

# Documentazione Progetto AppWeb - Gervasi Mariani

## INDICE

- 1) Componenti del gruppo
- 2) Configurazione WebStorm
  - a) Come avviare la BUILD
- 3) API
- 4) Strumenti utilizzati
- 5) Layout
- 6) Scelte progettuali
  - a) CurrentLetter
  - b) LoadMoreCocktail
  - c) Funzione di ricerca
  - d) Routing
- 7) Com'è strutturato il progetto
  - a) View
  - b) Components
    - i) Grid e table
    - ii) DrinkDetails
- 8) Problematiche riscontrate
  - a) Navigazione dei drink by id
- 9) Sviluppi futuri
  - a) Navigazione nei drink details by name
  - b) Filtraggio (per ingrediente)
  - c) Top ten cliccabile

## COMPONENTI DEL GRUPPO & REPOSITORY

Gervasi Alessandro - 866140

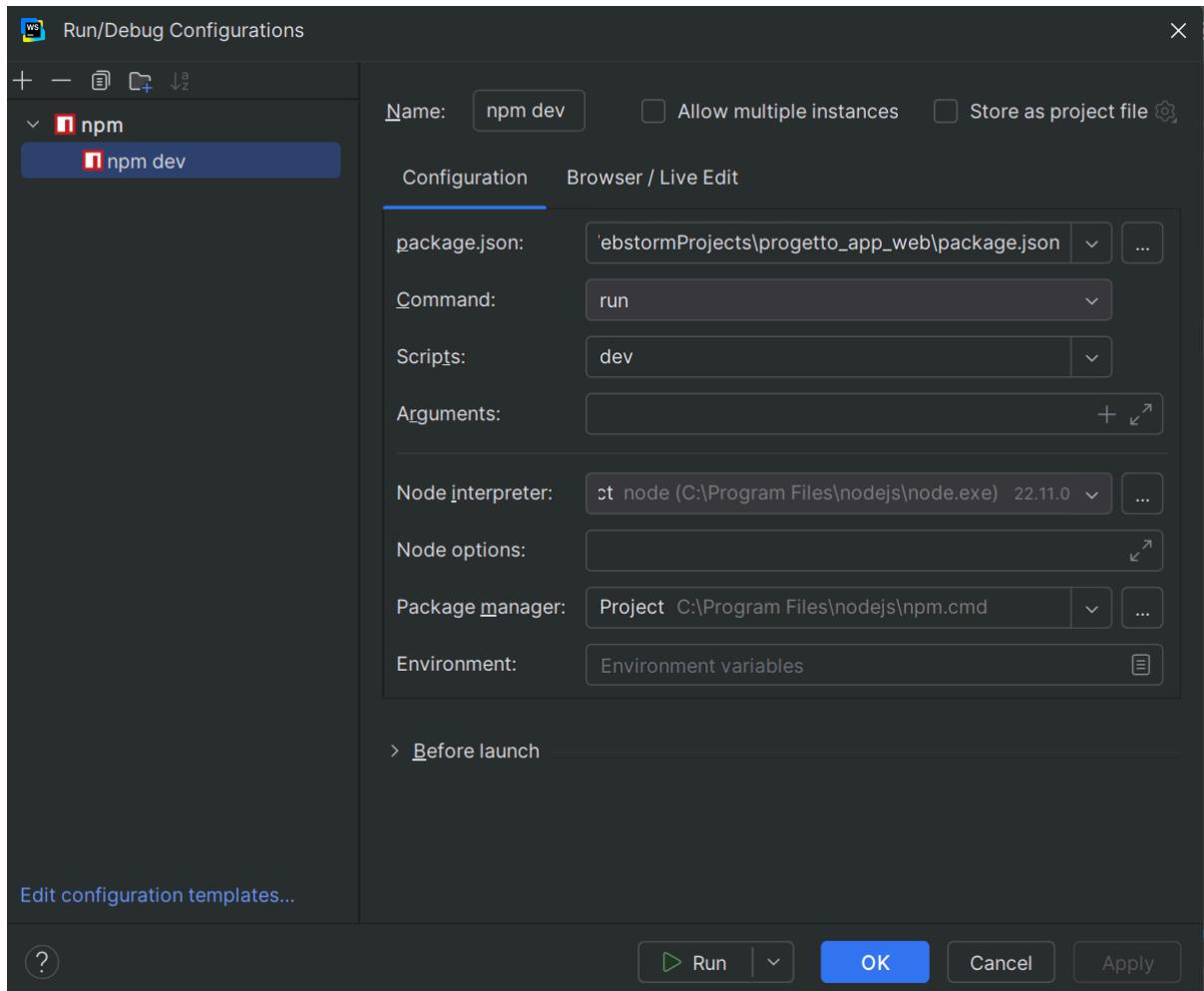
Mariani Michela - 866056

[Repository GitHub](#)

## CONFIGURAZIONE

Dopo aver clonato la repository basterà installare le dipendenze e aggiungere una nuova configurazione.

- install dependencies (run npm install)
- add new configuration (npm) ->



## Come avviare la BUILD

Per prima cosa bisognerà creare la build tramite il comando: npm run build. In questo modo verrà creata una cartella dist.

Per eseguirla basterà da terminale:

1. Installare (se non è già presente): npm install -g serve
2. Avviare il server nella cartella dist: serve -s dist

## SCELTA DELL'API

Per lo sviluppo di questo progetto abbiamo utilizzato l'API TheCocktailDB, accessibile a questo [link](#), in quanto ci sembrava in linea con i requisiti richiesti e offriva un database ben strutturato di cocktail e ingredienti.

Inoltre, l'API supporta ricerche avanzate, consentendo di filtrare i cocktail per nome, ingrediente, categoria o tipo di bicchiere. Abbiamo cercato di sfruttare queste funzionalità per realizzare al meglio il nostro progetto.

L'uso di questa API ci ha permesso di integrare facilmente un'ampia gamma di ricette di cocktail, offrendoci diverse immagini e dettagli. Grazie a queste caratteristiche, TheCocktailDB si è rivelata una scelta ideale per il nostro scopo.

**Funzionalità utilizzate:**

per nome= [www.thecocktaildb.com/api/json/v1/1/search.php?s=margarita](http://www.thecocktaildb.com/api/json/v1/1/search.php?s=margarita)

per id = [www.thecocktaildb.com/api/json/v1/1/lookup.php?i=11007](http://www.thecocktaildb.com/api/json/v1/1/lookup.php?i=11007)

per lettera = [www.thecocktaildb.com/api/json/v1/1/search.php?f=a](http://www.thecocktaildb.com/api/json/v1/1/search.php?f=a)

## STRUMENTI UTILIZZATI

- **Figma:** software utilizzato per creare la moodboard e i mockup;
- **Gemini:** AI utilizzata per generare l'immagine Hero;
- **WebStorm:** ide utilizzato per il progetto;
- **React-router-dom:** libreria utilizzata per gestire il Routing in React;
- **Bootstrap:** framework CSS utilizzato per la personalizzazione grafica;
- **Reactstrap:** libreria utilizzata per la navbar;
- **React-Icons:** libreria utilizzata per personalizzare l'icona del menù.

# LAYOUT

## Moodboard

Prima ancora di decidere come strutturare il nostro sito, abbiamo scelto di creare la moodboard in modo da avere già un'idea sullo stile e i colori da utilizzare. Come si può dedurre dal risultato finale, il mood scelto si ispira ai salotti Jazz, al lusso e al desiderio.



## Font e colori

Il font scelto per i titoli è *Warren*. Abbiamo selezionato questo font in quanto, seppur non immediato da leggere, ci sembrava molto caratteristico ed estroso; inoltre, abbiamo pensato che l'effetto finale fosse simile a quello dei quadri con le luci Neon che molto spesso sono presenti nei bar, con un tocco di personalità in più.



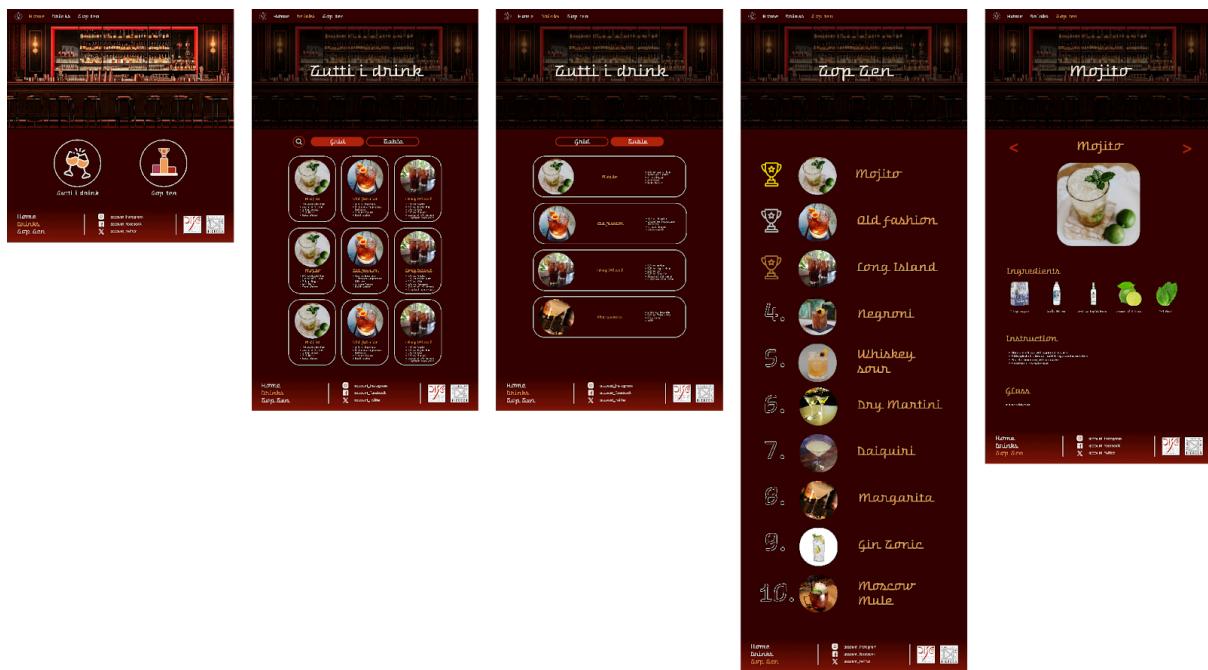
Un ragionamento simile è stato seguito nella scelta del font per il corpo, il Tilt Neon. Questo font, più essenziale, garantisce un'ottima leggibilità e chiarezza.

I colori scelti si ispirano in tutto e per tutto alla moodboard. In particolare sono stati utilizzati

- **#330102** per lo sfondo
- **#B0240A** per i bottoni e le icone
- **#F29F58** per i titoli e gli elementi attivi
- **#F8EDE2** per i testi

## Mockup

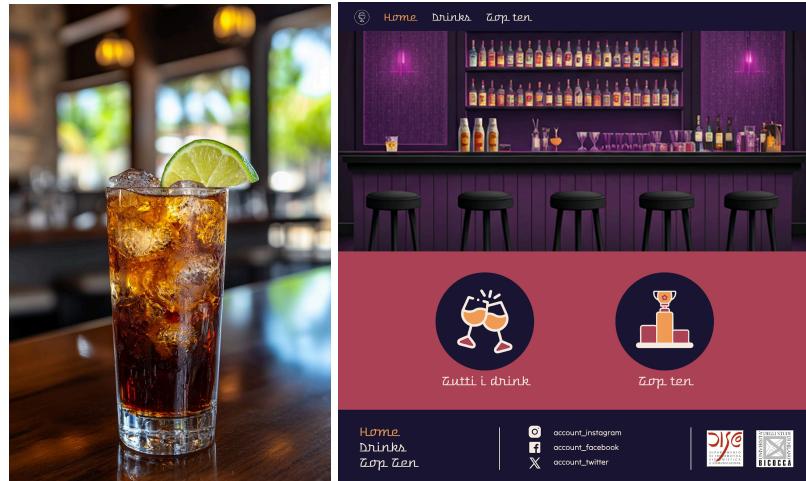
Una volta determinati il mood e i colori, siamo passati alla realizzazione del mockup tramite Figma, per assicurarci di avere una solida base da cui partire nella realizzazione effettiva del sito.



Alcune note grafiche che vogliamo sottolineare sono:

- **Il gradiente nel footer** -> scelta che si oppone al design flat e neumorfico in voga in questo periodo, ma comunque abbiamo deciso di inserire questo effetto perché ci ricordava molto la sfumatura di alcuni drink (come il Cuba Libre e il Long Island);
- **Colori** -> inizialmente avevamo valutato anche per delle sfumature di viola, colore associato al lusso e allo sfarzo, ma ci sembrava troppo pasteloso l'effetto finale per cui abbiamo optato per altro;
- **Hero** -> come Hero abbiamo scelto di farci generare una immagine da Gemini scrivendo il prompt a partire da immagini trovate online;

- **Responsive** -> il sito è completamente responsive, tuttavia non abbiamo realizzato i mockup di tutte le versioni perché non ci sembrava il focus principale di questo progetto.



Nota: purtroppo non siamo riusciti ad inserire tutte le funzionalità che avevamo pensato, ma questo verrà approfondito in una apposita sezione di questo documento.

# SCELTE PROGETTUALI

## Current Letter

Per la visualizzazione dei drink abbiamo pensato di disporli in ordine alfabetico tramite la funzionalità “per nome”. Quando termina la visualizzazione dei primi drink, appare un bottone “More Cocktails” che permette di caricare altri cocktails nella pagina.

```
const fetchCocktailsByLetter = async (currentLetter) : Promise<void> => { Show usages ↗ onlyMiki
    if (isSearching) return; // Evita il caricamento se è in corso una ricerca per nome

    setLoading( value: true);
    try {
        const response : Response = await fetch(
            input: `https://www.thecocktailedb.com/api/json/v1/1/search.php?f=${currentLetter}`
        );
        const data = await response.json();
        if (data.drinks) {
            setCocktails( value: (prevCocktails : any[]) => [...prevCocktails, ...data.drinks]);
        }
    } catch (error) {
        console.error("Errore nell'API:", error);
    }
    setLoading( value: false);
};
```

Caricamento dei drink per lettera corrente tramite fetch. Fetch chiama l’API, la gestione delle lettere avviene grazie a current letter, parametro che indica la lettera corrente.

```
// Funzione per caricare più cocktail passando alla lettera successiva
const loadMoreCocktails = () : void => { Show usages ↗ onlyMiki
    if (letter < 'z') {
        const nextLetter : string = String.fromCharCode( codes: letter.charCodeAt(0) + 1 );
        setLetter(nextLetter);
    }
};
```

LoadMoreCocktails è una funzione che permette di caricare altri cocktail. Questa viene invocata da un bottone presente alla fine della pagina.



## Funzione di ricerca

Abbiamo inserito anche la funzionalità di ricercare i cocktails per nome. Per effettuare la ricerca bisogna procedere schiacciando la lente di ingrandimento. Una volta fatto ciò si aprirà un campo di input in cui l'utente potrà scrivere il drink da ricercare. Per avviare la search basterà, in seguito, premere la scritta "Search" che effettuerà una chiamata all'API grazie alla funzionalità by name.



```
const fetchCocktailsByName = async () : Promise<void> => { Show usages ↗ onlyMiki +1
  if (inputVal.trim() === "") return;

  setLoading( value: true);
  setIsSearching( value: true);
  setCocktails( value: []);

  try {
    const response : Response = await fetch(
      input: `https://www.thecocktaildb.com/api/json/v1/1/search.php?s=${inputVal}`
    );
    const data = await response.json();
    if (data.drinks) {
      setCocktails(data.drinks);
    } else {
      setCocktails( value: []);
    }
  } catch (error) {
    console.error("Errore nell'API:", error);
  }
  setLoading( value: false);
};
```

FetchCocktailsByName è una funzione che richiama l'API considerando come parametro di ricerca inputVal, ovvero l'input inserito dall'utente. Questo andrà ad aggiungersi all'API in modo da completare il link. Il risultato della ricerca saranno i drink contenenti l'input dell'utente (quindi ad esempio se ci sono più Mojito verranno visualizzati tutti).



Nel caso in cui l'utente volesse ritornare alla lista dei drinks, basterà selezionare la freccia che appare al posto della lente di ingrandimento. A questo punto la pagina verrà ricaricata, grazie al comando `window.location.reload()`, all'interno della funzione `toggleVisibility`. Questa funzione viene richiamata al click del bottone della search (ovvero la lente di ingrandimento/freccia), se è presente la freccia la pagina viene ricaricata, altrimenti se è presente la lente di ingrandimento la pagina non viene ricaricata e il bottone diventa attivo cambiando da lente a freccia.

```
const toggleVisibility = () : void => { Show usages · onlyMiki
  if (isActive) {
    window.location.reload();
  } else {
    setHidden(!hidden);
    setActive( value: true);
  }
};
```

## Routing

Dopo aver effettuato l'import di “react-router-dom” abbiamo definito le rotte tramite i componenti `BrowserRouter`, `Routes` e `Route`.

```
<Routes>
  <Route path="/" element={<Home />} />
  <Route path="/drinks" element={<Drinks />} />
  <Route path="/topTen" element={<TopTen />} />
  <Route path="/drink/:id" element={<DrinkDetails />} />
</Routes>
```

Abbiamo definito un percorso per ogni pagina (URL).

```
const nav : [{exact: boolean, text: string...} = [
  {url: "/", text: "Home", exact: true},
  {url: "/drinks", text: "Drinks", exact: true},
  {url: "/topten", text: "TopTen", exact: true}
];
```

Questa lista l'abbiamo poi passata come props a MainTemplate. Abbiamo scelto di creare un componente unico che richiami Header e Footer in quanto tutte le nostre pagine hanno tali elementi.

```
const MainTemplate = (props) => { Show usages ▾ onlyMiki
  const {
    children,
    footerIgLink,
    footerIgName,
    footerFbLink,
    footerFbName,
    footerXLink,
    footerXName,
    navItems,
    logo
  } = props;
```

Infine abbiamo collegato la lista ad Header e Footer, passandola sempre come props.

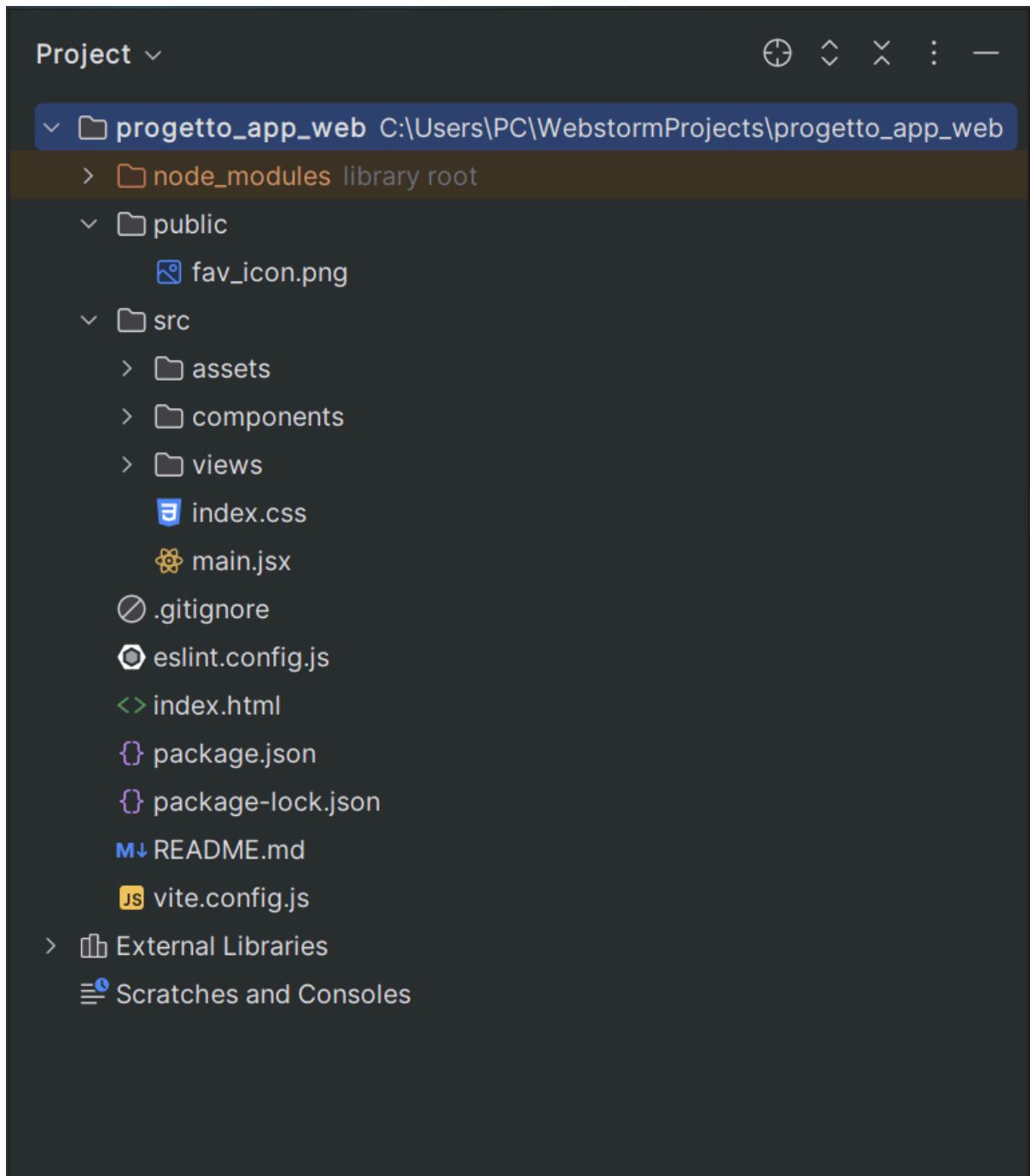
```
return(
  <>
    <Header
      logo={logo}
      navItems={navItems}
    />
    {children}
    <Footer
      igLink={footerIgLink}
      igName={footerIgName}
      fbLink={footerFbLink}
      fbName={footerFbName}
      xLink={footerXLink}
      xName={footerXName}
      navItems={navItems}
    />
  </>
);
```

A questo punto, al click di ogni item sarà possibile accedere alla corrispondente pagina, grazie a NavLink, un componente di react-router-dom.

# COM'È STRUTTURATO IL PROGETTO

Abbiamo suddiviso il progetto nelle seguenti cartelle:

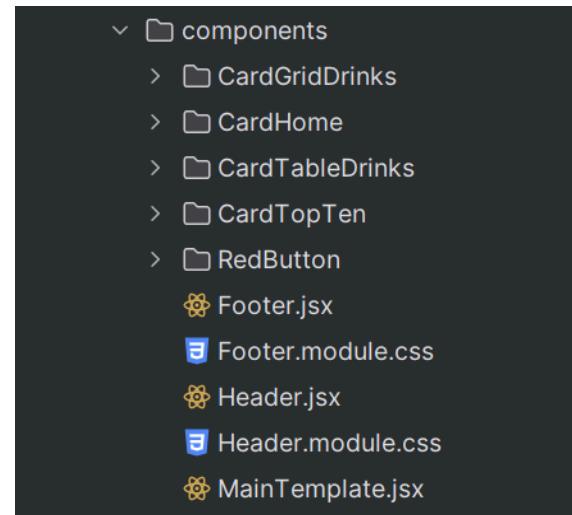
- **public**: contiene la fav\_icon;
- **src**: contiene ->
  - **assets**: in cui sono presenti le immagini e le icone utilizzate nel sito;
  - **components**: contiene tutti i componenti;
  - **views**: contiene le varie pagine visualizzabili



## Components

Sfruttando le funzionalità di React abbiamo scelto di realizzare i seguenti componenti:

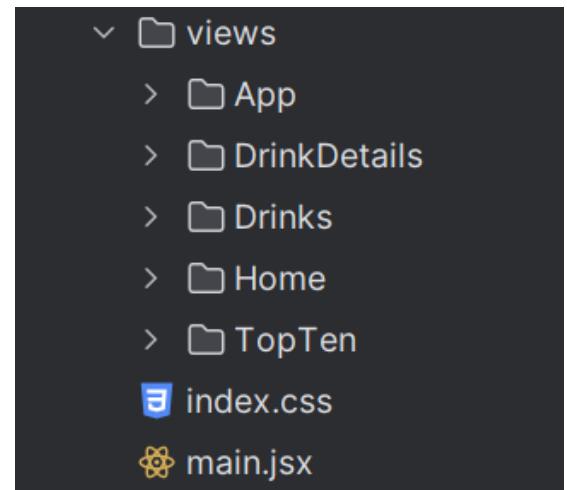
- **CardGridDrinks** e **CardTableDrinks**: utilizzati per la realizzazione della singola card del drink visualizzabile in modalità Grid o in modalità Table. Questo viene gestito all'interno della pagina Drinks.jsx;
- **CardHome**: utilizzato solamente nella Home, offre un collegamento diretto alle altre pagine del sito;
- **CardTopTen**: utilizzato solamente nella pagina Top Ten, descrive il singolo elemento. Questo viene poi richiamato dieci volte;
- **RedButton**: utilizzato per caricare più drink quando finisce la visualizzazione degli stessi nella schermata drink.



## Views

All'interno di Views sono presenti le singole pagine ovvero:

- **App**: è il componente principale dell'applicazione, funge da entry point e organizza la struttura dell'interfaccia utente;
- **DrinkDetails**: pagina interna, descrive il singolo Drink;
- **Drink**: pagina che permette la visualizzazione dei drink in modalità Grid o Table;
- **Home**: pagina iniziale, da una panoramica del contenuto del sito, presentando un collegamento diretto a Drink e Top Ten;
- **TopTen**: pagina che mostra i dieci drink più popolari.



## PROBLEMATICHE RISCONTRATE

### Navigare tra un drink e l'altro per id

La problematica principale in questo progetto è stata la navigazione interna tra i drink, quindi il passaggio da una pagina di DrinkDetails ad un'altra. La nostra idea iniziale era quella di permettere la navigazione tra un drink e l'altro tramite delle semplici frecce. Il problema è che avendo ordinato i drink in ordine alfabetico, e non in ordine di id, pur funzionando il tutto, non risultava coerente con l'ordine dato in Drinks (ovvero alfabetico).

Una soluzione che abbiamo pensato è stata quella di ordinare i drink per id, ma tramite un piccolo test utente, abbiamo compreso che l'ordine per id, per quanto avesse senso da un punto di vista del programmatore, per l'utente risultava insensato e scomodo. Ad esempio, se l'utente non volesse usare la funzione cerca, prevista nel nostro sito, per ritrovare un qualsiasi drink dovrebbe necessariamente sfogliare tutti i drink. Mettendo invece i drink in ordine alfabetico questo fatto non sussiste.

Di conseguenza, abbiamo preferito eliminare questa funzionalità, inserendo all'interno di DrinkDetails solamente una freccia che permette di ritornare a tutti i drink.

Codice usato per la navigazione tra una schermata DrinkDetails e l'altra.

```
//trovo il prossimo drink, controlla direzione (destra +1, sinistra -1)
//la direzione mi permette di incrementare o decrementare il nuovo drink
1. const findNextDrink = async (currentId, direction) => {
2.   let nextId = parseInt(currentId) + direction;
3.   while (nextId > 0) {
4.     try {
5.       const response = await
6.       fetch(`https://www.thecocktaildb.com/api/json/v1/1/lookup.php?i=${nextId}`);
7.       const data = await response.json();
8.       if (data.drinks) {
9.         navigate(`/drink/${nextId}`);
10.        return;
11.      }
12.    } catch (error) {
13.      console.error("Errore nell'API:", error);
14.    }
15.    nextId += direction;
16.  }
17.};
```

## SVILUPPI FUTURI

### Navigazione drink by name

Come già spiegato in precedenza, la navigazione dei drink avveniva tramite id. Permetteremo la navigazione tra DrinkDetails tramite name e lettera corrente.

### Opzioni di filtraggio avanzate

Attualmente è possibile ricercare un drink tramite la funzionalità “Search” che permette, inserendo una lettera o una parola, di ricercare gli elementi che presentano quell’input. Tuttavia, l’API permette di ricercare anche gli ingredienti e di filtrare anche in base all’alcolico, alla categoria e al bicchiere. Queste funzionalità non sono presenti nel nostro progetto, ma potrebbero essere implementate.

### TopTen interattiva

In questo momento la pagina TopTen è solamente una pagina di visualizzazione statica, vorremmo renderla interattiva mostrando i dettagli del drink selezionato.