

TravelBook

Sistema di prenotazioni alberghiere

Simone Moscatelli

Marco Paolini

Linguaggi Dinamici

Traccia 1:

SISTEMA DI PRENOTAZIONE ALBERGHIERE

Sistema di prenotazioni alberghiere Applicazione Web per la gestione di un sistema di prenotazioni alberghiere in stile Booking.com

L'applicazione è pensata per essere usata da utenti anonimi ed utenti registrati:

- I proprietari delle strutture possono, previa registrazione al sito, inserire le caratteristiche dell'albergo, comprensive di tipologia di struttura, servizi offerti, numero e tipo di camere disponibili, prezzi degli alloggi, locazione geografica.
- Gli utenti anonimi possono cercare alloggi disponibili in base al periodo di tempo e alle caratteristiche desiderate; nel momento in cui desiderano effettuare la prenotazione attraverso il sito devono fare il login con le credenziali inserite al momento della registrazione.

Il sistema deve consentire la ricerca delle strutture alberghiere in base alla disponibilità in un determinato periodo ed alla presenza o meno di determinate caratteristiche (es, presenza di piscina, spa, ristorante): le strutture trovate in seguito alla ricerca devono essere mostrate in ordine decrescente di prezzo complessivo per la durata dell'alloggio.

Il sistema deve gestire la disponibilità nelle strutture decrementandola all'atto di ogni prenotazione per il periodo indicato. Deve inoltre dare agli utenti registrati la possibilità di gestire (modificare/eliminare) le prenotazioni fatte precedentemente, modificando di conseguenza le prenotazioni. Infine, un utente registrato può fare mettersi in lista d'attesa per una struttura già occupata in un determinato periodo e ricevere una notifica nel momento in cui la struttura divenisse disponibile in base a cancellazioni di prenotazioni. Facoltativo Inserire un semplice meccanismo di voto da parte degli utenti registrati basato su valori numerici (es. voto da 1 a 5).

Il voto assegnato alla struttura verrà visualizzato nella pagina relativa, inoltre dovrà essere data la possibilità di ordinare le strutture disponibili determinare dalla ricerca in ordine di giudizio degli utenti (in alternativa al prezzo).

1 Introduzione	4
1.1 Introduzione	4
1.2 Home	5
1.3 Filtraggio ricerca	7
2 Ricerca	8
2.1 Introduzione ricerca.....	8
2.2 Filtraggio	8
2.3 Template.....	8
3 User	10
3.1 Introduzione user.....	10
3.2 Voti	10
3.3 Prenotazioni effettuate.....	11
3.4 Favoriti	11
3.5 Lista d’attesa	11
3.6 Aggiungi location/stanza.....	12
4 Login	13
4.1 Login	13
4.2 Registrazione.....	13
5 Appendici	14
5.1 Interfaccia di admin.....	14
5.2 Javascript.....	14
5.3 Test.....	15
5.4 Diagramma delle classi	15

INTRODUZIONE

1.1 INTRODUZIONE

TravelBook è un sito web realizzato con framework Django, permette di prenotare la propria vacanza in modo semplice e intuitivo.

Il sito si basa sullo sviluppo di punti fondamentali, quali:

- Autenticazione
 - o Registrazione
- Ricerca
 - o Prenotazione
 - o Aggiunta favoriiti
 - o Lista d'attesa
- Gestione profilo utente
 - o Modifica prenotazione
 - o Modifica favoriiti
 - o Modifica lista attesa

1.2 MODEL

Il file *models.py* contiene le classi:

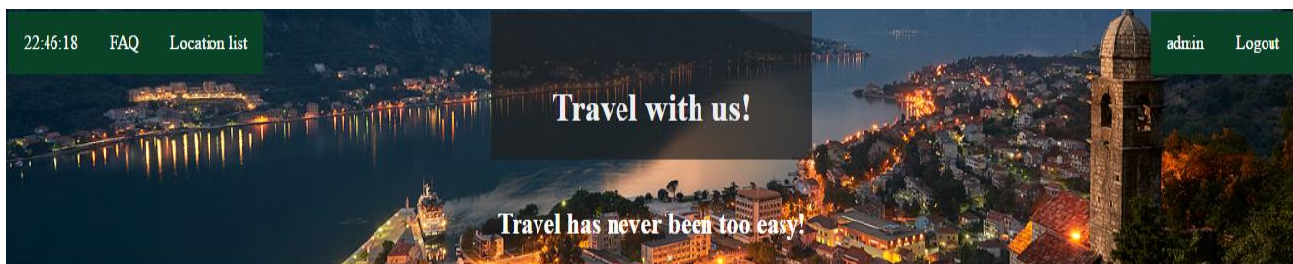
Location	<pre>class Location(models.Model): name = models.CharField(max_length=200) address = models.CharField(max_length=200) municipality = models.CharField(max_length=200) city = models.CharField(max_length=200) country = models.CharField(max_length=200) cap = models.IntegerField(default=0) telephone = models.CharField(max_length=200) stars = models.IntegerField(default=0) pool = models.BooleanField(default=False) WiFi = models.BooleanField(default=False) disabled_services = models.BooleanField(default=False) restaurant = models.BooleanField(default=False) car_parking = models.BooleanField(default=False) gym = models.BooleanField(default=False) bar = models.BooleanField(default=False) sports_activities = models.BooleanField(default=False) spa_services = models.BooleanField(default=False) description = models.CharField(max_length=1000) site_web = models.CharField(max_length=600) pub_date = models.DateTimeField('date published')</pre>
----------	--

Room	<pre>class Room(models.Model): id_location = models.ForeignKey(Location, related_name="room_set") number_room = models.IntegerField(default=0) price = models.FloatField(default=0.0) price_holiday = models.FloatField(default=0.0) n_people = models.IntegerField(default=0) conditioner = models.BooleanField(default=False) kitchenette = models.BooleanField(default=False) smoker_room = models.BooleanField(default=False) animals_accept = models.BooleanField(default=False)</pre>
RoomFavorite	<pre>class RoomFavorite (models.Model): room_favorite = models.ForeignKey(Room, related_name="room_favorite_set") user_name = models.ForeignKey(User, related_name="username_set")</pre>
RoomWaitingList	<pre>class RoomWaitingList (models.Model): room_waiting_list = models.ForeignKey(Room, related_name="room_waitinglist_set") user_waiting_list = models.ForeignKey(User, related_name="username_waitinglist_set") user_reservation = models.ForeignKey(User, related_name="username_reservation_set") check_in_waitinglist = models.DateTimeField('check_in date') check_out_waitinglist = models.DateTimeField('check_out date')</pre>
Prenotation	<pre>class Prenotation(models.Model): id_room = models.ForeignKey(Room, related_name="prenotation_set") id_user = models.ForeignKey(User, related_name="user_set") check_in = models.DateTimeField('check_in date') check_out = models.DateTimeField('check_out date')</pre>

1.2 HOME

Tre sezioni dividono la schermata di home:

Menù:

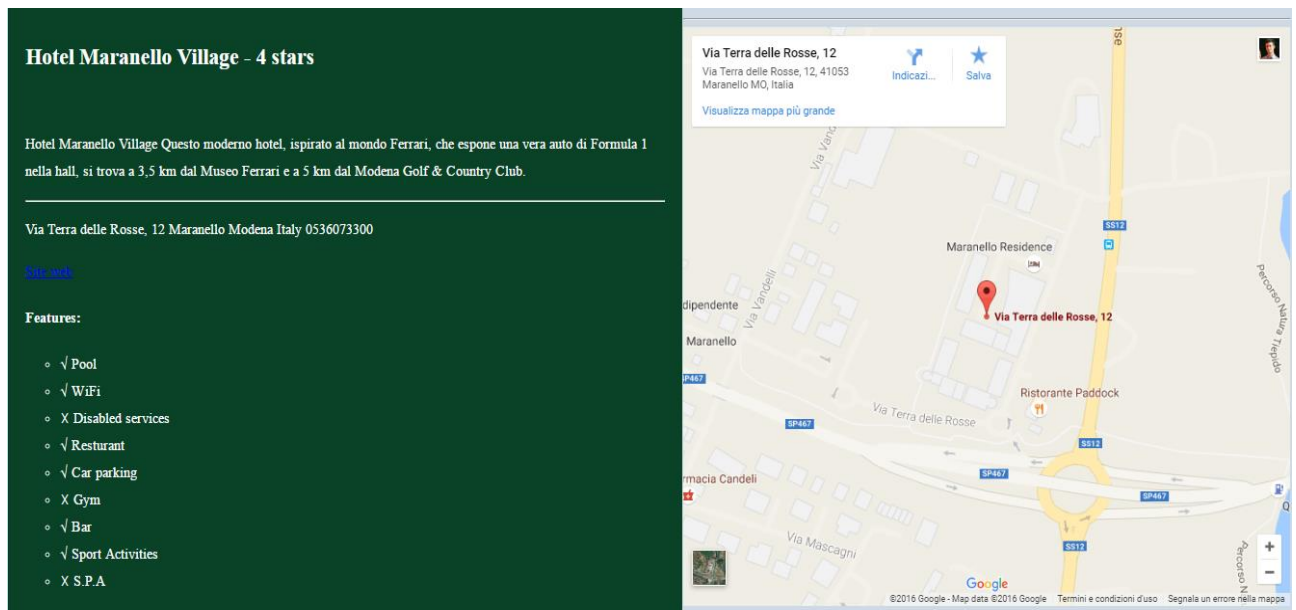


- FAQ

Frequently Asked Questions, come dice il nome, sono le domande e rispettive risposte più comuni, che ci sembrano le più utili per aiutare un utente al suo primo accesso al sito.

- LOCATION LIST

Contiene tutte le informazioni relative alle locations inserite nel database.



Come si vede dall' immagine, per ogni locations, oltre che le relative caratteristiche è possibile consultare il sito web della location e visualizzare la sua posizione su Google Maps.

La pagina corrente è stata resa possibile tramite il comando “render” nel file `views.py` di `TravelBook`.

Il quale (render) ritorna un `HttpResponse` e passa al template `index.html` un dictionary contenente tutte le istanze di `Location`.

```
from .models import Location

return render(request, 'TravelBook/index.html', {'location':
Location.objects.all()})
```

Il template semplicemente sarà costituito da un ciclo `for`, che stampa tutti i campi del model.

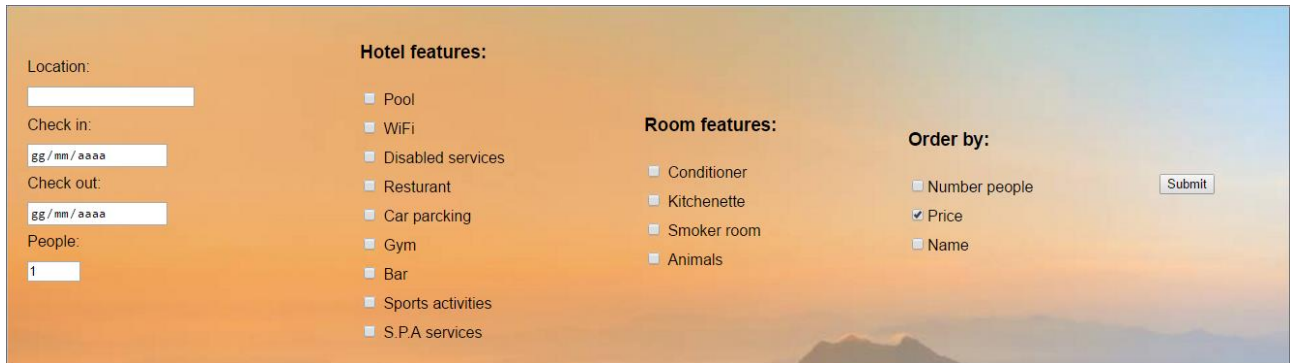
- ANONYMOUS-LOGIN-LOGOUT

- *Anonymous* compare agli utenti che navigano senza eseguire il login in modo da evidenziare che non avranno alcuni privilegi.
- *Login* serve agli utenti registrati a loggarsi col proprio username e password all'applicazione in modo da poter poi prenotare la vacanza desiderata o modificare una qualche prenotazione precedente. Gli utenti non registrati tramite Login possono accedere al form di registrazione ed una volta inseriti i dati richiesti possono iniziare ad usare l'applicazione in modo completo.

Un utente, una volta loggato, cliccando sul proprio nome può accedere alle proprie prenotazioni, votare e consultare i propri preferiti e liste d'attesa.

- *Logout* permette di uscire dal proprio profilo.

1.3 FILTRAGGIO RICERCA



The screenshot shows a search filter interface with a background image of a sunset over mountains. The interface is divided into several sections:

- Location:** A text input field.
- Check in:** A date input field with the placeholder "gg/mm/aaaa".
- Check out:** A date input field with the placeholder "gg/mm/aaaa".
- People:** A numeric input field with the value "1".
- Hotel features:** A list of checkboxes including Pool, WiFi, Disabled services, Restaurant, Car parking, Gym, Bar, Sports activities, and S.P.A services.
- Room features:** A list of checkboxes including Conditioner, Kitchenette, Smoker room, and Animals.
- Order by:** A list of checkboxes including Number people, Price (which is checked), and Name.
- Submit:** A button located to the right of the Order by section.

Permette di trovare le stanze disponibili per le proprie esigenze. E' possibile scegliere le caratteristiche degli hotel e delle stanze tramite semplici checkbox.

L'utente può ordinare i risultati in base al prezzo (default), numero di persone o nome.

RICERCA

2.1 INTRODUZIONE RICERCA

Tramite la sezione di ricerca un utente può cercare la propria Location (ordinati per voti in ordine crescente) tra quelle presenti nel database.

Dal punto di vista implementativo, le scelte dell'utente vengono raccolte nel template da un form, che permette di passare i parametri in input al search.

Prima di entrare nel search, vengono gestiti dalla *views*, dove vengono effettuati i vari filtraggi per adattare l'output della search ai parametri inseriti dall'utente.

2.2 FILTRAGGIO

E' necessario filtrare le stanze.

- *condition_name*

Contiene le sole stanze del database che hanno la location specificata dall'utente,

- *condition_num_people*

Contiene le sole stanze del database che possono ospitare quel numero di persone specificato.

- *condition_loc_features*

Contiene le sole stanze delle location con le caratteristiche specificate.

- *condition_room_feature*

Contiene le sole stanze con le caratteristiche indicate.

Infine le varie condition vengono intersecate per creare e successivamente passare un dictionary al template search.

2.3 TEMPLATE

Room 2 of Panoramic Hotel Plaza Room number: 2 Room people: 2 Price to night: 91.3 Features: <ul style="list-style-type: none">o ✓ Conditionero ✓ Kitchenetteo X Smoker Roomo X Animals Accept <div>To book</div> <div>Favorite</div>	Room 3 of Panoramic Hotel Plaza Room number: 3 Room people: 4 Price to night: 100.3 Features: <ul style="list-style-type: none">o ✓ Conditionero ✓ Kitchenetteo ✓ Smoker Roomo ✓ Animals Accept <div>Favorite</div> <div>Add to waiting list</div>	Room 1 of Panoramic Hotel Plaza Room number: 1 Room people: 4 Price to night: 120.5 Features: <ul style="list-style-type: none">o ✓ Conditionero X Kitchenetteo ✓ Smoker Roomo X Animals Accept <div>To book</div> <div>Favorite</div>
--	---	---

Le stanze di colore verde sono prenotabili, cioè per le caratteristiche indicate è possibile prenotare quella stanza, solo se l'utente è loggato, altrimenti verrà reindirizzato sulla pagina di login.

Mentre le stanze rosse, sono già prenotate da un altro utente, pertanto è possibile solo aggiungersi ad una lista d'attesa per ricevere una mail nel caso in cui l'utente che ha prenotato la stanza decide di cancellare la prenotazione.

Per entrambe le tipologie è possibile aggiungere la stanza selezionata nei preferiti.

USER

3.1 INTRODUZIONE USER

Se l'utente è loggato, dalla home può accedere alla propria area personale, in cui può consultare:

- Voti
- Prenotazioni effettuate
- Favoriti
- Lista d'attesa

-Add room/location

3.2 VOTI

Location: Hotel Maranello Village

Questo moderno hotel, ispirato al mondo Ferrari, che espone una vera auto di Formula 1 nella hall, si trova a 3,5 km dal Museo Ferrari e a 5 km dal Modena Golf & Country Club.

Room number: 3

Prenotation from: Oct. 22, 2016, midnight to Nov. 6, 2016, midnight

Location position:

Via Terra delle Rosse, 12 - Maranello - Modena - Italy

[Site web Hotel Maranello Village](#)

Vote your holiday:

In ogni prenotazione l'utente (che ha effettuato la prenotazione) può decidere se esprimere un voto da 1 a 5 alla propria vacanza.

Votando, verrà reindirizzato sulla pagina dei voti:

Last Vote Of this User:		
User	Location	Vote
admin	Hotel Al Cavallino Bianco	3.0
admin	Hilton Prague	3.0
admin	Abitalia tower plaza	5.0

Dove potrà visualizzare i suoi ultimi voti e la media dei voti di ogni locations.

Media voti, tra tutti i voti espressi dagli utenti, in modo tale che gli utenti possano farsi un'idea basata su i voti espressi da tutti.

Media Vote All User:

Location	Media
Hotel Maranello Village	1.000
Abitalia tower plaza	2.333
Hilton Prague	1.500
Hotel Al Cavallino Bianco	4.000

3.3 PRENOTAZIONI EFFETTUATE

Per ogni prenotazione l'utente può decidere se eliminarla tramite un click sul relativo bottone.

3.4 FAVORITI

Location: Hotel Maranello Village

Questo moderno hotel, ispirato al mondo Ferrari, che espone una vera auto di Formula 1 nella hall, si trova a 3,5 km dal Museo Ferrari e a 5 km dal Modena Golf & Country Club.

Room number: 3

Location position:

Via Terra delle Rosse, 12 - Maranello - Modena - Italy

[Site web Hotel Maranello Village](#)

Delete from favorites

In questa sezione l'utente può visualizzare solamente i propri preferiti, ed ha inoltre la possibilità di eliminarli.

3.5 LISTA D'ATTESA

Location: Hotel Al Cavallino Bianco

Situato a 3 minuti a piedi dalla spiaggia di Riccione, questo hotel moderno con arredi sui toni del bianco dista 10 minuti a piedi dalla fs della città e 6 km dal parco.

Room number: 7

User booked: paolo

User attend: admin

Location position:

Viale Dante, 105 - Riccione - Rimini - Italy

[Site web Hotel Al Cavallino Bianco](#)

Delete from waiting list

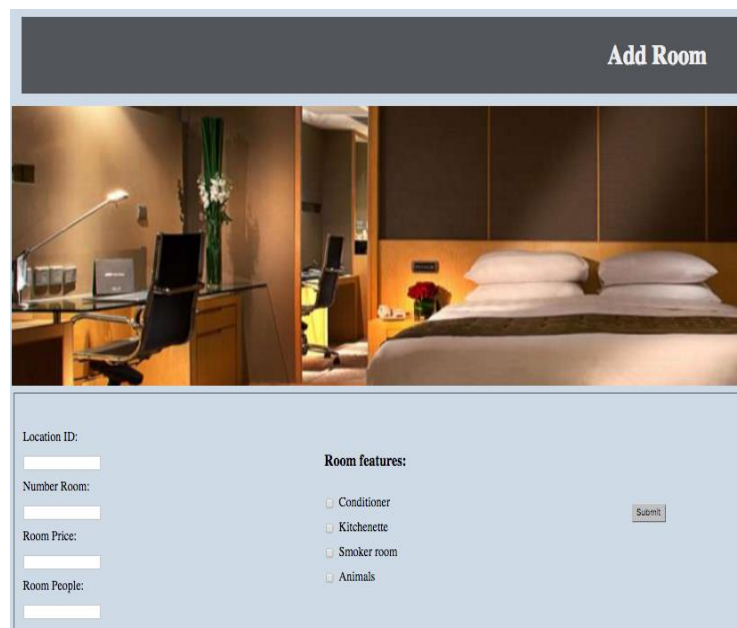
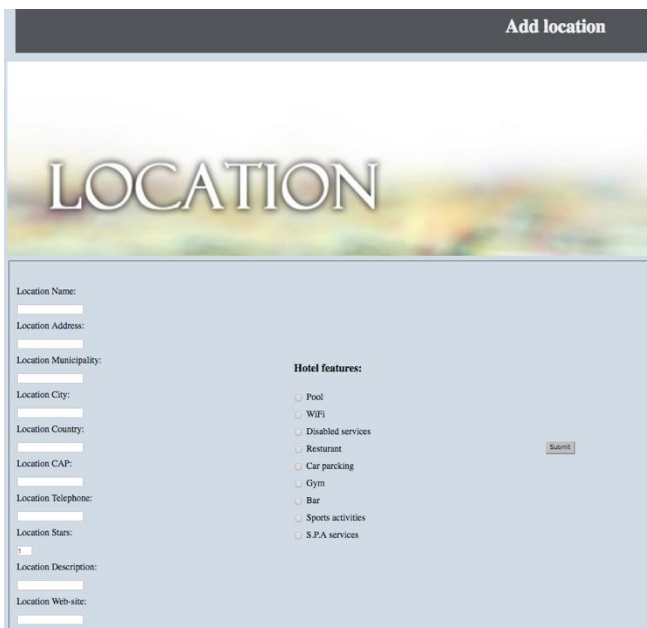
In questa sezione l'utente può visualizzare le Location per cui ha chiesto la lista d'attesa. Inoltre ha la possibilità di eliminarsi dalla lista d'attesa.

In modo automatico, quando viene eliminata una prenotazione, tramite un ciclo si controlla se esistono liste di attesa con lo stesso *id*, in questo caso viene inviata una mail all'utente che ha inserito quella prenotazione nella sua lista d'attesa.

send_mail(subject, message, from_email, to_email, fail_silently=False)

Per comodità, è stata commentata questa istruzione poiché per funzionare, è necessario disabilitare dei sistemi di sicurezza nei nostri account Gmail.

3.6 ADD LOCATION/ROOM



Queste sezioni vengono mostrate solo agli utenti con i permessi per accedervi, altrimenti si viene rimandati alla pagina di Login.

@permission_required('TravelBook.add_room')

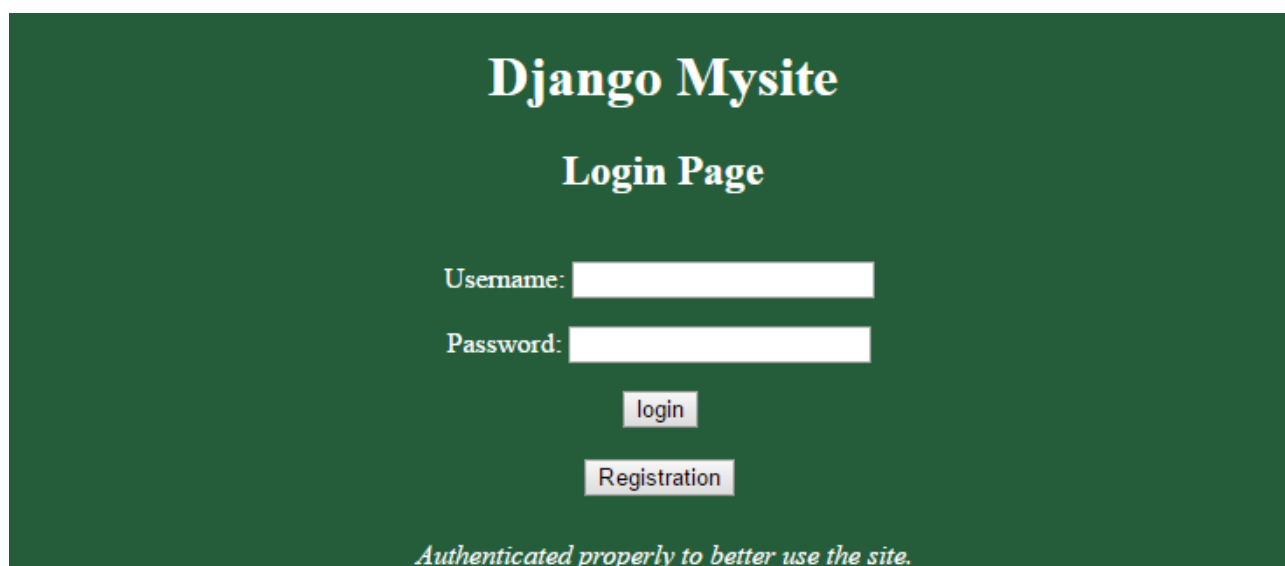
In queste sezioni si possono aggiungere room e location.

Nella relativa views, viene creata una nuova istanza di room o location se e solo se non è già presente nel database, questo è possibile tramite:

RoomFavorite.objects.get_or_create(room_favorite=get_object_or_404(Room, pk=r.pk), user_name=get_object_or_404(User, pk=u.id)).

LOGIN

4.1 LOGIN

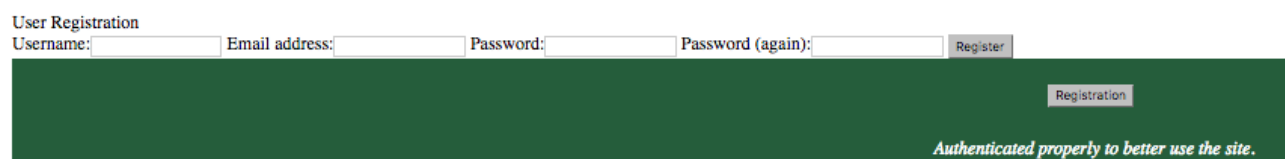


The screenshot shows a login page with a dark green background. At the top, the text "Django Mysite" is displayed in a large, white, serif font. Below it, "Login Page" is written in a smaller, white, serif font. There are two white input fields: the first is labeled "Username:" and the second is labeled "Password:". Below the password field is a small, light gray button labeled "login". Below the "login" button is another small, light gray button labeled "Registration". At the bottom of the page, the text "Authenticated properly to better use the site." is written in a small, italicized, white font.

Schermata che appare a qualsiasi utente anonimo che sta visitando il sito quando clicca su Login della Home. In questa schermata l'utente può loggarsi inserendo il proprio username e la propria password nei campi appositi.

Se un utente non è già registrato al sito allora può cliccare sul pulsante Registration per registrarsi al sito.

4.2 REGISTRAZIONE

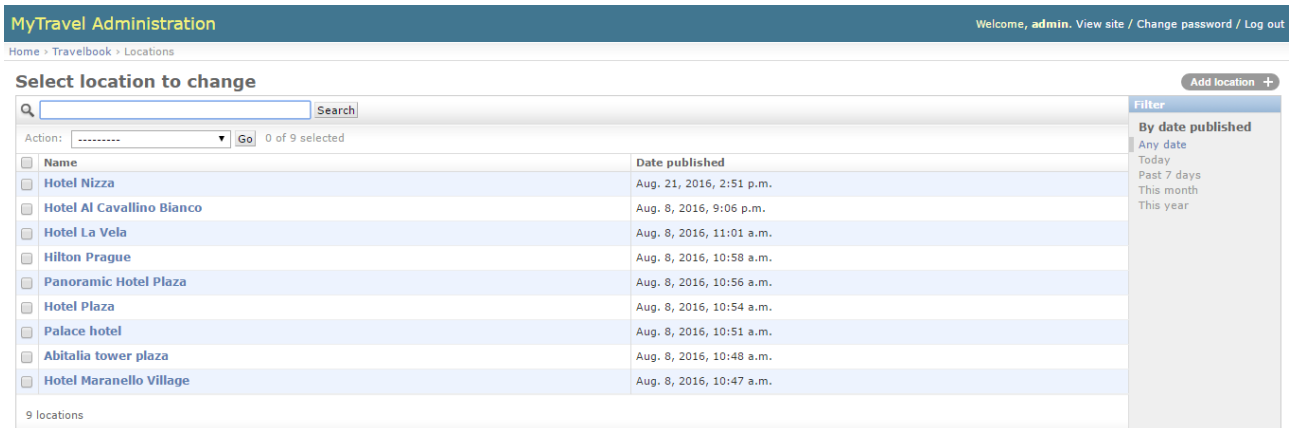


The screenshot shows a user registration page with a dark green background. At the top, the text "User Registration" is displayed in a small, white, sans-serif font. Below it, there are four white input fields: "Username:", "Email address:", "Password:", and "Password (again:". To the right of the "Password (again:" field is a small, light gray button labeled "Register". Below the "Register" button is another small, light gray button labeled "Registration". At the bottom of the page, the text "Authenticated properly to better use the site." is written in a small, italicized, white font.

Il nuovo utente inserendo uno username, il proprio indirizzo mail ed una password personale può registrarsi al sito in modo da poter effettuare prenotazione ed inserirsi in liste d'attesa.

APPENDICI

5.1 INTERFACCIA ADMIN



L'interfaccia di admin oltre a visualizzare tutte le location, permette:

- L'ordinamento in base al nome o la data di pubblicazione
Tramite il comando `list_display['name', 'pub_date']` di admin.
- L'aggiunta di una location
- Il filtro per la data di pubblicazione
Tramite il comando `list_filter['pub_date']` di admin.
- L'eliminazione di una o più location

5.2 JAVASCRIPT

In TravelBook è stato usato del codice Javascript per:

- Orologio
- Messaggi pop up
- Controlli per le form

OROGLOGIO	CONTROLLI
<pre>function time() { var today=new Date(); var h=today.getHours(); var m=today.getMinutes(); var s=today.getSeconds(); m = checkTime(m); s = checkTime(s); document.getElementById('txt').innerHTML = h+":"+m+":"+s; var t = setTimeout(function() {time()},500); }</pre>	<pre>function validateForm() { var x = document.forms["myForm"]["check_in"].value; if (x == null x == "") { alert("Check in must be filled out"); return false; } }</pre>

```
function checkTime(i) {
    if (i<10) {i = "0" + i};
    return i;
}
```

5.3 TEST

```
from django.test import TestCase
from TravelBook.models import Location,Room,Prenotation
from django.test.utils import setup_test_environment
from django.core.urlresolvers import reverse
from django.test.client import Client
# Create your tests here.
class Tests(TestCase):
    def test_index_view_with_no_locations(self):
        response=self.client.get('/TravelBook/index')
        self.assertEqual(response.status_code,200)
        self.assertContains(response,"No location exist")
        self.assertQuerysetEqual(response.context['location'],[])

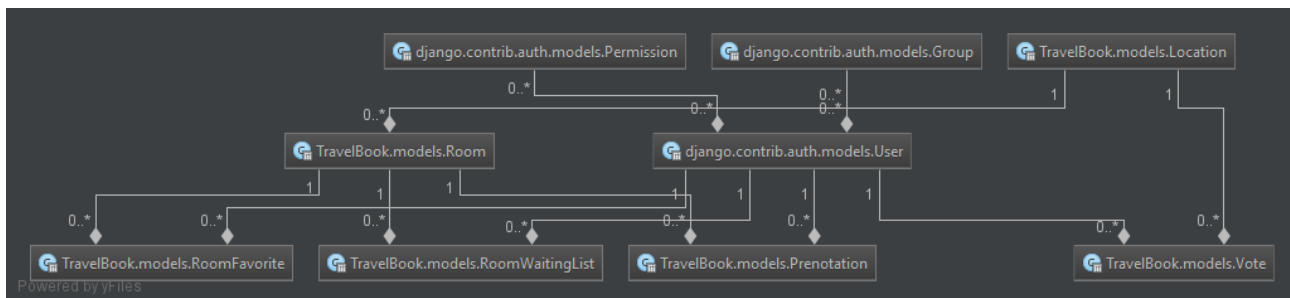
    def test_prenotation_with_wrong_check_out(self):
        prenotation=Prenotation.objects.all()
        data_in=[]
        data_out=[]
        for p in Prenotation.objects.all():
            data_in.append(p.check_in)
        for p in prenotation:
            data_out.append(p.check.out)
        i=0
        while i<len(data_in):
            date_verify=data_in[i]-data_out[i]
            if date_verify.days>0: self.assertTrue(False,"prenotazione con data antecedente")
            i=i+1
```

I due test fatti riguardano il primo la view, testando che la pagina “/TravelBook/index” non restituisca un errore e che siano presenti Location.

Il secondo test riguarda invece il codice interno, il fatto che una “Prenotazione” debba obbligatoriamente avere una data di Check-in(arrivo) che sia antecedente alla data di Check-out(partenza).

5.4 DIAGRAMMA DELLE CLASSI

5.4.1 Diagramma delle classi Django



5.4.2 Diagramma delle classi models

