

Know Your Garage

Applicazioni e Servizi Web

Malucelli Nicolò - 0001085006 {nicolo.malucelli@studio.unibo.it}

25 Gennaio 2024

Indice

1	Introduzione	2
2	Requisiti	2
	2.1 Requisiti funzionali	2
	2.2 Requisiti non funzionali	4
3	Design	5
	3.1 Design architetturale	5
	3.2 Design delle interfacce utente	5
	3.3 Storyboard	7
4	Tecnologie	15
	4.1 Client	15
	4.2 Server	16
	4.3 Comunicazione	17
5	Codice	17
	5.1 Notifica eventi real-time	17
	5.2 Aggiornamento dei parcheggi mostrati	18
6	Test	18
7	Deployment	20
	7.1 Download e preparazione	20
	7.2 Inizializzazione del database con i dati di test	20
	7.3 Esecuzione	21
	7.4 Utilizzo	21
8	Conclusioni	21

1 Introduzione

L'idea di un applicativo web per la gestione di parcheggi multipiano nasce con l'obiettivo di aiutare gli utenti nella ricerca di un posto auto per brevi o lunghe soste, e al contempo di facilitare i privati nella gestione e nella promozione delle proprie aree di sosta, soprattutto nel contesto dei grandi centri urbani.

Questo progetto, mirato a migliorare l'esperienza di gestori e utenti, si propone come una soluzione completa per la supervisione, la registrazione e l'accesso alle informazioni cruciali relative ai parcheggi.

L'applicativo web offre un ampio spettro di funzionalità progettate per soddisfare le esigenze specifiche dei gestori di parcheggi e degli utenti finali.

Tra le caratteristiche principali, i gestori possono gestire con facilità le informazioni dei parcheggi, come ad esempio capacità effettiva e tariffe; visualizzare statistiche dettagliate, come numero di soste in un dato giorno o abbonamenti venduti; e al tempo stesso monitorare in tempo reale l'occupazione di ciascun parcheggio.

Al contrario, gli utenti possono beneficiare di una mappa interattiva che mostra in tempo reale l'occupazione dei parcheggi in una determinata zona ed offre loro la possibilità di acquistare gli abbonamenti messi a disposizione dai gestori dei parcheggi. L'applicativo consente inoltre agli utenti di registrare le proprie auto e di accedere a uno storico dettagliato delle soste, nonché di visualizzare lo stato e lo storico dei propri abbonamenti.

All'interno di questo report saranno esplorate in dettaglio le diverse funzionalità offerte dall'applicativo, andando ad analizzare la sua usabilità, praticità ed efficacia nella gestione dei parcheggi multipiano.

2 Requisiti

2.1 Requisiti funzionali

Requisiti utente

- Utente finale:

- L'utente finale in possesso di un account può effettuare l'accesso utilizzando le proprie credenziali utente tramite una sezione di login.
- L'utente finale non ancora in possesso di un account può creare il proprio profilo utente tramite una sezione di registrazione.
- L'utente finale può visualizzare le informazioni relative al proprio account in una schermata profilo e modificare la propria password di accesso in un qualunque momento successivo al login.
- L'utente finale può effettuare la disconessione dal proprio account in un qualunque momento.
- L'utente finale può visualizzare la lista delle proprie auto registrate e consultare in tempo reale il loro stato (parcheggiata o non) ed il numero di abbonamenti attivi associati alle auto.

- L’utente può registrare una nuova auto nel sistema, specificando modello targa ed altri attributi.
- L’utente può rimuovere dal sistema una o più auto in precedenza registrate previa conferma di eliminazione.
- L’utente può visualizzare attraverso una mappa i parcheggi presenti in una determinata località ed i relativi abbonamenti messi a disposizione dai gestori. I parcheggi sono rappresentati da speciali marker posizionati sulla mappa stessa che permettono all’utente di conoscere in tempo reale l’occupazione corrente del parcheggio.
- L’utente può aggiornare la località mostrata dalla mappa interagendo con la mappa stessa ma anche attraverso l’uso di una barra di ricerca che permetta di inserire il nome di una nuova località da mostrare.
- L’utente può acquistare un abbonamento per un determinato parcheggio, qualora i posti disponibili siano superiori a zero, indicando la tipologia di abbonamento tra quelle messe a disposizione dal gestore del parcheggio e l’auto a cui associare l’abbonamento. Selezionati auto ed abbonamento, l’utente può completare l’acquisto inserendo i dati della propria carta.
- L’utente può visualizzare, per ciascuna delle proprie auto, lo storico degli abbonamenti posseduti e le relative informazioni, come date di inizio e di fine validità e stato dell’abbonamento (attivo o scaduto).
- L’utente può visualizzare, per ciascuna delle proprie auto, lo storico delle soste effettuate. Per ogni sosta, l’utente deve poter visualizzare la data e l’ora di inizio della sosta, nonché la durata effettiva della sosta.

- Gestore di parcheggi

- Il gestore di parcheggi in possesso di un account può effettuare l’accesso utilizzando le proprie credenziali di gestore tramite una sezione di login.
- Il gestore di parcheggi può visualizzare le informazioni relative al proprio account in una schermata profilo e modificare la propria password di accesso in un qualunque momento successivo al login.
- Il gestore di parcheggi può effettuare la disconnessione dal proprio account in un qualunque momento.
- Il gestore di parcheggi può visualizzare la lista dei propri parcheggi registrati, visualizzando per ciascuno di essi le relative informazioni, tra le quali il numero di posti occupati in tempo reale.
- Il gestore di parcheggi può modificare alcune informazioni relative ai propri parcheggi, come località e numero di posti auto disponibili
- Il gestore di parcheggi può registrare un nuovo parcheggio nel sistema, specificandone località e numero di posti.

- Il gestore di parcheggi può rimuovere dal sistema uno o più parcheggi in precedenza registrati previa conferma di eliminazione.
- Il gestore di parcheggi può visualizzare, per ciascun parcheggio in proprio possesso, la lista degli abbonamenti disponibili e modificarne le relative informazioni, come durata e costo.
- Il gestore di parcheggi può registrare un nuovo abbonamento nel sistema per un determinato parcheggio.
- Il gestore di parcheggi può rimuovere dal sistema uno o più abbonamenti in precedenza registrati previa conferma di eliminazione.
- Il gestore di parcheggi può visualizzare le statistiche di ciascun parcheggio in proprio possesso, come: numero di soste effettuate in un dato giorno, numero di abbonamenti venduti per ciascun tipo, numero di nuovi clienti nell'ultimo periodo di tempo
- Il gestore di parcheggi può visualizzare in tempo reale lo stato di occupazione di ciascun parcheggio in proprio possesso

Requisiti di sistema

- Si rende necessaria la distinzione tra le due diverse categorie di profili (utente finale e gestore di parcheggi).
- In seguito all'eliminazione di un parcheggio da parte di un gestore di parcheggi, gli utenti finali non devono essere più in grado di visualizzare il parcheggio sulla mappa interattiva ma ciò non deve affliggere la visualizzazione dello storico delle soste. Il parcheggio dovrà infatti continuare ad essere visibile anche se eliminato nella sezione dello storico.
- Lo stesso si applica per la gestione degli abbonamenti: un abbonamento eliminato non deve essere più acquistabile dagli utenti finali ma deve continuare ad essere visibile nella sezione abbonamenti acquistati qualora l'utente finale abbia acquistato un abbonamento di tale tipologia.
- Le password di utenti finali e gestori di parcheggi non devono essere salvate in chiaro all'interno della base di dati.

2.2 Requisiti non funzionali

- Rapida elaborazione delle richieste di gestori e utenti senza situazioni di blocco dovute a lunghi caricamenti.
- L'interfaccia utente deve essere intuitiva e facile da navigare, sia per i gestori di parcheggi che per gli utenti finali.
- Il sistema deve essere in grado di gestire un numero crescente di parcheggi, gestori e utenti senza che ciò comprometta in alcun modo le prestazioni.
- Il sistema dovrà risultare facilmente estendibile in vista di aggiornamenti futuri

3 Design

3.1 Design architetturale

Durante tutta la fase di progettazione si è scelto di adottare il principio KISS (Keep It Simple, Stupid), con l'obiettivo di semplificare il sistema e renderlo il più comprensibile ed intuitivo possibile. L'approccio KISS è stato adottato non soltanto per la parte grafica, ma anche per la progettazione dell'architettura del sistema, consentendo di ottenere un elevato grado di manutenibilità e di facilità nell'identificazione e nella soluzione degli errori.

Per quanto riguarda l'architettura del sistema, si è scelto di adottare lo stack MEVN, con la conseguente adozione delle seguenti tecnologie:

- MongoDB
- Express
- VueJs
- Node.js

L'architettura del sistema è strutturata in due parti fondamentali: frontend e backend. Il frontend è stato sviluppato utilizzando VueJs, consentendo così di ottenere un'interfaccia utente interattiva e dinamica. Il backend è invece basato su Node.js ed Express, ed utilizza MongoDB come database per garantire la persistenza delle informazioni.

La comunicazione tra queste due componenti avviene tramite l'utilizzo di REST API. Questo approccio favorisce la modularità del sistema, consentendo una migliore organizzazione delle risorse ed una netta separazione tra le risorse di diverso tipo.

Mentre le REST API sono eccellenti per quanto riguarda l'accesso asincrono alle risorse del sistema e risultano ottimali per gran parte delle interazioni tra frontend e backend, esse non sono sufficienti per soddisfare in maniera efficace tutti i requisiti dell'applicazione sopra descritti. Dai requisiti si evince infatti la necessità di inserire all'interno del sistema componenti per la comunicazione real-time e, sebbene sia possibile implementare questa comunicazione anche attraverso REST API, utilizzando ad esempio strategie come il polling, si è scelto di adottare una soluzione più efficace basata sui socket.

3.2 Design delle interfacce utente

Durante tutto il processo di progettazione dell'applicativo, si è seguito un approccio di tipo User-Centered, ponendo l'utente al centro di ogni scelta, con l'obiettivo di ottenere un'esperienza utente ottimale.

Per comprendere al meglio le esigenze degli utenti, sono state individuate una serie di personas, le quali hanno guidato il processo di progettazione dell'interfaccia utente, ottenendo così un design di tipo User-Centered.

Nello specifico, sono state prese in considerazione entrambe le categorie di utenti presenti all'interno del dominio applicativo: gli utenti finali ed i gestori di parcheggi e sono quindi state individuate tre differenti personas con annessi scenarios:

- Claudia

Claudia non possiede ancora un account all'interno del sistema, ma nei prossimi giorni visiterà con la propria famiglia la città di Cesena in una breve gita fuori porta e vorrebbe utilizzare l'applicativo per cercare un parcheggio che abbia posti disponibili

1. Claudia visita il sito e procede effettuando la registrazione del suo nuovo account
2. Una volta registrata, Claudia procede subito con la visualizzazione della mappa e nota, tra i tanti parcheggi, uno avente ancora alcuni posti disponibili
3. Claudia non procede acquistando alcun abbonamento, poiché la sua sosta non durerà più di alcune ore, tuttavia l'applicativo le è risultato utile nella ricerca rapida di un posto auto in una città a lei sconosciuta

- Francesco

Francesco è un utente navigato che possiede già un account all'interno del sistema. Di recente, Francesco ha venduto la propria vecchia auto e ne ha comprata una nuova, per questo motivo egli vuole aggiornare le informazioni presenti nel proprio account ed acquistare un abbonamento per la sua nuova vettura.

1. Francesco effettua l'accesso utilizzando le proprie credenziali già esistenti
2. Una volta effettuato l'accesso, Francesco si dirige nell'apposita sezione dedicata alle auto e rimuove l'auto che ha appena venduto dalla lista delle auto di sua proprietà
3. Dalla stessa pagina, Francesco procede poi con la registrazione della sua nuova auto, inserendo tutti i dati che il sito gli richiede
4. Francesco completa con successo il processo di registrazione della vettura ed è ora in grado di visualizzare l'auto appena registrata tra quelle di sua proprietà
5. Francesco si dirige ora nella sezione che mostra i parcheggi presenti nella propria zona e, una volta selezionato il parcheggio desiderato, procede con l'acquisto di un abbonamento annuale per la propria nuova auto
6. Una volta completato il processo di acquisto, Francesco è notificato a schermo della riuscita del pagamento

- Francesco si dirige ora nella sezione abbonamenti, dove è in grado di visualizzare l'abbonamento appena acquistato
- Comune di Cesena

Il Comune di Cesena ha appena terminato la costruzione di un nuovo parcheggio multipiano e vuole quindi procedere aggiornando il proprio account, inserendo, oltre al nuovo parcheggio, anche una serie di abbonamenti.

- Il dipendente del Comune di Cesena che si occupa della gestione dei trasporti, effettua il login all'interno dell'applicativo attraverso il portale amministratori, utilizzando le credenziali in possesso
- Effettuato l'accesso, il dipendente si reca nella sezione parcheggi e procede con la registrazione di un nuovo parcheggio
- Terminata la creazione del parcheggio, il dipendente si dirige nella sezione abbonamenti e procede, dopo aver selezionato il parcheggio appena creato, con l'inserimento degli abbonamenti giornaliero, settimanale e mensile.

3.3 Storyboard

L'idea alla base di Know Your Garage è quella di fornire un'esperienza utente personalizzata in base al profilo che si sta utilizzando; pertanto, il primo passo indispensabile per fruire del sito è effettuare l'accesso.



Figura 1: Schermata di login

Qual'ora l'utente non possiede un profilo, può procedere con la creazione di uno nuovo cliccando il tasto registrati posto al di sotto del pulsante login.

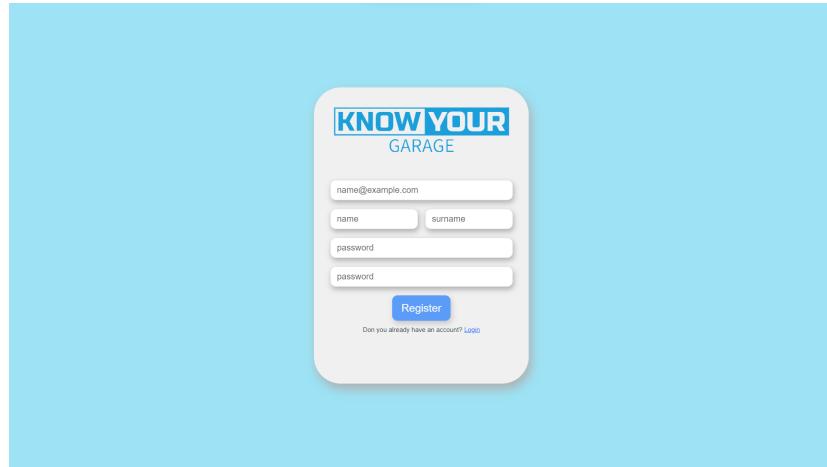


Figura 2: Schermata di registrazione

Una volta effettuato l'accesso, l'utente è indirizzato alla schermata principale, contenente una mappa dell'area circostante nella quale sono visualizzati i parcheggi presenti nelle vicinanze. Gli stessi parcheggi presenti sulla mappa sono anche riportati in una lista laterale così da rendere più immediata la ricerca di un determinato parcheggio e la visualizzazione generale dei posti disponibili.

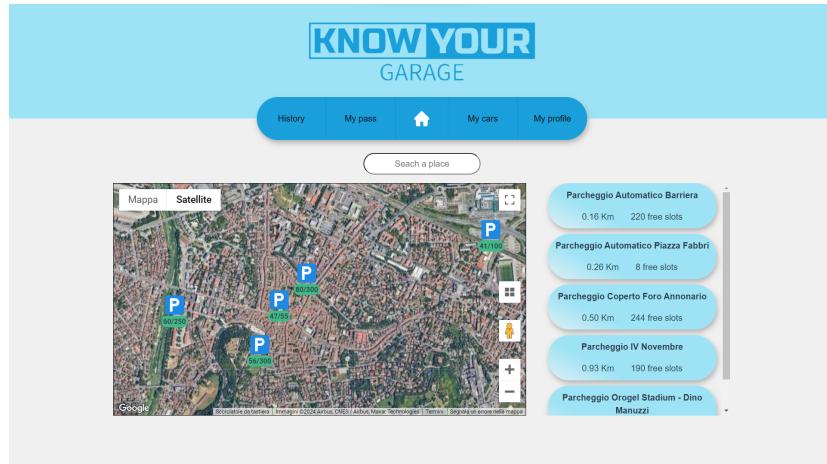


Figura 3: Home page

Cliccando uno dei marker presenti sulla mappa o un elemento dalla lista laterale, si aprirà una piccola finestra sopra al marker corrispondente che permetterà all'utente di acquistare un abbonamento per il parcheggio indicato (Fig.4).

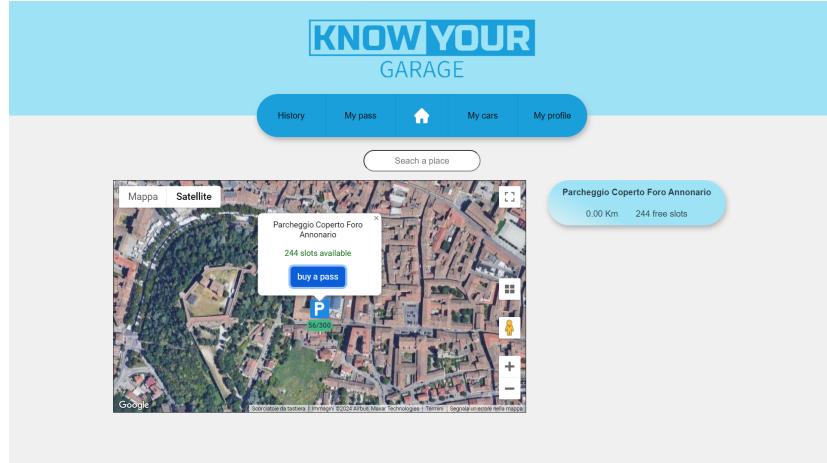


Figura 4: Home page - interazione con la mappa

cliccando sul pulsante ‘buy a pass’, si aprirà una finestra in overlay che consentirà all’utente di selezionare l’abbonamento (Fig.5a) e l’auto (Fig.5b) desiderate e di procedere con il pagamento (Fig.5c). L’utente può scorrere tra gli abbonamenti e le auto disponibili utilizzando le frecce a sinistra e a destra degli elementi selezionabili. Questa scelta di design, considerata semplice ed intuitiva, compare in molteplici occasioni in diverse pagine dell’applicativo.

L’utente può, in ogni momento, chiudere la finestra in overlay premendo la ‘x’ posta nell’angolo in alto a destra o cliccando in un qualsiasi punto al di fuori della finestra stessa.



Figura 5: Acquisto di un abbonamento

Cliccando sulla sezione ‘my pass’, l’utente può visualizzare tutti gli abbonamenti acquistati in passato per ogni auto da lui posseduta, visualizzando informazioni quali nome del parcheggio, tipo di abbonamento e data di inizio e

di fine di validità. Nell'ultima colonna della tabella è indicato invece lo stato dell'abbonamento: attivo, scaduto o non ancora attivo.

pass	garage	from	to	status
Abbonamento mensile	Parcheggio Automatico Barriera	01-01-2024	30-01-2024	active
Pass giornaliero	Parcheggio Coperto Foro Annuario	29-01-2024	29-01-2024	not active yet
Abbonamento mensile	Parcheggio Automatico Barriera	01-12-2023	30-12-2023	expired
Abbonamento mensile	Parcheggio Automatico Barriera	01-11-2023	30-11-2023	expired
Abbonamento mensile	Parcheggio Automatico Barriera	01-10-2023	30-10-2023	expired

Figura 6: Visualizzazione degli abbonamenti posseduti

Visitando invece la sezione ‘history’, l’utente può visualizzare lo storico delle soste effettuate da ciascuna delle proprie auto (Fig.7). In entrambe queste ultime due pagine, l’utente può scorrere l’auto visualizzata tra quelle registrate, cliccando sulle frecce poste a fianco dell’auto.

garage	day	start	end	duration
Parcheggio Automatico Barriera	25-01-2024	08:22	-	-
Parcheggio Automatico Barriera	24-01-2024	08:26	18:44	10h18
Parcheggio Automatico Barriera	23-01-2024	08:22	18:27	10h5
Parcheggio Automatico Barriera	22-01-2024	08:26	18:35	10h9
Parcheggio Automatico Barriera	21-01-2024	08:26	18:44	10h18

Figura 7: Visualizzazione dello storico delle soste

La lista delle auto registrate è invece visualizzabile visitando la sezione ‘my cars’. Da qui è possibile eliminare un’auto precedentemente registrata o registrare una nuova auto, cliccando sul pulsante ‘add a new car’.

Questa pagina consente inoltre di visualizzare lo stato corrente di un'auto, ovvero se parcheggiata o non parcheggiata, ed il numero di abbonamenti attualmente attivi per ogni auto registrata.

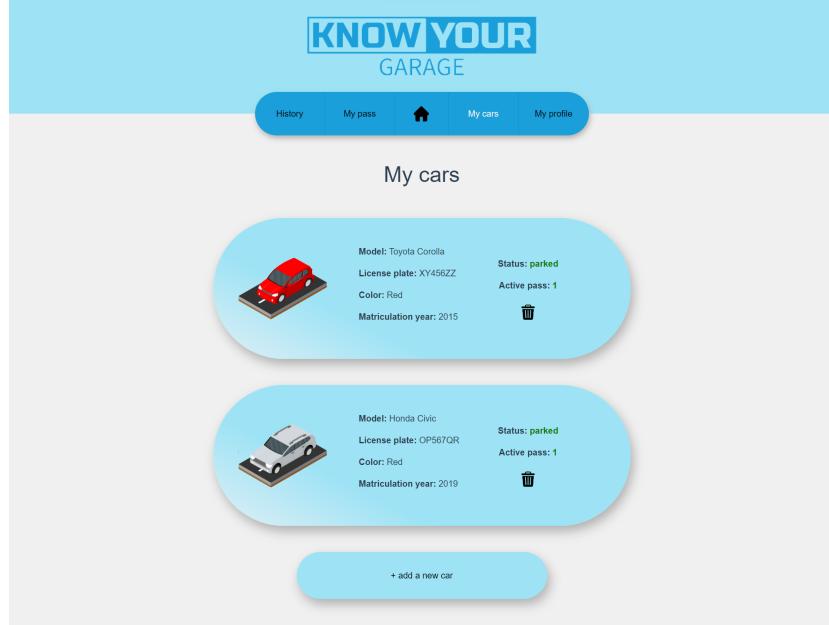


Figura 8: Visualizzazione delle auto registrate

Per effettuare il logout, è sufficiente visitare la pagina dedicata al profilo e cliccare sul tasto ‘logout’ (Fig.9a). Cliccando invece sul pulsante ‘change password’, la pagina si aggiornerà ed un form permetterà all’utente di inserire la propria nuova password (Fig.9b).



Figura 9: Profilo utente

Le informazioni visualizzate dal gestore di parcheggi in seguito all'accesso sono invece differenti. Una volta effettuato l'accesso, il gestore di parcheggi è indirizzato in una pagina dalla quale può visualizzare e modificare le informazioni riguardanti i propri parcheggi (Fig.10), oltre che a cancellare e creare nuovi parcheggi (Fig.11).

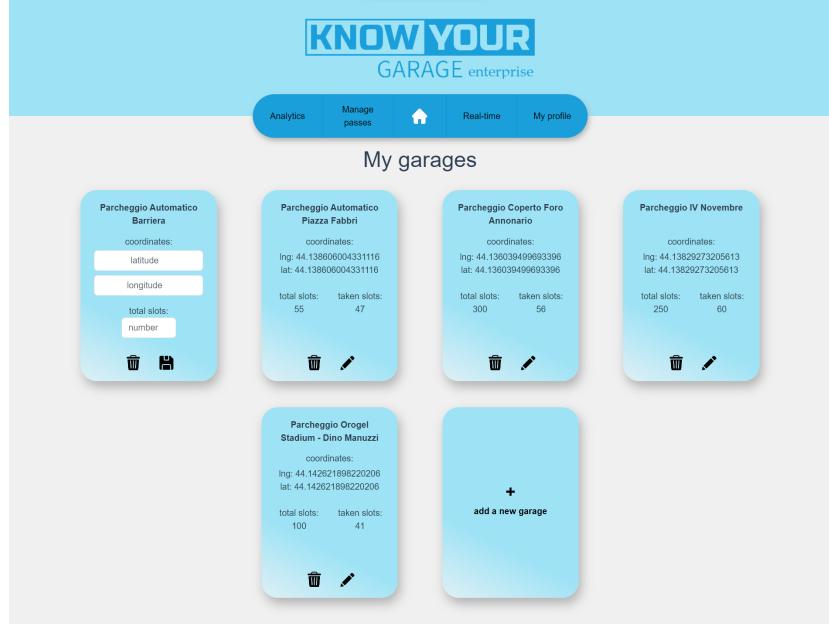


Figura 10: Visualizzazione dei parcheggi di un gestore

La modifica delle informazioni riguardanti un parcheggio esistente è possibile cliccando sulla matita posta al di sotto di ogni parcheggio. Cliccando su questo pulsante, la card diventerà modificabile e un pulsante salva apparirà al posto del pulsante modifica, consentendo all'utente di salvare le modifiche.

In maniera simile, il gestore può registrare un nuovo parcheggio, cliccando sulla card 'add a new garage' (figura 11).

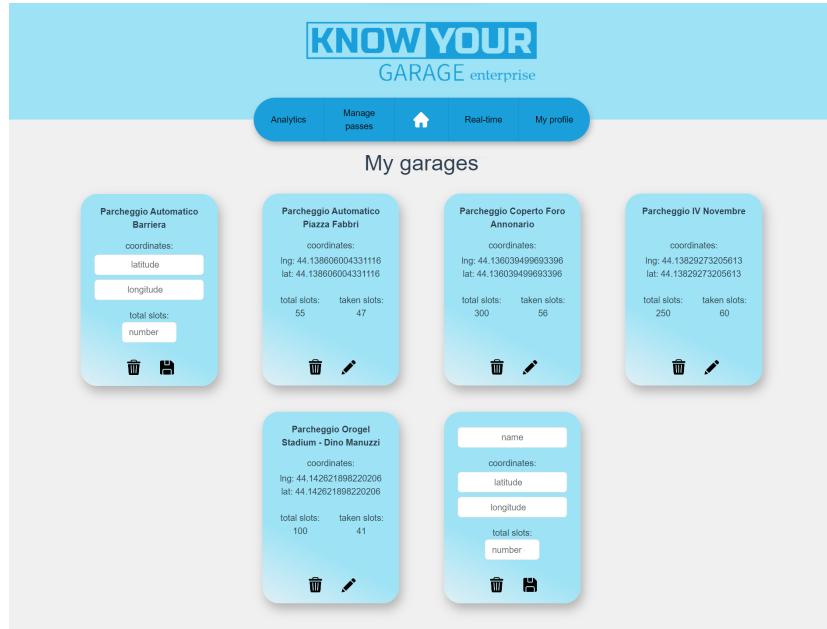


Figura 11: Registrazione di un nuovo parcheggio

Visitando la sezione ‘manage passes’ è invece possibile visualizzare gli abbonamenti associati a ciascun garage registrato, eliminare abbonamenti dal sistema o registrarne di nuovi indicando titolo, durata e costo dell’abbonamento. Come per i garage, anche per gli abbonamenti la modifica delle informazioni e la rimozione o l’aggiunta di nuovi abbonamenti avviene nello stesso modo, così da facilitare l’utente nell’utilizzo dell’applicazione.

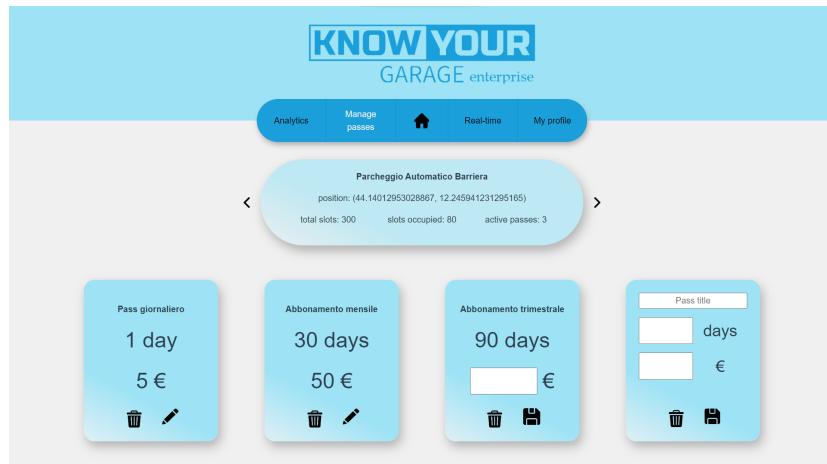


Figura 12: Visualizzazione, modifica e creazione degli abbonamenti

Nella sezione ‘analytics’, mostrata nella figura sottostante, il gestore di parcheggi può visualizzare per ciascun parcheggio registrato informazioni dettagliate riguardanti il numero di soste effettuate in un dato giorno e gli abbonamenti venduti per ciascuna tipologia. Nella tabella a destra sono invece mostrati il numero di abbonamenti venduti nell’arco di tempo indicato ed il numero di nuovi clienti, con indicazione sull’andamento (positivo o negativo).

Come accade anche per la pagina di gestione degli abbonamenti, il garage mostrato è selezionabile tra quelli registrati utilizzando le frecce poste a sinistra e a destra del garage.

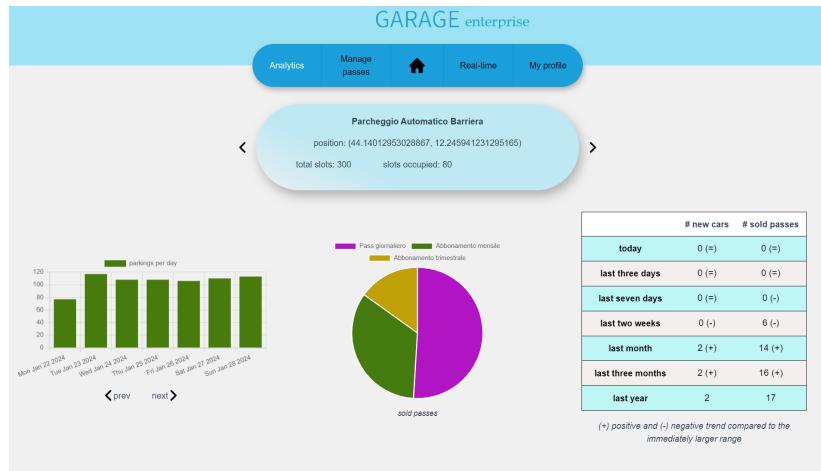


Figura 13: Statistiche sui parcheggi

Per concludere, nella sezione ‘real time’ (figura in basso) sono mostrati in tempo reale il numero di posti occupati per ciascun parcheggio (grafico a sinistra) e lo stato di occupazione percentuale (grafico a destra).

Questi grafici consentono di comparare velocemente l’attività dei parcheggi posseduti e possono aiutare nel comprendere meglio le esigenze degli utenti attraverso analisi comparative.

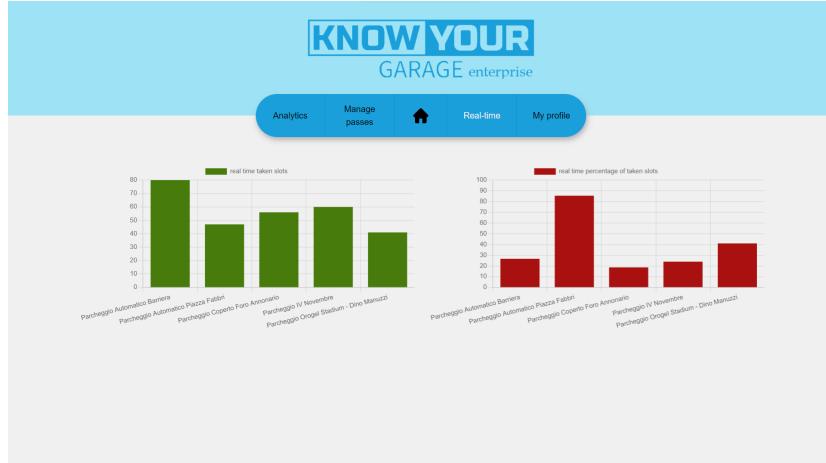


Figura 14: Informazioni in tempo reale

4 Tecnologie

Come anticipato nella sezione precedente, la soluzione adottata aderisce all’architettura MEVN, composta da MongoDB, Express, Vue.js e Node.js. Il sistema si suddivide in frontend e backend, utilizzando da un lato Vue.js per rendere l’interfaccia utente reattiva e dall’altro Node.js ed Express con MongoDB per la persistenza delle informazioni nel backend e al relativo accesso.

Oltre a queste tecnologie di base, ne sono state utilizzate altre per semplificare il processo di sviluppo, sia per quanto riguarda il frontend, che per il backend.

4.1 Client

Vite

L’utilizzo di *Vite* ha notevolmente accelerato il processo di sviluppo, sia per quanto riguarda la creazione dell’interfaccia utente, sia per quanto concerne l’interazione tra le diverse componenti del sistema.

Questa maggiore efficienza è da attribuire principalmente alla funzionalità di Hot Module Replacement, che consente di riflettere le modifiche al codice in maniera immediata nel browser, senza dover ricaricare ogni volta l’intera pagina.

Bootstrap

Per quanto riguarda la modellazione dell’interfaccia utente, si è fatto largo uso di Bootstrap.

Nonostante la stragrande maggioranza delle componenti sia stata realizzata da zero, in modo da conferire alla pagina uno stile originale e autentico, l’a-

dozione di Bootstrap è risultata essenziale per quanto concerne la disposizione degli elementi nella pagina attraverso l'utilizzo del grid system e delle flexbox.

L'utilizzo di Bootstrap ha semplificato notevolmente la gestione del ridimensionamento delle pagine, consentendo una disposizione dinamica degli elementi in base alle dimensioni della finestra.

Chart.js

La realizzazione dei grafici riassuntivi presenti nelle sezioni ‘analytics’ e ‘real time’ del profilo ‘gestore di parcheggi’ è avvenuta mediante l'utilizzo della libreria Chart.js.

In particolare, è stata adottato la libreria *vue-chartjs*, che agevola l'integrazione fluida dei grafici di Chart.js con la struttura organizzativa di Vue.js, non soltanto permettendo la creazione di rappresentazioni grafiche dettagliate e intuitive, ma ha anche ottimizzato l'interazione e la sincronizzazione tra i dati dinamici e la visualizzazione grafica sulla piattaforma.

Vue3-Google-Map

Uno degli aspetti principali dell'interfaccia degli utenti finali è sicuramente la presenza di una mappa interattiva che permette di visualizzare i parcheggi presenti nell'area mostrata, consentendo di visualizzare con facilità lo stato di occupazione di ciascun parcheggio in tempo reale.

Questa componente è stata realizzata facendo uso della libreria *Vue3-Google-Map*, la quale permette non soltanto di visualizzare una mappa statica, ma al contrario consente la creazione dinamica di marker personalizzati, attraverso i quali è stato possibile rappresentare con facilità i parcheggi e le relative informazioni.

Grazie all'utilizzo combinato di *Vue3-Google-Map* e *vue3-google-address-autocomplete* è stato possibile aggiungere una barra di ricerca dotata di autocompletamento che permette agli utenti di spostarsi velocemente da una località ad un'altra.

Crypto-js

In modo da consentire un maggiore livello di sicurezza, le password di utenti finali e gestori di parcheggi sono processate attraverso una funzione hash prima di essere inviate al server e quindi salvate nel database.

4.2 Server

Mongoose

Come descritto in precedenza, la persistenza dei dati è stata ottenuta grazie ad un database costruito su MongoDB.

Mongoose non solo permette di interfacciarsi con il database eseguendo operazioni quali interrogazioni, modifica o inserimento di nuovi dati, ma consente

anche di definire in maniera dettagliata lo schema dei dati, definendo per ogni tipologia di informazione i suoi attributi, la loro tipologia ed eventuali proprietà aggiuntive come ad esempio un valore di default.

4.3 Comunicazione

Axios

La maggior parte della comunicazione tra client e server avviene per mezzo di richieste HTTP.

Le richieste sono realizzate attraverso l'uso della libreria *axios*, che offre un modo di interfacciarsi con il server che si sposa bene con l'idea alla base di Vue. Le chiamate sono realizzate in maniera asincrona e la gestione delle risposte avviene per mezzo di callback.

La gestione delle richieste e delle relative risposte è lasciata in gran parte alle singole componenti. Per esempio la *CarCard* si occupa personalmente di effettuare la rimozione della relativa auto dal database, così facendo si ottiene una suddivisione delle responsabilità e la pagina principale risulta essere più semplice e comprensibile.

Socket.io

La notifica real time riguardante l'aggiornamento di posti disponibili in un determinato parcheggio avviene per mezzo di una comunicazione basata su socket. Quando un'auto parcheggia, il server emette una notifica e le componenti interessate reagiscono all'evento in base alla callback definita in fase di progettazione.

Come in precedenza, anche nel caso della comunicazione tramite socket, la gestione dell'evento è definita nelle singole componenti, rispettando così il Single Responsibility Principle.

5 Codice

5.1 Notifica eventi real-time

Il primo blocco di codice (listing 1) appartiene al backend. In questo blocco di codice è definita una funzione che emette un evento sul topic *parkings_changed* quando invocata.

Quando questa funzione viene chiamata, il server invia l'evento *parkings_changed* a tutti i client connessi.

```
1 exports.parkingChanged = (newParking) => {
2     io.emit('parkings_changed', newParking)
3 }
```

Listing 1: Emissione dell'evento

Il secondo blocco di codice (listing 2) descrive invece come una CarCard generica reagisce all'evento sopra descritto. Nello specifico, la CarCard verifica se l'evento riguarda l'auto che rappresenta e nel caso così fosse procede ad aggiornare il proprio campo *parked* con il valore corretto.

```

1 this.socket.on('parkings_changed', (newParking) => {
2   if(newParking.car_id === this.car._id.toString()){
3     if(newParking.end === null){
4       this.parked = true
5     }else{
6       this.parked = false
7     }
8   }
9 })

```

Listing 2: Reazione all'evento

5.2 Aggiornamento dei parcheggi mostrati

Questa porzione di codice (listing 3) mostra come i parcheggi mostrati sulla mappa sono aggiornati al variare della località mostrata.

Quando i confini della mappa cambiano, un evento cattura il fenomeno e invoca il metodo *boundsChanged*. Se il tempo trascorso dall'ultimo aggiornamento è inferiore al mezzo secondo, l'aggiornamento non è effettuato, questo per evitare di effettuare troppe richieste al server avendo lo stesso risultato.

Al contrario, se dall'ultimo aggiornamento è trascorso più di mezzo secondo, i marker presenti sulla mappa e nella lista adiacente ad essa sono aggiornati.

```

1 boundsChanged(){
2   if((new Date() - this.lastUpdateTime) < 500){
3     return
4   }
5   this.updateMarkers()
6 }

```

Listing 3: Aggiornamento dei parcheggi mostrati

6 Test

In modo da garantire il corretto funzionamento del sistema, ogni API è stata testata attraverso l'uso dello strumento Postman prima di essere effettivamente implementata all'interno del sistema.

Il risultato finale è stato fatto testare da membri esterni dal team di sviluppo così da evidenziare problemi non emersi nelle precedenti fasi di progettazione e implementazione. I tester hanno valutato l'aspetto grafico e l'usabilità di entrambe le tipologie di utenti (utenti finali e gestori di parcheggi), fornendo feedback risultati poi importanti per il miglioramento del sistema.

Dalle valutazioni effettuate sono emerse le seguenti considerazioni riguardanti le euristiche di Nielsen:

1. **Visibilità dello stato del sistema:** l'utente è sempre a conoscenza dello stato del sistema. Ogni form, una volta sottomesso, restituisce all'utente un feedback immediato. Ad esempio, durante la modifica della password, gli errori causati dall'inserimento di dati errati nel form, sono notificati attraverso un breve testo che spiega la causa dell'errore, come "la password corrente è errata" o "le password di verifica non coincide con la nuova password".
2. **Corrispondenza tra il mondo reale e il sistema:** numerosi elementi presenti nell'applicativo sviluppato possiedono una corrispondenza con concetti comunemente noti, come ad esempio un cestino per indicare l'eliminazione dell'elemento, una matita per indicare invece una modifica, o un floppy disk per salvare le modifiche effettuate.
3. **Controllo e libertà dell'utente:** all'interno di ogni sezione del sito, l'utente ha il totale controllo del sistema ed è in grado di cambiare pagina in qualsiasi momento, senza vincoli da parte del sistema.
4. **Consistenza e standard:** tutte le sezioni dell'applicativo adottano la stessa paletta di colori; inoltre, nelle differenti sezioni aventi simili scopi sono presenti elementi ricorrenti, così da limitare la conoscenza richiesta per l'utilizzo. Ad esempio l'eliminazione di un elemento avviene sempre attraverso un click su un cestino, o la navigazione di auto o parcheggi per la visualizzazione di statistiche di dettaglio avviene sempre utilizzando due piccole frecce poste a sinistra e a destra dell'elemento.
5. **Prevenzione degli errori:** l'applicativo cerca di ridurre al minimo la possibilità che l'utente commetta un errore durante il suo utilizzo adottando un aspetto minimale e fornendo indicazioni per quanto riguarda le operazioni più complesse e composte da più passaggi come ad esempio l'acquisto di un abbonamento.
6. **Riconoscimento anziché ricordo:** Le varie sezioni sono state progettate in modo tale da essere facilmente riconoscibili dall'utente ed il loro obiettivo intuibile.
7. **Flessibilità ed efficienza d'uso:** ogni funzionalità presente all'interno dell'applicativo richiede un numero limitato di passaggi per essere effettuata.
8. **Design estetico e minimalista:** l'interfaccia dell'applicativo è stata ideata in modo da contenere il minor numero di elementi possibile, così che l'utente possa concentrarsi sugli aspetti più importanti e non perda l'attenzione per colpa di altri elementi accessori.
9. **Riconoscimento, diagnosi e ripristino degli errori:** come anticipato, gli utenti sono notificati immediatamente della presenza di errori e della loro causa, pertanto possono facilmente procedere con la sua risoluzione.

10. **Aiuto e documentazione:** essendo l'applicativo molto intuitivo, per i motivi elencati in precedenza, le informazioni di aiuto sono presenti in pochissime sezioni del sistema. Per esempio, nella sezione ‘analytics’, vicino ad ogni grafico o tabella è presente una breve legenda descrivente il grafico che ne semplifica la sua interpretazione.

7 Deployment

7.1 Download e preparazione

1. Clonare il repository al seguente indirizzo:
 - https://github.com/NicoloMalucelli/know_your_garage
2. aprire un terminale sulla root del progetto ed eseguire i seguenti comandi:
 - (a) cd client
 - (b) npm install
 - (c) npm run-script build
 - (d) cd ../server
 - (e) npm install

7.2 Inizializzazione del database con i dati di test

I seguenti passaggi sono da svolgere qualora si voglia testare l'operatività del sistema. I dati che saranno importati sono dati di test creati in maniera artificiale, pertanto non corrispondono in alcun modo alla realtà.

1. Connettersi a MongoDB sulla porta 27017 utilizzando lo strumento MongoDB Compass
2. Creare un nuovo database dal nome **knowYourGarage** ed una prima collezione dal nome **cars**
3. Creare le seguenti nuove collezioni **admins**, **users**, **garages**, **parkings**, **passes** e **purchasedpasses**
4. Per ognuna delle collezioni create inserire i dati cliccando prima sulla collezione stessa e poi sul pulsante *addData*. I file json contenenti le collezioni si trovano nella cartella *./database* posizionata nella root del progetto.

7.3 Esecuzione

- Aprire un terminale sulla root del progetto ed eseguire i seguenti comandi:
 1. cd server
 2. npm start
- Aprire un secondo terminale sulla root del progetto ed eseguire i seguenti comandi:
 1. cd client
 2. npm run-script serve

7.4 Utilizzo

- Per utilizzare l'applicativo con un account utente finale, visitare
 - `http://localhost:4173/`È possibile quindi creare un nuovo profilo utente o utilizzare uno di quelli già esistenti, come ad esempio:
 - **email:** mario.rossi@example.com
 - **password:** password
- Per utilizzare un account con un account gestore di parcheggi visitare la sezione admin del sito:
 - `http://localhost:4173/admin`ed effettuare l'accesso con le seguenti credenziali:
 - **email:** comune.cesena@example.com
 - **password:** password

8 Conclusioni

A progetto concluso mi posso ritenere soddisfatto del risultato finale in quanto rispetta tutti i requisiti posti in fase di analisi, offrendo al tempo stesso un'interfaccia semplice ed intuitiva che riduce al minimo la possibilità di errore da parte dell'utente.

La navigazione e l'utilizzo del sito avvengono in maniera fluida senza alcun tipo di blocco o di lunghi caricamenti e l'interfaccia risulta reattiva alla dimensione della finestra.

Il progetto mi ha permesso di sperimentare con mano tutte le fasi di realizzazione di un sito web, consentendomi di unire tra loro le varie tecnologie viste durante le lezioni di laboratorio in un sistema coeso e perfettamente scalabile.

La realizzazione di questo progetto ha sicuramente aumentando le mie competenze in ambito di sviluppo di applicativi web basati sulle tecnologie più recenti.