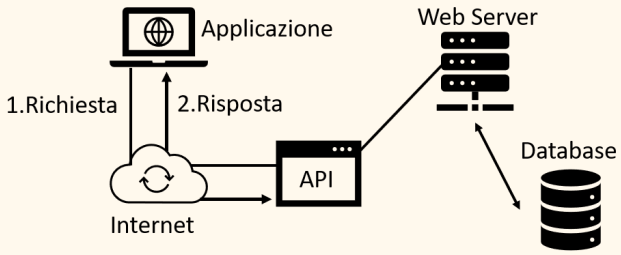
ITS - REACT JS
GLOSSARIO

—

API
Application Programming Interface
Insieme di procedure che consentono di eseguire determinate funzioni su un sistema integrato. **Nel web**, le API sono principalmente dei servizi che consentono di scaricare o modificare dati su un **server**. Queste API sono note come **Rest API** e hanno due caratteristiche principali: il **path** (ovvero l'indirizzo web da contattare) e il **metodo** (ovvero la tipologia di comando che devono eseguire).

API REST
Representational state transfer Application Programming Interface
Sono una tipologia di **API** esposte da un server che rispondono tramite un **path** (ovvero l'indirizzo web o URL da contattare) e sono caratterizzate da un **metodo** di chiamata (ovvero la tipologia di comando che devono eseguire).



Un esempio di **path** può essere il seguente: `https://server.remoto.it/api/v1/users`.

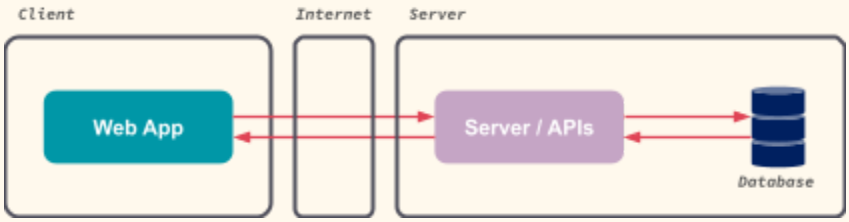
I metodi principali sono invece: **GET** (utilizzata per recuperare un'informazione), **POST** (utilizzata per salvare una **nuova** informazione sul server), **PATCH** e **PUT** (utilizzate per

modificare un dato già presente sul server), **DELETE** (utilizzata per eliminare una informazione già presente sul server).

I metodi **POST, PATCH, PUT** sono dotati di **body**, ovvero il contenitore in cui collocare i dati nuovi da comunicare al server.

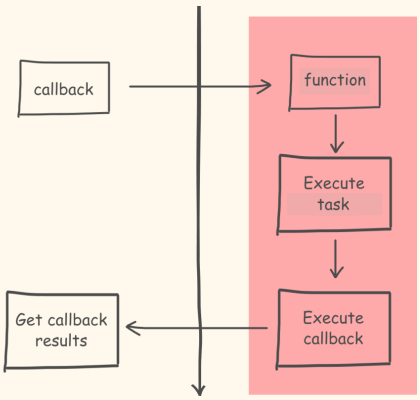
ARCHITETTURA

L'architettura telematica è il progetto esecutivo che descrive l'intera struttura delle informazioni e dei componenti ed apparati informatici e di comunicazione che le gestiscono.



CALLBACK

Si parla di **funzione di callback** quando una funzione è trasmessa a un'altra funzione come parametro/argomento. Questa funzione di callback può essere utilizzata all'interno della funzione che la riceve tra i suoi parametri.



L'uso di funzioni di callback è molto frequente nella programmazione JavaScript. Ad esse si ricorre ad esempio, nell'esecuzione di alcuni metodi e funzioni predefinite, nelle azioni asincrone come nelle chiamate HTTP o nella gestione di eventi.

CICLO DI VITA

Lifecycle

Il ciclo di vita di un prodotto, processo o attività, indica tutte le fasi che ne contraddistinguono la vita utile, dalla creazione alla distruzione, passando per tutte le fasi o stati che ne rappresentano l'evoluzione.

Nel ciclo di vita di un componente **React** possiamo distinguere essenzialmente tre fasi: l'inizializzazione del componente (`mount`), l'aggiornamento (`update`) delle sue proprietà (`props`) o del suo stato (`state`) e la rimozione del componente (`unmount`).

Il ciclo di vita di una componente React è sostanzialmente una macchina a stati.

Nei componenti di classe, il ciclo di vita fondamentale passa per le seguenti funzioni:

- `componentDidMount`, eseguito automaticamente subito *dopo* l'inizializzazione del componente;
- `componentDidUpdate`, eseguito automaticamente subito *dopo* l'aggiornamento di state e/o props del componente;
- `componentWillUnmount`, eseguito automaticamente subito *prima* della distruzione del componente e la sua rimozione dal DOM;

COMPONENTE

Uno degli elementi che, composti, concorrono a formare un insieme complesso.

In **React**, è concettualmente una funzione **JavaScript** e permette di suddividere la UI di una web app in pezzi sempre più piccoli. Al centro di tutto c'è il concetto di riusabilità: lo scopo infatti è proprio quello di creare componenti **riutilizzabili** da componenti più grandi e così via.

CLASSE

Una classe, nella programmazione orientata agli oggetti, è un costrutto di un linguaggio di programmazione usato come modello per creare oggetti. In **JavaScript**, invece, una classe è un tipo di funzione. Le classi sono dichiarate con la parola chiave `class`. Il codice dichiarato con `function` e `class` restituiscono entrambi una funzione, ma la classe possiede un costruttore. A differenza delle funzioni, le classi possono estendere altre classi e possono estendere delle interfacce. Le interfacce esistono solo in **TypeScript**.

Una classe essenziale di React si indica in Typescript in questo modo:

```
class App extends React.Component<P, S> {
   constructor(props: P) {
      super(props)
      this.state = {}
   }
   componentDidMount() { }
   componentDidUpdate() { }
   componentWillUnmount() { }
   render() {
      return <p>Hello World</p>
   }
}
```

CLIENT

Il termine client indica una componente che accede ai servizi o alle risorse di un'altra componente, detta server. Il software client in genere è di limitata complessità, limitandosi normalmente ad operare come interfaccia verso il server, spesso utilizzabile da un utente attraverso una interfaccia grafica.

DESTRUTTURAZIONE (JavaScript)

In **JavaScript**, la destrutturazione di un array o di un oggetto è una tecnica introdotta con lo standard ES2015, che permette di creare delle variabili in modo veloce e facilmente comprensibile dalle proprietà (key) di un oggetto o dalle posizioni dei valori di un array.

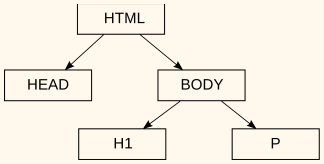
La destrutturazione `const [primo, secondo, terzo] = [0, 1, 2]` consente di creare tre **nuove** variabili che contengono gli elementi dell'**array**, nello **stesso ordine** con cui sono specificate le variabili tra le parentesi quadre.

La destrutturazione `const { nome, cognome, lavoro } = utente` consente di creare tre **nuove** variabili che contengono i valori delle **chiavi** contenute nell'**oggetto** `utente` che si chiamano esattamente **come** le variabili.

DOM

Document Object Model

Il DOM, letteralmente modello a oggetti del documento, è una forma di rappresentazione dei documenti strutturati come nodi, annidati l'uno dentro all'altro. Il codice sorgente di una pagina HTML rappresenta il DOM in forma testuale.



ENDPOINT

In campo web, si chiama endpoint un punto di accesso su una rete o su internet, che rappresenta un dispositivo connesso oppure un punto a cui un dispositivo può connettersi.

Tipicamente, un endpoint è rappresentato da un URL.

FETCH (Web API)

L'API Fetch fornisce un'interfaccia JavaScript per l'accesso e la manipolazione di parti della pipeline HTTP, come richieste e risposte. È integrata nel browser e fornisce una funzione `fetch()` globale che fornisce un modo semplice e logico per recuperare le risorse in modo asincrono attraverso la rete.

La funzione fetch prevede una gestione delle chiamate asincrone basata sulle promise e consente, tra le altre cose, di effettuare chiamate API Rest per attivare funzionalità di un server, come il recupero di dati o la modifica degli stessi da e verso uno specifico URL.

In quanto promise, la funzione `fetch` può essere trattata in modo sincrono o asyncrono (v. promise).

FRAMEWORK

Un framework è un'architettura logica di supporto (spesso un'implementazione logica di un particolare design pattern) sulla quale un software può essere progettato e realizzato. L'utilizzo di un framework impone dunque al programmatore una precisa metodologia di sviluppo del software. Per questo motivo "framework" è un concetto opposto a "libreria".

Angular è un esempio di framework.

FUNZIONE

Una funzione è un insieme di **istruzioni** racchiuse in un **blocco di codice**, che può essere contraddistinto da un **nome**, può eventualmente accettare **argomenti** o **parametri** di ingresso e restituire **valori** (return).

HTTP

Hypertext transfer protocol

È un protocollo a livello applicativo usato come principale sistema per la trasmissione d'informazioni sul web ovvero in un'architettura tipica client-server. Le specifiche del protocollo sono gestite dal World Wide Web Consortium (W3C)

IMPLEMENTAZIONE

In informatica, la realizzazione concreta di una procedura a partire dalla sua definizione logica. Si tratta, in poche parole, del codice scritto per risolvere una problematica, una funzionalità o un processo logico.

ISTANZA

Un'istanza di un'applicazione, di una funzione o di una porzione di codice è la rappresentazione in memoria RAM della classe a cui l'oggetto corrisponde. Questa istanza è normalmente interattiva e ha un ruolo attivo nella corrente sessione di utilizzo.

ITERAZIONE (CICLO)

I cicli, detti anche strutture iterative, hanno una grande importanza in programmazione. Grazie ad essi è possibile eseguire, per mezzo di poche righe di codice, un numero potenzialmente infinito di istruzioni. Sostanzialmente si tratta di ripetere un determinato numero di istruzioni fino a che non si verifica una certa condizione.

JSON

JavaScript Object Notation

È un formato adatto all'interscambio di dati fra applicazioni client/server.

JSX

JavaScript XML

È un'estensione della sintassi del linguaggio **JavaScript** ed è utilizzato per rappresentare funzioni in modo simile ad **HTML**. Viene utilizzato per esprimere contemporaneamente il markup di HTML, arricchendolo con le logiche JavaScript nello stesso file.

React DOM si occupa di trasformare il codice JSX in HTML+JS tradizionale.

LIBRERIA

Lo scopo delle librerie software è fornire una collezione di entità di base pronte per l'uso ovvero **riuso di codice**, evitando al programmatore di dover riscrivere ogni volta le stesse funzioni o strutture dati e facilitando così le operazioni di sviluppo e manutenzione.

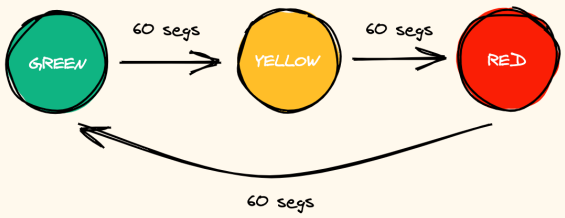
La libreria è quindi una collezione di utility a disposizione dello sviluppatore, che possono essere usate anche solo parzialmente, con piena libertà. Per questo motivo, "libreria" è un concetto opposto a "framework".

React è una libreria UI.

MACCHINA A STATI

Una macchina a stati, o "automa a stati finiti", modella tutte le transizioni di un'entità attraverso i possibili stati che può avere in un dato momento.

Un semaforo segue ad esempio i seguenti stati: verde, giallo, rosso.



Il ciclo di vita di un componente React segue uno schema a stati, ma verso una sola direzione: dal `mount` all'`unmount`, passando per uno stato iterativo di `update`.

MULTI-PAGE APPLICATION

Applicazione su pagine multiple

Un'applicazione multipagina è un'applicazione web composta da un gran numero di pagine che vengono completamente aggiornate ogni volta che i dati cambiano su di esse.

Qualsiasi modifica o trasferimento di dati al server porta a una **nuova pagina** visualizzata nel browser.

PHP è un esempio di linguaggio le cui pagine vengono renderizzate nel server.

PATTERN

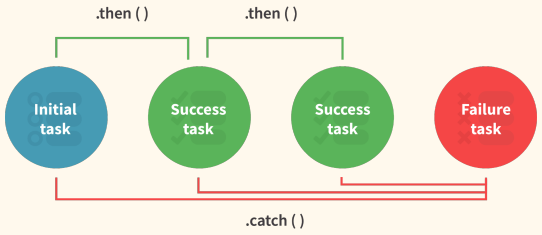
Un pattern, o un *design pattern*, è una soluzione progettuale a un problema ricorrente. La sua progettazione segue sempre delle linee guida più o meno fisse, atte a risolvere il problema nello stesso modo o tramite soluzioni simili.

POLYFILL

Un polyfill è una porzione di codice o libreria, realizzato in **JavaScript**, che consente alle funzionalità che ci si aspetta di lavorare nei browser moderni di funzionare nei browser più vecchi. Se il browser non supporta una funzione come `Array.isArray(arr)`, il polyfill può essere utilizzato per aggiungere temporaneamente questa capacità. Il codice polyfill è contenuto nell'insieme di codice della web app.

PROMISE

Le Promise in **JavaScript** sono state concepite per rappresentare **operazioni incomplete** al momento corrente che saranno però complete in futuro; per questo motivo parliamo di un costrutto adottato nel caso di **elaborazioni asincrone** e differite. Il concetto è *simile* a quello della callback.



In ES6 la funzione fetch è una promise e permette di effettuare una chiamata API.

È possibile gestire le promise in due modi, vale a dire il metodo “then” e il metodo “async/await”.

Nel primo caso, la promise viene eseguita in modo completamente asincrono, senza bloccare l’esecuzione del codice in essa contenuto:

```
// chiamata API
fetch(URL, { method: ‘GET’ }).then((res) => {
   // estrazione del body in JSON (anch’essa promise)
   return res.json()
}.then((body) => {
   console.log(body)
}).catch((e) => {
   console.error(“An error occurred”, e)
})
```

Il secondo caso, eseguito all’interno di una funzione asincrona, permette di attendere il risultato della promise e blocca l’esecuzione del codice seguente fintanto che essa non sia stata risolta.

```
const getData = async () => {
   try {
      // chiamata API
      const res = await fetch(URL, { method: ‘GET’ })
      const body = await res.json()
      console.log(body)
   } catch(e) {
      console.error(e)
   }
}
```

PROPRIETÀ

Una proprietà (in inglese prop o property) è una caratteristica che descrive un oggetto.

In React, le props sono tutti quegli oggetti, funzioni, stringhe, numeri, array o variabili in generale che vogliamo inviare ad un componente.

Le props devono essere read-only (in sola lettura): ciò sta a significare che un componente non deve mai modificare le props che riceve in input dal padre. Il padre ha la possibilità di distruggere un figlio in qualunque momento e crearne uno nuovo modificandone le props in ingresso.

PROTOCOLLO

Un protocollo è una serie di standard che regolano la successione e lo scambio di informazioni tra due dispositivi o processi.

Il protocollo HTTP è uno dei protocolli interessati dalla navigazione nel web.

REACT

React è una libreria **JavaScript** per la creazione di interfacce utente (UI, User Interface). Sviluppata nel 2013 all'interno di Facebook, adesso React è una libreria open-source supportata da una grande community di programmatori.

React consente di sviluppare applicazioni dinamiche che non necessitano di ricaricare la pagina per visualizzare i dati modificati (SPA). Inoltre nelle applicazioni React le modifiche effettuate sul codice si possono visualizzare in tempo reale, permettendo uno sviluppo rapido, efficiente e flessibile delle applicazioni web.

React è una tecnologia chiave per la carriera di Front End Developer e Mobile Developer.

SERVER

Il server è componente hardware (computer, macchina virtuale...) che attraverso un software specifico eroga un servizio. Questo servizio tipicamente espone delle API che possono essere utilizzate da uno o più client.

SESSIONE

Una sessione è l'attività svolta dall'utente dal momento in cui accede a una pagina dell'applicazione, fino a quando chiude il browser (comprende tutte le pagine aperte nella stessa finestra del browser).

SINGLE-PAGE APPLICATION

Applicazione su singola pagina

Con Single-page application o SPA si intende un'applicazione web o un sito web che può essere usato o consultato su una singola pagina web con l'obiettivo di fornire un'esperienza utente più fluida e simile alle applicazioni desktop dei sistemi operativi tradizionali.

In un'applicazione su singola pagina tutto il codice necessario (HTML, JavaScript e CSS) è recuperato in un singolo caricamento della pagina e dati aggiuntivi sono richiesti al server quando necessario, di solito in risposta ad azioni dell'utente.

Create React App genera una Single-Page Application renderizzata interamente sul browser.

STATO

Situazione, condizione di qualcosa in uno specifico momento.

In React, lo state è un contenitore di informazioni che possono variare all'interno di un componente di classe.

UI

User Interface

L'interfaccia grafica, nota anche come GUI (Graphical User Interface), è un tipo di interfaccia utente che consente l'interazione uomo-macchina in modo visuale utilizzando rappresentazioni grafiche, come bottoni, testo, tabelle, immagini e altro.

La UI è una delle componenti della UX.

URL

Uniform Resource Locator

L'URL è una sequenza di caratteri che identifica univocamente l'indirizzo di una risorsa su una rete di computer, come ad esempio un documento, un'immagine, un video, una API, tipicamente presente su un server e resa accessibile a un client. Generalmente, viene utilizzato per individuare uno specifico endpoint per una chiamata API Rest.

UX

User Experience

È un termine utilizzato per definire la relazione tra una persona e un prodotto, un servizio, un sistema. La UX quindi parte dai comportamenti dell'utente e dai suoi bisogni, da ciò che apprezza, quali capacità e limiti possiede. Le attività di design e sviluppo riconducibili alla User Experience punteranno dunque a valorizzare non il prodotto in sé, ma il vissuto dell'utente in relazione ad esso.

La UX non deve essere confusa con la user interface (UI): la UI è solo una delle componenti della UX.