

SITO WEB PER LA GESTIONE DI LOCAZIONI A STUDENTI

Progetto Didattico
Tecnologie Web
Gruppo 17
Anno Accademico 2021-2022

Studenti:

Bartolini Lorenzo
Messina Francisco
Kchelfi Mohamed Amine
Turhani Klea

Professore:

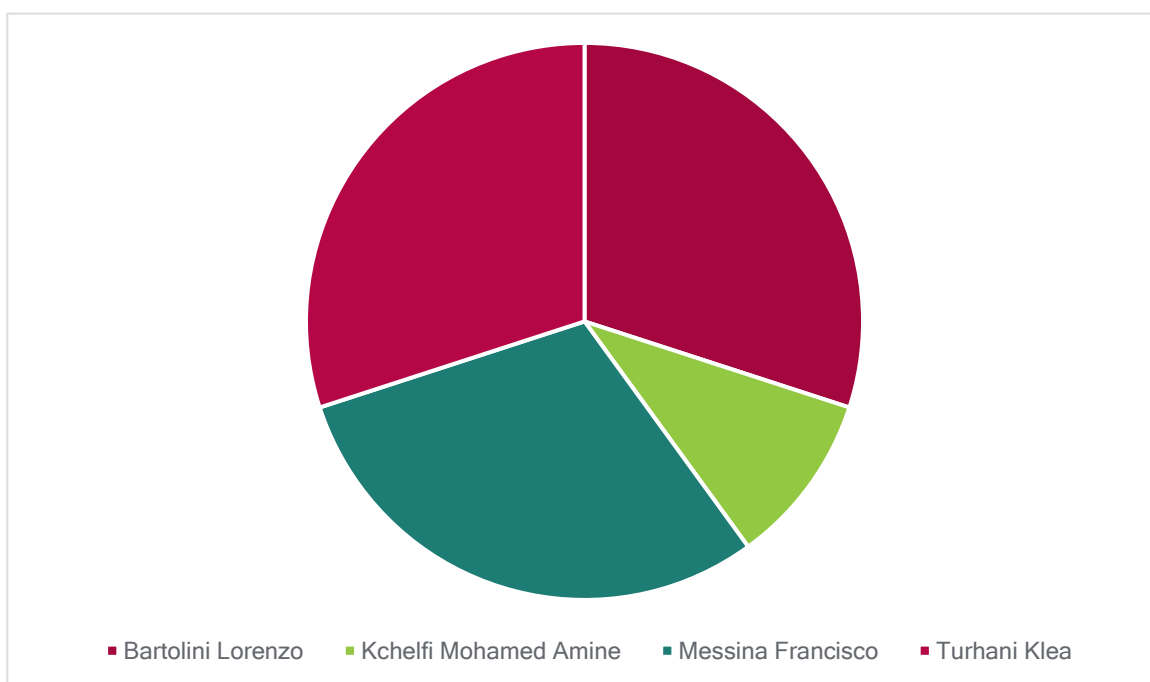
Cucchiarelli Alessandro



Contributo al Progetto

Espresso in percentuale

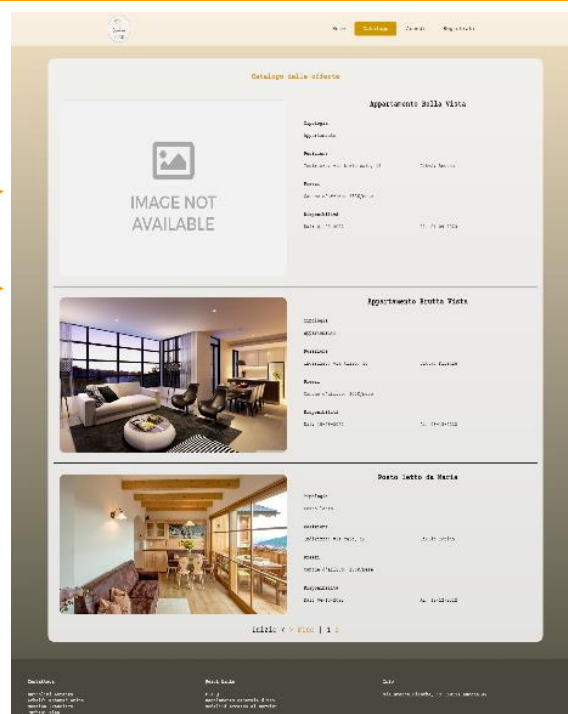
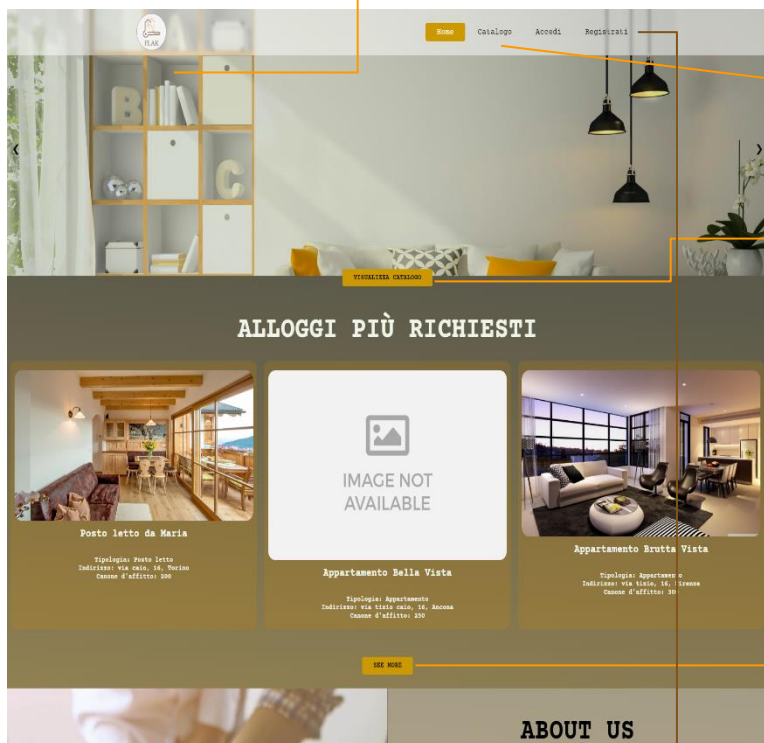
Bartolini Lorenzo	30%
Kchelfi Mohamed Amine	10%
Messina Francisco	30%
Turhani Klea	30%



Descrizione del sito Livello 1

Il **logo** contiene le iniziali dei nostri nomi.

La **navbar** è composta da:
Logo, Home, Catalogo, Accedi e Registrati.



CATALOGO

Tutti gli utenti, anche quelli **non registrati**, possono visualizzare tutte le offerte, cliccando su **Catalogo**.

Ogni scheda mostra il **nome** dell'alloggio, la **foto**, se disponibile, e alcune informazioni di riepilogo. Cliccando sulla scheda si accede alla pagina dell'alloggio, che contiene tutte le informazioni complete dell'alloggio.

Area di registrazione

Chi sei?

Locatore

Nome

NomeConUnNumero111

Inserire solo lettere maiuscole o minuscole

Cognome

Inserisci cognome

Username

Inserisci username

Email

EmailErrata

E-mail non valida

Gender

REGISTRATI

Questa è la *form di Registrazione*.

Una volta compilati tutti i campi, l'utente viene registrato e il sito effettua automaticamente il login.

Se l'utente ha già un account può cliccare direttamente sul bottone **Accedi** nella navbar.

Su questa form è attiva la **validazione lato client**, oltre che a quella **lato server**. Ogni volta che l'utente inserisce un valore in un elemento di input, l'evento *onchange* attiva una funzione JavaScript che aggiunge nella pagina, in corrispondenza dell'elemento di input modificato, un'indicazione sugli eventuali errori che l'utente ha commesso durante l'inserimento.

Il **footer** contiene i **contatti** degli sviluppatori, i **quick links** (Faq, Regolamento generale d'uso e Modalità accesso ai servizi) e altre **info** (la posizione dell'università e la documentazione).

Livello 2 (LOCATORE)

Una volta che l'utente ha eseguito l'accesso come locatore, la navbar cambierà di conseguenza, aggiungendo le sezioni Account, Messaggi, I miei Alloggi e Logout.

L'utente esegue il logout e ritorna alla homepage pubblica.

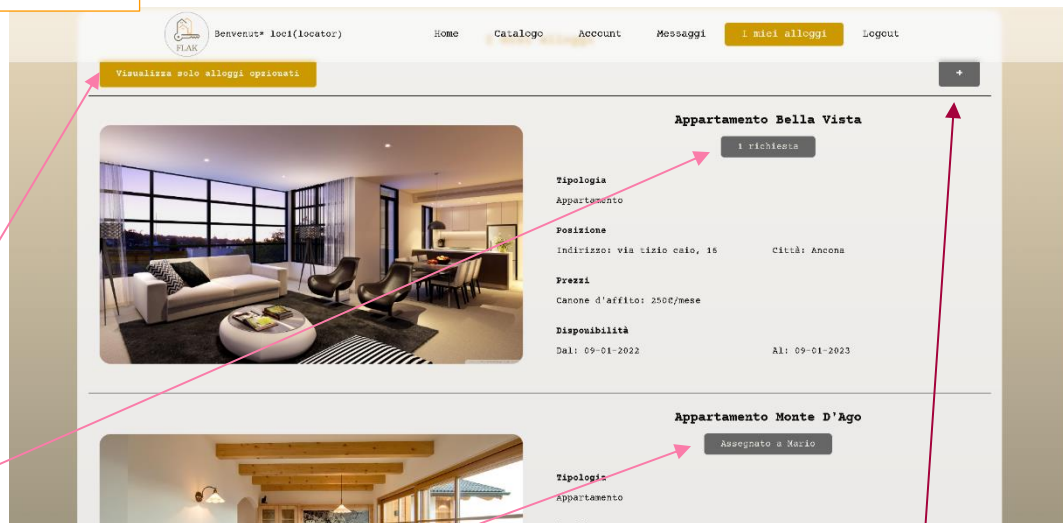
I MIEI ALLOGGI

Il locatore consulta in questa pagina la lista di tutte le offerte che ha pubblicato.

Cliccando sul tasto **Visualizza solo alloggi opzionati**, viene mostrata la lista dei soli alloggi a cui qualcuno ha fatto richiesta. Non vengono mostrati gli alloggi assegnati.

In ogni scheda alloggio il locatore visiona il **numero delle richieste** effettuate dai locatari, se l'alloggio non è stato assegnato.

In caso contrario, viene mostrato il nome dell'utente a cui l'alloggio è stato **assegnato**.



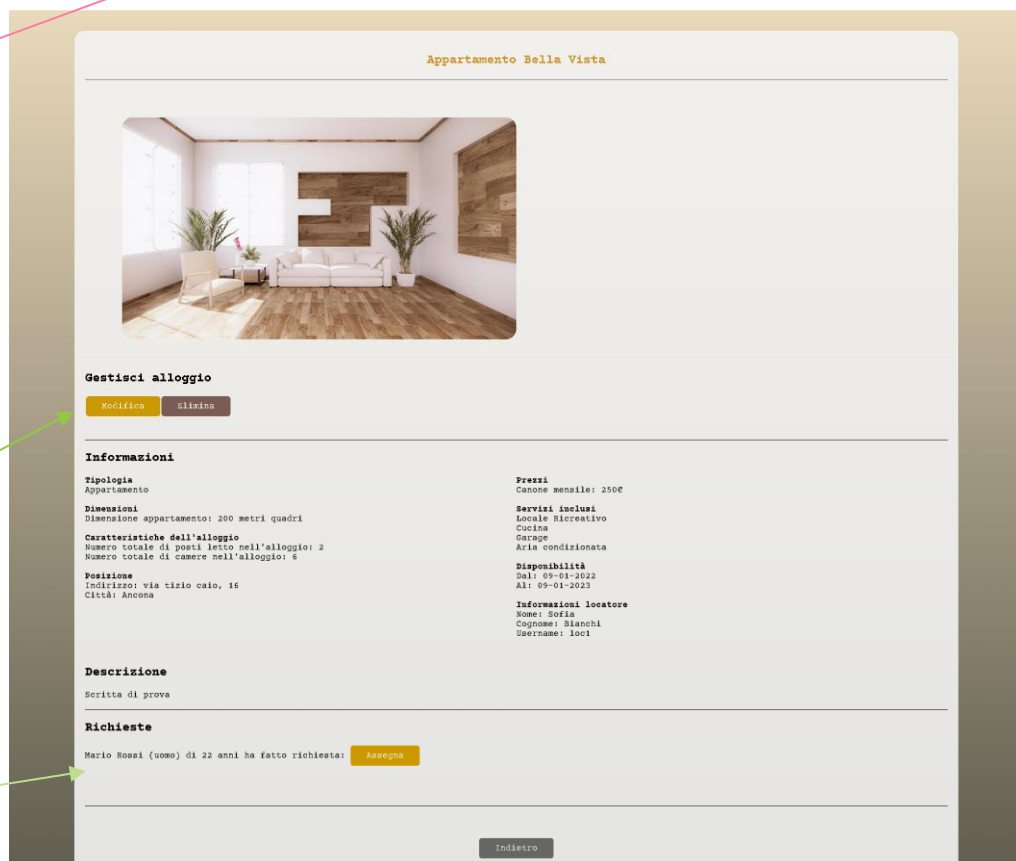
Cliccando su questo pulsante il locatore può inserire un alloggio.

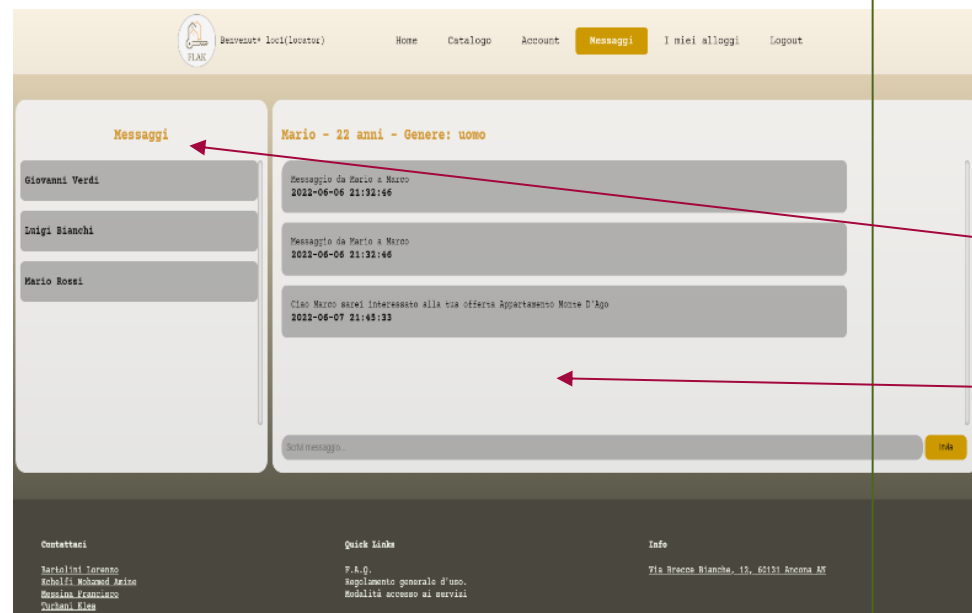
DETTAGLI ALLOGGIO

Se il locatore clicca sul nome dell'appartamento o del posto letto, vede le **informazioni** relative all'alloggio.

Quando il locatore visualizza la scheda dell'alloggio. Qui, per i soli proprietari dell'alloggio, vengono mostrati due bottoni per **modificare** o **cancellare** l'alloggio. **Non è più possibile modificare o cancellare l'alloggio una volta assegnato.**

Nella parte bassa della pagina è presente la lista con tutte le **richieste** ricevute. Il locatore può decidere a quale utente assegnare l'alloggio, cliccando sul bottone **Assegna**, mostrato a fianco del nome dell'utente che ha fatto richiesta.





MESSAGGI

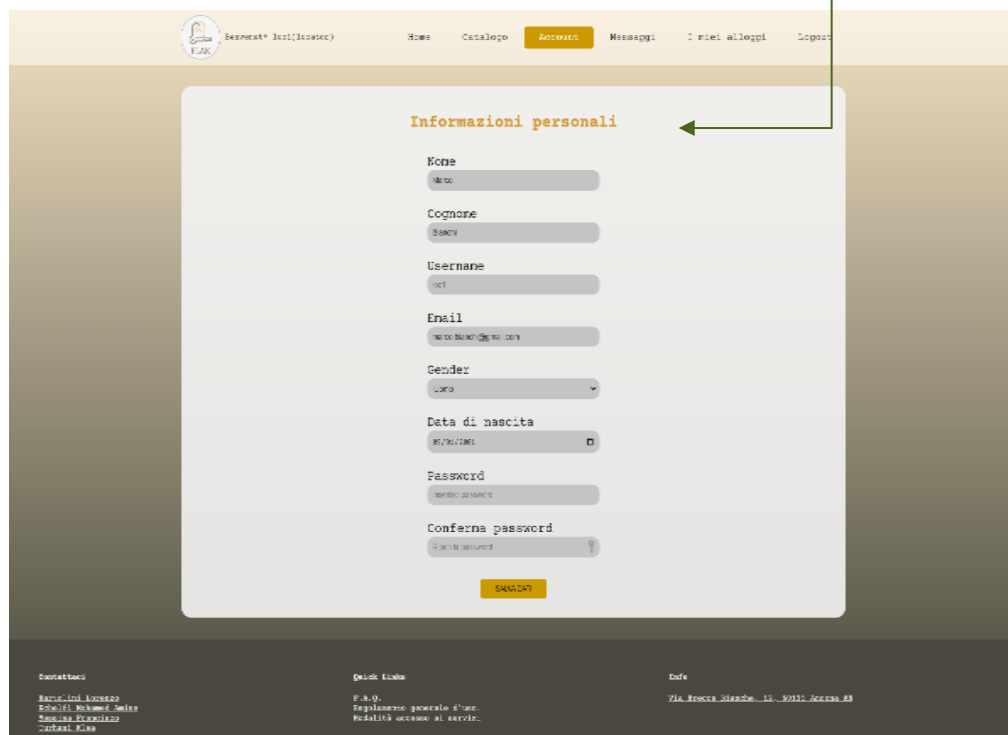
In questa pagina il locatore può consultare e rispondere ai messaggi che gli sono stati inviati dai potenziali locatari.

La sezione **Messaggi**(a sinistra) riporta la lista di tutti i potenziali locatari che hanno scambiato almeno un messaggio con il locatore(vedi pagina 11).

Nella sezione a destra, invece, compare la lista di tutti i messaggi scambiati con l'utente selezionato.

Ciascun messaggio contiene un testo e la data di inoltro del messaggio.

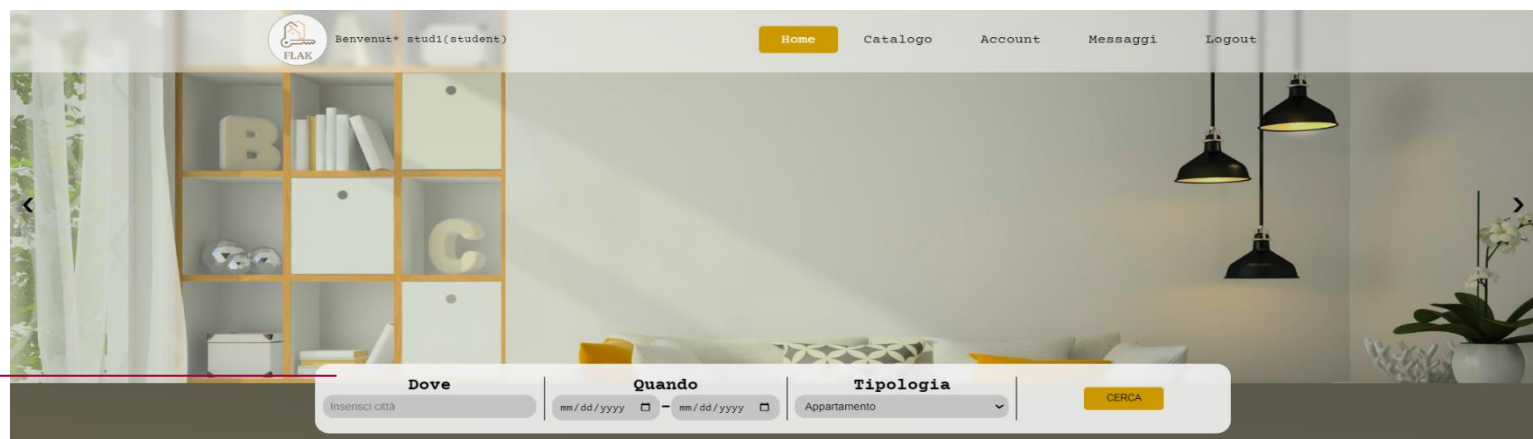
Nella parte alta della pagina compaiono i dettagli relativi all'utente con cui si sta interagendo.



ACCOUNT

In questa sezione ogni utente(sia locatori che locatari) può vedere, cambiare e salvare i suoi informazioni personali.

Livello 3 (LOCATARIO)



Una volta che l'utente ha eseguito l'accesso, la navbar cambierà di conseguenza, aggiungendo le sezioni **Account, Messaggi e Logout**.

Nella homepage si aggiunge, inoltre, una **barra di ricerca**, in modo da permettere al locatario di effettuare una ricerca direttamente dalla homepage. In questa pagina i parametri di ricerca sono limitati, infatti è possibile inserire soltanto il *nome di città, il periodo di locazione e la tipologia (appartamento o posto letto)*.

FILTRI NEL CATALOGO

Nel catalogo si aggiunge una nuova funzionalità, ovvero la **barra di ricerca laterale**, che amplia quella già vista nella Homepage.

Prima di tutto occorre selezionare la **tipologia di alloggio** da filtrare, ovvero **Appartamento, Posto Letto** oppure **Entrambi**.

In base a questa selezione, **la form si amplia**, mostrando solo i parametri specifici oppure entrambi.

Esempio: un parametro specifico è la *dimensione dell'appartamento*, poiché si applica soltanto agli appartamenti e non ai posti letto.

Nota sul filtraggio

Nel caso in cui decidessi di visualizzare **entrambe le tipologie** di alloggio, ma applicassi un **filtro specifico** che si *applica ad una sola tipologia*, allora verrebbero mostrati, oltre agli alloggi filtrati per il filtro specifico, anche tutti gli alloggi a cui il filtro non si applica.

Esempio: se volessi visualizzare **tutti gli alloggi di entrambe le tipologie** che hanno una certa *dimensione dell'appartamento*, visualizzerei tutti gli appartamenti che rispettano la *specificità* e in coda tutti i posti letto, siccome per loro quella caratteristica non è definita.

Oltre ai campi che riguardano le caratteristiche dell'alloggio, abbiamo aggiunto la **possibilità** da parte dello studente di **visualizzare soltanto gli alloggi per cui il Locatario ha fatto richiesta** oppure **quelli che gli sono stati assegnati**.

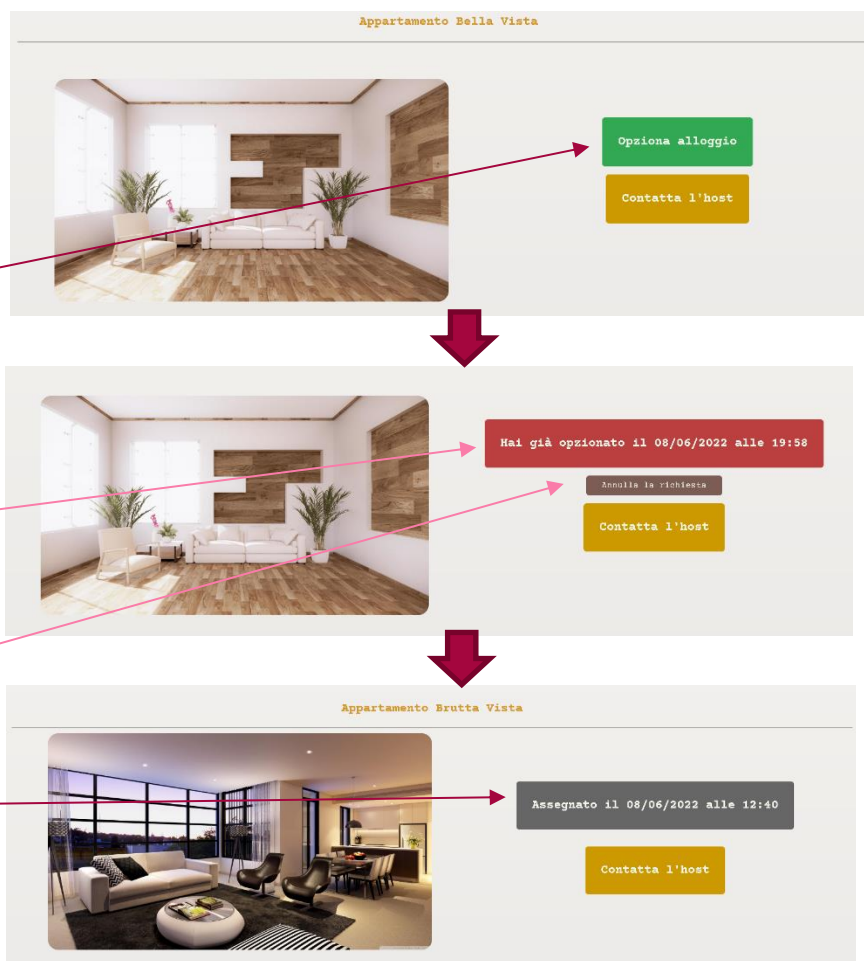
DETTAGLI ALLOGGIO

Se lo studente clicca sul nome dell'appartamento o del posto letto, vede le **informazioni** relative all'alloggio.

Inoltre, quando lo studente visualizza la scheda dell'alloggio, nel caso in cui l'alloggio non è stato ancora assegnato, vengono mostrati due bottoni per **opzionare l'alloggio** o **contattare l'host**.

Una volta che l'utente ha opzionato l'alloggio, la sua richiesta verrà registrata e non gli sarà più possibile effettuare un'opzione. E' possibile **annullare una richiesta** cliccando sul bottone **Annulla la richiesta** (**richiede conferma**).

Quando il locatore, proprietario dell'alloggio, **accetta la richiesta** del potenziale locatario e gli assegna l'alloggio, verrà mostrata la data di assegnazione.

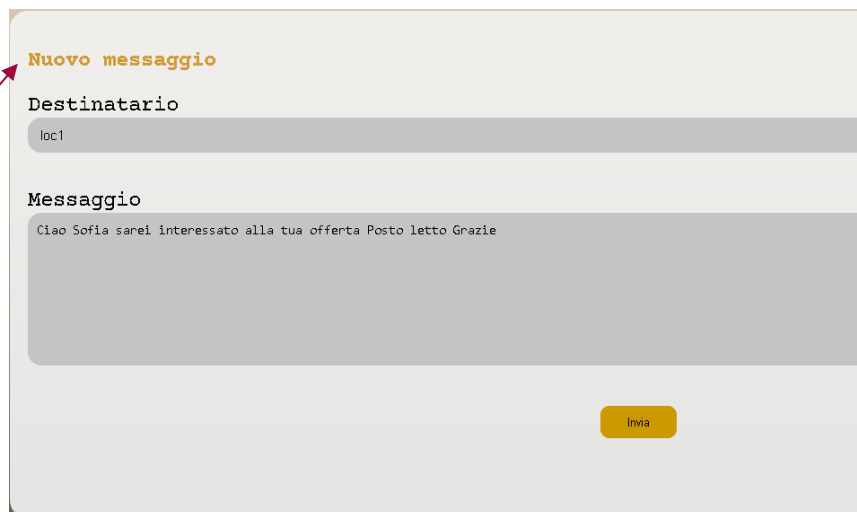


MESSAGGI

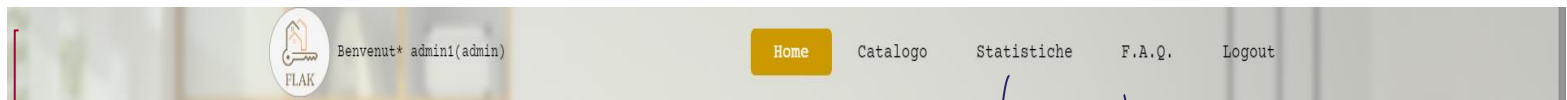
Nella pagina dei messaggi, lo studente ha la possibilità di **visualizzare e rispondere** a tutti i messaggi, esattamente come il locatore. In più, può iniziare una **nuova conversazione** con un locatore con cui non ha mai scambiato un messaggio.

Questo può essere fatto in più modi. Cliccando su **Nuovo Messaggio**, direttamente nella pagina messaggi, cliccando su **Contatta l'host** nella pagina dell'alloggio oppure, sulla stessa pagina, cliccando su **Opziona alloggio**.

Infatti quando l'utente clicca sul tasto **Opziona alloggio**, oltre a registrare la richiesta, verrà aperta anche la scheda **Nuovo Messaggio**, con un messaggio preimpostato e il **destinatario** già selezionato.



Livello 4 (ADMIN)



Una volta che l'utente ha eseguito l'accesso, la navbar cambierà di conseguenza, aggiungendo le sezioni Statistiche e FAQ.

Statistiche generali amministratore

Quando: 05/06/2022 - 07/06/2022 | Tipologia: Entrambi | Filtro

Tipologia	Totale offerte	Offerte opzionate	Alloggi locati
Appartamento	0	1	1
Posto letto	0	1	0
Totale	0	2	1

FILTRI STATISTICHE

E' possibile filtrare le statistiche per **tipologia** e in base al **periodo temporale**.

Nota: per periodo temporale si intende che verranno visualizzate il numero di tutte le **offerte pubblicate** nel *periodo temporale specificato*, il numero di tutte le **richieste effettuate** nel *periodo temporale specificato* da studenti e il numero di tutti gli **alloggi assegnati** dai locatori nel *periodo temporale specificato*.

STATISTICHE

L'amministratore può visualizzare in una tabella il numero delle **offerte totali** presenti sul sito, il numero di **richieste** effettuate da potenziali locatori e il numero di **alloggi assegnati**(locati). Ogni statistica è divisa in base alla tipologia di alloggio a cui si fa riferimento.

Nota: quando un alloggio viene assegnato, tutte le **richieste pendenti** vengono **eliminate** e, pertanto, queste non verranno considerate nel conteggio.

Frequently Asked Questions

Aggiungi FAQ

1. Come posso creare un account?
Basta cliccare sul pulsante Registrati nella navbar del sito.

2. Come inserisco un alloggio?
Dopo aver effettuato l'accesso come locatore, è sufficiente selezionare la sezione I Miei Alloggi e cliccare sul pulsante Aggiungi.

3. Come posso opzionare un alloggio?
Dopo aver fatto l'accesso come locatore, è sufficiente consultare il Catalogo, selezionare un alloggio e premere sul pulsante Opziona Alloggio.

Sicuro? Si No

Modifica Elimina

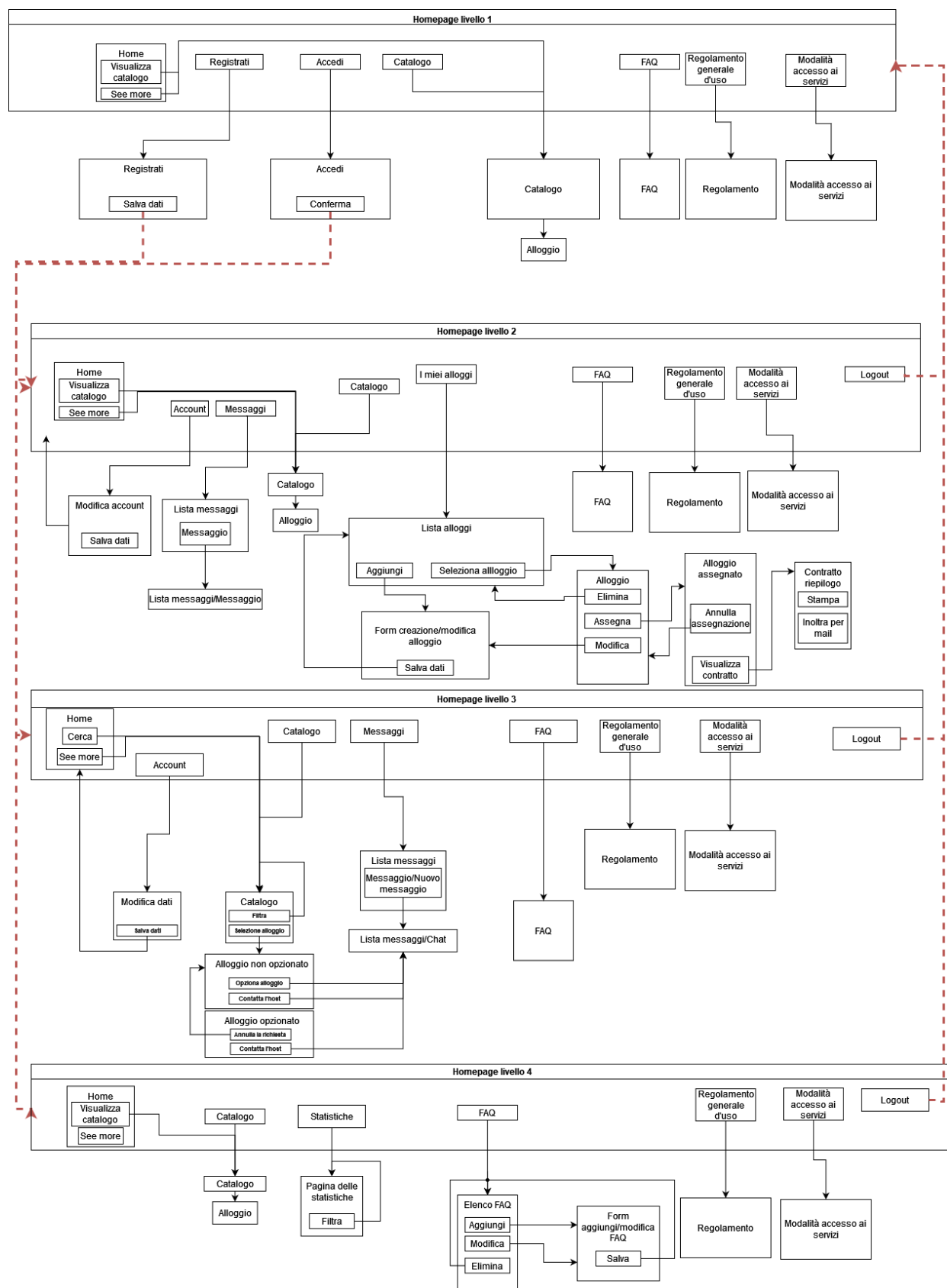
Modifica Elimina

FAQ

L'amministratore può aggiungere, modificare ed eliminare le FAQ presenti sul sito.

Quando si clicca sul pulsante elimina, compaiono due pulsanti per confermare o annullare l'operazione.

Organization chart



Soluzioni adottate

1. Relazione Many to Many

Per la gestione di alcuni dati abbiamo scelto di utilizzare la relazione ManyToMany, in cui ogni elemento di una tabella viene associato ad uno o più elementi di un'altra tabella tramite una tabella intermedia, detta *tabella pivot*, che rappresenta la relazione tra i due elementi.

Questo tipo di relazione è quella che si stabilisce, ad esempio, tra i potenziali locatari e gli alloggi a cui questi ultimi fanno richiesta. Infatti uno studente può opzionare uno o più alloggi e lo stesso alloggio può essere opzionato da uno o più studenti.

Laravel mette a disposizione il `metodo belongsToMany()`, il quale permette di **estrarre dal database** per un certo alloggio, tutti **gli utenti che hanno fatto richiesta**. Il metodo prende in ingresso il nome della tabella intermedia, il nome della chiave primaria della prima entità(gli studenti) e il nome della chiave primaria delle entità a cui essa è legata(alloggi).

accStudId	userId	acclId	relationship
2	1	2	assigned
3	1	3	optioned
4	5	3	optioned
6	1	1	assigned

Figura 1: tabella pivot accomodation_student

userId	role	name	surname	username
1	student	Mario	Rossi	stud1
2	locator	Sofia	Bianchi	loc1
3	locator	Luigina	Bianchi	loc2
4	admin	Giovanni	Verdi	admin1
5	student	Lario	Lario	lariolario

Figura 3: tabella users

acclId	userId	imageId	name	tipology	city
1	2	2	Appartamento Bella Vista	0	Ancona
2	6	3	Appartamento Brutta Vista	0	Firenze
3	6	4	Posto letto da Maria	1	Torino
4	6	2	Appartamento Tavernelle	0	Milano

Figura 2: tabella alloggi(accomodations)

Abbiamo creato nella classe model associata alla tabella degli alloggi, il metodo *optioningStudents()*, che si occupa di estrarre dal database tutti gli utenti che hanno fatto richiesta per un dato alloggio.

```

84 public function optioningStudents() {
85     return $this->belongsToMany(User::class, 'accomodation_student', 'accId', 'userId')
86         ->wherePivot('relationship', 'optioned');
87 }

```

Figura 4: riga 84 dal file `app/model/resources/accomodation.php`

Una soluzione simile è stata adottata per gestire i messaggi scambiati tra gli utenti. In questo caso è la tabella dei messaggi stessa ad essere utilizzata come relazione tra due utenti.

	messageId	senderId	recipientId	text	created_at
	1	1	2	Messaggio da Mario a Marco	2022-06-09 17:33:16
	2	2	4	Messaggio da Marco a Giovanni	2022-06-09 17:33:16
	3	2	4	Messaggio da Marco a Giovanni pt.2	2022-06-09 17:33:16

userId	role	name	surname
1	student	Mario	Rossi
2	locator	Sofia	Bianchi

Figura 5: tabella messaggi

Nelle figure è rappresentata la relazione tra l'utente 1 e l'utente 2. Avendo l'utente 1 inviato un messaggio all'utente 2, i loro id sono stati memorizzati nella tabella messaggi, insieme al testo e la data di invio del messaggio scambiato.

Per estrarre dal database tutti gli interlocutori relativi ad un particolare utente, ovvero i contatti, ci siamo avvalsi nuovamente del metodo *belongsToMany()*.

Prima vengono estratte tutte le persone a cui l'utente selezionato ha inviato almeno un messaggio. Poi vengono estratte, a loro volta, tutte quelle a cui l'utente ha inoltrato almeno un messaggio.

```

69 public function getContacts() {
70     $userWroteToMe = $this->belongsToMany(User::class, 'messages', 'recipientId', 'senderId')
71         ->get();
72     $userIWroteTo = $this->belongsToMany(User::class, 'messages', 'senderId', 'recipientId')
73         ->get();
74
75     return $userIWroteTo->merge($userWroteToMe);
76 }

```

Figura 6: riga 69 dal file `app/model/user.php`

2. Implementazione del CSS

Abbiamo cercato di ridurre al minimo il codice CSS proveniente da sorgenti esterne per disporre di una maggiore flessibilità nei contenuti prodotti. Unica eccezione è la barra di navigazione superiore.

Abbiamo cercato di fare in modo che la struttura della pagina si adatti a dimensioni diverse della finestra di visualizzazione. In particolare, nel caso in cui la larghezza della pagina diventi inferiore a 800 pixel, alcuni elementi vengono riposizionati, in modo da adattarsi ad una modalità di visualizzazione verticale. Quelli che si trovavano uno di fianco all'altro vengono incolonnati, mentre gli elementi presenti nella barra di navigazione vengono nascosti e sostituiti da un menù a tendina.

Questi cambiamenti risultano particolarmente evidenti nel catalogo di livello 3. Infatti, in caso di restringimento dell'area di visualizzazione, l'elenco delle offerte viene riposizionato al di sotto della barra di ricerca, una di seguito all'altra.

3. Gestione delle immagini

Quando l'utente carica un'immagine per un alloggio, viene attivata una funzione jQuery che controlla la dimensione della stessa. Nel caso in cui la **dimensione inserita sia superiore a quella consentita**(2Mb), verrà visualizzato sulla pagina un **messaggio di errore**. Inoltre *tutte* le **richieste AJAX vengono bloccate**, per evitare di inviare al server anche l'immagine con dimensioni al di sopra di quelle consentite. Dalle nostre prove, infatti, risulta che se l'immagine eccede i 5Mb, il server produce un errore relativo al superamento della dimensione massima del Payload.

```

119 |         if ({inputVal.size > 2097152})
120 |         {
121 |             $("#image").parent().find('.errors').html(' ');
122 |             $("#image").after(getFileErrorHtml());
123 |
124 |             sendAjax = false;

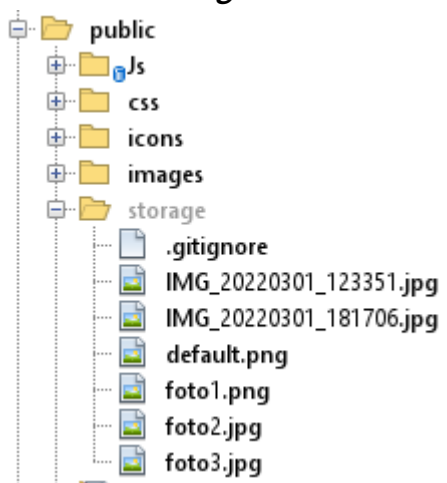
```

Figura 7: riga 119 dal file public/Js/FormValidation.js

Nota: le **richieste AJAX** servono per la **validazione in tempo reale della form** di inserimento di un alloggio. Infatti ogni volta che un elemento della form perde il focus, vengono inviati al server tutti i dati inseriti dall'utente. Il server può rispondere con la lista di tutti i messaggi di errore. Quando la richiesta si è attivata per la perdita del focus di un elemento, verrà visualizzato l'errore relativo solo a

quell'elemento, mentre se si è cliccato sul bottone di submit, allora verranno mostrati tutti gli errori.

Se l'immagine rispetta i criteri richiesti, verrà **memorizzata sfruttando il metodo `store()` messo a disposizione da Laravel**. Questo metodo, oltre a memorizzare il file nella cartella predefinita *storage*, **rinomina il file con un codice univoco**, per evitare errori nel caso in cui l'utente inserisca inavvertitamente un file con lo stesso nome di uno già presente sul server. Per recuperare le immagini salvate dalla cartella *storage*, è stato necessario attivare il comando *php artisan storage:link*, il quale crea un collegamento nella cartella *public*, direttamente alla cartella *storage*.



Se le immagini non fossero presenti nella cartella *public*, il client non potrebbe accedervi con una richiesta http per visualizzarle. Nella tabella *images* del database verrà creato, inoltre, un riferimento al file caricato. Viene, infatti, conservato il codice univoco generato al momento della memorizzazione da parte del metodo *store()*.

```

95
96
97
98
99
100
101
102
103
104
105

public function addAccommodation(AccommodationRequest $request) {
    $user = Auth::user();

    if ($request->hasFile('image')) {
        $file = $request->image;
        $destinationPath = '';
        $file->store($destinationPath, 'public');
        $imageName=$file->hashName();
    } else {
        $imageName = null;
    }
}

```

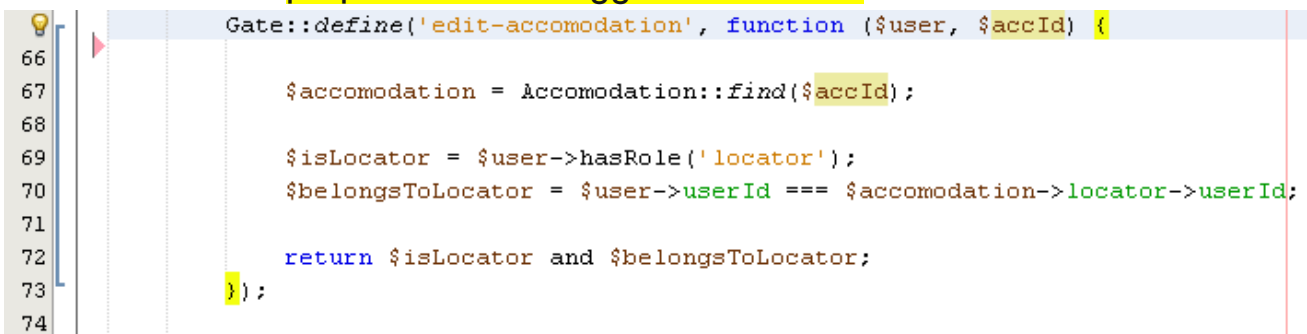
Figura 8: riga 94 dal file `app/Http/Controllers/LocatorController.php`

Nota: per semplicità non è possibile cancellare un'immagine dopo che questa è stata aggiunta, ma solamente cambiarla con una nuova. Tutte le immagini rimangono sul server anche dopo che l'alloggio è stato eliminato.

4. Gestione delle autorizzazioni

Per gestire l'autorizzazione dei contenuti presenti sul sito, abbiamo utilizzato la funzionalità messa a disposizione dalla *facade Gate* di Laravel.

Ad esempio il Gate *edit-accomodation* autorizza i soli **utenti di livello 2**(locatori) che siano anche **proprietari dell'alloggio considerato**.



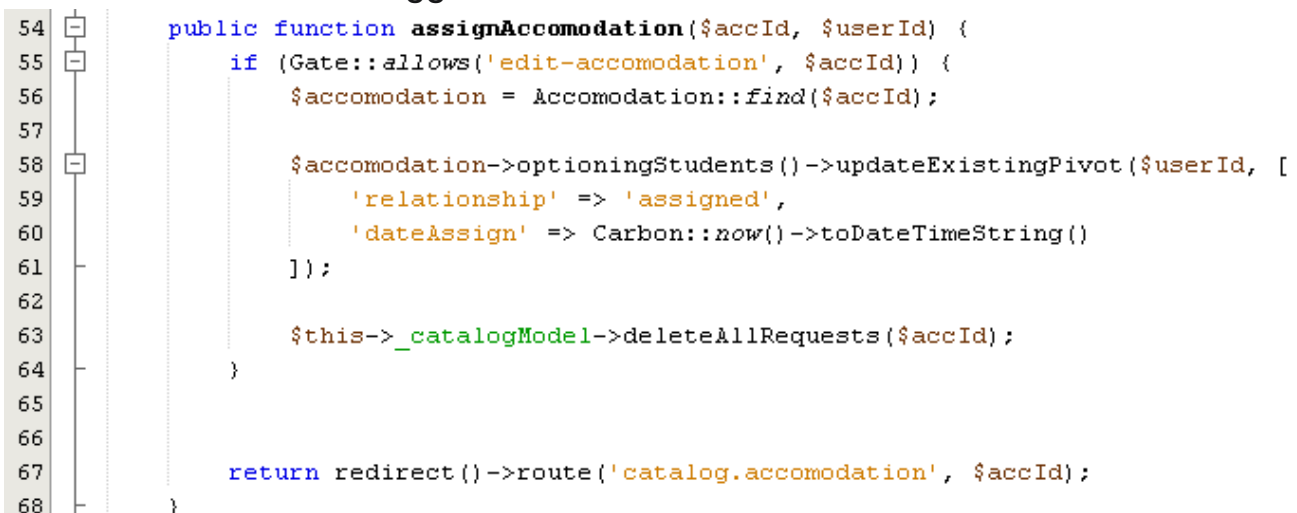
```

66 Gate::define('edit-accomodation', function ($user, $accId) {
67     $accomodation = Accomodation::find($accId);
68
69     $isLocator = $user->hasRole('locator');
70     $belongsToLocator = $user->userId === $accomodation->locator->userId;
71
72     return $isLocator and $belongsToLocator;
73 });
74

```

Figura 9: riga 65 dal file `app/Providers/AuthServiceProvider.php`

Per semplicità abbiamo aggiunto questo controllo solo sul metodo che permette di assegnare un alloggio. In maniera del tutto analoga sarebbe possibile applicare lo stesso controllo anche agli altri metodi del controller che modificano le informazioni di un alloggio.



```

54 public function assignAccommodation($accId, $userId) {
55     if (Gate::allows('edit-accomodation', $accId)) {
56         $accomodation = Accomodation::find($accId);
57
58         $accomodation->optioningStudents()->updateExistingPivot($userId, [
59             'relationship' => 'assigned',
60             'dateAssign' => Carbon::now()->toDateTimeString()
61         ]);
62
63         $this->_catalogModel->deleteAllRequests($accId);
64     }
65
66
67     return redirect()->route('catalog.accomodation', $accId);
68 }

```

Figura 10: riga 54 dal file `app/Http/Controllers/LocatorController.php`

Lo stesso gate è stato utilizzato per gestire le autorizzazioni nelle viste. Infatti i pulsanti per modificare e cancellare un alloggio compaiono solo a quegli utenti autorizzati dal gate (vedi riga 93 del file `resources/views/accomodation.blade.php`).

Per autorizzare soltanto gli **utenti che sono stati assegnati ad un alloggio**, abbiamo introdotto il Gate *is-assigned-student*, il quale controlla che l'id dell'utente loggato corrisponda all'id dell'utente a cui è stato assegnato l'alloggio.

```

83 | Gate::define('is-assigned-student', function ($user, $accomodation) {
84 |     $assignedStudent = $accomodation->assignedStudents->first();
85 |
86 |     if ($assignedStudent) {
87 |         return $assignedStudent->userId === $user->userId;
88 |     } else {
89 |         return false;
90 |     }
91 | });

```

Figura 11: riga 83 dal file `app/Providers/AuthServiceProvider.php`

Gli utenti autorizzati da questo gate sono i soli che conoscono a chi è stato assegnato un alloggio, insieme ai proprietari dell'alloggio stesso. Abbiamo preferito nascondere questa informazione a tutti gli altri utenti, che, infatti, sapranno che quell'alloggio è stato assegnato, ma non a chi.

```

257 | @can('is-assigned-student', $accomodation)
258 | @if($accomodation->hasBeenAssigned())
259 | <div class='margin-t-small border-t'>
260 |     <h2 class='pad-tb-small border-t'>Contratto</h2>
261 |     <p>Assegnato il {{ $accomodation->dateAssign() }} alle {{ $accomodation->timeAssign() }} a te</p>
262 | </div>
263 | @endif
264 | @endcan

```

Figura 12: riga 257 dal file `resources/views/accomodation.blade.php`

5. Funzionalità opzionali

Segnaliamo l'implementazione di una delle due funzionalità opzionali, ovvero quella relativa alla stipula del contratto.

Come già discusso, abbiamo memorizzato nel database, nella tabella *accomodation_student*, quale utente è stato assegnato a quale alloggio e in quale data. Pertanto per generare il contratto, oltre a recuperare dall'alloggio il proprietario, è stato sufficiente estrarre dalla tabella di cui sopra il locatario assegnato, per poi riempire il documento con le loro informazioni.

```

37 public function showContract($accId) {
38     $accomodation = Accomodation::find($accId);
39     $student = $accomodation->assignedStudents()->first();
40     $locator = $accomodation->locator;
41
42     return view('contract')
43         ->with('accomodation', $accomodation)
44         ->with('student', $student)
45         ->with('locator', $locator);
46 }

```

Figura 13: riga 29 dal file `app/Http/Controllers/LocatorController.php`

Per l'inoltro via email, è stato necessario utilizzare una funzione JavaScript per estrarre dalla pagina il testo contenuto nei paragrafi per poi costruire la stringa che compone il corpo dell'email.

```

18 var mailToLink = "?subject=FLAK - Assegnazione alloggio&" + "body=" + encodeURIComponent(formattedBody);
19 var mailToAnchor = document.getElementById('mailto');
20 mailToAnchor.href += mailToLink;

```

Figura 14: file `public/Js/createEmailText.js`