

# **WEBCOMICS**

Web app per pubblicare e leggere comics

Alessandro Canossa - Matricola: 128434

2022

# Indice

<b>1</b>	<b>Requisiti</b>	<b>1</b>
1.1	Utenti . . . . .	1
1.2	Funzionalità . . . . .	1
<b>2</b>	<b>Descrizione del progetto</b>	<b>3</b>
2.1	Tipologie di utenti . . . . .	3
2.2	Database e modelli . . . . .	4
2.2.1	User . . . . .	5
2.2.2	CoinsPurchase . . . . .	5
2.2.3	Genre . . . . .	5
2.2.4	Comic . . . . .	6
2.2.5	Chapter . . . . .	6
2.2.6	ChapterImage . . . . .	6
2.2.7	BuyList . . . . .	7
2.2.8	Rating . . . . .	7
2.2.9	Like . . . . .	7
2.2.10	Library . . . . .	8
2.2.11	ReadHistory . . . . .	8
2.2.12	Comment . . . . .	8
2.2.13	Modelli app comics . . . . .	9
2.2.14	Modelli app users . . . . .	10
2.2.15	Modelli app comment . . . . .	11
2.2.16	Schema completo . . . . .	12
2.3	Passaggi logici . . . . .	13
<b>3</b>	<b>Tecnologie usate</b>	<b>16</b>
3.1	Backend . . . . .	16
3.2	Frontend . . . . .	16
<b>4</b>	<b>Organizzazione del software</b>	<b>17</b>
4.1	users . . . . .	17
4.2	comics . . . . .	17
4.3	comments . . . . .	17
<b>5</b>	<b>Scelte effettuate</b>	<b>18</b>
5.1	Viste . . . . .	18
5.2	AJAX . . . . .	18
5.3	AUTH . . . . .	18
5.4	Pannello admin . . . . .	18
5.5	Immagini . . . . .	18

<b>6</b>	<b>Test eseguiti</b>	<b>19</b>
6.1	userManager . . . . .	19
6.1.1	test_create_user . . . . .	19
6.1.2	test_create_superuser . . . . .	19
6.2	UserModel . . . . .	19
6.2.1	setUp . . . . .	19
6.2.2	test_login . . . . .	19
6.2.3	test_login_invalid_credentials . . . . .	19
6.2.4	test_create_existing_user . . . . .	19
6.2.5	test_create_user_with_email_invalid . . . . .	19
6.2.6	test_create_user_with_username_invalid . . . . .	19
6.2.7	test_create_user_with_email_and_username_invalid . . . . .	20
6.2.8	test_create_user_with_password_invalid . . . . .	20
6.3	UserViews . . . . .	20
6.3.1	setUp . . . . .	20
6.3.2	test_logout . . . . .	20
6.3.3	test_change_username . . . . .	20
6.3.4	test_change_username_invalid_username . . . . .	20
6.3.5	test_change_username_invalid_username_length . . . . .	20
6.3.6	test_change_password . . . . .	20
6.3.7	test_change_password_invalid_old_password . . . . .	20
6.3.8	test_buy_coins . . . . .	21
6.3.9	test_buy_coins_invalid_amount . . . . .	21
6.3.10	test_buy_coins_invalid_amount_string . . . . .	21
6.3.11	test_buy_coins_invalid_amount_zero . . . . .	21
6.3.12	test_buy_coins_not_logged . . . . .	21
6.3.13	test_buy_coins_entry_created . . . . .	21
6.3.14	test_buy_coins_entry_created_invalid_amount . . . . .	21
6.3.15	test_become_creator . . . . .	21
6.3.16	test_become_creator_not_logged . . . . .	21
6.3.17	test_become_creator_already_creator . . . . .	22
<b>7</b>	<b>Risultati ottenuti</b>	<b>23</b>
7.1	Homepage . . . . .	23
7.2	Series . . . . .	23
7.3	Comic . . . . .	24
7.4	Chapter . . . . .	24
7.5	Login . . . . .	25
7.6	Register . . . . .	25
7.7	User Settings . . . . .	26
7.8	Market . . . . .	26
7.9	My comics . . . . .	27
7.10	Add comic . . . . .	27

7.11 Add chapter . . . . .	28
<b>8 Problemi riscontrati</b>	<b>29</b>

# 1 Requisiti

Il progetto ha lo scopo di realizzare una web app per la pubblicazione di comics da parte di qualsiasi utente, e permetterne la lettura agli altri utenti a fronte di un pagamento con una moneta virtuale (coins).

Di conseguenza i requisiti dell'applicazione sono:

## 1.1 Utenti

- **Anonimo:** Può visualizzare i comics presenti ed effettuare ricerche di vari comics. Può creare un account registrandosi.
- **Utente:** Può comprare *coins* e acquistare i capitoli dei comics che vuole leggere. Relativamente ai comics può dare un voto da 1 a 10 e salvarlo nella libreria. Relativamente ai capitoli: può acquistarli, leggerli, mettere like, commentare e rispondere ai commenti. Infine può fare richiesta di diventare Creatore.
- **Creatore:** può creare nuovi comic e caricare i relativi capitoli. Da una pagina apposta può controllare le informazioni dei propri comics, come voto, salvataggi in libreria e visite totali. Infine può aggiornare lo stato dei comics.

## 1.2 Funzionalità

- **Creazione di comics:**  
Durante la creazione di un comics bisogna fornire:
  - Immagine di copertina
  - Titolo
  - Genere (1 o più)
  - Sinossi
- **Aggiunta di un capitolo:**  
Una volta scelto il comic per cui aggiungere il capitolo, bisogna fornire:
  - Numero del capitolo
  - Immagini del capitolo (1 o più), con i titoli ordinati in senso di lettura.
- **Commenti:**  
Un utente può scrivere commenti sotto i vari capitoli, eliminarli, oppure rispondere a commenti di altri utenti. Nel profilo personale è visualizzabile l'elenco di tutti i commenti di un utente.
- **Libreria:**  
Un utente può aggiungere un comic alla propria libreria, rimuoverlo e visualizzare l'intera libreria.

- **Cronologia di lettura:**

Un utente può visualizzare e modificare la propria cronologia di lettura.

- **Rating di un comic:**

Un utente può assegnare un voto da 1 a 10 ad un comic. Il voto può anche essere modificato oppure rimosso.

- **Like ai capitoli:**

Un utente che ha comprato tale capito può aggiungere o togliere un like ai capitoli che ha comprato.

- **Coins:**

Nella pagina del market un utente può comprare una quantità di coins a sua scelta.

- **Registrazione e login:**

Chiunque può creare un profilo fornendo: username, nome, cognome, email e password.

- **Diventare creatori:**

Qualunque utente può diventare creatore mediante un pulsante nelle impostazioni del suo profilo.

- **Visualizzazione comic:**

Nella homepage vengono mostrati i comics di cui sono stati aggiunti capitoli recentemente. Nella sezione “serie” è possibile filtrare i comics in questi modi:

- Selezionando uno o più generi di comic
- Selezionando uno o più stati di pubblicazione
- Per nome, tramite barra di ricerca

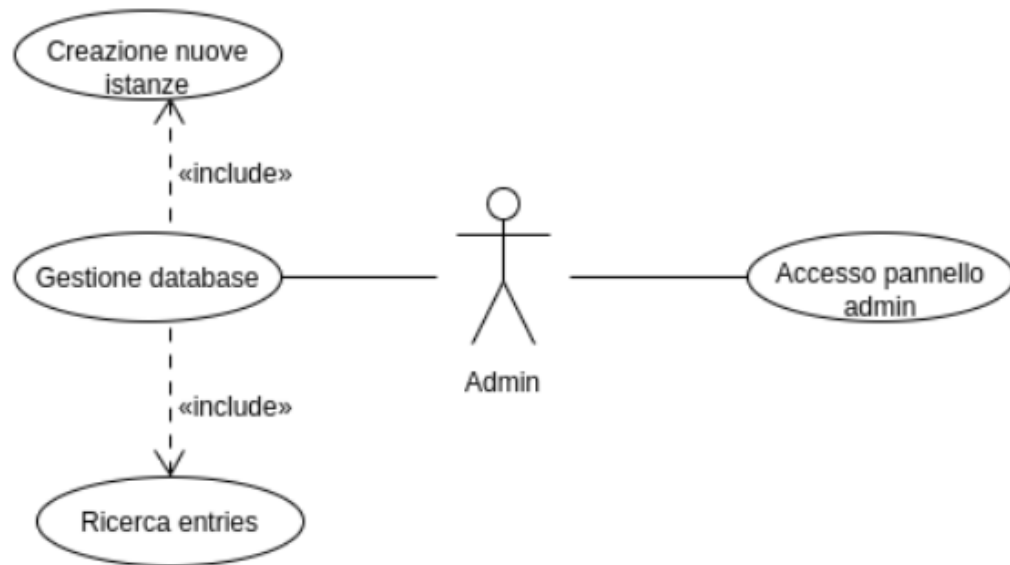
Inoltre i risultati possono essere ordinati per:

- Rating
- In ordine alfabetico
- Numero di aggiunte in libreria

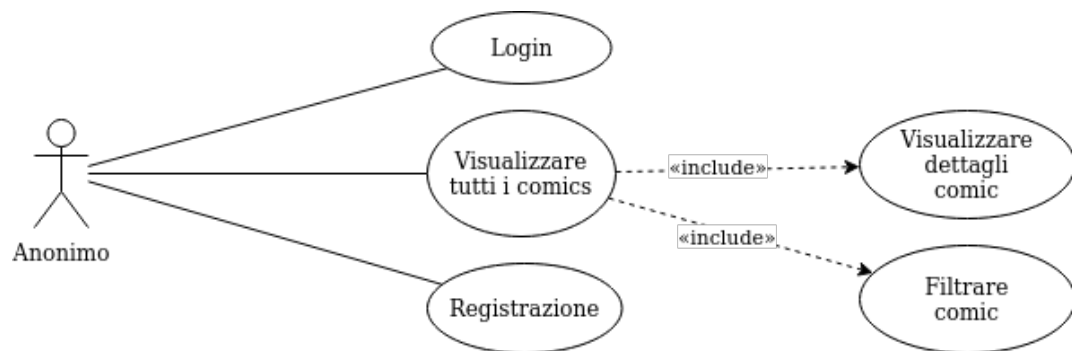
## 2 Descrizione del progetto

### 2.1 Tipologie di utenti

- **Admin:** ha la capacità di accedere al pannello amministrativo e creare, modificare o eliminare tutti i tipi di oggetti all'interno del database, può inoltre effettuare ricerche sulle entries del database e filtrare i risultati.



- **Anonimo:** un utente non registrato ha la possibilità di effettuare ricerche e vedere tutti i comics disponibili. Può vedere la pagina dettagliata di un comic, ma non può accedere ai vari capitoli. Infine può creare un account.

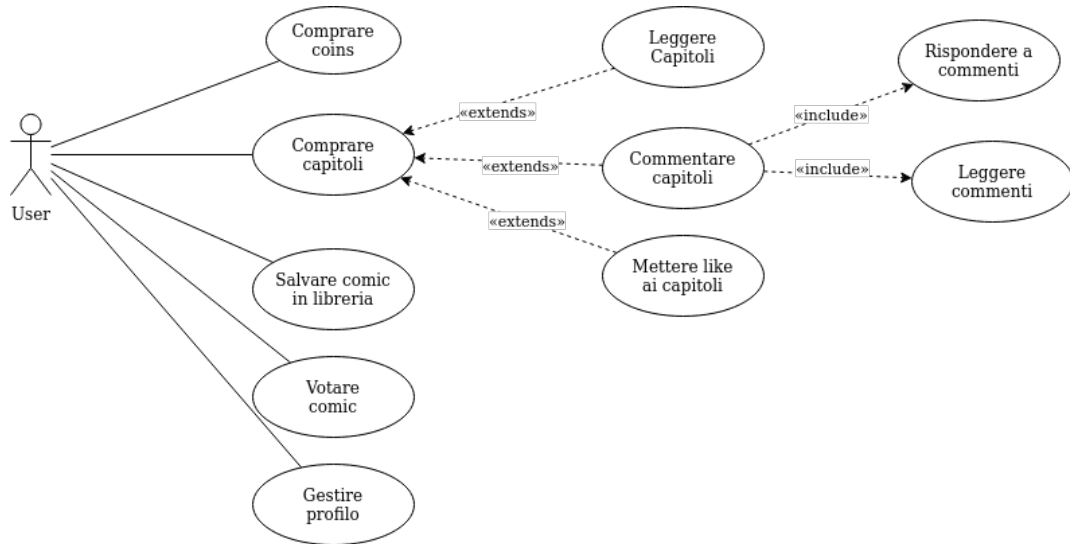


- **Utente:** un utente registrato mantiene tutte le capacità di un utente anonimo. L'utente può assegnare un voto da 1 a 10 ad un comic, modificare il suo voto oppure rimuoverlo. Può anche salvare il comic nella propria libreria. In più ha la possibilità di acquistare *coins* che sono una moneta che può essere utilizzata per acquistare i vari capitoli di un comic.

Una volta acquistato un capitolo, l'utente può leggerlo e mettere un *like*, può lasciare commenti sotto di esso e anche rispondere ad altri commenti.

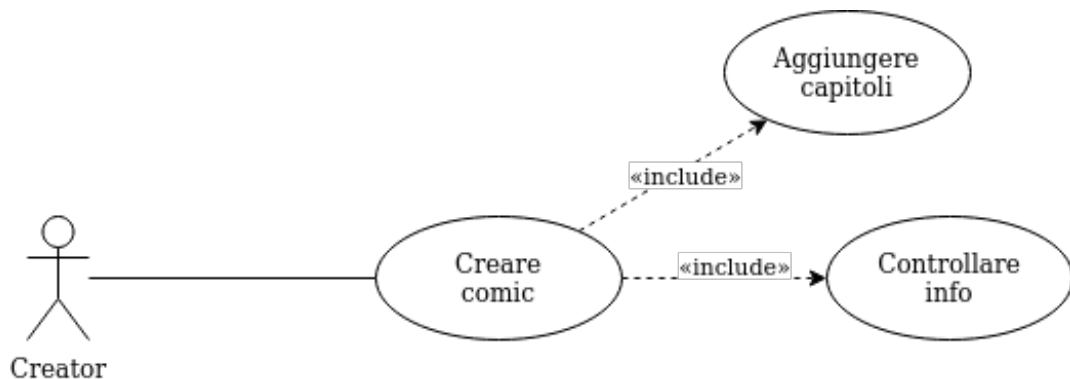
Nella pagina riservata all'utente, questo può controllare i propri commenti, i capitoli acquistati, i commenti lasciati, i capitoli letti e la propria libreria.

Infine può modificare i propri dati personali, e può anche diventare **Creator** facendo richiesta.



- **Creator:** un utente con questo ruolo ha la possibilità di creare nuovi comics e caricarne i relativi capitoli.

Dalla propria pagina utente, può vedere i comics creati e le varie statistiche relative ai vari comics.



## 2.2 Database e modelli

Il database utilizzato per lo sviluppo del progetto è stato **Sqlite**, scelto per la sua leggerezza e facilità di utilizzo. Nulla impedisce però di potersi spostare ad altri database in caso di nuovi futuri requisiti.

I modelli (tabelle) utilizzati all'interno del database sono stati:



### 2.2.1 User

Modello usato per gestire le tipologie di utenti che possono utilizzare il sito. Per la realizzazione è stato esteso il modello `AbstractUser`, presente nell'insieme di classi messe a disposizione dal framework Django, in modo da avere già diversi campi disponibili per l'uso.

User <AbstractUser>	
<b>id</b>	<b>BigAutoField</b>
coins	IntegerField
<i>date_joined</i>	<i>DateTimeField</i>
email	EmailField
<i>first_name</i>	<i>CharField</i>
<i>is_active</i>	<i>BooleanField</i>
<i>is_creator</i>	<i>BooleanField</i>
<i>is_staff</i>	<i>BooleanField</i>
<i>is_superuser</i>	<i>BooleanField</i>
<i>last_login</i>	<i>DateTimeField</i>
<i>last_name</i>	<i>CharField</i>
<i>password</i>	<i>CharField</i>
<i>username</i>	<i>CharField</i>

### 2.2.2 CoinsPurchase

Modello usato per registrare le transazioni di acquisto di monete di un utente. Ha utente come chiave esterna per identificare l'utente che ha effettuato la transazione.

CoinsPurchase	
<b>id</b>	<b>BigAutoField</b>
<b>user</b>	<b>ForeignKey (id)</b>
coins	PositiveIntegerField
date	DateTimeField

### 2.2.3 Genre

Modello usato per gestire i generi dei comics.

È stato scelto di utilizzare una tabella a se stante in vista di eventuali inserimenti di nuovi generi.

Genre	
<b>id</b>	<b>BigAutoField</b>
name	CharField

#### 2.2.4 Comic

Modello usato per gestire i dati di uno specifico comic. Ha utente come chiave esterna per identificare l'utente che ha caricato il comic. I campi follows, rating e watches vengono aggiornati in automatico. Le immagini delle cover vengono salvate in una cartella che viene creata in caso non esista.

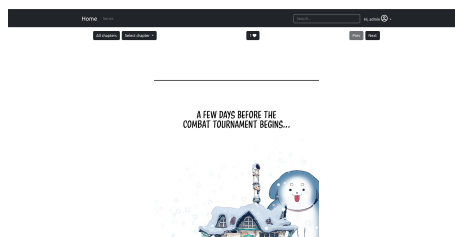
La tabella **Comic** ha una relazione Many-to-Many con la tabella **Genre**, siccome un comic può avere più generi.

Comic	
<b>id</b>	<b>BigAutoField</b>
<b>creator</b>	<b>ForeignKey (id)</b>
cover	DynamicImageField
follows	PositiveBigIntegerField
rating	FloatField
status	CharField
summary	TextField
title	CharField
watches	PositiveBigIntegerField

#### 2.2.5 Chapter

Modello usato per gestire i dati di un capitolo di uno specifico comic. Anche in questo caso il campo like viene aggiornato in automatico.

La colonna **chapter-num** indica il numero del capitolo relativamente al comic.



#### 2.2.6 ChapterImage

Modello usato per gestire le immagini di un capitolo di uno specifico comic. È stato utilizzato un modello a se stante siccome il numero di immagini inseribili non è stato limitato.

ChapterImage	
<b>id</b>	<b>BigAutoField</b>
<b>chapter</b>	<b>ForeignKey (id)</b>
image	DynamicImageField

### 2.2.7 BuyList

Modello usato per gestire i capitoli acquistati da ogni utente. Il modello è utile per consentire l'accesso ad un capitolo ai soli utenti che lo hanno acquistato.

BuyList	
<b>id</b>	<b>BigAutoField</b>
<b>chapter</b>	<b>ForeignKey (id)</b>
<b>user</b>	<b>ForeignKey (id)</b>
date	DateField

### 2.2.8 Rating

Modello usato per gestire i voti di un utente ad un determinato comic. L'utilità del modello sta nel fatto che permette la modifica e la rimozione del voto da parte di un utente.

Rating	
<b>id</b>	<b>BigAutoField</b>
<b>comic</b>	<b>ForeignKey (id)</b>
<b>user</b>	<b>ForeignKey (id)</b>
rating	PositiveIntegerField

### 2.2.9 Like

Modello usato per gestire i like di un utente ad un determinato capitolo. Come per il Rating, questo modello permette l'aggiunta e la rimozione di likes.

Like	
<b>id</b>	<b>BigAutoField</b>
<b>chapter</b>	<b>ForeignKey (id)</b>
<b>user</b>	<b>ForeignKey (id)</b>

### 2.2.10 Library

Modello usato per gestire la libreria di comic di un utente.

Library	
id	BigAutoField
comic	ForeignKey (id)
user	ForeignKey (id)

### 2.2.11 ReadHistory


Modello usato per gestire la cronologia di lettura di un utente.

ReadHistory	
id	BigAutoField
chapter	ForeignKey (id)
user	ForeignKey (id)
date	DateTimeField

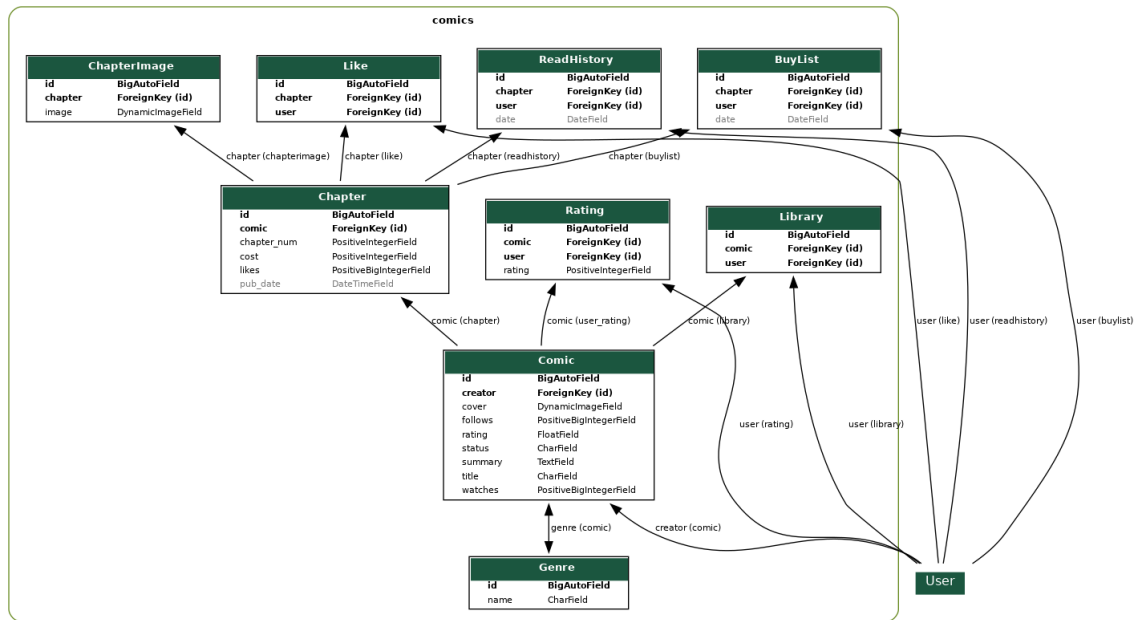
### 2.2.12 Comment

Modello usato per gestire i commenti di un utente ad un determinato capitolo. Questo modello ha il campo **reply-to** che permette di rispondere ad altri commenti.

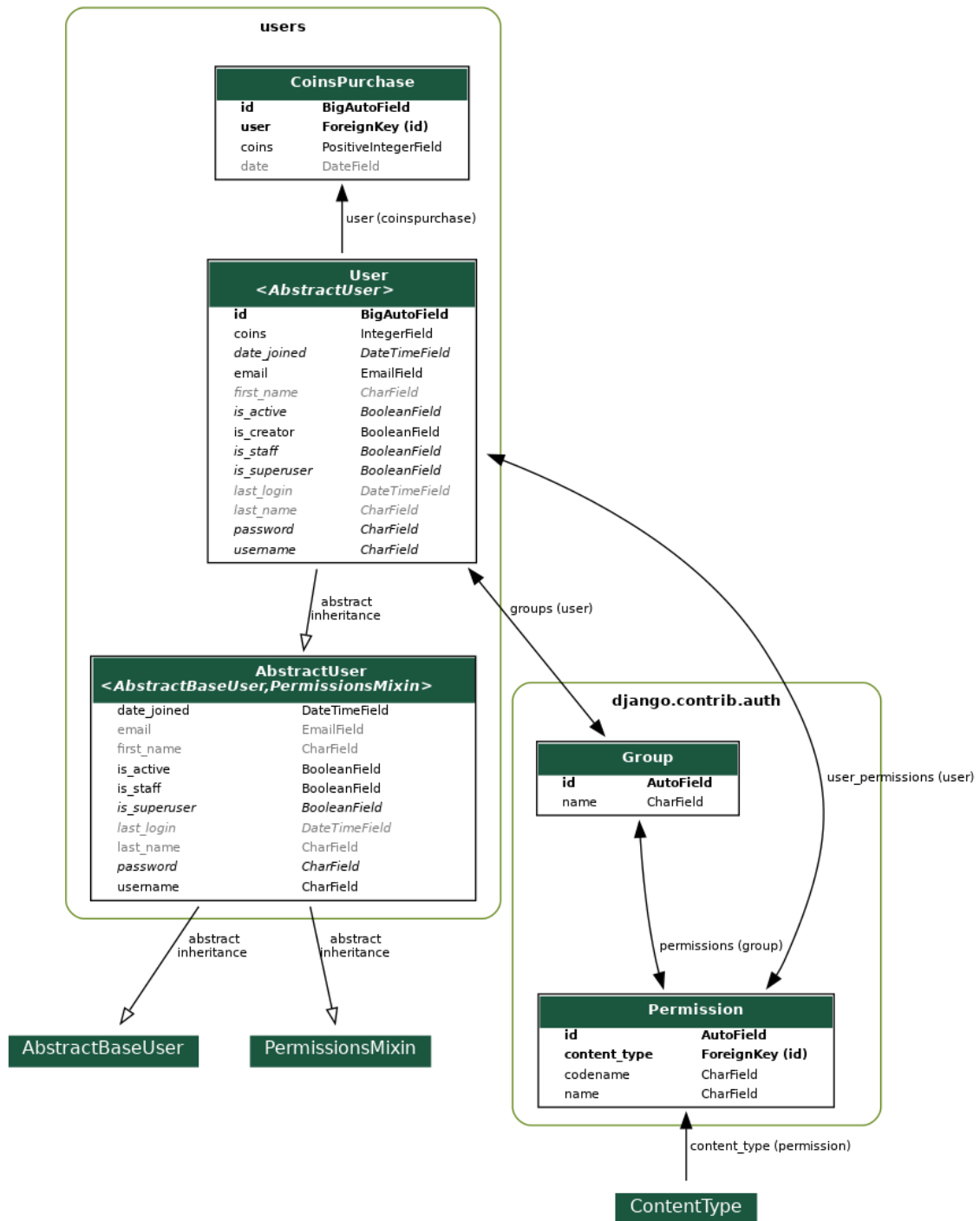
Comment	
id	BigAutoField
chapter	ForeignKey (id)
reply	ForeignKey (id)
user	ForeignKey (id)
body	TextField
created_on	DateTimeField



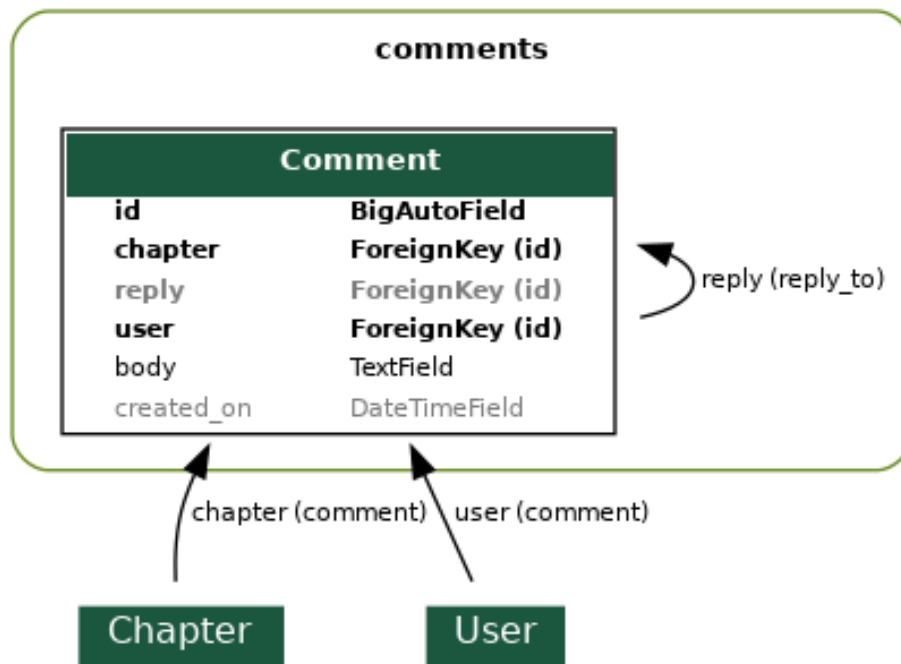
## 2.2.13 Modelli app comics



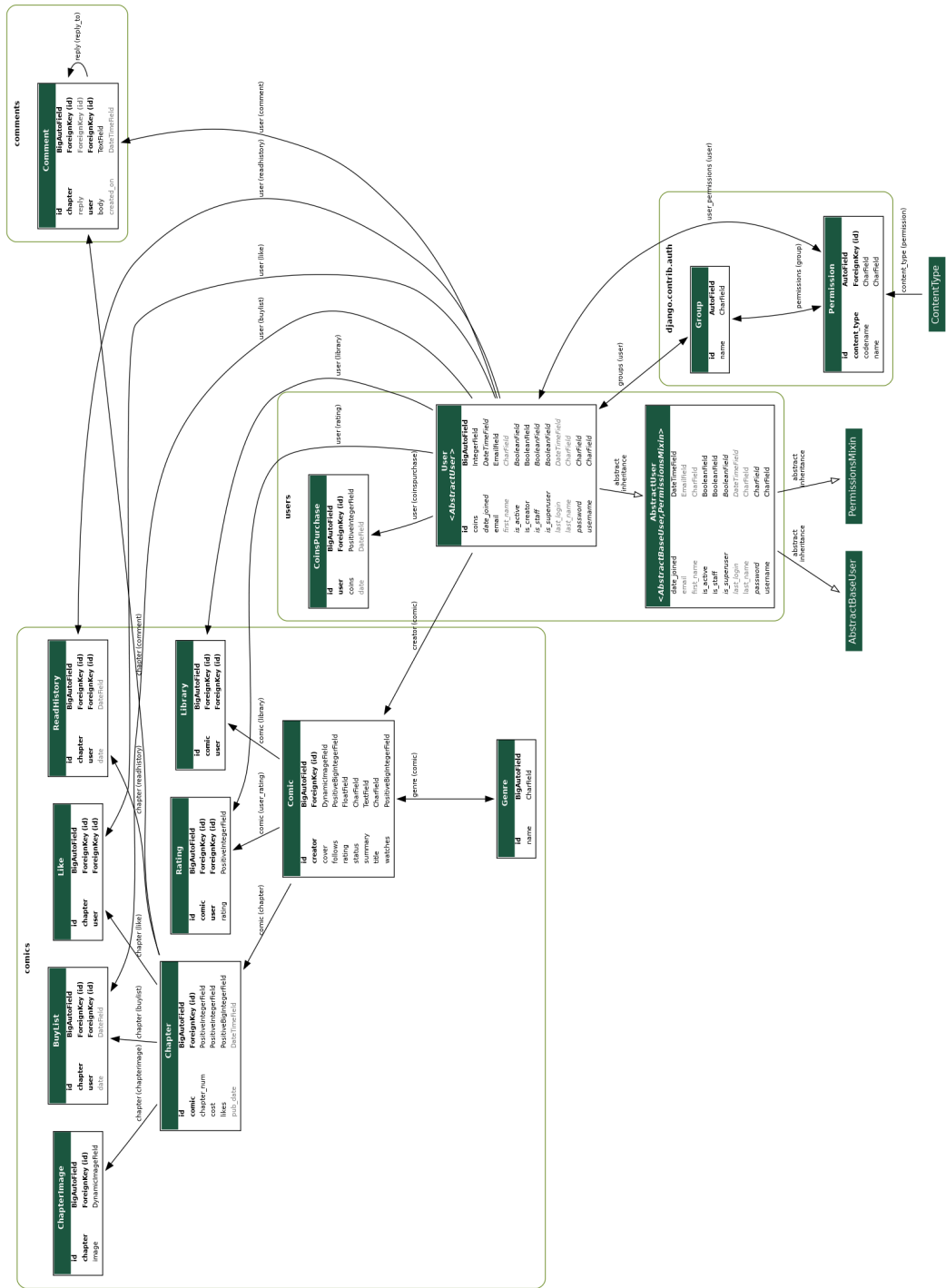
## 2.2.14 Modelli app users



### 2.2.15 Modelli app comment



## 2.2.16 Schema completo





## 2.3 Passaggi logici

- **Registrazione:**

1. L'utente entra nel sito.
2. Preme il tasto **Sign up** nella navbar.
3. Compila il form di registrazione e preme il tasto **Create account**.

- **Login:**

1. L'utente entra nel sito.
2. Preme il tasto **Sign in** nella barra di navigazione.
3. Compila il form di login e preme il tasto **Sign in**.

- **Logout:**

1. L'utente ha effettuato il login.
2. Preme sull'icona utente nella barra di navigazione.
3. Nel menù a tendina preme **Sign out**.

- **Cambio password:**

1. L'utente ha effettuato il login.
2. Preme sull'icona utente nella barra di navigazione.
3. Nel menù a tendina preme **User settings**.
4. Nella barra di navigazione preme il tasto **settings**.
5. Compila il form di cambio password e preme il tasto **Submit**.

- **Cambio username:**

1. L'utente ha effettuato il login.
2. Preme sull'icona utente nella barra di navigazione.
3. Nel menù a tendina preme **User settings**.
4. Nella barra di navigazione preme il tasto **settings**.
5. Compila il form di cambio username e preme il tasto **Submit**.

- **Comprare coins:**

1. L'utente ha effettuato il login.
2. Preme sull'icona utente nella barra di navigazione.
3. Nel menù a tendina preme sull'icona **coins**.

4. Inserisce la quantità di coins da comprare e preme il tasto **Buy**.

- **Comprare un capitolo:**

1. L'utente ha effettuato il login.
2. Preme sul comic desiderato.
3. Nella pagina del comic, preme sul capitolo desiderato.
4. Nel popup che appare preme il pulsante **Buy**.

- **Dare un voto a un comic:**

1. L'utente ha effettuato il login.
2. Preme sul comic desiderato.
3. Nella pagina del comic, preme sul tasto con una icona di una stella.
4. Nel menù a tendina che compare preme sul voto scelto.

- **Aggiungere un comic alla libreria:**

1. L'utente ha effettuato il login.
2. Preme sul comic desiderato.
3. Nella pagina del comic, preme sul tasto **Bookmark**.

- **Scrivere un commento:**

1. L'utente ha effettuato il login.
2. Preme sul comic desiderato.
3. Nella pagina del comic, preme sul capitolo desiderato.
4. Scorre la pagina fino alla sezione commenti.
5. Compila il form di commento e preme il tasto **Post**.

- **Cercare un comic:**

1. Inserisce il nome del comic nella barra di ricerca.
2. Preme il tasto **Invio** sulla tastiera.

- **Diventare Creator:**

1. L'utente ha effettuato il login.
2. Preme sull'icona utente nella barra di navigazione.
3. Nel menù a tendina preme **User settings**.
4. Nella pagina utente navigare fino alla sezione **Settings**.

5. Premere il tasto **Become Creator**.

- **Creare un nuovo comic:**

1. L'utente ha effettuato il login.
2. Preme sull'icona utente nella barra di navigazione.
3. Nel menù a tendina preme **user settings**.
4. Nella pagina utente navigare fino alla sezione **My comics**.
5. Premere il pulsante **Add new comic**.
6. Compilare il form di creazione del comic e preme il tasto **Add comic**.

- **Creare un nuovo capitolo:**

1. L'utente ha effettuato il login.
2. Preme sull'icona utente nella barra di navigazione.
3. Nel menù a tendina preme **user settings**.
4. Nella pagina utente navigare fino alla sezione **My comics**.
5. Premere il pulsante **Add chapter** sul comic desiderato.
6. Compilare il form di creazione del capitolo e preme il tasto **Add new chapter**.

## 3 Tecnologie usate

### 3.1 Backend

Per il backend della web app è stato utilizzato il framework web visto a lezione, ovvero Django. Si è scelto di adottare questo framework perché permette una grande velocità di sviluppo una volta che si è capito il meccanismo di funzionamento. Permette anche di astrarre tutte le complessità legate alla gestione di un database e alla creazione dinamica dei templates delle pagine con i dati ottenuti dal database.

Per la maggior parte delle viste è stato utilizzato il template engine di Django. Ogniuna di tali viste è legata ad un template e gestisce la logica di ottenimento di quello che deve essere fatto vedere a schermo e/o la modifica di questi oggetti all'interno del database.

Sono state però create anche delle viste che forniscono dei più semplici servizi, come per esempio l'aggiunta di un voto ad un comic, le quali restituiscono solo messaggi di successo o di errore.

Come database è stato utilizzato **SQLite3**, siccome è molto leggero, l'app non necessita di funzioni di gestione del database particolari e anche perché è il database che viene fornito di base da Django.

### 3.2 Frontend

Per la realizzazione del Frontend sono stati usati i template di Django, che permettono di creare pagine dinamiche con i dati ottenuti dal database. Inoltre forniscono anche la possibilità di creare componenti HTML riutilizzabili e la possibilità di estendere pagine con altri template mediante l'utilizzo dei blocchi.

Come libreria grafica è stata utilizzata **Bootstrap**, per la facilità di creazione degli oggetti. Per la dinamicità delle pagine è stato usato anche **Javascript**, e in particolare **AJAX**, per la gestione di bottoni e piccoli form, ad esempio per il tasto di voto.

## 4 Organizzazione del software

Il progetto è stato diviso in tre app:

### 4.1 users

Applicazione in cui sono gestiti tutti i dettagli di un utente all'interno del sito; qui è salvato il modello che rappresenta un utente, che estende quello base di Django aggiungendo maggiori dettagli, e anche il modello che rappresenta gli acquisti di *coins*. Inoltre in questa applicazione sono state scritte tutte le views che permettono la creazione e la modifica di un utente, oltre che la gestione del login, del logout e del cambio della password.

### 4.2 comics

In questa app sono presenti tutti i modelli che riguardano i Comic nella loro interezza, come il modello *Comic*, che rappresenta un singolo Comic, e il modello *Chapter* che rappresenta un singolo capitolo di un Comic, ma anche modelli come *Library* che rappresenta la libreria di Comic di un utente.

L'applicazione contiene anche tutta la logica per la creazione dei comic e dei capitoli, per la ricerca con filtri dei comic, per la gestione dei voti, per la gestione dei comic in libreria, per l'acquisto dei capitoli e la loro visualizzazione.

### 4.3 comments

Questa applicazione si occupa di gestire i soli commenti. È stata tenuta separata dalle altre due app siccome in questo modo si ha la possibilità di usarla in altri progetti andando a modificare solo poche cose.

## 5 Scelte effettuate

### 5.1 Viste

Le viste sono state realizzate mediante funzioni, invece che classi, per la possibilità di personalizzazione di funzionamento, a discapito del possibile utilizzo di mixins. Questo ha portato a dover effettuare spesso controlli sui permessi degli utenti.

### 5.2 AJAX

L'utilizzo di AJAX è stato scelto per la possibilità di gestione dinamica di pagine altrimenti statiche. Principalmente è stato utilizzato per bottoni e piccoli form che non avrebbe avuto senso gestire con pagine separate.

### 5.3 AUTH

L'utilizzo di django auth è stato limitato al solo login e logout, per evitare di andare a creare pagine superflue, come quella di conferma cambio password.

Per le notifiche di conferma cambio password e altre notifiche simili sono stati utilizzati gli *alert* di Javascript in combo con AJAX.

### 5.4 Pannello admin

Il pannello di gestione dell'admin è stato esteso con funzioni di ricerca e filtraggio per ogni modello presente. Questo permette una ricerca più efficiente e una visualizzazione più compatta.

### 5.5 Immagini

La gestione delle immagini inizialmente si è rivelata abbastanza complicata, ma grazie al pacchetto *django-dynamic-image* si è potuto creare dinamicamente i path di salvataggio delle immagini, in base a comic e numero di capitolo.

## **6 Test eseguiti**

Il progetto non è stato testato nella sua interezza, ma si è testato principalmente l'app users, per verificare che le varie viste e il model manger funzionassero a dovere.

### **6.1 UserManager**

#### **6.1.1 test\_create\_user**

Test utilizzato per controllare che il nuovo utente sia creato correttamente, con i campi popolati come voluto.

#### **6.1.2 test\_create\_superuser**

Test utilizzato per controllare che il nuovo super utente sia creato correttamente, con i campi popolati come voluto.

### **6.2 UserModel**

#### **6.2.1 setUp**

Funzione che crea l'utente usato per i vari test.

#### **6.2.2 test\_login**

Test utilizzato per verificare che il login avvenga correttamente.

#### **6.2.3 test\_login\_invalid\_credentials**

Test utilizzato per verificare che il login fallisca con credenziali errate.

#### **6.2.4 test\_create\_existing\_user**

Test utilizzato per verificare l'impossibilità di creare un utente con credenziali di un utente già esistente.

#### **6.2.5 test\_create\_user\_with\_email\_invalid**

Test utilizzato per verificare l'impossibilità di creare un utente con un indirizzo email non valido.

#### **6.2.6 test\_create\_user\_with\_username\_invalid**

Test utilizzato per verificare l'impossibilità di creare un utente con un username non valido.

#### **6.2.7 test\_create\_user\_with\_email\_and\_username\_invalid**

Test utilizzato per verificare l'impossibilità di creare un utente con un indirizzo email e un username non validi.

#### **6.2.8 test\_create\_user\_with\_password\_invalid**

Test utilizzato per verificare l'impossibilità di creare un utente con una password non valida.

### **6.3 UserViews**

#### **6.3.1 setUp**

Funzione che crea l'utente usato per i vari test.

#### **6.3.2 test\_logout**

Test utilizzato per verificare che la view di logout riporti alla home page.

#### **6.3.3 test\_change\_username**

Test utilizzato per verificare che la view di cambio username cambi correttamente l'username.

#### **6.3.4 test\_change\_username\_invalid\_username**

Test utilizzato per verificare che la view di cambio username non cambi l'username con un username non valido.

#### **6.3.5 test\_change\_username\_invalid\_username\_length**

Test utilizzato per verificare che la view di cambio username non cambi l'username con un username di lunghezza non valida.

#### **6.3.6 test\_change\_password**

Test utilizzato per verificare che la view di cambio password cambi correttamente la password.

#### **6.3.7 test\_change\_password\_invalid\_old\_password**

Test utilizzato per verificare che la view di cambio password non cambi la password se la password vecchia inserita è sbagliata.



### **6.3.8 test\_buy\_coins**

Test utilizzato per verificare che la view di acquisto di coins aggiorni correttamente il numero di coins dell'utente.

### **6.3.9 test\_buy\_coins\_invalid\_amount**

Test utilizzato per verificare che la view di acquisto di coins non aggiorni il numero di coins dell'utente se viene inserito un valore negativo.

### **6.3.10 test\_buy\_coins\_invalid\_amount\_string**

Test utilizzato per verificare che la view di acquisto di coins non aggiorni il numero di coins dell'utente se viene inserito un valore non numerico.

### **6.3.11 test\_buy\_coins\_invalid\_amount\_zero**

Test utilizzato per verificare che la view di acquisto di coins non aggiorni il numero di coins dell'utente se viene inserito un valore nullo.

### **6.3.12 test\_buy\_coins\_not\_logged**

Test utilizzato per verificare che la view di acquisto di coins riporti alla pagina di login se l'utente non è loggato.

### **6.3.13 test\_buy\_coins\_entry\_created**

Test utilizzato per verificare che la view di acquisto di coins crei una entry nella tabella delle transazioni.

### **6.3.14 test\_buy\_coins\_entry\_created\_invalid\_amount**

Test utilizzato per verificare che la view di acquisto di coins non crei una entry nella tabella delle transazioni se viene inserito un valore non valido.

### **6.3.15 test\_become\_creator**

Test utilizzato per verificare che la view per diventare creator cambi correttamente il campo is\_creator dell'utente se l'utente è loggato.

### **6.3.16 test\_become\_creator\_not\_logged**

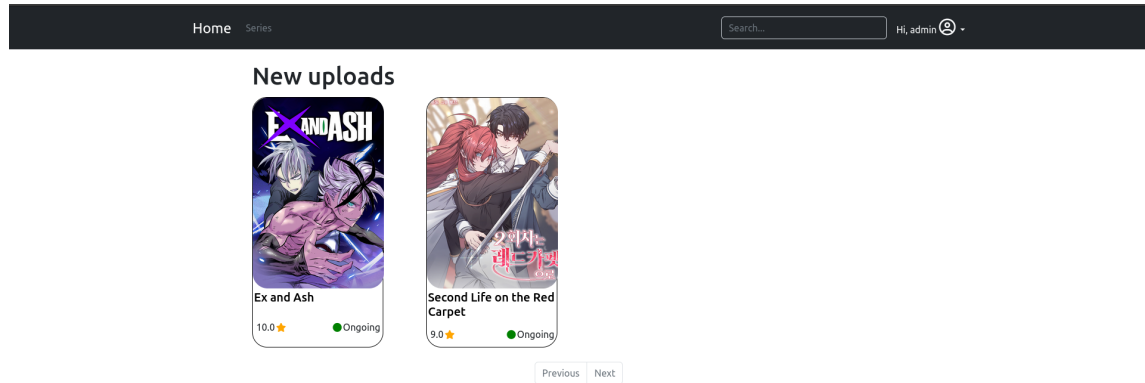
Test utilizzato per verificare che la view per diventare creator non cambi il campo is\_creator dell'utente se l'utente non è loggato.

#### **6.3.17 test\_become\_creator\_already\_creator**

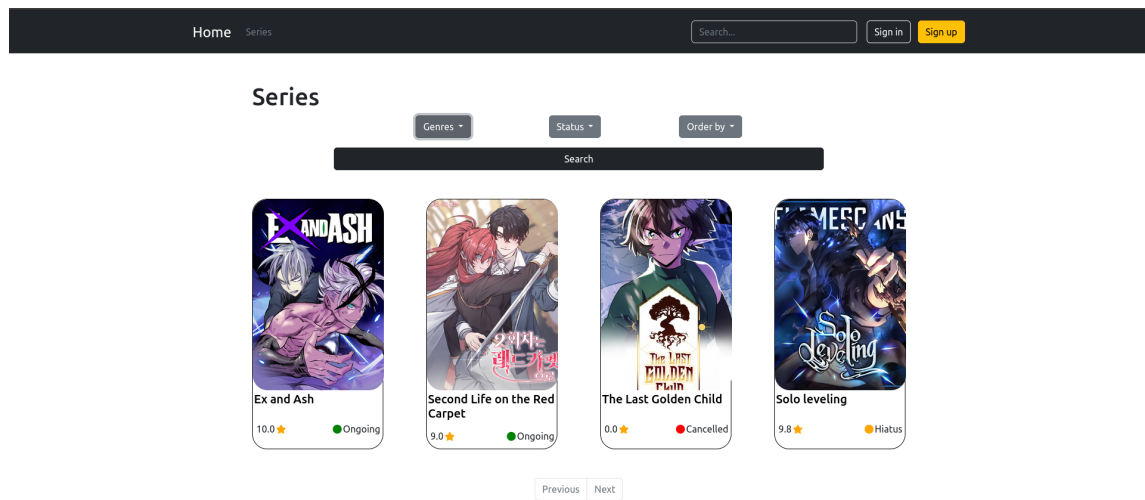
Test utilizzato per verificare che la view per diventare creator non cambi il campo is\_creator dell'utente se l'utente è già creator.

## 7 Risultati ottenuti

### 7.1 Homepage

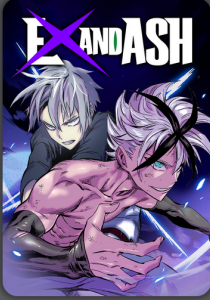


### 7.2 Series



## 7.3 Comic

[Home](#) [Series](#)  Hi, alle



### Ex and Ash

KimSehoon

10.0 Ongoing

Action Adventure Fantasy Shounen

#### Synopsis

The duo decide to become knights after a fateful meeting with Ryu, the Best Sword of Knighthood! At the testing grounds of the Knights where only the strongest of humanity gather, Ex and Ash must compete and emerge victorious against the Elite Nobles who have everything! Will they become knights? And what cruel destiny lays ahead of them...?

39 1

#### Chapters

Chapter 0

Chapter 1

## 7.4 Chapter

[Home](#) [Series](#)  Hi, admin

[All chapters](#) [Select chapter ▾](#)

[Prev](#) [Next](#)

A FEW DAYS BEFORE THE  
COMBAT TOURNAMENT BEGINS...



## 7.5 Login

[Home](#) [Series](#)

[Sign in](#) [Sign up](#)

Please sign in

[Sign in](#)

## 7.6 Register


[Home](#) [Series](#)

[Sign in](#) [Sign up](#)

Register

[Create account](#)

## 7.7 User Settings

[Home](#) [Series](#)  Hi, alle 


[Library](#) [Comments](#) [History](#) [Buy list](#) [Settings](#)

**Change Username**  
Current Username: **alle**  
New username

**Change Password**  
Old password  
  
New password  
  
Confirm password

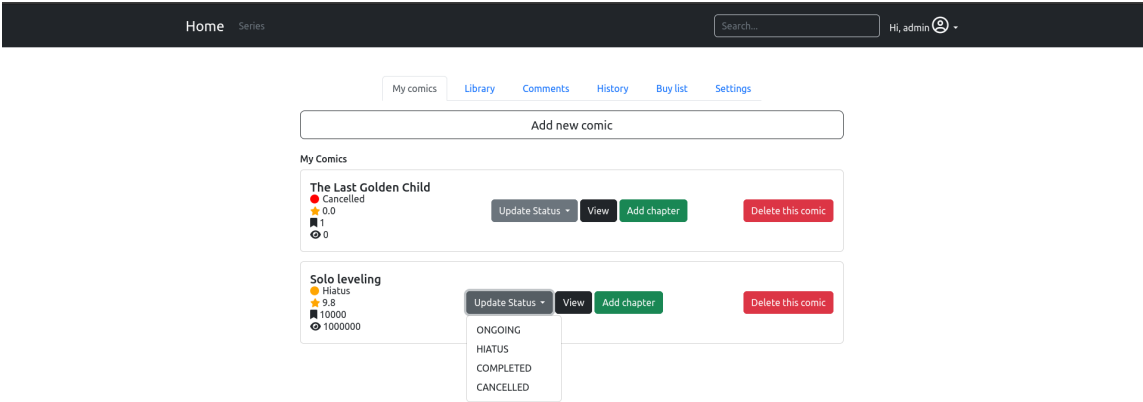
**Creator**

## 7.8 Market

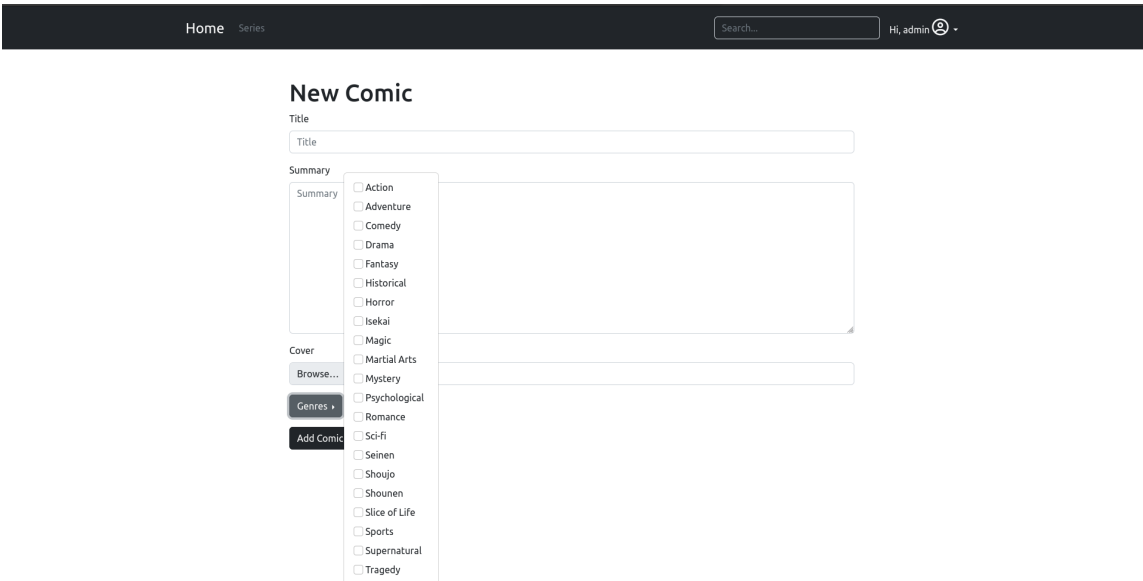
[Home](#) [Series](#)  Hi, alle 

**Coins Shop**  
500 Coins = 2.50€  
1000 Coins = 5.00€  
2000 Coins = 10.00€


# 7.9 My comics



# 7.10 Add comic



## 7.11 Add chapter

[Home](#) [Series](#)  Hi, admin 

### Add new chapter to: The Last Golden Child

Chapter number

Images

Browse...

No files selected.

Add new chapter



## 8 Problemi riscontrati

Oltre alla familiarizzazione con Django, i problemi riscontrati sono stati pochi. Oltre al problema del salvataggio delle immagini, risolto con un pacchetto di python, l'unico altro problema riscontrato è stato passare il token CSRF (cross-site request forgery) in modo corretto tramite AJAX. Fortunatamente sono state trovate soluzioni su internet, grazie ad altre persone che avevano già riscontrato questo problema.