

Meteo in Tempo Reale

Matteo Celardo 984457

PWM - secondo semestre A.A. 2021/2022

Indice

1	Introduzione	2
1.1	Analisi dei requisiti	2
1.1.1	Destinatari	2
1.1.2	Modello di valore	2
1.1.3	flusso di dati	3
1.1.4	Aspetti tecnologici	3
2	Interfacce	5
2.1	Index.ejs	5
2.2	Previsioni	7
2.2.1	Regionali.ejs	7
2.2.2	CitCercata.ejs	8
2.3	Login e Registrazione	8
2.3.1	Login.ejs	8
2.3.2	Registrazione.ejs	9
2.3.3	AreaPersonale.ejs	9
3	Architettura	11
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
4	Codice	13
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
5	Possibili sviluppi e conclusioni	15
6	Bibliografia	16

Capitolo 1

Introduzione

1.1 Analisi dei requisiti

1.1.1 Destinatari

L'applicazione è progettata per andare in contro a qualunque tipo di utente, dal più al meno esperto, grazie ad un'interfaccia semplice ed intuitiva, la quale permette di trovare facilmente le informazioni relative al meteo in tempo reale della città a cui si è interessati tramite la barra di ricerca o tramite le città impostate tra i propri preferiti, previa registrazione e introduzione delle stesse nell'apposito spazio all'interno dell'area personale.

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 meteorologiche delle città visualizzate saranno caricate in modalità asincrona, riducendo al minimo il periodo in cui la pagina non risponde agli input dell'utente.

Attualmente, 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 si recano sulla piattaforma web saranno principalmente persone che hanno la necessità di sapere le condizioni meteo di una città di loro interesse, motivo per cui i contenuti sono organizzati in modo tale da fornire informazioni su richiesta esplicita dell'utente tramite una barra di ricerca oppure tramite gli elementi salvati nell'area personali una volta effettuato il login.

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 di dati

Il flusso dei dati all'interno dell'applicazione é unicamente in formato JSON: lo scambio di dati tra client e server, tra client e API o tra server e API é 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 vare 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 colsed 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;
- *pw*: 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 e bootstrap*: realizzazione della struttura delle pagine e gestione degli stili con bootstrap;
- *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 dei comuni e delle regioni italiani);
- *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 3 API:
 - *openWeather*: API usata per ottenere le condizioni meteo in tempo reale;
 - *pexels*: API utilizzata per reperire le immagini mostrate nelle pagine;
 - *comuni ITA*: API usata per ottenere i nomi di regioni, province e comuni italiani.

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*.

In ogni pagina, il contenuto del tag *head* e alcuni script (necessari per dark mode toggle, bootstrap e ricerca di una città) sono aggiunti tramite l'utilizzo di *partials*, elementi supportati da EJS.

2.1 Index.ejs

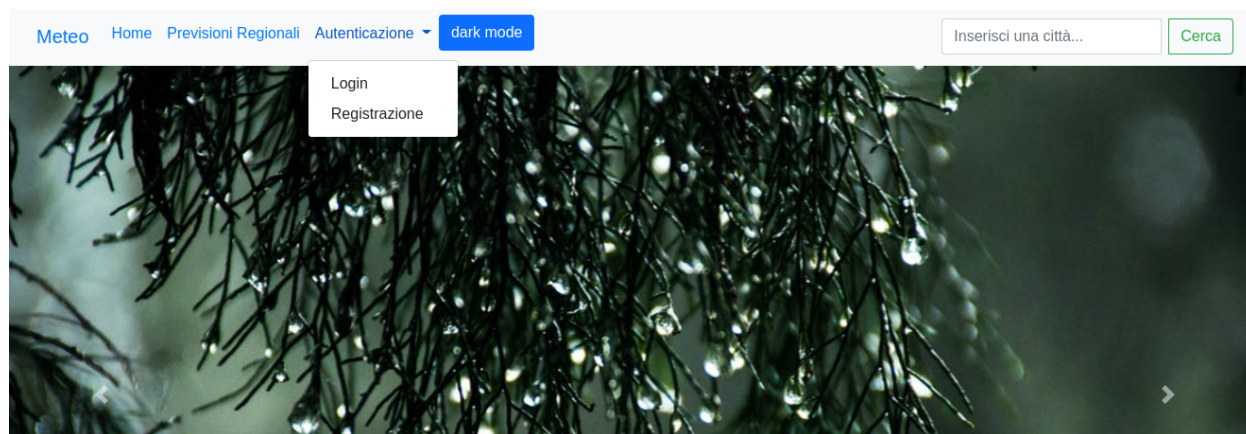


Figura 2.1: Metà superiore dell'home page

Nella metà superiore dell'index troviamo una navbar semplice ed intuitiva: abbiamo infatti la possibilità di vedere le previsioni delle province di ogni regione cliccando su *Previsioni Regionali*, di autenticarci facendo il login o la registrazione tramite il menù a tendina apribile cliccando su *Autenticazione* (menù che sarà sostituito con un altro a seguito del login per poter effettuare il logout e accedere all'area personale), di cambiare la visualizzazione della pagina dalla modalità *dark mode* a quella *light mode* ed infine di cercare una città tramite l'apposita barra di ricerca in alto a destra.

Al di sotto della navbar, troviamo uno slider temporizzato che mostra immagini relative a condizioni climatiche differenti.

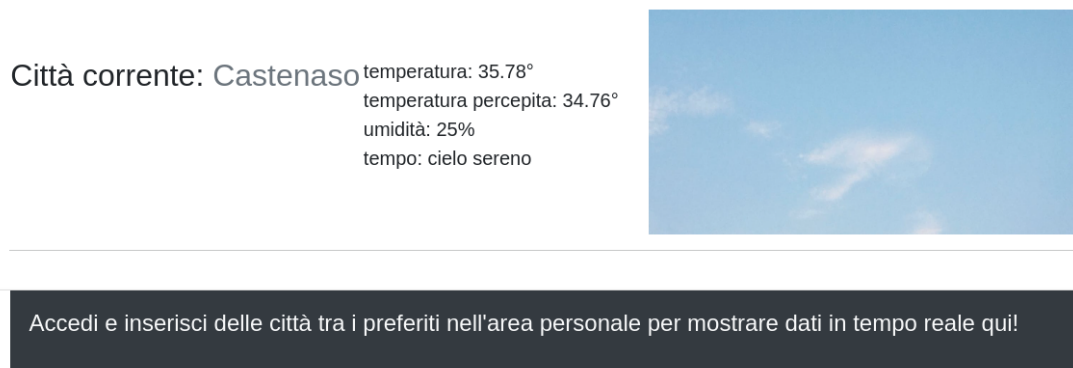


Figura 2.2: Metà inferiore dell'home page

Superato lo slider, troviamo una sezione dedicata alle condizioni meteo della città in cui ci troviamo (le coordinate sono ottenute con l'oggetto *navigator.geolocation*, motivo per cui utilizzare una connessione da telefono cellulare garantirà una maggiore precisione) a seguito della quale è presente un'area dedicata alla visualizzazione delle città salvate tra i preferiti nell'area personale accessibile dopo il login.

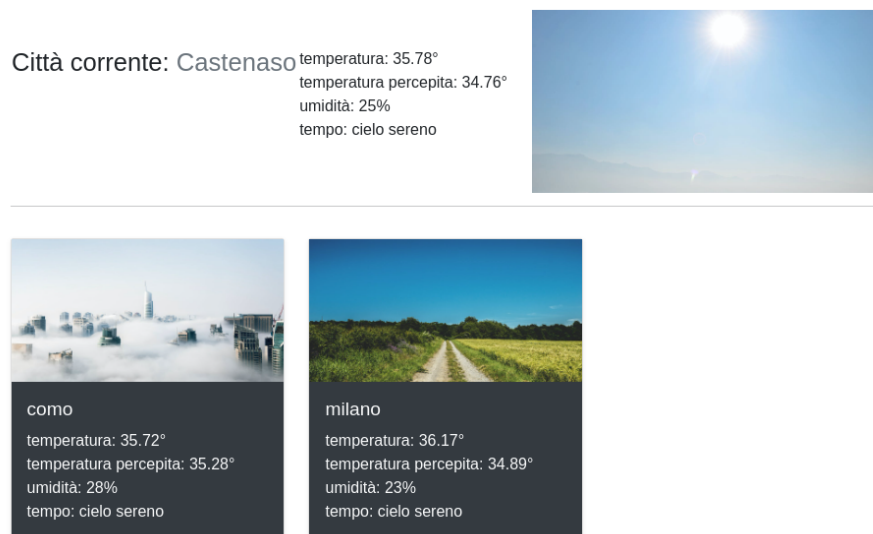


Figura 2.3: Metà inferiore dell'home page dopo aver effettuato il login

2.2 Previsioni

2.2.1 Regionali.ejs



Figura 2.4: Pagina dedicata alle previsioni regionali prima di cliccare su una regione

In questa pagina è possibile visualizzare le previsioni relative a tutte le province di una regione premendo sul tasto della regione voluta.

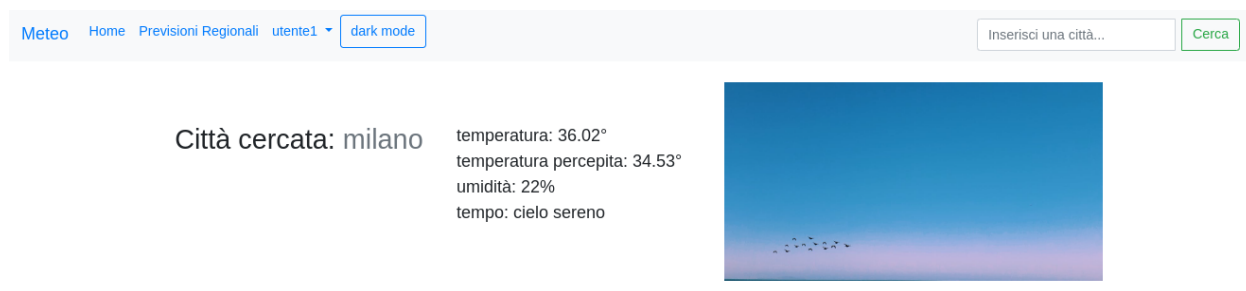
Fatto ciò, verrà visualizzato per ogni provincia un riquadro analogo a quelli presenti nell'index dopo il login con le informazioni metereologiche in tempo reale.

molise



Figura 2.5: Nell'immagine di esempio, la regione selezionata è stata il Molise

2.2.2 CitCercata.ejs



The screenshot shows the top navigation bar with links: Meteo, Home, Previsioni Regionali, utente1, and a dark mode toggle. A search input field contains 'Inserisci una città...' and a green 'Cerca' button. Below the navigation bar, the page displays 'Città cercata: milano' followed by weather details: 'temperatura: 36.02°', 'temperatura percepita: 34.53°', 'umidità: 22%', and 'tempo: cielo sereno'. To the right of the text is a rectangular image showing a clear blue sky with a few birds flying.

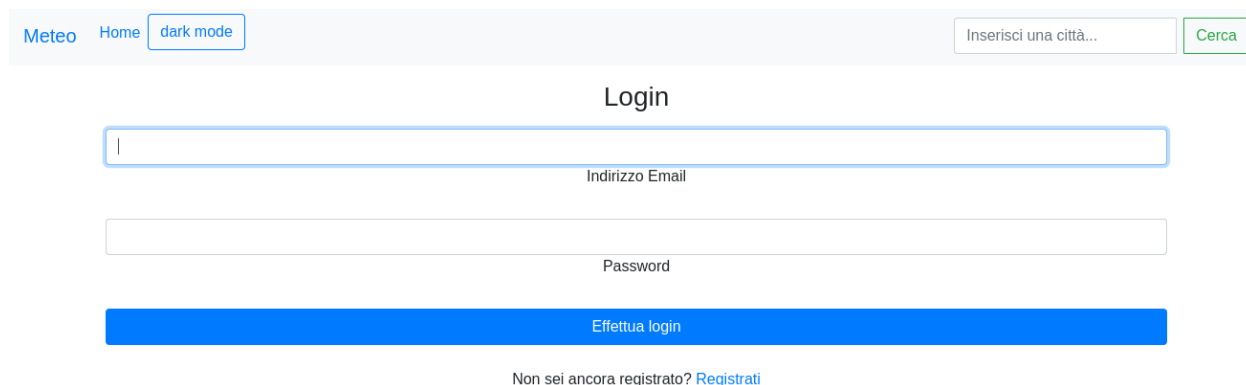
Figura 2.6: Pagina visualizzata a seguito della ricerca di una città nella navbar

Attraverso la barra di ricerca in alto a destra nella navbar, è possibile cercare il nome della città per cui si vogliono ottenere le condizioni meteo in tempo reale.

Scrivendo il nome, verrà visualizzato a sinistra dell'input una scritta che informerà l'utente se la città inserita risulta esistente o meno (il sistema è strutturato per non essere *case sensitive*). Nel caso in cui la città risulti esistente, verrà visualizzata la pagina in immagine, in caso contrario si rimarrà sulla pagina da cui si aveva fatto la ricerca.

2.3 Login e Registrazione

2.3.1 Login.ejs



The screenshot shows the login page. The top navigation bar is identical to the previous one. The main content area is titled 'Login' and contains two input fields: 'Indirizzo Email' and 'Password'. Below these fields is a blue button labeled 'Effettua login'. At the bottom, there is a link: 'Non sei ancora registrato? [Registrati](#)'.

Figura 2.7: 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, gestire la modalità di visualizzazione della pagina oppure cercare una città.

2.3.2 Registrazione.ejs

autenticati'." data-bbox="114 125 875 325"/>

Figura 2.8: 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



Figura 2.9: 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.

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.

The image shows a web form titled "Modifica i tuoi dati qui". It contains four input fields, each followed by a blue button to update the data. The fields and buttons are: 1. A text input field labeled "username" with a blue button labeled "Aggiorna username". 2. A text input field labeled "Indirizzo Email" with a blue button labeled "Aggiorna e-mail". 3. A text input field labeled "Password" with a blue button labeled "Aggiorna password". 4. A text input field labeled "Città da aggiungere ai preferiti" with a blue button labeled "Aggiorna città preferite".

Figura 2.10: 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

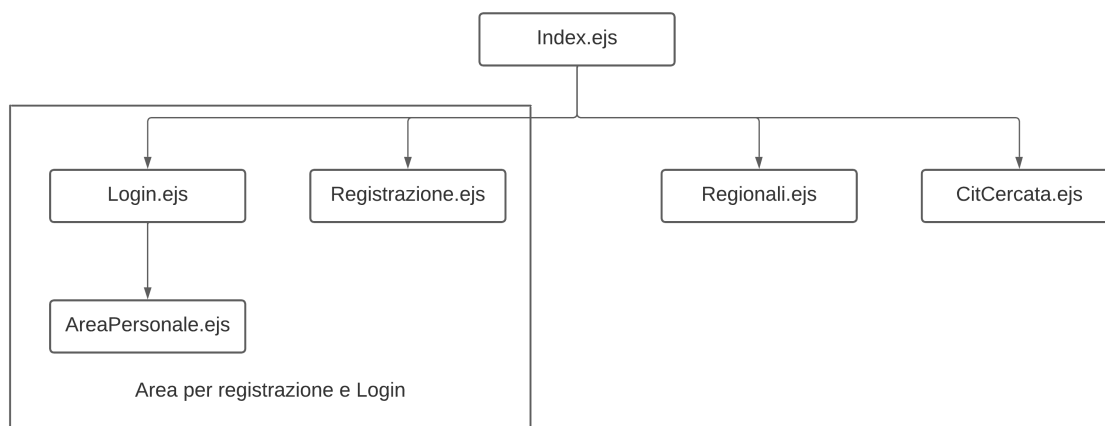


Figura 3.1: Architettura gerarchica delle pagine

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;
- */verificaCredenziali* (GET): indirizzo raggiunto dal form nella pagina di login per verificare le credenziali inserite;
- */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;
- */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;
- */aggiornaCit* (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;
- */citCercata* (GET): indirizzo usato per verificare se la città cercata esista o meno;
- */meteoCitCercata* (GET): pagina che mostra il meteo in tempo reale della città cercata;
- */previsioniRegionali* (GET): pagina per mostrare le previsioni di tutte le regioni;
- */previsioniRegionali/regione* (GET): indirizzo raggiunto per ottenere le province di una data regione.

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;
- *pw*: 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 resituito un *cookie* valido per 30 minuti e contenente 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 (importato con un partial file);
- una navbar contenente:
 1. il nome del sito (Meteo);
 2. un link alla Home in modo da poterci sempre tornare in qualunque momento;
 3. un link alla pagina per il meteo in tempo reale delle province di ciascuna regione;
 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. bottone per fare il toggle tra light e dark mode;
 6. barra di ricerca per ottenere il meteo in tempo reale di una città.

I punti 3 e 4 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 non sono stati integrati file *.css* personalizzati in quanto tali funzioni sono state assolve tramite le classi di bootstrap.

4.3 API

Nel progetto sono state integrate 3 API:

- *openWeather*: API usata per ottenere le condizioni meteo in tempo reale;
- *peexels*: API in grado di fornire immagini a partire da una query contenente un filtro che descriva il soggetto mostrato;
- *comuni ITA*: API in grado di fornire l'elenco dei comuni presenti in Italia, nonché delle regioni e delle province di ogni regione.

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/MatteoCelardo/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/MatteoCelardo/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

Come detto in precedenza, l'applicazione si occupa di fornire le condizioni meteo in tempo reale relative ad una o più città italiane.

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 comuni ITA: <https://comuni-ita.herokuapp.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/MatteoCelardo/progettoPWM>;
- bootstrap: <https://getbootstrap.com>