Università del Salento

Dipartimento di Ingegneria dell'Innovazione

Corso di Laurea Triennale in Ingegneria dell'Informazione

TESI DI LAUREA IN PRINCIPI DI INGEGNERIA DEL SOFTWARE

IMPLEMENTAZIONE DELLE SPECIFICHE DI DEMAND-RESPONSIVE DESIGN SU TECNOLOGIA REACT

Relatore

Prof. Roberto Vergallo

Laureando

Piero De Lorenzis

Matricola n° 20053187

ANNO ACCADEMICO 2022/2023

Indice

1. Introduzione

1.1 Contesto

Nell'era moderna, l'informatica e le tecnologie digitali hanno assunto un ruolo centrale nella vita quotidiana, influenzando profondamente la nostra interazione con l'ambiente circostante. Parallelamente, la crescente consapevolezza dell'impatto ambientale delle nostre azioni ha portato a una crescente attenzione verso la sostenibilità e la riduzione delle emissioni di gas serra, in particolare la CO_2, che contribuiscono al cambiamento climatico.

In questo ambito, la ricerca e lo sviluppo di soluzioni innovative che possano unire l'avanzamento tecnologico con l'obiettivo di ridurre le emissioni di carbonio assumono un ruolo di fondamentale importanza.

La presente tesi si inserisce all'interno di questo contesto, concentrandosi sull'elaborazione di una soluzione tecnologica che mira a ridurre le emissioni di CO_2 attraverso la modifica dinamica del design di siti web responsive[1]. L'obiettivo principale è lo sviluppo di un algoritmo, realizzato con la libreria React, in grado di adattare la grafica di un sito web in base al valore di "carbon intensity" nel momento in cui l'utente sta navigando il sito stesso. Per "carbon intensity" si intende la quantità in grammi di CO_2 emessa per unità di energia elettrica prodotta (1 kWh).

Questa soluzione può aiutare a sensibilizzare gli utenti sull'impatto ambientale del consumo di energia legato alla loro navigazione online.

[1]Sito web che adatta il proprio layout al formato del dispositivo su cui è visualizzato.

1.2 Obiettivi

Gli obiettivi di questa tesi sono:

- analizzare un algoritmo in grado di adattare in modo dinamico l'interfaccia di siti web come conseguenza della variazione del valore di carbon intensity;
- realizzare un sito web che sia in grado di testare l'efficacia di tale algoritmo in diversi contesti;
- valutare l'impatto ambientale di questa soluzione, calcolando la possibile riduzione di CO_2 emessa;
- individuare potenziali modifiche future dell'algoritmo per migliorare maggiormente l'impatto ambientale;

Attraverso questo sviluppo si vuole dimostrare che la tecnologia informatica può essere impiegata in modo innovativo per affrontare le sfide ambientali attuali e promuovere un futuro a basse emissioni di CO_2.

2. Stato dell'arte

2.1 Review scientifica

In questa sezione si analizzano alcuni studi sull'efficienza energetica delle applicazioni mobili.

In particolare si è preso in considerazione l'articolo 'Catalog of Energy Patterns for Mobile Applications'[2] pubblicato nel 2019. Questo paper presenta un catalogo di 22 design pattern[1] per migliorare l'efficienza energetica delle applicazioni mobili. I pattern sono stati identificati analizzando i commit[2], le issue[3] e le pull request[4] di 1027 app Android e 756 app iOS. Di seguito sono elencati i design pattern più inerenti a questo progetto.

- Dark UI Colors: fornisce un tema colore scuro dell'interfaccia utente per risparmiare batteria sui dispositivi con schermi AMOLED[5].

[1]Soluzione generica e riutilizzabile a un problema comune nell'ambito dello sviluppo del software o del design di sistemi complessi.

[2]Un commit nel controllo della versione è l'atto di registrare e salvare le modifiche al codice sorgente di un progetto software, insieme a una descrizione delle modifiche.

[3]Le issue sono strumenti di tracciamento nel controllo della versione utilizzati per registrare problemi, richieste di funzionalità, attività e discussioni all'interno di un progetto software.

[4]Una pull request è una richiesta per unire le modifiche apportate in un branch separato di un repository al branch principale.

[5]Uno schermo AMOLED è un tipo di display che emette luce direttamente dai pixel, consentendo neri profondi, colori vivaci e design sottile.

- Reduce Size: riduce il più possibile la dimensione dei dati durante la loro trasmissione.

- Avoid Extraneous Graphics and Animations: utilizzo moderato di grafica e animazioni per migliorare l'esperienza utente.

Questi design pattern sono una base di partenza per lo sviluppo del progetto.

Inoltre, dalla discussione presente nell'articolo si evince che gli sviluppatori di app Andorid (25%) sono maggiormente attenti all'utilizzo di pratiche per l'efficenza energetica, rispetto agli sviluppatori iOS (10%). Questo per vari motivi: meccanismi di gestione energetica implementati dal sistema, documentazione dei framework, differenze negli utenti target, differenze nei dispositivi target, ecc.

A completamento di questa revisione scientifica, si sono esplorate ulteriori risorse e informazioni tra cui il magazine online `branch.climateaction.tech`. Questo magazine, pubblicato dalla 'Green Web Foundation', si è dimostrato utile per accedere a contenuti dettagliati sull'intersezione tra tecnologia digitale, sviluppo sostenibile e riduzione delle emissioni di carbonio. Il magazine offre una varietà di articoli, approfondimenti e ricerche che esplorano le sfide e le opportunità legate all'ottimizzazione energetica delle nuove tecnologie. Di seguito sono riportate le instantanee delle diverse modalità di visualizzazione del sito.

Figura 2.1: Branch magazine - Modalità LOW

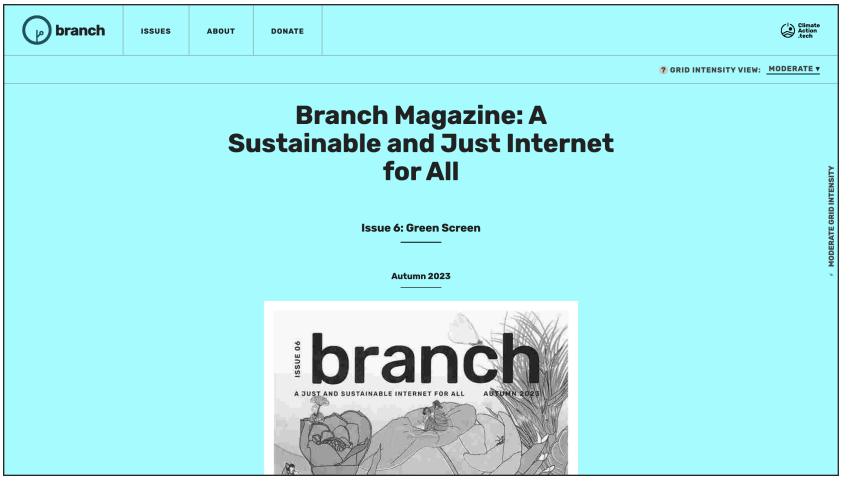

Figura 2.2: Branch magazine - Modalità MEDIUM

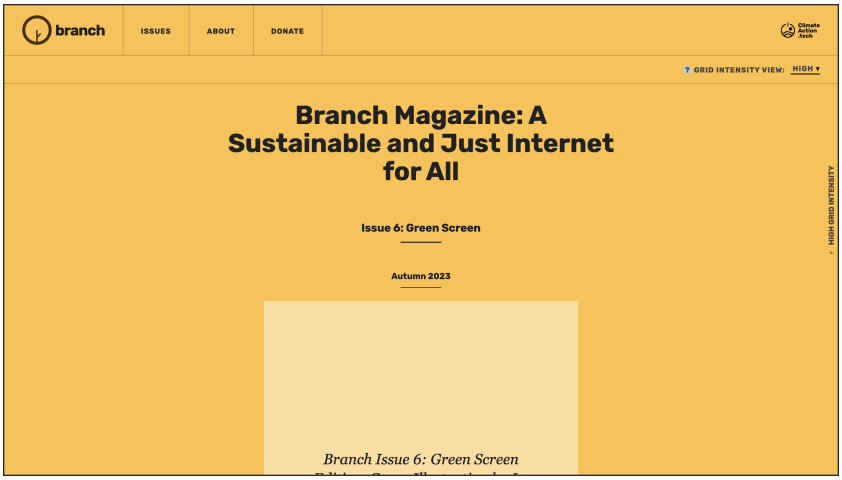

Figura 2.3: Branch magazine - Modalità HIGH

Analizzando il sito tramite lo strumento DevTools di Chrome, si è approfondito il funzionamento e la logica che permettono allo stesso di modificare il suo design. Nella sezione Rete di DevTools si è visto come al click del bottone LIVE viene effettuata una richiesta GET all'API `api.carbonintensity.org.uk` verso l'endpoint `/intensity/{from}/fw24h`. Questa chiamata restituisce il payload

```
{
  "data":[
   {
     "from":
     "to":
     "intensity": {
        "forecast":
        "actual":
        "index":
     }
   }]
}
```

La variabile `index` è una misura dell'intensità di carbonio rappresentata su una scala compresa tra "very low", "low", "moderate", "high", "very high".

Inoltre nel file `(index)`, presente nella scheda Sorgenti di DevTools, si è osservato che è presente una funzione chiamata `changeGridIntensity(intensity)`. Questa funzione imposta il design del sito tramite l'utilizzo di variabili CSS e modifica dinamicamente i percorsi delle immagini all'interno degli elementi `<figure>`.

La struttura del sito web è stata d'ispirazione per lo sviluppo di questo progetto.

2.2 Review tecnologica

Nella ricerca e nello sviluppo di questo progetto orientato alla sostenibilità ambientale, ci si è basati anche su principi e linee guida importanti che affrontano l'impatto ambientale causato dai siti web. Uno di questi principi è emerso dall'articolo 'Web Sustainability Guidelines'[3] pubblicato dalla W3C, l'organizzazione che sviluppa standard web globali e presentato il 13 settembre 2023 alla conferenza TPAC (Technical Plenary and Advisory Committee). L'articolo affronta la questione delle emissioni generate dai siti web e fornisce strategie per controllarle.

Questo sottolinea come le emissioni prodotte dai siti web possano essere generate in molteplici modi e come sia importante adottare misure per limitarle. In particolare, si concentra sulle 'CSS Preference Queries', che consentono di adattare il design dei componenti del sito in base alle preferenze degli utenti. Questo include la possibilità di disabilitare le animazioni, rimuovere i colori, adattarsi alla luminosità disponibile o addirittura offrire una versione della pagina più leggera in termini di larghezza di banda, in base alla richiesta dell'utente. Queste modifiche consentono di offrire un'esperienza di navigazione web meno impattante.

L'articolo definisce i criteri di successo per le 'CSS Preference Queries' e sottolinea il loro impatto positivo sull'ambiente, sull'equità sociale, sull'accessibilità, sulle prestazioni, e sull'economia.

Tali strategie sono state adottate nell'implementazione di questo progetto per ridurre l'impatto ambientale del sito web, consentendo ai visitatori di personalizzare la loro esperienza e ridurre il consumo di risorse digitali.

Di seguito, saranno esplorate in dettaglio il modo in cui sono state applicate queste linee guida per promuovere uno sviluppo web sostenibile.

Per lo sviluppo del progetto sono state prese in considerazione diverse tecnologie.

2.2.1 JavaScript

JavaScript è un linguaggio di programmazione ampiamente utilizzato per lo sviluppo di siti web e non solo. È un linguaggio di scripting ad alto livello che aggiunge interattività e dinamicità alle pagine web. JavaScript è principalmente utilizzato lato client[6], cioè eseguito direttamente nel browser web[7] dell'utente per migliorare l'esperienza dello stesso durante la navigazione sul web. Tuttavia, è anche utilizzato lato server[8] (ad esempio, con Node.js) per lo sviluppo di siti web basate su server.

2.2.2 Node.js

Node.js è un ambiente di runtime[9] JavaScript, open source[10] e multipiattaforma che consente agli sviluppatori di eseguire codice JavaScript lato server. È basato sul motore JavaScript V8 di Google Chrome. Node.js è noto per la sua velocità ed efficienza nell'esecuzione di operazioni di I/O[11] asincrone, il che lo rende particolarmente adatto per applicazioni web in tempo reale e per la gestione di molteplici connessioni simultanee. Node.js include una vasta libreria di moduli predefiniti che semplificano lo sviluppo di applicazioni. Inoltre, è possibile utilizzare il sistema di gestione dei pacchetti npm (Node Package Manager) per installare e gestire facilmente librerie e moduli di terze parti.

[6] Applicazione software o dispositivo che interagisce con i server web per accedere e visualizzare contenuti e risorse su Internet.

[7] Applicazione software progettata per consentire agli utenti di navigare e visualizzare pagine web su Internet.

[8] Computer o sistema informatico dedicato all'archiviazione, alla gestione e alla distribuzione di contenuti e risorse su Internet.

[9] Ambiente software in cui un programma o un'applicazione può essere eseguito.

[10] Modello di sviluppo e distribuzione del software in cui il codice sorgente del programma è reso disponibile pubblicamente e può essere utilizzato, studiato, modificato e distribuito liberamente da chiunque.

[11] Input/Output, ovvero operazioni di ingresso/uscita.

2.2.3 React

React è una libreria JavaScript open source sviluppata da Facebook per la creazione di interfacce utente (UI) per siti web. È ampiamente utilizzata per la costruzione di componenti UI riutilizzabili e reattivi, ciò la rende particolarmente adatta per lo sviluppo di siti web a singola pagina e per la gestione dinamica dell'interazione utente. React favorisce la suddivisione dell'interfaccia utente in componenti riutilizzabili. Questi componenti possono essere definiti in modo da reagire a cambiamenti nello stato del sito, consentendo un aggiornamento efficiente delle parti interessate dell'interfaccia utente quando i dati cambiano. React utilizza un concetto chiamato "Virtual DOM" per migliorare le prestazioni. Invece di manipolare direttamente il DOM[12] del browser, React crea una rappresentazione virtuale del DOM in memoria e aggiorna solo le parti effettivamente cambiate. Questo riduce le operazioni costose di aggiornamento del DOM e migliora la reattività del sito.

2.2.4 React-responsive-carousel

React-responsive-carousel è un modulo React utilizzato per la creazione di caroselli o slideshow responsive all'interno dei siti React. Un carosello è un componente UI comunemente utilizzato per visualizzare una serie di elementi (come immagini o contenuti) in sequenza, con la possibilità di scorrere tra di essi in modo automatico o manuale.

[12]Document Object Model (Modello ad Oggetti del Documento), è una rappresentazione gerarchica e strutturata dei documenti HTML, XHTML o XML all'interno di un browser web. Il DOM rappresenta ogni elemento presente in una pagina web come un oggetto, consentendo agli sviluppatori web di accedere, manipolare e modificare dinamicamente il contenuto e la struttura di una pagina web attraverso scripting, come JavaScript.

2.2.5 Bootstrap

Bootstrap è un framework[13] di sviluppo frontend[14] open source ampiamente utilizzato per la creazione di siti web responsive e moderni. È stato creato da Twitter e successivamente reso open source, diventando uno dei framework più popolari e utilizzati nella comunità di sviluppatori web.

2.2.6 Axios

Axios è una libreria JavaScript ampiamente utilizzata per effettuare richieste HTTP[15] da un'applicazione client a un server. Questa libreria è comunemente utilizzata nei siti web per effettuare chiamate AJAX (Asynchronous JavaScript and XML) e per comunicare con API (Application Programming Interface) esterne o con un server backend. Axios supporta anche l'invio di richieste con autenticazione, inclusa l'autenticazione basata su token, che è comune nei siti web moderni.

2.2.7 HTML

HTML, acronimo di "HyperText Markup Language" (Linguaggio di Marcatura Ipotestuale), è il linguaggio di marcatura standard utilizzato per creare pagine web. È uno dei componenti fondamentali di base di Internet ed è utilizzato per la strutturazione e la presentazione dei contenuti all'interno di un browser web. HTML è un linguaggio standard riconosciuto da tutti i browser web moderni e garantisce che le pagine web siano visualizzate correttamente su diversi dispositivi. HTML è spesso utilizzato in combinazione con altri

[13]Insieme di strumenti, librerie, linee guida e convenzioni di sviluppo software che fornisce una struttura predefinita per la creazione e la gestione di applicazioni o progetti software.

[14]Parte visibile e accessibile di un'applicazione che comprende l'interfaccia utente (UI) e l'esperienza utente (UX).

[15]Acronimo di "Hypertext Transfer Protocol" (Protocollo di Trasferimento di Iperscritti) è il protocollo di comunicazione utilizzato per la trasmissione di dati su Internet.

linguaggi e tecnologie web, come CSS per la formattazione e il layout delle pagine, e JavaScript per aggiungere interattività e funzionalità dinamiche. Insieme, questi tre linguaggi costituiscono i pilastri principali dello sviluppo web moderno, consentendo la creazione di siti web interattivi, dinamici e ben formattati.

2.2.8 SCSS

SCSS, acronimo di "Sassy CSS" o "Syntactically Awesome Style Sheets," rappresenta un'evoluzione dell'ormai onnipresente CSS[16] (Cascading Style Sheets). Progettato per migliorare la sintassi e l'efficienza della scrittura dei fogli di stile per il web, SCSS è un preprocessore CSS ampiamente utilizzato che offre funzionalità avanzate, facilitando la gestione della presentazione e del layout delle pagine web. Questo strumento è particolarmente utile per sviluppatori web e designer, poiché semplifica notevolmente la creazione e la manutenzione dei fogli di stile.

2.2.9 IntelliJ Idea

IntelliJ IDEA è un ambiente di sviluppo integrato (IDE) utilizzato principalmente per lo sviluppo di software in linguaggio Java ma supporta anche altri linguaggi di programmazzione tra cui Kotlin, Groovy, Scala, JavaScript, TypeScript, HTML, CSS, SQL e molti altri. È un IDE molto popolare tra gli sviluppatori e offre un insieme di strumenti avanzati per la scrittura e la gestione del codice. Inoltre offre un debugger altamente efficace, che semplifica il debug[17] del codice, la gestione dei breakpoint[18] e l'analisi dello stato del programma.

[16]Linguaggio di marcatura utilizzato per definire l'aspetto e la formattazione di un documento HTML o XML.

[17]Abbreviazione di "debugging" è il processo di identificazione, isolamento e correzione di errori o bug in un programma software o in un sistema informatico.

[18]Punti di interruzione del codice.

2.2.10 Google Chrome

Google Chrome è un popolare browser web sviluppato da Google. Il browser scarica i dati dalle pagine web tramite il protocollo HTTP o HTTPS e li interpreta per visualizzare il contenuto della pagina, che può includere testo, immagini, video, script interattivi e altro ancora. Chrome è noto per il suo ampio supporto per le tecnologie web moderne, tra cui HTML5, CSS3 e JavaScript avanzato, consentendo agli sviluppatori di creare siti web ricchi e interattivi. Chrome include un set completo di strumenti di sviluppo (DevTools) che consentono agli sviluppatori web di ispezionare il codice sorgente delle pagine web, testare e debuggare le loro applicazioni web.

2.2.11 Electricity Maps

Electricity Maps è un servizio online open source che fornisce informazioni in tempo reale sulla produzione elettrica e sulle emissioni di carbonio associate in diverse regioni del mondo. Electricity Maps visualizza la produzione di energia elettrica da fonti diverse, tra cui energia rinnovabile (come solare ed eolica), energia nucleare, energia da combustibili fossili e altre fonti. Il servizio calcola il valore di carbon intensity associato alla produzione di energia elettrica in ciascuna regione. Ciò permette agli utenti di vedere quanto la produzione di energia influisce sull'ambiente e sul cambiamento climatico. Electricity Maps ha anche una vasta gamma di funzionalità che consentono agli utenti di analizzare i dati energetici in dettaglio. Ad esempio, gli utenti possono visualizzare i dati storici sulla produzione e il consumo di energia elettrica, confrontare i dati di diversi periodi dell'anno e analizzare il mix energetico di un singolo paese o regione.

3. Studio di fattibilità

3.1 Architettura logica

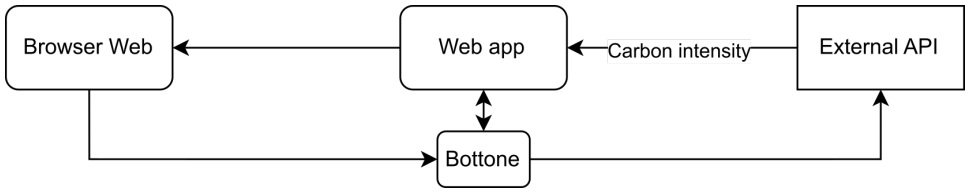

Figura 3.1: Architettura logica del progetto

Il diagramma 3.1 rappresenta l'architettura logica del progetto. L'utente, tramite il browser web, interagisce con un bottone che invia una richiesta a un'API esterna per ricevere il valore di carbon intensity. Questo valore viene poi utilizzato dal sito web per modificare il suo design.

3.2 Scelte tecnologiche

Per la realizzazione di questo progetto si è scelto di utilizzare la libreria React per vari motivi.

- Component-Based: React utilizza un'architettura basata su componenti, ciò significa che è possibile suddividere l'interfaccia utente in piccoli componenti. Questo approccio facilita la gestione, la manutenzione e l'aggiornamento del codice.

- Riutilizzabilità del Codice: grazie ai componenti, è possibile riutilizzare facilmente il codice in diverse parti del sito web, riducendone così la sua duplicazione per accelerare lo sviluppo.

- Virtual DOM: React utilizza un Virtual DOM per ottimizzare le operazioni di rendering[1]. Questo significa che React aggiorna solo le parti dell'interfaccia utente che sono state effettivamente modificate, migliorando le prestazioni complessive del sito.

- Flessibilità: React è flessibile e può essere utilizzato con altre librerie e framework, se necessario. È possibile incorporare facilmente componenti React in applicazioni esistenti.

- Strumenti di Sviluppo: React offre una serie di strumenti di sviluppo per il debugging, tra cui l'estensione React DevTools per browser.

Per la gestione della carbon intensity, che viene utilizzata all'interno dei componenti React, si è preferito utilizzare la libreria Axios per effettuare chiamete GET all'API esterna Electricity Maps. Di seguito sono elencati alcuni dei principali motivi che hanno portato alla scelta di Axios rispetto ad altre librerie.

- Gestione delle Richieste Asincrone: React è orientato agli stati e spesso gestisce dati asincroni. Axios semplifica la gestione delle richieste asincrone, in quanto supporta le Promises[2], questo significa che successo ed insuccesso sono facilmente gestibili quando si effettuano richieste HTTP.

- Compatibilità con i Browser: Axios è compatibile con la maggior parte dei browser moderni e fornisce un'implementazione consistente delle spe-

[1]Processo attraverso il quale un sistema informatico o un'applicazione visualizza graficamente i dati o il contenuto su uno schermo o un dispositivo di output.

[2]Oggetto che rappresenta l'eventuale completamento o fallimento di un'operazione asincrona.

cifiche XMLHttpRequest[3]. Ciò lo rende una scelta affidabile per progetti web cross-browser[4].

Per il collegamento all'API di Electricity Maps sarà utilizzato l'endpoint `https://api-access.electricitymaps.com/free-tier/carbon-intensity/latest?zone=IT` da cui verrà ottenuta la seguente struttura dati:

```
{
    "zone": "IT-CSO",
    "carbonIntensity":
    "datetime":
    "updatedAt":
    "createdAt":
    "emissionFactorType":
    "isEstimated":
    "estimationMethod":
}
```

Infine per la gestione di grafiche e icone del sito web è stato utilizzato il framework Bootstrap per i seguenti motivi:

- Risparmio di Tempo: Bootstrap fornisce un insieme di componenti e stili predefiniti, come modali, barre di navigazione, form, bottoni, e molto altro. Utilizzando Bootstrap, si risparmia tempo nello sviluppo poiché non è necessario scrivere stili CSS personalizzati o reimplementare componenti comuni da zero.

- Responsività: Bootstrap è progettato per la creazione di siti web responsive, per questo i suoi componenti si adattano automaticamente a diver-

[3]Insieme di standard e documenti tecnici che definiscono il comportamento e le funzionalità di un oggetto JavaScript ampiamente utilizzato per effettuare richieste HTTP asincrone.

[4]Capacità di un sito web di funzionare correttamente su una varietà di browser diversi.

se dimensioni di schermo e dispositivi, come computer desktop, tablet e telefoni cellulari.

- Personalizzazione: anche se Bootstrap offre stili predefiniti, è altamente personalizzabile. Si possono estendere e sovrascrivere le regole CSS di Bootstrap per adattarle alle esigenze specifiche del progetto.

- Compatibilità con React: Bootstrap può essere integrato facilmente in progetti React. Esistono librerie React specifiche, ad esempio React-Bootstrap, che offrono componenti Bootstrap come componenti React riutilizzabili.

3.3 Architettura fisica

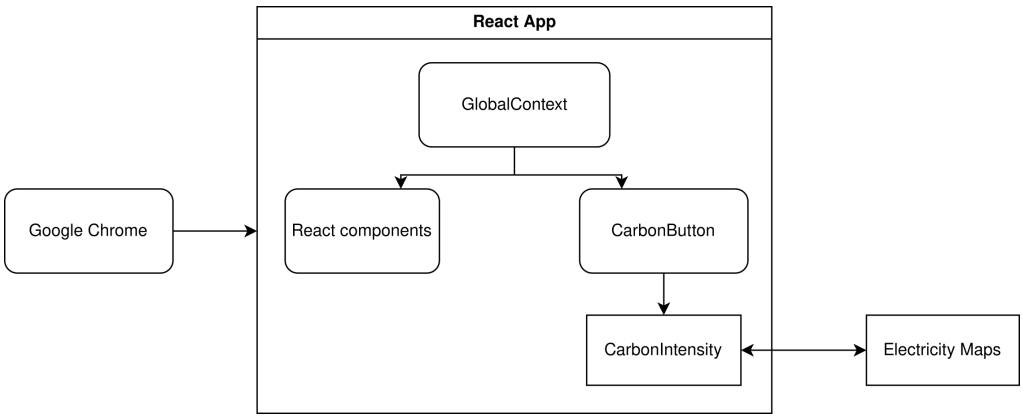

Figura 3.2: Architettura fisica del progetto

Il diagramma 3.2 rappresenta l'architettura fisica del progetto che raggruppa diversi componenti. Di seguito è data una descrizione per ciascun componente.

- Google Chrome: è il browser web utilizzato per lo sviluppo di questo progetto.

- GlobalContext: è il componente React utilizzato per memorizzare all'interno del progetto la modalità di visualizzazione del sito, in base al valore di carbon intensity ricevuto da Electricity Maps.

- React components: sono i componenti React che compongono il sito web. Essi aggiornano il loro design in base alla modalità di visualizzazione impostata dall'utente, o in automatico se la modalità scelta è quella live.

- CarbonButton: è il componente utilizzato per la creazione di un bottone che permette di modificare la modalità di visualizzazione del sito (live, lowCarbon, mediumCarbon, highCarbon), in base al valore di carbon intensity ricevuto da Electricity Maps.

- CarbonIntensity: è la funzione adibita a effettuare la richiesta GET all'API di Electricity Maps tramite l'utilizzo della libreria Axios. La funzione gestisce anche eventuali errori che potrebbero verificarsi durante la richiesta HTTP.

- Electicity Maps: è l'API che viene richiamata dalla funzione `CarbonIntensity()` per ricevere i dati di carbon intenisty.

4. Progettazione

4.1 Casi d'uso

Attori:

- utente;
- react App;
- Electricity Maps.

Analisi casi d'uso:

- visualizzazione sito web:
    1. l'utente richiede il sito web da visualizzare;
    2. l'applicazione react invia una richiesta GET a Electricity Maps;
    3. Electricity Maps restituisce il valore di carbon intensity;
    4. l'applicazione react calcola il valore di displayMode;
    5. l'applicazione react carica la pagina web nella modalità calcolata;
    6. l'utente richiede una nuova modalità di visualizzazione;
    7. l'applicazione react invia una richiesta GET a Electricity Maps;
    8. Electricity Maps restituisce il valore di carbon intensity;
    9. l'applicazione react calcola il valore di `displayMode`;
    10. l'applicazione react aggiorna i componenti da visualizzare.

4.2 Diagramma dei componenti

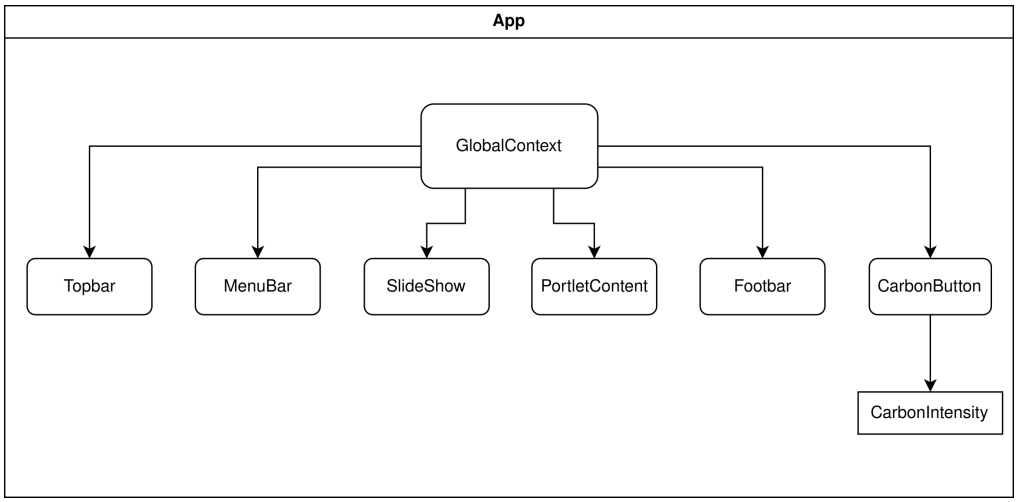

Figura 4.1: Diagramma dei componenti

Nel diagramma dei componenti 4.1 sono rappresentati tutti i componenti realizzati per il funzionamento del sito web e i loro collegamenti. Tutti i componenti sono collegati al componente `GlobalContext`, poiché all'interno di quest'ultimo è presente la variabile di stato `displayMode`, la quale tiene traccia della modalità di visualizzazione del sito, utilizzata da tutti i componenti per aggiornare il loro design. Inoltre, è rappresentata anche la funzione `CarbonIntensity()` richiamata dal componente `CarbonButton`.

4.3 Diagramma di sequenza

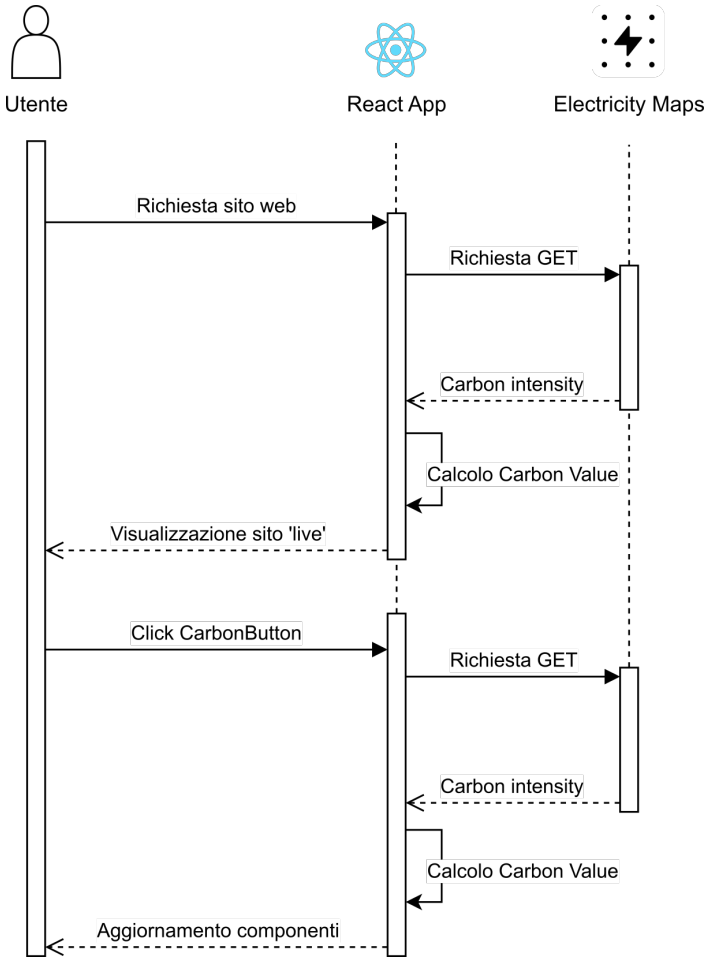

Figura 4.2: Diagramma di sequenza

Nel diagramma di sequenza 4.2 è rappresentato il flusso logico che permette la visualizzazione del sito web. Il flusso parte dall'utente, il quale richiede la visualizzazione del sito, e successivamente quest'ultimo si occupa di caricare i componenti che andranno a formare la pagina web, ed eventualmente aggiornare il loro design, dopo aver richiesto i dati di carbon intensity a Electricity Maps.

5. Sviluppo

5.1 Creazione del progetto React

Per la realizzazzione di questa applicazione è stato utilizzato il gestore di pacchetti NPM di Node.js. Tramite questo strumento è stato eseguito il comando `npx create-react-app nome-progetto` da terminale, che permette di creare un progetto React da zero. Tale comando genera la struttura base del progetto, inclusi i file e le directory essenziali per iniziare a sviluppare il sito, come ad esempio il file `App.js`. Questo file rappresenta il componente "padre" all'interno del quale verranno inseriti tutti gli altri componenti.

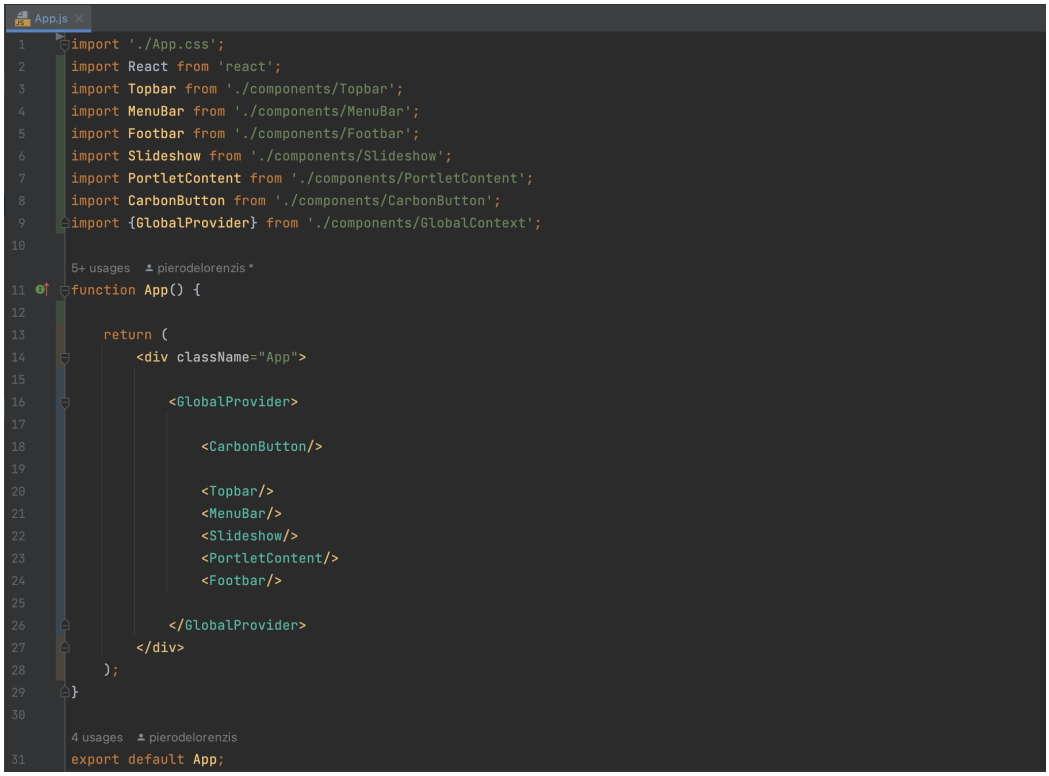

Figura 5.1: Componente App.js

5.2 Implementazione del GlobalContext

Per il corretto funzionamento del sito è stato realizzato il componente denominato `GlobalContext`, utile alla memorizzazione della variabile di stato `displayMode`, la quale tiene traccia della modalità di visualizzazione del sito web scelta dall'utente. A tale scopo, è stata utilizzata la funzione `createContext()` offerta dalla libreria di React. Questa funzione permette di creare un *contesto* che funge da contenitore centrale per la variabile di stato `displayMode`. Successivamente, è stato definito un Provider, `GlobalContext.Provider`, che ha il compito di fornire il valore di `displayMode` a tutti i componenti figli. Infine, il provider è stato richiamato nel componente padre `App.js`, e al suo interno sono stati inseriti tutti i componenti realizzati per la creazione del sito web.

Figura 5.2: Componente GlobalContext.js

5.3 Creazione della funzione CarbonIntensity

Per effettuare le richieste GET all'API di Electricity Maps è stata creata una funzione apposita chiamata `CarbonIntensity()`. All'interno di questa funzione è stato inserito l'endpoint al quale effettuare la richiesta per ricevere i dati di carbon intensity relativi alla zona `IT` (Italia). Per effettuare tale richiesta è stata utilizzata la libreria Axios. Essendo la richiesta GET una chiamata autenticata è stato configurato l'header[1] con una `key` fornita da Elettricity Maps, necessaria per l'autenticazione. Inoltre, in caso di errore durante la richiesta è stata prevista una stampa in console dell'eventuale errore.

Figura 5.3: Funzione CarbonIntensity.js

[1]Intestazione contenente ulteriori informazioni sulla richiesta HTTP da inviare.

5.4 Implementazione del CarbonButton

Dopo la creazione del `GlobalContext` e della funzione `CarbonIntesity` è stato implementato il componente `CarbonButton`. Questo componente ha lo scopo principale di creare un bottone utile all'utente per scegliere la modalità di visualizzazione del sito web. Per fare ciò, all'interno del componente sono state create quattro funzioni, `handleLiveClick()`, `lowCarbon()`, `mediumCarbon()`, `highCarbon()`, corrispondenti alle quattro possibili modalità di visualizzazione.

La funzione `handleLiveClick()` è legata alla modalità di visualizzazione LIVE. Quando la funzione viene richiamata, effettua una richiesta asincrona alla funzione `CarbonIntensity()`, quindi attende che quest'ultima completi la chiamata e restituisca il valore di carbon intensity ricevuto da Electricity Maps. Successivamente, il codice verifica il valore di `carbonIntensity`. Se quest'ultimo è maggiore di 400 viene impostato lo stato `displayMode` su `highCarbon`, tra 250 e 400 viene impostato su `mediumCarbon`, altrinmenti su `lowCarbon`.

Le altre tre funzioni citate in precedenza impostano semplicemente lo stato di `displayMode` rispettivamente su `lowCarbon`, `mediumCarbon` e `highCarbon`.

Queste quattro funzioni vengono richiamate alla pressione di quattro pulsanti che compongono il componente `CarbonButton`.

Infine, è importante notare che questo componente presenta una funzione `useEffect()`, esposta dalla libreria di React, utilizzata per effettuare la prima chiamata alla funzione `CarbonIntesity()` quando viene caricata la pagina web per la prima volta, richiamando al suo interno la funzione `handleLiveClick()`. I valori 400 e 250 utilizzati per la modifica della variabile di stato `displayMode` sono stati scelti dopo un'analisi dello storico dei valori di carbon intensity della zona `IT`, visualizzabili sul sito di Electricity Maps.

CarbonButton.js

```
import React, {useContext} from 'react';
import {useState, useEffect} from 'react';
import './Style.scss';
import GlobalContext from './GlobalContext';
import CarbonIntensity from "./CarbonIntensity";
3 usages
const CarbonButton = () => {

    const {displayMode, setDisplayMode} = useContext(GlobalContext);

    // Effettua la chiamata iniziale al caricamento della pagina
    useEffect( effect: () => {
        handleLiveClick()
    }, deps: []);

    2 usages
    const handleLiveClick = async () => {
        CarbonIntensity().then((carbonIntensity) => {
            if (carbonIntensity > 400) {
                setDisplayMode('highCarbon');
            } else if (carbonIntensity > 250) {
                setDisplayMode('mediumCarbon');
            } else {
                setDisplayMode('lowCarbon');
            }
        });
        console.log(displayMode);
    };
```

CarbonButton.js

```
    const lowCarbon = () => {
        setDisplayMode('lowCarbon')
        console.log(displayMode);
    }
    1 usage
    const mediumCarbon = () => {
        setDisplayMode('mediumCarbon')
        console.log(displayMode);
    }
    1 usage
    const highCarbon = () => {
        setDisplayMode('highCarbon')
        console.log(displayMode);
    }
```

```
    const [menuVisible, setMenuVisible] = useState( initialState: false);
    2 usages
    const toggleMenu = () => {
        setMenuVisible(!menuVisible);
    };
    return (
        <div>
            <button className='carbon-button' onClick={toggleMenu}>
                <svg xmlns="http://www.w3.org/2000/svg" width="16" height="16" fill="currentColor">
                    <path d="M11.251.068a.5.5 0 0 1 .227.58L9.677 6.5H13a.5.5 0 0 1 .364.843l-8 8.5a.5.5 0 0 1-.842-.49L
                </svg>
            </button>
            {menuVisible && (
                <div className={`menu ${menuVisible ? 'visible' : ''}`}>
                    <div className='close'>
                        <button className='menu-close' onClick={toggleMenu}>x</button>
                    </div>
                    <p className='mb-0'>CARBON INTENSITY</p>
                    <button className='menu-item-live' onClick={handleLiveClick}>LIVE</button>
                    <button className='menu-item' onClick={lowCarbon}>LOW</button>
                    <button className='menu-item' onClick={mediumCarbon}>MEDIUM</button>
                    <button className='menu-item' onClick={highCarbon}>HIGH</button>
                </div>
            )}
        </div>
    );
};
2 usages
export default CarbonButton;
```

Figura 5.4: Componente CarbonButton.js

5.5 Creazione componenti grafici e funzionali

Per testare il funzionamento dell'intero progetto si è scelto di creare un mockup[2] del sito dell'ateneo Unisalento. Dopo un'analisi generale del sito di riferimento, consultabile al link `www.unisalento.it`, si è proceduto alla creazione dei componenti principali che formano la home page del sito. In ordine sono stati creati `Topbar`, `MenuBar`, `Slideshow`, `PortletContent`, `Footbar`. Per la creazione dei componenti grafici, come colonne e righe, è stato utilizzato il framework Bootstrap. L'installazione di tale framework è stata eseguita lanciando da terminale il comando `npm install react-bootstrap bootstrap`. Per la creazione dello slideshow di immagini che è presente nel sito è stata utilizzata la libreria `react-responsive-carousel`, installandola con il comando `npm install react-responsive-carousel`, e successivamente è stato inserito il componente `Carousel` all'interno del file `Slideshow.js`.

Per la modifica dinamica del design dei componenti sono state utilizzate diverse logiche.

Innanzitutto si è scelto di importare dinamicamente le immagini presenti nei componenti per permettere all'intero sito web di scaricare solo i media necessari.

Per eseguire questa logica all'interno dei componenenti in cui sono presenti delle immagini sono state create due funzioni: `switchToImage1()` e `switchToImage2()`.

Queste sono delle funzioni asincrone che, tramite l'operatore `await`, attendono il caricamento dei moduli delle immagini attraverso il comando `import`. Dopo aver importato ciascun modulo di immagine, la funzione utilizza il comando `setCurrentImage()` per impostare i valori delle variabili di stato

[2]Simulazione visuale di come dovrebbe apparire l'elemento o il sistema finale.

`currentImage`, che saranno poi richiamate nel parametro `src` degli elementi HTML `<img>`. In particolare, la funzione `switchToImage1()` carica le immagini ad alta definizione visualizzabili nella modalità `lowCarbon`, mentre la funzione `switchToImage2()` importa le immagini a bassa risoluzione visualizzabili nella modalità `mediumCarbon`.

Infine, tramite uno `useEffect` di React sono richiamate le due funzioni in base al valore della varibile di stato `displayMode`, accessibile a tutti i componenti tramite il componente `GlobalContext`.

Per la modalità di visualizzazione `highCarbon` si è scelto di non mostrare nessuna immagine; viene invece visualizzato il parametro `alt` presente negli elementi `<img>`. Nel componente `Slideshow` si è scelto di nascondere completamente il `Carousel` per una migliore fruizione della pagina web.

Grazie a questo approccio, durante il caricamento del sito web, sono richiesti e scaricati solo i media necessari per la visualizzazione della pagina web nella modalità scelta dall'utente.

Di seguito sono riportati alcuni dei componenti creati.

MenuBar.js

```
import React, {useContext} from 'react';
import './Style.scss';
import GlobalContext from './GlobalContext';

3 usages
const MenuBar = () => {

    const {displayMode} = useContext(GlobalContext);

    const menuItems = [
        {label: 'ATENEO', link: '/home'},
        {label: 'DIDATTICA', link: '/about'},
        {label: 'RICERCA', link: '/services'},
        {label: 'TERZA MISSIONE', link: '/contact'},
        {label: 'INTERNATIONAL', link: '/blog'},
    ];

    return (
        <div className={`custom-row ${displayMode}`}>
            {menuItems.map((item : {label: string, link: string} , index : number ) => (
                <a key={index} href={item.link} className="custom-column">
                    {item.label}
                </a>
            ))}
        </div>
    );
};

2 usages
export default MenuBar;
```

Figura 5.5: Componente MenuBar.js

Slideshow.js

```jsx
import React, {useContext, useEffect} from 'react';
import {Carousel} from 'react-responsive-carousel';
import 'react-responsive-carousel/lib/styles/carousel.min.css';
import './Style.scss';
import GlobalContext from './GlobalContext';

3 usages
const Slideshow = () => {

    const {displayMode} = useContext(GlobalContext);

    const [currentImage1, setCurrentImage1] = React.useState( initialState: null);
    const [currentImage2, setCurrentImage2] = React.useState( initialState: null);
    const [currentImage3, setCurrentImage3] = React.useState( initialState: null);
    const [currentImage4, setCurrentImage4] = React.useState( initialState: null);

    1 usage
    const switchToImage1 = async () => {
        const imageModule1 = await import('../images/cesine_scavo.jpg');
        const imageModule2 = await import('../images/master_23-24.jpg');
        const imageModule3 = await import('../images/portale_studenti.jpg');
        const imageModule4 = await import('../images/studenti_dallalto.jpeg');
        setCurrentImage1(imageModule1.default);
        setCurrentImage2(imageModule2.default);
        setCurrentImage3(imageModule3.default);
        setCurrentImage4(imageModule4.default);
    };
    1 usage
    const switchToImage2 = async () => {
        const imageModule1 = await import('../images/cesine_scavo_low.jpg');
        const imageModule2 = await import('../images/master_23-24_low.jpg');
        const imageModule3 = await import('../images/portale_studenti_low.jpg');
        const imageModule4 = await import('../images/studenti_dallalto_low.jpg');
        setCurrentImage1(imageModule1.default);
        setCurrentImage2(imageModule2.default);
        setCurrentImage3(imageModule3.default);
        setCurrentImage4(imageModule4.default);
    };

    useEffect( effect: () => {
        // La logica che determina quale funzione chiamare in base al valore di carbon intensity
        if (displayMode === 'mediumCarbon') {
            switchToImage2();
        } else if (displayMode === 'lowCarbon') {
            switchToImage1();
        }
    }, deps: [displayMode]);

    if (displayMode === 'highCarbon') {
        return null;
    }

    return (
            <div className={`carosello`}>
                <Carousel
                    showArrows={false} // Nasconde le frecce di scorrimento
                    showThumbs={false} // Nasconde le miniature
                    showStatus={false} // Nasconde lo stato del carosello
                    infiniteLoop={true} // Attiva il looping infinito
                    autoPlay={true} // Avvia lo slideshow automaticamente
                    interval={7000} // Intervallo di tempo tra le immagini (in millisecondi)
                >
                    <div>
                        <img src={currentImage1} alt="Immagine 1"/>
                    </div>
                    <div>
                        <img src={currentImage2} alt="Immagine 2"/>
                    </div>
                    <div>
                        <img src={currentImage3} alt="Immagine 3"/>
                    </div>
                    <div>
                        <img src={currentImage4} alt="Immagine 4"/>
                    </div>
                </Carousel>
            </div>
    );
};
2 usages
export default Slideshow;
```

Figura 5.6: Componente Slideshow.js

5.6 Creazione foglio di stile

Durante la creazione dei componenti è stato anche scritto il foglio di stile SCSS, `style.scss`, nel quale sono state inserite tutte le impostazioni grafiche di ogni elemento utilizzato all'interno dei componenti.

Inoltre, si è scelto di applicare un filtro di colorazione in scala di grigi ad alcuni elementi del sito, attivo solo nella modalità di visualizzazione `highCarbon`. Ciò è stato fatto per sensibilizzare l'utente al cambiamento di design del sito web.

All'interno del foglio di stile sono state inserite anche le logiche che permettono al sito web di essere responsive, e quindi di adattarsi a tutti i dispositivi utilizzati per la sua fruizione. Questa logica è stata implementata tramite l'utilizzo del parametro `@media`, il quale permette di specificare le modifiche da apportare ai componenti grafici quando si ha una variazione della dimensione del dispositivo.

Di seguito sono riportati alcuni esempi dell'implementazione di queste logiche all'interno del foglio di stile.

```
78  /* Stile per la colonna destra contenente le icone e lo sfondo a strisce */
79  .topbar-right {
80    display: flex;
81    align-items: center;
82    justify-content: flex-end;
83
84    &.highCarbon {
85      background-image: url('../images/bkg-line_low.png');
86    }
87
88    &.mediumCarbon {
89      background-image: url('../images/bkg-line_low.png');
90    }
91
92    &.lowCarbon {
93      background-image: url('../images/bkg-line.png');
94    }
95  }
```

Figura 5.7: Esempio stile dinamico.js

```scss
/* MenuBar */
.custom-row {
  display: flex;
  justify-content: space-between;
  width: 1170px;
  height: 24px;
  margin: 0 auto;

  &.highCarbon {
    filter: grayscale(100%);
  }
}

@media (max-width: 1199px) {
  .custom-row {
    width: 956px;
  }
}

@media (max-width: 991px) {
  .custom-row {
    width: 736px;
  }
}

@media (max-width: 768px) {
  .custom-row {
    display: none;
  }
}
```

Figura 5.8: Esempio filtro e @media

6. Validazione sperimentale

6.1 Test funzionale

Per tesatare il funzionamento del sito web, sono state eseguite una serie di prove per verificare che le tre modalità di visualizzazione restituiscano effettivamente un design diverso e adattabile al dispsitivo.

Di seguito sono mostrate delle instantanee del sito web.

Figura 6.1: Sito web - Modalità LOW

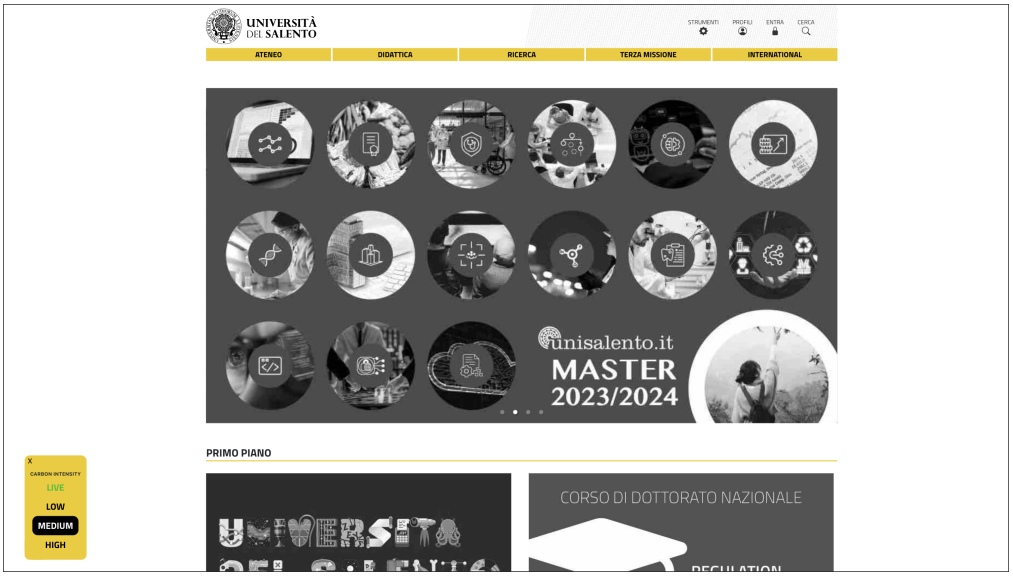

Figura 6.2: Sito web - Modalità MEDIUM

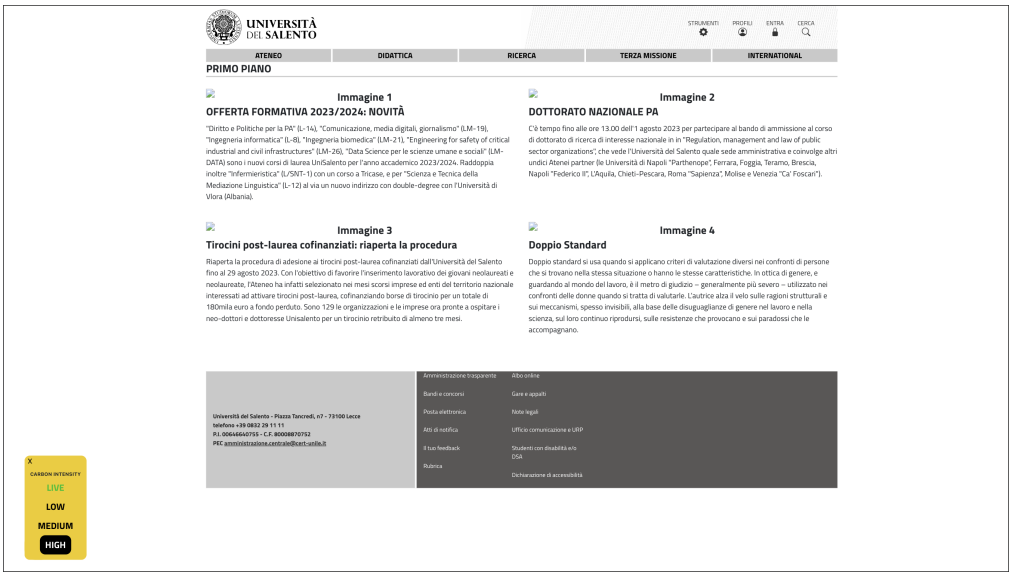

Figura 6.3: Sito web - Modalità HIGH

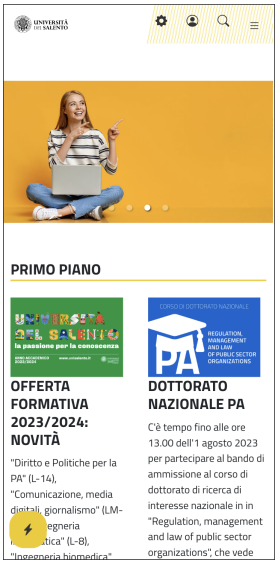

Figura 6.4: Sito web mobile

6.2 Sperimentazione

Al fine di verificare il reale risparmio di CO_2 nelle diverse modalità, si è scelto di calcolare la quantità di dati trasferiti, attraverso la rete, durante la visualizzazione del sito. Questa misurazione è stata effettuata mediante l'utilizzo dello strumento DevTools, incluso nel browser Google Chrome. In particolare, nella sezione denominata Rete di quest'ultimo, è possibile accedere a un riepilogo riguardante la quantità di dati richiesti dal sito web al server (in questo caso, lo stesso computer utilizzato per condurre i test), al fine di visualizzare tutti i componenti del sito.

Per conferire autenticità ai dati presentati da Google Chrome, si è provveduto a eliminare la cache del browser dopo ciascuna visualizzazione del sito in una specifica modalità, in modo da svuotare la memoria del dispositivo utilizzato per la navigazione, poiché i browser, dopo ogni visualizzazione di un sito web, conservano una serie di dati, quali ad esempio le immagini. Ciò avviene al fine di evitare un nuovo download[1] dei medesimi dati dal server che ospita

[1]Processo di ricezione di dati o informazioni da un server remoto o da una fonte esterna e salvataggio di questi ultimi sul proprio dispositivo o sistema locale.

il sito, contribuendo in tal modo a ottimizzare il processo di caricamento della pagina web e renderlo più veloce.

Di seguito sono riportati i dati visualizzati all'interno dello strumento `DevTools` di Google Chrome, da cui sono stati estratti i valori relativi ai dati trasferiti sulla rete (seconda colonna della tabella).

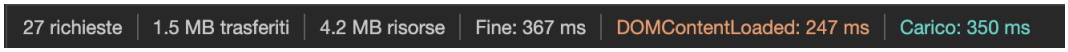

Figura 6.5: Dati trasferiti - Modalità LOW

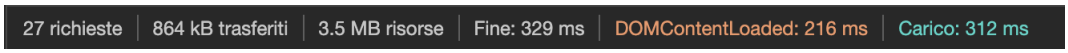

Figura 6.6: Dati trasferiti - Modalità MEDIUM

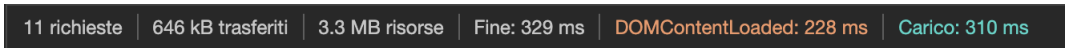

Figura 6.7: Dati trasferiti - Modalità HIGH

A questo punto è stato possibile calcolare l'effettiva CO_2 risparmiata effettuando degli appositi calcoli.

I dati utilizzati per il calcolo sono i seguenti:

- $TransferDataLowMode = 1,5$ MB $\rightarrow$ 1536 kB
- $TransferDataMediumMode = 864$ kB
- $TransferDataHighMode = 646$ kB
- $CarbonIntensity = 374$ g/kWh
- $EnergyIntensity$[1]=0,025 kWh/GB $\rightarrow = 0,000000023841858$ kWh/kB

Si ricorda, come descritto in precedenza, che il valore di *carbonIntensity* è stato ottenuto calcolando la media dei valori riportati nello storico di carbon intenity, relativamente alla zona 'IT' (Italia).

A partire da questi dati sono stati calcolati i kWh totali consumati nelle tre modalità di visualizzazione:

$$kWhTot$$

Low Mode: $EnergyIntensity \cdot TransferDataLowMode = 0,000036$ kWh

Medium Mode: $EnergyIntensity \cdot TransferDataMediumMode = 0,000020$ kWh

High Mode: $EnergyIntensity \cdot TransferDataHighMode = 0,000015$ kWh

Dopo aver calcolato i kWh totali è stato possibile calcolare gli effettivi grammi di CO_2 consumati:

Low Mode: $kWhTot \cdot CarbonIntensity = 0,013$ g

Medium Mode: $kWhTot \cdot CarbonIntensity = 0,0074$ g

High Mode: $kWhTot \cdot CarbonIntensity = 0,0056$ g

Questi dati si riferiscono ai grammi di CO_2 consumati da una singola visualizzazione del sito web.

In seguito a questi calcoli si è voluto fare una proiezione su larga scala dei risultati ottenuti.

Utilizzando il sito `www.similarweb.com` sono stati prelevati i dati riguardanti le visualizzazioni del sito dell'ateneo Unisalento, di cui se ne riportano i valori.

- Visualizzazioni totali (Periodo giugno 2023 - agosto 2023) = 1568000
- Media pagine per visualizzazione = 5,31

Utilizzando quindi i dati raccolti si è calcolata la quantità totale di CO_2, ipotizzando che i valori siano uguali durante tutto l'arco dell'anno.

- Low Mode (per trimestre): $0,013$ g $\cdot 1568000 \cdot 5,31 = 104,56$ kg
- Low Mode (per anno): $0,013$ g $\cdot 6272000 \cdot 5,31 = 432,95$ kg
- Medium Mode (per trimestre): $0,0074$ g $\cdot 1568000 \cdot 5,31 = 59,52$ kg

- Medium Mode (per anno): $0,0074 \text{ g} \cdot 6272000 \cdot 5,31 = 246,45 \text{ kg}$
- High Mode (per trimestre): $0,0056 \text{ g} \cdot 1568000 \cdot 5,31 = 46,62 \text{ kg}$
- High Mode (per anno): $0,0056 \text{ g} \cdot 6272000 \cdot 5,31 = 186,50 \text{ kg}$

Dai seguenti dati si evince che tra la modalità MEDIUM e la modalità LOW si ha un risparmio effettivo di:

- $45,04$ kg trimestrali
- $186,5$ kg annuali

Mentre tra la modalià HIGH e la modalità LOW si ha un risparmio effettivo di:

- $57,94$ kg trimestrali
- $246,45$ kg annuali

Pertanto, dai risultati ottenuti si può dimostrare che tra la modalità MEDIUM e la modalità LOW si ha un risparmio del $43,07\%$, mentre tra la modalità HIGH e la modalità LOW si ha un risparmio del $56,92\%$.

7. Conclusioni e Sviluppi futuri

Questo progetto ha raggiunto i suoi obiettivi proponendo una soluzione in grado di migliorare l'efficienza energetica e ridurre le emissioni di CO2 dei siti web.

In particolare si è implementato un algoritmo, tramite l'utilizzo della libreria React, in grado di adattare in modo dinamico l'interfaccia di siti web.

Per testare la funzionalità dell'algoritmo si è realizzato un sito web utilizzando come riferimento il sito dell'ateneo Unisalento.

Infine si è valutato l'impatto ambientale della soluzione proposta, avvalendosi dello strumento DevTool e della piattaforma `www.similarweb.com`.

Grazie a questa simulazione accurata si è dimostrato che il risparmio di CO_2 può arrivare fino a 186,50 Kg ipotizzando all'incirca 6 milioni di visualizzazioni annuali.

Lo studio quindi dimostra che l'uso di tecnologie avanzate può aiutare l'ambiente riducendo le emissioni di gas serra.

Il progetto ha anche individuato alcuni possibili sviluppi futuri per migliorare l'algoritmo implementato, come ad esempio:

- l'utilizzo dei filitri colore per risparimare energia nei dispositivi che utilizzano display AMOLED;
- la possibilità di modificare dinamicamente il valore di soglia di carbon intensity per le tre modalità di visualizzazione in modo da avere un valore più accurato.

- l'implementazione dell'algoritmo proposto sull'intero sito web di riferimento, e non solo sulla home page come fatto durante la sperimentazione, per valutare il reale risparmio di CO_2.

Elenco delle figure

Bibliografia

[1] IEA (2020), The carbon footprint of streaming video: fact-checking the headlines, IEA, Paris https://www.iea.org/commentaries/the-carbon-footprint-of-streaming-video-fact-checking-the-headlines

[2] Cruz, L., & Abreu, R. (2019). Catalog of energy patterns for mobile applications. Empirical Software Engineering, 24, 2209-2235.

[3] W3C (2023), Web Sustainability Guidelines (WSG) 1.0, TPAC 2023, Spain https://w3c.github.io/sustyweb/