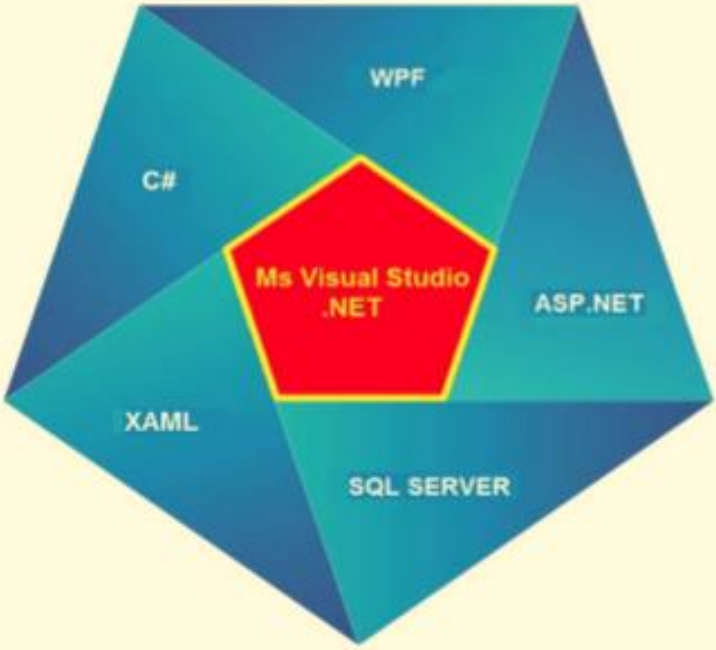
გია სურგულაძე,
ლილი პეტრიაშვილი,
მარინე ბიტარაშვილი

კორპორაციული მენეჯმენტის დაპროგრამების ტექნოლოგია



„IT-კონსალტინგის ცენტრი“

გია სურგულაძე, ლია პეტრიაშვილი,
მარინე ბიტარაშვილი,

კორპორაციული მენეჯმენტის სისტემების დაპროგრამების ტექნოლოგია:

WPF.NET, SQL Server & UML

(საკურსო პროექტის დამხმარე სახელმძღვანელო)

დამტკიცებულია:
სტუ-ს „IT-კონსალტინგის"
სამეცნიერო ცენტრის
სარედაქციო-საგამომცემლო
კოლეგიის მიერ

თბილისი
2017

უაკ 004.5

განიხილება პროგრამული აპლიკაციების დეველოპმენტის (გამოყენებითი კომპიუტერული სისტემების დაპროგრამების) საფუძვლები. კერძოდ, შემოთავაზებულია აღნიშნულ დისციპლინაში საკურსო პროექტის შესრულების მეთოდოლოგია და პროგრამული პაკეტის აგების ინსტრუმენტული საშუალებები. ასევე სამაგიდო (Windows) და ინტერნეტული (Web) აპლიკაციების აგების ეტაპები და პროცედურები, MsVisual Studio.NET Framework 4.5 პლატფორმაზე C# და XAML ენების, MsSQL Server მონაცემთა ბაზის და ADO.NET დრაივერის გამოყენებით.

განკუთვნილია ინფორმატიკის საგანმანათლებლო პროგრამის „მართვის ავტომატიზებული სისტემების (პროგრამული ინჟინერიის)“ სპეციალობის ბაკალავრიატის მეოთხე კურსის სტუდენტებისათვის, საკურსო პროექტის შესასრულებლად - დამამთავრებელი კომპლექსური ნაშრომის სახით.

რეცენზენტები:

პროფ. ე. თურქია,
პროფ. თ. სუხიაშვილი

გია სურგულაძის რედაქციით

ISBN 978-9941-0-9843-7

სარჩევი

შესავალი:
საკურსო პროექტის მიზანი, ამოცანის დასმა და შესრულების გეგმა

საკურსო პროექტი სრულდება ბაკალავრიატის ბოლო სემესტრში და მისი მიზანია სტუდენტის ცოდნის შემაჯამებელი ნაშრომის შესრულება დაპროგრამების თანამედროვე ტექნოლოგიების გამოყენებით (Windows და Web აპლიკაციების პაკეტების სახით). პროექტი კომპლექსური, ინტერდისციპლინური ხასიათისაა. იგი სტუდენტისაგან მოითხოვს საპრობლემო სფეროს ობიექტ-ორიენტირებული მოდელირების, ანალიზისა და დაპროექტების მეთოდოლოგიის ცოდნას საერთაშორისო სტანდარტების გათვალისწინებით, კორპორაციის მონაცემთა განაწილებული ბაზების აგებისა და ვიზუალური დაპროგრამების მეთოდების და ინსტრუმენტული საშუალებების გამოყენების უნარ-ჩვევებს [1,2].

სტუდენტი ინდივიდუალურად (ან ჯგუფურად რამდენიმე სტუდენტი) მიიღებს მასწავლებლისგან კონკრეტულ დავალებას (ამოცანას) საკურსო პროექტის შესასრულებლად. საპრობლემო სფეროს შერჩევა დასაშვებია სტუდენტის კონსულტაციის გზით პედაგოგთან, რათა თემატიკა იყოს აქტუალურიც და სტუდენტისათვის საინტერესო.

კორპორაციული საპრობლემო სფერო შეიძლება იყოს ნებისმიერი დარგის ობიექტი. მაგალითად, განათლების სფერო: უნივერსიტეტი, ფაკულტეტი, კათედრა, სკოლა და ა.შ.; სამინისტროს ნებისმიერი დეპარტამენტი, სოფლის მეურნეობის სფერო: სასოფლო სამეურნეო პროდუქციის წარმოება, ფერმა და ა.შ.; საბანკო სისტემა: კლიენტთა ანაბრების მართვა, კრედიტების დეპარტამენტი, რისკების ანალიზი და ა.შ., ბიზნესის და კომერციის სფერო: მარკეტინგის დეპარტამენტი, სასაწყობო მეურნეობა, სუპერმარკეტი და ა.შ. ან სხვა სფეროები (აეროპორტი, ტრანსპორტი,

აფთიაქები, ბიბლიოთეკა, მუზეუმები, კინო-თეატრები, სპორტი, სხვადასხვა სახის ელექტრონული ცნობარი და ა.შ.).

მაგალითისათვის განვიხილოთ საილუსტრაციო ამოცანა - „კორპორაციაში (ან ფირმაში) ახალი თანამშრომლის მიღება და ხელფასის დანიშვნის ავტომატიზებული სისტემა“.

ამოცანის გადასაწყვეტად საჭიროა:

1) ჩამოვაყალიბოთ შინაარსობრივად (ტექსტური ფორმით) ფირმაში ახალი თანამშრომლის მიღების არსებული ბიზნეს–პროცესი და ბიზნესწესები. შედეგად უნდა მივიღოთ ასაგები სისტემის ფუნქციური მოთხოვნილებები: როლები და მათი ფუნქციები (ანუ რომელი თანამდებობრივი როლები მონაწილეობს ახალი თანამშრომლის მიღების პროცესში და რა ფუნქციებს ასრულებენ ისინი აქ). ობიექტ-ორიენტირებული მოდელირების მიზნით გამოვიყენებთ უნიფიცირებული მოდელირების ენის (UML ტექნოლოგიის) UseCase და Activity დიაგრამებს [1,2]. მოდელების ასაგებად გრაფიკული სახით ვიყენებთ Microsoft Visual Studio.NET ინტეგრირებულ გარემოს Modeling Projects-ის საფუძველზე [3-5];

2) მიღებული ბიზნესპროცესების და ბიზნესწესების საფუძველზე (Activity-D) ამოცანისათვის: „ორგანიზაციაში ახალი თანამშრომლის მიღება და ხელფასის დანიშვნა“ უნდა აიგოს კონკრეტული როლის კომპიუტერთან მუშაობის „სცენარი“ UML-ის მიმდევრობითობის დიაგრამის სახით (Sequence-D);

3) დასმული ამოცანისათვის კლასების განსაზღვრა (Class-D) და კლასთა-ასოციაციის მოდელის (Class-assotiation-D) აგება არსებული კავშირების (მემკვიდრეობითი, აგრეგატული, ასოციაციური და რელაციური) გათვალისწინებით;

4) მონაცემთა ბაზის დაპროექტება და აგება დასმული ამოცანისათვის. საჭიროა გამოვლინდეს დოკუმენტთა ფორმები და ინფორმაცია: *საწყისი* (ნორმატიული: თანამდებობები, ხელფასები,

დაქვითვების სკალა; *ოპერატიული:* ყოველდღიური ან თვიური აღრიცხვის ტაბელები, შვებულების განრიგი, საავადმყოფო ბიულეტენი); *გამომავალი* (უწყისი თანამშრომელთა ანგარიშებზე ხელფასების გადასარიცხად);

4) განისაზღვროს მონაცემთა ბაზის სტრუქტურა არსთა დამოკიდებულების მოდელით. აიგოს ცხრილები (Tables) და ER დიაგრამა Ms SQL Server Management Studio-ს პაკეტით. შეივსოს ცხრილები რამდენიმე კონკრეტული სტრიქონით;

6) შემუშავდეს მომხმარებელთა ინტერფეისები (სამუშაო ფორმები), MsVisual_Studio.NET Framework 4.5 ინტეგრირებულ გარემოში Visual C# და WPF Application პროექტის სახით;

7) შემუშავდეს ხელფასის დარიცხვის ამოცანის პროცესების შესრულების მეთოდები;

8) შემუშავდეს მომხმარებელთა ინტერფეისებისათვის ხელფასების MsSQL Server ბაზასთან კავშირის და მონაცემთა ამორჩევის, ჩამატების, წაშლის და მოდიფიკაციის მეთოდები, შესაბამისი C# კოდებით;

9) SQL-ენაზე შეიქმნას საილუსტრაციო სტანდარტული მოთხოვნები, რომლებიც განთავსდება Button ღილაკების პანელზე;

10) ჩატარდეს აგებული სისტემის ტესტირება კონკრეტულ მონაცემებზე;

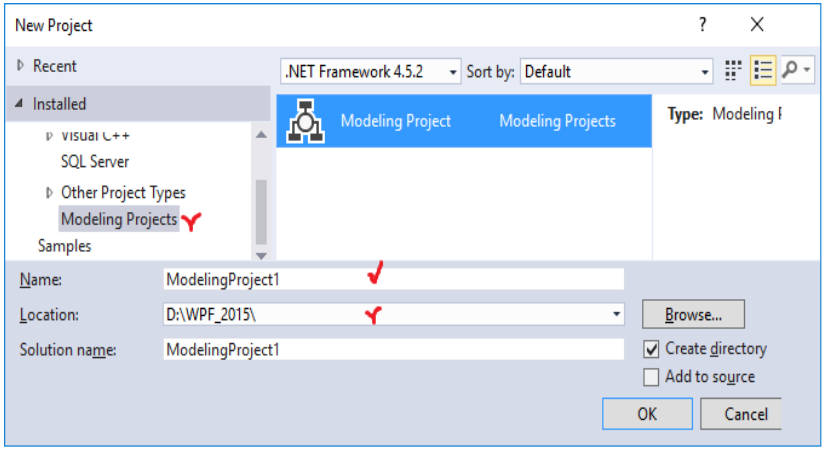
11) მომზადდეს სისტემის სადემონსტრაციო ვერსია და პროექტის აღწერა მომხმარებელთა ინსტრუქციებით.

I თავი
ბიზნესპროცესების აღწერა და სისტემის დაპროექტება UML ენაზე

1.1. სისტემის ფუნქციურ მოთხოვნილებათა განსაზღვრა: როლები / ფუნქციები (Use Case / Activity-D)

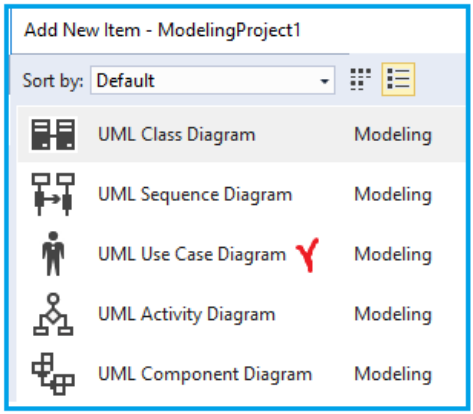
განვიხილოთ ბიზნესპროცესების ობიექტ-ორიენტირებული ანალიზისა და დაპროექტების კონკრეტული ამოცანის მოდელირების საკითხები MsVisual Studio.NET 2015 პლატფორმაზე [4]. დიაგრამების ასაგებად გამოიყენება ModelingProject შაბლონი.

MsVisual Studio.NET 2015 პლატფორმის სამუშაო გარემოში ვირჩევთ სტრიქონს - Modeling Projects (ნახ.1.1), აგრეთვე ვუთითებთ Name-ს და Location-ს. ბოლოს OK.

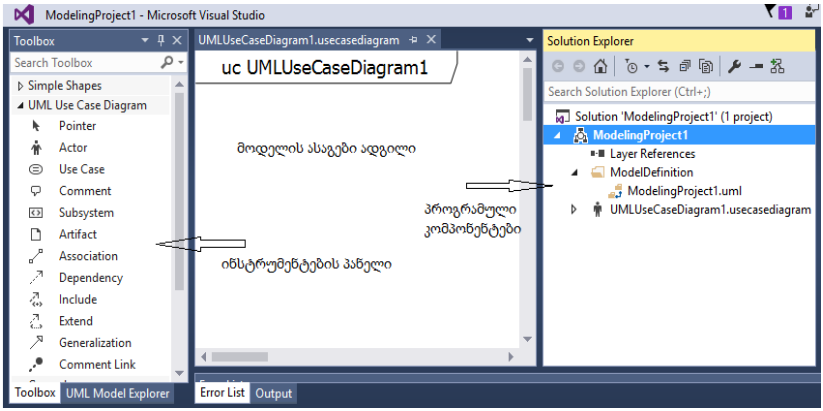


ნახ.1.1. პროექტის აგების დასაწყისი (Ms VisualStudio .NET 2015)

თუ სახელი არ მიეთითება, მაშინ სისტემა თვითონ ქმნის პროექტს ModelingProject1 სახელით, ხოლო Sollution Explorer-ში ჩვენ ვირჩევთ Add new Item-ით ჩვენთვის საჭირო ფუნქციას, ამ შემთხვევაში Use Case Diagram (ნახ.1.2). მიიღება 1.3 ნახაზზე ნაჩვენები საწყისი მდგომარეობა.

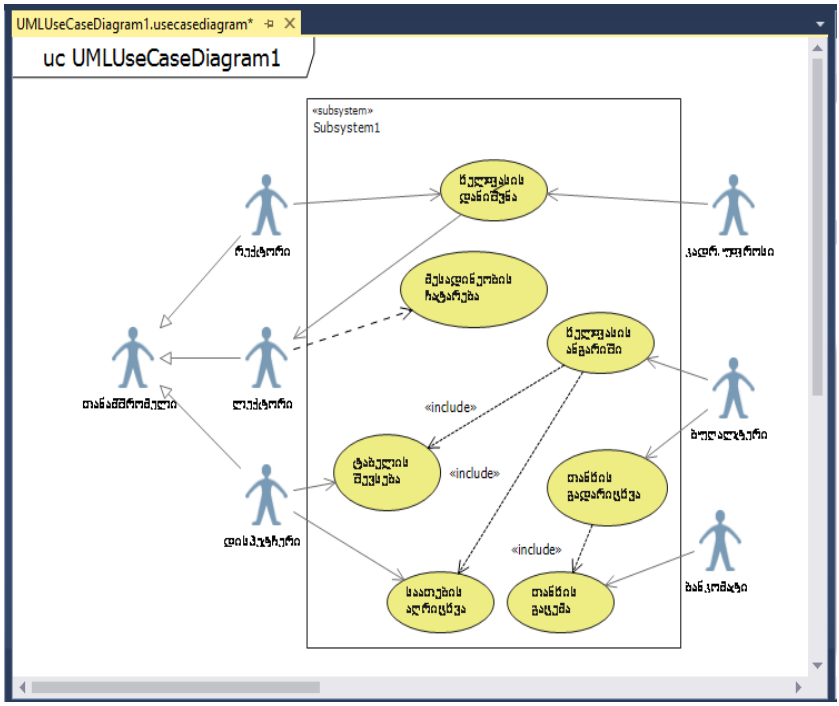


ნახ.1.2



ნახ.1.3. Use Case მოდელის აგების ინტერფეისი (MsVisualStudio .NET 2015)

1.4 ნახაზზე მოცემულია ხელფასის დარიცხვის ამოცანის ბიზნესპროცესში მონაწილე მომხმარებელთა როლები (Actors) და ფუნქციები (Actions), რომელიც აგებულია გამოყენებით-შემთხვევათა (UseCase-D) დიაგრამის საფუძველზე.



ნახ.1.4. UseCase დიაგრამის ფრაგმენტი

როლი განსაზღვრავს კომპიუტერთან მომუშავის სტატუსს, მაგალითად, ლექტორი, დისპეტჩერი, ბუღალტერი და სხვ. ფუნქცია არის პროცედურა, რომელსაც როლი ასრულებს. მაგალითად, დისპეტჩერი აღრიცხავს ლექტორის მიერ ჩატარებულ მეცადინეობებს, ბუღალტერი ანგარიშობს თანამშრომელთა ხელფასს და ა.შ. როლებისა და ფუნქციების საშუალებით განისაზღვრება კომპიუტერული სისტემის მომხმარებელთა პრივილეგიები და ბაზიდან ამა თუ იმ მონაცემთა მიღების უფლებები. რექტორი, ლექტორები და დისპეტჩერი ქვეკლასებია განზოგადებული (Subclass, inheritance) კლასისა - თანამშრომელი, ამიტომაც ისარი მიმართულია მისკენ („მშობლისკენ“).

ფუნქცია (ოვალი) არის ქმედება, რომელსაც როლი ასრულებს. მაგალითად, „ხელფასის დანიშვნა": პიროვნება გარკვეული საკონკურსო წესების საფუძველზე და პირადი განცხადებით საბუთებს წარუდგენს კადრების განყოფილებას (ან საკონკურსო კომისიას), სადაც გარკვეული ეტაპების გავლის შემდეგ, თუ საკითხი დადებითად გადაწყდა, კადრების ინსპექტორი მოამზადებს ბრძანების პროექტს და რექტორის ხელმოწერის შემდეგ ახალ თანამშრომელს, მაგალითად, ლექტორს დაენიშნება თვიური ხელფასი (დავუშვათ 1500 ლარი).

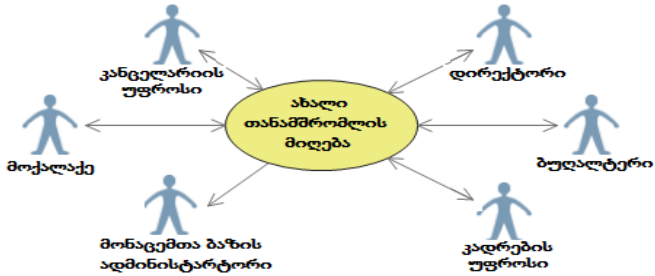
ფუნქცია „მეცადინეობის ჩატარება" ევალება ლექტორს და იგი სადისპეტჩერო კომპიუტერში „რეგისტრაციით" აფიქსირებს „ხელმოწერას". არდაფიქსირება ნიშნავს „გაცდენას".

ფუნქციები „მეცადინეობის ჩატარების აღრიცხვა" და „ელ-ტაბელის შევსება" ევალება კომპიუტერ-დისპეტჩერს, რომელიც ლექტორთა ელ-აღრიცხვის ჟურნალიდან მონაცემებს გადაიტანს ელ-ტაბელში. ესაა ყოველთვიური ელ-დოკუმენტი (ფაილი), რომელიც გადაეცემა ბუღალტერიას.

ფუნქცია „ხელფასის ანგარიში" ევალება ბუღალტერს. კათედრაზე N თანამშრომელია სხვადასხვა ხელფასით. ამიტომ იგი კათედრიდან გადმოცემულ სატაბელო მონაცემებს შეადარებს თავის კომპიუტერში არსებულ ინფორმაციას და შემდეგ დაიანგარიშებს თითოეული ლექტორისათვის დარიცხულ თანხებს, დაქვითვებს, სხვადასხვა გადასახადს და ბოლოს ხელზე ასაღებ თანხას. შედეგები გადაიგზავნება შესაბამის ლექტორთა კონკრეტულ ანგარიშებზე ელექტრონული ანგარიშსწორების მიზნით. 1.4 ნახაზზე ყოველ გამოყენებით შემთხვევას ანუ ფუნქციას (ოვალს) შეესაბამება ერთი აქტიურობის ანუ ქმედების დიაგრამა (Activity Diagram).

თუ ფუნქციას რამდენიმე როლი ასრულებს (ნახ.1.5), მაშინ საჭიროა განისაზღვროს თითოეულის კონკრეტული ოპერაცია (მოქმედება) და შესრულების მიმდევრობის რეგლამენტი (ვინ, რა, როდის - უნდა შეასრულოს).

1.6 ნახაზზე განიხილება ამ ფუნქციის შესაბამისი აქტიურობის დიაგრამა: „ახალი თანამშრომლის მიღება".



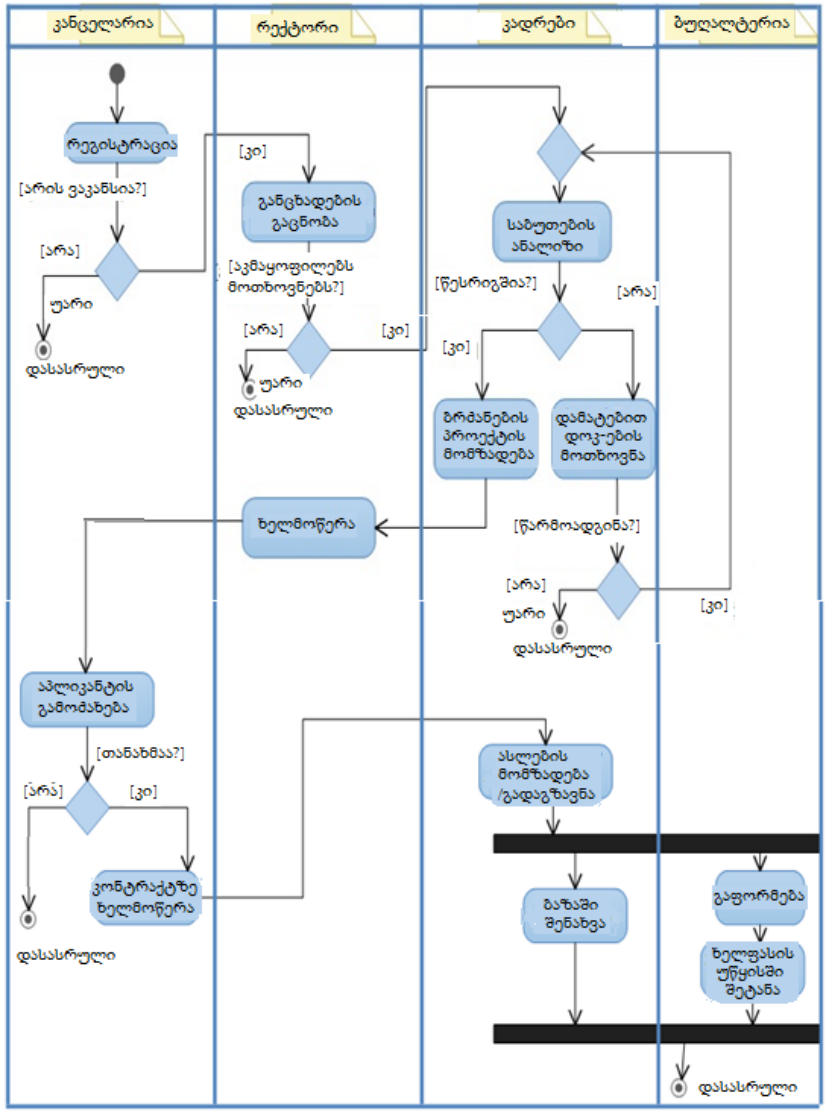
ნახ.1.5

აქტიურობის დიაგრამაში ბილიკები ასახავს როლების მართვის სფეროებს. მაგალითად, კანცელარიაში შემოდის პიროვნების განცხადება ამა თუ იმ ვაკანსიის დაკავების შესახებ. განცხადება გადის რეგისტრაციას კანცელარიაში. თუ ორგანიზაციაში არ არის ვაკანტური ადგილი, განმცხადებელი ღებულობს უარს. თუ ვაკანსია არსებობს, განცხადება გადაეცემა დირექტორს, რომელიც გადახედავს რა კანდიდატების მონაცემებს, პირადი მოსაზრებით აარჩევს საუკეთესოს, დაადებს რეზოლუციას და გადასცემს კადრების განყოფილებას. კადრების ინსპექტორი გადაამოწმებს ვაკანსიის არსებობას და პიროვნების შრომითი საქმიანობის დოკუმენტაციას.

თუ ყველაფერი წესრიგშია, მოამზადებს ბრძანების პროექტს და გადასცემს დირექტორს ხელმოსაწერად.

კანცელარია უგზავნის შეტყობინებას (ურეკავს ტელეფონზე) განმცხადებელს კონტრაქტზე ხელმოწერის მიზნით. ხელმოწერის შემდეგ, კადრების ინსპექტორი ამზადებს ბრძანების ასლებს, რომელთაგან ერთი მიდის ბუღალტერიაში, სადაც მას ხელფასი ენიშნება, მეორე სისტემის მონაცემთა ბაზის ადმინისტრატორთან - კომპიუტერში შესატანად. კანცელარიაში ასევე ამზადებენ პირადობის მოწმობას.

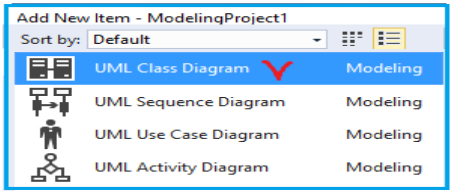
Activity-D: ამოცანა - „ახალი თანამშრომლის მიღება“



ნახ.1.6. Activity დიაგრამის ფრაგმენტი

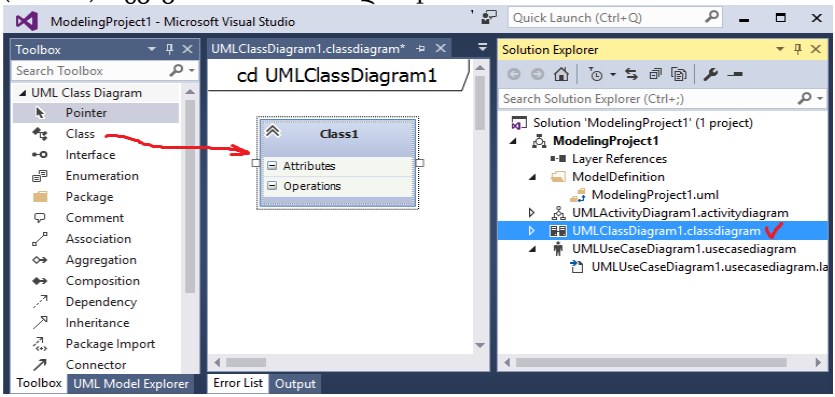
1.2. კლასების აღწერა და კლასთაშორის ასოციაციის დიაგრამა (Class-D, StateChart-D)

კლასი არაერთგვაროვან მონაცემთა სტრუქტურაა (int, string, double და სხვ.), რომელსაც ქმნის მომხმარებელი და მასში შესაძლებელია ზოგიერთს ხილვადობის private მოდიფიკატორი ჰქონდეს (ანუ იყოს დამალული „სხვებისთვის“). ზოგადად, კლასი არის „დასახელების“, „კლასის მონაცემების“ და „კლასის მეთოდების“ ინკაფსულაცია. Ms Visual Studio.NET-ში კლასთა მოდელის ასაგებად ვიყენებთ Add New Item და მის ტიპს (ნახ.1.7).



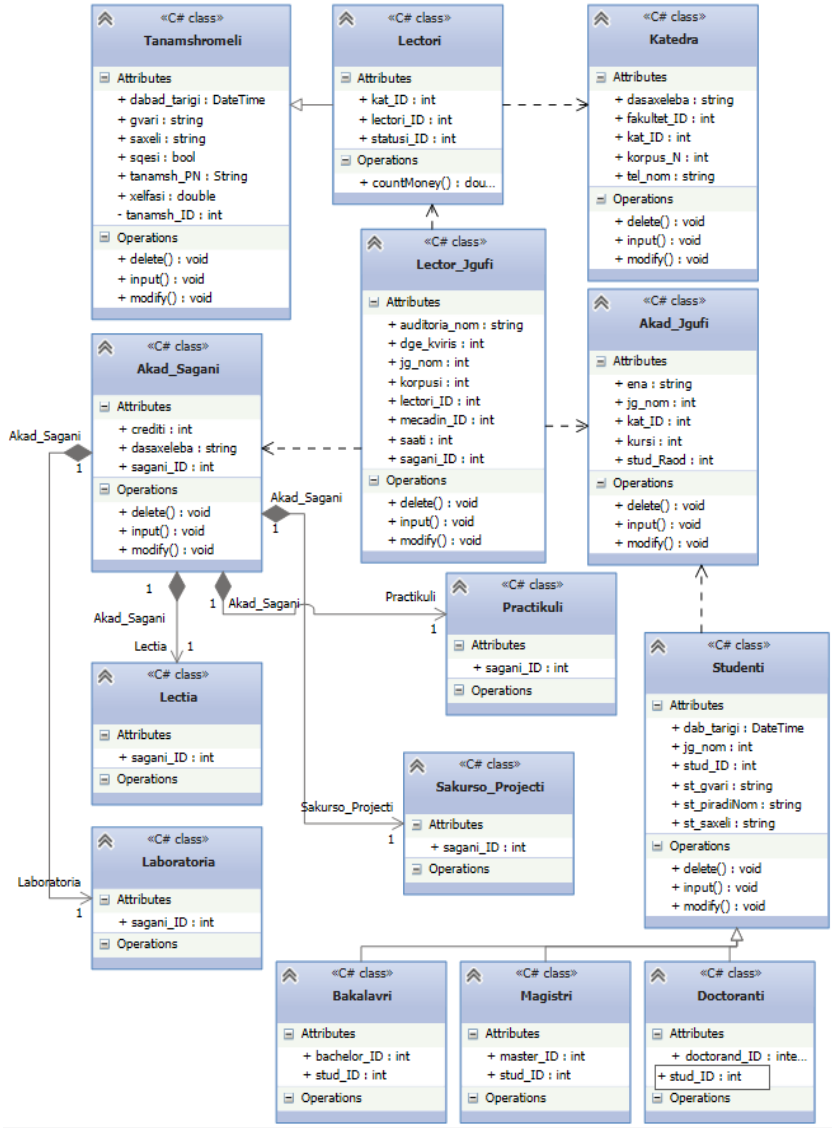
ნახ.1.7. კლასების დიაგრამის დამატება პროექტში

ამგვარად, კლასების ასაგებად მიიღება შემდეგი ინტერფეისი (ნახ.1.8). შევავსოთ Attributes და Operators.



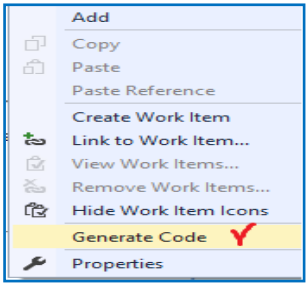
ნახ.1.8. კლასების დიაგრამის ასაგები ინტერფეისი

ჩვენი მართვის სფეროს შესაბამისი კლასები, მაგალითად, ასე უნდა გამოიყურებოდეს (ნახ.1.9). გამოყენებულია მემკვიდრეობითობის, კომპოზიციის და ასოციაციის კლასთაშორისი კავშირები.

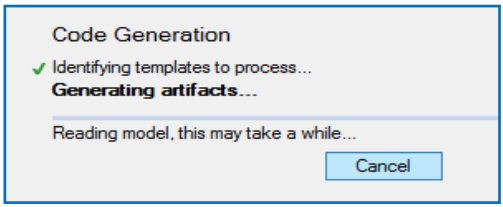


ნახ.1.9. სასწავლო პროცესის კლასები
(VisualStudio .NET 2015)

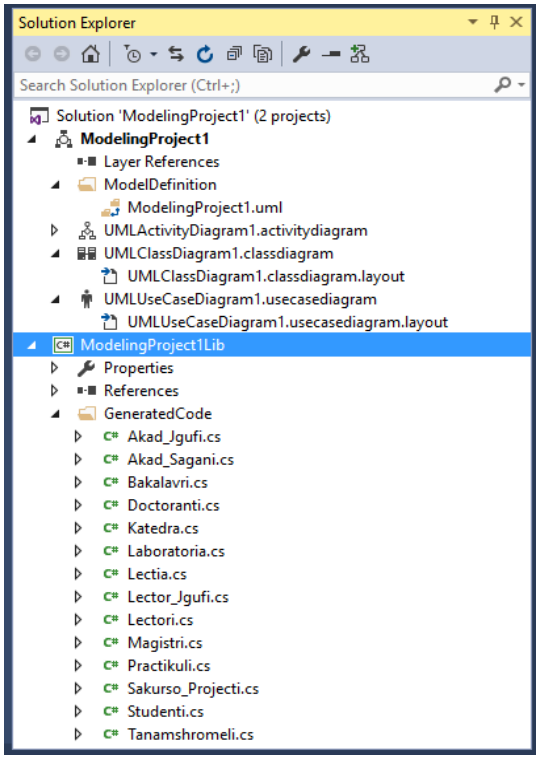
მიღებული კლასთა-ასოციაციის დიაგრამის საფუძველზე ვახდენთ C# კოდის გენერაციას კონტექსტური მენიუთი (ნახ.1.10). გამოდის შეტყობინება გენერაციის პროცესის შესახებ (ნახ.1.11), რომელიც გრძელდება რამდენიმე წუთი შედეგით (ნახ.1.12).



ნახ.1.10



ნახ.1.11



ნახ.1.12. კლასებისთვის გენერირებული C# კოდები

განვიხილოთ რამდენიმე C# კოდი Solution Explorer-იდან. მაგალითად Lectori.cs (ლისტინგი1.1) და Lectia.cs (ლისტინგი1.2).

```
//--ლისტინგი_1 ----Lectori --------------------------
// <auto-generated>
//------------------------------------------------------
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
public class Lectori : Tanamshromeli  // მემკვიდრეობითობა!!!
 {
    public virtual int lectori_ID
       { get; set; }
    public virtual int statusi_ID
       { get; set; }
    public virtual int kat_ID
       { get; set; }
    public virtual double countMoney()
       {
        throw new System.NotImplementedException();
       }
 }

//-- ლისტინგი_1.2  ----Akad_Sagani --------------------
// <auto-generated>
//------------------------------------------------------
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

public class Akad_Sagani
{
   public virtual int sagani_ID
       { get; set; }
   public virtual string dasaxeleba
       { get; set; }
   public virtual int crediti
       { get; set; }
```

```
    public virtual Sakurso_Projecti Sakurso_Projecti
                                                // !!! კომპოზიცია
        { get; set; }
    public virtual Laboratoria Laboratoria // !!! კომპოზიცია
        { get; set; }
    public virtual Practikuli Practikuli // !!! კომპოზიცია
        { get; set; }
    public virtual Lectia Lectia   // !!! კომპოზიცია
        { get; set; }
    public virtual void input()
        {
          throw new System.NotImplementedException();
        }
    public virtual void delete()
        {
          throw new System.NotImplementedException();
        }
    public virtual void modify()
        {
          throw new System.NotImplementedException();
        }
}
```

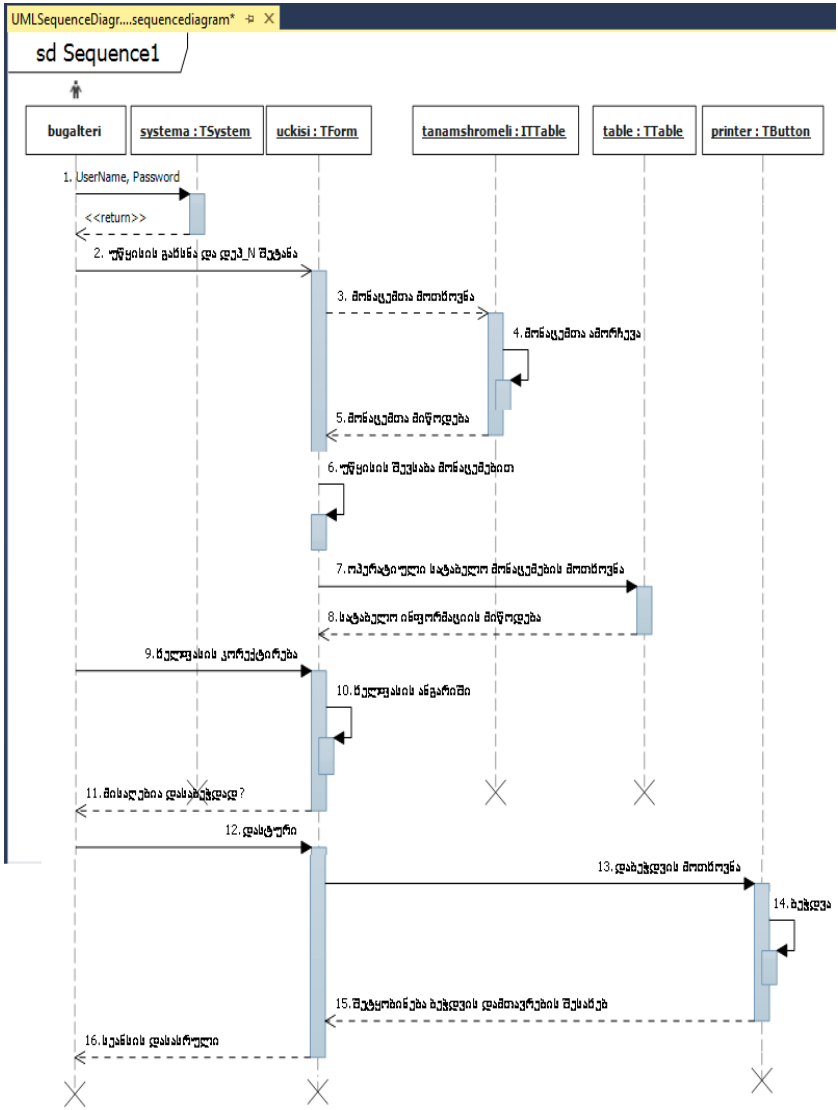
1.3. ინტერფეისები და სცენარები: მიმდევრობითობის დიაგრამა (Sequence-D)

სცენარი არის რომელიმე როლის (Actor) კლასებსა და მის ობიექტებთან მუშაობის პროცესის თანამიმდევრობის აღწერა კონკრეტული ამოცანის (Action) გადასაჭრელად კომპიუტერზე. ამგვარად, ჯერ უნდა აიგოს სცენარი, თუ როგორ იმუშავებს მომხმარებელი კომპიუტერთან და შემდეგ მოხდეს მისი პროგრამული რეალიზაცია (მაგალითად, WPF ტექნოლოგიით).

სცენარის ასაგებად ჩვენ გამოვიყენებთ მიმდევრობითობის (Sequence-D) დიაგრამას (ნახ.1.13).

მიმდევრობითობის დიაგრამაზე კი შეტყობინებები (Messages) და ოპერაციები დალაგებულია მათი შესრულების მიმდევრობით დროში.

ამოცანა: „ხელფასის დარიცხვის უწყისის მომზადება და ბეჭდვა“



ნახ.1.13. მიმდევრობითობის დიაგრამა (Sequence–D)

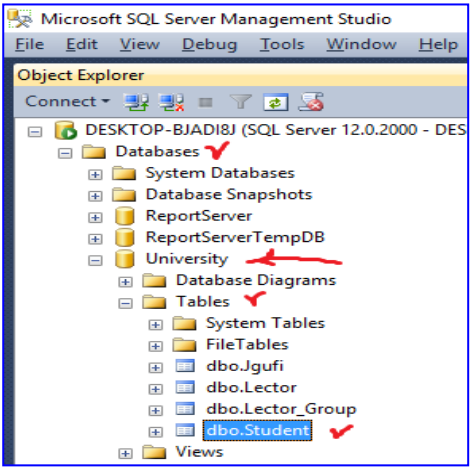
II თავი
მონაცემთა ბაზის აგება Ms SQL Server -ის გამოყენებით

2.1. მონაცემთა ბაზის და ცხრილების (Tables) შექმნა

პროგრამულ აპლიკაციას სჭირდება მონაცემთა ბაზა, რომელშიც შეინახავს ან რომლიდანაც ამოარჩევს მისთვის აუცილებელ ინფორმაციულ ფრაგმენტებს. ჩვენ შერჩეული გვაქვს MsSQL Server 2014 მონაცემთა ბაზების მართვის სისტემა, რომლის გამოყენებითაც უნდა შეიქმნას ბაზა და ცხრილები.

დავალება_1: ამოუშვით Ms SQL Server Management Studio პაკეტი და შექმენით თქვენი პროგრამული აპლიკაციისთვის საჭირო მონაცემთა ბაზა და ცხრილების სტრუქტურები.

მაგალითად, განვიხილოთ უნივერსიტეტის მონაცემთა ბაზა (University) ოთხი ცხრილით, რომლებშიც განთავსებულია ინფორმაცია სტუდენტების, ლექტორების, ჯგუფების და აკადემიური საგნების შესახებ (ნახ.2.1).



ნახ.2.1. მბ - University 4 ცხრილით

ცხრილში ***სტუდენტი*** (Student) პირველადი გასაღებური ატრიბუტია St_ID ინდექსი, მისი მეორეული გასაღები - Gr_Nom, რომლითაც უკავშირდება ცხრილს ***ჯგუფი*** (Jgufi), პირველადი ინდექსით Gr_Nom. ესაა კავშირი 1:N, რომელიც ასახავს ბიზნესწესს (არსებულ კანონზომიერებას), რომ ერთი სტუდენტი შეიძლება იყოს მხოლოდ ერთ ჯგუფში და ერთ ჯგუფში შეიძლება იყოს რამდენიმე (N) სტუდენტი. ასევე, ***ლექტორი*** (Lector) ასწავლის რამდენიმე (N) ჯგუფს, მაგრამ ჯგუფსაც ჰყავს რამდენიმე (M) ლექტორი. ესაა M:N კავშირი. მისი რეალიზაცია არაა შესაძლებელი ***ლექტორი***-ს და ***ჯგუფი***-ს პირდაპირი კავშირით (განმეორებადი ველების პრობლემა !). ამისათვის შემოტანილია დამატებითი ცხრილი (რელაცია) ***ლექტორი_ჯგუფი*** (Lector_Group). მასში შედგენილი ინდექსი იქნება L_JG, რომელიც უკავშირდება ცალ-ცალკე ***ლექტორს*** (L_ID) და ***ჯგუფს*** (Gr_Nom). 2.2 ა–დ ნახაზებზე მოცემულია ეს სტრუქტურები.

DESKTOP-BJADI8J.Un...sity - dbo.Student

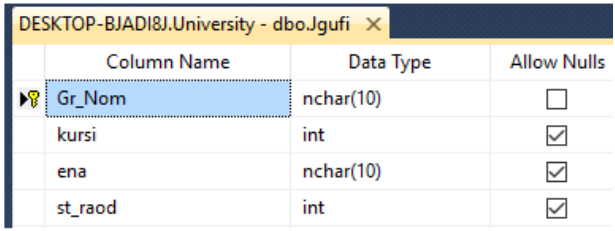
Column Name	Data Type	Allow Nulls
St_ID	int	☐
Name	nchar(10)	☑
Forname	nchar(10)	☑
Sex	nchar(10)	☑
Gr_Nom	nchar(10)	☑
Tel	nchar(10)	☑

ნახ.2.1-ა. „Student“ ცხრილის სტრუქტურა

DESKTOP-BJADI8J.Un...rsity - dbo.Lector

Column Name	Data Type	Allow Nulls
L_ID	int	☐
Name	nchar(25)	☑
FirstName	nchar(15)	☑
Age	int	☑
Sex	nchar(10)	☑
Status	nchar(20)	☑
Phone	nchar(10)	☑

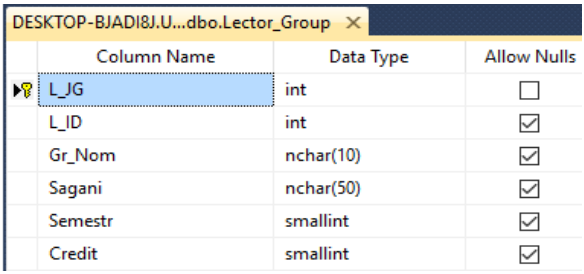
ნახ.2.1-ბ. „ლექტორი“ცხრილის სტრუქტურა



DESKTOP-BJADI8J.University - dbo.Jgufi

Column Name	Data Type	Allow Nulls
Gr_Nom	nchar(10)	☐
kursi	int	☑
ena	nchar(10)	☑
st_raod	int	☑

ნახ.2.1-გ. „ჯგუფი“ ცხრილის სტრუქტურა



DESKTOP-BJADI8J.U...dbo.Lector_Group

Column Name	Data Type	Allow Nulls
L_JG	int	☐
L_ID	int	☑
Gr_Nom	nchar(10)	☑
Sagani	nchar(50)	☑
Semestr	smallint	☑
Credit	smallint	☑

ნახ.2.1-დ. „ლექტორის_ჯგუფი“ ცხრილის სტრუქტურა

ასევე შესაძლებელია სხვა ცხრილების დამატება მონაცემთა ბაზაში, როგორიცაა მაგალითად, კათედრა, ფაკულტეტი, ხელფასის უწყისი, გამოცდა, სესიის შედეგები და ა.შ.

2.2. ბაზის ცხრილების შევსება მონაცემებით

მონაცემთა ბაზის აგების მომდევნო ეტაპზე საჭიროა ჩვენ მიერ შექმნილი ცხრილების შევსება ჩანაწერებით, რომლებიც ასახავს რეალურ (ან კვაზირეალურ) სიტუაციას.

დავალება_2. Ms_SQL Server ბაზის ცხრილებში შეიტანეთ მონაცემები. სტრიქონების რაოდენობა უნდა იყოს საკმარისი ექსპერიმენტების ჩასატარებლად (მინიმუმ 5 სტრიქონი).

2.2, ა–დ ნახაზებზე ნაჩვენებია შევსებული ცხრილები.

DESKTOP-BJADI8J.Un...sity - dbo.Student

St_ID	Name	Forname	Sex	Gr_Nom	Tel
1	დოლიძე	დიტო	მამრ	108150	222-22-22
2	გულუა	დათო	მამრ	108150	233-23-23
3	დოლიძე	გიორგი	მამრ	108155	255-55-55
4	თურქია	ქეთეკან	მდედრ	108153	277-77-77
5	თოფურია	ეკა	მდედრ	108150	239-39-39
10	ბახია	გიო	მამრ	108153	231-31-31
11	დვალი	დათო	მამრ	108251	211-11-11
12	ბურძგლა	ყვარყვარე	მამრ	108251	211-12-12
13	ბურძგლა	ოტელო	მამრ	108252	222-21-21
15	ჯაში	ჩარიტა	მდედრ	108253	222-55-55
16	ჯანაშია	მაჟუ	მდედრ	108253	222-33-33
17	ზვიადაური	ჯოყოლა	მამრ	108252	211-22-33

ნახ.2.2-ა. ცხრილი „სტუდენტი“

DESKTOP-BJADI8J.Un...rsity - dbo.Lector

L_ID	Name	FirstName	Age	Sex	Status	Phone
1	გოგიჩაიშვილი	გიორგი	50	მამრ	პროფესორი	221-11-11
2	დიდმანიძე	ვაჟა	45	მამრ	პროფესორი	222-21-21
3	ჩაჩანიძე	გურამ	40	მამრ	პროფესორი	222-23-23
4	ოდიშარია	კორნელი	40	მამრ	პროფესორი	233-33-33
5	თოფურია	ნინო	30	მდედრ	ასოც.პროფესორი	230-30-30
6	თურქია	ეკა	35	მდედრ	პროფესორი	237-37-37

ნახ.2.2-ბ. ცხრილი „ლექტორი“

DESKTOP-BJADI8J.University - dbo.Jgufi

Gr_Nom	kursi	ena	st_raod
108150	4	ქართული	28
108151	4	ქართული	30
108153	4	ინგლისური	14
108155	4	რუსული	18
108250	3	ქართული	24
108251	3	ქართული	20
108252	3	ინგლისური	24
108253	3	ქართული	28

ნახ.2.2-გ. ცხრილი „ჯგუფი“

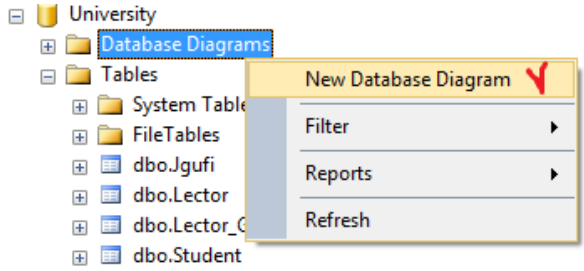
DESKTOP-BJADI8J.U...dbo.Lector_Group

L_JG	L_ID	Gr_Nom	Saqani	Semestr	Credit
1	1	108250	UML ტექნოლოგია	2	5
2	1	108251	UML ტექნოლოგია	2	5
3	2	108150	ინფორმაციული საზოგადოება	2	5
4	2	108251	ინფორმაციული საზოგადოება	2	5
5	2	108152	ინფორმაციული საზოგადოება	2	5
6	2	108155	ინფორმაციული საზოგადოება	2	5
7	3	108250	მართვის ავტომატიზებული მოდელები	1	4
8	3	108251	მართვის ავტომატიზებული მოდელები	1	4
9	4	108150	უსაფრთხოების სისტემები	2	5
10	4	108151	უსაფრთხოების სისტემები	2	5
11	4	108153	უსაფრთხოების სისტემები	2	5
12	5	108250	მონაცემთა ბაზის სისტემა SQL Server	2	5
13	5	108251	მონაცემთა ბაზის სისტემა SQL Server	2	5
14	11	108252	მონაცემთა ბაზის სისტემა SQL Server	2	5
15	11	108253	მონაცემთა ბაზის სისტემა SQL Server	2	5
16	9	108252	UML ტექნოლოგია	2	5
17	9	108253	UML ტექნოლოგია	2	5
18	10	108250	ვიზუალური დაპროგრამება C#	1	5
19	10	108251	ვიზუალური დაპროგრამება C#	1	5
20	10	108252	ვიზუალური დაპროგრამება C#	1	5
21	10	108253	ვიზუალური დაპროგრამება C#	1	5
22	10	108250	პროგრამული სისტემების მენეჯმენტი	2	5
23	10	108251	პროგრამული სისტემების მენეჯმენტი	2	5
24	6	108252	პროგრამული სისტემების მენეჯმენტი	2	5
25	6	108253	პროგრამული სისტემების მენეჯმენტი	2	5

ნახ.2.2-დ. ცხრილი „ლექტორის_ჯგუფი“
საგნების მიხედვით

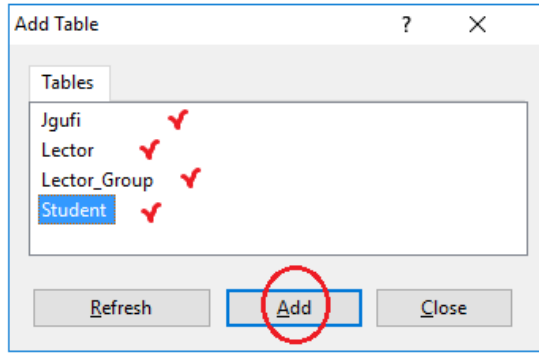
2.3. მონაცემთა ბაზის ცხრილთაშორის კავშირების აგება (Relationships)

მონაცემთა ბაზის ცხრილთაშორის კავშირების ასაგებად MsSQL Server Management Studio-აქვე მენიუდან გამოვიყენოთ Database Tools -> Relationships (ნახ.3.3–ა).



ნახ.2.3. ცხრილთაშორის კავშირების აგების სტრიქონი

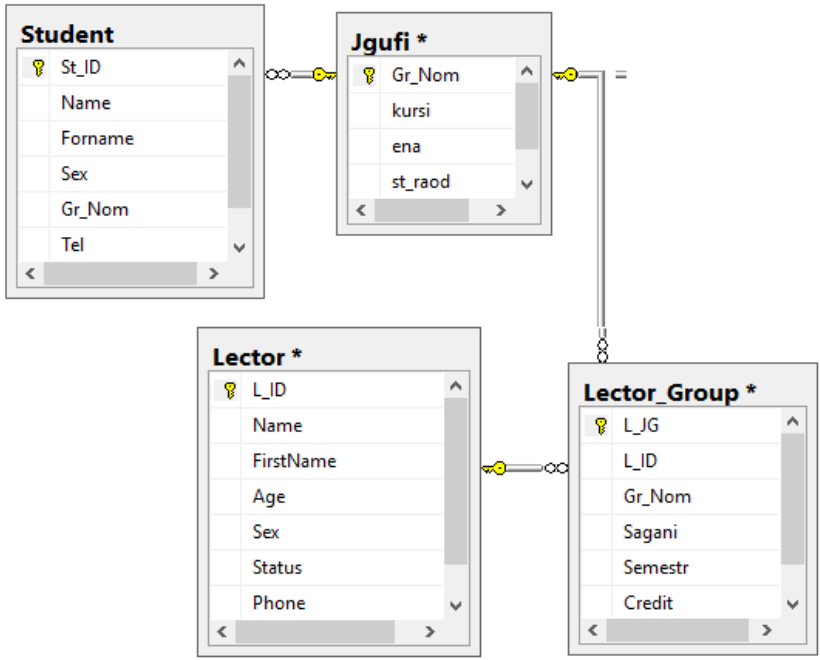
შედეგად სისტემა მოგვცემს ცხრილების არჩევის საშუალებას (2, 3 ან ყველა) Add ღილაკით (ნახ.2.4).



ნახ.2.3

ჩვენი შემთხვევისათვის ავირჩიოთ მიმდევრობით ყველა ცხრილი. უნდა მივიღოთ ისეთი სახის რელაციური კავშირების დიაგრამა, როგორიც მოცემულია 2.4 ნახაზზე. თუ რომელიმე

კავშირი არაა, მაშინ იგი ჩვენ თვითონ მაუსის დახმარებით უნდა შევაერთოთ. სისტემა ამოწმებს დაკავშირების მართებულობას.



ნახ.2.4. ცხრილთაშორის კავშირების დიაგრამა

ამგვარად, ჩვენ შევქმენით University მონაცემთა ბაზა MS SQL Server 2014 პაკეტის დახმარებით და იგი მზადაა მომავალი გამოყენებისთვის.

ამჯერად საჭიროა აიგოს მომხმარებლის ინტერფეისი Ms Visual Studio .NET 2015 პლატფორმაზე და მოხდეს მონაცემთა ბაზასთან ინტერაქტიული პროცედურების შესრულება. კერძოდ WPF აპლიკაციიდან მონაცემების ამორჩევა და მონაცემთა ბაზის განახლება C#-ის Insert, Update და Delete მეთოდებით.

ამ საკითხებს განვიხილავთ მომდევნო თავში.

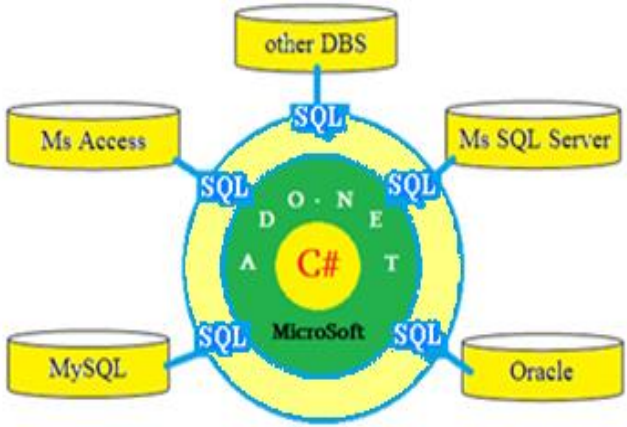
III თავი

მომხმარებელთა ინტერფეისის დამუშავება

3.1. პროგრამული აპლიკაციების კავშირი მონაცემთა ბაზებთან

მომხმარებელთა პროგრამულ აპლიკაციებს (დანართებს) აუცილებლად ესაჭიროება ურთიერთქმედება მონაცემთა ცენტრალიზებულ ან ლოკალურ ბაზებთან, როდესაც ისინი კლიენტის მანქანებზე ფუნქციონირებს.

Ms Visual Studio .NET Framework პლატფორმა მომხმარებელთა ინტერფეისების დასაკავშირებლად მონაცემთა ბაზებთან იყენებს ADO.NET ტექნოლოგიას (ActiveX Data Object) და SQL ენას (ნახ.3.1). პირველი არის დამაკავშირებელი დრაივერი C# (ან სხვა) ენასა და ბაზებს შორის, მეორე კი - მომხმარებლის საკონტაქტო ენაა ბაზებთან ე.წ. სტრუქტურირებულ მოთხოვნათა ენა [1,7]. აქ მას განვიხილავთ მოკლედ, გაცნობის დონეზე.



ნახ.3.1. C# <-> ADO.NET <-> SQL <–> DBS

მონაცემებთან მიმართვის ტრადიციული ტექნოლოგიები, ჩვეულებრივ, ახორციელებდა მონაცემების წვდომას წყაროსთან მუდმივი მიერთების გზით. ასეთი მოდელის გამოყენებისას პროგრამული აპლიკაცია გახსნის მონაცემთა ბაზასთან მიერთებას და არ დახურავს მუშაობის დამთავრებამდე.

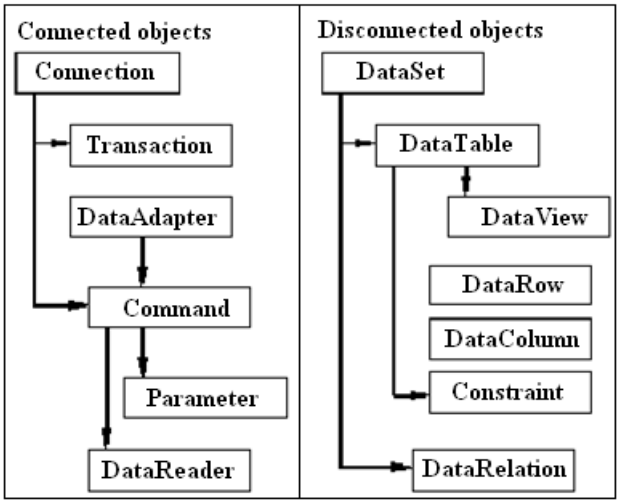
დანართის სირთულის ზრდასთან ერთად იზრდება მონაცემთა ბაზის კლიენტების რაოდენობაც, რაც არაეფექტურს ხდის ბაზასთან მუდმივი მიერთების ტექნოლოგიას. მაგალითად, აპლიკაცია კარგად ემსახურება ორ მიერთებულ კლიენტს, 10-თან უკვე უჭირს მუშაობა და 100-თან საერთოდ ვერ ფუნქციონირებს.

ADO.NET სისტემაში ეს პრობლემები წყდება მონაცემებთან მიმართვის ისეთი მოდელის გამოყენებით, როგორიცაა **გამოყოფილი** მონაცემები. ასეთი მოდელის შემთხვევაში მონაცემთა წყაროსთან მიერთება გახსნილია მხოლოდ გარკვეული პროცედურების შესასრულებლად. მაგალითად, თუ აპლიკაციას დასჭირდა მონაცემები ბაზიდან, იგი მიუერთდება მას ამ მონაცემების გადმოტვირთვამდე, შემდეგ კი მიერთება დაიხურება.

ასევე, როდესაც ხორციელდება მონაცემთა განახლება ბაზაში, მიერთება წყაროსთან განხორციელდება UPDATE ბრძანების შესრულების დამთავრებამდე, შემდეგ იგი დაიხურება. ამგვარად, მონაცემებთან მიერთების დროის (გახსნა-დახურვის პერიოდი) შემცირებით, ADO.NET უზრუნველყოფს სისტემური რესურსების ეკონომიურ გამოყენებას და მონაცემთა წვდომის ინფრასტრუქტურის მასშტაბირებას, მწარმოებლურობის შემცირების გარეშე.

3.2. ADO.NET დრაივერის არქიტექტურა

3.2 ნახაზზე ნაჩვენებია ADO.NET ობიექტური მოდელის შემადგენელი კლასები, რომელთა დანიშნულებასაც მოკლედ შევეხებით ამ პარაგრაფში. სისტემის ობიექტური მოდელი ორი ნაწილისგან შედგება: მარცხენა – მიერთებადი ობიექტები (Connected Objects) და მარჯვენა – განცალკევებადი ობიექტები (Disconnected Objects).



ნახ.3.2

მონაცემებთან მიმართვა ADO.NET-ში ხორციელდება ორი კომპონენტით:

- მონაცემთა ერთობლიობით (DataSet ობიექტით), რომელშიც მონაცემები ინახება ლოკალურ კომპიუტერში;
- მონაცემთა მიმწოდებლით (DataProvider პროვაიდერით), რომელიც ასრულებს შუამავლის ფუნქციას პროგრამასა და მონაცემთა ბაზას შორის.

ობიექტი DataSet. ესაა მონაცემთა წარმოდგენა (View) კომპიუტერის მეხსიერებაში მონაცემთა წყაროსგან იზოლირებულად. ეს ობიექტი შეიძლება განვიხილოთ, როგორც მონაცემთა ბაზის ფრაგმენტის ლოკალური ასლი (კოპიო).

DataSet-ში მონაცემთა ჩატვირთვა შესაძლებელია ნებისმიერი დასაშვები წყაროდან, მაგალითად, Ms SQL Server ბაზიდან ან XML ფაილიდან. დასაშვებია მეხსიერებაში ამ მონაცემებით მანიპულირება, აგრეთვე მათი განახლება მთავარი წყაროსაგან დამოუკიდებლად.

ობიექტი DataSet შედგება DataTable ობიექტთა ერთობლიობისგან (ის შეიძლება ცარიელიც იყოს ანუ არ შეიცავდეს არც ერთ DataTable-ს).

ყოველი DataTable ობიექტი კომპიუტერის მეხსიერებაში ასახავს ერთ ცხრილს. მისი სტრუქტურა შეიცავს ორ ერთობლიობას: DataColumns, რომელშიც თავსდება ცხრილის სვეტები და ცხრილის შეზღუდვათა ერთობლიობა. ეს ორი ერთობლიობა ქმნის ცხრილის სქემას.

DataTable ობიექტი შეიცავს აგრეთვე DataRows ერთობლიობას, რომელშიც ინახება DataSet ობიექტის მონაცემები.

გარდა ამისა, DataSet ობიექტი შეიცავს DataRelations ერთობლიობას, რომელიც უზრუნველყოფს კავშირის შექმნას სხვადასხვა ცხრილის სტრიქონებს შორის. DataRelations შეიცავს DataRelation ობიექტთა ერთობლიობას, რომლებიც განსაზღვრავს ცხრილთაშორის კავშირებს (მაგალითად, 1:M კავშირის სარეალიზაციოდ).

დაბოლოს, DataSet ობიექტი შეიცავს ExtendedProperties ერთობლიობას, რომელშიც შეინახება დამატებითი მონაცემები.

მონაცემთა პროვაიდერი. ესაა ურთიერთდაკავშირებულ კომპონენტთა ერთობლიობა, რომელიც უზრუნველყოფს ეფექტურ მაღალმწარმოებლურ კავშირს მონაცემთა ბაზასთან.

.NET Framework-ს აქვს ორი პროვაიდერი: SQL Server .NET Data Provider, რომელიც შექმნილია SQL Server 7.0 ან უფრო მაღალ ვერსიებთან სამუშაოდ და OleDb .NET Data Provider სხვა ტიპის მონაცემთა ბაზებთან დასაკავშირებლად.

მონაცემთა ნებისმიერი პროვაიდერი შედგება მსგავსი უნივერსალური კლასების კომპონენტებისგან:

- Connection, რომელიც უზრუნველყოფს მონაცემთა ბაზასთან მიერთებას;

- Command, რომელიც გამოიყენება მონაცემთა წყაროს სამართავად. იგი გამოიყენებს ბრძანებებს, რომლებიც არ აბრუნებს მონაცემებს, მაგალითად, INSERT, UPDATE და DELETE ან ბრძანებებს, რომლებიც აბრუნებს DataReader ობიექტს (მაგალითად, SELECT);

- DataReader გამოიყენება მხოლოდ ჩანაწერთა ერთობლიობის წასაკითხად მიერთებული მონაცემთა წყაროდან;

- DataAdapter შეავსებს გამოყოფილ DataSet ან DataTable ობიექტს და განაახლებს მათ შედგენილობას.

მონაცემებთან მიმართვა ხორციელდება შემდეგნაირად: ობიექტი Connection აყენებს დანართის (აპლიკაციის) მონაცემთა ბაზასთან მიერთებას, რომელიც პირდაპირ მისაწვდომია Command და DataAdapter ობიექტებისთვის. Command ობიექტი უზრუნველყოფს ბრძანებათა შესრულებას უშუალოდ მონაცემთა ბაზაში. თუ შესასრულებელი ბრძანება აბრუნებს რამდენიმე მნიშვნელობას, მაშინ Command ხსნის მათთან მიმართვას DataReader ობიექტის საშუალებით. მიღებული შედეგები შესაძლებელია დამუშავდეს უშუალოდ დანართის კოდით ან

DataSet ობიექტით, რომელიც შეივსება DataAdapter ობიექტის დახმარებით. მონაცემთა ბაზის განახლებისთვის ასევე გამოიყენება Command და DataAdapter ობიექტები.

ობიექტი Connection გვთავაზობს მიერთებას მონაცემთა ბაზასთან. Visual Studio .NET-ს აქვს Connection-ის ორი კლასი: SqlConnection (MsSQL_Server-თან შესაერთებლად) და OleDbConnection (სხვა ტიპის მონაცემთა ბაზებთან დასაკავშირებლად). მონაცემთა ბაზასთან კავშირის არხის გასახსნელი აუცილებელი მონაცემები ინახება Connection ობიექტის ConnectionString თვისებაში. ეს ობიექტი ინახავს აგრეთვე მეთოდებს, რომლებიც საჭიროა მონაცემთა დასამუშავებლად ტრანზაქციების გამოყენებით.

Command ობიექტს აქვს ორი კლასი: SqlCommand და OleDbCommand. იგი უზრუნველყოფს ბრძანებათა გამოყენებას მონაცემთა ბაზაზე, რომელთანაც დამყარებულია კავშირი (მიერთება). აქ შეიძლება გამოყენებულ იქნეს შენახვადი პროცედურები (Stored Procedures), SQL-ენის ბრძანებები, აგრეთვე ოპერატორები მთლიანი ცხრილების მისაღებად. Command ობიექტს აქვს სამი მეთოდი:

- Execute Non Query: იყენებს ბრძანებებს, რომლებიც არ აბრუნებს მონაცემებს, მაგალითად, INSERT, UPDATE და DELETE;
- Execute Scalar: იყენებს მოთხოვნებს მონაცემთა ბაზისადმი, რომლებიც აბრუნებს მხოლოდ ერთ მნიშვნელობას;
- Execute Reader: აბრუნებს საშედეგო ერთობლიობას DataReader ობიექტის საშუალებით.

ობიექტი DataReader გვთავაზობს ნაკადს მონაცემთა ბაზის ჩანაწერების ერთობლიობით, ოღონდ მხოლოდ ერთი მიმართულებით წასაკითხად. მონაცემთა პროვაიდერის სხვა კომპონენტებისგან განსხვავებით DataReader-ის ეგზემპლარების

შექმნა პირდაპირ არაა დასაშვები. მისი მიღება შეიძლება Command ობიექტის ExecuteReader მეთოდებით:

- SqlCommand.ExecuteReader მეთოდი აბრუნებს SqlDataReader ობიექტს;
- OleDbCommand.ExecuteReader მეთოდი კი - OleDbDataReader ობიექტს.

თუ DataReader ობიექტის შემცველი მონაცემების ჩაწერა დისკზე არაა საჭირო, მაშინ ეს სტრიქონები შეიძლება პირდაპირ გადაეგზავნოს დანართს. ვინაიდან დროის ნებისმიერ მომენტში მეხსიერებაში იმყოფება მხოლოდ ერთი სტრიქონი, DataReader ობიექტის გამოყენება თითქმის არ ამცირებს სისტემის მწარმოებლურობას, ოღონდ მოითხოვს მონოპოლიურ მიმართვას გახსნილ Connection ობიექტზე DataReader ობიექტის სასიცოცხლო დროის განმავლობაში.

ობიექტი DataAdapter არის ADO.NET-ის ძირითადი კლასი, რომელიც უზრუნველყოფს გამოყოფილ მონაცემებთან მიმართვას. არსებითად, იგი ასრულებს შუამავლის ფუნქციებს მონაცემთა ბაზისა და DataSet ობიექტის ურთიერთ-ქმედებისთვის.

Fill მეთოდის გამოძახებისას DataAdapter ობიექტი მონაცემებით შეავსებს DataTable-ს ან DataSet-ს მონაცემთა ბაზიდან. მონაცემების დამუშავების შემდეგ, რომლებიც ჩატვირთულია მეხსიერებაში, შესაძლებელია მოდიფიცირებული ჩანაწერების მოთავსება მონაცემთა ბაზაში, DataAdapter ობიექტის Update მეთოდის გამოძახებით. DataAdapter-ს აქვს ოთხი თვისება, რომლებიც მონაცემთა ბაზის ბრძანებებია:

- SelectCommand შეიცავს ტექსტს ან ბრძანების ობიექტს, რომელიც ახორციელებს მონაცემთა ბაზიდან ამორჩევას (მაგალითად, მეთოდი Fill);

- InsertCommand შეიცავს ტექსტს ან ბრძანების ობიექტს, რომელიც ახორციელებს სტრიქონის ჩასმას ცხრილში;

- DeleteCommand შეიცავს ტექსტს ან ბრძანების ობიექტს, რომელიც ახორციელებს სტრიქონის წაშლას ცხრილიდან;

- UpdateCommand შეიცავს ტექსტს ან ბრძანების ობიექტს, რომელიც ახორციელებს მნიშვნელობათა განახლებას მონაცემთა ბაზაში.

Update მეთოდის გამოძახებისას ყველა შეცვლილი მონაცემი კოპირდება DataSet ობიექტიდან მონაცემთა ბაზაში, შესაბამისი ბრძანებების - InsertCommand, DeleteCommand ან UpdateCommand გამოყენებით.

3.3. ინტერფეისის კავშირი მონაცემთა ბაზასთან

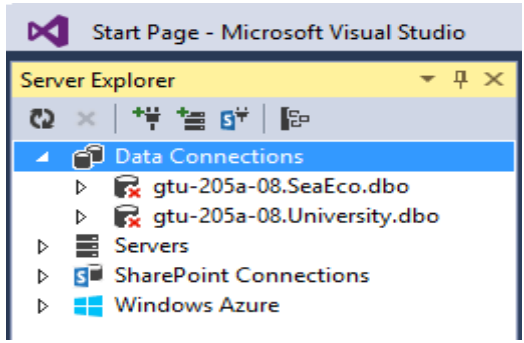
Visual Studio .NET სისტემას აქვს სტანდარტული ოსტატი პროგრამებისა და დიზაინერების სიმრავლე, რომელთა საშუალებითაც ადვილად და ეფექტურად ხორციელდება მონაცემებთან წვდომის არქიტექტურა აპლიკაციების დამუშავების პროცესში. ამასთან, ADO.NET ობიექტური მოდელის ყველა შესაძლებლობა მისაწვდომია პროგრამულად, რაც უზრუნველყოფს არასტანდარტული ფუნქციების რეალიზაციის ან დანართების აგების შესაძლებლობას, რომელიც მომხმარებლის მოთხოვნილებებზეა ორიენტირებული.

აქ გავეცნობით, როგორ დავუკავშირდეთ მონაცემთა ბაზას ADO.NET-ის გამოყენებით, როგორ ამოვიღოთ საჭირო მონაცემები და გადავცეთ პროგრამულ აპლიკაციას. ეს საკითხები შეიძლება შესრულდეს Visual Studio .NET-ის გრაფიკული ინსტრუმენტებით და პროგრამულად.

C# პროგრამულ აპლიკაციაში არსებობს მონაცემთა ბაზასთან მიერთების რამდენიმე ხერხი. ამის განხორციელება ყველაზე

მარტივია Visual Studio .NET-ის გრაფიკული ინსტრუმენტით. მონაცემთა წყაროსთან (DataSource) მიერთებისა და მართვისათვის გამოიყენება ფანჯარა Server Explorer.

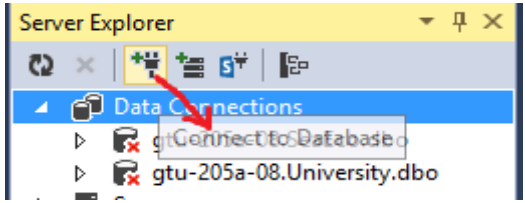
ძირითადი ამოცანა, რომელსაც აქ განვიხილავთ, არის ADO.NET პროგრამული პაკეტის გამოყენებით მომხმარებელთა სამუშაო ინტერფეისის დამუშავების სადემონსტრაციო მაგალითის აგება. ამასთანავე, მონაცემთა ბაზების სახით უნდა გამოვიყენოთ Ms SQL Server პაკეტით აგებული ცხრილები. .NET სამუშაო გარემოს ჩატვირთვის შემდეგ საჭიროა Server Explorer-ის გახსნა და ბაზებთან კავშირის შემოწმება (მაგალითად, ნახ.3.3).



ნახ.3.3. არაა კავშირი ბაზასთან

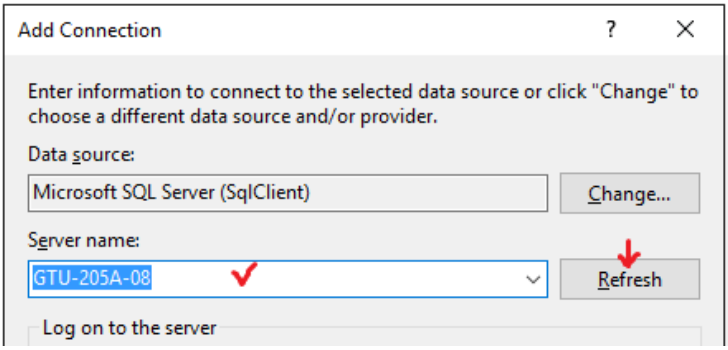
ახლა განვიხილოთ MsSQL Server ბაზასთან მუშაობის საკითხები.

Server Explorer-ის გრაფიკული მენიუდან ავირჩიოთ Connect to Database (ნახ.3.4).



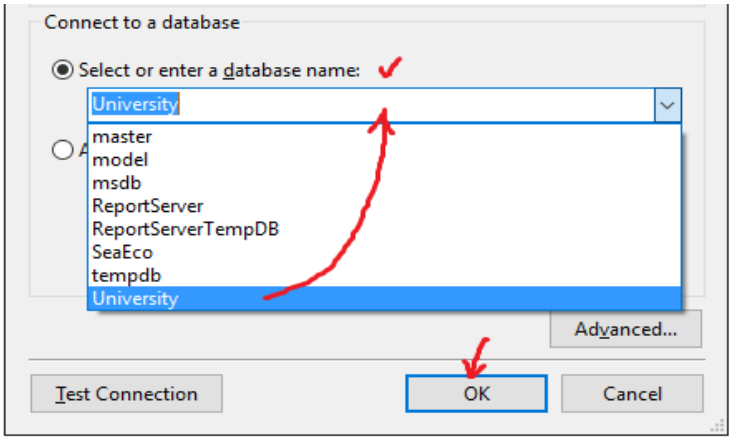
ნახ.3.4. დაკავშირების პროცესი

3.5 ფანჯრის Server name ველში ჩავწეროთ GTU-205A-08 და Refresh ღილაკით ავამუშავოთ პროცესი.



ნახ.3.5

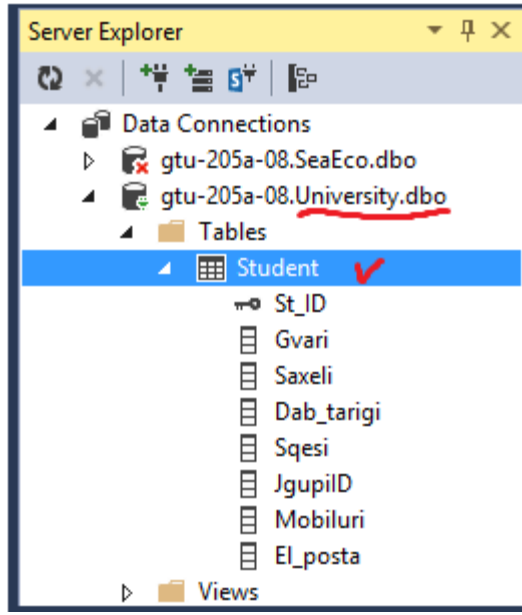
ამის შემდეგ Add Connection-ის ქვედა ნაწილში, კერძოდ, Select or enter Database name ველში ავირჩევთ სერვერზე არსებული ბაზებიდან ჩვენთვის საჭიროს (ნახ.3.6).



ნახ.3.6

საჭიროების შემთხვევაში მიეთითება User name და Password. ამჯერად ჩვენ ეს არ გვჭირდება.

შედეგად Server Explorer-ში გამოჩნდება 3.7 ნახაზზე მოცემული სურათი.



ნახ.3.7. კავშირი შედგა University ბაზასთან

როგორც ვხედავთ, Data Connection-ში უნივერსიტეტის მონაცემთა ბაზის ფაილი **University.dbo** გამოჩნდა, რომლის Tables შეიცავს Student ცხრილს, ველებით St_ID (პირველადი გასაღებურია) და სხვ. აქ JgupiID (მეორეული გასაღებურია).

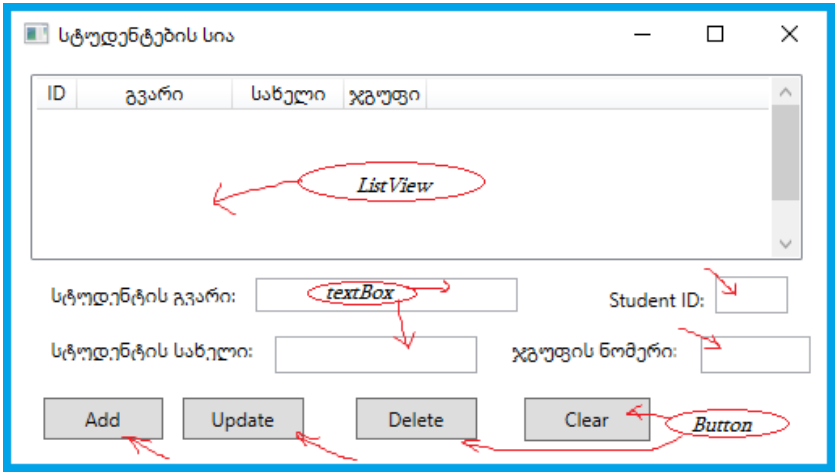
ამგვარად, University მონაცემთა ბაზა მზადაა პროგრამულ აპლიკაციასთან სამუშაოდ.

3.4. WPF აპლიკაციის კოდიდან SQL Server ბაზის წვდომა და მოთხოვნების დამუშავება

ახლა განვიხილოთ მომხმარებლის ინტერფეისის აგება WPF ტექნოლოგიით. იგი შედგება დიზაინის ეტაპის (ინტერფეისის სტრუქტურის და შიგთავსის განსაზღვრა) და ლოგიკური ნაწილის (საჭირო მეთოდების C# კოდები) პროგრამული რეალიზაციისაგან.

დავსვათ ამოცანა: საჭიროა მომხმარებლის ინტერფეისის აგება ListView მართვის ელემენტის გამოყენებით, რომელშიც აისახებაSQL Server-ის University.dbo მონაცემთა ბაზის განსაზღვრული ატრიბუტების (ველების) მნიშვნელობები (ბაზის ჩანაწერები) და შესაძლებელი იქნება ინტერფეისის Add, Update და Delete ღილაკებით მონაცემთა მანიპულირება (ბაზის განახლების ოპერაციების ჩატარება).

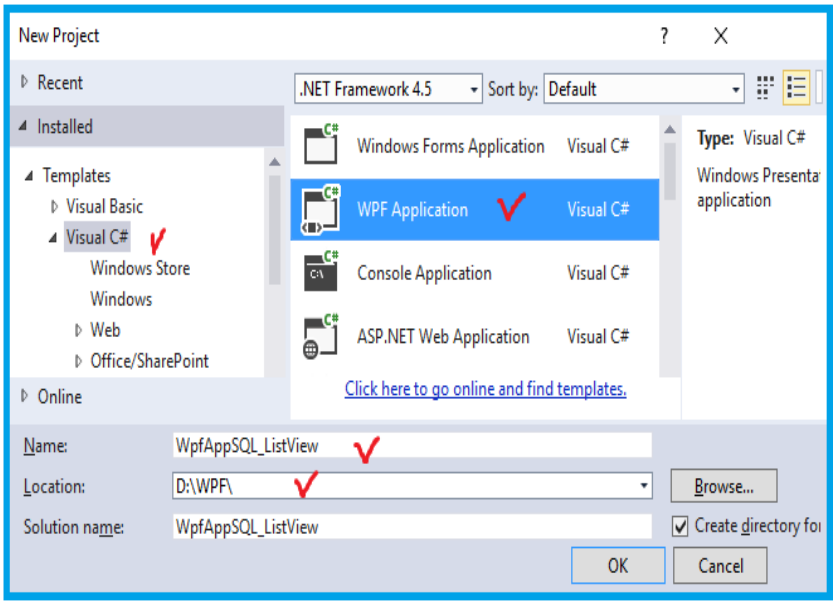
3.8 ნახაზზე მოცემულია ინტერფეისის მოდელი Visual Studio .NET 2015 სამუშაო გარემოში, Visual C# და WPF Application-ის საფუძველზე.



ნახ.3.8. მომხმარებლის ინტერფეისის მოდელი

- **პროექტის აგება.**

ავამუშავოთ Visual Studio .NET და შევქმნათ ახალი პროექტი სახელით WpfAppSQL_ListView, რომელსაც მოვათავსებთ D:\\ დისკოს WPF ფოლდერში (ნახ.3.9).

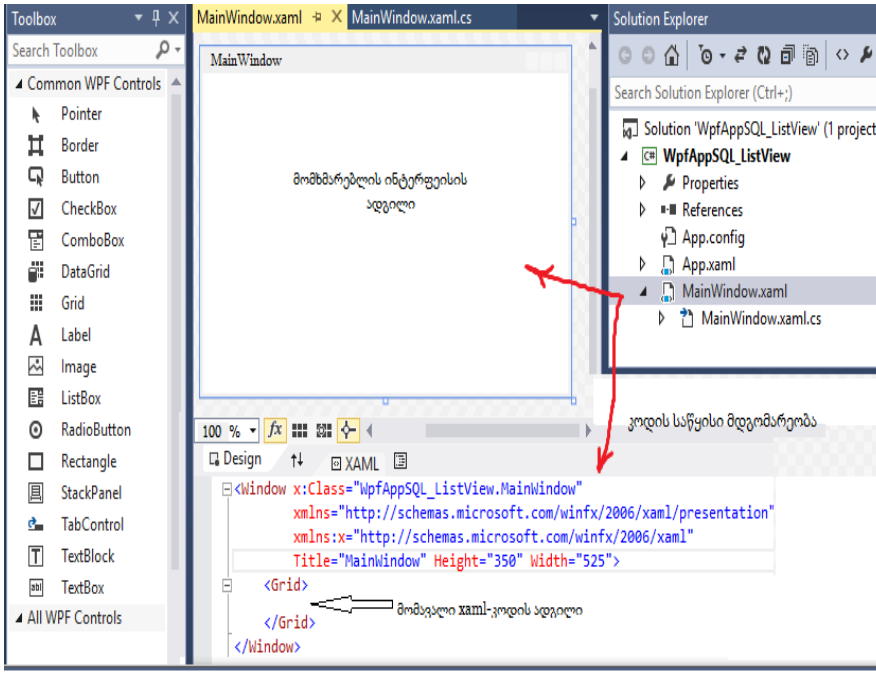


ნახ.3.9. პროექტის შექმნა

შედეგად ვღებულობთ 3.10 ნახაზზე მოცემულ მდგომარეობას. ინტერფეისზე უნდა გადმოვიტანოთ Toolbox პანელიდან ელემენტები ისე, როგორც 3.8 ნახაზზეა ნაჩვენები.

დაგვჭირდება ერთი ListView, ოთხი textBox და ოთხი Button. ბოლოს, როდესაც ინტერფეისის დიზაინი იქნება მზად, xaml ფაილი მიიღებს 3.1 ლისტინგში ნაჩვენებ სახეს. ფორმის Properties-ში Title-ს მნიშვნელობა შევცვალოთ ტექსტით: „სტუდენტების სია“

ღილაკების მოვლენების შესაბამისი მეთოდები (კოდები) ჯერ არაა დაწერილი C# ენაზე.



ნახ.3.10. WPF პროექტის საწყისი მდგომარეობა

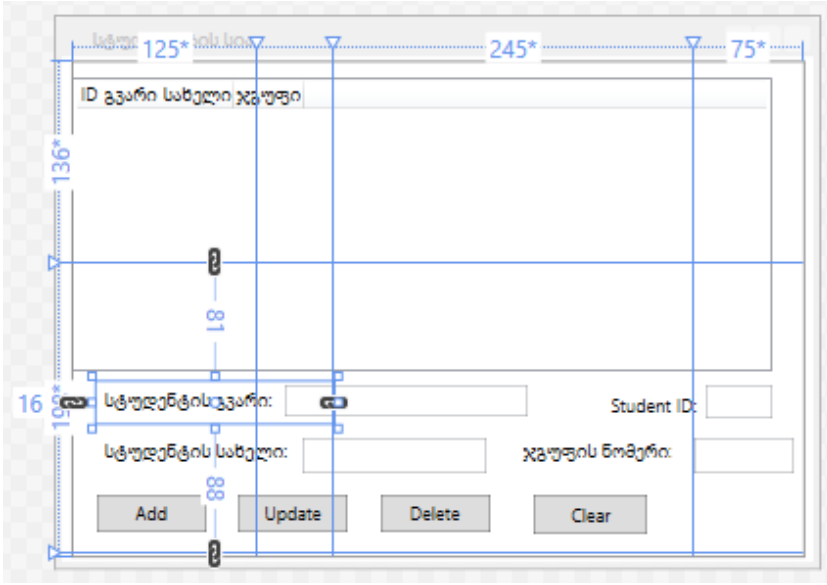
```
<!-- ლისტინგი_3.1 ----- mainWindow.xaml -------------------------
<Window x:Class="WpfAppSQL_ListView.MainWindow"
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml
                /presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  Title ="სტუდენტების სია" Height="375" Width="520"
                        Loaded="Window_Loaded" Background="White">
  <Grid Height="336" Width="497" Background="White" >
    <Grid.ColumnDefinitions>
      <ColumnDefinition Width="125*" />
      <ColumnDefinition Width="52*" />
      <ColumnDefinition Width="245*" />
```

```
      <ColumnDefinition Width="75*" />
    </Grid.ColumnDefinitions>
    <Grid.RowDefinitions>
      <RowDefinition Height="136*" />
      <RowDefinition Height="198*" />
      <RowDefinition Height="2*" />
    </Grid.RowDefinitions>
    <ListView Margin="-1,10,21,124" Name="listView1"
          ItemsSource="{Binding}" MinWidth="250" MinHeight="100"
                  Grid.ColumnSpan="4" Grid.RowSpan="2">
     <ListView.View>
      <GridView>
            <GridViewColumn Header="ID" DisplayMemberBinding=
                  "{Binding Path=St_ID}"></GridViewColumn>
       <GridViewColumn Header="გვარი" DisplayMember
                Binding="{Binding Path=Gvari}"></GridViewColumn>
       <GridViewColumn Header="სახელი" DisplayMember
              Binding="{Binding Path=Saxeli}"></GridViewColumn>
       <GridViewColumn Header="ჯგუფი" DisplayMemberBin
                ding="{Binding Path=JgupiID}"></GridViewColumn>
      </GridView>
      </ListView.View>
    </ListView>
<TextBox Margin="19,84,113,93" Name="textBox1" Grid.Row="1"
            Grid.Column="1" Grid.ColumnSpan="2" />
<TextBox Height="23" Margin="0,0,7,55" Name="textBox2"
        VerticalAlignment="Bottom" Grid.Row="1" Horizontal
            Alignment="Right" Width="68" Grid.Column="3" />
<Label Margin="16,81,0,88" Name="label1" Grid.Row="1"
      Content="სტუდენტის გვარი:" Grid.ColumnSpan="2"> </Label>
```

```
<Label Height="29" Margin="124,0,10,54" Name="label2"
      VerticalAlignment="Bottom" Grid.Row="1" Content="ჯგუფის
           ნომერი:" Grid.Column="2"/>
<Button Height="26" Margin="16,0,0,14" Name="btnAdd"
           VerticalAlignment="Bottom" Click="btnAdd_Click"
           Grid.Row="1" HorizontalAlignment="Left"
           Width="74">Add</Button>
<Button Height="26" Margin="112,0,0,14" Name="btnUpdate"
          VerticalAlignment="Bottom" Click="btnUpdate_Click"
          HorizontalAlignment="Left" Width="75"
          Grid.ColumnSpan="3" Grid.Row="1">Update</Button>
<Button Height="26" Margin="32,0,0,14" Name="btnDelete"
          VerticalAlignment="Bottom" Click="btnDelete_Click"
        Grid.Column="2" HorizontalAlignment="Left" Width="75"
        Grid.Row="1">Delete</Button>
<Button Height="27" Margin="136,0,31,13" Name="btnClear"
        VerticalAlignment="Bottom" Click="btnClear_Click"
        Grid.Column="2" Grid.Row="1">Clear</Button>
<Label Content="Student ID:" Grid.Column="2" Grid.Row="1"
        Height="26" HorizontalAlignment="Left"
        Margin="184,84,0,0" Name="label3"
        VerticalAlignment="Top" Width="70"
        Grid.ColumnSpan="2" />
<TextBox Grid.Column="3" Grid.Row="1" Height="23"
        HorizontalAlignment="Left" Margin="9,83,0,0"
        Name="textBox3" VerticalAlignment="Top" Width="45" />
<Label Margin="16,115,23,55" x:Name="label1_Copy"
        Grid.Row="1" Content="სტუდენტის სახელი:"
        Grid.ColumnSpan="2"/>
<TextBox x:Name="textBox4" Grid.Column="1"
       HorizontalAlignment="Left" Height="23"
```

```
        Margin="31,120,0,0" Grid.Row="1" TextWrapping="Wrap"
        VerticalAlignment="Top" Width="125"
        Grid.ColumnSpan="2"/>
  </Grid>
</Window>
```

3.11 ნახაზზე მოცემულია ლისტინგის შესაბამისი ინტერფეისის დიზაინის სურათი.



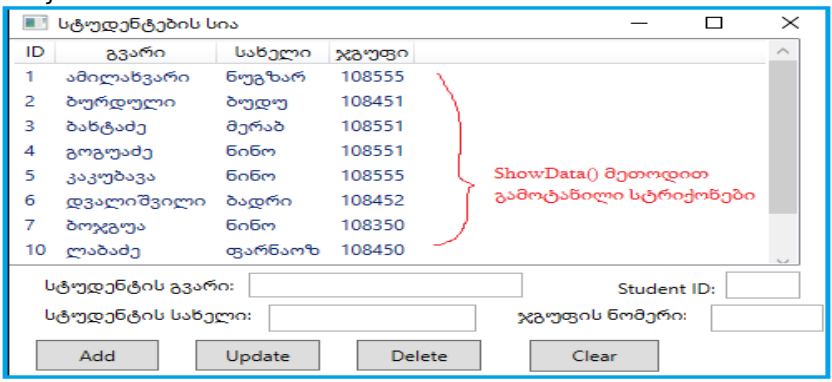
3.11. ინტერფეისის სტრუქტურა

Ms SQL Server ბაზის Student ცხრილის მონაცემთა სტრიქონების გამოსატანად ListView-ში „studID გვარი სახელი ჯგუფი“ ფორმატით დავწეროთ ShowData() მეთოდი, რომელსაც შემდგომ Add, Update და Delete ღილაკებიც გამოიყენებს მონაცემთა ასახვისათვის ეკრანზე (ლისტინგი_3.2 და ნახ.3.12).

```
//-- ლისტინგი_3.3 --- ShowData() მეთოდი --------
public void ShowData()
 {
   SqlConnection con = new SqlConnection(@"Data Source=GTU-
       205A-08;Initial Catalog=University;Integrated
                    Security=True");
   con.Open();
   SqlCommand comm = new SqlCommand("Select St_ID, Gvari,
       Saxeli, JgupiID from Student", con);
   DataTable dt = new DataTable();
   SqlDataAdapter da = new SqlDataAdapter(comm);
   da.Fill(dt);
   listView1.DataContext = dt.DefaultView;
 }
```

თავიდან, პროგრამის ამუშავებისას ეს მეთოდი გამოიძახება ავტომატურად Window_Loaded მეთოდით (ლისტინგი_3.3).

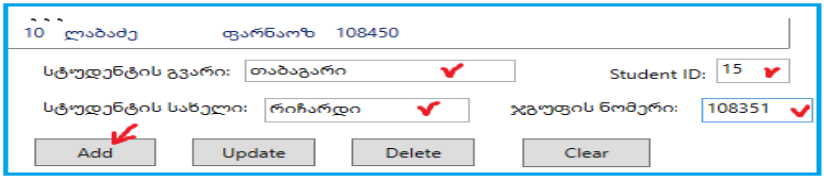
```
// -- ლისტინგი_3.3 ---- Loaded მეთოდი -------
private void Window_Loaded(object sender,RoutedEventArgs e)
   {
       ShowData();
   }
```



ნახ.3.12. მონაცემთა ბაზის ცხრილის სტრიქონები, გამოტანილი Window_Loaded მეთოდით

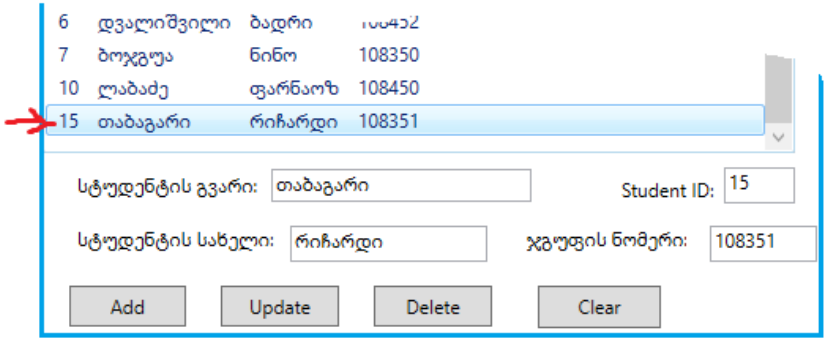
- **Add:** ბაზის ცხრილში ახალი სტუდენტის დასამატებლად უნდა შეივსოს Student ID, გვარი, სახელ და ჯგუფის ნომერი, შესაბამის textBox-ებში და Add ღილაკით ავამოქმედოთშენახვის პროცედურა. იგი არ მუშაობს, რადგან Add-ის შესაბამისი მეთოდის C# კოდე არ არსებობს. შევქმნათ იგი (ლისტინგი_3.4).

```
//--- ლისტინგი_3.4 --- Add() ღილაკი  ----------------
private void btnAdd_Click(object sender, RoutedEventArgs e)
    {
      string Gvari = textBox1.Text;
      string Saxeli = textBox4.Text;
      string JgupiID = textBox2.Text;
      string St_ID = textBox3.Text;
      SqlConnection con = new SqlConnection(@"Data Source=GTU-205A-
                  08;Initial Catalog=University;Integrated Security=True");
      con.Open();
      SqlCommand comm = new SqlCommand("insert into Student(Gvari,
                  Saxeli,JgupiID, St_ID) values(@Gvari,@Saxeli, @JgupiID,
                  @St_ID)", con);
      comm.Parameters.AddWithValue("@Gvari", textBox1.Text);
      comm.Parameters.AddWithValue("@Saxeli", textBox4.Text);
      comm.Parameters.AddWithValue("@JgupiID", textBox2.Text);
      comm.Parameters.AddWithValue("@St_ID", textBox3.Text);
      comm.ExecuteNonQuery();
      con.Close();
      ShowData();
    }
```



ნახ.3.13. ახალი სტუდენტის მონაცემების მომზადება

Add ღილაკის ამოქმედების შედეგი მოცემულია 3.14 ნახაზზე.



ნახ.3.14. StudentID=15 ჩაემატა ბაზაში

Clear ღილაკის ამოქმედებით სუფთავდება ოთხივე textBox-ის შიგთავსი. მისი კოდი მოცემულია 3.5 ლისტინგში.

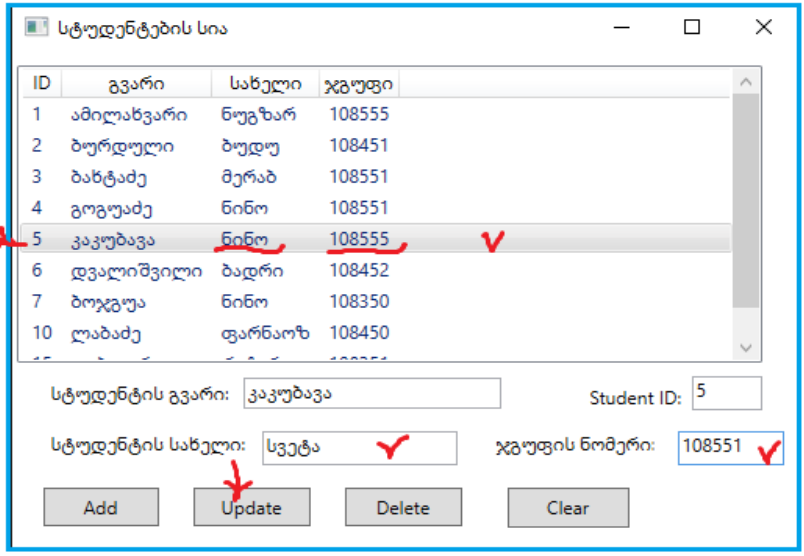
// -- ლისტინგი_3.5 ---- Clear() ღილაკი ----------

```
private void btnClear_Click(object sender, RoutedEventArgs e)
    {
       textBox1.Text = "";
       textBox2.Text = "";
       textBox3.Text = "";
       textBox4.Text = "";
    }
```

- **Update:** ბაზის ცხრილში მონაცემთა მნიშვნელობების შესაცვლელად დავწეროთ შემდეგი მეთოდი (ლისტინგი_3.6).

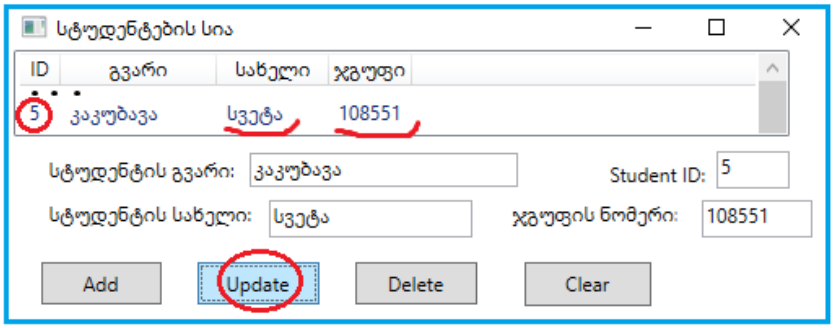
```
// -- ლისტინგი_3.6 ---- Update() ღილაკი ----------------------------
private void btnUpdate_Click(object sender, RoutedEventArgs e)
    {
       if (listView1.SelectedItems.Count > 0)
       {
          DataRowView drv = (DataRowView)listView1.SelectedItem;
          string id = drv.Row[0].ToString();
```

```
        SqlConnection con = new SqlConnection(@"Data Source=GTU-205A-08;
                Initial Catalog=University;Integrated Security=True");
        con.Open();
        SqlCommand comm = new SqlCommand("update Student  set
                Gvari=@Gvari, Saxeli=@Saxeli,JgupiID=@JgupiID where
                 St_ID=@St_ID", con);
        comm.Parameters.AddWithValue("@St_ID", id);
        comm.Parameters.AddWithValue("@Gvari", textBox1.Text);
        comm.Parameters.AddWithValue("@Saxeli", textBox4.Text);
        comm.Parameters.AddWithValue("@JgupiID", textBox2.Text);
        comm.ExecuteNonQuery();
        con.Close();
        ShowData();
    }
}
```



ნახ.3.15. სტუდენტის მონაცემების: სახელის და ჯგუფის შეცვლა

Update ღილაკის ამოქმედებით Student_ID=5 სტრიქონში მოხდება სახელის და ჯგუფის ნომრის შეცვლა ახალი მნიშვნელობებით. შედეგის სტრიქონი მოცემულია 3.16 ნახაზზე.

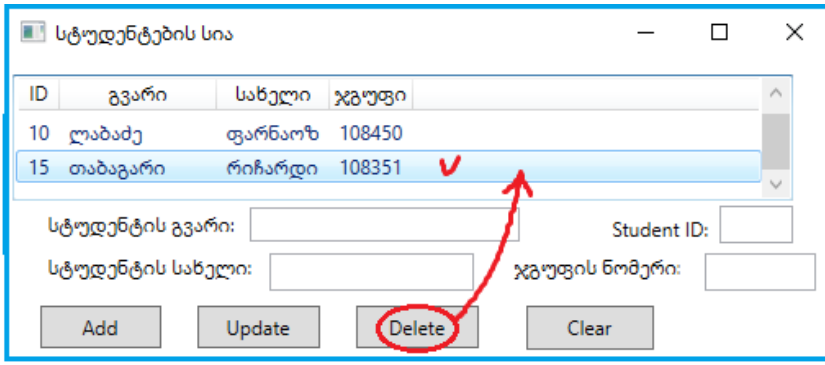


ნახ.3.16. Update() მეთოდით შეცვლილი სტრიქონი

- **Delete:** ბაზის ცხრილში სტრიქონის წასაშლელად დავწეროთ შემდეგი მეთოდი (ლისტინგი_3.7).

```
// -- ლისტინგი_3.7 ---- Delete() ღილაკი ----------------------------
private void btnDelete_Click(object sender, RoutedEventArgs e)
 {
   if (listView1.SelectedItems.Count > 0)
    {
      DataRowView drv = (DataRowView)listView1.SelectedItem;
      string id = drv.Row[0].ToString();
      SqlConnection con = new SqlConnection(@"Data Source=GTU-205A-08;
                  Initial Catalog=University;Integrated Security=True");
      con.Open();
     SqlCommand comm = new SqlCommand("delete from Student where
                  St_ID=@St_ID", con);
     comm.Parameters.AddWithValue("@St_ID", id);
     comm.ExecuteNonQuery();
     ShowData();
    }
 }
```

თუ სტუდენტი თაბაგარი გადავიდა სხვა უნივერსიტეტში, მაშინ მისი მონაცემები უნდა წაიშალოს ცხრილიდან (ნახ.3.17).



ნახ.3.17. Delete() მეთოდით წაიშლება წინასწარ მონიშნული სტრიქონი (St_ID=15)

შედეგად ინტერფეისის LisView-ში გაქრება ეს სტრიქონი, ასევე მონაცემთა SQL Server ბაზაშიც წაიშლება სტუდენტ თაბაგარის მონაცემები (რეალურად ინფორმაცია მის შესახებ რჩება არქივში).

3.8 ლისტინგში მოცემულია C# პროგრამის მთლიანი კოდი.

```
//--- ლისტინგი_3.8 ---------  მთლიანი კოდი ----------------------------
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
```

```
using System.Windows.Shapes;
using System.Data.SqlClient;   // !!!
using System.Data;             // !!!
namespace WpfAppSQL_ListView
{
   public partial class MainWindow : Window
    {
      public MainWindow()
        {
            InitializeComponent();
        }
private void Window_Loaded(object sender,RoutedEventArgs e)
        {
            ShowData();
        }
      public void ShowData()
        {
           SqlConnection con = new SqlConnection(@"Data
Source=GTU-205A-08;Initial Catalog=University;Integrated
Security=True");
           con.Open();
           SqlCommand comm = new SqlCommand("Select St_ID,
Gvari, Saxeli, JgupiID from Student", con);
           DataTable dt = new DataTable();
           SqlDataAdapter da = new SqlDataAdapter(comm);
           da.Fill(dt);
           listView1.DataContext = dt.DefaultView;
        }

private void btnAdd_Click(object sender, RoutedEventArgs e)
      {
          string Gvari = textBox1.Text;
          string Saxeli = textBox4.Text;
          string JgupiID = textBox2.Text;
          string St_ID = textBox3.Text;
```

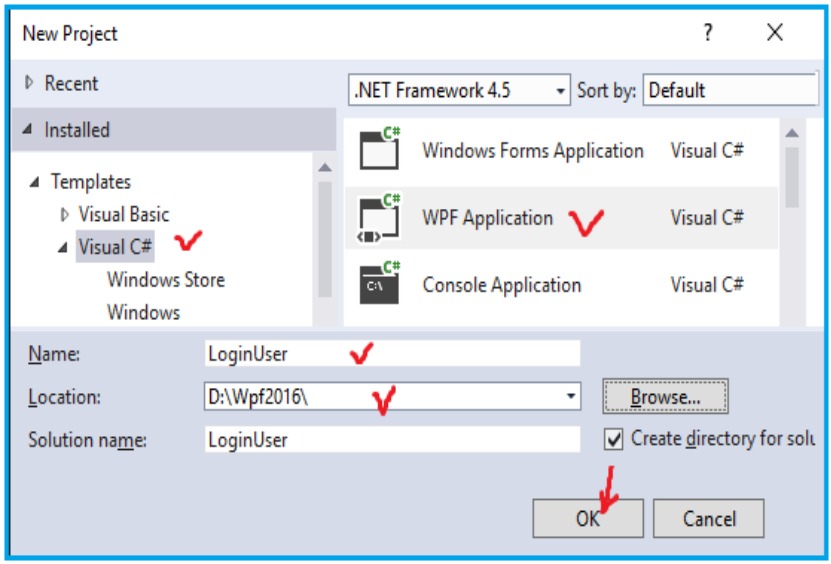
```
            SqlConnection con = new SqlConnection(@"Data
Source=GTU-205A-08;Initial Catalog=University;Integrated
Security=True");
            con.Open();
            SqlCommand comm = new SqlCommand("insert into
Student(Gvari,Saxeli,JgupiID, St_ID) values(@Gvari,@Saxeli,
@JgupiID, @St_ID)", con);
    comm.Parameters.AddWithValue("@Gvari", textBox1.Text);
    comm.Parameters.AddWithValue("@Saxeli", textBox4.Text);
    comm.Parameters.AddWithValue("@JgupiID", textBox2.Text);
    comm.Parameters.AddWithValue("@St_ID", textBox3.Text);
    comm.ExecuteNonQuery();
    con.Close();
    ShowData();
  }

private void btnUpdate_Click(object sender,RoutedEventArgs e)
 {
  if (listView1.SelectedItems.Count > 0)
   {
     DataRowView drv = (DataRowView)listView1.SelectedItem;
     string id = drv.Row[0].ToString();
     SqlConnection con = new SqlConnection(@"Data
        Source=GTU-205A-08;Initial
        Catalog=University;Integrated Security=True");
     con.Open();
     SqlCommand comm = new SqlCommand("update Student  set
         Gvari=@Gvari, Saxeli=@Saxeli,JgupiID=@JgupiID where
         St_ID=@St_ID", con);
    comm.Parameters.AddWithValue("@St_ID", id);
    comm.Parameters.AddWithValue("@Gvari", textBox1.Text);
    comm.Parameters.AddWithValue("@Saxeli", textBox4.Text);
   comm.Parameters.AddWithValue("@JgupiID", textBox2.Text);
    comm.ExecuteNonQuery();
    con.Close();
```

```
    ShowData();
   }
  }
private void btnDelete_Click(object sender,RoutedEventArgs e)
 {
   if (listView1.SelectedItems.Count > 0)
   {
     DataRowView drv = (DataRowView)listView1.SelectedItem;
     string id = drv.Row[0].ToString();
     SqlConnection con = new SqlConnection(@"Data
          Source=GTU-205A-08;Initial
          Catalog=University;Integrated Security=True");
    con.Open();
    SqlCommand comm = new SqlCommand("delete from
           Student where St_ID=@St_ID", con);
    comm.Parameters.AddWithValue("@St_ID", id);
    comm.ExecuteNonQuery();
    ShowData();
   }
 }

private void btnClear_Click(object sender, RoutedEventArgs e)
      {
          textBox1.Text = "";
          textBox2.Text = "";
          textBox3.Text = "";
          textBox4.Text = "";
      }
   }
}
```

3.5. WPF-აპლიკაცია მომხმარებლის სახელის და პაროლის კონტროლისათვის

კორპორაციული ავტომატიზებული სისტემებისათვის განსაკუთრებით მნიშვნელოვანია პროგრამული აპლიკაციის უსაფრთხოების უზრუნველყოფა. ავტორიზაციის მიზნით აქ გავეცნობით მომხმარებლის სახელისა და პაროლის კონტროლის სისტემის აგებას WPF ტექნოლოგიით.

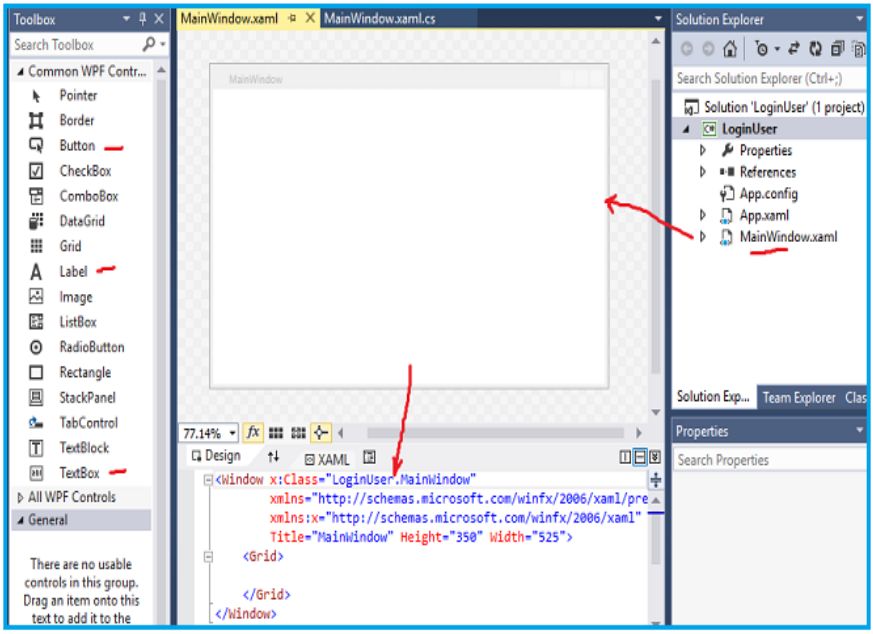
შევქმნათ ახალი Wpf Application პროექტი Login2test სახელით და შევინახოთ ახლადშექმნილ ფოლდერში (ნახ.3.18).



ნახ.3.18. პროექტის შექმნა

მიიღება სამუშაო გარემო (ნახ.3.19).

ინტერფეისის ასაგებად ToolBox-იდან გადმოვიტანოთ ორი Label (წარწერებისათვის მომხმარებლის სახელი და პაროლი), ერთი TextBox - მომხმარებლის სახელის შესატანად, ერთი PasswordBox - პაროლის შესატანად.



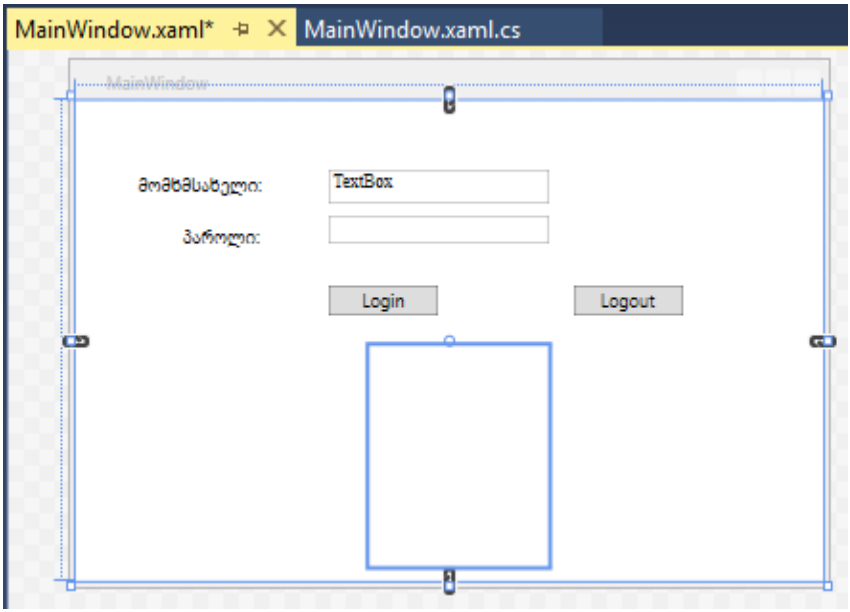
ნახ.3.19. პროექტის შექმნა

ფორმაზე დავდოთ აგრეთვე ორი Button, ერთი Login და მეორე Logout.

შედეგების საილუსტრაციოდ დავამატოთ ფორმაზე ერთი Image, რომელშიც სწორი პასუხის შემთხვევაში გამოვა ამ მომხმარებლის ფოტო. ამასთანავე გაქრება Login ღილაკი და გამოჩნდება Logout.

თუ სახელი ან/და პაროლი შეცდომითაა, მაშინ MessageBox-ში. გამოვა შეტყობინება არასწორი პასუხის შესახებ

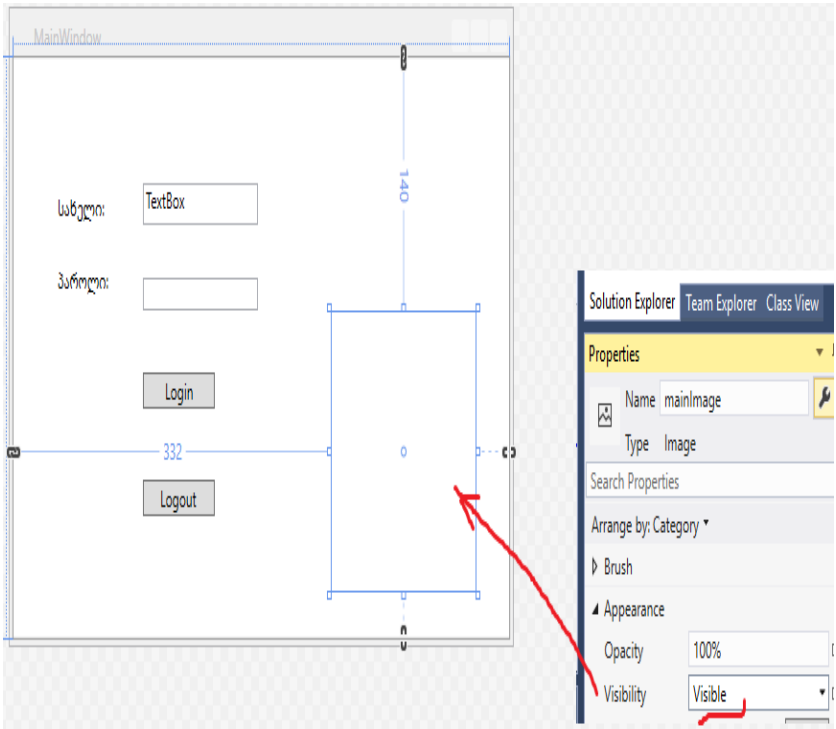
დიზაინის ფორმა ნაჩვენებია 3.20 ნახაზზე.



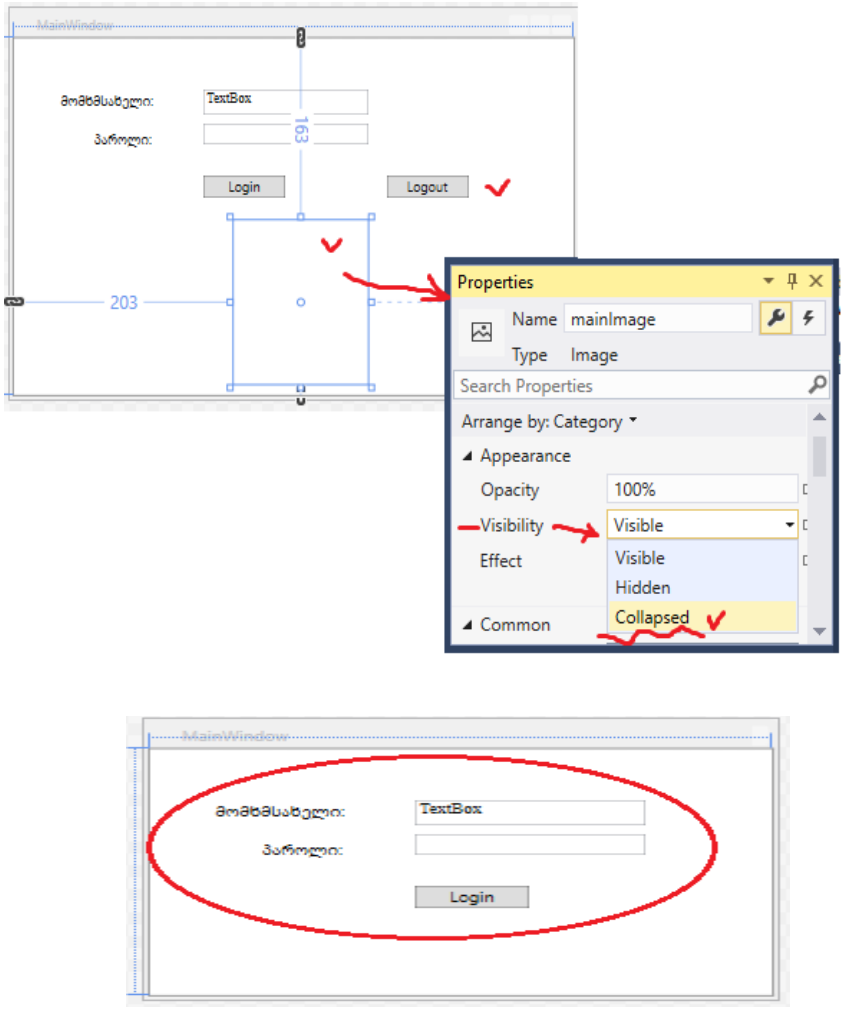
ნახ.3.20. ინტერფეისი

ელემენტების მდებარეობა შეიძლება შეიცვალოს დამპროექტებლის სურვილით.

Properties-ში Image და Logout Button-ისთვის ხილვადობა დავაყენოთ ***Visibility="Collapsed"*** (ნახ.3.21).



ნახ.3.21. თვისება თავიდან Visible-შია



ნახ.3.22. ბოლოს Collapsed-ში დაიმალა სურათი და Logout.

xaml - კოდი ჩავწეროთ 3.9 ლისტინგის მიხედვით:

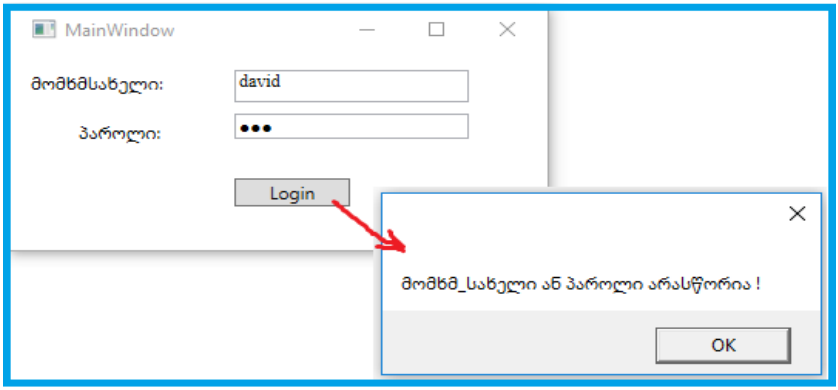
```
<!-- ლისტინგი_3.9 ----------------------------------------
<Window x:Class="test2Log.MainWindow"
    mlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Title="MainWindow" Height="350" Width="525">
        <Grid>
          <Label Content="მომხმ_სახელი:" HorizontalAlignment="Left"
Margin="38,42,0,0" VerticalAlignment="Top" Width="98"/>
          <Label Content="პაროლი:" HorizontalAlignment="Left"
Margin="69,77,0,0"  VerticalAlignment="Top" Width="67"/>
           <TextBox x:Name="UserName" HorizontalAlignment="Left"
Height="23" Margin="175,46,0,0" TextWrapping="Wrap" Text="TextBox"
VerticalAlignment="Top" Width="152" FontFamily="Times New Roman"/>
          <PasswordBox x:Name="PasswordBox" HorizontalAlignment="Left"
Margin="175,77,0,0" VerticalAlignment="Top" Width="152"/>
          <Button x:Name="LoginBTN" Content="Login"
HorizontalAlignment="Left"  Margin="175,123,0,0" VerticalAlignment="Top"
Width="75"/>
          <Button x:Name="LogoutBTN" Content="Logout"
HorizontalAlignment="Left" Margin="345,123,0,0" VerticalAlignment="Top"
Width="75" Visibility="Collapsed"/>
          <Image x:Name="mainImage" HorizontalAlignment="Left"
Height="146" Margin="203,163,0,0" VerticalAlignment="Top" Width="124"
                    Visibility="Collapsed"/>
           </Grid>
    </Window>
```

C# კოდის ტექსტი მოცემულია 3.9 ლისტინგში.

- Double click ღილაკზე Login და ჩავწეროთ კოდი:

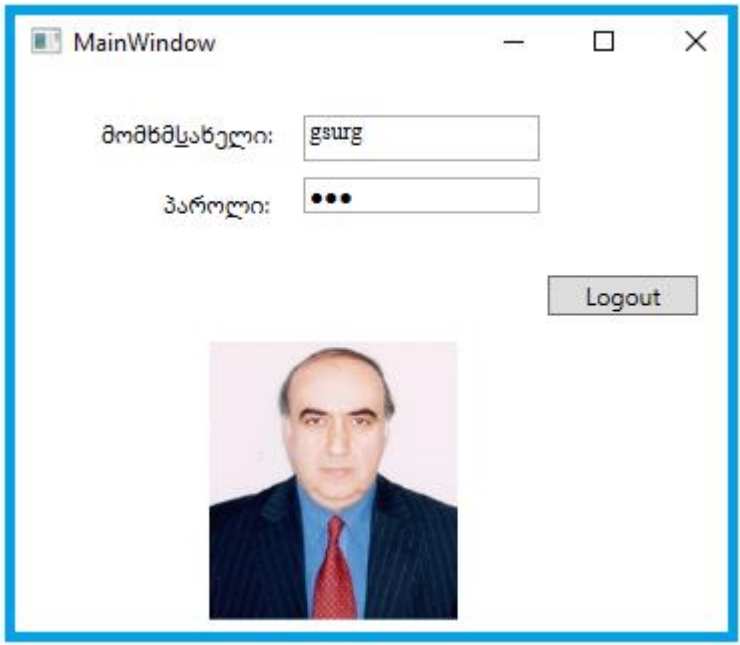
```
//-- ლისტინგი_3.10 ---------------------------------------
private void LoginBTN_Click(object sender, RoutedEventArgs e)
{
  if (!usernameTB.Text.Equals(“”) &&
                !PasswordBoxPB.Password.Equals(“”))
    {
      if (usernameTB.Text.Equals(“1″) &&
                    PasswordBoxPB.Password.Equals(“1″))
        {
          mainImage.Visibility = Visibility.Visible;
          LoginBTN.Visibility = Visibility.Collapsed;
          logoutBTN.Visibility = Visibility.Visible;
        }
       else
          MessageBox.Show(“Wrong Password”);
    }
   else
      MessageBox.Show(“Wrong Info”);
}
```

ავამუშავოთ პროგარმა და შევიტანოთ არასწორი პაროლი:



ნახ.3.23. არასწორი პაროლის შეტანის შემთხვევა

სწორი სახელის და პაროლის შემთხვევაში უნდა მივიღოთ 2.24. ნახაზზე მოცემული სურათი.



ნახ.3.24. სწორი პაროლის შეტანის შემთხვევა

Logout-ზე დაკლიკვით შევიტანოთ C# კოდი:

```
// -- ლისტინგი_3.11 ----------------------------
private void logoutBTN_Click(object sender, RoutedEventArgs e)
{
  mainImage.Visibility = Visibility.Collapsed;LoginBTN.Visibility =
  Visibility.Visible;logoutBTN.Visibility = Visibility.Collapsed;
}
```

მთლიანი C# კოდი მოცემულია 3.12 ლისტინგში.

```
//--- ლისტინგი_3.12 -------------------------------------
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
namespace test2Log
{
  public partial class MainWindow : Window
  {
     public MainWindow()
     {  InitializeComponent();
     }
 private void LoginBTN_Click(object sender, RoutedEventArgs e)
  {
    if (!UserName.Text.Equals("") &&
                !PasswordBox.Password.Equals(""))
        {
          if (UserName.Text.Equals("gsurg") &&
                    PasswordBox.Password.Equals("123"))
          {
             mainImage.Visibility = Visibility.Visible;
```

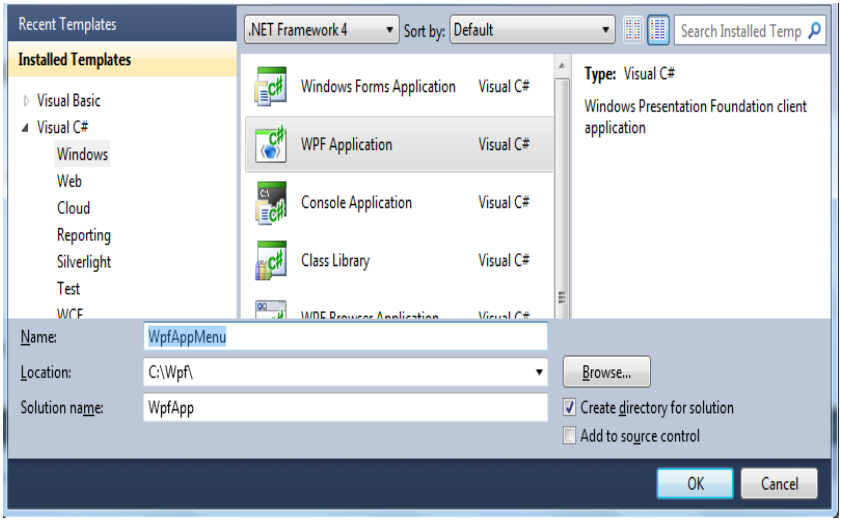
```
                LoginBTN.Visibility = Visibility.Collapsed;
                LogoutBTN.Visibility = Visibility.Visible;
            }
            else
                MessageBox.Show("მომხმ_სახელი ან პაროლი
                                                არასწორია !");
        }
        else
            MessageBox.Show("ინფორმაცია არაა !");
    }
 private void LogoutBTN_Click(object sender, RoutedEventArgs e)
    {   mainImage.Visibility = Visibility.Collapsed;
        LoginBTN.Visibility = Visibility.Visible;
        LogoutBTN.Visibility = Visibility.Collapsed;
    }
  }
}
```

3.6. WPF-ის მართვის ელემენტები: Menu, ToolBar, TabControl და ToolTip

განვიხილოთ WPF-აპლიკაციების ასაგებად Menu, ToolBar, TabControl და ToolTip მართვის ელემენტების გამოყენება. მომხმარებელთა ინტერფეისების ასაგებად WPF-ის ვიზუალური მართვის ელემენტების უფრო დეტალური გაცნობა შესაძლებელია მაიკროსოფთის ბიბლიოთეკის ელექტრონული ცნობარიდან:

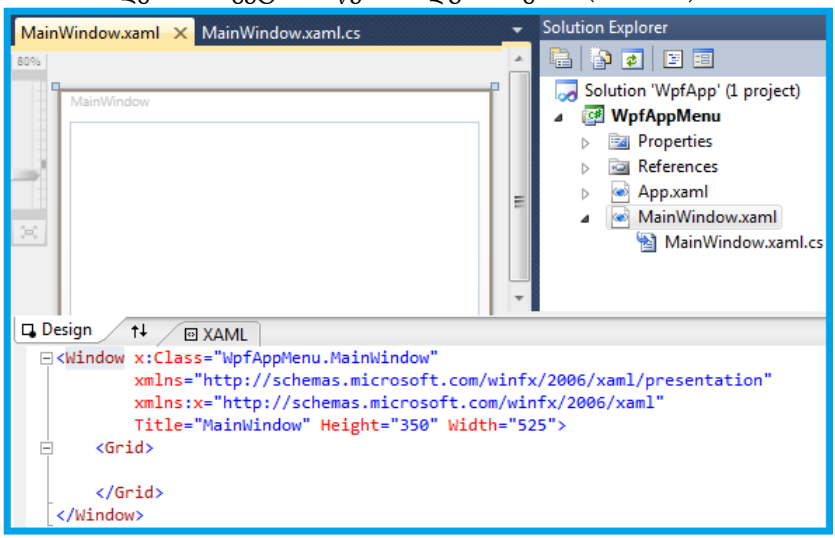
http://msdn.microsoft.com/en-en/library/ms752324.aspx

შევქმნათ ახალი პროექტი Visual Studio-ს Solution Explorer-ში WpfAppMenu სახელით (ნახ.3.25) :



ნახ.3.25

მიიღება პროექტის საწყისი მდგომარეობა (ნახ.3.26):



ნახ.3.26

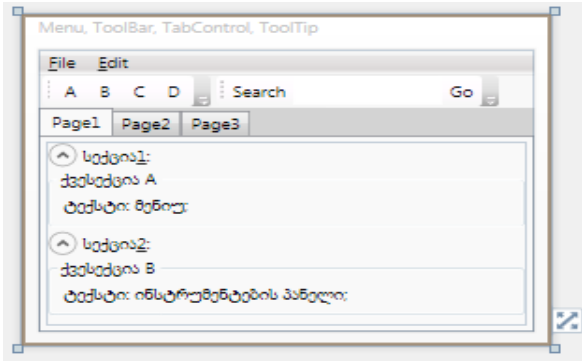
მომხმარებლის ინტერფეისის ფანჯრის დიზაინისათვის MainWindow.xaml კოდში ჩავწეროთ შემდეგი ტექსტი (ლისტინგი_3.13).

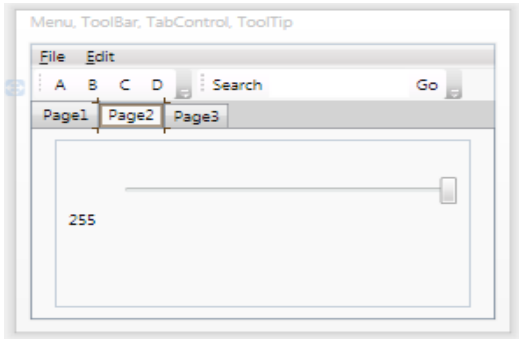
```
<!--- ლისტინგი_3.13 ----------------------------------------------------
<Window x:Class="WpfApp2.Window1"
    xmlns="http://schemas.microsoft.com/winfx/2006/
                                    xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Title="Menu, ToolBar, TabControl, ToolTip"
    SizeToContent="WidthAndHeight"
    ToolTipService.InitialShowDelay="0"
    ToolTipService.ShowDuration="500000" mc:Ignorable="d"
xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
xmlns:mc="http://schemas.openxmlformats.org/markup-
compatibility/2006" d:DesignHeight="270">

    <Window.ToolTip>
        <ToolTip x:Name="toolTip"
            Placement="RelativePoint"
            VerticalOffset="10"
            />
    </Window.ToolTip>

    <DockPanel Background="LightGray">
        <Menu DockPanel.Dock="Top">
            <MenuItem Header="_File">
            <MenuItem Header="E_xit" Click="ExitClicked" />
            </MenuItem>
            <MenuItem Header="_Edit">
                <MenuItem Header="_Cut" />
                <MenuItem Header="C_opy" />
                <MenuItem Header="_Paste" />
            </MenuItem>
        </Menu>
```

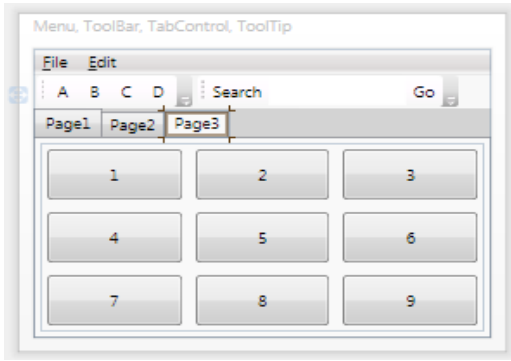
```
<ToolBarTray DockPanel.Dock="Top">
    <ToolBar>
        <Button Width="23">A</Button>
        <Button Width="23">B</Button>
        <Button Width="23">C</Button>
        <Button Width="23">D</Button>
    </ToolBar>
    <ToolBar Header="Search">
        <TextBox Width="100" />
        <Button Width="23">Go</Button>
    </ToolBar>
</ToolBarTray>

<TabControl>
    <TabItem Header="Page1">
        <StackPanel>
           <Expander Header="სექცია_1:"
                           IsExpanded="True">
              <GroupBox Header="ქვესექცია A">
                  <Label>ტექსტი: მენიუ; </Label>
              </GroupBox>
            </Expander>
           <Expander Header="სექცია_2:"
                           IsExpanded="True">
              <GroupBox Header="ქვესექცია B">
    <Label>ტექსტი: ინსტრუმენტების პანელი;</Label>
              </GroupBox>
          </Expander>

        </StackPanel>
    </TabItem>
    <TabItem Header="Page2">
```

```
            <StackPanel Orientation="Horizontal"
                        Margin="5" Width="297">
              <TextBlock Name="value" Margin="10"
                   Text="255" Width="25" Height="23" />
                <Slider Name="slider" Width="241"
                Minimum="0" Maximum="255" Value="255"
        ValueChanged="slider_ValueChanged" Height="77" />
              </StackPanel>
           </TabItem>
           <TabItem Header="Page3">
               <UniformGrid Rows="3" Columns="3">
         <Button ToolTip="ღილაკი 1" Margin="5">1</Button>
         <Button ToolTip="ღილაკი 2"  Margin="5">2</Button>
         <Button ToolTip="ღილაკი 3" Margin="5">3</Button>
         <Button ToolTip="ღილაკი 4" Margin="5">4</Button>
         <Button ToolTip="ღილაკი 5" Margin="5">5</Button>
         <Button ToolTip="ღილაკი 6" Margin="5">6</Button>
         <Button ToolTip="ღილაკი 7" Margin="5">7</Button>
         <Button ToolTip="ღილაკი 8" Margin="5">8</Button>
         <Button ToolTip="ღილაკი 9" Margin="5">9</Button>
          </UniformGrid>
        </TabItem>
      </TabControl>
    </DockPanel>
</Window>
```

შედეგში მიიღება სამი გვერდი Page1, Page2 და Page3 (ნახ.3.27).


Menu, ToolBar, TabControl, ToolTip
File Edit
A B C D
Search
Go
Page1 Page2 Page3
სექცია1:
ქვესექცია A
ტექსტი: მენიუ:
სექცია2:
ქვესექცია B
ტექსტი: ინსტრუმენტების პანელი:


Menu, ToolBar, TabControl, ToolTip
File Edit
A B C D
Search
Go
Page1 Page2 Page3
255

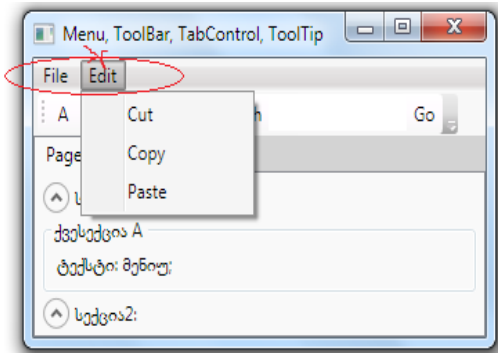

Menu, ToolBar, TabControl, ToolTip
File Edit
A B C D
Search
Go
Page1 Page2 Page3
1 2 3
4 5 6
7 8 9

ნახ.3.27

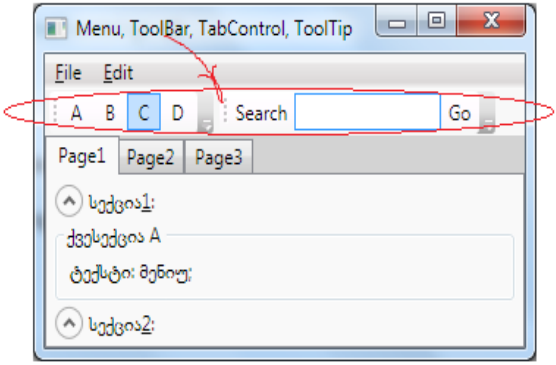
ახლა MainWindow.xaml.cs კოდისათვის შევიტანოთ შემდეგი ტექსტი (ლისტინგი_3.14):

```
//-- ლისტინგი_3.14 ------------------------------------
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;

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

 private void ExitClicked(object sender, RoutedEventArgs e)
        {
            this.Close();
        }

 private void slider_ValueChanged(object sender,
                    RoutedPropertyChangedEventArgs<double> e)
        {
            value.Text = ((int)slider.Value).ToString();
        }
    }
}
```

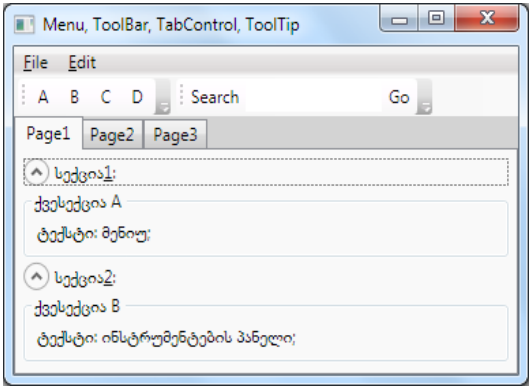
შედეგები ასახულია 3.28 (ა-ე) ნახაზებზე.



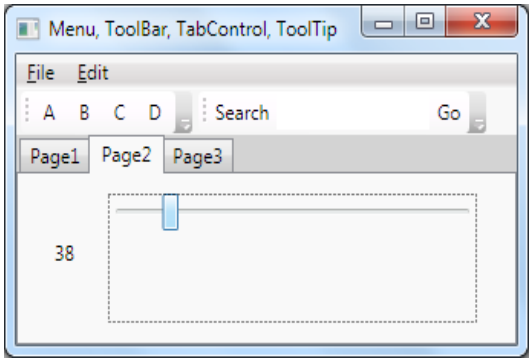
ნახ.3.28–ა



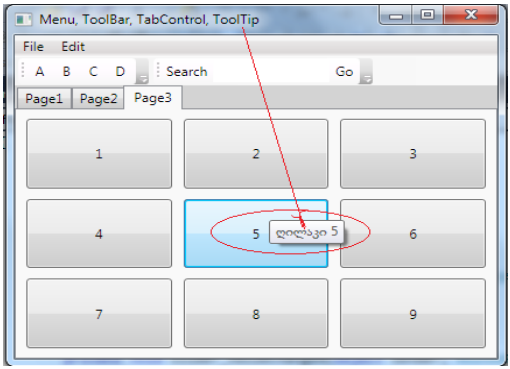
ნახ.3.28–ბ



ნახ.3.28–გ



ნახ.3.28–დ



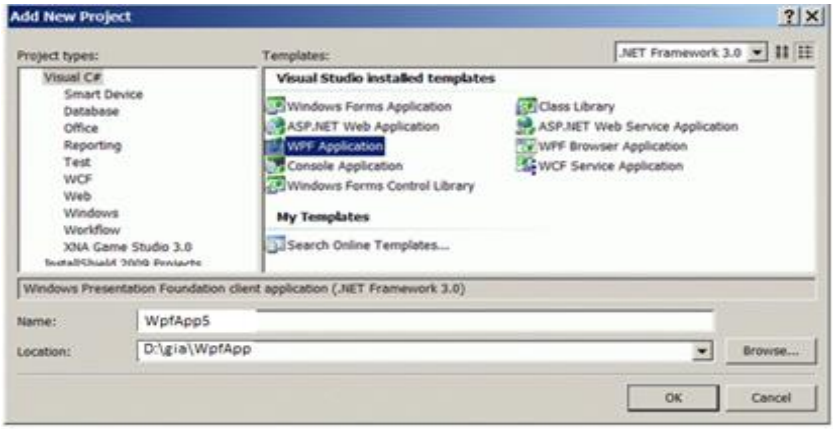
ნახ.3.28–ე

IV თავი
Web აპლიკაციების აგება WPF ტექნოლოგიით

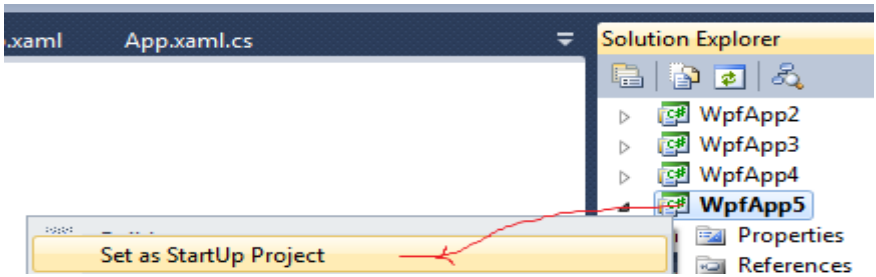
4.1. WPF-ში Web-გვერდების აპლიკაციების შექმნა

ერთ-ერთი მნიშვნელოვანი საკითხია WPF-ის ვებ-აპლიკაციის დამუშავება მომხმარებელთა ინტერფეისების ვიზუალური მართვის ელემენტებით. WPF-პლატფორმას აქვს ვებ-გვერდების შექმნის საშუალებები. ასეთი გვერდების გამოყენება ხშირად მომხმარებლისთვის უფრო მოსახერხებელია, ვიდრე ფანჯრული ორგანიზაცია. გვერდული დანართის (აპლიკაციის) შიგთავსი ჩაშენდება სპეციალურ ნავიგაციურ კარკასში, რომელსაც აქვს ნავიგაციური კავშირები და ნავიგაციური ჟურნალი. მისი ძირითადი (ფესვური) კლასია NavigationWindow, რომელიც ამატებს აპლიკაციისათვის **ნავიგაციის სტანდარტულ ინტერფეისს** და მისთვის საჭირო ინფრასტრუქტურას. NavigationWindow კლასი წარმოებულია Window-კლასიდან და აქვს წვდომა დანართის იგივე საშუალებებზე.

- წინა თავში აგებულ პროექტს Solution Explorer-ში დავამატოთ ახალი პროექტი WpfApp5 სახელით (ნახ.4.1).



ნახ.4.1



ნახ.4.2. ახალი პროექტი დამატებულია

- წავშალოთ ავტომატურად შექმნილი Window1.xaml ფაილი SolutionExplorer-ში და ჩავამატოთ მის ნაცვლად Window(WPF)-შაბლონით ახალი ფაილი NavExample.xaml სახელით;

- გავხსნათ App.xaml ფაილი და შევცვალოთ მასში ატრიბუტი StartupUri="NavExample.xaml";

- ავამუშავოთ პროექტი WpfApp5. დავრწმუნდეთ, რომ არაა შეცდომები;

- გავხსნათ ფაილი NavExample.xaml და შევასწოროთ მასში დესკრიპტორული კოდი:

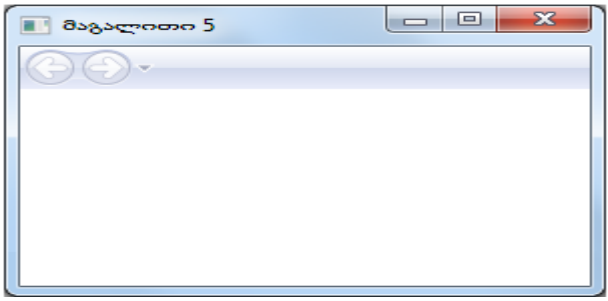
```
<NavigationWindow x:Class="WpfApp5.NavExample"
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Title="მაგალითი 5"
    Height="300"
    Width="300"
    WindowStartupLocation="CenterScreen"
        >
</NavigationWindow>
```

- გავხსნათ NavExample.xaml.cs ფაილი და შევასწოროთ C# კოდის ტექსტი:

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
```

```
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Shapes;

using System.Windows.Navigation;

namespace WpfApp5
{
    public partial class NavExample : NavigationWindow
    {
        public NavExample()
        {
            InitializeComponent();

            //this.Navigate(new Page1());
        }
    }
}
```

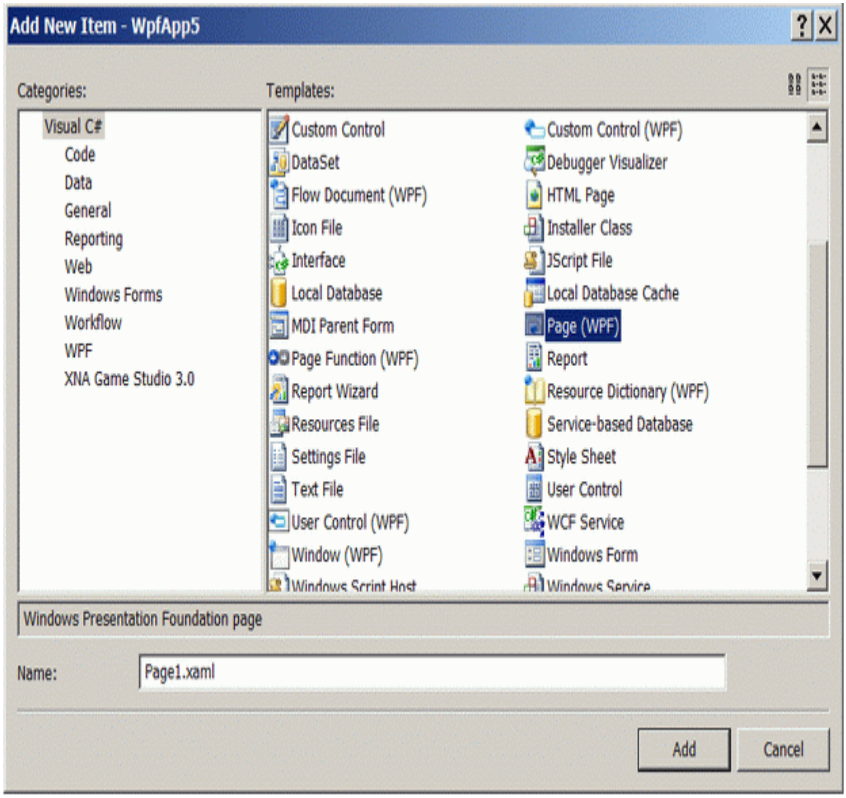
ავამუშავოთ პროექტი, შედეგი იქნება ნახ.4.3, ცარიელი გვერდი ნავიგაციური ელემენტით.



ნახ.4.3

- ნავიგაციური კვანძის შიგთავსი წარმოდგენილი უნდა იყოს კლასით, რომელიც წარმოებულია ბიბლიოთეკის Page-კლასისგან. შევქმნათ სამი გვერდი და მივაბათ ნავიგაციის კვანძს Navigate() მეთოდის დახმარებით.

- დავამატოთ პროექტს ახალი Page ელემენტი Page1.xaml სახელით.



ნახ.4.4

- შევასწოროთ Page1.xaml ფაილის ტექსტი:

```
<Page x:Class="WpfApp5.Page1"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/
        presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        >
    <StackPanel>
```

```
      <TextBlock TextAlignment="Center" FontSize="24">გვერდი 1
        </TextBlock>
      <TextBlock></TextBlock>
        <TextBlock>
      <Hyperlink Click="LinkClicked">მე-2 გვერდზე</Hyperlink>
        </TextBlock>
    </StackPanel>
</Page>
```

- შევასწოროთ Page1.xaml.cs ფაილის კოდი:

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
namespace WpfApp5
{
    public partial class Page1 : Page
    {
        public Page1()
        {
            InitializeComponent();
        }

 private void LinkClicked(object sender, RoutedEventArgs e)
        {
            this.NavigationService.Navigate(page2);
        }
    }
}
```

- დავამატოთ პროექტს ახალი Page ელემენტი Page2.xaml სახელით. შევასწოროთ xaml-კოდი:

```
<Page x:Class="WpfApp5.Page2"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/
          presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Title="Page2"
        >
    <StackPanel>
        <TextBlock TextAlignment="Center" FontSize="24">გვერდი
2</TextBlock>
    </StackPanel>
</Page>
```

- NavExample.xaml ფაილში დავამატოთ Source-ატრიბუტი, რომელიც მიუერთდება კარკასის გაშვებისას Page1.xaml საწყისი გვერდის შიგთავსს.

```
<NavigationWindow x:Class="WpfApp5.NavExample"
    ...
    WindowStartupLocation="CenterScreen"
    Source="Page1.xaml"
         >
</NavigationWindow>
```

- დავალება: აამუშავეთ პროექტი. შეამოწმეთ ღილაკების მუშაობისუნარიანობა.

დასკვნა:

ამგვარად, ჩვენ შევქმენით კარკასი და ორი ცარიელი გვერდი, რომლებიც არაფერს არ აკეთებს. თითოეული გვერდი ავტონომიურია, შეიძლება მათი შევსება ტულბოქსის ელემენტებით.

გვერდებს შორის გადასვლისას საჭიროა ვიცოდეთ ინფორმაციის გადაცემა ერთი გვერდიდან მეორეში. უნდა არსებობდეს საერთო საფოსტო ყუთი, რომელიც არ იქნება დამოკიდებული გვერდებზე.

WPF-ში მონაცემების გადასაცემად გვერდებს შორის იყენებენ ლექსიკონს (წყვილების მასივი: „გასაღები-მნიშვნელობა“) Application.Current.Properties, ან ინფორმაციის „ჩაკერვას“ უშუალოდ ახალი გვერდის ობიექტში.

- Page1-ზე დავამატოთ სახელმინიჭებული ტექსტური ველი შემდეგი სახით:

```
<Page x:Class="WpfApp5.Page1"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/
          presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Title="Page1"
        >
    <StackPanel>
        <TextBlock TextAlignment="Center"
                FontSize="24">გვერდი 1</TextBlock>
        <TextBlock></TextBlock>
        <Label>შეიტანეთ თქვენი სახელი: </Label>
        <TextBox Name="nameBox" Width="200"></TextBox>
        <TextBlock></TextBlock>
        <TextBlock>
     <Hyperlink Click="LinkClicked">მე-2 გვერდზე</Hyperlink>
        </TextBlock>
    </StackPanel>
</Page>
```

TextBox-ობიექტს მივანიჭეთ სახელი, რათა შეიძლებოდეს მასზე მიმართვა კოდიდან.

- შევცვალოთ Page1 გვერდის კოდი შემდეგი სახით;

```
using System;
...
namespace WpfApp5
{
  public partial class Page1 : Page
   {
    public Page1()
        {
```

```
            InitializeComponent();
        }
 private void LinkClicked(object sender, RoutedEventArgs e)
     {
       Page2 page2 = new Page2();
       page2.Message = nameBox.Text + " !!!"; // ინფორმაციის
                                              //„ჩაკერვა“ ობიექტში
       this.NavigationService.Navigate(page2);
     }
   }
}
```

- დავამატოთ Page2 გვერდზე სახელმინიჭებული ტექსტური ჭდე და ჰიპერლინკი Page3-ზე გადასასვლელად. აგრეთვე მომამზადეთ გვერდის მოვლენა Loaded და მოვლენა Click ჰიპერლინკისთვის.

```
<Page x:Class="WpfApp5.Page2"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/
          presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Title="Page2"
    Loaded="Page_Loaded"
        >
     <StackPanel>
         <TextBlock TextAlignment="Center"
                FontSize="24">გვერდი 2</TextBlock>
         <TextBlock></TextBlock>
         <TextBlock>მოგესალმებით </TextBlock>
         <Label Name="nameLabel"></Label>
         <TextBlock Margin="0,10">   <!--ზემოდან დაცილება -->
        <Hyperlink Click="LinkClicked">მე-3 გვერდზე</Hyperlink>
         </TextBlock>
     </StackPanel>
</Page>
```

Label ობიექტს მივანიჭეთ სახელი, რათა კოდიდან შეიძლებოდეს მასზე მიმართვა.

- დავამატოთ Page2-ის კოდს public თვისება, ტექსტურ ჭდეზე გადაცემული ტექსტის მისანიჭებელი კოდი Loaded-მოვლენაში და შემდეგ გვერდზე გადასვლის კოდი.

```
//--- ლისტინგი_4.1 ------------------------------------------
using System;
using System.Collections.Generic;
using System.Text;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
namespace WpfApp5
{
    public partial class Page2 : Page
    {  public Page2()
        {
            InitializeComponent();
        }
        string message;
        public string Message
        {
            set { message = value; }
        }
 private void Page_Loaded(object sender, RoutedEventArgs e)
        {
            nameLabel.Content = message;
        }
 private void LinkClicked(object sender, RoutedEventArgs e)
        {
            Page3 page3 = new Page3();
            this.NavigationService.Navigate(page3);
        }
    }
}
```

- ავამუშავოთ აპლიკაცია. ინფორმაცია გადაეცემა, ოღონდ ღილაკის ამოქმედებით.

(!) ნავიგაციის ღილაკით ახალი ინფორმაცია ტექსტური ველიდან არ გადაიცემა. ინფორმაცია აიღება ისტორიის ჟურნალიდან.

- Page3 გვერდისთვის შეავსეთ xaml-კოდი:

```
<!-- ლისტინგი_4.2 ------------------------------------
<Page x:Class="WpfApp5.Page3"
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Title="Page3">
    <StackPanel>
        <!--გვერდის კონტექსტის სათაური -->
        <TextBlock TextAlignment="Center"
                   FontSize="24">გვერდი 3
            <TextBlock.Margin>0,0,0,10</TextBlock.Margin>
        </TextBlock>
     <!--პირველი ღილაკი-->
        <Button Content="Push Me!" FontSize="22" Width="175"
                           Height="50" Click="Button_Click">
            <Button.Effect>
                <DropShadowEffect />
            </Button.Effect>
        </Button>
     <!--მეორე ღილაკი-->
        <Button FontSize="22" Height="50" Width="175"
                          Margin="0,10" Click="Button_Click">
            "დამკლიკე"
            <Button.Effect>
                <DropShadowEffect />
            </Button.Effect>
            <Button.Foreground>
          <LinearGradientBrush StartPoint="1,0" EndPoint="0,0">
                    <GradientStop Color="Red" Offset="0" />
```

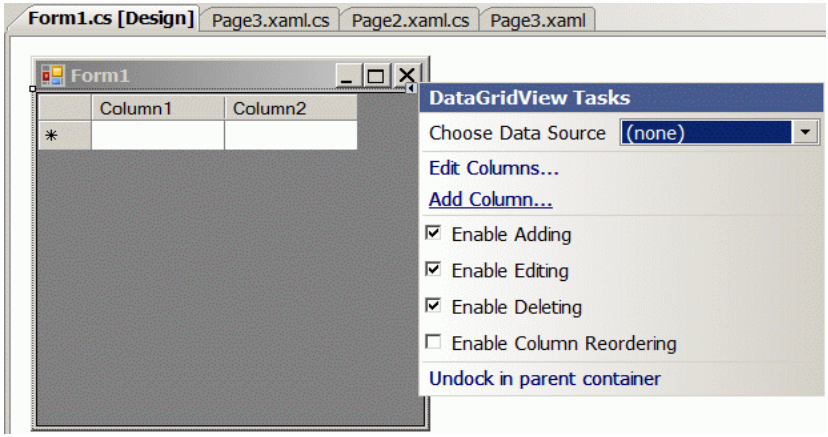
```
                <GradientStop Color="Orange" Offset=".17" />
                <GradientStop Color="Yellow" Offset=".33" />
                <GradientStop Color="Green" Offset=".5" />
                <GradientStop Color="CornflowerBlue"
                                        Offset=".67" />
               <GradientStop Color="Blue" Offset=".84" />
              <GradientStop Color="BlueViolet" Offset="1" />
              </LinearGradientBrush>
          </Button.Foreground>
          <Button.Background>
        <LinearGradientBrush StartPoint="0,0" EndPoint="1,0">
                 <GradientStop Color="Red" Offset="0" />
                <GradientStop Color="Orange" Offset=".17" />
                <GradientStop Color="Yellow" Offset=".33" />
                 <GradientStop Color="Green" Offset=".5" />
                 <GradientStop Color="CornflowerBlue"
                                        Offset=".67" />
                <GradientStop Color="Blue" Offset=".84" />
              <GradientStop Color="BlueViolet" Offset="1" />
               </LinearGradientBrush>
          </Button.Background>
        </Button>
    </StackPanel>
</Page>
```

Click - მოვლენის დამმუშავებელი უნდა ჩაიწეროს ხელით, რათა ის შეიქმნას კოდის ნაწილში.

- სანამ შევავსებთ Page3 გვერდის კოდის ნაწილს, საჭიროა პროექტს დაემატოს ახალი ფორმა. ამით შესაძლებელია WPF და Windows Forms ტექნოლოგიების ერთობლივად მუშაობის დემონსტრირება. ღილაკებზე დავდოთ ფორმის ამუშავების კოდი.

- დავამატოთ WpfApp5-ში ახალი ფორმა Form1.cs;

- Form1 ფორმაზე ტულბოქსიდან დავდოთ DataGridView ელემენტი. დავაყენოთ მისი თვისება Dock=Fill და დავამატოთ ინტელექტუალური დესკრიპტორიდან (SmartTag) ორი სვეტი;



ნახ.4.5

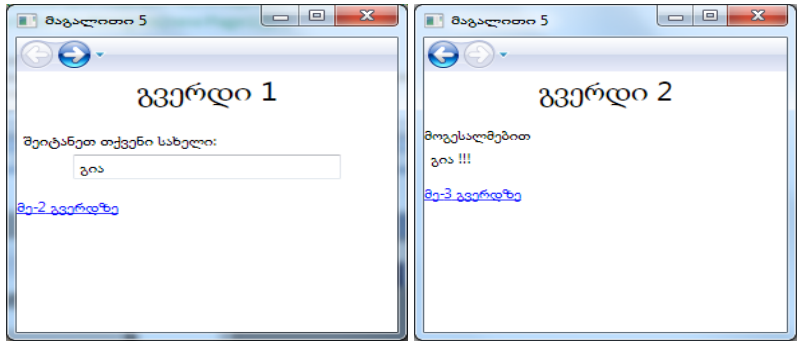
- ღილაკების საერთო დამმუშავებელი Page3 კოდის ნაწილში შევავსოთ ასე:

```
//--- ლისტინგი_4.3 -------------------------------
using System;
using System.Collections.Generic;
using System.Text;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
namespace WpfApp5
{
    public partial class Page3 : Page
    {
        public Page3()
        {
            InitializeComponent();
        }
```

```
 private void Button_Click(object sender, RoutedEventArgs e)
        {
            Form1 frm = new Form1();
            frm.ShowInTaskbar = false; // ფორმის ღილაკი არ
                                       //გამოჩნდეს ამოცანების პანელზე
            frm.Show();
        }
    }
}
```

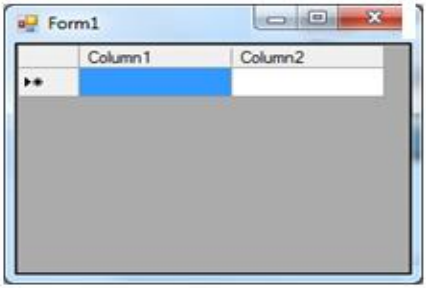
- ავამუშავოთ პროექტი. დავაკვირდეთ ახალი გვერდის დიზაინს. იხსნება Form1 -იც, რაც ადასტურებს WPF და Windows Form ტექნოლოგიების ერთობლივი მუშაობის შესაძლებლობას.

შედეგები (ნახ.4.6–ა,ბ,გ):



ნახ.4.6–ა

ნახ.4.6–ბ

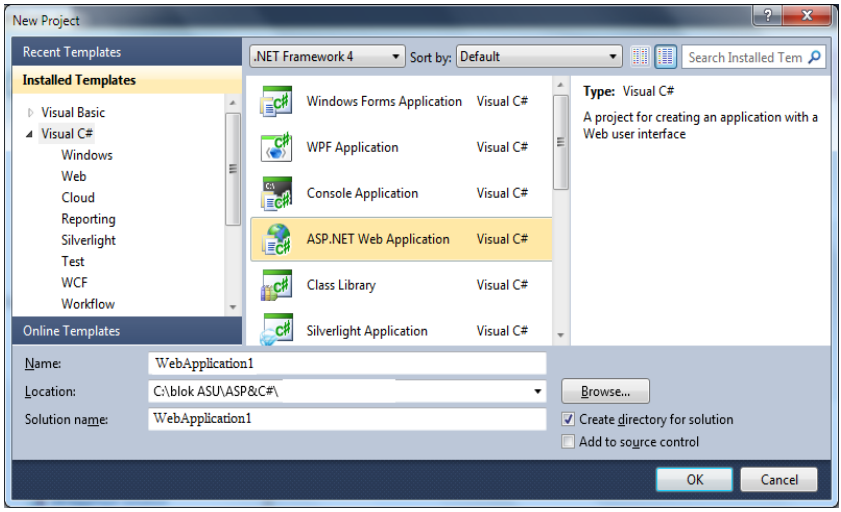


ნახ.4.6–გ

4.2. ASP.NET: ინტერაქტიული Web-გვერდის შექმნა

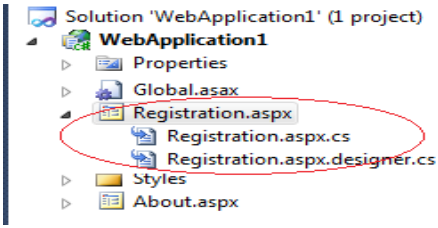
განვიხილოთ ამოცანა: Ms Visual Studio .NET პლატფორმაზე ASP.NET-ის გამოყენებით ავაგოთ ახალი Web-გვერდი, რომელზეც მომხმარებელი შეიტანს საკუთარ მონაცემებს და გადააგზავნის სერვერზე [1,6-8].

- შევქმნათ ახალი ASP.NET აპლიკაცია პროექტის სახელით WebApplication1 (ნახ.4.7).



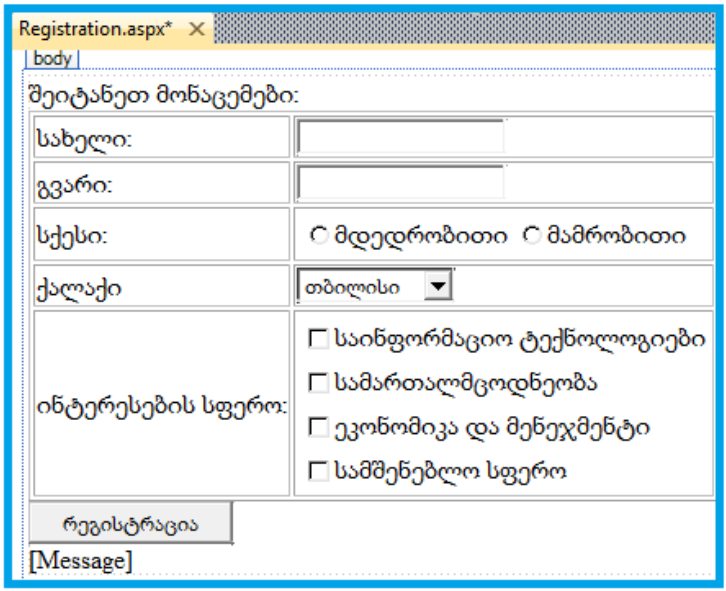
ნახ.4.7

- დავამატოთ ფაილები Registration.aspx და Registration.aspx.cs.



ნახ.4.8

Web-გვერდის სარეგისტრაციო ფორმის მაკეტი ნაჩვენებია 4.9 ნახაზზე.



ნახ.4.9

ფორმაზე მოთავსებულია სერვერული მართვის ელემენტები form, asp:TextBox, asp:DropDownList, asp:CheckBoxList, asp:Button, asp:Label და ა.შ., რომლებიც ასახულია Registration.aspx ფაილის 4.4 ლისტინგში. გახსენით Registration.aspx და შეიტანეთ შემდეგი კოდი:

```
<!— ლისტინგი_4.4 ———— –>
<%@ Page Language="C#" AutoEventWireup="true"
     CodeBehind="Registration.aspx.cs"
     Inherits="WebApplication1.Registration" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html >
 <head>
```

```
 <title>რეგისტრაციის ფორმა</title>
</head>
<body>
 <form method="post" runat="server" id="registration">
  შეიტანეთ მონაცემები:
  <table border="1">
 <tr>
  <td>სახელი:</td>
  <td>
<asp:TextBox id="FirstName" runat="server"></asp:TextBox>
  </td>
 </tr>
 <tr>
  <td>გვარი:</td>
  <td>
<asp:TextBox id="LastName" runat="server"></asp:TextBox>
  </td>
 </tr>
 <tr>
  <td>სქესი:</td>
   <td><asp:RadioButtonList id="Sex" runat="server"
            RepeatDirection="Horizontal">
       <asp:ListItem Value="მდედრობითი"></asp:ListItem>
       <asp:ListItem Value="მამრობითი"></asp:ListItem>
     </asp:RadioButtonList></td>
   </tr>
   <tr>
    <td>ქალაქი</td>
    <td><asp:DropDownList id="City" runat="server">
      <asp:ListItem Value="თბილისი"></asp:ListItem>
      <asp:ListItem Value="ქუთაისი"></asp:ListItem>
      <asp:ListItem Value="რუსთავი"></asp:ListItem>
      <asp:ListItem Value="გორი"></asp:ListItem>
      <asp:ListItem Value="ბათუმი"></asp:ListItem>
      <asp:ListItem Value="თელავი"></asp:ListItem>
     </asp:DropDownList></td>
   </tr>
   <tr>
     <td>ინტერესების სფერო:</td>
   <td>
```

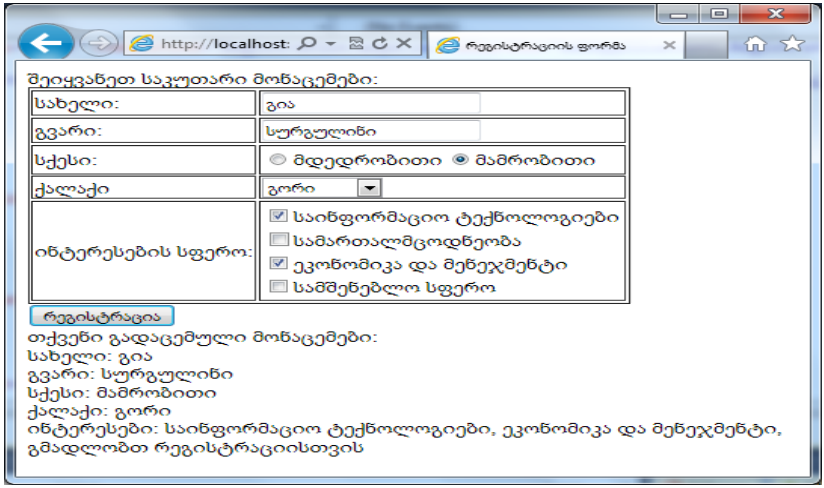
```
      <asp:CheckBoxList id="Interests" runat="server">
  <asp:ListItem Value="საინფორმაციო ტექნოლოგიები">
  </asp:ListItem>
<asp:ListItem Value="სამართალმცოდნეობა"></asp:ListItem>
<asp:ListItem Value="ეკონომიკა და მენეჯმენტი"></asp:ListItem>
<asp:ListItem Value="სამშენებლო სფერო"></asp:ListItem>
      </asp:CheckBoxList></td>
     </tr>
    </table>
   <asp:Button id="Register" runat="server" Text="რეგისტრაცია"
OnClick="Register_Click"></asp:Button>
    <br />
    <asp:Label id="Message" runat="server"></asp:Label>
  </form>
 </body>
</html>
```

Web-გვერდის ჩატვირთვის და მონაცემების შევსების შემდეგ „რეგისტრაცია“ Button-ის დაჭერისას გამოიძახება OnClick მოვლენაზე მიბმული მეთოდი Register_Click. ის აღიწერება C# კოდში, რომლის 4.5 ლისტინგი მოცემულია ქვემოთ. გავხსნათ Registration.aspx.cs ფაილი და შევიტანოთ შემდეგი კოდი:

```
// ── ლისტინგი_4.5 ──────────
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
namespace WebApplication1
{
  public partial class Registration : System.Web.UI.Page
    {
     protected void Page_Load(object sender, EventArgs e){ }
     protected void Register_Click(object sender, EventArgs e)
      {
       System.Text.StringBuilder sb = new
                                  System.Text.StringBuilder();
       sb.Append("თქვენი გადაცემული მონაცემები:<br>");
```

```
        sb.AppendFormat("სახელი: {0}<br>", FirstName.Text);
        sb.AppendFormat("გვარი: {0}<br>", LastName.Text);
        sb.AppendFormat("სქესი: {0}<br>", Sex.SelectedValue);
        sb.AppendFormat("ქალაქი: {0}<br>", City.SelectedValue);
        sb.Append("ინტერესები: ");

           foreach (ListItem item in Interests.Items)
              {
                if (item.Selected)
                   sb.AppendFormat("{0}, ", item.Value);
              }
              sb.Append("<br>გმადლობთ რეგისტრაციისთვის");
              Message.Text = sb.ToString();
           }
    }
}
```

Web-გვერდი და მისი შესრულების შედეგი მოცემულია 4.10 ნახაზზე.

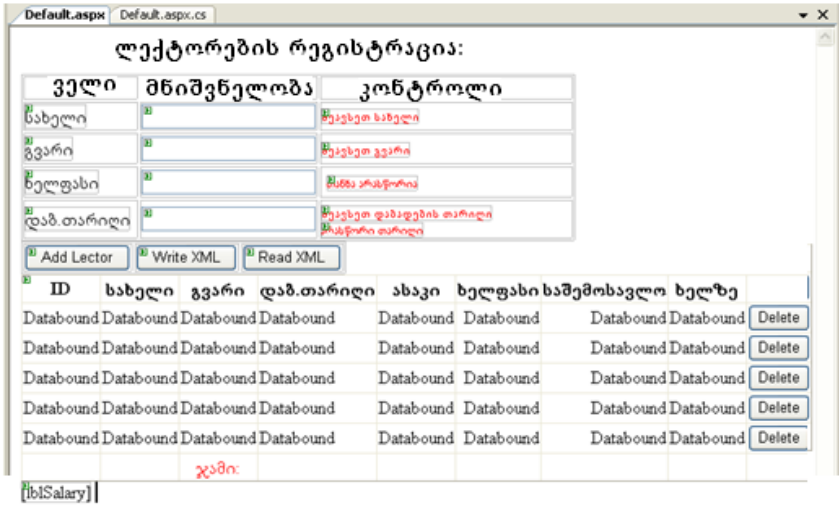


ნახ.4.10

4.3. DataSet / GridView ობიექტებთან მუშაობა და XML

ინტერაქტიულ რეჟიმში მონაცემების შეტანისას ერთ-ერთი მნიშვნელოვანი საკითხია მათი კონტროლის პროცედურების დამუშავება, შეტანილ მონაცემთა ეკრანზე ასახვის საშუალებების GridView / DataSet გამოყენება. აგრეთვე მეტად მოსახერხებელია შედეგების XML ფაილში შენახვა და XML ფაილიდან მათი ამოღების პროცედურების შექმნა.

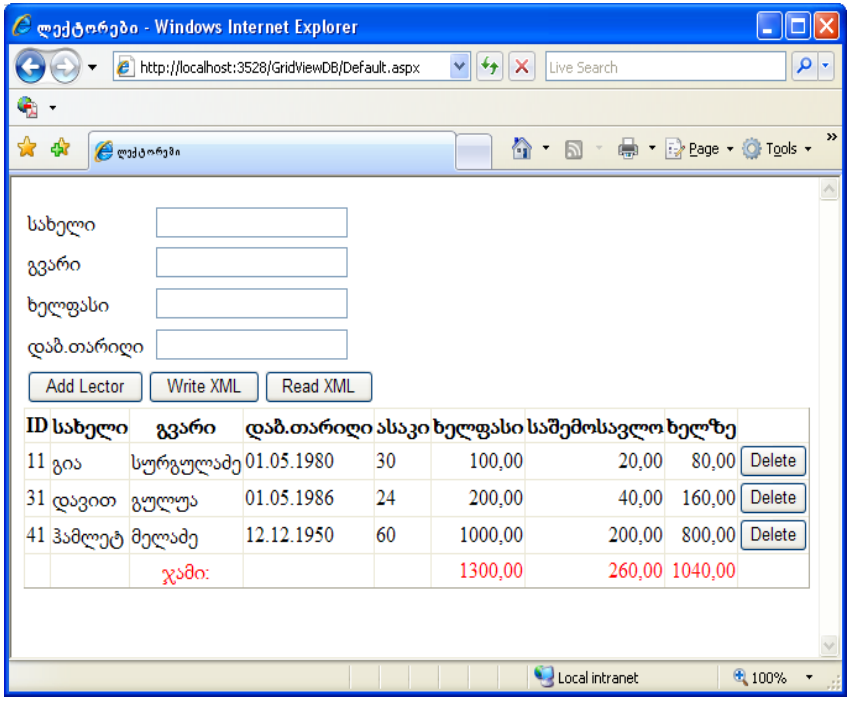
განვიხილოთ ასეთი ამოცანა: Visual Studio .NET გარემოში ავაგოთ Web-პროექტი ლექტორთა სარეგისტრაციო მონაცემების შესატანად. განვახორციელოთ მონაცემთა ვიზუალური და ავტომატური კონტროლის საშუალებების გამოყენება. Add Lector-ღილაკით შეტანილი მონაცემები აისახოს GridView ცხრილში უნიკალური ID-ს მქონე სტრიქონის სახით, მომზადდეს სარეგისტრაციო ცხრილი ახალი ინფორმაციის შესატანად (ნახ.4.11). ავტომატური კონტროლისთვის გამოიყენება ToolBox->Validation.



ნახ.4.11

- GridView ცხრილში ავაგოთ სვეტები შესაბამისი დასახელებებით. გარდა სარეგისტრაციო მონაცემებისა, დავამატოთ გაანგარიშებადი ველებიც: ***ასაკი, საშემოსავლო _გადასახადი, ხელზე_ასაღები_თანხა***. ეს ველები განისაზღვრება C#-კოდში;
- GridView ცხრილში დავამატოთ ახალი სვეტი Delete-ფუნქციით, არასასურველი სტრიქონის ოპერატიულად წასაშლელად;
- GridView ცხრილის Fother სტრიქონში გამოვიტანოთ „ჯამი:“ ხელფასის, საშემოსავლოს და ***ხელზე_ასაღები_თანხის*** სვეტებისათვის ჯამური მნიშვნელობების გამოსატანად.

4.12 ნახაზზე ნაჩვენებია Web-გვერდის მაკეტი ბრაუზერში.



ნახ.4.12

ქვემოთ აღიწერება ღილაკები, რომლებიც ფორმაზეა განლაგებული. მათი დანიშნულებაა მონაცემთა დამატება, კონტროლი, შენახვა XML ფაილში, შემდეგ კი ამ ფაილიდან ამოღება. განვიხილოთ მათი პროგრამული კოდების ლისტინგები:

```
// —— ლისტინგი_4.6—— Add_Lector ———
 protected void Button1_Click(object sender, EventArgs e)
   {
      DataSet dsLectors = Session["MyDataSet"] as DataSet;
      DataTable dtLectors = dsLectors.Tables["Lectors"];

      DataRow newlector = dtLectors.NewRow();
      //newlector["ID"] = "1";
      newlector["FirstName"] = txtFirstName.Text;
      newlector["LastName"] = txtLastName.Text;
      if (!String.IsNullOrEmpty(txtSalary.Text))
           newlector["Salary"] = Decimal.Parse(txtSalary.Text);

      newlector["BirthDate"] = DateTime.Parse(txtBirthDate.Text);
      dtLectors.Rows.Add(newlector);
      object sumSalary = dtLectors.Compute("SUM(Salary)", "");
      object sumTax = dtLectors.Compute("SUM(Tax)", "");
      object sumNettoSalary = dtLectors.Compute("SUM(NettoSalary)", "");
      //lblSalary.Text = sumSalary.ToString();
     GridView1.Columns[5].FooterText = String.Format("{0:F2}", sumSalary);
     GridView1.Columns[6].FooterText = String.Format("{0:F2}",sumTax);
 GridView1.Columns[7].FooterText = String.Format("{0:F2}", sumNettoSalary);

      Session["MyDataSet"] = dsLectors;
      GridView1.DataSource = dsLectors;
      GridView1.DataBind();
   }
```

- დავამატოთ ორი ღილაკი: Write_XML (სტრიქონების შესანახად XML ფაილში) და Read_XML (სტრიქონების ამოსაღებად XML ფაილიდან).

```
// ── ლისტინგი_4.7 ──  Write_XML ----------------
protected void Button2_Click(object sender, EventArgs e)
   {
      DataSet ds = Session["MyDataSet"] as DataSet;
      ds.WriteXml(Request.PhysicalApplicationPath + "\\lectors.xml");
      //Response.Redirect("~/lectors.xml");
   }

// ── ლისტინგი_4.8 ──  Read_XML ---------------
protected void Button3_Click(object sender, EventArgs e)
   {
      DataSet ds = Session["MyDataSet"] as DataSet;
      ds.ReadXml(Request.PhysicalApplicationPath + "\\lectors.xml");
      DataTable dtLectors = ds.Tables["Lectors"];
      object sumSalary = dtLectors.Compute("SUM(Salary)", "");
      object sumTax = dtLectors.Compute("SUM(Tax)", "");
      object sumNettoSalary = dtLectors.Compute("SUM(NettoSalary)", "");
      //lblSalary.Text = sumSalary.ToString();
      GridView1.Columns[5].FooterText = String.Format("{0:F2}", sumSalary);
      GridView1.Columns[6].FooterText = String.Format("{0:F2}", sumTax);
 GridView1.Columns[7].FooterText = String.Format("{0:F2}", sumNettoSalary);
      GridView1.DataSource = ds;
      GridView1.DataBind();
   }
```

- GridView ცხრილთან სამუშაოდ გამოიყენება DataSet ობიექტი, რომელსაც C#-კოდში აქვს შემდეგი სახე:

```
// ── ლისტინგი_4.9 ── DataSet ──
private DataSet GetDataSet()
   {
      DataTable lectors = new DataTable("Lectors");
```

```
//Add the DataColumn using all properties
DataColumn id = new DataColumn("ID");
id.DataType = typeof(int);
id.Unique = true;
id.AutoIncrement = true;
id.AutoIncrementSeed = 1;
id.AutoIncrementStep = 10;
id.AllowDBNull = false;
id.Caption = "ID";
lectors.Columns.Add(id);

//Add the DataColumn using defaults
DataColumn firstName = new DataColumn("FirstName");
firstName.DataType = typeof(string);
firstName.MaxLength = 35;
firstName.AllowDBNull = false;
lectors.Columns.Add(firstName);

DataColumn lastName = new DataColumn("LastName");
lastName.DataType = typeof(string);
lastName.MaxLength = 50;
lastName.AllowDBNull = false;
lectors.Columns.Add(lastName);

DataColumn salary = new DataColumn("Salary", typeof(decimal));
salary.DefaultValue = 0.00m;
lectors.Columns.Add(salary);

DataColumn birthDate = new DataColumn("BirthDate",
                                                typeof(DateTime));
//birthDate.DefaultValue = DateTime.Now;
birthDate.AllowDBNull = true;
lectors.Columns.Add(birthDate);

DataColumn age = new DataColumn("Age", typeof(DateTime));
age.ColumnMapping = MappingType.Hidden;
```

```
    age.Expression = "BirthDate";
    lectors.Columns.Add(age);

    DataColumn tax = new DataColumn("Tax", typeof(decimal));
    tax.ColumnMapping = MappingType.Hidden;
    tax.DataType = typeof(decimal);
    tax.Expression = "salary*0.2";
    lectors.Columns.Add(tax);

    DataColumn netto = new DataColumn("NettoSalary", typeof(decimal));
    netto.ColumnMapping = MappingType.Hidden;
    netto.DataType = typeof(decimal);
    netto.Expression = "salary - salary*0.2";
    lectors.Columns.Add(netto);

    ////Derived column using expression
    //DataColumn lastNameFirstName = new DataColumn("LastName and
    // FirstName");
    //lastNameFirstName.DataType = typeof(string);
    //lastNameFirstName.MaxLength = 70;
    //lastNameFirstName.Expression = "lastName + ', ' + firstName";
    //employee.Columns.Add(lastNameFirstName);
    DataSet ds = new DataSet();
    ds.Tables.Add(lectors);
    return ds;
  }
```

- XML ფაილს, მასში სტრიქონების (ობიექტების) ჩაწერის შემდეგ ექნება ასეთი სახე:

```
<!—— ლისტინგი_4.10 — XML ჩანაწერების შენახვის სტრუქტურა –>
<?xml version="1.0" standalone="yes"?>
<NewDataSet>
  <Lectors>
    <ID>11</ID>
    <FirstName>გია</FirstName>
```

```
    <LastName>სურგულაძე</LastName>
    <Salary>100</Salary>
    <BirthDate>1980-05-01T00:00:00+04:00</BirthDate>
  </Lectors>
  <Lectors>
    <ID>31</ID>
    <FirstName>დავით</FirstName>
    <LastName>გულუა</LastName>
    <Salary>200</Salary>
    <BirthDate>1986-05-01T00:00:00+04:00</BirthDate>
  </Lectors>
  <Lectors>
    <ID>41</ID>
    <FirstName>გიორგი</FirstName>
    <LastName>სურგულაძე</LastName>
    <Salary>1900</Salary>
    <BirthDate>1980-12-30T00:00:00+04:00</BirthDate>
  </Lectors>
</NewDataSet>
```

V თავი
ინსტრუქციები სისტემის საპილოტო ვერსიის და საკურსო პროექტის გასაფორმებლად

5.1. აპლიკაციის საპილოტო ვერსიის ტესტირება და სადემონსტრაციოდ მომზადება

პროგრამული აპლიკაციის საპილოტო ვერსია არის ასაგები რეალური სისტემის სადემონსტრაციო ვარიანტი, რომელშიც კარგად ჩანს საპრობლემო სფეროს ობიექტის ავტომატიზაციის ამოცანების სანიმუშო მაგალითები. აქ ალგორითმულად და პროგრამულად რეალიზებულია დეველოპერების მიერ კომპიუ-ტერული სისტემის მოდელი, თავისი მონაცემებით და მეთოდებით (ფუნქციონალობით).

ტესტირება მოიცავს ჩვენ მიერ აწყობილი სისტემის ცალკეული ამოცანების შესრულებაზე გაშვებას. საჭიროა მონაცემთა ბაზაში, ცხრილებში ინფორმაციის შეტანა-გამოტანისა და მონაცემთა კორექტირების პროცედურების შემოწმება, აგრეთვე მენიუს, ღილაკების და სხვა ვიზუალური კომპონენტების ფუნქციონირების სისწორის კონტროლი.

ბოლოს, უნდა დაიწეროს მოთხოვნები მონაცემთა ბაზიდან მონაცემთა ამოსაღებად და დასამუშავებლად. შემოწმდება, თუ რამდენად სწორად ასრულებს აგებული Windows ან Web აპლიკაციის პროგრამული კოდი წინასწარ გათვალისწინებულ დავალებებს.

6.2. საპრეზენტაციო ფაილის და საკურსო პროექტის დოკუმენტაციის მომზადება

საბოლოო შედეგების პრეზენტაციის მიზნით კომპიუტერისა და პროექტორის გამოსაყენებლად შეირჩევა აპლიკაციის პროექტის საილუსტრაციო მასალა: მიზანი, ამოცანები, გადაწყვეტა, შედეგები და რეკომენდაციები. საპრეზენტაციო სლაიდები მომზადდება Ms_PowerPoint ინსტრუმენტით.

პროექტის დოკუმენტაცია მზადდება MsWord ფაილის სახით, ნაბეჭდი A4 ფორმატით Sylfaen ფონტით, 11p და 1.15 ინტერვალით, არეები 2 სმ, ძირითადი ტექსტი 25–40 გვერდი.

6. გამოყენებული ლიტერატურა

1. ჩოგოვაძე გ., ფრანგიშვილი ა., სურგულაძე გ. მართვის საინფორმაციო სისტემების დაპროგრამების ჰიბრიდული ტექნოლოგიები და მონაცემთა მენეჯმენტი. მონოგრ., სტუ, „ტექნიკური უნივერსიტეტი“, თბ., 2017, -1001 გვ.

2. გოგიჩაიშვილი გ., ბოლხი გ., სურგულაძე გ., პეტრიაშვილი ლ. მართვის ავტომატიზებული სისტემების ობიექტ-ორიენტი-რებული დაპროექტების და მოდელირების ინსტრუმენტები (MsVisio, WinPepsy, PetNet, CPN). სტუ. „ტექნიკური უნივერსი-ტეტი”. თბ. 2013. -232 გვ.

3. Booch G., Jacobson I., Rumbaugh J. Unified Modeling Language for Object-Oriented Development. Rational Software Corporation, Santa Clara, 1996

4. სურგულაძე გ. დაპროგრამების მეთოდები და ინსტრუმენტები (UML, MsVisio, C++). სტუ. თბ., 2007. http://www.gtu.edu.ge/katedrebi/kat94/pdf/BC++B-22.pdf

5. გოგიჩაიშვილი გ., სუხიაშვილი თ. სისტემების ობიექტ–ორიენტირებული ანალიზი და დაპროექტება. სტუ. თბ., 2012. http://gtu.ge/books/suxi_gogichaishvili.pdf

6. სურგულაძე გ., ბულია ი. კორპორაციულ Web–აპლიკაცი–ათა ინტეგრაცია და დაპროექტება. მონოგრ., სტუ. თბ., 2012. ელ–ვერსია: http://www.gtu.ge/books/GiaSurg_Book_2012.pdf

7. სურგულაძე გ., ბულია ი., თურქია ე. Web-აპლიკაციების დამუშავება მონაცემთა ბაზების საფუძველზე (ADO.NET, ASP.NET, C#). სტუ, „ტექნიკური უნივერსიტეტი“. თბილისი, 2014. -189 გვ. http://gtu.ge/book/gia_sueguladze/SurgEka_ASP_NET.pdf

8. სურგულაძე გ. ვიზუალური დაპროგრამება C#_2010 ენის ბაზაზე. სტუ. თბ., სახელმძღ., 2011. -445 გვ., ბიბლ.ინდ. 681.03.06(02) 69 http://gtu.ge/View/index.html#http://gtu.ge/book/GiaSurg_C_2010.pdf

7. დანართები

7.1. დანართი N1

/საკურსო პროექტის სატიტულო გვერდი/

საქართველოს ტექნიკური უნივერსიტეტი

მართვის ავტომატიზებული სისტემების
(პროგრამული ინჟინერიის)
დეპარტამენტი (№805)

საკურსო პროექტი

დისციპლინაში: **„პროგრამული აპლიკაციების დეველოპმენტის საფუძვლები“**

ჯგ.№: 108 . . .
სტუდენტი:
თემა: ‘‘საკურსო პროექტის სათაური‘‘

ხელმძღვანელი:

ჩაბარების შედეგი:
თარიღი:

თბილისი 2017

7.2. დანართი N2

/საკურსო პროექტის საწყისი მონაცემების ფორმა/

პროექტის საწყისი მონაცემები

(სტუდენტი მიიღებს პროფესორისგან)

1. საპრობლემო სფერო (კვლევის ობიექტი)

2. მართვის სფერო (ფუნქციური ამოცანა)

3. კლასები (დასახელება/მონაცემები/მეთოდები)

4. ობიექტების რაოდენობა (კლასებში)

5. ინტერფეისის დიზაინი (ფორმის სტრუქტურა)

6. საანგარიშო მონაცემები

პროფ. ხელმოწერა: . . . . . .

თარიღი: . . . . .

7.3. დანართი N3
საკურსო პროექტის ინდივიდუალური და ჯგუფური თემები

1. საპრობლემო სფერო: **ბიბლიოთეკა**
    1.1. ობიექტი: მკითხველთა რეგისტრაცია
    1.2. ობიექტი: წიგნების ანბანური კატალოგები
    1.3. ობიექტი: წიგნების თემატური კატალოგები
    1.4. ობიექტი: წიგნების გაცემა/დაბრუნება
    1.5. ობიექტი: კადრები/ხელფასები

2. საპრობლემო სფერო: **ფაკულტეტი**
    2.1. ობიექტი: სტუდენტები, ჯგუფები, კურსები
    2.2. ობიექტი: ფაკულტეტის კათედრები, სპეციალობები
    2.3. ობიექტი: ჯგუფები, საგნები, კრედიტები
    2.4. ობიექტი: სტუდენტები, საგნები, გამოცდები, შედეგები
    2.5. ობიექტი: ლექტორები, კათედრები, ჯგუფები

3. საპრობლემო სფერო: **სუპერმარკეტი**
    3.1. ობიექტი: პროდუქტი, ფირმა, ფასი, კატეგორია
    3.2. ობიექტი: კლიენტთა მომსახურება, სალარო/ჩეკი
    3.3. ობიექტი: ელ–შეკვეთა ბინაზე მიტანით
    3.4. ობიექტი: დღიური/თვიური/წლიური ვაჭრობა
    3.5. ობიექტი: საწყობში პროდუქციის აღრიცხვა

4. საპრობლემო სფერო: **აფთიაქი**
    4.1. ობიექტი: მედიკამენტი, ქვეყანა, ფასი, კატეგორია
    4.2. ობიექტი: კლიენტთა მომსახურება, სალარო/ჩეკი
    4.3. ობიექტი: ელ–შეკვეთა ბინაზე მიტანით
    4.4. ობიექტი: დღიური/თვიური/წლიური ეკონომუკური მაჩვენებლები
    4.5. ობიექტი: აფთიაქის საწყობი
    4.6. ან სხვ.

5. საპრობლემო სფერო: **საწარმოო ფირმა**

5.1. ობიექტი: პროდუქტი, ფასი, კატეგორია

5.2. ობიექტი: პროდუქტი, თვითღირებულება, ფასი, მოგება, რენტაბელობა

5.3. ობიექტი: პროდუქტი, ნედლეული, მიმწოდებელი, ნედლეულის_ფასი

5.4. ობიექტი: თვიური/წლიური შემოსავალი, ხარჯი, მოგება

5.5. ობიექტი: ნედლეულის და მზა პროდუქციის საწყობები

6. საპრობლემო სფერო: **კლინიკა**

6.1. ობიექტი: პაციენტი, დაავადება, მკურნალი_ექიმი

6.2. ობიექტი: ექიმი, ნოზოლოგიური_განყოფილება, ოთახი, ტელეფონი

6.3. ობიექტი: პაციენტი, დაავადება, მკურნალობის_ფასი

6.4. ობიექტი: თვიური/წლიური შემოსავალი, ხარჯი, მოგება

6.5. ობიექტი: ნოზოლოგიური_განყოფილება, საწოლების რაოდენობა, მკურნალობის_ვადა, მდგომარეობა

7. საპრობლემო სფერო: **მარკეტინგი**

3.1. ობიექტი: ავტომანქანების ბაზარი (ფირმა, მოდელი, სხვ.)

3.2. ობიექტი: ავტოპროფილაქტიკა, მომსახურების აღრიცხვა

3.3. ობიექტი: კომპიუტერების მაღაზია

3.4. ობიექტი: წიგნების მაღაზია სექციების მიხედვით

3.5. ობიექტი: პარფიუმერიის მაღაზია

8. საპრობლემო სფერო: **რესტორანი**

8.1. ობიექტი: მენიუ, კერძები, ფასები

8.2. ობიექტი: წინასწარი დაჯავშნის ამოცანა

8.3. ობიექტი: კლიენტთა მაგიდის მომსახურება

8.4. ობიექტი: შეკვეთების მიღება კერძების ბინაზე მიტანით

8.5. ობიექტი: თვიური/წლიური შემოსავალი, ხარჯი, მოგება

9.. საკურსო თემების დაზუსტება ან სხვა სფეროებიდან შერჩევა ხდება პროფესორის და სტუდენტის კონსულტაციების პროცესში.

7.4. დანართი N4

საკურსო პროექტის რეპორტის სარჩევი

1. ასაგები კომპიუტერული სისტემის ფუნქციონალური და არაფუნქციონალური მოთხოვნები: როლების და მათი ქმედებების UML დიაგრამები (UseCase, Activity);

2. ელექტრონული დოკუმენტების ფორმები: საწყისი, ნორმატიული, ოპერატიული და გამომავალი;

3. სისტემის მონაცემთა ბაზის შექმნა ცხრილებით (Tables) და ჩანაწერებით MsSQL Server 2014 პაკეტით;

6. მომხმარებელთა ინტერფეისები (სამუშაო ფორმები), MsVisual Studio.NET ინტეგრირებულ გარემოში (Application WPF), XAML და C# -ენის საფუძველზე;

7. საპრობლემო სფეროს ფუნქციური ამოცანის (ჯგუფური პროექტის დროს – ამოცანების) გადაწყვეტის პროცესის ავტომატიზაცია (დაპროგრამება და ტესტირება);

8. მომხმარებლის ინტერფეისებში MsSQL Server ბაზისთვის მონაცემთა მანიპულირების (ამორჩევა, შეცვლა, წაშლა) განხორციელება C# კოდებით;

9. სისტემის დემოვერსია და საპრეზენტაციო სლაიდები.

8. შენიშვნებისთვის:

იბეჭდება ავტორთა მიერ
წარმოდგენილი სახით

გადაეცა წარმოებას 15.05.2017 წ. ხელმოწერილია დასაბეჭდად 20.05.2017 წ. ოფსეტური ქაღალდის ზომა 60X84 1/16. პირობითი ნაბეჭდი თაბახი 7. ტირაჟი 100 ეგზ.

Verba volant, scripta manent

საქართველოს ტექნიკური უნივერსიტეტის
„IT- კონსალტინგის ცენტრი“
თბილისი, მ. კოსტავას 77