



UNIVERSITÀ DI PISA

Dipartimento di Informatica
Corso di Laurea in Informatica (L-31)

Studio e realizzazione di una piattaforma per la programmazione e la gestione delle attività

Tutore accademico:

Alessio Conte

Candidata:

Anna Francesca Montagnoli
Matricola 578104

Tutore aziendale:

Domenico Arenga

Anno accademico:

2022/2023

Indice

1	Introduzione	1
1.1	Contesto del tirocinio	1
1.2	Descrizione del problema	1
1.3	Struttura e sviluppo del progetto	2
2	Tecnologie Utilizzate	4
2.1	PHP	4
2.2	Laravel	4
2.3	HTML	7
2.4	JavaScript	8
2.4.1	DataTables	8
2.4.2	FullCalendar	8
2.4.3	Chart.js	9
2.5	MariaDB	9
2.6	API e pacchetti	10
2.6.1	Autenticazione: <i>laravel/ui</i>	10
2.6.2	Permessi e ruoli: <i>Spatie Laravel-permission</i>	10
2.6.3	Generazione del QR Code: <i>Simple QrCode</i>	10
2.6.4	Scanner per il QR Code: <i>html5-qrcode</i>	11
2.6.5	Metodi di pagamento	11
2.6.6	reCAPTCHA	11
2.6.7	Leaflet e OpenStreetMap	12
2.7	AdminLTE	12
3	Database	13
3.1	Introduzione al database	13
3.2	Struttura del database	13
3.3	Interazione con il database	17
3.4	Seeding del database	18
4	Gestione degli utenti e autenticazione	19
4.1	Ruoli utenti e i loro permessi	19
4.2	Creazione degli utenti	20
4.3	Accesso degli utenti alla piattaforma	21

INDICE

4.4	Modifica informazioni utenti	22
5	Tesseramento	24
5.1	Criticità del vecchio sistema e soluzione progettata	24
5.2	Compilazione del form e validazione	25
5.3	Pagamento	29
5.4	Invio del QR Code tramite email	31
6	Eventi e luoghi	33
6.1	Ideazione del calendario	33
6.2	Implementazione del calendario	34
6.3	Prenotazioni eventi	35
6.4	Luoghi	36
7	Esperienza Utente	38
7.1	Introduzione alla UX	38
7.2	Progettazione della UX	39
8	Conclusione	45
8.1	Obiettivi raggiunti	45
8.2	Sviluppi futuri	46
8.3	Competenze acquisite	46

Capitolo 1

Introduzione

1.1 Contesto del tirocinio

La presente relazione descrive l'esperienza di tirocinio svolta dal 27/06/2023 al 08/09/2023 presso l'azienda Visual Engines srl, incentrata sulla realizzazione di una piattaforma per la programmazione di attività per un'associazione di promozione sociale. Il processo progettuale è iniziato dall'analisi dei requisiti, cercando di mettere a fuoco le necessità dei volontari dell'associazione, successivamente è passato allo sviluppo vero e proprio dell'applicazione web utilizzando il linguaggio richiesto dall'azienda, ovvero PHP e in particolar modo il framework open-source Laravel.

1.2 Descrizione del problema

La piattaforma è stata progettata per essere utilizzata dai volontari di Mangwana APS, un'associazione di promozione sociale che promuove iniziative di educazione ambientale, formativa e culturale. L'obiettivo principale del tirocinio è stato quello di sviluppare delle funzionalità utili ed intuitive per aiutare i volontari nel loro lavoro, nello specifico per la gestione dei tesseramenti e l'organizzazione degli eventi.

Un utente tramite il tesseramento, dopo aver pagato, può partecipare a tutti gli eventi organizzati dall'associazione. Questo sistema veniva gestito attraverso l'utilizzo di un form di Google compilato dall'utente, che doveva inoltre caricare un file contenente la conferma del pagamento effettuato. In questo modo il processo può risultare non ottimale ed eccessivamente lento, soprattutto nel caso in cui si ricevano molte richieste, poiché queste si devono accettare manualmente controllando se il pagamento dell'utente sia valido oppure no. Lo scopo della piattaforma che ho sviluppato è quello di rendere questa attività molto più veloce e affidabile.

Un ulteriore obiettivo è stato quello di implementare una soluzione immediata e semplice l'organizzazione degli eventi, che veniva svolta dagli operatori

attraverso l'utilizzo di Google Calendar. Ho integrato nella piattaforma un calendario, aggiungendo le funzionalità principali utilizzate dall'associazione in Google Calendar, ma rendendo il sistema maggiormente personalizzato rispetto alle necessità dei volontari.

1.3 Struttura e sviluppo del progetto

In relazione ai problemi descritti, l'applicazione web che ho sviluppato è stata ideata per prevedere una sezione accessibile a tutti (tesseramento), e un'altra sezione accessibile esclusivamente al personale autorizzato in cui si possono gestire e programmare le diverse attività ed iniziative che l'associazione porta quotidianamente avanti.

La sezione riguardante l'acquisto della tessera associativa prevede un form di registrazione in cui gli utenti devono inserire i propri dati di riferimento ed è stata integrata con piattaforme di pagamento come PayPal e Satispay, questo per consentire pagamenti contestuali al tesseramento. Da tale registrazione viene generato un codice QR legato alla registrazione effettuata che viene inviato all'utente tramite email. Il QR generato può essere validato, mediante uno scanner integrato nell'applicazione, dal personale dell'associazione per verificare il tesseramento agli eventi promossi dalla stessa. La parte riservata al personale autorizzato contiene diverse sezioni, tra cui:

- anagrafica utenti (registrati dal form indicato in precedenza);
- sezione dedicata alla gestione dei tesseramenti;
- sezione di programmazione attività con calendario per la gestione degli eventi (con relative operazioni create, read, update e delete);
- sezione relativa ai luoghi in cui si svolgeranno gli eventi, in cui è possibile creare, modificare o eliminare tali luoghi. Con integrazione di una mappa interattiva;

Inizialmente, non avendo nessuna conoscenza degli strumenti che avrei dovuto utilizzare, quali il framework Laravel e il linguaggio di programmazione PHP, ho dovuto dedicare del tempo ad approfondire tali argomenti e comprendere in modo dettagliato le loro capacità e potenzialità nello sviluppo di applicazioni web. Successivamente ho iniziato a pianificare la struttura del progetto, quindi pensare a come implementare le principali funzionalità della piattaforma e a come ottimizzarne l'utilizzo. L'implementazione ha avuto inizio con il semplice sviluppo del processo di autenticazione degli utenti, cioè le funzioni per la registrazione, il login e il logout, e di conseguenza la creazione del database, nello specifico la tabella per memorizzare gli utenti e le loro informazioni. Ho deciso di dividere gli utenti in tre ruoli, in base ai permessi che posseggono: Amministratore, Operatore e Base. Dopo aver gestito l'autenticazione e l'anagrafica utenti, ho implementato il meccanismo per il tesseramento. Quindi ho creato il form, accessibile a tutti, integrando i

sistemi di pagamento di PayPal e Satispay. Ho deciso di aggiungere anche la possibilità di pagare tramite bonifico IBAN, in questo caso i volontari possono visualizzare in una pagina tutte le richieste di tesseramento ricevute, per ognuna di esse controllare l'avvenuto pagamento e di conseguenza accettare o rifiutare tale richiesta. Successivamente ho integrato nella piattaforma la pagina per la scansione dei codici QR, che vengono inviati via email agli utenti dopo la conferma del tesseramento. Infine mi sono occupata della sezione dedicata all'organizzazione degli eventi. Qui, i volontari possono accedere ad un calendario, che agevola la creazione, la modifica e la cancellazione di questi. Inoltre, sulla pagina di ciascun evento, è disponibile la funzione per la registrazione delle prenotazioni. Queste vengono aggiunte in una tabella, che funge anche da lista di controllo, per tenere traccia della partecipazione delle persone all'evento. La sezione degli eventi è direttamente correlata a quella dei luoghi, poiché ogni evento richiede un'associazione con un luogo specifico in cui si svolgerà. Qui, tramite l'integrazione di una mappa, i volontari possono registrare modificare o eliminare i luoghi in modo efficiente.

Per quanto riguarda la struttura della relazione, i capitoli si strutturano in questo modo:

Nel prossimo capitolo sono illustrate le tecnologie che sono state utilizzate per la realizzazione della piattaforma web.

Il capitolo 3 è dedicato a descrivere la struttura del database utilizzato e l'interazione con esso.

Il capitolo 4 riguarda la progettazione dell'anagrafica utenti e il processo di autenticazione.

Il capitolo 5 descrive lo sviluppo della sezione dedicata al tesseramento.

Nel capitolo 6 è illustrata l'ideazione e l'implementazione delle sezioni degli eventi e dei luoghi.

Il capitolo 7 è dedicato all'esperienza utente, in cui vengono discusse le scelte progettuali per rendere l'utilizzo della piattaforma il più intuitivo possibile. L'ultimo capitolo è la conclusione, in cui viene riepilogato l'intero progetto, mettendo in evidenza i risultati ottenuti, i problemi affrontati e le possibili direzioni future per il miglioramento dell'applicazione web.

Capitolo 2

Tecnologie Utilizzate

2.1 PHP

“A popular general-purpose scripting language that is especially suited to web development. Fast, flexible and pragmatic, PHP powers everything from your blog to the most popular websites in the world.” [13]

PHP è un linguaggio di scripting server-side, nato per sviluppare pagine web dinamiche. È uno dei linguaggi di programmazione più usati per lo sviluppo web ed è di facile apprendimento, oltre ad essere potente e versatile. PHP offre una serie di funzioni e librerie dedicate al trattamento delle richieste HTTP, all'interazione con il database e alla gestione delle sessioni, che sono fondamentali per una piattaforma online.

2.2 Laravel

Laravel è un framework open source per lo sviluppo di applicazioni web in PHP. L'azienda ha proposto il suo utilizzo per diversi motivi:

Community attiva

Laravel è uno strumento ampiamente utilizzato ed ha una community molto attiva che fornisce supporto e condivide soluzioni per implementare le funzionalità più note. Questo ha reso il processo di sviluppo più semplice e veloce anche grazie ai pacchetti e alle API che ho potuto integrare nel progetto, di cui discuterò più avanti.

Architettura MVC

Laravel segue il pattern architetturale MVC (Model-View-Controller), che fornisce un ausilio per organizzare il codice in modo modulare, separando la lo-

gica dell'applicazione dall'implementazione dell'interfaccia utente e rendendo il codice più pulito, facilmente manutenibile e leggibile.

- **Controllers:** in Laravel i controllers sono contenuti all'interno della cartella *app/Http/Controllers*. I controllers consentono di raggruppare la logica per la gestione di particolari richieste in un'unica classe [4], ad esempio nel mio caso ho deciso di raccogliere le richieste riguardanti la gestione degli utenti in un unico controller chiamato *UserController*, la logica riguardante la gestione dei tesseramenti in *MembershipController* e così via;
- **Models:** i modelli sono contenuti all'interno della cartella *app/Models*. Ogni modello è collegato ad una tabella nel database ed è responsabile della gestione dei suoi dati, dalla creazione alla cancellazione. Laravel permette all'applicazione di interagire con i dati nel database tramite query effettuate sui modelli;
- **Views:** le viste, sono contenute nella cartella *resources/views* e contengono la parte grafica dell'applicazione;

In figura [2.1] è rappresentato come l'architettura MVC interagisce con l'utente e con il database. Un utente invia una richiesta HTTP al server, questa richiesta può essere un clic su un link, la compilazione di un modulo o qualsiasi altra azione che richieda una risposta dal server. Laravel utilizza un sistema di routing per indirizzare la richiesta al Controller corretto. Il Controller corrispondente riceve la richiesta e, se necessario, interagisce con il Model per ottenere i dati richiesti dal database. Una volta ottenuti i dati, il Controller li passa alla View corretta. La View elabora i dati ricevuti dal Controller e crea la pagina HTML da restituire al browser dell'utente.

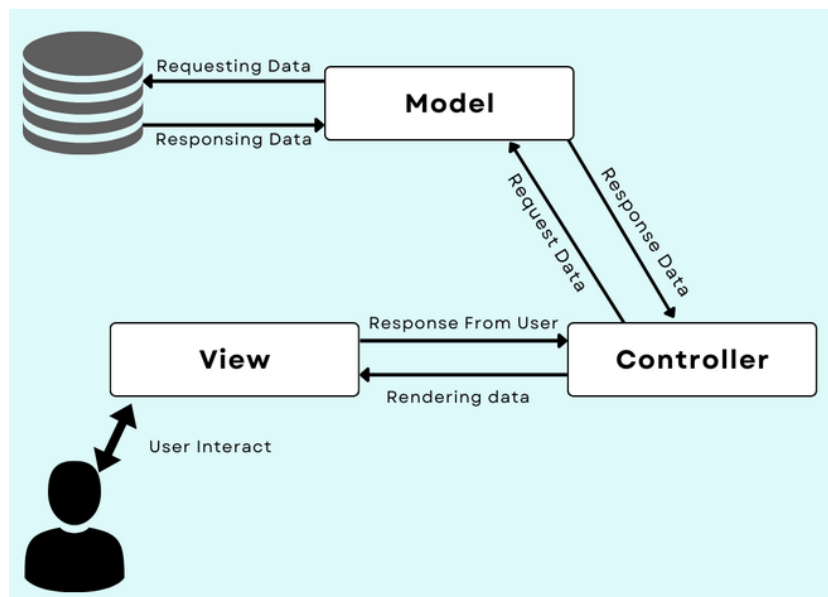


Figura 2.1: Schema che rappresenta l'architettura MVC. Immagine presa dal sito GeeksForGeeks [7]

Interazione con il database

Laravel rende l'interazione con i database molto semplice tramite l'utilizzo di una query builder oppure attraverso Eloquent ORM (Object-Relational Mapping), attraverso il quale per ogni tabella del database viene creato un modello, che viene utilizzato per interagire con la tabella corrispondente. Laravel supporta inoltre cinque database: MariaDB, MySQL, PostgreSQL, SQLite e SQL Server[8]. Per la piattaforma che ho implementato ho utilizzato MariaDB, di cui parlerò più avanti. Un'altra funzionalità vantaggiosa di Laravel, di cui ho fatto ampio uso, sono le migrations: uno strumento per il versioning del database. Ciascun file di migration rappresenta un'operazione da eseguire sul database dell'applicazione, con Laravel è possibile gestirli con appositi comandi, con lo scopo di muoversi tra le diverse versioni del database cancellando o ricreando le tabelle.

Sicurezza

Laravel offre diverse funzionalità per migliorare la sicurezza delle applicazioni web. Queste includono protezione contro attacchi CSRF (Cross-Site Request Forgery) e XSS (Cross-Site Scripting), autenticazione utente e crittografia dei dati. Inoltre la classe Hash di Laravel offre una funzione di hashing Bcrypt, che ho utilizzato per salvare le password nel database in modo sicuro.

Servizi di posta

Una funzione importante all'interno della piattaforma, è quella dell'invio di email agli utenti. Laravel ha reso considerevolmente semplice e veloce il suo sviluppo, infatti oltre a fornire un'API sulla libreria SwiftMailer, il framework supporta molteplici driver per inviare email tra cui Mailgun, che inizialmente ho scelto di utilizzare. Tuttavia nel testarlo mi sono resa conto che spesso le email inviate finivano nello spam del destinatario, per questo ho deciso di implementare l'invio di email utilizzando il server SMTP (Simple Mail Transfer Protocol) di Gmail, il quale garantisce che le email non vengano archiviate nella cartella dello spam del destinatario ed è la scelta predominante di molti utenti per la comunicazione via email grazie alla stabilità e le prestazioni costanti dei suoi server [15].

Routing semplice

In Laravel le routes consentono di indirizzare la richiesta HTTP ricevuta al controller desiderato, che gestirà le risposte. Questo meccanismo semplifica notevolmente la navigazione all'interno della piattaforma ed aumenta la leggibilità del codice, rendendolo maggiormente organizzato. È possibile inoltre controllare se l'utente che ha inviato la richiesta ha i permessi necessari per accedere a quel percorso URL, grazie ai middleware, che possono essere applicati a livello di route.

Un esempio di utilizzo dei middleware nelle routes è il seguente in figura [2.1], nella quale sono rappresentate le route a cui può accedere solo l'amministratore.

```
1 /**
2  * Chiama la funzione per la creazione di un nuovo utente
3  * Può essere eseguita solo dall'amministratore
4  */
5 Route::post('/users/create', 'UserController@userCreate')
6     ->middleware('auth')->middleware('can:edit-users');
7
8 /**
9  * Route per eliminare un utente
10 * può essere eseguita solo dall'amministratore
11 */
12 Route::get('users/delete/{id}', 'UserController@userDelete')
13     ->middleware('auth')->middleware('can:edit-users')
14     ->name('delete');
15
16 /**
17 * Routes per modificare il profilo di un utente
18 * può essere eseguita solo dall'amministratore
19 */
20 Route::post('/users/edit/{id}', 'UserController@userEdit')
21     ->middleware('auth')->middleware('can:edit-users');
```

Listing 2.1: Esempio dell'utilizzo dei middleware per alcune routes a cui può accedere solo l'amministratore, avendo il permesso "can:edit-users"

2.3 HTML

Per la progettazione dell'interfaccia dell'applicazione, ho optato per l'utilizzo di HTML (HyperText Markup Language), un linguaggio di markup che consente di definire la struttura e la disposizione degli elementi all'interno di una pagina web. Questa scelta è stata motivata dal fatto che HTML è lo standard per la creazione di interfacce utente ed è ampiamente supportato da tutti i moderni browser, garantendo così una corretta visualizzazione del contenuto di una pagina. Nel contesto della mia applicazione, ho integrato il codice HTML direttamente nei file con estensione .blade.php di Laravel, ovvero i template Blade, che consentono di includere codice PHP all'interno delle views, così da rendere più facile la realizzazione di alcune operazioni. Inoltre Blade possiede alcune scorciatoie per le strutture di controllo più comuni (ad esempio if, while, for), che si rivelano estremamente utili nell'implementazione dell'interfaccia [3]. Questo approccio mi ha consentito di sfruttare la potenza di Laravel nella gestione delle views, garantendo al contempo una struttura HTML ben formata e una presentazione visiva coerente su tutti i dispositivi e i browser supportati.

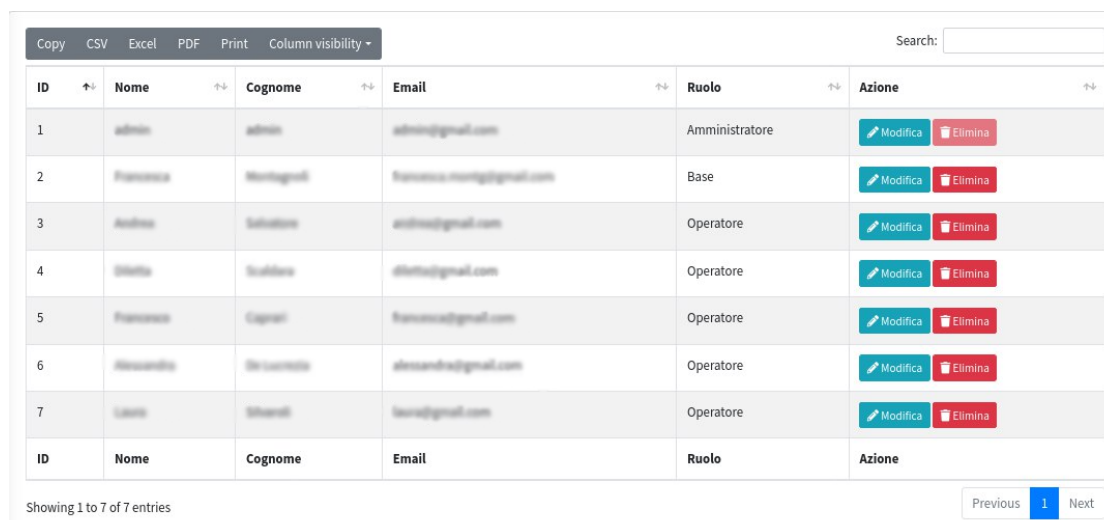
2.4 JavaScript

Per arricchire l'interfaccia grafica e conferirle un aspetto dinamico e interattivo, ho deciso di utilizzare JavaScript, un linguaggio di programmazione ampiamente sfruttato nello sviluppo di pagine web. Questa scelta mi ha permesso di introdurre diverse funzionalità avanzate all'interno dell'applicazione, anche grazie all'utilizzo di librerie JavaScript, tra cui le seguenti:

2.4.1 DataTables

DataTables è una libreria JavaScript che consente di creare e gestire tabelle interattive all'interno di pagine web. Questa libreria offre una serie di funzionalità avanzate per la visualizzazione e la manipolazione dei dati. Alcune delle caratteristiche principali di DataTables che ho utilizzato nella piattaforma sono: la ricerca e i filtri, che permettono all'utente di cercare e filtrare i dati nelle tabelle, l'ordinamento che consente di ordinare le colonne delle tabelle in ordine crescente o decrescente e la paginazione che è utile nel caso in cui sia presente una grande quantità di dati, infatti permette di dividerli in pagine più piccole per facilitarne la navigazione.

Un esempio di utilizzo di questa libreria all'interno della piattaforma, si trova nella sezione per la gestione degli utenti, in cui l'amministratore può visualizzare la lista di tutti gli utenti registrati, come mostrato in figura [2.2].



ID	Nome	Cognome	Email	Ruolo	Azione
1	admin	admin	admin@gmail.com	Amministratore	Modifica Elimina
2	Francesca	Montagni	francesca.montagni@gmail.com	Base	Modifica Elimina
3	Andrea	Salvatore	andrea@gmail.com	Operatore	Modifica Elimina
4	Stefano	Scudiero	stefano@gmail.com	Operatore	Modifica Elimina
5	Francesco	Cagnoli	francesco@gmail.com	Operatore	Modifica Elimina
6	Alessandra	De Lencastre	alessandra@gmail.com	Operatore	Modifica Elimina
7	Luca	Storati	luca@gmail.com	Operatore	Modifica Elimina

Figura 2.2: Datatable utilizzata per visualizzare la lista di tutti gli utenti registrati nella piattaforma (nella pagina Gestione Utenti accessibile solo dall'amministratore)

2.4.2 FullCalendar

FullCalendar è una libreria JavaScript per la creazione di calendari interattivi all'interno di applicazioni web. È ampiamente utilizzata per visualizzare eventi, impegni, appuntamenti e altre informazioni. Ho scelto di utilizzare questa

libreria per implementare una delle funzionalità più importanti dell'applicazione, ovvero l'organizzazione degli eventi. Tramite il calendario i volontari dell'associazione possono aggiungere e visualizzare i vari eventi, in particolare FullCalendar mi ha permesso di personalizzarne l'aspetto, infatti ho scelto di rappresentare gli eventi confermati di colore verde e quelli non confermati di rosso, al fine di offrire ai volontari una visualizzazione degli eventi più chiara e intuitiva.

2.4.3 Chart.js

Chart.js è una libreria JavaScript open-source utilizzata per creare grafici interattivi e dinamici di diverso tipo all'interno di applicazioni web. Questa libreria offre una soluzione semplice e flessibile per visualizzare dati in forma grafica, rendendo più comprensibili le informazioni complesse. Ho utilizzato Chart.js per aggiungere molteplici grafici all'interno del gestionale. Un esempio, rappresentato in figura [2.3] è quello presente nella pagina iniziale degli operatori, utile per visualizzare la crescita del numero dei tesseramenti effettuati durante l'anno.

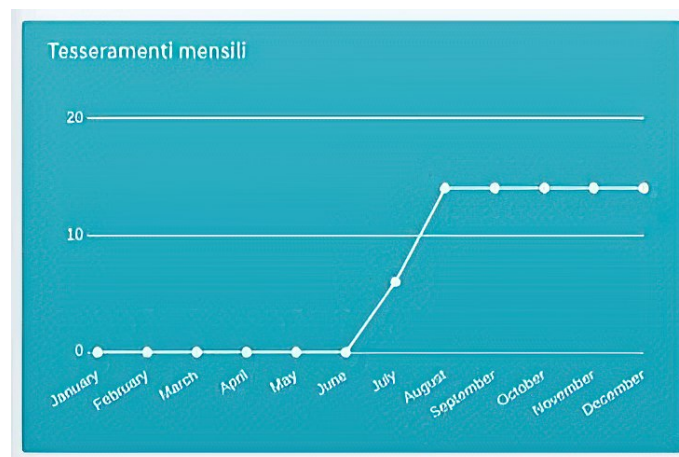


Figura 2.3: Grafico che mostra l'andamento del numero dei tesseramenti durante l'anno

2.5 MariaDB

MariaDB è un RDBMS (Relational Database Management System), è uno dei database relazionali più popolari ed è stato creato da un fork di MySQL. È stato costruito basandosi sui valori di performance, stabilità, e trasparenza[10].

Mi è stato consigliato dall'azienda prima di tutto perchè è open source, cioè è gratuito e offre accesso al suo codice sorgente. Inoltre riceve aggiornamenti e miglioramenti regolari, il che permette al DBMS di rimanere sempre aggiornato e al passo con i tempi. Infine nonostante i frequenti update, MariaDB offre la retrocompatibilità, ovvero la sua ultima versione è compatibile con le versioni precedenti.

Mettendo a confronto MariaDB con MySQL, come descritto nell'articolo [17], il primo ha maggiori punti di forza, infatti gli storage engines di MySQL: MyISAM e InnoDB, sono sostituiti in MariaDB da Aria e XtraDB. Aria offre un sistema di caching migliore, che fa la differenza per quanto riguarda le operazioni che richiedono un grande utilizzo del disco. XtraDB migliora invece tutti i problemi di InnoDB riguardanti la stabilità e le performance lente. In aggiunta MariaDB ha messo in atto delle ottimizzazioni per diverse query connesse all'accesso al disco, ad esempio operazioni di join, questo eleva ancora di più le performance del database scelto.

2.6 API e pacchetti

2.6.1 Autenticazione: *laravel/ui*

Per il processo di autenticazione degli utenti ho utilizzato il pacchetto *laravel/ui*: installandolo, sono state automaticamente create le views per la registrazione e per il login, che poi ho modificato al fine di rendere più adatte alla piattaforma, e diverse routes per il funzionamento vero e proprio dell'autenticazione. È stato generato anche un HomeController per gestire le richieste dopo il login e altri controller tra cui il RegisterController per gestire le registrazioni dei nuovi utenti, il LoginController per l'autenticazione vera e propria e i controller ForgotPasswordController e ResetPasswordController, contenenti la logica per il reset della password nel caso in cui venga dimenticata [2].

2.6.2 Permessi e ruoli: *Spatie Laravel-permission*

Per associare agli utenti ruoli e permessi ho utilizzato il pacchetto *laravel-permission* di Spatie. Una volta installato, permette di creare ruoli e permessi per gli utenti ed assegnarli ad essi, il tutto viene anche riportato nel database. Nel contesto della piattaforma gestionale ho creato tre ruoli, Amministratore, Operatore e Base ed ho collegato ad ognuno di essi diversi permessi, che serviranno ad accedere a determinate pagine della piattaforma.

2.6.3 Generazione del QR Code: *Simple QrCode*

Simple QrCode di Simple Software è un pacchetto per Laravel che semplifica la generazione di codici QR. Questo mi è stato di grande aiuto nel creare QR che contenessero un link ad una pagina della piattaforma, in particolare i codici generati devono essere inviati agli utenti tramite email e anche a questo scopo il pacchetto fornisce una funzione apposita, che può generare i codici con formato png, così da poter essere visualizzati facilmente all'interno dell'email.

2.6.4 Scanner per il QR Code: *html5-qrcode*

Affinché i volontari possano verificare i codici QR delle persone che si presentano agli eventi dell'associazione, è necessario fornire loro una pagina dotata di uno scanner per QR. Per integrare lo scanner nella piattaforma ho utilizzato la libreria *html5-qrcode*, attraverso la quale ho dovuto semplicemente includere nella vista della pagina di riferimento gli script contenuti nella documentazione della libreria.

2.6.5 Metodi di pagamento

Nella pagina per il tesseramento ho integrato diversi metodi di pagamento: PayPal, Satispay e bonifico tramite IBAN. Sia PayPal che Satispay funzionano tramite reindirizzamento, ovvero portano l'utente al sito web del servizio scelto per finalizzare il pagamento.

A questo scopo ho utilizzato il plugin *srmklive/laravel-paypal*, che fornisce un modo semplice per gestire i pagamenti tramite PayPal.

Per Satispay ho invece integrato la sua API, in particolare ho deciso di implementare il flusso di pagamento One-off-Web-redirect, quindi quando l'utente clicca sul pulsante scegliendo questo tipo di pagamento, viene reindirizzato al sito web di Satispay dove può effettuare la transazione.

2.6.6 reCAPTCHA

Nel form per il tesseramento, ho deciso di inserire anche il controllo di Google reCAPTCHA, per aggiungere un livello di sicurezza. Infatti reCAPTCHA utilizza un motore avanzato di analisi del rischio e sfide adattive per impedire a software dannosi di compiere attività abusive sul proprio sito web [14].

Per l'integrazione ho seguito la documentazione di Google reCAPTCHA v2, aggiungendo gli script necessari alla view corrispondente.

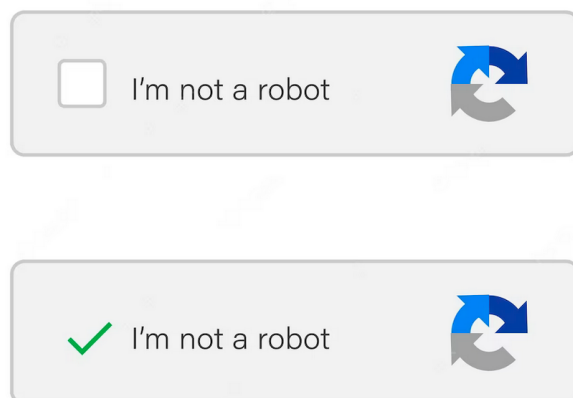


Figura 2.4: Google reCAPTCHA v2 utilizzato nel form del tesseramento

2.6.7 Leaflet e OpenStreetMap

Per integrare nel codice la mappa nella sezione Luoghi ho utilizzato Leaflet, una libreria JavaScript open source per mappe interattive. In particolare ho integrato nel codice la mappa OpenStreetMap ed ho aggiunto il plugin *perliedman/leaflet-control-geocoder*, che mi ha permesso di incorporare alla mappa il sistema di ricerca tramite indirizzo, come si può vedere in figura [2.5].

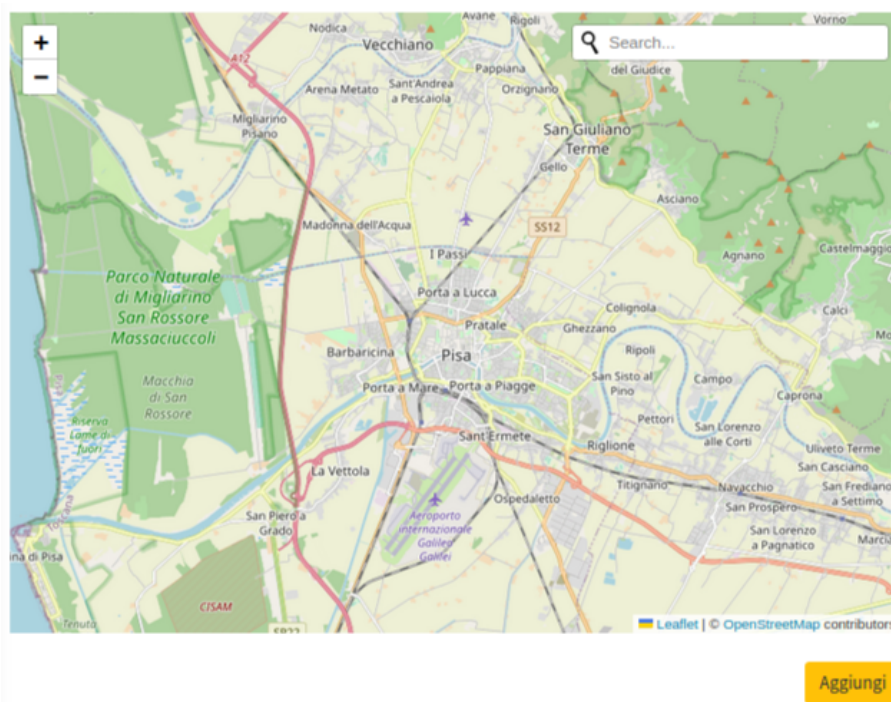


Figura 2.5: Integrazione di OpenStreetMap nella sezione per registrare i luoghi

2.7 AdminLTE

Per la parte grafica della piattaforma, mi è stato consigliato dall'azienda di utilizzare AdminLTE, un popolare template open source per applicazioni web, utilizzato in particolare per la creazione di pannelli di amministrazione e dashboard.

È basato sulla libreria Bootstrap, infatti utilizza molti suoi componenti per il design e il re-styling di plugin comunemente usati. L'utilizzo di questo template mi ha permesso di rendere più veloce lo sviluppo dell'interfaccia grafica e ha reso più facile il processo di apprendimento di HTML e CSS, che non conoscevo. Grazie a questo sono riuscita a personalizzare l'interfaccia a seconda dei bisogni della piattaforma.

Capitolo 3

Database

Il database ha un ruolo molto importante per quanto riguarda l'archiviazione e la gestione dei dati in modo efficace. In questo capitolo spiegherò la struttura e l'architettura che ho scelto per il database del gestionale, il design delle tabelle e le relazioni tra esse in modo da far comprendere come sia stato progettato per supportare le funzionalità chiave dell'applicazione web.

3.1 Introduzione al database

Nel contesto della piattaforma di programmazione attività, il database ha un ruolo centrale, si occupa infatti di amministrare i dati più importanti. In particolare le funzionalità che hanno la necessità di interagire con il database sono la registrazione e l'autenticazione degli utenti, la creazione e la gestione degli eventi e l'archiviazione degli utenti che hanno effettuato il tesseramento. Avendo la necessità di monitorare tutti questi dati, ho scelto di utilizzare un database relazionale, con cui ho creato molteplici tabelle, contenenti i dati da gestire, e le ho messe in relazione tra loro attraverso il meccanismo delle chiavi esterne e delle chiavi primarie.

Come già detto nel capitolo precedente, l'azienda mi ha consigliato di utilizzare il DBMS MariaDB, poiché è open source, permette di realizzare basi di dati relazionali ed è supportato da Laravel.

3.2 Struttura del database

Il database progettato è costituito da diverse tabelle, ciascuna contenente informazioni specifiche relative ai dati archiviati. Nelle figure [3.1] e [3.2], sono presentati gli schemi che mi hanno aiutato nella progettazione di questo database. A seguire, vengono elencate le tabelle presenti e spiegate le relazioni che intercorrono tra di esse, per comprendere meglio gli schemi sopra citati.

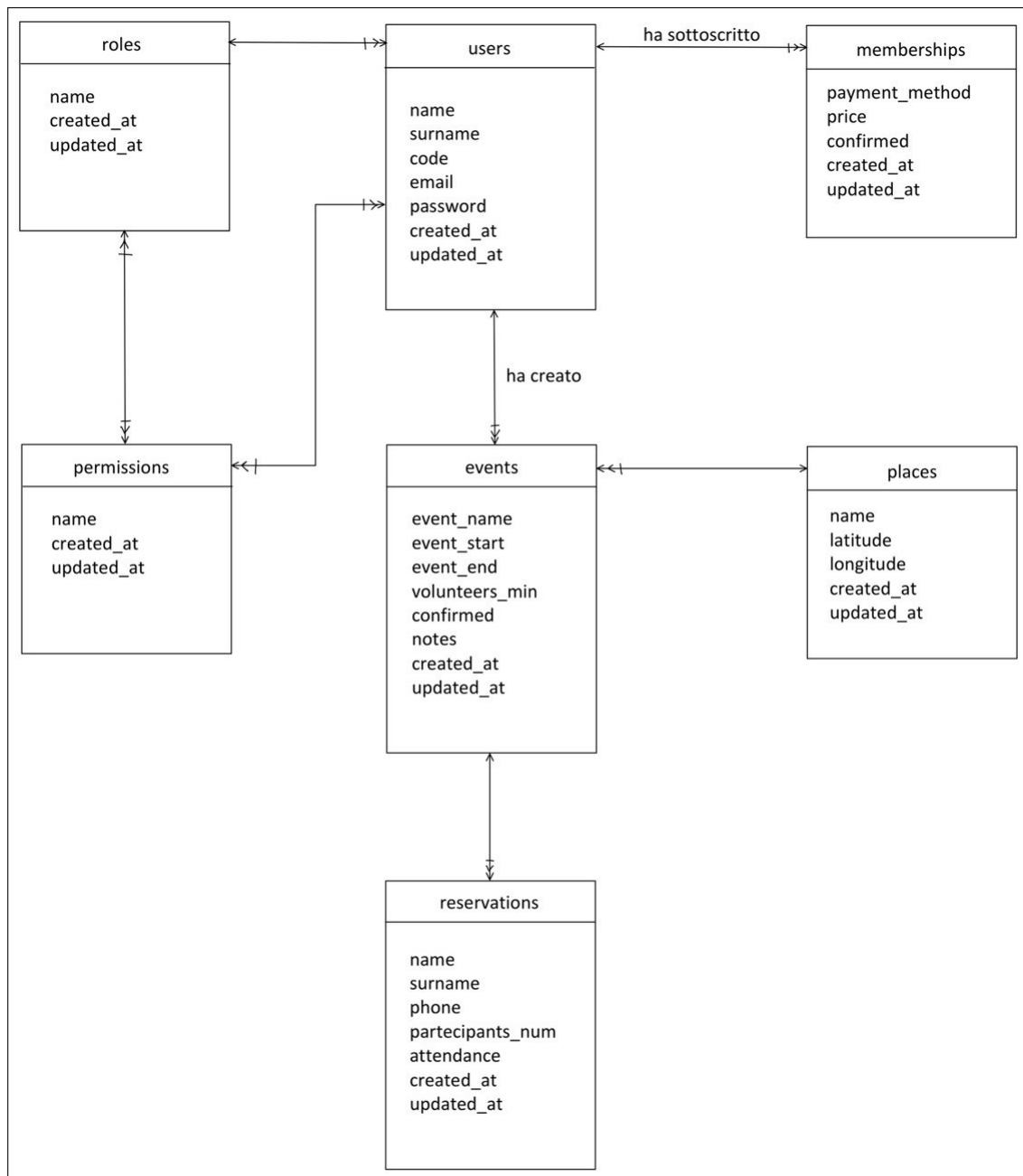


Figura 3.1: Schema concettuale del database, che rappresenta i tipi di dato da rappresentare, con i loro attributi e le loro relazioni

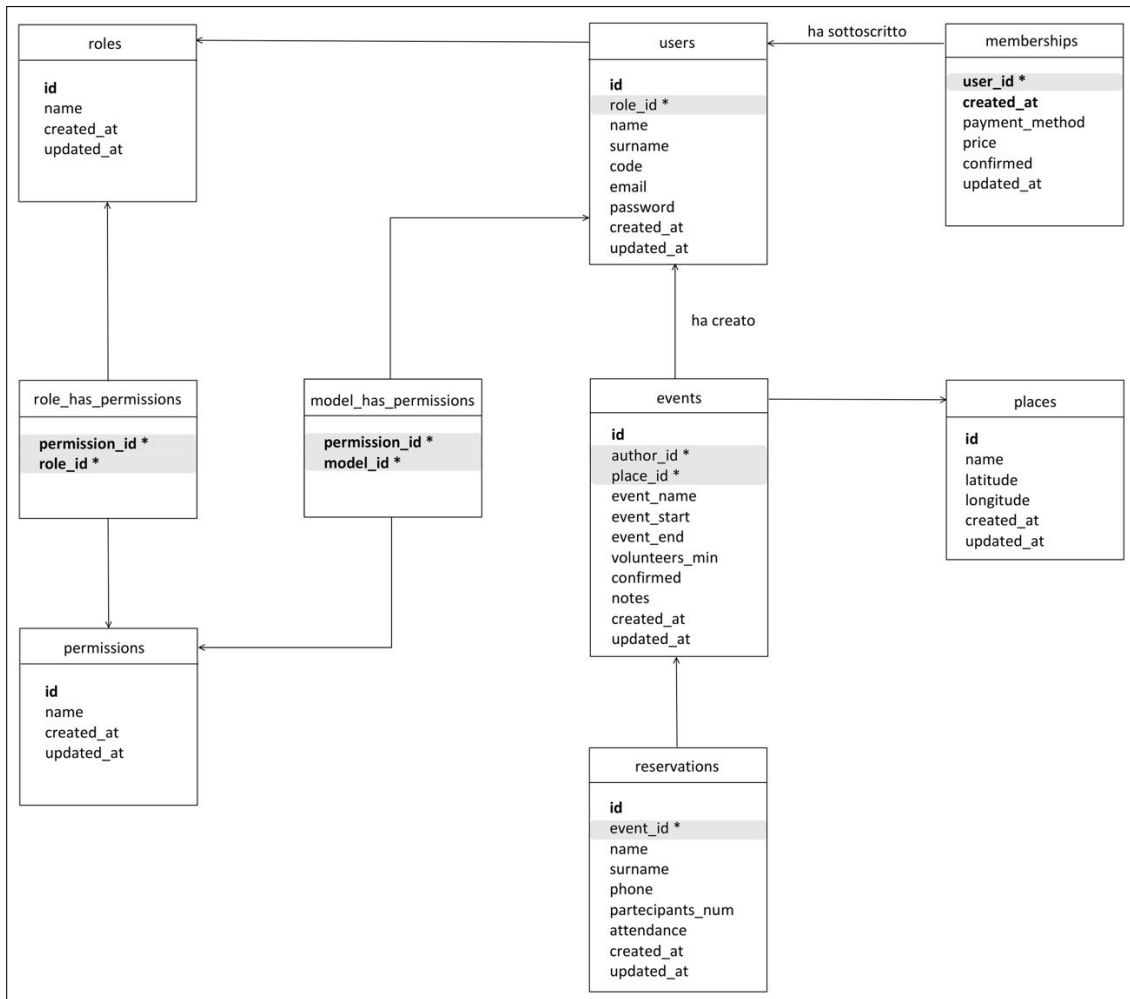


Figura 3.2: Schema logico relazionale del database, che rappresenta tutte le tabelle, con i loro attributi e le loro relazioni. Gli attributi in grassetto sono le chiavi primarie, mentre gli attributi evidenziati e seguiti da un asterisco sono le chiavi esterne

- **users** è la tabella che contiene la lista di tutti gli utenti registrati nell'applicazione e le loro informazioni. Ha come attributi id, role_id, nome, cognome, codice fiscale, email, password e i timestamp di creazione e aggiornamento. L'id è la chiave primaria ed è il codice univoco che distingue ciascun record della tabella, anche l'attributo email è univoco per ogni utente, infatti non si possono registrare due utenti diversi con la stessa email. La password può avere anche valore nullo, questo perché gli utenti che hanno il ruolo di Amministratore o Operatore necessitano di una password per accedere al gestionale, al contrario gli utenti con ruolo Base non devono poter effettuare il login nella piattaforma, quindi non la possiedono. Analogamente il codice fiscale può essere nullo, poiché gli utenti che hanno come ruolo Operatore o Amministratore, ma non hanno mai effettuato il tesseramento, non hanno dovuto mai inserirlo all'interno della piattaforma. L'attributo role_id è la chiave esterna che si riferisce alla tabella roles. Infatti le due tabelle sono in relazione uno a molti: un utente deve obbligatoriamente avere un ruolo e un ruolo può essere associato a più utenti.

- **memberships** è la tabella che contiene la lista di tutti i tesseramenti effettuati, sia confermati che da confermare. Ha come attributi `user_id`, metodo di pagamento, prezzo, una flag `confirmed` e i timestamp di creazione e aggiornamento. La combinazione degli attributi `user_id` e `created_at` identificano univocamente ogni riga della tabella, in questo modo non ho avuto la necessità di creare un nuovo attributo per la chiave primaria. L'attributo `user_id` è la chiave esterna, che corrisponde all'id, nella tabella `users`, dell'utente che ha effettuato il tesseramento. In questo modo la tabella `users` e la tabella `memberships` sono in relazione uno a molti. Infatti l'attributo `id` nella tabella `users` è progettato per contenere valori unici, mentre il campo `user_id` nella tabella `memberships` è progettato per consentire più istanze dello stesso valore, questo perché un utente può effettuare più tesseramenti, al massimo uno ogni anno.
- **places** è la tabella che contiene la lista di tutti i luoghi registrati. Ha come attributi `id`, nome del luogo, latitudine, longitudine e i timestamp di creazione e aggiornamento.
- **events** è la tabella che contiene la lista degli eventi creati. Ha come attributi `id`, `author_id`, `place_id`, nome dell'evento, le date di inizio e fine dell'evento, il numero minimo di volontari necessario, la flag `confirmed`, le note dell'evento e i timestamp di creazione e aggiornamento. L'`id` è la chiave primaria, `place_id` rappresenta il luogo dove si svolge l'evento ed è una chiave esterna che si riferisce alla tabella `places`, `author_id` è la chiave esterna che si riferisce alla tabella `users` e rappresenta l'utente che ha creato l'evento. Infatti le tabelle `places` e `events` sono in relazione uno a molti, poiché ad un luogo possono corrispondere molteplici eventi, anche le tabelle `users` e `events` hanno lo stesso tipo di relazione, poiché un utente può essere l'autore di vari eventi.
- **reservations** è la tabella che contiene le prenotazioni agli eventi, ha come attributi `id` della prenotazione, `event_id` dell'evento a cui è riferita la prenotazione, nome, cognome e numero di telefono di chi ha effettuato la prenotazione, numero di persone per cui si effettua la prenotazione e i timestamp di creazione e aggiornamento. L'`id` è la chiave primaria, mentre `event_id` è la chiave esterna che pone in relazione uno a molti la tabella `events` con la tabella `reservations`: ad un unico evento corrispondono molteplici prenotazioni.

Le seguenti tabelle sono state create automaticamente quando ho installato il pacchetto *laravel-permission* di Spatie per i ruoli e i permessi degli utenti:

- **roles** è la tabella che contiene la lista di tutti i ruoli che possono avere gli utenti, ovvero Amministratore, Operatore o Base. Ha come attributi `id`, nome e i timestamp di creazione e aggiornamento. L'`id` è la chiave primaria.

- **permissions** è la tabella che contiene la lista di tutti i permessi che può avere un ruolo e di conseguenza un utente. Ha come attributi id, nome e i timestamp di creazione e aggiornamento.
- **role_has_permissions** è la tabella che contiene la lista dei ruoli con i loro permessi associati. Ha come attributi l'id del permesso e l'id del ruolo, entrambi chiavi primarie.
- **model_has_permissions** è la tabella che contiene la lista dei modelli e i loro permessi associati. Ha come attributi l'id del permesso, il tipo del modello e l'id del modello, entrambi chiavi primarie.

3.3 Interazione con il database

Per gestire i dati nel database ho utilizzato Laravel Query Builder, che fornisce un'interfaccia conveniente e veloce per creare ed eseguire le query. Può essere utilizzata per la maggior parte delle operazioni e funziona perfettamente con tutti i sistemi di database supportati da Laravel [5].

Un'altra funzionalità che il framework offre per lavorare sul database e che ho utilizzato ampiamente è Eloquent: il sistema ORM (Object Relational Mapping) di Laravel. Questo sistema è orientato agli oggetti e astrae il database attraverso modelli. Semplifica notevolmente le operazioni di database, rendendo il codice più leggibile, manutenibile e flessibile. I modelli di Eloquent sono classi che rappresentano le tabelle del database, quando viene creato un modello viene stabilita una relazione tra la propria classe PHP e la tabella nel database. Utilizzando i modelli, si possono eseguire operazioni di interrogazione del database come selezionare, aggiornare, inserire e cancellare record senza scrivere query SQL direttamente. I modelli che ho inserito nel codice del progetto sono: 'User' che rappresenta gli utenti della piattaforma ed è collegato alla tabella users, 'Membership', collegato alla tabella memberships che rappresenta i tesseramenti, 'Event' ovvero gli eventi creati, corrispondente alla tabella events, 'Place' è il modello che si riferisce alla tabella places cioè i luoghi creati e infine 'Reservation', ovvero le prenotazioni agli eventi, collegato alla tabella reservations.

I sistemi che ho descritto sono tutti e due utilizzati per eseguire query senza usare direttamente SQL, ma hanno alcune differenze. Eloquent è più leggibile e facile da utilizzare, ma può essere meno flessibile soprattutto per operazioni complesse. Mentre il Query Builder offre un maggiore controllo ma a volte risulta meno leggibile. Quindi la scelta tra i due dipende dalle esigenze specifiche del progetto, anche se è comune utilizzare entrambi gli approcci in combinazione, sfruttando i punti di forza di entrambi, come ho fatto per la maggior parte delle query utilizzate nel codice della piattaforma.

Un esempio di come ho utilizzato Laravel Query Builder è il seguente:

```
1 DB::table('memberships')
2     ->join('users', 'memberships.user_id', '=', 'users.id')
3     ->where('users.code', $request->code)
4     ->whereYear('memberships.created_at', $year)
5     ->exists()
```

Listing 3.1: Query che verifica se esistono tesseramenti (registrati nella tabella "memberships") associati a un utente con un determinato codice fiscale (nella tabella "users") e creati in un anno specifico.

Il prossimo invece è un esempio di come ho utilizzato sia Eloquent che Laravel Query Builder:

```
1 Event::where('confirmed', 1)
2     ->where('event_start', '>', NOW())
3     ->orderBy('event_start', 'asc')
4     ->first();
```

Listing 3.2: Query che recupera l'evento confermato più vicino alla data corrente, dopo aver messo in ordine crescente gli eventi rispetto alla data di inizio

3.4 Seeding del database

Laravel permette di popolare il database con dati utilizzando le classi di seeding [6]. Una volta creata la classe e compilato il metodo `run` inserendo al suo interno i dati desiderati, è sufficiente eseguire il comando `db:seed` da terminale. Questo si occuperà di chiamare il metodo e quindi popolare il database con quei dati. Ho utilizzato questo meccanismo per creare la classe `RoleAndPermissionSeeder` che consente di inserire i ruoli di Amministratore, Operatore e Base nella tabella corrispondente e i vari permessi correlati. Inoltre ho creato lo `UserSeeder` per aggiungere al database un utente con ruolo Amministratore, in questo modo quando l'applicazione viene eseguita per la prima volta, si potrà accedere immediatamente con l'utente admin e iniziare a creare i profili dei volontari.

Capitolo 4

Gestione degli utenti e autenticazione

L'applicazione da me sviluppata ha come fondamento la gestione degli utenti e l'implementazione del meccanismo di autenticazione, ovvero il primo passo che devono effettuare i volontari per accedere alla pagina iniziale del gestionale. Durante il corso del tirocinio, ho posto particolare attenzione a delineare e implementare queste funzionalità iniziali e fondamentali per il funzionamento dell'intera piattaforma. In questo capitolo sono descritte le scelte progettuali che hanno guidato la creazione di una robusta struttura degli utenti e delle modalità di autenticazione all'interno della piattaforma.

4.1 Ruoli utenti e i loro permessi

Per la corretta progettazione della piattaforma, ho dovuto effettuare una divisione degli utenti in base ai ruoli, in particolare ho scelto di creare tre ruoli: Amministratore, Operatore, associato ai volontari che lavorano per l'organizzazione, e Base, associato a coloro che effettuano il tesseramento ma non sono volontari dell'associazione.

Nello specifico l'amministratore possiede maggiori permessi rispetto agli altri tipi di utente: accedendo alla sua pagina principale può creare gli utenti con ruolo di Amministratore o Operatore, infatti i volontari non possono registrarsi in modo individuale alla piattaforma, ma è l'amministratore che deve creare i loro profili. L'admin può inoltre accedere alla lista degli utenti registrati e modificare o eliminare del tutto i profili creati, anche quelli degli utenti base. Oltre a quelli appena descritti, l'amministratore possiede tutti i permessi degli operatori, che elencherò di seguito.

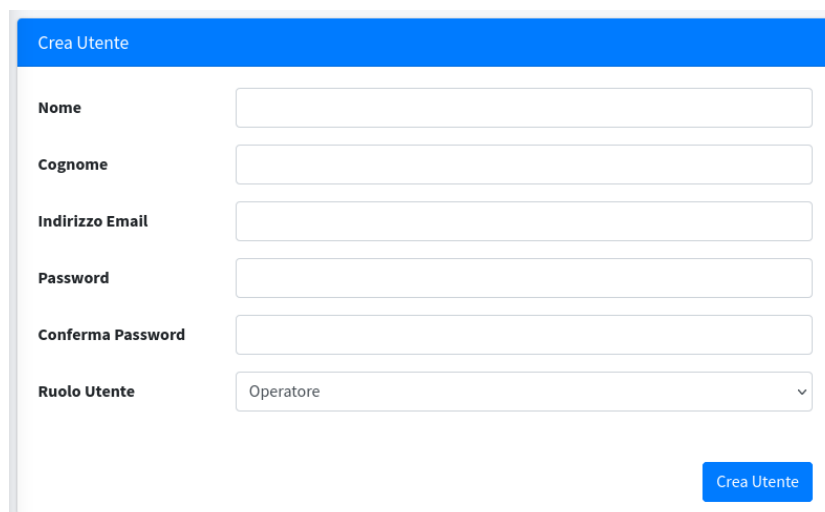
Gli operatori, accedendo alla loro pagina principale possono visualizzare la lista dei tesseramenti confermati e le loro informazioni. Hanno accesso alla lista delle richieste, non confermate, di tesseramento e il corrispondente file

di conferma del pagamento, caricato dall'utente che ha effettuato la richiesta. Da questa lista hanno la possibilità di accettare o rifiutare tali richieste. Per quanto riguarda gli eventi possono accedere al calendario, creare nuovi eventi e modificarli o eliminarli. Possono visualizzare la pagina delle prenotazioni di uno specifico evento, in cui è presente la lista di tutte le prenotazioni e un form per registrarne nuove. Gli operatori hanno inoltre il permesso di creare modificare o eliminare i luoghi, relativi agli eventi, registrati nella piattaforma. Infine hanno accesso allo scanner di codici QR, che utilizzando la fotocamera riesce a leggere i QR e controllare se il tesseramento di una persona è valido oppure no.

Gli utenti con il ruolo Base vengono creati quando viene effettuato e confermato un nuovo tesseramento. Nel caso in cui esista già un utente con quella email nella piattaforma, non ne viene creato uno nuovo ma il tesseramento viene collegato all'utente già esistente, senza modificarne il ruolo. Questo permette ad un operatore o amministratore di effettuare il tesseramento con la stessa email che utilizza per accedere al gestionale senza che venga creato un nuovo record nella tabella degli utenti con le stesse informazioni. Gli utenti base non hanno nessun permesso particolare e possono solamente accedere alla pagina per richiedere un tesseramento.

4.2 Creazione degli utenti

Poiché il compito di creare i profili degli amministratori e degli operatori è responsabilità dell'admin, nella piattaforma non è presente alcuna pagina per la registrazione individuale. L'amministratore deve effettuare il login e dalla sua pagina principale può accedere alla sezione per la creazione degli utenti. Dopo aver compilato e inviato il form, viene aggiunto l'utente appena creato nel database.



Crea Utente	
Nome	<input type="text"/>
Cognome	<input type="text"/>
Indirizzo Email	<input type="text"/>
Password	<input type="password"/>
Conferma Password	<input type="password"/>
Ruolo Utente	<input type="text" value="Operatore"/>
<input type="button" value="Crea Utente"/>	

Figura 4.1: Form che utilizza l'amministratore per creare gli utenti con ruolo Operatore o Amministratore

All'interno del form l'amministratore deve fornire nome, cognome, email, password e ruolo (operatore o amministratore) dell'utente da creare. Cliccando sul tasto 'Crea Utente', viene inviata una richiesta HTTP di tipo POST al server. Una volta ricevuta, viene passata alla route che definisce come la richiesta deve essere gestita, in questo caso viene chiamato il controller UserController e in particolare la funzione userCreate, che contiene la logica per creare un nuovo utente ed aggiungerlo al database, insieme alle sue informazioni.

```
1 /**
2  * Chiama la funzione per la creazione di un nuovo utente
3  * Può essere eseguita solo dall'amministratore
4  */
5 Route::post('/users/create', 'UserController@userCreate')
6     ->middleware('auth')->middleware('can:edit-users');
```

Listing 4.1: Route utilizzata per la creazione degli utenti, porta alla funzione userCreate del controller UserController

4.3 Accesso degli utenti alla piattaforma

Il login può essere effettuato solamente dagli utenti che hanno il ruolo di Operatore o Amministratore. La pagina iniziale dell'applicazione web contiene il form per il login dove si devono inserire la propria email e la password. Dopodiché viene inviata al server una richiesta HTTP di tipo POST e viene chiamata la route per il login, questa porta al LoginController, che verifica se le informazioni inserite nel form sono corrette, in quel caso viene effettuato l'accesso e viene caricata la pagina principale dell'utente, altrimenti compare un messaggio di errore. Questo meccanismo è stato implementato utilizzando il pacchetto *laravel/ui*, che ha automaticamente creato le routes e i controllers necessari per l'autenticazione.

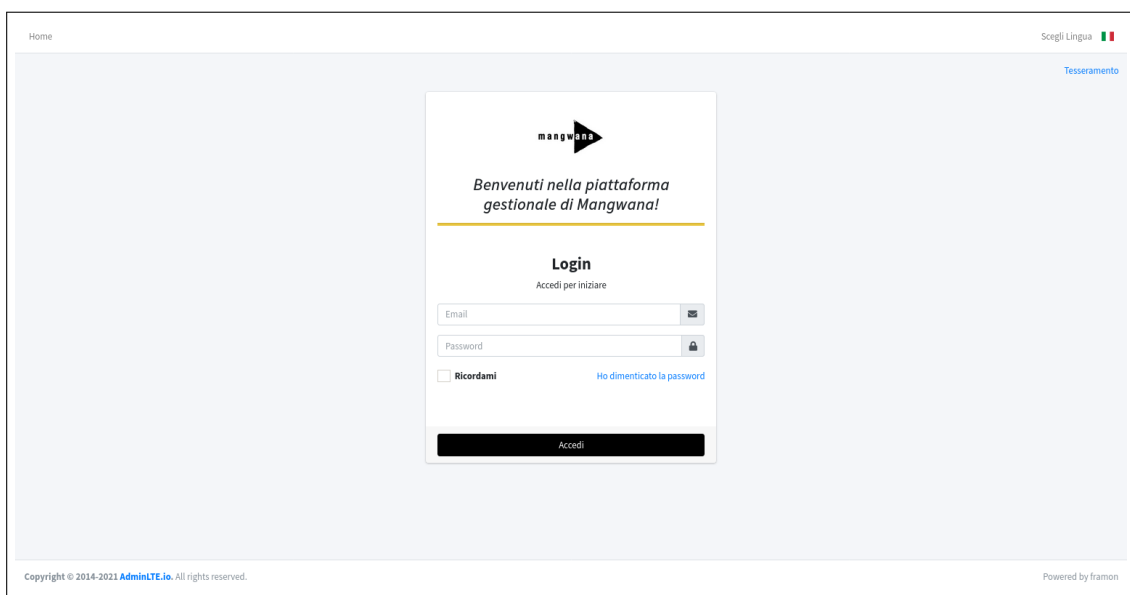


Figura 4.2: Pagina iniziale della piattaforma gestionale, dove i volontari possono effettuare l'accesso

4.4 Modifica informazioni utenti

Nella piattaforma ho inserito la possibilità di modificare le informazioni relative agli utenti registrati. Ci sono due casi d'uso principali: il primo, rappresentato in figura [4.3] in cui un utente, operatore o amministratore, vuole modificare le proprie informazioni (nome, cognome, email e password). Accedendo alla pagina per aggiornare il proprio profilo può compilare il form e dopo averlo inviato, il server riceve una richiesta di tipo POST e agisce di conseguenza chiamando la route e il controller corrispondenti, che si occuperanno di aggiornare e salvare le informazioni modificate nel database.

Figura 4.3: Pagina per l'aggiornamento del proprio profilo utente, in cui si può anche modificare la password

Questo meccanismo è gestito dalle routes in figura 4.2 e dal controller UserController, in particolare la funzione updateUser.

```
1 /**
2  * Route per visualizzare la pagina di aggiornamento del
   profilo utente
3  */
4 Route::get('users/update', function () {
5     return view('auth/update');
6 })->middleware('auth');
7
8 /**
9  * Route per aggiornare il profilo dell'utente autenticato
10 */
11 Route::post('users/update', 'UserController@update')
12     ->middleware('auth')->name('update');
```

Listing 4.2: Routes utilizzate per l'aggiornamento del profilo dell'utente autenticato

Il secondo caso d'uso, in figura [4.4], riguarda solamente l'amministratore, che oltre a poter aggiornare le proprie informazioni, ha la possibilità di modificare quelle degli altri utenti, ad eccezione della password. Infatti accedendo alla pagina per la gestione degli utenti, può selezionare l'operazione di modifica relativa ad uno specifico utente e verrà visualizzato un form, nel quale è possibile anche modificare il ruolo. Se un utente Base viene promosso ad Operatore o Amministratore, nel form comparirà una sezione dove inserire la password, per permettere all'utente di accedere al gestionale. Al contrario se un operatore o amministratore viene retrocesso a Base, allora la sua precedente password viene cancellata.

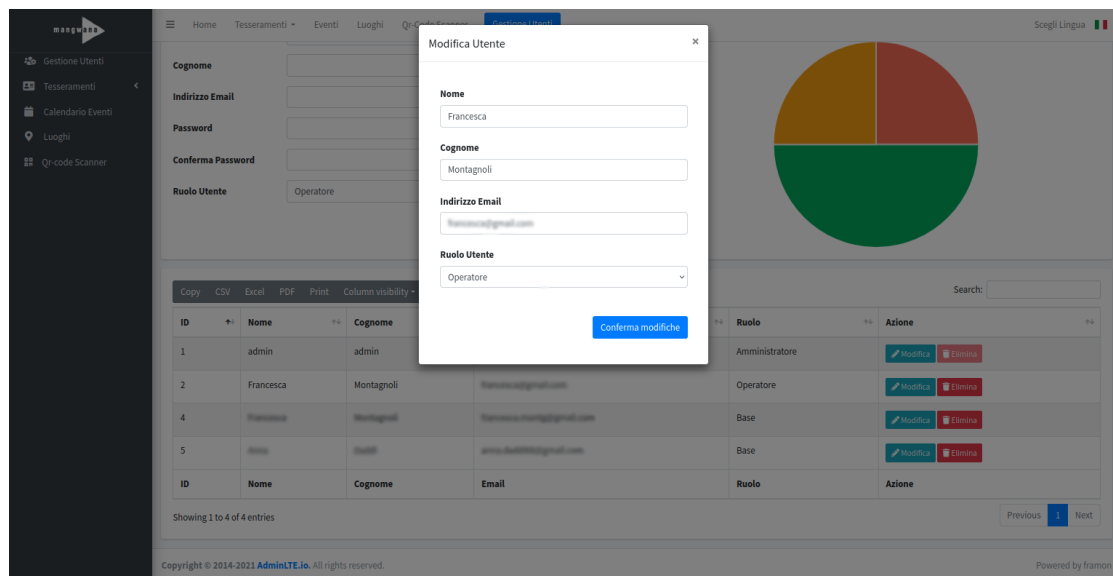


Figura 4.4: Pagina accessibile solo dall'amministratore, per modificare le informazioni degli utenti registrati nella piattaforma

In questo caso la route che si occupa di chiamare la funzione per modificare il profilo di un utente è la seguente, in figura 4.3, che passa la richiesta HTTP alla funzione `userEdit` di `UserController`.

```

1 /**
2  * Route per modificare il profilo di un utente
3  * puo' essere eseguita solo dall'amministratore
4  */
5 Route::post('/users/edit/{id}', 'UserController@userEdit')
6     ->middleware('auth')->middleware('can:edit-users');
```

Listing 4.3: Routes utilizzate per l'aggiornamento del profilo dell'utente autenticato

Capitolo 5

Tesseramento

Nel seguente capitolo, viene descritto un elemento fondamentale all'interno dell'associazione Mangwana: il processo mediante il quale i membri possono ottenere la tessera associativa. Questo documento fornisce loro l'accesso privilegiato a una serie di eventi e servizi organizzati durante l'anno. In questo capitolo, viene analizzato il processo di registrazione, i metodi di pagamento, e come il meccanismo di controllo tramite QR code è stato implementato per migliorare ulteriormente questo processo.

5.1 Criticità del vecchio sistema e soluzione progettata

Mangwana, essendo un'associazione di promozione sociale, offre la possibilità di acquistare la tessera associativa, con la quale si può partecipare a tutti gli eventi organizzati durante l'anno solare. Questo meccanismo veniva gestito attraverso un form di Google che l'utente poteva compilare inserendo le proprie informazioni e, dopo aver scelto il metodo di pagamento tra PayPal, Satispay o bonifico IBAN, doveva versare la quota associativa di €5.00 nella piattaforma scelta e caricare nel form un file contenente la conferma di avvenuto pagamento. Per i volontari risultava un lavoro eccessivamente laborioso e suscettibile ad errori, poiché una volta ricevuti i form compilati, erano obbligati a controllare se i pagamenti fossero stati ricevuti correttamente, e in caso positivo aggiungere gli utenti manualmente ad una lista, per tenere traccia dei tesseramenti. Un altro problema era la verifica dei tesseramenti per l'ammissione agli eventi. In questo caso i volontari dovevano consultare la lista degli utenti tesserati nell'anno corrente ed in base a quello ammetterli oppure no. Per questi motivi ho progettato un nuovo sistema per il tesseramento, che fosse automatizzato e rendesse il lavoro degli operatori molto più facile e veloce. Ho creato una pagina web, inserendo al suo interno un form HTML, che l'utente deve compilare. Ho mantenuto gli stessi metodi di pagamento già utilizzati dall'associazione, ma integrando PayPal e Satispay all'interno della

piattaforma in modo che quando l'utente sceglie uno dei due metodi viene reindirizzato alla pagina corrispondente al servizio per poter completare il pagamento. L'applicazione controlla se il pagamento è andato a buon fine e in quel caso aggiunge automaticamente l'utente alla tabella dei tesseramenti nel database. Per quanto riguarda il pagamento tramite bonifico con IBAN, come già accadeva, l'utente deve caricare un file che attesti l'avvenuto pagamento, dopodiché la richiesta di tesseramento viene inserita in una lista, a cui i volontari possono accedere per accettarla o rifiutarla.

Oltre al tesseramento online, è possibile richiedere la tessera anche durante gli eventi organizzati dall'associazione, pagando con contanti o carta. Per questo motivo ho aggiunto alla pagina degli operatori una sezione dove poter inserire i tesseramenti manualmente attraverso un form.

Per quanto riguarda la verifica del tesseramento per l'accesso agli eventi, ho aggiunto un meccanismo di controllo tramite QR code, con lo scopo di migliorare e velocizzare il sistema precedente. Infatti, una volta che un utente ha effettuato il pagamento per la tessera associativa e questa è stata confermata, viene inviata una email contenente un codice QR, che l'utente dovrà presentare agli eventi a cui desidera partecipare. I volontari attraverso la piattaforma, hanno accesso ad uno scanner per i QR, con il quale possono controllare se i tesseramenti sono validi oppure no. In sostanza, questa soluzione agevola il controllo dell'idoneità dei partecipanti all'evento, semplificando il processo e garantendo una migliore esperienza sia per i volontari che per gli utenti.

5.2 Compilazione del form e validazione

Tesseramento online

La pagina web per poter effettuare il tesseramento è aperta a tutti i tipi di utenti, anche a quelli non registrati nella piattaforma. L'utente è tenuto ad inserire le proprie informazioni personali nel modulo disponibile sulla pagina. Queste informazioni includono il nome, il cognome, l'indirizzo email e il codice fiscale. La raccolta di questi dettagli è fondamentale per garantire che il tesseramento sia associato all'utente corretto. Prima di procedere l'utente deve accettare i termini di servizio, dopodiché può selezionare il metodo di pagamento preferito tra PayPal, Satispay o bonifico bancario tramite IBAN. Per garantire la massima sicurezza durante il processo di registrazione, è stato integrato un reCAPTCHA v2 di Google all'interno del modulo. Questa misura serve a prevenire l'accesso automatizzato o non autorizzato e garantisce che le adesioni siano legittime. Nel caso l'utente scelga PayPal o Satispay come metodo di pagamento, il sistema lo reindirizza automaticamente al sito di pagamento corrispondente. Una volta completata la transazione, le informazioni inserite nel form vengono salvate nel database e, se l'utente non era ancora registrato nella piattaforma, viene inserito un nuovo record nella tabella 'users'. Nel caso in cui l'utente scelga l'opzione di pagamento tramite

bonifico bancario, dopo aver caricato un file con la conferma del pagamento, viene generata una richiesta di tesseramento. Gli operatori e gli amministratori hanno il compito di accettare o rifiutare tale richiesta accedendo alla sezione corrispondente nella piattaforma. In figura [5.2] è rappresentato il diagramma di attività relativo al tesseramento online, in cui si può notare la differenza della scelta tra il pagamento con Satispay o PayPal e quello con bonifico IBAN.

The screenshot shows the 'Tesseramento Mangwana APS' web form. At the top, there is a navigation bar with 'Home' on the left and 'Scegli Lingua' with an Italian flag on the right. A 'Tesseramento' link is also visible in the top right. The main content area features the Mangwana logo (a yellow arrow pointing right with the word 'mangwana' inside) and the title 'Tesseramento Mangwana APS'. Below this, there are four input fields: 'Nome', 'Cognome', 'Indirizzo Email', and 'Codice Fiscale'. A 'Privacy Policy' section follows, containing a detailed text about data protection and a right arrow icon. Below the privacy policy is a CAPTCHA section with a checkbox labeled 'Non sono un robot' and a reCAPTCHA logo. At the bottom, there is a section titled 'Scegliere il metodo di pagamento:' with three radio button options: 'PayPal', 'Satispay', and 'IBAN'. The footer contains the copyright notice 'Copyright © 2014-2021 AdminLTE.io. All rights reserved.' and the text 'Powered by framon'.

Figura 5.1: Pagina per il tesseramento online, accessibile a tutti i tipi di utente

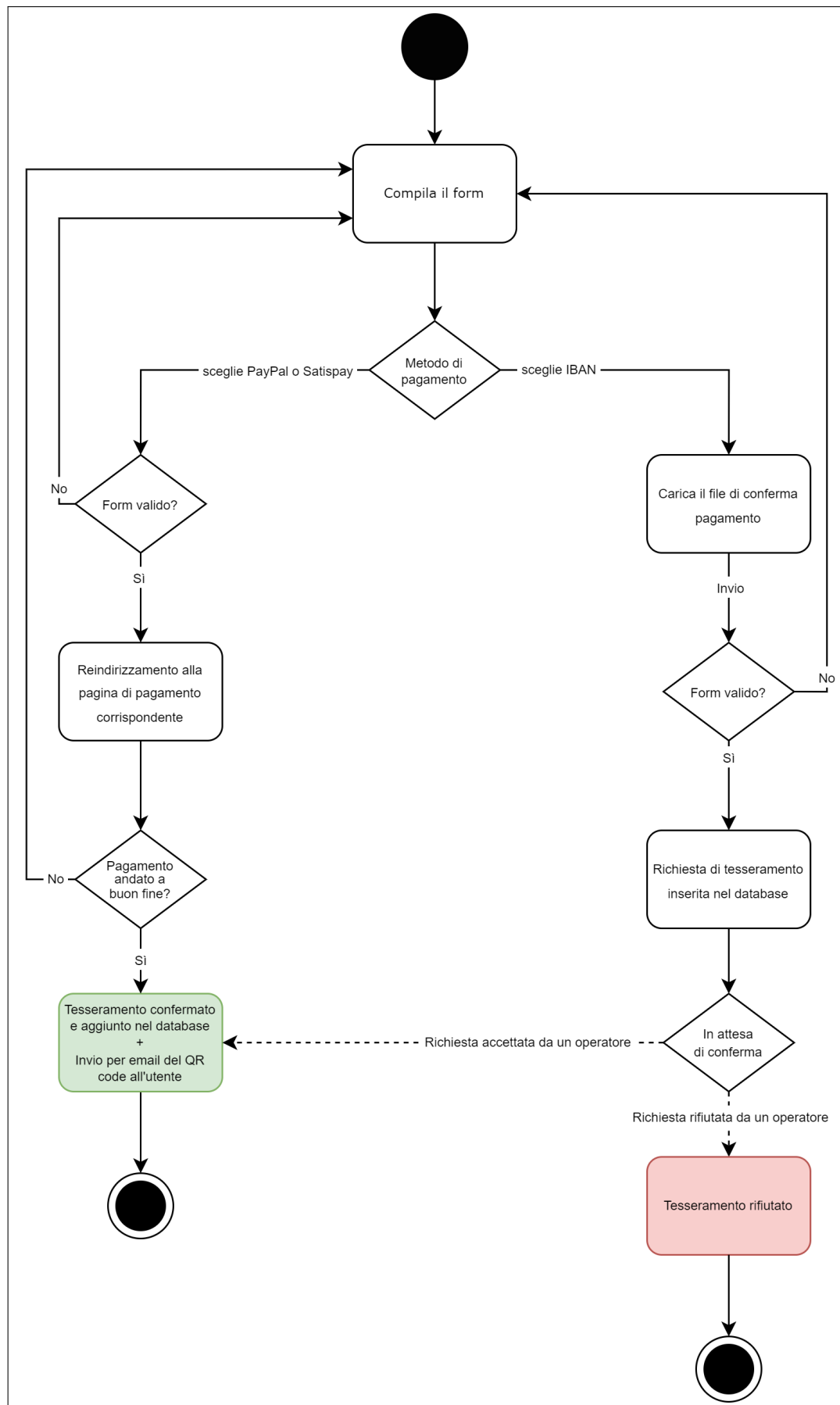
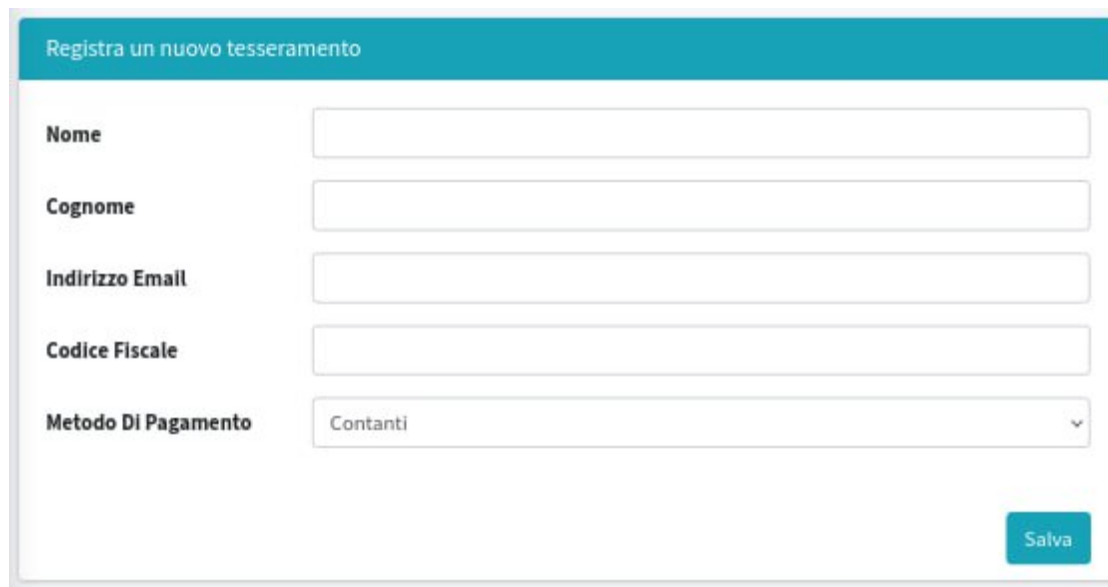


Figura 5.2: Diagramma di attività del tesseramento online

Tesseramento in loco

Oltre all'opzione di tesseramento online, gli utenti hanno la possibilità di richiedere la tessera associativa direttamente durante la partecipazione a un evento in loco. In questo scenario, i volontari dispongono di un'apposita funzionalità sulla piattaforma che consente loro di registrare tali richieste di tesseramento in modo efficiente. Il modulo da compilare raccoglie le stesse informazioni richieste nel processo di tesseramento online. Tuttavia, in questa situazione, i termini di servizio e il reCAPTCHA di Google non sono inclusi, poiché i controlli aggiuntivi non sono necessari. Anche per questo caso d'uso ho creato un diagramma di attività, rappresentato in figura [5.4].



The image shows a web form titled "Registra un nuovo tesseramento" (Register a new membership). It contains five input fields: "Nome" (Name), "Cognome" (Surname), "Indirizzo Email" (Email Address), "Codice Fiscale" (Tax Code), and "Metodo Di Pagamento" (Payment Method). The "Metodo Di Pagamento" field is a dropdown menu currently showing "Contanti" (Cash). A blue "Salva" (Save) button is located at the bottom right of the form.

Figura 5.3: Form per il tesseramento in loco, accessibile dagli operatori e amministratori

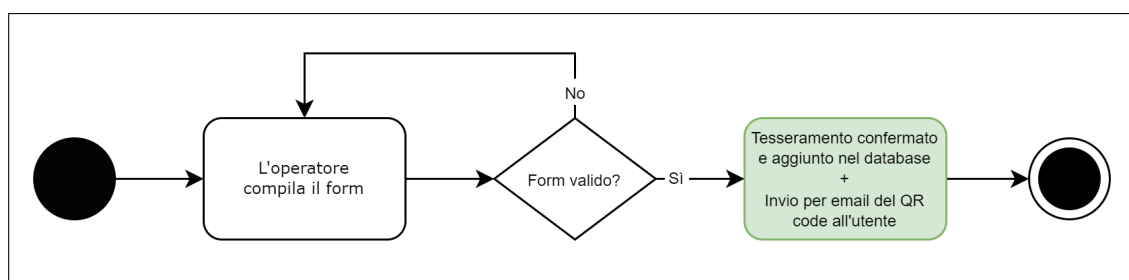


Figura 5.4: Diagramma di attività del tesseramento in loco

5.3 Pagamento

Per quanto riguarda il tesseramento online, l'utente può scegliere di utilizzare uno dei seguenti metodi di pagamento: PayPal, Satispay o bonifico tramite IBAN.

- **PayPal:** L'utente sarà reindirizzato alla pagina web di PayPal, dove potrà accedere al suo account ed effettuare il pagamento. Per implementare questo metodo ho utilizzato il pacchetto *srmlive/laravel-paypal*. Ho creato il controller `PayPalController`, contenente il codice che si occupa di reindirizzare l'utente alla pagina di PayPal, creare il pagamento e controllare che questo sia andato a buon fine. Se il pagamento viene ricevuto correttamente, il tesseramento viene confermato, mettendo ad 1 la flag `confirmed` nel record corrispondente della tabella `memberships` del database. Se l'utente non è già registrato nel database, verrà creato un nuovo profilo con ruolo `Base`. In caso contrario, il tesseramento sarà associato al record esistente nel database che condivide l'indirizzo email fornito durante la procedura di tesseramento.
- **Satispay:** Per implementare il pagamento tramite Satispay, ho integrato nel codice la sua API, seguendo la relativa documentazione. In particolare ho scelto di utilizzare il flusso di pagamento `Web Redirect`, in cui l'utente viene reindirizzato alla pagina web di Satispay, dove verrà visualizzato un QR code corrispondente al pagamento da effettuare. L'utente potrà quindi accedere dal suo cellulare all'applicazione di Satispay, scansionare il codice e concludere il pagamento.
L'immagine in figura [5.5] rappresenta il diagramma di sequenza del flusso `Web Redirect`.

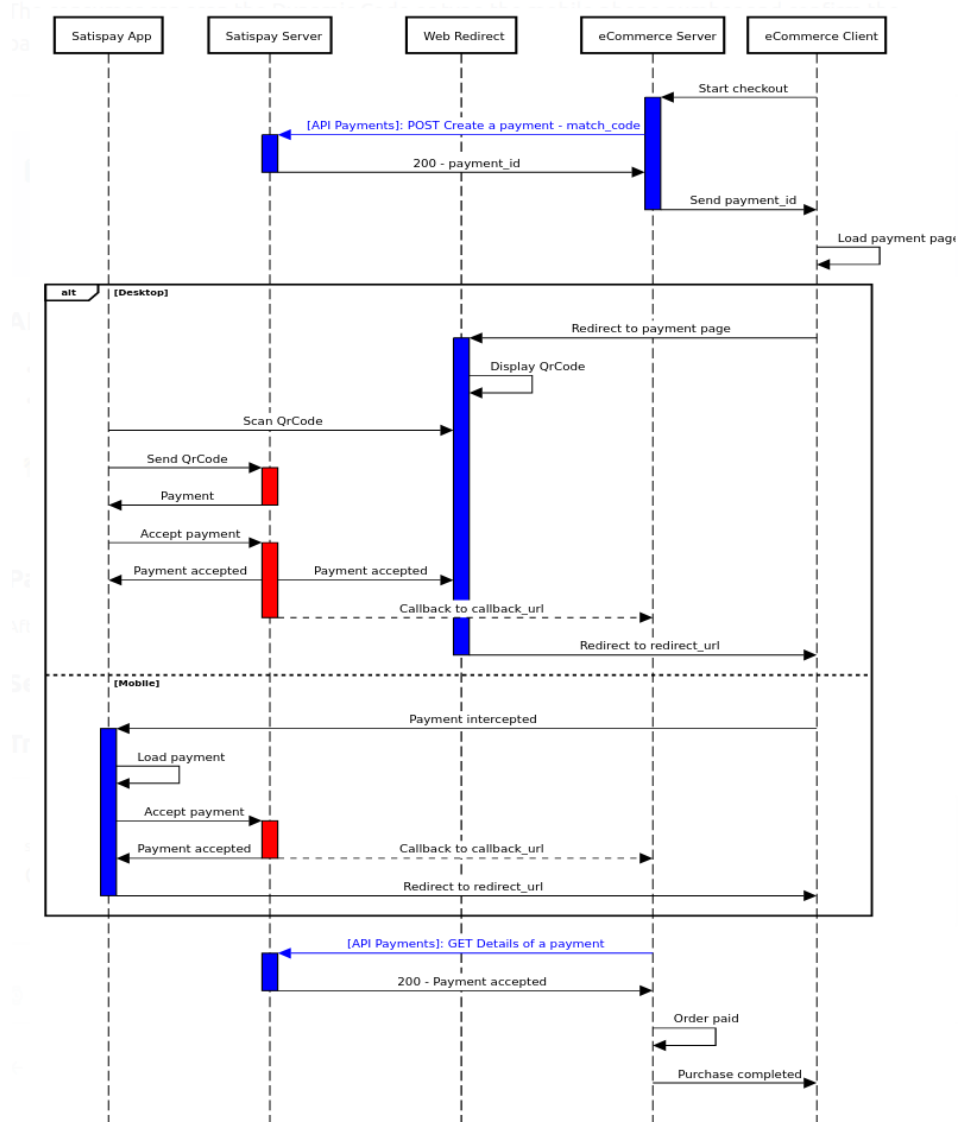


Figura 5.5: Diagramma di sequenza del meccanismo Web Redirect per il pagamento con Satispay, immagine presa da [12]

L'integrazione dell'API è contenuta nel controller SatispayController, che si occupa di creare e confermare il tesseramento e registrare un nuovo utente nella piattaforma se già non esisteva.

- **IBAN:** Scegliendo questo metodo di pagamento, l'utente deve caricare un file di conferma di avvenuto pagamento nel form per il tesseramento. In questo caso gli operatori o gli amministratori possono accedere alla lista delle richieste di tesseramento (in figura [5.6]), dove possono visualizzare il file caricato dall'utente. Una volta controllato se il pagamento è stato effettivamente ricevuto, i volontari possono accettare o rifiutare la richiesta. Se viene accettata, il tesseramento viene confermato e di conseguenza viene creato l'utente, se già non esisteva. Se viene rifiutata, le informazioni del tesseramento presenti nel database vengono eliminate. I file caricati dagli utenti nel form vengono salvati

in una cartella, all'interno della directory principale del progetto, chiamata 'pagamenti_da_confermare'. Il nome dei file viene modificato seguendo questo formato: 'membershipId+Cognome+Nome', questo permette una migliore organizzazione e identificazione dei file. Se la richiesta viene accettata il file viene spostato in una nuova cartella, chiamata 'pagamenti_confermati/annoTesseramento/meseTesseramento', altrimenti il file viene eliminato. In questo modo i file sono facilmente reperibili anche al di fuori della piattaforma web.

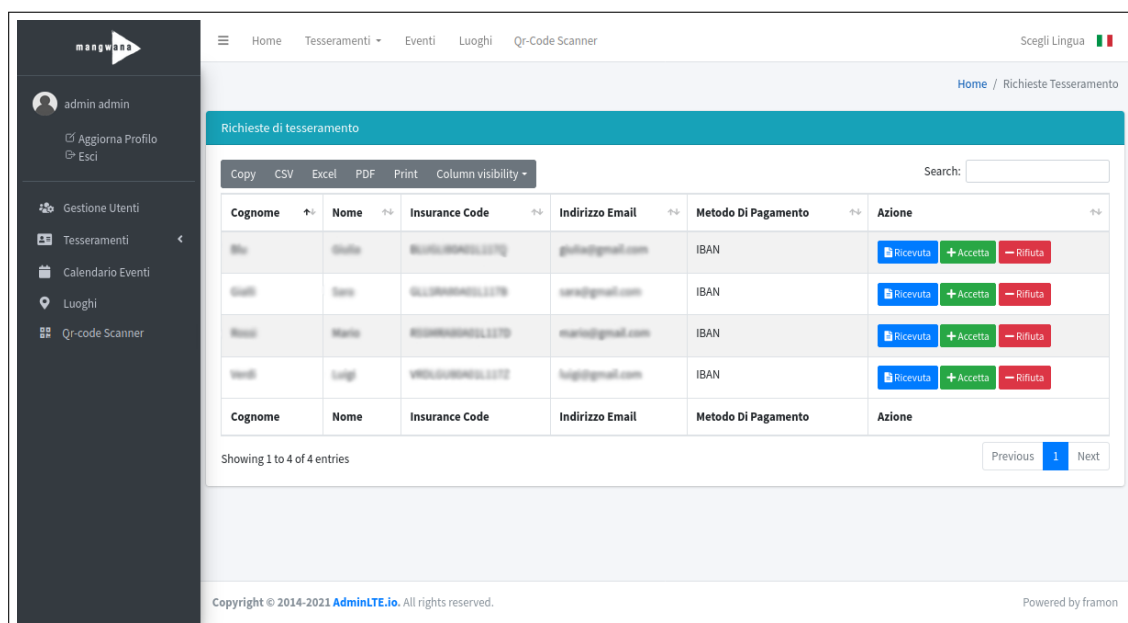


Figura 5.6: Pagina delle richieste di tesseramento

5.4 Invio del QR Code tramite email

Una volta che un tesseramento viene confermato, cioè quando la flag 'confirmed' nella tabella 'memberships' viene impostata a 1, l'utente che ha effettuato il tesseramento, riceve una email contenente un codice QR. Questo codice servirà al cliente quando vorrà partecipare agli eventi organizzati dall'associazione, infatti i volontari sempre tramite la piattaforma possono scansionare il codice e verificare se il tesseramento dell'utente è valido oppure no. Per generare il codice QR ho utilizzato il pacchetto *Simple QrCode* di Simple Software, con il quale è sufficiente una chiamata di funzione per ottenere il risultato desiderato. Il codice include un link alla pagina contenente le informazioni del tesseramento dell'utente. L'url in questione ha questa forma: 'membership-info/membershipId', dove membershipId è l'id del tesseramento che si trova nella tabella memberships del database. Se il tesseramento non esiste o non è stato confermato, la scansione del codice QR restituirà come risposta 'Tesseramento non valido'.

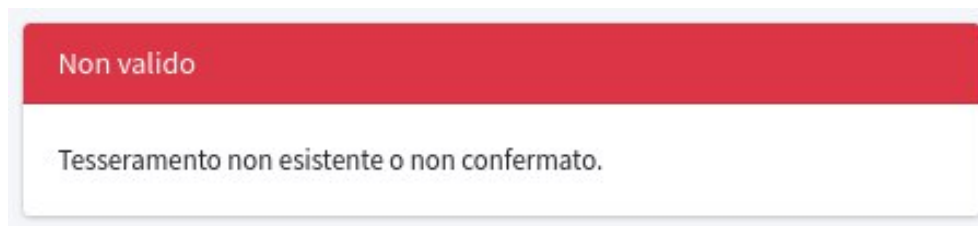


Figura 5.7: Cosa appare dopo aver scansionato un QR code non valido

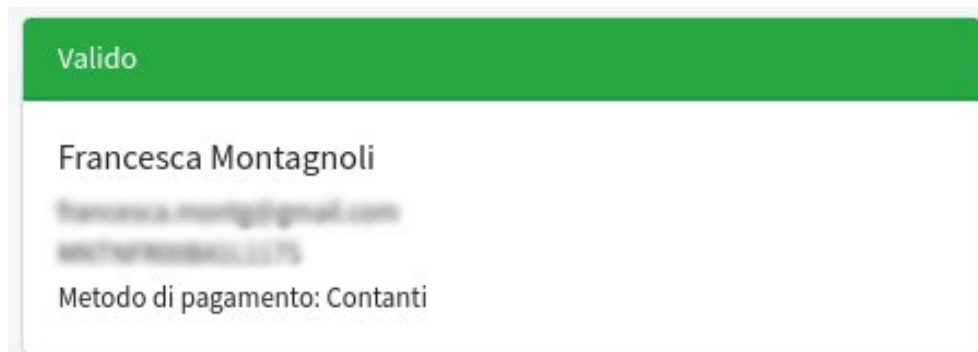


Figura 5.8: Cosa appare dopo aver scansionato un QR code valido

Per implementare lo scanner per i codici QR ho utilizzato la libreria *html5-qrcode*, inserendo nella view corrispondente gli script necessari per integrarlo. In questo modo, quando un volontario accede alla pagina contenente lo scanner, può attivare la fotocamera della macchina su cui viene eseguita l'applicazione web, e iniziare a scansionare. Ho inoltre implementato una funzione JavaScript che consente l'apertura immediata del link rilevato in una nuova scheda del browser una volta che il codice è stato letto. Questa caratteristica migliora notevolmente l'esperienza dell'utente, rendendo il processo di accesso alle informazioni del tesseramento più rapido.

Per la gestione dell'invio delle email inizialmente ho sperimentato l'utilizzo di Mailgun, un servizio di invio e ricezione email supportato da Laravel e consigliato dalla documentazione [9]. Tuttavia, durante una serie di test, ho notato che alcune caselle di posta elettronica non ricevevano le email o le classificavano come spam. A seguito di queste considerazioni, ho optato per Gmail come soluzione alternativa. Gmail offre una maggiore affidabilità nel garantire che le email inviate raggiungano correttamente le caselle di posta dei destinatari e hanno minori probabilità di essere erroneamente etichettate come spam. Per le email ho creato un template HTML personalizzato, all'interno del quale è stato integrato il codice QR precedentemente generato.

Capitolo 6

Eventi e luoghi

L'attività principale dell'associazione Mangwana è quella di organizzare eventi. Pertanto, è stato essenziale progettare ed implementare una sezione della piattaforma dedicata a semplificare il processo di pianificazione degli eventi per i volontari dell'associazione. In questo capitolo vengono presentate le metodologie e le strategie che ho adottato per sviluppare questa sezione chiave della piattaforma.

6.1 Ideazione del calendario

Fino a questo momento, i volontari si coordinavano per l'organizzazione degli eventi utilizzando Google Calendar. Dopo aver creato un evento per una data specifica, a seconda della disponibilità dei volontari, l'evento veniva effettivamente confermato e organizzato oppure annullato. Il processo di sviluppo è partito cercando di capire a fondo le necessità dei volontari. Prima di tutto, considerando che l'associazione aveva fino a quel momento utilizzato Google Calendar, ho deciso di integrare nella piattaforma un calendario, dove poter inserire gli eventi da organizzare, che fosse simile nell'interfaccia a quello di Google, così da rendere il passaggio alla piattaforma più agevole. Ho voluto però aggiungere delle funzionalità in più, che potessero essere utili all'associazione. Per questo motivo ho pensato di dividere gli eventi in due tipi, quelli confermati e quelli da confermare. Gli eventi necessitano infatti di un numero minimo di volontari che partecipino in quella giornata. In base a questo, quando si raggiunge il numero necessario di volontari disponibili, l'evento potrà essere confermato. Un evento quando viene creato per la prima volta ha lo stato di 'non confermato' e nel calendario appare di colore rosso. Se viene confermato diventerà di colore verde. Questa distinzione visiva rende più leggibile il calendario, e quindi rende anche più veloce l'organizzazione degli eventi. Nel calendario ho inoltre inserito diversi filtri, infatti si potranno visualizzare gli eventi in base al luogo in cui si svolgono, oppure si potrà scegliere di visualizzare solo gli eventi confermati o solo quelli non confermati.

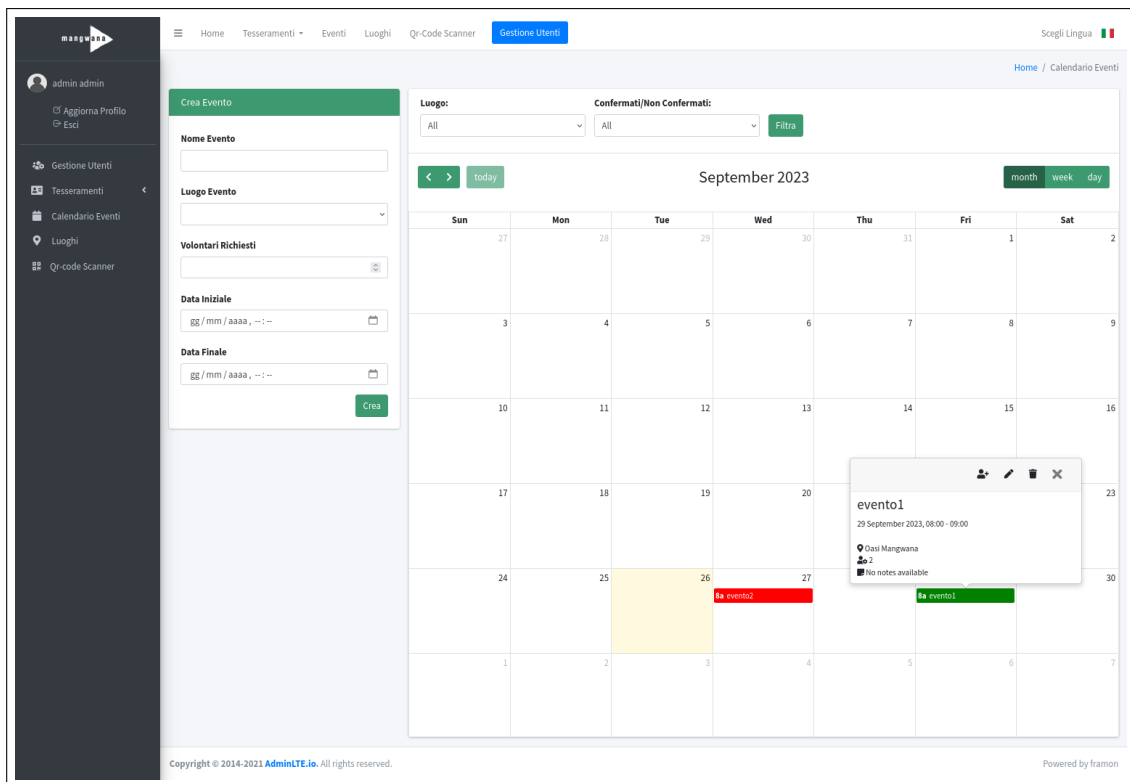


Figura 6.1: Pagina del calendario, per visualizzare e creare gli eventi

6.2 Implementazione del calendario

Per implementare il calendario nella piattaforma, ho fatto uso di FullCalendar, una libreria JavaScript open-source ampiamente utilizzata per la creazione e la gestione di calendari interattivi su pagine web (descritta in Sezione 2.4.2). Questa libreria offre una vasta gamma di funzionalità per la visualizzazione e la gestione degli eventi in formato di calendario, quindi mi è sembrata la scelta migliore. Il calendario si presenta con una vista al mese corrente, ma si può facilmente navigare tra i vari mesi e si può inoltre cambiare alla vista settimanale o giornaliera. Accanto al calendario ho inserito in HTML, un form per la creazione degli eventi, nel quale i volontari possono scegliere il nome dell'evento, la data di inizio e di fine, il luogo dove si svolgerà e il numero di volontari necessari per poter confermare l'evento. Come già detto in precedenza, un evento appena creato si presenterà di colore rosso nel calendario, e nel database avrà la flag confirmed con valore 0 poiché si tratta di un evento non ancora confermato. L'associazione potrà, in base alla disponibilità degli operatori, accedere alla pagina di modifica dell'evento e confermarlo, in questo modo la flag confirmed nel database passerà a 1 e l'evento verrà visualizzato di colore verde. Dalla stessa pagina si potranno inoltre modificare le informazioni dell'evento, nel caso in cui ci sia stato un errore alla creazione. Ovviamente sarà anche possibile per i volontari eliminare gli eventi, nel caso in cui si decidesse di non poterli organizzare. Tutte queste operazioni che si possono svolgere attraverso il calendario, sono gestite dal CalendarControl-

ler, che si occupa anche di gestire le prenotazioni degli eventi, descritte nella sezione seguente.

6.3 Prenotazioni eventi

Una funzionalità che ho deciso di aggiungere rispetto a Google Calendar è la capacità, all'interno del calendario stesso, di accedere alla pagina delle prenotazioni relative ad un evento. I volontari, qualora ricevessero una prenotazione per un evento specifico, possono inserirla nella piattaforma per monitorarla. Le informazioni che possono essere aggiunte a una prenotazione includono il nome e il cognome del prenotante, il numero di telefono (opzionale) e il numero di partecipanti. La gestione delle prenotazioni permette ai volontari di organizzare con precisione gli eventi e di avere conoscenza del numero di partecipanti attesi. All'interno della pagina è disponibile l'elenco delle prenotazioni ricevute, che ha anche il ruolo di lista di controllo. Infatti durante lo svolgimento di un evento, i volontari possono segnare la presenza dei partecipanti direttamente attraverso la piattaforma, eliminando la necessità di utilizzare carta e penna come avveniva in passato.

The screenshot displays a web application interface for event management. On the left is a dark sidebar with navigation links: 'admin admin', 'Aggiorna Profilo', 'Esci', 'Gestione Utenti', 'Tesseramenti', 'Calendario Eventi', 'Luoghi', and 'Qr-code Scanner'. The main content area has a top navigation bar with 'Home', 'Tesseramenti', 'Eventi', 'Luoghi', 'Qr-code Scanner', and 'Gestione Utenti'. Below this, the 'evento1' page shows event details: 'Data' (29 September 2023, 08:00 - 09:00), 'Luogo' (Oasi Mangwana), 'Numero totale di partecipanti' (14), and 'Numero di partecipanti presenti' (5). A green button 'Registra una nuova prenotazione' is visible. Below the details is a table titled 'Prenotazioni' with columns: 'Cognome', 'Nome', 'Numero di telefono', 'Numero di partecipanti', and 'Azione'. The table contains three entries. The first entry has a red 'X' icon in the 'Azione' column. The second and third entries have green checkmark icons. At the bottom of the table, it says 'Showing 1 to 3 of 3 entries'. The footer includes 'Copyright © 2014-2021 AdminLTE.io. All rights reserved.' and 'Powered by framon'.

Cognome	Nome	Numero di telefono	Numero di partecipanti	Azione
Smith	John		5	✖ Modifica Elimina
Wong	Maria		4	✓ Modifica Elimina
Wong	Luigi		5	✓ Modifica Elimina

Figura 6.2: Pagina per tenere traccia delle prenotazioni ad un evento

Per implementare la tabella delle prenotazioni come una lista di controllo, ho apportato una modifica alla struttura del database. In particolare, ho aggiunto un nuovo campo chiamato 'attendance' alla tabella 'reservations'. Inizialmente, quando una nuova prenotazione viene registrata nel sistema, questo campo viene impostato a 0. Durante lo svolgimento di un evento, i volontari hanno la possibilità di segnalare la presenza di una persona che ha prenotato. Questo viene fatto cliccando su un pulsante verde con una spunta accanto al nome della persona nella lista. In seguito a questa azione, la

riga corrispondente cambierà colore diventando grigia, e il valore della flag ‘attendance’ verrà modificato da 0 a 1, indicando che la persona è presente all’evento. Allo stesso modo, i volontari possono anche rimuovere la presenza di una prenotazione. Questa operazione avviene facendo clic su un pulsante rosso con una ‘x’ accanto al nome della persona. In questo caso, la riga tornerà al suo colore originario, e la flag ‘attendance’ verrà reimpostata a 0. Inizialmente, quando un operatore desiderava aggiungere o togliere la presenza di una prenotazione, il sistema utilizzava richieste sincrone al server. Ciò significava che ogni volta che veniva effettuata un’operazione di questo tipo, la pagina veniva completamente ricaricata per aggiornare i dati nella tabella delle prenotazioni. Tuttavia, ho ritenuto opportuno migliorare questa esperienza introducendo un sistema basato su richieste asincrone. L’obiettivo era evitare il ricaricamento completo della pagina ad ogni modifica, migliorando così la fluidità e la reattività dell’applicazione. A tale scopo, ho implementato l’uso delle richieste AJAX (Asynchronous JavaScript and XML), che si basano su uno scambio di dati in background fra web browser e server, consentendo così l’aggiornamento dinamico di una pagina web senza esplicito ricaricamento da parte dell’utente [1].

6.4 Luoghi

Una sezione, strettamente legata agli eventi, è quella relativa ai luoghi. Infatti quando si crea un evento, è necessario specificare il luogo in cui si svolgerà scegliendo dall’elenco di quelli registrati. Per aggiungere un nuovo luogo, basta accedere alla sezione apposita e cercare l’indirizzo richiesto sulla mappa di OpenStreetMap integrata nella pagina. Una volta inserito l’indirizzo è possibile spostare il marker nella posizione desiderata e salvare il luogo, che verrà registrato nel database con le sue informazioni ovvero nome, latitudine e longitudine.

Per quanto riguarda i dettagli dell’implementazione, la mappa è stata integrata utilizzando una libreria JavaScript chiamata *Leaflet* ed è stata configurata per visualizzare inizialmente la città di Pisa, che è il luogo principale in cui l’associazione organizza gli eventi. Per abilitare la ricerca tramite indirizzo ho utilizzato il plugin *leaflet-control-geocoder*. Quando gli utenti cercano un indirizzo, il sistema posiziona automaticamente un marker sulla mappa, che può essere spostato nella posizione desiderata. Ogni volta che il marker viene spostato vengono registrate la latitudine e la longitudine del punto preciso in cui si trova, così da salvare nel database la posizione corretta del luogo.

Nella stessa pagina, rappresentata in figura [6.3], è possibile consultare la lista dei luoghi già registrati, visualizzarli sulla mappa, o modificarne il nome.

Questa funzionalità è stata implementata considerando che l’associazione ha solitamente luoghi prestabiliti dove organizza i propri eventi. Tale approccio agevola notevolmente la gestione e il monitoraggio dei vari luoghi, permetten-

domi inoltre di aggiungere il filtro basato sui luoghi nel calendario. In questo modo, gli operatori possono rapidamente visualizzare tutti gli eventi relativi a uno specifico luogo, migliorando l'efficienza nell'organizzazione.

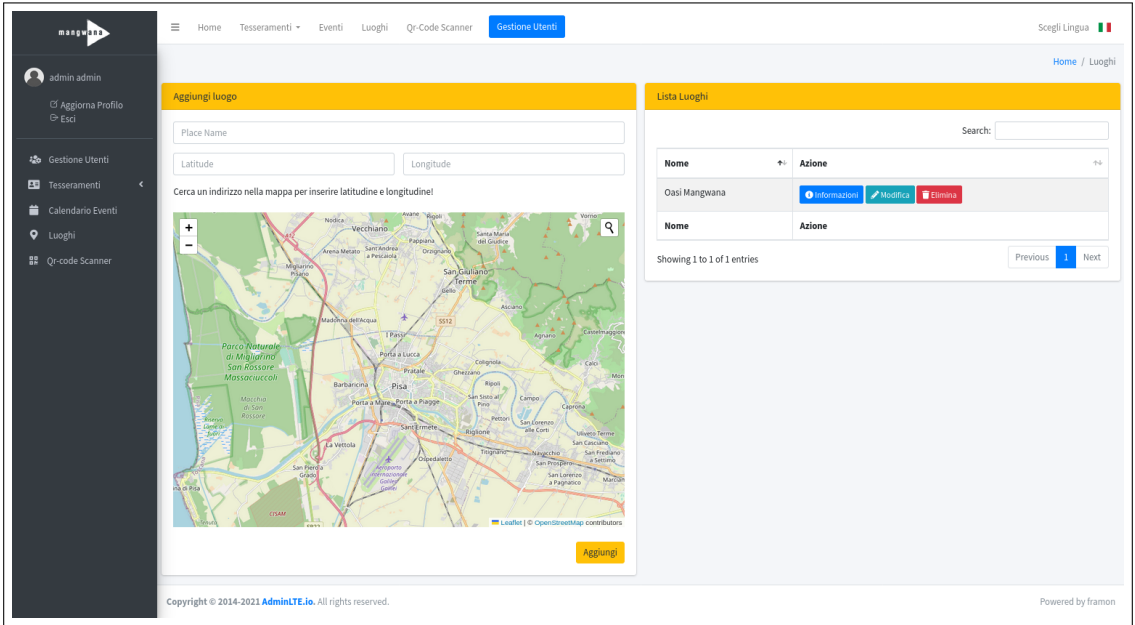


Figura 6.3: Pagina relativa ai luoghi, dove si può aggiungere, modificare o visualizzare uno specifico luogo

Capitolo 7

Esperienza Utente

La User Experience (UX), ovvero l'esperienza utente, ha un ruolo centrale nell'ambito dello sviluppo di qualsiasi piattaforma web o applicazione. Questo capitolo si concentra sull'analisi di come ho posto particolare attenzione al comfort, all'efficienza e alla soddisfazione degli utenti all'interno della piattaforma per la gestione delle attività dell'Associazione Mangwana.

7.1 Introduzione alla UX

La UX si riferisce alla percezione che un individuo ha quando interagisce con un prodotto, un servizio, un sito web o un'applicazione. Prende in considerazione tutti gli aspetti dell'interazione uomo-macchina, compresi aspetti pratici, emotivi ed estetici, al fine di creare un'esperienza soddisfacente e positiva per l'utente. Il processo di design della user experience consiste nell'assicurarsi che nessun aspetto dell'esperienza che l'utente ha con il prodotto o servizio avvenga senza un intento esplicito. Questo significa considerare tutte le possibili azioni che l'utente potrebbe compiere e comprendere le aspettative dell'utente in ogni step di questo processo [16]. Frank Guo, un noto UX architect, nel 2012 ha scritto un articolo in cui definisce la User Experience dividendola in quattro elementi fondamentali [11]:

- **Usabilità:** riguarda quanto facilmente gli utenti possono completare i task desiderati utilizzando un prodotto. L'usabilità è un concetto ampio, si basa sull'apprendimento, la reperibilità, la leggibilità dei contenuti e la facilità con cui gli utenti possono riconoscere le informazioni.
- **Valore:** è la misura di quanto il prodotto sia utile per l'utente, è la corrispondenza tra le funzionalità del prodotto e le necessità dell'utente.
- **Adottabilità:** si riferisce all'acquisto, al download e a quanto il prodotto sia efficace nel motivare l'utente ad iniziare ad utilizzarlo.

- **Desiderabilità:** ovvero quanto il prodotto è coinvolgente, è strettamente legata alle emozioni che l'utente prova quando lo utilizza.

7.2 Progettazione della UX

Il primo passo per progettare un'esperienza utente coinvolgente e di successo, riguarda la ricerca approfondita degli utenti e la definizione delle personas, ovvero i profili di utenti tipici. È necessario comprendere chi sono gli utenti che useranno il prodotto e quali sono i loro bisogni. Per quanto riguarda la parte gestionale della piattaforma gli utenti sono i volontari dell'associazione, mentre nella sezione per il tesseramento, che è aperta a tutti, gli utenti sono le persone interessate ad acquistare la tessera associativa di Mangwana. Dopo aver identificato il tipo di utenti che utilizzeranno la piattaforma, bisogna individuare i punti chiave del prodotto e concentrarsi su di essi per ottimizzare l'esperienza.

Le funzionalità chiave della piattaforma gestionale, sono la gestione dei tesseramenti, e in particolare le richieste di tesseramento, che devono essere accettate o rifiutate dai volontari. La gestione degli eventi, quindi la sezione del calendario e tutte le sue funzionalità. Per rendere la piattaforma facile da utilizzare ed immediata, ho progettato una pagina iniziale in modo tale che si potesse accedere a tutti i funzionamenti fondamentali da essa.

Pagina iniziale del gestionale

Come prima cosa mi sono concentrata sul progettare la schermata iniziale, che i volontari visualizzano subito dopo aver effettuato l'accesso nella piattaforma. Ho diviso la pagina in tre colonne di diverso colore, corrispondenti alle tre principali funzionalità del gestionale.

Come si può vedere in figura [7.1] la prima colonna è dedicata ai **tesseramenti**. Dalla card più in alto è possibile accedere alla pagina per la loro gestione, che contiene il form per la registrazione di nuovi tesseramenti durante gli eventi, e la tabella con la lista di tutti i tesseramenti confermati e le loro informazioni. La card successiva funge da segnalazione per i volontari riguardo al numero di richieste di tesseramento in sospeso, le quali necessitano ancora di verifica. Cliccando su questa card, si accede alla pagina contenente l'elenco completo delle richieste ancora da esaminare. Per aumentare la visibilità di questa importante sezione, ho incluso un'icona gialla che appare solamente quando ci sono richieste in sospeso, in modo da attirare immediatamente l'attenzione dell'utente. Alla fine è presente un grafico che permette di visualizzare l'andamento del numero dei tesseramenti registrati nel corso dell'anno corrente.

La seconda colonna riguarda gli **eventi**. La prima card permette di accedere alla pagina contenente il calendario, dalla quale è possibile visualizzare e

creare nuovi eventi. Immediatamente sotto c'è la card riguardante gli eventi in sospeso, ovvero quelli che devono essere confermati. Gli operatori dalla pagina iniziale possono immediatamente controllare se ci sono eventi che non hanno ancora raggiunto il numero minimo di volontari necessario. Cliccando sulla card viene aperta la pagina con il calendario. In fondo alla pagina, è presente un grafico che rappresenta il numero di eventi organizzati nei diversi mesi dell'anno corrente, insieme al numero di partecipanti agli stessi.

L'ultima colonna è incentrata sui **luoghi**. Dalla sezione più in alto è possibile aprire la pagina dove registrare nuovi luoghi e visualizzare quelli già esistenti. Mentre nella seconda card ho integrato la mappa con una vista di insieme sui luoghi già registrati nella piattaforma.

Per quanto riguarda la sezione per la **gestione degli utenti**, visibile solamente all'amministratore, ho inserito nell'header un pulsante di colore blu, per metterlo in risalto rispetto agli altri, dal quale potervi accedere.

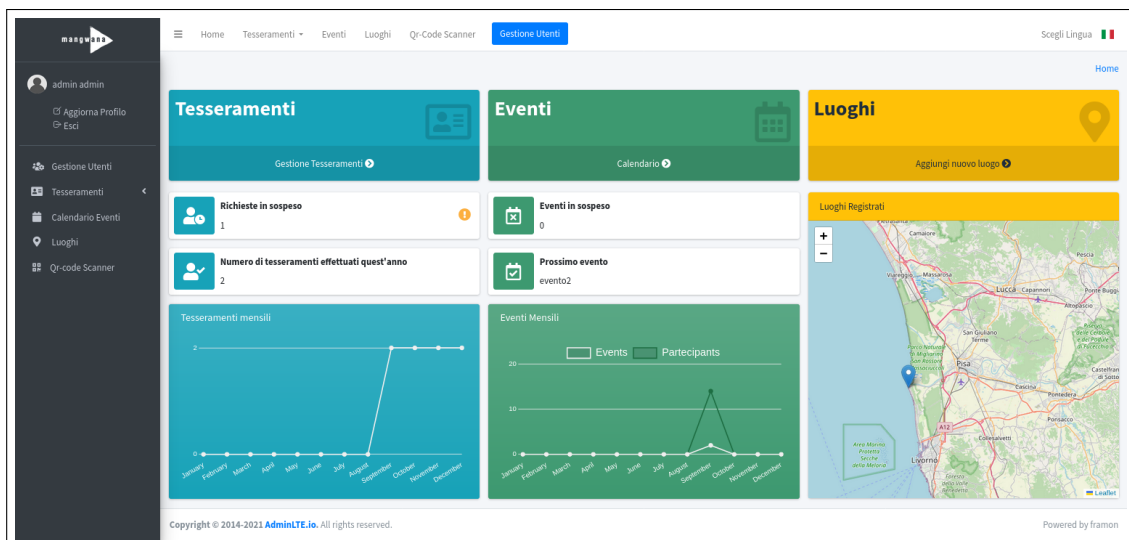


Figura 7.1: Pagina iniziale che i volontari visualizzano dopo aver effettuato l'accesso

Calendario

Un'altra pagina sulla quale ho posto particolare attenzione nella progettazione della UX è quella contenente il calendario degli eventi. Infatti come ho spiegato nella sezione [6.1 Ideazione del Calendario], ho voluto sviluppare un'interfaccia simile a quella di Google Calendar, per rendere agevole il passaggio da un sistema all'altro, ma più incentrata sui bisogni dell'associazione. Esattamente come in Google Calendar, quando un utente clicca su un evento viene aperto un popover nel quale è possibile visualizzare alcune informazioni sull'evento, quali la data, il luogo, il numero di volontari necessario e le note. Inoltre nella parte più alta del popover sono presenti vari pulsanti, rappresentati da delle icone che si riferiscono all'azione che permettono di compiere. In figura [7.2] si può vedere che il primo pulsante è quello dedicato alle prenotazioni dell'evento. Dalla pagina che si apre cliccandolo, si possono

aggiungere e visualizzare le prenotazioni di quello specifico evento, oltre ad alcune informazioni di base su di esso. Il secondo pulsante è quello di modifica, permette di accedere alla pagina dove poter modificare le informazioni riguardanti l'evento. Il terzo pulsante si occupa di eliminare l'evento, ma solamente dopo che l'utente ha confermato la sua volontà di eliminarlo attraverso un apposito popover. L'ultimo pulsante può essere utilizzato per chiudere il popover, tuttavia ho aggiunto la possibilità di chiuderlo anche quando si clicca al di fuori della sua area. In figura [7.2] è raffigurata l'interfaccia del popover nella piattaforma gestionale da me sviluppata, mentre in figura [7.3] l'interfaccia di Google Calendar. In questo modo è possibile notare le similarità e le differenze.

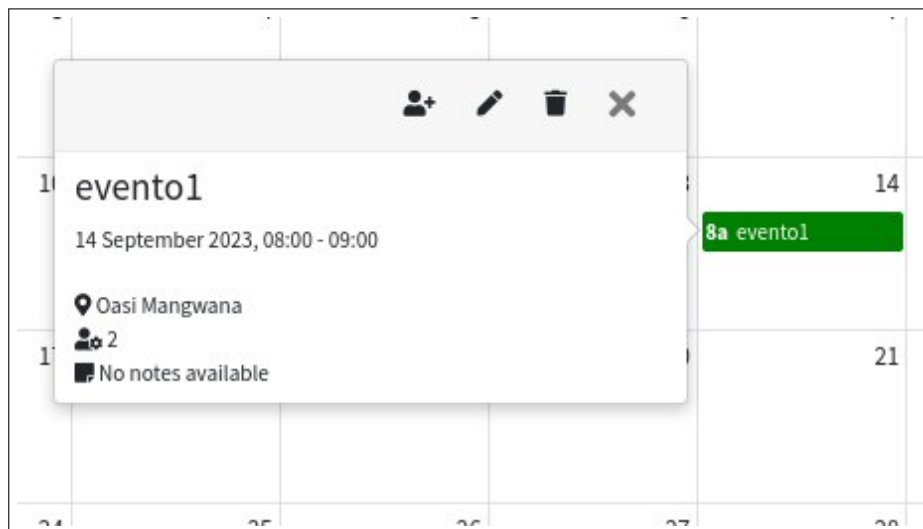


Figura 7.2: Popover che compare quando un utente clicca su un evento

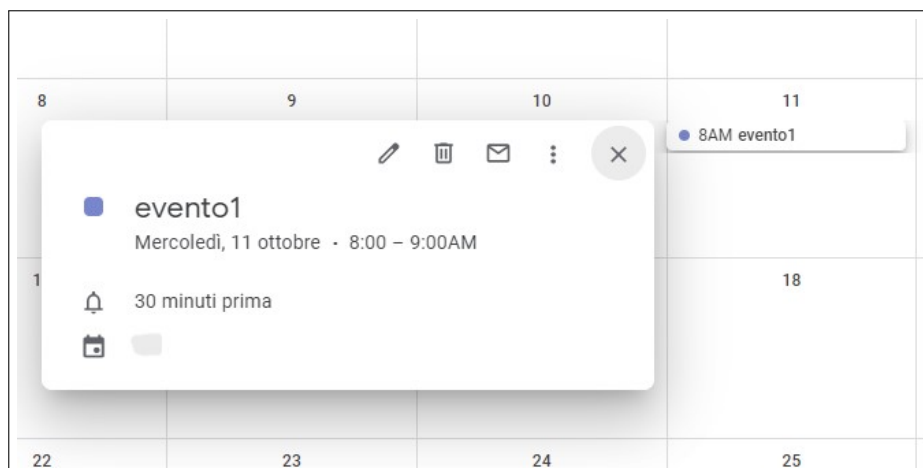


Figura 7.3: Interfaccia di Google Calendar

Sempre per quanto riguarda il calendario, come ho già spiegato nella sezione [6.1], ho deciso di distinguere visivamente gli eventi confermati (verdi) e quelli non confermati (rossi), per permettere ai volontari di capire intuitivamente lo stato degli eventi.

Feedback agli utenti

Un altro aspetto sul quale ho posto particolare attenzione sono i feedback forniti agli utenti. Infatti quando un volontario compie una qualsiasi modifica o azione all'interno della piattaforma, riceve una notifica immediata di successo o di fallimento tramite una finestra popup. Ho implementato questa funzionalità sfruttando *SweetAlert2*, una libreria JavaScript per la creazione di modali, notifiche e finestre di dialogo personalizzate e interattive all'interno delle applicazioni web. È importante sottolineare che ho differenziato visivamente le notifiche di successo da quelle di errore: le prime sono contrassegnate da un'icona verde con una spunta, mentre le seconde presentano un'icona rossa con una x. Entrambe le notifiche rimangono visibili per un periodo di 3 secondi prima di scomparire automaticamente dallo schermo.

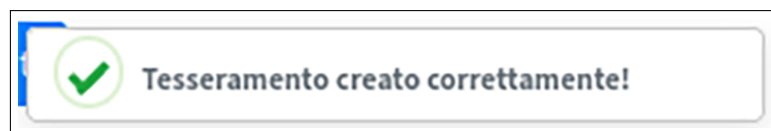


Figura 7.4: Notifica di successo



Figura 7.5: Notifica di errore

Un elemento importante della UX riguarda la prevenzione degli errori, che spesso viene ottenuta attraverso l'utilizzo di funzioni obbliganti, ovvero dei vincoli con lo scopo di impedire comportamenti inappropriati. Nel caso del gestionale quando un utente decide di eliminare un qualsiasi dato all'interno della piattaforma, ad esempio un evento oppure un tesseramento, appare sullo schermo una finestra modale che richiede la conferma dell'utente prima di procedere con l'eliminazione. Questo approccio è stato implementato per prevenire la rimozione accidentale di dati cruciali, garantendo che l'utente confermi esplicitamente la sua intenzione prima di procedere.

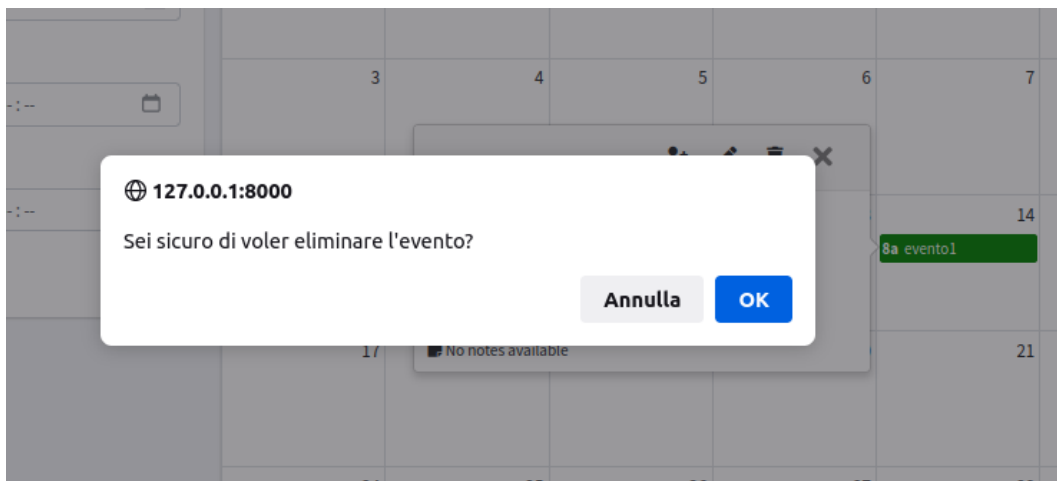


Figura 7.6: Finestra modale per la conferma dell'eliminazione di dati

Navigazione

Per migliorare l'usabilità dell'applicazione ho aggiunto una sidebar (figura [7.7]) e un header (figura [7.9]), in cui sono presenti dei link che permettono ai volontari di raggiungere tutte le sezioni della piattaforma da qualsiasi pagina si trovino. Questo rende la navigazione nel gestionale veloce e intuitiva, consentendo agli utenti di svolgere svariate operazioni nel minor tempo possibile.

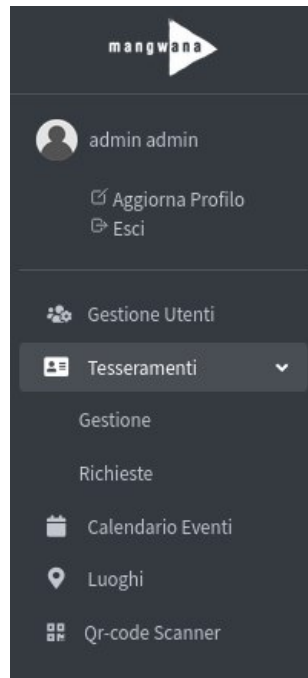


Figura 7.7: Sidebar

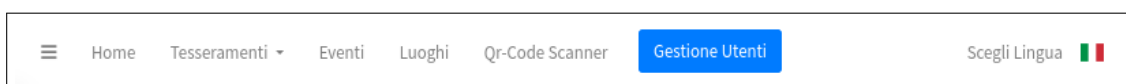


Figura 7.8: Header

Localizzazione

Nell'header, inoltre, è stata inserita la funzionalità di selezione della lingua, permettendo di scegliere tra italiano o inglese, contribuendo così a rendere l'applicazione più accessibile e adattabile alle preferenze linguistiche degli utenti. Un esempio di una pagina tradotta in inglese, attraverso la selezione della lingua, è rappresentato in figura [7.9].

The screenshot displays the 'Users Management' page in English. The header includes a 'Choose Language' dropdown set to 'EN'. The left sidebar contains navigation links for Home, Memberships, Events, Places, Qr-Code Scanner, and Users Management. The main content area shows details for 'evento1', including its date (29 September 2023, 08:00 - 09:00), place (Oasi Mangwana), total participants (9), and attending participants (5). A 'Register a new reservation' button is present. Below this is a 'Reservations' table with columns for Surname, Name, Phone Number, Participants Number, and Action. The table contains two entries. The footer shows the copyright notice 'Copyright © 2014-2021 AdminLTE.io. All rights reserved.' and 'Powered by framon'.

Surname	Name	Phone Number	Participants Number	Action
Messi	Messi		4	✓ Edit Delete
Leoni	Leoni		5	✗ Edit Delete

Figura 7.9: Esempio di una pagina con la selezione della lingua inglese

Capitolo 8

Conclusione

Nella presente relazione si è presentato il percorso di progettazione e sviluppo di una applicazione web gestionale per l'organizzazione di attività relative all'associazione Mangwana. In particolare per ogni funzionalità sviluppata, è stato descritto il processo di progettazione e quello di realizzazione, sono state analizzate le criticità dei sistemi precedentemente utilizzati dall'associazione e le soluzioni trovate per migliorarli. In questo capitolo verranno descritti gli obiettivi raggiunti, delineate alcune idee per il futuro dell'applicazione ed esposte le competenze acquisite durante il corso del tirocinio.

8.1 Obiettivi raggiunti

L'obiettivo principale di questo progetto era creare una piattaforma completa e funzionale per i volontari dell'associazione Mangwana. Questo obiettivo è stato pienamente raggiunto, la piattaforma offre una solida base per l'organizzazione degli eventi e la gestione dei tesseramenti, migliorando notevolmente l'efficienza rispetto ai sistemi utilizzati in precedenza. È stato creato un sistema di autenticazione in modo tale da permettere solamente al personale autorizzato di effettuare l'accesso nella piattaforma gestionale. È stata aggiunta una sezione riservata all'amministratore per la gestione degli utenti registrati. Dopodiché è stata realizzata la pagina, accessibile a tutti i tipi di utente, per effettuare il tesseramento, integrando i sistemi di pagamento di PayPal e Satispay, oltre ad un meccanismo per inviare richieste di tesseramento tramite IBAN. È stato creato un sistema per permettere ai volontari di controllare la validità dei tesseramenti durante gli eventi, attraverso codici QR. Infine è stata progettata una sezione dedicata all'organizzazione degli eventi, con integrazione di un calendario, e una sezione per aggiungere alla piattaforma, utilizzando una mappa integrata, i luoghi in cui solitamente si svolgono tali eventi. Un altro importante obiettivo era rendere la piattaforma facile da usare e intuitiva per gli utenti. Questo è stato raggiunto attraverso la

progettazione di un'interfaccia utente chiara e semplice e l'implementazione di feedback per gli utenti.

Una volta che lo sviluppo dell'applicazione è stato completato, i volontari dell'associazione Mangwana hanno potuto esaminare il lavoro svolto, ritenendolo valido, per questo motivo il deployment dell'applicazione è previsto per l'inizio del 2024.

8.2 Sviluppi futuri

Nonostante siano stati raggiunti tutti gli obiettivi prefissati all'inizio del tirocinio, possono essere attuati dei miglioramenti e progettate nuove funzionalità per ottimizzare l'uso della piattaforma. Prima di tutto l'applicazione web è stata sviluppata per essere utilizzata principalmente su computer e sebbene sia attualmente funzionante anche su dispositivi mobili, è possibile apportare miglioramenti all'interfaccia grafica al fine di ottimizzare la visualizzazione su schermi più piccoli.

Per quanto riguarda vere e proprie funzionalità ho pensato alle seguenti soluzioni:

- L'implementazione di un sistema di notifiche. Questo sistema avviserebbe i volontari su diverse informazioni importanti, ad esempio un promemoria sull'imminenza di eventi oppure notifiche relative alle richieste di tesseramento ricevute.
- La promozione degli eventi organizzati dall'associazione, mediante l'invio di email. Ovvero quando un evento viene confermato il sistema invia automaticamente una email, con le informazioni relative all'evento, a tutti i possessori della tessera associativa.
- L'integrazione con i social media, facilitando anche in questo caso la promozione degli eventi.
- L'implementazione di un sistema di chat interno dove i volontari possono discutere, condividere idee e comunicare in tempo reale.
- L'integrazione di altre piattaforme per il pagamento online oltre a quelle già utilizzate.

8.3 Competenze acquisite

Durante lo sviluppo di questo progetto ho avuto l'opportunità di acquisire nuove competenze che ritengo importanti per la mia crescita professionale. Inoltre ho potuto applicare nella pratica diversi concetti e conoscenze acquisite durante il percorso di studi della triennale, avendo modo di approfondire ulteriormente tali argomenti.

In primo luogo ho ottenuto conoscenze fondamentali riguardo alla progettazione e lo sviluppo di applicazioni web. Sono riuscita ad implementare un'intera piattaforma web, comprensiva sia della parte back-end che di quella front-end. È importante sottolineare che, prima di questo tirocinio, non avevo mai avuto l'occasione di lavorare direttamente sulla programmazione di un'interfaccia grafica, grazie a questa esperienza ho acquisito competenze fondamentali nell'uso di HTML, JavaScript e CSS, che sicuramente saranno utili per il mio percorso futuro.

Oltre a questo, non avendo conoscenze pregresse sul linguaggio PHP, né sul framework Laravel, attraverso lo studio iniziale e la successiva implementazione dell'applicazione sono riuscita ad ottenere una solida competenza in entrambi.

Ho approfondito la progettazione e la gestione di database relazionali, attraverso l'utilizzo di MariaDB. In particolare mettendo in pratica la scrittura di query SQL, che avevo precedentemente studiato durante il mio percorso di studi.

In conclusione, il tirocinio è stato un'esperienza che ha contribuito in modo significativo alla mia crescita accademica e professionale. Ho avuto l'opportunità di lavorare in un ambiente di lavoro aziendale, e apprendere i fondamenti dello sviluppo web.

Bibliografia

- [1] *AJAX - Wikipedia*. URL: <https://it.wikipedia.org/wiki/AJAX> (cit. a p. 36).
- [2] *Authentication - Laravel 7.x - The PHP Framework For Web Artisans*. URL: <https://laravel.com/docs/7.x/authentication> (cit. a p. 10).
- [3] *Blade Templates - Laravel 10.x - The PHP Framework For Web Artisans*. URL: <https://laravel.com/docs/10.x/blade> (cit. a p. 7).
- [4] *Controlllers - Laravel 10.x - The PHP Framework For Web Artisans*. URL: <https://laravel.com/docs/10.x/controllers> (cit. a p. 5).
- [5] *Database: Query Builder - Laravel 10.x - The PHP Framework For Web Artisans*. URL: <https://laravel.com/docs/10.x/queries> (cit. a p. 17).
- [6] *Database: Seeding - Laravel 10.x - The PHP Framework For Web Artisans*. URL: <https://laravel.com/docs/10.x/seeding> (cit. a p. 18).
- [7] *Introduction to Laravel and MVC Framework - GeeksforGeeks*. URL: <https://www.geeksforgeeks.org/introduction-to-laravel-and-mvc-framework/> (cit. a p. 5).
- [8] *Laravel - The PHP Framework For Web Artisans*. URL: <https://laravel.com/docs/10.x> (cit. a p. 6).
- [9] *Mail - Laravel 10.x - The PHP Framework For Web Artisans*. URL: <https://laravel.com/docs/10.x/mail> (cit. a p. 32).
- [10] *MariaDB Foundation - MariaDB.org*. URL: <https://mariadb.org/> (cit. a p. 9).
- [11] *More Than Usability: The Four Elements of User Experience, Part I :: UXmatters*. URL: <https://www.uxmatters.com/mt/archives/2012/04/more-than-usability-the-four-elements-of-user-experience-part-i.php> (cit. a p. 38).
- [12] *One-off - Web-redirect*. URL: <https://developers.satispay.com/docs/web-redirect-pay> (cit. a p. 30).
- [13] *PHP: Hypertext Preprocessor*. URL: <https://www.php.net> (cit. a p. 4).
- [14] *reCAPTCHA*. URL: <https://www.google.com/recaptcha/about/> (cit. a p. 11).
- [15] *Send Emails In Laravel 8 Using Gmail's SMTP Server*. URL: <https://www.twilio.com/blog/send-emails-laravel-8-gmail-smtp-server> (cit. a p. 6).

- [16] *UX Blog - Gli elementi della User Experience*. URL: <https://www.uxblog.it/gli-elementi-della-user-experience/> (cit. a p. 38).
- [17] *Why MariaDB? Advantages over MySQL / MariaDB*. URL: <https://mariadb.com/resources/blog/why-should-you-migrate-from-mysql-to-mariadb/> (cit. a p. 10).

Ringraziamenti

Inanzitutto vorrei ringraziare il mio tutor aziendale, Domenico Arenga, per avermi aiutato e guidato durante il corso del tirocinio presso l'azienda Visual Engines.

Ringrazio anche il mio tutor accademico, il Professore Alessio Conte, per avermi dato validi consigli sulla stesura della relazione.

Un grazie di cuore ai miei genitori, per essermi sempre stati vicini ed avermi sostenuto e incoraggiato costantemente durante tutto il mio percorso universitario. Anche se non ve lo dico mai vi voglio bene.

Grazie anche a tutta la mia famiglia per il sostegno e in particolare ai miei nonni per il loro affetto.

Ai miei amici, Gioia, Lorenzo, Valentina e Viola, anche se non riusciamo a vederci più spesso come una volta, siete sempre stati e continuate a essere di grande supporto, ormai da 10 anni.

Agli amici che ho conosciuto durante questo percorso, Alessandra, Andrea, Diletta, Francesco e Laura, grazie per aver condiviso insieme a me sia i bei momenti, che quelli più difficili. Avete reso questi anni di università speciali.