Progetto Tecnologie Web

Ristorante Belvedere

Autore:
Pasini Francesco

matricola 169109

Indice

1. Introduzione 3
- 1.1. traccia d'esame 3
- 1.2. tipologie di utenti 3
  - 1.2.1. utente non registrato 3
  - 1.2.2. cliente 3
  - 1.2.3. ristoratore 4

2. Descrizione del progetto 4
- 2.1. Use Case diagram 5
- 2.2. Class diagram 6
- 2.3. Activity diagram 7

3. Tecnologie e applicazioni utilizzate 8

4. Scelte progettuali 9

5. Testing 10

6. Conclusioni 11

1. Introduzione

Il progetto consiste nello sviluppo di una applicazione per gestire il ristorante tradizionale Belvedere, permettendo agli utenti di effettuare ordinazioni sul relativo menù e di inoltrare richieste lavorative riguardo agli annunci messi a disposizione dal locale.

1.1 Traccia

Sviluppare una web-app per la gestione delle ordinazioni del ristorante Belvedere, caratterizzata da diverse tipologie di utenti con diverse azioni possibili:

1.2 Tipologie di utenti

1.2.1 Utente non registrato (Anonymous user)

L'utente non registrato ha la possibilità di visualizzare la homepage del ristorante, dove sono presenti informazioni e contatti di quest'ultimo. Ha inoltre la possibilità di visualizzare il menù dove sono presenti i prodotti del ristorante.
Potrà quindi accedere alle seguenti pagine:

- **home :** pagina principale dove sono presenti informazioni e contatti.
- **menu :** pagina per la visualizzazione dei prodotti del ristorante.
- **login e registrazione :** ha la possibilità di registrarsi ed effettuare il login

1.2.2 Utente registrato (Cliente)

L'utente registrato presso il ristorante implementa tutte le azioni dell'utente anonimo, con l'aggiunta di funzionalità relative alla creazione e valutazione degli ordini e l'inoltro di richieste lavorative riguardanti gli annunci presenti. Il cliente ha inoltre la possibilità di modificare il suo profilo aggiungendo nome, cognome, numero di telefono e immagine.
Potrà quindi accedere alle seguenti pagine:

- **home :** pagina dove il cliente, oltre alle informazioni del ristorante, visualizzerà anche gli ordini effettuati e quelli conclusi, con la possibilità di effettuare una valutazione.
- **menu :** come per l'utente anonimo, ma con la possibilità di selezionare i prodotti e aggiungerli all'ordine.
- **carrello :** pagina nella quale verrà mostrato l'ordine corrente, con il resoconto ed eventuale possibilità di annullarlo o confermarlo.
- **lavora con noi :** pagina dove il cliente può visualizzare e rispondere agli annunci attivi, aggiungendo una descrizione che verrà visualizzata dal ristoratore.
- **impostazioni :** pagina relativa alle informazioni personali dell'utente, modificabili da quest'ultimo.

1.2.3 Utente Staff (Ristoratore)

L'utente staff gestisce gli ordini in attesa dei clienti tramite la sua homepage personalizzata, con la possibilità di concluderli. Può inoltre modificare il menù aggiungendo o eliminando i prodotti e gestire gli annunci e le richieste lavorative dei clienti tramite la relativa pagina. Potrà quindi accedere alle seguenti pagine:

- **home :** pagina principale dove visualizza gli ordini in attesa, con la possibilità di concluderli.
- **menu :** pagina che permette, oltre alla visualizzazione, di modificare il menù aggiungendo o eliminando i prodotti.
- **gestione annunci/richieste :** pagina dove vengono visualizzati gli annunci attivi e le richieste lavorative presentate dai clienti. Il ristoratore può creare o eliminare un annuncio e gestire le richieste, decidendo di scartarle o accettarle.

2. Descrizione del progetto

Per lo sviluppo del progetto viene utilizzato il framework Django, che lavora in python, per lo sviluppo dell'applicazione web. In seguito vi sono schemi dimostrativi riguardanti l'organizzazione ed il funzionamento del progetto.

2.1 Use Case diagram

Diagramma per la rappresentazione del progetto nella sua interezza, con al suo interno le funzionalità da implementare, i tipi di utente all'interno del sistema e le azioni da loro eseguibili.

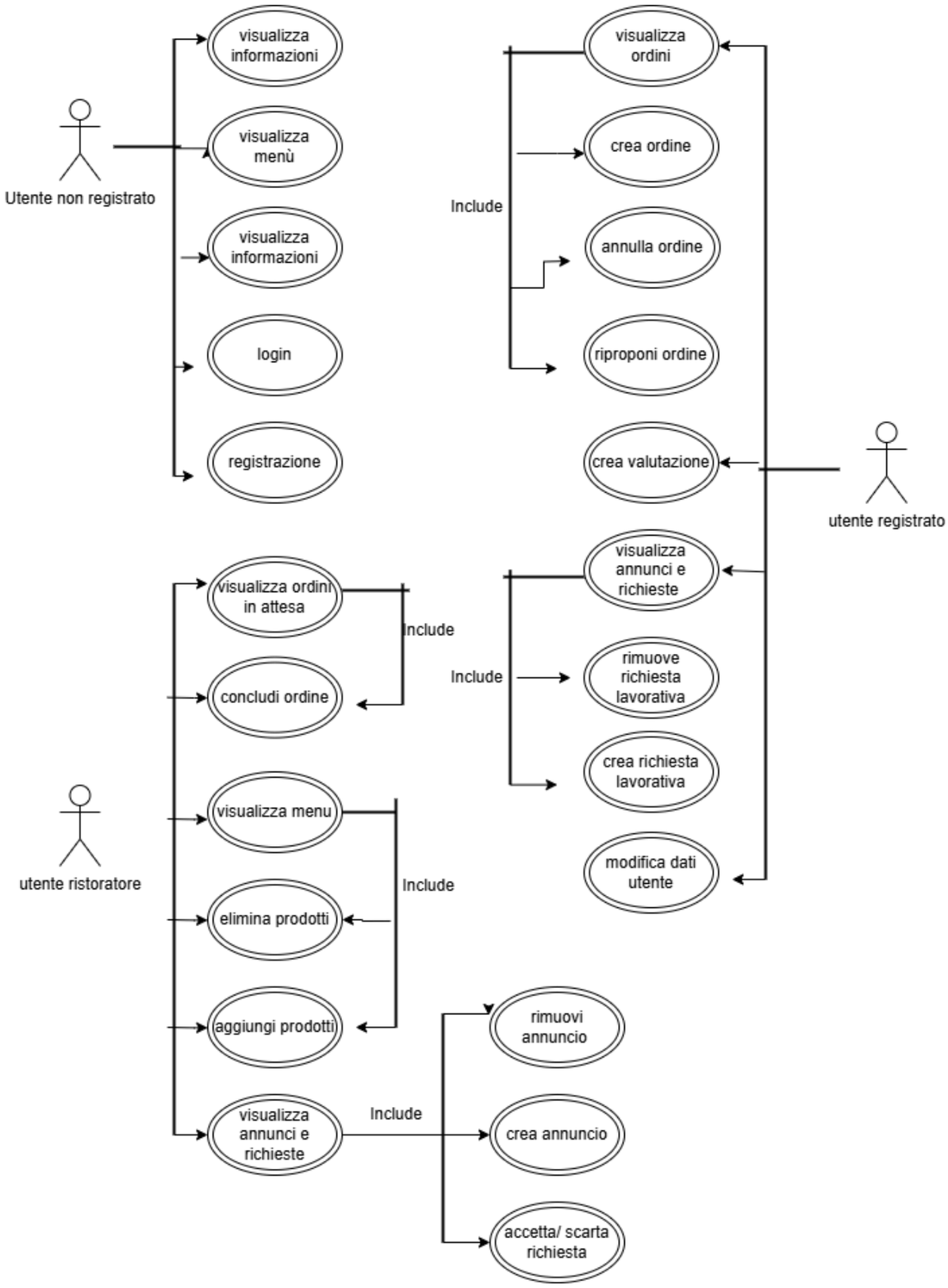

Figura 1 : Use Case diagram

2.2 Class diagram

Il Class diagram permette la rappresentazione grafica del model utilizzato e delle relazioni presenti al suo interno. Generato tramite l'utilizzo del comando graph_models e di graphviz.

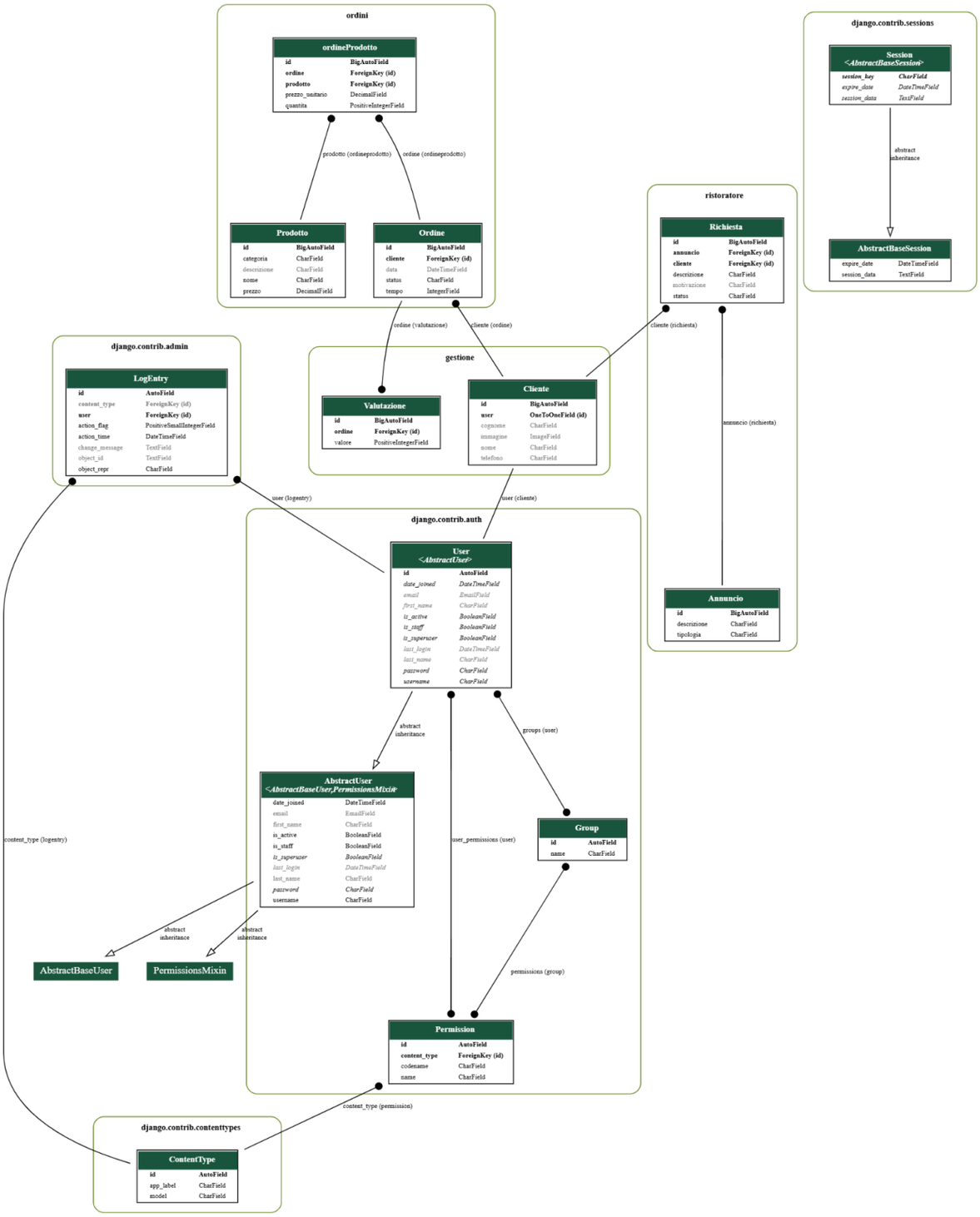

Figura 2: Class diagram del progetto

2.3 **Activity diagram**

Di seguito vengono mostrate le attività per due casi d'uso: il primo diagramma mostra le attività per poter inoltrare una richiesta lavorativa per un annuncio specifico.

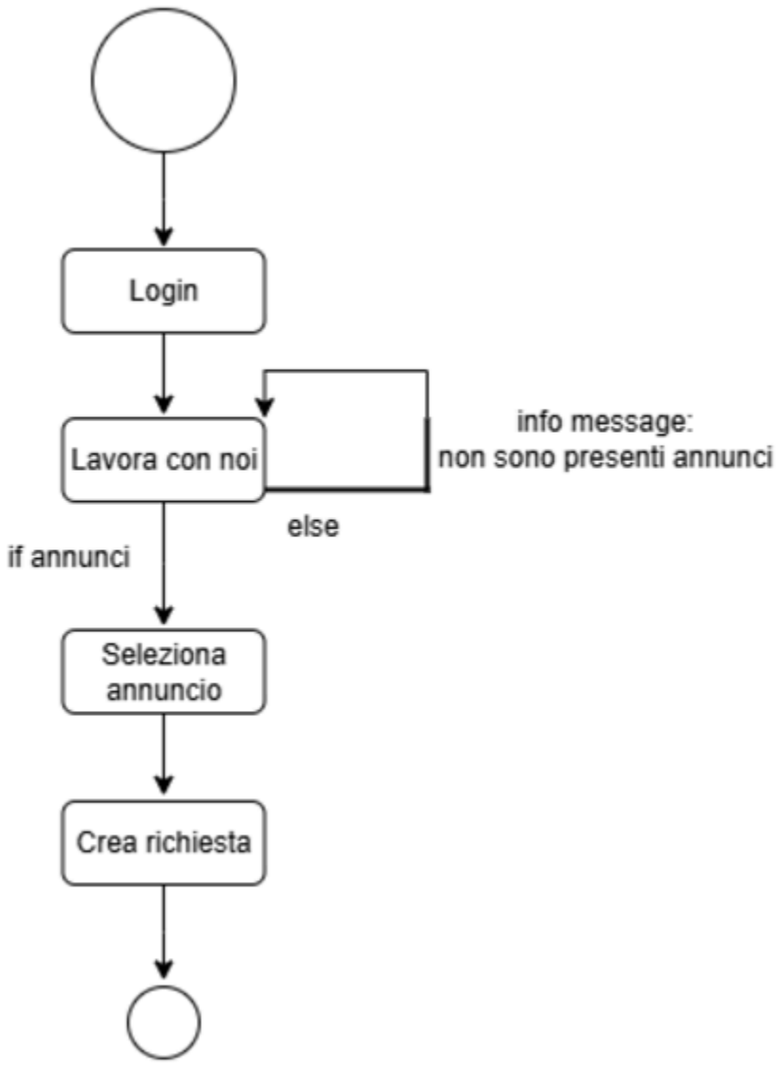

Figura 2: Activity diagram per l'inoltro di una richiesta lavorativa

Figura 4: activity diagram per la creazione di un ordine

3. Tecnologie utilizzate

La web-app è composta da tre applicazioni:

- gestione : dove si concentrano le funzionalità base dell'utente, come login e registrazione.
- ordini : dove vengono gestiti gli ordini effettuati dai clienti.

- ristoratore : dove si presentano le funzionalità dell'utente ristoratore.

Oltre alle sopracitate, vengono utilizzate le seguenti tecnologie:
- Bootstrap : per rendere l'interfaccia più curata e maggiormente navigabile.
- Pillow : utilizzata per l'aggiunta di immagini profilo da parte degli utenti.

4. Scelte progettuali

Di seguito vengono elencate le scelte progettuali relative al progetto, in modo da rendere l'applicazione più sicura e maggiormente navigabile da parte dell'utente:

- **Personalizzazione Dell'utente:** oltre al modello built-in di Django User, utilizzato per login e registrazione, è presente il modello Cliente che estende User tramite una relazione uno a uno, aggiungendo i campi nome, cognome, telefono e immagine.

- **Gruppi :** per poter rendere l'applicazione più sicura, vengono utilizzati due gruppi, che attraverso il decoratore personalizzato **allowed_user,** consentono l'accesso alle pagine solo ai presenti in un determinato gruppo:
  - **Customers :** tutti gli utenti che si registrano presso il sito vengono aggiunti a questo gruppo, in modo da poter usufruire di tutte le funzionalità messe a disposizione dal ristorante.
  - **Ristoratori :** inseriti da utenti con privilegi staff, possono eseguire azioni di gestione del ristorante, come modifica del menù e gestione di ordini, annunci e richieste lavorative.

- **Decoratori allowed_user e login_required :** implementano le logiche di sicurezza, utilizzati attraverso i gruppi sopracitati. Login_required (built-in Django) permette di accedere alle function views solo se il request user ha eseguito il login, mentre allowed_user permette l'accesso solo ad un determinato gruppo di appartenenza.

```
@login_required(login_url='login')
@allowed_users(allowed_roles=['Customers'])
def annulla_ordine(request):
    if request.user.is_authenticated:
        cliente = Cliente.objects.get(user = request.user)
        ordine = get_object_or_404(Ordine, cliente = cliente, status = 'nel carrello')
        ordine.delete()

    return redirect('menu')
```

Figura 5: function view per la quale è richiesto il login da parte dell'utente e la presenza nel gruppo Customers.

- **Messages framework e Script in JavaScript :** all'interno del progetto vengono utilizzati i messages per rendere più comprensibili le operazioni da parte dell'utente, come messaggi di errore o di operazione riuscita. Viene inoltre utilizzato uno script in

JavaScript collegato a messages che imposta un timer per nascondere i messaggi dopo 3 secondi. Lo script, inserito all'interno del template "script_messaggi.html", viene incluso in tutti gli altri template.

5. Testing

Il testing viene fatto tramite l'utilizzo della classe TestCase importata da Django.test, creando delle classi che ereditano da TestCase per ogni view o ogni model che si desidera testare. All'interno di ogni classe vengono definiti i metodi per testare le funzionalità e le viste dell'applicazione.
Sono stati effettuati i seguenti test:

- **Contenuto della homepage :** sono stati effettuati due test per verificare il contenuto della pagina "homepage", a seconda della tipologia di utente che effettua il login. Di seguito viene mostrato il test riguardante l'utente Cliente, dove una volta effettuato il login, viene controllata la presenza della stringa nella homepage. Se presente, verrà dato esito positivo.

```
def testVistaClienteHomepage(self):
    self.client.login(username = "prova", password = "qqqq1111")
    response = self.client.get(reverse('homepage'))
    self.assertEqual(response.status_code, HTTPStatus.OK)
    self.assertContains(response, "Ordini Conclusi:")
```

Figura 6: Metodo utilizzato per testare ciò che visualizza l'utente appartenente al gruppo Cliente.

- **Form per la creazione di un prodotto :** test effettuato per verificare la validità del form per la creazione di un prodotto da parte di un utente ristoratore. In questo caso viene passato al form il campo "prezzo" con valore negativo, per poi controllare la validità del form. Se il form non è valido, verrà dato esito positivo.

```
def testCreaProdotto(self):
    prezzo = -10
    form = AggiungiProdotto(prezzo)
    self.assertNotEqual(form.is_valid, True)
```

Figura 7: Metodo utilizzato per testare il form per la creazione di un prodotto da parte di un utente ristoratore

Per quanto riguarda i test effettuati sul contenuto della pagina "homepage", è stato utilizzato un metodo **SetUp** che permette di eseguire azioni prima di eseguire i test veri e propri. In questo caso, nel primo test riguardante il Cliente, nel metodo SetUp viene creato un utente e

il gruppo Customers a cui viene aggiunto. Stessa cosa accade per il secondo test riguardante l'utente ristoratore.

6. Conclusioni

L'applicazione è stata sviluppata rispettando tutti i requisiti presenti nella traccia inviata ai docenti. Il progetto, grazie al materiale presente all'interno del corso e alla documentazione on-line, non è mai stato in fase di stallo, se non in fasi leggermente critiche, come la risoluzione di determinati bug e logiche di migrazione, che hanno portato più volte alla completa ricostruzione del database.