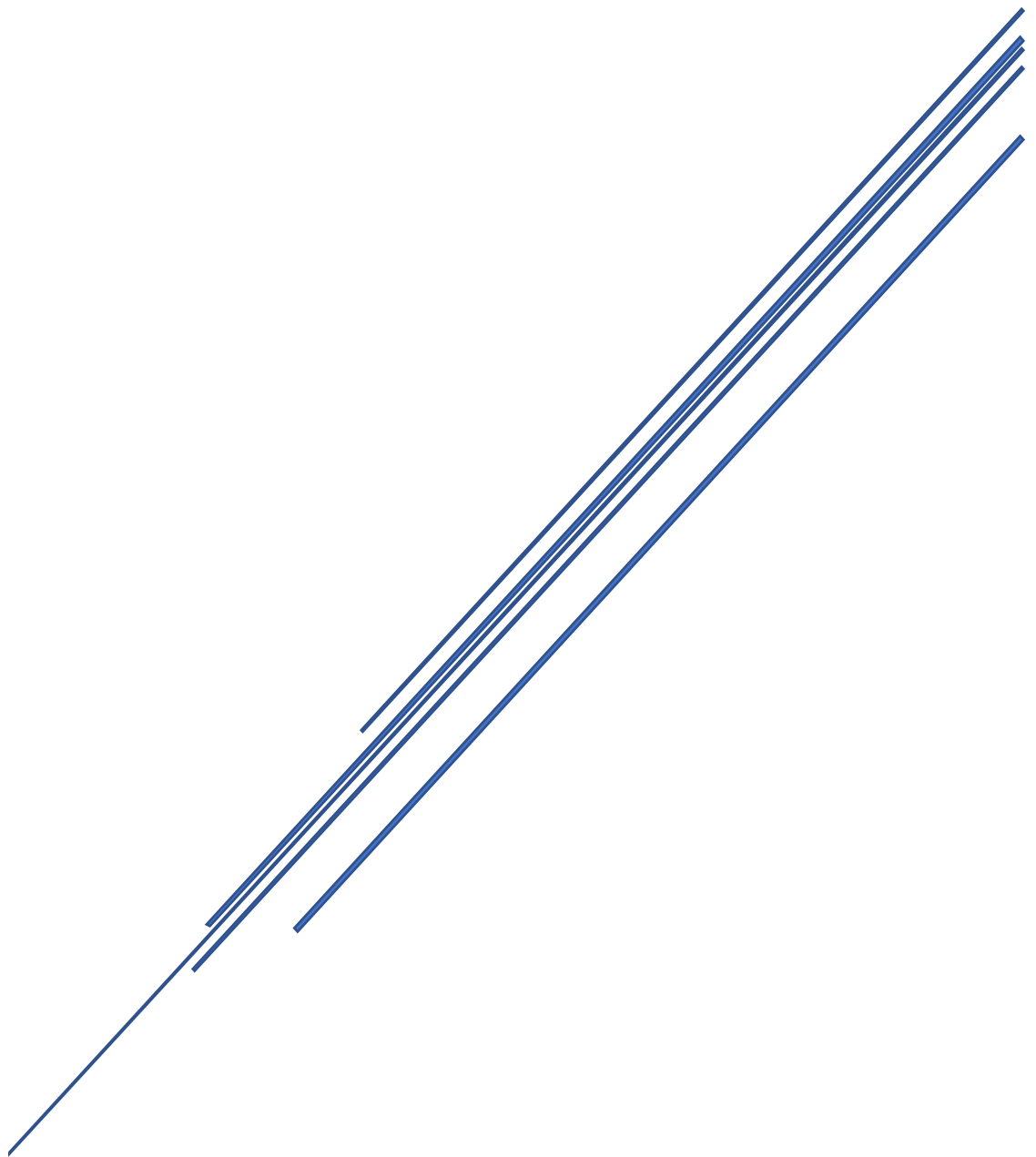


# MY BOOKING APP

Progetto Linguaggi Dinamici



Stefano Gregorio  
Marco Mezzetti

## Sommario

Sommario .....	1
<b>INTRODUZIONE .....</b>	<b>2</b>
<b>TRACCIA .....</b>	<b>2</b>
<b>1.1 INTRODUZIONE .....</b>	<b>3</b>
<b>1.2 MODELLO DEI DATI .....</b>	<b>4</b>
<b>FUNZIONE RICERCA E PRENOTAZIONE .....</b>	<b>5</b>
<b>2.1 FUNZIONE RICERCA .....</b>	<b>5</b>
<b>2.2.1 IMPLEMENTAZIONE .....</b>	<b>6</b>
<b>2.2 FUNZIONE PRENOTAZIONE .....</b>	<b>7</b>
<b>2.2.1 IMPLEMENTAZIONE .....</b>	<b>8</b>
<b>FUNZIONE DI LOGIN .....</b>	<b>9</b>
<b>3.1 PROCEDURA DI LOGIN .....</b>	<b>9</b>
<b>3.2 IMPLEMENTAZIONE .....</b>	<b>9</b>
<b>FUNZIONE DI REGISTRAZIONE UTENTE .....</b>	<b>9</b>
<b>4.1 PROCEDURA DI REGISTRAZIONE .....</b>	<b>9</b>
<b>4.2 SCELTE IMPLEMENTATIVE .....</b>	<b>10</b>
<b>4.3 IMPLEMENTAZIONE .....</b>	<b>11</b>
<b>FUNZIONE CREAZIONE HOTEL/CAMERA .....</b>	<b>12</b>
<b>5.1 FUNZIONE CREAZIONE HOTEL - CAMERA .....</b>	<b>12</b>
<b>5.2 IMPLEMENTAZIONI .....</b>	<b>13</b>
<b>PROCESSO GESTIONE PROFILO UTENTE .....</b>	<b>14</b>
<b>6.1 FUNZIONE GESTIONE PROFILO UTENTE .....</b>	<b>14</b>
<b>CONTROLLI JAVASCRIPT .....</b>	<b>15</b>
<b>VOTAZIONE .....</b>	<b>16</b>
<b>DIAGRAMMA DELLE CLASSI .....</b>	<b>17</b>

# INTRODUZIONE

## TRACCIA

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, foto, ...
- 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 determinate dalla ricerca in ordine di giudizio degli utenti (in alternativa al prezzo).

## 1.1 INTRODUZIONE

MyBookingApp è un' applicativo web per la gestione di un sistema di prenotazioni alberghiere realizzato attraverso l'utilizzo del framework Django.

Tale applicativo permette all' utente di prenotare una stanza in un determinato periodo.

Sono presenti 5 macro argomenti in cui è possibile racchiudere le principali funzioni dell'applicativo:

### 1. Processo di Ricerca e Prenotazione

In questo processo l' utente può eseguire la ricerca di una struttura alberghiera e delle stanze correlate inserendo le caratteristiche ed il periodo desiderati.

Come risultato della ricerca, saranno visualizzabili le camere con i servizi richiesti; in verde avremo quelle prenotabili e in rosso quelle occupate nel periodo cercato.

### 2. Processo di login

All'interno dell'applicativo è prevista una sezione per effettuare il login. Questa procedura è necessaria in quanto, come da specifiche di progetto, solo gli utenti loggati avranno la possibilità di prenotare una stanza o gestire il proprio profilo utente.

### 3. Processo di Registrazione Utente

Questo processo gestisce la registrazione all'applicativo web. Sarà possibile effettuare una registrazione, attraverso gli appositi pulsanti presenti nell' header del sito, per due tipi di profili differenti: Utente e Gestore.

Il primo avrà la possibilità di effettuare le prenotazioni, modificarle ed eleminarle, il tutto tramite la pagina di gestione apposita.

Il secondo, invece, avrà la possibilità di creare una struttura alberghiera e le relative stanze.

## 4. Processo Creazione Hotel/Stanza

Attraverso questo processo, l'utente avente permessi di gestione, potrà creare strutture alberghiere e relative stanze.

## 5. Processo Gestione Profilo Utente

Questo processo gestisce la userpage che è diversa a seconda dei permessi che l'utente possiede.

# 1.2 MODELLO DEI DATI

Il seguente capitolo mostra le scelte effettuate, durante il periodo di analisi, per quanto riguarda il modello dei dati delle entità presenti.

```
class Hotel(models.Model):
    nome = models.CharField(max_length=200)
    indirizzo = models.CharField(max_length=300)
    citta = models.CharField(max_length=200)
    stato = models.CharField(max_length=200)
    telefono_reges = RegexValidator(regex=r'^\d{9,15}$', message='Phone number must be entered in the format: '+9999999999'. Up to 15 digits allowed.')
    num_telefono = models.CharField(validators=[telefono_reges], max_length=15, blank=True)
    piscina = models.BooleanField(default=False)
    wifi = models.BooleanField(default=False)
    accesso_disabili = models.BooleanField(default=False)
    ristorante = models.BooleanField(default=False)
    parcheggio = models.BooleanField(default=False)
    palestra = models.BooleanField(default=False)
    bar = models.BooleanField(default=False)
    spa = models.BooleanField(default=False)
    descrizione = models.CharField(max_length=1000)
    sito = models.CharField(max_length=60)
    media_voto = models.FloatField(default=0)
    direttore = models.CharField(max_length=200)
    pub_date = models.DateField('date published')

    def __str__(self):
        return self.nome

class Stanza(models.Model):
    id_hotel = models.ForeignKey(Hotel)
    num_camera = models.PositiveIntegerField()
    prezzo = models.FloatField(validators=[MinValueValidator(1)])
    prezzo_festivita = models.FloatField(validators=[MinValueValidator(1)])
    num_persone = models.PositiveIntegerField(validators=[MinValueValidator(1)])
    aria_condizionata = models.BooleanField(default=False)
    camera_fumatori = models.BooleanField(default=False)
    animal = models.BooleanField(default=False)

    class Meta:
        unique_together = (('id_hotel', 'num_camera'),)

    def __str__(self):
        return self.id_hotel.nome + " - Camera Numero: " + str(self.num_camera)

class Prenotazioni(models.Model):
    id_stanza = models.ForeignKey(Stanza)
    id_user = models.ForeignKey(User)
    check_in = models.DateField('check_in')
    check_out = models.DateField('check_out')

class Voto(models.Model):
    hotel_id = models.ForeignKey(Hotel)
    user_id = models.ForeignKey(User)
    voto = models.FloatField(validators=[MinValueValidator(0), MaxValueValidator(5)])

    class Meta:
        unique_together = (('hotel_id', 'user_id'),)

    def __str__(self):
        return self.hotel_id.nome

class ListaAttesaStanza (models.Model):
    lista_attesa = models.ForeignKey(Stanza)
    user_id = models.ForeignKey(User)
    check_in_lista_attesa = models.DateField('check in date')
    check_out_lista_attesa = models.DateField('check out date')

    def __str__(self):
        return self.lista_attesa.id_hotel.nome + "Camera num: " + str(self.lista_attesa.num_camera)
```

# FUNZIONE RICERCA E PRENOTAZIONE

In questo capitolo andremo ad illustrare come viene eseguita la funzione di ricerca e quella di prenotazione.

## 2.1 FUNZIONE RICERCA

La prima pagina che risulta visibile una volta avviata la web application è la pagina di ricerca.

Questa pagina, al primo avvio, presenta un header contenente alcuni pulsanti di navigazione e un form non compilato.

I pulsanti che possono essere visualizzati nella barra variano in base alle seguenti regole:

- I pulsanti di registrazione utente e registrazione gestore verranno visualizzati solo se l'utente in sessione non risulta loggato
- Il pulsante di Login risulterà presente solo nel caso in cui l'utente non risulti loggato nella sessione attuale. A Login avvenuto, tale pulsante si trasformerà in un pulsante di Logout.
- Il pulsante a fianco di Login/Logout, visualizzerà il valore "Anonymous" se l'utente non risulta loggato, mentre in caso contrario visualizzerà il nome dell'utente loggato.



Per quanto riguarda il form di ricerca, saranno visualizzati i campi contenenti le caratteristiche per il filtraggio delle strutture e delle relative camere, un campo, non obbligatorio, per il filtraggio delle strutture in funzione della città di appartenenza e due campi per la scelta delle date di check-in e check-out.

Il risultato della ricerca sarà un elenco delle strutture disponibili con le caratteristiche inserite dall'utente. Le camere libere saranno identificate da un colore di sfondo verde, mentre avranno un colore di sfondo rosso le stanze non prenotabili. Per le camere disponibili comparirà un pulsante per effettuare la prenotazione mentre per le camere prenotate avremo un pulsante per aggiungere la stanza richiesta alla propria wishlist.

## 2.2.1 IMPLEMENTAZIONE

Quando l'utente avvia MyBookingApp, viene eseguita la funzione “search” , dal file view.py, strutturata nel seguente modo:

- Quando atterriamo sulla pagina iniziale, viene visualizzato solo il form di ricerca.
- Una volta compilato il form e premuto il tasto “Cerca”, allora viene eseguita la “search” che filtra gli hotel presenti nel database. Il processo di filtraggio avviene seguendo le

informazioni presenti nel form di ricerca, effettuando un ulteriore controllo per verificare la disponibilità delle stanze nel periodo selezionato dall'utente. E' stato scelto inoltre di escludere dall'elenco delle strutture alberghiere visualizzate tutte quelle che non possedevano stanze correlate.

I dati passati alla pagina "indexsearch.html" tramite la funzione render sono:

- Hotel filtrati
- Form di ricerca
- Stanze filtrate
- Stanze prenotate

```
return render(request, 'indexsearch.html',
               {'data': data, 'risultati_hotel': Filtered_hotels, 'form_search': form_search_hotel,
                'stanze_filtrate': risultati_stanze, "stanze_prenotate": prenotazioni_filtrate,
                "id_prenotate": p})
```

## 2.2 FUNZIONE PRENOTAZIONE

Nel caso in cui venisse scelto di proseguire nella prenotazione, verranno eseguiti due controlli.

Il primo controllo è la verifica che l'utente che sta procedendo alla prenotazione sia loggato; in caso negativo, quest'ultimo verrà dirottato sulla pagina di login.

Superato questo primo controllo, ne verrà effettuato un secondo che controlla il gruppo di permessi di cui fa parte l'utente loggato. Se l'utente che sta tentando la prenotazione fa parte del gruppo Utente, quest'ultimo verrà dirottato sulla pagina di riepilogo della prenotazione stessa, potendo poi confermarla. Se, in caso contrario, l'utente dovesse fare parte del gruppo Direttore, l'accesso gli sarà negato in base ad una scelta fatta in fase implementativa. Questo poiché è stato deciso che l'utente facente parte del gruppo Direzione potesse solo gestire le strutture alberghiere e le relative camere.

Dopo questi due controlli, se la prenotazione risulta essere andata a buon fine, l'utente verrà reindirizzato alla pagina di ricerca

Nel caso in cui la stanza prescelta dovesse essere prenotata, l'utente potrà aggiungerla alla wishlist. Se la stanza aggiunta in wishlist dovesse risultare disponibile a causa della modifica o della cancellazione della prenotazione relativa, gli utenti in wishlist per quella stanza in quel periodo riceveranno un messaggio di posta elettronica.



MyBookingApp

stefano Logout

# Riepilogo Prenotazione

Hotel:	Mondial Resort & Spa
Stanza:	2
Indirizzo:	Via Duca della Vittoria 129/131, 55044
Città:	Marina di Pietrasanta
Utente Prenotazione:	stefano
Data Check In:	2018-03-11
Data Check Out:	2018-03-12

Conferma Prenotazione

## 2.2.1 IMPLEMENTAZIONE

Per quanto riguarda la funzione di prenotazione, abbiamo due casi: l'utente risulta loggato o non loggato. Nel caso invece l'utente risulti loggato, verrà controllato il suo gruppo di permessi e nel caso quest'ultimo facesse parte del gruppo Utente, verrà compilato il form e avremo la conferma della prenotazione. In caso di utente non loggato, la funzione salverà i dati inerenti alla prenotazione, reindirizzando lo user alla funzione di login. A quel punto l'utente, una volta loggato, avrà la possibilità di confermare la prenotazione che comporterà la creazione di un oggetto contenente:

- Identificativo della camera
- Identificativo dell'utente che sta effettuando la prenotazione
- Data di check in
- Data di check out

# FUNZIONE DI LOGIN

## 3.1 PROCEDURA DI LOGIN

Questa procedura ha il compito di effettuare l'autenticazione all'utente.

La procedura di login può essere scatenata nelle seguenti occasioni:

- Espressa volontà dell'utente tramite pressione dell'apposito pulsante
- Tentativo di prenotazione di una camera senza che sia avvenuta l'autenticazione

## 3.2 IMPLEMENTAZIONE

Distinguiamo due situazioni possibili:

- Se la richiesta è una "GET", verrà restituito il form per il login non compilato.
- Se la richiesta è una "POST", alla pressione del tasto di submit del form di login verranno presi in considerazione i dati inseriti (username e password) e verrà tentata l'autenticazione.

Sia in caso positivo che in caso negativo, la funzione di login avrà come risultato il reindirizzamento alla pagina di ricerca dell'applicazione.



# FUNZIONE DI REGISTRAZIONE UTENTE

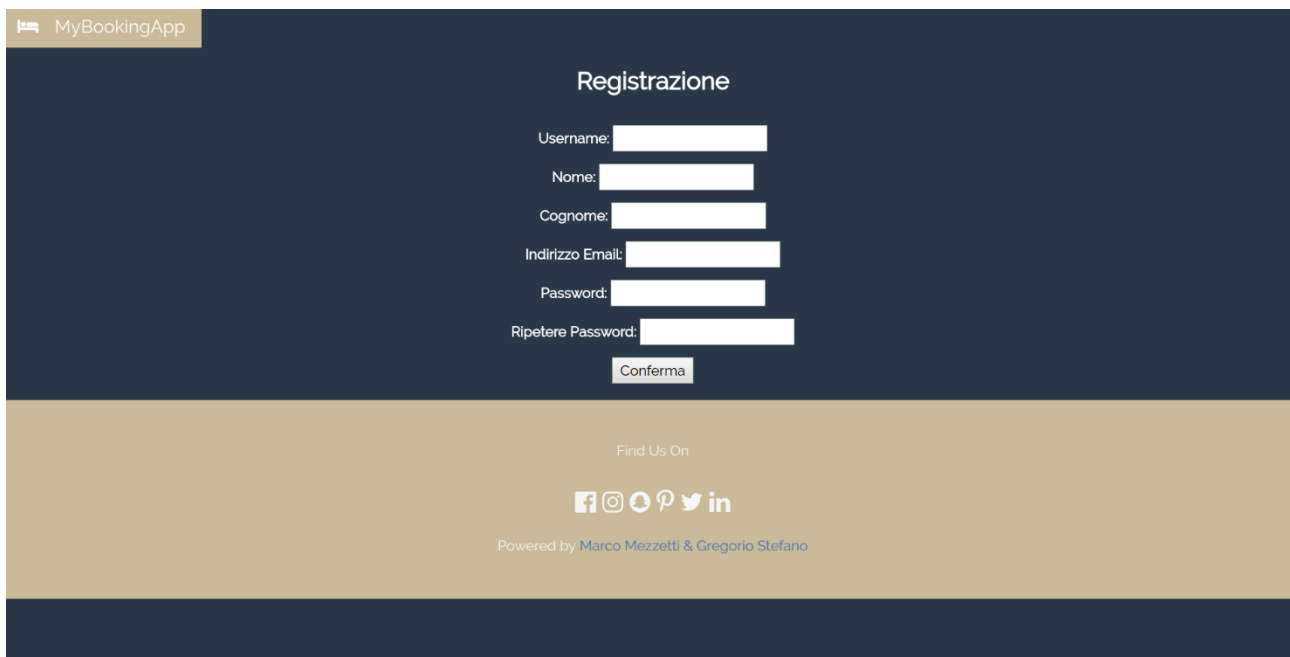
## 4.1 PROCEDURA DI REGISTRAZIONE

Questa funzione dà la possibilità ad un utente che sta eseguendo MyBookingApp di registrarsi e quindi di poter accedere alle funzioni avanzate che l'applicativo offre, senza fermarsi alla sola ricerca.

La procedura di registrazione viene eseguita solo sotto espressa volontà dell'utente all'evento di pressione dell'apposito pulsante, presente nell'header dell'applicazione stessa. Tale pulsante sarà visibile solo nel caso in cui, nella sessione attuale, non risulti nessun utente autenticato.

Durante questa procedura, l'utente avrà la possibilità di inserire i dati personali, oltre ad una password per effettuare l'autenticazione.

Il form per la registrazione dell'utente risulterà come nella figura sottostante:



The image shows a dark-themed header for 'MyBookingApp'. In the center, there is a registration form titled 'Registrazione'. The form contains the following fields: 'Username:', 'Nome:', 'Cognome:', 'Indirizzo Email:', 'Password:', and 'Ripetere Password:'. Each field is followed by a white input box. Below the 'Ripetere Password:' field is a 'Conferma' button. At the bottom of the header, there is a section with the text 'Find Us On' followed by social media icons for Facebook, Instagram, YouTube, Pinterest, Twitter, and LinkedIn. Below the icons, it says 'Powered by Marco Mezzetti & Gregorio Stefano'.

## 4.2 SCELTE IMPLEMENTATIVE

In fase di analisi per lo sviluppo dell'applicazione è stato deciso di differenziare le capacità in possesso di un utente comune rispetto a quelle di un gestore di strutture alberghiere.

Per questo motivo, sono stati creati due gruppi di permessi, rispettivamente il gruppo "Utente" e il gruppo "Direttore".

Gli utenti appartenenti al primo gruppo avranno la possibilità di prenotare le camere per il periodo desiderato, aggiungere quelle non disponibili ad una lista dei desideri, oltre che ad accedere alla userpage.

Per quanto riguarda gli utenti appartenenti al gruppo "Direzione", questi ultimi avranno la possibilità di aggiungere una o più nuove strutture alberghiere e le relative stanze. In questa fase di sviluppo di progetto non è stata implementata una pagina di gestione delle strutture, ampliamento che sarebbe bene fare in eventuali sviluppi futuri.

## 4.3 IMPLEMENTAZIONE

Nella funzione di registrazione dell'utente, viene effettuato un controllo per impedire l'accesso a tale funzione ad utenti già autenticati.

Nel caso in cui non risulti autenticato alcun utente, se la procedura è stata richiamata tramite la funzione "GET", verrà restituito il form per la registrazione non compilato.

Nel caso in cui fosse richiamata tramite la funzione "POST", se il form risulta valido, verranno analizzate le informazioni inserite dall'utente e creato un nuovo utente.

A seconda della scelta effettuata dall'utente in fase di registrazione, ovvero l'iscrizione come utente di direzione o meno, esso verrà assegnato al gruppo di permessi relativo.

```
def register_gestore(request):
    if not request.user.is_authenticated():
        if request.method == 'POST':
            form = RegistrationForm(request.POST)
            if form.is_valid():
                user = User.objects.create_user(
                    username=form.cleaned_data['username'],
                    first_name=form.cleaned_data['first_name'],
                    last_name=form.cleaned_data['last_name'],
                    password=form.cleaned_data['pwd1'],
                    email=form.cleaned_data['email']
                )
                print()
                group = Group.objects.get(name='Direzione')
                user.groups.add(group)
                return render(request, 'successocreazione direttore.html')
            else:
                form = RegistrationForm()
                return render(request, 'register.html', {'form': form})
        else:
            return render(request, 'accessonegato.html')
```

```
def register_user(request):
    if not request.user.is_authenticated():
        if request.method == 'POST':
            form = RegistrationForm(request.POST)
            if form.is_valid():
                user = User.objects.create_user(
                    username=form.cleaned_data['username'],
                    first_name=form.cleaned_data['first_name'],
                    last_name=form.cleaned_data['last_name'],
                    password=form.cleaned_data['pwd1'],
                    email=form.cleaned_data['email']
                )
                group = Group.objects.get(name='Utente')
                user.groups.add(group)
                return render(request, 'successocreazione utente.html')
            else:
                form = RegistrationForm()
                return render(request, 'register.html', {'form': form})
        else:
            return render(request, 'accessonegato.html')
```

# FUNZIONE CREAZIONE HOTEL/CAMERA

## 5.1 FUNZIONE CREAZIONE HOTEL - CAMERA

Questa funzione dà la possibilità ad un utente facente parte del gruppo di permessi “Direzione” di aggiungere una o più strutture alberghiere e le relative camere.

Come anticipato nel capitolo precedente, è stato scelto, in fase di sviluppo, di dare questa possibilità solamente a tali utenti, differenziandoli dagli utenti in grado di prenotare gli alloggi.

Le funzioni di creazione di una nuova struttura alberghiera e delle relative camere sono accessibili tramite la pagina di gestione dell’utente Direzione. Tale pagina presenterà due bottoni, uno per funzione.

Nel caso in cui la scelta ricadesse sulla creazione di una struttura alberghiera, il form che si presenterà all’utente sarà quello presente nell’immagine seguente:



Form for creating a hotel structure. The form is displayed on a light brown background and contains the following fields and options:

- Nome:
- Indirizzo:
- Città:
- Stato:
- Num telefono:
- Sito:
- Descrizione:
- Piscina: ☐
- WiFi: ☐
- Accesso disabili: ☐
- Ristorante: ☐
- Parcheggio: ☐
- Palestra: ☐
- Bar: ☐
- Spa: ☐
- Aggiungi

Nel caso in cui la scelta ricadesse sull'aggiunta una stanza, la pagina sarà la medesima, ma con la presenza del form relativo, come in figura:



Id hotel:

Num camera:

Prezzo:

Num persone:

Prezzo festività:

Aria condizionata: ☐

Camera fumatori: ☐

Animali: ☐

## 5.2 IMPLEMENTAZIONI

Per quanto riguarda queste due funzioni, l'accesso è precluso agli utenti non appartenenti al gruppo di direzione. Questo controllo è effettuato al primo avvio di entrambe le funzioni e, in caso negativo, l'utente verrà rediretto ad una pagina che figurerà un messaggio di accesso negato.

In caso contrario invece all'utente sarà presentato il form per la creazione di una struttura alberghiera o della relativa camera, a seconda della scelta effettuata.

I due form sono stati creati utilizzando i Django Form, come è possibile notare nelle immagini seguenti:

```
class AddRoomForm(forms.ModelForm):
    class Meta:
        model = Stanza
        fields = ('id_hotel', 'num_camera', 'prezzo', 'num_persone', 'prezzo_festivita', 'aria_condizionata', 'camera_fumatori', 'animali')

    def __init__(self, *args, **kwargs):
        user = kwargs.pop('user')
        super(AddRoomForm, self).__init__(*args, **kwargs)
        self.fields['id_hotel'].queryset = Hotel.objects.filter(direttore=user)

class AddHotelForm(forms.ModelForm):
    class Meta:
        model = Hotel
        fields = ('nome', 'indirizzo', 'citta', 'stato', 'num_telefono', 'sito', 'descrizione', 'piscina', 'WiFi', 'accesso_disabili', 'ristorante', 'parcheggio', 'palestra', 'bar', 'spa')
```

In particolare, per quanto riguarda il form per la creazione di una stanza, la dropdown relativa alla scelta della struttura alberghiera correlata, sarà popolata solamente dagli hotel in gestione all'utente attualmente loggato, tramite la funzione di init.

# PROCESSO GESTIONE PROFILO UTENTE

## 6.1 FUNZIONE GESTIONE PROFILO UTENTE

Questa pagina avrà due layout diversi a seconda del gruppo di appartenenza dell'utente che sta lanciando il sito. Nel caso in cui l'utente appartenga al gruppo di permessi "UTENTE" avremo la seguente schermata:



Nel caso in cui l'utente che tenta l'accesso a tale funzioni risulti essere appartenente al gruppo di direzione, gli saranno presentati due pulsanti per l'aggiunta di una nuova struttura alberghiera o di una nuova stanza.



# CONTROLLI JAVASCRIPT

È stato inserito un controllo javascript sulle date inserite nel form di ricerca. Tali controlli andranno ad analizzare le date inserite evitando l'immissione di una data di check-in antecedente alla data odierna o successiva alla data di check-out inserita.

In entrambi i casi, l'evento che scatena la ricerca verrà bloccato.

```
function DateControll()
{
    var check_in_month = document.getElementById("id_check_in_month");
    var check_out_month = document.getElementById("id_check_out_month");
    var check_in_day = document.getElementById("id_check_in_day");
    var check_out_day = document.getElementById("id_check_out_day");
    var check_in_year = document.getElementById("id_check_in_year");
    var check_out_year = document.getElementById("id_check_out_year");

    var todayMonth = new Date().getMonth() + 1;
    {#alert(todayMonth)#}
    var todayDay = new Date().getUTCDate();
    {#alert(todayDay)#}

    if ( check_in_month.options[check_in_month.selectedIndex].value < todayMonth)
    {
        event.preventDefault();
        alert("ERRORE DATA ANTECEDENTE A OGGI")
    }
    else if (check_in_month.options[check_in_month.selectedIndex].value >= todayMonth && check_in_day.options[check_in_day.selectedIndex].value < todayDay)
    {
        event.preventDefault();
        alert("ERRORE DATA ANTECEDENTE A OGGI")
    }
    else if (check_in_year.options[check_in_year.selectedIndex].value > check_out_year.options[check_out_year.selectedIndex].value)
    {
        event.preventDefault();
        alert("ERRORE ANNO ! LA DATA DI CHECKIN RISULTA ESSERE MAGGIORE DELLA DATA DI CHECKOUT")
        return false;
    }
    else if (check_in_month.options[check_in_month.selectedIndex].value > check_out_month.options[check_out_month.selectedIndex].value )
    {
        event.preventDefault();
        alert("ERRORE MESE! LA DATA DI CHECKIN RISULTA ESSERE MAGGIORE DELLA DATA DI CHECKOUT")
        return false;
    }
    {#else if (check_in_day.options[check_in_day.selectedIndex].value > check_out_day.options[check_out_day.selectedIndex].value)#}
    else if (check_in_day.selectedIndex > check_out_day.selectedIndex)
    {
        event.preventDefault();
        alert("ERRORE GIORNO! LA DATA DI CHECKIN RISULTA ESSERE MAGGIORE DELLA DATA DI CHECKOUT")
        return false;
    }
}
```



# VOTAZIONE

Ogni utente ha la possibilità di esprimere un giudizio su un Hotel in cui ha soggiornato.

Nello sviluppo di questa specifica sono stati impostati due vincoli chiave:

- L'utente può esprimere un voto solo dopo il giorno di check-out
- Lo stesso utente non può esprimere due volte un giudizio per la stessa struttura

```
def Votazione(request):
    hotels = Hotel.objects.all()
    voto = request.POST['stelle']
    hotel_id = int(request.POST['HotelID'])

    try:
        obj = Voto.objects.get_or_create(hotel_id_id=hotel_id, user_id_id=request.user.id, voto=voto)

        VotiHotel = {}

        for hId in Hotel.objects.all():
            votoTOT = 0
            counter = 0
            for record in Voto.objects.all():
                if hId.id == record.hotel_id_id:
                    counter += 1
                    votoTOT += record.voto

            if counter != 0:
                VotiHotel[hId.id] = votoTOT / counter

        for key, value in VotiHotel.items():
            for h in Hotel.objects.all():
                print(key, h.id)
                if key == h.id:
                    print("Aggiorno il voto ")
                    Hotel.objects.filter(id = key).update(media_voto = value)

    return HttpResponseRedirect('/myBookingApp_2/userpage')
```

# DIAGRAMMA DELLE CLASSI

