Izrada web aplikacije za poslovanje mjenjačnice

Nedić, Dario

Master's thesis / Diplomski rad

2018

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

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:200:016832

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-04-19**

Repository / Repozitorij:

Faculty of Electrical Engineering, Computer Science and Information Technology Osijek

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

Sveučilišni studij

IZRADA WEB APLIKACIJE ZA POSLOVANJE MJENJAČNICE

Diplomski rad

Dario Nedić

Osijek, 2018.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Obrazac D1: Obrazac za imenovanje Povjerenstva za obranu diplomskog rada

Osijek, 07.05.2018.

Odboru za završne i diplomske ispite

Imenovanje Povjerenstva za obranu diplomskog rada

Ime i prezime studenta:	Dario Nedić
Studij, smjer:	Diplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	D-788R, 25.09.2017.
OIB studenta:	11109933900
Mentor:	Izv. prof. dr. sc. Krešimir Nenadić
Sumentor:	
Sumentor iz tvrtke:	
Predsjednik Povjerenstva:	Doc.dr.sc. Alfonzo Baumgartner
Član Povjerenstva:	Doc.dr.sc. Mirko Köhler
Naslov diplomskog rada:	Izrada web aplikacije za poslovanje mjenjačnice
Znanstvena grana rada:	**Informacijski sustavi (zn. polje računarstvo)**
Zadatak diplomskog rada:	Kratko opisati način rada mjenjačnica valuta. Dati prijedlog rada web aplikacije koja može olakšati rad mjenjačnice. Opisati tehnologije koje se koriste u izradi web aplikacije. Testirati rad web aplikacije za neke testne podatke.
Prijedlog ocjene pismenog dijela ispita (diplomskog rada):	Izvrstan (5)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 3 bod/boda Postignuti rezultati u odnosu na složenost zadatka: 2 bod/boda Jasnoća pismenog izražavanja: 3 bod/boda Razina samostalnosti: 3 razina
Datum prijedloga ocjene mentora:	07.05.2018.

Potpis mentora za predaju konačne verzije rada u Studentsku službu pri završetku studija:	Potpis:
	Datum:

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

IZJAVA O ORIGINALNOSTI RADA

Osijek, 30.05.2018.

Ime i prezime studenta:	Dario Nedić
Studij:	Diplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	D-788R, 25.09.2017.
Ephorus podudaranje [%]:	10%

Ovom izjavom izjavljujem da je rad pod nazivom**: Izrada web aplikacije za poslovanje mjenjačnice**

izrađen pod vodstvom mentora Izv. prof. dr. sc. Krešimir Nenadić

i sumentora

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

Potpis studenta:

Sadržaj

1. UVOD 1
2. UVOD U RAD MJENJAČNICE 2
   2.1. Opis rada 2
   2.2. Valute 2
   2.3. Tečajne liste 3
   2.4. Osnovni procesi u radu mjenjačnice 3
   2.5. Zahtjevi prema aplikaciji 4
3. KORIŠTENI ALATI I TEHNOLOGIJE 5
   3.1. ASP.NET 5
   3.2. MVC 6
      3.2.1. Razor 7
   3.3. Entity Framework 9
      3.3.1. Code-First 11
      3.3.2. LINQ upiti 12
   3.4. ASP.NET Identity 14
   3.5. Microsoft ReportViewer 14
4. IZRADA APLIKACIJE 15
   4.1. Projektna struktura 15
   4.2. Kreiranje baze podataka 16
   4.3. Autentifikacija i autorizacija korisnika 19
      4.3.1. Kreiranje novih korisnika i prijava 19
      4.3.2. Ograničavanje pristupa 22
   4.4. Izrada izvještaja 24
   4.5. Pregled i kreiranje transakcija 26
   4.6. Dohvaćanje i učitavanje tečajnih lista 28
   4.7. Implementacija blagajne 32
5. ZAKLJUČAK 36

LITERATURA 37

SAŽETAK 38

ŽIVOTOPIS 40

PRILOZI 41

1. UVOD

U današnje vrijeme, tvrtke koje se bave razvojem i implementacijom poslovnih rješenja, svoje proizvode najčešće isporučuju klijentima u dva oblika, kao desktop ili kao web aplikacije. Desktop aplikacije se pokreće na klijentskom računalu i tradicionalno su ograničene hardverom na kojem se pokreću. Razvijaju se i instaliraju na određeni operativni sustav, a mogu imati stroge hardverske zahtjeve koji moraju biti ispunjeni kako bi aplikacija ispravno funkcionirala. Web aplikacije su programska rješenja kojima se pristupa putem internetskog preglednika koristeći internet ili intranet. Glavna prednost im je što su dostupne u bilo koje vrijeme s bilo kojeg mjesta, koristeći računalo ili mobilni uređaj. Web aplikacije nije potrebno periodički nadograđivati na svakom računalu s kojeg im se pristupa, jer im se pristupa identično kao i ostalim web stranicama, s bilo kojeg računala putem internet preglednika. Upravo je to razlog zbog kojeg se tvrtke okreću web aplikacijama kao načinu isporuke svojih programskih rješenja.

Zadatak ovog rada je izrada poslovne web aplikacije za poslovanje mjenjačnice. Aplikacija je namijenjena mjenjačnicama koje mogu imati više djelatnika. Omogućuje im kreiranje transakcija u obliku otkupa i prodaje stranih valuta, svakodnevno povlačenje najnovijih tečajnih lista, izmjenu standardnih valuta, praćenje vlastitog prometa, generiranje potvrda transakcija kao i generiranje izvještaja otkupa i prodaje stranih sredstava plaćanja.

Aplikacija je realizirana uz pomoć sljedećih tehnologija i programskih jezika:

- C#
- ASP. NET MVC
- Entity Framework
- Microsoft Report Viewer
- HTML, CSS, Bootstrap
- jQuery i dr.

U prvom dijelu rada detaljno su opisane specifikacije zahtjeva za izradu aplikacije za poslovanje jedne mjenjačnice. Drugi dio rada objašnjava tehnologije i pristup izradi aplikacije. U trećem dijelu je objašnjeno programsko rješenje te dijelovi koda koji su od velike važnosti za najbitnije funkcionalnosti aplikacije. Na kraju rada slijedi zaključak s osvrtom na cijelu izradu aplikacije, mogućnost poboljšanja te novih spoznaja do kojih je autor došao prilikom izrade.

2. UVOD U RAD MJENJAČNICE

U sljedećem poglavlju opisat će se kako funkcioniraju ovlaštene mjenjačnice te koji su osnovni pojmovi i procesi u radu ovlaštene mjenjačnice.

2.1. Opis rada

Osnovna zadaće mjenjačnice je omogućiti klijentima razmjenu valute u obliku otkupa ili prodaje. Mjenjačnice ostvaruju dobit prodajom valute po višem tečaju od onog po kojem kupuju istu valutu, ili uz izmjenu provizije (naknade) koju naplaćuju za obavljanje transakcije. Tečaj je omjer vrijednosti valute jedne države u odnosu na vrijednosti valute neke druge države. Tečaj se formira u skladu s trenutnim vrijednostima određenih valuta na svjetskom financijskom tržištu [1]. Ovlaštene mjenjačnice su partneri s bankama i od njih moraju preuzimati svakodnevne tečaje. U tom slučaju mjenjačnice određuju svoju dobit provizijom na određene valute ili transakcije.

2.2. Valute

Valuta je novčana jedinica neke države i služi kao zakonsko sredstvo plaćanja i razmjene. Svaka valuta je identificirana jedinstvenom troslovnom i brojčanom oznakom. Jedinstvena troslovna oznaka valute definirana je međunarodnim standardom ISO 4217 koji se koristi u bankarstvu i općenito u poslovanju. Prve dvije oznake predstavljaju naziv zemlje prema ISO 3166 standardu, dok treća oznaka u većini slučajeva predstavlja naziv valute [2].

Primjer: HRK → HR - Hrvatska; K – kuna

USD → US - Sjedinjene Američke države (*United States*); D – dolar

Osim troslovne oznake postoji i troznamenkasta brojčana oznaka za svaku valutu. Brojčana oznaka je često identična standardnoj brojčanoj oznaci države po ISO 3166-1 standardu [2].

Primjer: HRK → 191

USD → 840

2.3. Tečajne liste

Ovlaštene mjenjačnice dobivaju tečajne liste od ugovorne partnerske banke na način da banka svakodnevno šalje tekstualne datoteke s tečajima ili se mjenjačnice spajaju na web servis banke te tako razmjenjuju podatke. Tečajna lista sadrži sljedeće informacije: naziv zemlje kojoj valuta pripada, troslovna oznaka valuta, brojčana oznaka valute, jedinica (paritet), kupovni tečaj, prodajni tečaj i srednji tečaj.

Različite financijske transakcije koriste se različitim tečajevima. Pri prodaji deviza svojim klijentima, mjenjačnice koriste objavljeni prodajni tečaj, dok pri kupnji deviza od klijenata, koriste kupovni tečaj.

2.4. Osnovni procesi u radu mjenjačnice

Osnovna i svakodnevna radnja u poslovanju mjenjačnice je otkup i prodaja deviza. Za svaki oblik takve financijske transakcije potrebno je spremiti podatke o transakciji poput podataka o valuti koja se otkupljuje ili prodaje, tečaju, iznosu transakcije u domaćoj i stranoj valuti. Uz te podatke mjenjačnice bi trebale unijeti podatke o samom klijentu koji vrši transakciju. Unosi se je li osoba koja vrši transakciju domaći ili strani klijent, prezime i ime klijenta, adresa, država i broj dokumenta. Ovlašteni mjenjač dužan je unijeti serijske brojeve novčanica koje je klijent predao mjenjačnici zbog obaveza koje su propisane odredbama Zakona o sprječavanju pranja novca. Nakon što je transakcija potvrđena, ispisuju se tri potvrde od kojih jedna ide klijentu, jedna mjenjačnici i jedna banci koja je ugovorni partner s mjenjačnicom. Mjenjačnice svakodnevno izrađuju razne vrste izvještaja koje se krajem radnog dana predaju banci. Dnevnik otkupa ili prodaje je izvještaj koji prikazuje sve otkupe ili prodaje deviza u traženom vremenskog periodu pri čemu se dobije ukupan isplaćeni iznos u domaćoj valuti te iznos provizije. Izvještaj rekapitulacije otkupljenih ili prodanih deviza je sličan izvještaj kao dnevnik i on prikazuje sve transakcije u traženom vremenskom periodu ali su transakcije grupirane po broju tečajne liste i po valuti. Sve transakcije vezane su za trenutno pokrenutu blagajnu čije se stanje obračunava na kraju smjene ili radnog dana. Na početku radnog dana ili smjene unosi se početno stanje blagajne a na kraju djelatnik unosi iznos dotacijskih sredstava dobivenih pomoću rekapitulacijskog izvješća te tako zaključuje blagajnu i predaje otkupljene devize i pripadajuće izvještaje ugovornoj banci.

2.5. Zahtjevi prema aplikaciji

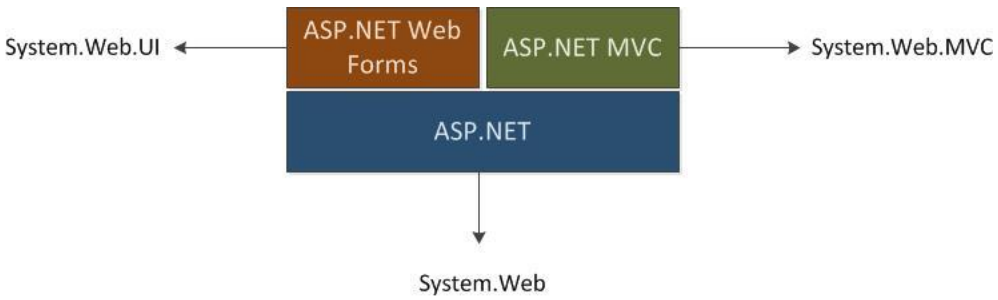
Zadatak ovog rada je napraviti web aplikaciju koja će omogućiti sve prije spomenute procese u poslovanju mjenjačnice. Aplikacija je namijenjena mjenjačnicama na prostoru Bosne i Hercegovine, što znači da je domaća valuta u kojoj se obavljaju sve transakcije - konvertibilna marka (KM). Budući da mjenjačnice mogu imati više djelatnika s različitim korisničkim ulogama, potrebno je implementirati, osim autentifikacije, i autorizacijski sustav koji će omogućiti određene akcije odgovarajućim korisnicima. Aplikacija mora omogućiti ručno učitavanje tečajnih datoteka ili dohvaćanje svakodnevnih tečajnih listi s web servisa ugovorne banke. Budući da izrađena aplikacija u fazi razvoja ne radi s ugovornom bankom, tečajne liste se dohvaćaju s web servisa Centralne Banke Bosne i Hercegovine. Aplikacija mora omogućiti unos svih potrebnih podataka prilikom obavljanja otkupa ili prodaje deviza. Osim obavljanja transakcija omogućeno je kreiranje raznih propisanih izvještaja spremnih za ispis.

3. KORIŠTENI ALATI I TEHNOLOGIJE

U ovom dijelu rada bit će se objašnjene najvažnije tehnologije i alati pri izradi aplikacije. Njihova konkretna implementacija kao i manje važni alati i biblioteke iz aplikacije prikazat će se u praktičnom dijelu rada.

3.1. ASP.NET

ASP.NET je Microsoftovo razvojno okruženje koje omogućuje izradu dinamičkih web stranica, interaktivnih web aplikacija i servisa na *.NET* platformi. Sve *ASP.NET* stranice se izvršavaju na poslužiteljskoj strani i tako generiraju traženi sadržaj klijentu. Budući da *ASP.NET* pripada *.NET* platformi, za razvoj aplikacije može se koristiti bilo koji programski jezik koji se izvodi na zajedničkoj jezičnoj okolini (engl. *common language runtime, CLR*) što omogućuje izvršavanje različitih programskih jezika na različitim arhitekturama. Najpoznatiji i najkorišteniji jezici su *C#, Visual C++* i *Visual Basic*. Na slici 3.1. prikazani su okviri u kojim se razvijaju web aplikacije na *.NET* platformi. *ASP.NET Web Forms* je razvojni okvir koji omogućava brzu izradu web aplikacija pomoću raznih komponenti grafičkog sučelja (kontrola), slično principu *Windows Forms* desktop aplikacija. Kontrole su objekti upravljani događajima (engl. *event-driven*) iz kojih se generira HTML sadržaj. *Web Forms* okvir omogućava odvajanje prezentacijskog dijela koda od logike aplikacija ali ne pruža dovoljnu kontrolu nad generiranjem HTML sadržaja koji se prezentira klijentu. Zbog poteškoća prilikom automatiziranog testiranja ovakvih aplikacija, ali i nedovoljne kontrole nad generiranim HTML sadržajem, današnje web aplikacije se sve više razvijaju u *ASP.NET MVC* okviru koji omogućava potpunu kontrolu i testiranje nad svim komponentama web aplikacije.



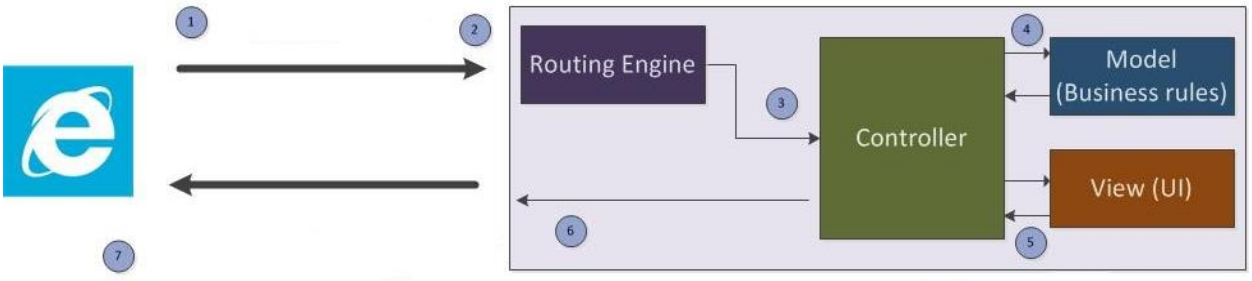
Sl. 3.1. Okviri za razvoj web aplikacija na .NET platformi [3]

3.2. MVC

ASP.NET MVC je tehnologija proizašla iz *ASP.NET-a* koja služi za izradu web aplikacija korištenjem model-pogled-kontroler (engl. *Model-View-Controller – MVC*) obrasca dizajna aplikacija. MVC obrazac se u računalnoj znanosti pojavio 1979. godine kada ga je osmislio Trygve Reenskaug. Izvorni naziv je bio *Thing-Model-View-Editor* te je kasnije pojednostavljen i preimenovan u *Model-View-Controller* ili skraćeno MVC [4].

Primarna prednost spomenutog obrasca je odvajanje dijelova aplikacije u zasebne komponente ovisno o njihovoj namjeni (engl. *separation of concerns within an application*) [4]. Komponente u MVC-u su:

- Model – sadrži ili predstavlja podatke s kojima se radi. Može biti jednostavni prikaz modela (engl. *view model*) koji samo predstavlja podatke koji se razmjenjuju između pogleda i kontrolera; ili može biti model domene (engl. *domain model*) koji sadrži podatke, operacije, transformacije i pravila za manipulacijom tih podataka [5].
- Kontroler (engl. *controller*) – sloj koji prihvaća korisničke zahtjeve i upravlja (posreduje) komunikacijom između modela i pogleda
- Pogled (engl. *view*) – sloj koji definira korisničko sučelje aplikacije, prezentacijski sloj)



Sl. 3.2. Prikaz tijeka rada MVC aplikacije prilikom pristupa korisnika [3]

Na slici 3.2. prikazani su koraci kako se generira sadržaj korisniku koji pristupa MVC aplikaciji:

1) Korisnik upisuje željenu adresu (URL) u preglednik i pristupa aplikaciji
2) Zahtjev dolazi do poslužitelja i prosljeđuje se sustavu za rutiranje (engl. *routing*) unutar aplikacije

3) Sustav za rutiranje na osnovu URL uzorka određuje odgovarajući kontroler za taj zahtjev
4) Ovisno o zahtjevu, kontroler komunicira s modelom kako bi dohvatio potrebne podatke iz baze podataka
5) Nakon što kontroler primi potrebne podatke, dohvaća odgovarajući pogled (engl. *View*) i puni ga podacima
6) Kontroler vraća pogled s podacima (najčešće u obliku HTML sadržaja a može i JSON ili XML)
7) Poslužitelj odgovara početnom zahtjevu tako što generira traženi sadržaj klijentu

U *ASP.NET MVC* okviru, kontroleri su C# klase koje obično nasljeđuju klasu `System.Web.Mvc.Controller`. Svaka javna (engl. *public*) metoda unutar takve klase je akcijska (engl. *action*) metoda koja je povezana s URL-om kroz sustav za rutiranje. Kada se pošalje zahtjev putem URL-a povezanog s akcijskom metodom, izvršavaju se operacije nad modelom domene i odabire se odgovarajući pogled koji će se prikazati klijentu [5].

3.2.1. Razor

ASP.NET MVC okvir koristi *Razor* mehanizam za pogled (engl. *view engine*) koji je odgovoran za prikaz sadržaja klijentu koji se šalje iz kontrolera i modela. *Razor* procesira sadržaj i čeka instrukcije, obično da dinamički ubaci sadržaj koji će se prezentirati klijentu [5]. Na ovaj način se poslužiteljski kôd (engl. *server code*) ubacuje u statični sadržaj stranice kojeg čini HTML, CSS, JavaScript i običan tekst.

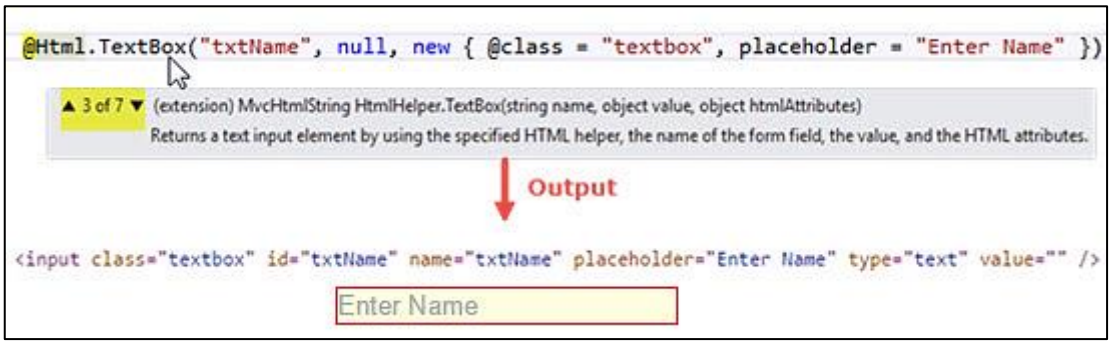
ASP.NET web stranice s *Razor* sintaksom imaju posebnu ekstenziju: *.cshtml*. Na osnovu te ekstenzije poslužitelj prepoznaje da se na stranici koristi *Razor* te prvo izvršava njegove naredbe [6]. *Razor* sintaksa bazirana je na *C#* programskog jeziku i unutar web stranica započinje sa znakom `@`.

Na slici 3.3. je primjer kako se dohvaćaju metode i članovi modela kroz `@Model` objekt. Uvodna naredba `@model` definira tip objekta koji je proslijeđen pogledu od strane akcijske metode. *Razor* mehanizam će se detaljnije u radu prikazati u poglavljima vezanim za izradu aplikacije.

```
@model Razor.Models.Product

@{
    Layout = null;
}

<!DOCTYPE html>

<html>
<head>
    <meta name="viewport" content="width=device-width" />
    <title>Index</title>
</head>
<body>
    <div>
        @Model.Name
    </div>
</body>
</html>
```

Sl. 3.3. Primjer Razor koda [6]

Razor koristi HTML pomoćne elemente(engl. *HTML Helpers*) za generiranje dinamičkog sadržaja u pogledima. *HTML Helpers* je skup pomoćnih metoda koje generiraju niz znakova(engl. *string*). Taj niz znakova može biti običan tekst ili neki od HTML elemenata poput paragrafa, polja za unos teksta, liste, forme i mnogih drugih. *HTML Helper* metode nije nužno koristiti pri izradi *ASP.NET MVC* aplikacije ali one znatno pomažu bržem razvoju aplikacije. Na slici 3.4. prikazana je jednostavna forma za unos teksta napisana pomoću *HTML Helper* metode i kako ista forma izgleda u *HTML-u* kada se prezentira klijentu.



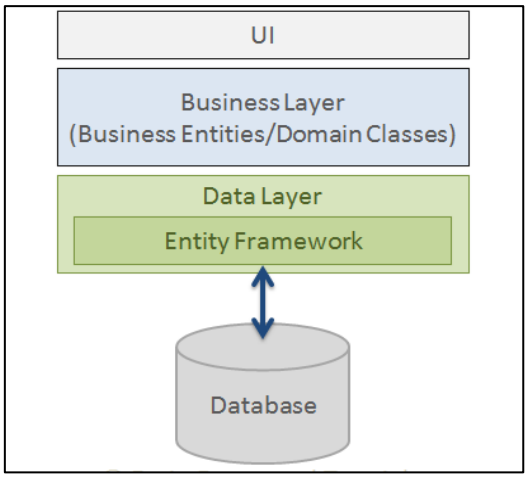
Sl. 3.4. Forma za unos teksta napisana koristeći HTML Helper metodu [6]

Na ovom primjeru je vidljivo da korištene metode *HTML Helpera* rezultiraju standardnim HTML kodom što omogućava *front-end* dizajnerima neovisan rad s obzirom na korištenje *Razora*. Još jedna velika prednost *Razor* mehanizma za poglede je što unutar *Visual Studio* razvojnog okruženja, *IntelliSense*[1] alat ima potpunu podršku za *Razor* sintaksu.

[1] *IntelliSense* je Microsoftov alat za pametno prepoznavanje i dovršavanje koda ugrađen u Visual Studio razvojno okruženje

3.3. Entity Framework

Entity Framework (EF) je objektno – relacijski razvojni okvir razvijen u *ADO.NET–u* (engl. *ActiveX Data Objects .NET*)[2] [7]. Razvio ga je Microsoft i sadržava skup tehnologija koje omogućavaju razvoj aplikacija koje su pretežito orijentirane podacima. Glavna karakteristika ovog okvira je objektno relacijsko preslikavanje (engl. *Object-relational mapping – ORM*) – tehnika za transformaciju podataka iz relacijske baze podataka u objekte u određenom programskom jeziku[8]. Omogućava da se tablice, procedure i pogledi iz relacijske baze podataka predstave kao objekti u određenom objektno orijentiranom programskom jeziku. Rezultat toga je da programeri ne moraju pisati tradicionalne SQL upite nad bazom podataka, nego mogu koristiti metode i svojstva nad spomenutim objektima. Na slici 3.5. prikazano je kojoj razini pripada *Entity Framework* unutar aplikacije. Nalazi se između poslovnog sloja i baze podataka i sprema podatke iz poslovnih klasa ili entiteta u bazu podataka i obratno.



Sl. 3.5. Prikaz Entity Frameworka unutar arhitekture aplikacije [9]

Entitet je klasa u domeni aplikacije koja je uključena kao svojstvo tipa `DbSet<TEntity>` u izvedenoj kontekstnoj klasi. Kontekstna klasa se koristi za upite i spremanje podataka u bazu. To je najvažnija klasa u *Entity Frameworku* i u njoj se nalazi konfiguracija klasa domene, mapiranja, prijenosa podataka i ostalo. *Entity Framework* mapira svaki entitet u tablicu i svako svojstvo entiteta u stupac tablice u bazi podataka [9].

[2] ADO.NET je skup komponenti koji omogućava pristup i manipulaciju podacima u relacijskim bazama podataka. ADO.NET pripada osnovnoj klasi biblioteke Microsoft .NET Framework-a.

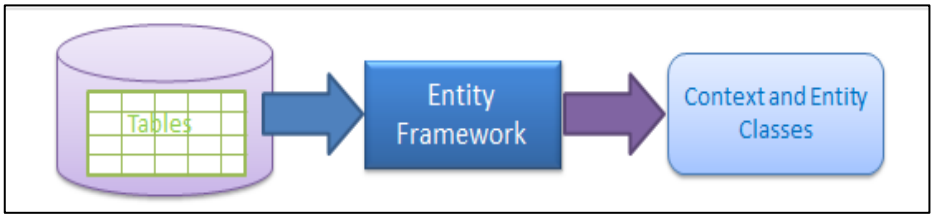
Dvije glavne prednosti korištenja *Entity Frameworka* su:

1. Veliki dio koda se generira automatski i nema potrebe za pisanjem većinskog programskog koda za pristup podacima. Time se povećava produktivnost programera i smanjuje se vrijeme razvoja budući da se fokus može prebaciti na bitnije dijelove aplikacije
2. Pojednostavljivanje upita - jedinstveni jezik za upite (*LINQ*) olakšava i ubrzava pisanje upita nad podacima, neovisno iz kojeg su izvora

Kako bi se aplikacija realizirala pomoću *Entity Framework* okvira, potrebno je prvo definirati pristup razvoja. Postoje tri pristupa razvoja pomoću *Entity Frameworka*:

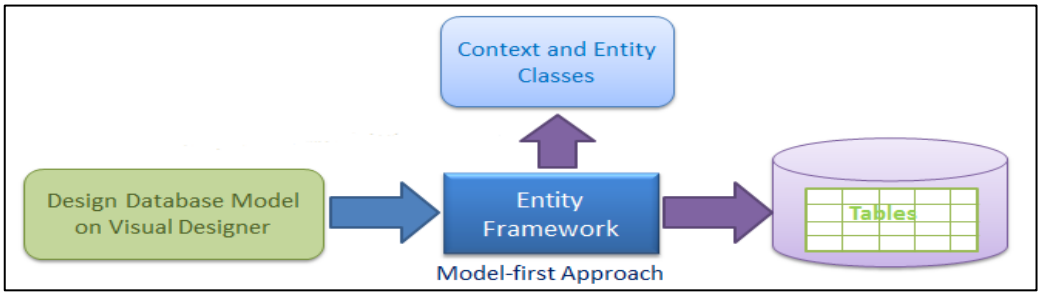
- Baza - prvo (eng. Database-First)
- Model - prvo (eng. Model-First)
- Kôd - prvo (eng. Code-First)

Baza – prvo je pristup u kojem se kontekst i entiteti generiraju iz već postoje baze podataka koristeći vizualni EDM[3] čarobnjak koji je integriran u *Visual Studio*. Preko čarobnjaka je potrebno samo napraviti konekciju na postojeću bazu i zatim odabrati tablice, procedure ili poglede koji se žele prebaciti u model.



Sl. 3.6. Database-First pristup [9]

Model – prvo je pristup koji omogućava kreiranje modela podataka u vizualnom dizajneru iz kojeg se zatim generiraju entiteti, kontekstna klasa i baza podataka. Ovaj pristup je koristan ako nema već postojeće baze podataka a preferira se vizualni dizajner za modeliranje baze.

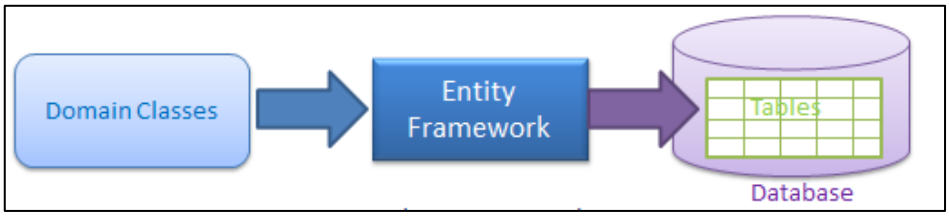


Sl. 3.7. Model-First pristup [9]

[3] EDM (engl. *Entity Data Model*) predstavlja konceptualni model, model između baze podataka i domene

3.3.1. Code-First

Kôd – prvo pristup se koristi kada ne postoji gotova baza za aplikaciju. Za razliku od ostalih pristupa, kod ovoga se pristupa prvo kreiraju *POCO* (engl. *Plain old CLR objects*) klase i kontekstna klasa, iz kojih se zatim koristeći migracije, kreira baza podataka. *POCO* klase sadržavaju atribute koji odgovaraju stupcima tablica, odgovarajuće *get* i *set* metode za pristup tim atributima i odgovarajuće konstruktore. Primjer *POCO* klase je prikazan na slici 3.9.



Sl. 3.8. Code–First pristup [9]

```
public class Student
{
    public int StudentID { get; set; }
    public string StudentName { get; set; }
    public DateTime? DateOfBirth { get; set; }
}
```

Sl. 3.9. Klasa Student koja predstavlja klasu domene

Na slici 3.9. prikazana je klasa `Student` koja predstavlja klasu domene unutar aplikacije. Spomenuta klasa postaje entitet tek kada se uključi kao svojstvo tipa `DbSet<TEntity>` kontekstne klase `SchoolContext` prikazanoj na slici 3.10.

```
public class SchoolContext : DbContext
{
    public SchoolContext()
    {
    }
    public DbSet<Student> Students { get; set; }
}
```

Sl. 3.10. Kontekstna klasa

Na ovaj način se, prilikom mapiranja, od klase `Student` stvara tablica u bazi podataka pod imenom `Students` i sadrži stupce: `StudentID`, `StudentName` i `DateOfBirth` s pripadajućim tipovima.

Baza podataka koju će *Entity Framework* stvoriti prilikom ovog pristupa ovisi o baznom konstruktoru kontekstne klase i parametrima koje prima. Ukoliko bazni konstruktor nema parametara EF će stvoriti bazu podataka na lokalnom *SQLEXPRESS* poslužitelju. Konstruktoru se mogu predati parametri poput imena baze ili konekcijskog stringa, i na osnovu njih EF će kreirati bazu podataka.

Budući da se model baze podataka često mijenja prilikom izrade aplikacije, neophodna je sinkronizacija modela s bazom. *Entity Framework* koristi migracije, mehanizam koji omogućava da se shema baze podataka automatski ažurira ukoliko se promjeni model, bez gubitka postojećih podataka [9]. Migracije se mogu konfigurirati da budu automatske ili temeljene na kodu. Ukoliko nisu automatske, unutar *Package Manager Console* je potrebno:

- Omogućiti *code-based* migracije naredbom `Enable-Migrations`
- Kreirati migraciju naredbom npr. `Add-Migration CreatedStudentTable`
- Ažurirati bazu podataka naredbom `Update-Database`

Ovim naredbama se kreira klasa s nazivom migracije i sadrži metode `Up()` i `Down()` koje sadrže kôd za stvaranje i brisanje objekata u bazi podataka.

Migracije pružaju mogućnost vraćanja baze na prethodno stanje ukoliko se za to pojavi potreba. Potrebno je upisati naredbu `Update-Database -TargetMigration` i naziv prethodne migracije. Prednost kôd – prvo pristupa i migracija je što olakšava istovremeni rad više programera koji mogu pratiti promjene nad shemom baze podataka.

3.3.2. LINQ upiti

Upit je izraz koji preuzima podatke iz nekog izvora podataka. Upiti se obično izražavaju posebnim jezikom upita. Kod relacijskih baza podataka obično se koristi SQL (engl. *Structured Query Language*). U sklopu *Entity Frameworka* koristi se programski jezik integriranih upita - LINQ (engl. *Language-Integrated Query*). LINQ služi kao dodatak *C#* programskom jeziku i nudi jednostavan i konzistentan način rada s kolekcijama podataka iz raznih izvora. Pomoću LINQ-a se može pretražiti svaka kolekcija u memoriji koja implementira `IEnumerable<T>` sučelje (engl. *interface*). LINQ omogućuje i pretraživanje kolekcija koje se dinamički popunjavaju iz udaljenog izvora podataka, kao što je npr. *SQL Server*. Takve kolekcije dodatno implementiraju `IQueryable<T>` sučelje [10].

LINQ upiti se mogu pisati koristeći:

- metodnu sintaksu (engl. *method syntax*)
- sintaksu izraza upita (engl. *query syntax*)

Metodna sintaksa koristi se metodama za pozivanje LINQ operatora. Većina operatora prima *lambda* izraz kao argument koji olakšava razumijevanje i formiranje upita. Ulazni argument *lambda* izraza odgovara jednom ulaznom elementu kolekcije nad kojom se vrši upit. Metodna sintaksa je fleksibilna i omogućuje ulančavanje više operatora kako bi se formirali složeniji upiti.

Sintaksa izraza upita deklarativno poziva LINQ operatore i nalikuje SQL sintaksi. Ovi upiti uvijek počinju izrazom `from` a završavaju izrazom `select` ili `group` [10]. Primjer LINQ upita u obje sintakse prikazan je na slici 3.10. gdje će oba upita vratiti isti rezultat.

```
string[] studenti = {"Dario","Marko","Vedran","Josip"};

//metodna sintaksa
IEnumerable<string> upit1 = studenti.Where(s => s.Contains("o"))
                                    .OrderBy(s => s.Length)
                                    .Select(s => s.ToUpper());
//sintaksa izraza upita
IEnumerable<string> upit2 = from s in studenti
                            where s.Contains("o")
                            orderby s.Length
                            select s.ToUpper();
```

Sl. 3.10. Primjer metodne sintakse i sintakse izraza upite kod LINQ upita

Važno je napomenuti da se oba načina pisanja LINQ upita mogu međusobno kombinirati.

Neke od prednosti korištenja LINQ izraza u odnosu na tradicionalne SQL upite su:

- znatno smanjuje količinu koda
- pruža bolje razumijevanje napisanog koda
- omogućuje kreiranja upita nad podacima iz različitih izvora

3.4. ASP.NET Identity

ASP.NET Identity je sustav članstva unutar *ASP.NET* okvira koji omogućava jednostavnu implementaciju autentifikacije i autorizacije korisnika. Sustav kreira sve potrebne tablice za korisnike i njihove uloge (engl. *roles*) te stvara kontrolere, modele i poglede koji omogućavaju korištenje registracije i prijave korisnika u aplikaciji.

Osim standardnog registriranja i prijave pomoću korisničkog imena i zaporke, *Identity* nudi mogućnost prijave pomoću računa s društvenih mreža poput *Facebooka, Twittera,* i drugih [11]. *ASP.NET Identity* koristi se *Entity Framework* okvirom i *Code-First* pristupom kod kreiranja potrebnih modela u bazu podataka. Korisnik na jednostavan način može promijeniti početne postavke *Identity* sustava, mijenjajući nazive tablica ili uvjeta potrebnih za prijavu i registraciju. Primjer mijenjanja ovih postavki prikazat će se u poglavljima vezanim za izradu aplikacije.

3.5. Microsoft ReportViewer

Microsoft ReportViewer je mehanizam za pregledavanje i kreiranje izvještaja (engl. *Reports*). *ReportViewer* je kod *ASP.NET Web Forms* aplikacija sastavna kontrola dok kod *ASP.NET MVC-a* nema potpunu podršku. Kako bi se *ReportViewer* mogao koristiti u *ASP.NET MVC* aplikacijama potrebno je dodati odgovarajuće `Microsoft.ReportViewer` reference u projekt. Izvještaji se mogu kreirati grafički pomoću dizajnera izvještaja (engl. *Report Designer*) ili uređivanjem XML koda. Kreirani izvještaji imaju *.rdlc* ekstenziju.

Izvještaji su usko vezani s podacima koji dolaze iz seta podataka (engl. *DataSet*) ili kolekcijama objekata tipa `IEnumerable` [12]. Set podataka može biti bilo koja tablica iz baze podataka a kolekcija objekata se može poslati iz kontrolera unutar MVC aplikacije. Ukoliko se naknadno promjeni shema seta podataka nakon što je izvještaj već definiran, potrebno je ažurirati cijeli izvještaj [12]. U poglavljima vezanim za izradu rada detaljnije će biti prikazano kreiranje izvještaja i seta podataka s kojim izvještaj radi.

4. IZRADA APLIKACIJE

U ovom dijelu rada bit će objašnjena izrada i struktura aplikacije. Prikazat će se projektna struktura, baza podataka, izrada korisničkog sučelja i važniji dijelovi koda koji omogućavaju najvažnije funkcionalnosti aplikacije.

4.1. Projektna struktura

Pri stvaranju nove aplikacije u *ASP.NET MVC* okviru, automatski se stvara projekt s određenim datotekama i direktorijima. Kreirani direktoriji odgovaraju pravilima korištenja MVC obrasca po kojim se svi kontroleri nalaze u direktoriju `Controllers`, svi modeli su u `Models` direktoriju a svi pogledi se nalazu u poddirektoriju koji sadrži ime odgovarajućeg kontrolera unutar `Views` direktorija. U tablici 4.1. prikazana je struktura aplikacije s predodređenim i dodanim direktorijima.

Tab. 4.1. Popis direktorija u projektu i kratki opis sadržaja

Direktorij	Sadržaj
`/App_Data`	JSON datoteka s popisom standardnih valuta, baza podataka s *.mdf* ekstenzijom.
`/App_Start/`	Konfiguracijske klase za sustave: rutiranja, autentifikacije, pakete skripti i stilova i dr.
`/Content`	CSS datoteke, slike, i određene *.js* biblioteke
`/Controllers`	Kontroleri
`/fonts`	Fontovi
`/Helpers`	Pomoćne klase za dohvaćanje i učitavanje tečaja, učitavanje standardnih valuta, klase za pretvorbu brojeva u riječi itd.
`/Migrations`	Migracijske datoteke stvorene pomoću *Code-First* pristupa
`/Models`	Modeli za autentifikaciju, entiteti, modeli za poglede i klase koje proširuje postojeće entitete
`/Reports`	Izvještaji s *.rdlc* ekstenzijom
`/Scripts`	JavaScript skripte, Bootstrap i dr.
`/Views`	Pogledi raspoređeni po poddirektorijima koji nose naziv pripadajućeg kontrolera

4.2. Kreiranje baze podataka

Baza podataka korištena u aplikacija je stvorena pomoću *Code-First* pristupa unutar *Entity Framework* okvira, što znači da se sve tablice u bazi podataka generiraju iz entiteta – klasa domene. Fizički model baze podataka smješten je na *LocalDB* inačici *SQL Server Express*[4] poslužitelja. Baza podataka se nalazi u `App_Data` direktoriju i sadrži *.mdf* ekstenziju. Kako bi aplikacija koristila spomenutu bazu podataka potrebno je dodati kôd za spajanje (engl. *connection string*) u datoteku `Web.config`:

```
<add name="ApplicationDbContext" connectionString="Data Source=.\SQLEXPRESS;Integrated
Security=SSPI;AttachDBFilename=|DataDirectory|in2exchangeDb.mdf;UserInstance=True"
providerName="System.Data.SqlClient" />
```

Način na koji *Entity Framework* iz entiteta kreira tablice objašnjen je u poglavlju 3.3. Kontekstna klasa iz koje nastaju sve tablice prikazana je na slici 4.1.

```
namespace in2exchange.DAL
{
    public class ApplicationDbContext : IdentityDbContext<ApplicationUser>
    {
        public ApplicationDbContext()
            : base("in2exchangeDb", throwIfV1Schema: false)
        {
        }

        public DbSet<Promet> Prometi { get; set; }
        public DbSet<Valuta> Valute { get; set; }
        public DbSet<Vlasnik> Vlasnici { get; set; }
        public DbSet<Tecaj> Tecaji { get; set; }
        public DbSet<ZaglavljeTecaja> ZaglavljaTecaja { get; set; }
        public DbSet<Blagajna> Blagajne { get; set; }
        public DbSet<Specifikacija> Specifikacije { get; set; }
        public DbSet<ZaglavljeSpecifikacije> ZaglavljaSpecifikacija { get; set; }


        public static ApplicationDbContext Create()
        {
            return new ApplicationDbContext();
        }
    }
}
```

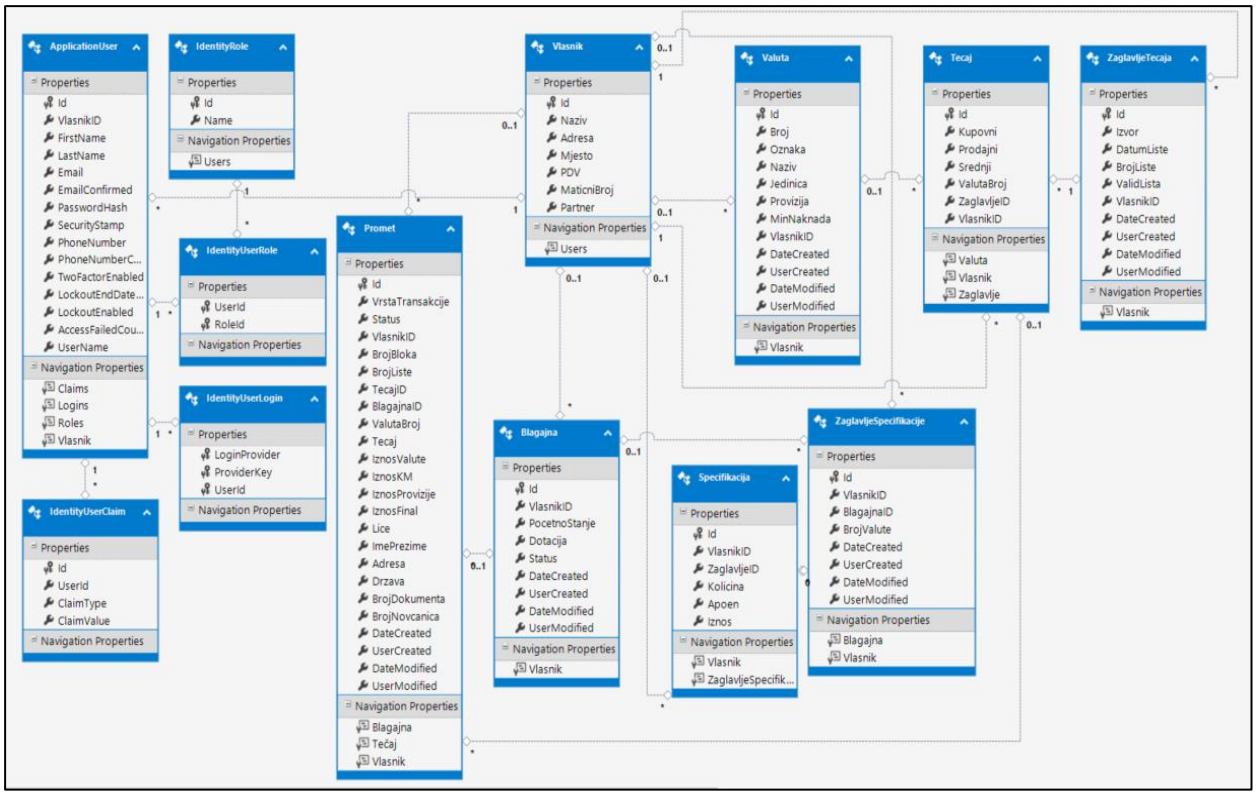
Sl. 4.1. Kontekstna klasa iz koje nastaju sve tablice pomoću Code-First pristupa

[4] LocalDb SQL Server Express je posebna i jednostavnija inačica SQL Servera za programere koja omogućava upravljanje relacijskim bazama podataka i koja je besplatna za preuzimanje.

U bazi podataka se nalaze i tablice generirane pomoću *ASP.NET Identity* sustava koje omogućavaju autentifikaciju i autorizaciju korisnika. U tablici 4.2. prikazan je popis svih tablica u bazi podataka i njihov opis.

Tablica 4.2. Popis tablica u bazi podataka i kratki opis

Naziv tablice	Opis
`AspNetUsers`	Sadrži korisniče podatke: korisničko ime, email, zaporku, ime, prezime i dr.
`AspNetRoles`	Sadrži popis svih korisničkih uloga u aplikacija
`AspNetUserRoles`	Povezuje korisničke uloge s korisnicima
`AspNetUserLogins`	Sadrži podatke o prijavi korisnika
`AspNetUserClaims`	Sadrži podatke o zahtjevima korisnika
`Vlasnici`	Sadrži podatke o svakoj pojedinoj mjenjačnici koja koristi aplikaciju: naziv, adresa, mjesto, banka partner itd.
`Valute`	Podaci o valuti: broj, oznaka, naziv, jedinica, minimalna naknada i dr.
`Tecaji`	Podaci o tečaju: broj valute za koju važi tečaj, kupovni, prodajni, broj zaglavlja liste i dr.
`ZaglavljaTecaja`	Podaci o tečajnoj listi: izvor, datum liste, broj liste dr.
`Blagajne`	Podaci o blagajni kojoj mjenjačnici pripada: početno stanje, dotacijska sredstva, status i dr.
`Prometi`	Podaci o transakcijama: datum kreiranja, valuta, iznosi, vrsta transakcije, podaci o klijentu, broj novčanica i dr.
`Specifikacije`	Podaci o specifikaciji novčanica stranih valuta: broj valute, količina, apoeni, broj zaglavlja kojem pripada i dr.
`ZaglavljaSpecifikacija`	Podaci o zaglavlju specifikacije: datum kreiranja, broj blagajne za koju vrijedi i dr.
`_MigrationHistory`	Popis svih migracija, omogućuje vraćanje baze podataka na prethodno stanje



Sl. 4.2. Prikaz modela koji predstavljaju tablice i veze između njih

Na slici 4.2. su prikazani entiteti – modeli iz kojih se stvaraju identične tablice u bazi podataka. Najvažnija tablica u bazi podataka je tablica `Vlasnici` na koju su povezane skoro sve ostale tablice. Sa svakom od povezanih tablica, tablica `Vlasnici` povezana je u omjeru `0..1:*` ili `1:*` (engl. *one-to-many*) što pokazuje da jedna mjenjačnica (vlasnik) može posjedovati jednu ili više instanci povezanih tablica. Razlog važnosti tablice `Vlasnici` je tome je što aplikaciju koriste korisnici(engl. *Application user*) od kojih svaki pripada jednog grupi tj. vlasniku koji predstavlja jednu mjenjačnicu. Aplikacija mora prikazati samo podatke koji pripadaju odgovarajućoj mjenjačnici u kojoj djelatnik (korisnik) radi.

4.3. Autentifikacija i autorizacija korisnika

4.3.1. Kreiranje novih korisnika i prijava

Zbog osjetljivosti podataka s kojim aplikacija radi, nužno je zaštiti podatke od neovlaštenog pristupa. Autentifikacija korisnika u aplikaciji implementirana je pomoću *ASP.NET Identity* sustava opisanog u poglavlju 3.4. Spomenuti sustav sam generira potrebne tablice, kontrolere, modele i poglede. Iako spomenuti sustav nudi već gotovo rješenje za registraciju i prijavu, važno je napomenuti da razvijena aplikacija ne koristi sustav registracije. Razlog tome je što postoji korisnik s administratorskom ulogom koji ima mogućnost stvaranja novih mjenjačnica (vlasnika) koje će koristiti aplikaciju. Na ovaj način administrator je pružatelj usluga aplikacije i on omogućuje mjenjačnicama pristup aplikaciji.

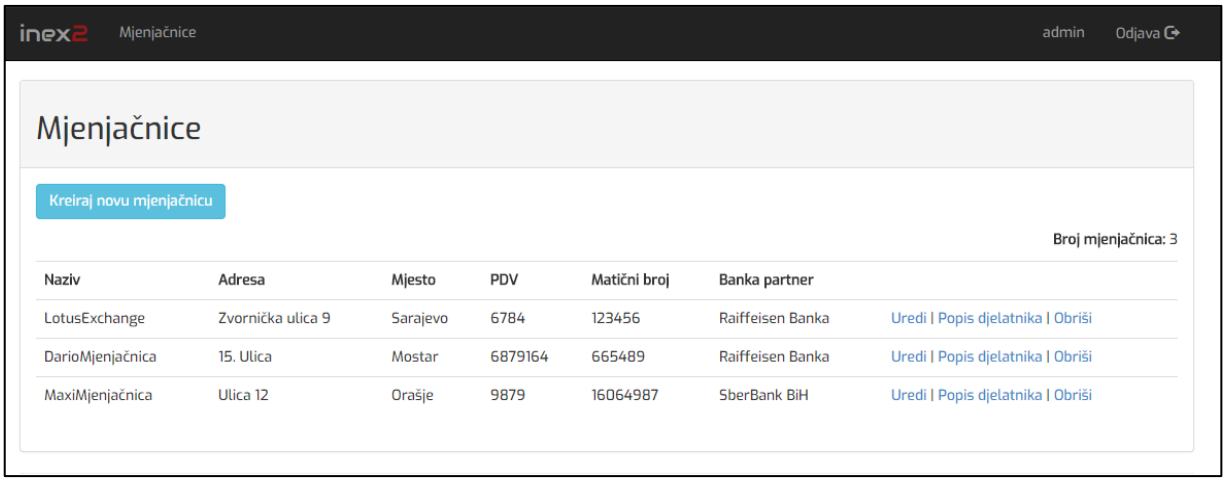
Pri prvom pokretanju aplikacije, ukoliko već spomenuta uloga ne postoji u bazi podataka, kreira se korisnik i dodjeljuje mu se administratorska uloga (Sl. 4.3.). Kôd za spomenutu akciju registrira se unutar funkcije `Application_Start()` u `Global.asax` datoteci koja obrađuje sve događaje (engl. *events*) na razini aplikaciji(engl. *application-level*) i trenutne sesije (engl. *session-level*).

```
var RoleManager = new RoleManager<IdentityRole>(new RoleStore<IdentityRole>(context));
var UserManager = new UserManager<ApplicationUser>(new UserStore<ApplicationUser>(context));
// Kreiranje administratorske uloge i početnog admin korisnika
if (!RoleManager.RoleExists("Admin"))
{
    // kreiranje uloge
    var role = new IdentityRole
    {
        Name = "Admin"
    };
    RoleManager.Create(role);
    //kreiranje korisnika administratora
    var adminUser = new ApplicationUser
    {
        UserName = "admin",
        Email = "admin@in2exchange.com",
        VlasnikID = 1002,
    };
    string adminPassword = "admin123";
    IdentityResult newUser = UserManager.Create(adminUser, adminPassword);
    //Dodjeljivanje uloge "Admin" novokreiranom korisniku - administratoru
    if (newUser.Succeeded)
    {
        var result1 = UserManager.AddToRole(adminUser.Id, "Admin");
    }
}
```

Sl. 4.3. Kôd za kreiranje administratorskog korisnika i uloge

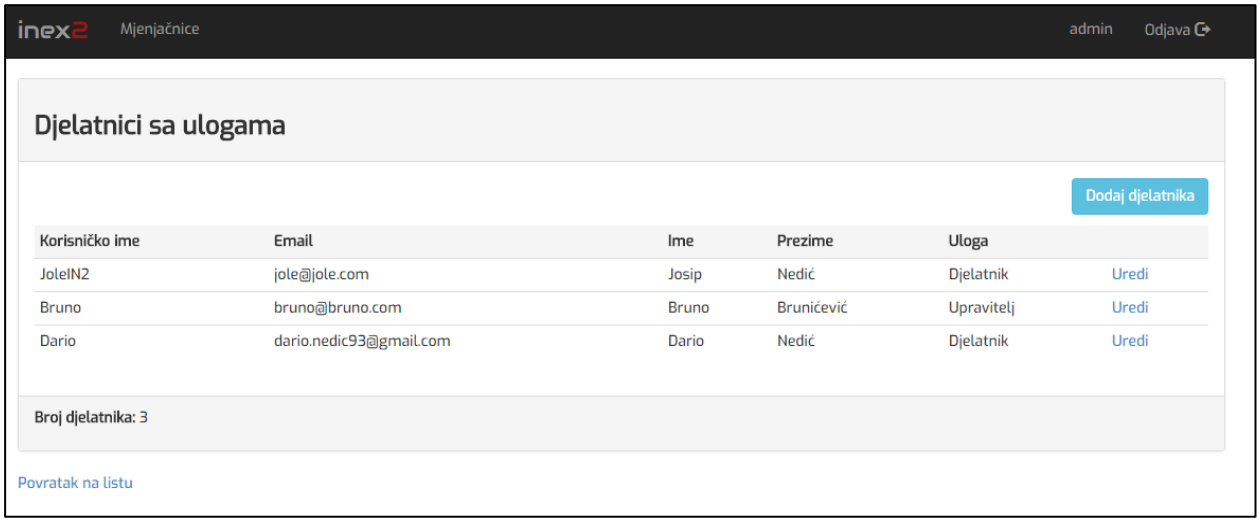
Osim uloge administratora, postoji uloga upravitelja mjenjačnice i uloga djelatnika a te korisnike kreira administrator. Spomenute uloge i moguće akcije koje one omogućuju pojasnit će se u sljedećem podpoglavlju.

Administrator se prijavljuje predodređenim korisničkim imenom i zaporkom, nakon čega mu se prikazuje administratorsko sučelje aplikacije, kao što je prikazano na slici 4.4.



Sl. 4.4. Prikaz administratorskog sučelja

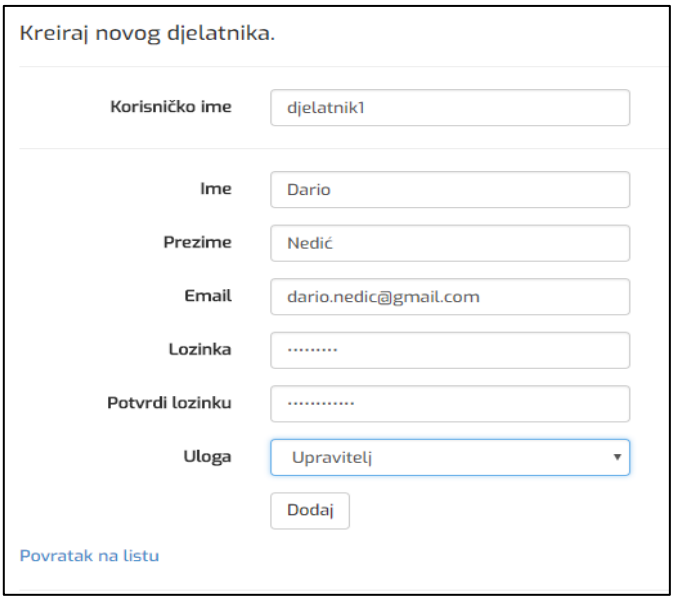
Nakon prijave, administrator može obavljati sve CRUD[5] operacije nad mjenjačnicama. Nakon što je kreirana nova mjenjačnica potrebno je dodati njezine djelatnike putem sučelja prikazanog na slici 4.5. i slici 4.6. koje se otvara klikom na poveznicu *Popis djelatnika*



Sl. 4.5. Administratorsko sučelje za pregled djelatnika unutar mjenjačnice

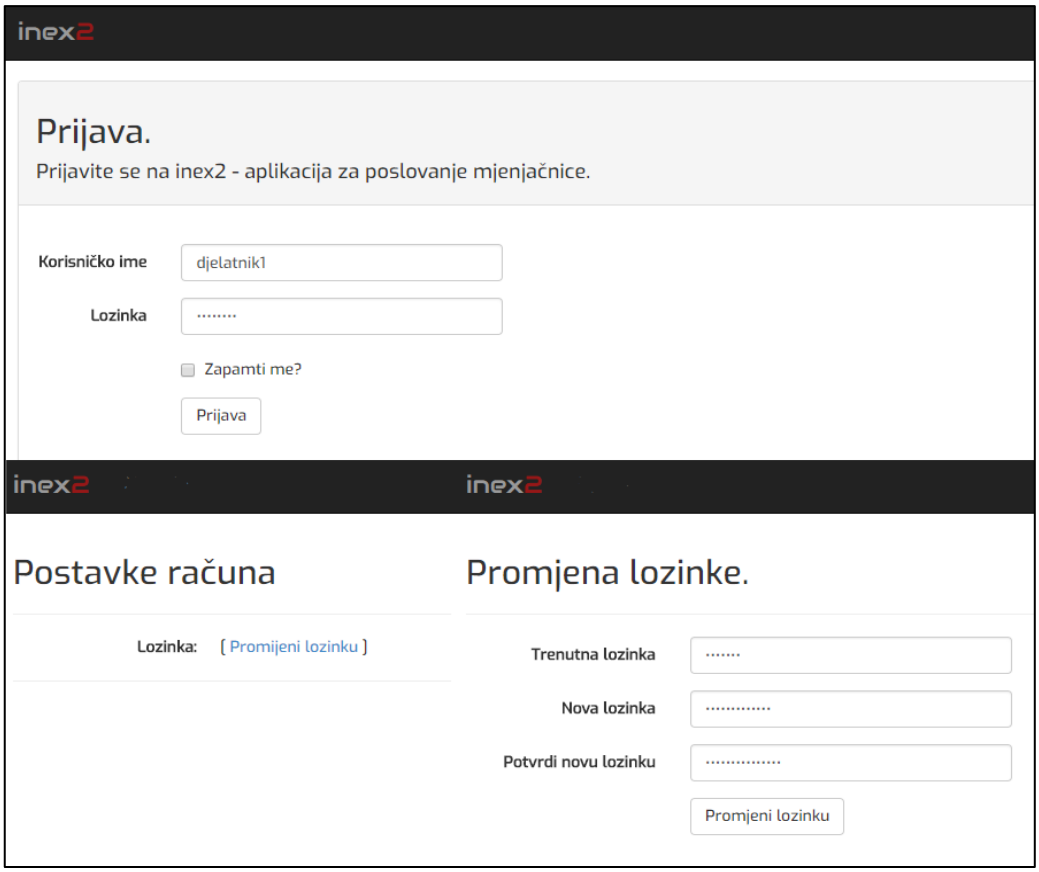
[5] CRUD (engl. Create-Read-Update-Delete) – kratica koja označava operacije kreiranja, čitanja, izmjenjivanja i brisanja podataka

Klikom na gumb *Dodaj djelatnika* otvara se novi pogled za kreiranje novog djelatnika (Sl. 4.6.). Obavezan je unos korisničkog imena i zaporke dok su ostala polja neobavezna.



Sl. 4.6. Pogled za kreiranje novog djelatnika

Nakon što administrator kreira novog djelatnika, istom djelatniku je omogućen pristup aplikaciji pomoću kreiranog korisničkog imena i početne zaporke. Važno je napomenuti da se novom djelatniku nakon prijave u aplikaciju, nudi mogućnost promjene postojeće zaporke pomoću spomenutog *ASP.NET Identity* sustava (Sl. 4.7.).



Sl. 4.7. Sučelje za prijavu korisnika i promjenu zaporke

4.3.2. Ograničavanje pristupa

Pri kreiranju novog djelatnika (Sl. 4.6.) administrator za njega može odabrati ulogu običnog djelatnika ili ulogu upravitelja mjenjačnice. Razlika u ulogama je što upravitelj ima veće ovlasti i može generirati drugačije izvještaje od djelatnika. Upravitelj može kreirati nove djelatnike, jednako kao i administrator cijelog sustava, ali samo u sklopu svoje mjenjačnice.

Ograničavanje pristupa unutar MVC obrasca se implementira koristeći atribut `[Authorize]` kojim se označava kontroler čije akcije i poglede treba zaštititi od neovlaštenog pristupa. Spomenuti atribut može biti bez dodatnih parametara ili može sadržavati parametre za ulogu ili korisničko ime. Ukoliko nema parametara, to znači da je pristup kontroleru, i svemu što on sadrži, dozvoljen samo prijavljenim korisnicima. Na slici 4.8. prikazan je način ograničavanja pristupa po ulozi gdje kontroleru i pogledima mogu pristupiti samo upravitelji.

```
[Authorize(Roles = "Manager")]
public class VlasnikController : BaseController
{
    ApplicationDbContext db = new ApplicationDbContext();
    // GET: PopisDjelatnika
    public ActionResult Index()
    {
        return View();
    }
```

Sl. 4.8. Dozvoljen pristup kontroleru samo korisniku s ulogom Manager

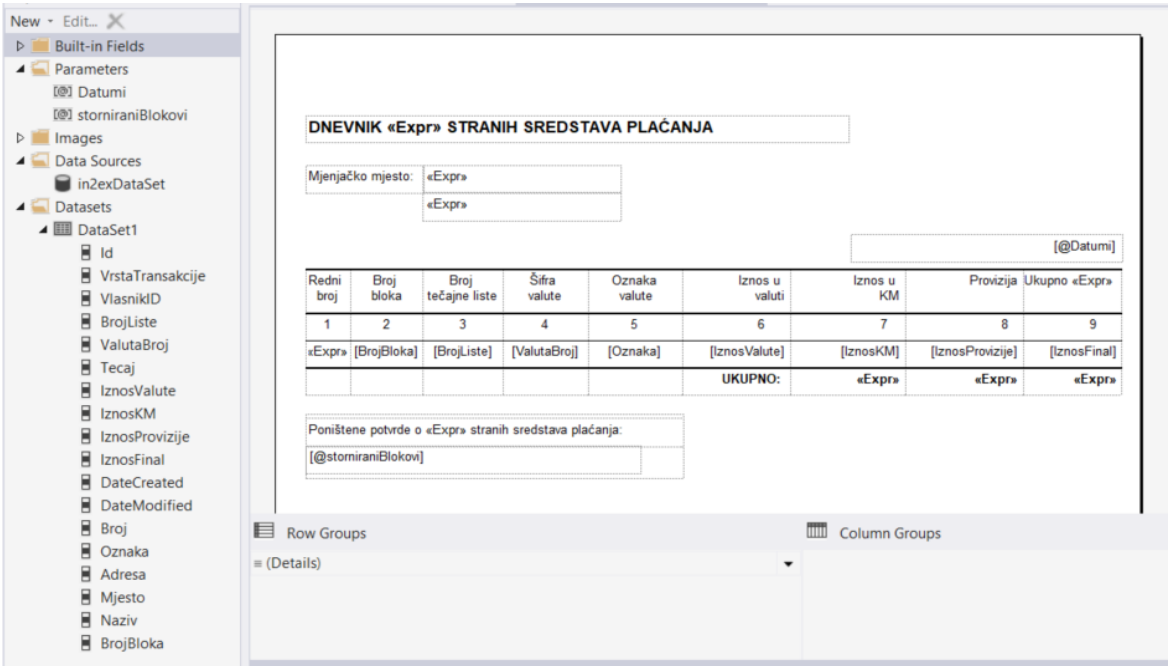
S obzirom da se i pogledi mijenjaju ovisno o korisničkoj ulozi, u pogledima se ispituje autentifikacija i autorizacija korisnika pomoću Razor sintakse i uvjetnih naredbi. Na slici 4.9. prikazan je kôd za navigacijsku traku na kojoj se pokazuje poveznica za popis djelatnika samo korisničkog ulozi upravitelja.

```
<div class="navbar-collapse collapse">
    <ul class="nav navbar-nav">
        @if (Request.IsAuthenticated)
        {
            if (User.IsInRole("Admin"))
            {
                <li>@Html.ActionLink("Mjenjačnice", "Index", "Admin")</li>
            }

            else
            {
                <li>@Html.ActionLink("Transakcije", "Index", "Promet")</li>
                <li>@Html.ActionLink("Tečajne liste", "Index", "Zaglavlje")</li>
                <li>@Html.ActionLink("Valute", "Index", "Valuta")</li>
                <li>@Html.ActionLink("Blagajna", "Index", "Blagajna")</li>
                if (User.IsInRole("Manager"))
                {
                    <li>@Html.ActionLink("Djelatnici", "Index", "Vlasnik")</li>
                }
            }
        }
    </ul>
    @Html.Partial("_LoginPartial")
</div>
```

Sl. 4.9. Kôd za navigacijsku traku

4.4. Izrada izvještaja

Izvještaji se u aplikaciji izrađuju pomoću grafičkog dizajnera (engl. *Report Designer*) unutar *Visual Studio* razvojnog okruženja (Sl. 4.10.). On omogućuje ubacivanje raznih grafičkih elemenata u izvještaj poput tekstualnih okvira, tablica, linija, grafova, lista, Sl. i dr.



Sl. 4.10. Prikaz grafičkog dizajnera za izvještaje

Izvještaj prikazuje podatke koje prima u obliku seta podataka koji predstavlja podskup baze podataka i temelji se na upitu koji se izvodi na jednoj ili više tablica baze podataka. Spomenuti set podataka se puni podacima iz kontrolera koji su obično rezultat nekog upita. Osim seta podataka, izvještaj može prikazati i pojedinačne podatke u obliku parametara koji se također šalju iz kontrolera. Primjer koda koji kreira lokalni izvještaj i dodjeljuje mu podatke je prikazan na slici 4.11.

```
//REPORT: Transakcija report
public ActionResult TransakcijaReport(int id)
{
    //kreiranje instance lokalnog reporta
    LocalReport lr = new LocalReport();
    string path = Path.Combine(Server.MapPath("~/Reports"), "TransakcijaReport.rdlc");
    if (System.IO.File.Exists(path))
    {
        //dodjeljivanje već postojećeg .rdlc izvještaja novokreiranoj instanci
        lr.ReportPath = path;
    }
    else
    {
        return View("Index");
    }
    //upit za dohvaćanje svih transakcija unutar pripadajućeg vlasnika (mjenjačnice) i spajanje tablice
    var promet = (from p in db.Prometi
                  where p.Id == id
                  join v in db.Valute on p.ValutaBroj equals v.Broj
                  where v.VlasnikID == VlasnikId
                  join o in db.Vlasnici on p.VlasnikID equals o.Id
                  select new
                  {
                      p.Adresa,
                      p.BrojDokumenta,
                      p.BrojListe,
                      p.BrojNovcanica,
                      p.DateCreated,
                      p.Drzava,
                      p.Id,
                      p.BrojBloka,
                      p.ImePrezime,
                      p.IznosFinal,
                      p.IznosKM,
                      p.IznosProvizije,
                      p.IznosValute,
                      p.Lice,
                      p.Tecaj,
                      p.VlasnikID,
                      p.VrstaTransakcije,
                      v.Broj,
                      v.Oznaka,
                      o.Mjesto,
                      adresaVlasnika = o.Adresa,
                      o.Naziv
                  }).ToList();
    //spremanje upita u set podataka "DataSet1"
    ReportDataSource rd = new ReportDataSource("DataSet1", promet);
    lr.DataSources.Add(rd);

    //funckija koja vraće izvještaj u PDF formatu
    return RenderReport(lr);
}
```

Sl. 4.11. Kreiranje i popunjavanje izvještaja o transakciji

Primjer popunjenog izvještaja bit će prikazan u sljedećem poglavlju.

4.5. Pregled i kreiranje transakcija

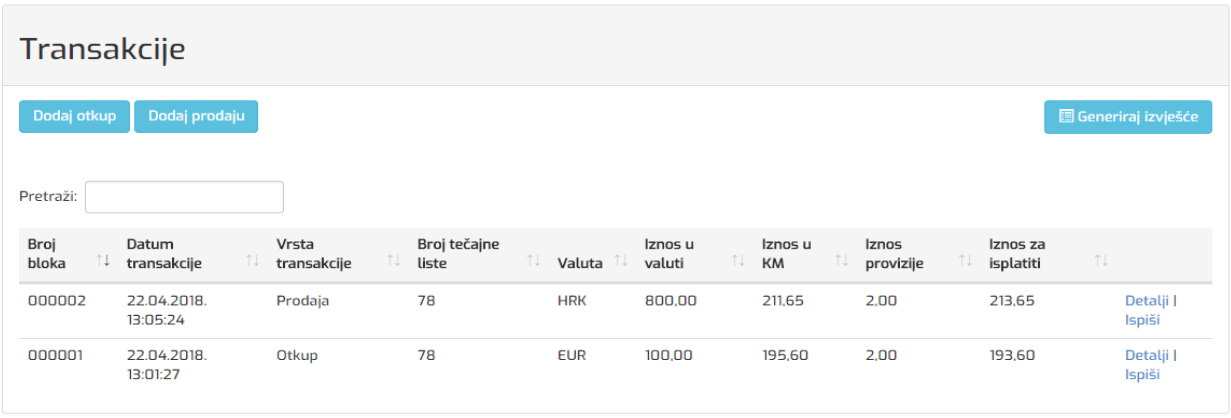
Nakon objašnjenja važnih dijelova aplikacije potrebno je prikazati rezultate izrade aplikacije. U ovom poglavlju bit će prikazan rad aplikacije i objasnit će se njezine funkcionalnosti za običnog djelatnika mjenjačnice.

Nakon prijave u aplikaciju korisniku se prikazuje početno sučelje izrađeno pomoću *Bootstrap*[6] teme koja je sastavni dio početne *ASP.NET MVC* aplikacije.

Sl. 4.12. Prikaz početnog sučelja

Djelatniku je omogućeno otvaranje drugih sučelja kroz navigacijsku traku na kojoj se nalaze opcije kreiranja transakcija, pregleda tečajni listi, pregleda popisa valuta ili pregleda stanja blagajne.

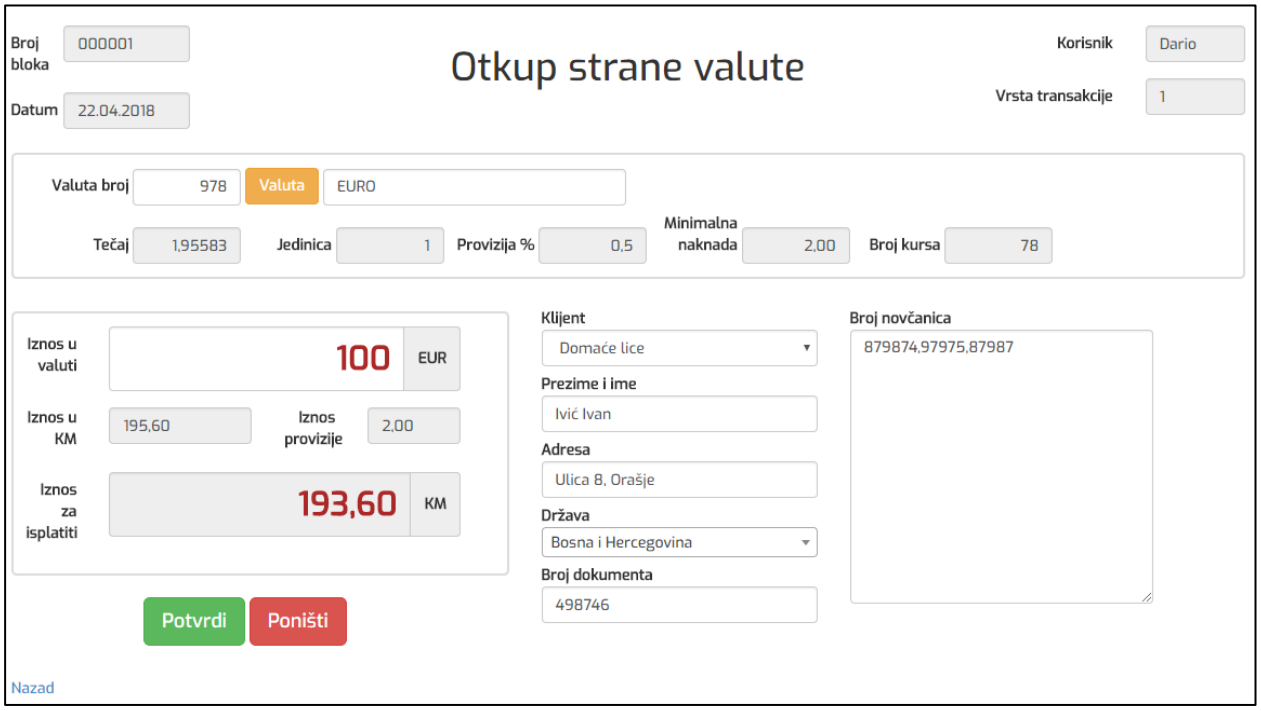
Klikom na poveznicu *Transakcije* unutar navigacijske trake, djelatniku se otvara novo sučelje koje omogućava pregled svih dosadašnjih transakcija (Sl. 4.13.). U navedenom sučelju koristi se posebna biblioteka *jQuery DataTables* koja omogućava naprednu interakciju s HTML tablicom u obliku sortiranja stupaca, pretraživanja, paginacije i dr.



Sl. 4.13. Sučelje za prikaz izvršenih transakcija

[6] *Bootstrap* je *front-end* web razvojni okvir koji sadrži gotove HTML i CSS dizajnerske obrasce za tipografiju, gumbe, forme, navigacijsku traku i druge dijelove korisničkog sučelja.

Klikom na jedan od ponuđenih gumbova, djelatnik odabire vrstu nove transakcije. Ovisno o odabiru otvara mu se sučelje pripadajuće transakcije. Na slici 4.14. prikazano je sučelje za otkup strane valute.

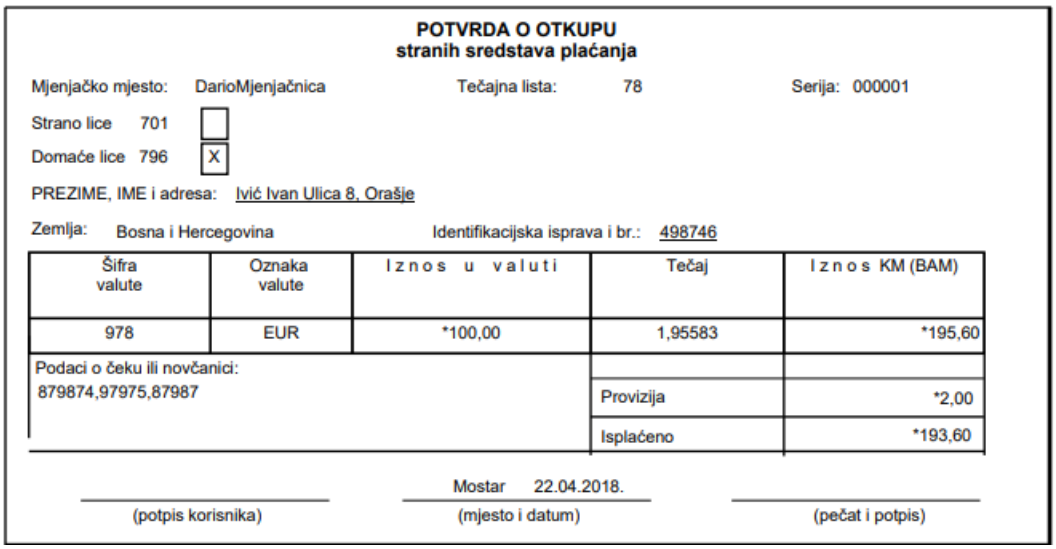


Sl. 4.14. Prikaz sučelja za ispunjavanje transakcije

Prvi korak pri ispunjavanju transakcije je da djelatnik unosi broj valute nakon čega se dohvaćaju podaci o trenutno važećem tečaju za unesenu valutu. Dohvaćanje podataka o tečaju izvršava se asinkrono pomoću *jQuery AJAX* poziva prema poslužitelju što ne zahtjeva ponovno učitavanje stranice. Ostale forme za unos, validacija i kalkulacija pretvorbe valuta izrađena je koristeći *jQuery*[7].

Nakon ispunjavanja i potvrđivanja transakcije djelatniku se u pregledniku otvara popunjeni izvještaj transakcije u PDF formatu koji je spreman za ispis (Sl. 4.15.).

[7] jQuery je biblioteka koja olakšava upotrebu JavaScripta na web stranicama



POTVRDA O OTKUPU
stranih sredstava plaćanja

Mjenjačko mjesto: DarioMjenjačnica Tečajna lista: 78 Serija: 000001

Strano lice 701 ☐

Domaće lice 796 X

PREZIME, IME i adresa: Ivić Ivan Ulica 8, Orašje

Zemlja: Bosna i Hercegovina Identifikacijska isprava i br.: 498746

Šifra valute	Oznaka valute	Iznos u valuti	Tečaj	Iznos KM (BAM)
978	EUR	*100,00	1,95583	*195,60
Podaci o čeku ili novčanici: 879874,97975,87987				
			Provizija	*2,00
			Isplaćeno	*193,60

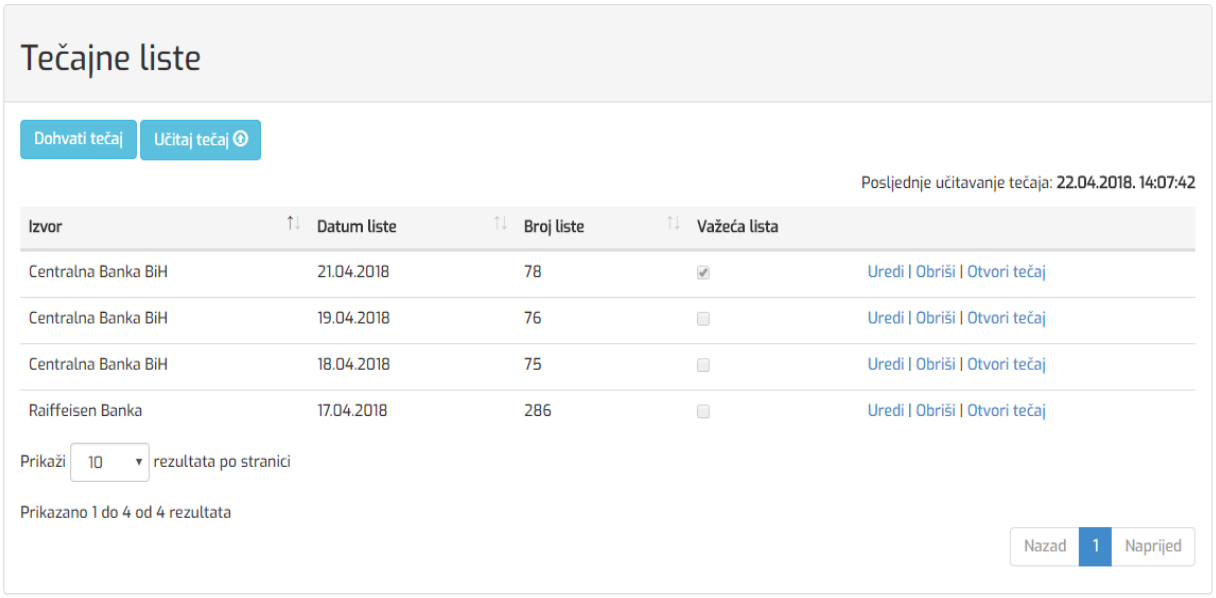
Mostar 22.04.2018.

(potpis korisnika) (mjesto i datum) (pečat i potpis)

Sl. 4.15. Izvještaj (potvrda) transakcije

4.6. Dohvaćanje i učitavanje tečajnih lista

Klikom na poveznicu *Tečajne liste* na navigacijskoj traci, djelatniku se prikazuje sučelje s popisom svih dosadašnjih tečaja i omogućuje se dohvaćanje tečajne liste ili učitavanje vlastite tečajne liste (Sl. 4.16.).



Sl. 4.16. Prikaz sučelja za tečajne liste

Dohvaćanje liste se obavlja tako da se pristupa web servisu Centralne Banke Bosne i Hercegovine. Pristupa se preko strukturiranog URL-a koristeći klasu `WebClient` (Sl. 4.17.) koja vraća najnoviju tečajnu listu u obliku JSON[8] zapisa (Sl. 4.18.)

```
public void JsonToTecaj()
{
    WebClient client = new WebClient();
    string currentDate = DateTime.Now.Date.ToString("MM/dd/yyyy");
    string url = "http://www.cbbh.ba/CurrencyExchange/GetJson?date=" + currentDate;
    var response = client.DownloadString(url);

    var result = JsonConvert.DeserializeObject<JsonTecaj>(response);
    InitializeTecaj(result, false);
}
```

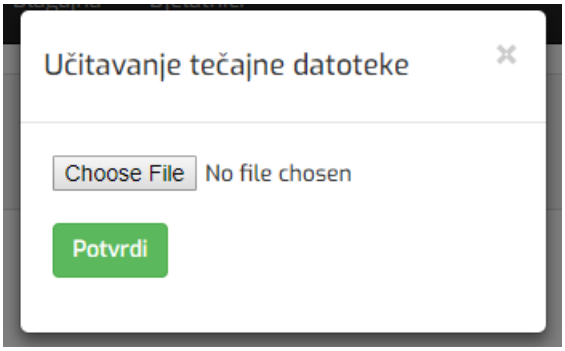
Sl. 4.17. Funkcija za dohvaćanje tečajne liste


```
{
  "CurrencyExchangeItems": [
    {
      "Country": "EMU",
      "NumCode": "978",
      "AlphaCode": "EUR",
      "Units": "1",
      "Buy": "1,95583",
      "Middle": "1,95583",
      "Sell": "1,95583"
    },
    {
      "Country": "Australija",
      "NumCode": "036",
      "AlphaCode": "AUD",
      "Units": "1",
      "Buy": "1,220635",
      "Middle": "1,223694",
      "Sell": "1,226753"
    },
```


Sl. 4.18. Tečajna lista u JSON zapisu

Nakon dohvaćanja tečajne liste, istu je potrebno deserijalizirati pomoću klase `JsonConvert` koja omogućuje pretvorbu podataka JSON tipa u *.NET* objekte (Sl. 4.17.). Podaci iz objekta se zatim spremaju u bazu podataka u tablicu `Tecaji` i tablicu `ZaglavljaTecaja`.

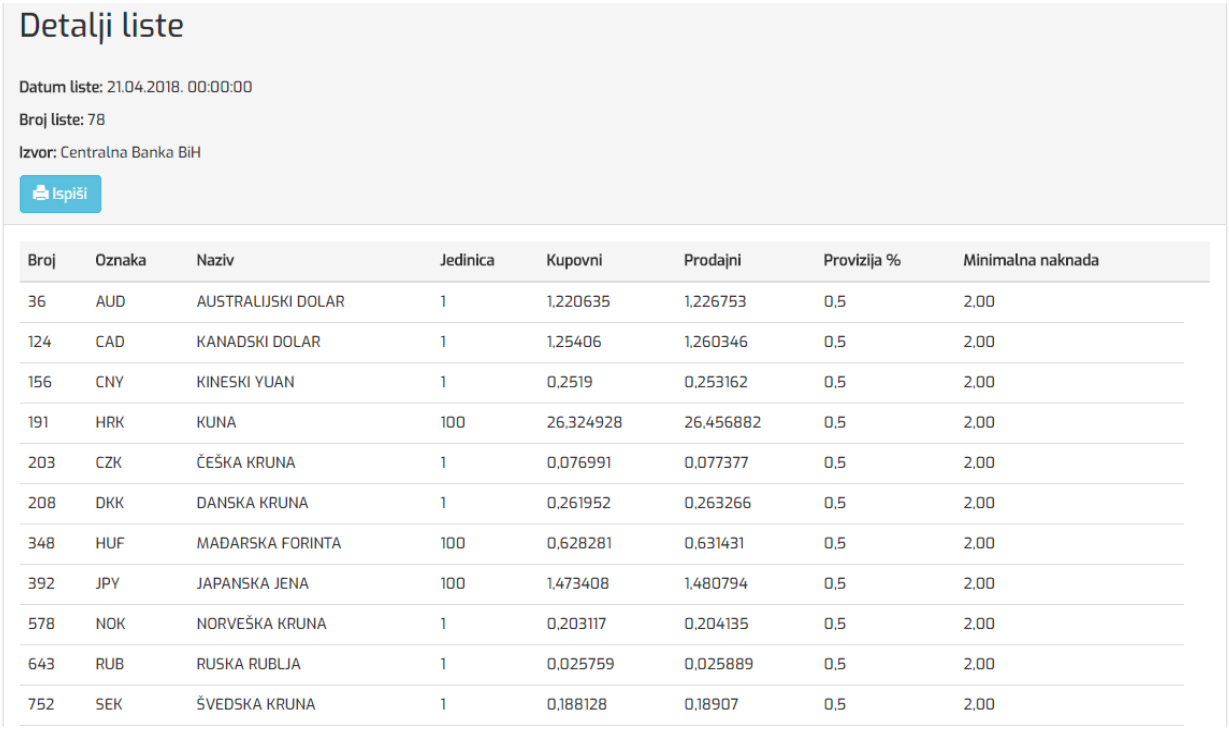
Učitavanje tečaja se koristi kada ugovorna banka pošalje mjenjačnici tečajnu datoteku. Ova datoteka obično dolazi u tekstualnom obliku i formatirane je strukture. Klikom na gumb *Učitaj tečaj* djelatniku se otvara modalni prozor u kojem u kojem može izabrati lokalno spremljenu datoteku koju želi učitati (Sl. 4.19.).



Sl. 4.19. Prikaz modalnog prozora za učitavanje liste

[8] JSON (engl. *JavaScript Object Notation)* je tekstualni i strukturirani format zapisa, namijenjen razumljivoj razmjeni podataka

Nakon učitavanja, tekst iz datoteke se parsira i podaci se raspoređuju te spremaju u odgovarajuće tablice. Ukoliko se tečajna dohvaća, kao izvor liste navodi se Centralna Banka Bosne i Hercegovine, dok kod učitavanja tečajne liste, izvor je naziv banke koja je ugovorni partner mjenjačnice (Sl. 4.16.). Detalji tek kreirane tečajne liste se mogu pogledati klikom na poveznicu *Pregled liste* u tablici tečajnih lista pri čemu se prikazuje popis svih valuta i pripadajućih tečajeva.



Detalji liste

Datum liste: 21.04.2018. 00:00:00

Broj liste: 78

Izvor: Centralna Banka BiH

Ispiši

Broj	Oznaka	Naziv	Jedinica	Kupovni	Prodajni	Provizija %	Minimalna naknada
36	AUD	AUSTRALIJSKI DOLAR	1	1,220635	1,226753	0,5	2,00
124	CAD	KANADSKI DOLAR	1	1,25406	1,260346	0,5	2,00
156	CNY	KINESKI YUAN	1	0,2519	0,253162	0,5	2,00
191	HRK	KUNA	100	26,324928	26,456882	0,5	2,00
203	CZK	ČEŠKA KRUNA	1	0,076991	0,077377	0,5	2,00
208	DKK	DANSKA KRUNA	1	0,261952	0,263266	0,5	2,00
348	HUF	MAĐARSKA FORINTA	100	0,628281	0,631431	0,5	2,00
392	JPY	JAPANSKA JENA	100	1,473408	1,480794	0,5	2,00
578	NOK	NORVEŠKA KRUNA	1	0,203117	0,204135	0,5	2,00
643	RUB	RUSKA RUBLJA	1	0,025759	0,025889	0,5	2,00
752	SEK	ŠVEDSKA KRUNA	1	0,188128	0,18907	0,5	2,00

Sl. 4.20. Popis svih tečajeva u tečajnoj listi

Klikom na gumb *Ispiši* otvara se izvještaj koji sadrži popis svih valuta i pripadajućih tečaja unutar te tečajne liste. Navedeni izvještaj se otvara u novom prozoru preglednika u PDF obliku spreman za ispis (Sl. 4.21.). Mjenjačnice bi trebale, nakon svake nove tečajne liste, na vidljivom mjestu istaknuti trenutnu aktivnu tečajnu listu.

TEČAJNA LISTA BROJ: 78

Mjenjačko mjesto: DarioMjenjačnica
15. Ulica Mostar

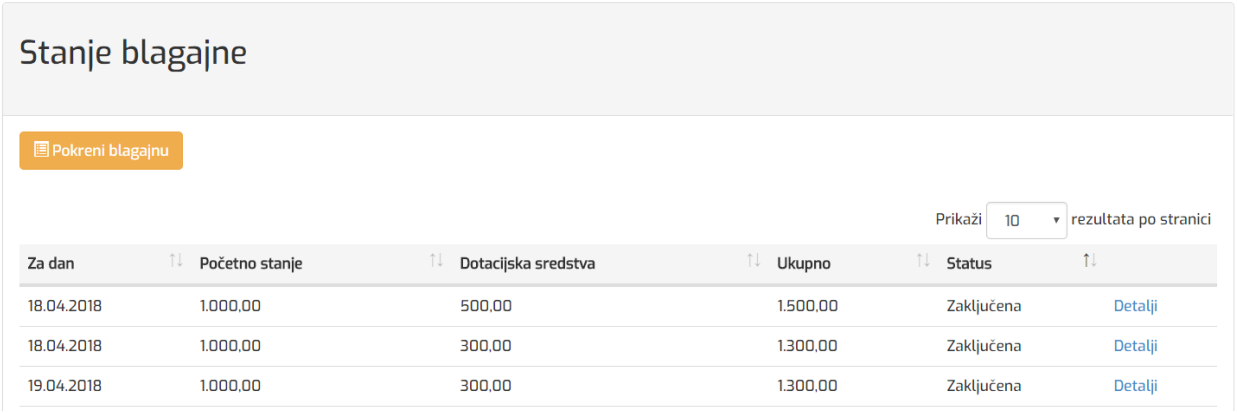
21.04.2018.

Valuta	Oznaka	Naziv	Jedinica	Kupovni	Prodajni	Provizija %
36	AUD	AUSTRALIJSKI DOLAR	1	1,220635	1,226753	0,50
124	CAD	KANADSKI DOLAR	1	1,25406	1,260346	0,50
156	CNY	KINESKI YUAN	1	0,2519	0,253162	0,50
191	HRK	KUNA	100	26,324928	26,456882	0,50
203	CZK	ČEŠKA KRUNA	1	0,076991	0,077377	0,50
208	DKK	DANSKA KRUNA	1	0,261952	0,263266	0,50
348	HUF	MAĐARSKA FORINTA	100	0,628281	0,631431	0,50
392	JPY	JAPANSKA JENA	100	1,473408	1,480794	0,50
578	NOK	NORVEŠKA KRUNA	1	0,203117	0,204135	0,50
643	RUB	RUSKA RUBLJA	1	0,025759	0,025889	0,50
752	SEK	ŠVEDSKA KRUNA	1	0,188128	0,18907	0,50
756	CHF	ŠVICARSKI FRANAK	1	1,629858	1,638028	0,50
826	GBP	BRITANSKA FUNTA	1	2,226898	2,23806	0,50
840	USD	AMERIČKI DOLAR	1	1,584971	1,592915	0,50

Sl. 4.21. Izvještaj tečaja

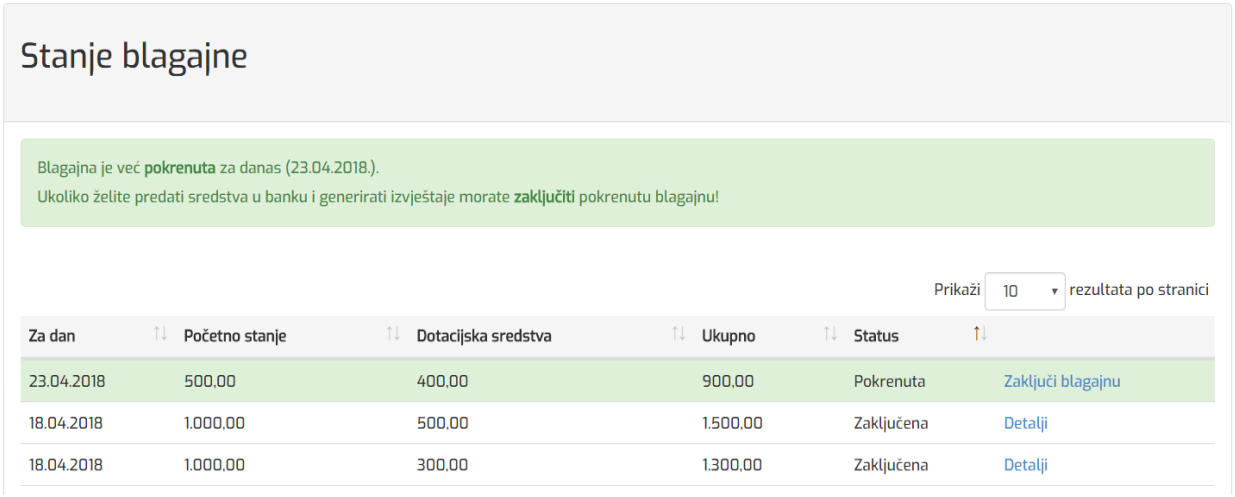
4.7. Implementacija blagajne

Tablica `Blagajne` predstavlja drugu najvažniju tablicu u aplikaciji. Sve transakcije vezanu su za samo jednu blagajnu, onu koja je trenutno pokrenuta. Na početku svakog radnog dana ili smjene, djelatnik treba pokrenuti blagajnu klikom na gumb *Pokreni blagajnu* na prikazanom sučelju (Sl. 4.22.). Na prikazanom sučelju se nudi mogućnost pregleda dosadašnjih blagajni kao i detalji svake od njih.



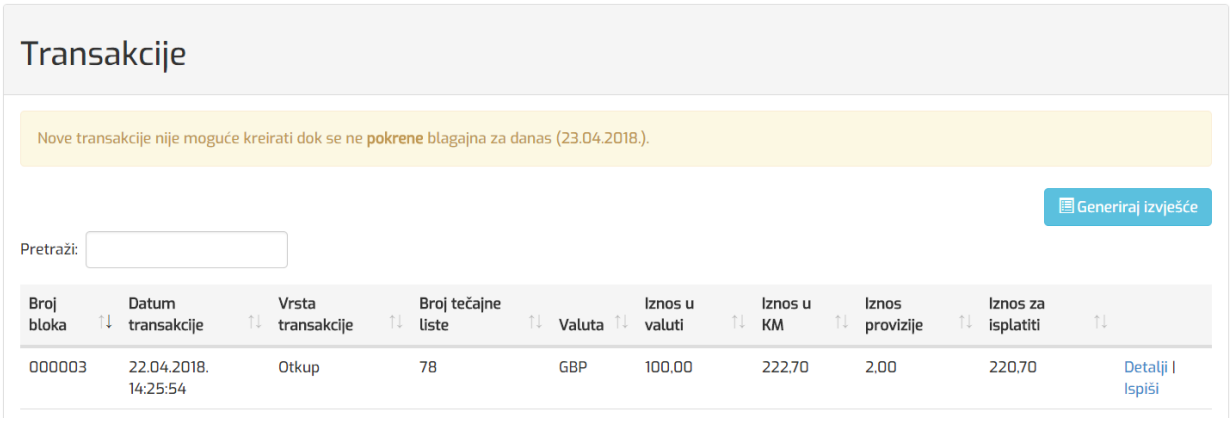
Sl. 4.22. Prikaz sučelja za upravljanje blagajnom

Pri pokretanju blagajne djelatnik unosi iznos početnog stanja blagajne i dotacijskih sredstava – iznosa koji mu banka daje prilikom predaje rekapitulacijskog izvještaja. Nakon što ispuni potrebnu formu, kreira se nova blagajna za trenutni dan s statusom *Pokrenuta* (Sl. 4.23.).



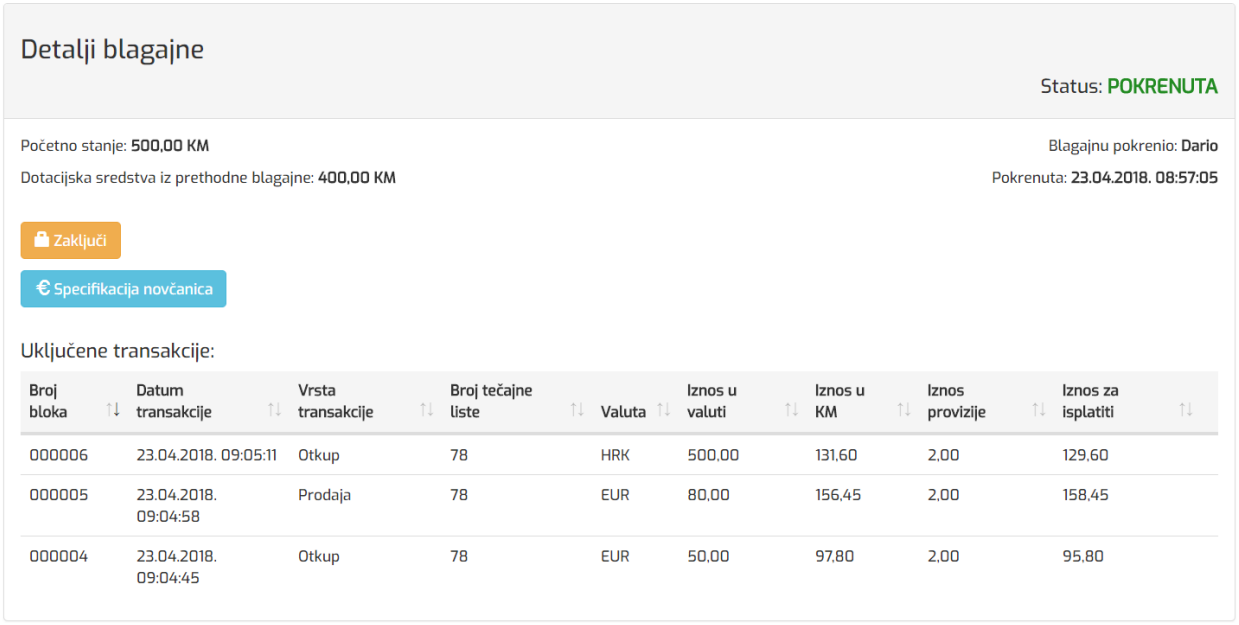
Sl. 4.24. Prikaz sučelja nakon pokretanja blagajne

Transakcije otkupa i prodaje se mogu obavljati samo ako postoji trenutno pokrenuta blagajna. U protivnom će se djelatniku na sučelju za upravljanje transakcijama prikazati poruka da nema trenutno pokrenute blagajne i gumbi za transakcije se neće prikazati (Sl. 4.25.)

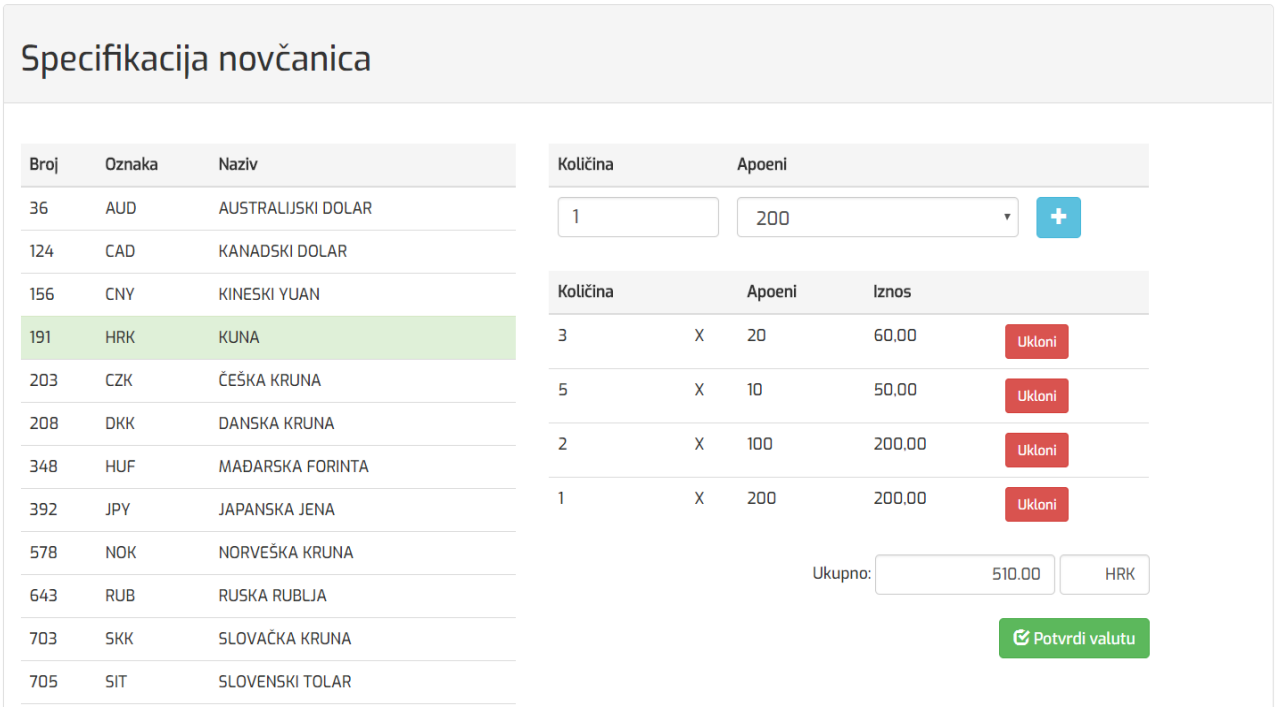


Sl. 4.25. Onemogućavanje transakcija ako nema pokrenute blagajne

Klikom na poveznicu *Zaključi blagajnu* djelatniku se prikazuje sučelje s detaljima pokrenute blagajne gdje su jasno prikazane sve transakcije koje spadaju pod trenutnu blagajnu. Uz to, prikazan je status blagajne s informacijama o djelatniku i vremenu pokretanja blagajne (Sl. 4.26.)

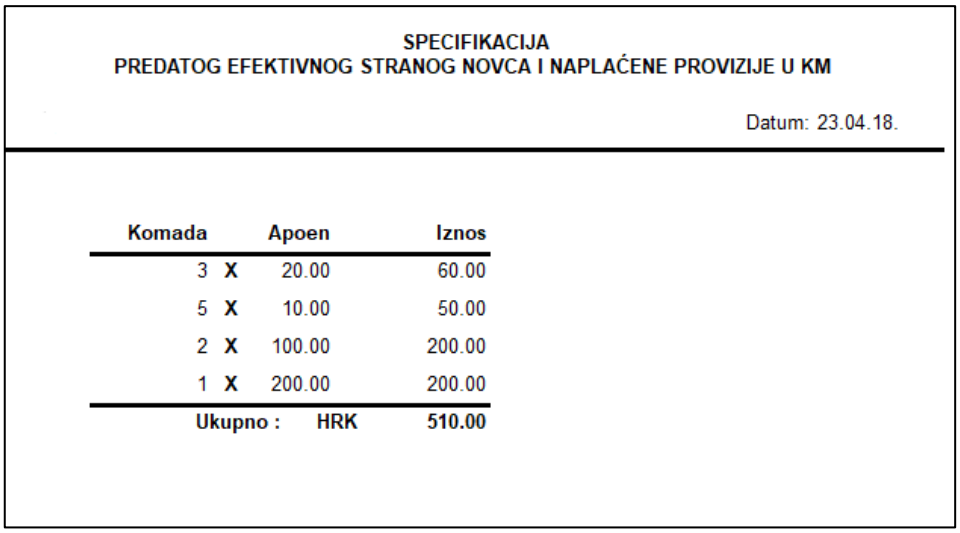


Na prikazanom sučelju korisnik može specificirati novčanice stranih valuta prije predaje izvještaja banci. Klikom na gumb *Specifikacija novčanica* otvara se novo sučelje gdje korisnik specificira novčanice po valutama (Sl. 4.27.).



Sl. 4.27. Sučelje za specifikaciju novčanica

U tablici s lijeve strane djelatnik odabire valutu čije novčanice želi specificirati a na desnoj strani unosi količinu i apoene pri čemu se prikazuje ukupan iznos za odabranu valutu. Klikom na *Potvrdi valutu* sprema unesene podatke. Cijeli proces specifikacije novčanica je dinamičan i bez osvježavanja trenutne stranice zbog korištenja *jQuery* biblioteke. Nakon potvrde u novom prozoru preglednika se prikazuje izvještaj specifikacije koji se predaje banci zajedno s ostalim izvještajima (Sl. 4.28.).



SPECIFIKACIJA
PREDATOG EFEKTIVNOG STRANOG NOVCA I NAPLAĆENE PROVIZIJE U KM

Datum: 23.04.18.

Komada		Apoen	Iznos
3	X	20.00	60.00
5	X	10.00	50.00
2	X	100.00	200.00
1	X	200.00	200.00
Ukupno :		**HRK**	**510.00**

Sl. 4.28. Izvještaj specifikacije stranog novca

Ukoliko djelatnik na sučelju za detalje blagajne (Sl. 4.26.) klikne na gumb *Zaključi*, mijenja se status blagajne u *Zaključena* i u novom prozoru preglednika se otvara izvještaj *Obračun stanja blagajne* koji sadrži podatke o stanju blagajne kada se uzmu u obzir svi otkupi i prodaje deviza (Sl. 4.29.).

OBRAČUN DNEVNE BLAGAJNE MJENJAČNICE U KM

Mjenjačko mjesto: DarioMjenjačnica, 15. Ulica, Mostar

I	Početno stanje	500,00
II	Primljeno po osnovu dotacije	400,00
III	Primljeno po Dnevniku prodaje stranih sredstava plaćanja (kolona 7)	156,45
IV	Primljeno na ime provizije (kolona 8 iz Dnevnika otkupa + Dnevnika prodaje)	2,00
A)	**Ukupno primljeno (II+III+IV)**	**558,45**
V	Izdato po Dnevniku otkupa stranih sredstava plaćanja (kolona 7)	229,40
VI	Predato Banci na ime provizije (isto kao i stav IV)	4,00
B)	**Ukupno izdato (V+VI)**	**233,40**
Stanje blagajne (I + A - B) na dan		**825,05**

Specifikacija

Komada		*Apoena*		*Iznos*
_____	X	1.000,00	=	_____
_____	X	500,00	=	_____
_____	X	200,00	=	_____
_____	X	100,00	=	_____
_____	X	50,00	=	_____
_____	X	20,00	=	_____
_____	X	10,00	=	_____
_____	X	5,00	=	_____
_____	X	1,00	=	_____
_____	X	0,50	=	_____
_____	X	0,20	=	_____
_____	X	0,10	=	_____
_____	X	0,05	=	_____

Mostar, 23.04.2018.

Mjesto i datum

Potpis i pečat

Sl. 4.29. Izvještaj za obračun stanja blagajne

5. ZAKLJUČAK

Većina postojećih aplikacija za poslovanje mjenjačnica na tržištu postoji samo u obliku desktop aplikacije. Desktop aplikacije zahtijevaju instalaciju na klijentsko računalo te mogu biti ograničene hardverom na kojem se pokreću. Samim time ih je teže održavati i ažurirati. Ovaj diplomski rad je nastao iz ideje da se omogući upravljanje poslovanjem mjenjačnice putem moderne web aplikacije. Sve što je potrebno za pristup aplikaciji je internetska veza i internetski preglednik.

Izrađena aplikacija omogućuje djelatnicima mjenjačnica olakšano upravljanje poslovanjem kroz intuitivno korisničko sučelje i niz funkcionalnosti koje nudi. Aplikacija omogućava upravljanje djelatnicima, kreiranje transakcija, dohvaćanje i učitavanje tečajnih listi, generiranje propisanih izvještaja spremnih za ispis, upravljanje blagajnom i drugo. Pri izradi aplikacije korištene su brojne tehnologije i metode a poseban naglasak je na razvojnom okviru ASP.NET i MVC obrascu oblikovanja aplikacije.

Ovaj rad je rezultirao uspješnom izradom web aplikacije koja nudi mogućnost modernijeg i pristupačnijeg načina upravljanja poslovanjem mjenjačnice. Izrađena aplikacija može se koristiti u poslovne svrhe za ovlaštene mjenjačnice na prostoru Bosne i Hercegovine. Uz daljnju nadogradnju i proširenja, aplikacija se može prilagoditi i za druge zemlje.

LITERATURA

[1] Mjenjačnice, http://mjenjacnice.hr/mjenjacnice-novac, pristup ostvaren 25.03.2018.

[2] ISO 4217, https://hr.wikipedia.org/wiki/ISO_4217, pristup ostvaren 25.03.2018.

[3] How Web Works & ASP.Net MVC fits into Web Application Development, http://www.dotnetodyssey.com/asp-net-mvc-5-free-course/how-web-works-asp-net-mvc-fits-web-application-development/

[4] G. Booch, Object-oriented Analysis and Design with Applications: 2nd edition, Addison Wesley, 2007, http://kmvportal.co.in/Course/OOAD/object-oriented-analysis-and-design-with-applications-2nd-edition.pdf, pristup ostvaren 25.03.2018.

[5] A. Freeman, Pro ASP.NET MVC 5, 5th edition, Apress, 2013, http://enos.itcollege.ee/~ijogi/Nooks/Pro%20ASP.NET%20MVC%205/Pro%20ASP.NET%20MVC%205.9781430265290.pdf, pristup ostvaren 25.03.2018.

[6] Introduction to ASP.NET Web Programming Using the Razor Syntax, https://docs.microsoft.com/en-us/aspnet/web-pages/overview/getting-started/introducing-razor-syntax-c, pristup ostvaren 04.04.2018.

[7] Entity Framework, https://docs.microsoft.com/en-us/ef/ef6/, pristup ostvaren 26.03.2018.

[8] T. Dykstra, R. Anderson, Getting Started with Entity Framework 6 Code First using MVC 5: Step By Step Guide, Microsoft, 2014

[9] What is Entity Framework, http://www.entityframeworktutorial.net/, pristup ostvaren 27.03.2018.

[10] LINQ upiti, http://www.mikroknjiga.rs/Knjige/CT5Z/08_CT5Z.pdf, pristup ostvaren 04.04.2018.

[11] Introduction to ASP.NET Identity, https://docs.microsoft.com/en-us/aspnet/identity/overview/getting-started/introduction-to-aspnet-identity, pristup ostvaren 10.04.2018.

[12] ReportViewer Controls (Visual Studio), https://msdn.microsoft.com/en-us/library/ms251671.aspx, pristup ostvaren 12.04.2018.

SAŽETAK

Ovaj rad opisuje izradu poslovne web aplikacije za poslovanje mjenjačnice korištenjem obrasca oblikovanja model-pogled-kontroler (engl. *Model-View-Controler – MVC*) unutar *ASP.NET MVC* razvojnog okvira. Cilj je prikazati i opisati mogućnosti *ASP.NET MVC* tehnologije, uz niz drugih korištenih alata i programskih okvira pri izradi poslovne web aplikacije. Izrađena aplikacija omogućuje korisnicima kreiranje transakcija u obliku otkupa i prodaje stranih valuta, svakodnevno dohvaćanje ili učitavanje najnovijih tečajnih lista, praćenje vlastitog prometa, generiranje potvrda transakcija kao i generiranje raznih propisanih izvještaja spremnih za ispis. Opisane su korištene metode i tehnologije te njihova upotreba i primjena u izradi web aplikacije iz praktičnog dijela rada. Naglasak je na primjeni MVC obrasca oblikovanja koji je temelj korištenog programskog okvira *ASP.NET MVC,* kao i *Entity Framework* okviru i *Code First* pristupu u spomenutoj aplikaciji.

Ključne riječi: mjenjačnica, web aplikacija, ASP.NET MVC, Entity Framework, Code First, izvještaji, Identity

ABSTRACT

Development of a Web Application for Currency Exchange Business

This graduate paper describes the development of a web application for currency exchange business using the Model-View-Controller (MVC) design pattern within the ASP.NET MVC framework. The goal is to demonstrate and describe the capabilities and features of ASP.NET MVC technology, along with several other tools and frameworks used in the development of a business web application. Implemented application offers users to create transactions in the form of purchases and sales of foreign currencies, daily fetching or uploading of the latest exchange rates, monitoring of own turnover, generation of transaction invoices and generation of various prescribed reports ready for print. Various methods and technologies used in the development of web applications are described. The emphasis is on the implementation of the MVC design pattern which is the basis of the used ASP.NET MVC framework along with the Entity Framework and Code First approach in mentioned application.

Key words: exchange office, web application, ASP .NET MVC, Entity Framework, Code First, reports, Identity

ŽIVOTOPIS

Dario Nedić rođen je u Vinkovcima 11.listopada 1993. godine. Osnovnu školu završio u Tolisi (BiH) u razdoblju od 2000. do 2008. godine. Upisuje opću gimnaziju u Orašju 2008. godine, koju završava 2012. godine. Nakon toga upisuje Elektrotehnički fakultet u Osijeku, preddiplomski studij, smjer računarstvo kojega završava 2015. godine. 2016. godine upisuje diplomski studij na prijašnje spomenutom fakultetu, sada Fakultet elektrotehnike, računarstva i informacijskih tehnologija, smjer računarstvo, izborni blok programsko inženjerstvo kojega trenutno pohađa. Trenutno je zaposlen kao junior softverski inženjer u tvrtki megaIN2 u Orašju (BiH) koja je članica IN2 grupe.

PRILOZI

Na CD-u:

1. Diplomski rad „Izrada web aplikacije za poslovanje mjenjačnice.docx“
2. Diplomski rad „Izrada web aplikacije za poslovanje mjenjačnice.pdf“
3. Izvorni kôd aplikacije