7 WEB APP SVILUPPATA

La batteria, vero cuore delle auto elettriche, è l'elemento più importante, complesso e costoso di tutto il veicolo. Non rappresenta, infatti, soltanto la riserva energetica, ma un vero e proprio organo vitale con una tecnologia in costante evoluzione. La batteria è vista come una sorta di serbatoio in cui immagazzinare l'energia necessaria a far muovere l'auto, ma in realtà è molto più di questo. Anche se la spinta arriva dal motore, le prestazioni di quest'ultimo sono influenzate direttamente dalla batteria e dalla sua capacità di erogare l'energia. Quindi, si può dire che in realtà il vero "motore" dell'auto elettrica sia proprio la batteria. Per valutare le prestazioni delle batterie, e di conseguenza dell'auto, bisogna considerare la sua capacità, ovvero la quantità di energia che può immagazzinare, il modo e la rapidità con cui l'energia viene immagazzinata ed erogata al motore, tutte caratteristiche espresse da amperaggio (quantità di energia che può essere trasmessa in un secondo)e da voltaggio (corrisponde alla forza, dunque alla rapidità, con cui una quantità di energia apri a 1 Ampere viene trasmessa).Dunque, è facile comprendere come all'aumentare di voltaggio e amperaggio la batteria risulti più potente perché il flusso di corrente che entra o esce sarà maggiore per volume e intensità.

Lo scopo principale di questa web app è fornire un esempio del modello dei dati da associare al monitoraggio delle batterie di auto elettriche insieme ad un semplice meccanismo per la loro analisi, monitoraggio e visualizzazione.

Il codice di questa web app è scaricabile tramite il seguente link github: https://github.com/1092407/tesi

Tra i vari file è presente anche “istruzioni.pdf” dove sono contenute istruzioni per testare la web app.

7.1 Descrizione funzionalità

Web app per una casa automobilistica di auto elettriche utile per registrare clienti, gestire il parco auto e per visualizzare dati sulle batterie montate sulle auto vendute.

Livelli di accesso al sito e funzionalità:

1) Parte pubblica

Qui ci sono informazioni generali come “chi siamo”,” dove siamo”,privacy,ecc. Chiunque può accedere alla parte pubblica, dove c’è anche la possibilità di effettuare il login e andare in una dei livelli successivi in base al proprio livello di utenza (casa automobilistica o cliente).Non c’è la possibilità di registrarsi in quanto la casa automobilistica è già pre-registrata nel database e i clienti possono fare login solo dopo che la casa automobilistica ,nella sua area privata del sito, crea un profilo per ciascuno e fornisce loro le credenziali di accesso. In questo modo solo chi acquista un’auto può usare la web app per visualizzare i dati.

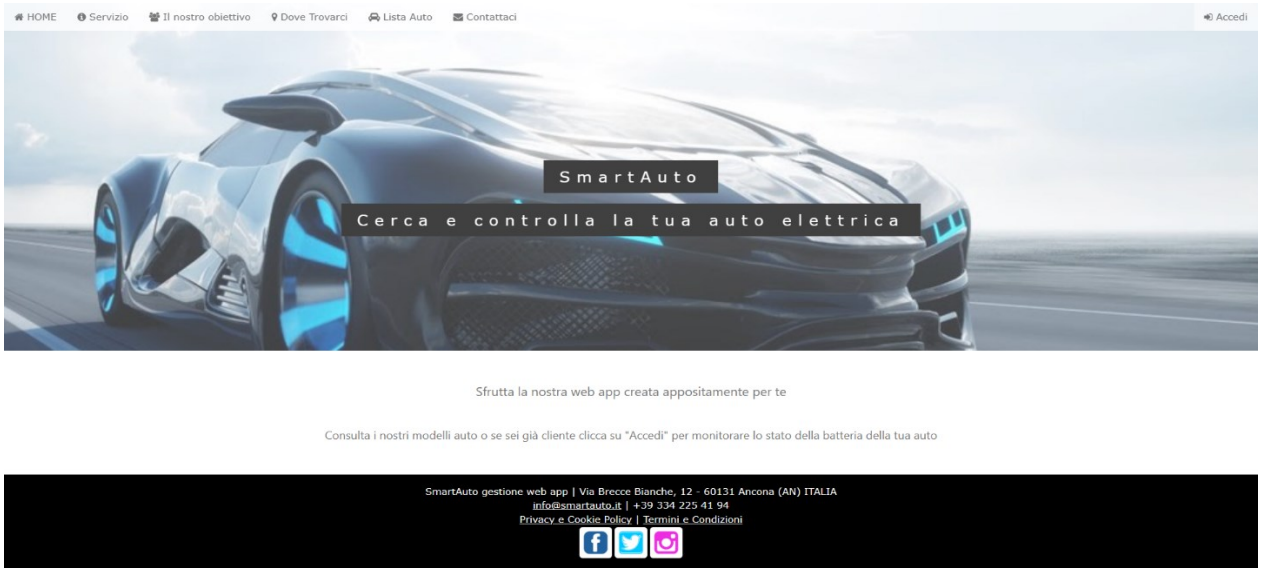

Figura 34: Homepage pubblica

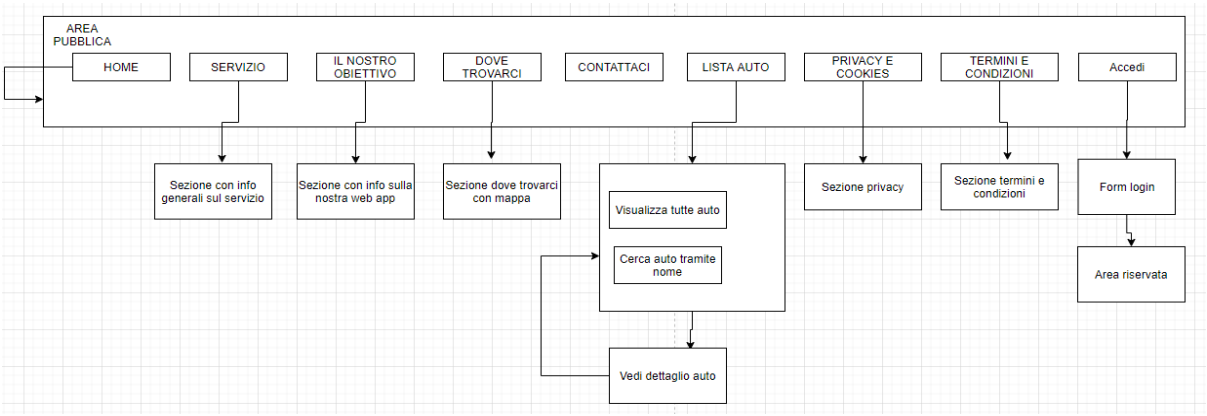

Figura 35: Schema dei link area pubblica

2) Livello casa automobilistica

Account pre-registrato nel database.

Crea il profilo ad ogni nuovo cliente che acquista un'auto, avendo anche la possibilità di modificare/eliminare dati relativi ad ogni cliente.

Gestisce una lista con tutte le auto che sono nel suo parco auto, avendo possibilità di inserire/modificare/eliminare dati relativi ad esse; queste sono anche visualizzabili nella parte pubblica per attirare nuovi potenziali clienti. Visualizza i dati sulle batterie delle auto dei clienti.

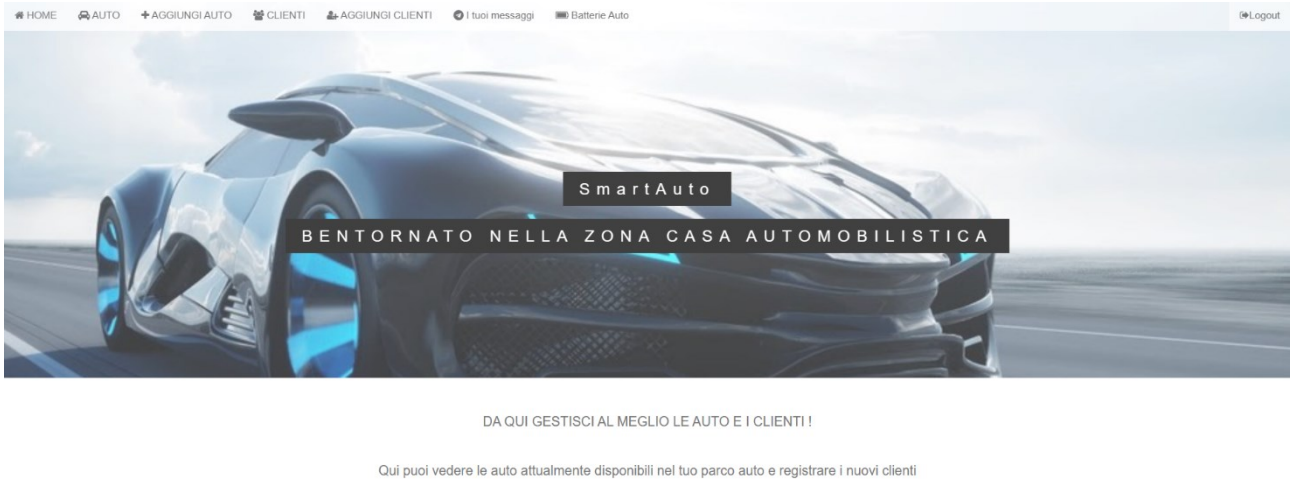

Figura 36: Homepage livello casa automobilistica

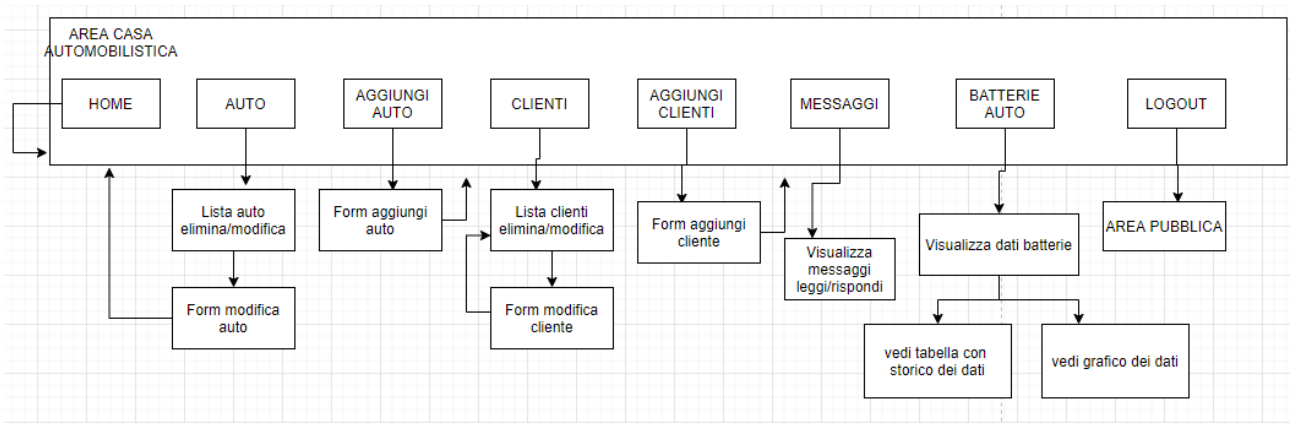

Figura 37: Schema dei link area casa automobilistica

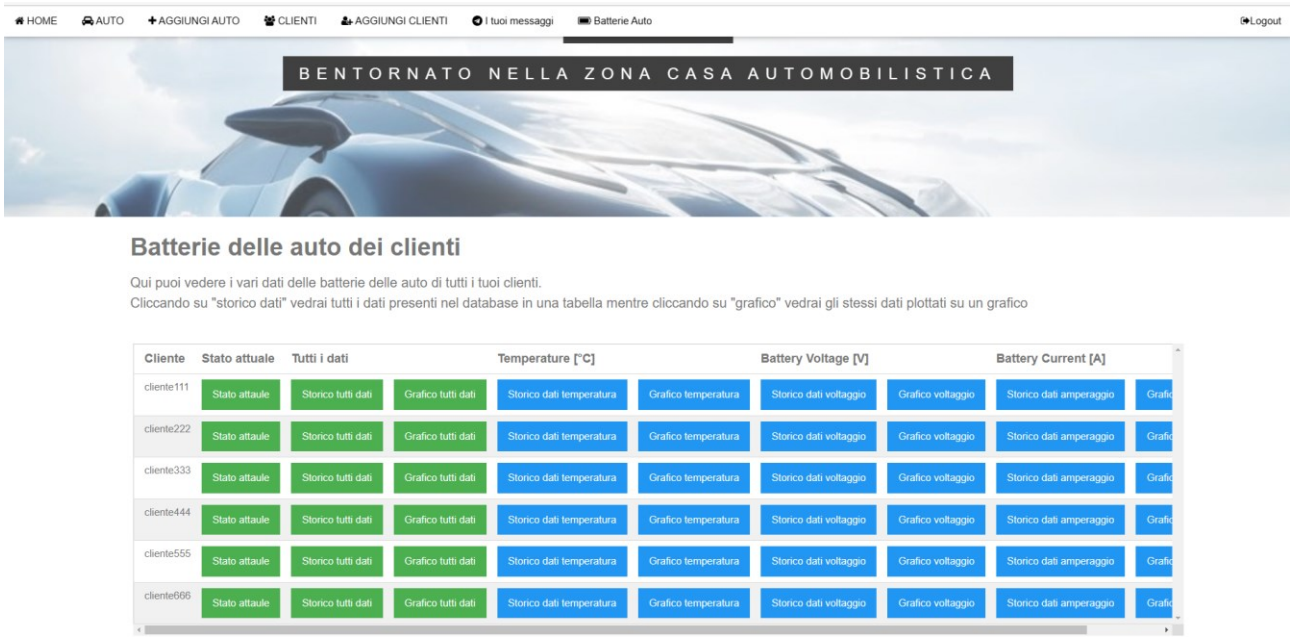

Figura 38: porzione della schermata dove la casa automobilistica può scegliere quali dati andare a visualizzare relativamente alla batteria di un certo cliente

3) Livello cliente

Ogni cliente può visualizzare dati relativi alla batteria della propria auto, ma questi sono meno dettagliati e specifici rispetto a quelli che può visualizzare il fornitore.

Ogni cliente ha una chat dedicata con la casa automobilistica attraverso la quale può richiedere assistenza o ulteriori informazioni sulla propria auto.

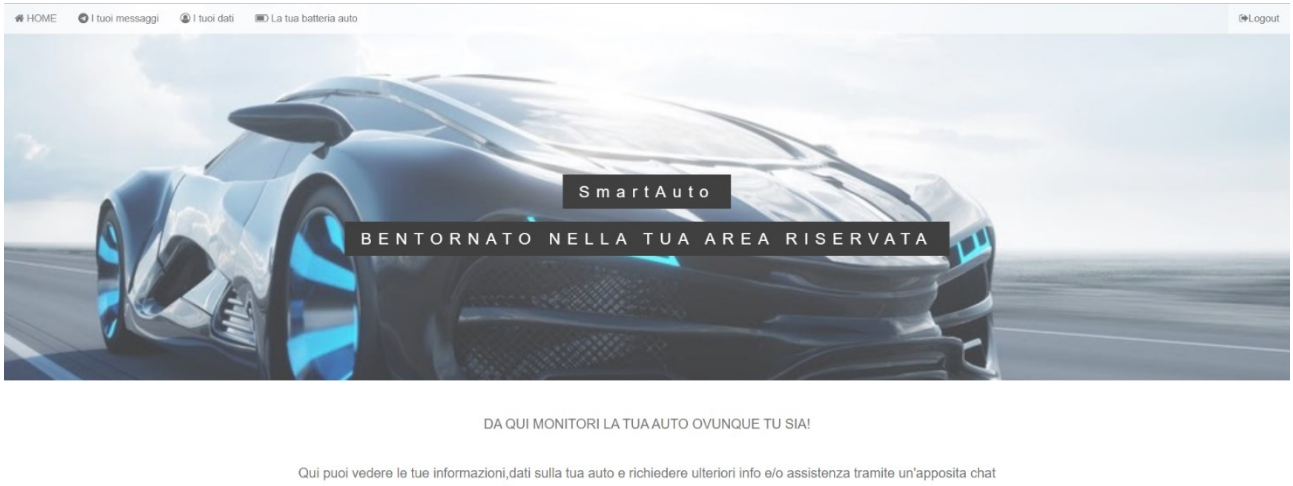

Figura 39 : Homepage livello cliente

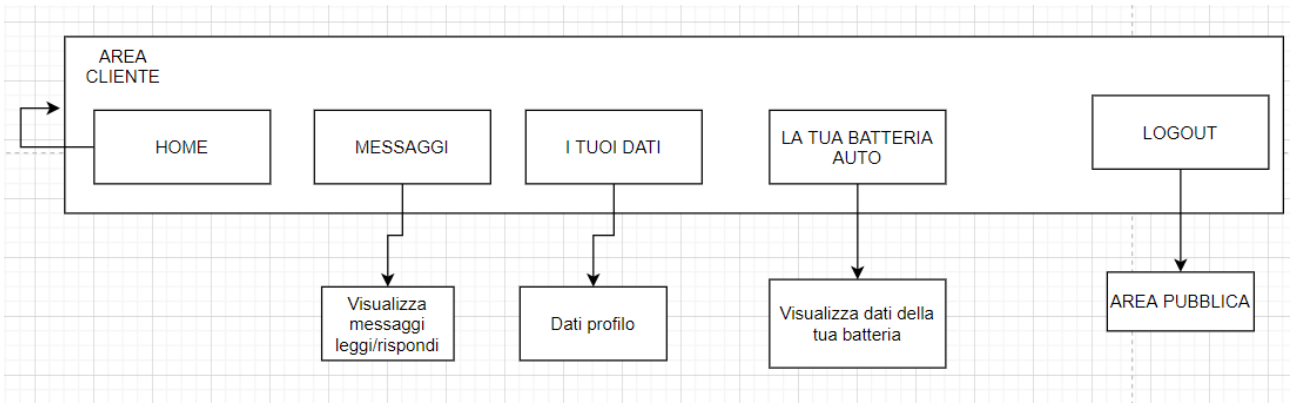

Figura 40: Schema dei link area cliente

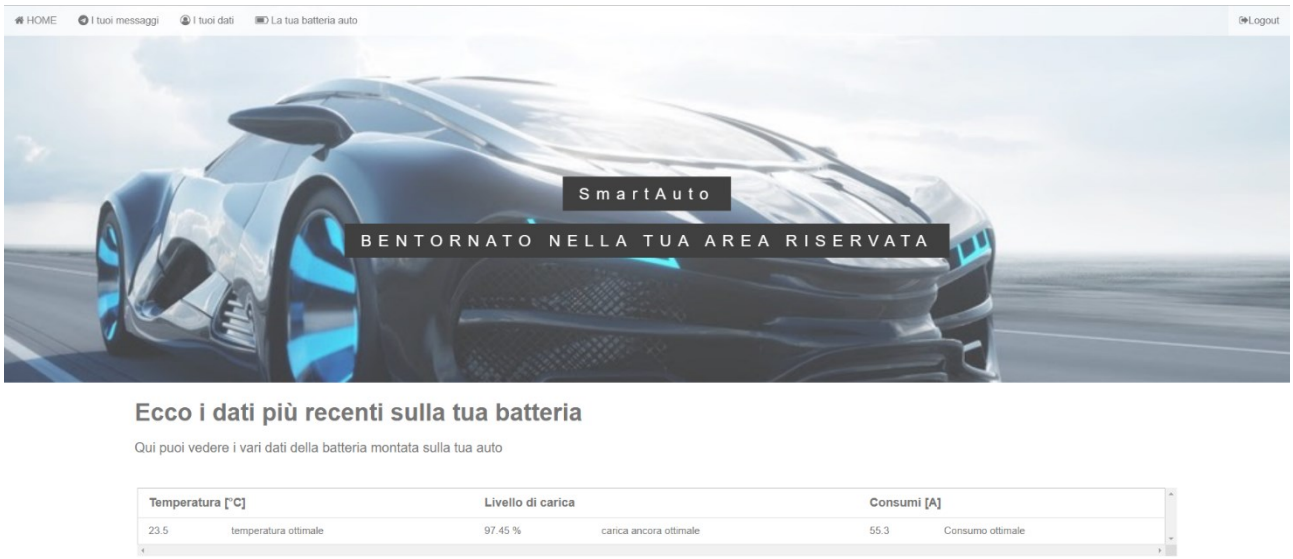

Figura 41: schermata dove un cliente può vedere lo stato attuale della batteria della sua auto

7.2 Dati utilizzati

In uno sviluppo futuro di questo progetto i dati della batteria saranno inviati in tempo reale in cloud e monitorati dall'utente. In questa tesi i dati sono presi dal dataset di dati reali disponibile online (https://ieee-dataport.org/open-access/battery-and-heating-data-real-driving-cycles). I dati sono stati acquisiti nel 2020 da Matthias Steinstraeter, Johannes Buberger e Dimitar Trifonov della Technical University of Munich, Institute of Automotive Technology.

Questi dati sono stati inseriti nel database collegato alla web app tramite il meccanismo dei "seeders", come meglio descritto nella sezione "Implementazione".

In particolare questi dati sono stati registrati durante diversi viaggi di guida reale con un'automobile elettrica ed utilizzati durante uno studio per la convalidazione di un modello completo di veicolo costituito dal gruppo propulsore e dal circuito di riscaldamento dove, per ogni viaggio, ad intervalli regolari di 100 millisecondi , sono stati acquisiti dati ambientali (come temperatura ed altitudine), dati del veicolo (come velocità e acceleratore), dati sulla batteria

(tensione, corrente, temperatura, SoC) e dati sul circuito di riscaldamento (come temperatura interna, potenza di riscaldamento, ecc.).
I dati si riferiscono a 70 percorsi di durata di circa 30 minuti per un percorso di circa 20 Km. Per ogni viaggio sono riassunte le seguenti informazioni generali:

- Trip #
- Date
- Route/Area
- Weather
- Battery Temperature (Start) [°C]
- Battery Temperature (End)
- Battery State of Charge (Start)
- Battery State of Charge (End)
- Ambient Temperature (Start) [°C]
- Target Cabin Temperature
- Distance [km] Duration [min]
- Fan

Durante ogni viaggio sono acquisiti i seguenti dati con tempo di campionamento di 100ms:

- Time [s]
- Velocity [km/h]
- Elevation [m]
- Throttle [%]
- Motor Torque [Nm]
- Longitudinal Acceleration [m/s^2]
- Regenerative Braking Signal
- Battery Voltage [V]
- Battery Current [A]
- Battery Temperature [°C]
- max. Battery Temperature [°C]
- SoC [%]
- displayed SoC [%]

- min. SoC [%]
- max. SoC [%)
- Heating Power CAN [Kw]
- Heating Power LIN [W]
- Requested Heating Power [W]
- AirCon Power [Kw]
- Heater SignalHeater Voltage [V]
- Heater Current [A]
- Ambient Temperature [°C]
- Coolant Temperature Heatercore [°C]
- Requested Coolant Temperature [°C]
- Coolant Temperature Inlet [°C]
- Heat Exchanger Temperature [°C]
- Cabin Temperature Sensor [°C]

Inoltre sono acquisiti con tempo di campionamento di 100ms i seguenti dati:

- Environment
    - Slope [rad]
    - Wind Speed [m/s]
    - Ambient Temperature [C]
- Longitudinal Vehicle
    - Longitudinal Acceleration [m/s^2]
    - Speed [m/s]
    - Position [m]
    - Traction Force [N]
    - Drive Power [W]
- Drive Train
    - Drive Torque [Nm]
    - Wheel Torque [Nm]
    - Drive Force Front [N]
    - Drive Force Rear [N]
    - Roll Resistance Coeff [-]
    - Mass Jred [kg]

    - Drive Current [A]
    - Engine Speed[1/min]
    - Wheel Rot Speed [rad/s]
    - max Charging Power [W]
    - EM Power [Kw]
- Battery
    - Pack Voltage [V]
    - Battery Current [A]
    - Power [Kw]
    - SOC [%]
    - Ageing SOH [-]
    - Mean Cell Temperature [C]
- Consumption
    - Power [Kw]
    - Energy [kWh]
    - [kWh/100km]
- High Voltage Heater
    - Cabin Temperature [C]
    - Heating Power [Kw]
    - Heating Power inclPeak [Kw]
    - Heatexchanger Inlet Coolant Temp [C]
    - Heatexchanger Outlet Air Temp [C]

- Auxiliaries
    - Seat Heating [W]
    - Brake Lights [W]
    - Rest Lights [W]
    - Window Lifters [W]
    - Infotainment [W]
    - Wipers [W]
    - Blower [W]
    - CU Sensors [W]
    - Auxiliaries Total [W]

7.3 Implementazione

Il framework utilizzato per lo sviluppo è stato ”Laravel 6” in quanto è una versione LTS (Long Term Support) , cioè con affidabile supporto e con aggiornamenti relativi a sicurezza, manutenzione e funzionalità per un lungo periodo di tempo che la rendono una versione stabile a seguito dei molti test approfonditi che ha superato.
Altro strumento fondamentale è stato XAMPP. Si tratta di un software composta da Apache http Server, MySQL (o una sua fork chiamata MariaDB) e PHP che facilita l’installazione e la gestione
degli strumenti più comuni per lo sviluppo di applicazioni web, raggruppandoli in un unico luogo, e particolarmente utile per il testing.

7.3.1 Visualizzazione dei dati

Oltre ad essere visualizzati tramite tabelle HTML, alcuni dati delle batterie sono visualizzabili tramite appositi grafici che sfruttano “Laravel Charts”, un’apposita libreria di grafici creata per supportare Laravel nella visualizzazione di dati.

Time	Temperature [°C]	Battery Voltage [V]	Battery Current [A]	SOC [%]
2022-03-11 12:03:00	23.5	389.8	55.3	75
2022-03-11 12:02:00	23.6	391.3	45.8	75.3
2022-03-11 12:01:00	23.8	381.4	23.8	75.5
2022-03-11 12:00:00	22.5	387.7	27.8	75.7
2022-03-10 12:05:00	24.5	391.3	57.8	69.1
2022-03-10 12:04:00	23.5	381.4	19.8	69.2
2022-03-10 12:03:00	21.5	389.8	37.8	69.4
2022-03-10 12:02:00	22.5	376.2	17.8	69.6
2022-03-10 12:01:00	23.5	387.7	12.8	69.8
2022-03-10 12:00:00	24.5	391.3	21.8	69.9
2022-03-09 12:04:00	26.1	376.2	35.5	72.1
2022-03-09 12:03:00	26.5	382.3	37.8	72.3
2022-03-09 12:02:00	25.5	381.4	47.8	72.4
2022-03-09 12:01:00	22.7	387.7	23.8	72.6
2022-03-09 12:00:00	22.5	391.3	57.8	72.8
2022-03-08 12:03:00	22.7	376.2	37.8	83
2022-03-08 12:02:00	23.5	392.1	77.8	83.3
2022-03-08 12:01:00	23.6	389.8	57.8	83.5
2022-03-07 12:00:00	24.5	392.3	47.8	85.9

Figura 42: esempio di una tabella HTML per leggere lo storico dei dati presenti nel database

Laravel Charts è una libreria di grafici disponibile come pacchetto PHP in grado di generare un elevato numero di combinazioni di grafici. L'API di Chart è progettata per essere estensibile e personalizzabile, consentendo di utilizzare rapidamente qualsiasi opzione nella libreria JavaScript.

Oggi i grafici sono per lo più realizzati in JavaScript, ma i dati che contengono molto probabilmente provengono da un database e Laravel è quello che li gestisce. Questa libreria offre la possibilità di non usare direttamente codice JavaScript perché crea una variabile PHP a cui viene associato un grafico, consentendone la personalizzazione. [52]

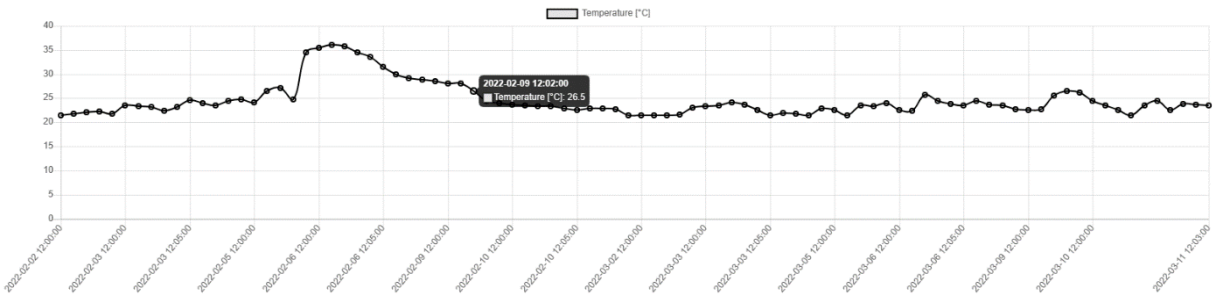

Figura 43: esempio di un grafico utilizzato nella web app e che legge dati sulla temperatura della batteria

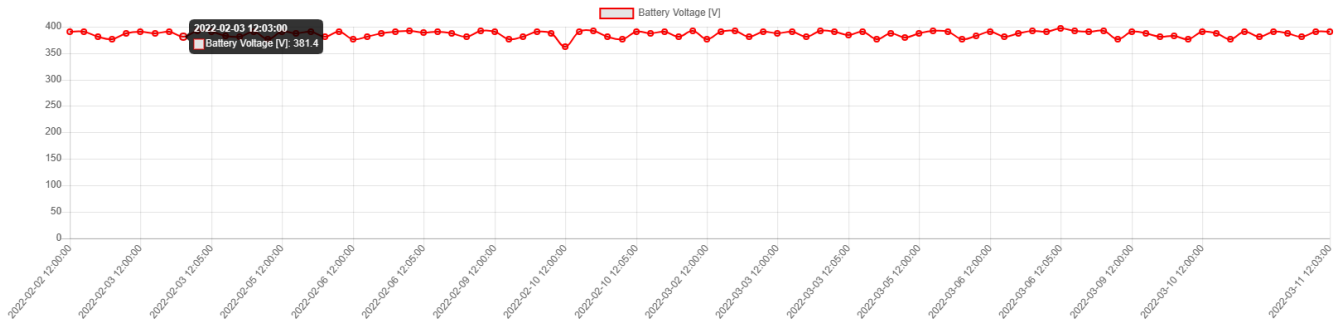

Figura 44: esempio di un grafico utilizzato nella web app e che legge dati sul voltaggio della batteria

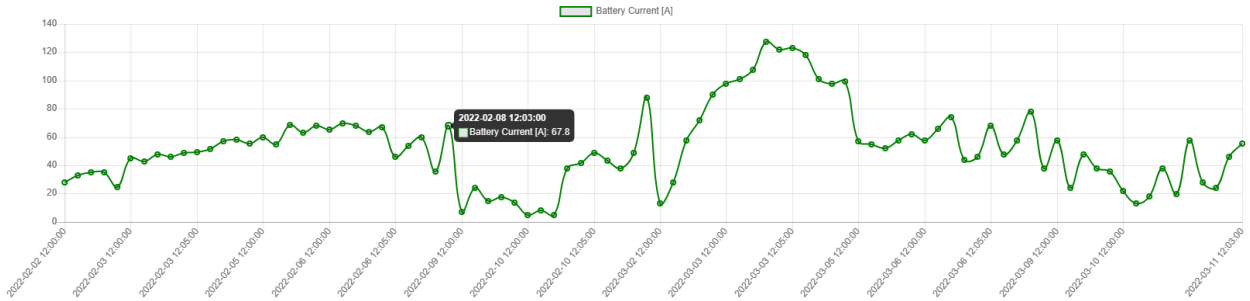

Figura 45: esempio di un grafico utilizzato nella web app e che legge dati sulla corrente della batteria

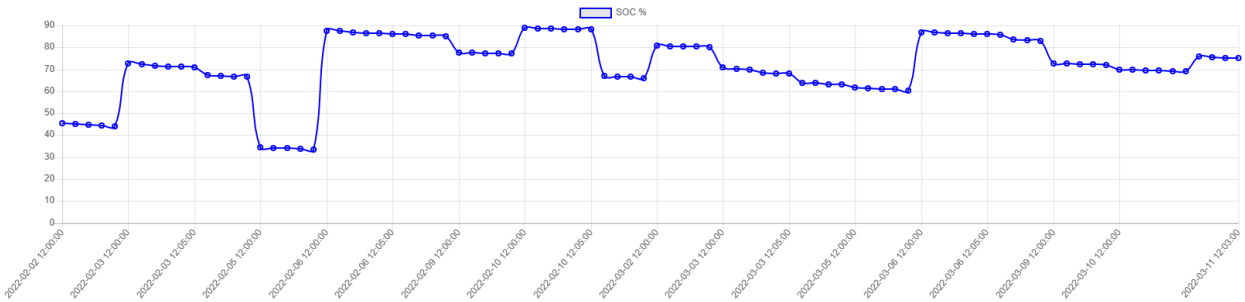

Figura 46: esempio di un grafico utilizzato nella web app e che legge dati sulla SOC della batteria

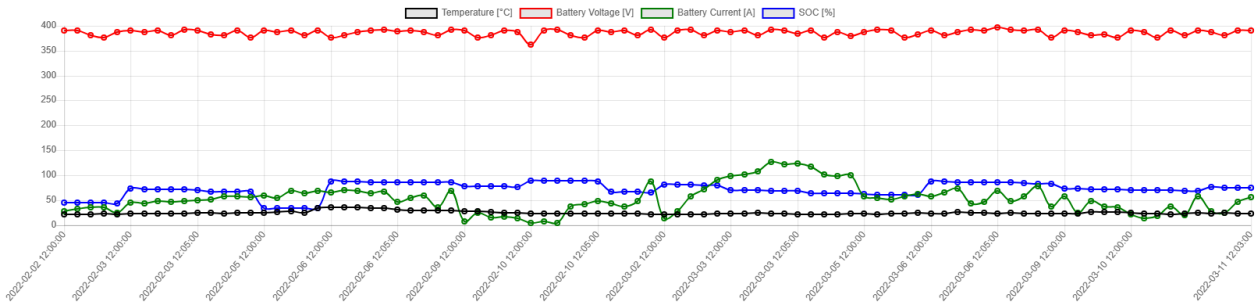

Figura 47: esempio di un grafico utilizzato nella web app e che legge contemporaneamente tutti i della batteria

In particolare, per ogni grafico, viene riportato il tipo di dato che viene visualizzato, la data della rilevazione e il valore stesso del dato. Di seguito si riporta, come esempio, la funzione “ShowChartAll()” presente all’interno di “app/http/Controllers/CasaAutoController” che si occupa di recuperare i dati dal database e mandarli alla view dove vengono visualizzati attraverso un grafico.

```php
//questa per vedere dati di tutte le misurazioni
public function ShowChartAll($cliente){
$chart = new examplechart;
   $allid=Misurazioni::where("cliente",$cliente)->select("id")->orderBy("data", "asc")->get()->toArray();  //sono id delle misur
   $valoriTemp=[]; //una per  ogni tipo di dato
   $valoriVolt=[];
   $valoriAmp=[];
   $valoriSoc=[];  //ora popolo i vettori con i dati del db per poi passarli al grafico

   for ($i=0;$i<count($allid);$i++){
   $app1=Misurazioni::where("id",$allid[$i])->value("temperatura"); //uso una variabile di appoggio
   $valoriTemp[$i]=$app1;
   }

   for ($i=0;$i<count($allid);$i++){
   $app2=Misurazioni::where("id",$allid[$i])->value("voltaggio");
   $valoriVolt[$i]=$app2;
   }

   for ($i=0;$i<count($allid);$i++){
   $app3=Misurazioni::where("id",$allid[$i])->value("amperaggio");
   $valoriAmp[$i]=$app3;
   }

   for ($i=0;$i<count($allid);$i++){
   $app4=Misurazioni::where("id",$allid[$i])->value("soc");
   $valoriSoc[$i]=$app4;
   }

   $lab=[];  //questa per segnare le date sulle label:ne uso una sola perchè sono uguali per i vari tipi di dati che recupero
   for ($i=0;$i<count($allid);$i++){
   $appoggio=Misurazioni::where("id",$allid[$i])->value("data");
   $lab[$i]=$appoggio;
   }

   $chart->labels(array_values($lab));
   $chart->dataset('Temperature [°C]', 'line', array_values($valoriTemp))->options(['borderColor'=>'black','fill'=> 'false']);
   $chart->dataset('Battery Voltage [V]', 'line', array_values($valoriVolt))->options(['borderColor'=>'red','fill'=> 'false']);
   $chart->dataset('Battery Current [A]', 'line', array_values($valoriAmp))->options(['borderColor'=>'green','fill'=> 'false']);
   $chart->dataset('SOC [%]', 'line', array_values($valoriSoc))->options(['borderColor'=>'blue','fill'=> 'false']);
 return view('sample_view', compact('chart'));
}
```

Figura 48 : funzione che recupera i dati del database di una certa batteria e li usa per costruire il grafico

7.3.2 Modello dei dati

Laravel consente la gestione dell'evoluzione della struttura della base di dati e dei suoi contenuti durante lo sviluppo di un'applicazione attraverso Migrations e Seeding.

Migrations è un meccanismo di definizione della struttura del DB (tabelle relazioni tra di esse) che consente il versioning :

-Basato sulla definizione di classi che estendono la classe Laravel predefinita Illuminate\Database\Migrations\Migration

- Le classi consentono di definire lo schema delle tabelle utilizzando la FACADE Schema in due metodi pubblici: up() e down()
- I file che definiscono le classi sono memorizzati nella directory database/migrations

Seeding è un processo di popolamento delle tabelle del DB con valori utili per lo sviluppo ed il test dell'applicazione:
- Basato sulla definizione di classi che estendono la classe Laravel predefinita Illuminate\Database\Seeder
- Ogni classe contiene un unico metodo pubblico, run(), che usa il Query Builder di Laravel per inserire valori nelle tabelle
- I file che definiscono i seeder sono memorizzati nella directory database/seeds

C'è la necessità di definire un meccanismo di interfaccia tra l'applicazione Laravel e le sorgenti di dati persistenti, in particolare lo scambio di dati con il DBMS. L'interfaccia si sviluppa su due livelli:

-Connessione dell'applicazione Laravel al DBMS tramite il meccanismo del PDO (PHP Data Object: consente astrazione dei dati in modo indipendente dal DBMS) attraverso i file ".env" e "config/database.php"

```
'connections' => [

    'mysql' => [
        'driver' => 'mysql',
        'url' => env('DATABASE_URL'),
        'host' => env('DB_HOST', '127.0.0.1'),
        'port' => env('DB_PORT', '3306'),
        'database' => env('DB_DATABASE', 'tesi'),
        'username' => env('DB_USERNAME', 'root'),
        'password' => env('DB_PASSWORD', ''),
        'unix_socket' => env('DB_SOCKET', ''),
        'charset' => 'utf8mb4',
        'collation' => 'utf8mb4_unicode_ci',
        'prefix' => '',
        'prefix_indexes' => true,
        'strict' => true,
        'engine' => null,
        'options' => extension_loaded('pdo_mysql') ? array_filter([
            PDO::MYSQL_ATTR_SSL_CA => env('MYSQL_ATTR_SSL_CA'),
        ]) : [],
    ],

],
```

Figura 49: dettaglio file "config/database"

```
APP_NAME=Laravel
APP_ENV=local
APP_KEY=base64:ZB0JX5iNWdu9MfZn2Pzu6PFza/dZ+ExvmIRqKoc/ToA=
APP_DEBUG=true
APP_URL=http://localhost

LOG_CHANNEL=stack

DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=tesi
DB_USERNAME=root
DB_PASSWORD=

BROADCAST_DRIVER=log
CACHE_DRIVER=file
QUEUE_CONNECTION=sync
SESSION_DRIVER=file
SESSION_LIFETIME=120
```

Figura 50 : file ".env"

-Accesso alle funzioni del DBMS per la manipolazione dei dati del database e la mappatura degli stessi dati in strutture dati PHP. Ci sono diversi metodi e componenti, ma il più efficace è Eloquent,ovvero il componente di Laravel che implementa un ORM (Object Relational Mapping) :è una tecnica di programmazione che effettua il mapping tra i dati visti come relazioni dal DBMS e il mondo PHP che li vede come oggetti.

Tramite appositi comandi Artisan (l'interfaccia a linea di comando di Laravel), una volta stabilita la connessione tra applicazione e DBMS, è possibile creare le tabelle del database proprio come sono state definite le migrations e poi popolarle con i dati inseriti nel seeder. In questo modo si può cambiare velocemente il DBMS collegato alla web app perché tramite due semplici comandi le tabelle e i dati (presenti nelle migrations e nei seeders) possono essere "spostati" da un DBMS all'altro senza la necessità di riscrive tutte le query ogni volta. [9]

All'interno dei files "database/migrations" e "databse/seeds" sono visibili le tabelle ed i dati inseriti ed utilizzati per le fasi di sviluppo e test della web app sviluppata.

7.3.3 Package utilizzati

I package o plugin sono lo strumento primario per estendere le funzionalità di Laravel. Sono componenti software integrabili nel framework e sviluppate per fornire nuove funzionalità o estendere quelle esistenti

Si dividono in :

- Stand alone: progettati per l'uso di qualsiasi package PHP
- Specifici: nati per integrarsi con lo specifico workflow di Laravel e le sue componenti

Questi vengono installati tramite Composer, un *dependecy manager* scritto in PHP che ci permette di installare e aggiornare le librerie esterne in modo molto semplice. Una volta definite le librerie che ci servono, Composer le scarica automaticamente e le inserisce nella cartella "vendor".

I package utilizzati in questa web app sono:

a)Laravel Collective

Package specifico che integra nel framework componenti:

- HTML : aiuta generare form e componenti HTML complesse con costrutti sintattici più semplici

-remote:per gestire le connessioni SSH all'interno dell'applicazione

-Annotation :per inserire annotazioni nel codice

```
{{ Form::open(array('route' => 'inserisciauto_post', 'files' => true, 'class' => 'contact-form')) }}
                <div class="wrap-input">
                    {{ Form::label('', '', ['class' => 'fa fa-picture-o']) }}
                    {{ Form::label('foto', 'Foto ', ['class' => 'label-input ']) }}
                    {{ Form::file('foto', ['class' => 'input', 'id' => 'foto']) }}
                    @if ($errors->first('foto'))
                    <ul class="errors">
                        @foreach ($errors->get('foto') as $message)
                        <li>{{ $message }}</li>
                        @endforeach
                    </ul>
                    @endif
                </div>
                <div class="wrap-input">
                    {{ Form::label('', '', ['class' => ' fa fa-id-card-o']) }}
                    {{ Form::label('marca', 'Marca', ['class' => 'label-input']) }}
                    {{ Form::text('marca', '', ['class' => 'input', 'id' => 'marca']) }}
                    @if ($errors->first('marca'))
                    <ul class="errors">
                        @foreach ($errors->get('marca') as $message)
                        <li>{{ $message }}</li>
                        @endforeach
                    </ul>
                    @endif
                </div>
                <div class="wrap-input">
                    {{ Form::label('', '', ['class' => 'fa fa-id-card-o']) }}
                    {{ Form::label('modello', 'Modello', ['class' => 'label-input ']) }}
                    {{ Form::text('modello', '', ['class' => 'input', 'id' => 'surname']) }}
                    @if ($errors->first('modello'))
                    <ul class="errors">
                        @foreach ($errors->get('modello') as $message)
                        <li>{{ $message }}</li>
                        @endforeach
                    </ul>
                    @endif
                </div>
            <div class="wrap-input">
                {{ Form::label('descrizione', 'Descrizione', ['class' => 'label-input']) }}
                {{ Form::textarea('descrizione', '', ['class' => 'input descrizione', 'id' => 'descrizione']) }}
                @if ($errors->first('descrizione'))
                <ul class="errors">
                    @foreach ($errors->get('descrizione') as $message)
                    <li>{{ $message }}</li>
                    @endforeach
                </ul>
                @endif
            </div>
            <div class="container-form-btn">
                {{ Form::submit('Inserisci', ['class' => 'my-button']) }}     </div>      {{ Form::close() }}
```

Figura 51 : esempio di una form costruita sfruttando Laravel Collective

b) Laravel UI

Package molto diffuso perché tramite la componente “auth” vengono integrati nell’applicazione tutti gli elementi necessari per gestire il processo di autenticazione. Inoltre, mette a disposizione la “facade auth” (facade è un pattern che fornisce un’interfaccia semplice per accedere a sottosistemi molto complessi) che consente di accedere ad informazioni legate all’utente attualmente autenticato memorizzato nelle sessioni PHP e attiva il processo di logout

7.3.4 Autenticazione e autorizzazione

L’autenticazione è il processo che verifica l’identità di un utente, che si qualifica attraverso le sue credenziali (solitamente username e password). In Laravel questo processo si basa su una serie di componenti già configurate al momento della creazione di un nuovo progetto che aiutano a gestire:

-I driver di autenticazione, cioè le sorgenti di informazioni per l’autenticazione

-Il processo di autenticazione attraverso una serie di controller predefiniti e associati diverse fasi

-La persistenza dell’identità di un utente utilizzando le sessioni PHP (di default)

L’autorizzazione è un’attività che verifica se qualcuno (ad esempio un utente) ha il permesso di compiere un’azione su una risorsa. In Laravel ci sono diversi meccanismi possibili per gestire e definire le autorizzazioni. Quello utilizzato in questa web app sfrutta i Gates, ovvero funzioni che definiscono le condizioni per cui l’utente autenticato può eseguire una certa azione , definiti in “app/Providers/AuthServiceProvider”. Le regole così definite possono essere sfruttate utilizzando il “Middleware Authorize” (un software che funge da intermediario tra applicazioni, strumenti e database) in modo tale che il meccanismo di autorizzazione entri in azione prima dell’attivazione di una rotta o di un’azione di un controller.

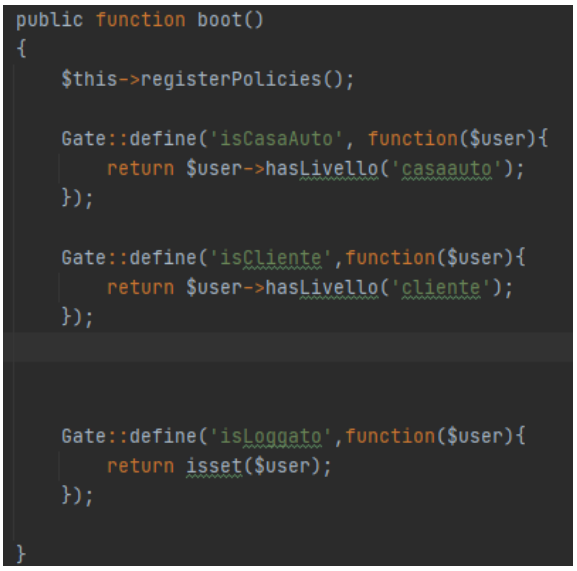

Figura 52 : porzione del file dove vengono definiti i Gates

Questi concetti sono stati utilizzati per garantire ai diversi tipi di utente (definiti in 6.1) l'accesso alle sole funzionalità dedicate in base al corrispettivo livello di utenza, evitando che un utente possa accedere a dati che non può visualizzare ed impedire che compia azioni a lui non concesse.

Infatti diverse rotte possono essere attivate con parametri passati nell' URL: questi in teoria possono essere modificati dall'utente per accedere a pagine e/o informazioni a cui normalmente non ha accesso. Il meccanismo dei Gates serve proprio ad evitare l'accesso non autorizzato perché prima dell'attivazione di una rotta o di un'azione del controller l'applicazione stessa, tramite il middleware, svolge dei controlli preventivi al fine di verificare se l'accesso a certe risorse è autorizzato oppure no.

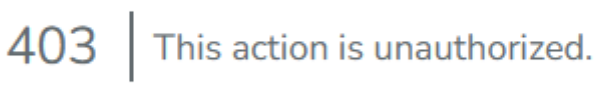

Figura 53 : messaggio mostrato dal browser se si tenta di effettuare azione non autorizzata

7.3.5 Password criptate

Per proteggere dati molto sensibili, come ad esempio le password degli utenti registrati, Laravel mette a disposizione la Facade “hash”. In questo modo se un utente con intenzioni malevole dovesse accedere al database dove sono contenuti i dati, quelli criptati sarebbero comunque protetti.

In questa web app la facade “hash” è stata appunto utilizzata per criptare le password degli utenti:
- all’interno della directory “database/seeds” per le password degli utenti inseriti tramite il seeding del database
- All’interno di “app/http/Controllers/CasaAutoController” nella funzione “storecliente()” quando la casa automobilistica , tramite apposita form, va a creare il profilo di un nuovo utente inserendo tutti i dati, password compresa

```
//serve per registrare un nuovo cliente
public function storecliente (NewClienteRequest $request){
      $cliente= new Users;  //con la s sta sotto package mod
      $cliente->fill($request->validated());  //serve per la
     $appoggio1=$cliente['password'];        // devo utilizz
      $appoggio2=Hash::make($appoggio1);    // password affi
      $cliente['password']=$appoggio2;        // per il nuov
      $cliente->save();  //salva dati nel db
```

Figura 54 : porzione della funzione “storecliente()”

7.3.6 Sanificazione di variabili

Laravel permette di "sanificare” i valori delle variabili PHP passate al template Blade applicando la funzione PHP ” htmlspecialchars” a fini di sicurezza: se la variabile passata contiene del malware (cioè del codice con intenzioni malevole) questa funzione è in grado di convertire i caratteri speciali di una stringa (quindi tutti quei caratteri che potenzialmente

identificano delle istruzioni di codice da eseguire) nei corrispondenti codici HTML, evitando quindi che eventuale codice malware possa essere eseguito e causare danni.[9], [53]
La variabile $prova per essere sanificata deve essere passata alla view con la seguente sintassi : {{$prova}}. Di seguito un esempio estrapolato proprio dal codice della web app.

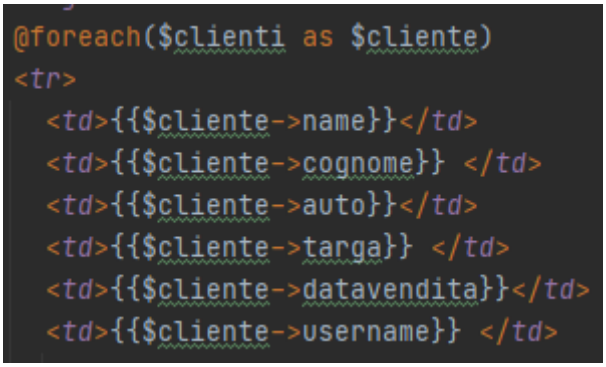

Figura 55 : dettaglio di una view dove la variabile $clienti, utilizzata all'interno di un foreach, viene utilizzata solo dopo essere stata sanificata

7.3.7 MySQL

MySQL è un sistema open source di gestione di database relazionali SQL sviluppato e supportato da Oracle, utilizzato per archiviare i dati necessari all'applicazione web sviluppata. Tramite "phpMyAdmin", un'applicazione disponibile con Xampp, è possibile gestire il database MySQL utilizzato per gestire i dati della web app. Di seguito si riporta una porzione dell'app con le tabelle create.

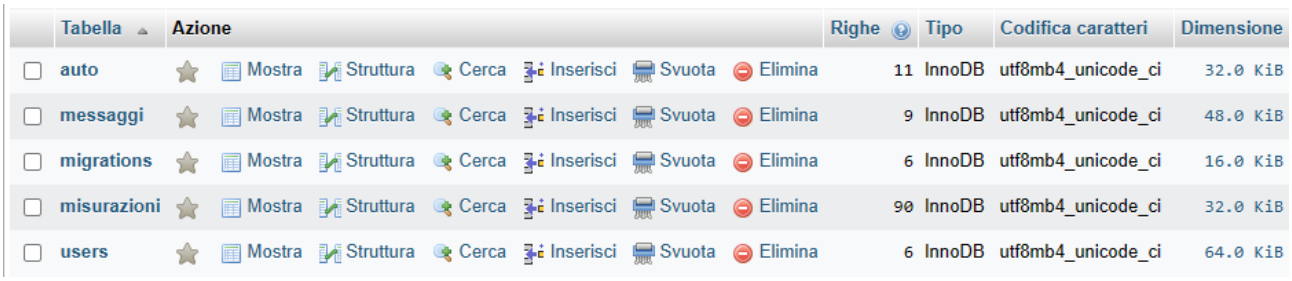

Figura 56 : porzione della schermata di phpMyAdmin con le tabelle del database

In particolare la tabella:

-auto: serve per gestire le auto della casa automobilistica, visibili anche dall'area pubblica

-messaggi: per gestire i messaggi che i clienti possono scambiarsi con la casa automobilistica

-misurazioni: per gestire i dati sulle varie misurazioni che i sensori effettuano sulle auto vendute

-users: per gestire gli utenti in base al loro livello

7.3.8 Web Server Apache

Apache HTTP Server, o più comunemente Apache, è il nome di un server web libero sviluppato dalla Apache Software Foundation. È la piattaforma server Web modulare più diffusa, in grado di operare su una grande varietà di sistemi operativi. È un software che realizza le funzioni di trasporto delle informazioni, di internet e di collegamento, ed ha il vantaggio di offrire funzioni di controllo per la sicurezza come quelle effettuate da un proxy.Il suo compito è stabilire una connessione tra un server e i browser dei visitatori del sito web (Firefox, Google Chrome, Safari, ecc.) mentre inviano file avanti e indietro tra di loro (struttura client-server). Quando un visitatore vuole caricare una pagina sul tuo sito web, il suo browser invia una richiesta al server e Apache restituisce una risposta con tutti i file richiesti (testo, immagini, ecc. .).Il server e il client comunicano tramite il protocollo HTTP e il server web Apache è responsabile della comunicazione fluida e sicura tra le due macchine.[54]

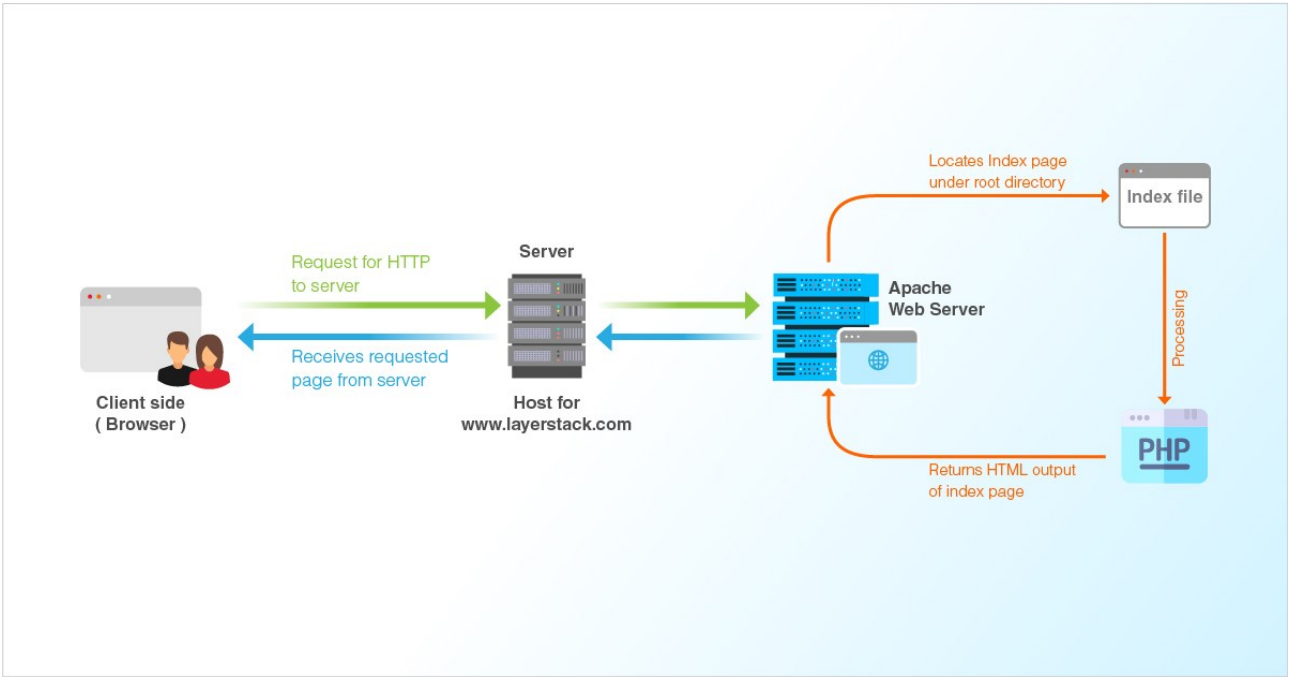

Figura 57 : Web server apache [54]

Apache è altamente personalizzabile grazie alla sua infrastruttura open source. Per questo motivo, gli sviluppatori web e gli utenti possono adattare il codice sorgente in base al tipo di sito web che stanno creando. Apache inoltre fornisce numerosi moduli che consentono agli amministratori del server di attivare e disattivare funzionalità aggiuntive. Il web server Apache dispone di moduli per la sicurezza, la memorizzazione nella cache, la riscrittura degli URL, l'autenticazione della password e altre funzionalità. Per configurare Apache si deve accedere al file ".htaccess".