Usporedba pristupa za izradu web aplikacija

Nazlić, Jagoda

SVEUČILIŠTE U SPLITU
PRIRODOSLOVNO MATEMATIČKI FAKULTET

DIPLOMSKI RAD

USPOREDBA PRISTUPA ZA IZRADU WEB APLIKACIJA

Jagoda Nazlić

Split, rujan 2023.

Temeljna dokumentacijska kartica

Sveučilište u Splitu Diplomski rad

Prirodoslovno-matematički fakultet

Odjel za Informatiku

Ruđera Boškovića 33, 21000 Split, Hrvatska

USPOREDBA PRISTUPA ZA IZRADU WEB APLIKACIJA

Jagoda Nazlić

SAŽETAK

Tema ovog rada je usporedna analiza različitih pristupa prilikom izrade web aplikacija. Rad opisuje faze izrade web aplikacije, ali i različite tehnologije kojima je izrada omogućena. Osnovna usporedba tehnologija proizlazi iz dva temeljno različita pristupa, pristup s prilagođenim kodom u odnosu na pristup izrade pomoću sustava za upravljanje sadržajem. Rad je potkrepljen s dva prototipa web stranica napravljenih navedenim pristupima.

Ključne riječi: Web aplikacija, WordPress, Okviri za izradu web aplikacija, CMS alati

Rad je pohranjen u knjižnici Prirodoslovno-matematičkog fakulteta, Sveučilišta u Splitu

Rad sadrži: 51 stranicu, 20 grafičkih prikaza, dvije tablice i 21 literaturni navod. Izvornik je na hrvatskom jeziku

Mentor: **Doc. dr. sc. Divna Krpan**, *docent Prirodoslovno-matematičkog fakulteta, Sveučilišta u Splitu*

Ocjenjivači: **Doc. dr. sc. Divna Krpan**, *docent Prirodoslovno-matematičkog fakulteta, Sveučilišta u Splitu*

Doc. dr. sc. Goran Zaharija, *docent Prirodoslovno-matematičkog fakulteta, Sveučilišta u Splitu*

Doc. dr. sc. Monika Mladenović, *docent Prirodoslovno-matematičkog fakulteta, Sveučilišta u Splitu*

Basic documentation card

University of Split

Master Thesis

Faculty of Science

Department of Informatics

Ruđera Boškovića 33, 21000 Split, Hrvatska

COMPARISON OF WEB APPLICATION DEVELOPMENT APPROACHES

Jagoda Nazlić

ABSTRACT

The topic of this thesis is a comparative analysis of different approaches to the development of web applications. The paper describes the stages of making a web application, but also the different technologies that make it possible. The basic technology comparison stems from two fundamentally different approaches, the custom code approach versus the build approach using a content management system. The work was supported by two prototype websites created using the approaches mentioned.

Key words: Web application, WordPress, Web application frameworks, CMS tools

Thesis deposited in library of Faculty of science, University of Split

Thesis consists of: 51 pages, 20 figures, two tables and 21 references. Original language is Croatian

Supervisor: **Doc. dr. sc. Divna Krpan**, *Assistant Professor of Faculty of Science, University of Split*

Reviewers: **Doc. dr. sc. Divna Krpan**, , *Assistant Professor of Faculty of Science, University of Split*

Doc. dr. sc. Goran Zaharija, *Assistant Professor of Faculty of Science, University of Split*

Doc. dr. sc. Monika Mladenović, *Assistant Professor of Faculty of Science, University of Split*

ZAHVALA

Zahvaljujem se svojoj mentorici doc.dr.sc. Divni Krpan jer je podržavala moje ideje i interese te me savjetovala i pronalazila načine kako bi ih predočila u radu.

Također se zahvaljujem svojoj obitelji i prijateljima na neizmjernoj i stalnoj podršci tijekom realiziranja ovog rada.

Sadržaj

Uvod

Ovaj diplomski rad razvijen je s ciljem ispitivanja današnjih tehnologija, načina izrade i životnog ciklusa tipične web aplikacije. Naglasak je na usporedbi alata koji omogućavaju razvoj pisanjem koda i onih koji zahtijevaju vrlo malo ili ništa programiranja (eng. *code* i *no-code tools*).

Općeniti razvoj web-a, pa tako i sadržaja na njemu, doveo je do eksponencijalnog širenja broja različitih tehnologija za izradu web aplikacija. U moru softvera, alata i okruženja za razvoj, teško se odlučiti na konačan odabir, te pri tom jamčiti učinkovitost i trajnost funkcionalnosti aplikacije.

Ipak, neovisno o okruženju u kojem se razvija, životni ciklus svake web aplikacije sadrži iste faze razvoja pomoću kojih je moguće oformiti referentne točke kako bi stigli do konačne isporuke i održavanja.

Motiv istraživanja i analize predstavljene u radu proizlazi iz današnjeg standarda postojanja web promidžbe sadržaja i potrebe za pronalaskom što održivijih metoda kako bi se isto učinkovitije i jednostavnije ostvarivalo. Cilj je opisati i približiti moderne tehnologije za izradu web aplikacija i web stranica kao i usporediti moguće pristupe.

Struktura rada prati uobičajene zahtjeve klijenata, te pregled okruženja i tehnologija kao i mogućih mjera zaštite u svrhu stvaranja sigurne web aplikacije. Analiza je u konačnici potkrijepljena praktičnim primjerom izrade dviju web stranica napravljenih u različitim okruženjima.

1. Izrada web aplikacije

Svjetska mreža (eng. *World Wide Web*) postala je javno dostupna 90-ih godina prošlog stoljeća čime se pojavila mogućnost šireg i sveobuhvatnijeg korištenja Interneta. Temeljena je na klijent-poslužitelj računarstvu, koje počiva na tri fundamentalna koncepta: metodi za imenovanje i referiranje na dokumente (URL), jeziku za pisanje dokumenata (HTML) i protokolu po kojemu će klijent i poslužitelj komunicirati (HTTP) [1]. URL identificira IP adresu računala, dokument u sustavu tog računala i protokol za komunikaciju s tim objektom. Dokumenti su u ovom slučaju upravo datoteke napisane u HTML-u, a u njima su upute što će se i na koji način prikazati na pregledniku. Navedeno je moguće zbog komunikacijskog zahtjev-odgovor protokola - HTTP, koji definira osam osnovnih operacija među kojima su najpoznatije GET i POST.

U skladu s ovim načelima, mnogi su programski jezici razvijeni upravo kako bi preglednik ili server, uz prikaz podataka, mogao izvršavati skripte što je i omogućilo razvoj web aplikacija, te Internet učinilo platformom za kreiranje i implementaciju istih.

Web aplikacija je računalni program za obavljanje određenih funkcija koji se pokreće preko Interneta koristeći web preglednik [1]. Može služiti za izvršavanje funkcija širokog raspona - od relativno jednostavnih, do visoko zahtjevnih aplikacija u vidu obavljanja kupnje, prodaje, upravljanja zalihama u stvarnom vremenu i slično [2].

Važno je spomenuti da se paralelno s razvojem Interneta, zbog robusnosti, nepreglednosti i neodrživosti različitih programa, pojavila potreba za granom računarstva koja će izvršavati i definirati procese izrade, testiranja i implementacije programa. Stoga je 1968. nastalo programsko inženjerstvo koje nudi sistematske pristupe razvoju programa temeljene na standardnim komponentama, modularnom dizajnu i definiranim procesima. Upravo problemi s kojima se suočava programsko inženjerstvo, te njegovi pristupi i prijedlozi usko se vežu i za razvoj web aplikacija kako bi se osigurala i optimizirala sama realizacija i održivost programa kao proizvoda.

Ovo poglavlje obuhvaća pregled nekoliko mogućih pristupa programskog inženjerstva pri izradi web aplikacije, ali i opis razvojnih faza životnog ciklusa iste.

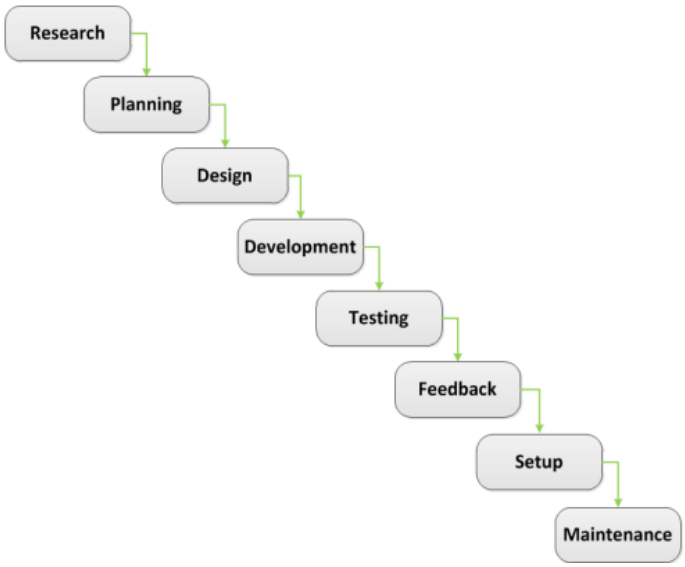
1.1. Pristupi razvoju web aplikacija

Kreiranje web aplikacije započinje odabirom pristupa, odnosno paradigme po kojoj će se podijeliti i odvijati razvojne faze. Ovaj odabir općenito ovisi o zahtjevima koje web aplikacija treba ispunjavati, kao i o timu programera koji je razvija.

Kroz povijest razvoja programskog inženjerstva, nastale su različite metodologije koje definiraju skup pravila i uputa namijenjenih istraživanju, planiranju, oblikovanju, razvoju, testiranju, isporuci te održavanju proizvoda. Kako tih metodologija danas ima vrlo mnogo, u ovom poglavlju opisat će se pet najčešće primjenjivanih u razvoju web aplikacija.

Model vodopada (eng. *Waterfall model)*

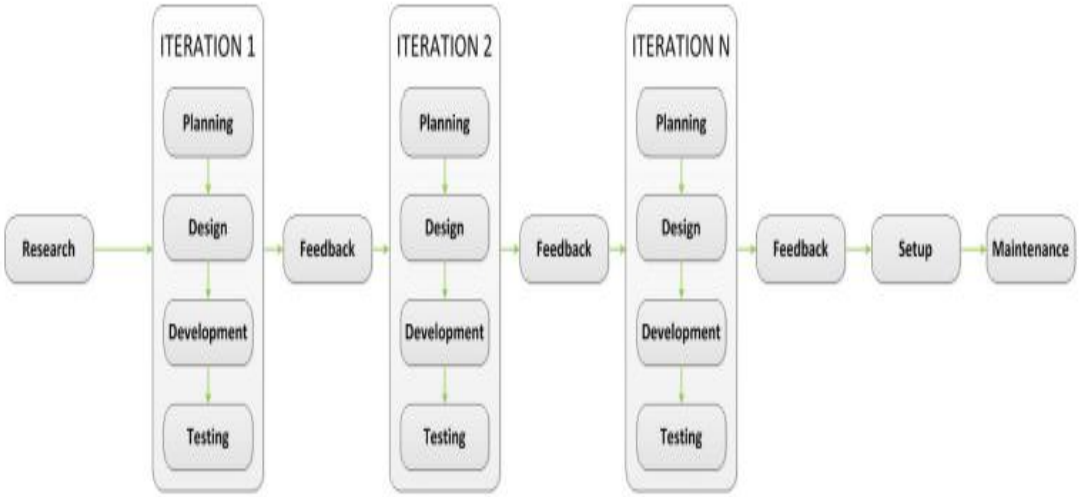
Ovaj je model prvi prepoznati pristup razvoju programa, a naglašava pomno planiranje koje iziskuje opsežnu dokumentaciju za svaku fazu razvoja projekta. Radi se o linearnom procesu u kojem svaka faza započinje tek nakon što je prethodna u potpunosti završena. Ovo je predvidljiva metodologija koja zahtjeva strogo planiranje proizvoda, a najveća mana je što svaka promjena u nekoj od prethodnih fazi zahtjeva neizbježne izmjene u svim koracima koji su uslijedili nakon nje [3]. Model vodopada je, u konačnici, dobar odabir kada se radi o projektima malih razmjera u kojima su zahtjevi jasni, a detaljno planiranje može biti lako izvedeno.



Slika 1.1 Model Vodopada [3]

- Inkrementalni model (eng. *Incremental model*)

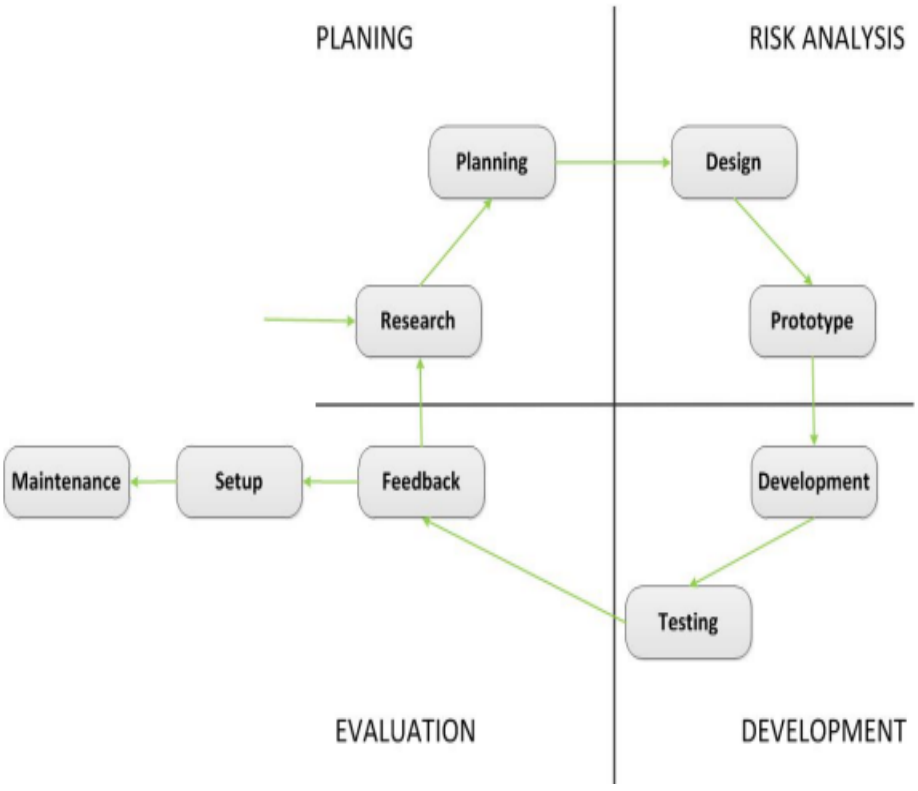
Model temeljen na inicijalnoj specifikaciji po kojoj se projekt razvija. Kako ovaj model ne zahtjeva podrobnu dokumentaciju i detaljno početno planiranje, razvoj se dijeli na više iteracija dok se ne realizira konačna verzija proizvoda. Svaka se razvojna faza izrađuje kao i u vodopadnom modelu, ali do faze testiranja. Nakon testiranja modela i primitka povratne informacije, model se ponovo prilagođava i započinje sa slijedećom iteracijom. Ovakav se proces ponavlja dok model ne postane potpuno funkcionalna aplikacija koja odgovara svim zahtjevima kupca [3].



Slika 1.2 Iterativni Model [3]

- Spiralni model (eng. *Spiral model*)

  Ovaj model predstavlja pristup u kojem se fokus postavlja na definiranje ciljeva izradom prototipa i procjene rizika. Kasnije, ono što je uočeno kao loša implementacija iz prethodne iteracije, modificirati će se u sljedećoj. Izvođenjem svakog koraka u petlji, dolazi do povećanja broja detalja i funkcionalnosti. Ovaj je model koristan kada područje primjene web aplikacije nije jasno ili strogo definirano, već su očekivane izmjene i naknadne optimizacije. Nedostatak pristupa po spiralnom modelu je činjenica da je cijela faza oblikovanja uvjetovana procjenom rizika (uočavanje i ispravljanje tehničkih, financijskih, operacijskih i ekonomskih mogućih ranjivosti) stoga je izvedba projekta po uzoru na ovaj model često skuplja od ostalih.

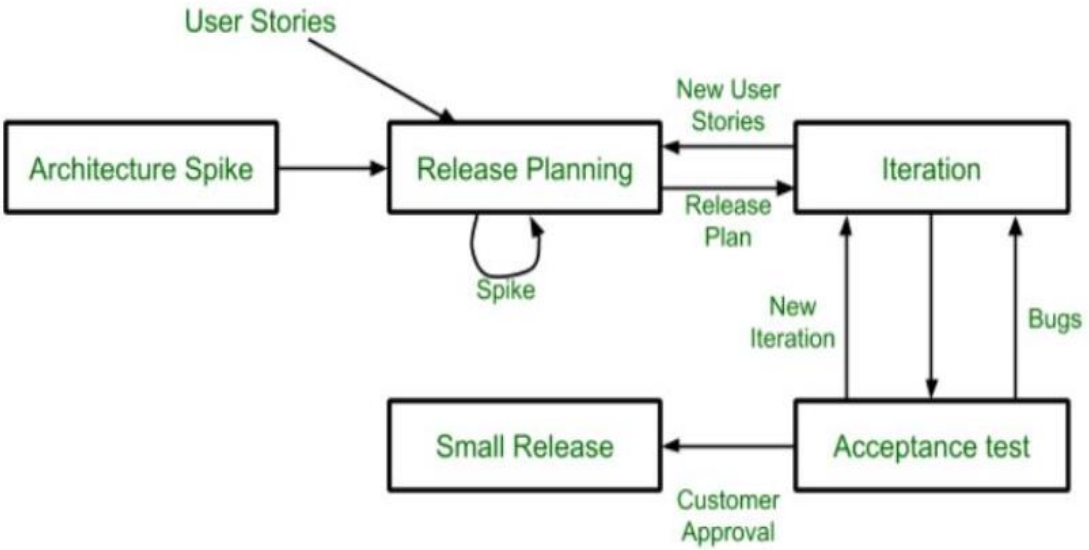


Slika 1.3 Spiralni Model [3]

- Model Ekstremnog programiranja (eng. *Extreme Programming model*)

Predstavlja način izrade prema kojem se razvoj odvija programiranjem u paru. Takav način dovodi do brze i uzastopne izrade novih verzija proizvoda.

Vlasnik projekta cijelo je vrijeme uključen u aktivnosti utječući time na brzinu izrade kao i donošenje odluka na licu mjesta. Vlasništvo koda ostaje kolektivno, a integracija se odvija kontinuirano. Nove verzije aplikacije su redovite kao i povratne informacije od kupca. Glavne mane ovakvog modela su nedostatak dokumentacije, ali i zahtijevanje redovitih sastanaka cijelog tima.

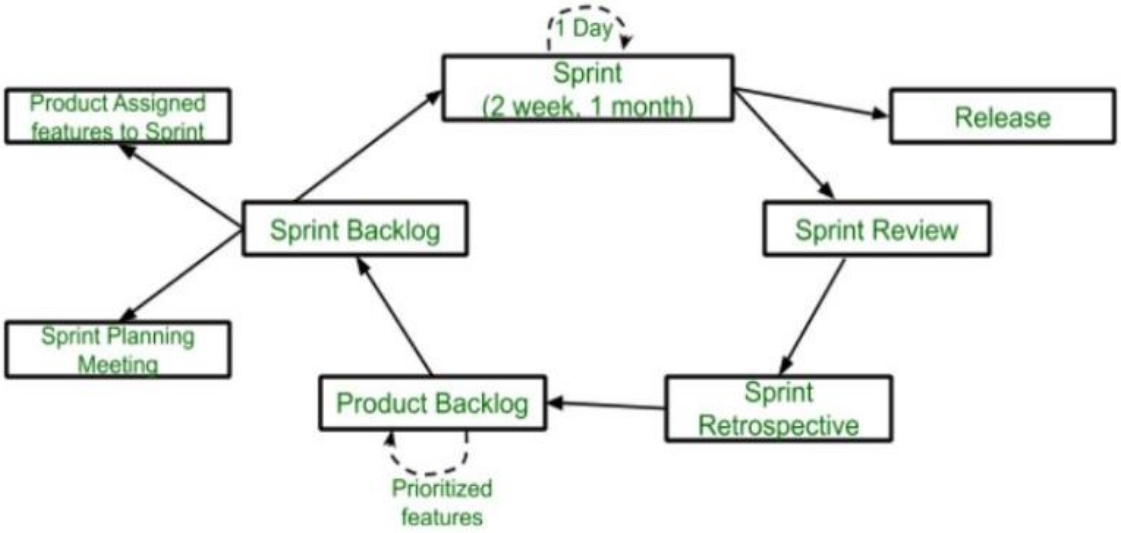


Slika 1.4 Model Ekstremnog programiranja [17]

- Agilni model (eng. *Agile model*)

Model koji je nastao s ciljem povećanja produktivnosti u razvoju softvera, ali se do danas proširio i na druge grane proizvodnje (marketing, projektiranje, itd.). Ovakav model alternativa je tradicionalnim sekvencijalnim pristupima, kao što su model vodopada ili iterativni model, jer ne postoje stroge faze razvoja već se napredak vrednuje nakon proteka unaprijed definiranog vremena. Najčešće se radi o vremenskom rasponu od dva ili četiri tjedna ovisno o složenosti faze.

Kao i u ekstremnom programiranju, i ovdje je vlasnik proizvoda iznimno uključen u razvoj, tako što sam definira važnost određene funkcionalnosti, a očekivane verzije rezultata mjere se uglavnom u tjednima (zadaci određenog sprinta moraju biti izvršeni u roku od dva tjedna) [4]. Agilan pristup promovira održiv način razvoja i omogućava brzu prilagodbu izmjenama. *Scrum* je primjer metodologije koja zagovara agilni pristup. Ova metodologija danas je najučestalija primjena koncepta programskog inženjerstva u razvoju timskih projekata i zahtjeva da svi članovi budu jednako uključeni u prijedlozima mogućih rješenja, te u dnevnim sastancima kao i retrospektivnim pregledima o uspješnosti urađenih poslova.



Slika 1.5 Agilni (*Scrum*) Model [17]

Opisane metodologije samo su neki od primjera pristupa razvoja programa u programskom inženjerstvu. Njihova glavna podjela odvija se na razinama „teške“ i „lake“ kategorije (eng. *Heavyweight and Lightweight methodologies*). Teške metodologije prigodne su za projekte u kojima se zahtjevi vrlo vjerojatno neće mijenjati, a složenost softvera dopušta

detaljno planiranje. S druge strane, metodologije lake kategorije koriste se za izradu aplikacija za koje su planirane naknadne izmjene i dodavanje funkcionalnosti koje su danas česte, posebno kod razvoja web aplikacija. To proizlazi iz konstantne potrebe za nadogradnjom, mijenjanjem i optimizacijom aplikacije koja se razvija [3].

Kako bi u konačnici odabrali ispravan pristup za razvoj web aplikacije, potrebno je uzeti u obzir više utjecajnih faktora među kojima su primarno prepoznavanje zahtjeva projekta, procjena obujma i složenosti, fleksibilnost i prilagodljivost projekta, uključenost kupca ili vlasnika, mogućnost upravljanja rizicima, te iskustvo i znanje tima programera koji će izraditi projekt. Ipak, ni jedna od metodologija nije nužno isključivo primjenjiva na konkretan projekt, stoga je sasvim moguće razviti hibridan pristup prateći više od jedne metode istovremeno prilikom izrade aplikacije [4].

1.2. Faze razvoja web aplikacija

Životni ciklus web aplikacije podijeljen je u više standardiziranih razvojnih faza: istraživanje i prikupljanje zahtjeva kupca, planiranje (odabir metodologije, kreiranje skupova zadataka), oblikovanje i dizajn, implementaciju programa, testiranje i vrednovanje, te na posljetku – održavanje. Ovisno o odabranom modelu po kojem će se izraditi aplikacija, navedene faze mogu biti drugačije raspoređene ili nazivane. Slijedi opis pojedinačnih faza definiranih u vodopadnom modelu.

1. Analiza i prikupljanje zahtjeva

Iako se u ovoj fazi ne programira i ne isporučuje nikakav konkretan kod, ona ima vrlo važnu ulogu u smanjenju troška i rizika u sveukupnom rezultatu razvoja aplikacije. U slučaju da se ova faza ne izvrši kvalitetno može doći do krivog shvaćanja zahtjeva koje aplikacija mora ispunjavati kao i do nedostatka dobrog menadžmenta koji definira pristup ostalim fazama ciklusa razvoja. Prva faza razvoja usmjerena je na smanjenje upravo tih rizika kreirajući strateške planove po kojima će se ciljevi ostvariti [5]. Rezultati početne analize i prikupljeni zahtjevi dalje će utjecati na sam razvoj aplikacije.

Za početak, potrebno je prikupiti informacije o očekivanim funkcionalnostima aplikacije. Ovaj korak podrazumijeva analizu mogućnosti izvedbe traženih funkcionalnosti kao i definiranje osnovnog tijeka rada i budućeg korištenja aplikacije. Navedeno je moguće

postići kroz komunikaciju s kupcem i ciljanim korisnicima koji zapravo i definiraju svrhu aplikacije kao i koje su funkcionalnosti potrebne za svaku komponentu pojedinačno.

Uz razne grafičke prikaze, kao što su dijagrami odnosa subjekata i dijagrami toka podataka, osoba zadužena za analizu aplikacije razvija spomenutu strategiju, dijeleći stavke aplikacije na manje module, odnosno konkretne komponente koje će se kasnije razvijati kao zasebni programi.

Analitičar je također dužan prikupiti zahtjeve vezane za dizajn aplikacije kako bi se izgradio učinkovit grafički prikaz proizvoda i odredilo koliko je različitih prikaza i stranica potrebno za učinkovito korištenje. Cilj je što kvalitetnije osmisliti i razraditi korištenje kako bi se informacije prikazivale na korisniku intuitivnim mjestima [6].

Posljednji korak ove faze je dokumentiranje zahtjeva u formalnom i provjerljivom obliku pomoću kojeg se može započeti s drugim korakom – odabirom adekvatne metodologije i razvojnog okruženja.

2. Planiranje razvoja

Jednom kada su svi zahtjevi prikupljeni i pohranjeni u formalnoj dokumentaciji, razrađuju se i imenuju funkcionalnosti na razini komponenti koje aplikacija sadrži. Ova se faza u praksi često preklapa s prvom obzirom da je u njoj potrebno potvrditi i pregledati osmišljene ciljeve. Ovdje se definira baza podataka – relacijska ili ne relacijska, njeni entiteti i atributi, ali i tehnologija, jezik i razvojno okruženje.

U ovom je trenutku potrebno osmisliti daljnje procese. Današnje web aplikacije zahtijevaju visoku razinu interakcije, moderan dizajn i brojne učinkovite funkcije pri korištenju. Vrlo je važno biti u toku s trenutno potvrđenim i provjerenim alatima i tehnologijama kako bi izvedba bila ostvarena na najvišem mogućem nivou.

Ono što je također važno naglasiti je planiranje dizajna. Obzirom da kupac i/ili budući korisnici uglavnom nisu programeri, učinkovitost web aplikacije prvotno se ocjenjuje kroz dizajn i vizualnu strukturu sadržaja. U fazi planiranja potrebno se osvrnuti na prethodni korak analize zahtjeva i u najvišoj mogućoj mjeri ciljati na ostvarivanje ispravnog i potpunog korisničkog iskustva.

Sveukupno planiranje uvelike ovisi o timu ili pojedincu predviđenima za programiranje aplikacije. Stoga, unatoč zahtjevima, važno je imati na umu skup i područje vještina razvojnih programera kao i njihovu spremnost za suradnju.

3. Oblikovanje i dizajn

Dizajn aplikacije dijeli se na dva glavna zadatka: informacijski i grafički dizajn [5].

Informacijski dizajn predstavlja podjelu i strukturu dokumenata sadržanih u aplikaciji, a odnosi se na raspodjelu, sadržaj i mogućnost pristupa. Ovdje je još uvijek vrlo važna aktivna interakcija s klijentom kako bi pojedinačne komponente potvrdile svoj status u cjelokupnoj svrsi aplikacije. Mogući problemi proizlaze iz korisničkog nerazumijevanja mogućnosti implementacije određenih funkcionalnosti, stoga je važno da ih adekvatna osoba uputi u ograničenja i realnu izvedbu.

Grafički dizajn predstavlja izgled zaslona, boje, animacije i interakciju u direktnom korištenju aplikacije. Danas postoje brojni alati koji simuliraju scenarije na grafičkom nivou korištenja web aplikacije. Među njima, danas je vrlo popularna Figma, alat koji omogućava izradu realnih sučelja prilikom poziva svake funkcionalnosti unutar aplikacije. Po uzoru na ovako izrađene grafike, klijent može procijeniti koliko je zadovoljan osnovnom idejom ili predložiti određene izmjene, dok je zauzvrat programeru proslijeđen grafički predložak po kojoj će planirati kod, komponente i fizičku strukturu programa kojeg razvija.

Rezultat ovog koraka je detaljan projektni dokument koji opisuje strukturu web aplikacije, strukturu baze podataka i sve funkcionalnosti [5].

Uzimajući u obzir cjelokupan razvoj, rano testiranje financijski je i vremenski isplativije od testiranja kada se aplikacija smatra dovršenom. To uključuje testiranje dizajna web aplikacije u odnosu na prethodno postavljene ciljeve kako bi se utvrdilo može li aplikacija ispravno isporučiti očekivane rezultate.

4. Implementacija programa

Jedna od važnijih razlika *desktop* aplikacije i web aplikacije je u tome što korištenje web aplikacije ne zahtjeva nikakvu instalaciju ili preuzimanje aplikacije i njenih datoteka već je dovoljno imati pristup internetu i web pregledniku. Upravo je zbog te razlike razvoj web aplikacija otvoreniji agilnim pristupima čime se unaprjeđenja i promjene mogu brzo i lako ostvarivati [1].

U fazi implementacije programa započinje konačna realizacija dogovorenih ciljeva. U formiranim timovima, napredak se cijelo vrijeme nadzire, čime se evidentira vrijeme i uspješnost izvršavanja zadanih manjih dijelova web aplikacije. Rezultat implementacije su

dva dokumenta. Prvi dokument sadrži detaljne upute korištenja web aplikacije koja se razvija, dok drugi predstavlja potpun izvorni kod kojim je implementirana.

Jednom kada su podaci pohranjeni na serveru, glavne obveze razvojnih programera svode se na sigurnosno kopiranje sadržaja i dijeljenje istog. Nakon ovog koraka, slijedi sama validacija, a potom preostaje dovršavanje aplikacije u produkciji.

5. Testiranje i vrednovanje

Svaki bi program trebao biti temeljito testiran i provjeren prije faze isporuke kako bi se provjerio rad funkcionalnosti te osiguralo da sve poveznice i upiti rade. Postoje razne metode testiranja, a u većim firmama ovaj posao obavlja cijeli tim za provjeru svih scenarija korištenja prije nego aplikacija postane dostupna.

Test korisnikova prihvaćanja, beta testiranje, funkcionalno testiranje, testiranje po metodama bijele i crne kutije samo su neki od primjera iz današnje prakse ispitivanja kvalitete. Što su problemi ili pogreške ranije uočeni to je iste lakše popraviti, stoga se testiranje ne odvija samo u ovoj fazi već i kroz prethodne.

Jednom kada je testiranje aplikacije uspješno provedeno s pozitivnim odgovorom testera, aplikacija je spremna za isporuku iako se samo testiranje mora nastaviti i u fazi održavanja.

Nakon ove faze moguće je provesti analizu ukupnih troškova, ranjivih točaka i potrebnih poboljšanja.

6. Isporuka i održavanje aplikacije

Današnje aplikacije razvijaju se po principu kontinuirane integracije, ciljajući na lakše implementiranje mogućih adaptacija kao i preinaka prethodno strukturiranih komponenti aplikacije. Ova faza razvoja zapravo nikad ni ne završava obzirom da se održavanje proteže dokle god je i aplikacija postojana i dostupna.

S jedne strane, ovo se odnosi na redizajniranje sučelja u skladu s modernim standardima s vremena na vrijeme, ali i na redovito ažuriranje i provjeru alata koji su korišteni u izradi aplikacije.

U ovoj fazi najvažnija je potvrda da aplikacija izvršava funkcionalnosti u zadanim okvirima. Da bi se to uspostavilo, potrebno je osigurati pravilno funkcioniranje vatrozida, pošte, slanje i odgovore HTTP zahtjeva te SQL upita.

Bitno je naglasiti da se spomenuto dodavanje novih značajki i funkcionalnosti odvija upravo u ovoj fazi razvoja.

1.3. Sigurnost web aplikacija

Kao što je opisano, prilikom izrade bilo kakve web stranice, brojni su faktori koje je potrebno razmotriti kako bi ista u konačnici bila stabilna i upotrebljiva u stvarnom vremenu. Danas zaista možemo reći da je zaštita web stranica obavezna mjera opreza za programere u svim aspektima dizajna i budućeg korištenja. Stoga, između odabira okruženja i tehnologija za razvoj, kao i poslužitelja odnosno domene, važno je obratiti pažnju na uobičajene sigurnosne ranjivosti stranice.

Kako bi posljednja faza razvoja web aplikacije bila uspješna, potrebno je poznavati osnovne koncepte sigurnosti i najčešće metode narušavanja iste.

Pojam sigurnosne ranjivosti stranice predstavlja puteve kojima je moguće izvršiti sigurnosne prijetnje webu, odnosno internetski rizici računalne sigurnosti koji mogu oštetiti uređaje, sustave i mreže, izložiti korisnike mrežnoj i podatkovnoj šteti i uzrokovati neželjene radnje ili događaje.

Internet je mjesto prepuno raznog sadržaja stoga su i brojni mogući napadi kao i slučajna rušenja web stranica. Učestalo svjedočimo stranicama koje su odjednom nedostupne ili prikazuju štetan sadržaj za koji nisu prvotno namijenjene. Važno je spomenuti i eksponencijalan rast broja web aplikacija i stranica koje omogućavaju online plaćanja i autentifikacije prilikom čega zahtijevaju unos brojeva kreditnih kartica, pinova, mail adresa i šifri. Hakiranje ovakvih stranica ugrožava identitet i financijsku sigurnost pojedinaca čiji su podaci javno otkriveni i dostupni.

Slijedeća lista sigurnosnih prijetnji na web-u predstavlja najčešće načine kojima se omogućava neovlašteno pristupanje, izmjenjivanje, uništenje ili ometanje web stranica i aplikacija.

- SQL injekcije

SQL injekcije predstavljaju jednu od ranjivosti web sigurnosti koja omogućava napadačima pristup skrivenim podacima pohranjenim u bazi. Ovom metodom ometaju se

ciljani upiti i rad s podacima, a napadač može ubacivati zlonamjerne naredbe u svrhu narušavanja stabilnosti i funkcionalnosti stranice.

Uspješan napad na bazu podataka može lažirati korisničke račune stvoriti nove ili obrisati postojeće. Također, na ovaj je način moguće novom identitetu dodijeliti administratorske ovlasti pa tako prisvojiti sva prava i u konačnici bazu učiniti neupotrebljivom.

Važno je napomenuti da, iako su poruke poslužitelja baze vrlo korisne prilikom razvojne faze aplikacije, jednom kada aplikacija postane javna i aktivna ove je poruke potrebno onemogućiti ili bilježiti u datoteku s ograničenim pristupom [11].

- Napad skriptama (*Cross-Site Scripting* – XSS)

XSS je klasa napada na aplikacije koji se odvijaju pomoću umetanja skripti s klijentske strane u preglednik. Na ovaj način napadač može zahtijevati autorizacijski kolačić konkretnog korisnika stranice te tako preuzeti sve ovlasti koje korisnik ima. Ovakvom metodom moguće je pristupiti tajnim podacima s kreditnih kartica ili detaljima o korisniku kao i promijeniti šifre i korisničko ime.

Podjela XSS tipa napada temelji se na načinu prema kojem stranica vraća ubačene skripte u preglednik – reflektirana i postojana ranjivost stranice.

Reflektirana ranjivost stranice javlja se kada se korisnički podaci proslijeđeni na poslužitelj nepromijenjeni odmah prikazuju na web pregledniku. Ovo znači da će se i sve skripte iz izvornog korisničkog sadržaja izvršiti prilikom učitavanja nove stranice. Ako su primjerice nekakvi podaci za pretragu na stranici kodirani kao parametri URL-a, a isti se javljaju i u rezultatima pretrage, napadač lako može konstruirati link koji šalje drugom korisniku a sadrži zlonamjernu skriptu. Kao što je prethodno navedeno, i ova će se skripta izvršiti čim se stranica negdje učita i tako dovesti do gubitka sigurnosti na stranici.

Druga spomenuta ranjivost je postojana, a odnosi se na slučaj gdje je zlonamjerna skripta unaprijed ugrađena u web stranicu, te se izvršava kada je stranica učitana i korisnici je zatim nesvjesno pokreću. U ovakvim slučajevima napadač čak ne mora ni biti u direktnom kontaktu s korisnicima čije podatke preuzima, stoga je ugrađivanje zlonamjernih skripti vrlo popularan i moćan način ugrožavanja sigurnosti web stranica.

U oba slučaja, napadi su omogućeni preko izvršavanja GET i POST zahtjeva. Između ostalog tu su još i podaci o kolačićima prikazani na stranici, a i datoteke koje je korisnik preuzeo ili učitao.

Danas mnogi zastupljeni web okviri (uključujući Angular, ReactJS i Vue) imaju automatiziranu zaštitu od ovakve vrste napada i narušavanja rada stranice. Takvi okviri prepoznaju i modificiraju korisničke podatke tako da se ti podaci ne mogu koristiti za pokretanje skripte te nije moguće zlonamjerno utjecati na izvršavanje koda na poslužitelju.

Ipak, dobro je znati kako preventivno djelovati pri razvoju aplikacije – najsigurnije je isključiti/ukloniti HTML elemente koji podržavaju ubacivanje skripti (npr. <script>, <object>, <embed> i <link>) [11].

- Krivotvorenje zahtjeva (*Cross-Site Request Forgery* – CSRF)

Kao i XSS napadi, i ovaj tip narušavanja sigurnosti web stranice je *cross-site* jer koristi različite web stranice ili elemente za slanje zahtjeva unutar aplikacije. CSRF šalje HTTP zahtjev svaki put kada korisnik otvori web lokaciju koja sadržava zlonamjeran kod, nakon čega se isti izvršava bez daljnje radnje korisnika i na razne načine ugrožava sigurnost stranice i korisnika.

Najčešće se ovakvi napadi događaju kroz neželjenu email poštu. Poruke sadrže zlonamjerne linkove na stranice dok je korisnik autentificiran i na taj način daje napadaču pristup pregledniku, a samim time daljnji zahtjevi aplikacije smatraju se legitimnima jer se šalju zajedno s kolačićem žrtvine sesije.

Kada se odvija CSRF narušavanje sigurnosti, napadač uvijek ima konkretni cilj koji želi izvršiti. Da bi mu to uspjelo, aplikacija koju narušava mora koristiti sesijske kolačiće za provjeru autentičnosti zahtjeva korisnika. Također, napadač mora poznavati parametre zahtjeva kojeg želi izvršiti.

Navedeno se najčešće realizira uz pomoć GET i POST zahtjeva. GET zahtjevi se u aplikaciji koriste kao naredbe pomoću kojih se događa promjena stanja (primjerice promjena lozinke). Skripta se tada pokreće kada korisnik klikne na poveznicu, moguće i na sliku. U oba slučaja, GET zahtjev će izvršiti skriptu bez da to korisnik zna i na taj način uzrokovati narušavanje sigurnosti.

POST zahtjevi se najčešće koriste za promjene stanja u aplikaciji te ih napadač također može koristiti kako bi poslao željene vrijednosti izravno na poslužitelj. Jednostavan primjer izvođenja navedenog je preko, recimo <form> HTML elementa gdje napadač može navesti korisnika da klikne na *Submit* čime će se u ovom slučaju neželjena skripta izvršiti automatski.

Slično je izvedivo i s PUT ili DELETE zahtjevima, premda danas većina preglednika ima ugrađenu automatsku provjeru prilikom izvođenja istih zbog restrikcija zadanih u pravilima istog podrijetla (*same-origin policy*) kao i pravilima međusobnog dijeljenja resursa (*cross-origin resource sharing* – CORS).

Učinak CSRF napada ovisi o razni prava koje ciljani korisnik ima unutar aplikacije. Za prosječnog korisnika, uspješan CSRF napad uobičajeno izvrši promjene u vidu lozinke i e-mail adrese ili prenese podatke na drugi račun. Ako su u pitanju administratorske ovlasti, ovakav napad predstavlja prijetnju potpunog rušenja ili prenamjene aplikacije. Kako bi se CSRF napadi spriječili potrebno je osigurati aplikaciju na način da napadač ne može slati proizvoljne zahtjeve u ime korisnika koji je autentificiran.

Najčešća tehnika za postizanje zaštite u ovom slučaju je korištenje CSRF tokena. Takvi su tokeni jedinstveni i stvaraju se po sesiji, a generira ih aplikacija koja ih dodjeljuje klijentu. Oni se zatim šalju natrag zajedno sa zahtjevom kojeg poslužitelj potom provjerava. Na ovaj način svaki zahtjev koji ne dolazi iz istog porijekla u sesiji, jednostavno se odbaci čime uspješno štiti aplikaciju od CSRF napada.

Slanje dvostrukog kolačića kao i sprječavanje više kolačića na istom mjestu su također moguće mjere zaštite od ovakve vrste napada. Ono što se još pokazalo vrlo učinkovitim za API krajnje putanje je usporedba prilagođenih zaglavlja. Pomoću ovih zaglavlja sprječava se slanje zahtjeva među različitim domenama što uspješno eliminira opasnost od CSRF napada [11].

- Uskraćivanje usluge (*Distributed Denial of Service* – DDoS)

Još jedna česta metoda mogućeg napada na stranicu je DDoS – distribuirano uskraćivanje usluge, gdje više kompromitiranih sustava „napada“ ciljanu aplikaciju ili stranicu. Rade to tako da svojim zahtjevima preplave resurse poslužitelja i blokiraju legitimne korisnike.

U tipičnom DDoS napadu, napadač identificira ranjivu web aplikaciju kojoj može slati neumjerenu količinu naizgled uobičajenih zahtjeva nakon čega aplikacija postaje preopterećena prometom te uskraćuje uslugu.

Ovakav napad uzrokuje preopterećenje napadnute aplikacije koja posljedično ne može odgovoriti ili reagira vrlo sporo na standardni promet. Vlasnici zaraženih računala obično ne znaju da su im računala ugrožena, a pritom trpe i gubitak same usluge [11].

Obrane od DDoS napada funkcioniraju tako da blokiraju pretjeran promet, dok uobičajeni propuštaju. Međutim, obrana nije dio same aplikacije već se većinom nalazi u web poslužitelju.

1.3.1. Konkretni koraci za zaštitu web aplikacija

Prije svega, važno je napomenuti da su opisani napadi mogući jedino ako aplikacija nema dobro implementiranu zaštitu. Najčešći uzrok su podaci koji dolaze iz preglednika. Stoga, prije ikakvog korištenja istih, bitno je provesti provjeru kako bi u prikaze, SQL upite ili pozive stigli isključivo potpuno ispravni podaci – oni koji su aplikaciji potrebni za izvršavanje funkcionalnosti. Navedeno se najviše odnosi na URL parametre, GET i POST zahtjeve, HTTP zaglavlja, kolačiće i datoteke koje korisnik može prenijeti.

Prvi, može se reći obavezan, korak za zaštitu je konfiguracija poslužitelja da koristi HTTPS komunikacijski protokol koji šifrira podatke poslane od klijenta. Na ovaj se način osigurava da podaci za prijavu, kolačići, podaci iz POST zahtjeva i informacije zaglavlja nisu lako dostupni napadaču.

Ovisno o tipu aplikacije ili stranice, važno je koristiti učinkovito upravljanje lozinkama poticanjem upotrebe jakih lozinki. Učestala praksa je i dvofaktorska autentifikacija koja uz lozinku, korisnika za prijavu traži da unese još jedan podatak u obliku koda poslanog na hardver koji korisnik posjeduje (primjerice SMS poruka na mobitelu).

Minimalan prikaz podataka – samo onoliko koliko je dovoljno legitimnom korisniku da prepozna ispravnost. Klasičan primjer je unos osjetljivih podataka kao što su brojevi s kreditne kartice. Korisnik ne mora nužno vidjeti više od posljednja četiri broja s kartice koja je sačuvana da bi prepoznao o kojoj se radi. Stoga, zaštita osobnih podataka može biti izvedena tako da sustav prikazuje samo ograničen broj znakova kako netko tko nije ovlašten ne bi mogao kopirati potpunu informaciju i dalje ju koristiti.

Ipak, danas brojni uobičajeni web okviri pomažu ublažavanju raznih ranjivosti pa je jedan od koraka koje razvojni programer mora obaviti prikupljanje znanja o ugrađenim alatima za zaštitu prije samog odabira razvojnog okruženja.

2. Okruženje i tehnologije za razvoj

Nove generacije web aplikacija postaju sve složenije u vidu funkcionalnosti koje mogu izvršavati, a u skladu s tim zahtjevima razvijaju se i nadograđuju okruženja u kojima se iste razvijaju.

Odabir tehnologije i okruženja za razvoj danas se može podijeliti na dva osnovna pravca – izrada prilagođenih aplikacija „od početka“ kodirajući sve sastavne komponente ili izrada uz pomoć sustava za upravljanje sadržajem (CMS). Oba pristupa imaju svoje prednosti i mane, a u konačnici će prevagnuti onaj koji više zadovoljava potrebe aplikacije, financijske mogućnosti klijenta, te tehnološko poznavanje alata od strane izvođača.

Prednosti CMS-a pristupa najčešće su vezane uz vrijeme potrebno za realizaciju obzirom da ti alati nude unaprijed definirane komponente, vizualne blokove, stalna ažuriranja, te u većini slučajeva ne zahtijevaju nikakvo poznavanje programiranja. S druge strane, korištenje CMS alata za razvoj često košta mnogo više od prilagođenog programiranja zbog cijena potrebnih dodataka. Isto tako, dizajn u ovakvim okruženjima je poprilično ograničen i standardiziran, stoga je teško postići autentičnost u korisničkom iskustvu, a bilokakve prilagođene komponente je puno teže realizirati. Zbog popularnosti i količine unaprijed definiranih dijelova arhitekture aplikacije izrađene u CMS alatu, ovakvi su sadržaji podložniji sigurnosnim rizicima i gubicima.

Prilagođeno izrađene aplikacije u početku mogu biti skuplje zbog potrebe angažiranja programera koji će je realizirati, ali zahtijevaju manje naplaćivanih paketa, dodataka i tema u odnosu na CMS alate. Ovaj pristup je adekvatniji ako klijent traži jedinstven dizajn ili planira stvaranje vlastitog branda. Za razliku od CMS izgrađenih aplikacija gdje pristup uređivanju sadržaja ima veća skupina ljudi, prilagođeno programirane aplikacije ne omogućavaju pristup kodu drugima, stoga posao uvelike ovisi o programerima koji su je realizirale. Također, ovaj pristup najčešće zahtjeva više vremena za isporuku gotovog proizvoda.

Ovo poglavlje rada opisuje danas najkorištenije tehnologije i okruženja za razvoj web aplikacija analizirajući mogućnosti navedenih pravaca.

2.1. Sustavi za upravljanje sadržajem

Ako se izradi web stranice/aplikacije pristupi načinom koji ne zahtijeva pisanje koda, potrebno je odabrati adekvatan sustav za upravljanje sadržajem – CMS. CMS je softverski alat koji omogućava kreiranje, upravljanje, mijenjanje i održavanje web stranica pomoću sučelja koje je jednostavno za korištenje, a dodatne izmjene na stranici realiziraju se instalacijom i/ili kupovinom tema i dodataka [18]. Ovakvi alati odvajaju sadržaj od koda koji je unaprijed definiran, stoga zahtijevaju samo upravljanje sadržajem kako bi se izgradila stranica. Kako su podržani u online okruženju, CMS sustavi danas omogućuju da više korisnika paralelno mijenja i upravlja informacijama i sadržajem na stranici što može uvelike olakšati i ubrzati izradu u stvarnom vremenu.

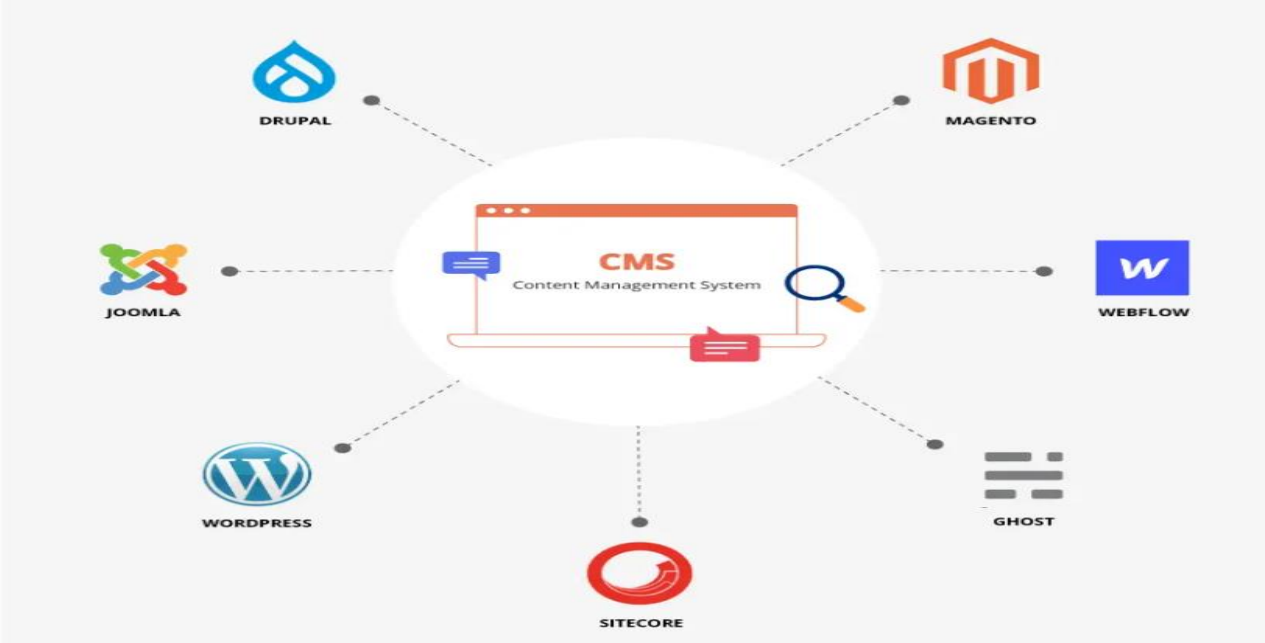
CMS alati omogućuju prikaz i upravljanje različitih sadržaja stoga se podjela CMS-a definira na razini tipa sadržaja za koji se koriste. Među najčešće korištenim podgrupama su: ECM – nadopunjeni sustav za upravljanje sadržajem koji omogućava korištenje i upravljanje dokumentima u vidu bilanci, računa i slično, WCMS – sustav za upravljanje web sadržajem poput web aplikacija, LCMS – omogućava upravljanje sadržajem za učenje [19].

Sustavi za upravljanje sadržajem imaju brojne mogućnosti, a sve osiguravaju brže i efikasnije rukovanje sadržajem u odnosu na prilagođeno programiranje web stranica.

Slika 2.1 Mogućnosti CSM alata [20]

Prilikom odabira određenog CMS-a za izradu web stranice, važno je razmotriti mogućnosti i ograničenja dostupnih alata, jednostavnost korištenja, ali i potrebna predznanja. Danas je velik broj web stranica predstavljen pomoću nekog CMS alata, što podrazumijeva stalne optimizacije istih kako bi dodavanje, uređivanje, objavljivanje i održavanje vremenom bilo naprednije i u skladu s modernim standardima prisutnosti sadržaja na Internetu.

Među najpoznatijim i najkorištenijim takvim sustavima danas nalaze se: WordPress, Joomla, Drupal i sl.



Slika 2.2 Logo oznake popularnih CMS alata [18]

2.1.1. WordPress

Nastao 2003., WordPress je prvotno predstavljen kao CMS alat za jednostavnu izradu i uređivanje blogova. Kako je rasprostranjenost CMS alata rasla, tako je i WordPress evoluirao u alat koji omogućava funkcionalnosti za kreiranje web stranica različitih sadržaja.

Danas je WordPress najkorišteniji CMS alat pomoću kojeg je moguće zadovoljiti brojne uobičajene funkcionalnosti i sučelja web stranica. WordPress je osmišljen kao web okvir s modernim i intuitivnim grafičko-korisničkim sučeljem. Izrađen je u PHP programskom jeziku, te podržava MySQL za rad s bazom podataka, a pomoću ovog alata je jednostavno ostvariti strukturirani i dinamični sadržaj na Internetu. WordPress je jednostavan i

skalabilan, a sadrži velik broj postojećih predložaka i tema što ga čini popularnim za razvoj stranica različitih namjena.

Osnovni WordPress paket je besplatan i dostupan svima uz moguću nadogradnju kupovinom tema i dodataka (eng. *plugins*). Instalacijom i kreiranjem WordPress računa korisnik ostvaruje pristup sučelju koje sadrži alatnu traku s osnovnim izbornicima kao i link na stranicu koja se kreira, glavni izbornik s uobičajenim WordPress mogućnostima kao što su postavke, opći dizajn, sigurnost, WordPress SEO i slično, te radna površina na kojoj je vidljivo koji su paketi, teme i dodaci uključeni.

Opisano je sučelje karakteristika WordPress-a, te neovisno o tipu stranice ili instaliranoj temi, ono sadrži iste komponente stoga je vremenom snalaženje sve jednostavnije.

Što se korisničke podrške tiče, WordPress održava i nadograđuje velika zajednica programera zaposlenika i volontera, što ovaj CMS čini dobrim izborom ako je planirano dugoročno održavanje stranice. Iako je WordPress kao alat konstantno razvijan i nadograđivan, da bi se web stranice uspješno kreirale, potrebno je voditi računa o dodatnim instalacijama u vidu paketa s potrebnim dodacima.

Uz općenitu nadogradnju mogućnosti, paketi, dodaci i teme mogu također i poboljšati nedostatke WordPress-a. Dodaci (eng. *plugins*), poput Elementor-a, YoastSEO-a, Jetpack-a i sličnih omogućuju jačanje sigurnosti stranice, širenje mogućnosti, prilagodbe stranice pretraživačima, i razne druge optimizacije. Vizualne mogućnosti WordPress stranice predstavlja odabrana tema među kojima su najpoznatije Divi, Astra, OceanWP i Ultra. Neovisno o kojoj se temi radi, sve podržavaju WordPress dodatke. Uz blokove za oblikovanje i sučelja dobivena odabranim temama, nije potrebno pisanje koda, ali u WordPress-u je također uvijek moguće dodati prilagođeni kod u vidu CSS klasa i identiteta, kao i određene jQuery naredbe za upravljanje ponašanja komponenti.

2.1.2. Sigurnost WordPress alata

Kao najpopularniji sustav za upravljanje sadržajem danas, WordPress koristi velik broj ljudi i kompanija, čineći ga tako dostupnijim, ali i ranjivijim.

Kako se, primjerice, do log-in forme za administratora stranice dolazi lako, dodajući samo /*wp-admin* na URL adresu, vrlo je važno imati čvrstu lozinku, ali i dodatak *WPS Hide Login* [20]. Uz ovaj dodatak, za zaštitu cijele stranice i korisničkog WordPress profila,

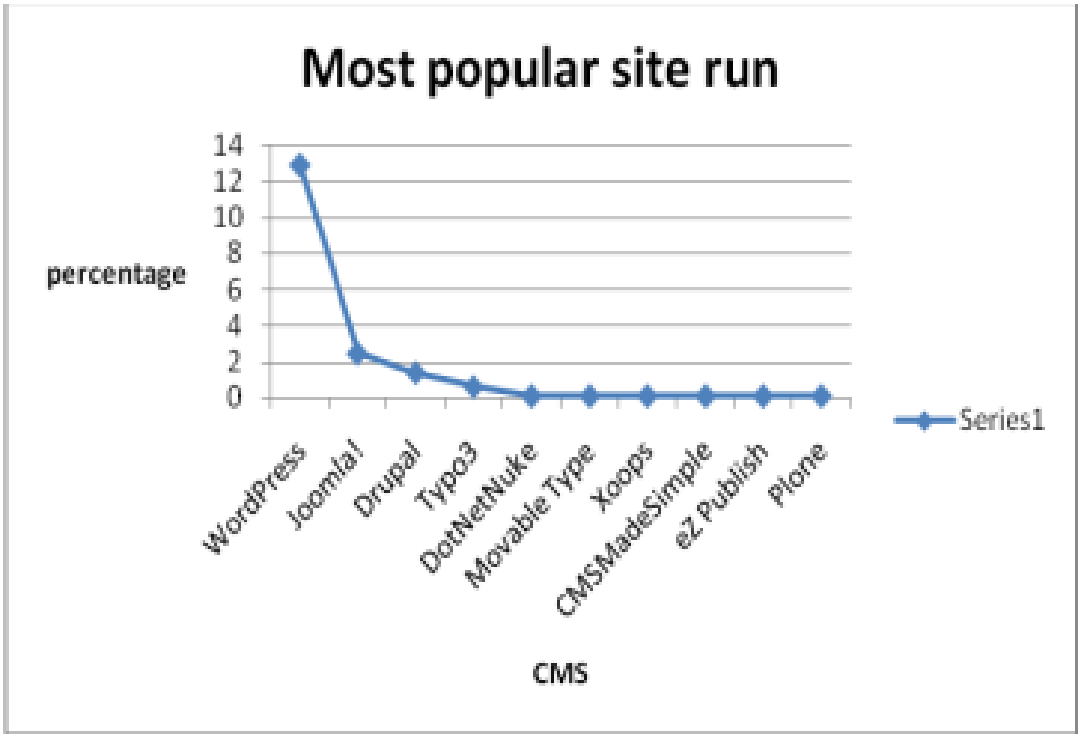
moguće je instalirati *iThemes Security, Wordfence Security, WP Security* i slične. Svi ti dodaci integrirani zajedno, održavaju vatrozid, limitiraju opasnost od narušavanja stranice ili gubljenja podataka, detektiraju viruse i sprječavaju nasumične pokušaje upisivanja URL adresa [20].

Redovitim ažuriranjem samog WordPress-a, ali i svih dodataka, povisuje se razina sigurnosti (za sve uobičajene ranjivosti) jer programeri korisničke podrške često nadograđuju i popunjavaju ranjivosti u kodu. Svaka stranica rađena u WordPress-u štiti se i kreiranjem sigurnosnih kopija tako da bez obzira na napade, stranica uvijek može biti vraćena u izvornom obliku. Većina tema u WordPress-u također ima implementiranu mogućnost dodavanjem CAPTCHA servisa u područjima kontakt forme ili poljima za prijavu. Ova značajka osigurava izbjegavanje neželjene pošte, ali i sprječava botove koji mogu učitati datoteke sa zlonamjernim softverom.

2.1.3. Usporedba CMS alata

Prototip projekta ovog rada napravljen je u WordPress-u stoga je upravo na njega stavljen naglasak u poglavlju, ali kao što je spomenuto, postoje i mnogi drugi CMS alati na tržištu.

Poznati primjeri su recimo, Joomla i Drupal od kojih svaki, baš kao i WordPress, imaju svoje prednosti i mane.



Slika 2.2 Omjer korištenja CMS alata [21]

Kako je prikazano na slici 2.2 Joomla je drugi najkorišteniji CMS alat na svijetu. Ovaj je CMS također besplatan i dostupan svima, a zahtijeva dodatan trošak u vidu instalacije dodatnih paketa i alata. Ovisno o potrebama sadržaja na stranici, u Joomla-i je, kao i u WordPress-u, moguće pisati vlastiti kod kako bi se prilagodili prikazi i mogućnosti. Laka je i intuitivna za korištenje stoga je pogodna za početnike [21].

Joomla ima snažnu korisničku podršku i mnogo ponuđenih ekstenzija za poboljšanje performansi, održivosti i dodatnih mogućnosti. Ipak, ažuriranje paketa se ne odvija ni približno redovito kao u WordPress-u, što stranice izgrađene u Joomla-i čini podložnijim sigurnosnim rizicima i nekompatibilnosti instaliranih dodataka [20].

Na trećem mjestu po zastupljenosti CMS alata danas, nalazi se Drupal koji se eksponencijalno razvija. Što se vrste sadržaja tiče, Drupal podržava kreiranje velikih i kompleksnih projekata, a pristup razvojnom kodu prilikom izrade, u odnosu na prethodna dva CMS-a, je najveći.

Drupal-ovo sučelje razlikuje se od WordPress-ovog ili Joomla-inog sučelja jer je kompleksnije, te nudi više mogućnosti upravljanja. Za korištenje ovog CMS-a ipak je potrebno poznavanje rada sustava i programiranja stoga nije prigodan za početnike ili osnovne korisnike [20].

U nastavku je prikazana usporedna tablica opisanih sustava za upravljanje sadržajem koja obuhvaća osnovne karakteristike važne prilikom odabira alata.

KARAKTERISTIKA	WORDPRESS	JOOMLA	DRUPAL
Jednostavnost korištenja	Jednostavan za početnike	Umjereno za početnike	Zahtijeva iskustvo
Korisnička podrška	Velika zajednica i resursi	Aktivna zajednica	Manja, tehnički stručna zajednica
Pristup kodu i programiranje	Upotrebljiv bez programiranja	Bolji performans programiranjem	Napredno programiranje
Broj dostupnih dodataka i tema	Preko 58000 i 7500	Preko 8000 i 1000	Oko 45000 i 2000

Tablica 2.1 Usporedba značajki CMS alata

Konačan izbor među CMS alatima za izradu web stranice trebao bi prvenstveno ovisiti o zahtjevima stranice, tehničkom znanju korisnika alata i mogućoj budućoj nadogradnji [21].

2.2. Izrada prilagođenih aplikacija

U slučaju razvoja web sadržaja programiranjem, potrebno je poznavati sintaksu programskog jezika koji će se koristiti te strukturu web aplikacije. Svaka web aplikacija sastoji se od tri komponente – korisničkog sučelja (eng. *frontend*) , servera (eng. *backend*) i baze podataka. Kako bi se optimiziralo korištenje i povećao broj mogućnosti, uz navedene komponente, aplikacija može koristiti dodatne okvire (eng. *frameworks*) i biblioteke (eng. *libraries*). Prilikom izrade prilagođenih stranica, potrebno je isprogramirati i integrirati sve navedene dijelove aplikacije.

Općenito opisano, korisničko sučelje predstavlja sve vizualne elemente aplikacije. Korisnik ostvaruje interakciju s aplikacijom upravo preko ove komponente pomoću koje se zahtjevi prosljeđuju ostatku sustava.

Uz HTML i CSS, osnovna tehnologija prilikom takvog razvoja je JavaScript jezik za skriptiranje usklađen po ECMAScript standardu. Danas je 98.7% svih web stranica i aplikacija napisano pomoću JavaScript-a [8]. JavaScript podržava različite programske paradigme uključujući vođenje događajima, proceduralno i imperativno programiranje. Podržava sučelja za programiranje aplikacije (eng. *Application Programming Interface*) – API-je, također podržava rukovanje regularnim izrazima, rad s datotekama, tekstom i standardiziranim strukturama podataka. JavaScript omogućava dinamičko ažuriranje i mijenjanje sadržaja korištenjem visoko naprednih funkcija i uključivanjem „vanjskih“ biblioteka.

Sve navedene značajke ovog programskog jezika opravdavaju njegovu popularnost i čine ga prikladnim odabirom za razvoj web aplikacija.

Među novije razvijenim, ali sve popularnijim programskim jezikom za razvoj web aplikacija nalazi se TypeScript. Ovaj se jezik doslovno može shvatiti kao nadskup JavaScript-a jer koristi istu sintaksu te je podržan u istim okvirima i platformama.

Jedina i glavna razlika ovih dvaju jezika je što su TypeScript-om uvedeni tipovi, klase i moduli samo kako bi se obogatile mogućnosti JavaScript-a [9].

Neovisno o jeziku u kojem će korisničko sučelje biti isprogramirano, kako bi realizacija uopće bila moguća, potrebno je odabrati okvir koji podržava taj jezik čime se omogućava ispravno okruženje za izradu. Web okviri uvelike olakšavaju razvoj jer sadrže različite klase, unaprijed definirane dijelove koda, funkcije i razne moguće pomoćne module za izvršavanje uobičajenih procesa kako bi se pojednostavila izrada. Među najčešćim okvirima za korisničko sučelje danas su ReactJS, Angular i Vue.

Nakon poslanog zahtjeva s korisničke strane, server obrađuje dobivene podatke, a potom komunicira s bazom podataka nakon čega vraća rezultat sučelju. Za razvoj servera danas najčešće se koriste programski jezici poput PHP-a, Java-e ili Python-a.

Baza podataka, stoga, služi za pohranu, upravljanje i dohvaćanje podataka potrebnih za rad aplikacije. One mogu biti relacijske ili ne-relacijske, a odabir ovisi o kompleksnosti i stvarnoj ovisnosti podataka koje aplikacija prikazuje i zahtjeva. Najčešće relacijske baze pišu se u MySQL-u ili Postgresql-u, a što se ne-relacijskih tiče, među najpopularnijima danas je MongoDB.

I baza podataka i server pripadaju *backend* dijelu svake aplikacije, a prilikom odabira okvira i biblioteka za razvoj ovih komponenti, potrebno je pronaći kompatibilnu kombinaciju kojom će integracija s korisničkim sučeljem biti omogućena.

2.2.1. Okviri za izradu prilagođenih aplikacija

Okviri se općenito dijele na dvije kategorije – klijent strana (često nazivana *frontend*) i server stranu (u ovom kontekstu, *backend*). *Backend* služi za implementaciju CRUD operacija, sigurnosnih provjera vezanih za same podatke i drugih značajki vezanih za upravljanje podacima koje aplikacija sadrži ili prima prilikom korištenja, dok je *frontend* strana zadužena za upravljanje procesima generiranja i oblikovanja sadržaja na sučelju, te početno procesuiranje ulaznih i izlaznih podataka [10].

Kako je već spomenuto, brojni okviri podržavaju JavaScript kao najpopularniji jezik u web razvoju. Ipak, Node.js je dosad već standardiziran alat koji se koristi ako se razvija JavaScript aplikacija [7].

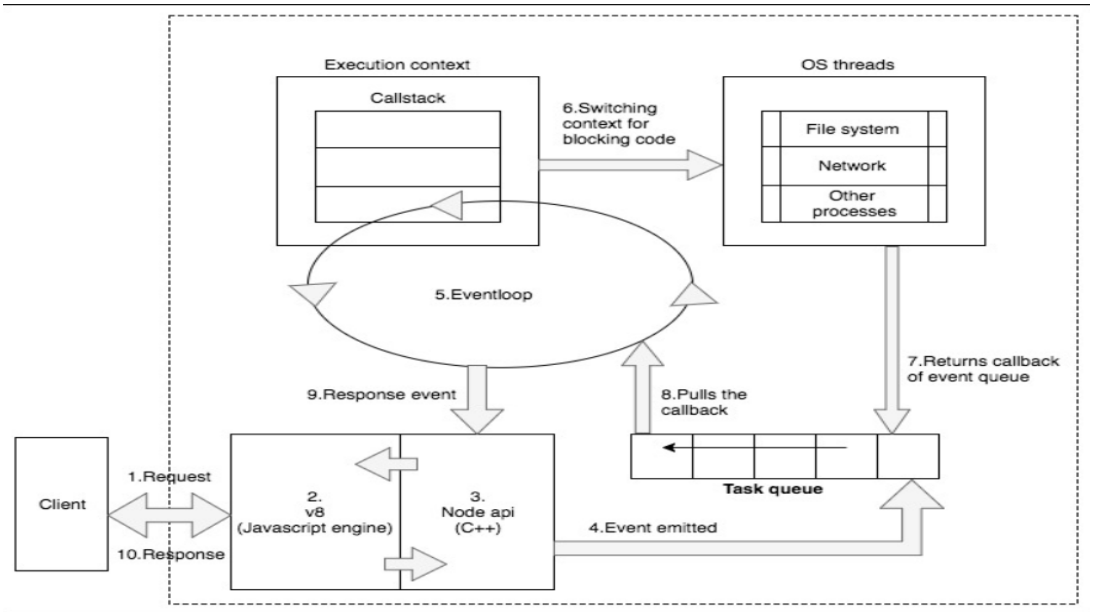
- Node.js

Razvijen je 2013., a osmišljen je kao modularna platforma koju je moguće nadograđivati dodatnim manjim okvirima u svrhu optimizacije JavaScript aplikacija. Node.js omogućava izradu klijent i server strane u jednom programskom jeziku.

U uobičajenim standardnim sinkronim okvirima, izvršavanje jedne operacije uvjetuje početak izvršavanja sljedeće. Node.js je pak asinkron, odnosno *event-driven* stoga sam model definira da se događaji odvijaju zasebno ovisno o pozivu. Node.js također dolazi s ugrađenim upraviteljem paketa - npm (eng. *Node Package Manager*) koji trenutno sadrži oko 150,000 dostupnih paketa, a preuzimaju se i integriraju u aplikaciju jednostavnom naredbom u terminalu. Potrebno je naglasiti da je Node.js dobro osmišljen za izradu aplikacija koje trebaju raditi u stvarnom vremenu kao što su online igre, alati za suradnju, chat aplikacije i slično.

Ovaj okvir najčešće se nadograđuje Express.js-om koji osigurava uspješno rukovanje međuprogramima u zahtjev-odgovor ciklusu, kao i usmjeravanje zahtjeva klijenata. Express.js također omogućava mehanizme za izradu predložaka u svrhu stvaranja dinamičkog sadržaja na web stranicama kao i brzo uspostavljanje veze s bazom podataka.

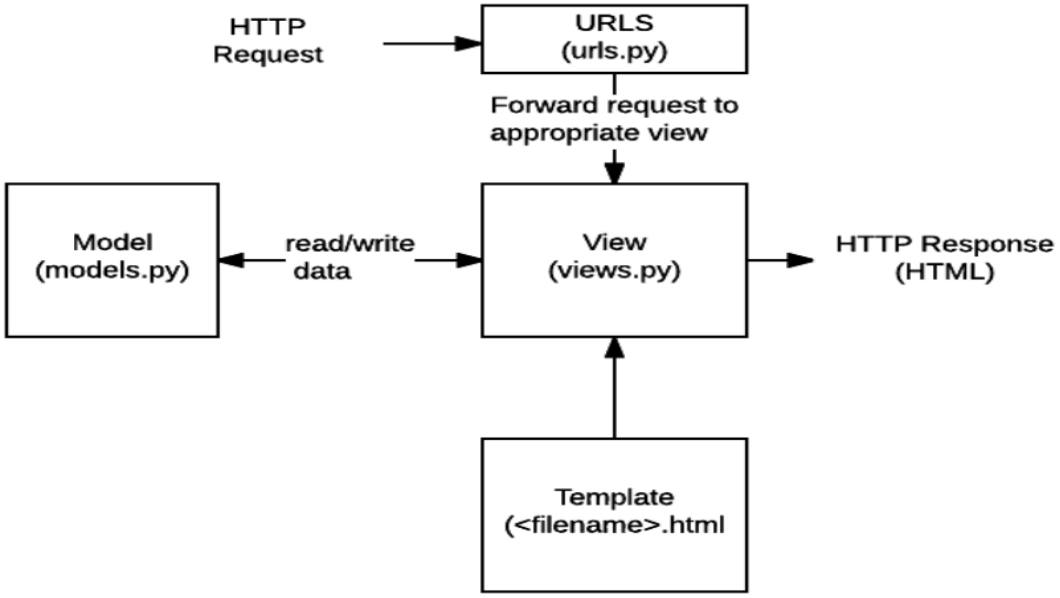
Jednostavno rečeno, u Node.js-u su implementirane mnoge funkcionalnosti i moduli koji su u potpunosti prilagođeni JavaScript jeziku, što ovaj okvir čini adekvatnim odabirom u razvoju web aplikacija.



Slika 2.3 Dijagram Node.js sekvenci [7]

- Django

U slučaju odabira Pythona kao razvojnog jezika, popularan okvir koji podržava sintaksu je Django. Django je primarno osmišljen kako bi olakšao razvoj *backend* strane aplikacije, ali ima mogućnost isporuke sadržaja u raznim formatima što ga čini kompatibilnim s brojnim okvirima *frontend* strane. Django automatizira sigurnosne mjere u svrhu zaštite računa, šifri i podataka korisnika [11]. Svaki dio u Django arhitekturi neovisan je o ostalima što cjelokupnu strukturu projekta u ovom okviru čini skalabilnom i lakom za modificiranje. Django web aplikacije uobičajeno sadrže grupirane dijelove koda od kojih se svaka vrsta procesa odvija u zasebnim dokumentima.

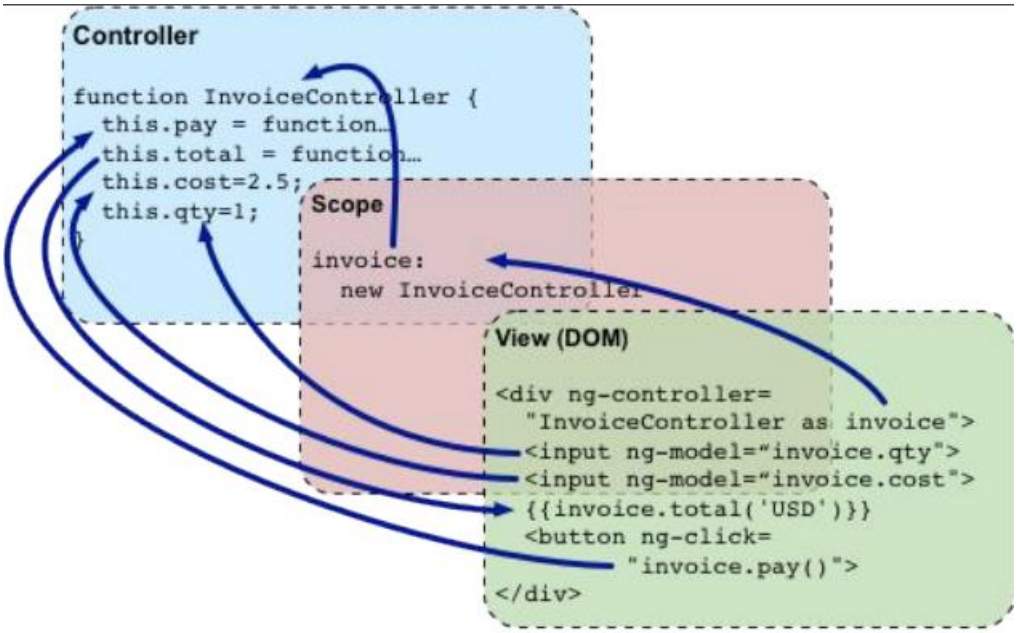


Slika 2.4 Arhitektura Django aplikacije [11]

- AngularJS

Okviri za implementaciju klijent strane web aplikacije su brojni i mnogo je poznatih alata, a među njima se nalazi i AngularJS razvijen u TypeScriptu koji u kombinaciji s Node.js-om omogućava realizaciju modernih web aplikacija. Osnovna arhitektura AngularJS aplikacija temelji se na standardnoj MVC strukturi uz dodatak komponenti kao što su servisi, direktive i datoteke. Organizacija koda moguća je na više različitih načina, primjerice pisanje koda na istoj razini, stereotipno kategoriziranje, struktura po specifikacijama, itd. [12]. AngularJS također sadrži brojne direktive koje omogućavaju lakše upravljanje DOM-om za postupanje s HTML dokumentima poput ngIf, ngFor,

ngController, ngShow, ngModel i još mnoge. Direktive predstavljaju snažnu tehnologiju koja podržava izgradnju obnovljivih komponenti čime se uspješno sprječava dupliciranje i općenita redundancija koda [12].

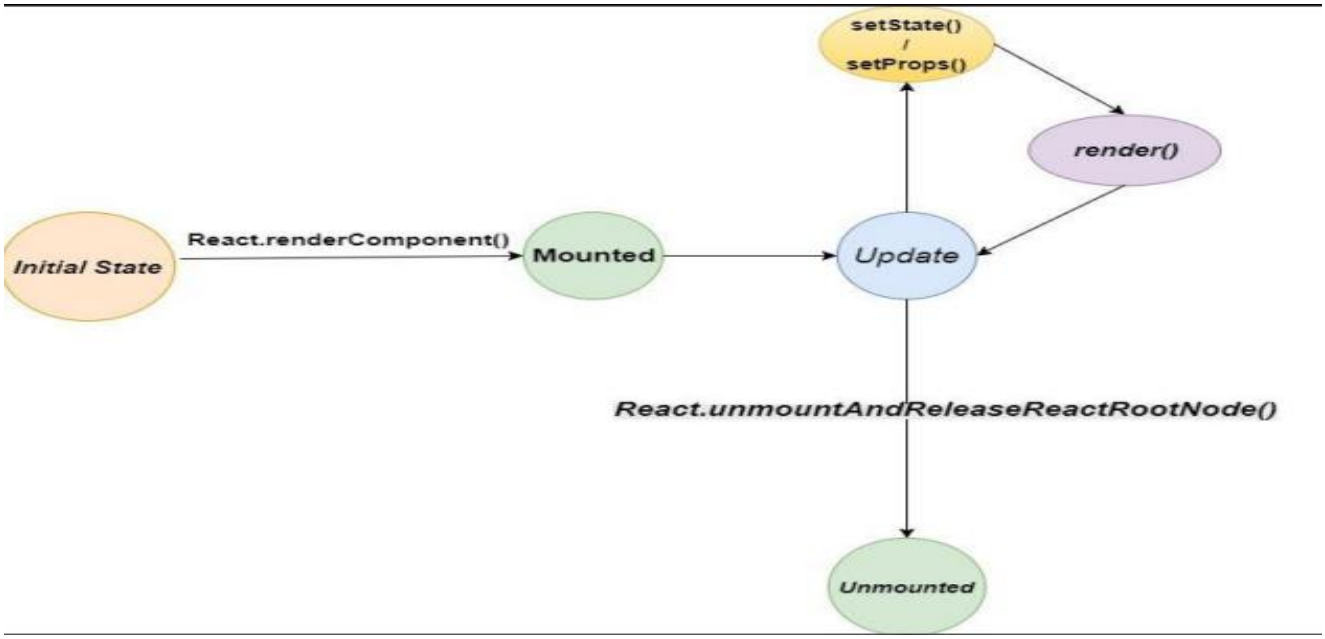


Slika 2.5 Dijagram interakcije AngularJS komponenti [12]

- ReactJS

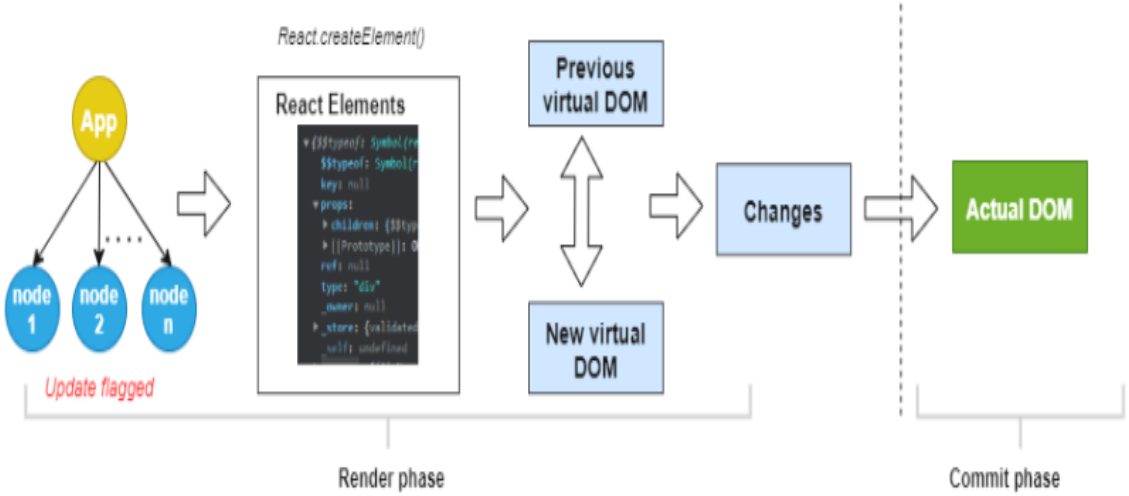
Također zasnovan na komponentama, ReactJS postao je uobičajen okvir za izvedbu interaktivnih korisničkih sučelja. I on uključuje klasičnu MVC strukturu programa, a u osnovi omogućuje razvoj velikih i složenih aplikacija temeljenih na webu koje mogu promijeniti svoje podatke bez naknadnih osvježavanja stranica. Cilj mu je pružiti bolja iskustva korisnika brzim i snažnim razvojem web aplikacija. ReactJS može se integrirati s drugim JavaScript bibliotekama ili okvirima u MVC, kao što je opisani AngularJS [13].

Svaka komponenta u ReactJS-u ima strogo definiran životni ciklus u koji je podijeljen na faze montiranja (eng. *mounting*), ažuriranja (eng. *updating*) i demontiranja (eng. *unmounting*) komponente. Proces implementacije komponente započinje podizanjem, odnosno *mounting*-om iste nakon čega prelazi u fazu ažuriranja. Korištenjem *setState*() metode ili mijenjanjem parametra, odvija se ažuriranje, a zatim se stablo komponenti regenerira i te time minimalizira manipuliranje stvarnim DOM-om. U konačnici, stanje prelazi u demontirano, a događa se kada aplikacija ne zahtjeva nova renderiranja opisane komponente (Slika 2.6).



Slika 2.6 Životni ciklus ReactJS komponenti [13]

Glavna značajka po kojoj se ReactJS razlikuje u odnosu na druge klijentske okvire je način generiranja sučelja. ReactJS u fazi renderiranja poziva virtualni DOM, a tek nakon renderiranja pristupa stvarnom DOM-u.



Slika 2.7 Faze renderiranja u ReactJS-u [13]

Tablica 2.2 prikazuje danas popularne kombinacije programskih jezika i okvira za izradu prilagođenih aplikacija.

KOMBINACIJA	BACKEND	FRONTEND
MERN	Node.js + Express	React
LAMP	Linux + Apache + MySQL + PHP	HTML + CSS + JavaScript (jQuery)
MEAN	Node.js + Express + MongoDB	AngularJS
MEVN	Node.js + Express + MongoDB	Vue
RoR (Ruby on Rails)	Ruby + Rails + Postgresql ili MySQL	HTML + CSS + JavaScript

Tablica 2.2 Uobičajene kombinacije tehnologija za izradu prilagođenih web aplikacija

2.2.2. Sigurnost web okvira za prilagođene aplikacije

Kao što je prethodno opisano, prilikom odabira web okvira koji će se koristiti za izradu aplikacije, moguće se odlučiti za različite vrste istih, ovisno o opisu funkcionalnosti i željenom načinu rada aplikacije. Neovisno o tome radi li se u konačnici o *Model-View-Controller, Component based* ili *Full-Stack* tipu okvira, među najvažnijim faktorima ubraja se razina implementirane zaštite odabranog.

Slijedi lista uobičajenih okvira i njihove mogućnosti zaštite od uobičajenih prijetnji sigurnosti na webu:

- Django – već spomenuti web okvir koji je najčešći izbor za programiranje u Python programskom jeziku. Ovaj okvir ima ugrađen *QuerySet* API za sprječavanje SQL injekcija. To radi tako što automatski parametizira upite na bazu kako bi se spriječilo ubacivanje. Django također omogućava umetanje CSRF tokena unutar *form* elementa. Ovaj okvir između ostalog olakšava programerima svojim šablonama koje osiguravaju od XSS napada kao i dobro implementiran sustav za provjeru autentičnosti.

- Node.js – također opisani okvir na strani poslužitelja za JavaScript aplikacije. On dolazi s mogućnošću pokretanja web aplikacije pomoću HTTPS-a za zaštitu podataka. Ipak, kako je originalno koncipiran minimalistički, za daljnju nadogradnju sigurnosti potrebno je koristiti okvire koji dolaze kao dodaci Node-u, primjerice Express.js, Next.js. Ni oni ne omogućavaju potpunu zaštitu pa se za razvoj sigurne aplikacije traži instalacija dodatnih paketa i biblioteka (cookie-session, helmet, mongoose, itd.). Node.js dakle uvelike ovisi o razvojnoj zajednici, ali pozitivna strana je da je zajednica vrlo jaka stoga postoji mnogo dobro razvijenih modula koji uz ostale mogućnosti nude i dobru razinu sigurnosti.
- ASP.NET Core – nadogradnja .NET okvira za razvoj web aplikacija. On koristi Razor - koji HTML kodira prema zadanim postavkama sprječavajući time XSS tip napada, za prikaz u pregledniku. Na sličan način osigurani su i SQL upiti, kao i zahtjevi za autentifikaciju (Identity). Još jedna velika prednost ovog okvira je laka mogućnost zaštite podataka koja se realizira jednostavnim instanciranjem instance pružatelja zaštite od koje se zatim zatraži da podatke učini apstraktnima. Okvir pomoću ovoga sam rotira ključeve, vrijednosti i pohranu.
- Java Spring – vrlo popularan i robustan okvir za izradu aplikacija. Spring u sebi sadrži okvir za zaštitu (Spring Security) koji vodi računa o ispravnosti autentifikacije i autorizacije u aplikaciji. Isto tako, postoji okvir za čuvanje šifriranih podataka za pristup drugim uslugama ili aplikacijama. Springova metoda *HtmlUtils.htmlEscape* brine o XSS napadima tako što izbjegava znakove niza koji se vraćaju u korisničko sučelje.

Navedeno je samo par primjera okvira koji su danas često u uporabi, ali za koji god da se okvir u konačnici odlučimo bitno je da znamo odgovoriti na osnovna pitanja o sigurnosti. Ima li taj okvir, primjerice, ugrađen sustav za autentifikaciju koji garantira da će samo korisnici s pravima imati pristup aplikaciji, da li okvir ima siguran sustav za pohranu podataka, ima li sigurne prakse kodiranja i slično.

2.3. Posluživanje web aplikacije

Proces premještanja koda s lokalnog računala na javni poslužitelj u razvoju aplikacije naziva se raspoređivanje, odnosno objavljivanje (eng. *deployment*) aplikacije. Ovo je

moguće pomoću poslužitelja (eng. *host*) koji pruža web lokaciju na kojoj će web aplikacija biti dostupna.

Prije ove faze potrebno je provesti više koraka kako bi se isporuka aplikacije uspješno izvršila, a i održavanje iste omogućilo. Ovo se ponajviše odnosi stvaranje sigurnosne kopije i inačica web aplikacije, prolaznost testova u vidu uspješnog izvršavanja funkcionalnosti, formatiranja i slično.

Objavljivanje koda danas je standardizirano pomoću distribuiranog sustava kontrole verzija – GIT-a. GIT se koristi se za upravljanje kodom tijekom razvoja, a kasnije i održavanja. Pomoću GIT-a moguće je stvarati takozvane grane (eng. *branch*) koje sadrže dijelove koda s određenom funkcionalnošću. Na ovaj se način izrada web aplikacije dijeli u manje segmente koda koji se tek nakon provjere ispravnosti spajaju (eng. *merge*) u glavnu granu s ostatkom koda. Kako bi ovo bilo moguće *online*, postoje brojni GIT poslužitelji koji se koriste kroz cijeli proces razvoja aplikacije, od koordinacije rada u timu do finalne isporuke ukupnog sadržaja aplikacije. Primjeri popularnih GIT poslužitelja danas su GitHub, GitLab, Bitbucket i slični. Jednom kada glavna grana sadrži potpun kod aplikacije, ostvaruje se uvjet za isporuku aplikacije uživo, što prvenstveno zahtjeva odabir poslužitelja.

Ovaj proces je toliko uhodan tako da je konačan repozitorij moguće gotovo automatski prenijeti na poslužitelj. Današnje prakse GIT poslužitelja automatiziraju objavljivanje novih verzija aplikacije čim se promjene u kodu dogode i pohrane na *main* grani.

Kako bi aplikacija bila manje izložena riziku prekida pružanja usluge na web-u ili uspješno prikazala izmjene funkcionalnosti, dobra je praksa koristiti poslužitelj koji ima implementiran automatiziran sustav zaštite. Još jedna važna stavka koja se razmatra pri odabiru poslužitelja je širina mrežnog pojasa (eng. *bandwidth*) tog poslužitelja, a konačna odluka ovisi o obujmu aplikacije, odnosno količini sadržaja koji se pohranjuje. Dobro je znati i da velik broj poslužitelja danas osigurava automatsku sigurnosnu kopiju aplikacije kako bi zaštita od gubitka bila osigurana.

Intuitivno korištenje jedna je od najvažnijih tehničkih značajki kod poslužitelja. Cijeli proces isporuke i posluživanja aplikacije, uvelike je olakšan ako se koristi poslužitelj koji ima jednostavno strukturirane upute i zahtijevane korake. Programerima web aplikacija ovo značajno štedi vrijeme, a to je najjednostavnije izvedivo ako se koriste popularni okviri i/ili sustavi za upravljanja sadržajem.

Posljednja značajka web poslužitelja koja se uzima u obzir je sama mogućnost skalabilnosti. Jednom isporučena web stranica nije nužno gotov proizvod u vidu sadržanih funkcionalnosti, ali i izgleda, stoga je dobro odabrati poslužitelj koji podržava izmjene i rast poslužene web aplikacije.

Danas su dostupne različite vrste tipova poslužitelja, a oni se dijele na: dijeljeno posluživanje (eng. *shared hosting*), posluživanje privatnog virtualnog servera (eng, *virtual private server hosting*), posluživanje posvećeno jednoj aplikaciji (eng. *dedicated hosting*), posluživanje na oblaku (eng. *cloud hosting*), čak i posluživanje WordPress upravljanjem (eng. *WordPress managed hosting*).

Dijeljeno posluživanje prigodno je za male stranice i *start-up* projekte jer je vrlo ekonomično i sigurno za aplikacije s očekivanim malim prometom dok s druge strane, ne štiti od mogućih prekida usluga i sporijeg učitavanja.

Za aplikacije koje imaju veliku količinu prometa, adekvatno je posluživanje pomoću privatnog virtualnog poslužitelja ili posluživanja na oblaku, koji imaju dodatne sigurnosne prilagodbe kao i održavanje brzine povezivanja.

Ovisno o odabiru tipa poslužitelja kreću se cijene istog, a raspon je uokviren od desetak do 80 eura godišnje. Cijena je naravno, definirana prema čimbenicima kao što su pohrana, broj web stranica, IP adresa, SSL certifikati, korisnička podrška i moguće dodatne opcije [14].

Među popularnim poslužiteljima danas nalazimo Netlify, koji predstavlja posluživanje statičkih i dinamičkih web aplikacija putem oblaka, koristeći moderne okvire i alate kao što su Vue, AngularJS, ReactJS, i drugi. Moguće je povezati GitHub, GitLab, ili Bitbucket račun s ovim poslužiteljem, kako bi se kod automatski isporučivao na globalnu mrežu. Netlify ima web sučelje i API-je za upravljanje stranicama i lokacijama. Također, kod objavljen ovim putem ima obilježja kao što je HTTPS, prilagođene domene, preusmjeravanja, forme, i više.

Netlify naplaćuje na temelju potrebnog *bandwith-a,* tako što cijena raste u skladu s količinom podataka koju zahtjeva posluživanje aplikacije. Moguće je odabrati različite planove i razine, ovisno o značajkama i funkcionalnosti.

Što se skalabilnosti tiče, Netlify dopušta globalni razvoj poslužene aplikacije distribucijom sadržaja preko CDN-a, a pomoću njega osigurano je automatsko održavanje, infrastruktura i dostupnost.

Vrlo zastupljen poslužitelj je i Bluehost koji nudi čak šest vrsta posluživanja uključujući: dijeljeno posluživanje, posvećeno posluživanje, posluživanje preko privatnog virtualnog servera, dijeljeno posluživanje WordPress-a, upravljano posluživanje WordPress-a (WP Pro) i WooCommerce posluživanje za e-trgovine.

Kao i drugi, ovaj poslužitelj nudi različite planove ponude i cijene, a svaki od njih uključuje besplatnu domenu godinu dana, besplatan CDN, besplatan SSL certifikat, skalabilnost, cjelodnevnu korisničku podršku.

Uz razne druge prednosti, jedna od važnijih značajki Bluehost-a je mogućnost dijeljenog i samostalnog posluživanja dizajniranog za WordPress stranice. U principu, sve su mogućnosti iste kao i kod običnog dijeljenog posluživanja, ali s dodatkom prilagodbe WordPress specifičnim karakteristikama.

Iako, ako je planirana stranica velikog obujma ili zahtjeva stalnu nadogradnju, bolja opcija za posluživanje je preko WP Pro Bluehost plana.

Uz Netlify i Bluehost, na listi popularnih poslužitelja nalaze se još i Ionos, HostGator, DreamHost i Inmotion. Sve ih karakteriziraju danas standardizirane značajke posluživanja, a određeni faktori čine ih adekvatnim za specifičan tip web aplikacija.

3. Usporedba pristupa izrade aplikacija na prototipovima

Ovaj rad potkrijepljen je izrađenim prototipovima projekata u vidu dviju web stranica razvijenim različitim pristupima. Obe su stranice isključivo informativne, te do sada ne zahtijevaju pohranjivanje ili zahtjevnije operacije nad podacima. Vlasnici proizvoda su dvije firme koje ga koriste za promoviranje vlastitog poslovanja – čarter firma i firma za energetsko savjetovanje i projektiranje.

U ovom poglavlju za svaku navedenu web stranicu, opisan je pristup vlasniku projekta, svrha aplikacije, odabir tehnologija i okruženja, koraci pri izradi, raspodjela posla kao ograničenja i problemi.

3.1. Svrha aplikacija i prikupljanje zahtjeva

Kako je spomenuto radi se o dvije stranice s istim ciljem, ali različitim obujmom sadržaja. U konačnici, svrha web stranice je prezentiranje poslova koje tvrtka obavlja.

- Čarter agencija

  Vlasnik projekta ove stranice osnivač je tek formirane čarter kompanije koja planira iznajmljivati brodove nudeći planirana putovanja, gurmansko jedrenje, osiguravanje smještaja i prijevoza kao i organiziranje događaja poput vjenčanja ili drugih slavlja.

  Danas se ovakvi izleti rezerviraju i kupuju gotovo isključivo online, pa je potreba za web stranicom od početka razvoja neupitna.

  U komunikaciji s kupcem, dogovorena je struktura buduće stranice, ali i opseg mogućih zahtjeva i značajki, te okvirni budžet.

- Firma za energetsko projektiranje

  Veća je kompanija od prethodne, nastala 2015. godine, a još od 2017. promovirala je ponudu usluga preko web stranice.

Ipak, razvojem sveukupnih web aplikacija i stranica, ali i širenjem područja posla koji obavlja, ova se tvrtka našla u potrebi za rekonstruiranjem i redizajniranjem sadržaja web stranice.

Ovo je ujedno i konzultantska kompanija koja radi na implementaciji i zamjeni starih rasvjetnih tijela led lampama, ali i drugih energetski obnovljivih sustava. Ima specifičan način poslovanja, vođen ESCO (eng. *Energy savings company*) modelom kojim promovira zelenu energiju, održivost i uštedu.

Također nude stupanje u kontakt preko web-a, bilo da se radi o prijavi za posao ili početku suradnje sa zainteresiranim kupcem.

Kako su područja ove kompanije opsežnija, sadržaj stranice je znatno veći od prethodno opisane firme.

3.2. Faze izrade

Po uzoru na ranije opisan životni ciklus web aplikacija, razvoj web stranica obuhvaćao je iste faze razvoja.

Poglavlje navodi i opisuje procese izrade u svim fazama za obje aplikacije pojedinačno.

3.2.1. Prototip aplikacije korištenjem prilagođenog kodiranja

1. Analiza prikupljenih zahtjeva

U ovom slučaju zahtjevi su prikupljani u više iteracija s obzirom da je i sama tvrtka tek osmišljavala poslovni plan i prikupljala resurse.

Kako je uspješnost poslovanja čarter firmi vrlo ovisna o vremenskim okvirima (usluge su moguće samo određen broj mjeseci), prvi i može se reći najvažniji zahtjev, bio je poštivanje roka za isporuku stranice. Cilj je bio realizirati sadržaj na web-u do najkasnije početka proljeća te godine.

Drugi zahtjev vezan za sadržaj je da stranica bude konstruirana intuitivno s naglascima na brodove koji se iznajmljuju, ali i dodatnim uslugama u sklopu kompanije.

Što se tiče zahtjeva vezanih za interakciju korisnika i stranice, tražena je obična forma koja prosljeđuje upit na mail adresu vlasnika.

Dizajn stranice mora biti moderan i prilagođen ciljanoj publici, te odgovarajući (eng. *responsive design*) za sve veličine različitih ekrana s kojih se može pristupiti stranici. Sama struktura mora biti skalabilna u vidu dodavanja i/ili mijenjanja ponude brodova i usluge.

2. Planiranje razvoja

Razvoj stranice planiran je u dvočlanom timu, kojeg čine studenti PMFST-a, jedan od njih programer s iskustvom. Tehnologija i okruženje odabrani su po posljednjim standardima u razvoju web aplikacija, ali i dotadašnjim susretom programera s istim.

Kao programski jezik odabran je TypeScript, potpomognut Node.js-om u paru s Next.js-om, pomoću kojeg je podržan razvoj korištenjem ReactJS-a. Dizajn je ostvaren uz pomoć Tailwind CSS-a, modernog okvira za CSS koji je opisan u slijedećoj fazi. Također je postavljena MongoDB baza podataka kako bi se u budućnosti rastom sadržaja moglo brzo i efikasno pristupati i učinkovito podijeliti podatke.

Slika 3.1 Glavne značajke Next.js-a [7]

Da bi se omogućila suradnja između programera, praćenje razvoja podijeljenog na manje zadatke, te pristup ukupnom kodu napravljen je GitLab repozitorij s glavnom granom koja je u konačnici „podignuta“ na web pomoću Netlify-ja.

Tijek razvoja osmišljen je tako da se ukupan obujam posla dijeli na manje cjeline u obliku zasebnih grana na GitLab-u, te po izvršavanju spajanje u glavnu granu s ostatkom dovršenih funkcionalnosti. Primjeri takvih zadataka su izrada zasebnih komponenti koje

predstavljaju određeno sučelje ili dio stranice, osiguranje responzivnosti, funkcije alatne trake, prilagodba i dodavanje slika i slično.

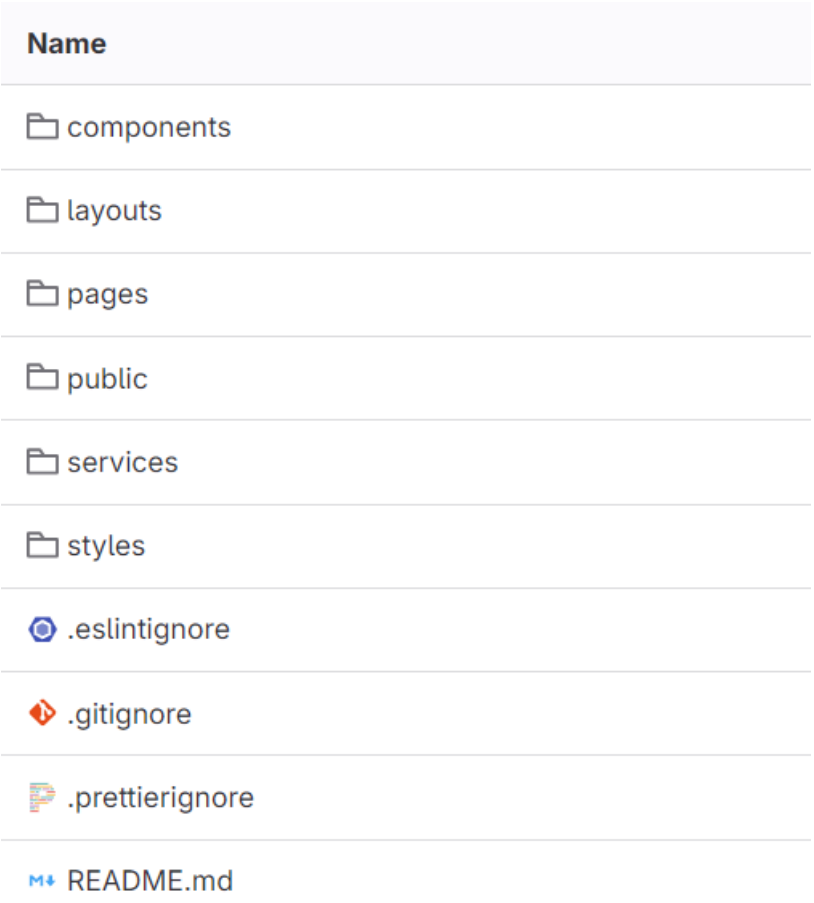
Sve navedene tehnologije opisane su u prethodnim poglavljima rada, a u početnoj fazi projekta, programeri su poznavali način funkcioniranja cijele arhitekture kao i sintaksu korištenog jezika.

3. Oblikovanje i dizajn

Kako bi se osiguralo dobro korisničko iskustvo (eng. *User Expirience*), u dogovoru s kupcem, definirale su se glavne komponente stranice i njihova lokacija, kao i tijek korištenja stranice.

Odabir boja, slika, fonta, te drugih multimedijalnih sadržaja vođen je primjerima modernih stranica sličnog sadržaja, a realiziran pomoću Tailwind CSS klasa i komponenti.

Informacijski dizajn rezultat je podjele koda na dokumente i mape prikazane na sljedećoj slici.



Slika 3.2 Struktura datoteka stranice

U mapi komponente sadržane su sve komponente koje oblikuju stranicu. U njoj se nalazi zaseban kod za svaki modul stranice, od zaglavlja i podnožja do komponenti koje su izrađene za gumb, ikone, logo tvrtke, meni, formu i druge.

Mapa rasporedi (eng. *Layouts*) dijeli kod na tri osnovne podjele glavnih komponenti – jedna u slučaju prikaza početne stranice, druga za prikaz samostalnog broda kao proizvoda i zadani prikaz, u slučaju dodavanja novih putanja.

Mapa Stranice sadrži sve komponente u obliku stranica do kojih se može doći a prikazuju se kao nova stranica. Tu u pripada glavna stranica, kontakt forma kao zasebna putanja, stranica za ne postojeću putanju – 404, te stranica s informacijama o pravilima zaštite privatnosti.

U javnom folderu sačuvane su sve slike koje se prikazuju na stranici, a u servisima je sadržana logika za implementaciju prikaza učitavanja stranice i slanja maila. Mapa Stilovi ima samo jedan dokument u kojem su napisana zadana pravila i odabrani stilovi koji se pozivaju u svim dijelovima stranice.

Ovakva struktura aplikacije, uvelike je olakšala snalaženje tokom pisanja koda, a konzistentnost u pridržavanju strukture otvorila je mogućnost jednostavne nadogradnje postojećih, ali i dodavanje mogućih budućih komponenti ili funkcionalnosti (npr. dvojezičnost).

Kao što je rečeno stil stranice u potpunosti je nastao na temelju TaiwindCSS-a i njegovih komponenti kao što je karusel, kartice sa slikama i tekstom, dizajn forme, i slično, a brojne klase ovog alata učinile su stranicu visoko adaptivnom i responzivnom. Pristupimo li stranici s laptopa, tableta ili mobitela različitih veličina, sadržaj će se prikazati prilagođen veličini i mogućnostima ekrana bez nelogičnosti ili gubitaka.

4. Implementacija

U fazi implementacije ove stranice, aktivno se radilo na traženju materijala, slika, informacija o kompaniji, kao i programiranje samog sučelja. Provjere i potvrde od strane kupca, definirale su tempo kreiranja. Kako je stranica razvijena u timu od dvoje ljudi, dizajn, kodiranje, ugovaranje sastanaka i reorganizacija podijelila se ravnomjerno.

Projekt je došao do točke gdje kupcu možemo pokazati konkretna rješenja, prikaze i funkcionalnosti.

Postavljeni su naslovi i SEO, te više nije bilo promjena u danim zahtjevima.

Konačno, dovršen kod podignut je na GitLab-u te je započelo finalno testiranje svih komponenti.

5. Testiranje i validacija

Kako razvijena stranica nema kompleksne mogućnosti u korisničkoj interakciji, dovoljno je bilo osnovno testiranje za provjeru pristizanja maila i valjanosti popunjene forme. Provjerene su sve putanje, linkovi i akcije koje se pokreću klikom na gumb.

Nakon utvrđivanja ispravnosti, pomoću Netlify-ja, stranica je producirana i javno dostupna. Kao što je prethodno opisano, Netlify ima vrlo interaktivno, intuitivno i jednostavno sučelje za implementaciju web stranica. Uz pomoć tog sučelja definirano je ime domene, IP adresa te konfiguriran DNS.

Naknadne evaluacije uključivale su provjeru responzivnosti i izgleda na različitim pretraživačima kao i implementiranu razinu SEO-a.

6. Održavanje

Stranica je „oživjela“ prije godinu i šest mjeseci otkada kompanija nije više ni znatno rasla. Drugi oblici promidžbe, društvene mreže, koristile su sadržaj napravljene stranice kako bi se firma konzistentno prezentirala i vremenom kreirala svoj jedinstveni brend.

U dogovoru s kupcem, status stranice redovito je provjeravan, a cilj je obraćanje pažnje na održavanje relevantnosti sadržaja i praćenje današnjih trendova.

3.2.2. Prototip aplikacije izrađen u sustavu za upravljanje sadržajem

Ovaj projekt još uvijek je u implementacijskoj fazi unatoč predviđanju da će do jeseni biti u potpunosti završen. Razni faktori utjecali su na odgodu krajnjeg rezultata, ali manjak zaposlenika u kompaniji uvelike otežava i usporava proces izrade. Tvrtka u ovoj fazi doživljava eksponencijalan rast obujma poslova, a isti su tempirani u ovisnosti o natječajima raznih gradova i općina, stoga je teško pronaći vrijeme za posvećenu isporuku teksta i slika.

1. Analiza prikupljenih zahtjeva

Kako je postojeća stranica ove kompanije već zastarjela, osnovni zahtjev kupca bila je modernizacija prikaza sadržaja kao i dodavanje novih poslovnih prilika, buduće poslovne politike i referenciranje na dosada izrađene projekte.

Stranica mora imati dobro kategorizirane naslove i lokacije, a ciljana publika su ujedno kupci i potencijalni zaposlenici.

Među osnovnim zahtjevima naglašeno je da stranica mora predstavljati ozbiljnost tvrtke i mogućnost pregleda svih aspekata poslovanja na nivou konkurentnih kompanija.

Mogućnost stupanja u kontakt preko stranice realiziran je slično kao i kod prethodnog projekta uz uvjet da se u formu može učitati pdf ili word datoteka kako bi zainteresirani budući zaposlenici mogli priložiti svoj životopis ili portfolio.

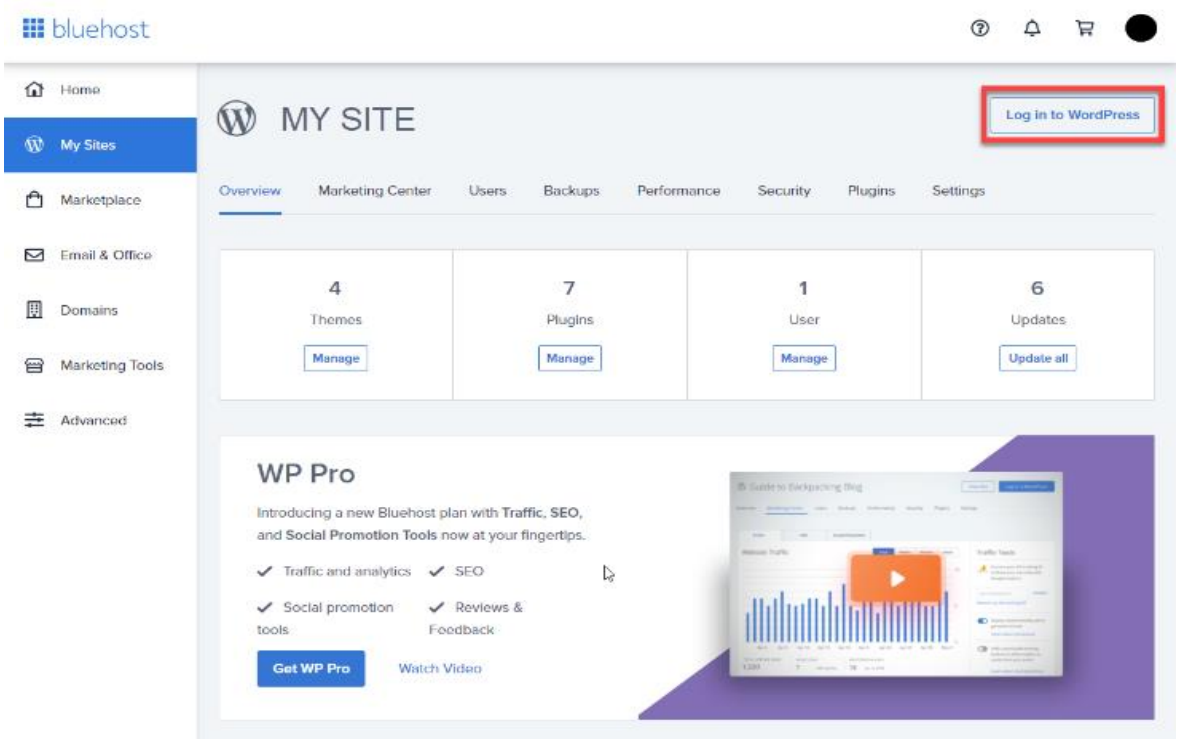
U trenutku dogovaranja posla, krajnji rok isporuke nije definiran, a bliska suradnja i komunikacija je obavezna zbog prikupljanja sadržaja i činjenica vezanih uz opise kompanije, projekata, servisa.

2. Planiranje razvoja

Pomoću sugestija drugih programera i informacija o konkurentnim tvrtkama, odabran alat za razvoj stranice je WordPress. WordPress je popularan CMS koji se konstantno ažurira, nadograđuje i optimizira.

Ovaj alat dozvoljava izgradnju stranice bez pisanja koda, uz moguće uvođenje dodatnih CSS svojstava ili klasa, što cijeli proces čini podosta lakšim, ali i drugačijim u odnosu na prethodno opisane tehnologije.

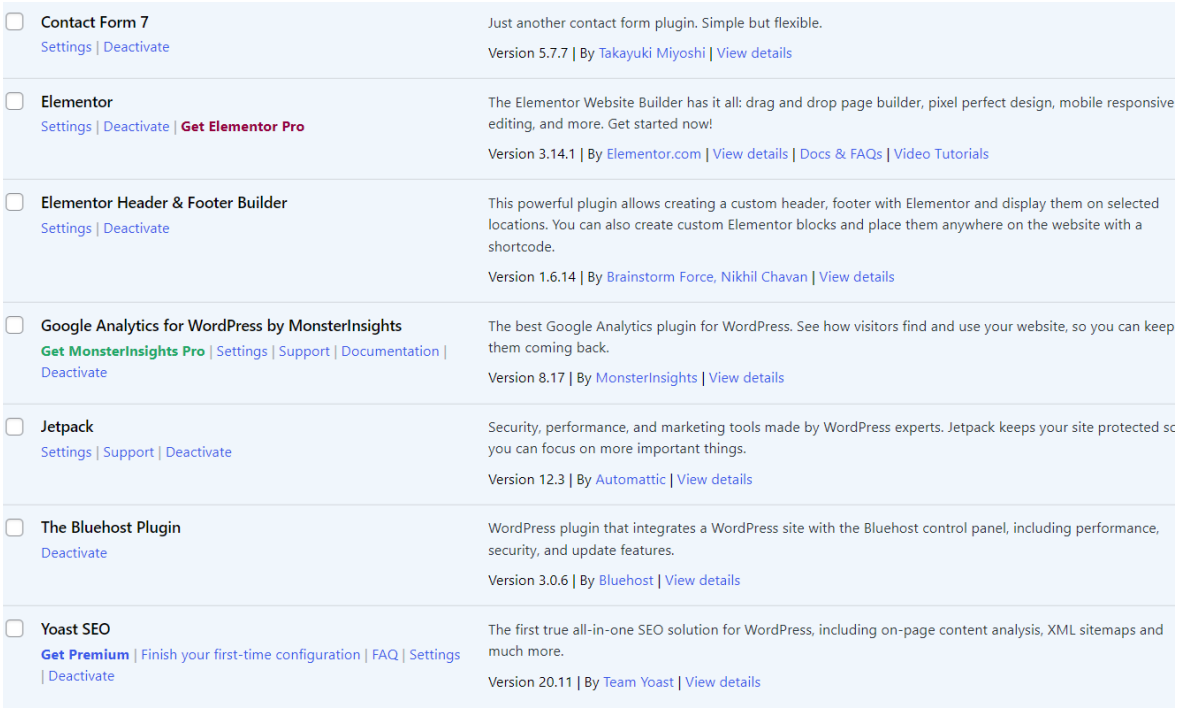
Dodatne tehnologije potrebne za realizaciju su poslužitelj i WordPress dodaci. Kao poslužitelj odabran je Bluehost zbog statusa visoko odobrenog alata za integraciju s WordPress stranicama.



Slika 3.3 Bluehost sučelje

WordPress ima mogućnost dodavanja tema u kojima se zatim odvija razvoj, a dolaze sa setom gotovih predložaka raznih kategorija stranica i dijelova istih. Za ovaj projekt odabrana je i korištena Divi Premium tema s više od 2000 modernih predložaka.

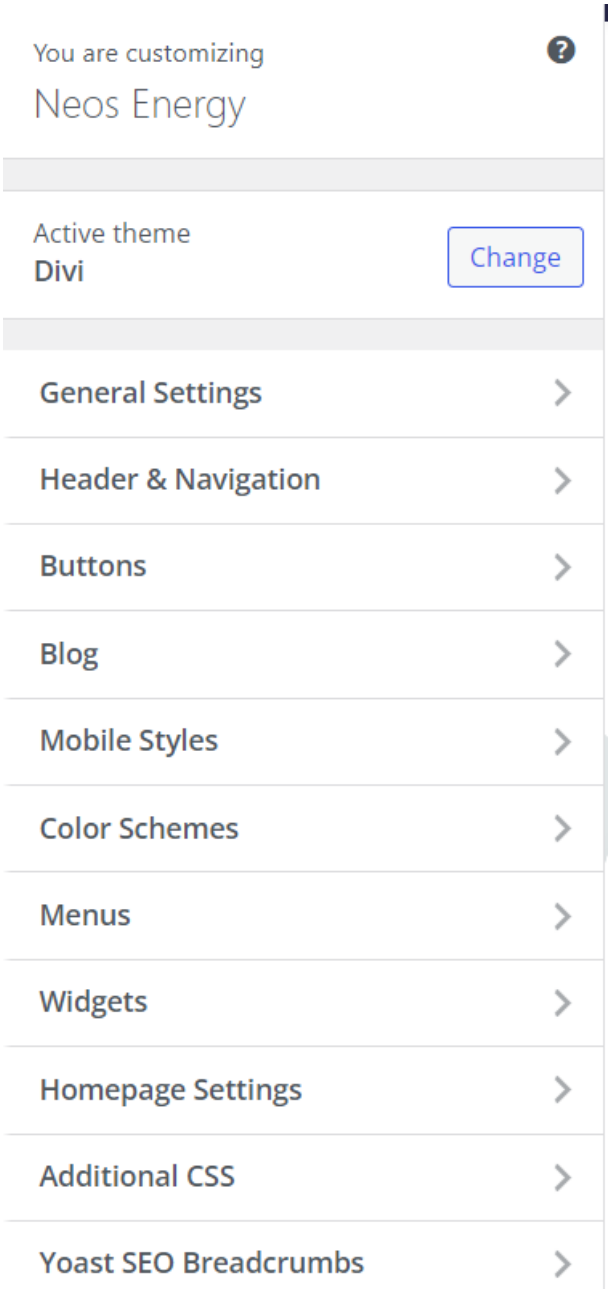
Uz temu i poslužitelj, u WordPress su instalirani dodatni plug-inovi za kvalitetnije i jednostavnije rukovanje stranicom. To su ovom primjeru: *Contact Form 7, Elementor, Google Analytics for WordPress, Yoast SEO,* te *Bluehost plugin.*



Slika 3.4 Prikaz instaliranih dodataka

3. Oblikovanje i dizajn

Početne postavke dizajna definiraju se u samom WordPress-u, a iste se dodjeljuju svim elementima stranice. Na tom se mjestu kreira navigacijska traka, paleta osnovnih boja, stranice, linkovi, zadani font i generalni raspored.



Slika 3.5 Alatna traka WordPress stranice

Nakon ovih postavki započinje proces oblikovanja stranice, u ovom slučaju u Divi Builder-u. Unatoč gotovim predlošcima iz teme, stranica je dizajnirana korištenjem Divi pojedinačnih komponenti, ali i kreiranjem sadržaja unošenjem sekcija koda. Zadržane su dvije osnovne boje i logo sa stare stranice.

```
<script
src="https://cdnjs.cloudflare.com/ajax/libs/slick-
carousel/1.6.0/slick.js"></script>
<script>
jQuery(document).ready(function() {
  jQuery('.divilife-3-col-feature-blurb-slider').slick({
    dots: true,
    slidesToShow: 3,
    slidesToScroll: 1,
    autoplay:true,
    responsive: [
      {
        breakpoint: 980,
        settings: {
          slidesToShow: 2
        }
      },
      {
        breakpoint: 767,
        settings: {
          slidesToShow: 1
        }
      }
    ]
  });
});
</script>
```

Slika 3.6 Primjer dodatnog koda za promjenu izgleda komponente

Ukupan sadržaj proteže se na četiri stranice od kojih jedna predstavlja početnu, a druge se odnose na konkretne aspekte tvrtke – stranica s informacijama vezanim za kompaniju, stranica s referencama i stranica sa servisima.

Sam WordPress kao alat pokazao se vrlo praktičnim i korisnim u razvoju web stranica. Oblikovanje pomoću ovog sustava za upravljanje, nakon savladavanja osnova, zaista je intuitivno, brzo i lako, stoga ne čudi da je danas velika većina informativnih stranica izgrađena upravo koristeći ovo okruženje.

Uz samo slaganje sadržaja, oblikovanje stranice zahtijevalo je korištenje alata poput Adobe express-a za stvaranje prigodne multimedije i grafičkog dizajna.

Što se responzivnosti stranice tiče, gotovi predlošci imaju implementirane specifikacije za standardizirane veličine ekrana. U slučaju ručnog kreiranja pojedinih komponenti Divi dopušta odabir prikaza ekrana preko kojeg se može podešavati veličina ili razmaci u vidu margina ili podstave u pikselima, postocima i drugim mjernim jedinicama poznatih u klasičnom CSS-u. Navedeno možda predstavlja najvažnije modifikacije potrebne da sadržaj bude posložen kako je zamišljeno, ali iste su preinake moguće za veličinu, okvire, sjenu, transformaciju, animaciju i filtere. Još jedna krucijalna mogućnost kod razvoja različitih prikaza na različitim ekranima je odabir vidljivosti – WordPress Divi nudi

jednostavne *check-box*e za biranje na kojim rasponima dimenzija će se odabran sadržaj prikazati, a u kojima „neće ni postojati“.

Informacijski dizajn u WordPressu sam je po sebi posložen stoga nije bilo potrebe za razvojem istog.

4. Implementacija

Implementacija WordPress aplikacije svodi se na automatizirane radnje potrebne za odabir poslužitelja i dodavanje sadržaja. Bluehost na jednom mjestu nudi imenovanje i zakupljivanje domene, kao i direktnu poveznicu s WordPress sadržajem. Alat za upravljanjem domenom u Bluehost-u ima opcije podešavanja razina sigurnosti, konfiguracije za DNS i Google konzole za pretraživanje.

Ono što je također vrlo važno kod ovako kreiranih stranica je aktivno ažuriranje korisnika, dodataka i tema jer se redovite promjene konstantno prilagođavaju posljednjim standardima web stranica, a starije verzije ostaju nepodržavane što može dovesti do slabljenja kvalitete i sigurnosti postojeće stranice.

Neos stranica još uvijek je u fazi implementacije u vidu konkretne razrade sadržaja teksta i prezentacije projekata kompanije.

5. Testiranje i validacija

Osnovno testiranje stranice dosada je provedeno, ali tek nakon produkcije slijedi testiranje cjelokupne uspješnosti implementacije proizvoda. Moguća usporavanja validacije uzrokuju Divi sučelja koja ne dozvoljavaju prikaz trenutnog stanja stranice dok se nalazimo u razvojnoj fazi, ali WordPress ima prikaz opciju koja prezentira kako su osmišljene sve uređene stavke.

Problem u evaluaciji može predstavljati „prikrivenost“ stvarnih skripti koje se izvršavaju u pozadini dok kreacija stranice ostaje vidljiva samo na grafičkoj prezentaciji.

6. Održavanje

Održavanje ove stranice planirano je po standardiziranim uputama za razvoj web stranica pomoću WordPressa. Ovo se odnosi na spomenuto redovno ažuriranje dodataka i stranice. Kao što Bluehost omogućava uređivanje podataka o domeni i SEO-u u stvarnom vremenu, tako se i koristeći WordPress lako postižu ažuriranja, dodavanje sadržaja, izmjene postojećih i slično.

Pomoću implementiranih i dodatno instaliranih alata predstavljaju se direktni potrebni koraci za optimizaciju i nadogradnju stranice. Takvi alati navode standardizirane sheme izgradnje, ali i automatsko održavanje.

3.3. Problemi i moguće optimizacije

Kao u razvoju svakog projekta, i u navedenim slučajevima postojali su tehnički problemi i potrebne optimizacije.

U prvom slučaju, stranica je razvijena prilagođenim programiranjem. Odabir tog pristupa većinski je uvjetovan zahtjevom klijenta zbog predviđanja mogućih slabosti CMS alata. S druge strane, predznanja razvojnih programera obuhvaćala su klasično kodiranje web sadržaja. Kako vremenski rok nije predstavljao problem u kontekstu realizacije projekta, a potreba za održavanjem je svedena na osnovno godišnje plaćanje godišnjeg prava na domenu i poslužitelj, donijeta je odluka o prilagođenim alatima (TypeScript + ReactJS + Node.js).

Kod izrade drugog prototipa, ipak je odabran pristup realizacije preko CMS alata – WordPress-a na što su utjecali razni faktori. Među njima je prvenstveno planirana sustavna nadogradnja informacija i adaptacija sadržaja. WordPress se zbog toga pokazao kao bolji odabir zbog mogućnosti upoznavanja ostatka zaposlenika s osnovama kako bi se aktivno moglo raditi na mijenjanju sadržaja kao i činjenica da stranica zahtjeva standardiziranu promidžbu takvog tipa poslovanja.

Dok velike aplikacije i sustavi uglavnom imaju uhodane prakse održavanja i razvoja sigurnih značajki, mlađim i manje iskusnim programerima razvoj zaštićenih i održivih aplikacija predstavlja zaseban problem u nedostatku konkretnih slučajeva i znanja.

Kod individualnog razvoja aktivne web stranice, kao što je slučaj sa slijedećim web stranicama, već na samom početku postoji velik broj nepoznanica zbog robusnosti planiranih alata za realizaciju. Od prvotnog kreiranja okruženja, web okviri i alati nude svoje sustave za osiguravanje stranice i sam proces time često postaje slijepo praćenje danih uputa. U konačnici, kada je stranica plasirana i dostupna korisnicima, često je upitno koliko je ista ranjiva ili otporna na moguće napade.

Ipak, stranice moraju bit održive i dostupne stoga je potrebno obratiti pažnju na ostale prijetnje sigurnosti.

- Express.js i Netlify

Kao što je spomenuto u prethodnom poglavlju, Node.js jedan je od najzastupljenijih alata za razvoj aplikacija pisanih u JavaScript-u ili TypeScript-u. Standardan server okvir Node.js-a je Express. U ovom primjeru kao web poslužitelj odabran je Netlify – alat za izradu visokoučinkovitih i dinamičnih web stranica ili aplikacija, kao i e-trgovina.

Netlify omogućava dvije glavne značajke Express aplikacijama: usmjeravanje (eng. *routnig*) i međuprograme (eng. *middleware*). Također, on automatski pruža sigurnu komunikaciju putem HTTPS enkripcije.

Što se uskraćivanja usluge tiče, Netlify štiti od DoS-a tako što prati anomalije u obrascima prometa, prepoznaje i blokira zlonamjerni promet, te osigurava da web stranice i aplikacija ostanu dostupne na Internetu.

Uz automatiziranu zaštitu pomoću ovog okvira moguće su dodatne opcije jačanja sigurnosti koje se odabiru ovisno o tipu aplikacije. U slučaju da se na stranici nalazi privatan sadržaj namijenjen samo za određen skup korisnika, potrebno je postaviti odgovarajuću zaštitu pristupa web stranici. To uključuje zaštitu na razini timske prijave lozinkom koja se može konfigurirati za svaku stranicu u timu ili po razini. Uz pomoć *Netlify Identity* ovakva se zaštita brzo i lako postavlja.

Ono što je primjerice može biti konkretno korisno nakon isporuke stranice je praćenje aktivnosti kako bi klijent bio informiran o broju posjeta i vremena provedenog na proizvodu. S druge strane, programerima Netlifyjev *Site Analytics* omogućava uočavanje neuobičajenih aktivnosti i obrazaca prometa na stranici. *Log Drains* također doprinosi ovakvoj vrsti mjere zaštite utoliko što se njime nadziru logovi prometa, funkcija, funkcija za analizu, upozorenja i postojanost podataka [15].

- WordPress i Bluehost

Velik broj stranica danas napravljen je uz pomoć besplatnog sustava za upravljanje sadržajem – WordPress. Kako korištenje ovog alata ne zahtjeva znanje iz programiranja, izrada web stranica postala je dostupna svima.

Posluživanje stranice razvijene u WordPress alatu često se realizira pomoću Bluehost plana prilagođenom ovoj tehnologiji.

Ipak, kako je WordPress besplatan i lak za korištenje, više nego za ikoji alat potrebno je provjeriti koliko sigurna ovako napravljena stranica može biti i koje mjere zahtjeva kako bi mogli reći da je zaštićena.

Neke od najčešćih ranjivosti vezane konkretno za WordPress su zastarjeli softveri, slabe lozinke, nesigurne teme i dodaci ili nezaštićen poslužitelj.

Jedna od najvažnijih radnji u svrhu zaštite ovakve stranice je redovito ažuriranje WordPress jezgre, tema i instaliranih dodataka. Na ovaj se način svakom verzijom optimiziraju dotad registrirane ranjivosti u kodu. Kao još jedna automatska mjera često se predlaže instalacija dodatka (*plugin*-a) za sigurnost. Ovakvi dodaci redovito skeniraju i blokiraju sigurnosne prijetnje, implementiraju vatro zid (*firewall*) aplikacije, te prate DNS promjene.

Redovita izrada sigurnosne kopije također igra važnu ulogu u slučaju uspješnog napada na stranicu. Neovisno o rezultatu napada, stranicu je uvijek moguće vratiti u izvornom obliku pomoću ove kopije [16].

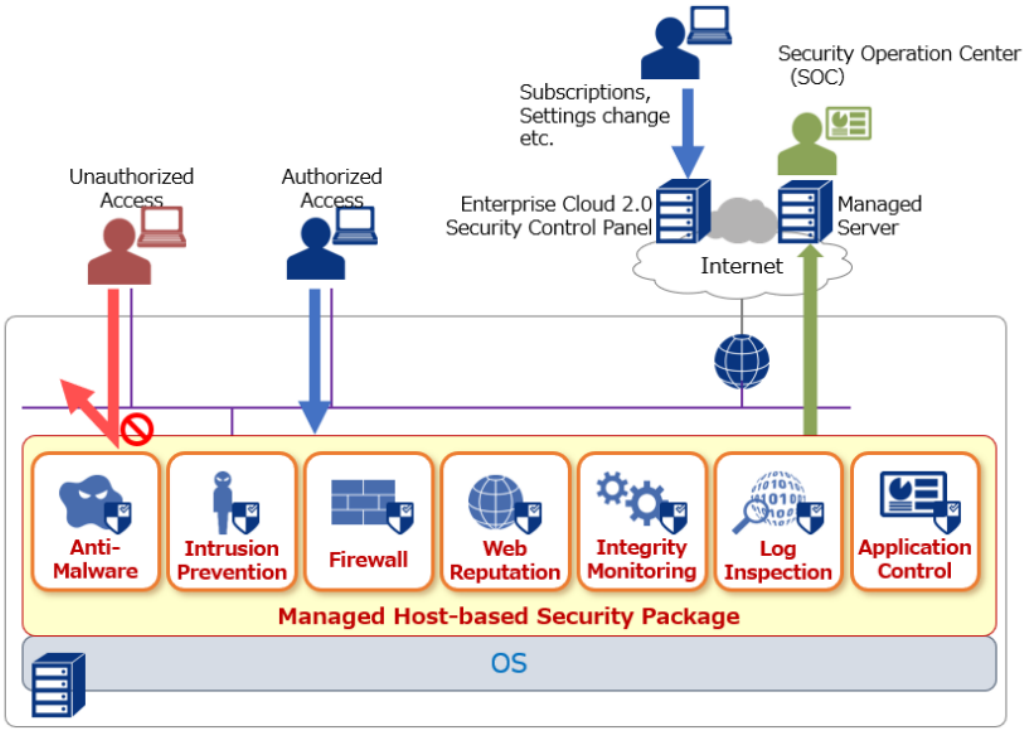
Kao što je opisano, sam WordPress nudi razne mogućnosti podizanja razine sigurnosti, kao i okviri u obliku tema koji se koriste pri izradi. Ipak, veliku ulogu u osiguravanju zaštite ima upravo i poslužitelj.

Bluehost je zbog pouzdanosti i kvalitete popularan poslužitelj koji u sebi sadrži uspostavljene protokole za zaštitu WordPress stranica. Nudi besplatnu sigurnosnu kopiju podataka i jednostavan povratak jednom aktivnih stranica. Također i preko ovog alata, automatizirana je konfiguracija HTTPS enkripcije.

Sigurnost web stranice ima jednako važnu ulogu kao i odabir alata, dizajna i okruženja pri razvoju iste.

U moru informacija i mogućnosti koje razni pomoćni softveri nude, bitno je znati raspoznati uobičajene ranjivosti kao i moguće prevencije napada i rušenja stranice.

Kao programeri, dužni smo klijentima osigurati održivu i funkcionalnu stranicu na što duži vremenski period, a to možemo postići samo ako smo dovoljno upućeni u sigurnosne mjere i potencijalne propuste.



Slika 3.7 Primjer sigurnosnog paketa poslužitelja [16]

Prilikom razvoja stranice pomoću ReactJS-a i TailwindCSS.a, mnogi su nedostatci u prikazu sadržaja koji se vežu za kvalitetu i potpunost sadržaja. Kako su sami okviri u ovom slučaju vrlo opsežni, teško je u kratkom vremenskom roku uzeti u obzir i implementirati sve utjecajne faktore. Stranica Rest&Explore može biti uvelike optimizirana izmjenom sadržaja, ali i gradnjom samog brenda.

Što se WordPress-a tiče najveće ograničenje su striktno definirane mogućnosti koje smanjuju utjecaj prilagođen konkretnim zahtjevima klijenta, podređujući se potrebi za konstantnim ažuriranjem alata na koje sam programer nema utjecaj.

Primjer je konkretna DIVI Premium tema pomoću koje je sučelje za razvoj sadržaja formirano kao predefinirane komponente s ograničenim mogućnostima za izmjenu. Posljedično, kako bi se razvila prilagođena web stranica, potrebno je ipak poznavati osnove kodiranja u JavaScript-u, kao i moguće CSS klase i svojstva za izmjenu dizajna i/ili funkcionalnosti određenih elemenata na stranici.

Problemi koji su se pojavili u izradi ovih stranica početno se vežu na olako pristupanje klijenta zahtjevima koje stranica mora ispunjavati, a zatim iskustvu i poznavanju alata programera koji ih razvija.

Postoji i problem prekomjerne automatizacije alata koji se primjenjuju jer se događa prevelika generalizacija problema koji su realno, individualizirani. Ovo dovodi do manjka mogućnosti prilagodbe i vlastoručnog podešavanja postavki i potreba konkretne stranice. Ipak, razvojem tehnologije, velika većina uobičajenih i neuobičajenih zahtjeva nailazi na rješenja u budućim razvojima alata.

Navedene prepreke ne smiju ostati zanemarene pogotovo jer su obje stranice osmišljene tako da se kontinuirano dodaju i izmjenjuju određene funkcionalnosti i prikazi. S jedne strane, čarter agencija planira rast ponude stoga je važno razmotriti način pohranjivanja podataka, u ovom slučaju ostvaren pomoću kreirane MongoDB baze. Također, u skoroj budućnosti, realan zahtjev klijenta može biti mogućnost rezerviranja termina online kao i eventualno plaćanje predujma ili rezervacije preko forme na stranici. Ovakvi bi se zahtjevi također realizirali u kontekstu ReactJS-a i Node.js-a kao danas već standardizirane funkcionalnosti na webu.

Firma za energetsko projektiranje do sada već ima izvršene brojne projekte prikazane kao reference, ali će u budućnosti mijenjati smjer poslovanja, a pri tom će i broj izvedenih projekata sustavno rasti. Zbog toga je potrebno planirati mogući prostor i raspored kako bi reference ostale naglašene i prikazane na adekvatan način. I u ovom je slučaju moguće zaprimiti nove zahtjeve klijenata u vidu održavanja i izrade *newslettera,* dodavanja novih polja i podataka na formi za prijavu, automatiziranje prikupljanja prijava za posao i slično.

Ipak, ništa od navedenog ne odmiče od uobičajenih mogućnosti web stranice pa je za očekivati da se unutar odabranih okvira isto može i implementirati. Razvoj web stranica je danas uvelike olakšan i ubrzan pomoću raznih dostupnih tečaja i izvora s interneta.

Zaključak

Od samih početaka ekonomije, kupac definira potrebe na tržištu, a u današnjem vremenu ekstremne digitalizacije, jedna od osnovnih potraživanih stavki je aktivni marketing koji se manifestira u obliku web stranice i/ili web aplikacije.

Čest slučaj je situacija u kojoj kupac ne poznaje osnove programiranja, okruženja ili prakse izrade stranice, čime svoje zahtjeve nužno ne prilagođava realnim mogućnostima izvedbe.

Prilikom izrade opisanih projekata, komunikacija s kupcem ima krucijalnu ulogu kako bi konačan cilj bio ispunjen. S druge strane, neovisno radi li se o web aplikaciji ili web stranici, uspješnosti izrade iste uvelike ovisi o standardiziranim alatima korištenim prilikom izrade.

Primjenom različitih pristupa izveden je zaključak o adekvatnoj arhitekturi i okruženju razvoja stranice ovisno o potrebama iste. Ako se radi o isključivo informativnoj stranici ili prezentaciji tvrtke na webu, moguće je da je današnji najbolji izbor WordPress zbog uhodanih praksi praćenja modernih trendova.

Ubrzan razvoj i ponuda aktualne tehnologije potiče programere da aktivno uče i analiziraju moguće optimizacije projekata koje razvijaju. Stoga, današnje klasično obrazovanje ne može pokriti sve mogućnosti dostupne u ovoj domeni te odabir okruženja ovisi o individualnom istraživanju i prilagodbu aktivnih tehnologija.

U konačnici, rezultat krajnje izvedbe isključivo ovisi o zadovoljstvu kupca čiji kriteriji i zahtjevi moraju biti ispunjeni.

Literatura

[1] M. Jazayeri, »Some Trends in Web Application Development,« u *Future of Software Engineering (FOSE '07)*, Minneapolis, MN, USA, 2007.

[2] M. e. a. Curphey, »A guide to building secure web applications,« *The Open Web Application Security Project*, svez. 1, br. 1, 2002.

[3] M. L. Despa, »Comparative study on software development methodologies.,« *Database Systems Journal*, svez. V, pp. 37-54, 2014.

[4] AcqNotes, »Software Development Approaches,« AcqNotes, July 2023. [Mrežno]. [Pokušaj pristupa https://acqnotes.com/acqnote/careerfields/software-development-approaches].

[5] D. Howcroft i J. Carroll, » A proposed methodology for Web development,« 2000.

[6] A. M. French, »Web development life cycle: a new methodology for developing web applications,« *Journal of Internet Banking and Commerce*, svez. 2, 2011.

[7] K. Lei, Y. Ma i Z. Tan, »Performance comparison and evaluation of web development technologies in php, python, and node. js.,« u *2014 IEEE 17th international conference on computational science and engineering*, 2014.

[8] » Usage Statistics of JavaScript as Client-side Programming Language on Websites,« w3techs.com, July 2023. [Mrežno].

[9] Designveloper, »The 5 Best Web App Languages in 2023,« June 2023. [Mrežno]. Available: https://www.designveloper.com/blog/web-app-languages/.

[10] S. Kadi, »Measuring Maintainability and latency of Node. js frameworks,« 2021.

[11] »Resources for Developers by Developers,« MDN Web Docs, [Mrežno]. Available: https://developer.mozilla.org/en-US/.

[12] R. Branas, »AngularJS essentials,« *Packt Publishing*, 2014..

[13] S. Aggarwal, »Modern web-development using reactjs,« *International Journal of Recent Research Aspects*, br. 5.1, pp. 133-137, 2018.

[14] K. Kaan, »10 Best Web Hosting Services,« Forbes Advisor, July 2023. [Mrežno]. Available: https://www.forbes.com/advisor/business/software/best-web-hosting-services/.

[15] »Netlify security overview,« NetlifyDocs, [Mrežno]. Available: https://docs.netlify.com/platform/security/.

[16] WooNinjas, »Top 10 WordPress Security Mistakes (And How To Avoid Them),« [Mrežno]. Available: https://wooninjas.com/wordpress-security-mistakes/.

[17] SakshiBhakhra, geeksforgeeks, [Mrežno]. Available: https://www.geeksforgeeks.org/difference-between-scrum-and-xp/.

[18] K. Ragala, »CMS For Web Development: Types, Features, and Examples,« Knowlegehut, 16 August 2023. [Mrežno]. Available: https://www.knowledgehut.com.

[19] P. Lončar, »Analiza modernih sustava za upravljanje sadržajem na webu,« Rujan 2022. [Mrežno]. Available: https://urn.nsk.hr/urn:nbn:hr:228:864114.

[20] K. B., Moderni sustavi za upravljanje sadržajem, Zagreb, 2019..

[21] D. J. B. P. Savan K.Patel, "Performance Analysis of Content Management Systems-Joomla, Drupal and WordPress," *International Journal of Computer Applications (0975 – 8887)*, vol. Volume 21, no. 4, 2011.

Sažetak

Rad obuhvaća analizu razvojnih faza i tehnologija korištenih pri izradi aktualnih web aplikacija. Podijeljen na dva dijela, teoretski sažima moguća rješenja i pristupe izradi web sadržaja, a praktično prikazuje razvoj dviju web stranica u potpuno različitim okruženjima.

Kako razvoj web stranice, izrada strukture, te mogućnost ostvarivanja funkcionalnosti uvelike ovise o pogodnosti okvira u kojima ih razvijamo, važno je osiguravanje odabirom adekvatnih i održivih alata.

Praktični dio ukazuje na bitnu razliku uloženog vremena i rezultata koja potvrđuje potrebu za konstantnim praćenjem trendova u razvoju web sadržaja.

Ključne riječi: Web aplikacija, WordPress, Okviri za izradu web aplikacija, CMS alati.

Summary

The paper includes an analysis of the development stages and technologies used in the creation of current web applications. Divided into two parts, it theoretically summarizes possible solutions and approaches to creating web content, and practically shows the development of two websites in completely different environments.

Since the development of the website, the creation of the structure, and the possibility of achieving functionality largely depend on the convenience of the framework in which we develop them, it is important to ensure by choosing adequate and sustainable tools. The practical part indicates the significant difference between the time invested and the results, which confirms the need for constant monitoring of trends in the development of web content.

Keywords: Web application, WordPress, Web application frameworks, CMS tools.

Skraćenice

URL	*Uniform Resource Locator*	jedinstveni lokator sadržaja
HTML	*HyperText Markup Language*	hipertekstualni označni jezik
HTTP	*HyperText Transfer Protocol*	protokol prijenosa hiperteksta
IP	*Internet Protocol*	Internetski protokol
CGI	*Common Gateway Interface*	Standardno klijent/server sučelje
SQL	*Structured Query Language*	Strukturirani jezik upita
CSS	*Cascading Style Sheets*	Liste kaskadnih stilova
CRUD	*Create Read Update Delete*	Stvori Čitaj Ažuriraj Obriši
DOM	*Document Object Model*	Model Predmeta Dokumenta
DDoS	*Distributed Denial of Service*	Distribuiran gubitak usluge
SSL	*Secure Socket Layer*	Sloj Sigurnih Ključeva
CDN	*Content Delivery Network*	Mreža za isporuku sadržaja
SEO	*Search Engine Optimization*	Optimizacija pretraživača
DNS	*Domain Name System*	Sustav naziva domene
CMS	*Content Managment System*	Sustav upravljanja sadržajem
ECM	*Enterprise Content Management*	Upravljanje poslovnim sadržajem
WCMS	*Web Content Management System*	Web Sustav za upravljanje sadržajem
LCMS	*Learning Content Management System*	Sustav za upravljanje sadržajem učenja