Razvoj web aplikacija pomoću Blazor tehnologije

Baltić, Danijel

Sveučilište Jurja Dobrile u Puli

Fakultet ekonomije i turizma

„Dr. Mijo Mirković“

Danijel Baltić

RAZVOJ WEB APLIKACIJA POMOĆU BLAZOR TEHNOLOGIJE

Diplomski rad

Pula, 2023.

Sveučilište Jurja Dobrile u Puli

Fakultet ekonomije i turizma

„Dr. Mijo Mirković“

RAZVOJ WEB APLIKACIJA POMOĆU BLAZOR TEHNOLOGIJE

Diplomski rad

Danijel Baltić

JMBAG: 0303067488, redovan student

Studijski smjer: Poslovna ekonomija - Informatički menadžment

Kolegij: Web tehnologije

Mentor: dr.sc. Goran Matošević

Pula, srpanj 2023.

IZJAVA O AKADEMSKOJ ČESTITOSTI

Ja, dolje potpisani _____, kandidat za prvostupnika _____ovime izjavljujem da je ovaj Završni rad rezultat isključivo mojega vlastitog rada, da se temelji na mojim istraživanjima te da se oslanja na objavljenu literaturu kao što to pokazuju korištene bilješke i bibliografija. Izjavljujem da niti jedan dio Završnog rada nije napisan na nedozvoljen način, odnosno da je prepisan iz kojega necitiranog rada, te da ikoji dio rada krši bilo čija autorska prava. Izjavljujem, također, da nijedan dio rada nije iskorišten za koji drugi rad pri bilo kojoj drugoj visokoškolskoj, znanstvenoj ili radnoj ustanovi.

Student

_____

U Puli, _____, _____ godine

IZJAVA

o korištenju autorskog djela

Ja, _____ dajem odobrenje Sveučilištu Jurja Dobrile u Puli, kao nositelju prava iskorištavanja, da moj završni rad pod nazivom

_____

_____

_____ koristi na način da gore navedeno autorsko djelo, kao cjeloviti tekst trajno objavi u javnoj internetskoj bazi Sveučilišne knjižnice Sveučilišta Jurja Dobrile u Puli te kopira u javnu internetsku bazu završnih radova Nacionalne i sveučilišne knjižnice (stavljanje na raspolaganje javnosti), sve u skladu s Zakonom o autorskom pravu i drugim srodnim pravima i dobrom akademskom praksom, a radi promicanja otvorenoga, slobodnoga pristupa znanstvenim informacijama.

Za korištenje autorskog djela na gore navedeni način ne potražujem naknadu.

U Puli, _____ (datum)

Potpis

_____

Sadržaj

1. Uvod

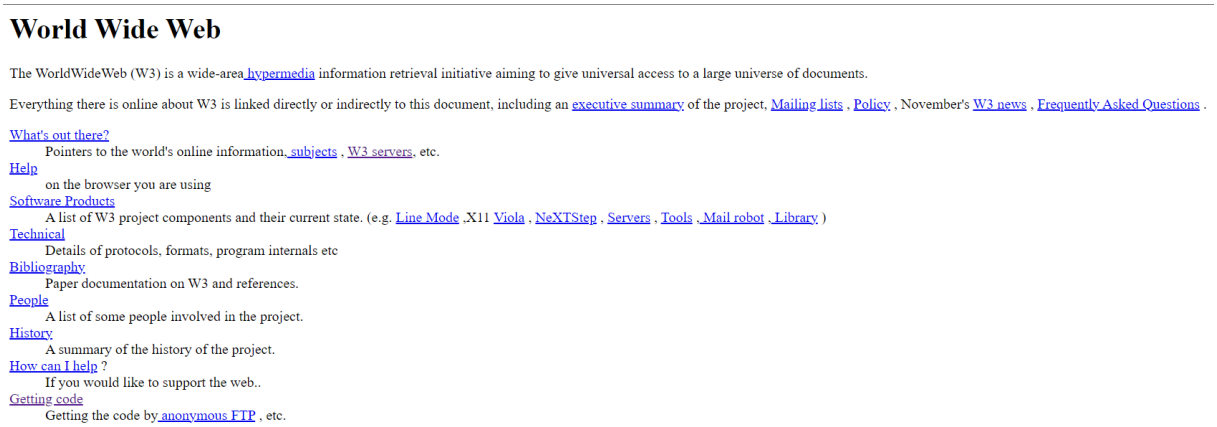
Web aplikacije su interaktivna programska rješenja koja su najčešće spremljena na nekom udaljenom serveru, a korisnik njima pristupa putem internetskih preglednika. Usprkos velikoj popularnosti web aplikacija, mobilne aplikacije ih zamjenjuju zbog same popularnosti pametnih telefona, ali vjeruje se da postoji i dalje velika potražnja za web rješenjima. Prema istraživanju Erica Enge u 2020 godini 68% internetskih web mjesta posjećeno je putem mobitela, što je u odnosnu na 2019 skok od 5% (Enge, 2021). Dok su prema istom istraživanju internetske posjete preko stolnih računala pale za 3% sa 32%, na 29%, te su i pretraživanja putem tableta također u padu sa 5% na 3%. To istraživanje se provelo pomoću „Google Analytics' Benchmarking" uz preko 30 trilijuna posjeta i u 2019 kao i u 2020 godini. Stoga svaka web aplikacija bi čak više trebala gledati dizajn i praktičnost same aplikacije koja se prikazuje na mobilnim uređajima nago prikaza na stolnom računalu.

Web aplikacije će i dalje igrati značajnu ulogu u svim poslovnim sektorima, ne samo u IT industriji, gdje se najčešće koriste. Web aplikacije imaju značajnu prednost jer ne zahtijevaju instalaciju na tvrdi disk i pristupa im se jednostavno putem Internet preglednika. Neke od prednosti su da se isti sadržaj prikazuje svim korisnicima bez obzira na njihov operacijski sustav. Nema potrebe za nadogradnjama koje korisnik mora instalirati, a promjene se mogu izvršavati online.

2. Web Aplikacije i njihova povijest

2.1 Povijest web aplikacija

Britanski znanstvenik po imenu Tim Berners-Lee, koji je bio zaposlen u laboratoriju Europske organizacije za nuklearna istraživanja (CERN), osjetio je frustraciju zbog raznolikih informacija koje su bile spremljene na različitim računalima i zahtijevale ponovno prijavljivanje za pristup. Berners-Lee se zapitao: „Zar ne postoji drugi način povezivanja i dijeljenja informacija, neki imaginarni sustav koji ih sve povezuje?“ (Nožinić, 2016) te je tako nastala prva Internet stranica. Navedena stranica nije izgledala ni približno onome što danas imamo. I danas joj se može pristupiti pomoću URL-a: http://info.cern.ch/hypertext/WWW/TheProject.html. Njen izgled je prikazan na Slici 1. Radi se o jednostavnoj HTML stranici sa kratkim tekstom i nekoliko poveznica.



Slika 1: Snimka zaslona prve web stranice

Izvor: Snimka zaslona internetske stranice sa linka http://info.cern.ch/hypertext/WWW/TheProject.html

Sam princip i način na koji je nastala ova prva web stranica postoji i danas, a to je da istim podacima može pristupiti svako sa internetskom vezom i pravom Internet adresom.

2.2 Web 1.0

Web 1.0 predstavlja prvu razinu razvoja web stranica. Ideja web 1.0 je bila slična onoj od Berners-Leea. Da veći broj ljudi dođe brže i efikasnije do istih podataka, sadržaja i informacija. Ona je zapravo bila smo za čitanje, nije bilo nikakve interakcije klijenta i web stranice. Ovo razdoblje je trajalo od kreiranje prve Berners-Leeove stranice 1990 godine pa sve do kraja tisućljeća odnosno do početka 2000 godine. Web 1.0 podrazumijeva izradu statičkih HTLM stranica koje se povezuju „hyperlinkovima“. Kasnije je HTML 3.2 uveo tablice i forme koje su se koristile za slanje podataka putem elektroničke pošte. Potom su došli i GIF-ovi i ostale grafike koje se spremaju kao datoteke na serveru (Terra, 2023). Web 1.0 je usmjeren na dostavu informacija, no nije omogućavao interakciju s tim informacijama u virtualnom okruženju.

2.3 Web 2.0

Dok je Web 1.0 bio samo za čitanje i koristio samo statični sadržaj bez korisničke interakcije sa istim, web 2.0 je postao prava mala revolucija u internetskom sadržaju jer su konačno korisnici mogli sudjelovati u stvaranju sadržaja. Dakle web 2.0 je naglašavao interakciju sa klijentima, korisnički generiran sadržaj (UGC), lakoću korištenja i poboljšanu kompatibilnosti različitih uređaja i sustava (Terra, 2023). Web 2.0 ne odnosi se na nikakva posebna tehnička poboljšanja Interneta. Jednostavno se odnosi na promjenu u načinu korištenja Interneta u 21. stoljeću. U novom dobu, postoji viši stupanj dijeljenja informacija i povezanosti među sudionicima. (Kenton, 2022) Tako dolazimo do zaključka da je Web 2.0 bio koncentriran na krajnjeg korisnika i njegovo iskustvo i doživljaj web sadržaja. Točno u tom trenutku se stvaraju prve društvene mreže, elektronička komunikacija dobiva na značaju, a to sve možemo zahvaliti novim tehnologijama, socijalnim potrebama pa i komercijalnim interesima. Neke od značajki Web 2.0 su:

- Korisnici generiraju sadržaj: Web 2.0 potiče sudjelovanje korisnika i omogućava im stvaranje, dijeljenje i uređivanje sadržaja na webu. Primjeri uključuju blogove, wiki stranice, forume i društvene mreže.
- Interaktivnost i suradnja: Web 2.0 pruža interaktivno iskustvo korisnicima. Korisnici mogu sudjelovati u različitim aktivnostima kao što su komentiranje,

ocjenjivanje, dijeljenje, sudjelovanje u igrama i suradnja na projektima putem online alata za suradnju.

- Društvene mreže: Web 2.0 je obilježen razvojem društvenih mreža kao što su Facebook, Twitter, Instagram i LinkedIn. Društvene mreže omogućavaju korisnicima da se povežu, dijele sadržaj, komuniciraju i izgrađuju mreže s drugim korisnicima.
- Rich Internet Applications (RIA): Web 2.0 je donio razvoj naprednih web aplikacija koje su slične desktop aplikacijama po svojim mogućnostima. RIA koristi tehnologije poput AJAX-a[1] (Asynchronous JavaScript and XML) kako bi pružio bogatije korisničko iskustvo, brže učitavanje i dinamičke funkcionalnosti.
- Povezivanje i integracija: Web 2.0 omogućava povezivanje i integraciju različitih web servisa i aplikacija. Primjeri uključuju mogućnost dijeljenja sadržaja s jedne web stranice na drugu putem widgeta i API-ja koji omogućavaju razmjenu podataka između različitih platformi.
- Personalizacija i prilagođavanje: Web 2.0 pruža mogućnost prilagodbe i personalizacije sadržaja i usluga prema preferencijama korisnika. Korisnici mogu prilagoditi sučelje, postavke i sadržaj kako bi odgovarali njihovim potrebama i interesima.

2.4 Web 3.0

Kao sljedeća faza evolucije interneta, Web 3.0 ima za cilj pružiti inteligentno, autonomno i personalizirano korisničko iskustvo. Web 3.0 je treća generacija Interneta koja se trenutno gradi, gdje će web stranice i aplikacije moći obraditi informacije na pametan način sličan ljudskom putem tehnologija poput umjetne inteligencije (AI), strojnog učenja (ML), „Big data“, tehnologija distribuirane baze podataka (DLT) i drugih (Vermaak, 2022). U eri Web 3.0, procesi stvaranja sadržaja i donošenja odluka uključivat će i ljude i strojeve, olakšavajući inteligentno generiranje i distribuciju visoko prilagođenog sadržaja pojedinačnim korisnicima. Ova nova faza interneta također će

[1] Ajax je tehnika koja koristi „XMLHttpRequest“ objekt koji pak šalje zahtjev za podacima na server i JavaScript i HTML DOM za prikazivanje ili korištenje tih podataka

naglasiti decentralizaciju, pružajući korisnicima veću kontrolu nad njihovim podacima i online interakcijama. Neke od ključnih karakteristika web 3.0:

- Decentralizacija – Web 3.0 bi trebao omogućiti kreiranje decentraliziranih aplikacija i platformi koje distribuiraju kontrolu i vlasništvo prema korisnicima i zajednicama.
- Semantički web – koncept razvijen s ciljem poboljšanja weba tako da se informacije na njemu mogu bolje razumjeti i obrađivati od strane računala.
- Umjetna inteligencija i strojno učenje – automatiziranim zadatcima i omogućavanju stroja da uči i adaptira se, s vremenom takve tehnologije će sudjelovat u kreiranju pametnijeg i prilagodljivijeg digitalnog eko sustava.
- Interoperabilnost – Omogućiti će besprijekornu integraciju i dijeljenje podataka diljem digitalnog svijeta. Odnosno omogućiti će veću sposobnost sustava za pružanje i dijeljenje usluga i informacija.
- Korisnička kontrola i privatnost – kroz korištenje blockchain tehnologija i decentralizacije, korisnici će biti u mogućnosti održati vlasništvo nad svojim podatcima te će time moći zaštiti svoju privatnost od iskorištavanja trećih strana.

3. Razvoj web aplikacija

Razvoj web aplikacija je proces stvaranja interaktivnih aplikacija koje korisnici mogu koristiti putem web preglednika. Ovaj proces obuhvaća niz faza koje omogućavaju programerima da planiraju, izrade i održavaju aplikaciju kako bi zadovoljili potrebe korisnika.

3.1. Faze razvoja aplikacija

Svaka faza ima svoje specifične zadatke i ciljeve, a uspješno upravljanje svim fazama ključno je za razvoj visokokvalitetnih web aplikacija koje zadovoljavaju potrebe korisnika i ciljeve poslovanja. Faze razvoja su: planiranje i analiza, dizajn korisničkog sučelja (UI) i korisničkog iskustva (UX), razvoj, testiranje, implementacija i puštanje u produkciju, implementacija i puštanje u produkciju, održavanje i ažuriranje i praćenje i analiza.

3.1.1. Planiranje i analiza

Razvoj web aplikacija započinje planiranjem i analizom. Tim za razvoj definira svrhu aplikacije, identificira ciljnu publiku i istražuje tržište. Ova faza uključuje duboko razumijevanje poslovnih ciljeva kako bi se postavila osnovna strategija projekta. Analizom se identificiraju ključne značajke i funkcionalnosti aplikacije, a planiranje uključuje definiranje rasporeda rada, budžeta i resursa. Ova faza igra ključnu ulogu u usmjeravanju cijelog razvojnog procesa.

3.1.2. Dizajn korisničkog sučelja (UI) i korisničkog iskustva (UX)

Nakon planiranja, slijedi faza dizajna korisničkog sučelja i korisničkog iskustva. UI ili korisničko sučelje se odnosi na fizički izgled i interakciju sa korisnikom pomoću boja, fontova, ikona, tipki, slika te samu organizaciju elemenata na ekranu dok UX ili korisničko iskustvo se odnosi na ukupni doživljaj korisnika poput brzine učitavanja, jednostavnost navigacije, prikazivanje relativnog sadržaja. U ovoj fazi stvaraju se nacrti sučelja, prototipovi i dizajn korisničkog sučelja. Nacrti sučelja pomažu u postavljanju rasporeda elemenata na stranici, dok stvaranje prototipova omogućuje

interaktivno isprobavanje koncepta prije nego što se krene u razvoj stvarnog koda. Dizajneri također stvaraju vizualni identitet aplikacije, uključujući boje, tipografiju, ikone i slike.

3.1.3. Razvoj

Nakon što je dizajn definiran, dolazi vrijeme za razvoj. Ova faza podijeljena je na backend i frontend razvoj. Backend razvoj uključuje pisanje koda koji upravlja poslovnom logikom i pristupom bazi podataka. Programeri koriste različite programski jezike i okvire, ovisno o zahtjevima projekta. Frontend razvoj, s druge strane, usredotočuje se na izradu korisničkog sučelja vidljivog korisnicima putem web preglednika. Ovdje se koriste jezici kao što su HTML, CSS, JavaScript ali i drugi.

3.1.4. Testiranje

Nakon što je aplikacija razvijena, provodi se niz testiranja kako bi se osigurala njena funkcionalnost i kvaliteta. To uključuje jedinično testiranje (testiranje pojedinih komponenata koda), funkcionalno testiranje (testiranje kako aplikacija radi kao cjelina), testiranje performansi (testiranje brzine i odgovora aplikacije) i sigurnosno testiranje (identifikacija potencijalnih ranjivosti).

3.1.5. Implementacija i puštanje u produkciju

Kada su svi testovi uspješno prošli, aplikacija se implementira na web poslužitelju i pušta u produkciju kako bi postala dostupna korisnicima. Ova faza zahtijeva pažljivo praćenje kako bi se osigurala stabilnost aplikacije.

3.1.6. Održavanje i ažuriranje

Nakon lansiranja, aplikacija zahtijeva redovito održavanje. Tim za održavanje rješava greške, dodaje nove značajke i osigurava sigurnost aplikacije. Održavanje je kontinuiran proces koji osigurava kvalitetu i konkurentnost aplikacije.

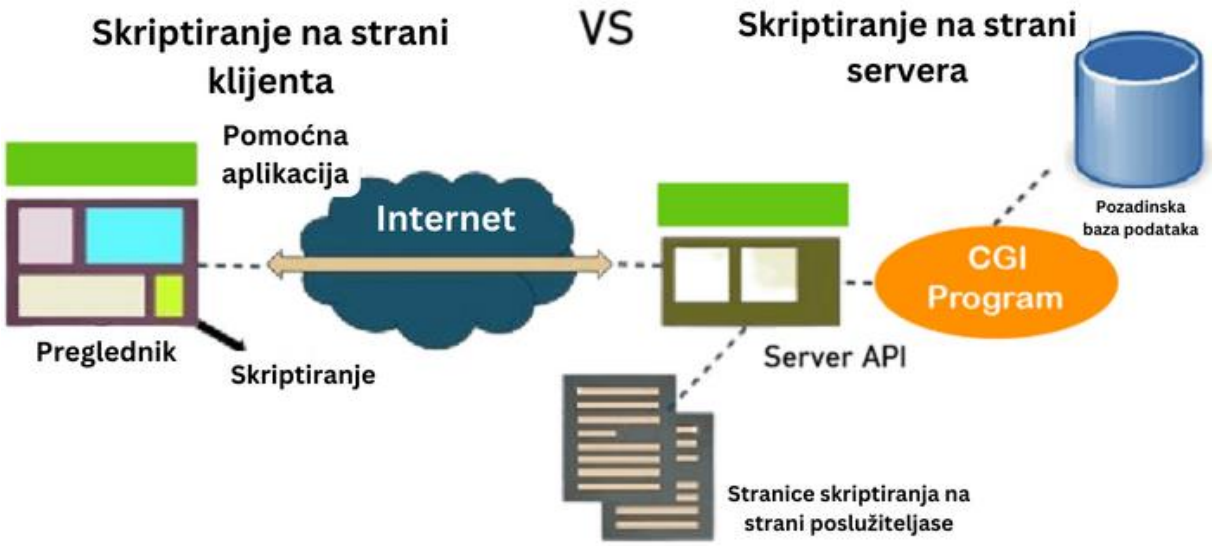
3.1.7. Praćenje i analiza

Tijekom cijelog životnog ciklusa aplikacije, važno je pratiti njenu izvedbu i korisničko iskustvo. Analitika se koristi kako bi se razumjelo kako korisnici interagiraju s aplikacijom, identificirali problemi i prilagodile strategije kako bi se postigli poslovni ciljevi.

Ovaj proces razvoja web aplikacija osigurava da aplikacija zadovoljava potrebe korisnika, poslovnih ciljeva i tržišta. Svaka faza ima svoju svrhu i značajku, a timski rad i dobro upravljanje ključni su za uspješan razvoj aplikacije.

4. Tehnologije za web aplikacije

Sve web tehnologije se mogu podijeliti na dvije vrste, a to su serverske tehnologije i alati (Back-end) te klijentske tehnologije i alati (Front-end). Serverske tehnologije i alati se koriste za kreiranje procesa i zadataka koji će se odvijati na serverskoj strani web aplikacije i te procese klijenti ne vide, dok klijentske tehnologije služe za prikaz podataka krajnjem korisniku i interakciju s njim. To je jedan od glavnih zadataka klijentskih tehnologija i alata, da prikažu i interaktivno predstave podatke koje je server obradio i koji se nalaze na serveru.



Slika 2: Client-side vs Server-side usporedba

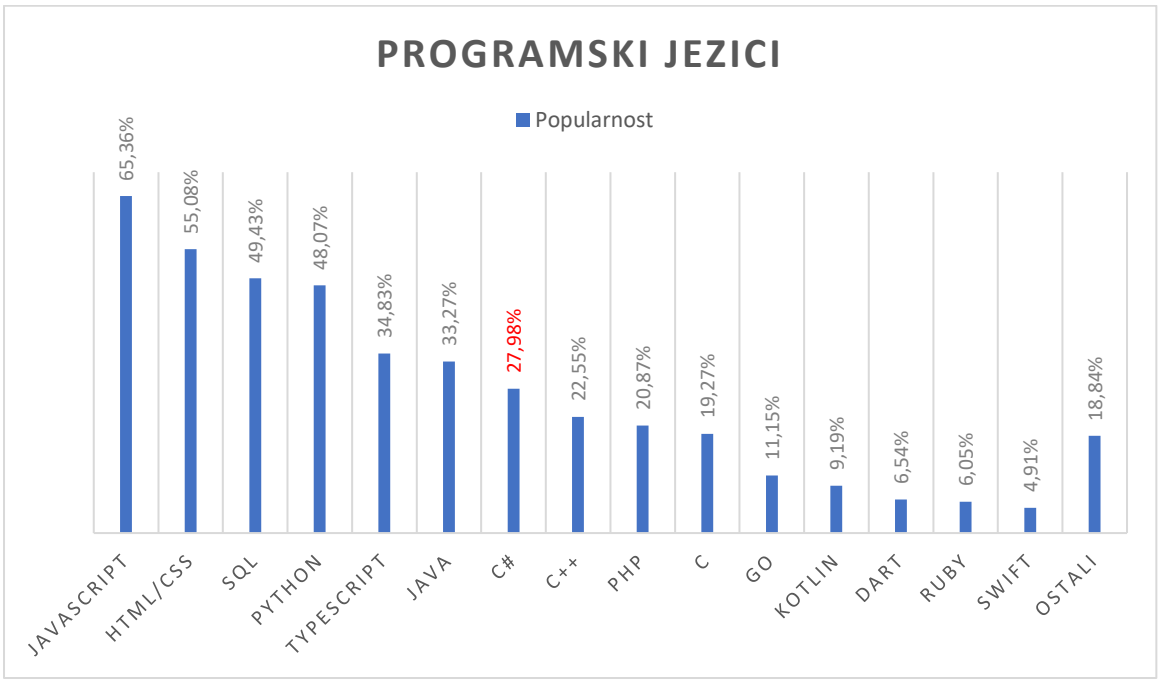
Izvor: Stack: Advantages and Disadvantages of Client-side scripting
https://www.javatpoint.com/advantages-and-disadvantages-of-client-side-scripting

4.1 Programski jezici

Programski jezici su skup naredbi pisanih od strane programera kako bi isporučili upute računalu da izvrši i riješi neki zadatak (Tuama, 2022). Svaki programski jezik ima svoje specifičnosti i svojstva koja ga čine korisnim za određene namjene. Postoji nekoliko vrsta programskih jezika, uključujući proceduralne, objektno orijentirane, funkcionalne i skriptne jezike.

Proceduralni programski jezici se fokusiraju na korake koji se moraju poduzeti za rješavanje problema. Oni se sastoje od niza uputa koje se redom izvršavaju. Objektno orijentirani jezici, s druge strane, naglašavaju objekte kao temeljne elemente programiranja, gdje objekt predstavlja instancu klase i sadrži metode i svojstva. Funkcionalni jezici se temelje na matematičkim funkcijama koje obrađuju podatke i vraćaju rezultate, dok su skriptni jezici često korišteni za automatizaciju zadataka i programiranja web stranica.

Neki od popularnih programskih jezika danas su C#, Python, Java, JavaScript, Ruby, PHP itd. Svaki od ovih jezika ima svoje prednosti i nedostatke, a izbor ovisi o namjeni i zahtjevima projekta. Programski jezici se kontinuirano razvijaju i ažuriraju kako bi pratili brze promjene u tehnologiji i potrebama korisnika, stvarajući tako novije i moćnije alate za programiranje. Na Slici 3 možemo vidjeti koji su najpopularniji jezici korišteni na najpopularnijem forumu za rješavanje programerskih problema „Stack Overflow“.



Slika 3: Tablica Popularnosti programskih jezika na forumu "Stack Owerflow"

Izvor: Stack Overflow: https://survey.stackoverflow.co/2022/#technology-most-popular-technologies

Jedna od najznačajnijih karakteristika modernih programska jezika je fleksibilnost i skalabilnost. To znači da se mogu koristiti za razne vrste aplikacija, od malih skripti do velikih poslovnih sustava. Također, mnogi moderni jezici imaju bogatu zajednicu

korisnika i razvojnih inženjera koji doprinose daljnjem razvoju jezika i razvoju pripadajućih biblioteka i okvira za rad. To omogućuje programerima da brže razvijaju aplikacije, rješavajući probleme kroz zajedničku suradnju i dijeljenje znanja. Ukratko, programski jezici su neophodni alati u današnjem digitalnom svijetu, a njihova primjena se stalno razvija i unapređuje.

4.1.1 JavaScript

JavaScript je stvoren u tvrtki Netscape Communications 1995. godine od strane Brendana Eicha (Naveed Fida, 2023). Netscape i Eich su osmislili JavaScript kao skriptni jezik koji će se koristiti uz njihov najvažniji web preglednik, Netscape Navigator. U početku poznat kao LiveScript, Netscape je promijenio ime u JavaScript kako bi ga mogli predstaviti kao partnera za programski jezik Java, koji je proizvod njihovog partnera, tvrtke Sun Microsystems. Međutim, osim nekih površinskih sintaktičkih sličnosti, JavaScript nije na bilo koji način povezan s programskim jezikom Java.

JavaScript je skriptni ili programski jezik koji omogućuje implementaciju složenih značajki na web stranicama (MDN contributors, 2023). Svakom interaktivnom elementu na web stranicama koji prelazi granice statičnih informacija najvjerojatnije je potreban JavaScript - jezik za skriptiranje ili programiranje koji omogućuje implementaciju složenih značajki. Ova vrsta elemenata uključuje ažuriranja vremenskih informacija, interaktivne karte, animirane 2D/3D grafike, video boksove s pomicanjem i mnoge druge. U paleti standardnih web tehnologija, JavaScript predstavlja treći sloj nakon HTML-a i CSS-a.

U 2008. godini stvaranje Googleovog otvorenog izvornog koda Chrome V8, JavaScript alata visokih performansi, predstavljalo je ključni trenutak za JavaScript. Naknadna širenja brzih JavaScript alata omogućila su programerima izgradnju sofisticiranih aplikacija temeljenih na pregledniku s performansama koje se mogu usporediti s desktop i mobilnim aplikacijama.

Uskoro nakon toga, Ryan Dahl objavio je otvoreno okruženje za izvođenje koda zvan Node.js, koje je omogućilo pokretanje JavaScript koda izvan preglednika. To je oslobodilo JavaScript od okvira preglednika i direktno dovelo do trenutne popularnosti JavaScripta. Danas možete koristiti JavaScript za pisanje različitih vrsta aplikacija, uključujući aplikacije za preglednik, poslužitelj, mobilne i desktop aplikacije. Većina

većih IT tvrtki, uključujući Facebook, Twitter, Netflix i Google, koristi JavaScript u svojim proizvodima.

JavaScript zajednica danas je vjerojatno najaktivnija programerska zajednica. Često se čini da svaki tjedan donosi nove alate, okvire i biblioteke. Dostupni su predprocesori i prevoditelji koji omogućuju prevođenje modernih JavaScript programa u oblike koji se mogu pokrenuti na starijim JavaScript platformama, kao i prevoditelji za potpuno nove jezike koji koriste JavaScript kao ciljni jezik. JavaScript standard je dokument koji se neprestano razvija, s kontinuiranim poboljšanjima koja se ubrzano uvode. JavaScript alati gotovo jednako brzo uključuju ove promjene. Novi operativni sustavi koji se razvijaju (poput Google-ovog Fuchsia) dodaju podršku za izradu nativnih aplikacija u JavaScriptu. Sve ovo ukazuje na to da JavaScript ima izuzetno uzbudljivu budućnost.

4.1.2 TypeScript

TypeScript je strogo tipiziran što znači da ima stroga pravila za provjeru tipova i ne dopušta implicitne konverzije između različitih tipova. Također je objektno orijentiran i kompilirani programski jezik koji se temelji na JavaScriptu. Stvoren je kao nad skup JavaScript jezika s ciljem pružanja naprednijeg alata u različitim situacijama. (Shubel, 2022). Glavni arhitekt iza TypeScripta je Anders Hejlsberg, dizajner C# u Microsoftu. TypeScript je otvorenoga koda kojeg podržava Microsoft i smatra se i jezikom i skupom alata.

Prednost korištenja TypeScripta leži u tome da omogućava dobivanje više informacija o kodu putem označenog tipkanja. Na taj način postoji mogućnost provjere i verifikacije koda prije nego što ga se pokrene. To omogućava kvalitetniju tehničku dokumentaciju koja će biti korisna drugim programerima. Također, zahvaljujući naprednoj automatskoj interferenciji u TypeScriptu, nije obavezno kodiranje u tipovima. Programeri mogu postupno prilagoditi svoj kod TypeScriptu. Općenito, TypeScript ima za cilj unaprijediti učinkovitost i produktivnost programera kroz smanjenje pogrešaka, rješavanje problema i pružanje naprednih alata. Jedna od njegovih ključnih prednosti je mogućnost izvođenja kao pozadinskog procesa, što pruža podršku za kompilaciju i integraciju s razvojnim okruženjima (IDE). Ovaj aspekt čini TypeScript korisnim alatom za programere jer im omogućuje rad u ugodnom razvojnom okruženju s naprednim

značajkama. Kroz smanjenje grešaka i pružanje boljih alata, TypeScript pomaže programerima u isporuci visokokvalitetnog koda i olakšava održavanje i razvoj velikih projekata.

4.1.3 PHP

PHP je serverski skriptni jezik otvorenog koda kojeg mnogi programeri koriste za razvoj web aplikacija (Chris, 2021). Prema istraživanju Web Technology system, PHP je korišten čak u 78.1% od svih web stranica, uključujući istranice koje imaju veliki broj posjetitelja poput Facebooka ili Wikipedie (Astari, 2023). Tijekom kreiranja i u samim početcima PHP jezika, ta skraćenica je predstavljala Personal Homepage (Osobna Početna Stranica). Danas ona ipak ima drugo značenje jer je i jezik poprimio novi izgled u odnosu na onaj koji je kreiran još 1994. godine. PHP je serverski skriptni jezik, što znači da se instrukcije u skripti izvršavaju na poslužitelju. Poslužitelj zatim pruža podatke na zahtjev, upravlja zahtjevima i pristupa informacijama u bazi podataka. Najčešća primjena PHP-a je generiranje dinamičkog web sadržaja, čitanje i pisanje podataka u datotekama i bazama podataka. Osim u pregledniku (browseru), može se izvoditi i u naredbenom retku (command line).

PHP ima niz prednosti koje su doprinijele njegovoj popularnosti kao omiljenom jeziku za web poslužitelje već više od 15 godina. A neke od prednosti su: Platformska neovisnost, odnosno neovisno o operacijskim sustavima, nije mu bitno dali koristite Windows, Mac ili Linux, jer radi na svim platformama. Izvorni kod je dostupan svima, tj. on je „Open sorce“ i svi mogu pristupit kodu i nadograđivat ga po želji. Lak je za učenje početnicima, a poželjno bi bilo osnovno poznavanje HTML, CSS i JavaScripta. Također PHP se može spojiti sa svim bazama podataka kako relacijskim tako i ne relacijskim. Lako i brzo se povezuje sa MySQL-om, Postgresom, MongoDB-om ili bilo kojom drugom bazom. PHP također uživa veliku podršku IT zajednice te službena dokumentacija pruža detaljne vodiče o tome kako se koristiti.

Jedna od popularnijih web aplikacija koja koristi PHP je WordPress – jedan od najpoznatijih sustava za upravljanje sadržajem (eng. Content Management system - CMS) i jednostavno kreiranje web stranica. PHP koriste i druge CMS platforme poput Drupal-a, Joomla i Magento. Mnoge platforme za web hosting, kao što su BlueHost, SiteGround, Hostinger i druge, koriste PHP za pružanje svojih usluga.

4.1.4 Python

Python je objektno orijentiran, interpreterski i interaktivan programski jezik (Holden, 2018). Python je kreirao Guido van Rossum, a prvi put je objavljen 20. veljače 1991. Ime programskog jezika Python dolazi od stare BBC televizijske humorističke skeč-serije pod nazivom Monty Python's Flying Circus (Python Institute, 2021).

Slika 4: Logo Python programskog jezika

Izvor: Preuzeto sa: https://towardsdatascience.com/the-zen-of-python-a-guide-to-pythons-design-principles-93f3f76d088a

Iako je prvotno bio napravljen o strane jednoga čovjeka, kasnije se razvijao najčešće sa anonimnim IT stručnjacima i drugim IT entuzijastima. Danas se on održava od strane „Python Software Foundation“, neprofitne organizacije i zajednice koja razvija, unaprjeđuje, proširuje i promovira Python programski jezik i njegovo okruženje (Python Institute, 2021).

Python, jedan od najpopularnijih programskih jezika na svijetu, integriran je u puno softverskih rješenja od Netflix-ovog algoritma za preporuke do softvera koji upravlja autonomnim vozilima (Coursera authors, 2023). Python je često korišten za razvoj web stranica i softvera, automatizaciju zadataka, analizu i vizualizaciju podataka. Budući da ga je relativno lako naučiti, Python je usvojen od strane mnogih koji nisu programeri, poput računovođa i znanstvenika, za izvršavanje različitih svakodnevnih zadataka. Python se često koristi za razvoj "back end" dijela web stranica ili aplikacija, onog dijela

koji korisnik ne vidi. Uloga Pythona u razvoju weba može uključivati slanje podataka sa i prema poslužiteljima, obradu podataka i komunikaciju sa bazama podataka, URL preusmjeravanje te pružanja sigurnosti korisničkih podataka. Python nudi nekoliko okvira (frameworks) za razvoj web aplikacija. Najčešće se koriste Django i Flask. A neke od popularnih rješenja koja su napravljena pomoću Python programskog jezika su Google, Meta, Vennmo, Spotify, Netflix i Dropbox. (Coursera authors, 2023)

4.1.5 Java

Java je programski jezik, dizajniran za stvaranje sadržaja za rani World Wide Web (Whitmore, 2023). Razvio se u jedan od najboljih i najcjelovitijih alata za stvaranje poslovnih web stranica, mobilnih aplikacija i drugih tehnologija softvera na strani poslužitelja i klijenta. Java je stvoren u tvrtki Sun Microsystems, Inc., gdje je James Gosling vodio tim istraživača u naporu da stvore novi jezik koji bi omogućio uređajima da međusobno komuniciraju (The Editors of Encyclopaedia Britannica, 2023). Rad na jeziku započeo je 1991., a uskoro se fokus tima preusmjerio u novi sektor, World Wide Web (The Editors of Encyclopaedia Britannica, 2023). Java je prvi put objavljena 1995., a Java-ina sposobnost pružanja interaktivnosti i multimedije pokazala je da je posebno pogodna za Web stranice i aplikacije (The Editors of Encyclopaedia Britannica, 2023). Java je programski jezik koji je objektno orijentiran, što znači da sve u Javi predstavlja "objekt". Ova karakteristika omogućuje fleksibilnost za stvaranje prilagođenog koda. Java koristi sintaksu koja je slična C++ programskom jeziku, što olakšava učenje web razvoja većini programera. Java je dobro organizirana, fleksibilna i moćna, a također je popularna, pa je lako pronaći resurse i pomoć za razvijanje u Javi.

Jedna od najvećih karakteristika Jave je njezina neovisnost o platformi. To znači da se Java kod može pokrenuti na bilo kojem uređaju, bez obzira na operativni sustav. Može se koristiti u „Front-end“ i u „Back-end“ razvoju, a mogućnosti upotrebe Jave u web razvoju gotovo su neograničene. Java se koristi za različite vrste usluga i dinamičkog sadržaja, uključujući online trgovine, tražilice, sustave za upravljanje sadržajem, igre, društvene mreže i oglašavanje. Java se može koristiti za razvoj mobilnih aplikacija. Java web programer ne mora uložiti dodatni rad da bi svoje web aplikacije učinio dostupnima na svim platformama. Java također nudi opsežnu standardnu biblioteku koja pruža alate za pomoć web programerima u uobičajenim zadacima poput unosa i ispisa, umrežavanja i kreiranja grafičkih i korisničkih sučelja.

Mnoge aplikacije se razvijaju koristeći ovaj popularni jezik. Među najpoznatijima su: Spotify, Twitter te Opera Mini (Bluein, 2020).

4.1.6 Ruby

Ruby je računalni programski jezik koji je razvio Yukihiro Matsumoto 1995. godine. Želio je stvoriti fleksibilan, objektno orijentiran jezik kojeg će programeri lakše koristiti.

Apple, GitHub, Hulu, ZenDesk, Basecamp, Airbnb i Urban Dictionary su web stranice razvijene pomoću Rubyja, što pokazuje njegovu sposobnost. Ruby je opće korišteni jezik koji je popularniji u industriji nego u znanosti ili akademiji. Velike tehnološke tvrtke širom svijeta koriste ga za izgradnju web aplikacija, proto tipiziranje, analizu podataka i drugo. Ruby se uglavnom koristi za izgradnju web aplikacija, ali ima i širu primjenu u drugim programerskim projektima. Osim što je koristan za razvoj web aplikacija, Ruby se često koristi za izgradnju aplikacija poslužitelja i obradu podataka. Također je popularan za web scraping, što uključuje prikupljanje podataka s web stranica, te crawling, što podrazumijeva automatsko pretraživanje i indeksiranje web sadržaja. Ruby ima jednostavnu i intuitivnu sintaksu koja olakšava programiranje i pisanje čistog i čitljivog koda. Zbog svoje fleksibilnosti i velike zajednice programera, Ruby je postao popularan jezik za različite vrste projekata.

Vodeći okvir za izvođenje Rubyja je Ruby on Rails, iako postoje i drugi okviri. Ruby on Rails je predstavljen 2004. godine i pojednostavio je korištenje jezika. Mnogi ljudi koji govore engleski smatraju da je lako naučiti i koristiti Ruby zbog njegove sličnosti s engleskim jezikom. Ruby program je besplatan i „open-source“, što omogućuje korisnicima da dijele ideje i poboljšanja. Ruby zajednica obično se fokusira na razvoj web aplikacija i stvorila je obiman skup programskih elemenata.

Jedan od nedostataka Rubyjevog pristupa prijateljskom korisničkom sučelju je da se greške mogu sakriti, što čini teže pronalaženje i rješavanje problema u kodu, uglavnom zato što dokumentacija za Ruby nije toliko kompletna kao za neke druge jezik (Coursera authors, 2023). Iako se Ruby on Rails može koristiti za različite namjene, neki projekti mogu se bolje ostvariti koristeći ovaj jezik nego drugi. Postoje područja u kojima je Ruby on Rails izuzetno učinkovit i primjeren, kao što su razvoj aplikacija za e-trgovinu, aplikacije nalik društvenim mrežama, SaaS projekti, aplikacije za prijenos uživo i platforme za vijesti, trgovanje i analizu podataka.

4.1.7 Swift

Swift je snažan i intuitivan programski jezik za iOS, iPadOS, macOS, tvOS i watchOS. Pisanje Swift koda je interaktivno, njegova sintaksa je sažeta, ali izražajna, te uključuje moderne značajke koje programeri vole. Swift kod je dizajniran da bude siguran i proizvodi softver koji radi izuzetno brzo. Nastao je kao rezultat najnovijih istraživanja u području programskih jezika, koja su kombinirana s desetljećima iskustva u izgradnji Apple platformi. API u Swiftu koriste imenovane parametre koji se izražavaju u čistoj sintaksi, što čini kod lakšim za čitanje i održavanje. Npr. u Swiftu se ne mora koristiti točku-zarez, što se ne rijetko zna zaboraviti u pisanju koda u drugim jezicima. Upravljanje memorijom je automatsko i koristi strogo determinističko prebrojavanje referenci, što održava uporabu memorije na minimumu bez prekomjernog opterećenja skupljanja smeća. Moguće je pisati istodobni kod pomoću jednostavnih ugrađenih ključnih riječi koje definiraju asinkrono ponašanje, čime se kod čini čitljivijim i manje sklon pogreškama.

Swift.org je otvorena platforma na kojoj se razvija Swift uz prisutnost izvornog koda, alata za praćenje grešaka, foruma i redovnih verzija za razvoj dostupnih svima. Ova široka zajednica programera, kako unutar tvrtke Apple, tako i stotine vanjskih suradnika, zajedno rade na tome da Swift bude još bolji (Developer authors, 2023). Postoji još širi raspon blogova, podcasta, konferencija i online sastanaka gdje programeri u zajednici dijele svoja iskustva kako bi ostvarili puni potencijal Swifta (Developer authors, 2023).

Iako nije čest izbor za web razvojne projekte, Swift ima nekoliko web okvira koji olakšavaju proces razvoja. Među tim okvirima su Vapor, Perfect i Kitura. Poznate aplikacije koje su izgrađene pomoću Swifta uključuju: Apple Inc, Uber, Lyft, IBM, Slack, WhatsApp, LinkedIn, Strava, Yahoo Weather i Firefox.

4.1.8 C#

„C sharp“ je objekrno orijentirani programski jezik kojeg je razvio Microsoft. Koristi se za razvoj različitih vrsta aplikacija, poput Windows desktop aplikacija, web aplikacija, mobilnih aplikacija, i gara, itd. Popularnost je stekao zbog svoje jednostavne sintakse, robusnih značajki i podrške za moderan programski model. Najznačajnija

osoba u njegovom razvoju je bio Anders Hejlsberg koji je u siječnju 1999. godine formirao tim kako bi razvili novi objektno orijentirani programski jezik pod nazivom Cool (C-like Object Oriented Language) (Kostanjevac, 2018). Microsoft je planirao zadržati ime „Cool“ kao ime jezika ali od toga su odustali zbog autorskih prava nad sličnim nazivom i zaštitnim znakom (Kostanjevac, 2018). Do vremena kada je .NET projekt bio javno najavljen u srpnju 2000. godine na profesionalnoj konferenciji developera (eng. Professional Developers Conference), jezik je bio preimenovan u C#. (Kostanjevac, 2018).

Cilj programskog jezika C# je bio olakšati učenje programiranja te omogućiti brži razvoj rješenja kako bi se natjecao s drugim programskim jezicima. Microsoft podržava alate i sustave koje nudi ovaj programski jezik. Te je razvijen je s ciljem da podrži ažurirane funkcionalnosti za različite softverske aktivnosti i omogući veću produktivnost u web aplikacijama. C# je fokusiran na efikasno korištenje memorije i smanjenje potrebe za energijom. Njegov cilj je podržati velike sustave koji rade na izvršavanju više zadataka istovremeno. ASP.NET je alat koji se koristi za kreiranje web aplikacija u programskom jeziku C#. Ovaj alat omogućava izradu web stranica i aplikacija koje su podržane na Windows serverima. C# se temelji na reaktivnom programiranju, što znači da se programi pišu tako da reagiraju na događaje koje korisnik stvara, kao što su klikovi mišem, pritisci na tipkovnicu, dodirni zasloni i druge geste koje se koriste na računalima, pametnim telefonima i tabletima. Osim toga, C# omogućuje asinkrono programiranje, što znači da se više zadataka može obavljati istovremeno, što čini aplikacije reaktivnijima na korisnikovu interakciju.

Popularne web stranice i web aplikacije koje koriste C# kao programski jezik su: XBOX, Skype, Internet Explorer, Visual Studio, SQL Server, Paintbrush, Microsoft Office softver.

4.2 Okvir za razvoj web aplikacija

Web okviri (ili okviri za razvoj web aplikacija, eng. „Frameworks“) su unaprijed izgrađene sekcije koda koje omogućuju programerima da brzo ubace neke od najčešćih komponenti u razvoju web stranica i aplikacija (Wintemute & Blackmon, 2023). Ovi alati sadrže funkcionalnosti predložaka koji vam omogućuju prikazivanje informacija unutar preglednika. Također pružaju okruženje za upravljanje protokom

informacija putem skriptiranja te nudi niz aplikacijskih programskih sučelja (API-ja) za pristup podacima iz baze podataka. Većina okvira također uključuje alate kojima web-developeri mogu izgraditi sustav za upravljanje sadržajem (CMS) za upravljanje digitalnim informacijama na web stranicama i Internetu. Najznačajnije prednosti za web-developere koji koriste web okvir proizlaze iz činjenice da je učinkovit, ima dobar nivo podrške, visoku razinu sigurnosti te uključuje mogućnost integracije (Greene & Greene, 2023).

Web okviri su iznimno ekonomični kako za programere tako i za klijente jer su otvorenog koda. Njihova besplatna dostupnost ne umanjuje njihovu kvalitetu. Još jedna važna prednost za web-developere je učinkovitost. Web okviri eliminiraju potrebu za pisanjem ponavljajućeg koda, što omogućuje brže izgradnju web stranica i aplikacija. Osim toga, web okvir dolazi s podrškom tima i naprednim sigurnosnim značajkama, što znači da možete biti sigurni da će stručnjaci rješavati eventualne probleme. Jedna od najkorisnijih značajki web okvira je integracija koja omogućuje povezivanje drugih alata, poput baza podataka, s okvirom.

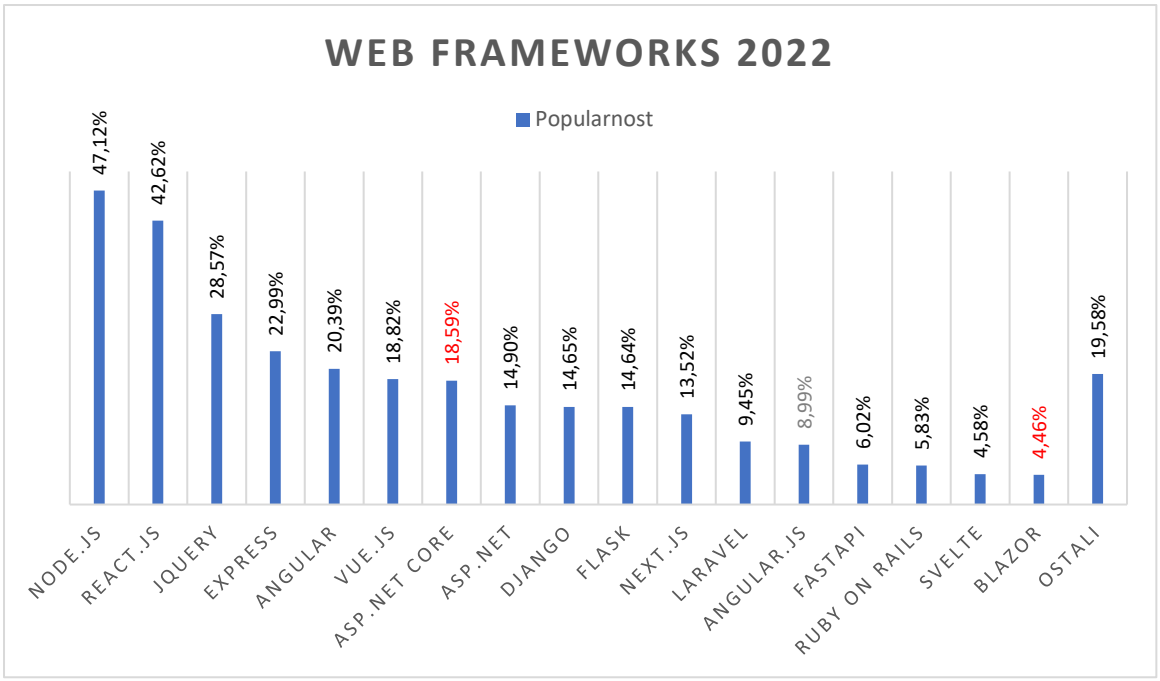
Ipak, treba napomenuti da programeri često ističu najveće nedostatke korištenja web okvira, koji se uglavnom odnose na ograničenja u promjeni samog okvira. Paradigme programiranja i dizajn su vrlo restriktivni, što može izazvati frustraciju kod web-developera. Drugi nedostatak je da neki programeri smatraju da je teško naučiti jezik koji stoji iza web okvira. Nerijetko se web-developeri usredotoče na učenje samog okvira, bez potpunog razumijevanja programskog jezika koji je njemu temelj, što može otežati potpuno razumijevanje web okvira u cjelini.

Također razlikujemo takozvane „Front-end“ i „Baack-end“ web okvire. Ili kako ih još nazivamo klijentski web okviri i server web okviri. Oba okvira imaju zadatak ubrzat i poboljšat programerima razvoj web rješenja. Svi ti web okviri rade na nekim programskim jezicima pa tako za „front-end“ se koriste uz jezike klasične za web poput HTML i CSS još i JavaScript, JQuery pa čak i C#. Dok „Back.end“ web okviri koriste jezike poput JavaScripta (uz node.js), Python, PHP, Ruby i .NET. Neki od najpoznatijih web okvira za „Front-end“ su React, Ember, Vue.js, Angular, Svelite a dok je za „Back-end“ se ističu: Node.js, Express, ASP.NET Core, Django, Flask, Laravel itd.

Kao što i to sami nazivi okvira web „Front-end“ okviri su orijentirani izgledu, tj. Dizajnu samih aplikacija, SEO optimizaciji i performansama aplikacije, dok „Back-end“

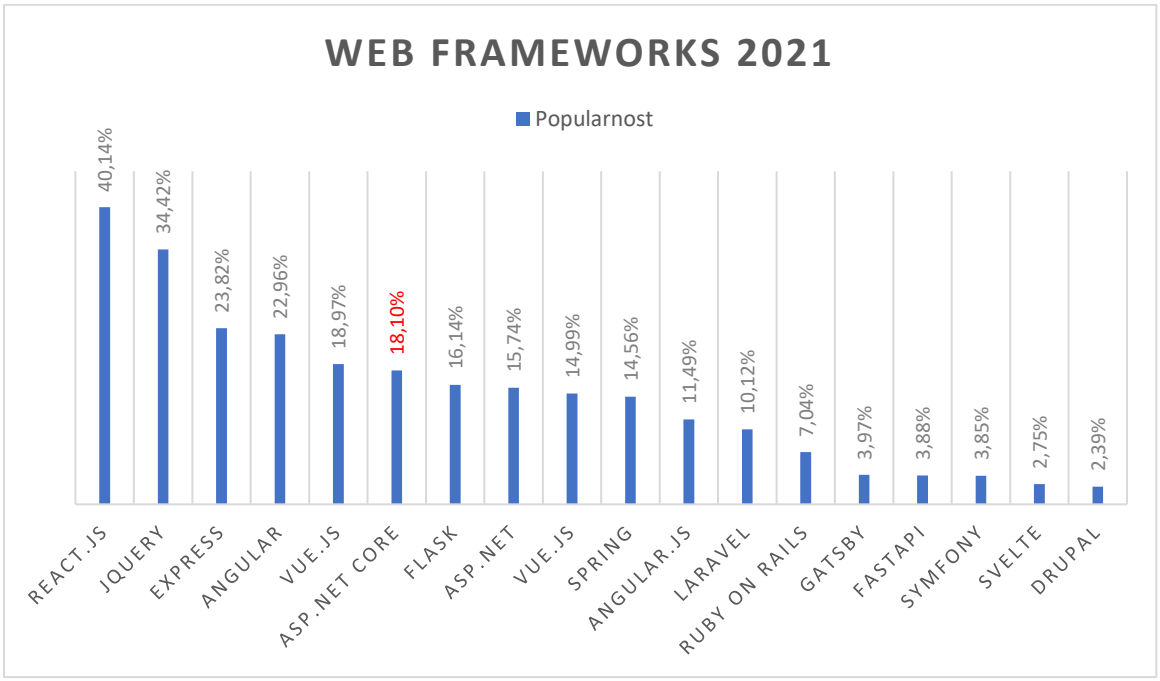
okviri su zaduženi za ono što ne vidimo kao korisnik aplikacije, odnosno zadužen je za upravljanje bazom podataka, sigurnošću, usmjeravanje pomoću URL, odnosno cijela struktura i funkcionalnosti aplikacije kojom se korisnik služi.

Na slici 4 vidimo koji su najpopularniji web okviri na godišnjem ispitivanju jednog od najpoznatijeg foruma za rješavanje programerskih problema, Stack Owerflow. Ovakvo istraživanje se provodi svake godine, a cilj mu je upznati svoju publiku i saznati što programeri najviše koriste i tko su oni. Svake godine prikupe od 60 000 do 70 000 korisnika koji su voljni ispuniti njihovu anketu, tako da možemo reći da ovo istraživanje ima dobru podršku i realan prikaz stanja na „tržištu“. Primjećuje se kako većina programera koristi i traži savjete u vezi Node.js, što ukazuje na to da je Node.js najraširenija "Back-end" solucija za web aplikacije. S druge strane, drugi najpopularniji web okvir je "Front-end" web okvir, ato je React.js. To nam otkriva da većina web aplikacija trenutno koristi Node.js za svoj "Back-end" dok za "Front-end" koriste React.js. Također, istaknuta su dva web okvira koja su koristili za kreiranje web aplikacije kao primjera u ovom radu. Primjećuje se da je "Back-end" web okvir koji su koristili nešto popularniji od "Front-end" okvira. Razlog tome je što se smatra da Microsoft, vlasnik oba okvira, ne provodi dovoljno marketinga za svoj "Front-end" okvir koji je relativno nov u usporedbi s Reactom i drugim okvirima, što također utječe na njegovu popularnost. Unatoč tome, Blazor web okvir pokazuje dobar trend jer Blazor nije bio naveden na popisu popularnosti u ispitivanju Stack Overflowa 2021 (Slika 5).



Slika 5: Tablica popularnosti po web okvirima prema "Stack Overflow-u"

Izvor: Stack Overflow: https://survey.stackoverflow.co/2022/#technology-most-popular-technologies



Slika 6: Tablica popularnosti po web okvirima prema "Stack Overflow-u"

Izvor: Podatci sa: https://insights.stackoverflow.com/survey/2021#most-popular-technologies-webframe

4.2.1 React

ReactJS je jedna od najpopularnijih JavaScript web okvira za razvoj mobilnih i web aplikacija. React je razvijen od strane Jordana Walkea, softverskog inženjera u Meta kompaniji, koji je prvo objavio rani prototip React-a pod nazivom "FaxJS". Prvi put je React implementiran na Facebook-ovom „News Feedu“ 2011. godine, a kasnije se proširio na Instagram 2012. godine. Nakon toga, React je postao „open source“ projekt na konferenciji JSConf US u svibnju 2013. godine. Kreirao ju je Facebook, a React sadrži kolekciju ponovno upotrebljivih JavaScript kodova koji se koriste za izgradnju korisničkog sučelja (UI) nazvanih komponente.

Samo s osnovnim poznavanjem JavaScript-a, programeri mogu brzo naučiti koristiti React jer se temelji na čistom JavaScriptu i komponentnom pristupu.

Nakon stjecanja osnovnog razumijevanja JavaScripta, preporučljivo je fokusirati se na proučavanje ReactJS-a kako bi se pojednostavnio proces razvoja korisničkog sučelja. Kroz upotrebu React-a, moguće je iskoristiti komponente koje su već razvijene u drugim aplikacijama. Zahvaljujući otvorenosti koda ReactJS-a, moguće je unaprijed izgraditi ove komponente, čime se znatno smanjuje vrijeme potrebno za razvoj složenih web aplikacija. React Native je JavaScript okvir otvorenog koda koji je izgrađen na temelju React biblioteke. Programeri ga koriste za stvaranje React aplikacija koje rade na iOS i Android platformama. React Native koristi izvorne ili nativne Application Programming Interfaces (API-je) kako bi renderirao mobilne komponente korisničkog sučelja pomoću Objective-C (za iOS) ili Java (za Android). Zbog toga programeri mogu kreirati komponente koje su specifične za svaku platformu i dijeliti izvorni kod između različitih platformi.

Integracija JSX-a olakšava pisanje React komponenti - korisnici mogu kombinirati JavaScript objekte s HTML tipografijom i oznakama. JSX također pojednostavljuje višestruko renderiranje funkcija, čime se održava jednostavnost koda, ali ne smanjuje mogućnosti aplikacije. JSX je ekstenzija JavaScripta koja omogućuje pisanje HTML koda u JavaScript datoteku.

Službeni alat za naredbeni redak (CLI) Reacta, nazvan Create React App, dodatno pojednostavljuje razvoj jednostraničnih aplikacija. Ovaj alat ima moderni proces postavljanja izgradnje s prethodno konfiguriranim alatima i izvrsno je za učenje ReactJS-a (A., 2023). Virtual DOM omogućava ReactJS-u da ažurira stablo DOM-a

na najefikasniji način. Virtual DOM-a jedan je od faktora koji utječu na brže učitavanje stranica. React pruža podršku pretraživačima u navigaciji web aplikacija putem izvršavanja ili generiranja sadržaja na serveru te potom šalje korisniku gotovu HTML stranicu. Ovo rješava jedan od glavnih problema s kojima se susreću web stranice koje koriste velike količine JavaScripta, jer takve stranice često predstavljaju izazov za pretraživače i vremenski su zahtjevnije za pretraživanje.

Stotine vodećih svjetskih tvrtki poput Netflix-a, Airbnb-a i American Express-a koriste React za izgradnju svojih web aplikacija.

4.2.2 Vue.Js

Vue je JavaScript web okvir za izgradnju korisničkih aplikacija (Brewster, 2022). Temelji se na standardnom HTML-u, CSS-u i JavaScriptu te pruža deklarativni i komponentni model programiranja koji pomaže u učinkovitom razvoju korisničkih aplikacije, bilo da su jednostavna ili složena (Tutorials Point Authors, 2023). Započeti s korištenjem VueJS-a je vrlo jednostavno. Svaki programer sa osnovnim znanjem JavaScripta može brzo razumjeti i izgraditi interaktivna web aplikacije. Osnovna biblioteka Vue.js također se temelji na CSS-u, HTML-u i JavaScriptu - bitnim za svaki razvojni projekt na webu. VueJS je razvio Evan You, bivši zaposlenik Googlea. Prva verzija VueJS-a objavljena je u veljači 2014. Sada je vrlo popularan, s preko 64,828 zvjezdica na GitHubu (Tutorials Point Authors, 2023).

Vue.js se ističe svojim laganim dizajnom, s malom veličinom paketa od samo 21 KB. Ta kompaktnost mu pruža prednost u performansama u odnosu na konkurenciju, omogućujući brže izvršavanje. Važno je spomenuti korištenje virtualnog DOM-a, koji ubrzava proces renderiranja, rezultirajući brzim i učinkovitim ažuriranjem korisničkog sučelja. Vue komponente mogu se kreirati koristeći dva različita API stila: Options API i Composition API. VueJS, poput drugih okvira poput Reacta i Embra, koristi virtualni DOM. Umjesto izravnih promjena na DOM-u, stvara se kopija DOM-a koja se pohranjuje kao JavaScript podaci. Kada se naprave promjene, one se primjenjuju na kopiju JavaScript podataka, a zatim se ta struktura uspoređuje s izvornim podacima. Konačne promjene se ažuriraju na stvarnom DOM-u, što rezultira promjenama koje korisnik vidi na web stranici. Ova metoda je korisna u smislu

optimizacije jer je manje resursno zahtjevna i promjene se mogu brže primijeniti. Vue.js je razvijen s naglaskom na izradu prototipova.

Evan You je pronašao inspiraciju za stvaranje Vue.js-a dok je radio na izradi prototipova unutar preglednika dok je bio zaposlen u Googleu. Može biti izuzetno koristan za dodavanje funkcionalnosti postojećim aplikacijama. Zahvaljujući svojoj maloj veličini i oslanjanju na JavaScript, integracija s bilo kojim postojećim JavaScript projektom trebala bi biti relativno jednostavna. Također treba napomenuti da je Vue.js kompatibilan s mnogim „Back-end“ tehnologijama poput PHP-ovog Laravela, Express.js-a, Python-ovog Djangoa i Ruby on Rails-a. Poznate tvrtke koje koriste Vue.js su: Trustpilot, Trivago, Accenture i Statista.

4.2.3 Angular

Angular je otvoreni web okvir za aplikacije temeljen na TypeScript-u. On omogućuje različite aktivnosti u razvoju weba i unapređuje aplikacije ili web stranice s potrebnim funkcionalnostima. Podržava različite platforme poput mobilnih uređaja, weba i desktopa.

Angular je robustan okvir za izradu "Front-end" aplikacija koji pruža sveobuhvatna rješenja za razvoj sofisticiranog poslovnog softvera. Ako je projekt složena web aplikacija s mnogo značajki, tada je Angular najbolji izbor za takav projekt. Razlozi zbog kojih programeri i tvrtke odabiru Angular su: čist kod, brzo testiranje, dobra mogućnost otklanjanja pogrešaka i poboljšani CSS stilovi. Također je važno napomenuti da Angular ima skup dobro integriranih biblioteka. One pružaju mnogo mogućnosti za stvaranje funkcionalnosti, uključujući alate za komunikaciju između klijenta i poslužitelja, upravljanje usmjeravanjem po web aplikaciji i upravljanje obrascima.

On je rješenje za web razvoj projekata koji zahtijevaju brzo i efikasno skaliranje. Svojom komponentnom arhitekturom i bogatim izborom alata za razvoj omogućuje jednostavno ažuriranje, proširivanje funkcionalnosti i visoku performansu. Angular se koristi za izradu foruma, poslovnih portala, platformi za grupno financiranje, tržišta i popisa nekretnina. Neke od poznatih tvrtki koje koriste Angular su: GitHub, BMW(kalkulator cijena, pronalazač ponuda), Nike, YoutubeTV, Firebase, i Adobe (Dyachenko, 2022). AngularJS je okvir, koji pruža strukturu za dinamičke web

aplikacije. On proširuje HTML s dodatnim atributima kako bi aplikacije bile osjetljivije na korisnike. Također ubrzava razvoj korisničkog sučelja i koristi se za izradu jednostraničnih aplikacija (SPA) ili progresivnih web aplikacija (PWA). Značajne tvrtke koje koriste AngularJS na svojim web stranicama: NBC, Intel, ABC News, Sprint, Walgreens, Google, Amazon, Udemy, Snapchat (Dyachenko, 2022).

Angular ima izuzetnu funkcionalnost. Ne moraju se koristiti dodatne biblioteke kako bi se stvorila osnovna funkcionalnost za aplikaciju. To osigurava veću sigurnost i poboljšanu kvalitetu koda. Angular komponente su neovisne jedinice koje se mogu ponovno koristiti u različitim dijelovima aplikacije. Ove komponente mogu biti forme, polja za pretraživanje, kalendari i druge. Omogućuje programerima da osiguraju održavanje proizvoda bez dodatnog napora budući da je dobro odvojene komponente lako ažurirati, poboljšati ili zamijeniti. Izvanredan je okvir za razvoj proizvoda u različitim područjima i tržištima. Bez obzira radi li se o aplikacijama za putovanja, platformama za osiguranje ili online bankarstvu, programeri imaju obilje alata koji im omogućuju brzo pretvaranje ideja u stvarnost, a korisnici dobivaju visokokvalitetno korisničko sučelje. Angular podržava Google, što znači da su sva njegova ažuriranja i verzije kompatibilne s proizvodima tvrtke. Osim toga, okvir ima veliku zajednicu koja pomaže u brzom rješavanju bilo kakvih pitanja i prevladavanju izazova.

4.2.4 Ruby on rails

Okvir Ruby on Rails (RoR) proizlazi iz programskog jezika Ruby. Povijest razvoja Ruby on Rails web aplikacija započinje s Davidom Heinemeier Hanssonom koji je radio na alatu za upravljanje projektima nazvanom Basecamp. On je poboljšao Ruby kod i stvorio okvir koji je danas poznat kao Ruby on Rails (RoR) (Engine Yard Team, 2023).

Ruby on Rails ne zahtijeva od programera da deklarira sitnice poput tipova varijabli. Umjesto toga, okvir podržava snažne konvencije imenovanja. Prvo slovo varijable određuje njezinu namjeravanu upotrebu. Budući da je RoR napisan u Rubyju, okvir slijedi iste konvencije imenovanja, što olakšava smanjivanje koda. Jedna od najvećih prednosti RoR-a je brzina. RoR podržava brzi razvoj aplikacija (RAD), što omogućuje brže kreiranje funkcionalnih aplikacija. Efikasan obrazac „Model View Controller“ (MVC) u RoR-u smanjuje potrebu za ponavljanjem posla. RoR u potpunosti

iskorištava ovu prednost tako što se fokusira na brzu izradu prototipova i iteracije za nove funkcionalnosti. Ovo rezultira manjim brojem grešaka, poboljšanom prilagodljivošću i fleksibilnošću, te intuitivnijim kodom. Ruby on Rails je otvoreni serverski web okvir što znači da postoji puno biblioteka.

Za razvoj u RoR-u koristi se RubyGems, alat za upravljanje paketima sličan npm-u koji se koristi u JavaScript projektima, kao što je instalacija Node.js-a. Važno je napomenuti da ROR zajednica može unaprijediti neke od ovih „open-source“ alata, čineći ih još dostupnijima i korisnijima, a na GitHubu postoji obilje podrške zajednice.

Zahvaljujući MVC-u, RoR programerima je postalo izuzetno vremenski učinkovito kreirati i održavati web aplikacije. „Model view controller“ ima tri sloja: sloj koji radi s logikom povezanom s podacima poput ažuriranja i prijenosa podataka. Može se zamisliti kao „Back-end“ koji radi s poslovnom logikom. Slijedi View, što je samo „Front-end“. To mogu biti stvari poput HTML datoteka, PDF-a, XML-a, RSS-a, u osnovi bilo što s čime korisnik vizualno intrigira. I na kraju Controller, koji pomaže View-u i Modelu da komuniciraju jedan s drugim. Može se zamisliti kao posrednik između „Back-end“ i „Front-end“. Web okvir RoR ima mnogo vodiča na svojoj stranici o stvaranju boljeg i učinkovitijeg koda.

Određena područja u kojima se ističe su: razvoj aplikacija za e-trgovinu, aplikacije slične društvenim mrežama, SaaS projekti, aplikacije za prijenos uživo te platforme za vijesti, trgovanje i analizu podataka. Postoji mnoštvo web stranica koje koriste RoR kao web okvir na strani poslužitelja za svoj „Back-end“. Neke od njih su: Github, Airbnb, Shopify, Basecamp.

4.2.5 JQuery

JQuery je JavaScript biblioteka koja pomaže pojednostaviti i standardizirati interakcije između JavaScript koda i HTML elemenata (Melville, 2018). JavaScript omogućava interaktivnost i dinamičnost web stranica, a JQuery je alat koji olakšava taj proces (Melville, 2018). Iako JQuery ima mnogo korisnih funkcionalnosti i može biti ključan dio web razvojnog procesa, on se smatra bibliotekom, a ne web okvirom.

Web developer John Resig predstavio je JQuery 2006. godine. JQuery biblioteka je razvijena kako bi se prevladale razlike u implementaciji JavaScripta u

preglednicima i pomogla programerima da napišu manje koda dok obavljaju zadatke kao što su: manipulacija HTML elementima na web stranici, dinamička promjena CSS-a, reagiranje na događaje kao što su klikovi mišem i pritisci tipki te obrada Ajax zahtjeva za ažuriranje sadržaja stranice bez ponovnog učitavanja. JQuery nudi selektore koji omogućuju dohvaćanje elemenata iz DOM-a temeljem različitih kriterija poput naziva oznake, ID-a, naziva CSS klase, naziva atributa, vrijednosti, pozicije u hijerarhiji i drugih. Dostupni su posebni efekti na DOM elemente, poput prikazivanja ili skrivanja elemenata, promjene vidljivosti, klizanja, animacija i slično. Pruža obilan skup značajki koje povećavaju produktivnost programera omogućujući im da napišu manje koda koji je lako čitljiv. Te pruža izvrsnu podršku za različite preglednike bez potrebe za pisanjem dodatnog koda.

U usporedbi s drugim JavaScript bibliotekama i čistim JavaScriptom, JQuery je puno jednostavniji za korištenje. Njegova sintaksa je jednostavna, a za postizanje istih rezultata koristi se puno manje linija koda.

4.2.6 Django

Django je Python web okvir koji omogućuje brz razvoj sigurnih web stranica visoke razine održivosti. Django je nastao između 2003. i 2005. godine kao rezultat rada web tima zaduženog za izradu i održavanje novinskih web stranica. Kroz razvoj tih web stranica, tim je primijetio ponavljanje i mogućnost ponovne uporabe mnogih zajedničkih kodova i obrazaca dizajna. Taj zajednički kod je postupno evoluirao u generički okvir za web razvoj, koji je u srpnju 2005. godine službeno pokrenut kao Django projekt.

Dizajniran je sa ciljem da ubrza proces razvoja aplikacija, omogućujući programerima da brzo prevedu svoje koncepte u završene aplikacije. Nudi širok spektar dodataka koji olakšavaju rješavanje uobičajenih zadataka u web razvoju. S ovim okvirom, može se iskoristiti desetke dostupnih dodataka koji se odnose na autentifikaciju korisnika, upravljanje sadržajem, mapiranje web stranica, generiranje RSS feed-ova i mnoge druge zadatke - sve to već u osnovnoj instalaciji. Django ima ozbiljan pristup sigurnosti i pomaže programerima da izbjegnu mnoge uobičajene sigurnosne propuste poput SQL injection-a, cross-site scripting-a, cross-site request forgery-ja i clickjacking-a. Ovaj okvir ima ugrađene mehanizme koji provjeravaju

autentičnost korisnika, pružajući siguran način za upravljanje korisničkim računima i lozinkama.

Django okvir se temelji na Pythonu, koji je dinamičan programski jezik visoke razine. Ima fleksibilan i dobro strukturiran administracijski panel. Django se temelji na principu "Don't Repeat Yourself" (DRY), što omogućuje programerima jednostavno postavljanje apstrakcija umjesto ponavljanja uobičajenih softverskih obrazaca ili primjenu normalizacije podataka. Ova praksa omogućuje izbjegavanje enkripcije i potencijalnih grešaka na jedinstven način. Čak i koristeći stariju verziju okvira, redovito se objavljuju sigurnosna ažuriranja koja popravljaju greške i poboljšavaju sigurnost. Django ima LTS (Long-term Support) verziju. Među glavnim prednostima korištenja Djanga za „Back-end“ web razvoj je njegov „Representational State Transfer“ (REST) okvir, koji je popularan skup alata za izgradnju API-ja (Daftari, 2022). Snaga Django-vog REST okvira može se procijeniti iz činjenice da su potrebne samo tri linije koda za izgradnju API-ja spremnog za korištenje (Daftari, 2022).

Osigurava sigurno upravljanje korisničkim računima i lozinkama putem izbjegavanja uobičajenih pogrešaka, kao što je skladištenje podataka o sesiji u kolačićima koji su ranjivi. Umjesto toga, Django koristi kolačiće samo za pohranu ključeva, dok su stvarni podaci sigurno pohranjeni u bazi podataka. Besplatan je i otvorenog koda, ima uspješnu i aktivnu zajednicu, izvrsnu dokumentaciju i mnogo opcija za besplatnu i plaćenu podršku.

4.2.7 Node.JS

Node.js je serverska platforma temeljena na JavaScript alatima u Google Chromeu. Napravio ju je Ryan Dahl 2009. godine, a najnovija verzija je v0.10.36 (Ravikiran, 2023). On podržava razvoj serverskih i mrežnih aplikacija otvorenog koda na više platformi. Node.js koristi JavaScript kao jezik programiranja, a podržan je na OS X-u, Microsoft Windows-u i Linux-u. Node.js dolazi s velikom bibliotekom JavaScript modula, što olakšava izradu web aplikacija u Node.js-u.

Node.js se može koristiti i na „Front-end“ i na „Back-end“. Korištenje Node.js-a ima glavnu svrhu - pružanje brzine. Ovaj web okvir omogućuje brzu obradu podataka, interakciju između klijenta i poslužitelja te ubrzani razvoj. Node.js omogućuje dvosmjerne veze usmjerene na događaje između klijenta i poslužitelja, što omogućuje

iniciranje komunikacije i razmjenu podataka. Node.js „Back-end“ prikazivanje pruža izuzetnu vidljivost web stranicama, omogućavajući učinkovitu optimizaciju React projekata od strane pretraživača. Zbog visoke brzine i performansi koje pruža, Node.js se smatra najboljim okvirom za unapređenje SEO[2] strategija. Web aplikacije razvijene pomoću Node.js-a jednostavne su za implementaciju. Ovaj okvir omogućuje programerima relativno jednostavno dovršavanje i testiranje svih komponenti web aplikacije. Mnoge agencije za razvoj progresivnih web aplikacija smatraju ovaj okvir vrlo korisnim jer pomaže ubrzati konačnu implementaciju web aplikacija.

Zbog svog jedinstvenog pristupa I/O operacijama, Node.js se ističe u situacijama koje zahtijevaju skalabilnost i visoke performanse u stvarnom vremenu na web poslužiteljima. Jedna od mana Node.js-a je nedostatak dosljednosti. API za Node.js se često mijenja, a ažuriranja često nisu kompatibilna s prethodnim verzijama. Kao rezultat, programeri su prisiljeni prilagođavati stari kod kako bi bio kompatibilan s novim verzijama. Node.js je trenutno vrlo tražen i sve popularniji. Kompanije poput Netflix-a, Uber-a, PayPal-a, Twitter-a i mnoge druge poznate tvrtke aktivno koriste Node.js.

4.3 Microsoft-ova web rješenja

Microsoftova web rješenja omogućuju programerima izgradnju visokokvalitetnih web aplikacija, pružajući im alate, platforme i infrastrukturu potrebnu za razvoj, implementaciju i upravljanje aplikacijama u oblaku. Ove tehnologije se kontinuirano razvijaju i nadograđuju kako bi podržale najnovije trendove i zahtjeve u svijetu web razvoja. Neke od najpoznatijih tehnologija su: ASP.NET, Azure, Visual Studio, Visual Studio Code, Microsoft 365, Blazor te TypeScript.

4.3.1 .Net ekosustav

.NET ekosustav je razvijen kako bi programerima pružio moćan skup alata, tehnologija i okvira za razvoj različitih vrsta aplikacija. Prvi put je predstavljen 2002. godine kao .NET Framework, platforma za razvoj Windows aplikacija. To je uključivalo Common Language Runtime (CLR) koji je omogućio višejezičnu podršku. .NET

[2] SEO (eng. „Search Engine Optimization“) je proces unaprjeđenja web stranica i aplikacija kako bi bili čitljiviji pretraživačima poput Google i Microsoft Bing

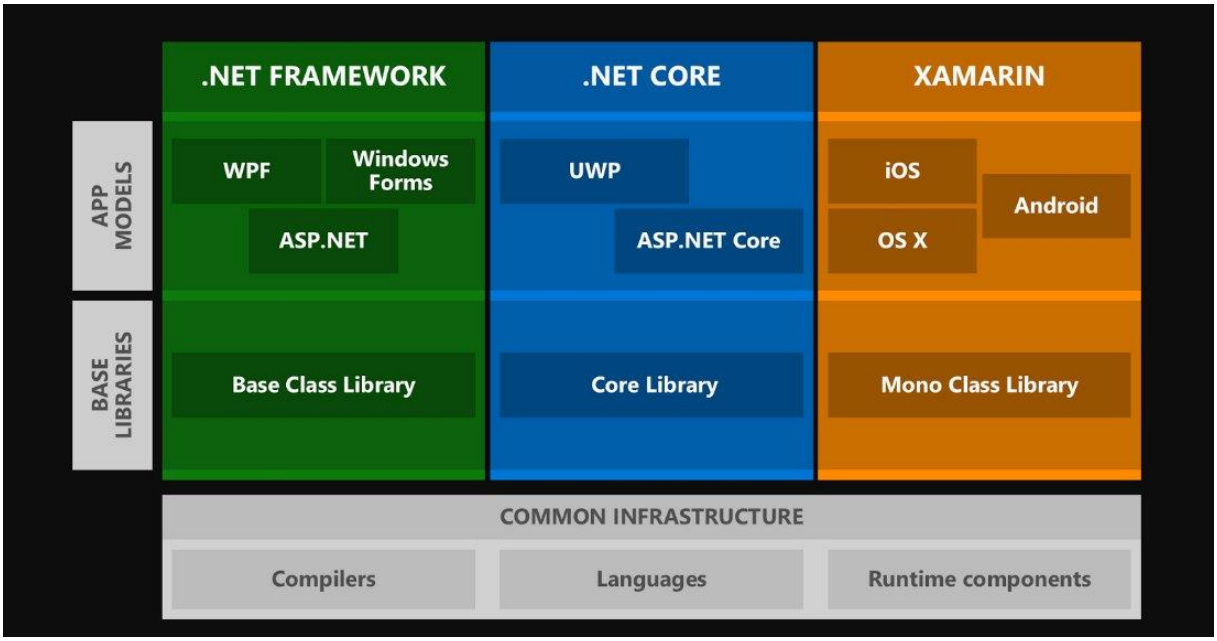
ekosustav omogućava programerima da razvijaju aplikacije koje rade na različitim operativnim sustavima, uključujući Windows, Linux i macOS. To je postignuto putem .NET Core (sada .NET) verzije, što je omogućilo višeplatformski razvoj. Koristi se za razvoj raznovrsnih aplikacija, uključujući web stranice, web aplikacije, desktop aplikacije, mobilne aplikacije, cloud aplikacije i aplikacije za Internet stvari (IoT). To znači da programeri mogu koristiti .NET za gotovo svaku vrstu softverskog projekta. Također, podržava i različite programerske jezike, pri čemu je najpopularniji C#. Osim toga, postoje i drugi jezici kao što su F# i Visual Basic. .NET također nudi različite okvire za razvoj aplikacija, kao što su ASP.NET za web aplikacije, Xamarin za mobilne aplikacije i Windows Presentation Foundation (WPF) za desktop aplikacije.

Ključne komponente .NET ekosustava su:

- Common Language Runtime (CLR): CLR je srce .NET-a. To je komponenta koja omogućava izvođenje .NET koda, upravljanje memorijom i upravljanje iznimkama. CLR omogućava interoperabilnost različitih jezika.
- .NET Class Library: Ova knjižnica sadrži većinu osnovnih klasa i funkcija koje programeri koriste pri izgradnji aplikacija. To uključuje rad s nizovima, datotekama, mrežom i mnoge druge funkcionalnosti.
- Visual Studio: To je integrirano razvojno okruženje (IDE) koje programerima omogućava izradu, testiranje i održavanje .NET aplikacija. Visual Studio nudi bogat set alata za razvoj, od rješavanja grešaka do razvoja korisničkog sučelja.
- ASP.NET i ASP.NET Core: Ovi okviri koriste se za izradu web aplikacija i web servisa. ASP.NET Core je posebno važan za razvoj moderne, brze i sigurne web aplikacije.
- Xamarin: Ovaj okvir omogućava razvoj mobilnih aplikacija za različite platforme, uključujući iOS, Android i Windows, koristeći C# i .NET.

Danas .NET ekosustav čini .NET 5, .NET 6, .NET 7 i budući .NET 8. Ove verzije podržavaju širok spektar aplikacija, uključujući web, desktop, mobilne, cloud i IoT aplikacije. Razvojni programeri mogu koristiti različite jezike poput C#, F# i Visual Basic kako bi izgradili aplikacije za različite platforme. Pored toga, .NET Core (sada .NET 7) i Visual Studio Code postali su popularni alati za razvoj aplikacija na različitim

platformama, uključujući Linux i macOS. Microsoft također nudi Azure, svoj cloud servis, koji je duboko integriran s .NET ekosustavom.



Slika 7: Prikaz .Net ekosustava

Izvor: A Brief Walk Through .Net Ecosystem: https://milan.milanovic.org/post/a-brief-walk-through-net-ecosystem/

4.3.2 .Net Framework

.NET Framework je originalna implementacija .NET-a. Podržava pokretanje web stranica, usluga, aplikacija za radnu površinu i više na sustavu Windows. (Microsoft, 2022) .NET Framework je razvojni okvir koji je razvio Microsoft s ciljem olakšavanja kreiranja i izvođenja različitih vrsta aplikacija na Windows platformi. Ovaj okvir pruža opsežnu biblioteku i funkcionalnosti za razvoj raznolikih aplikacija, uključujući aplikacije za desktop, web aplikacije i usluge, kao i aplikacije za uređaje poput pametnih telefona i tableta.

Jedna od ključnih komponenti .NET Frameworka je Zajedničko jezično izvršno okruženje (CLR), virtualna mašina koja omogućava izvođenje programa napisanih u različitim programskim jezicima podržanim unutar okvira. To omogućava programerima da koriste jezik s kojim su najugodniji, dok se aplikacije ipak izvršavaju na istoj CLR platformi.

4.3.3 .Net Core

.NET je implementacija prenosiva na više platformi za pokretanje web stranica, usluga i aplikacija za konzolu na sustavima Windows, Linux i macOS. .NET je otvorenog koda na GitHubu. Prethodno se nazivao .NET Core. (Microsoft, 2022). .NET Core je dobio naziv "Core" zato što obuhvaća osnovne značajke koje su prisutne u .NET okviru. Glavni cilj .NET Core-a je omogućiti otvoreno kodnu i kompatibilnost sa više platformi .NET okvira kako bi se mogao koristiti u okruženjima s ograničenim resursima. To uključuje minimalni set značajki potrebnih za pokretanje osnovne aplikacije u .NET Core-u, dok se druge napredne značajke mogu učitati pomoću paketa iz „NuGeta“.

Međutim, važno je napomenuti da je .NET Core, kasnije preimenovan u .NET 5, pa potom u .NET 6 i .NET 7, postao nasljednik originalnog .NET Frameworka. Ova nova verzija pruža višu razinu performansi, bolju modularnost i podršku za više platformi, uključujući Windows, Linux i macOS. Osim toga, .NET Core uključuje novi model aplikacija nazvan "ASP.NET Core" za razvoj web aplikacija.

Prednosti .NET Core u odnosu na „.NET Framework“ su:

1. Više platformska podrška: .NET Core radi na različitim operacijskim sustavima kao što su Windows, macOS i Linux, omogućavajući razvoj i izvršavanje aplikacija na različitim platformama.
2. Poboljšane performanse: .NET Core je optimiziran za visoke performanse, pružajući brže izvršavanje aplikacija i bolje iskorištavanje resursa sustava.
3. Usporedna verzija: .NET Core omogućuje različite verzije okvira da koegzistiraju na istom računalu, što olakšava migraciju postojećih aplikacija i održavanje više verzija okvira.
4. Novi API: .NET Core donosi nove API-je i poboljšane značajke koje nisu prisutne u .NET okviru, omogućavajući programerima da koriste najnovije tehnologije i funkcionalnosti.
5. Otvoren kod: .NET Core je potpuno otvorenog koda, što znači da je dostupan i pristupačan širokoj zajednici razvojnih programera. To potiče suradnju, inovacije i transparentnost u razvoju okvira.

4.3.4 ASP.NET Core

ASP.NET Core je napredni, otvoreno kodni okvir visokih performansi koji podržava razvoj na više platformi kojeg je razvio Microsoft. Omogućuje izgradnju modernih aplikacija koje su spremne za rad u oblaku i na internetu.

Programeri su koristili ili trenutno koriste ASP.NET 4.x za razvoj web aplikacija. Međutim, s pojavom ASP.NET Core-a, dogodila se revizija ASP.NET 4.x, s arhitektonskim promjenama koje su rezultirale tanjim i modularnijim okvirom. (Roth, et al., 2022). Ovaj okvir pruža integraciju s popularnim klijentskim okvirima i bibliotekama poput Blazora, Angulara, Reacta i Bootstrapa. ASP.NET Core se sastoji od biblioteka koje su napisane u .NET Standard formatu. To znači da biblioteke napisane s .NET Standard 2.0 mogu raditi na bilo kojoj .NET platformi koja podržava .NET Standard 2.0.

4.3.5 Microsot Azure

Microsoft Azure, nekada poznat kao Windows Azure, je javna računalna platforma u oblaku koju pruža Microsoft. Ona nudi širok spektar usluga u oblaku koje uključuju računalstvo, analitiku, pohranu podataka i umrežavanje. 2008. godine, Microsoft je prvi put objavio svoje planove za uvođenje usluge računalstva u oblaku pod imenom Windows Azure. Nakon dostupnosti preglednih verzija i razvoja, usluga je službeno lansirana početkom 2010. godine. Do početka 2014. Microsoft je dodao i ažurirao širok raspon usluga uključujući Azure SQL, Windows Azure CTP, Windows Azure Connect, Upravitelj prometa i HPC planer.

Korisnici imaju mogućnost odabira između raznih usluga kako bi razvijali i skalirali nove aplikacije ili pokretali postojeće aplikacije u javnom oblaku. Cilj platforme Azure je pružiti podršku tvrtkama u suočavanju s izazovima i ostvarivanju njihovih organizacijskih ciljeva. Ona pruža alate koji podržavaju sve vrste industrija, uključujući e-trgovinu, financije i razne druge tvrtke, te je kompatibilna s tehnologijama otvorenog koda. Time korisnicima pruža fleksibilnost da koriste željene alate i tehnologije. Osim toga, Azure nudi četiri različite oblike računalstva u oblaku: infrastrukturu kao uslugu (IaaS), platformu kao uslugu (PaaS), softver kao uslugu (SaaS) i funkcije bez poslužitelja. Microsoft nudi sljedećih pet različitih opcija korisničke podrške za Azure:

Osnovni ili temeljni, Developer, Standard, Profesionalni izravni, Enterprise (Premier) (Bigelow, 2022).

Azure se također koristi kao platforma za smještaj baza podataka u oblaku. Organizacije imaju mogućnost iskoristiti gotovo neograničene kapacitete pohrane u oblaku kako bi pohranile velike skupove podataka, obavljale analitičke zadatke i zatim odbacivale podatke kada postanu zastarjeli ili više nisu potrebni - sve to bez potrebe za nabavom ili postavljanjem hardvera u lokalnom podatkovnom centru. Microsoft pruža relacijske baze podataka bez poslužitelja poput Azure SQL, kao i ne relacijske baze podataka kao što je NoSQL. Osim toga, platforma se često koristi za sigurnosno kopiranje i oporavak od rušenja i drugih poteškoća. Mnoge organizacije koriste Azure za arhiviranje podataka kako bi ispunile zahtjeve za dugoročno zadržavanje podataka ili oporavak od rušenja (DR).

4.3.6 Blazor

Blazor je moderna tehnologija razvoja web korisničkog aplikacija koju je razvio Microsoft (Bernasconi, 2023). Omogućuje razvoj modernih web aplikacija, kao što su visoko interaktivne jednostranične aplikacije (SPA) (Bernasconi, 2023). Microsoft je objavio Blazor model hostinga na strani poslužitelja u rujnu 2019. Kao dio .NET platforme, ima pristup golemom ekosustavu kojeg pruža .NET.

Blazor koristi HTML, CSS i C# za razvoj interaktivnih web korisničkih aplikacija na strani klijenta. Blazor također koristi Razor za kreiranje dinamičkog HTML sadržaja unutar web aplikacije. Omogućava programerima da lako kombiniraju HTML i C# kod, što rezultira čistim i čitljivim kodom. Osim toga, Razor također podržava kontrolu toka, uvjete, i mnoge druge konstrukcije koje čine razvoj web aplikacija jednostavnim i efikasnim. Umjesto da se oslanja na Microsoft-specifične tehnologije poput XAML-a, Blazor koristi standardne jezike poput HTML-a i CSS-a za opisivanje korisničke aplikacije. Prednost korištenja Blazora leži u činjenici da se mogu koristiti iste komponente neovisno o modelu hostinga koji je odabran. Može se stvoriti biblioteka klasa koja sadrži Blazor komponente i podijeliti ih među različitim aplikacijama. Nije bitno koji model hostinga koristi svaka pojedinačna aplikacija; sve one mogu pristupiti istoj biblioteci klasa i koristiti Blazor komponente.

Postoje četiri modela hostinga: Blazor WebAssembly koji u potpunosti se izvodi na uređaju klijenta što omogućuje bolje korisničko sučelje, ali ukoliko korisnik nema dovoljno snažan uređaj mogao bih imati poteškoća sa korištenjem iste. Blazor server pak se pokreće na serveru koristeći ASP.NET i skupa sa SIgnalR se ažurira promjene u stvarnom vremenu. Dakle sa ovim načinom server je odgovoran za prikazivanje sučelja korisniku što štedi resurse korisnika. Blazor Hybrid je najnoviji i najnapredniji model hostinga koji je dostupan. Ovaj model omogućuje djelomično izvođenje aplikacije na izvornoj platformi poput Electrona ili eksperimentalnog Mobile Blazor Bindingsa, dok drugi dio aplikacije radi u web pregledniku. Glavna svrha Blazor Hybrida je kombiniranje web tehnologija za implementaciju aplikacije s pristupom izvornim API-ima. Te Blazor United koji je još u razvojnoj fazi, a ideja je da koristi prednosti Blazor WebAssembly-a i Blazor Servera. Odnosno developeri će moći odlučiti hoće li se aplikacija pokretati na serveru ili na uređaju korisnika. Trebao bi pružati prednosti Blazor servera za zahtjeve niske latencije odnosno vremena čekanja od trenutka kada se pošalje zahtjev za podatcima do trenutka kada se isti ne prikažu, i prednosti Blazor WebAssembly-a za respozivna korisnička sučelja.

Prednost Blazora je što omogućuje web stranici da pokrene aplikaciju unutar preglednika. Nakon što je aplikacija preuzeta, moguć je prekida veze s poslužiteljem. Ipak, aplikacija nastavlja raditi, iako ne može dohvatiti nove podatke s poslužitelja. Može potpuno iskoristiti sposobnosti i resurse korisničkog uređaja. Nije potrebno koristiti ASP.NET Core web poslužitelj za hosting aplikacije. Značajno se smanjuje opterećenje poslužitelja ubrzavanjem vremena učitavanja jer su potrebne samo promjene u Document Object Modelu (DOM).

Tvrtke mogu iskoristiti Blazor razvojne usluge u raznim situacijama. Primjerice, kada žele brzo razviti aplikaciju, razviti MVC ili SPA aplikaciju s naglaskom na snažno korisničko sučelje (UI), ili kada razvijaju CRM za klijente ili javni sustav koji zahtijeva robotizirane funkcionalnosti i učinkovito korisničko iskustvo (UI/UX).

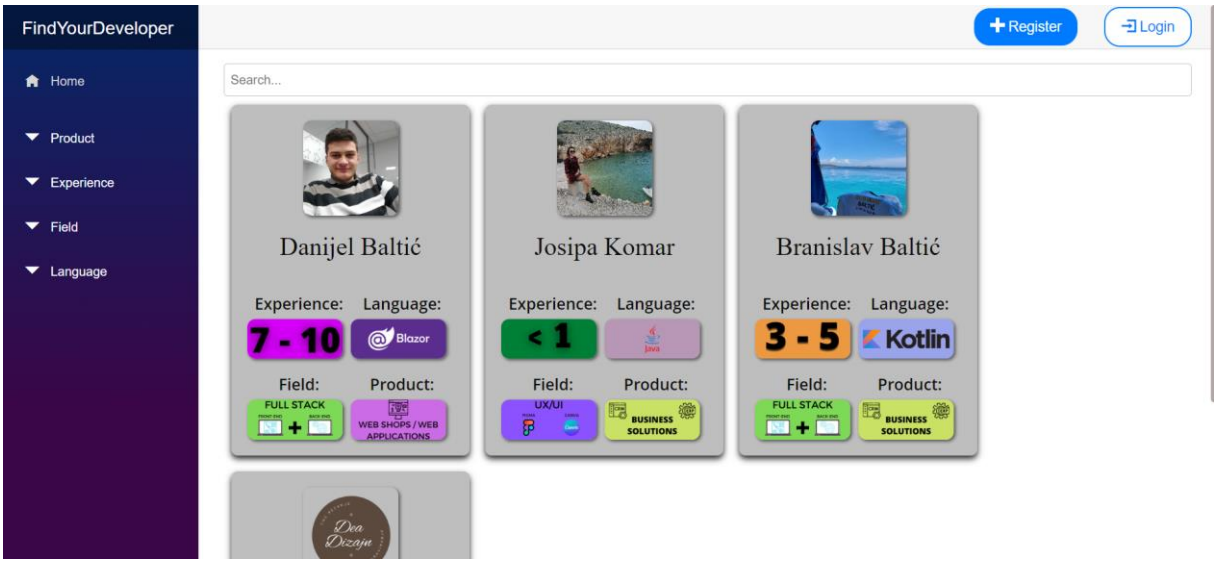
5. „Find Your Developer“ aplikacija

Kreiranje ove aplikacije predstavlja demonstraciju mogućnosti korištenih Mirosoftovih tehnologija. Uz korištenje Blazor WebAssembly za „Front-end“ dio aplikacije, ASP.NET Core web okvira za „Back-end“ dio aplikacije i SQL Servera za bazu podataka gdje su svi korisnički podaci uspješno pohranjeni. Sve navedene tehnologije su dio Microsoftovog ekosustava, što donosi prednosti velike korporacije u smislu dostupnosti raznovrsnih proizvoda i kvalitetne dokumentacije.

5.1 Predstavljanje aplikacije

Ideja ove aplikacije je napraviti jedan univerzalni popis programera (developera) koji su spremni i u mogućnosti odrađivati neke zadatke u određenim područjima. Ovdje će developeri imati priliku predstaviti svoje ključne vještine kao i iskustvo koje imaju do sada, te će imati i mjesto gdje će detaljno opisati i predstaviti sebe potencijalnim poslodavcima. Poslodavci će moći pretraživati te programere po karakteristikama koje njima odgovaraju, a to su: iskustvo, proizvodi s kojim su se programeri specijalizirali, područje koje su se specijalizirali i tehnologija tj. programski jezik u kojem su se specijalizirali. Jednom kada se napravi račun svi su u mogućnosti pregledati profili i pretražiti ga.

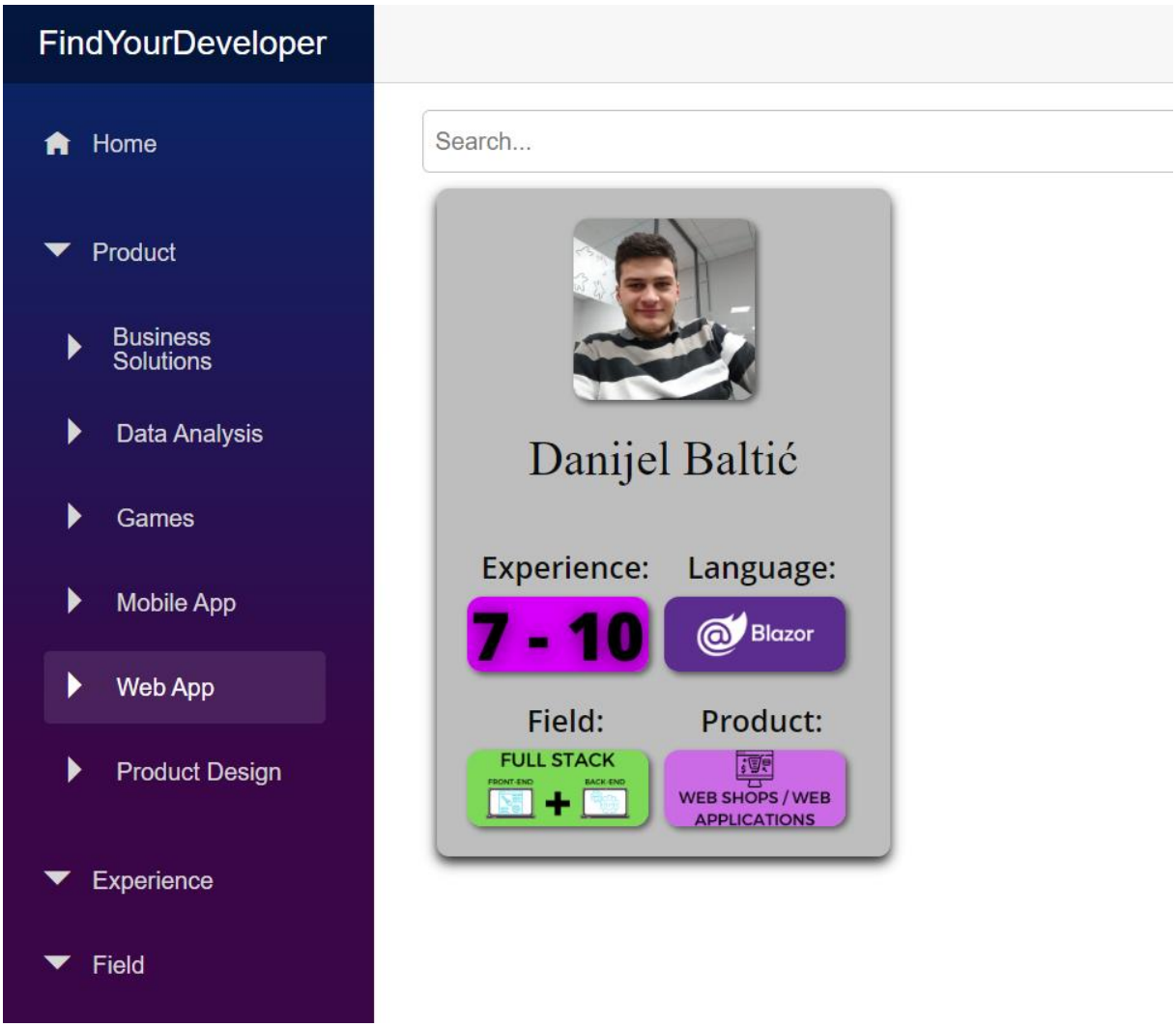
Početna stranica se sastoj od par ključnih komponenata, a to su: gornja navigacijska traka, bočna navigacijska traka te sadržaj same stranice. Te navigacijske trake ostaju kroz cijelu aplikaciju da se korisnici mogu u svakom trenutku prebaciti na neku drugu lokaciju na aplikaciji. U gornjem desnom kutu navigacijske trake nalazi se naslov aplikacije koji se prostire istom širinom kao i bočna navigacijska traka. Ova dizajnerska odluka daje aplikaciji elegantan i suvremen izgled. Također, naslov je naglašen tamnijom bojom u usporedbi s bočnom trakom, naglašavajući njegovu važnost. Dok je na drugoj strani navigacijske trake se nalazi dva gumba koja se mijenjaju u odnosu na stanje u kojem se klijent nalazi.



Slika 8: Snimka zaslona početne stranice aplikacije

Izvor: Izrada autora

Svaki gumb na navigacijskoj traci (osim „home“), predstavlja filtere koji su dostupni u aplikaciji. Klikom na neki od ta 4 gumba pojavljuje se padajući izbornik sa nešto više opcija filtriraju profile. Filtriranje funkcionira bez potrebe za osvježavanjem stranice



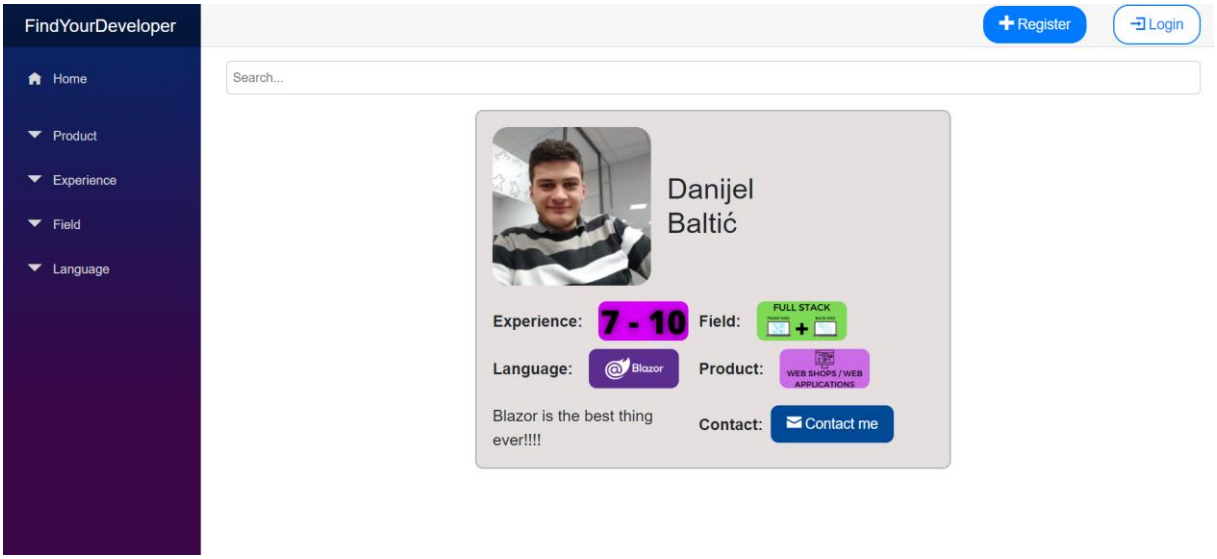
Slika 9: Snimka zaslona bočne navigacijske trake i prikaz funkcionalnosti filtriranja

Izvor: Izrada autora

Unutar sadržaja, ugrađeno je polje za pretraživanje koje omogućuje ručno upisivanje i pretraživanje profila koristeći ime, prezime ili čak ključne riječi iz opisa profila. Neposredno ispod polja za pretraživanje smještene su kartice profila koje prikazuju osnovne detalje. Te kartice profila sadrže osnovne detalje o profilu kao što su: Profilna slika, ime i prezime, te kategorije koje su vlasnici profila istaknuli, kao što su iskustvo, jezik tj. tehnologija koju su usavršili, polje u kojem su se specijalizirali te što su ima najbolji proizvodi koje su prethodno rješavali.

Kada korisnik klikne bilo gdje na karticu profila, otvara se novi prozor koji prikazuje proširenu verziju te kartice. Ova verzija je nešto veća i sadrži sve informacije koje se nalaze na osnovnoj kartici prikazanoj u popisu svih profila, ali i dodatne informacije

poput opisa profila. U tom prozoru nalazi se i gumb "Kontaktiraj me" koji omogućuje korisniku da direktno pošalje e-mail vlasniku profila.



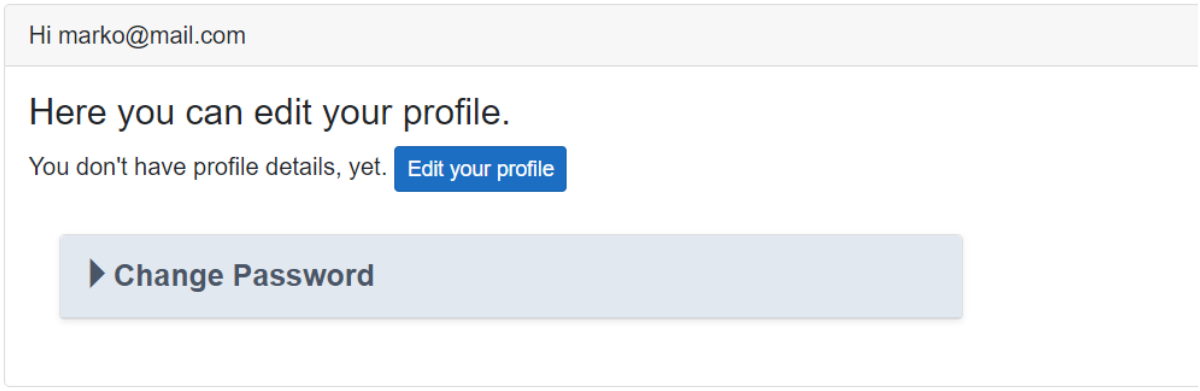
Slika 10: Snimka zaslona aplikacije, njenog detaljnog prikaza profila

Izvor: Izrada autora

Registracija i prijava su dvije stranice koje dijele sličnosti u pogledu jednostavnosti i intuitivnosti njihovog dizajna. Na ovim stranicama postoje polja za unos korisničke elektroničke pošte i lozinke

Kod postupka registracije novog korisnika polja koja treba ispuniti su: elektronička pošta, (koja mora zadovoljiti zadanu formu i ne smije biti ista) i lozinka. Nakon uspješne registracije korisničkog računa, aplikacija će korisnika preusmjeriti na prijavu u aplikaciju kako bi se prijavio i pristupio svom novootvorenom računu.

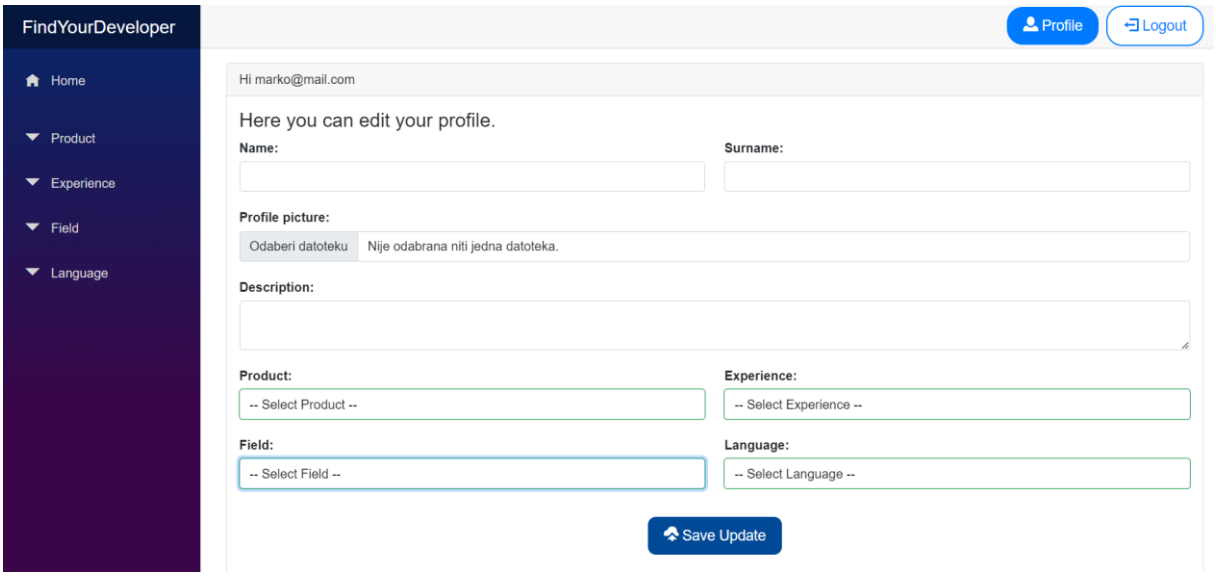
Stranica "Your Profile" predstavlja centralno mjesto za prikaz i uređivanje svih korisničkih podataka. Ova stranica pruža sveobuhvatan prikaz korisničkog profila. U sadržaju prozora pruža se smjernica da korisnik popuni svoj profil. Klikom na gumb "Edit your profile", otvara se obrazac za uređivanje profila. Ispod toga nalazi se padajući izbornik koji omogućuje promjenu lozinke.



Slika 11: prikaz stranice za ažuriranje podataka prilikom prve registracije

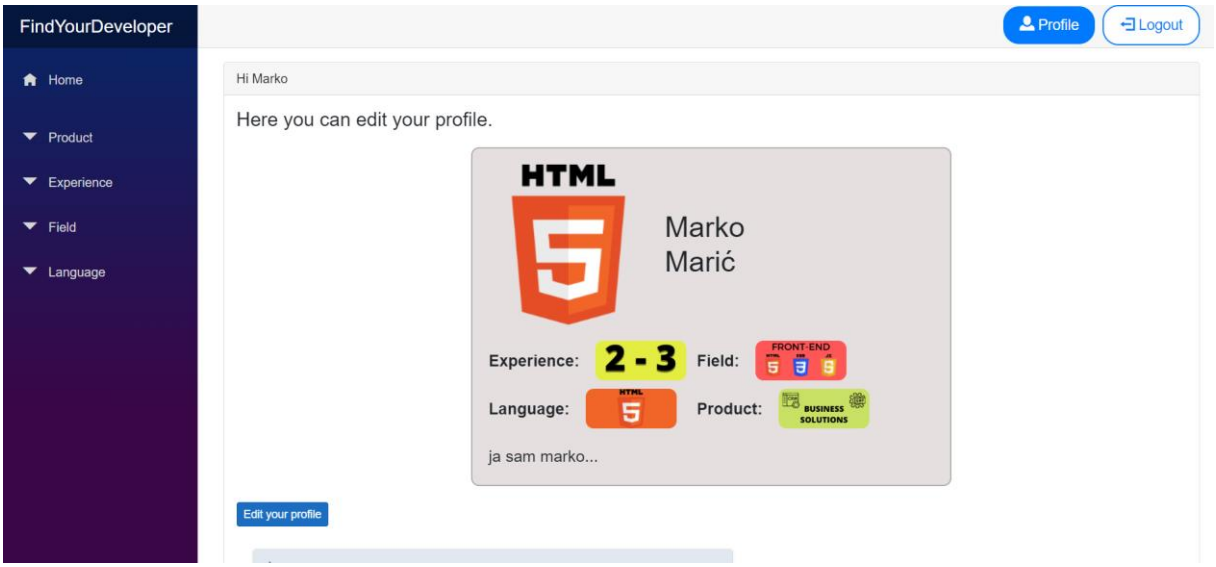
Izvor: Izrada autora

Nakon klika na gumb "Edit your profile", prozor se proširuje kako bi se smjestio cjelokupni obrazac za popunjavanje i ažuriranje profila, a rečenica koja poziva na klik na gumb "Edit your profile" zajedno s gumbom nestaju. Prikazani obrazac uključuje sljedeće elemente: "Name" (Ime), "Surname" (Prezime), "Profile Picture" (Profilna slika), „Description“ (Opis), i padajući izbornici naziva „Product“ (proizvod), „Experience“ (Iskustvo), „Field“ (Polje) i „Language“ (Jezik). Korisnik može i prenijeti svoju sliku u profil klikom na „Profile Picture“ koji otvara prozor za odabir fotografije s njihovog uređaja. Korisniku je omogućen odabir preferiranog proizvoda (koji je istaknut kao najjača strana korisnika). Moguće je odabrati: Poslovna rješenja, Analiza podataka, Igrice, Mobilne aplikacije, Web aplikacije i Dizajn proizvoda. Uz proizvode nalazi se još jedan padajući izbornik pod nazivom "Experience", koji označava korisnikovo iskustvo. Klikom na taj padajući izbornik, korisnicima su na raspolaganju nekoliko opcija, uključujući: manje od godinu, između jedne i dvije godine, između dvije i tri godine, između tri i pet godina, između pet i sedam godina, između sedam i deset godina te više od deset godina. Polje "Field" omogućuje odabir područja u kojem se korisnik specijalizirao, s opcijama poput "Front-end", "Back-end", "Full-stack" i "UX & UI". Opcije za padajući izbornik "Language" uključuju Angular, Blazor, C#, C++, CSS, Django, Figma, Canva, GO, HTML, INK, Java, JQuery, JavaScript, Kotlin, MySQL, Python, React, Ruby, Swift, Vue.js i Flutter.



Slika 12: Snimka zaslona aplikacije, forma za ažuriranje profila

Izvor: Izrada autora



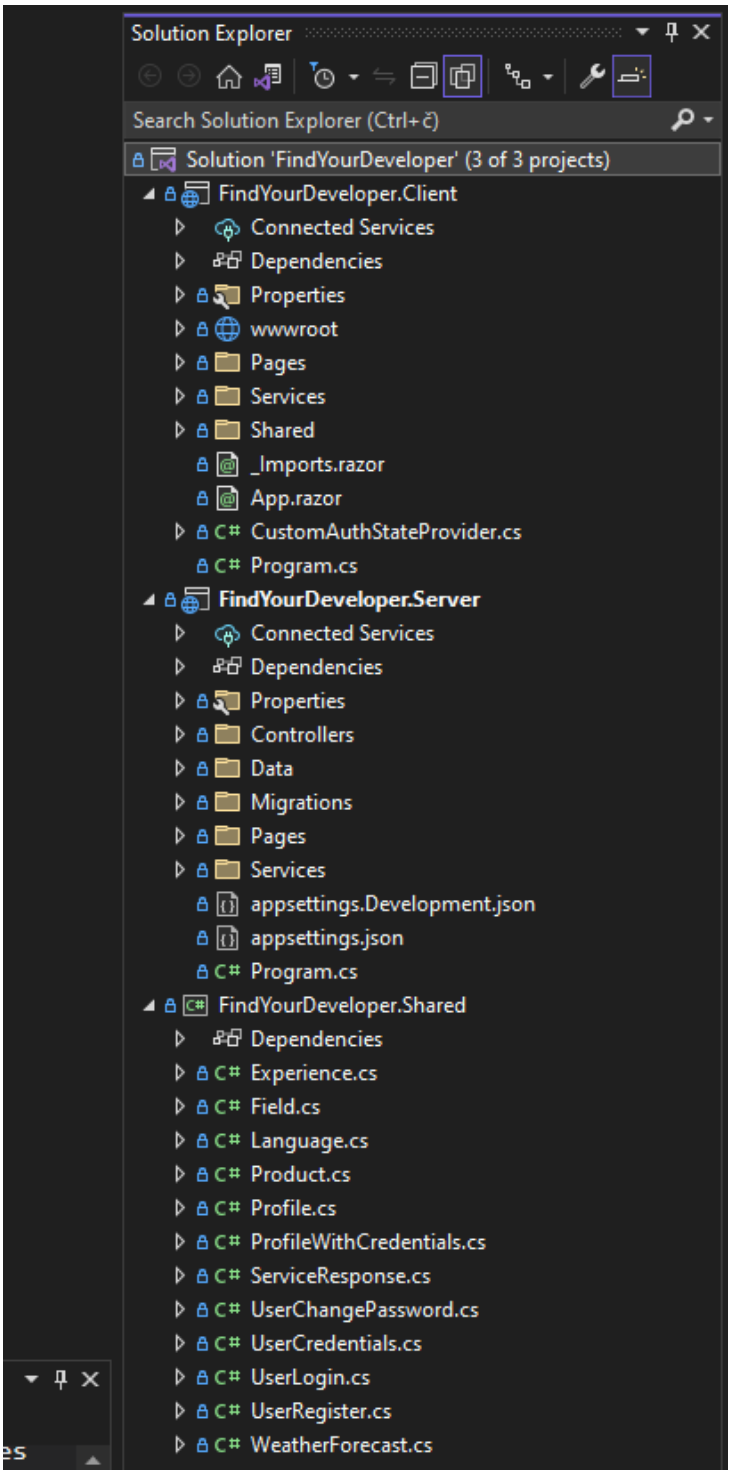
Slika 13: snimka zaslona aplikacije, prikaz korisničkovog profila

Izvor: Izrada autora

Ispod svih ovih elemenata, korisniku je uvijek na raspolaganju dodatna opcija u obliku padajućeg izbornika s pratećom formom. Naziv padajućeg izbornika je "Change Password" što označava "Promjena lozinke".

5.2 Tehničke karakteristike

Blazor Webassembly aplikacije se sastoje od tri ključna projekta, a to su „.Client“, „.Server“ i „.Shared“ prikazani na slici 14. i opisani u nastavku.



Slika 14: Snimka zaslona izgleda strukture aplikacije

Izvor: Izrada autora

5.2.1 FindYourdeveloper.Client

„Client“ dio aplikacije je podređen kao što i sama riječ kaže klijentu, dakle ta se komponenta orijentira na klijenta i ondje je cijeli „Front-end“ dio ove aplikacije. Prva datoteka u klijent-ovom dijelu aplikacije je „Conected Services“ koja ima značajnu ulogu u pojednostavljivanju procesa integracije vanjskih API-jeva i usluga u Blazor aplikaciju. Ova mapa omogućuje generiranje koda i pristup API-jima popularnih cloud usluga poput Azure-a, AWS-a i drugih. Korištenje povezanih usluga omogućuje brzo dodavanje funkcionalnosti aplikaciji bez potrebe za ručnim pisanjem cjelokupnog koda za interakciju s API-jima.

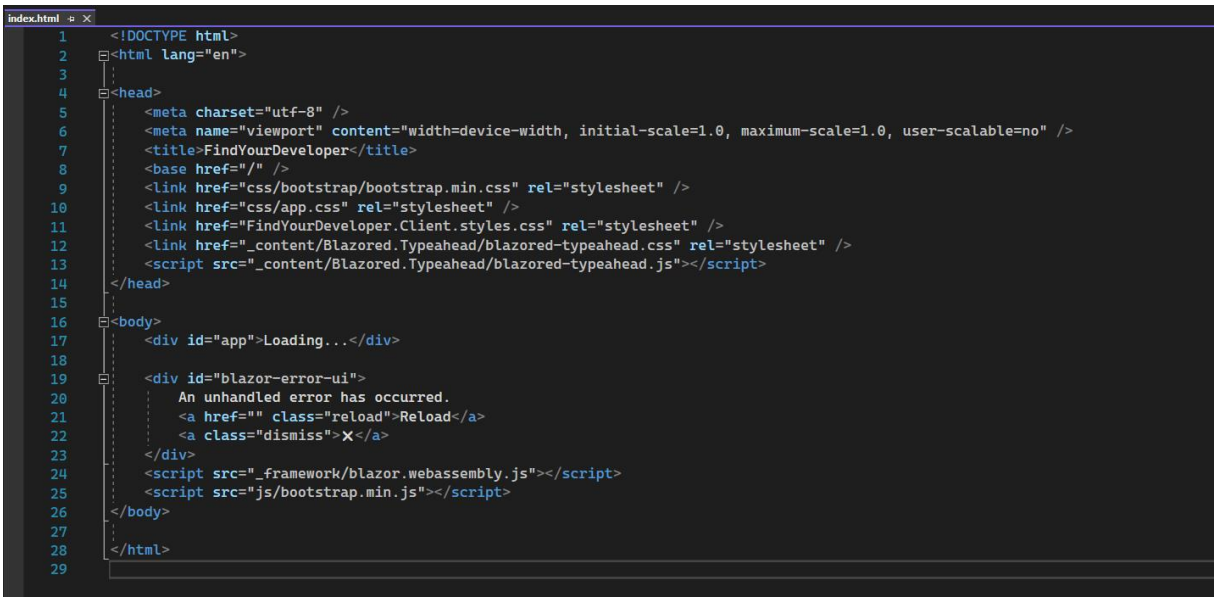
Nakon toga dolazi mapa nazvana "Dependencies" koja ima ulogu upravljanja vanjskim bibliotekama i paketima koji se koriste u Blazor aplikaciji. Ona omogućuje jednostavno integriranje vanjskog koda i resursa u projekt, što proširuje funkcionalnost aplikacije ili omogućuje korištenje već postojećeg koda iz drugih izvora. Ova mapa omogućuje precizno upravljanje svim potrebnim ovisnostima kako bi se osigurala stabilnost i optimalno funkcioniranje aplikacije. Tu dodajemo vanjske biblioteke koje su i ovdje morale biti korištene. Također tu ažuriramo i upravljamo verzijama biblioteke i pakete. U ovoj mapi možemo naći još par mapa sa raznim podatcima poput: „Analyzers“, „Framework“ gdje možemo provjeriti i ažurirati .Net verziju koju koristi aplikacija, „Packages“ gdje su svi paketi i biblioteke koje su instalirane za taj dio aplikacije, te imamo još i „Projects“ gdje vidimo sa kojim projektom je povezan ovaj projekt, u ovome slučaju je to „.Shared“ projekt.

Potom ide mapa po imenu „Properties“ koja sadrži jednu datoteku po imenu „launchSettings.json“ u toj datoteci se sadrži dijelovi koda i naredbe po kojima se aplikacije pokreće kako lokalno tako i na web preglednicima. Tu možemo naći sve detalje u pokretanju aplikacije kao i što je najvažnije „sslPort“ preko kojeg lokalno dolazimo do naše aplikacije.

Slika 15: Snimka zaslona rzvojnog okruženja za launchSettings.json datoteku

Izvor: Izrada autora

Potom slijedi mapa pod imenom „wwwroot“ to je folder koji ima generalni CSS za cijelu aplikaciju kao i različite ikone i slike koje se koriste u aplikacijama radi bržeg dohvaćanja. Takav način dohvaćanja slika i ikona se ne preporučuje jer povećava veličinu aplikacije što može rezultirati težem i zahtjevnijem otvaranju iste u korisnikovom Internet pregledniku. Tako je u CSS datoteci upisan i „Bootstrap“ koji olakšava pristup osnovnim ikonama i stilovima za uređivanje web aplikacija. Uz CSS, ikone i slike, u aplikaciji se također nalazi HTML datoteka koja služi kao osnovna točka za pokretanje cijele aplikacije. Ova datoteka je ključna jer web preglednici putem nje pristupaju svim ostalim sadržajima unutar aplikacije, što je čini izuzetno važnim dijelom iste.



Slika 16: Snimka zaslona rzvojnog okruženja, index.html datoteka

Izvor: Izrada autora

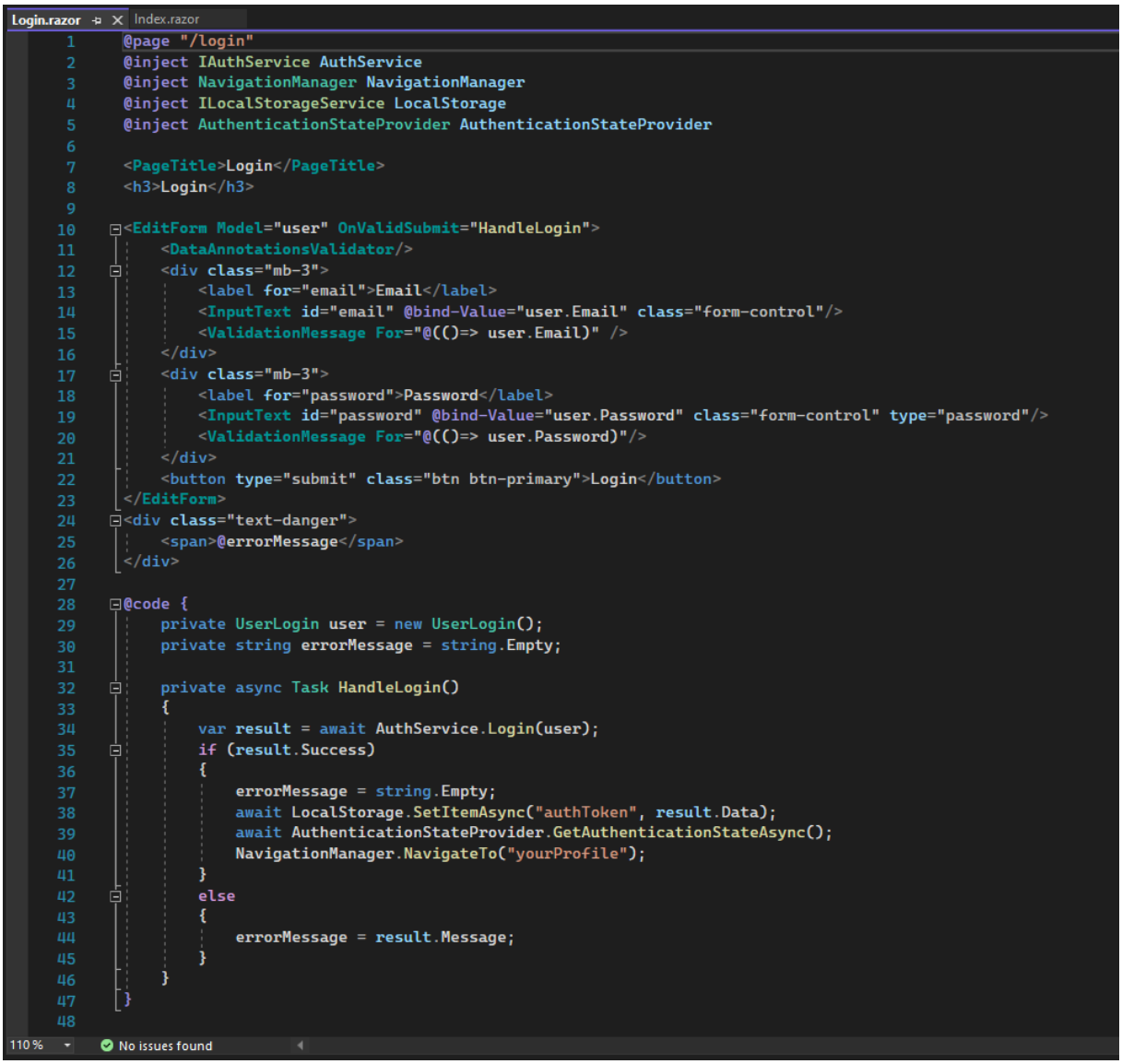
Nakon toga slijedi važna datoteka nazvana "Pages", koja sadrži sve stranice aplikacije. Osim stranica, korisnici mogu pohranjivati i odgovarajuće CSS datoteke povezane s pojedinom stranicom. Sve stranice imaju ekstenziju ".razor", budući da se za izradu stranica i komponenata koristi Blazor-ova .razor sintaksa.

Ovaj jezik omogućuje korisnicima da na elegantan način kombiniraju dinamički HTML i C# kod. Uz poznavanje HTML-a i C#, korisnicima neće biti teško koristiti ovaj jezik i njegovu sintaksu. Na primjer, mogu koristiti razne C# značajke na nekoliko načina, kao što su "inline code" ili "code block", a moguće je i izdvajanje koda u zasebne datoteke.

Važno je pridržavati se odgovarajućeg imenovanja datoteka. CSS datoteke povezane sa stranicama moraju imati isto ime kao i stranice, s dodatnom ekstenzijom koja označava tip datoteke. Primjerice, CSS datoteka povezana s razor stranicom može se nazvati "stranica.razor.css", dok se logički dio, odnosno C# kod za istu stranicu, može nazvati "stranica.razor.cs".

Također, važno je naglasiti da razor sintaksa pruža mogućnost korištenja direktiva u stranicama. Ove direktive se obično nalaze na vrhu stranica i počinju simbolom "@". One se koriste za definiranje prostora imena, postavljanje naslova

stranice, nasljeđivanje šablona i druge konfiguracijske postavke. Također uz pomoć razor sintakse možemo „pozvati“ C# kod u html dokument pomoću „@“ znaka. Tako je u ovom primjeru i kroz cijeli projekt ta mogućnost korištena više puta. Tako da ovdje vidimo kako razor uključuje C# kod koji bi trebao poslati upisane podatke korisnika na server (@bind-Value=“user.Email“) ili u slučaju krivo napisane e-mail adrese se opet poziva pomoću razora i C# odgovarajuća poruka „@(()=> user.Email)“.



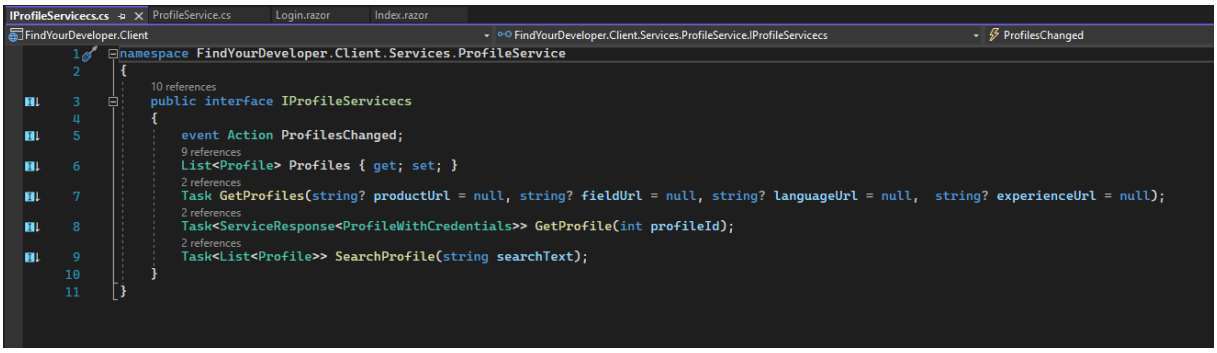
Slika 17: Snimka zaslona rzvojnog okruženja, login.razor kod

Izvor: Izrada autora

Nakon toga dolazi datoteka nazvana "Service", koja se koristi za pozivanje svih API usluga iz Server projekta. Radi lakšeg snalaženja kroz kod aplikacije, svaka usluga ima vlastitu datoteku koja odgovara njenom kontekstu i svrsi.

U datoteci "AuthService" se nalaze sve usluge vezane uz autentifikaciju korisnika. Ove usluge se koriste u različitim dijelovima aplikacije, jer je važno da aplikacija uvijek zna je li korisnik registriran i prijavljen ili ne. Također, imamo datoteku "EditProfileService" koja se fokusira na podatke pojedinog korisnika. Ovdje se nalaze sve API usluge potrebne za prikazivanje korisničkog profila i ažuriranje tih podataka. Iako stranica "YourProfile.razor" koristi neke usluge iz datoteke "AuthService", poput promjene lozinke i dohvaćanja trenutnog korisnika. Slijede četiri slične, ali jednako važne datoteke. Svaka od njih je zadužena za prikazivanje filter gumba na bočnoj navigacijskoj traci. Datoteke "ExperienceService", "FieldService", "LanguageService" i "ProductService" pružaju API pozive vezane za te filtre. Zadnja datoteka, "ProfileService", također ima važnu ulogu. Ovdje se pozivaju svi API-ji vezani uz prikaz profila. S obzirom na različite načine prikaza profila, ova datoteka sadrži različite API pozive, kao što su prikaz detaljnog profila, prikaz profila pretraženog po ključnoj riječi iz pretraživača, prikaz svih profila i prikaz profila s određenim odabranim filtrom.

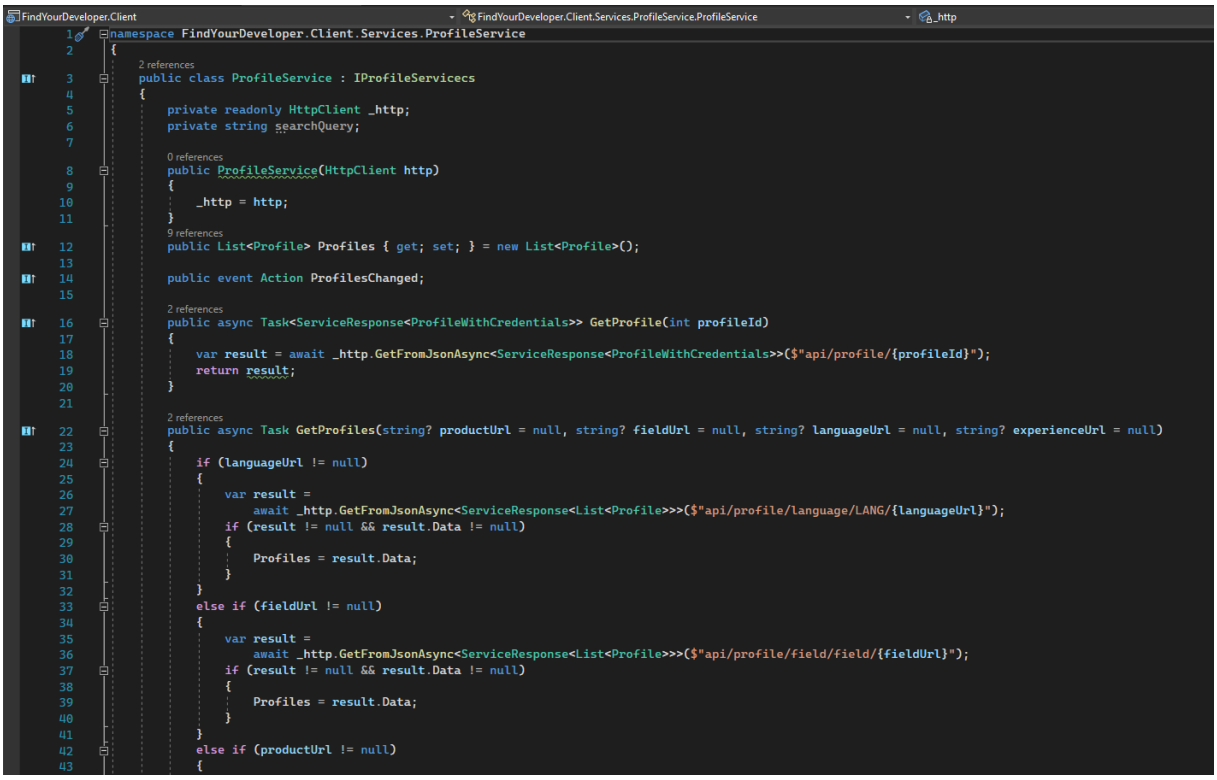
Sve navedene datoteke sastoje se od dva dokumenta koda koji završavaju s ".cs" ekstenzijom, što ukazuje da se u tim dokumentima pišu C# funkcije. S ciljem poboljšanja čitljivosti i razumijevanja koda, svaki API ima dva odvojena dokumenta. Prvi dokument deklarira i prosljeđuje razor stranicama kako bi se koristile u njima, dok istovremeno komunicira s drugim dokumentom koji obavlja sam poziv API-ja. Ovaj drugi dokument poznat je kao "Interface" ili sučelje.



Slika 18: Snimka zaslona rzvojnog okruženja, kod deklariranja API-ja

Izvor: Izrada autora

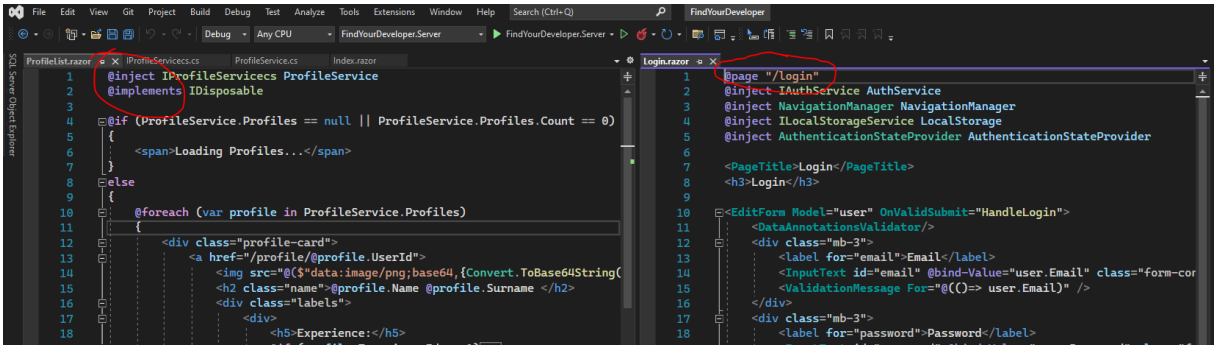
Nadalje, drugi dokument unutar datoteke služi za komunikaciju s kontrolerom u server projektu i dohvaćanje API funkcija. Ovdje se dohvaćaju podaci iz baze podataka i generiraju URL-ovi za pristup tim pozivima. Na primjer, postoji petlja koja se koristi za početnu stranicu i provjerava koje podatke želimo dohvatiti s obzirom na dostupne filtere u aplikaciji i korisnikove odabire. Petlja provjerava svaki filter jedan po jedan, a ako nijedan filter nije zadovoljen, poziva se kontroler na serveru koji vraća kompletnu listu profila. Osim toga, postoji događaj (event) koji prati promjene i ponovno aktivira provjeru ukoliko se neka promjena dogodi, što omogućuje prikazivanje željenih rezultata bez ponovnog učitavanja stranice. Ovaj dio koda je jednostavan, ali iznimno koristan jer omogućuje brzo funkcioniranje aplikacije. Također, važno je napomenuti da je ovaj dio morao biti deklariran i u sučelju kako bi se mogao pozvati u određenim razor komponentama.



Slika 19: Snimka zaslona rzvojnog okruženja, dio kod klase pozivanja API-ja kod pozivanja profila

Izvor: Izrada autora

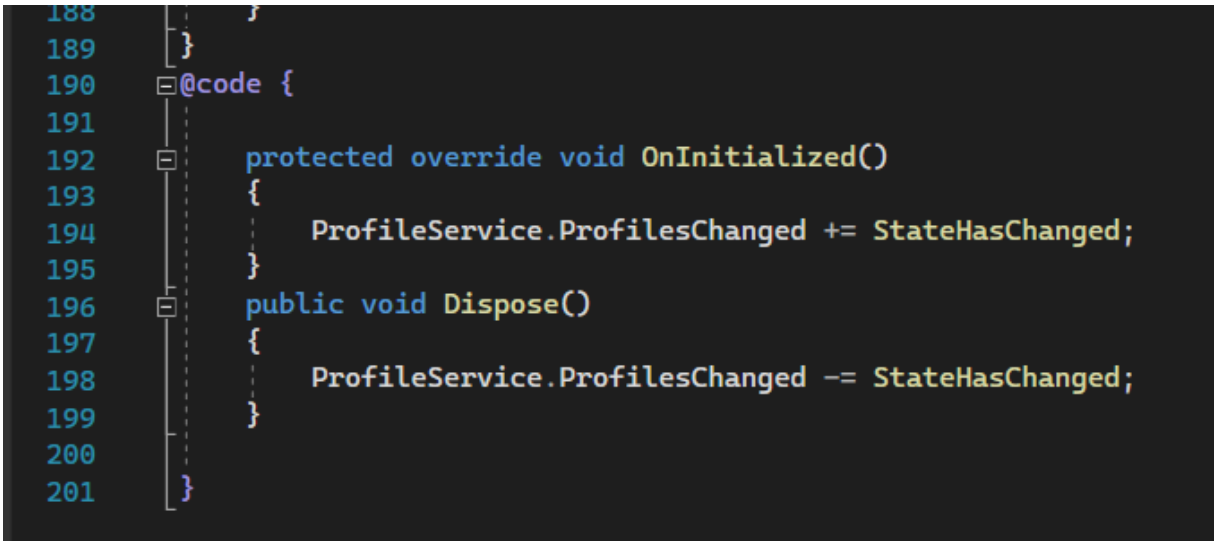
Nakon toga slijedi datoteka "Shared", koju ne treba miješati s aplikacijom "Shared", budući da imaju potpuno različite funkcije. Te datoteke nisu stranice kao što je to slučaj s datotekom "Pages". Iako na prvi pogled izgledaju slično, imaju različitu svrhu. Jedina razlika je u tome što stranice moramo dohvatiti putem URL-ova, dok do komponenti dolazimo putem stranica. Odnosno komponente integriramo u stranice ili pak druge neke komponente koje opet moraju biti pozvane na nekoj od stranice. Blazor također dopušta i zvanje stranica u stranicama odnosno stranica može imati i svrhu komponente. Kod stranica samo deklariramo da je to stranica „@page“ predznakom te potom u navodne znake staviti „/“ kao razdvajanje od glavnog URL-a i naziv linka kojeg želimo da bude istaknut, te potom možemo koristiti sve ostale funkcije razor jezika. Dok je najčešće naslovna strana ta koja nema nikakvog imena pa onda samo stoji znak „/“.



Slika 20: snimka zaslona rzvojnog okruženja, razlika u kodu između stranice i komponente

Izvor: Izrada autora

U datoteci "Shared" se nalazi veliki broj komponenti koje se koriste na različitim stranicama, pri čemu se često koriste za pozivanje API-ja radi jednostavnijeg i preglednijeg koda. Ovaj pristup kodiranju i kreiranju aplikacija ili sadržaja poznat je kao bločno programiranje, gdje se sadržaj aplikacije sastavlja iz više komponenti. Ovo se čini radi olakšanog otkrivanja grešaka i čitljivosti koda. Naravno, API pozive je moguće izvršiti i direktno unutar stranica, što se u ovom slučaju također činilo. API-je pozivamo tako što prvo uključimo sučelje koje je kreirano ili pozvano na Server aplikaciji, a zatim, pomoću C# jezika, postavljamo uvjete za prikazivanje određenih podataka iz baze podataka.

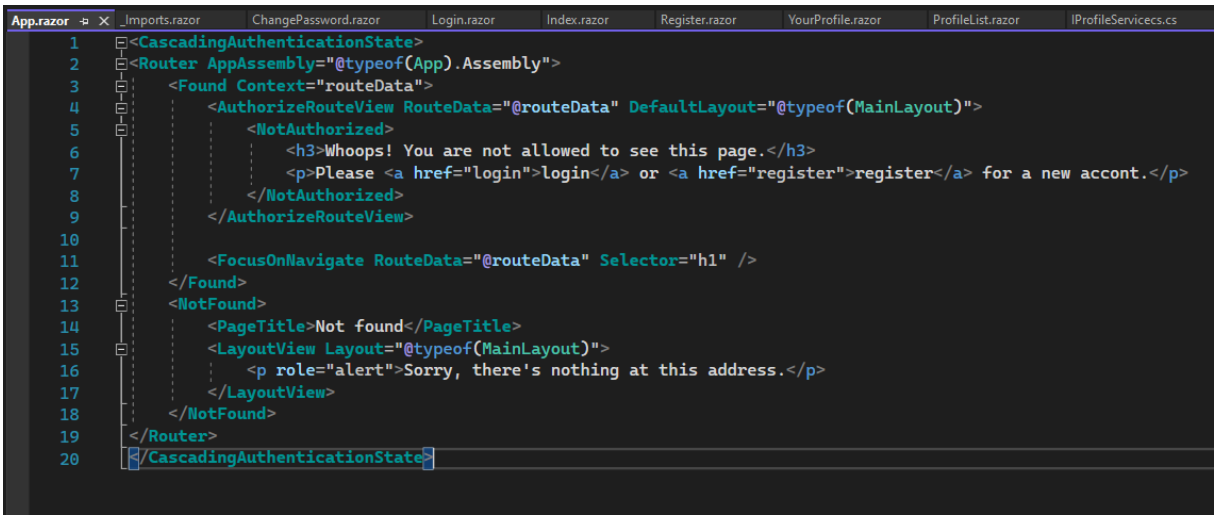


Slika 21: Snimka zaslona rzvojnog okruženja, dio koda koji provjerava dali je došlo do promjene u odabiru filtera profila

Izvor: Izrada autora

Potom imamo datoteku pod imenom „_Imports.razor“ što označava da je ona pisan u razor sintaksi, a njena jedina svrha je da uključi i učini dostupnima sve pakete usluge kroz cijeli aplikaciju. To olakšava pristup istima, u protivnom bi uvijek morali, gdje god u „Client“ dijelu aplikacije koristili neku od usluga i paketa posebno koristi dodatnu direktivu „@using“ potom onda tek koristili, što usporava i otežava pisanje koda. Također ova datoteka se pokazala ključnom u rješavanju najčešćih problema s kojima se susreću korisnici Balzora, jer programeri su često zaboravljali uključivat na ovaj način pakete i usluge. To je riješeno na ovakav način i dobra praksa je prilikom kreiranja usluge ili instalacijom određenog paketa, odmah istu unijeti u ovaj dokument i time učiniti svoju uslugu ili paket dostupni cijelom projektu.

Zatim, postoji važna datoteka koja obično ne sadrži mnogo linija koda, ali je izuzetno moćna - "App.razor". U ovom dokumentu se također koristi razor sintaksa. Ona predstavlja glavnu komponentu korisničkog sučelja, poznatu kao "UI". Definira korijensku komponentu aplikacije koja se prikazuje pri pokretanju aplikacije. Ovdje se mogu kontrolirati ključni aspekti aplikacije, poput rasporeda, adresiranja, konfiguracije usluga i autorizacije. Npr., koristi se "CascadingAuthenticationState" kako bi se kontroliralo što registrirani korisnik može vidjeti, a što ne. Drugim riječima, može se odrediti što će se ispisati korisniku ako pokuša pristupiti neovlaštenom dijelu aplikacije.



Slika 22: Snimka zaslona rzvojnog okruženja, App.razor kod

Izvor: Izrada autora

Tako da se posebno po komponentama odlučuje i označava što se može vidjeti a što pak ne može. Ukoliko želimo da stranica bude cijela samo za prijavljene korisnike tada samo pomoću razor atributa proglasimo stranicu strogo za autorizirane u vrhu koda, a ispod svog uključivanja (Inject). To sintaksa je jednostavna i razumljiva tako da će svatko vrlo jednostavno ju moći koristiti, a to je: „@attribute [Authorize]“. Ukoliko pak želimo samo dio stranice ili komponente tada unutar HTML dijela razor stranice deklariramo autorizirani pogled te u tom dijelu možemo staviti što se prikazuje registriranom korisniku, a što ne registriranom korisniku sa jednostavnom sintaksom.

```
<AuthorizeView>
    <Authorized>
        <a href="yourProfile" class="register-button"><i class="oi oi-person"></i> Profile</a>
        <button class="login-button" @onclick="Logout"><i class="oi oi-account-logout"></i> Logout</button>
    </Authorized>
    <NotAuthorized>
        <a href="register" class="register-button"><i class="oi oi-plus"></i> Register</a>
        <a href="login" class="login-button"><i class="oi oi-account-login"></i> Login</a>
    </NotAuthorized>
</AuthorizeView>
```

Slika 23: Snimka zaslona rzvojnog okruženja, prikaz sintakse reguliranja što može registrirani i ne registrirani korisnik vidjeti

Izvor: Izrada autora

Zatim, postoji datoteka koja omogućuje pristup informacijama o stanju autentifikacije u aplikaciji. U ovoj datoteci nalazi se klasa koja upravlja autentifikacijom u aplikaciji, nazvana "CustomAuthStateProvider". U glavnoj metodi "GetAuthenticationStateAsync()", provjerava se postoji li token za autentifikaciju pohranjen u lokalnom skladištu. Ako postoji, taj token se analizira kako bi se dobio pristup podacima o korisniku. Također, token se postavlja kao autorizacijski "header" za sve HTTP zahtjeve. U slučaju da token nije valjan ili ne postoji, on se briše iz lokalnog skladišta, a korisnički identitet se postavlja kao prazan identitet. Nakon toga, stvara se instanca "ClaimsPrincipal" koja predstavlja registriranog korisnika s izvučenim podacima iz tokena. Zatim se stvara instanca "AuthenticationState" koja sadrži informacije o trenutnom stanju autentifikacije. Konačno, obavještava se da se stanje autentifikacije promijenilo korištenjem metode "NotifyAuthenticationStateChanged()". Na taj način osigurava se da informacije o autentifikaciji budu ažurirane i dostupne u cijeloj aplikaciji.

I zadnja, vjerojatno najbitnija datoteka u klijentskom dijelu Blazor aplikacije je "Program.cs". Ona se koristi za konfiguriranje i pokretanje same aplikacije. U ovoj datoteci se konfigurira i pokreće klijentska aplikacija, postavlja "HttpClient" za komunikaciju s poslužiteljem, konfigurira "Dependency Injection" kontejner, definiraju se klijentske usluge i usmjeravanje, te se podešava lokalizacija. Ova datoteka omogućuje prilagodbu i pokretanje klijentskog dijela aplikacije prije nego što se izvrši. Sve usluge koje komuniciraju sa Server dijelom aplikacije su izgrađene upravo ovdje.

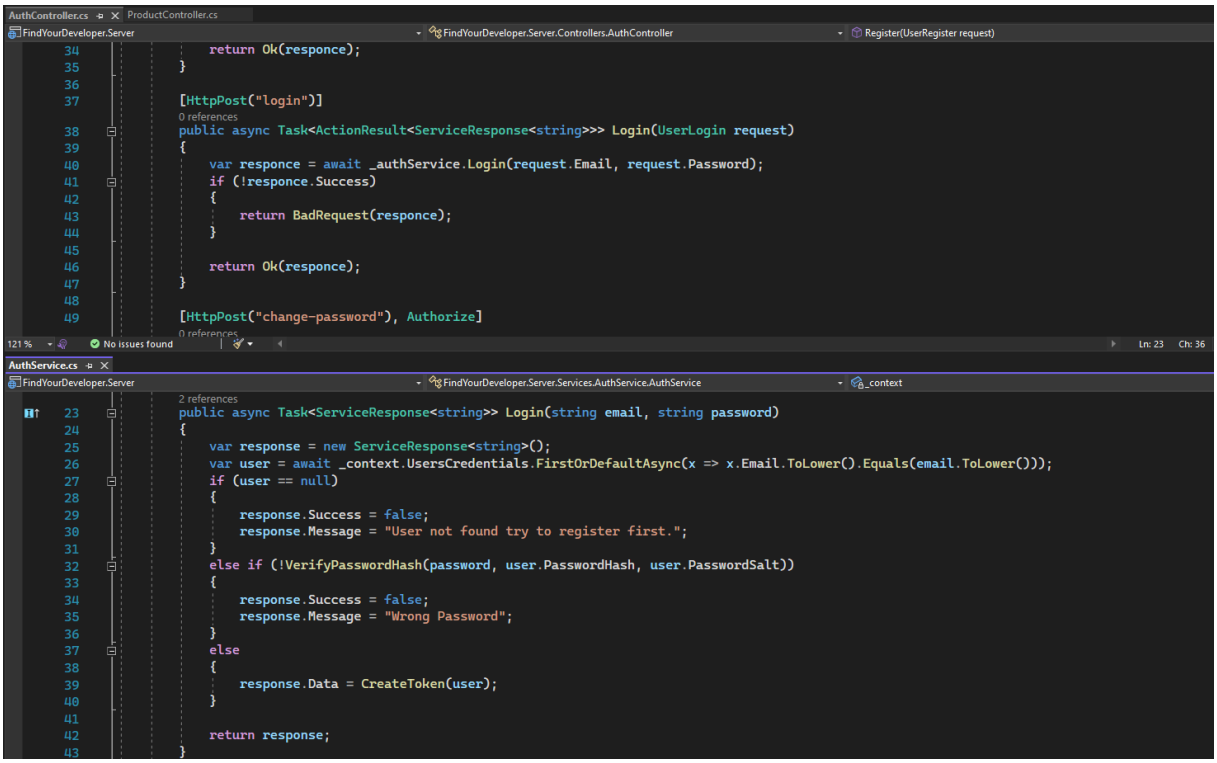
5.2.2 FindYourDeveloper.Server

Server projekt u Blazor WASM aplikaciji je zadužen za pružanje serverske podrške klijentskom dijelu aplikacije. On obavlja različite zadatke koji su potrebni za pravilan rad i interakciju između klijenta i poslužitelja. Tako da Server projekt u Blazoru zapravo služi kao „Back-end“ sa svojim funkcionalnostima ali i uključuje posluživanje statičnih datoteka.

I ovaj projekt ima prve tri datoteke jednake kao i projekt klijenta, a to su „Connected Service“, „Dependencies“ i „Properties“. Tako i maju i iste funkcionalnosti kao one u klijentu.

Sljedeći specifična datoteka je „Contoler“ gdje imamo za svaki poziv tj. HTTP zahtjev sa klijentske strane aplikacije, kojeg obrađuje isti te šalje rezultat dalje na obradu klijentu. Kada klijentska strana aplikacije šalje HTTP zahtjev, taj zahtjev se prenosi na server, gdje kontroleri služe kao ulazna točka za obradu tog zahtjeva. Kontroleri sadrže različite metode koje su označene atributima HTTP adresiranja, poput [HttpGet] ili [HttpPost], kako bi se odredilo kako će se zahtjev obraditi. Na primjer, [HttpGet] označava da se metoda koristi za obradu GET zahtjeva. U ovoj aplikaciji ih ima jako puno jer se različiti podatci dohvaćaju iz baze na različiti način.

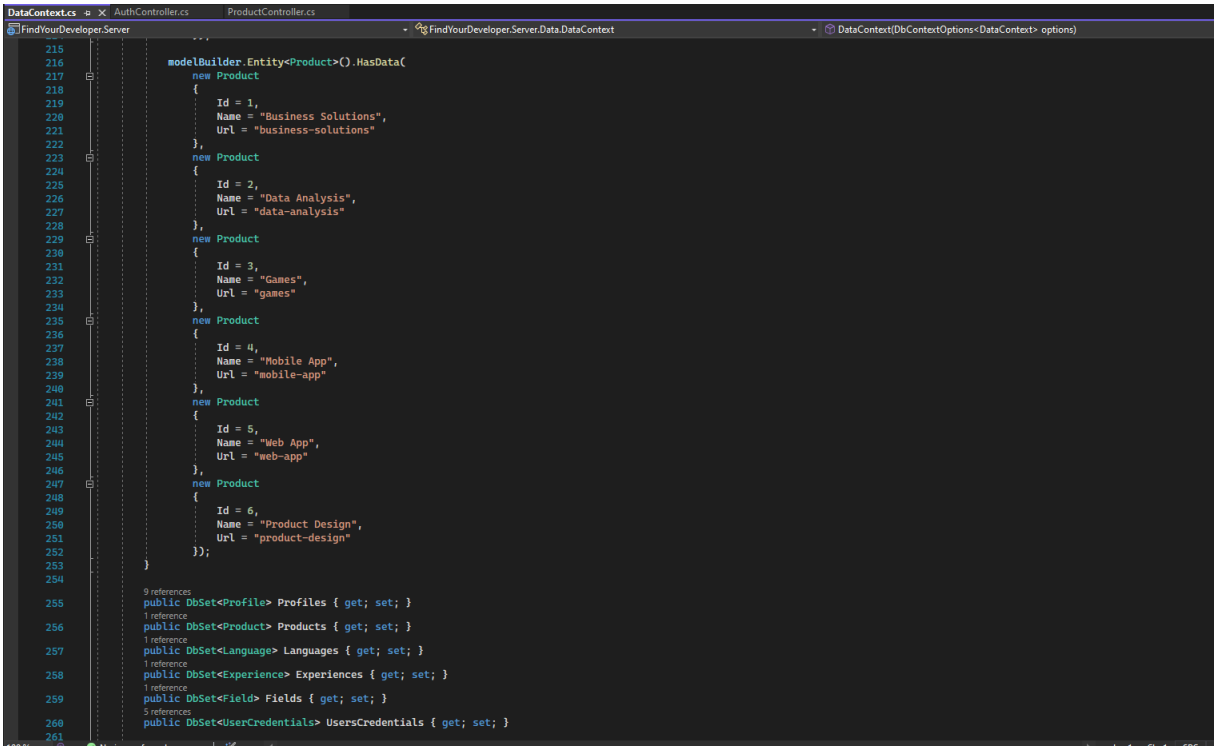
Unutar kontrolera, logika potrebna za obradu zahtjeva se može implementirati. To može uključivati dohvaćanje podataka iz baze podataka, izvođenje određenih operacija, provjeru autentifikacije ili autorizacije, te izgradnju odgovora koji se šalje natrag klijentu. Međutim, radi bolje čitljivosti koda, logiku se često prenosi na različite "Service" klase unutar istog projekta. Ovaj pristup se obično naziva "Thin Controller" i suprotan je "Fat Controller" pristupu, gdje se kompletna logika piše unutar samog kontrolera, rezultirajući velikim datotekama s mnogo linija koda.



Slika 24: snimka zaslona rzvojnog okruženja, "Thin Controller " kod i njegova logika na "Service"

Izvor: Izrada autora

Zatim se nalazi datoteka "Data", koja sadrži jednu izuzetno bitnu klasu po imenu "DataContext.cs". Ta klasa služi kao kontekst baze podataka za Blazor aplikaciju na server strani. Ona definira model podataka, konfigurira entitete i pruža pristup bazi podataka putem svojstva "DbSet<>" kako bi se izvršile operacije nad bazom podataka. Uz to, bilo je potrebno konfigurirati neke fiksne vrijednosti za filtere, što je postignuto kroz metodu "OnModelCreating" koja nadjačava osnovnu implementaciju iz "DbContext"-a. Ova metoda se koristi za konfiguriranje modela podataka pomoću fluentnog API-ja pruženog od strane "Entity Framework Core-a". U konkretnom kodu, metoda se koristi za unos početnih podataka u bazu za entitete "Field", "Language", "Experience" i "Product". Za svaki entitet se poziva metoda "HasData" kako bi se definirali početni podaci. Klasa "DataContext" također deklarira svojstva tipa "DbSet<>" za svaki entitet, kao što su "Profiles", "Products", "Languages", "Experiences", "Fields" i "UsersCredentials". Ova svojstva predstavljaju odgovarajuće tablice u bazi podataka i omogućuju izvršavanje operacija nad bazom podataka, kao što su upiti, umetanje, ažuriranje i brisanje podataka, putem Entity Framework Core-a.



Slika 25: snimka zaslona rzvojnog okruženja, dio "DataContext.cs" koda

Izvor: Izrada autora

"Migrations" datoteka sadrži dokumentirane zapise o promjenama u vezi s bazom podataka. Ona sadrži sve informacije o načinima komunikacije s bazom podataka u trenutku sinkronizacije s SQL Serverom koji je korišten za stvaranje baze podataka. Prilikom migracije baze podataka, generira se nova datoteka koja opisuje sve promjene u odnosu na bazu, uključujući kreiranje novih tablica, veza između tablica i ostalih promjena. Ova datoteka služi kao mehanizam provjere prije nego što se promjene primijene na bazu podataka. Provjeravamo ispravnost odnosa između tablica i, ako je sve ispravno, migracija se izvršava i promjene se primjenjuju u bazi podataka.

Tu je možda i malo neobična datoteka "Pages" koja se koristi za pohranjivanje Razor komponenti koje generiraju HTML sadržaj na server strani. Ove Razor komponente izvršavaju se na serveru i generiraju HTML koji se šalje klijentu. Jedna uobičajena komponenta koja se nalazi u datoteci "Pages", pa tako i ovdje, je "Error.razor" stranica. Ova stranica je namijenjena prikazu grešaka koje se javljaju tijekom izvođenja aplikacije. Kada se dogodi greška u aplikaciji, korisnik će biti preusmjeren na "Error.razor" stranicu kako bi mu se prikazala odgovarajuća poruka o grešci.

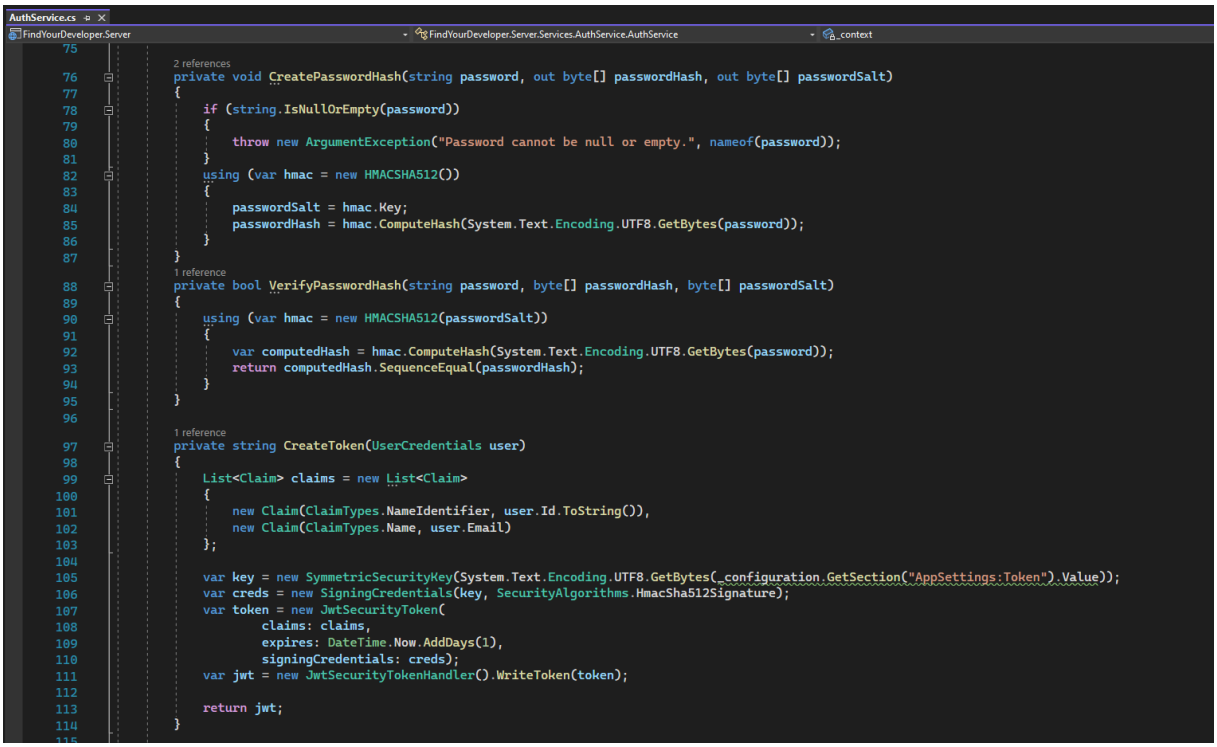
Posljednja, ali i najbitnija datoteka je "Services", koja je prethodno spomenuta prilikom opisa kontrolera. Ovdje je napisana sva logika za dohvaćanje i obradu podataka iz tablica. Struktura je slična kao i u "Client" dijelu, s mapama ili datotekama za pojedine usluge, poput "AuthService" koji pruža usluge autentifikacije korisnika. Također, imamo "Interface" ili sučelje u kojem se deklariraju i prosljeđuju kontroleru, te klase u kojima je implementirana logika.

Specifično, u datoteci "AuthService.cs" nalaze se različite metode koje su možda najkompliciranije u cijeloj aplikaciji. Tu su metode za prijavljivanje i registraciju korisnika, kao i metode za kreiranje ključeva za lozinke, budući da lozinke ne smiju biti spremljene kao obični tekst u bazi podataka radi sigurnosti. Umjesto toga, koriste se polja bajtova, poput "passwordHash" i "passwordSalt", kako bi se osigurala visoka razina sigurnosti aplikacije i korisničkih podataka.

Ovdje se također nalazi metoda za kreiranje tokena ili ključeva za verifikaciju korisnika, što je već spomenuto u "Client" projektu. Također postoje metode za

promjenu lozinke, provjeru dostupnosti korisničkog imena ili e-maila te metoda za dohvaćanje trenutnog korisnika koja se koristi pri ažuriranju korisničkih podataka.

Iako su ostale usluge također bitne za funkcionalnost aplikacije, ova datoteka se ističe kao najkompleksnija. Uz to, postoje i druge datoteke s pratećim uslugama, poput "EditProfileService" koji je odgovoran za promjene u bazi podataka i dohvaćanje podataka o trenutnom korisniku. "ExperienceService", "FieldService", "LanguageService" i "ProductService" služe za dohvaćanje podataka iz tablica kako bi se istaknuli filteri. Također postoji datoteka "ProfileService" koja se koristi za dohvaćanje općenitih podataka o profilima, ovisno o odabranom filteru, ali i detaljnih informacija o pojedinačnom profilu.



Slika 26: Snimka zaslona rzvojnog okruženja, prikaz logike kreiranja lozinke i kreiranja tokena

Izvor: Izrada autora

Zatim se nalaze dva dokumenta koji na prvi pogled izgledaju isto, ali zapravo imaju neke razlike. Radi se o "appsettings.json" i "appsettings.Development.json" datotekama koje omogućavaju definiranje različitih parametara kao što su veze s bazom podataka, API ključevi, postavke autorizacije, URL-ovi i druge konfiguracijske

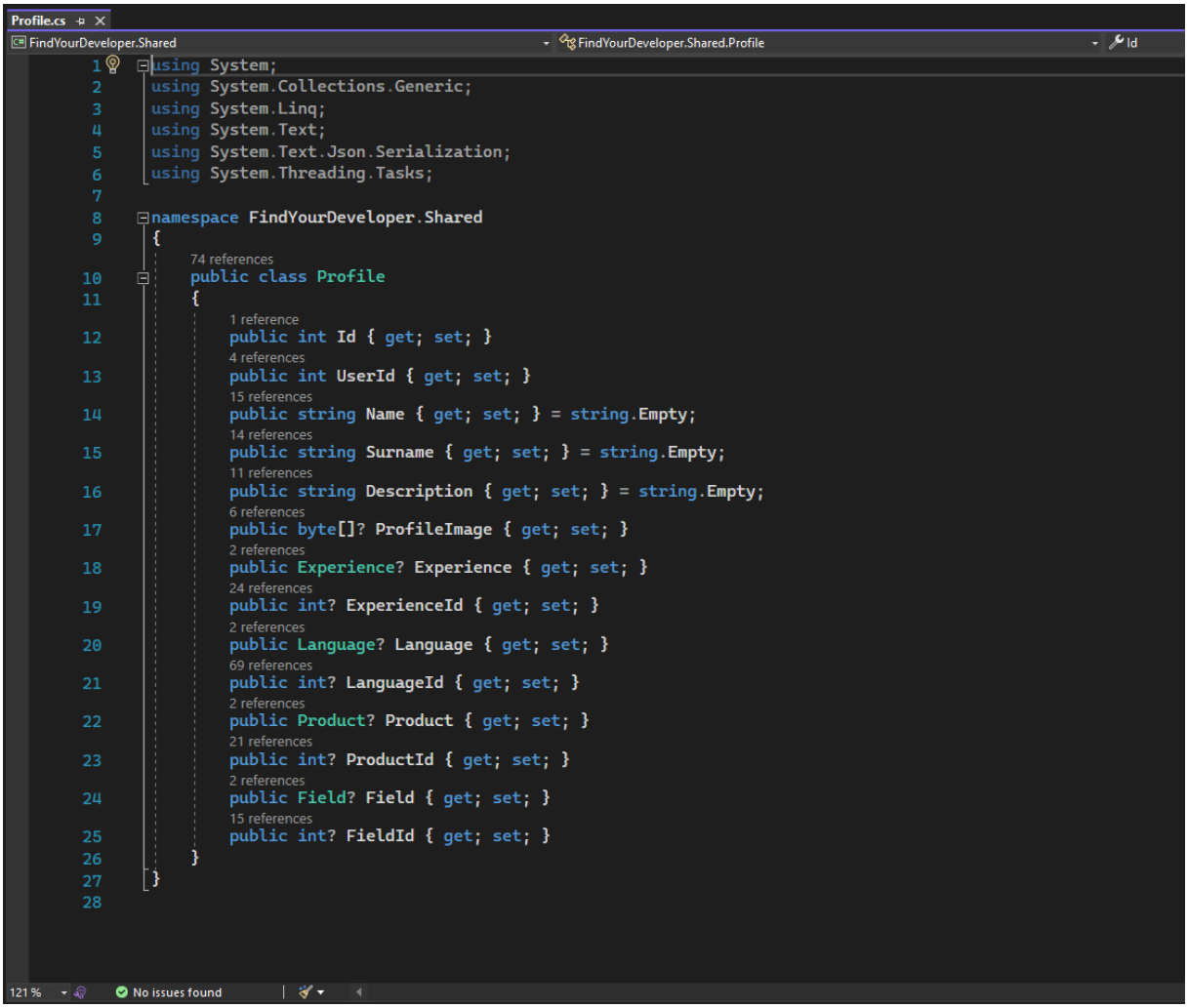
vrijednosti koje aplikacija koristi. Datoteka "appsettings.json" sadrži osnovne postavke koje se primjenjuju na cijelo okruženje aplikacije. Ona definira globalne konfiguracijske vrijednosti koje se koriste tijekom izvršavanja aplikacije. S druge strane, datoteka "appsettings.Development.json" je specifična za razvojno okruženje i sadrži dodatne postavke koje se primjenjuju samo tijekom razvoja aplikacije. Ova datoteka omogućuje podešavanje razvojnih parametara poput razine dnevnika (log level), konfiguracije baze podataka i drugih postavki koje su specifične za razvojno okruženje. Upotreba odvojenih datoteka za konfiguraciju omogućuje fleksibilnost tijekom razvoja i omogućuje različite postavke za različita okruženja kao što su razvoj, testiranje i produkcija. Aplikacija će koristiti odgovarajuću datoteku ovisno o okruženju u kojem se izvršava, pri čemu će postavke iz "appsettings.json" biti zamijenjene postavkama iz "appsettings.Development.json" tijekom razvoja.

Zadnji dokument u Server projektu je "Program.cs" i ima ključnu ulogu prilikom pokretanja i konfiguriranja aplikacije. Ova datoteka sadrži glavnu metodu "Main", koja je ulazna točka za izvršavanje aplikacije. U njoj se konfiguriraju različiti dijelovi aplikacije prije njihovog pokretanja. Njeni ključni aspekti su: Konfiguriranje „hosta“, konfiguriranje servisa, konfiguriranje aplikacije i samo pokretanje aplikacije. Ona služi kao glavni kontrolni dio za konfiguraciju i pokretanje server dijela aplikacije.

5.2.3 FindYourDeveloper.Shared

"Shared" projekt predstavlja zajednički prostor koji je dostupan i "Client" i "Server" projektima. U tom projektu se nalaze razni resursi, komponente i logika potrebni za oba dijela aplikacije. Ovo smanjuje dupliranje koda, čime programerima omogućava efikasniji i brži razvoj aplikacija, a samu aplikaciju čini lakšom za održavanje jer je bolje organizirana. Glavna uloga ovog projekta je modeliranje podataka koji se koriste na klijentskoj i serverskoj strani aplikacije. To omogućava dosljedno i jednostavno dijeljenje podataka između klijenta i servera te olakšava upravljanje podacima u cijeloj aplikaciji. Također, cijele komponente mogu se dijeliti ako su spremljene u ovom projektu, ukoliko je to potrebno za dijeljenje između ova dva projekta. Osim toga, mogu se dijeliti i logika ili funkcionalnosti između ova dva projekta, kao što su pristupi podacima, vanjski API-ji i druge korisne utilitarne funkcionalnosti.

U ovome projektu postoji nekoliko datoteka i one su sve iste naravi. Njihova svrha je da modeliraju podatke iz baze podataka. Tj. da prikažu te podatke jednako klijentu i server aplikaciji. Dakle osim kompletnih i detaljnih podataka iz baze tu su i neke kombinacije podatak koje mogu poslužiti u oba projekta. Tako npr. imamo „UserLogin.cs“ za kojeg ne postoji tablica u bazi podataka, gdje korisnik unosi vrijednosti i popunjava ta dva polja „Email“ i „Password“ te se onda uspoređuje sa tablicom u bazi podataka. Tako da imamo više vrsta modela u ovome projektu, a to su oni koji su točno zapisani u tablici i one koje koristimo trenutno za uspoređivanje ili pak ispisa greški iz baze podataka („ServiceResponse.cs“).



Slika 27: Snimka zaslona rzvojnog okruženja, kod modela profila

Izvor: Izrada autora

6. Zaključak

Web aplikacije igraju ključnu ulogu u današnjem digitalnom okruženju, pružajući korisnicima jednostavan pristup različitim uslugama i sadržajima putem interneta. Razvoj tehnologija koje podupiru web aplikacije neprestano napreduje, a njihova sve veća dostupnost omogućava programerima stvaranje inovativnih i konkurentnih aplikacija. Te tehnologije uključuju programski jezik JavaScript, koji je bez sumnje najpopularniji jezik za web razvoj. JavaScript, zajedno s brojnim web okvirima i bibliotekama, predstavlja kamen temeljac modernih web aplikacija. Popularni okviri kao što su React, Angular i Vue.js omogućavaju programerima da brzo razvijaju složene aplikacije uz pomoć JavaScripta.

Web razvoj također vidi uspon novih tehnologija kao što je Blazor, koji donosi alternativu JavaScriptu. Iako nije među najpopularnijim web okvirima, Blazor ima svoju snagu. Primjećuje se rast njegove popularnosti, što je rezultat njegovih inovativnih značajki i praktičnosti. Važno je napomenuti da je Blazor razvijen od strane Microsofta, što pruža pouzdanost i sigurnost korisnicima. Ovo je posebno značajno s obzirom na širok spektar usluga koje Microsoft pruža u IT industriji. Korisnici i razvijatelji mogu se osloniti na Blazor i očekivati kontinuiranu podršku i nadogradnje. To osigurava svijetlu budućnost za ovaj alat. Detaljna dokumentacija, kvaliteta proizvoda i rastuća zajednica pružaju jamstvo da učenje Blazora i povezanih alata donosi stabilan i perspektivan put za web razvoj. (Duckett, 2014)

Literatura

A., J., 2023. *What Is React & How Does It Actually Work?: Hostinger.* [Mrežno] Available at: https://www.hostinger.com/tutorials/what-is-react [Pristupano 15 svibanj 2023].

Arritola, T. & Watterson, K., 2022. *Blazor in action.* 1st ur. New York: Manning Publications.

Astari, S., 2023. *What Is PHP? Learning All About the Scripting Language: Tutorials: Hostinger:.* [Mrežno] Available at: www.hostinger.com/tutorials/what-is-php/ [Pristupano 1 lipanj 2023].

Bernasconi, C., 2023. *Blazor Basics: What is Blazor—Introduction to Blazor Development: Telerik.* [Mrežno] Available at: https://www.telerik.com/blogs/blazor-basics-introduction-blazor-development [Pristupano 17 svibanj 2023].

Bigelow, S. J., 2022. *Microsoft Azure: Techtarget.* [Mrežno] Available at: https://www.techtarget.com/searchcloudcomputing/definition/Windows-Azure [Pristupano 22 svibanj 2023].

Bluein, C., 2020. *List of Top Mobile and Web Applications Built on Java: Openxcell.* [Mrežno] Available at: https://www.openxcell.com/blog/applications-of-java/ [Pristupano 14 svibanj 2023].

Brewster, C., 2022. *What Is Vue.js? The Pros and Cons of Vue.js in 2023: trio.* [Mrežno] Available at: https://www.trio.dev/blog/why-use-vue-js [Pristupano 15 svibanj 2023].

Chris, K., 2021. *What is PHP? The PHP Programming Language Meaning Explained: FreeCodecamp.* [Mrežno] Available at: https://www.freecodecamp.org/news/what-is-php-the-php-programming-language-meaning-explained/ [Pristupano 13 svibanj 223].

Coursera authors, 2023. *Ruby vs. Python: Pros, Cons, and Where to Start: Coursera.* [Mrežno] Available at: https://www.coursera.org/articles/ruby-vs-python [Pristupano 14 svibanj 2023].

Coursera authors, 2023. *What Is Python Used For? A Beginner's Guide: Coursera.* [Mrežno] Available at: https://www.coursera.org/articles/what-is-python-used-for-a-beginners-

guide-to-using-python
[Pristupano 14 svibanj 2023].

Crockford, D., 2017. *JavaScript: The Good Parts.* 4TH ur. s.l.:CreateSpace Independent Publishing Platform.

Daftari, S., 2022. *10 Reasons Why Django Web Development with Python is Most Popular for Backend Web Development: Kellton.* [Mrežno] Available at: https://www.kellton.com/kellton-tech-blog/why-django-web-development-with-python-for-backend-web-development
[Pristupano 17 svibanj 2023].

Developer authors, 2023. *Swift: Developer.* [Mrežno] Available at: https://developer.apple.com/swift/
[Pristupano 14 svibanj 2023].

Duckett, J., 2014. *Web design with HTML, CSS, Javascript and Jquery set.* 1st ur. s.l.:Wiley.

Dyachenko, A., 2022. *Why Use Angular For Web Development? Everything You Should Know About Angular Development Services: Cadabra.* [Mrežno] Available at: https://cadabra.studio/blog/why-use-angular-development-services-pros-cons/
[Pristupano 15 svibanj 2023].

Enge, E., 2021. *Mobile vs Desktop Usage in 2020: Perficient.* [Mrežno] Available at: https://www.perficient.com/insights/research-hub/mobile-vs-desktop-usage
[Pristupano 31 svibanj 2023].

Engine Yard Team, 2023. *Ruby On Rails Web Development: Engine Yard.* [Mrežno] Available at: https://www.engineyard.com/blog/ruby-on-rails-web-development/
[Pristupano 16 svibanj 2023].

Felke-Morris, T., 2017. *Web Development and Design Foundations with HTML5.* 4th ur. s.l.:Pearson.

Haverbeke, M., 2018. *Eloquent JavaScript: A Modern Introduction to Programming.* 3rd ur. San Francisco: No Starch Press.

Holden, S., 2018. *The Python Wiki: Wiki.Python.* [Mrežno] Available at: https://wiki.python.org/moin/
[Pristupano 14 svibanj 2023].

Kenton, W., 2022. *What Is Web 2.0? Definition, Impact, and Examples: Investopedia.* [Mrežno]
Available at: https://www.investopedia.com/terms/w/web-20.asp
[Pristupano 7 svibanj 2023].

Kostanjevac, J., 2018. *BRZI RAZVOJ APLIKACIJA U PROGRAMSKOM JEZIKU C#,* Šibenik: Dabar.

MDN contributors, 2023. *What is JavaScript?: MDN web docs.* [Mrežno] Available at: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/What_is_JavaScript
[Pristupano 9 svibanj 2023].

Melville, N., 2018. *The Benefits of jQuery in Front-End Website Development: Wearediagram.* [Mrežno]
Available at: https://www.wearediagram.com/blog/the-benefits-of-jquery-in-front-end-website-development
[Pristupano 17 svibanj 2023].

Microsoft, 2022. *What is .NET Framework?: Microsoft.* [Mrežno]
Available at: https://dotnet.microsoft.com/en-us/learn/dotnet/what-is-dotnet-framework
[Pristupano 30 Kolovoz 2023].

Naveed Fida, V. P. R., 2023. *Introduction to Programming with JavaScript: Launchschool.* [Mrežno]
Available at: https://launchschool.com/books/javascript
[Pristupano 9 svibanj 2023].

N. & Greene, N., 2023. *What is a web framework?: evolve.* [Mrežno]
Available at: https://evolve.ie/q-and-a/what-is-a-web-framework/
[Pristupano 15 svibanj 2023].

Nožinić, M., 2016. *Evo kako je izgledala i čemu je služila prva web stranica: Zimo dnevnik.hr.* [Mrežno]
Available at: https://zimo.dnevnik.hr/clanak/evo-kako-je-izgledala-i-cemu-je-sluzila-prva-web-stranica-2---453481.html
[Pristupano 7 svibanj 2023].

Python Institute, 2021. *Python® – the language of today and tomorrow: Python Institute.* [Mrežno]
Available at: https://pythoninstitute.org/about-python
[Pristupano 14 svibanj 2023].

Ravikiran, A. S., 2023. *How to use Node Js for Backend Web Development in 2023: SimpliLearn.* [Mrežno]
Available at: https://www.simplilearn.com/tutorials/nodejs-tutorial/nodejs-backend
[Pristupano 16 svibanj 2023].

Rippon, C., 2021. *ASP.NET Core 5 and React: Modern full-stack web development using .NET 5, React 17, and TypeScript 4.* 2nd ur. Birmingham: Packt Publishing.

Robbins, J. N., 2012. *Learning Web Design: A Beginner's Guide to HTML, CSS, JavaScript, and Web Graphics.* 4th ur. s.l.:O'Reilly Media.

Roth, D., Anderson, R. & Luttin, S., 2022. *Overview of ASP.NET Core: Microsoft Learn.* [Mrežno]
Available at: https://learn.microsoft.com/en-us/aspnet/core/introduction-to-aspnet-core?view=aspnetcore-7.0
[Pristupano 17 svibanj 2023].

Sharma, A., 2018. *Blazor Quick Start Guide: Build web applications using Blazor, EF Core, and SQL Server.* s.l.:Packt Publishing.

Shubel, M., 2022. *What Is TypeScript?: The NewStack.* [Mrežno] Available at: https://thenewstack.io/what-is-typescript/ [Pristupano 26 svibanj 2023].

Terra, J., 2023. *What is Web 1.0, Web 2.0, and Web 3.0? Definitions, Differences & Similarities: Simplilearn.* [Mrežno] Available at: https://www.simplilearn.com/what-is-web-1-0-web-2-0-and-web-3-0-with-their-difference-article
[Pristupano 7 svibanj 2023].

The Editors of Encyclopaedia Britannica, 2023. *Java: Britannica.* [Mrežno] Available at: https://www.britannica.com/technology/Java-computer-programming-language
[Pristupano 14 svibanj 2023].

Tuama, D. Ó., 2022. *What Is A Programming Language?: CodeInstitute.* [Mrežno] Available at: https://codeinstitute.net/global/blog/what-is-a-programming-language/ [Pristupano 1 lipanj 2023].

Tutorials Point Authors, 2023. *VueJS - Overview: Tutorials Point.* [Mrežno] Available at: https://www.tutorialspoint.com/vuejs/vuejs_overview.htm [Pristupano 15 svibanj 2023].

Vermaak, W., 2022. *What is Web 3.0? Decentralized Internet Explained: Alexandria.* [Mrežno]
Available at: https://coinmarketcap.com/alexandria/article/what-is-web-3-0 [Pristupano 31 svibanj 2023].

Whitmore, R., 2023. *How Is Java Used In Web Development?: OnlineSchoolsReport.* [Mrežno]
Available at: https://www.onlineschoolsreport.com/how-is-java-used-in-web-development/
[Pristupano 14 svibanj 2023].

Wintemute, D. & Blackmon, S., 2023. *The Best Computer Frameworks: Bestcolleges.* [Mrežno]
Available at: https://www.bestcolleges.com/bootcamps/guides/most-popular-web-frameworks/
[Pristupano 13 svibanj 2023].

Popis slika

Sažetak

Razvoj web aplikacija pomoću Blazor tehnologije donosi brojne prednosti i mogućnosti za programere. Blazor je moderna tehnologija koja omogućava izradu interaktivnih web aplikacija koristeći C# i .NET platformu. Umjesto tradicionalnog razdvajanja između klijentske i serverske strane, Blazor omogućava programerima da koriste isti jezik i okruženje za razvoj i izvršavanje aplikacija na obje strane.

Jedna od ključnih prednosti Blazora je sposobnost izvođenja koda na klijentskoj strani koristeći WebAssembly (Wasm). Ovo omogućava izvršavanje C# koda u web pregledniku, što rezultira bržim i interaktivnijim korisničkim sučeljem. Klijentska strana aplikacije može se izvoditi lokalno na korisnikovom uređaju, smanjujući opterećenje na serverskoj strani i poboljšavajući performanse.

Blazor podržava različite pristupe razvoju, uključujući Blazor WebAssembly i Blazor Server. Blazor WebAssembly omogućava potpuno klijentsko izvršavanje aplikacije, dok Blazor Server koristi SignalR za održavanje veze između klijenta i servera. Oba pristupa imaju svoje prednosti i mogu se prilagoditi specifičnim zahtjevima projekta.

Koristeći Blazor, programeri mogu iskoristiti moćan C# jezik i bogatstvo .NET ekosustava za izradu reaktivnih korisničkih sučelja, upravljanje događajima, izvođenje validacije podataka i interakciju s „Back-end“ servisima. Blazor također podržava upotrebu postojećih .NET biblioteka, što olakšava ponovno korištenje koda i integraciju s postojećim aplikacijama.

Razvoj web aplikacija pomoću Blazora donosi i jednostavnost održavanja. Budući da se koristi isti jezik i okruženje na obje strane, programeri imaju jedinstveno iskustvo razvoja i mogu dijeliti kôd, komponente i logiku između klijenta i servera. To rezultira manje dupliciranog koda, većom produktivnošću i lakšim održavanjem aplikacije.

Ključne riječi: Blazor, web aplikacija, C#.

Summary

The development of web applications using Blazor technology brings numerous advantages and possibilities for developers. Blazor is a modern technology that allows the creation of interactive web applications using C# and the .NET platform. Instead of the traditional separation between client-side and server-side, Blazor enables developers to use the same language and environment for development and execution on both sides.

One of the key benefits of Blazor is the ability to run code on the client side using WebAssembly (Wasm). This allows the execution of C# code in the web browser, resulting in faster and more interactive user interfaces. The client side of the application can be executed locally on the user's device, reducing the load on the server-side and improving performance.

Blazor supports different development approaches, including Blazor WebAssembly and Blazor Server. Blazor WebAssembly enables full client-side execution of the application, while Blazor Server uses SignalR to maintain the connection between the client and the server. Both approaches have their advantages and can be adapted to specific project requirements.

By using Blazor, developers can leverage the power of the C# language and the richness of the .NET ecosystem to create reactive user interfaces, handle events, perform data validation, and interact with backend services. Blazor also supports the use of existing .NET libraries, making it easy to reuse code and integrate with existing applications.

The development of web applications using Blazor also brings simplicity to maintenance. Since the same language and environment are used on both sides, developers have a unified development experience and can share code, components, and logic between the client and the server. This results in less duplicated code, increased productivity, and easier maintenance of the application.

Keywords: Blazor, web applications, C#