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

**Studente:** Gherardi Gianmattia  
**Matricola:** STA07049/L8  
**Progetto:** Videonoleggio (Rent Net)  
**Repository:** https://github.com/ggherardi/videonoleggio

1. **Tecnologie utilizzate**

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

* PHP 7
* DB MySQL (InnoDB)
* JavaScript ES 6
* jQuery 3.2.1
* BootStrap 4
* CSS3

Il gestionale è stato realizzato secondo il paradigma SPA (Single Page Application): non appoggiandosi ad alcun framework JavaScript esterno, è stato realizzato un meccanismo di gestione delle view ad hoc, utilizzando JavaScript e come, uniche librerie esterne, jQuery e BootStrap 4. L’accesso al DB è garantito tramite dei servizi realizzati in PHP 7, sotto forma di classi, che comunicano con altrettante classi JavaScript che fanno da proxy ed effettano le chiamate REST.  
Il server utilizzato è un Raspberry 3 modello B+, sul quale risiedono il DB e il file system dal quale attinge il web server.

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**

All’accesso degli utenti, se l’autenticazione andrà a buon fine verrà creato un cookie http-only per mantenere la sessione di quell’utente attiva: questo cookie ha durata di 12 ore e conterrà i dati del contesto attuale più un token, criptati. La creazione del Token avviene utilizzando due diverse chiavi di cifratura e la stringa JSON contenente i dati. 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 assieme al cookie criptato 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. Inoltre, per garantire una maggiore sicurezza, determinati endpoint verificheranno che la Delega dell’utente che ha effettuato la chiamata al sia sufficiente per poter consumare il servizio stesso. In caso contrario, verrà restituito un altro messaggio di accesso negato e verrà fatto un redirect su una pagina di errore.

1. **Navigazione del sito**

Essendo una SPA (Single Page Application), esisterà una singola pagina (index.html). Tramite il menù di navigazione un Controller si occuperà di richiamare la view corrispondente alla voce selezionata, e caricare HTML e JavaScript relativi, sostituendo l’HTML del container principale.

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 le operazioni eseguite ed eventuali eccezioni. Ogni singola chiamata al servizio verrà dotata di un ID di correlazione (CorrelationID) che permetterà di tracciare in sequenza tutte le operazioni effettuate dal servizio stesso sul log, fino al termine dell’esecuzione.

1. **Analisi funzionale**

Al fine di realizzare un applicativo conforme alla esigenze del cliente, è stato necessario rivedere con il cliente stesso alcune delle richieste avanzate e dei dettagli relativi al funzionamento aziendale, così da poter risolvere ambiguità che avrebbero impedito una corretta redazione dello schema concettuale.

* “*La società ha 6 fornitori che dopo 90 giorni di affitto ritirano la merce*”.

**Domanda**: è necessario gestire e predisporre un sistema di alert per la scadenza dei 90 giorni assieme a un sistema di reportistica tabellare? Bisogna inoltre gestire in qualche modo la mancata riconsegna dopo i 90 giorni?

**Risposta**: non è necessario predisporre un sistema di avvisi, una tabella in cui sono presenti i giorni mancanti alla riconsegna è sufficiente. Non serve la gestione della mancata riconsegna.

* “*Ogni settimana avviene uno scarico dei nuovi video e la restituzione dei video scaduti*”.

**Domanda**: chi si occuperà dello scarico fisico e virtuale dei video? Come avviene la procedura di scarico/carico?

**Risposta**: principalmente saranno gli addetti ad occuparsi dello scarico/carico dei video, sia fisici che virtuali. Ogni settimana ci sarà un trasportatore che porterà i nuovi video e caricherà quelli vecchi.

**Domanda**: è dunque necessario predisporre un componente di gestione degli ordini che si interfacci con i sistemi dei fornitori?

**Risposta**: no. Utilizziamo un applicativo per gli ordini che è il medesimo per tutti i fornitori. E’ sufficiente avere un report di tutti i video attualmente in magazzino.

* “*I punti vendita noleggiano i video a giorni interi e le tariffe sono decrescenti all’aumentare dei giorni di noleggio*”.

**Domanda**: le tariffe sono sempre le stesse o vanno gestite tramite un pannello di amministrazione?

**Risposta**: attualmente sono fisse e decise dal proprietario, ma in futuro potrebbe essere necessarie gestirle tramite un pannello amministrativo.

* “*Il responsabile del punto vendita può applicare gli sconti ad alcuni clienti fidelizzati, ma la politica degli sconti deve essere codificata*”.

**Domanda**: la funzione di creazione e modifica dei record relativi ai clienti sarà disponibile a tutti i tipi di utenza, eccetto la scelta della fidelizzazione?

**Risposta**: esatto, tutti i dipendenti potranno creare utenze, ma unicamente i responsabili o il proprietario potranno scegliere il tipo di fidelizzazione da applicare ai clienti.

* “*Ogni cliente dovrà firmare una liberatoria per la legge sulla privacy e si dovrà tenere traccia delle coordinate del documento cartaceo corrispondente*”

**Domanda**: può essere utile un meccanismo di upload del file cartaceo, così da averlo a disposizione per la stampa in qualsivoglia momento senza dover necessariamente andare a cercare il cartaceo in archivio?

**Risposta**: sì, può esserci molto utile.

* “*Prenotazione dei titoli in uscita da parte dei clienti*”.

**Domanda**: è necessario gestire le prenotazioni in modo tale che interagiscono con il conteggio dei noleggi effettuabili, o è sufficiente garantire un sistema di report che mostri quanti film sono stati prenotati?

**Risposta**: è sufficiente una reportistica.

* “*I clienti dovranno essere rintracciabili e deve essere tenuta traccia del cliente e dei video che ha in noleggio*”.

**Domanda**: la visualizzazione dei video in noleggio (attivi e passati) dovrà essere condivisa da tutti i punti vendita, o dovranno essere visualizzati i noleggi relativi al singolo punto vendita?

**Risposta**: ogni punto vendita dovrà visualizzare i propri noleggi attivi e passati relativi a quel cliente.

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

*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.

1. 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.
2. 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.

1. SearchCarsMake: effettua la ricerca nel Database della marca automobilistica passata come parametro (anche parziale).
2. 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.

1. 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.
2. SearchCities: effettua la ricerca delle città passate come parametro (anche parziale)

* MessageService.php (Servizio)

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

1. InsertNewMessage: inserisce nel Database il messaggio passato come parametro.
2. InsertNewConversation: inserisce nel Database una nuova conversazione.
3. GetExistingConversation: interroga il Database per recuperare un’eventuale conversazione già esistente tra due utenti.
4. GetConversations: recupera dal Database tutte le conversazioni esistenti per l’utente passato come parametro.
5. 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.

1. InsertDetails: inserisce nel Database i dettagli personali compilati dall’utente.
2. RetrieveDetails: recupera i dettagli personali dell’utente passato come parametro.

* TripService.php (Servizio)

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

1. SaveNewTrip: inserisce nel Database il Passaggio passato come parametro.
2. 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.
3. InsertBooking: inserisce nel Database una nuova prenotazione.
4. GetBookingForUser: recupera dal Database tutte le prenotazioni per l’utente.
5. GetBookingForTrip: recupera dal Database tutte le prenotazioni per il Passaggio.
6. GetExistingBooking: recupera dal Database un’eventuale prenotazione già esistente per lo user e il Passaggio specificati.
7. 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

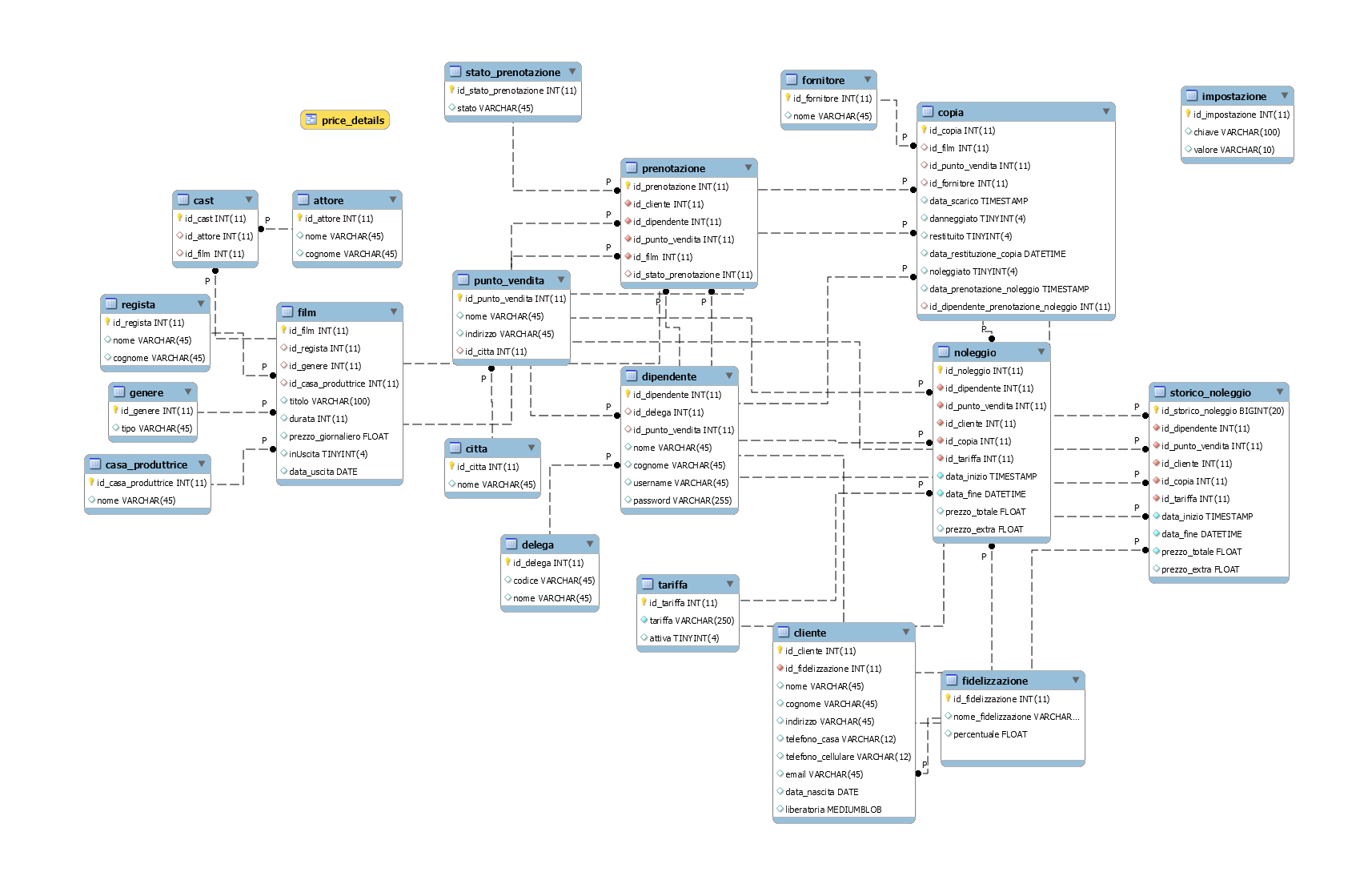
Classe che gestisce la crezione dei log e la scrittura sugli stessi.

* 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.

1. **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*

1. **Configurazioni server**

Per l’ambiente di sviluppo è stato utilizzato IIS, mentre per quello di produzione, essendo hostato su un Raspberry, è stato utilizzato Apache.