UNIVERSITÀ DEGLI STUDI ROMA TRE

Dipartimento di Ingegneria Informatica
Corso di Laurea in Ingegneria Informatica

Progettazione e sviluppo di un'interfaccia web in Laravel e Vue.js per il monitoraggio interattivo e la gestione di stazioni di ricarica per veicoli elettrici

Candidato

Ted Jevel Quilenderino Bonga

Matricola 546569

Relatore

Prof. Paolo Merialdo

Relatrice esterna

Prof.sa Silvia Canale

Anno Accademico 2023-2024

Questa è la dedica

Ringraziamenti

Grazie a tutti

Terminologia e definizioni

EV	Electric Vehicle
EVSE	Electric Vehicle Supply Equipment
Charge Point	Punto di ricarica
Charging Station	Stazione di ricarica
OCPP	OCPP (Open Charge Point Protocol) è uno protocollo standard aperto di comunicazione per la gestione delle stazioni di ricarica dei veicoli elettrici

Indice

Progettazione e sviluppo di un'interfaccia web in Laravel e Vue.js per il monitoraggio interattivo e la gestione di stazioni di ricarica per veicoli elettrici

Ted Jevel Quilenderino Bonga

Ottobre 2024

Capitolo 1

Introduzione

Il tema di questa tesi riguarda lo sviluppo e aggiornamento di un applicazione Web per la gestione di una stazione di ricarica per veicoli elettrici (EV). In particolare, tale applicazione (alla quale d'ora in avanti si farà riferimento con "Portale" presenta due scopi principali:

- Monitoraggio in tempo reale delle stazioni di ricarica: Il Portale permette di monitorare lo stato delle stazioni di ricarica in tempo reale, visualizzando informazioni dettagliate su ogni punto di ricarica, come la disponibilità (Available), la preparazione (Preparing), la ricarica in corso (Charging), eventuali errori (Faulted), e altri stati operativi definiti dal protocollo OCPP. Questo consente agli operatori di avere una visione completa e aggiornata della situazione, migliorando l'efficienza nella gestione delle risorse e nel supporto agli utenti.
- Aggiornamenti reattivi senza necessità di refresh della pagina web:
  - Una delle principali caratteristiche del Portale è la sua reattività. Ogni volta che un Charge Point cambia stato, questa modifica viene immediatamente riflessa nel portale senza dover ricaricare la pagina. Questa funzionalità è resa possibile grazie all'utilizzo di WebSocket, che permettono di ricevere aggiornamenti in tempo reale e migliorano l'esperienza utente, rendendo l'applicazione più dinamica e interattiva.

Capitolo 2

Tecnologie utilizzate

Strumenti Utilizzati

In questo capitolo verranno analizzati gli strumenti principali utilizzati per lo sviluppo del Portale web, con particolare attenzione ai framework Laravel 8 e Vue.js.

2.0.1 Laravel 8

Laravel 8 è un framework PHP moderno ed efficiente per lo sviluppo di applicazioni web. La sua architettura MVC (Model-View-Controller) consente una gestione strutturata e scalabile del codice, facilitando la separazione tra logica di business e presentazione dell'interfaccia utente. Grazie a un ampio ecosistema di pacchetti e una community attiva, Laravel offre numerose funzionalità integrate, tra cui gestione delle rotte, autenticazione, accesso ai database e molto altro, permettendo di accelerare il processo di sviluppo.

2.0.2 Vue.js

Vue.js è un framework JavaScript progressivo per la costruzione di interfacce utente interattive e reattive. Grazie al binding bidirezionale dei dati (two-way data binding) e alla gestione del DOM virtuale, Vue.js è particolarmente adatto per lo sviluppo di single-page applications (SPA). Integrabile con facilità in progetti esistenti e altamente

modulare, Vue.js supporta componenti riutilizzabili e reattività avanzata, rendendolo ideale per applicazioni web dinamiche e scalabili.

2.0.3 Stazione di Ricarica Alfen

Le stazioni di ricarica Alfen sono soluzioni avanzate per la ricarica di veicoli elettrici, note per la loro affidabilità e versatilità. Queste stazioni offrono funzionalità di monitoraggio in tempo reale e sono dotate di sistemi di sicurezza integrati per garantire la protezione durante la ricarica. Grazie alla compatibilità con piattaforme di gestione come quelle basate su Laravel e Vue.js, le stazioni Alfen consentono una gestione semplificata e personalizzata, rendendole ideali per infrastrutture di ricarica scalabili e efficienti.

Figura 2.1: Colonnina Alfen

2.0.4 Pusher

Pusher è una piattaforma di messaggistica in tempo reale che semplifica l'implementazione di WebSocket nelle applicazioni web. Consente agli sviluppatori di creare facilmente funzionalità interattive, come chat in tempo reale e aggiornamenti dinamici, senza la necessità di gestire direttamente le complessità associate alla gestione delle connessioni WebSocket. Pusher offre un'interfaccia API intuitiva e scalabile, facilitando la comunicazione bidirezionale tra client e server. Nel contesto del nostro progetto, Pusher è stato utilizzato per implementare aggiornamenti in tempo reale degli stati delle charge-point, migliorando così l'esperienza utente e garantendo che le informazioni siano sempre aggiornate senza necessità di ricaricare la pagina. [**?**], [**?**], [**?**], etc.

Capitolo 3

Architettura di riferimento

Nel contesto dello sviluppo di applicazioni web moderne, la necessità di interazioni in tempo reale è diventata sempre più cruciale. In questo capitolo, verrà esplorata la tecnologia WebSocket, che consente una comunicazione bidirezionale tra client e server. In particolare, ci concentreremo su come questa tecnologia possa essere utilizzata per rendere dinamico il cambiamento di stati delle charge-point in un'applicazione Vue.js. Attraverso un'analisi dettagliata delle classi e delle strutture implementate, dimostreremo come la tecnologia WebSocket possa migliorare l'esperienza utente, permettendo aggiornamenti immediati dello stato delle stazioni di ricarica senza necessità di ricaricare la pagina. Questo approccio non solo ottimizza le prestazioni dell'applicazione, ma contribuisce anche a una gestione più efficiente delle risorse in un contesto di Smart City.

3.0.1 WebSocket

Un task iniziale fu quello di studiare come utilizzare la tecnologia Websocket per rendere dinamico il cambiamento degli stati di una charge-point. Questo è cruciale per migliorare l'esperienza utente, poiché consente di visualizzare aggiornamenti in tempo reale senza la necessità di ricaricare la pagina. Ad esempio, gli stati di una charge-point possono variare da "available" a "preparing", "charging", "finishing", e nuovamente a "available".

Implementando un server WebSocket, è possibile stabilire una connessione persi-

stente tra il client e il server. Ciò consente al server di inviare messaggi al client non appena vi sono cambiamenti nello stato delle charge-point. In Vue.js, ciò si traduce in una reattività immediata dell'interfaccia utente, migliorando notevolmente l'interazione e l'usabilità dell'applicazione. Utilizzando le funzionalità di Vue.js come il binding dei dati, si possono aggiornare dinamicamente le informazioni visualizzate senza necessità di aggiornamenti manuali della pagina. Questo approccio non solo ottimizza la performance dell'app, ma rende anche l'esperienza utente più fluida e intuitiva.

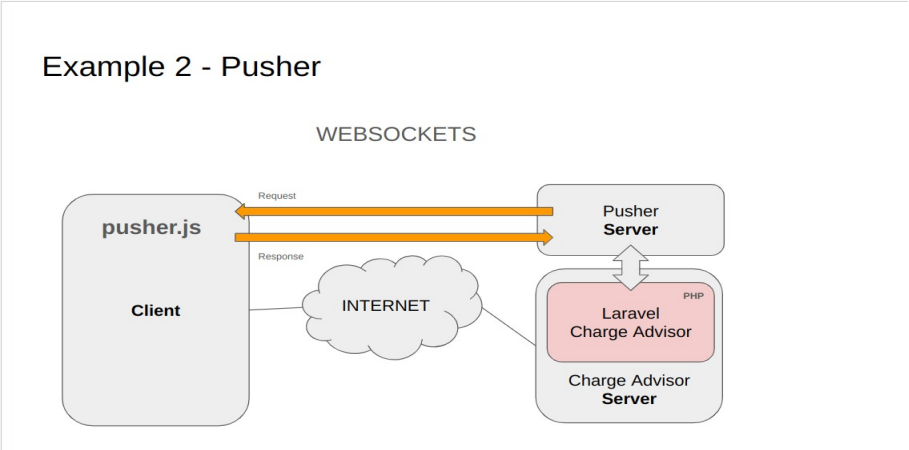

Figura 3.1: Architettura WebSocket con Pusher

```
pusher.js
25  //Importing Echo and Pusher
26  import Echo from 'laravel-echo';
27  window.Pusher = require('pusher-js');
28
29  //Initializing Laravel Echo with Pusher
30  window.Echo = new Echo( options: {
31      broadcaster: 'pusher',
32      key: process.env.MIX_PUSHER_APP_KEY,
33      cluster: process.env.MIX_PUSHER_APP_CLUSTER,
34      forceTLS: true,
35      wsHost: window.location.hostname,
36      wsPort: 4500,
37      disableStats: true,
38  });
```

Figura 3.2: Collegamento al nostro backend al server di Pusher per utilizzare la tecnologia Websocket

3.0.2 Confronto tra WebSocket, Polling System e Pushing System

Nel contesto delle applicazioni web in tempo reale, esistono diversi approcci per la comunicazione tra server e client. Tra questi, i principali sono:

- **Polling System**: il client invia richieste periodiche al server per verificare la presenza di nuovi dati
- **Pushing System (WebSocket**: il server può inviare aggiornamenti in tempo reale al client non appena i dati cambiano.

3.0.2.1 Polling System

Il *Polling* è un metodo tradizionale in cui il client esegue richieste HTTP al server a intervalli regolari (per esempio ogni dieci secondi) per ottenere aggiornamenti. Sebbene sia *semplice* da implementare, presenta diversi svantaggi:

- Aumento del carico sul server, poiché ogni richiesta viene trattata come una nuova connessione
- Ritardo negli aggiornamenti, poiché i dati vengono aggiornati solo a intervalli predefiniti
- Inefficienza nel consumo di banda, in quanto molte richieste potrebbero non restituire dati aggiornati

3.0.3 Websocket

Con l'uso di Websocket, si stabilisce una connessione persistente tra client e server, consentendo un flusso di dati bidirezionale senza la necessità di nuove richieste HTTP. Questo approccio offre diverse vantaggi rispetto al *Polling*:

- **Aggiornamento in tempo reale**: il server può notificare immediatamente i client sui cambiamenti di stato
- **Riduzione del carico sul server**: una singola connessione rimane aperta, riducendo l'overhead delle richieste HTTP ripetute.

- **Minore latenza**: il tempo di risposta tra aggiornamenti è praticamente immediato.

Nel caso della gestione delle charge-point, l'utilizzo di WebSocket con Pusher permette di aggiornare in tempo reale lo stato delle stazioni, garantendo una sincronizzazione ottimale tra client e server.

Confronto tra i metodi

La seguente tabella riassume le principali differenze tra questi approcci:

Metodo	Comunicazione	Efficienza	Latenza	Complessità
Polling	Unidirezionale	Bassa	Alta	Bassa
WebSocket	Bidirezionale	Alta	Bassa	Media

Tabella 3.1: Confronto tra Polling, WebSocket e SSE

L'adozione di Websocket si rivela dunque la scelta più scelta più efficace per applicazioni che necessitano di aggiornamenti in tempo reale, come il monitoraggio dinamico delle charge-point.

Capitolo 4

Analisi dei Requisiti

Il portale prevede l'interazione con un insieme ben definito di attori. In particolare, il portale web è progettato esclusivamente per l'uso da parte degli operatori incaricati della gestione e del monitoraggio delle stazioni di ricarica per veicoli elettrici. I clienti finali, ovvero gli automobilisti che desiderano ricaricare il proprio veicolo, non hanno accesso diretto al portale e utilizzano invece altre interfacce o strumenti per l'attivazione e il pagamento delle sessioni di ricarica (eg. HMI: Human Machine Interface).

4.0.1 Tipi di utente

Gli operatori del sistema possono essere suddivisi in diverse categorie, ciascuna con permessi e funzionalità specifiche:

- **Amministratori**: hanno pieno accesso al sistema, possono gestire utenti, configurare le stazioni di ricarica e monitorarie l'attività complessiva.
- **Operatori tecnici**: si occupano del monitoriraggio delle stazioni di ricarica, intervenendo in caso di guasti o anomalie

Capitolo 5

Progettazione dell'Interfaccia Utente

In questo capitolo tratterò sull'aspetto concettuale e visivo dell'interfaccia

Capitolo 6

Implementazione della Soluzione

Screen di porzione del codice e commento con foto all'interfaccia Web

Capitolo 7

Conclusioni e sviluppi futuri

Conclusioni finali e futuro sviluppo del portale, sviluppare una conessione websocket senza utilizzare un applicazione di terze parti (Pusher) e parlare per esempio di estendere l'utilizzo del Portale non solo agli operatori ma anche ai clienti finali.. @thesis•, author = •, title = •, type = •, institution = •, year = •, OPTsubtitle = •, OPTtitleaddon = •, OPTlanguage = •, OPTnote = •, OPTlocation = •, OPTmonth = •, OPTisbn = •, OPTchapter = •, OPTpages = •, OPTpagetotal = •, OPTaddendum = •, OPTpubstate = •, OPTdoi = •, OPTeprint = •, OPTeprintclass = •, OPTeprinttype = •, OPTurl = •, OPTurldate = •, [?]