Socijalna mreža za učenje

Breznički-Herceg, Matej

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Preddiplomski sveučilišni studij računarstva

SOCIJALNA MREŽA ZA UČENJE

Završni rad

Matej Breznički-Herceg

Osijek, 2024.

SADRŽAJ

1. UVOD

U ovom radu opisan je postupak izrade društvene mreže za učenje i dijeljenje obrazovnih članaka. Svaki obrazovni članak se sastoji od blokova, a svaki blok može kao sadržaj imati tekst, sliku ili videozapis. Svaki obrazovni članak ima naslovni blok koji je obavezan i bez njega nije moguće objaviti članak. Osim objavljivanja obrazovnih članaka moguće je dodavanje alternativnih blokova na postojeće blokove obrazovnog članka. Ako jedan od korisnika nije zadovoljan načinom na koji određeni blok objašnjava temu članka taj korisnik može dodati alternativni blok sa svojim objašnjenjem na postojeći blok, pod uvjetom da su sadržaji blokova istog tipa. Moguće je glasati za blokove i alternativne blokove odabirom gore ili dolje glasa te će se prikazati blok s najvećim rezultatom glasanja. U slučaju da originalni blok i alternativni blok imaju isti rezultat glasanja prikazuje se originalni blok. Za prije navedene funkcionalnosti korisnik se treba prijaviti kao jedan od dva dostupna tipa korisničkog računa. Prvi tip je uređivač koji može dodavati obrazovne članke i alternativne blokove, ali ne može glasati. Drugi tip je čitatelj koji može glasati, ali mu nije dozvoljeno objavljivanje sadržaja. O tipu korisničkog računa korisnik odlučuje prilikom njegovog kreiranja. Postoji i admin kao treći tip korisničkog računa, ali nije dostupan korisnicima. Ako korisnik nije prijavljen on može samo gledati sadržaj, ne može ga kreirati niti glasati.

Većina sadržaja ovog rada opisuje razvoj s poslužiteljske strane (engl. *backend*), ali je opisan i razvoj funkcionalnosti s korisničke strane (engl. *frontend*). Postupak kreiranja korisničkog sučelja nije naveden. Unutar ovog rada su prezentirane društvene mreže i druge internetske aplikacije koje dijele sličnosti s ovim projektom. Navedene su tehnologije koje su korištene prilikom izrade ovog projekta kao što su web okvir i baza podataka. Važan segment ovog rada odnosi se na izradu samog projekta koji uključuje kreiranje projekta unutar web okvira, povezivanje projekta s bazom podataka te kreiranje i upravljanje korisnicima i obrazovnim člancima unutar projekta. Zadnji segment opisuje rad aplikacije iz korisničke perspektive gdje objašnjava kako se kretati unutar aplikacije, kako se prijaviti i odjaviti te kako dodavati sadržaj.

1.1. Zadatak završnog rada

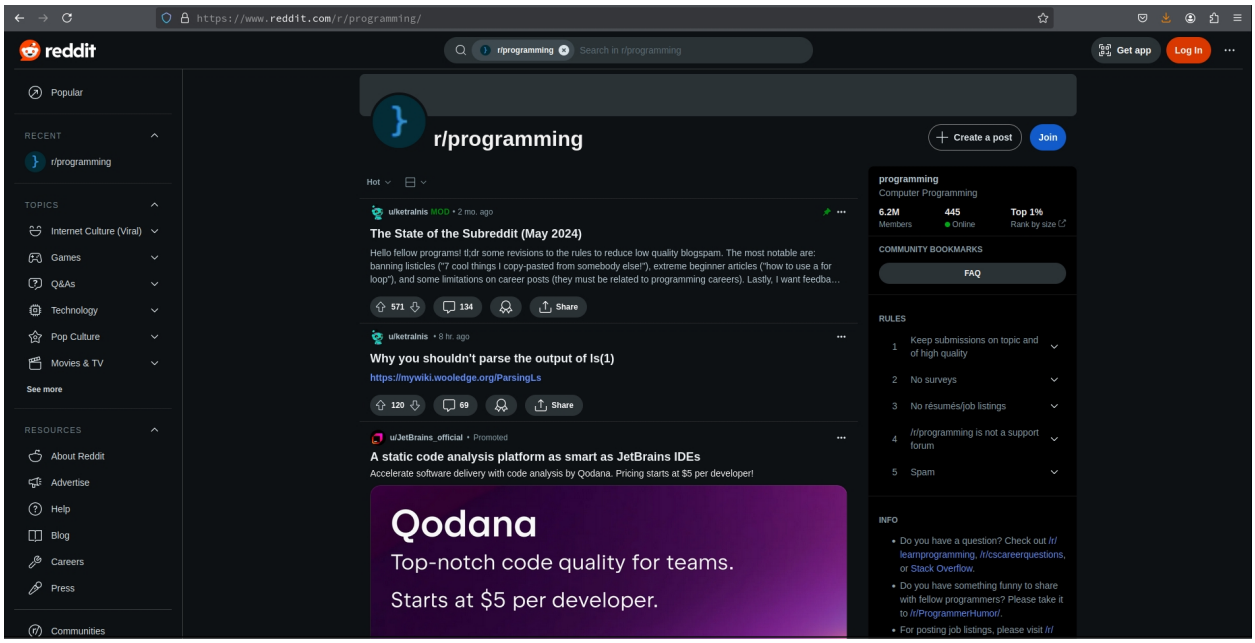
Zadatak ovog rada je dizajnirati društvenu mrežu za dijeljenje obrazovnih članaka. Svaki članak je sastavljen od tekstualnih, slikovnih i video blokova. Korisnici se dijele na uređivače, koji kreiraju obrazovne članke i ako je potrebno alternativne blokove, te čitatelje, koji glasaju za određeni blok i njegove alternative.

2. PREGLED PODRUČJA I SLIČNA RJEŠENJA

Aplikacija je originalno osmišljena, odnosno ne postoji aplikacija poznata autoru koja je identična aplikaciji iz ovog rada, ali postoje aplikacije koje imaju neke slične funkcionalnosti. U nastavku su navedene aplikacije koje imaju funkcionalnosti kao što su glasanje i blokovski prikaz sadržaja koje ima i ova aplikacija.

2.1. Reddit

Reddit (Slika 2.1.) je društvena mreža podijeljena u grupe zvane *subredditi* u kojima korisnici objavljuju i komentiraju sadržaj te glasaju s gore glasom ili dolje glasom. *Subreddit* ima prefiks r/, a korisnici imaju prefiks u/. Svaki *subreddit* je specijaliziran za određen sadržaj. Moderatori nadgledaju *subreddit* i brinu se da se na njemu ne objavljuje sadržaj neprimjeren tom *subredditu*. Reddit ima tražilicu na kojoj je moguća pretraga *subreddita* i korisnika, ali ne i objavljenog sadržaja. Za objavljivanje i glasanje je potreban korisnički račun, ali za gledanje objava korisnički račun je nepotreban. Svaki korisnik s obzirom na broj gore glasova i dolje glasova na svojim objavama i komentarima dobiva određen broj karme.



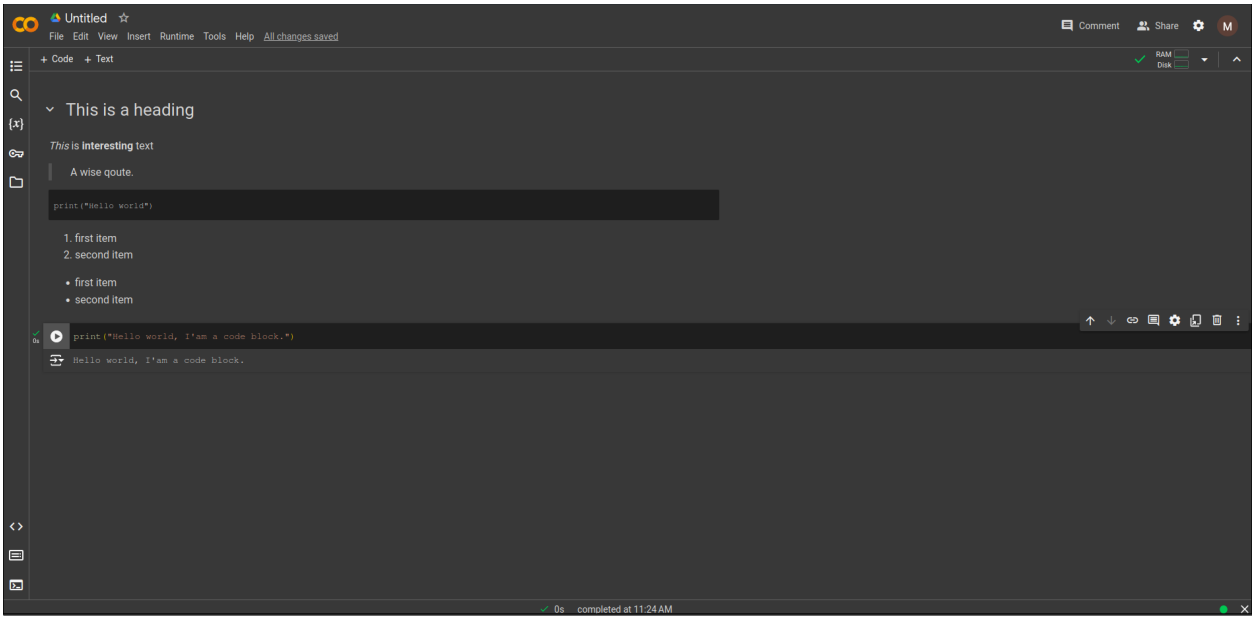
Slika 2.1. Web aplikacija Reddit [13]

Glavna sličnost Reddita i aplikacije iz ovog završnog rada je mogućnost gore i dolje glasanja, razlika je jedino u tome za što se glasa. U slučaju Reddita glasa se na objavama i komentarima kako bi se te objave i komentari bolje rangirali i preporučili većem broju korisnika. U slučaju ove aplikacije glasa se na blokovima unutar članka kako bi se alternativni blokovi mogli

rangirati i iz njih izdvojiti najbolje objašnjenje teme u članku. Također dosta velika sličnost je odabir tipa sadržaja koji se objavljuje. Na Redditu kao i u ovoj aplikaciji će se većinom objavljivati tekstualni sadržaj, ali je moguća i objava slika i videozapisa, jedina razlika je da će u ovoj aplikaciji biti moguće objaviti više raznih tipova sadržaja u jednom članku dok Reddit može objavljivati tekst i sliku u jednoj objavi, ali ne i videozapise.

2.2. Google Colab

Google Colab (Slika 2.2.) je platforma koja omogućuje izradu virtualnih bilježnica za dokumentaciju i edukaciju primarno namijenjena podatkovnim znanostima i istraživanju umjetne inteligencije. Bilježnice se izrađuju od skupa tekstualnih blokova i blokova koda. Blokove je moguće dodavati ispod ili iznad postojećeg bloka te ih je moguće brisati neovisno o poziciji bloka unutar bilježnice. U blokove koda je moguće pisati kod programskog jezika Python i izvršiti ga u web aplikaciji bez korištenja lokalnog prevoditelja. Za izvršavanje koda u pojedinom bloku je potrebno pritisnuti opciju *Play* koja se nalazi unutar bloka koda, a ako se želi izvršiti sve blokove koda u izborniku pod *Runtime* potrebno je odabrati *Run all*. Važno je napomenuti da je redoslijed izvršavanja blokova koda bitan. Unutar tekstualnog bloka je moguće dodavati slike, linkove, blokove citiranja (engl. *blockquote*) te poredane i neporedane liste. Tekst unutar blokova je moguće podebljati i nakositi te ga staviti kao zaglavlje ili ga formatirati kao kod. Za pisanje unutar tekstualnih blokova koristi se označni jezik Markdown koji se najčešće pojavljuje u README.md datotekama.



Slika 2.2. Web aplikacija Google Colab [14]

Google Colab prikazuje sadržaj u obliku blokova kao i aplikacija ovog projekta, ali u slučaju Google Colaba postoje samo dvije vrste blokova dok u slučaju ove aplikacije postoje tri vrste. Vrste blokova se razlikuju, u ovoj aplikaciji postoje tekstualni blokovi te blokovi za slike i videozapise dok kod Google Colaba postoje tekstualni i blokovi koda. Također u ovoj aplikaciji nije moguće dodavati blokove ispod ili iznad određenih blokova već ih je moguće dodavati samo na dno liste blokova. Moguće je brisati određeni blok kao i kod Google Colaba. Za razliku od Google Colaba, gdje se bilježnice moraju podijeliti s drugim korisnicima kako bi im bile dostupne, u ovoj aplikaciji će članci biti javno dostupni svim korisnicima.

2.3. Wikipedia

Wikipedia (Slika 2.3.) je internetska enciklopedija i jedna od najposjećenijih internetskih stranica. Na njoj se objavljuje razan obrazovni sadržaj iz različitih znanstvenih disciplina. Sadržaj je podijeljen na podnaslove kao što su povijest teme članka, tema članka u današnjem vremenu i ako tema ima podteme svaka ta podteme može biti poseban podnaslov. Sadržaj članaka je ispunjen poveznicama koje vode na druge članke. Wikipedia primarno ima sadržaj na engleskom jeziku, ali podržava i mnoštvo drugih jezika. Važno je napomenuti kako sadržaj članka može biti različit ovisno o jeziku na koji je članak preveden. Sadržaj stranice je održavan od strane volontera. Stranica ima mogućnost prijave i kreiranja korisnika.

Slika 2.3. Jedan od članaka na Wikipediji [15]

Glavna sličnost aplikacije ovog projekta i Wikipedije je objavljivanje obrazovnih članaka, jedina razlika je format u kojem su ti članci objavljeni. U slučaju Wikipedije članci su fragmentirani na

podnaslove, a u slučaju aplikacije ovog projekta članci su fragmentirani na blokove. Također u obje aplikacije je moguće objavljivati obrazovne članke različitih tema. Članke je moguće tražiti uz pomoć tražilice, ali u slučaju Wikipedije tražilica je naprednija (sadržaj tražilice ne mora biti isti kao i naslov članka). Obje aplikacije imaju mogućnost kreiranja, prijave i odjave korisnika.

3. KORIŠTENE TEHNOLOGIJE

Unutar ovog poglavlja navedene su tehnologije korištene unutar ovog projekta. Za svaku tehnologiju su naveden osnovne informacije te za koju ulogu su korištene unutar projekta.

3.1. Python

Prema [1] Python je programski jezik koji se koristi u raznim područjima računarstva, a neka od njih su web programiranje, desktop aplikacije, podatkovne znanosti i strojno učenje. Postoji dosta korisničkih biblioteka specijaliziranih za prije spomenuta područja što čini Python dosta popularnim kod razvoja aplikacija. Druga prednost Pythona je njegova jednostavnost, odnosno lakše ga je naučiti od većine programskih jezika zbog njegove jednostavne sintakse. Python za razliku od ostalih programskih jezika ne koristi razdjelnik ; na kraju svake naredbe, već koristi prelazak u novi red kao razdjelnik. Također Python ne koristi vitičaste zagrade za označavanje blokova, već tabulator kako bi prepoznao nalaze li se naredbe u istom bloku. U ovom projektu je korištena mogućnost da se kreira virtualno okruženje uz pomoć Pythona. Verzija korištena u ovom projektu je Python 3.12.3. Python je unutar ovog projekta korišten kao glavni programski jezik s poslužiteljske strane.

3.2. Django

Prema [3] Django je web okvir namijenjen za razvoj aplikacija na poslužiteljskoj strani koji se fokusira na brzini razvoja i skalabilnosti. Programski jezik u kojem se razvijaju aplikacije je Python. Riječ je o web okviru otvorenog koda (engl. *open-source*) kojeg održava Django Software Foundation. Njegov rad se temelji na MVC (engl. *Model View Controller*) oblikovnom obrascu kod kojeg model rukuje s bazom podataka i njenim podacima te ih predaje kontroleru. Kontroler priprema podatke za slanje putem mreže prema pogledu koji prezentira podatke korisniku. Unutar Django okvira MVC ima drugačiji naziv komponenta gdje model predstavlja model, pogled predstavlja predložak (engl. *template*), a kontroler predstavlja pogled (engl. *view*), u nastavku ovog rada koristit će se imena komponenti MVC-a koje koristi Django. U ovom projektu korištena je verzija Django 5.0.6.

3.3. PostgreSQL

Prema [4] PostgreSQL je objektno-relacijska baza podataka koja je poznata po svojoj proširivosti i integritetu podataka. Otvorenog je koda i podržana je od strane svih popularnih operacijskih

sustava kao što su Linux, Windows, MacOS, BSD i Solaris. Omogućuje korisniku da kreira vlastite tipove podataka i funkcije. Koristi SQL jezik i podržava većinu njegovih standarda. Za ovaj projekt kako bi spojili bazu podataka s razvojnim okvirom Django potreban nam je dodatan paket psycopg2 kako bi međusobno mogli komunicirati. Paket je moguće preuzeti preko Pythonovog upravitelja paketima (engl. *package manager*) Pip. Verzija korištena u ovom projektu je PostgreSQL 16.2.

3.4. HTML

Prema [5] HTML (engl. *HyperText Markup Language*) je označni jezik za dizajniranje internetskih stranica. On definira sadržaj internetske stranice i jedna je od temeljnih tehnologija za dizajniranje internetskih stranica. Koristi oznake kako bi omogućio umetanje različitog sadržaja kao što je tekst, poveznice, slike i mnogih drugih. U ovom projektu se koristi HTML5 uz još dvije temeljne tehnologije u kreiranju internetskih stranica. Glavna uloga HTML-a u ovom projektu je kreiranje web stranica koje će služiti kao korisničko sučelje.

3.5. CSS

Prema [6] CSS (engl. *Cascading Style Sheets*) je jezik za stiliziranje internetskih stranica. On definira kako će internetska stranica biti prikazana korisniku. CSS je jedna od temeljnih tehnologija za dizajniranje internetskih stranica i standard za sve internetske preglednike. Glavna uloga CSS-a unutar projekta je stiliziranje web stranica kako bi bile privlačnije i intuitivnije.

3.6. JavaScript

Prema [7] JavaScript je programski jezik i treća od temeljnih tehnologija za dizajniranje web stranica. Uz pomoć njega se izvršavaju naredbe koje stranici daju interaktivna obilježja kao što su prikaz slike pritiskom na gumb ili ispis sadržaja pritiskom na drugi gumb. Važno je napomenuti da se koristi za programiranje s klijentske strane. JavaScript je moguće koristiti i izvan internetskog preglednika od kojih je najpoznatiji primjer Node.js okruženje koje omogućuje razvoj aplikacija s poslužiteljske strane, a ne klijentske. U ovom projektu JavaScript se koristi samo s klijentske strane u internetskom pregledniku, a korišteni standard je ES6.

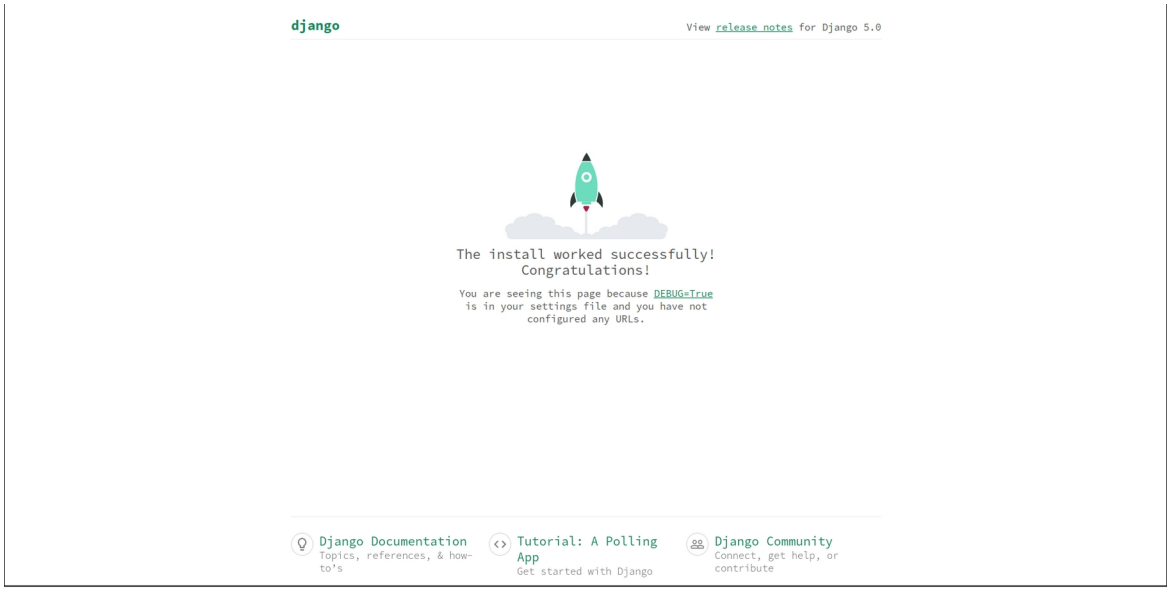
4. IZRADA APLIKACIJE

Unutar ovog poglavlja definirane su funkcionalnosti te njihova realizacija na poslužiteljskoj i klijentskoj strani. Za informaciju o ispravnoj uporabi aplikacije pogledati sljedeće poglavlje.

4.1. Kreiranje Django projekta

Kako bi se aplikacija kreirala prvo je potrebno kreirati Django projekt. Prije kreiranja projekta potrebno je kreirati virtualno okruženje. Virtualno okruženje je potrebno kako bi izolirali projekt od ostatka sustava i osigurali da aplikacija koristi ispravne verzije paketa. Za kreiranje virtualnog okruženja koristit će se Pythonovo virtualno okruženje koje omogućuje preuzimanje paketa preko Pythonovog upravitelja paketa Pip i spremanje tih paketa u virtualno okruženje. Kreiranje virtualnog okruženja i njegova aktivacija za Linux (Kod 4.1.) i Windows (Kod 4.2.) je prikazana u blokovima ispod.

Sljedeći korak je kreiranje Django projekta upisivanjem naredbe i imena projekta (Kod 4.3.) u naredbenu liniju. Nakon kreiranja ulazi se u direktorij projekta i pokreće ga se (Kod 4.4.). Odlaskom na http://127.0.0.1:8000/ dobiva se ispis (Slika 4.1.) koji govori da je instalacija bila uspješna. U slučaju da se taj ispis ne prikazuje kreiranje projekta nije uspjelo.



Slika 4.1. Ispis na http://127.0.0.1:8000/

Nakon kreiranja projekta potrebno je kreirati aplikaciju (Kod 4.5.), u ovom projektu aplikacija se zove app. Unutar Django okvira projekt se sastoji od jedne ili više aplikacija. Svaka aplikacija ima svoju zadaću unutar projekta. Tako je moguće imati aplikaciju za upravljanje obrazovnim

člancima, aplikaciju za web trgovinu i aplikaciju za privatni razgovor (engl. *chat*) unutar jednog projekta. Unutar ovog projekta stvorena je samo jedna aplikacija koja upravlja obrazovnim člancima. Nakon kreiranja aplikacije potrebno ju je uvrstiti unutar settings.py datoteke (Kod 4.6.) i definirati poveznice unutar urls.py datoteke projekta koje vode do urls.py datoteke kreirane aplikacije (Kod 4.7.).

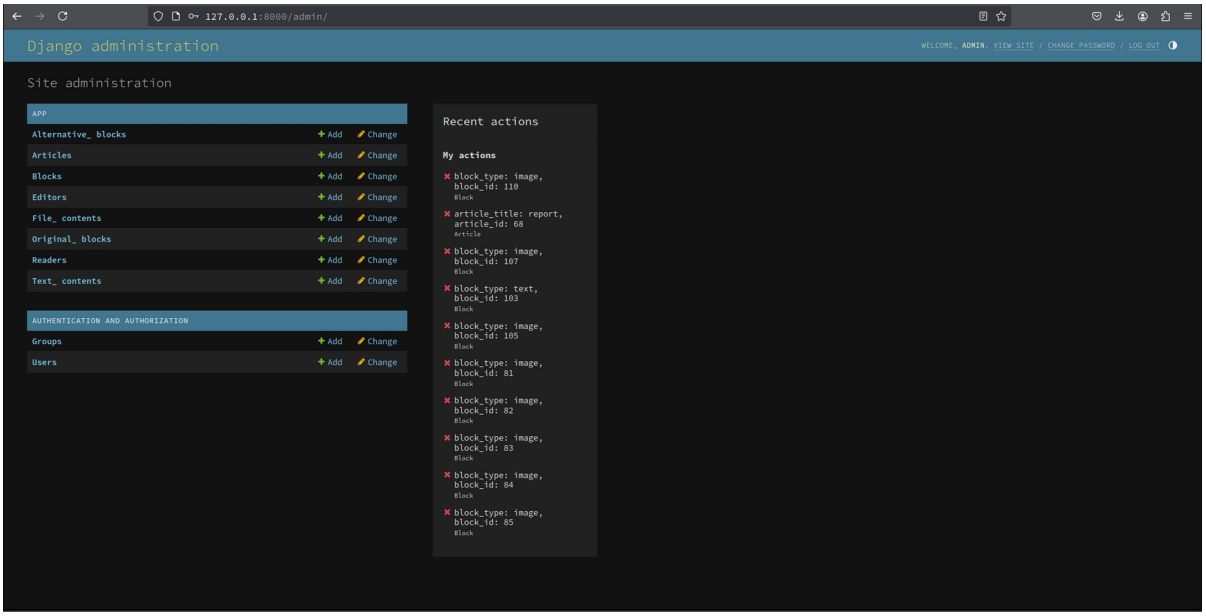
4.2. Povezivanje projekta i baze podataka

Django koristi SQLite bazu podataka, ali je preporuka za projekte koristiti neku od naprednijih baza podataka kao PostgreSQL koji će se koristiti u ovom projektu. Prije nego što se baza podataka poveže s projektom potrebno je preuzeti paket psycopg2 kako bi PostgreSQL baza i Django projekt mogli komunicirati. U slučaju da su svi paketi preuzeti preko requirements.txt datoteke psycopg2 paket bi već trebao biti preuzet.

Nakon preuzimanja potrebno je kreirati PostgreSQL bazu podataka i korisnika za tu bazu podataka. Kreiranu bazu i projekt je moguće spojiti odlaskom u datoteku settings.py u projektu te zamijeniti konfiguraciju SQLite baze s konfiguracijom PostgreSQL baze. Unutar konfiguracije (Kod 4.8.) je potrebno navesti podatke kao što su ime baze, korisnika, lozinku, poslužitelja i port na koji će se projekt spojiti.

4.3. Upravljanje korisnicima

Django okvir ima kreiran model korisnika koji će se koristiti za ovaj projekt, ali prije je potrebno kreirati administratora (engl. *superuser*). Administrator se kreira uz pomoć naredbenog retka (Kod 4.9.) i za njegovo kreiranje potrebno je korisničko ime, adresa e-pošte i lozinka s potvrdom lozine. Administrator je jedini korisnik koji ima pristup administratorskoj stranici (Slika 4.2.).



Slika 4.2. Administratorska stranica

Kao što je prije spomenuto Django razvojni okvir već ima model korisnika koji će se proširiti za potrebe ovog projekta. Tako se kreiraju dva modela unutar models.py datoteke, uređivač (Kod 4.10.) i čitatelj (Kod 4.11.), koji unutar sebe imaju korisnika kao atribut. Na taj način se dobiva odnos između novokreiranih modela i modela korisnika takav da novokreirani modeli stvaraju ulogu koju korisnik može imati. Osim prijašnja dva modela kreiran je i model za upravljanje korisnicima (Kod 4.12.) koji prijavljuje, odjavljuje i stvara nove korisnike.

Sada je uz pomoć kreiranih modela moguće realizirati sve bitne radnje za korisnike kao što su prijava i odjava te kreiranje novih korisnika. Za prijavu je potrebno definirati putanju unutar urls.py datoteke aplikacije (Kod 4.13.). Ta putanja pristupa pogledu za prijavu unutar view.py datoteke aplikacije (Kod 4.14.). Pogled za prijavu korisnika je zapravo funkcija koja prima zahtjev korisnika i ako zahtjev nema post metodu pogled samo vraća HTML dokument za prijavu. U slučaju da je zahtjev ima post metodu onda se preuzima korisničko ime i lozinka korisnika iz zahtjeva te se korisnik prijavljuje uz pomoć prije navedene klase za upravljanje korisnicima. U slučaju uspješne prijave korisnika se prebacuje na početnu stranicu, a u slučaju neuspješne korisnika se vraća na stranicu za prijavu s porukom greške.

Odjava korisnika realizirana je na sličan način. Prvo se definira putanja unutar urls.py datoteke aplikacije koja vodi do pogleda za odjavu (Kod 4.15.). Pogled za odjavu je također funkcija koja prima zahtjev, ali sada ne provjerava radi li se o post metodi ili ne. Unutar pogleda za odjavu se instancira prije spomenuta klasa za upravljanje korisnicima koja uz pomoć svoje metode za odjavu odjavljuje korisnika i učitava početnu stranicu.

Za izradu novog korisnika također je potrebno kreirati putanju unutar urls.py datoteke koja vodi do pogleda unutar views.py datoteke (Kod 4.16.). Pogled za kreiranje korisnika također prima zahtjev i u slučaju da nije riječ o post metodi pogled učitava HTML dokument za kreiranje korisnika. Unutar dokumenta za kreiranje korisnika se nalaze polja za unos podataka o korisniku kao što su korisničko ime, adresa e-pošte i lozinka, ali i mogućnost da se postane uređivač. U slučaju da se ta mogućnost ne odabere korisnik postaje čitatelj. Sva ta polja se nalaze unutar obrasca (engl. *form*). Poslani obrazac vraća se do pogleda za kreiranje korisnika gdje je sada metoda zahtjeva post pa se izvršavaju sljedeći koraci. Izdvajaju se korisničko ime, adresu e-pošte i dvije lozinke (druga lozinka je potvrda ispravnog unosa prve lozinke) te opciju za editora ako postoji. Uspoređuju se dvije lozinke i u slučaju da nisu jednake korisnika se vraća na stranicu za kreiranje korisnika s porukom greške. U slučaju da su lozinke jednake kreira se instanca klase za upravljanje korisnicima i ovisno o odabiru želi li korisnik biti uređivač ili ne želi kreira se uređivača ili čitatelja. Na kraju se usmjerava korisnika na stranicu za prijavu da se prijavi u sustav.

4.4. Rukovanje obrazovnim člancima i njihovim blokovima

Obrazovni članci u ovoj aplikaciji se sastoje od skupa blokova gdje svaki blok može imati niti jedan, jedan ili više alternativnih blokova. Obrazovni članak je definiran unutar models.py datoteke (Kod 4.17.) i kao atribute ima svoj naslov i autora koji je isključivo uređivač. Unutar modela se nalazi metoda za dobivanje kratkog sadržaja. Model također ima metodu uz pomoć koje dohvaća sve blokove koji pokazuju na njega i vraća ih pozivatelju metode. Blokovi (Kod 4.18.) kao atribute imaju tip bloka gdje se definira hoće li blok imati tekstualan ili nekakav drugi sadržaj, rezultat koji je odnos između gore i dolje glasova i autor koji također može biti samo uređivač. Kao metode blok ima metodu računanje rezultata i metodu za dohvaćanje sadržaja bloka. Sam sadržaj je pohranjen u modelu sadržaja koji je dio modela blok. Od modela sadržaja postoje dva u aplikaciji od kojih je jedan tekstualan (Kod 4.19.), a drugi datotečni (Kod 4.20.). Sadržaj je odvojen u posebne klase kako bi se s blokovima moglo rukovati neovisno o njihovom sadržaju. Također ako se želi dodati novi tip sadržaja ne mora se kreirati novi blok, nego samo novi sadržaj koji će postati atribut tog bloka. Osim do sada definiranih modela potrebna su još dva da bi aplikacija ispravno radila, a to su originalni (Kod 4.21.) i alternativni (Kod 4.22.) blokovi. Uloga tih blokova je organizacija blokova u strukturu sličnu stablu kako bi rukovanje blokovima bilo lakše. Svaki originalni blok pokazuje na članak kojem pripada i sadrži metode za dobivanje najboljeg bloka od svojih alternativa i sebe te metodu za dobivanje svih alternativa.

Svaki alternativni blok pokazuje na svoj originalni blok, a od metoda ima metodu koja vraća sve njegove alternative preko svog originalnog bloka. Modeli originalnog i alternativnog bloka kao atribut imaju blok.

Izlistaj obrazovnih članaka je realiziran na sljedeći način. Definira se putanja u urls.py datoteci aplikacije do pogleda za izlistaj članaka (Kod 4.23.). Pogled za izlistaj provjerava je li korisnik koristio tražilicu. Ako nije onda se uzimaju pet najnovijih članaka iz baze podataka i predaju se početnoj stranici koja ih prikazuje korisniku. U slučaju da je korištena tražilica uzima se sadržaj tražilice i ako je prazan prikazuje se pet najnovijih članaka, a ako ima neki sadržaj unutar sebe prikazuju se članci koji imaju naslov isti tom sadržaju tražilice.

Postoji i mogućnost da se izlistaju svi članci kojima je autor trenutni korisnik. Unutar urls.py datoteke aplikacije definira se putanja do pogleda. Unutar views.py datoteke definira se pogled koji će prikazati samo korisnikove članke (Kod 4.24.). Taj pogled prima zahtjev i iz njega izdvaja trenutno prijavljenog korisnika. Potom filtrira sve obrazovne članke i izdvaja one kojima je autor trenutni korisnik. Te članke predaje početnoj stranici koja ih izlistava uz prikladan naslov.

Klikom na određeni članak otvara se nova stranica gdje su izlistani svi blokovi tog članka. Prilikom klika na članak odabire se putanja iz datoteke urls.py datoteke koja aktivira pogled za detaljan prikaz obrazovnog članka (Kod 4.25.). Taj pogled osim zahtjeva prima i identifikacijsku oznaku odabranog članka. Uz pomoć identifikacijske oznake uzima članak iz baze podataka i koristi njegovu metodu za dohvaćanje blokova i te blokove sa naslovom i autorom članka predaje stranici za izlistaj blokova odabranog članka. Svaki izlistani blok je ili originalni blok ili alternativni blok s boljim rezultatom glasanja.

Svaki izlistani blok članka ima poveznicu koja vodi prema njegovim alternativama. Klikom na poveznicu preuzima se putanja iz urls.py datoteke. Putanja aktivira pogled za izlistaj alternativa (Kod 4.26.) iz views.py datoteke. Pogled osim zahtjeva prima i identifikacijsku oznaku odabranog bloka. Preko identifikacijske oznake dobiva se odabrani blok iz baze i provjerava se je li riječ o originalnom ili alternativnom bloku kako bi ga se dohvatilo iz baze. Neovisno je li riječ o alternativnom ili originalnom bloku poziva se metoda za dohvaćanje alternativnih blokova odabranog bloka i predaje ih stranici za izlistaj alternativnih blokova koja će ih prikazati korisniku.

Postoji još jedna važna putanja koja vraća korisnika s izlistaja svih alternativa na stranicu detaljnog prikaza obrazovnog članka. Ona poziva pogled (Kod 4.27.) koji prima zahtjev i identifikacijsku oznaku bloka kojem su izlistane alternative. Provjerava je li riječ o originalnom ili alternativnom bloku te ako je riječ o alternativnom bloku dohvaća njegov originalni. Od originalnog bloka se dohvaća identifikacijska oznaka obrazovnog članka koja se predaje putanji za detaljan prikaz obrazovnog članka.

4.5. Manipulacija blokovima i obrazovnim člancima

Do sada je objašnjeno kako poslužitelj rukuje blokovima i kako ih prikazuje, a u nastavku slijedi objašnjenje kako korisnici mogu upravljati blokovima, dodavati ih, glasati gore ili dolje određeni blok te kreirati obrazovne članke. Kako bi se ove radnje mogle realizirati prvo je potrebno izmijeniti models.py datoteku dodavanjem modela glasova (Kod 4.28.) koji ima atribute čitatelja, blok za koji se glasalo i razliku rezultata. Navedeni model predstavlja vezu više naprema više unutar baze podataka između čitatelja i bloka. Također dodana je funkcija koja provjerava tip datoteke koja se sprema unutar modela datotečnog sadržaja (Kod 4.29.). Naime moguće je ubaciti datoteku neželjenog tipa i spremiti ju što je nepoželjno zato što korisnici mogu ubaciti skripte koje će naštetiti poslužitelju. Da bi se to spriječilo uvodi se *validator* koji provjerava tip datoteke. Django ima *validator* za provjeru datoteke, ali on provjerava tip datoteke samo po ekstenziji datoteke što nije poželjno zato što se ekstenzija datoteke može promijeniti na prihvatljivu ekstenziju. Da bi se to izbjeglo kreiran je novi *validator* koji provjerava sadržaj datoteke po MIME tipu. Potrebno je modificirati i model čitatelja s dodatnim metodama (Kod 4.30.). Prva metoda odgovara na pitanje je li čitatelj glasao na određenom bloku. Druga i treća metoda govore je li čitatelj gore ili dolje glasao. Zadnje dvije metode gore glasaju i dolje glasaju određeni blok.

Glasanje je realizirano tako da se zahtjev s identifikacijskom oznakom bloka i dodatnom vrijednošću šalje post metodom poslužitelju preko obrasca. Unutar pogleda za izlistaj alternativnih blokova dodan je uvjet koji ispituje radi li se o post metodi (Kod 4.31.). Ako je riječ o post metodi dohvaća se blok iz baze preko njegove identifikacijske oznake te se glasa gore ako je dodatna vrijednost veća od 0, a u suprotnom glasa se dolje. Unutar pogleda za izlistaj alternativa je dodana lista koja sprema rezultat glasanja trenutnog korisnika. Ta lista će na korisničkoj strani prikazati čitatelju za koje je blokove glasao.

Uređivačima je dozvoljeno dodavati alternativne blokove na postojeće blokove obrazovnog članka bez obzira jesu li oni autor članka ili nisu. Pritiskom na opciju za dodavanje alternativnog bloka prikazuje se stranica za dodavanje blokova koja može biti jedna od tri stranice (ovisno o tipu bloka kojem se dodaje alternativni blok), a to su dodavanje tekstualnog bloka, dodavanje bloka sa slikom i dodavanje bloka s videozapisom. Odabirom opcije putanja iz urls.py pokreće pogled (Kod 4.32.) koji provjerava ima li zahtjev post metodu, ako nema onda učitava jednu od prije spomenutih stranica za unos blokova. Ako ima post metodu onda se kreira novi blok i novi sadržaj za taj blok te provjerava je li blok kojem se dodaje alternativa originalni blok ili alternativni blok. U slučaju da je blok kojem se dodaje alternativa alternativni blok onda se od njega dobiva njegov originalni blok. Kreira se novi alternativni blok koji prima novokreirani blok i pokazuje na prije određeni originalni blok. Na kraju se korisnika prebacuje na prikaz svih alternativnih blokova uključujući i novododani blok.

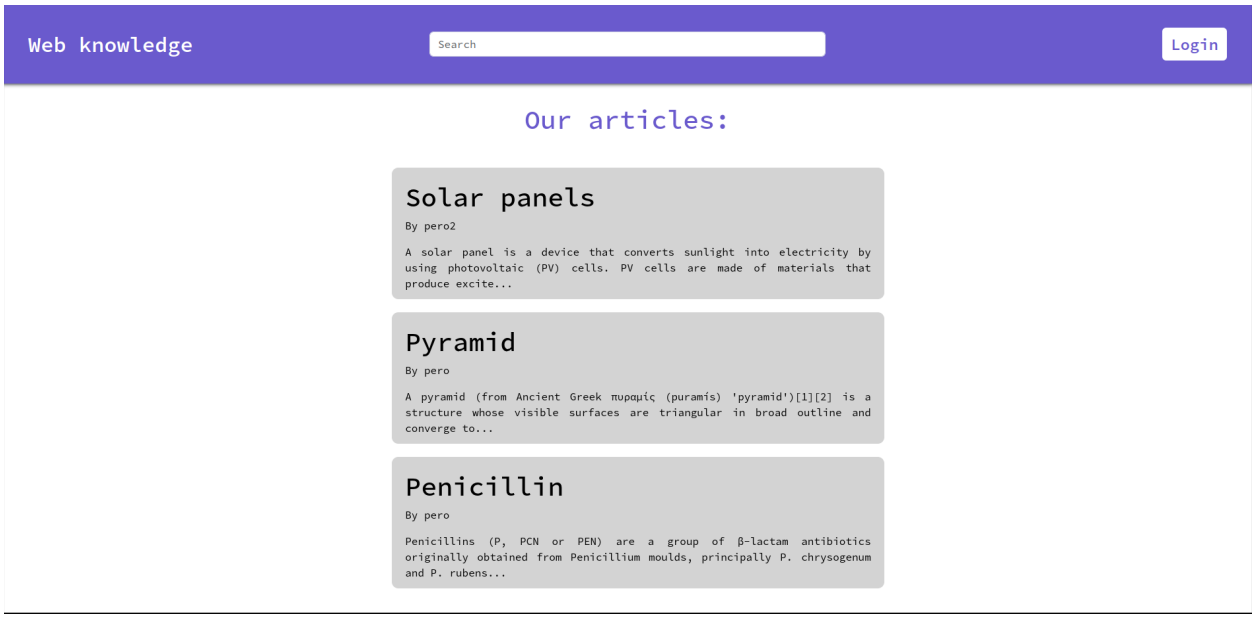
Uređivačima je dozvoljeno kreiranje novih obrazovnih članaka. Odabirom opcije za kreiranje novog članka unutar urls.py datoteke aplikacije se odabire putanja koja aktivira pogled za kreiranje novog obrazovnog članka (Kod 4.33.). Pogled prima zahtjev i provjerava ima li zahtjev post metodu. Ako nema pogled predaje korisniku HTML dokument za kreiranje novog obrazovnog članka. Kreiranje članka se radi lokalno na klijentovom računalu uz pomoć JavaScripta. Unutar dokumenta se nalaze tekstualno polje za naslov te dva gumba, jedan za dodavanje novog bloka i drugi za slanje obrazovnog članka poslužitelju. Dodavanjem novog bloka otvara se prozor unutar kojeg je moguće staviti sadržaj i kreirati novi blok. Kreirani blok se dodaje u polje blokova te se uz pomoć dvije funkcije (Kod 4.34. i Kod 4.35.) dodaje unutar HTML-ove liste i obrasca. Nakon dodavanja bloka prikaz novog bloka je moguće vidjeti unutar HTML dokumenta. Važno je napomenuti da se taj blok nalazi samo lokalno na korisnikovom računalu te ako se članak ne objavi na poslužitelju neće se spremiti u bazu i prikazati drugim korisnicima. Kreirane blokove je moguće izbrisati, u tom slučaju se odabrani blok briše iz HTML-ove liste i obrasca te se neće više prikazivati korisniku niti poslati na poslužitelja. Slanjem blokova na poslužitelj se šalje sadržaj blokova, njihov tip, naslov članka i broj blokova, odnosno koliko ih je poslano. Ti podaci se na poslužitelju izdvajaju iz zahtjeva unutar for petlje i kreiraju se blokovi te se spremaju na poslužitelja. U slučaju da se dogodi iznimka (engl. *exception*) zbog zabranjenog tipa datoteke onda se svi prijašnji spremljeni blokovi članka brišu iz baze i javlja se korisniku da je došlo do greške zbog postavljanja zabranjenog tipa datoteke na poslužitelja.

5. KORIŠTENJE APLIKACIJE

Unutar ovog poglavlja definirane su sve funkcionalnosti bitne korisniku te kako ih je poželjno koristiti. Za informaciju o tehničkoj pozadini aplikacije najbolje je proučiti prijašnje poglavlje.

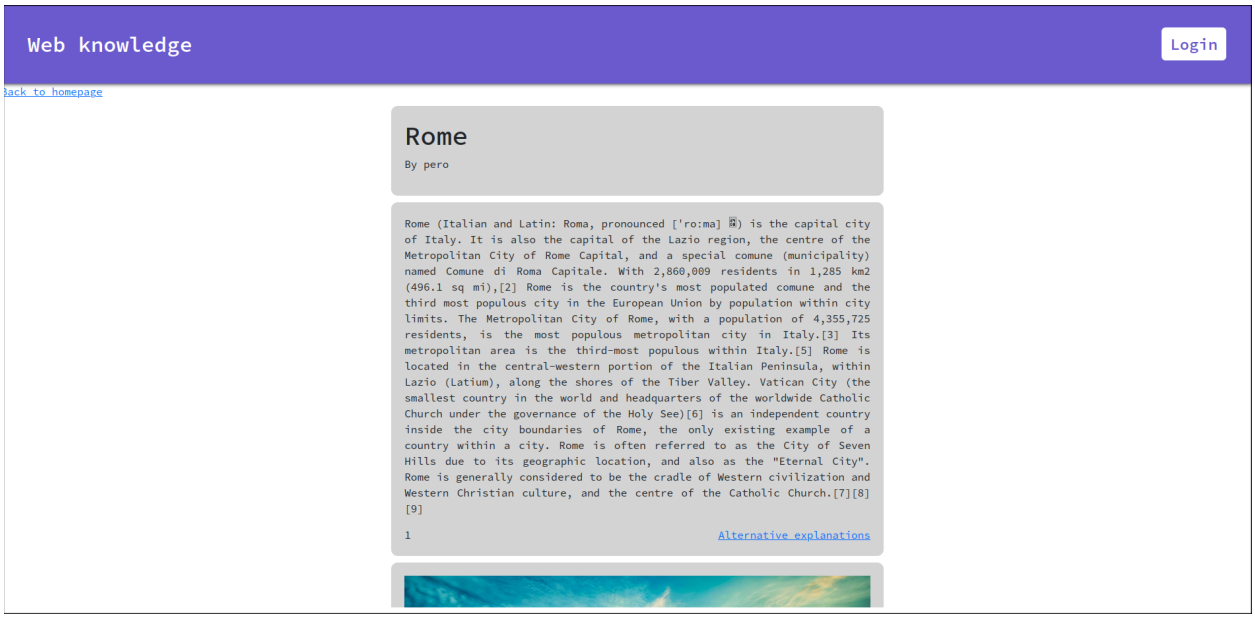
5.1. Navigacija unutar aplikacije

Prilikom dolaska na početnu stranicu (Slika 5.1.) izlistat će se pet najnovijih obrazovnih članaka iz baze. Na navigacijskoj traci nalazi se tražilica unutar koje je moguće upisati naslov traženog članka. U slučaju da se upiše samo dio naslova članka, članak se neće prikazati korisniku pod rezultatima pretrage. U slučaju da više članaka ima isti naslov svi od članaka s tim naslovom će se prikazati korisniku. Za povratak na početnu stranicu moguće je koristiti mogućnost za povratak web preglednika ili unijeti prazan sadržaj unutar tražilice.



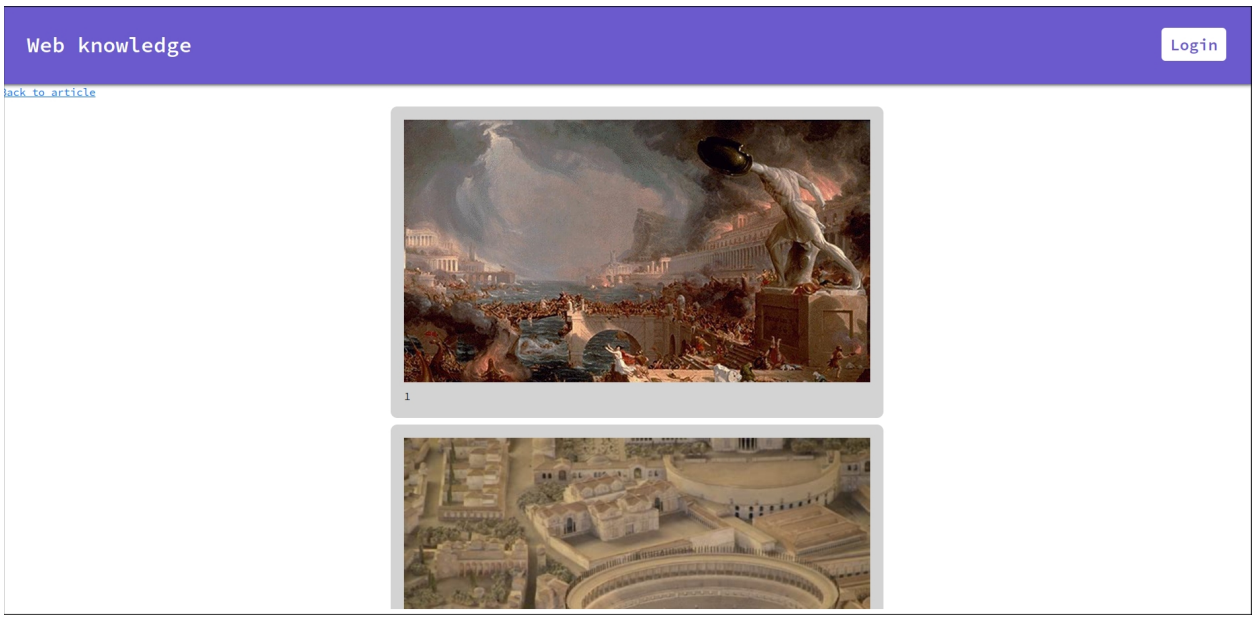
Slika 5.1. Početna stranica

Za pregled sadržaja članka i njegovih blokova potrebno je odabrati željeni članak klikom. Pred korisnikom se prikazuje sadržaj tog članka raspoređen unutar blokova (Slika 5.2.). Slikovne blokove nije moguće pobliže pogledati klikom na njih. Blokovi s videozapisima imaju svoje sučelje. Na vrhu liste svih blokova se nalazi naslov obrazovnog članka. Svaki blok na dnu ima prikazan rezultat glasanja i poveznicu na njegove alternativne blokove koji imaju drugačija objašnjenja. Za povratak na početnu stranicu preporuka je korištenje poveznice u lijevom kutu ispod navigacijske trake.



Slika 5.2. Stranica za detaljan prikaz članka

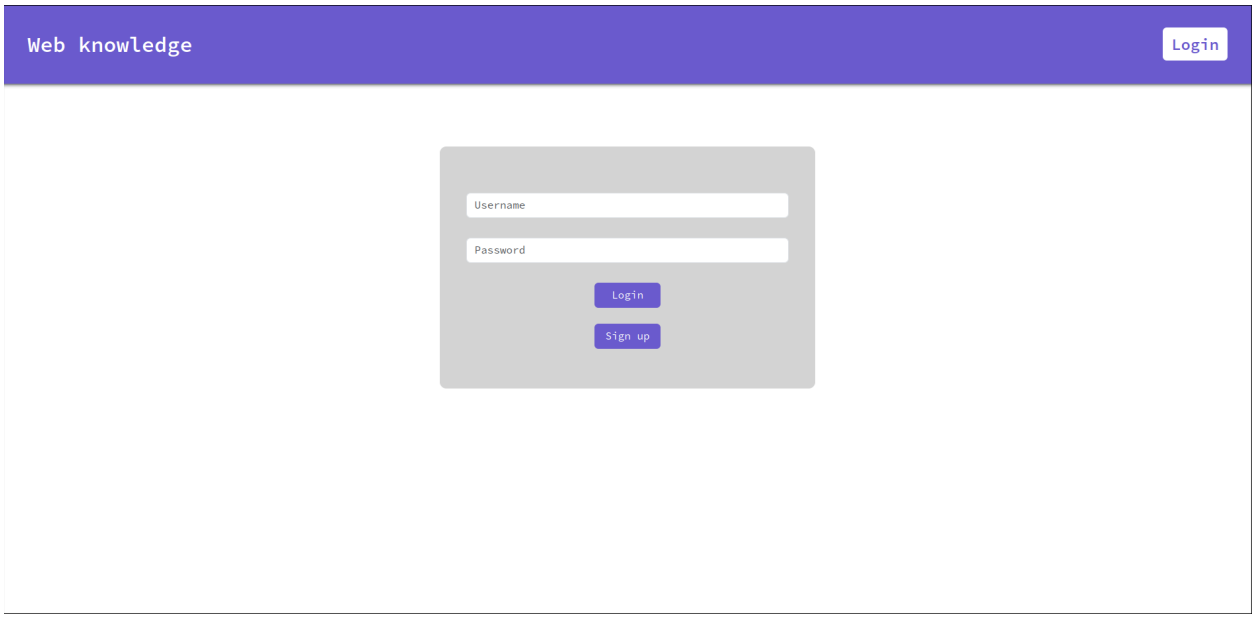
Odabirom poveznice za prikaz alternativnih blokova, korisnika se odvodi na novu stranicu koja prikazuje sve alternativne blokove odabranog bloka i taj sam odabrani blok (Slika 5.3.). Alternativni blokovi su istog tipa kao i odabrani blok, odnosno ako je odabrani blok slikovni blok i alternativni blokovi su također slikovni blokovi. Navedeno vrijedi za tekstualne i blokove s videozapisima. Alternativni blokovi su poredani po rezultatu glasanja. Za vraćanje na detaljni prikaz članka preporuka je upotreba poveznice u gornjem lijevom kutu ispod navigacijske trake.



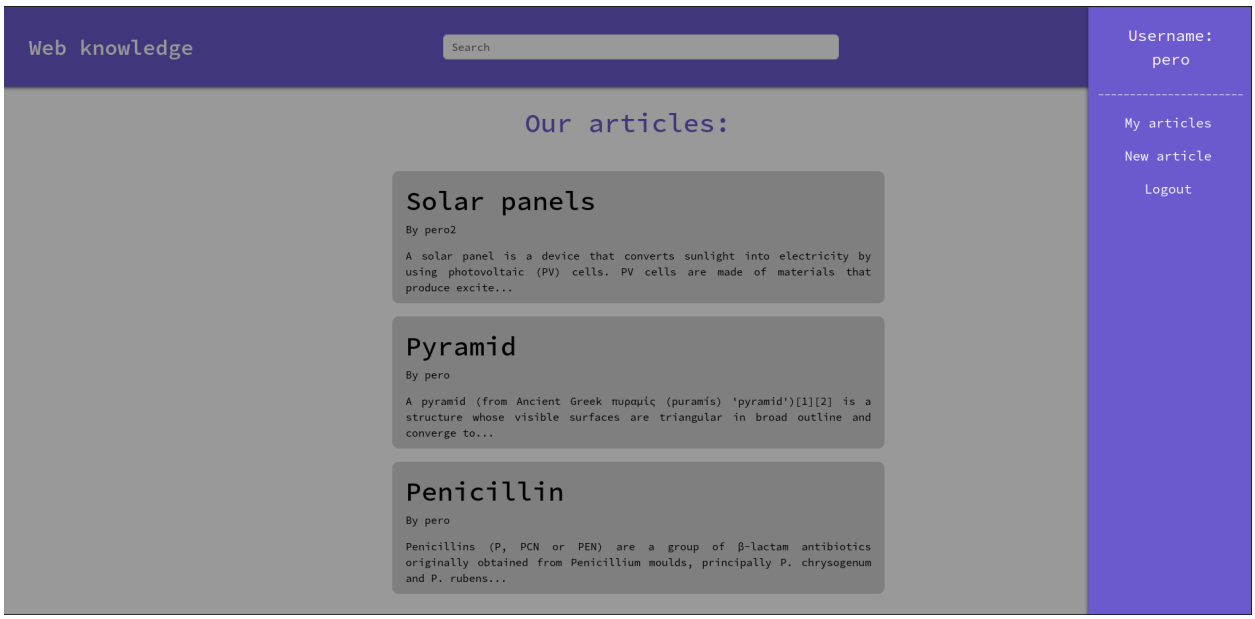
Slika 5.3. Stranica s alternativnim blokovima

5.2. Prijava, odjava i stvaranje novih korisnika

Moguće je prijaviti se kao korisnik klikom na gumb za prijavu na navigacijskoj traci. Nakon klika na gumb za prijavu prikazuje se stranica za prijavu s dva tekstualna polje, jedno za korisničko ime i drugo za upis lozinke (Slika 5.4.). Pritiskom na gumb za prijavu korisnik se prijavljuje i njegova navigacijska traka se mijenja. Sada više nije vidljiv gumb za prijavu, već korisničko ime korisnika. Klikom na korisničko ime otvara se bočni izbornik s dodatnim korisničkim mogućnostima (Slika 5.5.). Mogućnosti korisnika variraju ovisno o ulozi korisnika. Uređivači imaju mogućnost za dodavanje novih članaka, prikaz vlastitih članaka i odjavu. Čitatelji imaju samo mogućnost za odjavu. Administrator ima mogućnost za administrativnu stranicu i odjavu.

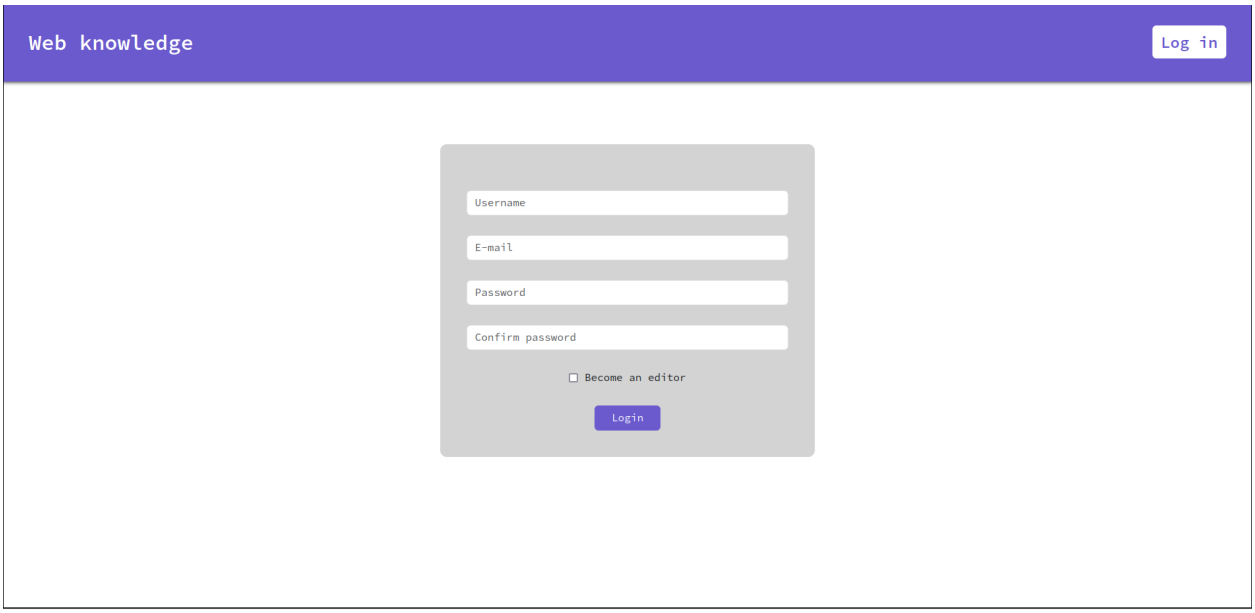


Slika 5.4. Stranica za prijavu



Slika 5.5. Bočni izbornik kod uređivača

Za kreiranje novog korisničkog računa potrebno je odabrati opciju Sign up na zaslonu za prijavu koja korisnika vodi na zaslona za kreiranje novog korisnika (Slika 5.6.). Potrebno je ispuniti četiri tekstualna polja, polje za korisničko ime, adresu za e-poštu te lozinku i potvrdu lozinke. Bitno je da lozinka i potvrda lozinke budu iste, u suprotnom se korisnik neće kreirati. Osim navedenih polja postoji i mogućnost da korisnik postane uređivač gdje je potrebno staviti kvačicu ako novokreirani korisnik želi biti uređivač. Ako se prije spomenuta mogućnost ostavi prazna onda je novokreirani korisnik čitatelj. Pritiskom na gumb kreira se novi korisnik i vraća se na zaslon za prijavu.



Slika 5.6. Stranica za kreiranje korisnika

Za odjavu korisnik samo treba pritisnuti svoje korisničko ime i odabrati mogućnost za odjavu. Odabirom mogućnosti za odjavu korisnik se odjavljuje i šalje na početnu stranicu.

5.3. Dodavanje alternativnih blokova

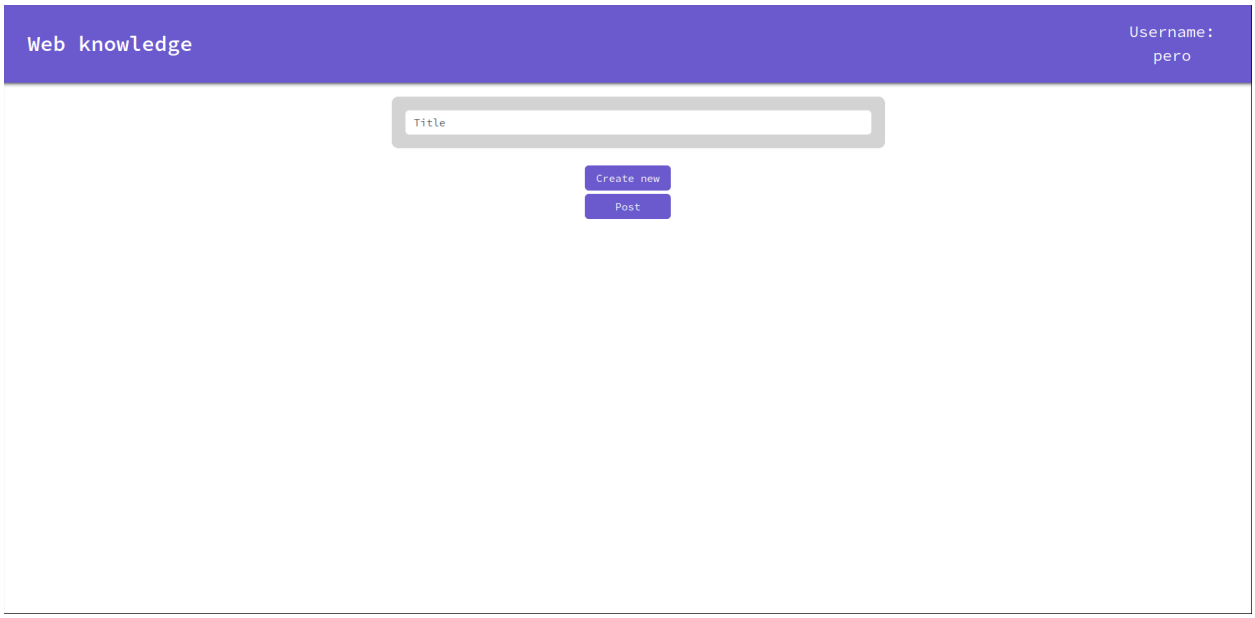
Kako bi se nadodali alternativni blokovi potrebno je učitati stranicu za detaljan prikaz članka kojem se želi dodati alternativni blok i prijaviti se kao uređivač, u suprotnom mogućnost za dodavanje alternativnog bloka neće biti vidljiva. Odabirom mogućnosti otvara se nova stranica unutar koje je moguće unijeti sadržaj bloka (Slika 5.7.). Tip alternativnog bloka koji će se dodati ovisi o tipu bloka kojem se dodaje alternativni blok. Ako se dodaje alternativa tekstualnom bloku, onda će novi blok također biti tekstualni. Isto vrijedi za slikovne i blokove s videozapisima. Pritiskom na gumb za dodavanje alternative korisnika se prebacuje na stranicu za izlistaj alternativnih blokova gdje su izlistani svi blokovi koji su alternativa bloku kojem se dodaje alternativni blok zajedno sa njim samim. Za povratak na detaljan prikaz članka preporuka je korištenje opcije u gornjem lijevom kutu ispod navigacijske trake.

Slika 5.7. Stranica za dodavanje alternativnog tekstualnog bloka

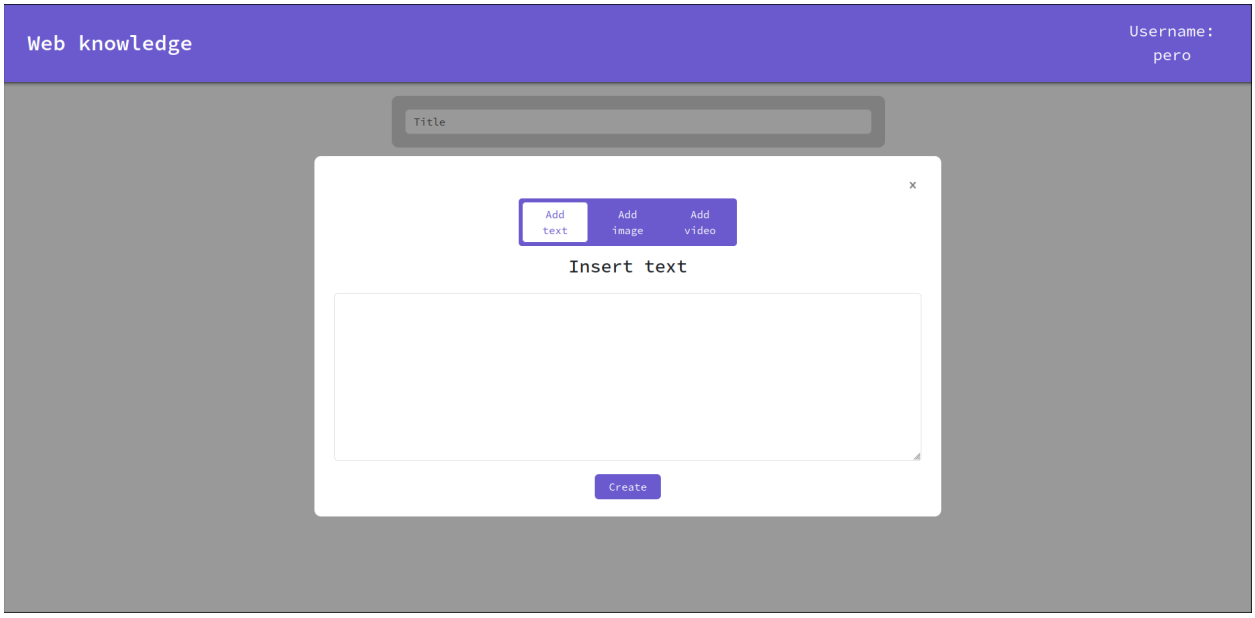
5.4. Dodavanje novih obrazovnih članaka

Za dodavanje novih obrazovnih članaka korisnik treba biti prijavljen kao uređivač, u suprotnom nije moguće pristupiti stranici za dodavanje novih članaka. Stranici za dodavanje članaka se pristupa putem bočnog izbornika uređivača. Dolaskom na stranicu za dodavanje članaka moguće je vidjeti tekstualno polje za naslov koje je obavezno te dva gumba, jedan za dodavanje blokova i

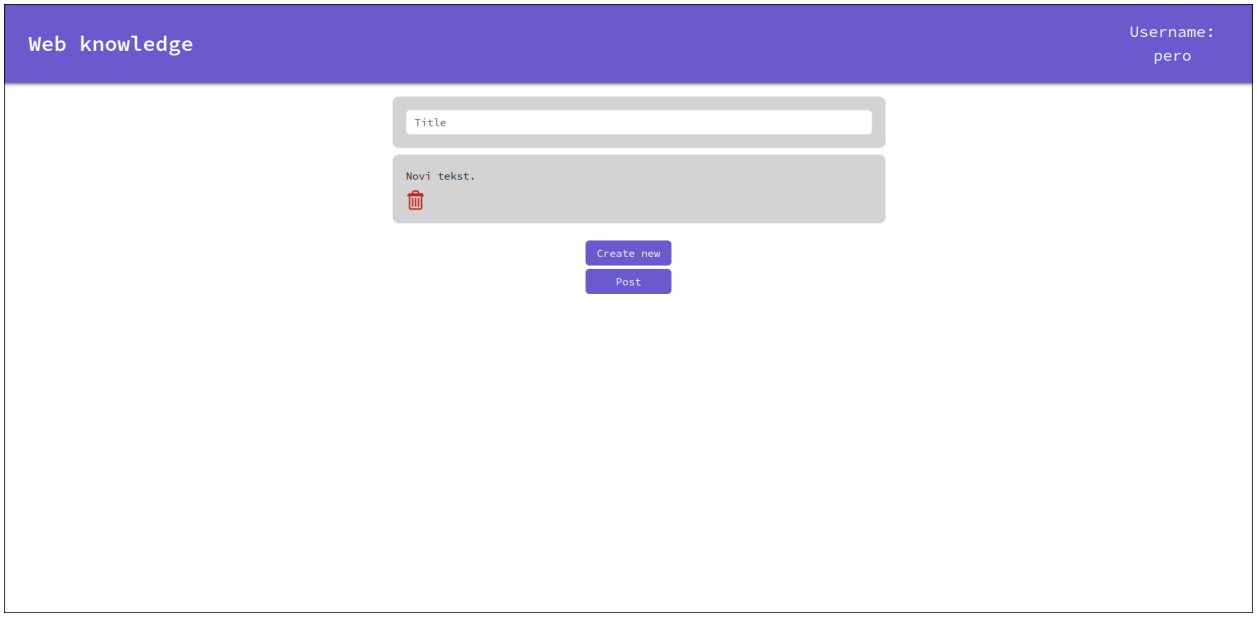
drugi za slanje obrasca s blokovima na poslužitelj (Slika 5.8.). Klikom na gumb za dodavanje bloka otvara se novi prozor unutar kojeg je moguće odabrati tip bloka koji se želi dodati (Slika 5.9.). Za dodavanje slika dopuštene su samo datoteke .jpg, .png i .webp, a za dodavanje videozapisa dozvoljene su samo .avi i .mp4 datoteke. Dodavanjem novog bloka dodani blok se prikazuje na stranici onako kako bi izgledao da se postavi na poslužitelja (Slika 5.10.). Dodane blokove je moguće maknuti samo prije nego se članak postavi na poslužitelja, nakon postavljanja članka više nije moguće uređivanje. Bitan je redoslijed kojim se dodaju blokovi i nije moguće mijenjati njihov redoslijed nakon što se dodaju, ali je moguće izbrisati ih i tako im izmijeniti redoslijed. Slanjem članka na poslužitelja, članak se obrađuje i ako dođe do neke neželjene radnje, kao što je zabranjen tip datoteke, poslužitelj preusmjeruje korisnika natrag na stranicu za unos novog članka. U slučaju da obrada članka prođe korisnik se prebacuje na detaljan prikaz tog članka.



Slika 5.8. Stranica za dodavanje članaka



Slika 5.9. Prozor za dodavanje blokova



Slika 5.10. Prikaz novih blokova

6. ZAKLJUČAK

Prilikom izrade ovog projekta korišten je Django web okvir i jezik Python ze kreiranje funkcionalnosti s poslužiteljske strane te PostgreSQL kao baza podataka. S klijentske strane su korišteni JavaScript za realiziranje funkcionalnosti te HTML i CSS za izradu korisničkog sučelja.

Prije izrada aplikacije potrebno je kreirati virtualno okruženje i aktivirati ga. Unutar tog virtualnog okruženja će se stvarati aplikacija. Projekt je potrebno konfigurirati i spojiti ga s bazom podataka te kreirati administratora. Django aplikacije se temelje na MVC obrascu. Svaka funkcionalnost ima svoj pogled koji rukuje modelima i obavlja svoj posao te obavještava korisnika putem predloška. Django okvir ima svoj model korisnika koji je proširen za potrebe ovog projekta. Za rukovanje korisnicima kreiran je model koji izvršava prijavu, odjavu i kreiranje novog korisnika. Model obrazovnog članka i model bloka su dva temeljna modela ove aplikacije, a dizajnirani su tako da jedan članak ima više blokova. Taj odnos je napravljen uz pomoć modela originalnog i alternativnog bloka. Svaki članak ima više originalnih blokova koji pokazuju na njega, a svaki originalni blok ima više alternativnih blokova. Tako se dobiva struktura stabla kojom je lako manipulirati. Blokovi mogu imati tekstualni i datotečni sadržaj što je problem zato što korisnik može postaviti na poslužitelja neželjen tip datoteke. Kako bi se to spriječilo kreiran je *validator* koji provjerava datoteku po MIME tipu. U slučaju da je MIME tip datoteke dozvoljen datoteka se sprema na poslužitelja, a u slučaju da nije obavještava se korisnika o grešci. Dodavanje članaka i njihovih blokova radi se lokalno na korisnikovu računalu uz pomoć JavaScripta. Važno je izdvojiti dvije funkcije od kojih jedna dodaje kreirani blok u HTML listu, a druga u HTML obrazac. Članak se sprema na poslužitelja kada korisnik pritisne gumb za spremanje članka.

U budućnosti preporuka je poboljšanje tražilice tako da sadržaj tražilice ne mora biti isti kao i naslov, nego da se prikažu članci koji u naslovu imaju sadržaj tražilice. Također preporuka je implementiranje funkcije koja će korisniku prikazati još pet novih članaka na početnoj stranici uz početnih pet članaka.

LITERATURA

[1] Python, Python Software Foundation, dostupno na: https://www.python.org/about/ [18. lipnja 2024.]

[2] D. Amos, D. Bader, J. Jablonski, F. Heisler, Python Basics: A Practical Introduction to Python 3, Real Python, 2012.

[3] Django, Django Software Foundation, dostupno na: https://www.djangoproject.com/foundation/ [18. lipnja 2024.]

[4] PostgreSQL, PostgreSQL razvojni tim, dostupno na: https://www.postgresql.org/about/ [18. lipnja 2024.]

[5] HTML, MDN, dostupno na: https://developer.mozilla.org/en-US/docs/Web/HTML [18. lipnja 2024.]

[6] CSS, MDN, dostupno na: https://developer.mozilla.org/en-US/docs/Web/CSS [18. lipnja 2024.]

[7] JavaScript, MDN, dostupno na: https://developer.mozilla.org/en-US/docs/Web/JavaScript [18. lipnja 2024.]

[8] Django tutorial, Django Software Foundation, dostupno na: https://docs.djangoproject.com/en/5.0/intro/tutorial01/ [20. lipnja 2024.]

[9] Spajanje Django razvojnog okvira i baze podataka, Django Software Foundation, dostupno na: https://docs.djangoproject.com/en/5.0/topics/install/#database [20. lipnja 2024.]

[10] Django autentifikacija korisnika, Django Software Foundation, dostupno na: https://docs.djangoproject.com/en/5.0/topics/auth/default/ [22. lipnja 2024.]

[11] Django proširivanje modela korisnika, Django Software Foundation, dostupno na: https://docs.djangoproject.com/en/5.0/topics/auth/customizing/ [22. lipnja 2024.]

[12] Javascript document objekt, W3 School, dostupno na: https://www.w3schools.com/Js/js_htmldom_document.asp [23. lipnja 2024.]

[13] Slika zaslona podreddita r/programing, dostupno na: https://www.reddit.com/r/programing/ [17. rujna 2024.]

[14] Slika zaslona web aplikacije Google Colab, dostupno na: https://colab.research.google.com/ [17.rujna 2024.]

[15] Slika zaslona članka na Wikipediji, dostupno na: https://en.wikipedia.org/wiki/Rome [17. rujna 2024.]

SAŽETAK

Ključne riječi: blok, Django, model, MVC, obrazovni članak

U ovom radu opisan je postupak izrade društvene mreže za objavljivanje obrazovnih članaka. Za izradu su korišteni Django web okvira i PostgreSQL baza podataka. Django koristi MVC oblikovni obrazac za svoje aplikacije, ali s drugim nazivima (pogled je unutar Django okvira predložak, a kontroler je pogled). Korisnici društvene mreže mogu odabrati jedan od dva tipa korisničkog računa, uređivača ili čitatelja, koji su kreirani uz pomoć proširivanja postojećeg modela korisnika unutar Django okvira. Za prijavu, odjavu i kreiranje korisnika implementiran je poseban model za upravljanje korisnicima. Obrazovni članci se sastoje od blokova, a svaki blok može imati alternativne blokove. Moguće je glasati za alternativne blokove gdje se onaj s najviše glasova prikazuje na stranici obrazovnog članka. Svaki blok može imati tekstualni, slikovni ili video sadržaj. Kako bi se spriječilo postavljanje neželjenih tipova datoteka dodan je validator datoteka. Kreiranje obrazovnih članaka se događa lokalno na korisnikovom računalu i ne sprema se na poslužitelja sve dok ga korisnik ne odluči objaviti.

ABSTRACT

Key words: blok, Django, educational article, model, MVC

This paper describes the process of creating a social network for publishing educational articles. The Django web framework and PostgreSQL database were used in the creation of this project. Django uses the MVC design pattern for its apps, but with different names (the view within the Django framework is the template and the controller is the view). Users can choose one of two user account types, editor or reader, which are created by extending the existing user model within the Django framework. For login, logout, and user creation a special user management model was implemented. Educational articles are made up of blocks and each block can have alternative blocks. It is possible to vote for alternative blocks where the one with the most votes is displayed on the educational article page. Each block can have text, image or video content. To prevent unwanted file types from being uploaded a file validator has been added. The creation of educational articles takes place locally on the user's computer and is not stored on the server until user publishes the article.

PRILOZI

```
$ python -m venv env
$ source env/bin/activate
```

Kod 4.1. Kreiranje i pokretanje virtualnog okruženja u Linuxu

```
python -m venv env
env\Scripts\activate.bat
```

Kod 4.2. Kreiranje i pokretanje virtualnog okruženja u Windows cmd

```
$ django-admin startproject web_knowledge
```

Kod 4.3. Kreiranje Django projekta

```
$ python manage.py runserver
```

Kod 4.4. Pokretanje Django projekta

```
$ python manage.py startapp app
```

Kod 4.5. Kreiranje aplikacije app

```
INSTALLED_APPS = [
      'app.apps.AppConfig',
      'django.contrib.admin',
      'django.contrib.auth',
      'django.contrib.contenttypes',
      'django.contrib.sessions',
      'django.contrib.messages',
      'django.contrib.staticfiles',
]
```

Kod 4.6. Dodavanje aplikacije app unutar settings.py datoteke

```
urlpatterns = [
      path("", include("app.urls")),
      path('admin/', admin.site.urls, name="admin"),
]
```

Kod 4.7. Dodavanje urls.py datoteke aplikacije unutar urls.py datoteke projekta

```
DATABASES = {
      'default': {
            "ENGINE": "django.db.backends.postgresql_psycopg2",
            "NAME": "django_app_db",
            "USER": "django_user",
            "PASSWORD": "django",
            "HOST": "localhost",
            "PORT": "5432",
      }
}
```

Kod 4.8. Konfiguracija baze podataka u settings.py datoteci

```
$ python manage.py createsuperuser
```

Kod 4.9. Kreiranje administratora

```
class Editor(models.Model):
      user = models.OneToOneField(User, on_delete=models.CASCADE)
      created_at = models.DateTimeField(auto_now_add=True)

      def __str__(self):
            return f"username: {self.user.username}, id: {self.user.id}"
```

Kod 4.10. Model uređivača u models.py datoteci

```
class Reader(models.Model):
      user = models.OneToOneField(User, on_delete=models.CASCADE)
      created_at = models.DateTimeField(auto_now_add=True)

      def __str__(self):
            return f"username: {self.user.username}, id: {self.user.id}"
```

Kod 4.11. Model čitatelja u models.py datoteci

```
class UserManager(BaseUserManager):
      def createUser(self, username, email, password=None):
            user = User(username=username, email=email)
            user.set_password(password)
            user.save()
            return user

      def createEditor(self, username, email, password=None):
            user = self.createUser(username=username, email=email,
            password=password)
            editor = Editor(user=user)
            editor.save()

      def createReader(self, username, email, password=None):
            user = self.createUser(username=username, email=email,
            password=password)
            reader = Reader(user=user)
            reader.save()

      def login_user(self, request, username, password):
            user = authenticate(request=request, username=username,
            password=password)
            if user is not None:
                  login(request=request, user=user)
                  return True
            else:
                  return False

      def logout_user(self, request):
            logout(request=request)
```

Kod 4.12. Model za upravljanje korisnicima u models.py datoteci

```
urlpatterns = [
      path("", views.index, name="index"),
      path("myArticles/", views.myArticles, name="myArticles"),
      path("<int:article_id>/", views.articleDetails, name="articleDetails"),
      path("alternatives/<int:block_id>/", views.alternatives,
            name="alternatives"),
      path("alternatives/<int:block_id>/returnToArticle/",
            views.returnToArticle, name="returnToArticle"),
      path("alternatives/<int:block_id>/addTextAlternative/",
            views.addNewTextAlternative, name="addNewTextAlternative"),
      path("alternatives/<int:block_id>/addImageAlternative/",
            views.addNewImageAlternative, name="addNewImageAlternative"),
      path("alternatives/<int:block_id>/addVideoAlternative/",
            views.addNewVideoAlternative, name="addNewVideoAlternative"),
      path("addNewArticle/", views.addNewArticle, name="addNewArticle"),
      path("login/", views.login, name="login"),
      path("logout/", views.logout, name="logout"),
      path("signUp/", views.signUp, name="signUp"),
] + static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

Kod 4.13. urls.py datoteka aplikacije

```
def login(request):
      if request.method == "POST":
            username = request.POST["username"]
            password = request.POST["password"]
            userManager = UserManager()
            is_logedin = userManager.login_user(request, username, password)
            if is_logedin:
                  return redirect(reverse("index"))
            else:
                  messages.error(request=request, message="Login failed,
                              please try again")
                  return redirect(reverse("login"))
      return render(request, "app/login.html")
```

Kod 4.14. Pogled za prijavu korisnika

```
def logout(request):
      userManager = UserManager()
      userManager.logout_user(request)
      return redirect(reverse('index'))
```

Kod 4.15. Pogled za odjavu korisnika

```
def signUp(request):
      if request.method == "POST":
            username = request.POST["username"]
            email = request.POST["email"]
            password1 = request.POST["password1"]
            password2 = request.POST["password2"]
            try:
                  is_editor = request.POST["isEditor"]
            except:
                  is_editor = False
            if password1 == password2:
                  userManager = UserManager()
                  if is_editor:
                        userManager.createEditor(username=username,
                              email=email, password=password1)
                  else:
                        userManager.createReader(username=username,
                              email=email, password=password1)
                  return redirect(reverse("login"))
            messages.error(request=request, message="Incorrect password
                        confirmation")
            return redirect(reverse("signUp"))
      return render(request, "app/signUp.html")
```

Kod 4.16. Pogled za izradu korisnika

```
class Article(models.Model):
      title = models.CharField()
      author = models.ForeignKey(Editor, on_delete=models.CASCADE)
      created_at = models.DateTimeField(auto_now_add=True)

      def __str__(self):
            return f"article_title: {self.title}, article_id: {self.id}"

      def getBlocks(self):
            original_blocks = list(Original_Block
                              .objects.filter(article=self))
            blocks = []
            for block in original_blocks:
                  best = block.getBestBlock()
                  blocks.append(best.base_block)
                  return blocks

      def getInfo(self):
            blocks = self.getBlocks()
            info = ""
            for block in blocks:
                  if block.block_type == "text":
                        info = block.getContent()
                        break
            if len(info) > 150:
                  if info[149] != ".":
                        return f"{info[0:150]}..."
                  else: return f"{info[0:150]}.."
            else:
                  return info
```

Kod 4.17. Model obrazovnog članka

```
class Block(models.Model):
      block_type = models.CharField()
      score = models.IntegerField()
      author = models.ForeignKey(Editor, on_delete=models.CASCADE)
      created_at = models.DateTimeField(auto_now_add=True)

      def __str__(self):
            return f"block_type: {self.block_type}, block_id: {self.id}"

      def getContent(self):
            if self.block_type == "text":
                  content = self.text_content.text
            if self.block_type == "video" or self.block_type == "image":
                  content = self.file_content.file.url
            return content

      def countVotes(self):
            votes = Votes.objects.filter(block=self)
            new_score = 0
            for vote in votes:
                  new_score += vote.score_diff
            self.score = new_score
            self.save()
```

Kod 4.18. Model bloka

```
class Text_Content(models.Model):
      block = models.OneToOneField(Block, on_delete=models.CASCADE)
      text = models.CharField()

      def __str__(self):
            return f"block_id: {self.block.id}, content_id: {self.id}"
```

Kod 4.19. Model tekstualnog sadržaja

```
class File_Content(models.Model):
      block = models.OneToOneField(Block, on_delete=models.CASCADE)
      file = models.FileField(upload_to="files/", validators=[validate_file])

      def __str__(self):
            return f"block_id: {self.block.id}, content_id: {self.id}"
```

Kod 4.20. Model datotečnog sadržaja

```
class Original_Block(models.Model):
      article = models.ForeignKey(Article, on_delete=models.CASCADE)
      base_block = models.OneToOneField(Block, on_delete=models.CASCADE)

      def __str__(self):
            return f"article_title: {self.article.title}, original_block_id:
            {self.id}"

      def getBestBlock(self):
            blocks = Alternative_Block.objects.filter(original_block=self)
            best = self
            for block in blocks:
                  if block.base_block.score > best.base_block.score:
                        best = block
            return best

      def getAlternatives(self):
            alternatives =
                        Alternative_Block.objects.filter(original_block=self)
            alternatives = list(alternatives)
            blocks = []
            blocks.append(self.base_block)
            for alternative in alternatives:
                  blocks.append(alternative.base_block)
            blocks.sort(reverse=True, key=lambda x:x.score)
            return blocks
```

Kod 4.21. Model originalnog bloka

```
class Alternative_Block(models.Model):
      original_block = models.ForeignKey(Original_Block,
                        on_delete=models.CASCADE)
      base_block = models.OneToOneField(Block, on_delete=models.CASCADE)

      def __str__(self):
            return f"original_block_id: {self.original_block.id},
            alternative_block_id: {self.id}"

      def getAlternatives(self):
            alternatives = self.original_block.getAlternatives()
            return alternatives
```

Kod 4.22. Model alternativnog bloka

```
def index(request):
      try:
            title = request.GET["search"]
            if title == "":
                  articles = Article.objects.all().order_by("created_at")
                              .reverse()[0:5]
                  header = "Our articles:"
            else:
                  articles = Article.objects.filter(title=title)
                  header = f"Results fo {title}:"
      except:
            articles = Article.objects.all().order_by("created_at")
                        .reverse()[0:5]
            header = "Our articles:"
            context = { "header": header, "articles": articles }
      return render(request, "app/index.html", context)
```

Kod 4.23. Pogled za izlistaj članaka

```
def myArticles(request):
      articles = Article.objects.filter(author=request.user.editor)
      context = { "header": "Your articles:", "articles": articles }
      return render(request, "app/index.html", context)
```

Kod 4.24. Pogled za izlistaj korisnikovih članaka

```
def articleDetails(request, article_id):
      article = Article.objects.get(id=article_id)
      blocks = article.getBlocks()
      context = { "title": article.title, "author":
                  article.author.user.username, "blocks": blocks }
      return render(request, "app/articleDetails.html", context)
```

Kod 4.25. Pogled za izlistaj blokova članka

```
def alternatives(request, block_id):
      block = Block.objects.get(id=block_id)
      other_block = Original_Block.objects.filter(base_block=block)
      if len(other_block) == 0 :
            other_block = Alternative_Block.objects.get(base_block=block)
            blocks = other_block.getAlternatives()
      else:
            other_block = Original_Block.objects.get(base_block=block)
            blocks = other_block.getAlternatives()
      context = { "block_id": block_id, "blocks": blocks }
      return render(request, "app/alternatives.html", context)
```

Kod 4.26. Pogled za prikaz alternativnih blokova

```
 def returnToArticle(request, block_id):
       original = Original_Block.objects.filter(base_block=block_id)
       if len(original) == 0:
             alternative_block =
                   Alternative_Block.objects.get(base_block=block_id)
             original = alternative_block.original_block
       else:
             original = Original_Block.objects.get(base_block=block_id)
       article_id = original.article.id
       return redirect(reverse('articleDetails', args=[article_id]))
```

Kod 4.27. Pogled za povratak na detaljan prikaz članka

```
class Votes(models.Model):
      reader = models.ForeignKey(Reader, on_delete=models.CASCADE)
      block = models.ForeignKey(Block, on_delete=models.CASCADE)
      score_diff = models.IntegerField()

      def __str__(self):
            return f"reader: {self.reader.user.username}, block_id:
            {self.block.id}"
```

Kod 4.28. Model glasova

```
def validate_file(file):
     mime_types = ['video/x-msvideo', 'video/mp4', 'image/jpeg',
                       'image/png', 'image/webp']
     file_type = magic.from_buffer(file.read(1024), mime=True)
     if not file_type.__str__() in mime_types:
           raise ValidationError("File type is not supported")
```

Kod 4.29. Funkcija za validaciju vrste datoteke

```
    def hasVoted(self, block):
          vote = Votes.objects.filter(reader=self, block=block)
          if len(vote) == 0:
                return False
          else:
                return True

    def hasUpvoted(self, block):
          if self.hasVoted(block=block):
                vote = Votes.objects.get(reader=self, block=block)
                if vote.score_diff > 0:
                      return True
          return False

    def hasDownvoted(self, block):
          if self.hasVoted(block=block):
                vote = Votes.objects.get(reader=self, block=block)
                if vote.score_diff < 0:
                      return True
          return False

    def upvote(self, block):
          if self.hasVoted(block=block):
                vote = Votes.objects.get(reader=self, block=block)
                if vote.score_diff <= 0:
                      vote.score_diff = 1
                      vote.save()
                else:
                      vote.score_diff = 0
                      vote.save()
          else:
                vote = Votes(reader=self, block=block, score_diff=1)
                vote.save()
          block.countVotes()

    def downvote(self, block):
          if self.hasVoted(block=block):
                vote = Votes.objects.get(reader=self, block=block)
                if vote.score_diff >= 0:
                      vote.score_diff = -1
                      vote.save()
                else:
                      vote.score_diff = 0
                      vote.save()
          else:
                vote = Votes(reader=self, block=block, score_diff=-1)
                vote.save()
          block.countVotes()
```

Kod 4.30. Metode dodane u model čitatelja

```
def alternatives(request, block_id):
      if request.method == "POST":
            voted_block_id = request.POST["block_id"]
            voted_block = Block.objects.get(id=voted_block_id)
            vote = int(request.POST["vote"])
            user = request.user
            if vote > 0:
                  user.reader.upvote(voted_block)
            else:
                  user.reader.downvote(voted_block)
            return redirect(reverse('alternatives', args=[block_id]))

      block = Block.objects.get(id=block_id)
      other_block = Original_Block.objects.filter(base_block=block)
      if len(other_block) == 0 :
            other_block = Alternative_Block.objects.get(base_block=block)
            blocks = other_block.getAlternatives()
      else:
            other_block = Original_Block.objects.get(base_block=block)
            blocks = other_block.getAlternatives()
      user = request.user
      score_list = []
      if user.is_authenticated:
            users = Reader.objects.filter(user=user)
            if len(users) > 0:
                  reader = users[0]
                  for block in blocks:
                        if reader.hasUpvoted(block=block):
                              score_list.append(1)
                        elif reader.hasDownvoted(block=block):
                              score_list.append(-1)
                        else:
                              score_list.append(0)
      context = { "block_id": block_id, "blocks": blocks, "score_list":
                  score_list }
      return render(request, "app/alternatives.html", context)
```

Kod 4.31. Nadograđeni pogled za prikaz alternativnih blokova

```
def addNewTextAlternative(request, block_id):
      if request.method == "POST":
            text = request.POST['blockText']
            new_block = Block(block_type="text", author=request.user.editor,
                  score=0)
            content = Text_Content(text=text, block=new_block)
            original = Original_Block.objects.filter(base_block=block_id)
            if len(original) == 0:
                  alternative_block =
                        Alternative_Block.objects.get(base_block=block_id)
                  original = alternative_block.original_block
            else:
                  original = Original_Block.objects.get(base_block=block_id)
            alternative = Alternative_Block(base_block=new_block,
                                        original_block=original)
            new_block.save()
            content.save()
            alternative.save()
            return redirect(reverse('alternatives', args=[block_id]))
      return render(request, "app/addNewTextAlternative.html")
```

Kod 4.32. Pogled za dodavanje alternativnog tekstualnog bloka

```
def addNewArticle(request):
      if request.method == "POST":
            title = request.POST["title"]
            new_article = Article(title=title, author=request.user.editor)
            new_article.save()
            count = int(request.POST['count'])
            blocks = []
            for i in range(count):
                  block_type = request.POST[f"type{i}"]
                  new_block = Block(block_type=block_type,
                                    author=request.user.editor, score=0)
                  new_block.save()
                  if block_type == "text":
                        content = request.POST[f"content{i}"]
                        new_content = Text_Content(text=content,
                                          block=new_block)
                  else:
                        content = request.FILES[f"content{i}"]
                        new_content = File_Content(file=content,
                                          block=new_block)
                        try:
                              new_content.full_clean()
                        except ValidationError:
                              for block in blocks:
                                    block.delete()
                              new_block.delete()
                              new_article.delete()
                              messages.error(request=request,
                                    message="Unsupported file type")
                              return redirect(reverse("addNewArticle"))
                  new_content.save()
                  original = Original_Block(base_block=new_block,
                                    article=new_article)
                  original.save()
                  blocks.append(new_block)
            return redirect(reverse('articleDetails', args=[new_article.id]))
      return render(request, "app/addNewArticle.html")
```

Kod 4.33. Pogled za dodavanje članka

```
function appendBlockToList(blocks, i){
      var block = blocks[i];
      var list = document.getElementById("blockList");
      var listElement = document.createElement("li");
      var divElement = document.createElement("div");
      listElement.setAttribute("id", "li" + String(i));
      if(block.type == "text"){
            divElement.innerHTML = block.content;
      }
      if(block.type == "image"){
            var imgElement = document.createElement("img");
            var src = URL.createObjectURL(block.content.files[0]);
            imgElement.setAttribute("src", src);
            imgElement.setAttribute("alt", "Image");
            imgElement.setAttribute("height", "300");
            divElement.appendChild(imgElement);
      }
      if(block.type == "video"){
            var videoElement = document.createElement("video");
            var src = URL.createObjectURL(block.content.files[0]);
            videoElement.setAttribute("src", src);
            videoElement.setAttribute("controls", "true");
            videoElement.setAttribute("height", "300");
            divElement.appendChild(videoElement);
      }
      divElement.appendChild(document.createElement("br"));
      var deleteButton = document.createElement("button");
      deleteButton.setAttribute("onclick", "remove(blocks, " + String(i) +
      ")");
      deleteButton.setAttribute("id", "button" + String(i));
      deleteButton.innerHTML = "Remove";
      divElement.appendChild(deleteButton);
      listElement.appendChild(divElement); list.appendChild(listElement);
}
```

Kod 4.34. JavaScript funkcija za dodavanje bloka u HTML listu

```
function appendBlockToForm(blocks, i){
      var block = blocks[i];
      let form = document.getElementById("blockForm");
      var inputType = document.createElement("input");
      inputType.setAttribute("type", "hidden");
      inputType.setAttribute("name", "type" + String(i));
      inputType.setAttribute("id", "type" + String(i));
      inputType.setAttribute("value", block.type);
      var inputContent = document.createElement("input");
      if(block.type == "text"){
            inputContent.setAttribute("type", "hidden");
            inputContent.setAttribute("value", block.content);
      }
      else{
            inputContent.setAttribute("type", "file");
            inputContent.setAttribute("style", "display: none");
            inputContent.setAttribute("accept", ".jpg,.png,.jpeg,.webp, .mp4,
            .avi"); block.content.name = "content" + String(i);
            inputContent.files = block.content.files;
      }
      inputContent.setAttribute("name", "content" + String(i));
      inputContent.setAttribute("id", "content" + String(i));
      form.appendChild(inputContent); form.appendChild(inputType);
}
```

Kod 4.35. JavaScript funkcija za dodavanje bloka u HTML obrazac