Razvoj web aplikacija korištenjem okvira Hugo

Rošić, Antonio

Undergraduate thesis / Završni rad

2021

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:211:250900

Rights / Prava: Attribution-NonCommercial 3.0 Unported/Imenovanje-Nekomercijalno 3.0

Download date / Datum preuzimanja: **2024-12-01**

SVEUČILIŠTE U ZAGREBU
FAKULTET ORGANIZACIJE I INFORMATIKE
VARAŽDIN

Repository / Repozitorij:

Faculty of Organization and Informatics - Digital Repository

zir.nsk.hr

dabar
DIGITALNI AKADEMSKI ARHIVI I REPOZITORIJI

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Antonio Rošić

RAZVOJ WEB APLIKACIJA KORIŠTENJEM OKVIRA HUGO

ZAVRŠNI RAD

Varaždin, 2021.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Antonio Rošić

Matični broj: 0016078811

Studij: Primjena informacijske tehnologije u poslovanju

RAZVOJ WEB APLIKACIJA KORIŠTENJEM OKVIRA HUGO

ZAVRŠNI RAD

Mentor:

dr. sc. Matija Novak

Varaždin, travanj 2021.

Antonio Rošić

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

U radu se kao glavna tema obrađuje okvir Hugo za izradu Web aplikacije. Na početku se opisuje što su to Web aplikacije i Web stranice te koje sve vrste postoje, a nakon toga prezentirane su razlike, prednosti i nedostaci između njih. U nastavku se spominju i drugi okviri te je dana kratka usporedba s Hugo okvirom. Ukratko se obrađuje i jezik oznaka(*eng. markup*) kao i Netlify. Netlify je platforma koja se koristi za posluživanje (*eng. hosting*) web aplikacija kao i statičkih web stranica. Nakon toga slijedi detaljniji opis Hugo okvira. U drugom dijelu rada opisuje se postupak izrade Web aplikacije pomoću okvira Hugo popraćeno slikama i kodom koji se koristi u izradi. Na kraju se opisuju prednosti i nedostaci okvira kao i zaključak na osnovi rada sa samim okvirom.

Ključne riječi: Web, Web aplikacija, Web stranica, Generatori statičkih Web stranica, Hugo okvir, GitHub, Netlify, Markd

Sadržaj

1. Uvod 1

2. Razvoj Web aplikacija 2

2.1 Što su Web Aplikacije? 2

2.2 Kako web aplikacije rade? 2

2.3 Aplikacija ili Stranica 3

2.4 Prednosti i nedostaci web aplikacija 4

3. Vrste Web aplikacija 5

4. Generatori statičkih web aplikacija 6

4.1 Razlika između statičkih generatora i sustava za upravljanjem sadržaja 7

4.2 Jekyll 8

4.3 Gatsby 8

4.4 Hugo 9

4.5 Usporedba Generatora 9

5. Markdown i JAMstack 11

5.1 Markdown 11

5.2 JAMstack 12

6. Hugo Okvir 14

6.1 Konfiguracija i struktura 14

6.2 Teme i predlošci 16

6.3 Hugo moduli 17

6.4 Smještaj i implementacija 18

6.4.1 GitHub 18

6.4.2 Netlify 19

7. Čitanje podataka u okviru Hugo 20

8. Kreiranje višejezičnih aplikacija u okviru Hugo 21

9. Izgradnja aplikacije u okviru Hugo 23

9.1 Instalacija Hugo okvira i kreiranje stranice 23

9.2 JavaScript dio 25

9.3 HTML dio 27

9.4 WordPress 29

9.5 Testiranje 30

10. Prednosti i nedostaci u okviru Hugo 39
11. Zaključak 40
Popis literature 41
Popis slika 44
Popis tablica 45
Prilozi 46

1. Uvod

Danas postoji puno okvira za razvoj web aplikacija, no relativno malo okvira koji su generatori statičnih web stranica kao što je Hugo. Prije nego što se opišu okviri potrebno je objasniti što je to web aplikacija.

Jedna od definicija web aplikacija jest: „Web aplikacija je računalni program koji koristi web preglednike i web tehnologiju za izvršavanje zadataka putem Interneta.“(Gibb, 2016).

Da bi web aplikaciju učinili dostupnom korisniku, moramo imati poslužitelja (*eng. server*) na koji se smjeti aplikacija da bi svaki korisnik mogao doći do nje i koristiti je. Poslužitelj je program ili uređaj koji ima funkcionalnosti za aplikacije, programe, stranice te omogućava dijeljenja sadržaja i podataka kao i usluživanje korisnika. Server se koristi zbog toga što može uslužiti veliki broj klijenata u isto vrijeme. Razlika između web aplikacija izgrađenih pomoću statičkih generatora i tradicionalnih programa je u korištenju poslužitelj. Statički generatori koriste poslužitelj da aplikaciju svakom korisniku dostave identičnu i podaci se spremaju na pregledniku ili samom uređaju korisnika, te pomoću programa dohvaća podatke s trećih stranica i prikazuje, a tradicionalni način korištenja je da se na poslužitelju spremi aplikacija kao i svi podaci koje korisnik unosi i koristi te prikazuje podatke i sadržaj koji je spremljen u bazi.

U nastavku ovoga rada, nakon predstavljanja pojma web aplikacija, slijedit će obrada generatora web aplikacija. Predstavit će se njihove općenite značajke i određene vrste te izvršiti usporedba s Hugom.

Glavni dio ovoga rada je okvir Hugo te ćemo njega najviše obraditi i objasniti. Dio rada će se usmjeriti na obradu Hugo okvira, njegova načina rada i određenih mogućnosti te kako se s njime realizira projekt i na koji način

Praktični dio čini drugi dio rada u kojem će se napraviti web aplikaciju pomoću okvira Hugo. Sam postupak izrade i realizacije projekta će se detaljno opisati i objasniti. Također, drugi dio sadržava i usporedbu učitavanja web aplikacije realizirane preko okvira Hugo kao i preko WordPress-a. Rad sadržava i slike, kao i sam kod koji se koristiti u projektu.

2. Razvoj Web aplikacija

U uvodnom dijelu rada dana je kratka definicija web aplikacija, a u ovom poglavlju dan je detaljniji opis što je to web aplikacija, objašnjeno je kako web aplikacije rade te koja je razlika između web aplikacije i web stranice.

2.1 Što su Web Aplikacije?

„Web aplikacija, koja se često naziva i web app (aplikacija), interaktivni je računalni program izgrađen s web tehnologijama (eng*. HyperText Markup Language - HTML*, eng*. Cascading Style Sheets - CSS* i JavaScript), koji pohranjuje (Baza podataka, datoteke) i manipulira podacima (eng*. Create, Read, Update,Delete - CRUD*), a koristi ga tim ili jedan korisnik za obavljanje zadataka putem Interneta. CRUD je popularna kratica i u središtu je razvoja web aplikacija. Kratica je stvaranje, čitanje, ažuriranje i brisanje. Web aplikacijama se pristupa putem web preglednika, kao što je Google Chrome, i često uključuju mehanizam za prijavu / registraciju.“ (Johnston, 2020).

Ovo je jedna od definicija web aplikacije, postoje male razlike u njezinom definiranju, pa tako prema Shklar, (2009) web aplikacija je klijent poslužitelj veza koja koristi web preglednik kao svoj klijentski program. Pruža interaktivnu uslugu putem web poslužitelja preko interneta. Također jednu od prvih web aplikacija izradio je Tim Barners-Lee u CERN-u za traženje brojeva u internetskom (eng*. On-line*) imeniku.

Prema mojim saznanjima, web aplikacija je skup stranica koja treba biti interaktivna, odnosno korisnik vrši interakciju zahtjev-odgovor preko preglednika i pristupa do nje preko interneta. Stoga nemamo sadržaj koji možemo samo čitati, nego možemo i utjecati na njega. Također web aplikacija bi trebala imati komunikaciju s više korisnika preko mreže.

Web aplikacije mogu biti jednostavne ili kompleksne, od jedne stranice pa do velikih aplikacija s više stranica međusobno povezanih. Neke od web aplikacija su: Web mail, online kalkulator, Google Docs.

2.2 Kako web aplikacije rade?

Rad web aplikacije se može objasniti u sljedećih nekoliko koraka. Prema Filip Media d.o.o, (2020) prvo korisnik šalje zahtjev preko preglednika prema poslužitelju, nakon toga web poslužitelj šalje zahtjev prema aplikaciji na poslužitelju koja obradi zahtjev i generira rezultat

te ga nakon toga šalje nazad na poslužitelj koji isti taj rezultat prosljeđuje prema klijentu u web preglednik.

Web aplikacije su većinom napravite s web tehnologijama koje podržavaju preglednici a to su većinom HTML, CSS i JavaScript, koje naš preglednik može čitati i izvesti odgovarajuće operacije. HTML nam služi za izgradnju stranice te pruža osnovnu strukturu stranici koju potom pomoću CSS i JavaScript možemo poboljšavati. CSS se koristi za oblikovanje i izgled same stranice, dok JavaScript kontrolira različite elemente. Više o ovome možete saznati na sljedećoj blogu[1], Kolowich Cox, (2020).

Web aplikacije su napravljene tako da nema potrebe za njihovim skidanjem te im se može pristupiti putem mreže. Korisnici mogu pristupiti aplikaciji preko različitih preglednika. Web aplikacije mogu ali i ne moraju imati bazu. Također one su skupina statičkih i dinamičkih web stranica, a i sama web aplikacija može biti statička ili dinamička a o tome više u nastavku.

2.3 Aplikacija ili Stranica

„Ključna razlika između web aplikacije i web stranice je u načinu na koji komuniciramo sa svakom od njih. Web aplikacije definiraju se njihovim unosom - mi stvaramo, čitamo, ažuriramo i brišemo podatke unutar web aplikacije. Web stranice definiraju se njihovim izlazom - na web stranicama čitamo vijesti, marketinške informacije i česta pitanja.“(Johnston, 2020)

Neki korisnici web aplikacija i web stranica će ih smatrati istima kada stignu na njih, a neki će i web aplikacije i web stranice nazivati web stranicama. Ali dolaskom na web aplikaciju ili web stranicu se može uočiti razlika, a jedna od glavnih razlika je što na stranicama možemo samo čitati vijesti i informacije dok na aplikacijama imamo interakciju sa stranicom te možemo dodavati stavke ili brisati te ih uređivati. Kao još jedna od razlika između web aplikacija i stranica je i ona koja se odnosi na sigurnost. Web stranice su sigurnije pošto nema interakcije već samo čitamo informacije koje su nam prikazane dok kod web aplikacija postoje dinamički dijelovi te interakcija kao unos podataka na samu stranicu pa postoji i veća mogućnost da se desi napad na samu web aplikaciju.

Prema Fowler & Stanwick, (2004) web aplikacija je web stranica koja ima mogućnost interakcije i funkcionalne elemente. Također još navodi da aplikacije mogu izvoditi širok raspon funkcionalnosti te da ih je teže izraditi za razliku od web stranica. Postoje statičke i dinamičke

[1] *https://blog.hubspot.com/marketing/web-design-html-css-javascript*

web aplikacije kao i statičke i dinamičke web stranice tako da nam razlika između web stranice i web aplikacije postaje sve tanja i teže je razlikovati.

Imamo i neke osnovne razlike između web aplikacije i stranice koje su prikazane u sljedećoj tablici.

Tablica 1. Razlika između web aplikacija i web stranica, Izvor: (Sugandha, 2020)

Web aplikacija	**Web stranica**
Web aplikacija namijenjena je interakciji s krajnjim korisnicima.	Web stranica u osnovi sadrži statički sadržaj
Korisnik web aplikacije može čitati sadržaj web aplikacije i također manipulirati podacima	Korisnik web stranice može samo čitati sadržaj web stranice ali ne i manipulirati
Stranicu web aplikacije trebalo bi prethodno sastaviti prije implementacije	Web stranicu nije potrebno unaprijed sastaviti
Funkcija web aplikacije prilično je složena	Funkcija web stranice je jednostavna
Web aplikacija interaktivna je za korisnika	Web stranica nije interaktivna za korisnika
Mogućnosti preglednika uključenih web aplikaciju su visoke	Mogućnosti preglednika uključenih u web stranicu su visoke
Integracija je složena za web aplikaciju zbog svoje složenosti	Integracija je jednostavna za web stranicu
Web aplikacija uglavnom zahtijeva provjeru autentičnosti	Provjera autentičnosti na web stranici nije potrebna
Primjer:Amazon,Facebook	Primjer:Vijesti, Aktu stranica

2.4 Prednosti i nedostaci web aplikacija

Prema Horizont, (n.d.) kao prednosti web aplikacija možemo navesti: aplikacijama pristupamo samo preko preglednika, mogućnost pristupa bez obzira na uređaj i sustav koji koristimo, ne zauzima lokalnu memoriju računala, bez obzira na broj korisnika imamo dobre performanse, nadogradnja se radi na poslužitelju a ne na lokalnim računalima i svi koriste istu verziju aplikacije. Neki od nedostataka su: za rad potreban internet, brzina ovisi o mreži i poslužitelju, problem sigurnosti zbog mogućeg napada na poslužitelj, vlasnik aplikacije ima kontrolu nad aplikacijom kao i podacima, nedovoljna usuglašenost web standarda, potrebno znanje i poznavanje tehnologija za kreiranje web aplikacija.

3. Vrste Web aplikacija

Razlikuje se veći broj web aplikacija i web stranica, čije će se vrste u nastavku nabrojati te opisati o svakoj po nešto. Web stranice mogu biti statičke i dinamičke.

- **Statičke web stranice** – su stranice koje su napisane u HTML jeziku te na zahtjev za web stranicom vratit će odgovor u obliku stranice bez dodatne obrade i neće nam ponuditi mogućnost interakcije sa stranicom. Sadržaj na ovakvim stranicama se neće promijeniti osim ako se ručno ne promijeni sami kod stranice.
- **Dinamičke web stranice** – bazirana na HTML i CSS kao i statičke ali da bi dinamička stranica bila funkcionalna potrebni su nam jezici kao što su PHP (eng*. Hypertext Preprocessor*), JavaScript, ASP (eng*. Active Server Pages*), CGI (eng*. Common Gateway Interface*) koji se nalaze na stranci poslužitelja.

Vrste web aplikacija prema Svitla, (2019) su:

- **Progresivne web aplikacije** (eng*. Progressive Web App-PWA*) – su dosta slične nativnim aplikacijama nastale spajanjem ideje web stranice i mobilne aplikacije, te imaju neke njihove funkcije kao što su izvanmrežni rad, instalacija na uređaj.
- **Aplikacije na jednoj stranici** (eng*. Single-page applications - SPA*) – su aplikacije s jednom stranicom koje se dinamički ažuriraju dok korisnik vrši promjenu prepisivanjem trenutne stranice, ne učitava cijelu stranicu samo promjenu koju je korisnik napravio.
- **Statična web aplikacija** – je bilo koja web aplikacija koja može biti isporučena izravno do preglednika krajnjeg korisnika bez izmjena na strani poslužitelja HTML, CSS ili JavaScript sadržaja. Iako ovo može obuhvaćati vrlo ravne, nepromjenjive web stranice poput korporativne web stranice, statične web aplikacije obično se odnose na bogate web stranice koje koriste tehnologije u pregledniku umjesto na poslužitelju za isporuku dinamičnog sadržaja.“(StaticApps, 2021).

Prema StaticApps, (2021) glavna razlika koju možemo uočiti između tradicionalnih i statičkih web aplikacija su da same web aplikacije mogu koristiti veliki broj tehnologija, okvira i programskih jezika te da se HTML dokument kreira na poslužitelju i pošalje klijentu kao gotov dokument, dok se kod tradicionalnih za sve to pobrine preglednik.

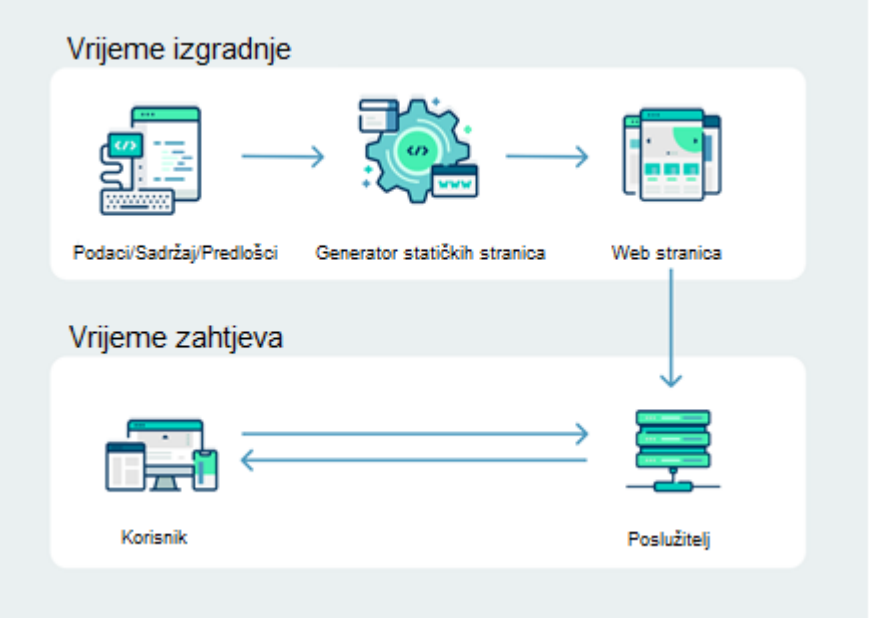
4. Generatori statičkih web aplikacija

„Generator statične web stranice (eng. *Static Site Generator*) je alat koji generira cjelovitu statičku HTML web stranicu na temelju sirovih podataka i skupa predložaka. U osnovi, statički generator web mjesta automatizira zadatak kodiranja pojedinih HTML stranica i priprema te stranice da ih prije vremena posluže korisnicima. Budući da su ove HTML stranice unaprijed izrađene, mogu se vrlo brzo učitati u korisnikovim preglednicima.“ (Cloudeflare, 2021).

Generator prevodi kod oznaka i stil (eng.*style*) jezika koji smo pisali u HTML i CSS. Jedna od razlika između skupa web aplikacija i generatora statičkih stranica je u tome da statički generator ne čeka dobivanje zahtjeva za stranicom kako bi nakon toga generirao stranicu i poslao korisniku nego unaprijed generira sve mogućnosti te kada stigne zahtjev za stranicom odmah je šalje korisniku, čime se skraćuje vrijeme obrade.

Neki od statičkih generatora su: Jekyll, Hugo, Gatsby, Harp, Next.js, DocPad, Octopress, Hexo a pojedini će se u okviru rada i detaljnije opisati.

Sljedeća slika prikazuje proces od vremena izgradnje do trenutka kada korisnici pošalju zahtjev za traženom stranicom.



Slika 1. Proces izrade stranice, Izvor: (Hawksworth, 2020)

Prema Buckler, (2016); Rinaldi, (2021) neki od razloga zašto bi koristili generatore statičkih stranica su:

- **Brzina** – zbog toga što u generiranje stranica nije uključen poslužitelj, niti se poziva baza podatka. Ne generira stranice na zahtjev, nego se unaprijed uslužuje generirane stranice što ga čini bržim od dosadašnjih rješenja.
- **Sigurnost** – niti jedna stranica nije 100% sigurna, ali neka od rješenja nude veću sigurnost od drugih. Obzirom da kod statičkih stranica nemamo bazu podataka te malo, ili ništa funkcija na poslužitelju smanjuje se mogućnost napada.
- **Bolje performanse** – statička stranica je unaprijed predmemorirana stranica koja ne ističe (eng*. expire*) za razliku od većine sustava za upravljanjem sadržaja (eng. Content Management System - CMS).
- **Jednostavnost** – statičku stranicu možemo smjestiti (eng*. hosting*) na bilo koji poslužitelj koji može vratiti HTML dokument i nema potrebe za dodatnim instalacijama dodataka ili aplikacija što je čini dosta jednostavnom za korištenje.
- **Kontrola i testiranje** – možemo ažurirati stranicu dok je dostupna (eng*. online*), odnosno ona se ažurira na novu verziju. čim nova verzija stranice dobije odobrenje. Ako koristimo GitHub i pisali smo kod u Markdown-u te ako se pojavi greška, odnosno neće se moći ažurirati stranica pa se stara verzija vrati nazad, a stranica ponovno postane dostupna.
- **Nije posve statička** – statički generator stranica može stvoriti dinamičku aplikaciju kao i bilo koju drugu aplikaciju. JAMstack (JavaScript, APIs, Markup - JAM) stranice mogu biti dinamičke korištenjem API-a (eng*. Application Programming Interface*) i servisa u vrijeme izgradnje stranice na strani klijenta, a ako ne riješimo neke dinamičke dijelove u vrijeme izrade imamo JavaScript na strani klijenta.

4.1 Razlika između statičkih generatora i sustava za upravljanjem sadržaja

CMS je softver koji pomaže kod same izrade web stranice te kako bi njezin sadržaj kreirali, izmijenili i upravljali s njim. Pomaže da se izgradi stranica, a da se pritom ne mora pisati kod od početka ili uopće imati znanje kako pisati kod. Jedan od poznatijih i najviše korištenih CMS-ova je WordPress.

Na samim počecima kodiranja stranica sve se radilo ručno i HTML se nije mijenjao nakon pisanja, a generirao se unaprijed. Pojavom CMS-a programerima je olakšano što neke stvari više nisu morali ručno iznova pisati već su se pojavili predlošci i automatska generiranja

stranica. CMS radi na principu da se podaci spremaju u bazu podataka. Postupak rada CMS-a se odvija tako da od baze zatraži sadržaj koji će se pojaviti na stranici, identificira predložak u koji ide sadržaj koji se zatraži, generira se stranica s tim sadržajem i predloškom te da se takva stranica posluži korisniku.

Statički generatori stranica kombiniraju ova dva pristupa. Korisniku se pošalje gotova stranica koja se ne generira na zahtjev korisnika već je sve izgenerirano unaprijed i posluži se bez uplitanja baze, a također omogućava i upotrebu predložaka kao i mogućnost automatskog generiranja stranica.

4.2 Jekyll

Prema Maroli et al., (2021) Jekyll je jedan od prvih statičkih generatora stranica kojeg je kreirao suosnivač GitHub-a Tom Preston-Werner 2008. godine. Kreiran je pomoću Ruby programskog jezika. Jekyll je pokretač (eng. *Engine*) kojeg koristi GitHub a koji se može iskoristiti za smještaj stranice u njegovo skladište (eng. *repository*) i to besplatno. Sadržaj se piše u Markdown-u, a za predložak (eng. *templating*) se koristi Liquid engine. Jekyll nam omogućava da napravimo bilo koju vrstu web stranice, ali najviše se koristi i namijenjen je za izradu blogova. Jedna od značajki kao i kod ostalih statičkih generatora je i ta da nije potreban PHP ili baza podataka. Ako posjedujete blog u WordPress, Blogger, Drupal, Joomla, imate mogućnost da jednostavno migrirate u Jekyll. Može djelovati i kao lokalni poslužitelj koji generira HTML, CSS i JavaScript datoteke iz predloška Markdown, Sass ili CoffeeScript. Na njihovoj službenoj stranici[2] možete više saznati o Jekyll-u.

4.3 Gatsby

Prema Gatsby, (2021) Gatsby je mlađi statički generator od Jekyll-a, ali ništa manje loš. Prema službenoj stranici kreiran je 2015. godine od strane Kyle Mathewsa. Zasnovan je na bazi React-a i pokreće ga GraphQL. Pitanje koje se može postaviti, Treba li za rad s Gastby poznavati React i GraphQL. Poznavanje React-a i GraphQL-a je poželjno, ali nije presudno za sam rad s njim. Ako želite pristupiti podacima na jednostavan način Gatsby je odličan izbor. Ima mogućnost prikupljanja podataka iz Markdown-a, JSON (eng. *JavaScript Object Notation*), CMS, API-a. Gatsby nalikuje više na moderne na korisničkoj strani (eng. *front-end*) okvire nego na stare statične generatore stranica. Za Gatsby se kaže da je popularizirao statičke generatore temeljene na JavaScript-u kao i upotrebu GraphQL u JAMstack aplikacijama.

[2] https://jekyllrb.com/

Dolazi i s dodatcima (eng. *plugin*) koji mu omogućavaju dodatne funkcionalnosti. Također ako se želi prikupiti podatke iz WordPress-a potrebno je samo pronaći pravi dodatak. Gatsby, kao i ostale statičke generatore stranica krasi ga brzina kreiranja statičkih web stranica.

4.4 Hugo

Prema Hugo, (2021) je kreirao Steve Francia 2013. godine i napisan je u GO (Golang) programskom jeziku. Reklamira se kao najbrži i najpopularniji okvir za izgradnju web stranica. Na GitHub stranici ima preko 50 tisuća zvjezdica, što ga u ovom trenutku čini jednim od najpopularnijih. Naknadno u radu će se prikazati usporedba po brzini generatora kako bi se vidjelo je li on zaista najbrži, kao što se reklamira. Hugo ima dobru Markdown podršku za kratke kodove, a sadržaj se također može pisati u Markdown-u. Radi na principu da uzme direktorij sa sadržajem i predlošcima te od njih napravi cjelovitu HTML stranicu. U Hugo okviru se pomoću Markdown-a može deklarirati sadržaj, kao i meta podatke i to u TOML (eng. *Tom's Obvious, Minimal Language*), JSON, YAML (eng. *YAML Ain't Markup Language*). Web stranice koje su napravljene u Hugo okviru mogu se koristiti tj. biti smještene na GitHub, Netlify, Dropbox, S3 kao i drugim servisima za smještanje. Hugo je napravljen za kreiranje blogova, kao i Jekyll, ali također može praviti i druge stranice. Na službenoj stranici[3] Hugo okvira možete više saznati o njemu, a u nastavku rada će se detaljnije opisati sam okvir za kreiranje statičkih stranica, njegove mogućnosti i sam princip rada.

4.5 Usporedba Generatora

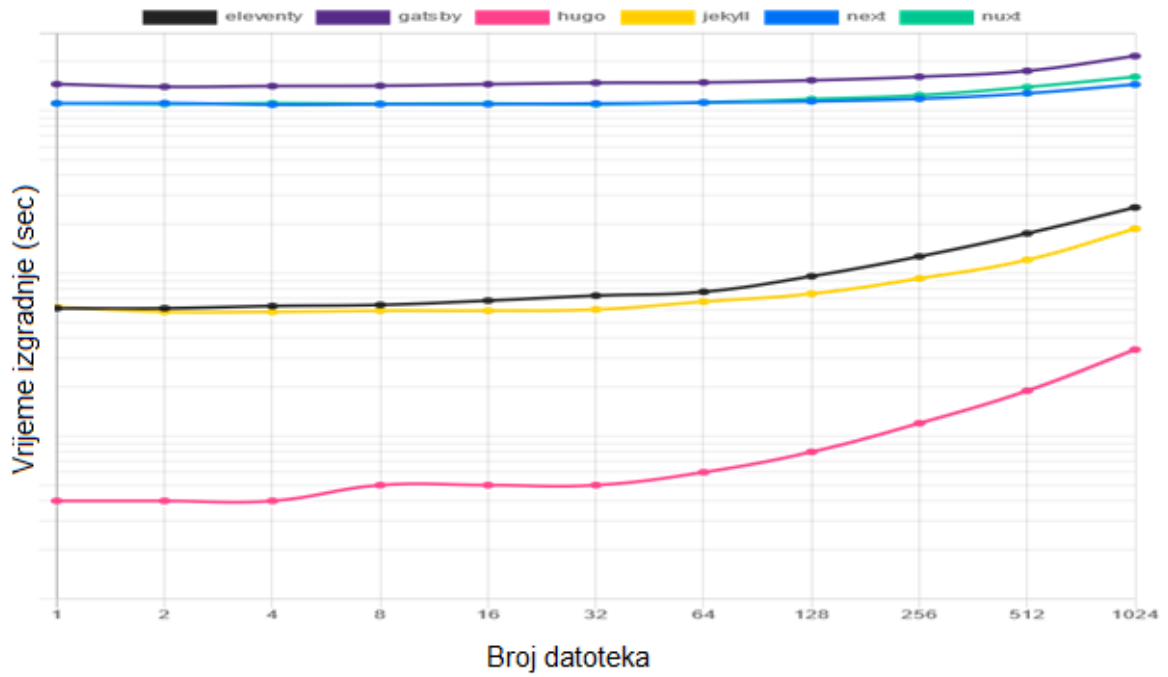
U sljedećoj tablici prikazat će se neke od usporedbi između ovih generatora.

Tablica 2. Usporedba Generatora, Izvor: (Shaleynikov, 2020)

	Hugo	**Jekyll**	**Gatsby**
Jezik	Go	Ruby	JavaScript
Predlošci	Go	Liquid	React
Integracija	Markdown,GitLab Pages	Algoria,GitLab Pages	React,GraphQL,Webpack
Licenca	Apache 2.0	MIT	MIT
GitHub zvijezde	~50988	~42467	~49642

[3] https://gohugo.io/

Sljedeća slika prikazuje brzinu generatora u obradi datoteka koje su smještene na Github-u. Test je rađen na maloj stranici koja sadrži 1024 dokumenta. Osim već opisana tri generatora, u testu se nalaze i drugi te se može uvidjeti usporedba između njih.



Slika 2. Usporedba generatora po brzini, Izvor: (C Davis, 2020)

Kao što je već rečeno, Hugo se reklamira kao najbrži generator. No tu tvrdnju se može i treba provjeriti. Stoga prema prethodnoj slici može se utvrditi kako je sama tvrdnja točna, odnosno Hugo jest najbrži generator od šest generatora koji su prikazani. Za rezultate test nad velikim brojem datoteka kao i jednoj datoteci i možete posjetiti stranicu[4] netlify, (2021).

U praktičnom dijelu rada napravit ćemo također usporedbu po brzini te ćemo prikazati rezultate koje smo dobili tijekom izrade projekta.

[4] https://ssg-build-performance-tests.netlify.app/

5. Markdown i JAMstack

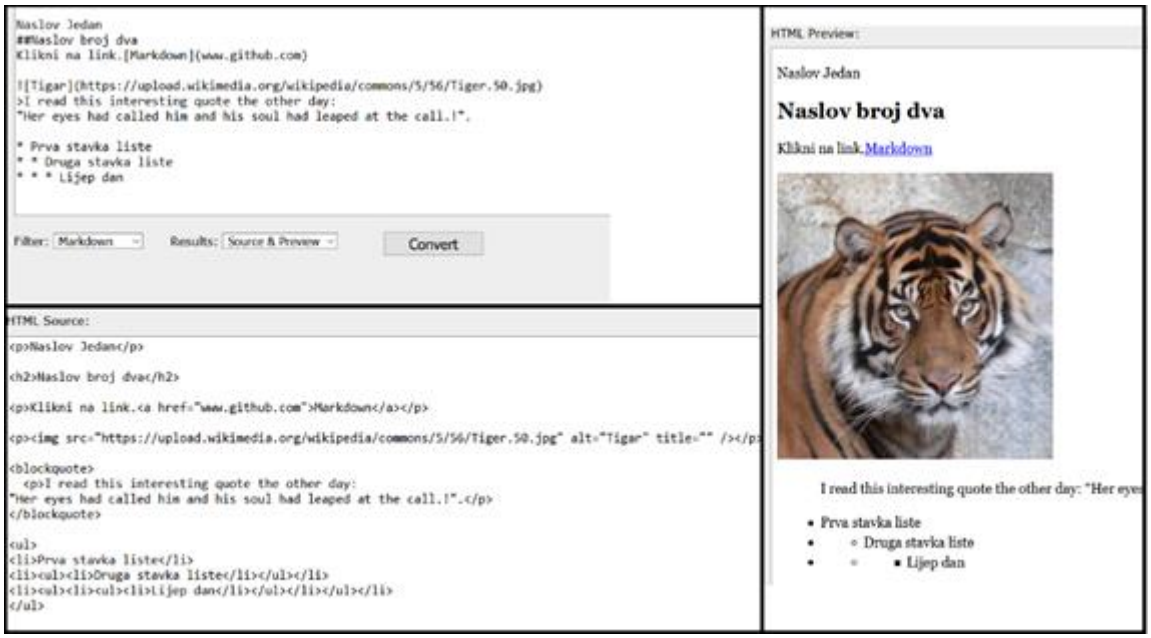
5.1 Markdown

Markdown je LML (eng. *Lightweight markup Language*) koji je nastao 2004. godine, a stvorili su ga John Gruber i Aaron Swartz. Nakon što su napravili Markdown, John je rekao: „Markdown je alat za pretvaranje teksta u HTML za web pisce. Markdown vam omogućuje da pišete koristeći jednostavan-za-čitanje, jednostavan-za-pisanje format običnog teksta, a zatim pretvoriti u strukturno valjan XHTML (ili HTML).“(Gruber, 2021)

Markdown omogućava pisanje teksta u običnom editoru, kao što je Notepad koji se potom na lagan način pretvori u HTML. Kada se piše u njemu datoteka se mora spremiti pod ekstenzijom „.md“ da bi dokument bio spremljen kao Markdown. Dok pišemo u Notepad-u promijena nije odmah vidljiva, treba pogledati kao PDF ili u modu pregleda (eng. *preview*) da bi vidjeli što je napisano. Razlog tomu je što kada se piše dokument u Markdown sintaksi na tekst se dodaju samo atributi koje treba pretvoriti da bi se vidjelo kako tekst izgleda s atributom koji je stavljen. Kako bi se navedeno izbjeglo, mogu se koristiti editori koji podržavaju Markdown sintaksu i prikazuju odmah formatirani tekst. Ono što Markdown omogućava je pisanje dokumenta bez tagova te se cijeli tekst koji pišemo nalazi se između „<body> </body>“ tagova HTML-a. Velika većina statičkih generatora podržava Markdown te se koristi zbog svoje jednostavnosti i mogućnosti brzog pisanja.

Sljedeća slika prikazuje kako izlgeda tekst pisan u Markdownu, pretvoren u HTML te kako bi izgledao na stranici. U primjeru se mogu vidjeti neki od osnovnih atributa uz tekst i kako bi taj tekst izgleda s njima. Ovaj primjer je napravljen u online editoru kojeg se može naći na Daring fireball stranici[5].

[5] https://daringfireball.net/projects/markdown/dingus



Slika 3. Primjer Markdown-a, Izvor: (Gruber, 2021)

5.2 JAMstack

Prema Netlify, (2021) JAMstack je arhitektura koja se gradi na mnogim alatima te omogućava programerima veliku produktivnost. Naziv JAMstack je došao od Matt Biilmann-a i Vhris Bach-a kada su radili na mogućnostima razvoja na Netlify-u i shvatili da nema jednostavnog načina na koji bi mogli opisati arhitektonski naziv.

Prema cloudflare, (2021) JAMstack je pristup izgradnji s prednje strane (eng*. Frontend development*) sučelja i sadržaja s kojim se korisnik komunicira te omogućava programerima brzo postavljanje statičkih web stranica za korisnike.

JAMstack izraz prestavlja JavaScript, APIs, Markup. JavaScript je programski jezik kojeg koriste web aplikacije, API predstavlja način na koji se traži podatke od treće strane za određenu web aplikaciju, a jezik oznaka pruža upute pregledniku za formatiranje. JAMstack stranica koja se poslužuje izravno s CDN-a (eng. *content delivery network*) može se isporučiti vrlo brzo i sigurno. Ono što omogućava brzinu i sigurnost kao što je ranije u radu navedeno, jest činjenica da je web stranica statička te dohvaća čisti HTML koji se unaprijed generiran, a sadržaj preko API-a popunjava s treće strane. Umjesto da se na stranci poslužitelja pokreću aplikacije za generiranje dinamičkog sadržaja, dinamičke komponente aplikacije se temelje na API i JavaScript koje komuniciraju s pozadinskim uslugama.

Prema Netlify, (2021) razlozi i prednosti korištenja JAMstack-a te najbolja praksa izgradnje aplikacije se može vidjeti u nastavku. Odnosno zašto JAMstack govori sljedećih šest značajki:

- **Sigurnost** – Nema poslužitelja i baze podataka
- **Skalabilnost** – Kompenzacija veliko broja korisnika preko CDN-a
- **Izvođenje** – Veća brzina izrade i usluživanja
- **Cijena** – Troškovi posluživanja su mali ili ih nema
- **Održavanje** – Jednom kada se izradi i posluži veći dio posla je gotov
- **Iskustvo programera** – Brži i usmjereniji razvoj

Tijek izgradnje i najbolja praksa su: razvoj, kontrola verzije, automatska izgradnja, poslužiti stranicu.

6. Hugo Okvir

Hugo je statički generator stranica, brz i moderan, napisan je u programskom jeziku GO te je dizajniran da bi kreiranje stranica ponovno bilo zabavno. Statički generatori su korisni kada se prave manje zahtjevne stranice, stoga je i Hugo prilagođen kreiranju jednostavnijih web stranica i aplikacija. Kod kreiranja stranice potreban je predložak, te sadržaj koji će popuniti taj predložak. Predložak predstavlja vizualni dio te zajedno s CSS i JS (JavaScript) tvori temu. Za popunjavanje predloška sa sadržajem koriste se datoteke u HTML ili Markdown formatu. Hugo je optimalan za korisnike zbog toga što se stranice više pregledavaju već uređuju i ažuriraju. Također, tu dolazi njegova brzina do izražaja, zbog toga što se stranica izradi tijekom kreiranja ili ažuriranja te se poslužuje korisnicima, za razliku od sustava koji dinamički grade stranicu sa svakim zahtjevom korisnika.

Kako bi se izgradi web stranica ili aplikacija u Hugo okviru potrebno je skinuti i instalirati Hugo na radnu jedinicu, kreirati novu stranicu, dodati temu, dodati sadržaj, pokrenuti server i testirati stranicu, urediti temu i sadržaj te gotovu stranicu objaviti.

6.1 Konfiguracija i struktura

Hugo se može skinuti s GitHub stranice u zip formatu te nakon toga instalirati. Instalacija se u Windows okruženju radi na sljedeći način. Potrebno je odrediti mjesto na kojem će se instalirati okvir Hugo te na tome mjestu kreirati mapa pod nazivom Hugo. Unutar mape se kreiraju dvije dodatne podmape *bin* i *sites*. Skinuta zip datoteka se raspakira u podmapu bin, te preko PowerShell dodaje izvršnu datoteku hugo.exe u PATH i to preko naredbe

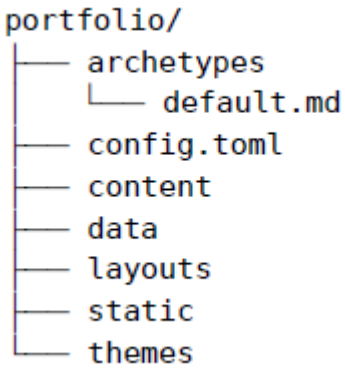
```
set PATH=%PATH%;C:\Hugo\bin.
```

Nakon što se instalira Hugo, može se krenuti s kreiranjem stanice. U naredbeni redak (eng*. Command Prompt - CMD*) se unese naredba `hugo help` kako bi se provjerilo je li je instalacija uspješno završena. Iz bin podmape se prebacuje u drugu podmapu *sites*, i u njoj se kreira nova stranica. Kreiranje nove stranice se izvodi putem ove naredbe:

```
C:\Hugo\Sites> hugo new site naziv stranice
```

Kada se kreira stranica može se krenuti na dodavanje teme i sadržaja.

Nakon kreiranja stranice u podmapi *sites* pojavljuju su se nove datoteke i mape koje predstavljaju strukturu stranice i one izgledaju ovako:



Slika 4. Struktura stranice, Izvor: (Hogan, 2020)

Prema Atishay, (2019):

- **Archetypes** - Sadržava predloške za stvaranje novog sadržaja. Moguće je napraviti predložak za sadržaj u Markdown, kojeg Hugo koristi da napravi kostur sadržaja. Zbog nepostojanja baze podataka, metapodaci se trebaju upisivati ručno, a kada bi postojao predložak, on bi to odradio, umjesto ručnog upisivanja.
- **Config.toml** – Predstavlja konfiguracijsku datoteku web stranica koja sadrži i metapodatke koji se primjenjuju na cijelu stranicu, kao naziv teme do svih ostalih parametara koje koristi Hugo kod kreiranja sadržaja.
- **Content** – Obuhvaća sav sadržaj sa stranice u Markdown ili HTML formatu. Sav sadržaj koji se sprema na bazu kod Hugo okvira se sprema u ovu mapu.
- **Data** – Ova mapa sadrži statičke i strukturne podatke za web stranicu u YOML, TOML ili JSON formatu, te mogu biti dostupni kao globalne varijable kroz web stranicu.
- **Layouts** – u ovom folderu se nalazi HTML datoteka za različite tipove stranica, i koristi se da bi se nadjačali dijelovi teme koje se žele promijeniti a da se ne mora promijeniti samu temu.
- **Static** – Ovdje se sprema statički sadržaj kao što je fonts, pdf, slike ili sadržaj koji se skida onakav kakav jest.
- **Themes** – U ovoj mapi se nalazi tema koja se koristi za izgled podataka kao i same stranice. Tema je napisana u Go jeziku i može biti nadjačana iz mape Layouts.
- **Assets** – Unutar ove mape se spremaju slike, javascript, i css datoteke koje su dostupne svim stranicama u aplikaciji. U starijim verzijama Hugo okvir je koristio *static* mapu za ove datoteke. Mapa dopušta da Hugo promijeni veličinu slike, smanji JavaScript i konvertira SCSS i CSS.

- **Resources** – Dok procesuira podatke, Hugo sprema rezultate većih operacija u ovu mapu. Sadrži različite dodatke kao skripte i stilove.
- **Public** – Predstavlja mapu koja je glavna izlazna mapa u Hugo okviru, ovdje se generira izlazni HTML koji se uslužuje.
- **Go.mode** i **go.sum** – Ovo je datoteka koja ima listu svih stvari o kojima projekt ovisi i koristi da bi upravljao s modulima.

Kao što se može vidjeti na listi se nalaze i dodatne mape koje nisu bile prikazane na slici koja predstavlja strukturu stranice. Sve navedene mape se mogu pojaviti kad se kreira neka stranica ili aplikacija. Slika predstavlja početnu strukturu projekta koja se pojavi kad se kreira prva stranica.

6.2 Teme i predlošci

Kod kreiranja stranica, blogova, aplikacija koriste se gotove teme koje pojednostavnjuju i ubrzavaju izradu samih. Koriste se gotove teme koje se mogu prilagoditi po korisnikovoj želji. Hugo na svojim službenim stranicama ima veliki izbor tema gdje svatko može naći nešto za svoj projekt koji izrađuje. Teme se vrlo lako primjenjuju na stanice, mogu se ručno skinuti sa stranice i dodati u mapu teme, a može i preko par komandi. Svaka tema uz sebe ima uputstva kako da se primjeni na stranicu. Jedan od primjera kako dodati temu se može vidjeti u nastavku. Prvo je potrebno klonirati temu direktno u direktorij.

```
$ git clone https://github.com/your-identity/hugo-theme-dimension.git
themes/dimension
```

Potom ide:

```
$ git submodule add https://github.com/your-identity/hugo-theme-
dimension.git themes/dimension
```

Zatim se otvori *config.toml* te promijeni *baseURL* i *title*. Moguće je pokrenuti stranicu s temom na lokalnom uređaju i provjeriti kako tema izgleda s jednostavnom naredbom:

```
$ hugo server -t dimension
```

I na sljedećem se linku[6] može pogledati.

Prema Hogan, (2020) temu je moguće i generirati na temelju stanice koja se napravi. Tema se može napraviti tako da u mapi *themes* kreiramo novi direktorij, ali Hugo ima mogućnost koja omogućava generiranje teme preko naredbe.

[6] http://localhost:1313

Kada se unese naredba: `$ hugo new theme generator`

Ova naredba kreira mapu *generator* unutar *themes* i pripadajuće podmape unutar mape *generator*. Kako se u novo kreiranoj podmapi *layouts* ne nalazi ništa, treba kopirati ili prebaciti datoteke iz mape u kojoj se datoteke nalaze. Kako se ne bi ručno kopiralo, postoji sljedeća naredba:

```
$ mv layouts/index.html themes/basic/layouts/index.html
$ mv layouts/_default/single.html
themes/basic/layouts/_default/single.html
```

Zatim je potrebno otvoriti *config.toml* i dodati na kraju naziv teme da bi tema bila gotova.

Nakon opisivanja tema, važno je istaknuti ponešto o predlošcima. Prema Hugo, (2021) Hugo za izradu predložaka koristi Go *html/templates* i *text/templates* biblioteku kao osnovu. Go predlošci su HTML datoteke s dodatkom varijabli i funkcija koje su dostupne unutar {{}} zagrada. Postoje pred definirane varijable koje se označavaju s *."Naslov" (točka Naslov)* i prilagođene varijable koje se označavaju s *$adresa.* Parametri funkcija se razdvajaju s razmakom, a zagrade se mogu koristiti i za grupiranje. Hugo dolazi sa samo nekoliko osnovnih funkcija, ali pruža mogućnost za proširenje. Funkcije se pozivaju imenom nakon čega slijede parametri odvojeni razmakom. Kod dodavanja funkcije potrebno je ponovo generirati stranicu. Mjesto predložaka uvijek kreće s *layouts/* .Imamo mogućnost korištenja samo djelomičnog (eng. *partial*) predloška.

6.3 Hugo moduli

„Moduli su zbirka povezanih Go paketa koji su povezani kao jedna cjelina unutar neke verzije“ (Dimoski, 2021).

Prema Gandhi, (2020) Hugo je u verziji 0.56 predstavio sustav modula. Moduli su blokovi u Hugo okviru i mogu biti cijeli projekt kao modul, ili samo manji dio, koji pruža jedan, ili više dijelova komponenti, ili tema. Predstavljaju direktorije koji imaju poddirektorije kao što su *static*, *data*, *layouts*. Ako se u projektu koji se radi nalaze moduli kao cjeloviti projekt, ili, samo neki dijelovi, dopušteno im je da koriste datoteke pohranjene u udaljenim spremištima i iskoristi kao svoje. Postoji mogućnost i da se iskoriste moduli koji nisu iz Hugo okvira. Kao što se drugi moduli mogu iskoristiti u projektu tako postoji mogućnost da projekt postane modul te se iskoristi u nekim sljedećim projektima. Hugo module pokreću Go moduli. Moduli moraju biti označeni i poredani po verzijama, tako oznaka za module izgleda kao v0.1.0 što predstavlja v(glavni).(manji).(zakrpa). Također, mogu se iskoristiti za dodavanje teme, kada se doda tema kao modul nije potrebno kopirati temu i stavljati datoteke u svoje mape, Hugo okvir se brine za

njihovo preuzimanje i korištenje. Također se koriste i za kratke kodove (eng. *shortcodes*), statičke datoteke i td..

6.4 Smještaj i implementacija

Generator statičkih stranica kao što je Hugo kreira statičke aplikacije i stranice koje se mogu smjestiti i objaviti na dosta mjesta, tako će se nabrojati par mjesta a neka od njih i dodatno opisati i objasniti postupak smještanja. Neka od najpopularnijih mjesta su: Netlify, GitHub, GitLab, Render, Bitbucket kao i mnogi drugi. U nastavku će se prikazati rješenje za GitHub i Netlify.

6.4.1 GitHub

Prema službenoj stranici GitHuba-a, on je platforma za smještaj koda, kontrolu verzije kao i suradnju više osoba na istom projektu.

Kako bi objavili stranicu na GitHub-u i da bi ona postala javno dostupna potrebno je napraviti sljedeće korake. Prvo je potrebno instalirati Git ako se već nije instaliran na kompjuteru jednostavnom naredbom: `git –version`.

Nakon toga potrebno je kreirati repozitorij, tako da se posjeti GitHub stranica, ode na stvaranje novog repozitorija, otvori se forma u koju se piše ime repozitorija u formatu: *<username>.github.io,* označi se javni, ili privatni repozitorij i ide na kreiranje te se on i kreira.

Poslje toga potrebno je kreirati na kompjuteru stranicu, blog, ili aplikaciju koju želimo, skinuti temu za željenu stranicu, otvoriti *config.toml te* treba promijeniti *baseurl* u ovakav format:

```
baseurl = https://[your github username].github.io/
```

Prije nego što se objavi stranica, potrebno je pogledati na lokalnom kompjuteru kako bi izgledala tako da se pokrenu sljedeće komande:

```
cd ime_stranice
hugo server
```

te nakon toga se posjeti *localhost:1313* i pogleda kako stranica izgleda.

Zatim se ode na *public* mapu te izvrši sljedeći kod koji će poslati datoteke na GitHub i stranica postaje dostupna:

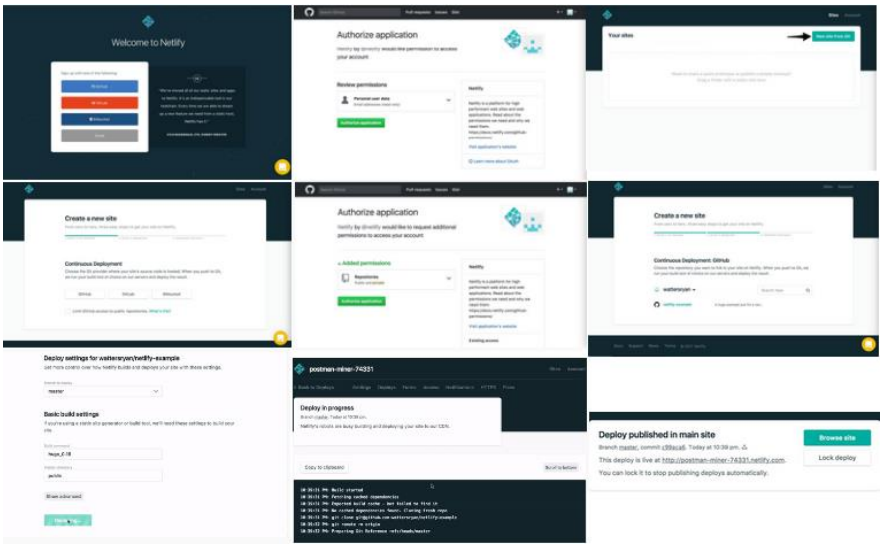
```
Cd public
git init
git add .
git remote add origin https://github.com/username/username.github.io.git
git commit -m "first commit"
git push origin master
```

Nakon što se napravi sve što je prethodno opisano, ode se na GitHub, na postavke i pri dnu se pojavi link u obliku [*https://[your github username].github.io*] gdje piše da je stranica objavljena i da se može pristupiti s toga linka.

6.4.2 Netlify

Prema Hugo, (2019) i službenoj stranici, Netlify je tvrtka koja nudi mogućnost, implementacije, suradnje i izgradnje web aplikacija u oblaku (eng. *cloud*). Kako bi smjestili svoju stranicu, blog, ili aplikaciju na Netlify i da postane javno dostupna potrebno je napraviti sljedeće.

Prvo se mora kreirati račun na njihovoj stranici. Račun se brzo i jednostavno kreira jednostavnim odabirom kreiraj preko GitHub-a, ako se ima GitHub račun, ako ne posjedujete HitHub račun može se prijaviti sa GitLab računa ili BitBucket. Ne posjedujete niti jedan od navedenih računa nije problem može se prijaviti i preko Email adrese ili napraviti neki od navedenih računa. Nakon kreiranja i prijave ide se na kreiraj novu stranicu s GitHub-a. Netlify će zatražit će dopuštenje da pristupi Github repozitoriju da može povući datoteke s njega, što se treba odobriti. Ako postoji više repozitorija odabere se samo jedan koji će se koristiti u izgradnji stranice. Poslije odabira otvara se prozor u kojem se odabire granu (eng. branch) - *master*, naredba za izgradnju (eng. *Build command*) – *Hugo,* te direktorij u kojem se kreira - *public*. Sljedeće dugme je izgradnja. Nakon što se izvrši naredba izgradi, klikom na dugme pojavljuje se prozor koji obavještava da je stranica gotova i da se može posjetiti na sljedećem linku. Odabirom linka pokreće se stranica i javno je dostupna svima. U sljedećih par slika vidi se postupak kreiranja:



Slika 5. Kreiranje stranice u Netlify, Izvor: (Hugo, 2019)

7. Čitanje podataka u okviru Hugo

Prethodno su navedene mape u strukturi stranice te čemu pojedina mapa služi. Tako za sadržaj Hugo čita markdown datoteku koja se nalazi u *content* mapi. Slike i ostala imovina (eng. *Assets*) se čita iz mape *static,* a konfiguracija se čita iz korijenske mape spremljena u TOML ili YAML datoteci.

TOML i YAML su minimalni formati za konfiguracijske datoteke, čitljivi su i jednostavni te koriste sveprisutne tipove podataka. U TOML i YAML datoteku se mogu zapisivati i čitati podaci koji nisu konfiguracijski. Podatkovne datoteke se pohranjuju u podatkovnu mapu, te ta mapa može biti kao mini baza podataka u koju se zapisuju podaci s naše stranice ili aplikacije. Kao što je već navedeno Hugo prihvaća TOML, YAML i JSON datoteke. U sljedećem primjeru može se vidjeti kako to izgleda.

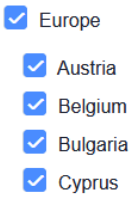
Prema (Cresswel, 2018) primjer zapisa i čitanja svih država u Europi izgleda ovako: U *data* mapi napravi se još jedna podmapa, koja se nazove države a unutar te mape se nalazi još jedna naziva kontinenti. Unutar mape kontinenti se napravi datoteka europa.yaml u koju će se zapisati sve zemlje u Europi. Zapis zemalja bi izgledao ovako:

```
zemlja:
  - name: Austria
  - name: Belgija
  - name: Bugarska
  - name: Cipar
```

Kako bi iskoristili te podatke koristi se *range* funkcija koja prolazi kroz datoteku sa *.Site.Data.* Ovako bi izgledao kod za ispis zemalja iz Europe:

```
{{ range .Site.Data.države.kontinenti.europa.zemlja }}
  <li>
    <input class="material-icons" type="checkbox" />
    <label>{{ .name }}</label>
    </input>
  </li>
{{ end }}
```

{{ .name }} ispisuje popis zemalja. To bi izgledalo ovako:



Slika 6. Ispis zemalja, Izvor (Cresswel, 2018)

8. Kreiranje višejezičnih aplikacija u okviru Hugo

Okvir Hugo pruža i mogućnost pravljenja višejezične aplikacije, a ako aplikacija postane popularna te se želi prevesti na druge jezike, Hugo pruža i tu mogućnost.

Prema Bodrov-Krukowski (2019) treba se prevesti sve od samog sadržaja pa do stringova koji se nalaze unutar HTML datoteke kao što su meni, naslovi. . . . Također, treba promijeniti i URL te će se unutar njega nalaziti i LANG_CODE (eng*. Language Code*), što će označavati jezik koji se trenutno prikazuje na našoj stranici. U nastavku se može vidjeti kako bi to izgledalo: [http://localhost:1313/LANG_CODE/knjige/index]. U nastavku će se prikazati i sam kod aplikacije koja se izrađuje ili prevodi na dva jezika.

Na početku prvo što se treba napraviti je odrediti koji jezici se planiraju koristiti u web aplikaciji. U opisanom slučaju određen je engleski i hrvatski jezik. Sljedeće se ažurira *config.toml* datoteka, tako da se unesu jezici koji su se odabrali i stavi zadani jezik, na taj se način specificira jezik i spremi se.

```
#config.toml
defaultContentLanguage = "en"
[languages]
  [languages.en]
    title = „Site name“
    LanguageName= „English“
    [languages.en.menu]
        [[languages.en.menu.main]]
            name = "Home"
            url = "/index"
            weight = 0


  [languages.hr]
    title = „Naziv Stranice“
    LanguageName= „Hrvatski“
    [languages.hr.menu]
        [[languages.hr.menu.main]]
            name = "Početna"
            url = "/index"
            weight = 0
```

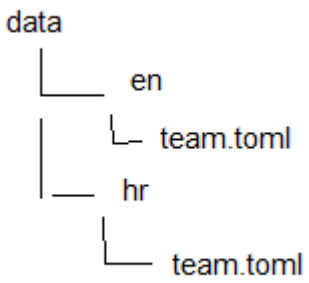
U kodu iznad se može vidjeti specificiranje za sadržaj kao i za izbornik. U nastavku će se prikazati kod za odabir jezika koji se želi koristiti, kao i na koji način se može prevesti sadržaj, stranica po stranica ili cijela mapa.

```
<ul>
{{ range $.Site.Home.AllTranslations }}
<li><a href="{{ .Permalink }}">{{ .Language.LanguageName }}</a></li>
{{ end }}
</ul>
```

Može se prevesti datoteka kao i cijela mapa. Kod prijevoda datoteke se u nazivu se stavlja i oznaka jezika, te bi naziv izgledao ovako: `/content/about.en.md`

Na onom jeziku na kojem je datoteka napisana taj se dodatak stavi u samo ime datoteke. Dok se kod prijevoda mape napravi mapa jezika, u koju se stavljaju datoteke u tom jeziku bez posebnog dodatka u nazivu za jezik u kojem je pisana. Naziv u tom slučaju bi izgledao ovako: `/content/english/about.md`

Za razliku od sadržaja i izbornika, datoteke podataka nisu svjesne jezika te za svaki jezik treba napraviti posebnu datoteku. Struktura datoteka u tom slučaju izgleda ovako:



Slika 7. Struktura datoteka podataka

Hugo podržava višejezičnost te ima mogućnost izrade stranica i aplikacija na više jezika kao i prijevod postojećih. Da bi izradili višejezičnu aplikaciju potrebno je definirati jezike koji se koriste, napraviti izbornik za jezike, prevesti izbornik, napraviti mapu u koju se stave sve datoteke u tom jeziku.

9. Izgradnja aplikacije u okviru Hugo

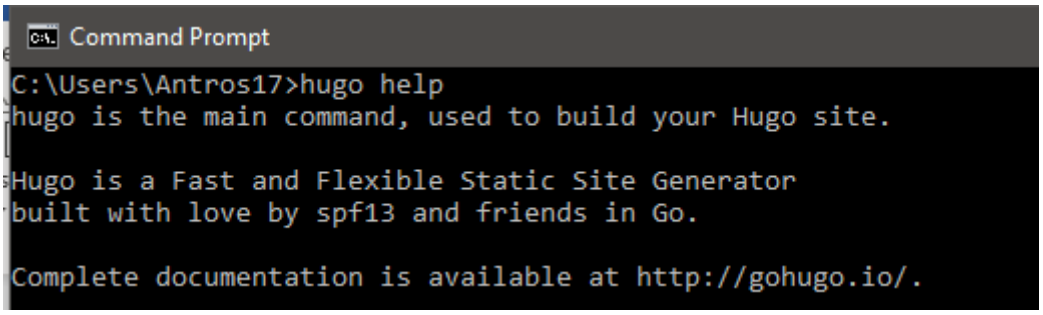
U praktičnom dijelu rada će se izraditi jednostavna aplikacija u Hugo okviru te nakon toga prebaciti tu istu aplikaciju u WordPress. Uradit će se testovi kako bi se usporedilo te dvije aplikacije koje se temelje na različitim rješenjima, a napravit će se i test na čistom kodu koji će se vrtjeti na python serveru. Na ovaj način će se usporediti jedna aplikacija koja sadrži isti kod, te će provesti isti test za svaku. Svi koraci koji se naprave u realizaciji projekta biti će prikazani u radu i dokumentirani.

9.1 Instalacija Hugo okvira i kreiranje stranice

Za praktični dio je odlučeno napraviti jednostavnu aplikaciju koja predstavlja policu knjiga. U aplikaciji će se nalaziti prva stranica s opisom aplikacije, druga stranica na koju se mogu unijeti knjige te treća stranica s knjigama koje preporučuje autor. Kako Hugo nema server i namijenjen je za izradu blogova, neće biti mogućnost zapisa na server ili spremanja podataka, već samo prikaz unesenih knjiga preko međunarodnog standardnog knjižnog broja (*International Standard Book Number* - ISBN). Aplikacija je zamišljena kao aplikacija za dijeljenje i prijedlog knjiga. Unosom knjiga imamo popis koji možemo poslati drugoj osobi kao spisak za čitanje. Može se napraviti i stalni spisak knjiga koji prikazuje knjige koje čitamo ili smo pročitali, ali se aplikacija mora pokrenuti lokalno i knjige unositi direktno u kod i to iz razloga koji je naveden ranije, jer ne postoji server na kojem se mogu spremiti podaci. Za dobavljanje knjiga koristi se ISBN broj koji preko JS dohvati knjigu te prikaže na stranici.

Prvi korak u realizaciji je instalacija Hugo okvira. Instalacija je preuzeta je sa GitHub stranice [7] na koju usmjerava službena stranica Hugo okvira. Instalacija se napravi kao što je u radu gore već opisano. Kreirana je mapa koja ima dvije podmape *bin* i *Sites*. U *bin* se raspakira zip u kojem se nalaze tri datoteke *hugo.exe*, *licence* i *readme.md*. Nakon toga u CMD se upisala naredba `hugo help` koja je izbacila poruku da je Hugo uspješno instaliran.

[7] https://github.com/gohugoio/hugo/releases

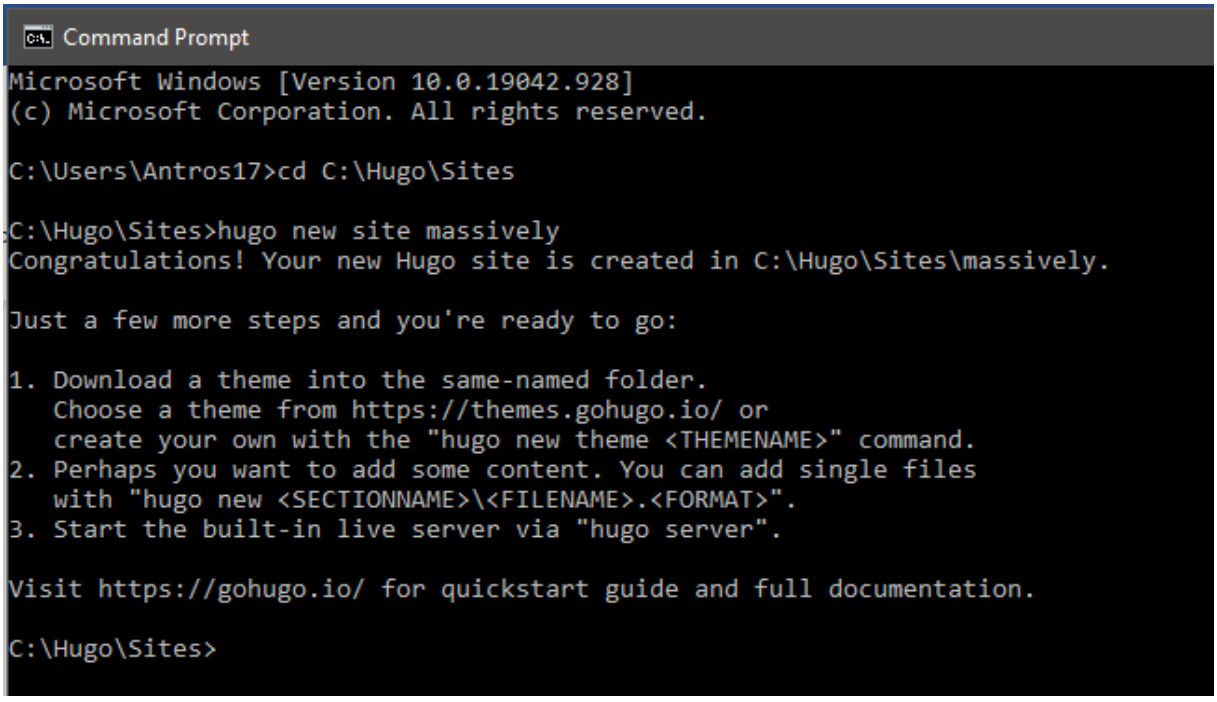


Slika 8. Terminal Hugo instal

Poslije instalacije kreirana je nova stranica u Hugo okviru te nakon toga odabrana tema za aplikaciju. Nova stranica se napravi s naredbom u CMD-u.

```
C:\Hugo\Sites> hugo new site massively.com.
```

Tema je pronađena na služenoj stranici[8]. Tema koja je odabrao je Massively, a na stranici postoje i uputstva kako primijeniti temu.



Slika 9. Kreiranje nove stranice

Nakon što je tema skinuta i stranica kreirana, tema se primjene na stranici. Tema se primjenjena na stranicu tako da se zip mapa koja je skinuta, raspakirana u mapu *themes* koja se nalazi u korijenskoj mapi same stranice. Kako bi pogledali kako stranica izgleda, pokrenemo Hugo server lokalno s naredbom `hugo server`. Kako bi se stranica pogledala, server se mora

[8] https://themes.gohugo.io/

pokrenuti iz mape gdje nam se nalazi aplikacija. Kad je server pokrenut može se pogledati stranica na [http://localhost:1313/] linku.

9.2 JavaScript dio

Nakon što se pogledala stranica, uvidio njezin način rada, započeto je njezino uređivanje. Otvoren je kod, te izbrisan nepotreban dio koda, a napisan novi. Napravljena je i JS datoteka u kojoj se nalaze funkcije.

```
function getBook(isbn){
    const url = 'http://openlibrary.org/isbn/'+isbn+'.json'
    return $.getJSON(url, function(book) { addCard(isbn, book); });
}
```

Prva funkcija *getBook* uzima id koji se unese kao ISBN na stranici za unos knjige. Te na stranici[9] pronađe knjigu s tim istim ISBN-om i vrati kao JSON dogovor. Jquery vrati *url* te podatke kao funkciju *addCard*.

```
function addCard(isbn, book){
    var cover = "http://covers.openlibrary.org/b/id/" + book.covers[0] + "-
M.jpg";
    var title = book.title;
    var card = $("#"+isbn);
    var cover = '<center><img src="' + (cover) + '" width="180"
height="260"></img></center>'
    card.append(cover);
    card.append(book.title);
    if(book.hasOwnProperty('by_statement')){
        card.append("<br>" + book.by_statement);
    }
    else if(book.hasOwnProperty('authors')){
        card.append("<br>" + book.authors[0]);
    }
}
```

[9] http://openlibrary.org/

U drugoj funkciji *addCard* nalaze se dva parametra *isbn* i *book*. Unutar funkcije se ispiše *isbn* i svi podaci koji su vraćeni kao JSON. Book su svi podaci koji su vraćeni kao JSON te preko .*cover*(omot) pristupamo omotu. U varijablu *cover* sprema se omot od knjige a preko niza se uzme omot na poziciji „0“ ako ih ima više. Na isti način u varijablu *title* se spremi naziv knjige. Kako se ima više *div* elemenata, preko varijable *card* kažemo koji točno *div* želimo, *id* u elementu *div* i *isbn* u varijabli *card* su isti.

Preko *append* dodajemo omot i naslov na kraj skripte koja se nalazi na *html* stranici. Kako neke od knjiga imaju zapisane *autora* a druge *by statement*, preko if uvjeta i *hasOwnProperty* kažemo ako ima *by_statment* ispiši njega, a u slučaju da ga nema ispiši *autora*.

```
function addInputISBN(){
    $("#inputISBN").on('keypress', function (event) {
        if (event.key === 'Enter') {
            var newISBN = $("#inputISBN").val();
            var bookHolder = '<div class="col-4"><span class="image"
            id="'+newISBN+'"></span></div>';
            $("#bookshelf").append(bookHolder);
            getBook(newISBN);

            $('#'+newISBN).on('click', function(){
                $('#'+newISBN).parent().remove();
            });
        }
    });
}
```

Treća funkcija *addInputISBN* , je namijenjena za unos novoga ISBN-a. Na HTML stranici postoji *div* element za unos knjige preko *ISBN-a*. Njegov id je isti kao naziv funkcije pa funkcija zna što i gdje da radi. On ('keypress'). je događaj (eng. *event*) koji nam kaže ako se nešto desi, u ovome slučaju pritisak bilo kojeg dugmeta na tastaturi pozove ovu funkciju i pokrene event. If uvjet će se izvršiti ako je u eventu pritisnuta tipka *Enter*. *NewISBN* je varijabla u koju se sprema novi uneseni ISBN. Varijabla *bookHolder* dodaje novi *div* element za novu knjigu. Na kraju se nalazi on ('click') event koji se izvodi ako se klikne tipka na mišu, nije bitno koja, nije postavljen uvjet. Ako se stisne tipka ona briše knjigu iz popisa. Može se izbrisati samo ona knjiga koja je novo dodana, te se briše i element *parent* kao i cijeli div. To se moralo napraviti pošto bi se izbrisao samo *span* element a div bi ostao, i na tome mjestu bi se nalazilo prazno polje na koje se ne bi mogla smjetiti nova knjiga.

9.3 HTML dio

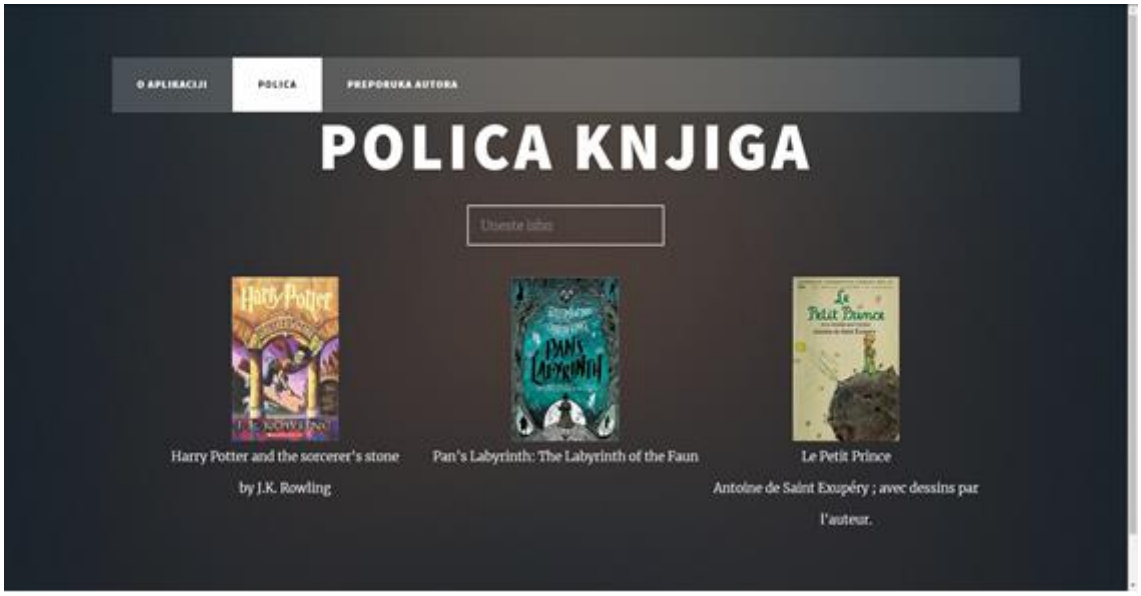
HTML dio koda za stranicu izgleda kao u nastavku:

```
<!DOCTYPE HTML>
<html>
  <head>
    <title>Polica</title>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1,
      user-scalable=no" />
    <link rel="stylesheet" href="assets/css/main.css" />
    <noscript><link rel="stylesheet" href="assets/css/noscript.css"/>
      </noscript>
    <script src="assets/js/jquery.min.js"></script>
    <script src="assets/js/books.js"></script>
  </head>
  <body class="is-preload">
    <div id="wrapper" class="fade-in">
      <div id="intro">
        <nav id="nav">
          <ul class="links">
            <li><a href="generic.html">O aplikaciji</a></li>
            <li class="active"><a href="index.html">Polica</a></li>
            <li><a href="elements.html">Preporuka autora</a></li>
          </ul>
        </nav>
        <h1>Polica knjiga</h1>
          <div><input type="text" id="inputISBN" placeholder="Uneste
isbn"><br></div><script>addInputISBN();</script>
            <div class="box alt">
              <div class="row gtr-50 gtr-uniform" id="bookshelf">
                <div class="col-4"><span class="image" id="978-
0439708180"><script>getBook(document.currentScript.parentNode.id);</script>
</span></div>
                <div class="col-4"><span class="image"
id="9780062414465"><script>getBook(document.currentScript.parentNode.id);</
script></span></div>
                <div class="col-4"><span class="image"
id="9780156503006"><script>getBook(document.currentScript.parentNode.id);</
script></span></div>
          </div>
        </div>
      </div>
    </div>
    <script src="assets/js/jquery.scrollex.min.js"></script>
    <script src="assets/js/jquery.scrolly.min.js"></script>
    <script src="assets/js/browser.min.js"></script>
    <script src="assets/js/breakpoints.min.js"></script>
    <script src="assets/js/util.js"></script>
    <script src="assets/js/main.js"></script>

  </body>
</html>
```

HTML koji se nalazi iznad, predstavlja uz CSS vizualni dio aplikacije. Element *div* koji ima klasu *box alt* je element koji nam odredi granice gdje će se knjige smjestiti, a element *div* klase

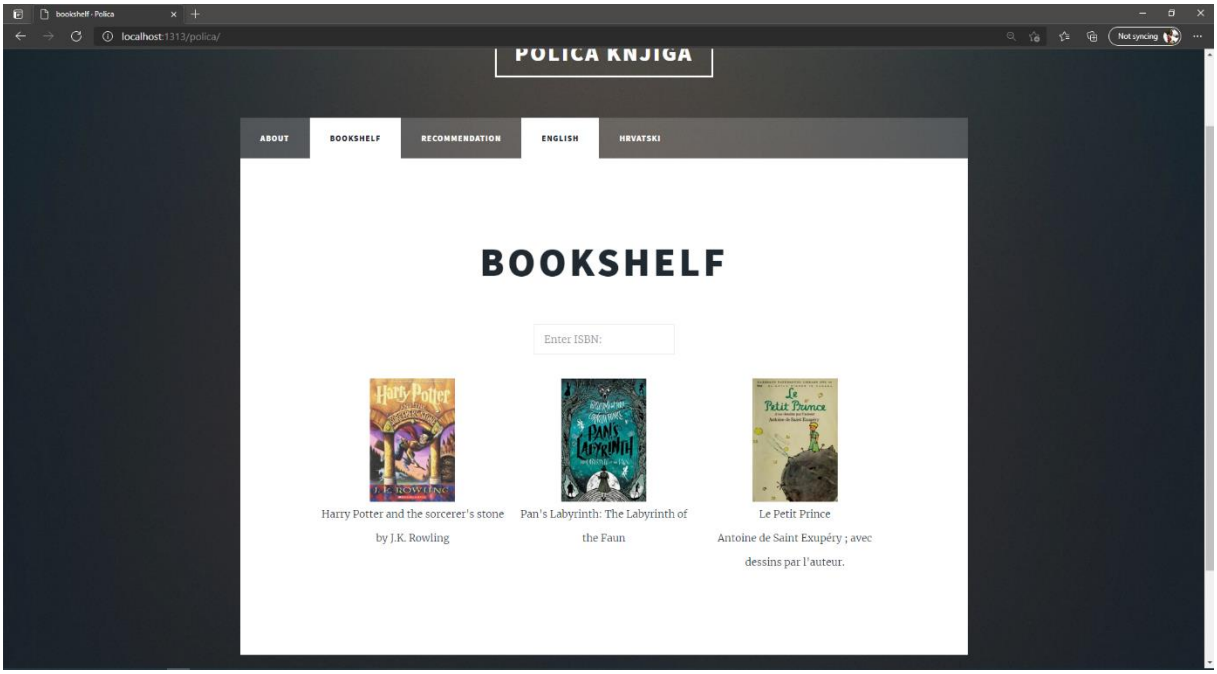
col-4 je element koji nam stavi granice za svaku knjigu i raspodijelio ekran te smjesti tri knjige u jedna red jednako razmaknute. Također unutar toga *div* elementa nalazi se i scripta koja poziva *getBook* funkciju. Ovako izgleda stranica polica knjiga:



Slika 10. Izgled stranice polica HTML

U nastavku možete vidjeti par slika koje prikazuju kako izgleda aplikacija kada se pokrene preko Hugo servera:

Slika 11. Izgled aplikacije početna stranica



Slika 12. Izgled aplikacije stranica unos knjiga engleski prijevod

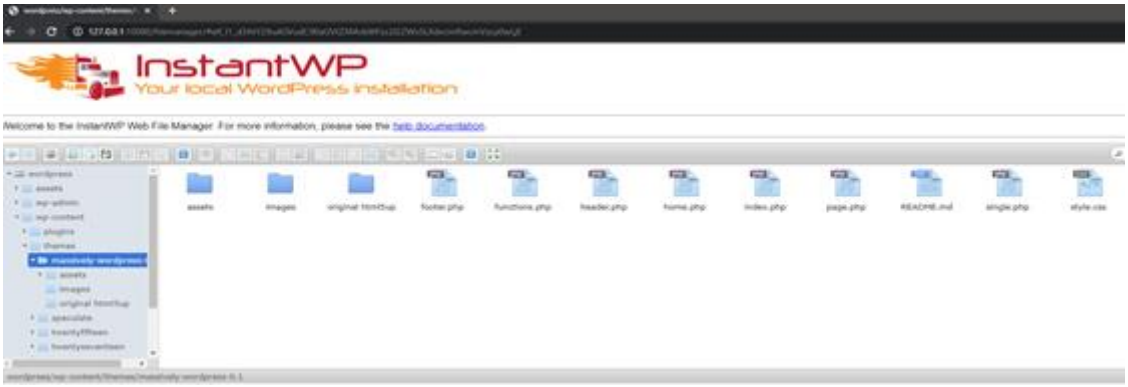
9.4 WordPress

Nakon što je napravljena aplikacija, ista se napravila i u WordPress-u da bi se mogao provesti test i usporediti stranice. Prvo se pokušao instalirati WordPress i postavio za rad. Skinuo se, instalirao te za server skinuo program AMPPS (Apache Web Server, MySQL, PHP, Perl, Python and SOFTACULOUS). Ali postojali su određeni problema s pokretanjem i postavljanjem te se nije uspjelo na taj način, pa se instalirao InstatnWP koji se može pronaći na njihovoj stranici [10]. Instalacija je uspiješno provedena te je program postavljen za rad. Za WordPress smo također pronašli istu temu kao što je iskorištena i za Hugo okvir na GutHub stranici[11].Postavljanje teme u WordPress-u nakon instalacije se realizira tako da se na **Theme**>**Add new**> doda zip koji je skinut s temom, te >**Activate**>**Visit site** i aplikacija je pokrenuta.

Za potrebe testa neće se brisati sadržaj koji je doša s temom već će se samo ubaciti kod koji je gore napisan. Ubacili smo JavaScript dokument, te kod za dodavanje kao i prikaz knjiga dodan je na početnu stranicu. To će se napraviti da bi se imalo što više podataka i teksta za učitavanje u pregledniku da vrijeme učitavanja bude duže i da se uspije vidjeti razliku između

[10] https://instantwp.com/
[11] https://github.com/nickolasnikolic/massively-wordpress/releases/tag/0.1

Hugo okvira i WordPress-a u testovima. Također će se napraviti nova aplikacija i u Hugo okviru s istom temom i kodom kao u WordPress-u da bi uvjeti za test bili isti. Stranica izgleda isto kao i u Hugo okviru a sučelje WordPress-a izgleda ovako:

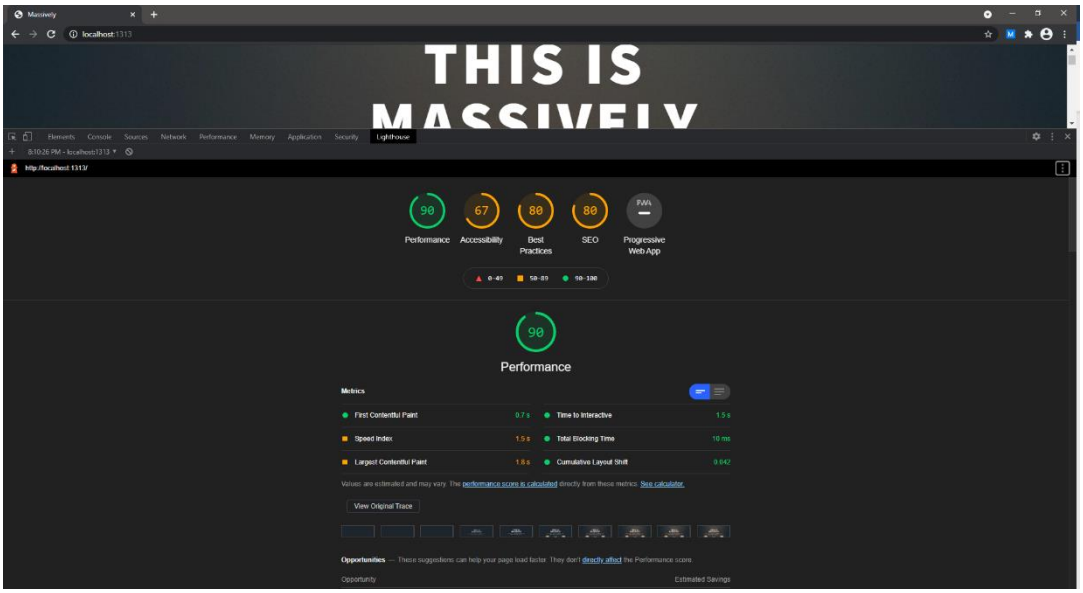


Slika 13. WordPress sučelje

Prvi test je proveden na Hugo okviru, nakon toga u WordPress-u te na kraju pokrenuti kod preko lokalnog python servera i prikazani rezultate u nastavku.

9.5 Testiranje

Prvo će se testirati aplikacija tako što će se pokrenuti preko Hugo okvira na lokalnom serveru, te će se na Chrome pregledniku pokrenuti snimanje performansi i učitavanja kao i pokrenuti Lighthouse alat koji će provjeriti aplikaciju. U nastavku će se moći vidjeti slike testa i rezultata. Prvi test je pomoću alata Lighthouse. Pokrenut je Hugo server te aplikacija preko njega, otvorena u Chrome pregledniku. Otvoren je Inspect element (Pregled elemenata), te u izborniku izabran Lighthouse i pokrenut test. Na sljedećoj slici se mogu vidjeti rezultati provedenog testa.

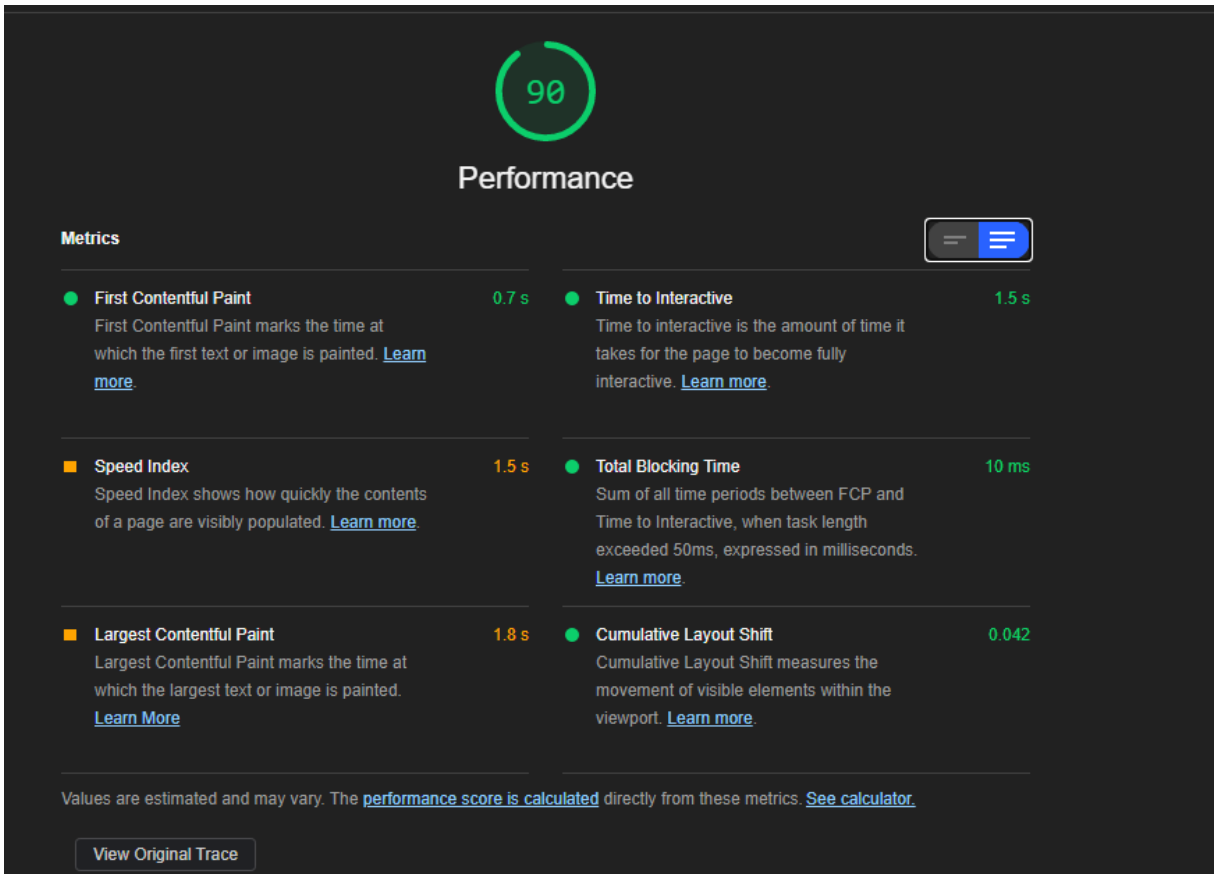


Slika 14. Test Hogo okvira pomoću alata Lighthouse

Na slici se vide rezultati pet parametara koji se dobiju kada napravimo test pomoću alata Lighthouse, a to su:

- **Performanse** (eng. *Performance*) – Mjerenje vremena do interaktivnosti da se može utvrditi kašnjenje kao i pojava prve boje naše aplikacije.
- **Pristupačnost** (eng. *Accessibility*) – Provjerava obične probleme koji bi spriječili korisnika da pristupi sadržaju.
- **Najbolja praksa** (eng. *Best Practices*) – On mjeri sve od upotrebe HTPPS kod aplikacije do ispravnih omjera slika.
- **SEO** – Provjera primjere iz prakse da provjeri dali je web aplikacija vidljiva.
- **Progresivna Web aplikacija** (eng. *Progressive Web App*) – provjera dali je progresivna stranica.

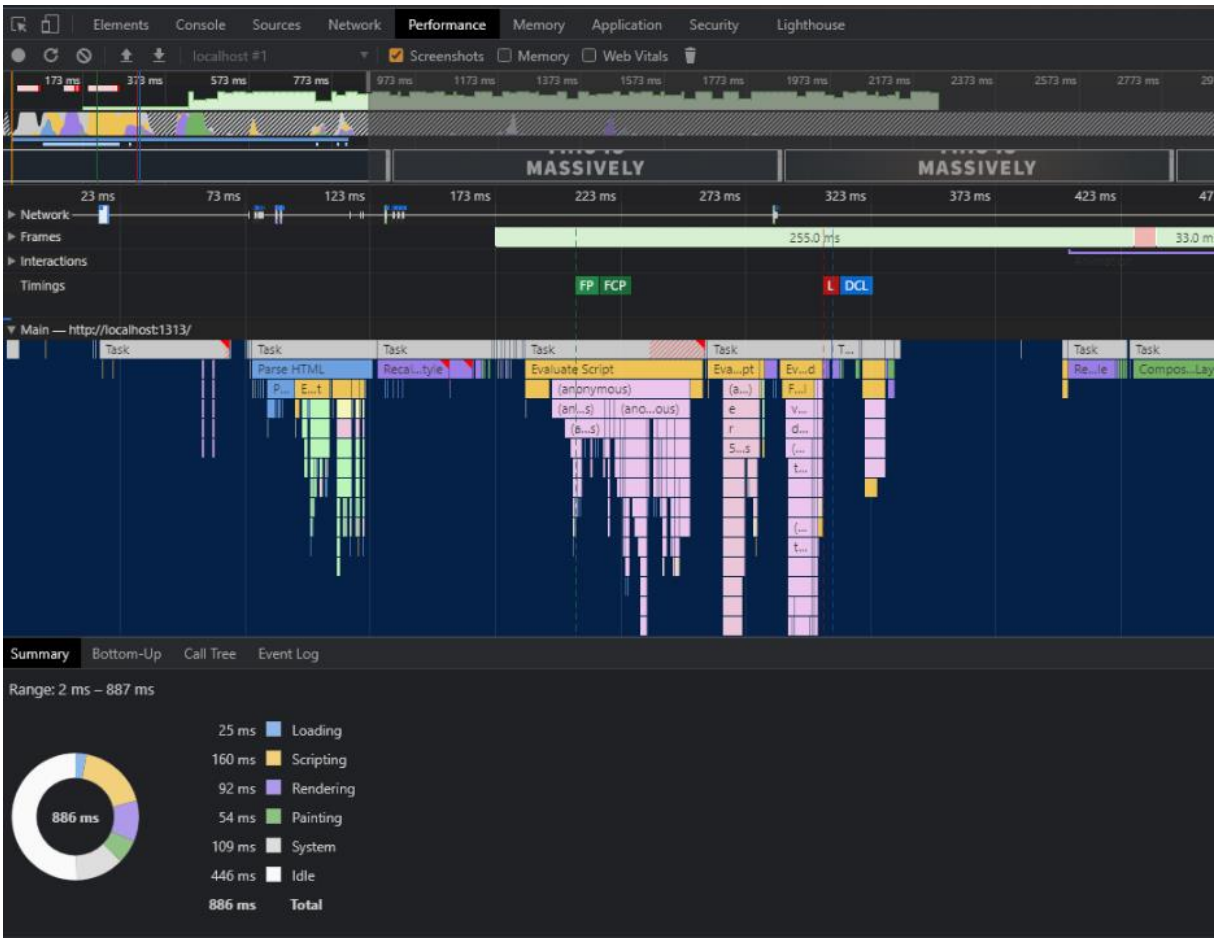
Na sljedećoj slici se vide još detaljnije informacije i objasnit ćemo što nam ti brojevi predstavljaju.



Slika 15. Rezultati prvog testa lightouse Hugo

Na slici gore imamo šest parametara koji prikazuju rezultate tijekom istog testa koji se proveo na slici kore a oni su:

- **Prva sadržajna boja** (eng. *First Contentful Paint*) – Označava vrijeme do pojave prvog teksta ili slike koji su vidljivi korisniku.
- **Index brzine** (eng. *Speed Index*) – Računa brzinu učitavanja sadržaja stranice.
- **Najveća sadržajna slika** (eng. *Largest Contentful Paint*) – prikazuje vrijeme prikaza najvećeg teksta ili slike.
- **Vrijeme do interakcije** (eng. *Time to Interactive*) – Označava vrijeme prije nego korisnik može u potpunosti pristupiti stranici i njenom sadržaju.
- **Ukupno vrijeme blokiranja** (eng. *Total Blocking Time*) – Zbroj vremen između FCP i vremena interakcije.
- **Kumulativni pomak izgleda** (eng. *Cumulative Layout Shift*) – Mjeri vrijeme pomaka vidljivih elemenata u vidnom polju.

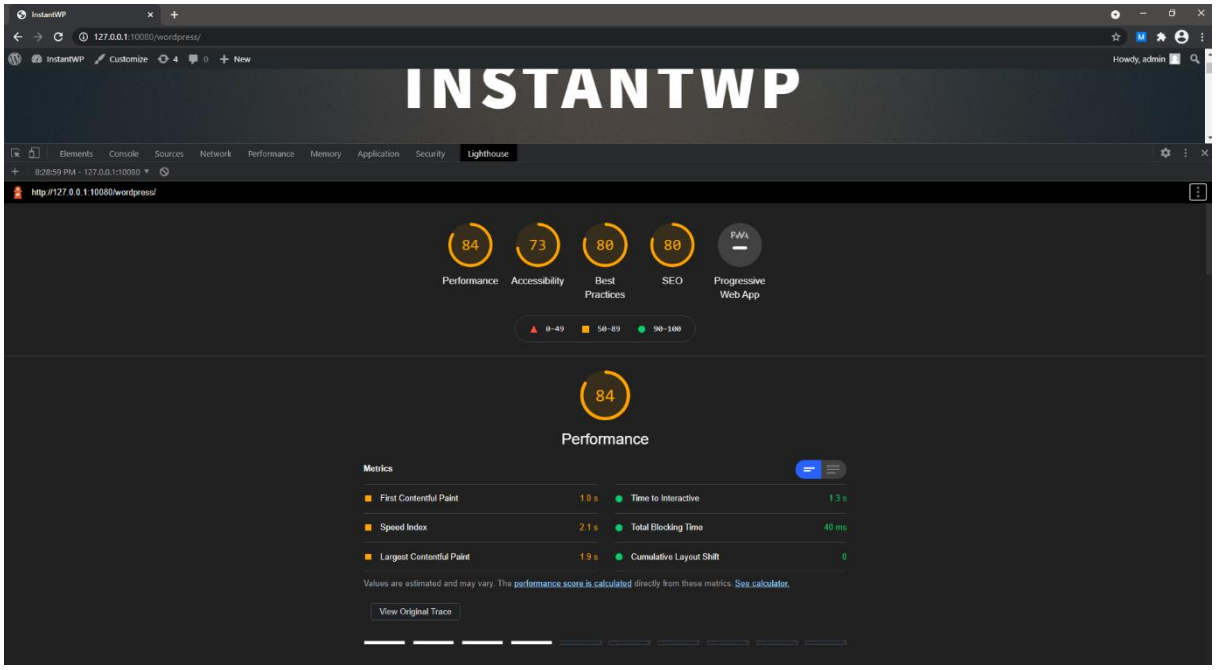


Slika 16. Rezultati testa performansi brzine Hugo okvira

Rezultat testa performansi prikazuje da je ukupno vrijeme učitavanja 886 milisekundi. Iznad možemo vidjeti i grafički prikaz učitavanja kao i rezultate, a dole su ispisani u brojevima. A ukupno vrijeme je raspoređeno na ovih šest stavki.

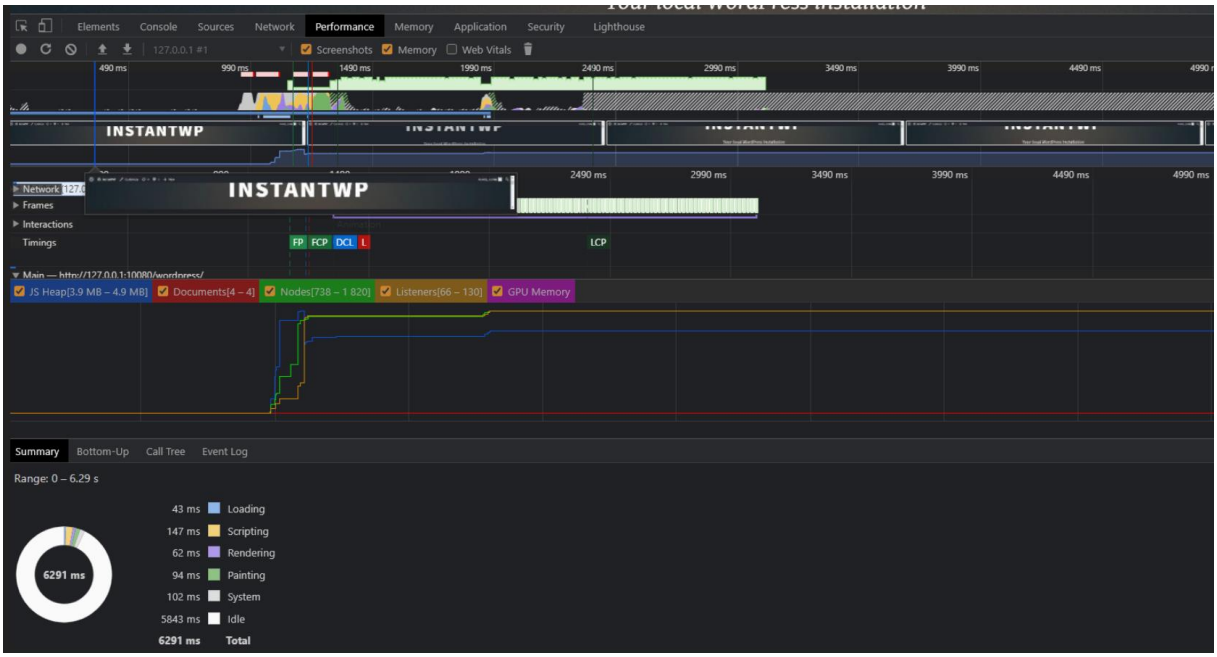
- **Učitavanje** (eng. *Loading*) – Vrijeme učitavanja stranice
- **Skripte** (eng. *Scripting*) – Vrijeme izvršavana skripte
- **Prikazivanje** (eng. *Rendering*) – Vrijeme prikazivanja
- **Slika** (eng. *Painting*)
- **Sustav** (eng. *System*)
- **Mirovanje / čekanje sustava** (eng. *Idle*)
- **Ukupno** – Vrijeme do korištenja stranice

Sljedeća slika nam prikazuje aplikaciju u WordPress-u gdje je napravljen također Lighthouse test.



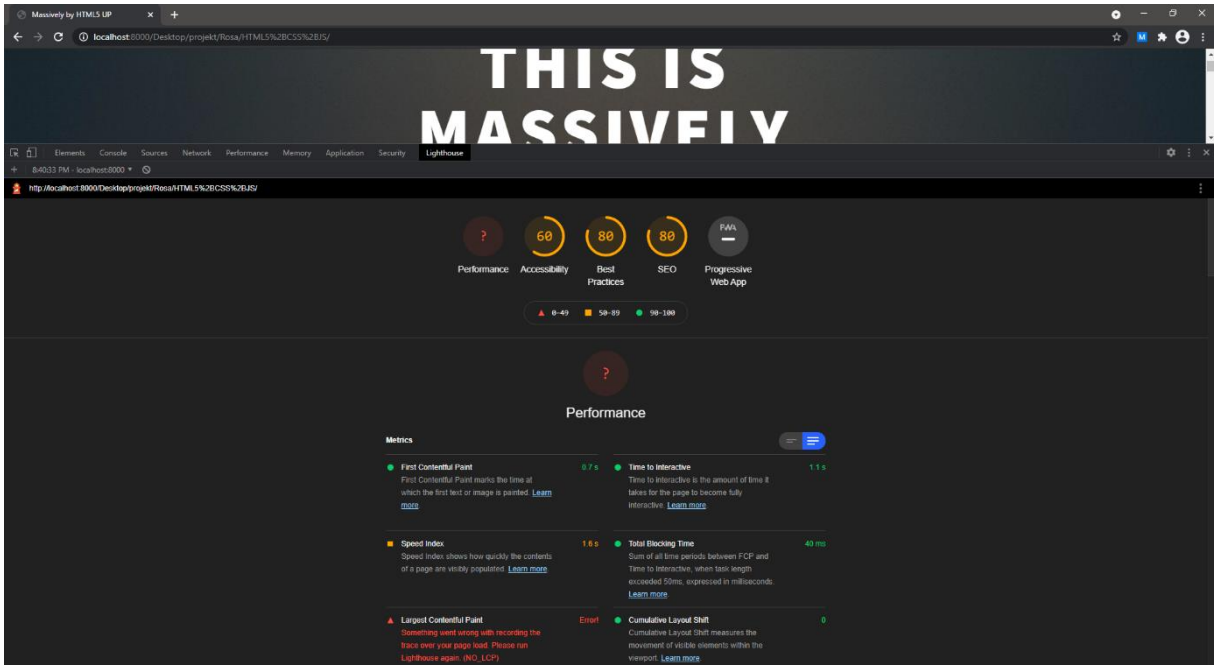
Slika 17. Rezultati testa Lighthouse WP

Ovdje se mogu vidjeti rezultate testa performansi koji su se napravili u pregledniku Chrome za WordPress u grafičkom prikazu.

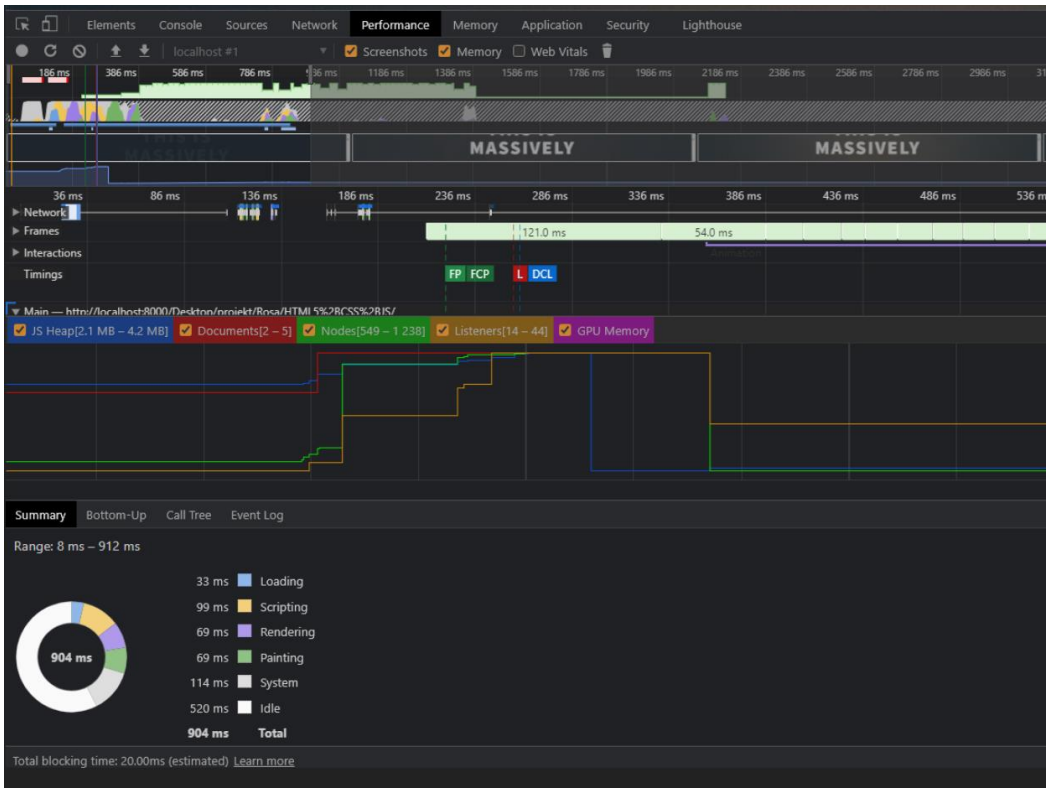


Slika 18. Rezultati testa performanse WordPress-a

Rezultate testa će se prokomentirati na kraju kad se prikažu u tablici i usporede.



Slika 19. Rezultati testa lighthouse na serveru python



Slika 20. Rezultati testa performanse na python serveru

U sljedeće dvije tablice se mogu vidjeti uspoređeni rezultati koje smo gore prikazali.

Tablica 3. Prikaz rezultata testa Lighthouse

	Hugo		**WordPress**		**Python**	
	Prvo učitavanje	***Peto učitavanje***	***Prvo učitavanje***	***Peto učitavanje***	***Prvo učitavanje***	***Peto učitavanje***
Performanse (eng. *Performance*)	90	-	84	-	-	87
Pristupačnost (eng. *Accessibility*)	67	67	73	62	60	60
Najbolja praksa (eng. *Best Practices*)	80	80	80	80	80	80
SEO	80	80	80	80	80	80
Progresivna Web aplikacija (eng. *Progressive Web App*)	-	-	-	-	-	-
Prva sadržajna boja (eng. *First Contentful Paint*)	0,7 s	0,7 s	1 s	0,9 s	0,7 s	0,8 s
Index brzine (eng. *Speed Index*)	1,5 s	1,6 s	2,1 s	2,6 s	1,6 s	1,8 s
Najveća sadržajna slika (eng. *Largest Contentful Paint*)	1,8 s	-	1,9 s	-	-	1,9 s
Vrijeme do interakcije (eng. *Time to Interactive*)	1,5 s	1,5 s	1,3 s	1,4 s	1,1 s	1,2 s
Ukupno vrijeme blokiranja (eng. *Total Blocking Time*)	10 ms	40 ms	40 ms	20 ms	40 ms	80 ms
Kumulativni pomak izgleda (eng. *Cumulative Layout Shift*)	0,042	0	0	0	0	0

Prva tablica prikazuje rezultate koji su nastali prilikom korištenja alata Lighthouse unutar preglednika u testiranju aplikacije a druga tablica prikazuje rezultate prilikom testa performansi u pregledniku Chrome.

Tablica 4. Prikaz testa performanse

	Hugo		**WordPress**		**Python**	
	Prvo učitavanje	***Peto učitavanje***	***Prvo učitavanje***	***Peto učitavanje***	***Prvo učitavanje***	***Peto učitavanje***
Učitavanje (eng. *Loading*)	25 ms	26 ms	43 ms	34 ms	33 ms	30 ms
Skripte (eng. *Scripting*)	160 ms	165 ms	147 ms	136 ms	99 ms	89 ms
Prikazivanje (eng. *Rendering*)	92 ms	63 ms	62 ms	39 ms	69 ms	61 ms
Slika (eng. *Painting*)	54 ms	44 ms	94 ms	25 ms	69 ms	43 ms
Sustav (eng. *System*)	109 ms	76 ms	102 ms	112 ms	114 ms	102 ms
Mirovanje sustava (eng. Idle)	446 ms	289 ms	5843 ms	443 ms	520 ms	507 ms
Ukupno	886 ms	663 ms	6291 ms	789 ms	904 ms	832 ms

Prva tablica nam prikazuje podatke koji pokazuju da je Hugo okvir bolji od WordPress-a u skoro u svim poljima u prvom učitavanju aplikacije. Tako u prvom polju performanse kao i Prva sadržajna boja se može vidjeti da je Hugo brži od WordPress-a, što znači da se sadržaj brže pojavljuje i učitava na stranici, a index brzine to samo potvrđuje te je brzina učitavanja puno veća. Jedino što me iznenadilo u ovome testu je rezultat vremena do interakcije gdje je malo bolje vrijeme imao WordPress. Hugo se brže učitava te se prije učita u preglednik ali po ovome rezultatu WordPress ipak ima malo bolje vrijeme interakcije. Možemo pretpostaviti da možemo pristupiti stranici i krenuti na njoj raditi dok još nije cijela učitana u preglednik za što ipak treba više vremena od Hugo okvira. Također imamo i rezultate petog učitavanja stranice gdje se može vidjeti da su rezultati približno isti kao kod prvog učitavanja i nisu se previše mijenjali. Test za python je prikazao približno slične rezultate u prvo i petom učitavanju te ga po

rezultatima možemo smjestiti između Hugo okvira i WordPress-a. U petom učitavanju nismo dobili rezultate za performanse kao i za najveću sadržajnu sliku, dok smo kod python dobili rezultat za performanse koji nismo imali u prvom učitavanju. Test se provodio na lokalnom računalu te je i to moglo utjecati na rezultate testa, ali su svi testovi urađeni u istim uvjetima.

Druga tablica prikazuje da je ukupno vrijeme učitavanja stranice kod Hugo okvira puno brže od WordPress-a. Tako za Učitavanje stranice Hugo okviru je potrebno duplo manje vremena kao i za prikaz prve slike. Ostali parametri su približno isti osim Idle gdje je WordPress imao nagori rezultat te je uvelike utjecalo na njegov rezultat u ovome testu. Nakon petog učitavanja možemo vidjeti da se vrijeme kod WordPress-a drastično popravilo te da je u nekim pokazateljima i bolji od Hugo okvira. Test na python serveru ima približno slične vrijednosti te se ovoga puta našao na zadnjoj poziciji po ukupnom vremenu za razliku od prvog učitavanja. WordPress je pokazao najveći napredak i najviše se smanjila vrijednost mirovanje sustava.

Na slikama koje su prikazane gore može se vidjeti i na grafičkom dijelu prikaza rezultata da se stranica kod Hugo okvira brže učitava i prikazuje nego što je slučaj kod WordPress-a. Ovime je Hugo dokazao svoju prednost što se tiče brzine.

Iz priloženih tablica se može vidjeti uz usporedbu vrijednosti rezultata da Hugo okvir ima malo bolje rezultate od WordPressa kao i od koda koji smo pokrenuti na python serveru. Ovime je Hugo dokazao da je brzina jedna od prednosti koja ga izdvaja od drugih, te možemo reći da je Hugo najbrži okvir od tri što su testirana.Također dok se kreirala aplikacija za test u Hugo okviru kao i WordPress-u, uočilo se da je lakše instalirati, pokrenuti server i krenuti s radom kod Hugo okvira neko WordPress-a. Također lakše je bilo umetnuti temu i urediti je u Hugo okviru nego kod WordPress-a. Najveća razlika se primijeti u Idle kao i učitavanju gdje je Hugo puno bolji od preostala dva rješenja što pokazuje test. Moje osobno mišljenje na osnovi provedenih testova, bolje je koristi Hugo za izradu manjih i jednostavnijih aplikacija zbog njegove brzine i jednostavnosti, a za složenije aplikacije mogu se koristiti drugi programi koji su namijenjeni za tu svrhu. Test se proveo na osobnom računalu te je možda i to utjecalo na rezultat testa, ali svi su testovi provedeni na istom kodu i na istom računalu i pod istim uvjetima. Zbog toga rezultate možemo smatrati relevantnima.

10. Prednosti i nedostaci u okviru Hugo

Kroz izradu praktičnog dijela rada uočene su neke od prednosti i nedostatke okvira Hugo. Kao glavni nedostatak okvira Hugo mogu se navesti nemogućnost izrade složenih aplikacija. Njegova najveća prednost je kao što smo mogli vidjeti u testu koji smo proveli, brzina. Izradom jednostavnijih aplikacija brzina dolazi do izražaja, kao i lakoća izrade jednostavnih aplikacija. Zbog svoje specifičnosti kao statički generator sigurnost je jedna od prednosti Hugo okvira, a to se najviše odnosi na činjenicu što nema baze koja bi bila meta napada. Kao prednost Hugo okvira mogu navesti i jednostavnost instalacije kao i kreiranja same stranice, bloga ili aplikacije. Što je kroz primjer izgradnje i testiranja uočeno. Vrlo jednostavno je instaliran Hugo okvir te primjenjena tema, kreirana aplikacija i provedeno testiranje, što nije bio slučaj kod korištenja WordPress-a. S WordPress-om smo imali manje probleme kod instalacije i pokretanja servera i zbog nepoznavanja samog programa trebalo je puno više vremena da se primjeni tema i izradi aplikacija te se ista testira. Kao još jedan od nedostataka možemo navesti i nepostojanje velike količine dodataka koji su dostupni za Hugo okvir.

Prema Atishay, (2019) ovo su još neke od prednosti. **Velika zajednica** povezana na forumima koja je uvijek spremna dati savjet i pomoći u optimizaciji stranice. **Samostalnost** je jedna od prednosti, ranije smo naveli da je izostanak dodataka nedostatak ali, može biti i prednost. Izostanak dodataka omogućava veću fleksibilnost, kao i mogućnosti održavanja. Tako sa svakom novom verzijom nemamo problema s migracijom dodataka. **Skalabilnost** kao još jedna od prednosti omogućava da obradimo veliki broj stranica s višejezičnim sadržajem i velikim brojem korisnika, kao i podržanost velikog broja ulaznih i izlaznih formata.

11. Zaključak

Okvir Hugo je jednostavan i prema testiranju najbrži statički generator (od tri testirana) koji je namijenjen za izradu blogova, informativnih stranica. U njemu se mogu napraviti i jednostavne aplikacije. Test koji je proveden na kraju je ukazao da je Hugo okvir najbrži kao i da je dosta jednostavan za izradu jednostavnih rješenja. Nakon petog učitavanja može se vidjeti da i WordPress kao i python su se približili Hugo okviru po brzini. Moj osobni stav na osnovi rezultata koje smo dobili je da ipak za jednostavna rješenja bolji Hugo okvir dok bi za kompleksija koristio npr. WordPress. Statički generatori imaju dosta prednosti kod izrade blogova, statičkih stranica, jednostavnih statičkih aplikacija nad drugim rješenjima te polako postaju sve popularniji. Sigurnost kao jedna od bitnijih stavki kada je u pitanju zaštita podataka kao i zaštita samih aplikacija. Tu vidimo veliku prednost statičkih generatora u usporedbi s drugih tradicionalnim rješenjima koja koriste baze. Nedostatak statičkih generatora kao i Hugo okvira je nemogućnost pravljenja velikih i kompliciranijih aplikacija, ali kako nisu niti namijenjeni za takve stvari pitanje je, da li je to zaista nedostatak. U svojem polju djelovanja generatori su najbolji, te dosta jednostavni za korištenje. Pomoću Hugo okvira svoje blogove i stranice mogu napraviti i prosječni korisnici (koji nisu IT stručnjaci) zbog njegove jednostavnosti kao i dostupnih materijala koji su nam lako dostupni. Na kraju se može zaključiti da je okvir Hugo pogodan za izradu manjih i jednostavnijih aplikacija kao i blogova ili informativnih stranica te je preporuka autora da se za takve Web stranice koriste statički generatori (u ovome slučaju Hugo) nego neka druga rješenja.

Popis literature

Atishay, J. (2019). *Hugo in Action: Static sites and dynamic Jamstack apps* (Version 8). MEAP.

Bodrov-Krukowski, I. (2019). *A Beginner's Guide to Creating a Static Website with Hugo*. SitePoint.

Buckler, C. (2016). *7 Reasons to Use a Static Site Generator - SitePoint*. Sitepoint.Com. Preuzeto 18.04.2021. s https://www.sitepoint.com/7-reasons-use-static-site-generator/

C Davis, S. (2020). *Comparing Static Site Generator Build Times*. Css-Tricks.Com. Preuzeto 18.04.2021. s https://css-tricks.com/comparing-static-site-generator-build-times/

Cloudeflare. (2021). *What is a static site generator?* Cloudflare.Com. Preuzeto 18.04.2021. s https://www.cloudflare.com/en-gb/learning/performance/static-site-generator/

cloudflare. (2021). *What is JAMstack?* Cloudflare.Com. Preuzeto 18.04.2021. s https://www.cloudflare.com/en-gb/learning/performance/what-is-jamstack/

Cresswel, H. (2018). *Working with Data in Hugo*. Harrycresswell.Com. Preuzeto 18.04.2021. s https://harrycresswell.com/articles/data-hugo/

Dimoski, M. (2021). *Modules*. Github.Com. Preuzeto 18.04.2021. s https://github.com/golang/go/wiki/Modules#modules

Filip Media d.o.o. (2020). *Što je WEB aplikacija?* Geek.Hr. Preuzeto 18.04.2021. s https://geek.hr/pojmovnik/sto-je-web-aplikacija/

Fowler, S., & Stanwick, V. (2004). *Web Application Design Handbook*. Morgan Kaufmann.

Gandhi, B. (2020). *How to use Hugo Modules*. Geeksocket.In. Preuzeto 18.04.2021. s https://geeksocket.in/posts/hugo-modules/

Gatsby. (2021). *The Fastest Frontend for the Modern Web | Gatsby*. Gatsbyjs.Com. Preuzeto 18.04.2021. s https://www.gatsbyjs.com/

Gibb, R. (2016). *What is a Web Application?* Blog.Stackpath.Com. Preuzeto 18.04.2021. s https://blog.stackpath.com/web-application/

Gruber, J. (2021). *Daring Fireball*. Daringfireball.Com. Preuzeto 18.04.2021. s https://daringfireball.net/projects/markdown/

Hawksworth, P. (2020). *What is a Static Site Generator? And 3 ways to find the best one*. Netlify.Com. Preuzeto 18.04.2021. s https://www.netlify.com/blog/2020/04/14/what-is-a-static-site-generator-and-3-ways-to-find-the-best-one/

Hogan, B. P. (2020). *Build Websites with Hugo: Fast Web Development with Markdown*. Pragmatic Bookshelf.

Horizont. (n.d.). *Web aplikacije - prednosti i nedostatci*. Horizont.Com. Preuzeto 18.04.2021. s https://www.horizont.com.hr/web-aplikacije---prednosti-i-nedostatci-3-blog

Hugo. (2019). *Host on Netlify*. Gohugo.Io. Preuzeto 18.04.2021. s https://gohugo.io/hosting-and-deployment/hosting-on-netlify/

Hugo. (2021). *The world's fastest framework for building websites | Hugo*. Gohugo.Io. Preuzeto 18.04.2021. s https://gohugo.io/

Johnston, J. (2020). *A beginners guide to web application development (2021)*. Budibase.Com. Preuzeto 18.04.2021. s https://www.budibase.com/blog/web-application-development/

Kolowich Cox, L. (2020). *Web Design 101: How HTML, CSS, and JavaScript Work*. Blog.Hubspot.Com. Preuzeto 18.04.2021. s https://blog.hubspot.com/marketing/web-design-html-css-javascript

Maroli, A., Taillandier, F., & Rogers, M. (2021). *Jekyll • Simple, blog-aware, static sites | Transform your plain text into static websites and blogs*. Jekyllrb.Com. Preuzeto 18.04.2021. s https://jekyllrb.com/

netlify. (2021). *Static Site Generators Build Performance Testing*. Netlify.App. Preuzeto 18.04.2021. s https://ssg-build-performance-tests.netlify.app/

Netlify. (2021). *Jamstack*. Jamstack.Com. Preuzeto 18.04.2021. s https://jamstack.org/what-is-jamstack/

Rinaldi, B. (2021). *Pick the Best Static Site Generator for 2021 - Snipcart*. Snipcart.Com. Preuzeto 18.04.2021. s https://snipcart.com/blog/choose-best-static-site-generator

Shaleynikov, A. (2020). *Static Site Generators Overview: Gatsby vs. Hugo vs. Jekyll*. Dzone.Com. Preuzeto 18.04.2021. s https://dzone.com/articles/static-site-generators-overview-gatsby-vs-hugo-vs

Shklar, L. (2009). *Web Application Architecture: Principles, Protocols and Practices* (2nd ed.). Wiley. https://learning.oreilly.com/library/view/web-application-architecture/9780470518601/ch01.html

StaticApps. (2021). *StaticWeb - Most Common Web Applications*. Staticapps.Org. Preuzeto 18.04.2021. s https://www.staticapps.org/

Sugandha. (2020). *Difference Between Web application and Website*. Geeksforgeeks.Org.

Preuzeto 18.04.2021. s https://www.geeksforgeeks.org/difference-between-web-application-and-website/

Svitla. (2019). *Key Differences: Website vs Web Application*. Svitla.Com. Preuzeto 18.04.2021. s https://svitla.com/blog/website-vs-web-application

Popis slika

Slika 1. Proces izrade stranice, Izvor: (Hawksworth, 2020)..... 6
Slika 2. Usporedba generatora po brzini, Izvor: (C Davis, 2020).....10
Slika 3. Primjer Markdown-a, Izvor: (Gruber, 2021) 12
Slika 4. Struktura stranice, Izvor: (Hogan, 2020).....15
Slika 5. Kreiranje stranice u Netlify, Izvor: (Hugo, 2019) 19
Slika 6. Ispis zemalja, Izvor (Cresswel, 2018).....20
Slika 7. Struktura datoteka podataka.....22
Slika 8. Terminal Hugo instal.....24
Slika 9. Kreiranje nove stranice 24
Slika 10. Izgled stranice polica HTML.....28
Slika 11. Izgled aplikacije početna stranica.....28
Slika 12. Izgled aplikacije stranica unos knjiga engleski prijevod 29
Slika 13. WordPress sučelje.....30
Slika 14. Test Hogo okvira pomoću alata Lighthouse 31
Slika 15. Rezultati prvog testa lightouse Hugo.....32
Slika 16. Rezultati testa performansi brzine Hugo okvira 33
Slika 17. Rezultati testa Lighthouse WP 34
Slika 18. Rezultati testa performanse WordPress-a.....34
Slika 19. Rezultati testa lighthouse na serveru python.....35
Slika 20. Rezultati testa performanse na python serveru 35

Popis tablica

Tablica 1. Razlika između web aplikacija i web stranica, Izvor: (Sugandha, 2020) 4
Tablica 2. Usporedba Generatora, Izvor: (Shaleynikov, 2020)..... 9
Tablica 3. Prikaz rezultata testa Lighthouse 36
Tablica 4. Prikaz testa performanse 37

Prilozi

1. Rezultati testa Lighthouse prvo i peto učitavanje u pdf formatu