**Noleggio Auto – Prà Mattia**

**Descrizione del progetto lato codice:**

**Installazione**

È necessario scrivere i dettagli del database (host, porta, username e password, nome del database) nel file .env (sintassi CHIAVE=VALORE) situato nella root.

Il database è composto da 6 tabelle abbastanza semplici --> importare dal backup *'noleggio\_auto[EXPORT].sql*'.

Per importare il database: `*mysql -u username -p database\_name < noleggio\_auto[EXPORT].sql*`

Nel file .env è presente un parametro `TOKEN\_KEY`: si tratta della chiave di cifratura usata nell'algoritmo di hashing (MD5) durante la generazione dei token delle prenotazioni.

**Spiegazioni delle classi:**

* **Token:** Il token viene generato usando una mia personale implementazione seguendo circa il meccanismo di sicurezza usato dai JsonWebToken (ho solo voluto tenerli molto più semplici e rapidi per non utilizzare librerie esterne al progetto per fare quanto di cui io avessi bisogno).
* **Template:** L'intero progetto si basa su una serie di pagine PHP che tramite una classe Template che legge, dato in input il nome del file HTML contenente il markup della pagina (arricchito da dei placeholders che permetteranno in runtime di iniettare contenuto dinamico nel punto giusto).

Questa classe legge il file HTML, lo gestisce iniettando parti comuni (parte del HEAD, l'header della pagina, il footer, ...) e lascia in modo trasparente i placeholders non relativi a dei componenti HTML (situati nella cartella ~/src/components).

In runtime, dopo che gli script PHP recuperano e generano l'HTML dinamico richiamando la funzione `putDynamicContent` possono iniettare in modo molto semplice nella pagina i markup dinamici, sostituendoli ai placeholders.

* **Database:** La classe Database permette in modo semplice di aprire una connessione al database e fornisce una serie di metodi pubblici per eseguire le operazioni di SELECT, INSERT INTO, UPDATE e DELETE sul database SQL (MySQL/MariaDB).

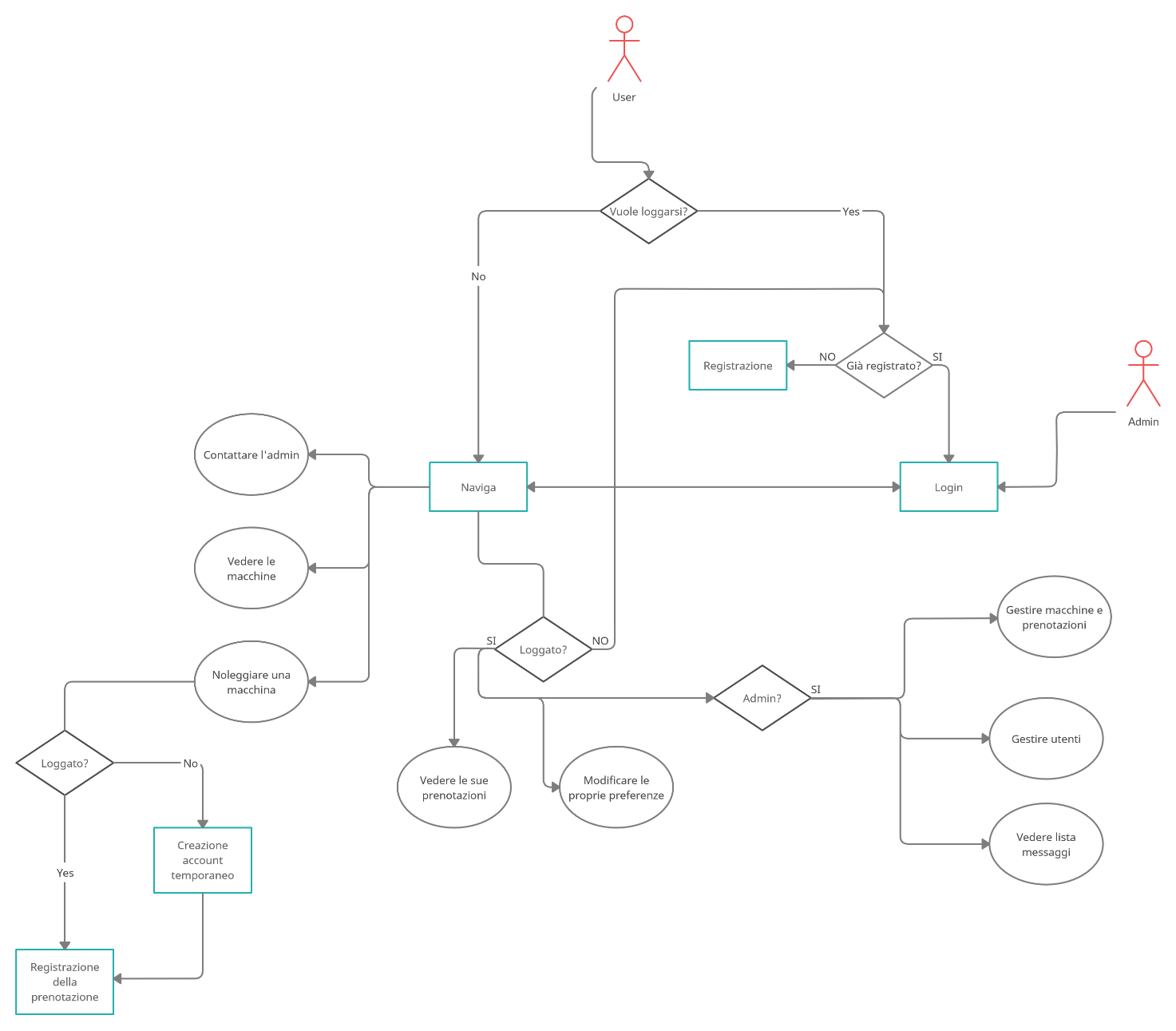
È stato preferito un approccio di questo tipo per poter gestire meglio gli errori.

* **User:** La classe User permette, ad ogni avvio di pagina (alla pari del `session\_start()`) di instanziare un oggetto User che, se l'utente è loggato nella sessione corrente, recupera dalla sessione, altrimenti questo oggetto avrà l'attributo `loggedIn` a false e permetterà in fase di runtime di gestire correttamente il login dell'utente.

Fornisce inoltre metodi di aggiornamento, login e creazione dell'utente. I metodi di signup e signupTemp sono statici (non richiedono l'istanziazione dell'oggetto User) e permettono di creare nel database i due tipi di utenti "normali", ovvero utenti 'user' e utenti 'temp': i primi sono gli utenti classici mentre i secondi solo gli utenti che non si sono voluti registrare al sito ma hanno effettuato comunque una prenotazione.

* **Errors:** La classe Errors mi permette di gestire gli errori in modo molto più semplice, andando a gestire in modo efficace ed efficiente ogni tipo di errore, sia fatale che non.
* **Utils:** La classe Utils è una classe che contiene vari metodi statici con funzioni utili un po' a tutte le pagine.

**Diagramma UML e del flusso delle funzioni:**



**Diagrammi delle classi:**

|  |  |
| --- | --- |
| **Users** | |
| Username | String |
| Password | String |
| Email | string |
| Scope | String |
| Name | String |
| Surname | String |
| Age | Int |
| City | Char (2) |
| Fav\_car | String |

|  |  |
| --- | --- |
| **Cars** | |
| Id | int |
| Brand | String |
| Model | String |
| Image | String |
| Price | Int |
| Engine | Int |
| Seats | Int |
| Transmission | String |
| Short\_description | String |
| Category | String |
| Speed | Int |
| GPS | Boolean |
| Tank | Int |
| Description | String |
| Img1 | String |
| Img2 | String |
| Img3 | String |
| Age | Int |

|  |  |
| --- | --- |
| **Stock** | |
| Car\_id | Int |
| Quantity | int |

|  |  |
| --- | --- |
| **Rents** | |
| Id | Int |
| Car\_id | Int |
| User\_id | String |
| City | Char (2) |
| startDate | Date |
| Duration | Int |

|  |  |
| --- | --- |
| **Cities** | |
| Code | Char (2) |
| Name | String |
| Address | String |

|  |  |
| --- | --- |
| **Messages** | |
| Id | Int |
| User\_id | String |
| Object | String |
| Message | String |
| Answered | Boolean |