Aplikacija za bilježenje rezultata streljaštva u programskom jeziku C#

Tumbas, Leo

Undergraduate thesis / Završni rad

2023

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Josip Juraj Strossmayer University of Osijek, Faculty of Electrical Engineering, Computer Science and Information Technology Osijek / Sveučilište Josipa Jurja Strossmayera u Osijeku, Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:200:269771

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-05-10**

Repository / Repozitorij:

Faculty of Electrical Engineering, Computer Science and Information Technology Osijek

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

Sveučilišni studij računarstva
Smjer računalno inženjerstvo

APLIKACIJA ZA BILJEŽENJE REZULTATA STRELJAŠTVA U PROGRAMSKOM JEZIKU C#

Završni rad

Leo Tumbas

Osijek, 2023.

SADRŽAJ

1. UVOD 1

1.1. Zadatak završnog rada 1

2. PROBLEM BILJEŽENJA REZULTATA U STRELJAŠTVU 2

2.1. Općenito o sportskom streljaštvu 2

2.2. Postojeća rješenja za bilježenje rezultata u streljaštvu 2

2.2.1. Shooting Sports Cloud 3

2.2.2. Shooters Diary 4

2.2.3. TargetScan ISSF Pistol & Rifle 5

2.2.4. Air Shooting Diary: Pro Notes 6

3. MODEL APLIKACIJE ZA BILJEŽENJE REZULTATA U STRELJAŠTVU 7

3.1. Funkcionalnosti aplikacije 7

3.1.1. Registracija korisnika 7

3.1.2. Prijava korisnika 7

3.1.3. Uređivanje korisničkih podataka 7

3.1.4. Brisanje korisničkog računa 8

3.1.5. Odjava korisnika 8

3.1.6. Unos novog zapisa 8

3.1.7. Pregled postojećih zapisa 8

3.1.8. Izmjena postojećeg zapisa 9

3.1.9. Brisanje postojećeg zapisa 9

3.2. Podržane discipline i pripadajući načini bodovanja 9

3.2.1. 10 m zračni pištolj 9

3.2.2. 10 m zračna puška 9

3.2.3. 25 m pištolj 10

3.2.4. 25 m pištolj brzo gađanje 10

3.2.5. 50 m puška trostav 10

3.2.6. Trap i skeet 10

4. PROGRAMSKO RJEŠENJE ZA BILJEŽENJE REZULTATA U STRELJAŠTVU 12

4.1. Korišteni alati i tehnologije 12

4.1.1. Visual Studio 2022 Preview 12

4.1.2. Microsoft SQL Server 2022 Developer 12

4.1.3. Docker 12

4.1.4. ASP.NET Core 13

4.1.5. Entity Framework Core 13

4.1.6. Programski jezik C# ... 13
4.1.7. HTML ... 13
4.1.8. CSS ... 13
4.1.9. JavaScript ... 14
4.2. Poslužiteljsko programsko rješenje ... 14
4.2.1. Baza podataka ... 14
4.2.2. Aplikacijsko programsko sučelje ... 16
4.3. Korisničko programsko rješenje ... 17
4.3.1. Autentifikacija korisnika ... 17
4.3.2. Uređivanje korisničkih podataka ... 19
4.3.3. Brisanje korisničkog računa ... 20
4.3.4. Odjava korisnika ... 21
4.3.5. Unos novog zapisa ... 22
4.3.6. Pregled postojećeg/-ih zapisa ... 23
4.3.7. Izmjena postojećeg zapisa ... 25
4.3.8. Brisanje postojećeg zapisa ... 25
5. UPORABA I TESTIRANJE APLIKACIJE ... 27
5.1. Uporaba aplikacije ... 27
5.2. Ispitivanje aplikacije ... 35
5.2.1. Ispitivanje točnosti izračuna vrijednosti ... 35
5.2.2. Ispitivanje izmjene općih podataka zapisa ... 36
5.2.3. Ispitivanje izmjene podataka pogodaka ... 37
6. ZAKLJUČAK ... 38
LITERATURA ... 39
POPIS SLIKA ... 41
SAŽETAK ... 42
ABSTRACT ... 43
ŽIVOTOPIS ... 44
PRILOZI ... 45

1. UVOD

Sportsko streljaštvo je streljački sport koji se može pobliže opisati kao „precizno streljaštvo“. U sportskom streljaštvu vrednuje se isključivo preciznost, po čemu se razlikuje od praktičnog streljaštva koje vrednuje i brzinu strijelca. Od prvih zapisa o sportskom streljaštvu do danas, sportsko streljaštvo je od sporta kojim se bave imućni i svi koji se žele doimati imućnima, postalo sport kojim se bavi vrlo mala skupina ljudi i koji iz godine u godinu ima sve manju publiku. Zbog ovakvog „izumiranja“ sporta, digitalizacija je gotovo nepostojeća te se i dalje najčešće rezultati pohranjuju u bilježnicama i ručno obrađuju.

Ovaj završni rad predstavlja internetsku aplikaciju kao jednu od mogućnosti digitalizacije pohrane rezultata sportskog streljaštva. Cilj aplikacije je korisniku omogućiti pohranu i pregled ostvarenih rezultata u bilo kojem trenutku i s bilo kojeg mjesta u svijetu. Izrađena aplikacija prije korištenja funkcionalnosti rada s rezultatima, od korisnika zahtijeva izradu korisničkog računa, kojeg će kasnije koristiti za prijavu te unos i pregled zapisa. Također, aplikacija je sposobna koristeći unesene podatke o pogodcima izračunati rezultate pojedinih serija unutar zapisa te rezultat cijelog zapisa, prema načinima bodovanja podržanih disciplina.

U drugom poglavlju općenito je opisano sportsko streljaštvo kroz povijest te trenutna situacija u svijetu i Republici Hrvatskoj, i objašnjeno je stanje digitalizacije sportskog streljaštva te su navedena četiri postojeća rješenja slična rješenju koje predstavlja ovaj rad. Treće poglavlje definira funkcionalnosti aplikacije i navodi relevantna pravila disciplina podržanih u aplikaciji te njihove načine bodovanja rezultata. U četvrtom poglavlju prikazuje se programsko rješenje ostvareno definiranim zahtjevima, dok peto poglavlje prikazuje očekivanu uporabu aplikacije te ispitivanje bitnih funkcionalnosti aplikacije.

1.1. Zadatak završnog rada

Cilj završnog rada je analizirati i objasniti potrebu za aplikacijom za bilježenje rezultata u streljaštvu te izraditi takvu aplikaciju i testirati ju. Prilikom izrade aplikacije vrlo je važno uzeti u obzir razinu poznavanja rada na računalu potencijalnih korisnika, i osigurati neometani i kontinuirani pristup aplikaciji iz bilo kojeg mjesta u svijetu.

2. PROBLEM BILJEŽENJA REZULTATA U STRELJAŠTVU

2.1. Općenito o sportskom streljaštvu

Pisana povijest sportskog streljaštva počinje 1720-ih godina, a 1784. godine je već osnovano i prvo hrvatsko udruženje sportskih strijelaca – *Građansko streljačko društvo „Osijek 1784“*, koje je ujedno i najstariji hrvatski sportski klub. Prema [1], prvi zapisi o međunarodnim udruženjima sportskih strijelaca potječu iz 1824. godine, kada je osnovano *Švicarsko društvo karabinjera (fran. Société Suisse des Carabiniers)*, a status olimpijskog sporta je sportsko streljaštvo dobilo već s prvim Ljetnim Olimpijskim igrama održanima 1896. godine.

Međunarodna federacija sportskog streljaštva (eng. International Shooting Sport Federation) danas je krovna svjetska organizacija sportskog streljaštva, koja se desetljećima bori sa savezima popularnih sportova za pozornost i medijski prostor uvođenjem novih formata i promjenama postojećih. Promjene nisu postigle željene rezultate, već su postigle potpuno suprotni ishod od željenoga: prema [2], od 1896. godine je ukinuto sveukupno 45 međunarodno priznatih formata natjecanja te je danas svega 19 formata međunarodno priznato. Kao olimpijski sport, prema [3], danas broji najmanje formata natjecanja na Ljetnim Olimpijskim igrama u svojoj povijesti, svega 7 formata, dok je na vrhuncu bilo 1972. godine, kada je na Ljetnim igrama brojilo 11 formata.

U Republici Hrvatskoj, prema [4, str. 7], strijelci se osim u 19 međunarodno priznatih formata mogu natjecati i u 14 nacionalno priznatih formata natjecanja. Razvoj univerzalno primjenjivog digitalnog alata za bilježenje rezultata u streljaštvu otežan je prvenstveno velikim razlikama u formatima specifičnima za nacionalne saveze jer ne postoji dovoljna potražnja koja bi opravdala resurse uložene u prilagodbu aplikacije svim disciplinama svih nacionalnih saveza.

2.2. Postojeća rješenja za bilježenje rezultata u streljaštvu

Digitalni alati za bilježenje rezultata dostupni su pretežito u obliku aplikacija za mobilne uređaje na *Google Play* i *App Store* platformama te gotovo ne postoje u obliku internetskih aplikacija, kakvog obrađuje ovaj rad. Postojeća rješenja, od kojih su dolje neka opisana, ili su previše zahtjevna za prosječnog korisnika i zahtijevaju vrlo skupu dodatnu opremu, ili nisu fleksibilna i dostupna s bilo kojeg uređaja te su razvijena za discipline specifične za neki od nacionalnih saveza, a ne za međunarodno priznate discipline. Iz tih razloga, još uvijek su najčešći načini praćenja rezultata strijelaca kroz njihovu karijeru praćenje korištenjem običnih bilježnica ili radne knjige programa *Microsoft Excel*. Niska razina digitalizacije procesa treniranja očita je posljedica niske

mogućnosti zarade na razvoju računalne podrške specijalizirane za streljaštvo, što je pak posljedica niske popularnosti sporta na svjetskoj razini.

2.2.1. Shooting Sports Cloud

Na slici 2.1. prikazano je sučelje internetske aplikacije Shooting Sports Cloud, koju je razvila tvrtka SIUS AG, prema [5], priznati međunarodni proizvođač streljačke opreme i sustava elektronskog ocjenjivanja pogodaka. SIUS AG je tržišni monopolist na području sustava elektronskog ocjenjivanja pogodaka, što i potvrđuju na svojoj internetskoj stranici: „SIUS je službeni pružatelj usluge bilježenja rezultata ISSF-a i jedini dobavljač sustava ocjenjivanja pogodaka Svjetskih prvenstava i natjecanja za olimpijske kvote. Već godinama, SIUS ima jedini sustav s odobrenjem ISSF-a za sve discipline.“ Ova internetska aplikacija je pokušaj ojačavanja monopola, budući da je za bilježenje rezultata moguće koristiti isključivo njihove najnovije sustave ocjenjivanja te nije podržan ručni unos rezultata koji se želi pohraniti u aplikaciju. U pogledu funkcionalnosti pregleda i analize zapisa rezultata, aplikacija ne nudi ništa osim unosa, pohrane, pregledavanja, i brisanja zapisa. Sve funkcionalnosti se provode na jednom zapisu odjednom, aplikacija nema mogućnost izvršavanja zahtjeva nad više zapisa istovremeno. Osim praćenja rezultata, prema [6] aplikacija nudi mogućnost organizacije i upravljanja natjecanjima te može služiti kao baza podataka članova kluba ili saveza. Prednosti aplikacije su kontinuirana dostupnost i mogućnost korištenja s mnogo različitih vrsta uređaja, dok su nedostaci ograničenost na SIUS ekosustav uređaja, nemogućnost ručnog unosa zapisa rezultata te nemogućnost statističke analize zapisa rezultata strijelca. Aplikacija je dostupna na linku: https://shootingsportscloud.com/.

Slika 2.1. ***Prikaz sučelja aplikacije Shooting Sports Cloud***

2.2.2. Shooters Diary

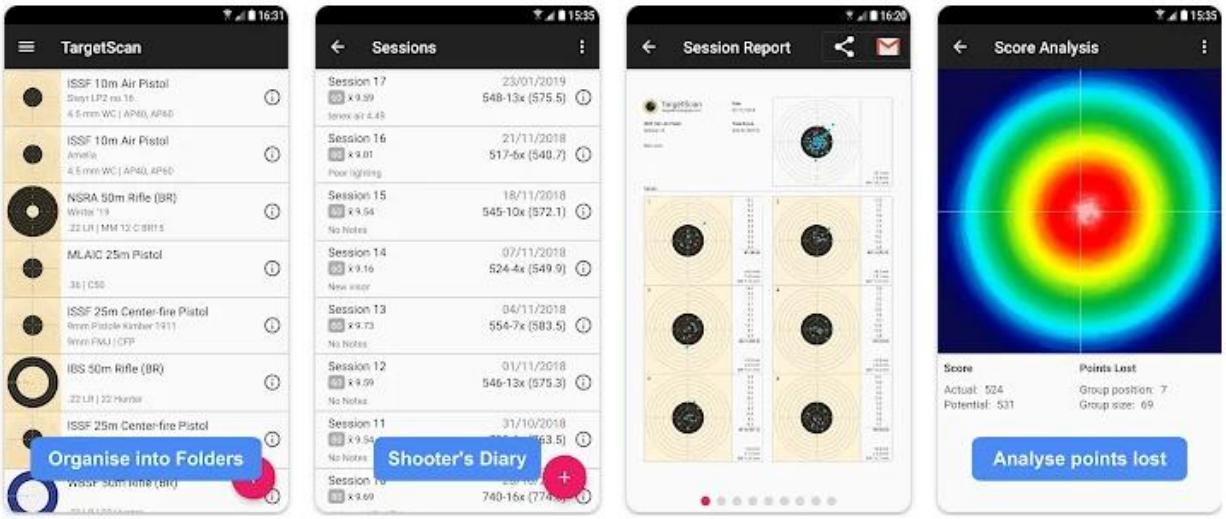
Na slici 2.2. prikazano je sučelje mobilne aplikacije Shooters Diary razvojnog inženjera KVSmartSoftware. Aplikacija je namijenjena mobilnim uređajima s Android operativnim sustavom te je dostupna na Google Play platformi, no zadnje ažuriranje je objavljeno 2019. godine, pa zastarjeli izgled aplikacije ne iznenađuje. Opis aplikacije navodi kako aplikacija podržava 18 formata natjecanja, od čega su samo 3 međunarodno priznati, dok su ostali specifični za američki

nacionalni savez. Od funkcionalnosti, aplikacija pruža mogućnost lokalne pohrane rezultata, ali i uvoz i izvoz rezultata u obliku CSV datoteka te vizualizaciju pogodaka na meti i statističku analizu zapisa rezultata. Prednosti su mogućnost vizualizacije pogodaka i analiza zapisa rezultata, dok je nedostatak, za prosječnog korisnika, nezgrapan postupak prijenosa podataka u obliku CSV datoteka.

Slika 2.2. ***Prikaz sučelja aplikacije Shooters Diary***

2.2.3. TargetScan ISSF Pistol & Rifle

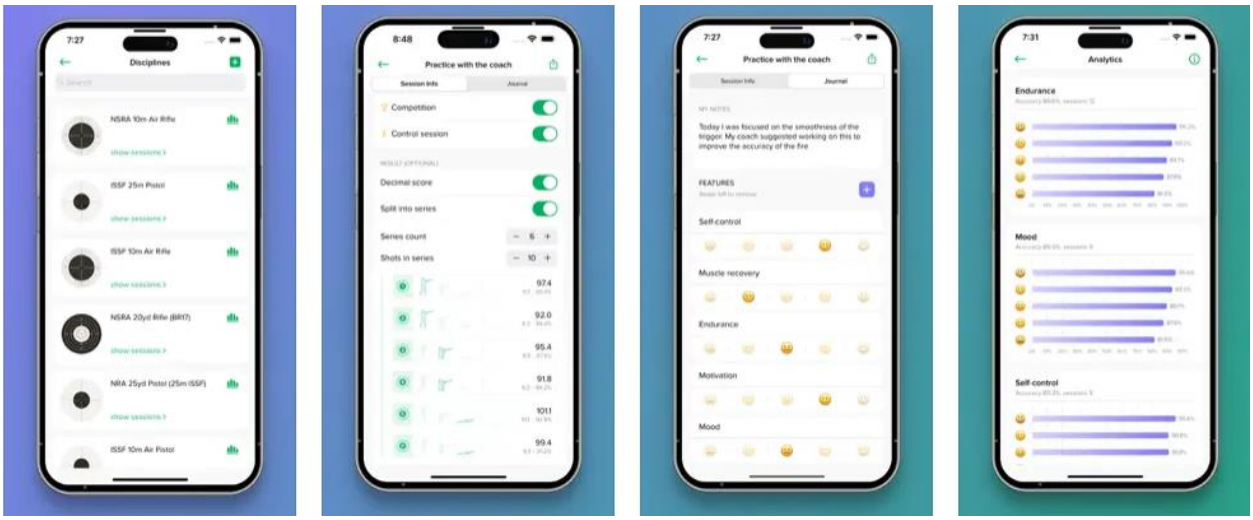
Na slici 2.3. prikazano je sučelje mobilne aplikacije TargetScan ISSF Pistol & Rifle razvojnog inženjera Deep Scoring Ltd. Ova je aplikacija također namijenjena uređajima s Android operativnim sustavom i dostupna je na Google Play platformi. Aplikacija podržava preko 180 različitih formata natjecanja te pruža mogućnost unosa zapisa rezultata računalnom obradom fotografija meta, što je glavna prednost ove aplikacije, dok je nedostatak lokalna pohrana s mogućnošću dijeljenja samo individualnih zapisa rezultata.



Slika 2.3. ***Prikaz sučelja aplikacije TargetScan ISSF Pistol & Rifle***

2.2.4. Air Shooting Diary: Pro Notes

Na slici 2.4. prikazano je sučelje mobilne aplikacije Air Shooting Diary: Pro Notes razvojnog inženjera Siarhei Maskvin. Aplikacija je namijenjena iPhone, iPad, i iPod Touch uređajima te je dostupna na App Store platformi. Opis aplikacije ne navodi koliko točno disciplina podržava, no ova je mobilna aplikacija prva na ovom popisu koja podržava pohranu zapisa rezultata na oblaku. Prednost je što osim podataka o ostvarenim rezultatima, korisnik može unijeti i podatke o svom psihičkom i fizičkom stanju tijekom treninga ili natjecanja, u obliku pisanog teksta, odgovaranja na evaluacijska pitanja, i zvučnog zapisa.



Slika 2.4. ***Prikaz sučelja aplikacije Air Shooting Diary: Pro Notes***

3. MODEL APLIKACIJE ZA BILJEŽENJE REZULTATA U STRELJAŠTVU

3.1. Funkcionalnosti aplikacije

3.1.1. Registracija korisnika

Prilikom posjete internetske stranice na kojoj je smještena aplikacija, korisniku su pružene mogućnosti registracije i otvaranja novog korisničkog računa te prijave u postojeći korisnički račun. Ako korisnik odabere opciju registracije, internetska aplikacija prikazuje obrazac za unos podataka za otvaranje novog korisničkog računa. Obavezni podaci koji se traže od novog korisnika su ime i prezime, datum rođenja, spol, broj telefona, adresa elektroničke pošte, koja će dalje u programskom rješenju biti korištena kao jedinstveni identifikator korisnika, i lozinka, koja mora biti dugačka najmanje 10 znakova i mora sadržavati barem jedno malo slovo, jedno veliko slovo, jednu znamenku, i jedan specijalni znak. Budući da se adresa elektroničke pošte koristi kao jedinstveni identifikator, nije dopušteno otvaranje više od jednog korisničkog računa s istom adresom. Iz tog razloga, u slučaju da adresa elektroničke pošte već postoji u bazi podataka aplikacije, aplikacija prikazuje odgovarajuću poruku s objašnjenjem pogreške koja se dogodila. Aplikacija također prikazuje poruku s pogreškom u slučaju da lozinka ne ispunjava sva propisana pravila.

3.1.2. Prijava korisnika

Kada korisnik na početnoj stranici internetske aplikacije odabere mogućnost prijave u postojeći korisnički račun, internetska aplikacija prikazuje obrazac za unos podataka za prijavu u postojeći korisnički račun. Podaci koji se traže od korisnika su adresa elektroničke pošte i lozinka. Iz sigurnosnih razloga, ne postoji mogućnost da korisnik ostane prijavljen na uređaju dok se internetskom pregledniku ne obrišu kolačići, već sve prijave vrijede određeni vremenski interval postavljen u samom programskom rješenju. U slučaju da je korisnik pokušao izvršiti prijavu, ali su unesena adresa ili lozinka netočni, aplikacija prikazuje odgovarajuću poruku s pogreškom. U slučaju da ne postoji korisnički račun s navedenom adresom, aplikacija također prikazuje odgovarajuću pogrešku.

3.1.3. Uređivanje korisničkih podataka

Kada je korisnik prijavljen u aplikaciju, ima mogućnost unutar sučelja aplikacije odabrati opciju uređivanja podataka svog korisničkog računa. Sve podatke koje je moguće unijeti prilikom registracije korisnika, koja je opisana u potpoglavlju 3.1.1, osim lozinke, moguće je unijeti, urediti

ili izmijeniti prilikom uređivanja korisničkih podataka. U slučaju neke od pogreški koje su spomenute u potpoglavlju 3.1.1, aplikacija prikazuje odgovarajuće poruke s pogreškama.

3.1.4. Brisanje korisničkog računa

Prijavljeni korisnik unutar sučelja aplikacije može odabrati opciju brisanja svog korisničkog računa i svih svojih podataka iz baze podataka aplikacije. Brisanje podataka se događa bespovratno, zbog čega je potrebna potvrda unosom trenutne adrese elektroničke pošte korisničkog računa i trenutne lozinke korisničkog računa. U slučaju neke od mogućih pogreški, aplikacija prikazuje odgovarajuće poruke s pogreškama.

3.1.5. Odjava korisnika

Prijavljeni korisnik unutar sučelja aplikacije ima opciju odjave iz aplikacije. U slučaju odabira odjave, aplikacija uklanja autorizacijski token iz kolačića internetskog preglednika i korisniku prikazuje početnu stranicu.

3.1.6. Unos novog zapisa

Prijavljeni korisnik unutar sučelja aplikacije može unositi zapise o rezultatima. Prilikom unosa novog zapisa, aplikacija prikazuje obrazac u kojemu se kao obavezni podaci unose disciplina streljačkog događaja, datum ostvarenog rezultata, vrijeme (sati i minute) početka, vrijeme kraja, ime streljane na kojoj je ostvaren rezultat, mjesto i država gdje se streljana nalazi, te podaci o pojedinim pogodcima. Aplikacija prikazom poruke s pogreškom upozorava korisnika u slučaju da neki od obaveznih podataka nedostaje ili je unesena nedopuštena vrijednost. Neobavezni podaci o gađanju koji se mogu unijeti su prosječna temperatura zraka, tlak zraka, brzina vjetra, smjer vjetra, te tekstualne bilješke o vremenskim uvjetima i ostalim uvjetima na streljani, bilješke o korekcijama izvršenima na oružju, i bilješke o psihofizičkom stanju tijekom gađanja. Postoji mogućnost odustajanja od unosa novog zapisa, pri čemu se gube sve do tada u obrazac upisane vrijednosti.

3.1.7. Pregled postojećih zapisa

Prijavljeni korisnik unutar sučelja aplikacije ima mogućnost pregleda svih zapisa svojih rezultata. Aplikacija u tu svrhu prikazuje popis svih zapisa s osnovnim informacijama svakog zapisa: kratica discipline, datum ostvarenog rezultata, vremena početka i kraja gađanja, i naziv streljane. Popis je poredan prema datumima zapisa, od najnovijeg do najstarijeg zapisa. Pritiskom na tipku pored podataka o zapisu, otvara se detaljni prikaz svih podataka zapisa na kojemu se vide svi podaci koje je korisnik unio.

3.1.8. Izmjena postojećeg zapisa

Kada je prijavljeni korisnik otvorio detaljni prikaz podataka nekog zapisa, ima mogućnost uređivanja podataka tog zapisa. Aplikacija dopušta unos, uređivanje, i izmjenu svih podataka koji se mogu pohraniti za zapis rezultata, uz nužne provjere opisane u potpoglavlju 3.1.6. U slučaju da neki od nužnih uvjeta nije ispunjen, aplikacija prikazuje odgovarajuću poruku s pogreškom i ne dopušta pohranu izmijenjenih podataka. Korisnik ima mogućnost u svakom trenutku odustati od pohrane izmijenjenih podataka pritiskom na odgovarajuću tipku, pri čemu aplikacija ponovno prikazuje sve korisnikove zapise.

3.1.9. Brisanje postojećeg zapisa

Prijavljeni korisnik ima mogućnost brisanja zapisa o rezultatima iz baze podataka aplikacije. Kada se korisnik nalazi na prikazu zapisa u obliku popisa, pritiskom na tipku pored podataka o zapisu može odabrati opciju brisanja zapisa. Također, kada se korisnik nalazi na detaljnom prikazu jednog zapisa, pritiskom na odgovarajuću tipku može odabrati da se taj zapis ukloni iz baze podataka. Nakon uspješnog uklanjanja, aplikacija učitava prikaz svih korisnikovih zapisa.

3.2. Podržane discipline i pripadajući načini bodovanja

Prilikom izrade aplikacije, u obzir su uzete samo discipline u kojima se sportaši mogu natjecati na Ljetnim Olimpijskim igrama 2024. godine te su njihovi načini bodovanja opisani u slijedećim potpoglavljima.

3.2.1. 10 m zračni pištolj

Prema [7, str. 441], u disciplini 10 m zračni pištolj svaki natjecatelj, s udaljenosti od 10 metara od mete, ispaljuje 6 serija po 10 hitaca, sveukupno 60 hitaca. Svaki pogodak se boduje cjelobrojnom vrijednosti između 0 i 10, uključujući granice, te se broji i koliko je ostvareno muševa (eng. *bullseye*), odnosno pogodaka decimalne vrijednosti 10.4 ili veće. U slučaju izjednačenja po bodovima skupljenima u 6 serija, natjecatelji se rangiraju prema broju ostvarenih muševa.

3.2.2. 10 m zračna puška

Prema [8, str. 405], u disciplini 10 m zračna puška svaki natjecatelj, također s udaljenosti od 10 metara od mete, ispaljuje 6 serija po 10 hitaca, sveukupno 60 hitaca. Svaki se pogodak boduje decimalnom vrijednosti između 0.0 i 10.9, uključujući granice, te se također broji i koliko je ostvareno muševa.

3.2.3. 25 m pištolj

Prema [7, str. 426–427, 441], u disciplini 25 m pištolj svaki natjecatelj, s udaljenosti od 25 metara od mete, ispaljuje 2 faze od 6 serija po 5 hitaca, odnosno 2 faze od 30 hitaca, odnosno sveukupno 60 hitaca.. Prva faza, odnosno prvih 30 hitaca, ispaljuje se u metu za precizno gađanje te se pogodci boduju cjelobrojnim vrijednostima između 0 i 10, uključujući granice, te se također broji i koliko je ostvareno muševa. Druga faza, odnosno drugih 30 hitaca, ispaljuje se u metu za brzo gađanje, te se pogodci boduju s 0 ako su izvan mete ili ako su ispaljeni nakon isteka vremena, odnosno cjelobrojnim vrijednostima između 5 i 10, uključujući granice, ako su ispaljeni unutar vremena i ako su unutar mete. Tijekom brzog gađanja se također broji koliko je ostvareno muševa.

3.2.4. 25 m pištolj brzo gađanje

Prema [7, str. 424–425, 441], u disciplini 25 m pištolj brzo gađanje svaki natjecatelj, također s udaljenosti od 25 metara od mete, ispaljuje 2 faze od 6 serija po 5 hitaca, odnosno 2 faze od 30 hitaca, odnosno sveukupno 60 hitaca. Svaki pogodak boduje se s 0 ako je izvan mete ili ako je ispaljen nakon isteka vremena, odnosno cjelobrojnom vrijednošću između 5 i 10, uključujući granice, ako je ispaljen unutar vremena i ako je unutar mete, te se broji i koliko je ostvareno muševa.

3.2.5. 50 m puška trostav

Prema [8, str. 402–405], u disciplini 50 m puška trostav svaki natjecatelj, s udaljenosti od 50 metara od mete, ispaljuje 3 faze od 4 serija po 10 hitaca, odnosno 3 faze od 40 hitaca, odnosno sveukupno 120 hitaca. Prva faza ispaljuje se iz klečećeg stava, druga iz ležećeg, te posljednja iz stojećeg stava. Neovisno o stavovima, sve se serije jednako boduju. Svaki pogodak boduje se decimalnom vrijednosti između 0.0 i 10.9, uključujući granice, te se broji i koliko je ostvareno muševa.

3.2.6. Trap i skeet

Trap i skeet discipline su discipline sportskog streljaštva u kojima se natjecatelji natječu u gađanju glinenih golubova ispaljivanjem iz sačmarica, s udaljenosti od 70 metara. Prema [9, str. 407–411], u disciplini trap svaki natjecatelj gađa 5 rundi od 25 glinenih golubova, odnosno sveukupno 125 glinenih golubova, te ima pravo ispaliti dva hica u svaki glineni golub u pokušaju da ga pogodi, zbog čega će se u ovoj disciplini brojati glineni golubovi, a ne hici. Svaki se pokušaj, odnosno glineni golub, ocjenjuje kao pogođeni (oznaka 1) ili promašeni (oznaka 0), bez brojčanih vrijednosti. Rezultat jedne runde je broj pogođenih glinenih golubova u toj rundi, a sveukupni

rezultat je zbroj rezultata svih rundi. Prema [9, str. 416–424], u disciplini skeet svaki natjecatelj također gađa 5 rundi od 25 glinenih golubova, odnosno sveukupno 125 glinenih golubova, bez prava ispaljivanja dva hica u svaki glineni golub, već samo jedan. Način bodovanja jednak je disciplini trap.

4. PROGRAMSKO RJEŠENJE ZA BILJEŽENJE REZULTATA U STRELJAŠTVU

U ovom poglavlju opisani su korišteni alati i tehnologije te je objašnjeno programsko rješenje na strani poslužitelja, koje se sastoji od baze podataka i aplikacijskog programskog sučelja, i programsko rješenje na strani korisnika, koje je sâma aplikacija za bilježenje rezultata.

4.1. Korišteni alati i tehnologije

Rješenja opisana u ovom radu u potpunosti su izrađena u razvojnom okruženju Visual Studio koristeći ASP.NET Core razvojni okvir, a pokrenuta su na CentOS poslužitelju koristeći Docker platformu kroz bash ljusku. Baza podataka nalazi se na istom poslužitelju u instanci Microsoft SQL Server sustava. Za izradu aplikacijskog programskog sučelja korišten je isključivo programski jezik C#, dok su za izradu aplikacije korišteni i jezici HTML, CSS, i JavaScript.

4.1.1. Visual Studio 2022 Preview

Prema [10], Visual Studio 2022 Preview je integrirano razvojno okruženje preporučeno za razvoj programskih rješenja .NET platforme. Može se koristiti za pisanje i uređivanje programskog kôda, izradu izvršnih datoteka, testiranje programskih rješenja, otklanjanje pogrešaka, i objavljivanje programskog kôda i gotovog rješenja. Nudi podršku za analizu programskog kôda i prepoznavanje pogrešaka u svim jezicima koji su korišteni za izradu projekta: C#, HTML, CSS, i JavaScript, ali podržava i mnoge druge.

4.1.2. Microsoft SQL Server 2022 Developer

Prema [11], Microsoft SQL Server 2022 Developer je sustav za upravljanje relacijskim bazama podataka. Developer izdanje uključuje sve funkcionalnosti Enterprise izdanja, ali omogućuje isključivo razvoj i testiranje aplikacija, bez komercijalne uporabe Developer izdanja. Ovakav sustav je dovoljan za potrebe rada.

4.1.3. Docker

Prema [12], Docker je platforma korištena za testiranje izrađenih aplikacija. Aplikacije su „upakirane“ u jedinice zvane „Docker kontejneri“ (eng. *Docker containers*) koje sadrže sve potrebno aplikaciji za ispravan rad, poput biblioteka. Platforma je korištena za testiranje rada aplikacije za bilježenje rezultata u streljaštvu, i za testiranje rada aplikacijskog programskog sučelja.

4.1.4. ASP.NET Core

Prema [13], ASP.NET Core je više-platformski razvojni okvir otvorenog izvora. Osim za izradu internetskih aplikacija i aplikacijskih programskih sučelja, može se koristiti i za izradu IoT (eng. *Internet of Things*) aplikacija te pozadinskih procesa mobilnih aplikacija. Za pisanje programskog kôda mogu se koristiti mnogi objektno-orijentirani jezici, neki od najčešćih su C#, C++, Visual Basic, i F# jezici.

4.1.5. Entity Framework Core

Prema [14], Entity Framework (EF) Core je lagana, proširiva, i višeplatformska inačica otvorenog izvora tehnologije pristupa podacima Entity Framework. EF Core se može koristiti kao *object-relational mapper* u projektima izrađenima koristeći .NET razvojni okvir, odnosno kao alat koji preslikava klase definirane u C# programskom jeziku u entitete u relacijskoj bazi podataka.

4.1.6. Programski jezik C#

Prema [15], C# je moderan visoko-razinski objektno-orijentirani programski jezik iz C-obitelji jezika, zbog čega je sličan C i C++ jezicima, ali je sličan i Java i JavaScript jezicima. Trenutno je najčešće korišteni jezik pri izradi programskih rješenja baziranih na .NET razvojnom okviru. Neke od prednosti jezika C# su lambda funkcije, rukovanje iznimkama, LINQ (eng. *Language INtegrated Query*) upiti, asinkrone operacije, i standardizirani sustav tipova varijabli.

4.1.7. HTML

Prema [16], HTML je skraćenica za HyperText Markup Language. HTML je jezik koji definira značenje i strukturu internetskog sadržaja, ali se uz njega uobičajeno koriste i jezici CSS i JavaScript. Riječ „hypertext“ u punom nazivu jezika se odnosi na poveznice koje međusobno povezuju internetske stranice.

4.1.8. CSS

Prema [17], CSS je skraćenica za Cascading Style Sheets. CSS je stylesheet jezik te se koristi za opisivanje izgleda elemenata dokumenata napisanih u HTML i XML jezicima. Jezik opisuje kako se elementi trebaju prikazati na ekranu, papiru, u govoru, ili u drugim medijima. CSS je jedan od najvažnijih jezika otvorenog Interneta i standardiziran je prema W3C (eng. *World Wide Web Consortium*) specifikacijama.

4.1.9. JavaScript

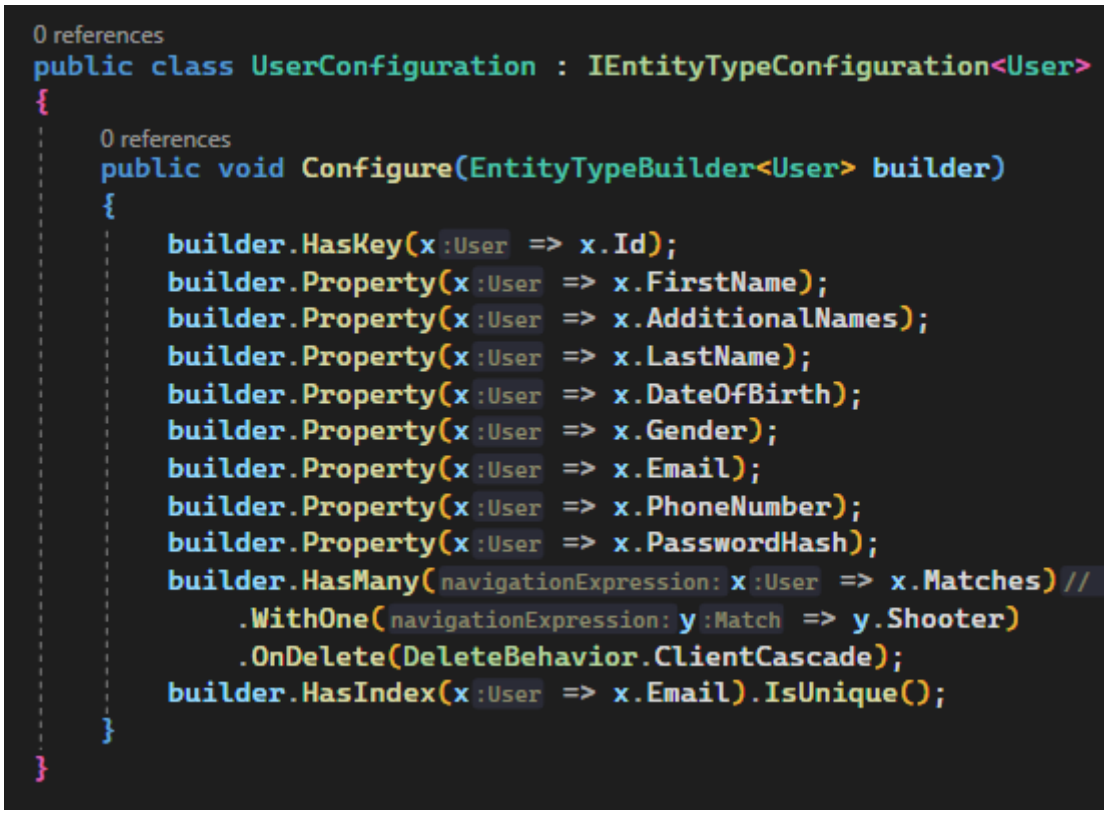
Prema [18], JavaScript je lagani prevođeni ili just-in-time prevođeni programski jezik najčešće korišten kao skriptni jezik internetskih stranica. Dinamički je jezik koji se izvodi na jednoj procesorskoj niti te podržava objektno-orijentirani, imperativni, i deklarativni stil pisanja programskog kôda.

4.2. Poslužiteljsko programsko rješenje

Programsko rješenje na strani poslužitelja sastoji se od baze podataka i aplikacijskog programskog sučelja. Aplikacijsko programsko sučelje izrađeno je radi odvajanja dužnosti obrade podataka i upravljanja pohranom podataka na zasebno programsko rješenje te radi osiguravanja mogućnosti razvoja aplikacija za razne platforme, operativne sustave, i uređaje bez potrebe za ponovnim pisanjem programskog kôda za interakciju s bazom podataka.

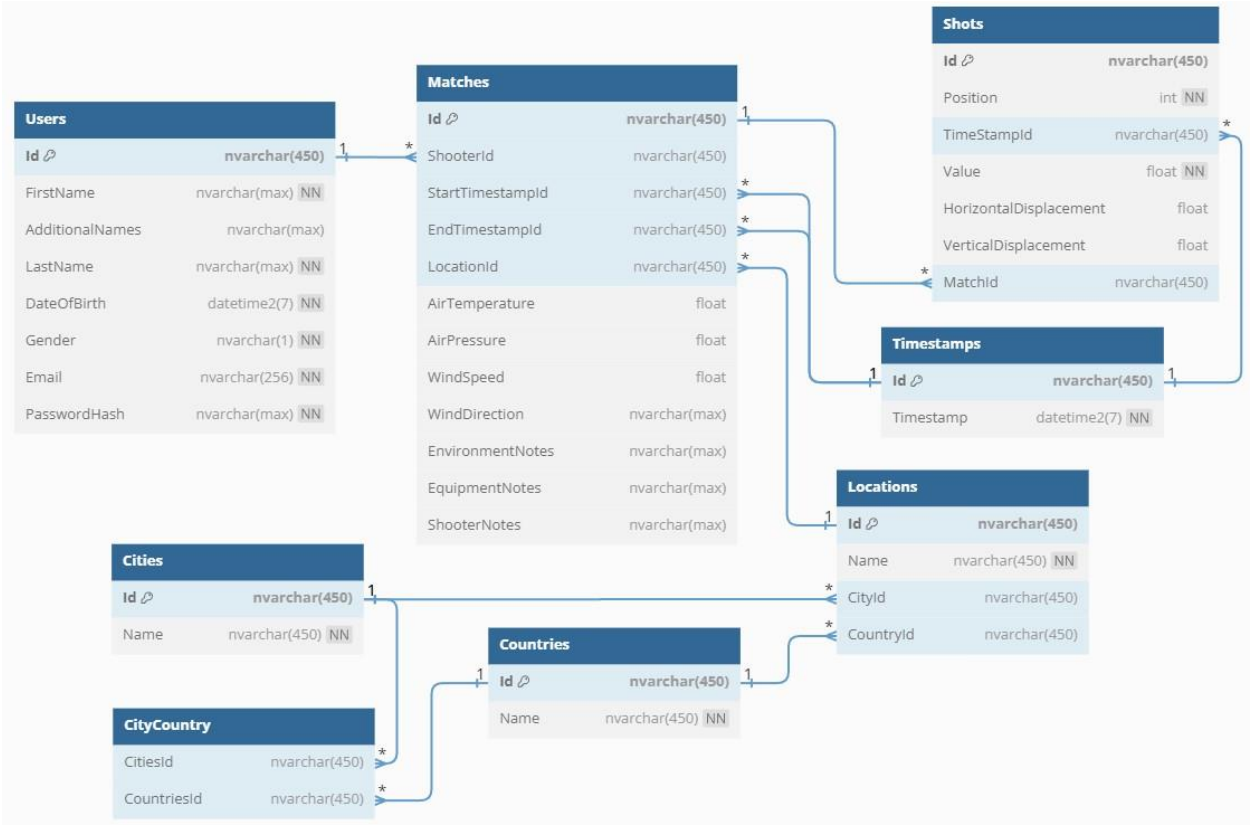
4.2.1. Baza podataka

Baza podataka u koju se pohranjuju svi podaci koji su potrebni aplikacijskom programskom sučelju, i u konačnici aplikaciji za bilježenje rezultata, opisana je programskim jezikom C# unutar jednog od četiriju projekata koji zajedno čine aplikacijsko programsko sučelje. Kao prvi korak definiranja baze podataka, svi podaci koji su potrebni za ispravan rad aplikacije podijeljeni su u sljedeće entitete: korisnik, zapis, pogodak, vremenska oznaka, lokacija, grad, i država. Zatim je svaki od entiteta zapisan u obliku klase te je koristeći Entity Framework Core za svaku od klasa definiran način zapisivanja podataka instance klase, odnosno objekta, u bazu podataka. U ovom koraku određuje se koji se podaci pohranjuju u bazu podataka, koji od tih podataka smije biti pohranjen kao prazna vrijednost, te koji od tih podataka je relacijski povezan s drugim entitetom. Primjer definiranog načina zapisivanja podataka objekta u bazu podataka koristeći metode dane tehnologijom Entity Framework Core nalazi se na slici 4.1.



Slika 4.1. ***Prikaz definiranja načina zapisivanja podataka o korisniku***

Kada su za sve entitete, odnosno klase, definirani načini zapisivanja podataka u bazu podataka, iz razvojnog okruženja Visual Studio koristeći konzolu NuGet upravitelja paketima pokreće se izrada baze podataka na instanci Microsoft SQL Server sustava prema danim uputama. Baza podataka koja je nastala primjenom uputa zapisanih u programskom kôdu aplikacijskog programskog sučelja prikazana je relacijskim modelom na slici 4.2.



Slika 4.2. ***Relacijski model nastale baze podataka***

4.2.2. Aplikacijsko programsko sučelje

Izrađeno aplikacijsko programsko sučelje ima dvije važne zadaće: upravljanje pohranom i dohvaćanjem podataka i upravljanje pravima pristupa podacima, što odražavaju i upravljači koje sučelje sadrži: jedan upravljač koji izlaže metode upravljanja podacima i jedan upravljač koji izlaže metode autentifikacije korisnika. Kako je već objašnjeno, upravljanje podacima, odnosno bazom podataka, provodi se koristeći metode dane tehnologijom Entity Framework Core, a upravljanje pravima pristupa podacima provodi se metodom Bearer autentifikacije koristeći JWT (eng. *JSON Web Token*) tokene. Svaki HTTP (eng. *Hyper Text Transfer Protocol*) zahtjev prema upravljaču podacima mora u zaglavlju sadržavati valjani JWT token koji identificira korisnika, odnosno podnositelja HTTP zahtjeva. Sučelje izdaje JWT tokene isključivo prilikom autorizacije prethodno registriranih korisnika, koja je prikazana programskim kôdom na slici 4.3.

```
[HttpPost, Route(template: "login"),
ProducesResponseType(statusCode: StatusCodes.Status404NotFound),
ProducesResponseType(statusCode: StatusCodes.Status401Unauthorized),
ProducesResponseType(statusCode: StatusCodes.Status200OK),
ProducesResponseType(statusCode: StatusCodes.Status400BadRequest)]
0 references
public async Task<IActionResult> Login([FromBody] LoginRequestDto loginRequest)
{
    ModelState.ClearValidationState(key: nameof(loginRequest));

    if (TryValidateModel(loginRequest, prefix: nameof(loginRequest)))
    {
        User? user = await _userService.GetByEmailAsync(loginRequest.Email);

        if (user is null)
            return NotFound(loginRequest.Email);

        if (!user.PasswordHash.Equals(loginRequest.PasswordHash))
            return Unauthorized();

        return Ok(await _userService.Login(user));
    }

    return ValidationProblem(ModelState);
}
```

Slika 4.3. ***Prikaz programskog kôda autorizacije korisnika na upravljaču sučelja***

U slučaju slanja HTTP zahtjeva prema upravljaču podacima bez tokena u zaglavlju, ili s nevažećim tokenom u zaglavlju, sučelje bez daljnjih provjera odgovara na zahtjev s HTTP statusom 401, koji poručuje klijentu da nije autoriziran i nema ovlasti slati zahtjeve na upravljač.

4.3. Korisničko programsko rješenje

Programsko rješenje na strani korisnika je sâma aplikacija za bilježenje rezultata streljaštva. Pri izradi aplikacije, razmotrene su potrebe potencijalnih korisnika te njihove prosječne računalne vještine. Aplikacija je izrađena s primarnim ciljevima jednostavnosti i jasnoće uporabe, što se odražava u minimalističkom vizualnom identitetu i u jasnim povratnim porukama o događajima i greškama.

4.3.1. Autentifikacija korisnika

Prvi korak u uporabi aplikacije, nakon učitane početne stranice, je registracija, odnosno otvaranje korisničkog računa. Registracija se provodi uz zahtjeve prethodno opisane u potpoglavlju 3.1.1. Nakon što aplikacija utvrdi da svi uneseni podaci poštuju navedene zahtjeve, prosljeđuje ih

aplikacijskom programskom sučelju na obradu te se zatim čeka povratna informacija. Ako je povratna informacija sučelja pozitivna, odnosno otvoren je novi korisnički račun, aplikacija prikazuje poruku potvrde i preusmjerava korisnika na stranicu za prijavu, odnosno autorizaciju. U slučaju da je povratna informacija sučelja negativna, aplikacija prikazuje poruku o grešci i predlaže korisniku provjeru podataka i ponovni pokušaj. Isječak programskog kôda metode registracije koji priprema podatke sadržaja zahtjeva, šalje zahtjev te obrađuje povratnu informaciju vidljiv je na slici 4.4.

```
UserRequest.FirstName = Request.Form["UserRequest.FirstName"].ToString();
UserRequest.AdditionalNames = Request.Form["UserRequest.AdditionalNames"].ToString();
UserRequest.LastName = Request.Form["UserRequest.LastName"].ToString();
UserRequest.DateOfBirth = DateTime.Parse(Request.Form["UserRequest.DateOfBirth"].ToString());
UserRequest.Gender = Request.Form["UserRequest.Gender"].ToString()[0];
UserRequest.PhoneNumber = Request.Form["UserRequest.PhoneNumber"].ToString();
UserRequest.Email = Request.Form["UserRequest.Email"].ToString();
Password = Request.Form["Password"].ToString();

ModelState.ClearValidationState(key: nameof(Register));
if (!TryValidateModel(nameof(Register)))...

var request = new HttpRequestMessage(HttpMethod.Post,
    requestUri: "http://172.19.0.3:80/artemis/auth/register");
var client = _httpClientFactory.CreateClient();

RegistrationRequestDto registrationRequest = new(UserRequest)...;

var json :string = JsonConvert.SerializeObject(registrationRequest);
var content = new StringContent(json, Encoding.UTF8, mediaType: "application/json");
request.Content = content;
HttpResponseMessage response = await client.SendAsync(request);

if (response.StatusCode.Equals(obj: HttpStatusCode.Conflict))...

if (response.StatusCode.Equals(obj: HttpStatusCode.Created))...
```

Slika 4.4. ***Prikaz isječka programskog kôda metode registracije novog korisnika***

Kada korisnički račun već postoji u bazi podataka, korisnik se mora autorizirati kako bi pristupio svojim podacima. Autorizacija, odnosno prijava, druga je mogućnost dostupna na početnoj stranici aplikacije. Prijava se provodi unosom adrese elektroničke pošte i lozinke korisničkog računa u aplikaciju. Ako aplikacija utvrdi da podaci odgovaraju očekivanim oblicima, prosljeđuje ih HTTP zahtjevom sučelju na obradu te očekuje povratnu informaciju. U slučaju da je povratna informacija pozitivna, aplikacija iz sadržaja HTTP odgovora izdvaja JWT token i u obliku kolačića ga posprema u predmemoriju internetskog preglednika te korisnika preusmjerava na pregled postojećih zapisa. Ako povratna informacija nije pozitivna, odnosno sadrži informaciju o pogrešci,

aplikacija prikazuje odgovarajuću poruku korisniku i upućuje ga na provjeru podataka prije ponovnog pokušaja autorizacije. Isječak programskog kôda metode autorizacije koji priprema podatke sadržaja zahtjeva, šalje zahtjev te obrađuje povratnu informaciju vidljiv je na slici 4.5.

```
email = Request.Form["email"].ToString();
password = Request.Form["password"].ToString();

ModelState.ClearValidationState(key: nameof(Login));
if (!TryValidateModel(nameof(Login)))...

var request = new HttpRequestMessage(HttpMethod.Post,
    requestUri: "http://172.19.0.3:80/artemis/auth/login");
var client = _httpClientFactory.CreateClient();

LoginRequestDto loginRequest = new()...;

var json:string = JsonConvert.SerializeObject(loginRequest);
var content = new StringContent(json, Encoding.UTF8, mediaType: "application/json");
request.Content = content;
HttpResponseMessage response = await client.SendAsync(request);

if (response.StatusCode.Equals(obj: HttpStatusCode.OK))...

if (response.StatusCode.Equals(obj: HttpStatusCode.Unauthorized))...

if (response.StatusCode.Equals(obj: HttpStatusCode.NotFound))...
```

Slika 4.5. ***Prikaz isječka programskog kôda metode prijave postojećeg korisnika***

4.3.2. Uređivanje korisničkih podataka

Kako je predviđeno u potpoglavlju 3.1.3., autorizirani korisnik ima mogućnost u bilo kojem trenutku unutar aplikacije urediti podatke svog korisničkog računa. Svi podaci uneseni prilikom registracije, osim lozinke, mogu biti uređeni ili u potpunosti izmijenjeni, no moraju poštivati pretpostavljene zahtjeve. Ako korisnik zatraži izmjenu podataka i aplikacija utvrdi da nove vrijednosti ispunjavaju zahtjeve, prosljeđuje ih sučelju unutar sadržaja HTTP zahtjeva, dok unutar zaglavlja zahtjeva dodaje JWT autorizacijski token. Ako odgovor sučelja poručuje da su podaci uspješno izmijenjeni, aplikacija prikazuje poruku potvrde i preusmjerava korisnika na prikaz postojećih zapisa. U slučaju da je došlo do problema prilikom pokušaja izmjene podataka, aplikacija prikazuje odgovarajuće poruke o pogrešci. Isječak programskog kôda metode izmjene korisničkih podataka vidljiv je na slici 4.6.

```
UserRequest.Id = response.Content.ReadFromJsonAsync<UserDto>().Result!.Id;
UserRequest.FirstName = Request.Form["UserRequest.FirstName"].ToString();
UserRequest.AdditionalNames = Request.Form["UserRequest.AdditionalNames"].ToString();
UserRequest.LastName = Request.Form["UserRequest.LastName"].ToString();
UserRequest.DateOfBirth = DateTime.Parse(Request.Form["UserRequest.DateOfBirth"].ToString());
UserRequest.Gender = Request.Form["UserRequest.Gender"].ToString()[0];
UserRequest.PhoneNumber = Request.Form["UserRequest.PhoneNumber"].ToString();
UserRequest.Email = Request.Form["UserRequest.Email"].ToString();

request = new HttpRequestMessage(HttpMethod.Post,
    requestUri: "http://172.19.0.3:80/artemis/auth/update");
UserUpdateRequestDto updateRequest = new(UserRequest);

var json :string = JsonConvert.SerializeObject(updateRequest);
var content = new StringContent(json, Encoding.UTF8, mediaType: "application/json");
request.Content = content;
request.Headers.Authorization = new AuthenticationHeaderValue(scheme: "Bearer", parameter: bearerToken);
response = await client.SendAsync(request);

if (response.StatusCode.Equals(obj: HttpStatusCode.Conflict))...

if (response.StatusCode.Equals(obj: HttpStatusCode.OK))...

if (response.StatusCode.Equals(obj: HttpStatusCode.NotFound))...

if (response.StatusCode.Equals(obj: HttpStatusCode.Unauthorized))...
```

Slika 4.6. ***Prikaz isječka programskog kôda metode uređivanja podataka postojećeg korisnika***

4.3.3. Brisanje korisničkog računa

Svaki autorizirani korisnik kroz sučelje aplikacije ima pravo u bilo kojem trenutku i obrisati svoj korisnički račun. Brisanjem korisničkog računa aplikacijsko programsko sučelje pokreće i postupak brisanja svih zapisa povezanih za taj korisnički račun, stoga je ovaj postupak vrlo destruktivan. Iz tog razloga, iako je korisnik već autoriziran, kako bi mogao obrisati svoj korisnički račun mora ponovno unijeti adresu elektroničke pošte povezanu s računom i lozinku korisničkog računa. Nakon što aplikacija utvrdi da je korisnik autoriziran te da uneseni podaci odgovaraju očekivanim oblicima podataka, prosljeđuje unesene podatke sučelju u sadržaju HTTP zahtjeva te JWT autorizacijski token prosljeđuje unutar zaglavlja zahtjeva. Ako sučelje potvrdi da je korisnički račun uspješno obrisan, aplikacija prikazuje poruku potvrde, iz kolačića internetskog preglednika uklanja autorizacijski token, te klijenta preusmjerava na početnu stranicu aplikacije. U slučaju da se dogodila pogreška ili su unesene vrijednosti netočne, aplikacija prikazuje odgovarajuće povratne informacije. Isječak programskog kôda metode odgovorne za brisanje korisničkog računa prikazan je na slici 4.7.

```
if (!Request.Cookies.TryGetValue("Bearer", out string? bearerToken))...

var request = new HttpRequestMessage(HttpMethod.Get,
    requestUri: "http://172.19.0.3:80/artemis/auth/get-user/by-id");
var client = _httpClientFactory.CreateClient();
var header = new AuthenticationHeaderValue(scheme: "Bearer", parameter: bearerToken);
request.Headers.Authorization = header;
HttpResponseMessage response = await client.SendAsync(request,
    HttpCompletionOption.ResponseHeadersRead); // Task<HttpResponseMessage>

if (response.StatusCode.Equals(obj: HttpStatusCode.Unauthorized))...

if (!response.StatusCode.Equals(obj: HttpStatusCode.OK))...

UserDeleteRequest.Id = response.Content.ReadFromJsonAsync<UserDto>().Result!.Id;
UserDeleteRequest.Email = Request.Form["UserDeleteRequest.Email"].ToString();
Password = Request.Form["Password"].ToString();

UserDeleteRequest.PasswordHash = Convert.ToHexString(inArray: SHA512.HashData(
    source: Encoding.Default.GetBytes(Password))).ToLower(); // string

request = new HttpRequestMessage(HttpMethod.Delete,
    requestUri: "http://172.19.0.3:80/artemis/auth/delete");
request.Headers.Authorization = header;
response = await client.SendAsync(request);

if (response.StatusCode.Equals(obj: HttpStatusCode.Conflict))...

if (response.StatusCode.Equals(obj: HttpStatusCode.OK))...
```

Slika 4.7. ***Prikaz isječka programskog kôda metode brisanja korisničkog računa***

4.3.4. Odjava korisnika

Autorizirani korisnik može u bilo kojem trenutku odabrati opciju odjave iz aplikacije. U tom slučaju, aplikacija uklanja autorizacijski token iz predmemorije internetskog preglednika te klijenta preusmjerava na početnu stranicu aplikacije. Programski kôd metode odjave korisnika u cijelosti je prikazan na slici 4.8.

```
5 references
public class Logout : PageModel
{
    0 references
    public RedirectResult OnGet()
    {
        Response.Cookies.Delete(key: "Bearer");
        return Redirect(url: "/Index");
    }
}
```

Slika 4.8. ***Prikaz programskog kôda metode odjave korisnika***

4.3.5. Unos novog zapisa

Svrha aplikacije je digitalna pohrana rezultata u streljaštvu, stoga mora postojati i način unosa tih zapisa. Opis funkcionalnosti unosa novog zapisa dan je potpoglavljem 3.1.6., a opisi disciplina i načini bodovanja dani su potpoglavljem 3.2. Kada autorizirani korisnik potvrdi završetak popunjavanja podataka zapisa o rezultatu i zatraži pohranu u bazu podataka, aplikacija prvo provjerava jesu li ispunjeni svi zahtjevi. Osim što se provjerava jesu li uneseni svi obavezni podaci, provjerava se i odgovaraju li vrijednosti pogodaka mogućim vrijednostima prema pravilima disciplina i načinima bodovanja, npr. ako korisnik odabere unos trap ili skeet zapisa, provjerava se nalaze li se sve vrijednosti pogodaka u skupu dopuštenih vrijednosti {0, 1}. Ako aplikacija utvrdi da su svi zahtjevi ispunjeni, aplikacija prema unesenim imenima države, grada, i lokacije od aplikacijskog programskog sučelja zahtjeva entitet odgovarajuće lokacije, te od sučelja prema unesenim vremenima početka i kraja zapisanog događaja zahtjeva entitete vremenskih oznaka. Iz entiteta aplikacija izdvaja jedinstvene identifikatore, odnosno primarne ključeve, te ih ugrađuje u sadržaj HTTP zahtjeva zajedno s ostalim podacima o novom zapisu, dok u zaglavlje zapisuje JWT autorizacijski token. Nakon slanja zahtjeva aplikacija iščekuje odgovor sučelja te ako je odgovor pozitivan, odnosno poručuje da je zapis uspješno unesen u bazu podataka, prikazuje poruku potvrde unosa i korisnika preusmjerava na pregled svih postojećih zapisa. U slučaju pogrešaka, bilo u unesenim vrijednostima ili u komunikaciji sa sučeljem, aplikacija prikazuje odgovarajuće poruke greške i iščekuje korisničku akciju. Isječak programskog kôda koji prikazuje osnovnu logiku metode unosa novog zapisa nalazi se na slici 4.9., a potpuni programski kôd dostupan je u prilogu 3. budući da se cijela metoda sastoji od preko 200 linija programskog kôda.

```
var request = new HttpRequestMessage(HttpMethod.Get,
    requestUri: "http://172.19.0.3:80/artemis/auth/get-user/by-id");
var client = _httpClientFactory.CreateClient();
var header = new AuthenticationHeaderValue(scheme: "Bearer", parameter: bearerToken);
request.Headers.Authorization = header;
HttpResponseMessage response = await client.SendAsync(request,
    HttpCompletionOption.ResponseHeadersRead); // Task<HttpResponseMessage>

if (response.StatusCode.Equals(obj: HttpStatusCode.Unauthorized))...

if (response.StatusCode.Equals(obj: HttpStatusCode.NotFound))...

if (!response.StatusCode.Equals(obj: HttpStatusCode.OK))...

user = response.Content.ReadFromJsonAsync<UserDto>().Result!;

request = new HttpRequestMessage(HttpMethod.Get,
    requestUri: "http://172.19.0.3:80/artemis/data/match/get/by-user");
request.Headers.Authorization = header;
response = await client.SendAsync(request, HttpCompletionOption.ResponseHeadersRead);

if (response.StatusCode.Equals(obj: HttpStatusCode.Unauthorized))...

if (response.StatusCode.Equals(obj: HttpStatusCode.NotFound))...

if (!response.StatusCode.Equals(obj: HttpStatusCode.OK))...

matches = response.Content.ReadFromJsonAsync<List<MatchOutputDto>>().Result!
    .OrderByDescending(x :MatchOutputDto => x.StartTimestamp.TimeStamp.Date).ToList(); //
return Page();
```

Slika 4.9. ***Prikaz isječka programskog kôda metode unosa novog zapisa***

4.3.6. Pregled postojećeg/-ih zapisa

Nakon autorizacije, odnosno prijave, korisnik je preusmjeren na pregled postojećih zapisa. Na ovom pregledu vidljivi su svi zapisi koje je korisnik pohranio u bazu podataka, u obliku popisa definiranog u potpoglavlju 3.1.7. Korisnik također može pristupiti ovom pregledu posjetom početne stranice uz uvjet prisutnog valjanog autorizacijskog tokena u predmemoriji internetskog preglednika. Posjetom stranice na kojoj je dostupan pregled svih unesenih zapisa korisnika, prvo se provjerava valjanost autorizacijskog tokena, a zatim se od sučelja zahtijevaju podaci o unesenim zapisima u sažetom obliku, odnosno sadržavajući samo podatke koji su prikazani za svaku stavku popisa. U slučaju da autorizirani korisnik prvi puta otvara aplikaciju, odnosno u bazi podataka ne postoji niti jedan zapis povezan s korisnikom, aplikacija korisnika preusmjerava na unos novog zapisa, koji je opisan u prethodnom potpoglavlju. Isječak programskog kôda koji prikazuje osnovnu logiku metode dohvaćanja pregleda svih zapisa prikazan je na slici 4.10.

```
var request = new HttpRequestMessage(HttpMethod.Get,
    requestUri: "http://172.19.0.3:80/artemis/auth/get-user/by-id");
var client = _httpClientFactory.CreateClient();
var header = new AuthenticationHeaderValue(scheme: "Bearer", parameter: bearerToken);
request.Headers.Authorization = header;
HttpResponseMessage response = await client.SendAsync(request,
    HttpCompletionOption.ResponseHeadersRead); // Task<HttpResponseMessage>

if (response.StatusCode.Equals(obj: HttpStatusCode.Unauthorized))...

if (response.StatusCode.Equals(obj: HttpStatusCode.NotFound))...

if (!response.StatusCode.Equals(obj: HttpStatusCode.OK))...

user = response.Content.ReadFromJsonAsync<UserDto>().Result!;

request = new HttpRequestMessage(HttpMethod.Get,
    requestUri: "http://172.19.0.3:80/artemis/data/match/get/by-user");
request.Headers.Authorization = header;
response = await client.SendAsync(request, HttpCompletionOption.ResponseHeadersRead);

if (response.StatusCode.Equals(obj: HttpStatusCode.Unauthorized))...

if (response.StatusCode.Equals(obj: HttpStatusCode.NotFound))...

if (!response.StatusCode.Equals(obj: HttpStatusCode.OK))...

matches = response.Content.ReadFromJsonAsync<List<MatchOutputDto>>().Result!
    .OrderByDescending(x :MatchOutputDto => x.StartTimestamp.TimeStamp.Date).ToList(); // L
return Page();
```

Slika 4.10. ***Prikaz isječka programskog kôda metode pregleda popisa postojećih zapisa***

Za svaki od zapisa prikazanih u popisu, korisnik može odgovarajućim tipkama pored zapisa odabrati pregled svih unesenih podataka zapisa ili brisanje zapisa iz baze podataka, koje se provodi prema opisu metode u potpoglavlju 4.3.8. Ako odabere pregled svih unesenih podataka, aplikacija preusmjerava korisnika na odgovarajući pregled, na kojem je osim pregleda svih podataka o zapisu moguće pritiskom na odgovarajuću tipku izmijeniti sve unesene podatke, prema metodi opisanoj u potpoglavlju 4.3.7., ili ukloniti prikazani zapis iz baze podataka, prema metodi opisanoj u potpoglavlju 4.3.8. Isječak programskog kôda metode koja upravlja pregledom jednog zapisa prikazan je na slici 4.11.

```
if (!Request.Cookies.TryGetValue("Bearer", out string? bearerToken))...

if (matchId.IsNullOrEmpty())...

var request = new HttpRequestMessage(HttpMethod.Get,
    requestUri: $"http://172.19.0.3:80/artemis/data/match/get/by-id?id={matchId}");
var client = _httpClientFactory.CreateClient();
var header = new AuthenticationHeaderValue(scheme: "Bearer", parameter: bearerToken);
request.Headers.Authorization = header;
HttpResponseMessage response = await client.SendAsync(request);

if (response.StatusCode.Equals(obj: HttpStatusCode.NotFound))...

if (response.StatusCode.Equals(obj: HttpStatusCode.Unauthorized))...

if (!response.StatusCode.Equals(obj: HttpStatusCode.OK))...

MatchOutputDto dto = response.Content.ReadFromJsonAsync<MatchOutputDto>().Result!;
if (Match.ConvertMatch.TryGetValue(dto.Type,
        out Func<MatchOutputDto, Match>? creator))
{
    Match = creator(dto);
    Match.Shots = Match.Shots.OrderBy(x :Shot => x.Position).ToList();
    MatchResult = Match.GetMatchResult();
    return Page();
}
```

Slika 4.11. ***Prikaz isječka programskog kôda metode pregleda jednog zapisa***

4.3.7. Izmjena postojećeg zapisa

Svaki uneseni zapis moguće je u bilo kojem trenutku izmijeniti, uz uvjet prethodne uspješne autorizacije. Odabirom odgovarajuće opcije, aplikacija prikazuje unesene podatke zapisa u obliku obrasca, poput obrasca koji se prikazuje kada korisnik unosi novi zapis. Funkcionalnost izmjene postojećeg zapisa prethodno je opisana u potpoglavlju 3.1.8. U izrađenoj aplikaciji proces izmjene podataka postojećeg zapisa vrlo je sličan procesu unosa novog zapisa, kako sa strane klijenta, tako i sa strane poslužitelja, zbog čega se kao isječak programskog kôda koji prikazuje osnovnu logiku metode izmjene podataka zapisa prilaže slika 4.9.

4.3.8. Brisanje postojećeg zapisa

Svaki uspješno uneseni zapis može se ukloniti iz baze podataka odabirom odgovarajuće opcije s pregleda svih unesenih zapisa i s pregleda svih podataka unesenih zapisa. Za uklanjanje zapisa potrebna je prethodna uspješna autorizacija korisnika te svaki korisnik može ukloniti isključivo vlastite zapise iz baze podataka. Ako korisnik odabere opciju uklanjanja zapisa, aplikacija

prosljeđuje zahtjev sučelju te iščekuje povratnu informaciju. Ako je operacija uspješno izvršena, aplikacija prikazuje poruku potvrde te preusmjerava korisnika na pregled svih unesenih zapisa. U slučaju da operacija nije uspješno izvršena, aplikacija prikazuje odgovarajuću poruku pogreške i iščekuje korisničku akciju. Isječak programskog kôda metode koja izvršava brisanje postojećeg zapisa nalazi se na slici 4.12.

```
if (!Request.Cookies.TryGetValue("Bearer", out string? bearerToken))...

if (matchId.IsNullOrEmpty())...

var request = new HttpRequestMessage(HttpMethod.Delete,
    requestUri: $"http://172.19.0.3:80/artemis/data/match/delete?id={matchId}");
var client = _httpClientFactory.CreateClient();
var header = new AuthenticationHeaderValue(scheme: "Bearer", parameter: bearerToken);
request.Headers.Authorization = header;
HttpResponseMessage response = await client.SendAsync(request);

if (response.StatusCode.Equals(obj: HttpStatusCode.OK))
{
    TempData["AlertSuccess"] = "Match deleted successfully.";
    return RedirectToPage("/Interface");
}

if (response.StatusCode.Equals(obj: HttpStatusCode.Unauthorized))...

if (response.StatusCode.Equals(obj: HttpStatusCode.NotFound))...
```

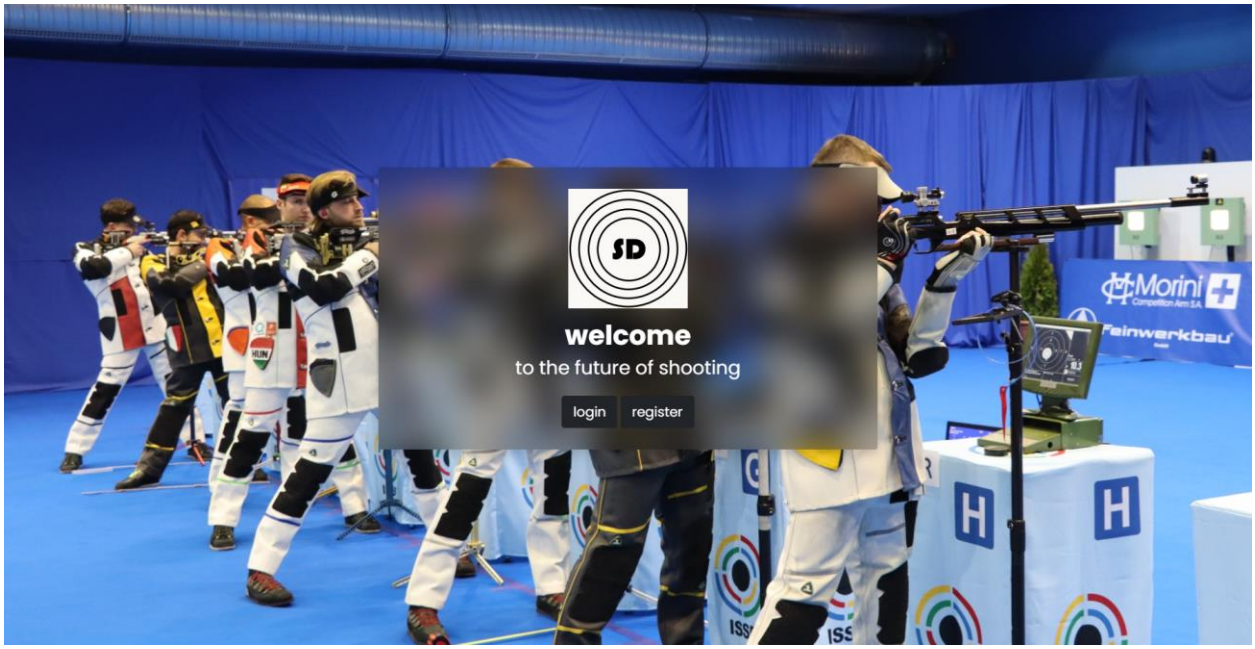
Slika 4.12. ***Prikaz isječka programskog kôda metode brisanja zapisa***

5. UPORABA I TESTIRANJE APLIKACIJE

U ovom poglavlju prikazano je i opisano ponašanje aplikacije prilikom ispravnog korištenja aplikacije te je izvršeno testiranje najvažnijih funkcionalnosti aplikacije, radi provjere ispravnosti izrađene aplikacije. Aplikacija je dostupna na https://shooting.advanc.eu.

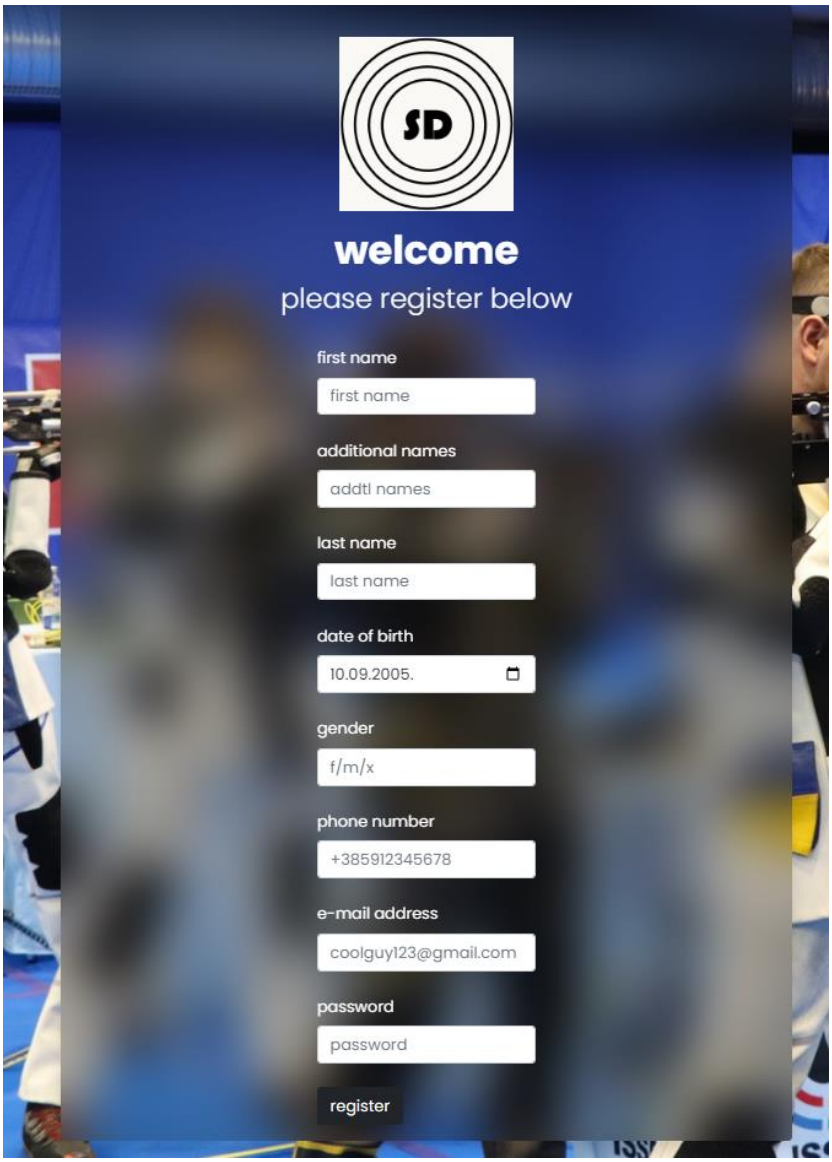
5.1. Uporaba aplikacije

Prilikom pristupa aplikaciji, aplikacija prikazuje početnu stranicu na kojoj su dostupne opcije prijave, odnosno autorizacije, i otvaranja novog korisničkog računa, odnosno registracije. Početna stranica aplikacije prikazana je na slici 5.1.



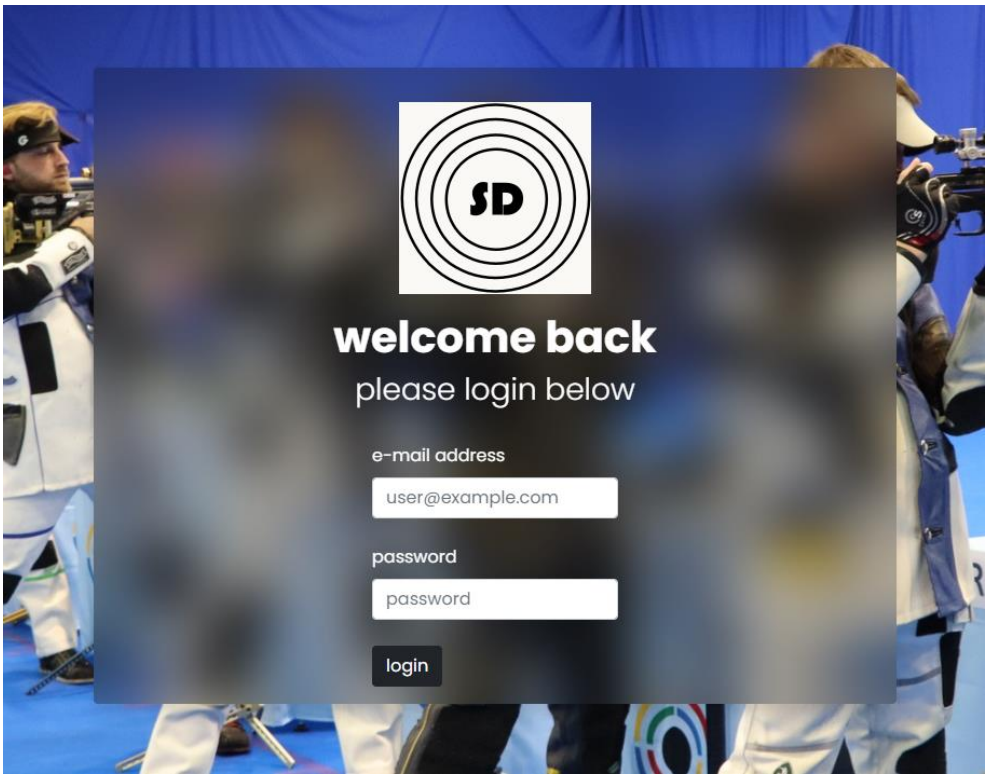
Slika 5.1. ***Prikaz početne stranice aplikacije***

Ako postojeći korisnik posjećuje domenu s valjanim autorizacijskim tokenom u predmemoriji internetskog preglednika, korisniku se umjesto početne stranice prikazuje obrazac za unos novog zapisa, prikazan na slici 5.4., ako ne postoji niti jedan zapis povezan s korisnikom u bazi podataka, ili pregled postojećih zapisa, prikazan na slici 5.5. U slučaju novog korisnika, sljedeći korak je odabir opcije registracije, čime se prikazuje obrazac prikazan na slici 5.2.



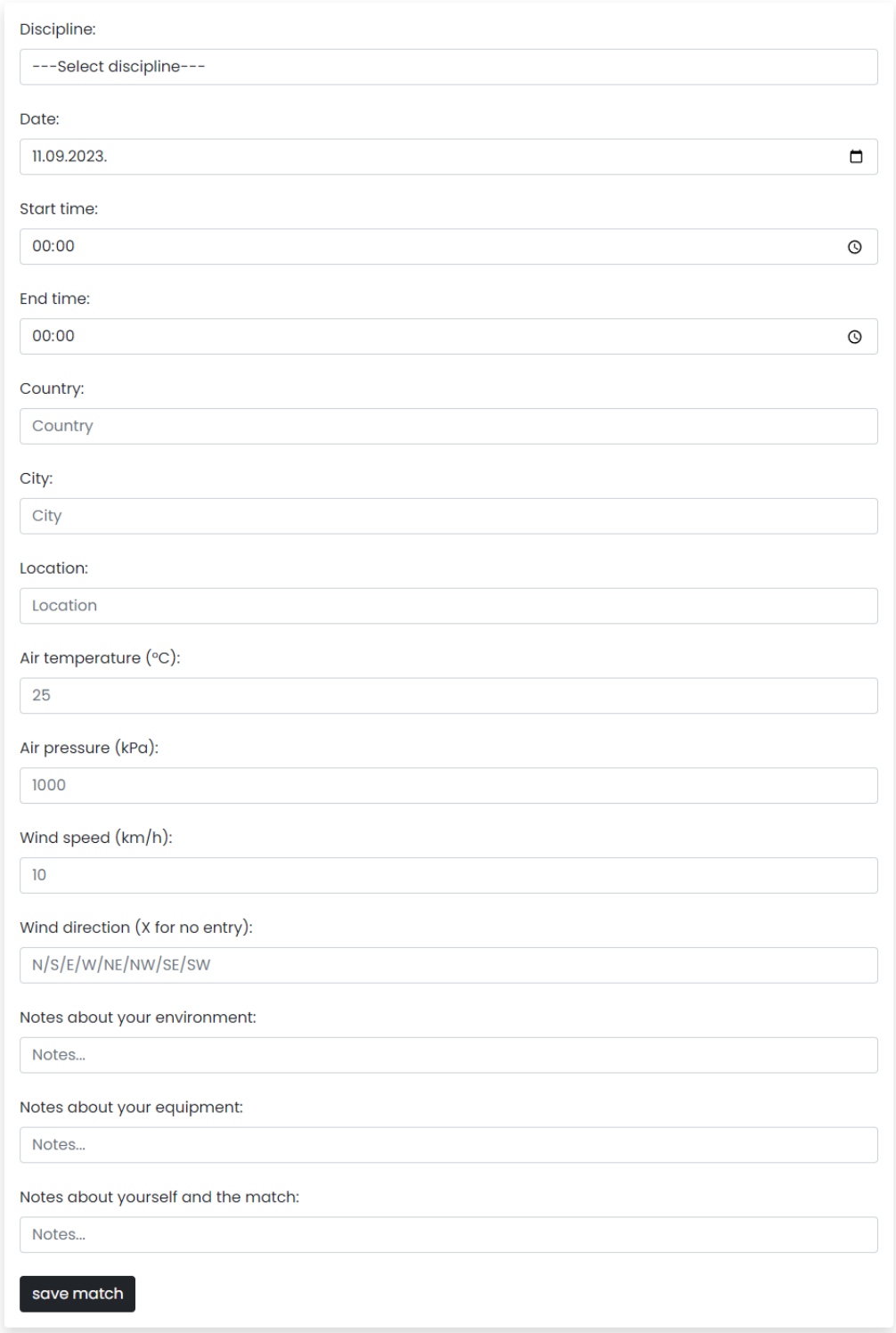
Slika 5.2. ***Prikaz obrasca registracije novog korisnika***

Nakon unosa traženih podataka, korisnik pritiskom na tipku na dnu obrasca može zatražiti otvaranje novog korisničkog računa. Ako su uneseni svi obavezni podaci, definirani u potpoglavlju 3.1.1., te ako adresa elektroničke pošte nije prethodno unesena u bazu podataka prilikom otvaranja drugog korisničkog računa, aplikacija prikazuje poruku o uspješno otvorenom računu na vrhu stranice i preusmjerava korisnika na obrazac za prijavu u aplikaciju, prikazan na slici 5.3.



Slika 5.3. ***Prikaz obrasca autorizacije postojećeg korisnika***

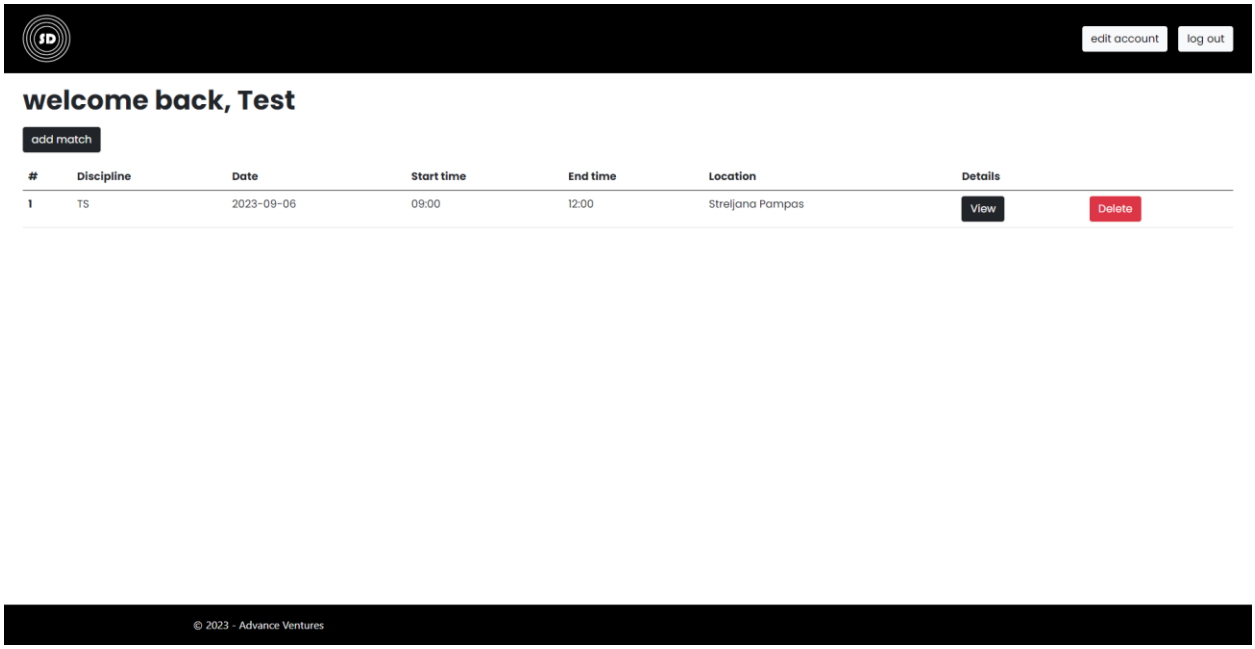
Nakon unosa adrese elektroničke pošte i lozinke, pritiskom na tipku na dnu obrasca aplikacija pokreće metodu autorizacije korisnika, ako svi uneseni podaci odgovaraju predviđenim oblicima. Ako postoji korisnički račun povezan s unesenom adresom elektroničke pošte, i ako je unesena lozinka ispravna, aplikacija prikazuje potvrdu o uspješnoj autorizaciji i korisnika preusmjerava na unos novog zapisa ako u bazi podataka ne postoji niti jedan zapis povezan s korisničkim računom, prikazan na slici 5.4., ili na pregled unesenih zapisa, prikazan na slici 5.5.



Slika 5.4. ***Prikaz obrasca unosa novog zapisa***

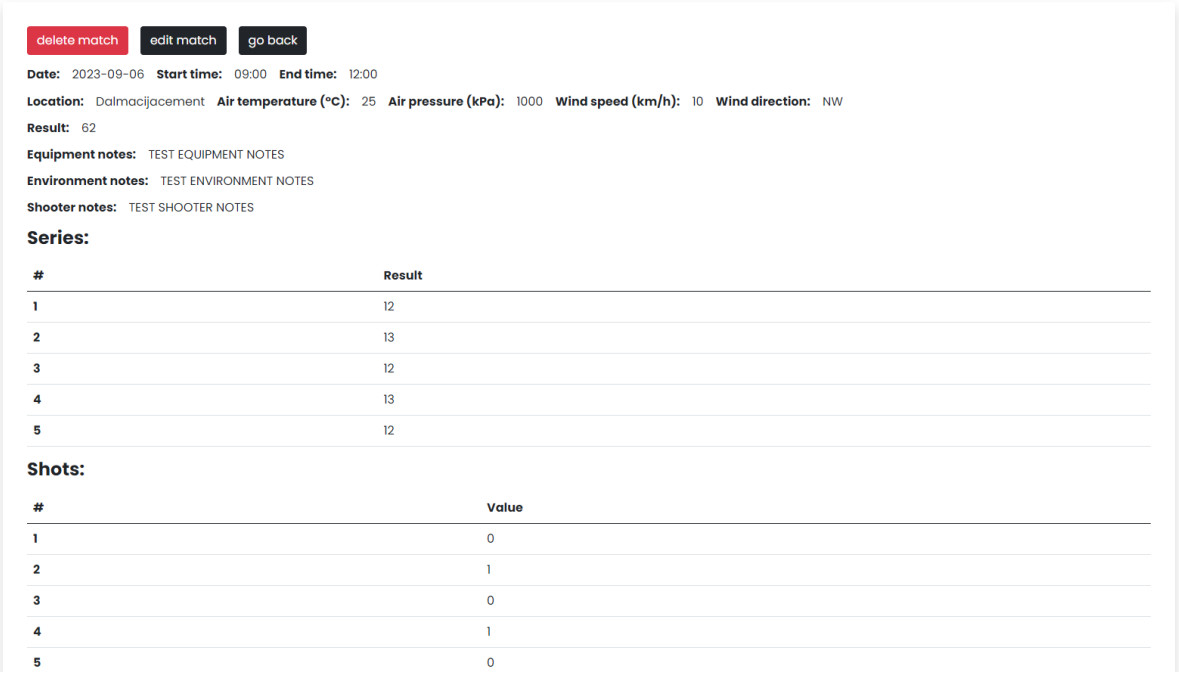
Osim unosa novog zapisa, korisnik može na ovoj stranici odabrati i uređivanje podataka korisničkog računa i odjavu iz aplikacije. Kada je korisnik dovršio unos podataka novog zapisa,

pritiskom na tipku na dnu obrasca pokreće proces provjere unesenih vrijednosti i, ako su unesene sve obavezne vrijednosti te sve unesene vrijednosti poštuju sva postavljena ograničenja, pohrane zapisa u bazu podataka. Nakon uspješne pohrane zapisa u bazu podataka, aplikacija prikazuje poruku o uspješnoj pohrani te preusmjerava korisnika na pregled unesenih zapisa, prikazan na slici 5.5.



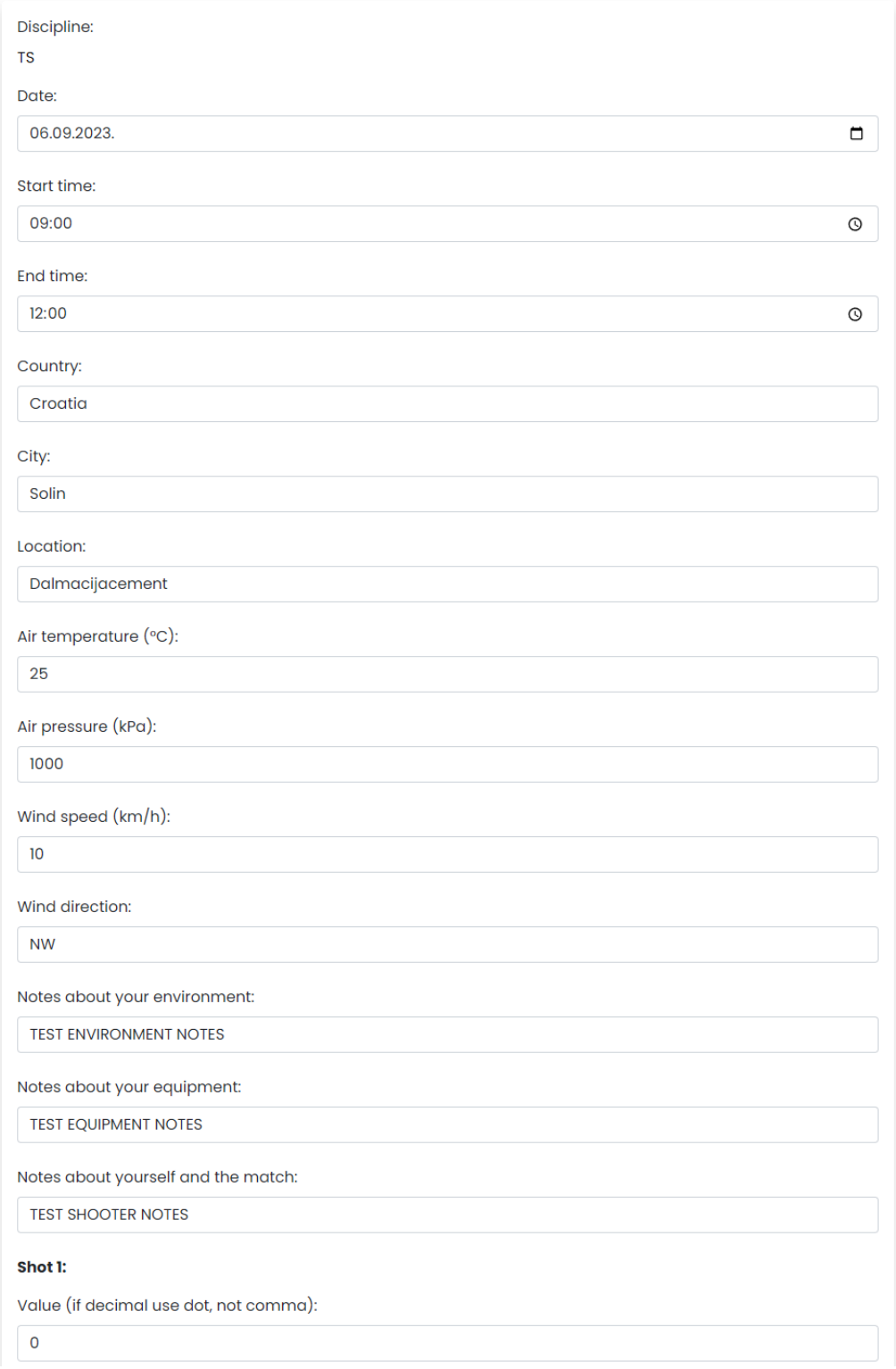
Slika 5.5. ***Prikaz pregleda svih unesenih zapisa***

Korisnik na pregledu zapisa može odabrati unos novog zapisa, pregled ili uklanjanje nekog od postojećih zapisa, uređivanje podataka korisničkog računa, ili odjavu iz aplikacije. U slučaju da odabere pregled svih podataka unesenog zapisa, aplikacija preusmjerava korisnika na stranicu prikazanu na slici 5.6.



Slika 5.6. ***Prikaz pregleda svih podataka unesenog zapisa***

Prikaz na slici 5.6. prilagođava se unesenim podacima zapisa te disciplini zapisa, kako bi prikazani bili isključivo podaci koji su uneseni. Na pregledu svih podataka zapisa, korisnik ponovno može odabrati uklanjanje zapisa iz baze podataka, ali može i odabrati uređivanje podataka zapisa. Pretpostavimo da je korisnik pogriješio pri unosu nekog podatka te želi izmijeniti podatak. Pritiskom na odgovarajuću tipku korisnik je preusmjeren na stranicu prikazanu na slici 5.7.

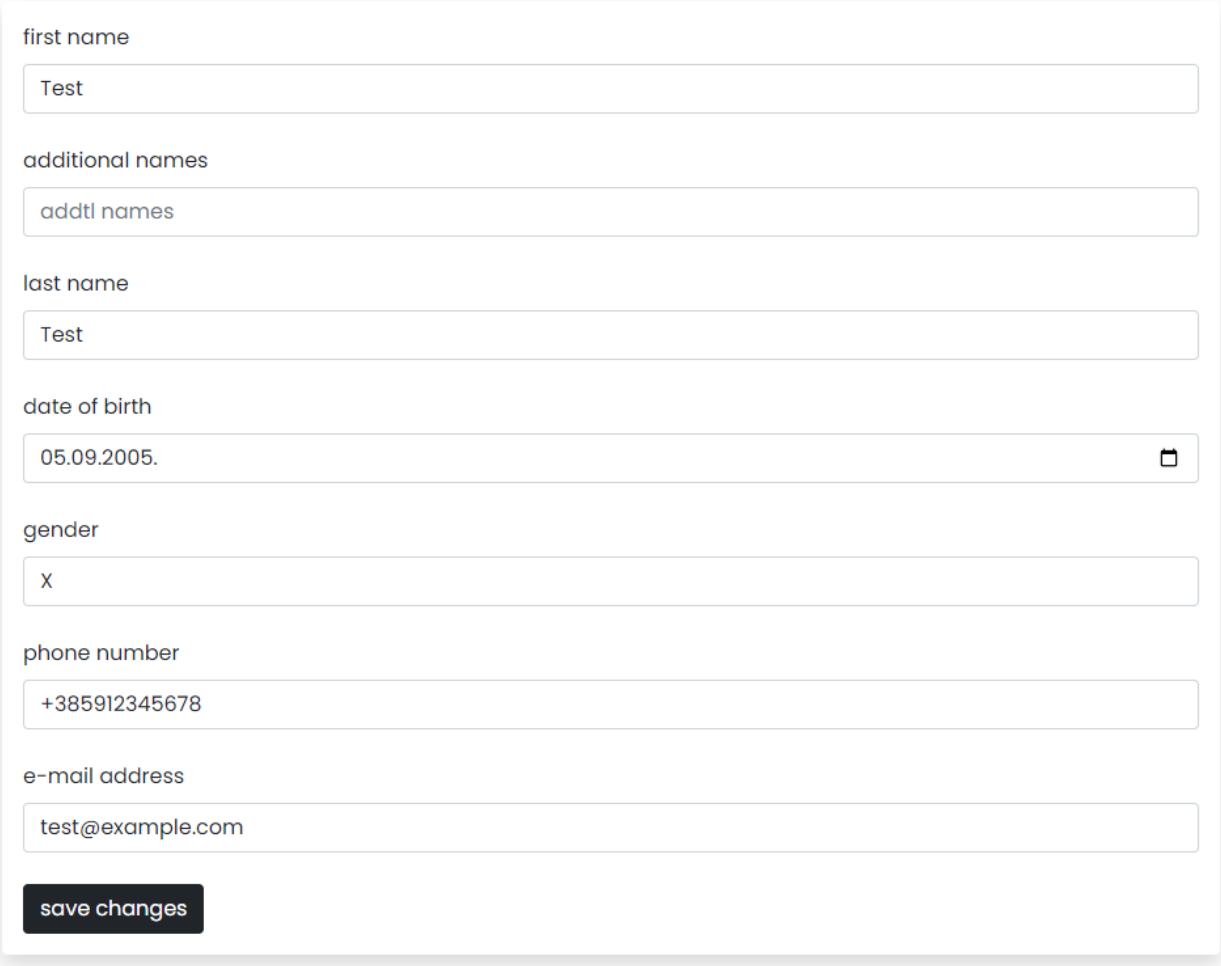


Slika 5.7. ***Prikaz obrasca izmjene podataka zapisa***

Nakon unosa željenih izmjena, korisnik pritiskom na tipku na dnu obrasca potvrđuje kraj izmjena te aplikacija provjerava ispravnost unesenih podataka i, ako su svi podaci ispravno uneseni,

pokreće pohranu podataka u bazu podataka. Nakon uspješne pohrane izmijenjenih podataka, aplikacija prikazuje poruku potvrde uspješne pohrane te korisnika preusmjerava na prikaz pregleda svih unesenih zapisa, prikazan na slici 5.5.

Pretpostavimo da je korisnik pogriješio i prilikom unosa nekog od podataka prilikom registracije te odabere mogućnost izmjene korisničkih podataka. Pritiskom na odgovarajuću tipku na stranici pregleda svih unesenih podataka, korisnik je preusmjeren na stranicu prikazanu na slici 5.8.



Slika 5.8. ***Prikaz obrasca izmjene podataka korisničkog računa***

Pritiskom na tipku na dnu obrasca, nakon unosa željenih promjena, korisnik potvrđuje kraj promjena te aplikacija provjerava ispravnost podataka. Kada podaci poštuju ispunjavaju očekivane zahtjeve, aplikacija prosljeđuje zahtjev za pohranom izmijenjenih podataka te u slučaju uspješne pohrane prikazuje poruku potvrde i preusmjerava korisnika na pregled unesenih zapisa, prikazan na slici 5.5. U slučaju neuspješne pohrane prikazuje poruku o pogrešci i iščekuje korisničku akciju.

5.2. Ispitivanje aplikacije

Nakon izrade aplikacije, važno je provesti ispitivanja rada aplikacije kako bi utvrdili ispravnost aplikacije. U nastavku su opisana tri provedena ispitivanja aplikacije, zajedno s rezultatima provedenih ispitivanja. Prvo ispitivanje aplikacije provjerava izračunavaju li se vrijednosti, poput rezultata serija, prema očekivanjima. Drugo ispitivanje aplikacije provjerava pohranjuju li se pravilno izmjene općih podataka zapisa, poput bilješki, datuma, i sličnoga, dok treće ispitivanje aplikacije provjerava pohranjuju li se pravilno izmjene podataka pogodaka unutar zapisa, budući da je proces pohrane izmijenjenih podataka pogodaka nešto zahtjevniji od procesa pohrane općih podataka zapisa. Za potrebe sva tri ispitivanja stvoren je novi zapis u aplikaciji.

5.2.1. Ispitivanje točnosti izračuna vrijednosti

Aplikacija ima mogućnost izračunavanja rezultata svake serije te sveukupnog rezultata zapisa, pri čemu aplikacija uzima u obzir pravila bodovanja discipline tog zapisa. Za potrebe ispitivanja ove funkcionalnosti, stvoren je zapis o događaju discipline 10m zračna puška, te su vrijednosti pogodaka popunjene prema primjeru na slici 5.9. Na slici su prikazane vrijednosti prvih 10 pogodaka, a ostalih 50 pogodaka je popunjeno ponavljanjem istog niza 5 puta.

Shots:

#	Value	Timestamp	Horizontal displacement	Vertical displacement
1	10	10:00	1.1	-1.1
2	10.1	10:02	2.2	-2.2
3	10.2	10:04	3.3	-3.3
4	10.3	10:06	4.4	-4.4
5	10.4	10:08	5.5	-5.5
6	10.5	10:10	6.6	-6.6
7	10.6	10:12	7.7	-7.7
8	10.7	10:14	8.8	-8.8
9	10.8	10:16	9.9	-9.9
10	10.9	10:18	1.1	-1.1

Slika 5.9. ***Isječak podataka o pogodcima iz zapisa korištenog za ispitivanje aplikacije***

Prema načinu bodovanja discipline definiranom u potpoglavlju 3.2.2., pretpostavlja se da je rezultat svake serije 104.5, da se u svakoj seriji nalazi 6 muševa, da je rezultat cijelog zapisa 627.0, te da se u cijelom zapisu nalazi 36 muševa. Na slici 5.10. prikazani su podaci istog zapisa koji pokazuju kako se očekivane vrijednosti podudaraju s vrijednostima koje je aplikacija izračunala.

Date: 2023-09-05 **Start time:** 10:00 **End time:** 12:00

Location: Streljana Pampas **Air temperature (°C):** 25 **Air pressure (kPa):** 1000 **Wind speed (km/h):** 10 **Wind direction:** N

Result: 627.0 **Inner 10s:** 36

Equipment notes: TEST EQUIPMENT NOTES

Environment notes: TEST ENVIRONMENT NOTES

Shooter notes: TEST SHOOTER NOTES

Series:

#	Result	Inner 10s
1	104.5	6
2	104.5	6
3	104.5	6
4	104.5	6
5	104.5	6
6	104.5	6

Slika 5.10. ***Prikaz podataka o zapisu korištenom za ispitivanje aplikacije***

5.2.2. Ispitivanje izmjene općih podataka zapisa

Svrha aplikacije je digitalna pohrana podataka o ostvarenim rezultatima u streljaštvu, stoga je osim ispravne pohrane novog zapisa važno integrirati i mogućnost izmjene postojećih zapisa, radi mogućnosti ljudske pogreške. Na slici 5.11. prikazani su opći podaci zapisa korištenog za ispitivanje mogućnosti izmjene općih podataka zapisa.

Date: 2023-09-05 **Start time:** 10:00 **End time:** 12:00

Location: Streljana Pampas **Air temperature (°C):** 25 **Air pressure (kPa):** 1000 **Wind speed (km/h):** 10 **Wind direction:** N

Result: 627.0 **Inner 10s:** 36

Equipment notes: TEST EQUIPMENT NOTES

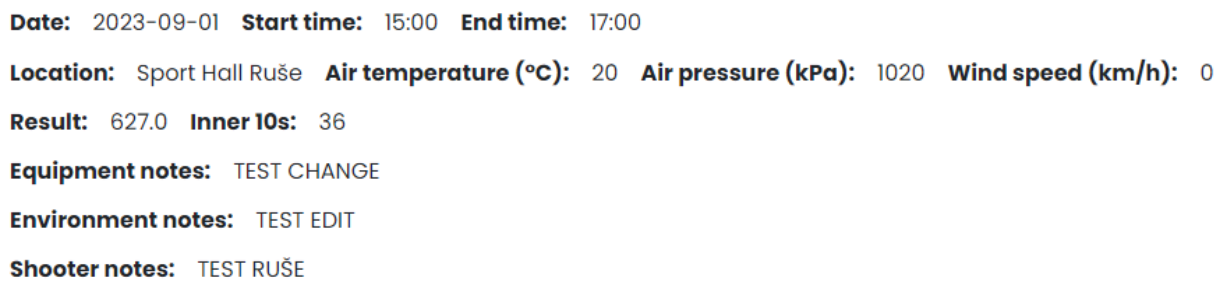
Environment notes: TEST ENVIRONMENT NOTES

Shooter notes: TEST SHOOTER NOTES

Slika 5.11. ***Prikaz općih podataka zapisa korištenog za ispitivanje aplikacije***

Sve vidljive podatke unosi korisnik, osim ukupnog rezultata zapisa i broja muševa u zapisu, koje aplikacija izračunava prilikom dohvaćanja zapisa iz vrijednosti pogodaka zapisa. Za potrebe ispitivanja ispravnosti pohrane izmijenjenih podataka zapisa izmijenjeni su svi podaci te su na slici 5.12. prikazane nove vrijednosti općih podataka zapisa. Vidljivo je kako su sve izmijenjene vrijednosti uspješno pohranjene te da su ukupni rezultat i broj muševa ostali nepromijenjeni, kako

je bilo i očekivano. Smjer vjetra nije prikazan jer je unesena vrijednost 'X', koja označava neprisutnost vjetra.



Slika 5.12. ***Prikaz izmijenjenih općih podataka zapisa korištenog za ispitivanje aplikacije***

5.2.3. Ispitivanje izmjene podataka pogodaka

Posljednje izvršeno ispitivanje je ispitivanje ispravnosti izmjene podataka o pogodcima. Za potrebe ispitivanja ispravnosti ove funkcionalnosti, izmijenjene su vrijednosti prvih 10 pogodaka zapisa. Njihove vrijednosti prije izmjene vidljive su na slici 5.9., budući da je za sva ispitivanja aplikacije korišten isti zapis. Vrijednosti su izmijenjene na način da su sve vrijednosti pomaknute za jedno mjesto unaprijed (vrijednosti prvog pogotka postaju vrijednostima drugog pogotka), pri čemu su vrijednosti posljednjeg pogotka upisane na mjesta vrijednosti prvog pogotka. Prikaz vrijednosti prvih 10 pogodaka zapisa nalazi se na slici 5.13., pri čemu se može zaključiti kako je pohrana izmijenjenih vrijednosti ispravno izvršena te je proizvela očekivane rezultate.

Shots:

#	Value	Timestamp	Horizontal displacement	Vertical displacement
1	10.9	10:18	1.1	-1.1
2	10	10:00	1.1	-1.1
3	10.1	10:02	2.2	-2.2
4	10.2	10:04	3.3	-3.3
5	10.3	10:06	4.4	-4.4
6	10.4	10:08	5.5	-5.5
7	10.5	10:10	6.6	-6.6
8	10.6	10:12	7.7	-7.7
9	10.7	10:14	8.8	-8.8
10	10.8	10:16	9.9	-9.9

Slika 5.13. ***Prikaz izmijenjenih vrijednosti pogodaka zapisa korištenog za ispitivanje aplikacije***

6. ZAKLJUČAK

Sportsko streljaštvo jedan je od slabije zastupljenih sportova u medijskoj sferi, zbog čega je put do opće modernizacije i digitalizacije praćenja rezultata i napretka sportaša dug i pun prepreka. Cilj završnog rada bio je izraditi aplikaciju za bilježenje rezultata u streljaštvu koju bi mogli koristiti i korisnici slabijih vještina uporabe računala.

Izrađena aplikacija od svakog korisnika prije pohranjivanja zapisa očekuje otvaranje korisničkog računa kojeg će kasnije koristiti za prijavu. Autorizirani korisnik unutar aplikacije može pohranjivati zapise o ostvarenim rezultatima u sedam podržanih disciplina sportskog streljaštva, koje su izabrane prema zastupljenosti na Olimpijskim igrama. Sve zapise je moguće naknadno izmijeniti te ih je moguće i ukloniti iz baze podataka. Podatke unesene prilikom registracije je također moguće izmijeniti, osim lozinke, te je moguće i ukloniti cijeli korisnički račun.

Rad aplikacije je ispitan s tri ispitivanja, koja su osmišljena s obzirom na ključne funkcionalnosti aplikacije. Sva tri ispitivanja dokazala su da aplikacija ispravno ispunjava svoje zadatke, no postoje mogućnosti za poboljšanje iskustva korisnika aplikacije. U nastavku razvoja aplikacije korisničko iskustvo moglo bi se poboljšati integracijom tehnologije optičkog prepoznavanja znakova, kako bi korisnici imali mogućnost unosa zapisa rezultata prijenosom fotografija ili dokumenta s rezultatima na poslužitelj, umjesto ručnog ispunjavanja obrasca.

LITERATURA

[1] International Sport Shooting Federation, „History“ [online]. Dostupno na: https://www.issf-sports.org/theissf/history.ashx. [Pristupljeno: 5.9.2023.].

[2] International Sport Shooting Federation, „World championships“ [online]. Dostupno na: https://www.issf-sports.org/theissf/championships/world_championships.ashx. [Pristupljeno: 5.9.2023.].

[3] International Sport Shooting Federation, „Olympic games“ [online]. Dostupno na: https://www.issf-sports.org/theissf/championships/olympic_games.ashx. [Pristupljeno: 5.9.2023.].

[4] Hrvatski streljački savez, „Pravila natjecanja za sve streljačke discipline“ [online]. Dostupno na: https://hss-csf.hr/dokumenti/dokumentacija/HSS%20-%20Pravila%20natjecanja%20HSS%202022.pdf. [Pristupljeno: 5.9.2023.].

[5] SIUS AG, „SIUS AG | Electronic Scoring Systems, Effretikon, Official ISSF Results Provider“ [online]. Dostupno na: https://www.sius.com/en. [Pristupljeno: 5.9.2023.].

[6] SIUS AG, „Shootingsportscloud-EN.pdf“ [online]. Dostupno na: https://drive.google.com/file/u/0/d/1ofRDPXN7TCBBS5chAi0JVStG5CtWmKE2/view?pli=1&usp=embed_facebook. [Pristupljeno: 5.9.2023.].

[7] Međunarodna federacija sportskog streljaštva, „ISSF Pravila za pištolj 2017.pdf“ [online]. Dostupno na: https://hss-csf.hr/dokumenti/dokumentacija/ISSF%20Pravila%20za%20pi%C5%A1tolj%202017.pdf. [Pristupljeno: 5.9.2023.].

[8] Međunarodna federacija sportskog streljaštva, „ISSF Pravila za pušku 2017.pdf“ [online]. Dostupno na: https://hss-csf.hr/dokumenti/dokumentacija/ISSF%20Pravila%20za%20pu%C5%A1ku%202017.pdf. [Pristupljeno: 5.9.2023.].

[9] International Sport Shooting Federation, „Shotgun rules“ [online]. Dostupno na: https://www.issf-sports.org/getfile.aspx?mod=docf&pane=1&inst=462&file=Shotgun_Rules.pdf. [Pristupljeno: 5.9.2023.].

[10] Microsoft, „What is Visual Studio?“ [online], 05-svi-2023. Dostupno na: https://learn.microsoft.com/en-us/visualstudio/get-started/visual-studio-ide?view=vs-2022. [Pristupljeno: 5.9.2023.].

[11] Microsoft, „Editions and supported features of SQL Server 2022 - SQL Server“ [online], 06-srp-2023. Dostupno na: https://learn.microsoft.com/en-us/sql/sql-server/editions-and-components-of-sql-server-2022?view=sql-server-ver16. [Pristupljeno: 5.9.2023.].

[12] Amazon, „What is Docker? | AWS“ [online]. Dostupno na: https://aws.amazon.com/docker/. [Pristupljeno: 5.9.2023.].

[13] Microsoft, „Overview of ASP.NET Core“ [online], 15-stu-2022. Dostupno na: https://learn.microsoft.com/en-us/aspnet/core/introduction-to-aspnet-core?view=aspnetcore-7.0. [Pristupljeno: 5.9.2023.].

[14] Microsoft, „Overview of Entity Framework Core - EF Core“ [online], 25-svi-2021. Dostupno na: https://learn.microsoft.com/en-us/ef/core/. [Pristupljeno: 5.9.2023.].

[15] Microsoft, „A tour of C# - Overview - C#“ [online], 04-svi-2023. Dostupno na: https://learn.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/. [Pristupljeno: 5.9.2023.].
[16] Mozilla Corporation, „HTML: HyperText Markup Language | MDN“ [online], 17-srp-2023. Dostupno na: https://developer.mozilla.org/en-US/docs/Web/HTML. [Pristupljeno: 5.9.2023.].
[17] Mozilla Corporation, „CSS: Cascading Style Sheets | MDN“ [online], 22-srp-2023. Dostupno na: https://developer.mozilla.org/en-US/docs/Web/CSS. [Pristupljeno: 5.9.2023.].
[18] Mozilla Corporation, „JavaScript | MDN“ [online], 08-srp-2023. Dostupno na: https://developer.mozilla.org/en-US/docs/Web/JavaScript. [Pristupljeno: 5.9.2023.].

POPIS SLIKA

Slika 2.1 Prikaz sučelja aplikacije Shooting Sports Cloud 4
Slika 2.2 Prikaz sučelja aplikacije Shooters Diary 5
Slika 2.3 Prikaz sučelja aplikacije TargetScan ISSF Pistol & Rifle 6
Slika 2.4 Prikaz sučelja aplikacije Air Shooting Diary: Pro Notes 6
Slika 4.1 Prikaz definiranja načina zapisivanja podataka o korisniku 15
Slika 4.2 Relacijski model nastale baze podataka 16
Slika 4.3 Prikaz programskog kôda autorizacije korisnika na upravljaču sučelja 17
Slika 4.4 Prikaz isječka programskog kôda metode registracije novog korisnika 18
Slika 4.5 Prikaz isječka programskog kôda metode prijave postojećeg korisnika 19
Slika 4.6 Prikaz isječka programskog kôda metode uređivanja podataka postojećeg korisnika .. 20
Slika 4.7 Prikaz isječka programskog kôda metode brisanja korisničkog računa 21
Slika 4.8 Prikaz programskog kôda metode odjave korisnika 22
Slika 4.9 Prikaz isječka programskog kôda metode unosa novog zapisa 23
Slika 4.10 Prikaz isječka programskog kôda metode pregleda popisa postojećih zapisa 24
Slika 4.11 Prikaz isječka programskog kôda metode pregleda jednog zapisa 25
Slika 4.12 Prikaz isječka programskog kôda metode brisanja zapisa 26
Slika 5.1 Prikaz početne stranice aplikacije 27
Slika 5.2 Prikaz obrasca registracije novog korisnika 28
Slika 5.3 Prikaz obrasca autorizacije postojećeg korisnika 29
Slika 5.4 Prikaz obrasca unosa novog zapisa 30
Slika 5.5 Prikaz pregleda svih unesenih zapisa 31
Slika 5.6 Prikaz pregleda svih podataka unesenog zapisa 32
Slika 5.7 Prikaz obrasca izmjene podataka zapisa 33
Slika 5.8 Prikaz obrasca izmjene podataka korisničkog računa 34
Slika 5.9 Isječak podataka o pogodcima iz zapisa korištenog za ispitivanje aplikacije 35
Slika 5.10 Prikaz podataka o zapisu korištenom za ispitivanje aplikacije 36
Slika 5.11 Prikaz općih podataka zapisa korištenog za ispitivanje aplikacije 36
Slika 5.12 Prikaz izmijenjenih općih podataka zapisa korištenog za ispitivanje aplikacije 37
Slika 5.13 Prikaz izmijenjenih vrijednosti pogodaka zapisa korištenog za ispitivanje aplikacije 37

SAŽETAK

U ovom završnom radu izrađeno je programsko rješenje za bilježenje rezultata u sportskom streljaštvu korištenjem razvojnog okruženja Visual Studio, razvojnog okvira ASP.NET Core, te C#, HTML, CSS, i JavaScript jezika. Za pohranu podataka koristi se relacijska baza podataka kojom upravlja instanca Microsoft SQL Server sustava, dok se izrađeno programsko rješenje sastoji od aplikacijskog programskog sučelja i internetske aplikacije. Sve tri komponente pokrenute su na CentOS poslužitelju korištenjem Docker platforme. U programskom rješenju podržano je sedam disciplina sportskog streljaštva koje su prisutne na Olimpijskim igrama. Osim jednostavne pohrane unesenih podataka, aplikacija je sposobna izračunati rezultate pojedinih serija unutar zapisa te ukupni rezultat zapisa poštujući pravila bodovanja podržanih disciplina. Aplikacija je ispitana trima slučajevima uporabe te rezultati potvrđuju ispravnost i uporabljivost aplikacije.

Ključne riječi: aplikacijsko programsko sučelje, digitalizacija sporta, internetska aplikacija, sportsko streljaštvo

ABSTRACT

Title: Application for recording shooting results in the C# programming language

In this final thesis, a software solution for storing sport shooting results has been developed using the Visual Studio development environment, the ASP.NET Core framework, and the C#, HTML, CSS, and JavaScript languages. A relational database managed by an instance of the Microsoft SQL Server system is used for data storage, and the developed software solution consists of an application programming interface and a web application. All three components are hosted on a CentOS server using the Docker platform. The software solution supports seven sport shooting disciplines present at the Olympic Games. In addition to simple data storage, the application is capable of calculating the results of individual series within result records and the overall score of a record, while adhering to the scoring rules of the supported disciplines. The application was tested on three use cases, and the testing results confirm the correctness and usability of the application.

Keywords: application programming interface, sport digitalization, web application, sport shooting

ŽIVOTOPIS

Leo Tumbas rođen je u Osijeku 9. travnja 2002. godine. Od 2014. godine bavi se sportskim streljaštvom u Građanskom streljačkom društvu „Osijek 1784“, i predstavlja Republiku Hrvatsku na Europskim prvenstvima u streljaštvu 2020. i 2021. godine. Srednjoškolsko obrazovanje završava 2020. godine u III. gimnaziji Osijek. Od listopada 2020. godine pohađa prijediplomski sveučilišni studij Računarstva, smjer Računalno inženjerstvo, na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija Osijek. Od listopada 2021. do listopada 2022. godine služi kao zamjenik člana Studentskog zbora Fakulteta elektrotehnike, računarstva i informacijskih tehnologija Osijek, a od listopada 2022. do lipnja 2023. godine kao zamjenik predsjednika istog zbora i predstavnik studenata u Fakultetskom vijeću istog fakulteta. Na Fakultetu je od ožujka 2022. godine aktivan i kao laboratorijski demonstrator na laboratorijskim vježbama kolegija Programiranje I, Programiranje II, i Digitalna elektronika. Od lipnja 2023. godine služi kao predstavnik studenata tehničkih znanosti u Studentskom zboru Sveučilišta Josipa Jurja Strossmayera u Osijeku. Od rujna 2023. godine član je Hrvatskog instituta kibernetičke sigurnosti i Studentske sekcije istog instituta. Dobitnik je preko 30 medalja i sportskih ostvarenja te mnogih priznanja i nagrada: od kluba dobiva u prosincu 2017. godine priznanje za iznimne rezultate u 333. natjecateljskoj sezoni, od Hrvatskog streljačkog saveza dobiva u lipnju 2022. godine zvanje „Majstor strijelac“, i od Zajednice osječkog sporta u srpnju 2022. godine dobiva stipendiju za darovite sportaše. Također, u listopadu 2021. godine za uspješno studiranje dobiva STEM stipendiju Ministarstva znanosti i obrazovanja Republike Hrvatske. U listopadu 2020. godine biva svrstan u VI. kategoriju sportaša prema Hrvatskom olimpijskom odboru, a u travnju 2022. godine biva svrstan u V. kategoriju sportaša.

_____

Potpis autora

PRILOZI

Prilog 1. Završni rad u formatu .docx

Prilog 2. Završni rad u formatu .pdf

Prilog 3. Izvorni kôd internetske aplikacije (dostupan na https://github.com/antimonious/Athena)

Prilog 4. Izvorni kôd aplikacijskog programskog sučelja (dostupan na https://github.com/antimonious/Artemis)