

Progetto di Tecnologie Web

SmartStore di Luca Carnevale (matr. 131694)

Indice

1. Descrizione del progetto
2. Diagrammi
3. Tecnologie utilizzate
4. Test
5. Screenshot
6. Conclusioni

1 – Descrizione del progetto

Il progetto consiste nella progettazione di un sito di e-commerce in stile eBay o Amazon.

L'obiettivo è quello di offrire a chiunque la possibilità di vendere o acquistare qualsiasi tipo di prodotto in maniera estremamente semplice e veloce.

Il progetto è suddiviso nelle seguenti app :

- **accounts**, nella quale viene gestita la parte relativa agli account degli utenti, quali registrazione e gestione degli indirizzi
- **core**, nella quale viene gestita tutta la parte relativa alle varie pagine del sito, ad esempio la homepage, gli acquisti e le vendite degli utenti ecc.
Le maggiori difficoltà sono state riscontrate durante l'implementazione di questa applicazione.
- **smartstore**, applicazione di base dalla quale parte tutto il progetto

1.2 – Tipologie di utenti presenti

- **Guest** : non è autenticato e ha la possibilità di visitare liberamente il sito, vedere e cercare i prodotti in vendita e i loro dettagli, cercare gli utenti e vedere cosa vendono e navigare per categoria; non potrà invece vedere la sua lista dei consigliati o acquistare prodotti finché non sarà registrato.
- **Utente** : dopo essersi registrato attraverso l'apposito tasto, questo tipo di utente può fare qualsiasi cosa sul sito, dall'acquisto alla vendita di prodotti, alla visualizzazione di una pagina di "articoli consigliati" creata per lui sulla base dei suoi acquisti, sulla media dei prezzi degli articoli acquistati e sulla condizione degli oggetti.
Un utente ovviamente, non può acquistare i prodotti da lui messi in vendita.
- **Admin** : L'admin è colui che si occupa della gestione del sito da dietro le quinte; esso potrà eliminare articoli (che magari non rispettano dei requisiti), utenti e controllare tutti i movimenti relativi ad ordini, pagamenti e articoli che vengono inseriti sul sito.

1.3 – Gestione degli ordini

I prodotti possono essere acquistati solo da utenti registrati e loggati; se il carrello è vuoto, all'utente apparirà un messaggio del tipo "Il carrello è vuoto.", altrimenti, aggiungendo dei prodotti al carrello verrà creato un nuovo ordine.

Nella pagina relativa al carrello, sono elencati tutti i prodotti aggiunti dall'utente con la possibilità di rimuoverli, il totale del carrello che comprende i prezzi di tutti gli articoli e due pulsanti, uno per il checkout e uno per tornare indietro e continuare con lo shopping; appena l'utente procede con il checkout, verrà reindirizzato in un'altra pagina nella quale inserirà l'indirizzo di spedizione (Stato, Città, Via, CAP, Interno(opzionale)) ed eventuali note aggiuntive, opzionali anch'esse e un metodo di pagamento.

Una volta che l'utente avrà inoltrato l'ordine, all'interno del suo profilo troverà un'apposita pagina che riassume tutti gli ordini da lui effettuati, descritti da un numero che rappresenta il numero d'ordine (globale), la data dell'ordine e il totale; è inoltre presente un pulsante che permette di vedere i dettagli dell'ordine, ovvero i prodotti acquistati dall'utente, il relativo prezzo e il totale dell'ordine.

1.4 – Vendere un prodotto

La vendita di un prodotto può essere effettuata come nel caso dell'acquisto, solo da utenti registrati e loggati.

Questa azione può essere eseguita attraverso il pulsante "Vendi un articolo" presente nella homepage o nella pagina del profilo utente.

Da qui, l'utente dovrà inserire tutti i dettagli del prodotto quali nome, descrizione, immagine rappresentante il prodotto, condizione dell'oggetto, luogo in cui si trova l'oggetto e CAP; tutti questi campi sono obbligatori.

Una volta che il prodotto viene messo in vendita, sarà visibile da chiunque sul sito.

L'utente potrà quindi vedere il proprio oggetto sia tra gli articoli in vendita all'interno del sito che sul suo profilo, dalla quale ha anche la possibilità di modificarlo o rimuoverlo.

Nel momento in cui l'oggetto viene venduto, verrà riportato nella sezione "Vendite" all'interno del profilo utente.

1.5 – Homepage

Nella homepage vengono visualizzati innanzitutto gli ultimi articoli inseriti, divisi per condizione "Nuovo" e "Usato" e se l'utente è loggato, anche una sezione relativa ai "Consigliati per te"; sono presenti anche due pulsanti, dei quali uno permette di visualizzare tutti gli articoli in vendita e un altro permette di vendere un oggetto.

Nella barra di navigazione troviamo inizialmente i pulsanti per la registrazione e il login di un utente; qualora l'utente è loggato, apparirà tutta la parte con le informazioni relative al profilo di quest'ultimo quali vendite, acquisti, profilo, cambio password e logout.

1.6 – Consigliati per te -> Recommendation System

Il sistema di raccomandazione che ho incluso nel progetto consiste nel mostrare ad un utente ulteriori articoli da poter acquistare.

Per implementarlo nel progetto è stata creata una view “*RecommendedItem*” che tiene conto degli acquisti precedentemente effettuati, in particolar modo delle condizioni (nuovo o usato) degli articoli acquistati e della media dei totali relativi agli acquisti effettuati.

```
138 |
139 | class RecommendedItem(models.Model):
140 |     num_item = models.PositiveIntegerField(default=0)
141 |     condizioneN = models.PositiveIntegerField(default=0)
142 |     condizioneU = models.PositiveIntegerField(default=0)
143 |     condizione0 = models.PositiveIntegerField(default=0)
144 |     user = models.ForeignKey(User, on_delete=models.CASCADE)
145 |     prezzo = models.FloatField(validators=[MinValueValidator(0.0)], default=0)
146 |     sum_prezzo = models.FloatField(default=0)
147 |
148 |     def __str__(self):
149 |         return self.user.username
150 |
151 |     class Meta:
152 |         verbose_name_plural = 'Recommended Item'
153 |
```

Questo modello altro non fa che salvare i dati degli ordini di un utente, in particolare tiene conto del numero di articoli acquistati, del numero di articoli con condizione *Nuova* e *Usata* e della media dei prezzi degli articoli acquistati.

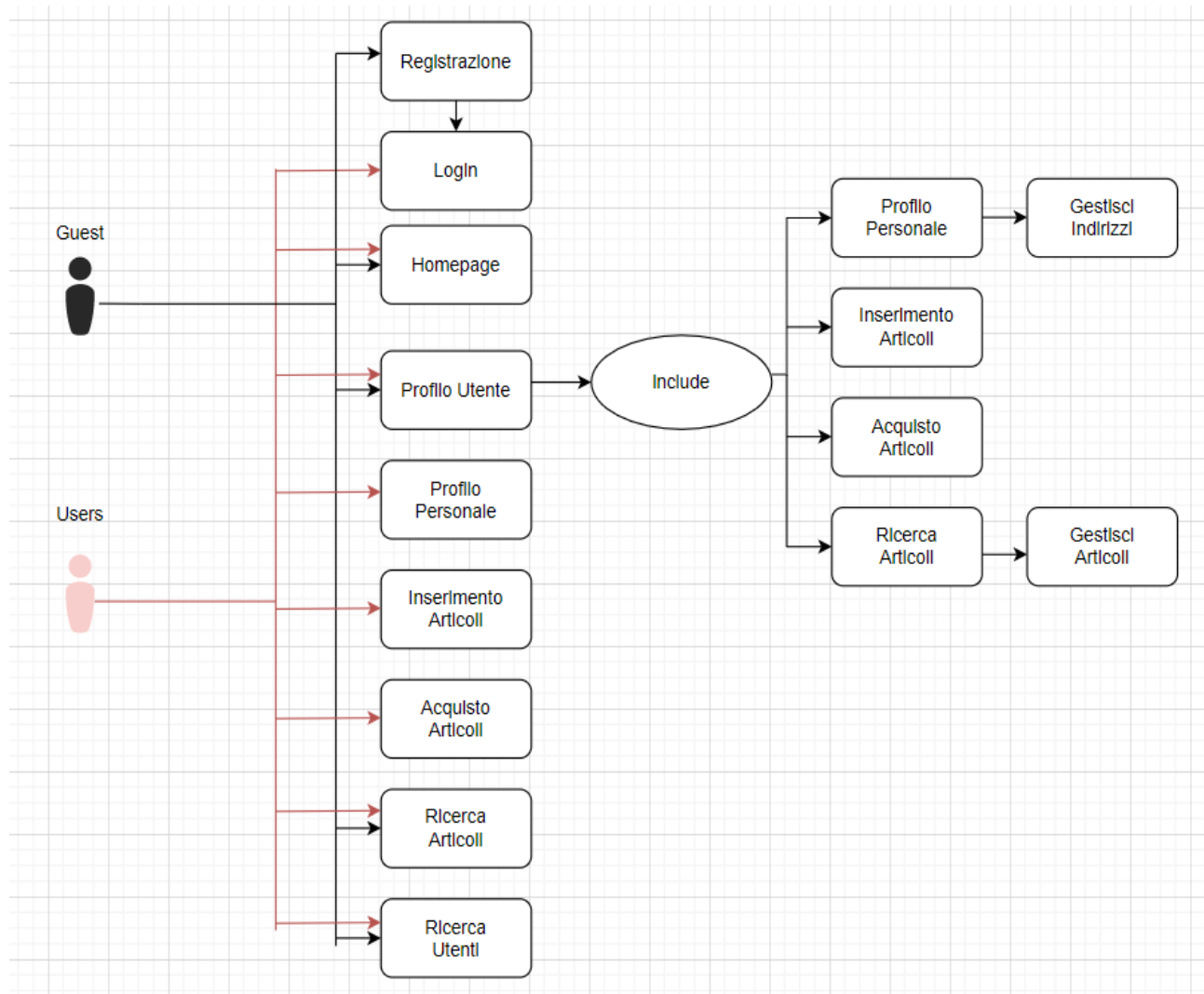
A questo punto viene creata una lista di raccomandazioni nella quale vengono aggiunti gli articoli che rispettano determinati canoni di raccomandazione, in questo caso una combinazione tra condizione e prezzo dell'articolo.

Quindi, per capire meglio :

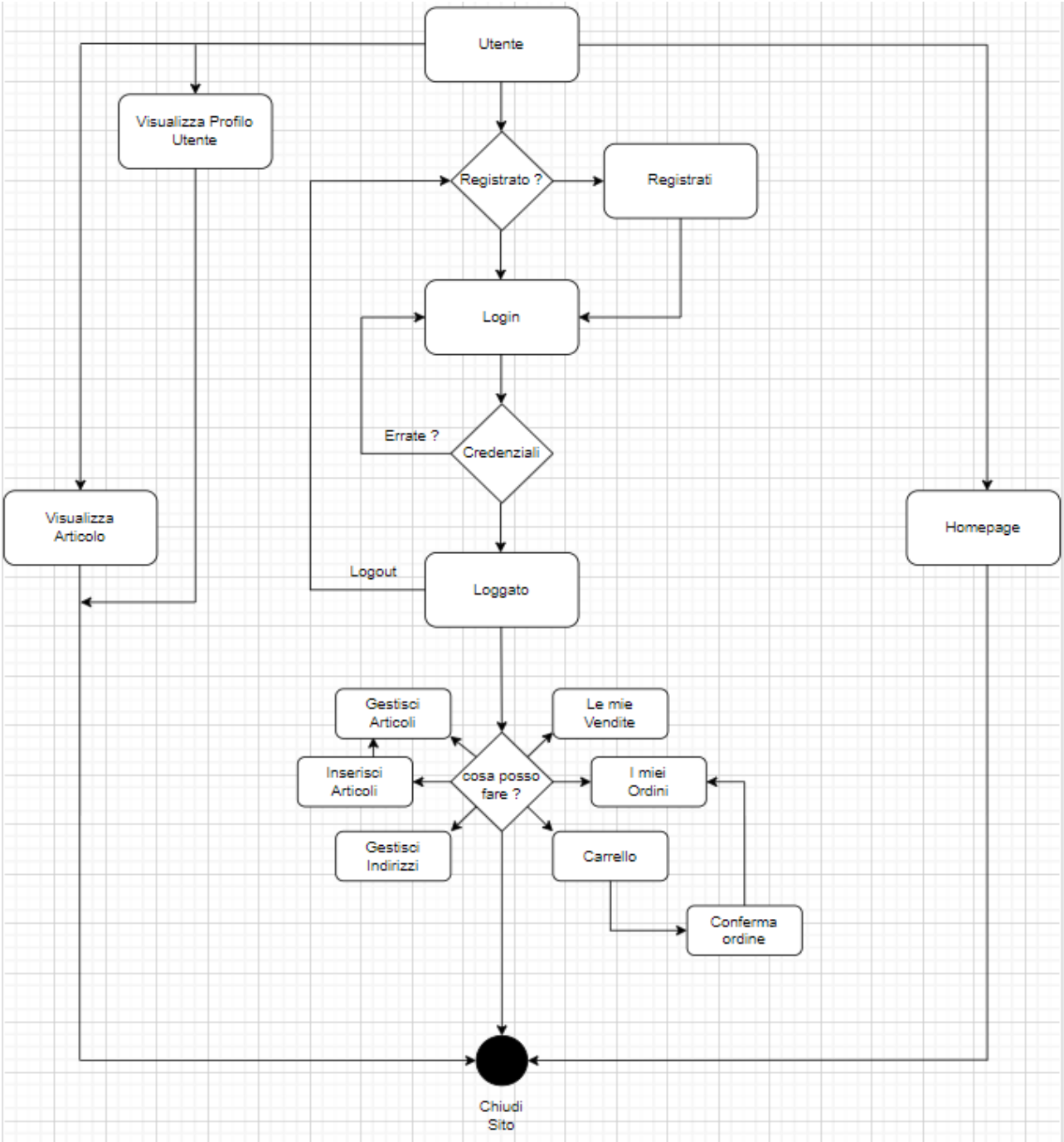
- si controlla se il prezzo dell' articolo rientra, ad esempio, nella fascia di media > 100 € oppure < 100 €
- si controlla la condizione dell' articolo e il numero di articoli ordinati per ogni condizione
- se questi requisiti sono soddisfatti, l' articolo viene aggiunto alla lista “Consigliati per te” e vengono mostrati nella homepage dell' utente a partire da quello con il prezzo minore.

2 – Diagrammi

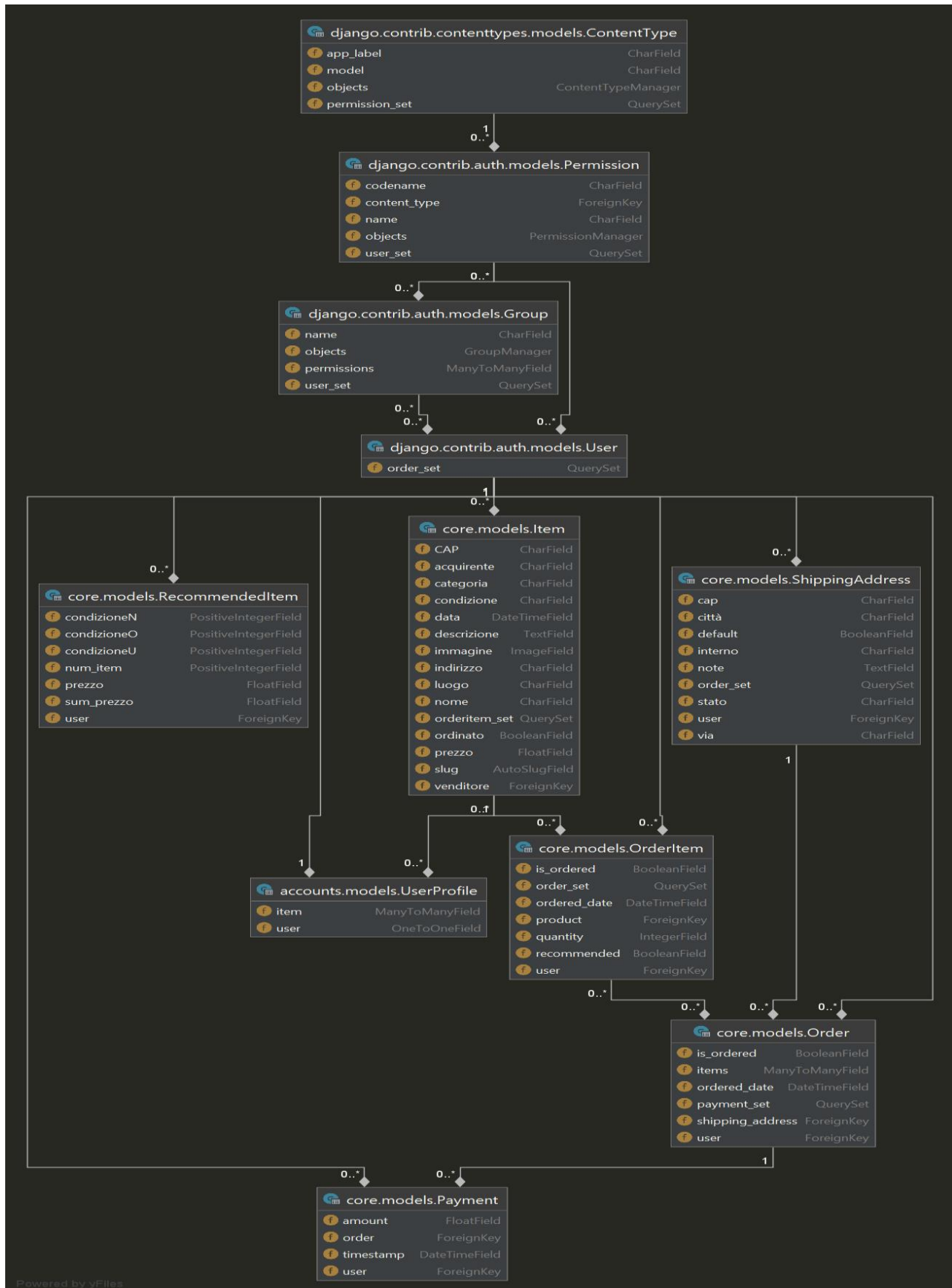
2.1 – Diagrammi dei casi d'uso



2.2 – Diagramma di utilizzo



2.3 – Diagramma delle classi



3 – Tecnologie utilizzate

3.1 – Django

Questo progetto è basato su Django, un Framework MVC (Model View Controller) per lo sviluppo Web.

La parte Model di Django è implementata direttamente come collezione di classi Python che andranno a rappresentare le tabelle del database.

La parte View tratta di funzioni Python che gestiscono il flusso dell' applicazione; in particolar modo in questa parte si definiscono i comportamenti che avranno le pagine in base alle azioni dell' utente.

La parte Controller è realizzata tramite i file urls.py, che permettono di mappare le URL sulle opportune richieste dall' utente.

Django di default presenta un' interfaccia di amministrazione che permette di gestire comodamente il database dell' applicazione.

Uno dei punti di forza di Django è la riusabilità dei componenti: in particolare permette di dividere il progetto in varie app in modo da facilitare la riusabilità e la leggibilità del codice.

3.2 – Javascript

Javascript è un linguaggio di programmazione interpretato e orientato agli oggetti.

È conosciuto come linguaggi di scripting client-side per pagine web e viene utilizzato principalmente per il design delle pagine web e per stabilire il comportamento delle pagine web quando viene invocato un evento da parte dell' utente.

3.3 – Bootstrap

Bootstrap è un framework e rappresenta una delle soluzioni più utilizzate per la progettazione dei template per le pagine web, soprattutto per quanto riguarda lo sviluppo *responsive*, ovvero quando sono presenti dei contenuti dinamici.

Permette inoltre di creare progetti in maniera estremamente rapida e veloce.

3.4 – Ajax

Ajax è una tecnica di sviluppo software per la realizzazione di applicazione web interattive.

Lo sviluppo di applicazioni HTML con Ajax si basa su uno scambio asincrono di dati in background fra web browser e server, la quale consente di aggiornare dinamicamente una pagina web senza esplicito ricaricamento da parte dell' utente.

Lo scambio di dati è quindi asincrono perché essi vengono richiesti al server e caricati in background senza interferire con il comportamento della pagina esistente.

4 – Test

4.1 – Test login (accounts)

In questo test, si prova il login di un utente e si controlla che le credenziali inserite siano corrette; il test viene provato sia con credenziali errate che con credenziali corrette.

Qualora risultino corrette, il login viene effettuato correttamente.

```
'''
Test per il login di un utente
Prima viene testato con credenziali errate, poi con credenziali corrette
'''
class LoginTest(TestCase):

    def setUp(self):
        self.test_user = User.objects.create_user(username='test', password='gamberetto', email='test@test.com')

    def test_login(self):
        wrong_credential = {'username': 'ciao', 'password': 'ciao'}
        true_credential = {'username': 'test', 'password': 'gamberetto'}
        t_cred = self.client.login(**true_credential)
        w_cred = self.client.login(**wrong_credential)

        self.assertTrue(t_cred)
        self.assertFalse(w_cred)
```

4.2 – Test inserimento articolo (core)

Qui si prova a mettere in vendita un articolo, in particolar modo viene verificato che nessun campo sia lasciato vuoto e che tutti siano inseriti in forma e maniera corretta.

```
def test_item_create(self):
    '''
    Verifico che nell' inserimento di un articolo i campi obbligatori siano rispettati
    '''
    self.client.login(**self.credential)
    response = self.client.post('/inserisci_articolo/', {})
    self.assertFormError(response, 'form', 'nome', 'Questo campo è obbligatorio.')
    self.assertFormError(response, 'form', 'prezzo', 'Questo campo è obbligatorio.')
    self.assertFormError(response, 'form', 'categoria', 'Questo campo è obbligatorio.')
    self.assertFormError(response, 'form', 'descrizione', 'Questo campo è obbligatorio.')
    self.assertFormError(response, 'form', 'immagine', 'Questo campo è obbligatorio.')
    self.assertFormError(response, 'form', 'condizione', 'Questo campo è obbligatorio.')

    self.assertTemplateUsed(response, 'core/inserisci_articolo.html')
    self.assertEqual(response.status_code, 200) # verifica per capire se il template utilizzato è quello corretto

    response = self.client.post('/inserisci_articolo/',
                                {'nome': 'Computer Asus', 'prezzo': '1250', 'categoria': 'I',
                                 'descrizione': 'funzionante',
                                 'immagine': 's',
                                 'condizione': 'U'})
    self.assertTemplateUsed(response, 'core/inserisci_articolo.html')
    self.assertEqual(response.status_code, 200)
```


4.3 – Test eliminazione articolo (core)

In questo test si prova l'eliminazione di un articolo.

Siccome può essere effettuata solo dal venditore dell'oggetto, si controlla che chi sta tentando di eliminare l'oggetto sia quest'ultimo, altrimenti si testa il reindirizzamento alla homepage.

```
'''
Test per verificare che l'eliminazione di un articolo vada a buon fine
'''

def test_item_delete(self):
    # con utente autenticato
    self.client.login(**self.credential)
    response = self.client.get('/item/' + str(self.item.id) + '/delete/')
    self.assertEqual(response.status_code, 200)
    response = self.client.post('/item/' + str(self.item.id) + '/delete/', {})
    self.assertRedirects(response, '/user/' + self.user.username + '/')
    self.client.logout()

    # con utente autenticato ma non creatore
    self.client.login(**self.credential2)
    response = self.client.get('/item/' + str(self.item.id) + '/delete/')
    self.assertTemplateNotUsed(response, 'core/item_delete.html')
```

4.4 – Test reindirizzamento login (core)

Qui si verifica che prima di mettere in vendita un articolo, un utente sia loggato.

Qualora non lo fosse, si viene reindirizzati alla pagina di login altrimenti si passa al test (4.2) dove si controlla che un articolo venga inserito correttamente.

```
'''
Test per verificare se all'inserimento di un articolo, un utente non loggato
viene reindirizzato prima nella schermata di login.
'''

def test_login_required(self):
    response = self.client.get('/inserisci_articolo/')
    self.assertRedirects(response, '/accounts/login/?next=/inserisci_articolo/')
    # 302 --> FOUND: pagina esiste ma non puoi entrarci
    self.assertEqual(response.status_code, 302)
```

4.5 – Test visualizzazione profilo utente (core)

Ho testato il corretto reindirizzamento nella pagina del profilo di un utente.

Se l'utente è loggato e vuole visualizzare il suo profilo verrà reindirizzato nella sua pagina personale che utilizza uno specifico template; se invece l'utente vuole visualizzare la pagina

del profilo di un altro utente verrà reindirizzato in un' altra pagina ma che utilizza un template diverso.

```
'''
Test della visualizzazione del profilo di un utente autenticato
e per la visualizzazione del profilo di altri utenti.
'''
def test_userProfileView(self):
    # con utente autenticato
    # su il tuo profilo
    self.client.login(**self.credential)
    response = self.client.get('/user/' + self.user.username + '/')
    self.assertTemplateUsed(response, 'core/profilo.html')
    self.assertTrue(response.status_code, 200)

    # sul profilo degli altri
    response = self.client.get('/otheruser/' + self.user2.username + '/')
    self.assertTemplateNotUsed(response, 'core/profilo.html')
    self.assertTemplateUsed(response, 'core/user_profile.html')

    # con utente non autenticato
    self.client.logout()
    response = self.client.get('/otheruser/' + self.user.username + '/')
    self.assertTemplateUsed(response, 'core/user_profile.html')
    self.assertTrue(response.status_code, 200)
```

4.6 – Test cambio indirizzo utente (core)

Qui si testa il cambiamento di un indirizzo di un utente, in particolare si prova il corretto cambiamento dei campi di un indirizzo di un utente e soprattutto che questa azione sia fatta dall' utente creatore dell' indirizzo.

```
'''
Test per il cambio dell' indirizzo di un utente
'''
def test_address_change(self):
    # con utente autenticato
    self.client.login(**self.credential)
    response = self.client.get('/user/address/' + str(self.address.id) + '/modify/')
    self.assertEqual(response.status_code, 200)
    response = self.client.post('/user/address/' + str(self.address.id) + '/modify/', {})
    self.assertFormError(response, 'form', 'città', 'Questo campo è obbligatorio.')
    self.assertFormError(response, 'form', 'via', 'Questo campo è obbligatorio.')
    self.assertFormError(response, 'form', 'stato', 'Questo campo è obbligatorio.')
    self.assertFormError(response, 'form', 'cap', 'Questo campo è obbligatorio.')
    response = self.client.post('/user/address/' + str(self.address.id) + '/modify/',
                               {'città': 'chieti', 'via': 'strada', 'stato': 'it', 'cap': '85710'})
    self.assertRedirects(response, '/user/' + self.user.username + '/address_page/')
    self.client.logout()

    # con utente autenticato ma non creatore
    self.client.login(**self.credential2)
    response = self.client.get('/user/address/' + str(self.address.id) + '/modify/')
    self.assertTemplateUsed(response, 'core/homepage.html')
    self.assertTemplateNotUsed(response, 'accounts/address_change.html')
```

4.7 – Test eliminazione indirizzo utente (core)

Test per l'eliminazione dell'indirizzo di un utente che può essere eliminato solo dall'utente che lo ha creato.

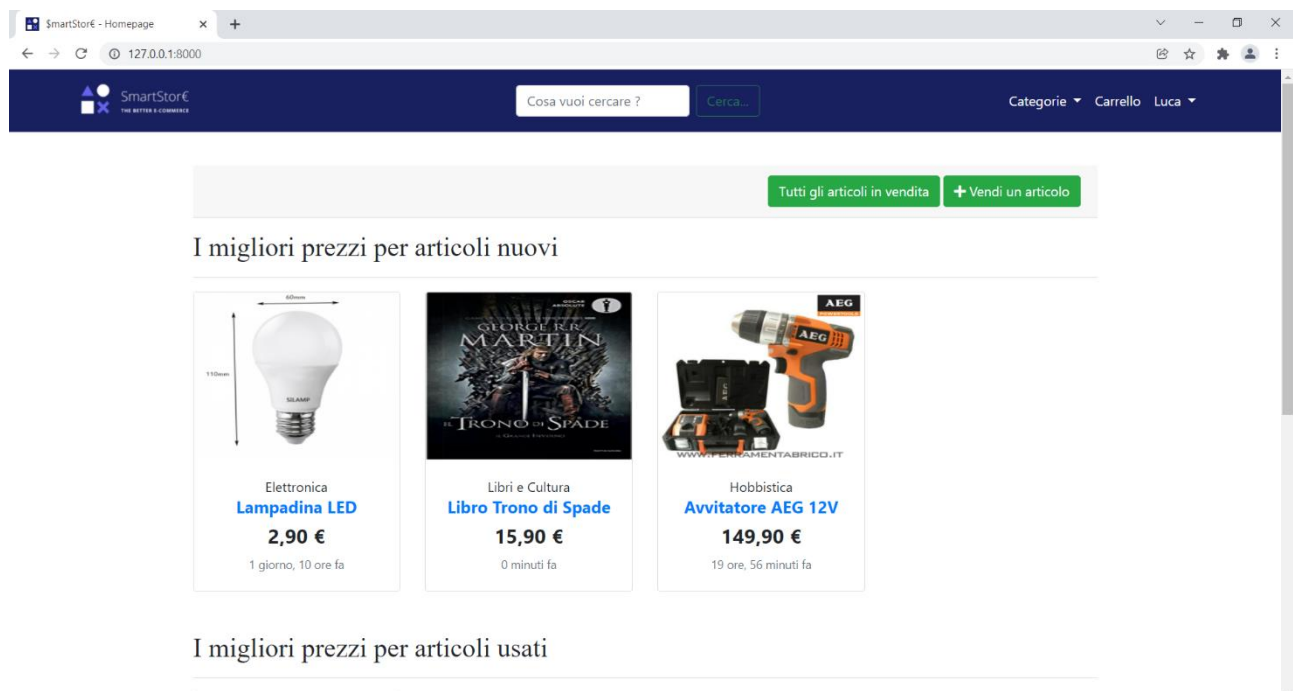
```
'''
Test per l'eliminazione di un indirizzo dell'utente
'''

def test_address_delete(self):
    # con utente autenticato
    self.client.login(**self.credential)
    response = self.client.get('/user/address/' + str(self.address.id) + '/delete/')
    self.assertEqual(response.status_code, 200)
    response = self.client.post('/user/address/' + str(self.address.id) + '/delete/', {})
    self.assertRedirects(response, '/user/' + self.user.username + '/address_page/')
    self.client.logout()

    # con utente autenticato ma non creatore
    self.client.login(**self.credential2)
    response = self.client.get('/user/address/' + str(self.address.id) + '/delete/')
    self.assertTemplateNotUsed(response, 'accounts/address_delete.html')
```

5 – Screenshot

5.1 - Homepage



5.2 – Profilo utente

SmartStore€ - Profilo di Luca

127.0.0.1:8000/user/Luca/


Cosa vuoi cercare? Cerca...

Categorie Carrello Luca



Luca, ecco il riepilogo del tuo profilo

[+ Vendi un articolo](#) [Indietro](#)


Libro Trono di Spade





Quantità :
Categoria : Libri e Cultura
Prezzo : 15,90 €

Avvitatore AEG 12V



Quantità :
Categoria : Hobbistica
Prezzo : 149,90 €

1

5.3 – Profilo di un utente visualizzato da un altro utente

SmartStore€ - Profilo di Luca

127.0.0.1:8000/user/Luca/


Cosa vuoi cercare? Cerca...

Categorie Carrello Federico

Articoli in vendita di Luca


[Indietro](#)

Libro Trono di Spade



Quantità :
Categoria : Libri e Cultura
Prezzo : 15,90 €

Avvitatore AEG 12V



Quantità :
Categoria : Hobbistica
Prezzo : 149,90 €

Hai bisogno di aiuto, informazioni o chiarimenti? [Contatta Luca](#)

1

5.4 – Visualizzazione del carrello

SmartStore€ - Carrello x TecnologieWeb_SmartStore_1310 x +


127.0.0.1:8000/order_summary/

SmartStore€ THE BETTER E-COMMERCE

Cosa vuoi cercare? Cerca...

Categorie Carrello Federico

Riepilogo del carrello

Nome	Categoria	Condizione	Quantità	Prezzo	Elimina
Libro Trono di Spade	Libri e Cultura	Nuovo	1	15,90 €	
Totale				15,90 €	

[Continua lo shopping](#) [Checkout](#)

5.5 – Riepilogo delle vendite di un utente

SmartStore€ - Vendite Federico x +

127.0.0.1:8000/user/Federico/vendite/

SmartStore€ THE BETTER E-COMMERCE

Cosa vuoi cercare? Cerca...

Categorie Carrello Federico

Federico, ecco il riepilogo delle tue vendite

[Grafico](#) [Indietro](#)

Nome	Prezzo di vendita	Acquirente	Indirizzo di spedizione	Data di vendita
Felpa Nike Rossa	45,90 €	Marco	86010 - Campobasso Polese, 45	Sabato 18 Dicembre 2021 23:13
Pianta con vaso	21,90 €	Nella	86019 - Baranello Strada, 1	Sabato 18 Dicembre 2021 11:53

[Indietro](#)

5.6 – Risultati di una ricerca

SmartStore€ - Cerca nel Sito x TecnologieWeb_SmartStore_1310 x +

127.0.0.1:8000/cerca/?kw=!

SmartStore€ THE BETTER E-COMMERCE


Cosa vuoi cercare? Cerca...

Categorie Carrello Federico


Risultati della ricerca

[Indietro](#)


Articoli




Libri e Cultura
Libro Python
35,90 €
15 minuti fa



Libri e Cultura
Libro Trono di Spade
15,90 €
20 minuti fa



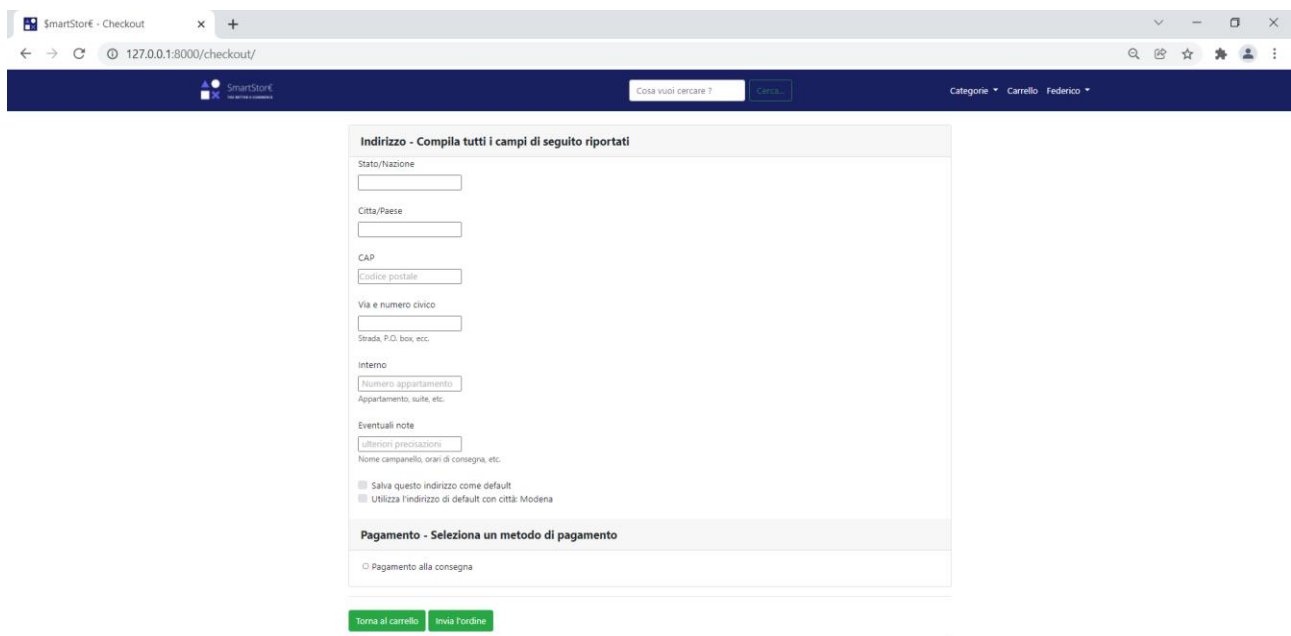
Elettronica
Lampadina LED
2,90 €
1 giorno, 10 ore fa



Hobbistica
Levigatrice da Banco
85,00 €
1 giorno, 22 ore fa

Utenti

5.7 – Pagina di Checkout



The screenshot shows a web browser window with the address bar displaying "SmartStore - Checkout" and the URL "127.0.0.1:8000/checkout/". The page header includes the SmartStore logo, a search bar with the placeholder "Cosa vuoi cercare?", and navigation links for "Categorie", "Carrello", and "Federico".

The main content area is divided into two sections:

- Indirizzo - Compila tutti i campi di seguito riportati**
 - Stato/Nazione:
 - Città/Paese:
 - CAP: (with a small "Codice postale" label)
 - Via e numero civico: (with a small "Strada, P.O. box, ecc." label)
 - Interno: (with a small "Numero appartamento" label)
 - Appartamento, suite, etc.:
 - Eventuali note: (with a small "Ulteriori precisazioni" label)
 - Nome campanello, orari di consegna, etc.:
 - ☐ Salva questo indirizzo come default
 - ☐ Utilizza l'indirizzo di default con città: Modena
- Pagamento - Seleziona un metodo di pagamento**
 - ☐ Pagamento alla consegna

At the bottom of the form, there are two green buttons: "Torna al carrello" and "Invia l'ordine".

6 – Conclusioni

Al progetto si potrebbero aggiungere anche altri aspetti, come una pagina relativa ai feedback con i commenti dell' acquirente sia sul servizio, in modo da valutare l' affidabilità di un venditore che sul prodotto, dalla quale gli utenti potranno farsi un' idea del funzionamento o se vale la pena acquistarlo, oltre anche ad una procedura di reso e rimborso.

In conclusione sono molto entusiasta di aver iniziato ad imparare molte nuove tecniche, metodi e framework per la creazione di questo sito web e sono molto soddisfatto del lavoro svolto.