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:

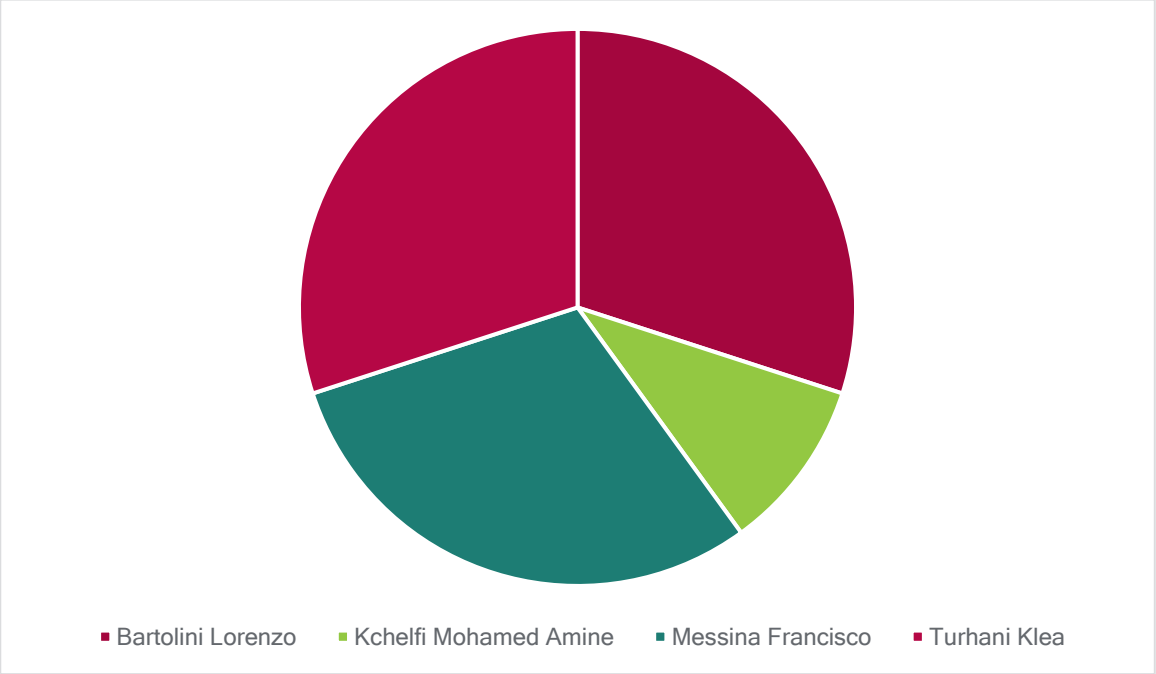
Cucchiarelli Alessandro

FLAK

Contributo al Progetto

Espresso in percentuale

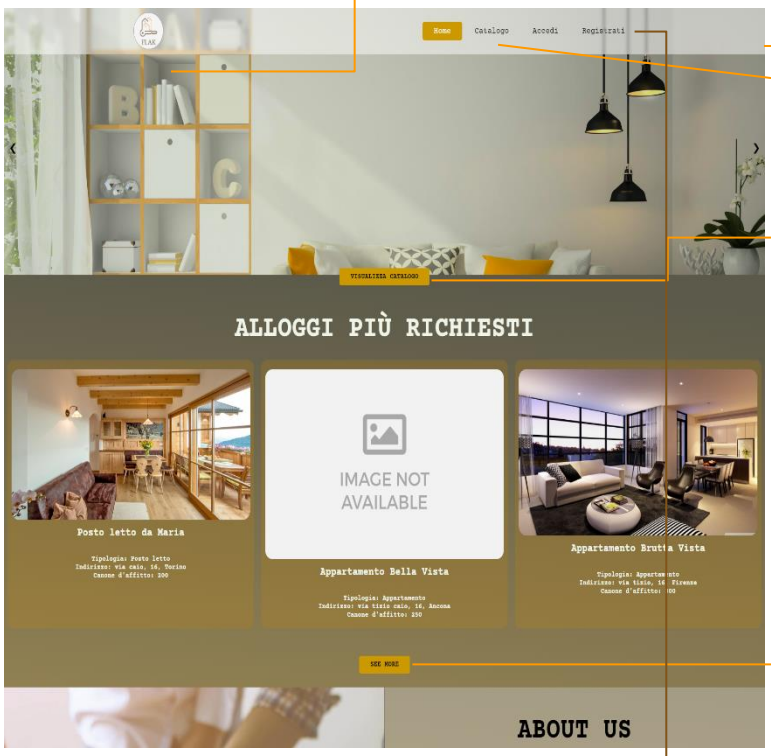
Bartolini Lorenzo	30%
Kchelfi Mohamed Amine	10%
Messina Francisco	30%
Turhani Klea	30%

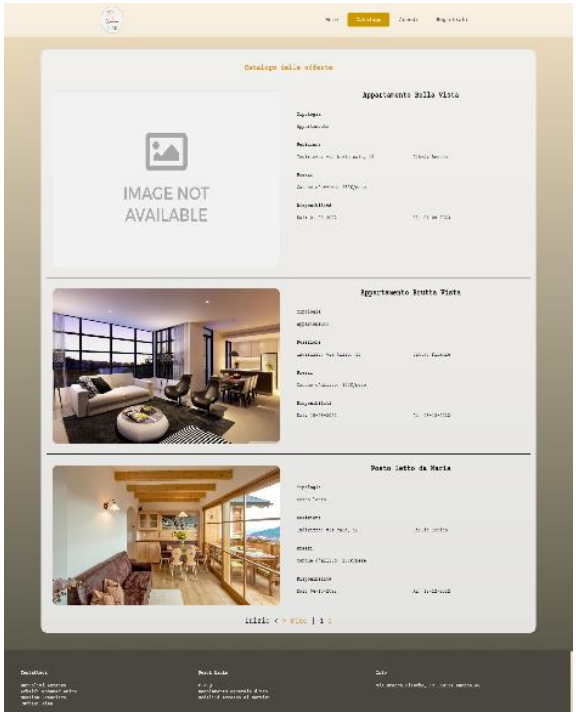


Descrizione del sito
Livello 1

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.

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).

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.

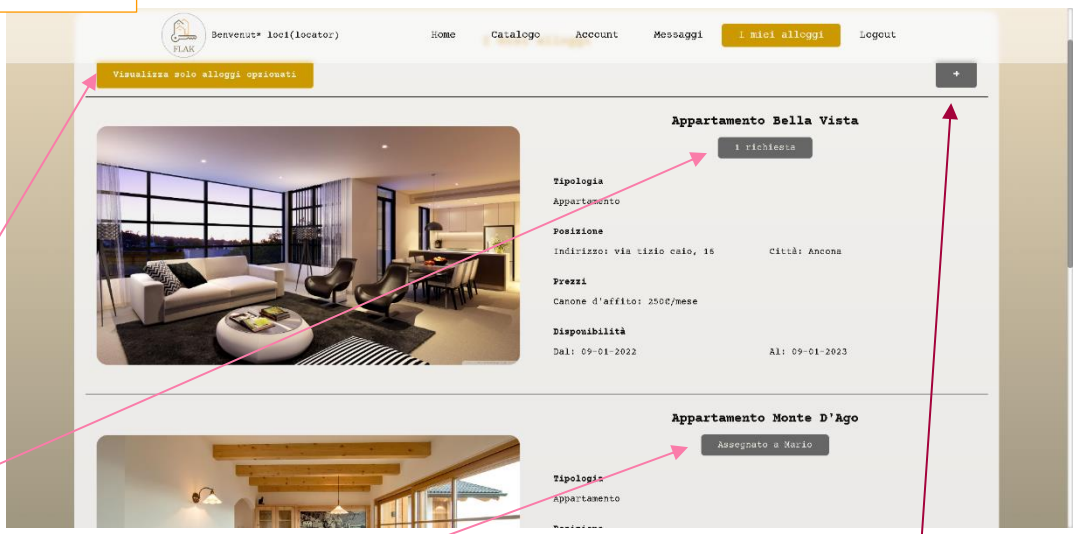
I MIEI ALLOGGI

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

Quando clicca sul tasto **visualizza solo alloggi opzionati,** viene visualizzata solo la lista di alloggi a cui qualcuno ha fatto richiesta.

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.**



\+

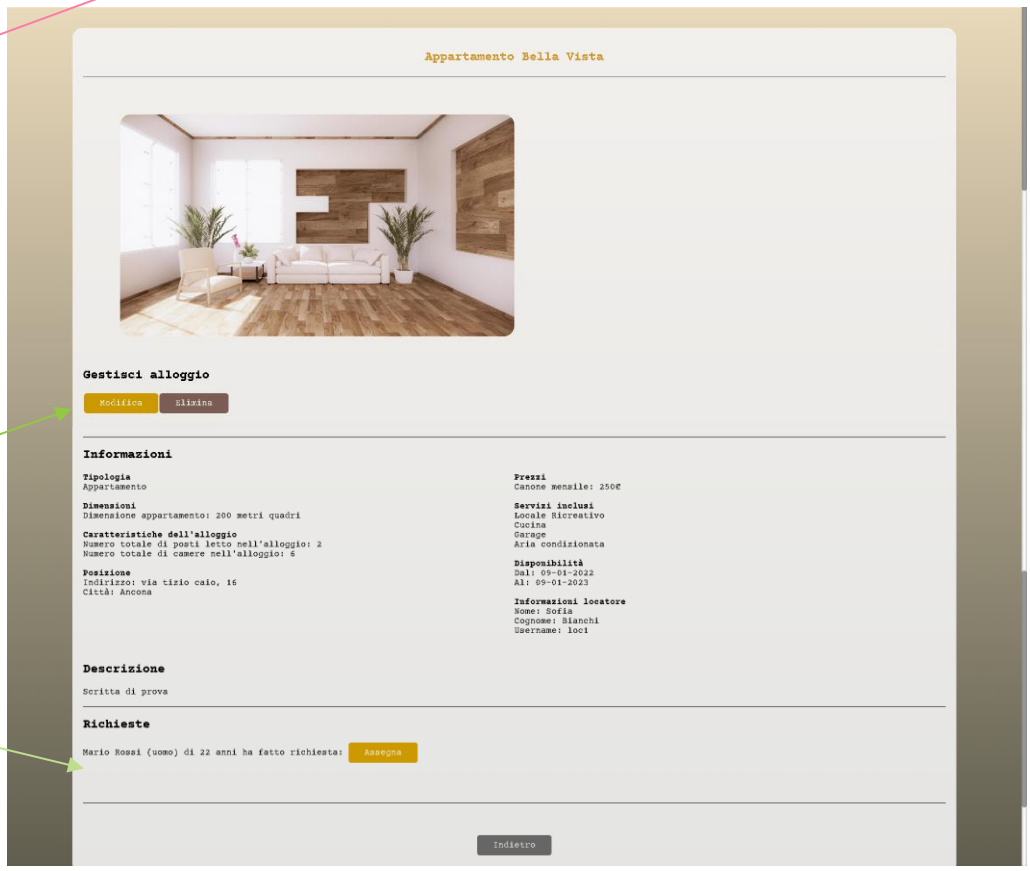
Cliccando su questo pulsante il locatore può inserire un alloggio.

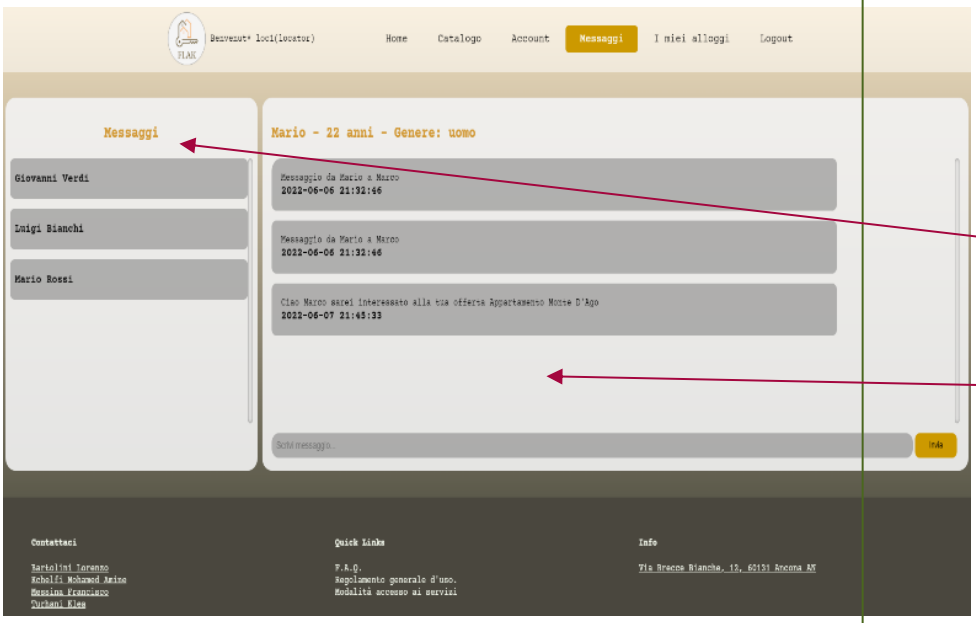
DETTAGLI ALLOGGIO

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, vengono mostrati due bottoni per ***modificare*** o ***cancellare*** l'alloggio.

Nella parte bassa della pagina è presente la lista con tutte le **richieste**. 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

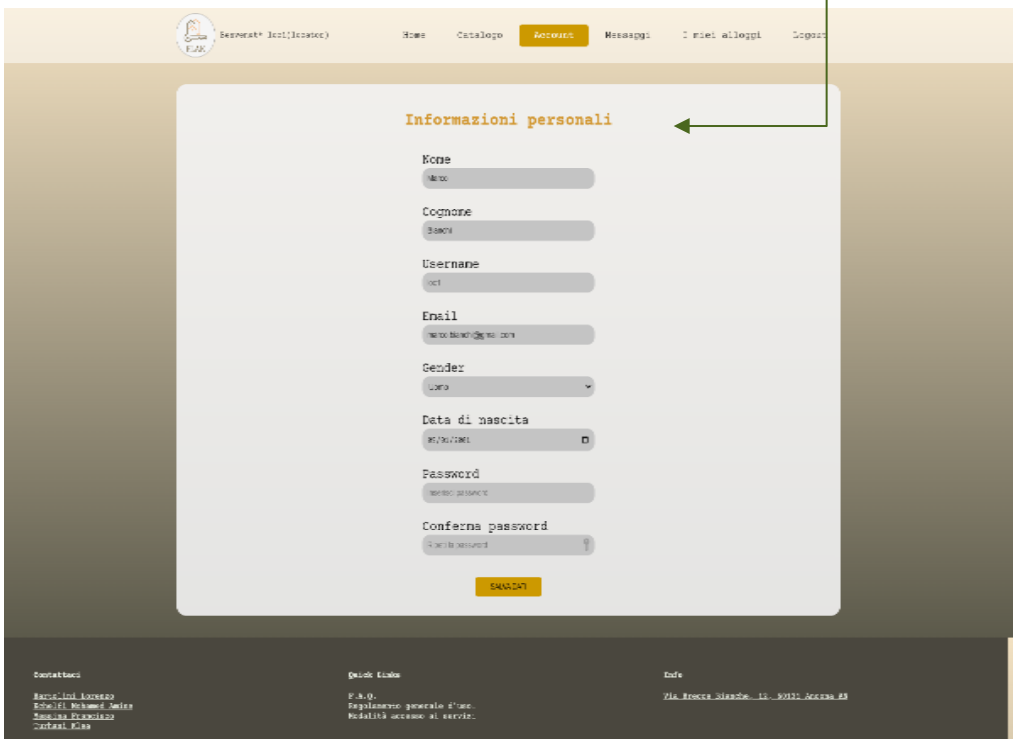
In questa pagina il locatore può consultare e rispondere ai messaggi che gli sono stati inviati dai potenziali locatari.

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

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 (STUDENTE)

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 citta, 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 **tutte le tipologie** che hanno una certa dimensione dell'appartamento, visualizzerei tutti gli appartamenti che rispettano la specifica 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 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.

Quando lo studente visualizza la scheda dell'alloggio, se 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).

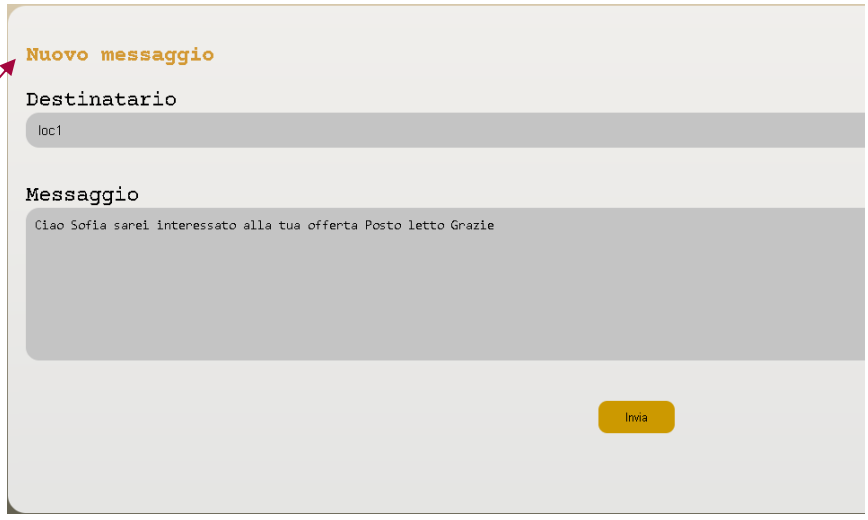
Quando il locatore, proprietario dell'alloggio, **accetta la richiesta** del potenziale locatario e gli assegna l'alloggio, verrà mostrata la data di assegnazione.

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.

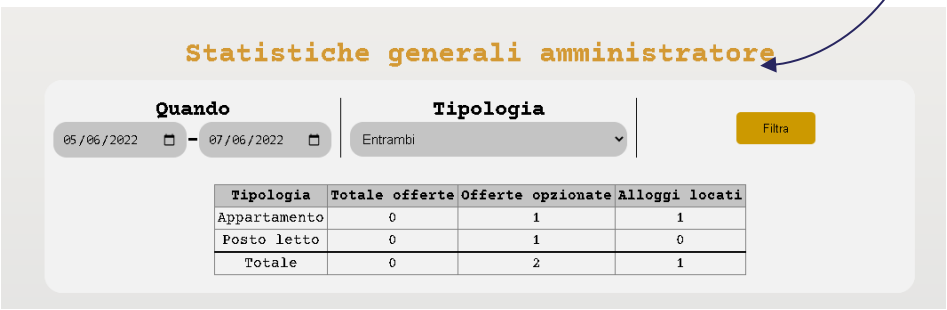
Questo può essere fatto in più modi. Cliccando su **Nuovo Messaggio,** direttamente nella pagina messaggi, cliccando su **Contatta l'host** nella pagina dell'alloggio oppure, sulla stessa pagina, cliccando su **Opziona alloggio.**

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

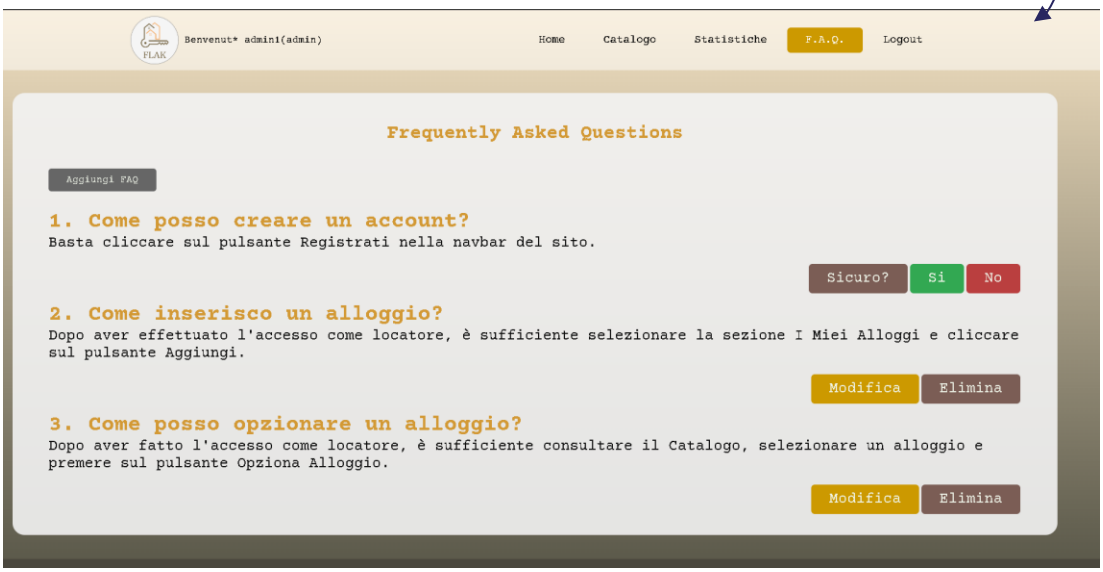
L'amministratore può visualizzare in una tabella il numero delle **offerte totali** presenti sul sito, il numero di **richieste** effettuate da potenziali locatari e il numero di **alloggi assegnati**(locati), ognuna divisa in base alla tipologia di alloggio a cui si fa riferimento.

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

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* e il numero di tutti gli **alloggi assegnati** nel *periodo temporale specificato*.

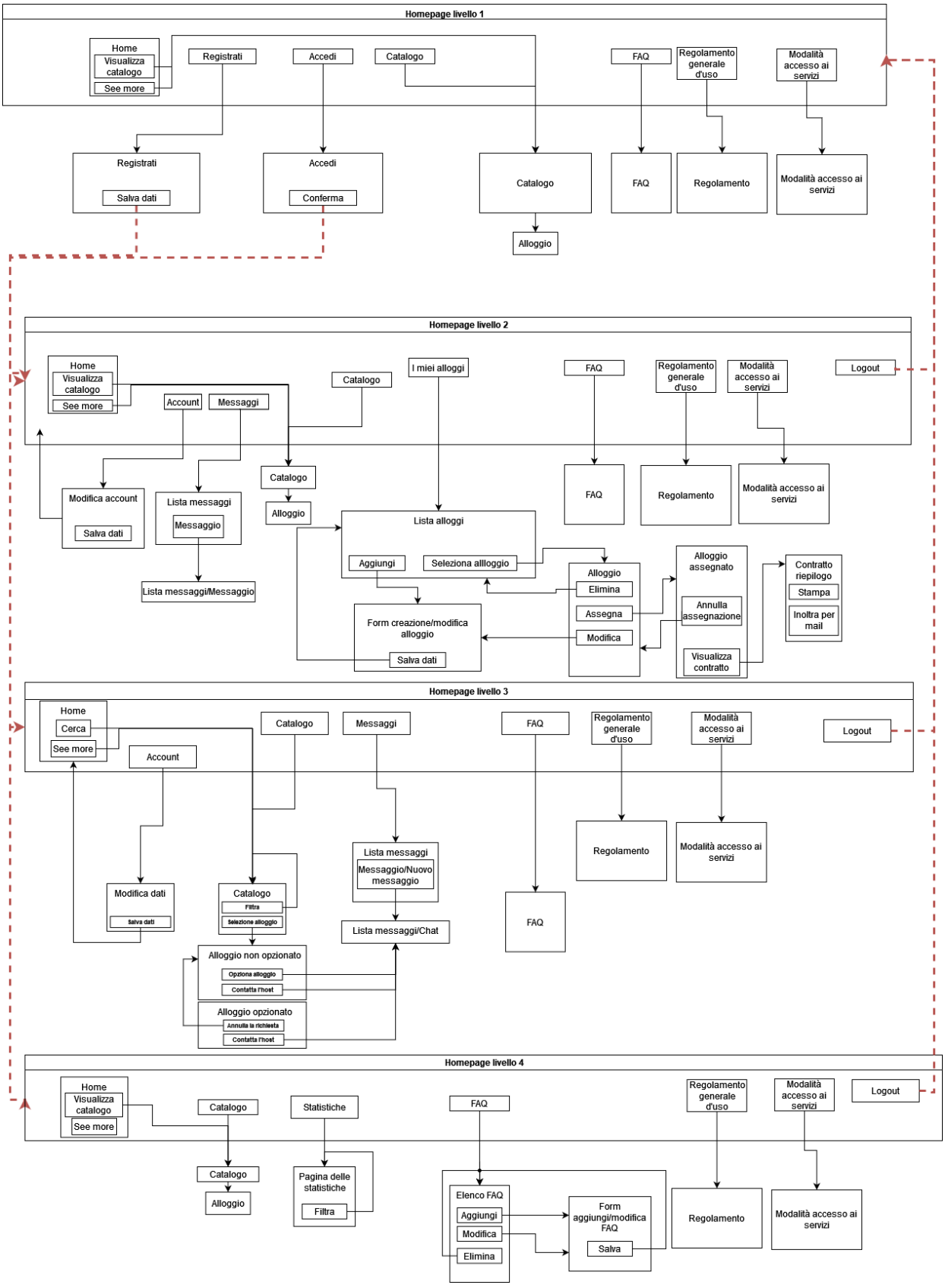


FAQ

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 far richiesta ad uno o più alloggi e lo stesso alloggio può venir richiesto da uno o più studenti.

Laravel mette a disposizione il metodo *belongsToMany(),* il quale permette di estrarre dal DB 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	accId	relationship
2	1	2	assigned
3	1	3	pptioned
4	5	3	pptioned
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

accId	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 utenti con cui un particolare utente ha scambiato almeno un messaggio, ovvero i contatti da mostrare nel sistema di messaggistica, ci siamo avvalsi nuovamente del metodo *belongsToMany().* Prima estraiamo tutti gli utenti a cui l'utente selezionato ha inviato un messaggio e poi tutti quelli a cui egli ha inviato un messaggio a sua volta e uniamo i risultati.

```
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.

Una circostanza in cui questi cambiamenti risultano particolarmente evidenti è nel catalogo di livello 3. Infatti, in caso di restringimento dell'area di visualizzazione, l'elenco delle offerte vengono riposizionate al di sotto della barra laterale di ricerca, una sotto l'altra.

3. Gestione delle immagini

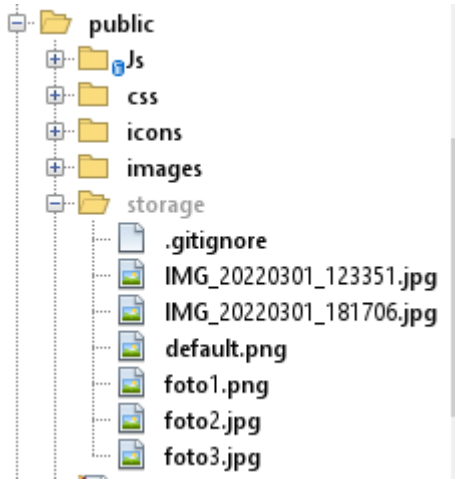
Quando l'utente carica un'immagine per un alloggio, viene attivata una funzione jQuery che controlla la dimensione della stessa. Nel caso in cui la dimensione inserita sia superiore a quella consentita(2Mb), verrà visualizzato un messaggio di errore. Inoltre *tutte* le richieste AJAX vengono bloccate, 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. Le richieste AJAX servono per la validazione in tempo reale della validazione della form di inserimento di un alloggio.

```
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

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.*



Questo è essenziale perché la cartella *public* è l'unica che accessibile al client quando effettua una richiesta di un documento, durante la navigazione web. Nella tabella *images* del database verrà creato, inoltre, un riferimento al file caricato, memorizzando il codice univoco generato al momento della memorizzazione dal metodo *store().*

```
    public function addAccomodation(AccomodationRequest $request) {
95      $user = Auth::user();
96
97      if ($request->hasFile('image')) {
98          $file = $request->image;
99          $destinationPath = '';
100         $file->store($destinationPath, 'public');
101         $imageName=$file->hashName();
102
103     } else {
104         $imageName = null;
105     }
```

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.