

<b>PREDMET:</b> Razvoj Programskih Rješenja	<b>NASTAVNI ANSAMBL</b>
<b>AK. GOD. :</b> 2017/2018	<b>PROFESOR:</b> Dženana Đonko (ddonko@etf.unsa.ba)
<b>RESURS:</b> Laboratorijska vježba 5	<b>ANSAMBL:</b> Hasić Haris, Hodžić Kerim
<b>DATUM OBJAVE:</b> 13.11.2017	Denašević Emir, Hasanbašić Denis, Mahmutović Mustafa, Ramić Benjamin

### Cilj vježbi:

- Upoznavanje i rad sa osnovnim Windows Forms aplikacijama
- Upoznavanje i rad sa osnovnim događajima (*events*) kontrola

**Napomena:** Prije izrade vježbe je obavezno pročitati predavanja vezana za tematiku vježbe i upoznati se sa osnovnim konceptima jezika C# i objektno orijentisanog programiranja.

## LABORATORIJSKA VJEŽBA 5

### Upoznavanje sa Windows Forms aplikacijama

#### Windows Forms

U prethodnim vježbama je bilo riječi o Windows Forms (skraćeno WinForms) skupu klâsa za razvoj Windows aplikacija korištenih kroz konzolnu aplikaciju. U ovoj vježbi će biti riječi o razvijanju takvih aplikacija kroz okruženje koje je u VS alatu namjenski razvijeno samo za njih, a to su WinForms aplikacije.

**Primjer 1, Postavka:** Potrebno je kreirati aplikaciju koja će omogućiti registraciju i menadžment korisnika bilo koje firme ili usluge sa interfejsom koji izgleda kao naprikazano na slici ispod.

Menadžment Korisnika

Unos

Ime:

Prezime:

Spol: ☐ Muško ☐ Žensko

Grad:

Br. Tel.:

Datum Rođenja: nedjelja, 06. novembar 2016.

Korisničko ime:

Lozinka:

☐ Administrator ☐ Moderator

Ponisti Unesi

Pretraga

Ime:

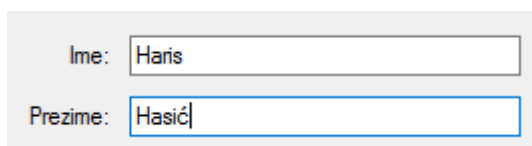
Brisanje

Učitaj Sliku...

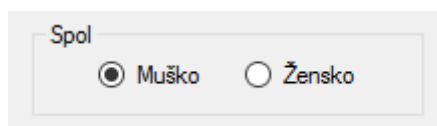
Detaljne informacije možete naći [ovdje](#).

Vidimo da se slika sastoji iz dosta različitih kontrola koja su pojašnjena na predavanjima i sa kojima su studenti do sada dobro upoznati. Pri izradi ovog primjera će primarni fokus biti pojašnjenje koje se kontrole koriste u kojim situacijama i koji se njihovi osnovni događaji koriste, a ne sama njihova struktura i detaljna analiza koja je objašnjena na predavanjima. Takođe je bitno napomenuti da je uz svaku kontrolu vezan i njen osnovni događaj, a kroz ovaj primjer će se ilustrovati upotreba tih događaja i pored toga će biti nabrojani i ostali korisni događaji vezani uz tu kontrolu.

Prva stvar koja je uočljiva jeste da je potreban unos imena i prezimena. Za to se koriste kombinacija **label**-a (koje se inače koriste da se prikaže nepromjenjivi tekst na formi) i **textBox**-ova (koji se primarno koriste da se unosi tekst).



Bitno je napomenuti da je osnovni događaj **textBox** kontrole **TextChanged** koji se dešava kada se mijenja tekst koji se nalazi unutar **textBox**-a, ali u ovom slučaju nam nije potrebna njegova upotreba. Sljedeća stvar jeste odabir pola korisnika. Prvo što se može uočiti da su za tu namjenu upotrebljene dvije **radioButton** kontrole. Naime, **radioButton** kontrole same po sebi ne sadrže informacije već uglavnom služe za odabir međusobno isključivih opcija, što pol i jeste po svojoj prirodi. Takođe je bitno uočiti da su oba **radioButton**-a smještena skupa unutar jednog **groupBox**-a. **GroupBox** kontrole obično služe da se formiraju logičke cjeline što je i urađeno na nivou cijele forme grupišući mjesto za Unos, Pretragu i Brisanje, ali se vrlo često koriste i za grupisanje **radioButton** kontrola jer u C# ove kontrole gledaju koja im je zajednička tzv. **Parent** kontrola i na osnovu toga zaključuju koje kontrole su logički grupisane i koje će se kontrole međusobno isključivati. Da nismo stavili **groupBox** parent kontrola za oba **radioButton**-a bi bila forma. Osnovni događaj **radioButton** kontrole jeste **CheckedChanged** događaj koji se dešava kada se promijeni stanje **radioButton**-a sa čekirano na nečekirano i to se može iskoristiti npr. za prikaz dodatnih informacija ili za upozorenje.



Iduća stvar jeste odabir grada. Iako bi se ova opcija mogla implementirati kao jednostavni unos preko **textBox** kontrole, ovdje je ipak odabrana **comboBox** kontrola. To je urađeno za minimizaciju greške jer ako se korisniku aplikacije ostavi mogućnost da unosi ovakav tip informacije moguće su greške pri unosu, ali pošto su ove informacije uglavnom statične i nisu promjenjive dobro ih je staviti u **comboBox** gdje korisnik može jednostavno odabrati jednu od ponuđenih opcija. Takođe je bitno paziti i na slučaj kada korisnik ne odabere nijedno od ponuđenih opcija. Osnovni događaj je **SelectedIndexChanged** i **SelectedItemChanged** koji predstavljaju promjenu odabranog indeksa predmeta ili odabranog predmeta iz **comboBox** kontrole respektivno.



Iduća stvar jeste broj telefona. Broj telefona je najlakše zapisati kao string, ali je problem format broja telefona. To se obično radi preko regularnih izraza (Regex), ali ako gledamo striktno preko kontrola ovdje dolazi do izražaja **maskedTextBox** kontrola koja se ponaša isto kao i obični **textBox**, ali sadrži mogućnost podešavanja maske koja forsira unos u određenom formatu. Moguće je odabrati predefinisani kalup ili kreirati svoj.

Input Mask

Select a predefined mask description from the list below or select Custom to define a custom mask.

Mask Description	Data Format	Validating Type
Numeric (5-digits)	12345	Int32
Phone number	(574) 555-0123	(none)
Phone number no area co...	555-0123	(none)
Short date	12/11/2003	DateTime
Short date and time (US)	12/11/2003 11:20	DateTime
Social security number	000-00-1234	(none)
Time (European/Military)	23:20	DateTime
Time (US)	11:20	DateTime
Zip Code	98052-6399	(none)
<Custom>		(none)

Mask: (999) 000-0000 ☒ Use ValidatingType

Preview: ( ) - -

OK Cancel

Kada se odabere maska, u taj *textBox* je moguće unijeti samo podatke u takvom formatu.

Br. Tel.: (387) 62-123456

Sljedeće na redu je unos datuma rođenja. Kao što je to već spomenuto na prethodnim vježbama u C# postoji specijalan tip za unos datuma koji se naziva *DateTime*. Shodno tome postoji i specijalna kontrola za unos datuma koja se naziva *dateTimePicker*. Ova kontrola je u obliku kalendara na kojem se odabira datum. Osnovni događaj je *ValueChanged* koji se pokreće kada se promijeni datum na ovoj kontroli.

Datum Rođenja: nedjelja , 06. novembar 2016.

Korisničko ime:

Lozinka:

novembar 2016.

pon	uto	sri	čet	pet	sub	ned
31	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	1	2	3	4
5	6	7	8	9	10	11

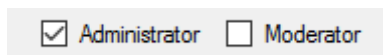
Today: 06.11.2016.

Informacije o korisničkom imenu i šifri/lozinki korisnika su takođe realizovane preko *textBox* kontrole s tom razlikom da je, radi veće sigurnosti, kod *textBox*-a za unos šifre podešena dodatna opcija kontrole *Password Character* koja maskira unos same šifre.

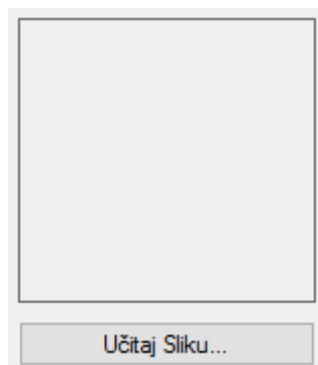
Korisničko ime: hharis

Lozinka: \*\*\*\*\*

Na kraju, kod unosa korisnika, implementirana je mogućnost određivanja da li je neki korisnik administrator, moderator ili nema nikakvu specijalnu funkciju. Pošto su to uključive opcije tj. može biti odabrano više opcija u isto vrijeme, one se realizuju preko kontrole koja se naziva **checkBox**, a ne preko **radioButton**-a kao ranije. Osnovni događaj je ovdje takođe **CheckedChanged** kao i kod **radioButton**-a.



U gornjem desno ćošku je postavljena **pictureBox** kontrola koja služi za prikazivanje slika na formi. Osnovni događaj **pictureBox** kontrole je **Click** koji se dogodi kada se klikne mišem.



Pošto slika koja će biti prikazana ovisi o samom korisniku, ova kontrola je na početku prazna, ali je ispod nje implementirana jedna **button** kontrola koja će služiti za učitavanje slike. Kako je osnovni događaj button kontrole **Click** koji se dešava kada se to dugme klikne, potrebno je na neki način omogućiti učitavanje slike kroz kôd. To se izvršava na način koji je prikazan ispod. Vidimo da smo koristili ključnu riječ **using** koja nam indicira da se radi o objektu koji nasljeđuje **IDisposable** interejs. To su obično privremeni objekti kao npr. *File*, *Font* ili u našem slučaju *FileDialog* koji nam trebaju samo na određeno vrijeme i onda postaju suvišni te se trebaju obrisati. U nastavku postavimo format fajlova koje je dozvoljeno učitavati i naznačimo da se učitani fajl učitava u naš **pictureBox**.

```
private void buttonUcitajSliku_Click(object sender, EventArgs e)
{
    using (OpenFileDialog dlg = new OpenFileDialog())
    {
        dlg.Title = "Izaberite sliku";
        dlg.Filter = "jpg files (*.jpg)|*.jpg";

        if (dlg.ShowDialog() == DialogResult.OK)
        {
            pictureBoxSlika.Image = new Bitmap(dlg.FileName);
        }
    }
}
```

U dnu logičke cjeline za unos korisnika su kreirane i dvije **button** kontrole za egzekuciju komandi i to za poništavanje dosadašnjeg unosa i za potvrdu i unos dosadašnjeg unosa.



Što se tiče, logičke cjeline za pretragu, tu se nalazi jedna label i jedna **textBox** kontrola koje su već opisani, ali i jedna **listBox** kontrola koja služi kao kontejner kolekcija. Naime, kontrole kao što su ranije spomenuti **comboBox**, u ovom slučaju **listBox**, **checkedListBox** ili **treeView** kontrola obično se koriste da na neki način prikazuju sve članove neke kolekcije. Zbog toga je

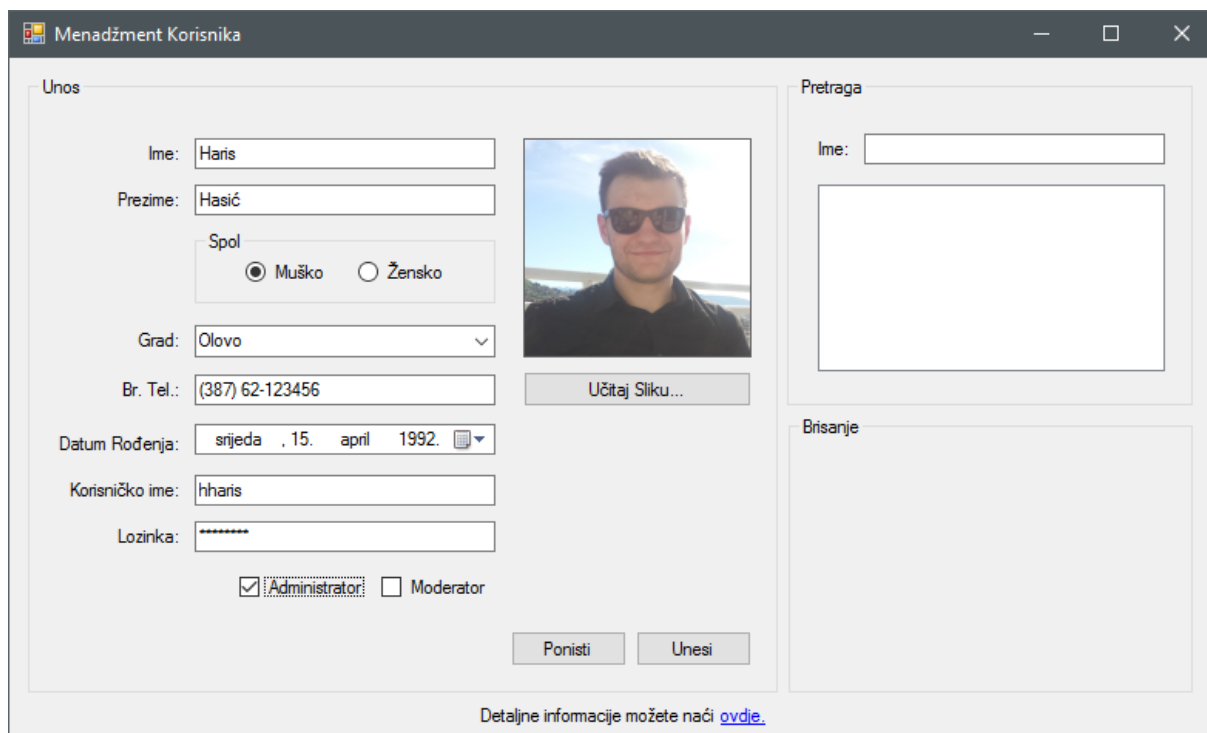
vrlo važno obratiti pažnju na preklapanje *toString()* metode ako su u pitanju objekti koje je kreirao korisnik.



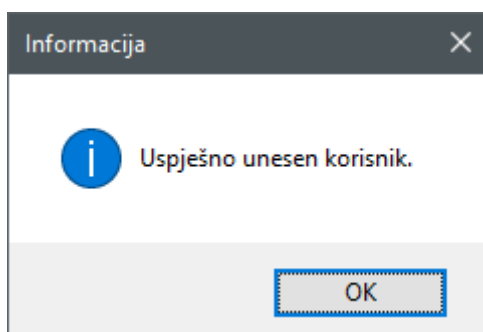
Nakon što korisnik unese ime u za to određeno polje, u *listBox* kontroli bi se trebali izlistavati svi objekti koji zadovoljavaju traženi uslov.

### ***Primjer 2, Izvršenje:***

Kada unesemo ispravno sve tražene informacije forma bi nam trebala izgledati kao što je to prikazano na slici ispod.



Kako je dobra praksa da se korisniku vrati određena povratna informacija kada je u interakciji sa formom, bitno je obavijestiti korisnika aplikacije da je njegov unos uspješno registrovan.



To se obično radi preko **MessageBox** kontrole koja služi za informisanje korisnika u određenim situacijama, a kako se koristi i kako se generalno kupe informacije i prenose u konkretan objekat je navedeno u nastavku.

```
private void buttonUnesi_Click(object sender, EventArgs e)
{
    Boolean musko = true;

    if (radioButtonMusko.Checked)
        musko = true;
    else
        musko = false;

    Korisnik k = new Korisnik(textBoxIme.Text, textBoxPrezime.Text, musko,
Convert.ToString(comboBoxGrad.SelectedItem), maskedTextBoxTelefon.Text,
dateTimePickerRodjenje.Value, textBoxUsername.Text, textBoxPass.Text,
pictureBoxSlika.Image, checkBoxAdmin.Checked, checkBoxModerator.Checked);

    korisnici.Add(k);

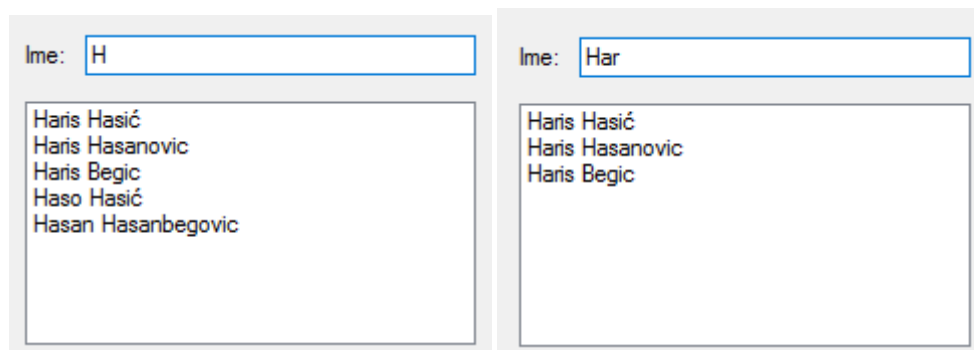
    MessageBox.Show("Uspješno unesen korisnik.", "Informacija",
MessageBoxButtons.OK, MessageBoxIcon.Information);

    pocistiInformacijeOUnosu();
}
```

Ono što se takođe može uočiti jeste da se poziva i metoda koja briše informacije o unosu. Veoma je bitno nakon završenog unosa kontrole restartovati na svoje početne vrijednosti, a to se radi na sljedeći način.

```
public void pocistiInformacijeOUnosu()
{
    textBoxIme.Text = "";
    textBoxPrezime.Text = "";
    radioButtonMusko.Checked = false;
    radioButtonZensko.Checked = false;
    comboBoxGrad.SelectedIndex = -1;
    maskedTextBoxTelefon.Text = "";
    dateTimePickerRodjenje.Value = DateTime.Now;
    textBoxUsername.Text = "";
    textBoxPass.Text = "";
    pictureBoxSlika.Image = null;
    checkBoxAdmin.Checked = false;
}
```

Što se tiče same pretrage, ona je zamišljena kao kombinacija textBox i listBox kontrola gdje bi se sve odvijalo dinamički tj. dok se unosi tekst, ažurira se i rezultat pretrage. To se može uraditi preko **textChanged** događaja, a izgled je prikazan na slici ispod.



The image displays two side-by-side screenshots of a Windows application interface. Both screenshots show a search form with a label 'Ime:' followed by a text input box. Below the input box is a list box containing five names: Haris Hasić, Haris Hasanovic, Haris Begic, Haso Hasić, and Hasan Hasanbegovic. In the left screenshot, the text input box contains the letter 'H'. In the right screenshot, the text input box contains the text 'Har'. The list box content is identical in both screenshots, suggesting a dynamic update of the list based on the input.

```
private void textBoxImePretraga_TextChanged(object sender, EventArgs e)
{
    listBoxPretraga.Items.Clear();

    foreach(Korisnik k in korisnici)
    {
        if ((k.Ime + " " + k.Prezime).Contains(textBoxImePretraga.Text))
        {
            listBoxPretraga.Items.Add(k);
        }
    }
}
```

Ono što je uočljivo jeste da se lista mora čistiti prije svakog upisa rezultata pretrage da se rezultat ne bi samo nadodavao nad već unesenim rezultatom.

**Veoma bitno:** Kada smo završili sve što se od nas zahtjevalo, možemo uočiti da je najbitnije za modelirane podatke i klase odabrati ispravne kontrole koje treba koristiti. Ako to odradimo ispravno, dosta posla smo riješili pri izgradnji samog interfejsa. Ipak, potrebno je posebno naglasiti, a i vidi se iz priloženog rješenja, da forma i sve što se dešava u događajima forme samo treba da se bavi prikupljanjem podataka, ubacivanjem istih kroz funkcije pozadinske logike (klasa), te ispisu dobijenih rezultata. Veoma je loša praksa da se ono što je usko vezano za klase, kao što su npr. validacije implementiraju u događajima, jer će klase ako se odvoje od prezentacijske logike patiti od mogućnosti pogrešnog unosa.

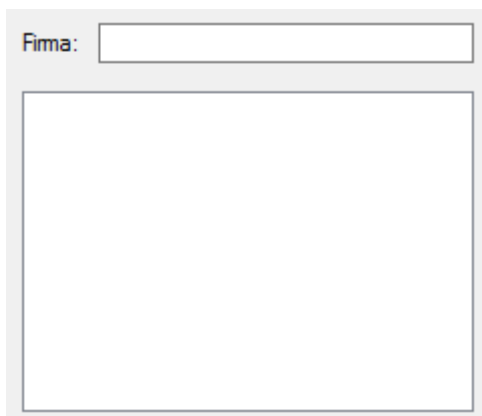
**Primjer 2:** Potrebno je u Korisnika dodati i mogućnost pretrage po firmama ili stranicama na kojima je prijavljen ili ranije bio prijavljen, a forma treba izgledati kao na slici ispod.

The screenshot shows a Windows application titled "Menadžment Korisnika". It features two main sections: "Unos" (Input) and "Pretraga" (Search). The "Unos" section contains several input fields: "Ime", "Prezime", "Spol" (with radio buttons for "Muško" and "Žensko"), "Grad" (a dropdown menu), "Br. Tel." (a field with a country code dropdown), "Datum Rođenja" (a date picker showing "nedjelja, 06. novembar 2016."), "Korisničko Ime", "Lozinka", and "Članstvo". There is also a "Učitaj Sliku..." button. The "Pretraga" section has an "Ime" field and a large empty box. Below the "Pretraga" section is a "Članstvo" section with a "Firma" field and another large empty box. At the bottom of the "Unos" section are "Ponisti" and "Unesi" buttons. A footer note at the bottom of the window says "Detaljne informacije možete naći [ovdje](#)."

U ovom slučaju smo dodali kao dodatnu kontrolu ***richTextBox*** koja nam služi kao kontejner u koji ćemo unositi informacije o prethodnim firmama. Naravno, to je moguće realizirati na jednostavniji način, ali ova kontrola je izabrana radi ilustracije samog primjera. Sam unos će se parsirati u odnos na delimiter “;”.

```
if(!richTextBoxClanstvo.Text.Equals(""))  
    k.Clanstvo = new List<String>(richTextBoxClanstvo.Text.Split(';'));
```

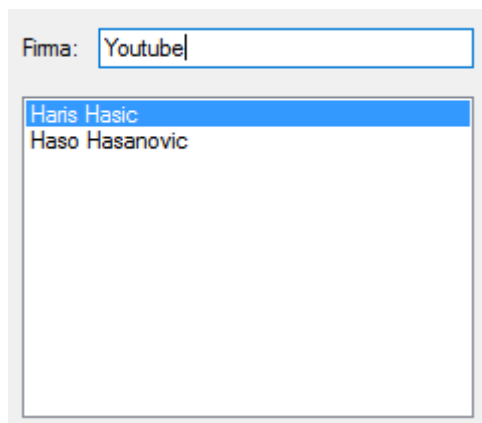
Dakle, rješenje ćemo proširiti tako što dodamo dio u formi koji će služiti za ovu vrstu pretrage, a koji izgleda na slici ispod.



Događaj će se takođe implementirati na *TextChanged* događaj, ali će se ovaj put za pretragu iskoristiti lambda izraz kao ilustracija pojednostavljenja koja nam donosi.

```
private void textBoxClanstvo_TextChanged(object sender, EventArgs e)  
{  
    var rezultati = korisnici.Where(item =>  
item.Clanstvo.Contains(textBoxClanstvo.Text));  
    listBoxClanstvo.DataSource = rezultati.ToList();  
}
```

Rezultat koji se dobije je prikazan na slici ispod.





## Zadaci za samostalan rad i vježbu

### Zadatak 1

Proširiti rješenje iz primjera tako da se napravi prozor za login sa kojeg se moguće logirati na korisnički profil na kome je moguće vršiti izmjene profila. Ako se klikne na link labelu „Registrujte se“ ona dovede do prozora iz prethodnog primjera.

### Zadatak 2

Proširiti rješenje iz primjera i zadatka 1 tako što ćete to rješenje transformisati u elektroničku prodavnicu gdje ćete dodati i registraciju proizvoda (implementirati sve pozadinske klase po principima OOP), te korisniku omogućiti kupovinu tih proizvoda i na taj način kreirati jednu ozbiljnu aplikaciju.

### Zadatak 3

Napraviti igricu po uzoru na igricu sheep game (<http://www.sheepgame.co.uk/>) tako što ćete implementirati da button ili pictureBox kontrole mijenjaju lokaciju ako preko njih pređe kursor miša.



### Zadatak 4

Napraviti memory igricu sa prikazom rezultata i početnim menijem.

