

Meteo

Matilde Labruzzo 987069

Corso di Programmazione Web e Mobile - A.A. 2021/2022

# Indice

<b>1</b>	<b>Introduzione</b>	<b>2</b>
1.1	Analisi dei requisiti . . . . .	2
1.1.1	Destinatari . . . . .	2
1.1.2	Modello di valore . . . . .	2
1.1.3	Flusso dei dati . . . . .	3
1.1.4	Aspetti tecnologici . . . . .	3
<b>2</b>	<b>Interfacce</b>	<b>5</b>
2.1	Index.ejs . . . . .	5
2.2	Aree . . . . .	7
2.2.1	Area.ejs . . . . .	7
2.3	Login e Registrazione . . . . .	8
2.3.1	Login.ejs . . . . .	8
2.3.2	Registrazione.ejs . . . . .	8
2.3.3	AreaPersonale.ejs . . . . .	9
<b>3</b>	<b>Architettura</b>	<b>11</b>
3.1	Struttura del sito . . . . .	11
3.2	Routing lato server . . . . .	11
3.2.1	Descrizione delle risorse . . . . .	12
3.2.1.1	Database . . . . .	12
3.2.1.2	Autenticazione . . . . .	12
<b>4</b>	<b>Codice</b>	<b>13</b>
4.1	HTML 5 . . . . .	13
4.2	CSS3 . . . . .	13
4.3	API . . . . .	13
4.4	Node.js . . . . .	14
4.5	Librerie esterne e JavaScript . . . . .	14
<b>5</b>	<b>Possibili sviluppi e conclusioni</b>	<b>15</b>
<b>6</b>	<b>Bibliografia</b>	<b>16</b>

# Capitolo 1

## Introduzione

### 1.1 Analisi dei requisiti

#### 1.1.1 Destinatari

Grazie a una interfaccia semplice ed intuitiva, qualsiasi tipo di utente può accedere e utilizzare facilmente l'applicazione, senza alcun tipo di livello di esperienza richiesto. Infatti, tramite la barra di ricerca, l'utente può cercare e trovare le informazioni metereologiche riguardanti la città desiderata. Inoltre, sempre attraverso la barra di ricerca, nella sezione autenticazione, può accedere a maggiori servizi, come ad esempio personalizzare le città preferite.

Nella versione attuale, non sono posti particolari vincoli di banda grazie alla sola presenza degli elementi utili alla navigazione, riducendo così la latenza il più possibile. Inoltre, le immagini relative alle condizioni metereologiche delle città visualizzate saranno caricate in modalità asincrona, riducendo al minimo il periodo in cui la pagina non risponde agli input dell'utente.

Si consiglia la navigazione via PC per una maggiore semplicità di lettura, ma è comunque possibile utilizzare un telefono grazie alla responsività degli elementi presenti nelle pagine.

Gli utenti che accedono alla piattaforma web sono spinti da motivazioni prettamente personali, in particolare dalla necessità di cercare informazioni metereologiche riguardo a una o più città. Di conseguenza, l'applicazione è progettata in modo tale da fornire le informazioni su richiesta esplicita dell'utente, attraverso la barra di navigazione o l'area personale, previa registrazione e login, in cui viene data la possibilità di salvare le città preferite.

#### 1.1.2 Modello di valore

L'applicazione si contraddistingue per essere intuitiva e veloce da usare, elementi sempre graditi durante l'esperienza utente.

Grazie alla disponibilità in tempo reale di informazioni relative al tempo, quest'applicazione potrebbe essere integrata con un'API in grado di fornire informazioni a sistemi automatici usati per svolgere operazioni in base al tempo atmosferico, risparmiando così ingenti somme di denaro in sensori e cablaggi.

Benché attualmente non presenti, sarebbe facile inserire banner pubblicitari o sponsor di sorta grazie alla struttura modulare del progetto.

Entrambi questi elementi accrescerebbero notevolmente il valore economico dell'applicazione in dipendenza, rispettivamente, al numero di sistemi automatici collegabili per utente o al numero di banner inseriti, motivo per cui risulta difficile formulare una stima esatta di valore economico.

### 1.1.3 Flusso dei dati

Il flusso dei dati all'interno dell'applicazione è unicamente in formato JSON: lo scambio di dati tra client e server, tra client e API o tra server e API è infatti interamente gestito attraverso l'inoltro di oggetti e stringhe JSON.

Grazie ad una struttura RESTful, le comunicazioni saranno unicamente aperte dal client nel momento in cui necessiterà di una risorsa (come il meteo relativo ad una città oppure un'altra pagina del sito), il quale richiederà ciò di cui necessita tramite oggetti JSON inviati al server, ricevendone altri in risposta che permetteranno di aggiornare un frammento della pagina o di effettuare il reindirizzamento.

I contenuti salvati sono interamente archiviati lato server tramite l'uso di un database MongoDB ad eccezione della preferenza espressa dall'utente per cambiare la pagina in *dark mode* oppure in *light mode*, la quale è salvata localmente attraverso localStorage.

Allo stato attuale, il progetto prevede solo costi per la manutenzione in up del server, senza che siano necessari particolari interventi di manutenzione periodici.

Il progetto utilizza unicamente librerie e API reperibili gratuitamente, ma sarebbe perfettamente possibile modificarle ed adottarne di closed source a pagamento con pochi e semplici aggiustamenti grazie ad una gestione modulare del codice.

### 1.1.4 Aspetti tecnologici

La trasmissione di dati può essere effettuata in chiaro ad eccezione della password usata per accedere alla propria area personale, la quale viene inviata sotto forma di SHA256 per evitare che sia leggibile.

Il database prevede la creazione di una singola collezione in cui ogni documento contiene:

- *\_id*: id univoco assegnato dal database in automatico;
- *user*: username;
- *email*: email dell'utente;
- *pwd*: password dell'utente sotto forma di digest SHA256;
- *pref*: array contenente le città messe tra i preferiti dall'utente; può essere vuoto.

Tecnologie utilizzate:

- *HTML 5*: realizzazione della struttura delle pagine;
- *Bootstrap e CSS3*: gestione degli stili;
- *Swiper API*: realizzazione degli swiper presenti all'interno delle pagine;
- *JavaScript*: realizzazione delle richieste al server e alle API, nonché del toggle della darkmode e l'aggiornamento di frammenti di pagina con dati ricevuti dal server;
- *Node JS*: implementazione del server con tutte le sue funzionalità (caricamento di tutte le pagine, invio di oggetti JSON per aggiornare frammenti di pagina, interrogazione dell'API per ottenere il nome delle città e delle aree geografiche);
- *Express*: framework utilizzato per semplificare il deploy del server;
- *JSON*: formato usato per la trasmissione dei dati;

- *localStorage*: usato per il salvataggio della preferenza utente per la visualizzazione della pagina (dark o light mode);
- *MongoDB*: database utilizzato per salvare le informazioni relative agli utenti registrati sull'app;
- *API*: nel progetto sono state utilizzate 4 API:
  - *openWeather*: API usata per ottenere le condizioni meteo in tempo reale;
  - *pexels*: API utilizzata per reperire le immagini mostrate nelle pagine;
  - *spott*: API utilizzata per ottenere i nomi delle città mondiali e per permettere la ricerca incrementale;
  - *rest countries*: API utilizzata per ottenere i nomi delle capitali delle diverse aree geografiche.

# Capitolo 2

## Interfacce

Per realizzare le pagine sono stati usati *HTML 5* e *Bootstrap*.

Gli elementi di personalizzazione, come il nome utente nella navbar una volta autenticati o il meteo in tempo reale delle città salvate tra i preferiti, sono inseriti utilizzando *EJS*.

### 2.1 Index.ejs

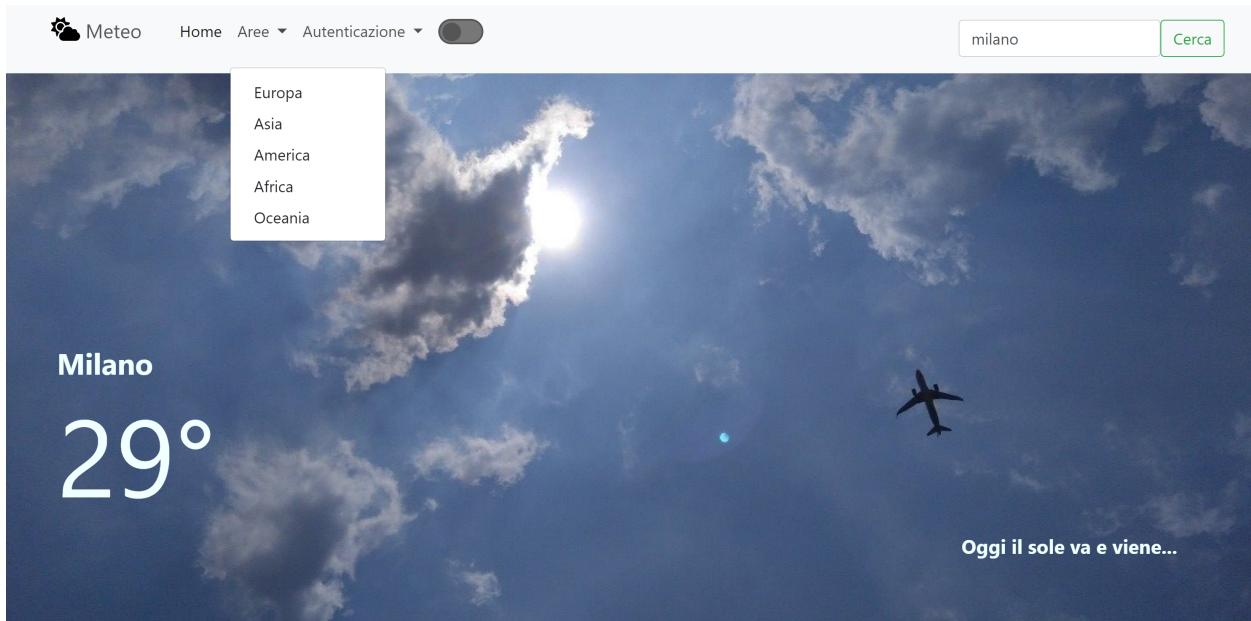


Figura 2.1: Metà superiore dell'home page

Nella parte superiore dell'index come prima cosa troviamo una navbar semplice con elementi completamente intuitivi: a partire da sinistra si ha il nome e il logo dell'applicazione, e il tasto home; grazie al menù a tendina *Aree* è possibile navigare per aree geografiche (i continenti) e cercare informazioni sulle città relative ad esse; cliccando invece sul menù *Autenticazione* l'utente può accedervi o registrarsi; a seguito del login, questo menù viene sostituito con un altro per poter effettuare il logout e per poter accedere all'area personale, dove sono salvate le città preferite dall'utente e dove quest'ultimo può cambiare le informazioni di accesso. Sempre sulla navbar troviamo un toggle che dà la possibilità all'utente di cambiare la modalità di visualizzazione, *light mode* o *dark mode*, a seconda della sua preferenza. Sulla destra troviamo invece una barra di ricerca.

Al di sotto della navbar troviamo un'immagine di copertina con le informazioni metereologiche relative alla città corrente, ottenute dalla geolocalizzazione e dall'apposita API; questi dati, così come l'immagine di sfondo, vengono sostituite dalla città cercata nella barra di ricerca.

### Città preferite



Figura 2.2: Metà inferiore dell'home page dopo aver effettuato il login

Superate le informazioni relative al servizio proposto, troviamo un'area apposita dove vengono visualizzati i dati metereologici delle città che sono state salvate dall'utente nell'apposita area personale, a cui si può accedere solamente una volta effettuato il login (o registrazione). Motivo per il quale, in assenza di accesso verrà visualizzata una scritta suggerendo all'utente di accedervi per poter visualizzare correttamente tutti i dati.

The image shows a horizontal slider (Swiper) containing five cards, each with a quote and the author's name. The cards are semi-transparent and overlap slightly.

- Card 1: "Il tramonto è delizioso, pioggia è rinfrescante, vento sù, la neve è esilarante esiste il cattivo tempo ma diversi tipi di bel tempo" - John Ruskin
- Card 2: "A volte vorrei essere il tempo: avreste parlato di me in ogni conversazione. E quando piovesse, sarei l'argomento del giorno" - John Mayer
- Card 3: "Chi desidera vedere l'arcobaleno, deve imparare ad amare la pioggia" - Paolo Coelho
- Card 4: "L'aria fresca della sera è il respiro del vento che si addormenta placido tra le braccia della notte" - Umberto Eco
- Card 5: "Può distinguere il mare da dove vi si riflette? O dire dove la pioggia e comincia la malinconia?" - Haruki Murakami

Below the slider, there are several footer links and a newsletter sign-up form:

- METEO**: Previsioni in tempo reale e in tutto il mondo
- AREE**: Europa, Asia, America, Africa, Oceania
- AUTENTICAZIONE**: Accedi, Registrati
- RIMANI AGGIORNATO**: Accedi o iscriviti se non sei già registrato per rimanere aggiornato!
- 

Figura 2.3: Swiper con citazioni e footer

Oltre l'area dedicata alle città preferite, invece, troviamo uno slider ottenuto con *Swiper API* in cui, scorrendo, si possono leggere alcune citazioni e aforismi sul meteo.

Infine, in fondo ad ogni pagina dell'applicazione troviamo un footer con le informazioni di riepilogo e i riferimenti alle pagine del sito web, così come la possibilità di inserire la propria email per rimanere aggiornato.

## 2.2 Aree

### 2.2.1 Area.ejs

Capitali dell'area: europe

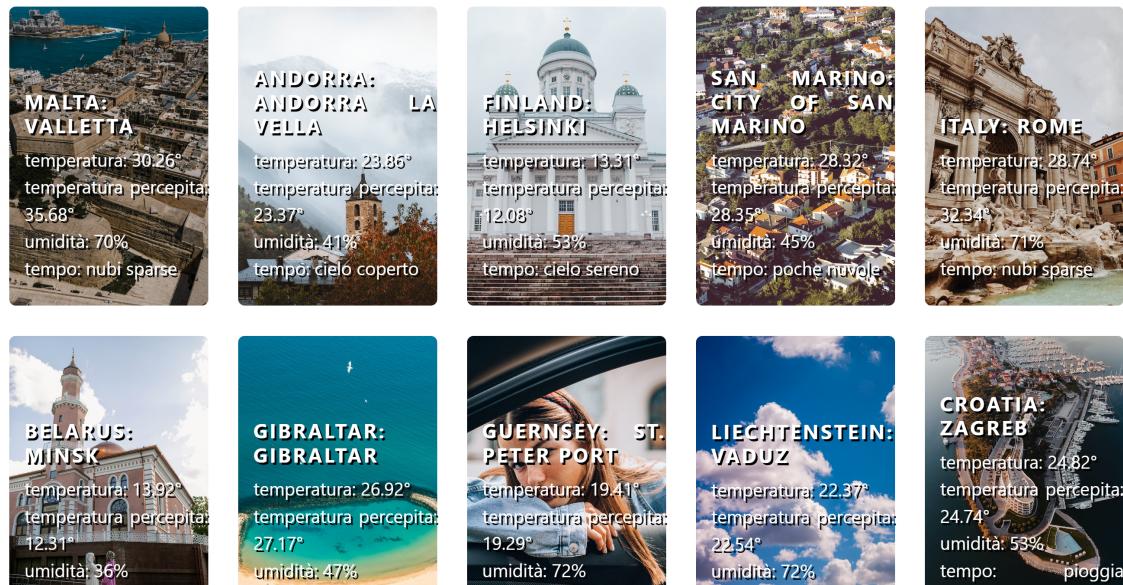


Figura 2.4: Pagina dedicata alle diverse aree geografiche

In questa pagina si trovano le informazioni metereologiche relative alle capitali delle diverse aree geografiche. La struttura della pagina, costituita da uno swiper a griglia oltre alla navbar semplificata e il footer, è uguale per tutti i continenti; infatti, quando viene selezionata l'area, ciò che cambia sono i dati delle città che vengono mostrate.

## 2.3 Login e Registrazione

### 2.3.1 Login.ejs

The screenshot shows a login form titled "Login". At the top left is a logo with a sun and clouds and the word "Meteo". To its right are links for "Home" and a dark grey toggle switch. The main area has two input fields: the top one is labeled "Username" and the bottom one is labeled "Password". Below these is a blue button with the text "Effettua login". At the bottom of the form is a link "Non sei ancora registrato? Registrati".

Figura 2.5: Pagina dedicata al login

Grazie a questa pagina è possibile effettuare il login nel caso si sia già registrati oppure di registrarsi attraverso l'apposito link al di sotto del form con le credenziali.  
La navbar è semplificata in modo da rendere l'interfaccia meno dispersiva e permette unicamente di tornare alla home e gestire la modalità di visualizzazione della pagina.

### 2.3.2 Registrazione.ejs

The screenshot shows a registration form titled "Registrazione". At the top left is a logo with a sun and clouds and the word "Meteo". To its right are links for "Home" and a dark grey toggle switch. The main area has three input fields: the first is labeled "Username", the second is labeled "Indirizzo Email", and the third is labeled "Password". Below these is a blue button with the text "Registrati". At the bottom of the form is a link "Già registrato? Accedi".

Figura 2.6: Pagina dedicata alla registrazione

In questa pagina verranno richieste le credenziali usate in seguito per effettuare il login (email e password) e un nome utente. Questi dati saranno modificabili in un secondo momento tramite l'area personale (**N.B.:** non è possibile avere più utenti con la stessa mail e/o lo stesso username).

Se necessario, è possibile passare alla pagina di login tramite il link in basso.

La navbar è semplificata nella stessa maniera della pagina di login.

### 2.3.3 AreaPersonale.ejs

The screenshot shows the 'AreaPersonale.ejs' page. At the top left is a logo with a sun and clouds, followed by the text 'Meteo'. To its right are links for 'Home' and 'utente1' with a dropdown arrow, and a dark grey toggle switch. Below this is a form with two rows of input fields. The first row contains 'nome utente:' with the value 'utente1' and an empty input field next to it. The second row contains 'email:' with the value 'utente1@mail.com' and an empty input field next to it. Underneath these is a section titled 'città preferite:' with a list of four cities: 'milano', 'bologna', 'firenze', and 'new york'. Each city entry has a blue horizontal bar below it with the word 'Rimuovi' at the end. The entire interface has a light grey background with dark grey header and form sections.

Figura 2.7: Metà superiore della pagina dedicata all'area personale

Nella metà superiore della pagina troviamo una navbar analoga a quella presente nell'index a seguito del login, a eccezione della barra di ricerca.

Al di sotto di essa, abbiamo i dati relativi all'utente (username e email. La password non viene mostrata per ragioni di sicurezza) e l'elenco delle città salvate tra i preferiti, se presenti.

## Modifica i tuoi dati qui

The screenshot shows a user interface for modifying personal data. At the top, a header reads "Modifica i tuoi dati qui". Below it is a "Username" input field with the placeholder "Username". A blue button labeled "Aggiorna username" is positioned above the input field. Below the input field is an "Indirizzo Email" input field with the placeholder "Indirizzo Email". A blue button labeled "Aggiorna e-mail" is positioned above the input field. Below the email input field is a "Password" input field with the placeholder "Password". A blue button labeled "Aggiorna password" is positioned above the input field. At the bottom, there is a "Città da aggiungere ai preferiti" input field with the placeholder "Città da aggiungere ai preferiti". A blue button labeled "Aggiorna città preferite" is positioned above the input field.

Figura 2.8: Metà inferiore della pagina dedicata all'area personale

A seguito di queste informazioni, troviamo una serie di form che permettono, attraverso chiamate asincrone al server, di modificare i dati relativi all'utente, quali:

- username;
- email;
- password;
- città salvate tra i preferiti

# Capitolo 3

## Architettura

### 3.1 Struttura del sito

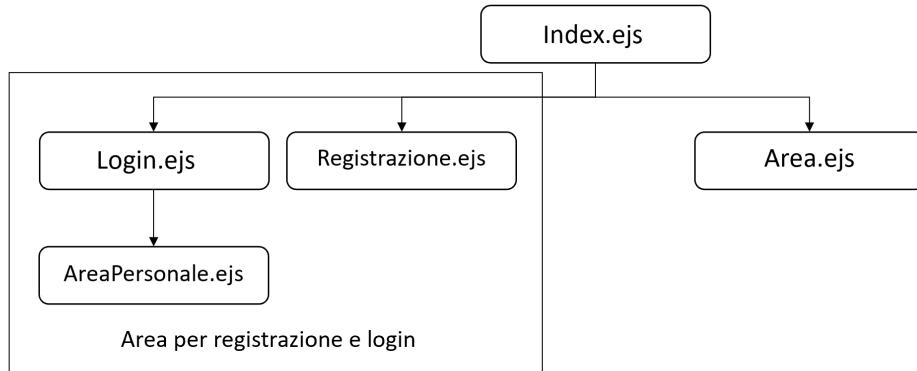


Figura 3.1: Architettura gerarchica delle pagine

Il punto di accesso al sito è pensato per essere costituito dall'index. Attraverso di essa sarà infatti possibile accedere tramite uno o più passaggi a tutte le pagine del sito (la navigazione da una pagina all'altra rimane comunque sempre disponibile attraverso la navbar, a prescindere da dove ci si trovi nel sito).

Per accedere alla pagina *AreaPersonale.ejs* e a qualunque funzione che richieda di essere loggati, sarà necessario aver effettuato realmente il login: in caso contrario, la richiesta verrà rifiutata dal server.

### 3.2 Routing lato server

Le route previste dal server sono:

- / (GET): index. Nel caso in cui si sia autenticati (attraverso un cookie), verrà modificata la navbar in modo da mostrare un menù a tendina per entrare nell'area personale al posto di quello per l'autenticazione; questo viene realizzato modificando il JSON inviato a EJS per renderizzare la pagina;

- */login* (GET): pagina di login. Nel caso il login vada a buon fine, si verrà reindirizzati all'index; altrimenti, si verrà reindirizzati a */login?auth=fail*, query che permette di mostrare un messaggio di errore;
- */registrazione* (GET): pagina per la registrazione. Sia che la registrazione vada a buon fine sia che dia errore, verrà mostrato un messaggio in relazione al risultato ottenuto dal server. Per effettuare il login bisognerà usare la pagina apposita;
- */verificaCredenziali* (GET): indirizzo raggiunto dal form nella pagina di login per verificare le credenziali inserite;
- */creaUtente* (POST): indirizzo raggiunto per creare un nuovo utente;
- */areaPersonale* (GET): pagina relativa all'area personale;
- */aggiornaDati* (POST): indirizzo raggiunto per modificare i dati relativi all'utente registrato;
- */aggiungiCit* (POST): indirizzo usato per aggiungere una città dai preferiti;
- */rimuoviCit* (PUT): indirizzo usato per rimuovere una città dai preferiti;
- */logout* (GET): indirizzo usato per eliminare il cookie usato per autenticare l'utente;
- */aree* (GET): pagina per mostrare le previsioni delle diverse aree geografiche (continenti);
- */aree/area* (GET): indirizzo raggiunto per ottenere i paesi e le capitali di una data area.

### 3.2.1 Descrizione delle risorse

#### 3.2.1.1 Database

Il motore utilizzato per realizzare il database è MongoDB.

Il database si compone di un'unica collezione (chiamata *users*), utilizzata per salvare i dati relativi agli utenti registrati. Ogni documento, come anticipato nell'introduzione, avrà la seguente struttura:

- *\_id*: id univoco assegnato dal database in automatico;
- *user*: username;
- *email*: email dell'utente;
- *pwd*: password dell'utente sotto forma di digest SHA256;
- *pref*: array contenente le città messe tra i preferiti dall'utente; può essere vuoto.

#### 3.2.1.2 Autenticazione

L'autenticazione sarà considerata corretta se il digest SHA256 della password e l'email inseriti dall'utente corrisponderanno a quelli di un documento presente nel database (si ricorda che username e email sono **univoci**).

Nel caso l'autenticazione vada a buon fine, verrà restituito un *cookie* valido per 30 minuti e contente lo username dell'utente. Se l'utente dovesse cambiare il suo username all'interno dell'area personale, il contenuto del cookie verrà aggiornato.

# Capitolo 4

## Codice

### 4.1 HTML 5

Il codice di tutte le pagine dell'applicazione è strutturato in modo da avere:

- lo stesso contenuto nel tag head;
- una navbar contenente:
  1. il nome del sito (Meteo) e la sua icona;
  2. un link alla Home in modo da poterci sempre tornare in qualunque momento;
  3. un menù a tendina per il meteo in tempo reale delle diverse aree geografiche (continenti);
  4. un menù a tendina per autenticarsi/registrarsi oppure per accedere all'area personale/effettuare il logout nel caso in cui si sia già autenticati;
  5. un toggle per scegliere tra light e dark mode;
  6. la barra di ricerca per ottenere il meteo in tempo reale di una città.

I punti 3 e 6 non saranno presenti nella navbar delle pagine dedicate a login e registrazione.

Ogni pagina avrà poi un contenuto personalizzato in base al suo scopo.

### 4.2 CSS3

Nel progetto, oltre ad utilizzare le classi di *Bootstrap* per gestire gli stili, sono stati integrati due file *css* che si possono trovare alla repository github <https://github.com/matildelabruzzo/ProgettoPwm> seguendo il percorso *assets/css*.

### 4.3 API

Nel progetto sono state integrate 4 API:

- *openWeather*: API usata per ottenere le condizioni meteo in tempo reale;
- *pexels*: API utilizzata per reperire le immagini relative alle città mostrate nelle pagine a partire da una query contenente un filtro che descrive la città mostrata;
- *spott*: API utilizzata per ottenere i nomi delle città mondiali e per permettere la ricerca incrementale;
- *rest countries*: API utilizzata per ottenere i nomi delle capitali delle diverse aree geografiche.

## 4.4 Node.js

Il server Node.js, implementato secondo paradigma *RESTful*, mette in pratica tutte le funzionalità descritte nei capitoli precedenti. Viene implementato grazie all'uso del framework *Express* e di *MongoDB* per memorizzare i dati degli utenti registrati.

La comunicazione avviene tramite l'uso di HTTP e le chiamate asincrone; le chiamate effettuate dal server al database sono anch'esse tutte asincrone.

Si può utilizzare un file presente in *assets/mongoDB* per configurare il database in automatico.

Per consultare il file *server.js* contenente il server in toto e il relativo codice, riferirsi alla repository github <https://github.com/matildelabruzzo/ProgettoPwm> seguendo il percorso *assets/node*.

## 4.5 Librerie esterne e JavaScript

Per consultare tutti i file JavaScript e il relativo codice, riferirsi alla repository github <https://github.com/matildelabruzzo/ProgettoPwm> nel percorso *assets/js*.

L'unica libreria esterna integrata è utilizzata per il calcolo dello SHA256 di una stringa è reperibile al link <https://cdnjs.cloudflare.com/ajax/libs/crypto-js/3.1.2/rollups/sha256.js>.

## Capitolo 5

# Possibili sviluppi e conclusioni

L'applicazione si occupa di fornire le condizioni meteo in tempo reale relative ad una o più città in tutto il mondo.

Alcuni possibili spunti di sviluppo per accrescere l'efficienza, la sicurezza e l'estetica dell'applicazione potrebbero essere:

- introduzione di librerie più efficienti rispetto a quelle attualmente integrate;
- efficientamento del codice JavaScript attualmente integrato attraverso un maggiore studio delle funzioni offerte ed eventuale introduzione di apposite librerie per svolgere tali scopi;
- introduzione di web worker per effettuare le richieste asincrone al server tramite fetch, caricare le immagini e effettuare il calcolo dello SHA256 delle password inserite dall'utente;
- introduzione di un sistema di comunicazione con chiavi SSL per l'utilizzo del protocollo HTTPS;
- introduzione di una navbar apposita per i dispositivi mobili.

Alcuni spunti di sviluppo per accrescere il valore commerciale dell'applicazione sono riassumibili in:

- introduzione di pubblicità all'interno del sito;
- introduzione di un'API per fornire ad altri sistemi il meteo in tempo reale.

# Capitolo 6

## Bibliografia

- API pexels: <https://www.pexels.com>;
- API openWeather: <https://www.pexels.com>;
- API città: <https://www.spott.dev/>;
- API paesi e capitali: <https://restcountries.com/>;
- Libreria usata per generare digest SHA256: <https://cdnjs.cloudflare.com/ajax/libs/crypto-js/3.1.2/rollups/sha256.js>;
- Repository github con l'intero progetto e la documentazione con LaTex sorgente: <https://github.com/matildelabruzzo/ProgettoPwm>;
- Bootstrap: <https://getbootstrap.com>;
- Swiper API: <https://swiperjs.com/swiper-api>;