Blog

11 Giugno 2021

Progetto di valutazione finale del corso Java base. Ogni prova è individuale e richiede la messa in pratica di tutti i concetti appresi durante il corso.

ISTRUZIONI PER LO SVILUPPO E LA CONSEGNA

Effettuare il fork del progetto presente nel repository: https://github.com/ddinuzzo/BlogAPI.git Clonare il repository sul proprio computer ed importare il progetto su Eclipse. Suddividere correttamente le classi in package. Dare dei nomi “parlanti” a classi, metodi, attributi e interfacce. Documentare tutti i metodi inserendo una descrizione chiara del funzionamento del metodo, dei parametri di input e output e delle eventuali eccezioni lanciate. Caricare il tutto sul proprio GitHub.

PANORAMICA

Si ha l’esigenza di gestire un blog di articoli multiutente. Gli utenti registrati potranno scrivere in bozza, pubblicare, modificare e eliminare i propri articoli, la lettura è concessa a tutti, anche ad utenti non registrati. Ad ogni articolo è associata una categoria ed una serie di tag. E’ possibile visualizzare gli articoli per categoria e per tag. Si deve offrire anche un servizio di ricerca filtrata con possibilità di filtrare per i seguenti campi:

- parola chiave che deve essere ricercata nel titolo, sottotitolo e nel testo dell’articolo;
- categoria;

- stato articolo;
- tags;
- autore.

SPECIFICHE

Di seguito tutti i requisiti che l'applicativo deve rispettare:

- [ ] ogni articolo ha due possibili stati: in bozza e pubblicato (campo obbligatorio)
- [ ] ogni articolo ha un titolo, un sottotitolo (opzionale), un testo, una categoria, un autore, una data di pubblicazione, di ultima modifica (opzionale) e di creazione
- [ ] ad ogni articolo è possibile associare 0 o più tag
- [ ] un utente loggato può:
    - [ ] scrivere un articolo
    - [ ] modificare i propri articoli (solo in stato bozza)
    - [ ] eliminare i propri articoli
    - [ ] associare tag ai propri articoli
    - [ ] ricercare un articolo per:
        - [ ] una stringa che verrà ricercata nel titolo, sottotitolo e nel testo dell'articolo (ricerca da effettuare in OR);
        - [ ] una categoria;
        - [ ] un tag;
        - [ ] un autore;
        - [ ] stato (se il giornalista ricerca per stato BOZZA vedrà solo i suoi articoli).
- [ ] un utente non loggato può:
    - [ ] visualizzare gli articoli per categoria
    - [ ] visualizzare un articolo (non può visualizzare articoli in stato bozza)
    - [ ] ricercare un articolo per:
        - [ ] una stringa che verrà ricercata nel titolo, sottotitolo e nel testo dell'articolo (ricerca da effettuare in OR);
        - [ ] una categoria;
        - [ ] un tag;
        - [ ] un autore;

RICHIESTE IMPLEMENTATIVE

- Progettare il database cercando di normalizzare quanto più possibile le tabelle
- Utilizzare correttamente incapsulamento, ereditarietà, classi astratte e interfacce pensando al riuso del codice
- Sfruttare i pattern MVC, DAO e DTO

SPECIFICHE DEI SERVIZI DA IMPLEMENTARE

Di seguito sono riportate le specifiche di sviluppo richieste per i vari servizi che dovrà offrire il sistema.

Endpoint categorie

`path: /api/categoria`

Recupero della lista di categorie

`GET` `/api/categoria`

Restituisce la lista di categorie presenti nel database

Status code restituiti:

- **200**: se sono state restituite delle categorie
- **404**: se non è presente alcuna categoria all'interno del database

Endpoint tag

`path: /api/tag`

Recupero della lista di tag

`GET` `/api/tag`

Restituisce la lista di tags presenti nel database

Status code restituiti:

- **200**: se sono stati restituiti dei tags
- **404**: se non è presente alcun tag all'interno del database

Endpoint Articoli

`path: /api/articolo`

Recupero di un singolo articolo

GET */api/articolo/<:id>*

Restituisce un singolo articolo in formato JSON identificato dall'id passato nella path variable `<:id>`.
L'endpoint è raggiungibile da tutti gli utenti (registrati ed anonimi), se l'id è relativo ad un articolo in stato bozza sarà restituito solo all'autore gli altri utenti otterranno uno status code 404.

Status code restituiti:

- **200**: se l'id passato come parametro è relativo ad un articolo in stato pubblicato o ad un articolo in stato bozza e l'utente che lo richiede ne è l'autore;
- **404**: se l'id passato non corrisponde a nessun articolo o se l'articolo che identifica è in stato bozza ma l'utente loggato non ne è l'autore oppure è un utente anonimo.

Ricerca articoli

GET */api/articolo*

Restituisce una lista di articoli eventualmente paginata in formato JSON.
L'endpoint è raggiungibile da tutti gli utenti (registrati ed anonimi). Gli utenti anonimi otterranno in output solo gli articoli in stato pubblicato mentre gli utenti loggati, oltre agli articoli in stato pubblicato, riceveranno anche i propri articoli in stato bozza.
Il servizio dovrà permettere di filtrare i risultati per testo (dovranno essere passati almeno 3 caratteri) cercando il testo passato come parametro nel titolo, sottotitolo e testo dell'articolo in OR.
Dovrà essere possibile ricercare un articolo anche per id, categoria, tag o autore.
Se vengono passati più filtri di ricerca tra i precedenti dovranno essere utilizzati in AND.

Status code restituiti:

- **200**: se la ricerca produce risultati
- **400**: se uno dei parametri passati in input non è formalmente corretto
- **404**: se la ricerca non produce alcun risultato

Inserimento articolo

POST */api/articolo*

Il servizio permetterà l'inserimento di un articolo ad un giornalista registrato. Dovrà prendere in input un articolo in formato JSON compilato in ogni sua parte ed indicare all'utente l'avvenuto inserimento senza restituire alcun valore in response. L'articolo inserito è sempre in bozza, il passaggio in stato pubblicato sarà effettuato da un altro servizio.

Status code restituiti:

- **204**: se l'articolo è stato inserito correttamente
- **400**: se uno dei parametri passati in input non è valorizzato o corretto
- **401**: se un utente non loggato prova ad effettuare l'inserimento di un articolo

Modifica articolo

PUT */api/articolo/<:id>*

Il servizio dovrà permettere l'update di un articolo passando in input un articolo in formato JSON. L'aggiornamento dovrà essere consentito solo all'autore dell'articolo.

Status code restituiti:

- **204**: se l'operazione di update va a buon fine
- **400**: se sono presenti parametri non formalmente corretti
- **401**: se un utente non loggato prova ad effettuare l'update di un determinato articolo
- **403**: se un utente loggato che non è l'autore dell'articolo prova ad effettuarne l'update
- **404**: se l'id passato in input non appartiene ad alcun articolo

Eliminazione articolo

`DELETE` *`/api/articolo/<:id>`*

Il servizio elimina un articolo presente all'interno del database. L'eliminazione è consentita solo all'autore dell'articolo dopo aver effettuato la login.

Status code restituiti:

- **204**: se l'eliminazione va a buon fine
- **401**: se un utente non loggato prova ad effettuare l'eliminazione di un articolo
- **403**: se l'utente loggato non è l'autore dell'articolo che cerca di eliminare
- **404**: se l'id passato in input non è associato ad alcun articolo presente nel database