AsteOnline

Violi Matteo

Indice

1 Traccia

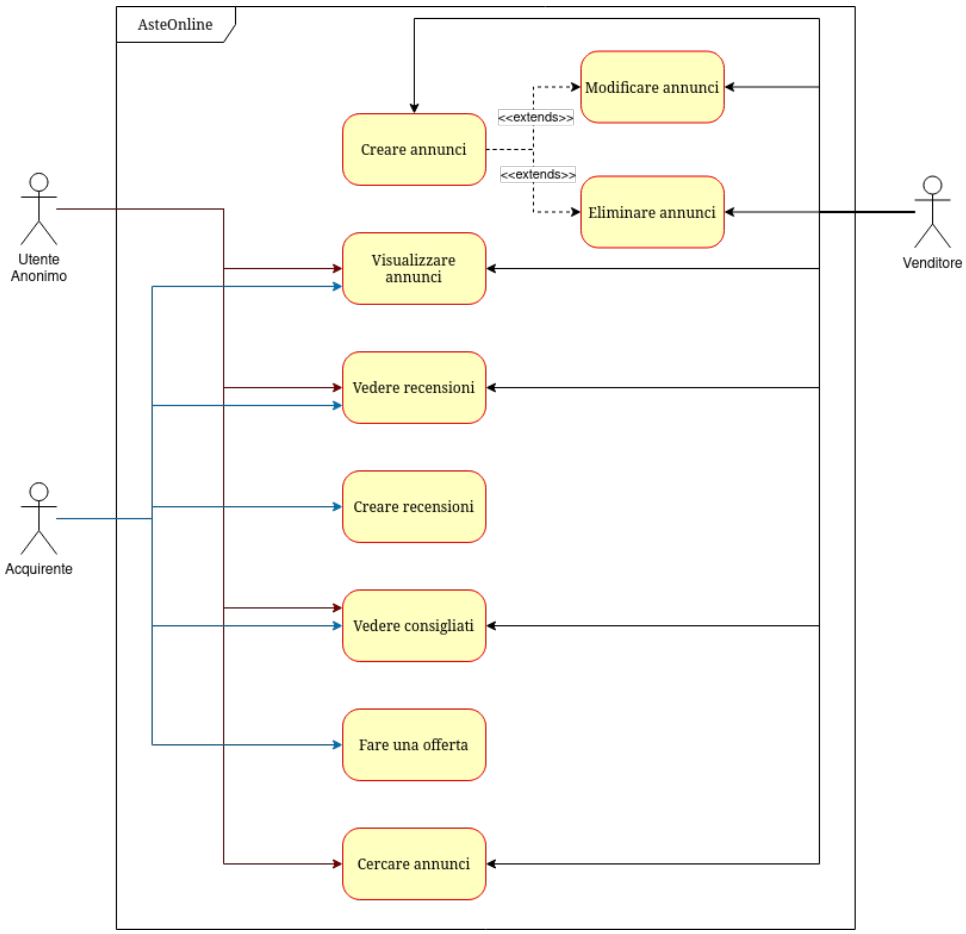
Portale di aste online
Creare un portale di aste online per la vendita di prodotti online in stile ebay, con piccole modifiche, che supporti i seguenti servizi:

- Gli utenti anonimi possono navigare il sito e visualizzare i prodotti disponibili e il prezzo temporaneo mentre il tempo rimanente è disponibile solo per quelli registrati.
- Ogni utente può registrarsi come venditore o come acquirente.
- Ogni utente registrato come fornitore deve poter caricare un nuovo prodotto, inserire una categoria, i dettagli (scheda tecnica), il prezzo di partenza ed una immagine (obbligatoria).
- Ogni utente registrato come acquirente può visualizzare le recensioni di altri utenti del venditore, confrontare prodotti simili (con schede tecniche simili), lasciare recensioni al venditore del prodotto.
- Ogni utente registrato come fornitore deve poter vedere informazioni sugli acquisti su tutti i prodotti che ha messo in vendita (statistiche sulle vendite).

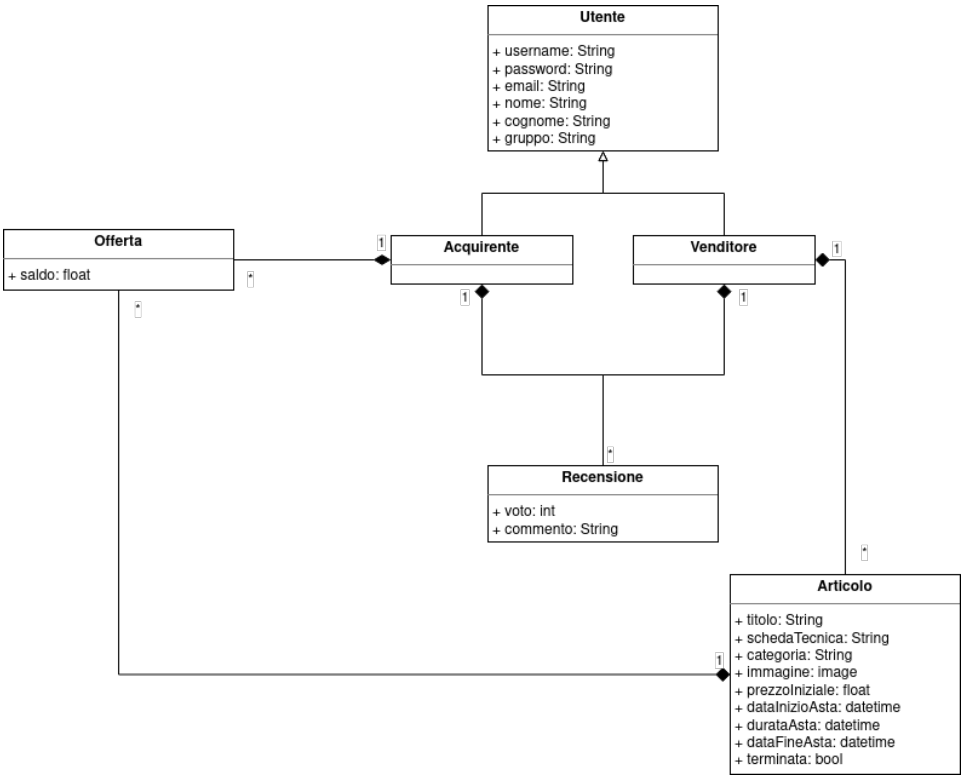
Dotare il sistema di un sistema di ricerca che permetta di selezionare i prodotti in base a diverse caratteristiche (prezzo, categoria e caratteristiche offerte, ...) e di un sistema di recommendation basato su prezzi, caratteristiche simili, suggerimenti del tipo 'chi ha comprato questo ha comprato anche ...' - combinazione di questi decisa dallo studente.

2 Descrizione del prodotto

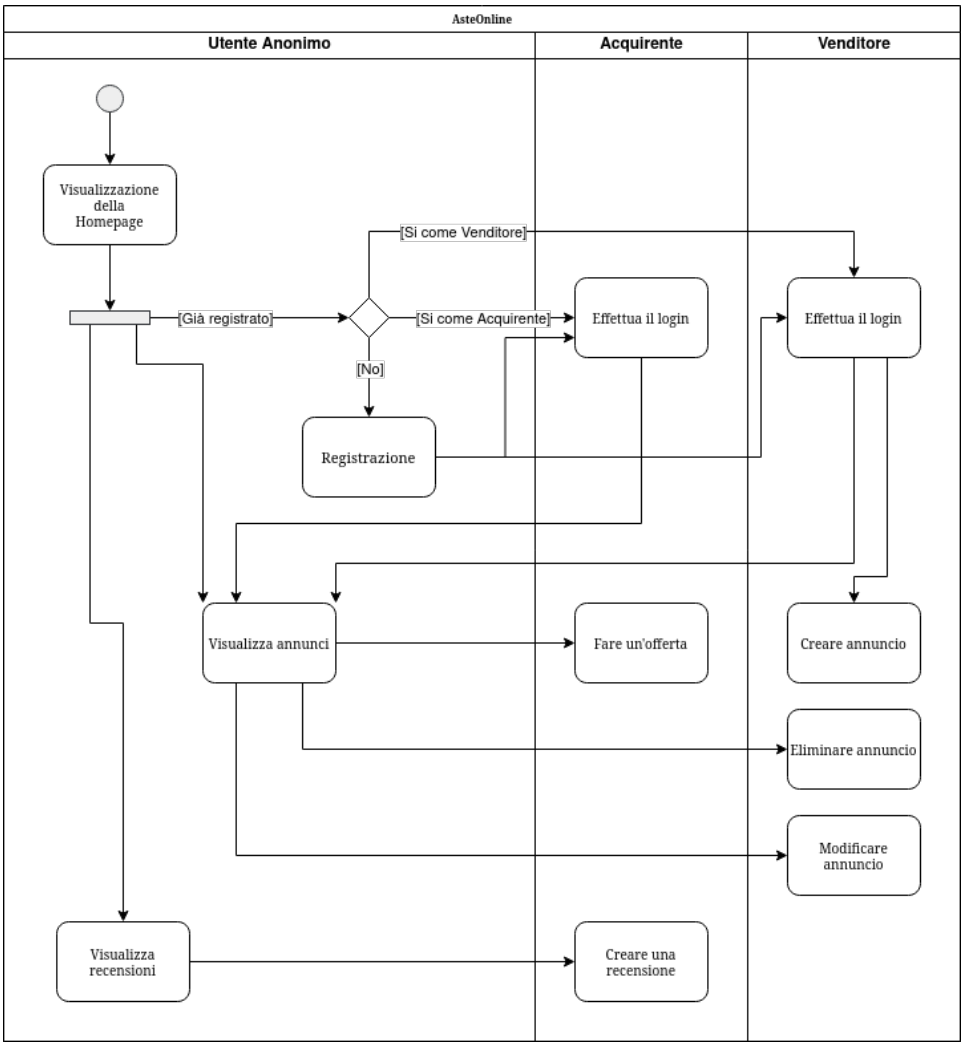
2.1 Diagramma Use Case UML



2.2 Diagramma Use Class UML



2.3 Diagramma delle attività



3 Tecnologie usate e motivazione

Le tecnologie utilizzate per questo progetto sono innanzitutto `python` che deve essere alla versione `3.10`. Altri pacchetti necessari da installare tramite pip sono:

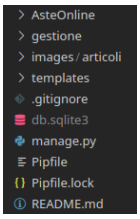
- `django`: che è il framework che ho utilizzato per costruire la piattaforma.
- `django-braces`: che fornisce delle View e dei Form astratti che permettono di implementare funzioni quale per esempio `LoginRequiredMixin` che permette di visualizzare la View solo agli utenti che hanno effettuato il login.

- `pillow`: è un pacchetto necessario per quanto riguarda l'upload di file in django.
- `docutils`: utilizzato per generare la documentazione per l'admin.

Infine è stato utilizzato Bootstrap 5 e Javascript per l'interfaccia grafica, la quale è stata presa in gran parte da gli esempi della documentazione ufficiale.

4 Organizzazione logica dell'applicazione

In questo progetto ogni View è class based, tipicamente ha un suo Form collegato e ha un template proprio.

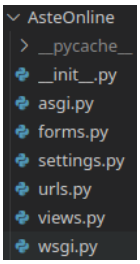
Questo è uno screenshot della cartella principale: 

Come possiamo vedere all'interno della cartella principale ci sono varie sotto cartelle e file utili:

- `AsteOnline`: contiene lo scheletro principale dell'applicazione.
- `gestione`: contiene l'applicazione `gestione` che si occupa di gestire la maggior parte della logica della piattaforma.
- `images/articoli`: contiene tutte le immagini inserite dai venditori per i loro annunci di vendita.
- `templates`: contiene i template delle View all'interno di AsteOnline

I file restanti sono il database, `manage.py` è necessario per il funzionamento dell'applicazione, `Pipfile` e `Pipfile.lock` che servono per l'ambiente virtuale, `.gitignore` per github e infine il file `README.md`.
Andando ad analizzare le varie sotto-cartelle e i file all'interno troviamo le seguenti informazioni:

4.1 AsteOnline

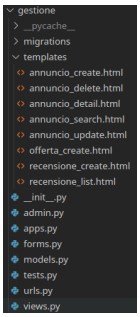


La cartella `AsteOnline` contiene lo scheletro della piattaforma, i file `__init__.py`, `asgi.py` e `wsgi.py` non sono stati modificati.

Mentre il file `settings.py` è stata aggiunta `gestione` tra le app installate e sono stati aggiunti i percorsi `MEDIA_URL` e `MEDIA_ROOT` per la gestione delle immagini caricate dagli utenti.
Nel file `urls.py` sono contenuti tutti i percorsi per le View implementate nel file `views.py` e anche l'url legato all'applicazione `gestione`.
Il file `forms.py` contiene i `VenditoreForm` e `AcquirenteForm` che in base al tipo di registrazione effettuata assegnano il gruppo. Nel file `views.py` abbiamo le seguenti View:

- `AcquirenteCreate` e `VenditoreCreate`: le due CreateView sono utilizzate per registrate gli utenti in base al loro ruolo.
- `UtenteCreate`: è una TemplateView che visualizza un template `preregistration.html` che è la pagina in cui decidi se registrarti come Venditore o come Acquirente.
- `HomeView`: è la View della homepage.
- `LogoutView`: è una View fatta per il logout, in quanto quella fornita da `auth_view` non si abbinava bene con il resto del progetto.

4.2 Gestione



La cartella `gestione` contiene due sotto-cartelle: `migrations`, che contiene le varie migrazioni dei Model, e `templates` che contiene i template delle View contenute le file `views.py`.
I file `__init__.py`, `admin.py` e `apps.py` non sono stati modificati.
Il file `tests.py` contiene i test e verrà trattato nella sezione apposita, `forms.py` contiene i Form per le View e hanno anche dei sistemi per controllare che non vengano inseriti caratteri speciali ed evitare attacchi di tipo **Injection**.
Nel file `models.py` abbiamo i seguenti model:

- `Articolo`: Articolo è il Model principale e tra gli attributi troviamo:
  - `venditore`: che è un CharField che contiene lo username del venditore, purtroppo non è di tipo ForeignKey perché durante lo sviluppo della piattaforma se il campo era ForeignKey non mi salvava l'entrata.
  - `titolo`: campo abbastanza autoesplicativo.

  - `schedaTecnica`: campo TextField per scrivere a piacimento la descrizione del prodotto.
  - `immagine`: è un ImageField, così da permettere agli utenti di caricare le immagini dei loro annunci, l'upload lo fa sulla cartella `images/articoli`.
  - `prezzoIniziale`: campo Decimal in cui ho assegnato un massimo di 7 cifre di cui solo due decimali.
  - `dataInizioAsta`: campo DateTimeField che viene inizializzato quando viene creato l'oggetto.
  - `durataAsta`: campo IntegerField che in base alla scelta del venditore cambierà la dataFineAsta.
  - `dataFineAsta`: campo DateTimeField che viene calcolato sommando durataAsta alla dataInizioAsta.
  - `terminato`: campo BooleanField che di default è negativo ma quando un utente visualizza l'annuncio scaduto diventa vero e l'annuncio non viene più visualizzato.

- `Offerta`: ha un campo ForeignKey che riguarda l'acquirente che la scrive, un altro campo ForeignKey che si collega all'articolo per cui viene fatta l'offerta e un saldo DecimalField guale al campo prezzoIniziale, però una volta salvato un nuova offerta vengono eliminate le offerte precedenti fatte allo stesso articolo.

- `Recensione`: ha due campi ForeignKey uno per il venditore e l'altro per l'acquirente, un TextField per esprimere il proprio commento verso il venditore e un voto che deve essere minimo 0 e massimo 5.

Il file `urls.py` contiene i vari indirizzi che portano alle varie View.
Nel file `views.py` abbiamo le seguenti View:

- `AnnuncioCreateView`: è una CreateView usata per aggiungere un annuncio nel database, prima però controlla che l'utente sia un `Venditore` e che il form sia valido.

- `AnnuncioDetailView`: è una DetailView che permette di visualizzare tutte le caratteristiche dell'articolo, restituisce anche gli articoli consigliati e il tempo che rimane prima della fine dell'asta.

- `AnnuncioUpdateView`: è una UpdateView usata per aggiornare delle caratteristiche di un articolo inserito, l'utente che la modifica deve essere il venditore stesso.

- `AnnuncioDeleteView`: è una DeleteView usata per eliminare un articolo, l'utente che la modifica deve essere il venditore stesso.

- `AnnuncioSearchView`: è una ListView che viene utilizzata per effettuare la ricerca di annunci tramite il titolo.
- `OffertaCreateView`: è una View che permette l'inserimento di una offerta per un prodotto, questa viene accettata se è maggiore dell'ultima e se è corretta.
- `RecensioneListView`: è una ListView che viene usata per vedere le recensioni di un venditore.
- `RecensioneCreateView`: è una CreateView usata per creare una recensione che poi viene attribuita ad un venditore.

5 Scelte effettuate

La ricerca di un articolo tramite la pagina apposita effettua una ricerca solo sul match del titolo, doveva essere tramite regex ma purtroppo SQLite non lo supporta.
Il recomendation system doveva basarsi anch'esso sulle regex ma per questioni software ho dovuto optare per il match tramite titolo oppure anche articoli della stessa categoria e con prezzi entro un range di 20$.

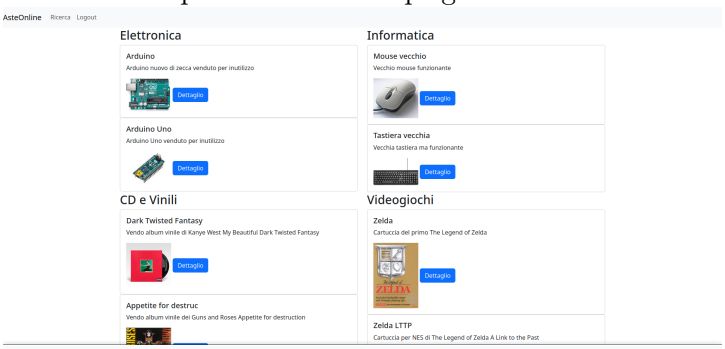
6 Test effettuati

I due test che possono essere effettuati sono:

- `AnnuncioDetailViewTestCase`: questo test crea nel database una tupla di un articolo pre-impostato, poi viene analizzata se `AnnuncioDetailView` viene visualizzata correttamente con tutte le informazioni precedentemente inserite.
- `ArticoloInputTestCase`: questo test cerca di effettuare una richiesta di tipo POST con degli input con caratteri speciali, che sono vietati, che restituirà il codice 302.

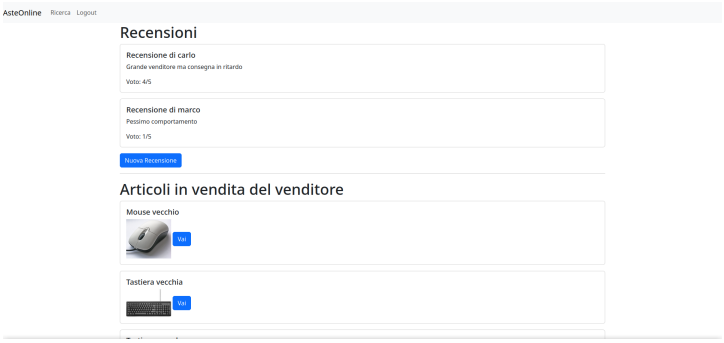
7 Risultati

Qua verranno proposti vari screenshot della piattaforma con spiegazione.

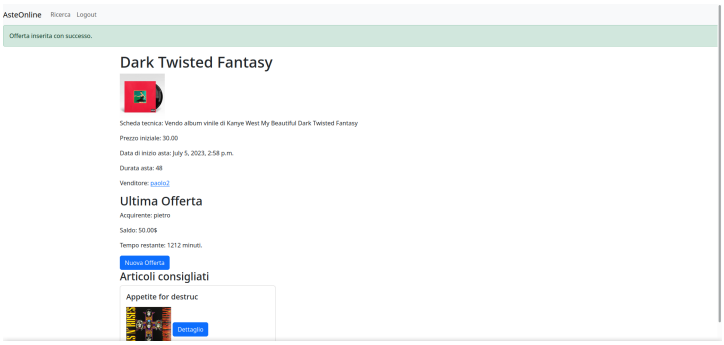


Questo è lo screenshot della homepage e come si può vedere c'è una barra di navigazione in alto che permette la navigazione, sotto troviamo i vari articoli divisi in base alla loro categoria.

Questo mostra il risultato di una ricerca basata sul titolo, si possono vedere i risultati della ricerca.



Qui viene mostrato il profilo del venditore in cui all'inizio ci sono le varie recensioni degli utenti e sotto gli articoli in vendita del suddetto venditore.



Qui infine viene mostrati tutti i dettagli dell'articolo ricercato, l'offerta (se c'è) più alta dell'utente e infine gli articoli consigliati.

8 Problemi riscontrati

Come detto precedentemente, uno dei problemi che ho affrontato è stata l'impossibilità di utilizzare le regex per i recomendation system. Un altro problema è stato implementare su consiglio del professore Capodieci un sistema di notifiche tramite email, purtroppo visto che la piattaforma veniva eseguita in locale non si poteva effettuare dei test sulla funzionalità e quindi è stata tolta.