

Relazione di Applicazioni Web

Il progetto è relativo ad un'app web che gestisce parcheggi che implementano una ricarica elettrica tramite dei robot chiamati MwBot utilizzando **ASP.NET Core** su **.NET Core 8** e **Entity Framework** e il **protocollo MQTT**.

Abbiamo utilizzato Razor per le view, è incentrato su pagine, dove ciascuna pagina rappresenta una singola unità di funzionalità e gestisce sia la logica di visualizzazione che quella di elaborazione.

Come database è stato utilizzato **Sqlite** con **Entity Framework Core**, le query sono eseguite tramite **LinQ**; abbiamo utilizzato il **repository pattern**.

Avvio

E' disponibile l'autenticazione google: bisogna inserire i secrets dell'utente ClientId e ClientSecret generati dalla console google sul Progetto.App. In caso non si posseggano, per avviare l'applicazione senza problemi, commentare riga 32 → 36 del file Program.cs del Progetto.App (è correttamente implementato ma lo lasciamo commentato per permettere test anche senza secrets)

Per testare applicazione web bisogna avviare Progetto.App e CamSimulator, che, nel contesto dell'applicazione rappresenta il sensore che legge la targa all'entrata / uscita di un singolo parcheggio.

Scegliere eventualmente il livello di log desiderato tramite l'appsettings.json sotto Progetto.App.

Eventualmente eseguire il comando update-database per aggiornare il database in base alle migration aggiunte.

Spiegazione database

_EFMigrationsHistory: rappresenta le migration

Tabelle AspNetUser: vengono utilizzate da IdentityUser per eseguire il login e i Claim per gestire i permessi degli utenti (ruoli indicati sotto)

Cars: rappresenta le entità delle macchine munite di targa, stato...

CurrentlyCharging: rappresenta le ricariche in corso / da pagare che stanno venendo gestite dagli MwBot. Quelle in corso da quelle terminate vengono differenziate da un flag ToPay, per venire poi storicizzate a pagamento completato.

ImmediateRequests: rappresenta le richieste di ricarica gestite dagli MwBot generate previa Reservation (prenotazione dell'utente premium) oppure quando l'utente sceglie di eseguire una ricarica dopo che la sua auto è stata rilevata in entrata nel parcheggio dal sensore.

MwBots: rappresenta le entità dell'mwbot, i loro stati...

Parkings: rappresenta i vari parcheggi gestiti dal sito.

ParkingSlots: rappresenta i singoli posti auto all'interno di un parcheggio

PaymentHistory: rappresenta lo storico dei pagamenti effettuati derivanti da sosta (Stopover) o ricarica (CurrentlyCharging).

Reservations: rappresenta le prenotazioni di ricarica fatte dagli utenti premium.

Stopover: rappresenta la sosta richiesta quando l'utente seleziona "sosta" in seguito alla macchina rilevata in entrata nel parcheggio.

Spiegazione Progetto

Paypal.REST presente ma non gestito, verrà implementato in futuro per PISSIR

CamSimulator è un'applicazione di tipo Windows Form che rappresenta la telecamera con rilevamento targa presente all'entrata di uno specifico parcheggio (da selezionare).

Rileva targhe in entrata / uscita: in seguito all'entrata, viene richiesto all'utente sulla pagina *Dashboard / Servizi* se effettuare una sosta o una ricarica.

Si interfaccia con la applicazione web tramite Post e Get a due endpoint ApiRest presenti in un controller dedicato.

Progetto.App è la nostra web app nonché root del progetto, del sito, con al suo interno configurazione all'avvio, log, controller con i vari endpoint con cui interfacciarsi per le richieste e le pagine Front-End di interfaccia.

Progetto.App Core (domain + application) rappresenta tutta la configurazione del database, i modelli utilizzati dall'app, le relative configurazioni del database, le migration, i validator dei modelli, le repository per interfacciarsi con il database e i servizi, in particolare MQTT.

3 tipi di utente :

- Utente base
- Utente premium
- Utente admin

Utente base può visionare i parcheggi disponibili, il loro stato d'occupazione ed effettuare ricariche e/o soste scegliendo dal menù "*Dashboard > Services*" se la loro auto viene rilevata dalla telecamera in entrata.

Utente premium, rispetto all'utente base, può scegliere prenotare una ricarica tramite il menù "*Dashboard > Reservation*".

Admin ha il controllo completo sui parcheggi; può quindi effettuare alcune operazioni CRUD non-critical tramite i vari menù, accendere / spegnere i robot, filtrare i pagamenti effettuati.

Spiegazione MQTT

All'avvio, l'applicazione web verifica gli MwBot che sono attualmente online e li istanzia come client, connettendoli al broker ed effettuando l'inizializzazione dei parametri.



Broker MQTT responsabile della ricezione e distribuzione dei messaggi tra gli MwBot e la loro gestione delle richieste, quali ad esempio: aggiornamento dello stato, gestione operazioni ricarica, interrogazioni a database, fornitura dati necessari.



Client MQTT è la singola istanza di un MwBot online, che, comunicando con il broker gestisce le richieste di ricarica (tramite la ChargeManager) e altre operazioni. Si iscrive ai topic pertinenti su cui ricevere messaggi in risposta dal broker.


Gli MwBot inviano periodicamente messaggi di stato e aggiornamenti (definiti nella classe `MqttClientMessage`). In base al tipo di messaggio (`MessageType`), gestisce richieste di ricarica: quando rileva una batteria scarica o riceve una nuova richiesta di ricarica, invia un messaggio al broker che risponde, una volta gestita la logica back-end, con i dettagli necessari per proseguire nell'operazione.

La ricarica è simulata in tutti i suoi aspetti: caricamento batteria macchina, scarica batteria mwbot mentre carichi, spostamento dock di ricarica - parcheggio e viceversa, ricarica mwbot.

E' implementato un meccanismo di riconnessione e recupero dello stato: se il client si disconnette, tenta di riconnettersi, riprendendo eventualmente l'operazione interrotta.



 **admin@admin.it** 



Password
Admin1234\$  


Sito
<https://localhost:7237/Identity/Account/Login> 

Modifica

Elimina

 **admin01@admin.it** 



Password
J4JTdN:3Zm5G_3_  



Sito
<https://localhost:7237/Identity/Account/Register> 


Note
Nessuna nota aggiunta

Modifica

Elimina

 **utente01@utente.com** 



Password
bHw2Hs3!4bzbq-aQ  



Sito
<https://localhost:7237/Identity/Account/Register> 


Note
Nessuna nota aggiunta

Modifica

Elimina

 **utentepremium01@upremium.com** 

Password
SgRE5GZp_rj6s2E  

Sito
<https://localhost:7237/Identity/Account/Register> 

Note
Nessuna nota aggiunta

Modifica

Elimina