03. REACT DESIGNING ARCHITECTURES

**SCARICO DELLE PARTI STATICHE:**

i file scaricati vanno poi a finire nella cache del browser in base a come è stata impostata

**RICHIEDO L'INTERSCAMBIO DEI DATI DI UN SERVER AL CLIENT:**

esistono vari tipi di richieste (una chiamata API) per ricevere dei dati come in differenti formati: json...

**MPA (multi-page application)**

multiplo sta per + file che vengono scaricati contemporaneamente quando si fa un'azione, quando si invia un nuova richiesta al server, viene richiesto un RELOAD della pagina web!

**SPA (single-page application)**

Solo la prima richiesta carica risorse statiche, la SPA non si va a ricaricare quando viene effettuata una richiesta al server, viene aggiornato dinamicamente tramite API.

le single page application NON sono usati per creare dei siti tradizionali, ma per creare APPLICAZIONI.

next.js espone il sito in maniera statica come se ci fossero + pagine (sitemap), al posto di una sola, PROBLEMA PER LA SEO E PER LA SCANSIONE DEI CRAWLER

04. REACT BEGIN CLIENT

**BUILDING BLOCKS**

HTML CSS + JS(programming language) + browser(running envroiment)

typescript rende l'applicazione + solida rispetto all'utilizzo solo di js (AGGIUNGE QUALCOSA A JS! non è un linguaggio di programmazione!)

**1. VULNERABILITA X CLIENT -> CODIFICA INSICURA**

Codice JavaScript vulnerabile può esporre al rischio di attacchi come **Cross-Site Scripting** (XSS) o **injection attacks**.

* **Esempio XSS**: Un'applicazione che permette commenti senza validazione potrebbe essere sfruttata da un attaccante per iniettare script malevoli nei campi di input, che poi rubano informazioni quando gli altri utenti visitano la pagina.
* **Esempio injection**: Un form di login che non valida correttamente l'input potrebbe accettare comandi SQL malevoli, consentendo ad un attaccante di accedere al database.

**2. VULNERABILITA X CLIENT -> DEBOLEZZA SICUREZZA NEI BROWSER**

Bug o mancanza di sicurezza possono esporre i dati memorizzati localmente (es. cookies o local storage).

**3. VULNERABILITA X CLIENT -> DIPENDENZA DALLA SICUREZZA INFRASTRUTTURALE**

Il **client** dipende da:

* **Protocolli sicuri** (es. SSL/TLS per connessioni criptate).
* **Implementazioni sicure nel backend** (es. OAuth2 per l'autenticazione).

05. REACT FRAMEWORKS LIBRARIES

**ANGULAR JS / ANGULAR**

C’è una bella differenza tra I 2, più o meno ecco le differenze principali

**AngularJS**:

* È la **versione originale** di Angular, introdotta da Google nel 2010.
* È un **framework basato su JavaScript** per la costruzione di applicazioni web.
* Utilizza un approccio **MVC** (Model-View-Controller) e si concentra sulla manipolazione del DOM direttamente.
* Non supporta **TypeScript** e si basa esclusivamente su JavaScript.

**Angular**:

* A partire dalla versione 2, Google ha completamente riscritto Angular, creando un framework moderno e diverso.
* **Angular (senza "JS")** è costruito per utilizzare **TypeScript** come linguaggio principale.
* Adotta un'architettura **component-based** (anziché MVC).
* Include miglioramenti significativi in termini di performance, modularità e strumenti di sviluppo come il CLI (Command Line Interface).
* Angular è pensato per lo sviluppo di applicazioni complesse e scalabili.

**REACT JS / REACT**

Il termine "React" è utilizzato per semplificare e rappresentare il nome commerciale e comunitario della libreria:

* **ReactJS** e **React** sono fondamentalmente lo stesso prodotto, come angualrJs si concentrava sulla manipolazione del DOM, per poi essere sostituito con **react-native**
* **ReactJS** è il nome completo originario della libreria, ma con il tempo, è diventato comune riferirsi a questa semplicemente come **React**.

**COME E QUALE FRAMEWORK/LIBRERIA SCEGLIERE?**

06. REACT FRAMEWORKS LIBRARIES

**Cos’è REACT?**

React è una libreria JavaScript open-source sviluppata da Meta (Facebook) **per creare interfacce utente (UI)**.

È utilizzata per costruire applicazioni web dinamiche, veloci e scalabili. React si focalizza esclusivamente sulla "vista" nell'architettura MVC (Model-View-Controller), ossia sulla parte che gestisce ciò che l'utente vede e con cui interagisce.

È una verisione migliorata che compone assieme **“HTML + JS”**

Collegamento a ES6 che viene usato molto all’interno di react, e alle promise che sono fondamentali per non intaccare la user exp dell’utente mentre fa qualche azione

**JSX, cos’è?**

**.JSX (JavaScript XML)** **è una sintassi speciale usata in React** che consente di scrivere strutture simili all'HTML direttamente nel codice JavaScript.

È un'estensione di JavaScript che rende il codice più leggibile e intuitivo, permettendo agli sviluppatori di definire facilmente l'interfaccia utente.

Immagine che contiene testo, schermata, software

Descrizione generata automaticamente

**.JSX** **NON viene eseguito direttamente nel browser**.

Prima di essere utilizzato, **viene trasformato in codice JavaScript** standard da strumenti come Babel. Questo significa che puoi scrivere codice simile a HTML nel tuo file JavaScript, ma alla fine React lo converte in chiamate JavaScript standard.

Per usare questo tipo di file **con TypeScript** usiamo **.TSX**

**Regole da seguire per usare JSX**

**Tutto è un componente!**

**Stato (varia) e proprietà (immutabili) esempio borraccia**

In react lo stato generalemnte è IMMUTABILE, ma può cambiare solo se c’è una causa, esistono appositamente dei meccaniscìmi, **REACT gestisce lo stato del componente**! Se al componente è stata modificata una proprietà viene distrutto e ricostruito da zero.

**Test per inziare a far girare react!**

Possiamo importare direttamente nell’html degli script contenenti delle librerire di REACT! Senza andare ad installare moltepilici moudli

Utilizzo anche di @babel per fare una traduzione del codice per farlo comprendere al browser