Gestione Organizzazione Eventi

**Documentazione**

1 Introduzione 3

1.1 Informazioni sul progetto 3

1.2 Abstract 3

1.3 Scopo 3

2 Analisi 4

2.1 Analisi del dominio 4

2.2 Analisi e specifica dei requisiti 4

2.3 Use case 11

2.4 Pianificazione 11

2.5 Analisi dei mezzi 13

2.5.1 Software 13

2.5.2 Hardware 13

3 Progettazione 14

3.1 Design dell’architettura del sistema 14

3.1.1 Model View Controller 14

3.1.2 Laravel 15

3.2 Design dei dati e database 17

3.3 Design delle interfacce 19

4 Implementazione 25

5 Test 37

5.1 Protocollo di test 37

5.2 Risultati test 41

5.3 Mancanze/limitazioni conosciute 45

6 Consuntivo 45

7 Conclusioni 47

7.1 Sviluppi futuri 47

7.2 Considerazioni personali 47

8 Glossario 47

9 Bibliografia 49

9.1 Bibliografia per articoli di riviste: 49

9.2 Bibliografia per libri 49

9.3 Sitografia 49

9.4 Indice delle figure 49

10 Allegati 50

# Introduzione

## Informazioni sul progetto

* Svolto dall’allievo: Gianni Grasso
* Docente responsabile: Guido Montalbetti
* Scuola e classe: SAMT I4AC
* Sezione: Informatica
* Periodo: 29.08.2022 – 07.12.2022

## Abstract

*With the progress of the technological development, many companies and organizations have had the need to digitalize their services. Event and event planning is one of the spheres that had to adapt to this transition most urgently. This is because of having to handle a large number of participants and establish roles and possible hierarchy. This project was assigned to me with the objective of simplifying the management and organization of small events. With this Web app it is possible to easily and intuitively create and manage events, participants, and hotels for overnight stays. The objective is to facilitate the experience of those who organize events and those who attend them.*

## Scopo

Lo scopo del progetto è quello di creare un’applicazione web per la gestione e l’organizzazione di eventi di massimo 400 persone, al fine di semplificare il più possibile la creazione di gruppi all’interno di esso e l’assegnazione delle stanze nella quale risiedono i partecipanti e i membri dello staff. Oltre alla gestione dei vari gruppi l’applicazione deve occuparsi di dividere i vari membri secondo la loro età e il loro sesso, inoltre occorre garantire la sicurezza degli utenti registrati e stabilire una gerarchia tra amministratori, membri dello staff (che si dividono in consiglieri e assistenti) e partecipanti degli eventi.

# 2 Analisi



## Analisi del dominio

Il progetto è pensato per l’organizzazione di eventi, di conseguenza è rivolto ad aziende che si occupano di questo tipo di servizio. Un sistema di gestione degli eventi semplifica il lavoro di chi si occupa di organizzare uno o più eventi di una certa capienza e garantisce ai partecipanti un’esperienza migliore, semplificando la comunicazione e la creazione di gruppi tra di loro. Anche il ruolo dei membri dello staff degli eventi è semplificato al massimo, per poter permettere loro di avere una gestione intuitiva dei gruppi che si formano all’interno dei vari eventi.

## Analisi e specifica dei requisiti

|  |  |
| --- | --- |
| **ID: REQ-001** | |
| **Nome** | Creazione database |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | L’applicativo deve prevedere un database per il salvataggio dei dati. |

|  |  |
| --- | --- |
| **ID: REQ-002** | |
| **Nome** | Login utenti |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | L’applicativo deve prevedere l'accesso tramite credenziali per gli utenti. |
| **Sotto requisiti** | |
| **001** | È necessaria una pagina di Login. |
| **002** | Le password devono essere criptate e deve essere fatto il salt. |

|  |  |
| --- | --- |
| **ID: REQ-003** | |
| **Nome** | Cambio password |
| **Priorità** | 3 |
| **Versione** | 1.1 |
| **Note** | I nuovi utenti devono obbligatoriamente cambiare la password al primo accesso al sito. |
| **Sotto requisiti** | |
| **001** | Sistema di login funzionante. |

|  |  |
| --- | --- |
| **ID: REQ-004** | |
| **Nome** | Importazione tramite csv |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | Gli utenti admin devono poter creare utenti tramite l’import di un file csv. |
| **Sotto requisiti** | |
| **001** | La struttura del database deve essere corretta. |
| **002** | Sistema di login funzionante. |

|  |  |
| --- | --- |
| **ID: REQ-005** | |
| **Nome** | Creazione di un evento |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | Gli utenti admin devono poter creare eventi. |
| **Sotto requisiti** | |
| **001** | La struttura del database deve essere corretta. |
| **002** | È necessaria una pagina apposita per la creazione di un evento. |
| **003** | Importazione csv funzionante. |

|  |  |
| --- | --- |
| **ID: REQ-006** | |
| **Nome** | Modifica di un evento |
| **Priorità** | 2 |
| **Versione** | 1.0 |
| **Note** | Gli utenti admin devono poter modificare eventi. |
| **Sotto requisiti** | |
| **001** | La struttura del database deve essere corretta. |
| **002** | È necessaria una pagina apposita per la modifica di un evento. |

|  |  |
| --- | --- |
| **ID: REQ-007** | |
| **Nome** | Eliminazione di un evento |
| **Priorità** | 2 |
| **Versione** | 1.0 |
| **Note** | Gli utenti admin devono poter eliminare eventi. |
| **Sotto requisiti** | |
| **001** | La struttura del database deve essere corretta. |
| **002** | Anche i collegamenti con gli utenti iscritti ed eventuali tabelle ponte devono essere eliminati. |

|  |  |
| --- | --- |
| **ID: REQ-008** | |
| **Nome** | Lista eventi |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | Ogni utente deve poter visualizzare la lista degli eventi alla quale partecipa. |
| **Sotto requisiti** | |
| **001** | Ci deve essere una relazione tra le tabelle utente ed evento del database. |

|  |  |
| --- | --- |
| **ID: REQ-009** | |
| **Nome** | Creazione di un hotel |
| **Priorità** | 2 |
| **Versione** | 1.1 |
| **Note** | Gli utenti admin devono poter creare hotel. |
| **Sotto requisiti** | |
| **001** | La struttura del database deve essere corretta. |
| **002** | È necessaria una pagina apposita per la creazione di un hotel. |

|  |  |
| --- | --- |
| **ID: REQ-010** | |
| **Nome** | Modifica di un hotel |
| **Priorità** | 2 |
| **Versione** | 1.1 |
| **Note** | Gli utenti admin devono poter modificare gli hotel. |
| **Sotto requisiti** | |
| **001** | La struttura del database deve essere corretta. |
| **002** | È necessaria una pagina apposita per la modifica di un hotel. |

|  |  |
| --- | --- |
| **ID: REQ-011** | |
| **Nome** | Eliminazione di un hotel |
| **Priorità** | 2 |
| **Versione** | 1.1 |
| **Note** | Gli utenti admin devono poter eliminare gli hotel. |
| **Sotto requisiti** | |
| **001** | La struttura del database deve essere corretta. |
| **002** | All’eliminazione di un hotel devono essere eliminate anche tutte le camere appartenenti ad esso. |

|  |  |
| --- | --- |
| **ID: REQ-012** | |
| **Nome** | Creazione camere |
| **Priorità** | 2 |
| **Versione** | 1.1 |
| **Note** | Gli utenti admin devono poter creare nuove camere per gli hotel. |
| **Sotto requisiti** | |
| **001** | La struttura del database deve essere corretta. |

|  |  |
| --- | --- |
| **ID: REQ-013** | |
| **Nome** | Dettagli hotel |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | Gli utenti admin devono poter visualizzare i dettagli di un hotel. |
| **Sotto requisiti** | |
| **001** | Ci deve essere una relazione tra le tabelle hotel e camera del database. |
| **002** | È necessaria una pagina apposta per i dettagli di un hotel. |
| **003** | Devono poter essere visualizzate tutte le camere di un hotel nella pagina dei dettagli di esso. |

|  |  |
| --- | --- |
| **ID: REQ-014** | |
| **Nome** | Lista hotel |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | Gli utenti admin devono poter visualizzare una lista di tutti gli hotel. |
| **Sotto requisiti** | |
| **001** | La struttura del database deve essere corretta. |

|  |  |
| --- | --- |
| **ID: REQ-015** | |
| **Nome** | Dettagli evento |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | Bisogna poter visualizzare i dettagli di un evento. |
| **Sotto requisiti** | |
| **001** | Bisogna poter visualizzare tutti i partecipanti di un determinato evento. |
| **002** | Bisogna poter visualizzare tutti i gruppi di un determinato evento. |
| **003** | Bisogna poter visualizzare tutte le compagnie di un determinato evento. |
| **004** | Bisogna poter visualizzare l’hotel e le camere di un determinato evento. |
| **005** | Bisogna poter cercare e filtrare tutti i partecipanti e i gruppi. |

|  |  |
| --- | --- |
| **ID: REQ-016** | |
| **Nome** | Organizzazione gruppi |
| **Priorità** | 2 |
| **Versione** | 1.0 |
| **Note** | Gli utenti devono essere divisi in gruppi da 9 a 12 secondo il sesso (maschio/femmina) e l’età (14-15, 15-16, 17-18). |

|  |  |
| --- | --- |
| **ID: REQ-017** | |
| **Nome** | Consiglieri |
| **Priorità** | 2 |
| **Versione** | 1.0 |
| **Note** | Devono essere presenti degli utenti consiglieri che possono creare gruppi all’interno degli eventi. |
| **Sotto requisiti** | |
| **001** | I consiglieri sono considerati membri dello staff. |

|  |  |
| --- | --- |
| **ID: REQ-018** | |
| **Nome** | Organizzazione compagnie |
| **Priorità** | 3 |
| **Versione** | 1.0 |
| **Note** | Deve essere possibile unire più gruppi, anche di sessi diversi, per creare delle compagnie. |

|  |  |
| --- | --- |
| **ID: REQ-019** | |
| **Nome** | Assistenti |
| **Priorità** | 3 |
| **Versione** | 1.0 |
| **Note** | Per ogni 4/5 compagnie deve esserci una coppia di assistenti. |
| **Sotto requisiti** | |
| **001** | Gli assistenti sono considerati membro dello staff. |
| **002** | Un utente assistente deve essere necessariamente membro dello staff (quindi almeno consigliere). |

|  |  |
| --- | --- |
| **ID: REQ-020** | |
| **Nome** | Assegnazione automatica |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | Alla creazione di un evento gli utenti devono essere automaticamente assegnati alle camere. |
| **Sotto requisiti** | |
| **001** | Le assegnazioni alle camere devono essere fatte in base ai criteri di sesso ed età. |

|  |  |
| --- | --- |
| **ID: REQ-021** | |
| **Nome** | Manuale di sistema |
| **Priorità** | 2 |
| **Versione** | 1.1 |
| **Note** | Deve essere presente un manuale di sistema per l’installazione dell’applicativo. |

## Use case

Figura : Use Case

*Oltre ai casi d’uso dello schema, ogni attore (partecipante, consigliere e admin) deve effettuare un* ***login al sito*** *per svolgere qualsiasi altra azione.*

In questo use case ci sono tre attori principali, i partecipanti degli eventi, i membri dello staff e l’amministratore. Un partecipante può visualizzare un evento e eventualmente le compagnie, i gruppi e i partecipanti di esso. I membri dello staff si dividono in consiglieri e assistenti. Entrambi hanno le stesse possibilità dei partecipanti di visualizzare gli eventi, i consiglieri dirigono un gruppo quindi hanno anche la possibilità di apportare delle modifiche ad esso dopo averlo visualizzato. Gli assistenti invece, oltre alle caratteristiche dei consiglieri, possono modificare le compagnie. Infine l’amministratore ha la possibilità di creare, cancellare e modificare eventi.

## Pianificazione

La pianificazione è stata fatta realizzando un diagramma di Gantt del progetto, ci sono quattro fasi principali, l’analisi, la progettazione, l’implementazione e i test. Nell’analisi l’obbiettivo è quello di raccogliere più informazioni possibili sul progetto e sui requisiti in modo da poter arrivare alla fase di progettazione, in cui si definisce come deve essere realizzato il progetto e come deve essere graficamente. L’implementazione è la fase in cui viene effettivamente realizzato il progetto (divisa in frontend e backend), infine la fase di test in cui si verifica che tutto funzioni correttamene. Il progetto verrà svolto seguendo una metodologia Waterfall.

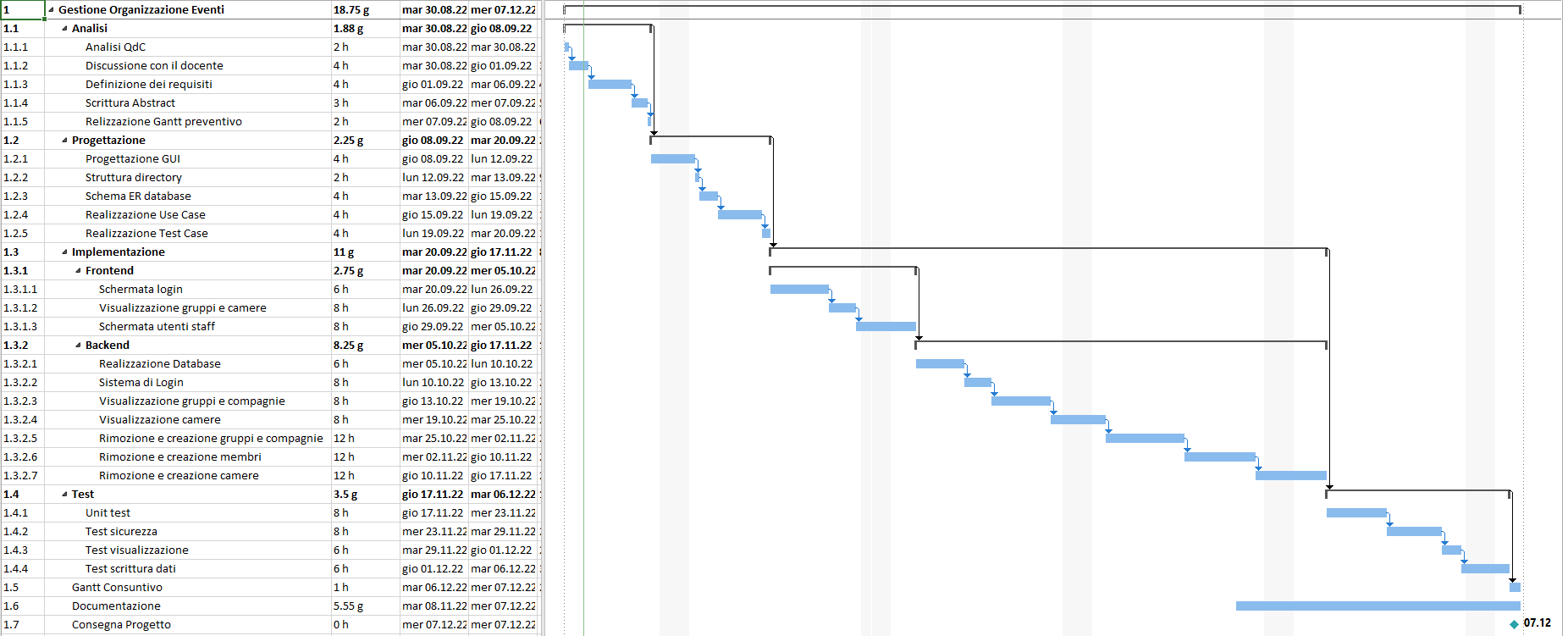


Figura : Diagramma di Gantt preventivo

## Analisi dei mezzi

### Software

Per la realizzazione di questo progetto sono state utilizzate le seguenti tecnologie e i seguenti software:

* Laravel
* Bootstrap
* MySQL
* PHP
* HTML5
* Javascript
* CSS

### Hardware

L’applicativo è stato pensato per essere messo sull’hosting della scuola o su un hosting esterno.

L’unica macchina che è stata usata per questo progetto è stata un PC scolastico.

# Progettazione

## Design dell’architettura del sistema

### Model View Controller

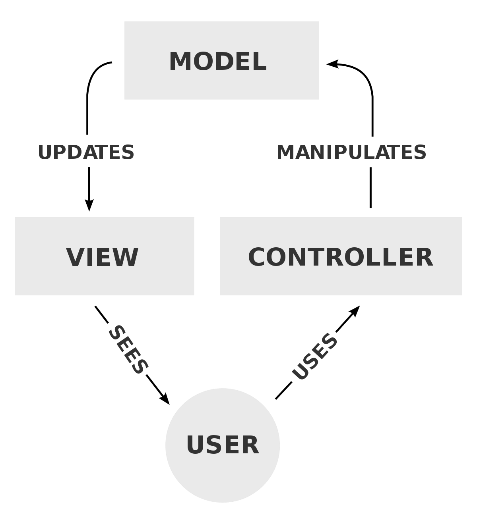
Per realizzare questo progetto è stato utilizzato il framework Laravel che implementa un’architettura MVC (Model View Controller). In una struttura di questo tipo ci sono tre cartelle principali, ognuna di esse contiene una parte del funzionamento del programma e possiede un ruolo specifico e fondamentale. Qui sotto è presente uno schema che riassume il funzionamento della struttura MVC:

Figura : Schema struttura MVC

Nella cartella View sono presenti tutti i file che contengono codice html, generalmente contengono poco codice php ed è quindi la parte grafica contenente tutte le schermate che vengono mostrate all’utente finale. Con l’utilizzo di Laravel, i file in questa directory implementano Blade, un Templating Engine dedicato che permette di inserire delle parti di codice php all’interno di un file html in modo semplice e veloce.

Nella cartella Model invece vengono messe tutte le classi php, generalmente sono una per ogni tabella del database. Anche in questo caso Laravel ha un’utilità facendo in modo che ogni classe sia direttamente associata ad una tabella del database. Le classi presenti in questa cartella saranno quindi prevalentemente vuote e quando ci sarà bisogno di utilizzare un attributo di una di queste classi, si potrà fare riferimento direttamente al database o viceversa.

Infine la cartella Controller è quella che collega i models e le views, all’interno di essa sono contenute le classi che ricevono effettivamente delle richieste da parte del browser che vengono elaborate, associate ai models e in seguito passate alle views che eseguono il templating e rendono l’applicazione dinamica.

### Laravel

Per quanto riguarda invece l’architettura di cartelle del progetto, è piuttosto complessa ed è anch’essa generata da Laravel, in questa documentazione vengono spiegate e approfondite solo le cartelle alla quale ho messo maggiormente mano e quelle che reputo più importanti affinché possa essere capito il progetto, non vengono approfondite tutte le cartelle e i file presenti nel framework. La struttura di cartelle generata da un progetto Laravel è la seguente:

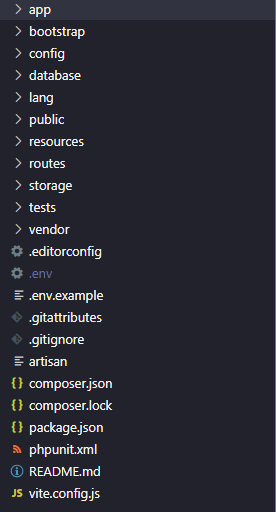


Figura : Struttura cartelle di un progetto Laravel

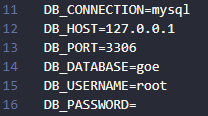
Analizzando in primo luogo i singoli file alla radice del progetto, tra i vari file di configurazione c’è il file ‘.env’. All’interno di esso sono presenti le configurazioni per quanto riguarda la connessione al database, in questo progetto mi sono affidato ad una banca dati MySQL:

Figura : Esempio di un file *.env*

È presente anche un file ‘.env.example’ che è pressoché identico ed è fornito al fine di facilitare lo sviluppo di un progetto svolto in team, visto che molti file di configurazione tra cui ‘.env’ sono presenti nel file ‘.gitignore’ e quindi non vengono caricati sul repository.

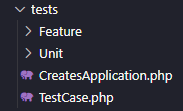
La cartella tests contiene tutti i file necessari per eseguire degli unit test sull’applicazione, che però non sono stati effettuati in questo progetto, l’applicazione è stata testata con test sull’applicativo funzionante e non sulle classi.

Figura : Cartella tests

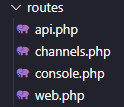
Una cartella che invece è molto importante e viene ampliamente utilizzata è la cartella routes. All’interno di essa si trova il file ‘web.php’, che contiene tutte le rotte dell’applicazione, si occupa di gestire le richieste GET e POST dell’applicazione e di richiamare l’opportuno controller.

Figura : Cartella routes

Nella cartella resources sono presenti tutti i file riguardanti lo stile dell’applicazione, grafica compresa. All’interno di essa si trova anche la cartella views che, come già menzionato, contiene tutte le viste e le schermate dell’applicazione.

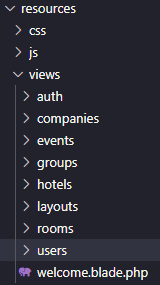


Figura : Cartella views in resources

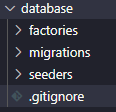
All’interno della cartella database sono presenti altre 3 directory, ciascuno con una funzione ben precisa legata alla banca dati dell’applicativo. La cartella factories contiene le classi utilizzate per generare dati verosimili per il testing del database, possono essere create delle classi in base ai models dell’applicazione e creare dati fittizi per la fase di test. La cartella seeder contiene invece le classi per inviare questi dati al database e si occupa quindi di eseguire le query per i test. Per quanto riguarda invece la cartella migrations, essa è sicuramente la più utilizzata poiché non è necessariamente legata a dati fittizi o usati per testare l’applicazione. Essa contiene infatti tutte le migrations che generano poi le tabelle effettive del database, grazie a questo sistema è possibile generare la banca dati comodamente e modificarne la struttura anche durante lo sviluppo del progetto.

Figura : Cartella database

Infine la cartella app, all’interno della quale solitamente si mette maggiormente mano durante un progetto Laravel. Essa contiene tutto quello che concerne il funzionamento pratico dell’applicazione, le due cartelle principali sono Controllers, situata all’interno della cartella http, e Models. Queste due directories hanno lo stesso ruolo delle corrispettive cartelle presenti in una qualsiasi struttura MVC.

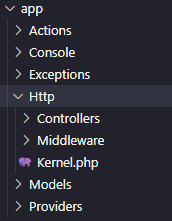


Figura : Cartella Controllers e Models in app

## Design dei dati e database

Per la progettazione del database è stato realizzato uno schema ER. Degli utenti registrati al sito bisogna conoscere le informazioni anagrafiche, inoltre esistono diversi tipi di utenti (partecipanti, consiglieri e assistenti, oltre all'utente admin). Ogni evento ha un proprio nome e una data, al suo interno ci sono diversi gruppi (divisi per sesso ed età), gli utenti iscritti ad un evento quindi faranno parte di un gruppo, un utente può appartenere ad un solo gruppo per evento e viene assegnato ad esso non appena entra a far parte dell'evento. Possono esistere più gruppi uguali, ma non all'interno dello stesso evento (come per gli utenti iscritti). Oltre ai gruppi ci sono poi le compagnie che sono unione di due gruppi (anche di sesso differente), non tutti i gruppi devono appartenere necessariamente ad una compagnia. Ogni evento ha poi un hotel nella quale pernottano i partecipanti, un evento ha un solo hotel ed ogni hotel può appartenere ad un solo evento.

Ogni hotel ha dei piani delle camere, una camera ha un numero univoco all'interno di un hotel specifico.

Infine ovviamente, ogni utente deve essere assegnato ad una camera, un utente può essere assegnato ad una sola camera per evento

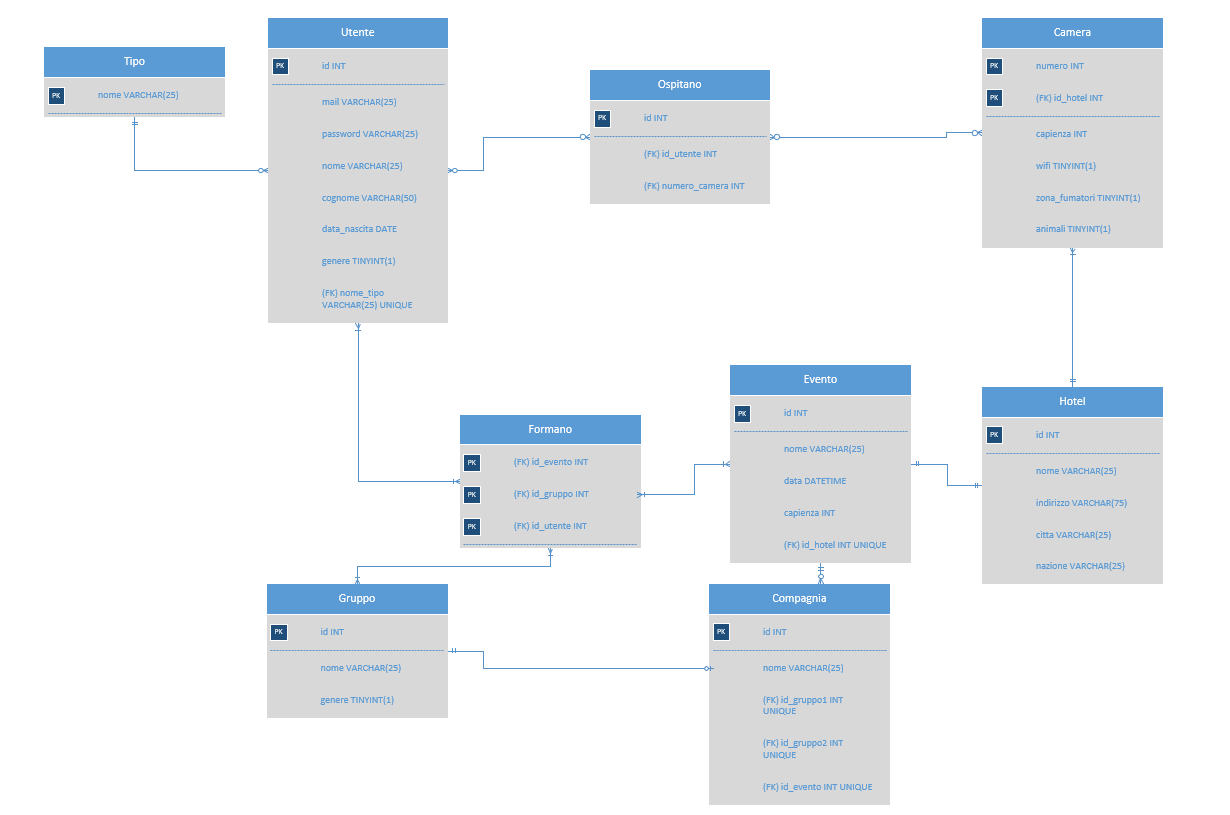


Figura : Schema ER del database

## Design delle interfacce

Figura : Schermata di login

In questa schermata è possibile effettuare il login nell’applicativo. Quando si preme il pulsante di login, se le credenziali non sono corrette appare un messaggio di errore nel form, altrimenti si viene indirizzati alla pagina degli eventi.

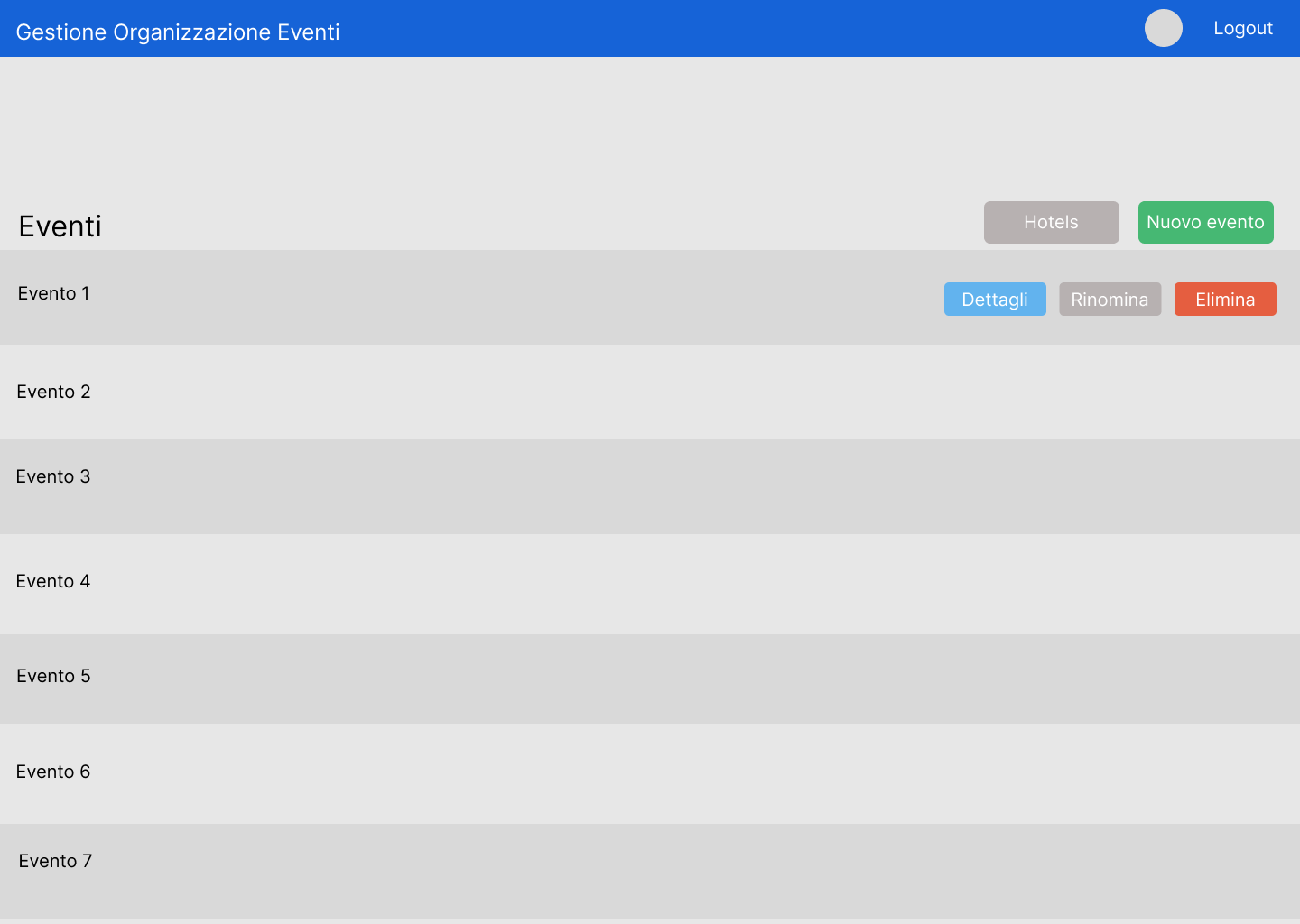


Figura : Schermata home per utenti admin

Nella pagina home degli utenti admin è presente una lista con tutti gli eventi con diversi bottoni per visualizzarli, modificarli o eliminarli. Sono anche presenti altri due bottoni, uno per la creazione di un nuovo evento ed un altro per passare alla schermata degli hotel.

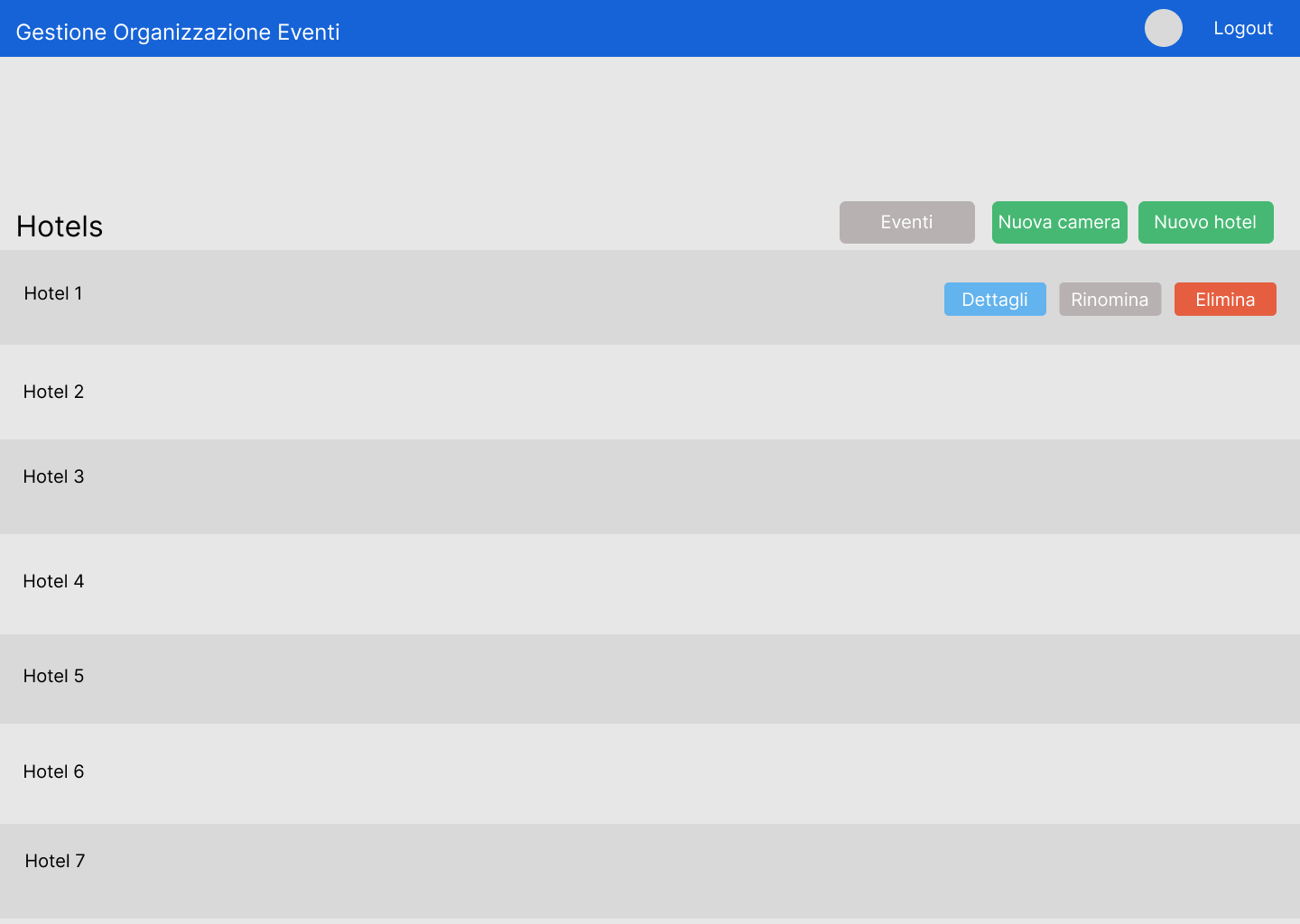
La pagina home degli utenti consiglieri e partecipanti è molto simile a quella degli admin. In questa pagina non sono presenti bottoni per la modifica o l’eliminazione degli eventi e non è possibile crearne di nuovi. Inoltre, a differenza di un amministratore, un utente normale vede soltanto gli eventi alla quale è iscritto nella lista.

Figura : Lista di hotel per utenti admin

Figura : Schermata home per utenti consiglieri e partecipanti

Pagina con la lista di tutti gli hotel, con la stessa struttura della pagina home per vedere gli eventi. Anche questa pagina è visualizzabile solo dagli utenti admin e offre le stesse funzionalità.

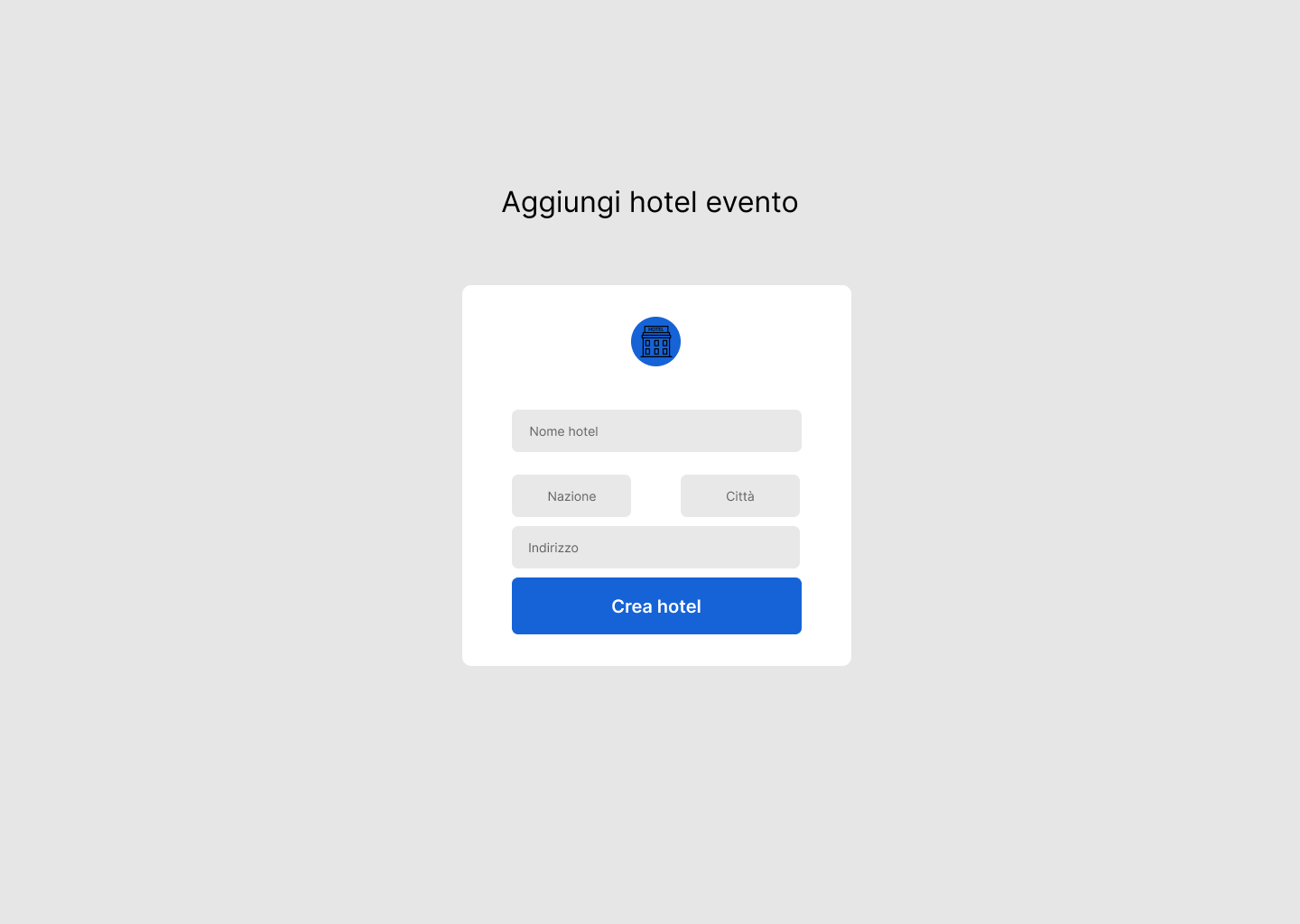
Qualora si premesse sul bottone per la creazione di un nuovo hotel verrà visualizzata questa pagina. Essa è disponibile solo agli amministratori e contiene un semplice form per la creazione di un nuovo hotel.

Figura : Pagina creazione hotel

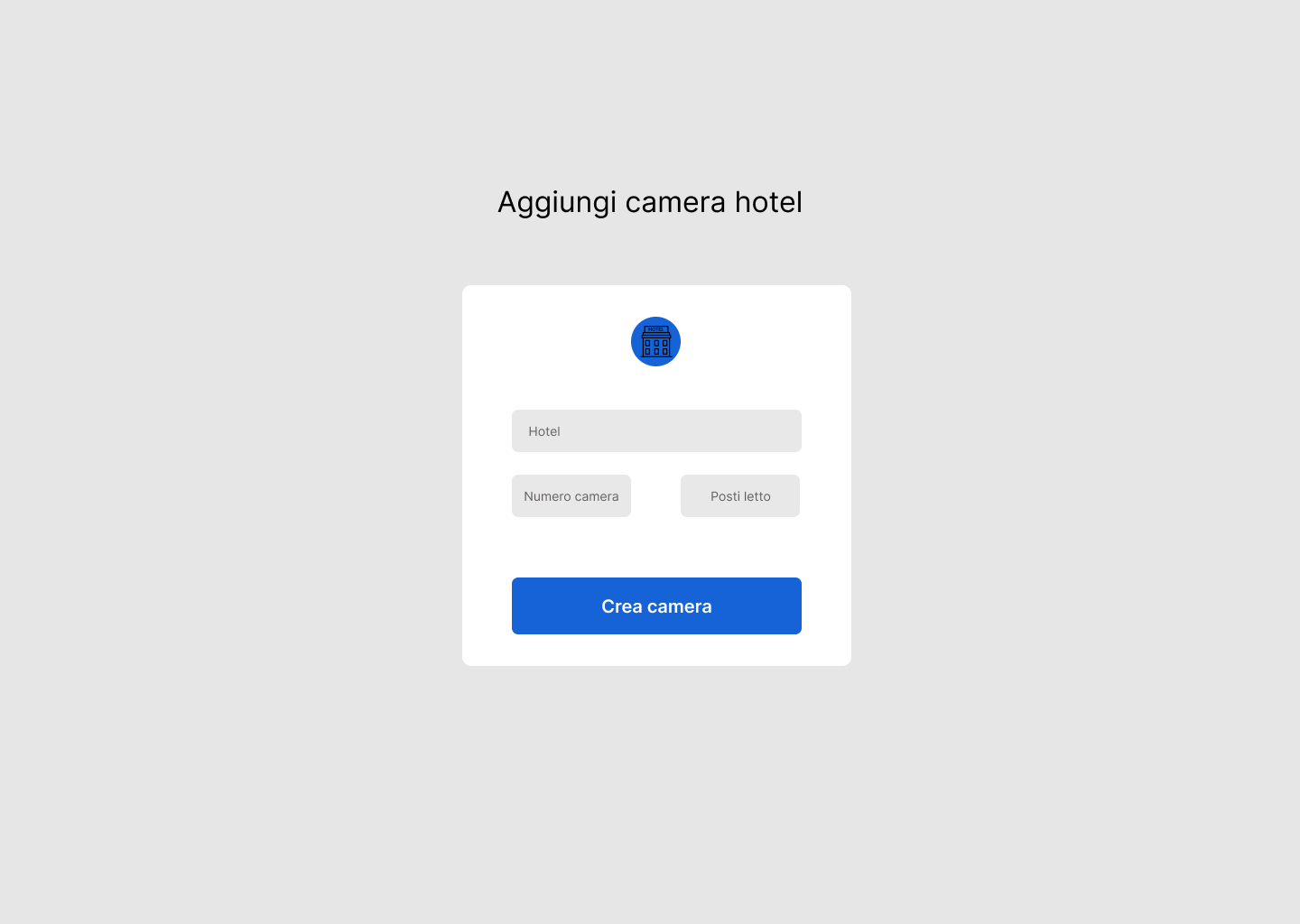


Figura : Pagina creazione camera

Nel caso in cui invece si premesse sul bottone per la creazione di una camera, la pagina visualizzata sarà questa. Questa pagina ha un funzionamento molto simile a quella dedicata alla creazione di un albergo. Selezionando il campo hotel, si può visualizzare una lista con tutti gli hotel creati per selezionare a quale hotel deve appartenere la camera.

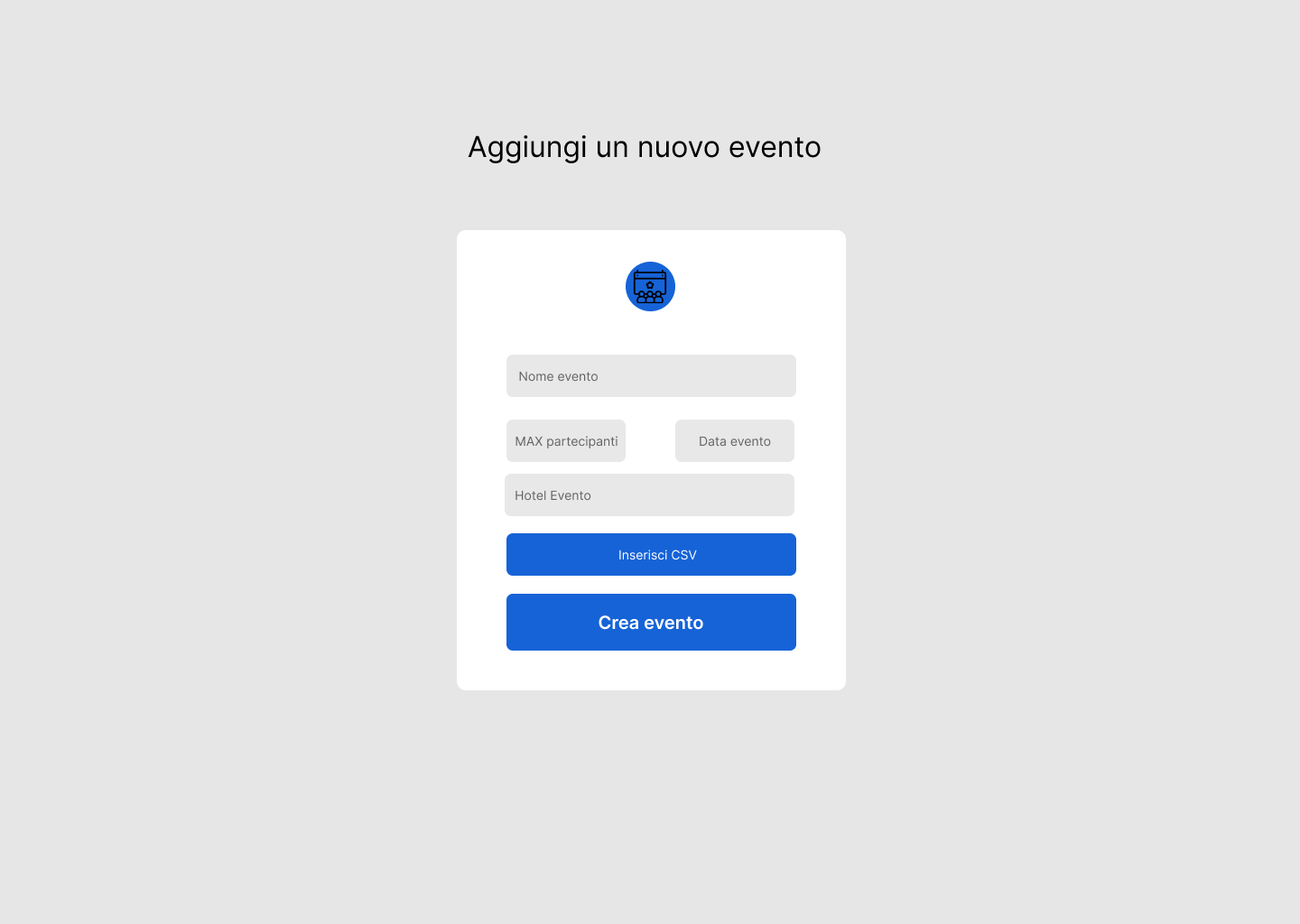
Questa è la pagina di creazione di un evento grazie alla quale l’amministratore può aggiungere un evento e i relativi partecipanti. Infatti, premendo sul bottone per l’inserimento di un file csv sarà possibile importare un file e verranno inseriti tutti i partecipanti contenuti in esso all’interno dell’evento creato.

Figura : Pagina creazione evento

Immagine che contiene tavolo

Descrizione generata automaticamente

Figura : Pagina dettagli hotel

In questa pagina è possibile visualizzare tutte le camere di un hotel e la relativa capienza. C’è anche la possibilità di eliminare una determinata camera.

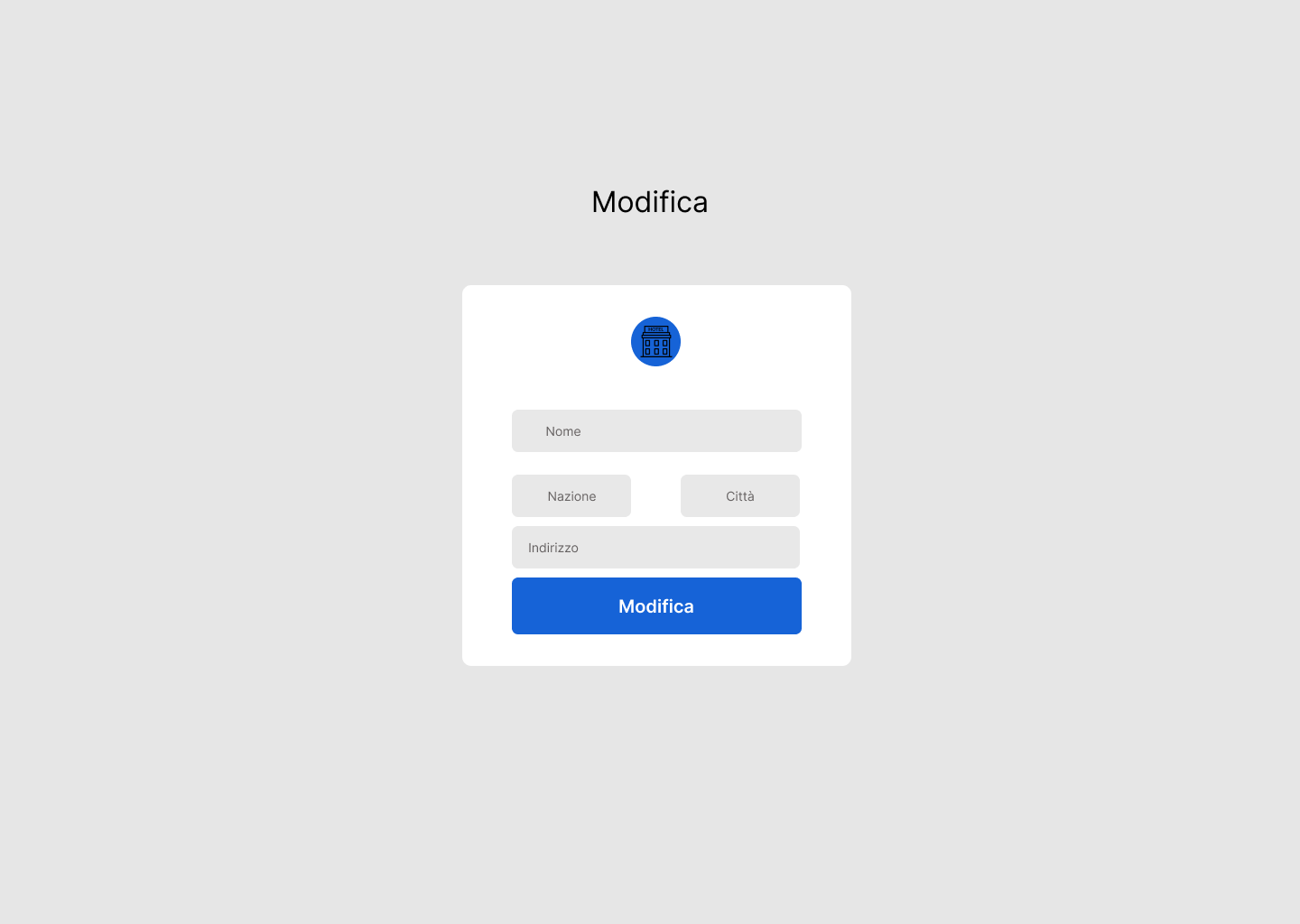
Tramite la seguente pagina si può modificare un hotel o un evento. Le pagine sono due e sono uguali, una per gli hotel ed una per gli eventi. L’unica differenza saranno i campi che andranno inseriti nel form per modificare rispettivamente l’uno o l’altro.

Figura : Pagina modifica eventi e hotel

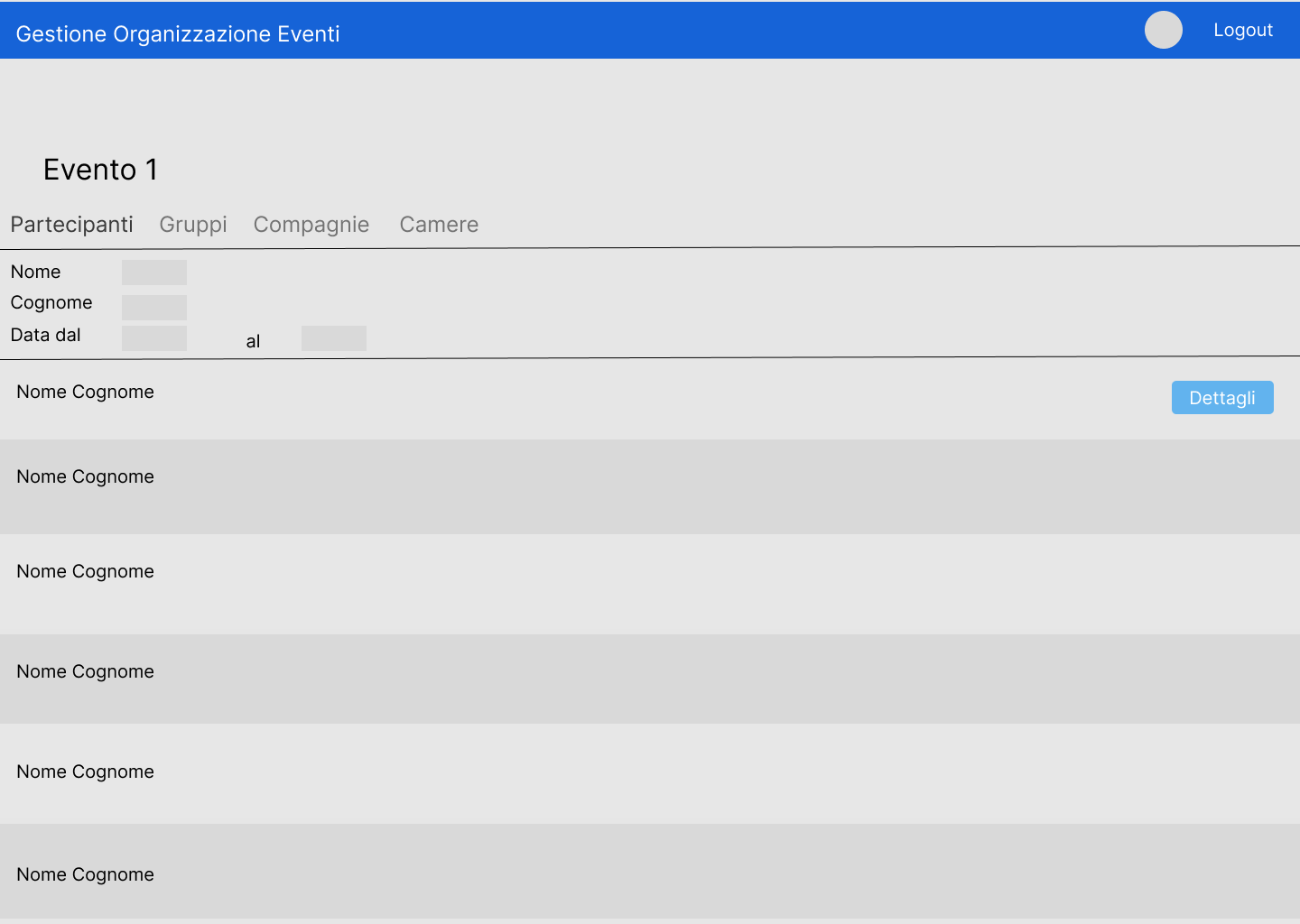


Figura : Pagina visualizzazione eventi

Questa sarà la pagina principale per ogni evento dove si potranno vedere tutte caratteristiche di un evento specifico. Sono presenti più schede per la visualizzazione di partecipanti, gruppi, compagnie e per le camere dell’hotel che ospita l’evento. Inoltre c’è una sezione dedicata alla ricerca e al filtraggio degli utenti e dei gruppi.

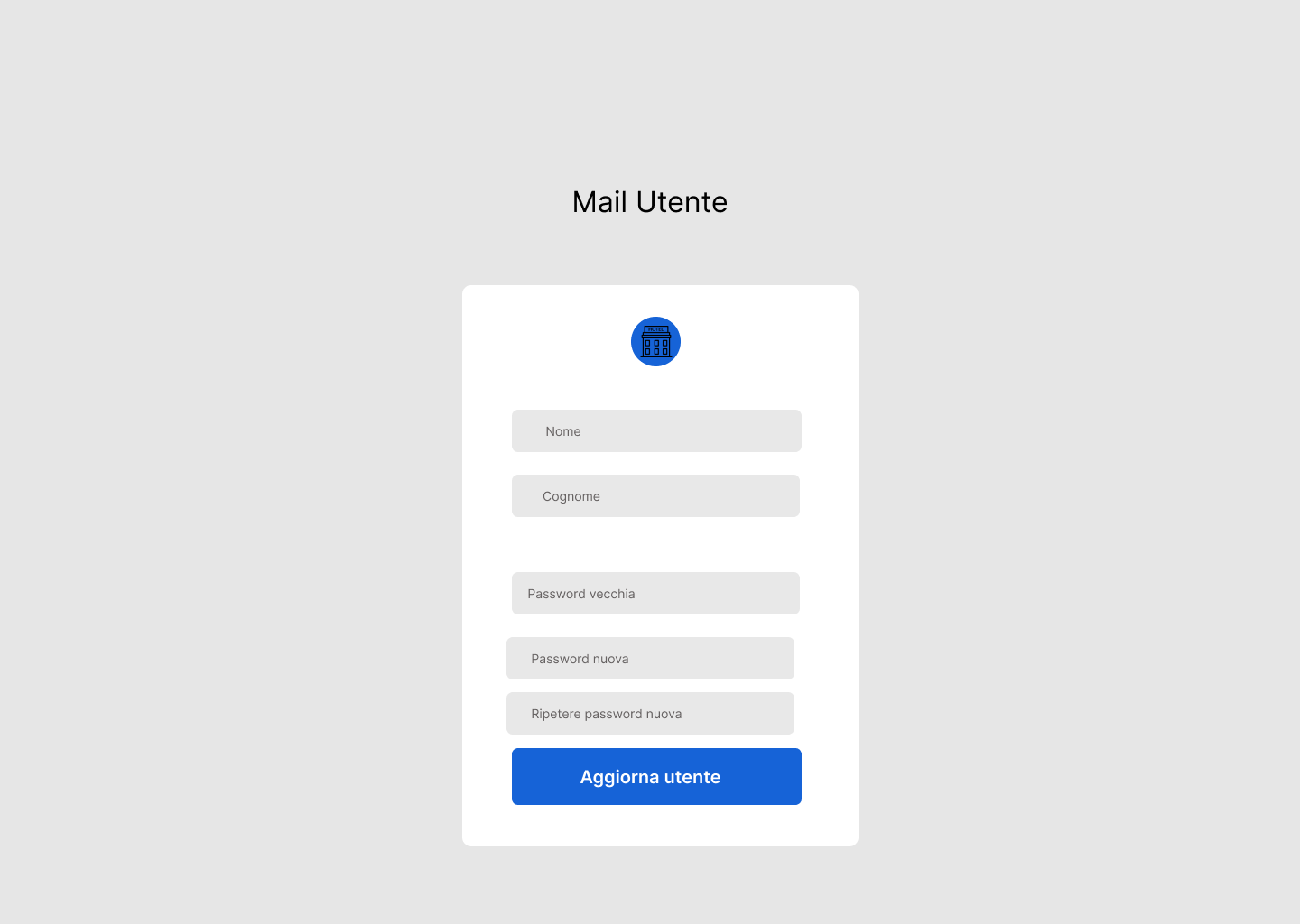
Infine, la pagine per modificare i dati del proprio utente. Al primo accesso l’utente è obbligato a cambiare la propria password che di default è *user*.

Figura : Pagina di modifica utente

# Implementazione

Per prima cosa, la schermata di login del sito è la seguente

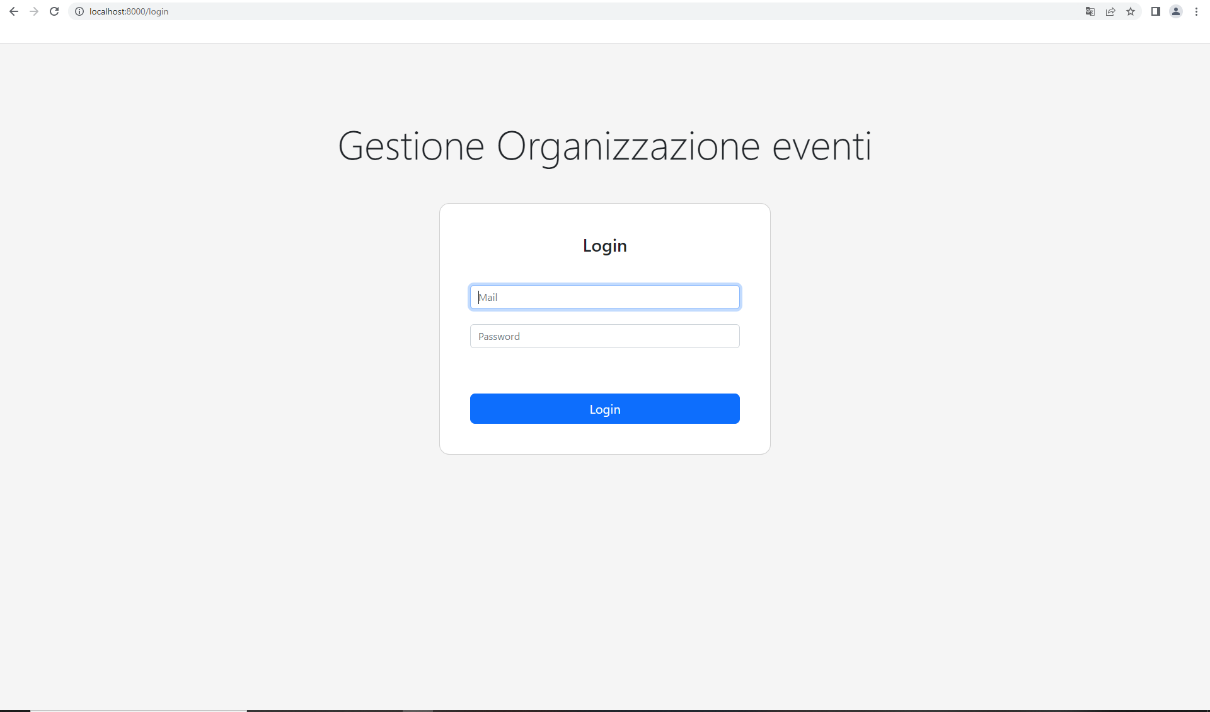


Figura : Implementazione login

La logica della pagina di login è stata implementata tramite Fortify, uno strumento per l’autenticazione backend di Laravel, non è quindi stato necessario l’utilizzo di un controller o altre classi, lo stesso discorso vale per la possibilità di cambiare password ed effettuare il logout. Esistono diversi metodi per implementare la logica di autenticazione con Laravel, si può scegliere di scrivere il codice a mano, usare npm o Fortify. Per includere Fortify nel progetto è necessario eseguire alcuni comandi da terminale:

composer require laravel/fortify

php artisan vendor:publish --provider="Laravel\Fortify\FortifyServiceProvider"

Successivamente eseguire il comando migrate per aggiornare il database

php artisan migrate

Immagine che contiene tavolo

Descrizione generata automaticamenteQuesta è la pagina principale degli utenti admin, qui vengono visualizzati tutti gli eventi e c’è la possibilità di gestirli

Figura : Implementazione lista eventi

Immagine che contiene testo

Descrizione generata automaticamenteAll’interno della view per gli eventi viene eseguito un controllo per verificare se l’utente che ha eseguito l’accesso al sito sia un utente admin oppure no. Esso viene fatto direttamente nella view.

Figura : View events.index

Se il tipo di utente è ‘admin’ viene mostrata quindi una tabella contenente anche i bottoni per la modifica e l’eliminazione di un evento mentre in caso contrario la tabella ha un solo bottone per evento, ovvero quello di dettagli.

Immagine che contiene testo

Descrizione generata automaticamenteIn entrambi i casi a passare i dati ci pensa il controller EventController, nel metodo index vengono presi i dati necessari e passati alla view.

Figura : Metodo index di EventController

Prima di tutto viene controllato se l’utente ha effettivamente eseguito l’accesso, nel caso in cui l’utente non sia amministratore vengono visualizzati solo gli eventi alla quale l’utente partecipa, questo viene fatto tramite una left join tra le tabelle events e links.

Immagine che contiene testo

Descrizione generata automaticamentePremendo il bottone elimina di un evento è possibile eliminarlo, questo avviene tramite il metodo destroy di EventController che prende come parametro l’istanza dell’evento selezionato. Questo metodo è molto semplice, ovviamente viene controllato se l’utente che lo esegue è admin e in caso positivo, viene fatta una query per l’eliminazione del record dell’evento, infine si viene reindirizzati alla pagina events.index.

Figura : Metodo destroy di EventController

Per quanto riguarda il bottone modifica invece, ci sono due metodi da prendere in considerazione, edit e update. Il primo viene richiamato premendo sul pulsante e il suo scopo è di prendere i dati necessari e passarli alla view events.edit, una pagina che contiene il form per modificare un evento.

Immagine che contiene testo, cellulare, screenshot, argento

Descrizione generata automaticamente

Figura : Metodo edit di EventController

Immagine che contiene testo

Descrizione generata automaticamenteLa view alla quale si viene mandati è la seguente, i campi vengono popolati con quelli dell’evento selezionato.

Figura : View events.edit

Immagine che contiene testo

Descrizione generata automaticamentePremendo il bottone Aggiorna Evento invece, viene richiamato il secondo metodo sopra citato, ovvero il metodo update, che si occupa di validare la richiesta e modificare effettivamente i dati all’interno del database.

Figura : Metodo update di EventController

Premendo sul bottone Dettagli invece è possibile visualizzare tutti i dettagli di un evento, ci sono diverse schede per ognuna delle caratteristiche presenti in un evento. Sono presenti le liste di utenti, gruppi compagnie e camere di un determinato evento.

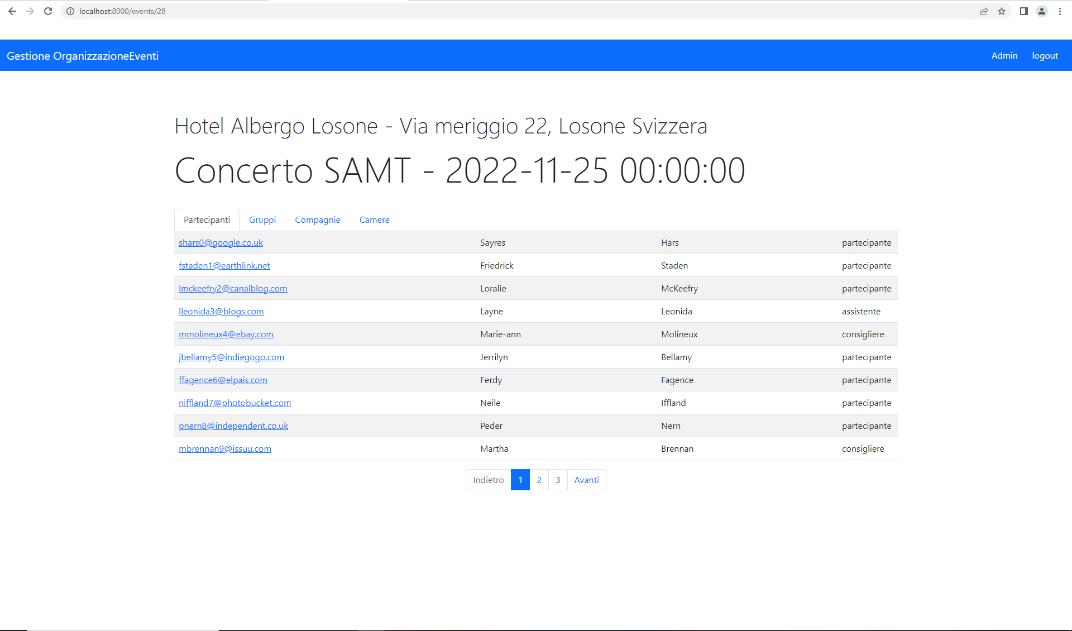


Figura : View events.show

Tutto questo viene gestito nel metodo show di EventController, tale metodo è piuttosto lungo poiché racchiude tutte queste funzionalità. Vengono fatte delle query con delle join per risalire agli utenti presenti in quel determinato evento, stessa cosa per l’hotel, le camere, i gruppi e le compagnie.

Immagine che contiene testo

Descrizione generata automaticamenteSuccessivamente viene fatto un controllo per verificare che l’utente abbia effettivamente il permesso di visualizzare i dettagli dell’evento (questo al fine di evitare di visualizzare un evento alla quale non si può accedere digitando il percorso nella barra di ricerca). Se l’utente non è admin viene quindi eseguita una query che ritorna tutti i records della tabella links dove l’id dell’utente è uguale a quello dell’utente che ha effettuato l’accesso e dove l’id dell’evento è uguale a quello dell’evento selezionato. Se tale query restituisce almeno un risultato, allora significa che l’utente è presente nell’evento e ha quindi il permesso di visualizzarne le informazioni.

Figura : Controllo permessi visualizzazione evento

Premendo sul proprio nome, ho sul link di un utente, è anche possibile visualizzare o modificare i dati del proprio utente. Questo viene fatto tramite la classe UserController, nel metodo show in cui viene passato un utente e viene ritornata la view users.show. All’interno della view viene fatto un controllo, si verifica se l’utente passato alla view è lo stesso che ha effettuato l’accesso. In questo modo quando si preme sul profilo di un altro utente vengono visualizzate le sue informazioni, mentre se lo si fa sul proprio utente è possibile cambiare le credenziali.

Immagine che contiene testo

Descrizione generata automaticamente

Figura 34: View visualizzazione e modifica del proprio utente

Figura 33: View visualizzazione di un utente

Immagine che contiene testo

Descrizione generata automaticamente

Se si cambiano dei valori e si preme il bottone Modifica viene richiamato il metodo update della classe UserController che, come nel caso degli eventi, aggiorna i dati nel database.

Una delle funzionalità fondamentali è la creazione di un evento, premendo sul bottone Crea Evento presente nella schermata principale (solo se si è admin), si viene indirizzati alla view events.create che contiene un form per la creazione di un nuovo evento. Ovviamente anche per accedere a questa view si passa prima dal controller EventController.

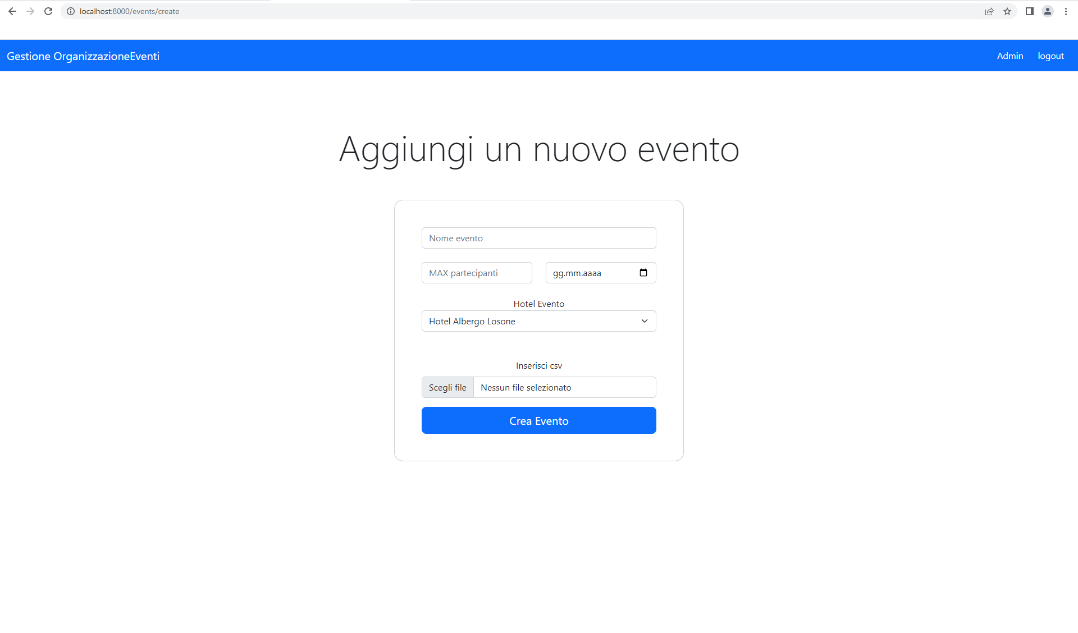


Figura : View events.create

Immagine che contiene testo

Descrizione generata automaticamenteCompilando tutti i campi e inviando il form viene invocato il metodo store di EventController, esso si occupa prima di tutto di creare l’evento e di inserire i dati necessari e successivamente di invocare il metodo uploadCsv che fa la maggior parte del lavoro.

Figura : Metodo uploadCsv di EventController

Questo metodo viene invocato subito dopo la creazione dell’evento. Quello che fa è prendere l’id dell’ultimo evento creato ed usarlo per l’inserimento di un record nella tabella links. Infatti, viene fatto un ciclo per ogni riga del file csv, ad ogni ciclo viene creato un nuovo utente e una nuova istanza di links all’interno della quale si inserisce l’utente appena creato, l’evento salvato e il gruppo che di default è quello con id 7 (un gruppo generico creato per inserire i partecipanti che non ne possiedono ancora uno). In questo modo una volta creato l’evento, gli utenti sono già tutti inseriti e i collegamenti sono stati fatti correttamente.

Premendo sul bottone Vedi Hotel apparirà una lista degli hotel disponibili. Per questa parte di implementazione viene usato il controller HotelController ma la logica dei metodi è analoga a quella usata fino ad ora (l’unica differenza è che invece di gestire gli eventi si gestiscono gli hotel). Anche questa view è accessibile solamente dagli utenti amministratori.

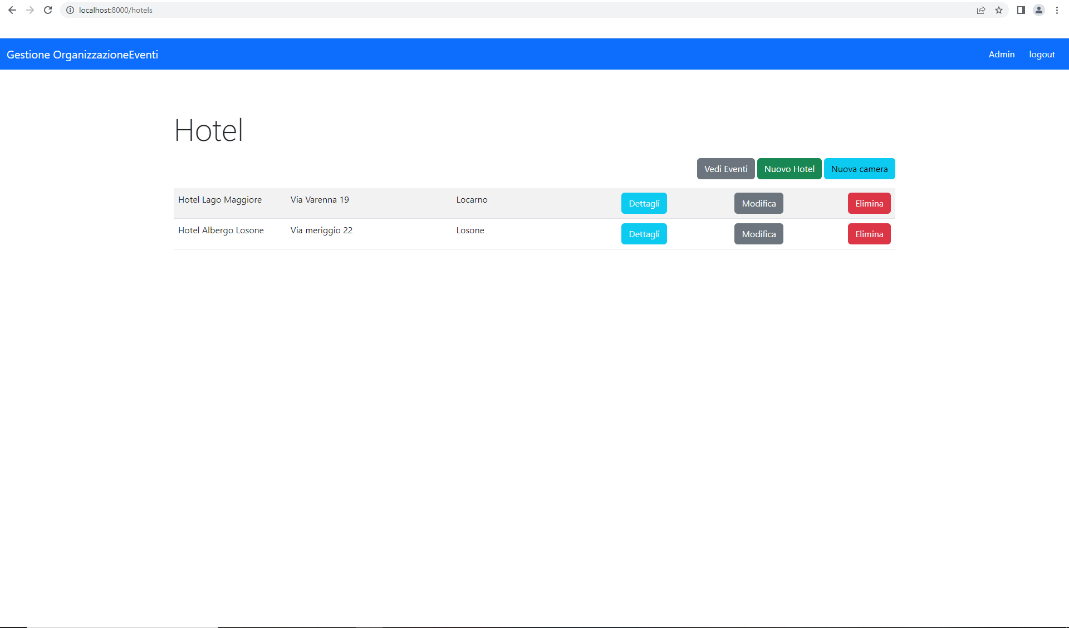


Figura : View hotels.index

Anche in questo caso, premendo il pulsante elimina, viene eseguito lo stesso procedimento usato nella classe EventController per eliminare un evento, in questo caso, se viene eliminato un hotel vengono eliminate anche tutte le camere (questo è implementato a livello di database).

Lo stesso discorso vale per il bottone modifica che ha lo stesso identico funzionamento del bottone di modifica di un evento, la view è la seguente.

Immagine che contiene testo

Descrizione generata automaticamente

Figura : View hotels.edit

Premendo sui dettagli di un hotel si visualizzano semplicemente tutte le camere. Ad occuparsi di questo è sempre il controller HotelController, viene fatta una semplice query che ritorna tutte le camere appartenenti all’hotel selezionato. Nella medesima schermata si possono anche eliminare camere, questo viene fatto tramite il controller RoomController.

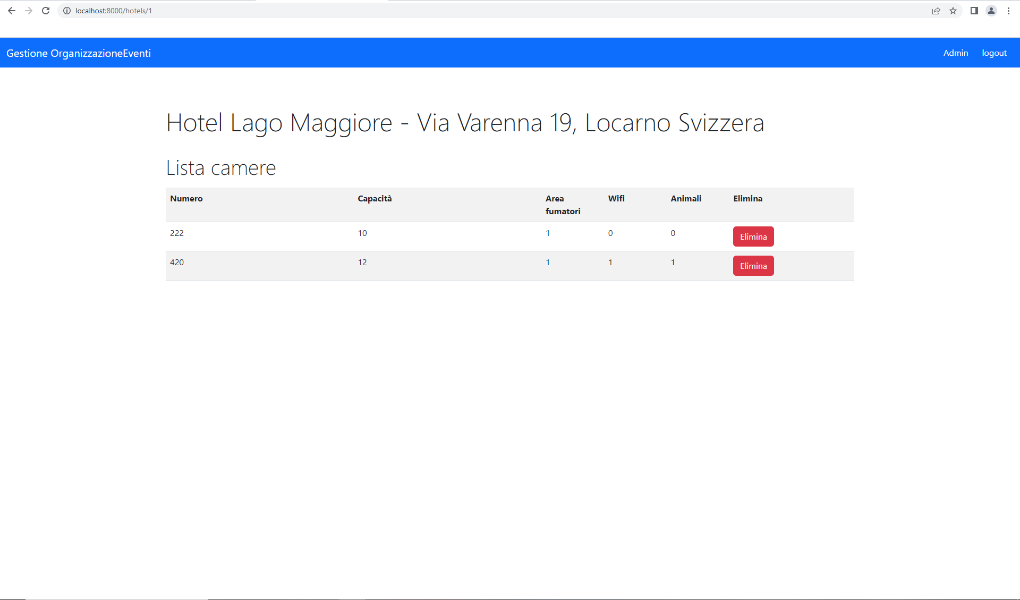


Figura : View hotels.show

C’è poi la possibilità di creare delle camere selezionando l’hotel nella quale si vuole inserire nel form della view rooms.create, gestita da RoomController. Il form è il seguente

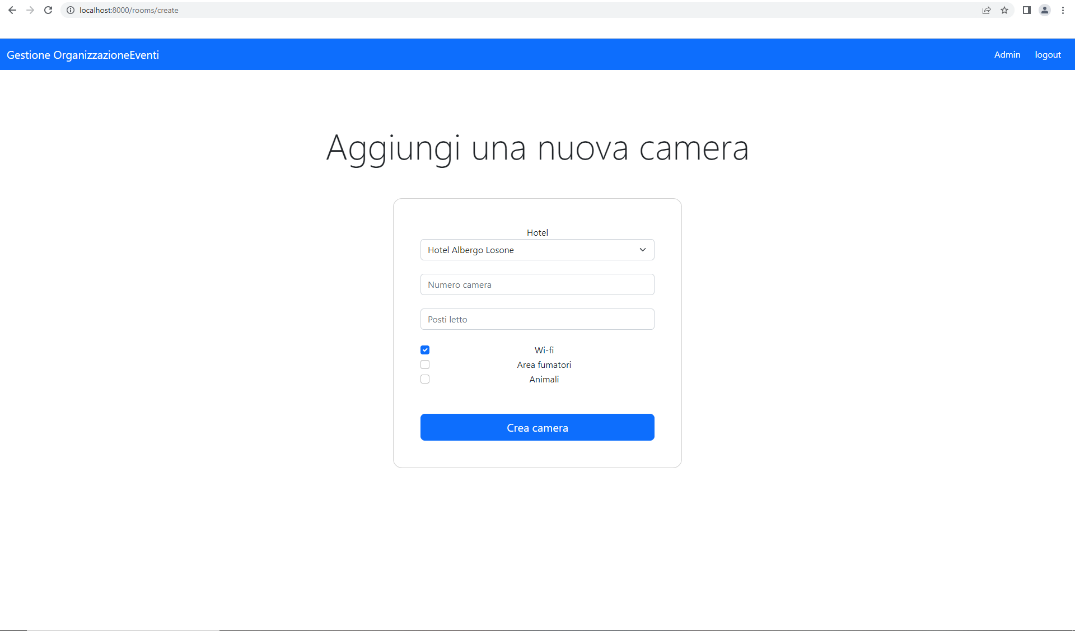


Figura : View rooms.create

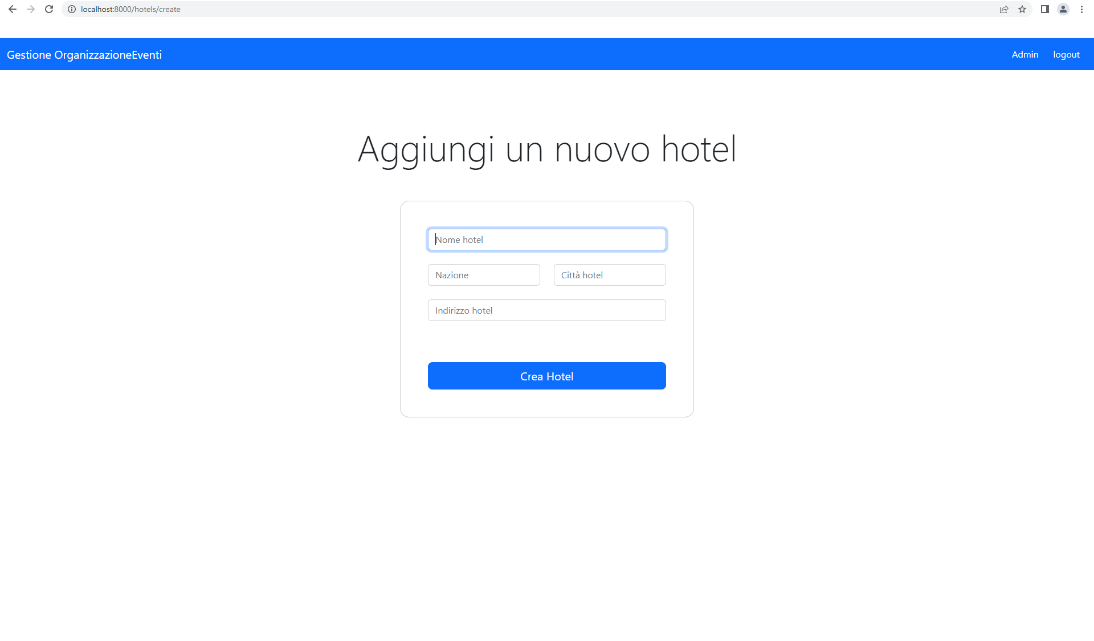
Lo stesso discorso vale anche per la creazione di un nuovo hotel.

Figura : View hotels.create

Le route sono gestite nel file web.php della cartella routes. In questo file vengono inseriti tutti i percorsi presenti nel sito, grazie a Laravel è possibile inserire delle risorse con il nome del controller per fare in modo che vengano create tutte le route necessarie. Questo però si può fare soltanto se non ci sono route personalizzate o con una struttura differente a quella fornita in automatico da Laravel, in quel caso, la rotta deve essere inserita a mano specificando i parametri e il controller che la gestisce.

Immagine che contiene testo

Descrizione generata automaticamente

Figura : file web.php contenente le routes

In molte rotte viene usato il termine middleware, essi rappresentano delle regole che devono essere soddisfatte per poter accedere a quella rotta. In questo caso si usa spesso il middleware auth che è già generato da Laravel ed integrato da Fortify, grazie a questo middleware, si potrà accedere a determinate route soltanto se si ha effettuato il login, in caso contrario l’utente viene riportato alla schermata iniziale in cui può effettuare l’accesso.

# Test

## Protocollo di test

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-001  REQ-001 | **Nome:** | Test creazione database |
| **Descrizione:** | Testare la creazione del database e la connessione dell’applicativo con esso. | | |
| **Prerequisiti:** | La struttura dell’applicativo deve essere stata creata correttamente. | | |
| **Procedura:** | 1. Avviare il servizio mysql. 2. Avviare l’applicativo MySQL Workbench. 3. Digitare il comando use dbgoe. 4. Digitare il comando show tables. | | |
| **Risultati attesi:** | Nel tool vengono visualizzate tutte le tabelle del database dbgoe. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-002  REQ-002 | **Nome:** | Test sistema di login |
| **Descrizione:** | Testare che il database permetta di fare il login e che le password siano cifrate. | | |
| **Prerequisiti:** | Database funzionante. | | |
| **Procedura:** | 1. Avviare il servizio mysql. 2. Avviare il web server di Laravel. 3. Eseguire l’accesso al sito tramite l’utente admin. 4. Fare il login nella console del database. 5. Digitare use dbgoe. 6. Digitare select password from users. | | |
| **Risultati attesi:** | Il login viene effettuato con successo e la password visibile nel database è cifrata con bycrypt. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-003  REQ-003 | **Nome:** | Test cambio password obbligatorio |
| **Descrizione:** | Testare che al primo accesso l’utente sia obbligato a cambiare la password. | | |
| **Prerequisiti:** | Logica di login funzionante. | | |
| **Procedura:** | 1. Creare un utente o utilizzare un utente non admin non ancora loggato. 2. Effettuare il login con tale utente. 3. Provare ad uscire dalla pagina di reset password. 4. Reimpostare la password utilizzandone una diversa. | | |
| **Risultati attesi:** | La password viene cambiata correttamente e al prossimo accesso bisogna inserire quella nuova. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-004  REQ-004  REQ-005 | **Nome:** | Test importazione csv e creazione evento |
| **Descrizione:** | Creare un nuovo evento ed inserire un file csv. | | |
| **Prerequisiti:** | Database e applicazione funzionante. | | |
| **Procedura:** | 1. Effettuare il login con un utente admin. 2. Premere sul bottone nuovo evento. 3. Riempire il form e selezionare un file csv valido 4. Premere il bottone crea evento. | | |
| **Risultati attesi:** | L’evento viene creato e gli utenti vengono aggiunti correttamente. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-005  REQ-006 | **Nome:** | Test modifica evento |
| **Descrizione:** | Modificare un evento esistente con un utente admin. | | |
| **Prerequisiti:** | Struttura database e controller funzionanti. | | |
| **Procedura:** | 1. Effettuare il login con un utente amministratore. 2. Selezionare un qualsiasi evento e premere il bottone modifica. 3. Modificare alcuni campi e premere aggiorna evento. | | |
| **Risultati attesi:** | L’evento selezionato viene effettivamente modificato. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-006  REQ-007 | **Nome:** | Test eliminazione evento |
| **Descrizione:** | Eliminare un evento esistente con un utente admin. | | |
| **Prerequisiti:** | Struttura database e controller funzionanti. | | |
| **Procedura:** | 1. Effettuare il login con un utente amministratore. 2. Selezionare un qualsiasi evento e premere il bottone elimina. | | |
| **Risultati attesi:** | L’evento sparisce dalla lista degli eventi visibili. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-007  REQ-008 | **Nome:** | Test lista eventi |
| **Descrizione:** | Accedere con un utente non admin e verificare che possa vedere solo gli eventi alla quale partecipa. | | |
| **Prerequisiti:** | Struttura database e controller funzionanti. | | |
| **Procedura:** | 1. Effettuare il login con un utente non admin. | | |
| **Risultati attesi:** | Viene visualizzata una lista con gli eventi alla quale l’utente è iscritto. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-008  REQ-009 | **Nome:** | Test creazione hotel |
| **Descrizione:** | Accedere con un utente admin e creare un nuovo hotel. | | |
| **Prerequisiti:** | Struttura database e controller funzionanti. | | |
| **Procedura:** | 1. Effettuare il login con un utente amministratore. 2. Premere sul bottone vedi hotel. 3. Premere sul bottone nuovo hotel e compilare il form. 4. Premere sul bottone crea hotel. | | |
| **Risultati attesi:** | L’hotel appena creato viene aggiunto alla lista degli hotel disponibili. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-009  REQ-010 | **Nome:** | Test modifica hotel |
| **Descrizione:** | Modificare un hotel esistente con un utente admin. | | |
| **Prerequisiti:** | Struttura database e controller funzionanti. | | |
| **Procedura:** | 1. Effettuare il login con un utente amministratore. 2. Selezionare un qualsiasi hotel e premere il bottone modifica. 3. Modificare alcuni campi e premere aggiorna hotel. | | |
| **Risultati attesi:** | L’hotel selezionato viene effettivamente modificato. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-010  REQ-011 | **Nome:** | Test eliminazione hotel |
| **Descrizione:** | Eliminare un hotel esistente con un utente admin. | | |
| **Prerequisiti:** | Struttura database e controller funzionanti. | | |
| **Procedura:** | 1. Effettuare il login con un utente amministratore. 2. Selezionare un qualsiasi hotel e premere il bottone elimina. | | |
| **Risultati attesi:** | L’hotel sparisce dalla lista degli hotel visibili. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-011  REQ-012 | **Nome:** | Test creazione camere |
| **Descrizione:** | Accedere con un utente admin e creare una nuova camera. | | |
| **Prerequisiti:** | Struttura database e controller funzionanti. | | |
| **Procedura:** | 1. Effettuare il login con un utente amministratore. 2. Premere sul bottone vedi hotel. 3. Premere sul bottone nuova camera e compilare il form. 4. Premere sul bottone crea camera. | | |
| **Risultati attesi:** | La camera appena creata viene aggiunta alla lista delle camere all’interno dell’hotel selezionato nel form. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-012  REQ-013  REQ-014 | **Nome:** | Test lista e dettagli hotel |
| **Descrizione:** | Accedere con un utente admin e visualizzare i dettagli di un hotel | | |
| **Prerequisiti:** | Struttura database e controller funzionanti. | | |
| **Procedura:** | 1. Effettuare il login con un utente amministratore. 2. Premere sul bottone vedi hotel. 3. Premere sul bottone dettagli di un qualsiasi hotel. | | |
| **Risultati attesi:** | Viene visualizzata la lista delle camere disponibili dell’hotel. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-013  REQ-015 | **Nome:** | Test dettagli evento |
| **Descrizione:** | Accedere con un utente e visualizzare i dettagli di un evento. | | |
| **Prerequisiti:** | Struttura database e controller funzionanti. | | |
| **Procedura:** | 1. Effettuare il login con un utente. 2. Premere sul bottone dettagli di un qualsiasi evento. | | |
| **Risultati attesi:** | Viene visualizzata una pagina con diverse schede relative ai componenti dell’evento e un form di ricerca per il filtraggio dei dati. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-014  REQ-016  REQ-017 | **Nome:** | Test organizzazione gruppi e consiglieri |
| **Descrizione:** | Accedere con un utente consigliere e creare un nuovo gruppo. | | |
| **Prerequisiti:** | Struttura database e controller funzionanti. | | |
| **Procedura:** | 1. Effettuare il login con un utente consigliere. 2. Premere sul bottone dettagli di un qualsiasi evento. 3. Premere sul bottone crea gruppo e compilare il form. 4. Premere sul bottone crea gruppo. | | |
| **Risultati attesi:** | Il gruppo viene effettivamente creato ed è visibile nella lista dei gruppi dell’evento. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-015  REQ-018  REQ-019 | **Nome:** | Test organizzazione compagnie e assistenti |
| **Descrizione:** | Accedere con un utente assistente e creare una nuova compagnia. | | |
| **Prerequisiti:** | Struttura database e controller funzionanti. | | |
| **Procedura:** | 1. Effettuare il login con un utente assistente. 2. Premere sul bottone dettagli di un qualsiasi evento. 3. Premere sul bottone crea compagnia e compilare il form. 4. Premere sul bottone crea compagnia. | | |
| **Risultati attesi:** | La compagnia viene effettivamente creata ed è visibile nella lista delle compagnie dell’evento. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-016  REQ-020 | **Nome:** | Test algoritmo di assegnazione. |
| **Descrizione:** | Creare un nuovo evento ed inserire un file csv ed un hotel esistente. | | |
| **Prerequisiti:** | Database e applicazione funzionante. | | |
| **Procedura:** | 1. Effettuare il login con un utente admin. 2. Premere sul bottone nuovo evento. 3. Riempire il form e selezionare un file csv valido 4. Premere il bottone crea evento. | | |
| **Risultati attesi:** | Gli utenti vengono assegnati automaticamente alle camere e divisi per sesso ed età. | | |

## Risultati test

|  |  |  |  |
| --- | --- | --- | --- |
| **Test case** | **Esito** | **Risultati** | **Data** |
| TC-001 | **TEST PASSATO** | Tutte le tabelle del database vengono visualizzate correttamente.  Figura : TC-001 | 06.12.2022 |
| TC-002 | **TEST PASSATO** | Le password sono cifrate correttamente e viene eseguito il salting.  Figura : TC-002 | 06.12.2022 |
| TC-003 | **TEST PASSATO** | Al primo login l’utente è obbligato a modificare la password che viene cambiata correttamente.  Figura : TC-003 | 06.12.2022 |
| TC-004 | **TEST PASSATO** | L’evento viene creato correttamente e accedendo alla schermata di dettaglio si possono visualizzare tutti gli utenti inseriti tramite il file csv.  Figura : TC-004 | 06.12.2022 |
| TC-005 | **TEST PASSATO** | L’evento viene modificato e i dati nuovi sono visualizzabili nella lista degli eventi. | 06.12.2022 |
| TC-006 | **PASSATO** | L’evento viene eliminato, gli utenti restano nel database ma i collegamenti con l’evento non sono più presenti. | 06.12.2022 |
| TC-007 | **TEST PASSATO** | L’utente vede unicamente la lista degli eventi a cui è iscritto.  Figura : TC-007 | 06.12.2022 |
| TC-008 | **TEST PASSATO** | L’hotel viene creato e aggiunto alla lista degli hotel disponibili.  Figura : TC-008 | 06.12.2022 |
| TC-009 | **TEST PASSATO** | L’hotel viene modificato e i dati nuovi sono visualizzabili nella lista degli hotel disponibili. | 06.12.2022 |
| TC-010 | **TEST PASSATO** | L’hotel viene eliminato e anche le camere al suo interno, se l’hotel è usato per un evento è necessario prima eliminare tale evento. | 06.12.2022 |
| TC-011 | **TEST PASSATO** | La camera viene creata e aggiunta alla lista delle camere dell’hotel selezionato nel form.  Figura : TC-011 | 06.12.2022 |
| TC-012 | **TEST PASSATO** | Vengono visualizzati i dettagli dell’hotel e la lista di tutte le camere.  Figura : TC-012 | 06.12.2022 |
| TC-013 | **TEST PASSATO** | Vengono visualizzati i dettagli dell’evento e le diverse schede disponibili.  Figura : TC-013 | 06.12.2022 |
| TC-014 | **TEST NON PASSATO** | La logica per distinguere il tipo di utente consigliere funziona ma il gruppo non viene aggiunto alla lista dei gruppi disponibili.  Figura : TC-014 | 06.12.2022 |
| TC-015 | **TEST NON PASSATO** | La logica per distinguere il tipo di utente assistente funziona ma non viene visualizzata la possibilità di aggiungere compagnie e svolgere funzionalità inerenti al ruolo. | 06.12.2022 |
| TC-016 | **TEST NON PASSATO** | L’algoritmo di assegnazione non è stato implementato nel codice e non viene eseguita l’assegnazione alle camere. | 06.12.2022 |

## Mancanze/limitazioni conosciute

Purtroppo, non sono stati soddisfatti tutti i requisiti del progetto, questo per lo più per mancanze legate al tempo più che per motivi di limitazioni nell’implementazione e nella scrittura del codice. Infatti, essendo stata la mia prima esperienza con un framework come Laravel, alcune ore di progetto sono state dedicate all’apprendimento e alla lettura della documentazione di questo strumento. I requisiti mancanti a progetto terminato sono comunque prevalentemente di priorità minore, come ad esempio la gestione di compagnie o la gestione dei gruppi da parte dei membri dello staff.

# Consuntivo

Di seguito è rappresentato il diagramma di Gantt consuntivo, esso rappresenta ciò che ho effettivamente fatto durante il progetto e non corrisponde perfettamente con la pianificazione iniziale. Infatti, sebbene la fase iniziale e quella di progettazione sono piuttosto fedeli a quanto pianificato, per quanto riguarda l’implementazione del codice e la fase di test c’è molta differenza. Questo a mio avviso è perché, non avendo mai utilizzato Laravel, alcune procedure e dinamiche non le conoscevo e pensavo di svolgere il lavoro in modo diverso. Per quanto riguarda i test, non avendo terminato il progetto, ho dovuto dedicare meno tempo ma sono comunque riuscito a svolgerli in modo ottimale.

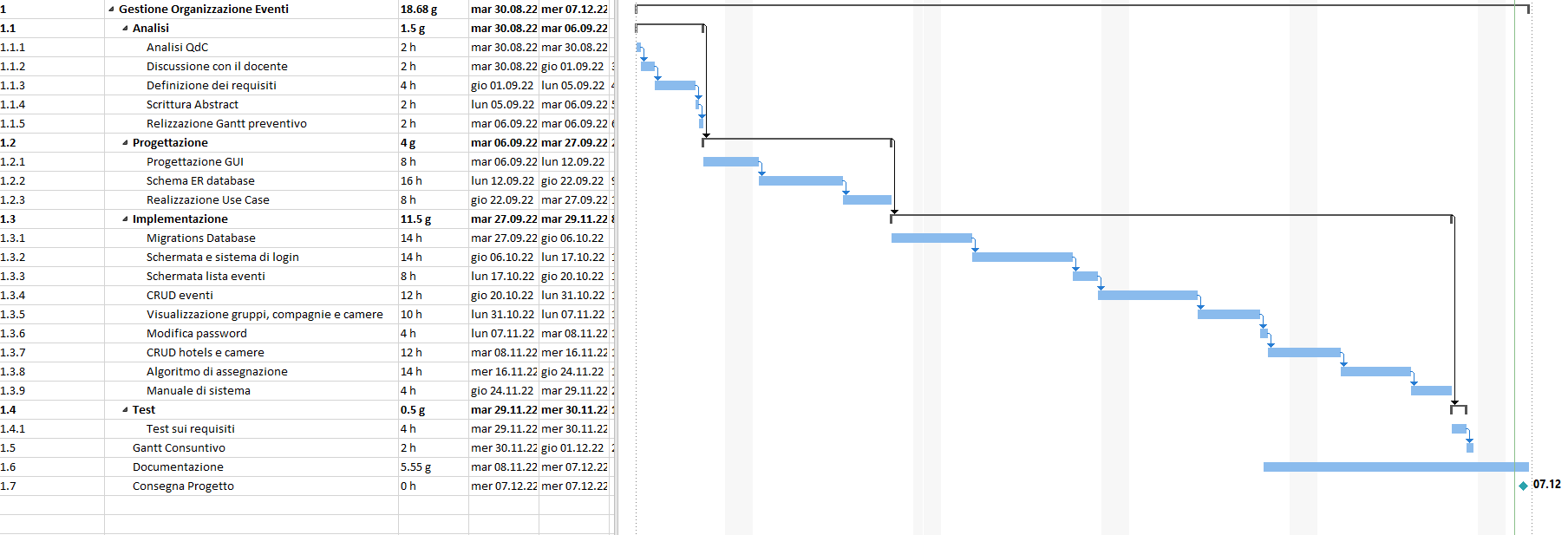


Figura : Diagramma di Gantt consuntivo

# Conclusioni

## Sviluppi futuri

Non avendo terminato il progetto, sono diversi gli sviluppi futuri possibili per migliorarlo. In primo luogo, ovviamente, si potrebbero implementare tutti i requisiti mancanti che non sono stati ancora realizzati, si potrebbe inoltre migliorare l’algoritmo di assegnazione per renderlo più efficiente e versatile per situazioni differenti. Il sito si potrebbe inoltre rendere accessibile localmente nella scuola tramite le credenziali fornite dal docente responsabile.

## Considerazioni personali

Anche se sono riuscito a rispettare la maggior parte dei requisiti del progetto e ho avuto la possibilità di imparare ad utilizzare un framework nuovo, non sono comunque pienamente soddisfatto dello svolgimento di questo progetto. Questo perché alla fine ci sono comunque dei requisiti mancanti che avrei voluto completare e ritengo che avrei potuto svolgere meglio la fase di progettazione, alla quale ho dovuto mettere mano più volte anche dopo la fase iniziale del progetto. Penso quindi che avrei dovuto dedicare più tempo per progettare l’applicazione e magari discutere maggiormente con il docente responsabile per i requisiti di progetto. Detto questo, è stato comunque molto utile e istruttivo, visto che non avevo mai realizzato un gestionale di questo tipo interamente da solo. Inoltre, anche se ho perso diverso tempo dietro a problemi legati a Laravel, penso che grazie a questo progetto in futuro riuscirò ad evitare le stesse problematiche e quindi potrei risparmiare diverso tempo nella realizzazione dei progetti successivi.

# Glossario

|  |  |
| --- | --- |
| **Termine** | **Descrizione** |
| Backend | Parte di programma non visibile dagli utenti. |
| Blade | Template Engine utilizzato dal framework Laravel. |
| Bootstrap | Bootstrap è un framework per il CSS. |
| Bycrypt | Funzione di hashing per la cifratura di stringhe. |
| CSS | Permette di modificare lo stile di una pagina Html. |
| Database | Banca dati, struttura all’interno della quale vengono salvati i dati. |
| Form | Un form è un’area all’interno di una pagina web all’interno della quale si possono inserire dei dati. |
| Framework | Sistema che consente di estendere le capacità di un linguaggio di programmazione aggiungendo e semplificando delle funzionalità. |
| Frontend | Parte di programma con la quale l’utente può interfacciarsi. |
| Gantt | Un diagramma che ha lo scopo di pianificare le attività di un progetto. |
| HTML | Linguaggio usato per le pagine web. |
| Javascript | Linguaggio di programmazione. |
| Laravel | Laravel è un framework PHP. |
| Login | L’azione di effettuare l’accesso su una piattaforma tramite delle credenziali. |
| MVC | Struttura di cartelle volta alla creazione di un applicativo. |
| MySQL | MySQL è un sistema di gestione di database relazionali. |
| PHP | Linguaggio di programmazione backend. |
| Salt | Il salt è l’azione di aggiungere una stringa prima o dopo un’altra stringa per fare in modo che il processo per passare da un hash alla stringa originale sia più complesso |
| Schema ER | Diagramma che rappresenta la struttura di un database. |
| Template Engine | Utilizzato per svolgere il templating, sostituire parti di codice Html per rendere un sito dinamico. |
| Use Case | Casi d’uso, è uno schema che contiene tutte le azioni che si possono effettuare in un sistema. |
| Waterfall | Metodologia di progettazione in cui le attività vengono svolte in modo sequenziale. |

# Bibliografia

## Bibliografia per articoli di riviste:

Non è stata utilizzata alcuna rivista.

## Bibliografia per libri

Non è stata utilizzato alcun libro.

## Sitografia

<https://laravel.com/docs/9.x>*, Laravel documentation*, 09-2022.

<https://www.positronx.io/laravel-import-expert-excel-and-csv-file-tutorial-with-example/>, *import a csv file*, 10-2022

<https://techvblogs.com/blog/laravel-import-export-excel-csv-file>*, import csv in Laravel*, 11-2022

<https://stackoverflow.com/>, *problems solving*, 09-2022

<https://laracasts.com/>, *problems solving about Laravel*, 10-2022

<https://www.youtube.com/@lesstheoryacademy6051/playlists>*, Laravel basic usage*, 09-2022

<https://codebrisk.com/blog/laravel-fortify-backend-controllers-scaffolding-for-auth>, *Fortify guide*, 10-2022

## Indice delle figure

[Figura 1: Use Case 11](#_Toc121296998)

[Figura 2: Diagramma di Gantt preventivo 12](file:///C:\Users\gianni.grasso\gestione-organizzazione-eventi\3_Documentazione\Documentazione.docx#_Toc121296999)

[Figura 3: Schema struttura MVC 14](file:///C:\Users\gianni.grasso\gestione-organizzazione-eventi\3_Documentazione\Documentazione.docx#_Toc121297000)

[Figura 4: Struttura cartelle di un progetto Laravel 15](file:///C:\Users\gianni.grasso\gestione-organizzazione-eventi\3_Documentazione\Documentazione.docx#_Toc121297001)

[Figura 5: Esempio di un file *.env* 15](file:///C:\Users\gianni.grasso\gestione-organizzazione-eventi\3_Documentazione\Documentazione.docx#_Toc121297002)

[Figura 6: Cartella tests 16](file:///C:\Users\gianni.grasso\gestione-organizzazione-eventi\3_Documentazione\Documentazione.docx#_Toc121297003)

[Figura 7: Cartella routes 16](file:///C:\Users\gianni.grasso\gestione-organizzazione-eventi\3_Documentazione\Documentazione.docx#_Toc121297004)

[Figura 8: Cartella views in resources 16](file:///C:\Users\gianni.grasso\gestione-organizzazione-eventi\3_Documentazione\Documentazione.docx#_Toc121297005)

[Figura 9: Cartella database 17](file:///C:\Users\gianni.grasso\gestione-organizzazione-eventi\3_Documentazione\Documentazione.docx#_Toc121297006)

[Figura 10: Cartella Controllers e Models in app 17](file:///C:\Users\gianni.grasso\gestione-organizzazione-eventi\3_Documentazione\Documentazione.docx#_Toc121297007)

[Figura 11: Schema ER del database 18](file:///C:\Users\gianni.grasso\gestione-organizzazione-eventi\3_Documentazione\Documentazione.docx#_Toc121297008)

[Figura 12: Schermata di login 19](file:///C:\Users\gianni.grasso\gestione-organizzazione-eventi\3_Documentazione\Documentazione.docx#_Toc121297009)

[Figura 13: Schermata home per utenti admin 19](file:///C:\Users\gianni.grasso\gestione-organizzazione-eventi\3_Documentazione\Documentazione.docx#_Toc121297010)

[Figura 14: Lista di hotel per utenti admin 20](file:///C:\Users\gianni.grasso\gestione-organizzazione-eventi\3_Documentazione\Documentazione.docx#_Toc121297011)

[Figura 15: Schermata home per utenti consiglieri e partecipanti 20](file:///C:\Users\gianni.grasso\gestione-organizzazione-eventi\3_Documentazione\Documentazione.docx#_Toc121297012)

[Figura 16: Pagina creazione hotel 21](file:///C:\Users\gianni.grasso\gestione-organizzazione-eventi\3_Documentazione\Documentazione.docx#_Toc121297013)

[Figura 17: Pagina creazione camera 21](file:///C:\Users\gianni.grasso\gestione-organizzazione-eventi\3_Documentazione\Documentazione.docx#_Toc121297014)

[Figura 18: Pagina creazione evento 22](file:///C:\Users\gianni.grasso\gestione-organizzazione-eventi\3_Documentazione\Documentazione.docx#_Toc121297015)

[Figura 19: Pagina dettagli hotel 22](file:///C:\Users\gianni.grasso\gestione-organizzazione-eventi\3_Documentazione\Documentazione.docx#_Toc121297016)

[Figura 20: Pagina modifica eventi e hotel 23](file:///C:\Users\gianni.grasso\gestione-organizzazione-eventi\3_Documentazione\Documentazione.docx#_Toc121297017)

[Figura 21: Pagina visualizzazione eventi 23](file:///C:\Users\gianni.grasso\gestione-organizzazione-eventi\3_Documentazione\Documentazione.docx#_Toc121297018)

[Figura 22: Pagina di modifica utente 24](file:///C:\Users\gianni.grasso\gestione-organizzazione-eventi\3_Documentazione\Documentazione.docx#_Toc121297019)

[Figura 23: Implementazione login 25](file:///C:\Users\gianni.grasso\gestione-organizzazione-eventi\3_Documentazione\Documentazione.docx#_Toc121297020)

[Figura 24: Implementazione lista eventi 26](file:///C:\Users\gianni.grasso\gestione-organizzazione-eventi\3_Documentazione\Documentazione.docx#_Toc121297021)

[Figura 25: View events.index 26](file:///C:\Users\gianni.grasso\gestione-organizzazione-eventi\3_Documentazione\Documentazione.docx#_Toc121297022)

[Figura 26: Metodo index di EventController 27](file:///C:\Users\gianni.grasso\gestione-organizzazione-eventi\3_Documentazione\Documentazione.docx#_Toc121297023)

[Figura 27: Metodo destroy di EventController 27](file:///C:\Users\gianni.grasso\gestione-organizzazione-eventi\3_Documentazione\Documentazione.docx#_Toc121297024)

[Figura 28: Metodo edit di EventController 28](file:///C:\Users\gianni.grasso\gestione-organizzazione-eventi\3_Documentazione\Documentazione.docx#_Toc121297025)

[Figura 29: View events.edit 28](file:///C:\Users\gianni.grasso\gestione-organizzazione-eventi\3_Documentazione\Documentazione.docx#_Toc121297026)

[Figura 30: Metodo update di EventController 29](file:///C:\Users\gianni.grasso\gestione-organizzazione-eventi\3_Documentazione\Documentazione.docx#_Toc121297027)

[Figura 31: View events.show 29](file:///C:\Users\gianni.grasso\gestione-organizzazione-eventi\3_Documentazione\Documentazione.docx#_Toc121297028)

[Figura 32: Controllo permessi visualizzazione evento 30](file:///C:\Users\gianni.grasso\gestione-organizzazione-eventi\3_Documentazione\Documentazione.docx#_Toc121297029)

[Figura 33: View events.create 32](file:///C:\Users\gianni.grasso\gestione-organizzazione-eventi\3_Documentazione\Documentazione.docx#_Toc121297030)

[Figura 34: Metodo uploadCsv di EventController 32](file:///C:\Users\gianni.grasso\gestione-organizzazione-eventi\3_Documentazione\Documentazione.docx#_Toc121297031)

[Figura 35: View hotels.index 33](file:///C:\Users\gianni.grasso\gestione-organizzazione-eventi\3_Documentazione\Documentazione.docx#_Toc121297032)

[Figura 36: View hotels.edit 34](file:///C:\Users\gianni.grasso\gestione-organizzazione-eventi\3_Documentazione\Documentazione.docx#_Toc121297033)

[Figura 37: View hotels.show 34](file:///C:\Users\gianni.grasso\gestione-organizzazione-eventi\3_Documentazione\Documentazione.docx#_Toc121297034)

[Figura 38: View rooms.create 35](file:///C:\Users\gianni.grasso\gestione-organizzazione-eventi\3_Documentazione\Documentazione.docx#_Toc121297035)

[Figura 39: View hotels.create 35](file:///C:\Users\gianni.grasso\gestione-organizzazione-eventi\3_Documentazione\Documentazione.docx#_Toc121297036)

[Figura 40: file web.php contenente le routes 36](file:///C:\Users\gianni.grasso\gestione-organizzazione-eventi\3_Documentazione\Documentazione.docx#_Toc121297037)

[Figura 41: TC-001 41](file:///C:\Users\gianni.grasso\gestione-organizzazione-eventi\3_Documentazione\Documentazione.docx#_Toc121297038)

[Figura 42: TC-002 41](file:///C:\Users\gianni.grasso\gestione-organizzazione-eventi\3_Documentazione\Documentazione.docx#_Toc121297039)

[Figura 43: TC-003 42](file:///C:\Users\gianni.grasso\gestione-organizzazione-eventi\3_Documentazione\Documentazione.docx#_Toc121297040)

[Figura 44: TC-004 42](file:///C:\Users\gianni.grasso\gestione-organizzazione-eventi\3_Documentazione\Documentazione.docx#_Toc121297041)

[Figura 45: TC-007 42](file:///C:\Users\gianni.grasso\gestione-organizzazione-eventi\3_Documentazione\Documentazione.docx#_Toc121297042)

[Figura 46: TC-008 43](file:///C:\Users\gianni.grasso\gestione-organizzazione-eventi\3_Documentazione\Documentazione.docx#_Toc121297043)

[Figura 47: TC-011 43](file:///C:\Users\gianni.grasso\gestione-organizzazione-eventi\3_Documentazione\Documentazione.docx#_Toc121297044)

[Figura 48: TC-012 44](file:///C:\Users\gianni.grasso\gestione-organizzazione-eventi\3_Documentazione\Documentazione.docx#_Toc121297045)

[Figura 49: TC-013 44](file:///C:\Users\gianni.grasso\gestione-organizzazione-eventi\3_Documentazione\Documentazione.docx#_Toc121297046)

[Figura 50: TC-014 44](file:///C:\Users\gianni.grasso\gestione-organizzazione-eventi\3_Documentazione\Documentazione.docx#_Toc121297047)

[Figura 51: Diagramma di Gantt consuntivo 46](file:///C:\Users\gianni.grasso\gestione-organizzazione-eventi\3_Documentazione\Documentazione.docx#_Toc121297048)

# Allegati

Ecco l’elenco degli allegati:

* Diari di lavoro
* Codici sorgente
* Manuale di sistema
* Qdc