Documentazione

Mensa Aziendale

Progetto e sviluppo interfacce utente

Progetto di Verlanti Emanuele

Sommario

INTRODUZIONE

Mensa Aziendale e’ un portale multiutente che utilizza il framework Django per la gestione di una mensa aziendale.

Questa applicazione web presenta un design semplice ed elegante e mette a disposizione all’utente un set di strumenti per garantire un’esperienza di navigazione funzionale e intuitiva.

L’ utente ha la possibilità di visitare Mensa Aziendale sia come *user anonimo* e quindi di vedere il menu e raccogliere tutte le informazioni utili, sia come user registrato, il quale e’ in grado di sfruttare a pieno ritmo la forza di Mensa Aziendale e far uso di un sacco di features all’interno del sito come poter ordinare, vedere come si preparano i piatti e altro ancora…

Mensa Aziendale non si rivolge solo al cliente (I*mpiegato*) ma grazie alla differenziazione utente, e’ possibile loggarsi anche come utente speciale (C*uoco*), il quale integra la lista degli ordini settimanali, la gestione completa del menu, l’inserimento e gestione di nuove creazioni e molto altro.

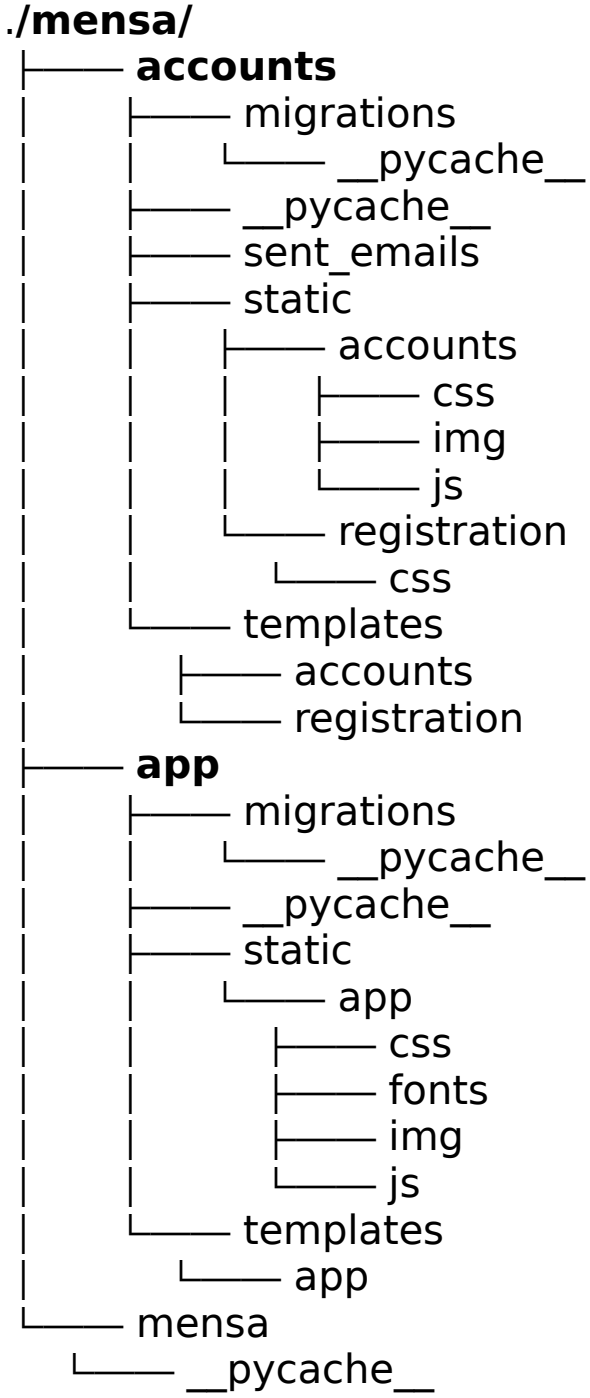
Una ulteriore possibilità e’ quella di utilizzare Mensa Aziendale con la modalità *Azienda*, la quale mette a disposizione varie views per avere il controllo sui costi di ogni dipendente sia giornaliero che settimanale e per avere una prima analisi sulle scelte dei piatti effettuate.

Mensa Aziendale e’ caratterizzato da uno stile unico e essenziale grazie all’utilizzo dei più’ famosi linguaggi di markup, styling e di scripting come html, css e javascript, inoltre utilizza anche alcune librerie esterne dei linguaggi stessi come Bootstrap e jQuery.

STRUTTURA GENERALE

Mensa Aziendale presenta la seguente struttura generale:

29 directories



Come si puo’ notare Mensa Aziendale e’ composto da ulteriori due sotto-applicazioni: app e accounts

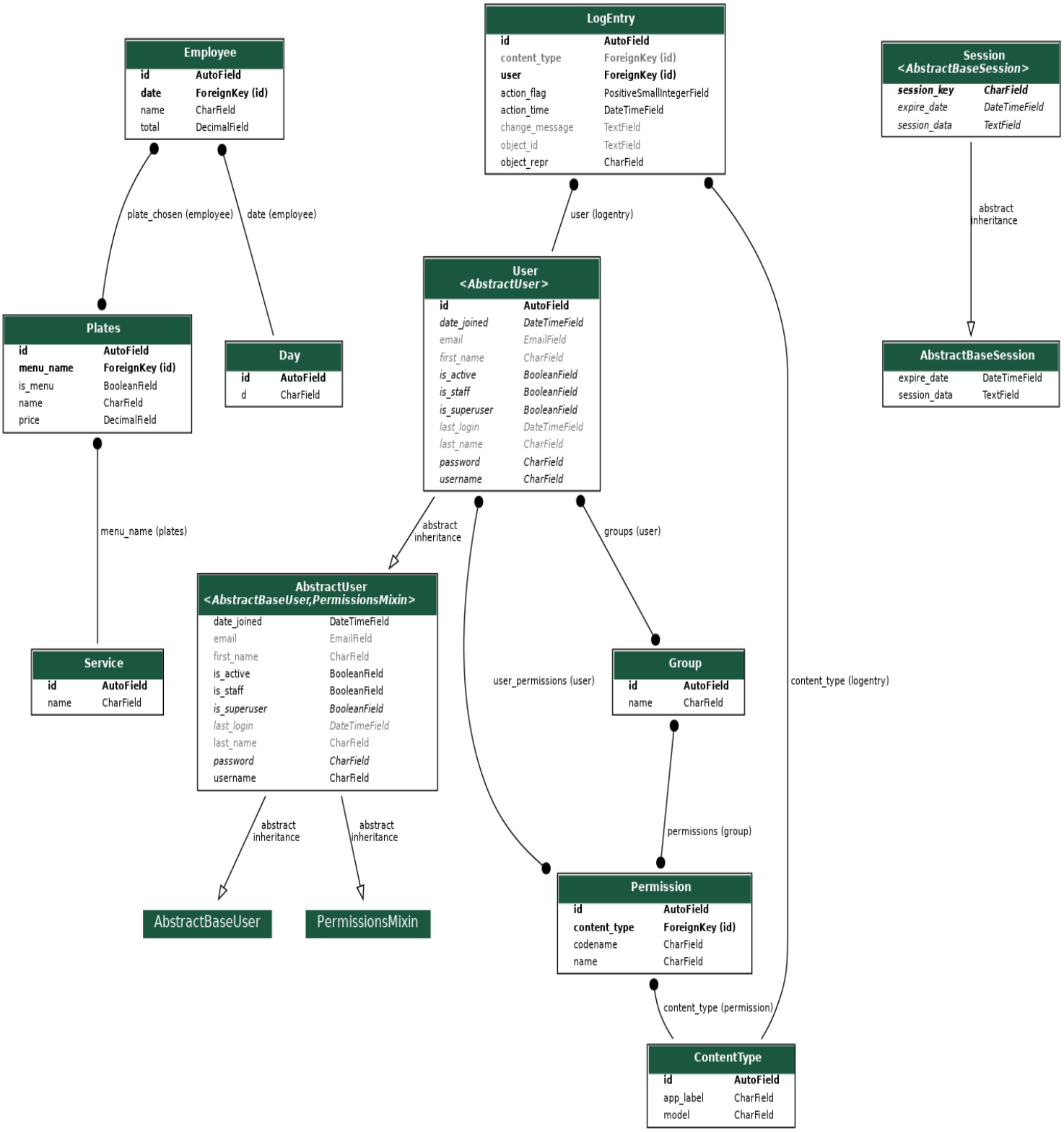
La prima ha il compito di gestire tutta quella parte che si occupa della parte di front-end e della manipolazione dei dati.

La seconda invece e’ specializzata nella gestione della parte multiutente ovvero si occupa della registrazione, login e reset password di ogni singolo utente.

La tipologia di DBMS scelta per Mensa Aziendale e’ sqlite3. La scelta di questo DMBS sta nella semplicità d’uso e di gestione e dell’integrazione di default con il framework Django.

Mensa Aziendale fa uso di un database memorizzato nel file db.sqlite3 , le cui tabelle (models nel linguaggio Django) verranno spiegate in modo dettagliato più avanti.

La seguente immagine mostra la struttura interna del database:



Welcome cuoco01_

*homepage cuoco01_*

Home

Menu

Ordina!

Reset password

Logout

menu di navigazione

Top 3

1. Branzino al forno 18.75%

2. Carbonara 12.5%

3. Tortelli verdi 12.5%

top 3 in statistiche

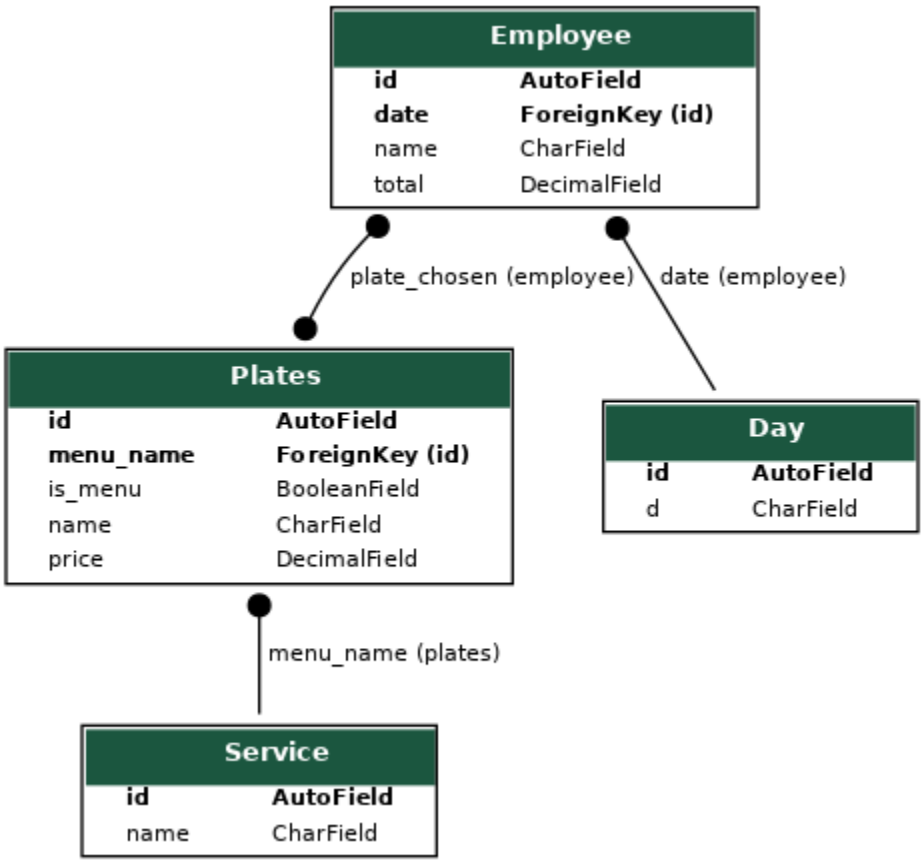
APP

App e’ una sotto-applicazione Django che permette a Mensa Aziendale di gestire tutta quella parte di front-end del sito web.

Questa applicazione fa uso di un singolo database per la memorizzazione dei dati.

Il database in uso presenta i seguenti modelli, tutti descritti nel file *.models*:

- Plates : registra le caratteristiche di un piatto.
- Service : registra le portate principale, ammette solo quattro valori (Primo,Secondo,Contorno,Dessert).
- Employee: registra tutti i dipendenti dell’azienda e le loro scelte.
- Day: registra i giorni della settimana.



Scelte implementative

In questa parte di documentazione sono presenti tutte le scelte implementative fatte.

app/models.py

L'APP utilizza al fine di limitare la tipologia di alcuni *fields* la classe Enumeratore.

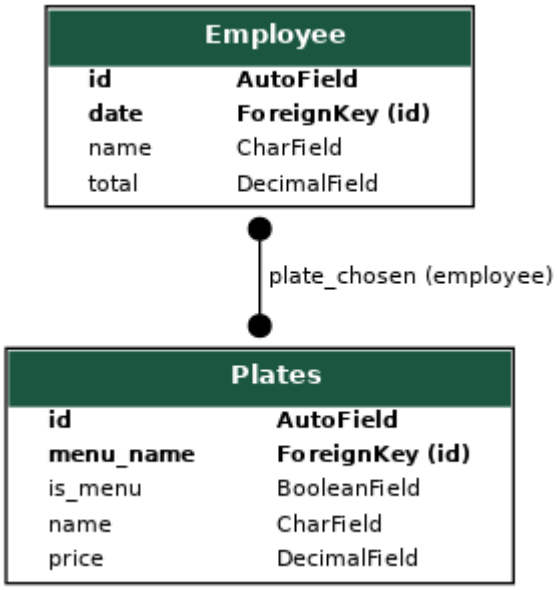
Fa uso dell' enumeratore *MenuChoice* per limitare la tipologia del field *name* nel model *Service*.

```
8
9
10  # Create your models here.
11  class MenuChoice(Enum):
12      """Scelte possibili per i nomi del Menu"""
13      PR = "Primo"
14      SC = "Secondo"
15      CN = "Contorno"
16      DS = "Dessert"
```

Fa uso dell'enumeratore WeekDay per limitare il fields d nel model Day.

```
37
38  class WeekDay(Enum):
39      """Giorni della settimana"""
40      MON = "Monday"
41      TUE = "Tuesday"
42      WED = "Wednesday"
43      THU = "Thursday"
44      FRI = "Friday"
45      SAT = "Saturday"
46      SUN = "Sunday"
```

Come da requisito nella struttura dell’App e’ presente anche una relazione Molti a molti gestita con l’uso di un field particolare messo a disposizione da Django chiamato ManytoMany field.




```python

#Ordine dell user
class Employee(models.Model):

    name = models.CharField(max_length=30)
    date = models.ForeignKey('Day',models.CASCADE)
    total = models.DecimalField(max_digits=8, decimal_places=2,default=Decimal('0.00'))
    plate_chosen = models.ManyToManyField('Plates')

```

```python
class Plates(models.Model):

    name = models.CharField(max_length=30)
    price = models.DecimalField(max_digits=8, decimal_places=2)
    is_menu = models.BooleanField(default=False)
    menu_name = models.ForeignKey('Service',models.CASCADE) # related_name='menu')

    def __str__(self):
        return self.name

```

app/views.py

In questo file sono presenti tutte le views che si collegano poi ai templates html.
Questo file mette a disposizione tutte le funzioni per la gestione dei piatti (aggiornamento, inserimento e cancellazione), gestione del Menu (cancellazione e inserimento), calcolo delle statistiche e gestione di un ordine.

In questo file sono anche implementate le funzioni per il calcolo dei costi giornalieri/settimanali e le varie scelte dei piatti giornalieri/settimanali.

```python
def daily_chose(dict_plate_scelte,plate,day):
    total_weekly = 0
    total = 0
    for d in day:
        for p in dict_plate_scelte.keys():
            for emp in Employee.objects.filter(plate_chosen=p,date=d):
                total += 1
            dict_plate_scelte[p].append(total)
            total = 0
    for p in dict_plate_scelte.keys():
        for w in dict_plate_scelte[p]:
            total_weekly += w
        dict_plate_scelte[p].append(total_weekly)
        total_weekly= 0
    #print(dict_plate_scelte)
    return dict_plate_scelte

```

```python
def daily_cost(dict_spese,user,day):
    total_weekly = 0
    total = 0
    for d in day:
        for u in dict_spese.keys():
            for emp in Employee.objects.filter(name=u.username,date=d):
                #print(u.username,d)
                #print(emp.name,emp.date,emp.total)
                total += emp.total
            dict_spese[u].append(total)
                #print(emp.name,emp.date,emp.total)
                total += emp.total
            dict_spese[u].append(total)
        for w in dict_spese[u]:
            total_weekly += w
        dict_spese[u].append(total_weekly)
        for w in dict_spese[u]:
            total_weekly += w
        dict_spese[u].append(total_weekly)
```

app/form.py

In app/form.py sono presenti tutti form presenti in Mensa Aziendale .

In questo file e’ presente una implementazione interessante infatti l’user nel gruppo impiegato può effettuare un ordine solo nei giorni successivi al giorno dell’ordine (sempre limitatamente alla settimana).

```
35
36  class OrdineForm(ModelForm):
37      date = forms.ModelChoiceField(queryset=Day.objects.filter(id__gte =
38                           weekdays(calendar.day_name[date.today().weekday()])))
39      plate_chosen = forms.ModelMultipleChoiceField(
40                          widget = forms.CheckboxSelectMultiple,
41                          queryset = Plates.objects.filter(is_menu=True))
42
43      class Meta:
44          model = Employee
45          fields = ["date","plate_chosen"]
46
```

ACCOUNTS

Accounts e’ una sotto-applicazione Django che permette a Mensa Aziendale di gestire tutta quella parte di amministrazione del sito web.

Infatti Accounts si occupa della creazione dell’utente e del suo smistamento nel gruppo prescelto (Cuoco, Impiegato, Azienda), reset della password dell’utente attraverso l’utilizzo della email.

Al fine di testare e provare il reset della password e’ stata implementata una cartella (sent_email) che simula l’invio di una email di conferma come la seguente :

```
Content-Type: text/plain; charset="utf-8"
MIME-Version: 1.0
Content-Transfer-Encoding: 7bit
Subject: Password reset on 127.0.0.1:8000
From: webmaster@localhost
To: admin@example.com
Date: Tue, 23 Apr 2019 12:03:38 -0000
Message-ID: <155602101874.4528.3804836940040502015@emanuele-pc>


You're receiving this email because you requested a password reset for your user accounts at 127.0.0.1

Please go to the following page and choose a new password:

http://127.0.0.1:8000/accounts/reset/Mg/55q-9e0a67a5c0f855ad0b69/

Your username, in case you've forgotten: admin

Thanks for using our site!


Thanks for using our site!

The 127.0.0.1:8000 team


```

File uses the Plain Text grammar

File uses the Plain Text grammar

Scelte implementative

Il task multiutente viene gestito in base al concetto di gruppo. Infatti un user puo' appartenere solo ad uno dei seguenti gruppi:

- Impiegato : fanno parte tutti gli users che vogliono sfogliare il menu e eseguire un ordine.

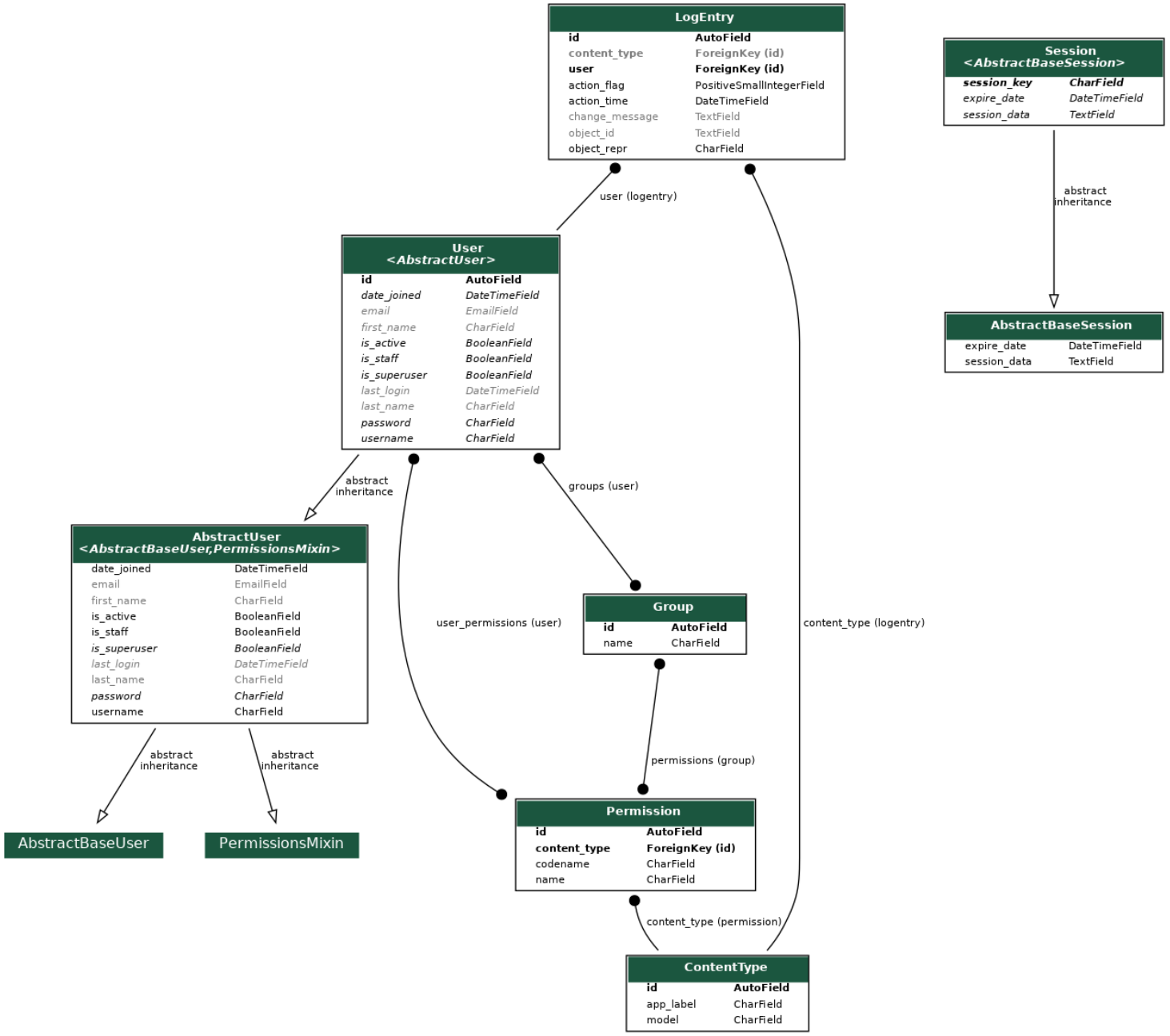
- Cuoco : tutti gli users di questo gruppo sono in grado di gestire il menu, vedere gli ordini settimanali e vedere i piatti nel database.

- Azienda: tutti gli users di questo gruppo sono in grado di gestire i dati relativi ai costi e i piatti scelti.

Ogni gruppo possiede permessi differenti.

accounts/views.py

In questo file e' presente solo la view della registrazione di un utente (Signup) perche' e' stata presa la decisione implementativa di far gestire la parte di login direttamente a Django, di conseguenza sono stati implementati solo i templates.


LogEntry
id AutoField
content_type ForeignKey (id)
user ForeignKey (id)
action_flag PositiveSmallIntegerField
action_time DateTimeField
change_message TextField
object_id TextField
object_repr CharField
Session
<AbstractBaseSession>
session_key CharField
expire_date DateTimeField
session_data TextField
abstract inheritance
AbstractBaseSession
expire_date DateTimeField
session_data TextField
user (logentry)
User
<AbstractUser>
id AutoField
date_joined DateTimeField
email EmailField
first_name CharField
is_active BooleanField
is_staff BooleanField
is_superuser BooleanField
last_login DateTimeField
last_name CharField
password CharField
username CharField
abstract inheritance
groups (user)
AbstractUser
<AbstractBaseUser,PermissionsMixin>
date_joined DateTimeField
email EmailField
first_name CharField
is_active BooleanField
is_staff BooleanField
is_superuser BooleanField
last_login DateTimeField
last_name CharField
password CharField
username CharField
user_permissions (user)
Group
id AutoField
name CharField
content_type (logentry)
abstract inheritance
abstract inheritance
permissions (group)
AbstractBaseUser
PermissionsMixin
Permission
id AutoField
content_type ForeignKey (id)
codename CharField
name CharField
content_type (permission)
ContentType
id AutoField
app_label CharField
model CharField

TESTING

Come da requisito Mensa Aziendale prevede una fase di testing.

Per questa fase viene utilizzata la libreria django *django.test.*

Il model scelto per la fase di testing e' il model Plates.
Sono stati effettuati test su tutti gli input possibili per questa classe.

```
6
7  class PlatesTestCase(TestCase):
8      # Test sugli input del modello Plates
9
10     # Test sul prezzo del piatto
11     def test_price_negative(self):
12         plate = Plates(50,"piatto_fake",4.20,False,0)
13         self.assertGreaterEqual(plate.price, 0,"Il prezzo non puo' essere negativo")
14
15     def test_format_price(self):
16         plate = Plates(50,"piatto_fake",443.25,False,0)
17         self.assertAlmostEqual(plate.price,round(plate.price,2))
18         self.assertLessEqual(plate.price,999999.99,"Prezzo troppo grande")
19
```

(Screenshot della classe parziale per eseguire il testing)

Per una visione piu' approfondita delle funzioni consultare il file tests.py nella directory app.

Oltre il controllo del *models* Plates, Mensa Aziendale prevede anche una fase di testing per le *views* piu' importanti e significative.

Di seguito e' presente uno screenshot che mostra alcune delle classi create per questa fase.

```python
class StatisticheViewTests(TestCase):
    def test_view_statistics(self):
        """
        Test sulla view app:statistics
        """
        response = self.client.get(reverse('app:statistics'))
        self.assertEqual(response.status_code, 200)

class MenuTestView(TestCase):
    def test_view_menu(self):
        """
        Test sulla view app:menu
        """
        response = self.client.get(reverse('app:menu'))
        self.assertEqual(response.status_code, 200)

```

(Screenshot delle classi che testano le views)

JAVASCRIPT

All’ interno di Mensa Aziendale sono presenti molteplici controlli e funzioni veramente interessanti nel linguaggio di scripting Javascript.

Di seguito il codice di alcune di queste funzioni implementate.

Funzione checkprice():

```
function checkprice() {
      var price =  document.getElementById("id_price").value;

      if(price<0 ){
          alert("Errore: Il prezzo non puo' essere negativo");
          return false;
      }

      alert("Piatto inserito correttamente");
      return true;
   }

```

Questa funzione controlla se il prezzo inserito non e’ un valore negativo.

Funzione updateCount()

```
$(document).ready(function(){
  $('#id_password1').keyup(updateCount);
  $('#id_password1').keydown(updateCount);

  $('#id_password2').keyup(updateCount2);
  $('#id_password2').keydown(updateCount2);


  function updateCount() {
        var cs = $(this).val().length;
        $('#characters').text(cs);
    }

    function updateCount2() {
          var cs = $(this).val().length;
          $('#characters2').text(cs);
      }

});

```

Questa funzione permette all’utente di conoscere quanti caratteri ha inserito ogni volta che inserisce una password.

Funzione checkpassword()

```
function checkpassword() {
    var valid = true;
    var pass1 =  document.getElementById("id_password1").value;
    var len_pass1 = $('#id_password1').val().length;
    var pass2 = document.getElementById("id_password2").value;
    var pass1 =  document.getElementById("id_password1").value;
    var len_pass1 = $('#id_password1').val().length;
    var pass2 = document.getElementById("id_password2").value;
        alert("Errore: Le due password non coincidono");
        valid = false;
    }
    if(len_pass1 < 8 ){

      //document.getElementById("helptext").innerHTML = "La tua password deve contenere almeno 8 caratter
      alert("Errore: La tua password deve contenere almeno 8 caratteri.");
      valid=false;
    }
    if (valid == false){
      return false;
    }
  //  document.getElementById("helptext").innerHTML = "";
    return true;
}
```

Questa funzione effettua alcuni importantissimi controlli sull’ inserimento della password per garantire una registrazione sicura all’utente.

Sono presenti ulteriori controlli e funzioni interessanti che possono essere approfonditi e visionati all’interno del codice di Mensa Aziendale.

LIBRERIE ESTERNE

Sono state utilizzate alcune librerie esterne (Django, python, html, css, javascript) per lo sviluppo e il testing di Mensa Aziendale .

Di seguito sono presenti i link.

Django-extensions

https://django-extensions.readthedocs.io/en/latest/

Coverage (per il testing)

https://coverage.readthedocs.io/en/latest/

Bootstrap

https://getbootstrap.com/

jQuery

https://jquery.com/

BIBLIOGRAFIA

I seguenti siti sono stati utilizzati per la creazione di Mensa Aziendale :

https://stackoverflow.com/

https://www.google.com/

https://www.w3schools.com/

https://www.djangoproject.com/

https://wsvincent.com/django-user-authentication-tutorial-signup/

https://readthedocs.org/projects/django/

https://realpython.com/testing-in-django-part-1-best-practices-and-examples/