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.

2) Comunicazione con il database

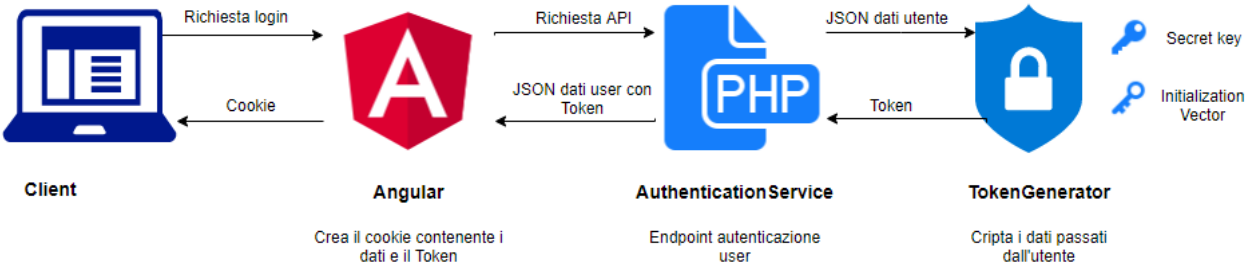
Alcune operazioni svolte dagli utenti implicheranno la persistenza di dati sul database MySQL. Questi dati verranno comunicati tramite delle chiamate REST a dei servizi PHP, che a loro volta andranno ad interrogare il database per restituire infine al client una response in formato JSON.

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.

3) 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.



4) 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.

5) Sistema di messaggistica

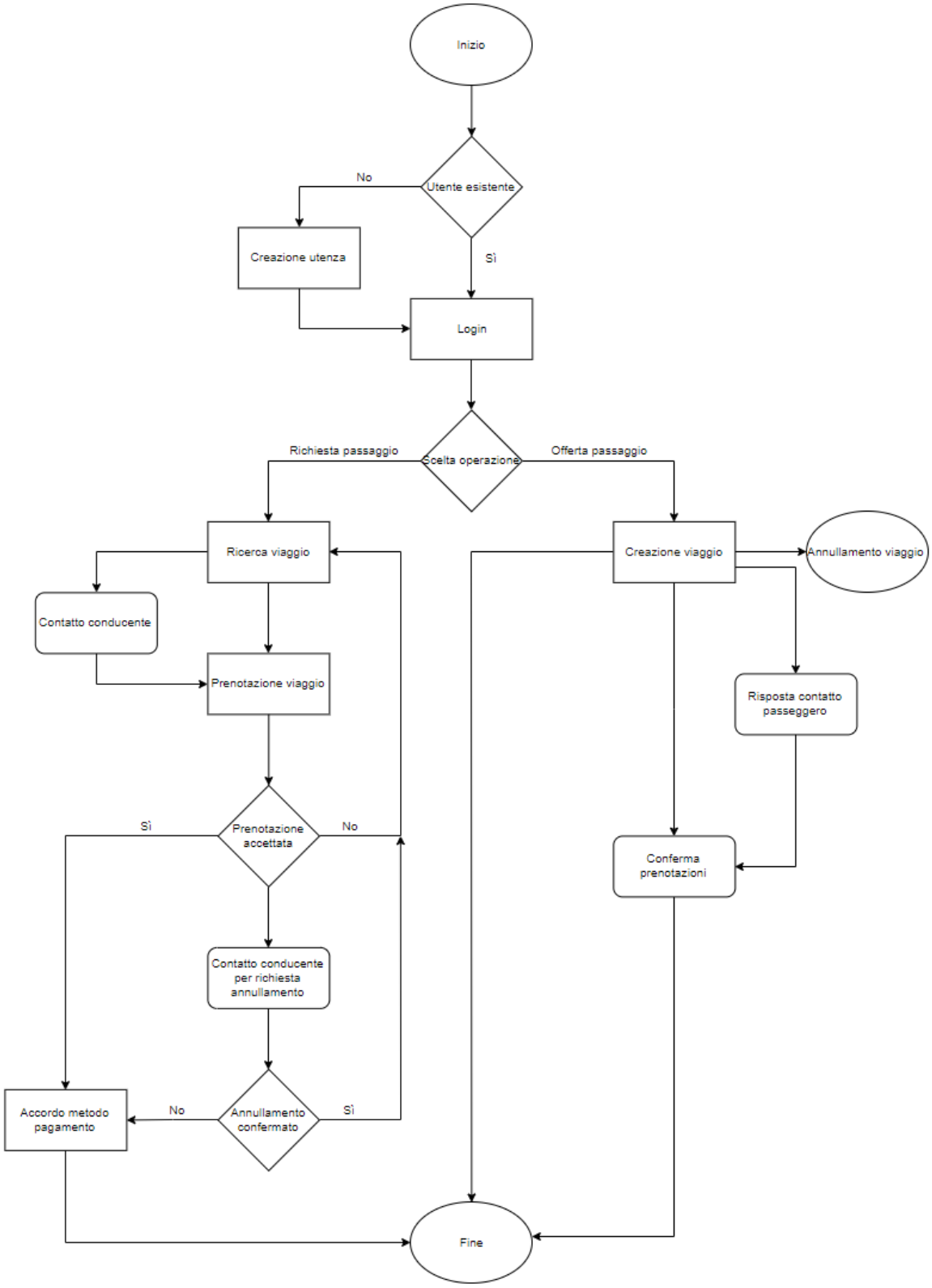
La modalità di comunicazione tra utenti è gestita in maniera similare ad una "chat". Nel componente relativo è presente una lista di conversazioni avviate con altri utenti: cliccando su una di queste conversazioni, viene effettuata una chiamata API che recupera tutti i messaggi relativi a quella conversazione: 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.

6) 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.

7) Flusso applicativo



8) 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 la lista delle prenotazioni di Passaggi effettuate. La lista si divide in richieste in corso e nell'archivio delle richieste passate. Da qui è possibile annullare iniziare una nuova conversazione con il Conducente, annullare la prenotazione e visionare i dettagli dei Passaggi tramite i pulsanti che richiamano il menù di callout.
- offeredtrips.component: contiene la lista dei Passaggi offerti. La lista si divide in Passaggi in corso e passati. Da qui è possibile gestire le prenotazioni, vedere i dettagli dei viaggi ed eventualmente annullarli tramite i pulsanti che richiamano i menù di callout.
- personaldetails.component: rappresenta la pagina dove è possibile modificare i propri dettagli personali tra cui il proprio automezzo: la ricerca di quest'ultimo avviene tramite il componente apiautocomplete.component, che utilizza delle chiamate API per ottenere la lista delle marche e dei modelli di auto.
- signup.component: è la pagina dove è possibile effettuare l'iscrizione al sito. Una volta creato il profilo, viene effettuato automaticamente il login e si viene reindirizzati al componente personaldetails.component.
- apiautocomplete.component: è un componente di supporto che viene riutilizzato da più componenti. Si tratta di una textbox dove, fornita una lista sorgente, viene effettuato un autocomplete sui valori della lista stessa. Una volta selezionato un elemento, è possibile utilizzarlo per compilare vari tipi di form.
- tripabstract.component: componente di supporto che rappresenta un elemento della lista dei Passaggi offerti quando vengono cercati tramite i form di ricerca. La View contiene il layout dell'elemento, mentre il Controller si occupa di reindirizzare l'utente una volta che viene effettuato il click sull'elemento stesso.

- findpassage.component: è il componente dove è possibile cercare i Passaggi offerti dai Conducenti e raffinare i risultati della ricerca effettuata. Utilizza il componente di supporto tripabstract.component per renderizzare i viaggi che soddisfano i filtri impostati dalla ricerca.
- pathchooser.component: è il componente dove viene effettuato il primo step di creazione dei Passaggi da parte dei Conducenti: tramite la customizzazione e l'utilizzo delle API di Google Maps è possibile vedere il percorso ricalcolato a seconda delle città che vengono selezionate da parte dell'utente. E' necessario scegliere una città di partenza e una di fine, mentre le tappe intermedie sono facoltative; la scelta viene effettuata tramite una versione modificata del componente apiautocomplete.component. Una volta completato l'inserimento delle città e della data e ora di partenza, è possibile proseguire allo step successivo.
- pricechooser.component: secondo step della creazione dei Passaggi da parte del Conducente. Qui è necessario scegliere il prezzo, il numero dei posti offerti ed eventualmente inserire una descrizione del Passaggio. Una volta compilati i campi obbligatori, è possibile cliccare sul pulsante di creazione del Passaggio.
- tripdetail.component: questo componente contiene i dettagli del Passaggio: qui è possibile decidere di mandare un messaggio al Conducente e/o effettuare la prenotazione al Passaggio stesso. Serve inoltre come pagina di riepilogo quando si vogliono visualizzare i dettagli dei Passaggi offerti o prenotati.

Script Servizi PHP

Nella cartella Models sono presenti le classi che vengono utilizzate dagli altri fogli PHP. Di seguito verranno elencati tutti gli script PHP che rappresentano i servizi utilizzati per le chiamate REST. Per ogni servizio verranno elencati e descritti gli endpoint.

- AuthenticationService.php (Servizio)

  Contiene gli endpoint che permettono di gestire l'autenticazione.

  a) SignUp: gestisce la creazione delle utenze. Viene effettuato un controllo per verificare che l'username o l'email non siano già stati inseriti nel DB.
  b) Login: consente agli utenti di effettuare il login al sito. Se l'autenticazione va a buon fine viene generato il Token di sicurezza che servirà poi ad Angular per creare il cookie di sessione.
- CarService.php (Servizio)

  Contiene gli endpoint che consentono di ricercare marca e modelli delle automobili inseriti nei componenti di autocomplete.

  a) SearchCarsMake: effettua la ricerca nel Database della marca automobilistica passata come parametro (anche parziale).
  b) SearchCarsModel: effettua la ricerca nel Database del modello dell'automobile passata come parametro (anche parziale)

- CitiesService.php (Servizio)

  Contiene gli endpoint per la gestione delle città che possono essere cercate dagli utenti per creare i viaggi.

a) InsertCities: endpoint che permette l'inserimento di una nuova città nel DB. E' stato utilizzato per effettuare la migrazione massiva da un CSV contenente una lista di città esterno alla tabella nel Database.
b) SearchCities: effettua la ricerca nel Database delle città passate come parametro (anche parziale)

- MessageService.php (Servizio)

  Contiene gli endpoint per la gestione del sistema di messaggistica interno al sito.

  a) InsertNewMessage: inserisce nel Database il messaggio passato come parametro.
  b) InsertNewConversation: inserisce nel Database una nuova conversazione.
  c) GetExistingConversation: interroga il Database per recuperare un'eventuale conversazione già esistente tra due utenti.
  d) GetConversations: recupera dal Database tutte le conversazioni esistenti per l'utente passato come parametro.
  e) GetMessages: recupera tutti i messaggi per la conversazione passata come parametro.

- PeopleDetailService.php (Servizio)

  Contiene gli endpoint per la gestione dei dettagli personali degli utenti.

  a) InsertDetails: inserisce nel Database i dettagli personali compilati dall'utente.
  b) RetrieveDetails: recupera dal Database i dettagli personali dell'utente passato come parametro.

- TripService.php (Servizio)

  Contiene gli endpoint per la gestione dei Passaggi e delle prenotazioni.

  a) SaveNewTrip: inserisce nel Database il Passaggio passato come parametro.
  b) GetTrips: recupera dal Database i Passaggi utilizzando diversi filtri: se è specificato l'ID del Passaggio, cercherà quest'ultimo; se è specificato l'ID del Conducente, cercherà tutti i Passaggi offerti da questi; se non sono specificati questi ID la query verrà effettuata utilizzando i filtri di ricerca.
  c) InsertBooking: inserisce nel Database una nuova prenotazione.
  d) GetBookingForUser: recupera dal Database tutte le prenotazioni per l'utente.
  e) GetBookingForTrip: recupera dal Database tutte le prenotazioni per il Passaggio.
  f) GetExistingBooking: recupera dal Database un'eventuale prenotazione già esistente per lo user e il Passaggio specificati.
  g) SetBookingStatus: modifica lo stato della prenotazione come specificato dal parametro.

- DBConnection.php

  E' la classe che gestisce la connessione con il Database e l'esecuzione delle varie query e delle transaction utilizzate da tutti i servizi. Qui è possibile inserire le credenziali di accesso al DB (attualmente in chiaro a scopo di sviluppo)

- Logger.php

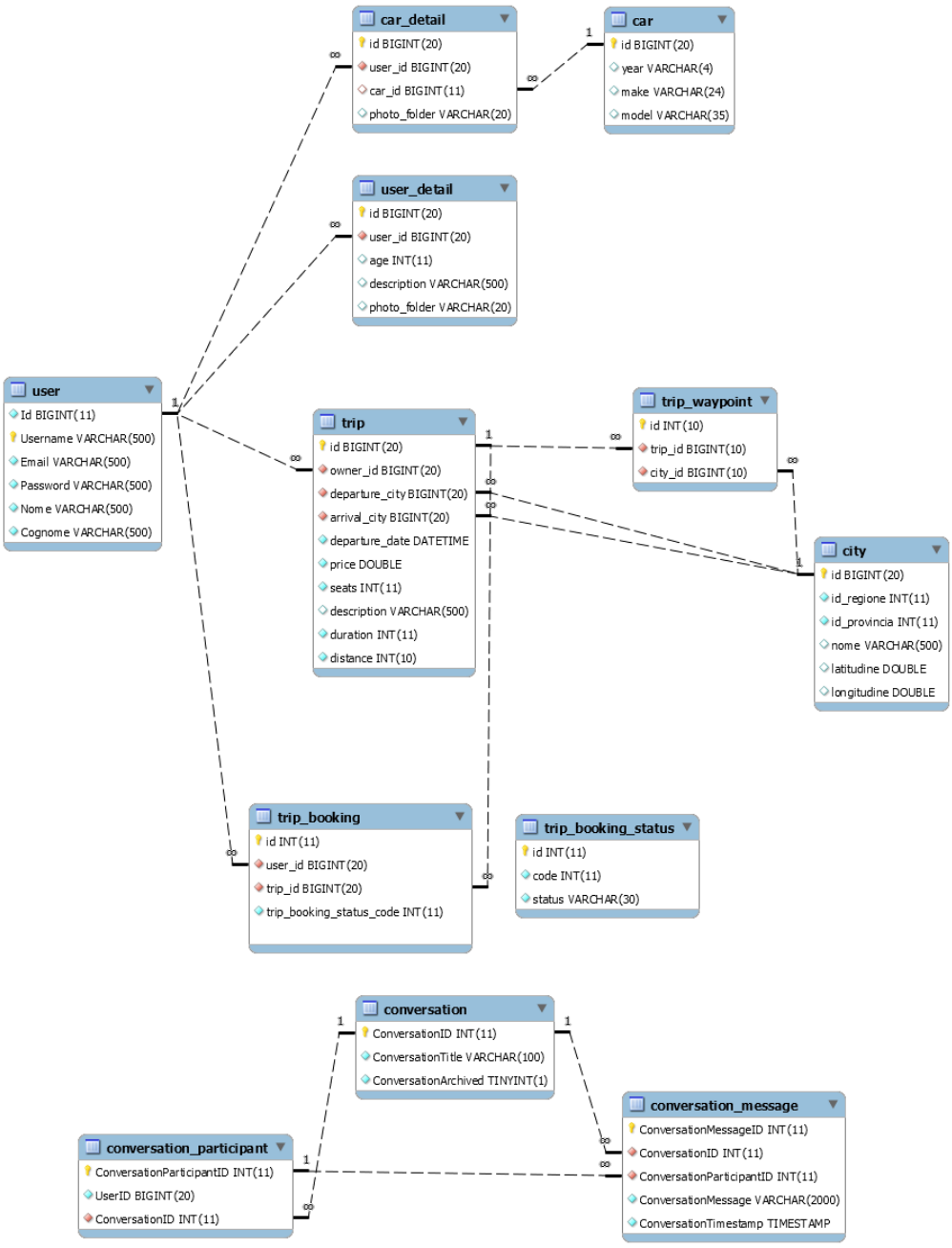
Classe che gestisce la crezione dei log e la scrittura sugli stessi.

- TokenGenerator.php

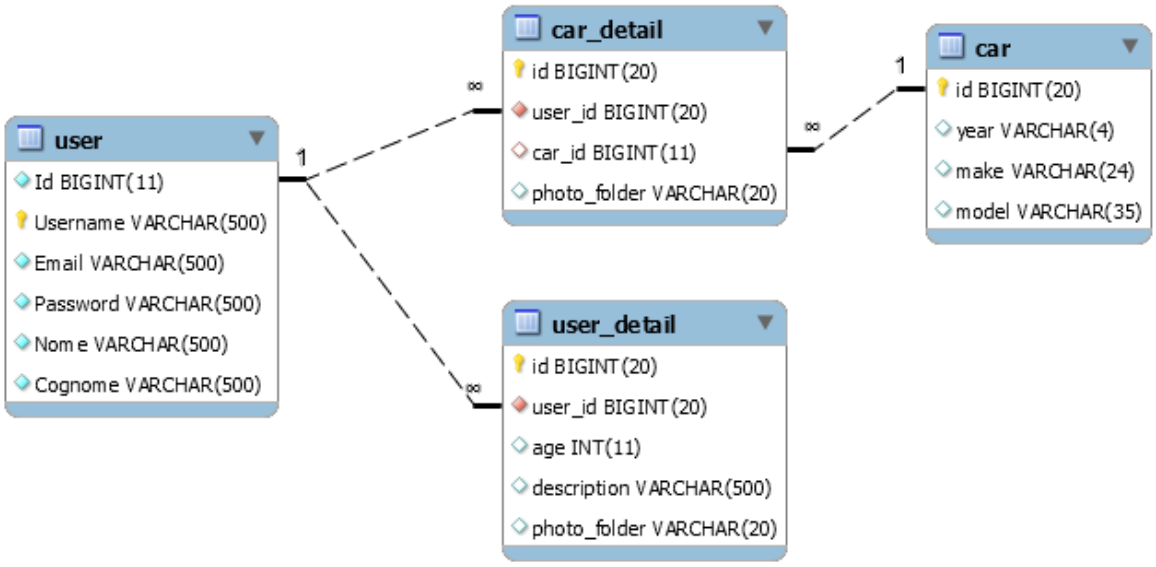
  Classe che gestisce la creazione dei Token di sicurezza e la validazione degli stessi quando vengono effettuate determinate chiamate API da parte di Angular.

9) Schema Database

- *Diagramma completo:*

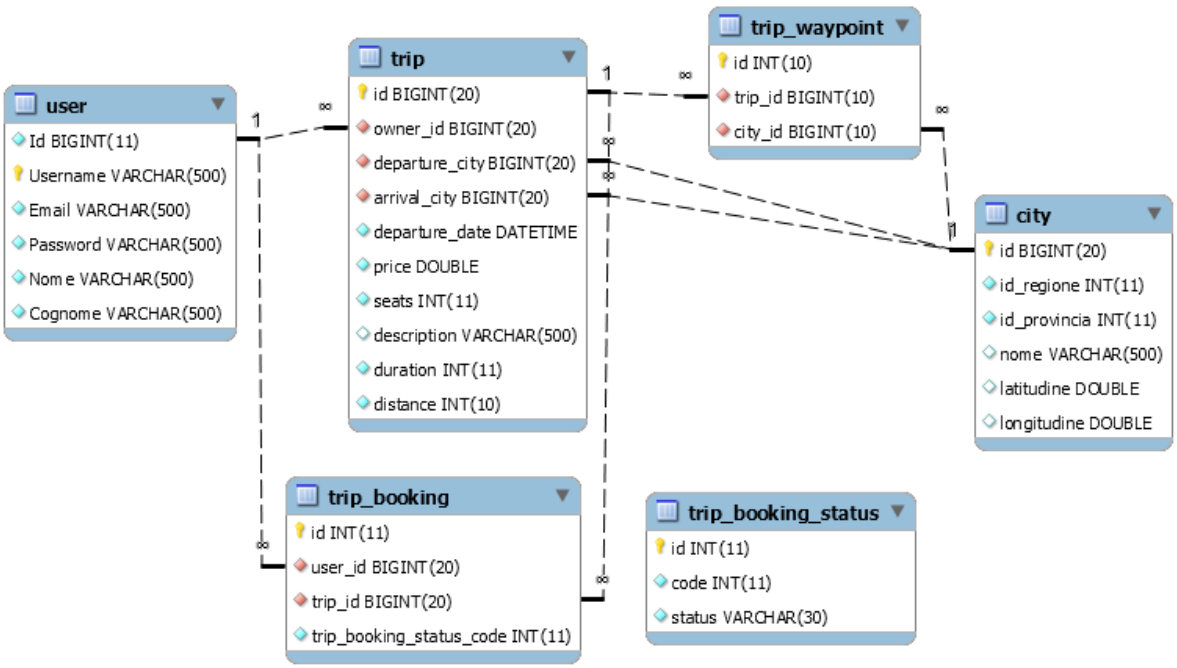


- *Diagramma funzionalità dettagli utente*



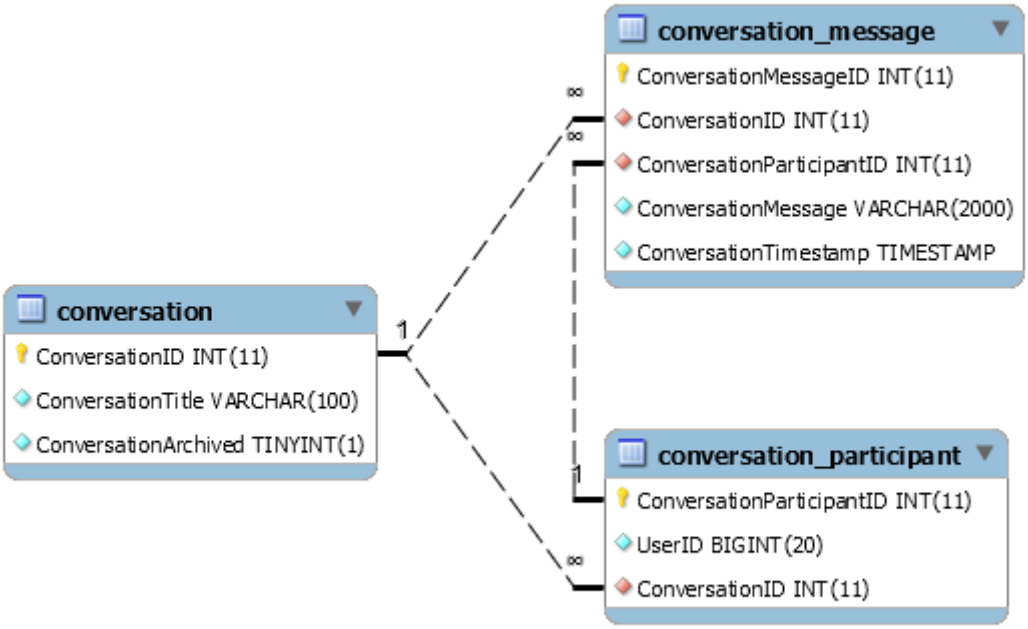
La tabella “car” è stata popolata tramite un’importazione di un CSV contenente vari modelli di automobili.

- *Diagramma funzionalità Passaggi*



La tabella “city” è stata popolata tramite l’importazione di un DB esterno contenente tutte le città italiane, mentre la tabella “trip_booking_status” contiene gli stati delle prenotazioni, ed è prepopolata dagli stessi.

- *Diagramma funzionalità messaggistica*



10) Configurazioni server

Per l'ambiente di sviluppo è stato utilizzato IIS, mentre per quello di produzione, essendo hostato da altervista.org, è stato utilizzato Apache.
Per il corretto funzionamento del sistema di Routing di Angular, è stato necessario applicare delle configurazioni a livello di riscrittura dell'URL. Difatti, sia Apache che IIS, interpretano le richieste di navigazione con gli URL nella stessa maniera, cioè cercando un file fisico che conterrà l'ubicazione cercata (ad es. "http://www.carsea.com/profile/mytrips.html" cercherà all'interno della cartella profile il file mytrips.html): tuttavia la Web Application è composta in modo tale da essere una SPA, di conseguenza non esisteranno altri file fisici oltre ad index.html. Per questo motivo, è stato necessario attivare la funzionalità di URL Rewrite per entrambi i server e creare un web.config per le regole di riscrittura degli URL per quanto riguarda IIS, ed agire a livello di htaccess per quanto riguarda Apache.

11) Librerie esterne

Per lo sviluppo del progetto sono state utilizzate librerie esterne, tutte sotto forma di componenti creati appositamente per le versioni di Angular superiori alla 5. Le librerie si possono anche trovare nel file README all'interno della solution, con i corrispondenti url di riferimento e i comandi da utilizzare per l'installazione delle librerie stesse tramite Node Package Manager (NPM).

- *Bootstrap*
  Utilizzato per rendere responsive il sito.
- *NG2-UI/Autocomplete*

Utilizzato per la gestione della ricerca con autocompletamento.

- *Google Maps API*
  Utilizzato per la gestione delle coordinate e per il tracciamento delle mappe dal componente di scelta dei viaggi.
- *Cuppalabs/CuppaSlider*
  Utilizzato per il menu a scomparsa laterale.
- *NG2-NoUiSlider*
  Utilizzato per il controllo di scelta dell'orario (tipo range).
- *NG2-Paginate*
  Utilizzato per la paginazione di alcune tabelle.
- *NG2-Component-Spinner*
  Utilizzato per mostrare un loader durante il recupero dei dati dal Database o durante il render di alcuni componenti.