Usporedba e-trgovina i rješenja za upravljanje sadržajem (CMS) bez grafičkog sučelja

Tomiek, Tomislav

Master's thesis / Diplomski rad

2023

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Zagreb, Faculty of Organization and Informatics / Sveučilište u Zagrebu, Fakultet organizacije i informatike**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:211:988868

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Tomislav Tomiek

Usporedba e-trgovina i rješenja za upravljanje sadržajem (CMS) bez grafičkog sučelja

DIPLOMSKI RAD

Varaždin, 2023.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Tomislav Tomiek

Matični broj: 49653

Studij: P*rogramsko inženjerstvo*

Usporedba e-trgovina i rješenja za upravljanje sadržajem (CMS) bez grafičkog sučelja

DIPLOMSKI RAD

Mentor:

doc. dr. sc. Matija Novak

Varaždin, rujan 2023.

Tomislav Tomiek

Izjava o izvornosti

Izjavljujem da je moj završni/diplomski rad izvorni rezultat mojeg rada te da se u izradi istoga nisam koristio drugim izvorima osim onima koji su u njemu navedeni. Za izradu rada su korištene etički prikladne i prihvatljive metode i tehnike rada.

Autor/Autorica potvrdio/potvrdila prihvaćanjem odredbi u sustavu FOI-radovi

Sažetak

Temeljni cilj ovog rada je usporediti različite gotove alate za upravljanje sadržajem (CMS) bez grafičkog sučelja. Istraživanje će se fokusirati na usporedbu klasičnog CMS-a bez grafičkog sučelja i CMS-a e-trgovine bez grafičkog sučelja. U radu će se detaljno objasniti što je CMS sustav, njegove glavne funkcionalnosti, značajke, prednosti i nedostaci. Također će se razjasniti koncept CMS-a bez grafičkog sučelja, kako radi, njegove funkcionalnosti i prednosti u usporedbi s klasičnim CMS-om. Glavni dio rada bit će usmjeren na usporedbu različitih alata otvorenog koda i komercijalnih alata s ciljem stvaranja klasičnog CMS-a i CMS-a e-trgovine bez grafičkog sučelja. Svaki alat će se temeljito analizirati, prikazati će se njihove glavne funkcionalnosti, cjenik poslovnih planova i opisati njihove prednosti i nedostaci. Dodatno, svaki alat će se ocijeniti na temelju instalacije, jednostavnosti korištenja, kvalitete, dizajna, pruženih funkcionalnosti i fleksibilnosti. Praktični dio rada uključuje kreiranje web stranice pod nazivom „Svijet u oblacima“ koja će se baviti tematikom IoT uređaja. Web stranica će se sastojati od dva dijela: prvi dio će biti web blog, koji će biti implementiran pomoću alata Contentful (klasičan CMS bez grafičkog sučelja), dok će drugi dio biti web trgovina implementirana pomoću alata Commercetools (CMS bez grafičkog sučelja za e-trgovinu). Kroz rad, bit će provedena usporedba između različitih alata te će se dati zaključak o uspoređenim alatima.

Ključne riječi: CMS, headless CMS, CMS e-trgovine, RESTful API, GraphQL, React, Contentful, Commercetools, JavaScript. Tailwind CSS

Sadržaj

1. Uvod

Sustavi za upravljanje sadržajem (CMS) igraju ključnu ulogu u organizaciji i prikazu sadržaja na različitim platformama, poput web aplikacija, desktop aplikacija i mobilnih aplikacija. Oni omogućavaju odvajanje sadržaja od programskog koda aplikacija, što olakšava upravljanje i ažuriranje sadržaja na tim platformama. U početku, klasični sustavi za upravljanje sadržajem pružali su gotova rješenja za web stranice. Korisnici su mogli odabrati dizajn i komponente stranica te bi dobili cijelu web stranicu s integriranim sustavom za upravljanje sadržajem. Primjeri takvih tradicionalnih rješenja uključuju WordPress, Joomla, Drupal, Wix i Squarespace. Međutim, s razvojem tehnologije, pojavile su se različite vrste CMS-ova, uključujući alate u oblaku i privatnim serverima. Ovi alati nude specifične funkcionalnosti i prilagođena rješenja za upravljanje sadržajem, što može stvoriti poteškoće pri odabiru pravog alata. Ovaj diplomski rad istražuje različite vrste CMS-ova i nudi smjernice za donošenje odluke pri odabiru najboljeg alata za upravljanje sadržajem bez grafičkog sučelja. Cilj je pružiti dublji uvid u probleme s kojima se susreću korisnici pri odabiru sustava za upravljanje sadržajem i ponuditi smjernice za rješavanje tih problema.

Prva poglavlja ovog rada detaljno će rasvijetliti koncept sustava za upravljanje sadržajem (CMS), istražujući njegove temeljne karakteristike, glavne funkcionalnosti i princip rada. Bit će analizirane i prednosti i nedostaci ovog sustava kako bi se stvorio cjelovit pregled o njegovoj ulozi i upotrebi.

Nadalje, usredotočit ćemo se na raznolikost CMS-ova, s posebnim naglaskom na klasične CMS-ove bez grafičkog sučelja i CMS-ove bez grafičkog sučelja prilagođene e-trgovini. Ovaj dio rada pružit će dublje razumijevanje komercijalnih i *open-source* alata dostupnih na tržištu, a svaki alat će biti analiziran, uspoređen s drugima i ocijenjen prema različitim kriterijima.

U sljedećem dijelu rada, istražit ćemo metode za dohvaćanje sadržaja iz udaljenih CMS-ova, uz poseban naglasak na popularnim JavaScript razvojnim okruženjima za izradu korisničkog sučelja. Ovaj dio će također naglasiti ključne funkcionalnosti tih okvira.

Konačno, posljednje poglavlje detaljno će opisati proces izrade web aplikacije „Svijet u oblacima“ koja integrira dva CMS-a bez grafičkog sučelja, Contentful i Commercetools. Ovaj praktični dio rada pružit će konkretne primjere kako se ovi sustavi koriste u stvarnom svijetu i kako se mogu integrirati u web aplikaciju.

2. Content Management System

Način kreiranja, upravljanja i dijeljenja sadržaja nije ograničen na 21. stoljeće niti na razdoblje kada je World Wide Web (www) nastao. Prvi pokušaji organizacije sadržaja datiraju unatrag tisućama godina, s ranim primjerima koji potječu iz razdoblja između 3. stoljeća prije nove ere i 3. stoljeća nove ere, posebno u Aleksandrijskoj knjižnici. Tijekom tog vremena, sadržaj se stvarao i dijelio u obliku papirusnih svitaka i kodeksa, čime se omogućavalo zadržavanje i pristup informacijama.

Daljnji značajniji trenutak u povijesti upravljanja sadržaja izbio je dolaskom industrijske revolucije i početkom drastičnog razvoja tehnologije - točnije u trenutku rođenja našeg dragocjenog interneta na početku 90-ih godina 20. stoljeća. Nakon rođenja weba, informacije su se počele gomilati. Izvori informacija, kao i sama njihova količina, s vremenom su sve veći, dok upravljanje informacijama i samim sadržajem postaje sve teže i kritičnije. Potreba za vođenjem sadržaja i upravljanje sadržajem postaje sve veća. Autor Barker [1] govori kako je sredinom 90-ih godina programiranje web stranica bilo kaotično zato što su sve *HyperText Markup Language* (HTML) datoteke bile prenatrpane ugniježđenim oznakama za tablice (eng. *table*) i fontovima. Dodatno govori kako je odvajanje programskog koda od stvarnog sadržaja bilo nemoguće. HTML stranice su u prošlosti sadržavale veliku količinu teksta samoga sadržaja koji se prezentirao na web stranicama te programskog koda za formatiranje sadržaja. Ovo je bilo izrazito nepraktično za čitanje programskog koda, a još više za uređivanje postojećih web stranica.

Na samom početku, pisanje web HTML stranica radilo se pomoću običnog uređivača teksta, što programerima nije olakšavalo posao jer su morali apsolutno sve znati te programski kod pisati ručno. Microsoft je nešto kasnije izdao alat *Visual Source Safe* koji je bio zadužen za upravljanje sadržajem te je ujedno pružao dodatne funkcionalnosti kao što su sigurnosne kopije, odabir verzija i zaključavanje datoteka sadržaja koji se prikazuje na web stranicama.

U današnje vrijeme, programski koda za izgled web stranica (eng. front-end) piše se pomoću popularnih tehnologija poput Reacta, Vue-a i Angulara baziranih na JavaScriptu. Ove tehnologije odvajaju konkretan sadržaj web stranica od stilskog programskog koda, a sam sadržaj se dinamički dohvaća i prikazuje na stranici. Navedeno odvajanje sadržaja i stilskog programskog koda može se ugrubo nazvati sustavom za upravljanje sadržajem (eng. *Content Management System* – CMS). Sam sadržaj se kreira, dodaje, oblikuje i ažurira na jednom mjestu, dok se programski kod za prezentaciju cijelog sadržaja kreira na posve drugom mjestu. Osoba koja dodaje sadržaj ne mora znati ništa o samom stilskom programskom kodu, nego samo dodaje sadržaj koji će se korisnicima prikazivati na web stranicama. CMS alati pomoću

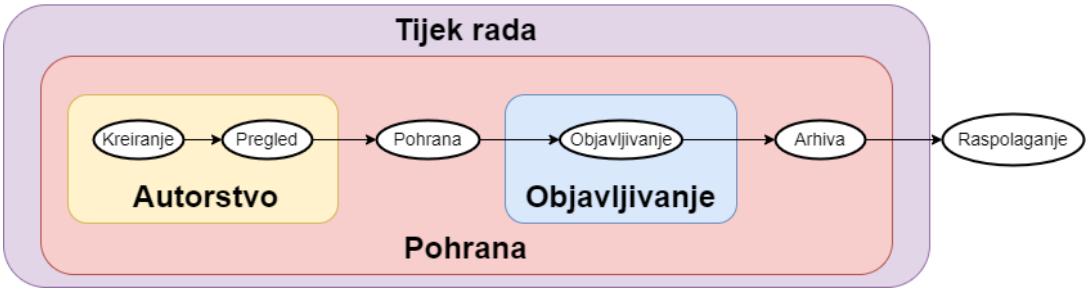
kojih moderatori sadržaja mogu ujedno uređivati dodani tekstualni zapis, a potom ga i pretpregledavati nazivaju se običnim (tradicionalnim) CMS alatima. S druge strane, postoje i CMS alati bez grafičkog sučelja (eng. *Headless CMS*), koji će se u nastavku rada biti razjašnjeni i obrazloženi. Autoru Lončar [2] navodi da su u današnje vrijeme najpopularniji tradicionalni CMS alati: WordPress, Joomia, Drupal, Wix, Shopify i Squarespace; dok bi se među popularnije CMS alate bez grafičkog sučelja svrstali Contentful, Commercetools, Kontent.ai, Strapi, Sanity, Ghost, itd.

Što je onda točno CMS? CMS je programska podrška koja pruža automatizaciju izvršavanja određenih zadataka koji su potrebni za efikasno upravljanje sadržajem. Autor Barker [1] navodi da je CMS višekorisnički poslužiteljski softver. Dodatno, CMS je u konstantnoj interakciji sa sadržajem koji je pohranjen u nekom obliku u repozitoriju koji se može nalaziti na istom poslužitelju kao i sam CMS paket ili se može nalaziti na nekom udaljenom poslužitelju u oblaku.

Svrha CMS sustava je omogućavanje stvaranja novog sadržaja, pružanje mogućnosti uređivanja postojećeg sadržaja, nuđenje mogućnosti uređivanja samoga sadržaja, kao i objavljivanje stvorenog sadržaja kako bi bio dostupan korisnicima. Sam CMS sustav sastoji se od mnogo manjih sustava koji mogu zasebno djelovati, no zajednički rad svih sustava čini jedan CMS sustav. Spomenuti dijelovi mogu biti: sučelje za dodavanje/uređivanje sadržaja, repozitorij, mehanizmi objavljivanja, pregled ili ispis podataka koji se generiraju (primjerice JSON format podataka), itd. Primarni cilj i ključna svrha svakog CMS sustava je upravljanje sadržajem.

Autor Barker [1] izjavljuje da ne postoji ujedinjena i prihvaćena definicija za CMS zato što shvaćanje CMS-a može drastično varirati, što naravno ovisi o pojedinom iskustvu, potrebama i osobnim stavovima programera – CMS posjeduje skup spornih najboljih praksi.

Autori Browning i Lowndes [3] definirali su funkciju CMS sustava i podijelili je u četiri kategorije: autorstvo, tijek rada (eng. *workflow*), pohrana i objavljivanje. Slika 1. prikazuje navedene kategorije funkcije CMS i životni vijek sadržaja.



Slika 1. Funkcije CMS-a i životni ciklus sadržaja [3]

Prva kategorija (autorstvo) podrazumijeva kreiranje samog sadržaja, gdje sadržaj može biti jedna rečenica ili velika količina teksta. Nakon kreiranja sadržaja, sadržaj se dodatno provjerava u svrhu ispravljanja eventualnih grešaka.

Druga kategorija je tijek rada koji sadrži korake između autorstva i publikacije. Ovdje autori navode primjere koraka tijeka rada, a to su provjera veze i pregled/objava od strane menadžera ili projektnog tima. Tijek rada je najčešće izvanmrežni proces koji nije ugrađen u softverske procese.

Kategorija pohrane podrazumijeva pohranu autorskog sadržaja na repozitorij. Dodatno, koristi se i verzioniranje sadržaja, u svrhu mogućnosti vraćanja na prijašnje verzije. Pohrana autorskih sadržaja se vrše u bazi podataka ili pomoću *eXtensible Markup Language* (XML) datoteka.

Posljednja kategorija je objavljivanje, što je zapravo proces dostavljanja pohranjenog sadržaja korisnicima. Ovdje je najčešće riječ o dohvaćanju i prikazu autorskih sadržaja na web stranicama korisnicima u obliku HTML-a. Postoje i manje popularni načini dostavljanja sadržaja, primjerice: elektronička pošta, Adobe *Portable Document Format* (PDF) dokument, itd.

2.1. Sadržaj Content Management System-a

Prije samoga početka kreiranja *Content Management System*-a potrebno je definirati sadržaj CMS-a. Mnogi smatraju da je sadržaj CMS-a samo običan podatak, informacija, sadržaj ili čak znanje. Potrebno je prepoznati što je točno sadržaj CMS-a, a što podatak ili informacija. Potrebno je razlikovati je li upravljanje sadržajem isto ili drugačije od upravljanja ostalim tipovima podataka, gdje autor knjige Barker [1] govori kako je ključna razlika između upravljanja sadržajem i upravljanja ostalih tipova podataka u tome što je kreiranje sadržaja znatno drugačije od samog korištenja prethodno kreiranog sadržaja. Općenito, sadržaj CMS-a ili bilo koji drugi sadržaj web stranice ovisi o osobi sadržaja koja kreira sadržaj zato što ona odlučuje koja će biti tema samog sadržaja, za koju publiku je sadržaj namijenjen, pod kojim kutom je potrebno prići sadržaju, koliko velik i opsežan sadržaj treba biti te koja je potreba sadržaja za neki dodatni medijski zapis.

Dobar i kvalitetan sadržaj trebao bi biti onaj koji se konstanto koristi i koji će se i u skoroj budućnosti koristiti i referencirati, no nažalost većina web sadržaja nema dugi životni vijek i uopće se ne koristi u budućnosti. Primarni cilj kreiranja sadržaja bi trebao biti njegovo korištenje u budućnosti, na način da se može koristiti desetljećima ili možda čak i stoljećima,

a da i dalje bude koristan. Osim njegove iskoristivosti u budućnosti, sadržaj bi trebao ispunjavati potrebe korisnika i podržavati glavni poslovni cilj.

Pod potrebe korisnika podrazumijeva se zanimanje korisnika za određeni sadržaji, čitanje tog sadržaja te kasnije ponovno vraćanje na isti sadržaj kako bi dobio još željenog sadržaja. Suprotno tome, kada je sadržaj loše strukturiran, nepotrebno velik ili netočan, korisnik će jednostavno doći, na brzinu pročitati i shvatiti da je traženi sadržaj njemu nepotreban te mu se nikad više neće vratiti. Autori Halvorson i Rach [4] predstavili su pet činjenica koje govore o poteškoćama popravljanja loše kreiranih sadržaja.

Prva činjenica koju su postavili bila je da se sadržaj predstavlja kao obična roba koja se može kupiti čak i za četiri dolara. Pod ovom činjenicom smatrali su da čim je sadržaja više, to je bolje. Razlog tome bili su bolji pouzdaniji alati za traženje njihovih sadržaja na webu. Ostali razlozi bili bi povećanje vrijednosti sadržaja čitateljima i povećanje konkurentske diferencijale, čime bi se privlačilo više korisnika na njihove sadržaje u odnosu na konkurente. Navedena činjenica koja uzrokuje problematičnost za povećanje kvalitete sadržaja leži i u tome što se sadržaj generirao od strane korisnika, gdje su korisnici bili urednici sadržaja, što zvuči jako primamljivo, ali nekonzistentno. Korisnici mogu kreirati sadržaj s iznimno malim financijskim troškovima, ali ne žele nepotrebno ulagati svoj trud u takve ili slične situacije.

Sljedeća činjenica koja se izjavljuje je nedovoljno vremena uloženo u kreiranje plana za definiranje sadržaja. Ovdje se misli da urednici sadržaja imaju premalen vremenski rok za stvaranje sadržaja, a pošto je vrijeme kritično, urednici najčešće su pod stresom i stvarali sadržaj tek toliko da imaju nešto za pokazati svojim poslovođama. Na taj način nastaje vrlo loš i nekvalitetan sadržaj.

Treća činjenica koja se objašnjava je „Radimo smrtne presude“. Autori knjige govore kako ljudi rade pogrešne presude te od svojih zaposlenika očekuju da će se sve izvršiti na vrijeme i kvalitetno, ali u realnosti stvaranje sadržaja uopće nije započelo ili je sadržaj jako niske kvalitete.

Posljednje dvije činjenice koje se spominju su: „Sadržaj je politički“ i „Previše je toga, nikada nećemo to prebroditi“.

2.2. Upravljanje sadržajem CMS-a

Sustav za upravljanje sadržajem je softverski paket ili alat koji se u potpunosti može pružati u oblaku putem web stranica ili integrirati na vlastite servere, gdje mu se pristupa pomoću razvijene aplikacije ili putem web preglednika. Kao što je već rečeno, CMS sustavi pružaju određenu razinu automatizacije pojedinih procesa za učinkovito upravljanje kreiranim

sadržajem te urednicima olakšavaju proces kreiranja novih sadržaja. Razvojnim inženjerima CMS sustava pruža se mogućnost učinkovitog i efikasnog kreiranja modela sadržaja. Dodatno, CMS sustav sastoji se od više manjih softvera, koji mogu zasebno djelovati te su u interakciji s kompletnim pohranjenim sadržajem koji se može nalaziti u željenoj bazi podataka ili repozitoriju. Baza podataka ili repozitorij može se nalaziti na istoj lokaciji kao i CMS sustav ili na nekom udaljenom serveru ili web servisu. Najvažnija funkcionalnost svakog CMS sustava je mogućnost efikasnog upravljanja svim kreiranim sadržajima. Autor Barker [1] navodi kako upravljanje CMS-om podrazumijeva da sustav omogućava:

- kreiranje novih željenih sadržaja na temelju prethodno kreiranog modela sadržaja;
- ažuriranje postojećih sadržaja;
- vraćati sadržaja na prethodne verzije i/ili vraćanje cijelog CMS sustava na prethodnu verziju;
- dodatno uređivanje tekstova sadržaja (font, veličina slova, podebljani tekst, tekst u kurzivu, itd.);
- objavljivanje kreiranog sadržaja, kako bi bio dostupan ostalim korisnicima.

Autor Boiko [5], navodi kako bi pri procesu upravljanja bilo kojim CMS sustavom, sustav trebao obavještavati korisnike o sljedećim važnijim informacijama:

- detalji o modelu sadržaja – sadrži popis svih komponenata i informacija o njihovim trenutnim životnim ciklusima;
- informacije o uspješnosti rada zaposlenika – pomoću kojih se doznaju razna zabušavanja i uska grla kod realizacije projekata;
- informacije o načinu iskorištavanja objavljenih sadržaja – saznaje se gdje i zašto se koristi objavljeni sadržaj, kao i informacije o neiskorištenom sadržaju koji je potrebno odstraniti;
- popis dopuštenih osoba i njihovih uloga – pomoću kojih se doznaje tko je za što zadužen i tko je najviše pridonio poduzeću.

Autor Boiko [5] je naveo da proces upravljanja CMS-om sadrži četiri kategorije:

- repozitorij – zadužen za pohranu sadržaja;
- administracija – zadužena za početnu konfiguraciju CMS sustava i postavljenje svih ostalih postavki sustava;
- tijek rada – gdje se kreiraju svi koraci modela sadržaja koji su potrebni kod ispunjavanja stvarnih sadržaja;
- kreiranje veza – pomoću kojih se dohvaćaju potrebni sadržaji iz repozitorija.

2.3. Tipovi podsustava CMS-a

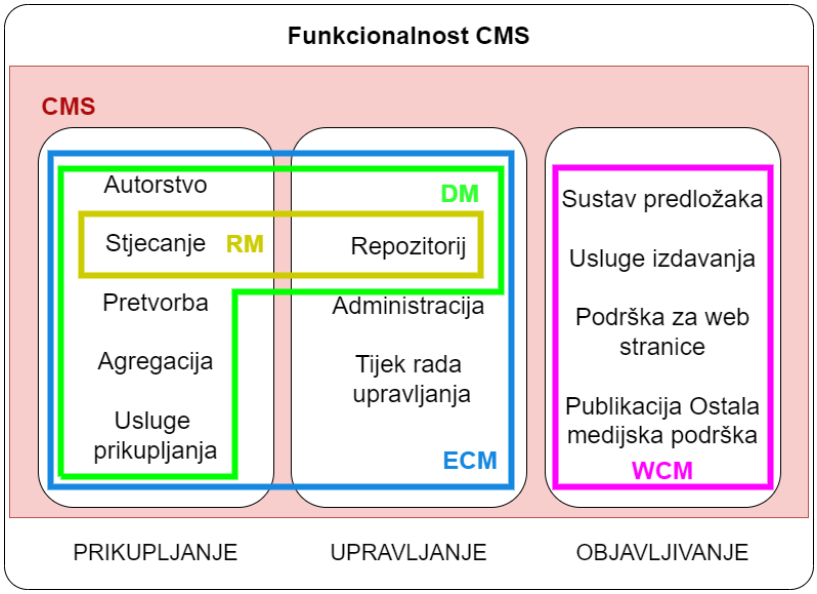
Ranije je bilo rečeno da se CMS sastoji od manjih sustava koji mogu i pojedinačno djelovati. Navedeni spomenuti manji sustavi su zapravo tipovi podsustava cijeloga CMS-a, a prema autoricama Benevolo i Negri [6] to bi bili:

- upravljanje sadržajem (eng. *Content management* – CM) – u teoriji prema autoricama CM je zapravo CMS koji sadrži sve ostale tipove podsustava CMS-a;
- upravljanje sadržajem web-a (eng. *Web Content Management* – WCM) – sustav koji vodi računa o velikoj količini informacija te njihovoj organizaciji i ažuriranju na web stranicama. WCM sustav je zapravo zadužen za upravljanje sadržajem bogatog teksta koji se prezentira na web stranicama i koji može biti tekst, audio, slika, videozapis, gif , kao i ostali tipovi multimedijskog zapisa;
- upravljanje dokumentima (eng. *Document management* – DM) – najstariji sustav za upravljanjem sadržajem (točnije dokumentima) koji olakšava stvaranje i pohranjivanje dokumenata u posebne repozitorije, gdje se poštuju unaprijed definirana pravila i meta-podaci;
- upravljanje zapisima (eng. *Record Management* – RM) – sustavi koji mogu upravljati velikom količinom podataka koji se pohranjuju u papirnatom obliku ili elektroničkom formatu. Dodatno osigurava sigurnu metodu pohrane podataka i pristupa podacima, vezu prema bazi podataka i konzistentna pravila za čuvanje dokumenata;
- upravljanje poslovnim sadržajem (eng. *Enterprise Content Management* – ECM) – sustav koji obuhvaća upravljanje svim tipovima podataka pojedinih poduzeća, kao što su papirnati oblici, izvješća, web stranice i sva digitalna imovina. ECM dodatno obuhvaća strategije, instrumente, procese i znanja korištene za upravljanje informacijama. Dvije glavne značajke ECM sustava su prikupljanje informacija i upravljanje istima.

Slika 2. prikazuje detaljniju podjelu CMS sustava na tipove podsustava, te dodatno prikazuje sve funkcionalnosti pojedinih podsustava koje se mogu dodatno razvrstati kroz sljedeće tri kategorije: prikupljanje, upravljanje i objavljivanje.

Kao što je već ranije bilo spomenuto, CMS se ne može definirati na jedinstven način pa je tako i autor Barker [1] naveo malo drugačije tipove podsustava CMS-a. Podijelio je sustav CMS na četiri podsustava: WCM, ECM, RM i upravljanje digitalnom imovinom (eng. *Digital Asset Management* – DAM). Barker je zamijenio gore navedeni podsustav *Document Management* novim podsustavom DAM koji je slični podsustavu WCM. DAM je zadužen za manipulaciju i upravljanje bogatim tekstom (kao što su slike, audio, video itd.). Dodatno, DAM

se koristi kod prezentacije podataka na web stranicama i ističe se kod meta-podataka, što je ujedno i zajedničko i kod definicije autorica Benevolo i Negri [6] za podsustav WCM.



Slika 2. Podjela CMS na tipove podsustava i njihove funkcionalnosti [6]

2.4. Prednosti i nedostaci

Prednosti koje CMS pruža su izrazito značajne, jer je pomoću CMS-a moguće upravljati cijelim sadržajem na jednostavan način. CMS pruža precizan način za praćenje stvorenog sadržaja i stanja u kojem se nalazi, mogućnost praćenja pristupa određenom sadržaju te međuovisnost i povezanost više različitih sadržaja. Barker [1] govori kako CMS pruža glavne kontrolne funkcionalnosti, kao što su:

- privilegije dostupnosti – tko može stvarati, uređivati ili brisati pojedine sadržaje;
- upravljanje stanjem i tijekom rada – navodi se trenutno stanje pojedinih sadržaja, primjerice: sadržaj je još u izradi, pušten je u pogon ili je bio nedovoljno razrađen te je stoga obrisan.
- verzioniranje sadržaja – koliko se i na koji način sadržaj mijenjao tijekom vremena, a dodatno nudi mogućnost vraćanja na prethodne verzije rada;
- upravljanje međuovisnostima – govori kako će se sadržaj ponašati kad se obriše njegova ovisnost prema drugim sadržajima te kolika je iskorištenost sadržaja;
- traženje i organizacija – predstavlja konkretno pronalaženje pojedinog sadržaja te njegovu grupaciju s ostalim sadržajima i/ili grupama.

Osim gore navedenih prednosti, CMS posjeduje i dodatne prednosti poput ponovne iskoristivosti postojećih sadržaja (eng. *Reusability*), što znači da se sadržaj CMS-a može višestruko i na bilo koji način iskoristiti kako bi se smanjilo dupliciranje koda i financijski troškovi te kako bi se uštedjelo na vremenu. CMS omogućuje i lagano uređivanje postojećeg sadržaja na jednom mjestu, pri čemu bi se izvršene promjene ažurirale na svim mjestima gdje se navedeni sadržaj koristi.

Najveći razlog uvođenja CMS-a bila bi dodatna efikasnost kod stvaranja ili uređivanja sadržaja. Urednici sadržaja pojedine web stranice najčešće žele stvoriti ili urediti više različitih sadržaja u što kraćem vremenskom roku. Ako je dobro stvoreno sučelje za uređivanje sadržaja te ako je ono dobro strukturirano, CMS urednicima pruža preciznost, efikasnost i konzistentnost kod stvaranja i uređivanja sadržaja.

Nadalje, CMS pruža automatizaciju i agregaciju. Na taj se način programerima pruža mogućnost prikazivanja sadržaja na razne načine u raznim formatima. Primjerice, pojedine je sadržaje moguće pretvarati u PDF zapise koji se mogu preuzeti i pohraniti. Nudi se i mogućnost objavljivanja sadržaja u realnom vremena na temelju korisničkih želja i specifikacija.

Prema autorima Browningu i Lowndesu [3], svaki CMS sustav trebao bi posjedovati tri glavne funkcionalnosti: verzioniranje sadržaja, tijek rada sadržaja i integraciju. Autori su razjasnili da pomoću tijeka rada sadržaja osiguravaju da sadržaj prolazi kroz postupak procjene, pregleda i/ili proces osiguranje kvalitete. Kod integracije je bitno da se sadržaj pohranjuje odvojeno od prezentacijskog koda kako bi se mogao višestruko koristiti kod istih ili različitih web stranica ili iskoristiti u drugačijem medijskom tipu, primjerice: kod generiranja PDF dokumenata, ispisa u oblike statistike ili računa, ili kod slanja elektroničke pošte. Pomoću navedenih triju funkcionalnosti moguće je izgraditi stabilan CMS sustav koji može biti kompleksan, opširan, dinamičan, te imati mogućnost uređivanja s više autora sadržaja.

Sljedeće natuknice prema autoru Davisu [7] su negativne strane korištenja CMS sustava:

- smanjena fleksibilnost;
- povećanjem kompleksnosti i specifičnosti aplikacije, vjerojatnost kompatibilnosti pojedinog CMS se smanjuje;
- ograničenost CMS sustava;
- mogućnost pogrešaka, koje mogu biti veće ako se je CMS pogrešno konfiguriran;
- veća upotreba resursa od strane poslužitelja;
- dodatna kompleksnost sustava, što zahtjeva dodatnu obuku programera i urednika sadržaja;

- potreba za dodatnim ažuriranjem CMS sustava;
- nemogućnost pregleda stvarnog izgleda sadržaja kod CMS-a bez grafičkog sučelja.

Manje izražena mana kod CMS sustava bila bi to što CMS ne generira sadržaj koji se korisniku prikazuje na ekranu, već upravlja sadržajima koje urednici ispunjavaju te iste konfigurira.

Sljedeća stvar koja bi se trebala podrazumijevati je marketing – CMS ne nudi nikakvu mogućnost automatskog marketinga kreiranih sadržaja. Posljednja stvar koju spominje autor Barker [1], koja se događa rjeđe, je da se urednicima sadržaja pruža previše bogat alat za uređivanje zbog čega su urednici prisiljeni koristiti puni potencijal alata. Time je moguće da tekst sadržaja sadrži previše podebljanog teksta ili previše teksta pisanog u kurzivu i slično, što dovodi do nekonzistentnog sadržaja.

3. Vrste CMS-a

Kao što postoje različiti sportovi – poput nogometa, tenisa, košarke, rukometa, itd., svaki se sport dodatno klasificira prema različitim timovima. Svaki tim u određenom sportu dodatno klasificiramo prema broju pobjeda te tako dobivamo popis najboljih timova. Dodatno, svaki tim možemo detaljnije specificirati – primjerice, prema najboljim taktikama, najboljim igračima, najboljim asistencijama, najboljim rekordima, itd. Isto tako možemo klasificirati i CMS sustave prema njihovoj glavnoj specijalizaciji i svrsi – točnije, prema njihovoj vrsti upotrebe. Ovisno o tome što točno klijenti ili korisnici zahtijevaju i specificiraju, odabire se određena vrsta CMS-a ili se od nule implementira novi CMS sustav koji će točno odgovarati klijentovoj specifikaciji. Postoji već velik broj web ili desktop verzija alata koji odgovaraju određenim vrstama CMS-a. Vrste CMS sustava dijelimo na:

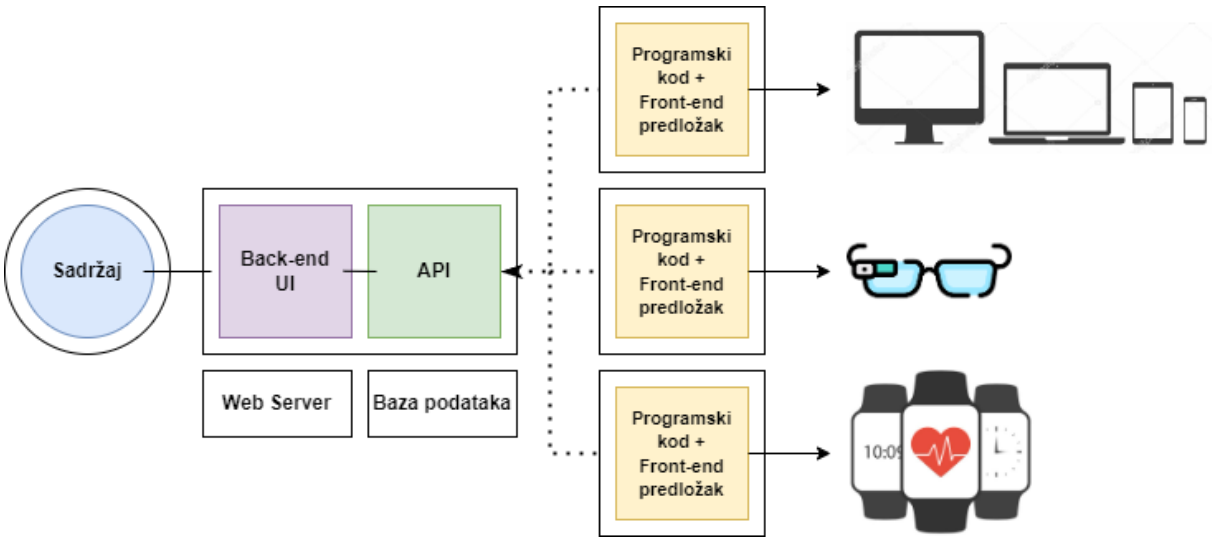
- tradicionalni CMS;
- CMS bez grafičkog sučelja (eng. *Headless CMS*);
- razdvojeni CMS (eng. *Decoupled CMS*);
- CMS za poduzeća (eng. *Enterprise CMS*);
- komponentni CMS (eng. *Component CMS*);
- CMS za e-trgovinu (eng. *e-commerce CMS*).

Sljedeći dijelovi rada će se primarno fokusirati na sustave za upravljanje sadržajem bez grafičkog sučelja. Takvi sustavi koriste web servise pomoću kojih se web stranicama pružaju potrebni podaci. Najpopularniji i najjednostavniji način za pružanje i dohvaćanje podataka na webu je uz pomoć RESTful (eng. *Representational State Transfer*) web servisa. Drugi popularni tip web servisa koji se koristi kod sustava za upravljanje sadržajem bez grafičkog sučelja je GraphQL.

3.1. CMS bez grafičkog sustava

Za CMS bez grafičkog sučelja ugrubo se može reći da odvaja cijeli sadržaj od prezentacijskog sloja. Korisnicima omogućuje pisanje, definiranje, grubo oblikovanje, modificiranje i objavljivanje željenog sadržaja na jednom mjestu, dok se prikaz pohranjenog sadržaja dohvaća putem više različitih kanala i prikazuje na front-endu na posve drugome mjestu. Nudi mogućnost prikaza istog sadržaja na različitim web preglednicima, mobilnim aplikacijama i ostalim uređajima, npr. Internet Stvari (eng. Internet of Things, IoT) koji naravno moraju biti spojeni na Internet jer se sadržaj dohvaća kroz razna sučelja za programiranje aplikacija (eng. *Application Programming Interface* – API) web servisa.

Korisnici i razvojni inženjeri ovog CMS sustava mogu odabrati bilo koju front-end tehnologiju za prezentaciju sadržaja na ekrane, gdje u posljednje vrijeme Angular, React i Vue dobivaju veliku popularnost kod izrade web aplikacija. Slika 3. prikazuje način rada CMS-a bez grafičkog sučelja. Headless CMS je izvrstan za tvrtke koje nude svoj sadržaj na mnogim i raznim IoT uređajima pošto je headless CMS vrlo fleksibilan kod dohvaćanja sadržaja. Headless CMS je izvrstan za tvrtke koje nude e-trgovinu, medijske izdavače i programske aplikacije za mobitele.



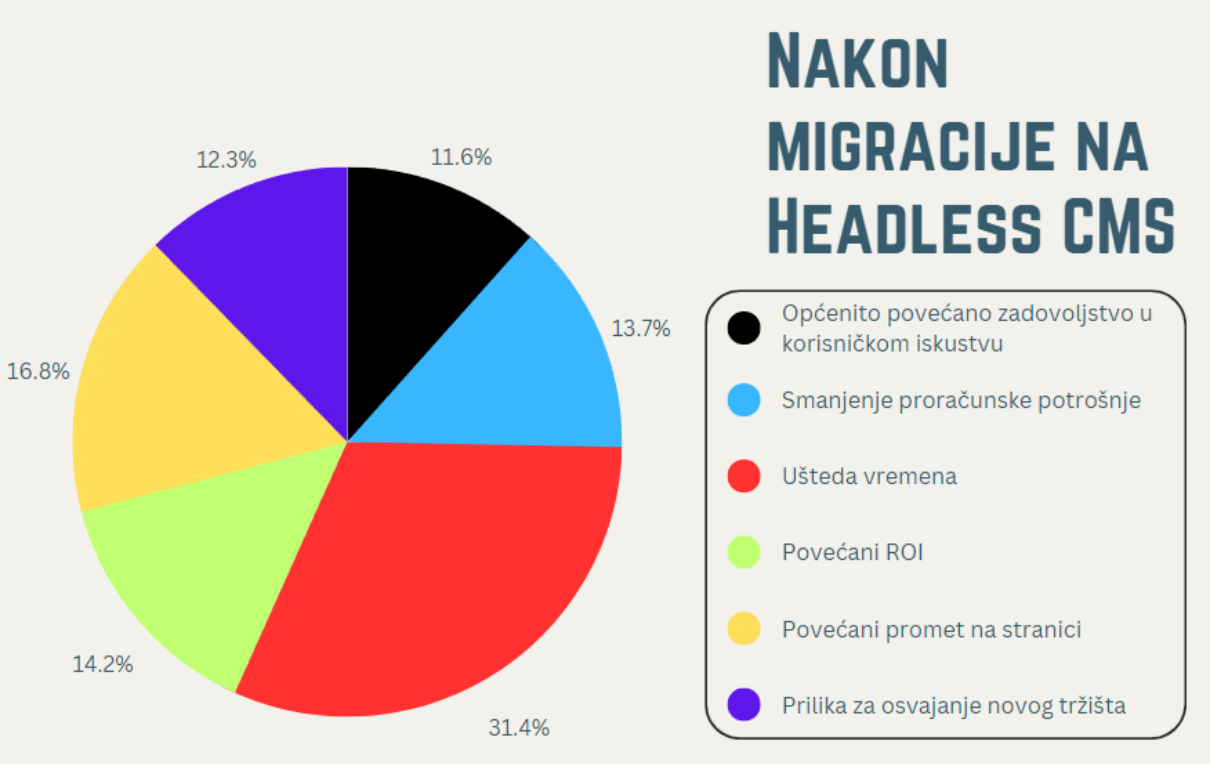
Slika 3. Prikaz načina rada Headless CMS-a [8]

Prednosti koje nosi Headless CMS prema tvrtki ThemeSelection i autorima Acharya i Signo [8], [9], [10] bile bi sljedeće:

- povećana prilagodljivost – jednostavnije postavljanje konfiguracije CMS sustava i smanjenje potrebe za upravljanjem infrastrukturom, što dovodi do bržeg pokretanja CMS sustava i programerima ostavlja više vremena za detaljniju prilagodbu CMS sustava;
- fleksibilnost – front-end programeri imaju potpunu kontrolu nad prikazom sadržaja i njegovim dizajnom, za razliku od limitiranih komponenti tradicionalnih CMS-ova za prikaz sadržaja;
- povećana sigurnost – nudi se bolja sigurnost nego u slučaju tradicionalnog CMS-a pošto je sadržaj odvojen od front-enda, a eventualna sigurnosna prijetnja na jednom sustavu ne utječe na drugi sustav. Dodatna sigurnost postiže se sigurnosnim API-ima (gdje se kodira sadržaj) pomoću kojih se dohvaća sadržaj;

- izvrsna kompatibilnost – mogućnost objavljivanja istog sadržaja na sve željene IoT uređaje bez dodatnih troškova za sadržaj;
- jednostavnije održavanje – iz razloga što je sadržaj odvojen od prezentacijskog koda, automatski se postiže jednostavnije i lakše održavanje samog sadržaja i programskog koda front-enda.

Slika 4. prikazuje pojedine prednosti koje su kompanije postigle migracijom na Headless CMS sustav.



Slika 4. Prikaz postignutih prednosti prelaskom na Headless CMS [11]

Headless CMS je dobio ime zato što odvaja back-end od front-end kod upravljanja sadržajem, pri čemu je front-end „glava“ (eng. *head*), a back-end „tijelo“ (eng. *body*). Neki headless CMS sustavi mogu dodatno biti klasificirani kao „softver kao usluga“ (eng. *Software as a Service* – SaaS) temeljen na oblaku pošto razne tvrtke nude svoje usluge CMS sustava u oblacima. Konkretni primjeri Headless CMS programskih rješenja otvorenog koda (eng. *open source*) bili bi Strapi, Directus i Ghost, dok bi primjeri komercijalnih rješenja bili Sanity, Contentful i Hygraph.

3.1.1. Komercijalna rješenja za CMS bez grafičkog sustava

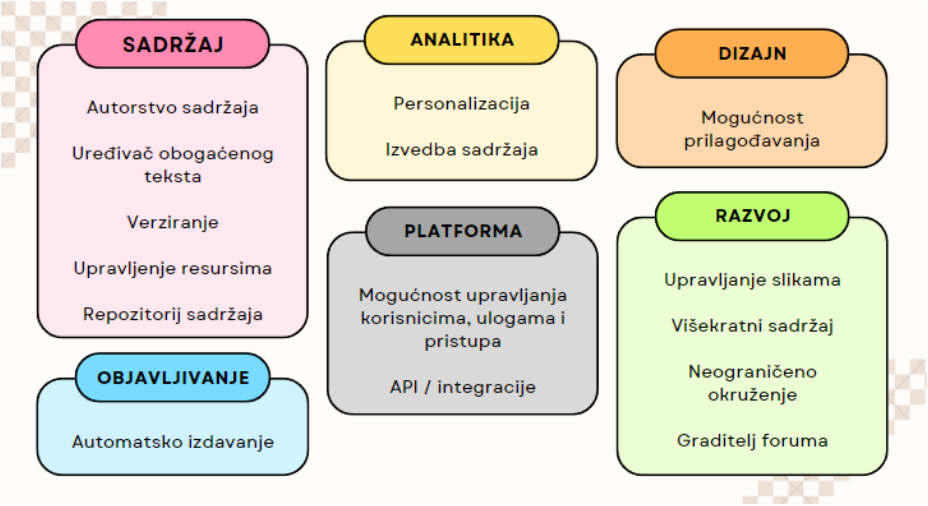
Sljedeći primjeri su prethodno spomenuta komercijalna rješenja / rješenja zatvorenog koda za CMS sustave bez grafičkog sučelja, kojima je detaljno objašnjena svrha, njihove glavne funkcionalnosti, kao i cjenik svakog CMS sustava. Naknadno će se komercijalna rješenja usporediti prema određenim kriterijima zadovoljstva korisnika.

3.1.1.1. Sanity

Tvrtka koja proizvodi programski alat Sanity osnovana je 2017. godine, a podržani jezik u samom alatu je engleski. Glavno sjedište sustava je u gradu San Francisco u Kaliforniji. Sanity je platforma za upravljanje strukturiranim sadržajem, gdje se svi podaci dohvaćaju pomoću API-ja. Sanity pruža mogućnost upravljanja tekstom, slikama i drugim medijskim zapisima. Pruža i mnoštvo API-ja, biblioteka i alata koji korisnicima pružaju mogućnost upravljanja i objavljivanja vlastitih sadržaja u oblacima. Prema samoj dokumentaciji programskog alata Sanity [12], on se temelji na tri važna koncepta:

- pohrana podataka za strukturirani sadržaj izvodi se u realnom vremenu i pruža mnoštvo API-ja za dohvaćanje sadržaja, upravljanje korisnicima i još mnogo toga;
- Sanity je sustav za upravljanje sadržajem otvorenog koda koji se nalazi na jednoj stranici (eng. *Single Page Application* – SPA). Sustav je izgrađen pomoću programskog okvira React;
- Sanity pruža mnoštvo paketa za razvoj softvera (eng. *Software Development Kit* – SDK), biblioteka i alata koji omogućavaju izvršavanje upita nad sadržajem, kako bi se sadržaju moglo pristupiti na što lakši način.

Pohrana sadržaja se vrši u stvarnom vremenu, pri čemu se koristi pozadina bez sheme. To znači da kod pohrane sadržaja ne postoje predefinirana pravila za pohranu sadržaja kao kod tradicionalnih baza podataka, nego se koristi JSON format dokumenata. Dohvaćanje sadržaja postiže se pomoću API-ja, a za upite se koristi upitni jezik (eng. *query language*) GROQ. GROQ je kreiran od strane Sanity-a te je snažan i inovativan programski jezik. GROQ je jednostavan za naučiti, a korisnici pomoću njega mogu detaljno specificirati i dodatno filtrirati koje podatke je potrebno dohvatiti i prikazati. Dodatna funkcionalnost Sanityjevog API-ja je slanje HTTP PUT i POST zahtjeva, pomoću kojih se specificiraju izmjene, odnosno kreira novi sadržaj. Ovi podaci se pohranjuju u JSON datotekama pomoću kojih se ažurira stanje u bazi podataka (kreiranje, modifikacija, brisanje podataka, itd.) koji Sanity naziva „jezerom sadržaja“ (eng. *Content Lake*). Glavne funkcionalnosti alata Sanity ilustrirane su na slici 5.



Slika 5. Prikaz glavnih funkcionalnosti alata Sanitya [13]

Sanity je alat zatvorenog koda koji pruža zauvijek besplatnu verziju CMS-a. Savršen je za osnivanje i razvoj poslovanja kojem je potreban sustav za upravljanje sadržajem bez grafičkog sučelja. Komercijalna verzija poslovanja za manja poduzeća nudi timsku opciju s relativno niskim troškovima, verzija za srednje poduzeća nudi poslovni paket gdje su troškovi već zahtjevniji, dok su za velika poduzeća troškovi po dogovoru, ovisno o zahtjevima CMS sustava. Sljedeća tablica 1. prikazuje cjenik i ključne funkcionalnosti s kojima dolazi svaki poslovni paket:

Tablica 1. Prikaz cjenika i funkcija svakog paketa Sanity Headless CMS-a

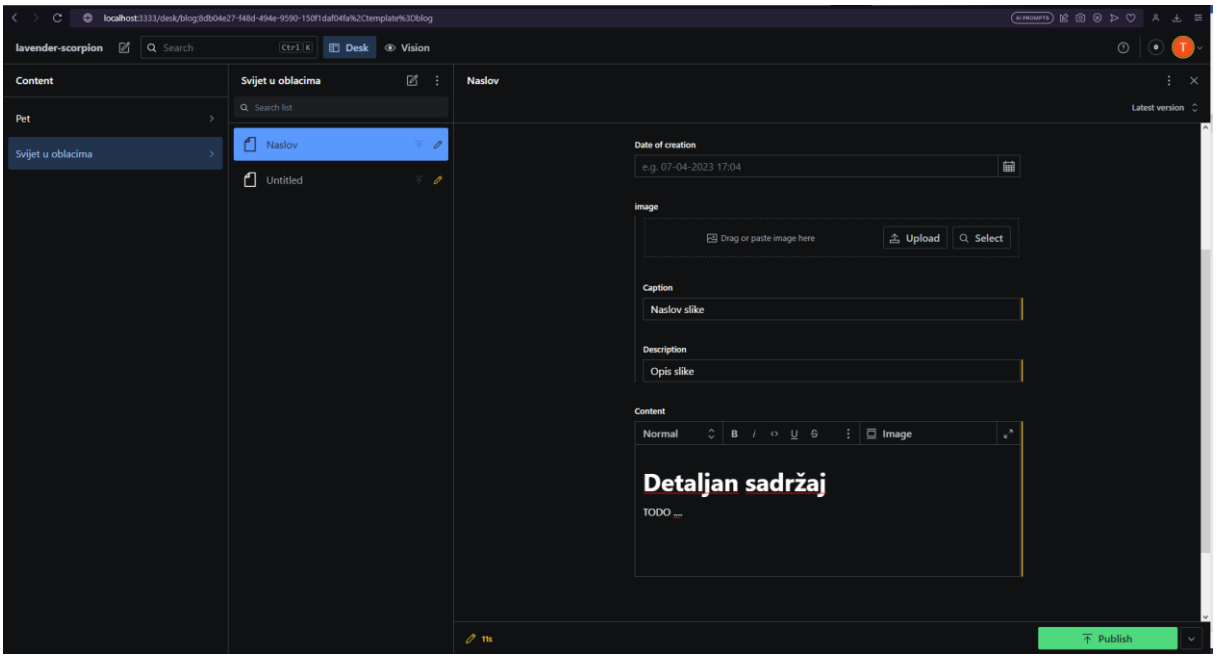
Besplatna verzija	**Timska verzija**	**Poslovna verzija**	**Enterprise**
$0	Od $99 mjesečno po projektu	Od $949 mjesečno po projektu	Cijena po dogovoru
Okruženje za uređivanje	10 ne-administratorskih korisnika	20 ne-administratorskih korisnika	Prilagođen broj korisnika i uloga
Baza podataka sadržaja u stvarnom vremenu	30 dana povijesnog zadržavanja promjena sadržaja	Napredno upravljanje skupom podataka	Prilagođena kvota korištenja
Sjajno iskustvo programera	Dodatna kvota	SAML SSO	Kontakt za uspjeh kupaca
Besplatni admin korsinici	Plaćanje kako se širi poduzeće za veće korištenje	90 dana povijesnog zadržavanja	365 dana povijesnog zadržavanja
3 ne-admin korisnika		Kvota velike količine	SLA i podrška 24/7

(Izvor: Paters [13])

Sanity programerima pruža potpunu slobodu kod kreiranja shema za svaki sadržaj koji se dodaje u sustav. Instalacija je vrlo jednostavna te se vrši preko naredbenog retka (eng. *command prompt* – CMD), pri čemu se pokreće lokalni Sanity server – naravno, nakon uspješne prijave u korisnički račun. Pomoću alata koji je lokalno podignut, moguće je kreirati, uređivati i objavljivati željeni sadržaj. Prije kreiranja samoga sadržaja razvojni inženjeri moraju kreirati .js datoteke sa shemama sadržaja te iste pohraniti u direktorij sheme projekta. Sljedeći dio koda prikazuje način kreiranja sheme modela sadržaja za praktični dio rada:

```
export default {
 name: 'blog',
 type: 'document',
 title: 'Svijet u oblacima',
 fields: [
   {name: 'title', type: 'string', title: 'Title'},
   {name: 'subTitle', type: 'string', title: 'Sub-Title'},
   {name: 'author', type: 'string', title: 'Author'},
   {name: 'dateOfCreation', type: 'datetime',
    title: 'Date of creation',
          options: { dateFormat: 'DD-mm-YYYY', timeFormat: 'HH:mm',
            timeStep: 15, calendarTodayLabel: 'Today'}},
   {title: 'image', name: 'Image', type: 'image',
      options: {hotspot: true},
     fields: [
       {name: 'caption', type: 'string', title: 'Caption',},
       {name: 'description', type: 'string', title: 'Description',}]},
   {name: 'content', type: 'array', title: 'Content', of: [
       {type: 'block'},
       {type: 'image'}]}]}
```

Nakon kreiranja željene shema sadržaja i importa u Sanity projekt, moguće je uređivati i popunjavati sadržaj pomoću lokalnog Sanity servera koji se pokreće na predefiniranom portu http://localhost:3333/desk. Slika 6. u nastavku prikazuje mogućnost kreiranja novog sadržaja pomoću navedenog CMS alata:



Slika 6. Prikaz dodavanja novog sadržaja pomoću alata Sanity [autorski rad]

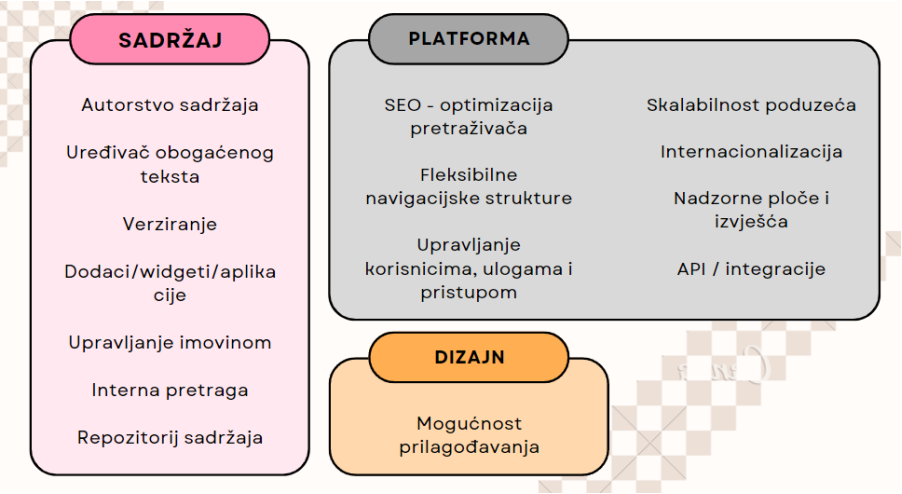
Jedna od pozitivnih strana sustava Sanity je velika fleksibilnost sustava po pitanju kreiranja modela/sheme sadržaja – korisnicima se prepušta potpuna kontrola nad modeliranjem sadržaja. Postoje predefinirana korisnička polja, kao što su dodavanje teksta, slika, datuma i drugih općenitih podataka koji se mogu pohranjivati u CMS sustavu, no za kompleksnije funkcionalnosti, kao što je višestruki odabir pojedinih predefiniranih vrijednosti, mogu se kreirati vlastite komponente (eng. *custom components*), što zahtjeva dodatno učenje i ulaganje vremena. Velika prednost sustava Sanity je leži u visokim performansama, velikoj brzini i izvrsnoj strukturiranosti. Dohvaćanje, kreiranje ili modificiranje sadržaja izvršava se gotovo instantno. Dodatna prednost alata Sanity je velika proširivost sustava korištenjem raznih dodataka (eng. *plugins*).

S druge strane, negativne strane sustava Sanity bile bi manjak adekvatne dokumentacije, kao i skromnost postojeće dokumentacije, što prisiljava korisnika na oslanjanje na forumske objave. Druga stvar koja upada u oči i koju su korisnici spomenuli bila bi prevelika kompleksnost. Primjerice, ako je potrebna neka jako specifična stvar za shemu sadržaja, potrebno je kreirati vlastitu komponentu za koju treba uložiti mnogo vremena i novaca. Dodatno, osim što je potrebno implementirati logiku vlastite komponente, potrebno je kreirati i cijeli dizajn komponente. Za početnike bi ovaj sustav mogao biti vrlo kompliciran.

3.1.1.2. Contentful

Tvrtka koja je programski alat Contentful osnovana je 2013. godine, a podržani jezik u samom alatu je engleski. Glavno sjedište sustava je u gradu Berlinu u Njemačkoj. Za razliku

od prethodno navedenog konkurentskog alata Sanity, Contentful ne pruža potpunu slobodu kod kreiranja sustava za upravljanje sadržajem, no pruža mnoštvo gotovih komponenti koje se mogu međusobno povezivati u zajednički model sadržaja (eng. *Content Model*). *Content Model* je zapravo prazni predložak koji se naknadno može neograničen broj puta ispunjavati te mu se može dodavati zaseban sadržaj. Glavna funkcionalnost koju Contentful Team višestruko spominje i navodi na svojim stranicama je da se sadržaj kreira samo jednom i potom objavi, a objavljeni sadržaj se potom s lakoćom može koristiti neograničen broj puta na bilo kojem IoT uređaju koji je povezan na Internet. Dodatno navode da u slučaju da se željeni sadržaj može razvrstati po tipovima podataka, *Content Model* se tada za željeni sadržaj može lagano kreirati te se sav sadržaj onda može pohraniti u Contentfulovoj bazi podataka. Contentful pruža mogućnost višestrukog autorstva nad jednim sadržajem, pri čemu autori sadržaja mogu uređivati i mijenjati jedan zajednički sadržaj. Nudi i mogućnost verzioniranja sadržaja, gdje se željene verzije sadržaja mogu objavljivati i puštati u pogon, itd. Funkcionalnosti alata Contentful-a mogu se pronaći na sljedećoj slici 7.



Slika 7. Prikaz glavnih funkcionalnosti alata Contentfula [14]

Contentful je također rješenje zatvorenog koda, ali pruža i zauvijek besplatnu verziju poslovnog paketa koja korisnicima ne nudi puni potencijal Contentfula. Drugi poslovni paket je „Osnovni“ paket koji košta 300 američkih dolara mjesečno, dok se Premium poslovni paket naplaćuje po dogovoru. Sljedeća tablica 2. prikazuje poslovne pakete i njihove mogućnosti alata Contentful.

Tablica 2. Prikaz cjenika i funkcija svakog poslovnog paketa Contentfula

Besplatna verzija	**Osnovna verzija**	**Premium verzija**
$0	$300 mjesečno, besplatna proba	Cijena po dogovoru
5 korisnika sustava	20 korisnika sustava	Tisuće korisnika sustava
4 standardne uloge	4 standardne uloge	Prilagođen broj uloga
2 korištena jezika	4 korištena jezika	Prilagođen broj jezika
Podrška zajednice	Tehnička podrška	Pristup uspjehu kupaca i profesionalnim uslugama
Task App & Compose App	Task App, Compose App & Launch App	Značajke upravljanja (npr. SSO, tijek rada aplikacije, orkestracija)
1 poslovni prostor	1 poslovni prostor	2 poslovna prostora
Nema mogućnost nadogradnje poslovnog prostora	Mogućnost nadogradnje na srednji poslovni prostor za $350 mjesečno	Mogućnost nadogradnje na srednje, velike i veće Premium prostore – dodatna naknadna

(Izvor: Sharma [14])

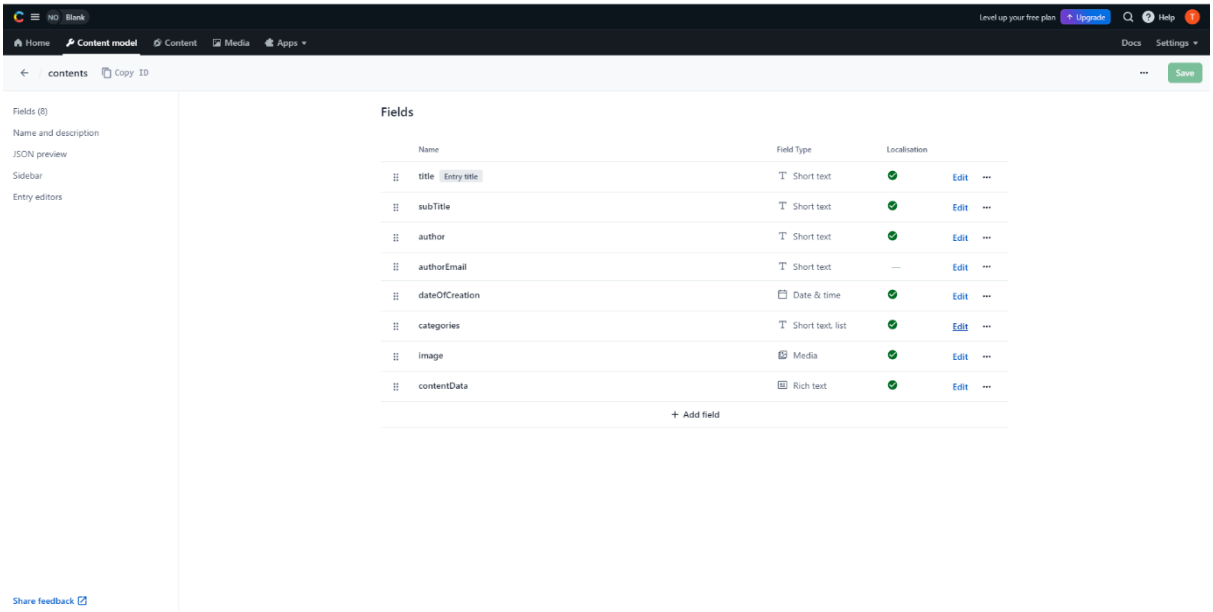
Contentful programerima pruža mogućnost kreiranja modela sadržaja pomoću predefiniranih atributa koji se mogu dodatno specificirati i definirati. Instalacija na računalo nije potrebna, već je dovoljno kreirati korisnički račun na njihovoj web stranici jer se cijeli alat nalazi u web pregledniku. Prvi korak kod kreiranja CMS sustava je kreiranje novog *Content Modela*, pri čemu korisnici mogu odabrati jedno od ponuđenih polja kao prvo polje modela:

- bogati tekst – namijenjen za stranicu o detaljima sadržaja pošto pruža mogućnost uređivanja teksta, dodavanja slika, videa, audio zapisa, tablica, itd.;
- običan tekst – namijenjen za naslove, podnaslove, paragrafe sadržaja i slično;
- brojčana vrijednost – namijenjena za brojčane vrijednosti kao što su ID, broj narudžbe i slično;
- datum i vrijeme;
- lokacija – koordinatne vrijednosti;
- medijski zapisi;
- Booleove (eng. *Boolean*) vrijednosti;
- JSON objekti – za pohranu podataka u JSON formatu;
- referenca – namijenjena za povezivanje jedne vrste *Content Modela* s drugim vrstama.

Svako polje modela posjeduje dodatne mogućnosti specificiranja i definiranja. Primjerice, svakom se polju može postaviti lokalizacija s kojom se omogućava da se sadržaj kreira na više različitih jezika. Dodatne mogućnosti bile bi postavljanje zadanih vrijednosti, definiranje obaveznih polja, mogućnost ograničavanja s minimalnim i maksimalnim

vrijednostima, validacija vrijednosti, mogućnost postavljanja višestrukih predefiniranih vrijednosti kao što je potvrdni okvir (eng. *checkbox*), itd.

Kad se kreira željeni oblik *Content Modela,* otvara se sljedeću stranica na kojoj se nalaze svi kreirani sadržaji. Na njoj se može dodavati sadržaj, postavljati datum objavljivanja sadržaja, pregledavati, uređivati i filtrirati objavljene sadržaje, kao i postavljati lokalizaciju. Administratori mogu dodavati svoje korisnike/radnike u sustav te im pritom dodijeliti odgovarajuće uloge (admin, autor, editor i prevoditelj). Sljedeća slika 8. prikazuje metodu kreiranja modela sadržaja u navedenom alatu.



Slika 8. Prikaz kreiranja modela sadržaja u Contentfulu [autorski rad]

Pozitivne strane navedenog alata bile bi lakoća kreiranja modela sadržaja i sama fleksibilnost alata. Contentful nije toliko fleksibilan Sanity, ali je zato mnogo jednostavniji i prilagođeniji korisnicima (eng. *user-friendly*). Primjerice, funkcionalnost koja nije bila realizirana kod alata Sanity (radi kompleksnosti alata) je da se urednicima sadržaja nije pružala mogućnost višestrukog odabira predefiniranih vrijednosti koje se potom mogu jednostavno odabrati ili prikvačiti. Ispostavlja se da je ova funkcionalnost vrlo jednostavna i logična u alatu Contentful. Sama dokumentacija o cijelom alatu Contentful je dobro definirana i jasna. Dohvat cijelih sadržaja je također jednostavno realiziran – sve što je potrebno napraviti je definirati podatke o klijentu za željeni radni prostor Contentfulovog projekta te specificirati što je sve potrebno za dohvatiti (primjerice: *content_type, locale, order, skip, limit*, određeno polje, itd.). Dobivene podatke je na kraju potrebno mapirati u željeni oblik – bio to objekt, niz ili običan tekstualni podatak. Contentful nudi mogućnost dohvaćanje sadržaja pomoću standardnog

RESTful API-ja ili modernijeg načina kao što je GraphQL. Contentful pruža snažan API i SDK, koji dolaze s opsežnom dokumentacijom. Dodatno, Contentful pruža mogućnost instalacije aplikacija trećih strana na poslovne prostore projekta, što je također vrlo jednostavno. Pojedini korisnici alata govore kako je cijeli alat dobro skalabilan za buduće modifikacije i ažuriranje, dok s druge strane pojedini korisnici govore kako je alat dobar, ali nije skalabilan. Alat je skalabilan, ali samo za manja i srednja poduzeća, dok kod velikih poduzeća koja zahtijevaju ogromne količine podataka skalabilnost alata pada.

Negativna strana koja se javlja kod alata Contentful bila bi uvođenja više od jednog jezika, zato što u njemu ne postoji nikakav oblik automatizacije ili direktnog prevođenja iz jednog jezika u drugi, već je isti sadržaj potrebno ponovno ispunjavati za svaki željeni jezik, što može dovesti do netočnih podataka ili nepreglednosti kod samog uređivanja sadržaja. Alat posjeduje prikaz editiranja sadržaja na temelju jednog jezika ili više jezika odjedanput, za što je potrebna prilagodba. Ostale negative strani koje se uočavaju bile bi nemogućnost grupiranja više različitih modela sadržaja u jedan (potrebno je kreirati vlastite funkcije koje bi „oponašale“ mogućnost grupiranja modela sadržaja), poteškoće kod nadogradnje Contentfulovih projekata iz jednog poslovnog paketa u drugi te dohvaćanje podataka pomoću GraphQL upita (upiti moraju biti dobro strukturirani i bez pogrešaka, pogotovo kod dubokih ugniježđenih podataka – inače postoji velika vjerojatnost za dobivanjem pogreške: '11000 GraphGL complexity error').

3.1.1.3. Sanity vs Contentful

Sljedeće usporedbe temelje se na prikupljenoj statistici koja se temelji na raznim anketama provedenim nad korisnicima navedenih alata. Statistiku je objavila web stranica G2 [15] koja je poznata po pronalaženju željenih softvera i usluga, a temelji se na više od 2.2 milijuna stvarnih recenzija i ocjena.

Prema ukupnoj ocjeni iz statistike, alat Sanity dobio je ocjenu 4.7 izračunatu na temelju 669 recenzija, dok je Contentful dobio ocjenu 4.2 od izračunatu na temelju 273 recenzije. Prema generalnoj usporedbi, alat Sanity je za nijansu bolji od alata Contentful, što se može vidjeti u sljedećoj tablici 3.

Generalno gledano, oba su alata su savršena za začetak poslovanje u kojem je potreban sustav za upravljanjem sadržajem bez grafičkog sučelja. Alat Sanity je malo bolji ako za manja poslovanja, dok je alat Contentful bolji za srednja i veća poslovanja, ali samo do određenih granica. Alat Sanity programerima pruža veću slobodu kod kreiranja modela sadržaja, ali s time dolazi i dodatna kompleksnost i potreba za dodatnim znanjem.

Tablica 3. Usporedba alata Sanity i Contentful

Generalna usporedba	Contentful	Sanity
Potrebne specifikacije alata	8.5	9.1
Lakoća korištenja	8.4	8.7
Jednostavnost postavljanja sustava	8.2	8.8
Jednostavnost administracije sustava	8.4	9.1
Kvaliteta podrške	8.4	9
Je li proizvod bio dobar partner u poslovanju?	8.5	9.3
Smjer proizvoda (% pozitivno)	7.4	9.2
Sadržaj		
Autorstvo sadržaja	8.3	9.3
Uređivač obogaćenog teksta	7.8	9
Verziranje	7.9	9.1
Dodaci/widgeti/aplikacije	7.7	8.3
Raspored sadržaja	8.4	8.4
Upravljanje imovinom	7.8	8.6
Interna pretraga	7.7	8.6
Repozitorij sadržaja	8.2	9.1
Funkcionalnost		
Razvojni alati	8.9	9.4
Automatizacija	9	8.8
Inscenacija web stranice	8.5	9.3
Objavljivanje		
Automatizirano izdavanje	8	8.9
Višejezično	8.8	8.1
AR/VR mogućnost	8	7.4
Platforma		
Zajednica korisnika	7.7	9
Optimizacija pretraživača (eng. Search Engine Optimization, SEO)	7.9	8.9
Fleksibilne navigacijske strukture	8.1	9.1
Upravljanje korisnicima, ulogama i pristupom	7.9	8.9
Skalabilnost poduzeća	8.3	8.8
Internacionalizacija	8.4	8.5
Nadzorne ploče i izvješća	8.2	8.5
API / integracije	8.7	9.2
Razvoj		
Upravljanje slikom	8.1	8.7
Višekratni sadržaj	8.8	9.2
Neograničena okruženja	7.7	8.7

(Izvor: G2 [15])

Alat Contentful je po tom pitanju malo manje fleksibilan od konkurentskog alata Sanity, ali programerima i dalje daje veliku slobodu kod kreiranja modela sadržaja. Contentful nije toliko kompleksan kao Sanity i temelji se na vrlo jednostavnim i logičnim principima kreiranja

modela sadržaja. Alat Contentful bolji je za *junior* programere, početnike i urednike sadržaja zato što je jednostavniji i sadrži jako detaljnu dokumentaciju za razvoj CMS sustava, dok je alat Sanity bolji za profesionalnu upotrebu i specifičnije CMS sustave jer je izrazito fleksibilan CMS sustav bez grafičkog sučelja.

3.1.2. Rješenja otvorenog koda za CMS bez grafičkog sustava

Sljedeći primjeri su rješenja otvorenoga koda za sustave za upravljanje sadržajem bez grafičkog sustava. Kao primjeri se navode i objašnjavaju alati Strapi i Directus, a potom se i međusobno uspoređuju.

3.1.2.1. Strapi

Tvrtka koja proizvodi programski alat Strapi osnovana je 2016. godine, a podržani jezici u samome alatu su engleski i francuski. Glavno sjedište sustava je u gradu Parizu, glavnom gradu Francuske. Programski alat Strapi jedan je od vodećih rješenja otvorenog koda za headless CMS sustave, a temelji se na čistoj JavaScript tehnologiji. Strapi nudi prilagodljivi API „iz kutije“ i razvojnim inženjerima pruža slobodu korištenja raznovrsnih i željenih programskih alata. Brojna poduzeća kao što su: IBM, NASA, Walmart, eBay, Rakuten ili Toyota koriste Strapi kako bi upravljali svojim sadržajem te isti dostavljali na raznovrsne IoT uređaje.

Strapi je vrlo fleksibilan alat koji razvojnim inženjerima daje slobodu kod kreiranja modela sadržaja, a ujedno daje i mogućnost modificiranja administratorskog panela i API-ja. Razvojni inženjeri mogu mijenjati i prilagođavati cijeli dizajn CMS sustava kako bi zadovoljili svoje potrebe i želje te kako bi on odgovarao svakom realnom slučaju korištenja. Strapi ima ugrađeno korisničko sučelje posebno za razvojne inženjere sustava te posebno za urednike sustava. Urednici sustava su zaduženi za sam sadržaj koji će biti dostupan nakon objavljivanja. Dohvaćanje sadržaja obavlja se preko običnog RESTful API-ja ili pomoću GraphQL-a.

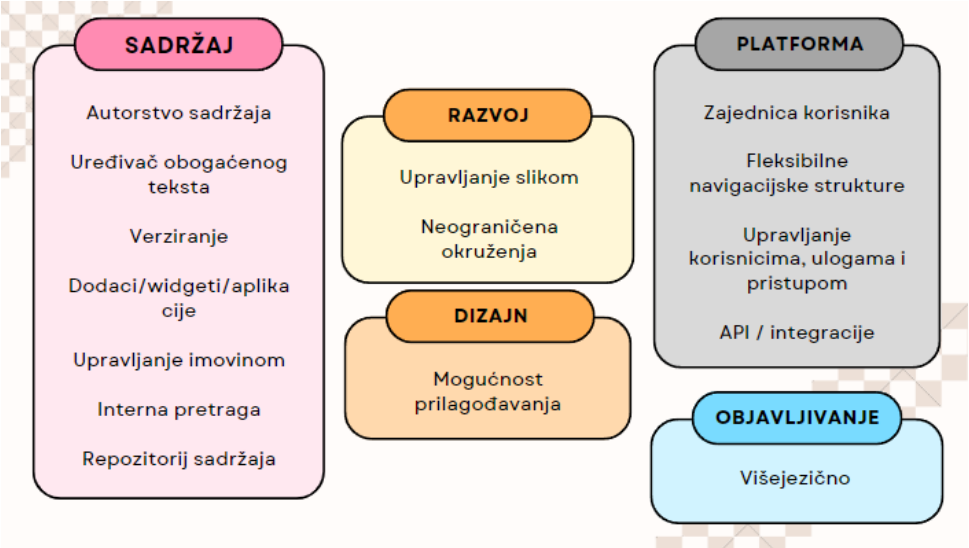
Strapi je rješenje otvorenoga koda, čija je cijela dokumentacija, temeljni kod i ostali povezani alati dostupna na njihovoj GitHub stranici. Pošto je Strapi otvorenoga koda njihovi developeri su naveli u dokumentaciji da cijene svaki doprinos, bilo to u vezi zahtjeva za značajke, izvješća o pogrešci ili ostalim stvarima koji su povezani s alatom. Strapi nudi četiri poslovna paketa za svoje klijente: besplatni paket / *„Community“*, *„Enterprise Edition“*, *„Cloud Pro*“ i *„Cloud Team“*. U sljedećoj tablici 4. nalaze se svaki poslovni paket i njihove specifikacije.

Tablica 4. Prikaz poslovnih paketa alata Strapia

<table>
<tr><th>Community</th><th>Enterprise Edition</th><th>Cloud Pro</th><th>Cloud Team</th></tr>
<tr><td>Besplatno,
uz jednokratnu kupnju</td><td>Cijene po dogovoru za jednog korisnika godišnje</td><td>$99 za 1 projekt mjesečno</td><td>$499 dolara za projekt mjesečno</td></tr>
<tr><td>Rješenje otvorenoga koda pod licencom MIT-a</td><td>Dnevnici revizije</td><td>Sve značajke Community paketa, plus:</td><td>Sve značajke paketa Cloud Pro, plus:</td></tr>
<tr><td>REST & GraphQL API</td><td>Jedinstvena prijava</td><td>Ugrađeni pružatelj usluge e-pošte</td><td>20 sjedećih mjesta</td></tr>
<tr><td>Do 3 zadane uloge</td><td>Tehnička podrška sa SLA-ovima</td><td>Globalni CDN</td><td>1.000.000 CMS unosa</td></tr>
<tr><td>Vrlo fleksibilna struktura sadržaja</td><td>Upravitelj uspjeha kupaca</td><td>Dnevnici u stvarnom vremenu</td><td rowspan="2">Veća ograničenja pohrane i propusnosti</td></tr>
<tr><td>Moćno upravljanje sadržajem</td><td>Tehnički ukrcaj</td><td>Obavijesti putem e-pošte/u aplikaciji</td></tr>
<tr><td>Beskrajne mogućnosti prilagodbe</td><td>SSO, podrška sa SLA-ovima</td><td>10 sjedećih mjesta</td><td rowspan="4">Dnevnici revizije (7 dana)</td></tr>
<tr><td>Neograničeno korištenje (unosi, API pozivi, sredstva itd.)</td><td>Tehničko uključivanje</td><td rowspan="3">100.000 CMS unosa</td></tr>
<tr><td>Neograničene lokacije</td><td rowspan="2">Ostale napredne značajke</td></tr>
<tr><td>Podrška zajednice</td></tr>
</table>

(Izvor: Almeida [16]

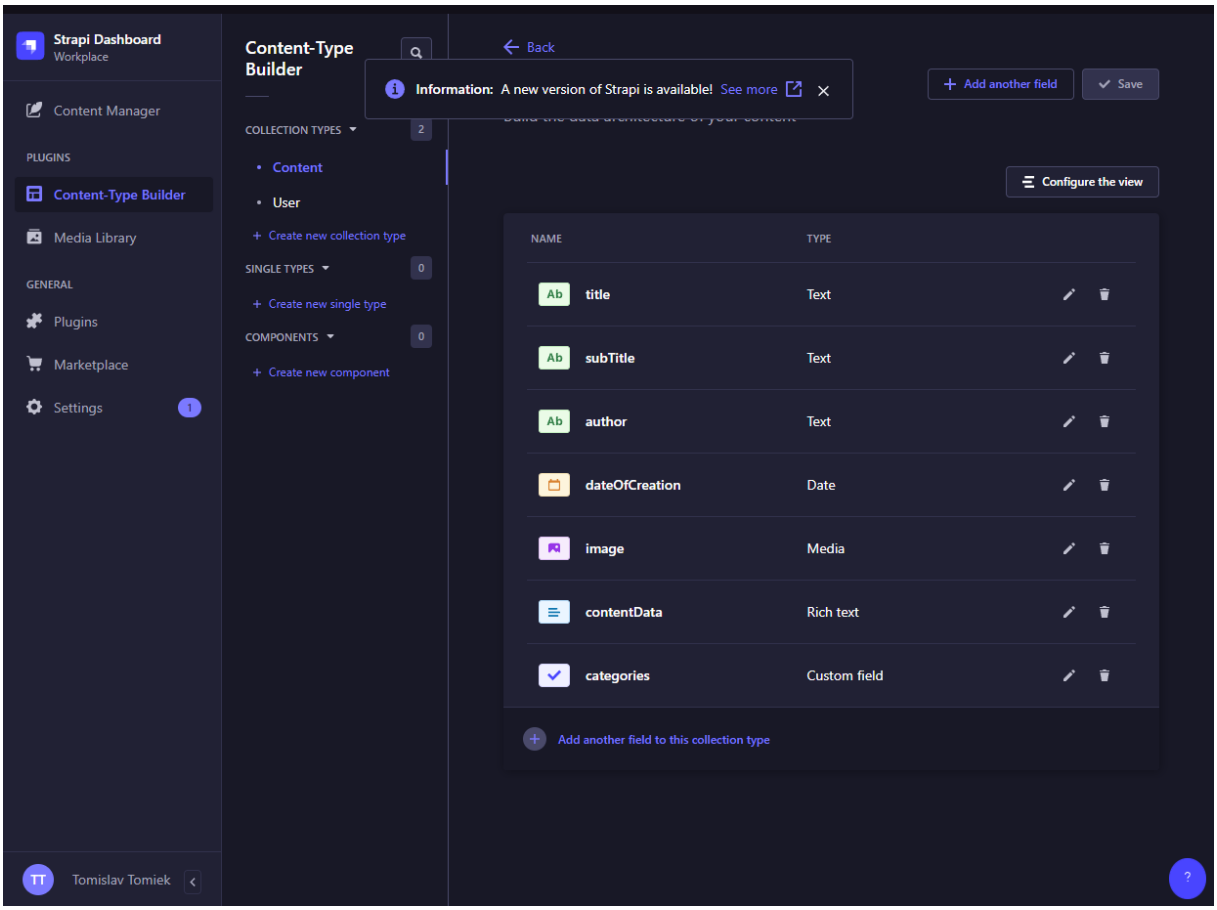
Glavne funkcionalnosti programskog alata Strapi prikazane su na slici 9.



Slika 9. Prikaz glavnih funkcionalnosti alata Strapia [16]

Instalacija alata je vrlo jednostavna, kao i kod alata Sanity. U željenom je direktoriju potrebno pokrenuti CMD, zatim pomoću Node.js-a instalirati alat izvršavanjem sljedeće linije

koda: '`npx create-strapi-app@latest my-project –quickstart`'. Nakon završetka instalacije, otvara se lokalno podignuta web stranica alata Strapi u web pregledniku, na kojoj razvojni inženjeri mogu kreirati modele sadržaja na način da novokreiranom kolekcijskom tipu (eng. *Collection Type*) dodaju željena polja i specificiraju njihove tipove. Izbor polja za model sadržaja u alatu Strapi malo je skromniji od prethodno navedenog konkurenta Contentful, ali je vrlo logičan i jednostavan za shvatiti. Svako polje modela sadržaja moguće je dodatno specificirati i ograničiti po želji. U slučaju da su potrebna dodatna polja, korisnici na jednostavan način mogu instalirati vlastito polje iz svoje trgovine, koja može biti komercijalna ili besplatna za preuzimanje. Drugi način za instalaciju dodatnih polja je stvaranje vlastitih komponenti. Primjerice, prilikom kreiranja prethodno spomenutog primjera za odabir kategorije članka, bilo je potrebno instalirati dodatno polje za višestruki odabir predefiniranih varijabli. Prikaz za kreiranje modela sadržaja pomoću navedenog alata može se vidjeti na slici 10. Programerima se pruža mogućnost mijenjanja dizajna i korisničkog sučelja za urednike CMS sustava koji popunjavaju model sadržaja stvarnim podacima – primjerice, moguće je promijeniti redoslijed polja za ispunu podataka i njihovu veličinu, blokirati ispunu polja, itd.



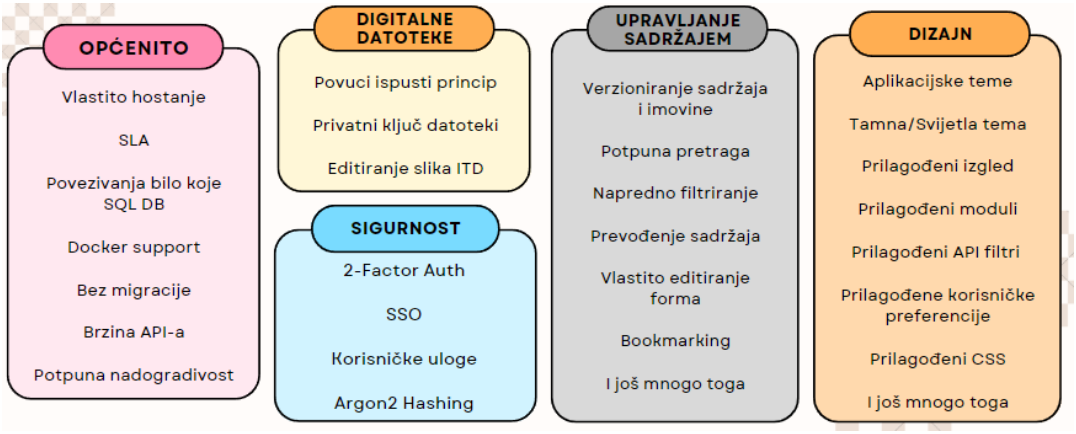
Slika 10. Prikaz kreiranja modela sadržaja alatom Strapia [autorski rad]

Pozitivne strani alata Strapi uglavnom bi odgovarale pozitivnim stranama prethodno spomenutih komercijalnih alata, međutim, dodatna prednost alata Strapi je potpuna kontrola nad cijelim dizajnom CMS sustava, kreiranjem modela sadržaja, podataka i modula. Omogućena je i dodatna fleksibilnost ugrađivanjem proširenja u slučaju da je korisnicima potrebna dodatna funkcionalnost sustava. Dodatna prednost alata leži u tome što je rješenje otvorenoga koda te svima dopušta pristup čitanja cijelog programskog koda, kao i prilagodbu po potrebi. Dodatno, alat je dizajniran na način da bude prilagođen korisniku, dok istovremeno pruža zadivljujuće iskustvo povezivanja svega s prilagođenim CMS sučeljem.

Negativna strana alata Strapi bio bi skroman i limitiran set opcija za formatiranje obogaćenog teksta, koji može služiti za prikaz sadržaja na stranici o detaljima pojedinog članka, proizvoda ili slično. Druga mana, koja se češće javlja kod korisnika alata bio bi zbunjujući dizajn specifikacije i dodjele uloga kod većih timova. Nedostatak pojedinih jezika i prevođenje iz jednog jezika u drugi stvara problem kod pojedinih korisnika, a lokalizacija je u slučaju medijskih zapisa otežana, ponekad i nemoguća. Jedan od većih nedostataka s kojim se Strapi susreće je nekonzistentno objavljivanje i ažuriranje dokumentacije te povremene greške u dokumentaciji, što dovodi do nepotrebnog gubljenja vremena za traženje grešaka ili novih funkcionalnosti kod najnovijih verzija alata.

3.1.2.2. Directus

Tvrtka koja proizvodi programski alat Directus osnovana je 2011. godine i sami alat posjeduje raznovrsne podržane jezike: od engleskog, njemačkog, poljskog pa sve do kineskog. Sveukupno podržava više od pedeset jezika. Glavno sjedište sustava je u gradskoj četvrti Brooklyn u New Yorku. Programski alat Directus je također alat otvorenog koda koji se bazira na tehnologiji TypeScript koja je izgrađena na način da se cijeli programski kod s lakoćom može modificirati i mijenjati po želji. Podaci i sadržaji CMS sustava Directus mogu se pohranjivati u novu ili postojeću SQL bazu podataka, gdje se za dohvaćanje sadržaja može koristiti REST ili GraphQL API, ovisno o preferencijama razvojnih inženjera. Kod samog dohvaćanja sadržaja iz SQL baze podataka, rezultati se direktno i dinamično transformiraju u korisne REST ili GraphQL krajnje točke pomoću kojih se zatim mogu prikazivati na željenim IoT uređajima. Directus pruža bogati skup alata koji su zaduženi za upravljanje sadržajem, vizualizaciju i povezivanje podataka. Front-end alat sadržava fleksibilno upravljanje sadržajem iz baze podataka koje se temelji na „povuci i ispusti“ (eng. *drag and drop*) principu za dodavanje novih polja i vrijednosti. Osim vlastito kreiranih polja nudi i mogućnost nadzornih ploča, raznih grafikona, upravljanje digitalnom imovinom, end-to-end internacionalizaciju i još mnogo toga. Važnije funkcionalnosti koje alat Directus nudi su istaknute na sljedećoj slici 11:



Slika 11. Prikaz glavnih funkcionalnosti alata Directusa [17]

Kao što je već spomenuto, Directus je rješenje otvorenoga koda, a to znači da je alat besplatan, jer se cijeli programski kod nalazi na vlastitoj javno dostupnoj GitHub stranici. Međutim, Directus svojim korisnicima nudi dvije glavne opcije vođenja CMS sustava. Prva opcija u potpunosti je temeljena na oblaku i nudi dva poslovna paketa: „Standard“ i „Enterprise“ oblak. Druga je opcija samostalno (*on-premise*) hostanje servera kod koje se također nude dva poslovna paketa: „Community“ i „Enterprise“ *Self-Hosted*. Sljedeća tablica 5. prikazuje sve poslovne pakete i njihove osobnosti s kojima dolaze:

Tablica 5. Prikaz svih poslovnih paketa alata Directus-a.

Standard Cloud	**Enterprise Cloud**	**Community Self-Hosted**	**Enterprise Self-Hosted**
$99 mjesečno	Cijena po dogovoru	Potpuno besplatno	Fleksibilni cjenik
Neograničeni broj korisnika i uloga	Sve iz paketa *Standard Cloud*, plus:	Neograničen broj korisnika i uloga	Neograničen broj korisnika i uloga
Neograničene zbirke i polja	Namjenska infrastruktura	Neograničeno korištenje	Neograničeno korištenje
Multi-tenant infrastruktura	Udaljen pristup bazi podataka	Besplatno i otvorenog koda	Besplatna neproizvodna upotreba
Globalni CDN	Premium podrška i SLA	Nema ograničenja značajki	
Bijela oznaka	SSO, uključujući LDAP i SAML		Cijene temeljene na propusnosti API-ja
Sigurnosne kopije i nadzor 24/7	Prilagođena proširenja		
Mjesečni ili godišnji uvjeti	Dostupne dodatne regije		

(Izvor: Monospace Inc [17])

Instalacija alata vrši se malo drugačije nego kod svih do sada spomenutih alata. Instalacija započinje instalacijom alata Docker koji je zaslužan za podizanje lokalnih servera i kontejnera. Nakon instalacije Dockera, u željenom direktoriju se kreira datoteka `docker-`

`compose.yml` u kojoj se specificiraju tajni ključ i tajnost pomoću kojih će se podaci kriptirati. Potom se dodatno specificira administrativni korisnički račun, a nakon toga i port na kojem će se podići server te se po želji definira baza podataka koja će se koristiti kod CMS sustava (zadana baza podataka je SQLite). Nakon konfiguracije *yml* datoteke, u istom se direktoriju pokreće u kojem se pomoću naredbe '`docker compose up`' pokreće lokalni server – CMS sustav alata Directus. Korisnici se u web pregledniku prijavljuju u definirani korisnički račun te potom imaju pristup punoj funkcionalnosti sustava Directus. Među do sad navedenim konkurentima, alat Directus pruža najveći izbor predefiniranih funkcionalnosti – od definiranja modela sadržaja, modificiranja dizajna modela sadržaja, korisničkih postavki i uloga pa sve do zadnjih detalja oko specifikacije polja za unos sadržaja. Korisničko sučelje je vrlo elegantno i moderno dizajnirano, a korisnici mogu modificirati dizajn svog korisničkog sučelja (mijenjati boju, dodavati nove korisnike u projekt s odgovarajućim i prilagođenim korisničkim ulogama i još mnogo toga).

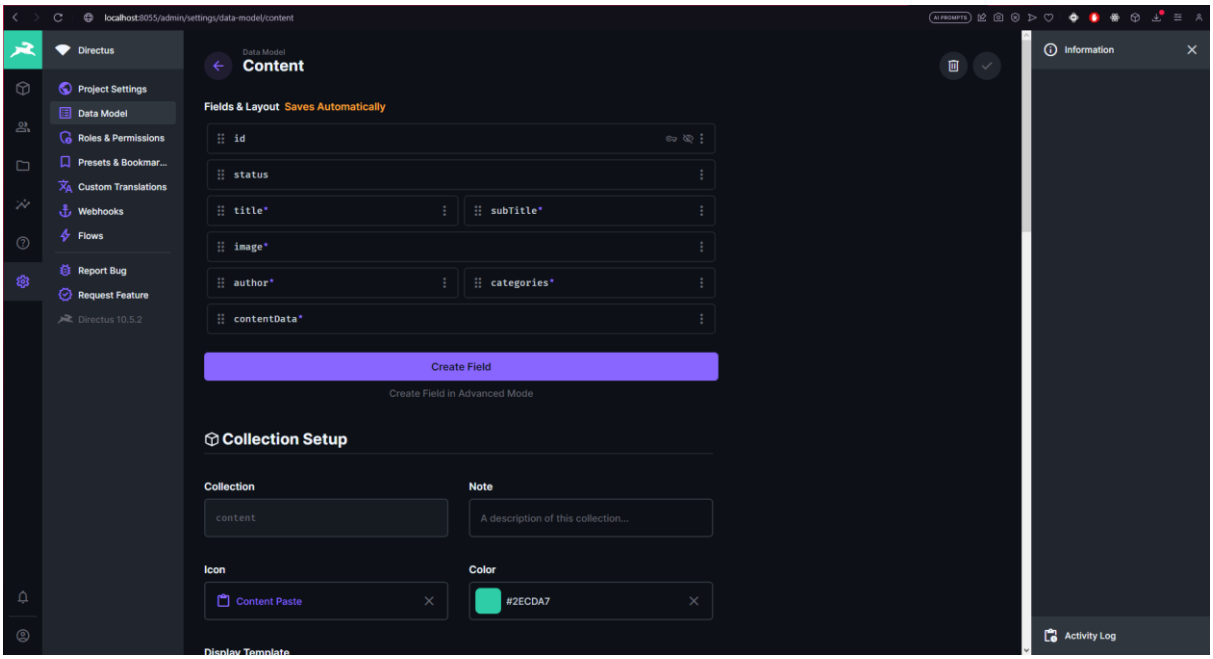
Kreiranje modela sadržaja je vrlo jednostavno i logično – kreira se novi *Data Model*, pri čemu se dodjeljuju odgovarajuće ime, ID i početne postavke za model, a zatim se dodaju odgovarajuća polja koja će model sadržaja sadržavati.

Svako polje je moguće dodatno dizajnirati – primjerice, moguće je dodati zamjenski tekst (eng. *placeholder*), ikonu za ilustraciju polja, mijenjati boju, dodati *slug*, limit znakova, font, promijeniti sam izgled prikaza polja kod uređivanja sadržaja, provjeriti ispravnost unesenog polja, itd. Svakom polju dodatno se može modificirati izgled i redoslijed, polja se mogu sortirati po želji, a moguće je dodati i ikonu te boju za svaki model sadržaja, itd. Alat korisnicima stvarno daje vrlo širok izbor dizajniranja vlastitih CMS sustava. Proces kreiranja prethodnog spomenutog primjera za model sadržaja blog stranice bio je vrlo jednostavan, a sve specifikacije modela sadržaja dodane su bez ikakvih poteškoća. Sljedeća slika 12. prikazuje kreiranje modela sadržaja u alatu Directus.

Dodavanje polja za odabir kategorija kako bi urednici sadržaja mogli odabrati jednu ili više predefiniranih vrijednosti za kategorije blog članka bilo je izrazito jednostavno, jer je ta funkcionalnost već ugrađena u alat Directus te ju korisnicima nudi kao jednu od mnogih zadanih opcija za polja kod modela sadržaja. Polja koje korisnici mogu odabrati su:

- obično polje za unos;
- polje s automatskom ispunom (eng. autocomplete input);
- Block Editor koji služi kao bogati tekst gdje urednici sadržaja mogu dodatno ukrašavati i dodavati dodatne medijske zapise i slično;
- Markdown;
- Tags;

- razni blokovi za selekciju (Toggle, Datetime, Respeator, Map, Coler, Dropdown, itd.);
- razni blokovi polja za povezivanje datoteka, slika, više različitih slika;
- Builder (M2A) – zaslužan za povezivanje više stvari u jednoj medijskoj datoteci;
- M2M, O2M, M2O – više na više, jedan na više, više na jedan;
- mogućnost prevođenja;
- blokovi za prezentaciju;
- blokovi za grupaciju;
- ostale funkcionalnosti.



Slika 12. Kreiranje modela sadržaja pomoću alata Directusa [autorski rad]

Jako velika prednost alata Directus leži u tome što korisnicima, koji posjeduju svoje podatke u SQL bazi podataka, pruža mogućnost da pomoću njihovih API-ja na jednostavan način mogu dohvaćati podatke iz SQL baze, pri čemu se dohvaćeni podaci automatski i dinamično transformiraju u REST točke (eng. *REST Endpoint*) preko kojih su dostupni za dohvaćanje s bilo kojeg IoT uređaja – nije potreba ikakva migracija podataka iz jednog oblika u drugi. Directus korisnicima pruža već izgrađenu front-end aplikaciju koja služi za administratorsku konfiguraciju CMS sustava, kao i front-end aplikaciju za uređivanje i dodavanje stvarnih sadržaja u sustav. Dizajn cijelog sustava prilagođen je korisnicima na način da svatko može upravljati sadržajem, čak i s minimalnim znanjem o programiranju. Alat se ističe modernim dizajnom za koji se može reći da je jedinstven jer ne slijedi neka nepisana

pravila kao većina drugih CMS alata bez grafičkog sučelja. Pruža velik skup funkcionalnosti koje omogućavaju izgradnju čak i najspecifičnijeg CMS sustava – alat je po tom pitanju vrlo fleksibilan. Dokumentacija alata se također redovito ažurira, a zajednica i cijeli Directus tim korisnicima pomažu u velikoj mjeri, na svaki način koji mogu.

Directus ima vrlo malo negativnih strana, od kojih bi najbitnije bile nestabilnost tek izdanih verzija te nekompatibilnost s dodacima (eng. *plugins/addons*) trećih strana.

3.1.2.3. Strapi vs Directus

Prema prikupljenoj statistici s web stranice G2 [18], alat Directus je ocijenjen ocjenom 5 na temelju 23 recenzije, dok je alat Strapi dobio nešto manju ocjenu, 4.5, na temelju 168 recenzija. Prema statistici, alat Directus je u gotovo svakoj kategoriji ocjenjivanja za nijansu bolji od alata Strapi. Potpuna statistika i popis kategorija ocjenjivanja može se vidjeti u tablici 6 koja je kreirana u kombinaciji statistika s web stranica G2 [18] i Restack [19].

Oba navedena alata izvrsne su opcije za poduzeća kojima je potreban sustav za upravljanjem sadržajem bez grafičkog sustava. Također su rješenja otvorenoga koda, čime se dobiva mogućnost modificiranja i proširivanja izvornog koda aplikacije po želji. Nadalje, oba alata namijenjena su poduzećima manjih kapaciteta – do pedeset zaposlenika. U slučaju da je potreban sustav koji zahtjeva veću fleksibilnost i specifičnost, alat Directus je za nijansu bolji, međutim, time se gubi na efikasnosti kod kreiranja modela sadržaja i samoj ispuni sadržaja stvarnim podacima jer su konfiguracija i ispunjavanje vremenski zahtjevniji. S druge strane, ako je potreban općenitiji model sadržaja, bolji je izbor alat Strapi.

Oba alata nude besplatnu („*Community*“) verziju. Također, u slučaju da su bitni izgled i cjelokupni dizajn alata, bez obzira na vrstu korisnika (administrator ili urednik), svakako se preporuča alat Directus, budući da korisnicima nudi visoku fleksibilnost i modifikabilnost dizajna. Nadalje, u slučaju da korisnici podatke pohranjuju u SQL bazu podataka, također se preporučuje alat Directus. Directus dohvaća podatke iz SQL baze podataka i rezultate pretvara oblik pogodan dinamičnim web servisima (API-ima), koji se potom mogu koristiti bilo gdje.

U slučaju da je korisnicima potreban ikakav dodatak ili proširenje sustava, preporuča se alat Strapi, budući da sadrži trgovinu s različitim dodacima koji se na jednostavan način instaliraju u sustav. Directus, s druge strane, ne pruža tu opciju.

Tablica 6. Prikaz rezultata uspoređivanja alata Strapia i Directusa

Generalne procjene	**Directus**	**Strapi**
Zadovoljava zahtjeve	9.3	8.7
Jednostavnost korištenja	9.7	9.1
Jednostavnost postavljanja	8.7	8.8
Jednostavnost administracije	9	8.8
Kvaliteta podrške	9.1	8.5
Je li proizvod bio dobar partner u poslovanju?	9.7	8.6
Smjer proizvoda (% pozitivno)	9.2	8.5
Sadržaj		
Autorstvo sadržaja	9	8.6
Uređivač obogaćenog teksta	9.2	8.2
Verziranje	7.4	7.9
Dodaci/widgeti/aplikacije	9.5	8
Proces odobravanja	7.1	7.6
Raspored sadržaja	7.8	7.5
Upravljanje imovinom	8.9	8.4
Interna pretraga	7.2	7.8
Objavljivanje		
Automatizirano izdavanje	7.2	7.9
Višejezičnost	7.9	8.1
Dizajn		
Širok izbor unaprijed izrađenih predložaka	9	7.5
Brendiranje/teme	9.1	7.1
Prilagodba	9.8	7.9
Razvojni proces		
Upravljanje slikom	8.7	8
Višekratni sadržaj	9	8.6
Neograničena okruženja	8.6	8.1
Form Builder	7.7	8.5
Platforma		
Zajednica korisnika	8.9	8.1
SEO	8.2	8.3
Fleksibilne navigacijske strukture	9.4	8
Upravljanje korisnicima, ulogama i pristupom	7.9	8.4
Skalabilnost poduzeća	8.5	8.5
Internacionalizacija	7.3	8
API / integracije	9	8.7
Analitika		
Personalizacija	7.4	7.9
Izvedba sadržaja	7.6	8.3

(Izvor: G2 i Restack [18] [19])

3.2. CMS za e-trgovinu

Kako se Internet sve više širi, tako se povećava i količina web trgovina i potreba za njima. Ljudi širom svijeta sve više preferiraju kupovati putem interneta, a time se povećava potreba za inovativnim načinom trgovanja putem interneta. Jedno od mogućih inovativnih rješenja bilo je modificirati CMS sustav na način da bude usmjeren prema korisnicima. Tako je nastao CMS e-trgovine (eng. *e-commerce*). Glavni ciljevi CMS-a e-trgovine prema autorima Abdullah, Ahmad, Ismail i Diah [20] bili bi:

- pružanje mogućnosti olakšane i brze kupnje putem interneta;
- olakšavanje praćenja i pretrage proizvoda kod e-trgovine;
- olakšavanje procesa transakcije za online plaćanje;
- pohrana stanja korisničke košarice;
- preporuka proizvoda za korisnike, itd.

Web stranice e-trgovina moraju biti kreirane tako da budu što prihvatljivije korisnicima, dakle web stranice moraju biti jednostavne i jasne. Druga stvar na koju je potrebno obratiti pozornost je sam dizajn web stranica, jer je način prikazivanja proizvoda korisnicima vrlo bitan. Autori su dodatno definirali da bi se bilo dobro pridržavati sljedećih četiriju funkcionalnosti kod izgradnje CMS-a za e-trgovine: upotrebljivost, pouzdanost, pogodnost i jednostavnost korištenja.

CMS e-trgovine posjeduje iste funkcionalnosti koje posjeduju i obični CMS sustavi, kao što su: upravljanje sadržajem (u ovom slučaju je riječ o upravljanju proizvodima), verzioniranje sadržaja, filtriranje sadržaja, objavljivanje sadržaja, potreba za web servisima/REST API-jem te ostale mogućnosti CMS sustava. Dodatne funkcionalnosti koje CMS e-trgovine posjeduju bile bi:

- upravljanje popustima ili grupama proizvoda;
- „Drip Marketing“ – služi za obavještavanje korisnika o proizvodima koji mu se preporučuju;
- elektronično plaćanje;
- sustav darovnih kartica;
- mogućnost višestruke valute;
- proces naručivanja;
- usklađenost sa standardom PCI DSS – kolekcija sigurnosnih standarda za Internet plaćanje;
- mogućnost upravljanja cijenama;
- katalog proizvoda;

- upravljanje proizvodnim katalozima;
- mogućnost recenzije i ocjenjivanja;
- mogućnost analitike prodaje;
- generiranje izvješća o prodaji;
- upravljanje otpremnom robom;
- korisnička košarica;
- integracija društvenih medija;
- integracija treće strane;
- ostala upravljanja web trgovinom.

Dok neka poduzeća nude gotove web aplikacije za e-trgovine, što uključuje upravljanje proizvodima i korisnicima, ostala poduzeća integriraju web servise kao što su REST API ili GraphQL s CMS sustavom te time nude bolje i fleksibilnije rješenje. Dodatno, nude gotova rješenja za popunjavanje kolekcija proizvoda, kao i upravljanje istima na jednom mjestu (bez obzira jesu li podaci pohranjeni u oblaku ili na privatnim serverima), dok se prikaz dinamički dohvaća i generira na željenim IoT uređajima.

3.2.1. Komercijalna rješenja za headless CMS e-trgovine

Sljedeće tri sekcije rada opisuju komercijalne alate (rješenja zatvorenoga koda) čija je svrha izgraditi CMS sustav bez grafičkog sučelja specifičan za e-trgovinu.

3.2.1.1. Commercetools

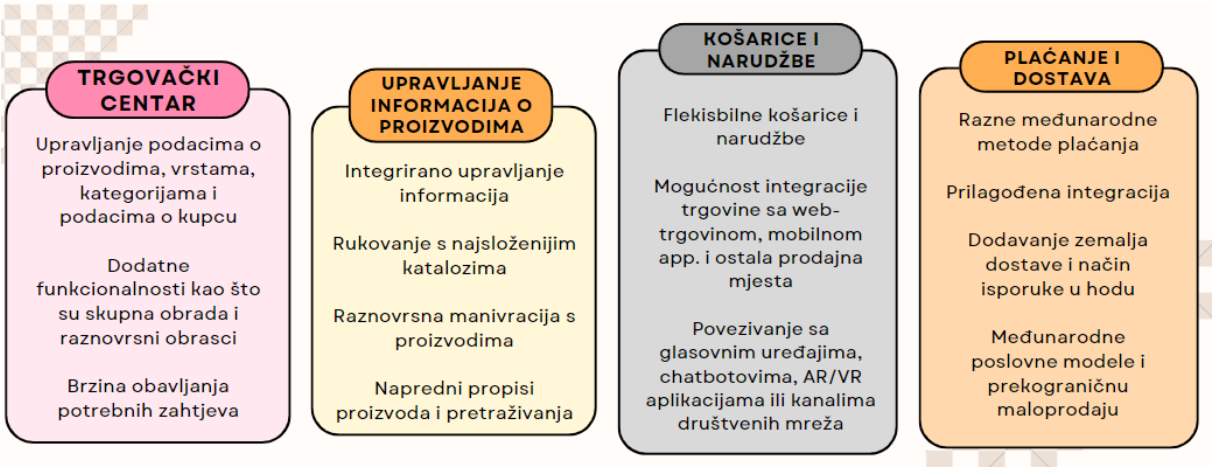
Programski alat Commercetools osnovan je 2006. godine i podržava dva standardna jezika – engleski i njemački. Glavno sjedište tvrtke je u gradu Münchenu u Njemačkoj. Commercetools svojim klijentima nudi beskonačno skaliranje projekata, pri čemu klijenti mogu upravljati s više robnih marki. Korisnici Commercetools-a imaju mogućnost širenja poslovanja na međunarodnoj razini.

Commercetools nudi rješenje za sve poslovne modele: B2B (eng. *Business to Business*), B2C (eng. *Business to Consumer*) i D2C (eng. *Direct to Consumer*). Arhitektura Commercetools-a zasniva se na skalabilnosti, agilnosti i fleksibilnosti, koje korisnicima pružaju veću slobodu pri izgradnji izvanrednih iskustava kod online trgovina. Poznate tvrtke poput Sephore, Ultra Beautyja i BMW-a odabrale su Commercetools kao svoj *headless* CMS sustav za e-trgovinu. Commercetools kombinira razne tehnologije za ostvarenje svojih ciljeva, ukratko poznate kao „MACH“:

- mikro-usluge (eng. Microservice – M) – služe kao neovisne usluge koje olakšavaju implementaciju novih značajki i dodirnih točaka;

- orijentirano prema API-ju (eng. API first – A) – centralizacija podataka i pružanje sadržaja bilo gdje;
- izvorno u oblaku (eng. Cloud-native – C) – usluge se nalaze u oblaku, čime se postiže neovisnost o poslužitelju te se omogućava automatsko ažuriranje i skaliranje;
- trgovina bez grafičkog sučelja – (eng. Headless commerce – H) – pruža maksimalnu fleksibilnost i brzinu svojim korisnicima.

Glavne funkcionalnosti koje pruža Commercetools nalaze se u slici 13.



Slika 13. Prikaz glavnih funkcionalnosti alata Commercetoolsa [21]

Commercetools u javnost nije dao cijene svojih poslovnih paketa, već je izdao nekoliko osnovnih informacija o svakom poslovnom paketu koje se nalaze u sljedećoj tablici 7.

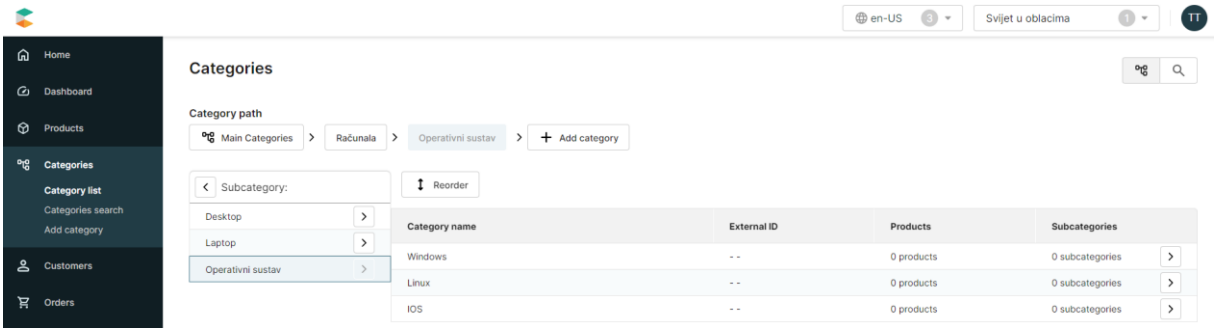
Tablica 7. Prikaz svih poslovnih paketa alata Commercetoolsa

Composable Commerce and Frontend	Composable Commerce for B2B	Composable Commerce B2C
Cijena po dogovoru	Besplatna proba	Besplatna proba
Pruža kompletan paket alata za poslove s korisnicima i programerima koji omogućavaju brzo stvaranje, optimizaciju i prilagođavanje digitalnih izloga	Tvrtkama nudi pomoć za isporuku izvanrednog korisničkog iskustva kupnje, bez obzira na veličinu tvrtke ili razinu složenosti kupnje	Tvrtkama nudi mogućnost rasta, isporuke izvanrednog iskustva kupnje, kao i mogućnost da ostanu različite od drugih dok brzo inoviraju kako bi bile spremne za ono što slijedi.

(Izvor: Rea [22]

Alat je dostupan nalazi na web stranici Commercetoolsa, a za pristup alatu potrebno je imati korisnički račun. Nakon registracije, korisnici mogu testirati sve funkcionalnosti alata kroz probni rok od šezdeset dana. Prilikom korištenja alata, korisnici prvo moraju definirati željene

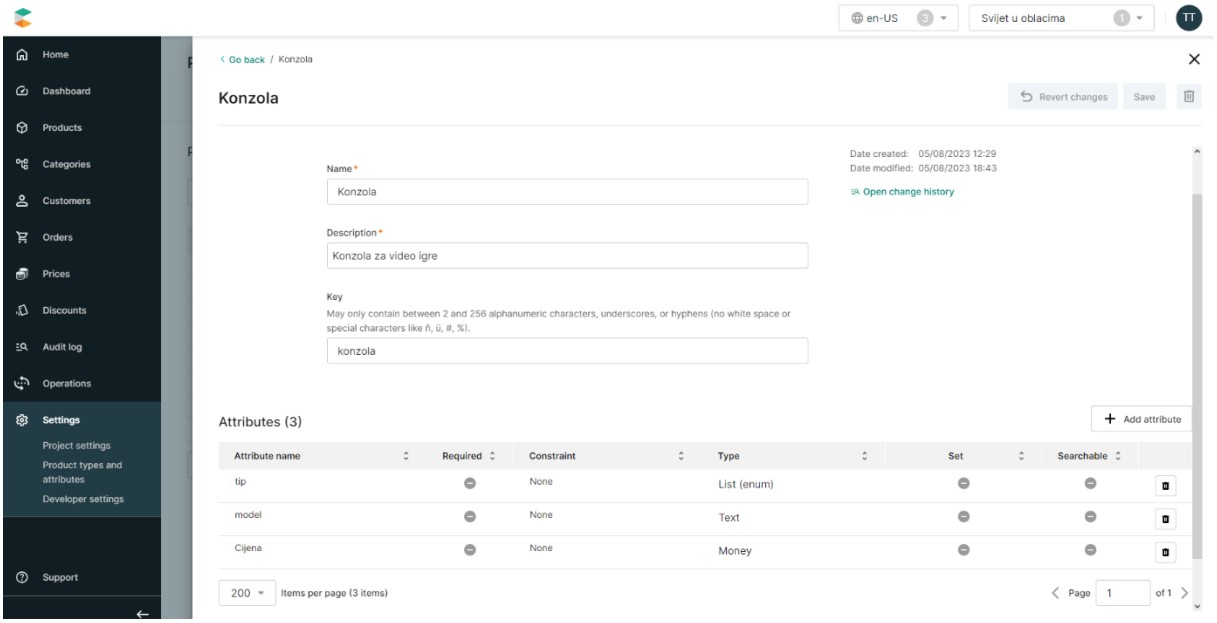
kategorije i potkategorije svojih proizvoda. Primjerice, kod jednostavne e-trgovine, čija su tematika IoT uređaji, kreiraju se sljedeće kategorije: računala, mobilni uređaji, konzole i ostala tehnika. Kategoriji računala se potom mogu dodati potkategorije kao što su desktop, laptop i operacijski sustav, gdje se kategorija „operacijski sustav“ može podijeliti na potkategorije: Windows, Linux i iOS. Primjer ovako organiziranih kategorija može se vidjeti na slici 14.



Slika 14. Prikaz kreiranja kategorija proizvoda u alatu Commercetools [autorski rad]

Nakon dodavanja kategorija, slijedi dodavanje tipova proizvoda i modela proizvoda koji će se koristiti prilikom dodavanja stvarnih proizvoda u sustav. Stvara se novi tip proizvoda, pri čemu se definira naziv, opis proizvoda te svi njegovi atributi koji će se ispunjavati prilikom dodavanja stvarnih proizvoda. Atributi tipova proizvoda mogu biti: Booleova vrijednost, brojčana vrijednost, tekst, novčana vrijednost, datum i vrijeme, lista te referenca. Osim običnih postavki tipa proizvoda postoje i napredne postavke tipa proizvoda. Primjer kreiranja tipa proizvoda, koji se nadovezuje na prethodno kreirane kategorije, izvodi se na način prikazan na slici 15.

Nakon kreiranja tipova proizvoda, korisnicima se konačno omogućuje unos stvarnih proizvoda u sustav. Kod dodavanja proizvoda u sustav, korisnik mora ispuniti generalne podatke o proizvodu, kao što su naziv, opis, kategorija i ostale generalne postavke. Alat potom nudi mogućnost kreiranja više varijacija proizvoda – primjerice, proizvod „kožna jakna“ varira na temelju veličine, širine ramena, duljine rukava i slično. U primjeru igraćih konzola, prilikom dodavanja proizvoda PlayStation 5, dodaju se dvije varijacije – s čitačem diska i bez čitača diska. Prilikom kreiranja varijacija proizvoda definiraju se opći podaci, a dodatno se mogu unijeti i fotografije za svaku varijaciju, cijena varijacije, njena specifikacija (valuta, cijena, datumi od/do kada vrijede, itd.) te informacija o dostupnosti proizvoda.



Slika 15. Prikaz kreiranja tipa proizvoda u alatu Commercetoolsu [autorski rad]

Postoje mnogobrojne pozitivne strane alata Commercetools:

- alat klijentima pruža veliku količinu funkcionalnosti pomoću kojih na jednostavan način mogu izgraditi web trgovine;
- vrlo je fleksibilan i korisnicima omogućuje izgradnju CMS sustava e-trgovine za bilo koje tipove i varijacije proizvoda;
- upravljanje narudžbama;
- upravljanje korisnicima i njihovom grupacijom;
- upravljanje košaricom;
- upravljanje cijenama, popustima i grupama popusta;
- kreiranje vlastitih krajnjih točaka za web servise;
- kvalitetna i detaljna dokumentacija koja se često ažurira.

Napredna svojstva i mnoštvo funkcionalnosti alata stvaraju i negativni efekt, koji se manifestira u obliku nekoliko mana alata:

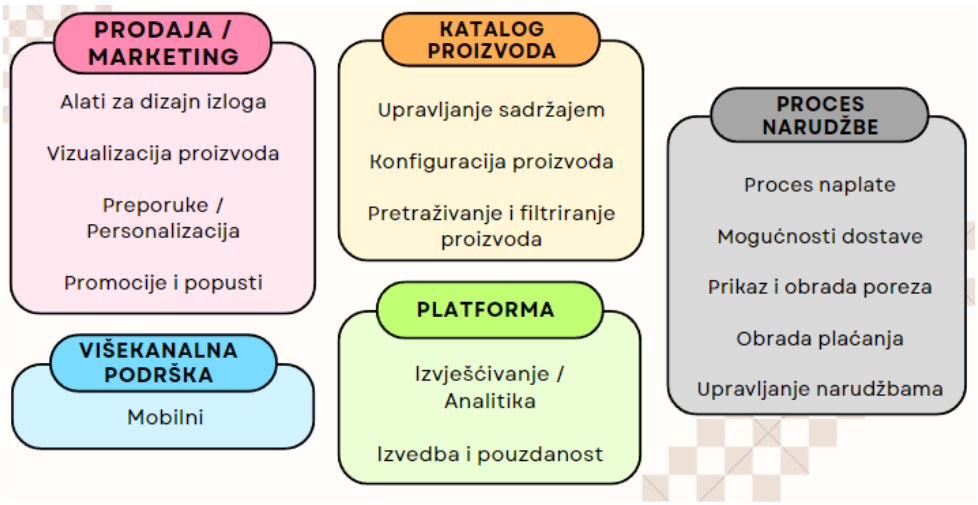
- prevelika kompleksnost aplikacije;
- potrebna dodatna znanja o samom alatu;
- neprilagođenost alata običnom korisniku koji ne posjeduje znanja o programiranju ili vođenju CMS sustava e-trgovine

3.2.1.2. Shopify

Tvrtka koja proizvodi programski alat Shopify osnovana je 2006. godine, a podržani jezik je engleski. Glavno sjedište tvrtke je u gradu Ottawa u Kanadi. Alat Shopify jedna je od vodećih platformi za upravljanje sadržajem e-trgovina bez grafičkog sučelja temeljenih na oblaku. Korisnicima pruža mogućnost vođenja više prodajnih kanala – primjerice: mobilni uređaji, društvene mreže, tržnice, fizičke lokacije i skočne trgovine.

Glavna svrha alata je kreiranje web trgovine na jednostavan način pomoću predefiniranih tema i blokova web trgovine, a korisnicima je dostupan i moćan stražnji ured (eng. *back-office*). Alat pruža i korisničko sučelje za trgovce koji su zaduženi za cijeli sadržaj e-trgovine, kao i jedinstven pogled na cjelokupno poslovanje. Na raspolaganju je i skup API-ja, prilagođeni tokovi košarica te naplata koju je moguće integrirati sa željenim sustavom.

Shopify trenutno pokreće više od 800 tisuća tvrtki u približno 150 zemalja. Poznatije tvrtke koje su se odlučile za platformu Shopify su: Tesla Motors, Budweiser, Red Bull, LA Lakers, New York Stock Exchange i GoldieBlox. Važnije prednosti i funkcionalnosti alata nalaze se u slici 16.



Slika 16. Prikaz glavnih funkcionalnosti alata Shopifya [23]

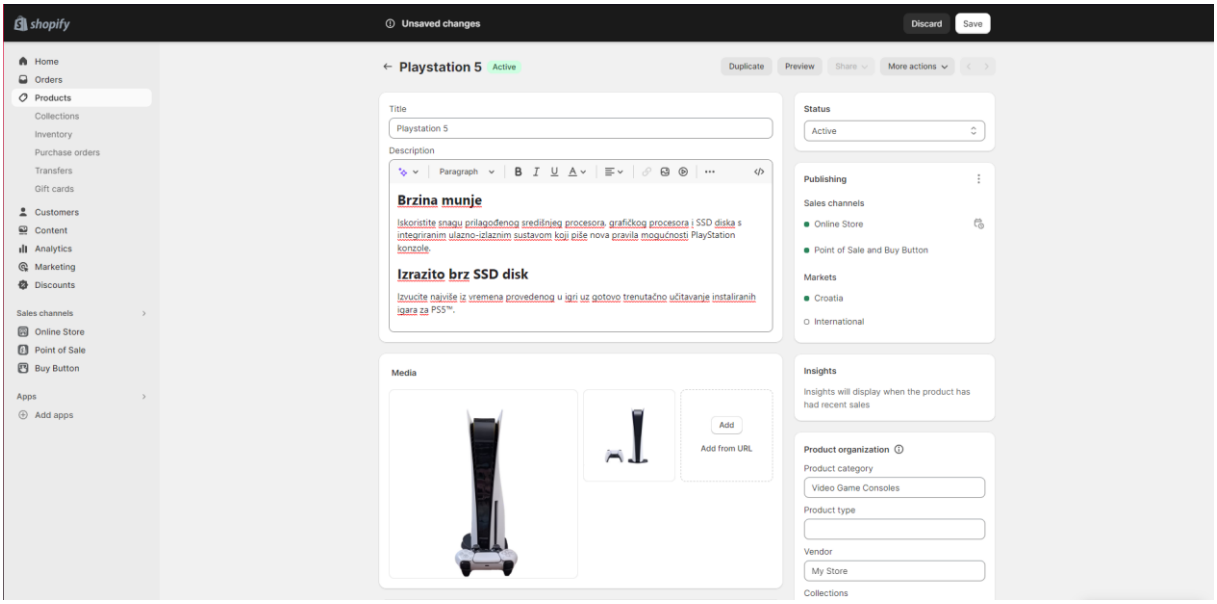
Shopify nudi tri poslovna plana: osnovni plan, Shopify plan i Napredni plan. Svaki se plan plaća na mjesečnoj bazi, a plaćanje na godišnjoj bazi korisnicima osigurava 25% popusta. Prva tri mjeseca su gotovo besplatna za svaki plan – potrebno je platiti samo jedan dolar mjesečno. Detalji o svakom poslovnom planu nalaze se u tablici 8.

Tablica 8. Prikaz svih poslovnih planova alata Shopifya

Osnovni plan	Shopify plan	Napredni plan
$32 mjesečno	$92 mjesečno	$399 mjesečno
Sve što je potrebno za kreiranje trgovine, slanje proizvoda i obradu plaćanja	Podizanje poslovanja na razinu s profesionalnim izvješćivanjem i više računa za osoblje	Najbolje od Shopify-ja uz prilagođena izvješća i najniže naknade za transakcije
Osnovna izvješća	Stručna izvješća	Alat za izradu prilagođenih izvješća
Do 1000 mjesta inventara	Do 1000 mjesta inventara	Do 1000 mjesta inventara
2 korisnička računa	5 korisničkih računa	15 korisničkih računa
Besplatna proba	Besplatna proba	Besplatna proba
x	x	Cijene dostave izračunate od trećih strana
x	Pružanje automatizacije e-trgovine	Pružanje automatizacije e-trgovine
x	x	Mogućnost upravljanja plaćanja carine i uvoznih troškova
Online trgovina, beskonačno mnogo proizvoda, 24/7 podrška, kanali za prodaju, ručno kreiranje naloga, kodovi za popust, besplatni SSL certifikat, oporavak napuštene košarice, darovne kartice i još mnogo toga		

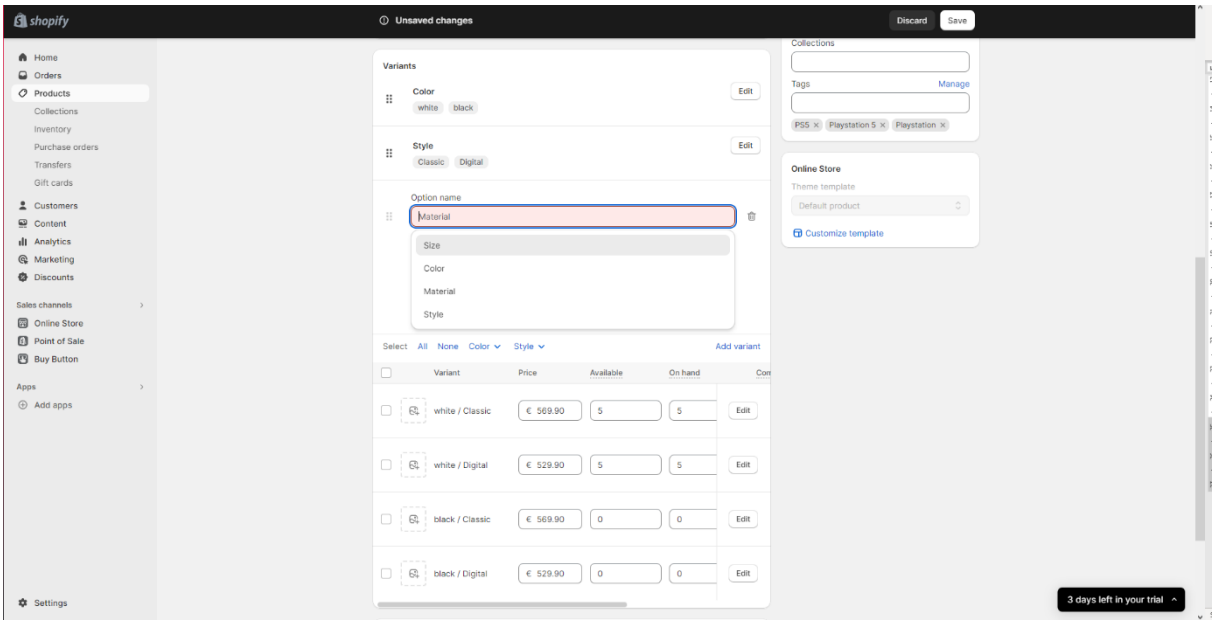
(Izvor: Shopify [24])

Shopify je vrlo jednostavan i prilagođen korisnicima te posjeduje bogatu predefiniranu listu kategorija proizvoda koje se kod kreiranja proizvoda jednostavno odaberu – korisnici ne trebaju kreirati popise i liste kategorija. Proces kreiranja novog proizvoda vrlo je jednostavan i brz – korisnici unose generalne podatke o proizvodu i opis proizvoda u obliku obogaćenog teksta, što im omogućava detaljno dizajniranje opisa proizvoda. Navedeni proces kreiranja novog proizvoda prikazan je na slici 17.



Slika 17. Kreiranje novog proizvoda u alatu Shopifyu [autorski rad]

Za svaki kreirani proizvod moguće je dodatno specificirati različite varijacije i na temelju varijacije definirati cijenu i ostale podatke o varijaciji. Primjerice, varijacije koje se po zadanim postavkama mogu postaviti za proizvode su veličina, boja, materijal i stil. Primjer specificiranja varijacije proizvoda prikazan je na slici 18.



Slika 18. Prikaz dodavanja varijacije o proizvodu [autorski rad]

Proizvodi se nakon unosa u sustav mogu pretpregledati, a nakon toga se može unijeti i informacija o dostupnosti proizvoda. Nakon unosa, proizvodi su dostupni preko Shopify web servisa. Alat pruža i mogućnost korištenja gotovih te kreiranja novih korisničkih sučelja za posjetitelje e-trgovine. Korisnici pomoću alata dodatno mogu upravljati: proizvodima, narudžbama, analitikom, izvješćima, marketingom, popustima, online trgovinom, online plaćanjem, korisničkom košaricom, itd.

Pozitivne strane alata Shopify uključuju:

- visok stupanj prilagodljivosti korisnicima;
- intuitivno i razumljivo korisničko sučelje;
- mogućnost stvaranja modernih web stranica za e-trgovinu i upravljanje proizvodima uz osnovna znanja o programiranju;
- integraciju Shopify projekta sa željenim web stranicama pomoću web servisa (GraphQL ili REST API);
- brojne funkcionalnosti pomoću kojih je izrada e-trgovine vrlo jednostavna;
- fleksibilnost (do određenih granica);

- mogućnost izrade detaljnog opisa proizvoda;
- tržnicu s dodacima trećih strana.

S druge strane, mane alata Shopify bile bi:

- manjak kvalitete i dostupnosti korisničke podrške;
- manjak fleksibilnosti – mogućnost prilagodbe predložaka je ograničena;
- financijski troškovi – provizija;
- kompleksnost određenih funkcionalnosti alata koja zahtijeva određena znanja i vještine.

3.2.1.3. Commercetools vs Shopify

Shopify se pokazao jako dobrim alatom na temelju statistike koju je objavila stranica G2. Dobio ocjenu 4.4 na temelju 4370 recenzija, dok je Commercetools dobio ocjenu 4.7 na temelju 11 recenzija, što je izrazito malen broj. Ukupna statistika nalazi se u tablici 9 koja je izgrađena na temelju kombinacije triju različitih statistika, objavljenih na sljedećim stranicama: G2, Gartner Peer Insights i SelectHub [25], [26], [27].

Tablica 9. Prikaz usporedbe Commercetoolsa i Shopifya

Opće ocjenjivanje	**Shopify**	**Commercetools**
Ukupno zadovoljstvo alatom	8.7	9.4
Jednostavnost korištenja	8.9	9.4
Jednostavnost postavljanja	8.8	9.4
Jednostavnost administracije	8.9	8.9
Kvaliteta podrške	8.4	9.2
Je li proizvod bio dobar partner u poslovanju?	8.8	9.4
Smjer proizvoda (% pozitivno)	8.7	10
E-trgovina bez grafičkog sučelja		
Odvajanje	8.3	8.1
REST API-ji	8.6	7.4
Prilagodba	8.3	8.6
Proširivost	8.6	9
Marketing		
Promocije i popusti	8.3	8
Recenzije proizvoda	8.3	8.6
SEO	8.2	8.8
Katalog proizvoda		
Upravljanje sadržajem	8.5	7.9
Konfiguracija proizvoda	8.5	9.1
Pretraživanje i filtriranje proizvoda	8.4	7.4
Proces naručivanja		

Proces naplate	8.8	8.4
Mogućnosti dostave	8.6	9.1
Obrada plaćanja	8.9	6.1
Upravljanje narudžbama	8.8	8.4
Upravljanje zalihama	8.6	5.7
Vise kanalno ispunjenje	8.5	8
Podrška za više kanala		
Mobilna platforma	8.8	9
Trgovina društvenih mreža	8.7	9.1
Služba za korisnike	8.5	6.6
Lokator trgovine	8.4	8.6
Integracija trgovine sa servisima treće strane	8.4	9
B2B	8.3	9.1
Platforma		
API	8.3	7.4
Prilagodba	8.4	6.4
Integracije s dodatnim alatima	8.6	7.6
Izvješćivanje / Analitika	8.4	6
Sigurnost	8.9	9.2
Upravljanje više lokacija	8.4	8
Administratorska prava pristupa	8.8	9.3
Internacionalizacija	8.5	8
Izvedba i pouzdanost	8.9	9.5

(Izvor: G2, Gartner Peer Insights i SelectHub [25], [26] i [27])

Shopify je izvrstan alat za manja poduzeća (do pedesetak zaposlenika), dok je Commercetools namijenjen poduzećima velikih kapaciteta (od tisuću zaposlenika na dalje). Ako se traži brzo rješenje za CMS sustav e-trgovine, odgovor na to pitanje je Shopify. Shopify omogućava kreiranje željene web stranice za e-trgovinu te ujedno pruža mogućnost upravljanja proizvodima, sve na jednom mjestu. S druge strane, ako je korisnicima potreban CMS sustav bez grafičkog sučelja za e-trgovinu velikih kapaciteta s mogućnošću jednostavnog proširenja i nadogradnje, bolji alat je Commercetools.

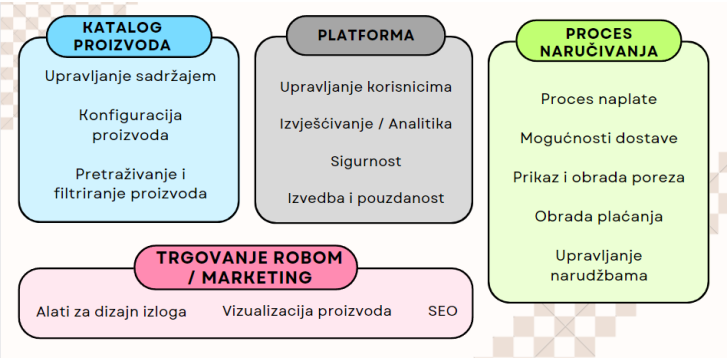
Alat Shopify namijenjen je vlastitoj upotrebi ili upotrebi od strane malih poduzeća jer pruža sučelje koje je prilagođeno običnim korisnicima koji ne posjeduju prevelika znanja o programiranju ili vođenju CMS sustava e-trgovine, dok je Commercetools namijenjen profesionalnoj upotrebi od strane većih poduzeća, koja mogu ili moraju utrošiti dodatno vrijeme na temeljitu konfiguraciju i postavljanje CMS sustava radi postizanja uspješnog poslovanja.

3.2.2. Rješenja otvorenog koda za headless CMS e-trgovine

U nastavku slijedi opis sustava za upravljanje sadržajem bez grafičkog sučelja e-trgovina čija su rješenja otvorenoga koda. Opisani alati su PrestaShop i OpenCart.

3.2.2.1. PrestaShop

Tvrtka koja proizvodi programski alat PrestaShop osnovana je 2007. godine, a podržani jezici su engleski, francuski, talijanski i tri druga jezika. Glavno sjedište tvrtke je u gradu Parizu, glavnom gradu Francuske. PrestaShop je jedno od vodećih rješenja otvorenog koda za CMS e-trgovine koje se bave trgovinom u Europi i Latinskoj Americi s preko 300.000 korisnika. Prema njihovom financijskom izvještaju, u 2021. godini ostvarili su preko 24 milijarde eura prometa u online prodaji. PrestaShop svojim klijentima nudi korisničko sučelje s cijelim dizajnom za obične korisnike koji pregledavaju i kupuju proizvode te za administrativne korisnike koji upravljaju proizvodima. Alat klijentima pruža mnogo dostupnih modula i tema za dizajniranje željenih web stranica, dok se administrativnim korisnicima pruža potpuna kontrolu nad kreiranjem CMS sustava i svim podacima u njemu. PrestaShop vjeruje da bi pomoću tehnologije poslovanje trebalo biti poboljšano, a ne ograničeno. Pružaju podršku korisnicima, posjeduju bogatu dokumentaciju te mnoštvo treninga i video zapisa namijenjenih korisnicima koji su tek započeli rad s PrestaShop projektom. PrestaShopova velika zajednica pozitivno doprinosi PrestaShopu koristeći blogove i forume. Slika 19. prikazuje glavne funkcionalnosti koje pruža alat PrestaShop.



Slika 19. Prikaz glavnih funkcionalnosti alata PrestaShopa [28]

Alat PrestaShop je u potpunosti besplatan i ne nudi ikavko rješenje u oblaku, ali koristi internu trgovinu pomoću koje korisnici mogu kupiti raznovrsne dodatke, teme, usluge i dizajne koji mogu biti besplatni, ali mogu koštati i do dvjestotinjak eura ili čak i više. Korisnici moraju uzeti u obzir i mjesečnu naknadu za pružanje usluge *hostinga* sustava. Cijene, naravno, ovise o korisničkim potrebama i željama.

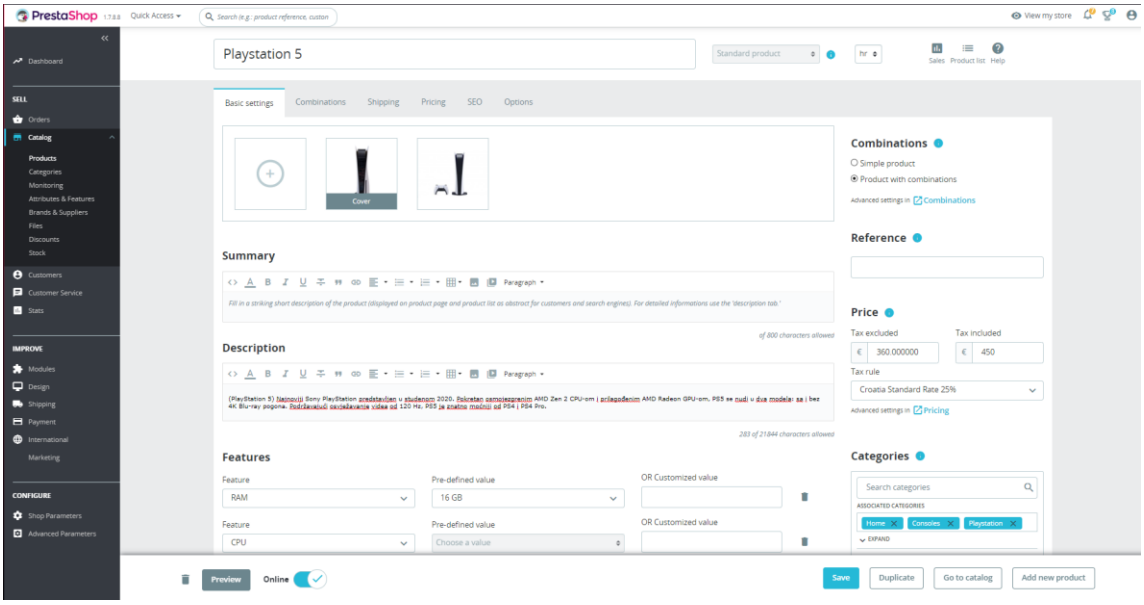
Instalacijski proces samog alata pokazuje se prilično kompliciranim zbog nemogućnosti instalacije posljednje verzija alata. Prilikom procesa instalacije, stalno je dolazilo do problema neuspješnog kreiranja datoteke s parametrima. Problem je bio riješen instalacijom prethodne, stabilne verzije alata. Podizanje servera zahtijevalo je virtualizacijski alat poput XAMPP-a ili Dockera.

Prije unosa proizvoda u sustav, potrebno je kreirati željene kategorije proizvoda. Nakon kreiranja kategorija proizvoda, potrebno je kreirati atribute te dodatne značajke proizvoda. Atributi su svojstva poput boje, dimenzija itd., dok su dodatne značajke istaknute posebne vrijednosti – primjerice, u slučaju računala, dodatne značajke bile bi RAM, CPU, GPU, SSD, HDD, itd. Sljedeći korak je kreiranje brendova proizvođača, pri čemu se mogu specificirati naziv, logo, adresa sjedišta, proizvodi, itd.

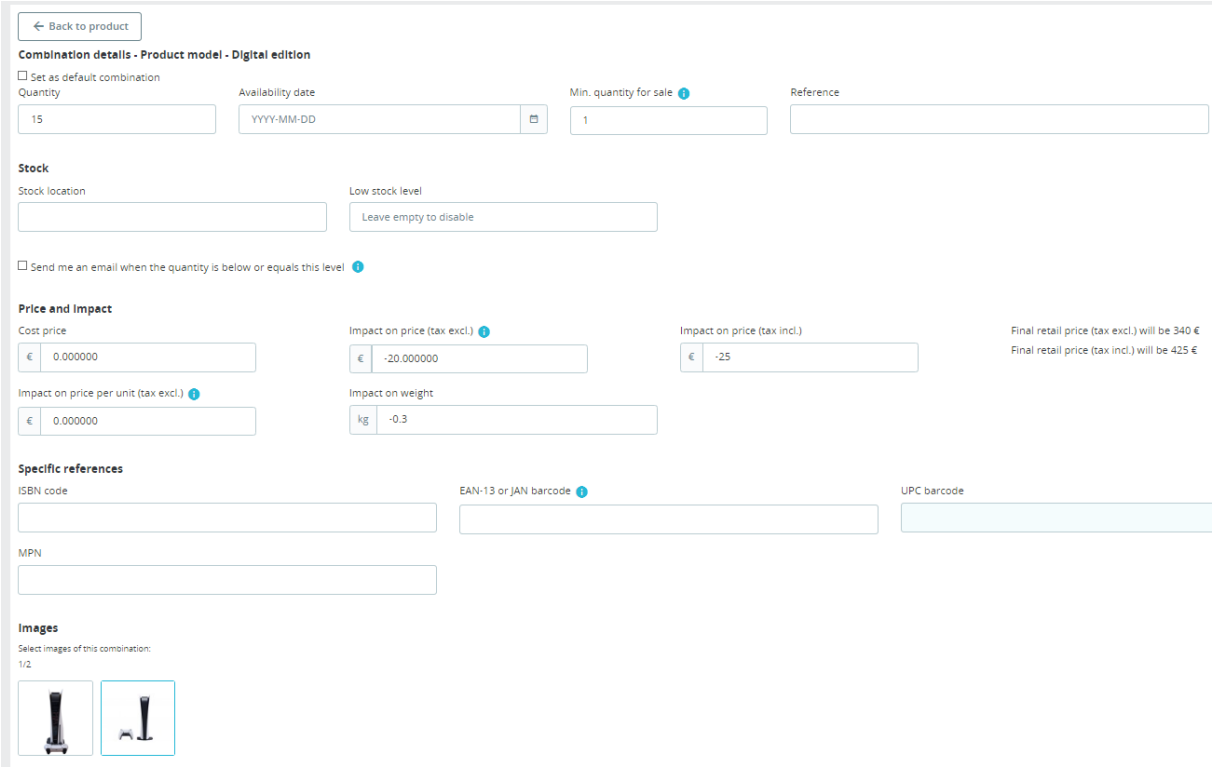
Naposljetku, unos proizvoda u sustav provodi se kroz četiri koraka:

1. unos općih postavke o proizvodu – naziv proizvoda, cijena, slike, opis, kratki opis, prethodno kreirani atributi, dodatne značajke i proizvođači, slični proizvodi i ostale opće stvari;
2. dodavanje kombinacija (varijacija) proizvoda
3. unos logističkih podataka – težina i veličina proizvoda, troškovi dostave i odabir željenih prijevoznika;
4. unos cijena proizvoda, SEO konfiguracija i ostale informacije o proizvodu.

Na slici 20 prikazan je način unosa novog proizvoda u alatu PrestaShop, dok je na slici 21 prikazan postupak kreiranja kombinacije(varijacije) proizvoda.



Slika 20. Prikaz dodavanja novog proizvoda u alatu PrestaShopu [autorski rad]



Slika 21. Prikaz dodavanja varijacije o proizvodu u alatu PrestaShopu [autorski rad]

Ostale mogućnosti koje nudi alat PrestaShop su: pregled narudžbi i faktura, izdavanje potvrda o kreditu i potvrda o dostavi, pohrana korisničkih košarica, praćenje potražnje o proizvodima, postavljanje filtera, upravljanje popustima i zalihama, pregled informacija o korisnicima, pregled korisničke podrške (praćenje pitanja i poruka korisnika te praćenje vraćenih proizvoda), praćenje raznovrsnih statistika o prodanim proizvodima, mogućnost nadogradnje CMS sustava pomoću raznih dodataka, postavljanje i mogućnost promjene dizajna web trgovine, postavljanje dostavljača, postavljanje metode plaćanja te postavljanje željenih postavki za internacionalizam.

Pozitivne strane alata PrestaShop su:

- besplatan CMS sustav e-trgovine;
- web servisi koji omogućavaju dohvat podataka;
- korisnička podrška putem raznih foruma, blogova i prijedloga koje dolaze od zajednice otvorenoga koda;
- mnogobrojne funkcionalnosti i značajke – više od 600 značajki poput: praćenja proizvoda, upozorenja o rasprodaji, kreiranja popusta na temelju jednog proizvoda, kategorija ili grupa proizvoda itd.;
- fleksibilnost;
- mogućnost vođenja sustava na međunarodnoj razini;

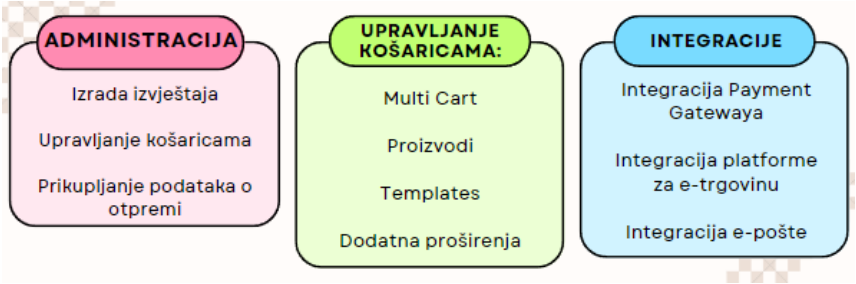
- napredan SEO – korisnici alata mogu specifično odrediti meta podatke i njihove opise za svaki proizvod, kategoriju proizvoda i URL;
- velik izbor opcija plaćanja – PayPal Standard i Pro verzija, Skrill, Stripe, WorldPay, offline obrada i ostale mogućnosti plaćanja;
- dobra prilagođenost korisničkih sučelja za kupce i administraciju s gotovim i prilagodljivim dizajnima i temama.

S druge strane, mane alate PrestaShop bile bi:

- prevelika kompleksnost i neprilagođenost početnicima;
- velik broj funkcionalnosti koje su potrebne za neke projekte su zaključane te ih je potrebno u internoj trgovini s dodacima;
- nepraktičnost alata kao CMS sustava bez grafičkog sučelja za e-trgovinu – potrebno je slati nekoliko zahtjeva kako bi se došlo do svih željenih podataka;
- manjak skalabilnosti;
- korisničko sučelje za mobilne uređaje.

3.2.2.2. OpenCart

Tvrtka koja proizvodi programski alat OpenCart osnovana je 2005. godine, a podržava više od 52 svjetska jezika. Glavno sjedište tvrtke je u Hong Kongu. OpenCart je jedan od brojnih alata otvorenoga koda temeljen na skriptnom jeziku PHP za prikaz, kupovanje, kreiranje i upravljanje proizvodima online trgovine. Alat je jednostavan za korištenje te korisnicima nudi početnu šablonu (eng. *template*) s cijelim dizajnom web trgovine. Kupci se mogu prijaviti u aplikaciju te pregledavati i kupovati proizvode, dok se administrativnim korisnicima pruža mogućnost upravljanja, dodavanja i ažuriranja željenih proizvoda. OpenCart pruža bogatu trgovinu s više od 13.000 modula, tema za pokretanje web trgovine i ostalih proširenja za rast i specijalizaciju korisničkog poslovanja. Pomoću OpenCartove trgovine, korisnici mogu pronaći prekrasnu temu za bilo koju tematiku svoje web trgovine, a dodatne usluge koje trgovina pruža su: integracije, razne usluge plaćanja, metode dostave, integracija društvenih medija, marketing računovodstvo, izvješćivanje, prodaja te jezični paketi. Pomoću OpenCart-a korisnici mogu brzo i jednostavno kreirati e-trgovinu koja će biti u potpunosti funkcionalna kroz svega nekoliko minuta. Glavne funkcionalnosti koje pruža OpenCart ilustrirane su na slici 22.



Slika 22. Prikaz glavnih funkcionalnosti alata OpenCarta [29]

OpenCart svojim korisnicima pruža besplatnu verziju alata, gdje su korisnici zaduženi za sve – konfiguraciju, postavljanje, upravljanje i hosting servera. OpenCart nudi i rješenje temeljeno u oblaku, gdje nude tri poslovna plana – Brončani, Srebrni i Zlatni plan. Poslovni planovi i njihove karakteristike prikazani su u tablici 10.

Tablica 10. Prikaz poslovnih planova alata OpenCarta

Brončani plan	**Srebrni plan**	**Zlatni plan**
$59 mjesečno	$99 mjesečno	$199 mjesečno
Najbolje za mala poduzeća	Najbolje za srednja poduzeća	Najbolje za velika poduzeća
1 CPU	2 CPU-a	4 CPU-a
2 GB RAM-a	4 GB RAM-a	8 GB RAM-a
Jedna domena dućana	Tri domene dućana	Deset domena dućana
5 GB mjesta za pohranu	25 GB mjesta za pohranu	75 GB mjesta za pohranu
1 manualni zahtjev za sigurnosnu kopiju	3 manualna zahtjeva za sigurnosnu kopiju	5 zahtjeva za sigurnosnu kopiju
x	SSL certifikat	SSL certifikat
X	Korisnička podrška putem e-maila	Korisnička podrška putem e-maila
X	x	Korisnička podrška putem telefona

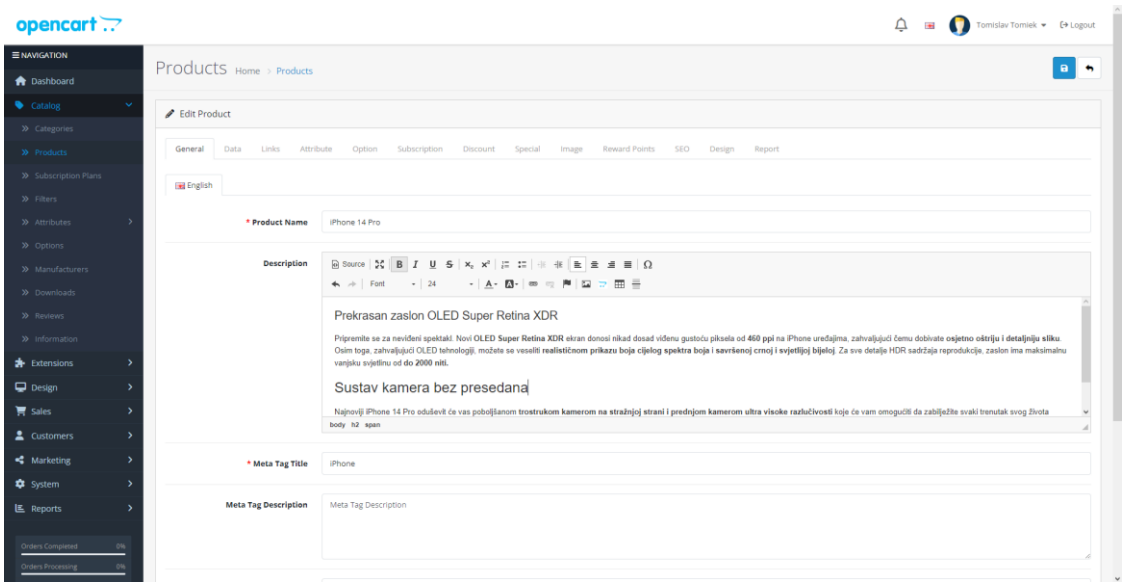
(Izvor: Tsang [29])

Instalacija alata OpenCart bila je malo kompleksnija od instalacije preostalih alata. OpenCart zahtijeva alate treće strana i bazu podataka kako bi mogao funkcionirati lokalno na računalu. Moguće ga je podići na željenom serveru ili lokalno, pomoću virtualizacijskih alata poput XAMPP-a ili Docker-a. Nakon instalacije alata na drugom serveru ili u virtualizacijskom alatu, isti je potrebno konfigurirati kako bi dobio svojstva headless CMS sustava, tj. kako bi se omogućio dohvat podataka pomoću web servisa.

Kreiranje novih proizvoda nije kompleksan proces – najprije se kreiraju željene kategorije i potkategorije proizvoda, a potom se prelazi na unos proizvoda. Alat pruža mnogo predefiniranih funkcija i konfiguracija za kreiranje proizvoda, a sam proces kreiranja proizvoda se provodi kroz sljedeće korake:

1. definiranje generalnih postavki o proizvodu – naziv proizvoda, opis proizvoda koji se može dodatno oblikovati po želji (obogaćeni tekst), kreiranje *tagova* i ključnih riječi;
2. dodavanje stvarnih podataka o proizvodu – model, SKU, UPC, cijena proizvoda, porez, podaci o dostupnosti, dodatne specifikacije o proizvodu (dimenzije, težina i slično);
3. definiranje povezanosti proizvoda – navode se kategorije proizvoda, filtracija, pohranjivanje, povezani proizvodi i slično;
4. definiranje atributa proizvoda – specifikacije o proizvodu (primjerice, atributi za procesor bili bi *socket*, tip procesora, broj jezgri, GPU, hladnjak, brzina i cache memorija);
5. definiranje dodatnih opcija koje kupci mogu odabrati prije same kupnje proizvoda – boja, oblik, itd.;
6. mogućnost definiranja popusta, specijalnih popusta, postavljanje jedne ili više slika o proizvodu, mogućnost definiranja potrebnog broja bodova koji su nužni kako bi kupci mogli kupiti proizvod (kupovanje proizvoda može se nagrađivati bodovima, koje kupci mogu trošiti za kupovanje određenih proizvoda po jeftinijoj cijeni), definiranje SEO, dizajn te definiranje izvještaja.

Na slici 23 prikazan je primjer kreiranja proizvoda pomoću alata OpenCart.



Slika 23. Prikaz kreiranja novog proizvoda u alatu OpenCart [autorski rad]

Potrebno je napomenuti da se filtri, atributi i opcije korišteni kod unosa proizvoda u sustav moraju unaprijed kreirati kako bi se mogli odabrati prilikom unosa proizvoda u sustav.

Pozitivne strane alata OpenCart su:

- mogućnost kreiranja vlastitih korisničkih uloga koje se dodjeljuju korisnicima u svrhu grupiranja uloga i ljudi;
- mogućnost kreiranja različitih web trgovina pomoću jednog korisničkog sučelja;
- mogućnost postavljanja različitih promocija, popusta i kupona za određene proizvode, kao bi se privukla pozornost korisnika;
- mogućnost kreiranja sigurnosne kopije cijelog CMS sustava i njegovo vraćanje na prethodne verzije sustava.

S druge strane, mane alata OpenCart su:

- alat je nastao 2005. godine kad je skriptni jezik PHP, u kojem je OpenCart napisan, bio popularan, a po današnjim standardima se smatra zastarjelom tehnologijom – otežana je izmjena dizajna na frontendu;
- ograničen izbor postavljanja načina dostave – kupci mogu birati samo između predefiniranih načina dostave;
- slabije performanse, posebice kod većih baza podataka ili u slučaju opterećenosti web stranice;
- premala skalabilnost alata;
- nedostatak izravne službe za korisnike – pitanja i problemi se rješavaju preko foruma, dokumentacije ili video zapisa.

3.2.2.3. PrestaShop vs OpenCart

Prema statistici koji je objavila stranica G2 [30], alat PrestaShop je dobio 4.3 ocjenu na temelju 147 recenzija, a OpenCart je dobio istu ocjenu 4.3 na temelju 102 recenzije. Prema cjelokupnoj usporedbi, alat OpenCart je za nijansu bolji od alata PrestaShop, što se može i vidjeti u tablici 11.

Oba alata su rješenja otvorenoga koda namijenjena za CMS sustav e-trgovine te su najbolji za poduzeća malih kapaciteta, do maksimalno 50 zaposlenika. Ako korisnici zahtijevaju rješenja bez grafičkog sučelja, svakako se preporuča odabir komercijalnih alata, pošto rješenja otvorenog koda ne nude dovoljno dobro prihvatljivo rješenje za web servise.

Alat OpenCart sa zadanim postavkama ne nudi nikakav web servis, već je potrebno kupiti željeni dodatak za omogućavanje web servisa iz njihove trgovine, što zahtijeva dodatne troškove, dok PrestaShop nudi vrlo skromni web servis – primjerice, za dohvat svih potrebnih informacija o proizvodu potrebno je slati desetak različitih zahtjeva prema krajnjoj točki alata,

što je izrazito neefikasno. Alati su dobra rješenja ako su korisnicima potrebna gotova rješenja koje nude korisničko sučelje za kupce i korisničko sučelje za administraciju proizvoda koje je strukturirano kao CMS e-trgovine.

OpenCart nudi fleksibilnije rješenje zato što nudi više postavki za definiranje proizvoda i njegovog tipa koje korisnici po želji mogu definirati i strukturirati. Nadalje, OpenCart je prilagođeniji korisnicima nego PrestaShop te je jednostavniji za korištenje, što je veliki plus ako su korisnici novi u industriji e-trgovine. S druge strane, PrestaShop nudi bogatiju trgovinu s dodacima pomoću kojih se mogu kreirati željeni CMS-ovi e-trgovine. Što se tiče mogućnosti plaćanja kupaca, OpenCart nudi veći izbor opcija načina plaćanja. OpenCart performansno malo nadmašuje PrestaShop, no ova varijabla se teško može procijeniti, zato što na nju utječu mnogi drugi kriteriji poput: broja objavljenih proizvoda, ukupnog broja posjetitelja, tvrtke za web hosting te cjelokupnog dizajna stranice. PrestaShop alat korisnicima nudi bolju SEO optimizaciju proizvoda zato što korisnici detaljno mogu specificirati svaki metapodatak i njegov opis, pri čemu postoji opcija kreiranja vlastitog URL-a za svaki proizvod, što dodatno pomaže kod SEO optimizacije.

Tablica 11. Usporedba alata OpenCarta i PrestaShopa

Opće ocjene	**OpenCart**	**PrestaShop**
Zadovoljavanje zahtjeva	8.8	8.1
Jednostavnost korištenja	8.5	7.7
Jednostavnost postavljanja	8.8	7.9
Jednostavnost administracije	8.8	7.7
Kvaliteta podrške	6.8	7.4
Je li proizvod bio dobar partner u poslovanju?	8.1	8.2
Smjer proizvoda (% pozitivno)	7.5	7.5
Administracija alata		
Izvještavanje	8.4	8.5
Upravljanje košaricama	9.3	8.3
Prikupljanje podataka o otpremi	9	8.2
Integracije		
Integracija Payment Gateway-a	8.9	8.8
Integracija platforme za e-trgovinu	8.4	9
Integracija e-pošte	8.3	8
Upravljanje košaricama		
Višestruka košarica	9	8.3
Proizvodi	9.3	8.9
Predlošci	8.6	8.3
Mogućnost ugrađivanja	8.7	8.6

(Izvor: G2 [30])

3.3. Ostale vrste CMS-a

Tradicionalni sustav za upravljanje sadržajem je klasičan pristup upravljanju sadržajem. Korisnicima tradicionalnog CMS sustava pruža se kompletan skup alata za kreiranje, uređivanje i objavljivanje sadržaja na internetu. Takvi sustavi pružaju predefinirane predloške za olakšanu izradu web stranica. Pružaju i sigurnosni postupak za pretpregled kreirane web stranice prije objavljivanja stranice na internet. Korisnici ni ne moraju znati programirati, već web stranice kreiraju pomoću spomenutih funkcionalnosti i samo popunjavaju web stranicu željenim sadržajem. Primjeri popularnijih tradicionalnih CMS alata su: WordPress, Joomla, Drupal i Magento.

Razdvojeni CMS alati su slični kao i CMS bez grafičkog sučelja, ali korisnicima pružaju veću kontrolu na prezentacijskom sloju. Naravno, kod razdvojenog CMS su front-end i back-end razdvojeni, a korisnici ovakvog CMS sustava moraju samo odabrati željene okvire i tehnologije koji će se koristiti u implementaciji aplikacije. Okviri i tehnologije koji se koriste moraju se implementirati tako da budu kompatibilni s razdvojenim CMS sustavom. Razdvojeni CMS sustav koristi se kada je korisnicima potrebna veća kontrola nad dizajnom web stranica i korisničkim iskustvima. Neki od popularnijih alata koji se mogu naći na tržištu razdvojenih CMS sustava su Strapi, Contentful i Kentico Kontent.

CMS alati za poduzeća najčešće se koriste kod velikih tvrtki koje zahtijevaju mnogobrojne i kompleksnije funkcionalnosti te veliku skalabilnost. Dodatne funkcionalnosti koje pružaju CMS sustavi za poduzeća su personalizacija samoga alata, upravljanje sustavom s više različitih web mjesta (eng. *multi-site management*), kao i mogućnost korištenja određenih procesa automatizacije tijeka rada. Najveći primjer ovakvog CMS sustava je Adobe Experience Manager (AEM).

Posljednja vrsta CMS sustava su komponentni CMS sustavi. Komponentni CMS sustav je sličan tradicionalnim CMS sustavima, uz tu razliku da se korisnicima pružaju predefinirane komponente ili pojedini blokovi web stranica pomoću kojih korisnici mogu graditi željenu web stranicu, baš kao da grade kuću ili skulpturu od lego kockica. Korisnici ne moraju znati programirati, već samo kombiniraju različite komponente i blokove, što dovodi do jednostavnog stvaranja dosljednog, responzivnog i vizualno privlačnog sadržaja. Komponentni CMS sustavi koriste se kada je potrebno brzo i učinkovito kreirati sadržaj te istim upravljati bez ikakve brige o HTML kodu, gdje korisnici do određene mjere imaju kontrolu nad prezentacijskim slojem. Neki od popularnijih primjera komponentnih CMS sustava bili bi: Contentstack, Kentico Kontent i Agility CMS.

4. Način dohvaćanja sadržaja iz headless CMS-a i njegovo korištenje

U posljednje vrijeme, sve veću i veću popularnost dobivaju sustavi za upravljanje sadržajem bez grafičkog sučelja, to jest CMS rješenja koja nude neku vrstu web servisa – bilo pomoću klasičnog RESTful API servisa ili nešto modernijom pristupom – upitnim jezikom za API-je (GraphQL). Odabir odgovarajućeg načina dohvata podataka ovisi o tvrtki te njenim potrebama i specifikacijama – oba načina imaju svoje prednosti i nedostatke.

Primjerice, pomoću GraphQL-a moguće je precizno definirati koji će se podaci dohvaćati od backend servera, što smanjuje mogućnost premalenog (eng. *underfetching*) ili preopširnog (eng. *overfetching*) dohvaćanja podataka. Nadalje, GraphQL nudi mogućnost dohvaćanja svih potrebnih resursa i informacija pomoću jednog zahtjeva.

REST API korisnicima omogućuje jednostavno rukovanje krajnjim točkama, pošto korisnici moraju točno definirati koju metodu koju žele obaviti na back-end serveru. Također, razvojnim inženjerima omogućuje kreiranje specifičnih točki sa specifičnim resursima, što pridonosi samoj optimizaciji servera.

4.1. RESTful API

RESTful API (eng. REpresentational State Transfer Application Programming Interface, REST API / RESTful API) je koncept koji se koristi kod razvoja web aplikacija, točnije kod kreiranja web servisa koji služe za komunikaciju između korisničkih aplikacija i servera, gdje se koriste standardni HTTP zahtjevi i željene metode za dohvaćanje ili ažuriranje podataka na konkretnim serverima.

Podaci i funkcionalnosti kod REST arhitekture smatraju se resursima prema dokumentaciji programskog jezika Java [31], a podacima se pristupa pomoću veze na webu, to jest pomoću *Uniform Resource Identifiera* (URI). Klijent i poslužitelj koriste standardna sučelja i protokole za razmjenu podataka. Dodatno, prema dokumentaciji, REST servisi koriste sljedeće principe, koji osiguravaju da aplikacija bude jednostavna, lagana i brza, su:

- identifikacija resursa putem URI-ja – svaki resurs dohvaća se pomoću jedinstvenog URI-ja;
- uniformno sučelje – definira se željena akcija nad željenim resursom kod svakog poziva, točnije, definirane su HTTP metode kao što su GET (za dohvaćanje

podataka), POST (za kreiranje podataka), PUT (za ažuriranje postojećih podataka ili kreiranje novih podataka) i DELETE (za uklanjanje postojećih podataka);

- samoopisne poruke – koriste se kod odgovora od strane servera. Poruke je potrebno strukturirati na način kako bi korisnička strana znala protumačiti i iskoristiti odgovor koji dobije od servera, točnije odgovor je potrebno specificirati odgovarajućim HTTP statusnim kodom (primjerice, kod 200 označava da je zahtjev uspješno izvršen, kod 404 označava da stranica ne postoji, itd.) i odgovarajućom porukom koja je najčešće JSON ili XML formatu.

4.2. GraphQL API

GraphQL je upitni jezik za API, koji je 2015. godine razvijen od strane Facebooka. Pomoću GraphQL-a je moguće precizno specificirati željene podatke koje je potrebno dohvatiti sa servera. GraphQL omogućava jednostavno, efikasno i fleksibilno rješenje za dohvaćanje ili ažuriranje željenih podataka na serveru. U službenoj dokumentaciji GraphQL-a [32] navedene su njegove osnovne karakteristike:

- usmjerenost prema proizvodu;
- hijerarhijska arhitektura – podaci se stvaraju i manipuliraju pomoću pogleda koji se temelji na hijerarhiji;
- strogo tipiziranje – GraphQL omogućuje željeno definiranje kompleksnih tipova podataka i njihovih relacija, što dovodi do visoke fleksibilnosti web servisa;
- odgovori specificirani od strane klijenta – klijenti dobivaju samo podatke koje su zatražili i ništa više, a smiju koristiti samo ono što im je i dodijeljeno;
- introspektivnost (omogućeno zahvaljujući GraphQL-u) – klijentima i njihovim alatima pruža se mogućnost dohvaćanja dodatnih informacija o shemi podataka dostupnih na serveru.

4.3. Korišteni okviri

CMS sustavi bez grafičkog sučelja, bez obzira koriste li se na tradicionalni način (blogovi) ili za e-trgovine, pružaju veliku fleksibilnost, budući da se sav dizajn i prikaz sadržaja prepušta razvojnim inženjerima i korisnicima sustava. Korisnici sustava željeni dizajn mogu kreirati u preferiranom programskom okviru, dok se cjelokupni sadržaj koji se treba prezentirati jednostavno može dohvatiti i prikazati pomoću jednog od prethodno spomenutih načina.

U posljednje vrijeme raste popularnost web aplikacija na jednoj stranici (eng. *Single Page Application* – SPA), gdje se koriste razvojni okviri bazirani na skriptnom jeziku JavaScript,

poput Reacta, Vue-a ili Angulara. Jedinstveni, najbolji razvojni okvir ne postoji, već ovisi o raznim kriterijima poput: kompleksnosti projekta, iskustva razvojnog tima, potrebe za skalabilnošću, brzini razvoja te performansama aplikacije. Svojstva koja su prema autoru Ericu [33] prednosti navedenih razvojnih okvira navedena su u tablici 12.

Tablica 12. Usporedba razvojnih okvira Angular, React i Vue

	Angular	React	Vue
Visoko uvažavanje TypeScript-a	X		
Naglašavanje smjernica i strukture kroz projekte	X		
Dolazi iz pozadine objektno orijentiranog programiranja	X		
Velika važnost fleksibilnosti		X	X
Veliki broj primjena	X	X	X
Lagani početni proces učenja			X
Naglasak na korištenju najnovijih, najpopularnijih tehnologija			X
Veliki ekosustav		X	
Odvajanje problema u jednu datoteku			X
Dizajneri moraju raditi s HTML kodom	X		X
Snažan fokus na korištenje JavaScripta		X	

(Izvor: Wohlgethan [33])

5. Razvoj aplikacije „Svijet u oblacima“

Web aplikacija „Svijet u oblacima“ pruža najnovije vijesti iz svijeta tehnologije, točnije pruža vijesti sve što je povezano sa: računalima, laptopima, mobilnim uređajima i *gaming* konzolama. Web aplikacija se sastoji od dva glavna dijela, a to su blog i web trgovina. Prvi dio se odnosi na web blog, gdje se prikazuju svi članci web stranice. Blog dio je kreirani pomoću komercijalnog CMS alata bez grafičkog sučelja Contentfula. Menedžment svih sadržaja web bloga se odvija u web pregledniku pomoću Contentfulovog alata. Dodavanje novog sadržaja, upravljanje i objavljivanje postojećih sadržaja, točnije blog članci se odvija pomoću Contentfula, dok prikaz željenih sadržaja se dohvaća pomoću REST API-a i dinamično generira na front-endu. Drugi dio aplikacije se odnosi na web trgovinu i upravljanje samih proizvoda web tržnice koji je realizirani pomoću komercijalnog alata Commercetoolsa. Commercetools pruža CMS e-trgovine, gdje se upravljaju sa svim proizvodima, kupcima trgovine, metodama plaćanja i valutama, metodama dostave i ostalim funkcionalnostima sustava. Objavljeni proizvodi se također dohvaćaju pomoću REST API-a i dinamično generiraju na front-endu web stranici.

Cijela web aplikacija je *responzivna* i dizajn će se prilagoditi odgovarajućoj veličini ekrana, što je realizirano uz pomoć Tailwinda. Pomoću Tailwinda konfigurirano je pet moguća prikaza dizajna za različite dimenzije korisničkog ekrana, točnije: za najmanje ekrane od 240 piksela nadalje je konfiguracija ekrana *extra small* (xs); od 480 piksela nadalje je konfiguracija *small* (sm); od 768 piksela nadalje je konfiguracija *medium* (md); od 976 piksela nadalje je konfiguracija *large* (lg); od 1440 piksela i nadalje je konfiguracija *ekstra large* (xl).

Početna stranica sadrži komponente za: korisničku navigaciju na web stranici; prijavu / registraciju korisnika; odabir željenog jezika; tražilicu za članke; početni članak, koji je ujedno i najnoviji članak; sekciju prikaza svih članaka, gdje korisnici mogu dodatno filtrirati članke po odabranim kategorijama članka i otvoriti svaki članak, kako bi vidjeli detalje o članku; sekciju prikaza najpopularnijih / najposjećenijih članaka; sekciju najbolje ocjenjenih članaka na temelju ostalih korisnika; sekciju preporuke članaka korisnicima, pod uvjetom da je trenutni korisnik prijavljeni i da je prijavljeni korisnik ocijenio bilo koji članak; podnožje (eng. *footer*) web stranice. Naravno navigacijski panel i footer je na svim stranicama isti i prisutan. Sljedeća stranica je stranica „O nama“, gdje se korisnicima prikazuje misija i vizija web stranice. Zatim dolazi stranica e-trgovine, gdje se korisnicima daje na izlog popis svih proizvoda koji su objavljeni u Commercetools platformi. Korisnici mogu otvoriti stranicu o detaljima proizvoda, gdje se prikazuju sve varijacije proizvoda, specifikacije o proizvodima, komentari i recenzije o proizvodima.

Neprijavljeni korisnici mogu samo: pregledavati sve članke, filtrirati članke i gledati detalje o članku i njegove komentare; pregledavati najpopularnije i najbolje ocjenjene članke; pretraživati članke pomoću tražilice; pregledavati e-trgovinu proizvoda, te otvarati ponuđene proizvode kako bi dobili detalje, specifikacije i recenzije od proizvoda; prijaviti ili registrirati se u sustav pomoću običnog računa ili putem Google računa.

Registrirani korisnici mogu sve što i neprijavljeni korisnici, te dodatno mogu: komentirati i davati ocjenu svakom članku i naravno mogu obrisati kreirani komentar; ako je prijavljeni korisnik komentirao bilo koji članak, dodatno se prikaže sekcija o preporuci članka koja će biti jedinstvena za svakog korisnika na početnoj stranici web aplikacije; kupovati željene proizvode; objavljivati recenzije o proizvodima; pregledavati košaricu i upravljati proizvodima u košarici (promjena količine ili potpuno ukloniti iz košarice); mijenjati korisnički profil (ime, prezime, korisničko ime, podaci o dostavi kupljenih proizvoda); pregledavati povijest kupljenih proizvoda; se odjaviti.

Moderatori mogu sve što registrirani korisnici mogu i dodatno mogu pregledavati statistiku posječenosti njihovih članaka. Stranica za statistiku članka prikazuje moderatoru: popis svih njegovih članaka; popularnost članka; prosječnu ocjenu; ukupan broj komentara za svaki članak; mogućnost pregleda komentara za svaki članak.

Administrativni korisnici mogu sve što i registrirani korisnici i dodatno mogu konfigurirati web stranicu, gdje mogu mijenjati prikaz sekcija sadržaja za popis: svih članka; najpopularnijih članka; najbolje ocijenjenih članka; preporuke članka korisnicima; pretraženih članka iz tražilice. Dodatno mogu dodavati nove moderatore web stranice i upravljati narudžbama (gdje mogu mijenjati status narudžbe da je narudžba poslana ili dostavljena)

5.1. Korištene tehnologije i radno okruženje

Cijeli dizajn i svi prikazi korisničkog sučelja su kreirani pomoću JavaScrip razvojnog okvira Reacta i stilskog CSS okvira Tailwind CSS. Pomoću okvira Reacta kreirane su sve potrebne stranice, komponente stranice i cijela logika svih stranica web aplikacije. Dodatno pomoću React-a su implementirane vlastite kuke za dohvaćanje svih sadržaja i proizvoda, koje vrše REST API zahtjeve prema krajnjim točkama od Contentfula, Commercetoolsa i back-end servera.

Back-end server je kreirani pomoću Node.js-a i koristi okvir Express kako bi se kreirale krajnje točke. Back-end server se koristi kako bi se provjerila autentifikacija korisnika i dodatno vrši komunikaciju sa Firebase bazom podataka.

Sustav za upravljanje sadržajem bez grafičkog sučelja Contentful služi za upravljanje svim člancima web bloga. Novi članci se kreiraju pomoću Contentfula, dok postojeći članci se mogu dodatno uređivati i objavljivati. Objavljeni članci se dohvaćaju na frontu u Reactu i dinamično se generiraju na ekranima korisnicima.

Isto tako vrijedi i za CMS bez grafičkog sučelja e-trgovine Commercetools služi za upravljanje svim proizvodima, metodama dostavljača, narudžbama, metodama plaćanja i ostalim funkcionalnosti koje nudi alat. Pomoću alata se kreiraju željeni tipovi proizvoda i sami stvarni proizvodi, te ostale željene postavke sustava, koje se onda potom mogu dohvaćati pomoću REST API zahtjeva i dinamično generirati na frontu web aplikaciji.

Baza podataka Firebase služi za pohranu svih ostalih informacija o korisnicima i podacima web stranice, koje su bile potrebne kod realizacije projekta. Podaci koji se pohranjuju u bazu podataka su: dodatne korisničke informacije (ime, prezime, adresa, grad, telefon itd.); komentari članaka; postavke web stranice „Svijet u oblacima“; statistika članaka namijenjeno za moderatore stranice; podaci o narudžbama.

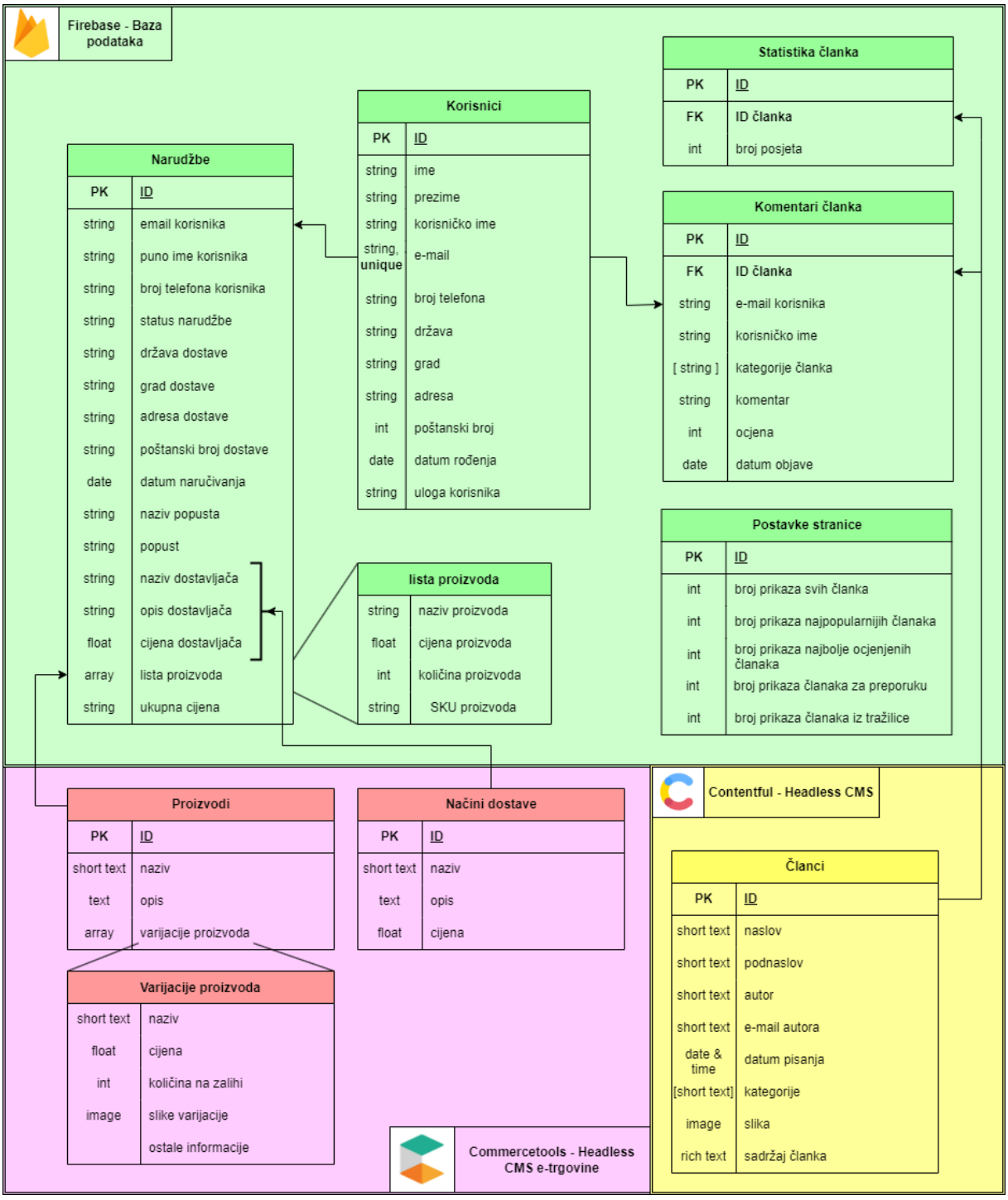
Cijeli programski kod za front-end dio aplikacije je moguće pronaći u priloženoj datoteci diplomskog rada, pod nazivom „Front-end dio aplikacije“ ili na GitHub stranici autora. Cijeli programski dio koda za back-end server se također nalazi u priloženoj datoteci, pod nazivom „Back-end dio aplikacije“ ili na GitHub stranici autora. Napomena, projekt ne funkcionira bez varijabli za okruženje (eng. *Environment Variables*) u kojoj su pohranjeni privatni ključevi, te iz tog razloga ne nalaze se u priloženim datotekama. Linkovi za GitHub repozitorije se nalaze u popisu za priloge.

5.2. ERA model i dijagram

Slika 24. prikazuje ERA model, koji je kreirani na temelju kombinacije Firebase baze podataka (označeno sa zelenom bojom), Contentfula (označeno sa žutom bojom) i Commercetoolsa (označenom sa svijetlo ljubičastom bojom). Na slici 24. u vezi alata Commercetoolsa nalaze se samo najvažniji dijelovi za proizvode, radi jednostavnosti prikaza na slici. U stvarnosti Commercetools sadrži mnogo kompleksniju i opširniju bazu podataka.

Firebase baza podataka nije klasična baza podataka, gdje su svi pojedini podaci striktno definirani, nego je polustrukturirana baza podataka. Podaci se mogu pohranjivati nekonzistentno i bilo kako, no radi jednostavnosti na frontu u Reactovom projektu svi podaci će biti konzistentni i striktno definirani. Tablica postavke stranice služi za pohranu prikaza sadržaja za određenu paginaciju na web stranici, točnije čuva količinu članaka koji će se prikazivati na jednoj stranici paginacije. Navedena tablica nije direktno povezana sa niti jednom

drugom tablicom, nego se koristi samo prilikom prvog posjećivanja web stranice. Tablice statistika i komentari članka koriste identifikacijske brojeve i ostale informacije koje se dobiju na temelju otvorenog članka na frontu iz alata Contentfula i dodatno koriste informacije iz tablice Korisnici, koji se pohranjuju u sesiju nakon prijave korisnika.



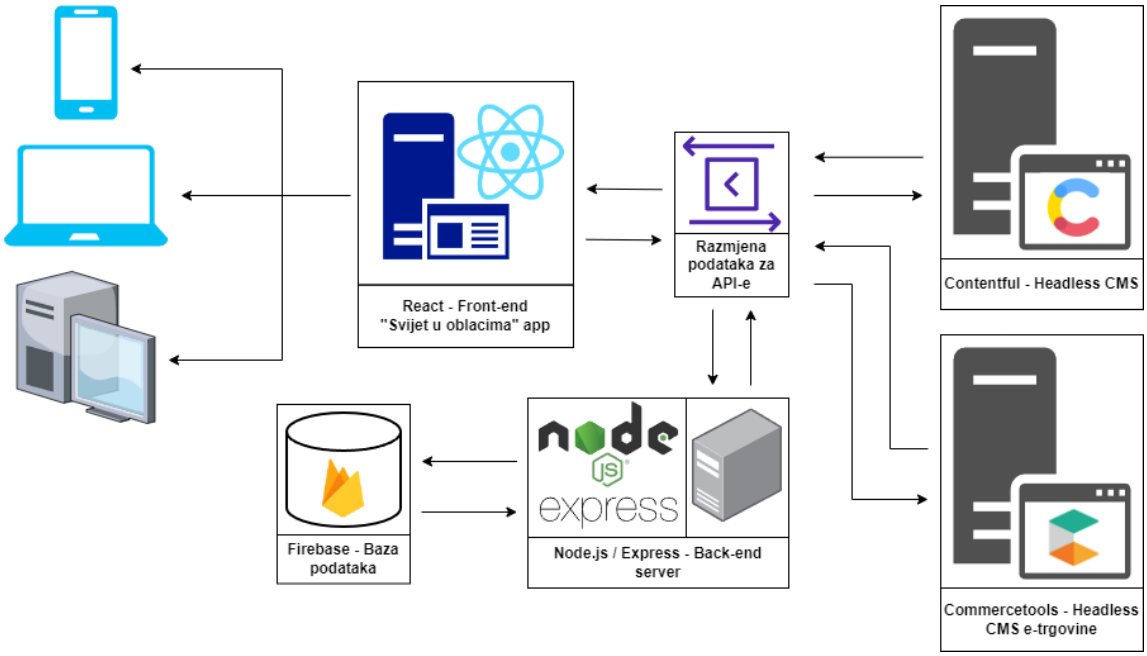
Slika 24. ERA model Firebasea, Contentfula i Commercetoolsa [autorski rad]

Tablica statistika članaka se koristi kako bi se pratila posječenost pojedinih članaka na web stranici, a tablica komentari članaka sadrži sve komentare prijavljenih korisnika. Kod realizacije za prijavu / registraciju korisnika koristi se Firebase autentifikacija, gdje se pohranjuje samo e-mail i lozinka korisnika, što je nažalost premalo informacija o korisniku. Radi tog razloga kreirana je tablica korisnici, koja sadrži sve potrebne informacije, kao što su ime, prezime, korisničko ime, podaci o dostavi i ujedno sadrži e-mail. E-mail kod korisnika mora biti jedinstveni, pošto on služi kao vanjski ključ pomoću kojeg se povezuje sa Firebase autentifikacijom, gdje korisnici dobivaju autentifikaciji token, koji je potreban kod back-end servera, radi sigurnosti baze podataka. Posljednja tablica kod Firebase baze je tablica narudžbe, gdje su pohranjeni svi potrebni podaci o naručenim proizvodima od korisnika. Tablica narudžbe sadrži podatke iz sesije od korisnika i ostale informacije koje se dobiju putem API-a iz Commercetoolsa na frontu. Navedena tablica dodatno sadrži atribut lista proizvoda koja je tipa polja (eng. *array*). Atribut lista proizvoda sadrži zasebno atribute pomoću kojih su pohranjene informacije o naručenim proizvodima, točnije naziv, cijenu, količinu i SKU proizvoda.

Tablica koja se nalazi na Contentfulu (označeno sa žutom bojom na slici 24.) reprezentira model sadržaja koji se koristi za pohranu i upravljanje svih članaka web stranice. Dvije stvari koje je potrebno istaknuti kod navedene tablice. Prva stvar je atribut kategorije, gdje se pohranjuje polje teksta, točnije nazivi pred-definiranih kategorija. Druga stvar je atribut sadržaj članka, koji je tipa bogati tekst. Pomoću bogatog teksta editori sadržaja mogu dodatno uređivati stilove teksta (od H1 do H6, podebljani, kurziv itd.), te dodatno dodavati slike, videozapise, tablice, citate, linkove i liste tekstova. Slika 25. prikazuje povezanost fronta Reactove aplikacije sa back-end serverom, Commercetoolsom i Contentfulom. Pomoću Reacta i Tailwind-a CSS se dohvaćeni sadržaji od CMS sustava i back-end servera dinamično generiraju na ekranima IoT uređajima.

Dodatno pomoću Tailwinda je definirani odgovarajući dizajn koji je prilagođeni za bilo koji veličinu ekrana IoT uređaja. Sadržaji koji su potrebni na frontu se dohvaćaju pomoću API zahtjeva na odgovarajuću krajnju točku, bilo to prema back-end serveru ili CMS sustavima Contentfulu ili Commercetoolsu.

Prilikom slanja zahtjeva prema određenoj krajnjoj točci je potrebno na frontu kratko vrijeme čekati odgovor od servera, gdje se korisnicima za to vrijeme čekanja prikazuje animacija učitavanje sadržaja (četvrtina kružnice koja se konstanto vrti u krug). Uspješni odgovori se obrađuju i korisniku se prikazuje dohvaćeni sadržaj, dok u slučaju pogrešaka (eng. *error*) se na odgovarajući način prikazuje poruka pogreške. Ponekada je potrebno prvo slati zahtjeve prema back-end servera i potom dohvaćene podatke od back-end servera se dalje dohvaćaju sadržaji iz CMS sustava koji se potom prikazuju korisnicima.



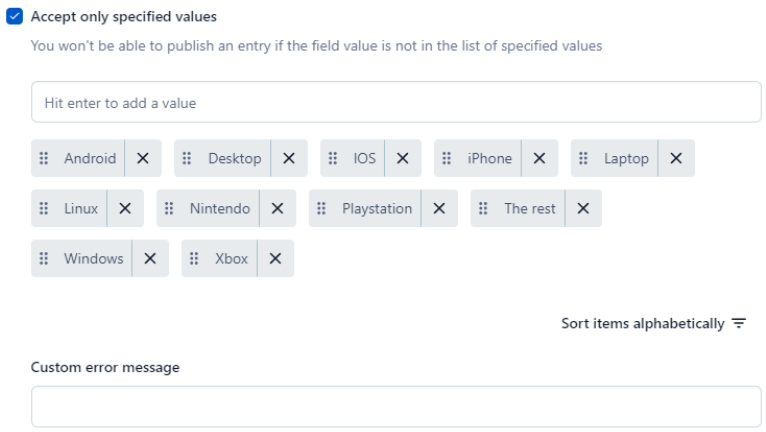
Slika 25. Komunikacija između front-enda i back-end servera [autorski rad]

5.3. Contentful

Contentful će se u web aplikaciji „Svijet u oblacima“ koristiti za upravljanjem cijelim sadržajem koji je povezani za blog dio aplikacije. Pomoću Contentfula će se kreirati novi članci, te uređivati i objavljivati postojeći članci. Objavljeni sadržaji, to jest članci će se pomoću API zahtjeva dohvaćati i dinamično generirati sadržaj na frontu u Reactovoj web aplikaciji. Navedeni alat je odabrani iz više razloga, prvi razlog je naravno što pruža korisnicima besplatno rješenje za sustave CMS-a bez grafičkog sučelja. Drugi razlog je to što pruža mnogobrojne funkcionalnosti s kojima je lagano izgraditi željene sustave CMS-a. Treći razlog je to što je izrazito dobro prilagođeni korisnicima, te njegovo korištenje izrazito lagano, logično i razumljivo. Posljednji razlog odabira navedenog sustava je to što je izrazito popularno rješenje i time je prilikom izbijanja bilo kakvih poteškoća (problemi, *bug*-ovi ili nejasnoće) brzo riješeno pomoću njihove službene dokumentacije ili pomoću različitih foruma Contentfula ili zajednice ljudi.

Prvi korak kod realizacije CMS-a bez grafičkog sučelja pomoću Contentfula je kreiranje korisničkog računa na njihovoj službenoj stranici. Nakon uspješne registracije u korisnički račun potrebno je kreirati model sadržaja, gdje se definiraju sva potrebna polja, te njihove tipove, ograničenja i ostale specifikacije. Prvo polje koje se kreira je polje naslova članka, gdje se definira da je polje kratkog teksta (eng. *short text*) i da je obavezno polje, zatim se na isti način kreira polje za podnaslov, autor članka i e-mail autora članka (potreban kako bi se znalo

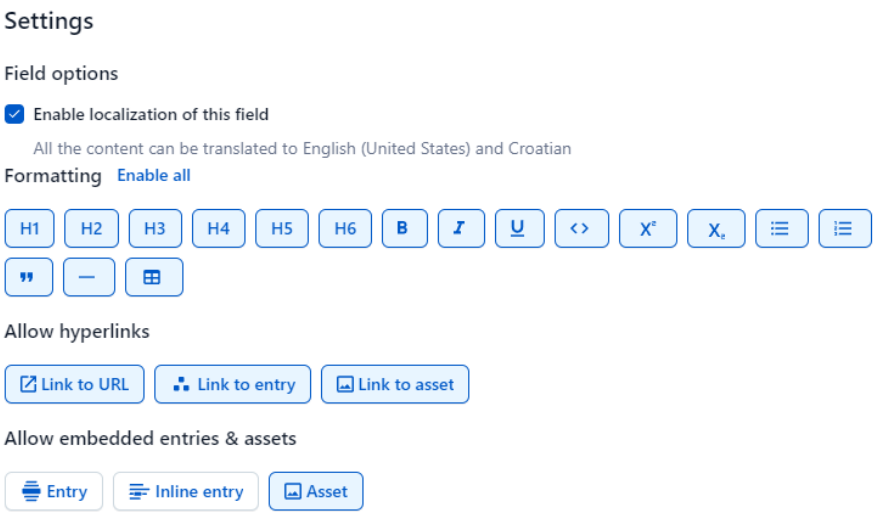
tko je moderator na web aplikaciji „Svijet u oblacima"). Zatim sljedeće polje je datum kreiranja članka koji je također obavezno polje i tipa datuma i vremena (eng. *date & time*). Sljedeće polje je polje kategorije, polje mora biti obavezno i dodatno se definira odabir iz pred-definirane liste kratkoga teksta. Navedenom polju se postavi opcija za odabir samo navedenih vrijednosti, gdje se dodaju nazivi određenih kategorija. Točnije nazivi kategorija su: Desktop; IOS; iPhone; Android; Windows; Linux; Laptop; Playstation; Nintendo; Xbox; The rest. Dodatno se mora još postaviti da je prikaz kategorije u obliku potvrdnog okvira (eng. checkbox), kako bi editori sadržaja bili u mogućnosti odabira jedne ili više kategorije određenog članka. Slika 26. prikazuje kreiranje navedenog polja za kategorije.



Slika 26. Prikaz kreiranja pred-definirane liste za odabir kategorije članka [autorski rad]

Sljedeće polje koje je također obavezno je polje slika članka koja se prikazuje kao glavna slika kod komponente za članka na početnoj stranici web aplikacije. Posljednje polje koje se dodaje na Contentfulovom modelu sadržaja je sami sadržaj članka koji će se prikazivati na stranici o detaljima pojedinog članka. Polje sadržaja članka je tipa bogatog teksta, što je bilo već ranije objašnjeno i omogućava editorima sadržaja dodatno oblikovanje teksta. Slika 27. prikazuje mogućnosti koje su dozvoljene za uređivanje teksta (na slici su označene sa svijetlo plavom bojom).

Dodatno kod svakog polje je moguće definirati: validaciju unesenih vrijednosti; limit broja dodanih medijskih zapisa; mogućnost limitiranog dodavanja određenog tipa medijskog zapisa; odabir izgleda prikaza kod ispune modela sadržaja (jedna linija za ispunu, URL, padajući izbornik, radio gumbi itd.); pred-definirane vrijednosti; opis koji služi za objašnjenje kod ispune polja za editore sadržaja. Primjer kreiranja modela sadržaja nalazi se na slici 8. u prethodnom dijelu rada.

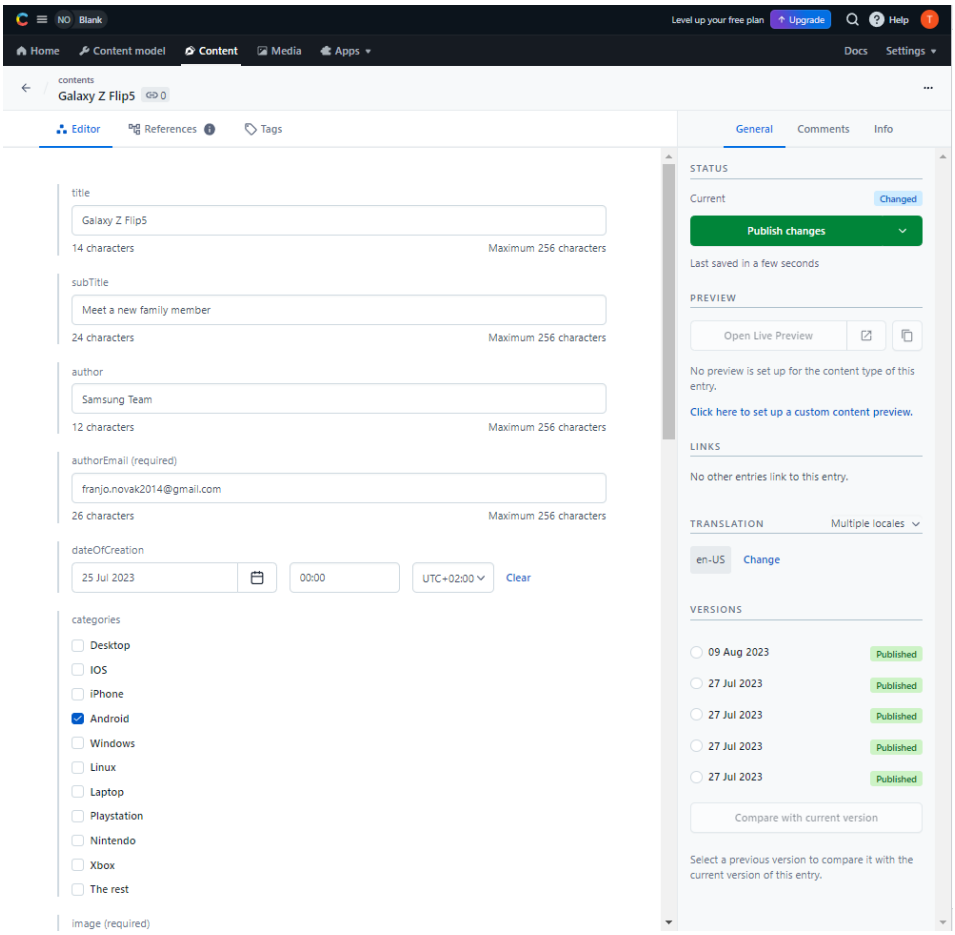


Slika 27. Prikaz mogućih dozvola za oblikovanje bogatog teksta [autorski rad]

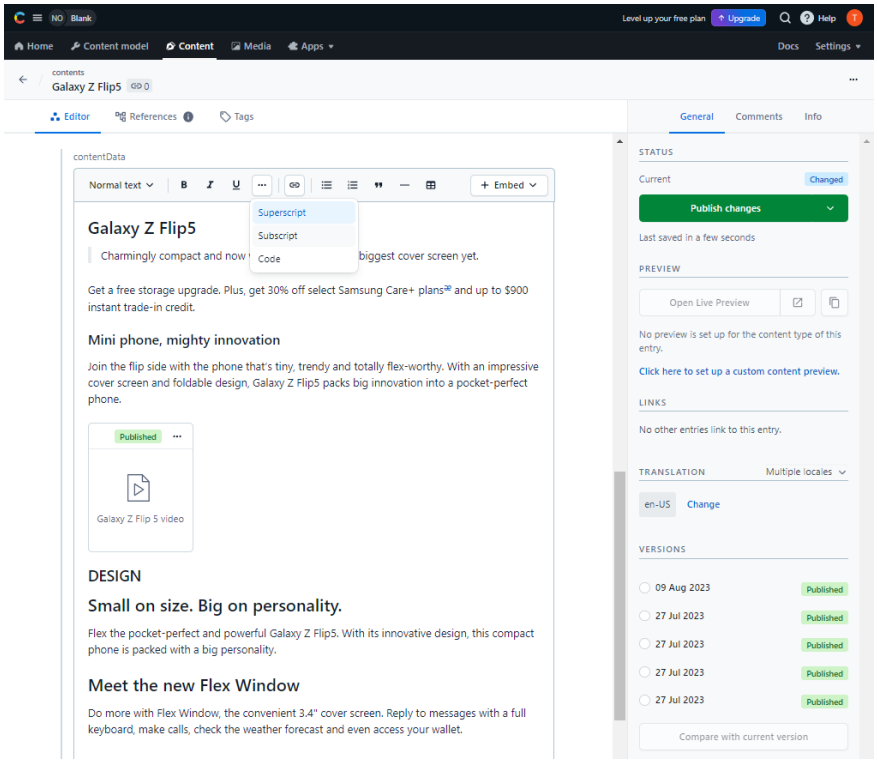
5.3.1. Kreiranje stvarnog sadržaja u Contentfulu

Kod kreiranja novog sadržaja otvara se stranica sa sadržajima na Contentfulu, gdje su prikazani svi dodani sadržaji u CMS sustav. Korisnici alata mogu filtrirati sadržaj po tipu modela sadržaja ili po statusu sadržaja (u izradi, objavljeni ili arhivirani). Za dodavanje novog sadržaja klikne se na gumb za dodavanje novog zapisa i odabere se željeni model sadržaja. Potom editori sadržaja mogu ispunjavati sva prethodno definirana polja modela sadržaja. Slika 28. prikazuje postupa dodavanja novog sadržaja / članka u CMS sustav.

Editorima sadržaja kod kreiranja bogatog teksta za sadržaj članka se omogućuje raznolike mogućnosti za dizajniranje teksta koji će se koristiti kod stranice o detaljima članka. Editori sadržaja mogu uređivati, te vizualno vidjeti uređeni tekst. Oblikovani tekst koji se dodaje u sustav neće identično izgledati kako ga vide editori kod unosa sadržaja, nego svako oblikovanje teksta će se posebno definirati i oblikovati pomoću Tailwinda CSS na frontu u React aplikaciji. Editori sadržaja kod pisanja bogatog teksta je omogućeno sljedeće: dodavanje normalnog teksta; dodavanje naslova veličine od H1 do H6; podebljani tekst; tekst u kurzivu; podcrtani tekst; dodavanje broja na eksponentu i donji indeks; odsječak za programski dio koda; dodavanje linkova; dodavanje sortirane i nesortirane liste; dodavanja citata; dodavanje horizontalne linije; dodavanje medijskog sadržaja. Slika 29. prikazuje primjer dodavanja bogatog sadržaja u alatu Contentfulu.



Slika 28. Postupak dodavanja novog sadržaja u CMS sustav [autorski rad]



Slika 29. Prikaz dodavanja bogatog teksta sadržaju [autorski rad]

5.3.2. Lokalizacija Contentfula

Contentful u besplatnoj verziji poslovnog paketa pruža klijentima nažalost samo dvije lokalizacije, pošto je cijela React aplikacija inicijalno napravljena da podržava tri zadana jezika (engleski, hrvatski i njemački). Pomoću lokalizacije moguće je postaviti da se sadržaji dohvaćaju na željenom jeziku. Inicijalni jezik je postavljeni na engleski i cijeli sadržaj će se dohvaćati na engleskom jeziku, ako sadržaj za koji se želi dohvatiti ne posjeduje prevedeni sadržaj ostalih jezika. Kod samog dodavanja sadržaja prevoditeljima sadržaja se pruža dva pogleda za prevođenje sadržaja. Prvi pogled je da se cijeli sadržaj dodaje posebno za prvi jezik posebno za drugi jezik. Drugi pogled je paralelna ispuna podataka sadržaja, gdje se za svako polje sadržaja dvostruko ispunjava, prvo na jednom jeziku, te potom na drugom jeziku.

Kako bi se omogućilo korištenje lokalizacije potrebno je u postavkama Contentfula dodati novi jezik koji će se koristiti za lokalizaciju. Kod dodavanja novog jezika, korisnici moraju prvo odabrati željeni jezik (kod zadanog projekta je odabrani hrvatski jezik), potom korisnici moraju odabrati rezervni jezik u slučaju da određeni sadržaj / članak ne posjeduje prevedeni sadržaj. Dodatno se omogućuje postavljanje postavki za slanje lokalizacije kod API zahtjeva, mogućnost uređivanja zapisa putem API-ja i mogućnost postavljanja da se može unašati prazna polja kod lokalizacije.

5.4. Back-end server

Bekend server je realizirani pomoću Node.js-a i njegovog okvira Express-a. Node.js je rješenje otvorenoga koda, temeljeno na *cross-platformsko* JavaScript *runtime* okruženju. Bekend server pruža REST API točke pomoću kojih se dohvaćaju potrebne informacije iz baze podataka Firebase. Back-end server je podijeljeni na dva dijela, prvi dio je kreirani s namjerom za neregistrirane korisnike, gdje se uglavnom smo dohvaćaju podaci, osim kod praćenje statistike preglednosti članka, gdje se zapisuje u bazu podataka. Drugi dio je namijenjeni za registrirane korisnike, gdje registrirani korisnici moraju posjedovati autorizacijski token, koji se dobije kod same prijave web stranice. Ovdje se radi o osjetljivim podacima korisnika ili se radi o ažuriranju podataka u Firebase bazi, te iz tog razloga je implementirati sigurnosni sistem sa Firebase tokenom.

Back-end server sadrži ukupno sedam općenitih URI putanja, gdje svaka putanja sadrži svoje zasebne krajnje točke sa određenim metodama zahtjeva. Što se tiče datoteka i direktorija servera je sljedeća u korijenskom direktoriju se nalazi: datoteka indeksa koja je zadužena za inicijalizaciju svih ostalih datoteka i naravno samo pokretanje servera; mapa sa usmjerivačima (eng. *routers*); mapa s dodacima (eng. *utils*). Mapa *routers* sadrži datoteke koje definiraju

potrebne krajnje točke za REST API, a mapa *utils* sadrži datoteku admin.js koja je zadužena za inicijalizaciju objekta za komunikaciju sa Firebase bazom podataka. Inicijalizira se jednom instanca objekta za komunikaciju sa bazom podataka, koji se potom koristi svugdje u back-end serveru.

Indeks datoteka sadrži dijelova programskog koda za: inicijalizaciju svih ostalih usmjerivača; postavljanje *Cross-Origin Resource Sharing* (CORS) pomoću kojega se omogućuje pristup određenim domenama, shemama i priključcima; inicijalizaciju baze podataka; postavljanje da se koristi JSON format zapisa podataka kod komunikacije sa back-end serverom. Sljedeći isječak programskog koda prikazuje funkciju pomoću koje se provjerava autentifikacija korisnika:

```
function checkAuth(req, res, next) {
  const { authorization } = req.headers;
  const authtoken = authorization && authorization.split(" ")[1];
  if (authtoken) {
    admin
      .auth()
      .verifyIdToken(authtoken)
      .then(() => {
        next();
      })
      .catch(() => {
        res.status(403).send("Unauthorized");
      });
  } else {
    res.status(404).send("Token not valid or not provided");
  }
}
```

Navedena funkcija najprije dohvaća token korisnika koji je zapakirani kao *string* podatak sa prefiksom *Bearer. Bearer* se koristi kod slanja tokena korisnika kako bi se točno moglo razlikovati od ostalih vrsta provjere autentičnosti. Varijabla admin se prethodno instancira i služi za komunikaciju sa Firebase bazom podataka. Ako je autentifikacijski token valjan vrši se funkcija željene krajnje točke, no ako je token nevažeći vraća se odgovora sa određenim statusom poruke i opisom poruke. Krajnje točke koje su postavljene prije provjere autentifikacije, nije potreban token. Ovdje se uglavnom dohvaćaju opće informacije koje su potrebne kod prikaza za neregistriranih korisnika na front-endu. Dok s druge strane krajnje točke koje se nalaze nakon provjere autentifikacije je potreban token, kako bi se izvršile funkcije zadanih krajnjih točaka. Ovdje je riječ o osjetljivim podacima korisnika ili je potrebno vršiti pojedina ažuriranja nad bazom podataka, gdje se prvo naravno mora provjeriti

autentifikacija korisnika. Sljedeći isječak koda prikazuje redoslijed učitavanja prethodno spomenutih krajnjih točaka i provjeru autentifikacije:

```
app.use(cors(corsOptions));
app.use(express.json());
app.use("/api", pageStatistics);
app.use("/api", postComments);
app.use("/api", pageSettings);
app.use(checkAuth);
app.use("/api", userData);
app.use("/api", postCommentUpload);
app.use("/api", pageSettingsUpdate);
app.use("/api", userOrder);
app.listen(5050, () => {
  console.log("Server is running on port 5050");
});
```

Prethodni isječak koda prikazuje kako krajnje točke za statistiku stranice, komentare članaka i postavke stranice nije potrebna autentifikacija, dok za ostale krajnje točke, točnije: podaci o korisniku, kreiranje komentara članaka, postavljanje postavki stranice i narudžbe korisnika je potrebna autentifikacija. Dodatno se može vidjeti da se kod svakog usmjerivača dodaje prefiks „/api“ na URI putanju. Nakon učitavanja svih krajnjih točaka pokreće se sami server koji je podignuti lokalno na portu 5050.

Datoteka admin,js kreirana je pomoću okvira firebase-admin i privatnog ključa projekta za bazu podataka. Datoteka prvo provjerava ako je već inicijalizirani objekt, te ako je objekt inicijalizirani vraća ga, ali ako nije onda kreira novi objekt pomoću privatnog ključa i URL-a na Firebase bazu podataka. Sljedeći isječak programskog koda prikazuje datoteku admin.js:

```
var admin = require("firebase-admin");
var serviceAccount = require("../../svijet-u-oblacima-firebase-
adminsdk.json");
if (!admin.apps.length) {
  admin.initializeApp({
    credential: admin.credential.cert(serviceAccount),
    databaseURL: "https://svijet-u-oblacima.firebaseio.com",
  });
}
module.exports = { admin };
```

5.4.1. Krajnje točke back-end servera

Prvi usmjerivač je postavke stranice – „pageSettings“, koji sadrži samo jednu metodu, točnije GET metodu za dohvaćanje informacije o postavkama stranice. Kod navedenog usmjerivača nije potrebna autentifikacija korisnika pošto su postavke stranice potrebne kod svih korisničkih uloga. Svi pojedini usmjerivači sadrže okvir Express kako bi se omogućile funkcije usmjerivača i varijabla objekta db, kreiran na temelju funkcije firestore() iz datoteke admin.js za uspostavljanje komunikacije sa Firebase bazom podataka. Realizacija kreiranja određenih metoda krajnjih točaka kreiraju se posebno GET, POST, PUT ili DELETE funkcije, kojima se definira URI putanja i njihova funkcionalnost. Sljedeći isječak programskog koda prikazuje način kreiranja pružanja web servisa za dohvaćanje postavki web stranice.

```
const express = require("express");
const router = express.Router();
router.get("/pageSettings", async (req, res) => {
  try {
    console.log("Getting page settings...");
    const pageSettings = await getPageSettings();
    if (pageSettings !== null) {
      res.status(200).json(pageSettings);
    } else {
      res.status(404).json({ message: "Page settings not found." });
    }
  } catch (error) {
    console.error("Error getting page settings:", error);
    res.status(500).json({ message: "Error getting page settings." });
  }
});
module.exports = router;
```

Prethodni isječak koda prikazuje realizaciju kreiranja web servisa za GET metodu dohvaćanja informacija o postavkama web aplikacije „Svijet u oblacima“ iz Firebase baze podataka. Prethodni isječak programskog koda prikazuje kako se kreiraju jednostavni web servisi. Prvo se definira varijabla *router* koji se kreira na temelju okvira Express-a i implementiraju se željene metode zahtjeva i njezina funkcionalnost. U prethodnom primjeru definirani je GET zahtjev za dohvaćanje postavki stranice, gdje se najprije definira URI putanje „/pageSettings“ i potom funkcionalnost krajnje točke. Potpuni URL putanja zadane krajnje točke izgleda sljedeće „http://localhost:5050/api/pageSettings“, radi prefiksa na URI dio putanja i root dio putanja servera. Funkcionalnost navedene krajnje točke je asinkrona funkcija (pošto se moraju čekati informacije iz Firebase baze podataka) u kojoj se poziva funkcija

„getPageSetting“. Ako se uspješno dohvate postavke stranice iz baze podataka šalje se odgovora sa statusom 200 i informacije o postavkama stranice u JSON formatu, dok u suprotnome ako nije bilo uspješno šalje se odgovor sa statusom 404 i pripadajućom porukom. Dodatno ako izbije nepoznata greška, šalje se odgovor sa statusom 500 i porukom greške. Na temelju prethodnog primjera kreirane su i ostale krajnje točke na sličan način, samo što je drugačiji URI dio putanja i funkcija za dohvaćanje podataka iz Firebase baze podataka. Pošto je prethodi dio programskoga koda sličan za ostale metode i krajnje točke se neću prikazivati u radu, nego će se objasniti najvažniji dijelovi i njezine funkcije za dohvaćanje podataka iz baze podataka. Na kraju svakog usmjerivača se naravno vraća kreirani usmjerivač sa kreiranim metodama web servisa. Sljedeći dio programskog koda prikazuje funkciju „getPageSettings“ za dohvaćanje podataka iz baze.

```
const { admin } = require("../util/admin");
const db = admin.firestore();
async function getPageSettings() {
  try {
    const pageSettingsRef = db.collection("pageSettings");
    const pageSettingsSnapshot = await pageSettingsRef.get();
    if (!pageSettingsSnapshot.empty) {
      const pageSettingsData = pageSettingsSnapshot.docs[0].data();
      if (pageSettingsData) {
        return pageSettingsData;
      }}
    return null;
  } catch (error) {
    throw new Error(error);
}}
```

Prethodna dio programskog koda prikazuje asinkronu funkciju za dohvaćanje postavki web stranice iz baze. Funkcija koristi prethodno spomenuti objekt db pomoću kojeg se definira iz koje tablice je potrebno dohvatiti podatke (db.collection("pageSettings")). Pošto navedena tablica u bazi posjeduje samo jedan zapis u kojoj su definirane postavke web stranice koristi se samo prva vrijednost polja. Ako dohvaćena vrijednost nije prazna vraćaju se postavke stranice, a ako je prazna vraća se *null* vrijednost.

Sljedeći usmjerivač je „pageSettingsUpdate“ koji je sličan kao prethodni usmjerivač, samo što on zahtjeva provjeru autentifikacije korisnika, te iz tog razloga je odvojeni od prethodnog usmjerivača. Sadrži dvije PUT metode, pomoću kojih se rade ažuriranja tablica u bazi podataka. Prva PUT metoda sadrži URI putanje „/pageSettings“ i funkciju pomoću koje se ažurira stanje tablice postavke stranice na nove vrijednosti koje su bile poslane u tijelu

zahtjeva web servisa. Druga PUT metoda sadrži URI putanje „/newModerator“ i funkciju pomoću koje se ažurira uloga korisnika u moderatora. Navedena krajnja točka prima e-mail korisnika za kojega je potrebno ažurirati ulogu u tijelu zahtjeva. Sljedeći isječak programskog koda prikazuje funkciju za ažuriranje stanje uloge korisnika na temelju proslijeđenog e-maila.

```
async function updateUsersRole(email) {
  try {
    const usersRef = db.collection("users");
    const userSnapshot = await usersRef.where("email", "==",
email).get();
    if (!userSnapshot.empty) {
      const userData = userSnapshot.docs[0].data();
      const userId = userSnapshot.docs[0].id;

      if (userData) {
        await usersRef.doc(userId).update({ role: "moderator" });
      }} else {
      throw new Error("User not found");
    }} catch (error) {
    throw new Error(error);
  }}
```

Prethodna navedena funkcija najprije dohvaća korisnika na temelju proslijeđenog e-maila, te ako se pronađe traženi korisnik njegova korisnička uloga se ažurira u moderatora. Ako se ne pronađe traženi korisnik u bazi podataka onda se baca iznimka sa odgovarajućom porukom.

Sljedeći usmjerivač je „pageStatistcs“ koji sadrži tri krajnje točke. Prva krajnja točka je GET zahtjev koja dohvaća popis identifikacijskih brojeva najposjećenijih članaka web aplikacije „Svijet u oblacima“. URI dio krajnje točke je „/pageStatistics/mostPopular“. Krajnja točka dobiva parametra limit i *skip* iz upita (eng. *query*) koje se nalazi na putanju URL-a. Upiti su parametri koji se nalaze na kraju URL-a nakon oznake upitnika (?). Limit parametar reprezentira količinu članaka koji se treba dohvatiti, a *skip* parametar reprezentira količinu članaka koji je potrebno preskočiti prilikom dohvaćanja identifikacijskih brojeva. Parametri *skip* i limit su potrebni kako bi se ispravno implementirala funkcionalnost za listanje članaka na frontu aplikacije. Sljedeći isječak programskog koda prikazuje funkciju za dohvaćanje popisa identifikacijskih brojeva najpopularnijih članaka na temelju limita i *skip* parametra, sortirano prema broju prosječenosti stranice:

```
const getMostPopularPosts = async (limit, skip) => {
  try {
    const pageStatisticsRef = db.collection("pageStatistics");
```

```
    const snapshot = await pageStatisticsRef
      .orderBy("views", "desc")
      .limit(limit)
      .offset(skip).get();
    const mostPopularPosts = [];
    snapshot.forEach((doc) => {
      const data = doc.data();
      mostPopularPosts.push(data.postID);
    });
    return mostPopularPosts;
  } catch (error) {
    throw error;}};
```

Sljedeća krajnja točka usmjerivača „pageStatistcs“ je POST metoda za dohvaćanje popisa članaka za moderatore web stranice. Koristi se POST metoda iz razloga što se na krajnju točku šalje popis ID-jeva u tijelu zahtjeva. Kod GET zahtjeva se inače ne šalje tijelo, nego se postavljaju parametri upita za dohvaćanje specifičnih podataka, a pošto se šalje popis s neograničenim brojem ID-a, bolje odgovara POST metoda za dohvaćanje podataka. Navedena točka se nalazi na „/modStatistics“ URI putanju i dohvaća iz baze podataka popis statistike članaka i komentare svakog članaka na temelju proslijeđenog popisa ID-eva potrebnih članaka. Posljednja krajnja točka navedenog usmjerivača je PUT metoda koja ažurira broj posjećenosti stranice. Prilikom svako posjeta određenog članka potrebno je povećati statistiku posjećenosti članka. Pošto se navedena točka najviše koristi i pošto je potrebno stalno ažurirati stanje statistike posjećenosti članaka u bazi podataka, implementirano je serijsko (eng. *batch*) ažuriranje baze podataka. Nakon svakog posjeta članka back-end server dobiva zahtjev i pohrani ga u listu, nakon što je pohranio deseti zahtjev vrši se serijsko ažuriranje baze podatak. Dodatno kod svakog dobivenog zahtjeva, provjerava se da li već postoji u listi, ako postoji poveća se brojač, ako ne postoji kreira se novi ID članka i postavlja mu se brojač na jedan u listi. Sljedeći isječak programskog koda prikazuje serijsko ažuriranje podataka u Firebase bazi:

```
const updatePageStatistics = async (batchData) => {
  try {
    const batch = db.batch();
    for (const data of batchData) {
      const { postID, counter } = data;
      const pageStatisticsRef = db
        .collection("pageStatistics")
        .where("postID", "==", postID);
      const pageStatisticsSnapshot = await pageStatisticsRef.get();
```

```
      if (!pageStatisticsSnapshot.empty) {
      const pageStatisticsData = pageStatisticsSnapshot.docs[0].data();
        const pageStatisticsId = pageStatisticsSnapshot.docs[0].id;
        if (pageStatisticsData) {
          const pageStatisticsDocRef = db
            .collection("pageStatistics")
            .doc(pageStatisticsId);
          batch.update(pageStatisticsDocRef, {
            views: pageStatisticsData.views + counter,
          });
        }} else {
      const pageStatisticsDocRef=db.collection("pageStatistics").doc();
        batch.set(pageStatisticsDocRef, { postID: postID, views: counter
});
      }}
    await batch.commit();
    console.log("Batch zahtev uspešno izvršen.");
  } catch (error) {
    console.error("Greška pri izvršavanju batch zahteva:", error);
  }};
```

Prethodna funkcija za svaki serijski zapis provjerava da li postoji zapis u bazi, ako ne postoji kreira novi zapis *betch*-a, ako postoji kreira *batch* zahtjev sa ažuriranim vrijednostima. Kada se prođe kroz sve serijske zahtjeve, na kraju se samo ažurira cijeli kreirani *batch* zahtjevi.

Sljedeći usmjerivači su „postComments“ i „postCommentUpload“ koji su odvojeni iz razloga što kod prvog usmjerivača nije potrebna autentifikacija, a kod drugog je potrebna autentifikacija korisnika. Prvi usmjerivač sadrži dvije krajnje točke, koje su obje GET metode. Prva GET metoda je zadužena za dohvaćanje komentare članaka na temelju njegovog ID-a, koji je dobiveni putem upita. Krajnja točka se nalazi na „/postComments“ dijelu URI-a. Druga GET metoda se nalazi na putanju „/postComments/bestScored“ i zadužena je za dohvaćanje popisa ID-eva najboljih ocijenjenih članaka iz baze podataka. Dodatno metoda prima parametre za limit i *skip* putem upita URL-a, kako bi se dohvaćali samo potrebni podaci za listanje aktivnih najbolje ocijenjenih članaka na frontu aplikacije. Sljedeći isječak programskog koda prikazuje funkciju dohvaćanja popisa ID-jeva najbolje ocijenjenih članaka:

```
const getBestScoredPostIdList = async (limit, skip) => {
  try {
    const postCommentsRef = db.collection("postComments");
    const postCommentsSnapshot = await postCommentsRef
      .orderBy("postId")
      .get();
```

```
  if (!postCommentsSnapshot.empty) {
    const postCommentsList = [];
    const postCommentsGroupedByPostId = {};
    postCommentsSnapshot.forEach((doc) => {
      const data = doc.data();
      const postId = data.postId;
      if (postCommentsGroupedByPostId[postId]) {
        postCommentsGroupedByPostId[postId].push(data);
      } else {
        postCommentsGroupedByPostId[postId] = [data];
      }
    });

    for (const postId in postCommentsGroupedByPostId) {
      const postComments = postCommentsGroupedByPostId[postId];
      let sum = 0;
      for (const postComment of postComments) {
        sum += postComment.rating;
      }
      const avgScore = sum / postComments.length;
      postCommentsList.push({
        postId: postId,
        avgScore: avgScore,
      });}
    postCommentsList.sort((a, b) => b.avgScore - a.avgScore);
    const postCommentsListLength = postCommentsList.length;
    const postCommentsListToReturn = [];
    for (let i = skip; i < postCommentsListLength; i++) {
      postCommentsListToReturn.push(postCommentsList[i]);
      if (postCommentsListToReturn.length === limit) {
        break;
      }}
    return postCommentsListToReturn;
  }
  return null;
} catch (error) {
  console.error("Error getting best scored post id list:", error);
  return null;
}};
```

Usmjerivač „postCommentUpload“ sadrži tri krajnje točke. Prva krajnja točka je GET metoda, koja se nalazi na „/userRecommendation“ URI putanju i zadužena je za dohvaćanje popisa kategorije koje se preporučuju korisniku. Navedena krajnja točka dobiva parametar e-maila korisnika putem upita URL-a, koji će se koristiti kod algoritma za preporuke. Algoritam za preporuku najprije dohvaća popis svih komentara na temelju proslijeđenog e-maila korisnika, zatim se iz popisa dohvaćenih korisničkih komentara kreira popis polja kategorije iz svakog dohvaćenog komentara. Zatim iz kreiranog popisa kategorije izbacuju se svi dupli zapisi i kreira se varijabla brojača koji sadrži sumu svih duplih zapisa. Na kraju se sortira kreirana lista kategorije preporuke korisnika od najvećeg brojača do najmanjeg brojača. Primjerice korisnik je komentirao članak koji sadrži kategorije A i B, te potom je komentirao drugi članak koji sadrži kategorije A i C. Algoritam za preporuku će korisniku najprije prikazati sve članke koje sadrže kategoriju A, pošto se on dva puta pojavljuje u popisu i potom će se prikazati članci sa kategorijama B i C, pošto se oni pojavljuju samo jednom u popisu. Sljedeći isječak programskog koda prikazuje algoritam za preporuku kategorija korisnika:

```
const getUserCategoryList = async (userEmail) => {
  try {
    const postCommentsRef = db.collection("postComments");
    const snapshot = await postCommentsRef
      .select("postCategory")
      .where("email", "==", userEmail)
      .get();
    const userCategoryList = [];
    snapshot.forEach((doc) => {
      const data = doc.data();
      userCategoryList.push(data.postCategory);
    });
    const categoryList = [];
    userCategoryList.forEach((category) => {
      category.forEach((categoryItem) => {
        categoryList.push(categoryItem);
      });});
    const categoryListCount = {};
    categoryList.forEach((category) => {
      if (categoryListCount[category]) {
        categoryListCount[category]++;
      } else {
        categoryListCount[category] = 1;
      }});
    const sortedCategoryList = Object.keys(categoryListCount).sort(
```

```
      (a, b) => categoryListCount[b] - categoryListCount[a]
    );
    return sortedCategoryList;
  } catch (error) {
    throw ("Error getting user category list:", error);
  }};
```

Druga krajnja točka prethodnog usmjerivača je metoda POST koja se nalazi na „/pageCommentsUpload“ putanju. Njezina zadaća je dodavanje novog komentara, koji se dobije iz tijela zahtjeva. Treća metoda točke je DELETE metoda, koja se nalazi na istom putanju kao i POST metoda. Zadaća krajnje točke je brisanje komentara na temelju proslijeđenog ID-a komentara koji se dobije iz upita URL-a.

Sljedeći usmjerivač je „userData“ koji sadrži metode GET, POST i PUT. Sve metode sadrže isto putanje, a to je „/users“. Prva metoda GET je zadužena za dohvaćanje informacija od korisniku iz baze podataka na temelju proslijeđenog e-maila. Sljedeća metoda POST je zadužena za kreiranje novog korisnika na temelju proslijeđenih informacija iz tijela zahtjeva. Posljednja metoda PUT je zadužena za ažuriranje korisničkih informacija koje se dobiju iz tijela zahtjeva.

Posljednji usmjerivač je „userOrder“ koji sveukupno sadrži četiri krajnjih točki. Prva krajnja točka je GET metoda zahtjeva koja se nalazi na putanju „/order“. Zadaća točke je dohvaćanje popisa svih narudžbi korisnika na temelju proslijeđenog e-maila korisnika koji se dobije iz upitu URL-a. Dodatno točka dobiva parametre limit i *skip* pomoću kojih se dobiva prikaz aktivnih narudžbi kod listanja na frontu aplikacije. Sljedeća metoda je također GET metoda, koja se nalazi na putanju „/getAllOrders“ i njezina zadaća je dohvaćanje popisa svih narudžbi svih korisnika, namijenjeno za administrativne korisnike. Metoda dobiva parametre *skip* i limit poslane putem upita URL-a. Sljedeća metoda je POST metoda koja se nalazi na putanju „/order“, te njezina funkcionalnost je kreiranje nove narudžbe na temelju poslanih JSON podataka putem tijela zahtjeva. Posljednja metoda je PUT metoda koja se nalazi na putanju „/order/updateOrderStatus“. Navedena metoda vrši ažuriranje stanja statusa narudžbe na temelju proslijeđenog ID-a narudžbe iz tijela zahtjeva i postavlja status narudžbe na vrijednost koju također dobije putem tijela zahtjeva.

5.5. Commercetools

Commercetools kao što je već bilo u prethodnom poglavlju bilo rečeno je komercijalno rješenje, koje pruža sustav za upravljanjem sadržajem bez grafičkog sučelja s glavnim fokusom na web trgovinu / e-trgovinu. Pomoću Commercetoolsa kreirani su raznovrsni proizvodi, varijacije proizvoda, specifikacije proizvoda, metode dostavljača i ostale korisne

funkcionalnosti koje pruža alat. Commercetools pruža svojim korisnicima pruža besplatnu probu s potpunim funkcionalnosti alata na dva puna mjeseca.

Kreiranje željenog CMS sustava za e-trgovinu započinje s kreiranjem korisničkog računa na njihovoj službenoj stranici. Nakon uspješne registracije, korisnicima se pruža potpuna kontrola nad kreiranjem CMS sustava. Prije nego što se započne realizacija željenog CMS sustava potrebno je postaviti željene lokalizacije i novčanu valutu sustava. U postavkama Commercetoolsu za lokalizaciju postavljena su dva jezika, engleski i hrvatski, a za novčanu valutu je postavljeni euro. Dodatno je moguće promijeniti geografske zone za dostavu robe, porez na robu, metode dostavljača, zemlje u kojoj se nalazi e-trgovina, konfiguraciju e-trgovine i ostale postavke. Kod metode dostavljača dodan je jedan europski dostavljač, a to je DHL, kojemu je bilo potrebno dodati naziv, opis, porez, geografsku zonu i cjenik dostavljača. Cijena dostavljača DHL je postavljena na 10 eura i dodatno ako se prekorači limit od 40 eura će dostava biti besplatna.

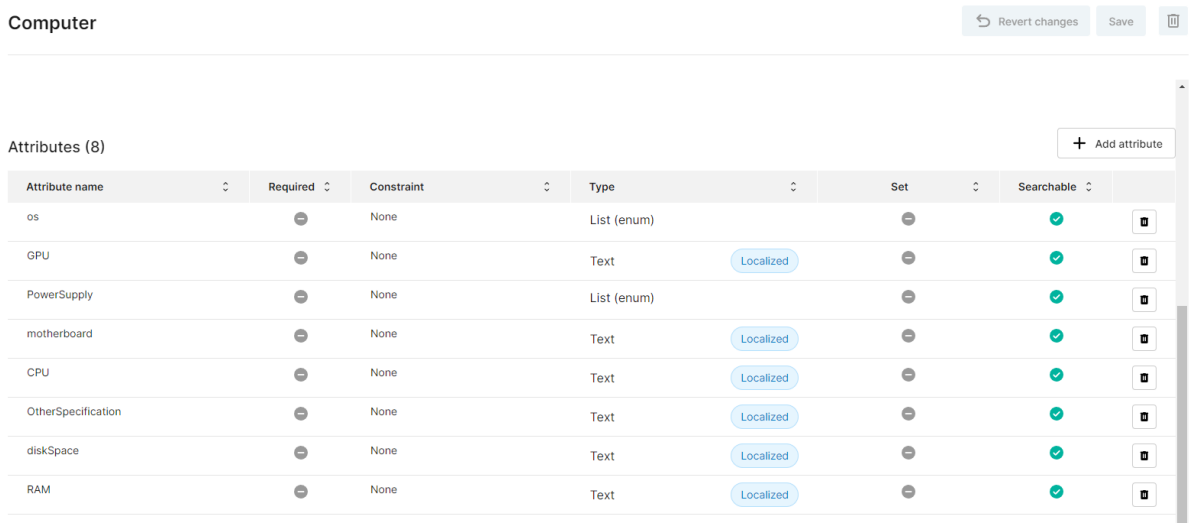
Također potrebno je unijeti željene kategorije proizvoda. Pošto je tematika web stranice „Svijet u oblacima“ bazirana na *IoT* uređajima popis kategorija i pod-kategorija je sljedeći:

1. Računala:
    1.1. Desktop
    1.2. Laptop
    1.3. Operativni sustav:
        1.3.1. Windows
        1.3.2. Linux
        1.3.3. IOS
2. Mobilni uređaji:
    2.1. iPhone
    2.2. Android
3. Konzole:
    3.1. Playstation
    3.2. Xbox
    3.3. Nintendo
4. Ostala tehnika

5.5.1. Kreiranje tipova proizvoda

Prvi korak kod realizacije CMS sustava je kreiranje željenih tipova proizvoda (eng. *product types*) i njihovih atributa. Pošto je tematika aplikacija „Svijet u oblacima“ usmjerena prema *IoT* uređajima, kreirana su tri tipa proizvoda (računalo, konzola i mobitel). Svaki tip proizvoda posjeduje svoje zasebne atribute koji će koristiti kod prikaza specifikacija o

varijacijama proizvoda na stranici o detaljima odabranog proizvoda na frontu aplikacije. Dodavanje novih tipova proizvoda i atributa se vrši u postavkama web alata Commercetoolsa pod opcijom tipovi proizvoda i atributa. Prvi tip proizvoda koji će se kreirati je računalo, gdje se dodaje samo naziv, opis i naziv ključa za željeni tip proizvoda. Nakon kreiranja tipa proizvoda dodaju se željeni atributi proizvoda. Svaki atribut posjeduje svoj tip, točnije moguće je odabrati jedan od sljedećih ponuđenih tipova: *boolean* vrijednost; tekst; broj; novčana vrijednost; datum i vrijeme; referenca; lista definiranih vrijednosti. Svaki kreirani atributi navedenog tipa proizvoda sadrži vrijednosti za naziv i opis atributa, gdje su postavljene vrijednosti za svaku lokalizaciju CMS sustava. Dodatno svakom atributu je omogućena postavka za lokalizaciju i mogućnost pretraživanja. Atributi koji su se kreirali za tip proizvoda računalo se nalaze na slici 30.

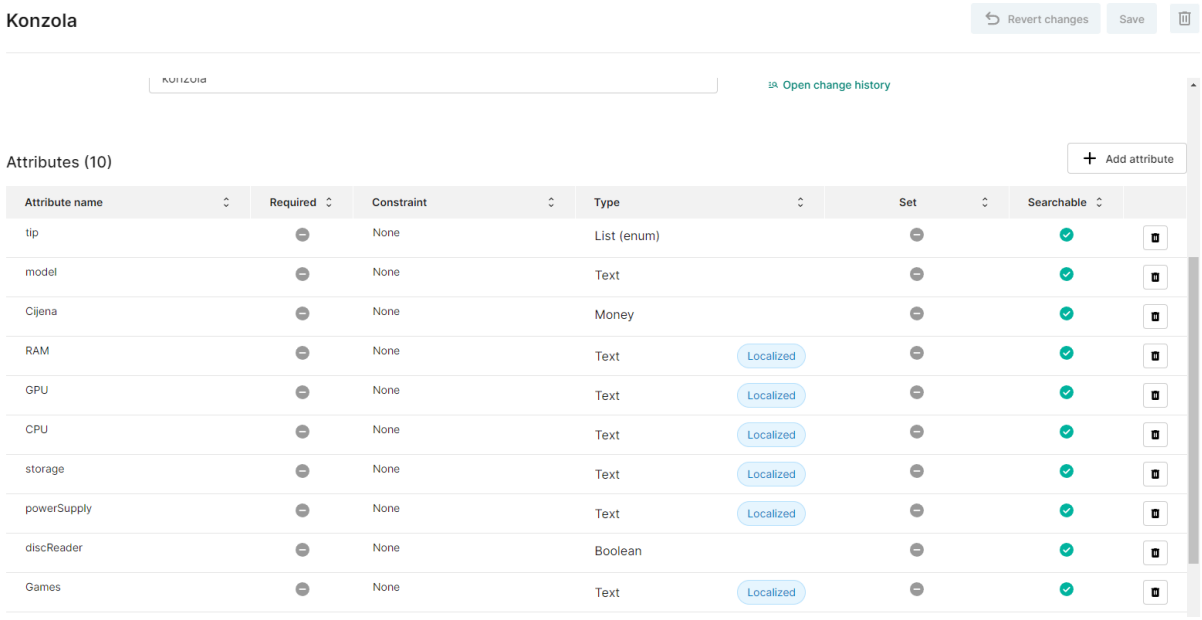


Slika 30. Prikaz kreiranih atributa za tip proizvoda računalo [autorski rad]

Atributi za operativni sustav (os) i napajanje (eng. *power supply*) je njihov tip atributa postavljeni na pred-definiranu listu, gdje su se kod operativnog sustava dodijeljene sljedeće vrijednosti: Windows 7, 8, 10 i 11; Linux; iOS. Kod atributa za napajanje, pred-definirane vrijednosti su se postavile na brojeve od 250 do 1600 sa razmacima od 50. Brojevi reprezentiraju snagu napajanja izraženo u vatima (W). Kod dodavanja stvarnih proizvoda u sustav, odabire se jedan od ponuđenih vrijednosti iz padajuće liste kod pred-definirane liste, a kod tipova atributa za tekst potrebno je unijeti stvarne vrijednosti, dodatno potrebno je unijeti vrijednosti za svaku postavljenu lokalizaciju sustava.

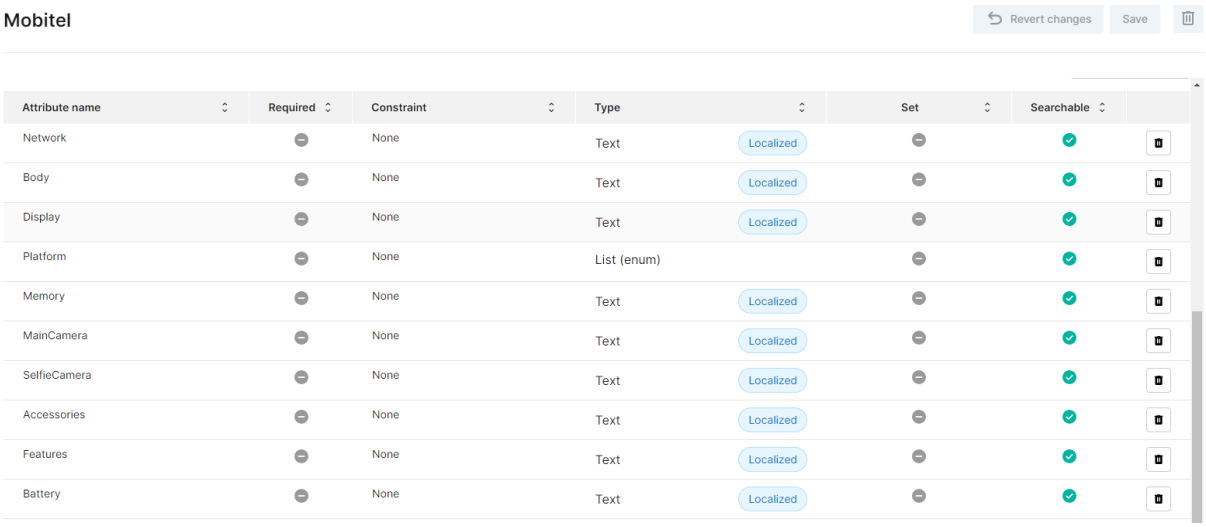
Sljedeći tip proizvoda je tip konzola. Navedeni tip proizvoda je jako slično prethodnom tipu proizvoda, posjeduje par dodatnih atributa i pojedini atributi su drugačiji. Svaki atribut posjeduje naziv i opis atributa za svaku pojedinu lokalizaciju i također su postavljene postavke

za lokalizaciju i mogućnost pretraživanja za svaki atributa. Popis svih atributa navedenog tipa proizvoda i tipovi atributa se nalazi na slici 31. Atribut pod nazivom „tip“ je tipa pred-definirana lista, koja sadrži sljedeće tri vrijednosti: Playstation, Xbox i Nintendo.



Slika 31. Prikaz kreiranih atributa za tip proizvoda konzola [autorski rad]

Posljednji tip proizvoda je tip mobitel, koji se može koristiti naravno za mobilne proizvode i dodatno mogu se koristiti za proizvode pametnih satova. Navedeni tip proizvoda posjeduje posve drugačije atribute od prethodnih tipova proizvoda. Popis svih kreiranih atributa za tip proizvoda mobitel nalazi se na slici 32. Svaki pojedini atribut naravno sadrži naziv i opis, gdje su vrijednosti dodijeljene za svaku pojedinu lokalizaciju i dodatno omogućene su postavke za lokalizaciju i mogućnost pretraživanja za svaki atribut kod tipa proizvoda mobitela. Atribut za platformu je tipa pred-definirane liste, koja sadrži vrijednosti android i iOS. Dodatno moguće je kreirati grupe atributa za proizvode, pomoću koji je moguće dodatno specificirati svaki proizvod i njegove varijacije proizvoda.

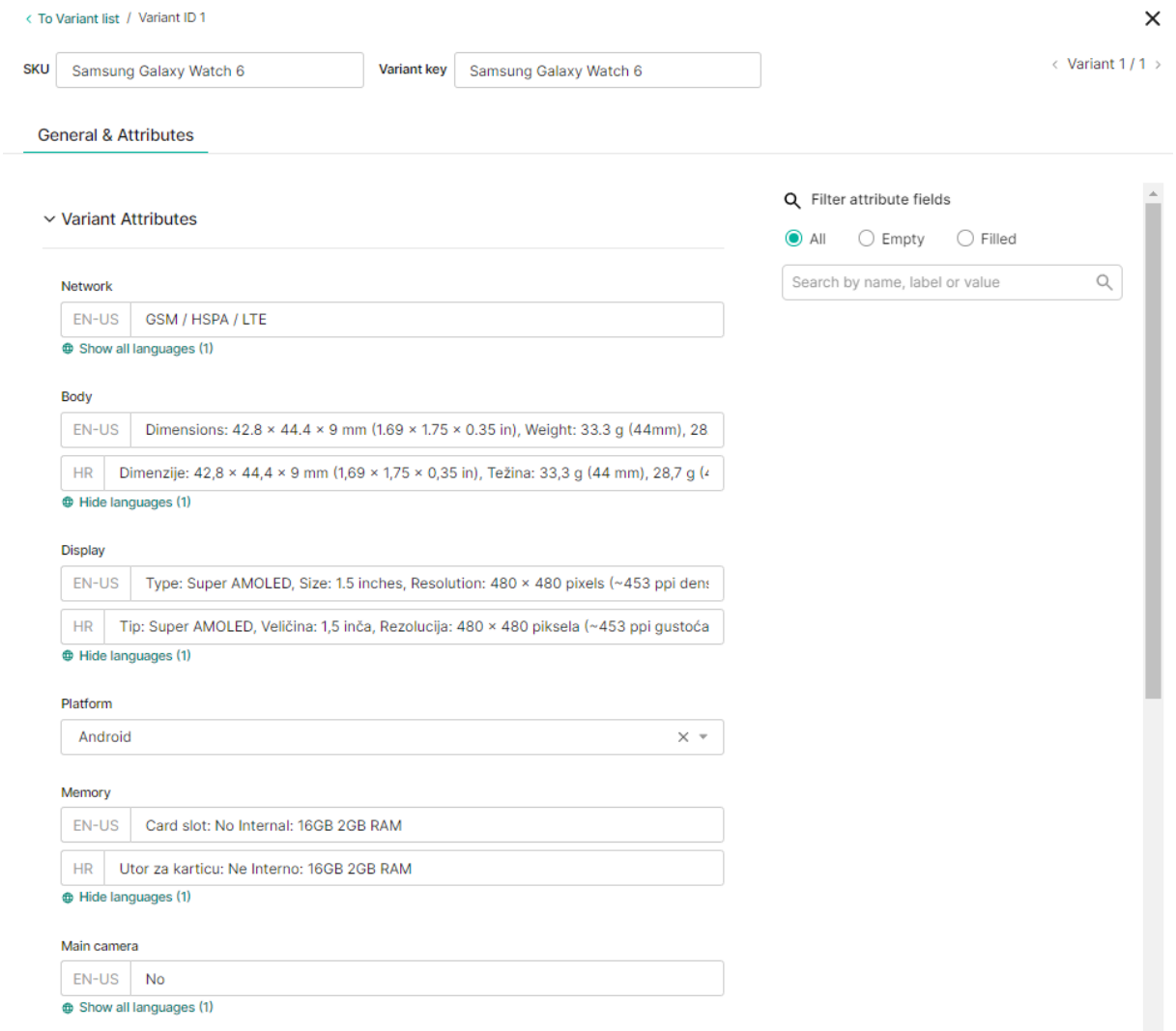


Slika 32. Prikaz kreiranih atributa za tip proizvoda mobitel [autorski rad]

5.5.2. Dodavanje novog proizvoda u sustav

Nakon što se kreiraju željeni tipovi proizvoda, konačno se mogu dodavati stvarni proizvodi u CMS sustav. Kod dodavanja novog proizvoda prvo je potrebno odabrati jedan od prethodno definiranog tipa proizvoda, primjerice odabere se mobitel. Zatim je potrebno unijeti naziv i opis proizvoda, posebno za svaku lokalizaciju, primjerice za naziv proizvoda (za svaku pojedinu lokalizaciju to može biti ista vrijednost, pošto se radi o nazivu proizvoda) se može unijeti novi pametni sat Samsung Galaxy Watch 6. Za opis sata se unese odgovarajući opis, posebno na engleskom i posebno na hrvatskom jeziku. Zatim je potrebno odabrati željene prethodno kreirane kategorije proizvoda, u ovom slučaju odabere se kategorija android. Zatim se odabire model cijene i model poreza na proizvod.

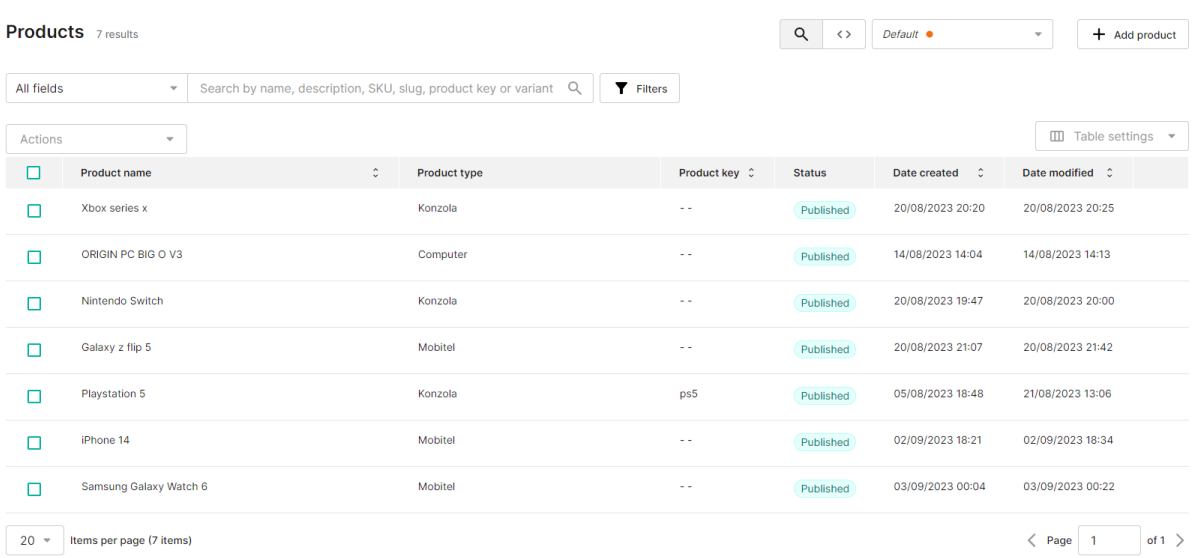
Nakon što se ispune općenite informacije o proizvodu, potrebno je kreirati barem jednu varijaciju proizvoda. Kod kreiranja varijacije proizvoda prvo se ispune SKU i varijacijski ključ (eng. *variant key*) koji će se koristiti kod prikaza naziva za varijaciju na frontu aplikacije. Navedene vrijednosti kod ovoga primjera to može biti ista vrijednost, točnije to može biti isti kao i sami naziv proizvoda, pošto za navedeni proizvod je planirana samo jedna varijacija. Kod ispunjavanja varijacije o proizvodu potrebno je ispuniti prethodne vrijednosti koje su se kreirale za atribute kod tipa proizvoda mobitel. Točnije ispunjavaju se vrijednosti za obje lokalizacije prethodno kreiranih atributa za tip proizvoda mobitel. Točan popis atributa koji se ispunjava se nalazi na prethodnoj slici 32. Slika 33. prikazuje kreiranje varijacije proizvoda prethodno spomenutog pametnog sata.



Slika 33. Kreiranje nove varijacije proizvoda pametnog sata S.G. Watch 6 [autorski rad]

Sljedeći korak je postavljanje postavki koje su povezane sa optimizacijom korisničkog pretraživanje putem tražilice. Nakon što se pohrani kreirani proizvod, omogućuje se dodavanje dodatnih specifikacija o proizvodu, kao što su cijene, količine na skladištu i dodavanje slika za svaku kreiranu varijaciju proizvoda. Kod kreirane varijacije o proizvodu potrebno je dodati željene slike o varijaciji kako bi se privukla pažnja kupaca. Nakon postavljenih željenih slika, potrebno je postaviti: cijenu varijacije; valutu; zemlje u kojoj je dostupna varijacija; grupa korisnika za koju je dostupna varijacija; kanal dostupnosti varijacije; datum do/od kada vrijedi varijacija itd. Zatim je potrebno postaviti informacije o skladištenju proizvoda. Informacije koje se postavljaju za skladištenje su: odabir skladišta u kojoj se nalazi varijacija; količina varijacije na zalihi; prosječno razdoblje obnavljanja zalihe; očekivana dostava novih naručenih zaliha. Konačno nakon što se unesu sve željene informacije o proizvodu i njihovim varijacija je moguće objaviti proizvod, koji će biti dostupan za pregled i naručivanje. Na temelju prethodnog

primjera na sličan način su kreirani i ostali proizvodi CMS sustava. Popis svih kreiranih proizvoda nalazi se na slici 34.



Slika 34. Popis svih kreiranih proizvoda sustava Commercetoolsa [autorski rad]

Posljednja stvar koju je potrebno napraviti kako bi navedeni proizvodi bili dostupni za dohvaćanje na frontu aplikacije je kreiranje API web servise za klijente. U postavkama za developere je potrebno kreirani nove web servise, gdje se samo kod prvog kreiranja web servisa dobiju informacije o tajnim ključevima za kreirani web servis, pa ih je potrebno odmah pohraniti na sigurno mjesto. Dodatno kod kreiranja web servisa odabiru se željene uloge dostupnosti, točnije ograničava se web servis. Primjerice kod prikaza svih proizvoda na frontu aplikacije, ograničava se web servis da on može samo dohvaćati popis svih proizvoda. Kod realizacije CMS sustava e-trgovine kreirana su dva web servisa, prvi pomoću kojeg se dohvaćaju svi proizvodi, kategorije i metode dostavljača iz sustava i drugi web servis za kreiranje recenzija o proizvodima i prava pisanja u CMS sustav.

5.6. React

Web aplikacija „Svijet u oblacima“ je realizirana pomoću JavaScript razvojnog okvira Reacta i stilskog okvira Tailwind CSS-a. Skoro svi pojedini dizajni su kreirani vlastoručno, no kod pojedinih dijelova dizajna je uzeta inspiracija od interneta, gdje su autori besplatno objavili njihove programske kodove pojedinih React komponente ili dijelove HTML koda.

Instalacija stilskog okvira Tailwind CSS-a je bila veoma jednostavna, točnije uz pomoć naredbene komande za npm instalaciju okvira. U novo kreiranoj Tailwind datoteci pod nazivom tailwind.config.js se konfiguriraju potrebne postavke, gdje je prvo potrebno odobriti koje sve

datoteke će koristiti Tailwind CSS, zatim se specificiraju teme i dodaci Tailwinda. Kreirana je tema za veličinu ekrana, gdje se specificiraju dimenzije ekrana, pomoću kojih je moguće definirati dizajn za svaku veličinu ekrana. Zatim su definirane boje koje će se koristiti u projektu (glavna boja je *cyan* i njezine varijacije, svijetlo crvena, crvena itd.). Posljednja stvar koja se je definirala kod teme Tailwinda je font teksta koji će se koristiti u projektu, a to je *Poppins* za sans-serif i *Merriweather* za serif.

Web aplikacija sadrži sveukupno šesnaest zasebnih stranica, koje se mogu razvrstati na pet glavna dijela web aplikacije, a to su: početna stranica; stranica o nama; shop dio aplikacije; prijava / registracija korisnika; stranice koje sadrže postavke korisničkih podataka, postavke stranice za administrativne korisnike i statistika za moderatore stranice. Početna stranica je povezana s CMS sustavom Contentfulom, gdje se prikazuje blog dio aplikacije i sve njegove funkcionalnosti. Shop dio aplikacije je povezani s CMS sustavom Commercetoolsom, gdje se prikazuje cijela e-trgovina web aplikacije i sve njezine funkcionalnosti.

Prvo što se učitava kod web aplikacije „Svijet u oblacima“ je datoteka index.js u koji se inicijaliziraju sve ostale potrebne datoteke kako bi aplikacija radila kako treba. Koriste se takozvane kuke (eng. *hooks*) u React projektu, pomoću kojih je moguće koristiti funkcije koje su se kreirale u drugim datotekama. Indeks.js datoteka koristi sljedeće kuke ReactDOM, Provider, App i store. Kuka ReactDOM dolazi iz paketa „react-dom/client“ i koristi se za renderiranje pridodane komponente App.js na ekranima korisnika. Sljedeća kuka Provider, koji dolazi od paketa „react-redux“, zajedno sa vlastito kreiranom datotekom store koriste se slanje informacija iz jedne stranice na drugu. Dodatno se inicijaliziraju datoteka i18m.js pomoću kojih se definira korišteni jezik web aplikacije. Glavna zadaća indeks.js datoteke je renderiranje App.js datoteke na ekranima korisnika. App.js datoteka sadrži cijelu logiku za navigaciju i URL putanja do svih stranica i renderira početnu stranicu web aplikacije.

Ukratko rečeno store kuka je realizirana pomoću paketa „@redujs/toolkit“, gdje se učitavaju potrebne datoteke, čija je zadaća pružanje komunikacije između dvije različite web stranice. Navedene datoteke su: authReducer; kangReducer; userOrderReducer; productDetailReducer. Datoteke koriste kuku createSlice pomoću kojih se definiraju željeno početno stanje podataka koji se šalje na drugu stranicu i definiraju se funkcije za postavljanje željenih stanja podataka.

I18n.js datoteka je zadužena za pružanje funkcije da korisnici mogu mijenjati korišteni jezik web aplikacije. Web aplikacija je realizirana da pruža lokalizaciju za tri jezika, a to su engleski, hrvatski i njemački. Kod realizacije za lokalizaciju koriste se paketi „i18next“ i „react-i18next“, pomoću kojih se definiraju željeni jezici. Kreirane su tri različite datoteke, posebno za svaki jezik. Datoteke sadrže varijable sa pripadajućim tekstom na pripadajućem jeziku, koji će

se prikazivati na ekranima korisnika. Svi tekstovi koji se prikazuju korisnicima će biti na aktivnom jeziku, što korisnici naravno mogu mijenjati. Kod prvog pokretanja web aplikacije početni aktivni jezik je postavljeni na engleski.

App.js datoteka sadrži cijelu logiku i URL putanja do svih ostalih stranica koje se koriste u projektu. Za realizacije navigacije web aplikacije koriste se kuke „Outlet“, „RouterProvider“ i „createBrowserRrouter“ od paketa „react-router-dom“-a. Najprije se inicijaliziraju sve potrebne datoteke svih stranica i potom se dodijele odgovarajuće URL putanja. Sljedeći isječak programskog koda prikazuje kreiranje router-a, koji će se koristiti za navigaciju do svih stranice web aplikacije.

```
const router = createBrowserRouter([
  {
    path: "/",
    element: <RootLayout />,
    errorElement: <PageNotFound />,
    children: [      {
        index: true, element: <HomePage />,
      },{
        path: "AboutPage", element: <AboutPage />,
      },{
        path: "detailPage/:postID", element: <DetailPage />,
      },{
        path: "login", element: <LoginPage />,
      },{
        path: "registration", element: <RegistrationPage />,
      },{
        path: "userSetting", element: <UserSettingPage />,
      },{
        path: "searchedContent/:searchedContent", element:
            <SearchedContentPage />,
      },{
        path: "pageSettings", element: <PageSettings />,
      },{
        path: "modStatistics", element: <ModStatisticsPage />,
      },{
        path: "shop", element: <Outlet />,
        children: [{
            index: true, element: <ShopPage />,
          },{
            path: "shopDetail/:productName",element: <ShopDetailPage />,
```

```
            },{
              path: "userCart", element: <UserCartPage />,
            },{
              path: "checkOutOrder", element: <CheckOutOrder />,
            },{
              path: "userShoppingHistory", element: <Outlet />,
              children: [{
                  index: true, element: <UserShoppingHistoryPage />,
                },{
                  path: "orderDetail/:orderNumber", element:
                      <OrderDitailPage />,
                },],
            },],
        },],
    },]);
function App() {
  return <RouterProvider router={router} />;
}
export default App;
```

Svaka pojedina navedena stranica („HomePage“, „AboutPage“, „DetailPage“ itd.) iz prethodnog isječka programskog koda sadrži logiku pomoću kojih se dohvaćaju potrebne informacije iz željenih web servisa i definiraju se potrebne komponente koje je potrebno generirati na korisničkim ekranima. Prikaz pojedinih stranica i njihova funkcionalnost će se objasniti malo kasnije u radu, prvo je potrebno razjasniti kako se dohvaćaju potrebni podaci iz različitih web servisa, koji će se potom i prikazivati na korisničkim ekranima.

5.6.1. Dohvaćanje podataka iz Back-end servera

Informacije koje je potrebno dohvaćati iz back-end servera je izrazito jednostavno i lagano, pošto su web servisi bili građeni paralelno sa front-endom aplikacije i veoma su prilagođeni frontu. Prilikom pozivanja zahtjeva prema back-end API-ju za dohvaćanje podataka (GET metoda) je potrebno samo definirati željeni URL. Kod specifičnih dohvaćanja podataka postavlja se *querry* dio na željenom URL-u. Kod manipulacije podataka nad bazom podataka potrebno je postaviti autentifikaciju kod slanja zahtjeva na web servis, gdje se za autentifikaciju postavlja *Bearer* i autentifikacijski token korisnika, koji se dobije prilikom prijave ili registracije u web aplikaciju. Prilikom POST ili PUT zahtjeva podaci koje je potrebno pohraniti ili ažurirati u bazu podataka se šalju u samome tijelu zahtjeva. Pošto su zahtjevi prema back-end serveru veoma jednostavni, te dobiveni podaci se ne moraju dodatno filtrirati i oblikovati, zahtjevi se direktno šalju i čekaju na pojedinim stranicama. Pojedini zahtjevi prema back-end

serveru se vrše u kombinaciji sa zahtjevima od Contentfula, te se iz tog razloga nalaze u istoj datoteci kao i zahtjevi od Contentfula, točnije u datoteci fetchContent.js. Sljedeći isječak koda prikazuje asinkronu funkciju za dohvaćanje popisa kategorija, koje će prijavljenom korisniku biti preporučene.

```
async function getUserReccomendationCategoryList(email, tokenKey) {
  const url = `${serverURL}/userRecommendation?userEmail=${email}`;
  try {
    const response = await fetch(url, {
      method: "GET",
      headers: {
        "Content-Type": "application/json",
        Authorization: `Bearer ${tokenKey}`,
      },
    });
    if (!response.ok) {
      throw new Error("Network response was not ok");
    }
    const data = await response.json();
    return data;
  } catch (error) {
    console.log("Error fetching user recomendation:", error);
    return [];
  }}
```

Prethodni isječak programskog koda prikazuje kako je realizirano dohvaćanje podataka GET metodom zahtjeva. Kao što je bilo prije rečeno kod GET zahtjeva za dohvaćanje specifičnih podataka, postavlja se *querry* dio na željenom URL-u, u prethodnom primjeru postavlja se e-mail prijavljenog korisnika za *querry* nad URL-om. Dodatno je potrebno postaviti token, pomoću kojeg je provjerena autentifikacija korisnika. Sljedeći isječak programskog koda prikazuje slanje zahtjeva POST metode kod registracije korisnika u web aplikaciju.

```
async function createNewUserInDB(userData, token) {
  const serverUrl = serverURL + "/users";
  let headers = new Headers();
  headers.append("Authorization", "Bearer " + token);
  headers.append("Content-Type", "application/json");
  try {
    const response = await fetch(serverUrl, {
      method: "POST",
      body: JSON.stringify(userData),
      headers: headers,
```

```
    });
    if (!response.ok) {
      throw new Error("Failed to register user");
    }
  } catch (error) {
    throw new Error("Failed to connect to the server");
  }}
```

Najprije se vrši registracija direktno na frontu prema Firebase bazi podataka, gdje se šalje e-mail i lozinka, kako bi se kreirao korisnički račun. Nakon uspješne registracije, Firebase vraća token pomoću kojeg je moguće dalje obraditi zahtjev. Daljnja obrada se vrši prema back-end serveru, gdje se kreira novi korisnik svim ispunjenim podacima o korisniku. Pošto Firebase autentifikacija pohranjuje samo podatke o e-mailu i lozinki i njezina funkcija je da pruža token korisnicima, potrebna je dodatna tablica u bazi za pohranu dodatnih informacija o korisnicima. Pohrana dodatnih informacija o korisniku je realizirano pomoću back-end servera, gdje se uneseni podaci od fronta šalju u tijelu zahtjeva prema back-end serveru. Dodatno kod slanja zahtjeva je potrebno postaviti odgovarajući: URL, koji se sastoji od općeg dijela, koji je uvijek isti kod svake krajnje točke back-end servera i dio koji je povezani za krajnju točku na koju se šalje zahtjev (opći dio je „http://localhost:5050/api“ i drugi dio za krajnju točku prethodnog primjera je „/users“); odgovarajuća metoda, a to je POST u prethodnom primjeru; zaglavlje (eng. *header*), gdje se postavlja korisnički token i dodatno je potrebno specificirati tip sadržaja koji se šalje u tijelu zahtjeva kod bilo koje POST metode. Tip sadržaja koji se šalje kod bilo koje POST metode ovoga projekta će uvijek biti isti, a to je JSON format. U slučaju izbijanja bilo kakvog problema, prikazati će se odgovarajuća poruka korisniku.

Na temelju prethodnih primjera, kreirani su svi ostali zahtjevi na sličan način, što se tiče back-end servera.

5.6.2. Dohvaćanje podataka iz Contentfula

Podaci iz Contentfula se dohvaćaju također putem REST API-a, no za razliku od back-end servera, dohvaćanje od Contentfula je malo kompleksnije i zahtjevnije. Potrebno je specificirati opcije, pomoću kojih se dohvaćaju specifične informacije iz Contentfula, zatim dohvaćene informacije je potrebno dodatno doraditi, formatirati i prilagoditi da odgovaraju front-endu aplikacije. Kreirana je posebna datoteka fetchContent.js, čija je zadaća vršenje komunikacije između fronta i Contentfula. Kod realizacije navedene datoteke koriste se kuke „useState“ i „useEffect“ od Reacta. UseEffect se koristi za ažuriranje stanja varijabli kada mu se pridoda nova vrijednost, a useEffect se koristi kada se želi provesti željena funkcija kada se promijeni vrijednost varijabli koje su postavljene kao uvjeti useEffect-a. Dodatno koristi se kuka „createClient“ iz paketa Contentfula, pomoću koje se uspostavlja komunikacija prema

Contentfulovim web servisima. Varijable koje je potrebno postaviti za komunikaciju za Contentful su poslovni prostor (eng. *space*), okruženje (eng. *environment*) i token za pristup. Sve potrebne varijable se dobiju iz Contentfulovog projekta na njihovoj stranici. Posljednja kuka koja se koristi je „useSessionStorage“ pomoću koje se dohvaća aktivni jezik, kojeg su postavili korisnici.

Svi zahtjevi koji se rade prema Contentfulu sadrže dvoje funkcije, dok pojedini zahtjevi mogu biti sastavljeni od tri funkcije. Zahtjevi koji sadrže tri funkcije rade najprije zahtjeve prema back-end servere i potom s dohvaćenim podacima vrše zahtjeve prema Contentfulu ili obrnuto prvo prema Contentfulu i onda prema beck-endu. Kod zahtjeva gdje su potrebni podaci samo iz Contentfula najprije se kreira varijabla opcije (eng. *options*) u prvoj funkciji, pomoću koje se specificiraju podaci koje je potrebno dohvatiti iz Contentfula. Druga funkcija koja je uvijek ista i upotrjebljava se kod svih zahtjeva koje se vrše prema Contentfulu, vrši zahtjev prema Contentfulu na temelju prethodno kreirane varijable za opcije. Dodatno druga funkcija dohvaćene podatke pretvara u prilagođeniji objekt, pomoću koje je mnogo lakše doći do dohvaćenih informacija. Sljedeći isječak programskog koda prikazuje drugu navedenu funkciju:

```
const fetchData = async (options) => { try {
    const request = client.getEntries(options);
    const response = await request;
    const contents = response.items.map((item) => {
      const {title, author, dateOfCreation, subTitle, categories, image,
            content} = item.fields;
      const id = item.sys.id;
      const img = image?.fields?.file?.url;
      return {title, author, dateOfCreation, subTitle, categories, img,
        id, content};
    });

    return { contents, totalCount: response.total };
  } catch (error) {
    console.log(error);
    return { error };
  }
};
```

Primjer dohvaćanja popisa sadržaja, točnije popis članaka iz Contentfula za prikaz članaka na blogu je prikazano u sljedećem isječku programskog koda:

```
export function useFetchContent(limit, skip, category = []) {
  const [loading, setLoading] = useState(true);
```

```
const [data, setData] = useState([]);
const [error, setError] = useState(null);
const [totalCount, setTotalCount] = useState(0);
const [languageStorage] = useSessionStorage("language", "en-US");
useEffect(() => {
  const options = {
    content_type: "contents",
    locale: languageStorage,
    order: "-fields.dateOfCreation",
    limit: limit,
    skip}: skip;
  if (category && category.length > 0) {
    options["fields.categories[in]"] = category.join(",");
   }
  fetchData(options).then(({ contents, totalCount, error }) => {
    if (!error) {
      setLoading(false);
      setData(contents);
      setTotalCount(totalCount);
    } else {
      setError(error);
  }});
}, [limit, skip, languageStorage, category]);
return { loadingData: loading, data, totalCount, error };}
```

Prva stvar koja se radi kod prethodno spomenutog isječka programskog koda postavljaju se kuke „useState“ za:

- učitavanje (eng. *loading*) – varijabla pomoću koje je moguće korisnicima prikazivati malu animaciju za učitavanje podataka, kada je potrebno čekati podatke iz pojedinih web servisa;
- podatke (eng. *data*) – služi za pohranu objekta dohvaćenih podataka iz Contentfula, gdje su podaci već prilagođeni za front-end aplikacije;
- pogreške (eng. *error*) – služi za pohranu greške, kada izbije bilo koja pogreška;
- ukupnu količinu podataka (eng. *total count*) – služi za pohranu ukupnog broja sadržaja koji se nalazi u Contentfulu. Ukupna količina članaka je potrebna na frontu kod prikaza listanja (eng. *pagination*) članaka;
- podatke o jeziku (eng. *language storage*) – služi za dohvaćanje aktivnog jezika koji se je prethodno pohranio u lokalnu memoriju web preglednika.

Dodatno u prethodnom primjeru funkcija dobiva varijable *skip*, limit i popis kategorija. Pomoću varijabli limit i *skip* se dohvaćaju samo trenutni aktivni sadržaji na frontu aplikacije, a varijabla kategorije sadrže popis kategorija za koje je potrebno dohvatiti sadržaje iz Contentfula. Zatim se definira varijabla za opcije, točnije u prethodnom primjeru se definira iz kojeg tipa modela sadržaja je potrebno dohvaćati sadržaje iz Contentfula, potom se definira lokalizacija (na kojem jeziku je potrebno dohvatiti sadržaje). Zatim se je definiralo sortiranje za dohvaćanje sadržaja, točnije ovdje se je definiralo da se dohvaćaju od najnovijih sadržaja do najstarijih sadržaja. Posljednje stvari koje su se dodale u varijablu opcije su varijable limit i *skip*. Limit obuhvaća količinu članaka koje je potrebno dohvatiti, a *skip* varijabla predstavlja količinu članaka koje je potrebno preskočiti kod dohvaćanja sadržaja iz Contentfula. Dodatno ako je korisnik filtrirao sadržaje prema kategorijama članaka (korisnici mogu odabrati jednu ili više kategorija iz padajuće liste), onda se dodatno postavlja opcija za dohvaćanje samo onih članaka koje sadrža kategorije koje je korisnik odabrao. Nakon definiranja željenih opcija, poziva se prethodno navedena druga funkcija, koja je zadužena za dohvaćanje sadržaja iz Contentfula. Ako su se uspješno dohvatili podaci iz Contentfula, varijable za učitavanje se postavlja na vrijednost *false*, postavlja se ukupna količina svih članaka i postavljaju se dohvaćeni sadržaji. U slučaju izbijanja bilo kakvog problema, postavlja se varijabla o pogreškama.

Ako korisnici na frontu zahtijevaju učitavanje 2. ili bilo koje druge stranice za listanje članaka, mijenjaju se varijable za limit i *skip*, što automatski dolazi do ponovnog dohvaćanja novih sadržaja iz Contentfula. Pošto su varijable limit i *skip* navedene kao uvjeti kod kuke „useEffect“, bilo kakva promjena navedenih varijabli aktivira se „useEffect“ i ponovno dohvaća nove sadržaje. Dodatni uvjeti kod „useEffect“-a su varijable za aktivni jezik i popis kategorija, kako bi se ponovno dohvatili sadržaji ako se promijeni aktivni jezik ili korisnici filtriraju sadržaje prema željenim kategorijama.

Kako je kreirani prethodni primjer zahtjeva, kreirani su svi ostali zahtjevi prema Contentfulu, naravno uz male promjene. Sljedeća kuka je „seFetchFirstContent“, pomoću koje se dohvaća najnoviji članak iz CMS sustava. Kod varijable za opcije dohvaćanja sadržaja postavljeni je limit na 1, a *skip* vrijednost na 0. Ostalo dio programskog koda je isti kao i kod prethodnog primjera.

Sljedeća kuka je „useFetchContentByMultipleId“, koja se sastoji od tri funkcija. Prvo se dohvaća lista identifikacijskih brojeva za najpopularnije članke web aplikacije iz back-end servera. Na temelju dohvaćenih informacija se dohvaćaju popis članaka iz Contentfula, tako da se kod varijable za opcije definira sljedeća linija „`"sys.id[in]": mostPopular PostsIds.join(",")`“. Pomoću navedene linije spajaju se svi dohvaćeni ID-jevi u jednu

tekstualnu vrijednost, pomoću koje je definirano koje sve sadržaje je potrebno dohvatiti. Dodatno se ograničava dohvaćanje sadržaja iz Contentfula varijablama za *skip* i limit.

Sljedeća kuka „useFetchContentBestScored“ je slična kao i prethodna kuka koja dohvaća najpopularnije članke. Ova funkcija dohvaća najprije popis ID-jeva najboljih ocijenjenih članaka iz back-end i potom na temelju dohvaćene liste ID-jeva se dohvaćaju sadržaji iz Contentfula.

Sljedeća kuka koja se je kreirala je „useFetchContentBySearch“ za dohvaćanje članaka na temelju korisničkog pretraživanja. Funkcija dobiva tekstualnu vrijednost koju je korisnik unio u tražilicu. Sljedeći isječak programskog koda prikazuje varijablu za opcije, koja se koristi kod navedene funkciju za pretraživanje:

```
const options = {
      content_type: "contents",
      locale: languageStorage,
      query: searchedData,
      limit: limit,
      skip: skip};
```

Sljedeća kuka je „useFetchContentByEmail“, gdje se najprije dohvaćaju sadržaji iz Contentfula na temelju e-maila za prijavljenog moderatora web aplikacije. Na temelju dohvaćenih sadržaja iz Contentfula kreira se popis ID-jeva pomoću koji se dohvaćaju informacije o statistici za svaki članak od prijavljenog moderatora od back-end servera. Kod dohvaćanja sadržaja od Contentfula ograničeno je sljedećim programskim kodom:

```
const options = {
      content_type: "contents",
      locale: languageStorage,
      "fields.authorEmail": email,
      select:
     "sys.id,fields.title,fields.dateOfCreation,fields.categories",};
```

Znači dohvaćaju se samo sadržaji koji sadrže e-mail prijavljenog korisnika i dodatno je ograničeno da se dohvaćaju samo one informacije koje je potrebno prikazati na frontu aplikacije.

Posljednja kuka koja se je kreirala za Contentful je „useFetchContentByUser Recomendation“ – najprije se dohvaćaju popis ID-jeva iz back-end i potom se dohvaća sadržaj iz Contentfula. Kuka je zadužena za prikaz članaka koji se preporučuje prijavljenom korisnicima.

5.6.3. Dohvaćanje podataka iz Commercetoolsa

Dohvaćanje podataka iz Comeercetoolsa je slično kao i kod Contentfula, samo što prilikom slanja bilo kojeg zahtjeva je potreban token za pristup podacima. Token se dobiva na temelju prethodno spomenutim tajnim podacima koji su se dobili prilikom kreiranja REAT API-a na web stranici od Commercetoolsa. Prvi korak za dohvaćanje podataka iz Commercetoolsa je dohvaćanje tokena, pomoću kojeg je onda moguća daljnja obrada zahtjeva. Sljedeći isječak programskog koda prikazuje funkciju za dohvaćanje tokena, na temelju varijabli „clientId“ i „clientSeret“ koji su dobiveni prilikom generiranja API točaka na Commercetools stranici, postoje dva para varijabli (jedan par za dohvaćanje podataka i drugi par za kreiranje recenzija proizvoda i pisanje u CMS sustav):

```
export function useFetchProductGetAccessToken(clientId, clientSecret) {
  const [accessToken, setAccessToken] = useState(null);
  useEffect(() => {
    const fetchAccessToken = async () => {
      const accessToken = await fetch(commercetoolsAuthUrl, {
        method: "POST",
        headers: {
          "Content-Type": "application/x-www-form-urlencoded",
          Authorization: `Basic ${btoa(`${clientId}:${clientSecret}`)}`,
      },});
      const data = await accessToken.json();
      setAccessToken(data.access_token);
    };
    fetchAccessToken();
  }, []);
  return accessToken;
}
```

Prethodni navedeni programski kod šalje POST zahtjev prema Commercetools krajnjoj točci, gdje je potrebno koristiti osnovno ovlaštenje / osnovnu prijavu (eng. *basic authorization*). Kod osnovne prijave varijabla „clientId“ se koristi kao korisničko ime, a „clientSecreat“ se koristi kao lozinka. Nakon što se dohvati token za pristup, šalje se drugi GET zahtjev pomoću kojeg se dohvaćaju podaci iz CMS sustava e-trgovine. Sljedeći isječak programskog koda prikazuje React kuku za dohvaćanje popisa proizvoda na temelju limit i *skip* parametra:

```
export function useFetchProductFromCommerceTools(accessToken, limit,
skip) {
  const [products, setProducts] = useState([]);
  const [loading, setLoading] = useState(true);
  const [totalCount, setTotalCount] = useState(0);
```

```
  useEffect(() => {
    const fetchProducts = async () => {
      try {
        const response = await fetch(
          `${commercetollsUrl}/products?limit=${limit}&offset=${skip}`,
          { headers: { Authorization: `Bearer ${accessToken}`}});
        const data = await response.json();
        setTotalCount(data.total);
        setProducts(data.results);
        setLoading(false);
      } catch (error) {
        console.error("Error fetching products:", error.message);
        setLoading(false);
    }};
    if (accessToken !== null) fetchProducts();
  }, [accessToken, limit, skip]);
  return { products, loading, totalCount };
}
```

Prethodan isječak programskog koda u principu radi na isti način kao i prethodni zahtjev od Contentfula. Jedino što je drugačije je naravno URL na koji se šalje zahtjev i dodatno je potreban token za autorizaciju.

Sljedeći zahtjev koji je kreirani je „useFetchShippingMethods“, pomoću kojih se dohvaćaju popis svih metoda dostavljanja. Metoda samo koristi drugačiji URL i token za pristup sadržaju kako bi dohvatila popis dostavljača.

Sljedeći zahtjev je „useFetchCategories“, pomoću kojeg se dohvaća popis svih kategorija iz sustava. Pošto prilikom dohvaćanja proizvoda dobivaju se identifikacijski brojeve za svaku kategoriju koju proizvodi posjeduju. Iz navedenog razloga potrebno je dohvatiti popis svih kategorija kako bi se korisnicima prikazale ispravne informacije na frontu aplikacije. Dodatno je postavljeni limit na vrijednost 100 prilikom dohvaćanja kategorija. Iz razloga što je zadana postavka za limit postavljena na vrijednost 20, a to je premaleni broj za dohvaćanje cijelog popisa kategorija od proizvodi.

Sljedeći zahtjev koristi token za kreiranje novih recenzija u sustav. Prilikom slanja zahtjeva korisnici mogu unijeti barem jednu od sljedećih vrijednosti: naslov, tekst i ocjenu recenzije za odabrani proizvod. Kod kreiranja zahtjeva najprije se definira tijelo zahtjeva. Tijelo zahtjeva sadrži varijable za naziv autora, naslov, tekst, ocjenu i podatke o recenziji, gdje se definira za što je recenzija (za proizvode) i vrijednost identifikacijskog broja (postavljeni je ID

proizvoda za kojeg se radi recenzija). Kreirano tijelo zahtjeva se postavi u POST zahtjev sa odgovarajućim URL i tokenom, koji se pošalje na Commercetools.

Sljedeći zahtjev koji se je kreirao je funkcija „useFetchProductReviews“ za dohvaćanje pohranjenih recenzija određenog proizvoda. Programski dio koda navedenog zahtjeva je veoma sličan kao i prethodni primjer programsko koda.

Posljednji zahtjev je „useRemoveReview“ pomoću kojeg korisnici mogu brisati njihove objavljene recenzije o proizvodima. Dodatno kod posljednjeg zahtjeva za brisanje recenzija URL putanja je malo drugačija, pošto osim ID-a recenzije potrebno je poslati i verziju recenzije. Primjerice URL može biti sljedeći „opći dio/reviews/ID proizvoda?version=verzija rcenzije“. Metoda zahtjeva se postavi na DELETE i potrebno je poslati token za autorizaciju.

5.6.4. Glavne funkcionalnosti stranica web aplikacije

Svaka stranica aplikacije „Svijet u oblacima“ sadržava: navigacijski dio: komponentu za odabir željenog jezika stranice; komponentu za tražilicu članaka; podnožje stranice. Navigacijski dio sadrži s lijeve strani navigaciju za glavne stranice, a s desne strane navigaciju za prijavu i registraciju korisnika.

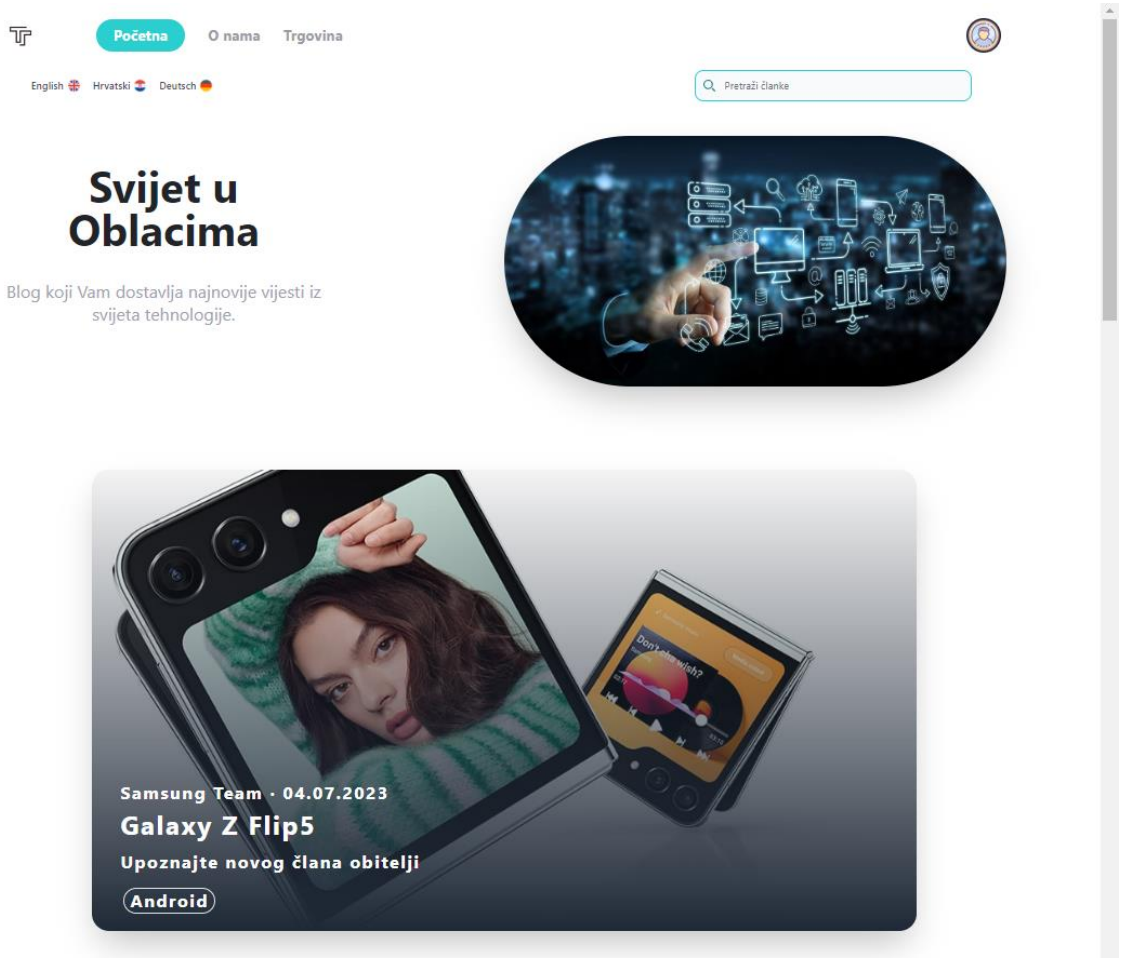
Komponenta za odabir željenog jezika sadrži opcije za engleski, hrvatski i njemački jezik. Dodatno uz navedeni tekst jezika se prikazuje mala ikona zastave za svaki jezik.

Komponenta za pretraživanje članaka pretražuje sav sadržaj iz Contentfula na temelju unesenih traženih vrijednosti korisnika. Nakon što korisnik unese prvo slovo u tražilicu, pojavljuje se gumb za dohvaćanje sadržaja. Nakon što korisnik klikne na gumb za traženje članaka, otvara se nova stranica, gdje se korisniku prikažu pronađeni članci ili u slučaju neuspješnog pronalaženja rezultata, kosniku se prikaže mala animacija za neuspješno pretraživanje.

Podnožje stranice prikazuje logo aplikacije „Svijet u oblacima“, linkove alata i njihov logo pomoću koji je realizirana aplikacija i tekst za autorska prava.

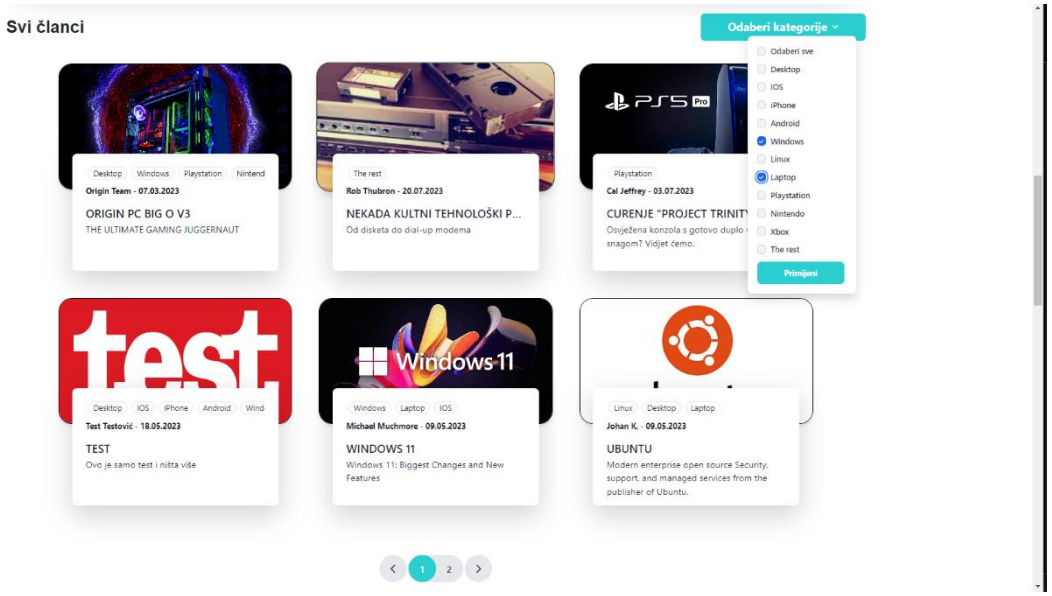
5.6.4.1. Početna stranica

Početna stranica najprije prikazuje naslov, podnaslov i naslovnu sliku web stranice. Potom se kroz skoro cijelu stranicu prikazuje najnoviji članak i njegove općenite informacije o članku, koji se dohvaća putem prethodno spomenutih zahtjeva prema Contentfulu. Slika 35. prikazuje navedene dijelove početne stranice „Svijet u oblacima“.



Slika 35. Prikaz prvog dijela početne stranice aplikacije „Svijet u oblacima“ [autorski rad]

Zatim se korisnicima prikazuje popis sljedećih x najnovijih članaka. Vrijednost x predstavlja količinu članaka koji će se prikazivati u popisu svih članaka, koje se nalaze ispod početnog članka. Početna postavka vrijednosti x iznosi 6. Navedenu vrijednost x mogu mijenjati administrativni korisnici u postavkama web stranice. Kod popisa koji prikazuje sve članke iz Contentfula, implementirano je straničenje (eng. *paging*). Korisnici mogu listati sve članke, od početnih pa sve do posljednjih članaka pomoću komponente za navigaciju za listanje članaka. Komponenta za listanje je kreirana tako da se može iskoristiti i kod ostalih popisa, kojima je potrebno straničenje. Kada korisnici kliknu na gumba za sljedeću stranicu ili bilo koji ponuđeni broj od straničenja, šalju se novi zahtjevi prema krajnjim točkama, dohvaćaju se podaci i potom prikazuju se korisnicima novi dohvaćeni podaci. Dodatno kod komponente za prikaz svih članaka je implementirani filter, pomoću kojeg korisnici mogu filtrirati sve članke prema kategorijama članaka. Kada se odabere jedna ili više kategorija poziva se prethodno spomenuti zahtjev za dohvaćanje novog popisa članaka koje sadrže odabranu/e kategoriju/e. Slika 36. prikazuje komponentu za prikaz svih članaka, koja se nalazi na početnoj stranici ispod komponente s najnovijim člankom.



Slika 36. Izgled komponente za prikaz svih članaka bloga [autorski rad]

Prethodna slika 36. dodatno prikazuje dizajn kreiranog filtra za kategorije, koji se nalazi na desnoj strani ekrana. Kada korisnici otvore filter, otvara im se cijeli popis kategorija članaka. Otvorene kategorije korisnici mogu selektirati i potom filtrirati članke. Na samome dnu navedene komponente nalazi se komponenta za straničenje članaka.

Sljedeća komponenta prikazuje popis najpopularnijih, točnije najposjećenijih članaka web stranice. Kod početne postavke za prikaz komponente najpopularnijih članaka postavljeno je da se prikazuje jedan redak članaka, točnije tri članka. Članke je naravno moguće listati, gdje se prilikom listanja na novu ili prethodnu stranicu dohvaćaju novi članci iz Contentfula. Izgled navedene komponente je moguće također promijeniti u postavkama stranice.

Sljedeća komponenta koja se prikazuje korisnicima je popis najbolje ocjenjenih članaka. Gleda se samo srednja vrijednost ocjeni svih komentara za svaki članak i korisnicima se potom prikazuju članci s najboljom srednjom vrijednosti pa sve do najlošijih. Prikaz dizajna komponente je isti kao i kod prethodne komponente za najpopularnije članke i moguće ga je mijenjati u postavkama.

Posljednja komponenta koja se prikazuje samo prijavljenim korisnicima je popis članaka koji se preporučuje prijavljenom korisniku. Sadržaji preporučeni za korisnika dohvaćaju se pomoću zahtjeva opisanog na 69. stranici ovoga rada. Dizajn navedene komponente za preporuku jednak je dizajnu prethodno dvije komponente, i također omogućuje prilagodbu broja prikazanih članaka za preporuku putem postavki stranice.

Posljednja stvar u vezi početne stranice, programski kod za početnu stranicu nalazi se u HomePage.js datoteci. Datoteka prilikom prvog posjećivanja korisnika na stranicu dohvaća

podatke o postavkama stranice i pohranjuje ih u lokalnu memoriju preglednika. Podaci o postavkama stranice sadrže informacije s kojima se generiraju komponente na početnu stranicu. Potom dohvaća sadržaj za generiranje prvog / najnovijega članka za početnu stranicu, zatim se dohvaća popis svih članaka (ograničeni sa varijablama za *skip* i limit), koji se potom prikazuju korisnicima. Prilikom čekanja podataka iz web servisa, korisnicima se prikazuje animacija za čekanje. Datoteka početne stranice zapravo definira sve ostale komponente koje je potrebno generirati na početnoj stranici i sadrži logiku za listanje i filtriranje komponente koja sadrži popis svih članaka.

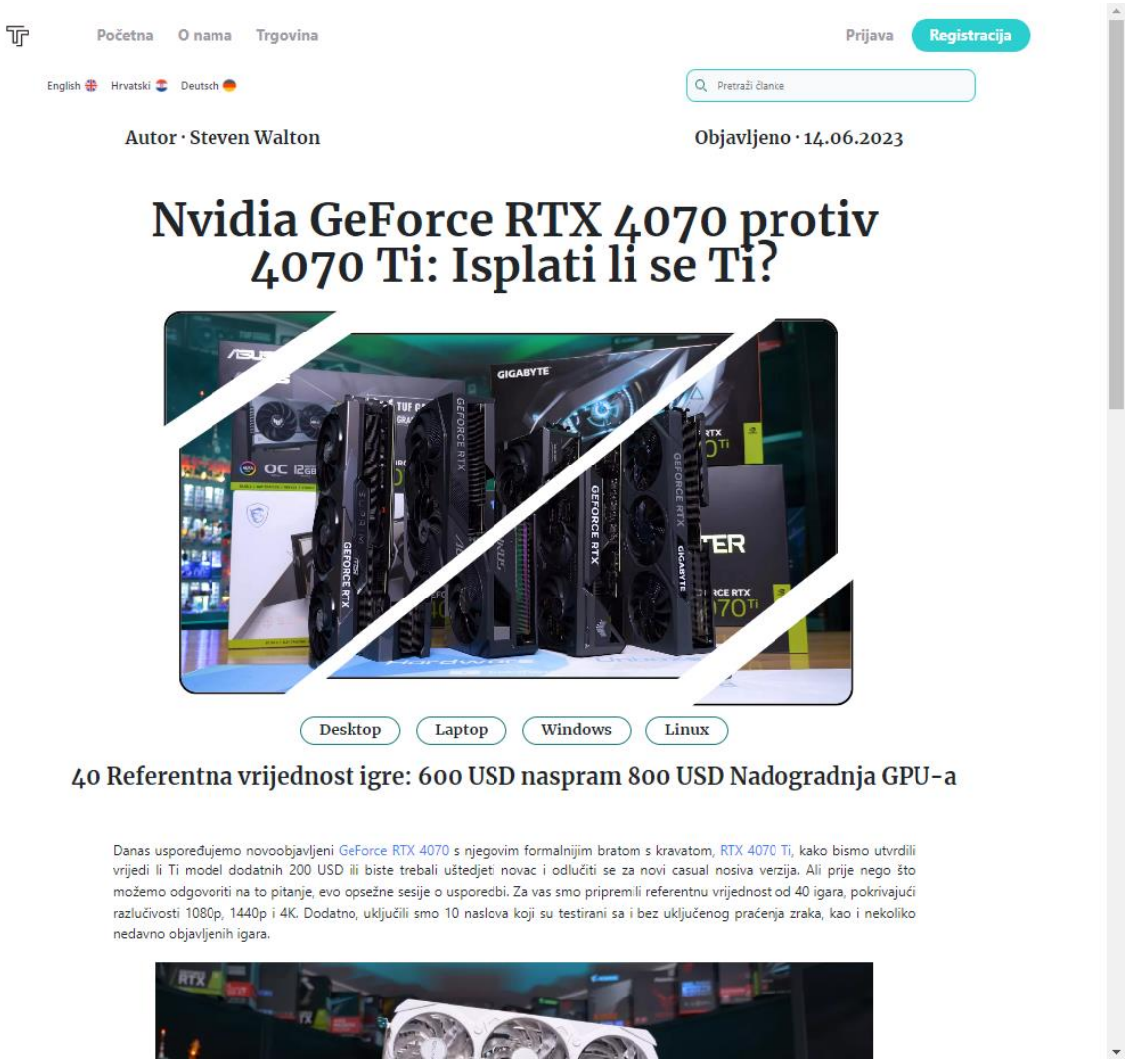
Svi pojedini članci koji se prikazuju na početnoj stranici, moguće ih je otvoriti i korisnicima se otvara nova stranica o detaljima odabranog članka. Stranica o detaljima najprije prikazuje: naziv autora; datum kreiranja članka; naslov članka; naslovnu sliku; popis kategorija članka; podnaslov članka; sadržaj o članku. Naslovna slika je rascijepana na četiri odvojena dijela, koje se spoje u jednu sliku kada korisnici priđu mišem pa slici. Sadržaj članka se dohvaća iz Contentfula kao bogati tekst. Bogati tekst sadrži raznovrsne stilove teksta i dodatno sadrži medijske zapise koje je potrebno generirati na stranici. Dohvaćeni obogaćeni tekst se ne direktno kopira iz Contentfula, nego se za svaki element bogatog teksta (prikazanu na slici 27.) generira HTML programski kod i njegov dizajn pomoću Tailwind CSS-a. Sljedeći programski kod prikazuje način kreiranja dizajna pojedinih elemenata bogatog teksta:

```
const H1 = ({ children }) => (
  <h1 className="text-3xl lg:text-5xl mb-10">{children}</h1>
);
const H2 = ({ children }) => (
  <h2 className="text-2xl lg:text-4xl mb-8">{children}</h2>
);
...
const Blockquote = ({ children }) => (
    <blockquote className="border-l-4 border-cyan italic my-8 pl-8 md:
            pl-12 ml-2 lg:ml-4"> {children} </blockquote>
);
...
const TextCode = ({ children }) => (
    <div>
      <pre className="bg-[#eee] overflow-auto max-w-4xl rounded-md p-4
my-8 mx-12 [counter-reset: linenumber;]">
        <code className="text-left whitespace-pre [word-spacing: normal]
[word-break]">
          {children}
        </code>
      </pre>
```

```
    </div>
  );
...
const AssetsHandler = (node) => {
    const { title, description, file } = node.data.target.fields;
    const mimeType = file.contentType;
    const mimeGroup = mimeType.split("/")[0];
    switch (mimeGroup) {
      case "image":
        return (
          <div className="flex flex-col items-center mb-8">
            <div className="container max-w-screen-lg mx-auto pb-4 flex
justify-center">
              <img alt={title} src={file.url} />
            </div>
            <p className="font-bold text-lg">{description}</p>
          </div>
        );
      case "video":
         ...
```

Prethodni isječak programskog koda prikazuje samo par elemenata. Na kraju se pomoću kuka „BLOCKS“, „INLINES“ i „MARKS“ iz paketa „@contentful/rich-text-types“ i kuke „documentToReactComponents“ iz paketa „@contentful/rich-text-react-renderer“ konfiguriraju da se za dohvaćane elemente iz bogatog teksta koriste prethodno spomenuti programski kod za prikaz elemenata sa prilagođenim dizajnom. slika 37. prikazuje stranicu o detaljima odabranog članka.

Na samome kraju stranice o detaljima članka prikazuju se komentari i ocjene ostalih korisnika. Na samome početku komentara sa desne strani se prikazuje prosječna ocjena članka. Prijavljenim korisnicima se generira komponenta kreiranje novih komentara. Korisnici mogu uklanjati njihove objavljene komentare članaka.

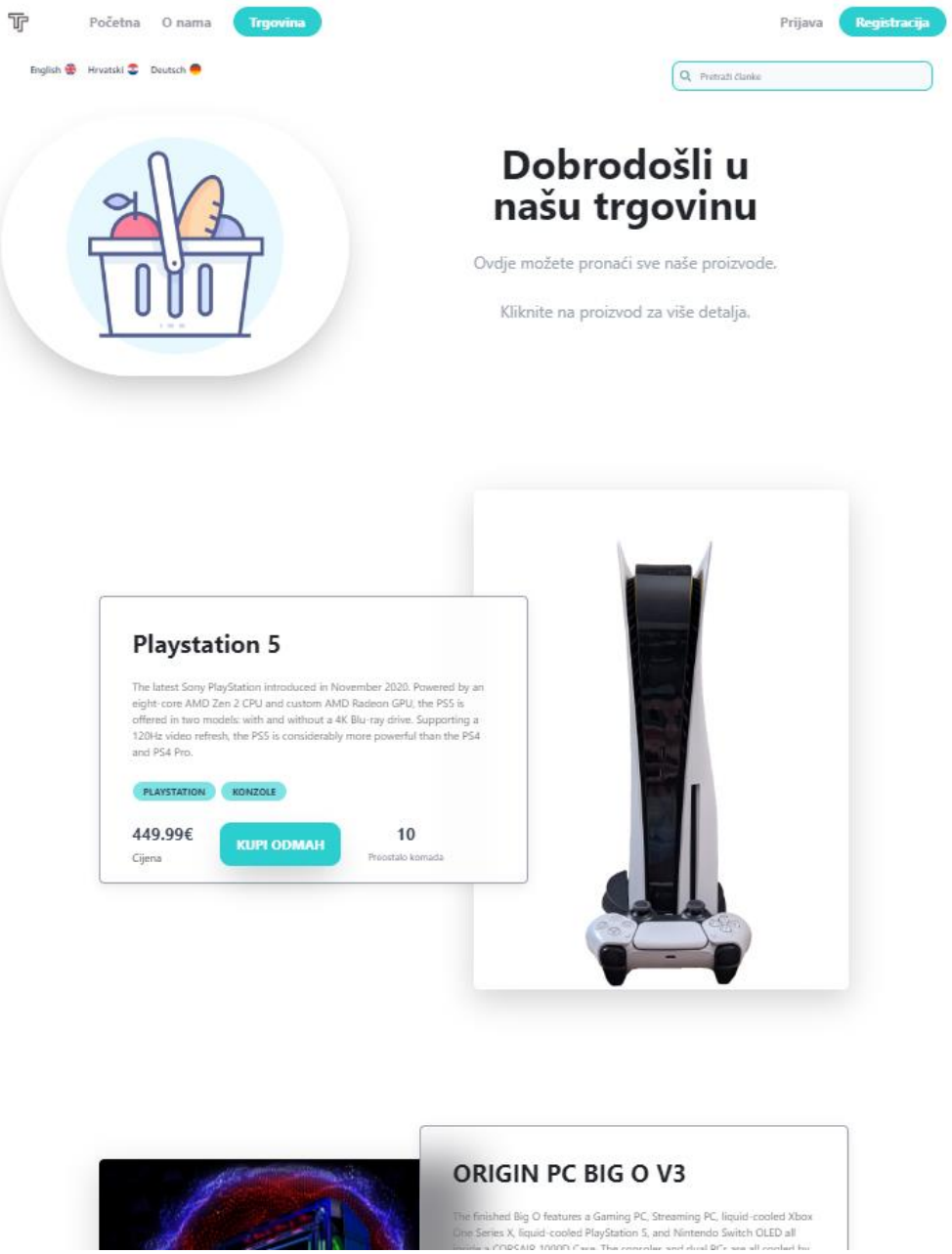


Slika 37. Prikaz stranice o detaljima članka [autorski rad]

5.6.4.2. E-trgovina

Stranica web trgovine korisnicima prvo prikazuje animaciju, koja reprezentira online trgovinu, zatim početni naslov i podnaslov. Nakon naslovne poruke korisnicima se prikazuju popis proizvoda koji su dohvaćeni pomoću prethodno spomenutih zahtjevima koji se nalazi na 87. stranici rada. Sami dizajn prikaza proizvoda je veoma modernog izgleda. Svaki izgled proizvoda je jedinstveni i malo je drugačiji od ostalih prikazanih proizvoda, što ovisi o veličini pohranjene slike u Commercetools sustavu i objavljenim sadržajima proizvoda. Dodatno prikaz proizvoda varira za svaki drugi proizvod. Tekst prvog proizvoda će se nalaziti na lijevo strani, a slika proizvoda na desnoj strani. Tekst drugog proizvoda će se nalaziti obrnuto na desnoj strani, a slika drugog proizvoda se nalazi na lijevoj strani. Treći proizvod se ponovno prikazuje kao prvi itd. Prijavljeni korisnici mogu kupovati na brzi način proizvode i na malo sporiji način. Brzi način je kada korisnici kliknu na gumb „kupi odmah“, prilikom listanja svih proizvoda, gdje se samo jedan proizvod doda u košaricu korisnika, a sporiji način je kada korisnik otvori

stranicu o detaljima proizvoda i potom doda željenu količinu proizvoda u košaricu. Slika 38. prikazuje web trgovinu aplikacije.



Slika 38. Prikaz e-trgovine web aplikacije [autorski rad]

Tekst svakog proizvoda sadrži: naziv; opis; kategorije; cijenu; količinu zalihe. Svaki proizvod korisnici mogu otvoriti kako bi vidjeli detalje i recenzije od otvorenog proizvoda.

Prilikom dohvaćanja podataka iz Commercetoolsa bilo je potrebno dobro razmotriti i proučiti dohvaćene JSON podatke. Pošto su dohvaćeni podaci bili zapakirani u polja podataka, gdje svako polje sadrži svoja polja podataka i ponovno svako polje polja sadrži svoje polja

podataka itd. Primjerice sljedeći isječak programskog koda prikazuje mapiranje dohvaćenih podataka iz Commercetoolsa u objekt podataka koji je mnogo jednostavniji za koristiti:

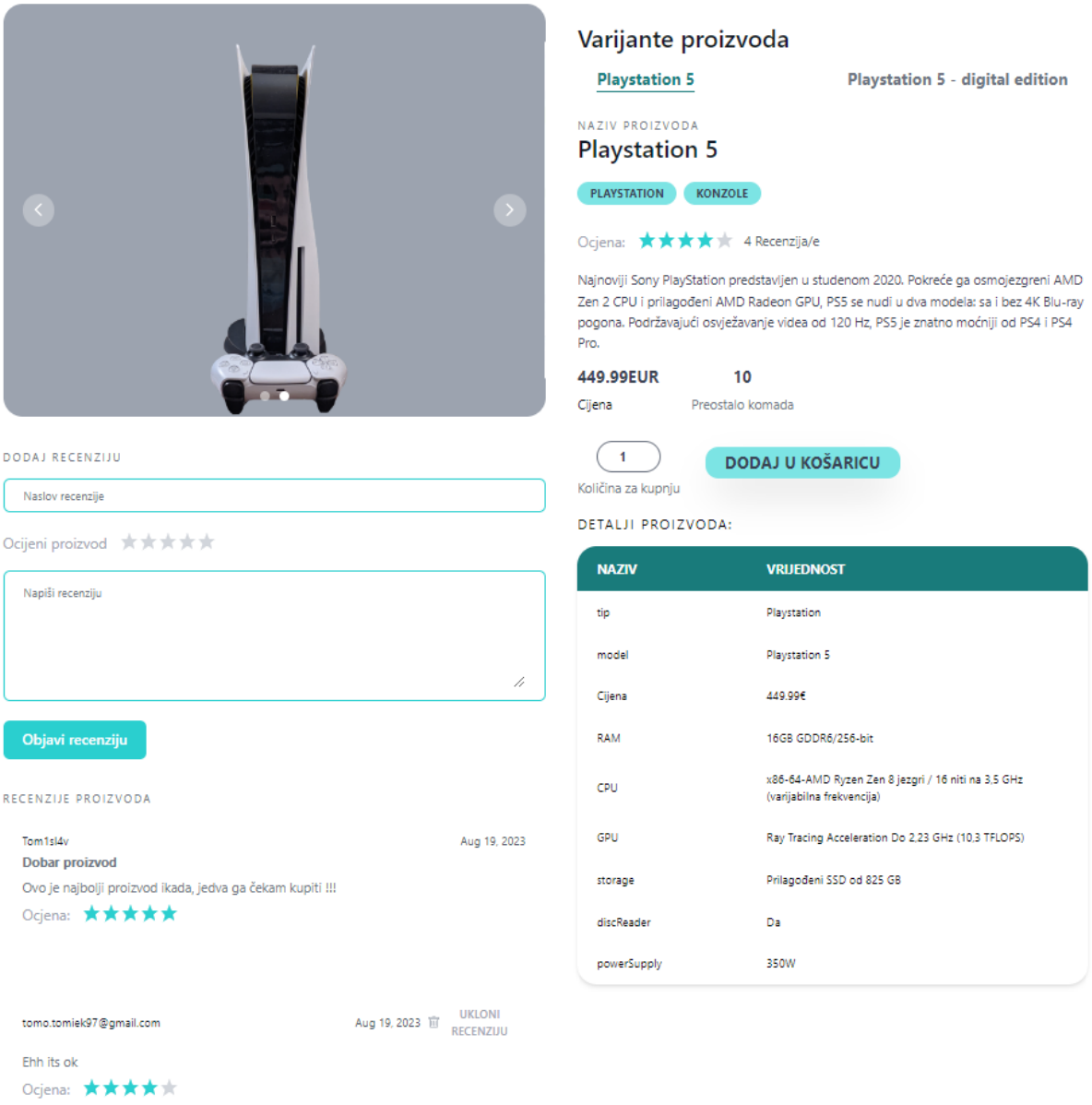
```
return {
      id: productId,
      variantId: productData.masterData.current.masterVariant.id,
      name: productData.masterData.current.name[languageStorage],
      price:
        productData.masterData.current.masterVariant.prices[0].value
          .centAmount / 100,
      currency:
        productData.masterData.current.masterVariant.prices[0].value
          .currencyCode,
      description: description,
      imageUrl:
productData.masterData.current.masterVariant.images[0].url,
      otherImages: imageList,
      category: productCategories,
      sku: productData.masterData.current.masterVariant.sku,
      quantity: getQuantity(
    productData.masterData.current.masterVariant.availability.channels),
      quantityInCart: 0,
      richText: getAtributsList(
        productData.masterData.current.masterVariant.attributes),
      varients: varientsList};
```

Prethodni isječak programskog koda prikazuje mapiranje dohvaćenih podataka za glavnu varijaciju (eng. *master variant*) proizvoda koji će se koristiti kod prikaza svih proizvoda web trgovine. Dodatno svaki proizvod može posjedovati svoje zasebne varijacije koje se mapiraju na sličan način kao i glavna varijacija proizvoda, samo što se iterira kroz svaku varijaciju i kreiraju se polja varijacija koje sadrže iste nazive varijabli kao i glavna varijacija proizvoda. Dodatno je bilo potrebno pripaziti na nazive varijabli dohvaćenih varijacija proizvoda, pošto su njihovi nazivi bilo malo drugačiji nego kod glavne varijacije. Prethodno su bile kreirane varijable koje sadrže opis proizvoda i kategorije proizvoda, koje su se koristile kod svih varijacija. Opis i popis kategorije je jednaki za svaku varijaciju proizvoda.

Dodatno je kreirana jedna funkcija za mapiranje količine zalihe varijacije i popis atributa, pošto su dohvaćeni podaci za količinu i popis atributa bili zapakirani na jednaki način za svaku varijaciju proizvoda. Kao što je već prije bilo rečeno popis kategorija za svaki proizvod sadrži identifikacijske brojeve kod dohvaćenih informacija o proizvodima, pa se iz navedenog razloga mora vršiti drugi zahtjev prema Commercetoolsu. Zahtjev za dohvaćanje kategorija je

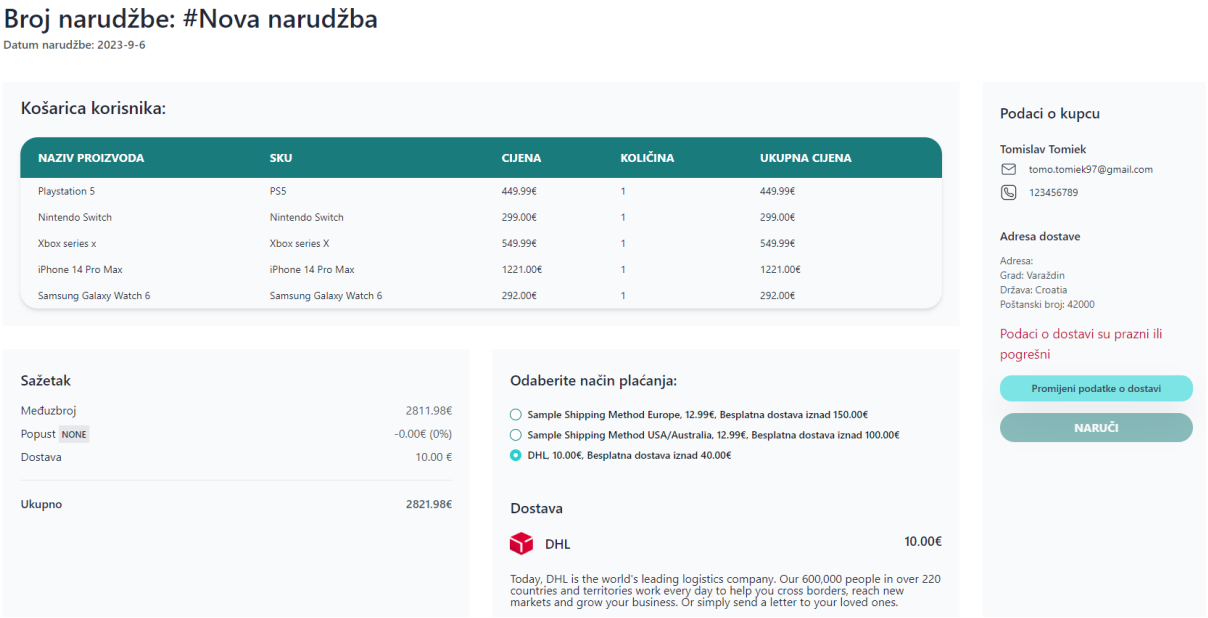
objašnjeni na 88. stranici rada i on je potreban kako bi se korisnicima prikazivali stvarni nazivi kategorija umjesto dohvaćeni ID-jevi kategorija.

Stranica o detaljima proizvoda sadrži veliku količinu informacija koje se korisnicima prikazuju. Stranicu je moguće podijeliti u pet manjih segmenta: odabir varijacije proizvoda; opće informacije o varijaciji proizvoda; specifikacije varijacije proizvoda; galerija slika varijacije; recenzije korisnika o otvorenom proizvodu. Pošto slika govori tisuću riječi, slika 39. objašnjava sve navedene segmente za stranicu o detaljima proizvoda.



Slika 39. Prikaz stranice o detaljima proizvoda [autorski rad]

Sljedeće stranice koje su povezane sa web trgovinom su stranice: korisnička košarica; stranica za plaćanje narudžbe korisnika; povijest naručivanja korisnika. Stranica za košaricu sadrži popis svih proizvoda koje je korisnik dodao u košaricu. Osim pregleda dodanih proizvoda u košaricu, korisnicima se pruža dodatna mogućnost za promjenu željenih količina dodanih proizvoda. Neželjene proizvode je moguće ukloniti iz košarice. Nakon što je korisnik potvrdio željene proizvode i njihove količine, potrebno je nastaviti kupnju pritiskom na gumb nastavi na naplatu. Potom se otvara nova stranica za plaćanje narudžbe, gdje korisnici postavljaju željenu metodu plaćanja i informacije o dostavi narudžbe. Prikazuje se informacije o kupcu, koje je bilo moguće postaviti prilikom: procesa registracije; naknadno pomoću stranice o postavkama korisničkog profila; prilikom navedene stranice o plaćanju narudžbe, pritiskom na gumb za promjenu informacija o dostavi. Ako pojedini podaci za dostavu nisu ispunjeni zabranjuje se naručivanje i korisnika se obavještava sa odgovarajućom porukom. Kada korisnici odaberu željenog dostavljača i postave ispravne informacije o dostavi, moguće je naručiti proizvode pritiskom na gumb naruči. Nakon što korisnik naruči proizvode, šalje se zahtjev prema back-end serveru, gdje se kreira naručena narudžba korisnika. Korisnika se preusmjerava na stranicu za povijest kupovine. Sljedeća slika 40. prikazuje stranicu za naplatu narudžbe korisnika.



Slika 40. Prikaz stranice za naplatu narudžbe korisnika [autorski rad]

Stranica za povijest kupovine korisnicima prikazuje popis svih naručenih narudžbi. Narudžbe se prikazuju pomoću tablice. Tablica prikazuje pet narudžbi koje je moguće listati. Svaka narudžba posjeduje svoj status, koji predstavlja trenutno stanje narudžbe. Stanje

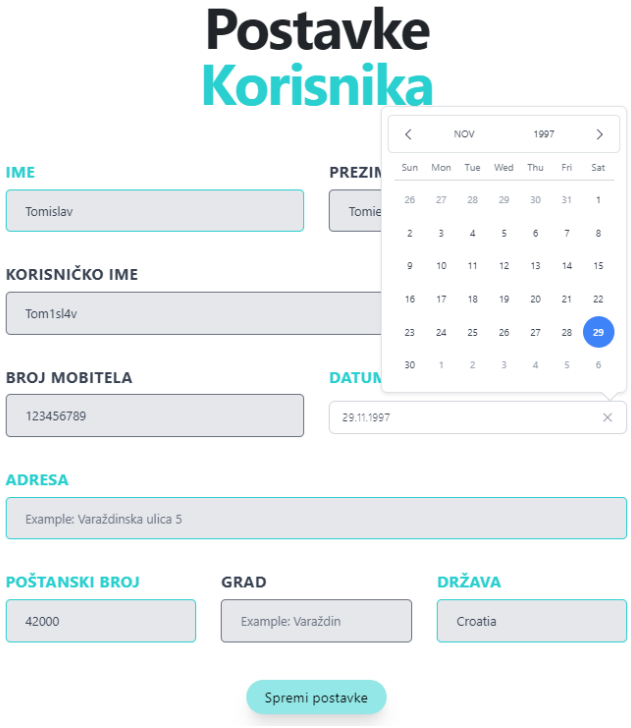
narudžbe može biti: procesiranje narudžbe (eng. processing), gdje se čeka potvrda od strane admina ili osobe koja je zadužena za pakiranje proizvoda i slanje proizvoda korisnicima; u procesu dostavljanja narudžbe (eng. *delivering*); dostavljeno (eng. *delivered*); otkazano (eng. *canceled*). Korisnicima se pruža mogućnost detaljnog pregleda narudžbi i otkazivanje narudžbi ako se narudžba nije procesirala.

Stranica detaljnog pregleda narudžbi se prikazuje na isti način kao i prethodna stranica za plaćanje narudžbi (slika 40.), samo što se umjesto mogućnosti za odabira željenog dostavljača, gumba za naručivanje i gumba za izmjenu informacija o dostavi prikazuje gumb za ispis narudžbe.

5.6.4.3. Ostale stranice aplikacije

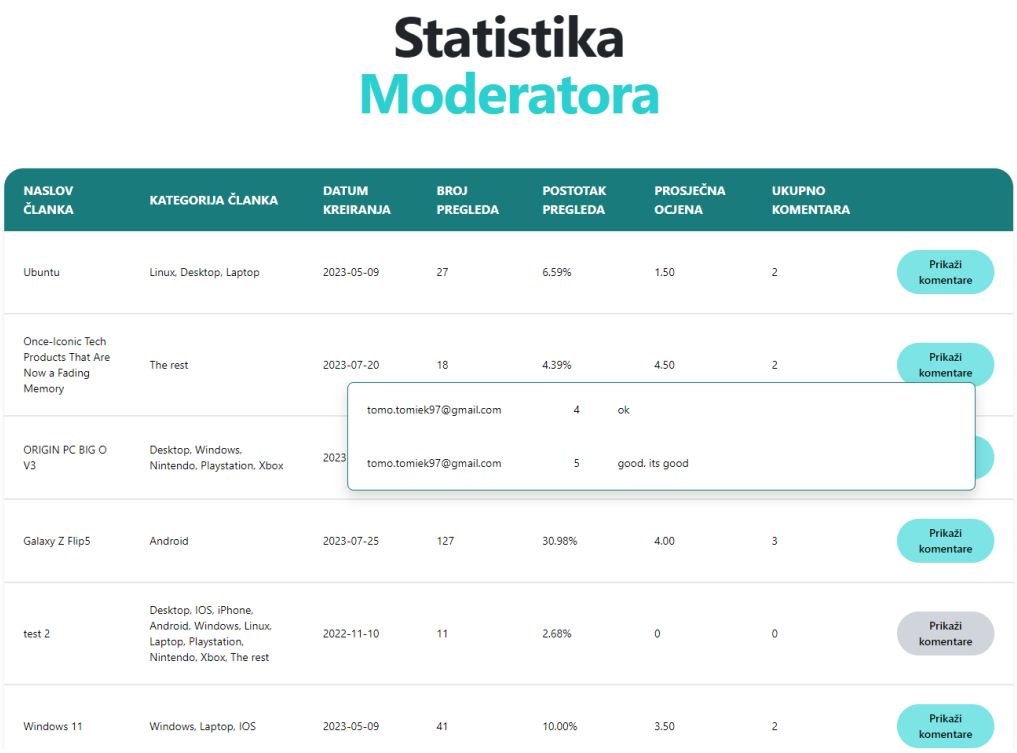
Stranica o nama je veoma jednostavna i pruža informacije o misiji i viziji web aplikacije. Stranica je izrazito privlačna, što je realizirano pomoću mnogih animacija i privlačnim slikama.

Prijavljeni korisnici posjeduju mogućnost izmjene njihovog korisničkog profila, koja se nalazi na stranici korisničkog profila. Korisnici trenutno mogu mijenjati sljedeće vrijednosti: ime; prezime; korisničko ime; broj mobitela; datum rođenja; adresu; poštanski broj; grad; državu. Informacije o dostavi nije potrebno postaviti prilikom procesa registracije, ali ih je potrebno postaviti ako korisnici naručuju proizvode iz web trgovine. Slika 41. prikazuje stranicu korisničkog profila.



Slika 41. Prikaz stranice korisničkog profila [autorski rad]

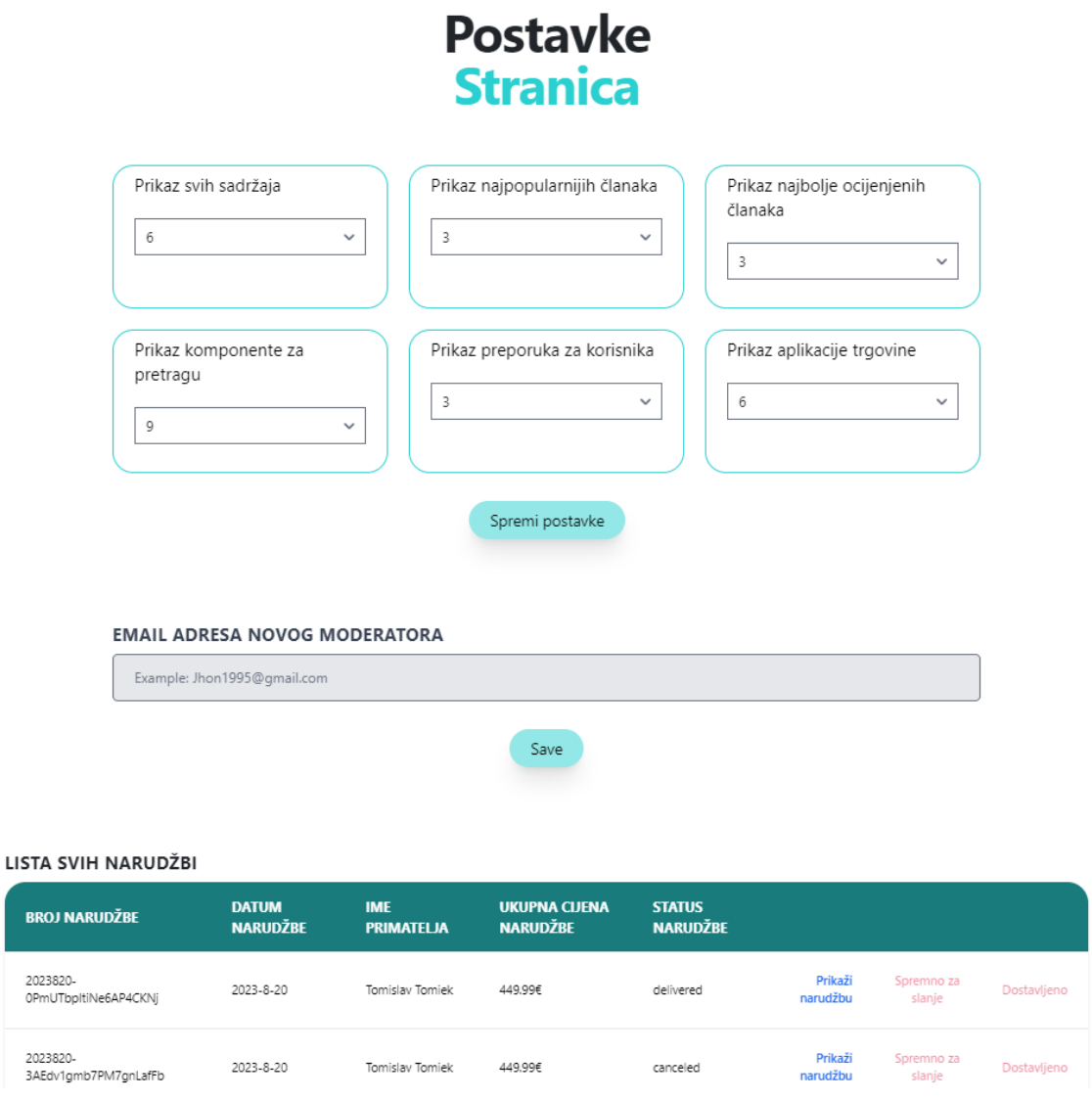
Sljedeća stranica je statistika članaka, namijenjena za kreatore sadržaja / članaka bloga. Stranica sadrži popis svih članaka koji su kreirali prijavljeni moderatori. Popis članaka se nalazi u tablici, gdje se prikazuje: naziv članka; kategorije članka; datum kreiranja članka; broj posječenosti članka; postotak pregleda članka na temelju ukupnom broju pregleda svih članaka; prosječnu ocjenu; količinu komentara; gumb za prikaz komentara određenog članka omogućuje moderatorima da vide sve komentare, e-mail adrese osoba koje su napisale komentare i ocjene. Slika 42 prikazuje stranicu za statistiku moderatora.



Slika 42. Prikaz stranice za statistiku moderatora [autorski rad]

Posljednja stranica je stranica za postavke web aplikacije, namijenjena za administrativne korisnike web aplikacije. Adminima se prvo prikazuje mogućnost za promjenu dizajna za prikaz članaka bloga i proizvode web trgovine. Moguće je promijeniti količinu prikazanih članaka i proizvoda za straničenje, posebno za svaki od sljedećih prikaza: popis svih članaka; najpopularniji članci; najbolje ocijenjeni članci; popis članaka za preporuku prijavljenih korisnika; popis članaka koji su pronađeni putem tražilice; popis svih proizvoda. Ako se iz bilo kojeg razloga ne uspiju dohvatiti postavke web aplikacije iz baze podataka prilikom prvog posjeta web stranice, postavljaju se zadane postavke stranice. Zadane postavke za popis svih članaka je postavljena na šest, popisi najpopularnijih, najbolje ocijenjenih i preporučeni članaka je postavljena na tri, popis za pretragu članaka je postavljena vrijednost devet i popis svih proizvoda je postavljena na četiri. Administrativni korisnici

posjeduju mogućnost dodavanja novih moderatora web aplikacije, gdje je potrebno unijeti ispravni e-mail korisnika. Ako je pogrešan ili nepostojeći uneseni e-mail korisnika, obavještava se admina sa odgovarajućom porukom. Posljednja stvar koja se prikazuje administrativnim korisnicima je lista narudžbi. Lista narudžbi se nalazi u tablici, koja prikazuje narudžbe u deset redaka. Tablica je implementirana pomoću straničenja, kako bi se lijepo prikazao popis svih narudžbi. Admini mogu svaku narudžbu: pregledati; zaprimiti, gdje se simulira pakiranje proizvoda i pošilja se narudžbe korisnicima pritiskom na gumb „Spremno za slanje“; postaviti stanje narudžbe da je dostavljeno pritiskom na gumb „Dostavljeno“, gdje se simulira da je dostavljač dostavio narudžbu korisnicima. Dodatna napomena admini ne mogu direktno postaviti stanje narudžbe na dostavljeno, nego je prvo potrebno narudžbu poslati i potom je moguće postaviti status narudžbe na dostavljeno. Otkazane narudžbe admini mogu samo pregledavati. Slika 43. prikazuje administrativnu stranicu postavki web aplikacije.



Slika 43. Prikaz administrativne stranice za postavke web aplikacije [autorski rad]

6. Zaključak

Općenito, bilo koji tip sustava za upravljanje sadržajem ima iznimno važnu i korisnu ulogu, posebice u slučaju poduzeća koja raspolažu velikim količinama podataka koje treba prikazati korisnicima. Sustavi za upravljanje sadržajem omogućuju korisnicima da odvoje sadržaj od ostatka programske logike. Sadržaj se kreira na jednom mjestu pomoću odabranog sustava za upravljanje sadržajem i nakon toga se jednostavno objavljuje. Objavljeni sadržaj lako je dohvatiti i prikazati na raznim IoT uređajima za koje se razvija aplikacija. Sustavi za upravljanje sadržajem pružaju korisnicima mogućnost upravljanja stvorenim sadržajem, omogućuju verzioniranje sadržaja i/ili cijelih sustava za upravljanje sadržajem te upravljanje svim korisničkim računima zaposlenika i njihovim ulogama. Većina sustava za upravljanje sadržajem također uključuje lokalizacijske mogućnosti, što omogućuje stvaranje sadržaja na različitim jezicima. Osim toga, ovi sustavi pružaju razne druge funkcionalnosti koje korisnicima olakšavaju učinkovito upravljanje i prikaz sadržaja.

Izbor vrste sustava za upravljanje sadržajem ovisi o specifičnim potrebama poduzeća. Ako je potrebna visoka fleksibilnost sustava, često se preferiraju sustavi za upravljanje sadržajem bez grafičkih sučelja. S druge strane, poduzeća koja se bave prodajom proizvoda često koriste sustave za upravljanje sadržajem e-trgovina. Ako je potreban sustav koji pruža gotova rješenja za korisnička sučelja i istovremeno pruža funkcionalnosti CMS sustava, tada se preferiraju tradicionalni sustavi za upravljanje sadržajem.

Nažalost, ne postoji univerzalno najbolje rješenje za sustave za upravljanje sadržajem bez grafičkih sučelja koje bi zadovoljilo sve potrebe korisnika. Svaki alat ima svoje prednosti i nedostatke, i pojedini alati bolje implementiraju određene funkcionalnosti CMS sustava u usporedbi s konkurencijom, dok za druge aspekte mogu biti manje adekvatni.

Primjerice, za korisnike koji zahtijevaju potpunu fleksibilnost u izgradnji CMS sustava, alat poput Sanity-a može biti optimalan jer omogućuje potpunu kontrolu nad kreiranjem modela sadržaja, iako zahtijeva dodatno znanje i vrijeme uloženo u razvoj. S druge strane, ako korisnicima treba manje fleksibilno rješenje, Contentful može biti dobar izbor jer pruža besplatno rješenje s jednostavnim sučeljem i mnogo gotovih funkcionalnosti koje se brzo mogu postaviti.

Kada je riječ o open-source rješenjima, Directus se izdvaja kao jedan od najboljih alata za definiranje modela sadržaja, s modernim izgledom i širokim spektrom funkcionalnosti za promjenu izgleda sustava. Međutim, važno je napomenuti da Directus nema internu tržnicu za integraciju funkcionalnosti trećih strana, što je dostupno kod ostalih konkurentnih alata. Također, treba uzeti u obzir da su posljednje verzije Directusa bile nestabilne te se nisu

preporučivale za produkciju. Rješenja otvorenog koda koja pružaju sustave za upravljanje sadržajem bez grafičkog sučelja za e-trgovinu generalno se ne preporučuju, jer obično dolaze s ograničenim brojem funkcionalnosti i teško je zadovoljiti potrebe korisnika.

Međutim, komercijalna rješenja predstavljaju drugačiju priču. CMS sustavi bez grafičkog sučelja za e-trgovinu pružaju korisnicima sve funkcionalnosti koje tradicionalni headless CMS sustavi nude, a dodatno imaju niz naprednih funkcionalnosti kojima mogu privući veći broj kupaca i unaprijediti poslovanje. Neke od tih funkcionalnosti uključuju: napredne postavke tražilice web trgovine; mogućnost raznih metoda plaćanja i dostave; jednostavno uvođenje popusta; generiranje raznih statistika o poslovanju. Između komercijalnih rješenja, Commercetools se izdvaja kao alat namijenjen srednjim i velikim poduzećima. Commercetools pruža napredne funkcionalnosti za kreiranje i upravljanje proizvodima, što omogućuje korisnicima da zadovolje i najspecifičnije potrebe svojih kupaca. Osim toga, Commercetools omogućuje precizno ograničavanje pristupa prilikom kreiranja API-a čime se postiže veća razina sigurnosti samog sustava te povjerljivih podataka kako kompanije tako i njezinih klijenata.

Konačan izbor alata ovisi o specifičnim potrebama i prioritetima korisnika pri čemu treba uzeti u obzir funkcionalnosti, dostupnost, cijenu i potrebno znanje za implementaciju odabranog sustava za upravljanje sadržajem.

Popis literature

[1] D. Barker, Web Content Management, OReilly Media, INC, 2016.

[2] P. Lončar, »Analiza modernih sustava za upravljanje sadržajem na webu,« 25. srpanj 2022. [Mrežno]. Available: https://repozitorij.oss.unist.hr/islandora/object/ossst:1596. [Pokušaj pristupa 22. lipnja 2023.].

[3] P. Browning i M. Lowndes, »Content Management Systems,« JISC TechWatch, 2001.

[4] K. Halvorson i M. Rach, Content Strategy for the Web, 2nd Edition, New Riders, 2012.

[5] B. Boiko, Content Management Bible, 2nd Edition, Wiley, 2005.

[6] C. Benevolo i S. Negri, »Evaluation of Content Management Systems (CMS): a Supply Analysis,« Academic Conferences Ltd, Genova, 2007.

[7] S. Davis, »The pros and cons of using a CMS to build your website,« Flicker Leap, 24. ožujak 2017. [Mrežno]. Available: https://flickerleap.com/pros-cons-using-cms-build-website/. [Pokušaj pristupa 4. srpnja 2023.].

[8] ThemeSelection, »The Best 10+ Open Source Headless CMS 2023,« 10. kolovoz 2022. [Mrežno]. Available: https://dev.to/theme_selection/the-best-10-open-source-headless-cms-40hf. [Pokušaj pristupa 6. srpnja 2023.].

[9] D. Parasad Acharya, »11 Best Open-Source Headless CMS to Try for Your Next Application,« Geekflare, 28. travanj 2023. [Mrežno]. Available: https://geekflare.com/open-source-headless-cms/. [Pokušaj pristupa 6. srpnja 2023.].

[10] P. Isabel Signo, »13 Open Source Headless CMS You Should Consider for Your Next Project,« OSSPH, 17. studeni 2022. [Mrežno]. Available: https://blog.ossph.org/open-source-headless-cms/. [Pokušaj pristupa 6. srpnja 2023.].

[11] A. Verma, »Headless CMS and Headless Commerce Statistics in 2023,« Experro, 6. travanj 2023. [Mrežno]. Available: https://www.experro.com/blog/headless-cms-headless-commerce-statistics/. [Pokušaj pristupa 6. srpnja 2023.].

[12] »A short introduction to Sanity.io,« Sanity, 7. prosinac 2022. [Mrežno]. Available: https://www.sanity.io/docs/a-short-introduction-to-sanity-io. [Pokušaj pristupa 6. srpnja 2023.].

[13] K. Paters, »Sanity Features,« G2, 2023. [Mrežno]. Available: https://www.g2.com/products/sanity/features. [Pokušaj pristupa 6. srpnja 2023.].

[14] R. Sharma, »Conetentful,« G2, 2023. [Mrežno]. Available: https://www.g2.com/products/contentful/features. [Pokušaj pristupa 1. kolovoza 2023.].

[15] G2, »Compare Contentful and Sanity,« G2 Team, [Mrežno]. Available: https://www.g2.com/compare/contentful-vs-sanity. [Pokušaj pristupa 15. kolovoza 2023.].

[16] M. Almeida, »Strapi,« G2, 2023. [Mrežno]. Available: https://www.g2.com/products/strapi/features. [Pokušaj pristupa 2. kolovoza 2023.].

[17] »Directus Key Features,« Monospace Inc, 2023. [Mrežno]. Available: https://directus.io/features/. [Pokušaj pristupa 3. kolovoza 2023.].

[18] G2, »Compare Directus and Strapi,« G2 Team, [Mrežno]. Available: https://www.g2.com/compare/directus-vs-strapi. [Pokušaj pristupa 15 kolovoza 2023.].

[19] Restack, »Directus vs. Strapi – Comparison Headless CMS,« Restack team, [Mrežno]. Available: https://www.restack.io/docs/directus-vs-strapi. [Pokušaj pristupa 15. kolovoza 2023.].

[20] E. N. Abdullah, S. Ahmad, M. Ismail i N. M. Diah, »Evaluating E-commerce Website Content Management System in Assisting Usability Issues,« 3. studeni 2021. [Mrežno].

Available: https://ieeexplore.ieee.org/abstract/document/9509991. [Pokušaj pristupa 3. kolovoza 2023.].

[21] V. Team, »The key features of the commercetools platform,« Vaimo, 2023. [Mrežno]. Available: https://www.vaimo.com/expertise/commercetools/. [Pokušaj pristupa 3. kolovoza 2023.].

[22] M. Rea, »Commercetools,« G2, 2023. [Mrežno]. Available: https://www.g2.com/products/commercetools/pricing. [Pokušaj pristupa 4. kolovoza 2023.].

[23] A. Lee, »Shopify,« G2, 2023. [Mrežno]. Available: https://www.g2.com/products/shopify/features. [Pokušaj pristupa 6. kolovoza 2023.].

[24] Shopify, »Shopify pricing,« [Mrežno]. Available: https://www.shopify.com/pricing. [Pokušaj pristupa 6. kolovoza 2023.].

[25] G2, »Compare Shopify and commercetools,« G2 Team, [Mrežno]. Available: https://www.g2.com/compare/shopify-vs-commercetools. [Pokušaj pristupa 6. kolovoza 2023.].

[26] Gartner Peer Insights, »commercetools vs Shopify,« Gartner Peer Insights Team, [Mrežno]. Available: https://www.gartner.com/reviews/market/digital-commerce/compare/commercetools-vs-shopify. [Pokušaj pristupa 6. kolovoza 2023.].

[27] SelectHub, »Shopify vs commercetools,« 24 srpanj 2023. [Mrežno]. Available: https://www.selecthub.com/ecommerce-platforms/shopify-vs-commercetools/. [Pokušaj pristupa 6. kolovoza 2023.].

[28] A. Bellanger, »PrestaShop,« G2, [Mrežno]. Available: https://www.g2.com/products/prestashop/features. [Pokušaj pristupa 11. kolovoza 2023.].

[29] A. Tsang, »OpenCart,« G2, [Mrežno]. Available: https://www.g2.com/products/opencart/reviews#details. [Pokušaj pristupa 8. kolovoza 2023.].

[30] G2, »Compare OpenCart and PrestaShop,« G2 Team, [Mrežno]. Available: https://www.g2.com/compare/opencart-vs-prestashop. [Pokušaj pristupa 15. kolovoza 2023.].

[31] »What Are RESTful Web Services?,« Oracle and/or its affiliates, 2013. [Mrežno]. Available: https://docs.oracle.com/javaee/6/tutorial/doc/gijqy.html. [Pokušaj pristupa 4. srpnja 2023.].

[32] GraphQL zajednica, »GraphQL,« Listopad 2021. [Mrežno]. Available: https://spec.graphql.org/October2021/. [Pokušaj pristupa 15. kolovoza 2023.].

[33] E. Wohlgethan, Supporting Web Development Decisions by, Hamburg: Faculty of Engineering and Computer Science, 2018..

Popis slika

Popis tablica

Prilozi

Link za GitHub repozitorij front-end dijela aplikacije:
https://github.com/Tom1sl4v97/Diplosmi_rad-frontend

Link za GitHub repozitorij back-end dijela aplikacije:
https://github.com/Tom1sl4v97/Diplomski_rad-backend