

## SEMINARSKI RAD - PROGRAMIRANJE ZA INTERNET

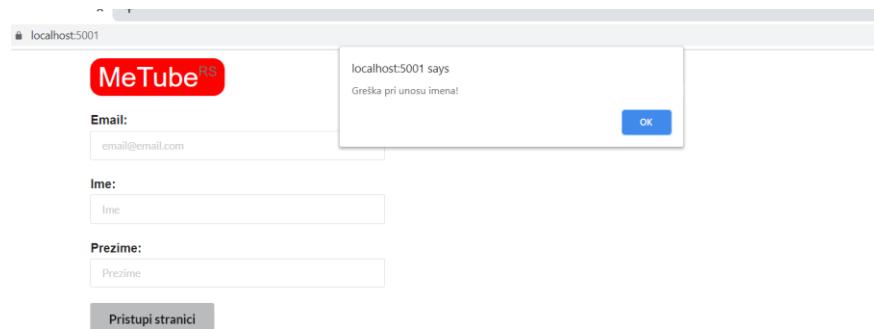
### Opis aplikacije

Ovaj seminarski rad treba da predstavlja sajt za pregled video snimak sa YouTube socijalne mreže. Pored toga, u pozadini ova aplikacija će čekati na okidač - pretraženi tekst koji u sebi ima reč koja ima veze sa fiktivnim likom iz serije "Bela lađa" Srećkom Šojićem. Kada se ta akcija desi, aplikacija će aktivirati metodu za slanje mejla (pozivnica za učlanjenje u fiktivnu političku stranku SZR - Stranka Zdravog Razuma) korisniku. Pre slanja mejla će se izvršiti upis podataka o korisniku u bazu kako bi se pratilo kome je poslat već mejl da ne bi došlo do ponavljanja.

Kada se pokrene aplikacija, pojaviće se forma za unos podataka. Tu je potrebno uneti email (ukoliko korisnik sa tom adresom već postoji, ostala polja će se sama popuniti), ime i prezime:

The screenshot shows a web browser window with the URL 'localhost:5001'. The page title is 'MeTube RS'. It contains three input fields: 'Email:' with the value 'email@email.com', 'Ime:' with the value 'ime', and 'Prezime:' with the value 'Prezime'. Below the fields is a grey button labeled 'Pristupi stranici'.

Ukoliko ne unesemo ime, pojaviće se prozor koji nas o tome obaveštava, a ista je situacija i za prezime i email, kao i email koji nije ispravan:



localhost:5001

**MeTube RS**

Email:  
email@email.com

Ime:  
Jovan

Prezime:  
Prezime

Pristupi stranici

localhost:5001 says  
Greška pri unosu prezimena!

OK

localhost:5001

**MeTube RS**

Email:  
email@email.com

Ime:  
Jovan

Prezime:  
Jovanovic

Pristupi stranici

localhost:5001 says  
Niste uneli email!

OK

localhost:5001

**MeTube RS**

Email:  
jovan.jovanovic@email.com

Ime:  
Jovan

Prezime:  
Jovanovic

Pristupi stranici

localhost:5001 says  
Greška pri unosu email-a!

OK

Nakon unošenja podataka, pojaviće se stranica sa video snimcima, slična YouTube stranici:

The screenshot shows a web browser window with the URL "localhost:5001". The main content is a video player from "MeTube RS" featuring a profile of a man with glasses and the word "OSTAVKA!" in large red letters. The video has a play button and social sharing options ("Гледајте к..." and "Дели"). Below the video is a caption: "POČELO JE! VUČIĆ PODNOŠI OSTAVKU NA VIDOVĐAN?!-SRBI...". A search bar is visible above the video player.

On the right side, there are four news thumbnails:

- POČELO JE! VUČIĆ PODNOŠI OSTAVKU NA VIDOVĐAN?!-SRBI...** (with a photo of a man speaking)
- VULIN: SRBIJA JE REGIONALNA TENKOVSKA SILA!** (with a photo of a man in a military uniform)
- HITNO UPOMENJE RHMZ! Stiže nam VELIKA OLUJA, padaće KISA I GRAD! - Srbija Online** (with a photo of a lightning storm)
- FRANCUZI KREĆU U OFANZIVU! Makron isplanirao dijalog Beograda i Prištine! - Srbija Online** (with a photo of political figures)

At the bottom right, there is a "SRBIJA" logo and the text "Srbija Online".

Aplikacija sve video snimke povlači direktno sa YT sajta preko API-a koji omogućava Google. Početni "search" termin je "Srbija" i zbog toga možemo na početku videti ponuđene video snimke vezane sa Srbiju.

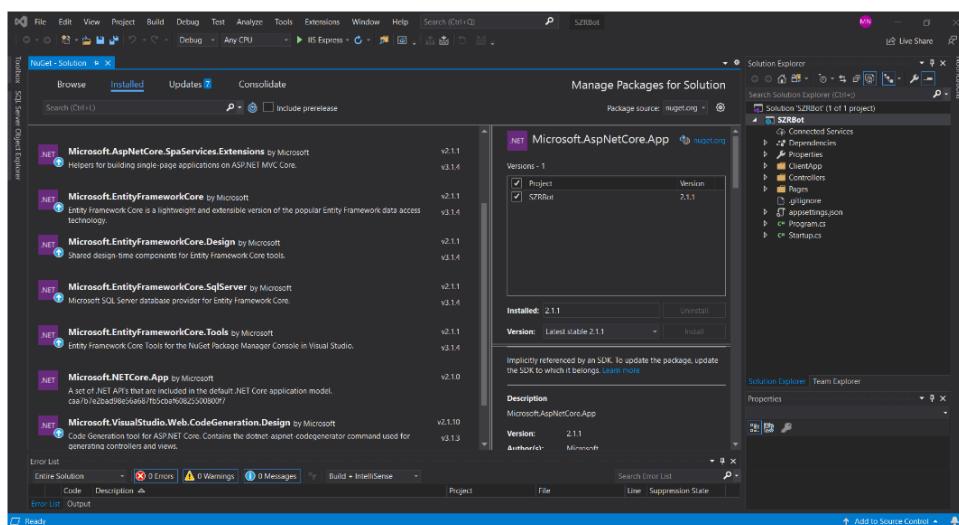
## Izrada aplikacije kroz korake

Za izradu aplikacije koristićemo okruženje Visual Studio 2019. Ova aplikacija je tipa ASP.NET Core i pisaćemo je u C# programskom jeziku. Koristićemo i framework-ove i biblioteke poput: React.js (za dizajn), Redux (za kreiranje akcija koje aktiviraju metode kontrolera), Entity Framework (za kreiranje i upotrebu baze podataka), Axios (za asinhronne zahteve Youtube-u) i Semantic-UI (za stilove).

Prvih 5 koraka su vezano za pripremu za rad.

1. Pokrenuti VS2019 i kreirati novi projekat. Izabrati ASP.NET Core Web Application. Zatim izabrati React.js and Redux kao template i označiti ASP.NET Core 2.1 verziju.
2. Kada je projekat kreiran, potrebno je instalirati dodatne pakete. Pomoću NuGet PM ćemo instalirati:
  - 1) Microsoft.EntityFrameworkCore (2.1.1);
  - 2) Microsoft.EntityFrameworkCore.SqlServer (2.1.1);
  - 3) Microsoft.EntityFrameworkCore.Tools (2.1.1);
  - 4) Microsoft.EntityFrameworkCore.Design (2.1.1);
  - 5) Microsoft.VisualStudio.Web.CodeGeneration.Design (2.1.10).

Nakon toga naši paketi će izgledati ovako:



3. Nakon dodavanja paketa pomoću NuGet-a, potrebno je otvoriti Command Line -> Developer Command Prompt, i uneti komandu "npm install --save axios" i pritisnuti enter. Na ovaj način smo instalirali axios biblioteku u projekat i trebalo bi da se pojavi ovakav prozor:

```
C:\X\Full Stack Development\Projects\SZRBot>npm install --save axios
npm WARN saveError ENOENT: no such file or directory, open 'C:\X\Full Stack Development\Projects\SZRBot\package.json'
npm notice created a lockfile as package-lock.json. You should commit this file.
npm WARN enoent ENOENT: no such file or directory, open 'C:\X\Full Stack Development\Projects\SZRBot\package.json'
npm WARN SZRBot No description
npm WARN SZRBot No repository field.
npm WARN SZRBot No README data
npm WARN SZRBot No license field.

+ axios@0.19.2
added 4 packages from 7 contributors and audited 4 packages in 1.273s
found 0 vulnerabilities

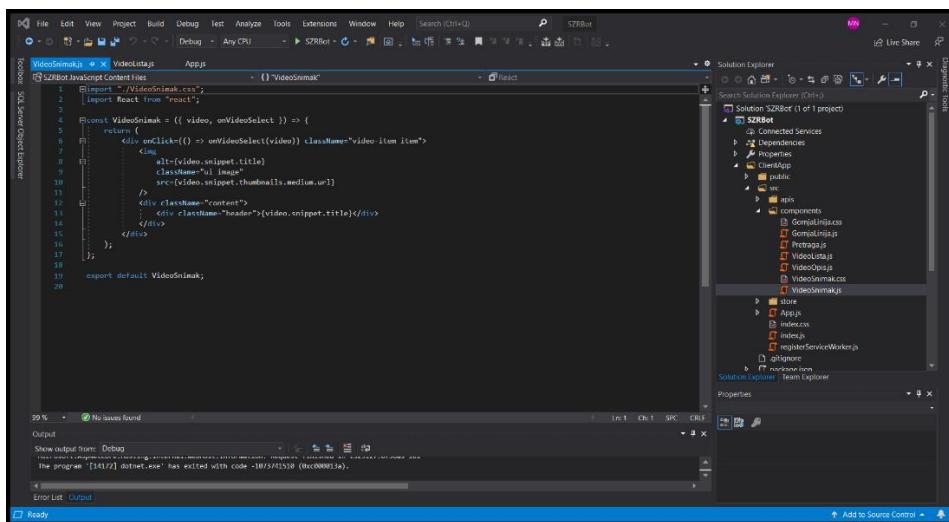
C:\X\Full Stack Development\Projects\SZRBot>
```

4. U Solution Explorer prozoru otvoriti ClientApp -> src -> components i izbrisati sve fajlove unutar njega jer su to nepotrebni fajlovi kreirani po default-u. Takođe, u folderu ClientApp -> src -> store izbrisati fajlove Counter.js i WeatherForecast.js. Unutar foldera Controllers potrebno je izbrisati fajl koji je tu: Sample.

5. Još jedna pripremna stavka je povezivanje aplikacije sa semantic-ui sajtom kako bi bilo omogućeno korišćenje stilskih klasa. Potrebno je dodati <link> tag u index.html fajl: <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/semantic-ui/2.4.1/semantic.min.css" />.

Nakon ovih 5 koraka, sledi 7 (6-12) koji će biti posvećeni komponentama i dizajnu aplikacije kao i učitavanju video snimaka sa YT platforme.

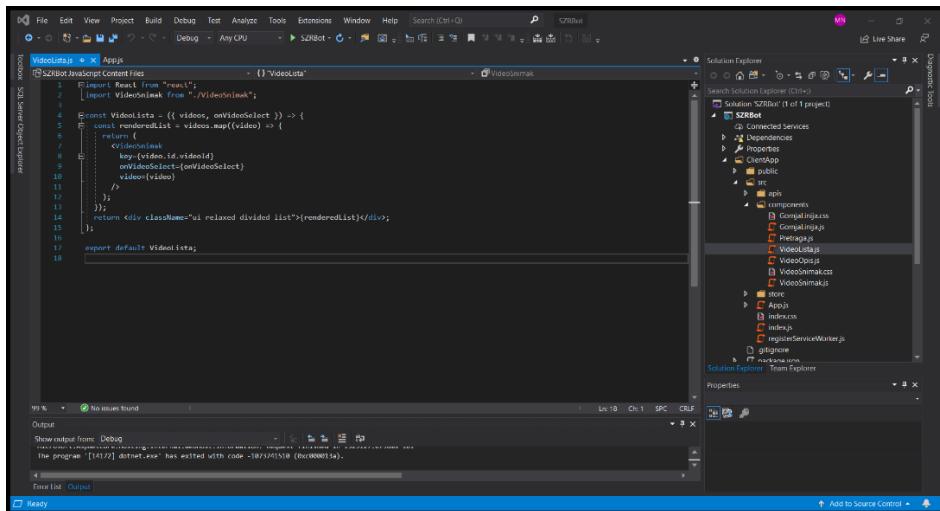
6. Nakon završene pripreme, prvi korak će biti dodavanja u folder components, fajlova vezanih za svaki video snimak pojedinačno, VideoSnimak.js (komponenta) i VideoSnimak.css (stil vezan za VideoSnimak komponentu):



Što se tiče VideoSnimak.css, kod je vrlo kratak:

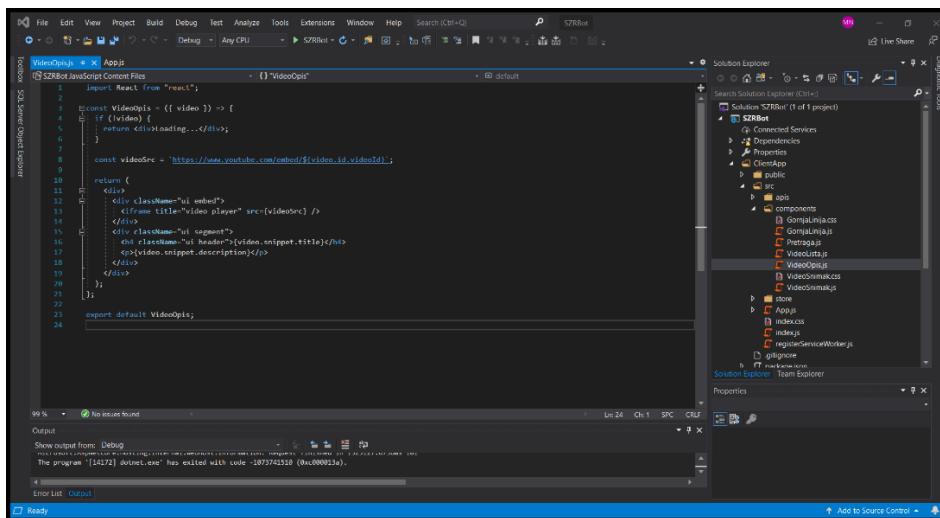
```
.video-item { display: flex !important; align-items: center !important; cursor: pointer; }
.video-item.item img { max-width: 180px; }
```

7. Sledeća komponenta je vezana za listu video snimaka (VideoSnimak komponenti). Kreirati fajl u components folderu pod imenom VideoLista.js i uneti kod sa slike:



```
File Edit View Project Build Debug Test Analyze Tools Extensions Window Help Search (Ctrl+Q) S78Bot
VideoLista.js  App.js
1 import React from "react";
2 import VideoSnimak from "./VideoSnimak";
3
4 const VideoLista = ({ videos, onVideoSelect }) => {
5   const renderedList = videos.map((video) => {
6     return (
7       <div key={video.id} onClick={onVideoSelect}>
8         <img alt={video.snippet.title}>
9         <VideoSnimak video={video}>
10        </VideoSnimak>
11      </div>
12    );
13  });
14
15  return <div className="ui relaxed divided list">{renderedList}</div>;
16}
17
18 export default VideoLista;
```

8. Sledeća na redu je komponenta VideoOpis koja izgleda ovako:



```
File Edit View Project Build Debug Test Analyze Tools Extensions Window Help Search (Ctrl+Q) S78Bot
VideoOpis.js  App.js
1 import React from "react";
2
3 const VideoOpis = ({ video }) => {
4   if (!video) {
5     return <div></div>;
6   }
7
8   const videoSrc = `https://www.youtube.com/embed/${video.id.videoId}`;
9
10  return (
11    <div>
12      <div className="ui embed">
13        <iframe title="video player" src={videoSrc} />
14      </div>
15      <div className="ui segment">
16        <div className="ui header">{video.snippet.title}</div>
17        <p>{video.snippet.description}</p>
18      </div>
19    </div>
20  );
21}
22
23 export default VideoOpis;
```

9. Komponenta koju sledeću treba kreirati vezana je za liniju za pretragu odnosno searchbar i zove se Pretraga:

```

1 import React from 'react';
2
3 class Pretraga extends React.Component {
4   state = { unos: '' };
5
6   onInputChange = (e) => {
7     this.setState({ unos: e.target.value });
8   };
9
10  onFormSubmit = (e) => {
11    e.preventDefault();
12    this.props.onFormSubmit(this.state.unos);
13  };
14
15  render() {
16    return (
17      <div className="search-bar__ul segment">
18        <form onSubmit={this.onFormSubmit} className="ui form">
19          <div className="field">
20            <label>Pretraga:</label>
21            <input
22              type="text"
23              value={this.state.unos}
24              onChange={this.onInputChange}
25            />
26          </div>
27        </form>
28      </div>
29    );
30  }
31
32  export default Pretraga;
33

```

10. Još je potrebno kreirati komponentu koja se odnosi na celu gornju traku. To će biti TopLine komponenta koja će imati svoj poseban stilski fajl TopLine.css i ova će biti smeštena u components folderu:

```

1 import React from 'react';
2 import './GornjaLinija.css';
3
4 class GornjaLinija extends React.Component {
5   render(props) {
6     return (
7       <div className="row">
8         <div className="column">
9           <div style={{ display: "inline-block" }}>
10             <h1 style={{ color: "white", background-color: "red", padding: "5px 10px 5px 10px", margin: "0px", border-radius: "10px" }}></h1>
11             <div style={{ float: "left" }}>
12               
13             </div>
14           </div>
15         </div>
16       </div>
17     );
18   }
19
20   export default GornjaLinija;
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
39

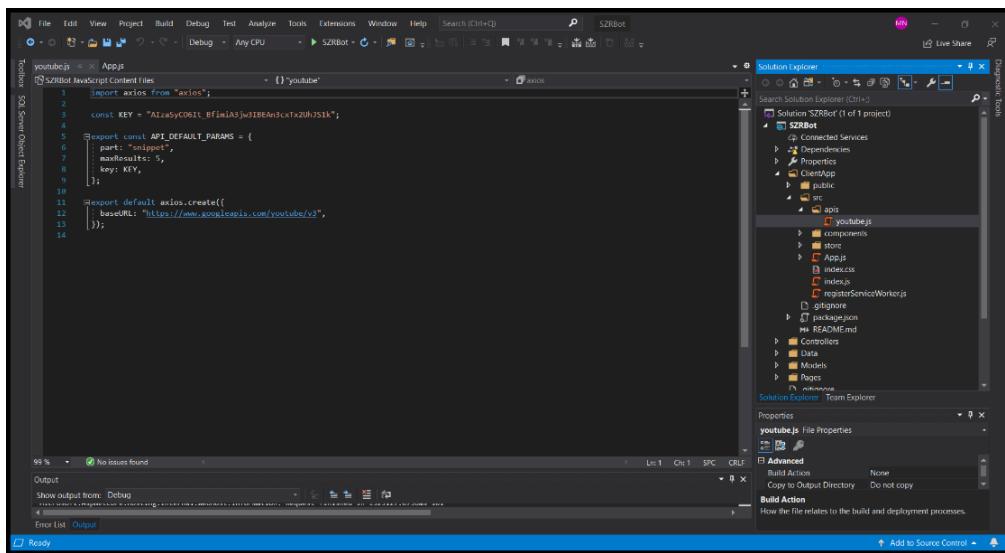
```

```

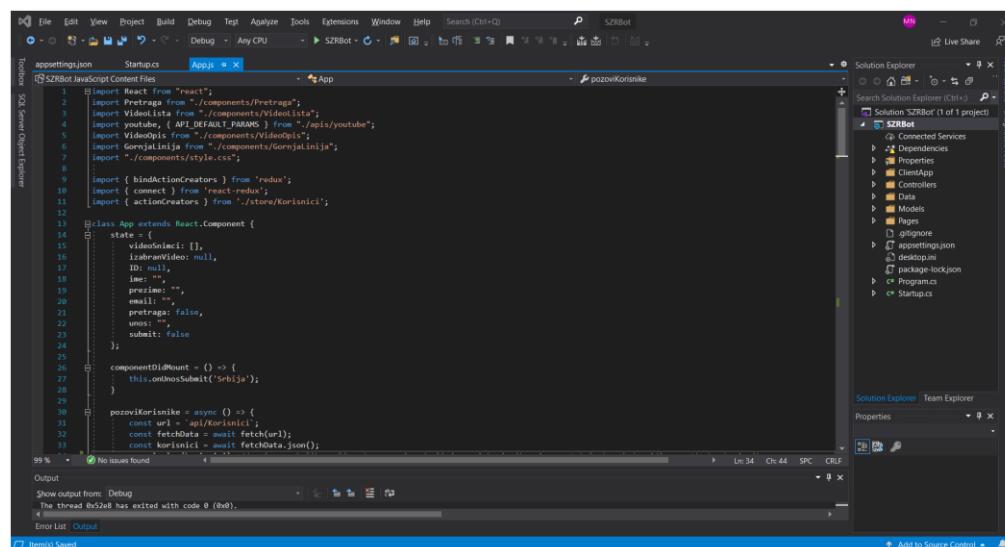
1 {
2   box-sizing: border-box;
3 }
4
5 .column {
6   float: left;
7   width: 50%;
8   padding: 10px;
9 }
10
11 :row-after {
12   content: "";
13   display: table;
14   clear: both;
15 }
16
17 .column img {
18   float: left;
19   top: 10px;
20   vertical-align: middle;
21   margin: 10px;
22 }
23
24 .column .top {
25   position: relative;
26   top: 10px;
27   margin: 10px;
28 }
29
30
31
32
33
34
35
36
37
38
39
39

```

11. Pre nego što uredimo App.js fajl, potrebno je u folderu src kreirati novi folder pod imenom apis u koji ćemo smestiti API za YouTube. Naime, potrebno je imati Google nalog i u pretraživaču pronaći sajt console.developers.google.com (Google API console sajt). Tu je potrebno kreirati novi projekat i nakon toga pritisnuti ENABLE APIs AND SERVICES i pronaći API pod imenom YouTube Data API v3. Nakon toga potrebno je kreirati kredencijale u sekciji Credentials (odgovoriti na pitanja i setapovati API). Ključ koji se izgeneriše potrebno je u fajlu youtube.js (u folderu apis) smestiti u posebnu konstantu pod imenom KEY. Taj svojevrsni ključ će biti jedina razlika u odnosu na dati string sa sledeće slike. Sve ostalo što je na slici će ostati isto (ključ je podešen na public tako da se može koristiti sa različitih računara ukoliko je neophodno):



12. Kao finalnu komponentu koja će obuhvatiti sve prethodne komponente i dati im smisao, potrebno je izmeniti App.js fajl tako da izgleda kao na slikama:



```

    31     const url = `api/korisnici`;
    32     const fetchData = await fetch(url);
    33     const korisnici = await fetchData.json();
    34     const korisniciId = //option: ukoliko zelimo da provjerimo da li je novi korisnik upisan u tabelu kao i da vidimo prethodne korisnike
    35     const email = e.target.value;
    36     for (var i = 0; i < korisnici.length; i++) {
    37       if (korisnici[i].email === this.state.email) postoji = true;
    38     }
    39     if (!postoji) {
    40       this.dodaKorisnik();
    41     }
    42   }
    43
    44   dodaKorisnik = () => {
    45     const korisnik = {ime: this.state.ime, prezime: this.state.prezime, email: this.state.email};
    46     this.props.dodaKorisnika(korisnik);
    47   }
    48
    49   onFormSubmit = async (unos) => {
    50     const response = await youtube.get(`search`, {
    51       params: {
    52         ...API_DEFAULT_PARAMS,
    53         q: unos,
    54       },
    55     });
    56     this.setState({
    57       videoSrcime: response.data.items,
    58       izabranVideo: response.data.items[0],
    59       unos: unos
    60     });
    61     if (unos === "Srbija") {
    62       if (this.proveraPretrage(unos)) this.pozovikKorisnika();
    63     }
    64   }
  
```

No issues found

Show output from: Debug  
The thread 0x52e8 has exited with code 0 (0x0).

Error List Output

```

    64   }
    65   proveraPretrage = (unos) => {
    66     const sojic = ["sojic", "sojic", "SOJIC", "SOJIC", "sojic", "sojic", "SOJIC", "SOJIC"];
    67     const srladja = ["srladja", "srladja", "SRЛАДЈА", "SRЛАДЈА"];
    68     const belaladjia = ["belaladjia", "belaladjia", "БЕЛАДЖИЈА"];
    69     const okidaci = [sojic.concat(srladja).concat(belaladjia)];
    70     var arr = unos.split(" ");
    71     var res = false;
    72     for (var i = 0; i < arr.length; i++) {
    73       if (okidaci[i].includes(arr[i])) {
    74         res = true;
    75       }
    76     }
    77     return res;
    78   }
    79
    80   onVideoSelect = (video) => {
    81     this.setState({izabranVideo: video});
    82   }
    83
    84   onFormaSubmit = (e) => {
    85     e.preventDefault();
    86     if (this.state.ime.length < 3) {
    87       alert('Unesite imena i prezimena');
    88       document.querySelector('#ime').setAttribute('style', 'border-color: red;');
    89     } else if (this.state.prezime.length < 1) {
    90       alert('Greška pri unosu prezimena');
    91       document.querySelector('#prezime').setAttribute('style', 'border-color: red;');
    92     } else if (this.state.email.length < 1) {
    93       alert('Niste uneli email');
    94       document.querySelector('#email1').setAttribute('style', 'border-color: red;');
    95     }
  
```

No issues found

Show output from: Debug  
The thread 0x52e8 has exited with code 0 (0x0).

Error List Output

```

    96     } else if (this.state.email.length < 1) {
    97       alert('Niste uneli email');
    98       document.querySelector('#email1').setAttribute('style', 'border-color: red;');
    99       document.querySelector('#ime').removeAttribute('style');
    100      document.querySelector('#prezime').removeAttribute('style');
    101      if (!this.state.email.includes('@')) {
    102        alert('Unesite email sa @ simbolom');
    103        document.querySelector('#email1').setAttribute('style', 'border-color: red;');
    104        document.querySelector('#prezime').removeAttribute('style');
    105        document.querySelector('#ime').removeAttribute('style');
    106      }
    107      this.setstate({submit: true});
    108    }
    109  }
    110
    111  onImeChange = (e) => {
    112    this.setState({ime: e.target.value});
    113  }
    114
    115  onPrezimeChange = (e) => {
    116    this.setState({prezime: e.target.value});
    117  }
    118
    119  onEmailChange = (e) => {
    120    this.setState({email: e.target.value});
    121    if (e.target.value.includes('@') && e.target.value.includes('.com')) {
    122      this.proveriPostojanje(e.target.value);
    123    }
    124  }
    125
    126  ...
  
```

No issues found

Show output from: Debug  
The thread 0x52e8 has exited with code 0 (0x0).

Error List Output

The screenshot shows the Visual Studio Code interface with the following details:

- Solution Explorer:** Shows the project "SZRBot (1 of 1 project)" with files like appsettings.json, Startup.cs, Program.cs, and Startup.csproj.
- Editor:** The file "appsettings.json" is open, displaying configuration code for a bot.
- Status Bar:** Shows "Ready" at the bottom left and "Ctrl + F5" as the active keybinding at the bottom right.

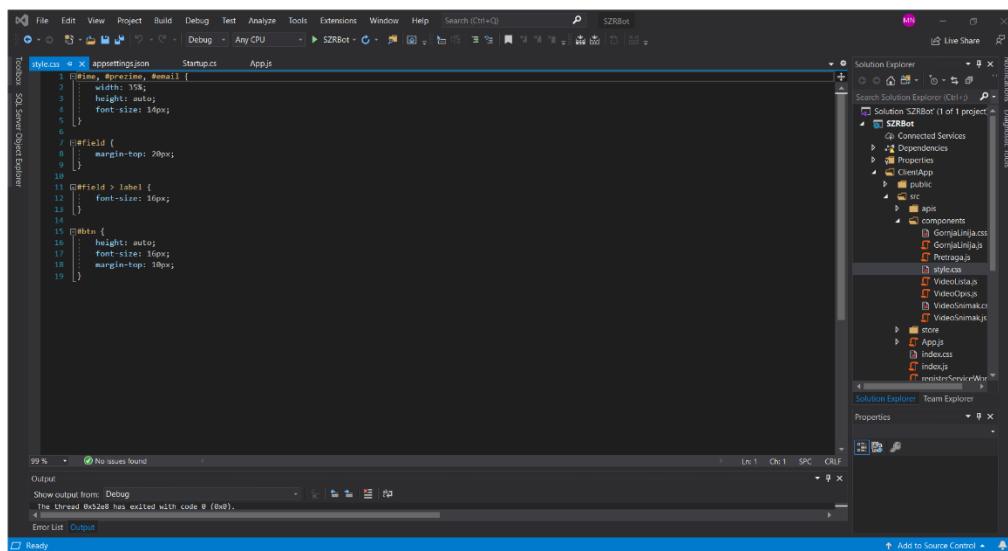
The screenshot shows the Microsoft Visual Studio interface with the following details:

- Code Editor:** The main window displays the file `app.js` from the project `SZRBot`. The code is a React component with state management and form handling logic.
- Solution Explorer:** Shows the project structure with files like `Startup.cs`, `app.js`, `settings.json`, and `Program.cs`.
- Task List:** Located at the bottom left, it shows "No issues found".
- Status Bar:** Shows "99 %", "Output", "Show output from: Debug", "The thread 0x526 has exited with code 0 (0x0).", and "Error List: Output".

The screenshot shows the Microsoft Visual Studio IDE interface. The code editor on the left displays a file named `index.js` containing React component code. The Solution Explorer on the right shows a project structure with files like `appsettings.json`, `Startup.cs`, and `index.js`. The status bar at the bottom indicates the code coverage is at 99%.

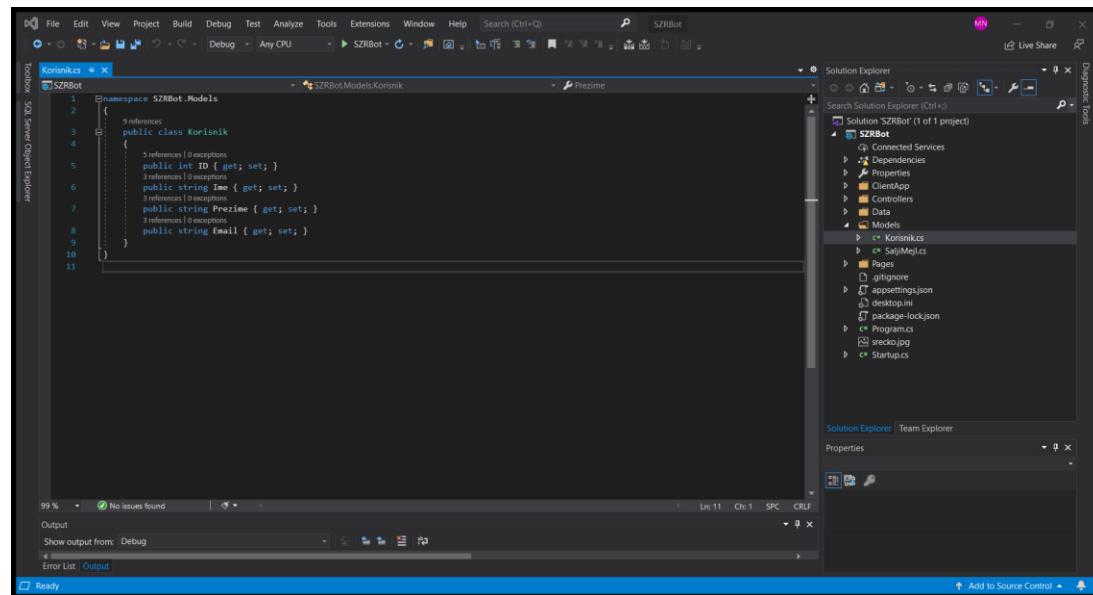
```
176         </div>
177         <div className="five wide column">
178             <VideoSelect
179                 onVideoSelect={(hi, onVideoSelect)}
180                 videos={this.state.videoSmic1}
181             />
182         />
183     </div>
184   </div>
185 </div>
186 </div>
187 <a href="https://icom8.com/icon/98957/user">User icon by Icons8</a>
188 </div>;
189
190 )
191
192 export default connect(
193   state => state.reducer,
194   dispatch = bindActionCreators(actionCreators, dispatch)
195 )(App);
196
```

Kod ove komponente još je samo potrebno dodati kratak stilski fajl style.css (za stilizovanje forme):



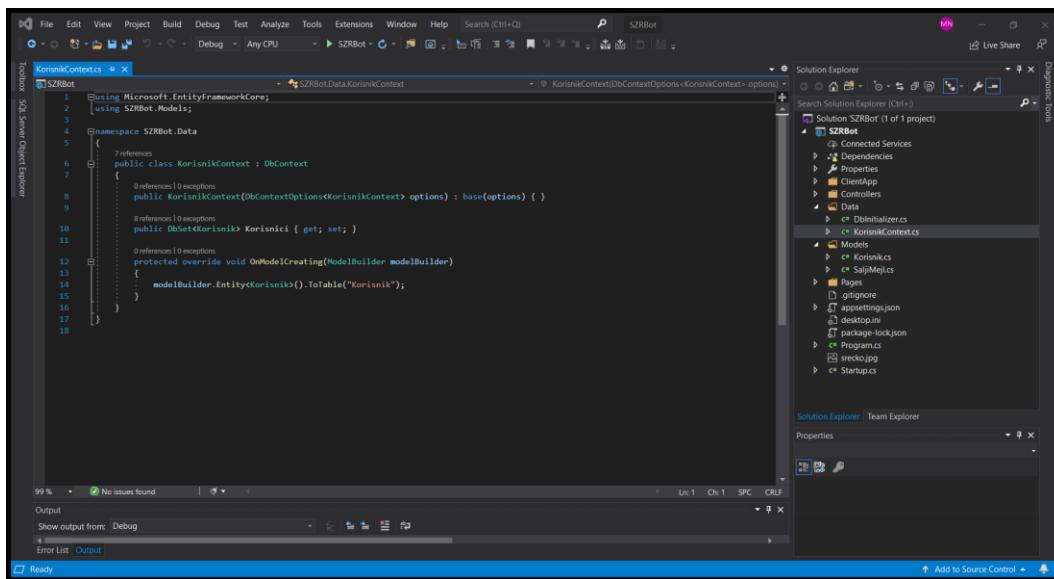
Kada je dizajn gotov, potrebno je okrenuti se bazi podataka. Za to nam služi Entity Framework i narednih 6 koraka (13-18) biće posvećeno kreiranju baze podataka sa tabelom za upis korisnika kojima je već poslat mejl.

13. Počemo sa principom code-first. To znači da će Entity kreirane klase kao i kontekst klasu pretvoriti u bazu podataka. Prvi korak je kreiranje foldera pod imenom Models unutar projekta. Nakon toga u njemu kreirati klasu pod Imenom Korisnik.cs i preuređiti je kao na slici:

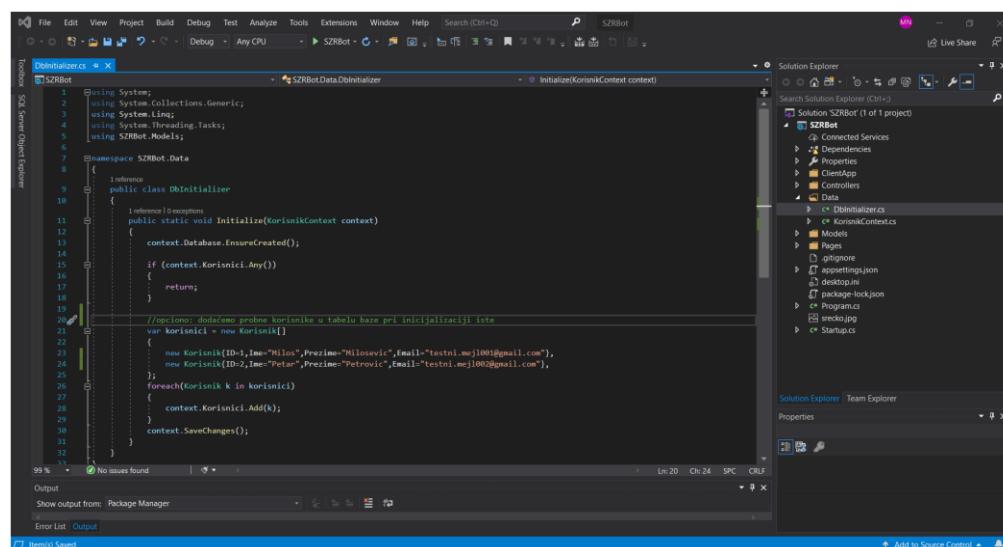


Ovako smo kreirali (kreiraće se pri prvom pokretanju aplikacije) ne samo klasu sa atributima već i kolone ID, Ime, Prezime i Email u tabeli Korisnik.

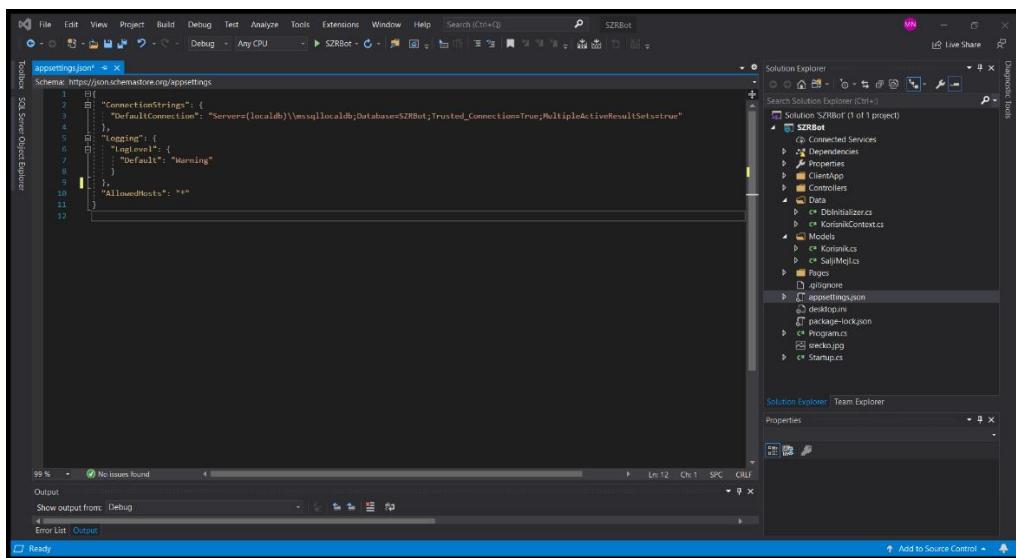
14. Nakon kreirane klase potrebno je napraviti novi folder u projektu pod imenom Data. U njega smestiti kontekst klasu pod imenom KorisnikContext koja će nasleđivati klasu DbContext:



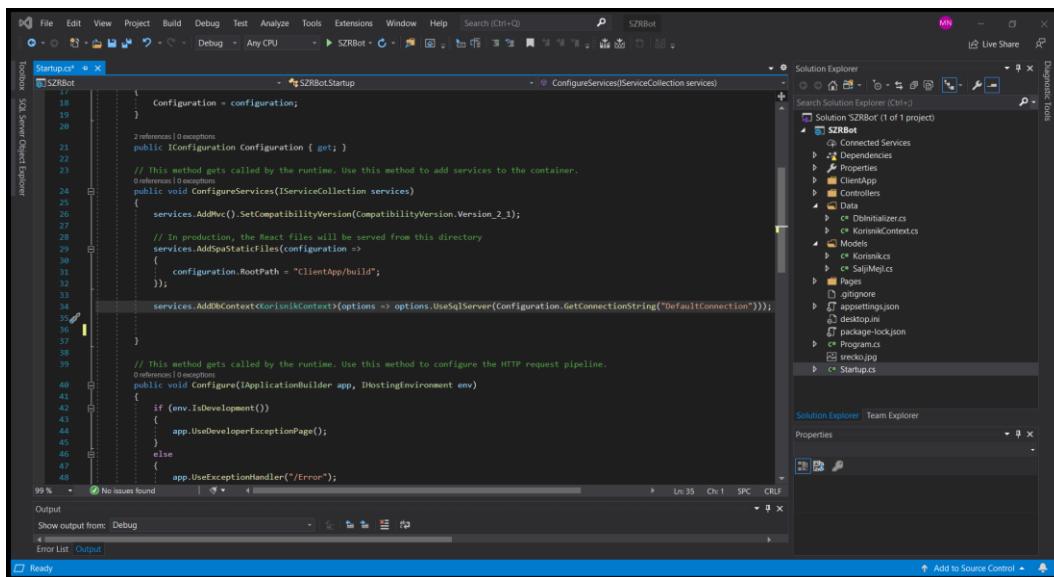
15. Sada je potrebno u istom folder Data napraviti klasu DbInitializer i ispisati kod:



16. Kada smo ovo sredili, moramo dodati konekcioni string u fajl appsettings.json. Ovaj konekcioni string će spajati aplikaciju sa bazom kreiranom na lokalnom serveru u folderu C:/Users. Srediti ovaj JSON fajl tako da izgleda ovako:



17. U fajlu Startup.cs potrebno je u funkciji ConfigureServices() dodati highlight-ovanu liniju koda kao na slici:



18. Nakon toga, u fajlu Program.cs potrebno je modifikovati Main metodu tako da izgleda kao na sledećoj slici:

```

using System;
using Microsoft.AspNetCore;
using Microsoft.AspNetCore.Hosting;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Logging;
using Microsoft.EntityFrameworkCore;

namespace SZRBot
{
    public class Program
    {
        public static void Main(string[] args)
        {
            var host = CreateWebHostBuilder(args).Build();

            using (var scope = host.Services.CreateScope())
            {
                var services = scope.ServiceProvider;
                try
                {
                    var context = services.GetRequiredService<KorisnikContext>();
                    DBInitializer.Initialize(context);
                }
                catch (Exception ex)
                {
                    var logger = services.GetRequiredService<ILogger<Program>>();
                    logger.LogError(ex, "Došlo je do greske prilikom učitavanja baze podataka.");
                }
            }

            host.Run();
        }
    }
}

```

Sada smo gotovi sa kreiranjem baze. Sledeća stvar koja nam je potrebna je kontroler sa metodama za upravljanje Http request-ovima (nama će biti bitni Get i Post).

19. Pre toga potrebno je napraviti klasu koju ćemo koristiti prilikom slanja mejla, tačnije za instanciranje objekta koji se odnosi na poruku (ima sve karakteristike specifične za mejl). Ta klasa se zove SaljiMejl i nalazi se u folderu Models:

```

namespace SZRBot.Models
{
    public class SaljiMejl
    {
        public string Primalac { get; set; }
        public string Subject { get; set; }
        public string Body { get; set; }
    }
}

```

20. Potreban nam je kontroler KorisniciController koji ćemo smestiti u folder Controllers. Kreiraćemo ga tako što ćemo desnim klikom na taj folder izabrati Add -> New Scaffolded Item... i izabrati Korisnik i KorisnikContext. U ovom kontroleru će se obavljati 3 bitna zadatka: čitanje postojećih korisnika iz tabele, dodavanje korisnika u tabelu i slanje mejla korisnicima koji su novi. Generisan kod ne treba brisati već samo izmeniti odgovarajuće metode: Ovako će ovaj fajl izgledati u kodu:

```

1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Net;
5 using System.Threading.Tasks;
6 using System.Web;
7 using System.Web.Mvc;
8 using Microsoft.AspNetCore.Http;
9 using Microsoft.AspNetCore.Mvc;
10 using Microsoft.EntityFrameworkCore;
11 using S2RBot.Data;
12 using S2RBot.Models;
13
14 namespace S2RBot.Controllers
15 {
16     [Route("api/[controller]")]
17     [ApiController]
18     public class KorisnikController : Controller
19     {
20         private readonly KorisnikContext _context;
21
22         public KorisnikController(KorisnikContext context)
23         {
24             _context = context;
25         }
26
27         // GET: api/Korisnik
28         [HttpGet]
29         [ProducesResponseType(typeof(IEnumerable<Korisnik>), StatusCodes.Status200OK)]
30         public IEnumerable<Korisnik> GetKorisnici()
31         {
32             var svakiKorisnik = _context.Korisnici.ToList();
33             return svakiKorisnik;
34         }
35
36         // GET: api/Korisnik/{id}
37         [HttpGet("{id}")]
38         [ProducesResponseType(typeof(Korisnik), StatusCodes.Status200OK)]
39         [ProducesResponseType(StatusCodes.Status404NotFound)]
40         public async Task<Korisnik> GetKorisnik([FromRoute] int id)
41         {
42             if (!ModelState.IsValid)
43             {
44                 return BadRequest(ModelState);
45             }
46
47             var korisnik = await _context.Korisnici.FindAsync(id);
48
49             if (korisnik == null)
50             {
51                 return NotFound();
52             }
53
54             return Ok(korisnik);
55         }
56
57         // POST: api/Korisnik/{id}
58         [HttpPost("{id}")]
59         [ProducesResponseType(typeof(Korisnik), StatusCodes.Status201Created)]
60         public async Task<Korisnik> PutKorisnik([FromRoute] int id, [FromBody] Korisnik korisnik)
61         {
62             if (!ModelState.IsValid)
63             {
64                 return BadRequest(ModelState);
65             }
66
67             if (id != korisnik.ID)
68             {
69                 return BadRequest();
70             }
71
72             try
73             {
74                 await _context.Entry(korisnik).State = EntityState.Modified;
75             }
76             catch (DbUpdateConcurrencyException)
77             {
78                 if (!KorisnikExists(id))
79                 {
80                     return NotFound();
81                 }
82                 else
83                 {
84                     throw;
85                 }
86             }
87
88             return NoContent();
89         }
90     }
91 }

```

```

1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Net;
5 using System.Threading.Tasks;
6 using System.Web;
7 using System.Web.Mvc;
8 using Microsoft.AspNetCore.Http;
9 using Microsoft.AspNetCore.Mvc;
10 using Microsoft.EntityFrameworkCore;
11 using S2RBot.Data;
12 using S2RBot.Models;
13
14 namespace S2RBot.Controllers
15 {
16     [Route("api/[controller]")]
17     [ApiController]
18     public class KorisnikController : Controller
19     {
20         // GET: api/Korisnik
21         [HttpGet]
22         [ProducesResponseType(typeof(IEnumerable<Korisnik>), StatusCodes.Status200OK)]
23         public IEnumerable<Korisnik> GetKorisnici()
24         {
25             var svakiKorisnik = _context.Korisnici.ToList();
26             return svakiKorisnik;
27         }
28
29         // GET: api/Korisnik/{id}
30         [HttpGet("{id}")]
31         [ProducesResponseType(typeof(Korisnik), StatusCodes.Status200OK)]
32         [ProducesResponseType(StatusCodes.Status404NotFound)]
33         public async Task<Korisnik> GetKorisnik([FromRoute] int id)
34         {
35             if (!ModelState.IsValid)
36             {
37                 return BadRequest(ModelState);
38             }
39
40             var korisnik = await _context.Korisnici.FindAsync(id);
41
42             if (korisnik == null)
43             {
44                 return NotFound();
45             }
46
47             return Ok(korisnik);
48         }
49
50         // POST: api/Korisnik/{id}
51         [HttpPost("{id}")]
52         [ProducesResponseType(typeof(Korisnik), StatusCodes.Status201Created)]
53         public async Task<Korisnik> PutKorisnik([FromRoute] int id, [FromBody] Korisnik korisnik)
54         {
55             if (!ModelState.IsValid)
56             {
57                 return BadRequest(ModelState);
58             }
59
60             if (id != korisnik.ID)
61             {
62                 return BadRequest();
63             }
64
65             try
66             {
67                 await _context.Entry(korisnik).State = EntityState.Modified;
68             }
69             catch (DbUpdateConcurrencyException)
70             {
71                 if (!KorisnikExists(id))
72                 {
73                     return NotFound();
74                 }
75                 else
76                 {
77                     throw;
78                 }
79             }
80
81             return NoContent();
82         }
83     }
84 }

```

```

1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Net;
5 using System.Threading.Tasks;
6 using System.Web;
7 using System.Web.Mvc;
8 using Microsoft.AspNetCore.Http;
9 using Microsoft.AspNetCore.Mvc;
10 using Microsoft.EntityFrameworkCore;
11 using S2RBot.Data;
12 using S2RBot.Models;
13
14 namespace S2RBot.Controllers
15 {
16     [Route("api/[controller]")]
17     [ApiController]
18     public class KorisnikController : Controller
19     {
20         // GET: api/Korisnik
21         [HttpGet]
22         [ProducesResponseType(typeof(IEnumerable<Korisnik>), StatusCodes.Status200OK)]
23         public IEnumerable<Korisnik> GetKorisnici()
24         {
25             var svakiKorisnik = _context.Korisnici.ToList();
26             return svakiKorisnik;
27         }
28
29         // GET: api/Korisnik/{id}
30         [HttpGet("{id}")]
31         [ProducesResponseType(typeof(Korisnik), StatusCodes.Status200OK)]
32         [ProducesResponseType(StatusCodes.Status404NotFound)]
33         public async Task<Korisnik> GetKorisnik([FromRoute] int id)
34         {
35             if (!ModelState.IsValid)
36             {
37                 return BadRequest(ModelState);
38             }
39
40             var korisnik = await _context.Korisnici.FindAsync(id);
41
42             if (korisnik == null)
43             {
44                 return NotFound();
45             }
46
47             return Ok(korisnik);
48         }
49
50         // POST: api/Korisnik/{id}
51         [HttpPost("{id}")]
52         [ProducesResponseType(typeof(Korisnik), StatusCodes.Status201Created)]
53         public async Task<Korisnik> PutKorisnik([FromRoute] int id, [FromBody] Korisnik korisnik)
54         {
55             if (!ModelState.IsValid)
56             {
57                 return BadRequest(ModelState);
58             }
59
60             if (id != korisnik.ID)
61             {
62                 return BadRequest();
63             }
64
65             try
66             {
67                 await _context.Entry(korisnik).State = EntityState.Modified;
68             }
69             catch (DbUpdateConcurrencyException)
70             {
71                 if (!KorisnikExists(id))
72                 {
73                     return NotFound();
74                 }
75                 else
76                 {
77                     throw;
78                 }
79             }
80
81             return NoContent();
82         }
83     }
84 }

```

The screenshot shows the Microsoft Visual Studio IDE interface. The main window displays the `KorisnikController.cs` file under the `SZBBot` project. The code implements a POST method to add a new `Korisnik` to the database. It includes validation logic for the `ModelState`, handling exceptions, and sending an email confirmation to the user's provided email address. The `mailModel` object contains the recipient's name, subject, and body, which is a template message in Serbo-Croatian. The `msg` variable is a `MailMessage` object initialized with the recipient's email and subject. The `msg.From` property is set to the application's email address, and the `msg.Body` property is set to the template string. The `msg.Previaue` property is also set to the template string. The `msg.To.Add()` method is used to add the user's email as the recipient. The `msg` object is then passed to the `SmtpClient` for sending.

```
return NotContent();  
}  
// POST: api/Korisnik  
[HttpPost]  
[Route("api/[controller]/[action]")]  
[ProducesResponseType(typeof(Korisnik), StatusCodes.Status201Created)]  
public async Task<ActionResult> PostKorisnik([FromBody]Korisnik korisnik)  
{  
    if (!ModelState.IsValid)  
    {  
        return BadRequest(ModelState);  
    }  
  
    _context.Korisnici.Add(korisnik);  
    await _context.SaveChangesAsync();  
  
    var mailModel = new MailModel();  
    MailMessage msg = new MailMessage();  
    mailModel.To = korisnik.Email;  
    mailModel.Subject = "SZB - Stranka Zdravog Recana";  
    string text = "Vitovani,Vam Vrhn  
    Vaš novi korisnik je uspešno prijavio poslovni račun za učlanjenje u stranju, zadnja kocja Vas neće lagneti više nad drugim, - - -  
    - jedino koja Vas neće centi više nego dragi i šteto tako jedino koja neće biti tu za Vas kada Vas nije potrebno - Stranaku Zdravog Recana ruku očekujemo da Vl, " + korisnik.Ime + " " + korisnik.Prezime + ", pristupite SZB-u" +  
    " i podilište nas a borbi za bolji život nos, partijskih kolega, no sledećim parlamentarnim izborima u junu 2020. godine Vrhn  
    - - -  
    Srećko Sojic";  
    mailModel.Body = text;  
  
    string emailAddress = mailModel.Previaue;  
    msg.To.Add(new MailAddress(emailAddress));  
  
    msg.From = new MailAddress("strankazdravogrecana2020@gmail.com", "Srećko Sojic");  
    msg.Previaue = msg.Body;  
    msg.Subject = mailModel.Subject;  
    msg.Body = mailModel.Body;  
    SmtpClient smpt = new SmtpClient("localhost");  
    smpt.Send(msg);  
}  
99 % No issues found
```

Output:  
Show output from: Debug  
Error List: Output

Solution Explorer Team Explorer Properties

The screenshot shows the Microsoft Visual Studio IDE interface. The main window displays the `KorisnikController.cs` file under the `SZRBot` project. The code implements a `DeleteKorisnik` action method that takes an `int id` parameter. It first checks if the `ModelState` is valid. If not, it returns a `BadRequest`. Then, it finds a `Korisnici` object by its ID. If the object is null, it returns a `NotFound`. Otherwise, it removes the object from the context and saves the changes asynchronously. The `FindSync` method is used to find the `Korisnici` object by its ID.

```
143     }
144 
145     // DELETE: api/Korisnik/5
146     [HttpDelete("{id}")]
147     public async Task<ActionResult> DeleteKorisnik([FromRoute] int id)
148     {
149         if (!ModelState.IsValid)
150         {
151             return BadRequest(ModelState);
152         }
153 
154         var korisnik = await _context.Korisnici.FindAsync(id);
155         if (korisnik == null)
156         {
157             return NotFound();
158         }
159 
160         _context.Korisnici.Remove(korisnik);
161         await _context.SaveChangesAsync();
162 
163         return Ok(korisnik);
164     }
165 
166     [Inference]
167     private bool KorisnikExists(int id)
168     {
169         return _context.Korisnici.Any(e => e.ID == id);
170     }
171 }
```

The Solution Explorer on the right shows the `SZRBot` project structure, which includes `Connected Services`, `Properties`, `ClientApp`, and `Controllers` (containing `KorisnikController.cs`). The `Startup.cs` file is also listed. The `Properties` folder contains `launchSettings.json` and `appsettings.json`.

21. Da bi nam slanje mejla bilo u potpunosti implementirano, potrebno je izvršiti male prepravke već kreiranih fajlova: appsettings.json i Startup.cs. Izmene su highlight-ovane na sledećim slikama:

```

23 // This method gets called by the runtime. Use this method to add services to the container.
24 public void ConfigureServices(IServiceCollection services)
25 {
26     // In production, the React files will be served from this directory
27     services.AddStaticFiles(configuration =>
28     {
29         configuration.RootPath = "ClientApp/build";
30     });
31 
32     services.AddDbContext<KorisnikContext>(options => options.UseSqlServer(Configuration.GetConnectionString("DefaultConnection")));
33 
34     var emailConfig = Configuration.GetSection("EmailConfiguration");
35     services.AddSingleton(emailConfig);
36 }
37 
38 // This method gets called by the runtime. Use this method to configure the HTTP request pipeline.
39 public void Configure(IApplicationBuilder app, IHostingEnvironment env)
40 {
41     if (env.IsDevelopment())
42     {
43         app.UseDeveloperExceptionPage();
44     }
45     else
46     {
47         app.UseExceptionHandler("/Error");
48         app.UseHsts();
49     }
50 
51     app.UseHttpsRedirection();
52 }
53 
54 
```

```

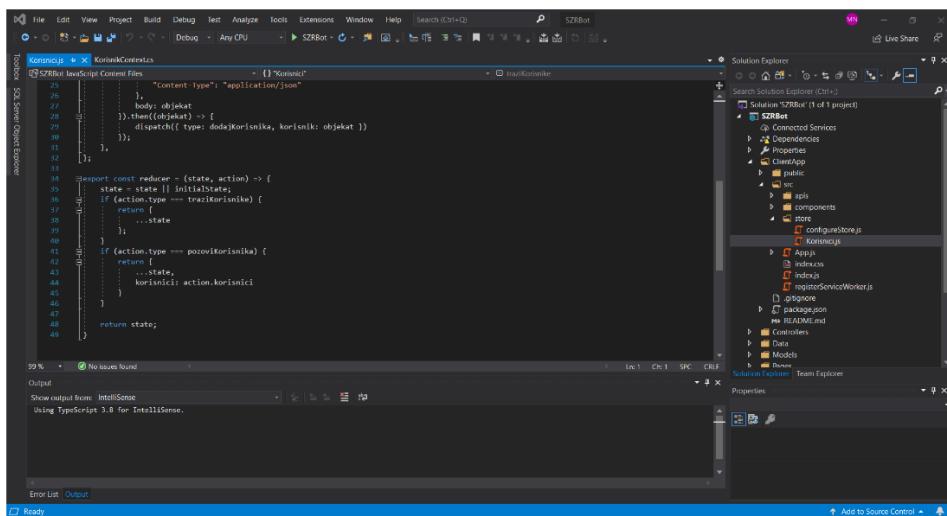
1 <?xml version="1.0" encoding="utf-8"?
2 <appSettings>
3   <add key="ConnectionString" value="Server=(localdb)\mssqllocaldb;Database=SZBot;Trusted_Connection=True;MultipleActiveResultSets=true"/>
4   <add key="logging" value="Warning"/>
5   <add key="EmailConfiguration" value="{
6     <smtpServer>: "smtp.gmail.com",
7     <port>: 587,
8     <username>: "strankardevprogramazn@gmail.com",
9     <password>: "zeekoskoje123"
10    }"/>
11   <add key="AllowedOrigins" value="*"/>
12 </appSettings>
13 
```

Na redu su koraci (22 i 23) koje treba odraditi kako bi se implementirao Redux store u folderu store.

22. Definisane su akcije koje šalju zahteve serveru (pozivaju RESTful funkcije unutar kontrolera):

```

1 const traziKorisnika = "TRAZI_KORISNIKA";
2 const pozoviKorisnika = "POZOVE_KORISNIKA";
3 const dodajKorisnika = "DODAJ_KORISNIKA";
4 const inicijalizate = [korisniks: []];
5 
6 const korisniks = [];
7 
8 export const actionCreators = {
9   traziKorisnika: () => async (dispatch, getState) => {
10     dispatch({ type: traziKorisnika });
11     const fetchUrl = `api/korisnik`;
12     const fetchAbela = await fetch(fetchUrl);
13     const korisniks = await fetchAbela.json();
14     dispatch({ type: pozoviKorisnika, korisniks });
15   },
16   dodajKorisnika: (korisnik) => async (dispatch, getState) => {
17     const url = `api/korisnik`;
18     const objekat = JSON.stringify(
19       { Ime: korisnik.Ime, prezime: korisnik.prezime, email: korisnik.email }
20     );
21     const fetchAbela = await fetch(url, {
22       method: "POST",
23       headers: {
24         Accept: "application/json",
25         "Content-Type": "application/json"
26       },
27       body: objekat
28     });
29     dispatch({ type: pozoviKorisnika, korisniks });
30   }
31 };
32 
```



```

SZRBot JavaScript Content Files
Korisnik.js
...
export const reducer = (state, action) => {
  state = state || initialState;
  if (action.type === traziKorisnike) {
    return [
      ...state,
      dispatch({ type: dodajKorisnika, korisnik: objekat })
    ];
  }
  if (action.type === pozoviKorisnika) {
    return [
      ...state,
      korisnici: action.korisnici
    ];
  }
  return state;
};

export const configureStore = (history, initialState) => {
  const reducers = {
    ...reducers,
    korisnici: korisnici.reducer,
  };

  const middleware = [
    thunk,
    routerMiddleware(history)
  ];

  // In development, use the browser's Redux dev tools extension if installed
  const enhancers = [];
  const isDevelopment = process.env.NODE_ENV === 'development';
  if (isDevelopment && window['Redux']) {
    enhancers.push(window['Redux'].devToolsExtension());
  }

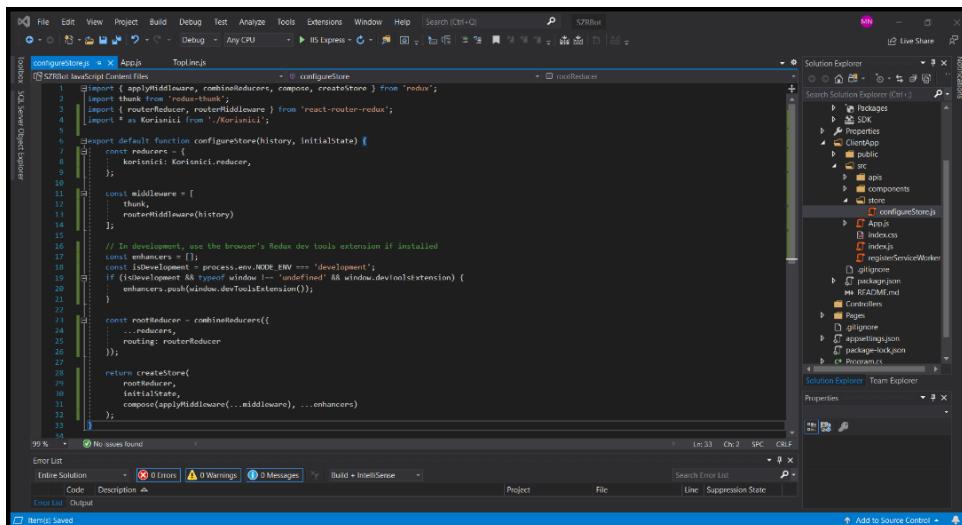
  const rootReducer = combineReducers({
    ...reducers,
    routing: routerReducer
  });

  return createStore(
    rootReducer,
    initialState,
    compose(applyMiddleware(...middleware), ...enhancers)
  );
};

59 %
No issues found
Output
Show output from: IntelliSense
Using TypeScript 3.8 for IntelliSense.
Error List Output
Ready Add to Source Control

```

23. U fajlu configureStore.js u folderu store, potrebno je prerađiti kod tako da izgleda kao sa slike:



```

SZRBot JavaScript Content Files
ConfigureStore.js
...
import { applyMiddleware, combineReducers, compose, createStore } from 'redux';
import thunk from 'redux-thunk';
import routerMiddleware from 'react-router-redux';
import * as Korisnici from './Korisnici';

export default function configureStore(history, initialState) {
  const reducers = {
    ...reducers,
    korisnici: Korisnici.reducer,
  };

  const middleware = [
    thunk,
    routerMiddleware(history)
  ];

  // In development, use the browser's Redux dev tools extension if installed
  const enhancers = [];
  const isDevelopment = process.env.NODE_ENV === 'development';
  if (isDevelopment && window['Redux']) {
    enhancers.push(window['Redux'].devToolsExtension());
  }

  const rootReducer = combineReducers({
    ...reducers,
    routing: routerReducer
  });

  return createStore(
    rootReducer,
    initialState,
    compose(applyMiddleware(...middleware), ...enhancers)
  );
};

59 %
No issues found
Error List
0 Errors | 0 Warnings | 0 Messages | Build + IntelliSense
Code Description
Error List Output
Item: Saved Add to Source Control

```

Ovime smo završili izradu aplikacije pa ceo projekat možemo Build-ovati i pokrenuti na SZRBot serveru. Finalni rezultat u slučaju ciljanog pretraživanja jeste mejl sličan mejlju sa slike:

Srećko Šojoč <strankazdravograzuma2020@gmail.com>  
кому ја

пон 25. мај 15:31 (пре 1 дана)

искључи за: хрватски

Poštovani,

Čast mi je da Vam ovom prilikom uručim pozivnicu za učlanjenje u stranku, jedinu koja Vas neće lagati više od drugih, jedinu koja Vas neće ceniti više nego drugi i isto tako jedinu koja neće biti tu za Vas kada Vam nije potrebno - Stranku Zdravog Razuma! Raširenih ruku očekujemo da Vi, Milos Nenadovic, pristupite SZR-u i podržite nas u borbi za bolji život nas, partijskih kolega, na sledećim parlamentarnim izborima u junu 2020. godine!

Srdačan pozdrav! Vaš jedini pravi,  
Srećko Šojoč

--  
This email has been checked for viruses by Avast antivirus software.  
<https://www.avast.com/antivirus>

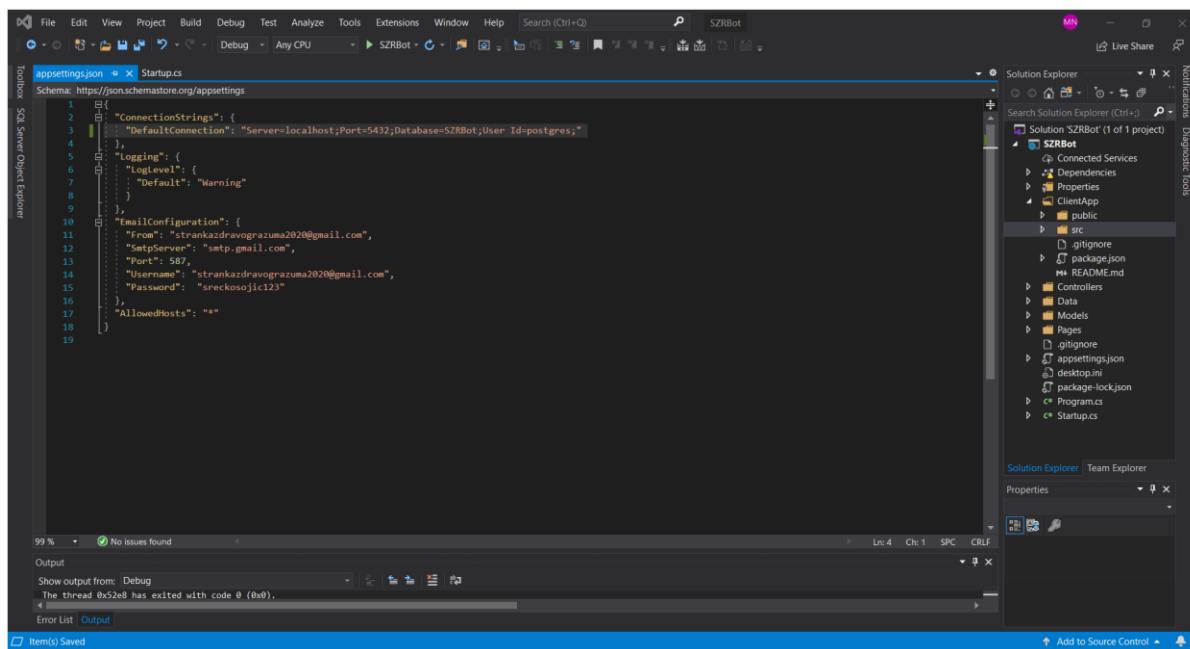
## Dodatak - izmena za open-source bazu podataka (PostgreSQL):

Kako bismo koristili open-source bazu umesto MS SQL, potrebno je izvršiti nekoliko sitnih izmena.

Pre svega potrebno je skinuti i instalirati PostgreSQL sa sajta <https://www.postgresql.org/download/windows/> (za korisnike windows operativnog sistema). Verzija koju ćemo instalirati je 10.13 (poslednja koja je dostupna za sve platforme).

Nakon kompletirane instalacije (uključujući i restartovanje računara), potrebno je pronaći fajl pg\_hba.conf koji se nalazi unutar foldera data u instaliranom direktorijumu PostgreSQL. Taj fajl je potrebno otvoriti pomoću tekstu editora (pokrenuti kao administrator). Nakon toga skrolovati do dna tog tekstualnog fajla gde se nalazi deo teksta koji podseća na tabelu sa informacijama o konekcijama kao i o metodi zaštite (md5). Taj deo teksta treba prepraviti tako da svuda gde piše md5 prepraviti u trust. Ovako se omogućuje modifikacija baze.

1. U fajlu appsetting.json potrebno je izmeniti konekcioni string ("Default Connection" property "Connection Strings" objekta) tako da izgleda kao sa slike:



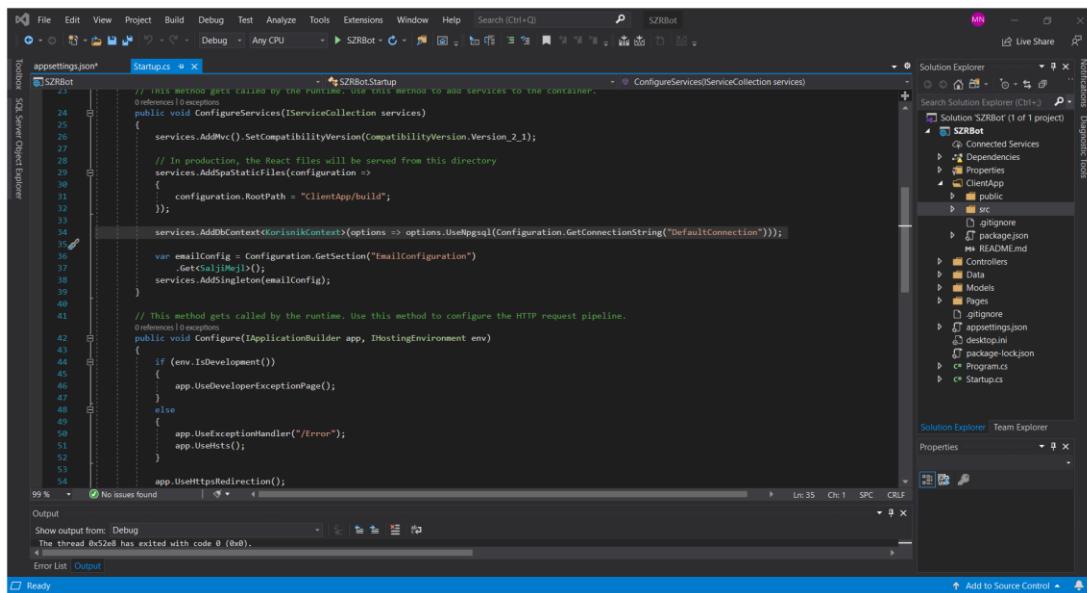
```

appsettings.json
Schema: https://json.schemastore.org/appsettings
1  {
2    "ConnectionStrings": {
3      "DefaultConnection": "Server=localhost;Port=5432;Database=SZRBot;User Id=postgres;"}
4    },
5    "Logging": {
6      "LogLevel": {
7        "Default": "Warning"
8      }
9    },
10   "EmailConfiguration": {
11     "From": "strankazdravograzu2020@gmail.com",
12     "SmtpServer": "smtp.gmail.com",
13     "Port": 587,
14     "Username": "strankazdravograzu2020@gmail.com",
15     "Password": "srebeckojic123"
16   },
17   "AllowedHosts": "*"
18 }
19

```

Linija koju treba izmeniti je higlight-ovana. Ovaj konekcioni string izuzima upotrebu Password polja a to nam je omogućeno modifikacijom fajla pg\_hba.conf gde smo za metodu autentifikacije umesto md5 stavili trust.

## 2. Naredna izmena je unutar Startup.cs fajla:



```

Startup.cs
1  // This method gets called by the runtime. Use this method to add services to the container.
2  public void ConfigureServices(IServiceCollection services)
3  {
4    services.AddMvc().SetCompatibilityVersion(CompatibilityVersion.Version_2_1);
5
6    // In production, the React files will be served from this directory
7    services.AddSpaStaticFiles(configuration =>
8    {
9      configuration.RootPath = "ClientApp/build";
10    });
11
12    services.AddDbContext<KorisnikContext>(options => options.UseNpgsql(Configuration.GetConnectionString("DefaultConnection")));
13
14    var emailConfig = Configuration.GetSection("EmailConfiguration")
15      .Get<EmailConfig>();
16    services.AddSingleton(emailConfig);
17
18
19    // This method gets called by the runtime. Use this method to configure the HTTP request pipeline.
20    public void Configure(IApplicationBuilder app, IHostingEnvironment env)
21    {
22      if (env.IsDevelopment())
23      {
24        app.UseDeveloperExceptionPage();
25      }
26      else
27      {
28        app.UseExceptionHandler("/Error");
29        app.UseHsts();
30      }
31
32      app.UseHttpsRedirection();
33
34      app.UseStaticFiles();
35
36      app.UseRouting();
37
38      app.UseAuthorization();
39
40      app.UseEndpoints(endpoints =>
41      {
42        endpoints.MapControllerRoute(
43          name: "default",
44          pattern: "{controller}/{action}/{id?}",
45          defaults: new { controller = "Home", action = "Index", id = null }
46        );
47      });
48    }
49  }
50
51
52
53
54

```

Highlight-ovanu liniju koda treba izmeniti tako da se poziva funkcija UseNpgsql() funkcija umesto UseSqlServer().