Noleggio Auto – Prà Mattia

Requisiti (simulati):

Buongiorno,

Siamo una compagnia emergente di autonoleggi di lusso e stiamo cercando di migliorare e ammodernare il nostro sistema di prenotazioni dei noleggi. Avremmo bisogno di un sistema stabile e robusto che permetta in modo semplice di mostrare e noleggiare per un periodo massimo di 14 giorni una delle nostre macchine in una delle nostre sedi.

Chiunque deve poter vedere le nostre magnificenze, che sia registrato o meno, perché vogliamo permettere a chiunque di noleggiare. Pensiamo dunque che sia la cosa migliore permettere di registrarsi per poi eventualmente fornire dei servizi personalizzati, ma non impedire a clienti non registrati di noleggiare.

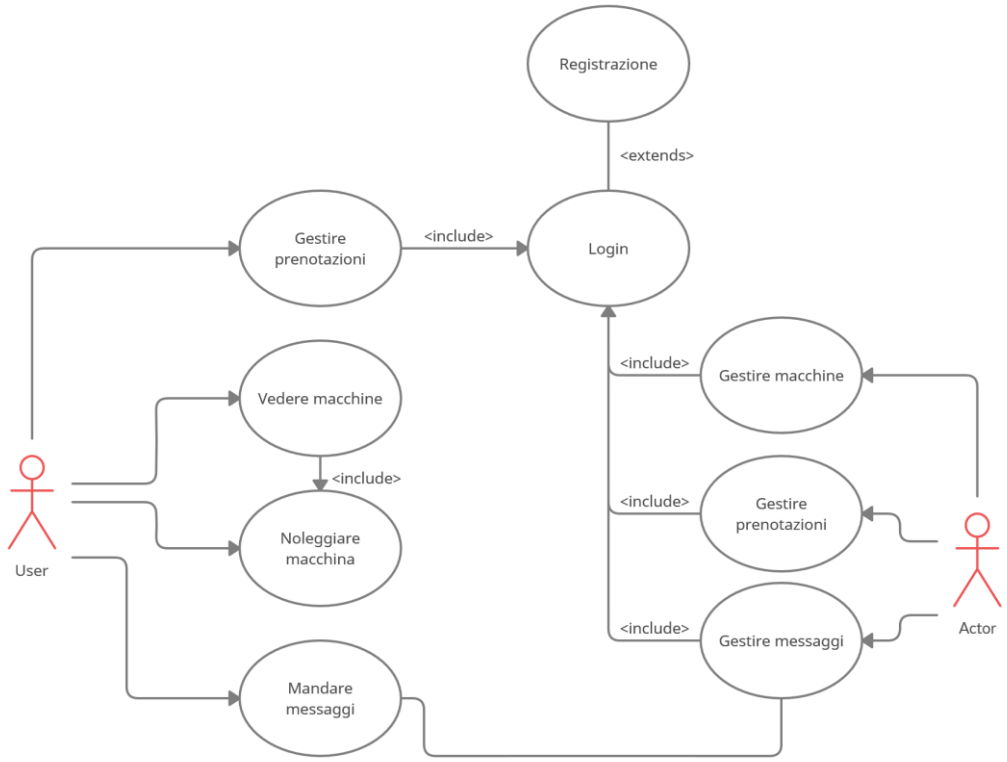
Vorremmo implementare un modo semplice e veloce di comunicazione tra i clienti, per qualunque loro bisogno, e noi di MattiaPrà – Autonoleggio, che ci permetta di rimanere sempre in contatto. In ultima richiesta, se possibile, chiederemmo un modo innovativo per i clienti di avere la prenotazione a portata di mano nel momento del ritiro della vettura.

MattiaPrà – Autonoleggio, Sezione innovazione

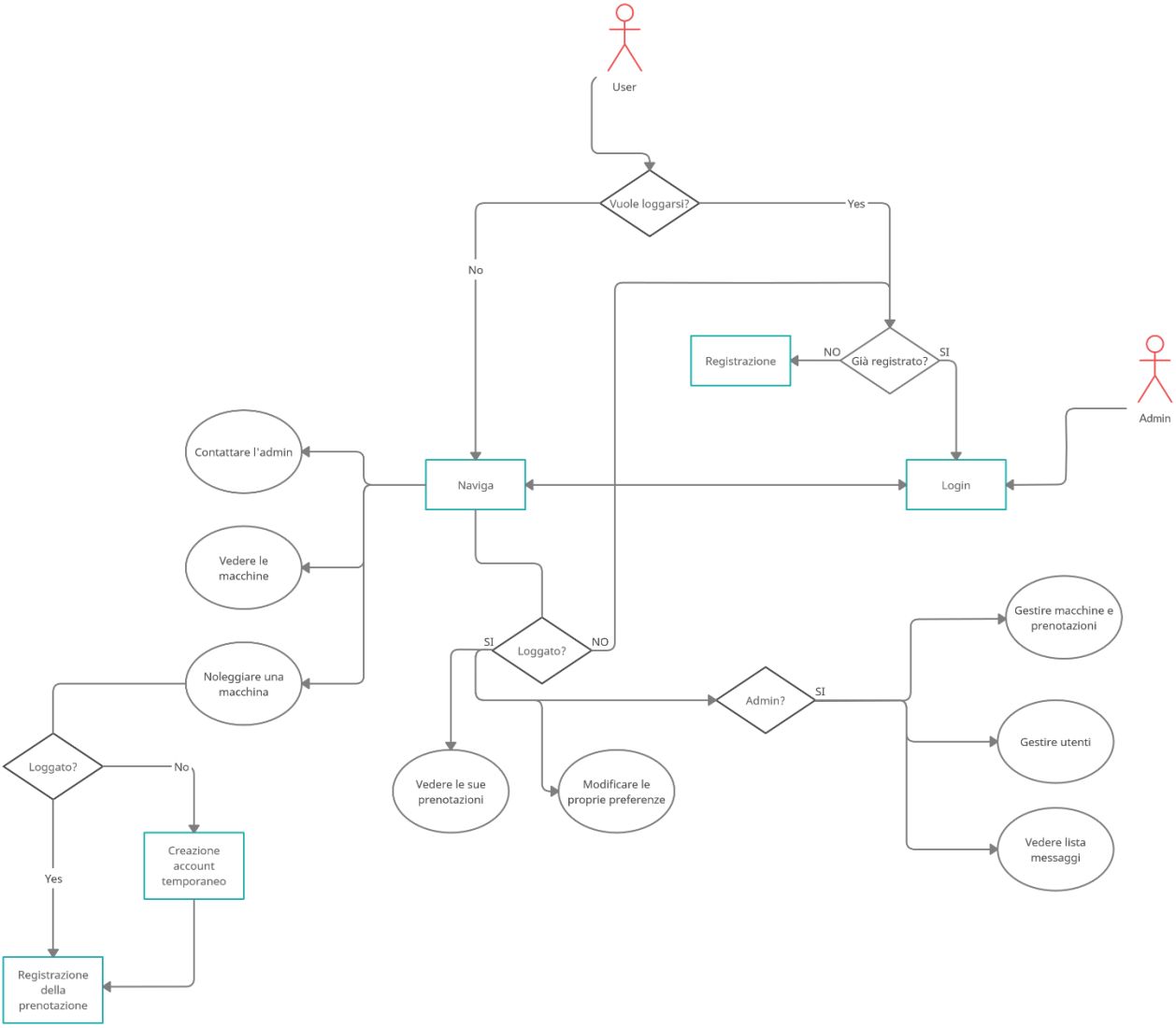
Storyboard:

1. Utente generico:
    a. Entra nel sito
    b. Cerca le macchine che gli interessano
    c. Vede le macchine disponibili
    d. Sceglie una macchina
    e. Prenota una macchina
    f. Usando la sua e-mail e il codice prenotazione in tempi differiti può vedere i dettagli della prenotazione
2. Utente loggato:
    a. Entra nel sito
    b. Fa il login
    c. Cerca le macchine (ordinate in base alle preferenze)
    d. Vede le macchine
    e. Sceglie una macchina
    f. Prenota una macchina
    g. Dalla pagina utente vede le sue prenotazioni
3. Admin:
    a. Entra nel sito
    b. Fa il login
    c. Va nella pagina amministratore
    d. Aggiora una prenotazione / aggiunge una macchina / rimuove un utente / …

Use case diagrams:



Flusso delle funzioni:



Class diagrams:

Users	
Username	String
Password	String
Email	string
Scope	String
Name	String
Surname	String
Age	Int
City	Char (2)
Fav_car	String
○ Registrazione utente (U, P, E, N, S, A, C, F): T/EX ○ Eliminazione (U) ○ Login (U, P) ○ Ottenimento (U)	

Stock	
Car_id	Int
Quantity	int
○ Aggiungere / rimuovere (ID) ○ Modificare (ID)	

È stata implementata una tabella indipendente per permettere in futuro di implementare una terza colonna con le città per avere quantità diverse in base alla sede

Cities	
Code	Char (2)
Name	String
Address	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
○ Aggiungere / rimuovere (ID) ○ Selezionare (ID or B, M, C)	

Rents	
Id	Int
Car_id	Int
User_id	String
City	Char (2)
startDate	Date
Duration	Int
State	String
○ Aggiungere (C, U, C, S, D) ○ Rimuovere (ID)	

Messages	
Id	Int
User_id	String
Object	String
Message	String
Answered	Boolean
○ Aggiungere (U, O, M) ○ Rimuovere (ID) ○ Rispondere / Aggiornare (A)	

Use case diagrams:

Use case	**Login**
Attori	○ User ○ Admin
Flusso "normale"	1. Utente apre la pagina di login 2. Presentata FORM che chiede username e password 3. Utente compila e invia 4. Utente viene reindirizzato alla pagina richiesta
Flussi "eccezionali"	○ Utente inserisce username e/o password sbagliati e/o inesistenti
Dipendenze da altri use-case	○ Registrazione: <extends>

Use case	**Registrazione**
Attori	User
Flusso "normale"	1. Utente apre la pagina di registrazione 2. Presentata FORM che chiede tutti i dati 3. Utente compila e invia 4. Utente viene reindirizzato alla pagina richiesta
Flussi "eccezionali"	Utente inserisce qualche dato sbagliato
Dipendenze da altri use-case	-

Use case	**Gestire prenotazioni**
Attori	○ User
Flusso "normale"	1. Utente apre pagina utente 2. Presentata una tabella con tutte le prenotazioni che può gestire 3. Utente può cliccare per aprire la scheda della prenotazione o cancellare la prenotazione
Flussi "eccezionali"	○ Utente apre/cancella una prenotazione che risulta cancellata
Dipendenze da altri use-case	○ Login: <include>

Use case	**Mandare messaggi**
Attori	○ User
Flusso "normale"	1. Utente apre pagina contatti 2. Presentata FORM di contatti 3. Utente manda la FORM
Flussi "eccezionali"	○ Utente non scrive nulla nel messaggio
Dipendenze da altri use-case	-

Use case	**Gestire messaggi**
Attori	○ Admin
Flusso "normale"	1. Admin apre la pagina amministratore 2. Mostrata tabella con tutti i messaggi da completare

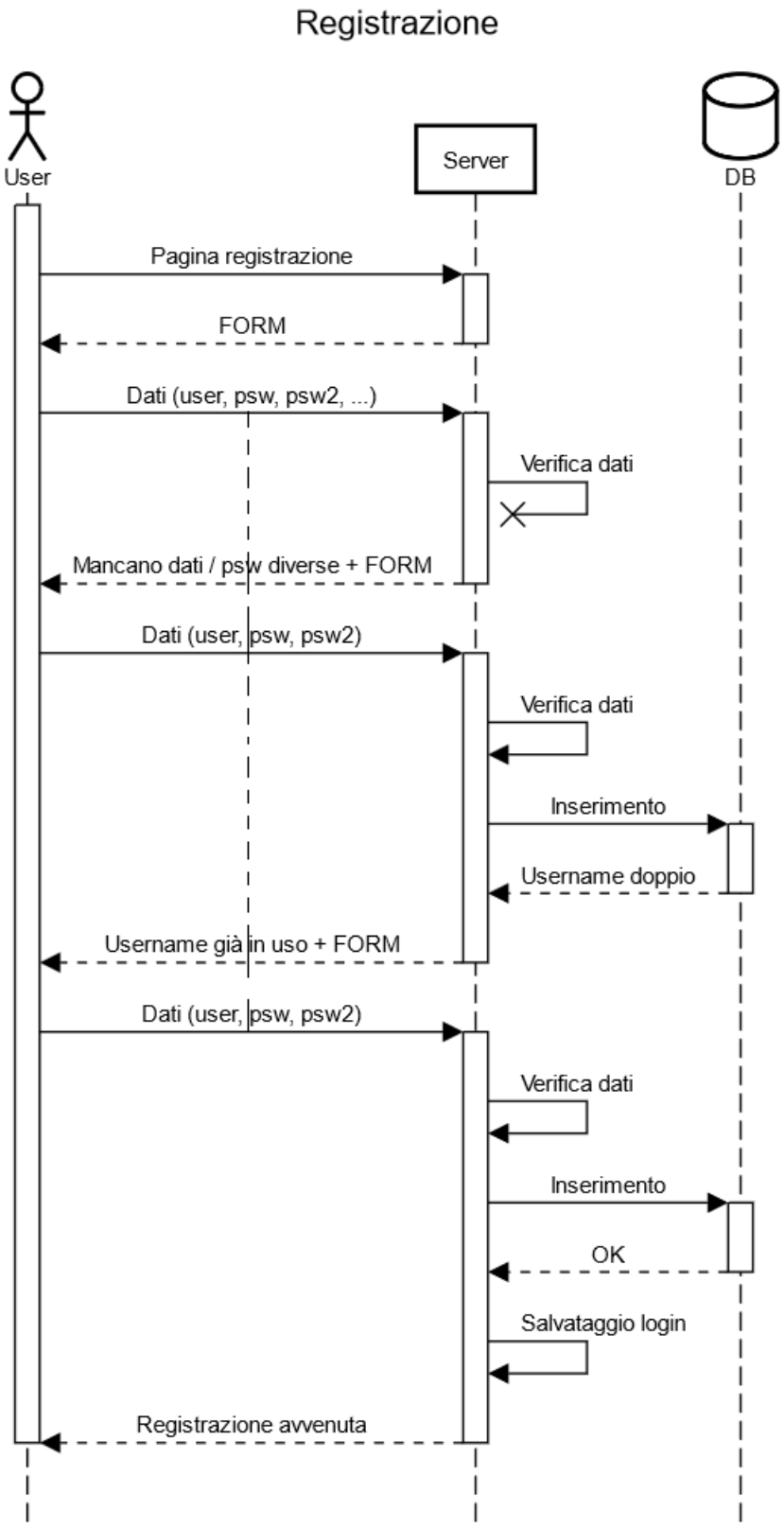
	3. Admin sceglie se rispondere al messaggio o segnarlo come "Completato"
Flussi "eccezionali"	-
Dipendenze da altri use-case	o Login: <include>

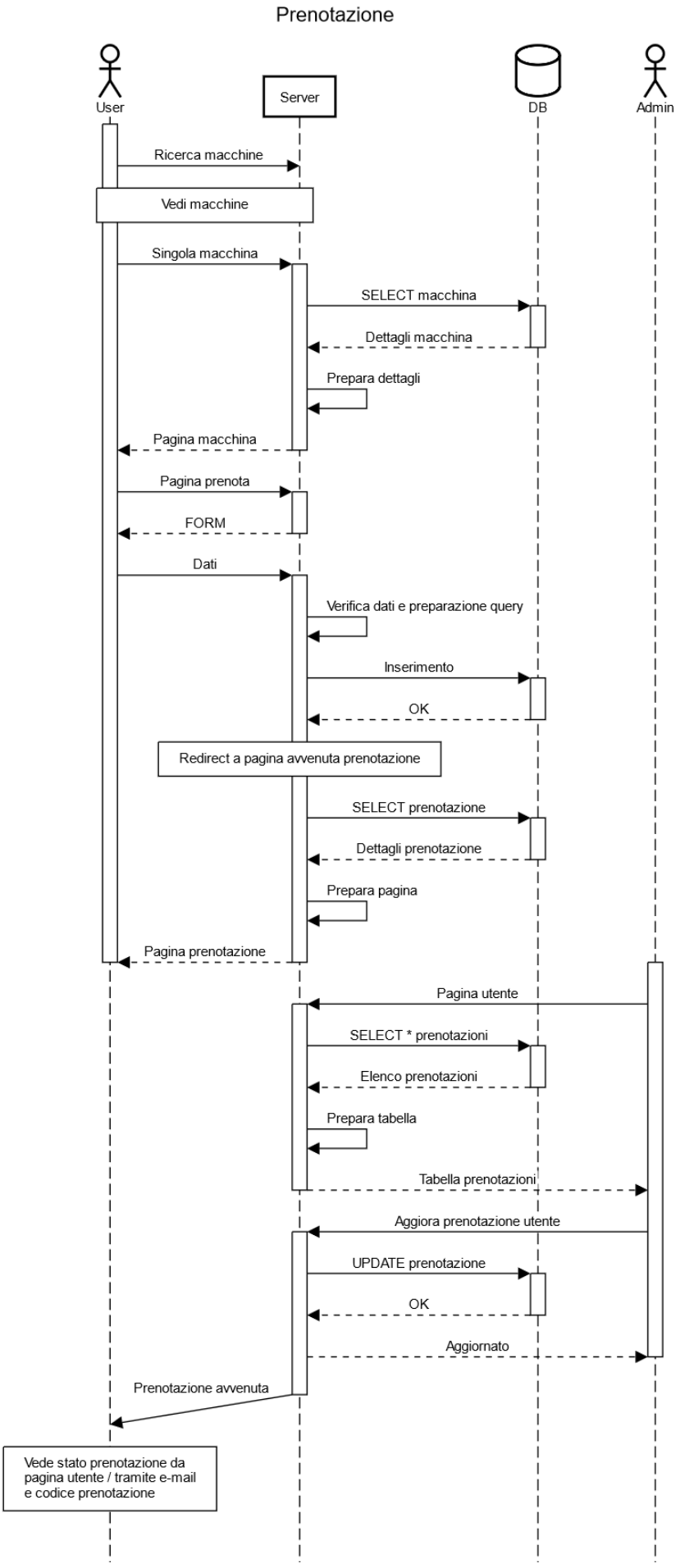
Use case	**Gestire macchine**
Attori	o Admin
Flusso "normale"	1. Admin apre la pagina amministratore 2. Mostrata tabella con tutti e le macchine 3. Admin sceglie se modificare una macchina o aggiungerne una nuova
Flussi "eccezionali"	o Admin vuole cancellare una macchina che non esiste o Admin vuole rimuovere di 1 quantità una macchina che ha già 0 quantità disponibili
Dipendenze da altri use-case	o Login: <include>

Use case	**Vedere macchine**
Attori	o User
Flusso "normale"	1. Utente apre la pagina macchine
Flussi "eccezionali"	o Non ci sono macchine
Dipendenze da altri use-case	-

Use case	**Prenotare macchina**
Attori	o User, Admin
Flusso "normale"	2. Utente dalla pagina macchine clicca su una macchina 3. Utente prende visione dei dettagli della macchina 4. Utente conferma di volere quella macchina 5. Viene presentato un FORM che chiede dei dettagli circa il noleggio 6. Utente invia il FORM 7. Utente viene reindirizzato alla pagina di prenotazione 8. Admin apre la pagina utente (admin) 9. Admin clicca per confermare/cancellare la prenotazione dell'utente
Flussi "eccezionali"	o Non ci sono macchine disponibili o Si verifica un errore durante la registrazione del noleggio
Dipendenze da altri use-case	o Vedere macchine <include>

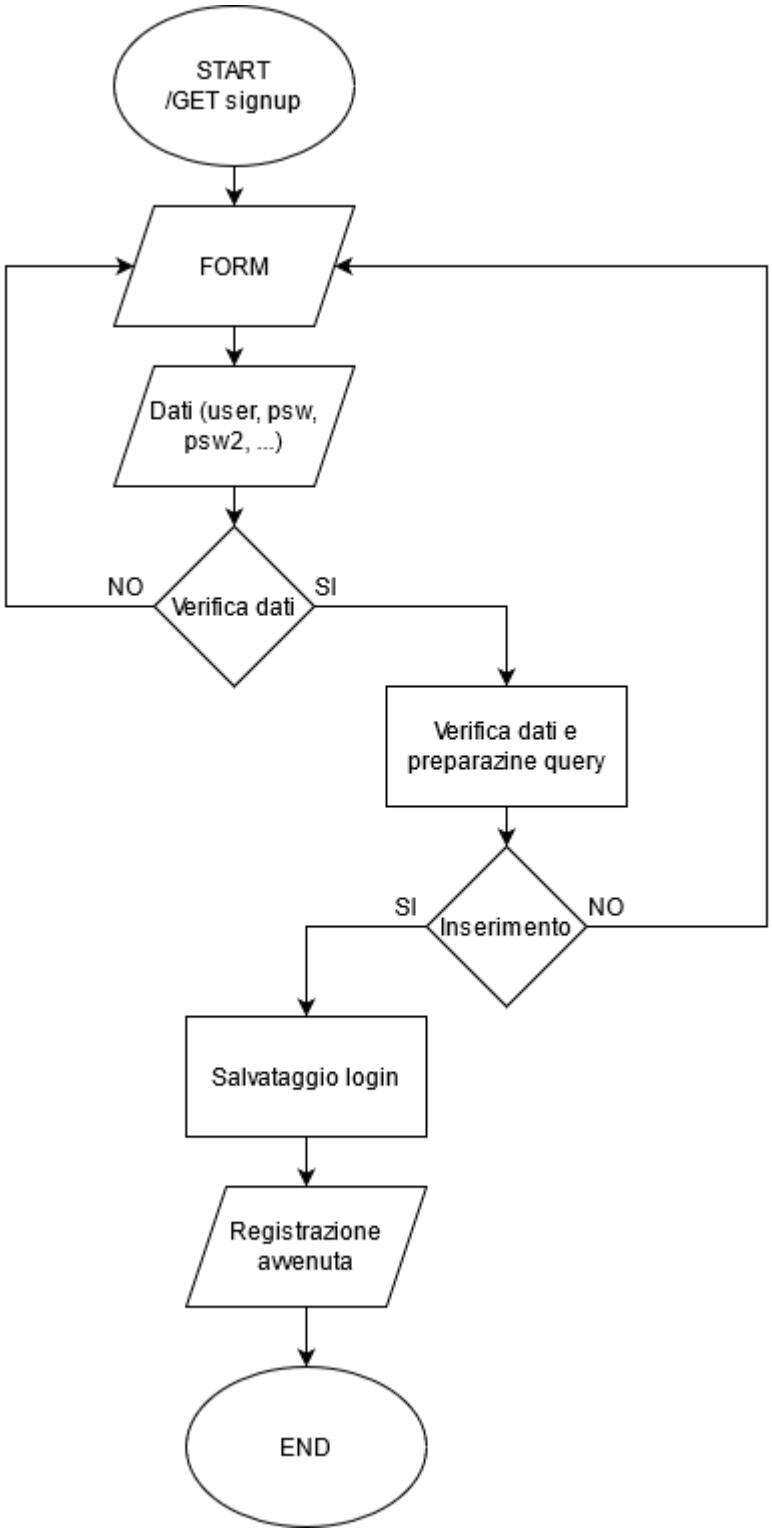
Sequence diagrams:



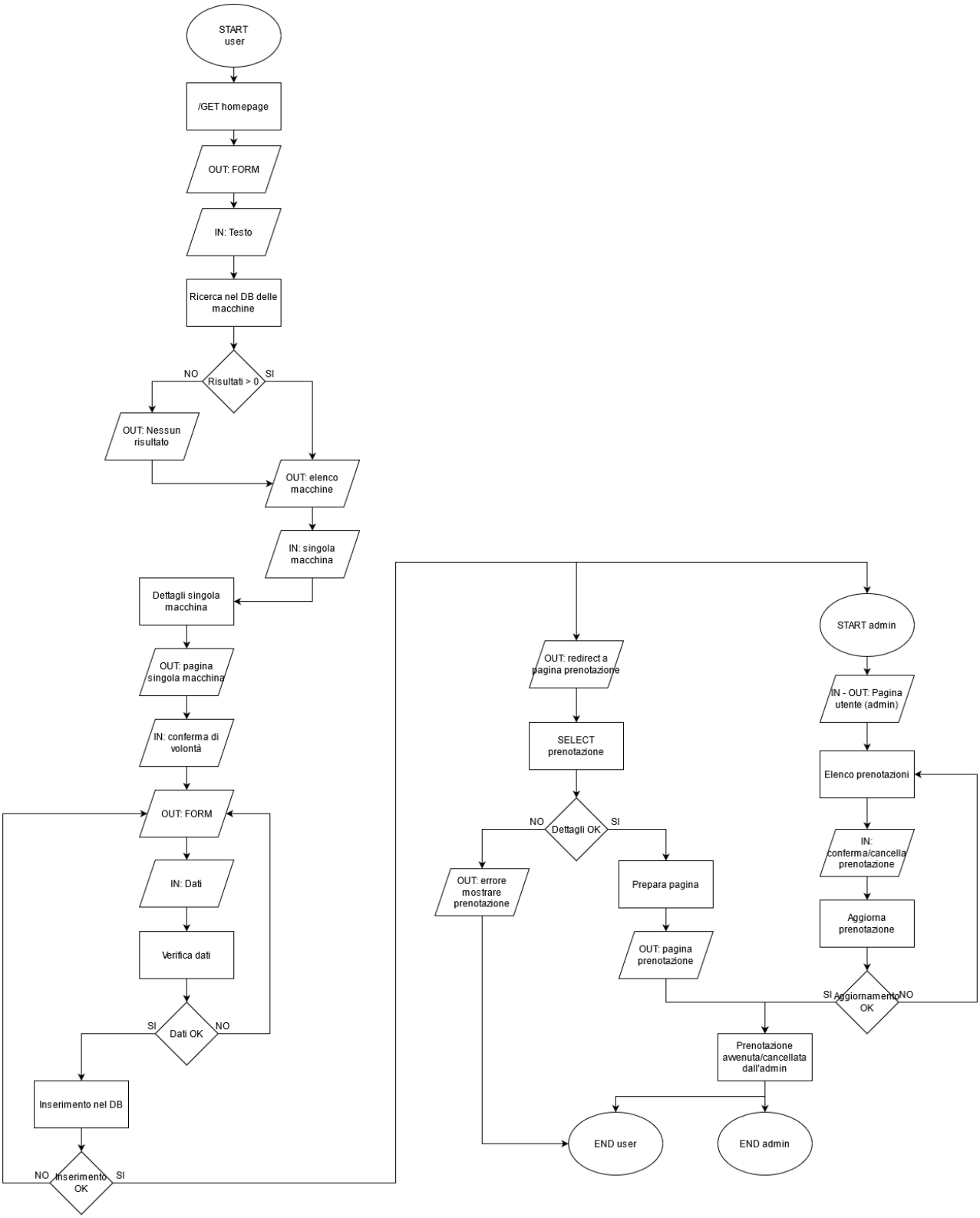

Prenotazione
User
Server
DB
Admin
Ricerca macchine
Vedi macchine
Singola macchina
SELECT macchina
Dettagli macchina
Prepara dettagli
Pagina macchina
Pagina prenota
FORM
Dati
Verifica dati e preparazione query
Inserimento
OK
Redirect a pagina avvenuta prenotazione
SELECT prenotazione
Dettagli prenotazione
Prepara pagina
Pagina prenotazione
Pagina utente
SELECT * prenotazioni
Elenco prenotazioni
Prepara tabella
Tabella prenotazioni
Aggiora prenotazione utente
UPDATE prenotazione
OK
Aggiornato
Prenotazione avvenuta
Vede stato prenotazione da pagina utente / tramite e-mail e codice prenotazione

Activity diagrams:

Registrazione



Prenotazione macchina



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.

Per il corretto funzionamento del sito è necessario configurare correttamente il rewrite degli URLs, in uno dei due metodi:

1. Cambiare la DocumentRoot del webserver dal file httpd.conf:
   DocumentRoot "C:/xampp/htdocs/Pra"
   <Directory "C:/xampp/htdocs/Pra">
2. Impostare dal file /utils/require.php il path per sovrascrivere il *DocumentRoot* e impostarlo alla cartella del progetto (istruzioni commentate nel file), successivamente inserire nella root del webserver (solitamente *htdocs*) un file *.htaccess* che contenga questa regola:
   RewriteEngine on
   RewriteCond %{REQUEST_URI} !Pra/
   RewriteRule (.) /Pra/$1 [L]*

Il progetto viene consegnato in una cartella *Pra.zip* che contiene ed è già configurata seguendo il metodo 2 precedentemente descritto. Questa cartella contiene la cartella contenente il codice del progetto (*Pra*) e il file *.htaccess* da mettere nella root di *htdocs*. Andando quindi ad esplodere lo zip direttamente in *htdocs*, avendo la cartella *Pra* figlia diretta di *htdocs*, il progetto risulterà funzionante senza ulteriori interventi.

Per usufruire del sito, navigare su http://localhost/

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.

Utilizzo delle sessioni:

Le sessioni sono utilizzate per due scopi distinti:

1. **Mantenimento della sessione di login** → in ogni pagina viene istanziato un oggetto User che nel costruttore verifica che esista una sessione di login attiva, recupera i dati salvati di quest'ultima (nome, cognome, username dell'utente, in quanto necessari in ogni pagina per stampare l'header), altrimenti questo oggetto avrà un attributo *loggedIn* settato a false, permettendo di gestire correttamente i vari casi
2. **Fase di noleggio** →
    a. Utente effettua una ricerca testuale su 3 campi (Brand, Model e Category) del veicolo tramite richiesta GET da pagina Home a pagina */cars*
    b. Nella pagina */cars* gli ID delle macchine sono inseriti come value di tanti *<button type="submit">* con lo stesso name, racchiusi tutti in un grande form contenitore
    c. Cliccando sul bottone di submit di una macchina viene richiesta tramite GET la pagina */cars/item.php?* la quale salva nella sessione l'ID della macchina recuperato tramite GET, così da non doverlo passare ogni volta tramite richiesta HTTP
    d. Nella pagina */cars/item.php* l'utente cliccando sul pulsante di prenota viene riportato alla pagina */cars/rent.php* (accessibile anche per link diretto dalla Homepage) controlla l'esistenza dell'ID della macchina in sessione, se presente disabilita la SELECT di scelta della macchina, altrimenti resta attiva.
    e. Eseguendo il submit del form lo script PHP controlla l'esistenza nell'array $_REQUEST di una chiave "car_id" in quanto l'utente potrebbe sempre decidere di effettuare il reset del form in */cars/rent.php* e quindi cambiare la macchina scelta tramite il "flusso normale"
    f. Lo script PHP registra la prenotazione e salva l'ID della prenotazione in sessione, così da poter mostrare in modo efficace la pagina di avvenuta prenotazione (*/cars/rented.php*) senza dover passare parametri tramite querystring (nonostante questo metodo sia comunque permesso per poter visualizzare le prenotazioni in tempi differiti al processo di prenotazione)

    L'alternativa all'uso delle sessioni per questa fase prevedeva l'uso massivo di querystring "imposte dal server" e l'uso di campi hidden nei form per poter salvare i dati