Metode autentikacije i autorizacije u web aplikacijama

Jurič, Tomislav

SVEUČILIŠTE U SPLITU
PRIRODOSLOVNO MATEMATIČKI FAKULTET

ZAVRŠNI RAD

METODE AUTENTIKACIJE I AUTORIZACIJE U WEB APLIKACIJAMA

Tomislav Jurič

Split, rujan 2024.

„Zahvaljujem se mojim roditeljima, mentoru, prijateljima, kolegama i svima onima koji su mi bili neizmjerna podrška tijekom studiranja“

Temeljna dokumentacijska kartica

Završni rad

Sveučilište u Splitu
Prirodoslovno-matematički fakultet
Odjel za informatiku
Ruđera Boškovića 33, 21000 Split, Hrvatska

Metode autentikacije i autorizacije u web aplikacijama

Tomislav Jurič

SAŽETAK

Cilj ovog rada je predstaviti različite metode autentikacije i autorizacije u web aplikacijama, implementirati ih te na temelju njihove usporedbe iznijeti zaključak o svakoj predstavljenoj i implementiranoj metodi. U ovom radu predstavit će se i tehnologije i alati koji su se koristili za implementaciju korištenih metoda.

Ključne riječi: sesija, token, Auth0, Google, React, JavaScript, Node.js, Express.js, JWT
Rad je pohranjen u knjižnici Prirodoslovno-matematičkog fakulteta, Sveučilišta u Splitu

Rad sadrži: 35 stranica, 27 slika i 14 literaturnih navoda. Izvornik je na hrvatskom jeziku.

Mentor: **Dr. sc. Goran Zaharija**, docent Prirodoslovno-matematičkog fakulteta, Sveučilišta u Splitu

Ocjenjivači: **Dr. sc. Goran Zaharija**, docent Prirodoslovno-matematičkog fakulteta, Sveučilišta u Splitu

Dr. sc. Divna Krpan, docent Prirodoslovno-matematičkog fakulteta, Sveučilišta u Splitu

Dr. sc. Monika Mladenović, docent Prirodoslovno-matematičkog fakulteta, Sveučilište u Splitu

Rad prihvaćen: **rujan 2024.**

Basic documentation card

Thesis

University of Split
Faculty of Science
Department od Informatics
Ruđera Boškovića 33, 21000 Split, Croatia

Authentication and Authorization Methods in Web Applications

Tomislav Jurič

ABSTRACT

The goal of this thesis is to present different methods of authentication and authorization in web applications, to implement them and, based on their comparison, to present a conclusion about each presented and implemented method. This thesis will present the technologies and tools that were used to implement the methods used.

Key words: session, token, Auth0, Google, React, Javascript, Node.js, Express.js, JWT
Thesis deposited in library of Faculty of Science, University of Split

Thesis consist of: 35 pages, 27 figures and 14 references. Original language: Croatian

Supervisor: **Goran Zaharija, Ph.D.** Assistant Professor of Faculty of Science, University of Split

Reviewers: **Goran Zaharija, Ph.D.** Assistant Professor of Faculty of Science, University of Split

Divna Krpan, Ph.D. Assistant Professor of Faculty of Science, University of Split

Monika Mladenović, Ph.D. Assistant Professor of Faculty of Science, Univeristy of Split

Thesis accepted: **September, 2024.**

IZJAVA

kojom izjavljujem s punom materijalnom i moralnom odgovornošću da sam završni rad s naslovom METODE AUTENTIKACIJE I AUTORIZACIJE U WEB APLIKACIJAMA izradio samostalno pod voditeljstvom dr. sc., Gorana Zaharije, U radu sam primijenio metodologiju znanstvenoistraživačkog rada i koristio literaturu koja je navedena na kraju završnog rada. Tuđe spoznaje, stavove, zaključke, teorije i zakonitosti koje sam izravno ili parafrazirajući naveo u završnom radu na uobičajen, standardan način citirao sam i povezao s fusnotama s korištenim bibliografskim jedinicama. Rad je pisan u duhu hrvatskog jezika.

Student

Tomislav Jurič

Sadržaj

Uvod

Autentikacija i autorizacija su dva vitalna procesa informacijske sigurnosti koje administratori koriste za zaštitu sustava i informacija. Autentikacijom se provjerava identitet korisnika ili usluge, a autorizacijom se određuju njihova prava pristupa. Iako ova dva pojma zvuče slično, igraju različite, ali jednako bitne uloge u osiguravanju aplikacija i podataka. Razumijevanje razlika je ključno. U kombinaciji, oni određuju sigurnost sustava. Ne možete imati sigurno rješenje ako niste ispravno konfigurirali autentikaciju i autorizaciju. [1]

Autentikacija i autorizacija se koriste u svakodnevnom životu. Jedan od primjera gdje se koristi je odlazak na let zrakoplovom. Autentikacija se koristi kada osoba koja je kupila kartu za let prolazi sigurnosnu provjeru u zračnoj luci pokazujući osobnu iskaznicu kako bi potvrdila svoj identitet. Ako sigurnosna provjera bude uspješna, osobi se da karta za ukrcavanje. Autorizacija se koristi kada osoba pokaže svoju kartu za ukrcavanje stjuardesi kako bi se mogla ukrcati na određeni zrakoplov kojim treba letjeti. Stjuardesa mora ovlastiti osobu da se može ukrcati na let i ući u zrakoplov.

Danas gotovo svaka web stranica ima mogućnost prijave korisnika. Ovaj rad će se fokusirati na pregled najčešće korištenih načina u web developmentu.

1. Osnovni pojmovi

1.1. Autentikacija

Autentikacija(eng. *Authentication*) je proces koji potvrđuje da je netko ili nešto ono za što se predstavlja. Tehnološki sustavi obično koriste neki oblik provjere autentičnosti kako bi osigurali pristup aplikaciji ili njenim podacima. Na primjer, kada trebate pristupiti online stranici ili usluzi, obično morate unijeti svoje korisničko ime i lozinku. Zatim, iza kulisa, uspoređuje korisničko ime i lozinku koje ste unijeli sa zapisom koji ima u svojoj bazi podataka. Ako se podaci koje ste poslali podudaraju, sustav pretpostavlja da ste važeći korisnik i odobrava vam pristup. Autentikacija sustava u ovom primjeru pretpostavlja da samo vi znate ispravno korisničko ime i lozinku. Stoga vam daje autentikaciju korištenjem principa nečega što samo vi znate. [1]

Svrha autentikacije je potvrditi da je netko ili nešto ono ili što tvrdi da jest. Postoje mnogi oblici autentikacije. Na primjer, svijet umjetnosti ima procese i institucije koje potvrđuju da je slika ili skulptura djelo određenog umjetnika. Isto tako, vlade koriste različite tehnike provjere autentičnosti kako bi zaštitile svoju valutu od krivotvorenja. Tipično, provjera autentičnosti štiti vrijedne predmete, a u informacijskom dobu štiti sustave i podatke.

Autentikacija identiteta je proces provjere identiteta korisnika ili usluge. Na temelju tih informacija sustav zatim korisniku daje odgovarajući pristup. Na primjer, recimo da imamo dvoje ljudi koji rade u kafiću, Luciju i Marka. Lucija je voditeljica kafića, dok je Marko konobar. Kafić koristi POS (eng. *Point of Sale*) sustav gdje konobari mogu staviti narudžbe. U ovom primjeru, POS bi koristio neki postupak za provjeru identiteta Lucije ili Marka prije nego što im dopusti pristup sustavu. Na primjer, može ih tražiti korisničko ime i lozinku ili će možda trebati skenirati palac na čitaču otiska prsta. Budući da kafić mora osigurati pristup svom POS-u, zaposlenici koji koriste sustav moraju potvrditi svoj identitet putem postupka autentikacije.

Osiguravanje autentičnosti subjekta može se odrediti pomoću tri čimbenika. Što više faktora subjekt može pružiti, to više povjerenja možete dati u taj subjekt: [2]

- nešto što znate - stavka koje je subjekt svjestan ili o kojoj zna
- nešto što imate - stavka koju subjekt posjeduje
- nešto što jeste - karakteristika subjekta

Posjedovanje specifičnog znanja koje je jedinstveno za subjekt jedna je od metoda provjere autentičnosti. Primjeri ovog faktora uključuju lozinku, osobni identifikacijski broj (PIN) ili zaporku. Lozinka se općenito kombinira s jedinstvenim identifikatorom kao što je korisničko ime (ili ID korisnika) i pruža dodatnu provjeru autentičnosti subjekta. Lozinka može imati višestruka ograničenja na temelju duljine, posebnih znakova, složenosti i faktora ponovne upotrebe ili može imati nikakva ograničenja. Što je lozinka specifičnija ili više jedinstvena, to je jača i stoga ima manje šanse da bude pogođena ili probijena. Administratori sustava određuju karakteristike lozinki, ali subjekti moraju zapamtiti lozinke. To je obično izazov jer se lozinke nikad ne smiju zapisivati.

Osim nečega što znate, nešto što „imate“ može pomoći u vašoj identifikaciji i/ili dokazivanju vaše tvrdnje o identitetu. Taj identifikator može biti kreditna kartica (bankomat), token, vozačka dozvola ili putovnica - sve što podržava vašu tvrdnju o identitetu jednostavno zato što to imate. Ovi oblici provjere autentičnosti ne zahtijevaju da zapamtite lozinku, ali oni su nešto što morate imati u svom posjedu za provjeru autentičnosti. Razmotrite primjer u kojem ste posjetili banku i zatražili povlačenje sredstava. Ne možete jednostavno prići šalteru i reći „Ja sam Bob, molim vas dajte mi 500 dolara.“ Osoba koja radi tamo će vas sigurno tražiti da dokažete svoj identitet. Najvjerojatnije biste udovoljili ovom zahtjevu tako da joj pokažete svoju vozačku dozvolu. Licenca sadrži vaše ime i sliku, a osoba ih koristi za provjeru autentičnosti prije nego što vam da gotovinu.

Provjera autentičnosti "nešto što jeste" temelji se na karakteristikama određene osobe. Karakteristike mogu biti glas, crte lica, uzorci mrežnice, rukopis i radnje koje se ponavljaju. Ove karakteristike se nazivaju biometrija. Tehnologija koja stoji iza biometrije uključuje skeniranje i analizu jedinstvenih karakteristika korisnika i njihovo uspoređivanje s informacijama prikupljenim tijekom upisa. Podaci o pojedincima mogu se koristiti za identifikaciju ili provjeru. [2]

1.2. Autorizacija

Autorizacija (eng. *Authorization*) je sigurnosni proces koji određuje razinu pristupa korisnika ili usluge. U tehnologiji koristimo autorizaciju kako bismo korisnicima ili uslugama dali dopuštenje za pristup nekim podacima ili izvođenje određene radnje. Ako ponovno pogledamo naš primjer kafića, Marko i Lucija imaju različite uloge u kafiću. Kako je Marko konobar, on može samo postavljati i pregledavati narudžbe. Lucija, s druge strane, u svojoj ulozi upraviteljice također može imati pristup ukupnim dnevnim prodajama. Budući da Marko i

Lucija imaju različite poslove u kafiću, sustav bi koristio njihov provjereni identitet kako bi svakom korisniku dao individualna dopuštenja. Ovdje je bitno uočiti razliku između autentikacije i autorizacije. Autentikacija provjerava korisnika (Lucija) prije nego što mu se dopusti pristup, a autorizacija određuje što mogu učiniti nakon što im sustav odobri pristup (pregled informacija o prodaji). [1]

Tumačenje autorizacije kao „odobrenja pristupa“ može se pratiti do koncepta AAA (eng. *Authentication, Authorization, Accounting*) koji je implementiran u mrežnom protokolu RADIUS (eng. *Remote Authentication Dial-In User Service*). Ovaj mrežni protokol podržava centralizirano upravljanje autentikacijom, kontrolom pristupa i računovodstvom za korisnike koji pristupaju mrežnim uslugama. RADIUS je razvio Livingston Enterprises, Inc. 1991. kao protokol za autentikaciju pristupnog poslužitelja i obračunski protokol, a kasnije je definiran kao IETF standard RFC2865 u 2000. Kako je RADIUS imao neke nedostatke, IETF je definirao novi i napredniji protokol pod nazivom Diameter kao RFC6733 u 2012. godini. [3]

Sustavi autorizacije postoje u mnogim oblicima u tipičnom tehnološkom okruženju. Na primjer, Liste kontrole pristupa (eng. ACL – *Access Control Lists*) određuju koji korisnici ili usluge mogu pristupiti određenom digitalnom okruženju. Ovu kontrolu pristupa ostvaruju provođenjem pravila dopuštanja ili zabranjivanja na temelju razine autorizacije korisnika. Na primjer, na bilo kojem sustavu obično postoje opći korisnici i super korisnici ili administratori. Ako standardni korisnik želi napraviti promjene koje utječu na njegovu sigurnost, ACL može zabraniti pristup. S druge strane, administratori imaju ovlaštenje za sigurnosne promjene, pa će im ACL to dopustiti.

Druga uobičajena vrsta autorizacije je pristup podacima. U svakom poslovnom okruženju obično imate podatke s različitim razinama osjetljivosti. Na primjer, možete imati javne podatke koje pronađete na web stranici tvrtke, interne podatke koji su dostupni samo zaposlenicima i povjerljive podatke kojima može pristupiti samo nekolicina pojedinaca. U ovom primjeru autorizacija određuje koji korisnici mogu pristupiti različitim vrstama informacija.

1.3. Odnosi autentikacije i autorizacije

Kao što je spomenuto, autentikacija i autorizacija mogu zvučati slično, ali svaka igra različitu ulogu u osiguravanju sustava i podataka. Nažalost, ljudi često koriste oba pojma naizmjenično jer se oba odnose na pristup sustavu. Međutim, to su različiti procesi. Jednostavnije rečeno, jedan provjerava identitet korisnika ili usluge prije nego što im se odobri pristup, dok drugi određuje što mogu učiniti kada dobiju pristup. [1]

Najbolji način da se ilustriraju razlike između ta dva pojma je jednostavan primjer. Recimo da ste odlučili otići i posjetiti dom prijatelja. Po dolasku pokucate na vrata, a prijatelj vam otvori. On vas prepoznaje (provjera autentičnosti) i pozdravlja vas. Budući da je provjera autentičnosti bila uspješna, sada vas prijatelj može pustiti u svoj dom. Međutim, na temelju vašeg odnosa, postoje određene stvari koje možete učiniti, a druge ne možete (autorizacija). Na primjer, možete ući u kuhinju, ali ne možete u njegov privatni ured. Drugim riječima, imate ovlaštenje za ulazak u kuhinju, ali pristup njegovom privatnom uredu je zabranjen.

Autentikacija i autorizacija slične su po tome što su dva dijela temeljnog procesa koji omogućuje pristup. Posljedično, ta se dva pojma često brkaju u informacijskoj sigurnosti jer dijele istu kraticu "auth". Autentikacija i autorizacija također su slične u načinu na koji obje iskorištavaju identitet. Na primjer, jedan provjerava identitet prije odobravanja pristupa, dok drugi koristi ovaj potvrđeni identitet za kontrolu pristupa.

Postavlja se pitanje što dolazi prije, autentikacija ili autorizacija? Odgovor bi bio da i autentikacija i autorizacija se oslanjaju na identitet. Kako ne možete autorizirati korisnika ili uslugu prije nego što ih identificirate, autentikacija uvijek dolazi prije autorizacije. Ponovno se možemo pozvati na naš primjer kafića kako bismo ilustrirali ovu tvrdnju.

Kao što je spomenuto, konobari mogu samo kreirati i pregledavati narudžbe, dok menadžeri također mogu pristupiti dnevnim podacima o prodaji. Ako POS sustav ne može identificirati koji korisnik pristupa sustavu, ne može pružiti ispravnu razinu pristupa. Autentikacija pruža potvrđeni identitet potreban autorizaciji za kontrolu pristupa. Kada se Marko ili Lucija prijave u sustav, aplikacija zna tko se prijavio i koju ulogu treba dodijeliti njihovom identitetu.

1.4. Modeli kontrole pristupa

Modeli kontrole pristupa jezgra su koja identificira kako korisnik pristupa objektu. Poduzeće određuje najbolji model na temelju toga kako je njegova organizacija strukturirana, politika unutar organizacije te koristi i rizika povezanih s implementacijom. Dostupno je nekoliko modela: [2]

- Obavezna kontrola pristupa (eng. MAC – *Mandatory Access Control*) – politika koju definira sustav
- Diskrecijska kontrola pristupa (eng. DAC - *Discretionary Access Control*) - politika definirana od strane vlasnika objekta

- Kontrola pristupa temeljena na ulogama (eng. RBAC - *Role-Based Access Control*) - politika definirana funkcijama koje korisnik obavlja unutar organizacije, na primjer, uloge mogu biti ljudski resursi ili financije
- Kontrola pristupa temeljena na atributima (eng. ABAC - *Attribute-based Access Control*) - politika je funkcija karakteristika subjekta

1.4.1. MAC

MAC omogućuje administratoru sustava održavanje sigurnosnog aspekta objekta. Uspostavio ga je TCSEC (eng. *Trusted Computer System Evaluation Criteria*) i definiran je kao "sredstvo za ograničavanje pristupa objektima na temelju osjetljivosti (kao što je predstavljeno oznakom) informacija sadržanih u objektima i formalnih ovlaštenja (tj. odobrenja) subjekata za pristup informacijama takve osjetljivosti. MAC se često koristi unutar državnih sustava. Njihove oznake osjetljivosti su Strogo povjerljivo (najviša), Tajno, Povjerljivo i Neklasificirano (najniža).

Pristup objektu temelji se na osjetljivosti objekta u odnosu na predmet. Pristup objektu povezan je s korisnikom koji mu pokušava pristupiti. Na primjer, ako objekt ima oznaku Tajno, subjekt koji pokušava pristupiti objektu mora imati dozvolu Tajno ili Strogo povjerljivo. Objektu nisu pridruženi ACL-ovi, a ni objekt ni korisnik sustava ne mogu promijeniti razinu osjetljivosti. Slično, subjekt s dopuštenjem za Strogo povjerljivo ima pristup objektu koji je na ili ispod razine dopuštenja. MAC se smatra jednom od najsigurnijih metoda pristupa jer zahtijeva da i objekt i subjekt imaju dodijeljene sigurnosne oznake. Često se koristi u višerazinskom sigurnosnom sustavu (MLS – eng. *multi-level security*). MLS sustav omogućuje računalnom sustavu da istovremeno obrađuje podatke različitih razina tajnosti i osigurava subjektu s ispravnim dopuštenjem pristup samo informacijama na njegovoj ili njezinoj razini ovlaštenja. Nasuprot tome, višestruko jednorazinsko okruženje (MSL – eng. *multiple single level*) ne dopušta miješanje različitih razina klasifikacije. Za svaku razinu klasifikacije koristio bi se zaseban sustav.

1.4.2. DAC

DAC model je najčešće korištena metoda kontrole pristupa. TCSEC ga definira ga kao „sredstvo za ograničavanje pristupa objektima na temelju identiteta subjekata i/ili grupa kojima pripadaju. Kontrole su diskrecijske u smislu da subjekt s određenim dopuštenjem pristupa može prenijeti to dopuštenje (možda neizravno) bilo kojem drugom subjektu (osim ako nije ograničen obveznom kontrolom pristupa).“

DAC omogućuje vlasniku resursa da upravlja tko može ili ne može pristupiti stavci. Vlasnici održavaju ovaj pristup putem ACL-ova i mogu dodijeliti delegatu pristup za prilagodbu ovih dopuštenja. Ovo uklanja potrebu da administratori sustava određuju važnost dokumenta i tko bi trebao imati potrebnu kontrolu. Stavlja odgovornost u ruke vlasnika resursa. Osim nekih visoko specijaliziranih slučajeva u obrambenoj industriji, svaki moderni operativni sustav podržava DAC.

1.4.3. RBAC

Kontrola pristupa temeljena na ulogama (RBAC) poznata je i kao ne-diskrecijska kontrola pristupa. Omogućuje pristup objektu na temelju uloge subjekta unutar sustava. Tri aspekta se uzimaju u obzir unutar RBAC sustava:

- Dodjela uloge: Subjekt može izvršiti transakciju samo ako je subjekt odabrao ili mu je dodijeljena uloga. Svi aktivni korisnici moraju imati aktivnu ulogu. Na primjer, ako je korisniku Kevinu dodijeljena uloga ljudski resursi, dopušteno mu je izvršavanje samo radnji koje ta uloga dopušta.
- Ovlaštenje uloge: Aktivna uloga subjekta mora biti ovlaštena za subjekt. Ovo osigurava da korisnici mogu preuzeti samo uloge za koje su ovlašteni.
- Autorizacija transakcije: Subjekt može izvršiti transakciju samo ako je transakcija ovlaštena za aktivnu ulogu subjekta.

Administriranje pristupa unutar RBAC sustava smatra se lakšim za administratora jer se pristup temelji na ulogama unutar organizacije i onome što svaka uloga smije raditi. Na primjer, administrator može definirati ulogu ljudskih resursa za cijelu njihovu organizaciju. Ako Kevin prijeđe iz odjela za ljudske resurse u odjel za financije, on se jednostavno uklanja iz uloge ljudskih resursa i postavlja na ulogu financija. Razdvajanje dužnosti proširuje kontrole RBAC-a. Na primjer, iako Kevinova uloga može biti financije, to ne znači da mu treba puni pristup svim financijskim podacima. Razdvajanje svake uloge na aktivnosti za koje su korisnici odgovorni omogućuje precizniju kontrolu pristupa. Ovo osigurava da niti jedan korisnik nema dovoljno kontrole da ugrozi sustav. Ovaj mehanizam pomaže u sprječavanju prijevara, osiguravajući da su najmanje dvije osobe potrebne za obavljanje kritičnog zadatka. Razdvajanje dužnosti također je povezano s načelom sigurnosti najmanje privilegija. Ovo načelo kaže da korisnik ne bi trebao imati više pristupa nego što mu je potrebno za obavljanje njegovog ili njezinog posla.

1.4.4. ABAC

Sustavi koji se temelje na kontroli pristupa temeljeni na atributima odobravaju pristup subjektu na temelju dodatnih atributa koje moraju provjeriti. Na primjer, kada pristupate sustavu koji je dostupan samo stanovnicima određenog grada, subjekt će možda morati unijeti adresu unutar tog grada. To omogućuje administratoru da ima precizniju mogućnost kontrole pristupa određenim objektima.

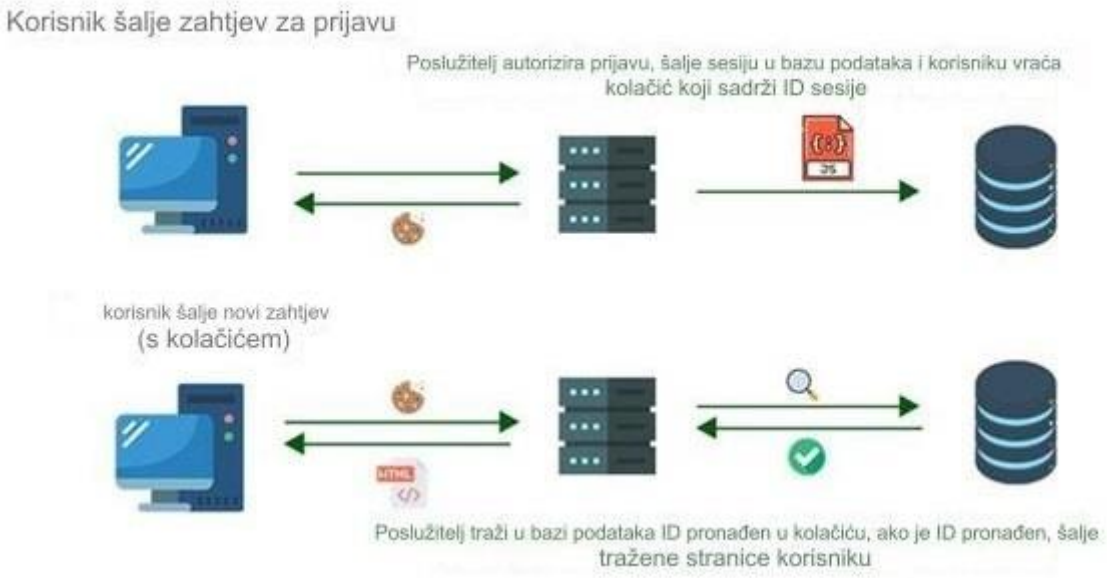
2. Vrste korištene autentikacije

2.1. Autentikacija temeljena na sesiji

Provjera autentičnosti temeljena na sesiji (eng. *Session-based authentication*) je tehnika provjere autentičnosti u kojoj koristimo sesije za praćenje autenticiranog korisnika. Sesija je mala datoteka, najvjerojatnije u JSON (eng. *JavaScript Object Notation*) formatu, koja pohranjuje informacije o korisniku, poput jedinstvenog ID-a, vremena prijave i isteka, itd. Generira se i pohranjuje na poslužitelju kako bi poslužitelj mogao pratiti zahtjeve korisnika. Korisnik prima neke od ovih podataka, posebno ID, kao kolačiće koji će biti poslani sa svakim novim zahtjevom, tako da poslužitelj može prepoznati ID i autorizirati zahtjeve korisnika. [4]

Evo kako radi tehnika upravljanjem sesijom:

1. Korisnik šalje poslužitelju zahtjev za prijavu.
2. Poslužitelj provjerava autentičnost zahtjeva za prijavu, šalje sesiju u bazu podataka i korisniku vraća kolačić koji sadrži ID sesije.
3. Sada korisnik šalje nove zahtjeve (s kolačićem).
4. Poslužitelj provjerava u bazi podataka za ID pronađen u kolačiću, ako je ID pronađen šalje tražene stranice korisniku.



Slika 1.Princip rada autentikacije temeljene na sesiji

Budući da su sesije pohranjene na poslužitelju, njegovi administratori imaju moć nad njima. Na primjer, ako sigurnosni tim posumnja da je račun ugrožen, može odmah poništiti ID sesije, tako da se korisnik odmah odjavi. S druge strane, budući da je sesija pohranjena na poslužitelju, poslužitelj je zadužen za traženje ID-a sesije koji korisnik šalje. To može uzrokovati probleme sa skalabilnošću. Kolačići mogu biti izloženi napadima krivotvorenja zahtjeva između stranica. Napadač može zavesti korisnika na neprijateljsku web stranicu, gdje neke JS skripte mogu iskorištavati kolačiće za slanje zlonamjernih zahtjeva poslužitelju. Još jedna ranjivost odnosi se na šanse za napad "čovjek u sredini" (eng. man-in-the-middle), gdje napadač može presresti ID sesije i izvršiti štetne zahtjeve prema poslužitelju.

2.2. Autentikacija na temelju tokena

Autentikacija na temelju tokena (eng. *Token-based authentication*) je postupak provjere identiteta provjerom tokena. U upravljanju pristupom, poslužitelji koriste autentikaciju tokenom za provjeru identiteta korisnika, API-ja (eng. *Application programming interface*), računala ili drugog poslužitelja. Token je autorizacijska datoteka u koju se ne može ništa mijenjati. Generira ga poslužitelj pomoću tajnog ključa, korisnik ga šalje i pohranjuje u svojoj lokalnoj pohrani. Kao i u slučaju kolačića, korisnik šalje ovaj token poslužitelju sa svakim novim zahtjevom, tako da poslužitelj može provjeriti njegov potpis i autorizirati zahtjeve. [4]

Evo kako radi tehnika autentikacije na temelju tokena:

1. Korisnik šalje poslužitelju zahtjev za prijavu.

2. Poslužitelj autorizira prijavu i šalje token korisniku.
3. Sada korisnik šalje novi zahtjev (s tokenom).
4. Poslužitelj provjerava je li token valjan ili ne, ako je token valjan korisniku šalje tražene stranice.

Slika 2.Autentikacija na temelju tokena (Napomena: Ovo nisu datoteke za provjeru autentičnosti, one su za autorizaciju. Dok prima token, poslužitelj ne traži tko je korisnik, on jednostavno autorizira zahtjeve korisnika oslanjajući se na valjanost tokena.)

Tokeni mogu biti korisni kada korisnik želi smanjiti broj slanja svojih vjerodajnica. U slučaju povezivanja između poslužitelja, korištenje vjerodajnica postaje teško, a tokeni prevladavaju ovaj problem. Štoviše, poslužitelji koji koriste tokene mogu poboljšati svoje performanse jer ne moraju neprestano pregledavati sve detalje sesije kako bi autorizirali zahtjeve korisnika. Međutim, detalji provjere autentičnosti pohranjeni su na klijentu, tako da poslužitelj ne može izvršiti određene sigurnosne operacije kao u metodi sesije. Poslužitelj ne provjerava autentičnost korisnika, tako da povezivanje tokena s korisnikom može biti teže. Ako neki napadač uspije dobiti valjani token, može imati neograničen pristup bazama podataka poslužitelja. Ako poslužitelj generira ključeve pomoću starijih algoritama, ti se ključevi mogu probiti.

2.2.1. JSON Web Token

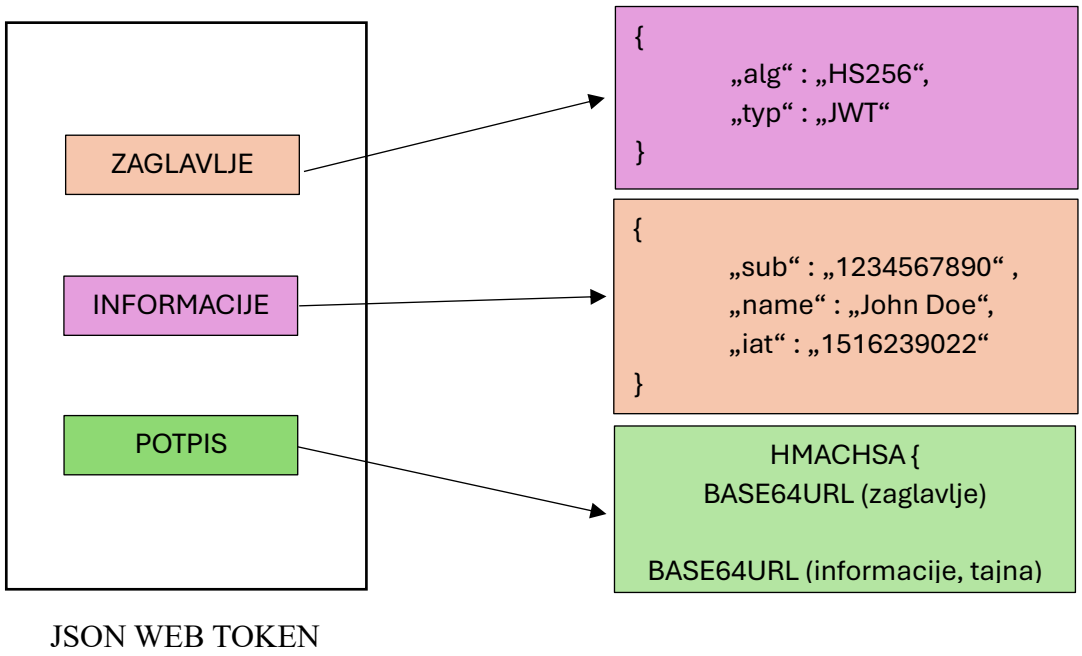
JSON Web Token (JWT) je lagano sredstvo za razmjenu podataka između dviju strana kako bi se olakšala autentikacija, autorizacija i sigurnost. Svaka JWT izjava pohranjena je kao JSON entitet, a svaki JSON entitet se koristi kao otvoreni tekst JSON web enkripcije ili kao sadržaj JSON web potpisa (JWS – eng. *JSON Web Signature*) koji omogućuje digitalnu zaštitu i

autentikaciju zahtjeva s kodom za autentikaciju poruke (MAC – eng. *Message Authentication Code)*.

Prije nekoliko godina, token je bio samo niz bez inherentne vrijednosti prije revolucije JWT-a, npr. „2pWS6RQmdZpE0TQ93X“. Ovaj je token kasnije provjeren na poslužitelju gdje su bile pohranjene izjave tokena. Nedostatak ovog pristupa je da svaki put kada se token koristi, potreban je pristup bazi podataka (ili predmemorija). Danas JWT-ovi kodiraju izjave i provjeravaju vlastite izjave (potpisivanjem). Tako se razvijaju kratkotrajni JWT-ovi bez stanja (samostalni, ne oslanjaju se ni na koga drugog). Nema potrebe za bazom podataka. Time je opterećenje baze podataka eliminirano, a dizajn je pojednostavljen jer samo poslužitelj koji izdaje JWT-ove mora razmišljati o ulasku u bazu podataka/sloj postojanosti (token osvježavanja). [5]

Tri su glavne komponente JWT-a:

1. Zaglavlje (eng. *Header*)
2. Informacije (eng. *Payload*)
3. Potpis (eng. *Signature*)



Slika 3.Struktura JSON Web Tokena

S ove tri komponente, JWT-ovi omogućuju razvojnim programerima da izgrade tok provjere autentičnosti ili autorizacije bez stanja koji je lako skalabilan i eliminira potrebu da poslužitelji održavaju informacije o sesiji. Sva tri dijela su Base64Url kodirani nizovi spojeni točkama ('.').

Zaglavlje je JSON objekt koji obično sadrži dva svojstva: vrstu tokena (JWT) i korišteni algoritam šifriranja (npr. HMAC SHA256, RSA itd.).

Informacije su JSON objekti u kojem žive svi preneseni podaci. Nazivaju se i zahtjevom, ovi podaci obično sadrže informacije o korisniku (korisničko ime, adresa e-pošte), podatke o sesiji (IP adresa, vrijeme ili zadnja prijava) ili dopuštenja za autorizaciju (uloge ili grupe kojima korisnik pripada). [6]

Postoje četiri vrste potraživanja:

- Uobičajeno korišteno: registrirano i javno
- Ne koristi se često: privatno i prilagođeno

Potpis se stvara potpisivanjem Base64Url kodiranog zaglavlja i korisnog sadržaja tajnim ključem i algoritmom koji su odredili programeri. Koristi se za provjeru je li pošiljatelj JWT-a onaj za kojeg se predstavlja i za osiguranje integriteta tokena.

JWT-ovi funkcioniraju kodiranjem skupa zahtjeva u kompaktan, siguran URL string. Ovaj string se može lako prenijeti preko mreže i biti ovjeren od strane primatelja.

Evo općeg pregleda rada JWT-ova:

1. Izdavatelj JWT-a stvara novi JWT objekt i postavlja zahtjeve koje želi uključiti u token.
2. Izdavatelj potpisuje JWT objekt koristeći tajni ključ ili par javni/privatni ključ.
3. Rezultat koji se dobije je JWT koji je kompaktan, siguran URL string koji se onda može prenijeti preko mreže.
4. Primatelj JWT-a provjerava potpis JWT-a koristeći javni ili tajni ključ.
5. Ako je potpis valjan, primatelj može vjerovati zahtjevima u JWT-u.

Korištenje JWT-a ima nekoliko prednosti:

- Podrška za više domena (eng. *Cross-domain support*): Za razliku od kolačića, JWT-ovi se mogu koristiti na različitim domenama i pod-domenama, što ih čini idealnim za implementacije jedinstvene prijave (eng. SSO - *Single-Sing On*)
- Samostalnost i proširivost: budući da JWT već sadrže potrebne informacije o korisniku, smanjuju potrebu za dodatnim upitima prema bazi podataka za korisničke podatke.

Štoviše, JWT-ovi se mogu proširiti prilagođenim zahtjevima za uključivanje dodatnih informacija prema potrebi, što omogućuje veću fleksibilnost.

- Prilagođeno mobilnim uređajima: JWT tokeni izvrstan su izbor za autentikaciju mobilnih aplikacija zbog svoje kompaktne veličine i bez statusa. Omogućuju besprijekornu integraciju s API-jima i mogu uvelike smanjiti opterećenje poslužitelja.
- Poboljšana sigurnost: JWT-ovi se mogu šifrirati radi zaštite osjetljivih podataka, osiguravajući da samo namijenjeni primatelji mogu čitati sadržaj tokena. Štoviše, korištenje digitalnih potpisa osigurava da se token ne mijenja tijekom prijenosa.

Nakon navedenih prednosti postoje i neki nedostatci JWT-ova. Iako su JWT-ovi dobar izbor za aplikacije kojima je potreban kompaktan i jednostavan za korištenje format tokena, najbolje je izbjegavati njihovu upotrebu:

- Kada informacije (eng. *payload*) sadrže osjetljive podatke. JWT-ovi nisu šifrirani, a sadržaj može pročitati svatko tko mu pristupi.
- Kada aplikacija ima stroga ograničenja veličine mrežnih zahtjeva. JWT-ovi mogu postati veliki ako sadrže mnogo zahtjeva.
- Kada je aplikacija ranjiva na napade ponavljanja. JWT-ovi mogu biti ranjivi na napade ponavljanja ako nemaju način da ih spriječe.
- Kada je aplikacija ranjiva na napade „čovjeka u sredini“ (eng. MITM - *man-in-the-middle*). JWT-ovi mogu biti ranjivi na MITM napade ako nisu potpisani pomoću snažnog algoritma.

2.3. Auth0

Auth0 je platforma koju tvrtke i web programeri koriste za provjeru identiteta korisnika prije nego što mu daju pristup web stranicama i aplikacijama. To je fleksibilan, siguran i jednostavan način da pustite prave kupce unutra, dok zlonamjerne i lažne strane držite van.

Iako programeri weba i aplikacija mogu ugraditi alate za upravljanje identitetom korisnika i pristupom (eng. CIAM - *Customer identity and access management*) u vlastite prilagođene platforme, upotreba robusne usluge kao što je Auth0 može znatno olakšati implementaciju i nadzor sigurnosti i usklađenosti. Gotove platforme poput Auth0 atraktivna su rješenja za manje tvrtke koje nemaju vlastitu stručnost, kao i za veće tvrtke koje, zbog svoje veličine i složenosti, imaju koristi od centralizirane, kompatibilne CIAM platforme. [7]

Ključne značajke Auth0:

1. Jedinstvena prijava (eng. *Single Sign-On*)

Jedna od temeljnih značajki Auth0 je jedinstvena prijava (SSO). Ova tehnologija omogućuje korisnicima da se prijave na više aplikacija ili web stranica koristeći isti skup vjerodajnica za prijavu. Ne samo da je ovo praktičnije za korisnike jer eliminira potrebu za unosom više lozinki, već također povećava sigurnost centraliziranjem procesa zaštite.

SSO je posebno koristan za tvrtke koje imaju više web-mjesta i aplikacija, bilo internih, usmjerenih prema klijentima ili kombinacije oba. Auth0 omogućuje zaposlenicima i klijentima pristup cijelom paketu aplikacija i stranicama proizvoda tvrtke s jednom prijavom. Omogućuje zaposlenicima i partnerima pristup raznim internim web portalima bez potrebe za višestrukim korisničkim imenima i lozinkama. Ukratko, pojednostavljuje proces za sve.

2. Više opcija prijave

Postoje tri načina: prijava putem društvenih mreža, autentikacija bez lozinke i veze bez lozinke.

Prijava putem društvenih mreža je SSO opcija koja omogućuje korisnicima da se prijave koristeći svoje postojeće vjerodajnice s drugog računa, kao što su Facebook, Google ili Amazon. Auth0 podržava više od 30 opcija za prijavu na društvenim mrežama, a tvrtke mogu odabrati koje će dopustiti klijentima da koriste za prijavu.

Autentikacija bez lozinke je metoda koja u potpunosti eliminira potrebu za korisničkim imenom i lozinkom. Korisnicima omogućuje prijavu pomoću otiska prsta (TouchID) ili putem prepoznavanja lica (FaceID).

Također uklanjajući potrebu za vjerodajnicama za tradicionalnu tekstualnu prijavu, metoda veze bez lozinke omogućuje korisnicima da se prijave pomoću jednokratne lozinke (eng. *one-time password*) poslane na njihov telefon putem SMS-a ili na njihovu adresu e-pošte.

3. Autentikacija s više faktora

Auth0 dodatno poboljšava sigurnost korištenjem Autentikacije s više faktora (eng. *Multi-Factor Authentication)* za provjeru identiteta korisnika prije odobravanja pristupa. MFA je dodatni sloj sigurnosti koji od korisnika zahtijeva da daju više od jedne informacije za provjeru prije prijave.

Auth0 podržava nekoliko različitih faktora za provjeru autentičnosti, uključujući:

- Jednokratne lozinke (OTP) poslane SMS-om ili e-poštom
- Jednokratne lozinke (OTP) isporučene glasovnim pozivom

- Push obavijesti poslane korisničkim uređajima
- WebAuthn prijava bez lozinke putem kriptografije s javnim ključem ("sigurnosni ključevi")
- WebAuthn prijava bez lozinke putem biometrije uređaja (prepoznavanje otiska prsta ili lica)

Slika 4.Logo tvrtke Auth0

3. Opis praktičnog rada

Cilj praktičnog rada je implementirati prethodno opisane načine provjere autentičnosti na jednostavnoj web aplikaciji sa ciljem demonstracije i usporedbe različitih načina.

3.1. Zahtjevi implementacije

Zahtjevi implementacije definiraju što treba biti obuhvaćeno u svakoj fazi razvoja kako bi se osiguralo da sve metode provjere autentičnosti budu implementirane na ispravan način i omogućile usporedbu njihovih karakteristika. Za autentikaciju temeljenu na sesiji treba implementirati da se po uspješnoj prijavi korisnika stvori sesija za njega i treba omogućiti upravljanje tim sesijama na strani poslužitelja. Za autentikaciju na temelju tokena treba implementirati da se po uspješnoj prijavi pošalje token korisniku. Za autentikaciju preko Auth0 treba implementirati mehanizam jedinstvene prijave (eng. SSO – *Single Sign On*) gdje se korisnik može prijaviti jednim identitetom na više aplikacija te mu omogućiti višefaktorsku autentikaciju. Trebalo bi i implementirati autentikaciju preko društvenih mreža da korisnik ne mora stvarati zasebno email i lozinku za svaku web stranicu ili aplikaciju već da može upotrijebiti postojeći račun na društvenim mrežama.

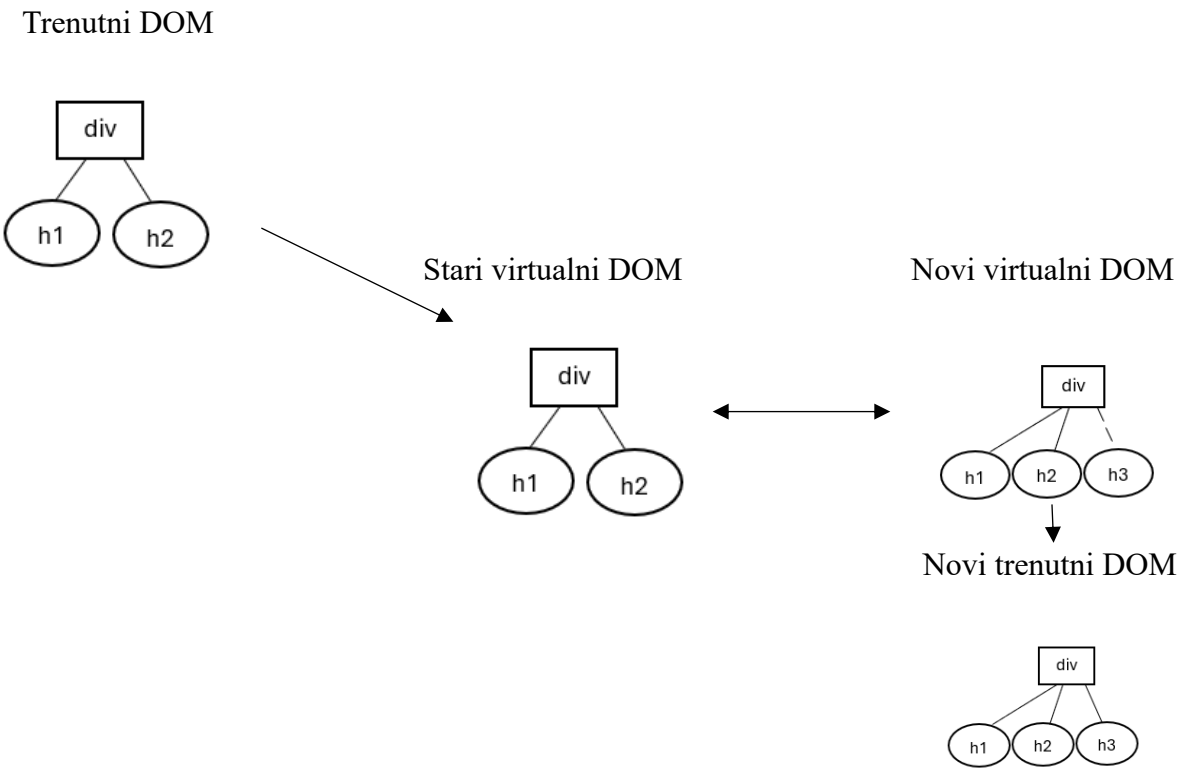
Za ispunjenje svih tih zahtjeva koristili smo neke tehnologije i alate koji će biti opisani u nastavku.

3.2. Korištene tehnologije i alati

3.2.1. React

React je JavaScript biblioteka za izradu korisničkih sučelja na webu. React je deklarativna biblioteka temeljena na komponentama koja programerima omogućuje izradu komponenti korisničkog sučelja za višekratnu upotrebu i slijedi pristup virtualnog DOM-a (eng. *Document Object Model*), koji optimizira izvedbu renderiranja minimiziranjem DOM ažuriranja. React je brz i dobro radi s drugim alatima i bibliotekama. React su izmislili programeri Facebooka koji su tradicionalni DOM smatrali sporim. Implementacijom virtualnog DOM-a React je riješio ovaj problem i brzo stekao popularnost.

React funkcionira tako da se stvara virtualni DOM u memoriji umjesto da izravno manipulira DOM-om preglednika. Izvodi potrebne manipulacije unutar ove virtualne reprezentacije prije primjene promjena na stvarni DOM preglednika. React je učinkovit, mijenja samo ono što zahtijeva modifikaciju. [8]



Slika 5.Princip rada Reacta

3.2.1.1. Značajke Reacta

React je jedna od najzahtjevnijih JavaScript biblioteka jer je opremljena gomilom značajki koje je čine bržom i spremnom za proizvodnju.

Neke od njih su:

1. Arhitektura temeljena na komponentama: React pruža značajku za rastavljanje korisničkog sučelja na manje, samostalne komponente. Svaka komponenta može imati vlastito stanje i rekvizite (eng. *props*).
2. JSX (JavaScript sintaksna ekstenzija): JSX je proširenje sintakse za JavaScript koje programerima omogućuje pisanje koda sličnog HTML-u (eng. *HyperText Markup Language*) unutar svojih JavaScript datoteka. Čini React komponente čitljivijim i izražajnijim.
3. Virtualni DOM: React održava lagani prikaz stvarnog DOM-a u memoriji. Kada dođe do promjena, React učinkovito ažurira samo potrebne dijelove DOM-a.
4. Jednosmjerno vezanje podataka(eng. *One-way data binding*): Sam naziv govori da se radi o jednosmjernom protoku. Podaci u Reactu teku samo u jednom smjeru, tj. podaci se prenose odozgo prema dolje, tj. od nadređenih komponenti do podređenih komponenti. Svojstva (props) u podređenoj komponenti ne mogu vratiti podatke svojoj nadređenoj komponenti, ali mogu komunicirati s nadređenim komponentama za izmjenu stanja u skladu s danim ulazima.
5. Performanse: React koristi virtualni DOM i ažurira samo izmijenjene dijelove. Dakle, ovo ubrzava rad DOM-a. DOM se izvršava u memoriji tako da možemo stvoriti zasebne komponente što čini DOM bržim.
6. Komponente: React dijeli web stranicu na više komponenti jer se temelji na komponentama. Svaka komponenta je dio dizajna korisničkog sučelja koji ima svoju logiku i dizajn. Dakle, logika komponente koja je napisana u JavaScriptu čini je lakšom i bržom te se može ponovno koristiti.
7. Jednostrane aplikacije (eng. *Single-Page Applications*): React se preporučuje u stvaranju SPA-ova, omogućujući glatko ažuriranje sadržaja bez ponovnog učitavanja stranice. Njegov fokus na komponente za višekratnu upotrebu čini ga idealnim za aplikacije u stvarnom vremenu.

3.2.1.2. Životni ciklus React komponenti

Svaka React komponenta ima vlastiti životni ciklus, životni ciklus komponente može se definirati kao niz metoda koje se pozivaju u različitim fazama postojanja komponente. React automatski poziva te metode u različitim točkama životnog ciklusa komponente. Razumijevanje ovih faza pomaže u upravljanju stanjem i učinkovitom optimiziranju komponenti.

Tri su faze životnog ciklusa komponenti:

1. Komponenta se dodaje (eng. *mount*) na zaslon (prvo renderiranje)
2. Komponenta se osvježava (eng. *update*) kad primi novi props ili se promijeni njeno stanje
3. Komponenta se uklanja (eng. *unmount*) sa ekrana kada je više nije potrebno prikazivati.

Dodavanje kompente (eng. *mounting*) je prva faza metode životnog ciklusa u Reactu. To je proces stvaranja komponente i njezinog dodavanja u DOM. Druga faza je ažuriranje komponente koja se u metodama životnog ciklusa Reacta primjenjuje kada se komponenta ažurira. Ažurira se kad god dođe do promjene stanja ili propsa komponente. Posljednja faza je uklanjanje komponente gdje se komponenta uklanja iz DOM-a.

3.2.2. Javascript

JavaScript je lagani interpretirani programski jezik s rudimentarnim objektno orijentiranim mogućnostima. Jezgra opće namjene jezika ugrađena je u Netscape Navigator i druge web preglednike i uljepšana za web programiranje dodatkom objekata koji predstavljaju prozor web preglednika i njegov sadržaj. Ova „klijentska“ verzija JavaScripta omogućuje uključivanje „izvršivog sadržaja“ u web stranice. To znači da web stranica više ne mora biti statički HTML, već može uključivati dinamičke programe koji komuniciraju s korisnikom, kontroliraju preglednik i dinamički stvaraju HTML sadržaj.

Sintaktički, jezgra JavaScript jezika nalikuje C, C++ i Javi, s programskim konstrukcijama kao što su naredba if, petlja while i operator &&. Međutim, sličnost završava ovom sintaktičkom sličnošću. JavaScript je netipiziran jezik, što znači da varijable ne moraju imati naveden tip. Objekti u JavaScriptu više su poput Perlovog asocijativnog niza nego što su strukture u C-u ili objekti u C++-u ili Javi. Također, kao što je spomenuto, JavaScript je čisto interpretirani jezik, za razliku od C i C++, koji se prevode, i za razliku od Jave, koja se prevodi u bajt-kod prije interpretacije. [9]

JavaScript je ono što vam omogućuje interakciju s velikom većinom web stranica koje posjećujete. Bilo da se radi o ispunjavanju obrazaca, listanju kartama ili registraciji za događaj, najvjerojatnije je programiranje u JavaScriptu ono što vam to omogućuje. JavaScript je jedan od najkorištenijih programskih jezika u svijetu. Bez toga, gledali bismo stranice koje ne rade ništa osim prikaza slika i teksta. HTML je možda okosnica web-stranice, a CSS (eng. *Cascading Style Sheets*) dodaje stil, ali JavaScript programiranje je ono što joj daje život. Bez ovog nevjerojatnog jezika, tvrtke bi bile na gubitku. Tvrtke ovise o JavaScript jeziku za interakciju sa svojim klijentima u današnjem online svijetu. Web preglednici napravljeni su da razumiju HTML i CSS i pretvaraju te jezike u vizualni prikaz na ekranu. Da bi to učinio, preglednik koristi ono što se naziva mehanizam za izgled ili renderiranje. Ovo je dio web preglednika koji razumije HTML i CSS. Preglednik također sadrži ono što se zove JavaScript tumač. To je dio preglednika koji razumije JavaScript i može izvršiti korake JavaScript programa.

JavaScript postoji već gotovo tri desetljeća, a njegova svestrana priroda, mogućnost servisiranja frontend i backend razvoja, učinila ga je glavnim osloncem u alatima većine programera. 69.8 % ljudi kaže da je JavaScript njihov preferirani jezik za kodiranje. Neki od razloga zašto je jezik tako popularan su da je on jedan od rijetkih programskih jezika koji se može koristiti u svim popularnim preglednicima, može se koristiti za izradu aplikacija na mobilnim uređajima kao i na webu i da postoji mnogo okvira i biblioteka spremnih za rad koji mogu pokrenuti projekte, štedeći puno troškova, vremena i truda razvojnim timovima. [10]

Slika 6.Logo JavaScripta

3.2.3. Node.js

Node.js je open-source okruženje za izvođenje JavaScripta na više platformi i biblioteka za pokretanje web aplikacija izvan preglednika klijenta. Ryan Dahl razvio ga je 2009. Programeri koriste Node.js za izradu web aplikacija na strani poslužitelja, a savršen je za aplikacije s velikim brojem podataka budući da koristi asinkroni model (model vođen događajima). Node.js je runtime okruženje koje omogućuje izvršavanje JavaScripta na strani poslužitelja. Izgrađen je

na V8 JavaScript motoru iz Chromea, koji prevodi JavaScript u učinkovit strojni kod. Node.js radi na arhitekturi baziranoj na jednoj niti vođenoj događajima, koristeći petlju događaja za rukovanje višestrukim istodobnim operacijama bez blokiranja.

Kada klijent pošalje zahtjev Node.js poslužitelju, zahtjev se dodaje u red događaja. Petlja događaja kontinuirano provjerava ovaj red i obrađuje svaki zahtjev. Ako zahtjev uključuje I/O (eng. *Input/Output*) operaciju, Node.js ga prebacuje u jezgru sustava, koja njime rukuje asinkrono. Nakon što je I/O operacija završena, jezgra (eng. *kernel*) obavještava Node.js, izvršavajući odgovarajuću funkciju povratnog poziva. Ovaj ne-blokirajući I/O i model vođen događajima omogućuje Node.js da učinkovito obrađuje mnogo istodobnih veza, što ga čini idealnim za izgradnju skalabilnih mrežnih aplikacija visokih performansi. [11]

Neke od značajki Node.js-a:

1. Asinkroni u prirodi i vođenim događajima: Poslužitelji izrađeni s Node.js-ovima nikada ne čekaju API. Bez čekanja podataka iz API-ja, izravno prelazi na sljedeći API. Dakle, svi API-ji Node.js-a potpuno su ne-blokirajući. Kako bi primio i pratio sve odgovore prethodnih API zahtjeva, slijedi mehanizam vođen događajima. Stoga možemo reći da su svi Node.js API-ji po prirodi ne-blokirajući.
2. Arhitektura bazirana na jednoj niti: S petljom događaja, arhitekturu baziranu na jednoj niti prati Node.js i za ovu arhitekturu Node.js je čini skalabilnijom. Za razliku od drugih poslužitelja, oni stvaraju ograničene niti za obradu zahtjeva, dok za mehanizam vođen događajima Node.js poslužitelji odgovaraju na ne-blokirajući ili asinkroni način i iz tog razloga Node.js postaje skalabilniji. Ako usporedimo Node.js s drugim tradicionalnim poslužiteljima poput Apache HTTP (eng. *Hypertext Transfer Protocol*) poslužitelja, onda možemo reći da Node.js obrađuje veći broj zahtjeva. Nakon programa s jednom niti slijedi Node.js i to omogućuje Node.js-u obradu ogromne količine zahtjeva.
3. Skalabilnost: U današnje vrijeme većina tvrtki zahtijeva skalabilni softver. Node.js rješava jedno od najhitnijih pitanja u razvoju softvera, a to je skalabilnost. Istodobnim zahtjevima može se upravljati vrlo učinkovito pomoću Node.js-a. Node.js koristi modul klastera za upravljanje tako što balansira opterećenja za sve aktivne CPU (eng. *Central processing unit*) jezgre. Najprivlačnija značajka Node.js-a je da može horizontalno napraviti particiju aplikacije, a taj postupak se postiže zbog upotrebe podređenih procesa. Koristeći ovu značajku, različite verzije aplikacije pružaju se različitoj ciljanoj publici, a također im omogućuje prilagodbu da udovolje željama klijenata.

4. Brzo vrijeme izvršenja koda: V8 JavaScript runtime motor koristi Node.js, a to također koristi Google Chrome. Hub osigurava omotač za JavaScript i iz tog razloga motor za vrijeme izvođenja (eng. *runtime motor*) i proces prepozicije zahtjeva unutar Node.js-a postaju brži.
5. Kompatibilnost na različitim platformama: Različite vrste sustava kao što su Windows, UNIX, LINUX, MacOS i drugi mobilni uređaji mogu koristiti Node.js. Za generiranje samodostatne izvedbe, može se upariti s bilo kojim odgovarajućim paketom.
6. Koristi JavaScript: Iz perspektive inženjera, vrlo je važan aspekt Node.js-a da ovaj okvir koristi JavaScript. Većina programera je upoznata s JavaScriptom, tako da im postaje mnogo lakše koristiti Node.js.
7. Brzo strujanje podataka: Vrijeme obrade podataka koji su poslani različitim tokovima traje dugo, dok za obradu podataka Node.js-u treba vrlo malo vremena i to vrlo velikom brzinom. Node.js štedi puno vremena jer Node.js istovremeno obrađuje i učitava datoteke. Kao rezultat toga, Node.js poboljšava ukupnu brzinu protoka podataka i videozapisa.
8. Nema međuspremnika: Podaci se nikada ne spremaju u međuspremnik u Node.js aplikaciji.

Slika 7.Logo Node.js-a

3.2.4. Express.js

Express.js je brz, fleksibilan i minimalistički web okvir za Node.js. To je zapravo alat koji pojednostavljuje izradu web aplikacija i API-ja pomoću JavaScripta na strani poslužitelja. Express je open-source koji razvija i održava zaklada Node.js. Express.js nudi robustan skup značajki koje povećavaju vašu produktivnost i pojednostavljuju vašu web aplikaciju. Olakšava organiziranje funkcionalnosti vaše aplikacije s posrednim softverom i usmjeravanjem. Dodaje korisne uslužne programe Node HTTP objektima i olakšava iscrtavanje dinamičkih HTTP

objekata. Express je prilagođen korisniku (eng. *user-friendly*) okvir koji pojednostavljuje proces razvoja Node aplikacija. Koristi JavaScript kao programski jezik i pruža učinkovit način za izradu web aplikacija i API-ja. S Expressom možete jednostavno upravljati rutama, zahtjevima i odgovorima, što čini proces stvaranja robusnih i skalabilnih aplikacija mnogo lakšim. Štoviše, to je lagan i fleksibilan okvir koji je jednostavan za naučiti i dolazi s opcijama middlewarea. [12]

Ključne značajke Express.js-a:

1. Middleware i usmjeravanje: Definiranje jasnih putova (ruta) unutar vlastite aplikacije omogućuje lakoću rukovanjem dolaznim HTTP zahtjevima (GET, POST, PUT, DELETE). Implementiranje funkcija za višekratnu upotrebu (middleware) omogućuju presretanje zahtjeva, stvaranje odgovora, provjeru autentičnosti, bilježenje i raščlanjivanje podataka.
2. Minimalistički dizajn: Express.js slijedi jednostavnu i minimalističku filozofiju dizajna. Ova jednostavnost omogućuje brzo postavljanje poslužitelja, definiranje ruta i učinkovito rukovanje HTTP zahtjevima. To je izvrstan izbor za izradu web aplikacija bez nepotrebne složenosti.
3. Fleksibilnost i prilagodba: Express.js ne nameće strogu arhitekturu aplikacije. Svoj kod možete strukturirati prema svojim željama. Bilo da gradite RESTful API ili potpunu web-aplikaciju, Express.js prilagođava se vašim potrebama.
4. Moć šablona: Uključenje mehanizama za izradu predložaka kao što su Jade ili EJS za generiranje dinamičkog HTML sadržaja poboljšava korisničko iskustvo.
5. Posluživanje statične datoteke: Bez napora poslužite statične datoteke kao što su slike, CSS i JavaScript iz određenog direktorija unutar vaše aplikacije.
6. Node.js integracija: Express.js se neprimjetno integrira s temeljnim funkcijama Node.js omogućujući iskorištavanje snage asinkronog programiranja i arhitekture vođene događajima.

Express.js omogućuje izradu širokog spektra web aplikacija poput RESTful API-ja, aplikacija u stvarnom vremenu i jednostranih aplikacija. Razvijanje robusnih API-ja koji se pridržavaju REST (eng. *Representational state transfer*) arhitektonskog stila omogućuju komunikaciju s drugim aplikacijama i frontend sučeljima, jednostrane aplikacije dohvaćaju i ažuriraju sadržaj na strani klijenta dok se aplikacije u stvarnom vremenu koriste za chat ili alat za zajedničko uređivanje.

Slika 8.Express.js

3.2.5. Mongo DB

MongoDB je NoSQL program za upravljanje bazom podataka otvorenog koda. NoSQL (ne samo SQL) koristi se kao alternativa tradicionalnim relacijskim bazama podataka. NoSQL, nije alat, već metodologija sastavljena od nekoliko komplementarnih i konkurentskih alata. Primarna prednost NoSQL baze podataka je ta što, za razliku od relacijske baze podataka, može učinkovito rukovati nestrukturiranim podacima kao što su dokumenti, e-pošta, multimedija i društveni mediji. Nerelacijske baze podataka ne koriste principe RDBMS (eng. *Relational Data Base Management System*) i ne pohranjuju podatke u tablice, shema nije fiksna i ima vrlo jednostavan model podataka. Umjesto toga, koriste identifikacijske ključeve i podaci se mogu pronaći na temelju dodijeljenih ključeva. [13]

NoSQL baze podataka vrlo su korisne za rad s velikim skupovima distribuiranih podataka. MongoDB je alat koji može upravljati informacijama o dokumentima, pohranjivati ili dohvaćati informacije. MongoDB se koristi za pohranu velikih količina podataka, pomažući organizacijama da pohrane velike količine podataka, a da i dalje rade brzo. Organizacije također koriste MongoDB za njegove ad-hoc upite, indeksiranje, balansiranje opterećenja, agregaciju, izvršavanje JavaScript-a na strani poslužitelja i druge značajke.

Jezik strukturiranih upita (eng. SQL - Structured Query Language) je standardizirani programski jezik koji se koristi za upravljanje relacijskim bazama podataka. SQL normalizira podatke kao sheme i tablice, a svaka tablica ima fiksnu strukturu. Umjesto korištenja tablica i redaka kao u relacijskim bazama podataka, kao NoSQL baza podataka, MongoDB arhitektura

sastoji se od zbirki i dokumenata. Dokumenti se sastoje od parova ključ-vrijednost - osnovne jedinice podataka MongoDB-a. Zbirke, ekvivalent SQL tablicama, sadrže skupove dokumenata. MongoDB nudi podršku za mnoge programske jezike, kao što su C, C++, C#, Go, Java, Python, Ruby i Swift. [14]

Značajke MongoDB-a:

- Replikacija: Skup replika su dvije ili više MongoDB instanci koje se koriste za pružanje visoke dostupnosti. Skupovi replika sastoje se od primarnih i sekundarnih poslužitelja. Primarni MongoDB poslužitelj obavlja sve operacije čitanja i pisanja, dok sekundarna replika čuva kopiju podataka. Ako primarna replika ne uspije, onda se koristi sekundarna replika.
- Skalabilnost: MongoDB podržava vertikalno i horizontalno skaliranje. Vertikalno skaliranje funkcionira dodavanjem više snage postojećem stroju, dok horizontalno skaliranje funkcionira dodavanjem više strojeva resursima korisnika.
- Balansiranje opterećenja: MongoDB upravlja balansiranjem opterećenja bez potrebe za zasebnim, namjenskim balansiranim opterećenjem, kroz okomito ili horizontalno skaliranje.
- Bez sheme: MongoDB je baza podataka bez shema, što znači da baza podataka može upravljati podacima bez potrebe za nacrtom.
- Dokument: Podaci u MongoDB-u pohranjuju se u dokumente s parovima ključ-vrijednost umjesto redaka i stupaca, što podatke čini fleksibilnijima u usporedbi sa SQL bazama podataka.

4. Primjeri implementacije

U skladu sa zahtjevima i korištenjem alata koji su navedeni u prethodnim poglavljima implementirana su 4 načina prijave koja će se u ovome poglavlju detaljnije opisati.

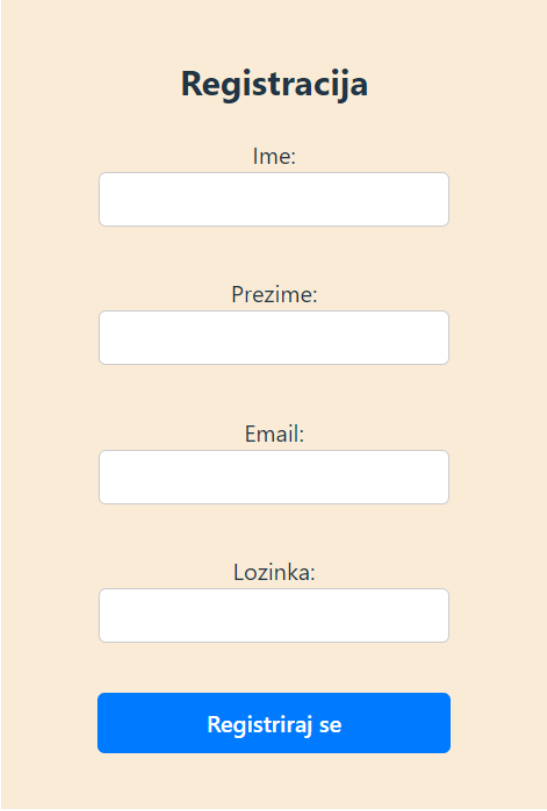
Ta 4 načina prijave su sljedeća:

1. Prijava preko sesije
2. Prijava preko JSON Web Tokena
3. Prijava preko Auth0
4. Prijava preko socijalne mreže (Googlea)

Slika 9.Naslovna stranica aplikacije

Osim te 4 prijave dodana je i registracija za korisnike kako bi se mogli prijaviti preko sesije ili preko JSON Web Tokena. Kod prijave preko Auth0 korisnik se može registrirati preko Auth0 sustava, dok god prijave preko Googlea korisnik mora imati Google račun kako bi se mogao prijaviti u sustav.

Polje „Registracija“ sadrži 4 elementa: ime, prezime, email i lozinku.



Slika 10.Registracija

Ime, prezime i lozinka ne smiju biti prazni, a email pored toga što ni on ne smije biti prazan mora imati i znak „@“ inače se korisnik neće moći registrirati. Nakon što korisnik ispuni sva 4 polja klikom na „Registriraj se“ obavit će registraciju te će moći se prijaviti preko sesije i preko JSON Web Tokena.

4.1. Prijava preko sesije

Ovdje je Express.js funkcija koja obrađuje POST zahtjeve na ruti „/registracija“. Cilj je registrirati novog korisnika i spremiti ga u bazu podataka:

```
app.post("/registracija", async (req, res) => {
  try {
    const hashLozinka = await bcrypt.hash(req.body.password, 10);
    const noviKorisnik = new Korisnik({ ...req.body, password: hashLozinka });
    await noviKorisnik.save();
    res.status(201).send("Korisnik uspješno registriran");
  } catch (error) {
    res.status(500).send(error.message);
  }
});
```

Slika 11.Express.js funkcija za registraciju

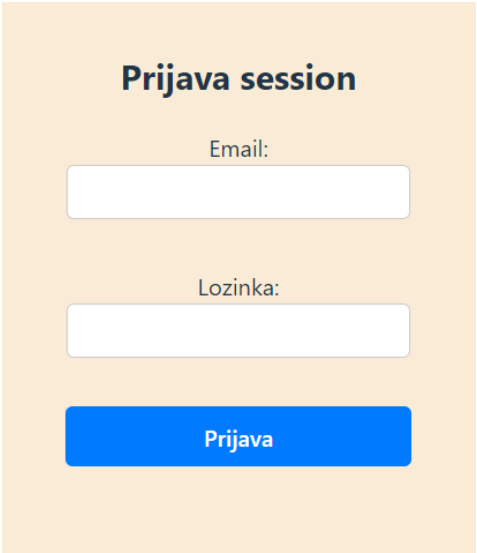
Za prijavu preko sesije koristili smo modul Node.js-a „express-session“. Pogledajmo kod:

```
app.post('/prijavasession', async (req, res) => {
  try {
    const korisnikBaza = await Korisnik.findOne({ email: req.body.email });
    if (korisnikBaza && await bcrypt.compare(req.body.password, korisnikBaza.password)) {
      req.session.korisnik = korisnikBaza;
      res.send('Prijava uspješna');
    } else {
      res.status(401).send('Neispravni podaci za prijavu');
    }
  } catch (error) {
    res.status(500).send(error.message);
  }
});
```

Slika 12.Express.js funkcija za prijavu putem sesije

Ključna stvar u ovome kodu je linija: *req.session.korisnik = korisnikBaza;* Ovdje se koristi „express-session“ middleware koji omogućava pohranu korisnikovih podataka u sesiju. Na taj način se svi budući zahtjevi od tog korisnika mogu prepoznati na temelju sesije.

Na frontend dijelu odabirom polja „Prijava session“ korisnika će se odvesti na stranicu za prijavu putem sesije:

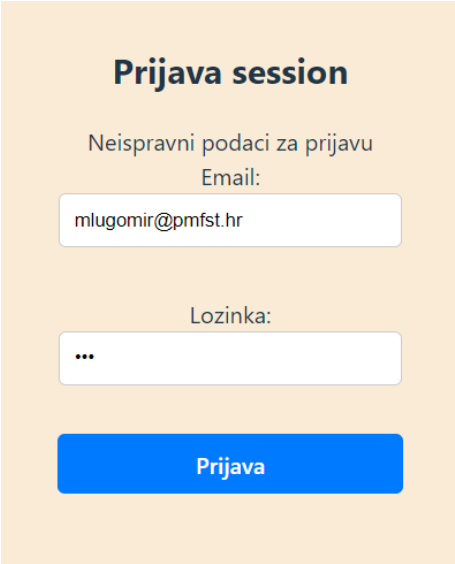


Slika 13.Prijava putem sesije

Kao što se može vidjeti, od korisnika se traži da unese email i lozinku. Ako su email i lozinka točni klikom na „Prijava“ korisnik će se uspješno prijaviti preko sesije i doći na stranicu koja ga informira da je na nju došao prijavom putem sesije.

Slika 14.Izgled stranice nakon uspješne prijave preko sesije

U slučaju da korisnik ne unese točan email ili lozinku, ispisat će mu se poruka: „Neispravni podaci za prijavu“.



Slika 15.Izgled stranice nakon neuspješne prijave (sesija)

4.2. Prijava preko JSON Web Tokena

Za prijavu preko JSON Web Tokena koristili smo JSON Web Token. Kada korisnik pošalje zahtjev za prijavu s emailom i lozinkom kod će provjeriti njegove vjerodajnice i, ako su točne, generira JWT koji vraća korisniku. JWT se zatim koristi za autentikaciju korisnika u budućim zahtjevima. Pogledajmo kod:

```
app.post('/prijavajwt', async (req, res) => {
  try {
    const korisnikBaza = await Korisnik.findOne({ email: req.body.email });
    if (korisnikBaza && await bcrypt.compare(req.body.password, korisnikBaza.password)) {
      const token = jwt.sign({ idKorisnika: korisnikBaza.email }, "tajniKljuc", { expiresIn: "1h" });
      res.json({ token });
    } else {
      res.status(401).send("Neispravni podaci za prijavu");
    }
  } catch (error) {
    res.status(500).send(error.message);
  }
});
```

Slika 16.Express.js funkcija za prijavu putem JWT-a

Ako su korisnički podaci točni, generira se JWT pomoću biblioteke „jsonwebtoken“. Metoda *jwt.sign()* kreira token koji sadrži korisnički mail kao korisnički podatak. *„tajniKljuc“* je tajni ključ koji se koristi za potpisivanje tokena. Ovaj ključ treba biti zaštićen i ne smije se dijeliti. Trajanje tokena se postavlja na jedan sat (*expiresIn: „1h“*). Nakon isteka tog vremena, token postaje nevažeći i korisnik će morati ponovno izvršiti prijavu. Ako je prijava uspješna, vraća se token kao odgovor u JSON formatu.

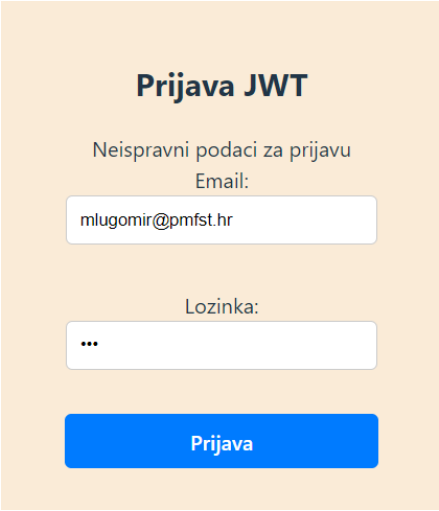
Frontend dio je skoro pa isti kao i za prijavu putem sesije, jedina je razlika što se radi o prijavi putem JSON Web Tokena pa je u naslovu te stranice „Prijava JWT“. Na tu se stranicu dođe odabirom polja „Prijava JWT“:

Slika 17.Prijava putem JWT-a

Kao i kod prijave putem sesije, i ovdje se traži unos emaila i lozinke. Ako su email i lozinka točni klikom na „Prijava“ korisnik će se uspješno prijaviti preko JSON Web Tokena i doći na stranicu koja ga informira da je na nju došao prijavom putem JSON Web Tokena.

Slika 18.Izgled stranice nakon uspješne prijave preko JWT-a

Kao i kod prijave putem sesije, u slučaju pogrešno unesenog maila ili lozinke će se ispisati poruka „Neispravni podaci za prijavu“.



Slika 19.Izgled stranice nakon neuspješne prijave (JWT)

4.3. Prijava preko Auth0

Za prijavu preko Auth0 koristili smo OpenID Connect (OIDC) protokol putem biblioteke „express-openid-connect“.

```
const config = {
  authRequired: false,
  auth0Logout: true,
  secret: "rtFF8rLtFRTudSKVzjAHiwvJ1uTrblQ13Vln8sf3jHWnr1lu-f4noNeIiXByDMxl",
  baseURL: "http://localhost:3000",
  clientID: "Nz2r5NUdlvzeh3E0j1akjp2NRr8mtVZP",
  issuerBaseURL: "https://dev-8lmk4u6v26bx1vkh.eu.auth0.com"
};

app.use(auth(config));
```

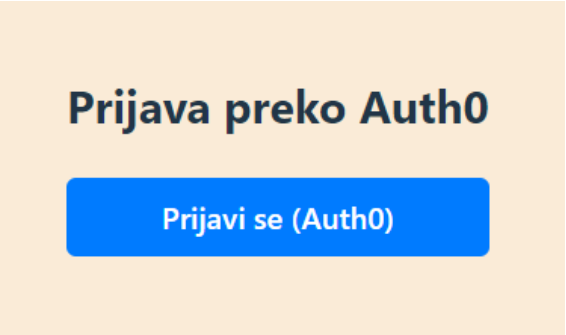
Slika 20.Konfiguracijske postavke za integraciju s Auth0

Objašnjenje pojedinih opcija u konfiguraciji:

- authRequired: false – ova postavka omogućuje da se koristi autentikacija samo za određene rute dok ostatak aplikacije može biti javno dostupan
- auth0Logout: true – ova postavka omogućuje da će korisnik biti odjavljen i s Auth0 sesije, a ne samo iz lokalne aplikacije
- secret – ovo je tajni ključ koji se koristi za potpisivanje i verifikaciju kolačića sesije
- baseURL – osnovni URL naše aplikacije, koristi se za izgradnju preusmjeravanja nakon prijave i odjave
- clientID – ID klijenta dobiven od Auth0

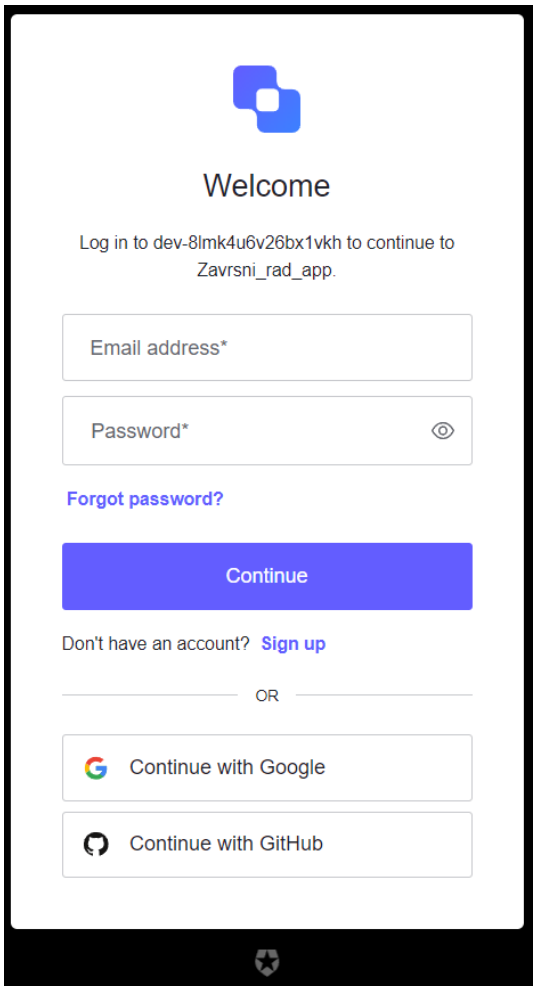
- issuerBaseUrl – URL izdavača koji se koristi za autentikaciju, lokacija gdje Auth0 poslužuje sve svoje OIDC zahtjeve, uključujući prijave i odjave.

Frontend dio je drugačiji u odnosu na prethodne načine prijave, kada se odabere polje „Prijava sa Auth0“ pokaže se nova stranica koja ima samo opciju gumba „Prijavi se (Auth0)“:



Slika 21.Izgled stranice prijave preko Auth0

Klikom na gumb otvori se nova stranica sa URL-om izdavača:



Slika 22.Prijava preko Auth0

Da bi se uspješno prijavilo potrebno je registrirati se preko opcije „Sign up“ gdje će biti potrebno unijeti email i lozinku ili se može prijaviti preko Github ili Google računa ukoliko ih korisnik posjeduje. Nakon uspješne prijave korisnika se preusmjerava na novu stranicu koja će ga informirati da je do nje došao putem Auth0 prijave. Ispod te poruke bit će i gumb za odjavu gdje će Auth0 izbrisati sesiju korisnika tako da bi neki drugi korisnik se mogao prijaviti na isto računalo.

Slika 23.Izgled stranice nakon uspješne prijave sa Auth0

4.4. Prijava preko socijalne mreže (Googlea)

Za prijavu preko Googlea koristili smo biblioteku „reactjs-social-login“.

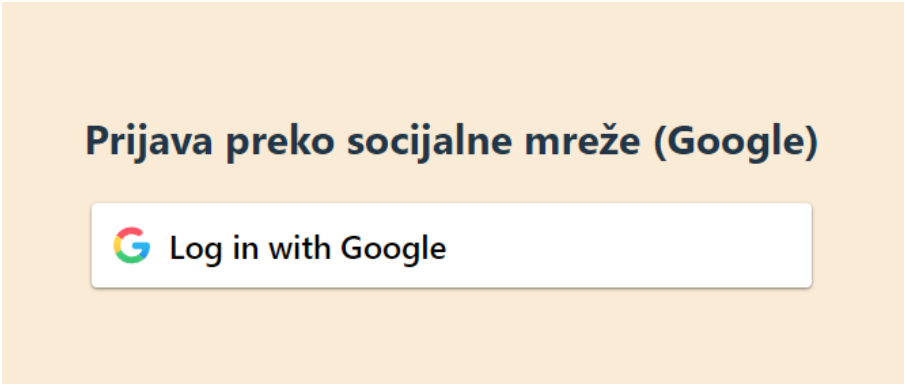
```
return (
      <div className={`App ${provider && profile ? 'hide' : ''}`}>
        <h2 className='title'>Prijava preko socijalne mreže (Google)</h2>

        <LoginSocialGoogle
          isOnlyGetToken
          client_id={"93111026364-rhlkk6mu8r41ej1fnntuqmag3uhagij2.apps.googleusercontent.com"}
          redirect_uri={"http://localhost:3000"}
          onLoginStart={onLoginStart}
          onResolve={({ provider, data }) => {
            setProvider(provider)
            setProfile(data)
            navigateTo('/racungoogle');
          }}
          onReject={(err) => {
            console.log(err)
          }}
        >
          <GoogleLoginButton />
        </LoginSocialGoogle>
      </div>
)
```

Slika 24.Implementacija prijave preko Googlea

Objašnjenje pojedinih opcija u konfiguraciji:

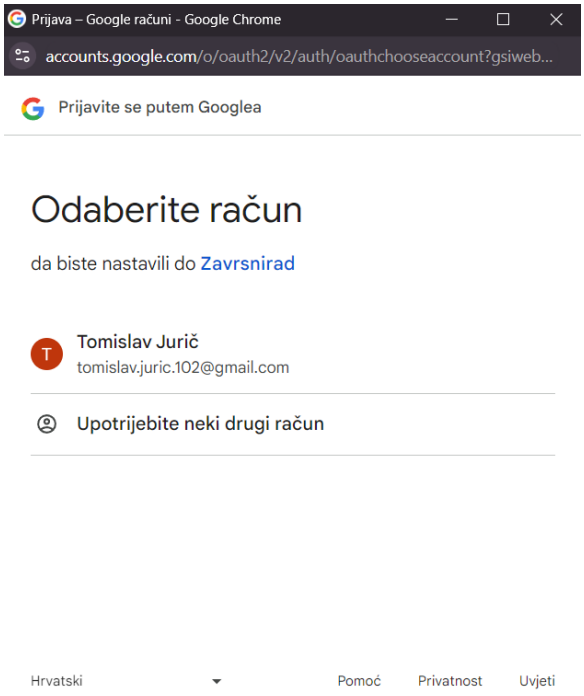
- <LoginSocialGoogle> - komponenta za prijavu putem Googlea, pružena od strane biblioteke „reactjs-social-login“. Omogućuje korisniku prijavu koristeći svoj Google račun.
- isOnlyGetToken prop – Ovaj prop je postavljen na true i to znači da se aplikacija fokusira na dobivanje Google tokena (autorizacijskog tokena) kao rezultat prijave
- client_id - ovdje se pruža Client ID generiran u Google Developer Console prilikom postavljanja Google OAuth aplikacije
- redirect_uri – parametar koji određuje gdje će se korisnik preusmjeriti nakon uspješne prijave
- onLoginStart – funkcija koja se poziva da prijava započinje
- onResolve – funkcija koja se poziva kada je prijava uspješno izvršena
- onReject – funkcija koja se poziva u slučaju da prijava nije uspješna
- <GoogleLoginButton> - komponenta iz biblioteke „react-social-login-buttons“ koja prikazuje stiliziranu tipku za prijavu putem Googlea

Nakon što se na početnoj stranici odabere polje „Prijava preko socijalne mreže Google“ otvorit će se nova stranica koja ima opciju prijave preko Google mreže.



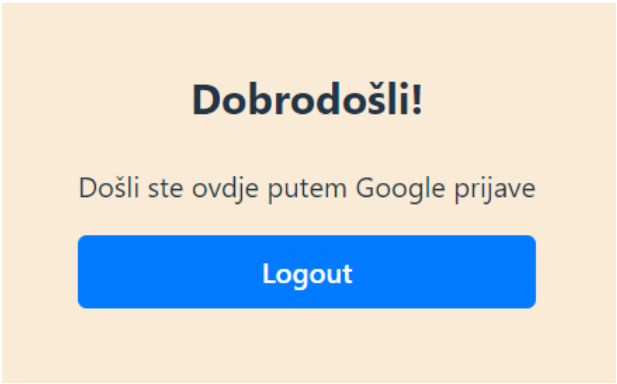
Slika 25.Izgled stranice za prijavu preko Googlea

Klikom na gumb otvara se novi prozor:



Slika 26.Prozor za prijavu preko Googlea

Nakon što se odabere račun Google obradi prijavu i dođe se do nove stranice koja informira korisnika da je do nje došao putem Google prijave. I ovdje se nalazi gumb za odjavu kako bi se neki drugi korisnik na istom računalu mogao prijaviti preko Googlea



Slika 27.Izgled stranice nakon uspješne prijave preko Googlea

Zaključak

Autentikacija i autorizacija su vrlo važni pojmovi , prisutni su gotovo svugdje na internetu. Web trgovine, društvene mreže, bankarski sustavi i mnoge druge web stranice i aplikacije koriste autentikaciju i autorizaciju, funkcioniranje takvih sustava ne bi bilo moguće bez njih. Uloga autentikacije je osigurati da sustav točno identificira korisnika i provjeri je li onaj tko tvrdi da jest. Ta sigurnosna mjera je bitna jer omogućuje zaštitu korisničkih podataka i sprječava neovlaštene pristupe. Autorizacija određuje što korisnik smije raditi unutar sustava. Bitna je zbog kontrole pristupa jer ne mogu svi korisnici imati iste ovlasti te se može ograničiti pristup osjetljivim podacima.

U ovom radu smo implementirali 4 metode autentikacije: autentikaciju temeljenu na sesiji, autentikaciju na temelju tokena i to koristeći JSON Web Token, autentikaciju sa Auth0 te autentikaciju sa socijalnim (društvenim) mrežama koristeći mrežu Google. Autentikacija temeljena na sesiji je najlakša metoda za implementaciju od svih implementiranih jer zahtijeva samo osnovnu pohranu sesije na poslužitelju i korištenje kolačića na klijentskoj strani, ali je zato i najmanje sigurna metoda od implementiranih jer je podložna mnogim napadima poput otmice sesije (eng. *session hijackling*). Autentikacija na temelju tokena (JSON Web Token) ima najsloženiju implementaciju od svih implementiranih jer treba implementirati generiranje, slanje i provjeru tokena, što je složenije od sesija te se ne koriste vanjski servisi kao u autentikaciji sa Auth0 i socijalnim mrežama koji sami nude već potrebne mehanizme. Ova metoda je sigurnija od autentikacije temeljene na sesiji, ali Auth0 i Google imaju veće sigurnosne standarde od autentikacije na temelju tokena jer ih njihovi servisi stalno održavaju. Kod metoda autentikacije sa Auth0 i Googleom smo koristili vanjske servise Auth0 i Google Cloud Console koji su olakšali implementaciju autentikacije. Obje metode imaju visoku sigurnost, dok je ipak autentikacija sa Auth0 najbolja metoda od svih implementiranih jer nudi mogućnost prijave na više načina (email i lozinka, društvene mreže, višefaktorsku autentikaciju). Ostale metode nemaju toliku mogućnost načina prijave.

Tako se može zaključiti ukoliko korisniku treba jednostavna autentikacija, a nije mu toliko bitna sigurnost onda je najbolji izbor autentikacija temeljena na sesiji, a u svim ostalim slučajevima najbolji izbor je autentikacija sa Auth0.

Literatura

[1] OneLogin, »Authentication vs. Authorization,« [Mrežno]. Available: https://www.onelogin.com/learn/authentication-vs-authorization.

[2] B. Ballad, T. Ballad i E. Banks, Access Control, Authentication, and Public Key Infrastructure, Jones & Bartlett Publishers, 2010..

[3] A. Jøsang, »A Consistent Definition of Authorization,« 13 Rujan 2017..

[4] GeeksforGeeks, »Session vs Token Based Authentication,« 4 Srpanj 2022.. [Mrežno]. Available: https://www.geeksforgeeks.org/session-vs-token-based-authentication/. [Pokušaj pristupa 7 Rujan 2024.].

[5] P. Mahindrakar i U. Pujeri, »Insights of JSON Web Token,« 6 Ožujak 2020..

[6] Descope, »What Is a JWT & How It Works,« 30 Ožujak 2024.. [Mrežno]. Available: https://www.descope.com/learn/post/jwt. [Pokušaj pristupa 7 Rujan 2024.].

[7] G. Pagel, »Auth0: Technical overview and key benefits,« 26 Lipanj 2024.. [Mrežno]. Available: https://www.weareplanet.com/blog/what-is-auth0. [Pokušaj pristupa 7 Rujan 2024.].

[8] GeeksforGeeks, »React Introduction,« 11 Ožujak 2024.. [Mrežno]. Available: https://www.geeksforgeeks.org/reactjs-introduction/. [Pokušaj pristupa 7 Rujan 2024.].

[9] D. Flanagan, JavaScript: The Definitive Guide, O'Reilly Media, 2011..

[10] B. O'Grady, »JavaScript Demystified: Why You Should Learn This Essential Language,« [Mrežno]. Available: https://codeinstitute.net/global/blog/what-is-javascript-and-why-should-i-learn-it/.

[11] T. Sufiyan, »What Is Node.js? A Complete Guide for Developers,« 9 Srpanj 2024.. [Mrežno]. Available: https://www.simplilearn.com/tutorials/nodejs-tutorial/what-is-nodejs#what_is_nodejs. [Pokušaj pristupa 7 Rujan 2024].

[12] GeeksforGeeks, »Express.js Tutorial,« 15 Srpanj 2024.. [Mrežno]. Available: https://www.geeksforgeeks.org/express-js/. [Pokušaj pristupa 7 Rujan 2024.].

[13] C. Győrödi, R. Győrödi, A. Olah i G. Pecherle, »A Comparative Study: MongoDB vs. MySQL,« 11 Lipanj 2015..

[14] A. Gillis i B. Botelho, »MongoDB,« Ožujak 2023.. [Mrežno]. Available: https://www.techtarget.com/searchdatamanagement/definition/MongoDB. [Pokušaj pristupa 12 Rujan 2024.].

Popis slika