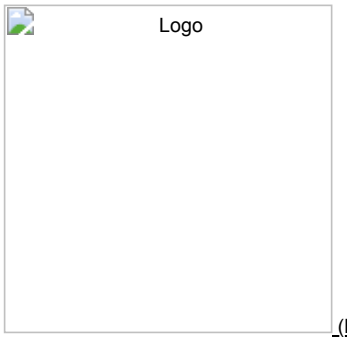


(https://github.com/othneildrew/Best-README-Template)

LookThatParty

Esame finale di Stefano Goffi

▶ Tabella dei contenuti

Informazioni

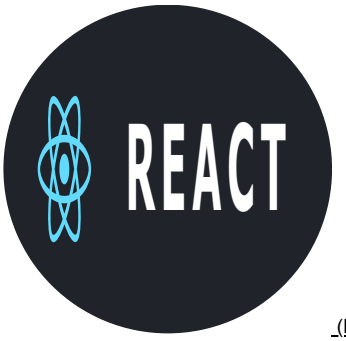
Il progetto LookThatParty è una piattaforma/applicazione basata su React che permette di visualiizare una lista di eventi e di guardarne i particolari e dettagli per proi prenotare quello più interessante per l'utente.

Perchè avete bisogno di LookThatParty:

- Facile e veloce
- Design minimal e dark mode
- Sempre aggiornato sugli eventi

Tecnologie Utilizzate

il progetto è costruito con React.js e Tailwind



(https://reactjs.org/)

(https://tailwindcss.com/)

Utilizzo

Per inizializzare il progetto aprire il terminale e digitare

- ```
  npm install
  ```

Avvio

Spostartsi nella cartella src e digitare i seguementi comandi
  ```

- `cd src`
- `npm run dev`

RootMap

```
|-- src/
| |-- components/
| | |-- EventCard.tsx
| | |-- EventCardDetail.tsx
| | |-- Dish.tsx
| | |-- Header.tsx
| | |-- NavBar.tsx
| | |-- Player.tsx
| | |-- Reservation.tsx
| | |-- Success.js
| | |-- ...
| |
| |-- hooks/
| | |-- useClubDetail.ts
| | |-- useClubs.ts
| |
| |-- logics/
| | |-- Logics.tsx
| |
| |-- pages/
| | |-- HomePage.tsx
| | |-- DetailsPage.tsx
| |
| |-- repo/
| | |-- event.types.ts
| | |-- index.tsx
| |
| |-- assets/
| | |-- img/
| | | |-- logo.png
| |
| | |-- gif/
| | | |-- header.gif
| |
| | |-- song/
| | | |-- quicksand.ogg
| |
| |-- App.scss
| |-- App.tsx
| |-- main.tsx
| |-- index.scss
| |-- ...
```

Dipendenze

`react-router-dom`

La libreria `react-router-dom` offre un sistema di routing per l'applicazione React, consentendo la navigazione tra le diverse pagine o viste senza dover ricaricare completamente la pagina.

- `Link`: Componente utilizzato per creare link tra diverse pagine nell'applicazione, facilitando la navigazione senza ricaricare la pagina.
- `HashLink`: Simile a `Link`, ma utilizzato specificamente per la navigazione all'interno della stessa pagina utilizzando hash.
- `useParams`: Hook che consente di accedere ai parametri definiti nell'URL delle rotte.

`react-router-dom` versione 6

Nella versione 6 di `react-router-dom`, sono introdotti nuovi componenti e funzioni per la gestione delle rotte.

- `createBrowserRouter`: Funzione utilizzata per creare un router personalizzato per l'applicazione. È utile quando si desidera estendere o personalizzare il comportamento del router di base.
- `RouterProvider`: Componente che fornisce il contesto del router a tutti i componenti figli, consentendo loro di accedere alle informazioni sulle rotte.
- `Navigate`: Componente utilizzato per navigare a una determinata rotta, offrendo funzionalità avanzate per la gestione della navigazione.

Roadmap

- [x] Aggiunto Readme
- [x] Implementata GIF nell'header
- [ ] Aggiunto Media Player
- [ ] Multi-lingua
  - [ ] Inglese
  - [ ] Francese
  - [ ] Tedesco
  - [ ] Cinese

Template

Template 1: `Card`

Descrizione

Template di una card per la visualizzazione della locandina, il nome dell'evento e i principali dati. Il template è implementato per ogni elemento di un array di risposta di una promise

Utilizzo

```
import { Link } from "react-router-dom";
import { HashLink } from "react-router-hash-link";
import { EventType } from "../repo/event.types";
import "./EventCard.scss";
import { Logics } from "../logics/Logics";
type ClubCardType = {
    club: EventType;
    goToDetail: string;
};
const Logica = new Logics();
const EventCard = ({ club, goToDetail }: ClubCardType) => {
    const {
        name,
        includedDrinks,
        tags,
        description,
        isAperitivoIncluded,
        dresscode,
        coverImage,
        date,
        price,
    } = club;
    return (...)
```

in questo caso card ha le dipendenze di react-dom-routing per la componentizzazzione, come parametro richiede la l'oggetto che contiene i dati che verrano usati nella card e un indice, utile per il routing alla pagina dettaglio

Template 2: `Details`

Descrizione

Template di dettaglio di una card, oltre ai dati utili per la card ci sono altri dati come description, un array di informazioni sull'evento e in ultima posizone il costo del biglieto

Utilizzo

```
import { EventDetailType } from "../repo/event.types";
import { Logics } from "../logics/Logics";
import "./EventCard.scss";
import Dish from "./Dish";
import NavBar from "./NavBar";
import Reservation from "./Reservation";

const Logica = new Logics();
type ClubCardDetailType = {
    club: EventDetailType;
};
const EventCardDetail = ({ club }: ClubCardDetailType) => {
    const {
        id,
        name,
        description,
        date,
        isAperitivoIncluded,
        tags,
        price,
        coverImage,
        dresscode,
        includedDrinks,
        includedDishes,
    } = club;
    return (...)
```

molto dimile a card, utilizza tipi differenti, uno più dettagliato dell'altro, insieme condividono le medesime dipendenze

Template 3: `Navbar`

Descrizione

template della navbar per la navigazione tra le pages, in questo caso il template è molto minimal è contiene giusto il logo e il nome del progetto

Utilizzo

```
import logo from "../assets/img/logo.png";
const NavBar = () => {
    return (
        <nav className="flex-no-wrap top-0 w-full z-10 bg-white border-gray-200 dark:bg-gray-900">
            <div className="max-w-screen-xl flex flex-wrap items-center justify-between mx-auto p-2">
                <a
                    href="/home"
                    className="flex items-center space-x-3 rtl:space-x-reverse"
                >
                    <img src={logo} className="h-9" alt="LookThatParty Logo" />
                    <span className="self-center text-2xl font-semibold whitespace-nowrap dark:text-white">
                        LookThatParty!
                    </span>
                </a>
            </div>
        </nav>
    );
};

export default NavBar;
```

Template 4: `Header`

Descrizione

template dell header, primissima cosa che vede l'utente appena entra sul progetto, un testo centrale con sotto una gif che richiama l'atmosfera del sabato sera

Utilizzo

```
import NavBar from "./NavBar";
import video from "../assets/gif/header.mp4";
import "./Header.scss";

const Header = () => {
    return (
        <>
            <header>
                <NavBar />
                <video autoPlay muted loop className="h-full w-full">
                    <source src={video} type="video/mp4" />
                    Il browser non supporta questa video
                </video>
                <div className="content">
                    <h1 className="big-title">LookThatParty!</h1>
                    <p>Prenota ora la tua serata perfetta□□□□</p>
                </div>
            </header>
        </>
    );
};

export default Header;
```

abbiamo l'import del video e della navbar, la quale non viene inserita nella homepage con l'header ma il primo è contenuto nel secondo, si è optato cosi per una correttezza sintassica e per ottimizare il codice lato SEO

Classe Logics

La classe `Logics` fornisce una serie di metodi utili per la formattazione delle date e la stilizzazione del testo. È progettata per essere utilizzata come un'utility per gestire operazioni comuni su dati e testo all'interno dell'applicazione.

Utilizzo

La classe `Logics` è progettata per essere utilizzata nel seguente modo:

date

Sottoclasse di Logics per gestire i metodi di formattazione della data

.formatt

Formatta una data in un formato specifico (gg/mm hh:mm)

```
import { Logics } from "./path/to/Logics";

// Creazione di un'istanza di Logics
const logic = new Logics();

// Utilizzo dei metodi di formattazione della data
const formattedDate = logic.date.formatt("2023-12-31T18:00:00");
console.log(formattedDate); // Output: "31 Dec 18:00"
```

.getHours

`formatt(date: string): string`: Formatta una data in un formato specifico (hh:mm)

```
import { Logics } from "./path/to/Logics";

// Creazione di un'istanza di Logics
const logic = new Logics();

// Utilizzo dei metodi di formattazione della data
const hours = logic.date.getHours("2023-12-31T18:00:00");
console.log(hours); // Output: "18:00"
```

.increment

Incrementa una data in base al numero di prenotazioni.

```
import { Logics } from "./path/to/Logics";

// Creazione di un'istanza di Logics
const logic = new Logics();

// Utilizzo dei metodi di formattazione della data
const incrementedDates = logic.date.increment("2023-12-31T18:00:00", 3);
console.log(incrementedDates); // Output: Array di date incrementate
```

style

Sottoclasse di Logics per gestire i metodi di stilizzazione del testo

.capitalize

Capitalizza la prima lettera di ogni parola in una stringa.

```
import { Logics } from "./path/to/Logics";

// Creazione di un'istanza di Logics
const logic = new Logics();

// Utilizzo dei metodi di stilizzazione del testo
const capitalizedText = logic.style.capitalize("hello world");
console.log(capitalizedText); // Output: "Hello World"
```

.allergenToIcon

Mappa un allergene a un'emoji corrispondente.

```
import { Logics } from "./path/to/Logics";

// Creazione di un'istanza di Logics
const logic = new Logics();

// Utilizzo dei metodi di stilizzazione del testo
const allergenIcon = logic.style.allergenToIcon("gluten");
console.log(allergenIcon); // Output: "□"
```

Contatti

Stefano Goffi - @Smoffi_pk (https://www.instagram.com/smoffi_pk/) - stefano.goffi@edu.itspiemonte.it

Github: https://github.com/StefanoGoffi (https://github.com/StefanoGoffi)