# Implementacija protokola za udaljeni pristup na primjeru web aplikacije

**Blavicki, Sandro**



**SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU**

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

**Diplomski sveučilišni studij**

# IMPLEMENTACIJA PROTOKOLA ZA UDALJENI PRISTUP NA PRIMJERU WEB APLIKACIJE

**Diplomski rad**

**Sandro Blavicki**

**Osijek,2020.**

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

**Obrazac D1: Obrazac za imenovanje Povjerenstva za diplomski ispit**

**Osijek, 11.09.2023.**

**Odboru za završne i diplomske ispite**

# Imenovanje Povjerenstva za diplomski ispit

| | |
|---|---|
| **Ime i prezime Pristupnika:** | Sandro Blavicki |
| **Studij, smjer:** | Diplomski sveučilišni studij Računarstvo |
| **Mat. br. Pristupnika, godina upisa:** | D-1105R, 13.10.2020. |
| **OIB studenta:** | 82640700036 |
| **Mentor:** | doc. dr. sc. Tomislav Galba |
| **Sumentor:** | , |
| **Sumentor iz tvrtke:** | |
| **Predsjednik Povjerenstva:** | izv. prof. dr. sc. Alfonzo Baumgartner |
| **Član Povjerenstva 1:** | doc. dr. sc. Tomislav Galba |
| **Član Povjerenstva 2:** | prof. dr. sc. Krešimir Nenadić |
| **Naslov diplomskog rada:** | Implementacija protokola za udaljeni pristup na primjeru web aplikacije |
| **Znanstvena grana diplomskog rada:** | **Programsko inženjerstvo (zn. polje računarstvo)** |
| **Zadatak diplomskog rada:** | Potrebno je napraviti web aplikaciju (npr. Spring framework) s više korisničkih rola (minimalno dvije od kojih je jedna administratorska). Administratorskom računu potrebno je omogućiti udaljeni pristup računalu običnog korisnika (minimalno prikaz ekrana) koristeći neki od protokola za udaljeni pristup (npr. RFB protokol). Tema rezervirana: Sandro Blavicki). |
| **Prijedlog ocjene pismenog dijela ispita (diplomskog rada):** | Izvrstan (5) |
| **Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:** | Primjena znanja stečenih na fakultetu: 3 bod/boda<br>Postignuti rezultati u odnosu na složenost zadatka: 3 bod/boda<br>Jasnoća pismenog izražavanja: 3 bod/boda<br>Razina samostalnosti: 3 razina |
| **Datum prijedloga ocjene od strane mentora:** | 11.09.2023. |

| | |
|---|---|
| Potvrda mentora o predaji konačne verzije rada: | *Mentor elektronički potpisao predaju konačne verzije.* |
| | Datum: |

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

# IZJAVA O ORIGINALNOSTI RADA

**Osijek, 03.10.2023.**

| | |
|---|---|
| **Ime i prezime studenta:** | Sandro Blavicki |
| **Studij**: | Diplomski sveučilišni studij Računarstvo |
| **Mat. br. studenta, godina upisa:** | D-1105R, 13.10.2020. |
| **Turnitin podudaranje [%]:** | 6 |

Ovom izjavom izjavljujem da je rad pod nazivom **: Implementacija protokola za udaljeni pristup na primjeru web aplikacije**

izrađen pod vodstvom mentora doc. dr. sc. Tomislav Galba

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 studenta:

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

**IZJAVA**

**o odobrenju za pohranu i objavu ocjenskog rada**

kojom ja __Sandro Blavicki_____, OIB: __82640700036_____, student Fakulteta elektrotehnike, računarstva i informacijskih tehnologija Osijek na studiju _Diplomski sveučilišni studij Računarstvo_, kao autor ocjenskog rada pod naslovom: _Implementacija protokola za udaljeni pristup na primjeru web aplikacije**,**

dajem odobrenje da se, bez naknade, trajno pohrani moj ocjenski rad u javno dostupnom digitalnom repozitoriju ustanove Fakulteta elektrotehnike, računarstva i informacijskih tehnologija Osijek i Sveučilišta te u javnoj internetskoj bazi radova Nacionalne i sveučilišne knjižnice u Zagrebu, sukladno obvezi iz odredbe članka 83. stavka 11. *Zakona o znanstvenoj djelatnosti i visokom obrazovanju* (NN 123/03, 198/03, 105/04, 174/04, 02/07, 46/07, 45/09, 63/11, 94/13, 139/13, 101/14, 60/15).
Potvrđujem da je za pohranu dostavljena završna verzija obranjenog i dovršenog ocjenskog rada. Ovom izjavom, kao autor ocjenskog rada dajem odobrenje i da se moj ocjenski rad, bez naknade, trajno javno objavi i besplatno učini dostupnim:

a) široj javnosti
b) studentima/icama i djelatnicima/ama ustanove
c) široj javnosti, ali nakon proteka 6 / 12 / 24 mjeseci (zaokružite odgovarajući broj mjeseci).

**U slučaju potrebe dodatnog ograničavanja pristupa Vašem ocjenskom radu, podnosi se obrazloženi zahtjev nadležnom tijelu Ustanove.*

__Osijek, 03.10.2023._____ _____

(mjesto i datum) (vlastoručni potpis studenta/ice)

# Sadržaj

# 1. UVOD

Cijeli svijet doživio je velike promjene kao posljedica Covid-19 pandemije. Promijenjeni su načini na koje ljudi žive, rade i obrazuju se. Velike promjene vidljive su u području obrazovnog sustava i u području tehničke industrije, gdje su od ključne važnosti mogućnost udaljenog pristupa i mogućnost međusobne suradnje u stvarnome vremenu. Jedna od tehnologija koje predvode digitalnu transformaciju današnjeg svijeta zove se VNC (engl. *Virtual Network Computing*). VNC je grafički sustav za dijeljenje radne površine i pruža korisnicima mogućnost upravljanja računala na udaljenim lokacijama. Uz današnja događanja, široka primjena VNC tehnologije je uočljivija nego ikada, zbog čega je i velika pažnja na istraživanju i implementiranju VNC tehnologije.

Cilj ovog završnog rada je dizajnirati i implementirati web aplikaciju koja će pružati mogućnosti VNC tehnologije, te tako na fleksibilan i pristupačan način pružiti mogućnosti pristupa udaljenim računalima.

Diplomski rad sadrži šest poglavlja. Prvo poglavlje je uvodno poglavlje u kojem je opisati cilj ovog diplomski rada. U drugom poglavlju opisano je područje teme rada te pregled sličnih postojećih programskih rješenja koja se nalaze na tržištu i pružaju slične usluge. U trećem poglavlju opisani su zahtjevi na web aplikaciju, predložena je arhitektura rješavanja problema te su opisane tehnologije korištene pri izradi projektnog rješenja. Četvrto poglavlje sadrži opis kako se implementira razvijeno programsko rješenje. U petom poglavlju prikazano je i opisano korisničko sučelje te način uporabe aplikacije. U šestom poglavlju izveden je zaključak na temelju programskog rješenja.

## 1.1. Zadatak diplomskog rada

Potrebno je napraviti web aplikaciju (npr. Spring framework) s više korisničkih uloga (minimalno dvije od kojih je jedna administratorska). Administratorskom računu potrebno je omogućiti udaljeni pristup računalu običnog korisnika (minimalno prikaz ekrana) koristeći neki od protokola za udaljeni pristup (npr. RFB protokol).

# 2. PREGLED PODRUČJA TEME

U narednom dijelu opisan je način rada VNC tehnologije te su opisana poznata rješenja koja nude sličnu uslugu kao i izrađeno programsko rješenje.

## 2.1. VNC

Virtualno mrežno računarstvo (VNC) je tehnologija koja mijenja način na koji gledamo i koristimo računalne sustave olakšavanjem pristupa i kontrole udaljenih računala. Ključ uspjeha VNC tehnologije je mogućnost rješavanja tradicionalnih problema računarstva specifičnih za lokaciju, čime se proširuje područje digitalne interakcije.

VNC radi na grafičkom sustavu dijeljenja radne površine koji koristi protokol *Remote Frame Buffer* (RFB). Ovaj sustav omogućuje prijenos događaja tipkovnice i miša s računala zvano preglednik ili klijent, na drugo, udaljeno računalo zvano poslužitelj, koje zauzvrat šalje nazad pregledniku ažurirano stanje zaslona. RFB protokol, kao središnja logika VNC tehnologije, prilično je jednostavan i temelji se na prijenosu grafičkih informaciju poput slika, korisničkih sučelja i vizualnih reprezentacija. RFB protokol radi na razini međuspremnika okvira, što predstavlja međuspremnik grafičkog elementa gdje se pohranjuju prikazani podaci. Radeći na ovoj razini, RFB protokol čini VNC sustav neovisnim o operacijskom sustavu na kojem radi.

VNC tehnologija obično se sastoji od VNC poslužitelja i VNC preglednika. Računalo s kojeg se prenose događaji tipkovnice i miša naziva se klijent ili preglednik. VNC preglednik radi na računalu koje korisnik želi koristiti za kontrolu poslužitelja. On tumači i prikazuje podatke poslane s poslužitelja i omogućuje korisniku interakciju s poslužiteljem. VNC poslužitelj radi na udaljenom računalu s kojim se želi upravljati. Kao odgovor na prenesene događaje tipkovnice i miša, udaljeno, poslužitelj računalo zauzvrat pregledniku šalje natrag ažuriranja grafičkog zaslona. Ovaj proces omogućuje pregledniku daljinsko upravljanje poslužiteljem kao da se koristi njime. Cijeli proces opisan u ovome poglavlju ilustriran je na slici 2.1.. [1]


**Slika 2.1.** *Dijagram VNC tehnologije*

## 2.2. Pregled sličnih rješenja

U narednom dijelu opisano je pet popularnih aplikacija koje nude implementaciju VNC tehnologije te daljinskog upravljanja računalima.

### 2.2.1. Microsoft Teams

Microsoft Teams popularna je platforma za suradnju i komunikaciju koja timovima omogućuje zajednički rad na daljinu. Nudi širok raspon značajki dizajniranih za povećanje produktivnosti i pojednostavljenje komunikacije unutar organizacija. Jedna bitna značajka Microsoft Teamsa je njegova mogućnost daljinskog pristupa. Ona omogućuje pojedincima da ostanu povezani i učinkovito surađuju, bez obzira na njihovu fizičku lokaciju. [2]

Uz funkcionalnosti daljinskog pristupa timovi mogu komunicirati, dijeliti datoteke, održavati sastanke i surađivati na projektima u stvarnom vremenu, bez obzira na njihovu geografsku lokaciju. Iskorištavanjem robusnih alata za komunikaciju i suradnju poput Microsoft Teams, organizacije mogu prevladati geografske barijere, poboljšati produktivnost i potaknuti osjećaj jedinstva među svojim distribuiranim timovima.


**Slika 2.2.** *Microsoft Teams logo*

### 2.2.2. Webex

Webex je vodeća platforma za video konferencije i suradnju koja pojedincima i timovima omogućuje povezivanje, komunikaciju i suradnju na daljinu. Nudi niz značajki dizajniranih za olakšavanje učinkovitih virtualnih sastanaka, prezentacija i suradničkog rada, što ga čini popularnim izborom za male i velike organizacije. Webexova funkcionalnost daljinskog pristupa omogućuje korisnicima da se nesmetano povežu i surađuju s bilo koje lokacije, promičući produktivnost i učinkovitu komunikaciju. Neke od značajki Webexa su virtualni sastanci,

dijeljenje zaslona i sadržaja, alati za kolaboraciju, alati za snimanje i pregledavanje sastanaka te veliki naglasak na sigurnost i privatnost. [3]


**Slika 2.3.** *Webex logo*

### 2.2.3. TeamViewer

TeamViewer je široko korišten software za daljinski pristup i podršku koji korisnicima omogućuje povezivanje i kontrolu udaljenih računala ili uređaja s bilo kojeg mjesta na svijetu. To je svestran alat koji nudi niz značajki za olakšavanje daljinske suradnje, tehničke podrške i rješavanja problema. S TeamViewerom korisnici mogu sigurno pristupati udaljenim sustavima i upravljati njima, što ga čini popularnim izborom za pojedince, tvrtke i IT profesionalce. Neke od značajki TeamViewera su daljinski pristup računala, mogućnost dijeljenja podataka udaljenim računalima, mogućnost održavanja online sastanaka i prezentacija. Moguće je uspostaviti stalnu vezu s udaljenim računalom, čime se omogućuje daljinsko upravljanje bez potrebe za interakcijom korisnika na udaljenom računalu. TeamViewer je kompatibilan s raznim operacijskim sustavima, među kojima su Windows, macOs, Linux, iOS i android. Ta kompatibilnost omogućuje korisnicima da daljinski upravljaju računala čiji operacijski sustav se razlikuje od njihovog. [4]


**Slika 2.4.** *TeamViewer logo*

### 2.2.4. RealVNC

RealVNC je programsko rješenje za udaljenu radnu površinu koje korisnicima omogućuje daljinski pristup i upravljanje računalom s drugog uređaja. Korisnicima omogućuje daljinski pristup i upravljanje stolnim računalima, poslužiteljima i mobilnim uređajima s bilo kojeg mjesta u svijetu. RealVNC koristi protokol udaljenog međuspremnika okvira za prijenos radne površine udaljenog računala na lokalno računalo, omogućujući korisnicima interakciju s udaljenim računalom kao da sjede ispred njega. Softver nudi napredne sigurnosne značajke, kao što su enkripcija, provjera autentičnosti korisnika i prolazak kroz vatrozid, kako bi se osigurao siguran daljinski pristup. RealVNC je dostupan za Windows, macOS, Linux i Raspberry Pi platforme, kao i za iOS i Android mobilne uređaje. [5]

**Slika 2.6.** *RealVNC logo*

### 2.2.5. AnyDesk

AnyDesk je programsko rješenje za udaljenu radnu površinu koje korisnicima omogućuje siguran pristup i kontrolu udaljenog računala s drugog računala ili mobilnog uređaja. Omogućuje brz i pouzdan način daljinskog pristupa datotekama, programima i drugim resursima na udaljenom računalu kao da sjedite ispred njega. AnyDesk koristi napredne protokole kriptiranja i provjere autentičnosti kako bi osigurao da su sve udaljene veze sigurne i privatne. Koriste ga pojedinci i tvrtke za podršku na daljinu, rad na daljinu i suradnju na daljinu. AnyDesk je dostupan za Windows, macOS, Linux, Android i iOS. [6]

**Slika 2.5.** *AnyDesk logo*

# 3. ZAHTJEVI APLIKACIJE I KORIŠTENE TEHNOLOGIJE I ALATI

U narednom poglavlju opisani su zahtjevi aplikacije te su objašnjene tehnologije i alati koji su korišteni u izradi projektnog rješenja.

## 3.1. Zahtjevi na web aplikaciju

U ovoj web aplikaciji potrebno je implementirati tri vrste korisnika koji će koristiti ovu aplikaciju. Među te tri uloge nalaze se obični korisnik, moderator i administrator. Svaki korisnik mora imati svoju osobnu stranicu gdje može uređivati osobne podatke.

Sučelje za obične korisnike treba pružati mogućnost kreiranja zahtjeva za daljinsku pomoć. Zahtjevi za pomoć trebaju sadržavati opis problema koji su korisnici susreli. Običnom korisniku također je potrebno omogućiti prijavu na VNC predavanje.

Moderatorima je potrebno omogućiti sve što običan korisnik može napraviti. Sučelje za moderatore treba omogućiti prikaz svih zahtjeva za pomoć koju su korisnici napravili. Sučelje također treba omogućiti moderatoru daljinsko upravljanje računala korisnika koji je napravio zahtjev. Moderatorima je potrebno omogućiti kreiranje i održavanje VNC predavanja.

Sučelje za administratore treba omogućiti administrator korisnicima sva prava koja imaju obični korisnici i moderatori. Administratori trebaju moći kreirati zahtjev za daljinsku pomoć i pružiti daljinsku pomoć drugima. Administratoru je potrebno omogućiti brisanje korisnika, brisanje bilo kojeg zahtjeva za pomoć ili predavanje, te upravljanje korisnicima i korisničkim ulogama.

## 3.2. Pregled korištenih tehnologija

U narednom dijelu opisane su tehnologije korištene pri izradi programskog rješenje.

### 3.2.1. .NET Framework

.NET Framework je besplatan programski okvir, otvorenog koda (engl. *Open-source*), dizajniran za razvijanje dinamičkih web stranica, usluga i aplikacija. Razvijen je i održavan od strane Microsofta kao dio .NET platforme.

U svojoj srži, .NET omogućuje programerima da razvijaju robusne, skalabilne i sigurne web aplikacije koristeći jezike koji su u skladu s *Common Language Infrastructure*, poput C# i Visual Basic. Programski okvir za razvijanje web aplikacija pruža model koji sadrži sve servise potrebne

za razvoj poslovnih aplikacija. Ovaj model sastoji se od niza kontrola, protokola i biblioteka koje omogućuju obradu web zahtjeva, upravljanje informacijama o stanju, povezivanje s bazom podataka i mnogi drugi. [7]

Dvije ključne komponente .NET Frameworka su web forme i MVC (engl. *Model-View-Controller*). Web forme nude model vođen događajima i vizualni dizajner, dok MVC programski okvir pruža način izgradnje dinamičkih web aplikacija temeljen na obrascima koji nudi veću kontrolu nad HTML-om, JavaScriptom i CSS-om naspram tradicionalnih web formi.

ASP.NET također pruža ključne značajke kao što su autentikacija korisnika, autorizacija korisnika i njihovih radnji te upravljanje stanjem, koje čuva podatke za više korisničkih zahtjeva. .NET Framework podržava i uobičajene standarde kao što su HTTP i HTTPS čime omogućuje osiguravanje sigurnosti web aplikacije.

### 3.2.2. Entity Framework

Entity Framework je moćan programski okvir za objektno-relacijsko mapiranje (engl. *Object-Relational Mapping, ORM*) koji je razvio Microsoft. Služi kao sloj apstrakcije između relacijskih baza podataka i objektno orijentiranih programskih modela u .NET aplikacijama. Entity Framework značajno poboljšava produktivnost razvoja, upravljanje podacima i održavanje aplikacije automatiziranjem koda za pristup podacima koje programeri obično trebaju napisati. [8]

Ideja iza Entity Frameworka je omogućavanje programerima rad s podacima u smislu objekata i svojstava specifičnih za domenu, kao što su kupac i adresa kupca, bez potrebe da se brinu o tablicama i stupcima u kojima su podaci spremljeni u bazi podataka. Entity Framework se brine o povezivanju s bazom podataka, izvršavanjem naredbi i pretvaranjem rezultata u objekte kojima se može manipulirati u kodu te tako pojednostavljuje proces razvijanja aplikacije.

Entity Framework podržava tri pristupa modeliranja sheme baze podataka:

1. *Database-First* pristup koristi postojeću bazu podataka te na temelju nje generira EDM (engl. *Entity Data Model*)
2. *Model-First* pristup dozvoljava programerima da razviju EDM na temelju kojih će se generirati shema baze podataka

3. *Code-First*, što je ujedno i pristup je odabran u ovome projektu, je pristup koji pruža programerima mogućnost da razviju klase domene na temelju kojih se generira baza podataka.

Entity Framework također podržava pisanje upita bazi podataka koristeći .NET jezike pomoću LINQ (engl. *Language Integrated Query*) umjesto tradicionalnog SQL-a. LINQ osigurava sigurnost tipa podataka, smanjuje broj grešaka tijekom razvijanja aplikacije te pruža IntelliSense podršku u Visual Studio razvojnom okruženju.

### 3.2.3. TightVNC Server

TightVNC Server je software otvorenog koda koji omogućuje udaljeni pristup i kontrolu radne površine putem interneta. TightVNC Server je poslužiteljska komponenta VNC tehnologije koja radi na principu RFB protokola. [9]

TightVNC Server robustan je alat s jednostavnim korisničkim sučeljem. Podržava istovremeno dijeljenje zaslona prema više VNC preglednika, postavljanje zaporke za uspostavljanje veze za udaljeni pristup te pruža opciju dvosmjernog prijenosa međuspremnika.

### 3.2.4. noVNC

noVNC je JavaScript biblioteka otvorenog koda koja nudi VNC klijent visokih performansi. Zahvaljujući uporabi HTML5 WebSocketa, Canvasa i lokalnog spremišta, noVNC nudi laku implementaciju VNC klijenta unutar modernih web preglednika.

Jedna od ključnih značajki noVNC klijenta je njegova neovisnost o operacijskom sustavu. Kako se pokreće unutar web preglednika, moguće ga je koristiti na bilo kojem uređaju koji podržava moderne web preglednike. noVNC također podržava i kriptirane WebSocket veze (wss://) te tako omogućuje sigurnu vezu s VNC poslužiteljem. [10]

noVNC se spaja s VNC poslužiteljem koristeći RFB protokol što mu omogućuje daljinsko upravljanje. Kako VNC poslužitelji koriste TCP vezu, noVNC je potrebno spojiti s *proxy* poslužiteljem koji će služiti kao prevoditelj između TCP i WebSocket protokola.

### 3.2.5. WebSockify

WebSockify je ključan alat pri implementaciji WebSocket funkcionalnosti unutar aplikacija koje zahtijevaju *real-time* dvosmjernu komunikaciju između klijenta i poslužitelja. Razvijen kao dio noVNC projekta, WebSockify je most između WebSocket i TCP protokola koji dopušta web aplikacijama da se spoje na bilo koji servis, aplikaciju ili poslužitelj koji podržava TCP protokol.

WebSockify radi na temelju osluškivanja WebSocket veza te uspostavom TCP veze s ciljanim klijentom. Nakon uspostave TCP veze, WebSockify dvosmjerno šalje podatke između WebSocket i TCP veze, te tako služi kao most među njima.

WebSockify podržava kriptirane WebSocket veze, te tako pruža mogućnost uspostave sigurne veze bez mogućnosti prisluškivanja ili petljanja s podacima. Prvobitno razvijen pomoću Python programskog jezika, sada podržava razne jezike poput C, Node.Js, Ruby. Ovaj veliki raspon programskih jezika omogućuje relativno laku implementaciju u raznim razvojnim okruženjima. [11]

### 3.2.6. Unity Container

Unity Container je lagani, proširivi spremnik (engl. *Container*) korišten za ubrizgavanje ovisnosti (engl. *Dependency injection*) koji pomaže pri upravljanju stvaranja zavisnih objekata. Obično se koristi u .NET aplikacijama radi postizanja inverzije kontrole (engl. *Inversion of control*) koristeći tehniku *dependency injection*. Unity container odgovoran je za kreiranje instanci objekata registriranih tipova i upravljanjem njihovog životnog vijeka, čime ostvaruje fleksibilnost, labavo povezani (engl. *Loose couple*) kod koji je lako testirati i održavati.

Za razliku od klasičnog pristupa gdje klase instanciraju objekte klasa o kojima ovise, Unity Container omogućava deklariranje ovisnosti unutar konstruktora, svojstva ili metode klase. Unity Container zatim ubrizgava instance zavisnih klasa za vrijeme izvođenja, čime omogućava fleksibilniji kod koji je lako testirati.

# 4. PROGRAMSKO RJEŠENJE WEB APLIKACIJE

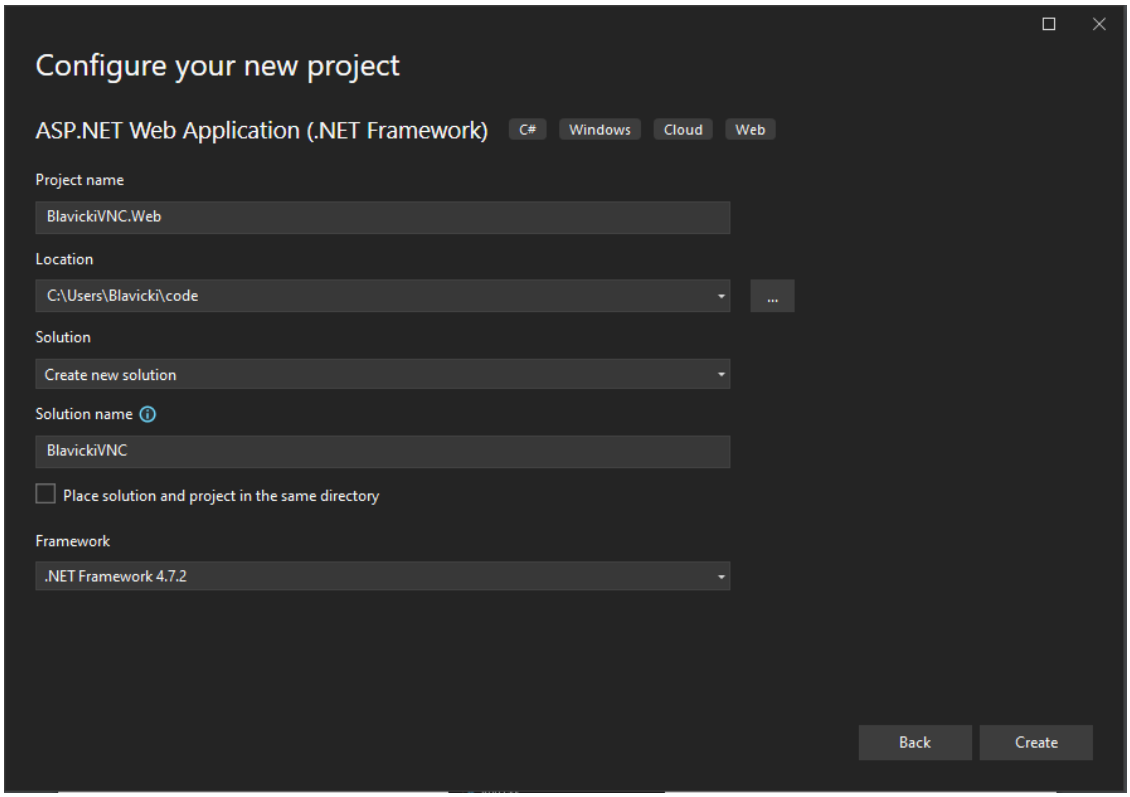
U narednom poglavlju opisano je programsko rješenje aplikacije. Opisano je postavljanje razvojne okoline projekta, postojeći servisi i repozitoriji te svi kontroleri i njihove funkcionalnosti.

## 4.1. Postavljanje razvojne okoline

U narednom dijelu opisan je postupak postavljanja razvojne okoline projekta i svih alata korištenih u izradu projekta.

### 4.1.1. Stvaranje projekta

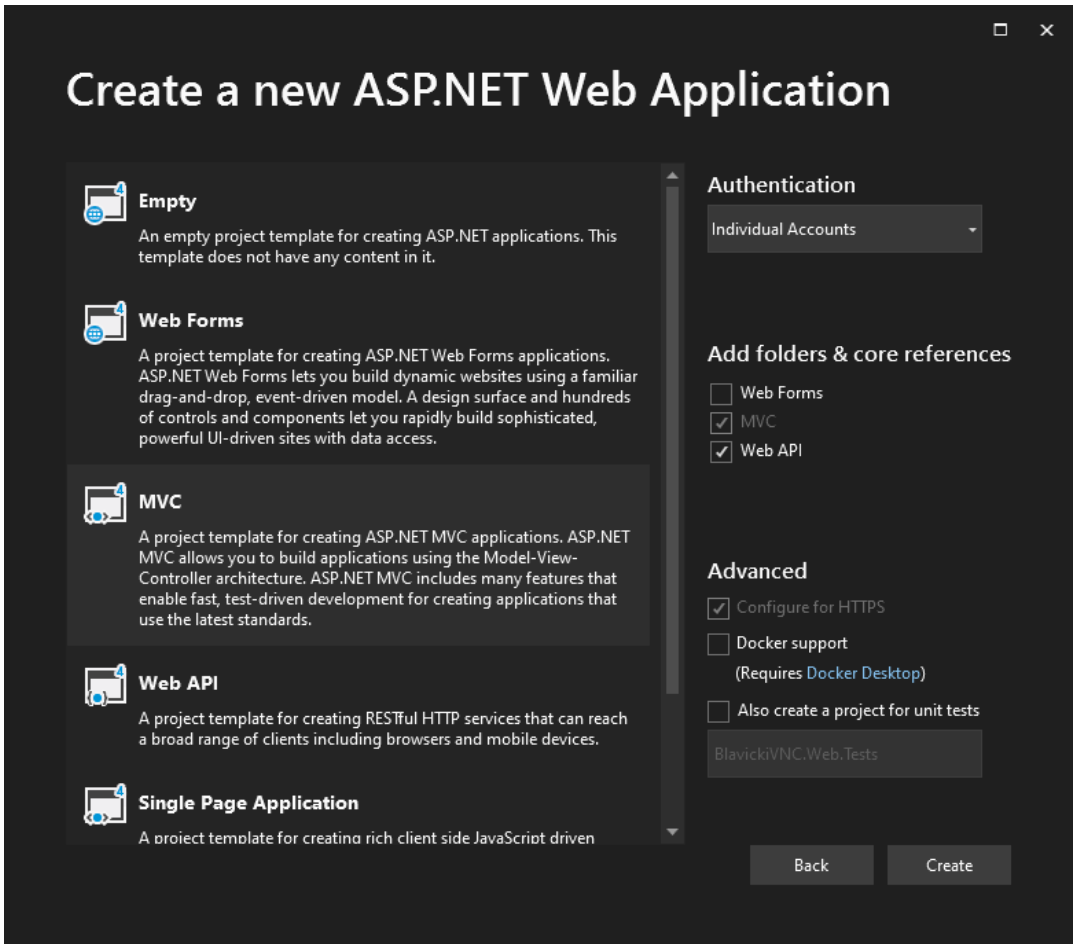
Projekt je kreiran unutar Visual Studio 2022 aplikacije kao ASP.NET Web aplikacija koja koristi .NET Framework verzije 4.7.2. Ime projekta i rješenja (engl. *solution*) nazvano je BlavickiVNC kao što je vidljivo na slici 4.1..




**Slika 4.1.** *Konfiguracija projekta*

Prilikom odabira kakva će ASP.NET Web aplikacija ovaj projekt biti, važno je istaknuti da je odabran MVC template projekta. Za odabir autentikacije potrebno je odabrati "*Individual Accounts*" kako bi se koristila ugrađena Identity autentikacija. Također, potrebno je odabrati i Web

API pod "*Add folders & core references*" dijelom. Navedene postavke mogu se vidjeti na slici 4.2..




**Slika 4.2.** *Kreiranje projekta*

Odabiranje navedene konfiguracije kreirat će projekt s MVC strukturom. Kreirani projekt sadrži osnovne pakete među kojima su EntityFramework, Identity autentikacija, Razor te drugi potrebni paketi za rad web aplikacije.

### 4.1.2. Postavljanje baze podataka

U ovome projektu odabrana je MySQL baza podataka. Kako bi se u projektu mogla implementirati MySQL baza podataka, potrebno je instalirati dva paketa, "*MySql.Data*" i "*MySql.Data.EntityFramework*". Instalacija ova dva paketa nudi alate potrebne za uspostavu rada između EntityFrameworka i MySQL baze podataka. Kako bi se mogla povezati aplikacija s bazom podataka, unutar aplikacije potrebno je pohraniti niz za vezu (engl. *Connection String*). *Connection string* spremljen je unutar "Secrets" direktorija. Preporučeno je da se "Secrets"

direkotorij uključi u *.gitignore* datoteku kako bi se spriječilo objavljivanje tajni korištenih u aplikacije na internetu.

Kako bi se omogućio rad Entity Frameworka s MySQL bazom podataka, kreiran je direktorij Context. Unutar *Context* direktorija definirane su klase koje omogućuju uspostavu sesije s bazom podataka. Najvažnija klasa među njima je *BlavickiVNCContex*. Instanca klase *BlavickiVNCContext* predstavlja jednu sesiju između baze podataka i aplikacije. Druge klase poput *DbContextCollection*, *DbContextScope* i *DbContextService* pružaju dodatne funkcionalnosti za lakše kreiranje i upravljanje instancom konteksta.

*DbContext* klasa je središnja komponenta Entity Frameworka koja predstavlja sesiju s bazom podataka. Njena odgovornost je pratiti promjene koje su napravljene na tijelima (engl. *Entities*) te da te promjene spremi unutar baze podataka.

*IdentityDbContext* klasa nasljeđuje *DbContext* klasu te pruža sav kod potreban za *code-first* pristup mapiranja baze podataka te postavljene potrebne vrijednosti svojstva *DbSet* za upravljanje tablica unutar baze podataka potrebnih za sustav autentikacije.

Prilikom kreiranja projekta, .NET Framework automatski kreira *ApplicationDbContext* klasu smještenu unutar *IdentityModels* datoteke. Unutar ovog projekta, ta klasa je preimenovana u *BlavickiVNCContext* te premještena u *Context* direktorij. Klasa *BlavickiVNCContext* središnja je komponenta koja spaja bazu podataka s Identity autentikacijom i Entity Frameworkom.

*BlavickiVNCContext* klasa sadrži *OnModelCreating* metodu koju je moguće prepisati. Prepisivanjem metode možemo postići da implicitno naredimo kako će izgledati baza podataka. Identity sustav nudi automatsko kreiranje tablica, no imena tablica sadrže “aspnet” prefiks. Radi jednostavnosti imenovanja, u ovome projektu implicitno su zadana imena tablica za pohranjivanje modela koje koristi Identity sustav bez prefiksa, kao što je vidljivo na slici 4.3..

BlavickiVNC - BlavickiVNCContext.cs

```
12  public class BlavickiVNCContext : IdentityDbContext<ApplicationUser>
13      {
14          public BlavickiVNCContext() : base("BlavickiVNCContext", throwIfV1Schema: false) { }
15
16          public DbSet<VncRequest> Requests { get; set; }
17          public DbSet<Lecture> Lectures { get; set; }
18          public DbSet<LectureApplication> LectureApplications { get; set; }
19
20          public static BlavickiVNCContext GetContext()
21          {
22              {
23                  return new BlavickiVNCContext();
24              }
25          }
26
27          protected override void OnModelCreating(DbModelBuilder modelBuilder)
28          {
29
30              base.OnModelCreating(modelBuilder);
31
32              modelBuilder.Entity<ApplicationUser>().ToTable("Users");
33              modelBuilder.Entity<IdentityRole>().ToTable("Roles");
34              modelBuilder.Entity<IdentityUserRole>().ToTable("UserRoles");
35              modelBuilder.Entity<IdentityUserClaim>().ToTable("UserClaims");
36              modelBuilder.Entity<IdentityUserLogin>().ToTable("UserLogins");
37
38              modelBuilder.Entity<Lecture>()
39                  .HasMany(lecture => lecture.Applications);
40
41              modelBuilder.Entity<VncRequest>()
42                  .HasRequired(v => v.User)
43                  .WithMany()
44                  .HasForeignKey(v => v.UserId)
45                  .WillCascadeOnDelete(true);
46
47              modelBuilder.Entity<Lecture>()
48                  .HasRequired(l => l.User)
49                  .WithMany()
50                  .HasForeignKey(l => l.UserId)
51                  .WillCascadeOnDelete(true);
52
53              modelBuilder.Entity<LectureApplication>()
54                  .HasRequired(la => la.User)
55                  .WithMany()
56                  .HasForeignKey(la => la.UserId)
57                  .WillCascadeOnDelete(true);
58
59              modelBuilder.Entity<LectureApplication>()
60                  .HasRequired(la => la.Lecture)
61                  .WithMany(l => l.Applications)
62                  .HasForeignKey(la => la.LectureId)
63                  .WillCascadeOnDelete(true);
64
65          }
66      }
```

**Slika 4.3.** *BlavickiVNCContext klasa*

### 4.1.3. Entity Framework migracije

Kako bi se omogućile Entity Framework migracije, potrebno je otvoriti *Package Manager Console* te unijeti naredbu *Enable-Migrations*. Pozivanjem naredbe *Enable-Migrations*, Entity Framework kreira *Migrations* direktorij sa *Configuration.cs* datotekom. Datoteka *Configuration.cs* sadrži *Configuration* klasu koja je tipa *DbMigrationsConfiguration*. Unutar *Configuration* klase može se odabrati koji generator koda migracije te SQL generator koji će se koristiti prilikom kreiranja migracija. *Configuration* klasa također sadrži i *Seed* metodu koju je moguće prepisati (engl. *Override*). Prepisivanjem *Seed* metode moguće je ubrizgati podatke prilikom kreiranja baze podataka.

Prije nego što se ažurira baza podataka na temelju generirane migracije, prvo je potrebno napisati *Seed* metodu koja će osigurati da se u bazi podataka nalaze podaci koji su potrebni za ispravan rad. Kao što je vidljivo na slici 4.4., unutar *Seed* metode *Configuration* klase prvo se kreiraju korisničke uloge "Admin" i "Moderator". Nakon kreiranja korisničkih uloga, moguće je kreirati korisnika s imenom "admin@example.com", te kreiranom korisniku pružiti ulogu "Admin". Ažuriranje baze podataka s napisanom *Seed* metodom osigurat će da se ti podaci nalaze unutar baze podataka nakon ažuriranja baze.

BlavickiVNC - Configuration.cs

```
13  internal sealed class Configuration : DbMigrationsConfiguration<BlavickiVNCContext>
14  {
15      public Configuration()
16      {
17          AutomaticMigrationsEnabled = false;
18          SetSqlGenerator("MySql.Data.MySqlClient", new MySqlGenerator());
19          CodeGenerator = new MySqlMigrationCodeGenerator();
20      }
21
22      protected override void Seed(BlavickiVNCContext context)
23      {
24          var userManager = new UserManager<ApplicationUser>(new UserStore<ApplicationUser>(context));
25          var roleManager = new RoleManager<IdentityRole>(new RoleStore<IdentityRole>(context));
26
27          const string name = "admin@example.com";
28          const string password = "Admin@123";
29          const string adminRoleName = "Admin";
30          const string moderatorRoleName = "Moderator";
31
32          var adminRole = roleManager.FindByName(adminRoleName);
33          if (adminRole == null)
34          {
35              adminRole = new IdentityRole(adminRoleName);
36              var roleresult = roleManager.Create(adminRole);
37          }
38
39          var moderatorRole = roleManager.FindByName(moderatorRoleName);
40          if (moderatorRole == null)
41          {
42              moderatorRole = new IdentityRole(moderatorRoleName);
43              var roleresult = roleManager.Create(moderatorRole);
44          }
45
46          var user = userManager.FindByName(name);
47          if (user == null)
48          {
49              user = new ApplicationUser { UserName = "admin", Email = name };
50              var result = userManager.Create(user, password);
51              result = userManager.SetLockoutEnabled(user.Id, false);
52          }
53
54          var rolesForUser = userManager.GetRoles(user.Id);
55          if (!rolesForUser.Contains(adminRole.Name))
56          {
57              var result = userManager.AddToRole(user.Id, adminRole.Name);
58          }
59      }
60  }
```

**Slika 4.4.** *Configuration klasa*

Kako bi se kreirala početna migracija, potrebno je ponovno otići u *Package Manager Console* te unijeti naredbu *Add-Migration Base*. Naredba *Add-Migration Base* kreira migraciju s imenom *Base* unutar *Migrations* direktorija. Entity Framework kao prefiks imena migracije stavlja vremenska oznaku trenutka stvaranja migracije kako bi kasnije znao kojim redoslijedom su migracije bile generirane. Zahvaljujući *IdentityDbContext* klasi koju je naslijedila

*BlavickiVNCContext* klasa, u datoteci migracije generiran je kod koji je odgovoran za stvaranje tablica potrebnih za rad Identity autentikacije. Kreirane tablice su:

- Users – sadrži podatke o korisniku kao što su *ID*, *username*, *email*, hash vrijednost lozinke itd.
- Roles – sadrži sve uloge definirane u sustavu i njih vrijednosti koje ih opisuju poput *ID* i imena
- UserRoles – tablica koja spaja *Users* i *Roles* tablice te sadrži podatke o korisnicima i njihovim ulogama
- UserClaims - tablica koja sadrži korisnikova prava koja su dodijeljena korisniku
- UserLogins tablice – sadrži korisnikove podatke za prijavu pomoću eksternih alata poput Google, Facebook itd.

Nakon napisane logike za *Seed* metodu i generirane migracije, moguće je ažurirati bazu podataka. Da bi se baza podataka ažurirala, potrebno je ponovno otvoriti *Package Manager Console* te unijeti naredbu *Update-Database*. *Update-Database* metoda prolazi kroz sve generirane migracije unutar *Migrations* direktorija te ih uspoređuje sa *__MigrationHistory* tablicom unutar baze podataka kako bi se ustanovilo koje migracije nisu primijenjene na bazi podataka. Entity Framework, nakon što utvrdi koje migracije nisu primijenjene, primjenjuje migracije na bazu podataka onim redoslijedom kojim su bile generirane. Tablicu *__MigrationHistory* nakon prve, *Base* migracije može se vidjeti na slici 4.5..

| | MigrationId | ContextKey | Model | ProductVersion |
|---|---|---|---|---|
| ▶ | 202306231133258_Base | BlavickiVNC.Web.Migrations.Configuration | BLOB | 6.4.4 |
| * | NULL | NULL | NULL | NULL |

**Slika 4.5.** *__MigrationHistory tablica*

### 4.1.4. Identity autentikacija

Odabir "*Individual Accounts*" autentikacije tijekom kreiranja projekta generiralo je osnovnu strukturu potrebnu za logiku autentikacije. Među generiranim datotekama nalaze se *Startup.Auth.cs* i *IdentityConfig.cs* datoteke koje su smještene unutar *App_Start* direktorija.

za konfiguriranje ASP.NET Identity sustava za autentikaciju unutar aplikacije korištena je *IdentityConfig.cs* datoteka. Identity nudi niz klasa koje je moguće naslijediti kako bi se upravljalo sustavom. Implementacije tih klasa nalaze se unutar *IdentityConfig.cs* datoteke.

Središnja klasa Identity autentikacije koju je potrebno implementirati zove se *UserManager* klasa. *UserManager* klasa korištena je za upravljanje korisnicima unutar aplikacije te nam nudi mogućnost namještanja postavki Identity autentikacije. Prilikom nasljeđivanja potrebno je pružiti tip podatka koji predstavlja korisnika kako bi Identity znao s kojim tipom će raditi. Kao što je vidljivo na slici 4.6, definirana je *ApplicationUserManager* klasa koja nasljeđuje pruženu *UserManager* klasu. Unutar klase definirana su ograničenja za ime i lozinku korisnika, omogućena je funkcionalnost ključanja korisnikovog računa nakon 5 neuspješnih prijava, definirani su pružatelji *two-factor* autentikacije te servisi poput *EmailService* i *SMSService* koji su potrebni za pravilan rad sustava.

BlavickiVNC - IdentityConfig.cs

```
83  public class ApplicationUserManager : UserManager<ApplicationUser>
84  {
85      public ApplicationUserManager(IUserStore<ApplicationUser> store)
86          : base(store)
87      {
88      }
89
90      public static ApplicationUserManager Create(IdentityFactoryOptions<ApplicationUserManager> options, IOwinContext context)
91      {
92          var manager = new ApplicationUserManager(new UserStore<ApplicationUser>(context.Get<BlavickiVNCContext>()));
93
94          manager.UserValidator = new UserValidator<ApplicationUser>(manager)
95          {
96              AllowOnlyAlphanumericUserNames = false,
97              RequireUniqueEmail = true,
98          };
99
100         manager.PasswordValidator = new PasswordValidator
101         {
102             RequiredLength = 6,
103             RequireNonLetterOrDigit = true,
104             RequireDigit = true,
105             RequireLowercase = true,
106             RequireUppercase = true,
107         };
108
109         manager.UserLockoutEnabledByDefault = true;
110         manager.DefaultAccountLockoutTimeSpan = TimeSpan.FromMinutes(5);
111         manager.MaxFailedAccessAttemptsBeforeLockout = 5;
112
113         manager.RegisterTwoFactorProvider("Phone Code", new PhoneNumberTokenProvider<ApplicationUser>
114         {
115             MessageFormat = "Your security code is {0}"
116         });
117         manager.RegisterTwoFactorProvider("Email Code", new EmailTokenProvider<ApplicationUser>
118         {
119             Subject = "Security Code",
120             BodyFormat = "Your security code is {0}"
121         });
122         manager.EmailService = new EmailService();
123         manager.SmsService = new SmsService();
124         var dataProtectionProvider = options.DataProtectionProvider;
125         if (dataProtectionProvider != null)
126         {
127             manager.UserTokenProvider =
128                 new DataProtectorTokenProvider<ApplicationUser>(dataProtectionProvider.Create("ASP.NET Identity"));
129         }
130         return manager;
131     }
132 }
```

**Slika 4.6.** *ApplicationUserManager klasa*

Za rad s korisničkim ulogama, Identity nudi *RoleManager* klasu. Kao i kod *UserManager* klase, *RoleManager* klasu potrebno je implementirati te pružiti tip podatka koji predstavlja korisničku ulogu unutar aplikacije. *RoleManager* klasa nudi nam mogućnost upravljanja ulogama. Za upravljanje ulogama implementirana je *ApplicationRoleManager* klasa koja nasljeđuje *RoleManager* klasu.

Klasa koju nudi Identity i koja je odgovorna za prijavu i odjavu korisnika unutar aplikacije naziva se *SignInManager*. Poput *UserManager* klase, za ispravan rad, *SignInManager* klasi potrebno je pružiti tip podatka koji predstavlja korisnika. Unutar ove aplikacije, *UserManager* klasa implementirana je preko *ApplicationSignInManager* klase.

Za kreiranje servisa za slanje poruka, Identity nudi *IIdentityMessageService* sučelje (engl. *Interface*). Implementaciju *EmailService* i *SmsService* klase može se vidjeti na slici 4.7..

BlavickiVNC - IdentityConfig.cs

```
23  public class EmailService : IIdentityMessageService
24  {
25      private const string BlavickiVncEmail = "blavickivnc@gmail.com";
26      private const string BlavickiVncName = "BlavickiVNC";
27
28      public Task SendAsync(IdentityMessage message)
29      {
30          var fromAddress = new MailAddress("sandro.blavicki@gmail.com", BlavickiVncName);
31          var toAddress = new MailAddress(message.Destination, "BlavickiVNC user");
32          const string fromPassword = "rboxcilypyndpmmi";
33          string subject = message.Subject;
34          string body = message.Body;
35
36          var smtp = new SmtpClient
37          {
38              Host = "smtp.gmail.com",
39              Port = 587,
40              EnableSsl = true,
41              DeliveryMethod = SmtpDeliveryMethod.Network,
42              UseDefaultCredentials = false,
43              Credentials = new NetworkCredential(fromAddress.Address, fromPassword)
44          };
45
46          using (var emailMessage = new MailMessage(fromAddress, toAddress)
47          {
48              Subject = subject,
49              Body = body,
50              IsBodyHtml = true,
51          })
52          {
53              smtp.Send(emailMessage);
54          }
55
56          return Task.FromResult(0);
57      }
58  }
59
60  public class SmsService : IIdentityMessageService
61  {
62      private const string AccountSid = "ACfe3cb5f5062f640228ed3691ed2a93d6";
63      private const string AuthToken = "2460fae7aec0eba2becb64815a22be96";
64      private const string BlavickiVNCNumber = "+17063883639";
65
66      public SmsService()
67      {
68          TwilioClient.Init(AccountSid, AuthToken);
69      }
70
71      public async Task SendAsync(IdentityMessage message)
72      {
73          try
74          {
75              await MessageResource.CreateAsync(
76                  new PhoneNumber(message.Destination),
77                  from: new PhoneNumber(BlavickiVNCNumber),
78                  body: message.Body);
79          } catch(Exception error)
80          {
81              return;
82          }
83      }
84  }
```

**Slika 4.7.** *EmailService i SmsService*

*Startup.Auth.cs* datoteka korištena je za konfiguriranje autentikacijske međuprogramske opreme (engl. *Middleware*) unutar ASP.NET aplikacije. Unutar datoteke definirana je *ConfigureAuth* metoda pomoću koje se definira postavke autentikacije za aplikaciju. *ConfigureAuth* metoda pruža mogućnost brojnih postavki poput definiranja tipa autentikacije, vanjskih alata za prijavu (Google, Facebook), korištenje *cookie* funkcionalnosti i drugih. Postavke korištene unutar ove aplikacije vidljive su na slici 4.8..

BlavickiVNC - Startup.Auth.cs

```
16  public void ConfigureAuth(IAppBuilder app)
17  {
18      app.CreatePerOwinContext(BlavickiVNCContext.GetContext);
19      app.CreatePerOwinContext<ApplicationUserManager>(ApplicationUserManager.Create);
20      app.CreatePerOwinContext<ApplicationSignInManager>(ApplicationSignInManager.Create);
21      app.CreatePerOwinContext<ApplicationRoleManager>(ApplicationRoleManager.Create);
22
23      app.UseCookieAuthentication(new CookieAuthenticationOptions
24      {
25          AuthenticationType = DefaultAuthenticationTypes.ApplicationCookie,
26          LoginPath = new PathString("/Account/Login"),
27          Provider = new CookieAuthenticationProvider
28          {
29              OnValidateIdentity = SecurityStampValidator.OnValidateIdentity<ApplicationUserManager, ApplicationUser>(
30                  validateInterval: TimeSpan.FromMinutes(30),
31                  regenerateIdentity: (manager, user) => user.GenerateUserIdentityAsync(manager))
32          }
33      });
34      app.UseExternalSignInCookie(DefaultAuthenticationTypes.ExternalCookie);
35
36      app.UseTwoFactorSignInCookie(DefaultAuthenticationTypes.TwoFactorCookie, TimeSpan.FromMinutes(5));
37
38      app.UseTwoFactorRememberBrowserCookie(DefaultAuthenticationTypes.TwoFactorRememberBrowserCookie);
39
40      app.UseFacebookAuthentication( new FacebookAuthenticationOptions
41      {
42         AppId= "571061021888822",
43         AppSecret = "3ee0acd6343b8218141237bea2d2b40d",
44         SignInAsAuthenticationType = DefaultAuthenticationTypes.ExternalCookie,
45         CallbackPath = new PathString("/signin-facebook")
46
47      });
48
49      app.UseGoogleAuthentication(new GoogleOAuth2AuthenticationOptions()
50      {
51          ClientId = "1091185998444-e9qp6hkep0mg9n4j57teve9qed5fa6dq.apps.googleusercontent.com",
52          ClientSecret = "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"
53      });
54  }
```

**Slika 4.8.** *Startup.Auth ConfigureAuth metoda*

### 4.1.5. NPM

Visual studio nudi automatsku potporu za NPM (engl. *Node package manager*). Kako bi se unutar aplikacije koristio NPM, potrebno je kreirati *package.json* datoteku u korijenskom direktoriju projekta.. Za potrebe ovoga projekta potrebno je koristiti „@novnc/novnc“ paket.

Nakon unošenja ovisnosti koje su korištene u aplikaciji, koristeći komandnu liniju, potrebno je pozvati naredbu *npm install*. Naredba *npm install* kreirat će *node_modules* direktorij te dohvatiti sve ovisnosti korištene u projektu.

### 4.1.6. WebSockify

WebSockify je neovisan projekt koji je potrebno koristiti unutar aplikacije radi uspostave komunikacije između VNC poslužitelja i aplikacije. Kako bi se WebSockify koristio, unutar

projekta stvoren je *Tools* direktorij. WebSockify potrebno je skinuti sa službene Github stranice kao ZIP datoteku te otpakirati unutar *Tools* direktorija pod imenom websockify-js. Raspakirani websockify-js direktorij unutar sebe sadrži websockify direktorij do kojeg je potrebno navigirati koristeći komandnu liniju. Unutar websockify direktorija, potrebno je pozvati naredbu „*npm install*“ koja će instalirati sve ovisnosti potrebne za rad websockify projekta. Za pokretanje websockify *proxy* poslužitelja implementirana je skripta unutar *package.json* datoteke, kao što je vidljivo na slici 4.9..

BlavickiVNC - package.json

```
{
  "version": "1.0.0",
  "name": "blavicki-vnc",
  "scripts": {
    "websockify": "node Tools/websockify-js/websockify/websockify.js"
  },
  "private": true,
  "dependencies": {
    "@novnc/novnc": "^1.4.0"
  }
}
```

**Slika 4.9.** *Datoteka package.json*

Kako bi se poslužitelj pokrenuo, potrebno je pozvati naredbu „*npm run websockify*“ sa zadanim adresama. Kao što je vidljivo na slici 4.10., za svrhu ovog projekta pokrenut je websockify poslužitelj koji s aplikacije prima WebSocket podatke na adresu localhost:6080 te ih prevodi i prosljeđuje VNC poslužitelju na adresu localhost:5901.

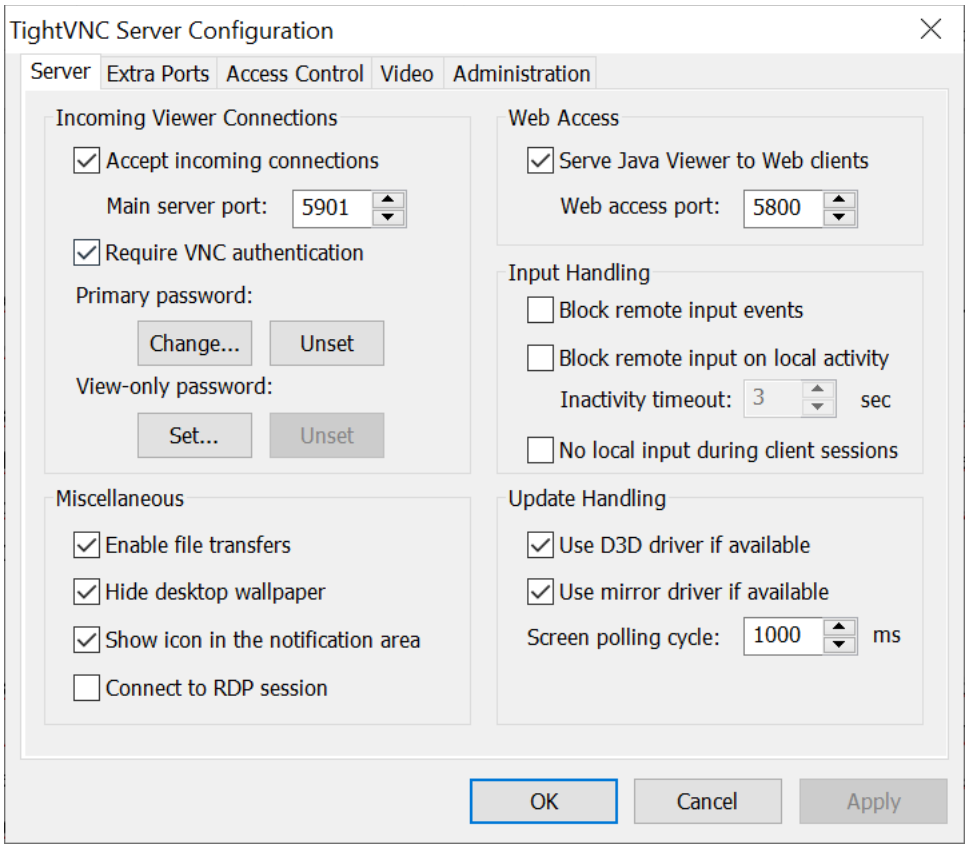
```
PS C:\Users\Blavicki\code\FinalDIplomski\BlavickiVNC\BlavickiVNC> npm run websockify localhost:6080 localhost:5901

> blavicki-vnc@1.0.0 websockify
> node Tools/websockify-js/websockify/websockify.js localhost:6080 localhost:5901

WebSocket settings:
    - proxying from localhost:6080 to localhost:5901
    - Running in unencrypted HTTP (ws://) mode
```

**Slika 4.10.** *Pokretanje WebSockify proxy poslužitelja*

### 4.1.7. TightVNC Server

Kako bi se omogućilo daljinsko upravljanje, potrebno je koristiti VNC poslužitelj koji će biti instaliran na računalu koje se želi upravljati. Za potrebe ovog projekta, instaliran je TightVNC Server aplikacija. Prilikom instalacije TightVNC aplikacije, ponudit će se izbornik za unos lozinke kako bi se zaštitilo pristupanje i kontroliranje udaljenog računala. Kao što je vidljivo na slici 4.11.,

TightVNC Server nudi jednostavno grafičko sučelje. Adresa na koju TightVNC Server prima podatke od VNC Viewer aplikacije je 5901.




**Slika 4.11.** *TightVNC Server*

### 4.1.8. Unity Container

Prilikom instalacije potrebnih paketa za korištenje Unity Container-a, unutar *App_Start* direktorija kreira se *UnityConfig.cs* datoteka. Unutar navedene datoteke potrebno je definirati sve tipove klasa koje će Unity Container ubrizgavati pomoću *RegisterTypes* metode. Konačan izgled *RegisterTypes* metode može se vidjeti na slici 4.12..

BlavickiVNC - UnityConfig.cs

```
43      public static void RegisterTypes(IUnityContainer container)
44      {
45
46          var clientHandler = new HttpClientHandler {
47                  UseDefaultCredentials = true,
48                  Proxy = null,
49                  UseProxy = false,
50                  UseCookies = false
51              };
52
53          _client = new HttpClient(clientHandler);
54
55          ServicePointManager.DefaultConnectionLimit = 100;
56          ServicePointManager.SecurityProtocol |=
57              SecurityProtocolType.Ssl3
58              | SecurityProtocolType.Tls12
59              | SecurityProtocolType.Tls11
60              | SecurityProtocolType.Tls;
61
62          // Instances
63          container.RegisterInstance(typeof(BlavickiVNCContext), new BlavickiVNCContext());
64          container.RegisterInstance(typeof(HttpClient), _client);
65
66          // Data Services
67          container.RegisterType<IDbContextService, DbContextService>();
68
69          // Services
70          container.RegisterType<IEncryptionService, EncryptionService>();
71          container.RegisterType<IRequestService, RequestService>();
72          container.RegisterType<ILectureService, LectureService>();
73          container.RegisterType<ILectureApplicationService, LectureApplicationService>();
74
75          // Repositories
76          container.RegisterType<IRequestRepository, RequestRepository>();
77          container.RegisterType<ILectureRepository, LectureRepository>();
78          container.RegisterType<ILectureApplicationRepository, LectureApplicationRepository>();
79      }
80  }
```

**Slika 4.12.** *Unity Container RegisterTypes metoda*

## 4.2. Repozitoriji

Za razvijanje projekta korištena je repozitorijska arhitektura. Obrazac repozitorija (enlg. *Repository pattern*) je obrazac koji se odnosi na arhitekturu projekta i podrazumijeva odvajanje logike za dohvaćanje podataka iz baze podataka od ostatka aplikacije. Korištenjem obrasca repozitorija postiže se čisto odvajanje problema među komponentama, te kod postaje lakši za održavanje, testiranje te ponovnu uporabu.

Glavne komponente obrasca repozitorija su sučelje, konkretna implementacija repozitorija koja implementira sučelje repozitorija, modela koji predstavljaju strukturu podataka unutar baze

podataka te ostatak aplikacije koji koristi repozitorije. Zahvaljujući sučeljima, moguće je koristiti Unity Container kako bi se, pomoću *dependency injection*, instance repozitorija ubrizgavale gdje su potrebne.

Kako svaki repozitorij sadrži logiku za kreiranje, ažuriranje, brisanje i dohvaćanje podataka pomoću identifikacije, za potrebe projekta razvijena je osnovna (engl. *Base*) klasa repozitorija koja pruža zajedničku logiku. Svaka konkretna implementacija repozitorija za određene tipove nasljeđuje osnovnu klasu repozitorija te se time postiže da nije potrebno duplicirati kod među repozitorijima. U konkretnim implementacijama repozitorija nalazi se logika koja je specifična za repozitorij te koju nije moguće dijeliti s ostalim repozitorijima.

Za potrebe aplikacije kreirana su tri klase repozitorija: *LectureRepository*, *LectureApplicationRepository* te *RequestRepository*. Zbog jednostavnosti koda, prikazat će se samo *LectureApplicationRepository* klasa, koju je moguće vidjeti na slici 4.13.. Klasa *LectureApplicationRepository* nasljeđuje *BaseRepository* klasu kojoj pruža tip podatka te kontekst koji je potreban za rad s bazom podataka te implementira *ILectureApplicationRepository* sučelje. Repozitorij pruža logiku za dohvaćanje svih prijava na predavanje pomoću korisnikove identifikacije, identifikacije predavanja te dohvaćanje prijave putem korisnikove identifikacije i identifikacije predavanja.

BlavickiVNC - LectureApplicationRepository.cs

```
12  public class LectureApplicationRepository
13      : BaseRepository<BlavickiVNCContext, LectureApplication>, ILectureApplicationRepository
14  {
15      public LectureApplicationRepository(IDbContextService dbContextService) : base(dbContextService)
16      {
17          DefaultIncludes.Add(nameof(LectureApplication.User));
18      }
19
20      public ICollection<LectureApplication> GetByUser(string userId)
21      {
22          return GetData()
23              .Where(lectureApplication =>
24                      !lectureApplication.Deprecated
25                      && lectureApplication.UserId == userId
26                  )
27              .ToList();
28      }
29
30      public ICollection<LectureApplication> GetByLecture(int lectureId)
31      {
32          return GetData()
33              .Where(lectureApplication =>
34                      !lectureApplication.Deprecated
35                      && lectureApplication.LectureId == lectureId
36                  )
37              .ToList();
38      }
39
40
41      public LectureApplication GetByUserAndLecture(string userId, int lectureId)
42      {
43          return GetData()
44              .First(lectureApplication =>
45                  !lectureApplication.Deprecated
46                  && lectureApplication.UserId == userId
47                  && lectureApplication.LectureId == lectureId);
48      }
49  }
```

**Slika 4.13.** *LectureApplicationRepository klasa*

## 4.3. Servisi

Unutar projekta također je korišten arhitektonski obrazac servisne arhitekture (engl. *Service-Oriented Arhitecture*). Servisna arhitektura podrazumijeva korištenje servisnih klasa kojima se enkapsulira poslovna logika i operacije koje su u interakciji s repozitorijima, vanjskim servisima ili drugim ovisnostima. Cilj servisne arhitekture je postizanje čistog odvajanja problema tako što izolira složenost poslovne logike od logike korisničkog sučelja i logike pristupanja podacima u bazi podataka.

Kod obrasca servisne arhitekture, poput obrasca repozitorija, glavne komponente su servisni sučelja, konkretna implementacija klase servisa, model podatka koji reprezentira podatke u bazi podataka te ostatak aplikacije koji koristi servisne klase.

Kako bi se omogućilo korištenje metoda repozitorija putem servisnih klasa te kako bi se spriječila duplikacija koda, nalik na osnovnu klasu repozitorija, kreirana je osnovna servisna klasa kojom se konkretnoj servisnoj klasi pruža osnovna zajednička logika.

Za potrebe projekta kreirane su tri servisne klase: *RequestService*, *LectureService* i *LectureApplicationService*. Zbog jednostavnosti koda, u ovome radu opisana je samo *LectureApplicationService* klasa, koju je moguće vidjeti na slici 4.14..

BlavickiVNC - LectureApplicationService.cs

```
11  public class LectureApplicationService
12      : BaseEntityService<ILectureApplicationRepository, LectureApplication>, ILectureApplicationService
13  {
14      public LectureApplicationService(IDbContextService dbContextService, ILectureApplicationRepository entityRepository)
15          : base(dbContextService, entityRepository)
16      {
17      }
18
19      public ICollection<LectureApplication> GetByUser(string userId)
20      {
21          return WithinContext(context =>
22          {
23              return EntityRepository.GetByUser(userId);
24          });
25      }
26
27      public void DeleteByLecture(int lectureId)
28      {
29          WithinContext(context =>
30          {
31              var lectureApplications = EntityRepository.GetByLecture(lectureId);
32              EntityRepository.DeleteMultiple(lectureApplications);
33          });
34      }
35
36      public void DeleteByUserAndLecture(string userId, int lectureId)
37      {
38          WithinContext(context =>
39          {
40              var lectureApplication = EntityRepository.GetByUserAndLecture(userId, lectureId);
41
42              EntityRepository.Delete(lectureApplication);
43          });
44      }
45  }
```

**Slika 4.14.** *LectureApplicationService klasa*

Prilikom definiranja konkretne klase servisa, osnovnoj (*base*) klasi potrebno je pružiti klasu modela te klasu repozitorija koji je odgovoran za model kojim se upravlja. Nasljeđivanjem osnovne klase servisa omogućuje se kreiranje, ažuriranje, reaktiviranje, brisanje te dohvaćanje podataka iz baze podataka za definirani model. Implementacijom sučelja servisne klase omogućuje

se korištenje Unity Container radi ubrizgavanja ovisnosti Unity Container. Unutar konkretne servisne klase definira se logika koja je specifična za taj servis te koju nije moguće dijeliti s ostalim servisima. Unutar *LectureApplicationService* klase definirana je metoda za dohvaćanje svih prijava na predavanje putem korisnikove identifikacije, brisanje svih prijava koristeći identifikaciju predavanja te brisanje prijave koristeći korisnikovu identifikaciju i identifikaciju predavanja.

Osim navedene tri servisne klase koje rade s modelima, kreirana je i *EncryptionService* klasa koja se koristi za kriptiranje i dekriptiranje lozinki VNC servera koje je korisnik upisao prilikom kreiranja zahtjeva za pomoć tj. prilikom prijave na predavanje.

## 4.4. Kontroleri

.NET Framework nudi prikazivanje tj. pripremanje sadržaja stranice na strani poslužitelja (engl. *Server-Side Rendering*). *Server-side rendering* omogućuje brzo učitavanje stranice, gdje nije potrebno čekati da bi se JavaScript dokumenti učitali. Prilikom slanja GET HTTP zahtjeva na određene rute kontrolera, kontroler kao odgovor šalje HTML dokument s podacima koje je potrebno prikazati trenutnom korisniku. U svrhu ovog projekta razvijeno je sedam kontrolera koji će detaljnije biti opisani u narednom dijelu.

Kako bi se prikazali odgovarajući podaci na zaslonu, kontroleri uz zaslon također šalju i podatke, bilo to objekt, kolekcija objekata ili *ViewModel.* Svaki zaslon aplikacije koji zahtjeva korisnikovu interakciju i unos podataka posjeduje odgovarajući *viewmodel* koji se prosljeđuje zaslonu. Unutar tih *viewmodela* definirani su određeni uvjeti zavisno o tome koji su podaci traženi. *Viewmodeli* smješteni su unutar *ViewModels* direktorija koji se nalazi unutar *Models* direktorija. *Models* direktorija u odgovarajućem *ViewModels* poddirektoriju.

### 4.4.1. HomeController

*HomeController* jednostavan je kontroler koji je odgovoran za tri zaslona: početnu stranicu aplikacije, *About* stranica te *Contact* stranica. Sve tri rute dostupne unutar *Home* kontrolera moguće je vidjeti na slici 4.15..

BlavickiVNC - HomeController.cs

```
12  // GET: /
13  public ActionResult Index()
14  {
15      return View();
16  }
17
18  // GET: /Home/About
19  public ActionResult About()
20  {
21      ViewBag.Message = "BlavickiVNC application description page.";
22
23      return View();
24  }
25
26  // GET: /Home/Contact
27  public ActionResult Contact()
28  {
29      ViewBag.Message = "BlavickiVNC contact page.";
30
31      return View();
32  }
```

**Slika 4.15.** *HomeController*

### 4.4.2. AccountController

Sva logika potrebna za sustav autentikacije nalazi se unutar *AccountController* klase. *AccountController* klasa sadrži instance *ApplicationSignInManager* i *ApplicationUserManager* klasa te pomoću njih upravlja s korisnicima. *ViewModeli* odgovorni za *AccountController* nalaze se unutar *AccountViewModels.cs* datoteke.

Za logiku registracije korisnika u sustavu definirana je adresa „/Account/Register“ kojoj se mogu slati GET i POST zahtjevi, kao što je vidljivo na slici 4.16.. Slanjem GET zahtjeva preglednik se preusmjerava na zaslon za Registraciju. Pozivanjem POST zahtjeva s odgovarajućim podacima započinje se proces registraicije. Prilikom uspješne registracije, korisniku se na email adresu šalje poveznica pomoću koje korisnik potvrđuje svoju email adresu te ga se preusmjerava na “DisplayEmail” zaslon. Ako je došlo do grešaka, poruka odgovarajuće greške prikazuje se na zaslonu.

BlavickiVNC - AccountController.cs

```
140  // GET: /Account/Register
141  [HttpGet]
142  [AllowAnonymous]
143  public ActionResult Register()
144  {
145      return View(new RegisterViewModel());
146  }
147
148  // POST: /Account/Register
149  [HttpPost]
150  [AllowAnonymous]
151  [ValidateAntiForgeryToken]
152  public async Task<ActionResult> Register(RegisterViewModel model)
153  {
154      if (ModelState.IsValid)
155      {
156          var user = new ApplicationUser { UserName = model.UserName, Email = model.Email };
157          var result = await UserManager.CreateAsync(user, model.Password);
158          if (result.Succeeded)
159          {
160              var code = await UserManager.GenerateEmailConfirmationTokenAsync(user.Id);
161              var callbackUrl = Url.Action(
162                      "ConfirmEmail",
163                      "Account", new { userId = user.Id, code = code },
164                      protocol: Request.Url.Scheme
165                  );
166              await UserManager.SendEmailAsync(
167                  user.Id,
168                  "Confirm your account",
169                  @"Please confirm your account by clicking this link:
170                  <a href=\"" + callbackUrl + "\">link</a>"
171                  );
172              ViewBag.Link = callbackUrl;
173              return View("DisplayEmail");
174          }
175          AddErrors(result);
176      }
177
178      return View(model);
179  }
```

**Slika 4.16.** *Register ruta*

"DisplayEmail" zaslon jednostavan je zaslon, bez vlastite rute unutar kontrolera, pomoću kojeg se korisnika obavještava da je email poslan na korisnikovu email adresu te da ju potrebno potvrditi.

Poveznica koja je poslana na korisnikovu email adresu usmjerava korisnika na "ConfirmEmail" rutu. Ako su ruti poslani ispravni podaci, korisnikovi podaci unutar baze podataka su ažurirani, te vrijednost *EmailConfirmed* se stavlja na "*true*", te se korisnika preusmjerava na "ConfirmEmail" zaslon. Unutar tog zaslona, korisniku se zahvaljuje što je potvrdio email adresu te ga navodi da se prijavi.

Kao što se može vidjeti na slici 4.17., za logiku prijave korisnika definirane su dvije rute. Pozivanjem GET HTTP zahtjeva na adresi „/Account/Login“, korisnika se preusmjerava na zaslon za prijavu. Pozivanjem POST HTTP zahtjeva na istu tu adresu će pokrenuti proces prijave korisnika. U slučaju da su podaci koji su poslani ispravni, zavisno o tome je li korisnik omogućio autentikaciju u dva koraka, korisnika će se preusmjeriti na zaslon sigurnosnog koda ili prijaviti u sustav i preusmjeriti na početnu stranicu ako značajka *two-factor authentication* nije omogućena. Ako je prilikom prijave došlo do pogreške, korisniku će se na zaslon prikazati odgovarajuća poruka greške.

BlavickiVNC - AccountController.cs

```
57  // POST: /Account/Login
58  [HttpPost]
59  [AllowAnonymous]
60  [ValidateAntiForgeryToken]
61  public async Task<ActionResult> Login(LoginViewModel model, string returnUrl)
62  {
63      if (!ModelState.IsValid)
64      {
65          return View(model);
66      }
67
68      var result = await SignInManager.PasswordSignInAsync(
69              model.UserName,
70              model.Password,
71              model.RememberMe,
72              shouldLockout: true
73          );
74      switch (result)
75      {
76          case SignInStatus.Success:
77              return RedirectToLocal(returnUrl);
78          case SignInStatus.LockedOut:
79              return View("Lockout");
80          case SignInStatus.RequiresVerification:
81              return RedirectToAction(
82                  "SendCode",
83                  new { ReturnUrl = returnUrl, RememberMe = model.RememberMe });
84          case SignInStatus.Failure:
85          default:
86              ModelState.AddModelError("", "Invalid login attempt.");
87              return View(model);
88      }
89  }
```

**Slika 4.17.** *Login ruta*

Ako je korisnik omogućio *two-factor authentication*, nakon unosa podataka za prijavu, korisnika se preusmjerava na adresu “/Account/SendCode”. “SendCode” zaslon nudi korisniku izbornik za

odabir načina dobivanja sigurnosnog koda, zavisno o tome kako je korisnik postavio svoj račun. Slanje koda moguće je obaviti na dva načina, slanje koda na korisnikov email račun ako je email adresa potvrđena te slanje putem SMS poruke ako je korisnik dodao svoj telefonski broj. Nakon odabira načina slanja sigurnosnog koda, korisnika se preusmjerava na zaslon "VerifyCode".

"VerifyCode" ruta sadrži GET i POST zahtjeve. Prilikom ispravnog unosa sigurnosnog koda, korisnika se preusmjerava na adresu pohranjenu unutar *ReturnUrl* parametra, tj. na početnu stranicu. Ako je poslani kod pogrešan, odgovarajuća poruka pogreške prikazuje se na zaslonu.

Ako je korisnik zaboravio svoju lozinku, korisnik može pristupiti "/Account/ForgotPassword" ruti. "ForgotPassword" ruta prihvaća GET i POST zahtjeve. Nakon što korisnik unese svoju email adresu, korisnika se preusmjerava na "ForgotPasswordConfirmation" zaslon. Ako je korisnik upisao nepostojeću email adresu, poruka greške neće se prikazati radi zaštite podataka, kako se ne bi moglo saznati koji sve korisnici koriste aplikaciju.

Nakon što korisnik pritisne poveznicu koju je dobio na svojoj email adresi, korisnika se preusmjerava na „ResetPassword" rutu prema kojoj je moguće slati GET i POST zahtjeve. Ako su podaci koji su poslani prošli validaciju, no korisnik ne postoji, te prilikom uspješne promjene lozinke, korisnika se preusmjerava na „ResetPasswordConfirmation" zaslon.

Ako korisnik prilikom prijave odluči prijaviti se putem vanjskog pružatelja, odabirom željenog pružatelja korisnik šalje zahtjev na „/Account/ExternalLogin" rutu. „ExternalLogin" ruta preusmjerava korisnika na zaslon za prijavu odabranog pružatelja koristeći *ChallengeResult* klasu.

Nakon prijave putem vanjskog pružatelja, korisnika se preusmjerava na rutu „/Account/ExternalLoginCallback" gdje se nalazi logika za odluku preusmjeravanja korisnika. Ako vanjski pružatelj nije poslao informacije potrebne za autentikaciju, korisnika se preusmjerava na „Login" zaslon. Korisnika se preusmjerava na različite rute, zavisno o tome kako je korisnik postavio svoj račun.

U situaciji gdje se korisnik prijavio putem vanjskog pružatelja, a ne posjeduje lokalni račun, korisnika se preusmjerava na „/Account/ExternalLoginConfirmation" zaslon gdje je potrebno unijeti korisničko ime. Ako korisničko ime i email adresa nisu zauzeti, kreira se korisnika, prijavljuje u sustav te preusmjerava.

Kako bi se korisnik mogao odjaviti iz sustava, kreirana je „/Account/LogOff“ ruta koja prima POST zahtjev. Prilikom slanja POST zahtjeva na navedenu rutu, korisnika se odjavljuje iz sustava i preusmjerava na početnu stranicu. *LogOff* rutu moguće je vidjeti na slici 4.18..

BlavickiVNC - AccountController.cs

```
370  // POST: /Account/LogOff
371  [HttpPost]
372  [ValidateAntiForgeryToken]
373  public ActionResult LogOff()
374  {
375      AuthenticationManager.SignOut();
376      return RedirectToAction("Index", "Home");
377  }
```

**Slika 4.18.** *LogOff ruta*

### 4.4.3. ManageController

Za upravljanje korisničkim računom kreiran je *ManageController*. Pomoću *ManageController*-a, korisnik može promijeniti svoje podatke kao što su mijenjanje lozinke, kreiranje računa ako se korisnik registrirao putem vanjskih poveznica (Google, Facebook), dodati ili ukloniti svoj telefonski broj, upravljati načinima na koji se može prijaviti, upravljati sa svojim zahtjevima za pomoć ili prijavama za predavanja.

Odlaskom na svoju korisničku stranicu, korisnik pristupa „Indeks“ ruti *Manage* kontrolera. Kontroler kao odgovor šalje zaslon sa svim informacijama potrebnim za pravilno prikazivanje zaslona poput osobnih informacija o korisniku, poruka koje će biti prikazane u slučaju vršenja neke radnje te informacije o korisnikovom računu kako bi se određene funkcionalnosti mogle omogućiti tj. onemogućiti. Rutu „Index“ *Manage* kontrolera moguće je vidjeti na slici 4.19..

BlavickiVNC - ManageController.cs

```
42  // GET: /Manage/Index
43  [HttpGet]
44  public async Task<ActionResult> Index(ManageMessageId? message)
45  {
46      ViewBag.StatusMessage =
47          message == ManageMessageId.ChangePasswordSuccess ? "Your password has been changed."
48          : message == ManageMessageId.SetTwoFactorSuccess ? "Your two factor provider has been set."
49          : message == ManageMessageId.Error ? "An error has occurred."
50          : message == ManageMessageId.AddPhoneSuccess ? "The phone number was added."
51          : message == ManageMessageId.RemovePhoneSuccess ? "Your phone number was removed."
52          : message == ManageMessageId.RemoveLoginSuccess ? "The external login was removed."
53          : message == ManageMessageId.CreateLocalAccountSuccess ? "Successfully created a local account"
54          : "";
55
56      var userId = User.Identity.GetUserId();
57      var user = await UserManager.FindByIdAsync(userId);
58      if (user == null)
59      {
60          return View("Error");
61      }
62      var userRequests = _requestService.GetByUser(userId, true);
63      var userLogins = await UserManager.GetLoginsAsync(userId);
64      var otherLogins = AuthenticationManager
65          .GetExternalAuthenticationTypes()
66          .Where(auth => userLogins.All(ul => auth.AuthenticationType != ul.LoginProvider))
67          .ToList();
68
69      ViewBag.ShowRemoveButton = user.PasswordHash != null || userLogins.Count > 1;
70
71      var userApplications = _lectureApplicationService.GetByUser(userId);
72
73      userApplications = userApplications.Select(application =>
74      {
75          application.Lecture = _lectureService.GetById(application.LectureId);
76          return application;
77      }).ToList();
78
79      var userLectures = _lectureService.GetByUser(userId);
80
81      var model = new IndexViewModel
82      {
83          HasPassword = HasPassword(),
84          PhoneNumber = await UserManager.GetPhoneNumberAsync(userId),
85          TwoFactor = await UserManager.GetTwoFactorEnabledAsync(userId),
86          UserLogins = userLogins,
87          OtherLogins = otherLogins,
88          BrowserRemembered = await AuthenticationManager.TwoFactorBrowserRememberedAsync(userId),
89          Requests = userRequests,
90          LectureApplications = userApplications,
91          Lectures = userLectures,
92      };
93      return View(model);
94  }
```

**Slika 4.19.** *Manage Index ruta*

Ako je korisnik izvršio prijavu putem vanjskih usluga, korisniku će na njegovoj stranici biti omogućena funkcionalnost postavljanja lokalne lozinke. Za postavljanje lokalne lozinke potrebno je pristupiti „/Manage/SetPassword“ ruti. Prilikom slanja podataka provjerava se ispravnost

podataka, te ako su podaci ispravni, spreman lozinku u bazu podataka. Ako dođe do pogreške prilikom ažuriranja korisničkog računa, poruka greške prikazuje se na zaslonu.

Pri situaciji gdje je korisnik izvršio proces registracije putem email adrese i lozinke, *SetPassword* funkcionalnost je onemogućena, dok je funkcionalnost mijenjanja korisničke lozinke omogućena. Da bi korisnik promijenio svoju lozinku, potrebno je otići na rutu „/Manage/ChangePassword“. Nakon validacije poslanih podataka, korisnikova lozinka se mijenja, korisnika se ponovno prijavljuje te ga se preusmjerava na korisnikovu stranicu.

Funkcionalnost dodavanja telefonskog broja dostupna je samo ako korisnik nije ranije dodao svoj telefonski broj. Da bi dodao svoj telefonski broj, potrebno je otići na rutu „/Manage/AddPhoneNumber“. Nakon slanja svog telefonskog broja, kreira se sigurnosni kod na temelju korisnikove identifikacije i njegovog telefonskog broja. Generirani sigurnosni kod šalje se putem SMS poruke korisniku te ga se preusmjerava na rutu „/Manage/VerifyPhoneNumber“.

Da bi potvrdio svoj telefonski broj, potrebno je da korisnik na „VerifyPhoneNumber“ zaslonu unese sigurnosni kod koji je primio putem SMS poruke. Ako je kod ispravan, korisnikov telefonski broj dodaje se u bazu podataka, korisnika se ponovno prijavljuje u sustav te ga se preusmjerava na korisnikovu stranicu.

Nakon uspješnog dodavanja telefonskog broja, korisniku se omogućavaju dvije dodatne funkcionalnosti, mijenjanje telefonskog broja te uklanjanje telefonskog broja.

Za mijenjanje telefonskog broja korištene su iste rute kao pri dodavanju telefonskog broja, „/Manage/AddPhoneNumber“ i „/Manage/VerifyPhoneNumber“, stoga nije potrebno ponovno objašnjavati proces.

Za uklanjanje telefonskog broja korištena je ruta „/Manage/RemovePhoneNumber“. Logika uklanjanja telefonskog broja je jednostavna, vrijednost broja postavlja se na vrijednost „*null*“, korisnika se ponovno prijavljuje te ga se preusmjerava na korisnikovu stranicu. Prilikom greške, odgovarajuća poruka šalje se sa zaslonom kako bi se mogla prikazati korisniku.

Na korisnikovoj stranici također je moguće upravljati s kojim je vanjskim pružateljima moguće vršiti proces prijave. Prilikom pritiska željenog vanjskog pružatelja, korisnik šalje POST HTTP zahtjev na rutu „/Manage/LinkLogin“, koja samo preusmjerava zahtjev na *ChallengeResult* klasi

koja se nalazi unutar *AccountController* klase. Ako je prijava vanjskog pružatelja uspješna, korisnika se preusmjerava na korisnikovu stranicu.

Kako bi uklonio vanjskog pružatelja za prijavu, korisnik mora imati kreiran lokalni račun, tj. postavljenu lozinku, ili mora imati više od jednog vanjskog pružatelja za prijavu. Da bi uklonio vanjskog pružatelja, potrebno je poslati zahtjev na rutu „/Manage/RemoveLogin“, gdje se uklanja korisnikov vanjski pružatelj za prijavu, ponovno ga se prijavljuje u sustav te preusmjerava na korisnikovu stranicu.

Na korisnikovoj stranici također je moguće i omogućiti/onemogućiti *two-factor authentication*. Kako bi se omogućila autentikacija u dva koraka, potrebno je poslati zahtjev na rutu „/Manage/EnableTFA“, gdje se mijenja korisnikova vrijednost *TwoFactorEnabled* svojstva u bazi podataka na „*true*“ te ponovno prijavljuje korisnika u sustav kao što je vidljivo na slici 4.20..

BlavickiVNC - ManageController.cs

```
188  // POST: /Manage/EnableTFA
189  [HttpPost]
190  [ValidateAntiForgeryToken]
191  public async Task<ActionResult> EnableTFA()
192  {
193      var userId = User.Identity.GetUserId();
194      await UserManager.SetTwoFactorEnabledAsync(userId, true);
195      var user = await UserManager.FindByIdAsync(userId);
196      if (user != null)
197      {
198          await SignInAsync(user, isPersistent: false);
199      }
200      return RedirectToAction("Index", "Manage");
201  }
```

**Slika 4.20.** *EnableTFA ruta*

Kako bi se onemogućila funkcionalnost *two-factor authentication*, potrebno je poslati zahtjev na rutu „/Manage/DisableTFA“, gdje se mijenja korisnikova vrijednost *TwoFactorEnabled* svojstva u bazi podataka na „*false*“ te ponovno prijavljuje korisnika u sustav.

Korisnik također može postaviti da sustav zapamti njegov preglednik, te tako zaobići *two-factor authentication* ako koristi isti preglednik. *Two-factor authentication* i dalje će se provoditi ako korisnik koristi drugi preglednik. Da bi sustav zapamtio korisnikov preglednik, potrebno je poslati zahtjev na rutu „/Manage/RememberBrowser“.

Ako korisnik želi da sustav zaboravi njihov preglednik, potrebno je poslati zahtjev na rutu „/Manage/ForgetBrowser“.

Posljednja funkcionalnost koju *ManageController* nudi je brisanje prijave za predavanje. Kako bi korisnik izbrisao svoju prijavu na predavanje, potrebno je poslati zahtjev na „/Manage/CancelApplication“. Nakon uspješnog brisanja prijave, korisnika se preusmjerava na korisnikovu stranicu kao što je vidljivo na slici 4.21.. Ako korisnik pokuša obrisati prijavu koja nije njegova, korisnika se preusmjerava na korisnikovu stranicu bez pružanja dodatnih informacija.

BlavickiVNC - ManageController.cs

```
96  // GET: /Manage/CancelApplication
97  [HttpPost]
98  [Authorize]
99  [ValidateAntiForgeryToken]
100 public ActionResult CancelApplication(int applicationId)
101 {
102
103     var lectureApplication = _lectureApplicationService.GetById(applicationId);
104
105     if (lectureApplication.UserId != User.Identity.GetUserId()) {
106         return RedirectToAction("Index");
107     }
108
109     _lectureApplicationService.Delete(lectureApplication);
110
111     return RedirectToAction("Index");
112 }
```

**Slika 4.21.** *CancelApplication ruta*

### 4.4.4. UsersAdminController

*UsersAdminController* odgovoran je za upravljanje korisnicima te je dostupan samo za korisnike koji imaju ulogu administrator. Preko tog kontrolera, administratori mogu vidjeti sve korisnike i njihove uloge, vidjeti detalje o korisniku, mijenjati uloge korisnika te brisati korisnike.

Da bi se pristupilo početnom zaslonu *UsersAdmin* kontrolera, potrebno je otići na rutu „/UsersAdmin“ kao što je vidljivo na slici 4.22.. Prije slanja zaslona, kontroler prvo dohvaća sve korisnike, te za svakog korisnika dohvaća njegove uloge. Dohvaćeni podaci šalju se nazad uz odgovarajući zaslon.

BlavickiVNC - UsersAdminController.cs

```
53  // GET: /UsersAdmin/
54  [HttpGet]
55  public async Task<ActionResult> Index()
56  {
57      var users = await UserManager.Users.ToListAsync();
58
59      var tasks = users.Select(async user =>
60      {
61          var roles = await UserManager.GetRolesAsync(user.Id);
62
63          return new UsersViewModel
64          {
65              UserRoles = string.Join(", ", roles),
66              User = user
67          };
68      });
69
70      var usersViewModels = await Task.WhenAll(tasks);
71
72      return View(usersViewModels);
73  }
```

**Slika 4.22.** *UsersAdmin Index ruta*

Da bi se obrisao korisnik, potrebno je poslati zahtjev na rutu „/UsersAdmin/Delete“. Navedena ruta vraća zaslon potvrde, kako bi se izbjeglo slučajno brisanje korisnika. Nakon potvrde, šalje se zahtjev na rutu „/UsersAdmin/DeleteConfirmed“, gdje metoda prvo provjerava postoji li korisnik, a nakon uspostave da korisnik postoji, briše korisnika iz baze podataka. Brisanjem korisnika također se brišu i korisnikovi zahtjevi za pomoć, prijave na predavanja, korisnikova predavanja te sve prijave na njegova predavanja. Nakon uspješnog brisanja, korisnika se preusmjerava nazad na početnu stranicu *UsersAdmin* kontrolera.

Kako bi se promijenilo korisnikove uloge, potrebno je napraviti zahtjev na rutu „/UsersAdmin/Edit“. Metoda dodaje sve uloge koje su poslane, a koje korisnik ne sadrži te briše sve uloge koje korisnik sadrži, a koje nisu poslane. Nakon ažuriranja korisnikovih uloga, korisnika se preusmjerava na početni zaslon *UsersAdmin* kontrolera.

Kako bi se pristupilo *Details* zaslonu određenog korisnika, potrebno je pristupiti ruti „/UsersAdmin/Details“. Prije slanja zaslona, metoda prvo dohvaća korisnikove osobne podatke, korisnikove zahtjeve za pomoć te korisnikova predavanja.

### 4.4.5. RolesAdminController

*RolesAdminController* odgovoran je za upravljanje ulogama i njihovim korisnicima unutar aplikacije. Kao i kod *UsersAdmin* kotrolera, samo korisnici koji imaju administrator ulogu mogu pristupiti kontroleru.

Kako bi se pristupilo početnom zaslonu *RolesAdmin* kontrolera, potrebno je pristupiti ruti „/RolesAdmin“.

Kako bi se upravljalo korisnicima uloge, potrebno je otići na stranicu detalja određene uloge koja se nalazi na ruti „RolesAdmin/Details“ kao što je vidljivo na slici 4.23.. Prije slanja zaslona, metoda prvo dohvaća sve korisnike te filtrira one koji se nalaze u toj ulozi.

BlavickiVNC - RolesAdminController.cs

```
56  // GET: /Roles/Details/5
57  [HttpGet]
58  public async Task<ActionResult> Details(string id, RolesAdminMessageId? message)
59  {
60      ViewBag.StatusMessage =
61          message == RolesAdminMessageId.Error ? "An error has occurred."
62          : message == RolesAdminMessageId.RemoveUserSuccess ? "The user was successfully removed."
63          : "";
64
65      if (id == null)
66      {
67          return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
68      }
69      var role = await RoleManager.FindByIdAsync(id);
70      var users = new List<ApplicationUser>();
71
72      foreach (var user in UserManager.Users.ToList())
73      {
74          if (await UserManager.IsInRoleAsync(user.Id, role.Name))
75          {
76              users.Add(user);
77          }
78      }
79
80      ViewBag.Users = users;
81      ViewBag.UserCount = users.Count();
82      return View(role);
83  }
```

**Slika 4.23.** *Details ruta*

Kako bi se uklonio korisnik iz određene uloge, potrebno je poslati zahtjev na rutu „/RolesAdmin/RemoveUser“. Ako je uklanjanje korisnika uspješno, korisnika se preusmjerava na stranicu detalja uloge.

### 4.4.6. RequestController

*RequestController* odgovoran je za upravljanje VNC zahtjevima za pomoć.

Kako bi se pristupilo početnoj stranici *Request* kontrolera, potrebno je otići na rutu “/Request”. Prije slanja zaslona, metoda dohvaća sve zahtjeve koji nemaju svojstvo „*Deprecated*“ postavljeno na „*true*“, tj. sve zahtjeve koji nisu izbrisani.

Kako bi se kreirao novi VNC zahtjev za pomoć, potrebno je poslati zahtjev na rutu „/Request/Create“. Za kreiranje zahtjeva potrebno je navesti naslov, opis te lozinku pomoću koje će korisnici s ulogom administrator ili moderator pristupiti računalu. Radi sigurnosti korisnika, lozinka koju su pružili enkriptirana je prije spremanja u bazu podataka. Nakon uspješnog kreiranja zahtjeva, korisnika se preusmjerava na početni zaslon Request kontrolera.

Kako bi se obrisao zahtjev, potrebno je poslati zahtjev na rutu „/Request/Delete“.

Da bi se pružila pomoć korisnicima, potrebno je poslati zahtjev na rutu „/Request/RequestSession“. Kao što je vidljivo na slici 4.24., samo korisnici s ulogama administrator ili moderator mogu pružiti VNC pomoć. Metoda dohvaća VNC zahtjev za pomoć, dekriptira korisnikovu lozinku te vraća zaslon s podacima o VNC zahtjevu za pomoć.

BlavickiVNC - RequestController.cs

```
93  // GET: /Request/RequestSession
94  [HttpGet]
95  [Authorize(Roles ="Admin,Moderator")]
96  public ActionResult RequestSession(int requestId)
97  {
98      var request = _requestService.GetById(requestId);
99
100     request.Password = _encryptionService.Decrypt(request.Password);
101
102     return View(request);
103 }
```

**Slika 4.24.** *RequestSession ruta*

Kako bi se zaustavilo VNC pomaganje korisniku, potrebno je poslati zahtjev na rutu „/Request/EndRequestSession“. Metoda briše VNC zahtjev za pomoć, tj. postavlja vrijednost svojstva „*Deprecated*“ na „*true*“, te preusmjerava korisnika na početnu stranicu *Request* kontrolera. Metodu *EndRequestSession* moguće je vidjeti na slici 4.25..

BlavickiVNC - RequestController.cs

```
105  // POST: /Request/EndRequestSession
106  [HttpPost]
107  [Authorize(Roles ="Admin,Moderator")]
108  [ValidateAntiForgeryToken]
109  public ActionResult EndRequestSession(int requestId)
110  {
111      var request = _requestService.GetById(requestId);
112
113      _requestService.Delete(request);
114
115      return RedirectToAction("Index");
116  }
```

**Slika 4.25.** *EndRequestSession ruta*

Zahtjeve za pomoć moguće je reaktivirati slanjem zahtjeva na rutu „/Request/Reactivate”. Metoda dohvaća zahtjev te mijenja vrijednost svojstva „*Deprecated*“ na „*false*“.

### 4.4.7. LectureController

*LectureController* klasa odgovorna je za upravljanje VNC predavanjima.

Kako bi se pristupilo početnoj stranici *Lecture* kontrolera, potrebno je poslati zahtjev prema ruti “/Lecture”. Metoda dohvaća sva aktivna VNC predavanja te dohvaćene informacije šalje sa zaslonom.

Kreiranje novog VNC predavanja moguće je samo za moderator i administrator uloge, a može se izvršiti na ruti “/Lecture/Create”. Za kreiranje VNC predavanja potrebno je navesti naslov, opis te datum i vrijeme početka predavanja. Nakon uspješnog kreiranja VNC predavanja, korisnika se preusmjerava na početni zaslon *Lecture* kontrolera.

Za brisanje VNC predavanja, kreirana je ruta „/Lecture/Delete“. Brisanje VNC predavanja također briše i sve prijave na VNC predavanje

Za prijavu na VNC predavanje potrebno je pristupiti ruti „/Lecture/Apply“. Kako bi se korisnik prijavio na predavanje, od korisnika se zahtjeva unos lozinke kojom se pristupa njegovom VNC serveru. Kao i kod zahtjeva za pomoć, lozinke korisnika kriptiraju se prije spremanja u bazu podataka. *Apply* metodu *LectureController* klase moguće je vidjeti na slici 4.26..

BlavickiVNC - LectureController.cs

```
49  // GET: /Lecture/Apply
50  [HttpGet]
51  [Authorize]
52  public ActionResult Apply(int lectureId, string returnUrl)
53  {
54      var lecture = _lectureService.GetById(lectureId);
55
56      ViewBag.LectureId = lectureId;
57      ViewBag.ReturnUrl = returnUrl;
58      ViewBag.LectureTitle = lecture.Title;
59
60      return View(new ApplyLectureViewModel());
61  }
62
63  // POST: /Lecture/Apply
64  [HttpPost]
65  [Authorize]
66  [ValidateAntiForgeryToken]
67  public ActionResult Apply(int lectureId, string returnUrl, ApplyLectureViewModel model)
68  {
69      if (!ModelState.IsValid)
70      {
71          return View(model);
72      }
73
74      var lectureApplication = new LectureApplication
75      {
76          UserId = model.UserId,
77          LectureId = lectureId,
78          Password = _encryptionService.Encrypt(model.Password)
79      };
80
81      _lectureApplicationService.Create(lectureApplication);
82
83      return RedirectToLocal(returnUrl);
84  }
```

**Slika 4.26.** *Apply ruta*

U slučaju da se korisnik želi odjaviti s predavanja, tj. ukinuti prijavu, potrebno je poslati zahtjev na rutu “/Lecture/CancelApplication”.

Kako bi se pokrenulo VNC predavanje, potrebno je poslati zahtjev na rutu "/Lecture/LectureSession". Metoda *LectureSession* dohvaća predavanje iz baze podataka kao i sve prijave za predavanje. Prije slanja podataka sa zaslonom, metoda dekriptira lozinke korisnika, kao što je vidljivo na slici 4.27..

BlavickiVNC - LectureController.cs

```
143  // GET: /Lecture/LectureSession
144  [HttpGet]
145  [Authorize(Roles = "Admin,Moderator")]
146  public ActionResult LectureSession(int lectureId)
147  {
148      var lecture = _lectureService.GetById(lectureId);
149
150      foreach (var lectureApplication in lecture.Applications)
151      {
152          lectureApplication.Password = _encryptionService.Decrypt(lectureApplication.Password);
153      }
154
155      lecture.Applications = lecture.Applications.Where(application => !application.Deprecated).ToList();
156
157      return View(lecture);
158  }
```

**Slika 4.27.** *LectureSession ruta*

Za prekidanje predavanja, potrebno je poslati zahtjev na rutu "/Lecture/EndLectureSession". Metoda *EndLectureSession* briše sve prijave na predavanje te nakon toga briše i samo predavanje. Nakon brisanja, korisnika se preusmjerava na početnu stranicu *Lecture* kontrolera. Metodu *EndLectureSession* moguće je vidjeti na slici 4.28..

BlavickiVNC - LectureController.cs

```
160  // POST: /Lecture/EndLectureSession
161  [HttpPost]
162  [ValidateAntiForgeryToken]
163  [Authorize(Roles = "Admin,Moderator")]
164  public ActionResult EndLectureSession(int lectureId)
165  {
166      var lecture = _lectureService.GetById(lectureId);
167
168      foreach(var application in lecture.Applications)
169      {
170          _lectureApplicationService.Delete(application);
171      }
172
173      _lectureService.Delete(lecture);
174
175      return RedirectToAction("Index");
176  }
```

**Slika 4.28.** *EndLectureSession ruta*

### 4.4.8. Klijentski kod

Za uspostavu konekcije i daljinsko upravljanje računala koristi se noVNC paket koji je potrebno implementirati na klijentskoj strani koda. Za potrebe ovoga projekta kreirane su dvije JavaScript datoteke, *LectureSession.js* i *RequestSession.js*.

Na slici 4.29. može se vidjeti sadržaj *RequestSession.js* datoteka te način na koji, uz pomoć noVNC paketa, se uspostavlja konekcija s udaljenim računalom. Unutar *startClient* funkcije definirane su funkcije koje će biti izvršene prilikom određenih događaja. Unutar funkcije također se konfigurira način na koji je moguće upravljati i prikazivati uspostavljenu VNC vezu.

*Lecture.js* datoteka uspostavlja konekciju na isti način, no razlika je u tome što uspostavlja više konekcija, po jednu za svaku prijavu na predavanje. Još jedna razlika među navedene dvije datoteke je način na koji *Lecture.js* datoteka rukuje sa svakom konekcijom. Prilikom uspostave konekcije, korisnik može samo gledati zaslon udaljenog računala, a prilikom pritiska na željeni zaslon mijenja se veličina prikazanog zaslona te se omogućuje korisniku da ga upravlja.

BlavickiVNC - RequestSession.js

```
import RFB from "../node_modules/@novnc/novnc/core/rfb.js";

function startClient(vncPassword) {
    function connectedToServer(e) {
        status("Connected to server");
    }

    function disconnectedFromServer(e) {
        e.detail.clean ? status("Disconnected") : status("Something went wrong, connection is closed");
        $("#screen").hide();
    }

    function failedToConnect(e) {
        $(`#error-message`).show();
    }

    function status(text) {
        document.getElementById('status').textContent = text;
    }

    const host = window.location.hostname;
    const port = "6080";
    const path = "websockify";

    let url = "ws";

    url += "://" + host;
    if (port) {
        url += ":" + port;
    }
    url += "/" + path;

    try {
        const rfb = new RFB(document.getElementById("screen"), url, {
            credentials: { password: vncPassword },
        });

        rfb.addEventListener("connect", connectedToServer);
        rfb.addEventListener("disconnect", disconnectedFromServer);
        rfb.addEventListener("securityfailure", failedToConnect);

        rfb.clipViewport = true;
        rfb.dragViewport = true;
        rfb.scaleViewport = true;
        rfb.resizeSession = true;
    } catch (error) {
        console.error(error);
    }
}

$(function () {
    const request = $("#model-id").val();
    const password = $("#model-password").val();

    $("#error-message").hide();

    try {
        startClient(password);
        $("#top_bar").hide();
    } catch (error) {
        console.error(error);
    }
});
```

**Slika 4.29.** *RequestSession.js*

# 5. KORISNIČKO SUČELJE

U narednom poglavlju opisano je korisničko sučelje aplikacije. Slično načinu kako su kontroleri opisani, korisničko sučelje također će biti opisano prolazeći kroz kontrolera koji su odgovorni za prikazivanje stranica.

Tijekom opisivanja aplikacije neće biti prikazani cijeli zasloni nego samo dijelovi aplikacije koji se opisuju.

Navigacijska traka razlikuje se zavisno o tome je li korisnik prijavljen ili nije. Navigacijska traka također se može razlikovati za prijavljene korisnike, zavisno o njihovim dodijeljenim ulogama.

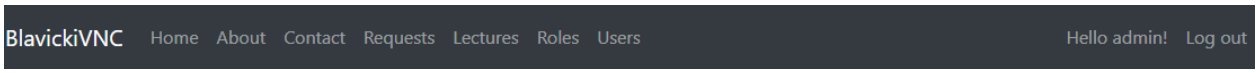
Kao što je vidljivo na slici 5.1., navigacijska traka za korisnike koji nisu prijavljeni sadrži tipke „Home“, „About“ „Contact“, „Register“ i „Login“.


**Slika 5.1.** *Navigacijska traka neprijavljenog korisnika*

Kada se korisnici prijave u aplikaciju, tipke „Register“ i „Login“ više se ne vide, a na njihovom mjestu stoji „Log out“ tipka te tipka za njihov korisnički račun, kao što je vidljivo na slici 5.2.. Prijavom u sustav, korisniku su također vidljive tipke „Request“ i „Lecture“.


**Slika 5.2.** *Navigacijska traka prijavljenog korisnika*

Ako prijavljeni korisnik ima ulogu administrator, na navigacijskoj traci također su mu dostupne i tipke „Roles“ i „Users“, što je moguće vidjeti na slici 5.3..

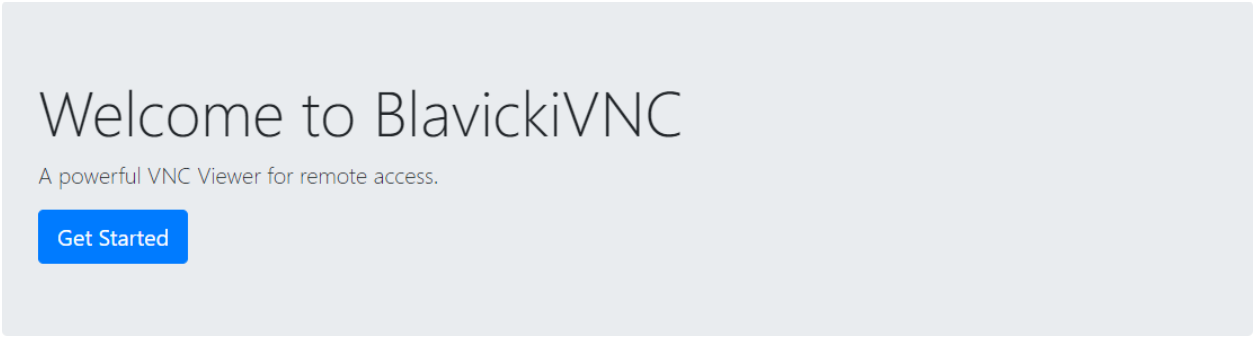



**Slika 5.3.** *Navigacijska traka administrator korisnika*

## 5.1. HomeController

*Home* kontroler odgovoran je za tri jednostavna zaslona a to su početni (engl. *Home*) zaslon, *About* zaslon te *Contact* zaslon.

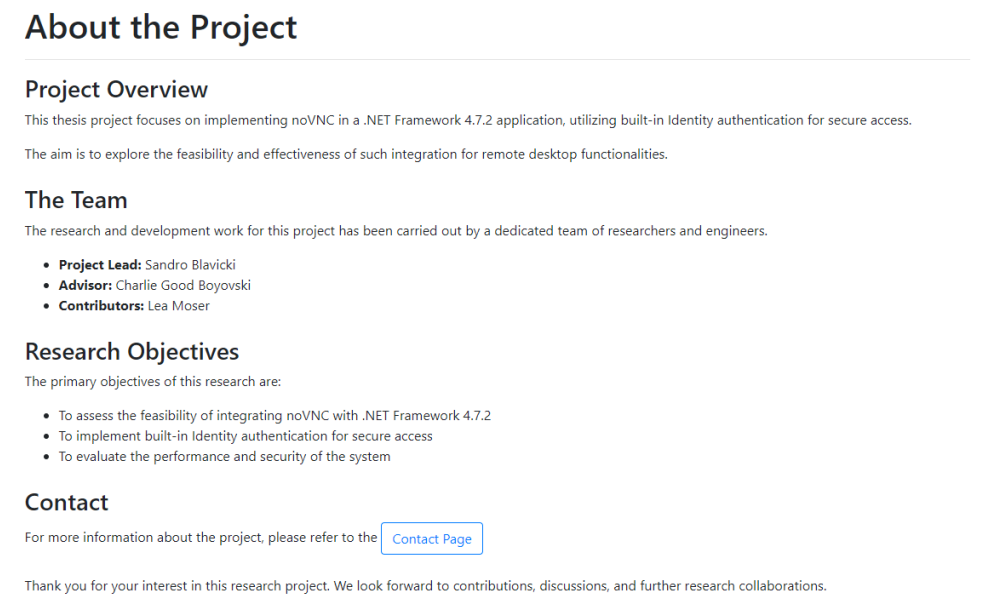
Na slici 5.4. moguće je vidjeti prikaz početnog zaslona aplikacije.




**Slika 5.4.** *Početni zaslon*

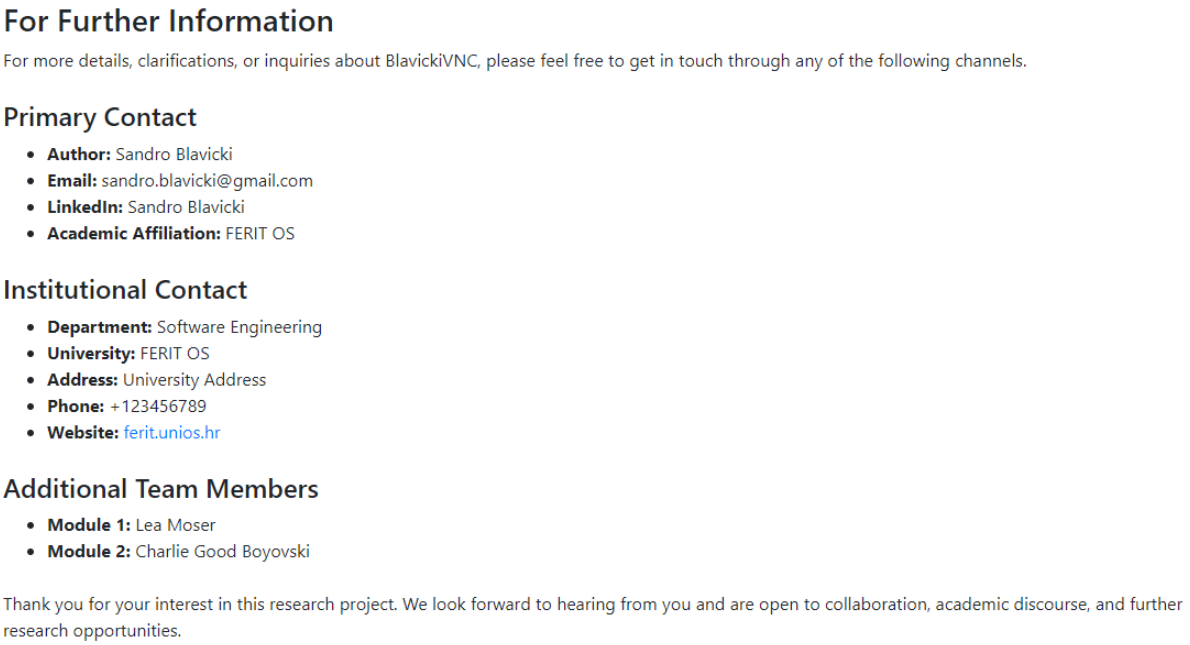
Ako korisnik nije prijavljen u aplikaciju, pritiskom na tipku „Get Started“ korisnika se preusmjerava na zaslon za prijavu. U situaciji gdje je korisnik prijavljen, korisnika se preusmjerava na početni zaslon *Request* kontrolera.

Na slici 5.5. može se vidjeti *About* zaslon. *About* zaslon jednostavan je zaslon te neće biti detaljnije opisan.




**Slika 5.5.** *About zaslon*

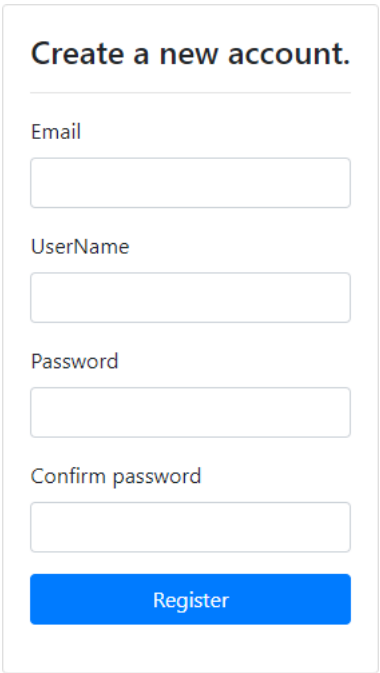
Kao i kod *About* zaslona, *Contact* zaslon također je jednostavan zaslon bez logike, stoga neće biti detaljnije opisan. Navedeni zaslon moguće je vidjeti na slici 5.6..




**Slika 5.6.** *Contact zaslon*

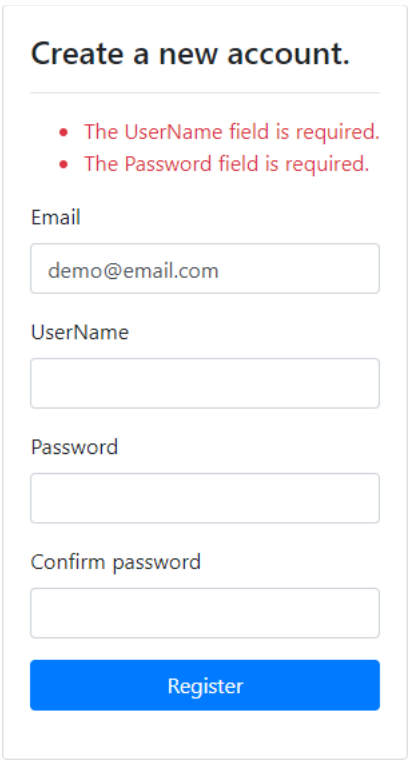
## 5.2. AccountController

Pritiskom na tipku „Register“ u navigacijskoj traci korisnika se preusmjerava na zaslon za registraciju. Zaslon za registraciju moguće je vidjeti na slici 5.7..




**Slika 5.7.** *Forma za registraciju*

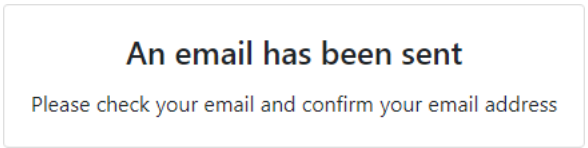
Ako su podaci koje je korisnik upisao neispravni, na zaslonu se prikazuje odgovarajuća poruka pogreške kao što je moguće vidjeti na slici 5.8..




**Slika 5.8.** *Forma za registraciju sa porukama greške*

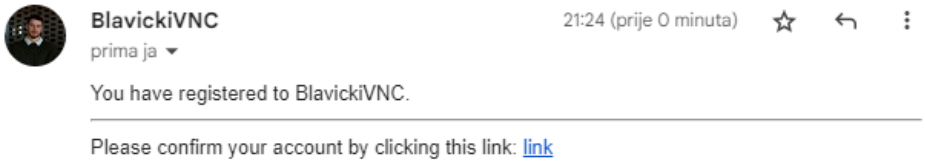
Kako bi se pojednostavilo opisivanje korisničkog sučelja, u narednim dijelovima prikazane su samo prazne forme bez prikazivanja zaslona u slučaju neispravnih podataka.

Ako su podaci ispravni, korisnika se preusmjerava na zaslon potvrde kao što je vidljivo na slici 5.9.. Na zaslonu se vidi poruka koja obavještava korisnika o poslanome mailu pomoću kojeg je potrebno potvrditi email adresu.



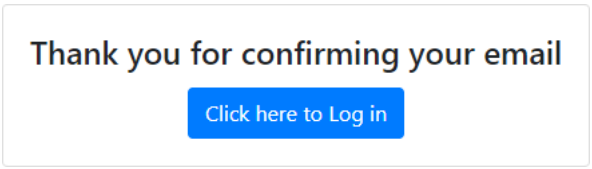

**Slika 5.9.** *Potvrda poslanog maila*

Mail koji je poslan može se vidjeti na slici 5.10..



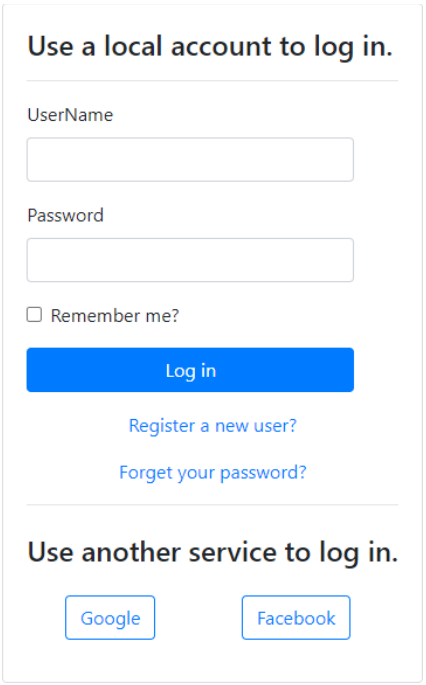

**Slika 5.10.** *Mail za potvrdu email adrese*

Pritiskom na poveznicu u poslanome mailu, korisnika se preusmjerava na zaslon gdje se može vidjeti da povratna poruka o korisnikovoj potvrdi email adrese. Navedenu poruku moguće je vidjeti na slici 5.11..




**Slika 5.11.** *Potvrda email adrese*

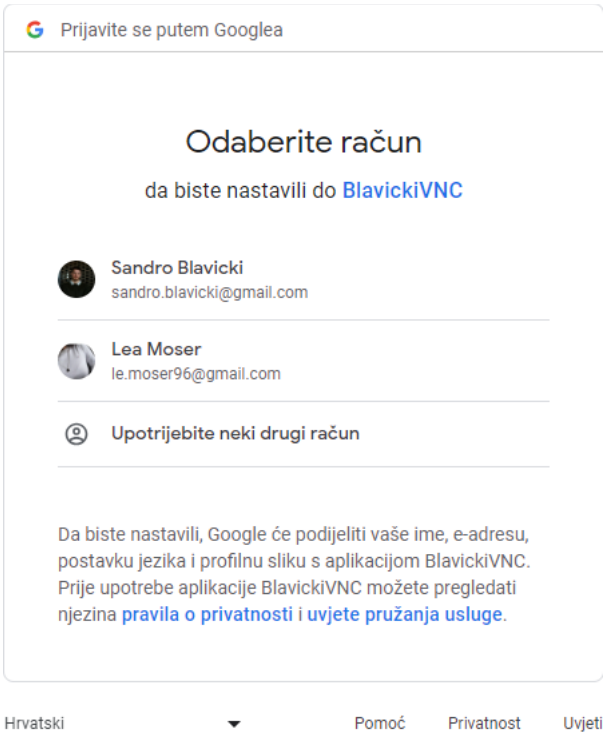
Nakon pritiska tipke „Click here to Log in“, korisnika se preusmjerava na zaslon za prijavu. Formu za prijavu moguće je vidjeti na slici 5.12.. Korisnika se prilikom uspješne prijave preusmjerava na početnu stranice aplikacije te mu se navigacijska traka mijenja kao što je opisano u ranijem poglavlju.




**Slika 5.12.** *Forma za prijavu*

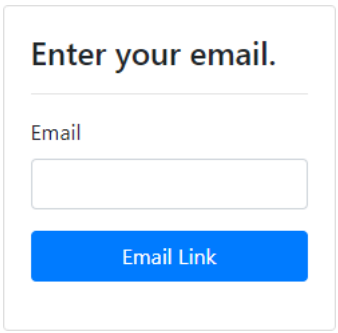
Prilikom pritiska tipke „Register a new user?“, korisnika se preusmjerava na zaslon za registraciju koji je opisan u prethodnom dijelu.

Ako se korisnik želi prijaviti putem vanjskog pružatelja, potrebno je pritisnuti tipku željenog pružatelja. Nakon odabira željenog pružatelja, korisnika se preusmjerava na zaslon koji je pružen od strane vanjskog pružatelja. Zaslon za prijavu putem „Google“ pružatelja moguće je vidjeti na slici 5.13..



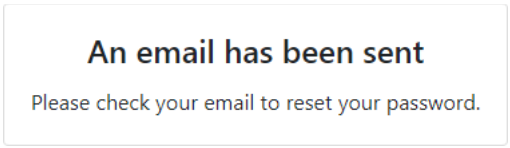

**Slika 5.13**. *Prijava putem Google računa*

Ako je korisnik zaboravio svoju zaporku, potrebno je pritisnuti tipku „Forget your password?“. Prilikom pritiska tipke, korisnika se preusmjerava na zaslon gdje je potrebno unijeti korisnikovu email adresu. Formu za unos email adrese moguće je vidjeti na slici 5.14.



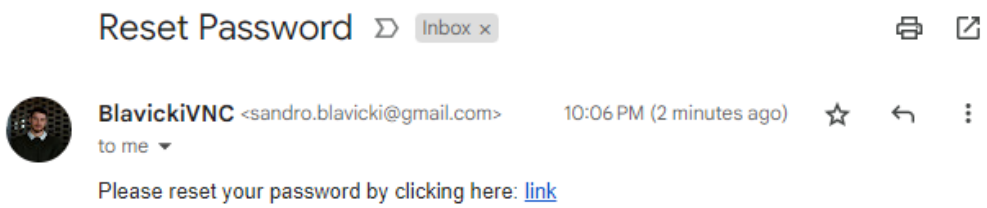

**Slika 5.14.** *Forma za forget password*

Nakon unosa email adrese i pritiska na tipku „Email Link“, korisnika se preusmjerava na zaslon potvrde gdje se prikazuje poruka da korisnik provjeri svoju email adresu kako bi resetirao svoju lozinku. Zaslon potvrde da je mail poslan može se vidjeti na slici 5.15..



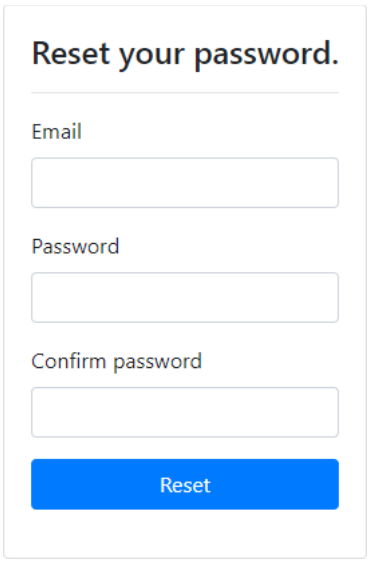

**Slika 5.15.** *Potvrda poslanog maila*

Poslani mail moguće je vidjeti na slici 5.16..



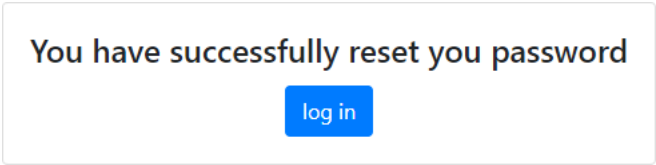

**Slika 5.16**. *Mail za resetiranje lozinke*

Pritiskom na poveznicu, korisnika se preusmjerava na zaslon koji sadrži formu za resetiranje lozinke. Navedenu formu moguće je vidjeti na slici 5.17..



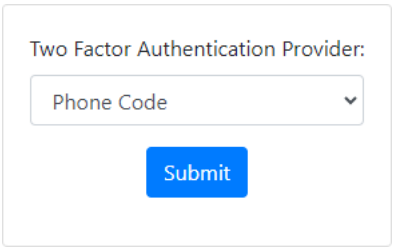

**Slika 5.17.** *Forma za resetiranje lozinke*

Ako su podaci koje je korisnik upisao ispravni, pritiskom na tipku „Reset“, korisnika se preusmjerava na zaslon potvrde. Na zaslonu je prikazana poruka koja potvrđuje da je korisnik uspješno resetirao svoju lozinku te je dostupna tipka „Log in“ što je moguće vidjeti na slici 5.18.. Pritiskom na tipku „Log in“, korisnika se preusmjerava na zaslon za prijavu koji je prethodno opisan.



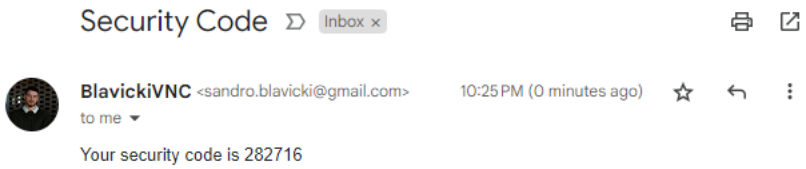

**Slika 5.18.** *Potvrda resetiranja lozinke*

Proces prijave može se razlikovati zavisno o tome je li korisnik omogućio funkcionalnost *two-factor authentication*. Ako funkcionalnost nije omogućena, korisnika se nakon unosa podataka preusmjerava na početni zaslon aplikacije. U slučaju da je korisnik omogućio autentikaciju u dva koraka, korisnika se preusmjerava na zaslon za odabir načina primanja sigurnosnog koda za autentikaciju. U aplikaciji je omogućeno slanje sigurnosnog koda pomoću email adrese i telefonskog broja, zavisno o tome koje podatke je korisnik potvrdio. Zaslon odabira pružatelja moguće je vidjeti na slici 5.19..



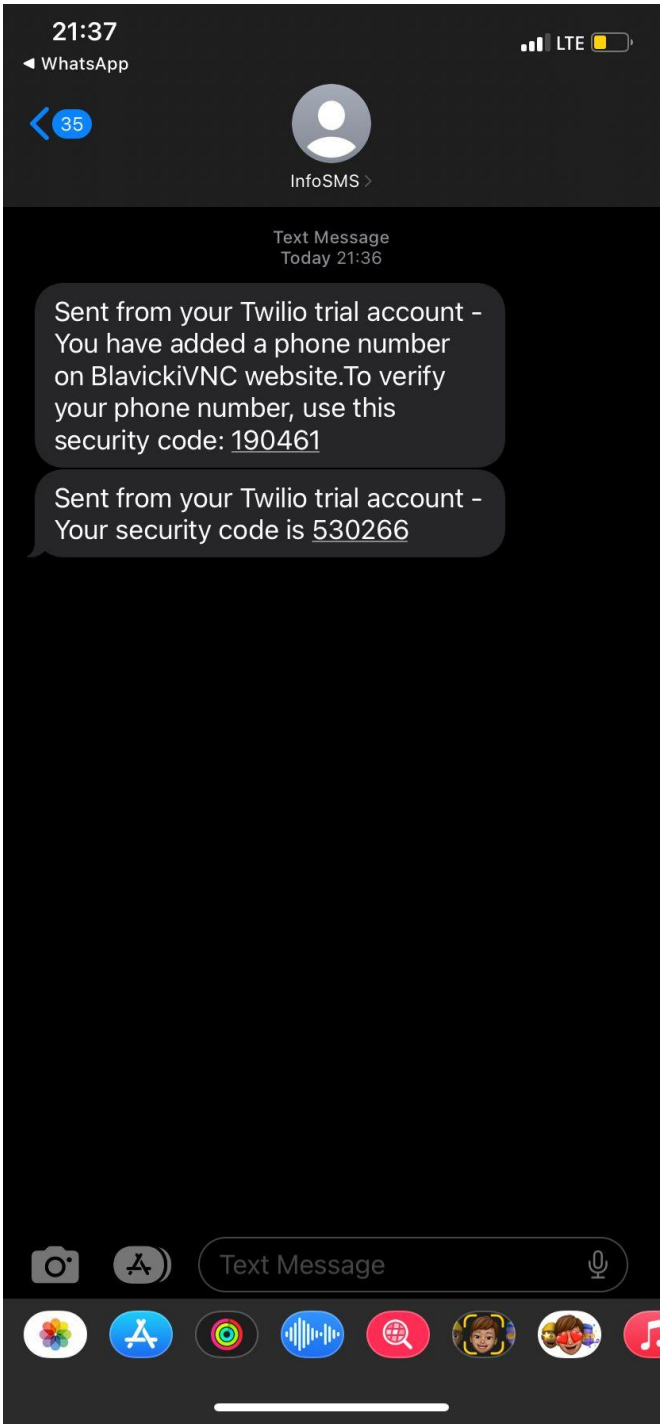

**Slika 5.19.** *Forma za odabir TFA pružatelja*

Odabirom željenog pružatelja i pritiskom tipke „Submit“, korisniku se na odabrani način šalje sigurnosni kod te ga se preusmjerava na zaslon gdje je potrebno unijeti poslani sigurnosni kod. Izgled poslanog maila može se vidjeti na slici 5.20.



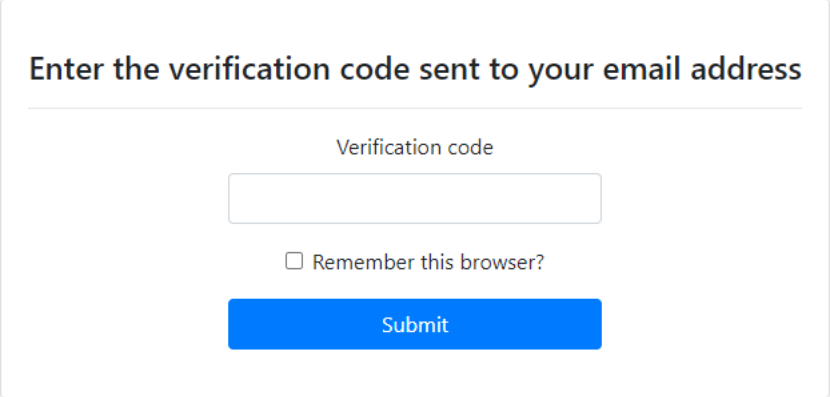

**Slika 5.20.** *Mail sa sigurnosnim kodom za prijavu*

Ako je korisnik odabrao da mu se sigurnosni kod pošalje pomoću SMS poruke, na mobilni uređaj poslana mu je SMS poruka koju je moguće vidjeti na slici 5.21.. Važno je za napomenuti da je druga vidljiva SMS poruka koja je poslana za vrijeme procesa prijave. Prva vidljiva SMS poruka poslana je tijekom procesa dodavanja telefonskog broja te će biti opisana u narednom dijelu.




**Slika 5.21.** *SMS poruka sa sigurnosnim kodom za prijavu*

Rečeno je da se korisnika preusmjerava na zaslon gdje je potrebno unijeti poslani sigurnosni kod. Navedeni zaslon moguće je vidjeti na slici 5.22.. Poruka na zaslonu varira zavisno o tome je li sigurnosni kod poslan putem email adrese ili telefonskog broja, no zbog trivijalnosti promjene prikazana je samo jedna varijanta.



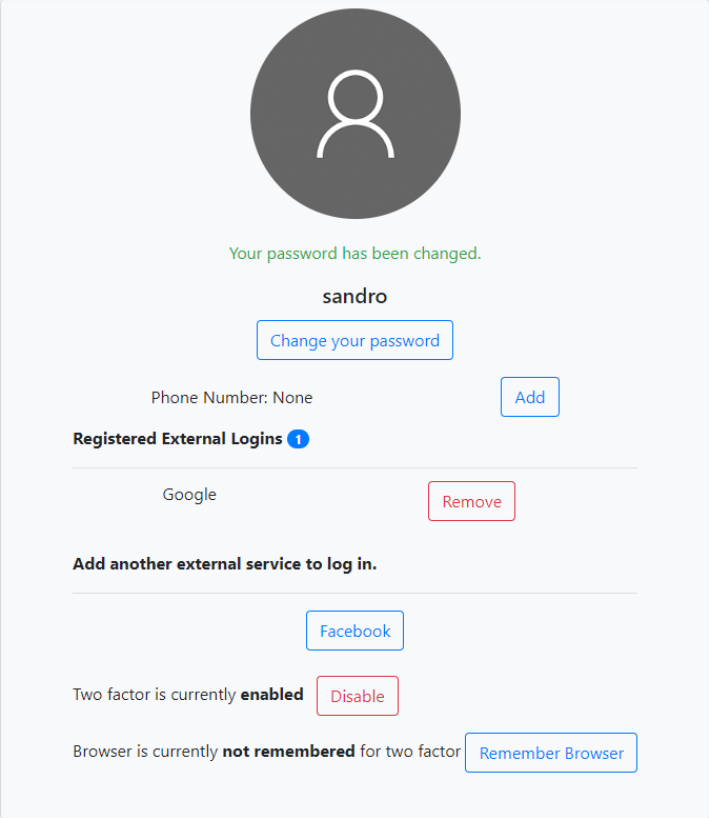

**Slika 5.22.** *Forma za unos sigurnosnog koda za prijavu*

Nakon unosa ispravnog sigurnosnog koda te pritiska tipke „Submit“, korisnika se prijavljuje i preusmjerava na početni zaslon aplikacije.

## 5.3. ManageController

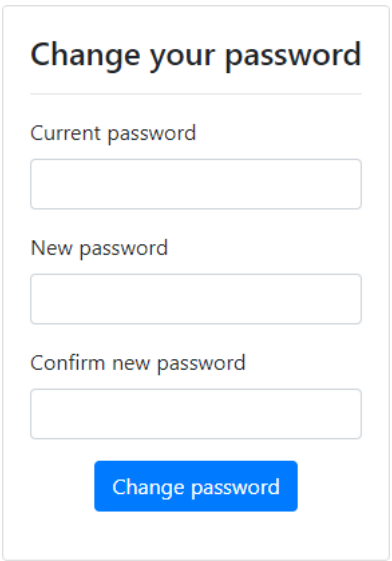
Zasloni *Manage* kontrolera odgovorni su za korisnički račun i upravljanje njime. Sadržaj koji je prikazan na zaslonu varira zavisno o tome kako je korisnik postavio svoj račun te kako se registrirao u aplikaciji. Sve varijacije opisane su u narednom dijelu, no kako bi opisivanje zaslona bilo jednostavno, različite varijacije samo su opisane. Na zaslonu se također prikazuju i poruke potvrde koje korisniku daju povratnu informaciju da je uspješno izvršio radnju.

Prvi dio korisnikove stranice sadrži sekciju koju je moguće vidjeti na slici 5.23.. U spomenutoj sekciji korisnik ima mogućnost postaviti podatke koji će utjecati na načine kojim se korisnik može prijaviti u aplikaciju.




**Slika 5.23.** *Sekcija osobnih podataka*

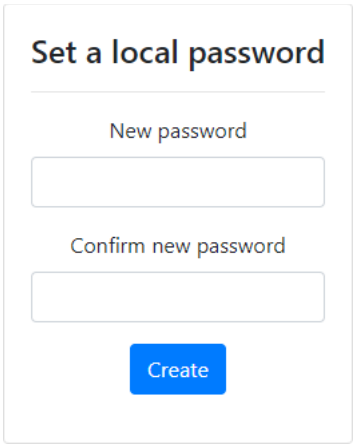
Zavisno o tome je li se korisnik registrirao putem email adrese i lozinke ili putem vanjskog pružatelja, tipka „Change your password" može se pronaći u dvije varijante. U slučaju gdje se korisnik registrirao u aplikaciju putem email adrese i lozinke, tipka će sadržavati tekst „Change your password" koji je vidljiv na prethodnoj slici. Pritiskom na tipku, korisnika se preusmjerava na zaslon za promjenu lozinke koji je vidljiv na slici 5.24..




**Slika 5.24.** *Forma za mijenjanje lozinke*

Ako su podaci koje je korisnik upisao ispravni, korisnika se nakon pritiska na tipku „Change password“ preusmjerava nazad na korisničku stranicu te se na zaslonu prikazuje poruka potvrde.

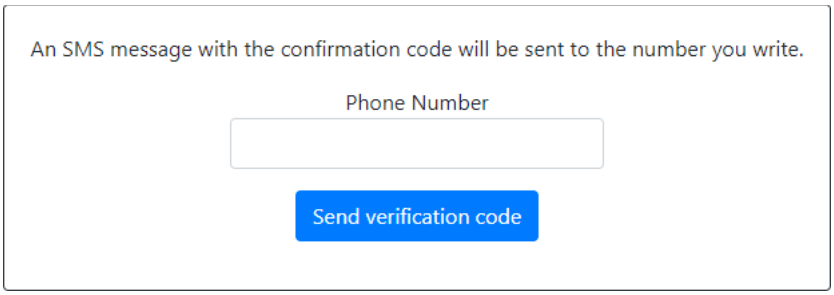
Druga varijanta sekcije za password se prikazuje kada korisnik proces registracije izvrši putem vanjskog pružatelja. Kako se registrirao putem vanjskog pružatelja, korisnik nema definiranu lozinku, te se tekst unutar tipke mijenja u „Set password“. Pritiskom na tipku „Set password“ korisnika se preusmjerava na zaslon za postavljanje lozinke. Navedeni zaslon moguće je vidjeti na slici 5.25..




**Slika 5.25.** *Forma za postavljanje lozinke*

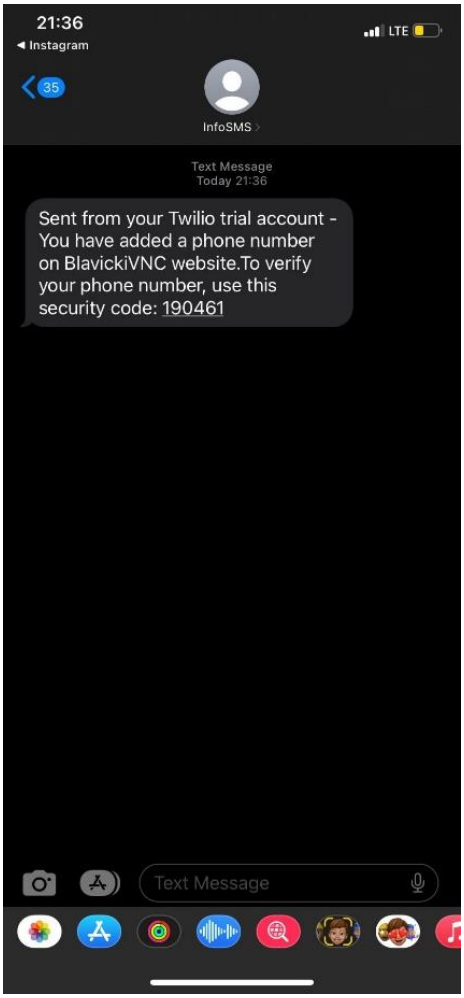
Nakon unosa ispravnih informacija, korisnika se preusmjerava na korisničku stranicu.

Sekcija za telefonski broj također se može pronaći u dvije varijante. Ako korisnik nije postavio telefonski broj, na zaslonu se prikazuje tipka „Add“. Nakon pritiska tipke, korisnika se preusmjerava na zaslon za unos telefonskog broja, koji je moguće vidjeti na slici 5.26..



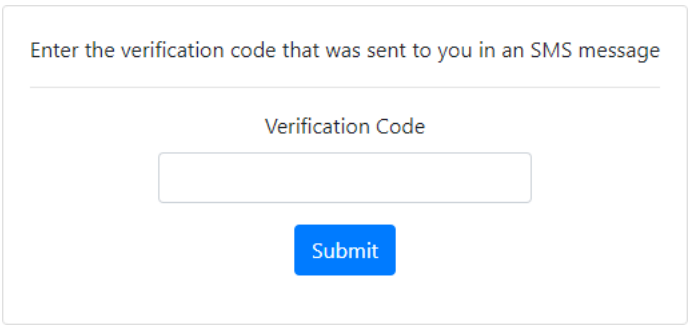

**Slika 5.26.** *Forma za unos telefonskog broja*

Nakon pritiska tipke „Send verification code”, sigurnosni kod šalje se na telefonski broj putem SMS poruke. Izgled poslane SMS poruke moguće je vidjeti na slici 5.27..



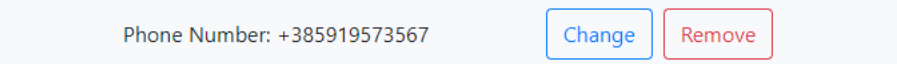

**Slika 5.27.** *SMS poruka sa sigurnosnim kodom za potvrdu telefonskog broja*

Uz slanje SMS poruke, korisnika se također preusmjerava na zaslon za unos sigurnosnog koda kako bi se potvrdio telefonski broj. Zaslon za potvrdu telefonskog broja moguće je vidjeti na slici 5.28.. Nakon unosa ispravnih podataka, korisnika se preusmjerava na korisničku stranicu te se prikazuje odgovarajuća poruka potvrde.




**Slika 5.28.** *Forma za unos sigurnosnog koda za potvrdu telefonskog broja*

Nakon unosa telefonskog broja, telefonski broj prikazuje se na korisničkoj stranici te se tipka „Add“ mijenja s tipkama „Change“ i „Remove“, kao što je vidljivo na slici 5.29..




**Slika 5.29.** *Sekcija telefonskog broja nakon dodavanja telefonskog broja*

Pritiskom na tipku „Change“, korisnika se preusmjerava na zaslon za unos telefonskog broja. Zaslon i proces unosa telefonskog broja prethodno je opisan.

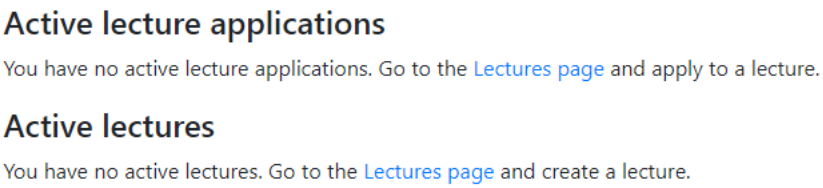
Pritiskom pritiska na tipku „Remove“, korisnikov telefonski broj se uklanja te se korisniku na zaslonu prikazuje poruka potvrde.

Sekcija za vanjske pružatelje za prijavu varira zavisno o tome je li korisnik povezao lokalni račun aplikacije s vanjskim pružateljem. U sekciji „Registered External Logins“ prikazani su svi korisnikovi registrirani vanjski pružatelji. Ako se korisnik registrirao putem vanjskog pružatelja, a nije postavio lozinku na lokalnom računu, korisniku nije omogućeno ukloniti vanjskog pružatelja s popisa registriranih vanjskih pružatelja. U slučaju da je korisnik postavio lozinku, na zaslonu mu se prikazuje tipka „Remove“, kao što je vidljivo na slici 5.23.. Pritiskom tipke „Remove“, vanjski pružatelj za prijavu se uklanja te se korisniku na zaslon prikazuje poruka potvrde.

Sekcija za *two-factor authentication* razlikuje se zavisno o tome je li korisnik omogućio navedenu funkcionalnost, te se prikladno tome prikazuju tipke „Enable“ tj. „Disable“. Pritiskom tipke „Enable“, korisnik omogućava autentikaciju u dva koraka, naglašeni tekst mijenja se u “enabled”, tipka mijenja boju u crvenu te se tekst mijenja u “Disable”. Pritiskom tipke “Disable” događa se suprotna radnja. Korisnik onemogućava funkcionalnost, tipka se mijenja nazad u “Enable” tipku te se naglašeni tekst mijenja u “Disabled”.

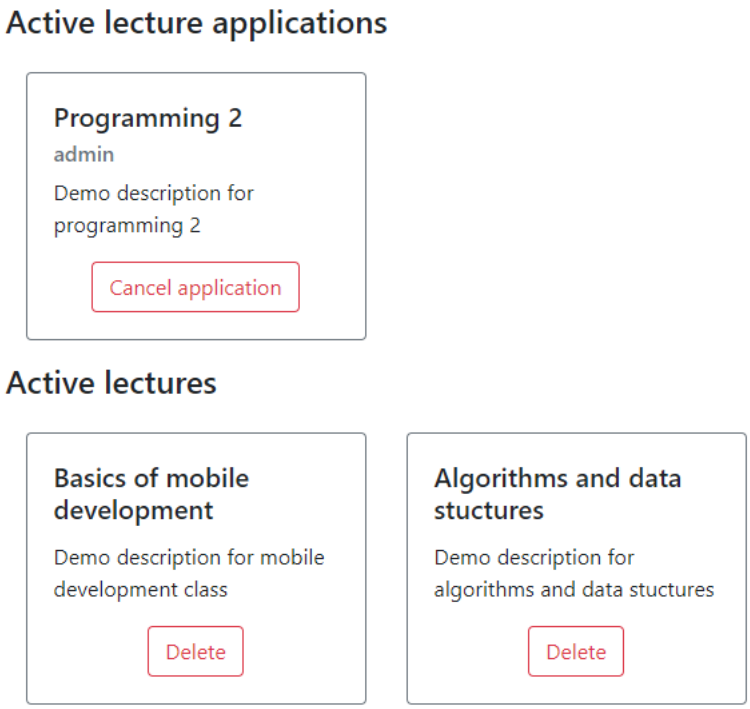
Sekcija “Remember Browser” mijenja izgled na sličan način kao i sekcija za *two-factor authentication* te radi jednostavnosti razlike neće biti opisana u detalje.

Korisnička stranica također sadrži i sekciju za predavanja. Na slici 5.30. prikazana je sekcija za predavanja u situaciji gdje korisnik ima ulogu administratora ili moderatora.




**Slika 5.30.** *Prazna sekcija predavanja*

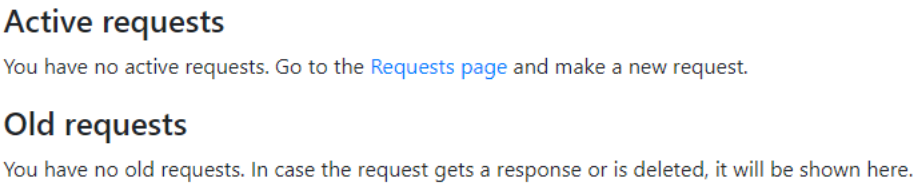
Ako korisnik nema trenutno aktivnih predavanja niti prijava na predavanja, korisniku se na zaslonu prikazuje poruka koja ga navodi da kreira predavanje tj. prijavi se na predavanje. Kako samo korisnici s ulogom administratora ili moderatora mogu kreirati predavanje, korisnicima koji nemaju ulogu administrator ili moderator sekcija „Active lectures“ neće biti prikazana. Ako se korisnik prijavio na neko predavanje i ima vlastita predavanja kreirana, na zaslonu će se prikazati prijave i predavanja kao što je vidljivo na slici 5.31..




**Slika 5.31.** *Popunjena sekcija predavanja*

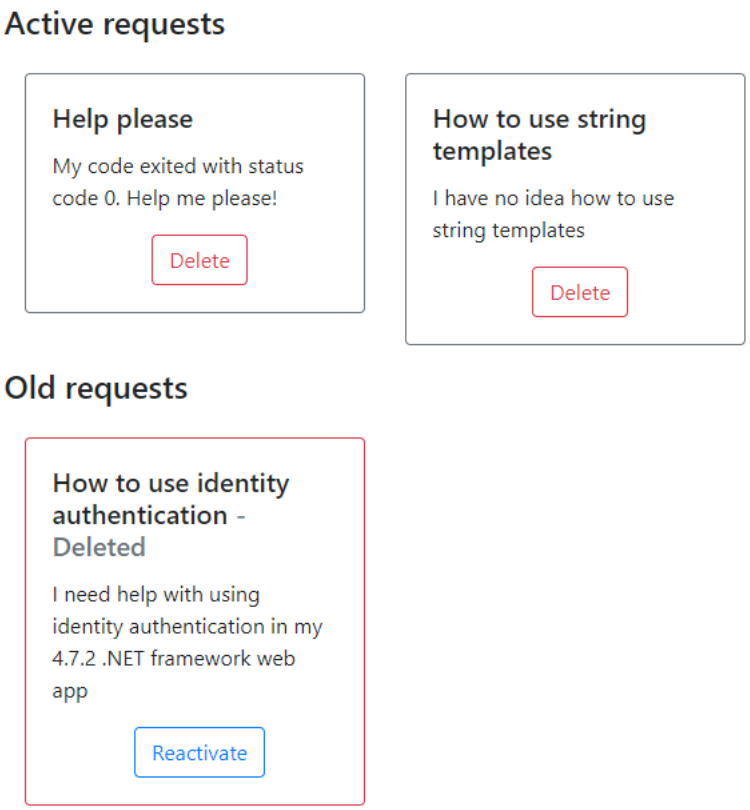
Pritiskom na tipku „Cancel application“, korisnik briše svoju prijavu na predavanje. Korisnik pritiskom na tipku „Delete“ briše vlastito predavanje.

Posljednja sekcija koja se nalazi na korisničkoj stranici je sekcija zahtjeva. Kao što je vidljivo na slici 5.32., ako korisnik nema kreiranih zahtjeva, korisniku se prikazuje poruka koja ga navodi da kreira zahtjev za pomoć.




**Slika 5.32.** *Prazna sekcija zahtjeva za pomoć*

Nakon kreiranja zahtjeva, korisnikovi zahtjevi prikazani su kao što je vidljivo na slici 5.33.. Zahtjevi za pomoć na koje su se korisnici s ulogom administratora ili moderatora odazvali prikazuju se u sekciji „Old request“. Stare zahtjeve za pomoć moguće je reaktivirati. Navedena funkcionalnost omogućena je u slučaju da korisnik s ulogom administrator ili moderator nije uspio pomoći korisniku te je korisniku i dalje potrebna pomoć.

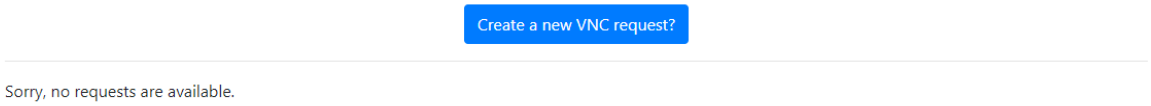



**Slika 5.33.** *Popunjena sekcija zahtjeva za pomoć*

Pritiskom na tipku „Delete“, korisnik briše zahtjev za pomoć te se zahtjev premješta u „Old requests“ sekciju. Pritiskom na tipku „Reactivate“, korisnik reaktivira zahtjev za pomoć te se reaktivirani zahtjev za pomoć može pronaći u sekciji „Active requests“.

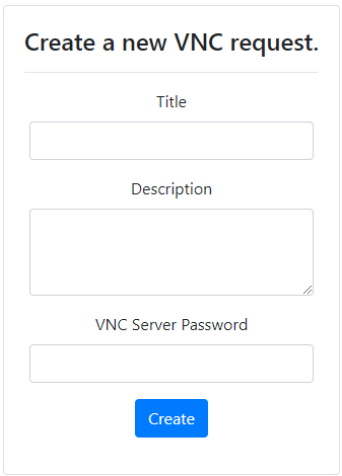
## 5.4. RequestController

*RequestController* odgovoran je za prikazivanje zaslona vezanih za VNC zahtjevima za pomoć. Početnu stranicu bez aktivnih zahtjeva može se vidjeti na slici 5.34..




**Slika 5.34.** *Prazna početna stranica Request kontrolera*

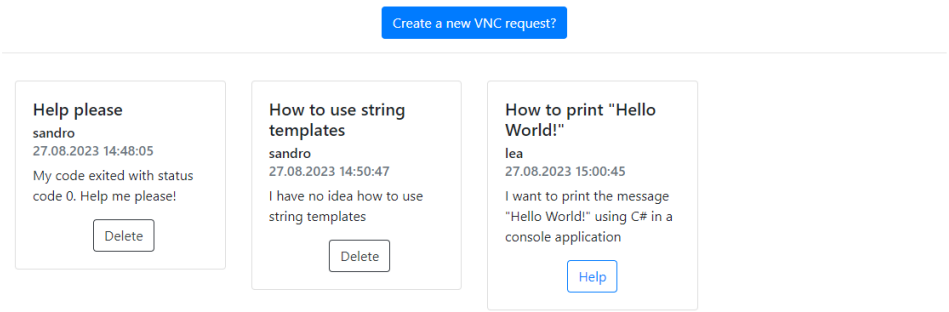
Pritiskom na tipku „Create a VNC request?", korisnika se preusmjerava na zaslon za kreiranje VNC zahtjeva za pomoć. Zaslon za kreiranje zahtjeva za pomoć moguće je vidjeti na slici 5.35..




**Slika 5.35.** *Forma za kreiranje zahtjeva za pomoć*

Nakon unosa potrebnih podataka i pritiskom na tipku „Create", korisnika se preusmjerava nazad na početni zaslon *Request* kontrolera gdje je moguće vidjeti kreirani zahtjev za pomoć.

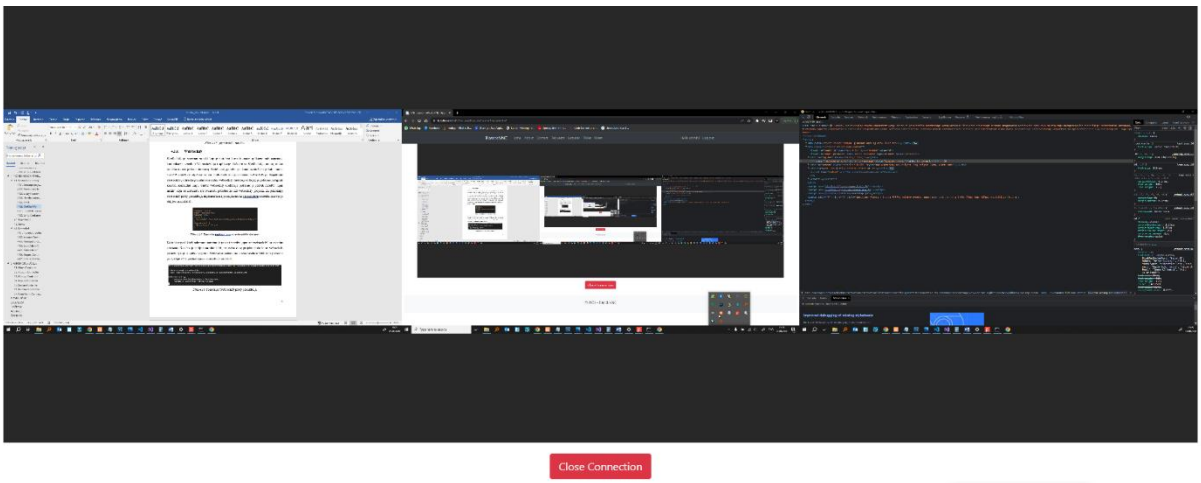
Ako postoje aktivni zahtjevi za pomoć, početni zaslon *Request* kontrolera izgledat će kao on a slici 5.36..




**Slika 5.36.** *Popunjena početna stranica Request kontrolera*

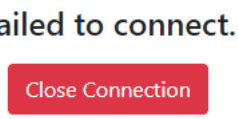
Korisnici imaju mogućnost obrisati vlastiti zahtjev za pomoć pritiskom na tipku „Delete“. Korisnici koji imaju ulogu administrator ili moderator imaju mogućnost odazvati se na zahtjev za pomoć pritiskom na tipku „Help“.

Pritiskom na tipku „Help“, korisnika se preusmjerava na zaslon na kojem je moguće upravljati udaljenim računalom. Prikaz zaslona za vrijeme upravljanjem udaljenim računalom moguće je vidjeti na slici 5.37.. Aplikacija je testirana na lokalnome serveru te je za vrijeme testiranja korišteno tri monitora, koje je moguće vidjeti na slici. Pritiskom tipke „Close connection“, zahtjev za pomoć se briše i korisnika se preusmjerava nazad na početni zaslon *Request* kontrolera.




**Slika 5.37.** *VNC sesija zahtjeva za pomoć*

Ako nije bilo moguće uspostaviti konekciju s udaljenim računalom, na zaslonu će se prikazati odgovarajuća poruka koju je moguće vidjeti na slici 5.38..

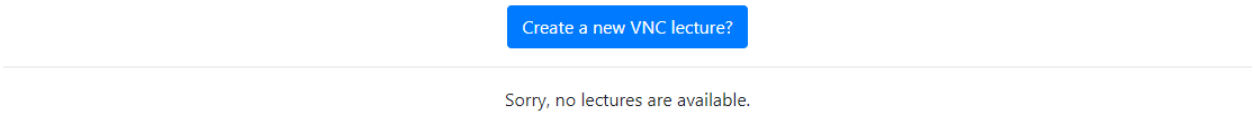



**Slika 5.38.** *Neuspjela konekcija sesije zahtjeva za pomoć*

## 5.5. LectureController

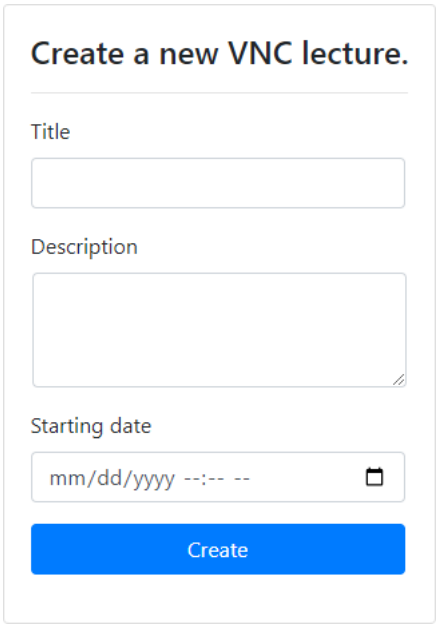
*LectureController* odgovoran je za sve zaslone vezane za VNC predavanja.

Početnu stranicu bez aktivnih predavanja moguće je vidjeti na slici 5.39.. Važno je za napomenuti da je tipka „Create a new VNC lecture?“ vidljiva samo za korisnike s ulogom administrator ili moderator.



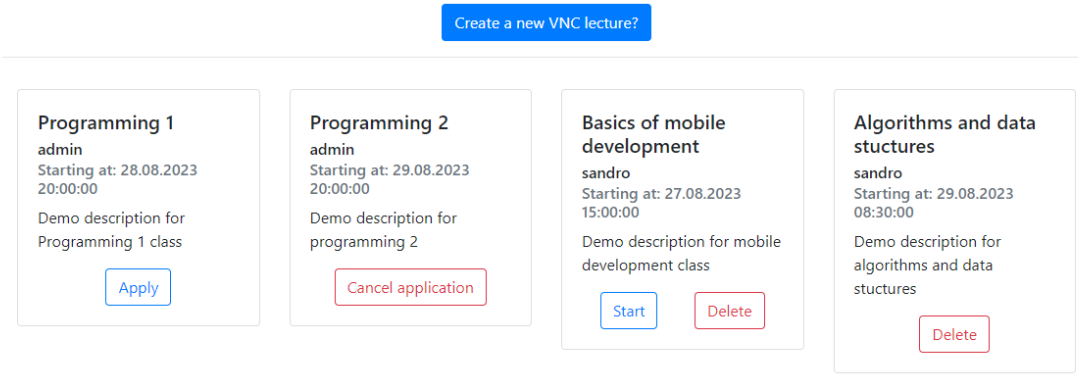

**Slika 5.39.** *Prazna početna stranica Lecture kontrolera*

Pritiskom na tipku „Create a new VNC lecture?“, korisnika se preusmjerava na zaslon za kreiranje VNC predavanja. Navedeni zaslon moguće je vidjeti na slici 5.40.. Nakon što korisnik unese podatke te pritisne na tipku „Create“, korisnika se preusmjerava nazad na početni zaslon *Lecture* kontrolera.



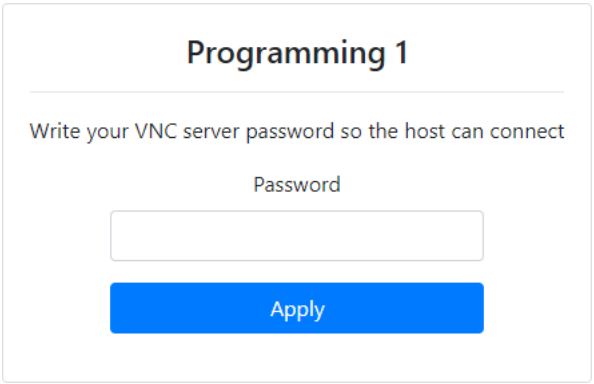

**Slika 5.40.** *Forma za kreiranje predavanja*

Ako postoje aktivna predavanja, početni zaslon *Lecture* kontrolera izgleda kao na slici 5.41.. Kao što se vidi na slici. Korisnik ima mogućnost obrisati vlastita predavanja pritiskom na tipku „Delete“. Važno je za napomenuti da korisnici s ulogom administrator imaju pravo obrisati bilo koje predavanje, neovisno o tome jesu li ga oni kreirali. Pritiskom na tipku „Cancel application“, korisnik može poništiti svoju prijavu na VNC predavanje.



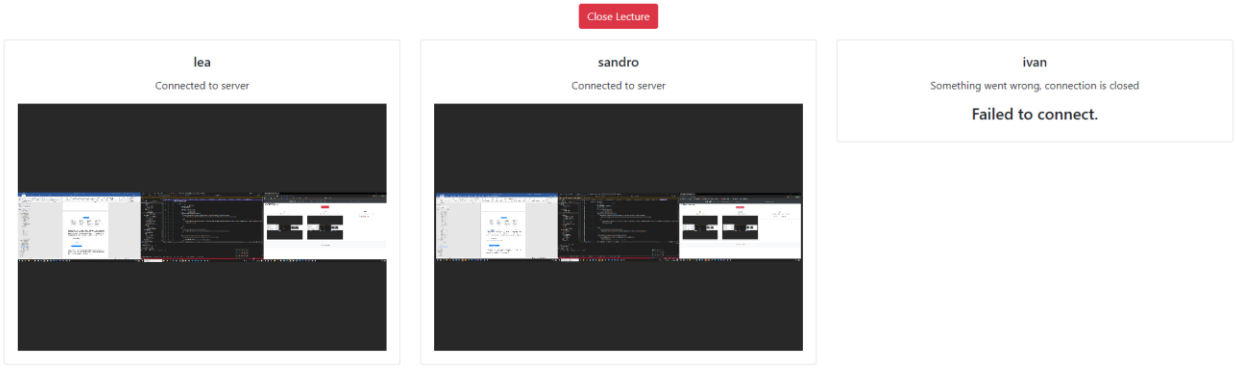

**Slika 5.41.** *Popunjena početna stranica Lecture kontrolera*

Pritiskom na tipku „Apply", korisnika se preusmjerava na zaslon za prijavu na VNC predavanje. Navedeni zaslon moguće je vidjeti na slici 5.42.. Kako bi se osoba koja drži predavanje mogla spojiti na korisnikovo računalo, od korisnika se traži šifra pomoću koje će se konekcija ostvariti. Pritiskom na tipku "Apply", korisnika se preusmjerava nazad na početni zaslon *Lecture* kontrolera.



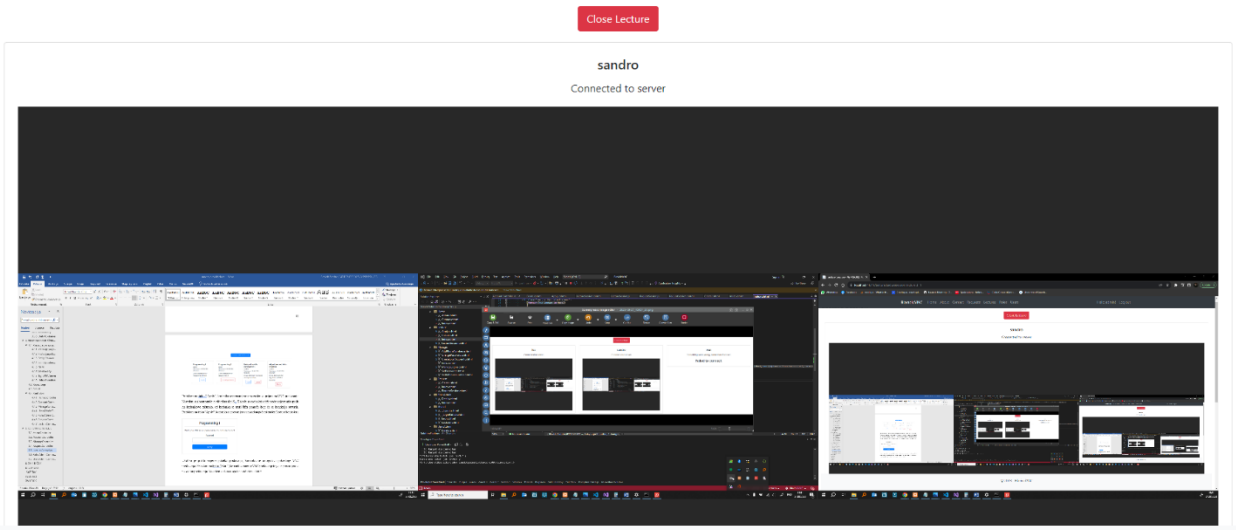

**Slika 5.42.** *Forma za prijavu na predavanje*

Ako je prošlo vrijeme početka predavanja, korisniku se omogućava pokretanje VNC predavanja. Pritiskom na tipku "Start", korisnik započinje VNC predavanje te ga se preusmjerava na zaslon predavanja. Navedeni zaslon moguće je vidjeti na slici 5.43.. Ako nije moguće ostvariti konekciju s nekim od udaljenih računala, odgovarajuća poruka prikazuje se u kartici korisnika s kojim nije moguće ostvariti konekciju.




**Slika 5.43.** *Sesija predavanja*

Kada su kartice minimizirane, predavač može samo gledati zaslone udaljenih računala. Pritiskom na karticu računala kojim želi upravljati, prikaz zaslona udaljenog računala se povećava te predavač dobiva mogućnost upravljanja udaljenim računalom, što je moguće vidjeti na slici 5.44.. Ako predavač želi zaustaviti kontrolu udaljenog računala, potrebno je pritisnuti karticu kako bi se ponovno minimizirala.



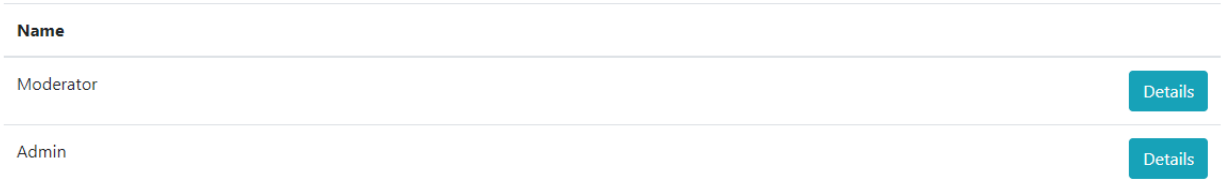

**Slika 5.44.** *Kontroliranje zaslona tijekom sesije predavanja*

Pritiskom na tipku “Close Lecture”, korisnik gasi VNC predavanje te ga se preusmjerava nazad na početni zaslon *Lecture* kontrolera.

## 5.6. RolesAdminController

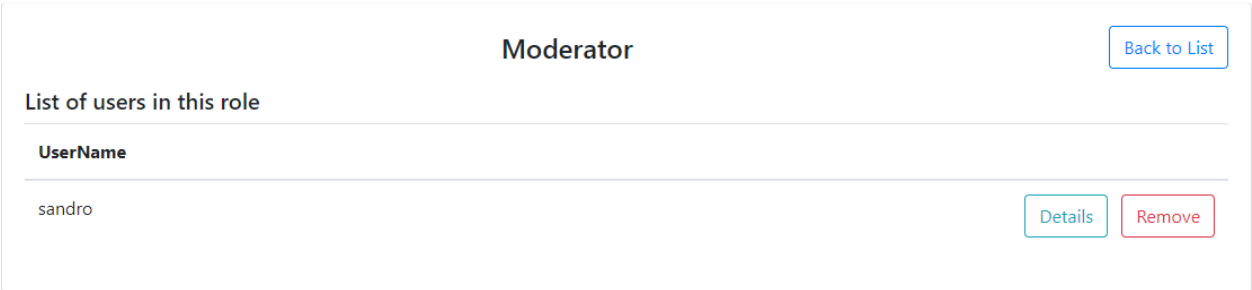
*RolesAdminController* odgovoran je za upravljanje ulogama unutar aplikacije. Važno je za napomenuti da su zasloni *RolesAdmin* kontrolera dostupni samo korisnicima koji imaju ulogu administrator.

Početni zaslon *RolesAdmin* kontrolera moguće je vidjeti na slici 5.45..




**Slika 5.45.** *Početna stranica RolesAdmin kontrolera*

Da bi korisnik vidio detalje uloge, potrebno je pritisnuti tipku „Details“ od odgovarajuće uloge koju želi vidjeti. Pritiskom tipke “Details”, korisnika se preusmjerava na zaslon gdje su prikazani detalji uloge. Navedeni zaslon moguće je vidjeti na slici 5.46..



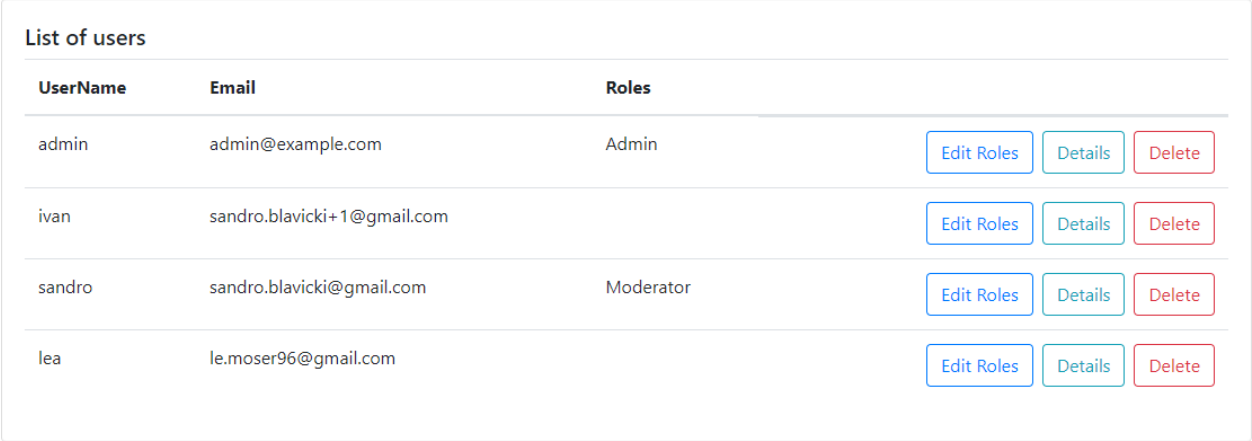

**Slika 5.46.** *Stranica detalja uloge*

Na stranici detalja uloge može se vidjeti sve korisnike koji pripadaju toj ulozi. Pritiskom na tipku “Details”, korisnika se preusmjerava na stranicu detalja korisnika, koja će biti opisana u narednom poglavlju. Pritiskom na tipku “Remove”, odabranome korisniku uklanja se uloga. Pritiskom na tipku “Back to List”, korisnika se preusmjerava nazad na početni zaslon *RolesAdmin* kontrolera.

## 5.7. UsersAdminController

*UsersAdminController* odgovoran je za zaslone pomoću kojih se upravlja korisnicima u aplikaciji. Važno je za napomenuti da, kao i kod *RolesAdmin* kontrolera, zaslonima *UsersAdmin* kontrolera moguće je pristupiti samo ako korisnik ima ulogu administrator.

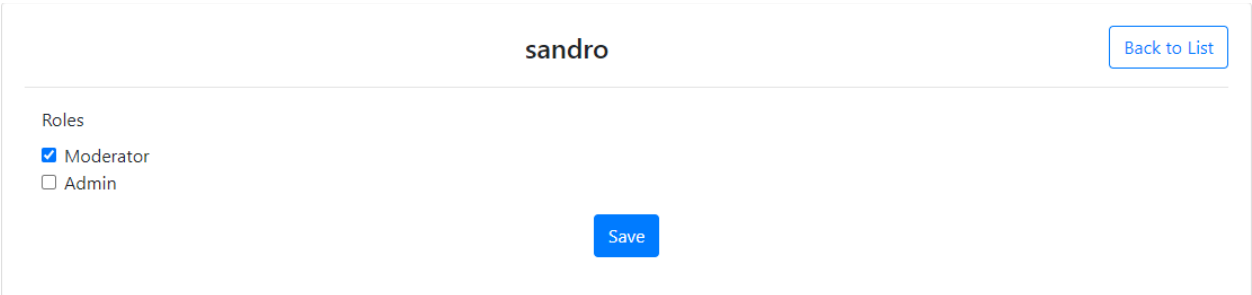
Početni zaslon *UsersAdmin* kontrolera moguće je vidjeti na slici 5.47..




**Slika 5.47.** *Početna stranica UsersAdmin kontrolera*

Pritiskom na tipku „Delete“, korisnik s ulogom administrator može obrisati odabranog korisnika iz aplikacije.

Pritiskom na tipku „Edit Roles”, korisnika se preusmjerava na zaslon za odabir uloga za željenog korisnika. Navedeni zaslon moguće je vidjeti na slici 5.48.. Pritiskom na tipku „Save“, korisnika se preusmjerava nazad na početni zaslon *UsersAdmin* kontrolera. Pritiskom na tipku „Back to List“ također korisnika preusmjerava nazad na početni zaslon *UsersAdmin* kontrolera.

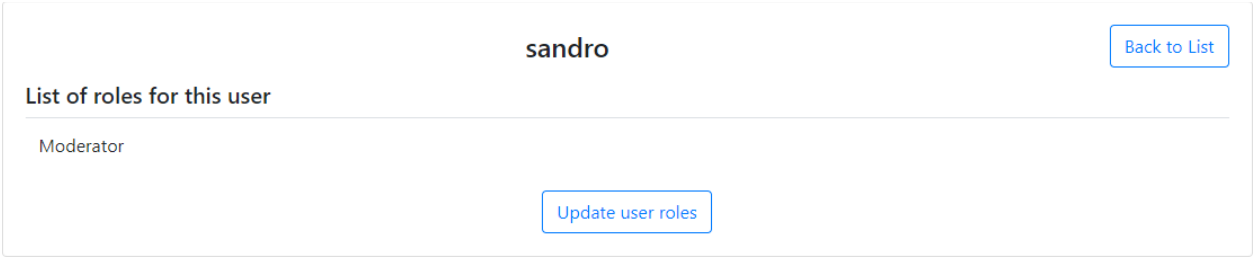



**Slika 5.48.** *Forma za mijenjanje korisnikovih uloga*

Pritiskom tipke „Details“ na početnom zaslonu *UsersAdmin* kontrolera, korisnika se preusmjerava na stranicu detalja odabranog korisnika.

Stranica detalja korisnika sadrži tri sekcije, a to su sekcija općih informacija o korisniku, sekcija s predavanjima i sekcija sa zahtjevima za pomoć.

Na slici 5.49. moguće je vidjeti sekciju općih informacija o korisniku. Unutar sekcije prikazane su korisnikove uloge, korisničko ime, te tipke „Back to List“ i „Update user roles“.

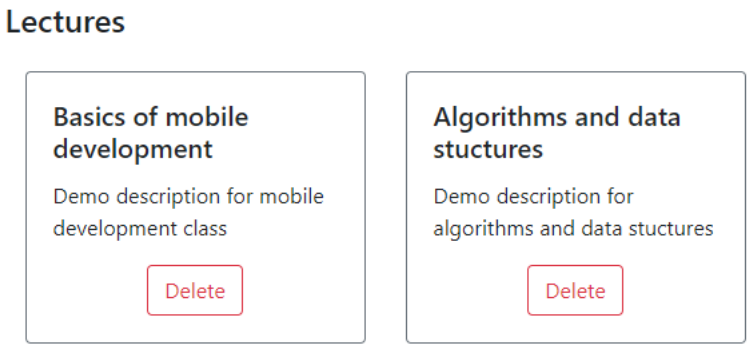



**Slika 5.49.** *Sekcija općih informacija na stranici detalja korisnika*

Pritiskom na tipku „Back to List“, korisnika se preusmjerava na početni zaslon *UsersAdmin* kontrolera.

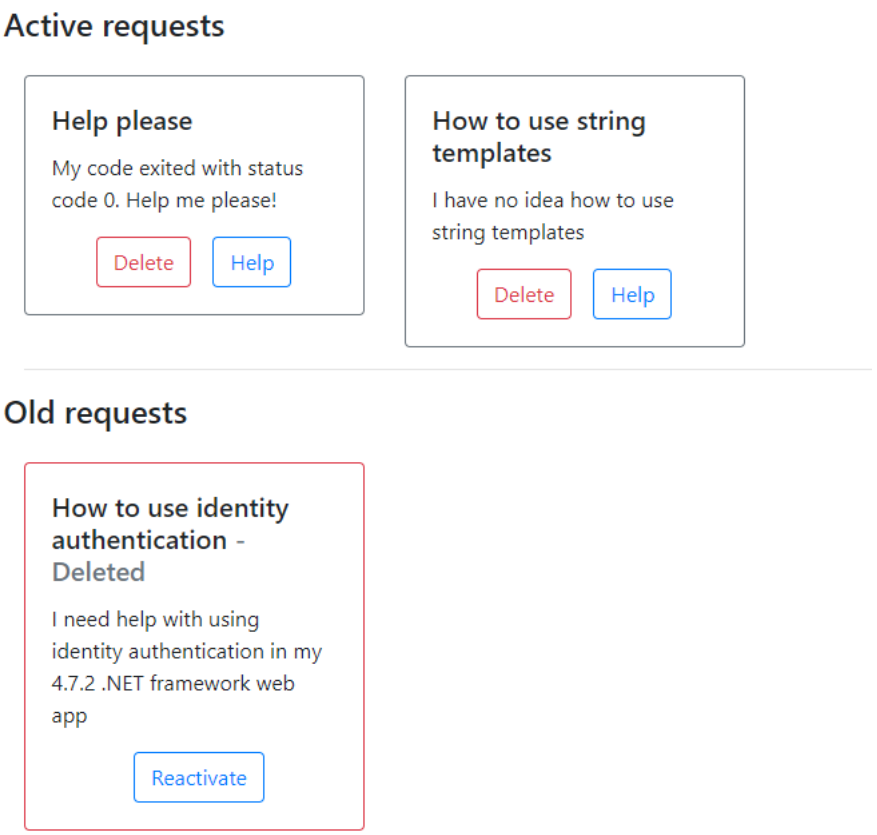
Pritiskom na tipku „Update user roles“, korisnika se preusmjerava na zaslon za odabir korisničkih uloga, koji je opisan u ranijem poglavlju.

Druga sekcija je sekcija o korisnikovim predavanjima. Važno je za napomenuti da se ova sekcija prikazuje samo ako odabrani korisnik ima ulogu administrator ili moderator. Navedenu sekciju moguće je vidjeti na slici 5.50.. Korisnici koji imaju ulogu administrator imaju pravo obrisati bilo koje predavanje, stoga pritiskom na tipku „Delete“, odabrano predavanje se briše.




**Slika 5.50.** *Sekcija predavanja na stranici detalja korisnika*

Posljednja sekcija na zaslonu je sekcija za VNC zahtjeve za pomoć. U navedenoj sekciji prikazani su svi aktivni i stari zahtjevi za pomoć koje je korisnik kreirao, što je moguće vidjeti na slici 5.51..




**Slika 5.51.** *Sekcija zahtjeva za pomoć na stranici detalja korisnika*

Admin korisnik može obrisati zahtjev za pomoć pritiskom na tipku „Delete“, reaktivirati zahtjev pritiskom na tipku „Reactivate“ ili se odazvati na zahtjev pritiskom na tipku „Help“.

# 6. ZAKLJUČAK

Diplomski rad detaljno opisuje proces izrade web aplikacije koja pruža mogućnost daljinskog upravljanja računalima koristeći VNC (RFB) protokol. Diplomski rad opisuje način korištenja .NET framework tehnologije kako bi se ostvarila željena funkcionalnost. Kako je sigurnost korisnika veliki prioritet kod aplikacija koje pružaju mogućnost daljinskog upravljanja, u diplomskom radu demonstrira se kako se može izraditi sigurnu aplikaciju koristeći Identity autentikaciju. Diplomski rad dokazuje kako se koristeći vanjske pakete poput noVNC i WebSockify može kreirati korisne proizvode koji imaju potencijal promijeniti način na koji ljudi rade i educiraju se.

# LITERATURA


[1] All you need to know about VNC remote access technology, https://discover.realvnc.com/what-is-vnc-remote-access-technology, (stranica posjećena 22.6.2023.)

[2] What is Microsoft Teams, https://support.microsoft.com/en-us/topic/what-is-microsoft-teams-3de4d369-0167-8def-b93b-0eb5286d7a29, (stranica posjećena 22.06.2203.)

[3] Webex Meetings, https://www.webex.com/suite/meetings.html, (stranica posjećena 22.06.2203.)

[4] What is TeamViewer? Remote Work Made Easy, https://blog.invgate.com/what-is-teamviewer (stranica posjećena 22.06.2203.)

[5] RealVNC Connect, https://www.realvnc.com/en/connect (stranica posjećena 02.10.2023.)

[6] AnyDesk, https://anydesk.com/en (stranica posjećena 02.10.2023.)

[7] What is .NET Framework? https://dotnet.microsoft.com/en-us/learn/dotnet/what-is-dotnet-framework, (stranica posjećena 22.06.2203.)

[8] Entity Framework, https://learn.microsoft.com/en-us/aspnet/entity-framework, (stranica posjećena 23.06.2203.)

[9] What is TightVNC? https://www.tightvnc.com/intro.php, (stranica posjećena 23.06.2203.)

[10] noVNC – the open source VNC client, https://novnc.com/info.html, (stranica posjećena 23.06.2203.)

[11] https://github.com/novnc/websockify, (stranica posjećena 23.06.2203.)

# SAŽETAK

Ovaj diplomski rad bazire se na ASP.NET web aplikaciji koja pruža funkcionalnost VNC Viewer aplikacije. Aplikacija je izrađena koristeći .NET Framework tehnologiju. U radu je opisan način rada RFB i VNC tehnologija te prikazana primjena VNC tehnologije unutar web okruženja. Opisan je proces implementacije i uporabe Microsoftovog Identity sustava za autentikaciju. Aplikacija podržava tri korisničke uloge te sadrži sustav autorizacije korisnika.

**Ključne riječi**: .NET Framework , Identity, RFB, uloge, VNC

## ABSTRACT

This thesis is based on an ASP.NET web application that provides the functionality of the VNC Viewer application. The application was made using .NET Framework technology. The paper describes the way RFB and VNC technologies work and shows the application of VNC technology within the web environment. The process of implementing and using Microsoft's Identity system for authentication is described. The application support three user roles and contains a user authorization system.



**Keywords:** .NET Framework , Identity, RFB, roles, VNC

# ŽIVOTOPIS

Sandro Blavicki rođen je 03. prosinca 1998. godine u Slavonskom Brodu. Osnovnu školu pohađao je u Orašju, Monterreyu i Slavonskom Brodu gjde ju i završava u osnovnoj školi Bogoslav Šulek. Nakon završene osnovne škole upisuje prirodoslovno-matematičku gimnazivju Matija Mesić u Slavonskom Brodu koju završava 2017. godine. Nakon srednje škole upisuje preddiplomski studij računarstva na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija u Osijeku. Nakon završetka preddiplomskog studija upisuje diplomski studij računarstva, smjer programsko inženjerstvo, na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija u Osijeku. Za vrijeme diplomskog studija odlazi na ERASMUS+ stručnu praksu u firmu Infineon Technologies u Villach, Austriji. Od srpnja 2023. godine radi u Infineonu.

_____

Sandro Blavicki

# PRILOZI

Prilog 1. Diplomski rad u docx formatu

Prilog 2. Diplomski rad u PDF formatu

Prilog 3. Programsko rješenje BlavickiVNC aplikacije:
https://github.com/sooondro/BlavickiVNC