TailwindCSS preprocesor

Seljan, Lucija

Sveučilište
Sjever

Završni rad br. 829/MM/2023

TailwindCSS preprocessor

Lucija Seljan, 0336046518

Varaždin, lipanj 2023. godine

Sveučilište
Sjever

Odjel za Multimediju

Završni rad br. 829/MM/2023

TailwindCSS preprocessor

Student

Lucija Seljan, 0336046518

Mentor

Vladimir Stanisavljević, mr. sc.

Varaždin, lipanj 2023. godine

Sveučilište Sjever
Sveučilišni centar Varaždin
104. brigade 3, HR–42000 Varaždin

Prijava završnog rada

Definiranje teme završnog rada i povjerenstva

ODJEL Odjel za multimediju

STUDIJ preddiplomski stručni studij Multimedija, oblikovanje i primjena

PRISTUPNIK Lucija Seljan | JMBAG 0336046518

DATUM 28.06.2023. | KOLEGIJ Programski alati 2

NASLOV RADA TailwindCSS preprocesor

NASLOV RADA NA ENGL. JEZIKU TailwindCSS preprocesor

MENTOR Vladimir Stanisavljević | ZVANJE viši predavač

ČLANOVI POVJERENSTVA

1. izv.prof.dr.sc Dean Valdec - predsjednik povjerenstva
2. doc.dr. Andrija Bernik - član povjerenstva
3. mr.sc. Vladimir Stanisavljević, v.pred. - mentor
4. pred. Dražen Crčić, dipl.ing. - zamjenski član
5.

Zadatak završnog rada

BROJ 829/MM/2023

OPIS

CSS pretprocesori danas su jedan od glavnih alata koji programeri koriste kako bi bolje organizirali oblikovanje HTML dokumenata, osigurali bolju kompatibilnost kroz razne verzije internetskih preglednika i skratili vrijeme razvoja korisničkih sučelja kroz ugrađene mogućnosti pretprocesora. Vrlo važan detalj zbog kojeg pri tom koriste pretprocesore su varijable, mixinovi i slično. TailwindCSS je proširenje CSS-a koji omogućuje korištenje stvari kao što su varijable, ugniježđena pravila, inline uvoz i još mnogo toga. Također pomaže u boljem organiziranju koda te omogućuje brže stvaranje i održavanje stilskih listova.
U radu je potrebno:
* opisati osnove CSS-a, njegovu ulogu u oblikovanju Web stranica te analizirati njegove mogućnosti,
* opisati postupak instalacije i osnovnog korištenja TailwindCSS-a, detaljno obraditi mogućnosti i sintaksu preprocesora i na primjerima pokazati njegovo korištenje,
* osmisliti i oblikovati demo web stranicu na kojoj će biti prikazane glavne mogućnosti, načini korištenja TailwindsCSS-a s pripradnim primjerima koda,
* ukratko usporediti TailwindCSS sa sličnim sustavima
Detaljno opisati sve korištene tehnologije i korake u radu potrebne da bi se ostvarilo traženo te detaljno opisati stečena iskustva i postignute rezultate.

ZADATAK URUČEN 26.07.2023 | POTPIS MENTORA Vlados

REPUBLIKA HRVATSKA SVEUČILIŠTE SJEVER

Predgovor

Interes, želja za učenjem i usavršavanjem vještina za kvalitetnog front-end programera prethodili su nastanku ovog završnog rada. U budućnosti se želim posvetiti front-end razvoju i postati jedan od stručnjaka u tome području. Nadam se da će ovaj rad koristiti svima koji započinju putovanje u svijetu programiranja i učenja Tailwind CSS-a.

Također, želim izraziti svoju zahvalnost obitelji i prijateljima koju su me podržavali u svim područjima tijekom studija. Želim posebno zahvaliti mentoru Vladimiru Stanisavljeviću na pomoći koju mi je pružio prilikom izrade ovog završnog rada.

Sažetak

Svrha rada je analiza CSS okvira kod izrade web aplikacija. Rad je podijeljen na dva dijela. Prvi dio rada objašnjava pojam Weba, proces kod izrade korisničkog iskustva i korisničkog sučelja. Također, obrađuje se sam CSS i CSS pre-procesori te se analiziraju popularni CSS okviri. U drugom, praktičnom dijelu rada, bit će izrađena web aplikacija pomoću odabranog CSS okvira. Za izradu web aplikacije koristi se Visual Studio Code razvojno okruženje. Biblioteka pomoću koje je aplikacija programirana zove se React, a glavni koraci tijekom izrade web aplikacije detaljno su prikazani i objašnjeni putem prigodnih slika. Nakon izrade aplikacija će biti prebačena na za to predviđeni internetski server.

Ključne riječi: web, web aplikacija, kroničko iskustvo, korisničko sučelje, CSS, CSS okvir, CSS pre-procesor

Summary

The purpose of this work is to analyse CSS frameworks in the development of web applications. The paper is divided into two parts. The first part explains the concept of the web, the process of creating user experience and user interfaces. It also covers CSS itself and CSS preprocessors and analyses popular CSS frameworks. In the second practical part of the work, a web application will be developed using the chosen CSS framework. Visual Studio Code development environment is used for creating the web application. The library used for programming the application is called React, and the main steps in the development of the web application are thoroughly presented and explained through relevant images. After the development, the application will be deployed to a designated web server.

Keywords: web, web application, user experience, user interface, CSS, CSS framework, CSS preprocessor

Popis korištenih kratica

HTML HyperText Markup Language
Sintaksa za obilježavanje hipertektualnih dokumenata.

CSS Cascading Style Sheets
Sintaksa za oblikovanje hipertekstualnih dokumenta.

SASS Syntactically Awesome Style Sheets
CSS pred-procesor.

LESS Leaner Style Sheets
CSS pred-procesor.

NPM Node Package Manager
Velika kolekcija gotovih JavaScript modula.

FTP File Transfer Protocol
Standardni mrežni protokol koji se koristi za premještanje datoteka s jednog servera na drugi putem mreže temeljene na TCP-u.

HTTP Hyper Text Transfer Protocol
Protokol za prijenos informacija na Webu.

URI Uniform Resource Identifier
Jedinstveni identifikator resursa je niz znakova koji identificira logički ili fizički resurs.

W3C World Wide Web Consortium
Organizacija koja se bavi standardizacijom tehnologija za web.

UX User Experience
Korisničko iskustvo.

UI User Interface
Korisničko sučelje.

SEO Search Engine Optimization
Optimizacija web stranica za tražilice.

Sadržaj

1. Uvod

Internet je sam po sebi brzorastuća mreža koja sa sobom iz dana u dan donosi mnogobrojne inovacije. Pruža neusporedivu količinu mogućnosti rješenja te olakšava i ubrzava protok informacija. Međutim, kada se govori o samom Internetu prva asocijacija su web stranice ili web aplikacije. To je danas jedno od najrazvijenijih područja IT industrije stoga je neophodno da web programeri neprestano nadograđuju svoje znanje i budu u toku s vremenom i novim tehnologijama.

Cilj završnog rada je analiza CSS Tailwind okvira prilikom izrade web aplikacije, navesti neka osnovna svojstva i pružati bolji uvid u njegove mogućnosti i prikazati koliko može ubrzati proces izrade.

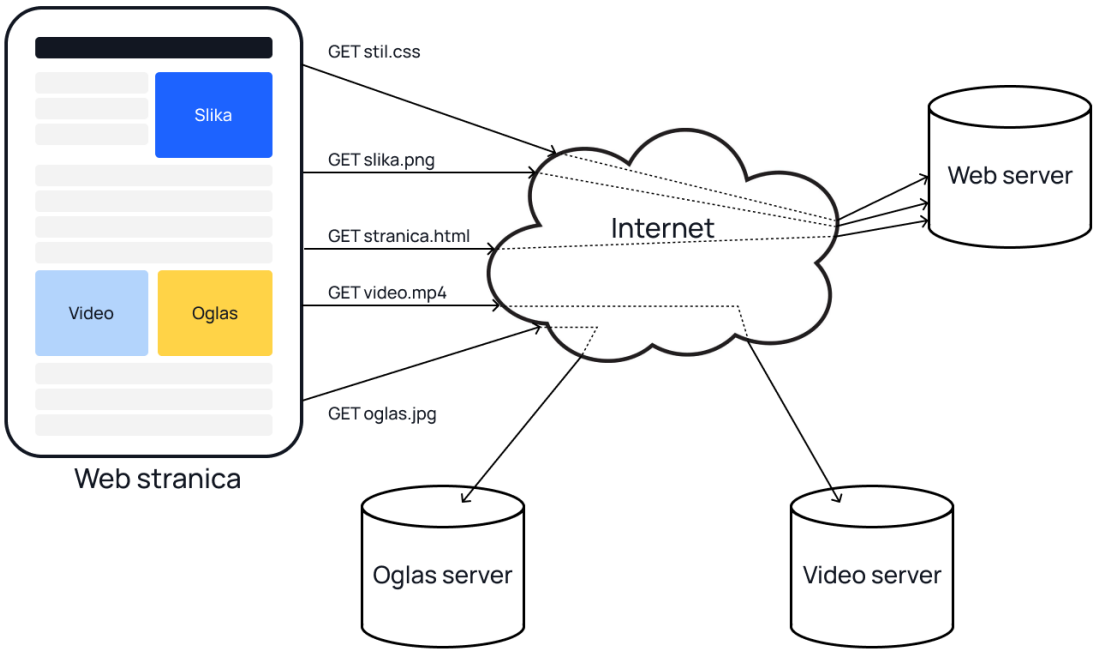
Prvi dio rada će obuhvatiti objašnjenje weba, što su to web preglednici i HTTP protokol, proces izrade korisničkog iskustva i sučelja. Također, analizirati će se CSS i CSS pre-procesori te će biti navedeni i popularni CSS okviri, s naglaskom na Tailwind CSS. Nakon tog dijela slijedi drugi dio rada u kojem će biti prikazan razvoj web aplikacije pomoću odabranog CSS okvira. Osim toga prilikom razvoja aplikacije biti će korišteno za to prikladno razvojno okruženje, a za samo programiranje će biti korištena React biblioteka. Svi koraci, od toga kako instalirati Tailwind CSS i React do korištenja klasa i elementa biti će prikazani pomoću koda i slika. Nakon izrade web aplikacije će biti postavljena za na to predviđeno mjesto, internetski server.

2. Web preglednici i HTTP protokol

Pojma web preglednik postao je dio svakodnevnice, međutim samo nekolicina populacije zna kako web funkcionira i koji su sve protokoli važni kako bi on bio onakav kakvim ga poznajemo. Ovo poglavlje će ukratko objasniti što je to HTTP protokol i kako rade web preglednici.

2.1. HTTP protokol

HTTP (eng. *Hyper-Text Transfer Protocol*) je protokol koji služi da dohvaćanje HTML dokumenata. Osnovna namjena mu je prikaz, objavljivanje HTML dokumenata, odnosno web stranica. Potpuni dokument se rekonstruira iz različitih dohvaćenih pod dokumenata, na primjer, tekst, opis izgleda ili stil, slike, videozapisi, itd. što prikazuje slika 1. [27]



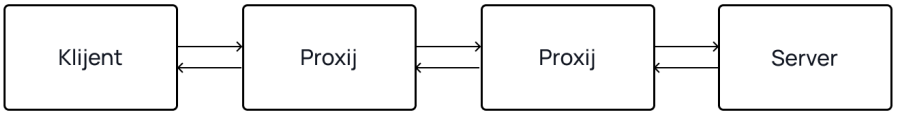
Slika 1. Prikaz HTTP protokola

Smatra se temeljem razmjene podataka na webu i funkcionira na principu klijent-poslužitelj. U nastavku će biti dodatno pojašnjeno što obilježava HTTP protokol.

Klijent-poslužitelj model protokola radi na zahtjev. Klijenti i poslužitelji komuniciraju razmjenom pojedinačnih poruka. Prvo klijent šalje poruke, obično je to web preglednik,

koje se nazivaju (eng. *requests*), a zatim poslužitelj šalje poruke koje se nazivaju odgovori (eng. *responses*).

Između klijenta i poslužitelja nalaze se brojni entiteti, proxiji (eng. *proxies*), koji izvode niz različitih operacija i djeluju kao pristupnici ili pred memorija. Na slici 2. prikazane su osnovne komponente HTTP sustava.



Slika 2. Komponente HTTP sustava

Također, HTTP se zbog svoje proširivosti ne koristi samo za dohvaćanje hipe tekstualnih dokumenata, već se koristi i za slike, videozapise ili objavu sadržaja na poslužiteljima, HTML obrasci. HTTP se isto tako može koristiti za ažuriranje web stranica na zahtjev.

Isto tako, HTTP protokol često zna biti predmet napada jer je on ujedno i ne zaštićen. Kako bi se spriječili takvi napadi stručnjaci su se dogovorili da se podatci koji se šalju putem protokola šifriraju pomoću SSL certifikata (Secure Sockets Layer) ili TLS protokola (Transport Layer Security) te je onda naziv tog protokola HTTPS protokol, a slovo „s“ znači siguran (eng. *Secure*). Razlika između HTTP i HTTPS svodi se i na optimizaciju tražilice. Stranice koje koriste HTTPS protokol imaju bolji SEO (eng. *Search Engine Optimization*) na listi pretrage. Također, ako stranica koristi HTTP protokol šalje se upozorenje korisniku koji zatim mora proći dodatni postupak kako bi pristupio željenoj stanici. [6]

2.2. Rad web preglednika

Korisnici žele da se njihov web sadržaj brzo učitava i s kojim mogu lako stupiti u interakciju. Stoga bi programer trebao nastojati postići ova dva cilja. Da bi razumjeli kako poboljšati izvedbu, potrebno je ponajprije razumjeti kako web preglednici funkcioniraju. U ovom poglavlju biti će obrađene glavne funkcije web preglednika i njegove glavne komponente.

2.2.1. Glavna funkcija web preglednika

Glavna funkcija weba je prezentacija web resurs kojeg korisnik odabere, na zahtjev od poslužitelja i prikaza istog u prozoru preglednika. Resurs je obično HTML dokument, ali može biti i PDF, slika, video ili neka druga vrsta sadržaja. Lokaciju resursa određuje korisnik pomoću jedinstvenog identifikatora izvora, URI (eng. *Uniform Resource Identifier*).[7]

Način na koji preglednik tumači i prikazuje HTML datoteke određen je u HTML i CSS specifikacijama. Sve specifikacije održava organizacija za standardizaciju weba, W3C (eng. *World Wide Web Consortium*). Preglednici su se godinama prilagođavali smo dijelu specifikacija i razvijali su vlastita proširenja. Što je dovodilo do problema s kompatibilnošću za web autore. Međutim, danas većina preglednika odgovara navedenim specifikacijama.[8]

Nadalje, što se tiče korisničkih sučelja web preglednika, svi imaju zajedničke elemente. Neki od tih elemenata su:

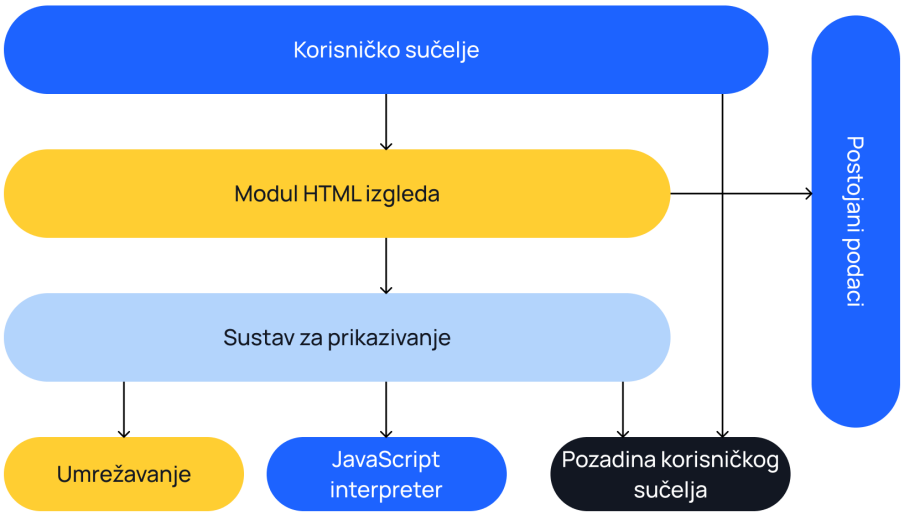
- Adresna traka za URI
- Gumbi za povratak na prethodnu stranicu ili na stranicu unaprijed
- Gumbi za osvježavanje ili zaustavljanje osvježavanja ili zaustavljanje učitavanja trenutnih dokumenata
- Početnu tipku koja vas vodi na početnu stranicu

Također, izgled korisničkog sučelja preglednika nije definirano ni u jednoj formalnoj specifikaciji, ono proizlazi kao rezultat dobre prakse oblikovane tijekom godina iskustva i preglednika koji oponašaju jedni druge. HTML5 specifikacija ne definira elemente korisničkog sučelja koje bi preglednik morao imati, ali navodi neke uobičajene elemente kao što su: adresna, statusna i alatna traka.[7]

2.2.2. Glavne komponente web preglednika

Glavne komponente web preglednika možemo podijeliti u sedam glavnih kategorija. Svaka od njih biti će ukratko pojašnjena u nastavku ovog dijela seminara, a kako bi se što bolje shvatili pojmovi i koja je njihova uloga na kraju će se nalaziti slika 3. koja će prikazati sve komponente web preglednika u njihovom međuodnosu.[28]

1. Korisničko sučelje (eng. *The user interface*): ono se sastoji od trake adrese, gumb za natrag/naprijed, izbornik za označavanje, itd. To je svaki dio prikaza osim prozora u kojem korisnik vidi traženi sadržaj, odnosno web stranicu.
2. Modul HTML izgleda (eng. *The browser engine*): osnova komponenta svakog web preglednika. Glavni zadatak je pretvorba HTML dokumenta i drugih resursa za prikaz na uređaju. Također, bavi se raspodjelom radnji među korisničkim sučeljem i sustavom za prikaz.
3. Sustav za prikazivanje (eng. *The rendering engine*): bavi se prikazom traženog sadržaja. Na primjer, ako je traženi sadržaj HTML dokument, sustav za prikazivanje provodi analizu HTML-a i CSS-a i zatim analizirani sadržaj prikazuje na zaslonu uređaja.
4. Umrežavanje (eng. *Networking*): koristi se kod različitih zahtjeva kao što je HTTP protokol, i za različite implementacije na platforme iza sučelja neovisno o samoj platformi.
5. Pozadina korisničkog sučelja (eng. *UI backend*): koristi se za izradu osnovnih widgeta kao što su kombinirani okviri i prozori. Putem njega se izlaže generičko sučelje koje nije specifično za samu platformu. U pozadini se koriste metode korisničkog sučelja operativnog sustava.
6. JavaScript interpreter (eng. *JavaScript interpreter*): koristi se za izvršavanje JavaScript koda.
7. Pohrana podataka (eng. *Data storage*): sam naziv govori za sebe, radi se o pohrani podatak. Preglednik će ponekad trebati lokalno spremiti određenu vrstu podataka, kao što su kolačići (eng. *cookies*).[8]



Slika 3. Glavne komponente web preglednika

3. Dizajn korisničkog iskustva i sučelja

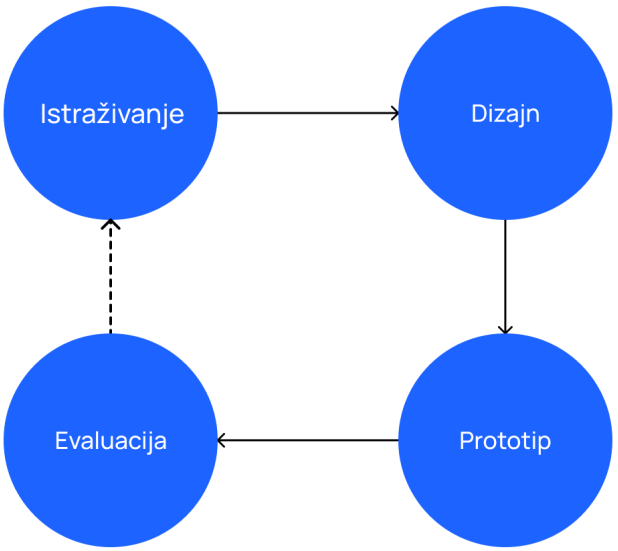
U digitalnom svijetu naglasak se uvelike stavlja na dizajn korisničkog sučelja, jer ono je prvo što privuče korisnika. Međutim, iako je dizajn izniman, ako web aplikacija nije funkcionalna, u smislu korisnik se teško snalazi, ima previše nelogičnosti i otežano mu je korištenje to može negativno utjecati na iskustvo korisnika te postoji mogućnost prestanka korištenja aplikacije.[30]

Cilj je uvijek pružiti vrijednost korisniku, ali i ispunjenje poslovnih ciljeva i na taj način vraćati vrijednosti poslovanju. Ukratko, ako je korisnik postigao svoj cilj, dobio je pozitivno korisničko iskustvo – na brz i jednostavan način pronašao je željenu informaciju. Veće su šanse da se vrati ili preporuči aplikaciju dalje, što uvelike utječe na samu uslugu ili proizvod. Iz tih razloga postoje posebna područja zanimanja koja se bave dizajnom korisničkog iskustva i dizajnom korisničkog sučelja (eng. *UX/UI Design*). Oni uvelike sudjeluju s timom programera tijekom procesa izrade aplikacije. Iz tog razloga, poželjno je i za programera da je upoznat bar s osnovnim načelima dizajna kako bi što lakše mogao implementirati dizajn u funkcionalnu aplikaciju, i tako pridonio stvaranju pozitivnog korisničkog iskustva.[10]

3.1. Dizajn korisničkog iskustva

Dizajn korisničkog iskustva (eng. *User Experience Design*) je usmjeren na stvaranje korisničkog iskustva, bio to proizvod ili usluga, a naglasak se stavlja na rješavanje problema korisnika. Proces dizajna korisničkog iskustva je struktura koja se koristi kako bi dizajner na što lakši i efektivniji način prepoznao problem i pronašao rješenje za isti. UX proces osigurava da cijeli tim zna koja je vizija proizvoda i na taj način osigurava da se potrebe korisnika uzimaju u obzir prilikom svakog koraka.[30]

UX proces uglavnom se sastoji od ova četiri koraka: Prikupljanje informacija – Istraživanje, Dizajn, Prototip, Evaluacija. Bitno je za naglasiti da koraci međusobno mogu ispreplitati, a ne pratiti linearnu formu razvoja. Može se stići do faze izrade prototipa i onda doći do spoznaje da je ideja manjkava i da u suštini ne rješava problem korisnika. To bi bio ukratko proces UX dizajna, a svaki od tih koraka moguće je vidjeti na slici 4. Također, u nastavku će svaki od njih biti pobliže objašnjen.[9]



Slika 4. UX proces

3.1.1. Prikupljanje informacija – Istraživanje

Prva faza UX procesa započinje istraživanjem korisnika. Ona uključuje razgovor ili promatranje korisnika, ciljana skupina ljudi koja bi koristila proizvod ili uslugu, kako bi se shvatili problemi, ali i zahtjevi od rješenja.

Postoje dvije vrste istraživanja korisnika. Prvo je kvalitativno. Ono uključuje misli, osjećaje i mišljenja korisnika. Drugo je kvantitativno. Ono se više fokusira na mjerenje podataka kao što su broj klikova na određeni gumb ili koliko je vremena potrebno za obavljanje određenog zadatka. .[10]

Metode istraživanja korisnika su mnogobrojne i one uključuju intervjue s korisnicima, ankete, broj klika mišem, praćenje pogleda i A/B testove. U fazi istraživanja također se provodi i istraživanje konkurenata kako bi se vidjelo što je već na tržištu.

Potom slijedi analiza dobivenih podataka i koncipiranje rezultata u smislene i djelotvorne cjeline. Zatim je te iste rezultate potrebno predstaviti ostalim dionicima tima.

Prilikom faze istraživanja dizajneri koriste različite alte. Oni ovise o odabranim metodama, ali i potrebama za određeni projekt.

Na temelju dobivenih rezultata istraživanja trebali bi moći definirati glavni problem korisnika koji je potrebno riješiti. Taj dio istraživanja još se naziva i faza definiranja problema u kojoj se obično izrađuje izjava o problemu.

Nakon što je problem jasno definiran, i svi dijelovi istraživanja provedeni dizajneri prelaze na slijedeću fazu UX procesa. .[9]

3.1.2. Dizajn

U ovoj fazi dizajneri već imaju po nekoliko rješenja za definirani problem. Od svih mogućih rješenja odabiru ono koje je najizvedljivije i koje će najvjerojatnije zadovoljiti potrebe korisnika. Zatim se to isto rješenje počinje dizajnirati.

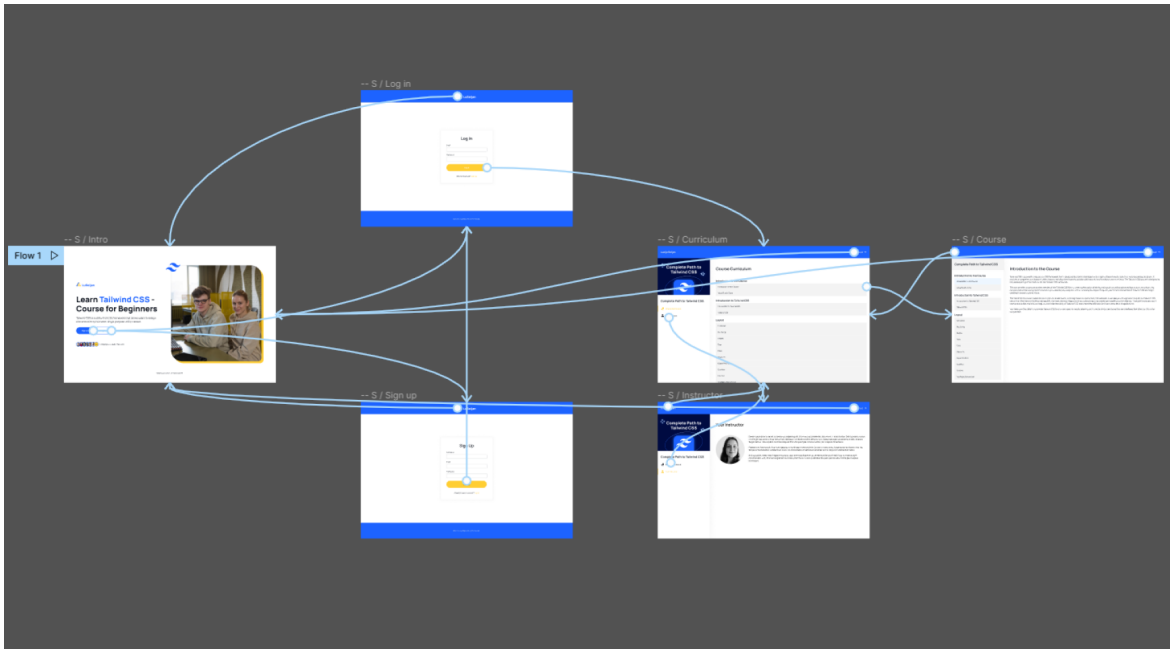
Prvo se izrade skice, početne ideje, uglavnom se rade samo pomoću papira i olovke, jer to je još uvijek brži način ispravljanja i vizualizacije različitih ideja. Nakon što se izrade skice razmotri se i informacijska arhitektura (organizacija sadržaja i informacija) te se mapira tok korisnika.

Iz sve toga proizaći će skice dizajna proizvoda, ili poznatiji naziv žičani okviri (eng. *wireframes*). Skice dizajna proizvoda mogu se izraditi ručno ili digitalno korištenjem raznih alata za izradu okvira. Oni su jako niske vjerojatnosti i ne sadrže boje, fontove, itd. što se može vidjeti na primjeru slike 5. Služe isključivo da bi se dao jasan uvid u to kako će elementi i struktura funkcionirati međusobno unutar korisničkog sučelja, sa svrhom lakšeg proces izrade završnog dizajna. .[9]

Slika 5. Skica dizajna proizvoda

3.1.3. Prototip

Nakon faze skiciranja slijedi izrada prototipa (eng. *prototypes*). Prototip predstavlja simulaciju finalnog proizvoda nastalog u nekom od alata, npr. Figma. Na slici 6. prikazan je prototip izrađen u Figma alatu.



Slika 6. Prototip web aplikacije

Prototip pokazuje kako će korisnici stupati u interakciju i komunicirati s gotovim proizvodom. Interakcija je uvelike važna da se korisnicima stvori dojam kako će ustvari proizvod funkcionirati i omogućuje potpuno doživljaj, jer se unutar interaktivnog prototipa može kliknuti.

Svrha prototipova je dati Vam nešto za testiranje prije finalne izrade proizvoda i na taj način spriječiti mogući gubitak novaca i vremena. Omogućava se sigurnost da je rješenje koje je dizajnirano riješilo problem te da je jednostavno i pristupačno. Također, osigurava da korisnici stupaju u komunikaciju i interakciju s proizvodom na način na koji je tim namjeravao.[9]

3.1.4. Evaluacija

Evaluacija je faza provjere valjanosti, a odnosi se na testiranje dizajna kako bi se utvrdila učinkovitost proizvoda i zadovoljava li on rješenje problema za korisnika. Ovaj korak u procesu zahtjeva UX testiranje na stvarnim ili reprezentativnim korisnicima. Ono

daje priliku identifikacije problema s dizajnom, ali ujedno da se i isprave pogreške prije procesa razvoja, što na kraju štedi vrijeme, novac i nezadovoljstvo korisnika. Ukratko, UX testiranje omogućava dvije opcije, potvrdu valjanosti dizajna ili njegovo poništavanje. Određuje hoće li se ići u fazu razvoja proizvoda ili se vratiti na prethodne korake kao bi se napravile potrebne prilagodbe unutar dizajna.[9]

3.2. Dizajn korisničkog sučelja

Dizajn korisničkog sučelja (eng. *User Interface Design*) je kreativni proces kojim se oblikuje izgled i stil sučelja s ciljem stvaranja jednostavnog i ugodnog korisničkog iskustva. U suštini, korisničko sučelje predstavlja točku interakcije između ljudi i uređaja, te je vrlo bitan aspekt u dizajniranju kvalitetnih proizvoda. Dizajniranje sučelja uključuje različite elemente, kao što su interaktivnost, vizualni dizajn i informacijska arhitektura, kako bi se stvorila intuitivna i funkcionalna sučelja koja olakšavaju navigaciju korisnicima. Tako dizajn korisničkog sučelja pokriva sljedeće aspekte:

- Interaktivnost: tu se razmatraju ponašanja i funkcionalnosti korisničkog sučelja i njegovih elemenata. Primjerice, kako se gumb ponaša kada se klikne na njega, te kako se različiti elementi međusobno povezuju..
- Vizualni dizajn: je također vrlo bitan, jer utječe na dojam korisnika o proizvodu. Boje, tipografija, slike i grafika, logotipi, ikone i razmaci, sve to utječe na dojmljivost sučelja i stvaranje prepoznatljivog vizualnog identiteta proizvoda.
- Informacijska arhitektura: je važna jer se odnosi na organiziranje sadržaja unutar sučelja, kako bi se olakšalo korisnicima pronalaženje i korištenje funkcionalnosti koje su im potrebne.

Dobro dizajnirano korisničko sučelje ključno je za stvaranje kvalitetnog korisničkog iskustva, te zbog toga dizajn sučelja predstavlja jedan od najbitnijih elemenata u razvoju proizvoda. [11]

3.2.1. Proces dizajna korisničkog sučelja

Proces dizajna korisničkog sučelja moguće je podijeliti u pet koraka:

1. Uvažavanje konteksta

Kako bi dizajn korisničkog sučelja za određeni proizvod bio uspješan, dizajneri moraju imati jasno razumijevanje o tome tko će koristiti sučelje i u koju svrhu. Stoga, prvi korak je upoznavanje s ciljnom publikom i svrhom proizvoda.

2. Analiza konkurencije

Analiza konkurencije omogućuje dizajnerima da saznaju što rade drugi u istoj branši te što njihovi korisnici očekuju od proizvoda. Na taj način, dizajneri mogu stvoriti korisničko sučelje koje je prepoznatljivo i intuitivno za korisnike.

3. Dizajn zaslona i UI elemenata

Važan dio procesa dizajna je sam dizajn korisničkog sučelja. U ovoj fazi, dizajneri će stvoriti zaslone, definirati korisničko iskustvo te dizajnirati razne elemente poput ikona, gumba, slika, boja, tipografije, animacija i interakcije.

4. Izrada žičanih modela i prototipova

Dizajneri stvaraju žičane modele, niske vjernosti (eng. *low-fidelity*), kako bi prikazali različite mogućnosti i raspored elemenata na zaslonu. Kako se razvijaju, modeli postaju sve konkretniji, s većom vjernosti (eng. *high-fidelity*), a služe za modeliranje konačnog izgleda i dojma proizvoda te omogućuju uvid u interakciju između korisnika i proizvoda.

5. Razvoj

Nakon što je korisničko sučelje finalizirano, programeri rade na pretvaranju prototipova u stvaran proizvod. Razvoj može biti izazovan jer mogu postojati povratne informacije od drugih sudionika koji zahtijevaju ponavljanje dizajna. [11]

3.2.2. Osnovni principi UI dizajna

Dizajniranje korisničkog sučelja je složen proces koji zahtijeva pažljivo promišljanje kako bi se osiguralo intuitivno i ugodno iskustvo za korisnike. Ovdje su neka od načela koja pomažu dizajnerima u tom procesu:

1. Dosljednost

Konzistentnost je ključna za stvaranje vizualno privlačnog i jednostavnog korisničkog sučelja. Kada korisnici vide iste fontove, boje, gumbe i ikone na različitim dijelovima sučelja, olakšava im se snalaženje i korištenje aplikacije.

2. Predvidljivost

Korisnici imaju očekivanja o tome kako će korisničko sučelje funkcionirati. Stoga je važno da dizajneri stvore sučelje koje je intuitivno za korištenje, a koje odgovara korisničkim potrebama i očekivanjima. Na taj način korisnici će se lakše orijentirati i lakše doći do cilja.

3. Povratne informacije

Davanje povratnih informacija korisnicima tijekom njihovog kretanja kroz korisničko sučelje ključno je za stvaranje ugodnog iskustva. Korisnici moraju biti informirani o tome što se događa kada izvrše određenu radnju. Povratne informacije mogu biti vizualne ili u obliku poruka koje korisnicima pomažu da se kreću po sučelju.

4. Fleksibilnost

Korisničko sučelje mora biti dosljedno, ali istovremeno i fleksibilno kako bi korisnici mogli izvršiti svoje zadatke. To znači da dizajneri moraju stvoriti sučelje koje se može prilagoditi različitim potrebama i željama korisnika.

5. Učinkovitost

Korisnici trebaju biti u mogućnosti brzo naučiti kako koristiti sučelje i izvršavati zadatke učinkovito. Stoga je važno da se novim korisnicima pruže smjernice, dok iskusni korisnici trebaju imati pristup prečicama koje ubrzavaju proces izvršavanja zadataka.

6. Pristupačnost

Korisničko sučelje mora biti dostupno svim korisnicima, uključujući i one s invaliditetom. To zahtijeva stvaranje sučelja koje ima dovoljno kontrasta između teksta i pozadine, jasne gumbe i dodirne točke te stvaranje dizajna koji se prilagođava različitim veličina.[11]

3.2.3. Alati i softveri za dizajn korisničkog sučelja

Dizajneri korisničkog sučelja koriste razne alte i softvere koji im pomažu u stvaranju funkcionalnih i vizualno privlačnih korisničkih sučelja. Neki od najpopularnijih alata današnjice su:

- Sketch: je alat vektorske grafike koji se koristi za izradu žičanog modela, prototipova i dizajna. Ovaj industrijski standardni alat koriste UX i UI dizajneri.[12]
- Adobe Xd: je vektorski alat za dizajn koristan za sve faze procesa, od ranih ideja dizajna pa sve do animacija i interaktivnih prototipova.[13]
- Figma: je brzi i web-bazirani alat za dizajn sučelja. Kao i Sketch i Adobe Xd, Figma nudi uređivač vektorske grafike koji podržava dizajn od početne ideje do interaktivnog prototipa.[14]

4. Osnovno poznavanje CSS-a

CSS ili kaskadna lista stilova (eng. *Cascading Style Sheets*) je stilski jezik koji se koristi za selektivno stiliziranje HTML ili XML dokumenata. CSS opisuje kako se elementi trebaju prikazati na ekranu ili na drugim medijima. Poput HTML-a CSS nije programski jezik, a nije ni jezik za označavanje.[15]

CSS je jedan od temeljnih jezika otvorenog koda i predstavlja standard za sve web preglednike prema specifikacijama koje je navela W3C organizacija. Također, CSS se prethodno razvijao sinkrono, što je pružalo mogućnost verzija najnovijih preporuka. Tako se prva inačica CSS-a počela razvoj 1996., a završila 2008. godine. Nakon toga uslijedile su još dvije verzije CSS2 i CSS3, također trenutno se razvija i CSS4 verzija ili se barem uvelike raspravlja o njezinom razvitku. Međutim, neki tvrde da nakon CSS-a 2.1, opseg specifikacije značajno se povećao da je postalo učinkovitije razvijati i objavljivati preporuke po modulu. Tako CSS moduli sada imaju brojeve verzija ili razine, kao što je CSS Color Modul Level 5. Tako je cilj ovog poglavlja proći osnove CSS-a i njegove mogućnosti.[29]

4.1. Povezivanje s HTML-om

Povezivanje HTML-a i CSS-a može se izvršiti na tri načina:

1. Inline (eng. *Inline*): dodavanje CSS-a unutar HTML-a pomoću atributa style.

```
<h1 style="color:red;">Naslov prve razine i crvene boje</h1>
```

Programski kod 1. Inline dodavanje CSS-a

2. Interno (eng. *Internal*): CSS se dodaje u <head> dijelu dokumenta pomoću elementa <style>

```
<head>
   <style>
      h1 {color: red;}
   </style>
</head>

<body>
   <h1> Naslov prve razine i crvene boje</h1>
```

```
</body>
```

Programski kod 2. Interno dodavanje CSS-a

3. Vanjsko (eng. *External*): CSS se dodaje u <head> dijelu dokumenta pomoću elementa <link>, a datoteka u kojoj su pohranjena CSS svojstva završava ekstenzijom .css

```
<head>
   <link rel="stylesheet" href="stil.css"></style>
</head>

<body>
   <h1>Naslov prve razine i crvene boje</h1>
</body>
```

Vanjska datoteka: stil.css

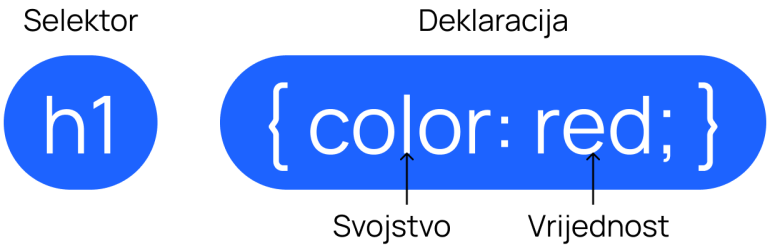
```
h1 {
  color: red;
}
```

Programski kod 3. Vanjsko dodavanje CSS-a

4.2. Sintaksa

HTML i CSS sintakse međusobno se razlikuju. CSS sintaksa sastoji se samo od tri dijela: selektor, svojstvo i vrijednost. Primjer je moguće vidjeti na slici 7.

- Selektor: je oznaka za HTML element na koji želimo primijeniti određeno svojstvo. Svojstva su načini na koje je moguće stilizirati HTML element.
- Deklaracija: se sastoji od svojstva i vrijednosti. Oni su smješteni među vitičastim zagradama, a odvajaju se dvotočkom. Nakon svojstva uvijek se piše točka-zarez kako bi se svojstva međusobno odvojila. [17]



Slika 7. CSS sintaksa

4.3. Selektori

Postoje mnogobrojne vrste selektora. Pomoću selektora odabiremo elemente koje želimo stilizirati. Iako ih ima mnogo mogu se podijeliti u određene kategorije, a ovo su neke od najčešćih:

1. Selektor elementa

Označava sve HTML elemente navedene vrste. Kao primjer mogu se selektirati svi naslovi veličine jedan: h1.

2. Selektor identifikatora ili ID selektor

Odabire id atribut elementa. Svaka id vrijednost treba biti jedinstvena. Selektor ima prefiks # (eng. *hash*). Kao primjer: #id-selektor koji selektira <p id="id-selektor">.

3. Selektor klase ili Class selektor

Odabire atribut klase elementa. Na stranici se može odabrati jedan ili više elemenata. Selektor ima prefiks . (eng. *dot*). Kao primjer: .class-selektor koji selektira <h1 class="class-selektor">.

4. Selektor atributa

Odabiru elemente prema atributu. Selektori su okruženi uglatim zagradama []. Stil se primjenjuje na sve elemente s tim atributom. Kao primjer: img[src] odabire <img src="slika.png">, ali ne i <img>.

5. Selektor pseudo-klase

Odabiru elemente prema pseudo-klasi. Obično se nalaze u kombinaciji sa selektorima elemenata. Kao primjer: a:hover odabire <a>, ali samo kada se pokazivač miša nalazi iznad linka.

6. Globalni selektori

Oni koriste * (eng. *asterisk*). Odabire sve elemente na stranici.

4.4. Fontovi

Svojstva fonta u CSS-u nasljeđuju se s roditeljskih elemenata na djecu. To znači da ako se font ne odredi za neki element, taj element će naslijediti font od svog roditeljskog elementa. CSS svojstvo font koristi se za postavljanje jednog ili više od sljedećih svojstava: font-style, font-weight, font-variant, font-size, line-height i font-family. Ova svojstva određuju oblik, veličinu i stil teksta.

```
font: italic bold normal small 1.2em Tahoma, sans-serif;
```

Programski kod 4. Font

Ovaj primjer će primijeniti sljedeće stilove teksta:

- italic - stil teksta
- bold - podebljanje teksta
- normalna - varijanta teksta
- small - veličina teksta
- 1.2em - visina linije teksta
- Tahoma - primarna vrsta fonta
- sans-serif - rezervna vrsta fonta

Svojstvo font-family omogućava postavljanje niza fontova koje će preglednik koristiti za prikazivanje teksta unutar elementa. Moguće je odabrati specifično ime fonta ili ime generičkog fonta. Kako bi se osigurali jedinstven prikaz web stranice na različitim platformama, potrebno je navesti niz imena fontova. Različite platforme imaju različite ugrađene vrste fontova, stoga je preporučljivo navesti više imena fontova u listi. Na kraju liste, poželjno je dodati i ime generičkog fonta koji će se koristiti za prikaz teksta ako niti jedan od prethodno navedenih fontova nije dostupan na računalu preglednika.

4.5. Model okvira

Model CSS okvira (eng. *Box Model*) se koristi kada je riječ o dizajnu i izgledu. To je okvir koji se nalazi oko svakog HTML elementa na kojeg se mogu primjenjivati različita oblikovanja. Sastoji se od margina (eng. *margin*), obruba (eng. *border*), ispune (eng. *padding*) i sadržaja (eng. *content*).[19]

1. Margina

Označava područje izvan obruba. Služi kako bi se HTML elementi međusobno razmaknuli. Proziran je. Poziva se svojstvom margin i vrijednost joj može biti negativna.

2. Obrub

Obrub je okvir koji obavija sadržaj i ispunu. Moguće mu je odrediti stil, veličinu i boju. Svojstvo pomoću kojeg se poziva je border.

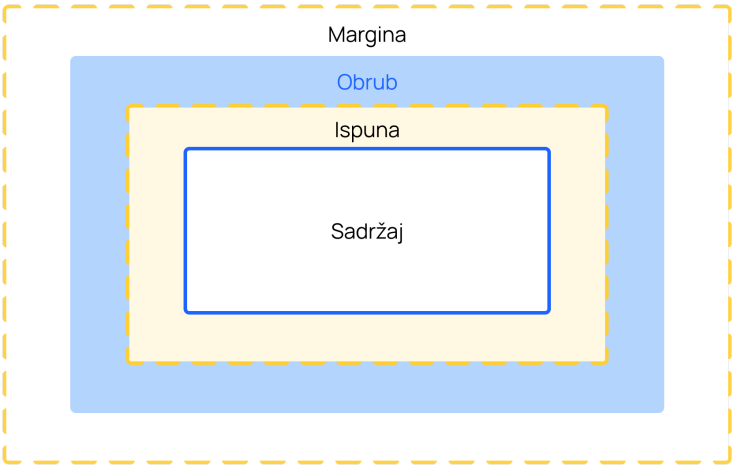
3. Ispuna

Prazno područje oko sadržaja HTML elementa. Proziran je. Mijenja se pomoću svojstva padding, a vrijednost joj ne može biti negativna.

4. Sadržaj

Područje unutar kojeg se pokazuje sadržaj. Sadržaj može biti: tekst, slika, itd. Također, postoji mogućnost oblikovanja sadržaja pomoću dva svojstva: visina (eng. *height*) ili širina (eng. *width*).

Na slici 8. može se vidjeti primjer modela okvira:



Slika 8. Primjer modela okvira

4.6. Fleksibilni raspored okvira

Fleksibilni raspored okvira (eng. *Flexbox*) ili ponekad se još naziva i model fleksibilne kutije, je dizajniran kao jednodimenzionalni model izgleda i kao metoda koja omogućava raspodjelu prostora između stavki u sučelju te poravnanje elemenata. Opisivanjem fleksibilnog rasporeda okvira kao jednodimenzionalnog modela, naglašava se činjenica da se ovaj modul bavi rasporedom elemenata u samo jednoj dimenziji - bilo horizontalno kao redak, ili vertikalno kao stupac.[30]

Fleksibilni raspored okvira je cijeli skup svojstava, a ne samo jedno. Neka od svojstava se postavljaju na tako zvani „roditeljski“ fleksibilni kontejner (eng. *flex container*), dok drugi služe za postavljanje rasporeda njegovih „potomaka“ (eng. *flex-items*), elemenata. Nadalje tijekom rada s fleksibilnim modelom potrebno je razmišljati o dvije osi – glavna os i poprečna os. Glavna os definira se pomoću svojstva flex-direction, a poprečna os se zatim smješta okomito na nju. [20]

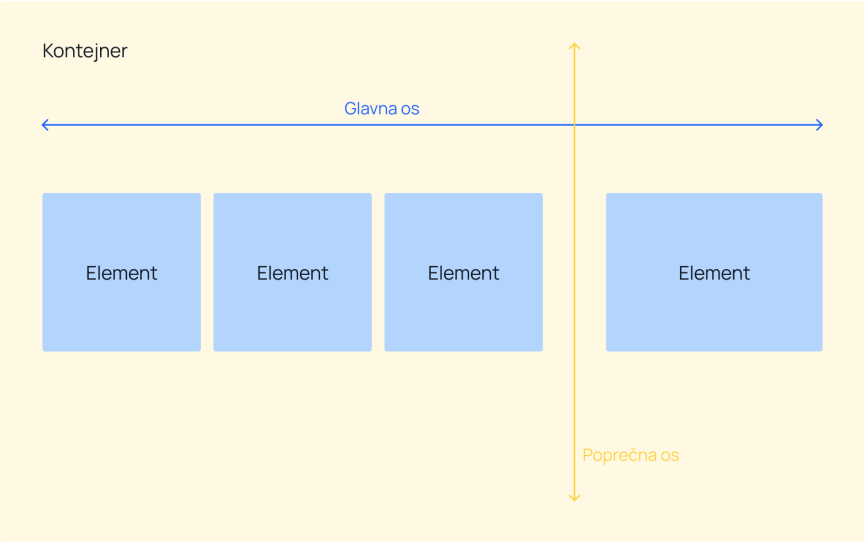
4.6.1. Fleksibilni raspored okvira i vrijednosti roditeljskog kontejnera

1. display: služi za definiranje fleksibilnog kontejnera; inline ili block ovisno o zadanoj vrijednosti
2. flex-direction: služi za postavljanje glavne osi, definirajući smjer u kojem će se poravnati elementi unutar njega. Početna zadana vrijednost je u smjeru retka (eng. *row*) s lijeva na desno. Također, može poprimiti i vrijednosti: column (raspored u stupcima), row-reverse (raspored elemenata u retku, ali obrnutim smjerom) i column-reverse (obrnuti raspored elemenata u stupcu, odozdo prema gore).
3. flex-wrap: služi kao dopuštenje prelamanja stavki po potrebi, u slučaju da sve ne stane u jedan red. Vrijednosti koje može poprimiti su: nowrap (zadano), wrap, wrap-reverse.
4. flex-flow: ovo je skraćenica za flex-direction i flex-flow, koja omogućuje definiranje glavne i poprečne osi spremnika. Zadana vrijednost joj je row nowrap (redak bez prijeloma).
5. justify-content: definira poravnanje po dužini glavne osi. Vrijednosti koje može poprimiti su: flex-start (zadano), flex-end, center, space-between, space-around i space-evenly.

6. align-items: služi za poravnanje sadržaja na poprečnoj osi (os koja je okomita na glavnu os). Vrijednosti koje može poprimiti su: stretch (zadano), flex-start/start/self-start, flex-end/end/self-end, center i baseline.
7. align-content: služi za poravnavanje linija fleksibilnog kontejnera kada unutar njega postoji dodatni prostor na poprečnoj osi. Vrijednosti koje može poprimiti su: normal (zadano), flex-start/start, flex-end/end, center, space-bwtween, space-around, space-evenly i stretch.

4.6.2. Fleksibilni raspored okvira i vrijednosti elemenata unutar kontejnera

1. order: prema zadanim vrijednostima elementi su raspoređeni prema izvornom redoslijedu. Stoga, pomoću svojstva order mogu će promijeniti redoslijed prikaza elementa unutar kontejnera. Vrijednosti se postavljaju u brojkama.
2. flex-gow: definira koju količinu prostora unutar fleksibilnog kontejnera će zauzeti određeni element.
3. flex-shrink: definira mogućnost smanjena određenog elementa ako je potrebno.
4. flex-basis: definira zadanu veličinu elementa prije raspodjele preostalog prostora.
5. flex: je skraćenica za flex-grow, flex-shrink i flex-basis.
6. align-self: omogućava nadjačavanje zadanog poravnanja za pojedinačne elemente unutar fleksibilnog kontejnera. Vrijednosti koje može poprimiti su: auto, flex-start, flex-end, center, baseline i stretch.



Slika 9. Prikaz fleksibilnog okvira

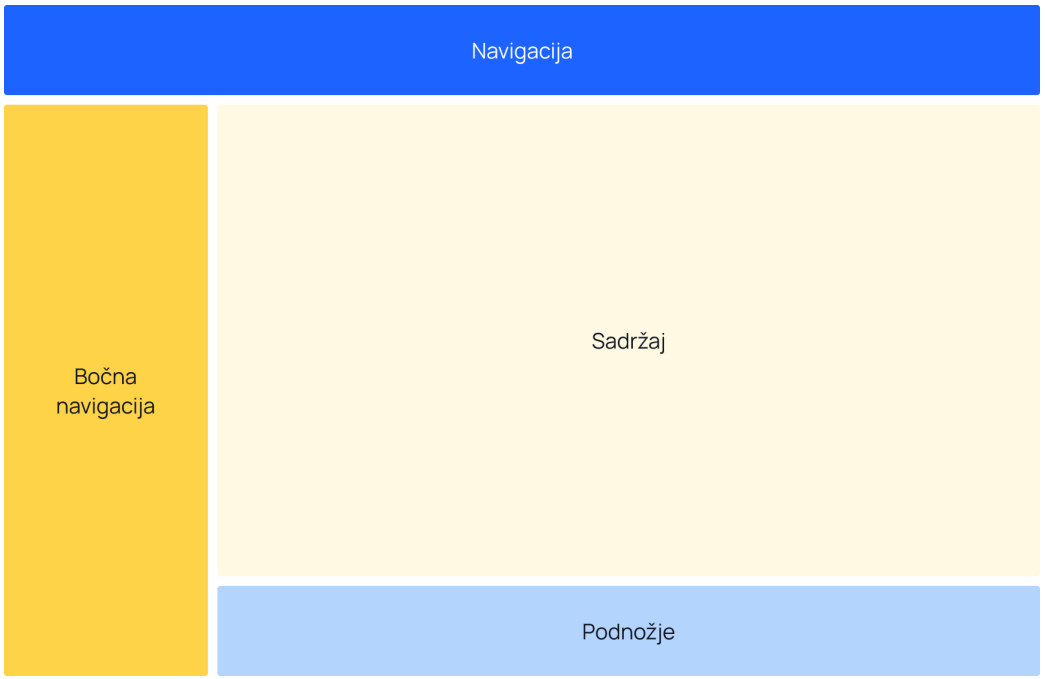
4.7. Rešetka

Rešetka (eng. *Grid*) je inovativni dvodimenzionalni sustav rasporeda temeljen na mreži koji mijenja način na koji se dizajniraju korisnička sučelja u usporedbi s bilo kojim prethodnim sustavom web rasporeda. Iako se CSS oduvijek koristi za oblikovanje web stranica, nikada nije u potpunosti obavljao svoj posao. Rešetka je prvi CSS sustav stvoren posebno za rješavanje problema s rasporedom izgleda web stranice.

Za početak, potrebno je definirati kontejner elementa kao rešetku s display: grid, postaviti veličine stupaca i redaka pomoću grid-template-columns i grid-template-rows, a zatim postaviti njegove podređene elemente u rešetku pomoću grid-column i grid-row. Slično flexboxu, izvorni redoslijed stavki u rešetki nije važan.

Može ih se postaviti u bilo kojem redoslijedu, što omogućuje lako preuređivanje rešetke pomoću medijskih upita. Moguće je definirate izgled cijele web stranice, a zatim ga potpuno preuredite kako bi se prilagodio različitim širinama zaslona.

Rešetka je jedan od najmoćnijih CSS sustava. Na slici 10. prikazan je osnovni raspored web stranice kreiran pomoću rešetke.[4]



Slika 10. Primjer rasporeda web stranice pomoću CSS Grid-a

4.8. Adaptivni dizajn

Medijski upiti su ključna komponenta adaptivnog dizajna koji omogućuje primjenu CSS stilova ovisno o prisutnosti ili vrijednosti karakteristika uređaja. Upiti se najčešće

primjenjuju na temelju veličine okvira za prikaz kako bi se napravili izbori izgleda za uređaje s različitim veličinama zaslona. Na primjer, može se primijeniti manja veličina fonta za uređaje s malim zaslonima, povećati razmak između odlomaka kada se stranica gleda u portretom načinu ili povećati veličinu gumba na zaslonima na dodir.

U CSS-u, koriste se @media pravilo za uvjetnu primjenu dijela lista stilova na temelju rezultata medijskog upita. Također, može se primijeniti uvjetna primjena cijele tablice stilova, pomoću @import.[3]

Za postavljanje veličine okvira za prikaz dodaje se <meta> oznaka svim web stranicama u <head> dio HTML dokumenta:

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

Programski kod 5. Postavljanje veličine okvira za prikaz

5. CSS pre-procesori

CSS pre-procesori su alati koji se koriste za olakšavanje pisanja CSS koda. Oni omogućuju programerima da koriste više napredne funkcije koje nisu dostupne u osnovnom CSS-u, kao što su varijable, funkcije, ugnježđivanje selektora i drugo.

Kada se koristi CSS pre-procesor, programer piše CSS kod u pre-procesorskom jeziku koji se zatim prevodi u običan CSS kod koji se koristi u web stranicama. Najpopularniji CSS pre-procesori su Sass, Less i Stylus.[25]

5.1. Sintaksa

Sass (eng. *Syntactically Awesome Style Sheets*) je jedan od najpopularnijih CSS pre-procesora koji koristi SCSS (Sassy CSS) sintaksu. SCSS je sintaksa koja se vrlo slična standardnoj CSS sintaksi, s dodatkom naprednih značajki poput varijabli, funkcija, ugnježđivanja selektora i slično.

```
$primary-color: #4286f4;

nav {
  background-color: $primary-color;
  ul {
    list-style: none;
    li {
      display: inline-block;
      a {
        color: #fff;
        text-decoration: none;
        &:hover {
          color: darken($primary-color, 10%);
        }
      }
    }
  }
}
```

Programski kod 6. Primjer SCSS koda

Less je još jedan popularan CSS pre-procesor koji koristi svoju sintaksu, sličnu SCSS-u. Jedna od razlika između Less-a i Sass-a je da Less koristi manje znakovne interpunkcije i izražajnije ključne riječi za povećanje čitljivosti koda.

```
@primary-color: #4286f4;

nav {
  background-color: @primary-color;
  ul {
    list-style: none;
    li {
      display: inline-block;
      a {
        color: #fff;
        text-decoration: none;
        &:hover {
          color: darken(@primary-color, 10%);
        }
      }
    }
  }
}
```

Programski kod 7. Primjer Less koda

Stylus je još jedan popularan CSS pre-procesor koji koristi jednostavnu i fleksibilnu sintaksu. Stylus također nudi napredne značajke poput varijabli, funkcija i ugnježđivanja selektora.

Glavna prednost korištenja CSS pre-procesora je olakšavanje pisanja CSS koda i organizacije stilova, što čini održavanje web stranica lakšim i bržim.[26]

5.2. Varijable

Jedna od najkorisnijih značajki CSS pre-procesora su varijable. Varijable omogućuju programerima da definiraju vrijednosti koje se često koriste u CSS-u pre-procesoru, poput boja, fontova i dimenzija, i da ih koriste u cijelom stilskom dokumentu. Korištenje

varijabli olakšava održavanje koda jer se vrijednosti mogu jednostavno promijeniti na jednom mjestu, a da se to odrazi na cijelom dokumentu.[25]

```
$primary-color: #4286f4;
$secondary-color: #d9edff;

body {
  background-color: $secondary-color;
  color: $primary-color;
}

a {
  color: $primary-color;
  &:hover {
    color: darken($primary-color, 10%);
  }
}
```

Programski kod 8. Primjer korištenja varijabli u SCSS-u

U ovom primjeru, varijable $primary-color i $secondary-color se definiraju na početku dokumenta i koriste se kasnije u stilovima za tijelo stranice i veze. Ako se u budućnosti odluči promijeniti primarnu boju, potrebno je promijeniti samo vrijednost $primary-color varijable, a to će automatski primijeniti promjene na svim elementima koji koriste tu varijablu.

Less i Stylus također imaju sličnu sintaksu za definiranje varijabli. Less koristi @ umjesto $ za definiranje varijabli, dok Stylus koristi jednostavnu sintaksu bez posebnog znaka za definiranje varijabli.[25][26]

5.3. Operatori

CSS pre-procesori podržavaju različite vrste operatora, koji se koriste za izvođenje matematičkih operacija, uspoređivanje vrijednosti i druge zadatke u CSS-u.

Jedan od najčešće korištenih operatora u CSS pre-procesorima je aritmetički operator, koji se koristi za izvođenje matematičkih operacija, poput dodavanja, oduzimanja, množenja i dijeljenja. Ovo je korisno kada se definiraju dimenzije, poput visine, širine, margina, i drugih sličnih primjena.

```
$base-font-size: 16px;
$line-height: 1.5;

h1 {
  font-size: $base-font-size * 2;
  line-height: $base-font-size * $line-height;
}

p {
  font-size: $base-font-size;
  margin-bottom: $base-font-size / 2;
}
```

Programski kod 9. Primjer korištenja aritmetičkog operatora u SCSS-u

U ovom primjeru, $base-font-size varijabla se koristi za definiranje osnovne veličine fonta, a $line-height varijabla se koristi za definiranje omjera visine linije i fonta. Te varijable se koriste za izračunavanje vrijednosti fonta, linije i margine, koje se kasnije primjenjuju na naslove i paragrafe.

Osim aritmetičkih operatora, CSS pre-procesori također podržavaju usporedne operatore, poput ==, !=, <, >, <= i >=, logičke operatore, poput i (eng. *and*), ili (eng. *or*) i ne (eng. *not*), te operator ? za uvjetno dodjeljivanje vrijednosti.

Također, bitno je napomenuti da se operatori u različitim CSS pre-procesorima mogu malo razlikovati, a sintaksa može varirati od pre-procesora do pre-procesora.[25][26]

5.4. At-pravila

At-pravila su posebni CSS izrazi koji počinju znakom @, a koriste se za definiranje pravila koja se primjenjuju na cijeli dokument, a ne samo na pojedine elemente. Uz pomoć CSS pre-procesora, at-pravila se mogu koristiti za generiranje koda, uvoz drugih datoteka, postavljanje varijabli, itd.

```
@import 'variables';

body {
  background-color: $background-color;
```

```
  color: $text-color;
}
```

Programski kod 10. Primjer korištenja at-pravila @import u SCSS-u

U ovom primjeru, @import at-pravilo se koristi za uvoz datoteke "variables.scss" koja sadrži definicije varijabli za boje i ostale stilove koji se koriste na stranici. Nakon toga, te varijable se mogu koristiti u ostatku dokumenta.

```
@mixin flex-center {
  display: flex;
  justify-content: center;
  align-items: center;
}

.container {
  @include flex-center;
  width: 100%;
  height: 100%;
}
```

Programski kod 11. Primjer korištenja at-pravila @mixin u SCSS-u

U ovom primjeru, @mixin at-pravilo se koristi za definiranje miksina flex-center, koji se koristi za stiliziranje elementa u sredinu ekrana. Nakon toga, @include at-pravilo se koristi za uključivanje tog miksina u stilsku deklaraciju za .container element.

Ostali česti primjeri at-pravila u CSS pre-procesorima uključuju @extend, @if, @for, @each i @while. .[25][26]

6. CSS okviri

CSS okviri (eng. *CSS frameworks*) su alati koje programeri koriste kako bi im olakšali rad. Umjesto da kreću od nule, okviri omogućuju programerima brzo kreiranje korisničkih sučelja koja se mogu ponavljati i mijenjati tijekom trajanja projekta. Osnovni cilj CSS okvira je standardiziranje prakse web razvoja, što ih čini korisnima za veće projekte, veće timove i one koji trebaju dizajnirati teme koje se mogu koristiti za više projekata.[22]

CSS okvir je skup unaprijed pripremljenih CSS stilova koji su spremni za korištenje. Namijenjeni su za tipične zadatke poput postavljanja navigacijskih traka kako bi se olakšala implementacija na web mjestu. Mogu se nadograđivati pomoću drugih tehnologija, poput JavaScript-a.[1]

CSS okviri ne samo da štede vrijeme, već omogućuju brz i učinkovit rad. Korištenjem standardiziranog okvira, programeri mogu lako razumjeti kod drugih programera umjesto da svaki od njih piše kod od početka za svaki projekt. To poboljšava međusobnu komunikaciju i razumijevanje, što značajno pomaže timovima da postignu ciljeve projekta brže i s manje poteškoća.[2]

6.1. Podjela CSS okvira

CSS okvire možemo podijeliti na dvije glavne kategorije: komponentno orijentirani okviri i okviri koji se temelje na klasama.

Komponentno orijentirani okviri su obično bazirani na gotovim komponentama koje se mogu ponovno koristiti u različitim dijelovima web stranice. To čini razvoj bržim i jednostavnijim, jer programeri ne moraju ponovno stvarati svaku komponentu za svaki dio web stranice. Primjer jednog takvog okvira je Bootstrap, koji dolazi s mnoštvom predložaka i gotovih komponenti poput tipografije, obrazaca, kartica, navigacijskih traka, itd.[22]

Okviri temeljeni na klasama omogućuju programerima da koriste skup klasa za postavljanje različitih stilova i ponašanja na HTML elemente. Programeri mogu odabrati klase koje su im potrebne i primijeniti ih na elemente u svojoj web stranici. To omogućava fleksibilnost i prilagodljivost u dizajnu, jer programeri mogu koristiti različite klase za različite dijelove svoje web stranice. Jedan od primjera ovakvog okvira je Tailwind CSS, koji dolazi s velikim brojem klasa za postavljanje različitih stilova na

elemente poput gumba, tablica, forme, itd. Kako je glavna tema ovog završnog rada Tailwind CSS okvir on će biti detaljnije objašnjen u nastavku poglavlja.[2]

6.2. Tailwind CSS okvir

Tailwind CSS je okvir za CSS koji omogućuje programerima jednostavno stiliziranje web stranica. Razlikujući se od drugih okvira, Tailwind CSS pruža jednostavne klase koje programeri mogu koristiti za brzo izradu prilagođenih dizajna bez nametanja unaprijed definiranih stilova. Ovaj CSS okvir niske razine je izuzetno prilagodljiv i omogućuje programerima potpunu kontrolu nad dizajnom svoje web stranice.

Korištenje Tailwind CSS-a omogućuje programerima stvaranje jedinstvenih korisničkih sučelja kombiniranjem više klasa, bez ograničenja unaprijed definiranim predlošcima ili tehničkim specifikacijama. Za korištenje Tailwind CSS-a, samo je potrebno osigurati CSS datoteku koju će konfiguracijska datoteka obraditi i proizvesti željeni izlaz. Konfiguracijska datoteka omogućuje web dizajnerima da prilagode izgled i dojam svoje web stranice.[21]

Zbog svoje prilagodljivosti i jednostavnosti, Tailwind CSS je postao omiljeni alat među programerima koji žele brzo stvoriti jedinstvena korisnička iskustva. Tailwind CSS je uslužni CSS okvir i predstavlja sjajan alat za sve web programere koji žele brzo i jednostavno stvoriti web stranice s prilagođenim dizajnom.[22]

6.2.1. Instalacija

Tailwind CSS može se instalirati i implementirati na dva načina: putem CDN-a ili putem NPM-a.

1. Putem CDN-a:
    a. Korak 1: Dodajte CDN link u zaglavlje HTML dokumenta

```
<link href="https://cdn.tailwindcss.com" rel="stylesheet">
```

2. Putem NPM-a:
    a. Korak 1: Instalirajte Tailwind putem NPM -a pomoću naredbe:

```
npm install tailwindcss
```

    b. Korak 2: Stvorite datoteku za konfiguraciju pomoću naredbe:

```
npx tailwindcss init
```

c. Korak 3: Konfigurirajte svoj projekt tako da koristite Tailwind CSS. U glavnoj datoteci dodajte sljedeći kod:

```
@tailwind base;
@tailwind components;
@tailwind utilities;
```

d. Korak 4: Kompajlirajte CSS datoteku koristeći sljedeću naredbu:

```
npx tailwindcss build styles.css -o output.css
```

Ova naredba će generirati datoteku "output.css" koja će sadržavati sve stilove definirane u vašoj CSS datoteci.[23]

6.2.2. Fontovi

Fontovi sami po sebi igraju ključnu ulogu u dizajnu, jer oni doprinose ukupnom izgledu i osjećaju sadržaja. On utječe na čitljivost teksta, dojam koji ostavlja vizualni identitet i na kraju može imati i emocionalni učinak na korinika.

Stoga kada se koristi Tailwind CSS za stiliziranje web stranice, važno je uzeti u obzir nekoliko ključnih čimbenika koji utječu na fontove. Neki od njih su:

1. Veličina fonta

Tailwind CSS ima predodređene veličine fontova koje se mogu koristiti putem klasa poput .text-xs, .text-sm, .text-base, .text-lg i .text-xl. Važno je odabrati veličinu fonta koja je primjerena za sadržaj koji prikazujete, kao i za veličinu zaslona.

2. Debljina fonta

Debljina fonta može se kontrolirati pomoću klasa poput .font-thin, .font-normal, .font-bold i .font-semibold.

3. Boja fonta

Tailwind CSS sadrži mnoge klase boja fontova, kao što su .text-black, .text-white i .text-blue-500. Važno je odabrati boju fonta koja se dobro uklapa u dizajn web stranice i čini tekst lako čitljivim.

4. Poravnanje fonta

Tailwind CSS omogućuje lako poravnavanje teksta pomoću klasa poput .text-left, .text-center i .text-right. Preporuka je da se font u većini slučajeva poravna lijevo, pogotovo na manji ekranima radi bolje čitljivost.

5. Uvećanje fonta

Tailwind CSS omogućuje jednostavno povećanje fonta pomoću klasa poput .text-lg, .text-xl i .text-2xl. Bitno je upotrebljavati ove klase umjereno kako bi se spriječilo da tekst postane prevelik i nečitljiv.

6.2.3. Boje

Tailwind CSS koristi sistem naziva boja koji omogućuje lako korištenje boja u dizajnu web stranice. Ključne značajke uključuju:

- Predefinirani nazivi boja koji pokrivaju sve osnovne boje.
- Mogućnost definiranja prilagođenih boja korištenjem HEX, RGB ili HSL vrijednosti.
- Ugrađenu podršku za varijacije boja, što omogućuje brzo generiranje svijetlih i tamnih verzija boja.
- Mogućnost korištenja tema boja za brzo postavljanje paleta boja za cijelu web stranicu.

Ovo su neki od primjera korištenja za rad s bojama:

- bg-{color} - postavlja pozadinsku boju elementa na navedenu boju
- text-{color} - postavlja boju teksta elementa na navedenu boju
- border-{color} - postavlja boju granice elementa na navedenu boju
- hover:bg-{color} - postavlja pozadinsku boju elementa na navedenu boju kada se preko nje pređe mišem
- bg-opacity-{value} - postavlja prozirnost pozadinske boje elementa na navedenu vrijednost
- from-{color} i to-{color} - definiraju gradijent od jedne boje do druge

6.2.4. Ispuna, obrub i margina

Korištenje ispravnih vrijednosti za ispunu, obrub i marginu ključno je za postizanje željenog izgleda i rasporeda elemenata na web stranici. U Tailwind CSS-u, ove značajke se lako kontroliraju koristeći brojne klase.

Za postavljanje ispune u Tailwind CSS-u, važno je zapamtiti da postoji niz klasa koje omogućuju precizno kontroliranje ispune u različitim smjerovima (gore, dolje, lijevo, desno) i u različitim veličinama. Također, dostupne su i klasne skupine koje omogućuju dodavanje i oduzimanje ispune na različitim razlučivostima zaslona (npr. md:pt-4: služi za dodavanje 4 točke ispune samo za veličinu zaslona medium i veću).

Kod postavljanja obruba u Tailwind CSS-u, postoji niz klasa koje omogućuju kontrolu debljine, stila i boje obruba. Također je moguće koristiti klasu „rounded“ za zaobljavanje rubova elementa i „border-0“ za uklanjanje obruba.

Prilikom postavljanja margine u Tailwind CSS-u, postoje klase koje omogućuju precizno kontroliranje margine u različitim smjerovima i veličinama. Isto kao i kod ispune, dostupne su i klase koje omogućuju dodavanje i oduzimanje margine na različitim veličinama zaslona.

6.2.5. Fleksibilni raspored okvira i rešetka

Fleksibilni raspored okvira i rešetka su popularni CSS modeli za koji omogućavaju precizno pozicioniranje i raspoređivanje elemenata na web stranici. Tako Tailwind CSS ima klase koje omogućavaju rad s tim modelima.

Za fleksibilan raspored okvira Tailwind CSS uključuje ove klase:

- flex - omogućuje korištenje fleksibilnog rasporeda okvira
- flex-row i flex-col - postavlja smjer rasporeda elemenata u redove ili stupce
- justify-{alignment} - postavlja horizontalnu poziciju elemenata (npr. justify-center za centriranje)
- items-{alignment} - postavlja vertikalnu poziciju elemenata (npr. items-center za centriranje)
- flex-wrap i flex-nowrap - omogućuje ili sprječava prelamanje elemenata na novi red
- flex-1 - dodjeljuje specifičnu vrijednost fleksibilnosti elemenata

Za rešetku Tailwind CSS koristi slijedeće klase:

- grid - omogućuje korištenje rešetke
- grid-cols-{number} - definira broj stupaca
- grid-rows-{number} - definira broj redaka
- col-span-{number} - definira koliko stupaca će element zauzeti u rešetki
- row-span-{number} - definira koliko redaka će element zauzeti u rešetki
- gap-{size} - postavlja veličinu praznine između elemenata

6.2.6. Adaptivni dizajn

Adaptivni dizajn je ključan za modernu web stranicu kako bi se osiguralo da se sadržaj pravilno prikazuje na različitim uređajima i veličinama ekrana. Tailwind CSS ima mnogo klasa koje se koriste kod izrade adaptivne web stranice. Također, Tailwind CSS ima ugrađene osnovne točke prekida koje se mogu koristiti za prilagodbu sadržaja različitim veličinama ekrana. Ove točke mogu se koristiti kao prefiksi za klase kako bi se definirala veličina ekrana na kojoj će se primijeniti određeni stil.

Ovo su osnovne točke prekida, navedene od najmanje do najveće veličine ekrana:

- sm: „small“ – veličina: 640px - @media (min-width: 640px) { ... }
- md: „medium“ – veličina: 768px - @media (min-width: 768px) { ... }
- lg: „large“ – veličina: 1024px - @media (min-width: 1024px) { ... }
- xl: „extra large“ – veličina: 1280px - @media (min-width: 1280px) { ... }
- 2xl: „double extra large“ – veličina: 1536px - @media (min-width: 1536px) { ... }

Za primjer možemo uzeti klasu text-xl koja će se primijenit na stilizaciju teksta veličine xl na svim veličinama ekrana, dok će klasa sm:text-xl primijeniti stilizaciju teksta veličine xl samo na „small“ ekrane i veće.

Iako su točke prekida u Tailwind CSS-u unaprijed zadane moguće ih je prilagoditi prema potrebi. To olakšava kreiranje prilagođenih točaka prekida za specifične potrebe određene web stranice. Primjer takve prakse biti će prikazan u nastavku rada.

6.3. Usporedba pre-procesora (Sass) i okvira (Tailwind CSS)

	Sass	Tailwind CSS
Definicija	Pruža mogućnost izrade prilagođenih stilova	Pruža gotove klase koje se mogu kombinirati
Veličina	Veći i složeniji	Manji i jednostavniji
Struktura	Koristi strukturu datoteka i hijerarhiju	Koristi oblikovani pristup s mnogo klasa
Stilizacija	Samostalna stilizacija komponenti	Korištenje klasa za primjenu stilova
Prilagodba	Velika mogućnost prilagodbe i izmjene	Manja prilagodljivost, ali brza izrada prototipova
Složenost	Potrebno je više vremena za postavljanje	Brže postavljenje zbog gotovih klasa
Potrebno znanje i vještine	Potrebno je napredno poznavanje CSS-a	Potrebno je osnovno poznavanje CSS-a i razumijevanje klasa
Popularnost	Dugo korišten i popularan	Kraće je korišteno, ali brzo je stekao popularnost, posebno u React zajednici

Prema tablici moglo bi se zaključiti da je Tailwind CSS bolji u nekim stavkama od Sass-a, a prema njima neki od razloga zašto bi koristili Tailwind CSS su sljedeći:

1. Brza izrada

   Tailwind CSS pruža gotove klase koje se brzo primjenjuju na elemente bez potrebe za pisanjem ili konfiguracijom stilova. Što uvelike ubrzava sam proces izrade aplikacije.

2. Dosljednost

Tailwind CSS koristi unaprijed definiran stil klasa što rezultira dosljednim izgledom, ali i ponašanjem elemenata na različitim dijelovima aplikacije. Također, pruža mogućnost kombinacije različitih klasa kako bi se postigao željeni izgled.

3. Održavanje

   Jednostavan je prilikom održavanja. Kako se koriste klase za stiliziranje mogu se pronaći specifični stilovi koji se primjenjuju na elemente. Ovo omogućuje brzi pronalazak i izmjenu bez pojedinačnog pronalaženja pravila i njihovih izmjena.

4. Prilagodljivost

   Iako ima unaprijed definirane klase Tailwind CSS omogućuje prilagodbu klasa putem konfiguracijske datoteke. Unutar nje moguće je prilagoditi boje, razmake, tipografiju i ostale stilove kako bi odgovarali vašem projektu. Također, nudi mogućnosti različitih pluginova za još veću prilagodbu.

5. Gotove komponente

   Osim što ima mogućnost prilagodbe stila Tailwind CSS pruža i neke od gotovi komponenti koje može samo implementirati unutar vlastitog koda.

Iz ovoga možemo zaključiti da odabir tehnologije uvelike ovisi o potrebama projekta, osobnim preferencijama i razini iskustva programera. Tako će se u nastavku rada prikazati proces razvoja jednostavne web aplikacije stilizirane pomoću Tailwind CSS -a.

7. Razvoj web aplikacije

U ovom dijelu će te saznati sve detalje o procesu razvoja web aplikacije pomoću Tailwind CSS okvira. Tailwind CSS je odabran kao glavni okvir za razvoj aplikacije jer, kao što je već spomenuto u prethodnom poglavlju, pruža programeru fleksibilnost u kreiranju vlastitih komponenti koje su potrebne za projekt, umjesto da koristi već gotove komponente koje ne bi mogle u potpunosti zadovoljiti sve potrebe projekta. Korištene tehnologije uključuju HTML, CSS, JavaScript/React, Firebase Auth, FileZilla i dodatne pakete koji će biti detaljno opisani u nastavku poglavlja.

7.1. Opis web aplikacije

Naziv web aplikacije je "Complete Path to Tailwind CSS". Ova aplikacija služi kao vodič (eng. *tutorial*) koji će olakšati početnicima učenje Tailwind CSS okvira. Web aplikacija nudi razne funkcionalnosti i informacije, uključujući:

1. Registracija

   S obzirom na to da se koristi Firebase Auth servis, postupak registracije novih korisnika se odvija isključivo putem tog servisa. Nakon što korisnik unese tražene podatke i pritisne gumb „Sign Up“, njegovi podaci se verificiraju te se šalju Firebase Auth servisu. Taj servis potom kreira novog korisnika koristeći prethodno poslane podatke, a nakon uspješnog stvaranja korisnika, servis šalje obavijest o uspješnosti natrag.

2. Prijava

   Za prijavu se također koristi Firebase Auth servis. Nakon što se korisnik registrira, sljedeći korak je prijava, koju može izvršiti tako što unese svoje podatke - u ovom slučaju, e-mail adresu i lozinku. Nakon što korisnik klikne na „Log in“ gumb, servis provjerava postojanje korisničkih podataka u bazi. Ako korisnik nije pronađen u bazi, sustav prikazuje obavijest i nudi mu mogućnost registracije. No, ako korisnik postoji, servis mu omogućava pristup materijalima i vodičima kako bi mogao početi učiti..

3. Informacije o vodiču

   Nakon uspješne prijave u sustav, otvara se posebna stranica s navigacijskom trakom koja uključuje gumb za odjavu i naziv korisnika. Osim toga, dostupna je i

bočna navigacija za jednostavnu navigaciju između te stranice i informacijske stranice o autoru. Na samoj stranici možete pronaći sadržaj vodiča.

4. Informacije o instruktoru/autoru

Osim početne stranice, postoji i zasebna stranica koja uključuje sve komponente osim glavnog dijela sadržaja. Glavna razlika između ove stranice i početne stranice je u tome što ovdje nije prikazan sadržaj vodiča, već se umjesto toga pružaju osnovne informacije o autoru i projektu.

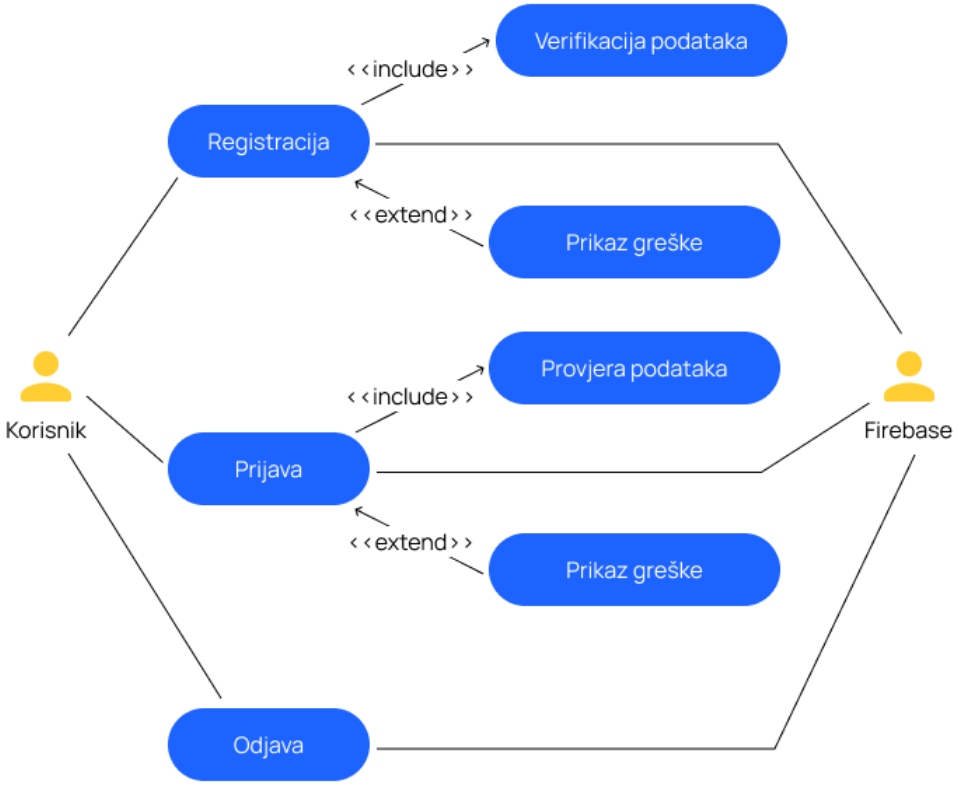
5. Vodič o Tailwind CSS-u

Kada se koristi vodič za Tailwind CSS, primijetit će se promjene u navigacijskoj traci. Umjesto naziva korisnika, pojavljuje se kućica koja korisnika vodi na početnu stranicu vodiča, dok gumb za odjavu ostaje na istom mjestu. Također, bočna navigacija sada vodi korisnika kroz dijelove vodiča, a ne nudi mogućnost navigacije između početne stranice i stranice o autoru. Sadržaj se prikazuje na temelju poglavlja na kojem se korisnik trenutno nalazi.

7.2. Dijagram slučaja upotrebe

Za određivanje potreba i zahtjeva web aplikacije za različite namjene slučajeva, proces se služi dijagramom slučaja upotrebe (eng. *Use case diagram*). Svaki slučaj upotrebe predstavlja cjelovitu aktivnost koju možemo opisati niz radnji i događaja.

U prethodnom poglavlju opisana web aplikacija nije opširna pa smo koristili dijagram slučaja korištenja za cijelu aplikaciju. U dijagramu su prikazani jednostavni scenariji registracije te uspješne i neuspješne slučajeve prijave koji su već opisani. U dijagramu slučaja upotrebe za ovu aplikaciju postoje dva sudionika: primarni i sekundarni. Korisnik je primarni sudionik, a Firebase Auth je sekundarni sudionik. [24]



Slika 11. 7.2. Prikaz dijagrama slučajeva upotrebe

7.3. Programi korišteni kod izrade aplikacije

Tijekom izrade aplikacije korištena su dva softverska alata - Figma i Visual Studio Code. U daljnjem tekstu, svaki od ovih alata će biti detaljnije opisan i objašnjen.

7.3.1. Figma

Figma je suvremeni alat za dizajniranje koji omogućuje korisnicima da stvore, dijele i surađuju na različitim vrstama digitalnih dizajna, uključujući grafički dizajn, interakcijski dizajn, prototipove i još mnogo toga. Figma nudi niz korisnih značajki kao što su jednostavno upravljanje resursima, automatsko generiranje koda i lako dijeljenje dizajna s drugim korisnicima. Ovaj alat je dostupan kao web aplikacija i kao aplikacija za stolna računala (Windows i MacOS) te podržava integraciju s drugim alatima kao što su Sketch i Adobe Creative Cloud.

Figma je postao vrlo popularan alat među dizajnerima i razvojnim timovima zbog svoje jednostavnosti korištenja, fleksibilnosti i suradničkih mogućnosti baš zbog tih značajki, Figma je korišten prilikom dizajna lo-fi i hi-fi modela web aplikacije. On je osigurao preciznost i efikasnost u procesu dizajniranja.

7.3.2. Visual Studio Code

Visual Studio Code je besplatni programski uređivač razvijen od strane Microsofta. To je visoko prilagodljiv alat za uređivanje koda koji pruža podršku za širok raspon programskih jezika, uključujući JavaScript. Ovaj alat dolazi s mnogim korisnim značajkama poput izvorne integracije s Gitom, automatskim ispravljanjem grešaka i naprednim funkcijama traženja koda.

Visual Studio Code se može koristiti na svim glavnim operativnim sustavima, uključujući Windows, macOS i Linux. Sve u svemu, Visual Studio Code je moćan programski uređivač koji pruža širok raspon funkcija i prilagodljivosti, što ga čini popularnim alatom među programerima širom svijeta.

7.4. Izrada aplikacije

U ovom poglavlju biti će prikazani osnovni procesi instalacije i postavljanja projekta te na kraju i sama izrada. Sve će biti popraćeno slikama koda i primjerima, odnosno rezulatatima.

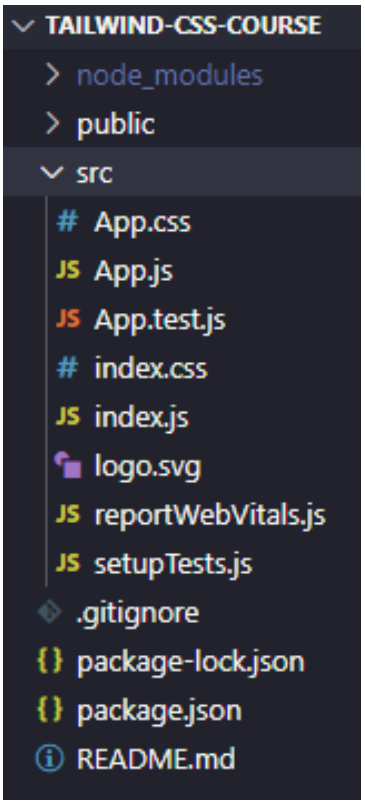
7.4.1. Instalacija React-a i Tailwind CSS-a

Tijekom izrade aplikacije koristi se npm (eng. *Node Package Manager*) kako bi se instalirale početne datoteke. Npm je standard pomoću kojeg se upravlja paketima unutar JavaScripta te uvelike ubrzava proces izrade.

```
npx create-react-app tailwind-css-course
```

Slika 12. Izrada početnih datoteka projekta

Pomoću naredbe iznad izrađena je početna struktura React projekta. Prije nego se počne s izradom projekta potrebno je izbrisati sve nepotrebne datoteke unutar src mape te izraditi dobru strukturu projekta kako se u kasnijem dijelu izrade ne bi desili problemi tijekom stvaranja sve većeg broja datoteka.



Slika 13. Prikaz početne strukture projekta

Poslije postavljanja osnovne strukture projekta vrši se instalacija Tailwind CSS-a te njegova konfiguracija. Instalacija Tailwind CSS-a izvršena je putem npm-a. Također, cijeli proces instalacije detaljno je opisan na Tailwind CSS web stranici pod kategorijom Getting Started -> Installation.

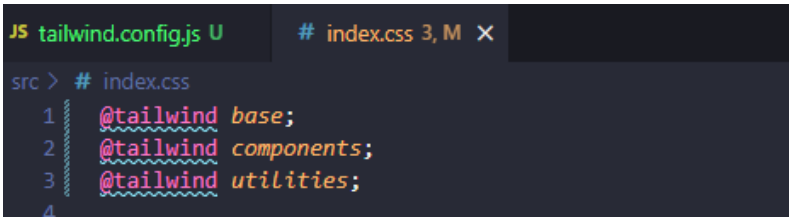
```
PS C:\Users\selja\OneDrive\Desktop\tailwind-css-course> npm install -D tailwindcss
>> npx tailwindcss init
```

Slika 14. Instalacija i konfiguracija Tailwind CSS-a

Konfiguracijska datoteka omogućuje promjenu ili dodavanje Tailwind CSS klasa te pridruživanje ili prilagođavanje vrijednosti. Na slici 15. prikazan je primjer kako izgleda prilagođena konfiguracijska datoteka.

Slika 15. Prikaz konfiguracije datoteke

Također, kako bi Tailwind radio potrebno je još dodati Tailwind direktive unutar glavne CSS datoteke.

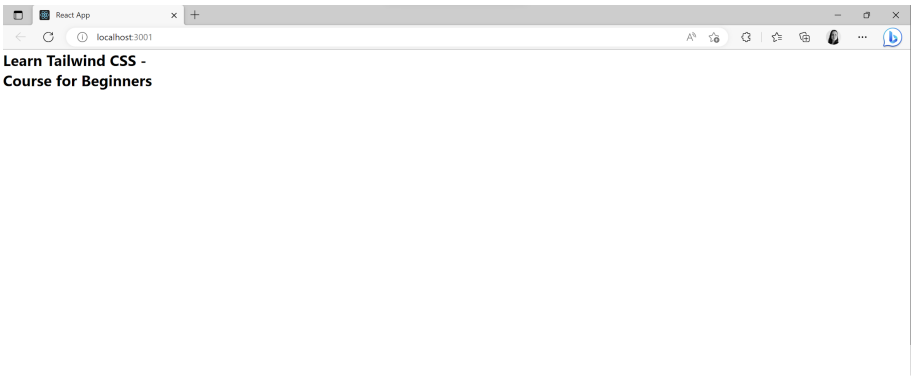


Slika 16. Dodavanje Tailwind direktive u CSS

I onda ostaje još samo početak korištenja Tailwind CSS-a unutar projekta. Na slici 17. prikazan je primjer koda korištenja Tailwind CSS-a, a na slici 18. rezultati unutar preglednika.

```
JS App.js M ×
src > JS App.js > ...
1  import React from "react";
2
3  function App() {
4    return (
5      <div>
6        <h1 className="text-2xl font-bold">Learn Tailwind CSS - <br />
7        Course for Beginners</h1>
8      </div>
9    );
10 }
11
12 export default App;
```

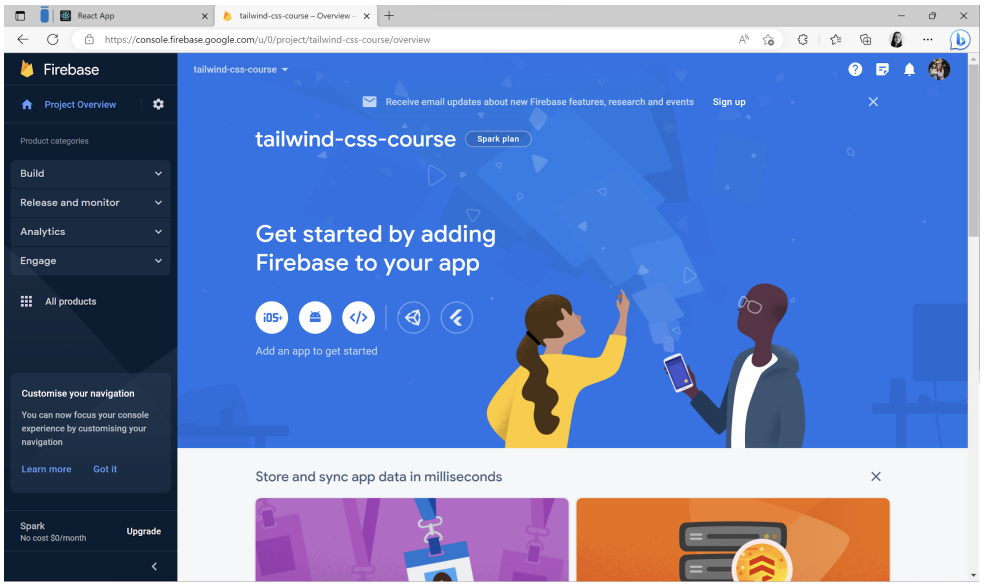
Slika 17. Primjer Tailwind CSS koda



Slika 18. Rezultati primjer korištenja Tailwind CSS koda

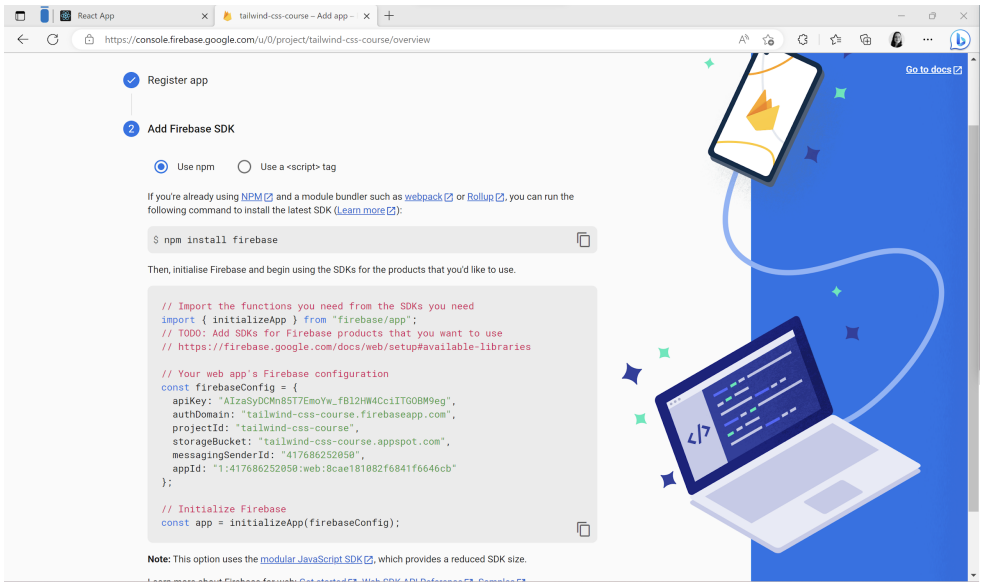
7.4.2. Instalacija i postavljanje Firebase autentifikacije

Prvi korak prilikom postavljanja Firebase autentifikacije je kreiranje naziva projekta. Nakon kreiranog naziva. Koji je za u ovome projektu: tailwind-css-course. I onda nam se prikaže zaslon na slici 19 na kojem odabiremo način implementacije, u some slučaju ikona </>, kao način implementacije za web.



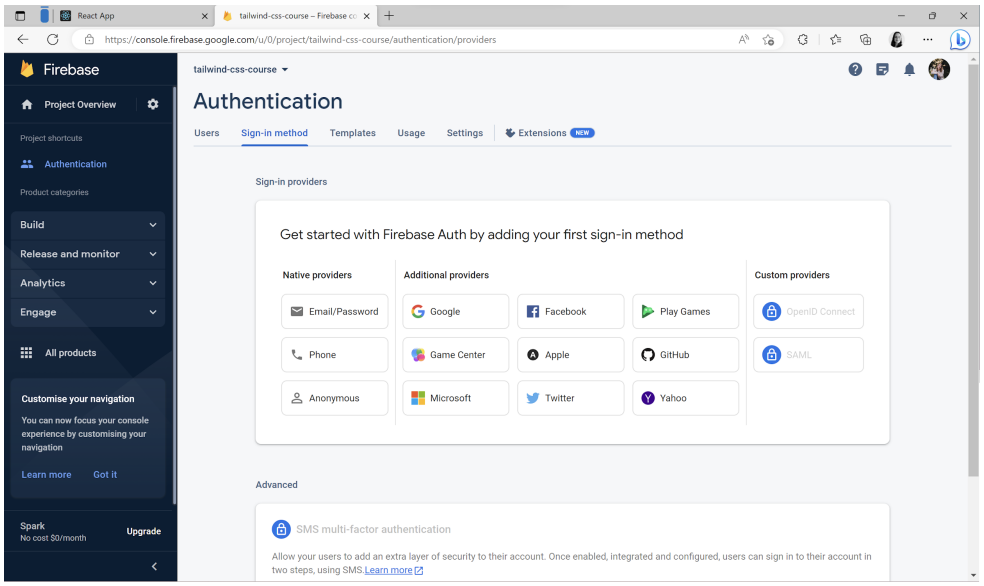
Slika 19. Početni zaslon Firebase autetnifikacije

Nakon registracije aplikacije potrebno je dodati komplet za razvoj softvera, SDK (eng. *Software Development Kit*) kojeg ćemo instalirati pomoću npm-a unutar aplikacije. Također, kako bi Firebase funkcionirao potrebno je dodati funkcije iz SDK unutar same aplikacije, konfiguracijski dio i potom inicijalizirati Firebase, koji su prikazani na slici 20.

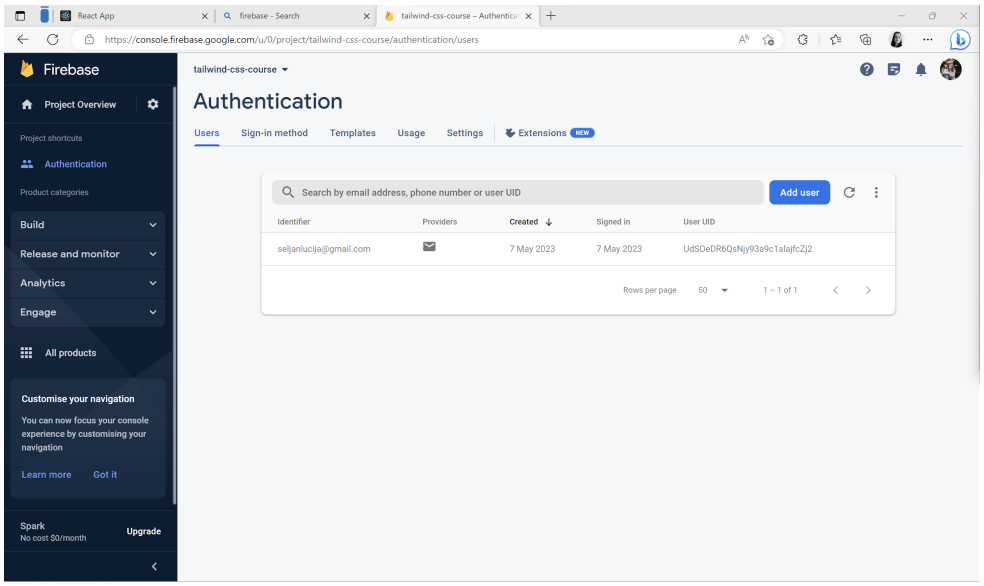


Slika 20. Prikaz kompleta za razvoj softvera

Nakon toga potrebno je odabrati način prijave, u ovome projektu korišten je jednostavan način pomoću e-maila i zaporke. Zaslon odabira moguće je vidjeti na slici 21, a na slici 22 prikazan je uspješno dodan korisnik.



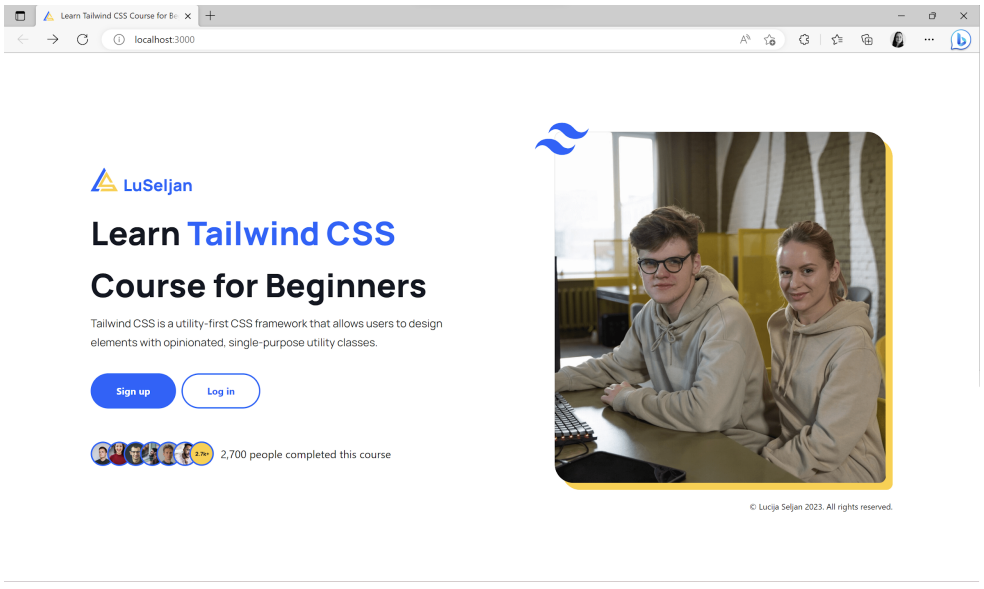
Slika 21. Prikaz odabira načina autentifikacije



Slika 22. Prikaz uspješno dodanog korisnika

7.5. Prikaz dijelova aplikacije

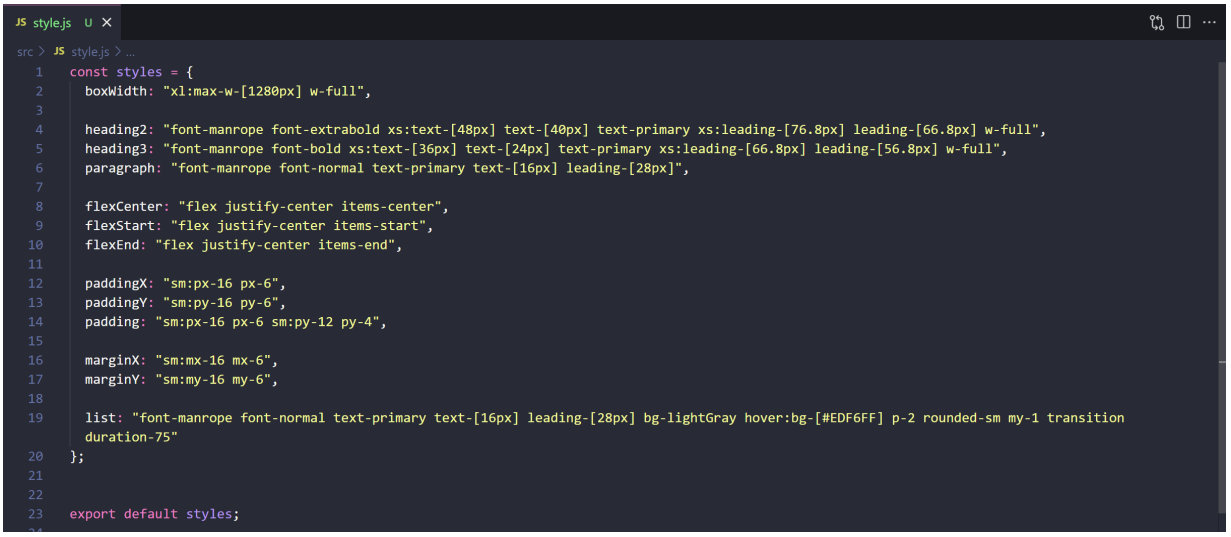
Ovo poglavlje predstaviti će glavne dijelove aplikacije. Tako je na slici 23 prikazana početna (eng. *Home*) stranica koja nudi opciju prijave ili registracije ako ste novi korisnik.



Slika 23. Prikaz početne stranice

Pošto web aplikacija sadrži poveći broj komponenti, biti će prikazani samo neki najvažniji dijelovi aplikacije. Iz tog razloga u nastavku je prikazan kod početne stranice koji će u nastavku biti detaljnije objašnjen.

Na početku datoteke, koda, uvoze se sve potrebite komponente, funkcije i datoteke pomoću naredbe import. Kako bismo mogli koristiti prilagođenu Tailwind CSS datoteku potrebno ju je i uvesti u sam kod, a ona se uvedena odmah prva po redu pod nazivom styles, a dalje se koristi na način da se implementira unutar elementa pomoću svojstva className={`${styles.flexStart}`}, konkretno ova naredba služi za postavljanje fleksibilnog modela prikaza u kojem su elementi u retku, centrirani i započinju od početka. Slika 24 prikazuje style.js datoteku i sva njena oblikovanja.



Slika 24. Prikaz style.js datoteke

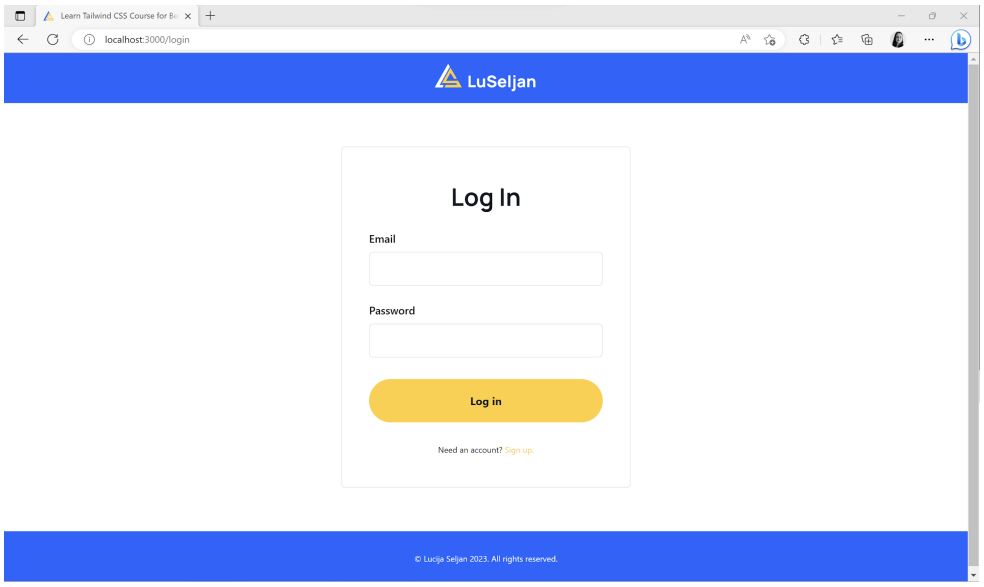
Također, kako bi se stranica prikazivala ne željeni način kako na malim tako i na velikim zaslonima dodano joj je još jedno oblikovanje .boxWidth unutar kojeg je definirano da je na zaslonima većim od 1700px kontejner unutar kojeg se nalazi sadržaj maksimalne širine od 1280px. Svrha toga je da se sadržaj prikazuje u centru stranice i da nije previše raširen na ekranu. Isto tako prilikom mobilnog prikaza sadržaj mijenja svoj raspored te iz reda prelazi u stupce, to se definira pomoću naredbe flex-col, a kako bi se na stolnom prikazu i dalje sadržaj prikazo u stupcima dodana md:flex-row naredba koja nakon točke prekida od 1060px sadržaj vraća u redak.

```
import styles from "../../style";
import { logo, introImage, people } from "../../assets";
import { Link } from 'react-router-dom';

const Home = () => {
     return (
       <div className={`${styles.flexStart} ${styles.paddingX}`}>
         <div className={`${styles.boxWidth}`}>
             <section className={`flex md:flex-row flex-col ${styles.paddingY}
             xs:h-[100vh]`}>
              <div className={`flex-1 ${styles.flexStart} flex-col xl:px-0
              sm:px-16 px-6`}>
               <div className="flex flex-row items-center py-4">
                   <img src={logo} alt="logo" className="h-[42px]" />
               </div>
                    <h2 className={`${styles.heading2}`}>Learn
                     <span className="text-blue">Tailwind CSS</span> <br />
                     Course for Beginners
                    </h2>
               <p className={`${styles.paragraph} mt-1`}>Tailwind CSS is a
               utility-first CSS framework that allows users to design elements with
               opinionated, single-purpose utility classes.</p>
               <div className="flex flex-row items-start my-8">
               <Link to="/signup">

               <button class="border-2 border-blue bg-blue text-white font-bold
               py-3.5 px-9 rounded-full text-sm">
                   Sign up
               </button>
               </Link>
```

```
            <Link to="/login">
            <button class="border-2 border-blue text-blue font-bold py-3.5 px-9
            rounded-full text-sm mx-2">
                Log in
            </button>
            </Link>
            </div>
            <div className="flex flex-row items-center my-4">
                <img src={people} alt="people" className="h-[38px]" />
                 <span className="ml-2.5">2,700 people completed this
                 course</span>
            </div>
          </div>
          <div className={`flex-1 ${styles.flexEnd} flex-col xl:px-0 sm:px-16
           px-6 py-6`}>
            <img src={introImage} alt="Intro image with people" />
            <div className="mt-4 text-xs">
             <p>© Lucija Seljan {new Date().getFullYear()}. All rights
             reserved.</p>
            </div>
          </div>
        </section>
      </div>
    </div>
  )
}

export default Home;
```

Programski kod 12. Prikaz programskog koda Home komponente

Nadalje, slika 25 prikazuje ekran za prijavu ili registraciju korisnika. Na vrhu stranice nalazi se navigacija koja je zauzima punu širinu ekrana te je unutar nje smješten logo koji je ujedno i link na početnu stranicu. Ispod navigacije nalazi se forma za prijavu ili registraciju koja se sastoji od naslova, opisa polja za unos, polja za unos podataka, gumba za prijavu ili registraciju i na kraju se nalazi odjeljak s tekstom i linkom koji nudi mogućnost kretanje između stranica za prijavu i registraciju. I na samom kraju se nalazi podnožje koje sadrži znak za autorska prava (eng. *copyright*), naziv autora i godinu.



Slika 25. Prikaz stranice za prijavu

```
<div className={`${styles.flexCenter} py-4 w-full bg-blue self-start`}>
       <Link to='/'>
         <img src={logoWhite} alt="Logo" className='h-[42px]'/>
       </Link>
</div>
```

Programski kod 13. Primjer stiliziranja div elementa pomoću Tailwind CSS-a

U kodu 13 koristi se oblikuje se div element. Evo objašnjenja Tailwind CSS klasa koje se koriste:

- `${styles.flexCenter}`: Ova klasa se dobiva iz prethodno definirane „styles“ datoteke. Postavlja element na sredinu kontejnera koristeći Flexbox, centrirajući ga vodoravno i okomito.
- „py-4“: Dodjeljuje se ispuna od 4 piksela okomito na gornji i donji dio elementa.
- „w-full“: Postavlja širinu elementa na 100% širine kontejnera.
- „bg-blue“: Postavlja pozadinsku boju elementa na plavu boju.
- „self-start“: Poravna element na vrh kontejnera.

Dalje, unutar ovog div elementa koristi se React komponenta „Link“ iz biblioteke „react-router-dom“ za stvaranje linka koji vodi na putanju „/“. Unutar „Link“ komponente, nalazi se „img“ element koji prikazuje logo. Ovaj „img” element ima postavljenu visinu od 42 piksela (h-[42px]) pomoću Tailwind CSS klase.

Slično oblikovanje koristi se i prilikom oblikovanja podnožja, jedina razlika je što se ne prikazuje logo i link na početnu stranicu nego je prikazan tekst unutar paragraf elementa.

```
<form onSubmit={handleSubmit} className='max-w-[500px] sm:w-[30%] my-16 p-10
border rounded-md self-center'>
       <h3 className={`${styles.heading3} text-center`}>Log In</h3>
         <div className='flex flex-col py-2'>
           <label className='py-2 font-medium'>Email</label>
           <input
             onChange={(e) => setEmail(e.target.value)}
             className='border p-3 rounded-md'
             type='email'
           />
       </div>
       <div className='flex flex-col py-2'>
         <label className='py-2 font-medium'>Password</label>
            <input
              onChange={(e) => setPassword(e.target.value)}
              className='border p-3 rounded-md'
              type='password'
            />
       </div>
       <button className='w-full py-[20px] my-6 rounded-full text-primary font-bold
,      bg-yellow'>
           Log in
       </button>
       <p className='py-2 text-center  text-xs'>
          Need an account?{' '}
         <Link to='/signup' className='text-yellow'>
            Sign up.
         </Link>
       </p>
</form>
```

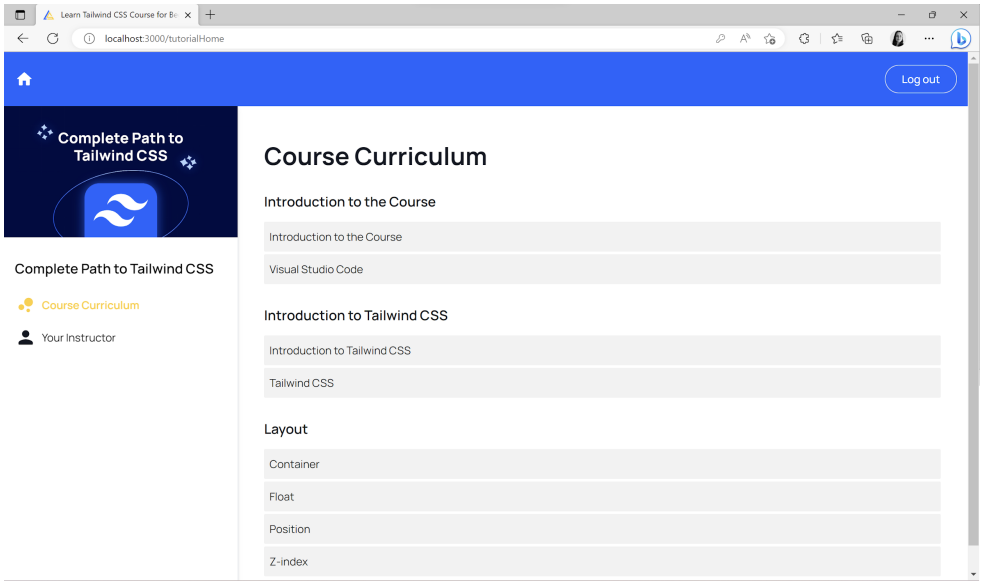
Programski kod 14. Prikaz programskog koda komponente za prijavu

Kod 14 prikazuje primjer kako oblikovati formu za prijavu. U nastavku slijedi objašnjenje dijelova koda klasa koje se koriste i nisu objašnjenje u prijašnjem dijelu rada:

- „max-w-[500px]“: Postavlja maksimalnu širinu forme na 500 piksela.
- „sm:w-[30%]“: Postavlja širinu forme na 30% širine kontejnera za ekrane s veličinom „sm“ (srednje).
- „border“: Dodjeljuje obrub formi.
- „rounded-md“: Daje minimalno zaobljenje rubovima forme.
- „self-center“: Centrira formu unutar kontejnera.
- „${styles.heading3}“: Ova klasa se dobiva iz prethodno definirane styles datoteke. Definira stil naslova trećeg reda.
- „font-bold“: Postavlja podebljanje tekstu.

Također, koriste se i varijante ovih klasa kao što je „rounded-full“ koji u potpunosti zaobljuje rubove elementa (primijenjen na elementu gubma) te se sva dodatna oblikovanje, odnosno klase mogu pronaći na službenoj Tailwind CSS web stranice pod određenim kategorijama.

Na slici 26 prikazan je početni zaslon nakon što se korisnik ulogira u vodič. Vodič se sastoji od dvije fiksno pozicionirane navigacije i sadržaja vodiča. Prva navigacije je plava na vrhu koja je sličnog oblikovanja kao i prethodna, razlika je jedino u tome što su u ovoj elementi međusobno razmaknuti. Sastoji se od ikone kućice koja vodi na početnu stranicu vodiča i gumba za odjavu. Druga navigacija nalazi se s lijeve strane ekranskog prikaza i ona prikazuje naslovnu sliku vodiča i dva linka koji omogućavaju kretnju između početne stranice i stranice o instruktoru. Treći dio, odnosno vodič prikazuje sam sadržaj koji je oblikovan na način da svaki naslov teme zauzima punu širinu prostora, ima svijetlo sivi pozadinu i na prelazak miša boja pozadine poprimi svijetlo plavu boju.



Slika 26. Prikaz početne stranice vodiča

```
<aside class="w-full sm:w-[346px] sm:fixed top-[80px] sm:h-screen border-r
shadow-md">
       <img src={tutorial} alt="Complete Path to Tailwind CSS" className='hidden
       sm:block' />
      <ul className='p-4 mt-[80px] sm:mt-0'>
       <h2 className="font-manrope text-xl font-semibold py-4">Complete Path to
       Tailwind CSS</h2>
             <li className='py-2'>
              <Link to='/tutorialHome' className='flex items-center text-yellow
              font-semibold font-manrope'>
                 <svg>---</svg>
               Course Curriculum
              </Link>
             </li>
             <li className='py-2'>
               <Link to='/instructor' className='flex items-center font-manrope'>
                 <svg>---</svg>
               Your Instructor
               </Link>
             </li>
       </ul>
</aside>
```

Programski kod 15. Prikaz koda bočne navigacije

Programski kod 15 prikazuje bočnu navigaciju na stranici. U nastavku će biti objašnjeni glavni dijelovi i oblikovanja.

- „w-full sm:w-[346px] sm:fixed top-[80px] sm:h-screen border-r shadow-md“: Postavlja širinu elementa na 100% širinu elementa na svim ekranskim prikazima, a za ekrane širine „sm“ (srednje) i veće postavlja fiksnu širinu od 346 piksela. Navigacija je postavljeno fiksno na vrhu stranice, s odstupanjem od vrha od 80 piksela. Također, kako bi se navigacija odvojila od ostatka sadržaja dodan joj je desni obrub i sjena.
- : Prikazuje sliku koja je skrivena na manji ekranima pomoću klase „hidden“, ali je prikazana na ekranima širine „sm“ i većima od toga pomoću klase „block“.

- <ul className='sidebar-nav p-4 mt-[80px] sm:mt-0'>: Element koji definira neuređenu listu „ul“ koja služi za prikaz linkova u navigaciji. Ima unutarnji razmak od 4 jedinice prostora na sve strane i vanjski razmak na vrhu od 80 piksela. Na ekranima širine „sm“ i većima, vanjski razmak na vrhu je nula. Zatim, unutar bočne navigacije se nalaze „li“ elementi koji je postavljen unutarnji vertikalni razmak od 2 jedinice prostora.
- <Link to='/tutorialHome' className='flex items-center text-yellow font-semibold font-manrope'>: Prikazuje link koji ima stilizaciju za fleksibilno poravnanje elemenata unutar linka. Tekst je obojen u žutu boju i polu-podebljan pošto se korisnik nalazi na aktivnoj stranici. Međutim, stranica koja nije aktivna ima standardno oblikovanje tekst tamne boje i normalne debljine.

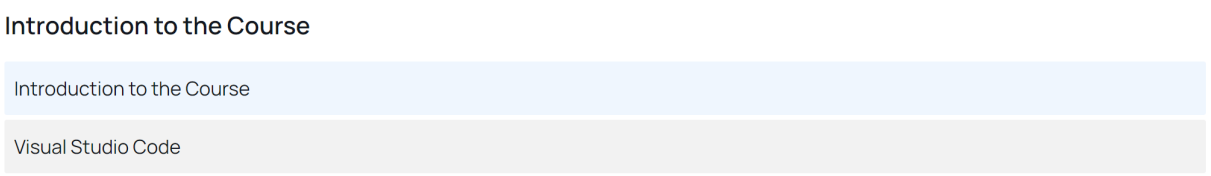
```
<main class="flex-1 sm:ml-[346px] sm:mt-[80px] sm:p-10 px-4 py-10">
      <h3 class={`${styles.heading3} pb-5`}>Course Curriculum</h3>
      <article>
            <h4 className="font-manrope font-semibold text-primary text-[20px]
            leading-[28px] mb-4">Introduction to the Course</h4>
             <ul>
                  <li className={`${styles.list}`}>
                  <Link to="/courseIntro">Introduction to the Course</Link>
                  </li>
                  <li className={`${styles.list}`}>
                  <Link to="/vsc">Visual Studio Code</Link>
                  </li>
             </ul>
      </article>
       <article className='mt-8'>...</article>
</main>
```

Programski kod 16. Prikaz glavnog sadržaja aplikacije

Programski kod 16. prikazuje glavni sadržaj aplikacije, odnosno sadržaj vodiča koji su ustvari linkovi na lekcije. U nastavku će biti objašnjeni elementi i Tailwind CSS klase koje su se koristile da bi se postigao željeni rezultat:

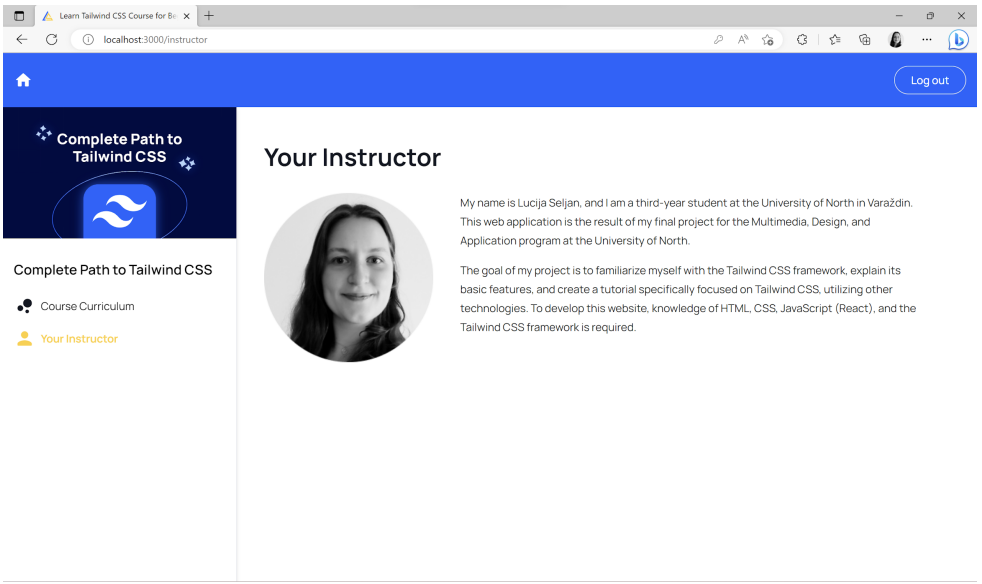
- „sm:ml-[346px] sm:mt-[80px] sm:p-10 px-4 py-10“: Postavlja glavni sadržaj na ekranima širine „sm“ i većima vanjski razmak s lijeve strane na 346 piksela, također postavlja i unutarnji razmak na sve strane od 10 jedinica prostora.

- „<h3 class={${styles.heading3} pb-5}>Course Curriculum</h3>“: Prikazuje naslov treće razine i koristi stilizaciju definiranu unutar „styles.heading3“ i ima unutarnji razmak na donji dio od 5 jedinica prostora.
- Zatim dolazimo do <article> elementa koji definira dio po dio sadržaj vodiča. Unutar njega nalazi se naslov četvrte razine: „<h4 className="font-manrope font-semibold text-primary text-[20px] leading-[28px] mb-4">“, koji koristi stilizaciju polu-podebljanog fonta, veličinu teksta od 20 piksela, primarnu boju teksta, prored od 28 piksela i unutarnji razmak od 4 jedinice prostora na donji dio.
- Unutar „ul“ liste nalaze se „li“ stavka koja sadrži link na drugu stranicu.
    - „<li className={${styles.list}}>“: Stilizacija definirana unutar CSS modula „styles.list“ koja ima sljedeća oblikovanja: definira font „Manrope“, primarnu boju teksta koji je veličine 16 piksela s proredom od 28 piksela, boja pozadine je svijetlo siva, a na prijelaz miša poprima svijetlo plavu boju (#EDF6FF), ima postavljen unutarnji razmak, zaobljene rubove malog radijusa, vanjski razmak i tranziciju od 75 milisekundi.



Slika 27. Prikaz rezultata oblikovanja "${styles.list}"

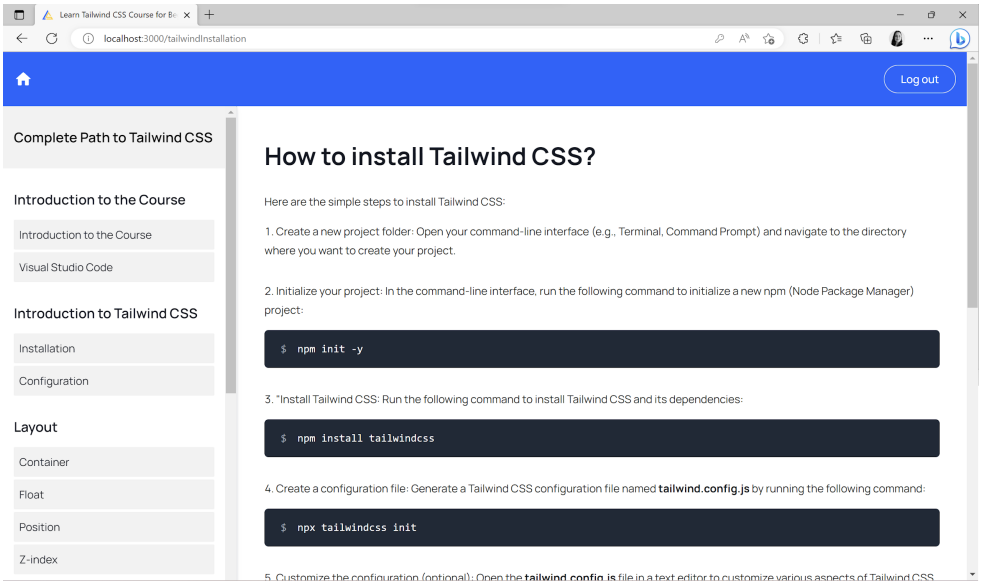
Na slici 28 prikazana je stranica o instruktoru, kreatoru vodiča. Stranica se sastoji od svih elemenata kao i početna: navigacija na vrhu, bočna navigacija i glavnog sadržaja. Stoga, ona ne će biti ponovo objašnjavana. Međutim, bitno je za naglasiti da ima razlike u glavnom sadržaju koji ovdje prikazuje sliku autora i tekst o njemu koji su stilizirani pomoću postavljanja fleksibilnog rasporeda elementa. Tako da se na većim ekranima sadržaj prikazuje u redu od dva stupca, dok je na manji zaslonima sadržaj smješten u jednom stupcu, prvo slika, a zatim tekst.



Slika 28. Prikaz stranice o instruktoru

Nakon prikaza početnog dijela vodiča prelazi se na sam vodič koji je prikazan na slici 29. Vodič se sastoji od svih elemenata kao i prethodne stranice ima navigaciju na vrhu, bočnu navigaciju i glavni sadržaj. Bočna navigacija u ovome slučaju nema dva linka nego se sastoji od linkova koji omogućavaju kretnje kroz lekcije vodiča.

Nadalje unutar glavnog sadržaja imamo prikaz dijelova koda, a njegovo oblikovanje biti će dodatno prikazano i objašnjeno u nastavku.



Slika 29. Prikaz stranice vodiča

```
<code class="flex mt-4 text-sm sm:text-base text-left items-center space-x-4 bg-gray-800 text-white rounded-md p-4 pl-6">
	<span class="flex gap-4">
		<span class="shrink-0 text-gray-400">$span>
		<span class="flex-1">npm install tailwindcss</span>
	</span>
</code>
```

Programski kod 17. Prikaz oblikovanja okvira za dijelove koda

Programski kod 17 prikazuje okvir unutar kojeg je smješten primjer programskog koda kojeg korisnik može kopirati i zatim koristiti unutar svoje aplikacije. Na <code> element i njegove unutarnje elemente primijenjene su različite klase Tailwind CSS-a kako bi postigao određeni izgled. U nastavku slijedi opis koda:

- <code class="flex mt-4 text-sm sm:text-base text-left items-center space-x-4 bg-gray-800 text-white rounded-md p-4 pl-6">: Ovaj dio koda unutar <code> elementa poprima tamno sivu pozadinu s bijelim tekstom, zaobljenim rubovima, unutarnjim razmakom i tekstom koji ima lijevo poravnanje. Također, koristi fleksibilan raspored elemenata i dodaje horizontalni razmak između elemenata.
- : Postavlja element koji omogućava fleksibilan raspored ostalih elemenata koji se nalaze unutar njega i dodaje razmak između njih od 4 jedinice prostora.
- $: Postavlja <span> element fiksne veličine i sivom bojom teksta (gray-400) te prikazuje simbol dolara ($).
- npm install tailwindcss: Postavlja element da zauzima ostatak preostalog prostora.

7.6. Prijenos web aplikacije na web server

Za prijenos web aplikacije koristi se FileZilla. Softver koji pruža mogućnost prijenos datoteka putem interneta koji se često koristi kao FTP klijent. Nakon instalacije programa, otvaramo sučelje i pristupamo „Site Manageru“ putem opcije „File“. Tu unesemo potrebne podatke: naziv poslužitelja, FTP protokol, korisničko ime i lozinku. Nakon toga, kliknemo na „Connect“ i ako su svi podatci točni povezani smo s poslužiteljem.

U sučelju programa, lijevi dio panela prikazuje sadržaj mape na našem računalu, dok su u desnom panelu prikazuje sadržaj našeg prostora na poslužitelju. Nadalje, potrebno je u desnom panelu otvoriti mapu „public_html“. Zatim odaberemo datoteke na svojem računalu koje se nalaze u lijevom panelu i povlačimo ih mišem u desni panel. Na taj način smo prenijeli sadržaj naše web stranice.

Završni rad koji opisuje navedeni proces može se pronaći na sljedećoj poveznici: http://arwen.unin.hr/~luseljan/zavrsni-rad/ .

7.7. Problem kod prijenosa aplikacije

Prilikom prijenosa aplikacije na server javio se problem učitavanja sadržaja. Aplikacija je radila na lokalnom poslužitelju, ali nakon izgradnje i prijenosa na server sadržaj se nije želio renderirati. Problem je bio u <BrowserRouter> elementu kojeg je potrebno zamjeniti s <HashRouter> elementom. Razlika je u tome što HashRouter simulira rutiranje unutar web stranice. To znači da će se sve rute prikazivati kao fragmenti URL-a nakon heš znaka.

Nakon što sam naprsvila zamjenu ta dva elementa te ponovo izgradila web aplikaciju i prenijela ju na server sve je radilo normalno.

8. Zaključak

Cilj ovog završnog rada bio je analizirati CSS Tailwind okvir prilikom izrade web aplikacije, prikazati njegova osnovna svojstva te pružiti bolji uvid u njegove mogućnosti.

Prvi dio rada fokusirao se na objašnjavanje osnovnih pojmova weba, uključujući web preglednike i HTTP protokol. Također, ukratko je objašnjen proces izrade korisničkog iskustva i sučelja. Nadalje, analizirani su CSS i CSS pre-procesori, te su navedeni popularni CSS okviri, a posebna pažnja bila je stavljena na Tailwind CSS.

U drugom dijelu rada prikazan je konkretan razvoj web aplikacije koristeći odabrani CSS okvir, Tailwind CSS. U tu svrhu korišteno je prikladno razvojno okruženje, samo programiranje je obavljeno uz pomoć React biblioteke. Svi koraci, od instalacije Tailwind CSS-a i React-a te strukturiranja datoteka do korištenja klasa i elemenata, detaljno su prikazani kroz primjere koda i prigodne slike. U konačnici, web aplikacija je postavljena na odgovarajući internetski server.

Prilikom izrade aplikacije uočene su prednosti korištenja Tailwind CSS okvira. Tailwind CSS pruža mnogobrojne predefinirane stilove i klase koje omogućavaju brzo oblikovanje elemenata sučelja. Sintaksa omogućava jednostavno kombiniranje klasa i prilagodbu dizajna, čime se ubrzava proces stiliziranja. Također, Tailwind CSS promovira ponovnu upotrebu koda i skalabilnost, što olakšava održavanje i nadogradnju web aplikacije.

Kada se uzmu u obzir sve navedene prednosti može se zaključiti da je Tailwind CSS moćan alat koji može značajno ubrzati proces izrade web aplikacija. Također, omogućava programerima da se više usredotoče na funkcionalnost i korisničko iskustvo, dok istovremeno pruža fleksibilnost u dizajnu.

9. Literatura

[1] https://www.geeksforgeeks.org/introduction-to-tailwind-css/s, dostupno 26.0.4.2023.

[2] https://www.simplilearn.com/tutorials/css-tutorial/css-framework, dostupno 26.04.2023.

[3] https://developer.mozilla.org/en-US/docs/Web/CSS/Media_Queries, dostupno 24.04.2023.

[4] https://css-tricks.com/snippets/css/complete-guide-grid/, dostupno 24.04.2023.

[5] An overview of HTTP - HTTP | MDN (mozilla.org), dostupno 05.04.2023.

[6] https://www.cloudflare.com/en-gb/learning/ssl/why-is-http-not-secure/, dostupno 05.04.2023.

[7] https://developer.mozilla.org/en-US/docs/Web/Performance/How_browsers_work, dostupno, 07.04.2023.

[8] https://web.dev/howbrowserswork/, dostupno, 07.04.2023.

[9] https://www.uxdesigninstitute.com/blog/ux-design-process/, dostupno 13.04.2023.

[10] https://uxdesign.cc/5-steps-of-the-ux-design-process-to-practice-dc2ab8981895, dostupno 13.04.2023.

[11] https://www.uxdesigninstitute.com/blog/what-is-ui-design/, dostupno 14.04.2023.

[12] https://www.sketch.com/, dostupno 14.04.2023.

[13] https://www.adobe.com/, dostupno 14.04.2023.

[14] https://www.figma.com/, dostupno 14.04.2023.

[15] https://developer.mozilla.org/en-US/docs/Web/CSS, dostupno 17.04.2023.

[16] https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/CSS_basics#what_is_css, dostupno 17.04.2023.

[17] https://www.w3schools.com/css/css_syntax.asp, dostupno 17.04.2023.

[18] https://www.mojwebdizajn.net/, dostupno 18.04.2023.

[19] https://developer.mozilla.org/en-US/docs/Learn/CSS/Building_blocks/The_box_model, dostupno 19.04.2023.

[20] https://css-tricks.com/snippets/css/a-guide-to-flexbox/, dostupno 22.04.2023.

[21] https://www.geeksforgeeks.org/introduction-to-tailwind-css/, dostupno 30.04.2023.

[22] Klimm, M. C. (2021). Design Systems for Micro Frontends-An Investigation into the Development of Framework-Agnostic Design Systems using Svelte and Tailwind CSS (Doctoral dissertation, Hochschulbibliothek der Technischen Hochschule Köln).

[23] Zhao, Z. (2023). Build A Live News Application With Next. js 13.

[24] Chang, K. N., & Henderson, P. (2011). A Practice of UML for Web Development. In Proceedings of the International Conference on Software Engineering Research and Practice (SERP) (p. 1). The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp)

[25] Mičuda, D. (2019). CSS proširenja kroz sustav SASS (Završni rad). Koprivnica: Sveučilište Sjever. Preuzeto s https://urn.nsk.hr/urn:nbn:hr:122:892697

[26] Kukec, E. (2017). CSS proširenja kroz sustav LESS (Završni rad). Koprivnica: Sveučilište Sjever. Preuzeto s https://urn.nsk.hr/urn:nbn:hr:122:437513

[27] Chatzoglou, E., Kouliaridis, V., Kambourakis, G., Karopoulos, G., & Gritzalis, S. (2023). A hands-on gaze on HTTP/3 security through the lens of HTTP/2 and a public dataset. *Computers & Security*, *125*, 103051.

[28] García, B., Ricca, F., del Alamo, J. M., & Leotta, M. (2023). Enhancing Web Applications Observability through Instrumented Automated Browsers. *Journal of Systems and Software*, *203*, 111723.

[29] Wilson, D., Hassan, S. U., Aljohani, N. R., Visvizi, A., & Nawaz, R. (2023). Demonstrating and negotiating the adoption of web design technologies: Cascading Style Sheets and the CSS Zen Garden. *Internet Histories*, *7*(1), 27-46.

[30] Deac, L. (2014). Modern web design techniques: a practical approach.

Popis slika

Programski kodovi

Prilozi

[1] Figma dokument s dizajnom i materijalima koji su korišteni prilikom izrade web aplikacije, ali i pisanja rada: https://www.figma.com/file/VbRmuPsxmboZRI4lx9kHMY/Zavrsni-rad---Tailwind-CSS-okvir-otvorenog-koda?type=design&node-id=11%3A295&t=DXcThoInS7vFf9u2-1

[2] Objavljeni praktični dio rada na arwen.unin.hr servisu: http://arwen.unin.hr/~luseljan/zavrsni-rad/

Sveučilište
Sjever

SVEUČILIŠTE
SJEVER

IZJAVA O AUTORSTVU

Završni/diplomski rad isključivo je autorsko djelo studenta koji je isti izradio te student odgovara za istinitost, izvornost i ispravnost teksta rada. U radu se ne smiju koristiti dijelovi tuđih radova (knjiga, članaka, doktorskih disertacija, magistarskih radova, izvora s interneta, i drugih izvora) bez navođenja izvora i autora navedenih radova. Svi dijelovi tuđih radova moraju biti pravilno navedeni i citirani. Dijelovi tuđih radova koji nisu pravilno citirani, smatraju se plagijatom, odnosno nezakonitim prisvajanjem tuđeg znanstvenog ili stručnoga rada. Sukladno navedenom studenti su dužni potpisati izjavu o autorstvu rada.

Ja, LUCIJA SOJAN *(ime i prezime)* pod punom moralnom, materijalnom i kaznenom odgovornošću, izjavljujem da sam isključivi autor/ica završnog/diplomskog *(obrisati nepotrebno)* rada pod naslovom TAILWINDCSS PREPROCESSOR *(upisati naslov)* te da u navedenom radu nisu na nedozvoljeni način (bez pravilnog citiranja) korišteni dijelovi tuđih radova.

Student/ica:
(upisati ime i prezime)

Lucija Sojan
(vlastoručni potpis)

Sukladno čl. 83. Zakonu o znanstvenoj djelatnost i visokom obrazovanju završne/diplomske radove sveučilišta su dužna trajno objaviti na javnoj internetskoj bazi sveučilišne knjižnice u sastavu sveučilišta te kopirati u javnu internetsku bazu završnih/diplomskih radova Nacionalne i sveučilišne knjižnice. Završni radovi istovrsnih umjetničkih studija koji se realiziraju kroz umjetnička ostvarenja objavljuju se na odgovarajući način.

Sukladno čl. 111. Zakona o autorskom pravu i srodnim pravima student se ne može protiviti da se njegov završni rad stvoren na bilo kojem studiju na visokom učilištu učini dostupnim javnosti na odgovarajućoj javnoj mrežnoj bazi sveučilišne knjižnice, knjižnice sastavnice sveučilišta, knjižnice veleučilišta ili visoke škole i/ili na javnoj mrežnoj bazi završnih radova Nacionalne i sveučilišne knjižnice, sukladno zakonu kojim se uređuje znanstvena i umjetnička djelatnost i visoko obrazovanje.