Upravljanje pristupom i identitetom u poslužiteljskim i klijentskim applikacijama

Turjak, Tea

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Sveučilišni diplomski studij računarstva

UPRAVLJANJE PRISTUPOM I IDENTITETOM U POSLUŽITELJSKIM I KLIJENTSKIM APLIKACIJAMA

Diplomski rad

Tea Turjak

Osijek, 2024.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Obrazac D1: Obrazac za ocjenu diplomskog rada na sveučilišnom diplomskom studiju

Ocjena diplomskog rada na sveučilišnom diplomskom studiju

Ime i prezime pristupnika:	Tea Turjak
Studij, smjer:	Sveučilišni diplomski studij Računarstvo
Mat. br. pristupnika, god.	D-1254R, 07.10.2021.
JMBAG:	0165079490
Mentor:	izv. prof. dr. sc. Zdravko Krpić
Sumentor:	
Sumentor iz tvrtke:	Nemanja Plavšić
Predsjednik Povjerenstva:	prof. dr. sc. Krešimir Nenadić
Član Povjerenstva 1:	izv. prof. dr. sc. Zdravko Krpić
Član Povjerenstva 2:	izv. prof. dr. sc. Mirko Köhler
Naslov diplomskog rada:	Upravljanje pristupom i identitetom u poslužiteljskim i klijentskim applikacijama
Znanstvena grana diplomskog rada:	**Informacijski sustavi (zn. polje računarstvo)**
Zadatak diplomskog rada:	U teorijskom dijelu diplomskog rada potrebno je razraditi pojam upravljanja pristupom i identitetom te načine ostvarivanja sigurnosti na web (frontend) i mobilnim aplikacijama. Osim toga, kategorizirati i napisati pregled postojećih tehnoloških rješenja i tehnika za navedenu svrhu. Na temelju teorijskog dijela te karakteristikama postojećih programskih rješenja potrebno je definirati funkcionalne zahtjeve na poslužiteljsku applikaciju kao i na klijentske applikacije za primjer upravljanja članova nogometnog kluba. Potrebno je definirati uloge članova kluba i
Datum ocjene pismenog dijela diplomskog rada od strane mentora:	23.09.2024.
Ocjena pismenog dijela diplomskog rada od strane mentora:	Izvrstan (5)
Datum obrane diplomskog rada:	1.10.2024.
Ocjena usmenog dijela diplomskog rada (obrane):	Izvrstan (5)
Ukupna ocjena diplomskog rada:	Izvrstan (5)
Datum potvrde mentora o predaji konačne verzije diplomskog rada čime je pristupnik završio sveučilišni diplomski studij:	04.10.2024.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

IZJAVA O IZVORNOSTI RADA

Osijek, 04.10.2024.

Ime i prezime Pristupnika:	Tea Turjak
Studij:	Sveučilišni diplomski studij Računarstvo
Mat. br. Pristupnika, godina upisa:	D-1254R, 07.10.2021.
Turnitin podudaranje [%]:	12

Ovom izjavom izjavljujem da je rad pod nazivom**: Upravljanje pristupom i identitetom u poslužiteljskim i klijentskim applikacijama**

izrađen pod vodstvom mentora izv. prof. dr. sc. Zdravko Krpić

i sumentora

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis pristupnika:

SADRŽAJ

1. UVOD 1

2. UPRAVLJANJE PRISTUPOM I IDENTITETOM 3

2.1. Postojeća tehnološka rješenja i tehnike za upravljanje pristupom i identitetom 5

3. ZAHTJEVI NA SOFTVER 7

3.1. Funkcionalni zahtjevi na programsko rješenje 7

3.2. Nefunkcionalni zahtjevi na softver 11

4. PROGRAMSKO RJEŠENJE 13

4.1. Korištene tehnologije, alati i programski jezici 13

4.1.1. Java 13

4.1.2. Spring Boot 14

4.1.3. Typescript 14

4.1.4. Angular 14

4.1.5. Android Studio 14

4.1.6. Keycloak 14

4.1.7. Docker 15

4.1.8. Git 15

4.1.9. Cypress 15

4.1.10. Appium 15

4.2. Pozadinski dio ostvarenog programskog rješenja 15

4.3. Korisnički dio ostvarenog programskog rješenja u web aplikaciji 25

4.4. Korisnički dio ostvarenog programskog rješenja u mobilnoj aplikaciji 29

5. VALIDACIJA PROGRAMSKOG RJEŠENJA 32

5.1. Validacija web aplikacije 32

5.2. Validacija mobilne aplikacije 34

6. ZAKLJUČAK 36

LITERATURA 37

ŽIVOTOPIS 42

1. UVOD

Upravljanje pristupom i identitetom (engl. *Identity and Access Management*, IAM) u poslužiteljskim i klijentskim aplikacijama predstavlja problem s kojim se mnogi programeri susreću tijekom svoje karijere. Svrha upravljanja pristupom i identitetom u aplikacijama osigurati je pravilnu dodjelu identiteta korisnicima te kontroliranje njihovog pristupa podacima i funkcionalnostima aplikacije koju koriste. Takav pristup provjeri i dodjeli dopuštenja korisnicima osigurava zaštitu korisničkih i poslovnih podataka te sprječava moguću zloupotrebu aplikacije bilo da je zloupotreba uzrokovana unutarnjim ili vanjskim faktorima.

U današnjem svijetu, s ubrzanim porastom razvoja interneta i razvitkom aplikacija, povećava se potreba za boljim i kvalitetnijim osiguravanjem privatnosti podataka i upravljanja pristupom. Sve aplikacije koje su povezane s pohranom podataka ili omogućavaju pristup podacima imaju barem neki oblik identifikacije kako bi se potvrdio korisnikov identitet. Bilo da se govori o najjednostavnijem načinu identifikacije putem korisničkog imena i lozinke ili upotrebom tokena, digitalnih certifikata, dvostruke provjere identiteta te biometrije, identifikacija je postala nezamjenjiv korak pri korištenju aplikacija i smatra se normom za kreiranje pouzdanih i efikasnih aplikacija.

Identifikacija također služi za zaštitu podataka i korisničkih informacija u slučaju internetskih napada. Većina napada usmjerena na tvrtke i ustanove događa se s namjerom krađe podataka. Najčešća motivacija napadača je moguća preprodaja povjerljivih podataka, ucjena te prijevara i optužba (sabotaža) konkurentske tvrtke na poslovnom tržištu. Također se napadaju pojedinačni korisnici aplikacija pri čemu dolazi do zloupotrebe korisničkih podataka među koje najčešće ubrajamo krađu osobnih, bankovnih i zdravstvenih podataka. Napadači koriste nezakonito prikupljene podatke kako bi se mogli lažno predstavljati na internetu i dobiti pristup financijskim sredstvima koje žrtva posjeduje.

S obzirom na porast internetskih napada u posljednjih nekoliko godina, programeri neprestano unapređuju sigurnost postojećih aplikacija i aplikacija u razvoju. Sustavno poboljšavanje provjere identiteta i uloge te njihovih protokola i procesa te korištenje novih tehnologija i alata za osiguravanje i zaštitu aplikacije uvelike smanjuje vjerojatnost uspjeha mogućih napada na samu aplikaciju. Nove tehnologije, protokoli i pristupi korišteni pri utvrđivanju pouzdanosti i zaštićenosti aplikacije predstavljaju problem napadačima kojima je potrebno dulje vrijeme za

pronalazak načina na koji mogu zavarati i narušiti normalan rad aplikacije i sigurnost njezinih podataka.

U ovome radu proučavat će se teorija upravljanja pristupom i identitetom te će biti osmišljeno i realizirano programsko rješenje koje predstavlja aplikaciju za upravljanje nogometnim klubom i njegovim korisnicima. Posebna pažnja bit će posvećena provedbi implementacije autentifikacije i autorizacije u programskom rješenju upotrebom Keycloaka, alata za upravljanje pristupom i identitetom. Problemi koji će biti rješavani ostvarenim programskim rješenjem obuhvaćat će odvajanje pristupa podacima prema korisnikovim ulogama te sposobSlikanosti korisnikove prijave u aplikaciju ovisno o tipu klijenta kojem pristupa.

U drugome poglavlju rada dana je teoretska podloga vezana uz upravljanje pristupom i identitetom u aplikacijama, opisani su načini ostvarivanja sigurnosti aplikacija koji se odnose na web i mobilne aplikacije te su prikazana postojeća tehnološka rješenja i tehnike koja se koriste u svrhu zaštite aplikacija. U trećem poglavlju rada obuhvaćen je prikaz zahtjeva na softver koji se definiraju prije početka rada i izrade aplikacije. Ostvareno programsko rješenje koje je implementirano na web i mobilnoj aplikaciji opisano je detaljnije u četvrtom poglavlju zajedno s kratkim opisom tehnologija korištenih za njegovu realizaciju. U petom poglavlju obrađuje se validacija dobivenog programskog rješenja i prikazani su rezultati testiranja i analiziranja funkcionalnosti i rada aplikacije.

2. UPRAVLJANJE PRISTUPOM I IDENTITETOM

Upravljanje pristupom i identitetom (engl. *Identity and Access Management*, IAM) predstavlja sve alate, tehnologije i pristupe koji se koriste prilikom dodjeljivanja korisničkih identiteta i uloga te određivanja prava pristupa svake uloge koja se nalazi pod njihovim nadzorom. Pod upravljanje pristupom i identitetom spadaju provjera identiteta (autentifikacija), provjera uloge te upravljanje i nadzor nad ulogama korisnika, njihovih korisničkih podataka te pristupom na podatke i funkcionalnosti pojedine aplikacije (autorizacija). Autentifikacija se uvijek obavlja prva nakon koje slijedi proces autorizacije. Upravljanje pristupom i identitetom je iznimno važno za ostvarivanje i održavanje sigurnosti aplikacije, prema [1]. Pri kreiranju IAM-a za organizaciju potrebno je odrediti tip aplikacije koju će IAM koristiti, koje će sve uloge biti zastupljene i koja prava pristupa će svaka uloga imati te mehanizme i protokole koji će se koristiti za provedbu provjere identiteta i uloge kao što je opisano u [2].

Osim što garantira sigurnost aplikacije, IAM omogućava bržu dodjelu uloga novim članovima sustava te poboljšava produktivnost korisnika aplikacije jer korisnici imaju pristup svim podacima koji su im potrebni za rad. Unatoč svim pogodnostima IAM-a postoje i rizici pri njegovom korištenju. Programeri moraju biti oprezni pri definiranju korisničkih uloga i njihovog prava pristupa kako ne bi dodjeli pristup ulozi koja za to nema pravo ili izostavili pravo ulozi koje joj je potrebno za obavljanje poslova, prema [3].

Koraci rada IAM-a mogu se funkcionalno opisati kao sljedeći:

1) Prijava korisnika
    a. korisnik unosi svoje korisničke podatke u aplikaciju:
        i. korisničko ime
        ii. i korisničku lozinku.
2) Autentifikacija
    a. korisnički podaci se šalju na autentifikacijski servis (IAM poslužitelj),
    b. izvršava se provjera korisničkih podataka,
    c. ako su korisnički podaci ispravni, generira se JSON (engl. *JavaScript Object Notation*) Web Token (JWT) koji se sastoji od:
        i. Zaglavlja (engl. *header*) – specificira tip JWT tokena,
        ii. Tijela (engl. *payload*) – sadrži podatke o korisniku koji se trebaju pružiti,

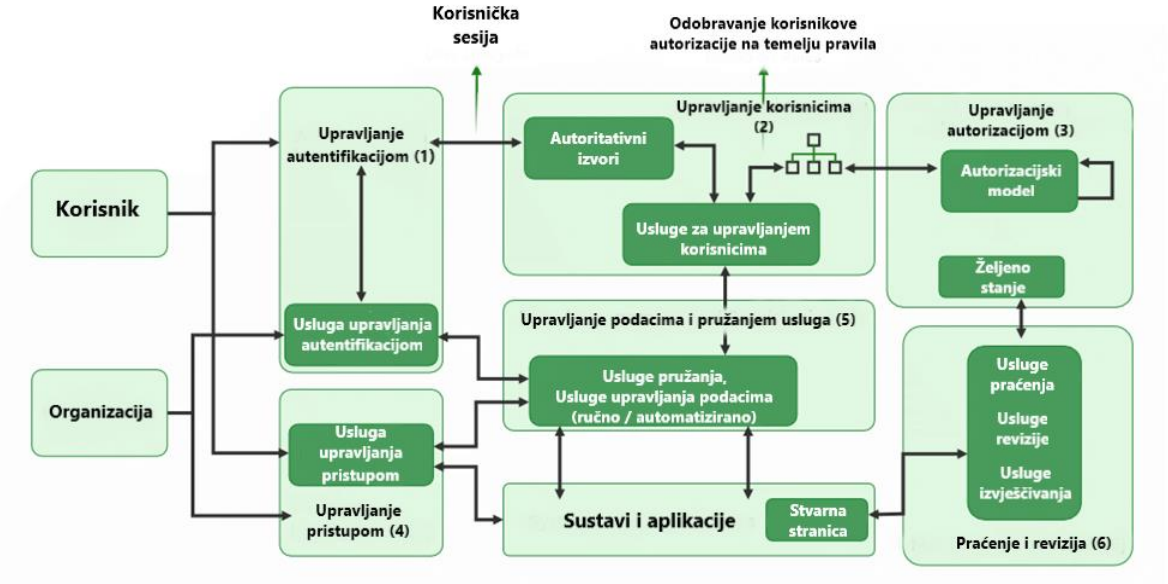
iii. Potpisa (engl. *signature*) – sadrži tajni ključ te osigurava kredibilitet sadržaja JWT tokena kako bi se spriječila njegova izmjena

d. kako bi se spriječila zlouporaba, token ima određeni vremenski period u kojem je ispravan, a u slučaju isteka valjanog perioda ponavlja se autentifikacija korisnika.

3) JWT token
   a. aplikacija sprema token dok god je korisnik prijavljen,
   b. aplikacija šalje token u svim budućih zahtjevima koji zahtijevaju korisnikovu autentifikaciju.

4) Autorizacija
   a. korisnik pošalje zahtjev koji je vezan uz pristup resursima koji su zaštićeni IAM poslužiteljima,
   b. korisnikovo pravo pristupa tim podacima se provjerava na temelju poslanog JWT tokena,
   c. u odnosu na stanje tokena, korisniku je pristup odobren ili uskraćen što je opisano u [4].

Opisani proces omogućava IAM servisima nesmetani rad u osiguravanju sigurnosti aplikacija. Primjer rada i principa IAM-a prikazan je na slici 2.1., prema [5].



Slika 2.1. Prikaz rada IAM-a.

2.1. Postojeća tehnološka rješenja i tehnike za upravljanje pristupom i identitetom

Prvom verzijom identifikacije korisnika smatra se identifikacija korištenjem korisničkog imena i lozinke. Budući da se s vremenom taj princip identifikacije pokazao nesigurnim pojavila se potreba koja je dovela do razvitka IAM-a i njegovih protokola te pristupa osiguravanja aplikacija i njihovih korisnika. Neki od najpoznatijih protokola i standarda koje IAM koristi za provjeru identiteta i uloge su:

a) LDAP (engl. *Lightweight Directory Access Protocol*),
b) SAML (engl. *Security Assertion Markup Language*),
c) SCIM (engl. *System for Cross-domain Identity Management*),
d) OAuth 2.0 (engl. *Open Authorization 2.0*),
e) OIDC (engl. *OpenID Connect*),
f) XACML (engl. *eXtensible Access Control Markup Language*),
g) RBAS (engl. *Role-Based Access Control*),
h) IdaaS (engl. *Identity as a Service*),
i) MFA(engl. *Multi-Factor Authentification*), prema [3, 6].

Kako bi se željeni protokoli i pristupi identifikacije lakše implementirali, stvoreni su IAM poslužitelji. IAM poslužitelji koriste se za lakšu implementaciju upravljanja pristupom i identitetom u aplikacijama. Među najpoznatije IAM poslužitelje otvorenog koda ubrajaju se:

a) Keycloak,
b) OpenIAM,
c) FreeIPA,
d) Gluu,
e) Okta,
f) Shibboleth pokazano u [7].

Primjeri aplikacija koje koriste IAM poslužitelje za upravljanje identitetom i pristupom su:

- Netflix - aplikacija za online gledanje serija i filmova, koristi Amazon Web Services (AWS) kao IAM poslužitelj kako bi osigurao sigurnu distribuciju sadržaja korisnicima te zaštitu njihovih osobnih podataka što je opisano u [8],

- Discord - platforma za komunikaciju, koristi Oktau kao IAM platformu za osiguravanje privatnost razmjene poruka između korisnika, prema [9],
- Salesforce - predstavlja softver za kreiranje strategije u odnosima između tvrtki i njezinih klijenata, koristi SCIM protokol za upravljanje pristupom i identitetom kako bi se osigurao pristup podacima vezanih uz provedene poslovne transakcije i zaštitili podaci njihovih klijenata prikazano u [10].

3. ZAHTJEVI NA SOFTVER

Za ostvarenje programskog rješenja iz praktičnog dijela rada i primjenu IAM-a u aplikaciji za upravljanje članovima i resursima nogometnog kluba, neophodno je utvrditi zahtjeve na softver. Da bi se ostvarili svi kriteriji i funkcionalnosti aplikacije, potrebno je precizno definirati funkcionalne i nefunkcionalne zahtjeve.

3.1. Funkcionalni zahtjevi na programsko rješenje

Funkcionalni zahtjevi na softver opisuju zahtjeve koje programer mora implementirati kako bi se korisnicima omogućilo korištenje programskog rješenja pri izvršavanju zadataka za koje je aplikacija rađena. Funkcionalni zahtjevi opisuju ponašanje aplikacije prema korisnikovim postupcima, a najčešće podrazumijevaju:

a) Autentifikaciju – potvrdu identiteta korisnika prilikom prijave putem korisničkih podataka,
b) Autorizaciju – provjeru prava pristupa podacima na temelju grupa ili uloga koje su dodijeljene korisniku,
c) Obradu podataka – kreiranje, mijenjanje i brisanje podataka u bazi aplikacije,
d) Izvršavanje radnji na temelju korisnikova unosa – u odnosu na korisnikovu radnju, aplikacija provodi zadane postupke, prema [11].

Za lakše formuliranje funkcionalnih zahtjeva razvijene su metode čija je svrha poboljšati i osigurati preglednost i jasnoću zahtjeva koji se definiraju. Neke od tih metoda su:

a) korisničke priče (engl. *User Story*, US) opisane u [12],
b) specifikacije funkcionalnih zahtjeva (engl. *Functional Requirement Specification*, FRS) navedene u [13],
c) razvoj softvera vođen ponašanjem (engl. *Behavior-Driven Development*, BDD), prema [14].

Funkcionalni zahtjevi na programsko rješenje nogometnog kluba raspisani su u obliku predloška „Dano-Kada-Tada“ (engl. *Given-When-Then*) opisanog u [15] koji se koristi zbog svoje jednostavne strukture i razumljivosti. „Dano-Kada-Tada“ predstavlja BDD predložak za korisničku priču koji se sastoji od tri dijela:

a) „Dano“ (engl. *Given*) – opisuje korisnikovu radnju ili dane podatke,
b) „Kada“ (engl. *When*) - označava trenutak u kojem korisnik izvede radnju,

c) „Tada“ (engl. *Then*) - predstavlja rezultat korisnikovih radnji.

Funkcionalni zahtjevi na web i mobilnu aplikaciju prikazani su u tablicama 3.1. i 3.2.

Tablica 3.1. Funkcionalni zahtjevi na programsko rješenje web aplikacije.

ZAHTJEV 1: USPJEŠNA PRIJAVA KORISNIKA U APLIKACIJU S KORISNIČKIM PODACIMA	
Ako je dana:	mogućnost prijave u web aplikaciju
Kada:	korisnik unese točne korisničke podatke i klikne na gumb za prijavu
Tada:	aplikacija dozvoljava korisniku pristup te je vidljiva početna stranica aplikacije
ZAHTJEV 2: NEUSPJEŠNA PRIJAVA KORISNIKA U APLIKACIJU PREMA KORISNIČKIM PODACIMA	
Ako je dana:	mogućnost prijave u web aplikaciju
Kada:	korisnik unese neispravne korisničke podatke i klikne na gumb za prijavu
Tada:	aplikacija ne dozvoljava korisniku pristup, povratak na ekran za prijavu
ZAHTJEV 3: DOHVAT PODATAKA O KORISNIKU	
Ako je dano:	da je korisnik prijavljen u web aplikaciji
Kada:	korisnik klikne na korisnikovo ime
Tada:	aplikacija daje pristup osobnim podacima korisnika
ZAHTJEV 4: DOHVAT PODATAKA KOJIMA KORISNIK IMA PRISTUP VEZANIH UZ NOGOMETNI KLUB	
Ako je dano:	da je korisnik prijavljen u aplikaciju, te ima pristup gumbovima za dohvat podataka iz baze podataka poslužiteljske aplikacije koji su generirani nakon prijave, ovisno o grupi kojoj korisnik pripada (npr. Igrač)
Kada:	korisnik stisne na gumb u odnosu na podatke kojima želi pristupiti (npr. Treninzi)
Tada:	aplikacija prikazuje podatke dohvaćene iz baze podataka poslužiteljskog dijela aplikacije (npr. Raspored treninga i njegove specifikacije)
ZAHTJEV 5: ODJAVA KORISNIKA IZ APLIKACIJE	
Ako je dano:	da je korisnik prijavljen u web aplikaciji
Kada:	korisnik stisne na gumb za odjavu iz aplikacije
Tada:	aplikacija izvršava odjavu korisnika, a korisnik je vraćen na stranicu za prijavu u web aplikaciju

Tablica 3.2. Funkcionalni zahtjevi na programsko rješenje mobilne aplikacije.

ZAHTJEV 1: USPJEŠNA PRIJAVA KORISNIKA U APLIKACIJU PREMA PRIPADNOJ GRUPI I KORISNIČKIM PODACIMA	
Ako je dana:	mogućnost prijave u mobilnu aplikaciju
Kada:	korisnik unese točne korisničke podatke i pripada grupi koja ima pristup aplikaciji mobilnog uređaja te klikne na gumb za prijavu
Tada:	aplikacija dozvoljava korisniku prijavu i pristupa glavnom ekranu aplikacije
ZAHTJEV 2: NEUSPJEŠNA PRIJAVA KORISNIKA U APLIKACIJU PREMA PRIPADNOJ GRUPI	
Ako je dana:	mogućnost prijave u mobilnu aplikaciju
Kada:	korisnik unese točne korisničke podatke, ali ne pripada grupi koja ima pristup aplikaciji mobilnog uređaja te klikne na gumb za prijavu
Tada:	aplikacija ne dozvoljava korisniku pristup, korisnik je vraćen na ekran za prijavu
ZAHTJEV 3: NEUSPJEŠNA PRIJAVA KORISNIKA U APLIKACIJU PREMA KORISNIČKIM PODACIMA	
Ako je dana:	mogućnost prijave u mobilnu aplikaciju
Kada:	korisnik unese neispravne korisničke podatke, te pripada grupi koja ima pristup aplikaciji mobilnog uređaja i klikne na gumb za prijavu
Tada:	aplikacija ne dozvoljava korisniku pristup, korisnik je vraćen na ekran za prijavu te je obaviješten da nije unio dobre korisničke podatke
ZAHTJEV 4: DOHVAT PODATAKA KOJIMA KORISNIK IMA PRISTUP VEZANIH UZ NOGOMETNI KLUB	
Ako je dano:	da je korisnik prijavljen, a ima pristup gumbovima za dohvat podataka iz baze koji su generirani nakon prijave, ovisno o grupi kojoj korisnik pripada (npr. Trener)
Kada:	korisnik stisne gumb u odnosu na podatke kojima želi pristupiti (npr. Statistika igrača prema utakmicama)
Tada:	aplikacija prelazi na novi ekran koji prikazuje podatke dohvaćene iz baze podataka poslužiteljske aplikacije (npr. Statistika svih igrača nogometnog kluba za utakmice)
ZAHTJEV 5: POVRATAK NA GLAVNU STRANICU APLIKACIJE	
Ako je dano:	da se korisnik se nalazi u stanju aplikacije koje je rezultat akcija u 4) zahtjevu
Kada:	korisnik stisne na gumb za povratak na glavni zaslon aplikacije
Tada:	aplikacija vraća korisnika na glavni zaslon aplikacije
ZAHTJEV 6: ODJAVA KORISNIKA IZ APLIKACIJE	

Ako je dano:	da je korisnik prijavljen u mobilnoj aplikaciji
Kada:	korisnik stisne gumb za odjavu iz aplikacije
Tada:	aplikacija izvršava odjavu korisnika, korisnik je vraćen na zaslon za prijavu

Kako bi se prikazao rad nogometnog kluba i ispoštovali navedeni funkcionalni zahtjevi, kreirano je sedam grupa kojima će se dodijeliti pripadajuće uloge za pristup resursima prilikom izrade programskog rješenja. Kreirane grupe bit će: igrači, treneri, doktori, financijsko osoblje, nogometno osoblje, administracijsko osoblje i vlasnici nogometnog kluba. Grupe kojima članovi nogometnog kluba će pripadati i uloge koje će naslijediti prikazane su u tablici 3.3.

Tablica 3.3. Prikaz grupa i pripadnih uloga za upravljanje članovima nogometnog kluba.

	Igrači	*Treneri*	*Doktori*	*Financijsko osoblje*	*Nogometno osoblje*	*Administracijsko osoblje*	*Vlasnici*
MATCH	X	X	X		X	X	X
CONTRACT	X			X		X	X
TRANSFER	X			X		X	X
PLAYER	X	X	X		X	X	X
STATISTICS MATCH	X	X	X			X	X
MEDICAL RECORD	X		X				
STATISTICS TRAINING	X	X	X			X	X
TRAINING	X	X	X		X	X	X
COACH		X				X	X
FINANCE STAFF				X		X	X
PURCHASE				X		X	X
FOOTBALL STAFF					X	X	X
OWNER						X	X
ADMINISTRATION STAFF						X	X
DOCTOR			X			X	X
MOBILE	X	X			X		X

Popis pojedinačnih uloga i njihovih predviđenih funkcija prikazani su u tablici 3.4.

Tablica 3.4. Svrha uloga za upravljanje nogometnim klubom.

ULOGE:	**FUNKCIJA***:*
ROLE_MATCH_ACCESS	omogućiti pristup podacima u tablici s podacima o odigranih i budućim utakmicama

ROLE_DOCTOR_ACCESS	omogućiti pristup podacima o doktorima koji su članovi nogometnog kluba
ROLE_TRAINING_ACCESS	omogućiti pristup podacima u tablici s podacima o održanim i budućim treninzima
ROLE_PLAYER_ACCESS	omogućiti pristup podacima o igračima koji su članovi nogometnog kluba
ROLE_COACH_ACCESS	omogućiti pristup podacima o trenerima koji su članovi nogometnog kluba
ROLE_CONTRACT_ACCESS	omogućiti pristup podacima o ugovorima koje su igrači sklopili s nogometnim klubom
ROLE_OWNER_ACCESS	omogućiti pristup podacima o vlasnicima nogometnog kluba
ROLE_TRANSFER_ACCESS	omogućiti pristup podacima o igračima koji prelaze ili odlaze u drugi nogometni klub
ROLE_FINANCE_STAFF_ACCESS	omogućiti pristup podacima o financijskom osoblju koji su članovi nogometnog kluba
ROLE_PURCHASE_ACCESS	omogućiti pristup popisu kupljenog materijala koji su potrebni za funkcioniranje nogometnog kluba i njegovih članova
ROLE_STATISTICS_MATCH_ACCESS	omogućiti pristup podacima o statistici igrača tijekom utakmice
ROLE_FOOTBALL_STAFF_ACCESS	omogućiti pristup podacima o nogometnom osoblju kluba
ROLE_ADMINISTRATION_STAFF_ACCESS	omogućiti pristup podacima o administracijskom osoblju kluba
ROLE_STATISTICS_TRAINING_ACCESS	omogućiti pristup podacima o statistici igrača tijekom treninga
MOBILE_ACCESS	omogućava korisnicima prijavu u aplikaciju na mobilnom uređaju

Uloge i odgovarajuće grupe za aplikaciju za upravljanje nogometnim klubom kreirat će se s pomoću Keycloak administracijske konzole prema navedenim specifikacijama tijekom stvaranja programskog rješenja. Svaka kreirana uloga nogometnog kluba dodijelit će se odgovarajućoj grupi.

3.2. Nefunkcionalni zahtjevi na softver

Nefunkcionalni zahtjevi na softver vezani su uz sposobnosti rada sustava koji pokreću programsko rješenje, a među njih najčešće ubrajamo:

a) Upotrebljivost – odnosi se na mogućnost intuitivne i efikasne interakcije klijenata s kreiranim sučeljem korisničke aplikacije ,
b) Pouzdanost – mogućnost aplikacije da funkcionira bez pogrešaka u određenom vremenskom periodu,
c) Performanse – mogućnost aplikacije da vrati ispravan željeni odziv u što kraćem vremenskom roku,
d) Sigurnost – sposobnost aplikacije da ne dopušta neovlašteni pristup aplikaciji i njezinim resursima,
e) Dostupnost – sposobnost aplikacije da funkcionira u vremenskom periodu u kojem je potrebna korisniku,
f) Skalabilnost – mogućnost aplikacije da ispravno izvršava tražene obveze unatoč povećanju broja korisnika ili dohvaćanjem veće količine podataka, prema [11].

Nefunkcionalni zahtjevi sustava na ostvareno programsko rješenje nogometnog kluba definirani su s ciljem postizanja kvalitetnog, sigurnog i pouzdanog korisničkog iskustva i rada aplikacije, a to su:

a) programsko rješenje treba koristiti OpenID Connect protokol kako bi se uspješno implementirala autentifikacija i autorizacija korisnika nogometnog kluba,
b) programsko rješenje treba izvršiti uspostavu i kreiranje klijenata te sigurnosti s pomoću Keycloak alata,
c) programsko rješenje treba koristiti Docker spremnike za pokretanje stvorenih aplikacija, za rad s bazom podataka kreiranoj na poslužiteljskoj aplikaciji i pokretanje Keycloak funkcionalnosti koje osiguravanju autentifikaciju i autorizaciju korisnika,
d) programsko rješenje treba omogućiti korisniku slanje HTTP zahtjeva putem korisničkog dijela aplikacije prema bazi podataka na poslužiteljskoj strani aplikacije,
e) programsko rješenje treba omogućiti dohvaćanja vrijednosti iz baze podataka u roku manjem od jedne sekunde,
f) programsko rješenje treba omogućiti korisniku prijavu na kreirani web ili mobilni klijent, ovisno o njegovom pravu pristupa,
g) programsko rješenje treba imati mogućnost proširenja i nadogradnje u budućnosti kako bi se moglo poboljšavati i unaprjeđivati dodatnim funkcionalnostima.

4. PROGRAMSKO RJEŠENJE

Programsko rješenje izrađeno u okviru ovoga rada kreirano je na primjeru upravljanja članovima nogometnog kluba i njegovim pripadnim resursima. Prilikom kreiranja programskog rješenja praćeni su i poštovani zadani zahtjevi na programsko rješenje. Uspješno su izvedeni zahtjevi vezani uz implementaciju autentifikacije i autorizacije korisnika u aplikaciju s pomoću Keycloaka kao IAM poslužitelja, korisnicima je omogućeno korištenje aplikacije i pristup podacima iz baze s poslužiteljskog dijela aplikacije te je ostvareno korištenje aplikacijskih resursa na više klijenata.

Prilikom izrade programskog rješenja korištene su sljedeće tehnologije, alati i programski jezici, koji su detaljnije opisani u potpoglavlju 4.1:

- Java,
- Spring Boot,
- TypeScript,
- Angular,
- Android Studio,
- Keycloak,
- Docker,
- Git,
- Cypress,
- Appium.

Za dobivena programska rješenja prikazani su provedeni postupci, mehanizmi i programski kod potreban za uspješnu implementaciju upravljanja pristupom i identitetom te komunikacija između klijentskih i poslužiteljskih aplikacija što je detaljnije opisano u potpoglavljima 4.2, 4.3 i 4.4.

4.1. Korištene tehnologije, alati i programski jezici

Za razvoj programskog rješenja korištene su tehnologije, alati i programski jezici koji su se pokazali pouzdanima, efikasnima i kvalitetnima u praksi za realizaciju sustava upravljanja pristupom i identitetom.

4.1.1. Java

Java je objektno-orijentirani programski jezik koji je značajno korišten u razvitku programskog rješenja, posebno za poslužiteljski dio aplikacije, infrastrukturu klijentski dio mobilne aplikacije.

Jedan je od najpopularnijih jezika zbog svoje prilagodljivosti i objektno-orijentirane paradigme koja ga čini izuzetno korisnim u softverskom razvoju, prema [16].

4.1.2. Spring Boot

Spring Boot je proširenje Springa, alata otvorenog koda te predstavlja okvir za razvoj aplikacija baziran na Java programskom jeziku. Spring Boot programerima olakšava razvoj te omogućuje brže pokretanje samostalnih aplikacija zbog čega je popularan za korištenje. Ima mogućnost pokretanja aplikacija bez dodatnog kreiranja programskog koda ili postavljanja XML konfiguracije što je vidljivo u [17].

4.1.3. Typescript

TypeScript je programski jezik baziran na JavaScript-u. Za razliku od JavaScripta, skalabilniji je, ima više jezičnih mogućnosti, validacije i podrške za održavanje koda. Može raditi pod istim uvjetima i platformama na kojima radi i JavaScript što je opisano u [18]. Korišten je za razvoj klijentskog dijela aplikacije u Angularu i njezino testiranje u Cypressu.

4.1.4. Angular

Angular je biblioteka otvorenog koda koja pretežito služi za kreiranje web aplikacija. Od jezika koristi HTTP i TypeScript. Omogućava uporabu AJAX-a, RESTful API-a, predložaka za izradu te drugih brojnih korisnih značajki, prema [19]. U ovome radu koristio se za izradu klijentskog dijela web aplikacije.

4.1.5. Android Studio

Android Studio integrirano je razvojno okruženje koje se koristi za razvoj Android aplikacija. Ima veliku podršku za Java i Kotlin programske jezike, navedeno u [20]. Predstavlja platformu koja je korištena za realizaciju programskog rješenja vezanog uz izradu mobilne aplikacije.

4.1.6. Keycloak

Keycloak je IAM poslužitelj otvorenog koda koji omogućuje upravljanje pristupom i identitetom. Pruža dobru provjeru identiteta i uloge te upravljanje i dodjeljivanje uloga korisnicima te omogućava samostalnu integraciju s web i mobilnim aplikacijama. Odabran je zbog svoje fleksibilnosti i pouzdanosti te ima podršku za sigurnosne standarde i protokole vezane uz provjeru identiteta i uloge korisnika što je naznačeno u [21]. Korišten je za kreiranje uloga koje se dodjeljuju korisnicima nogometnog kluba i provjeru prava pristupa korisnika nakon njihove prijave u

aplikaciju. Keycloak olakšava testiranje aplikacija jer posjeduje mogućnost lokalnog testiranja, što nije uvijek moguće kod servisa vezanih uz usluge na oblaku.

4.1.7. Docker

Docker je platforma koja programerima omogućava kreiranje i pokretanje aplikacija u izoliranim spremnicima. Napravljeni spremnici služe za pokretanje koda te uključuju sve potrebne alate i biblioteke neophodne za nesmetani rad aplikacije. Kreiranje i pokretanje spremnika može se izvršiti na svim uređajima s operacijskim sustavima koji podržavaju Docker, prema [22].

4.1.8. Git

Git je alat otvorenog koda čija je svrha rukovanje i verzioniranje programskog koda. Koristi se za praćenje promjena napravljenih prilikom pisanja koda te unaprjeđuje i olakšava timsku suradnju na projektima koji su u razvoju što je objašnjeno u [23].

4.1.9. Cypress

Cypress je alat otvorenog koda koji služi za automatizirano testiranje rada korisničkog dijela web aplikacija i stranica koje se pokreće u web pregledniku. Podržava TypeScript i JavaScript programske jezike za pisanje programskog koda za testiranje navedeno u [24].

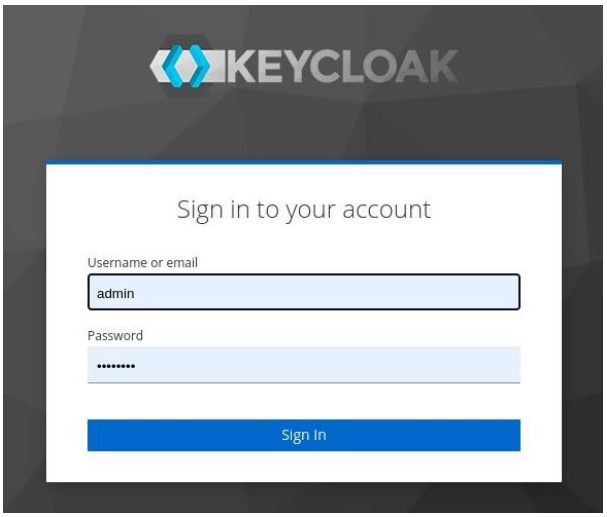
4.1.10. Appium

Appium je alat otvorenog koda koji se koristi za testiranje mobilnih aplikacija. Appium je HTTP server koji se koristi za automatizaciju testiranja. Podržava testiranje Android i iOS mobilnih uređaja te emulatora razvojnih platforma kojima se stvaraju mobilne aplikacije što je navedeno u [25]. Appium inspektor često se koristi uz Appium server radi efikasnijeg dohvaćanja jedinstvenih identifikacijskih vrijednosti elemenata u mobilnim aplikacija, vidljivo u [26] što omogućuje brže pisanje testova za testiranje aplikacije.

4.2. Pozadinski dio ostvarenog programskog rješenja

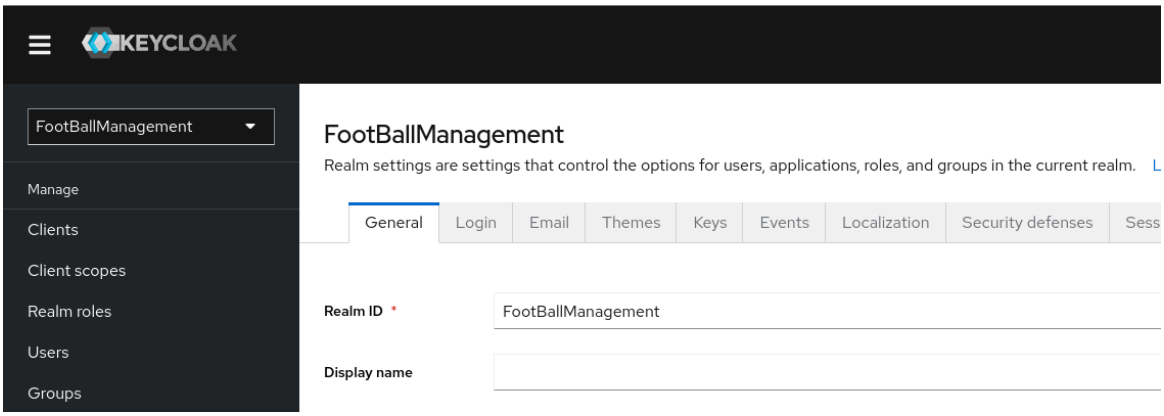
Za kreiranje pozadinske strane programskog rješenja i razvoja aplikacije, potrebno je napraviti Docker spremnike koji služe za prvobitno pokretanje Keycloak servisa i njegovih funkcionalnosti kojima se pristupa preko administracijske konzole. Koraci za pravilnu instalaciju Keycloaka objašnjeni su u [27]. Definira se port za lokalni poslužitelj (*localhost*) na kojem će se pozivati Keycloak. Konkretno, za programsko rješenje izrađeno u okviru ovog rada korišten je port 8000.

Završenom instalacijom Keycloaka putem Dockera u internetskom pregledniku pokreće se početna stranicu koja dopušta pristup administracijskoj konzoli te tehničkoj dokumentaciji Keycloaka. Klikom na gumb administracijske konzole (engl. *„Administration Console“*), programer se prijavljuje upisom korisničkih podataka definiranih prilikom instalacije Keycloak servisa. Prijava na administracijsku konzolu prikazana je na slici 4.1.



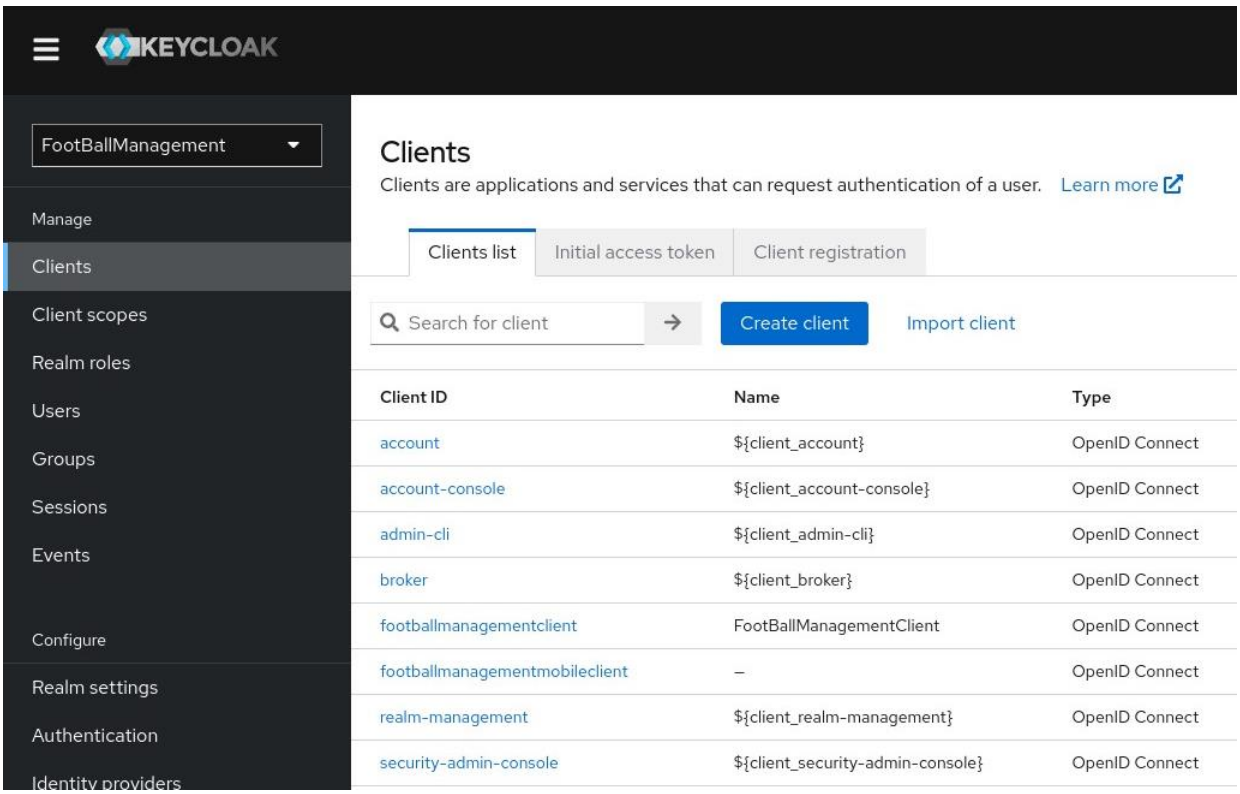
Slika 4.1. Prikaz prijave u administracijsku konzolu.

Uspješnom prijavom administracijskog korisnika usmjereni smo na administracijsku konzolu dostupnu na poveznici: *http://localhost:8000/admin/master/console/*. Administracijska konzola omogućava kreiranje vlastite Keycloak instance za podešavanje odgovarajućih parametara sustava. Instanca dozvoljava rukovanje postavkama vezanih uz autorizaciju i autentifikaciju korisnika, te stvaranje aplikacijskih klijenata, uloga i grupa za daljnji razvoj sigurnosti aplikacije. Kreirana *FootBallManagement* instanca sa željenim kriterijima prikazana je na slici 4.2.



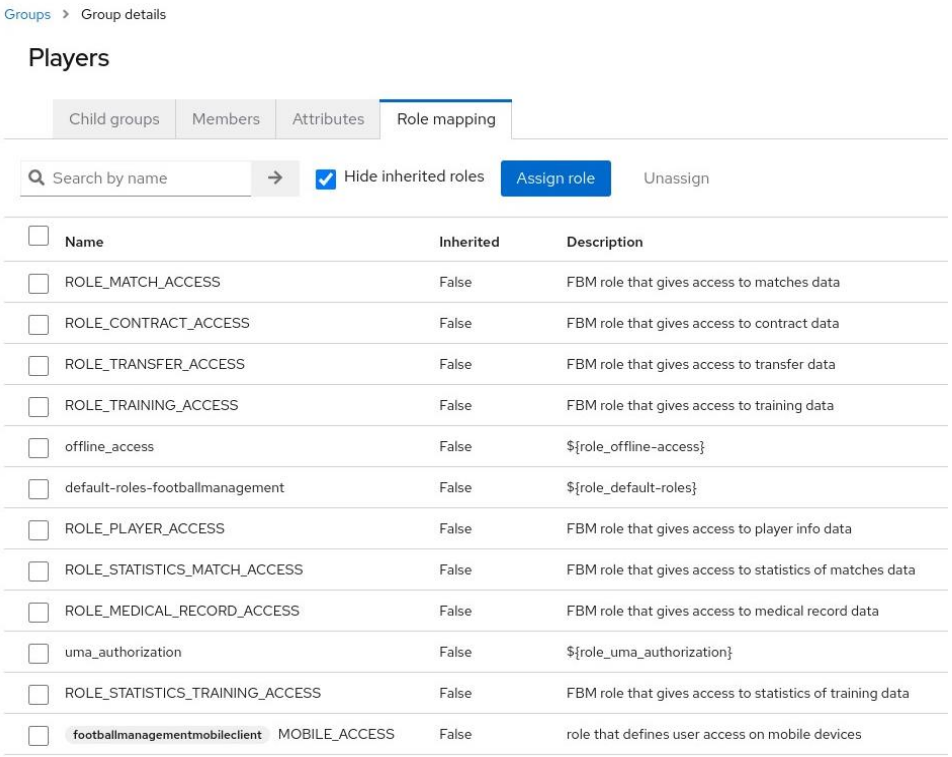
Slika 4.2. Prikaz kreirane *FootBallManagement* instance.

Na web pregledniku stvorena *FootBallManagement* instanca postaje dostupna preko vlastite poveznice *http://localhost:8000/admin/master/console/#/FootBallManagent/*. Administracijskom konzolom definiraju se klijenti potrebni za komunikaciju s poslužiteljskim i klijentskim aplikacijama, a za svakog novonastalog klijenta definiran je klijentov identifikacijski broj, njegov naziv, poveznice putem kojih klijent dozvoljava pristup zahtjevima, te uloge i grupe koje su specifične za pojedinog klijenta. Kreirani klijenti za web i mobilnu aplikaciju prikazani su slici 4.3.



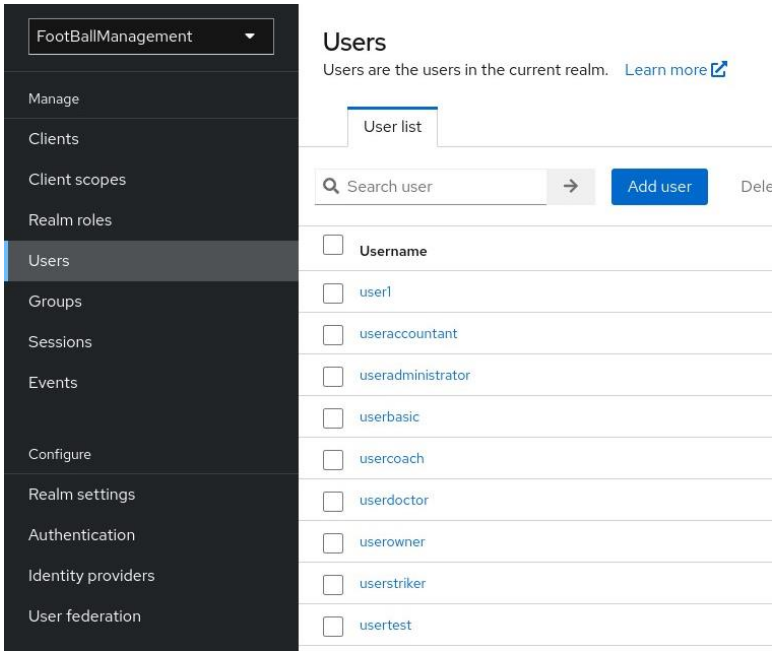
Slika 4.3. Prikaz klijenata *FootBallManagement* instance Keycloaka.

Uloge definirane na globalnoj razini *FootBallManagent* instance dostupne su za korištenje svim klijentima, dok su uloge napravljene za pojedinog klijenta dostupne samo za uporabu funkcionalnosti tog klijenta. Kreirane uloge mogu se grupirati, a te grupe dodaju se zadanim korisnicima pri čemu se korisniku dodjeljuje više uloga istovremeno. Primjer kreirane grupe i dodijeljenih uloga prikazan je na slici 4.4.



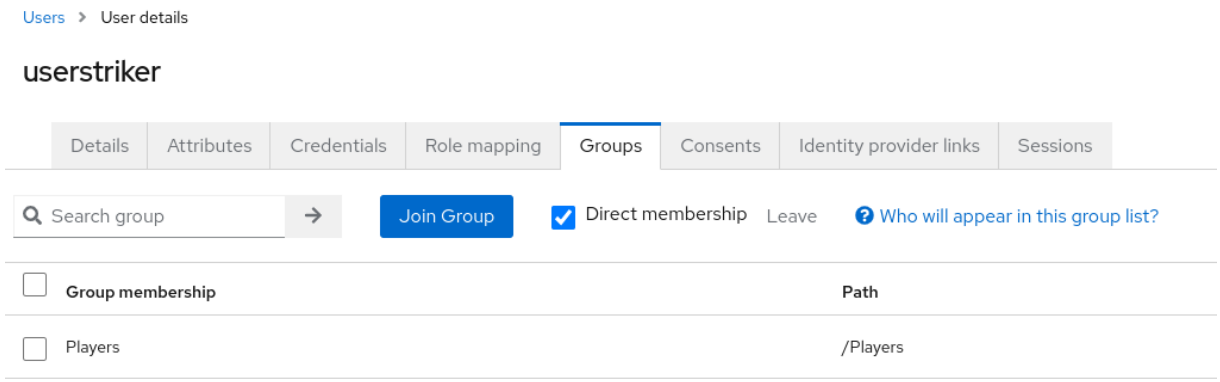
Slika 4.4. Primjer grupe i njezinih dodijeljenih uloga u Keycloak konzoli.

Administracijska konzola između ostalog ima sposobnost kreiranja korisnika čije aplikacije pristupaju Keycloak servisima. Prikaz kreiranih korisnika koji mogu pristupati aplikacijama koje koriste *FootBallManagent* instancu prikazan je na slici 4.5.



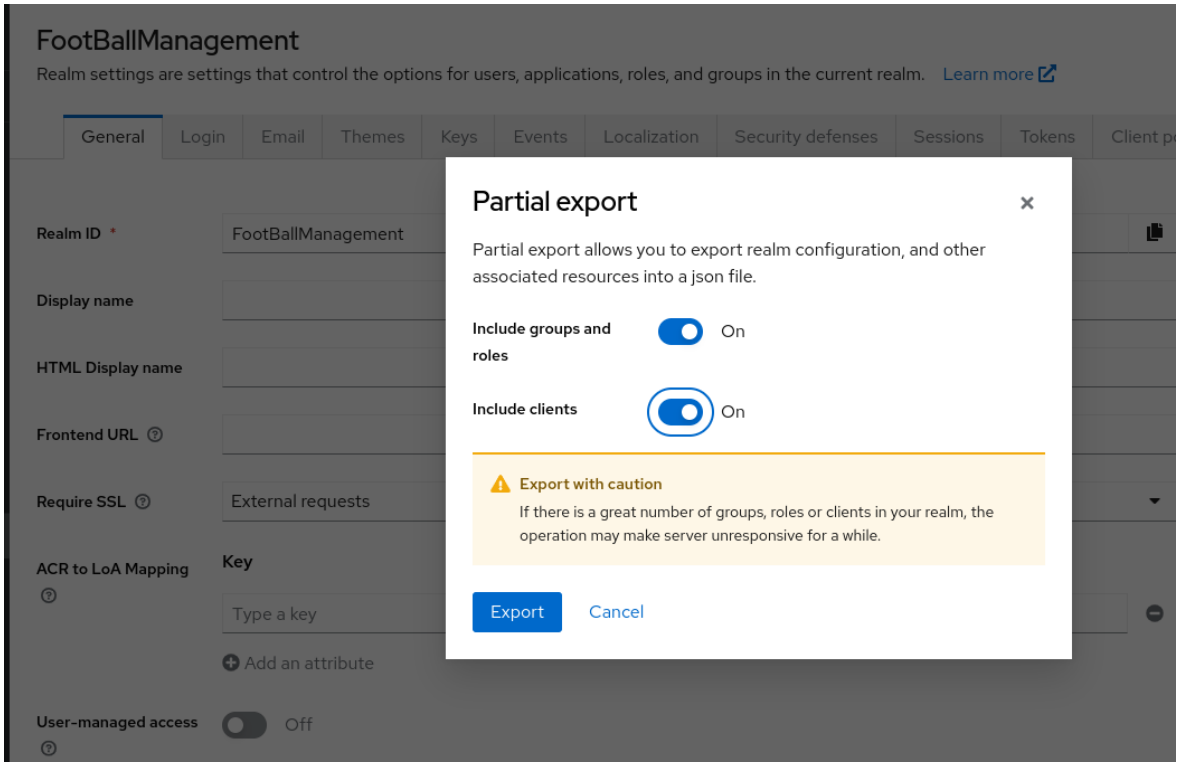
Slika 4.5. Prikaz kreiranih korisnika za *FootBallManagement*.

Svaki korisnik svrstan je u odgovarajuću grupu ovisno u njegovom položaju u hijerarhiji nogometnog kluba. Primjer stvorenog člana nogometnog kluba osoba je korisničkog imena *userstriker* dodijeljen grupi Igrača (engl. *Players*) što je prikazano na slici 4.6.



Slika 4.6. Prikaz grupe dodijeljene korisniku.

Nakon što su kreirani svi potrebni dijelovi za *FootBallManagement* instancu, programer je u mogućnosti izvesti JSON datoteku koja sadrži strukturu kreirane instance koja omogućuje daljnji razvoj aplikacije za rad s Keycloakom što je prikazano na slici 4.7.



Slika 4.7. Prikaz izvoza nepotpune konfiguracije *FootBallManagement* instance.

JSON datoteka, koja je za potrebe ostvarenog programskog rješenja nazvana *realm.json* sadržava nepotpunu konfiguraciju *FootBallManagement* instance. Korisnici i njihove dodijeljene uloge nisu spremljeni u datoteci. Podaci o kreiranim korisnicima mogu se dobiti putem HTTP GET zahtjeva koji treba biti izvršen dok je *FootBallManagement* instanca još uvijek aktivna. Zahtjev poslan zajedno s pristupnim tokenom prema *FootBallManagement* instanci vraća JSON vrijednosti za kreirane korisnike. Te vrijednosti dodaju se u izvezenu JSON datoteku *FootBallManagement* instance i implementiraju u Spring Boot poslužiteljskom dijelu aplikacije. Kreirani programski kod korišten za slanje zahtjeva i dohvat korisničkih podataka prikazan je na slici 4.8.

```
### Master Admin
POST {{keycloak_host}}/realms/master/protocol/openid-connect/token
Content-Type: application/x-www-form-urlencoded

client_id=admin-cli&username=admin&password=password&grant_type=password
> {% client.global.set("access_token", response.body.access_token); %}

### Get List Of Users
GET {{keycloak_host}}/admin/realms/{{realm}}/users
Authorization: Bearer {{access_token}}
Content-Type: application/json
```

Slika 4.8. Prikaz HTTP GET zahtjeva za dobivanje korisničkih podataka u JSON formatu.

Realm.json datoteka pomaže programeru izbjeći ponovno definiranje svih postavki i korisnika *FootBallManagement* instance. Ponovno pokretanje *FootBallManagement* instance, nakon implementacije *realm.json* datoteke u poslužiteljskom dijelu aplikacije, kreiraju se i pokreću Docker spremnici koji osiguravaju rad aplikacije i *FootBallManagement* instance.

Docker-compose.yml datoteka sadrži konfiguraciju koja podržava rad više Docker spremnika istodobno, neophodnih za nesmetano funkcioniranje aplikacije. Izvršenjem datoteke kreiraju se i pokreću potrebni spremnici koji služe za pokretanje:

- pozadinske strane aplikacije,
- Keycloaka za IAM,
- i PostgreSQL baze podataka.

Docker-compose.yml datoteka s njezinom konfiguracijom prikazana je na slici 4.9.

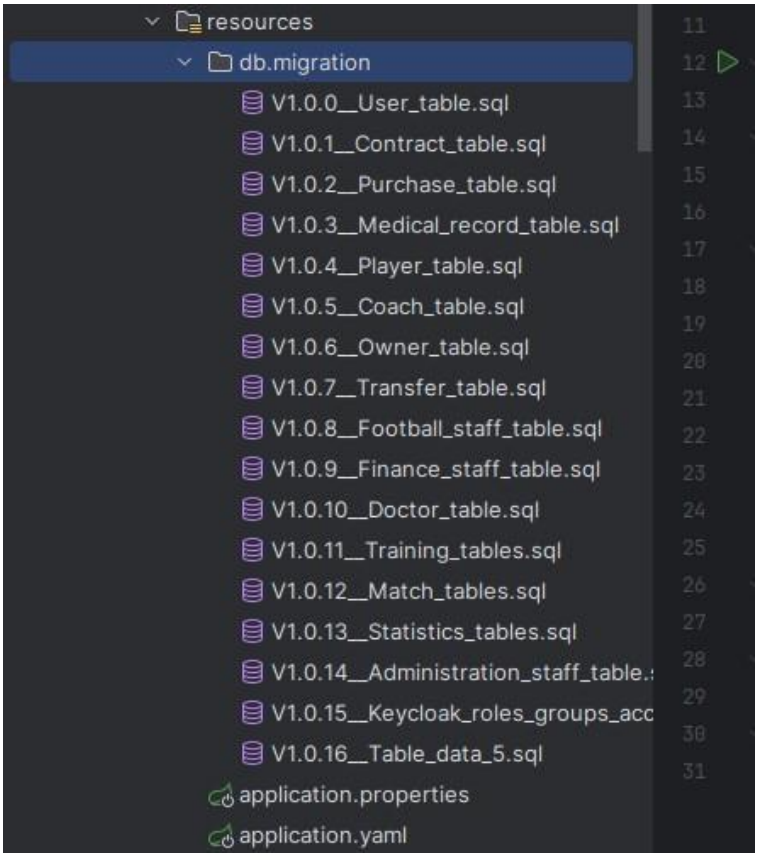
```yaml
version: '3.3'

services:
  postgres:
    image: "postgres"
    environment:
      POSTGRES_USER: root
      POSTGRES_PASSWORD: root
    ports:
      - "5432:5432"

  keycloak:
    image: quay.io/keycloak/keycloak:latest
    command:
      - start-dev
      - --import-realm
    environment:
      KC_DB: postgres
      KC_DB_URL_HOST: postgres
      KC_DB_URL_PORT: 5432
      KC_DB_URL_DATABASE: postgres
      KC_DB_USERNAME: root
      KC_DB_PASSWORD: root
      KEYCLOAK_ADMIN: admin
      KEYCLOAK_ADMIN_PASSWORD: password
    ports:
      - "8000:8080"
    depends_on:
      - postgres
    volumes:
      - "./realm.json:/opt/keycloak/data/import/realm.json"
```

Slika 4.9. *Docker-compose.yml* datoteka.

U bazi podataka tipa PostgreSQL kreirane su tablice za korisnike (engl. *user*) i ostale entitete koji su potrebni aplikaciji za upravljanje nogometnim klubom. Pri migraciji baze korišten je Flyway alat, koji podržava rad s PostgreSQL bazama podataka i SQL skriptama. Prikaz svih stvorenih tablica u bazi podataka vidljiv je na slici 4.10.



Slika 4.10. Baza podataka pozadinskog djela aplikacije.

Svim objektima koji su potrebni za rad aplikacije kreirana je njihova zasebna klasa entitet. Uz zasebnu entitet klasu, za svaki objekt kreiran je pripadni repozitorij koji omogućava pristup odgovarajućoj bazi podataka, kontroler kojim se pristupa tim podacima preko poslanih zahtjeva, servis koji upravlja logikom vezanom za kreiranje i rad s podacima te ostale komponente koje su potrebne za njihovo rukovanje u pozadinskom dijelu Spring Boot aplikacije.

U pozadinskoj strani aplikacije implementirana je cjelovita logika za uporabu Keycloak servisa i njegovih pogodnosti. Kada je HTTP zahtjev poslan na pozadinsku stranu aplikacije s pripadnim JWT tokenom korisnika, dolazi do njegove provjere korištenjem *KeycloakFilter.java* klase. *KeycloakFilter.java* klasa provjerava postoji li autorizacijsko zaglavlje u predanom tokenu (engl. *Bearer*). Taj token se onda dekodira te se potvrđuje njegova ispravnost uporabom tajnog ključa koji je dohvaćen iz Keycloaka. U slučaju da je token ispravan, zahtjev se obrađuje, inače se izbacuje iznimka koja prekida izvršenje zahtjeva. Ostale kreirane klase koje sudjeluju u toj provjeri su:

- *KeycloakPrincipal.java* – predstavlja trenutnog korisnika unutar aplikacije i sadrži njegove podatke i pripadni JWT token

- *KeycloakAuthenticationToken* – predstavlja trenutnu sesiju u kojoj je korisnik prijavljen

a prikazane su na slikama 4.11 i 4.12.

```java
package com.example.backend.security.keycloak_security;

import ...

public class KeycloakPrincipal implements KeycloakUserDetails {  10 usages  tturjak

    private final String userId;  2 usages
    private final String username;  2 usages
    private final List<SimpleGrantedAuthority> authorities;  2 usages
    private final AccessToken accessToken;  4 usages

    public KeycloakPrincipal(String clientId, AccessToken accessToken) {  1 usage  tturjak
        this.userId = accessToken.getSubject();
        this.username = accessToken.getPreferredUsername();
        this.authorities = AccessTokenUtils.getAuthorities( clientName: clientId, accessToken);
        this.accessToken = accessToken;
    }
```

Slika 4.11. Prikaz dijela *KeycloakPrincipal.java* klase.

```java
package com.example.backend.security.keycloak_security;

import ...

public class KeycloakAuthenticationToken extends AbstractAuthenticationToken {  5 usages  tturjak

    private final KeycloakPrincipal principal;  3 usages

    public KeycloakAuthenticationToken(KeycloakPrincipal principal) {  1 usage  tturjak
        super(principal.getAuthorities());
        this.setAuthenticated(principal.isEnabled());
        this.principal = principal;
    }

    @Override  no usages  tturjak
    public AccessToken getCredentials() { return principal.getAccessToken(); }

    @Override  tturjak
    public KeycloakPrincipal getPrincipal() { return principal; }
}
```

Slika 4.12. *KeycloakAuthenticationToken.java* klasa.

Komunikacija između pozadinske strane aplikacije te korisničke strane web i mobilne aplikacije ostvarena je kreiranjem *WebConfig.java* klase. *WebConfig.java* klasa omogućava primanje zahtjeva koji su poslani s definiranih izvorišta, a za trenutno programsko rješenje to su:

- izvorište korisničke strane web aplikacije: *http://localhost:4200*,

- izvorište mobilne aplikacije: *http://10.0.2.2:8080.*

što je prikazano na slici 4.13.

```
package com.example.backend.configuration;

import ...

@Configuration  tturjak
public class WebConfig implements WebMvcConfigurer {

    @Override  no usages  tturjak
    public void addCorsMappings(CorsRegistry registry) {
        registry.addMapping( pathPattern: "/**")
                .allowedOrigins( ...origins: "http://localhost:4200", "http://10.0.2.2:8080")
                .allowedMethods( ...methods: "GET", "POST", "PUT", "DELETE", "OPTIONS")
                .allowedHeaders( ...headers: "*")
                .allowCredentials( allowCredentials: true);
    }
}
```

Slika 4.13. Konfiguracija pozadinske strane aplikacije koja omogućava komunikaciju s korisničkom stranom aplikacije i mobilnom aplikacijom.

Pokretanjem pozadinske strane aplikacije omogućuje se njezino korištenje kao servisa za web i mobilno rješenje te su dostupne implementirane sigurnosne postavke koje se pokreću preko klase prikazane na slici 4.14.

```
@SpringBootApplication  tturjak
public class BackendApplication {

    public static void main(String[] args) { SpringApplication.run( primarySource: BackendApplication.class, ...args: args); }

}
```

Slika 4.14. Prikaz metode za pokretanje pozadinske strane aplikacije.

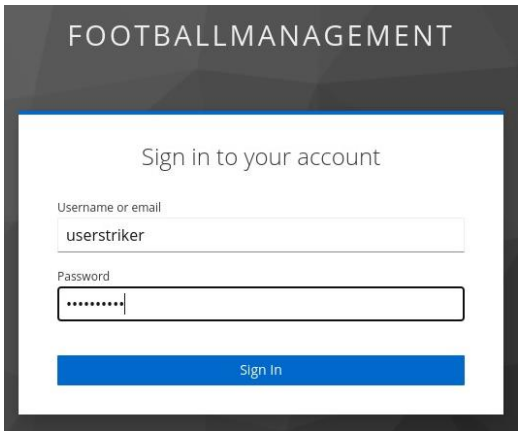
4.3. Korisnički dio ostvarenog programskog rješenja u web aplikaciji

Za kreiranje korisničke strane aplikacije web programskog rješenja korištena je Angular biblioteka. Potrebno je bilo inicijalizirati Keycloak pri pokretanju aplikacije te ga povezati s kreiranom Keycloak instancom *FootBallManagement* kako bi se omogućila prijava korisnika u Angular aplikaciju preko Keycloaka. Inicijalizacija Keycloaka za Angular aplikaciju prikazana je na slici 4.15.

```
function initializeKeycloak(keycloak: any) :() => any { Show usages  tturjak
  return () :any =>
      keycloak.init({
        config: {
          url: 'http://localhost:8000',
          realm: 'FootBallManagement',
          clientId: 'footballmanagementclient'
        },
        initOptions: {
          onLoad: 'login-required',
          checkLoginIframe: false
        }
      });
}
```

Slika 4.15. Inicijalizacija Keycloaka u Angular aplikaciji.

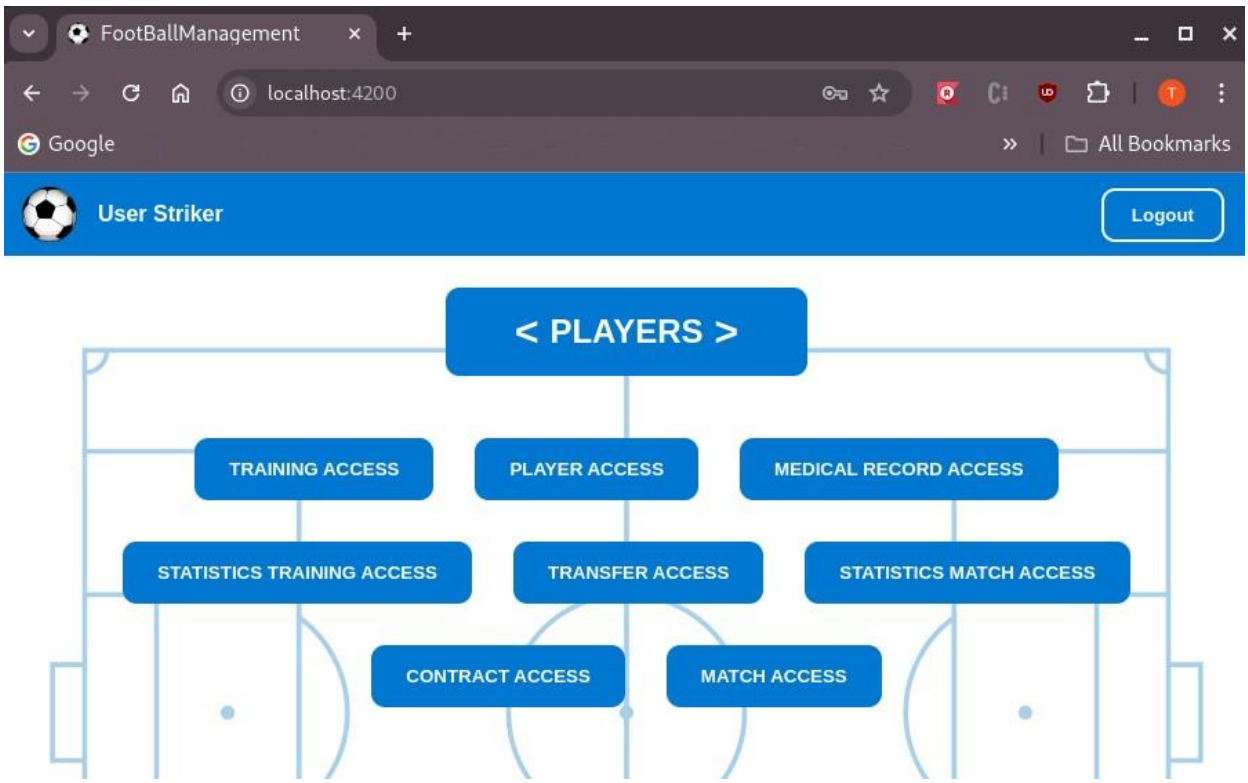
Inicijalizacija obuhvaća poveznicu kojom se pristupa Keycloaku, naziv instance, identitet klijenta kojem pristupamo te je postavljen zahtjev za autentifikaciju korisnika putem njegove prijave s korisničkim imenom i lozinkom. Time je omogućena prijava korisnika u aplikaciju. Uspješnim pokretanjem Angular aplikacije, web preglednik prikazuje stranicu za prijavu korisnika putem Keycloaka što je vidljivo na slici 4.16.



Slika 4.16. Prijava u aplikaciju preko Keycloak instance *FootBallManagement*.

Iznad polja za unos podataka za prijavu korisnika vidljiv je naziv Keycloak instance preko koje se odvija autentifikacija korisnika što je za ovo programsko rješenje *FootBallManagement*.

Uspješnom autentifikacijom web preglednik preusmjerava korisnika na početnu stranicu Angular aplikacije koja se nalazi na poveznici *http://localhost:4200*, što je prikazano na slici 4.17.



Slika 4.17. Prikaz stanja web preglednika nakon uspješne autentifikacije.

Kako bi se korisniku koji pristupa aplikaciji omogućio pristup svim resursima kojima ima pravo pristupa, potrebno je nakon uspješne prijave pročitati pristupni token. Iz pristupnog tokena čitaju se osnovne informacije o korisniku, kojoj grupi pripada te koje uloge nasljeđuje ta grupa. Prema dodijeljenim ulogama za korisnika generiraju se gumbovi koji korisniku omogućuju pristup podacima za koje ima dozvolu. Time se osigurava da korisnik ne može pristupiti drugim resursima za koje nema pravo pristupa. Metode koje prikupljaju i obrađuju te informacije nakon prijave prikazane su na slici 4.18.

```
ngOnInit() : void {  no usages  tturjak
  this.keycloakService.isLoggedIn().then(isLoggedIn : boolean  => {
    if (isLoggedIn) {
      console.log('User Profile:', this.keycloakService.getKeycloakInstance().idTokenParsed);
      this.updateUserInfo();
      this.generateRoleButtons();
```

Slika 4.18. Prikaz metoda pokrenutih uspješnom autentifikacijom.

Pritiskom na generirani gumb, korisnik šalje zahtjev na pozadinsku stranu aplikacije koja provjerava korisnikov identitet te ga autorizira za pristup podacima iz baze podataka. Primjer kreirane logike dohvaćanja podataka o igračima vidljiv je na slikama 4.19. i 4.20.

```
import { Injectable } from '@angular/core';
import { HttpClient, HttpHeaders } from '@angular/common/http';
import { AuthService } from './auth.service';
import { Observable } from 'rxjs';

@Injectable({  Show usages  tturjak
    providedIn: 'root'
})
export class ApiService {
    private apiUrl :string  = 'http://localhost:8080';

    constructor(private http: HttpClient, private authService: AuthService) {}


    async getPlayersData(): Promise<Observable<any>> {  Show usages  tturjak
        try {
            const jwtToken :string  = await this.authService.getJwtToken();
            if (!jwtToken) {
                throw new Error('JWT token not available');
            }

            const headers :HttpHeaders  = new HttpHeaders({
                'Authorization': `Bearer ${jwtToken}`
            });

            console.log('Headers retrieved:', headers);

            return this.http.get(`${this.apiUrl}/players`, { headers });
        } catch (error) {
            console.error('Error retrieving JWT token:', error);
            throw error;
```

Slika 4.19. Prikaz dijela API metode za dohvaćanje podataka s pozadinske strane aplikacije.

```
export class AuthService {
    constructor(private keycloakService: KeycloakService) {}  no usages  tturjak

    async getJwtToken(): Promise<string> {  Show usages  tturjak
        try {
            const token : string  = await this.keycloakService.getToken();
            console.log('Token retrieved:', token);
            return token;
        } catch (error) {
            console.error('Error retrieving token:', error);
            throw error;
        }
    }
```

Slika 4.20. Prikaz metode koja koristi JWT token za autorizaciju.

Kako bi se uspjeli dobiti podaci s baze podataka koja se nalazi na pozadinskoj strani aplikacije, autorizacijski servis od *FootBallManagement* instance zahtjeva JWT token. Nakon što je token primljen, Angular aplikacija šalje zahtjev pozadinskoj strani aplikacije s dobivenim tokenom:

- 'Authorization': 'Bearer ${jwtToken}'

te zahtjeva povrat podataka s tražene krajnje točke, primjerice, podatke o igračima:

- '${this.apiUrl}/players'.

Uspješno dohvaćanje podataka iz baze podataka s pozadinske strane aplikacije vidljivo je na slici 4.21.

Slika 4.21.Prikaz uspješnog dohvaćanja podataka.

Odjava korisnika iz aplikacije odvija se pozivanjem Keycloak servisa koji je omogućio prijavu korisnika. Brišu se podatci o prethodno prijavljenom korisniku te se za svakog novog prijavljenog korisnika ponavlja postupak dohvaćanja podataka kako bi korisnici imali uvid samo u svoje podatke.

4.4. Korisnički dio ostvarenog programskog rješenja u mobilnoj aplikaciji

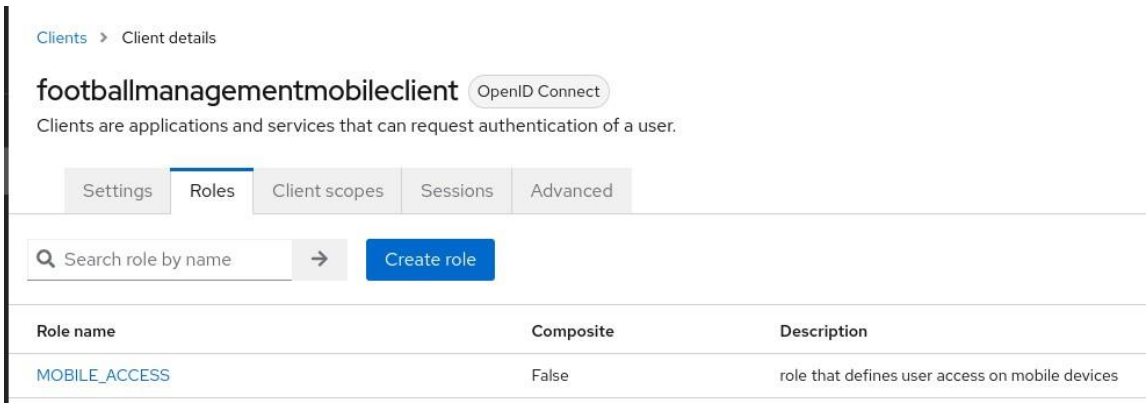
Za potrebne mobilnog programskog rješenja napravljen je zaseban klijent za mobilnu aplikaciju te je za njega definirana preusmjeravajuća poveznica koja omogućava da se korisnika preusmjeri u mobilnu aplikaciju nakon uspješne prijave. Korištena poveznica koja je postavljena na *footballmanagementmobileclient* klijentu za preusmjeravanje korisnika na mobilnu aplikaciju je *http://10.0.2.2:8000/callback*. Kao i u korisničkoj strani aplikacije web programskog rješenja, mobilna aplikacija poziva Keycloak servis za prijavu korisnika. Poveznica koji se poziva za Keycloak autentifikaciju kreirane *FootBallManagement* instance prikazana je na slici 4.22.

```
String kcLoginUrl = "http://10.0.2.2:8000/realms/FootBallManagement/protocol/openid-connect/auth" +
        "?response_type=code" +
        "&client_id=footballmanagementmobileclient" +
        "&redirect_uri=http://10.0.2.2:8000/callback" +
        "&scope=openid profile email";

kcLoginWebView.loadUrl(kcLoginUrl);
```

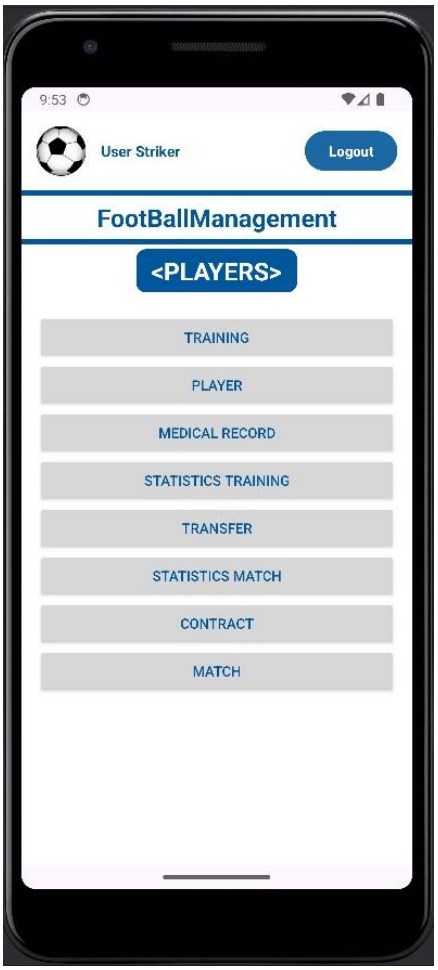
Slika 4.22. Prikaz poveznice za pristup prijavi Keycloak instance *FootBallManagement*-a.

Također, u klijentu mobilne aplikacije *footballmanagementmobileclient* dodana je uloga MOBILE_ACCESS koja pruža pristup korisnicima mobilnom klijentu što je prikazano na slici 4.23.



Slika 4.23. Prikaz uloge dodijeljene isključivo jednom klijentu.

Ako korisnik ne pripada grupi koja ima MOBILE_ACCESS kao dodijeljenu ulogu, onda nema pravo pristupa instanci *FootBallManagementa* preko mobilne aplikacije. Aplikacija prekida proces prijave te preusmjerava korisnika natrag na stranicu za prijavu. Ako je uloga prisutna, Keycloak preusmjerava korisnika na početnu stranicu mobilne aplikacije, prikazano na slici 4.24.



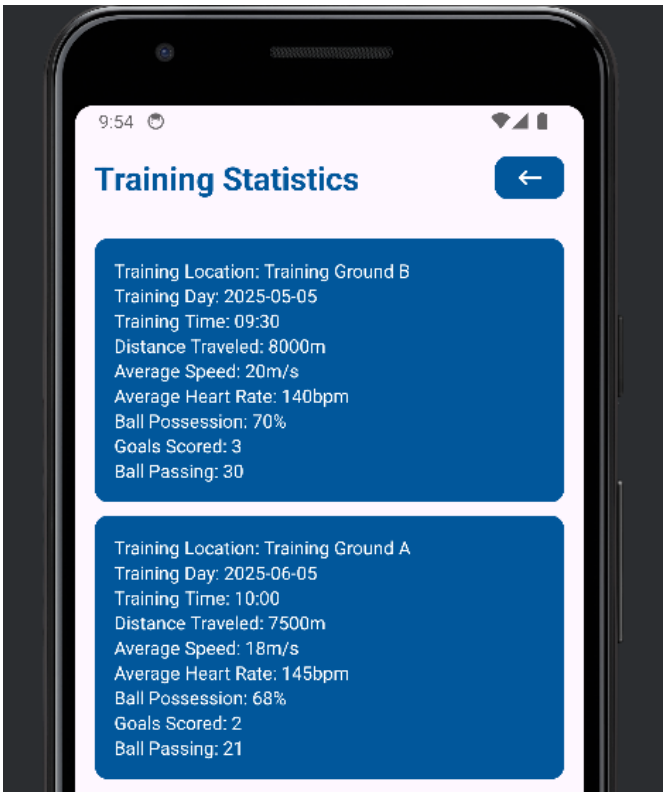
Slika 4.24. Prikaz mobilne aplikacije nakon uspješne prijave.

Primjer kreiranog API servisa za dohvaćanje podataka kojima korisnik pristupa prikazan je na slici 4.25.

```
JSONArray jsonArray = new JSONArray(responseData);
for (int i = 0; i < jsonArray.length(); i++) {
    JSONObject jsonObject = jsonArray.getJSONObject(i);
    Player player = new Player();
    player.setPlayerName(jsonObject.getString( name: "playerName"));
    player.setDateOfBirth("Date of Birth: " + jsonObject.getString( name: "dateOfBirth"));
    player.setPlayerRole("Role: " + jsonObject.getString( name: "playerRole"));
    player.setClubName("Club Name: " + jsonObject.getString( name: "clubName"));
    playerList.add(player);
}
adapter.notifyDataSetChanged();
```

Slika 4.25. Prikaz dijela API servisa za dohvaćanje podataka preko mobilne aplikacije.

Te metode osiguravaju mobilnim korisnicima nesmetano dohvaćanje podataka s pozadinske strane aplikacije. Primjer rezultata dohvaćenih podataka u mobilnoj aplikaciji prikazan je na slici 4.26.



Slika 4.26. Prikaz uspješno dohvaćenih podataka u mobilnoj aplikaciji.

Odjava korisnika iz mobilne aplikacije odvija se brisanjem svih kolačića koji su bili vezani uz prethodno kreiranog korisnika.

5. VALIDACIJA PROGRAMSKOG RJEŠENJA

Za potrebe testiranja i validacije programskog rješenja, dobivena programska rješenja analizirat će se s gledišta funkcionalnosti i sigurnosti. Zasebno će se testirati web i mobilna aplikacija kako bi se utvrdilo ispunjavaju li zadane funkcionalnosti te zadovoljavaju li odgovarajuću razinu sigurnosti i zaštite.

Analizirati i testirat će se slučajevi prijavljivanja u aplikacije i provjeravati dodijeljena razina prava pristupa korisnika nakon prijave u web i mobilnu aplikaciju. Testni slučajevi razmatrani prilikom prijavljivanja u aplikacije obuhvaćat će slučajeve uspješne i neuspješne prijave sa zadanim korisničkim podacima. U slučajevima uspješne prijave, za svakog korisnika analizirat će se razina pristupa i funkcionalnosti aplikacije kojima korisnik može pristupiti što će biti uspoređeno s njegovom dodijeljenom ulogom i pravima pristupa unaprijed utvrđenim prilikom definiranja zahtjeva na softver.

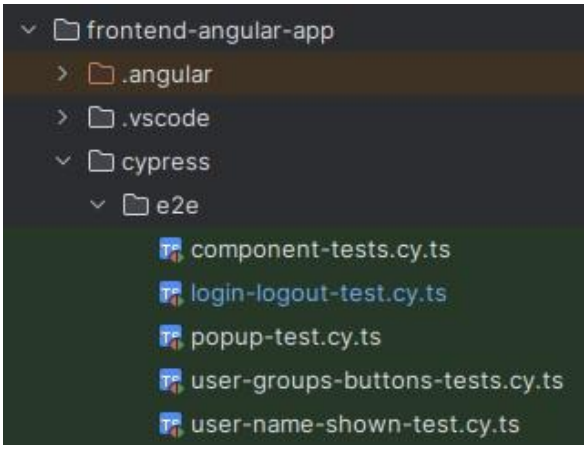
Za validaciju web aplikacije koristit će se alat za automatizirano web testiranje Cypress, detaljnije opisan u potpoglavlju 4.1.9. dok će se za testiranje mobilne aplikacije koristiti Appium alat za testiranje mobilnih aplikacija, koji je opisan u potpoglavlju 4.1.10.

5.1. Validacija web aplikacije

Testiranje web aplikacije provedeno je validacijom pozadinske i korisničke strane aplikacije programskog rješenja te korištenjem Cypressa, alata za automatizirano testiranje. Cypress je potrebno instalirati u direktorij korisničke strane programskog rješenja. Nakon uspješne instalacije alata, potrebno je stvoriti konfiguracijski dokument koji omogućuje pokretanje i izvršavanje Cypress testova. Datoteka potrebna za konfiguraciju Cypressa (*cypress.config.ts*) prikazana je na slici 5.1.

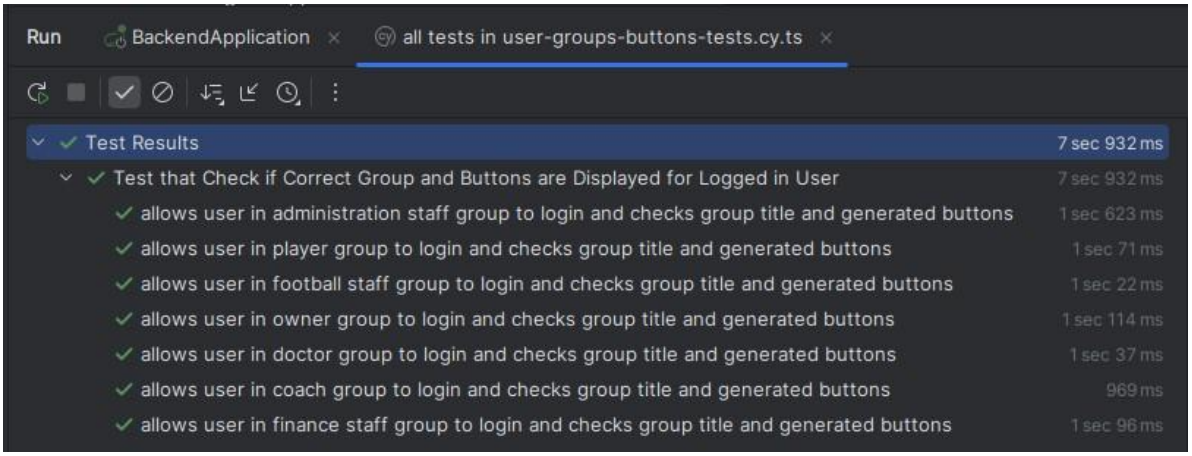
Slika 5.1. Prikaz cypress.config.ts konfiguracije za testiranje web aplikacije.

Konfiguracija Cypressa omogućuje pristup korisničkoj strani aplikacije za izvršavanje testova. Testovi se kreiraju u direktoriju *cypress* pod *e2e* poddirektorijem kako bi Cypress aplikacija mogla pronaći testove i pokrenuti ih. Prikaz testova web aplikacije vidljiv je na slici 5.2.



Slika 5.2. Prikaz Cypress testova za web aplikaciju.

Cypress testove moguće je pokrenuti u projektu ili unutar Cypress aplikacije. Neki od testova pokrenutih unutar korisničke strane aplikacije prikazani su na slici 5.3.



Slika 5.3. Prikaz pojedinih Cypress testova unutar projekta.

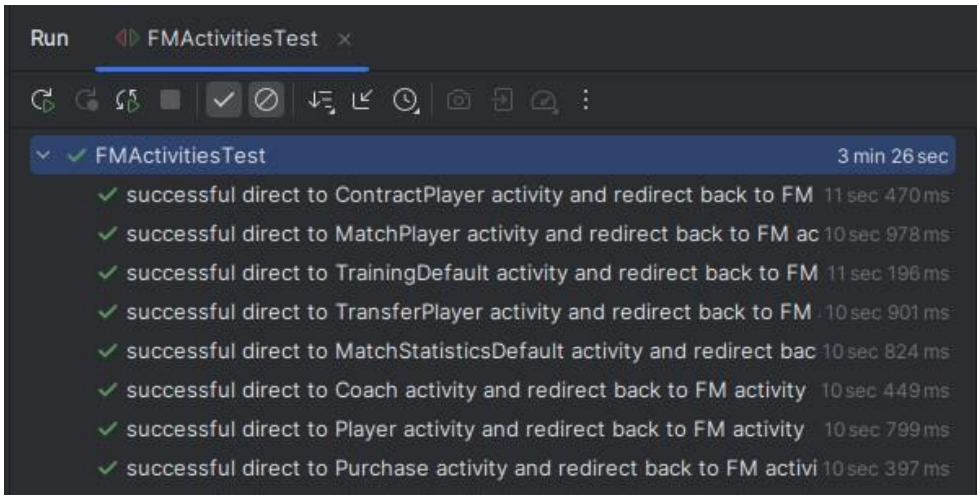
Cypress testovi pokrenuti direktno u korisničkoj strani aplikacije prikazuju konačan rezultat testova bez međukoraka koji se izvršavaju u pozadini. Međukoraci testiranja vidljivi su pokretanjem Cypress aplikacije. Otvaranjem Cypress aplikacije te odabirom testiranja komunikacije među krajnjim točkama (engl. *E2E Testing*) aplikacija je spremna izvršiti testiranje. Odabirom željene datoteke izvršavaju se pripadajući testovi. Svi testovi provedeni nad web aplikacijom uspješno su izvršeni te su navedeni u tablici 5.1.

Tablica 5.1. Uspješno izvršeni testovi provedeni nad web aplikacijom.

TEST	*RADNJE*
login-logout-test.cy.ts	• ispituje uspješnu prijavu korisnika i njegovu odjavu iz aplikacije • ispituje neuspješnu prijavu korisnika ako korisničko ime ne postoji u sustavu • ispituje neuspješnu prijavu korisnika ako korisnikova lozinka nije ispravna
user-name-shown-test.cy.ts	• ispituje prikaz ispravnog korisničkog imena u aplikaciji nakon prijave korisnika
popup-test.cy.ts	• ispituje prikaz osobnih podataka korisnika
user-groups-buttons-tests.cy.ts	• ispituje uspješnu generaciju gumbova za pristup podacima iz baze podataka s poslužiteljske aplikacije
component-tests.cy.ts	• ispituje komponentu za dohvaćanje podataka iz baze

5.2. Validacija mobilne aplikacije

Testiranje programskog rješenja mobilne aplikacije provodi se korištenjem Appium poslužitelja koji omogućava pokretanje testova za validaciju mobilne aplikacije. Uz njega koristi se i Appium inspektor za lakše kreiranje testova pomoću emulatora koji sadrži aplikaciju za testiranje. Prikaz pojedinih uspješno izvršenih Appium testova vidljiv je na slici 5.4.



Slika 5.4. Prikaz dijela uspješno izvršenih Appium testova za mobilnu aplikaciju.

Svi testovi unutar Appiuma provedeni nad mobilnom aplikacijom uspješno su izvršeni, a prikazani u tablici 5.2.

Tablica 5.2. Uspješno izvršeni testovi provedeni nad mobilnom aplikacijom.

TEST	*RADNJE*
LoginTest.java	• ispituje uspješnu prijavu korisnika i preusmjeravanje u mobilnu aplikaciju • ispituje neuspješnu prijavu korisnika s pomoću krivog korisničkog imena • ispituje neuspješnu prijavu korisnika s pomoću krive lozinke • ispituje neuspješnu prijavu korisnika zbog nedozvoljenog pristupa
LogoutTest.java	• ispituje uspješnu odjavu korisnika iz mobilne aplikacije i preusmjeravanja na Keycloak stranicu za prijavu korisnika
UserNameTest.java	• ispituje uspješno dohvaćanje korisničkog imena nakon prijave u mobilnu aplikaciju
UserGroupMembershipTest.java	• ispituje uspješno dohvaćanje korisnikove grupe nakon prijave u mobilnu aplikaciju
UserGroupMembershipAccessTest.java	• ispituje uspješnu generaciju gumbova za pristup podacima iz baze podataka za testiranje vezanih uz nogometni klub za korisnike koji pripadaju pojedinoj grupi
FMActivitiesTest.java	• ispituje pristup aktivnostima koje dohvaćaju podatke iz baze podataka zasebno
FMDataCheckActivitiesTest.java	• ispituje dohvaćanje podataka iz baze podataka za svaku aktivnost zasebno

6. ZAKLJUČAK

Upravljanje pristupom i identitetom u poslužiteljskim i klijentskim aplikacijama ključno je za ostvarivanje sigurnosti i pouzdanosti u modernim aplikacijama koje sadržavaju i obrađuju osjetljive podatke. Koristeći autentifikaciju i autorizaciju za provjeru identiteta korisnika i određivanja njihovih prava pristupa, osigurava se zaštita od zloupotrebe podataka iz baze poslužiteljske aplikacije i njezinih funkcionalnosti.

U ovom radu izrađeno je programsko rješenje za upravljanje nogometnim klubom i njegovim članovima, pri čemu je posebna pažnja posvećena implementaciji sigurnosnih mjera, i to korištenjem autentifikacije i autorizacije. Sigurnost podataka i korisnika postignuta je korištenjem Keycloak alata, poslužitelja za upravljanje pristupom i identitetom. Napravljeno programsko rješenje demonstrira rad i funkcionalnosti koje su potrebne kako bi se uspostavio jedan siguran IAM sustav.

Daljnje nadogradnje aplikacije mogle bi uključiti podršku za više nogometnih klubova istovremeno, mogućnost izmjene iz baze te bi se u njega mogle implementirati druge autentifikacijske metode, poput dvostruke provjere autentičnosti ili prijavu putem biometrije.

LITERATURA

[1] S. Gittlen, L.Rosencrance, „What is identity and access management? Guide to IAM“, [online], TechTarget, 2021, dostupno na: https://www.techtarget.com/searchsecurity/definition/identity-access-management-IAM-system, [25.6.2023.]

[2] „How to Implement Identity & Access Management (IAM) in 5 Easy Steps?“, [online], vSecureLabs, 2022, dostupno na: https://vsecurelabs.co/how-to-implement-identity-and-access-management/, [25.6.2023.]

[3] J. Turner, J. McCarthy, „The Definitive Guide to Identity and Access Management (IAM)“, [online], strongdm, 2023, dostupno na: https://www.strongdm.com/iam, [26.6.2023.]

[4] „OAuth 2.0“, [online], OAuth 2.0, dostupno na: https://oauth.net/2/, [30.7.2024.]

[5] „Architecture of Identity Access Management in Cloud Computing“, [online], GeeksForGeeks, dostupno na: https://www.geeksforgeeks.org/architecture-of-identity-access-management-in-cloud-computing/, [25.6.2023.]

[6] „8 Protocols Used in Identity and Access Management“, [online], SailPoint, 2021, dostupno na: https://www.sailpoint.com/identity-library/identity-management-protocols/ , [26.6.2023.]

[7] H. Mousa, „Best 5 Open Source Identity Management Solutions (IAM) For Enterprise for 2023“, [online], Medevel, 2023., dostupno na: https://medevel.com/5-iam-enterprise/, [26.6.2023.]

[8] L. F. Pantoffi, „Case Study Selection: Netflix’s Utilization of Cloud Services“ , [online], Medium, 2024, dostupno na: https://medium.com/@luisfernandopantolfi/case-study-selection-netflixs-utilization-of-cloud-services-da6627bf6004, [25.7.2024.]

[9] E. Guerrant, „Access: A new portal for managing intern authorization“, [online], Discord, 2024, dostupno na: https://discord.com/blog/access-a-new-portal-for-managing-internal-authorization, [25.7.2024.]

[10] „Identify Your Users and Manage Access“, [online], Salesforce, dostupno na: https://help.salesforce.com/s/articleView?id=sf.identity_overview.htm&type=5, [25.7.2024.]

[11] „Functional and Nonfunctional Requirements: Specification and Types“, [online], Altexsoft, 2024., dostupno na: https://www.altexsoft.com/blog/functional-and-non-functional-requirements-specification-and-types/, [21.6.2024.]

[12] „What is User Story?“, [online], Visual Paradigm, dostupno na: https://www.visual-paradigm.com/guide/agile-software-development/what-is-user-story/ , [13.9.2024.]

[13] „Functional Requirements“, [online], OfniSystems, dostupno na: https://www.ofnisystems.com/services/validation/functional-requirements/ , [14.9.2024.]

[14] „BDD (Behavior Driven Development)“, [online], Robot Framework, dostupno na: https://docs.robotframework.org/docs/testcase_styles/bdd, [14.9.2024.]

[15] „Given–When–Then“, [online], Agile Alliance, 2024., dostupno na: https://www.agilealliance.org/glossary/given-when-then/, [21.6.2024.]

[16] A. Radovan, I. Ogrizek Biškupić, J. Lopatić, „IT Employers' Expectations from their Employees Regarding Java Programming Language“, 11th International Conference on Information and Education Technolog, str. 233., 2023.

[17] „What is Java Spring Boot“, [online], Azure, dostupno na: https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/what-is-java-spring-boot/, [22.6.2023.]

[18] „What is TypeScript?“, [online], TypeScript, dostupno na: https://www.typescriptlang.org/, [22.6.2023.]

[19] „The web development framework for building the future“, [online], Angular, dostupno na: https://angular.io/, [22.6.2023.]

[20] „Android Studio“, [online], Android, dostupno na: https://developer.android.com/studio, [22.6.2023.]

[21] „Open Source Identity and Access Management“, [online], Keycloak, dostupno na: https://www.keycloak.org/, [22.6.2023.]

[22] „Develop faster. Run anywhere.“, [online], Docker, 2023, dostupno na: https://www.docker.com/, [22.6.2023.]

[23] „--distributed-even-if-your-workflow-isn“, [online], Git, dostupno na: https://git-scm.com/, [22.6.2023.]

[24] „Test. Automate. Accelerate.“, [online], Cypress, dostupno na: https://www.cypress.io/, [22.6.2023.]

[25] „Welcome-Appium Documentation“, [online], Appium, dostupno na: https://appium.io/docs/en/latest/, [15.6.2024.]

[26] „GUI Inspector tool for all kinds of apps, powered by Appium.“, [online], appium-inspector, dostupno na: https://github.com/appium/appium-inspector, [20.6.2024.]

[27] „Docker: Get started with Keycloak on Docker“, [online], Keycloak, dostupno na: https://www.keycloak.org/getting-started/getting-started-docker , [25.7.2024.]

SAŽETAK

Problem upravljanja identitetom i pristupom u poslužiteljskim i klijentskim aplikacijama ključan je izazov prilikom ostvarivanja sigurnosti aplikacija koje rukuju s osjetljivim podacima. U praksi privatnost podataka organizacija i korisnika aplikacije osigurava se uspostavom metoda autentifikacije i autorizacije. U ovom radu implementirano je programsko rješenje za upravljanje nogometnim klubom i njegovim članovima uključivanjem mehanizama za upravljanje pristupom i identitetom. Poslužiteljski dio programskog rješenja kreiran je korištenjem Spring Boot programskog okvira i pisan Java programskim jezikom. Za uspostavu autentifikacije i autorizacije korišten je Keycloak servis. Validacija rješenja provedena je testiranjem korisničkog dijela aplikacije i to web i mobilnog dijela, a na temelju korisnikove uporabe i dohvaćanja vrijednosti iz baze podataka poslužiteljskog dijela aplikacije ovisno o korisnikovom pravu pristupa.

Ključne riječi: autentifikacija, autorizacija, Keycloak, sigurnost, upravljanje identitetom i pristupom

ABSTRACT

Managing access and identity in server and client applications

The problem of identity and access management in server and client applications is a key challenge when securing applications that handle sensitive data. In practice, data privacy of organizations and users of the application is ensured by the implementation of authentication and authorization methods. In this work, software solution was implemented for management of a football club and its members using mechanisms for identity and access management. The server part of software solution was created using the Spring Boot framework and written in the Java programming language. For the establishment of authentication and authorization, Keycloak service was used. Validation of the solution was carried out by testing the user part of the application on web and mobile parts, based on user's use and retrieving values from the database of the service part of the application, depending on user's rights of access.

Keywords: authentication, authorization, identity and access management, Keycloak, security

ŽIVOTOPIS

Tea Turjak rođena je 10. rujna 1999. godine u Našicama. Školovanje započinje 2006. u Osnovnoj školi Dore Pejačević Našice koje završava 2014. godine. Iste godine upisuje prirodoslovno-matematičku gimnaziju u Srednjoj školi Isidora Kršnjavog u Našicama. Srednjoškolsko obrazovanje završava 2018. godine te nakon uspješno položene državne mature upisuje sveučilišni preddiplomski studij računarstva na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija u Osijeku. Sveučilišni preddiplomski studij računarstava završava 2021. godine te iste godine upisuje sveučilišni diplomski studij računarstva, modul programskog inženjerstva. Na drugoj godini diplomskog studija obavlja praksu u tvrtki DICE Digital Innovation Center.

_____

Potpis autora