Interakcija čovek - računar

Vežbe 4

Fakultet tehničkih nauka
Univerzitet u Novom Sadu

Tabelarni prikaz podataka

Uvod

Veliki broj aplikacija kao osnovu funkciju ima manipulisanje sa podacima. Podatke je moguće dodavati, menjati, brisati i generisati ali ni jedna od ovih funkcija ne bi imala smisla kada se ti podaci ne bi prikazivali korisniku. Prikaz podataka obezbeđuje se na više načina a radi preglednosti često se bira tabelarni prikaz.

Tema ovih vežbi biće tabelarni prikaz podataka korišćenjem WPF-a.

Primeri

Pri pokretanju aplikacije iz primera za vežbe 4, prvi prozor koji se otvara je *MainWindow* prozor. U XAML kodu ovog prozora može se primetiti da imamo jedan *DockPanel* koji služi da rasporedi komponente unutar prozora. (Slika 1)

Na vrhu (*DockPanel.Dock = "Top"*) nalazi se meni. Jedan od *item*-a je opcija za otvaranje prozora u kojoj se nalazi tabela sa ručno generisanim kolonama, drugi za tabelu sa automatski generisanim kolonama i treći je *databinding* koji povezuje *DataGrid* i *Grid* kontrolu.

Na *MenuItem* komponente zakačen je *Click* događaj. *Handler-i* za ove događaje možete videti u klasi *MainWindow.xaml.cs.* Da se podsetimo, *.xaml.cs* fajl je code-behind. (Slika 2)

```
<Window x:Class="PrimerCas4.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="Primeri upotrebe WPF" Height="350" Width="525">
    <DockPanel>
        <Menu DockPanel.Dock ="Top">
            <MenuItem x:Name="AutoColumns" Header="Auto generisane kolone" Click="AutoColumns_Click"/>
            <MenuItem x:Name="ManualColumns" Header="Ručno generisane kolone" Click="ManualColumns_Click"/>
            <MenuItem x:Name="Binding" Header="Data binding na element" Click="Binding_Click"/>
        </Menu>
        <Grid DockPanel.Dock="Bottom" VerticalAlignment="Stretch" HorizontalAlignment="Stretch">
        </Grid>
    </DockPanel>
</Window>
```

Slika 1. MainWindow.xaml

TableExampleAutoGenerated(), TableExampleManuallyGenerated() i TableExampleBinding() predstavljaju pozive konstruktora istoimenih klasa. Kada se prozor kreira, potrebno ga je i prikazati uz pomoć *Show* metode. (Slika 2)

TableExampleManuallyGenerated prikazuje tablerani prikaz sa ručnim generisanjem kolona dok je *TableExampleAutoGenerated* za automatsko generisanje.

```
public partial class MainWindow : Window
{
    public MainWindow()
    {
        InitializeComponent();
    }

    private void AutoColumns_Click(object sender, RoutedEventArgs e)
    {
        var s = new TableExampleAutoGenerated();
        s.Show();
    }

    private void ManualColumns_Click(object sender, RoutedEventArgs e)
    {
        var s = new TableExampleManuallyGenerated();
        s.Show();
    }

    private void Binding_Click(object sender, RoutedEventArgs e)
    {
        var s = new TableExampleBinding();
        s.Show();
    }
}
```

Slika 2. MainWindow.xaml.cs

Automatsko generisanje kolona

U ovom delu biće objašnjen princip ručnog generisanja kolona u tabeli.

Na slici 3. je prikazan XAML kod prozora koji u sebi sadrži jednu *child* kontrolu – *Grid* panel. *Grid* panel obuhvata *DataGrid* i *Button* kontrolu.

DataGrid kontrola se koristi kada je podatke potrebno prikazati u tabeli.

Kada se odabere automatsko kreiranje kolona, one se kreiraju u zavisnosti od podataka koji su prosleđeni tabeli (*DataGrid*-u).

```
<Window x:Class="PrimerCas4.Table.TableExampleAutoGenerated"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:PrimerCas4.Table"
        mc:Ignorable="d"
        Title="Table Example With Auto Generated Columns" Height="300" Width="350">
    <Grid>
        <DataGrid x:Name="dataGridStudenti" ItemsSource="{Binding Path=Studenti}" AutoGenerateColumns="True"
                  SelectionMode="Single" IsReadOnly="True" AutoGeneratingColumn="generateColumns" HorizontalAlignment="Left"
                  Height="156" Margin="10,47,0,0" VerticalAlignment="Top" Width="322"/>
        <Button x:Name="buttonObrisi" Content="Obrisi poslednjeg" HorizontalAlignment="Left" Height="28"
                  Margin="122,224,0,0" VerticalAlignment="Top"
                  Width="111" Background="White" Foreground="#FF707070" Click="obrisiStudenta"/>
    </Grid>
</Window>
```

Slika 3. TableExampleAutoGenerated.xaml

Bitni atributi za DataGrid:

x:Name – jedinstveni identifikator kontrole, u ovom slučaju tabele

ItemsSource - "izvor" podataka prikazanih u tabeli

Dodatno: *ItemsSources* se *bind*-uje na *Studenti,* listu studenata defisanu u *code-behind* (Slika 6)

Napomena: Da bi ovaj *binding* funkcionisao potrebno je postaviti *DataContext* – omogućavanje da ova kolekcija bude izvor podataka za *binding*. U ovom primeru, postavljanje *DataContext*-a se obavlja u *Code-behind*-u (Slika 6)

AutoGenerateColumns – atribut koji označava da li se kolone automatski generišu. Na ovom mestu je postavljen na vrednost *True* jer želimo iskoristiti automatsko generisanje kolona na osnovu atributa klase *Student*.

SelectionMode – označava koliko redova u tabeli je moguće selektovati. Na ovom mestu postavljena je vrednost *Single* jer se omogućava selektovanje samo jednog reda u tabeli.

Dodatno: Druga vrednost koja se može iskoristiti je *Extended* (objašnjenje iz dokumentacije : *Multiple items in the System.Windows.Controls.DataGrid can be selected at the same time*)

IsReadOnly – označava da li je moguće direktno menjati vrednosti u tabeli

Dodatno: *DataGrid* kontrolu je moguće direktno menjati (eng. *editable by default*), tako što dozvoljava da korisnik direktno iz tabele menja vrednosti podataka koji su joj prosleđeni.

AutoGeneratingColumn –pri automatskom generisanju kolona, moguće je postaviti neke od uslova.

Dodatno: U primeru sa slike 3, *AutoGeneratingColumn = "generateColumns"* obezbeđeno je da na generisanje SVAKE od kolona bude pozvana metoda *generateColumns* iz *code-behind*-a.

Za prvu i drugu kolonu neće biti ispunjen uslov *colNum == 3* dok će za 3. kolonu biti postavljena širina sa

e.Column.Width = new DataGridLength(1, DataGridLengthUnitType.Star)

Da bi lakše razumeli šta znači Data.GridLengthUnitType.Star potrebno je da navedemo kakve se merne jedinice mogu koristiti. (Slika 4)

```
...public enum DataGridLengthUnitType
{
    //
    // Summary:
    //     The size is based on the contents of both the cells and the column header.
    Auto = 0,
    //
    // Summary:
    //     The size is a fixed value expressed in pixels.
    Pixel = 1,
    //
    // Summary:
    //     The size is based on the contents of the cells.
    SizeToCells = 2,
    //
    // Summary:
    //     The size is based on the contents of the column header.
    SizeToHeader = 3,
    //
    // Summary:
    //     The size is a weighted proportion of available space.
    Star = 4
}
```

Slika 4. Tipovi za označavanje veličine

Za primer širine:

Auto – u zavisnosti od širine *Header*-a i širine sadržaja u ćelijama, određuje se i širina kolone.

Dodatno: Ovaj slučaj je *default* pa je zbog toga u našem primeru širina 1. i 2. kolone određena na ovaj način jer nije navedeno drugačije.

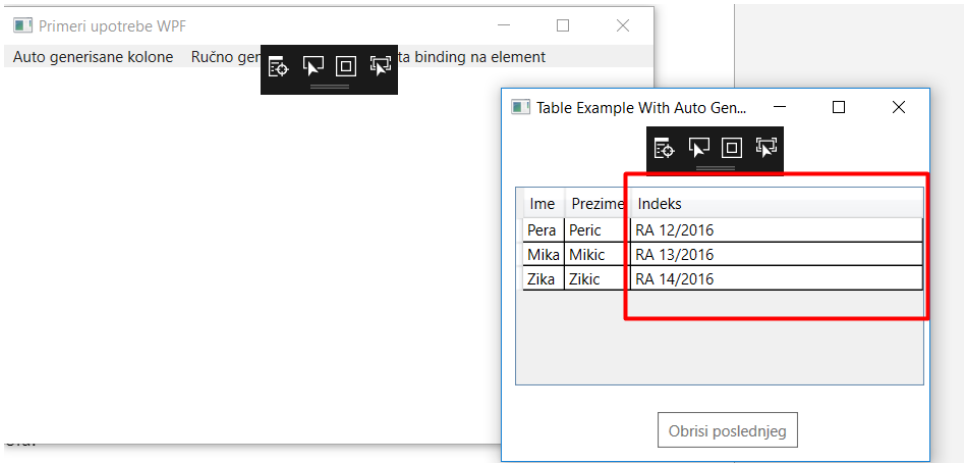
Pixel - širina kolone je tačna fiksna širina navedena u pixelima

SizeToCells - širina kolone se prilagođava širini najšire ćelije

SizeToHeader - širina kolone je širina koju zauzima *Header*

Star – širina kolone zauzima onoliko prostora koliko joj je na raspolaganju. Nakon zauzimanja svih *auto* i *fixed size* kolona, prosto koji preostaje se proporcijalno deli između svih *star-sized* kolona. Tako u našem slučaju 1 označava da se zauzme sav preostali prostor jer nema drugih *star-sized* kolona. (Slika 5)

e.Column.Width = new DataGridLength(1, DataGridLengthUnitType.Star)



Slika 5. Kolona Indeks zauzima preostali prostor

Napomena: Obavezno pogledati šta se dešava kada imamo dve kolone za koje je širina definisana na sledeći način:

Prva kolona: *DataGridLength(1, DataGridLengthUnitType.Star)*

Druga kolona: *DataGridLength(2, DataGridLengthUnitType.Star)*

```
public partial class TableExampleAutoGenerated : Window
{
    private int colNum = 0;
    public ObservableCollection<Student> Studenti
    {
        get;
        set;
    }
    public TableExampleAutoGenerated()
    {
        InitializeComponent();
        this.DataContext = this;
        Studenti = new ObservableCollection<Student>();
        Studenti.Add(new Student() { Ime = "Pera", Prezime = "Peric", Indeks = "RA 12/2016" });
        Studenti.Add(new Student() { Ime = "Mika", Prezime = "Mikic", Indeks = "RA 13/2016" });
        Studenti.Add(new Student() { Ime = "Zika", Prezime = "Zikic", Indeks = "RA 14/2016" });
    }

    private void generateColumns(object sender, DataGridAutoGeneratingColumnEventArgs e)
    {
        colNum++;
        if (colNum == 3)
            e.Column.Width = new DataGridLength(1, DataGridLengthUnitType.Star);
    }

    private void obrisiStudenta(object sender, RoutedEventArgs e)
    {
        if (Studenti.Count > 0)
        {
            Studenti.RemoveAt(Studenti.Count - 1);
        }
        else
        {
            MessageBox.Show("Nije moguce brisati iz prazne tabele.", "Greska!", MessageBoxButton.OK, MessageBoxImage.Error);
        }
    }
}
```

Slika 6. TableExampleAutoGenerated.xaml.cs

Klikom na dugme *ButtonObrisi* vrši se brisanje poslednjeg studenta iz kolekcije. (metoda *obrisiStudenta* sa slike 6)

Studenti.Count je veličina te kolekcije a da bi se moglo pristupiti elementu na poslednjoj poziciji, potrebno je dobaviti indeks tog elementa na način *Studenti.Count - 1.*

Da ne bi došlo do izuzetka, potrebno je prethodno proveriti da li je kolekcija prazna. Validaciju i rukovanje izuzecima detaljnije ćemo proći na nekim od sledećih vežbi.

Ručno generisanje kolona

U automatskom generisanju kolona, generiše se broj kolona na osnovu broja polja u klasi koja predstavlja izvor podataka.

Za razliku od automatskog generisanja, u ručnom generisanju moguće je odabrati podskup polja i odabrati različit redosled u njihovom prikazivanju. Ovo je poželjno u slučaju kada klasa sadrži veliki broj atributa od kojih nije potrebno/nije dozvoljeno prikazati sve.

Za odabir tipova i broja kolona potrebno je koristiti *Columns* atribut *DataGrid-a* i to kao na slici 7. Za svaku od kolona potrebno je odabrati tip kolone.

Tipovi kolona koji su omogućeni:

- DataGridTextColumn
- DataGridCheckBoxColumn
- DataGridComboBoxColumn
- DataGridHyperlinkColumn
- DataGridTemplateColumn

Kada se želi iskoristiti složeniji oblik za prikaz podataka u koloni (npr. *Date picker*) potrebno je iskoristiti *DataGridTemplateColumn* (primer je dat na linku https://stackoverflow.com/questions/18680083/how-to-add-a-datepicker-to-datagridtextcolumn-in-wpf).

U ovom primeru za sve tri kolone iskoristićemo *DataGridTextColumn*.

```
<Window x:Class="PrimerCas4.Table.TableExampleManuallyGenerated"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="Table Example With Manually Generated Columns" Height="300" Width="350">
    <Grid>
        <DataGrid x:Name="dataGridStudenti" ItemsSource="{Binding Studenti}" IsReadOnly="True"
                  SelectionMode="Single" AutoGenerateColumns="False" HorizontalAlignment="Left"
                  Height="158" Margin="10,47,0,0" VerticalAlignment="Top" Width="322">
            <DataGrid.Columns>
                <DataGridTextColumn Header="Ime" Binding="{Binding Ime}"/>
                <DataGridTextColumn Header="Prezime" Binding="{Binding Prezime}"/>
                <DataGridTextColumn Header="Indeks" Binding="{Binding Indeks}" Width="*"/>
            </DataGrid.Columns>
        </DataGrid>
    </Grid>
</Window>
```

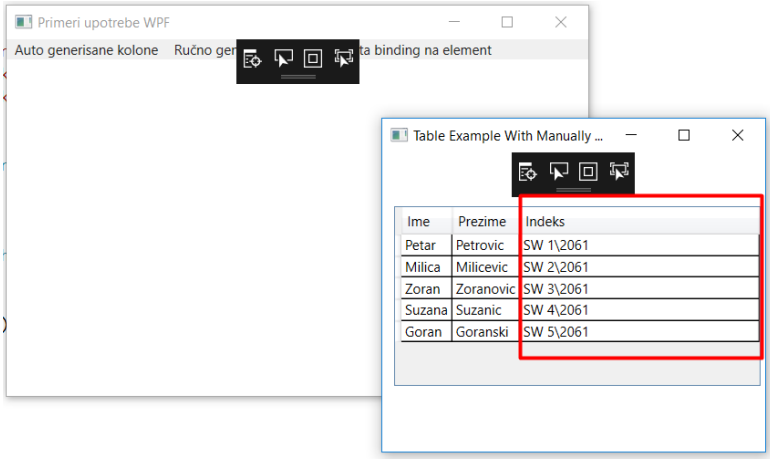
Slika 7. TableExampleManuallyGenerated.xaml

Na slici 7 je prikazano kako vrednost *Ime* polja iz klase *Student bind*-ujemo na kolonu *Ime.* Isti princip ponovljen je i za kolone *Prezime* i *Indeks*. Na taj način, kolone će sadržati vrednosti atributa *Ime, Prezime i Indeks.*

```
<DataGrid.Columns>
    <DataGridTextColumn Header="Ime" Binding="{Binding Ime}"/>
    <DataGridTextColumn Header="Prezime" Binding="{Binding Prezime}"/>
    <DataGridTextColumn Header="Indeks" Binding="{Binding Indeks}" Width="*"/>
</DataGrid.Columns>
```

Slika 8. Binding atributa klase na kolonu

Koloni *Indeks* je postavljena širina sa *Width = "*"* i označava *DataGridLengthUnitType.Star* vrednost o kojoj je prethodno bilo reč. *Indeks* kolona, u ovom slučaju, zauzima prostor do kraja tabele. (Slika 9)



Slika 9. Popunjavanje ostatka slobodnog prostora u tabeli

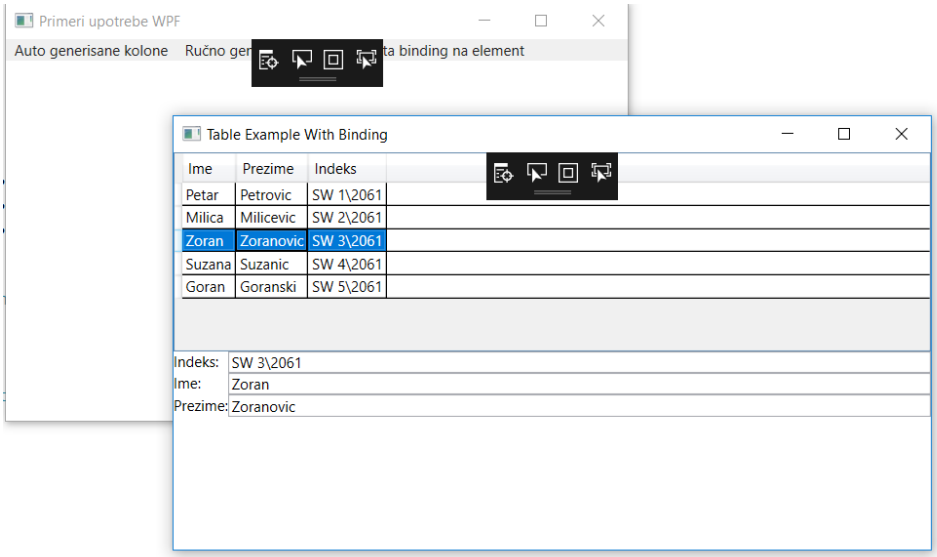
Popunjavanje liste studenata kao i omogućavanje da ova kolekcija bude izvor podataka za *binding* (postavljanje *DataContext*-a) vrši se u *code-behind*-u (Slika 10)

```
public partial class TableExampleManuallyGenerated : Window
{
    public ObservableCollection<Student> Studenti
    {
        get;
        set;
    }
    public TableExampleManuallyGenerated()
    {
        InitializeComponent();
        this.DataContext = this;
        Studenti = new ObservableCollection<Student>();
        Studenti.Add(new Student {Ime = "Petar", Prezime = "Petrovic", Indeks = "SW 1\\2061" });
        Studenti.Add(new Student { Ime = "Milica", Prezime = "Milicevic", Indeks = "SW 2\\2061" });
        Studenti.Add(new Student { Ime = "Zoran", Prezime = "Zoranovic", Indeks = "SW 3\\2061" });
        Studenti.Add(new Student { Ime = "Suzana", Prezime = "Suzanic", Indeks = "SW 4\\2061" });
        Studenti.Add(new Student { Ime = "Goran", Prezime = "Goranski", Indeks = "SW 5\\2061" });
    }
}
```

Slika 10. TableExampleManuallyGenerated.xaml.cs

Kombinacija DataGrid i Grid kontrole i DataBinding

U ovom primeru pokazano je kako selektovanjem na red u tabeli dolazi do popunjavanja polja u panelu ispod tabele. (Slika 11)



Slika 11. Rezultat selektovanje reda u tabeli

XAML kod za ovaj primer dat je na slici 12. *Grid* panel obuhvata *DataGrid* komponentu i novi *Grid* panel kao što je označeno. *DataGrid* predstavlja tabelu čiji redovi će se selektovati, dok označeni *Grid* panel obuhvata *TextBox* kontrole koje će se popunjavati sa ovim informacijama.

Kao i u prethodnim primerima, *ItemsSouce DataGrid*-a *bind*-ovan je na kolekciju *Studenti* i tim omogućava da u redovima u tabeli budu informacije o pojedinačnim studentima.

```
<Window x:Class="PrimerCas4.Table.TableExampleBinding"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="Table Example With Binding" Height="363" Width="636">
    <Grid>
        <Grid.RowDefinitions>
            <RowDefinition />
            <RowDefinition />
        </Grid.RowDefinitions>
        <Grid.ColumnDefinitions>
            <ColumnDefinition />
        </Grid.ColumnDefinitions>
        <DataGrid x:Name="dgrMain" ItemsSource="{Binding Path=Studenti}" IsReadOnly="True"/>
        <Grid Grid.Column="0" Grid.Row="1" DataContext="{Binding ElementName=dgrMain,Path=SelectedItem}">
            <Grid.RowDefinitions>
                <RowDefinition Height="Auto"/>
                <RowDefinition Height="Auto"/>
                <RowDefinition Height="Auto"/>
            </Grid.RowDefinitions>
            <Grid.ColumnDefinitions>
                <ColumnDefinition Width="Auto" />
                <ColumnDefinition />
            </Grid.ColumnDefinitions>
            <TextBlock Grid.Column="0" Grid.Row="0">Indeks: </TextBlock>
            <TextBlock Grid.Column="0" Grid.Row="1">Ime: </TextBlock>
            <TextBlock Grid.Column="0" Grid.Row="2">Prezime:</TextBlock>

            <TextBox Grid.Column="1" Grid.Row="0" Text="{Binding Path=Indeks}"></TextBox>
            <TextBox Grid.Column="1" Grid.Row="1" Text="{Binding Path=Ime}"></TextBox>
            <TextBox Grid.Column="1" Grid.Row="2" Text="{Binding Path=Prezime}"></TextBox>
        </Grid>
    </Grid>
</Window>
```

Slika 12. TableExampleBinding.xaml

U označenom *Grid* panelu se može primetiti kako je *Text* atribut prvog *TextBox*-a bind-ovan na *Indeks*. (Slika 13) *Text* atributi drugog i trećeg na *Ime* I *Prezime*.

```
        <DataGrid x:Name="dgrMain" ItemsSource="{Binding Path=Studenti}" IsReadOnly="True"/>
        <Grid Grid.Column="0" Grid.Row="1" DataContext="{Binding ElementName=dgrMain,Path=SelectedItem}">
            <Grid.RowDefinitions>
                <RowDefinition Height="Auto"/>
                <RowDefinition Height="Auto"/>
                <RowDefinition Height="Auto"/>
            </Grid.RowDefinitions>
            <Grid.ColumnDefinitions>
                <ColumnDefinition Width="Auto" />
                <ColumnDefinition />
            </Grid.ColumnDefinitions>
            <TextBlock Grid.Column="0" Grid.Row="0">Indeks: </TextBlock>
            <TextBlock Grid.Column="0" Grid.Row="1">Ime: </TextBlock>
            <TextBlock Grid.Column="0" Grid.Row="2">Prezime:</TextBlock>

            <TextBox Grid.Column="1" Grid.Row="0" Text="{Binding Path=Indeks}"></TextBox>
            <TextBox Grid.Column="1" Grid.Row="1" Text="{Binding Path=Ime}"></TextBox>
            <TextBox Grid.Column="1" Grid.Row="2" Text="{Binding Path=Prezime}"></TextBox>
        </Grid>
    </Grid>
</Window>
```

Slika 13. Bindovanje Text atributa TextBox kontrole na vrednosti selektovanog studenta

Da bi ovaj *binding* mogao da funkcioniše potrebno je u *DataContext* za ovaj *Grid* postaviti SELEKTOVANOG STUDENTA. (Slika 13)

U primerima sa prethodnih vežbi (Vežbe 3) mogli smo videti na koje sve načine je moguće uraditi *binding*. Slično tome, sada smo u *DataContext* postavili vrednost *SelectedItem* atributa *dgrMain* kontrole. *dgrMain* je jedinstveni identifikator *DataGrid* kontrole.

Na ovaj način, kada se selektuje red u tabeli *dgrMain*, atribut *SelectedItem* tabele *dgrMain* sadrži informaciju koji student je selektovan. S obzirom da smo ovu informaciju stavili u *DataContext* i *bindovali* vrednosti *TextBox* polja, svaka promena selektovanog reda rezultuje promenu i u *TextBox* kontrolama. (Slika 13)

Slika 14 prikazuje *code-behind* ovog primera.

```
public partial class TableExampleBinding : Window
{
    public ObservableCollection<Student> Studenti
    {
        get;
        set;
    }
    public TableExampleBinding()
    {
        InitializeComponent();
        this.DataContext = this;
        Studenti = new ObservableCollection<Student>();
        Studenti.Add(new Student {Ime = "Petar", Prezime = "Petrovic", Indeks = "SW 1\\2061" });
        Studenti.Add(new Student { Ime = "Milica", Prezime = "Milicevic", Indeks = "SW 2\\2061" });
        Studenti.Add(new Student { Ime = "Zoran", Prezime = "Zoranovic", Indeks = "SW 3\\2061" });
        Studenti.Add(new Student { Ime = "Suzana", Prezime = "Suzanic", Indeks = "SW 4\\2061" });
        Studenti.Add(new Student { Ime = "Goran", Prezime = "Goranski", Indeks = "SW 5\\2061" });
    }
}
```

Slika 14. TableExampleBinding.xaml.cs