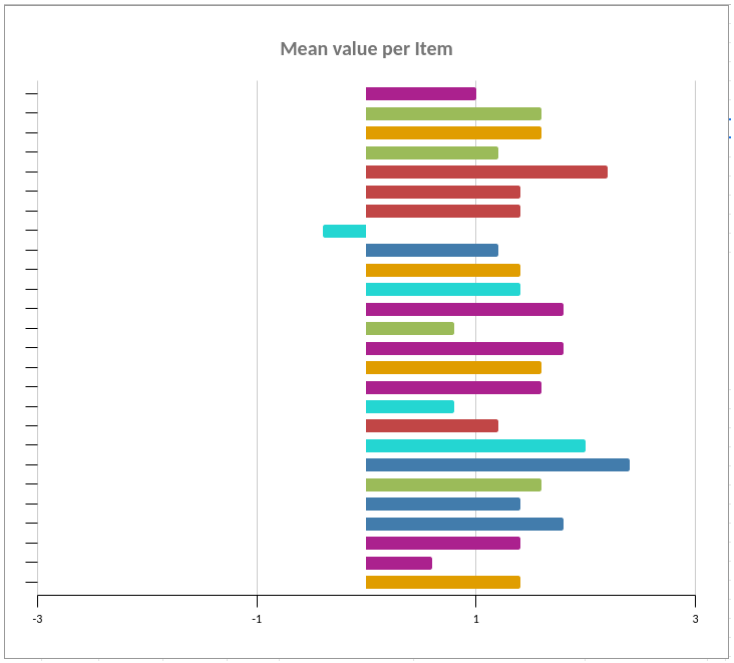


Figura 4.3: Rappresentazione grafica valore medio per ogni elemento

Unico neo risultante da questa prima valutazione risulta essere una propensione nel considerare la suite di Web Component più *imprevedibile* che *prevedibile*. Si renderà necessario indagare maggiormente la motivazione di questa sensazione alquanto comune, per poter cercare di limitare l'imprevedibilità del prodotto, affrontando probabilmente tale discorso con la community.

In figura 4.4 sono invece mostrati i *valori medi* e la *varianza* calcolata per ogni scala, quindi considerando i valori aggregati piuttosto che i valori singoli.

Tenendo presente che la scala di valutazione oscilla tra il valore negativo `-3` e il valore positivo `+3`, ci si può ritenere abbastanza soddisfatto del risultato aggregato ottenuto.

Anche in questo caso l'unico valore che stona un po' di più rispetto agli altri è quello della *Controllabilità*, quindi gli utilizzatori della suite non sente di avere totalmente il controllo durante l'introduzione dei componenti all'in-

terno dei loro progetti, rispecchiando l'opinione di *imprevedibilità* osservata negli elementi singoli.

UEQ Scales (Mean and Variance)		
Attrattività	1,367	0,02
Apprendibilità	1,300	0,04
Efficienza	1,700	0,04
Controllabilità	0,950	0,11
Stimolazione	1,550	0,04
Originalità	1,500	0,09

Figura 4.4: Valore medio e varianza per ogni scala

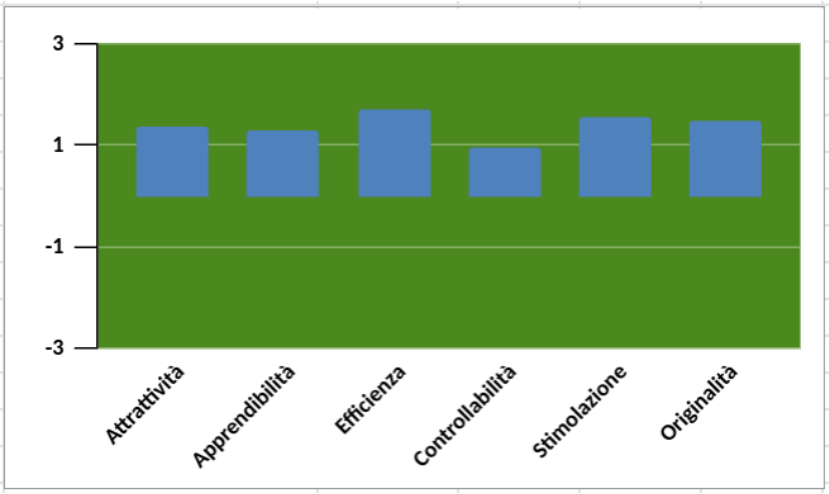


Figura 4.5: Rappresentazione grafica del valore medio di ogni scala su una scala di valori da -3 a +3

In figura 4.5 viene rappresentato il *valore medio* di ogni scala in base alla scala di valutazione da `-3` a `+3`, mentre in figura 4.6 viene fornita la stessa rappresentazione ma su una scala di valutazione da `-2` a `+2`, la quale viene consigliata rispetto alla prima poiché è raro ottenere valori medi che

vadano oltre a -2 e +2, quindi la seconda rappresentazione viene preferita per accentuare la bontà dei valori ottenuti (come in questo caso).

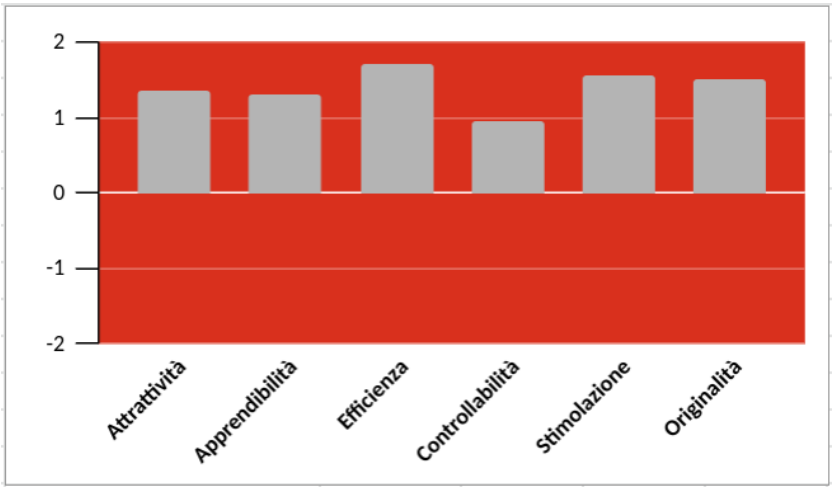


Figura 4.6: Rappresentazione grafica del valore medio di ogni scala su una scala di valori da -2 a +2

In figura 4.7 e 4.8 vengono riportati e rappresentati i *valori medi* della scala *Attrattività* e delle altre scale raggruppate nelle categorie delle qualità *Pragmatica* ed *Edonica*.

Pragmatic and Hedonic Quality	
Attrattività	1,37
Qualità Pragmatica	1,32
Qualità edonica	1,53

Figura 4.7: Media del valore di Attrattività e valori delle qualità Pragmatiche ed Edoniche

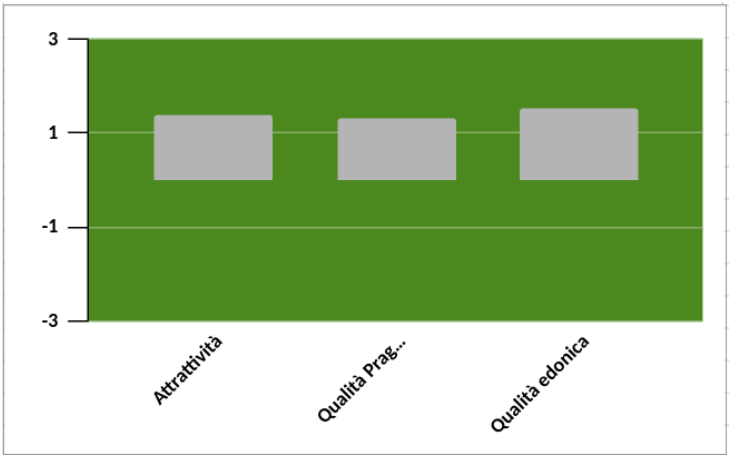


Figura 4.8: Rappresentazione grafica della media del valore di Attrattività e valori quelle qualità Pragmatiche ed Edoniche

Intervalli di confidenza

In questa sezione vengono calcolati gli *intervalli di confidenza* per i *valori medi* di ogni elemento (figura 4.10) e per ogni scala (figura 4.9).

Come è possibile notare, i valori degli intervalli per i singoli elementi sono abbastanza alti, indicando che i risultati ottenuti non sono abbastanza affidabili per poter trarre conclusioni molto affidabili, fatto dovuto con molta probabilità al basso numero di persone a cui è stato sottoposto il questionario.

Confidence intervals (p=0.05) per scale						
Scale	**Mean**	**Std. Dev.**	**N**	**Confidence**	**Confidence interval**	
Attrattività	1,367	0,139	5	0,122	1,244	1,489
Apprendibilità	1,300	0,209	5	0,183	1,117	1,483
Efficienza	1,700	0,209	5	0,183	1,517	1,883
Controllabilità	0,950	0,326	5	0,286	0,664	1,236
Stimolazione	1,550	0,209	5	0,183	1,367	1,733
Originalità	1,500	0,306	5	0,268	1,232	1,768

Figura 4.9: Intervalli di confidenza per ogni scala

Confidence interval (p=0.05) per item						
Item	Mean	Std. Dev.	N	Confidence	Confidence interval	
1	1,000	0,707	5	0,620	0,380	1,620
2	1,600	0,548	5	0,480	1,120	2,080
3	1,600	0,548	5	0,480	1,120	2,080
4	1,200	0,837	5	0,733	0,467	1,933
5	2,200	0,447	5	0,392	1,808	2,592
6	1,400	0,548	5	0,480	0,920	1,880
7	1,400	0,548	5	0,480	0,920	1,880
8	-0,400	1,140	5	0,999	-1,399	0,599
9	1,200	1,304	5	1,143	0,057	2,343
10	1,400	0,548	5	0,480	0,920	1,880
11	1,400	0,548	5	0,480	0,920	1,880
12	1,800	0,447	5	0,392	1,408	2,192
13	0,800	0,837	5	0,733	0,067	1,533
14	1,800	0,837	5	0,733	1,067	2,533
15	1,600	0,548	5	0,480	1,120	2,080
16	1,600	0,548	5	0,480	1,120	2,080
17	0,800	1,304	5	1,143	-0,343	1,943
18	1,200	0,837	5	0,733	0,467	1,933
19	2,000	0,707	5	0,620	1,380	2,620
20	2,400	0,548	5	0,480	1,920	2,880
21	1,600	0,548	5	0,480	1,120	2,080
22	1,400	1,140	5	0,999	0,401	2,399
23	1,800	0,837	5	0,733	1,067	2,533
24	1,400	0,548	5	0,480	0,920	1,880
25	0,600	1,140	5	0,999	-0,399	1,599
26	1,400	1,517	5	1,329	0,071	2,729

Figura 4.10: Intervalli di confidenza per ogni elemento

Distribuzione delle risposte

Il grafico in figura 4.11 mostra la distribuzione delle risposte per ogni elemento del questionario, confermando nuovamente la generale bontà del

prodotto ed accentuando il sentimento abbastanza comune di imprevedibilità, come già discusso nelle sezioni precedenti.

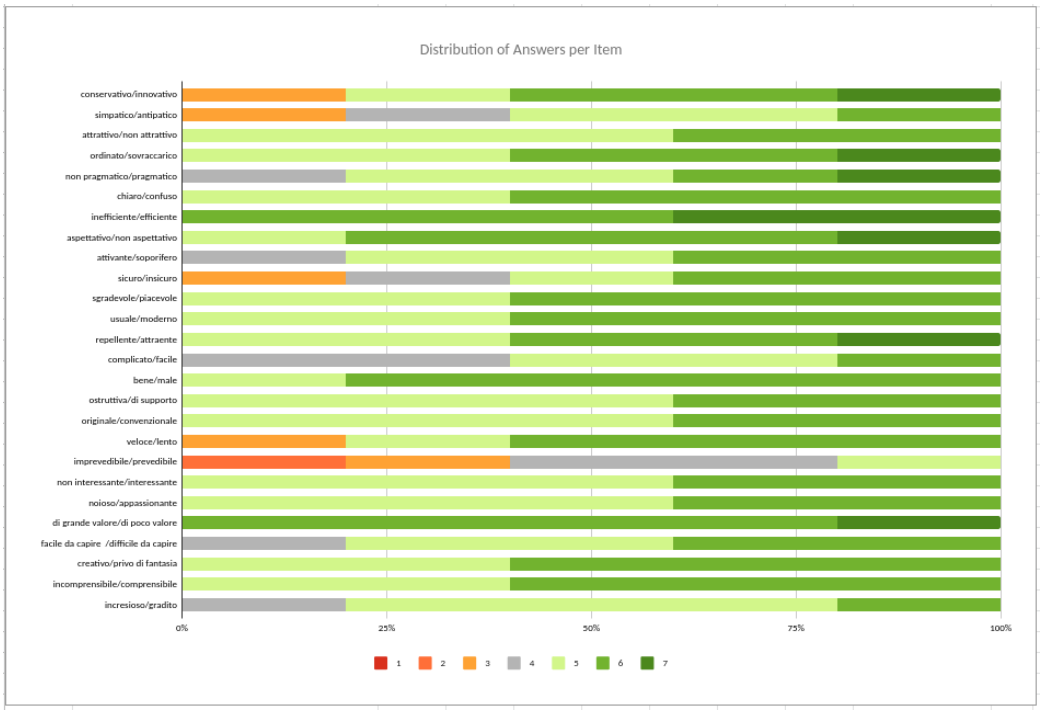


Figura 4.11: Rappresentazione grafica della distribuzione delle risposte per ogni elemento

Consistenza delle scale

In figura 4.12 viene fornita la tabella con calcolati i *coefficienti Alpha di Cronbachs*, mentre in figura 4.13 viene fornita la tabella con calcolati i *coefficienti di Lambda2 di Guttman*.

I valori dei coefficienti ottenuti indicano che i risultati del questionario non sono affidabili per essere utilizzati per effettuare valutazioni statistiche, come si evince dalla letteratura [45] [46].

I valori di tali coefficienti dovrebbero essere compresi nell'intorno [0, 1], poiché rappresentano una percentuale, mentre i valori osservati sono

quasi tutti negativi. Questa differenza si ha poiché, come già evidenziato precedentemente, il numero di partecipanti al questionario è molto esiguo.

Attrattività	
Items	Correlation
1, 12	-0,79
1, 14	0,42
1, 16	0,00
1, 24	-0,65
1, 25	-0,62
12, 14	-0,80
12, 16	-0,41
12, 24	0,41
12, 25	0,78
14, 16	0,87
14, 24	-0,33
14, 25	-0,89
16, 24	-0,17
16, 25	-0,72
24, 25	0,72
Average	-0,14
Alpha	**-3,10**
Conf. Int. Alpha (5%)	-22,68
	0,29

Apprendibilità	
Items	Correlation
2, 4	-0,33
2, 13	0,33
2, 21	-0,67
4, 13	-0,64
4 , 21	0,76
13, 21	-0,76
Average	-0,22
Alpha	**-2,53**
Conf. Int. Alpha (5%)	-21,40
	0,44

Efficienza	
Items	Correlation
9, 20	-0,84
9, 22	-0,91
9, 23	0,50
20, 22	0,88
20, 23	-0,33
22, 23	-0,68
Average	-0,23
Alpha	**-2,91**
Conf. Int. Alpha (5%)	-23,84
	0,38

Controllabilità	
Items	Correlation
8, 11	-0,48
8, 17	-0,57
8, 19	-0,62
11, 17	-0,21
11, 19	0,00
17, 19	0,81
Average	-0,18
Alpha	**-1,53**
Conf. Int. Alpha (5%)	-15,06
	0,60

Stimolazione	
Items	Correlation
5, 6	-0,41
5, 7	-0,41
5, 18	0,53
6, 7	1,00
6, 18	-0,76
7, 18	-0,76
Average	-0,13
Alpha	**-0,91**
Conf. Int. Alpha (5%)	-11,10
	0,70

Originalità	
Items	Correlation
3, 10	0,67
3, 15	-0,67
3, 26	-0,66
10, 15	-1,00
10, 26	-0,84
15, 26	0,84
Average	-0,28
Alpha	**-6,56**
Conf. Int. Alpha (5%)	-46,97
	-0,19

Figura 4.12: Valori del coefficiente Alpha di Cronbachs

Attrattività	
Lambda1	-3,71428571
Items	Covar^2
1, 12	0,04
1, 14	0,04
1, 16	0,00
1, 24	0,04
1, 25	0,16
12, 14	0,06
12, 16	0,01
12, 24	0,01
12, 25	0,10
14, 16	0,10
14, 24	0,01
14, 25	0,46
16, 24	0,00
16, 25	0,13
24, 25	0,13
Sum	1,29
Lambda2	**-1,20**

Apprendibilità	
Lambda1	-1,85714285
Items	Covar^2
2, 4	0,02
2, 13	0,02
2, 21	0,04
4, 13	0,20
4 , 21	0,12
13, 21	0,12
Sum	0,53
Lambda2	**-0,15**

Efficienza	
Lambda1	-4,71428571
Items	Covar^2
9, 20	0,23
9, 22	1,17
9, 23	0,19
20, 22	0,19
20, 23	0,01
22, 23	0,27
Sum	2,07
Lambda2	**-1,36**

Controllabilità	
Lambda1	-1,2352941
Items	Covar^2
8, 11	0,06
8, 17	0,46
8, 19	0,16
11, 17	0,01
11, 19	0,00
17, 19	0,36
Sum	1,05
Lambda2	**-0,25**

Stimolazione	
Lambda1	-1,1428571
Items	Covar^2
5, 6	0,01
5, 7	0,01
5, 18	0,03
6, 7	0,06
6, 18	0,08
7, 18	0,08
Sum	0,25
Lambda2	**0,03**

Originalità	
Lambda1	-1,1333333
Items	Covar^2
3, 10	0,03
3, 15	0,03
3, 26	0,19
10, 15	0,06
10, 26	0,31
15, 26	0,31
Sum	0,93
Lambda2	**-0,08**

Figura 4.13: Valori del coefficiente Lambda2 di Guttmans

Benchmark

Il grafico in figura 4.14 rappresenta la comparazione dei *valori medi* delle scale ottenuti nel questionario con un data set contenente le valutazioni di 21175 persone raccolti da 468 studi concernenti prodotti differenti (pagine Web, business software, shop online, social network) [43].

Per i dati raccolti, la suite di componenti si posizione in una posizione per lo più sopra la media, evidenziando ancora una volta però di peccare per quanto riguarda la *Controllabilità*.

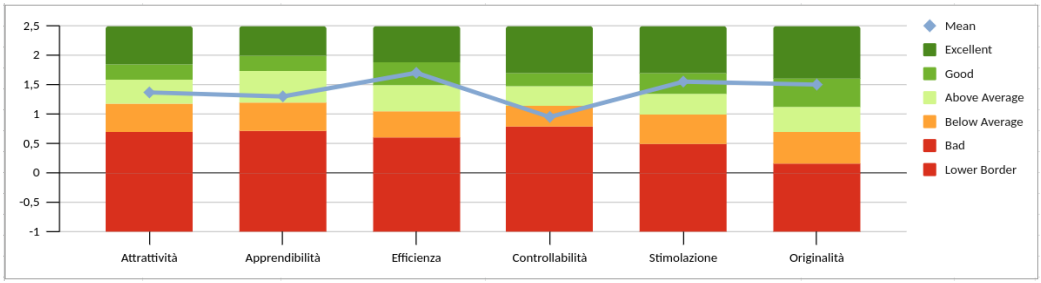


Figura 4.14: Rappresentazione grafica della comparazione con altri prodotti già sul mercato

Inconsistenze

La figura 4.15 riporta eventuali inconsistenze nelle risposte dei partecipanti al questionario. Le inconsistenze possono essere frutto di incomprensione della domanda o se è stata data una risposta in maniera non seria (ad esempio rispondendo a caso), tali risposte dovrebbero quindi essere rimosse dal conteggio per ottenere un'analisi dei dati qualitativamente migliore.

Vengono considerate risposte poco consistenti e affidabili quelle che presentano un valore maggiore di 3, mentre le risposte che presentano 2 o 3 inconsistenze devono essere considerate un po' sospette.

Nel caso del questionario in esame, non sono state riscontrate particolari inconsistenze all'interno delle risposte dei partecipanti.

Scales with inconsistent answers						
Attrattività	Apprendibilità	Efficienza	Controllabilità	Stimolazione	Originalità	Critical?
			1			1
						0
			1			1
						0
1		1				2

Figura 4.15: Incosistenze rilevate all'interno del questionario suddivise per scale

Calcolo del KPI

In questa sezione viene calcolato un singolo valore di KPI per quanto riguarda l'esperienza utente nei confronti del prodotto, in modo da poter fornire agli organi di controllo un unico valore numerico che attesti, seppur in modo abbastanza generico dal punto di vista tecnico, la bontà del prodotto in esame.

Tale valore viene calcolato partendo dalle risposte alle 6 domande (figura 4.16) che descrivono in linguaggio naturale le caratteristiche di ogni scala, risposte date su una scala di valori di Likert da 1 a 7.

Successivamente viene calcolato il valore di importanza relativo per ogni partecipante, che viene a sua volta moltiplicato col *valore medio* per la stessa scala relativo alle risposte dello stesso partecipante, i valori così ottenuti per ogni scala vengono sommati per ottenere il valore di KPI relativo ad ogni partecipante (figura 4.17).

Una volta calcolati tutti i valori di KPI per ogni singolo partecipante, si ottiene un valore di KPI complessivo dato dalla media di questi (figura 4.18).

Uno studio [47] portato avanti dai creatori dello UEQ attesta che il valore di KPI così calcolato oscilla in un intervallo tra `-0.286` e `+2.143`, sebbene un intervallo teorico tra `-3` e `+3` risulti possibile, quest'ultimo non verrà mai veramente osservato in applicazioni reali.

Enter the data concerning the importance of the scales below!						
Attrattività	Apprendibilità	Efficienza	Controllabilità	Stimolazione	Originalità	SUM
5	6	6	5	5	5	32
7	7	6	6	6	6	38
5	5	6	6	5	4	31
6	7	6	7	5	6	37
7	7	7	7	6	6	40

Figura 4.16: Risposte al questionario relativo alla sezione del KPI

Calculated relative importance						
Attrattività	Apprendibilità	Efficienza	Controllabilità	Stimolazione	Originalità	KPI
0,16	0,19	0,19	0,16	0,16	0,16	**1,43**
0,18	0,18	0,16	0,16	0,16	0,16	**1,47**
0,16	0,16	0,19	0,19	0,16	0,13	**1,44**
0,16	0,19	0,16	0,19	0,14	0,16	**1,44**
0,18	0,18	0,18	0,18	0,15	0,15	**1,21**

Figura 4.17: Importanza relativa calcolata per ogni scala

KPI 1,40	**STD 0,11**	**Confidence: 0,09**

Figura 4.18: Valore del KPI calcolato

Con un valore di KPI calcolato di `1.40`, ci si può ritenere contenti del risultato ad oggi ottenuto, rivelando quanto il lavoro compiuto si sia concretizzato in un valore aggiunto per l'azienda.

Le figure che seguono 4.20 e 4.19 riportano i *valori medi* osservati per ogni scala e ne danno una rappresentazione grafica, che si rende utile anche per gli organi di controllo per meglio valutare decisioni future.

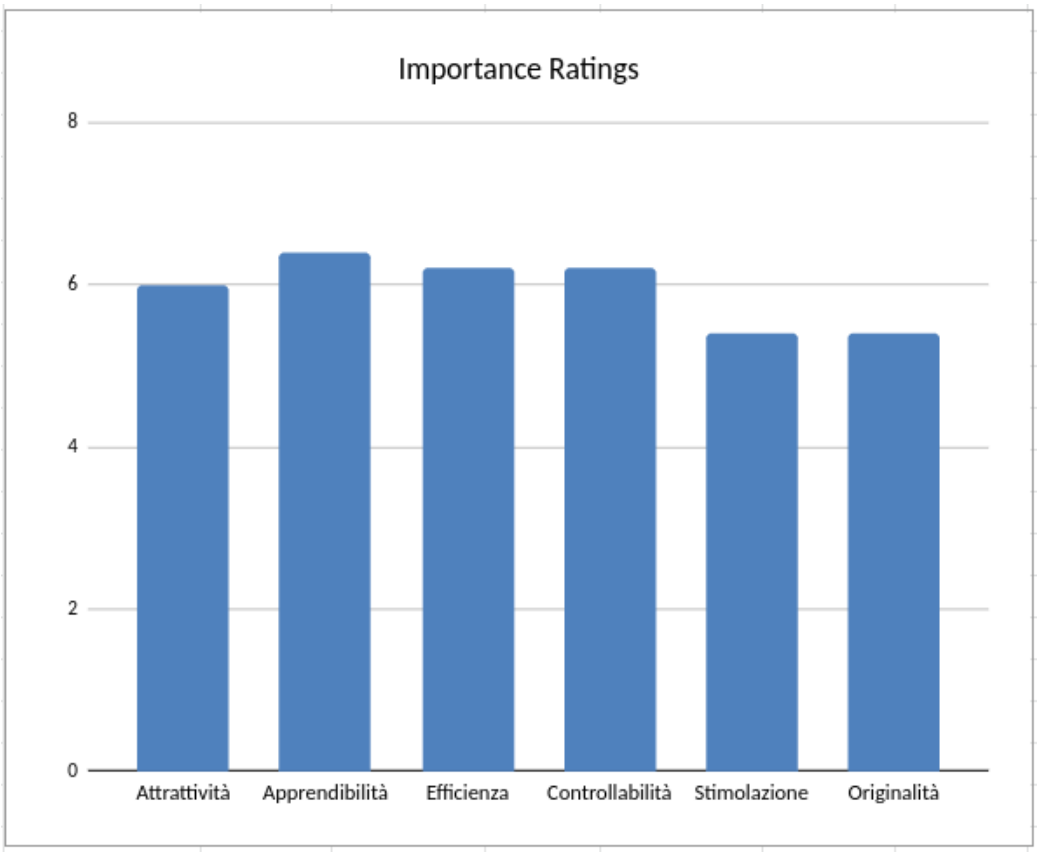


Figura 4.19: Rappresentazione grafica del valore medio dei risultati di importanza delle scale

Importance Ratings: Mean and Standard Deviation						
Skala	Mean	Std.Dev.	N	Confidence	Confidence Interval	
Attrattività	6,00	1,00	5	0,88	5,12	6,88
Apprendibilità	6,40	0,89	5	0,78	5,62	7,18
Efficienza	6,20	0,45	5	0,39	5,81	6,59
Controllabilità	6,20	0,84	5	0,73	5,47	6,93
Stimolazione	5,40	0,55	5	0,48	4,92	5,88
Originalità	5,40	0,89	5	0,78	4,62	6,18

Figura 4.20: Valore medio e deviazione standard dei risultati di importanza delle scale

4.2 Retrospettiva

Sono state riportate molte soddisfazioni nel risultato attualmente raggiunto, dalle metodologie adottate e dalle soluzione trovate. L'introduzione del progetto all'interno del monorepository esistente del *Design System* aziendale ha permesso una migliore gestione delle interdipendenze che lo sviluppo dei Web Component ha con gli altri progetti presenti. Anche l'integrazione di processi di automazione per la Continous Integration e Continous Delivery ha contribuito a migliorare l'esperienza di sviluppo e di rilascio, specialmente facilitando il lavoro di controllo di integrità delle modifiche introdotte.

Primo esempio concreto di sviluppo di un prodotto che fa uso del *Design System* (e quindi anche dei componenti personalizzati sviluppati, oggetto di tesi) si può ritrovare nel prodotto *Periodici Maggioli*[5], dove in figura 4.22 viene presentato il cambiamento avvenuto a partire dallo stato precedente visibile in figura 4.21.

[5] `https://www.periodicimaggioli.it/`

Figura 4.21: Periodici Maggioli prima della sua riscrittura

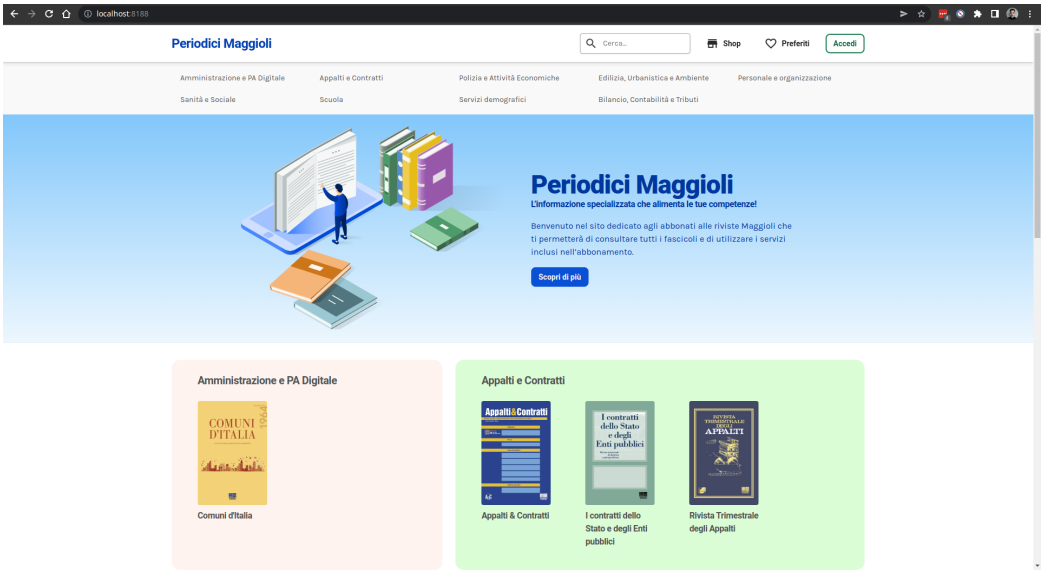


Figura 4.22: Periodici Maggioli sviluppato utilizzando il Design System aziendale

La compilazione del questionario da parte degli sviluppatori che hanno introdotto l'uso della suite dei componenti all'interno dei loro prodotti ha permesso di fornire alcuni risultati tangibili (sebbene il numero del campione non sia sufficiente per dare ufficialità ai dati ottenuti) utili a dare un riscontro concreto sull'utilità e utilizzabilità del lavoro svolto, sia al core team a cui è affidato lo sviluppo, sia agli organi di controllo.

4.2.1 Problematiche riscontrate

Durante lo sviluppo della suite, ci si è dovuti scontrare più volte con le peculiarità introdotte dall'utilizzo dello Shadow DOM, sia per quanto riguarda la fruizione dei componenti in ambienti diversi come Storybook o in produzione, sia per quanto riguarda la scrittura dei test unitari ed end-to-end, ma sempre trovando alla fine una soluzione.

Problematica ancora presente, e sulla quale porre abbastanza attenzione, risulta essere il supporto da parte della libreria Stencil ai `formAssociated`, di cui è stata data una rapida panoramica nella sezione 1.3.2. Questa funzionalità infatti risulta non essere ancora supportata da parte della libreria, come si evince da una issue[6] presente sul repository GitHub ufficiale di Stencil da più di due anni.

Questa mancanza di supporto ha generato non poche noie, poiché ciò rende molto più laborioso l'associazione di un Web Component, scritto con la libreria Stencil che utilizza lo Shadow DOM, e i form in cui è inserito.

Vedendo come la discussione nella issue abbia ripreso ad animarsi nell'ultimo periodo da parte della community, considerata anche la mole di lavoro necessaria per poter riuscire a far partecipare un componente personalizzato che usa lo Shadow DOM in maniera "nativa" con i form, è stato deciso di adottare un'altra soluzione.

[6] `https://github.com/ionic-team/stencil/issues/2284`

I componenti di input, intesi per lavorare anche in maniera "nativa" con i form, non fanno uso di Shadow DOM per essere completamente isolati dall'ambiente circostante, ma fanno uso degli *scoped style*[7].

Questa alternativa non isola il Web Component dalla struttura DOM nella quale viene posto, ma permette comunque al componente di utilizzare il proprio CSS per poter stilare i suoi componenti. Questo non impedisce comunque allo stile presente nel DOM di poter penetrare il componente e modificarne lo stile, ma questa eventualità, che è stata presa in considerazione, è stata valutata essere la scelta migliore per il momento.

4.2.2 Considerazioni finali

I soggetti a cui è stato somministrato il questionario provengono tutti da team di sviluppo diversi e che lavorano su prodotti diversi (quelli già citati nel capitolo 2.2), assicurando in questo modo una certa eterogeneità sui dati raccolti. Quello che purtroppo non è stato possibile assicurare è un numero elevato di partecipanti, dato che la suite di componenti è stata introdotta in pochi prodotti, e per ogni prodotto solamente una o due persone al massimo erano responsabili dell'integrazione e dell'utilizzo dei componenti.

I risultati ottenuti rispecchiano la buona andatura del progetto, come è possibile notare dagli alti valori della *media* per la maggior parte della scale nella figura 4.6 nella sezione 4.1.2, soprattutto per quanto riguarda l'*Efficienza*. A confermare ciò, anche i *valori medi* delle qualità pragmatiche e qualità edoniche riportate in figura 4.8, che riflettono l'*usabilità* e l'*interesse* suscitato in chi usa il prodotto, riportano valori relativamente alti (se consideriamo non l'intervallo tra `-3` e `+3` ma quello tra `-2` e `+2`).

Come già riportato, unico valore che rimane un po' sotto la media, come è anche possibile notare nel diagramma del benchmark in figura 4.14 che effettua il confronto del prodotto in esame con altri prodotti già presenti sul

[7] `https://stenciljs.com/docs/styling#scoped-css`

mercato, è la scala della *Controllabilità*. Una motivazione dietro a questo valore potrebbe essere la presenza di una documentazione alquanto scarna per quanto riguarda l'utilizzo dei componenti, ampliarla con esempi dell'utilizzo effettuati sia su ambiente JavaScript/TypeScript che HTML dovrebbero infatti essere presi in considerazione.

Si era già consci della poca attendibilità ai fini statistici dei risultati ottenuti (come è possibile notare nelle sezioni 4.1.2 e 4.1.2) data dalla bassa partecipazione al questionario, ciò non di meno, come riportato da Jakob Nielsen in un articolo del 2000 [4], il test di un prodotto con soli 5 utenti è sufficiente per scovare l'85% dei problemi di usabilità presenti. Tale valutazione deriva da un precedente studio condotto nel 1993 da Nielsen e Thomas K. Landauer [48], dove hanno formulato che il numero di problemi di usabilità trovati grazie a test di usabilità con n utenti è

$$N(1 - (1 - L)^n)$$

dove N è il numero totale di problemi di usabilità all'interno della progettazione e L è la percentuale di problemi di usabilità scoperti quando un solo utente viene testato, tipicamente osservato essere attorno a 31%.

In figura 4.23 viene rappresentato il grafico con i valori di L in base al numero di utenti sottoposti ai test di usabilità. Come si può notare, con 5 utenti si ottiene un'ottima percentuale di scoperta di problemi di usabilità.

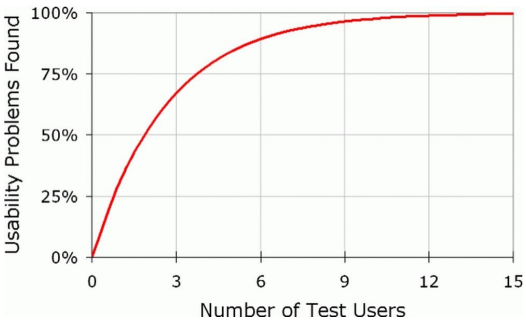


Figura 4.23: Curva rappresentate la percentuale di L in base al numero di utenti sottoposti ai test di usabilità [4]

Conclusioni

In questo lavoro di tesi sono state definite e valutate le azioni per la realizzazione, secondo i moderni standard Web, di una suite di componenti personalizzati da poter utilizzare all'interno dei propri prodotti che, unitamente all'uso di un design system aziendale, ha come scopo rendere facilmente riconoscibile e omogenee le interfacce utente sviluppate. I vantaggi derivanti non vengono goduti solo dagli utenti finali, che trovano nei vari prodotti offerti un'attenzione nell'offrire un'ottima esperienza di navigazione ed utilizzo ed un filo comune nell'interfaccia che sin da subito ne fa ricondurre la manifattura all'azienda, ma anche e soprattutto dagli sviluppatori chiamati a creare questi prodotto, che con le tecnologie e gli strumenti descritti in questo elaborato vengono sollevati dalla decisione di alcune scelte stilistiche ma principalmente dal dover riscrivere, per ogni nuovo prodotto, un nuovo set di componenti grafici. Nello specifico, questo lavoro di tesi si è concentrato su un caso di studio specifico, presso il Gruppo Maggioli.

L'analisi eseguita sul questionario redatto ha riportato dei risultati soddisfacenti per quanto riguarda l'apprezzamento del caso di studio da parte degli sviluppatori che sono stati coinvolti. Questo fa sì che si possa considerare il caso di studio come una ottima *prima prova sul campo*, confermando le scelte prese sulla tecnologia e sulle modalità adottate. Ciononostante, è bene sottolineare che il numero di partecipanti è stato comunque limitato; questo non ha quindi consentito di ottenere risultati validi ai fini statistici, ma solo considerazioni dal punto di vista qualitativo.

Si dovranno tenere ovviamente conto anche delle ombre emerse dal que-

stionario, principalmente per la sensazione di scarsa *Controllabilità* sentita da parte dei partecipanti, effettuando un'indagine volta a carpire le cause che hanno portato a tale valutazione, ma per il momento non è stato ancora deciso nulla in merito.

Gli sviluppi futuri inerenti alla suite di componenti riguarderanno principalmente l'aggiunta di ulteriori componenti, probabilmente quelli che interagiscono con gli oggetti `form` del DOM, andando inoltre a potenziare quella che è la documentazione attuale. In merito a ciò, al momento della stesura della tesi, sono in corso i lavori per pubblicare la prima versione del sito della documentazione del design system aziendale *Magma*, sviluppando facendo uso di moderne tecnologie e, ovviamente, della suite di Web component oggetto del caso di studio.

Una volta che la suite di componenti sarà stata arricchita di nuove funzionalità e sarà entrata in uso anche in altri team e prodotti, sarà necessario ripetere la somministrazione del questionario coinvolgendo un maggior numero di partecipanti, per poter effettuare un'analisi ritenuta maggiormente valida sia a fini statistici che come base su cui fondare future decisioni.

Appendice

Questionari somministrati ai team di sviluppo che hanno utilizzato la suite di componenti, tutti i camp sono obbligatori.

Valutazione utilizzo suite web components di Magma - Parte I

Aiutaci a capire come consideri il prodotto con cui stai lavorando, non ci sono risposte "giuste" o "sbagliate". Quello che conta è la tua opinione personale.

1. Fastidioso [1 2 3 4 5 6 7] Piacevole
2. Incomprensibile [1 2 3 4 5 6 7] Comprensibile
3. Creativo [1 2 3 4 5 6 7] Privo di fantasia
4. Facile da apprendere [1 2 3 4 5 6 7] Difficile da apprendere
5. Di grande valore [1 2 3 4 5 6 7] Di poco valore
6. Noioso [1 2 3 4 5 6 7] Appassionante
7. Non interessante [1 2 3 4 5 6 7] Interessante
8. Imprevedibile [1 2 3 4 5 6 7] Prevedibile
9. Veloce [1 2 3 4 5 6 7] Lento

10. Originale [1 2 3 4 5 6 7] Convenzionale
11. Ostruttivo [1 2 3 4 5 6 7] Di supporto
12. Buono [1 2 3 4 5 6 7] Scarso
13. Complicato [1 2 3 4 5 6 7] Facile
14. Repellente [1 2 3 4 5 6 7] Attraente
15. Usuale [1 2 3 4 5 6 7] Moderno
16. Sgradevole [1 2 3 4 5 6 7] Gradevole
17. Sicuro [1 2 3 4 5 6 7] Insicuro
18. Attivante [1 2 3 4 5 6 7] Soporifero
19. Conforme alle aspettative [1 2 3 4 5 6 7] Non conforme alle aspettative
20. Inefficente [1 2 3 4 5 6 7] Efficente
21. Chiaro [1 2 3 4 5 6 7] Confuso
22. Non pragmatico [1 2 3 4 5 6 7] Pragmatico
23. Ordinato [1 2 3 4 5 6 7] Sovraccarico
24. Invitante [1 2 3 4 5 6 7] Non invitante
25. Congeniale [1 2 3 4 5 6 7] Ostile
26. Conservativo [1 2 3 4 5 6 7] Innovativo

Valutazione utilizzo suite web components di Magma - Parte II

Aiutaci a capire come consideri il prodotto con cui stai lavorando, non ci sono risposte "giuste" o "sbagliate". Quello che conta è la tua opinione personale.

1. Il prodotto dovrebbe essere invitante, piacevole, congeniale e gradevole
   Per niente d'accordo [1 2 3 4 5 6 7] Molto d'accordo

2. Dovrei eseguire la mia mansione tramite il prodotto in maniera veloce, efficiente e pragmatica
   Per niente d'accordo [1 2 3 4 5 6 7] Molto d'accordo

3. Il prodotto dovrebbe essere facile da capire, chiaro, semplice e facile da imparare
   Per niente d'accordo [1 2 3 4 5 6 7] Molto d'accordo

4. L'interazione col prodotto dovrebbe essere prevedibile, sicuro e incontrare le mie aspettative
   Per niente d'accordo [1 2 3 4 5 6 7] Molto d'accordo

5. Usare il prodotto dovrebbe essere interessante, appassionante e attivante
   Per niente d'accordo [1 2 3 4 5 6 7] Molto d'accordo

6. Il prodotto dovrebbe essere innovativo, originale e progettato in modo
   Per niente d'accordo [1 2 3 4 5 6 7] Molto d'accordo

Bibliografia

[1] Custom elements everywhere, 2022. URL https://2021.stateofjs.com/en-US/.

[2] Maggioli Editore. Sito periodicimaggioli.it, 2022. https://periodicimaggioli.it.

[3] Maggioli R&D. Installazione pubblica di storybook, 2022. URL https://magma.maggiolicloud.it/storybook/.

[4] Jakob Nielsen. Why you only need to test with 5 users, 2000. https://www.nngroup.com/articles/why-you-only-need-to-test-with-5-users/.

[5] Ku. Chhaya A. Khanzode and Dr. Ravindra D. Sarode. Evolution of the world wide web: From web 1.0 to 6.0. *International Journal of Digital Library Services*, 2016.

[6] San Murugesan. *WEB APPLICATION DEVELOPMENT: CHALLENGES AND THE ROLE OF WEB ENGINEERING*, chapter 2. Human-Computer Interaction Series. Springer London, first edition, 2008. doi: /10.1007/978-1-84628-923-1.

[7] Mitra Bauer. Hello framework! a heuristic method for choosing front-end javascript frameworks. Master's thesis, Uppsala Universitet, 2021.

[8] State of js, 2022. URL https://2021.stateofjs.com/en-US/.

[9] Jarrod Overson and Jason Strimpel. *Developing Web Components: UI from JQuery to Polymer*. O'Reilly Media, Inc., 2015. ISBN 1491905700, 9781491905708.

[10] Michael Karén. The state of web components, 2019. URL https://medium.com/swlh/the-state-of-web-components-e3f746a22d75.

[11] Lior Chamla. The state of web components, 2017. URL https://medium.com/apprendre-le-web-avec-lior/a-brief-history-of-webcomponents-and-a-tutorial-to-make-yours-a52d329913e7.

[12] Sergey Borodanov. Overview of the web components in 2021, 2021. URL https://exyte.com/blog/web-components-technology.

[13] Kyle Buchanan. Web components and seo, 2019. URL https://medium.com/patternfly-elements/web-components-and-seo-58227413e072.

[14] Brian Harnish. Google discusses whether seo professionals can use web components for seo, 2022. URL https://iloveseo.com/news/google-discusses-whether-seo-professionals-can-use-web-components-for-seo/.

[15] MDN Web Component. documentazione ufficiale, 2022. URL https://developer.mozilla.org/en-US/docs/Web/Web_Components.

[16] Gil Fink. Understanding the customelementregistry object, 2018. URL https://gilfink.medium.com/understanding-the-customelement registry-object-74408a25a3d4.

[17] HTML Living Standard. documentazione ufficiale, 2022. URL https://html.spec.whatwg.org/multipage/custom-elements.html#custom-elements-face-example.

[18] Eric Bidelman. Custom elements v1 - reusable web components, 2019. URL https://web.dev/custom-elements-v1/.

[19] Stefan Nieuwenhuis. Lifecycle hooks in web components, 2019. URL `https://ultimatecourses.com/blog/lifecycle-hooks-in-web-components`.

[20] HTML Living Standard. documentazione ufficiale, 2022. URL `https://html.spec.whatwg.org/multipage/custom-elements.html#custom-element-reactions`.

[21] Arthur Evans. More capable form controls, 2019. URL `https://web.dev/more-capable-form-controls/#lifecycle-callbacks`.

[22] Lit. documentazione ufficiale, 2022. URL `https://lit.dev/docs/`.

[23] Stencil. documentazione ufficiale, 2022. URL `https://stenciljs.com/docs/introduction`.

[24] FAST. documentazione ufficiale, 2022. URL `https://www.fast.design/docs/introduction/#what-is-fast`.

[25] Gruppo Maggioli. Sito ufficiale, 2022. URL `https://maggioli.com/it-it`.

[26] Ilaria Vesentini. Maggioli, terza acquisizione in spagna nel giro di tre anni. *Il Sole 24 Ore*, page 14, 6 2019.

[27] Nicola Strazzacapa. Gruppo maggioli alla conquista della spagna. *Corriere Romagna*, page 24–25, 10 2020.

[28] Maggioli Tributi, 2022. `https://www.maggiolitributi.it`.

[29] Il gruppo maggioli svela il bilancio sociale. l'esercizio chiude con un +31% e nuove assunzioni, Luglio 2022. URL `https://www.riminitoday.it/economia/gruppo-maggioli-svela-bilancio-sociale-esercizio-chiude-crescita.html`.

[30] Gioele Masini. Meta style guide. Master's thesis, Alma Mater Studiorum - Università di Bologna - Campus di Cesena, 2021.

[31] Nrwl. documentazione ufficiale, 2022. URL https://nx.dev/getting-started/intro.

[32] Mario Nebl. repository github, 2022. URL https://commitlint.js.org/#/.

[33] Divya Verma. 7 of the best tools to create mockup design for website, 2022. URL https://www.webdew.com/blog/mockup-design-for-website.

[34] Figma. documentazione ufficiale, 2022. URL https://www.figma.com/.

[35] Storybook. documentazione ufficiale, 2022. URL https://storybook.js.org/.

[36] Leonardo Giroto. Visual regression testing, 2020. URL https://medium.com/loftbr/visual-regression-testing-eb74050f3366.

[37] Joel Arvidsson. documentazione ufficiale, 2022. URL https://loki.js.org/getting-started.html.

[38] Mustafa Ezer. Loki: A visual regression testing for your storybook projects, 2021. URL https://javascript.plainenglish.io/loki-a-visual-regression-testing-journey-on-projects-that-use-storybook-1ff5a8530a51.

[39] Nathan Kurtis. Why design system contributions matter, 2020. URL https://medium.com/eightshapes-llc/why-contributions-matter-22652d8676c6.

[40] Nathan Kurtis. Why design system contributions matter, 2020. URL https://medium.com/eightshapes-llc/defining-design-system-contributions-eb48e00e8898.

[41] Nathan Kurtis. Why design system contributions matter, 2020. URL https://medium.com/eightshapes-llc/design-system-subtasks-by-step-2f43d7d4bce0.

[42] GitLab. documentazione ufficiale, 2022. https://docs.gitlab.com/ee/ci/pipelines/.

[43] User Expierence Questionnaire. documentazione ufficiale, 2022. https://www.ueq-online.org/.

[44] Bettina Laugwitz, Theo Held, and Martin Schrepp. Construction and evaluation of a user experience questionnaire. In *HCI and Usability for Education and Work*, 4th Symposium of the Workgroup Human-Computer Interaction and Usability Engineering of the Austrian Computer Society, pages 63–76. Springer, 2008. doi: 10.1007/978-3-540-89350-9_6. URL https://www.researchgate.net/publication/221217803_Construction_and_Evaluation_of_a_User_Experience_Questionnaire.

[45] Statistics How To. Guttman's lambda-2: Definition, examples, 2022. https://www.statisticshowto.com/guttmans-lambda-2/.

[46] Statistics How To. Cronbach's alpha: Definition, interpretation, spss, 2022. https://www.statisticshowto.com/probability-and-statistics/statistics-definitions/cronbachs-alpha-spss/.

[47] Andreas Hinderks, Martin Schrepp, Francisco José Domínguez Mayoa, Escalona María José, and Jörg Thomaschewski. Developing a ux kpi based on the user experience questionnaire. *Computer Standards & Interfaces*, 65:38–44, July 2019. ISSN 0920-5489. doi: 10.1016/j.csi.2019.01.007. URL https://doi.org/10.1016/j.csi.2019.01.007.

[48] Jakob Nielsen and Thomas K. Landauer. A mathematical model of the finding of usability problems. CHI '93: Proceedings of the INTERACT '93 and CHI '93 Conference on Human Factors in Computing Systems, pages 206–213. Association for Computing Machinery, New York, NY, United States, 1993. doi: 10.1145/169059.169166. URL https://dl.acm.org/doi/10.1145/169059.169166.

[49] Il gruppo maggioli oltre la pandemia: chiude il 2020 con un bilancio positivo. *Corriere Romagna*, page 25, 4 2021.

[50] Open Web Components. Community: Base libraries, 2022. URL `https://open-wc.org/guides/community/base-libraries/`.