**Relazione tecnica Web Application CarSEA (BlaBlaCar)**

**Studente:** Gherardi Gianmattia  
**Matricola:** STA07049/L8  
**Progetto:** BlaBlaCar  
**Repository:** https://github.com/ggherardi/CarSEA

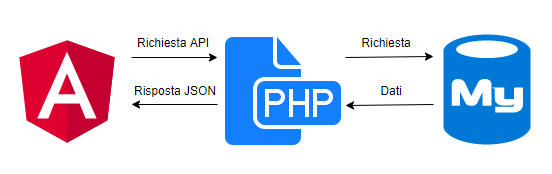
1. **Tecnologie utilizzate**

Per la realizzazione del progetto sono state utilizzate le seguenti tecnologie:

* Angular 5.x
* PHP 7.0
* DB MySQL
* CSS3

Essendo realizzato in Angular, il progetto avrà una struttura che segue il paradigma MVC (Model, View, Controller) e il codice sorgente sarà scritto in HMTL e CSS3 per quanto riguarda le View, e in TypeScript per quanto concerne invece Controller e Model. Per propria natura, il codice TypeScript viene compilato tramite un transpiler in JavaScript (compatibile ECMAScript 5), e dunque l’applicazione sarà eseguibile su qualunque piattaforma che supporti lo standard ECMAScript 5.   
Le chiamate REST si interfacceranno con dei servizi scritti in PHP i quali, a loro volta, comunicheranno con un database MySQL.

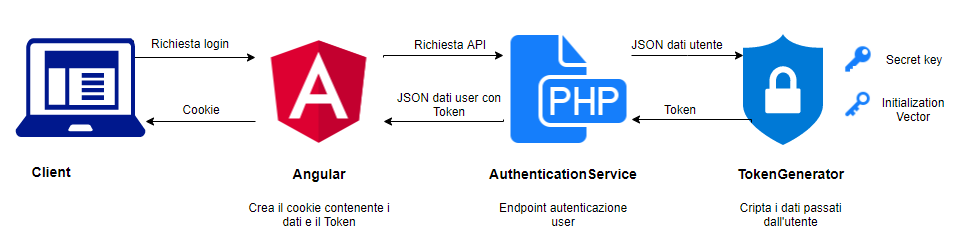
1. **Comunicazione con il database**

Alcune operazioni svolte dagli utenti implicheranno la persistenza di dati sul database MySQL. Questi dati verranno comunicati tramite delle chiamate REST a dei servizi PHP, che a loro volta andranno ad interrogare il database per restituire infine al client una response in formato JSON. 

Da notare che non tutte le richieste API potranno essere effettuate se l’utente non ha effettuato l’accesso al sito: alcuni endpoint necessitano, infatti, dei Token di sicurezza staccati durante l’autenticazione.

1. **Autenticazione e sicurezza**

Quando l’utente crea una nuova utenza, viene generata una tupla nel DB contenente i dati forniti durante la registrazione; le password vengono criptate in modo da non essere lasciate in chiaro e per renderle più sicure.  
Ogniqualvolta l’utente immetterà le sue credenziali di accesso, se l’autenticazione andrà a buon fine verrà creato un cookie per mantenere la sessione di quell’utente attiva: questo cookie ha durata di 5 ore e conterrà Username, ID, Nome e un Token criptato. La creazione del Token avviene utilizzando due diverse chiavi di cifratura e la stringa JSON contenente i dati sopracitati dell’utente. In questa maniera viene generato un Token che sarà indissolubilmente legato a queste informazioni, così da impedire tentativi di manomissione dei cookie e quindi delle chiamate ai servizi: infatti, alcuni endpoint (in particolare quelli che andranno a scrivere sul DB) necessitano della validazione del Token che verrà passato durante la chiamata; se la decriptazione non andasse a buon fine, la chiamata al servizio termina e viene restituito al client un messaggio di errore di autenticazione. In tal modo si può impedire che una persona esterna possa causare danni ad altri utenti.



1. **Navigazione del sito**

Essendo una SPA (Single Page Application), esisterà una singola pagina (index.html). All’interno di questa pagina è presente un tag particolare, cioè <app-root>. Questo tag corrisponde al componente di “root”, una sorta di Master Page che conterrà tutte le View del sito. La navigazione tra le pagine è dunque fittizia: quando si inserisce un certo URL nella barra degli indirizzi, Angular verificherà all’interno della sitemap (o mappa del sito) se quell’URL è presente o meno, e nel caso lo sia procederà a presentare il componente associato all’indirizzo. Se invece l’URL non fosse presente, è stato predisposta un componente di reindirizzamento con la dicitura “Pagina non trovata”, che procederà a reindirizzare l’utente alla homepage.

1. **Sistema di messaggistica**

La modalità di comunicazione tra utenti è gestita in maniera similare ad una “chat”. Nel componente relativo è presente una lista di conversazioni avviate con altri utenti: cliccando su una di queste conversazioni, viene effettuata una chiamata API che recupera tutti i messaggi relativi a quella conversazione: allo stesso tempo, viene scatenato un evento che in maniera ciclica interrogherà il DB alla ricerca di nuovi messaggi da parte della persona con la quale si sta comunicando. Questo evento termina quando si cambia conversazione o quando si esce dal componente di messaggistica, per evitare di sovraccaricare il server. Da notare che tutti i messaggi non sono criptati, di conseguenza è bene ricordare agli utenti di non fornire informazioni riservate quando si sta effettuando una conversazione con altri utenti.

1. **Sistema di logging**

E’ stato creato un sistema di logging che permette di tracciare le operazioni eseguite dai servizi PHP. Verrà creato ogni giorno (eventualmente da suddividere ulteriormente) un file di log che scriverà operazioni eseguite ed eventuali eccezioni. Ogni singola chiamata al servizio verrà dotata di un ID di correlazione (CorrelationID) che permetterà di seguire tutte le operazioni di quella richiesta sul log, fino al termine della stessa.

1. **Analisi codice sorgente e script PHP**

Di seguito si troverà la descrizione del funzionamento dei componenti Angular (Controller, Model e View) e degli script PHP:

*Componenti Angular: Model*

* api.service.ts: contiene la maggior parte delle chiamate API sotto forma di metodi proxy ad astrazione maggiore, utilizzabili dai Controller ed eventualmente da altri Model; gli endpoint e le azioni sono prestabilite, le uniche variabili sono gli argomenti passati durante l’utilizzo dei metodi stessi. Ogni chiamata restituisce un oggetto di tipo Observable che può essere utilizzato tramite effettuando il subscribe.
* constants.service.ts: contiene costanti che possono essere riutilizzate in tutto il progetto.
* cookies.ts: contiene tutti i metodi che permettono di maneggiare il o i cookie necessari al funzionamento del sito. Come default, il cookie della sessione viene codificato in base 64 prima di essere salvato sulla macchina dell’utente; inoltre, ogni chiamata API effettua il refresh del cookie della sessione.
* httpService.ts: contiene i metodi ad astrazione più bassa per effettuare le chiamate API. Poiché non viene utilizzata la funzione ajax di jQuery per effettuare le chiamate XmlHttpRequest, è stato necessario replicare la conversione automatica in formato querystring dei dati, altrimenti il PHP non sarebbe stato in grado di interpretare le informazioni ricevute.
* models.ts: contiene la struttura (classi) di tutti gli oggetti utilizzati nel progetto.
* shared.ts: contiene alcuni tra i metodi più comuni e condivisi in tutto il progetto, tra cui i metodi per effettuare le chiamate API ad astrazione intermedia, necessari per l’utilizzo corretto dei Token.
* storage.service.ts: contiene gli oggetti riutilizzabili tra i vari componenti, evitando l’utilizzo della querystring: quando viene istanziato un nuovo oggetto, viene istanziato come Singleton, in modo tale che sia univoco.
* utilities.service.ts: contiene alcuni metodi ausiliari per semplificare il funzionamento di alcuni componenti, come ad esempio la classe List che permette di eseguire operazioni più complesse rispetto agli array di default di JavaScript.

*Componenti Angular: Controller e View*

Le View in Angular hanno sempre l’estensione .html, mentre i Controller hanno come estensione “ts” (TypeScript). E’ inoltre possibile trovare i file “css” che contengono il css utilizzato in quella singola view. Poiché questi file sono legati strettamente fra di loro, nell’elenco di seguito verrà indicato per semplicità il solo nome del componente, ad eccezione di app.module.ts, poiché esso rappresenta non un componente ma bensì un modulo.

* app.module.ts: è il modulo in cui sono racchiusi tutti i Controller, le View e i Model. All’interno di questo file vengono effettuate tutti gli import di librerie esterne e le dichiarazioni di componenti interni al progetto. E’ inoltre possibile trovare i provider, che servono per poter utilizzare la Dependency Injection, cioè permettono ai vari costruttori dei componenti di riconoscere quale oggetto devono istanziare quando richiamati. Possiamo infine trovare la “sitemap” o “mappa del sito”, cioè una sorta di dizionario chiave valore che permette al Router di Angular di richiamare un certo componente anziché un altro quando si naviga verso una URL.
* app.component: è il componente principale della web application, che viene inizializzato sempre. Nel suo costruttore vengono inizializzate, tramite Dependency Injection, tutte le classi con i metodi che vengono utilizzati e condivisi dai i componenti figli: questi ultimi vengono “chiamati” navigando il sito tramite il Router, che li inserisce all’interno del “router-outlet”. Se dunque da un sub componente si volessero richiamare i Singleton, sarà necessario utilizzare le classi istanziate da app.component.
* body.component: rappresenta la home page, e contiene un primo form di ricerca dei Passaggi.
* footer.component: rappresenta il footer del sito, e contiene link ad eventuali pagine Facebook, Twitter ed altri social.
* header.components: contiene il menù superiore del sito, dove sono presenti il pulsante per mostrare il meù laterale, quello per tornare alla home page e il form di login e iscrizione.
* pagenotfound.component: rappresenta la view che viene mostrata quando l’utente inserisce un URL che non trova corrispondenza nella sitemap nell’app.module. Dopo due secondi, reindirizza alla homepage.
* sidebar.component: contiene la barra laterale del sito: attraverso di essa si possono raggiungere i componenti che rappresentano le funzionalità principali dell’applicazione. Se l’utente è loggato, sono presenti anche le voci che servono a raggiungere le funzioni legate alla gestione della propria utenza.
* adminpanel.component: contiene le statistiche del sito, tra cui quali sono i percorsi più comuni e i prezzi medi relativi.
* dashboard.component: componente di riepilogo dei dettagli principali dell’utente, serve ad altri user per visualizzare le informazioni sull’eventuale Conducente o Passeggero.
* messages.component: rappresenta il metodo di comunicazioni tra utenti, contiene una lista di conversazioni che riportano, ognuna, ad una lista di messaggi corrispondenti. Se si vuole mandare messaggi ad un Passeggero o ad un Conducente con il quale non si ha mai parlato, il componente predispone una nuova conversazione e un nuovo messaggio, che vengono salvati sul DB unicamente quando si preme “invio” o il pulsante di invio del messaggio. Il sistema di verifica di nuovi messaggi è attivo unicamente per la conversazione che è stata selezionata.
* mytrips.component: contiene il listato