Progetto finale dell’A.S. 2021-2022 **Un’ applicazione WEB per le previsioni del tempo***Introduzione alle tecnologie del progetto e stato del lavoro al 30/05/2022*

**Premessa e struttura di base del progetto**

Con questa relazione si cercherà di introdurre gli scopi fondamentali, analizzare e descrivere ogni aspetto delle varie tecnologie che comporranno il progetto finale dell’anno scolastico 2021-2022, che sarà completato durante le vacanze estive: non si intende quindi descrivere i requisiti e le funzionalità dell’applicazione che andrà realizzata, ma illustrarne i componenti fondamentali per la sua costruzione.  
L’idea iniziale per il progetto è quella di realizzare un’applicazione WEB per le previsioni del tempo: il progetto è quindi rivolto maggiormente al front-end, andando ad occuparsi poco del back-end, se non quando estremamente necessario.   
Detto questo, il progetto nasce dalla necessità di studiare e realizzare un software attraverso il framework Angular, al quale si agganciano diverse “tecnologie”: TypeScript, Node.js, SASS e Angular Material.   
Prima di parlare del framework vero e proprio, quindi, si andranno prima ad analizzare queste ultime.

**Introduzione a TypeScript e JavaScript**

TypeScript è un linguaggio di programmazione OOP e open source sviluppato da Microsoft a partire dal 2012, che attualmente si trova alla versione 4.6.  
TypeScript si basa su JavaScript, del quale si può considerare un’estensione. JavaScript è nato nel 1995 come linguaggio di scripting per la creazione di siti WEB dinamici: per questo motivo il livello di astrazione è molto più alto rispetto a linguaggi quali Java o C#. Nel corso del tempo JavaScript si è esteso sempre di più a progetti molto più ampi di semplici pagine HTML, fino ad essere impiegato persino nelle applicazione desktop, portando a dei problemi importanti nella fase di scrittura del codice: una delle caratteristiche che porta spesso a commettere errori nella scrittura del codice in JavaScript è il fatto che questo lasci libertà nei tipi delle variabili, quindi possiede una *tipizzazione* dinamica.  
TypeScript è nato proprio per ovviare a questo problema fondamentale, andando ad introdurre diverse funzionalità aggiuntive: la più importante è il controllo degli errori di scrittura statico (*static type-checking*), che permette di verificare la correttezza del codice dal punto di vista sintattico prima di eseguirlo.

**Tipi di dato in TypeScript**

Una volta introdotto un po’ il linguaggio, è bene introdurre quali sono i tipi di dato supportati da TypeScript.  
Innanzitutto, per dichiarare una variabile in TypeScript bisogna seguire una precisa sintassi:

**var**/**let** [*nome variabile*]**:** [*tipo dato*] **=** [*valore dato*]**;**

es.: **var** myVar**:** number **=** 123**;**

Detto questo, TypeScript supporta i tipi primitivi, che sono alla base anche di JavaScript, ovvero:

* string, che rappresenta una stringa;
* number, che rappresenta un qualunque numero;
* boolean, che rappresenta un valore booleano.

Supporta poi gli array, che vengono tipizzati attraverso i tipi primitivi, ma anche i Tuple, che sono particolari tipi di array che contengono elementi di tipi diversi e si possono dichiarare, per esempio, in questo modo:

var parola: [string, number] = ["Ciao", 4];

TypeScript, per questi motivi, si può adottare in progetti back-end e front-end.

**Node.js**

Un’altra tecnologia che sarà alla base del progetto sarà Node.js, un runtime system open source multipiattaforma OOP per l’esecuzione di codice JavaScript, costruita sul motore V8 di Google Chrome.  
La nascita di questo runtime ha permesso a JavaScript di diventare un linguaggio per il back-end, utile innanzitutto per la programmazione lato server delle pagine WEB, ma successivamente estesosi ad molti progetti in altri ambiti.

**Node Package Manager**

Node Package Manager (npm) è un gestore di pacchetti per Node.js. Nel progetto deve essere utilizzato per ottenere le varie librerie necessarie allo svolgimento del progetto.

**Esecuzione di codice TypeScript**

TypeScript può essere compilato ed eseguito attraverso un browser ECMAScript6 oppure attraverso Node.js: per compilare il codice di TypeScript in quest’ultimo modo è necessario installare il TypeScript Compiler, ottenibile attraverso npm:

**npm install -g typescript**

Per compilare si esegue il comando:

**tsc *nomeScript.ts***

In seguito alla compilazione di codice TypeScript, verrà restituito un codice JavaScript standard.

**Ulteriori integrazioni allo studio di TypeScript**

TypeScript ha moltissime altre possibilità, anche se non sono state esplicitate in questa relazione: nel corso dello svolgimento del progetto, lo studio di TypeScript verrà continuato e verrà integrato con esercizi di prova e didattici, fino a raggiungere l’implementazione completa nel progetto.

**SASS: Syntactically Awesome Style Sheets**

Un altro componente importante da analizzare prima di introdurre Angular, e in particolare Angular Material, è SASS (Sintactically Awesome Style Sheets)

**Il framework Angular**

La prima tecnologia che analizzeremo sarà Angular. Angular è un framework open source creato da Google per lo sviluppo di applicazioni WEB basato fondamentalmente su due tecnologie:

* Node.js, runtime basata su JavaScript;
* Typescript, un linguaggio di programmazione basato anch’esso su JavaScript.

Al momento si trova alla versione stabile 13.3.6, che sarà molto probabilmente quella adottata per lo svolgimento del progetto. Alla base del funzionamento di Angular si trovano i seguenti elementi i componenti (*components*), che sono il blocco principale per la costruzione delle pagine, che comprendono a loro volta:

* Un **modello HTML** (*HTML Template*);
* Una **classe di TypeScript**;
* Un **selettore CSS** (*CSS selector*), che definisce come il componente viene usato all’interno di un modello;
* Un **file di stile CSS** (*CSS* *stylesheet*), facoltativo.

Per realizzare un progetto con

Angular, essendo un framework, comprende diverse librerie utili nella creazione di un’applicazione WEB:

* **Angular Router**, che permette la navigazione in applicazione multi-page;
* **Gestione dei form**;
* **Gestione della comunicazione con un server attraverso il protocollo HTTP**, realizzato attraverso la classe @angular/common/[http](https://angular.io/api/common/http);
* **Esecuzione di test sulla propria applicazione** attraverso Karma;
* **Angular Internationalization**, che permette di tradurre il testo e mettere formati di dati specifici in modo da rendere disponibile un’applicazione in diverse lingue;
* **Angular Animations**, che permettono di realizzare animazioni sulle pagine HTML;
* **Angular Service Worker**, utilizzato per realizzare Progressive Web App;
* **Angular Web Worker**, che permette di eseguire le parti delle applicazioni che richiedono più CPU in un thread in background;
* **Angular Universal**, che esegue il rendering delle applicazioni Angular su un server, inviando al client solamente una pagina statica.

**Installazione di Angular**

Per l’installazione di Angular è necessario prima di tutto avere installato Node.js e npm.

**Angular CLI**

La Angular CLI è l’interfaccia a riga di comando usata per interfacciarsi con il framework. Serve fondamentalmente per:

* creare un progetto con Angular;
* generare nuovi componenti, o file in generale;
* aggiungere librerie ad un progetto esistente;
* compilare ed eseguire il proprio codice.

**Angular Material**

Con il framework Angular si può utilizzare qualunque tipo di libreria per la creazione di interfacce grafiche, ma quella con cui si integra maggiormente è Angular Material, sviluppata anche questa da Google: come Angular, anche Material si basa su dei componenti (*components*), che, combinati fra di loro, danno vita a delle pagine *responsive*. Nell’uso dei componenti in Material si può scegliere se affidarsi completamente ai componenti già fatti, oppure costruendoseli partendo dalle primitive offerte dal CDK (Component Dev Kit):  
questa parte di Material permette infatti di gestire il layout della pagina, il testo degli appunti, il copia-incolla, ma anche di impostare nella pagina dei componenti semplici da personalizzare.  
Angular Material, comunque, ha le sue radici in Material Design.

**Material Design**

Material Design è un insieme di linee guida create da Google per il front-end delle applicazioni , che si basano fondamentalmente su una struttura a griglia, similmente a Bootstrap.

**Ottenimento e gestione dei dati**

Per la realizzazione di un’applicazione per la visualizzazione delle previsioni del tempo si rende necessario ottenere tutti i dati riguardanti il meteo, pertanto si utilizzerà un API che permetta proprio questo: si è pensato di utilizzare OpenWeatherMap, che permette di fornire le previsioni di tutte le città nel mondo gratuitamente, secondo precise condizioni:

* ci si deve iscrivere, sempre gratuitamente, al sito;
* le chiamate del servizio al minuto NON devono essere superiori a 60;
* le chiamate del servizio al mese NON devono essere superiori a 1000000.

Nella sua offerta gratuita, vengono offerti dei servizi base:

* previsioni per cinque giorni, con intervalli di tre ore;
* mappe del meteo;
* accesso alla dashboard del meteo;
* API per ottenere le informazioni sull’inquinamento dell’aria;
* API per ricercare le previsioni in determinate località (*Geocoding API*);
* widget del meteo da aggiungere ai siti;
* uptime, ovvero disponibilità dei server, al 95%.

I dati vengono forniti al software in seguito a specifica richiesta al server di OpenWeatherMap:

https://api.openweathermap.org/data/2.5/weather?lat={**lat**}&lon={**lon**}&appid={**API key**}

dove **lat** e **long** sono rispettivamente latitudine e longitudine e **appid** è il codice dell’API assegnato all’iscrizione. La chiamata risponderà con un file JSON, che verrà poi deserializzato, nel progetto, attraverso i metodi di TypeScript, oppure di apposite librerie, come *typescript-json-serializer*, ottenibile attraverso npm. L’output, comunque, può essere modificato nel formato (XML, HTML o JSON), nella lingua e nell’unità di misura, sempre attraverso la chiamata al server.

**Stato del lavoro**

Attualmente il progetto è ancora in una fase preliminare: la maggior parte del tempo è stata dedicata allo studio delle varie tecnologie che lo compongono, andando a svolgere, nella pratica, solamente pochi esercizi per imparare come usarle, senza andare a realizzare dei risultati concreti.

**Considerazioni finali**

Vista la vastità e la complessità delle varie tecnologie che fanno parte del lavoro, la fase di studio e ricerca dovrà proseguire ulteriormente per poter essere certi di realizzare un risultato completo e accurato.

**Fonti della relazione**

*Angular*

* <https://angular.io/>
* <https://it.wikipedia.org/wiki/Angular>

*Angular Material*

* <https://material.angular.io/>

*Node.js*

* <https://nodejs.org/it/about/>
* <https://nodejs.org/dist/latest-v16.x/docs/api/>
* <https://it.wikipedia.org/wiki/Node.js>

*TypeScript*

* <https://www.typescriptlang.org/docs/handbook/>
* <https://it.wikipedia.org/wiki/TypeScript>
* <https://radixweb.com/blog/typescript-vs-javascript>
* <https://www.html.it/guide/guida-typescript/>
* <https://www.freecodecamp.org/news/an-introduction-to-typescript/>

*OpenWeatherMap*

* <https://openweathermap.org/price>
* <https://openweathermap.org/current>
* <https://openweathermap.org/api>