SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 2954

INTERNETSKI PORTAL ZA PRAĆENJE PROCESNIH VELIČINA MIKROMREŽE

Stjepan Lipnik

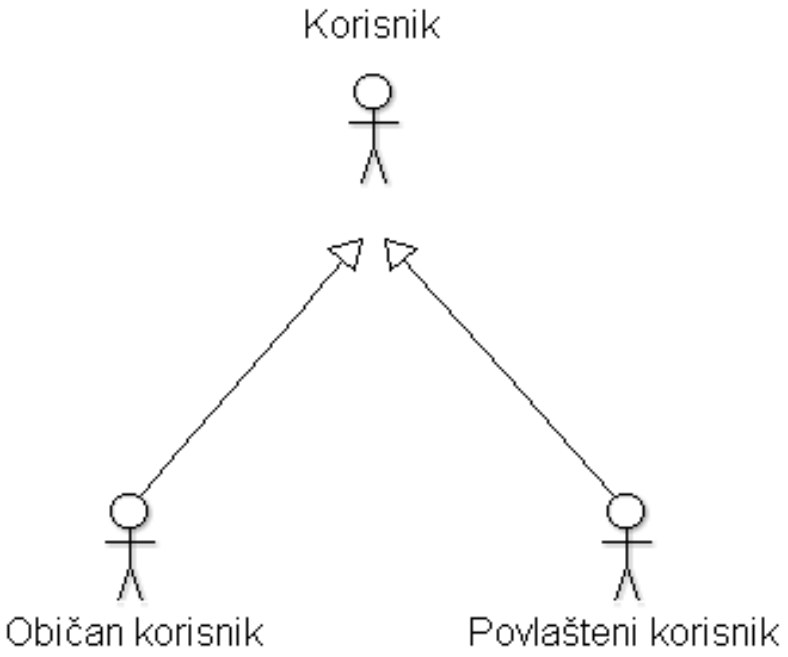
Zagreb, lipanj 2013.

Sadržaj

1. Uvod

Internetski portal mora primarno omogućiti praćenje raznih procesnih veličina, uz to treba omogućiti više vrsta korisnika (običan korisnik i povlašteni korisnik), odnosno uloga s različitim pravima i načinima korištenja portala. Također se sve mjerne veličine moraju arhivirati i, ovisno o pravima korisnika, biti pretražive.

Svaki korisnik se mora prvo registrirati na portalu kako bi se prijavio na isti i koristio ga. Običan korisnik nakon prijave na portal može vidjeti samo posljednjih nekoliko mjerenja, dok povlašteni korisnik može pretraživati sve arhivirane procesne veličine i ima mogućnost upravljanja korisnicima (dodaj, izbriši, promjena prava korisnika).

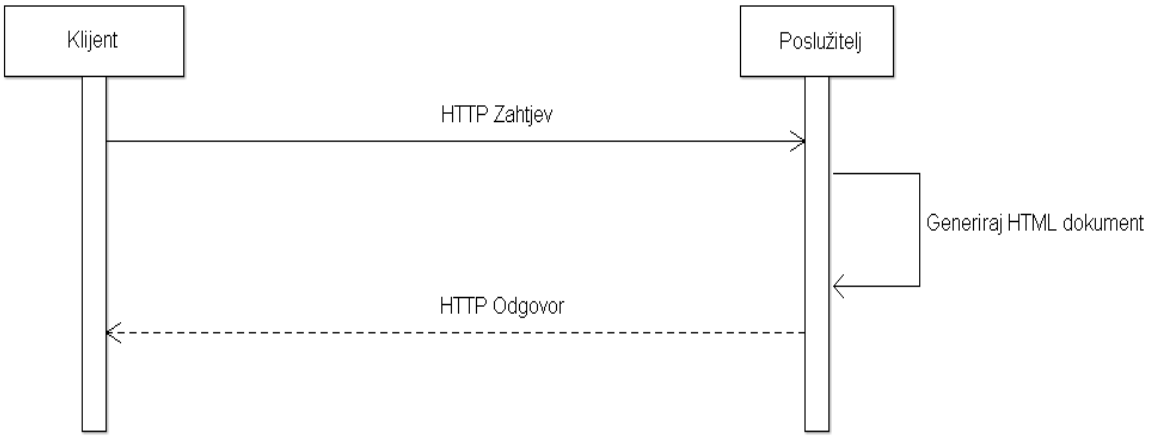


Slika 1. 1. Korisnici

Portal će biti izgrađen na ASP.NET platformi, u jeziku C# i u skladu sa MVC programskom arhitekturom.

2. ASP.NET

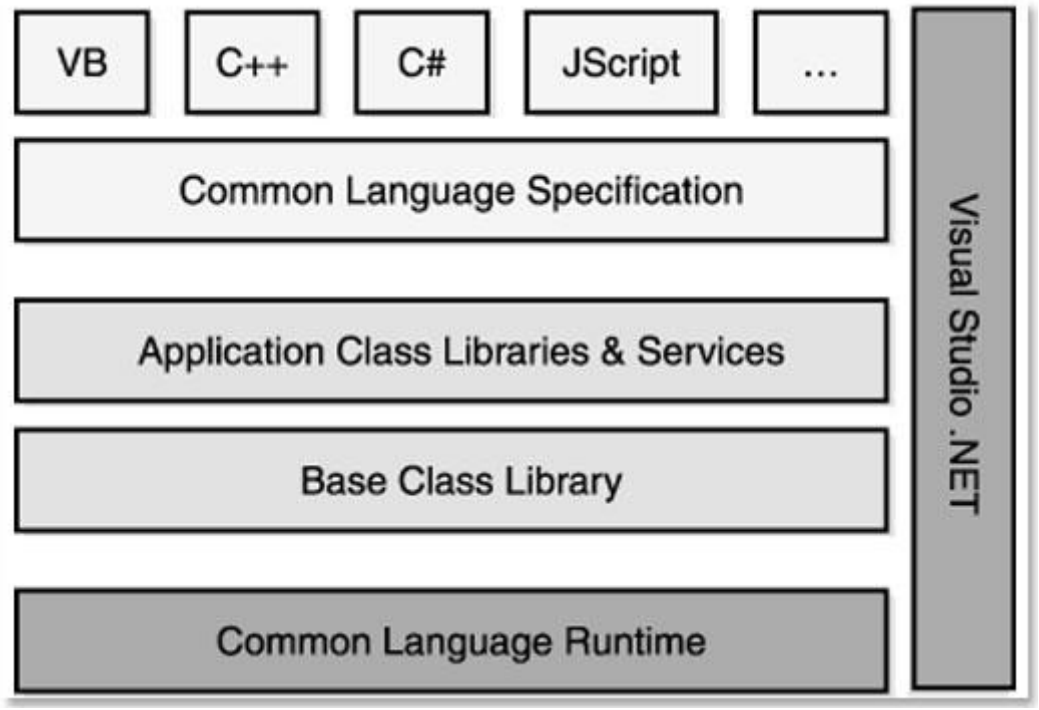
ASP.NET je razvojna platforma za razvijanje web aplikacija (dinamičkih web stranica), prva verzija ASP.NET 1.0 je izašla u siječnju 2002. godine zamjenjujući klasični ASP kao Microsoftov poslužiteljski skriptni jezik, trenutna stabilna verzija je ASP.NET 4.5. Dohvat dinamičke web stranice prikazan je na slici 2. 1., posljužitelj nakon zaprimljenog HTTP zahtjeva, umjesto vraćanja statičkog HTML dokumenta, dinamički generira traženi HTML dokument koji vraća u obliku HTTP odgovora, web aplikacija naravno može sadržavati i statičke web stranice koje nisu podložne interakciji sa strane klijenta.



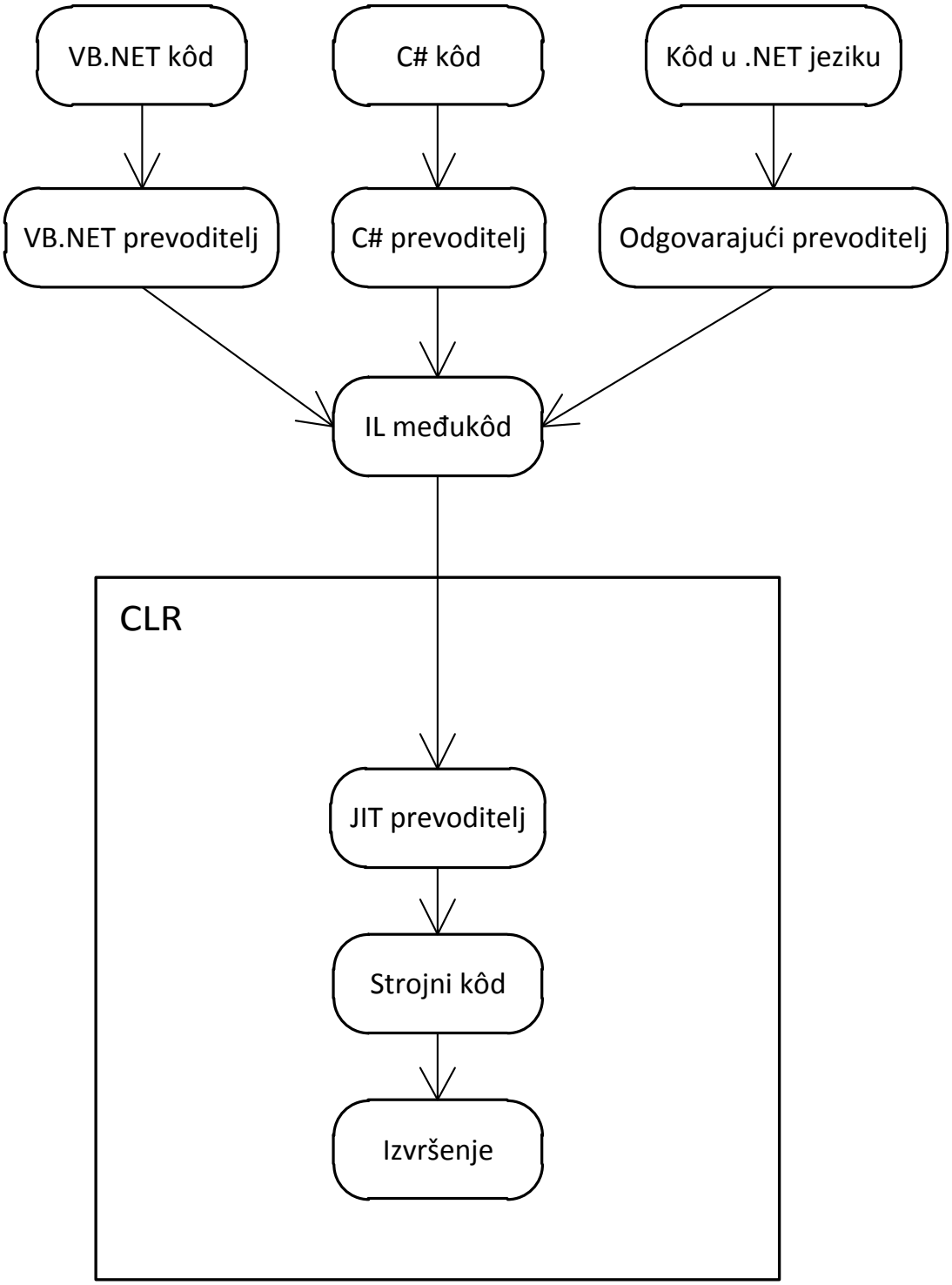
Slika 2. 1. Dohvat dinamičke web stranice

Više je razloga za prelazak na novu platformu, tj. sa ASP-a na ASP.NET, među kojima su najbitniji brzina i performanse. ASP skripte su se stvarale u skriptnim jezicija, kao npr. VBScript, koji nisu bili namjenjeni za web okruženje rezultirajući lošim performansama i brzinom. Loša brzina rezultat je interpretiranja kôda korištenog skriptnog jezika, a još jedna nuspojava korištenja skriptnih jezika je loše upravljanje memorijom što narušava performanse aplikacije. Dodatni razlozi za promjenu platforme su nedostatak korisnih „debugging“ alata kao i neupotrebljivost stare platforme za bilo kakve kompleksnije zadatake.

Najveća prednost ASP.NET-a je što se kôd prevodi, poboljšavajući brzinu. Kako je baziran na .NET frameworku (*Slika 2. 2.*) omugućava razvijanje web aplikacija na način sličan razvijanju desktop aplikacija sa svim mogućnostima objektno orijentiranog programiranja. Još jedna blagodat .NET frameworka je mogućnost razvoja u bilo kojem .NET jeziku (*Slika 2.3.*). Prevođenje se odvija u dvije faze, prva faza se sastoji od prevođenja, pomoću odgovarajućeg prevoditelja, u međujezik IL (*Microsoft Intermediate Language*) koji se izvodi na CLR-u (*Common Language Runtime*). CLR je kompleksno okruženje koje obavlja mnoge korisne zadatake: sakupljanje smeća, automatsko upravljanje memorijom, višedretvenost i slično. Druga faza prevođenja se odvija u trenutku obavljanja aplikacije. Tada se IL prevodi u strojni kôd, a ova faza poznata je kao i JIT prevođenje (*„just-in-time“).* JIT prevođenje se ne odvija pri svakom obavljanju, npr. svakom zatraživanju web stranice (što i nebi imalo smisla), već samo kod prvog obavljanja i kod svake promjene u kôdu.



Slika 2.2.Struktura .NET frameworka (Literatura 1.)



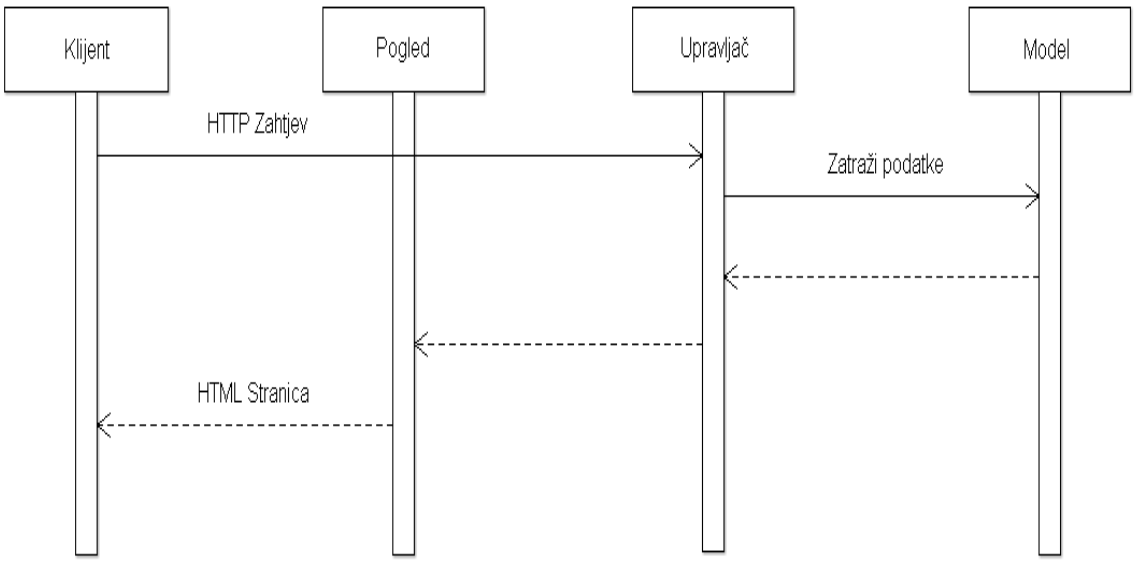
Slika 2. 3. Proces prevođenja

3. MVC

3. 1. Uvod

MVC (*Model-View-Controller*) je obrazac programske arhitekture široko prihvaćen za razvoj web aplikacija. Prednost MVC-a je što odvaja bilo kakav prikaz informacija od korisničke interakcije s istom, te omogućava nezavisan razvoj, testiranje i održavanje aplikacije. Sastavni dijelovi MVC-a su:

- Model (*Model*)
- Upravljač (*Controller*)
- Pogled (*View*)



Slika 3. 1. 1. Dohvat stranice u MVC arhitekturi

3. 2. Model

Model objedinjuje poslovnu logiku i sloj pristupa podacima. U aplikaciji postoje dva modela podataka, za korisnike i za mjerenja. Model za korisnike je automatski izgeneriran sa strane predloška MVC aplikacije. Izgenerirani model za korisnike sastoji se od više podmodela. Svaki podmodel predstavlja podskup svih

podataka korisnika. Podmodeli se koriste jer svi podaci korisnika nisu uvjek potrebni. Podmodeli za korisnika su:

- UserProfile: općeniti model korisnika sa općim podacima
- LocalPasswordModel: koristi se kôd izmjene lozinke korisnika
- LoginModel: koristi se kôd prijave korisnika
- RegisterModel: koristi se kôd registracije na portal

3. 3. Pogled

Pogledi definiraju izgled korisničkog sučelja, prikaz informacija, odnosno traženu web stranicu. Pogledi su HTML dokumenti koji sadrže skripte, u aplikaciji skripte su pisane u C# s *RAZOR view engine-om*. Glavna razlika između RAZOR-a i ASPX načina označavanja je što RAZOR koristi znak „@“ kako bi se označio blok C# kôda, za razliku od APSX-a gdje se koriste „<%=“ za otvaranje, a „%>“ za zatvaranje bloka. U RAZOR-u blok kôda nije potrebno eksplicitno zatvarati što daje programeru veću slobodu i fleksibilnost pri pisanju kôda.

Svi pogledi se prikazuju unutar zajedničkog predloška (*Layout*) kako bi aplikacija imala homogeni izgled kroz sve stranice aplikacije. Tablica 3. 3. 1. prikazuje sve poglede, odnosno dohvatljive web stranice.

Slika 3. 3. 1. Zajednički predložak (Layout)

Predložak sadrži naslov aplikacije, logo LARES-a i FER-a koji su ujedno i poveznice na njihove web stranice. Na desnoj strani predloška nalazi se izbornik za navigaciju koji sadrži poveznice *Home*, poveznica na početnu stranicu, *Users,*

poveznica na stranicu za upravljanje korisnicima (ova poveznica je vidljiva samo povlaštenom korisniku), *About,* poveznica na stranicu s općim informacijama, i *Contact,* poveznicu na stranicu s kontaktnim informacijama. Iznad izbornika za navigaciju se nalaze jos poveznice na stranicu za uređivanje korisničkog računa (*Hello, Korisničko ime!*) i za odjavu korisnika (*Log off*).

Pogled	Opis pogleda
Login	Prijava korisnika
Manage	Upravljanje korisničkim računom
Register	Registracija korisnika
Users	Upravljanje korisnicima (samo povlašteni korisnici)
About	Opće informacije
Chart	Grafički prikaz mjerenja
Contact	Kontakt informacije
Edit	Uređivanje mjerenja (samo povlašteni korisnici)
Index	Tablični prikaz mjerenja

Tablica 3. 3. 1. Popis svih pogleda s kratkim opisom

3. 4. Upravljač

Upravljač sadrži prezentacijsku logiku, obrađuje sve unose korisnika, obavlja izmjenu podataka s modelom i određuje pogled. Upravljač sadrži više akcija koje se izvode ovisno o unosu korisnika. Aplikacija sadrži dva upravljača:

- *Account:* sadrži sve akcije vezane uz korisnike
- *Home:* opće stranice portala *(Contact, About)* kao i sve akcije vezane uz prikaz mjerenja

Unos korisnika određuju tri elementa:

- Naziv upravljača
- Naziv akcije
- HTTP metoda

Naziv upravljača određuje koji upravljač obrađuje korisnikov unos, naziv akcije određuje akciju upravljača koja će se izvesti, dok HTTP metoda određuje jel klijent zahtijeva podatake (GET metoda) ili šalje podatke (POST metoda). Tablice 3. 4. 3. i 3. 4. 4. prikazuju moguće korisničke unose kao kombinacije upravljača, akcije i HTTP metode. U tablici je vidljivo da dvije akcije vezane uz isti upravljač mogu imati jednako ime, ali se u tome slučaju moraju razlikovati po HTTP metodi. To je uobičajena praksa jer je često potrebno dohvatiti stranicu (GET metoda) koja sadrži neku formu (npr. stranica za prijavu korisnika), nakon ispunjavanja forme podatke je potrebno poslati (POST metoda).

Neke akcije su definirane samo GET metodom, takve akcije služe za dohvat stranica koje ne sadrže nikakvu formu za ispunjavanje podataka (npr. stranica za grafički prikaz mjerenja), valja napomenuti da je svaka akcija definirana GET metodom povezana sa istoimenim pogledom kojeg će prezentirati kao odgovor na zahtjev (*Tablica 3. 4. 1.*). Akcije definirane samo POST metodom predstavljaju akcije koje obavljaju određenu izmjenu podataka, ali im nisu potrebni podatci iz ranije dohvaćene forme (npr. brisanje korisnika). Svaka akcija definirana POST metodom, nakon obrade podataka, također rezultira vraćanjem pogleda korisniku (*Tablica 3. 4. 2.*).

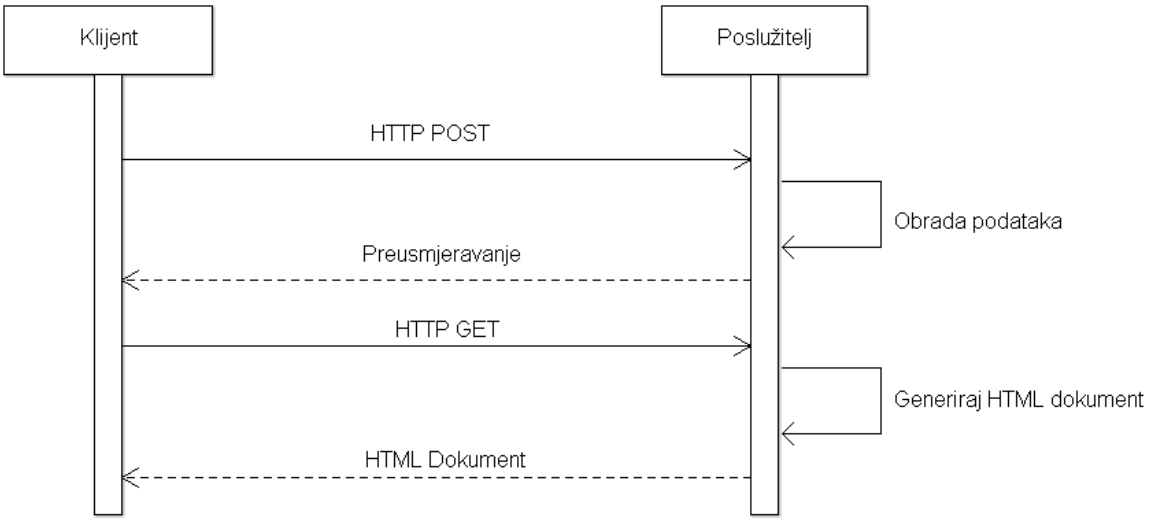
Upravljač	Akcija	Pogled
Account	Login	Login
	Register	Register
	Manage	Manage
	Users	Users
Home	Index	Index
	About	About
	Contact	Contact
	Edit/id	Edit
	Chart	Chart

Tablica 3. 4. 1. Akcije (GET metoda) s vezanim pogledom

Upravljač	Akcija	Pogled
Account	Login	Index
	LogOff	Login
	Register	Index
	Manage	Manage
	Delete/user	Users
	Change/id	Users
Home	Edit	Index
	Delete/id	Index

Tablica 3. 4. 2. Akcije (POST metoda) s vezanim pogledom

Iako akcije GET i POST metoda vraćaju poglede korisniku postoji bitna razlika, akcije s GET metodom vraćaju direktno pogled kao povratnu vrijednost, dok akcije s POST metodom pogled vraćaju indirektno preusmjeravajući korisnika na drugu akciju (definiranu GET metodom).



Slika 3. 4. 1. POST zahtjev

Upravljač	Akcija	HTTP Metoda	Opis
Account	Login	GET	Dohvat stranice za prijavu korisnika
	Login	POST	Prijava korisnika
	LogOff	POST	Odjava korisnika
	Register	GET	Dohvat stranice za registraciju korisnika
	Register	POST	Registracija korisnika
	Manage	GET	Dohvat stranice za uređivanje korisničkog računa
	Manage	POST	Unos promjena korisničkog računa
	Delete/user	POST	Brisanje odabranog korisničkog računa (samo povlašteni korisnici)
	Change/id	POST	Promjena korisničkih prava odabranog korisnika (samo povlašteni korisnici)
	Users	GET	Dohvat stranice za upravljanje korisnicima (samo povlašteni korisnici)

Tablica 3. 4. 3. Account upravljač s pridruženim akcijama, metodom i kratkim opisom

Upravljač	Akcija	HTTP Metoda	Opis
Home	Index	GET	Dohvat početne stranice nakon prijave korisnika sa tabličnim prikazom mjerenja
	About	GET	Dohvat stranice s općim informacijama
	Contact	GET	Dohvat stranice s kontaktnim informacijama
	Edit/id	GET	Dohvat stranice za uređivanje odabranog mjerenja (samo povlašteni korisnici)
	Edit	POST	Unos promjena mjerenja
	Delete/id	POST	Brisanje odabranog mjerenja (samo povlašteni korisnici)
	Chart	GET	Grafički prikaz mjerenja

Tablica 3. 4. 4. Home upravljač s pridruženim akcijama, metodom i kratkim opisom

4. Aplikacija

4. 1. Prijava korisnika

Stranica za prijavu korisnika na portal, dohvaća se GET zahtjevom */Account/Login* (Isječak 4. 1. 1.). Odgovarajuća akcija je predznačena atributom [*AllowAnonymous],* koji omogućava pristup stranici svim korisnicima, nije potrebna bilo kakva autorizacija, rezultat akcije je vraćanje pogleda *Login.*

```
// GET: /Account/Login

[AllowAnonymous]
public ActionResult Login(string returnUrl)
{
ViewBag.ReturnUrl = returnUrl;
return View();
}
```

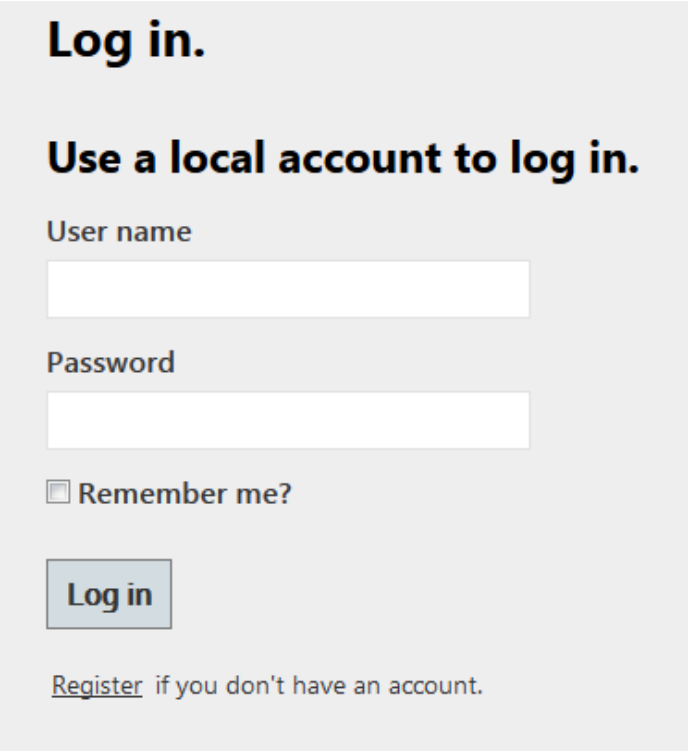
Isječak 4. 1. 1. Isječak iz AccountController.cs, GET: /Account/Login

Prije povrata pogleda u *ViewBag.ReturnUrl* sprema se znakovni niz *returnUrl* koji predstavlja URL stranice na koju će biti preusmjeren korisinik nakon uspješne prijave, za dohvat taj URL nije od posebne važnosti, ali će biti pri POST zahtjevu.

Stranica sadrži formu za unos podataka (Slika 4. 1. 1.), potrebnih za prijavu korisnika , potrebni podaci su:

- Korisničko ime (*User name*)
- Lozinka (*Password*)

Uz ključne podatake korisnika na formi se nalazi i *„Zapamti me?“ CheckBox (Remember me?).* Ukoliko se označi aplikacija će zapamtiti korisnika pa ne će biti potrebe za ponovnom prijavom korisnika. Osim forme za unos podataka bitni dio stranice je i poveznica za registraciju korisnika (*Register*) koja se nalazi odmah ispod forme. Poveznica preusmjerava korisnika na stranicu gdje može stvoriti novi korisnički račun.



Slika 4. 1. 1. Izgled forme za prijavu korisnika

Nakon ispunjavanja podataka, pritiskom na gumb „Log in“ šalje se POST zahtjev */Account/Login(Isječak 4. 1. 3.).* Odgovarajuća akcija je predznačena *[HttpPost]* atributom koji označava da se akcija poziva POST zahtjevom. Predznačena je i *[ValidateAntiForgeryToken]* atributom koji služi za zaštitu od krivotvorenja zahtjeva (ostali atributi objašnjeni su raniju u radu).

```
// POST: /Account/Login

[HttpPost]
[AllowAnonymous]
[ValidateAntiForgeryToken]
public ActionResult Login(LoginModel model, string returnUrl)
{
	if (ModelState.IsValid && WebSecurity.Login(model.UserName,
model.Password, persistCookie: model.RememberMe))
	{
		return RedirectToLocal(returnUrl);
	}

	// If we got this far, something failed, redisplay form
	ModelState.AddModelError("", "The user name or password provided is
incorrect.");
	return View(model);
}
```

Isječak 4. 1. 3. Isječak iz AccountController.cs, POST: /Account/Login

Akcija prvo provjerava valjanost modela, kako se radi o *LoginModel*-u vrši se provjera korisničkog imena i lozinke. U definiciji modela (*Isječak 4. 1. 4.*) su navedena polja pedznačena s atributom *[Reqired].*

```
public class LoginModel
{
        [Required]
        [Display(Name = "User name")]
        public string UserName { get; set; }

        [Required]
        [DataType(DataType.Password)]
        [Display(Name = "Password")]
        public string Password { get; set; }

        [Display(Name = "Remember me?")]
        public bool RememberMe { get; set; }
    }
```

Isječak 4. 1. 4. Isječak iz AccountModels.cs, LoginModel

Tim atributom se osigurava da navedena polja u formi budu ispunjena, ukoliko korisnik ostavi prazna polja bit će obaviješten odgovarajućim porukama. Osim valjanosti modela akcija provjerava i uspješnostprijave korisnika. Ukoliko model nije valjan ili ako prijava korisnika nije bila uspješna, korisnika se vraća na stranicu za prijavu korisnika uz odgovarajuće poruke zašto prijava nije bila uspješna. Ako je model valjan i prijava uspješna korisnika se preusmjerava na *returnUrl* ukoliko je isti lokalni URL aplikacije, ako nije lokalni URL aplikacije korisnika se preusmjerava na */Home/Index* (o tome brine funkcija *RedirectToLocal*).

Znakovni niz *returnUrl* je obično jednak URL-u zahtjevane stranice za koju korisnik nema potrebna prava pristupa, nakon takvog zahtjeva se korisnika preusmjerava na stranicu za prijavu korisnika gdje se može prijaviti s korisničkim računom dovoljnih prava. Uobičajena vrijednost *returnUrl*-a je */Home/Index* jer je to početna stranica aplikacije, a za koju je potrebno da je korisnik prijavljen. Stoga ukoliko korisnik nije prijavljen preusmjerava se na stranicu za prijavu korisnika gdje se može prijaviti važećim korisničkim računom.

4. 2. Registracija korisnika

Stranica za registraciju korisnika na portal, odnosno stvaranje novog korisničkog računa, dohvaća se GET zahtjevom */Account/Register,* a jedini zadatak odgovarajuće akcije (*Isječak 4. 2. 1.*) je prikaz pogleda *„Register“.* Stranica za registraciju je naravno javno dostupna kako bi svi posjetitelji aplikacije mogli stvoriti korisnički račun.

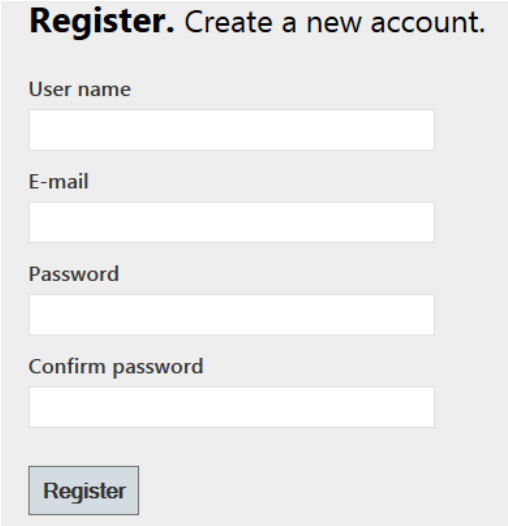
```
// GET: /Account/Register

[AllowAnonymous]
public ActionResult Register()
{
        return View();
}
```

Isječak 4. 2. 1. Isječak iz AccountController.cs, GET: /Account/Register

Stranica sadrži formu za unos podataka (*Slika 4. 2. 1.*) potrebnih za stvaranje novog korisničkog računa, traženi podaci su:

- Korisničko ime (*User name*)
- E-mail
- Lozinka (*Password*)
- Potvrda lozinke (*Confirm password*)

Konkretni izgled stranice ovisi o vrsti korisnika, razlika je u nekoliko vizualnih detalja, struktura forme je jednaka. Stranica se moze dohvatiti od strane posjetitelja ili običnog korisnika koji si želi otvoriti novi korisnički račun za aplikaciju ili od strane povlaštenog korisnika koji stvara novog korisnika, razlika je u značenju, posjetitelj ili običan korisnik stvara korisnički račun za sebe, dok povlašteni korisnik stvara korisnički račun za drugoga.



Slika 4. 2. 1. Izgled forme za registraciju korisnika

Ispunjavanjem potrebnih podataka i pritiskom na gumb *„Register“,* ili *„Create“* ako se stranica poziva kao povlašteni korisnik, šalje se POST zahtjev */Account/Register.*

```
// POST: /Account/Register

[HttpPost]
[AllowAnonymous]
[ValidateAntiForgeryToken]
public ActionResult Register(RegisterModel model)
{
       if (ModelState.IsValid)
       {
              try
              {
                     WebSecurity.CreateUserAndAccount(model.UserName,
model.Password, new { Email = model.Email });
                     Roles.AddUserToRole(model.UserName, "Regular");
                     if (User.IsInRole("Admin"))
                     {
                         return RedirectToAction("Users", "Account");
                     }
                     WebSecurity.Login(model.UserName, model.Password);
                     return RedirectToAction("Index", "Home");
              }
              catch (MembershipCreateUserException e)
              {
                     ModelState.AddModelError("",
ErrorCodeToString(e.StatusCode));
              }
       }
       return View(model);
}
```

Isječak 4. 2. 2. Isječak iz AccountController.cs, POST: /Account/Register

Odgovarajuća akcija prvo provjerava valjanost modela. U akciji se koristi „*RegisterModel*“ model podataka (*Isječak 4. 2. 3.*). Model zahtjeva da korisničko ime bude ispunjeno, e-mail bude ispunjen i odgovarajućeg formata, lozinka bude ispunjena i da ima minimalnu dužinu od 6 znakova, te da potvrda lozinke odgovara zadanoj lozinci. Ukoliko ispunjeni podaci korisnika ne odgovaraju zahtjevima modela korisnik će biti obaviješćen odgovarajućom greškom o ukrivom unosu. Ukoliko je model valjan pokašava se stvoriti korisnički račun (*WebSecurity.CreateUserAndAccount*), ako dođe do greške pri stvaranju računa greška se obrađuje i korisnika se obaviješćuje odgovarajućom porukom, obično poruka o već zauzetom korisničkom imenu. Nakon uspješnog stvaranja korisničkog računa provjerava se jeli korisnički račun stvorio povlašteni korisnik, ako je povlaštenog korisnika se preusmjerava natrag na stranicu za upravljanje korisnicima, */Account/Users.* Ako nije riječ o povlaštenom korisniku novog registriranog korisnika se prijavljuje te preusmjerava na početnu stranicu, */Home/Index.*

```
public class RegisterModel
    {
        [Required]
        [Display(Name = "User name")]
        public string UserName { get; set; }

        [Required]
        [Display(Name = "E-mail")]
        [RegularExpression(@"^([\w\.\-]+)@([\w\-
]+)((\.(\w){2,3})+)$",ErrorMessage="Invalid Email.")]
        public string Email { get; set; }

        [Required]
        [StringLength(100, ErrorMessage = "The {0} must be at least {2}
characters long.", MinimumLength = 6)]
        [DataType(DataType.Password)]
        [Display(Name = "Password")]
        public string Password { get; set; }

        [DataType(DataType.Password)]
        [Display(Name = "Confirm password")]
        [Compare("Password", ErrorMessage = "The password and confirmation
password do not match.")]
        public string ConfirmPassword { get; set; }
    }
```

Isječak 4. 2. 3. Isječak iz AccountModels.cs, RegisterModel

4. 3. Uređivanje korisničkog računa

Stranica za uređivanje korisničkog računa, odnosno promjena lozinka i brisanje računa, dohvaća se GET zahtjevom */Account/Manage.* Odgovarajuća akcija (*isječak 4. 3. 1.)* zahtjeva da je korisnik prijavljen na portal (atribut *[Authorize]*), ukoliko korisnik nije prijavljen preusmjerit će se na */Account/Login* gdje se može prijaviti s valjanim korisničkim računom .

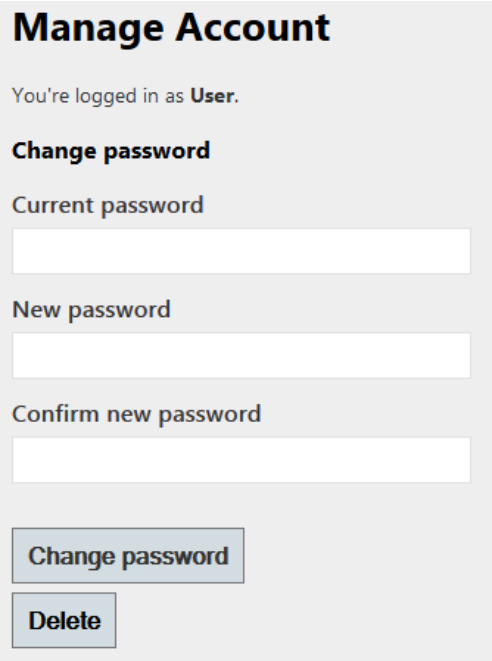
```
// GET: /Account/Manage
[Authorize]
public ActionResult Manage(ManageMessageId? message)
{
       ViewBag.StatusMessage =
       message == ManageMessageId.ChangePasswordSuccess ? "Your
password has been changed." : "";

       return View();
}
```

Isječak 4. 3. 1. Isječak iz AccountController.cs, GET: /Account/Manage

Akcija postavlja statusnu poruku (*ViewBag.StatusMessage*) koja će u slučaju da je korisnik već uspješno promjenio lozinku sadržavati poruku o uspješnosti promjene, poruka će biti prikazana u pogledu, ukoliko nije bilo promjene lozinke poruka će biti jednaka praznom nizu. Nakon postavljanja statusne poruke vraća pogled „*Manage".* Stranica sadrži formu za uređivanje korisničkog računa (*Slika 4. 3. 1.*), odnosno promjenu lozinke, potrebni podaci za promjenu lozinke su:

- Trenutna lozinka (*Current password*)
- Nova lozinka (*New password*)
- Potvrda nove lozinke (*Confirm new password*)

Osim forme stranica sadži i gumb „*Delete*" koji pokreće brisanje trenutnog korisnika.



Slika 4. 3. 1. Izgled forme za uređivanje korisničkog računa

Ispunjavanjem potrebnih podataka u formi i pritiskom gumba „*Change password*“ šalje se POST zahtjev */Account/Manage.* Odgovarajuća akcija (*Isječak 4. 3. 2.*) prvo provjerava valjanost modela podataka, koristi „*LocalPasswordModel“* model podataka (*Isječak 4. 3. 3.*). Model zahtjeva da trenutna lozinka bude ispunjena, nova lozinka bude ispunjena i da ima minimalnu duljinu od 6 znakova, te da potvrda lozinke odgovara novoj lozinci. Ukoliko neki od zahtjeva modela nije ispunjen korisnik će porukom biti obaviješćen o grešci. Ukoliko je model valjan pokušava se promjeniti lozinka u skladu s ispunjenim podacima (*WebSecurity.ChangePassword*). Ako je promjena lozinke neuspješna korisnika se obaviještava porukom o grešci, obično poruka o neispravnoj trenutnoj lozinci. Ako promjena lozinke bude uspješna korisnika se preusmjerava na */Account/Manage* gdje će biti prikazana poruka o uspješnosti promjene.

```
// POST: /Account/Manage

[HttpPost]
[ValidateAntiForgeryToken]
public ActionResult Manage(LocalPasswordModel model)
{
       if (ModelState.IsValid)
       {
              bool changePasswordSucceeded;
              try
              {
                     changePasswordSucceeded =
WebSecurity.ChangePassword(User.Identity.Name, model.OldPassword,
model.NewPassword);
              }
              catch (Exception)
              {
                     changePasswordSucceeded = false;
              }
              if (changePasswordSucceeded)
              {
                     return RedirectToAction("Manage", new { Message =
ManageMessageId.ChangePasswordSuccess });
              }
              else
              {
                     ModelState.AddModelError("", "The current password is
incorrect or the new password is invalid.");
              }
       }
       return View(model);
}
```

Isječak 4. 3. 2. Isječak iz AccountController.cs, POST: /Account/Manage

```
public class LocalPasswordModel
    {
        [Required]
        [DataType(DataType.Password)]
        [Display(Name = "Current password")]
        public string OldPassword { get; set; }

        [Required]
        [StringLength(100, ErrorMessage = "The {0} must be at least {2}
characters long.", MinimumLength = 6)]
        [DataType(DataType.Password)]
        [Display(Name = "New password")]
        public string NewPassword { get; set; }

        [DataType(DataType.Password)]
        [Display(Name = "Confirm new password")]
        [Compare("NewPassword", ErrorMessage = "The new password and
confirmation password do not match.")]
        public string ConfirmPassword { get; set; }
    }
```

Isječak 4. 3. 3. Isječak iz AccountModels.cs, LocalPasswordModel

Pritiskom na gumb „*Delete*“ pokreće se brisanje korisnika slanjem POST zahtjeva */Account/Delete/user.* Akcija za brisanje korisnika (*Isječak 4. 3. 4.*) služi kako za brisanje trenutnog korisnika tako i za brisanje drugog korisnika (povlašteni korisnik može brisati druge korisnike sa stranice za upravljanje korisnicima). Akcija prima kao parametar korisničko ime korisnika odabranog za brisanje, u slučaju brisanja trenutnog korisnika parametar je jednak korisničkom imenu trenutnog korisnika. Prvo se obavlja provjera jeli korisnik koji je poslao zahtjev ovlašten za brisanje odabranog korisnika, provjera se vrši na način da korisnik mora biti ili povlašteni korisnik ili jednak korisniku odabranom za brisanje. Nakon uspješne provjere briše se odabrani korisnik, nakon toga slijedi provjera jeli obrisan vlastiti ili tuđi korisnički račun. Uslučaju da je obrisan vlastiti korisnički račun korisnika se odjavljuje s aplikacije (*WebSecurity.Logout*) i preusmjerava na */Account/Login*, a ukoliko je izbrisan tuđi korisnički račun sa strane povlaštenog korisnika vrši se preusmjeravanje na */Account/Users,* tj. stranicu za upravljanje korisnicima.

```
// POST: /Account/Delete/user
        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Delete(string user)
        {
            //Security
            if (User.IsInRole("Admin") || User.Identity.Name==user)
            {
                string[] roles = Roles.GetRolesForUser(user);
                Roles.RemoveUserFromRoles(user, roles);
                Membership.DeleteUser(user);

                //Deleted self
                if (User.Identity.Name == user)
                {
                    WebSecurity.Logout();
                    return RedirectToAction("Login");
                }
                //Deleted other
                else
                {
                    return RedirectToAction("Users", "Account");
                }
            }
            else
            {
                return RedirectToAction("Login");
            }
        }
```

Isječak 4. 3. 4. Isječak iz AccountController.cs, POST: /Account/Delete/user

4. 4. Upravljanje korisnicima

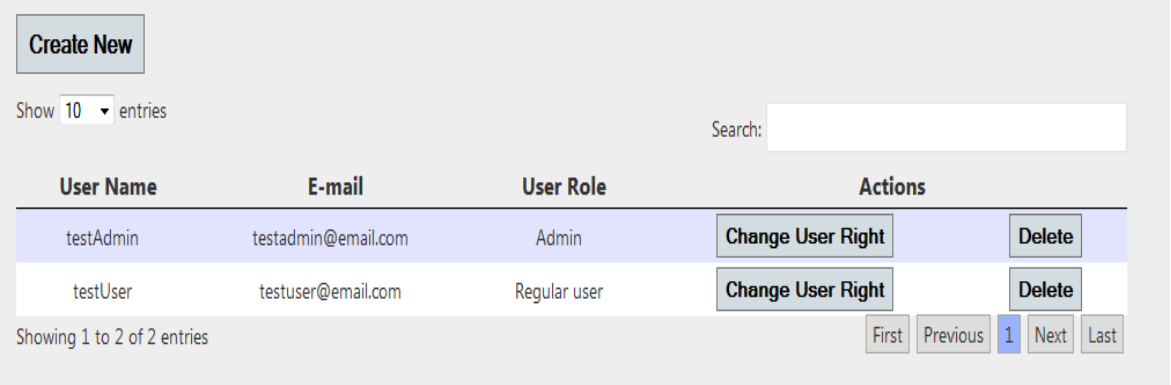
Stranica za upravljanje korisnicima je dostupna samo povlaštenima korisnicima, a služi za brisanje, dodavanje i promjenu prava korisnika. Dohvaća se GET zahtjevom */Account/Users.* Odogovarajuća akcija (*Isječak 4. 4. 1.*) osigurava dohvat stranice samo povlaštenim korisnicima, to konkretno osigurava atribut *[Authorize(Roles="Admin")]*. Akcija vraća pogled *Users* kojem predaje listu svih korisnika.

```
// GET: /Account/Users
[Authorize(Roles="Admin")]
public ActionResult Users()
{
        return View(db.UserProfiles.ToList());
}
```

Isječak 4. 4. 1. Isječak iz AccountController.cs, GET: /Account/Users

Stranica sadrži gumb „*Create New*“ za stvaranje novog korisnika, pretragu korisnika sa straničenjem i tablični prikaz osnovnih podataka korisnika sa gumbima za promjenu prava korisnika (*Change User Right*) i brisanje korisnika (*Delete*). Osnovni podaci korisnika su:

- Korisničko ime (*User Name*)
- E-mail
- Vrsta korisnika ili uloga (*User Role*)



Slika 4. 4. 1. Izgled prikaza korisnika

Odabirom gumba „*Create New*“ šalje se GET zahtjev */Account/Register* čime se korisnika preusmjerava na stranicu za registraciju (stvaranje) korisnika, a odabirom gumba „*Delete*“ šalje se POST zahtjev */Account/Delete/user* čime se briše odabrani korisnik. Posljednja funkcionalnost stranice za upravljanje korisnicima je promjena prava korisnika, pokreće se odabirom gumba „*Change User Right* “, čime se salje POST zahtjev */Account/Change/id.* Promjena prava se vrši na način da se pronađe korisnik s obzirom na šifru(*id*), zatim se provjerava jeli odabrani korisnik povlašteni korisnik, ako je mijenja se u običnog korisnika, a ako nije postaje povlašteni korisnik. Nakon uspješne promjene korisničkih prava korisnika se preusmjerava natrag na stranicu za upravljanje korisnicima.

```
// POST: /Account/Change/id
        [HttpPost]
        [ValidateAntiForgeryToken]
        [Authorize(Roles = "Admin")]
        public ActionResult Change(int id)
        {
            var user = db.UserProfiles.Find(id);
            if (Roles.IsUserInRole(user.UserName,"Admin"))
            {
                Roles.RemoveUserFromRole(user.UserName, "Admin");
                Roles.AddUserToRole(user.UserName, "Regular");
            }
            else
            {
                Roles.RemoveUserFromRole(user.UserName, "Regular");
                Roles.AddUserToRole(user.UserName, "Admin");
            }
            return RedirectToAction("Users", "Account");
        }
```

Isječak 4. 4. 2. Isječak iz AccountController.cs, POST: /Account/Change/id

4. 5. Početna stranica

Početna stranica aplikacije na koju se preusmjerava korisnika nakon uspješne prijave, a dohvaća se GET zahtjevom */Home/Index.* Za dohvat stranice je potrebno biti prijavljen na aplikaciju, to osigurava odgovarajuća akcija. Akcija (*Isječak 4. 5. 1.*) vraća pogled „*Index“* kojem predaje listu mjerenja koja će pogled prikazati. Broj mjerenja koja će akcija proslijediti pogledu ovisi o vrsti korisnika, ukoliko se radi o povlaštenom korisniku prosljeđuju se sva mjerenja, dok se kôd

običnog korisnika prosljeđuje samo određeni broj posljednjih mjerenja. Akcija dodatno i provjerava je li se dogodila greška pri pozivu odabiru vremenskog raspona kôd grafičkog prikaza mjerenja, ukoliko je poruka će se ispisati u pogledu.

```
// GET: /Home/Index
        [Authorize]
        public ActionResult Index(bool? error)
        {
             if (error!=null)
            {
                ViewBag.error = "Start must be before end";
            }
            short limit = 10;
            var mjerenja = GetMjerenja();
            mjerenja.Reverse();
            if (!User.IsInRole("Admin"))
            {
                return View(mjerenja.Take(limit));
            }
            return View(mjerenja);
        }
```

Isječak 4. 5. 1. Isječak iz HomeController.cs, GET: /Home/Index

Izgled i mogućnosti stranice ovise o vrsti prijavljenog korisnika. Početna stranica običnog korisnika sadrži samo prikaz određenog broja posljednjih mjerenja u tabličnom zapisu i gumb „*Chart*“ za grafički prikaz mjerenja iz tablice..

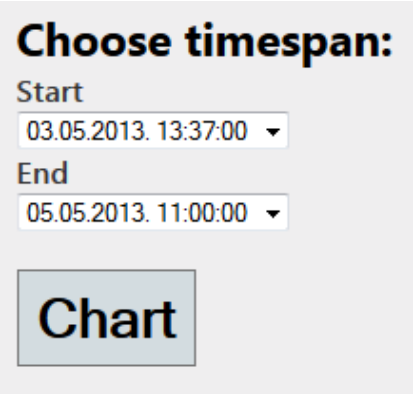
Mjerenje1	Mjerenje2	Mjerenje3	Mjerenje4	Mjerenje5	Mjerenje6	Mjerenje7	Mjerenje8	Vrijeme
1,28	1,613	2,28	2,2	2	1,099	1,35	0,995	05.05.2013. 11:00:00
1,25	1,6	2,19	2,19	2	1,111	1,34	0,899	05.05.2013. 10:58:00
1,24	1,556	2,16	2,14	2	1,116	1,345	0,889	05.05.2013. 09:34:00
1,23	1,599	2,09	2,12	1,99	1,119	1,35	0,8	05.05.2013. 08:12:00
1,22	1,632	2,17	2,099	1,998	1,123	1,35	0,9	05.05.2013. 07:18:00
1,23	1,644	2,18	2,08	1,999	1,126	1,35	1	05.05.2013. 05:55:00
1,24	1,667	2,2	2,045	2	1,132	1,36	1,1	05.05.2013. 04:56:00
1,23	1,767	2,15	1,99	2,022	1,087	1,37	1,28	05.05.2013. 00:00:00
1,24	1,799	2,08	2,12	2,01	1,09	1,37	1	04.05.2013. 20:30:00
1,3	1,809	3	2,13	2	1,11	1,37	1	04.05.2013. 14:59:00

Showing 1 to 10 of 10 entries

Chart

Slika 4. 5. 1. Izgled početne stranice za običnog korisnika

Početna stranica za povlaštenog korisnika također sadrži tablični prikaz mjerenja, s razlikom da se prikazuju sva mjerenja, formu za odabir vremenskog raspona mjerenja koja će biti prikazana u grafičkom obliku, stranica još sadrži straničenje, pretraživanje (pretraživanje se vrši po svim stupcima tablice), te mogućnosti uređivanja (gumb „*Edit*“) i brisanja (gumb „*Delete*“) pojedinog mjerenja.



Slika 4. 5. 2. Izgled forme za odabir vremenskog raspona

Odabirom gumba „*Chart*“ korisnika se preusmjerava na */Home/Chart*, tj. stranicu za grafički prikaz mjerenja unutra odabranog vremenskog raspona, odabirom gumba „*Edit*“ se korisnika preusmjerava na */Home/Edit/id*, odnosno stranicu za uređivanje mjerenja. Odabirom gumba „*Delere*“ pokreće se brisanje odabranog mjerenja slanjem POST zahtjeva */Home/Delete/id.* Odgovarajuća akcija pronalazi odabrano mjerenje po šifri (*id*), odabrano mjerenje se zatim briše, a korisnika se preusmjerava natrag na */Home/Index.*

```
// POST: /Home/Delete/id
        [HttpPost]
        [ValidateAntiForgeryToken]
        [Authorize(Roles = "Admin")]
        public ActionResult Delete(int id)
        {
            Mjerenje targetMjerenje = mjerenjaDB.Mjerenje.Find(id);
            mjerenjaDB.Mjerenje.Remove(targetMjerenje);
            mjerenjaDB.SaveChanges();
            return RedirectToAction("Index");
        }
```

Isječak 4. 5. 2. Isječak iz HomeController.cs, POST: /Home/Delete/id

4. 6. Uređivanje mjerenja

Stranica za uređivanje mjerenja dostupna je samo povlaštenim korisnicima za što brine odgovarajuća akcija (*Isječak 4. 6. 1.*). Dohvaća se GET zahtjevom */Home/Edit/id.* Odgovarajuća akcija pronalazi mjerenje po šifri(*id*) te se isto proslijeđuje pogledu gdje će vrijednosti odabranog mjerenja biti prikazane i omogućene za izmjenu.

```
// GET: /Home/Edit/id
        [Authorize(Roles = "Admin")]
        public ActionResult Edit(int id)
        {
            Mjerenje targetMjerenje = mjerenjaDB.Mjerenje.Find(id);

            return this.View(targetMjerenje);
        }
```

Isječak 4. 6. 1. Isječak iz HomeController.cs, GET: /Home/Edit/id

Stranica se sastoji od forme za uređivanje mjerenja. Forma se sastoji od tekstualnih polja za svaku mjerenu veličinu postavljenih na vrijednosti mjerenja prije uređivanja i gumb „*Save*“ za potvrdu kraja uređivanja.

```
// POST: /Home/Edit
        [HttpPost]
        [ValidateAntiForgeryToken]
        [Authorize(Roles = "Admin")]
        public ActionResult Edit(Mjerenje targetMjerenje)
        {
            if (ModelState.IsValid)
            {
                this.mjerenjaDB.Entry(targetMjerenje).State =
System.Data.EntityState.Modified;
                this.mjerenjaDB.SaveChanges();
                return this.RedirectToAction("Index");
            }
            return this.View(targetMjerenje);
        }
```

Isječak 4. 6. 2. Isječak iz HomeController.cs, POST: /Home/Edit

Nakon promjene željenih vrijednosti mjerenja, kraj uređivanja i spremanje uređenog mjerenja se obavlja odabirom gumba „*Save*“. Odabirom gumba „*Save*“ šalje se POST zahtjev */Home/Edit.* Odgovarajuća akcija (*Isječak 4. 6. 2.*) prvo provjerava je li model valjan, model će biti valjan ukoliko se u svakom tekstualnom polje nalazi znakovni niz koji se može pretvoriti u broj, u suprotnom model nije valjan te se korisnika obaviještava o krivom unosu. Ukoliko je model valjan uređivano mjerenje, koje akcija prima kao argument, se sprema sa novim, izmjenjenim podacima, a korisnikase preusmjerava natrag na */Home/Index.*

4. 7. Grafički prikaz

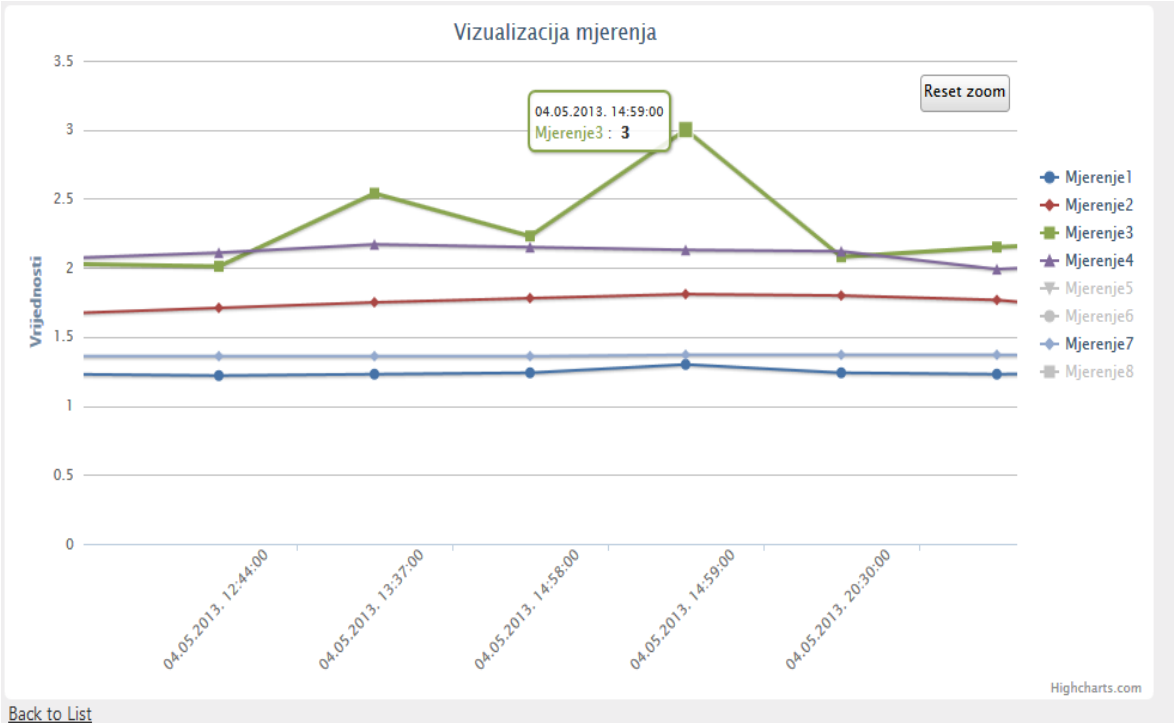
Stranica za grafički prikaz mjerenja, mjerenja se iscrtavaju u linijskom grafu gdje su vrijednosti X-osi vremena mjerenja, a vrijednosti Y-osi iznosi mjerenja. Sve veličine mjerenja se prikazuju na istom grafu. Dohvaća se GET zahtjevom */Home/Chart.* Odgovarajuća akcija (*Isječak 4. 7. 1.*) prvo obavlja provjeru vrste korisnika. Ako se radi o povlaštenom korisniku obavlja se provjera argumenta prvog mjerenja koje će se prikazati (*start*) i posljednjeg mjerenja koje će se prikazati (*end*), argumente *start* i *end* akciji prosljeđuje forma za odabir vremenskog raspona u pogledu *Index* za povlaštenog korisnika. Prvo se provjera jesu li argumenti nedefinirani, odnosno jednaki *null,* ukoliko jesu se postavljaju na prvo mjerenje, za *start,* odnosno zadnje mjerenje, za *end*. Nakon provjere definiranosti obavlja se provjera valjanosti, prvo mjerenje mora, po vremenu mjerenja, biti manje od zadnjeg mjerenja, u suprotnom se graf ne iscrtava, a korsinika se obavještava porukom o krivom odabiru raspona. Kad argumenti prođu sve provjere dohvaćaju se podaci za iscrtavanje. Ako je običan korisnik zatražio iscrtavanje grafa argumenti akcije se postavljaju tako da odgovaraju određenom broju posljednjih mjerenja, nakon postavljanja argumenta se dohvaćaju podaci za iscrtavanje. Nakon provjere vrste korisnika i dohvaćanja potrebnih podataka obavlja se inicijalizacija grafa (kôd izostavljen sa isječka 4. 7. 1.). Inicijalizacija se sastoji od postavljanja vrste grafa, dimenzija, naslova, podataka i raznih drugih značajki grafa. Inicijalizirani graf se prosljeđuje pogledu gdje će se i iscrtati.

```
// GET: /Home/Chart
        [Authorize]
        public ActionResult Chart(int? start, int? end)
        {
            //učitavanje podataka
            object[][] mjerenja;
            string[] categories;
            if (User.IsInRole("Admin"))
            {
                if (start == null)
                {
                    start = mjerenjaDB.Mjerenje.Select(a =>
a.MjerenjeID).Min();
                }
                if (end == null)
                {
                    end = mjerenjaDB.Mjerenje.Select(a => a.MjerenjeID).Max();
                }
                if (start > end)
                {
                    return RedirectToAction("Index", new {error=true });
                }
                categories = catData(start,end);
                mjerenja = chartData(start,end);
            }
            else
            {
                end = mjerenjaDB.Mjerenje.Select(a => a.MjerenjeID).Max();
                start =
mjerenjaDB.Mjerenje.ToArray().Reverse().Take(10).Select(a=>a.MjerenjeID).Min();
                categories = catData(start, end);
                mjerenja = chartData(start,end);
            }

            #region Highcharts line graph
            //inicijalizacija grafa
            #endregion

            return View(chart);
```

Isječak 4. 7. 1. Isječak iz HomeController.cs, GET: /Home/Chart

Graf je ostvaren pomoću *Highcharts* biblioteke. *Highcharts* biblioteka je biblioteka za izradu grafova napisana u HTML5/JavaScript-u i omogućuje izradu intuitivnih i interaktivnih grafova za web sranice i web aplikacije. Grafički prikaz se sastoji od X-osi, vremena mjerenja, Y-osi, iznosi mjerenih veličina, legende, popis mjerenih veličina, i samog grafa. Mjerene veličine možemo ukloniti ili dodati natrag na graf odabirom željene mjerne veličine u legendi. Detaljni prikaz jednog podatka se dobije pozicioniranjem pokazivača miša na željenu točku grafa, vrijeme mjerenja i iznos veličine se prikažu unutar pravokutinika pokraj odabrane točke. Moguće je i uvećeti dio grafa (*zoom*) na način da se željeni dio grafa označi klikom

i povlačenjem pokazivača miša (*click-drag select*). Nakon što se označeni dio grafa uveća pojavljuje se gumb „*Reset zoom*“ odabirom kojeg se poništavaju sva uvećanja grafa.



Slika 4. 7. 1. Primjer grafičkog prikaza mjerenja (graf)

4. 8. Automatsko dodavanje mjerenja

Automatsko dodavanje radi tako da sustav za mjerenje rezultate mjerenja zapisuje u datoteku koja će služiti kao međuspremnik, a aplikacija periodički provjerava datoteku, ukoliko ima novih mjerenja u datoteci aplikacija unosi nova mjerenja u bazu podataka. U trenutku izrade završnog rada zapisivanje mjernih rezultata u datoteku nije bilo funkcionalno, unatoč tome aplikacija sadrži ispravnu funkciju za automatski unos.

Unutar *Global.asax.cs* u funkciji *Application_Start* je instanciran pozadinski radnik(*BackgroundWorker),* nakon instanciranja se isti pokreće metodom *RunWorkerAsync.* Pozadinski radnik obavlja sav posao u pozadini paralelno s glavnim programom, ne ometajući korisnika u korištenju aplikacije.

```
private void bw_DoWork(object sender, DoWorkEventArgs e)
        {
            while (true)
            {
                using (var file = new System.IO.StreamReader("c:\\mjerenja.txt"))
                {
                    var mjerenjaDB = new Models.Database1Entities1();
                    while (file.EndOfStream != true)
                    {
                        var mjerenje = new Models.Mjerenje();
                        var line = file.ReadLine();
                        //obrada ucitane linije datoteke
                        //postavljanje podataka instance modela Mjerenje
                        //mjerenje.MjerenjeID = mjerenjaDB.Mjerenje
                                              //.Max(a => a.MjerenjeID) + 1;
                        //mjerenjaDB.Add(mjerenje);
                        //mjerenjaDB.SaveChanges();
                    }
                }
                using (var file= System.IO.File.Open("c:\\mjerenja.txt",
System.IO.FileMode.Truncate))
                { /*praznjenje datoteke*/}

                //cekanje do sljedece provjere (u milisekundama)
                System.Threading.Thread.Sleep(3600000);
            }
        }
```

Isječak 4. 8. 1. Isječak iz Global.asax.cs, funkcija bw_DoWork

Metoda *RunWorkerAsync* pokreće funkciju *bw_DoWork,* koja se nalazi u beskonačnoj petlji. Pomoću beskonačne petlje se ostvaruje cikličko provjeravanje datoteke uz vremenski interval čekanja između provjera. Funkcija otvara datoteku u koje se spremaju rezultati mjerenja i sve zapise sprema u bazu podataka. Konkretno spremanje treba dovršiti kada zapisivanje mjernih rezultata u datoteku bude funkcionalno. Potrebno će biti implementirati obradu pročitanog retka datoteke i postavljanje vrijednosti novog mjerenja sa pročitanim rezultatima mjerenja. Nakon spremanja mjerenja u bazu podaka funkcija prazni datoteku kako pri sljedećoj provjeri ne bi učitala opet ista mjerenja. Funkcija zatim čeka određeno vrijeme do ponovne provjere datoteke.

4. 9. Ostalo

Ostale mogućnosti aplikacije uključuju:

- dohvat stranice s općim informacija, GET */Home/About*
- dohvat stranica s kontaktnim informacijama, GET */Home/Contact*
- odjava korisnika s aplikacije, POST */Account/LogOff*

5. Zaključak

Razvijena aplikacija udovoljava zahtjevima zadatka. Omugućava više vrsta korisnika s različitim načinima uporabe aplikacije. Običnom korisniku je omogućen pregled samo posljednjih nekoliko mjerenja i grafički prikaz istih tih mjerenja. Povlaštenom korisniku je omogućeno pregledavanje i pretraživanje svih mjerenja i grafički prikaz mjerenja u odabranom vemeskom rasponu. Uz to povlašteni korisnik ima i sve administrativne funkcionalnosti za ostale korisnike.

Svi mjerni podaci će se automatski pohranjivati u bazu podataka kada sustav za zapisivanje mjerenja u datoteku bude funkcionalan. Tada će se funkcija za automatko dodavanje moći dovršiti, tj. implementirati dio funkcije za obradu pročitanog retka datoteke. Nije moguće taj dio implementirati ranije jer ovisi o formatu zapisa u datoteku. Automatsko dodavanje se vrši u pozadini aplikacije, te ne ometa korisnika u radu s aplikacijom.

Korištenjem MVC programske arhitekture omogućio se lagani razvoj aplikacije zbog podjele na modele, upravljače i poglede. Upravo tom podjelom se ostvaruje višeslojnost aplikacije i modularnost programskog kôda, a samim time se pospješuje i lakša nadogradnja aplikacije u budućnosti.

6. Literatura

1. Chetankumar Akarte, .NET Framework Overview, 2008.
   http://www.tipsntracks.com/6/dotnet-framework-overview.html
2. Ivica Glavaš, Dizajn WEB korisničkih sučelja, ASP.NET tehnologija, 2003.
   http://web.zpr.fer.hr/ergonomija/2003/glavas
3. Getting Started with ASP.NET MVC, 2013.
   http://www.asp.net/mvc
4. ASP.NET, 2013.
   https://en.wikipedia.org/wiki/ASP.NET
5. Model–view–controller, 2013.
   https://en.wikipedia.org/wiki/Model-view-controller
6. ASP.NET MVC 4, 2013.
   http://www.asp.net/mvc/mvc4
7. Highcharts JS, 2013.
   http://www.highcharts.com
8. Highcharts Options Reference, 2013.
   http://api.highcharts.com/highcharts

7. Sažetak i ključne riječi

INTERNETSKI PORTAL ZA PRAĆENJE PROCESNIH VELIČINA MIKROMREŽE

Ovaj rad predstavlja web aplikaciju za pregledavanje i spremanje raznih procesnih veličina. Aplikacija se zasniva na MVC programskoj arhitekturi, izgrađena je na ASP.NET platformi i napisana u C# jeziku. Glavni dio aplikacije je prezentacijska logika u upravljačima, *HomeController* i *AccountController,* koji sadrže akcije za sve korisničke unose, tj. HTTP zahtjeve (GET ili POST). Korisnički sustav je višerazinski s dvije vrste korisnika (povlašteni i obični), različitih korisničkih prava i mogućnosti. Aplikacija podržava tablični i grafički prikaz podaka. Tablični prikaz je pretraživ (za povlaštenog korisnika), a grafički prikaz interaktivan i lak za upotrebu. Svi mjerni podaci se čuvaju u bazi podataka, a novi podaci se dodaju automatski u pozadinskom procesu aplikacije.

Ključne riječi: ASP.NET, MVC, C#, web aplikacija

8. Abstract and keywords

INTERNET PORTAL FOR MONITORING PROCESS VARIABLES OF A MICROGRID

This paper presents a web application for viewing and storing various process values. The application is based on the MVC software arhitecture, built on the ASP.NET platform and written in the C# language. The main part of the application is the presentation login in the controllers, *HomeController* and *AccountController,* which contaion all actions for every user input, that is HTTP request (GET or POST). The user system is multileveled with two typres of users (privileged and regular), each with different user rights and possobilities. The application supports table and graphic view of data. The table view is serchable (for the privileged user) and the graphic view is interactive and user-friendly. All measurement data is stored in a database and new data is added automatically in a background process of the application.

Keywords: ASP.NET, MVC, C#, web application

11. Privitak

CD sadrži:

1. Završni rad (zavrsni_rad.docx)
2. Programski kôd (kôd.zip)