REACT

100% Operativo!

Guida Completa
Da zero alla realizzazione di una Web APP, in 24 ore con 5 giorni di accesso al VideoCorso Online di React

ing. Davide Copelli

Video-Corsi.com

Via Natisone 2 – 35010 Vigodarzere (PD)
Url: https://www.dcopelli.it/ – Email: web@dcopelli.it

Editore: Independently published
Redazione: Francesca e Chiara
Produzione: Davide C., Luca F., Giancarlo C.

Prima Edizione: marzo 2019
ISBN: 9781794464322

Stampato da Amazon.com

Indice

Capitolo 1

Introduzione all'applicazione NasdaqReact

1.1 Anteprima dell'applicazione

Per apprendere velocemente i concetti più importanti di React, svilupperemo un'applicazione chiamata "NasdaqReact", che puoi testare dal vivo a questo indirizzo: https://www.video-corsi.com/react/nasdaqreact/

Figura 1.1 – Interfaccia Utente (UI) dell'applicazione NasdaqReact

Se sei un esperto di mercato azionario, con pochi click scoprirai da solo a cosa serve e come funziona.

Viceversa, se anche tu appartieni a quella categoria di persone che ha sempre sentito

parlare di quotazioni in borsa al telegiornale ma non ci ha mai capito molto, è giunto il momento di provare a capire cosa succede ogni giorno nel mercato azionario più importante d'America per le società tecnologiche: il NASDAQ.

1.2 Funzionamento dell'applicazione

Lo scopo dell'applicazione è di analizzare in tempo reale l'andamento delle quotazioni delle più importanti società tecnologiche, come Amazon, Apple, Alphabet etc.

Tutti i dati da mostrare all'utente, dovranno essere recuperati da sorgenti esterne all'applicazione, e in più dovranno essere aggiornati in tempo reale.

Sfrutteremo un servizio gratuito, in grado di fornirci moltissime informazioni, che dovremo capire come scremare e rappresentare all'interno dell'interfaccia utente dell'applicazione.

Il tutto corredato non solo dalla visualizzazione del valore dell'azione e dalle variazioni percentuali, ma anche da un grafico, che permetterà all'utente di analizzare l'andamento nel tempo e capire meglio se è il momento di comprare oppure vendere.

Giusto per spiegare velocemente il funzionamento, come puoi vedere nella Fig. 1.1, ho un classico campo di ricerca in cui posso digitare le iniziali del nome dell'azione che intendo monitorare, ad esempio "Amaz", per individuare tutte le azioni la cui sigla inizia con "Amaz". Dopo aver cliccato il pulsante "Ok" è inviata la richiesta a un'API remota, che risponde inviando la lista di tutte le azioni trovate.

Da questa lista potrò decidere quali aggiungere nell'area a destra del campo di ricerca. Tra i dati visualizzati vi è la sigla dell'azione, il prezzo dell'ultima quotazione, il mercato di riferimento, l'ora, e le variazioni in percentuale e in punti rispetto al valore quotato nell'istante precedente.

Per ogni azione aggiunta, tramite un pulsante switch, posso iniziare a osservare in tempo reale i cambiamenti di prezzo dell'azione con la relativa ora in cui è avvenuta la quotazione, e la variazione sia in punti, sia in percentuale del prezzo.

In ogni istante inoltre posso decidere di visualizzare anche il grafico con rappresentato l'andamento del prezzo nel tempo. Grafico che si aggiornerà anch'esso in tempo reale, non appena l'azione cambia di prezzo.

Tramite il pulsante switch posso poi interrompere il monitoraggio ed eliminare l'azione dal pannello, cliccando sulla relativa icona rappresentativa della lettera "x".

Un'applicazione molto versatile, perché la logica che svilupperemo, potrà ad esempio, essere applicata per monitorare l'andamento della temperatura di un frigorifero "intelligente", collegato in rete con altri frigoriferi della mia ipotetica catena di cento ipermercati, e in grado di trasmettere questa informazione su un servizio sulle nuvole, accessibile tramite chiamate ad API dedicate.

Oppure potrei misurare il numero di visitatori all'interno di una specifica stanza di un museo, potendo così decidere eventuali azioni da fare per limitarne il flusso.

Insomma mille possibili utilizzi non appena riusciamo ad accedere a delle informazioni erogate in tempo reale, sia da archivi pubblici, sia privati.

Grazie alla particolare struttura dell'applicazione, avremo modo di affrontare tutti i più importanti concetti base e avanzati di React, potendoli applicare dal vivo e affrontando insieme tutte le possibili problematiche che la maggior parte degli sviluppatori di React prima o poi dovrà affrontare, con il vantaggio di poter vedere anche le possibili soluzioni e non solo la classica teoria.

In questo modo acquisirai velocemente la maggior parte delle competenze che ti serviranno per sviluppare applicazioni simili e via via più complesse.

1.3 Che cosa saprai fare al termine del libro?

Chi già mi conosce per aver seguito videocorsi online tramite la WebStartUniversity, sa che non mi piacciono le noiose e lunghe introduzioni che si leggono nei classici libri tecnici, su come deve essere usato il libro, la storia di un qualche linguaggio, del perché bisognerebbe impararlo, dei vantaggi, degli svantaggi e bla bla bla.

L'obiettivo infatti che mi sono prefissato durante la stesura del libro, è prima di tutto di non annoiarti con lunghi capitoli di tante parole e spiegazioni elementari che puoi trovare gratuitamente online in tutte le lingue, e secondo, di guidarti rapidamente (chi lavora o studia ha sempre poco tempo) nei meandri di React, in modo che al termine della lettura delle circa 200 pagine del libro, tu possa esclamare: - Bingo! Ora mi sento sicuro e ho finalmente capito come si può sviluppare una qualsiasi applicazione in React!

E' un obiettivo ambizioso perché non so quali siano le tue reali capacità, ma se fra 48 ore, ossia il tempo necessario a leggere le 200 e più pagine, non esclamerai questa frase, allora avrò miseramente fallito e la colpa sarà tutta mia (o quasi).

A parte gli scherzi, ormai sviluppo da 20 anni, quindi penso di conoscere a fondo le difficoltà che spesso chi è alle prime armi deve affrontare. Difficoltà che cercherò di superare con te accompagnandoti "per mano" nei punti più difficili.

Se ti perderai, potrai sempre scrivermi, o chattare con me, quindi non ti lascio scuse per lamentarti e "piagnucolare", come osservo in qualche commento online.

Se hai voglia d'imparare, e imparare fino in fondo, questo è il posto giusto per farlo: non sprecare l'opportunità. La mia disponibilità è garantita.

Fatta questa premessa, che assomiglia a quella che faccio a miei figli quasi ogni giorno, tralasciamo i classici capitoli che trovi nella maggior parte dei libri e vediamo subito come impostare l'ambiente di lavoro per iniziare a sviluppare la tua prima applicazione.

Capitolo 2

Impostare l'ambiente di lavoro

2.1 Installare Node.js

Per progettare le nostre prime applicazioni React, dobbiamo impostare l'ambiente di lavoro. Con ambiente di lavoro intendo l'insieme dei software e dei file e cartelle da predisporre nel proprio computer per lavorare con efficienza.

Fortunatamente è molto simile a quello che si utilizza in altre tecnologie e in altri framework.

Dovremo prima di tutto installare nel computer il software gratuito *Node.js* e conseguentemente *npm*. E' probabile che tu l'abbia già installato in passato se hai usato altri framework.

Puoi verificarlo subito aprendo la finestra terminale o prompt dei comandi e scrivendo questa riga (rispetta gli spazi):

```
node -v
```

Se compare un numero che indica una versione del software, allora significa che è già stato installato. Diversamente, dovrai accedere al sito *nodejs.org* e cliccare sul classico pulsante di download per scaricare il software e installarlo.

Questa chiaramente è la strada più semplice e conosciuta ma ne esistono altre.

Al termine, per verificare la corretta installazione apri nuovamente la finestra terminale e digita la riga precedente. Dovresti ora visualizzare un numero che indica la versione installata nel tuo computer.

Node.js è un software che si aggiorna molto rapidamente quindi eventualmente se lo hai già scaricato in passato e non conosci un sistema per tenerlo aggiornato, la

procedura più semplice è quella di rinominare la cartella “Node” installata in precedenza e rieseguire l'installazione del file scaricato.

2.2 Installare Visual Studio Code e le estensioni

L'altro pezzo importante che dobbiamo avere è un software per la scrittura del codice. Qui c’è proprio l’imbarazzo della scelta.

Il mio preferito è il “Visual Studio Code” anch'esso un software gratuito che puoi scaricare dal sito della Microsoft (https://code.visualstudio.com) oppure cercando la stringa “Visual Studio Code” in qualche motore di ricerca.

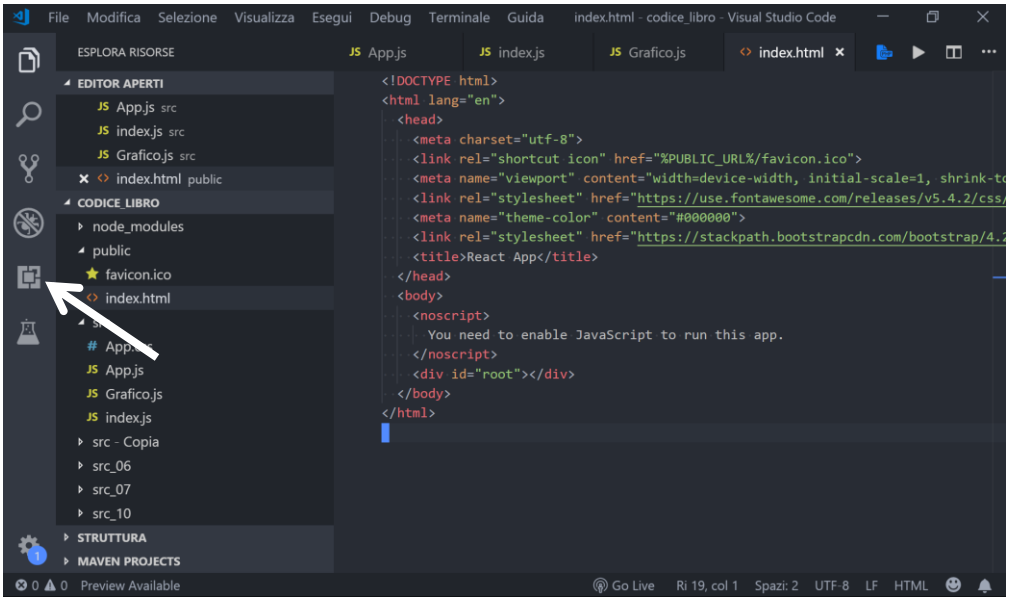

Figura 2.1 – Schermata di Visual Studio Code

Il vantaggio di questo software è che sono presenti molte estensioni che permettono di personalizzarne l’uso rendendolo più potente.

Potente soprattutto a livello di scrittura di codice, perché se clicchiamo sull’icona indicata in Fig. 2.1, si apre la finestra estensioni, grazie alla quale posso cercare tra una miriade di funzionalità da integrare nel software, ad esempio quelle per velocizzare la scrittura di blocchi standard di codice.

Se infatti cerchi la parola “React”, troverai decine e decine di estensioni create per questo e non solo.

Fai attenzione che c'è una differenza tra React e React Native. Quest'ultimo, è l'ambiente con cui si costruiscono applicazioni per i dispositivi mobili, sfruttando sempre React.
Una delle estensioni che ho installato io e che useremo qualche volta nel corso (non è obbligatoria), è quella che puoi vedere qui sotto, sviluppata dall'autore "dsznajder":

Figura 2.2 – Estensione per React creata da "dsznajder"

Nel corso degli anni ha subito numerose migliorie, ma usandola da un po' di tempo, devo dire che viene sufficientemente aggiornata in funzione di modifiche o nuove istruzioni introdotte in React.

Questa estensione ci permetterà di aggiungere diverse righe di codice digitando semplici abbreviazioni del tipo (imp, rce, est, e altre). La lista completa con il significato, la puoi vedere cliccando sulla voce dettagli.

Col tempo apprezzerai queste scorciatoie.

Per installare l'estensione sarà sufficiente selezionare quella d'interesse e cliccare sula voce "Installa". Al termine dovrai riavviare il software per confermare l'istallazione.

Altro strumento utile che ti consiglio di installare è quello che ci permetterà di visualizzare alcune informazioni sulla struttura della pagina che si genera con React. E' uno strumento che si installa all'interno del browser e si chiama "React Developer Tools".

E' disponibile sia per Chrome, sia per Firefox. Sarà sufficiente cercarlo all'interno di un qualsiasi motore di ricerca e una volta trovato, aggiungerlo alla lista di estensioni.

Figura 2.3 – Estensione “React Developer Toosl” per il browser Chrome

All'apertura nel browser di un'applicazione sviluppata in React, in alto a destra comparirà un'icona colorata con il simbolo di React che sfrutteremo nelle prossime lezioni per capire meglio la struttura di un’applicazione React.

Infine non devi dimenticare la guida ufficiale raggiungibile tramite il tramite link

https://www.reactjs.io

Come browser predefinito per testare le funzionalità che inseriremo durante lo sviluppo dell'applicazione, utilizzeremo Chrome, anche se gli stessi test possono essere fatti sfruttando un altro popolare browser come Firefox.

Questo termina il pacchetto di strumenti gratuiti da installare nel proprio computer per iniziare a scrivere le tue prime applicazioni React.

Capitolo 3

La struttura di un'applicazione React

3.1 Creare un progetto preconfigurato

React è una libreria JavaScript per la costruzione di componenti dell'interfaccia utente. A differenza di framework come Angular o Vue, React si occupa principalmente di gestire al meglio il livello Vista.

Avrai bisogno di librerie aggiuntive per gestire il flusso di dati in uscita e in entrata all'applicazione, il routing e la navigazione tra pagine, l'autenticazione etc.

L'applicazione che svilupperemo avrà bisogno solo di una parte di queste funzionalità, essendo un'applicazione a singola pagina "Single Page App" (SPA).

Sebbene per creare questo tipo d'applicazione sia sufficiente un singolo file HTML con una serie di librerie React, è utile avere una base da usare come punto di partenza.

In questo modo avremo un ambiente più strutturato rispetto al classico file HTML, e soprattutto già configurato con un server, in modo da poter elaborare e visualizzare in anteprima tutte le modifiche effettuate sul codice, oltre che a controllare in tempo reale eventuali errori prima ancora che l'applicazione venga mostrata nel browser.

Vediamo i passaggi per creare questa prima applicazione base preconfigurata.

La prima cosa da fare è creare una cartella di lavoro adibita a contenere tutti i tuoi progetti React. Dopo aver aperto la finestra di comando, dovrai spostarti in questa cartella con i soliti comandi DOS, o in alternativa selezionare la cartella in "Esplora risorse" e poi, tenendo premuto il tasto "maiuscolo" della tastiera, cliccare con il pulsante destro del mouse e selezionare la voce "Apri qui finestra di comando".

```
npx create-react-app nome_applicazione
```

Al posto di *nome_applicazione* dovrai sostituire il nome dell'applicazione che intendi sviluppare, il tutto senza inserire degli spazi nel nome e senza inserire lettere maiuscole.

Verrà creata una cartella apposita all'interno della quale saranno inseriti i diversi file che costituiranno un progetto React, con preimpostati tutti i comandi per visualizzare in anteprima sul browser l'applicazione e creare la versione di produzione.

Questa operazione può anche essere fatta direttamente all'interno di Visual Studio Code. Ipotizzando di aver già creato una cartella di lavoro di nome *CorsoReact* in una posizione qualsiasi del tuo computer (es. desktop), clicchiamo sul menu "File" e poi sulla voce "Apri Cartella", per selezionarla.

A questo punto, è possibile aprire la finestra terminale direttamente da Visual Studio Code tramite la combinazione di tasti CTRL+ò (lettera o accentata).

Sotto la schermate principale apparirà la finestra che vedi qui sotto:

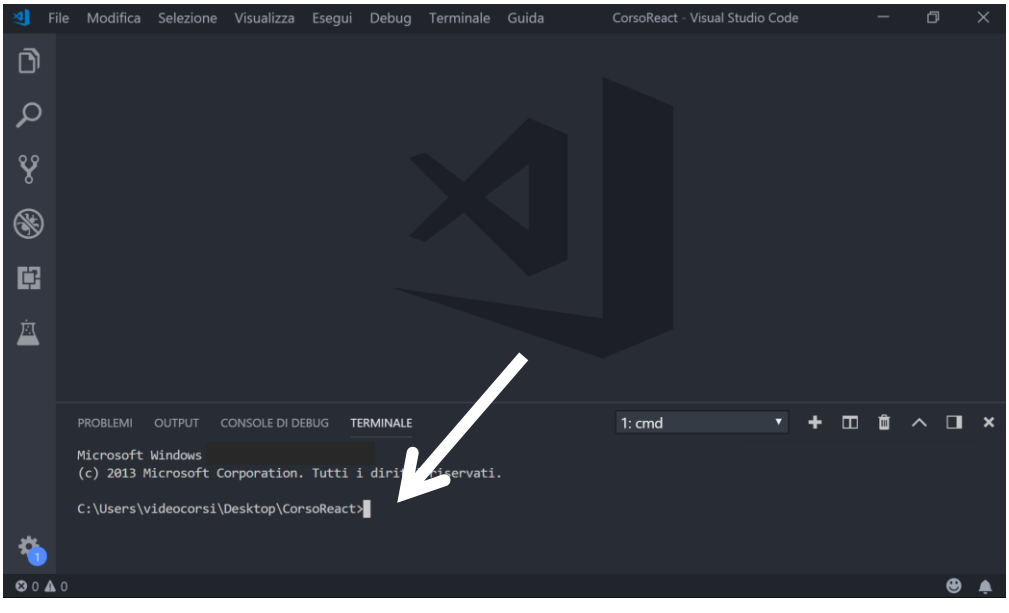

Figura 3.1 – Finestra Terminale interna a Visual Studio Code

su cui potrò digitare la riga vista in precedenza. Ipotizzando di scegliere *corsoreact* come nome dell'applicazione, scriveremo:

```
npx create-react-app corsoreact
```

Fai attenzione che questa operazione deve essere eseguita con una connessione ad Internet attiva, perché una parte dei file dovrà essere scaricata dalla rete.

3.2 I file che compongono un progetto React

Non appena termina l'esecuzione del comando precedente, se apriamo la cartella *corsoreact*, troveremo una lista di file e cartelle simile a questa:

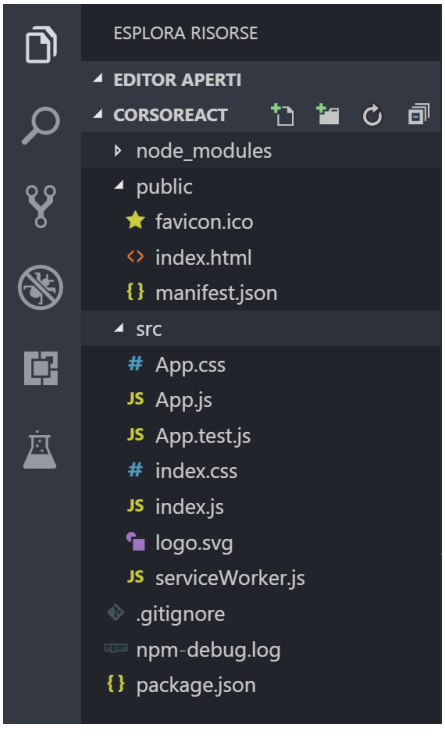

Figura 3.2 – Lista cartelle e file del progetto React preconfigurato

La cartella *node_module*, contiene una serie di librerie necessarie a far funzionare l'ambiente di test oltre che quelle specifiche per il funzionamento di un'applicazione React.

Poi troviamo una cartella *public* che è paragonabile alla classica cartella principale di un sito Web, in cui è presente un documento *index.html*, un'icona e un file *manifest.json*, necessario per lo sviluppo di Progressive Web App.

Infine vi è la cartella *src*, che contiene una serie di file CSS e JavaScript, e sulla quale andremo a lavorare per creare i diversi file JavaScript che costituiranno la nostra applicazione.

Entrando più in dettaglio, se apriamo il file *index.html*, presente all'interno della cartella src, la cosa più evidente che noteremo è l'assenza di codice HTML internamente al tag `body`, tranne un tag `div` con attributo `id` pari a `root`.

Una pagina simile visualizzata nel browser, sarebbe completamente bianca. In React invece è visualizzata in questo modo:

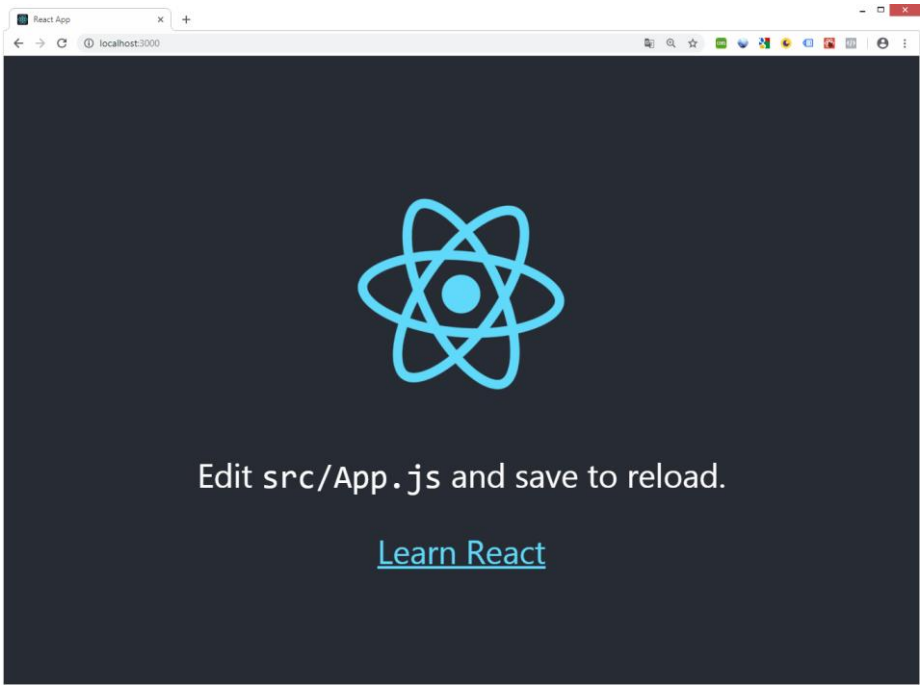

Figura 3.3 – Pagina principale applicazione React

Per verificarlo sul campo, ci avvaliamo dell'ambiente preconfigurato di test e nella finestra terminale, spostiamoci all'interno della cartella di lavoro *CorsoReact*, quindi all'interno di *NasdaqReact*, e (solo se sono su questa cartella) lanciamo la riga:

```
npm start
```

Questo comando aprirà "magicamente" in automatico la finestra del browser e caricherà per noi questo indirizzo http://localhost:3000, che ci permetterà di visualizzare proprio il contenuto di Fig.3.3.

Potrebbe venirti il dubbio che si tratti in realtà di un'altra pagina, rispetto al file *index.html* visto in precedenza. Se cliccassi sul pulsante destro del browser, per

visualizzare il codice sorgente, noteresti che si tratta dello stesso file, con la sola aggiunta di alcuni documenti JavaScript, non presenti nel file originale.
Questi sono i file che conterranno tutto il codice che impareremo a scrivere nei prossimi capitoli, compilato da Babel, affinché possa essere interpretato correttamente dal browser.

La domanda che potresti farti è da dove arrivi il codice che ci permette di visualizzare il logo rotante e le scritte sottostanti.

Qualche indizio lo possiamo ricavare sfruttando l'estensione del browser installata in precedenza nel Capitolo 1. Difatti cliccando sulla relativa icona, si apre una scheda che mostra l'insieme dei nodi rappresentativi dell'albero dei componenti di cui è costituita un'applicazione React.

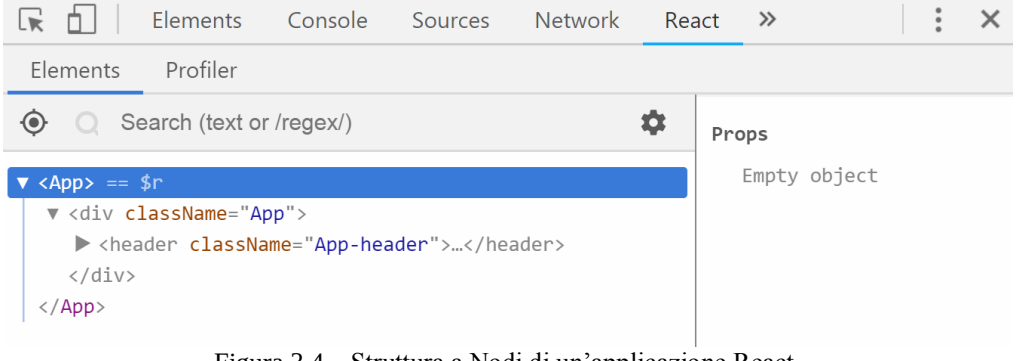

Figura 3.4 – Struttura a Nodi di un'applicazione React

Avremo modo di approfondire questi concetti nei prossimi capitoli ma la struttura di Fig. 3.4 ci aiuta a memorizzare un primo concetto importante di ogni applicazione React, ossia che ci deve sempre essere un solo nodo radice che serva da contenitore per tutti gli altri blocchi dell'applicazione.

Un po' come avviene per le pagine HTML con il tag `<body>`.

Il nodo radice dell'applicazione appena creata è il componente App (file *App.js*) che "qualcuno" è riuscito a iniettare nell'albero. E' proprio grazie ai file JavaScript visti in precedenza, che il browser è in grado di visualizzare quello inserito all'interno del file *App.js*.

Il merito dell'iniezione di queste informazioni è dato dal file *index.js* che troviamo sempre all'interno della cartella *src*.

Questo file tipicamente contiene una sola riga nell'ambiente preconfigurato, e in genere non si modifica.

La riga che troviamo è quella che, tramite il metodo render della libreria ReactDOM, va a individuare il nodo con attibuto `id="root"` all'interno del DOM e inserisce al suo interno le righe di un altro file JavaScript, che ha lo stesso nome di quello che troviamo indicato come tag HTML, nel primo argomento del metodo, nel nostro caso *App.j.*

Infatti se proviamo ad aprire quest'ultimo file, troveremo al suo interno una serie di righe che apparentemente assomigliano a codice HTML, ma che in realtà impareremo essere puro JavaScript, in particolare JavaScript XML o JSX. Per ora non preoccupiamoci più di tanto di capire il significato d'ognuna, perché le analizzeremo in dettaglio nei prossimi capitoli.

src/App.js

```
class App extends Component {
  render() {
    return (
      <div className="App">
        <header className="App-header">
          <img src={logo} className="App-logo" alt="logo" />
          <p>
            Edit <code>src/App.js</code> and save to reload.
          </p>
          <a
            className="App-link"
            href="https://reactjs.org"
            target="_blank"
            rel="noopener noreferrer"
          >
            Learn React
          </a>
        </header>
      </div>
    );
  }
}

export default App;
```

Il fatto che assomigli a codice markup HTML non è casuale, ma è una precisa scelta progettuale adottata dai creatori di React per rendere l'apprendimento di questa libreria semplice e immediato.

JSX in realtà è una notazione un po' controversa: alcuni sviluppatori l'apprezzano, mentre altri pensano che sia un enorme passo indietro. Invece di seguire un approccio classico per separare la logica dal markup, React ha deciso di combinarli insieme, utilizzando un linguaggio simile a XML.

Anche se il merito di mescolare il markup con JavaScript potrebbe essere discutibile, ha sicuramente un grosso vantaggio: l'analisi statica degli errori. Se si commette un errore di battitura nel markup JSX, il compilatore emetterà subito un errore, invece

di rimanere impassibile e lasciare allo sviluppatore l'onere di scovare il problema.

A onore del vero, JSX non è obbligatorio, nel senso che lo stesso identico risultato si potrebbe ottenere sfruttando i metodi interni alla libreria React, bypassando la fase di compilazione effettuata da Babel. Noi non vedremo questo approccio, perché a mio avviso JSX risulta sicuramente più rapido e indolore da imparare per chi già ha progettato pagine HTML, avendo una sintassi molto simile.

Grossolanamente possiamo dire che tutte le righe del file App.js interne a render() vengono "trasformate" in elementi del DOM e visualizzate dal browser grazie alle librerie JavaScript iniettate in precedenza nel file *index.html*.

Nel dettaglio, React utilizza il concetto di DOM virtuale per gestire il rendering delle pagine. Se hai familiarità con jQuery, sai che con quest'ultimo può manipolare ogni elemento del DOM di una pagina web HTML. Nella maggior parte delle applicazioni, questa interazione diretta non crea problemi se sono rispettate le regole base d'uso di jQuery.

Tuttavia, in alcuni casi, come il funzionamento di un'applicazione web in tempo reale molto interattiva, le prestazioni possono ridursi drasticamente.

Per contrastare questo, è stato inventato il concetto del DOM virtuale, che è sfruttato da molti moderni framework, incluso React. E' una sorta di DOM parallelo, ma con prestazioni ottimizzate.

Infatti, a differenza del classico DOM HTML, il DOM virtuale è più facile da manipolare ed è in grado di gestire velocemente numerose operazioni simultanee, senza influire sulle prestazioni della pagina.

React periodicamente confronta il DOM virtuale e il DOM HTML. Le differenze che riscontra, sono applicate al DOM HTML al fine di farlo corrispondere al DOM virtuale.

Non prendere paura se all'apparenza questi concetti che sembrano ostici da capire e imparare, perché ti diventeranno familiari via via che approfondiremo le basi di React.

Tornando alla pagina visualizzata nel browser, un'altra caratteristica importante dell'ambiente preconfigurato è che una qualsiasi modifica effettuata su un file del progetto, si riflette in automatico sulla pagina visualizzata all'interno del browser.

Questo permette di visualizzare in tempo reale, l'effetto di ogni modifica effettuata al codice dell'applicazione, senza dover manualmente ricaricare la pagina all'interno del browser.

Altro file è quello sfruttato quando si ha la necessità di sviluppare una progressive Web App (*serviceWorker.js*).

Per quanto riguarda i restanti file presenti nel progetto, noterai la presenza di fogli di stile .css

Questi sono usati per impostare lo stile da dare all'applicazione e tipicamente sono suddivisi in un file principale, *index.css* e una serie di file secondari, come ad esempio *App.css*, responsabili dello stile di particolari sezioni dell'applicazione.

Avremo modo comunque di approfondire questi concetti quando parleremo di come impostare lo stile dell'applicazione.

Capitolo 4

React Design Pattern: le basi

4.1 Suddividere in blocchi una SPA

React è una libreria nata per facilitare lo sviluppo di applicazioni web interattive, in particolare le SPA ossia le applicazioni costituite da un'unica pagina, anche se con il tempo sono nate diverse librerie finalizzate a estendere le funzionalità base.

Uno dei classici problemi che affligge uno sviluppatore che si accinge a imparare una nuova tecnologia o un nuovo linguaggio, è capire come strutturare il progetto sulla base del risultato finale che si vuole ottenere.

In particolare un progetto React può essere paragonato a una matriosca quindi sarà fondamentale capire uno dei concetti chiave della relazione tra matriosca esterna e matriosca interna: la relazione tra componente genitore e figlio.

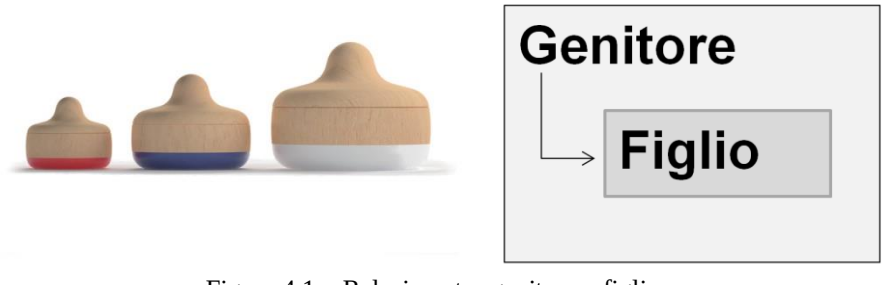

Figura 4.1 – Relazione tra genitore e figlio

Una volta capito questo concetto e capito come avviene lo scambio d'informazioni tra una matriosca esterna (genitore) e quelle interne dirette (figli) e così via fino a quella più interna, saremo in grado di creare una qualsiasi applicazione.

Un altro modo di rappresentare un'applicazione React a singola pagina è quello della figura dell'albero, esattamente come avviene per rappresentare le cartelle interne al tuo computer. Abbiamo sempre una radice, il disco fisso (genitore), che ha

all'interno una serie di cartella (figli).

Queste cartelle possono avere a loro volta altre sottocartelle, come evidenziato nella Fig. 4.2, in cui *App* è il genitore di *Menu* e di *ListaArticoli*.

Quest'ultimo è a sua volta genitore di più figli Articolo.

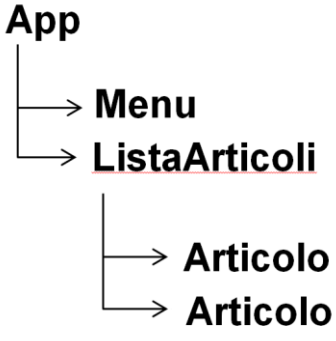

Figura 4.2 – Visualizzazione ad albero di un'applicazione

Si crea una relazione a cascata in cui è fondamentale capire come il genitore e il figlio possano comunicare tra di loro.

Sulla base di queste analogie, vediamo di fare un esempio più pratico basandoci sull'interfaccia di un'applicazione reale, come quella di Fig. 4.3.

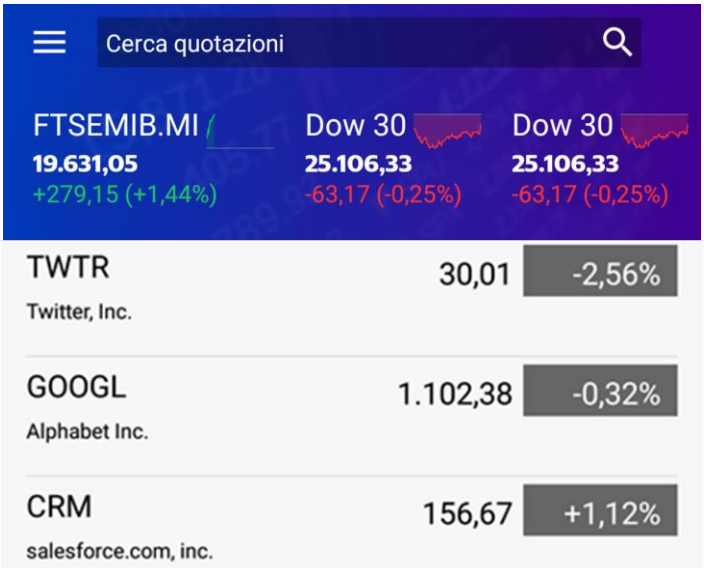

Figura 4.3 – Applicazione quotazioni bors

Fine Anteprima del libro

Ordinalo ADESSO su Amazon

Clicca qui

Oppure scrivici per ordinarlo direttamente da noi e risparmiare sulle spese di spezione