

Index

| | |
|---|----|
| Descrizione del progetto..... | 3 |
| Visualized Design of the System..... | 5 |
| Glossario..... | 6 |
| Database e Modelli..... | 7 |
| Diagramma stile UML dei modelli del Database..... | 9 |
| Tecnologie utilizzate..... | 10 |
| Organizzazione logica di applicazioni..... | 11 |
| Problemi riscontrati durante progettazione..... | 12 |
| [Problem 1] Universal Recommendation System..... | 12 |
| [Problem 2] User Specific Recommendation System..... | 13 |
| [Problem 3] Advanced Product Searching (Complex lookups with Q object)..... | 14 |
| [Problem 4] Application Form Management System..... | 15 |
| [Problem 5] Editing of Seller's Products and Creating New Products..... | 16 |
| [Problem 6] Personal Profile..... | 17 |
| Unit Testing..... | 18 |
| 1. Cart access/deleting system check..... | 18 |
| 2. Seller apply form..... | 18 |
| 3. Checking constraints of the Product model..... | 19 |
| 4. User creation form..... | 19 |
| 5. Access to the product creation page..... | 20 |
| 6. Access to the management page when user is not staff..... | 20 |
| Collection of screenshots from different pages..... | 21 |

Descrizione del progetto

Categoria del sito: E-commerce

Si suppone di dover realizzare un sito di e-commerce (stile Libraccio, AbeBooks, IBS, ...) per la vendita di libri nuovi ed usati. In particolare, il sito avrà come il compito principale di mettere in contatto persone in ricerca di libri. Per fornire il miglior servizio avremmo necessità di gestire le seguenti informazioni:

UTENTI

Gli utenti anonimi possono navigare il sito, fare le ricerche per venditori/libri e visualizzare i prodotti disponibili sul sito. Ogni utente anonimo può registrarsi in seguito come utente normale.

Gli utenti normali possono compilare nel proprio profilo personale un modulo di domanda per ammissione nel gruppo di venditori approvati del sito. Gli utenti che hanno compilato il modulo di domanda per ammissione possono vedere lo stato di progressione nel suo profilo personale. Lo stato di progressione può essere di seguenti tipi:

- in attesa di approvazione
- approvato (in seguito l'utente diventa ufficialmente un venditore approvato e può cominciare a vendere prodotti)
- respinto

Gli utenti venditori approvati nel suo profilo personale avranno la possibilità di creare nuovi prodotti compilando un form e visualizzare/modificare propri prodotti creati.

PROFILO PERSONALE

Ogni utente registrato dispone del proprio profilo personale compilato durante la fase di registrazione. Ogni utente registrato nel suo profilo avrà la possibilità di cambiare il suo nome, il cognome, l'email, la foto di profilo e la sua password.

PERSONALE

Il personale interno è in grado modificare, approvare/respingere lo stato di progressione delle domande di ammissione dei futuri venditori.

Il personale interno è in grado di creare/modificare nuove sezioni per prodotti (ad esempio una nuova sezione dedicata ai libri di educazione) disponibili per la scelta nella fase di creazione di prodotti e per la ricerca.

Il personale interno è in grado di creare prodotti a sua volta come venditori approvati. Il personale interno è in grado di apportare delle modifiche ai prodotti esistenti nel sistema.

CATALOGO VENDITORI APPROVATI

Il sito dispone di un catalogo di venditori approvati. È possibile fare la ricerca per nome di venditore, paese, telefono o email.

PRODOTTI

Ogni prodotto creato avrà le seguenti caratteristiche: Nome del libro, l'autore, l'editore, l'anno di pubblicazione, categoria, immagine, ed una descrizione.

Gli utenti possono visualizzare prodotti e fare le ricerche per prodotti e visualizzare la pagina del venditore del prodotto selezionato. Gli utenti possono aggiungere prodotti nel proprio carrello ed eventualmente procedere con pagamento.

USER SPECIFIC RECOMMENDATION SYSTEM

Il sito dispone di un User Specific Recommendation System in grado di suggerire prodotti simili agli utenti registrati in base alla loro storia di acquisto.

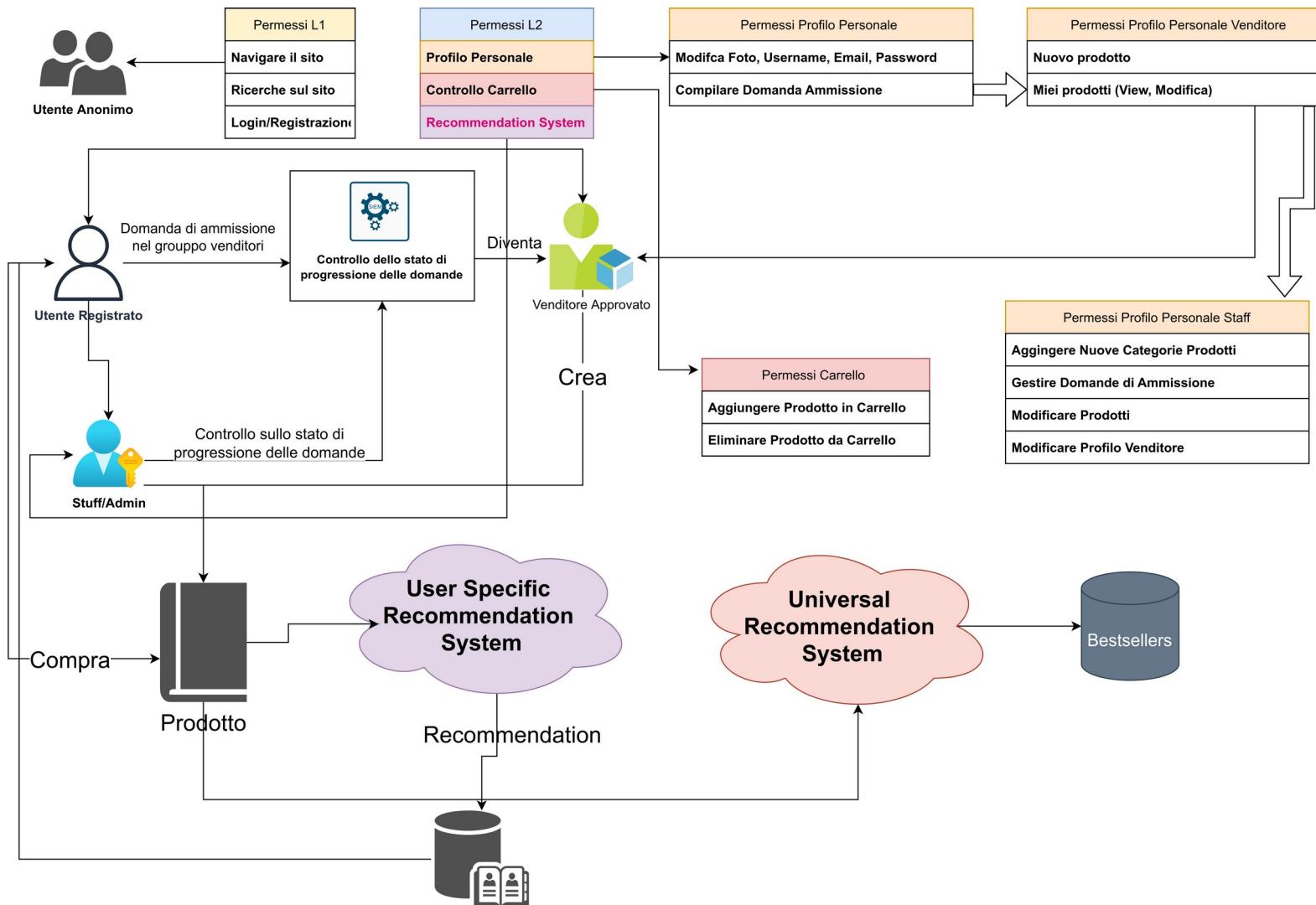
BESTSELLERS

Il sito dispone di Universal Recommendation System in grado di selezionare prodotti più acquistati dagli utenti e successivamente suggerirli agli utenti.

CARRELLO

Gli utenti registrati avranno la possibilità di aggiungere prodotti nel proprio carrello. Gli utenti registrati avranno la possibilità di eliminare prodotti dal proprio carrello.

Visualized Design of the System



Glossario

| <u>Nome</u> | <u>Sinonimi</u> | <u>Legame</u> |
|-------------------------------------|--|---|
| Utente Anonimo | | |
| Utente Normale | Normal User | Profilo |
| Utente Venditore | Seller, Approved Seller, Certified Seller | Profilo |
| Profilo | Profile | |
| Bestsellers | Top 4 Recommendation | Utente |
| User Specific Recommendation System | Recommended to You | Utente Normale, Utente Venditore |
| Prodotto | Libro, Product | Utente Venditore, Staff |
| Staff | Admin, Moderator | Utente Venditore, Panello di Ammissione |
| Panello di Ammissione | Application Form Management System, Panello di Controllo delle domande di ammissione | Staff, Utente Venditore |
| Carrello | Cart, My Cart | Utente Normale/Venditore/Staff |
| Catalogo Venditori | Catalogo Venditori Approvati | Utente Venditore |
| Venditore Staff | Certified | Prodotto, Staff |

Database e Modelli

È stato deciso di utilizzare un database relazionale lightweight come SQLite supportato di default da Django.

User Model

Come modello base per gli utenti è deciso di adottare modello base **User Model** di Django.

UProfile Model

Questo modello è di supporto al modello base di **User Model** per espandere funzionalità e permettere agli utenti di caricare immagini di profilo senza fare overriding del modello di default di Django.

```
class UProfile(models.Model):
    user = models.OneToOneField(User, on_delete=models.CASCADE, primary_key=True)
    profile_image = models.ImageField(
        upload_to='profileImages/',
        blank=True,
        max_length=40,
    )

    def __str__(self):
        return f'Profile of {self.user.username}'

    class Meta:
        verbose_name_plural = 'UProfiles'
        verbose_name = 'UProfile'
```

Seller Model

Questo modello è alla base di utenti venditori per differenziare il comportamento dal modello **User Model**.

```
class Seller(models.Model):
    created_by = models.OneToOneField(User, related_name='created_by', on_delete=models.CASCADE, primary_key=True)
    creation_date = models.DateTimeField(default=timezone.now)

    OPTIONS = (
        ('0', 'pending'),
        ('1', 'approved'),
        ('2', 'rejected'),
        ('3', 'undefined'),
    )
    status = models.CharField(default='0', choices=OPTIONS, max_length=20, blank=False)
    approved_by = models.ForeignKey(User, related_name='approved_by', on_delete=models.CASCADE, blank=True, null=True)
    seller_name = models.CharField(max_length=255, blank=False)
    country = CountryField()
    legal_seller_address = models.CharField(max_length=255, blank=False)
    contact_phone = models.PositiveIntegerField(blank=False)
    seller_email = models.EmailField(max_length=50, blank=False)
    description = models.TextField(max_length=300, blank=False)
    class Meta:
        verbose_name_plural = 'Sellers'
        verbose_name = 'Seller'
        ordering = ('-creation_date',)
```

Category Model

Questo modello è di supporto al modello **Product Model** per non inglobare troppi fields all'interno dello stesso relazione (tabella) e per differenziare i compiti meglio gestibili separatamente.

```
class Category(models.Model):
    name = models.CharField(null=False, max_length=60, blank=False)
    def __str__(self):
        return self.name
```

Product Model

Questo modello è un modello generico per tutti i prodotti dell'intero sistema che verranno aggiunti nel carrello e successivamente acquistati.

```
class Product(models.Model):
    user = models.ForeignKey(User, related_name='products', on_delete=models.CASCADE)
    category = models.ForeignKey(Category, null=False, blank=False, on_delete=models.CASCADE)
    product_image = models.ImageField(upload_to='productsImage/', blank=True, max_length=200)
    name = models.CharField(max_length=60)
    publisher = models.CharField(max_length=60, blank=False)
    year = models.IntegerField(blank=False)
    author = models.CharField(max_length=50, blank=False)
    created_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True)
    CONDITIONSP = (
        ('0', 'New'),
        ('1', 'Used as new'),
        ('2', 'Heavily used'),
        ('3', 'Bad'),
    )

    condition = models.CharField(default='0', max_length=30, choices=CONDITIONSP, blank=False)
    short_description = models.TextField(max_length=600, blank=False)
    description = models.TextField(max_length=2000, blank=True, )
    price = models.DecimalField(max_digits=8, decimal_places=2)

    class Meta:
        verbose_name_plural = 'Products'
        verbose_name = 'Product'
        ordering = ('-created_at',)

    def __str__(self):
        return f'Product {self.name} of user {self.user}'
```

Cart Model

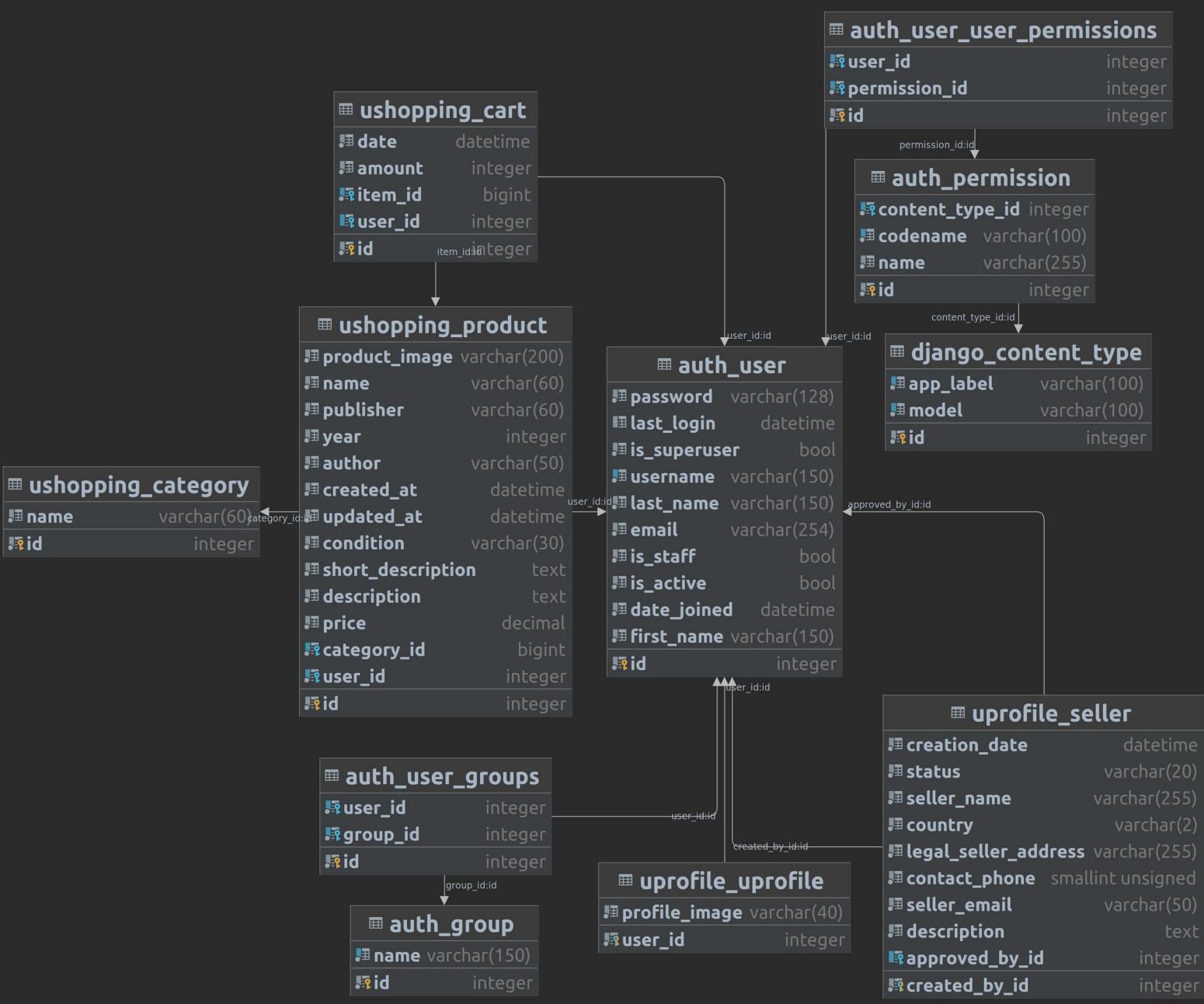
Questo modello è alla base di funzionalità integrante di carrello.

```
class Cart(models.Model):
    user = models.ForeignKey(User, related_name='userto', on_delete=models.SET_NULL, null=True)
    date = models.DateTimeField(auto_now=True)
    amount = models.IntegerField(blank=True)
    item = models.ForeignKey(Product, related_name='productto', on_delete=models.CASCADE)

    class Meta:
        ordering = ('-date',)

    def __str__(self):
        return self.item.name
```

Diagramma stile UML dei modelli del Database



Tecnologie utilizzate

FRONTEND

È stato deciso di utilizzare un framework di tipo free and open-source come Bootstrap per il suo approccio mobile-first, in modo da permettere un prototyping veloce e sicuro che garantisce visione piacevole dai diversi dispositivi mobile/desktop.

Inoltre, è stato deciso di aggiungere high quality, open-source icon library Bootstrap-icons per piacevole supporto visivo del testo.

BACKEND

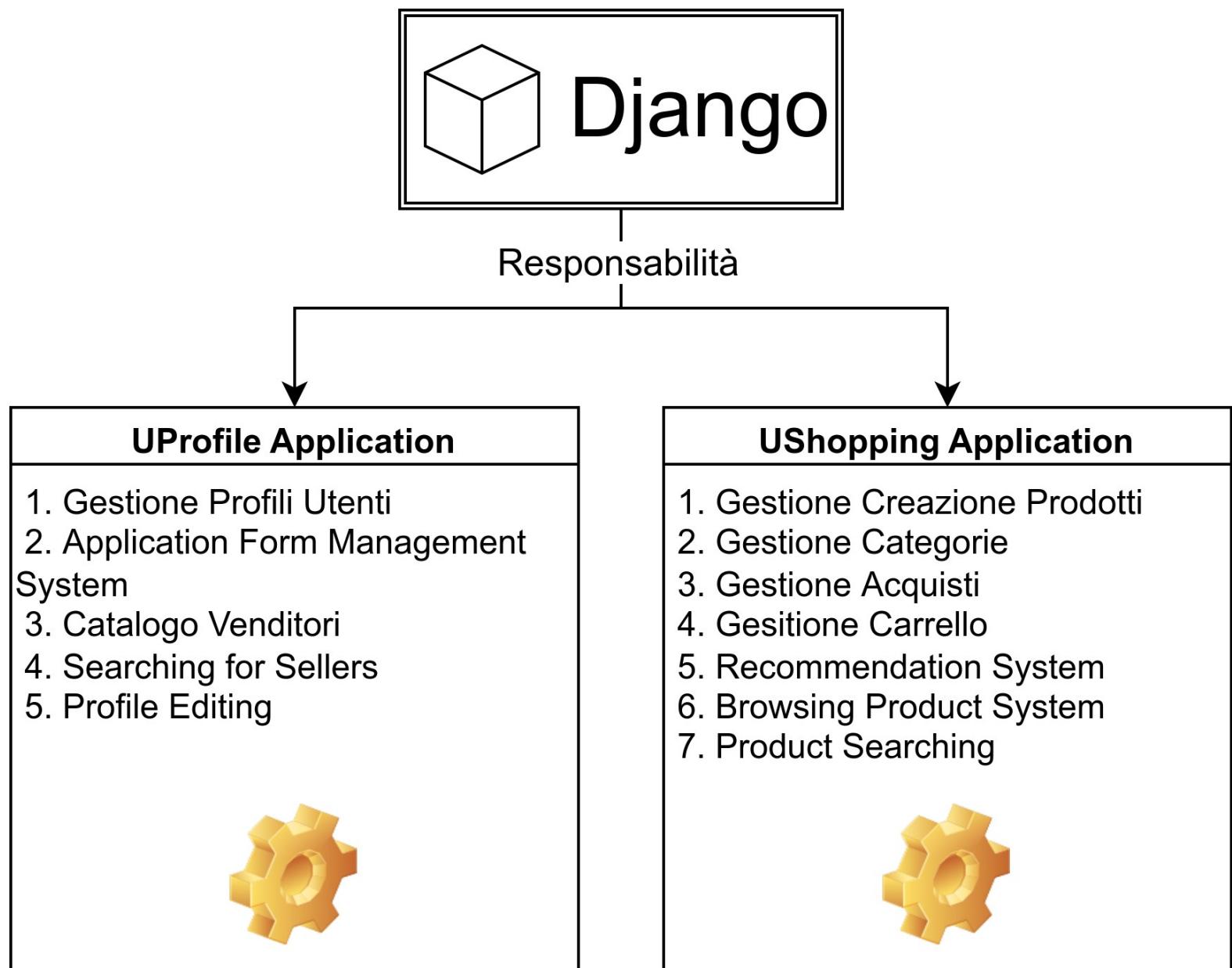
È stato utilizzato un framework di tipo free and open-source Django visto durante il corso di Tecnologie Web.

Come applicazioni aggiuntive al sistema di Django sono stati utilizzati:

- crispy_forms (<https://pypi.org/project/django-crispy-forms/>)
Per permettere di creare dei form eleganti udi avere funzionalità aggiuntive in maniera Don't repeat yourself (DRY).
- django_countries (<https://pypi.org/project/django-countries/>)
Per permettere di integrare un field **CountryField()** in un modello/form che permetterà agli utenti venditori di seleziona il paese di provvidenza e di effettuare la ricerca per questo field all'interno del catalogo di vendori approvati.
- Pillow (<https://pillow.readthedocs.io/en/stable/>)
Permette la gestione di immagini da parte di Pillow che permetterà agli utenti di caricare immagini di profilo e di prodotti.

Organizzazione logica di applicazioni

In questo diagramma viene illustrato la suddivisione logica di lavoro tra diversi parti del sistema.



Problemi riscontrati durante progettazione

[Problem 1] Universal Recommendation System

Selezionare prodotti più acquistati¹ dagli utenti e successivamente suggerirli agli utenti.

Formulazione utilizzando costrutti nativi di Django:

```
# Top 4 Recommended / BESTSELLERS
I = Cart.objects.values('item').annotate(total=Count('item')).order_by('-total')

top4 = list()
for x in I[:4]:
    top4.append(x['item'])

top4_list_items = list()

for x in top4:
    try:
        top4_list_items.append(Product.objects.get(id=x))
    except Exception:
        raise f"[Error]: Can't add item {x} to the top4_list_items. "

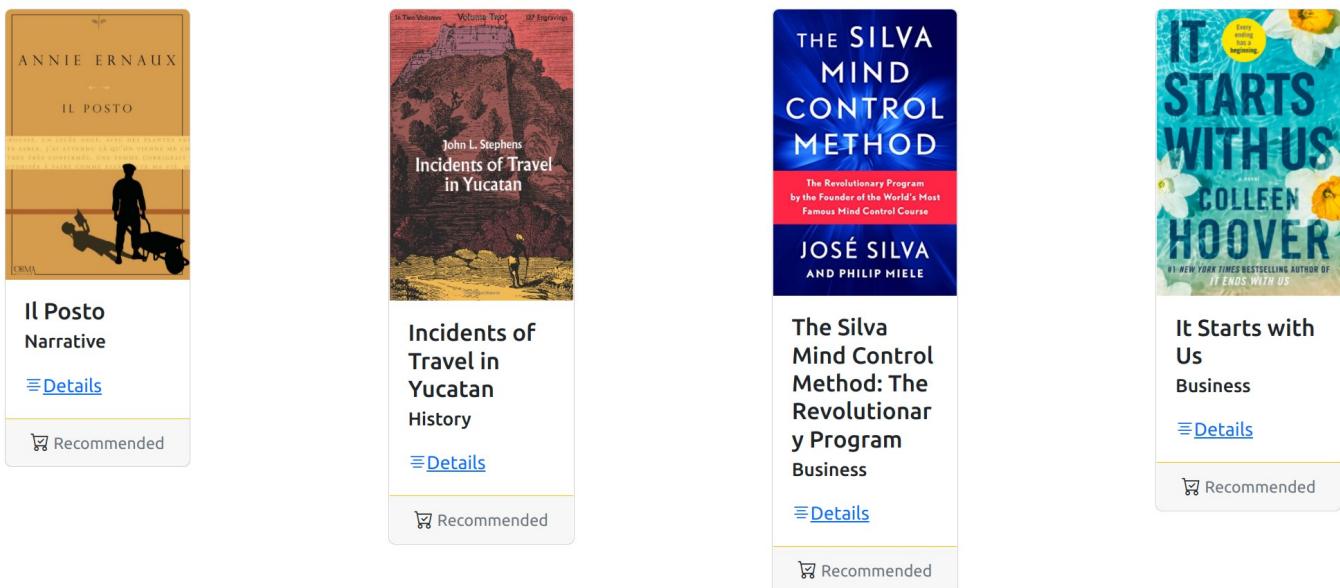
context['top4_list_items'] = top4_list_items
```

Figure 1: /ushopping/views.py (Line 93)

Formulazione in linguaggio SQL:

```
SELECT item_id, COUNT(*) AS TOTAL
FROM ushopping_cart
GROUP BY item_id
ORDER BY TOTAL DESC
```

Top 4 Recommended ↗



¹ Per semplicità implementativa, consideriamo prodotti più aggiunti nel carrello come prodotti acquistati. Ovviamente, lo stato di cose dovrà essere cambiato prima di project deployment.

[Problem 2] User Specific Recommendation System

La sezione di prodotti che verrà suggerita all'utente deve contenere prodotti simili in base alla sua storia di acquisto.

Formulazione utilizzando costrutti nativi di Django e metodi del modulo random² di Python:

```
# User Specific Recommendation System

if self.request.user.is_authenticated:

    if self.request.user.userto.values('item__category').exists():

        CategoryOfInterest = self.request.user.userto.values('item__category').first()['item__category']

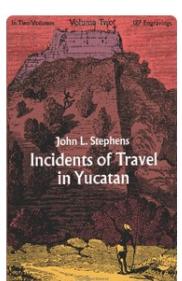
        if CategoryOfInterest:
            count = len(list(Product.objects.filter(category_id=CategoryOfInterest).all()))
            context['UserRecommendation'] = random.sample(list(Product.objects.filter(category_id=CategoryOfInterest).all()), count)
```

Figure 2: /ushopping/views.py (Line 110)

Formulazione in linguaggio SQL:

```
SELECT ushopping_product.category_id
FROM ushopping_cart, ushopping_product
WHERE ushopping_cart.user_id = "USER_ID"3 AND ushopping_cart.item_id =
ushopping_product.id
ORDER BY ushopping_cart.amount DESC
LIMIT 1
```

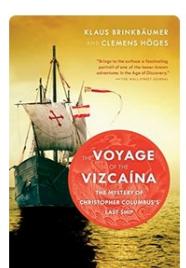
Recommended to You



Incidents of
Travel in
Yucatan
History

[≡ Details](#)

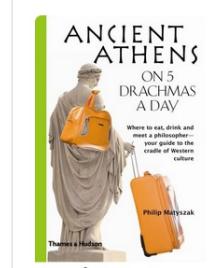
Recommended
to You



The Voyage
of the
Vizcaina
History

[≡ Details](#)

Recommended
to You



Ancient
Athens on 5
Drachmas a
Day
History

[≡ Details](#)

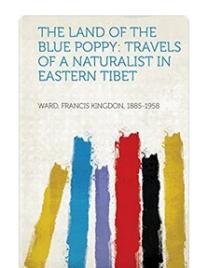
Recommended
to You



Iberia:
Spanish
Travels and
Reflections
History

[≡ Details](#)

Recommended
to You



Travels of a
Naturalist in
Eastern Tibet
History

[≡ Details](#)

Recommended
to You

2 [Lib/random.py](#)

3 Primary key dell'utente del modello default Django **User Model**.

[Problem 3] Advanced Product Searching (Complex lookups with Q object)

Il sistema deve supportare ricerche complesse per prodotti.

Un modo per realizzare data retrieval complesso è di sfruttare l'oggetto **Q object**⁴ che ci permette di incapsulare diversi keywords arguments ed applicare diversi operatori come &, |, ^ and ~ per combinare tutti i risultati in una formulazione compatta ed elegante.

```
def get_queryset(self):
    object_list = super().get_queryset()

    if self.request.GET.get('cfilter'):
        object_list = Product.objects.filter(Q(category__name=self.request.GET.get('cfilter')))

    if self.request.GET.get('as'):
        print(self.request.GET.get('as'))

        object_list = Product.objects.filter(
            Q(name__icontains=self.request.GET.get('as')) |
            Q(author__icontains=self.request.GET.get('as')) |
            Q(publisher__icontains=self.request.GET.get('as')) |
            Q(year__icontains=self.request.GET.get('as')) |
            Q(description__icontains=self.request.GET.get('as'))
        )

    return object_list
```

Figure 3: ushopping/views.py (Line 70)

The screenshot shows a web application interface for searching products. At the top, there are two search/filter sections: "Filter by Category" (dropdown set to "Computers & Tech" with "Apply Filter" button) and "Advanced Search" (input field "Search by Name, Description, Author, Publish" with "Search" button). Below these are three product cards:

- La bestia dentro** (Thriller): Book cover by Kevin Brooks. Summary: "Vorrei che fosse più facile. Vorrei potervi toccare la testa con le mani e trasferirne in voi quello che ho dentro. Vorrei che poteste essere me, solo per un attimo, così capireste esattamente come mi". Buttons: "Details", "Certified".
- Pezzettino** (Narrative): Book cover by Leo Lionni. Summary: "Pezzettino è in cerca della propria identità. È talmente piccolino, infatti, confronto ai suoi amici - tutti grandi e grossi, capaci di volare, nuotare, arrampicarsi - che si convince di essere un pezzo". Buttons: "Details", "Certified".
- NON APRIRE QUESTO LIBRO** (Illustrated): Book cover by Andy Lee. Summary: "L'hai già aperto? Spero che tu non l'abbia letto fino in fondo senza prima passare dalla cassa. Se invece ami le sorprese e vuoi disobbedire al divieto del bizzarro abitante di questo libro, preparati". Buttons: "Details", "Certified".

At the bottom, there are two more book covers: "ANNIE ERNAUX" and "ANCIENT ATHENS ON 5 DRACHMAS".

⁴ [django.db.models.Q](#)

[Problem 4] Application Form Management System

Il sistema deve disporre di un sistema in grado modificare, approvare/respingere lo stato di progressione delle domande di ammissione dei futuri venditori.

The screenshot shows a web-based application for managing seller applications. At the top, there's a navigation bar with links for Home, Browse our Collection, Sellers, Used Books, New Books, Start Selling, a search bar, and a user account dropdown set to 'admin'. Below the navigation, the URL 'Home > Profile > Manage application forms' is visible. A main title 'Currently pending application forms 2' is centered above three cards. The first card, 'Approved application forms 5', is collapsed. The second card, 'Rejected application forms 0', is also collapsed. The third card, 'Currently pending application forms 2', is expanded and contains two items:

- Fantastic Literature Limited**
Nov. 20, 2022, 10:05 p.m.
Visa & Mastercard accepted along with PayPal but not Amex. Returns accepted, if books don't meet your expectation! Please inform on day of receipt if returning. We can provide scanned images of any title if needed, just ask. Web site www.fantasticliterature.com - has new small press, magazines and
[Manage](#)
- Familienbibliothek "Tor zum Spreewald"**
Nov. 20, 2022, 10:03 p.m.
Familienbibliothek "Tor zum Spreewald" Siegfried Baschin AGB Allgemeine Geschäftsbedingungen der Familienbibliothek "Tor zum Spreewald" (Siegfried Baschin) in der Fassung vom 01.10.2021 I. Geltungsbereich Diese Allgemeinen Geschäftsbedingungen gelten für den Verkauf und die Lieferung von gebraucht
[Manage](#)

At the bottom of the page, there are navigation links: 'Page 1 of 2.', 'Next', and 'Last »'.

This screenshot shows a detailed view of a seller's application. The top navigation and user interface are identical to the previous screenshot. The URL now includes '/Detail view of a seller #4'. Key information displayed includes:
Created by User: testSeller3 with id=4
User's email: awdawD@gmail.com
Creation date: Nov. 20, 2022, 9:58 p.m.
Contact phone: 323123213

The form fields for the seller are filled with the following values:

- Seller name*: FalkMedien
- Legal seller address*: Erbach, HE
- Country*: Germany
- Contact phone*: 323123213
- Seller email*: dawew@gmail.com
- Description*: Die Versandkostenpauschalen basieren auf Sendungen mit einem durchschnittlichen Gewicht. Falls das von Ihnen bestellte Buch besonders schwer oder sperrig sein sollte, werden wir Sie informieren, falls zusätzliche Versandkosten anfallen.
- Status*: approved

A large blue 'Confirm' button is at the bottom right of the form.

[Problem 5] Editing of Seller's Products and Creating New Products

Gli utenti venditori approvati nel suo profilo personale avranno la possibilità di creare nuovi prodotti compilando un form e visualizzare/modificare propri prodotti creati.

Gli utenti venditori posso modificare solo propri prodotti creati.
Il sistema deve evitare modifiche di prodotti da parte dei venditori esterni a cui il prodotto non appartiene.

Soluzione adottata:

```
def test_func(self):
    if self.request.user.is_staff:
        return 1 # Accept

    if self.request.user.groups.filter(name='seller').exists(): # If user is part of seller group
        if self.request.user.products.filter(id=self.kwargs['pk']).exists(): # If product is related to user
            return 1 # Accept

    return self.handle_no_permission() # Reject others
```

The screenshot shows a web page for creating a new product. At the top, there is a navigation bar with links: Home, Browse our Collection, Sellers, Used Books, New Books, Start Selling, a search bar labeled 'Search...', and a user profile icon for 'testSeller6 | seller'.

The main content area is titled 'New Product'. It contains several input fields:

- Please enter product name.***: A text input field containing "Endless Night."
- Please enter publisher name of a product.***: A text input field containing "Fontana".
- Year***: A dropdown menu currently showing an empty value.
- Condition***: A dropdown menu currently showing "New".
- Please provide author name of a product***: A text input field containing "Agatha Christie".
- Please enter category of a product***: A dropdown menu currently showing an empty value.
- Please provide a short description of a product.***: A text input field containing "Enter short description here".
- Please provide full description of a product.**: A large text area for a full description, with the placeholder "Enter full description here".
- Product image***: A file input field with a 'Browse...' button. The status message "No file selected." is displayed.
- Please enter price of a product.***: A dropdown menu currently showing an empty value.

At the bottom right of the form is a blue button labeled "Confirm".

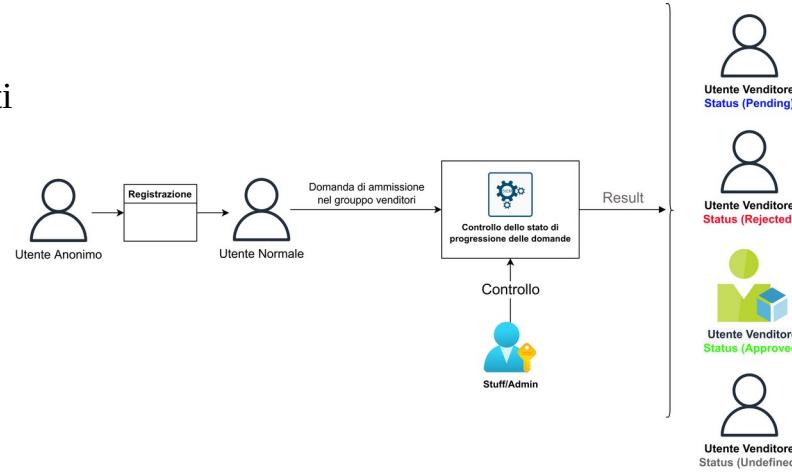
[Problem 6] Personal Profile

Gli utenti che hanno compilato il modulo di domanda per ammissione possono vedere lo stato di progressione nel suo profilo personale. Lo stato di **progressione** può essere di seguenti tipi:

- in attesa di approvazione
- approvato (in seguito l'utente diventa ufficialmente un **venditore approvato** e può cominciare a vendere prodotti)
- respinto

Soluzione adottata:

È necessario gestire diversi gruppi di utenti e differenziare comportamento in base al gruppo di appartenenza.



```

if Group.objects.filter(user=self.request.user).exists():
    # User did use MyUserCreationForm or is in a group
    context["group"] = Group.objects.filter(user=self.request.user).first().name.capitalize()
else:
    # User didn't use MyUserCreationForm
    if self.request.user.is_staff:
        context["group"] = 'Staff'
    else:
        context["group"] = 'Undefined'

# Check if user has UProfile
if UProfile.objects.filter(user=self.request.user.id).exists():
    context["uprofile"] = get_object_or_404(UProfile, user=self.request.user)
else:
    context["uprofile"] = False

if Seller.objects.filter(created_by=self.request.user.id).exists():
    # User has applied to become a seller or already approved seller. I can obtain an object for a template.
    context["seller"] = get_object_or_404(Seller, created_by=self.request.user)
else:
    # User is normal and did not apply yet to become a seller.
    context["seller"] = False
  
```

Figure 4: /uprofile/views.py (Line 106)

Figure 4: /uprofile/views.py (Line 106)

| Current Application Status | User Information |
|----------------------------|--|
| Pending | testUser2 normal testSeller1 seller testUser1 normal |
| Rejected | testUser2 normal testSeller1 seller testUser1 normal |
| Approved | testUser2 normal testSeller1 seller testUser1 normal |

Unit Testing

Da qui in poi, verranno mostrati dei test più critici per il corretto funzionamento del sistema nel suo complesso.

1. Cart access/deleting system check.

```
class TestDeleteOtherCartProducts(TestCase):
    """
    Init: Create User1 and User2. Create a new Category 'Horror' and Product with a name 'Green'.
    User1 will add a product with a name 'Green' to his cart.
    """

    def setUp(self):
        self.user1 = User.objects.create(username='user1', password='DpaDwdqwd!23', email='ates@tsr.com')
        self.user2 = User.objects.create(username='user2', password='DpaDwdqwd!23', email='atddaes@tsr.com')
        self.category = Category.objects.create(name='Horror')
        self.product = Product.objects.create(user=self.user1, category=self.category, name='Green', year='1992',
                                              publisher='dqw', condition='0', short_description='w', description='2', price='22')
        self.user1cart = Cart.objects.create(user=self.user1, amount=1, item=self.product)
        self.client = Client()

    """
    Test: User2 will try to delete cart item of User1. System should reject any actions from User1.
    """

    def test_user2_deletes_cartItem_of_user_1(self):
        self.client.force_login(self.user2)
        response = self.client.post(reverse('ushopping:cart-delete-view', kwargs={'pk': 1}), {'user_id': self.user2.id})
        self.assertEqual(response.status_code, 403)
        self.client.logout()

    """
    Test: User1 will try to delete item from his cart. System should allow this type of action.
    """

    def test_user1_access_to_his_delete_view(self):
        self.client.force_login(self.user1)
        response = self.client.get(reverse('ushopping:cart-delete-view', kwargs={'pk': 1}), {'user_id': self.user1.id})
        self.assertEqual(response.status_code, 200)
        self.client.logout()
```

2. Seller apply form.

```
class SellerForm(TestCase):
    """
    Test seller apply form.
    """

    def test_seller_apply_form_test1(self):
        form_data = {
            'seller_name': 'Test',
            'country': 'AZ',
            'legal_seller_address': 'asdsda',
            'contact_phone': '3213213',
            'seller_email': 'dawdsasd@gmai.com',
            'description': 'x'*100,
        }
        form = SellerApplyForm(data=form_data)
        self.assertTrue(form.is_valid())

    """
    Test seller name can not be empty.
    """

    def test_seller_apply_form_test2(self):
        form_data = {
            'seller_name': '',
            'country': 'Test',
            'legal_seller_address': 'ddsa@dasg**',
            'contact_phone': '3213213',
            'seller_email': 'dawd@gmai.com',
            'description': 'x'*100,
        }
        form = SellerApplyForm(data=form_data)
        self.assertFalse(form.is_valid())

    """
    Test seller description should contains at least 75 characters.
    """

    def test_seller_descriotion(self):
        form_data = {
            'seller_name': 'Test',
            'country': 'Test',
            'legal_seller_address': 'ddsa@dasg**',
            'contact_phone': '3213213',
            'seller_email': 'dawd@gmai.com',
            'description': 'x',
        }
        form = SellerApplyForm(data=form_data)
        self.assertFalse(form.is_valid())
```

3. Checking constraints of the Product model.

```
class ModelProductTests(TestCase):

    def setUp(self):
        self.user = User.objects.create_user(username='testuser', password='12345')

        self.C = Category.objects.create(name='TestCategory')
        Product.objects.create(user=self.user, category=self.C, name='T', year=123, price=23)

    """
    Number of products created should be equal to 1
    """

    def test_number_products_created(self):
        self.assertEqual(Product.objects.all().count(), 1)

    """
    New product must be part of a category.
    """

    def test_new_product_must_have_category(self):
        with self.assertRaises(IntegrityError):
            Product.objects.create(user=self.user)
```

4. User creation form.

```
class UserForm(TestCase):

    """
    Test user creation form.
    """

    def test_create_user_form(self):
        form_data = {'username': 'Test',
                    'first_name': 'Test',
                    'last_name': 'Test',
                    'email': 'ddsa@dasg.com',
                    'password1': 'tDwdWECWe2',
                    'password2': 'tDwdWECWe2',
                    }
        form = MyUserCreationForm(data=form_data)
        self.assertTrue(form.is_valid())

    """
    Test user edit form.
    """

    def test_user_edit_form(self):
        form_data = {
                    'first_name': 'Test',
                    'last_name': 'Test',
                    'email': 'ddsa@dasg.com',
                    }
        form = UserEditForm(data=form_data)
        self.assertTrue(form.is_valid())
```

5. Access to the product creation page.

```
class TestAccessCreationProduct(TestCase):

    def setUp(self):
        self.user = User.objects.create(username='test', password='DpaDwdqwd!23', email='ates@tsr.com')
        self.client = Client()

    """
    Test access to the product creation page when user is not a seller.
    """

    def test_user_access(self):
        self.client.force_login(self.user)
        response = self.client.post(reverse('ushopping:create-product'), {'user_id': self.user.id})
        response2 = self.client.get(reverse('ushopping:create-product'))
        self.assertEqual(response.status_code, 403)
        self.assertEqual(response2.status_code, 403)
```

6. Access to the management page when user is not staff.

```
class TestAccessManageApplications(TestCase):

    def setUp(self):
        self.user = User.objects.create(username='test', password='DpaDwdqwd!23', email='ates@tsr.com')
        self.user.is_staff = 1
        self.client = Client()

    """
    Test access to application management page when user is not staff.
    """

    def test_user_access(self):
        self.client.force_login(self.user)
        response = self.client.get(reverse('uprofile:sellers-app-forms'), {'user_id': self.user.id})
        response2 = self.client.post(reverse('uprofile:sellers-app-forms'))
        self.assertEqual(response.status_code, 403)
        self.assertEqual(response2.status_code, 403)
```

Collection of screenshots from different pages.

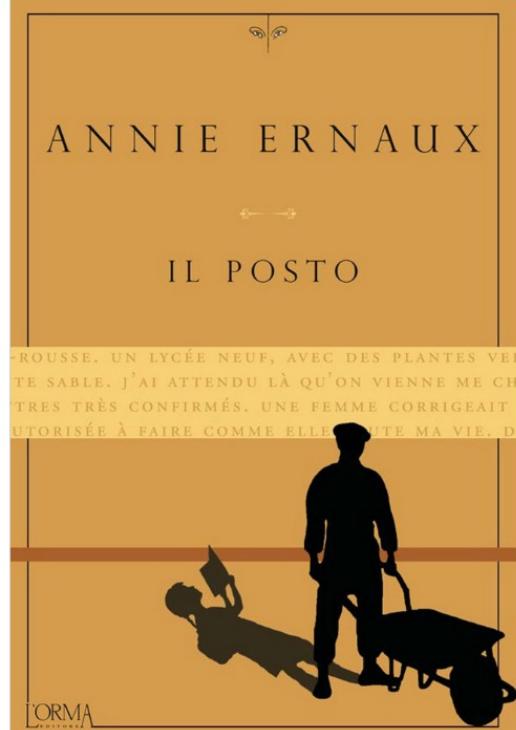
The screenshot shows the homepage of Right Books. At the top, there is a navigation bar with links for Home, Browse our Collection, Sellers, Used Books, New Books, Start Selling, a search bar, and Login/Sign up buttons. Below the navigation is a large logo featuring a blue graduation cap resting on an open book, with the text "RIGHT BOOKS FOR RIGHT PEOPLE" underneath. A central heading reads "Millions of books". Below this, a paragraph describes the website's mission: "Millions of books and other objects are listed for sale on our websites by thousands of sellers located in more than 50 countries. Our epic selection stretches from manuscripts created before the invention of the Gutenberg Press to the latest signed bestsellers. We have customers in almost every country of the world, who are buying books and other printed matter published in multiple languages." At the bottom of the main content area are two buttons: "Start Selling" and "Start Buying".

Figure 5: Homepage

The screenshot displays two pages from the Right Books website. On the left, the "Seller Catalog" page lists several certified sellers with their names, countries, legal addresses, and "Details" links. The sellers include "Fantastic Literature Limited" (United Kingdom), "Familienbibliothek 'Tor zum Spreewald'" (Germany), "FalkMedien" (Germany), "Good Books" (Germany), and "Fábulas Libros (Librería Jiménez-Bravo)" (Spain). Each seller has a "Certified Seller" badge. On the right, a detailed view of the "Familienbibliothek 'Tor zum Spreewald'" seller is shown. This view includes a large photo of the bookstore interior, a form for entering business details like name, address, and email, and a "Description" section containing the bookstore's terms and conditions. The URL in the browser bar is "Home > Browse our Collection > Seller Details".

Figure 7: Catalog of Approved/Certified Sellers

Figure 6: Detailed View of a Seller



Product Name
Il Posto

Author
Annie Ernaux

Publisher
Forno

Category
Narrative

Year
2022

Condition
New

Price
123.00 Euro

Description
Il Premio Nobel per la letteratura cerca l'eccellenza anche in ambiti non correlati alla narrativa. Tra i premiati figurano autori di saggistica (Winston Churchill e Bertrand Russell), poeti (come Pablo Neruda e T.S. Eliot), drammaturghi (come Harold Pinter e Nelly Sachs), una scrittrice di racconti (Alice Munro) e persino un cartuttore (Bob Dylan).

Information about Seller
This item is sold by our certified seller testSeller4. You can find all information about seller [here](#)

Certified Seller

Add to Cart
You logged in as a staff. You can edit this product.
[Edit Product](#)

Figure 8: Detailed View of a Product by Staff/Admin

A screenshot of the shopping cart page. It shows two items: 'The Silva Mind Control Method' by José Silva and Philip Miele, and 'Il Posto' by Annie Ernaux. Each item has a thumbnail image, title, author, a short description, and buttons for details, remove, and buy. The total amount is listed as 1.

[Page 1 of 2.](#) [Next](#) [Last »](#)

Figure 9: Cart View

First name*

This field is required.

Required. Only letters are allowed.

Last name*

This field is required.

Required. Only letters are allowed.

Email confirmation address*

This field is required.

Please enter email address.

Profile image
 No file selected.

Figure 10: Profile Editing

RIGHT BOOKS
FOR RIGHT PEOPLE

Home Browse our Collection Sellers Used Books New Books Start Selling Search... testUser1 | normal

Home > Profile > Update Password

Old password*

New password*

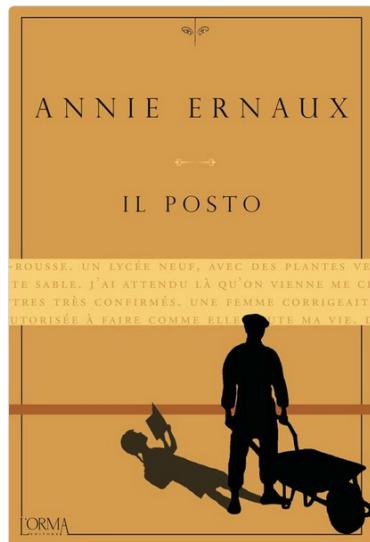
- * Your password can't be too similar to your other personal information.
- * Your password must contain at least 8 characters.
- * Your password can't be a commonly used password.
- * Your password can't be entirely numeric.

New password confirmation*

Figure 11: Password Changing View


[Home](#) [Browse our Collection](#) [Sellers](#) [Used Books](#) [New Books](#) [Start Selling](#)
Search...
testSeller4 | seller 

[Home](#) > [Profile](#) > [My Listings](#)



Il Posto

Narrative

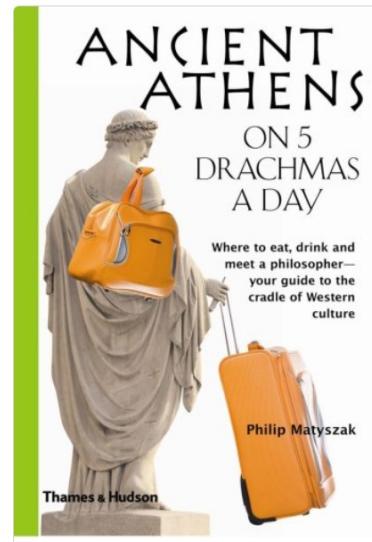
Author - Annie Ernaux

Publisher - Forno

Il Premio Nobel per la letteratura cerca l'eccellenza anche in ambiti non correlati alla narrativa. Tra i premiati figurano autori di saggistica (Winston Churchill e Bertrand Russell), poeti (come Pablo Neruda e T.S. Eliot), drammaturghi (come Harold Pinter e Nelly Sachs), una scrittrice di racconti

[View](#) [Edit](#)

123.00 Euro



Ancient Athens on 5

History

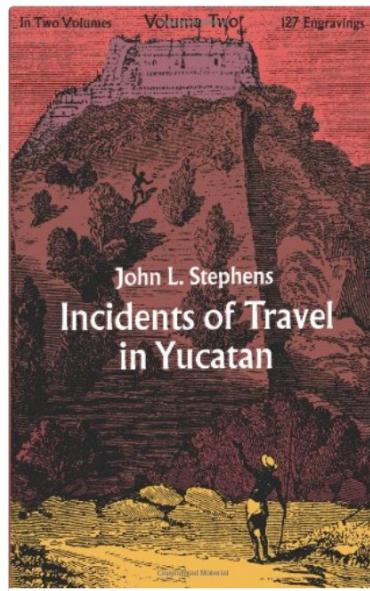
Author - Matyszak, Philip

Publisher - PPW

A lighthearted tour of ancient Athens profiles such locations as Delphi, the site of the Battle of Marathon, and the city of Athena while surveying the philosophers, artists, and gods who shaped famous sites, in a reference that also explores the hedonistic side of Athenian life. Original. 15,000 fi

[View](#) [Edit](#)

42.00 Euro



Incidents of Travel

History

Author - Stephens, John L.

Publisher - Alish

Volume 2 of two-volume set. Classic (1843) exploration of jungles of Yucatan, looking for evidences of Maya civilization. Extensive accounts of 44 Maya sites as well as of Yucatan folkways, manners, dress, ceremonies, amusements? all of which makes this a great travel book. Total in set: 127 engrav

[View](#) [Edit](#)

33.00 Euro

Figure 12: Seller's Listing View


[Home](#) [Browse our Collection](#) [Sellers](#) [Used Books](#) [New Books](#) [Start Selling](#)
Search...
testSeller4 | seller 

[Home](#) > [Profile](#) > [Edit My Product](#)

Name*

Publisher*

Year*

Condition*
 New

Author*

Category*

Description*
A lighthearted tour of ancient Athens profiles such locations as Delphi, the site of the Battle of Marathon, and the city of Athena while surveying the philosophers, artists, and gods who shaped famous sites, in a reference that also explores the hedonistic side of Athenian life. Original. 15,000 first printing.

Description
A lighthearted tour of ancient Athens profiles such locations as Delphi, the site of the Battle of Marathon, and the city of Athena while surveying the philosophers, artists, and gods who shaped famous sites, in a reference that also explores the hedonistic side of Athenian life. Original. 15,000 first printing.

Product image
Currently: [productimage/9780500287651-it.jpg](#) Clear
Change: [Browse...](#) No file selected.

Price*

[Confirm](#)

Figure 14: Seller's Editing View


[Home](#) [Browse our Collection](#) [Sellers](#) [Used Books](#) [New Books](#) [Start Selling](#)
Search...
testSeller4 | seller 

[Home](#) > [Profile](#) > [New Product](#)

Please enter product name.*

Please enter product name.

Please enter publisher name of a product.*

Year*

Condition*
 New

Please provide author name of a product*

Please enter category of a product*

Please provide a short description of a product.*

Please provide full description of a product.

Product image
Browse... No file selected.

Please enter price of a product.*

[Confirm](#)

Figure 13: Seller's Creating new Product View