Documentatie

1. Prezentare generala

Aplicatia presupune adaugarea de articole si editarea acestora. Un utilizator isi poate creea un cont si va putea adauga noi articole, edita articolele adaugate sau va putea chiar sa stearga articolele adaugate.

Aplicatia este structura pe doua pagini de adaugare de articole si editare de articole impreuna cu doua pagini de inregistrare si autentificare pentru utilizatori.

In pagina de register utilizatorul va putea sa-si creeze un cont prin adaugarea urmatoarele date: numele complet, adresa de email, un username ales si o parola. Iar in pagina de login se va putea autentifica folosind username-ul inregistrat si parola.

Dupa autentificarea utilizatorului va fi redirectionat in pagina “Home” unde va putea vizualiza toate articolele ce au fost adaugate de acesta sau de alti utilizatori. Pentru a adauga articole noi utilizatorul va trebui sa acceseze pagina “Add article” unde va completa doua campuri, “title” si “body”. Odata ce utilizatorul a adaugat un articol, acesta va fi din nou redirectionat catre pagina “Home”.

Prin accesarea unui articol utilizatorul va putea vizualiza continutul acestuia care contine titlul articolului, autor acestuia si corpul acestuia. Prin accesarea articolul adaugat de acesta, utilizatorul va mai putea vizualiza in pagina articolului si doua butoane, “Edit” si “Delete” pentru editare si stergere.

2. Structura aplicatiei

Fisierul principal al proiectului este app.js unde se face pornirea serverului, conexiunea la baza de date si

In folderul views se regaseste tot contintul HTML al aplicaitiei, fisierul layout avand structura de baza a paginii si totodata este declarata variabila content care este defapt contintul paginii in format dinamic. Pagina “Home” are continutul in fisierul index, unde se face un foreach pentru a afisa toate articolele din baza de date. Paginile “Add article” si “Edit article” au continutul HTML in fisierele add_article si respectiv, edit_article unde este specificat un formular care este completat de catre utilizator. Paginile “Register” si “Login” au continutul HTML in fisierele register si respectiv login unde este specificat un formular care este completat tot de catre utilizator.

In folderul models se vor regasi doua fisiere, article.js si respectiv user.js in care se creeaza cele doua tabele din baza de date. In fisierul article.js se afla structura tabelului Article din baza de date, in care sunt specificati 3 parametrii de tip string title, author si body si care trebuie sa fie diferiti de null in baza de date. In fisierul user.js se afla structura tabelului User din baza de date, in care sunt specificati 4 parametrii de timp string name, email, username si password si care, de asemenea, nu pot avea valoarea null in baza de date.

In folderul routes se afla doua fisiere articles.js in care se implementeaza functiile necesare afisarii, inserarii, editarii si stergerii articolele; si users.js in care se implementeaza functiile necesare inregistrarii, autentificarii si iesirii din cont a utilizatorului.

3. Functionalitatea aplicatiei pentru articole

In dezolvatarea acestei aplicatii am folosit baza de date MongoDB care este o bază de date NoSQL care utilizează un model de date bazat pe documente, oferind o alternativă la bazele de date relaționale tradiționale. MongoDB este un sistem de gestionare a bazelor de date foarte flexibil și scalabil, ideal pentru aplicații care necesită o structură dinamică a datelor și performanță ridicată. Pentru a simplifica interactiunea cu baza de date am folosit libraria mongoose, care este o bibliotecă Object Data Modeling (ODM) din MongoDB și utilizata în aplicatiile Node.js.

Pentru afisarea articolelor din pagina „Home”, mergem în fișierul app.js la apelul route pentru pagina „Home” și obsersam ca se apeleaza modelul Article, specific tabelului Article din baza de date, împreuna cu functia find, care folosește mongoose pentru a face un query în MongoDB și care selecteaza toate articolele fără nici un fel de conditiei și le aduce în fișierul index prin intermediul functiei render, la care primul parametru este fișierul HTML în care să se afiseze datele.

La adaugarea unui articol, din pagina „Add article” se completeaza inputurile specifice și odată apăsat butonul de submit se apeleaza formul din fișierul add_article, care la action are specificat route add din fișierul routes/articles.js. În acest fisier observam ca sunt doua route add, una prin get și una prin post, cea prin get verifica daca utilizatorul este conectat și afiseaza pagina în navbar; iar cea prin post mai întâi verifica daca inputurile din pagina sunt completate, iar la adugare se creeaza un nou model de tip Article care primește ca parametrii inputurile din baza de date și se prin intermediul functiei save, mongoose face un query în MongoDB prin care se insereaza un nou articole cu valorile specifice. Apoi prin intermediul functiei redirect, se face un redirect în pagina „Home”, iar functia flash afiseaza mesajul de succes „Article Added”.

Editarea articolelor se face prin apasarea butonului edit din pagina articolului, care redirectioneaza utilizatorul în pagina „Edit article” unde se apeleaza un form din fișierul edit_article. Acest form primește la actione route edit, din articles.js, aici observam ca sunt doua route edit, unul prin get și unul prin post. Route prin get verifica daca utilizatorul este autentificat și totodata daca utilizatorul este autorizat, adică daca utilizatorul este și autorul articolului, iar aceasta verificare se face in fișierul article, care este pagina articolului. În route prin post se verifica daca toate inputurile din form sunt completate, daca nu, prin intermediul functiei findById mongoose face un query în MongoDB prin care se cauta un articol în funcție de id-ul acestuia, iar daca da, se ia id-ul articolului din parametrii și se creeaza un nou obiect Article pentru obiectul cu id-ul respectiv și se actualizeaza noile valori pentru campuri.

Stergerea articolelor se face prin butonul delete din pagina articolului unde se apeleaza un ajax la onclick pe acest buton, acest ajax fiind în fișierul main.js din public/js/. Acest ajax face trimitere la fișierul articles.js din route, la un route de tip delete. În acest route se verifica daca acest utilizator este autentificat și daca este autorizat, la fel ca la functia de editare, iar prin intermediul functiei findById, mongoose face un query la MongoDB unde se cauta un articol pe baza id-ului, iar în cazul în care acesta este găsit se sterge prin intermediul funcție deleteOne în care mongoose face un query la MongoDB în care se sterge articolul pe baza id-ului dat.

4. Functionalitatea aplicației pentru utilizator

Un utilizator isi poate creea cont prin intermediul paginei de register care apeleaza un form din fisierul register. Acest form primeste la action un route register din fisierul users. In acest fisier observam sunt doua route cu denumirea de register, una prin get si una prin post. Route prin get aduce formul in pagina si afiseaza pagina HTML, iar route prin post mai intai verifica daca inputurile sunt completate, apoi creaza un nou model User cu valorile introduse, iar prin functia save, mongoose face un query in MongoDB prin care inserareaza datele introduse in form in baza de date. Apoi se face redirect in pagina de login impreuna cu un mesaj de succes.

Autentificare unui utilizator se face prin intermediul paginii login care apeleaza un form din fisierul login care primeste la action route login din fisierul users.js. In acest fisier observam ca sunt doua route cu denumirea de login, unul prin get si unul prin post. Route prin get aduce formul in pagina si afiseaza pagina HTML, iar route prin post face autentificarea utilizatorului prin intermediul librarieri passport, iar in caz de succes se face redirect catre pagina “Home”. Passport este un middleware de autentificare pentru Node.js, conceput pentru a gestiona autentificarea utilizatorilor în aplicații. Este extrem de flexibil și modular, acceptând diverse strategii de autentificare, inclusiv numele de utilizator și parola locale, OAuth și OpenID.

Deconectarea unui utilizator se face dând click pe butonul „Log Out” din navbar care apeleaza un route logout din fișierul users.js unde se apeleaza functia logoud, iar în caz de eroare se transmite mai departe mesajul prin intermediul functiei next, iar în caz de succes se face redirect către pagina login împreuna cu un mesaj de succes.