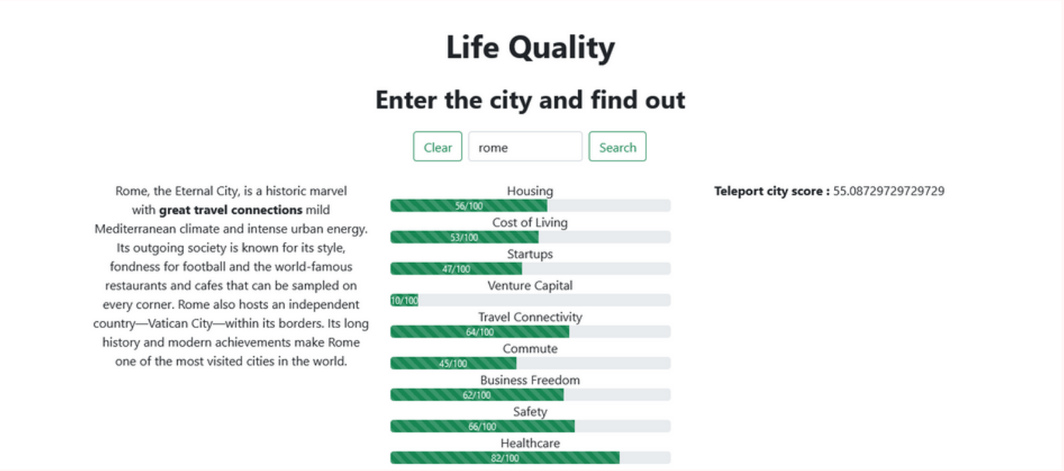
Progetto di Javascript Advanced



1. Descrizione del progetto
2. Spiegazione visiva della pagina
3. Suddivisione dei fogli html,css e js
4. Scelte logiche del codice js
5. Conclusioni personali

Super Guida di Start 2 Impact

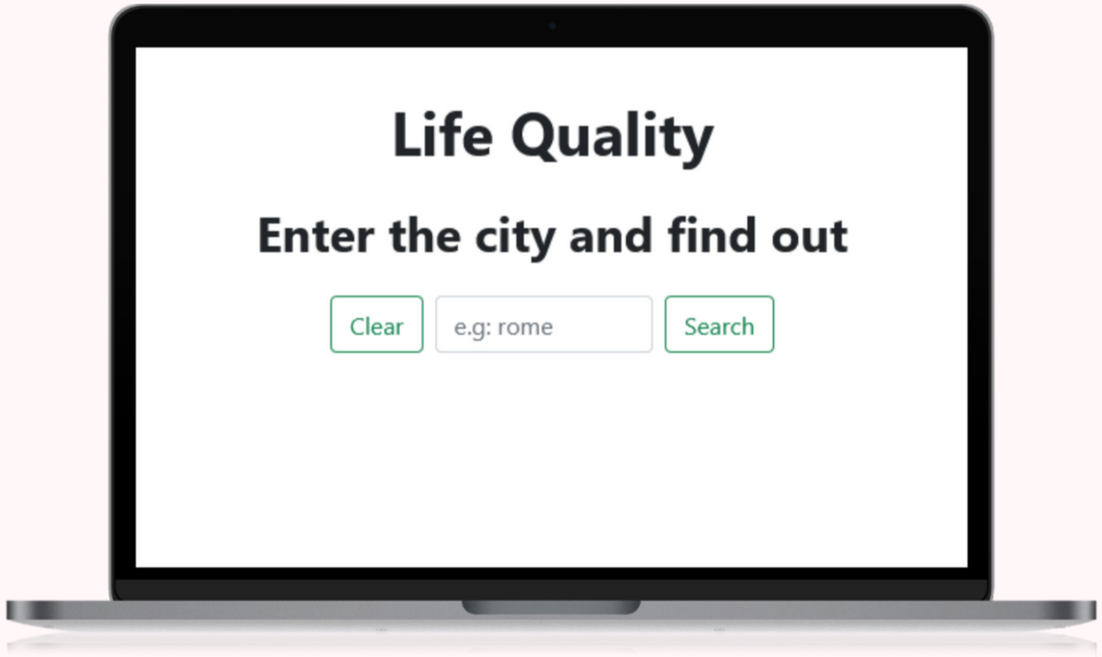
Descrizione del progetto

Questa applicazione fornisce informazioni relative alla qualità della vita nelle città. Utilizzando le API messe a disposizione da Teleport permette di ottenere una descrizione generale della città presa in questione, un voto per ogni parametro utile a fare un bilancio della qualità della vita ed un voto medio generale relativo alla città.

Ho scelto questo progetto tra i vari proposti, poichè amo molto viaggiare e realizzare un applicazione che possa essere utile ad un viaggiatore nella scelta della prossima meta, mi ha stimolato fin da subito.

1- Sezione di ricerca

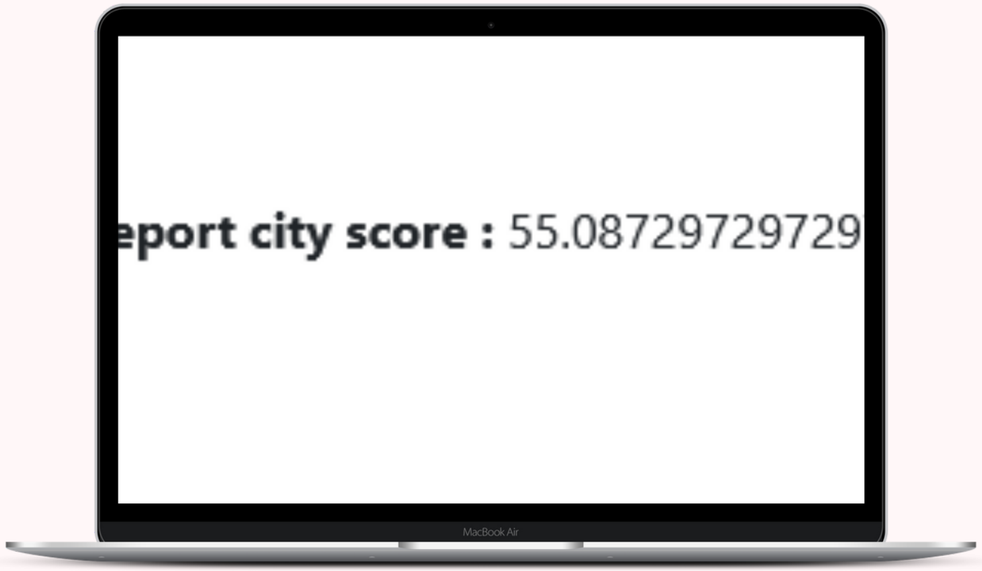
Appena si apre la pagina, questo è ciò che viene mostrato all'utente, un interfaccia molto semplice, composta da un titolo, una breve frase sottostante che funge da mini descrizione e anche incoraggiamento per l'utente a provare il servizio.



Le sezioni utili sono l'input di ricerca al centro, dove inserire il nome della città e ai suoi lati rispettavamente a destra e a sinistra, un bottone 'Search' per far partire la funzione principale ovvero quella di ricerca; Un bottone 'Clear' che permette di cancellare i risultati ottenuti, pulendo la pagina e ricaricandola.

2- Risultati della ricerca

Una volta cliccato il bottone di ricerca, ecco come vengono mostrati i risultati ottenuti, tre colonne in cui la prima è occupata dalla descrizione, la seconda da una 'pagella' di voti, la terza dal voto medio generale della città.



Il design della pagina è totalmente responsive e di conseguenza fruibile da tutti i dispositivi

Suddivisione dei fogli di codice

Il foglio index.html ho cercato di tenerlo il più pulito possiibile, anche in modo tale da poter sviluppare tutto in js. Il foglio contiente solo i vari link alle librerie (axios,jquery e bootstrap) e il src che rimanda al foglio di stile css.

index.html

```
<!DOCTYPE html>
<html lang="en" dir="ltr">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css" rel="stylesheet"
      <link rel="stylesheet" href="assets/CSS/styles.css">
    <title>EF - Life Quality</title>
     <link rel="icon" type="image/x-icon" href="assets/IMG/Web/fav3.png">
    <script src="https://unpkg.com/axios/dist/axios.min.js"></script>
    <script type = "text/javascript"
     src = "http://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js"></script>
  </head>
  <body id="body">
<div id="cont">

</div>

  <script src="assets/JS/dom.js"></script>
  <script src="assets/JS/main.js"></script>
  </body>
</html>

```

Suddivisione dei fogli di codice

Il css contiene poche righe di codice, la grande parte del lavoro la svolge bootstrap, qui ho dato solo un po di margini, dimensioni e disposizioni dei vari elementi.

index.html | styles.css | main.js

```css
.container-fluid{
  display: flex;
  justify-content: center;
  max-width: 20rem;
}

nav.navbar:nth-child(1){
  display: flex;
  justify-content: center;
}

.row{
  margin-top: 1rem;
  justify-content: center;
  text-align: center
}

.para{
  justify-content: center;
  text-align: center;
}


#city{
  margin-left: 0.48rem;
}
```

Suddivisione dei fogli di codice

Il primo foglio di js l'ho chiamato dom.js poichè l'ho utilizzato per creare tutti gli elementi statici che normalmente avrei implementato nel foglio html.

index.html | styles.css | main.js | dom.js

```
//creo la navbar di bootstrap
const nav = document.createElement("nav");
nav.classList.add("navbar","bg-transparent");
const cont = document.getElementById("cont");
cont.appendChild(nav);

//div principale
const mainDiv = document.createElement("div");
mainDiv.style.textAlign = "center";
nav.appendChild(mainDiv);

//h1
const h1 = document.createElement("h1");
h1.classList.add("h1-responsive","font-weight-bold","text-center","my-4");
h1.style.fontWeight = "bold";
mainDiv.appendChild(h1);
const testo = document.createTextNode("Life Quality");
h1.append(testo);

//div2
const div2 = document.createElement("div");
div2.style.textAlign = "center";
mainDiv.appendChild(div2);

//h2
const h2 = document.createElement("h2");
h2.classList.add("h1-responsive","font-weight-bold","text-center","my-4");
h2.style.fontWeight = "bold";
div2.appendChild(h2);
const testo2 = document.createTextNode("Enter the city and find out");
h2.append(testo2);
```

Suddivisione dei fogli di codice

Il secondo foglio JavaScript, 'main.js' contiente tutta la parte viva che permette all'applicazione di girare, ovvero mandare la richiesta a teleport e mandare a schermo i risultati.

index.html | styles.css | main.js

```
//dichiaro tutti gli elementi da intercettare
const city = document.getElementById("city");
const search = document.getElementById("search");
const clear = document.getElementById("clear");
const sum = document.getElementById("sum");
const categories = document.getElementById("cat");
const score = document.getElementById("score");

//apro la funzione per mandare la richiesta alle API di teleport
const api = async function(city){
  const request = await axios
  .get("https://api.teleport.org/api/urban_areas/slug:"+city+"/scores/")
  .then(response => response);
  console.log(request);

//descrizione città
  const summary = (request.data.summary);
  console.log(summary);
  sum.insertAdjacentHTML ("afterbegin", summary);

//score di ogni categoria della città
  const sectionCategories = request.data.categories.forEach(element =>{
    let section = document.createElement("section");
    section.classList.add("section_categories");
    categories.appendChild(section);
    let create = document.createElement("div");
    create.classList.add("categories");
    section.appendChild(create);

```

Scelte logiche di Javascript

DOM : per costruire gli elementi html, essendo la prima volta, li ho prima creati nel foglio html scegliendoli dalla libreria di bootstrap, dopo averli sviluppati e avergli dato la disposizione e lo stile che avevo in mente, sono passato al foglio 'main.js' per iniziare tutti i tentativi fino a che non avesse funzionato tutto correttamente.
Successivamente, quando ormai l'applicazione svolgeva tutte le funzioni richieste, ho cancellato gli elementi dal foglio html, lasciando solo un div vuoto da cui partire e mano mano li ho replicati ma implementandoli con javascript nel foglio 'dom.js'.
Ho trovato un po' di difficoltà nel creare elementi sfruttando i nomi delle classi di Bootstrap , ma devo dire che invece mi è venuto più semplice sfruttare i vari append() ed appendChild() poichè nella Super Guida erano stati trattati a sufficienza. La logica per svillupparli ognuno di questi è stata dichiarare l'elemento e allo stesso tempo crearlo, assegnarli la classe giusta e poi appenderli al div corrispondente.

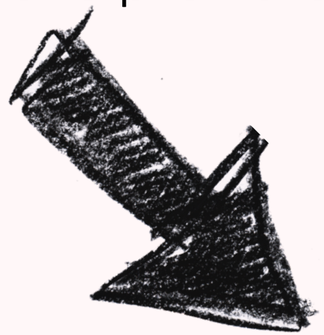
Scelte logiche di Javascript

MAIN : per sviluppare il codice attivo dell'app ho iniziato dichiarando tutti gli elementi da intercettare prendendoli dal dom tramite l'id assegnato, quindi, ho aperto la funzione 'api' facendo partire la richiesta utilizzando il metodo GET di axios. Seguendo i nomi degli elementi da mandare a schermo prendendoli dal json di riferimento di Teleport ho dichiarato il summary, le categories e il teleport city score, il passaggio più complesso per me è stato quello delle categories cercando un metodo che per ogni figlio dell'elemento venisse creato un titolo, una sezione, il valore e la barra in cui rappresentare quest'ultimo, dopo un po' di ricerche e tentativi tutto ha iniziato passo passo a funzionare. Infine, i due bottoni : search e clear, in modo tale che il primo mandi la richiesta e restituisca ciò che ottiene e inoltre che cambiando città non mandi a schermo la nuova ricerca sotto ai risultati precedenti ma piuttosto creare una nuova pagina, il secondo bottone ha la funzione di cancellare il campo di input, la pagina e ricaricare il tutto.

Conclusioni personali

Questo progetto senza mezzi termini mi ha inizialmente mandato totalmente nel pallone, una volta letta la consegna ho avuto un vuoto totale, probabilmente dovuto alla novità rispetto ai progetti passati che ho trovato nettamente più semplici e mettendomi in discussione anche dovuto magari ad uno studio teorico magari un po' approssimativo. Dopo un po' di "panico" iniziale , ho iniziato a rispolverare un po' i moduli della guida teorica attinenti all'applicazione pratica di questo progetto, ma ammetto senza successo. Ho trovato molto utile invece, effettuare molte ricerche esterne trovando tutorial, guide, esempi pratici che mi hanno dato un'idea abbastanza consistente di come muovermi, ho sfruttato anche il gruppo degli studenti su discord, che in parte mi hanno dato supporto.

Messi da parte i timori iniziali ho iniziato a buttare giù righe e righe di codice tentando, riuscendo, fallendo, ritentando e così via. Quando ho iniziato a vedere i primi risultati, ho ricevuto una spinta ad immergermi totalmente nel codice e questa motivazione mi ha accompagnato fino alla fine, ho sperimentato nuovi approcci di studio, di scrittura del codice e soprattutto una bella lezione per me stesso, spesso abituato a mollare di fronte alle prime difficoltà mettendo tutto in discussione cercando di scaricare la responsabilità su altri fattori. Sono contento di aver svolto questo progetto e di esserlo riuscito a concludere in meno tempo rispetto a quello che avevo previsto.



Link al progetto : https://github.com/emanuelefotia/Javascript-Advanced-Teleport-Life-Quality-EF
Link all'applicazione : https://ef-life-quality.netlify.app/

-Emanuele Fotia