**VISOKA ŠKOLA STRUKOVNIH STUDIJA ZA INFORMACIONE TEHNOLOGIJE**



**Projektni centar**

**Dokumentacija projekta Vizija**

**Tim Vizija** - Studenti:

Product owner: Mateja Jovanović 360/16, IT

Scrum master: Stefan Stanković 126/16, IT

Dizajner: Tamara Ristić 279/15, RM

Developer: Ilhan Kalač 429/16, IT

Developer: Pavle Lukić 439/16, IT

Developer: Igor Marjanović 457/16, IT

Datum predaje:3.2.2017

**Beograd**

**3.2018**

Sadržaj

[Uputstvo za upotrebu 2](#_Toc508877421)

[Rezime 2](#_Toc508877422)

[Struktura baze podataka 2](#_Toc508877423)

[Modeli 5](#_Toc508877424)

[Kontroleri 6](#_Toc508877425)

[Pogledi (Views) 7](#_Toc508877426)

[Literatura 7](#_Toc508877427)

# Uputstvo za upotrebu

1.Pokrenuti .sql fajl za inicijalizaciju baze podataka (neophodan program za rad sa SQL bazama).

2.U projetku (kodu) izmeniti podatke konekcionog stringa koji se nalazi u fajlu Web.config

- linija 12 u kodu, izbrisati

(U slučaju da dodje do konflikta sa bazom)

AttachDbFilename=|DataDirectory|\aspnet-TestFinal-20180311021134.mdf

-linija 13 u kodu promeniti parametar na : data source=”Ime lokalnog SQL servera “

data source=DESKTOP-KFQB44U\SQLEXPRESS

3.Van solution-a ručno izbrisati ceo sadržaj foldera TestFinal/App\_Data

4.Pokrenuti program

# Rezime

Projekat je zamišljen kao zatvoreni sistem, tj. jedino korisnici koji su prijavljeni mogu da imaju pristup funkcionalnostima aplikacije. Opcija za registraciju, tj. kreiranje novih korisnika, isključivo je dostupna korisnicima sa pravom admin-a.

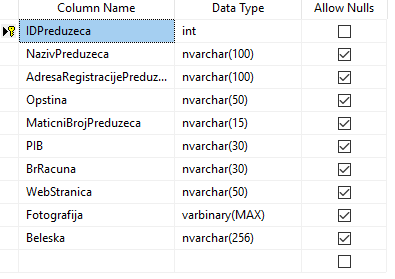
Rešenje je realizovano Database first pristupom te će prvo biti reči o strukturi baze podataka koja je zadužena za rad se popunjavanjem formulara. Nakon toga, ASP.net Identity sa svojom bazom, zadužen je za rad sa korisnicima i njihovim pravima pristupa. Modifikovani ASP.net entity framework implementiran je za rad sa bazom za formulare kroz MVC tehnologiju.

# Struktura baze podataka

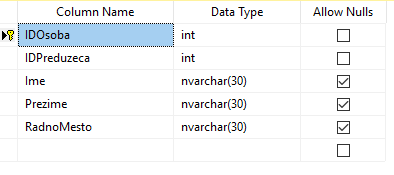
Baza podataka aplikacije realizivana je korišćenjem dve odvojene baze podataka, jedne zadužene za skladištenje podataka o popunjavanjima formulara, dok je druga baza namenjena isključivo za rad sa korisnicima.

Baza VizijaX sastoji se od četiri tabele, svaka sa podacima neophodnih za ispunjavanje uslova popunjavanja formulara.

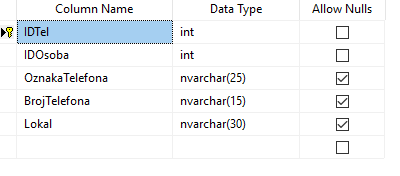
Tabela GlavnaTabela (Slika 1.), služi za popunjavanje glavnih podataka vezanih za preduzeće. Pomoću identity ključa IDPreduceca ova tabela nadovezuje se na tabelu Kontakt (Slika 2.) što predstavlja listu kontakata koje neko preduzeće poseduje. Podaci koji su cifre stavljeni su u varchar format da bi bilo moguće uneti veće cifre bez potencijalnog prekoračenja maksimalne veličine u memoriji koji zauzimaju vrednosni tipovi podatka rezervisanih za brojčane vredosti. Kontakt tabela poseduje polje IDOsoba koje je tipa identity i predstavlja strani ključ u tabelama KontaktMail (Slika 3.) i KontaktTelefon (Slika 4.), što ostvaruje mogućnost kontakta da poseduje više telefona i više mail adresa. Operacije unosa i čitanja podataka rešene su korišćenjem modela kreiranih na osnovu modifikovanog ASP.net entity framework-a. Za konzistentnost podataka uključena je kaskadna modifikacija i kaskadno brisanje podataka. Dijagram povezanosti tabela može se videti na slici 5.



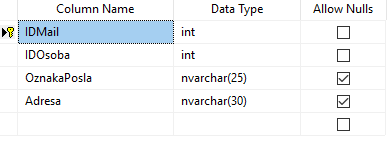
Slika 1. – GlavnaTabela



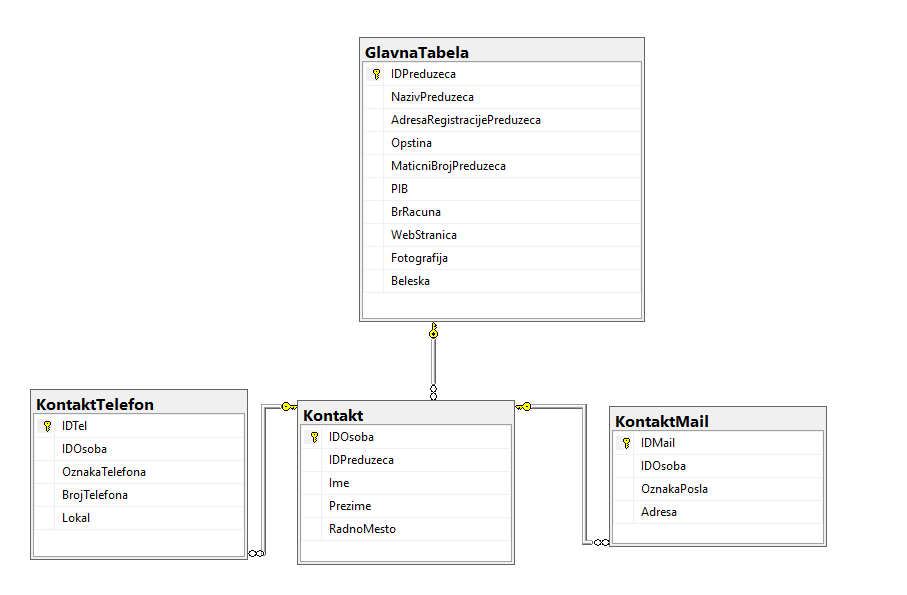
Slika 2. – Kontakt tabela



Slika 3. – KontaktTelefon - tabela



Slika 4. – KontaktMail – tabela



Slika 5. – Dijagram baze VizijaX

Baza podataka zadužena za upravljanje korisnika realizovana je upotrebom ASP.net Identity tehnologije. Podaci od interesa za rad sa korisnicima su Username, Password(Hash) i Email. Identity tehnologija nudi pet tabela za rad sa korisnicima od kojih su AspNetUsers, AspNetRoles i AspNetUserRoles glavne tabele za rad sa korisnicima. Tabela AspNetUsers sadrži sve neophodne podatke o korisniku osim njihovog pravila pristupa aplikaciji. Pomoću polja UserID iz tabele AspNetUsers preko podatka RoleID iz tabele AspNetUserRoles, svaki korisnik vezan je za jedan od tri moguća prava pristupa: Admin, RWE (read-write-execute), RE (read-execute) koja su definisana u tabeli AspNetRoles. Korisnici sa pravom Admin pristupa mogu koristiti sve funkcionalnosti aplikacije. RWE korisnici imaju pravo pristupa čitanju, modifikovanju i štampanju formulara kao i svih njihovih kontakata koji dalje poseduju svoje telefone i mail adrese. RE korisnici imaju pravo pristupa za čitanje i štampanje podataka vezanih za rad sa formularima.

# Modeli

Modeli su bazirani na ASP.net MVC tehnologiji sa modifikacijama vezanim za povezivanje svih tabela od interesa. Za svaku tabelu iz baze kreiran je model sa identičnim imenom kao u bazi podataka. Za glavnu tabelu pored podataka iz baze unet je atribut :

public HttpPostedFileBase ImageFile { get; set; }

koji ima ulogu posrednika u prihvatanju informacije o slici i prosleđivanju iste u odgovarajući atribut modela glavne tabele koji predstavlja podatak o slici (pečatu). Validacija unosa za svaki podatak od interesa obezbedjena je korišćenjem atributa iz System.ComponentModel.DataAnnotations, primer:

[StringLength(30, ErrorMessage = "Naziv preduzeća može imati najviše 30 karaktera.")]

        [Display(Name = "Naziv preduzeća")]

        [Required(ErrorMessage = "Neophodno popuniti naziv preduzeća.")]

        public string NazivPreduzeca { get; set; }

Ulančavanje tabela obezbeđeno je uvođenjem atributa Kontakts u modelu GlavnaTabela (primer veze GlavnaTabela – Kontakt, slično za sve ostale modele i njihove veze):

public virtual ICollection<Kontakt> Kontakts { get; set; }

Model za upravljanje korisnicima pozajmljuje dizajn od ASP.net identity tehnologije. Od glavnog interesa su view modeli koji se nalaze u AccountViewModels klasi, RegisterViewModel i LoginViewModel. Za prijavljivanje na aplikaciju neophodna su polja za korisničko ime i lozinku i potvrdu iste. Prilikom kreiranja korisnika, admin pored prethnodno navedenih informacija mora uneti i validnu mail adresu. Klasa RoleName služi za implementaciju autorizacije pristupa kontrolama i sadrži imena tri moguća prava pristupa. UserWithRoles predstavlja model koji pored korisničkih podataka povezuje i ime prava pristupa korisniku i koristi se da na stranici za pregled korisnika prikaže i prava pristupa koji svaki od postojećih korisnika poseduje. Model ResetUserPasswordViewModel služi za funkcionalnosti resetovanja korisničke lozinke.

Validacija unosa stavljena je na sva mesta poput validacije podataka u modelima za popunjavanje formulara, primeri:

[Required(ErrorMessage = "Polje za Korisničko ime mora biti popunjeno.")]

        [StringLength(20, ErrorMessage = "Username može imati najvise 20 karaktera.")]

        [Display(Name = "Korisničko ime")]

        public string UserName { get; set; }

[Required(ErrorMessage = "Polje za Lozinku ime mora biti popunjeno.")]

        [StringLength(100, ErrorMessage = "{0} mora biti bar {2} karaktera dug.", MinimumLength = 6)]

        [DataType(DataType.Password)]

        [Display(Name = "Lozinka")]

        public string Password { get; set; }

# Validacija unosa dodatno je obezbeđena ubacivanjem nove linije u FilterConfig.cs.

filters.Add(new HandleErrorAttribute());

Pored toga u Web.config dodate su sledeće linije:

    <customErrors mode="On" defaultRedirect="~/Error/NotFound" >

      <error statusCode="404" redirect="~/Error/NotFound" />

      <error statusCode="405" redirect="~/Error/NotFound" />

    </customErrors>

# Kontroleri

ASP.net entity framework dostavlja CRUD metode za rad sa modelima vezanih za popunjavanje formulara. Hendlovanje potencijalnih grešaka realizovano je upotrebom ručnog hendlovanja grešaka uz pomoć try-catch blokova koji u neželjenim situacijama vrše redirekciju korisnika na stanicu za greške i uporedno održavaju konzistentnog baze podataka nedozvoljavajući da se izvrše neželjene promene u bazi. Autorizacija pristupa svakoj od mogućih akcija realizovana je primenom provere atributa gde se koristi model RoleName, primer za pristup metodi kreiranja novih formulara preduzeća:

[Authorize(Roles =RoleName.RWE+ ","+ RoleName.Admin)]

        public ViewResult Create()

        {

            return View();

        }

Na drugim mestima gde je neophodno generisati drugačije pristupe, na primer različitih pogleda na formulare u zavisnosti da li se radi o korisniku sa RWE i Admin ili RE pristupom (primer grananja pregleda stranice koja predstavlja pogled na formulare):

public ViewResult Index()

        {

            if (User.IsInRole(RoleName.Admin) || User.IsInRole(RoleName.RWE))

            {

                return View(db.GlavnaTabelas.ToList());

            }

            return View("ReadIndex",db.GlavnaTabelas.ToList());

        }

Kontroler ErrorController sa metodom NotFound predstavlja hendler grešaka koja vrši redirekciju korisnika na globalnu stranicu za greške gde se nudi mogućnost korisnicima da se vrate na početnu stranicu aplikacije. KorisniciController zadužen je za prikaz i brisanje korisnika iz identity baze. AccountController radi ostale funkcionalnosti vezane za upravljanje sa korisnicima (Login, logout, kreiranje, itd.). Metoda EditRegister koja se nalazi u AccountControlleru je modifikovana register metoda koja ima ulogu menjanja podataka o korisnicima (ili očuvanje istih). Metoda ResetUserPassword ima ulogu resetovanja lozinki korisnika. Sve CRUD metode vezane za korisnike isključivo su dostupne samo adminu. Sve ostale metode(akcije) opisuju svoje uloge svojim imenom.

**Pogledi (Views)**

Pogledi su realizovani primenom modifikovanog ASP.net entity framework-a. Primenom Linq upita i korišćenjem ViewBag-ova omogućeno je prosleđivanje podataka na poglede u kojima je imalo potrebe prezentovati podatke koji drugačije nisu bili prosleđeni, primer prosleđivanja podataka o pravilima pristupa korisnika za generisanje pregleda korisnika sa svojim pravilima pristupa:

var sql = @"

            SELECT AspNetUsers.UserName, AspNetUsers.PasswordHash, AspNetUsers.Email,AspNetUsers.Id, AspNetRoles.Name As Role

            FROM AspNetUsers

            LEFT JOIN AspNetUserRoles ON  AspNetUserRoles.UserId = AspNetUsers.Id

            LEFT JOIN AspNetRoles ON AspNetRoles.Id = AspNetUserRoles.RoleId";

            var result = db.Database.SqlQuery<UserWithRoles>(sql).ToList();

            return View(result);

Primer za linq i viewbag-ove:

var kontakts = db.Kontakts.Include(k => k.GlavnaTabela).Where(k=> k.IDPreduzeca == id);

            ViewBag.Id = id;

            pom = id;

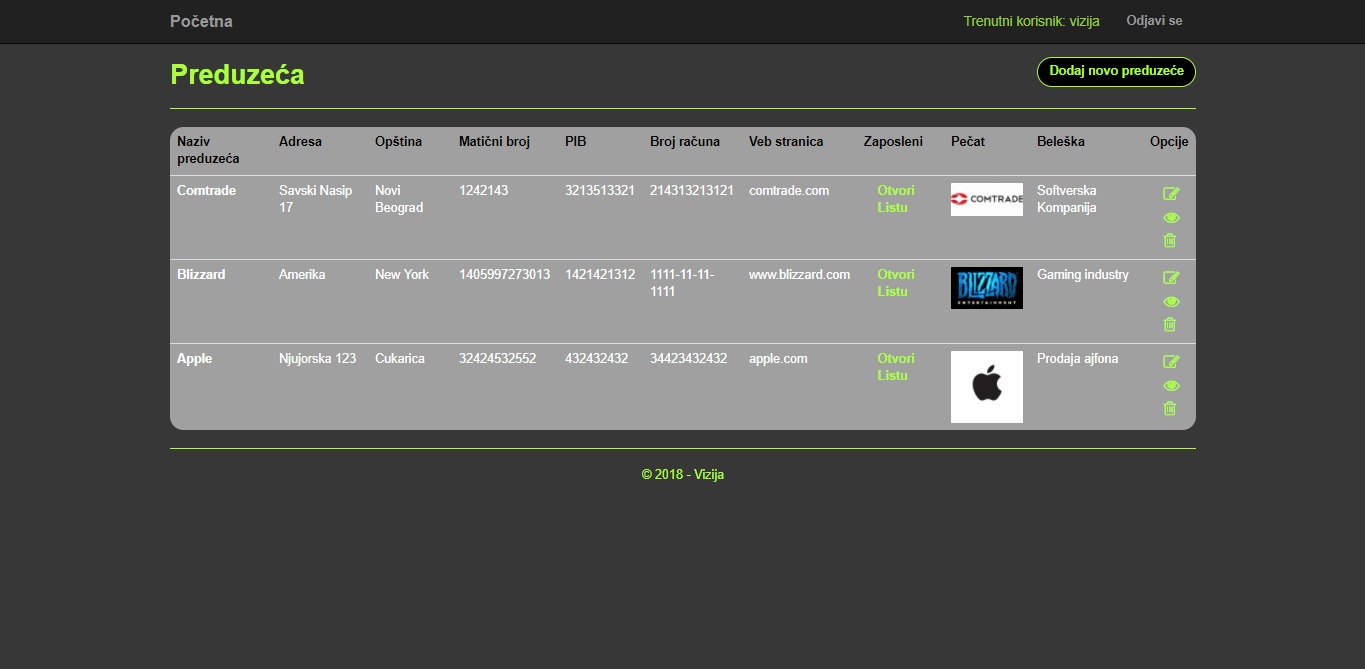
            ViewBag.Pom = db.GlavnaTabelas.Where(k => k.IDPreduzeca == id).Select(k => k.NazivPreduzeca).FirstOrDefault();

            if (User.IsInRole(RoleName.RE))

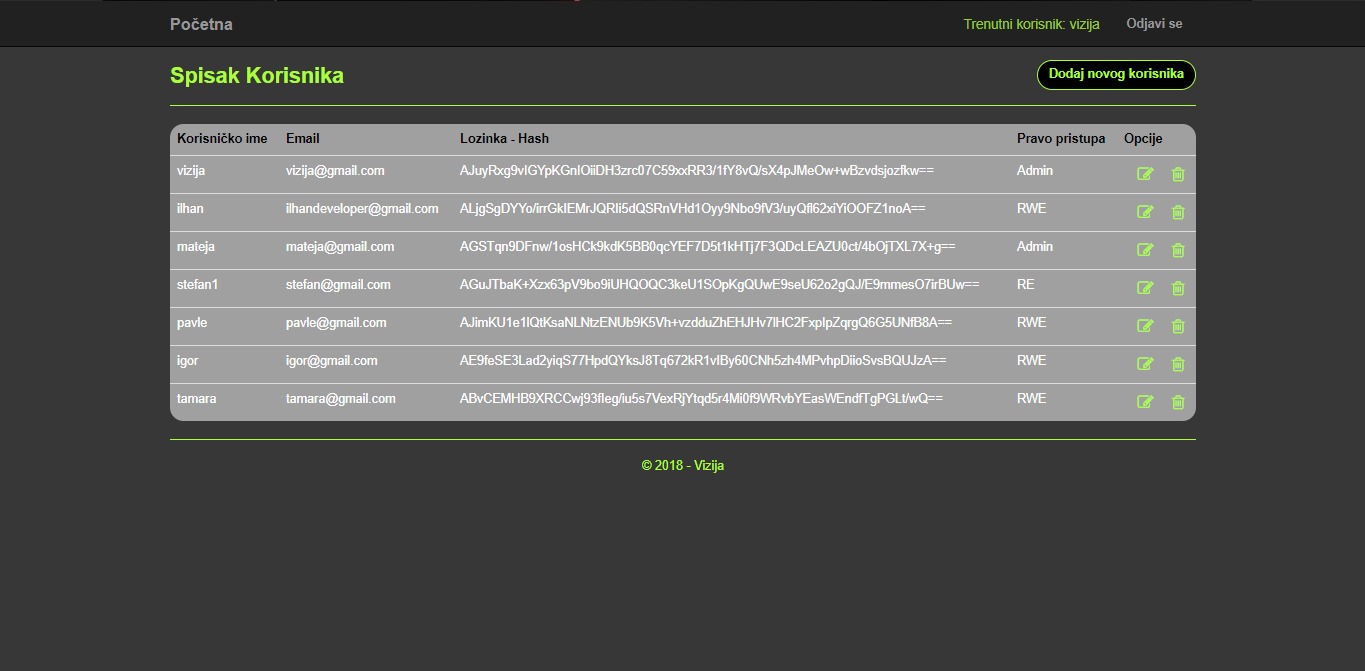
                return View("ReadIndex", kontakts.ToList());

            return View(kontakts.ToList());

# Pored toga svaki view koristi specifičan dizajn izgleda, primer (Slike 6. i 7.)



Slika 6. Pogled formulari preduzeća (GlavnaTabela)



Slika 7. Pogled za rad sa korisnicima

# Literatura

<https://msdn.microsoft.com/en-us/dn308572.aspx>

<https://stackoverflow.com/>

http://aryalnishan.com.np/