

Davide Benassi - Mat. 165336

Ask Support

Esame di Tecnologie Web
Corso di Laurea in Informatica - UNIMORE

Introduzione

Descrizione Generale

AskSupport è una piattaforma realizzata con l'obiettivo di **facilitare** sia per gli **utenti** che per le **aziende** la **gestione** del **supporto tecnico** e delle domande che potenziali clienti possono sottoporre a queste ultime.

La piattaforma nello specifico raccoglie sia gli utenti “esterni” (registrati e non), che le aziende con i propri dipendenti, in un unico sito web al fine di **ridurre al minimo la frammentazione di questo tipo di servizio**.

Funzionalità

Lato utente esterno la piattaforma si mostra come una vetrina di aziende che mettono a disposizione delle FAQs (Frequently Asked Questions) per le domande più comuni oltre alla possibilità di aprire tickets di supporto per problemi e dubbi più specifici.

Lato azienda invece, la piattaforma offre la possibilità di supervisione e gestione da parte di un amministratore, e la possibilità di far interagire i propri dipendenti direttamente con gli utenti, rispondendo alle loro richieste di supporto.

Più nello specifico:

- **Visualizzazione Aziende:** nella HomePage della piattaforma vengono mostrate tutte le aziende registrate, con nome, descrizione e logo.
- **Consultazione e ricerca sezione FAQs:** ogni azienda ha una pagina dedicata, visitabile dagli utenti, nella quale vengono mostrate le FAQs che l'azienda stessa ha creato. Esse sono composte da una domanda ed una risposta, e sono cercabili per parole chiave sia nella domanda, che nella risposta.
- **Profilo Utente:** è implementato un meccanismo di registrazione, ed un conseguente meccanismo di login, alla piattaforma come Utente o come Azienda. Registrandosi come azienda verrà registrato anche un Utente Amministratore che potrà registrare a sua volta i dipendenti della propria azienda.
- **Meccanismo di Ticketing:** gli Utenti registrati alla piattaforma possono fare una richiesta di apertura di un Ticket con una azienda. Quest'ultimo verrà gestito da un

dipendente della stessa e, per entrambi gli utenti, è disposta una Dashboard che permette la gestione, il monitoraggio e l'interazione con i propri Ticket.

Il flusso seguito da un Ticket, dalla creazione alla chiusura è spiegato in maniera più esaustiva nella sezione dedicata.

- **Dashboard per Amministratore Azienda:** ogni amministratore di un'azienda ha a disposizione una dashboard che gli permette, oltre alla registrazione dei dipendenti sopra citata, di poter rimuovere questi ultimi e di monitorare lo stato dei vari ticket, visualizzati anche graficamente.

Struttura del Progetto

Questo progetto è stato realizzato utilizzando il Web Framework Django.

Di seguito si evidenziano le scelte effettuate in fase di progettazione che hanno rappresentato la base dell'intero progetto.

Utenti & Permessi

Come accennato tra le funzionalità del sito, sono presenti varie tipologie di utenti, con ruoli, compiti e conseguenti permessi differenti.

- **Admin:** Django mette a disposizione un utente amministratore con ruolo di "super-utente", il quale ha potenzialmente pieno controllo su tutti gli elementi del sito, specialmente relativi agli utenti.
Nel caso specifico della piattaforma realizzata l'utente admin non svolge un ruolo operativo ma è più che altro sfruttabile in caso di malfunzionamenti o correzioni relative agli utenti ed ai gruppi di utenti.
- **Utente Anonimo:** agli utenti anonimi è concesso visualizzare l'elenco di tutte le aziende presenti sulla piattaforma e, per ognuna di esse, consultare la sezione di FAQs con la possibilità di utilizzare la funzione di ricerca su queste ultime.

- **Utente Registrato (User):**
 - Profilo Personale - modifica dei dati del profilo, compreso il caricamento di un'immagine profilo, e l'eliminazione dello stesso.
 - Interazione con le Aziende - in aggiunta alle funzionalità di un utente anonimo può fare richiesta di apertura di un ticket e scambiare messaggi con i dipendenti tramite i ticket aperti.
- **Dipendente Azienda (Employee):** tramite la propria dashboard può interagire con le richieste di apertura di ticket, accettandole e/o rifiutandole, e con i ticket stessi scambiando messaggi con gli utenti. Ha il compito di chiudere un ticket fornendo una motivazione e può visualizzare i messaggi di ticket gestiti dai propri colleghi. Sempre tramite la dashboard può creare una FAQ che verrà sottoposta alla approvazione dell'amministratore prima di essere pubblicata.
- **Amministratore Azienda:**
 - tramite la propria dashboard può registrare nuovi dipendenti ed eventualmente eliminarli, oltre a monitorare lo stato dei ticket totali, ed il grafico che mostra le percentuali di ticket aperti, in attesa e chiusi rispetto a quelli totali;
 - tramite la sezione apposita può creare nuove FAQs, ed eventualmente eliminarle. Ha il compito inoltre di approvare, e/o rifiutare, le FAQs create dai propri dipendenti;
 - può modificare il profilo aziendale cambiandone il nome. la descrizione ed il logo.

Database

A tempo di progettazione sono state definite, e successivamente implementate a livello di codice, le seguenti entità:

- **User**: entità per un utente generico del sistema fornita da Django, utilizzato come “base” per implementare gli utenti registrati alla piattaforma, gli employee e gli amministratori delle aziende.
- **UserProfile**: utente registrato, aggiunge la foto profilo ad uno User.
- **Company**: rappresenta l’azienda con uno User che rappresenta l’amministratore, un nome, una descrizione ed un logo.
- **EmployeeProfile**: rappresenta un dipendente ed è costituito da uno User al quale è stata aggiunta l’azienda (Company) di appartenenza.
- **Ticket**: entità più complessa del sistema in quanto presenta un riferimento allo UserProfile che lo ha creato, alla Company alla quale appartiene ed all’Employee che lo gestisce, oltre che ad uno stato ed ai vari campi che lo descrivono.
- **Message**: entità che definisce un singolo messaggio scambiato in un ticket, identificato principalmente dal corpo del messaggio, dall’utente che lo ha inviato e dal momento in cui è stato inviato.
- **FAQ**: rappresenta una FAQ composta da domanda, risposta e riferimento alla Company di appartenenza.

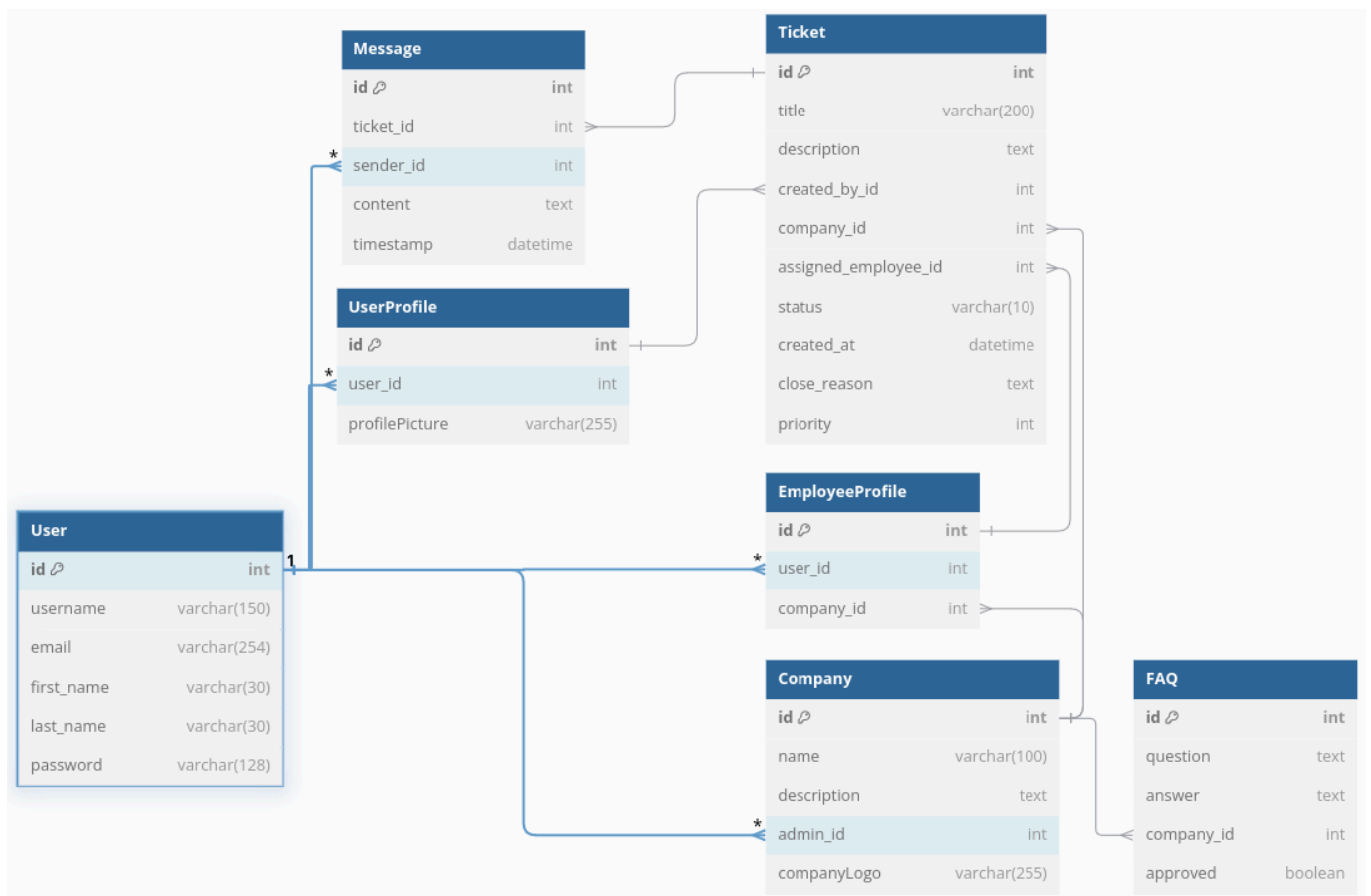
Per quanto riguarda le tipologie di relazioni tra le entità, Django integra e fornisce dei campi appositi per definire tali relazioni, quali:

- **ManyToManyField** (non utilizzato all’interno di questo specifico progetto): definisce una relazione di tipo molto-a-molti tra entità;
- **OneToOneField**: identifica una relazione di tipo uno-a-uno tra due entità. Nel progetto è principalmente utilizzata per “specializzare” uno User nei vari tipi di “Profili” possibili (UserProfile, EmployeeProfile e Company) aggiungendo i campi necessari;

- **ForeignKey:** permette di definire il fatto che un campo di una entità sia un “riferimento” ad un’altro tipo di entità.

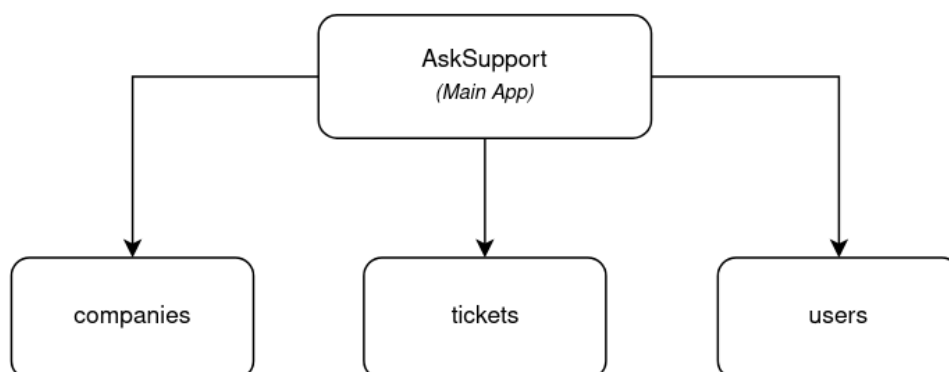
Di seguito è riportato graficamente lo schema del DataBase.

(*La tabella User è una rappresentazione dei dati essenziali, effettivamente utilizzati, derivanti dall’entità `django.contrib.auth.models.User`)



Suddivisione in App

Il progetto è stato suddiviso nelle seguenti “Django Apps” al fine di ottenere una migliore divisione logica delle sue varie componenti in vista di eventuali correzioni e/o estensioni future.



- **AskSupport:** directory di base del progetto, contiene i settings, le definizioni degli url principali ed una view di Login custom, pensata per redirigere i vari tipi di utenti su una pagina specifica una volta che la procedura di login è andata a buon fine.
- **Users:** app dedicata alla gestione degli User registrati alla piattaforma, compresa la loro pagina di gestione dei ticket e la personalizzazione del proprio profilo
- **Companies:** app in cui vengono gestite la maggior parte delle funzionalità dell'applicazione come le pagine delle aziende, i loro profili, la registrazione e la dashboard dei dipendenti (Employee), la dashboard dell'Amministratore dell'azienda e la sezione di FAQs.
- **Tickets:** app che contiene tutto ciò che riguarda i ticket nel loro singolo (non l'integrazione delle views delle dashboard ad esempio).
Questa app è stata logicamente divisa dalle altre in quanto i ticket costituiscono il punto di interazione tra gli User esterni e l'azienda e tutte le sue componenti ed utenti.

Tecnologie Utilizzate

Backend

L'intero progetto **AskSupport** è stato sviluppato utilizzando il framework web **Django**, il quale adotta il design pattern **MTV (Model-Template-View)**, derivazione della architettura **MVC (Model-View-Controller)**.

Questo modello permette una separazione chiara tra gestione dei dati (Model), logica applicativa (View) ed interfaccia utente (Template), il che rappresenta un grande vantaggio, sia in fase di progettazione che successivamente, durante lo sviluppo, in quanto fornisce delle linee guida robuste per strutturare l'intero progetto.

Il Framework si basa sul linguaggio **Python**, sia per la parte di logica applicativa che per la gestione dei dati nel database tramite l'**ORM (Object-Relational Mapping)** integrato. Questo strumento consente di interagire con il database in modo astratto senza scrivere direttamente query SQL.

Nello specifico, per questo progetto, il database utilizzato è **SQLite**, utilizzato di default da Django, in quanto offre una soluzione leggera per l'ambiente di sviluppo e test.

Frontend

Per quanto riguarda il frontend, Django fa uso di template scritti in linguaggio **HTML** nei quali può essere integrato il **DTL (Django Template Language)**.

Questo integra funzionalità aggiuntive all'interno delle pagine HTML per piccoli script (come cicli for o condizioni), e per l'interazione diretta con variabili definite a livello di logica applicativa.

Per la **personalizzazione** grafica delle pagine è stato invece utilizzato **Bootstrap**, un framework CSS che facilita la progettazione di pagine web grazie alla disponibilità di componenti predefiniti e classi di stile, riducendo il bisogno di scrivere codice CSS personalizzato.

Più nello specifico Bootstrap è stato utilizzato per fornire omogeneità a livello grafico alle pagine del sito e per poter disporre meglio gli elementi all'interno delle pagine stesse sfruttando layout a griglia.

Sono state infine integrate alcune funzionalità riguardanti l'interattività di alcune pagine tramite il linguaggio JavaScript.

Nello specifico:

- **AJAX (Asynchronous JavaScript and XML):** possibilità di inviare e ricevere dati dal server in modo asincrono, senza ricaricare l'intera pagina.
Questa funzionalità è stata sfruttata per il caricamento dei messaggi dei ticket, i quali altrimenti sarebbero dovuti essere precaricati all'interno della pagina, e non richiesti al server al momento dell'effettiva visualizzazione di tali messaggi, portando potenzialmente al caricamento di molti dati.
- **Chart.js:** libreria JavaScript utilizzata per renderizzare in maniera più immediata grafici basati sui dati del proprio sistema.
Utilizzato nella dashboard dell'Amministratore aziendale per visualizzare la distribuzione dei ticket in base al loro stato.

FAQs - Flusso Logico

Struttura

Una FAQ è costituita da una **domanda** e da una **risposta**. Inoltre, per avere il meccanismo di approvazione viene utilizzato un flag di stato.

Creazione

Una FAQ può essere creata dall'amministratore dell'azienda, ed in tal caso, nel momento della creazione, viene già "etichettata" come **approvata**.

The screenshot shows a web interface for creating and managing FAQs. At the top, it says 'FAQs for Tech Solutions S.r.l.' and 'Create a New FAQ'. There are two input fields: 'Question:' and 'Answer:'. Below these is a green 'Create FAQ' button. Underneath, there are two sections: 'Created FAQs' and 'Approved FAQs'. The 'Created FAQs' section has a sub-header 'FAQs to be approved' and lists two items: 'How does cloud computing work?' and 'What is the role of a firewall in a network?'. The 'Approved FAQs' section lists two items: 'What is the difference between RAM and storage?' and 'What is a VPN and why should I use it?'. Each item has a dropdown arrow on the right.

FAQs for Tech Solutions S.r.l.

Create a New FAQ

Question:

Answer:

Create FAQ

Created FAQs

FAQs to be approved

How does cloud computing work?

What is the role of a firewall in a network?

Approved FAQs

What is the difference between RAM and storage?

What is a VPN and why should I use it?

Se la FAQ è invece creata da un Employee viene “etichettata” come **non approvata** ed appare nella sezione dedicata nella dashboard dell’amministratore, il quale può approvarla o eliminarla.

FAQs for Tech Solutions S.r.l.

Create a New FAQ

Question:

Answer:

Create FAQ

Created FAQs

FAQs to be approved

How does cloud computing work?



Cloud computing allows users to access and store data on remote servers instead of local storage.

What is the role of a firewall in a network?



Approved FAQs

What is the difference between RAM and storage?



RAM is temporary memory used for running programs, while storage is permanent memory for saving files.

What is a VPN and why should I use it?



Pagina per creare le FAQ da parte di un dipendente

FAQs to be approved

How does cloud computing work?



Cloud computing allows users to access and store data on remote servers instead of local storage.

Delete FAQ

Approve FAQ

Approvazione di una FAQ da parte dell’amministratore

Visualizzazione per l'utente & Ricerca

Un utente, anonimo o registrato, può consultare la sezione delle FAQ di ogni azienda con la possibilità di utilizzare una barra di ricerca per cercare parole chiave nelle domande e nelle risposte.

Tech Solutions S.r.l.

A leading company in innovative tech solutions, providing cutting-edge software and hardware services.

Contact Email: alessandra.rossi@example.com

FAQ Session

Search

What is the difference between RAM and storage?



RAM is temporary memory used for running programs, while storage is permanent memory for saving files.

What is a VPN and why should I use it?



FAQ Session

Search

What is a VPN and why should I use it?



A VPN (Virtual Private Network) encrypts your internet connection, protecting your data from hackers and maintaining privacy.

Ticket - Flusso Logico

Struttura

Un ticket è strutturato principalmente da un **titolo**, una **descrizione**, ed una serie di **messaggi scambiati** tra l'utente creatore ed il dipendente che lo prende in carico.

Per definire se un ticket è aperto, chiuso o in attesa, è stato utilizzato un campo di stato.

Creazione

Un utente registrato alla piattaforma può fare richiesta di apertura di un ticket specificando titolo, descrizione e uno tra 4 livelli di priorità. Il ticket a quel punto viene creato per l'azienda in questione e messo in stato "pending".

Tutti i dipendenti dell'azienda, dalla loro dashboard vedono i "pending tickets", ordinati in base alla priorità decrescente, e possono accettarlo prendendolo in carico o rifiutarlo specificando una motivazione.

(* Il meccanismo della richiesta della motivazione al rifiuto viene riutilizzato nel momento della chiusura di un ticket da parte del dipendente gestore)

Open a Support Ticket

Title:
Ticket di Prova

Description:
La motivazione del ticket è la seguente

Priority:
High

Create Ticket

Utente crea il Ticket

Employee Dashboard

User: emp_ericci
Company: Tech Solutions S.r.l.

Open Tickets

Title	Description	Priority		
Ticket di Prova	La motivazione del ticket è la seguente	High	Chat	Close Ticket

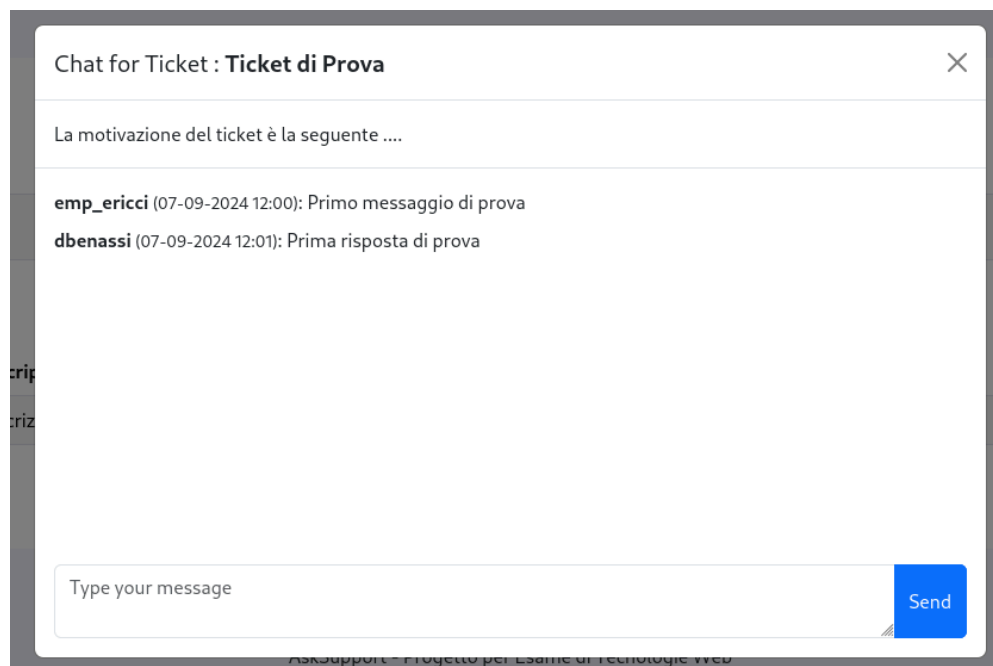
Pending Tickets

Title	Description	Priority	Created At		
Ticket 2	Descrizione di prova	Medium	Sept. 7, 2024, 9:57 a.m.	Accept	Reject

Dashboard dipendente con ticket accettati e pending

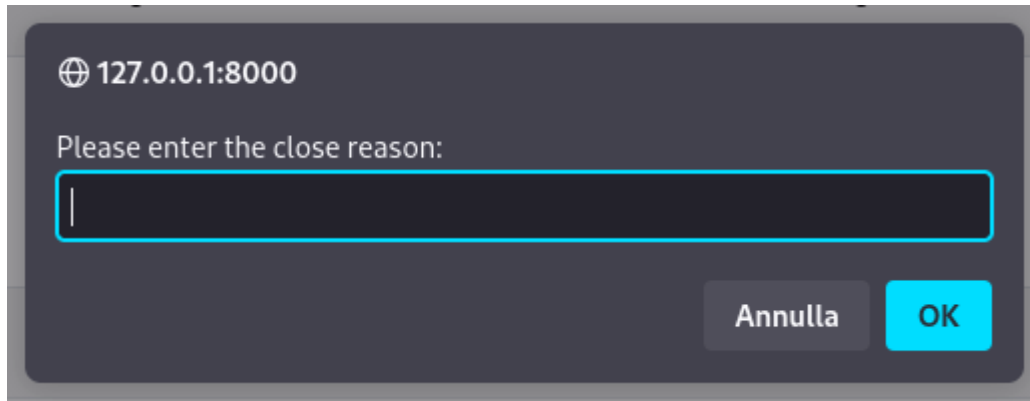
Scambio di messaggi

Utente e dipendente si possono scambiare messaggi finchè un ticket risulta aperto. Questo avviene tramite un oggetto grafico di tipo “Modal” che si apre e carica i messaggi successivamente alla pressione del bottone “Chat”



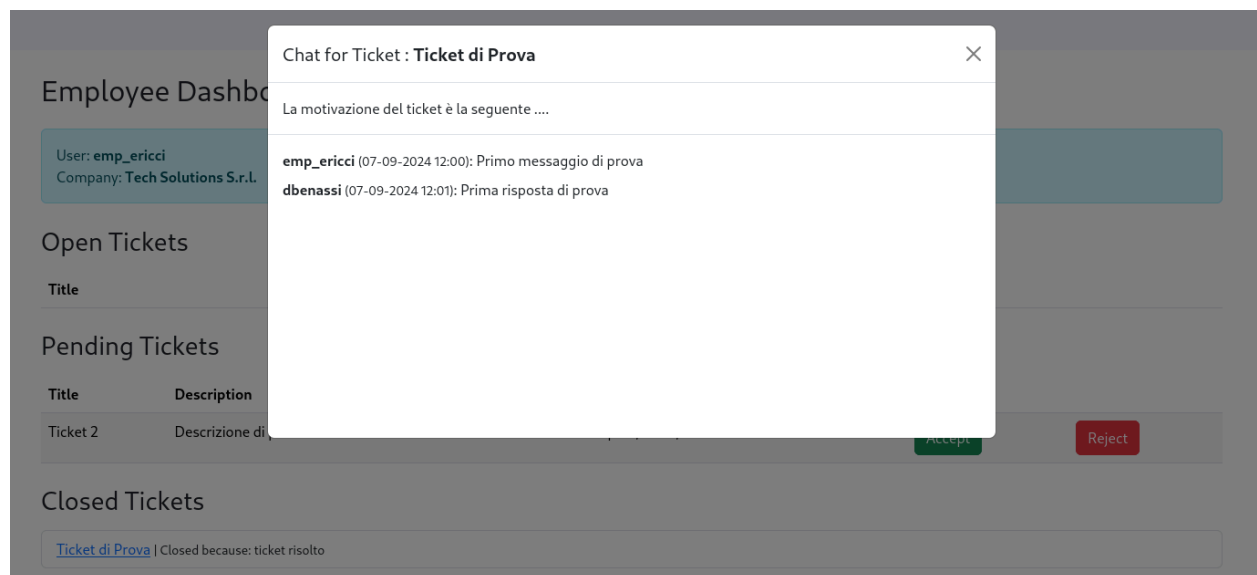
Chiusura & Ticket non gestiti

Come menzionato un ticket può essere chiuso sia rifiutandolo prima di aprirlo oppure dal dipendente gestore. In entrambi i casi viene chiesto di specificare una motivazione al fine di mostrare quest'ultima all'utente creatore del ticket.



A dark-themed modal dialog box with a title bar showing a globe icon and the IP address '127.0.0.1:8000'. The main text reads 'Please enter the close reason:'. Below this is a large, empty text input field with a light blue border. At the bottom right, there are two buttons: 'Annulla' (grey) and 'OK' (blue).

Inoltre, è implementata la visualizzazione, senza possibilità di scrittura, dei ticket chiusi (sia per i dipendenti che per gli utenti registrati) e dei ticket gestiti da altri dipendenti (per i dipendenti della stessa azienda).



The screenshot shows an 'Employee Dashboard' with a sidebar on the left containing sections for 'Open Tickets', 'Pending Tickets', and 'Closed Tickets'. The main area displays a chat window titled 'Chat for Ticket : Ticket di Prova'. The chat messages are:

- emp_ericci (07-09-2024 12:00): Primo messaggio di prova
- dbenassi (07-09-2024 12:01): Prima risposta di prova

Below the chat, there are 'Accept' and 'Reject' buttons. At the bottom of the dashboard, a status bar shows 'Ticket di Prova | Closed because: ticket risolto'.

Visualizzazione Grafico Ticket

Come precedentemente citato, l'amministratore di un'azienda ha una visione completa sulla situazione dei ticket in base al loro stato.

Questa funzionalità è prettamente di visualizzazione, non di interazione con i ticket aperti o in attesa.

Legend

Total Tickets:	2
Pending Tickets:	1
Open Tickets:	0
Closed Tickets:	1



Test

Al fine di verificare le funzionalità del proprio progetto, Django facilita la produzione di test attraverso alcune classi e metodi messi a disposizione dal framework stesso.

In questo caso sono stati realizzati dei test per verificare il corretto funzionamento del meccanismo di creazione, approvazione, lettura e ricerca delle FAQ; esse infatti coinvolgono i vari tipi di utenti registrati alla piattaforma, e perciò costituiscono un “target” valido per poter testare anche l’effettiva assegnazione dei corretti permessi.

Più nello specifico:

- Nella fase di inizializzazione dei test viene creata una Company, con il proprio Utente Amministratore e 3 FAQs di esempio, di cui una non approvata.

```
def setUp(self):
    # Creazione di un utente amministratore
    self.admin_user = User.objects.create_user(username='admin', password='adminpassword')
    admin_group = Group.objects.create(name='CompanyAdministrators')
    self.admin_user.groups.add(admin_group)

    # Creazione di un'azienda
    self.company = Company.objects.create(name='Test Company', description='A test company', admin=self.admin_user)

    # Creazione di alcune FAQ
    self.faq1 = FAQ.objects.create(question="What is this?", answer="This is a test FAQ.", company=self.company, approved=True)
    self.faq2 = FAQ.objects.create(question="How does it work?", answer="It works with magic.", company=self.company, approved=True)
    self.faq3 = FAQ.objects.create(question="Can I trust this?", answer="Of course!", company=self.company, approved=False)

    # Client per simulare le richieste
    self.client = Client()
```

- Successivamente sono stati prodotti alcuni test per verificare, ad esempio, il corretto rendering dell’elenco delle FAQs (la FAQ non approvata non deve essere renderizzata nella pagina visibile dall’utente finale, ed i meccanismi di approvazione ed eliminazione considerando i permessi dei vari tipi di utenti).

Di seguito si riporta qualche estratto dal file *companies/tests.py*

```
def test_faq_rendering(self):
    url = reverse('company-page', args=[self.company.id])
    response = self.client.get(url)
    (variable) status_code: int

    self.assertEqual(response.status_code, 200)
    self.assertTemplateUsed(response, 'company_page.html')
    self.assertContains(response, self.faq1.question)
    self.assertContains(response, self.faq2.answer)
    self.assertNotContains(response, self.faq3.question) # FAQ non approvata non dovrebbe essere visibile
```



```
def test_faq_approval_by_guest(self):
    # test approvazione FAQ da parte di un utente non registrato

    faq_to_approve = FAQ.objects.create(question="Is this pending?", answer="Yes.", company=self.company, approved=False)
    url = reverse('approve-faq', args=[faq_to_approve.id])

    response = self.client.post(url)

    faq_to_approve.refresh_from_db()
    # Controllo il redirect alla pagina di login
    self.assertRedirects(response, f'{settings.LOGIN_URL}&next={url}')

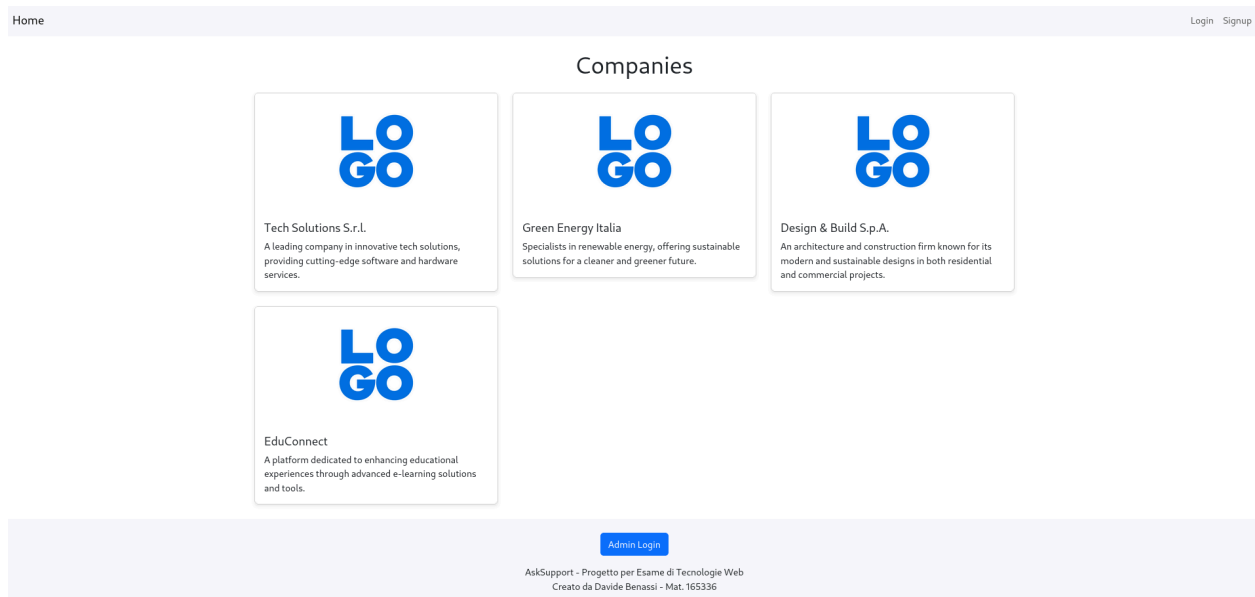
def test_faq_deletion_by_admin(self):
    # test eliminazione di una FAQ da parte dell'amministratore
    self.client.login(username='admin', password='adminpassword')

    url = reverse('delete-faq', args=[self.faq1.id])
    response = self.client.post(url)

    # verifica che la FAQ sia stata eliminata
    self.assertFalse(FAQ.objects.filter(id=self.faq1.id).exists())
```

Risultati & Visione d'insieme

La home page della piattaforma si presenta come segue:



In ogni pagina è integrata una **navbar** che aggiorna dinamicamente i propri elementi in base al tipo di utente loggato, permettendo di accedere alla pagina di personalizzazione del proprio profilo e/o alla propria dashboard per la gestione dei ticket, piuttosto che delle FAQs.

Oltre alla navbar è presente anche un **footer** che ha la funzione principale di mostrare informazioni generali e di avere un collegamento rapido all'accesso di Admin del sito web.

Personalizzazione del profilo

[Home](#) [dbenassi](#) [Tickets](#) [Logout](#)

Edit Profile

Update Profile Information

First name:


Last name:

Email address:

[Save Changes](#)

[Back to Profile](#)

Profile Picture




Profile Picture:

[Sfoglia...](#) Nessun file selezionato.

Ogni utente registrato, ed in maniera del tutto analoga l'amministratore di un'azienda, dispone di una pagina come quella mostrata per aggiornare i propri dati e per il caricamento delle immagini.

Eliminazione profilo

Utenti ed amministratori di aziende dispongono inoltre della possibilità di eliminare il profilo creato tramite un pulsante apposito, mostrato di seguito, che porterà l'utente ad una pagina di richiesta di conferma tramite password.



Davide Benassi

Username: dbenassi

Email: davide.benassi@example.com

[Edit Profile](#)

[Change Password](#)

[Delete Profile](#)

Confirm Account Deletion

Please enter your password to confirm deletion of your account. This action is irreversible.

- This field is required.

Password:

Enter your password to confirm deletion.

[Delete My Account](#)

Conclusioni

La piattaforma attualmente integra la maggior parte, se non tutte, le funzionalità necessarie per fornire supporto agli utenti tramite chat personali e per integrare più aziende in uno stesso “luogo” evitando che ognuna di esse debba avere il proprio personale servizio di ticketing.

Il progetto può sicuramente essere esteso aumentando, ad esempio, il livello di personalizzazione dei profili o aggiungendo delle funzionalità di gestione dei tempi (scadenze e priorità) e di notifica ad esempio tramite indirizzi mail.

La principale difficoltà riscontrata nella realizzazione del progetto è sicuramente legata ad una lenta curva di apprendimento del framework, il quale mette a disposizione una grandissima quantità di funzioni ed integrazioni, spesso già realizzate dagli sviluppatori stessi, ma che portano ad un primo approccio ad esso tendenzialmente non troppo immediato.

In conclusione, il progetto dispone di varie funzionalità legate alla comunicazione ed alla interazione tra le varie tipologie di utenti della piattaforma, lasciando lo spazio per future implementazioni.