



Univerzitet u Novom Sadu
Tehnički fakultet „Mihajlo Pupin“
Zrenjanin



SEMINARSKI RAD

Predmet: Programski prevodioci

Upoređivanje gramatike C# i Java - SEMANTIKA DOMENA: Evidencija fudbalskih igrača - tip 5/A

Predmetni nastavnik:
doc. dr Ljubica Kazi

Autor rada
Aleksa Cakić SI 23/17

Zrenjanin 2020.

1. Uvod

1.1 Malo o C#

Microsoft je od 2014 godine menjajući kompajler u C# programski jezik; znači danas C# programski jezik kompajlira kompajler za C# programski jezik; omogućio programerima da kompajler kodnog imena „Roslyn“ kao kompajler nije više zatvorena crna kutija, već je open-source kod. Ali je isto tako time omogućio da se prave još bolje promene na samom programskom jeziku, pa tako danas imate male promene u kodu koje prave velike promene i umanjuju pisanje koda ali će ih biti sve više. Već sad možete izguglati neke promene koje Microsoft već najavljuje pored mnogobrojnih promena koje su ove godine uvedene.

Za razliku od C# programskog jezika, C++ programski jezik se smatra nešto bržim što danas i nije toliko relevantno s obzirom da su kompjuteri danas mnogo brži, osim možda u nekim segmentima kad su u pitanju programiranje igrica, ali ništa toliko zastrašujuće što se ne može nadomestiti. Sa C++ programskim jezikom možete da programirate i na Linux operativnom sistemu i na OS X platformi jer C++ programski jezik komunicira direktno sa operativnim sistemom i kompajlira odmah mašinski kod, dok je C# zavisao od Microsoft .Net Framework-a koji je posrednik između vaše aplikacije i operativnog sistema. Međutim vi u C++ ne možete da programirate na primer WPF – Windows Presentation Foundation aplikacije dok je danas programirati Windows Forms zastarela tehnologija i pored toga što se mora znati radi starijih projekata. Sa C# programskim jezikom možete definitivno da uradite više i da kodirate mnogo jednostavnije i lakše nego sa C++ programskim jezikom i to uvek imajte na umu.

1.2 Malo o javi

Sada već davne 1995. godine je kompanija Sun Microsystems lansirala programski jezik Java, a koji se definiše kao objektno - orijentisani. Kasnije je kompanija Oracle otkupila pomenutu, te tako i postala zvaničan vlasnik ove tehnologije.

Princip na kome je ovaj programski jezik kreiran je primarno usmeren na jezik Oberon, ali i na mnoge druge, dok mu je sintaksa nalik C i C++ programskim jezicima. Ipak, ovaj tip programskog jezika ima značajno strožiji princip prevođenja, a potpuno je nezavisan od platforme na kojoj se radi. Posebna pažnja prilikom njegovog kreiranja je usmerena na upravljanje memorijom, a koja je kod Java programskog jezika u velikoj meri jednostavnija. Smatra se da je razlog za to vezan za veliku popularnost C programskog jezika.

Svrha primene ovog jezika je na prvom mestu vezana za davanje konkretnih instrukcija određenom uređaju, a kako bi on bio u mogućnosti da izvrši određene komande.

Upravo se Java programski jezik smatra jednim od najpopularnijih, a veruje se da je osnovni razlog za to vezan za činjenicu da se njegovo korišćenje uopšte ne naplaćuje. Budući da se korišćenje mnogih drugih programskih jezika gotovo uvek naplaćuje, sasvim je jasno i zbog čega upravo Java uživa toliku popularnost.

Druga prednost primene ovog programskog jezika se odnosi na činjenicu da se komande koje navodi Java, praktično rečeno mogu izvršavati na gotovo bilo kom računaru, jer je ona kreirana tako da uopšte ne zavisi od platforme koja se koristi na konkretnom tipu uređaja.

Svakako je prednost i ta što se baš Java koristi kada je potrebno pristupiti razvoju različitih vrsta aplikacija. Uz primenu tog programskog jezika se mogu praviti takozvane GUI aplikacije (Graphical User Interface), kao i apleti aplikacije (Applets), ali isto tako i one koje su poznate kao konzolne. Kada se karakteristike ovog programskog jezika uporede sa drugima, stiče se jasan utisak da je Java izuzetno jednostavna za korišćenje, pa se i to svakako može svrstati u jednu od mnogobrojnih prednosti koje se na njenu primenu odnose.

Zanimljivo je navesti i podatak da naziv ovog programskog jezika vodi poreklo upravo od istoimenog ostrva, a koje se nalazi na teritoriji Indonezije. Iako je Java primarno programski jezik, činjenica je da se ovim pojmom označava isto tako i sotverska platforma, koja je zadužena da pokreće aplikacije koje su u ovom programskom jeziku izrađene.

Neretko se događa da se pojam Java i JavaScript mešaju, ali za to ne postoji razlog, sobzirom na to da su u pitanju dva potpuno drugačija programska jezika, čije karakteristike i funkcije se svakako razlikuju. Činjenica je da ovaj programski jezik ima brojne prednosti, ali naravno da su prisutne i određene mane, odnosno nedostaci kada je u pitanju njegova primena. Najznačajniji nedostatak se odnosi na brzinu njenog rada, uzevši u obzir da se ova tehnologija relativno sporo pokreće, a u poređenju sa drugima koje su joj na neki način slične.

1.3 Malo o MSSQLU

Bez obzira koji programski jezik učite ili sa kojim programskim jezikom programirate iz hobija ili poslovno; od svakog programera se očekuje da poznaje rad sa bazama podataka kao i osnove strukturiranja relacionih baza podataka. Baza podataka vam je najjednostavnije rečeno kolekcija podataka smeštena u elektronskom formatu. Na engleskom jeziku se baza podataka kaže database ili skraćeno db. Inače prema Wikipedia-iji; baza podataka je organizovana kolekcija podataka za brzo pretraživanje i pristup; koja zajedno sa sistemom za održavanje i administraciju, organizovanje i memorisanje tih podataka čine sistem baze podataka. U školama će vam reći da je baza podataka kolekcija podataka koja se zapisuje u SQL server. Sve su ove definicije tačne ali je vama najvažnije da shvatite da je dobro organizovana baza podataka rešenje pola vašeg programerskog posla. Sve poznate baze podataka poput SQL, MySQL ili Oracle Database baza podataka; koriste isti standard za rad sa podacima i sve podržavaju SQL programski jezik. Kod Microsoft-a je to T-SQL i on je važan aspekt bez obzira na pojavu upitnog integrisanog jezika LINQ-a koji naveliko danas menja način programiranja ali definitivno nije zamena. T-SQL je jezik baza podataka i ne možete ga preskočiti već ga morate znati. Ali zato kad jednom savladate rad sa bazama podataka, lako će te moći koristiti sve vrste

baza podataka jer funkcionišu na istim standardnim principima. Ne morate vi biti stručnjak za baze podataka ali neke osnovne stvari morate poznavati kako bi ste uopšte mogli da koristite baze podataka u vašim programima, sajtovima, servisima ili sistemima. U školama računara se uglavnom rad sa bazama podataka ne svrstava u C# programski jezik, ali se uči uporedo. Tako ću vam i ja uporedo sa postovima C# programskog jezika pisati i postove za rad sa Microsoft SQL Server 2016 Express serverom.

(Microsoft SQL Server 2016)

Pre su programeri uglavnom koristili tekstualne ili binarne datoteke da skladište informacije koje koristi njihov program. Međutim na taj način podaci su se samo gomilali, ponavljali; zauzimali su dosta memorije; mnoge kolone su ostajale prazne, svaka datoteka je u suštini predstavljala samo jednu tabelu i zato se javila velika potreba za programima i serverima koji su pored skladištenja podataka mogle da uvedu pre svega organizaciju u podacima ali i mnoštvo tehnoloških robusnih novina koje danas serveri za baze podataka nude. Prva baza podataka sa kojom ste se možda susretali kroz Microsoft Office jeste Microsoft Access baza podataka. I ona takođe obuhvata tabele, poglede, uskladištene procedure, funkcije i druge objekte koje su neophodni za pravljenje sistema podataka ali navedena baza podataka je desktop aplikacija i može biti veoma problematična kad su u pitanju mnoštvo objekata, tabela, relacija i podataka i pristupanjem istih preko mreže. Zato je pametnije odmah koristiti neku od edicija Microsoft SQL Server baza podataka. Već više od deceniju postoje razne verzije i edicije Microsoft SQL Server-a , često se koriste starije verzije i one se stalno menjaju poput:

SQL Server Express Edition

SQL Server Workgroup Edition

SQL Server Developer Edition

SQL Server Standard Edition

SQL Server Enterprise Edition

SQL Server Mobile Edition

Naravno većina edicija se razlikuje pored tehničkih alata, ograničenja i mogućnosti; tako isto i po godinama izdanja. Microsoft SQL Server 2005 Express Edition i Microsoft SQL Server 2016 Express Edition se takođe razlikuju iako se te razlike često ne vide. Microsoft SQL Server 2016 Express je odličan izbor pre svega za početnike i zato što je besplatan dok su vam za druge edicije potrebne licence i mogu vam reći da one nisu ni malo jeftine. Zato moj izbor za vas je definitivno Microsoft SQL Server 2016 Express. Dobra vest je da Microsoft SQL Server 2016

Express pored 1 GB RAM-a, jednog procesora, nudi ograničenje skladištenja podataka na 10 GB. U ranijim verzijama ograničenje je bilo 4 GB. Sada dolazi sa Advanced Services i nećete imati komplikacije sa pravljjenjem dijagrama.

Malo o Postgresu
PostgreSQL ili jednostavnije Postgres je vrsta objektno-orijentisanih relacionih sistema za upravljanje bazama podataka (SUBP), pod open source licencom (otvoreni kod). Smatra se jednom od najpouzdanijih baza podataka. Najčešće koristi za web aplikacije i web baze podataka. Reklo bi se da je najveći konkurent MySQL-u.

PostgreSQL je bazično razvijen za rad na UNIX platformama, ali je portovan i na Linux, Windsows, macOS.

Prva zvanična verzija je objavljena 29. januara 1997. godine.

Inicijalno je razvijen u programskom jeziku C, ali takođe ima podršku za integraciju sa programskim jezicima kao što su Python, Perl, .NET, C++, Java, PHP, Ruby on Rails i drugi.

Postgres je naslednik Ingres-a koji je bio jedan od sistema baza podataka, razvijen između 1977. i 1985. godine. Zvanično, autor PostgreSQL-a je Michael Stonebraker, profesor na Kalifornijskom univerzitetu u Berkliju (UCB). Stonebrakerova ideja je bila da izgradi napredniju verziju Ingres-a koja je robusnija, uz bolje performanse. Stonebraker i njegove kolege u UCB-u su osam godina (1986-1994) razvijali sistem Postgres baze podataka. Stonebrakerove kolege Andrej Lu i Jolli Chen su dodatno poboljšali razvijeni sistem zamjenom POSTQUEL upitnog jezika (query language) sa popularnijim i najčešće korišćenim SQL.

Ta poboljšana verzija nazvana je Postgres95. Nakon toga, 1996. godine, Postgres95 je po prvi put ušao u softversku industriju i postao jedan od najrobusnijih i najčešće korišćenih servera otvorenog programskog koda (open source).

Osobine i prednosti PostgreSQL-a

Open Source SUBP (DBMS – Database Management System) – Prva glavna prednost korišćenja Postgres-a je to što je open source i može se prilagoditi prema zahtevima developera. Ova mogućnost prilagođavanja je izuzetno korisna u razvoju velikih aplikacija.

Velika razvojna zajednica (iliti velik community) – Postgres je na tržištu već više od 15 godina i njegova zajednica je u ovom trenutku izuzetno velika, što samim tim znači dobru podršku i pomoć pri rešavanju problema vezanih za rad i uporebu.

Isplativost – Postgres je izuzetno ekonomičan i ne zahteva mnogo obučavanja korisnika kako bi se naučilo kako koristiti i programirati za ovu bazu podataka. Takođe, zahtevi za održavanje i podešavanje Postgres baze podataka su relativno mali u odnosu na druge sisteme za upravljanje bazama podataka.

Portabilnost – Dobra stvar u Postgresu je to što je portabilan i prenosiv sa gotovo svim glavnim platformama i programskim jezicima. Ova baza podataka je idealna za aplikacije namenjene višestrukim platformama.

Alatke za razvoj i GUI – Server Postgres baze podataka ne zahteva obimne konfiguracije komandne linije. Razvijen je nekoliko alata i GUI interfejsa koji vam mogu pomoći u jednostavnoj instalaciji i upravljanju serverom baze podataka.

Pouzdanost i stabilnost – Postgres je svetski priznat kao najsigurnija i stabilnija baza podataka. Šanse da se uništi baza podataka su minimalne i čak i ako se baza podataka sruši, postoje načini i funkcije koje vam omogućavaju da obnovite i vratite podatke.

Nedostaci PostgreSQL-a

Performanse – za jednostavne operacije čitanja i brisanja, PostgreSQL postiže lošije rezultate u odnosu na, na primer, MySQL.

Popularnost – definitivno da nije toliko zastupljen, u odnosu na druge sisteme baza podataka i samim tim nema tako jaku podršku.

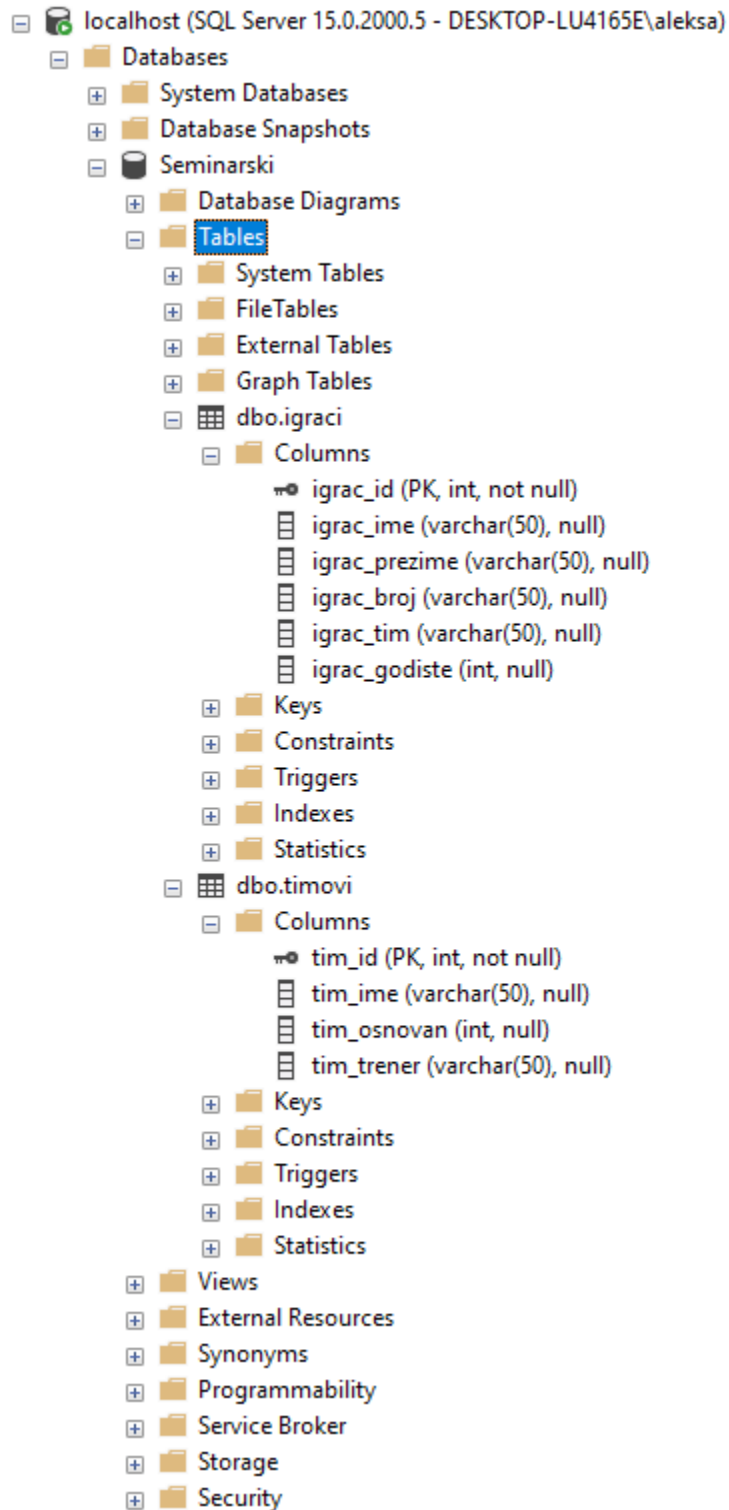
Hosting – nije toliko zastupljen kod provajdera i pružaoca hosting usluga.

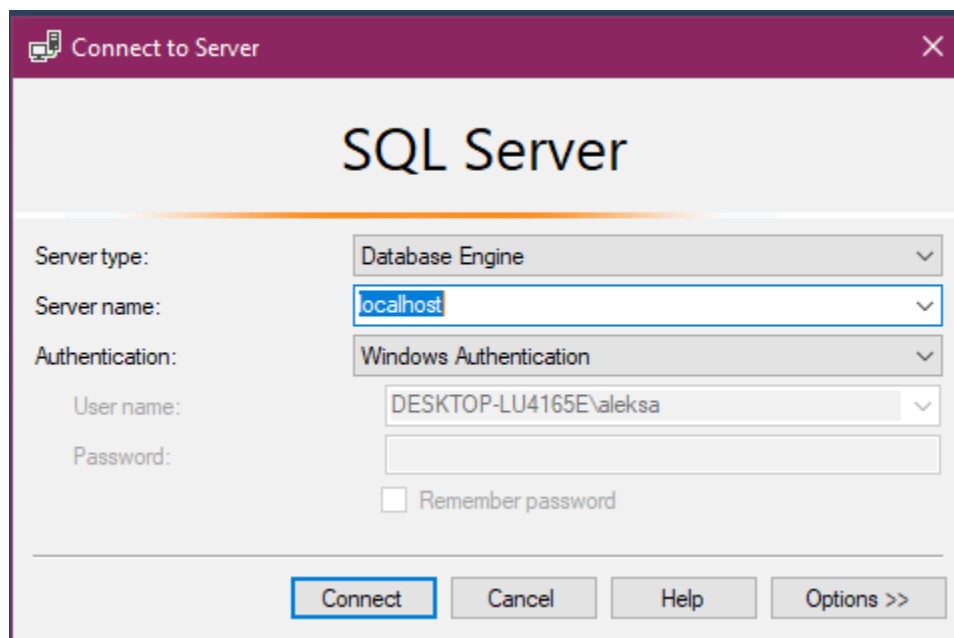
Instalacija i pripremanje projekta

Prvi primer projekta je u C#.

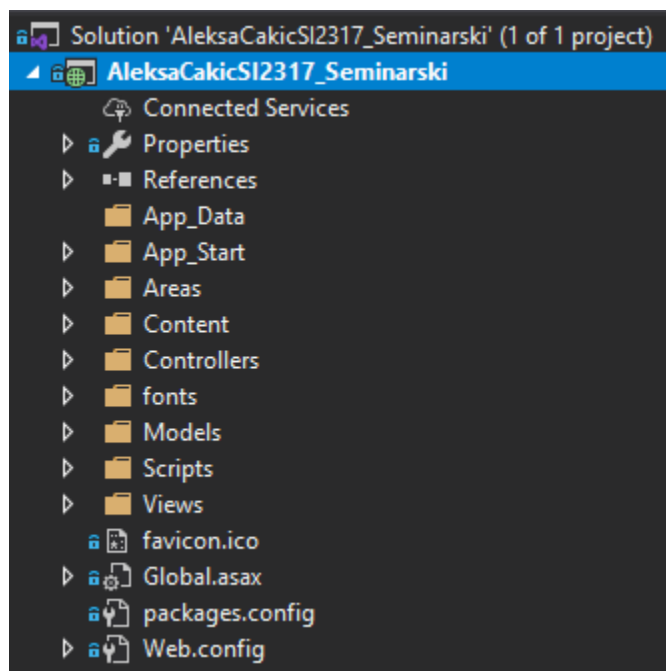
Izrada projekta je vrlo laka i brza, i ne treba biti uopšte upoznat sa tehnologijom kako bi se uradio kompletan projekat, čime se implicira lakoća. Ovo je možda i najbolja tehnologija za apsolutno početnike da se upoznaju sa Web servisima, ali definitivno ne za dalji razvoj.

Prateći slike, projekat se dalje razvijao:

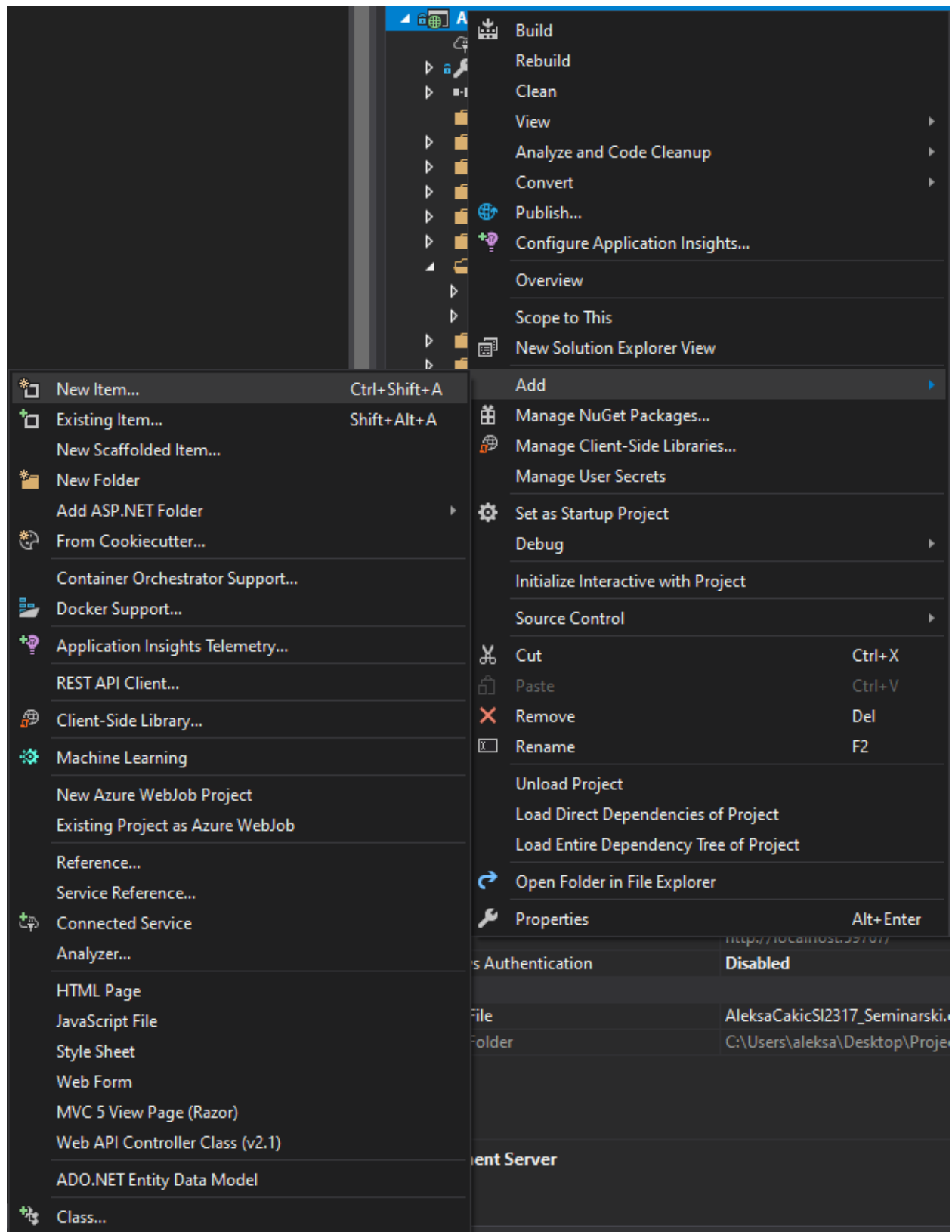




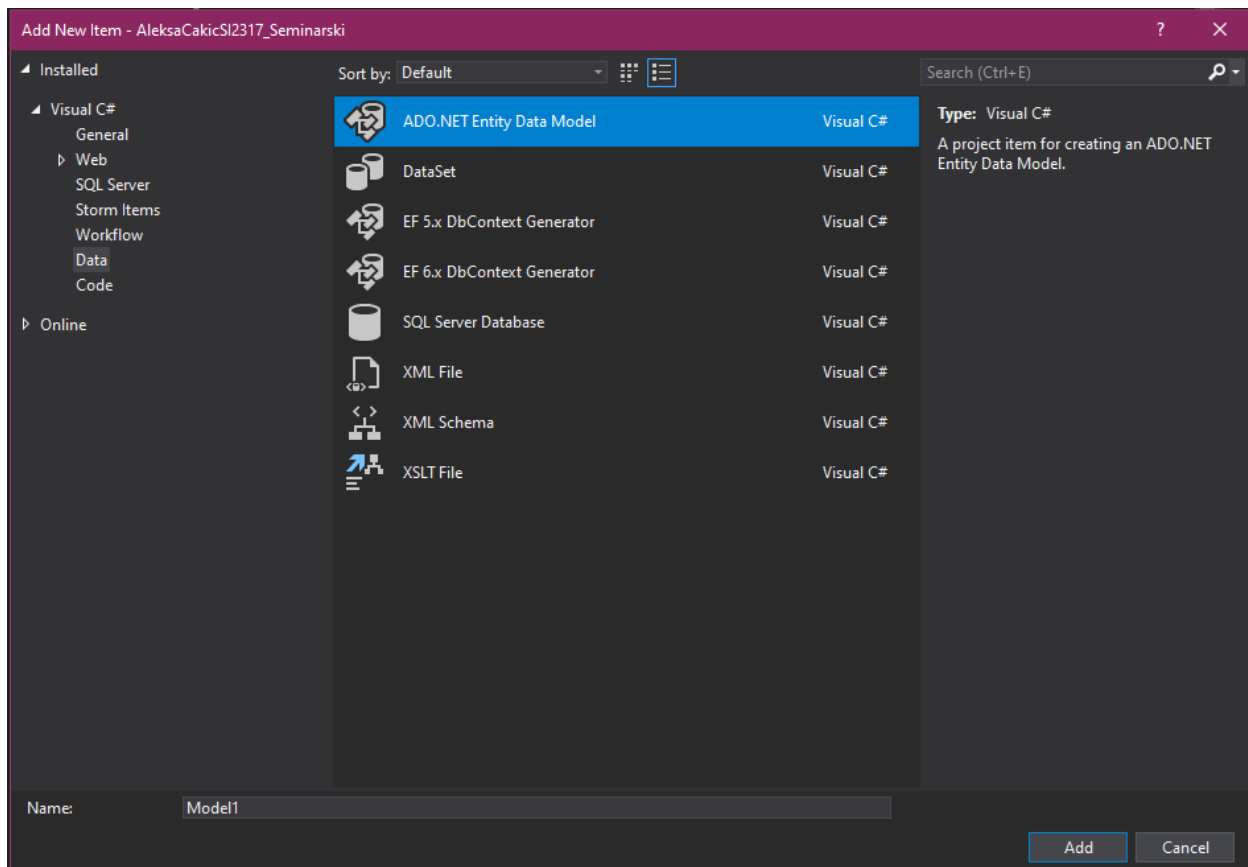
Povezivanje na server je prvo.

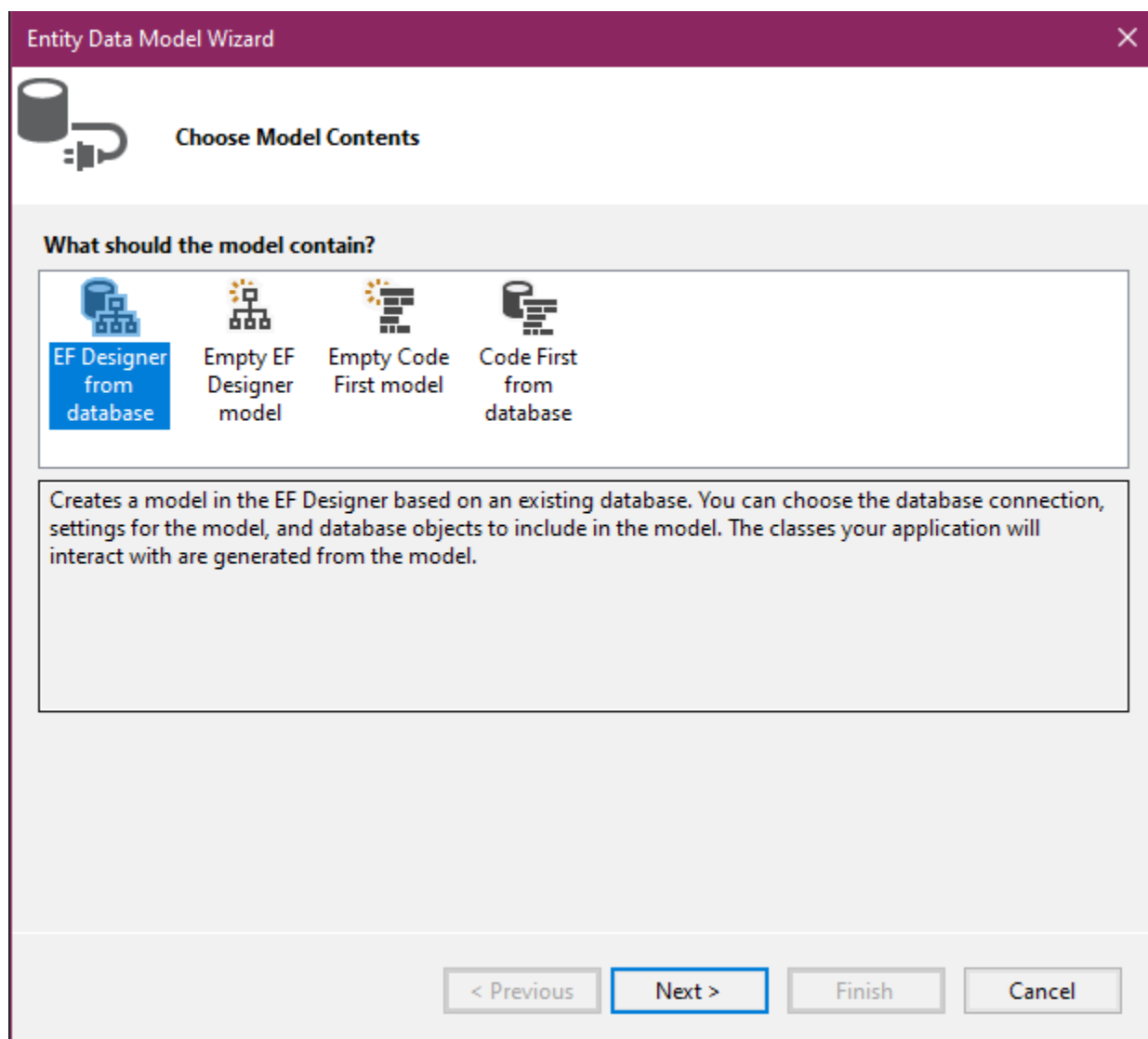


Struktura C# projekta kada se napravi.



Kada u MSSQL serveru napravite tabele, tako što će se desnim klikom na šemu I odabirom da želite da napravite tabelu, u VS-u će te dodati novi item, koji će biti tipa data entity kao na slici ispod.





Posle toga će vas dočekati sledeći prozori, gde I da ne znate engleski, možete se snaći vrlo lako.



Choose Your Data Connection

Which data connection should your application use to connect to the database?

[New Connection...](#)

This connection string appears to contain sensitive data (for example, a password) that is required to connect to the database. Storing sensitive data in the connection string can be a security risk. Do you want to include this sensitive data in the connection string?

- ☐ No, exclude sensitive data from the connection string. I will set it in my application code.
- ☐ Yes, include the sensitive data in the connection string.

Connection string:

☒ Save connection settings in Web.Config as:

[< Previous](#)[Next >](#)[Finish](#)[Cancel](#)

Connection Properties

Enter information to connect to the selected data source or click "Change" to choose a different data source and/or provider.

Data source:
[.NET Framework Data Provider for SQL Server] [Change...]

Server name:
[localhost] [Refresh]

Log on to the server

Authentication: [Windows Authentication]

User name: []

Password: []

☐ Save my password

Connect to a database

☒ Select or enter a database name:

[]

☐ A


master
model
msdb
Seminarski
tempdb

[Advanced...]

[Test Connection] [OK] [Cancel]

Posle nekoliko "Next" opcija, dočekaće Vas odabir baze podataka, sada je baza seminarski.

Entity Data Model Wizard

 Choose Your Database Objects and Settings

Which database objects do you want to include in your model?

Tables

dbo

igraci

sysdiagrams

timovi

Views

Stored Procedures and Functions

☒ Pluralize or singularize generated object names

☒ Include foreign key columns in the model

☒ Import selected stored procedures and functions into the entity model

Model Namespace:

SeminarskiModel2

< Previous

Next >


Finish

Cancel

Odaberite jednu ili obe tabele, rezultat će biti isti.

15

Entity Data Model Wizard

 Choose Your Data Connection

Which data connection should your application use to connect to the database?

desktop-lu4165e.Seminarski.dbo New Connection...

This connection string appears to contain sensitive data (for example, a password) that is required to connect to the database. Storing sensitive data in the connection string can be a security risk. Do you want to include this sensitive data in the connection string?

☐ No, exclude sensitive data from the connection string. I will set it in my application code.

☐ Yes, include the sensitive data in the connection string.

Connection string:

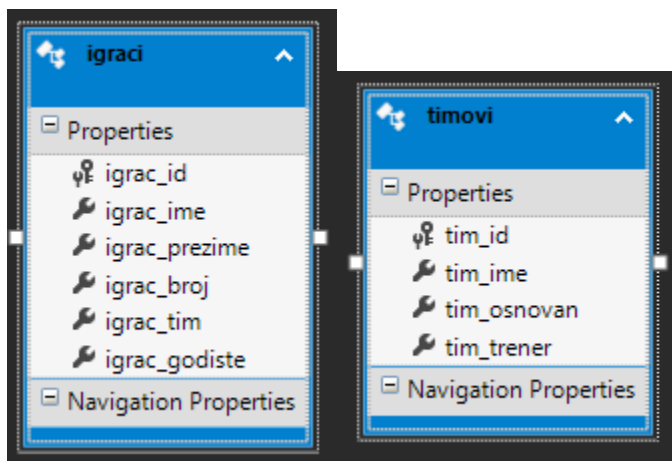
```
metadata=res://*/Model1.csdl|res://*/Model1.ssdl|
res://*/Model1.msl;provider=System.Data.SqlClient;provider connection string="data
source=localhost;initial catalog=Seminarski;integrated
security=True;MultipleActiveResultSets=True;App=EntityFramework"
```

☒ Save connection settings in Web.Config as:

SeminarskiEntities2

< Previous **Next >** Finish Cancel

I samo pritiskati “Next” I čekati da system uradi svoj deo posla – kreiranje celog backend projekta.



Na kraju će te imati modele kaon a slici iznad.

```
1  //-----
2  // <auto-generated>
3  //   This code was generated from a template.
4  //
5  //   Manual changes to this file may cause unexpected behavior in your application.
6  //   Manual changes to this file will be overwritten if the code is regenerated.
7  // </auto-generated>
8  //-----
9
10 namespace AleksaCakicSI2317_Seminarski.Models
11 {
12     using System;
13     using System.Collections.Generic;
14
15     public partial class igraci
16     {
17         public int igrac_id { get; set; }
18         public string igrac_ime { get; set; }
19         public string igrac_prezime { get; set; }
20         public string igrac_broj { get; set; }
21         public string igrac_tim { get; set; }
22         public Nullable<int> igrac_godiste { get; set; }
23     }
24 }
25
```

U folderu Models, moći ćete da vidite C# rendiciju tog istog modela koji je automatski generisan.

```

1  //-----
2  // <auto-generated>
3  //   This code was generated from a template.
4  //
5  //   Manual changes to this file may cause unexpected behavior in your application.
6  //   Manual changes to this file will be overwritten if the code is regenerated.
7  // </auto-generated>
8  //-----
9
10 namespace AleksaCakicSI2317_Seminarski.Models
11 {
12     using System;
13     using System.Collections.Generic;
14
15     9 references
16     public partial class timovi
17     {
18         3 references
19         public int tim_id { get; set; }
20         0 references
21         public string tim_ime { get; set; }
22         0 references
23         public Nullable<int> tim_osnovan { get; set; }
24         0 references
25         public string tim_trener { get; set; }
26     }
27 }

```

```

1  using System.Web.Http;
2  using System.Web.Http.Cors;
3
4  namespace AleksaCakicSI2317_Seminarski
5  {
6      1 reference
7      public static class WebApiConfig
8      {
9          1 reference
10         public static void Register(HttpConfiguration config)
11         {
12             // Web API configuration and services https://localhost:44309/
13             config.EnableCors(new EnableCorsAttribute(headers: "*", methods: "*", origins: "*"));
14
15             // Web API routes
16             config.MapHttpAttributeRoutes();
17
18             config.Routes.MapHttpRoute(
19                 name: "DefaultApi",
20                 routeTemplate: "api/{controller}/{id}",
21                 defaults: new { id = RouteParameter.Optional }
22             );
23         }
24     }
25 }

```

U webapiconfig hoćemo da podesimo CORS headere I port na kome će raditi aplikacija.

```

// PUT: api/igracis/5
[ResponseType(typeof(void))]
References
public IHttpActionResult Putigraci(int id, igraci igraci)
{
    //if (!ModelState.IsValid)
    //{
    //    return BadRequest(ModelState);
    //}

    if (id != igraci.igrac_id)
    {
        return BadRequest();
    }

    db.Entry(igraci).State = EntityState.Modified;

    try
    {
        db.SaveChanges();
    }
    catch (DbUpdateConcurrencyException)
    {
        if (!igraciExists(id))
        {
            return NotFound();
        }
        else
        {
            throw;
        }
    }

    return StatusCode(HttpStatusCode.NoContent);
}

```

Nakon toga, možemo poglati koji se endpointovi nalaze u Kontroleru, kako bismo mogli da im pristupimo preko Frontenda.

```
// GET: api/igracis
// References
public IQueryable<igraci> Getigracis()
{
    return db.igracis;
}
```

```
// POST: api/igracis
[ResponseType(typeof(igraci))]
// References
public IHttpActionResult Postigraci(igraci igraci)
{
    //if (!ModelState.IsValid)
    //{
    //    return BadRequest(ModelState);
    //}

    db.igracis.Add(igraci);
    db.SaveChanges();

    return CreatedAtRoute("DefaultApi", new { id = igraci.igrac_id }, igraci);
}

// DELETE: api/igracis/5
[ResponseType(typeof(igraci))]
// References
public IHttpActionResult Deleteigraci(int id)
{
    igraci igraci = db.igracis.Find(id);
    if (igraci == null)
    {
        return NotFound();
    }

    db.igracis.Remove(igraci);
    db.SaveChanges();

    return Ok(igraci);
}
```

```
// GET: api/timovis
[ResponseType(typeof(timovi))]
public IQueryable<timovi> Gettimovis()
{
    return db.timovis;
}
```

```
// POST: api/timovis
[ResponseType(typeof(timovi))]
public IHttpActionResult Posttimovi(timovi timovi)
{
    //if (!ModelState.IsValid)
    //{
    //    return BadRequest(ModelState);
    //}

    db.timovis.Add(timovi);
    db.SaveChanges();

    return CreatedAtRoute("DefaultApi", new { id = timovi.tim_id }, timovi);
}

// DELETE: api/timovis/5
[ResponseType(typeof(timovi))]
public IHttpActionResult Deletetimovi(int id)
{
    timovi timovi = db.timovis.Find(id);
    if (timovi == null)
    {
        return NotFound();
    }

    db.timovis.Remove(timovi);
    db.SaveChanges();

    return Ok(timovi);
}
```

```

// PUT: api/timovis/5
[ResponseType(typeof(void))]
//References
public IHttpActionResult Puttimovi(int id, timovi timovi)
{
    //if (!ModelState.IsValid)
    //{
    //    return BadRequest(ModelState);
    //}

    if (id != timovi.tim_id)
    {
        return BadRequest();
    }

    db.Entry(timovi).State = EntityState.Modified;

    try
    {
        db.SaveChanges();
    }
    catch (DbUpdateConcurrencyException)
    {
        if (!timoviExists(id))
        {
            return NotFound();
        }
        else
        {
            throw;
        }
    }

    return StatusCode(HttpStatusCode.NoContent);
}

```

2. JAVA SETUP

Za razliku od prostog načina autokonfiguracije, Java u stvari zahteva znanje i umeće. Uzimajući u obzir da je Java možda i najbolji način da se nauči nešto, autor ovog seminarskog rada je na najprostiji i bezbolniji način hteo da prikaže koliko je potrebno da se zna kako bi mogla da se uradi full stack aplikacija.

Iako je autor mogao prostim principom, linijom manjeg otpora, kao u C# projektu, da završi sve, ipak se odlučio da pokaže lepotu Java jezika, Docker virtualizacije i build toolova.

Prvo ćete pokrenuti docker, sačekati da se podigne i krene sa radom. I želite da budete sigurni da docker radi sa Linux kontejnerima, inače neće raditi kako treba, ili uopšte neće raditi, baš kao Windows.

Kada uđete u java projekat, sačekajte da se indeksiraju fajlovi ako koristite IntelliJ.

Zatim u build.gradle želite pokrenete instalaciju i skidanje svih dependencija.

Kada se sve to završi, pokrenuti App main metodu za podizanje Spring aplikacije.

Onda ući u front end folder, gde su vi Angular fajlovi i u terminal pokrenuti sledeće:

- npm install
- npm serve -o

Pre poslednjeg koraka, ustanoviti da imate nodejs i Angular installirano.



```
Microsoft Windows [Version 10.0.18363.900]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\aleksa\Desktop\Project\tfzrseminarski>docker-compose up
```

Docker se pokreće, I kada se virtuelne mašine podignu, preko docker compose fajla podižemo Postgres server definisan u fajlu bez potrebe da skidamo I instaliramo isti.

```

Microsoft Windows [Version 10.0.18363.900]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\aleksa\Desktop\Project\tfzrseminarski>docker-compose up
tfzrseminarski_db_1 is up-to-date
Attaching to tfzrseminarski_db_1
db_1 | The files belonging to this database system will be owned by user "postgres".
db_1 | This user must also own the server process.
db_1 |
db_1 | The database cluster will be initialized with locale "en_US.utf8".
db_1 | The default database encoding has accordingly been set to "UTF8".
db_1 | The default text search configuration will be set to "english".
db_1 |
db_1 | Data page checksums are disabled.
db_1 |
db_1 | fixing permissions on existing directory /var/lib/postgresql/data ... ok
db_1 | creating subdirectories ... ok
db_1 | selecting dynamic shared memory implementation ... posix
db_1 | selecting default max_connections ... 100
db_1 | selecting default shared_buffers ... 128MB
db_1 | selecting default time zone ... Etc/UTC
db_1 | creating configuration files ... ok
db_1 | running bootstrap script ... ok
db_1 | performing post-bootstrap initialization ... ok
db_1 | initdb: warning: enabling "trust" authentication for local connections
db_1 | You can change this by editing pg_hba.conf or using the option -A, or
db_1 | --auth-local and --auth-host, the next time you run initdb.
db_1 | syncing data to disk ... ok
db_1 |
db_1 |
db_1 | Success. You can now start the database server using:
db_1 |
db_1 |         pg_ctl -D /var/lib/postgresql/data -l logfile start
db_1 |
db_1 |
db_1 | waiting for server to start....2020-06-26 17:14:38.238 UTC [48] LOG:  starting PostgreSQL 12.3 (Debian 12.3-1.pgdg100+1) on x86_64-pc-linux-gnu, compiled by gcc (Debian 8.3.0-6) 8.3.0, 64-bit
db_1 | 2020-06-26 17:14:38.234 UTC [48] LOG:  listening on Unix socket "/var/run/postgresql/.s.PGSQL.5432"
db_1 | 2020-06-26 17:14:38.268 UTC [49] LOG:  database system was shut down at 2020-06-26 17:14:35 UTC
db_1 | 2020-06-26 17:14:38.267 UTC [48] LOG:  database system is ready to accept connections
db_1 | done
db_1 | server started

db_1 | 2020-07-01 08:07:15.893 UTC [1] LOG:  starting PostgreSQL 12.3 (Debian 12.3-1.pgdg100+1) on x86_64-pc-linux-gnu, compiled by gcc (Debian 8.3.0-6) 8.3.0, 64-bit
db_1 | 2020-07-01 08:07:15.895 UTC [1] LOG:  listening on IPv4 address "0.0.0.0", port 5432
db_1 | 2020-07-01 08:07:15.895 UTC [1] LOG:  listening on IPv6 address ":::", port 5432
db_1 | 2020-07-01 08:07:15.924 UTC [1] LOG:  listening on Unix socket "/var/run/postgresql/.s.PGSQL.5432"
db_1 | 2020-07-01 08:07:16.003 UTC [25] LOG:  database system was interrupted; last known up at 2020-06-29 19:09:14 UTC
db_1 | 2020-07-01 08:07:16.601 UTC [25] LOG:  database system was not properly shut down; automatic recovery in progress
db_1 | 2020-07-01 08:07:16.607 UTC [25] LOG:  redo starts at 0/16092F0
db_1 | 2020-07-01 08:07:16.607 UTC [25] LOG:  invalid record length at 0/16093D8: wanted 24, got 0
db_1 | 2020-07-01 08:07:16.607 UTC [25] LOG:  redo done at 0/16093A0
db_1 | 2020-07-01 08:07:16.654 UTC [1] LOG:  database system is ready to accept connections

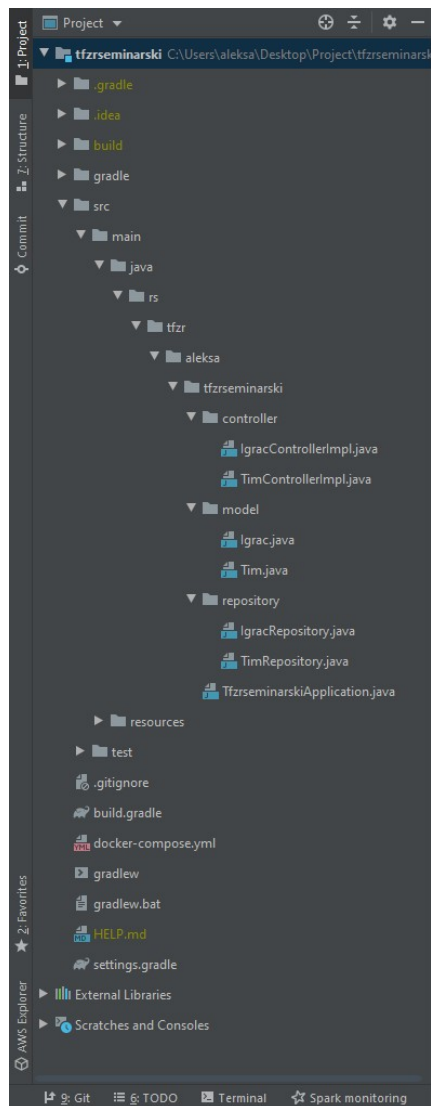
db_1 | /usr/local/bin/docker-entrypoint.sh: ignoring /docker-entrypoint-initdb.d/*
db_1 |
db_1 | 2020-06-26 17:14:38.316 UTC [48] LOG:  received fast shutdown request
db_1 | waiting for server to shut down....2020-06-26 17:14:38.321 UTC [48] LOG:  aborting any active transactions
db_1 | 2020-06-26 17:14:38.325 UTC [48] LOG:  background worker "logical replication launcher" (PID 55) exited with exit code 1
db_1 | 2020-06-26 17:14:38.325 UTC [50] LOG:  shutting down
db_1 | 2020-06-26 17:14:38.354 UTC [48] LOG:  database system is shut down
db_1 | done
db_1 | server stopped
db_1 |
db_1 | PostgreSQL init process complete; ready for start up.
db_1 |
db_1 | 2020-06-26 17:14:38.436 UTC [1] LOG:  starting PostgreSQL 12.3 (Debian 12.3-1.pgdg100+1) on x86_64-pc-linux-gnu, compiled by gcc (Debian 8.3.0-6) 8.3.0, 64-bit
db_1 | 2020-06-26 17:14:38.437 UTC [1] LOG:  listening on IPv4 address "0.0.0.0", port 5432
db_1 | 2020-06-26 17:14:38.438 UTC [1] LOG:  listening on IPv6 address ":::", port 5432
db_1 | 2020-06-26 17:14:38.445 UTC [1] LOG:  listening on Unix socket "/var/run/postgresql/.s.PGSQL.5432"
db_1 | 2020-06-26 17:14:38.467 UTC [57] LOG:  database system was shut down at 2020-06-26 17:14:38 UTC
db_1 | 2020-06-26 17:14:38.477 UTC [1] LOG:  database system is ready to accept connections
db_1 | 2020-06-26 17:15:37.543 UTC [64] FATAL:  password authentication failed for user "aleksa"
db_1 | 2020-06-26 17:15:37.543 UTC [64] DETAIL:  Role "aleksa" does not exist.
db_1 |         Connection matched pg_hba.conf line 95: "host all all all md5"
db_1 | 2020-06-26 17:15:37.543 UTC [65] FATAL:  password authentication failed for user "aleksa"
db_1 | 2020-06-26 17:15:37.543 UTC [65] DETAIL:  Role "aleksa" does not exist.
db_1 |         Connection matched pg_hba.conf line 95: "host all all all md5"
db_1 | 2020-06-26 17:53:36.972 UTC [1] LOG:  received smart shutdown request
db_1 | 2020-06-26 17:53:36.996 UTC [1] LOG:  background worker "logical replication launcher" (PID 63) exited with exit code 1
db_1 | 2020-06-26 17:53:47.005 UTC [81] FATAL:  terminating connection due to unexpected postmaster exit
db_1 | 2020-06-26 17:53:47.009 UTC [79] FATAL:  terminating connection due to unexpected postmaster exit
db_1 |
db_1 | PostgreSQL Database directory appears to contain a database; Skipping initialization

```

```

db_1 | 2020-06-29 14:43:30.904 UTC [1] LOG: starting PostgreSQL 12.3 (Debian 12.3-1.pgdg180+1) on x86_64-pc-linux-gnu, compiled by gcc (Debian 8.3.0-6) 8.3.0, 64-bit
db_1 | 2020-06-29 14:43:30.905 UTC [1] LOG: listening on IPv4 address "0.0.0.0", port 5432
db_1 | 2020-06-29 14:43:30.906 UTC [1] LOG: listening on IPv6 address ":::", port 5432
db_1 | 2020-06-29 14:43:30.915 UTC [1] LOG: listening on Unix socket "/var/run/postgresql/.s.PGSQL.5432"
db_1 | 2020-06-29 14:43:30.964 UTC [25] LOG: database system was interrupted; last known up at 2020-06-26 17:41:44 UTC
db_1 | 2020-06-29 14:43:31.077 UTC [25] LOG: database system was not properly shut down; automatic recovery in progress
db_1 | 2020-06-29 14:43:31.882 UTC [25] LOG: redo starts at 0/1679140
db_1 | 2020-06-29 14:43:31.882 UTC [25] LOG: invalid record length at 0/1679228: wanted 24, got 0
db_1 | 2020-06-29 14:43:31.882 UTC [25] LOG: redo done at 0/16791F0
db_1 | 2020-06-29 14:43:31.927 UTC [1] LOG: database system is ready to accept connections
db_1 | 2020-06-29 14:53:13.410 UTC [70] FATAL: database "testdb" does not exist
db_1 | 2020-06-29 14:53:14.906 UTC [71] FATAL: database "testdb" does not exist
db_1 | 2020-06-29 14:55:27.621 UTC [76] FATAL: database "testdb" does not exist
db_1 | 2020-06-29 14:55:29.127 UTC [77] FATAL: database "testdb" does not exist
db_1 | 2020-06-29 16:10:55.767 UTC [422] FATAL: database "seminarski.public" does not exist
db_1 | 2020-06-29 16:10:57.308 UTC [423] FATAL: database "seminarski.public" does not exist
db_1 | 2020-06-29 17:26:35.217 UTC [1] LOG: received smart shutdown request
db_1 | 2020-06-29 17:26:35.245 UTC [1] LOG: background worker "logical replication launcher" (PID 31) exited with exit code 1
db_1 |
db_1 | PostgreSQL Database directory appears to contain a database; Skipping initialization
db_1 |
db_1 | 2020-06-29 18:39:09.930 UTC [1] LOG: starting PostgreSQL 12.3 (Debian 12.3-1.pgdg180+1) on x86_64-pc-linux-gnu, compiled by gcc (Debian 8.3.0-6) 8.3.0, 64-bit
db_1 | 2020-06-29 18:39:09.931 UTC [1] LOG: listening on IPv4 address "0.0.0.0", port 5432
db_1 | 2020-06-29 18:39:09.931 UTC [1] LOG: listening on IPv6 address ":::", port 5432
db_1 | 2020-06-29 18:39:09.949 UTC [1] LOG: listening on Unix socket "/var/run/postgresql/.s.PGSQL.5432"
db_1 | 2020-06-29 18:39:09.973 UTC [25] LOG: database system was interrupted; last known up at 2020-06-29 16:23:32 UTC
db_1 | 2020-06-29 18:39:10.347 UTC [25] LOG: database system was not properly shut down; automatic recovery in progress
db_1 | 2020-06-29 18:39:10.352 UTC [25] LOG: redo starts at 0/16C7408
db_1 | 2020-06-29 18:39:10.352 UTC [25] LOG: invalid record length at 0/16C74F0: wanted 24, got 0
db_1 | 2020-06-29 18:39:10.352 UTC [25] LOG: redo done at 0/16C7488
db_1 | 2020-06-29 18:39:10.382 UTC [1] LOG: database system is ready to accept connections
db_1 | 2020-06-29 19:03:59.223 UTC [170] ERROR: cannot insert into column "tim_id"
db_1 | 2020-06-29 19:03:59.223 UTC [170] DETAIL: Column "tim_id" is an identity column defined as GENERATED ALWAYS.
db_1 | 2020-06-29 19:03:59.223 UTC [170] HINT: Use overriding system value to override.
db_1 | 2020-06-29 19:03:59.223 UTC [170] STATEMENT: insert into tim (tim_time, tim_osnovan, tim_trener, tim_id) values ($1, $2, $3, $4)
db_1 | 2020-06-29 19:13:22.284 UTC [1] LOG: received smart shutdown request
db_1 | 2020-06-29 19:13:22.296 UTC [1] LOG: background worker "logical replication launcher" (PID 31) exited with exit code 1
db_1 |
db_1 | PostgreSQL Database directory appears to contain a database; Skipping initialization
db_1 |

```



```
1  # Use postgres/example model.user/password credentials
2  # Author: Aleksa Cakic
3  version: '3.1'
4  >> services:
5  >    db:
6  >      image: postgres
7  >      restart: always
8  >      ports:
9  >          - "5432:5432"
10 >      environment:
11 >          POSTGRES_PASSWORD: root
12
```

Struktura java projekta I Docker compose fajl.

```

1  plugins {
2      id 'org.springframework.boot' version '2.3.1.RELEASE'
3      id 'io.spring.dependency-management' version '1.0.9.RELEASE'
4      id 'java'
5  }
6
7  group = 'rs.tfzr.aleksa'
8  version = '0.0.1-SNAPSHOT'
9  sourceCompatibility = '1.8'
10
11  repositories {
12      mavenCentral()
13  }
14
15  dependencies {
16      implementation 'org.springframework.boot:spring-boot-starter-thymeleaf'
17      implementation 'org.springframework.boot:spring-boot-starter-web'
18      developmentOnly 'org.springframework.boot:spring-boot-devtools'
19      runtimeOnly 'org.postgresql:postgresql'
20      testImplementation('org.springframework.boot:spring-boot-starter-test') {
21          exclude group: 'org.junit.vintage', module: 'junit-vintage-engine'
22      }
23
24      // https://mvnrepository.com/artifact/org.springframework.boot/spring-boot-starter-data-jpa
25      compile group: 'org.springframework.boot', name: 'spring-boot-starter-data-jpa', version: '2.3.1.RELEASE'
26      // https://mvnrepository.com/artifact/org.springframework.data/spring-data-jpa
27      compile group: 'org.springframework.data', name: 'spring-data-jpa', version: '2.3.1.RELEASE'
28
29  }
30
31  test {
32      useJUnitPlatform()
33  }
34
35
36  spring.datasource.url=jdbc:postgresql://localhost/seminarski
37  spring.datasource.username=postgres
38  spring.datasource.password=root
39  spring.jpa.generate-ddl=true

```

Gradle

I

Spring

konfiguracija.

```

1  package rs.tfzr.aleksa.tfzrseminarski;
2
3  import org.springframework.boot.SpringApplication;
4  import org.springframework.boot.autoconfigure.SpringBootApplication;
5
6  @SpringBootApplication
7  ▶ public class TfzrseminarskiApplication {
8
9      public static void main(String[] args) {
10         SpringApplication.run(TfzrseminarskiApplication.class, args);
11     }
12
13

```

```

1  -- Database: seminarski
2
3  -- DROP DATABASE seminarski;
4
5  CREATE DATABASE seminarski
6      WITH
7      OWNER = postgres
8      ENCODING = 'UTF8'
9      LC_COLLATE = 'en_US.utf8'
10     LC_CTYPE = 'en_US.utf8'
11     TABLESPACE = pg_default
12     CONNECTION LIMIT = -1;
13
14     COMMENT ON DATABASE seminarski
15         IS 'Baza za seminarski iz softverskih prevodioca';

```








```
1  create table igrac
2  (
3      igrac_id      integer not null
4      |      constraint igrac_pkey
5      |      primary key,
6      igrac_broj     integer,
7      igrac_godiste  integer,
8      igrac_ime      varchar(255),
9      igrac_prezime  varchar(255),
10     igrac_tim       varchar(255)
11 );
12
13 alter table igrac
14     owner to postgres;
```

Postgresql skripte

```
32      @Autowired
33      @ @
34      public Igrac(int igrac_id, String igrac_ime, String igrac_prezime, int igrac_broj, String igrac_tim, int igrac_godiste) {
35          this.igrac_id = igrac_id;
36          this.igrac_ime = igrac_ime;
37          this.igrac_prezime = igrac_prezime;
38          this.igrac_broj = igrac_broj;
39          this.igrac_tim = igrac_tim;
40          this.igrac_godiste = igrac_godiste;
41      }
42      @
43      public Igrac() { }
```



```

1      package rs.tfzr.aleksa.tfzrseminarski.model;
2
3      import org.springframework.beans.factory.annotation.Autowired;
4
5      import javax.persistence.*;
6      import java.io.Serializable;
7
8      @Entity
9      @Table(name = "igrac")
10      public class Igrac implements Serializable {
11
12         @Id
13         @Column(name = "igrac_id")
14         @GeneratedValue(strategy = GenerationType.AUTO)
15          private int igrac_id;
16
17         @Column(name = "igrac_ime")
18          private String igrac_ime;
19
20         @Column(name = "igrac_prezime")
21          private String igrac_prezime;
22
23         @Column(name = "igrac_broj")
24          private int igrac_broj;
25
26         @Column(name = "igrac_tim")
27          private String igrac_tim;
28
29         @Column(name = "igrac_godiste")
30          private int igrac_godiste;

```

```

78     @PutMapping("/{igracis/{id}") // <3
79     public ResponseEntity<Igrac> updateIgrac(@PathVariable("id") int igrac_id, @RequestBody Igrac igrac) {
80         System.out.println("Izmenjeni podaci za igraca " + igrac_id + "... ");
81
82         Optional<Igrac> igracData = repository.findById(igrac_id);
83
84         if (igracData.isPresent()) {
85             Igrac _igrac = igracData.get();
86             _igrac.setIgrac_ime(igrac.getIgrac_ime());
87             _igrac.setIgrac_prezime(igrac.getIgrac_prezime());
88             _igrac.setIgrac_broj(igrac.getIgrac_broj());
89             _igrac.setIgrac_godiste(igrac.getIgrac_godiste());
90             _igrac.setIgrac_tim(igrac.getIgrac_tim());
91             return new ResponseEntity<Igrac>(_igrac, HttpStatus.OK);
92         } else {
93             return new ResponseEntity<>(HttpStatus.NOT_FOUND);
94         }
95     }
96 }

```

```

48     @GetMapping("/{igracis}")
49     public List<Igrac> getSviIgraci() {
50         System.out.println("Uziman sve igrace");
51
52         List<Igrac> igraci = new ArrayList<>();
53         repository.findAll().forEach(igraci::add);
54
55         return igraci;
56     }
57
58     @PostMapping(value = "/igracis/napravi")
59     public Igrac postIgrac(@RequestBody Igrac igrac) {
60         Igrac _igrac =
61             repository
62                 .save(new Igrac(
63                     igrac.getIgrac_id(), igrac.getIgrac_ime(), igrac.getIgrac_prezime(), igrac.getIgrac_broj(), igrac.getIgrac_tim(), igrac.getIgrac_godiste()
64                 ));
65
66         return _igrac;
67     }
68
69     @DeleteMapping("/{igracis/{id}")
70     public ResponseEntity<String> obrisiIgraca(@PathVariable("id") int igrac_id) {
71         System.out.println("Brisanje igrac sa id: " + igrac_id + "... ");
72
73         repository.deleteById(igrac_id);
74
75         return new ResponseEntity<>("Igrac je uspesno obrisao!", HttpStatus.OK);
76     }







```

```

38     @GetMapping("/timovis")
39     public List<Tim> getSviTimovi() {
40         System.out.println("Uzimam sve timove");
41
42         List<Tim> timovi = new ArrayList<>();
43         repository.findAll().forEach(timovi::add);
44
45         return timovi;
46     }
47
48     @PostMapping("/timovis/napravi")
49     public Tim postTim (@RequestBody Tim tim) {
50         Tim _tim =
51             repository
52                 .save(new Tim(
53                     tim.getTim_id(), tim.getTim_ime(), tim.getTim_osnovan(), tim.getTim_trener()
54                 ));
55
56         return _tim;
57     }
58
59     @DeleteMapping("/timovis/{id}")
60     public ResponseEntity<String> obrisiTim(@PathVariable("id") int tim_id) {
61         System.out.println("Briše se tim sa id: " + tim_id + "... ");
62
63         repository.deleteById(tim_id);
64
65         return new ResponseEntity<>("Tim je uspesno obrisao!", HttpStatus.OK);
66     }

```

```

1      package rs.tfzr.aleksa.tfzrseminarski.model;
2
3      import org.springframework.beans.factory.annotation.Autowired;
4
5      import javax.persistence.*;
6      import java.io.Serializable;
7
8      @Entity
9      @Table(name = "tim")
10      public class Tim implements Serializable {
11
12         @Id
13         @Column(name = "tim_id")
14         @GeneratedValue(strategy = GenerationType.AUTO)
15          private int tim_id;
16
17         @Column(name = "tim_ime")
18          private String tim_ime;
19
20         @Column(name = "tim_osnovan")
21          private int tim_osnovan;
22
23         @Column(name = "tim_trener")
24          private String tim_trener;
25
26         @Autowired
27           @ public Tim(int tim_id, String tim_ime, int tim_osnovan, String tim_trener) {
28             this.tim_id = tim_id;
29             this.tim_ime = tim_ime;
30             this.tim_osnovan = tim_osnovan;
31             this.tim_trener = tim_trener;
32         }
33
34         @ public Tim() { }

```

```

68     @PutMapping("/timovis/{id}")
69     public ResponseEntity<Tim> updateTim(@PathVariable("id") int tim_id, @RequestBody Tim tim) {
70         System.out.println("Izmenjeni podaci za tim " + tim_id + "... ");
71
72         Optional<Tim> timData = repository.findById(tim_id);
73
74         if (timData.isPresent()) {
75             Tim _tim = timData.get();
76             _tim.setTim_ime(tim.getTim_ime());
77             _tim.setTim_osnovan(tim.getTim_osnovan());
78             _tim.setTim_trener(tim.getTim_trener());
79             return new ResponseEntity<>(repository.save(_tim), HttpStatus.OK);
80         } else {
81             return new ResponseEntity<>(HttpStatus.NOT_FOUND);
82         }
83     }
84 }
85

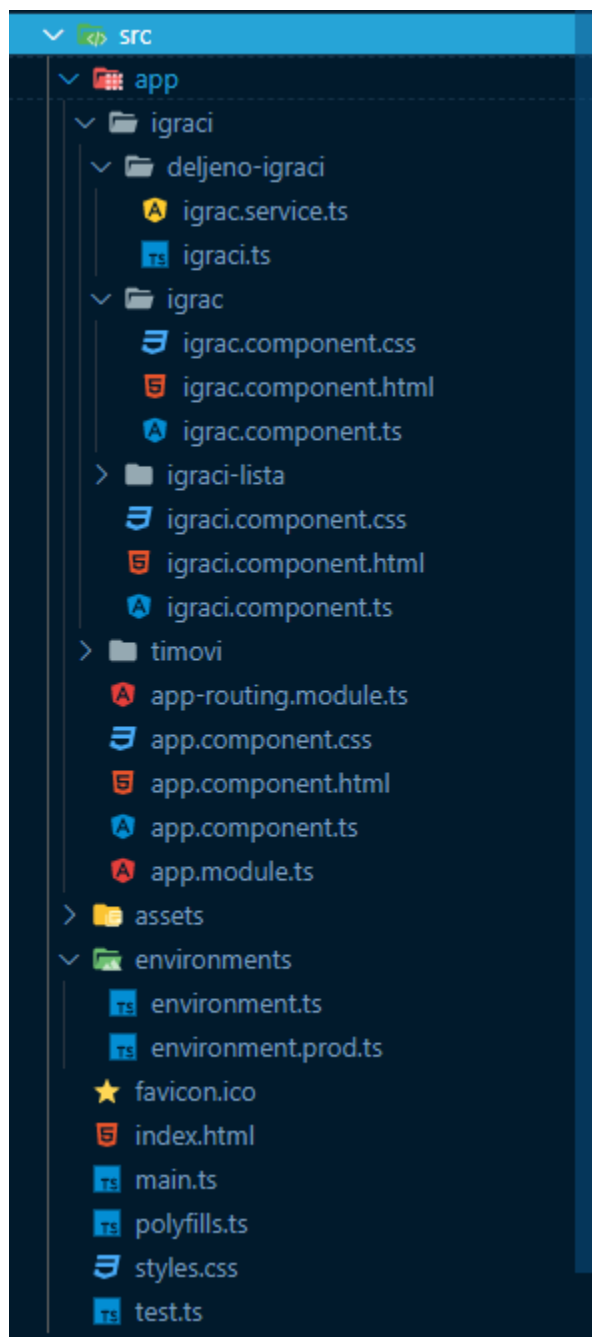
```

```

1     package rs.tfzr.aleksa.tfzrseminarski.repository;
2
3     import org.springframework.data.repository.CrudRepository;
4     import org.springframework.stereotype.Repository;
5     import rs.tfzr.aleksa.tfzrseminarski.model.Tim;
6
7     @Repository
8     public interface TimRepository extends CrudRepository<Tim, Integer> { }
9

```

Modeli Igraca I Timova I kontroleri za iste.



Front end struktura projekta – prikaz zadatka

```

1  import { NgModule } from '@angular/core';
2  import { Routes, RouterModule } from '@angular/router';
3  import { IgraciComponent } from './igraci/igraci.component';
4  import { TimoviComponent } from './timovi/timovi.component';
5
6  const routes: Routes = [
7    { path: 'igraci', component: IgraciComponent },
8    { path: 'timovi', component: TimoviComponent },
9  ];
10
11  @NgModule({
12    imports: [RouterModule.forRoot(routes)],
13    exports: [RouterModule]
14  })
15  export class AppRoutingModule { }
16

```

```

1  <div class="container" style="text-align: center;">
2
3    <nav class="navbar navbar-expand-lg navbar-dark bg-dark">
4      <a class="navbar-brand" href="#">
5        
9      <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarNav"
10        aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle navigation">
11        <span class="navbar-toggler-icon"></span>
12      </button>
13      <div class="collapse navbar-collapse" id="navbarNav">
14        <ul class="navbar-nav">
15          <li class="nav-item active">
16            <a class="nav-link" href="/">Glavna<span class="sr-only">(current)</span></a>
17          </li>
18          <li class="nav-item">
19            <a class="nav-link" routerLink="/igraci" routerLinkActive="active">Igraci</a>
20          </li>
21          <li class="nav-item">
22            <a class="nav-link" routerLink="/timovi" routerLinkActive="active">Timovi</a>
23          </li>
24        </ul>
25      </div>
26    </nav>
27    <div class="jumbotron jumbotron-fluid">
28      <div class="jumbotron">
29        <h1 class="display-4">Seminarski rad</h1>
30        <p class="lead">Tema: Upoređivanje gramatike programskog jezika C# i Java na primeru objektno-orijentisane aplikacije uz primenu baze podataka. TIP 5/a- 40 bodova</p>
31        <hr class="my-4">
32        <picture>
33          
34        </picture>
35        <p>
36          Front end i Back end aplikacija. Front end je uradjen u Angular 8. Back end je uradjen u ASP.NET. Baza podataka je MSSql koja je defaultna u C# projektima.
37          Vreme utroseno je svega par sati. Sledeca aplikacija je Java.
38        </p>
39        <a class="btn btn-primary btn-lg" href="http://www.tfzr.uns.ac.rs" role="button" target="blank_">Oficijelni Saj</a>
40      </div>
41    </div>
42
43    <!-- ===== <app-igraci></app-igraci> ===== -->
44  </div>
45
46  <router-outlet></router-outlet>

```



```

1  import { Component } from '@angular/core';
2
3  @Component({
4    selector: 'app-root',
5    templateUrl: './app.component.html',
6    styleUrls: ['./app.component.css']
7  })
8  export class AppComponent {
9    title = 'seminarskialeksacakicsi2317frontcsharp';
10 }
11

```

```

1  import { BrowserModule } from '@angular/platform-browser';
2  import { NgModule } from '@angular/core';
3
4  import { AppRoutingModule } from './app-routing.module';
5  import { AppComponent } from './app.component';
6  import { IgraciComponent } from './igraci/igraci.component';
7  import { IgracComponent } from './igraci/igrac/igrac.component';
8  import { IgraciListaComponent } from './igraci/igraci-lista/igraci-lista.component';
9
10 import { HttpClientModule } from '@angular/http';
11 import { FormsModule } from '@angular/forms';
12 import { TimoviComponent } from './timovi/timovi.component';
13 import { TimComponent } from './timovi/tim/tim.component';
14 import { TimoviListaComponent } from './timovi/timovi-lista/timovi-lista.component';
15
16 @NgModule({
17   declarations: [
18     AppComponent,
19     IgraciComponent,
20     IgracComponent,
21     IgraciListaComponent,
22     TimoviComponent,
23     TimComponent,
24     TimoviListaComponent,
25   ],
26   imports: [
27     BrowserModule,
28     AppRoutingModule,
29     HttpClientModule,
30     FormsModule
31   ],
32   providers: [],
33   bootstrap: [AppComponent]
34 })
35 export class AppModule { }
36

```



```

1  <!doctype html>
2  <html lang="en">
3
4  <head>
5    <meta charset="utf-8">
6    <title>Seminarskialeksacikicsi2317frontcsharp</title>
7    <base href="/">
8
9    <meta name="viewport" content="width=device-width, initial-scale=1">
10   <link rel="icon" type="image/x-icon" href="favicon.ico">
11   <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/css/bootstrap.min.css"
12     integrity="sha384-9aIt2nRpC12Uk9gS9baD1411NQApFmC26EwAOH8WgZl5MYYYxFfc+NcPb1dKGj7Sk" crossorigin="anonymous">
13   <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">
14 </head>
15
16 <body>
17   <app-root></app-root>
18   <!-- JS, Popper.js, and jQuery -->
19   <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"
20     integrity="sha384-DfXdz2htPH0lsSSs5nCTpuj/zy4C+OGpamoFVy38MVBnE+IbbVYUew+OrCXaRkfj"
21     crossorigin="anonymous"></script>
22   <script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.js"
23     integrity="sha384-Q6E9RHvbIyZFJoft+2mJbHaEWldlvI9IOYy5n3zV9zzTtmI3UksdQRVvoxMfooAo"
24     crossorigin="anonymous"></script>
25   <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/js/bootstrap.min.js"
26     integrity="sha384-OgVRvuATP1z7JjHLku0U7Xw704+h835Lr+6QL9UvYjZE3Ipu6Tp75j7Bh/kR0JKI"
27     crossorigin="anonymous"></script>
28 </body>
29
30 </html>
31

```

```

1  /* You can add global styles to this file, and also import other style files */
2  form.igrac-form {
3    background-color: gray;
4    padding: 10px;
5    border-radius: 4px;
6  }
7
8  form.tim-form {
9    background-color: gray;
10   padding: 10px;
11   border-radius: 4px;
12 }
13

```

```

1 <table class="table table-dark table-striped table-sm table-hover">
2
3 <thead>
4   <tr>
5     <th scope="col">Ime</th>
6     <th scope="col">Prezime</th>
7     <th scope="col">Broj</th>
8     <th scope="col">Tim</th>
9     <th scope="col">Godiste</th>
10  </tr>
11 </thead>
12 <tbody>
13   <tr *ngFor="let igrac of igracservice.ListaIgraca">
14     <td>{{ igrac.igrac_ime }}</td>
15     <td>{{ igrac.igrac_prezime }}</td>
16     <td>{{ igrac.igrac_broj }}</td>
17     <td>{{ igrac.igrac_tim }}</td>
18     <td>{{ igrac.igrac_godiste }}</td>
19
20     <td>
21       <a class="btn btn-warning" (click)="showEdit(igrac)">
22         <i class="fa fa-pencil"></i>
23       </a>
24       <a class="btn btn-danger" (click)="onDelete(igrac.igrac_id)">
25         <i class="fa fa-trash"></i>
26       </a>
27     </td>
28   </tr>
29 </tbody>
30
31 </table>
32

```

```

1  <form class="igrac-form" #igrac_form="ngForm" (ngSubmit)="OnSubmit(igrac_form)">
2
3  <input type="hidden" name="igrac_id" #igrac_id="ngModel" [(ngModel)]="igracservice.OdabraniIgraci.igrac_id">
4  <div class="form-row">
5    <div class="form-group col-md-6">
6      <input class="form-control" name="igrac_ime" #igrac_ime="ngModel" [(ngModel)]="igracservice.OdabraniIgraci.igrac_ime" placeholder="Ime igraca">
7    </div>
8
9    <div class="form-group col-md-6">
10     <input class="form-control" name="igrac_prezime" #igrac_prezime="ngModel" [(ngModel)]="igracservice.OdabraniIgraci.igrac_prezime" placeholder="Prezime igraca">
11    </div>
12
13    <div class="form-group col-md-6">
14     <input class="form-control" name="igrac_broj" #igrac_broj="ngModel" [(ngModel)]="igracservice.OdabraniIgraci.igrac_broj" placeholder="Broj dresa igraca">
15    </div>
16
17    <div class="form-group col-md-6">
18     <input class="form-control" name="igrac_tim" #igrac_tim="ngModel" [(ngModel)]="igracservice.OdabraniIgraci.igrac_tim" placeholder="Tim za koji igra">
19    </div>
20
21    <div class="form-group col-md-6">
22     <input class="form-control" name="igrac_godiste" #igrac_godiste="ngModel" [(ngModel)]="igracservice.OdabraniIgraci.igrac_godiste" placeholder="Datum rođenja">
23    </div>
24
25    <div class="form-group col-md-8">
26     <button type="submit" class="btn btn-lg btn-block btn-info">Unesi</button>
27    </div>
28
29    <div class="col-md-4">
30     <button type="button" class="btn btn-lg btn-block btn-dark" (click)="ResetForm(igrac_form)">Resetuj</button>
31    </div>
32  </div>
33 </form>
34
35
36

```

```

1  import { Component, OnInit } from '@angular/core';
2  import { IgracService } from '../deljeno-igraci/igrac.service';
3  import { NgForm } from '@angular/forms';
4  import { NullTemplateVisitor } from '@angular/compiler';
5
6  @Component({
7    selector: 'app-igrac',
8    templateUrl: './igrac.component.html',
9    styleUrls: ['./igrac.component.css']
10 })
11 export class IgracComponent implements OnInit {
12
13   constructor(public igracservice : IgracService) { }
14
15   ngOnInit() {
16     this.ResetForm();
17   }
18
19   ResetForm(form? : NgForm) {
20     if(form != null)form.reset();
21     this.igracservice.OdabraniIgraci = {
22       igrac_id : null,
23       igrac_ime : '',
24       igrac_prezime : '',
25       igrac_broj : null,
26       igrac_tim : '',
27       igrac_godiste : null,
28     }
29   }
30
31   OnSubmit(form : NgForm) {
32     if(form.value.igrac_id == null) {
33       this.igracservice.Postigraci(form.value).subscribe(data => {
34         this.ResetForm(form);
35         this.igracservice.GetIgraci();
36         console.log('Aleksa');
37       })
38     } else {
39       this.igracservice.Putigraci(form.value.igrac_id, form.value).subscribe(data => {
40         this.ResetForm(form);
41         this.igracservice.GetIgraci();
42       })
43     }
44   }
45 }
46

```

```

1  import { Component, OnInit } from '@angular/core';
2  import { IgracService } from '../deljeno-igraci/igrac.service';
3  import { Igraci } from '../deljeno-igraci/igraci';
4
5  @Component({
6    selector: 'app-igraci-lista',
7    templateUrl: './igraci-lista.component.html',
8    styleUrls: ['./igraci-lista.component.css']
9  })
10 export class IgraciListaComponent implements OnInit {
11
12     constructor(public igracservice : IgracService) { }
13
14     ngOnInit() {
15         this.igracservice.GetIgraci();
16     }
17
18     showEdit(igrac : Igraci) {
19         this.igracservice.OdabraniIgraci = Object.assign({}, igrac);
20         this.ngOnInit();
21     }
22
23     onDelete(id : number) {
24         if(confirm('Da li ste sigurni da zelite da obrisete dati unos?') == true) {
25             this.igracservice.DeleteIgrac(id).subscribe(x => {
26                 this.ngOnInit();
27             })
28         }
29     }
30 }
31

```

```

1  export class Igraci {
2      igrac_id : number;
3      igrac_ime : string;
4      igrac_prezime : string;
5      igrac_broj : number;
6      igrac_tim : string;
7      igrac_godiste : number;
8  }
9

```

src > app > igraci > deljeno-igraci > igrac.service.ts > IgracService > DeleteIgrac

```
1 import { Injectable } from '@angular/core';
2 import { Igraci } from '../deljeno-igraci/igraci';
3 import { Http, Response, Headers, RequestOptions, RequestMethod } from '@angular/http';
4 import { Observable } from 'rxjs/Observable';
5 import 'rxjs/add/operator/map';
6 import 'rxjs/add/operator/toPromise';
7
8 @Injectable({
9   providedIn: 'root'
10 })
11 export class IgracService {
12   OdabraniIgraci : Igraci;
13   ListaIgraca : Igraci[];
14   constructor(public http : Http) { }
15
16   Postigraci(igrac : Igraci) {
17     var body = JSON.stringify(igrac);
18     var headerpotion = new Headers({ "Content-Type" : "application/json" });
19     var requestoption = new RequestOptions({ method : RequestMethod.Post, headers: headerpotion });
20     // return this.http.post('https://localhost:44309/api/igracis', body, requestoption).map(x => x.json());
21     return this.http.post('http://localhost:8080/api/igracis/napravi', body, requestoption).map(x => x.json());
22   }
23   //8080
24   Putigraci(id, igrac) {
25     var body = JSON.stringify(igrac);
26     var headerpotion = new Headers({ "Content-Type" : "application/json" });
27     var requestoption = new RequestOptions({ method : RequestMethod.Put, headers: headerpotion });
28     // return this.http.put('https://localhost:44309/api/igracis/' + id, body, requestoption).map(x => x.json());
29     return this.http.put('http://localhost:8080/api/igracis/' + id, body, requestoption).map(x => x.json());
30   }
31
32   GetIgraci() {
33     // this.http.get('https://localhost:44309/api/igracis').map((data : Response) => {
34     this.http.get('http://localhost:8080/api/igracis').map((data : Response) => {
35       return data.json() as Igraci[];
36     }).toPromise().then(x => {
37       this.ListaIgraca = x;
38     })
39   }
40
41   DeleteIgrac(id : number) {
42     // return this.http.delete('https://localhost:44309/api/igracis/' + id).map(res => res.json());
43     return this.http.delete(['http://localhost:8080/api/igracis/' + id]).map(res => res.json());
44   }
45 }
46
```

```

1  import { Component, OnInit } from '@angular/core';
2  import { IgracService } from '../deljeno-igraci/igrac.service';
3
4  @Component({
5    selector: 'app-igraci',
6    templateUrl: './igraci.component.html',
7    styleUrls: ['./igraci.component.css']
8  })
9  export class IgraciComponent implements OnInit {
10
11    constructor(public igracservice : IgracService) { }
12
13    ngOnInit() {
14    }
15
16  }
17

```

```

1  <div class="container">
2    <div class="row">
3      <div class="col-md-6">
4        <app-igrac></app-igrac>
5      </div>
6      <div class="col-md-6">
7        <app-igraci-lista></app-igraci-lista>
8      </div>
9    </div>
10 </div>
11

```

Projekat celokupno nije oduzeo više od dva sata.

3. Greške

3.1 RunTime

Greške koje nastaju za vreme rada programa (runtime errors) – jesu greške koje kompajleri/interpreteri ne mogu da uoče ali se one ipak ispoljavaju u toku rada tj. korišćenja programa. Ove greške, ukoliko se ne „uhvate“ posebnim programskim strukturama u kodu (najčešće try-catch metoda), mogu da izazovu prestanak rada programa uz gubitak svih podataka i međurezultata neke obrade koji su postojali do momenta pojave greške. Tipični primeri ovakvih grešaka u kodu se pojavljuju kada se jednoj promenljivoj numeričkog tipa pokuša da dodeli vrednost tekstualnog tipa koja ne može da podleže konvertovanju u broj (osim u slučaju ako sam tekst nije i broj). Tu je zatim tipična greška deljenja nekog broja sa nulom, pokušaj povezivanja na server baze podataka kada su parametri konekcije loše zadati ili kada je server neaktivan i sl. Runtime greške, ukoliko postoji dobra praksa programiranja, mogu da budu uhvaćene, ubeležene (log sistem) i preduhitrene na način koji neće obustaviti dalji rad programa.

Izgled RunTime greške u C#

```
// PUT: api/timovis/5
[ResponseType(typeof(void))]
public IHttpActionResult Puttimovi(int id, timovi timovi)
{
    //if (!ModelState.IsValid)
    //{
    //    return BadRequest(ModelState);
    //}

    if (id != timovi.tim_id)
    {
        var a = id / 0;
        return BadRequest();
    }

    db.Entry(timovi).State = EntityState.Modified;

    try
    {
        db.SaveChanges();
    }
    catch (DbUpdateConcurrencyException)
    {
        if (!timoviExists(id))
        {
            return NotFound();
        }
        else
        {
            throw;
        }
    }

    return StatusCode(HttpStatusCode.NoContent);
}
```

Izgled RunTime greške u Javi

```
package rs.tfzr.aleksa.tfzrseminarski;

public class Greske {
    public static int promenljiva1 = 10;
    public static int getPromenljiva2 = promenljiva1/0;

    public int printuj() {
        System.out.print(getPromenljiva2);
        return getPromenljiva2;
    }
}
```

3.2 Logičke

Aplikacija je uspešno kompajlirana, prilikom rada ne prijavljuje nikakve greške niti prestaje sa radom. Sve je u redu, ali vaš program “samo” ne radi ono što treba da radi. Vraća pogrešne rezultate, ne upisuje podatke i tome slično. Ovo su verovatno greške koje je najteže otkriti jer su posledice lošeg dizajna aplikacije.

Izgled logičke greške u C#

```
// TODO - Ovde se nalazi LOGICKA GRESKA
// GET: api/timovis/5
[ResponseType(typeof(timovi))]
public IHttpActionResult Gettimovi(int id)
{
    timovi timovi = db.timovis.Find(id);
    if (timovi == null)
    {
        return NotFound();
    }
    else if (timovi == Controllers.igracisController)
    {
        Console.WriteLine("Logička greška ovde!");
    }

    return Ok(timovi);
}
```

Izgled logičke greške u Javi

```
int aleksa = 5 + 4 * 3 / 2;
int aleksa2 = (5 + 4) * (3 / 2);
int aleksa3 = (5 + 4) * (3 / 2);
int aleksa4 = (5 + (4 * 3) / 2);

public void logickaGreksa() {
    System.out.println("Aleksa: " + aleksa + " // ");
    System.out.println("Aleksa2: " + aleksa2 + " // ");
    System.out.println("Aleksa3: " + aleksa3 + " // ");
    System.out.println("Aleksa4: " + aleksa4 + " // ");
}
```

3.3 Sintaksičke

Slično se dešava i u programiranju. Pravila za pisanje programa su vrlo precizna i za svaki deo programa je tačno određeno kako se može napisati i šta u tom slučaju znači. Skup pravila za pisanje programa na nekom jeziku naziva se sintaksa. Sintaksna greška nastaje kada kompjuter ne razume neki deo tvog programa, zato što nije napisan tačno po pravilima.

Kao što smo videli, može biti dovoljno da se u rečenici zameni samo jedno slovo, pa da ona dobije smisao. Ipak, kompjuteru je sasvim svejedno da li je rečenica (program) skoro ispravna ili potpuno besmislena, poput „Dobrome dva nežno krug“ ili „adf#fgh\$ty!p o?e&c*nm^,.%“. Kompjuter zahteva potpunu preciznost u izražavanju i uvek samo prijavljuje šta tačno ne razume, jer nije sposoban da pogodi šta je trebalo da bude napisano i da sam ispravi grešku, ma koliko mala bila.

Izgled sintaksičke greške u C#

```
// TODO - Ovde se vidi sintaksicka greska
// GET: api/timovis
public IQueryable<timovi> Gettimovis()
{
    if (Gettimovi == null)
        System.out.println("Java i python sintaksa u c# programu, jer su bolji od c#-a")
        var takodjeMozeJS = function () {
            new Promise(nestoOvdeStaviti).then(handleOvuStvar, handleOdbijenuStvar);
            .catch(ovoNijeCsharp);
        }

    return db.timovis;
}
```

Izgled sintaksičke greške u Javi

```
def Sintaksicka_Greksa
    python_kod.os("sintaksa pythona")
    mesano.SaJsSintaksom(data => {
        Console.witeline("C#");
    }).then(uradiOvo, odbiOvo);
}
```

S