



**UNIVERSITÀ
DEGLI STUDI
DI UDINE**
hic sunt futura

**Dipartimento di Scienze
Matematiche, Informatiche e Fisiche**

TESI DI LAUREA IN
INFORMATICA

Sito web statico per gruppo di ricerca accademico: analisi e implementazione

CANDIDATO

Federico Dittaro

RELATORE

Prof. Marino Miculan

CORRELATORE

Dott. Matteo Paier

Anno accademico 2022-2023

CONTATTI DELL'ISTITUTO

Dipartimento di Scienze Matematiche, Informatiche e Fisiche

Università degli Studi di Udine

Via delle Scienze, 206

33100 Udine — Italia

+39 0432 558400

<https://www.dmif.uniud.it/>

Dedica da scrivere

Indice

1	Introduzione	1
2	Analisi del software	3
2.1	SSG	3
2.2	principali SSG	3
2.2.1	Hugo	3
2.2.2	Jekyll	4
2.2.3	Gridsome	5
2.2.4	Eleventy	6
2.2.5	Pelican	7
2.2.6	Tabella riassuntiva	8
2.3	Differenze tra siti statici e dinamici	8
2.4	Hugo	9
2.4.1	Formati supportati	9
2.4.2	Template actions	10
2.4.3	Plugin	10
2.4.4	Creazione del sito	12
2.4.5	Costruzione del sito	13
2.4.6	Hosting e distribuzione	13
2.4.7	Web Analytics	14
3	Realizzazione di un sito web	17
4	Caso di studio	19
5	Conclusioni	21
A	Glossario	23

1

Introduzione

In hac habitasse platea dictumst. Vestibulum consectetur dictum pellentesque. Suspendisse nunc neque, commodo ac imperdiet nec, sollicitudin vitae libero. Donec bibendum vel nunc vitae pharetra. In vel volutpat odio, et interdum dui. Duis mauris ligula, congue eget molestie at, tincidunt nec diam. Nam vitae eros nec arcu suscipit vehicula. Aliquam consectetur imperdiet elit, eget pretium arcu fringilla at. Maecenas [1] sed libero pulvinar, mattis tortor vel, fermentum enim.

2

Analisi del software

2.1 SSG

Gli SSG (in inglese, Static Site Generator) sono dei tool che permettono la creazione di tutti i contenuti presenti nei siti web a partire da file di configurazione e contenuti scritti in formati più generali (tipicamente markdown). La caratteristica principale di tali siti è che a fronte di una richiesta da parte dell'utente di visualizzare determinati contenuti del sito, il web server fornisce pagine statiche, delle quali l'utente non è in grado di modificare il contenuto né possiede alcun tipo di stato che ne permetta la personalizzazione. Non esiste quindi una elaborazione back-end sul lato server e non esistono database, qualsiasi funzionalità "dinamica" associata al sito statico viene eseguita sul lato client.

I principali vantaggi riguardanti la scelta di utilizzo di un SSG sono:

- Ottimizzazione delle prestazioni: avendo poche o nessuna parte dinamica sono più facili da ottimizzare ed il caricamento è molto rapido;
- Richiesta di meno risorse al server: dato che non è richiesta nessuna elaborazione lato server, quest'ultimo svolge meno lavoro migliorando prestazioni e scalabilità;
- Servizio di hosting molto economico: possono essere utilizzati per la pubblicazione servizi di host completamente gratuiti come GitHub Pages (esattamente come nel caso di studio);
- Maggiore sicurezza: non utilizzando server o database sono molto sicuri da eventuali attacchi esterni.

2.2 principali SSG

In questa sezione verranno menzionati i cinque principali SSG e discusse le loro differenze più significative.

2.2.1 Hugo

Hugo verrà brevemente presentato in modo da poterlo confrontare con gli altri principali SSG, per poi essere ripreso più approfonditamente nella Section 2.4 dato che si tratta del framework scelto per sviluppare il progetto.



Figura 2.1: Hugo logo

Hugo è un generatore di siti web statici scritto in Go ideato inizialmente da Steve Francia nel 2013 e successivamente sviluppato da Bjørn Erik. L'ultima versione, la 0.119.0 è stata rilasciata a settembre 2023. Per utilizzare Hugo non è necessario conoscere Go in quanto il sito web viene creato attraverso file HTML e CSS, inoltre c'è una separazione tra il contenuto e la presentazione permettendo così di modificare l'aspetto senza modificarne il contenuto. Oltre ai tipi di file citati precedentemente, Hugo supporta anche file di tipo javascript, Markdown, TOML, YAML e JSON.

Le informazioni necessarie per creare, modificare, stilizzare o eliminare pagine e/o contenuti sono racchiuse all'interno di specifiche cartelle che possono essere successivamente estese. La struttura generale è la seguente:

- la cartella *archetypes* contiene i file che vengono utilizzati come template per la creazione di nuovi contenuti del sito in modo da standardizzare la struttura ed il formato;
- la cartella *content* è forse la più importante in quanto contiene tutto il contenuto del sito. Al suo interno tutti i file sono in formato Markdown;
- la cartella *data* contiene esclusivamente file di tipo JSON, TOML, YAML o XML utilizzati per aggiungere strutture specifiche al sito;
- la cartella *layouts* contiene file HTML usati per creare l'aspetto visivo del sito;
- la cartella *static* contiene file statici come ad esempio immagini, file CSS, file Javascript;
- la cartella *themes* contiene i file che definiscono il tema del sito ed il suo aspetto visivo;
- il file *config.toml* oppure *config.yaml* è fondamentale in quanto rappresenta il file di configurazione e contiene informazioni globali come il titolo del sito, la sua descrizione e molto altro.

Hugo è noto per la sua velocità ed inoltre supporta una grande varietà di temi scaricabili direttamente dal sito ufficiale. A differenza di altri SSG, Hugo non è indicato solamente per la creazione di blog ma anche per la creazione di siti generici come ad esempio siti aziendali o, come nel caso di studio, per siti accademici. Hugo mette inoltre a disposizione una grande varietà di Plugin molto utili come ad esempio il servizio per la rappresentazione delle icone social o il supporto multilingua.

2.2.2 Jekyll

Jekyll è un generatore di siti web statici, ideato da Tom Preston-Werner, la prima versione del software risale al 2008 mentre l'ultima, la 3.9.3, è uscita a gennaio 2023. Jekyll si basa sul linguaggio Ruby, perciò richiede un'installazione ed una configurazione corretta e funzionante di tale ambiente. Successivamente si scarica la versione desiderata di Jekyll e si segue la procedura di installazione, così come descritta sulla documentazione. In Jekyll tutti i contenuti e i layout del sito vengono salvati localmente e vengono



Figura 2.2: Jekyll logo

classificati in una struttura a cartelle, principalmente orientata alla costruzione di blog.

Una volta creato il sito, la struttura trovata sarà la seguente:

- la cartella *_posts* contiene gli articoli del sito (composti da file Markdown);
- i contenuti delle pagine, sempre composti da file Markdown, sono salvati nella cartella *root*, in alternativa si può decidere di creare una gerarchia di sottocartelle per una migliore organizzazione dei contenuti;
- la cartella *_layouts* contiene i vari template del sito che decidono la grafica delle singole pagine e dei singoli articoli (questi file sono sempre di tipo HTML);
- la cartella *_site* contiene tutte le informazioni necessarie per esportare il sito funzionante nel dominio o in sistemi cloud;
- la cartella *_data* può essere creata per contenere dei file JSON in cui saranno costruiti dei database per immagazzinare stringhe, numeri e altri dati simili;
- la cartella *assets* contiene immagini, pdf o altri file statici per il sito.

Come Hugo anche Jekyll mette a disposizione centinaia di temi prefabbricati per aiutare lo sviluppo del sito web, ed entrambi forniscono degli shortcode, ovvero funzioni che permettono la comunicazione tra i layout delle pagine con i loro contenuti (ad esempio le template actions per Hugo). Anche Jekyll presenta una moltitudine di Plugin che possono essere integrati attraverso Ruby, permettendo di aggiungere e semplificare la costruzione di determinati servizi per il sito web.

Una delle differenze principali di Hugo rispetto a Jekyll è che il primo non è legato ad ambienti esterni, infatti dopo aver scaricato la versione desiderata ed estratto il contenuto nella cartella prescelta il software è pronto per essere usato, mentre Jekyll si deve appoggiare a Ruby.

In conclusione, Jekyll è un'ottima scelta se si ha familiarità con l'ambiente Ruby o se si vuole costruire un sito complesso usando gli innumerevoli Plugin e template messi già a disposizione.

2.2.3 Gridsome

Gridsome è un SSG molto recente, è stato infatti ideato da Johannes Schickling nel 2018 subendo poi miglioramenti negli anni successivi fino all'ultima versione disponibile, la 0.7.23, rilasciata a settembre 2021. Si tratta di un framework basato su Vue.js e GraphQL che permette di creare una configurazione "headless", consentendo così di sfruttare la separazione dei contenuti dalla loro presentazione. Le cartelle



Figura 2.3: Gridsome logo



Figura 2.4: Eleventy logo

di lavoro possono variare in base alla configurazione specifica di un progetto, ma in generale sono strutturate nel modo seguente:

- *src* è la cartella principale al cui interno si trova il contenuto sorgente. È suddivisa in sottocartelle come *assets* per file statici, *components* per componenti Vue.js, *layouts* per i layout del sito, *pages* per le pagine principali e *templates* per i template utilizzati per la generazione di pagine dinamiche;
- *static* è la cartella utilizzata per i file statici che verranno serviti direttamente, come immagini, file CSS o JavaScript;
- *.gridsome* è la cartella che contiene le configurazioni specifiche di Gridsome. Include i file di configurazione e i dati temporanei generati durante la compilazione del sito;
- *gridsome.config.js* è il file di configurazione principale, definisce le impostazioni globali e le opzioni del progetto.
- *package.json* è il file che definisce le dipendenze del progetto e gli script personalizzati.

Anche Gridsome offre una vasta raccolta di plugin che possono essere utilizzati per estendere le funzionalità del generatore ma a differenza degli altri SSG presentati non è così adatto ai principianti, necessita infatti di una certa esperienza nello sviluppo web per poter riuscire a trarre il massimo da questo software.

2.2.4 Eleventy

Eleventy (chiamato anche 11ty), come Gridsome, è un SSG molto recente ideato da Zach Leatherman nel 2018 e migliorato fino alla versione 2.0.0, rilasciata a febbraio 2023. La sua crescita è dovuta sia alla sua flessibilità e potenza ma anche al sempre maggior utilizzo delle piattaforme di hosting come "Chrome Developers" e "Netlify".

Questo software si basa su Node.js e javascript, il che richiede una buona conoscenza di quest'ultimo linguaggio per poter sfruttare a pieno la sua potenzialità. Eleventy supporta molteplici linguaggi di modello ma fondamentalmente si basa su Liquid, il che lo rende simile a Jekyll. Come tutti gli altri SSG supporta diversi linguaggi come ad esempio Markdown, JSON, YAML.

La struttura di base contiene almeno le seguenti cartelle:



Figura 2.5: Pelican logo

- *_includes*: in questa cartella possono essere messi frammenti di codice HTML o template come ad esempio intestazioni o piè di pagina;
- *_layouts*: questa cartella contiene i layout dei template come ad esempio un layout principale che include un'intestazione, piè di pagina e altri layout specifici;
- *_data*: questa cartella è utilizzata per i file di dati, come file JSON o YAML, che vengono utilizzati per fornire dati dinamici alle pagine del sito;
- *_pages*: in questa cartella vengono collocate le pagine del sito, utilizzando file che verranno convertiti in pagine HTML;
- *_posts*: questa cartella viene utilizzata per i post del blog o altri contenuti dinamici;
- *_assets*: questa cartella può essere utilizzata per archiviare i file statici come fogli di stile CSS, immagini e JavaScript. Tali file verranno copiati direttamente nella cartella *_site* durante la generazione del sito;
- *.eleventy.js*: questo è il file di configurazione principale di Eleventy, in cui è possibile personalizzare il comportamento del generatore, definire i percorsi delle cartelle e altro ancora.

Eleventy non è sicuramente il miglior SSG per iniziare a lavorare in questo campo, sia per la necessità di conoscere un linguaggio come javascript sia per la documentazione poco esaustiva e ancora in fase di sviluppo, tuttavia lo sforzo viene guadagnato in termini di velocità e prestazioni.

2.2.5 Pelican

Pelican è meno conosciuto e utilizzato rispetto agli altri presenti in questa sezione, ma si tratta comunque di una valida alternativa data la sua flessibilità e la possibilità di importare siti già esistenti creati con altre piattaforme come ad esempio Wordpress. Questo SSG è stato creato da Justin Mayer nel 2010, l'ultima versione disponibile è la 4.8.0 rilasciata a luglio 2023 ed è stato principalmente sviluppato per la creazione di blog. A differenza degli altri ruota completamente attorno a python, creando pagine statiche attraverso file markdown e reStructuredText.

la struttura generale delle cartelle è la seguente:

- *content*: è la cartella principale in cui verranno collocati articoli, pagine e contenuti del blog;
- *output*: questa cartella contiene l'output generato da Pelican, ovvero il sito web statico completo con tutti i file HTML, CSS, JavaScript e le altre risorse pronte per la pubblicazione;

- *settings*: in questa cartella possono essere collocati file di configurazione specifici per il progetto, utilizzati per personalizzare il comportamento di Pelican;
- *themes*: qui si possono collocare i temi o modelli per il sito web;
- *plugins*: utilizzata per estendere le funzionalità di Pelican. I plugin possono essere utilizzati per eseguire varie azioni, come la generazione automatica di mappe del sito o l'integrazione con servizi di terze parti;
- *media*: questa cartella può essere utilizzata per archiviare file multimediali, come immagini o video, che verranno inclusi nel sito web;
- *pages*: In questa cartella vengono collocate le pagine statiche che non sono articoli del blog;
- *archives*: questa cartella può essere utilizzata per archiviare articoli o pagine che non sono attualmente pubblici ma potrebbero essere utilizzati in futuro;
- *lib*: contiene script personalizzati o strumenti per il progetto Pelican.

Pelican offre inoltre una vasta serie di temi e plugin per estendere le funzionalità di base del generatore, come il supporto multilingua o la possibilità di importare dati di terze parti (es. Wordpress, feed RSS).

2.2.6 Tabella riassuntiva

La seguente tabella riassume le caratteristiche dei cinque principali SSG visti nelle sezioni precedenti.

	Anno creazione	Ultima versione	Linguaggio di base
Hugo	2013	0.119.0 settembre 2023	Go
Jekyll	2008	3.9.3 gennaio 2023	Ruby
Gridsome	2018	0.7.23 settembre 2021	Vue.js, GraphQL
Eleventy	2018	2.0.0 febbraio 2023	Node.js, javascript
Pelican	2010	4.8.0 luglio 2023	Python

Tabella 2.1: Tabella riassuntiva SSG.

2.3 Differenze tra siti statici e dinamici

Riprendendo la Section 2.1, un'ulteriore differenza che si può notare tra siti statici e dinamici riguarda l'architettura client/server utilizzata.

Per quanto riguarda i siti dinamici, l'architettura adottata prende il nome di WAMP/LAMP, l'acronimo è composto dalle seguenti parole: la W/L riguarda il sistema operativo utilizzato (rispettivamente Windows oppure Linux), la A sta per Apache ovvero il server http impiegato per eseguire il server web, la M per il DBMS MySQL e la P per PHP. Alcune architetture utilizzano al posto di MySQL il DBMS MariaDB e al posto del PHP i linguaggi Python oppure Pearl (l'acronimo rimane comunque invariato).

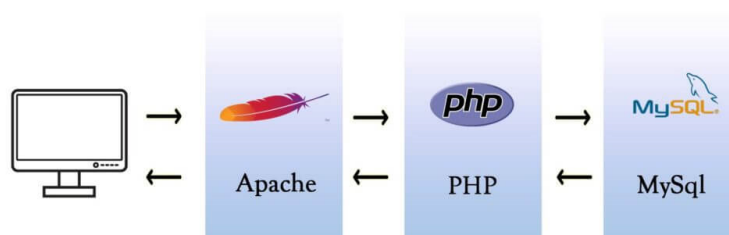


Figura 2.6: Architettura WAMP/LAMP

WAMP è quindi la piattaforma di sviluppo web grazie a cui è possibile realizzare dei siti web dinamici attraverso la programmazione con il linguaggio PHP, permettendo così la creazione dei siti direttamente sul computer. Utilizzare questa architettura attraverso il PHP offre diversi vantaggi in quanto si tratta di uno dei linguaggi più frequentemente utilizzati in rete ed inoltre è il linguaggio di riferimento per WordPress, la principale piattaforma software per lo sviluppo di siti dinamici.

Per quanto riguarda i siti statici invece, una volta generato il sito è sufficiente caricare i file su qualsiasi web server o servizio di hosting, non avendo infatti la necessità di eseguire codice dinamico (es. PHP) e di interfacciarsi con un database, la struttura risultante è molto più snella. Avere un'architettura di questo tipo permette di ottenere delle ottimizzazioni delle prestazioni e della velocità necessaria per servire la pagina richiesta.

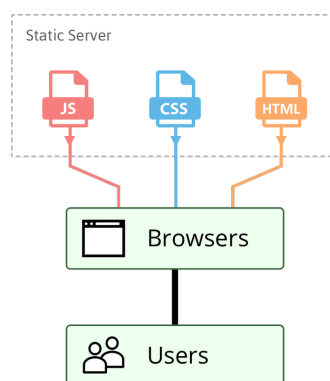


Figura 2.7: Architettura client/server SSG

2.4 Hugo

In questa sezione verrà ripreso ed ampliato il contenuto della Section 2.2.1 in quanto Hugo rappresenta l'SSG scelto per la realizzazione del caso di studio.

2.4.1 Formati supportati

Hugo supporta una grande varietà di formati, i principali sono i seguenti:

- **MARKDOWN**: formato di markup contenente esclusivamente il contenuto informativo, in questi file non viene definito nessun tipo di stile;

- **HTML:** formato utilizzato per definire l'organizzazione e la struttura delle pagine;
- **CSS:** formato utilizzato per definire lo stile delle pagine (es. colore, sfondo, dimensioni e tipo di carattere, ...);
- **JavaScript:** questi file vengono utilizzati principalmente per l'esecuzione di script, sono fondamentali per l'esecuzione di specifici task.
- **JSON:** vengono utilizzati per definire metadati e contenuti strutturati all'interno del sito web;
- **TOML, YAML:** questi file vengono utilizzati per definire o aggiungere dettagli strutturali o globali del sito e delle singole pagine.

Vengono inoltre supportati formati più specifici come `reStructuredText`, un formato markup utilizzato ad esempio per la documentazione tecnica, oppure `AsciiDoc`, anch'esso un formato markup spesso utilizzato in ambienti di sviluppo.

2.4.2 Template actions

Le Template actions, scritte in Go Template, si riferiscono a una serie di azioni o comandi disponibili nei modelli (templates) di Hugo, che vengono utilizzati per manipolare e visualizzare i dati all'interno dei contenuti. Le template actions consentono di eseguire operazioni come l'iterazione attraverso gli elenchi, il controllo di flusso condizionale e l'estrazione di valori dai dati dei contenuti. Vengono utilizzate direttamente all'interno dei file HTML all'interno delle doppie graffe.

Alcune delle template actions più comuni in Hugo includono:

- **range:** utilizzata per iterare all'interno di un'elenco, permette ad esempio di creare dei cicli.
- **if, with:** consentono di aggiungere logica condizionale nei modelli. Possono essere utilizzati per visualizzare o nascondere parti del contenuto in base a condizioni specifiche;
- **index:** utilizzata per accedere a elementi specifici all'interno di elenchi o dizionari;
- **partial:** per includere modelli o contenuti da altri file.

La Fig. 2.8 mostra un'esempio di utilizzo del controllo di flusso attraverso il costrutto "if", nello specifico verifica se nei parametri che descrivono la persona è presente un'immagine: in caso positivo mostra l'immagine corrispondente altrimenti un'immagine di default. La Fig. 2.9 mostra invece l'utilizzo della template action "partial", permettendo così di includere il file *recent.html*.

Nell template actions è possibile anche selezionare specifici parametri come ad esempio `{{ .Title }}` che permette di inserire il titolo della pagina corrente. Allo stesso modo si può ottenere la descrizione (`{{ .Description }}`) o il sommario (`{{ .Summary }}`).

2.4.3 Plugin

Hugo supporta una grande varietà di Plugin che permettono di estendere le funzionalità di base, in questa sezione ne verranno presentati due tra i più utilizzati.

Hugo-authors

Il plugin "hugo-authors" è progettato per semplificare la gestione dei dati degli autori. Consente di definire e visualizzare le informazioni sugli autori come nome, biografia, immagine del profilo e collegamenti ai social media, in modo più strutturato.

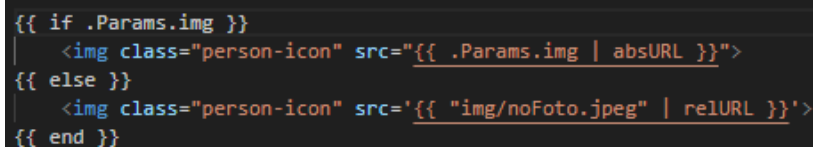
Per poter utilizzare questo plugin è necessario prima installarlo attraverso il modulo Hugo o includendolo nel file di configurazione (config.toml oppure config.yaml). Successivamente è possibile creare dei nuovi autori attraverso la seguente sintassi:

```
[author]
[[author.authors]]
name = "author name"
bio = "author bio"
image = "author.jpg"
social = [
{ name = "Twitter", link = "link" },
{ name = "LinkedIn", link = "link" }
]
```

A questo punto è possibile associare gli autori al rispettivo contenuto specificando nel *front-matter* del file markdown il nome corrispondente. Il nome svolge quindi la funzione di ID. Ad esempio:

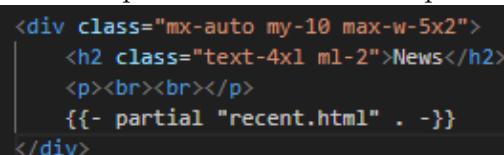
```
+++
title = "Il mio articolo"
author = "author name"
+++
```

Sfruttando infine le Template actions è possibile accedere alle caratteristiche dell'autore, ad esempio si può ricavare il nome scrivendo `{{ .Params.author }}` oppure applicando la stessa logica si possono



```
{{ if .Params.img }}
  
{{ else }}
  
{{ end }}
```

Figura 2.8: Esempio di utilizzo della template action "if"



```
<div class="mx-auto my-10 max-w-5x2">
  <h2 class="text-4xl ml-2">News</h2>
  <p><br><br></p>
  {{- partial "recent.html" . -}}
</div>
```

Figura 2.9: Esempio di utilizzo della template action "partial"

ricavare le varie informazioni inserite (ad esempio la biografia).

Servizio delle icone social

Questo plug-in non necessita di installazione e permette di visualizzare facilmente icone e collegamenti ai profili social. Per poterlo utilizzare bisogna aggiungere le seguenti righe di codice al file di configurazione

```
[social]
platforms = [
name = "Twitter", icon = "twitter" ,
name = "LinkedIn", icon = "linkedin" ,
name = "GitHub", icon = "github" ,
# Altre piattaforme social
]

icons = [
name = "twitter", link = "https://example.com/twitter-icon.svg" ,
name = "linkedin", link = "https://example.com/linkedin-icon.svg" ,
name = "github", link = "https://example.com/github-icon.svg" ,
# Altre icone social
]
```

Solo successivamente possono essere specificate all'interno della definizione degli autori (sezione precedente) o nel front matter di un contenuto. Allo stesso modo del plugin precedente, le informazioni possono essere richiamate all'interno dei file HTML sfruttando le Template actions (in questo caso tramite `{{ .Site.Data.social.icons }}`).

2.4.4 Creazione del sito

Dopo l'installazione di Hugo, per creare un nuovo sito è sufficiente digitare da terminale il seguente comando **hugo new site NOME_SITO**. In seguito a questo comando verrà costruita la struttura delle cartelle così come descritto nella Section 2.2.1 che può essere modificata secondo le esigenze.

Per poter visualizzare in locale il sito che si sta creando, è necessario lanciare da terminale il comando **hugo server** oppure **npm run start** (se è stato installato npm), in seguito il sito sarà visibile all'indirizzo `localhost:1313` (`127.0.0.1:1313`).

Altri comandi utili per la creazione del sito sono **hugo server -D** che, a differenza del comando precedente, permette di visualizzare anche i contenuti classificati come bozze ed il comando **hugo**, necessario per la costruzione del sito. Quest'ultimo comando verrà discusso nella sezione successiva.

2.4.5 Costruzione del sito

La costruzione del sito è divisa in due fasi principali: la fase di generazione e la fase di compilazione.

Fase di generazione

Il processo di generazione viene avviato attraverso il comando **hugo**, il quale genera un unico file CSS combinando i dati di file di template dei temi con eventuali fogli di stile personalizzati, unendo successivamente anche il contenuto dei file Markdown.

Fase di compilazione

Terminata la fase di generazione, Hugo applica i temi e genera il codice HTML e CSS per ogni pagina del sito. L'ultima fase prima di poter rendere le pagine distribuibili e visualizzabili consiste nel produrre un insieme di fogli di stile CSS, immagini e altri file statici, ciascuno rappresentante una pagina del sito. Una variante più efficiente è il comando **hugo - -minify** che ottimizza il caricamento delle pagine minimizzando la dimensione dei file attraverso l'eliminazione di spazi vuoti, commenti e caratteri non necessari.

Il risultato del processo di compilazione è la creazione di due cartelle:

- **public**: contiene il sito costruito ed è organizzata a sua volta in sottocartelle ognuna delle quali contiene una pagina singola oppure un gruppo di pagine correlate. Ogni sottocartella comprende un file HTML, i fogli di stile e i file JavaScript;
- **resources**: contiene le cache di output del processo di generazione; nello specifico, contiene risorse come immagini, file CSS, file JavaScript e altri assets statici utilizzati nel sito.

2.4.6 Hosting e distribuzione

In seguito alla generazione e compilazione, il sito può essere caricato su qualsiasi web server o servizio di hosting. Come detto precedentemente, non è necessario eseguire elaborazioni lato server oppure richieste ad un database, pertanto è possibile utilizzare servizi di hosting completamente gratuiti.

Un'esempio di questi servizi è *Github Pages*, offerto da GitHub. Quest'ultimo è uno dei servizi di hosting di repository Git più popolari al mondo che offre servizi per il controllo di versione, la collaborazione e l'hosting di progetti software (attraverso appunto Github Pages). Alcune delle principali funzionalità di GitHub includono il controllo di versione distribuito, il tracciamento delle modifiche, la collaborazione tra sviluppatori e molto altro. GitHub Pages è uno dei servizi offerti da Github che consente di pubblicare siti web statici direttamente da una repository GitHub, in particolare utilizza uno specifico branch chiamato "gh-pages" dal quale preleva il contenuto che viene poi pubblicato. Di default, l'indirizzo che viene assegnato è `"https://NOME_UTENTE.github.io/NOME_REPOSITORY/"` ma supporta anche domini personalizzati.

GitHub Pages tuttavia presenta dei limiti sulla dimensione dei siti che consente di hostare:

- Le dimensioni della repository utilizzata per creare il sito web non deve essere superiore ad un 1GB;
- Le risorse messe a disposizione sono limitate a 100GB di banda e 100.000 richieste al mese;
- Non si dovrebbero eseguire più di dieci "build" per ora, per evitare di sovraccaricare i server e mantenerli performanti per tutti gli utenti.

I limiti appena citati possono essere superati attraverso un piano di pagamento del servizio.

Altri limiti imposti e non superabili riguardano il contenuto del sito web, GitHub infatti non consente di utilizzare questo servizio per gestire attività online e servizi di software commerciale, inoltre l'utilizzo della pagine è soggetto ai Termini di servizio di Github, per cui sono vietati contenuti sessuali oppure violenti/minacciosi.

2.4.7 Web Analytics

Le web analytics sono il processo di raccolta, misurazione, analisi e reporting dei dati relativi all'uso di un sito web al fine di comprendere il comportamento degli utenti, valutare le prestazioni del sito e trarre informazioni utili per migliorare l'esperienza degli utenti. Le web analytics sono fondamentali per il miglioramento dell'usabilità, l'ottimizzazione dei motori di ricerca e l'analisi delle conversioni.

- **Raccolta dei Dati:** la raccolta dei dati inizia con l'implementazione di strumenti di analisi web che tracciano il comportamento degli utenti sul sito. Questi strumenti registrano dati come il numero di visite, il tempo trascorso sul sito, le pagine visitate e molto altro;
- **Misurazione delle Metriche Chiave:** le web analytics misurano una vasta gamma di metriche, tra cui il traffico del sito, l'origine del traffico, il tasso di rimbalzo, il tempo medio sulla pagina, le conversioni e altro ancora. Queste metriche consentono di valutare le prestazioni del sito;
- **Analisi dei Dati:** l'analisi dei dati comporta l'interpretazione delle metriche raccolte per comprendere i modelli e le tendenze. Ad esempio, analizzare quale fonte di traffico genera più conversioni o quali pagine del sito hanno un alto tasso di rimbalzo;
- **Reporting:** la creazione di report è un passaggio importante nelle web analytics. I report consentono di comunicare i risultati dell'analisi in modo chiaro e comprensibile per poter essere condivisi con i membri del team, i responsabili decisionali o i clienti;
- **Ottimizzazione:** le web analytics sono utilizzate per prendere decisioni sull'ottimizzazione del sito. Ciò può includere l'ottimizzazione del contenuto, delle conversioni, del carico delle pagine e altro ancora al fine di migliorare l'esperienza dell'utente e raggiungere gli obiettivi stabiliti;
- **Segmentazione degli Utenti:** le web analytics consentono di suddividere gli utenti in segmenti in base a criteri come la provenienza geografica, il comportamento di navigazione, il dispositivo utilizzato e altro ancora. Questa segmentazione aiuta a comprendere meglio il pubblico e a personalizzare l'esperienza dell'utente.

Uno delle più famose piattaforme di analisi web è sicuramente *Google Analytics* che utilizza come strumento di analitica i cookies. Nel sito web che si è realizzato con questo progetto, viene utilizzata come piattaforma di analisi uno strumento meno conosciuto di Google Analytics: *umami*. Si tratta di un'applicazione open-source compatibile con il GDPR che a differenza di Google Analytics non utilizza cookie di tracciamento ma allo stesso tempo permette di ottenere informazioni come le pagine visitate, il browser, il sistema operativo ed il dispositivo utilizzati, nonché il paese di provenienza. Queste informazioni di base possono poi essere ampliate comprendendo ad esempio specifici eventi di interesse per l'analisi del sito web. Umami mette inoltre a disposizione una dashboard per la presentazione del numero di visite nel corso del tempo, permettendo di visualizzare gli accessi nella giornata, nell'ultima settimana, nell'ultimo mese oppure nell'anno corrente.

L'utilizzo e l'implementazione di umami verranno riprese nel Chapter 4.

3

Realizzazione di un sito web

4

Caso di studio

5

Conclusioni

A

Glossario

Bibliografia

- [1] Donald E. Knuth. *The TeXbook*. Addison-Wesley, 1986.