Corso di Tecnologie Web e Mobile A.A. 2022-2023

LogBook.com

Ameglio Alberto [950144]

Author: Ameglio Alberto
Last modified: 05/03/2023 - version 1.0.0
First created: 05/03/2023

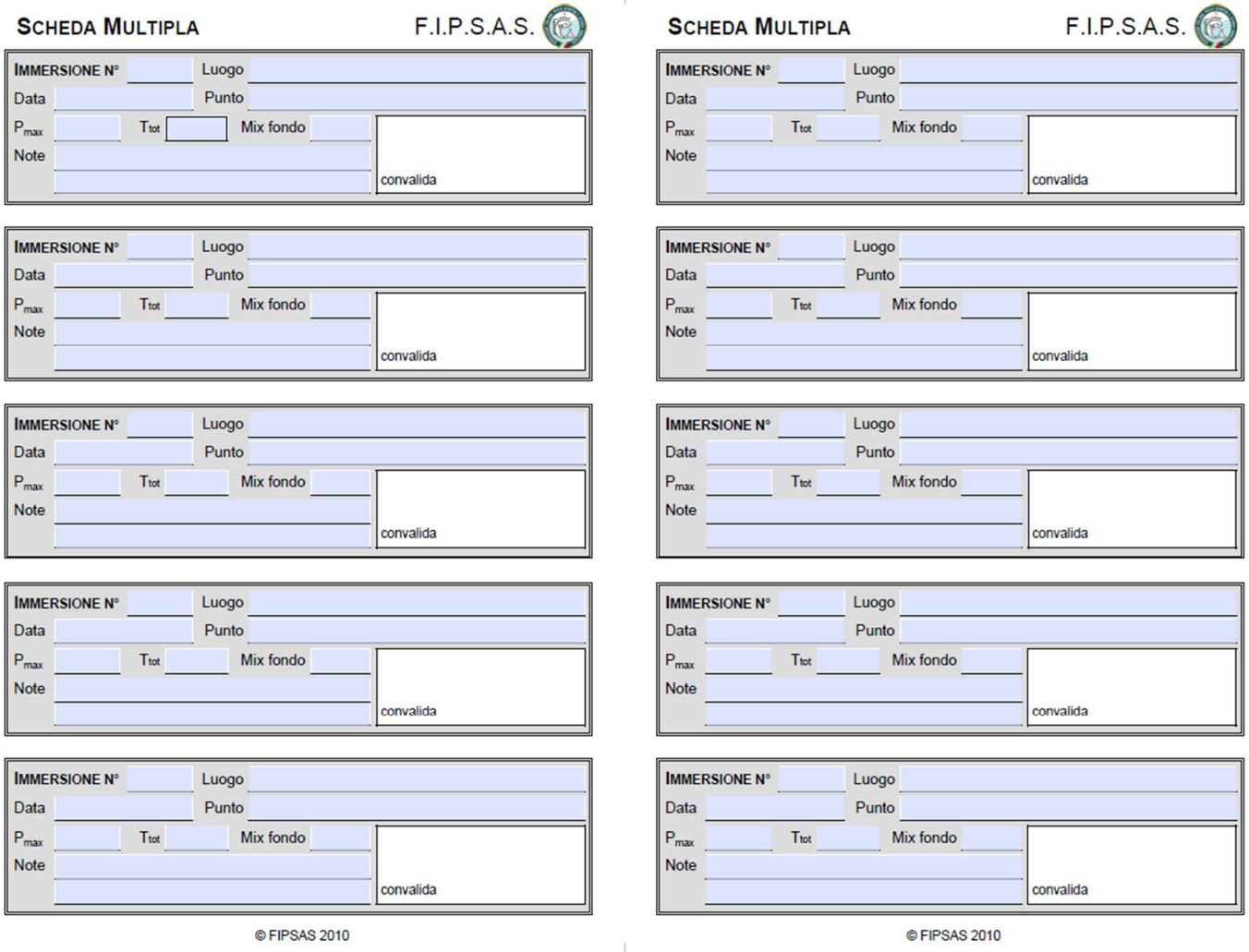
Indice

LogBook.com

LogBook online per il mantenimento delle proprie immersioni

Introduzione

Il progetto nasce da una esigenza legata ad una mia passione ovvero la subacquea. In questa disciplina sportiva è pratica comune annotare informazioni sull'attività svolta alla fine di una giornata di immersioni utilizzando supporti cartacei di questo tipo.



Questo può generare diversi problemi di seguito elencati:

- Mancata disponibilità del LogBook cartaceo
- Disponibilità di spazio per registrare nuove immersioni
- Limite al numero di annotazioni dato dal foglio prestampato
- Mancanza di spazio per attività particolari es. viaggi e lunghe trasferte

- La carta si può danneggiare perdendo lo storico delle attività
- Il LogBook o fogli ad esso integrato possono essere smarriti

Da questa esigenza è nato l'obiettivo di creare e sperimentare un prodotto informatico efficace sfruttando il più possibile i sistemi HTML5 e CSS3 e tutte le tecnologie client-side più moderne.

Al giorno d'oggi sembra infatti che la maggior parte dei programmatori Web dedichi la sua attenzione alle tecnologie di back-end (server-side), producendo servizi e applicativi efficienti ma nettamente carenti dal punto di vista dell'esperienza utente.

Non bisogna dimenticarsi che, per la natura pratica e materiale dell'essere umano, soprattutto nell'ambito delle applicazioni "consumer", la qualità della User Interface assume importanza predominante.

La soddisfazione utente deriva in primis dall'immediatezza e dal feeling legati all'interazione con un applicativo.

Solo successivamente, quando l'user sarà certo di aver operato correttamente (e questa sicurezza non può concretizzarsi, se l'interfaccia non è ben implementata), sopraggiungerà la soddisfazione nel vedere che il processo di elaborazione back-end delle informazioni è efficiente.

Senza una buona interfaccia front-end, quindi, è più difficile apprezzare la qualità della programmazione back-end.

La bontà di un'interfaccia consumer-oriented si misura in gran parte dalla sua auto-esplicatività.

Accentrare le operazioni, ridurre/guidare le scelte utente scremandole dal superfluo, ridurre il testo e aumentare l'iconografia descrittiva sono gli strumenti di cui si serve un abile programmatore Front-End per produrre una U.I. concretamente valida.

HTML, nelle sue declinazioni passate, si presta solo parzialmente a questi scopi, poiché non prevede, originariamente, facilitazioni per creare pagine marcatamente interattive.

Nasce così l'idea di creare una Web-App HTML5, pensata per essere utilizzata su devices con l'utilizzo di mouse o tastiera.

La Web-App in questione è stata denominata "LogBook.com", e svolge una funzione di registro dei dati delle immersioni.

Inoltre l'applicazione è dotata di un sistema di autentificazione che permette di utilizzare lo stesso device con un semplice utilizzo delle funzioni di login e logout e registrare online le immersioni di diversi utenti.

E' pensata per essere ospitata su un hardware dedicato, composto da un computer industriale low power e low cost integrato all'interno di un pannello dal design minimale.

Breve analisi dei requisiti

Destinatari

Il progetto è pensato per essere utilizzato da subacquei o apneisti, con particolare attenzione ai soggetti spiccatamente non pratici delle tecnologie informatiche. L'interfaccia, dunque, è stata resa auto-esplicante, facilmente applicabile anche nel suo primo utilizzo, senza necessità di alcun tipo di training pregresso. Al tempo stesso, però, e questa è stata la sfida maggiore, l'interfaccia risulta di piacevole utilizzo anche da parte di utenti esperti, poiché la sua semplicità non è conseguenza di limitazioni o compromessi fatti sulle funzionalità del prodotto, che risultano complete ed efficaci.

Questo porta a importanti conseguenze essendo il prodotto (software + hardware) vendibile a privati, a società sportive, a diving centers e agli addetti del settore.

Tutti questi soggetti potranno utilizzare efficacemente il prodotto, pur non avendo specifiche conoscenze informatiche.

Contenuti e modello di valore

Il valore del prodotto è dato dal cross-platform essendo compatibile e retro compatibile con tutti i sistemi presenti attualmente sul mercato.

L'idea su cui è basato il modello del valore è quella di usufruire con questa prima versione dei vantaggi dell'esternalità di rete ovvero nel diffonderne l'utilizzo agli addetti del settore. Per andare ad implementare nelle prossime versioni l'applicazione con maggiori informazioni per beneficiare dei vantaggi della digitalizzazione rispetto a un supporto cartaceo, fornendo tools per il calcolo della pianificazione della immersione o strumenti per lo studio del grafico dell'immersione.

Inoltre chi pratica questa disciplina è abituato ad utilizzare strumenti multimediali per il monitoraggio delle immersioni, quali computer subacquei, sonde, macchine fotografie o action cam, e l'idea per una prossima versione sarebbe quella di mettere a disposizione anche uno spazio di archiviazione dove l'utente potrà immagazzinare tutte i dati raccolti durante quella immersioni in modo da assicurarne il non smarrimento e un'archiviazione semplice e immediata.

Con lo sviluppo di questa funzione il modello del valore sarebbe dato da una prima versione con un limite di storage molto ridotto, con la possibilità successiva di essere integrata con un abbonamento per l'aggiunta di spazio di archiviazione extra, simile ai prodotti di cloud e di drive che si trovano oggi sul mercato.

Inoltre potrebbe essere integrata anche una funzione che permetta agli amministratori di un gruppo di utenti di monitorare e gestire le immersioni di più persone facilitando anche la gestione dei corsi formazioni da parte di istruttori o diving centers.

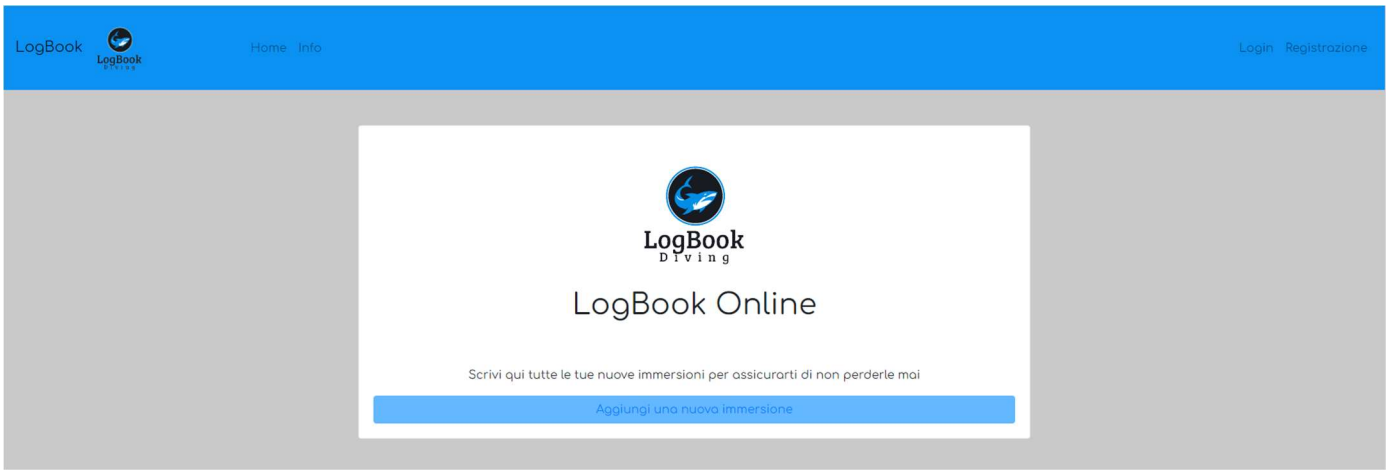
La scelta di creare un'architettura client-server, al posto che utilizzare tecnologie concepite per funzionare solo localmente, permette inoltre una grandissima espandibilità essendo possibile variare l'intera architettura server senza intaccare la funzionalità del front-end.

Un'architettura client-server, inoltre, apre la strada a future espansioni nelle quali più schermi faranno capo a un unico server, rendendo remoto il controllo di ogni singolo dispositivo.

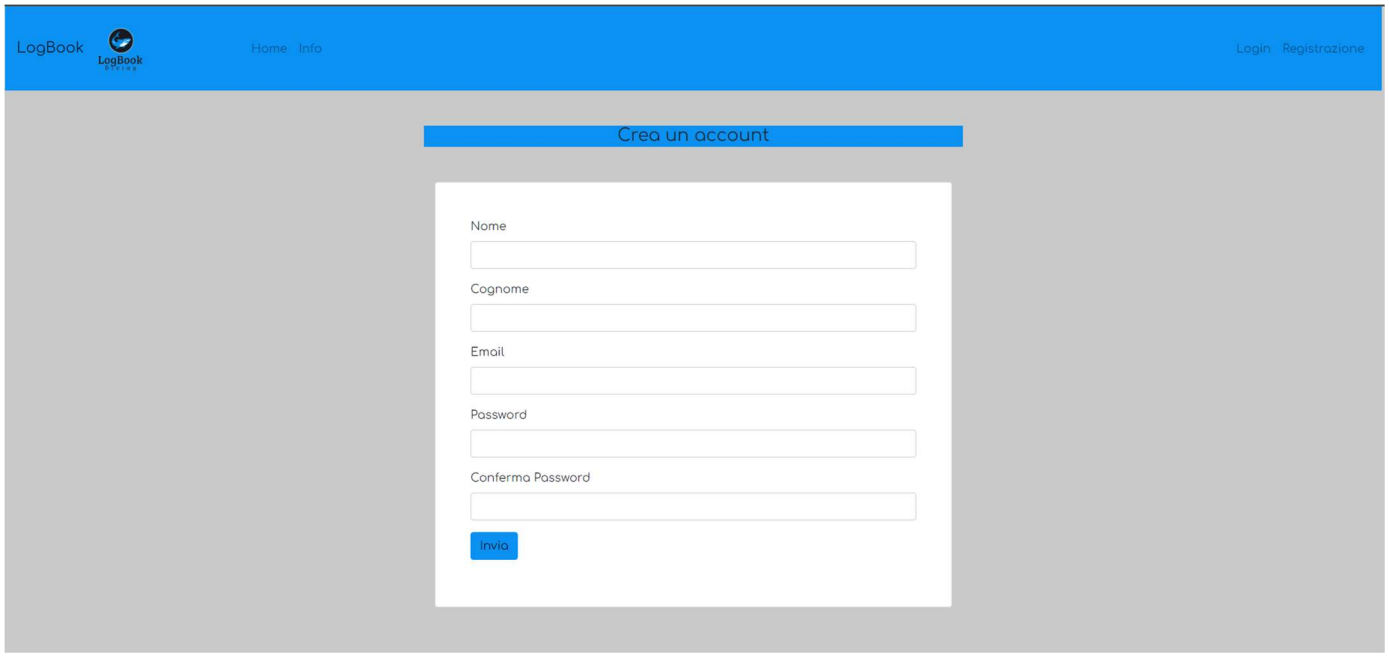
In sintesi, il concetto che più quantifica il valore e la bontà di questo progetto è la versatilità, sia di impiego che di implementazione. E questo è uno dei capisaldi delle tecnologie web.

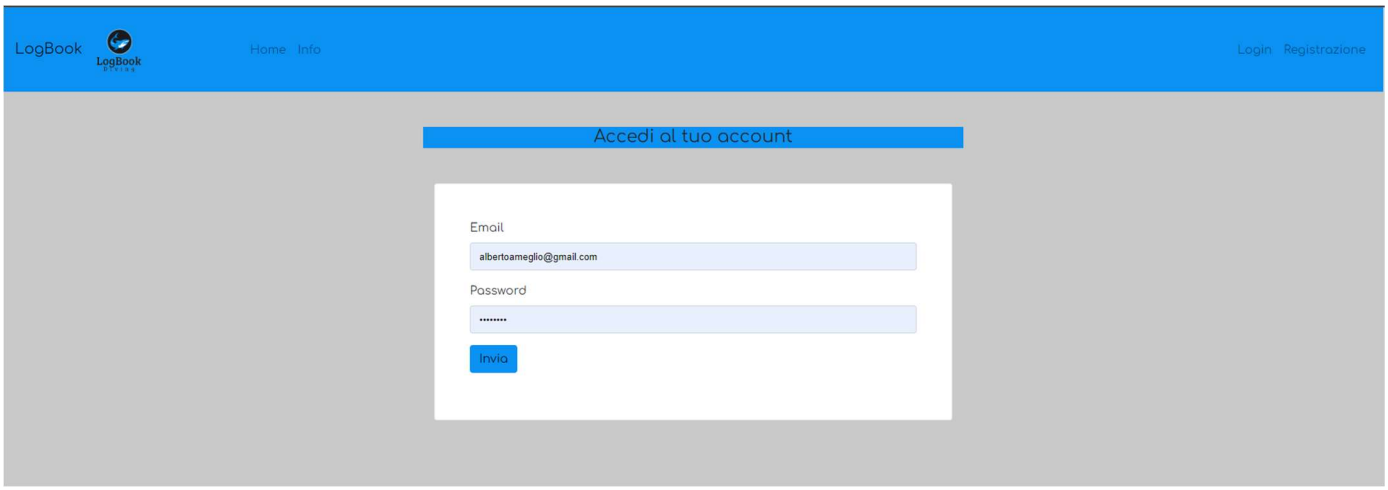
PANORAMICA DELL'INTERFACCIA UTENTE

➢ Index primo approccio all'app da parte dell'utente



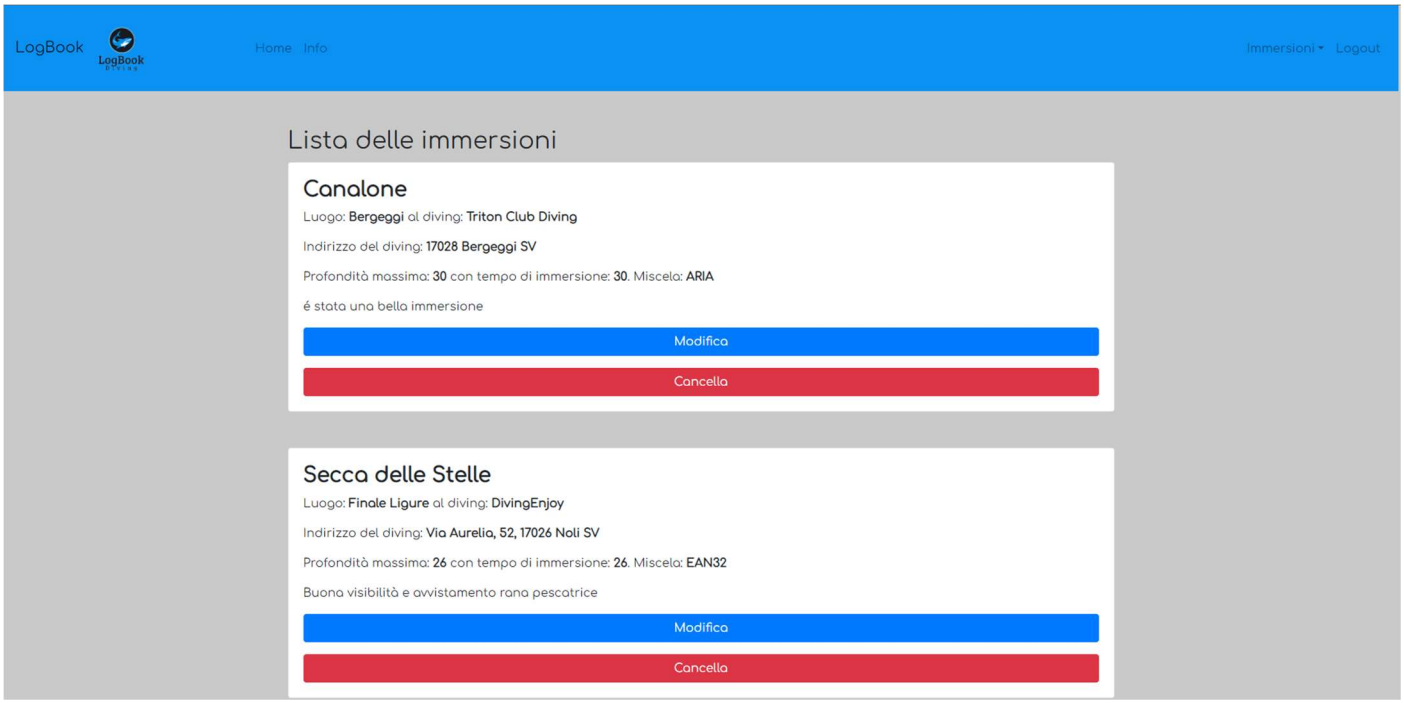
➢ Form di registrazione e login da parte dell'utente

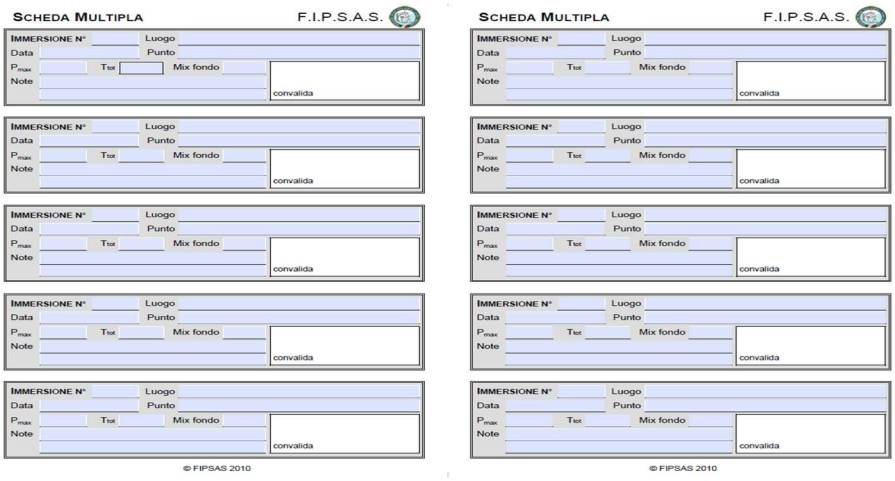




➢ Pagina di registrazione di una nuova immersione

➢ Pagina di visualizzazione della lista delle immersioni





Ci tengo a riportare di nuovo la scheda cartacea in modo da sottolineare meglio in lavoro che si è cercato di effettuare sulla User Experience

Aspetti tecnologici

Le tecnologie impiegate sono molteplici, frutto del desiderio di utilizzare le ultime evoluzioni dei linguaggi e degli strumenti a disposizione dei Web Developer più determinanti.

Grande attenzione è stata riservata nella scelta di ciascuna tecnologia, non solo un esercizio di stile ma un necessario strumento per l'operatività dell'applicazione.

Le tecnologie utilizzate sono state indentificate in base a un criterio di massima efficienza.

- **EXPRESS.JS**

Express.js, o semplicemente Express, è un framework per applicazioni web per Node.js, open source sotto Licenza MIT. È stato progettato per creare web application e API ed ormai definito il server framework standard de facto per Node.js.

E' un framework di backend per Node.js minimalista, veloce e simile a Sinatra che fornisce funzionalità e strumenti robusti per lo sviluppo di applicazioni di backend scalabili.

Fornisce il sistema di routing e funzionalità semplificate per estendere il framework, sviluppando componenti e parti più potenti a seconda dei casi d'uso dell'applicazione.

Il framework fornisce una serie di strumenti per applicazioni Web, richieste e risposte HTTP , routing e middleware per la creazione e la distribuzione di applicazioni aziendali su larga scala.

Fornisce inoltre uno strumento di interfaccia a riga di comando chiamato Node Package Manager (NPM), in cui gli sviluppatori possono procurarsi i pacchetti sviluppati, costringendoli così a seguire il principio **Don't Repeat Yourself (DRY).**

Il principio DRY mira a ridurre la ripetizione di schemi software, sostituendoli con astrazioni o utilizzando la normalizzazione dei dati per evitare la ridondanza.

Per questo progetto, partendo da Express, si è voluto utilizzare lo stack software MEVN

- **MEVN**

MEVN è uno stack software JavaScript gratuito e open source per la creazione di siti Web dinamici e applicazioni Web con i seguenti componenti:

1. **MongoDB:** MongoDB è il database NoSQL standard
2. **Express.js:** il framework di applicazioni Web predefinito per la creazione di app Web
3. **Vue.js**: il framework progressivo JavaScript è utilizzato per la creazione di applicazioni Web front-end ma non è stato implementato per questa app
4. **Node.js:** motore JavaScript utilizzato per applicazioni di rete e lato server scalabili.

- **NODE.JS**

Node.js è un ambiente di runtime JavaScript basato su V8, l'interprete JavaScript open source di Google utilizzato anche in Google Chrome. Node.js consente l'esecuzione di codice JavaScript su server-side, cioè sul lato server anziché sul lato client del web, server di streaming, applicazioni di elaborazione dati in tempo reale, applicazioni di elaborazione batch, applicazioni di automazione.

Node.js è costruito su un modello di I/O non bloccante, il che significa che le operazioni di input/output non bloccano il thread principale dell'applicazione, consentendo l'elaborazione asincrona e l'utilizzo efficiente delle risorse del sistema. Questo modello è implementato tramite il meccanismo di callback e l'API di eventi di Node.js.

Node.js fornisce anche un vasto ecosistema di moduli e pacchetti open source, disponibili tramite il suo gestore di pacchetti npm, che consente di installare e gestire le dipendenze di un progetto in modo facile e sicuro. Questo ecosistema di moduli copre una vasta gamma di funzionalità, tra cui server HTTP, middleware, librerie per la manipolazione di dati, librerie per la criptografia, librerie per la gestione di file, e molto altro.

- **MONGODB**

MongoDB è un sistema di gestione di database non relazionali, noto anche come database NoSQL. È stato progettato per gestire grandi quantità di dati in modo efficiente e scalabile, offrendo una maggiore flessibilità rispetto ai tradizionali database relazionali.
MongoDB utilizza un modello di dati basato su documenti. I dati sono memorizzati in documenti JSON (JavaScript Object Notation) all'interno di collezioni, mentre in un database relazionale si utilizzano tabelle. Ciò significa che i documenti possono contenere campi con diverse strutture e tipi di dati, consentendo una maggiore flessibilità nella modellazione degli stessi.
MongoDB è altamente scalabile e può gestire facilmente grandi volumi di dati distribuendo le richieste su un cluster di server. Grazie alla sua architettura, di tipo distribuito, consente di suddividere i dati in più nodi, ciascuno dei quali può essere replicato per garantire la disponibilità e la tolleranza ai guasti.
MongoDB offre un'ampia gamma di funzionalità avanzate, tra cui la ricerca testuale, l'aggregazione dei dati provenienti da più sorgenti, l'indicizzazione di testo completo e altro ancora. Inoltre, MongoDB supporta molte lingue di programmazione, inclusi JavaScript, Python, Java e C#, e dispone di una vasta gamma di driver di database e librerie di connettività.

- **HTML5**

HTML5 (HyperText Markup Language version 5) è la versione più recente del linguaggio di markup utilizzato per la creazione e la strutturazione di contenuti per il web. È stata rilasciata nel 2014 e offre una vasta gamma di nuove funzionalità e miglioramenti rispetto alle versioni precedenti di HTML, per questo progetto HTML5 è stato impiegato nella totalità dell'interfaccia utente.
I principali vantaggi di HTML5 rispetto alle precedenti versioni di HTML includono un miglior supporto per i media, una maggiore semantica, una migliore accessibilità, la validazione dei form lato client, il supporto per le applicazioni web e un miglioramento nella gestione della grafica. Questi miglioramenti rendono la creazione e la gestione dei contenuti web più efficiente, facile e avanzata. Inoltre HTML5 è progettata per migliorare la SEO rispetto alle versioni precedenti di HTML.

- **CSS3**

Il CSS (sigla di Cascading Style Sheets, in italiano fogli di stile a cascata), in informatica, è un linguaggio usato per definire la formattazione di documenti HTMLI principali vantaggi di CSS3 rispetto alle precedenti versioni includono una maggiore flessibilità e controllo con nuovi selettori e proprietà, il supporto nativo per le transizioni e le animazioni, il supporto per le media query per adattare lo stile del contenuto a diversi dispositivi, miglioramenti nella gestione dei font e delle immagini con nuove proprietà. Questi miglioramenti rendono la creazione e la gestione degli stili dei contenuti web più avanzati e precisi.

- **BOOTSTRAP**

  Bootstrap è un framework front-end open source per la creazione di siti web e applicazioni web responsivi e mobile-first. È stato sviluppato da Twitter e offre una vasta gamma di componenti UI, modelli di layout, tipografia e stili predefiniti per semplificare la progettazione e lo sviluppo di siti web moderni.
  Bootstrap utilizza una combinazione di HTML, CSS e JavaScript per offrire una vasta gamma di funzionalità, tra cui componenti UI come pulsanti, forme, modali, menu a discesa, e schede.

  Fornisce inoltre modelli di layout predefiniti, come griglie, container e spazi vuoti, per semplificare la progettazione di siti web responsivi.

  Bootstrap è altamente personalizzabile: nel design, nei colori e nello stile del proprio sito web; dispone di una vasta gamma di plugin ed estensioni, come datepicker, caroselli, mappe, che possono essere facilmente integrati nel proprio sito web.

  Grazie alla sua semplicità, flessibilità e vasta gamma di funzionalità, Bootstrap è diventato uno dei framework front-end più popolari e utilizzati per la creazione di siti web moderni e responsivi.

- **EXPRESS-HANDLEBARS 3.0.0**

  Express-handlebars è un motore di template per Node.js, che consente di generare dinamicamente pagine HTML basate su template predefiniti. È basato sul motore di template Handlebars e offre una sintassi facile da usare e da comprendere.

  Express-handlebars è progettato per essere utilizzato con il framework web Node.js Express, ma può essere utilizzato anche in modo indipendente. Esso consente di definire layout predefiniti, blocchi di contenuto, componenti e variabili da utilizzare nelle pagine generate dinamicamente.

  Esso fornisce una sintassi basata su tag HTML, con l'aggiunta di elementi speciali per la definizione di variabili, blocchi di contenuto e componenti. Ciò rende il codice del template molto più facile da leggere e da comprendere rispetto ad altri motori di template.

  Express-handlebars supporta anche la localizzazione, consentendo di generare pagine in base alla lingua o alla regione dell'utente. Inoltre, consente anche l'utilizzo di helper, che sono funzioni JavaScript personalizzate utilizzate per la manipolazione dei dati all'interno del template.

  Grazie alla sua facilità d'uso e alla sua sintassi intuitiva, express-handlebars è diventato uno dei motori di template più popolari per Node.js. Esso è utilizzato in una vasta gamma di applicazioni web per la generazione di pagine dinamiche e personalizzate.

- **EXPRESS-SESSION 1.15.6**

  Express-session è un middleware per il framework web Node.js Express, che consente di gestire le sessioni degli utenti all'interno di un'applicazione web. Esso fornisce un meccanismo per memorizzare le informazioni relative alla sessione dell'utente, come ad esempio l'ID della sessione, all'interno di un cookie o di uno storage persistente.

  Express-session fornisce un'interfaccia semplice per la gestione delle sessioni, consentendo di impostare e ottenere informazioni sulla sessione dell'utente in modo facile e sicuro. Gestisce in modo automatico l'invio e la ricezione dei cookie e si occupa della scadenza della sessione in modo da garantire la sicurezza delle informazioni dell'utente.

  Express-session supporta anche la configurazione di parametri avanzati, come ad esempio il tempo massimo di inattività consentito per la sessione o la crittografia delle informazioni della sessione. Inoltre supporta, la memorizzazione delle sessioni in memoria, su disco o in un database, offrendo una maggiore flessibilità nella scelta dello storage delle informazioni della sessione.
  Express-session è uno strumento essenziale per la gestione delle sessioni in un'applicazione web Node.js.

- **MONGOOSE 4.13.0**

  Mongoose è una libreria di oggetti di modellazione di documenti MongoDB per Node.js, che fornisce una soluzione semplice e potente per l'interazione con MongoDB. Essa consente di definire modelli di dati per i documenti MongoDB, offrendo una sintassi semplice e intuitiva per la creazione, la modifica, la ricerca e la cancellazione di documenti all'interno del database MongoDB.

  Mongoose fornisce un'interfaccia ad alto livello per l'interazione con MongoDB, semplificando l'accesso e la manipolazione dei dati all'interno del database. Essa consente di definire uno schema per i documenti MongoDB, offrendo una struttura rigida per i dati che rende più facile la validazione dei dati e la prevenzione di errori di inserimento.

  Mongoose supporta anche la validazione dei dati, offrendo una vasta gamma di funzionalità per la definizione di vincoli sui dati inseriti, come ad esempio il controllo della presenza di campi obbligatori, il controllo della lunghezza dei campi e il controllo della presenza di valori validi per i campi. Fornisce inoltre un sistema di middleware, consentendo l'aggiunta di funzionalità personalizzate alla logica di business delle applicazioni web. Questo permette di estendere le funzionalità di base offerte da Mongoose, permettendo una maggiore flessibilità nella gestione dei dati.

  Grazie alla sua facilità d'uso e alla sua vasta gamma di funzionalità, Mongoose è diventata una delle librerie più popolari per l'interazione con MongoDB all'interno di applicazioni web Node.js.. Viene utilizzata in una vasta gamma di applicazioni web, offrendo una soluzione potente e flessibile per la gestione dei dati all'interno di MongoDB.

- **BODY-PARSER 1.18.2**

  Body-parser è un middleware per il framework web Node.js Express, che consente di analizzare il corpo di una richiesta HTTP e di estrarre i dati al suo interno. Esso fornisce un meccanismo semplice e potente per gestire i dati inviati tramite richieste HTTP POST e PUT, consentendo di accedere a tali dati all'interno dell'applicazione web.
  Body-parser supporta diversi formati di dati, tra cui JSON, URL-encoded e multipart/form-data, offrendo una grande flessibilità nella gestione dei dati inviati tramite richieste HTTP. Esso consente di analizzare il corpo di una richiesta HTTP e di estrarre i dati al suo interno, consentendo di accedere a questi dati all'interno dell'applicazione web in modo facile e sicuro. Supporta inoltre la configurazione di parametri avanzati, come ad esempio la dimensione massima dei dati che possono essere inviati tramite una richiesta HTTP o la possibilità di analizzare solo parti specifiche del corpo della richiesta.
  Body-parser è uno strumento essenziale per la gestione dei dati inviati tramite richieste HTTP POST e PUT in un'applicazione web Node.js.

- **CONNECT-FLASH 0.1.1**

  Connect-flash è un middleware per il framework web Node.js Express, che consente di memorizzare messaggi di notifica temporanei e di mostrarli all'utente su richiesta. Offre un meccanismo semplice e potente per la gestione dei messaggi di notifica all'interno di un'applicazione web, consentendo di fornire feedback all'utente sulle azioni che ha appena eseguito.

  Connect-flash memorizza i messaggi di notifica nella sessione dell'utente, consentendo di mostrarli all'utente solo una volta e di rimuoverli immediatamente dopo che sono stati visualizzati. Ciò consente di fornire una user experience più fluida e di evitare la ripetizione dei messaggi di notifica.
  Inoltre, connect-flash fornisce anche un'interfaccia semplice per la gestione dei messaggi di notifica, consentendo di inserire, visualizzare e rimuovere messaggi in modo facile e sicuro. Esso supporta anche la configurazione di parametri avanzati, come ad esempio la durata massima di conservazione dei messaggi di notifica.

  Connect-flash è uno strumento essenziale per la gestione dei messaggi di notifica in un'applicazione web Node.js.

- **METHOD-OVERRIDE 2.3.10**

  Method-override è un middleware per il framework web Node.js Express, che consente di utilizzare metodi HTTP non supportati da HTML, come PUT e DELETE, attraverso richieste POST. Esso offre un meccanismo semplice e potente per utilizzare questi metodi in un'applicazione web, consentendo di gestire operazioni come l'aggiornamento e l'eliminazione di risorse in modo corretto.
  Method-override funziona aggiungendo un campo nascosto nel corpo della richiesta POST, contenente il metodo HTTP desiderato. Questo campo può essere letto dal middleware per sovrascrivere il metodo effettivo della richiesta, consentendo di eseguire operazioni come se il metodo HTTP corretto fosse stato utilizzato.
  Inoltre, method-override, supporta anche la configurazione di parametri avanzati, come ad esempio la definizione di una mappa di metodi personalizzata, consentendo di personalizzare il funzionamento del middleware in modo facile e sicuro.
  Method-override è uno strumento essenziale per la gestione di metodi HTTP non supportati da HTML in un'applicazione web Node.js.

- **PASSPORT 0.4.0 – PASSPORT LOCAL 1.0.0**

  Passport è un middleware per il framework web Node.js Express, che consente di gestire l'autenticazione degli utenti in modo facile e sicuro. Esso offre un'ampia gamma di strategie di autenticazione, che consentono di integrare l'autenticazione con diversi servizi, come ad esempio Facebook, Google e Twitter.

  Passport utilizza un approccio modulare, consentendo di utilizzare solo le strategie di autenticazione necessarie per l'applicazione web. Ciò consente di personalizzare l'autenticazione in base alle esigenze specifiche dell'applicazione, consentendo di ottenere un'esperienza utente più personalizzata.

  Inoltre, Passport fornisce anche un'interfaccia semplice per la gestione delle credenziali degli utenti, consentendo di verificare le credenziali e di generare sessioni di autenticazione in modo facile e sicuro.
  Passport-local è una strategia di autenticazione per Passport, che consente di gestire l'autenticazione degli utenti tramite username e password. Essa offre un meccanismo semplice e potente per la gestione delle credenziali degli utenti, consentendo di verificare le credenziali e di generare sessioni di autenticazione in modo facile e sicuro.

  Passport e Passport-local sono strumenti essenziali per la gestione dell'autenticazione degli utenti in un'applicazione web Node.js..

- **BCRYPTJS 2.4.3**

  Bcryptjs è una libreria Node.js per la crittografia delle password, che utilizza l'algoritmo bcrypt. Essa consente di proteggere le password degli utenti in modo sicuro e affidabile, utilizzando un hash unidirezionale per criptare la password.

  Bcryptjs offre un meccanismo semplice e potente per la crittografia delle password, consentendo di definire un parametro di lavoro (o cost factor) per regolare il tempo di elaborazione dell'algoritmo di hash. Ciò consente di aumentare la sicurezza delle password, rendendo più difficile per gli attaccanti craccare le password utilizzando attacchi di forza bruta.

  Inoltre, bcryptjs fornisce anche un'interfaccia semplice per la generazione dei salt, che sono utilizzati per aggiungere casualità alla password crittografata, rendendola ancora più sicura.

  Bcryptjs è uno strumento essenziale per la crittografia delle password in un'applicazione web Node.js.

- **JAVASCRIPT**

Il motore Javascript di Webkit è molto potente, e permette di avere una responsività dell'U.I. davvero notevole, soprattutto considerando la complessità del codice e i requirements molto limitativi dell'hardware low cost.

Tuttavia, a causa di queste ottimizzazioni nel motore di scripting, bisogna porre particolare attenzione all'insorgere di eventuali race-conditions nel codice, o di eventuali deadlocks.

Javascript infatti è single-threaded, ma gli eventi utente possono dare luogo a una queue di processamento che, a seconda delle implementazioni del motore di scripting (da ricordarsi che non sono open-source e non esenti da bug!), può dare luogo a esecuzioni di codice falsamente parallele, dove in alcuni casi due parti di codice Javascript vengono richiamate ed eseguite in time-sharing.

Questi conflitti sono pericolosi e vanno gestiti con semafori o con controlli, per evitare comportamenti imprevedibili dell'interfaccia utente.

Normalmente in un sito web la vita della singola pagina (e di conseguenza dell'istanza del motore Javascript) è breve, quindi eventuali errori o leak muoiono al primo refresh.

In un'interfaccia totalmente asincrona, la vita media della medesima pagina web è elevatissima, e bisogna evitare condizioni di deadlock, pena il blocco totale dell'interattività nella pagina o addirittura al crash del motore Javascript, fino a un forzato fresh che carichi una nuova istanza dello script

Architettura

Diagramma dell'ordine gerarchico delle risorse

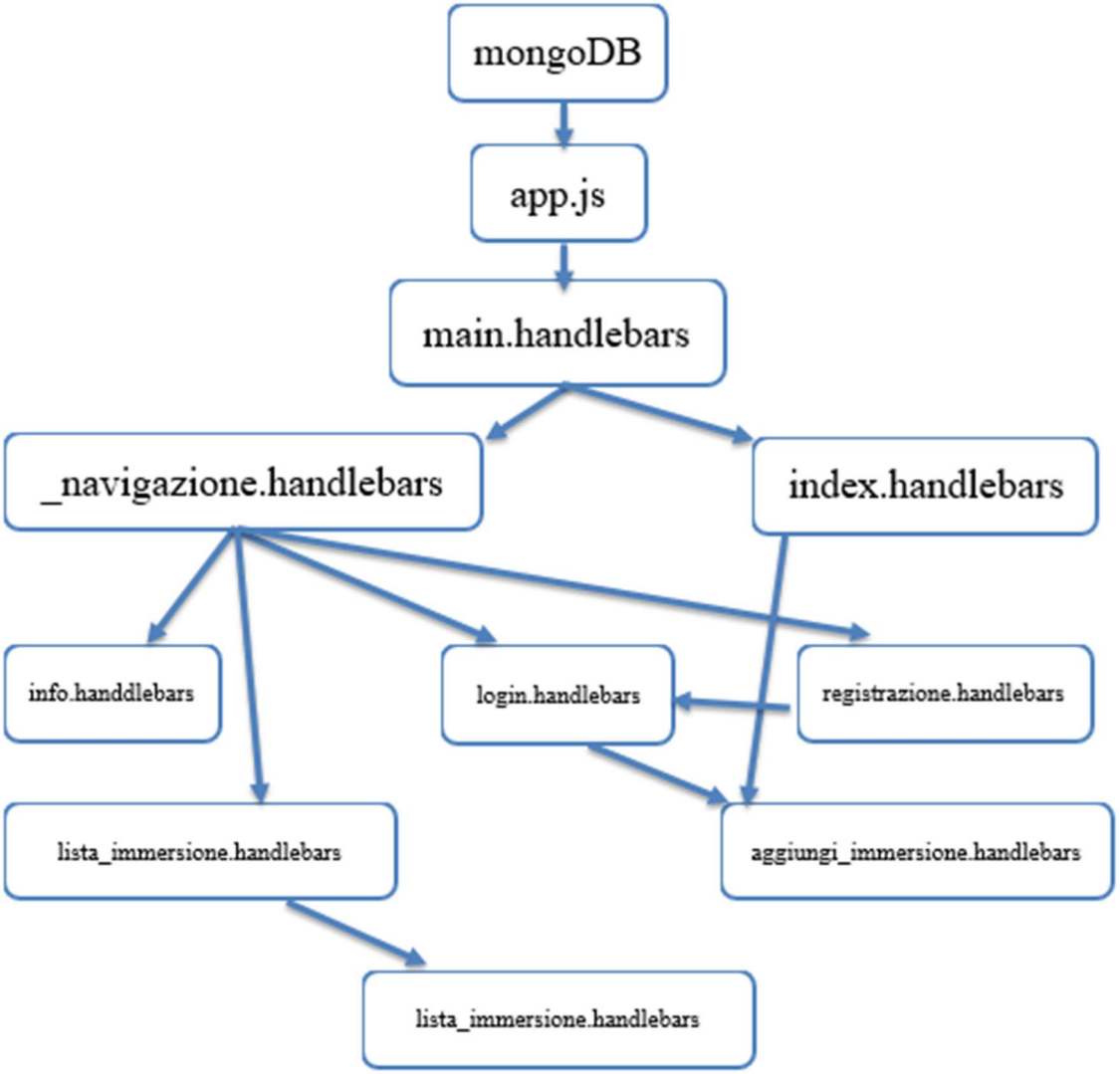


Figura 1: Architettura del sito dal punto di vista del Client

Codice

Frammenti del codice più significativo

HTML

```html
<!DOCTYPE html>
<html lang="it">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <meta name="keywords" content="LogBook,logbook,divebook,DiveBook" />
    <meta name="description" content="LogBook Diving online" />
    <meta name="author" content="Ameglio Alberto" />
    <title>LogBook</title>
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0-beta.2/css/bootstrap.min.css"
        integrity="sha384-PsH8R72JQ3SOdhVi3uxftmaW6Vc51MKb0q5P2rRUpPvrszuE4W1povHYgTpBfshb" crossorigin="anonymous">
    <link rel="stylesheet" href="css/custom.css">
    <link href="https://fonts.googleapis.com/css?family=Comfortaa" rel="stylesheet">
</head>

<body>
    <div class="container-fluid">
        {{> _navigazione}}
    </div>
    <div class="container">
        {{> _avvisi}}
        {{> _errori}}
        {{{body}}}
    </div>

    <script src="https://code.jquery.com/jquery-3.2.1.slim.min.js"
        integrity="sha384-KJ3o2DKtIkvYIK3UENzmM7KCkRr/rE9/Qpg6aAZGJwFDMVNA/GpGFF93hXpG5KkN"
        crossorigin="anonymous"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.3/umd/popper.min.js"
        integrity="sha384-vFJXuSJphROIrBnz7yo7oB41mKfc8JzQZiCq4NCceLEaO4IHwicKwpJf9c9IpFgh"
        crossorigin="anonymous"></script>
    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0-beta.2/js/bootstrap.min.js"
        integrity="sha384-alpBpkh1PFOepccYVYDB4do5UnbKysX5WZXm3XxPqe5iKTfUKjNkCk9SaVuEZflJ"
        crossorigin="anonymous"></script>
</body>

</html>
```

Formattazione conforme con HTML5 aggiunta dei meta tag necessari anche per la gestione della SEO integrazione dello stylesheet di Bootstrap nello head e caricamento degli script java in fondo al main in modo da velocizzare il caricamento.
In questa porzione vengono integrate tre librerie:

- La prima libreria è jQuery, una popolare libreria JavaScript utilizzata per semplificare la scrittura di codice JavaScript e rendere più facile la manipolazione del DOM.
- La seconda libreria è Popper.js, una libreria utilizzata per gestire la posizione e il comportamento degli elementi "popper" nel layout della pagina web.
- Infine, la terza libreria è Bootstrap, una libreria CSS e JavaScript utilizzata per creare layout responsivi, componenti UI e interazioni interattive su pagine web.

Queste librerie sono state importate da fonti esterne utilizzando i rispettivi URL e includono anche degli attributi di integrità e di cross origin per garantire che i file siano stati scaricati in modo sicuro e non siano stati modificati.

CSS3

```css
body{
    background-color: #b0b0b0ac;
    font-family: Comfortaa, sans-serif;
}

img{max-width: 30%}
```

SCHEMA MONGOOSE IMMERSIONI

```javascript
const mongoose = require('mongoose');

const ImmersioniSchema = mongoose.Schema({
    sito: {
        type: String,
        required: true
    },
    luogo: {
        type: String,
        required: true
    },
    diving: {
        type: String,
        required: true
    },
    indirizzo: {
        type: String,
        required: true
    },
    pmax: {
        type: String,
        required: true
    },
    tmax: {
        type: String,
        required: true
    },
    mixfondo: {
        type: String,
        required: true
    },
    contenuto: {
        type: String,
        required: true
    },
    utente: {
        type: String,
        required: true
    },
    data: {
        type: Date,
        default: Date.now
    }
})

mongoose.model('immersioni', ImmersioniSchema);
```

SCHEMA MONGOOSE UTENTE

```
1   const mongoose = require('mongoose');
2   const Schema = mongoose.Schema;
3
4   const UtentiSchema = new Schema({
5       nome: {
6           type: String,
7           required: true
8       },
9       cognome: {
10          type: String,
11          required: true
12      },
13      email: {
14          type: String,
15          required: true
16      },
17      password: {
18          type: String,
19          required: true
20      },
21      data: {
22          type: Date,
23          default: Date.now
24      }
25  })
26
27  mongoose.model('utenti', UtentiSchema);
```

CONNESSIONE MONGOOSE

```
24  //CONNESSIONE A MONGOOSE
25  mongoose.Promise = global.Promise;
26  mongoose.connect('mongodb://localhost/immersioni', {
27      useMongoClient: true,
28  })
29      .then(() => console.log(' Server connesso'))
30      .catch(err => console.log(err));
31
```

LISTEN MONGODB

```
304   const port = process.env.PORT || 3000;
305   app.listen(port, () => {
306       console.log(`server attivato sulla porta: ${port}`);
307   })
```

NAVBAR

```
<nav class="navbar navbar-expand-lg navbar-light mb-5" style="background-color: #0a91f2">
  <a class="navbar-brand" href="/">LogBook <b>     </b><img src="img/logo1.png" class="responsive" alt="">
  </a>
  <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarSupportedContent"
    aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation">
    <span class="navbar-toggler-icon"></span>
  </button>

  <div class="collapse navbar-collapse" id="navbarNav">
    <ul class="navbar-nav">
      <li class="nav-item">
        <a class="nav-link" href="/">Home</a>
      </li>
      <li class="nav-item">
        <a class="nav-link" href="/info">Info</a>
      </li>
    </ul>
    <ul class="navbar-nav ml-auto">
      {{#if user}}
      <li class="nav-item dropdown">
        <a href="#" class="nav-link dropdown-toggle" data-toggle="dropdown" id="navbarDropdownMenuLink">Immersioni</a>
        <div class="dropdown-menu">
          <a href="/lista_immersioni" class="dropdown-item">Lista Immersioni</a>
          <a href="/aggiungi_immersione" class="dropdown-item">Nuova Immersione</a>
        </div>
      </li>
      <li class="nav-item">
        <a href="/logout" class="nav-link">Logout</a>
      </li>
      {{else}}
      <li class="nav-item">
        <a href="/login" class="nav-link">Login</a>
      </li>
      <li class="nav-item">
        <a href="/registrazione" class="nav-link">Registrazione</a>
      </li>
      {{/if}}
    </ul>
  </div>
  </div>
</nav>
```

REQUIRE MODULI E RISORSE ESTERNE AD APP.JS

```
1   const express = require('express');
2   const exphbs = require('express-handlebars');
3   const bodyParser = require('body-parser');
4   const flash = require('connect-flash');
5   const session = require('express-session');
6   const methodOverride = require('method-override');
7   const bcrypt = require('bcryptjs');
8   const passport = require('passport');
9   const mongoose = require('mongoose');
10
11  const { accessoSicuro } = require('./helpers/accesso_privato');
12
13  const app = express();
14
15  //CARTELLE PER GESTIONE DELLE RISORSE STATICHE
16  app.use('/css', express.static(__dirname + '/assets/css'));
17  app.use('/img', express.static(__dirname + '/assets/img'));
18
19  //INTEGRAZIONE FILE CONFIG PASSPORT
20  require('./config/passport')(passport);
21
```

IMPLEMENTAZIONE DEL ROUTE DELLE PAGINE

```
86  //ROUTE PER PAGINA INDEX
87  app.get('/', (req, res) => {
88      const titolo = "LogBook Online";
89      res.render('index', { titolo: titolo });
90  })
91
92  //ROUTE PER PAGINA INFO
93  app.get('/info', (req, res) => {
94      res.render('info');
95  })
96
97  //ROUTE PER PAGINA LISTA IMMERSIONI
98  app.get('/lista_immersioni', accessoSicuro, (req, res) => {
99      Immersioni.find({ utente: req.user.id })
100         .sort({ date: 'desc' })
101         .then(immersioni => {
102             res.render('lista_immersioni', {
103                 immersioni: immersioni
104             });
105         });
106 });
107
108 //ROUTE PER PAGINA AGGIUNGI IMMERSIONI
109 app.get('/aggiungi_immersione', accessoSicuro, (req, res) => {
110     res.render('aggiungi_immersione');
111 })
```

```
113  //ROUTE PER PAGINA LOGIN
114  app.get('/login', (req, res) => {
115      res.render('login');
116  })
117
118  //ROUTE PER PAGINA REGISTRAZIONE
119  app.get('/registrazione', (req, res) => {
120      res.render('registrazione');
121  })
122
123
124  //ROUTE PER PAGINA MODIFICA IMERSIONE
125  app.get('/modifica_immersione/:id', accessoSicuro, (req, res) => {
126      Immersioni.findOne({
127          _id: req.params.id
128      })
129          .then(immersione => {
130              if (immersione.utente != req.user.id) {
131                  req.flash('msg_errore', 'Non puoi vedere questi contenuti');
132                  res.redirect('/lista_immersioni');
133              } else {
134                  res.render('modifica_immersione', {
135                      immersione: immersione
136                  });
137              }
138          });
139  });
140
```

INTEGRAZIONE DEI MIDDLEWARE

```
41  //MIDDLEWARE PER HANDLEBARS
42  app.engine('handlebars', exphbs({ defaultLayout: 'main' }));
43  app.set('view engine', 'handlebars');
44
45  //BODY PARSER MIDDLEWARE
46  app.use(bodyParser.urlencoded({ extended: false }));
47  app.use(bodyParser.json());
48
49  //OVERRIDE MIDDLEWARE
50  app.use(methodOverride('_method'));
51
52  //MIDDLEWARE PER SESSIONE
53  app.use(session({
54      secret: 'keyboard cat',
55      resave: false,
56      saveUninitialized: false,
57  }));
58
59  //PASSPORT MIDDLEWARE
60  app.use(passport.initialize());
61  app.use(passport.session());
62
63
64  //MIDDLEWARE PER MESSAGGI FLASH
65  app.use(flash());
66
67  //VARIABILI GLOBALI PER MESSAGGI FLASH
68  app.use((req, res, next) => {
69      res.locals.msg_successo = req.flash('msg_successo');
70      res.locals.msg_errore = req.flash('msg_errore');
71      res.locals.error = req.flash('error');
72      res.locals.user = req.user;
73      next();
74  });
```

FORM AGGIUNTA E DI MODIFICA IMMERSIONE

```
<h3><b>Modifica le tue immersioni</b></h3>

<div class="card card-body mt-5">
  <form action="/lista_immersioni/{{immersione.id}}?_method=POST" class="row g-3" method="post">
    <input type="hidden" name="_method" value="POST">
    <div class="form-group col-12">
    <label for="sito"><b>Immersione</b></label>
    <input type="text" class="form-control" name="sito" value="{{immersione.sito}}">
    </div>
    <div class="form-group col-md-6">
      <label for="luogo" class="form-label"><b>Luogo</b></label>
      <input type="text" class="form-control" name="luogo" value="{{immersione.luogo}}">
    </div>
    <div class="form-group col-md-6">
      <label for="diving" class="form-label"><b>Diving</b></label>
      <input type="text" class="form-control" name="diving" value="{{immersione.diving}}">
    </div>
    <div class="form-group col-12">
      <label for="indirizzo" class="form-label"><b>Indirizzo Diving</b></label>
      <input type="text" class="form-control" name="indirizzo" value="{{immersione.indirizzo}}">
    </div>
    <div class="form-group col-md-4">
      <label for="pmax" class="form-label"><b>Profondità massima</b></label>
      <input type="text" class="form-control" name="pmax" value="{{immersione.pmax}}">
    </div>
    <div class="form-group col-md-4">
      <label for="tmax" class="form-label"><b>Tempo totale</b></label>
      <input type="text" class="form-control" name="tmax" value="{{immersione.tmax}}">
    </div>
    <div class="form-group col-md-4">
      <label for="mixfondo" class="form-label"><b>Miscela di fondo</b></label>
      <input type="text" class="form-control" name="mixfondo" value="{{immersione.mixfondo}}">
    </div>
    <div class="form-grou col-12">
      <label for="contenuto"><b>Note</b></label>
      <textarea class="form-control" name="testo">{{immersione.contenuto}}</textarea>
    </div>
    <div class="form-grou col-12"></div><div class="form-grou col-12"></div><div class="form-grou col-12"></div>
    <div class="form-group col-12">
    <button type="submit" class="btn"  style="background-color: #0a91f2" name="aggiungi">Invia</button>
    </div>
  </form>
</div>
```

```
<h2 class="my-5"><b>Aggiungi una nuova Immersione</b></h2>

<form action="aggiungi_immersione" class="row g-3" method="post">
  <div class="form-group col-12">
    <label for="sito"><b>Immersione</b></label>
    <input type="text" class="form-control" name="sito">
  </div>
  <div class="form-group col-md-6">
    <label for="luogo" class="form-label"><b>Luogo</b></label>
    <input type="text" class="form-control" name="luogo">
  </div>
  <div class="form-group col-md-6">
    <label for="diving" class="form-label"><b>Diving</b></label>
    <input type="text" class="form-control" name="diving">
  </div>
  <div class="form-group col-12">
    <label for="indirizzo" class="form-label"><b>Indirizzo Diving</b></label>
    <input type="text" class="form-control" name="indirizzo">
  </div>
  <div class="form-group col-md-4">
    <label for="pmax" class="form-label"><b>Profondità massima</b></label>
    <input type="text" class="form-control" name="pmax">
  </div>
  <div class="form-group col-md-4">
    <label for="tmax" class="form-label"><b>Tempo totale</b></label>
    <input type="text" class="form-control" name="tmax">
  </div>
  <div class="form-group col-md-4">
    <label for="mixfondo" class="form-label"><b>Miscela di fondo</b></label>
    <input type="text" class="form-control" name="mixfondo">
  </div>
  <div class="form-group col-12">
    <label for="contenuto"><b>Note</b></label>
    <textarea class="form-control" name="testo"></textarea>
  </div>
  <div class="col-12">
    <button type="submit" class="btn" style="background-color: #0a91f2" name="aggiungi">Invia</button>
  </div>
</form>
```

GESTIONE FORM IMMERSIONI

```
//GESTIONE DEL FORM : AGGIUNGI
app.post('/aggiungi_immersione', accessoSicuro, (req, res) => {
    let errori = [];
    if (!req.body.sito) {
        errori.push({ text: 'Devi aggiungere un sito di immersione' });
    }
    if (!req.body.luogo) {
        errori.push({ text: 'Devi aggiungere il luogo' });
    }
    if (!req.body.diving) {
        errori.push({ text: 'Devi aggiungere il diving' });
    }
    if (!req.body.indirizzo) {
        errori.push({ text: 'Devi aggiungere l indirizzo del diving' });
    }
    if (!req.body.pmax) {
        errori.push({ text: 'Devi aggiungere la profondità massima raggiunta' });
    }
    if (!req.body.tmax) {
        errori.push({ text: 'Devi aggiungere il tempo totale di durata dell immersione' });
    }
    if (!req.body.mixfondo) {
        errori.push({ text: 'Devi aggiungere la miscela di immersione' });
    }
    if (!req.body.testo) {
        errori.push({ text: 'Devi aggiungere un contenuto' });
    }

    if (errori.length > 0) {
        res.render('aggiungi_immersione', {
            errori: errori,
            sito: req.body.sito,
            luogo: req.body.luogo,
            diving: req.body.diving,
            indirizzo: req.body.indirizzo,
            pmax: req.body.pmax,
            tmax: req.body.tmax,
            mixfondo: req.body.mixfondo,
            dettagli: req.body.testo
        });
    } else {
        //res.send('ok, funziono!');
        const nuovaImmersione = {
            sito: req.body.sito,
            luogo: req.body.luogo,
            diving: req.body.diving,
            indirizzo: req.body.indirizzo,
            pmax: req.body.pmax,
            tmax: req.body.tmax,
            mixfondo: req.body.mixfondo,
            contenuto: req.body.testo,
            utente: req.user.id
        }
        new Immersioni(nuovaImmersione)
            .save()
            .then(immersione => {
                req.flash('msg_successo', 'Immersione aggiunta correttamente');
                res.redirect('/lista_immersioni');
            })
    }
});

```

FORM REGISTRAZIONE

```
<h4 class="text-center col-sm-12 col-md-8 col-lg-8 m-auto" style="background-color: #0a91f2" >Crea un account</h4>
<div class="col-sm-12 col-md-8 col-lg-8 m-auto">
    <div class="card my-5">
        <form class="p-5 mb-3" action="/registrazione" method="post">
            <div class="form-group">
                <label for="nome">Nome</label>
                <input type="text" class="form-control" name="nome" value="{{nome}}">
            </div>
            <div class="form-group">
                <label for="cognome">Cognome</label>
                <input type="text" class="form-control" name="cognome" value="{{cognome}}">
            </div>
            <div class="form-group">
                <label for="email">Email</label>
                <input type="email" class="form-control" name="email" value="{{email}}">
            </div>
            <div class="form-group">
                <label for="password">Password</label>
                <input type="password" class="form-control" name="password" value="{{password}}">
            </div>
            <div class="form-group">
                <label for="password">Conferma Password</label>
                <input type="password" class="form-control" name="conferma_psw" value="{{conferma_psw}}">
            </div>
            <button type="submit" class="btn" style="background-color: #0a91f2">Invia</button>
        </form>
    </div>
</div>
```

GESTIONE ACCESSI CON LOGIN E LOGOUT

```
//GESTIONE FORM LOGIN
app.post('/login', (req, res, next) => {
    passport.authenticate('local', {
        successRedirect: '/lista_immersioni',
        failureRedirect: '/login',
        failureFlash: true
    })(req, res, next);
})

//GESTIONE LOGOUT
app.get('/logout', (req, res) => {
    req.logout();
    req.flash('msg_successo', 'Sei disconnesso. Ciao, alla prossima sessione');
    res.redirect('/');
});

```

Conclusioni

Conclusioni

Il progetto risulta ben strutturato e ciò permette delle buone fondamenta per implementazioni future, forse troppo per l'attuale funzionalità monolitica che offre.

Tuttavia, larghi tempi di sviluppo iniziali, se portano a un risultato più strutturato ed evitano la presenza di "Spaghetti Code", diventano un beneficio soprattutto qualora ilprodotto voglia essere approfondito e ampliato successivamente.

Nota bibliografica e sitografica

Risorse bibliografiche e sitografiche usate come riferimento e documentazione del lavoro.

(1) Express.js - https://expressjs.com/

(2) MongoDB - https://www.mongodb.com/it-it

(3) Bootstrap - https://getbootstrap.com/

(4) Express-handlebars - https://www.npmjs.com/package/express-handlebars https://handlebarsjs.com/

(5) Express-session - https://www.npmjs.com/package/express-session https://expressjs.com/en/resources/middleware/session.html

(6) Mongoose - https://mongoosejs.com/ https://www.npmjs.com/package/mongoose

(7) Connect-flash - https://www.npmjs.com/package/connect-flash

(8) Method-override - https://www.npmjs.com/package/method-override https://expressjs.com/en/resources/middleware/method-override.html

(9) Body-parser - https://www.npmjs.com/package/body-parser https://expressjs.com/en/resources/middleware/body-parser.html

(10) Passport – https://www.passportjs.org/ https://www.npmjs.com/package/passport https://heynode.com/tutorial/authenticate-users-node-expressjs-and-passportjs/

(11) Passport-local - https://www.passportjs.org/packages/passport-local/ https://github.com/jaredhanson/passport-local https://www.npmjs.com/package/passport-local

(12) Bcryptjs - https://www.npmjs.com/package/bcryptjs https://github.com/kelektiv/node.bcrypt.js/