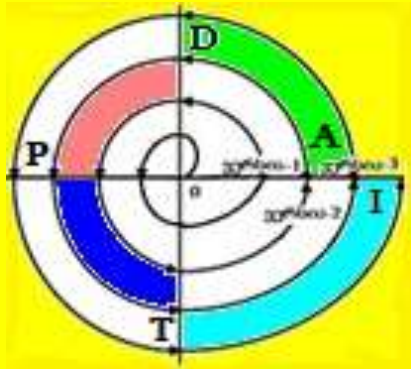


გია სურგულაძე, ეკატერინე თურქია,
 მარინე ბიტარაშვილი

პროგრამული ინჟინერიის საფუძვლები

(საკურსო პროექტის დამხმარე
 სახელმძღვანელო)



გია სურგულაძე, ეკატერინე თურქია,
მარინე ბიტარაშვილი

პროგრამული ინჟინერიის საფუძვლები

(საკურსო პროექტის დამხმარე სახელმძღვანელო)



რეკომენდებულია სტუ-ს
IT-კონსალტინგის ცენტრის
საგამომცემლო საბჭოს მიერ
5.02.22, ოქმი N7

თბილისი- 2022

უაკ 004.5

განიხილება გამოყენებითი პროგრამული აპლიკაციების დეველოპმენტის (კომპიუტერული სისტემების დაპროგრამების) საფუძვლების საწყისები. კერძოდ, შემოთავაზებულია აღნიშნულ დისციპლინაში საკურსო პროექტის შესრულების მეთოდოლოგია და პროგრამული ინსტრუმენტული საშუალებები. კერძოდ, Windows პლატფორმაზე MsVisual Studio.NET Framework გარემოში, C# ენის, MsSQL Server, MsAccess მონაცემთა ბაზების და ADO.NET დრაივერის პაკეტების გამოყენებით. პროგრამული აპლიკაციის დამუშავება ხდება პროგრამული უზრუნველყოფის კლასიკური, ობიექტ-ორიენტირებული მეთოდის სასიცოცხლო ციკლის ეტაპების საფუძველზე.

განკუთვნილია ინფორმატიკის საგანმანათლებლო პროგრამის ბაკალავრიატის სტუდენტებისთვის „პროგრამული ინჟინერიის“ კონცენტრაციით.

რეცენზენტები:

პროფ. თეიმერაზ სუხიაშვილი (ტ.მ.კ.),

პროფ. ლილი პეტრიაშვილი (ტ.მ.კ.)

პროფ. გია სურგულაძის რედაქციით

© სტუ-ს „IT კონსალტინგის სამეცნიერო ცენტრი“, 2022

ISBN 978-9941-8-3808-8



ყველა უფლება დაცულია, ამ წიგნის არც ერთი ნაწილი (იქნება ეს ტექსტი, ფოტო, ილუსტრაცია თუ სხვა) გამოყენება არც ერთი ფორმით და საშუალებით (ელექტრონული თუ მექანიკური), არ შეიძლება გამომცემლის წერილობითი ნებართვის გარეშე. საავტორო უფლებების დარღვევა ისჯება კანონით.

სარჩევი

შესავალი: საკურსო პროექტის მიზანი და ამოცანები	5
I თავი. ბიზნესპროცესების აღწერა და სისტემის დაპროექტება ..	7
1.1. პროცესებზე ორიენტირებული მოდელი – BPMN	7
1.2. ობიექტებზე ორიენტირებული მოდელი – UML: როლები/ფუნქციები: (UseCase/Activity-D)	13
1.3. კლასების აღწერა და კლასთაშორის ასოციაციის დიაგრამა (Class-D და StateChart-D)	17
1.4. ინტერფეისები და სცენარები (Sequence/Collaboration-D).....	18
1.5. საპრობლემო სფეროს ER მოდელი (Database-D)	20
II თავი. მონაცემთა ბაზების დაპროექტება და აგება	21
2.1. Ms SQL Server ბაზის შექმნა	21
2.1.1. ბაზის ცხრილების შექმნა (Tables)	23
2.1.2. ბაზის ცხრილების შევსება მონაცემებით	25
2.1.3. მონაცემთა ბაზის დიაგრამის აგება (Relationships)	28
2.2. Ms Access ბაზის შექმნა	30
2.2.1. ბაზის ცხრილების შექმნა (Tables)	30
2.2.2. ბაზის ცხრილების შევსება მონაცემებით	32
2.2.3. ბაზის ცხრილთაშორის კავშირების აგება (Relationships)	34
III თავი. პროგრამული აპლიკაცია SQL Server ბაზით	35
3.1. პროგრამული აპლიკაციების კავშირი მონაცემთა ბაზებთან (ADO.NET)	35
3.2. C# ენისა და SQL Server ბაზის ერთობლივი გამოყენება ADO.NET დრაივერით და DataGridView კლასით	42
3.2.1. ექსპერიმენტული მონაცემთა ბაზის მომზადება SQL Server პაკეტით	42
3.2.2. ახალი პროექტის შექმნა .NET პლატფორმაზე C# ენის გამოყენებით	44
3.2.3. პროექტთან მონაცემთა ბაზის მიერთება	45

3.2.3. DataGridView ელემენტის გააქტიურება და ცხრილის პარამეტრების განსაზღვრა	46
3.2.5. პროგრამული რეალიზაციის საკითხები	52
3.3. ADO.NET-ის DataSet-ობიექტის გამოყენება	58
3.4. პროგრამის რეფაქტორინგი და რეინჟინერინგი	72
3.5. პროგრამული აპლიკაციის მოდულური ტესტირება	78
IV თავი. მომხმარებლის ინტერფეისის დამუშავება	89
4.1. მომხმარებლის ინტერფეისის მთავარი მენიუს აგება	89
4.2. გრაფიკული მენიუს აგება	93
4.3. კონტექსტური მენიუს ვიზუალური აგება	96
4.4. ვიზუალური სტანდარტული დიალოგური საშუალებები ...	101
V თავი. Web-აპლიკაციების აგება ASP.NET ტექნოლოგიით	108
5.1. ASP.NET აპლიკაციის შექმნის ეტაპები	110
5.2. პროექტში ახალი ვებ-გვერდის დამატება	113
5.3. ინტერაქტიული Web-გვერდის შექმნა	115
5.4. DataSet / GridView ობიექტებთან მუშაობა და XML ფაილი ...	120
VI თავი. ინსტრუქციები სისტემის საპილოტო ვერსიისა და საკურსო პროექტის გასაფორმებლად	126
6.1. აპლიკაციის საპილოტო ვერსიის ტესტირება და სადემონსტრაციოდ მომზადება	126
6.2. საპრეზენტაციო ფაილის და ანგარიშის მომზადება	126
– გამოყენებული ლიტერატურა	127
– დანართი_1: საკურსო პროექტის სატიტულო გვერდი	129
– დანართი_2: საკურსო პროექტის ანგარიშის სარჩევი	130
– დანართი_3: საკურსო პროექტის თემები	131
– დანართი_4: საკურსო პროექტის ნიმუში	133

შესავალი:
საკურსო პროექტის მიზანი და ამოცანები

პროექტის მიზანია პროგრამული ინჟინერიის საფუძვლების დისციპლინაში Windows და Web აპლიკაციების დეველოპინგის (გამოყენებითი პროგრამული პაკეტების აგების) საწყისების შესწავლა. საერთაშორისო სტანდარტების მოთხოვნების გათვალისწინებით უახლესი ინფორმაციული ტექნოლოგიების (IT) პრაქტიკული გამოყენების უნარ-ჩვევების გამომუშავება. თანამედროვე უნიფიცირებული, ობიექტ-ორიენტირებული და პროცეს-ორიენტირებული ვიზუალური მეთოდების და ინსტრუმენტული საშუალებების ათვისება [1].

სტუდენტი ინდივიდუალურად (ან რამდენიმე სტუდენტი ჯგუფურად) მიიღებს კონკრეტულ დავალებას (ამოცანას) პროექტის შესასრულებლად რომელიმე საპრობლემო სფეროს კომპიუტერული მართვის საინფორმაციო სისტემის დაპროექტების და პროგრამული რეალიზაციის მიზნით.

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 ანაზრების მართვა, კრედიტების დეპარტამენტი და ა.შ., ბიზნესის და კომერციის სფერო: მარკეტინგის დეპარტამენტი, სასაწყობო მეურნეობა, სუპერმარკეტი და ა.შ. [2].

მაგალითისათვის ჩვენ წიგნში განვიხილავთ საილუსტრაციო ამოცანას – „ვირთაში ახალი თანამშრომლის მიღების და ხელფასის დარიცხვის ავტომატიზებულ სისტემას“.

ამოცანის გადასაწყვეტად საჭიროა:

პროგრამული ინჟინერიის საფუძვლები

1) აიგოს ფორმაში ახალი თანამშრომლის მიღების ამოცანის ბიზნესპროცესის BPMN (Business Process Model and Notation) დიაგრამა [3]. მის საფუძველზე შესაძლებელი იქნება კონკრეტული ამოცანის ფუნქციური პროცედურების სტრუქტურული ანალიზის ჩატარება;

2) განისაზღვროს ფუნქციების (მეთოდების) და მათი შემსრულებლების (როლების), აგრეთვე კონკრეტული ამოცანის შესრულების დიაგრამები (UseCase, Activity) UML ტექნოლოგიის გამოყენებით [2];

3) გამოვლინდეს დოკუმენტა ფორმები და ინფორმაცია:

– *საწყისი* (ნორმატიული: თანამდებობები, ხელფასები, დაქვითვების სკალა); – *ოპერატიული*: ყოველდღიური ან თვიური აღრიცხვის ტაბელები, შვებულების განრიგი, საავადმყოფო ბიულეტენი); – *გამომავალი* (უწყისი თანამშრომელთა ანგარიშებზე გადასარიცხად);

4) განისაზღვროს SQL Server მონაცემთა ბაზის სტრუქტურა, აიგოს ER დიაგრამა ცხრილებით (Tables), შეივსოს ისინი რამდენიმე სტრიქონით (არასრულად, ექსპერიმენტის მიზნით);

5) შემუშავდეს მომხმარებელთა ინტერფეისები (სამუშაო ფორმები), Visual Studio.NET Framework ინტეგრირებულ გარემოში Windows Forms Application და C# ენის საფუძველზე [4,5];

6) შემუშავდეს კონკრეტული ამოცანის (მაგ., ხელფასის დარიცხვის ან სხვ.) პროცესის შესრულების ალგორითმი (როგორც კლასის მეთოდი);

7) შემუშავდეს მომხმარებელის ინტერფეისის ამოცანების SQL Server ბაზასთან კავშირის და მონაცემთა ამორჩევა/ მოდიფიკაციის მეთოდები, შესაბამისი C# კოდებით;

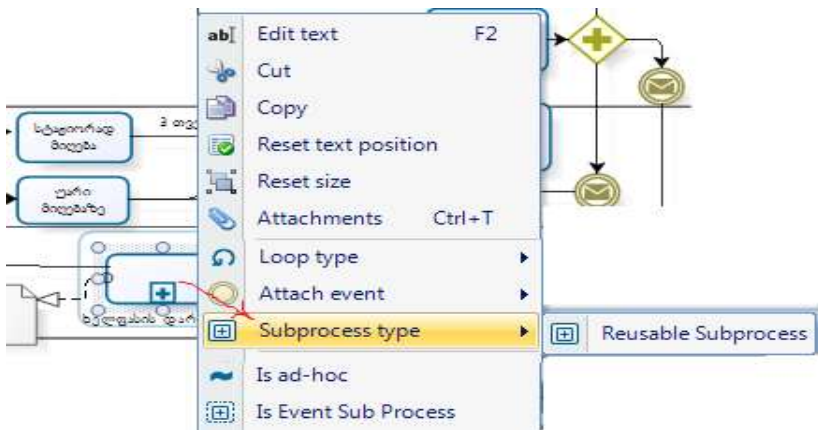
8) შეიქმნას საილუსტრაციო სტანდარტული მოთხოვნები, რომლებიც განთავსდება ინტერფეისის Button ღილაკებზე;

9) ჩატარდეს აგებული სისტემის მოდულური ტესტირება კონკრეტულ მონაცემებზე (Unit testing);

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

მთლიანი ბიზნესპროცესი დაყოფილია ქმედებებად (Activities), რომლებიც გამოსახულია მრგვალკუთხედებით, გადასასვლელები ქმედებებს შორის კი – ისრებით. დოკუმენტები, რომლებსაც საწყისად იყენებს ეს ქმედებები ან თვითონ ქმნის როგორც საშედეგოს, ნაჩვენებია კუთხეჩამოკვეცილი ფურცლის სახით. ეს ფურცლები შეერთებულია წყვეტილი ისრებით იმ ქმედებებთან, რომლებიც მათ ქმნის (ქმედებიდან გამომავალი ისარი) ან გამოიყენებს შესასვლელზე (ქმედებისკენ მიმართული ისარი) [3].

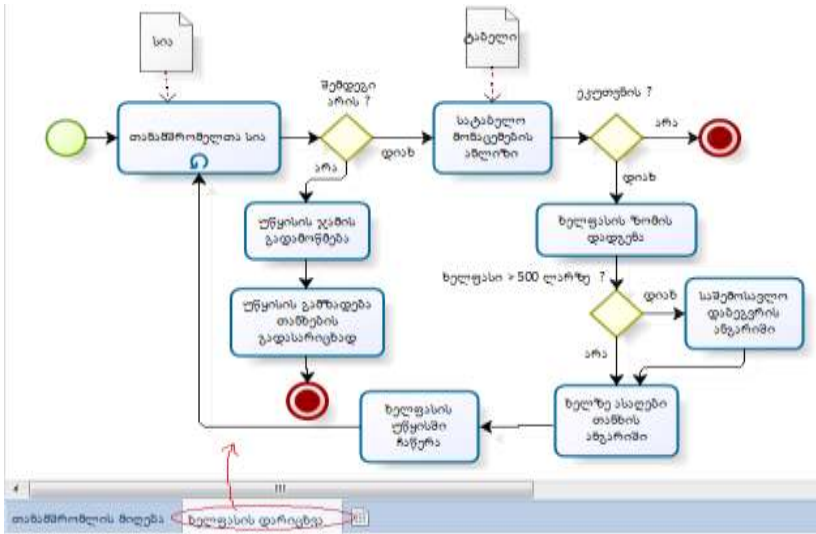
სქემაზე „+“-ით აღნიშნულია ქმედება, რომელიც ქვეპროცესია. მისთვის ცალკე აიგება დიაგრამა, მაგალითად, ბლოკზე „ხელფასის დარიცხვა“ მარჯვენა ღილაკით (ნახ.1.2) ავირჩევთ Subprocess პუნქტს.



ნახ.1.2. ქვეპროცესის შექმნის ინიცირება

1.3 ნახაზზე ნაჩვენებია ახლად აგებული ქვეპროცესის BPMN სქემა. ძირითადი პროცესის და ქვეპროცესების დიაგრამათა ასარჩევად გამოიყენება გადამრთველი სათაურებით (ქვედა მარცხენა კუთხეში).

პროგრამული ინჟინერიის საფუძვლები



ნახ.1.3. ხელფასის დარიცხვის ქვეპროცესის სქემა














ბიზნესპროცესების მოდელის ასაგებად გამოყენებულია BPMN ნოტაციის ელემენტები, რომლებიც ასახულია ქვემოთ მოცემულ ცხრილებში:

ქმედებები

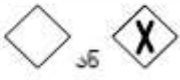





ამოცანა	სამუშაო ერთეული. თუ მას აქვს "+" სიმბოლო, იგი ქვეპროცესითაა
ტრანზაქცია	ლოგიკურად დაკავშირებული ქმედებების ერთობლიობაა
მოვლენითი ქვეპროცესი	მოთავსებულია სხვა პროცესის შიგნით. ის სრულდება, როცა ინიცირდება მისი საწყისი მოვლენა. მას შეუძლია მშობელი ქვეპროცესის შერყვეტა ან მის პარალელურად შესრულება
გამომმახებული ქმედება	არის შესასვლელი რერტილი გლობალურად განსაზღვრული ქვეპროცესის, რომელიც ხელმეორედ გამოიყენება ამ პროცესში

პროგრამული ინჟინერის საფუძვლები

ქმედებათა მარკერები
































	ქვეპროცესის მარკერი		შეტყობინების გაგზავნის ამოცანა
	ციკლის მარკერი		შეტყობინების მიღების ამოცანა
	პარალელური ეგზემპლარების მარკერი		მომხმარებლის ამოცანა
	მიმდევრობითი ეგზემპლარების მარკერი		არაავტომატიზებული ამოცანა
	ad hoc მარკერი		ამოცანა ბეზნესწესი
	კომპენსაციის მარკერი		ამოცანა სერვისი
			ამოცანა სცენარი

ლოგიკური ოპერატორები

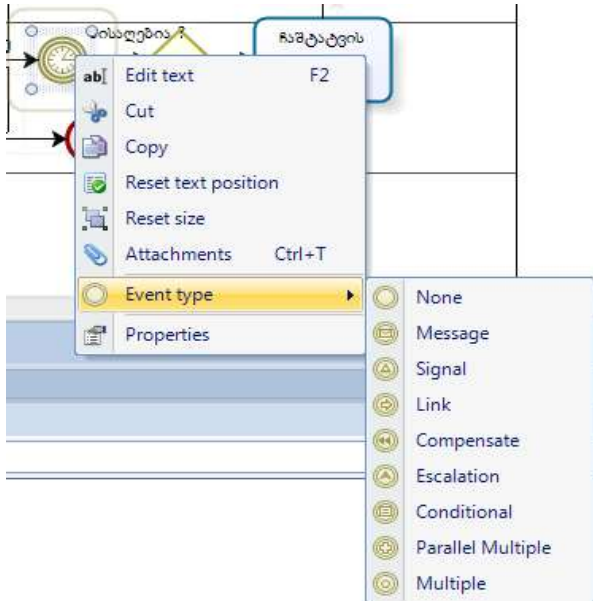
"ან" ოპერატორის გამორიცხვა მონაცემთა მართვისას (Data XOR)	
მოვლენითი "ან" ოპერატორი: ქმნის პროცესის ახალ ეგზემპლარს (Event XOR)	
"და" ოპერატორი (AND)	
"ან" ოპერატორი (OR): განშტოებისას აქტიურდება ერთი ან ყველა შტო, შერწყმისას ყველა მოქმედი შემავალი შტო იხურება	
რთული ოპერატორი, ამოღელირებს განშტოების და შერწყმის რთულ პირობებს	
მოვლენითი "და" ოპერატორი (ქმნის პროცესის ახალ ეგზემპლარს)	

პროგრამული ინჟინერიის საფუძვლები

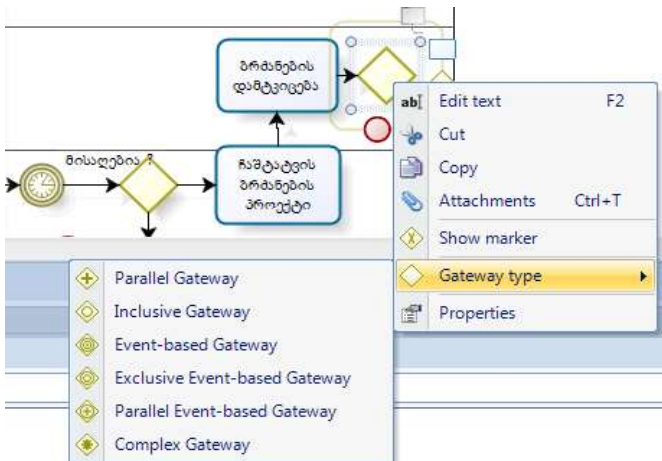
მოვლენები

მოვლენათა ტიპები	დაწყება	შუალედური	დასასრული
ჩვეულებრივი			
შეტყობინება			
წამზომი			
შეცდომა			
შეწყვეტა			
კომპენსაცია			
ბიზნესრესის შესრულება			
ბმული			
სიგნალი			
შედგენილი (სრულდება ერთი)			
პარალელური შედგენილი (სრულდება ყველა)			
ესკალაცია: საკითხის ატანა ორგ-იერარქიის ზედა დონეზე			
შეჩერება			

პროგრამული ინჟინერიის საფუძვლები



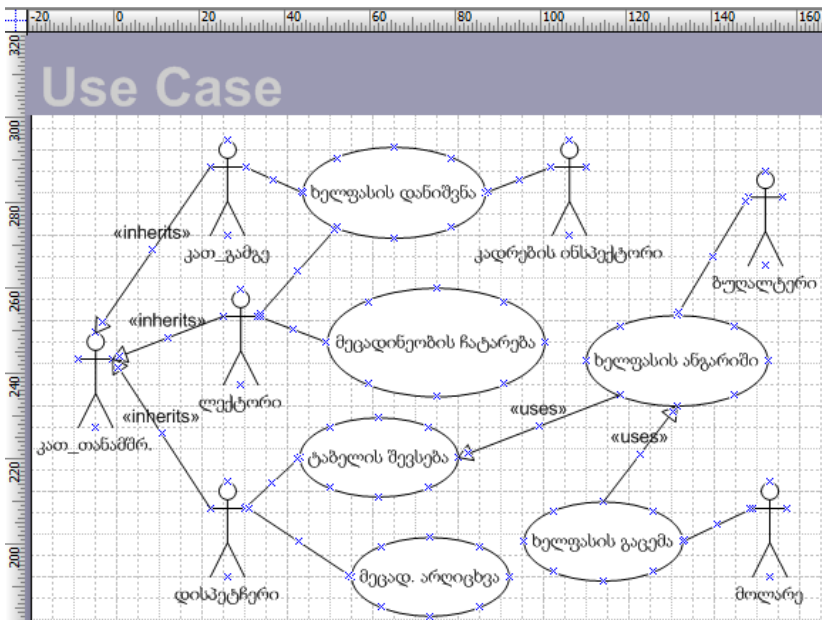
ნახ.1.4. მოვლენის ტიპის შერჩევა (Bizag გარემოში)



ნახ.1.5. ლოგიკური ოპერატორის ტიპის შერჩევა (Bizag გარემოში)

1.2. ობიექტებზე ორიენტირებული მოდელი – UML: როლები / ფუნქციები (Use Case / Activity-D)

BPMN-ის ალტერნატიული მეთოდოლოგიაა UML (Unified Modeling Language) [6-8]. იგი აქტიურად განვიხილოთ ბიზნესპროცესების ობიექტ-ორიენტირებული ანალიზისა და დაპროექტების MsVisio სისტემაში ჩვენი ამოცანის მოდელირების საკითხები. კერძოდ, პირველ რიგში უნდა განისაზღვროს ასაგები კომპიუტერული სისტემის ფუნქციური მოთხოვნები, თავისი ბიზნესპროცესებით და ბიზნესწესებით. 2.6 ნახაზზე მოცემულია ხელფასის დარიცხვის ამოცანის ბიზნესპროცესში მონაწილე მომხმარებელთა როლები (Actors) და ფუნქციები (Actions), გამოყენებით-შემთხვევათა (UseCase-D) დიაგრამის საფუძველზე.



ნახ.2.6. UseCase დიაგრამის ფრაგმენტი
(MsVisio გარემოში)

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. კათედრის გამგე, ლექტორები და დისპეტჩერი ქვეკლასებია განზოგადებული (Subclass, inheritance) კლასისა - კათედრის თანამშრომელი, ამიტომაც ისარი მიმართულია აქეთ.

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 500 ლარი).

ფუნქცია „მეცადინეობის ჩატარება“ ევალუა ლექტორს და იგი სადისპეტჩერო კომპიუტერში „თითის დაჭერით“ აფიქსირებს „ხელმოწერას“. არდაფიქსირება ნიშნავს „გაცდენას“.

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

ფუნქცია „ხელფასის ანგარიში“ ევალუა ბუღალტერს. კათედრაზე N თანამშრომელია სხვადასხვა ხელფასით. ამიტომ იგი კათედრიდან გადმოცემულ სატაბელო მონაცემებს შეადარებს თავის კომპიუტერში არსებულ ინფორმაციას და შემდეგ დაიანგარიშებს თითოეული ლექტორისათვის დარიცხულ

თანებს, დაქვითვებს, სხვადასხვა გადასახადს და ბოლოს ხელზე ასაღებ თანხას. შედეგები გადაიგზავნება შესაბამის ლექტორთა კონკრეტულ ანგარიშებზე ელექტრონული ანგარიშსწორების მიზნით. 1.6 ნახაზზე ყოველ გამოყენებით შემთხვევას ანუ ფუნქციას (ოვალს) შეესაბამება ერთი აქტიურობის ანუ ქმედების დიაგრამა (Activity Diagram).

თუ ფუნქციას რამდენიმე როლი ასრულებს (ნახ.1.7), მაშინ საჭიროა განისაზღვროს თითოეულის კონკრეტული ოპერაცია (მოქმედება) და შესრულების მიმდევრობის რეგლამენტი (ვინ, რა, როდის უნდა შეასრულოს).



ნახ.1.7

ქმედებათა დიაგრამა (Activity Diagram) ჰგავს ბიზნესპროცესების აღწერის BPMN სქემას, მაგრამ განსხვავებული ფორმით გამოირჩევა (Ms Visio ინსტრუმენტი).

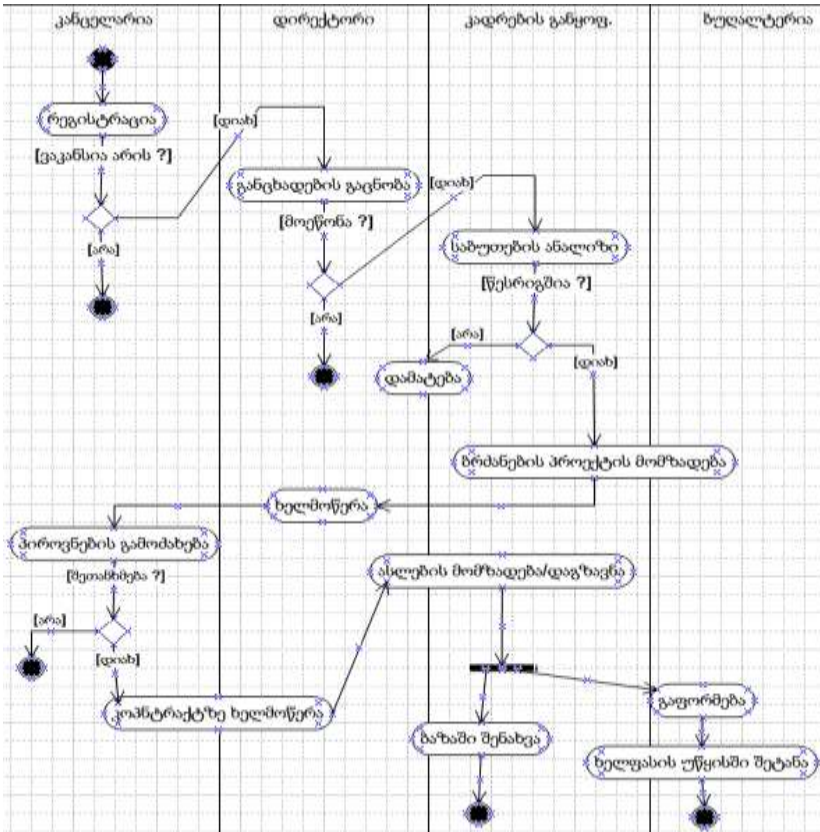
1.8 ნახაზზე განიხილება ამ ფუნქციის შესაბამისი აქტიურობის დიაგრამა: „კონტრაქტის გაფორმება ოფისში ახალი თანამშრომლის მისაღებად“.

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

პროგრამული ინჟინერიის საფუძვლები

გადაამოწმებს ვაკანსიის არსებობას და პიროვნების შრომის წიგნაკს.

Activity-D: ამოცანა - „ახალი თანამშრომლის მიღება“



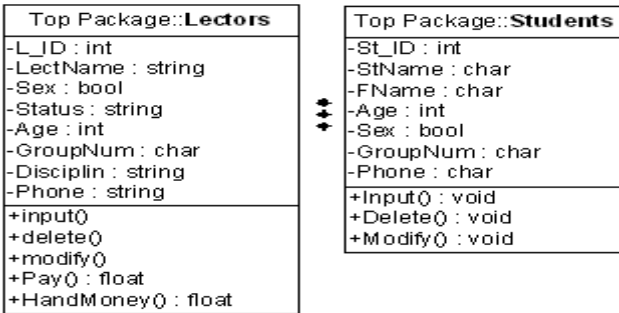
ნახ.1.8. Activity დიაგრამის ფრაგმენტი (MsVisio გარემოში)

თუ ყველაფერი წესრიგშია, მოამზადებს ბრძანების პროექტს და გადასცემს დირექტორს ხელმოსაწერად. კანცელარია უგზავნის შეტყობინებას (ურევკავს 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.

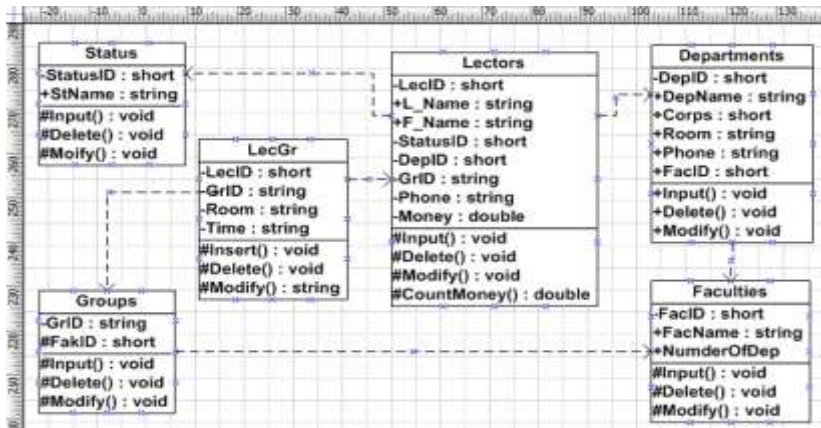
1.3. კლასების აღწერა და კლასთაშორის ასოციაციის დიაგრამა (Class-D, StateChart-D)

კლასი არის „დასახელების“, „კლასის მონაცემების“ და „კლასის მეთოდების“ ინკაფსულაცია. ჩვენი მართვის სფეროს შესაბამისი კლასები ასე უნდა გამოიყურებოდეს (ნახ.1.9):



ნახ.1.9. კლასები: Lectors და Students

კლასთაშორის კავშირების (Class Assotiation) ასაგებად Ms Visio გამოიყენება StateChart დიაგრამა (ნახ.1.10).

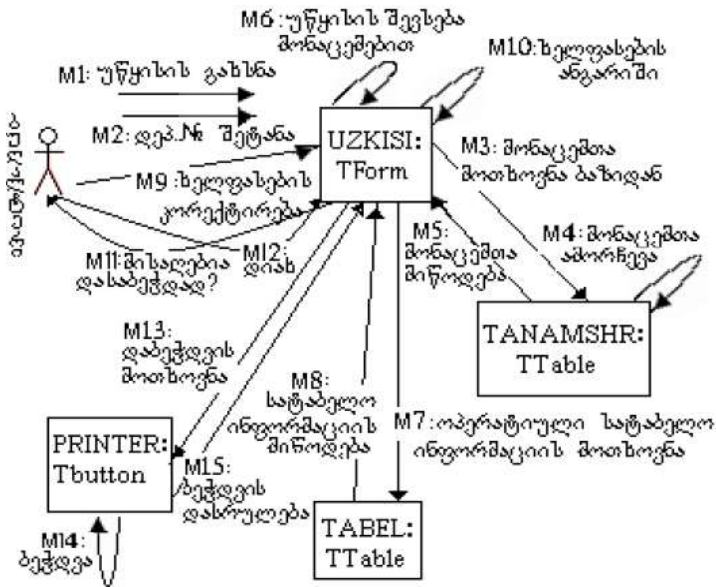


ნახ.1.10. კლასთაშორის კავშირები (StateChart-D)

1.4. ინტერფეისები და სცენარები
(Sequence-D, Collaboration-D)

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

სცენარის ასაგებად გამოიყენება მიმდევრობითობის (Sequence-D) და თანამოქმედების (Collaboration-D) დიაგრამები (ნახ.1.11-ა.ბ).



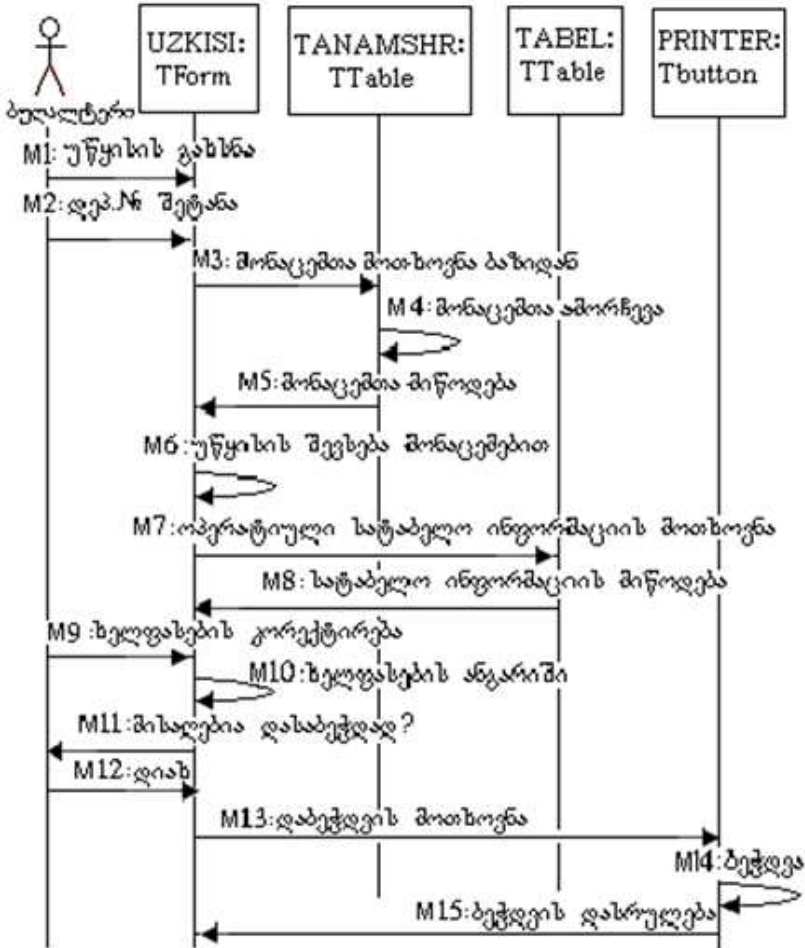
ნახ.1.11-ა. თანამოქმედების დიაგრამა (Collaboration-D)

თანამოქმედების დიაგრამაზე კლასებს შორის ურთიერთობა ასახულია შეტყობინებებისა და ინფორმაციული ნაკადების გაცვლის თვალსაზრისით ანუ რომელი კლასი რომელთან

პროგრამული ინჟინერიის საფუძვლები

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

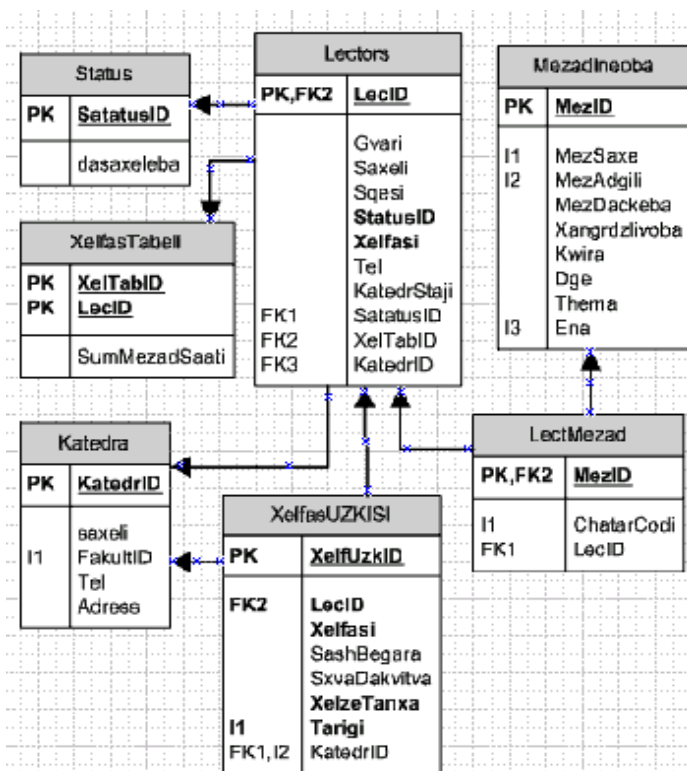
ამოცანა: „ხელფასის დარიცხვის უწყისის მომზადება და ბეჭდვა“



ნახ.1.11-ბ. მიმდევრობითობის დიაგრამა (Sequence-D)

1.5. საპრობლემო სფეროს ER მოდელი (Database-D)

საპრობლემო სფეროს კონცეპტუალური მოდელი ასახება არსთა დამოკიდებულების დიაგრამით (ER-D) ობიექტების, ატრიბუტებისა და ობიექტთაშორისი კავშირებით. დიაგრამას ვაგებთ ჩვენი კლასების საფუძველზე ხელფასის ამოცანისათვის MsVisio/Database ინსტრუმენტის გამოყენებით (ნახ.1.12).



ნახ.1.12. ER მოდელი

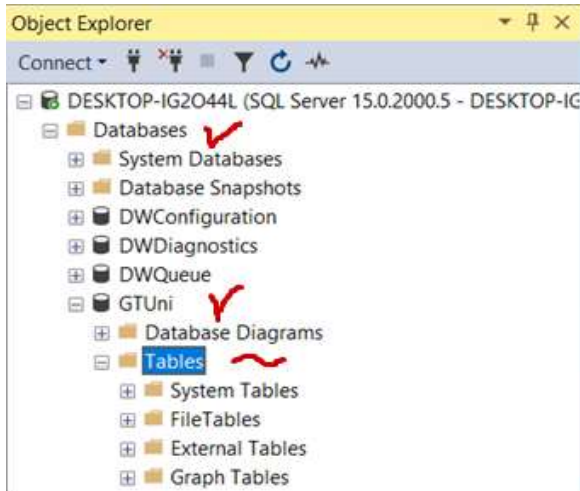
II თავი

მონაცემთა ბაზების დაპროექტება და აგება

2.1. Ms SQL Server ბაზის შექმნა

ჩვენ მიერ არჩეული დასაპროექტებელი საპრობლემო სფეროს კომპიუტერული სისტემის აგებას ვიწყებთ მისი შესაბამისი მონაცემთა ბაზის დაპროექტებით. საკურსო პროექტისათვის მონაცემთა ბაზას ვქმნით Ms SQL Server-ით [9]. შეიძლება როგორც Transact SQL ბრძანებების გამოყენებით (მაგალითად, Create Database ...), ასევე SQL Server Management Studio პაკეტით (გრაფიკული რედაქტორით).

მაგალითად, 2.1 ნახაზზე ჩვენ შევქმენით უნივერსიტეტის მონაცემთა ბაზა (GTUni).



ნახ.2.1. GTUni მონაცემთა ბაზის შექმნა Ms SQL Server Management Studio -ში

2.1.1. ბაზის ცხრილების შექმნა (Tables)

ჩვენი პროექტისთვის შევქმნათ ექვსი ცხრილი, რომლებშიც განთავსებულია ინფორმაცია სტუდენტების, ლექტორების, ჯგუფების, აკადემიური საგნების და დეპარტამენტების შესახებ.

მე-6 ცხრილი კი იქნება მათი დამაკავშირებელი, მაგალითად, ლექცია ან გამოცდა და ა.შ. 2.2 ნახაზზე მოცემულია GTUni ბაზის ეს ექვსი ცხრილი.



ნახ.2.2.

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 ბიზნესსწესს (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 **ლექტორის**-ს და **ჯგუფის**-ს პირდაპირი კავშირით (**განმეორებადი ველების პრობლემა**!). ამისათვის შემოტანილია დამატებითი ცხრილი (რეაღაცია) **ლექტორი_ჯგუფი** (Lector_Group). მასში შედგენილი ინდექსი იქნება L_JG, რომელიც უკავშირდება ცალკე **ლექტორს** (L_ID) და **ჯგუფს** (Gr_Nom). 2.3 ა-ვ ნახაზებზე მოცემულია ეს სტრუქტურები.

პროგრამული ინჟინერის საფუძვლები

Column Name	Data Type	Allow Nulls
St_ID	int	<input type="checkbox"/>
Name	nchar(15)	<input checked="" type="checkbox"/>
FirstName	nchar(15)	<input checked="" type="checkbox"/>
Sex	nchar(10)	<input checked="" type="checkbox"/>
Gr_Nom	nchar(10)	<input checked="" type="checkbox"/>
Mob	nchar(10)	<input checked="" type="checkbox"/>
eMail	nchar(20)	<input checked="" type="checkbox"/>
		<input type="checkbox"/>

ნახ.2.3-ა. „Student“ ცხრილის სტრუქტურა

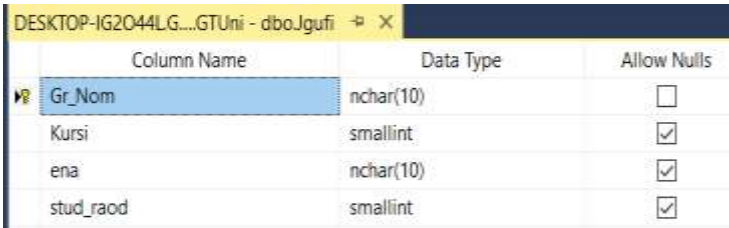
Column Name	Data Type	Allow Nulls
L_ID	smallint	<input type="checkbox"/>
Name	nchar(25)	<input checked="" type="checkbox"/>
FirstName	nchar(20)	<input checked="" type="checkbox"/>
Age	smallint	<input checked="" type="checkbox"/>
Sex	nchar(10)	<input checked="" type="checkbox"/>
Status	nchar(20)	<input checked="" type="checkbox"/>
Mob	nchar(10)	<input checked="" type="checkbox"/>
Dep_ID	smallint	<input checked="" type="checkbox"/>
		<input type="checkbox"/>

ნახ.2.3-ბ. „ლექტორი“ ცხრილის სტრუქტურა

Column Name	Data Type	Allow Nulls
DepID	smallint	<input type="checkbox"/>
Name	nchar(30)	<input checked="" type="checkbox"/>
Head_ID	smallint	<input checked="" type="checkbox"/>
Address	nchar(35)	<input checked="" type="checkbox"/>
Tel	nchar(10)	<input checked="" type="checkbox"/>

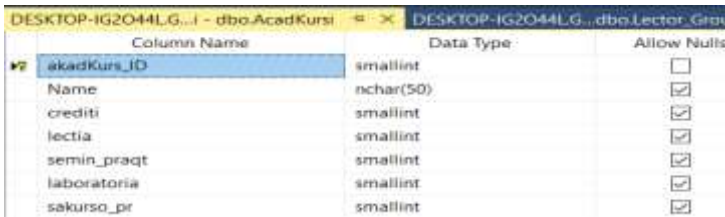
ნახ.2.3-გ. „დეპარტამენტი“ ცხრილის სტრუქტურა

პროგრამული ინჟინერის საფუძვლები



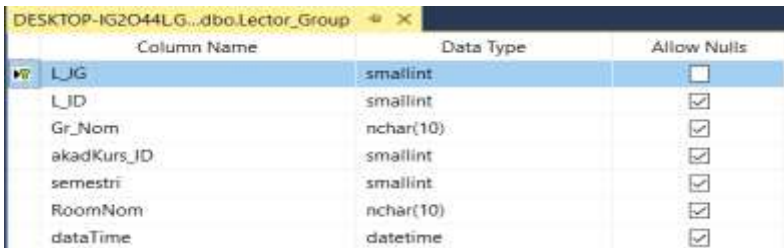
Column Name	Data Type	Allow Nulls
Gr_Nom	nchar(10)	<input type="checkbox"/>
Kursi	smallint	<input checked="" type="checkbox"/>
ena	nchar(10)	<input checked="" type="checkbox"/>
stud_raod	smallint	<input checked="" type="checkbox"/>

ნახ.2.3-დ. „ჯგუფი“ ცხრილის სტრუქტურა



Column Name	Data Type	Allow Nulls
akadKursi_ID	smallint	<input type="checkbox"/>
Name	nchar(50)	<input checked="" type="checkbox"/>
crediti	smallint	<input checked="" type="checkbox"/>
lectia	smallint	<input checked="" type="checkbox"/>
semin_praqt	smallint	<input checked="" type="checkbox"/>
laboratoria	smallint	<input checked="" type="checkbox"/>
sakurso_pr	smallint	<input checked="" type="checkbox"/>

ნახ.2.3-ე. „აკადემიური კურსი“ (საგნის) ცხრილის სტრუქტურა



Column Name	Data Type	Allow Nulls
L_JG	smallint	<input type="checkbox"/>
L_ID	smallint	<input checked="" type="checkbox"/>
Gr_Nom	nchar(10)	<input checked="" type="checkbox"/>
akadKursi_ID	smallint	<input checked="" type="checkbox"/>
semestri	smallint	<input checked="" type="checkbox"/>
RoomNom	nchar(10)	<input checked="" type="checkbox"/>
dateTime	datetime	<input checked="" type="checkbox"/>

ნახ.2.3-ვ. „ლექტორის-ჯგუფი“ ცხრილის სტრუქტურა

შესაძლებელია ასევე სხვა ცხრილების დამატება მონაცემთა ბაზაში, როგორცაა მაგალითად, სალექციო აუდიტორიები, ლაბორატორიები, ფაკულტეტები, ხელფასის უწყისები, საგანთა საკრედიტო სისტემის ცხრილები, სტუდენტთა სტატუსის ცხრილები, გამოცდის ჩატარების ცხრილები, გამოცდის შედეგების აღრიცხვის ცხრილები და ა.შ.

პროგრამული ინჟინერის საფუძვლები

საკურსო პროექტის შესრულების პირველივე სტადიაზე, სტუდენტ(ებ)ი (დამოუკიდებლად და ხელმძღვანელთან შეთანხმებით) განსაზღვრავენ ბაზის შემადგენელი ცხრილების რაოდენობას და შინაარსობრივ მხარეს. (პროექტი ხშირად არის გუნდური, მასში 2-5 სტუდენტი მონაწილეობს). ცხრილთა რაოდენობაც ამაზეა დამოკიდებული.

2.1.2. ბაზის ცხრილების შევსება მონაცემებით

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

დავალბა_2. Ms_SQL Server ბაზის ცხრილებში შეიტანეთ მონაცემები. სტრიქონების რაოდენობა უნდა იყოს საკმარისი ექსპერიმენტების ჩასატარებლად (მინიმუმ 5 სტრიქონი).

2.4, ა-ვ ნახაზებზე ნაჩვენებია ზემოთ აღწერილი სტრუქტურების შესწავლის საწყისი შევსებული ცხრილები.

SL_ID	Name	FirstName	Sex	Gr_Nom	Mob	eMail
1	ბერულავა	ანა	მდედრ	108851	599123456	aberul@gtu.ge
2	გულუა	დავით	მამრ	108850	577123456	dgulua@gtu.ge
3	დოლიძე	სანდრო	მამრ	108850	593123456	sdoli93@gtu.ge
4	ბაზია	გიორგი	მამრ	108852	599001122	gbakhi@gmail.com
5	თურქია	ქეთი	მდედრ	108852	55534455	kturkia@gtu.ge
6	კოსტავა	დავით	მამრ	108851	577454545	dkosta@gtu.ge
7	მაისურაძე	რეატი	მამრ	108851	577131313	rmaisa@gmail.com
8	მესხური	ცოტნე	მამრ	108853	577252525	tsotne7@gmail.com
9	მებელიშვილი	ამიკო	მამრ	108850	599112112	anebuli@gtu.ge
10	ულუელიძე	ვანო	მამრ	108850	593669977	vugheli@gtu.ge
11	ხრიკული	მია	მდედრ	108853	591222324	mikhru@gtu.ge
21	ვაჩნაძე	საბა	მამრ	108950	577303030	svachna@gtu.ge
22	თოფურია	გიორგი	მამრ	108950	593757575	gtopuri@gmail.com

ნახ.2.4-ა. ცხრილი „სტუდენტი“

პროგრამული ინჟინერიის საფუძვლები

DESKTOP-IG2O44LG... - dbo.Department		DESKTOP-IG2O44LG... - dbo.Department			
DeptID	Name	Head_ID	Adress	Tel	
1	პროგრამული ინჟინერია	1	კ-6; 225დ	2309525	
2	კომპიუტერული ინჟინერია	51	კ-6; 318დ	2306677	
3	ინფორმაციული ტექნოლოგიები	32	კ-6; 323დ	2112233	
4	გამოყენებითი ინფორმატიკა	60	კ-6; 312	2445566	
5	ხელოვნური ინტელექტი	80	კ-6; 555ა	2335577	

ნახ.2.4-ბ. ცხრილი „დეპარტამენტი“

DESKTOP-IG2O44LGTUni - dbo.Lector		DESKTOP-IG2O44LG... - dbo.Department		DESKTOP-IG2O44LG... - dbo.Department				
L_ID	Name	FirstName	Age	Sex	Status	MoB	Dep_ID	
1	სურგულაძე	გია	55	მამრ	პროფ	599373737	1	
2	ბატაძე	თენგიზ	53	მამრ	პროფ	577535353	1	
3	თოფურია	ნინო	30	მდედრ	ასოც.პროფ.	555000200	1	
4	ლვინვაძე	გელა	57	მამრ	პროფ	577666777	1	
5	თურქია	ეკა	31	მდედრ	მოწე.პროფ.	577252525	1	
6	პეტრიაშვილი	ლია	33	მდედრ	პროფ	577404040	1	
31	მეფარიშვილი	ბადრი	58	მამრ	პროფ	599112233	3	
32	ქართველიშვილი	სოსო	32	მამრ	პროფ	599445566	3	
33	ამილახვარი	ნუგზარ	37	მამრ	პროფ	599332211	3	
51	კეცაძე	შინა	37	მდედრ	პროფ	591377373	2	
52	კაკუბავა	ივერი	45	მამრ	პროფ	591515151	2	
60	ახონაძე	მურაბ	55	მამრ	პროფ	577111111	4	
61	კაშიბაძე	მარიამ	34	მდედრ	ასოც.პროფ.	599775577	4	
62	ოხანაშვილი	მაია	31	მდედრ	ასოც.პროფ.	599557755	4	
80	ჩხიძე	მარიამ	33	მდედრ	პროფ	591335577	5	

ნახ.2.4-გ. ცხრილი „ლექტორი“

DESKTOP-IG2O44LGTUni - dbo.Jgufi		DESKTOP-IG2O44LG...GTUni - dbo.Jgufi		
Gr_Nom	Kursi	ena	stud_raod	
108850	4	ქართული	20	
108851	4	ქართული	26	
108852	4	ქართული	16	
108853	4	ქართული	25	
108058	3	ინგლისური	35	
108039	3	რუსული	20	

ნახ.2.4-დ. ცხრილი „ჯგუფი“

პროგრამული ინჟინერის საფუძვლები

akadKurs_ID	Name	credits	lectia	semin_pragt	laboratoria	sakursa_pr
1	დამოგრამების საფუძვლები	6	15	0	45	0
2	შესავალი ქველურ ტექნოლოგიებში	5	15	30	0	0
3	ინფო-ტექნოლოგიები	5	15	0	30	0
4	პროგრამული ინჟინერიის საფუძვლები	6	15	0	30	15
5	აპლიკაციების დამოგრამება და მონაცემთა მენეჯმენტი	5	15	30	0	0
6	პროგრამული პროდუქტების დეველოპმენტი	6	15	0	30	15
7	ღრუბლოვანი ტექნოლოგიები	5	15	0	15	15
8	მონაცემთა ბაზების დამოგრამება	5	145	15	0	15
9	ბელოცური ინტელექტის საფუძვლები	5	15	30	0	0
10	კომპიუტერული ინჟინერიის საფუძვლები	5	15	30	0	0

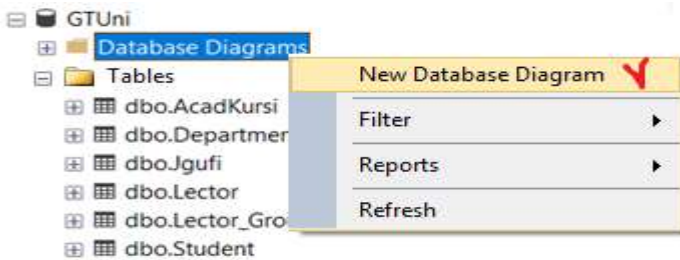
ნახ.2.4-ე. ცხრილი „აკადემიური კურსი“ (საგანი) კრედიტებით და საათებით

L_JG	L_ID	Gr_Nom	akadKurs_ID	semestri	RoomNom	dateTime
1	1	108850	6	1	06-207ა	2022-09-03 09:00:00..
2	1	108851	6	1	06-207ა	2022-09-03 11:00:00..
3	3	108852	6	1	06-204ა	2022-09-04 09:00:00..
4	3	108853	6	1	06-204ა	2022-09-04 13:00:00..
5	3	108850	7	1	06-230დ	2022-09-05 14:00:00..
6	3	108851	7	1	06-230დ	2022-09-05 14:00:00..
7	6	108952	5	2	06-207ა	2022-09-05 10:00:00..
8	6	108953	5	2	06-207ა	2022-09-04 10:00:00..
9	33	108850	8	2	06-405ბ	2022-09-06 09:00:00..
10	33	108851	8	2	06-405ბ	2022-09-06 09:00:00..
11	32	108058	10	2	09-208	2022-09-03 14:00:00..
12	51	108039	2	1	06-324	2022-09-03 10:00:00..

ნახ.2.4-ვ. ცხრილი „ლექტორის ჯგუფი“ საგნების მიხედვით

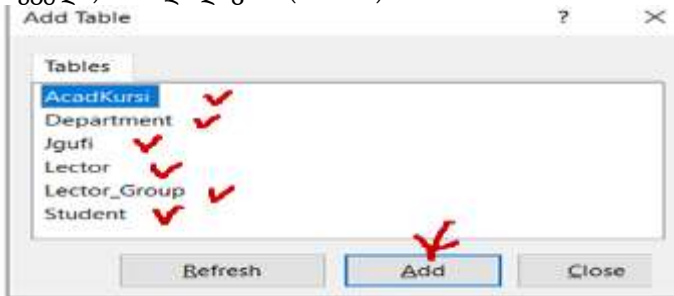
2.1.3. მონაცემთა ბაზის დიაგრამის აგება (Relationships in SQL)

მონაცემთა ბაზის ცხრილთაშორის კავშირების ასაგებად MsSQL Server Management Studio -ში GTUni-ზე თავუს მერჯვენა ღილაკით კონტექსტური მენიუდან გამოვიყენოთ Nedw Database Diagram (ნახ.2.5-ა).



ნახ.2.5 ცხრილთაშორის კავშირების აგების სტრიქონი

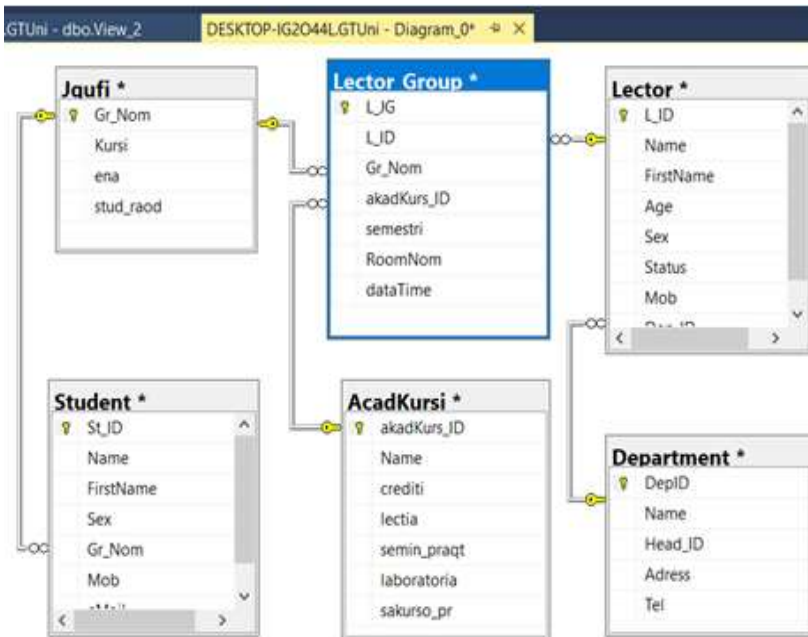
შედეგად სისტემა მოგვცემს ცხრილების არჩევის საშუალებას (2, 3 ან ყველა) Add ღილაკით (ნახ.2.6).



ნახ.2.6

ჩვენი შემთხვევისათვის ავირჩიოთ მიმდევრობით ყველა ცხრილი. უნდა მივიღოთ ისეთი სახის რელაციური კავშირების დიაგრამა, როგორც მოცემულია 2.6 ნახაზზე. თუ რომელიმე კავშირი არაა, მაშინ იგი ჩვენ თვითონ თავუს დახმარებით უნდა შევაერთოთ. სისტემა ამოწმებს დაკავშირების მართებულობას.

პროგრამული ინჟინერიის საფუძვლები



ნახ.2.6. ცხრილთაშორის კავშირების დიაგრამა

ამგვარად, ჩვენ შევქმენით GTUni მონაცემთა ბაზა MS SQL Server 2019/22 პაკეტის დახმარებით და იგი მზადაა მომავალი გამოყენებისთვის.

ამჯერად საჭიროა აიგოს მომხმარებლის ინტერფეისი Ms Visual Studio .NET 2015/2019 პლატფორმაზე და მოხდეს მონაცემთა ბაზასთან ინტერაქტიული პროცედურების შესრულება. კერძოდ Windows Forms Application-იდან მონაცემების ამორჩევა და მონაცემთა ბაზის განახლება C#-ის Insert, Update და Delete მეთოდებით. ასეთი CRUD ოპერაციები განხილულია მე-3 თავში.

2.2. Ms Access ბაზის შექმნა

ალტერნატიული მონაცემთა ბაზის სახით შესაძლებელია სხვა SQL (რელაციური ტიპის RDBMS) მონაცემთა ბაზების მართვის სისტემების გამოყენებაც. მაგალითად, Ms Access, PostgreSQL (უფასო პაკეტი), MySQL, MariaDB (უფასო პაკეტი), Oracle და სხვ. [10].

ჩვენ ამჯერად განვიხილავთ Ms Access, რომელიც საოფისე პაკეტის ერთ-ერთი ძირითადი კომპონენტია და მომხმარებელთა ფართო წრეზეა ორიენტირებული.

დავალემა: ამუშავეთ Ms_Access პაკეტი და შექმენით თქვენი პროგრამული აპლიკაციისთვის საჭირო მონაცემთა ბაზა და შემდეგ ცხრილების სტრუქტურები.

ჩვენ მაგალითისთვის ვიხილავთ უნივერსიტეტის მონაცემთა ბაზას (DB_AccessUni.accdb) ოთხი ცხრილით, რომლებშიც განთავსებულია ინფორმაცია სტუდენტების, ლექტორების, ჯგუფების და აკადემიური საგნების შესახებ.

2.2.1. ბაზის ცხრილების შექმნა (Tables)

ცხრილში *სტუდენტი* (Student) პირველადი გასაღებური ატრიბუტია St_ID ინდექსი, მისი მეორეული გასაღები - Jg_ID, რომლითაც უკავშირდება ცხრილს *ჯგუფი* (Jgufi), პირველადი ინდექსით Jg_ID. ესაა კავშირი 1:N, რომელიც ასახავს ბიზნესსწესს (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 *ლექტორის* და *ჯგუფის* პირდაპირი კავშირით (განმეორებადი ველების პრობლემა!). ამისათვის შემოტანილია დამატებითი ცხრილი (რელაცია) *ლექტორი_ჯგუფი* (Lect_Jgufi). მასში შედგენილი

პროგრამული ინჟინერის საფუძვლები

ინდექსი იქნება Lec_ID+Jg_ID, რომლებიც უკავშირდება ცალ-ცალკე **ლექტორს** და **ჯგუფს** (ნახ.2.7, ა-დ).

Field Name	Data Type
St_ID	Number
Gvari	Text
Saxeli	Text
Sqesi	Text
dab_celi	Number
Jg_ID	Number
telefoni	Text

ნახ.2.7-ა. „სტუდენტი“ ცხრილის სტრუქტურა

Field Name	Data Type
Lec_ID	AutoNumber
Gvari	Text
Saxeli	Text
Sqesi	Text
Dab_celi	Number
Statusi	Text
Xelfasi	Number
Kat_nom	Number
telefoni	Text

ნახ.2.7-ბ. „ლექტორი“ ცხრილის სტრუქტურა

Field Name	Data Type
Jg_ID	Number
Kursi	Number
Ena	Text
StudRaod	Number

ნახ.2.7-გ. „ჯგუფი“ ცხრილის სტრუქტურა

Field Name	Data Type
Lec_ID	Number
Jg_ID	Number
Sagani	Text
ECTS	Number

ნახ.2.7-დ. „ლექტორი_ჯგუფი“ ცხრილის სტრუქტურა

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

2.2.2. ბაზის ცხრილების შევსება მონაცემებით

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

დავალება. Ms_Access ბაზის ცხრილებში შეიტანეთ მონაცემები. სტრიქონების რაოდენობა უნდა იყოს საკმარისი ექსპერიმენტების ჩასატარებლად (მინიმუმ 5–20 სტრიქონი).

2.8 ა–დ ნახაზებზე ნაჩვენებია შევსებული ცხრილები.

St_ID	Gvari	Saxeli	Sqesi	dab_cel	Jg_ID	telefoni
1	ალავიძე	ალეკო	მამარ.	1990	108050	222-22-20
2	ზურდული	ნინო	მდედრ.	1991	108050	137-33-33
3	ზურდგლა	დიტო	მამარ.	1989	108835	599-10-20-20
4	გაბედავა	ვახტანგ	მამარ.	1992	108835	333-67-89
5	გაბელია	ნაზი	მდედრ.	1992	108935	222-45-67
6	დანელია	მიმოზა	მდედრ.	1990	108935	577-44-44-45
7	ხვიტია	ბუკუ	მამარ.	1992	108935	593-45-67-89
8	აკოფოვი	რობერტინო	მამარ.	1989	108051	297-11-11-11
9	დოლიძე	რიჩარდი	მამარ.	1990	108051	597-34-56-78
10	ზარანდია	მუშნი	მამარ.	1980	108836	597-12-23-34
11	თოფურია	ბლაზი	მდედრ.	1991	108936	577-10-10-10
12	კეკელია	კეკელა	მდედრ.	1992	108936	579-30-30-30
13	გალოგრე	ხვიჩა	მამარ.	1989	108937	270-44-44
14	დუნდუა	გოჩა	მამარ.	1991	108937	599-22-33-44
15	ვასაძე	სოკრატე	მამარ.	1985	108052	577-33-67-55
16	ჯალალონია	მაცი	მამარ.	1992	108937	577-99-00-00

ნახ.2.8-ა. ცხრილი „სტუდენტი“

Lec_ID	Gvari	Saxeli	Sqesi	Dab_peli	Statusi	Xelfasi	Kat_noir	telefoni
6	მართალი	ანი	მდედრ.	1970	ასოც.პროფესორი	864	94	599-23-23-23
7	კუცია	თეა	მდედრ.	1987	ლამორანტი	144	94	577-12-13-14
8	გაბედავა	ომიკო	მამარ.	1950	სრ.პროფესორი	1296	94	577-33-55-22
9	დვალი	დავითი	მამარ.	1950	სრ.პროფესორი	1296	86	599-99-90-90
10	ფიფია	კოჩი	მამარ.	1960	ასოც.პროფესორი	936	86	233-33-46
11	მეფარია	თეარისა	მდედრ.	1980	ას.პროფესორი	288	94	593-55-55-55
12	ნიინძე	ლია	მდედრ.	1980	ასოც.პროფესორი	864	94	577-78-89-90
13	ოდიშორია	ოდიშა	მამარ.	1956	ას.პროფესორი	576	86	236-37-38
14	სამუხრამე	ვახშაშა	მამარ.	1970	ასოც.პროფესორი	864	51	577-88-99-00

ნახ.2.8-ბ. ცხრილი „ლექტორი“

პროგრამული ინჟინერის საფუძვლები

Student	Lector	Jgufi	Lect_Jgufi	Relatio
Jg_ID	Kursi	Ena	StudRaod	
108050	2	ქართული	29	
108051	2	ქართული	30	
108059	2	რუსული	12	
108835	4	ქართული	29	
108836	4	ქართული	25	
108935	3	ქართული	28	
108936	3	ქართული	30	
108937	3	ქართული	17	
108940	3	ინგლისური	20	

ნახ.2.8-გ. ცხრილი „ჯგუფი“

ჩანაწერის წინ „+“ სიმბოლო ხსნის კავშირს მეორე, იერარქიულად დაქვემდებარებულ ცხრილთან (ნახ.2.8-ე).

Student	Lector	Jgufi	Lect_Jgufi	Katedi
Lec_ID	Gvarti	Saxeli	Sqesi	
6	ნარათელი	ანი	ქალი	
7	კუცია	თეა	ქალი	
8	გაბედავა	ოძიკო	კაცი	
	Jg_ID	Sagani	Add	
5	კომპიუტერის არქიტექტურა			
6	კომპიუტერის არქიტექტურა			
7	სერვერული ტექნოლოგიები			
8	სერვერული ტექნოლოგიები			
9	კომპიუტერის არქიტექტურა			
9	დევალი	დავითი	კაცი	
10	ფიფია	კოჩი	კაცი	

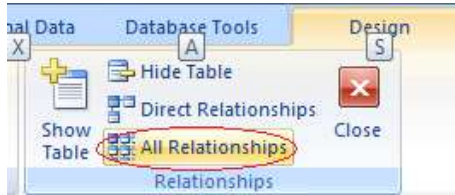
ნახ.2.8-დ. ცხრილი „ლექტორი_ჯგუფი“ საგნების მიხედვით

Student	Lector	Jgufi	Lect_Jgufi	Relatio
Lec_ID	Jg_ID	Sagani		
5	5	კომპიუტერის არქიტექტურა		
6	6	კომპიუტერის არქიტექტურა		
7	7	სერვერული ტექნოლოგიები		
8	8	სერვერული ტექნოლოგიები		
8	8	სერვერული ტექნოლოგიები		
9	9	კომპიუტერის არქიტექტურა		
9	7	მათემატიკა		
9	8	მათემატიკა		
10	12	მონაცემთა ბაზები		
10	13	მონაცემთა ბაზები		

ნახ.2.8-ე. კავშირი ორ ცხრილს შორის

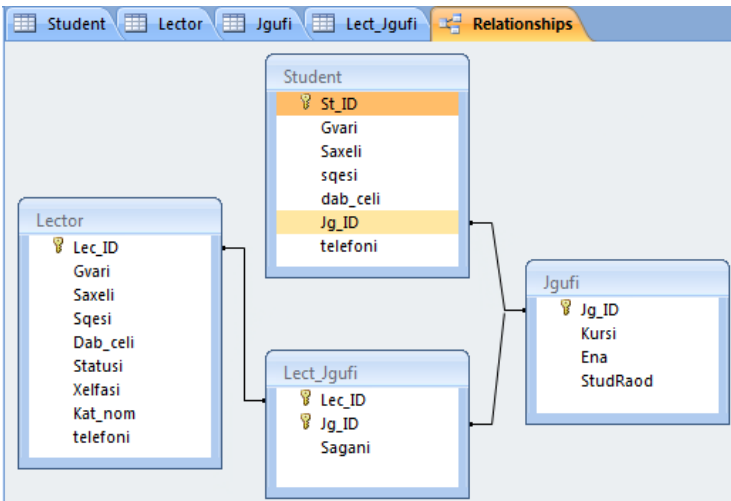
2.2.3. მონაცემთა ბაზის ცხრილთაშორის კავშირების აგება (Relationships)

მონაცემთა ბაზის ცხრილთაშორის კავშირების ასაგებად აქვე მენიუდან გამოვიყენოთ Database Tools -> Relationships (ნახ.2.9-ა).



ნახ.2.9-ა. ცხრილთაშორის კავშირების დიაგრამა

შედეგში უნდა მივიღოთ ისეთი სახის რელაციური კავშირების დიაგრამა, როგორც მოცემულია 2.9-ბ ნახაზზე.



ნახ.2.9-ბ. ცხრილთაშორის კავშირების დიაგრამა

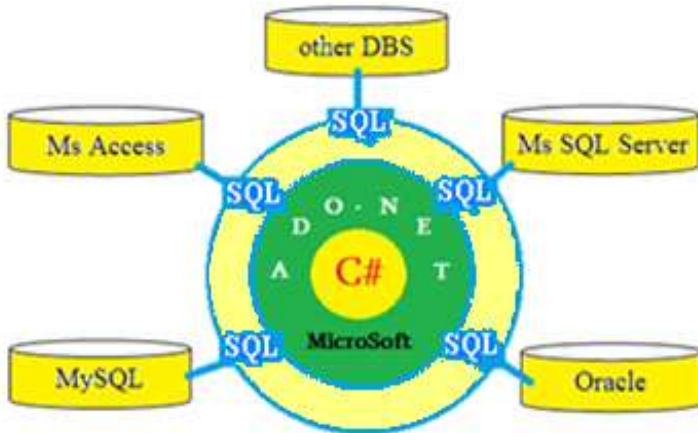
III თავი

პროგრამული აპლიკაცია SQL Server ბაზით

3.1. პროგრამული აპლიკაციების კავშირი მონაცემთა ბაზებთან (ADO.NET დრაივერი)

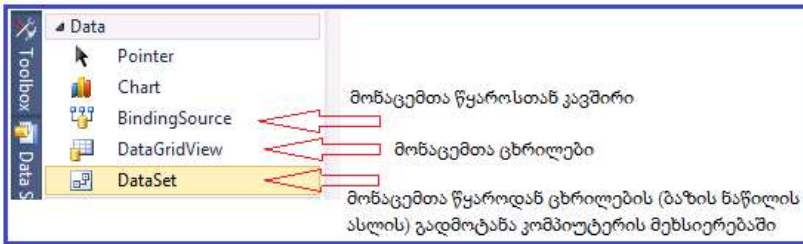
მომხმარებელთა პროგრამულ აპლიკაციებს (დანართებს) აუცილებლად ესაჭიროება ურთიერთქმედება მონაცემთა ცენტრალიზებულ ან ლოკალურ ბაზებთან, როდესაც ისინი მუშაობენ კლიენტის მანქანებზე.

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



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

C# ენა .NET გარემოში მონაცემთა ბაზებთან (DBS) სამუშაოდ გვთავაზობს შემდეგ კომპონენტებს (ნახ.3.2), რომელთა საფუძველია ADO.NET.



ნახ.3.2

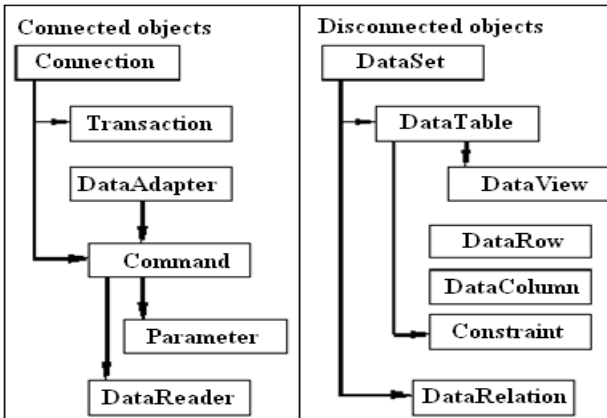
მონაცემებთან მიმართვის ტრადიციული ტექნოლოგიები, ჩვეულებრივ, ახორციელებდა მონაცემების წვდომას წყაროსთან მუდმივი მიერთების გზით. ასეთი მოდელის გამოყენებისას პროგრამული აპლიკაცია გახსნის მონაცემთა ბაზასთან მიერთებას და არ დახურავს მუშაობის დამთავრებამდე. დანართის სირთულის ზრდასთან ერთად იზრდება მონაცემთა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

➤ ADO.NET დრაივერის არქიტექტურა

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



ნახ.3.3

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 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 Access, SQL Server ბაზებიდან 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: DataColumn, რომელშიც თავსდება ცხრილის სვეტები და ცხრილის შეზღუდვათა ერთობლიობა. ეს ორი ერთობლიობა კმნის ცხრილის სქემას.

DataTable ობიექტი შეიცავს აგრეთვე DataRow 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. იგი 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 SqlDataReader ობიექტს (მაგალითად, SELECT);

- SqlDataReader 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 ახდენს აპლიკაციის მიერთებას მონაცემთა ბაზასთან, 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 SqlDataReader 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:

- 1) SqlConnection (MsSQL_Server-თან შესაერთებლად) და
- 2) OleDbConnection (სხვა ტიპის მონაცემთა ბაზებთან დასაკავშირებლად).

მონაცემთა ბაზასთან კავშირის არხის გასახსნელი აუცილებელი მონაცემები ინახება Connection ობიექტის თვისებაში ConnectionString. 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 SqlDataReader ობიექტის საშუალებით.

- **ობიექტი** SqlDataReader გვთავაზობს ნაკადს მონაცემთა ბაზის ჩანაწერების ერთობლიობით, ოღონდ მხოლოდ ერთი მიმართულებით წასაკითხად. მონაცემთა პროვაიდერის სხვა კომპონენტებისგან განსხვავებით SqlDataReader-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 SqlDataReader ობიექტის შემცველი მონაცემების ჩაწერა დისკზე არაა საჭირო, მაშინ ეს სტრიქონები შეიძლება პირდა-

პირ 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 ურთიერთქმედებისთვის.

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.

3.2. C# ენისა და SQL Server ბაზის ერთობლივი გამოყენება ADO.NET დრაივერით და DataGridView კლასით

მიზანი: ვიზუალური დაპროგრამების C# ენის, Ms SQL Server მონაცემთა ბაზების პაკეტის და ADO.NET დრაივერის ერთობლივი გამოყენებით პროგრამული აპლიკაციების აგების შესწავლა.

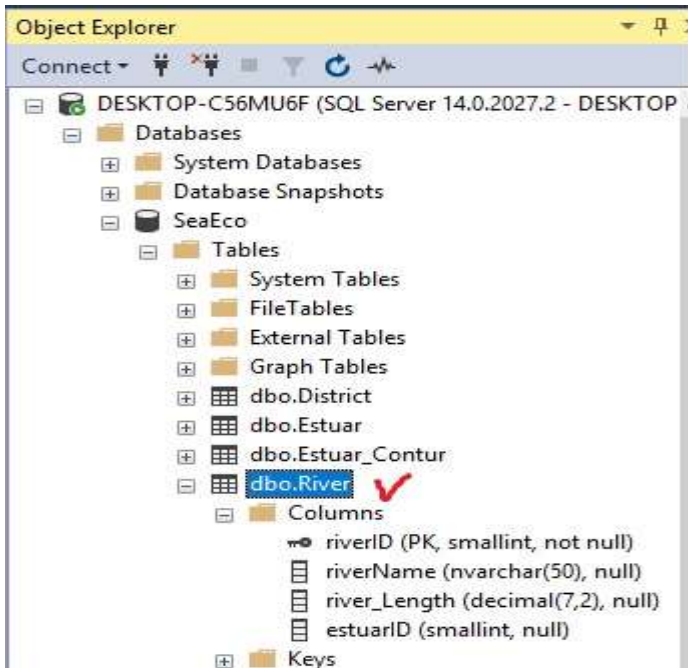
თავის შესავალში ჩვენ მოკლედ შევხებით ADO.NET-ის დანიშნულებას და მის ზოგიერთ ობიექტს, რომელთა საშუალებითაც ხორციელდება კავშირი C# ენაზე დაწერილ ინტერფეისსა და მონაცემთა ბაზას შორის.

ამოცანა_1: მოცემულია Ms SQL Server მონაცემთა ბაზა (მაგალითად, შავი ზღვის ეკოლოგიური მონიტორინგის სისტემა [11]), რომლის ერთ-ერთი ცხრილი (Table) არის River.dbo (მდინარეები). საჭიროა ავაგოთ C# პროექტი (მომხმარებლის ინტერფეისი), რომელიც მონაცემთა ბაზიდან ამოიღებს მდინარეების მონაცემებს DataGridView ცხრილში, შეძლებს Insert, Update და Delete ოპერაციების განხორციელებას. გამოყენებულ უნდა იქნას ADO.NET დრაივერის საშუალებები.

3.2.1. ექსპერიმენტული მონაცემთა ბაზის მომზადება SQL Server პაკეტით

3.4 ნახაზზე ნაჩვენებია საწყისი მონაცემთა ბაზა, რომელიც Ms SQL Server პაკეტითაა რეალიზებული. (ა) შეესაბამება ბაზის ცხრილების იერარქიასა და აქვე ჩანს River ცხრილის სტრუქტურაც (ბ), მონაცემთა შესაბამისი ტიპებით. (გ)-ზე მოცემულია ჩანაწერები, რომელთა შეტანა მოხდა წინასწარ (თუმცა ჩვენი პროგრამისთვის არაა აუცილებელი მისი არსებობა. შესაძლებელია იგი შეივსოს ინტერფეისიდან).

პროგრამული ინჟინერის საფუძვლები



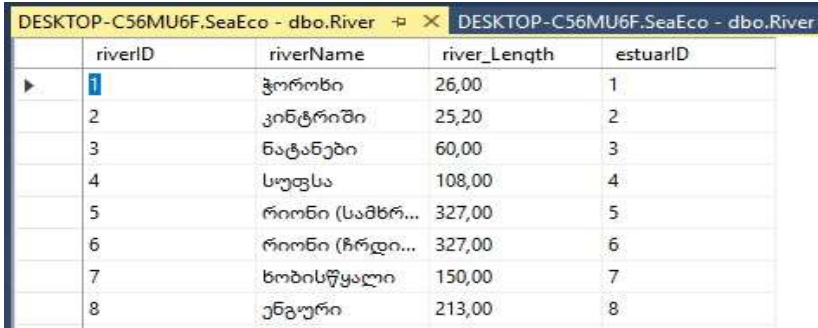
ნახ.3.4-ა. მონაცემთა ბაზა Ms SQL Server-ში

The screenshot shows the 'Structure' tab for the 'dbo.River' table. The table has the following columns:

Column Name	Data Type	Allow Nulls
riverID	smallint	<input type="checkbox"/>
riverName	nvarchar(50)	<input checked="" type="checkbox"/>
river_Length	decimal(7, 2)	<input checked="" type="checkbox"/>
estuarID	smallint	<input checked="" type="checkbox"/>

ნახ.3.4-ბ. River (მდინარეების) ცხრილის სტრუქტურა Ms SQL Server-ში

პროგრამული ინჟინერის საფუძვლები

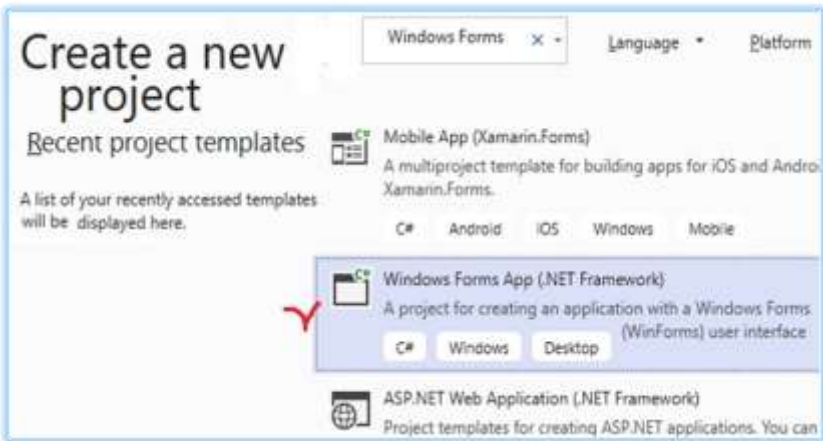


riverID	riverName	river_Length	estuarID
1	ჭორონი	26,00	1
2	კინტრიში	25,20	2
3	ნატანები	60,00	3
4	სუფსა	108,00	4
5	რიონი (სამხრ...	327,00	5
6	რიონი (ჩრდი...	327,00	6
7	სობისწყალი	150,00	7
8	ენგური	213,00	8

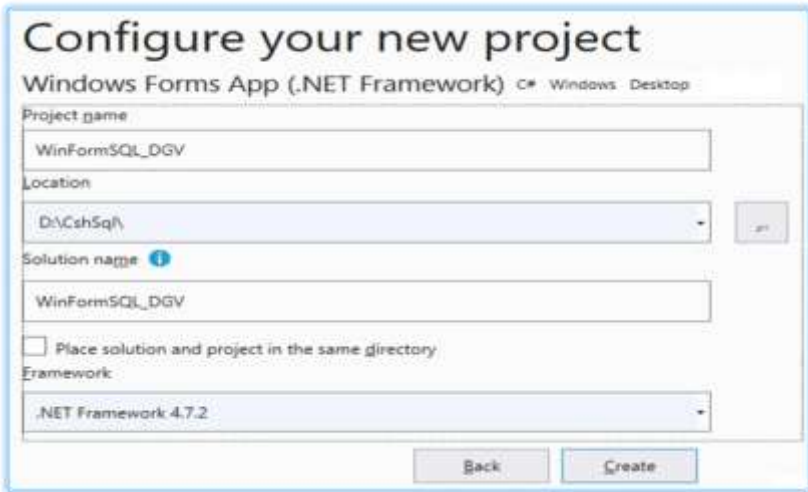
ნახ.3.4-გ. River (მდინარეების) საწყისი ცხრილი
Ms SQL Server-ში

3.2.2. ახალი პროექტის შექმნა .NET პლატფორმაზე C# ენის გამოყენებით

დავიწყოთ ახალი პროექტის აგება Ms Visual Studio.NET სამუშაო გარემოში. 3.5 ნახაზზე ნაჩვენებია პროექტის შექმნის პროცედურა.



ნახ.3.5-ა. WinFormSQL_DGV პროექტის შექმნა

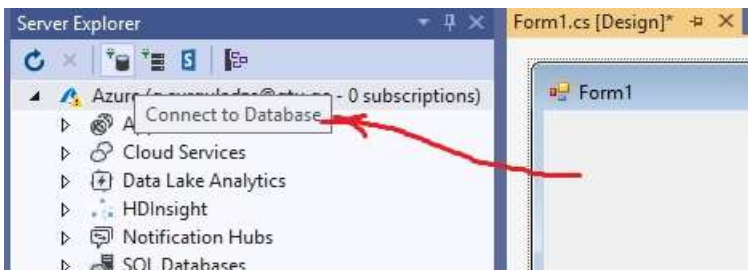


ნახ.3.5-ბ. WinFormSQL_DGV პროექტის შექმნა და განთავსება სისტემის კატალოგში

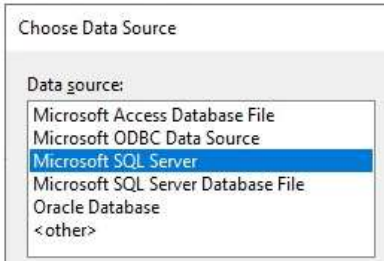
ვირჩევთ პროექტის სახელს (Name), მისი შენახვის ადგილს (Browse-ს დახმარებით) და Solution name-ს. შემდეგ OK.

3.2.3. პროექტთან მონაცემთა ბაზის მიერთება

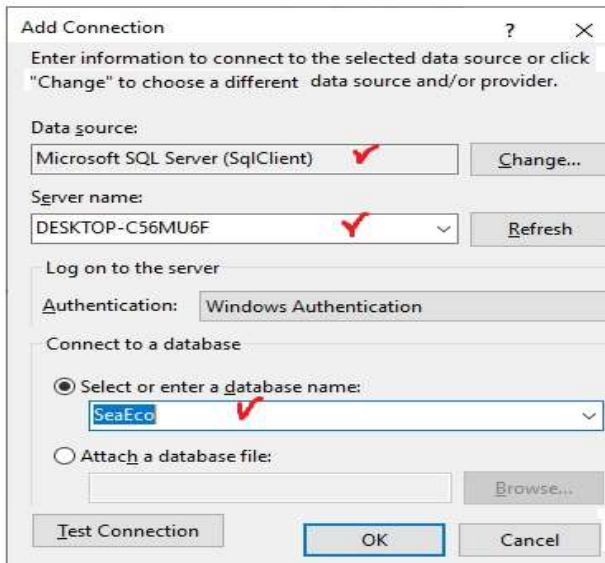
საჭიროა პროექტისთვის განვსახორციელოთ მონაცემა ბაზის მიერთება (Connect to Database), რაც 3.6 ნახაზზეა ნაჩვენები.



ნახ.3.6. Connect Database ამოქმედება



გამოიტანება ახალი ფანჯარა (ნახ.3.7), რომელშიც უნდა შეირჩეს შესაბამისი წყარო (Data Source), სერვერი (Server name) და მონაცემთა ბაზა (Database name).

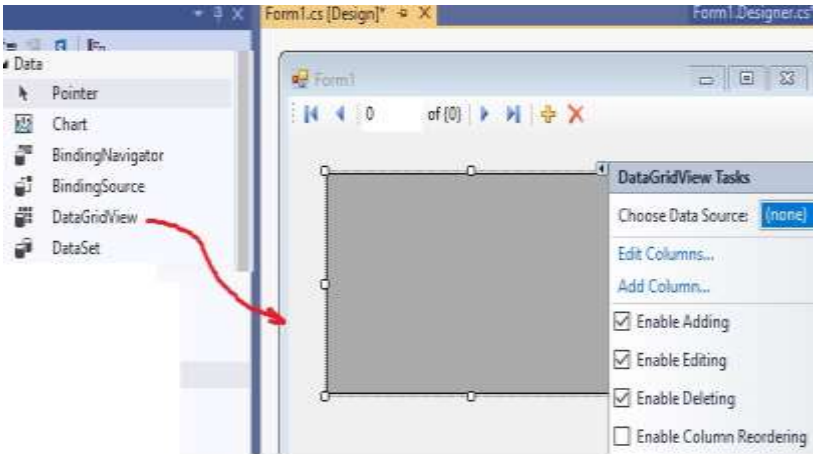


ნახ.3.7. მონაცემთა წყაროს, სერვერისა და ბაზის არჩევა

3.2.3. DataGridView ელემენტის გააქტიურება და ცხრილის პარამეტრების განსაზღვრა

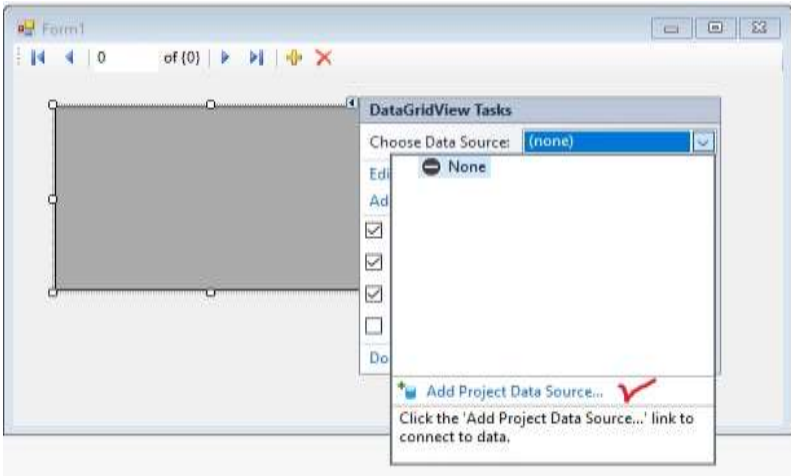
ინსტრუმენტების პანელიდან ფორმაზე გადავიტანოთ DataGridView ელემენტი და ზედა მარჯვენა კუთხე პატარა ისრით ავამოქმედოთ. მივიღებთ 3.8 ნახაზს.

პროგრამული ინჟინერის საფუძვლები



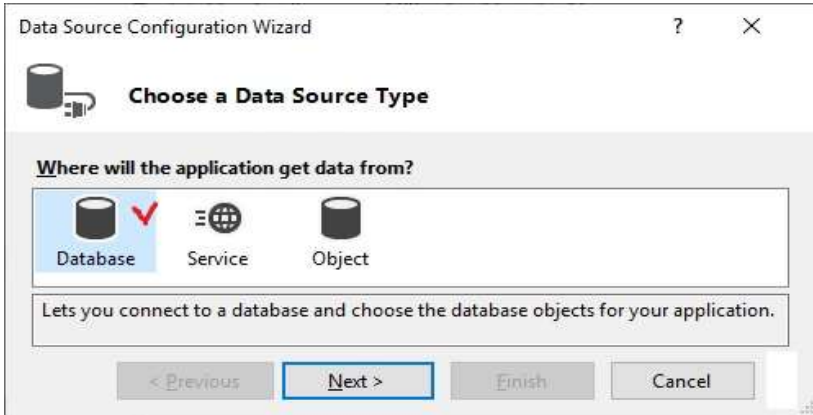
ნახ.3.8. პარამეტრების განსაზღვრა

ნახაზზე ჩანს, რომ ჩამატების, რედაქტირებისა და წაშლის ოპერაციები ნებადართულია (ჩეკბოქსები მონიშნულია). ავირჩიოთ Choose Data Source კომბობოქსის ღილაკი, მივიღებთ 3.9 ნახაზს.



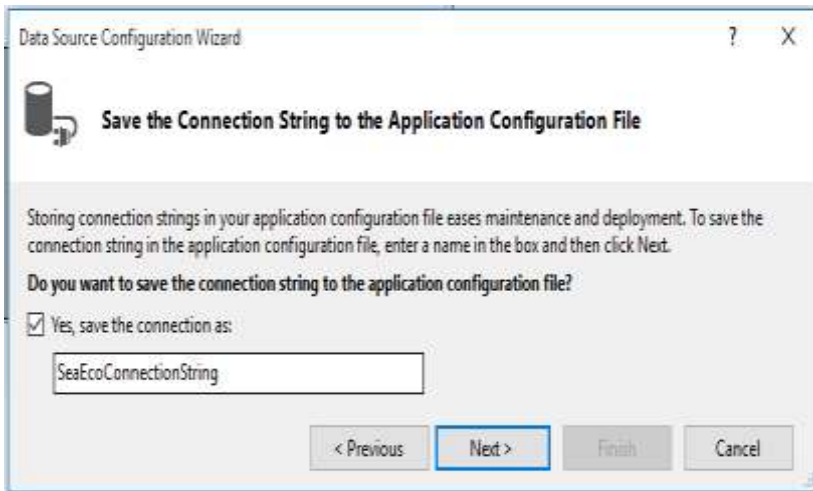
ნახ.3.9. მონაცემთა წყაროს პროექტის დამატება

ავამოქმედოთ Add Project Data Source და გადავიდეთ 3.10 ნახაზზე.



ნახ.3.10. მონაცემთა წყაროს ტიპის არჩევა

ვირჩევთ Database-სა და Next (ნახ.3.11).



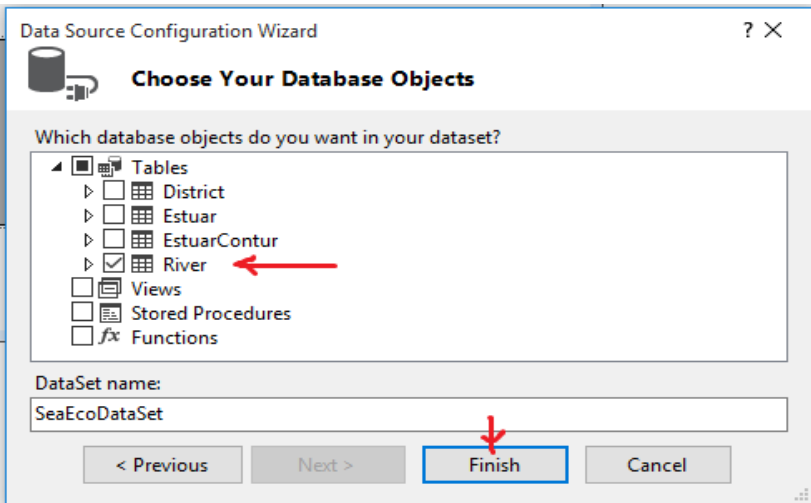
ნახ.3.11. მონაცემთა Connection (მიერთების) შერჩევა

Connection პარამეტრი ყველა კომპიუტერს ექნება თავისი. მისი განსაზღვრა შესაძლებელია Server Explorer-იდან (ჩვენ შემთხვევაში იგი არის: [DESKTOP-C56MU6F.SeaEco.dbo](#)). ბოლოს Next და გადავალთ 3.12 ნახაზზე.



ნახ.3.12. Connection String-ის შენახვა აპლიკაციის კონფიგურაციის ფაილში

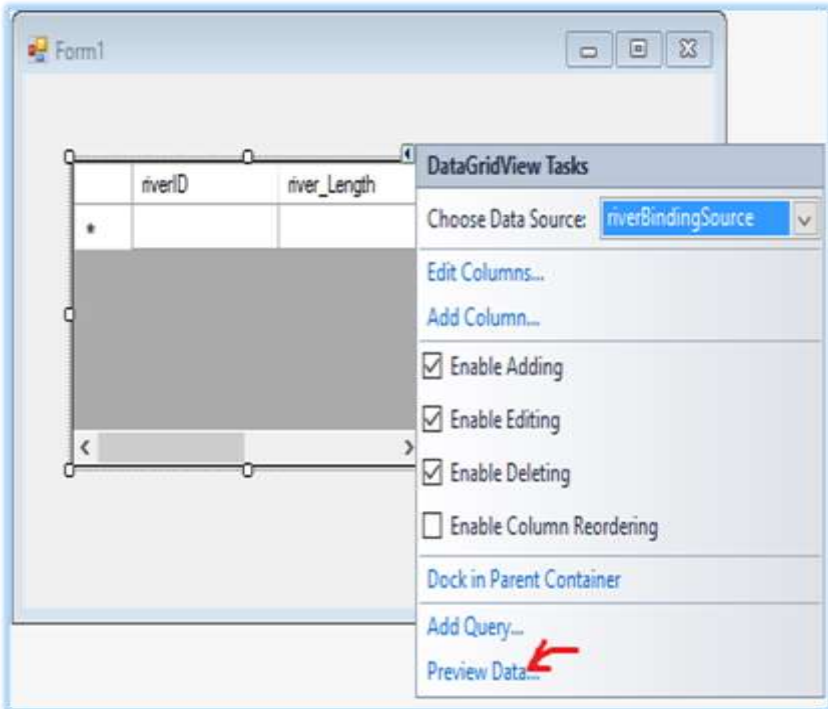
შემდეგ გამოიძახება მონაცემთა ბაზის ობიექტების არჩევის ფანჯარა (ნახ.3.13).



ნახ.3.13. ობიექტების მონიშვნა (მაგალითად, River)

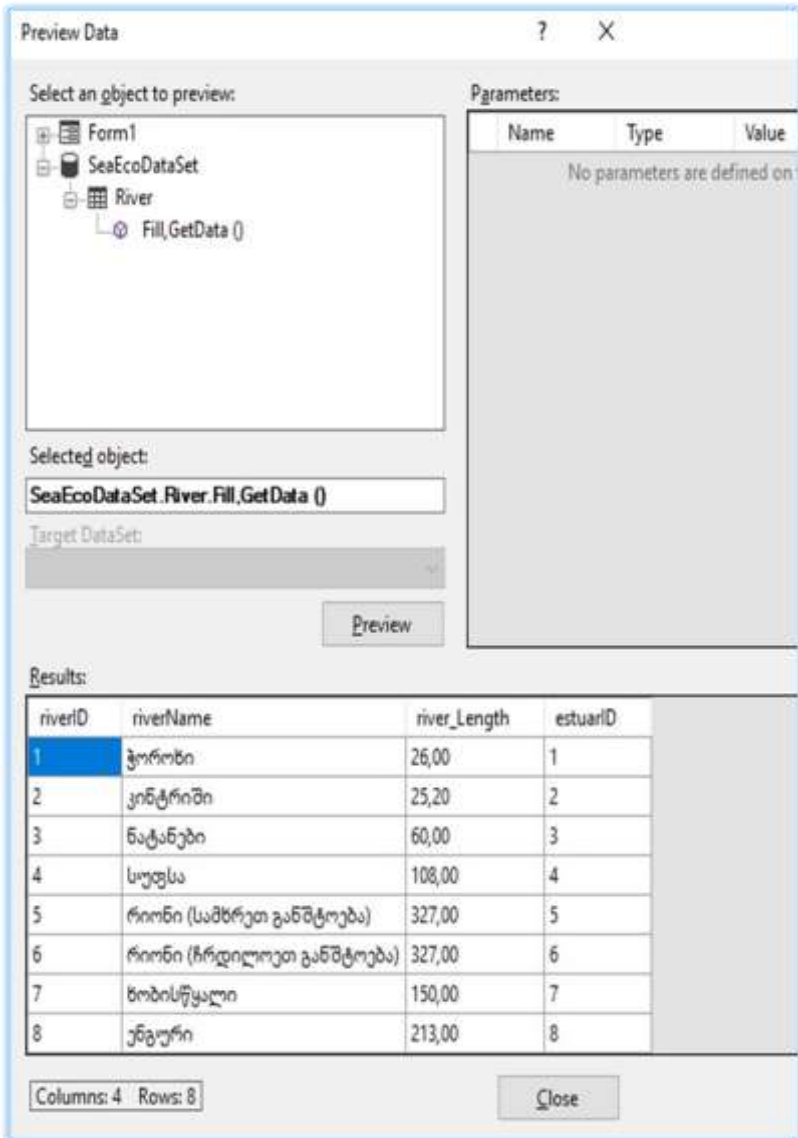
3.14 ნახაზზე ნაჩვენებია მონაცემთა წყაროს (Data Source) განსაზღვრის შედეგის მნიშვნელობა, ჩვენ შემთხვევაში იგი არის:

riverBindingSource.



ნახ.3.14. Data Source შედეგი: “riverBindingSource”

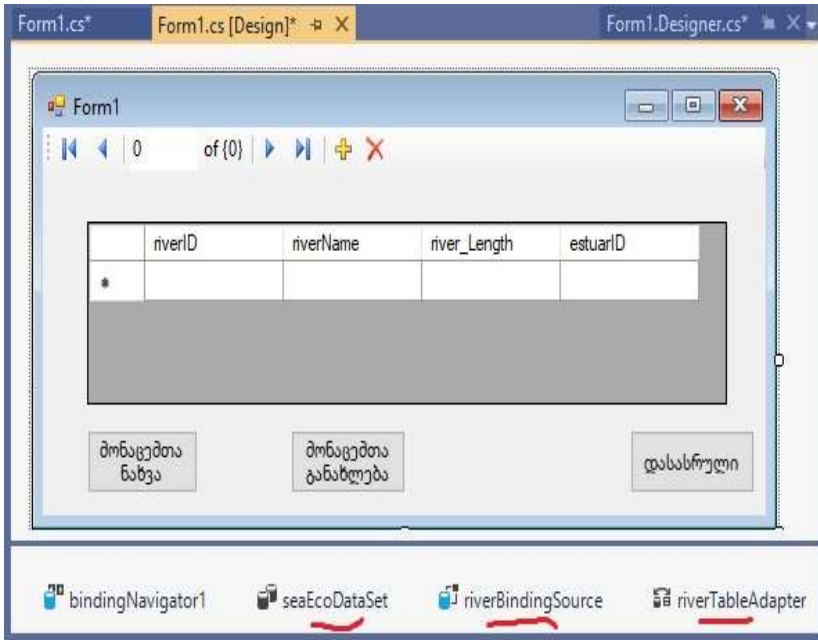
აქვე შეიძლება გამოვიყენოთ Preview Data ლინკი და დავათვალიეროთ წინასწარ მიერთებული ბაზის ცხრილის ჩანაწერები (ნახ.3.15).



ნახ.3.15. Preview Data ცხრილი

პროგრამული ინჟინერის საფუძვლები

ბოლოს, Form1-ს Properties-ში შევუცვალეთ სახელი „შავი ზღვის მდინარეები (საქართველოს საზღვრებში)“, ინსტრუმენტების პანელიდან გადმოვიტანეთ სამი ღილაკი (Button1,2,3), დავარქვათ სახელები. მიიღება 3.16 ნახაზზე მოცემული ფანჯარა.



ნახ.3.16. პროექტის ძირითადი ინტერფეისი

3.2.5. პროგრამული რეალიზაციის საკითხები

განვიხილოთ მომხმარებლის ინტერფეისის ღილაკების ფუნქციების დაპროგრამების საკითხი. უნდა განხორციელდეს SQL Server-ის შესაბამისი ბაზის ცხრილის ჩანაწერების დათვალიერება, ჩამატება, შეცვლა და წაშლა (ანუ CRUD - ოპერაციები).

თავდაპირველად ჩავამატოთ სახელსივრცის სტრიქონი:

```
using System.Data.SqlClient;
```

შემდეგ გამოვაცხადოთ გლობალური ცვლადები:

```
SqlDataAdapter sda;
```

```
SqlCommandBuilder scb;
```

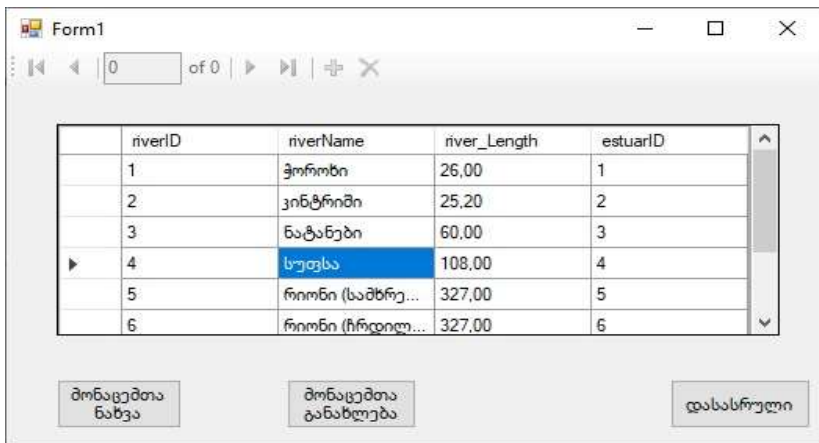
```
DataTable dt;
```

„მონაცემთა ნახვის“ ლილაკისათვის (button1) გვექნება შემდეგი კოდი:

```
private void Button1_Click(object sender, EventArgs e)
{
    SqlConnection con = new SqlConnection("Data Source=DESKTOP-
        C56MU6F; Initial Catalog = SeaEco; Integrated Security = True");
    sda = new SqlDataAdapter(@"SELECT riverID, river_Length,
        riverName, estuarID from River", con);
    dt = new DataTable();
    sda.Fill(dt);
    dataGridView1.DataSource = dt; }

```

პროგრამის ამუშავებით, თუ მასში არაა შეცდომები, მიიღება 3.17 ნახაზი.



ნახ.3.17. „მონაცემთა ნახვის“ (DataShow) ლილაკის ამოქმედებით მიღებული შედეგი

„მონაცემთა განახლების“ ლილაკის კოდი (Insert, Update, Delete) ნაჩვენებია ქვემოთ:

```
private void Button2_Click(object sender, EventArgs e)
{
    scb = new SqlCommandBuilder(sda);
    sda.Update(dt);
}
```

მთლიანი პროგრამის კოდი მოცემულია ლისტინგში.

```
// --- ლისტინგი --- Insert, Update, Delete for DataGridView_SQL----
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.SqlClient; // !!!

namespace WinFormSQL_DGV
{
    public partial class Form1 : Form
    {
        SqlDataAdapter sda;
        SqlCommandBuilder scb;
        DataTable dt;
        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
```

```

{
    // TODO: This line of code loads data into the
'seaEcoDataSet.River' table. You can move, or remove it, as needed.
    this.riverTableAdapter.Fill(this.seaEcoDataSet.River);

}
// მონაცემთა ნახვა -----
private void Button1_Click(object sender, EventArgs e)
{
    SqlConnection con = new SqlConnection("Data Source=DESK-
TOP-C56MU6F; Initial Catalog = SeaEco; Integrated Security = True");
    sda = new SqlDataAdapter(@"SELECT riverID,
        river_Length, riverName,
        estuarID from River", con);

    dt = new DataTable();
    sda.Fill(dt);
    dataGridView1.DataSource = dt;
}
// განახლება -----
private void Button2_Click(object sender, EventArgs e)
{
    scb = new SqlCommandBuilder(sda);
    sda.Update(dt);
}
private void Button3_Click(object sender, EventArgs e)
{
    Close();
}
}
}

```

3.18 ნახაზზე მოცემულია არსებულ ცხრილში ორი მნიშვნელობის (ესტუარისID) შეცვლა (ახალი რიცხვები ნაჩვენებია წრეში), შემდეგ 1-ელი და მე-2 ღილაკების ამოქმედებით ბაზაში

პროგრამული ინჟინერიის საფუძვლები

ჩაიწერება ეს შეცვლილი მნიშვნელობები (შეიძლება დაეხსნოს პროგრამა და თავიდან ავაშუშავოთ).

	მდინარის-ID	სიგრძე (კმ)	სახელი	ესტუარის-ID
	1	26,00	ჭოროზი (საქარ...	5
	2	25,20	კინტრიში	2
	3	60,00	ნატანები	3
	4	108,00	სუფსა	4
▶	5	327,00	რიონი (სამხრე...	1
	6	327,00	რიონი (ჩრდი...	6
	7	150,00	ზობისწყალი	7
	8	213,00	ენგური	8
*				

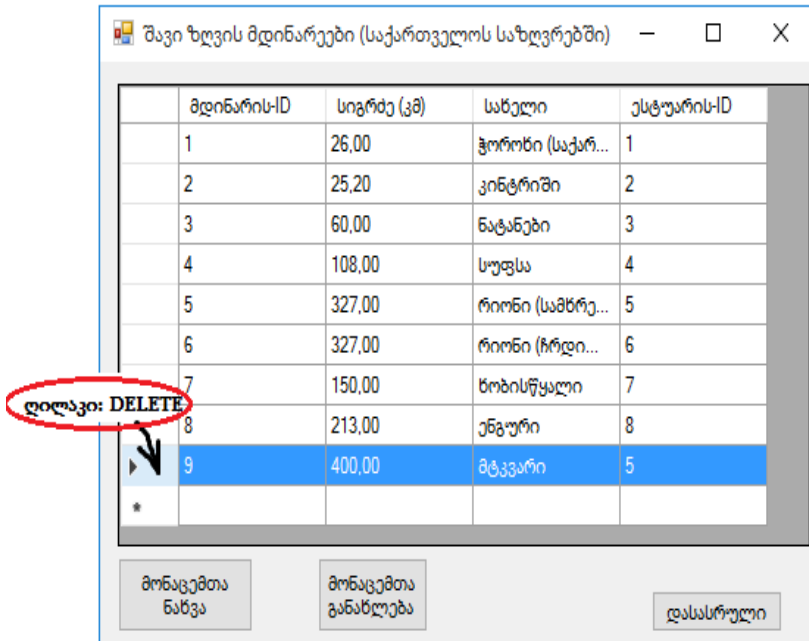
ნახ.3.18. Update ცვილების განხორციელება

	მდინარის-ID	სიგრძე (კმ)	სახელი	ესტუარის-ID
▶	1	26,00	ჭოროზი (საქარ...	1
	2	25,20	კინტრიში	2
	3	60,00	ნატანები	3
	4	108,00	სუფსა	4
	5	327,00	რიონი (სამხრე...	5
	6	327,00	რიონი (ჩრდი...	6
	7	150,00	ზობისწყალი	7
	8	213,00	ენგური	8
	9	400,00	მტკვარი	5
*				

ნახ.3.19. Insert ოპერაციის განხორციელება

პროგრამული ინჟინერიის საფუძვლები

ბოლოს ნაჩვენებია სტრიქონის წაშლის (Delete) ოპერაცია. იგი ხორციელდება მაგალითად, „მდინარის-ID“ შესაბამისი სტრიქონის მონიშვნით და შემდეგ კომპიუტერის კლავიატურის Delete- ღილაკის ამოქმედებით (ნახ.3.20).



ნახ.3.20. Delete ოპერაციის განხორციელება

ამით დავასრულეთ ვიზუალური C# ელემენტების საფუძველზე მონაცემთა ბაზასთან დაკავშირების რეალიზაციის ამოცანის განხილვა, რომლის შედეგადაც აგებულ იქნა შავი ზღვის ეკომონიტორინგის ინფორმაციული სისტემის ფრაგმენტი.

3.3. ADO.NET-ის DataSet-ობიექტის გამოყენება მონაცემთა ბაზასთან სამუშაოდ

განცალკევებული DataSet გამოიყენება ბაზის მონაცემებთან წვდომისათვის უწყვეტი კავშირის გარეშე და იძლევა მონაცემთა მოდიფიცირების შესაძლებლობას მხოლოდ კლიენტის მხარეს.

განცალკევებული (არადაკავშირებული) DataSet ნიშნავს, რომ მონაცემთა წყაროსთან მონაცემების წვდომისთვის არ არის საჭირო უწყვეტი კავშირი. ამ დროს, მონაცემთა ნაკრების კლასი მონაცემებს ინახავს კლიენტის მანქანაში.

DataSet კლასი შედის "System.Data" სახელთა სივრცეში და იგი შეიძლება შეიცავდეს მრავალ ცხრილს (Table), ხოლო „DataAdapter“ კლასი Disconnected Architecture-ში გამოიყენება მონაცემთა წყაროსა და მონაცემთა ნაკრებს შორის კომუნიკაციისთვის.

DataAdapter-ის გამოყენების ორი მეთოდი არსებობს:

- **Fill**(DataSet ds, string TableName);
- **Update**(DataSet ds, string TableName).

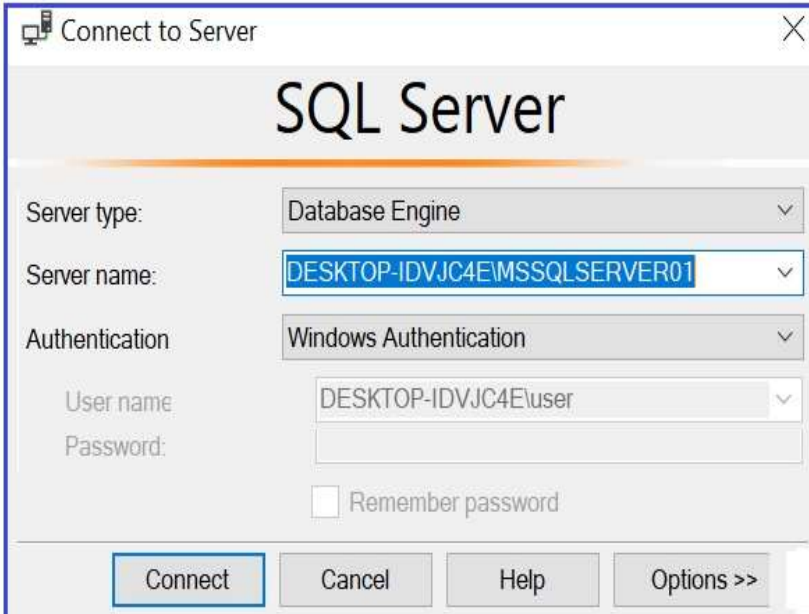
DataSet არის ცხრილების (Table) ნაკრები, სადაც თითოეული ცხრილი წარმოდგენილია DataTable კლასის სახით, ხოლო ყოველი DataTable შედგება სტრიქონებისა (Row) და სვეტების (Column) ნაკრებისგან. თითოეული სტრიქონი (ველი) წარმოდგენილია DataRow კლასის, ხოლო თითოეული სვეტი DataColumn კლასის სახით. ორივე მათგანი განისაზღვრება ინდექსის პოზიციით ან სახელით.

განვიხილოთ ამოცენები Dataset-ის მაგალითზე.

Ms SQL Server Management Studio-ს საშუალებით შევქმნათ მონაცემთა წყაროდ (DataSource) მარტივი მონაცემთა ბაზა, ხოლო აპლიკაცია შევასრულოთ visual studio 2019 ან 2022 ვერსიაში.

შექმნათ ცხრილი (Table):

- გავხსნათ Microsoft SQL Server Management Studio, გამოჩნდება დიალოგური ფანჯარა (ნახ.3.21).



ნახ.3.21

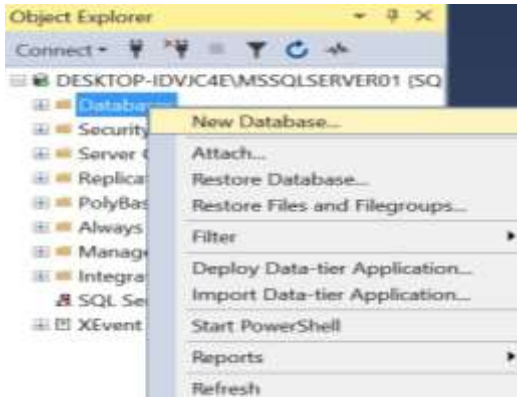
- დასაკავშირებლად ავამოქმედოთ Connect ღილაკი. სერვერის სახელი (Server name) გვჭირდება მონაცემთა ბაზაში (Database) არსებულ ცხრილთან (Table) დასაკავშირებლად.

შენიშვნა:

თუ უკვე გვაქვს მონაცემთა ბაზაში ცხრილი შექმნილი, მაშინ ეს საფეხური შეგვიძლია გამოვტოვოთ.

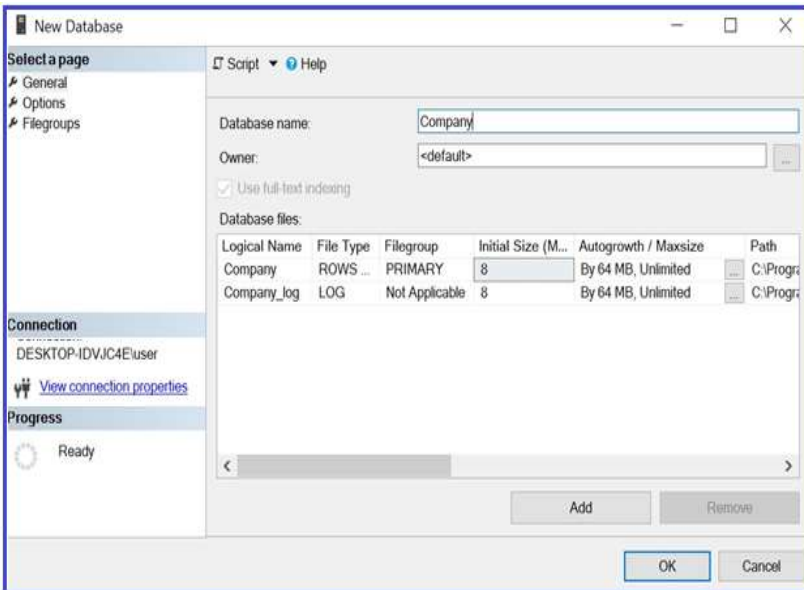
object explorer-ში მარჯვენა ღილაკით ავირჩიოთ Databases - > New Database (ნახ.3.22).

პროგრამული ინჟინერის საფუძვლები



ნახ. 3.22

მონაცემთა ბაზის სახელში ჩავწეროთ Company და დავაჭიროთ ღილაკს"Ok".

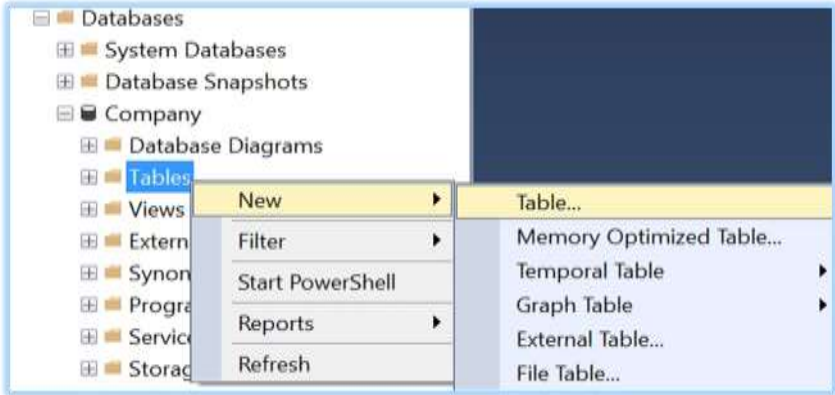


ნახ. 3.23

პროგრამული ინჟინერის საფუძვლები

Company მონაცემთა ბაზაში შევექმნათ Tanamshromeli1 ცხრილი. Databases-ზე მაუსის მარჯვენა ღილაკი და ვირჩევთ:

Tables -> New -> Table.



ნახ. 3.24

შევიტანოთ მონაცემები და ცხრილი შევიანახოთ სახელით Tanamshromeli1.

A screenshot of the SQL Server Enterprise Manager interface showing the table structure for 'Tanamshromeli1'. The table has the following columns:

Column Name	Data Type	Allow Nulls
Tanamsh_id	varchar(5)	<input type="checkbox"/>
Tan_Name	char(30)	<input type="checkbox"/>
Tar	date	<input checked="" type="checkbox"/>
Asaki	int	<input checked="" type="checkbox"/>
City	char(10)	<input type="checkbox"/>
Mob	varchar(20)	<input checked="" type="checkbox"/>
Xelf	int	<input checked="" type="checkbox"/>

ნახ. 3.25

პროგრამული ინჟინერიის საფუძვლები

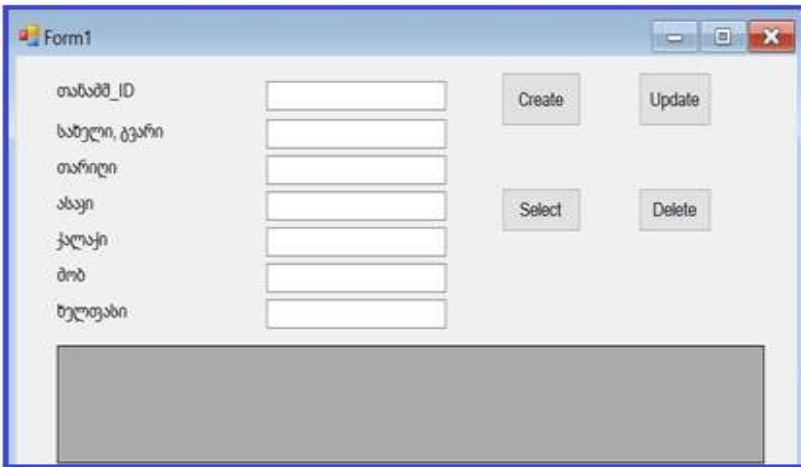
მარჯვენა ღილაკის საშუალებით გავხსნათ მონაცემების შეტანის ფანჯარა "Edit Top 200 Rows" და დავამატოთ რამდენიმე ჩანაწერი (3.26).



	Tanamsh...	Tan_Name	Tar	Asaki	City	Mob	Xelf
	E01	giorgi ty...	2022-01-...	25	batumi	7677876...	3000
	E02	maia shel...	2019-03-...	32	tbilisi	2343423...	5000
	E03	otar gulu...	2017-01-...	40	tbilisi	4545467...	6000
	E04	nino arje...	2018-04-...	36	batumi	5557676...	3000
	E05	nikoloz s...	2020-11-...	40	tbilisi	8886765...	4000
▶*	NULL	NULL	NULL	NULL	NULL	NULL	NULL

ნახ. 3.26

შევქმნათ აპლიკაცია განცალკევებული Dataset-ისთვის: გავხსნათ Visual Studio და შევქმნათ პროექტი (Windows Forms App .NET Framework). Windows ფორმაზე განვათავსოთ კომპონენტები (ნახ.3.27).



Form1

თანამშ_ID

სახელი, გვარი

თარიღი

ასაკი

ქალაქი

მობ

ხელფასი

Create Update

Select Delete

ნახ.3.27

დავამატოთ CRUD (Create, Read, Update and Delete) ფუნქციების ღილაკები.

```
// -- Form1.cs-ლისტინგი -----
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.Data.SqlClient;

namespace TanamshDataSet
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
            Sqlda = new SqlDataAdapter("Select * from Tanamshromeli",
ConnectionStrings);
            dataSet = new DataSet();
            Sqlda.Fill(dataSet, "Emp");
        }
        public string ConnectionString
        {
            get;
            set;
        }
    }
}
```



```
    } = @"Data Source=DESKTOP-IDVJC4E\MSSQLSERVER01;Initial  
Catalog=Company;Integrated Security=True;";  
    SqlDataAdapter sqlda;  
    DataSet dataSet;  
}  
}
```

1) კოდში თავდაპირველად ვამატებთ System.Data და System.Data.SqlClient სახელსივრცეებს, რათა შევძლოთ DataSet კლასის გამოყენება.

2) ვქმნით თვისებას ConnectionString, რომელიც მოიცავს სამ მნიშვნელობას:

- Data Source - სერვერის დასახელება;
- Initial Catalog - მონაცემთა ბაზის დასახელება;
- Integrated Security;

3) ვაცხადებთ SqlDataAdapter და DataSet კლასების ორ ობიექტს, Form1() {...} -ში ინიცირებით.

4) Then we declare Two objects of SqlDataAdapter and DataSet class, Initialize them inside Form1() {...}. მეთოდს

5) ვიყენებთ of DataSet Fill() მეთოდს Tanamshromeli1 ცხრილის მისაღებად. ჩვენ შეგვიძლია გამოვიყენოთ მარტივი კოდი of DataSet Fill(), როდესაც მხოლოდ ერთი ცხრილი (Table) გვაქვს. იმ შემთხვევაში თუ DataSet შედგება რამდენიმე ცხრილისგან, მაშინ ვსაზღვრავთ ცხრილის სახელს (მაგალითად, "Emp") ან ვიყენებთ ინდექსს თითოეული ცხრილისთვის ([0], [1], ...), რაც კარგ პრაქტიკად ითვლება.

//-- Create დილაკის ლისტინგი:

```
private void button1_Click(object sender, EventArgs e)
```

```
{
    DataRow dr = dataSet.Tables["Emp"].NewRow();
    dr[0] = Convert.ToInt32(textBox1.Text);
    dr[1] = textBox2.Text;
    dr[2] = textBox3.Text;
    dr[3] = Convert.ToInt32(textBox4.Text);
    dr[4] = textBox5.Text;
    dr[5] = textBox6.Text;
    dr[6] = Convert.ToInt32(textBox7.Text);

    dataSet.Tables["Emp"].Rows.Add(dr);
    MessageBox.Show("შეიქმნა ჩანაწერი..");
    Sqlda.Fill(dataSet);
}
```

- ვქმნით DataRow კლასის "dr" ობიექტს DataSet-ის Tanamshromeli1 ცხრილში ველის დასამატებლად;
- "dr" ობიექტის შექმნის შემდეგ ვაკავშირებთ TextBox-ების მნიშვნელობებს თითოეული ველის ინდექსთან;
- Add() მეთოდით ვამატებთ "dr" ობიექტს "Emp" ცხრილს.

```
//--- Select დილაკის ლისტინგი -----
private void button2_Click(object sender, EventArgs e)
{
    dataGridView1.DataSource = dataSet.Tables["Emp"];
}
```

ცხრილის (Table) მონაცემების გამოსატანად DataGridView-ს ვაკავშირებთ DataSet-ის "Emp" ცხრილთან, როგორც მონაცემთა წყაროსთან.

```
// ---- Update - ღილაკის ლისტინგი -----
private void button2_Click(object sender, EventArgs e)
{
    dataGridView1.DataSource = dataSet.Tables["Emp"];
}
private void button3_Click(object sender, EventArgs e)
{
    foreach (DataRow dr in dataSet.Tables["Emp"].Rows)
    { if (dr[0].ToString() == textBox1.Text)
        {
            try
            {
                dr[0] = Convert.ToInt32(textBox1.Text);
                dr[1] = textBox2.Text;
                dr[2] = textBox3.Text;
                dr[3] = Convert.ToInt16(textBox4.Text);
                dr[4] = textBox5.Text;
                dr[5] = textBox6.Text;
                dr[6] = Convert.ToInt32(textBox7.Text);
                MessageBox.Show("მონაცემები წარმატებით განახლდა  
ცხრილში. ბაზაში ჩაწერა ღილაკით – ბაზის განახლება..");
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }
            break;
        }
    }
}
```

```
// ---- Delete - დილაკის ლისტინგი -----
private void button4_Click(object sender, EventArgs e)
{
    foreach (DataRow dr in dataSet.Tables["Emp"].Rows)
    {
        if (dr[0].ToString() == textBox1.Text)
        {
            try
            {
                dataSet.Tables["Emp"].Rows.Remove(dr);
                MessageBox.Show("მონაცემი წარმატებით წაიშალა..");
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }
            break;
        }
    }
}
```

ამოცანა: განვიხილოთ აპლიკაცია, როდესაც DataSet-ის მონაცემებს არ ვიღებთ მონაცემთა ბაზიდან და თავად ვქმნით DataSet-ის ცხრილს (Table), სვეტებს და ვავსებთ ველებს მონაცემებით.

```
// ---- DataSet - დილაკის ლისტინგი ----
using System;
using System.Collections.Generic;
```

```
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
```

```
namespace WindowsFormsApp2
```

```
{
```

```
    public partial class Form1 : Form
```

```
    {
```

```
        public Form1()
```

```
        {
```

```
            InitializeComponent();
```

```
        }
```

```
        private void button1_Click(object sender, EventArgs e)
```

```
        {
```

```
            DataTable table1 = new DataTable();
```

```
            DataTable table2 = new DataTable();
```

```
            DataColumn dc11 = new DataColumn("ID", typeof(Int32));
```

```
            DataColumn dc12 = new DataColumn("Name", typeof(string));
```

```
            DataColumn dc13 = new DataColumn("City", typeof(string));
```

```
            table1.Columns.Add(dc11);
```

```
            table1.Columns.Add(dc12);
```

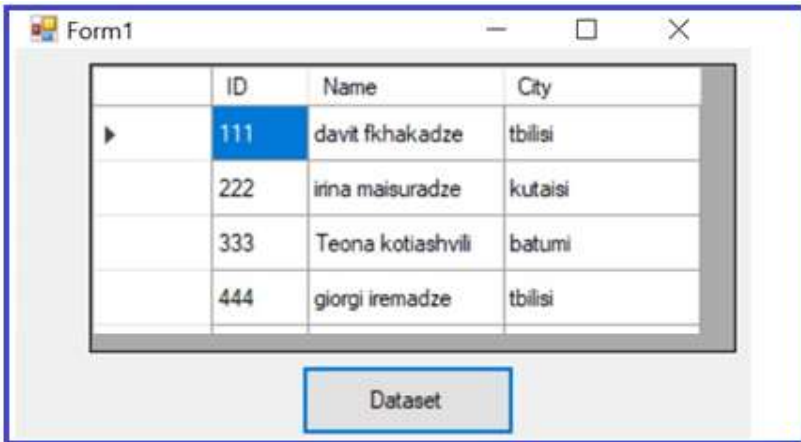
```
            table1.Columns.Add(dc13);
```

პროგრამული ინჟინერის საფუძვლები

```
table1.Rows.Add(111, "davit fkhakadze", "tbilisi");  
table1.Rows.Add(222, "irina maisuradze", "kutaisi");  
table1.Rows.Add(333, "Teona kotiaashvili", "batumi");  
table1.Rows.Add(444, "giorgi iremadze", "tbilisi");
```

```
DataSet dset = new DataSet();  
dset.Tables.Add(table1);  
dataGridView1.DataSource = dset.Tables[0];
```

```
}  
}  
}
```



ნახ.3.28

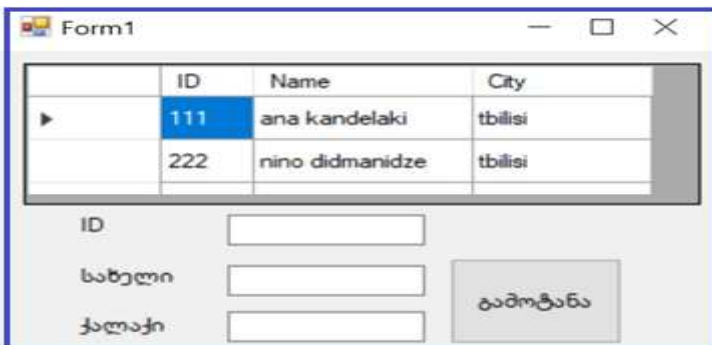
როგორც ვიცით, DataSet ობიექტი მონაცემებს ინახავს სვეტებისა და სტრიქონების ფორმატში, ამიტომ მონაცემების გამოსატანად ვიყენეთ მათ ინდექსებს.

„გამოტანა“ ღილაკით DataSet-დან პირველი ცხრილის მონაცემები გამოვიტანოთ TextBox-ებში, ხოლო DataGridView-ში გამოჩნდეს ცხრილის ყველა ჩანაწერი ფორმის ჩატვირთვისთანავე.

// ---- „გამოტანა“ ღილაკის ლისტინგი -----

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Xml.Linq;
using System.Data.SqlClient;
namespace datasetproject3
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
        DataTable EMPLOYEE;
        DataSet dset;
        private void Form1_Load(object sender, EventArgs e)
        {
            EMPLOYEE = new DataTable();
            DataColumn dc11 = new DataColumn("ID", typeof(Int32));
```

```
DataColumn dc12 = new DataColumn("Name", typeof(string));
DataColumn dc13 = new DataColumn("City", typeof(string));
EMPLOYEE.Columns.Add(dc11);
EMPLOYEE.Columns.Add(dc12);
EMPLOYEE.Columns.Add(dc13);
EMPLOYEE.Rows.Add(111, "ana kandelaki", "tbilisi");
EMPLOYEE.Rows.Add(222, "nino didmanidze", "tbilisi");
dset = new DataSet();
dset.Tables.Add(EMPLOYEE);
dataGridView1.DataSource = dset.Tables[0];
}
private void button1_Click(object sender, EventArgs e)
{
    textBox1.Text = dset.Tables[0].Rows[0][0].ToString();
    textBox2.Text = dset.Tables[0].Rows[0][1].ToString();
    textBox3.Text = dset.Tables[0].Rows[0][2].ToString();
}
}
```



ნახ.3.29

3.4. პროგრამის რეფაქტორინგი და რეინჟინერინგი

პროგრამულ ინჟინერიაში (ტექნოლოგიაში) ხშირად გამოიყენება ტერმინები „რეფაქტორინგი“ (Refactoring) და „რეინჟინერინგი“ (Re-engineering) [12].

„რეფაქტორინგი“ არის არსებული პროგრამული კოდის სისტემატური მოდიფიკაცია და სრულყოფა, მისი ფუნქციონირების სემანტიკის ძირფესვიანი *ცვლილების გარეშე*. კოდში ცვლილებები ხორციელდება ავტომატიზებული გარდაქმნებით, რომლებსაც .NET სამუშაო გარემო გვთავაზობს.

„რეინჟინერინგი“ კი გულისხმობს ფუნდამენტური ცვლილებების შეტანას პროგრამის სტრუქტურასა და დიზაინში – იქნება ეს პროგრამული უზრუნველყოფის ადაპტაცია ახალ აპარატურულ პლატფორმაზე, მისი პროგრამირების ენის შეცვლა თუ ახალ დიალექტზე გადატანა.

რეფაქტორინგის ტიპური მაგალითია – *მეთოდის სახელის შეცვლა*. ამოცანა მდგომარეობს რომელიმე კონკრეტული მეთოდისათვის *დასახელებისა* და მისი *განსაზღვრების* ცვლილების განხორციელებაში, ამასთანავე *მისი გამოყენების ყველა ადგილზე!*

თუ პროექტი საკმაოდ დიდია, მაშინ **ხელით** ასეთი ცვლილებების ჩატარების ამოცანის გადაწყვეტა საკმაოდ შრომატევადი და მოუხერხებელია. არაა გამორიცხული შემთხვევა, რომ რომელიმე მოდულში დაგვავიწყდეს ამ ცვლილების განხორციელება, რაც შემდგომში, პროგრამის ამუშავებისას, აუცილებლად გამოჩნდება შეცდომის სახით და მოგვიწევს პროექტის ხელახალი შესწორება.

არსებობს პროგრამული აპლიკაციის რეინჟინერინგის სამი ძირითადი ტიპის პრაქტიკა:

პროგრამული ინჟინერიის საფუძვლები

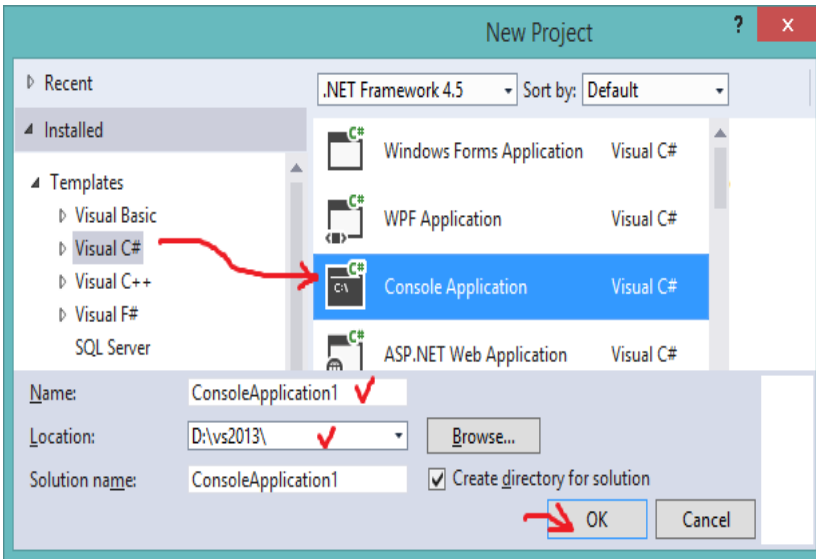
1. *პორტირება (Porting)* – როდესაც პროგრამა მორგებულია სხვა აპარატურაზე მუშაობისთვის;

2. *თარგმანი (Translation)* – როდესაც კოდი ითარგმნება ძველი (მემკვიდრეობითი) ენიდან ახალ (თანამედროვე) ენაზე;

3. *მიგრაცია (Migration)* – როდესაც კოდი გადადის ენის ახალ დიალექტზე მისი შინაგანი ბუნების შეცვლის გარეშე.

არსებითად, რეფაქტორინგი და რეინჟინერინგი მიზნად ისახავს არსებული კომპიუტერული კოდის გადაწერას პროგრამული უზრუნველყოფის შიდა დიზაინის, სტრუქტურის ან/და განხორციელების *გასაუმჯობესებლად*, მისი გარე ფუნქციონალობის შენარჩუნებით. უმეტესად ყოველი რეინჟინერინგი შედგება რეფაქტორინგული ინიციატივების ერთობლიობისგან.

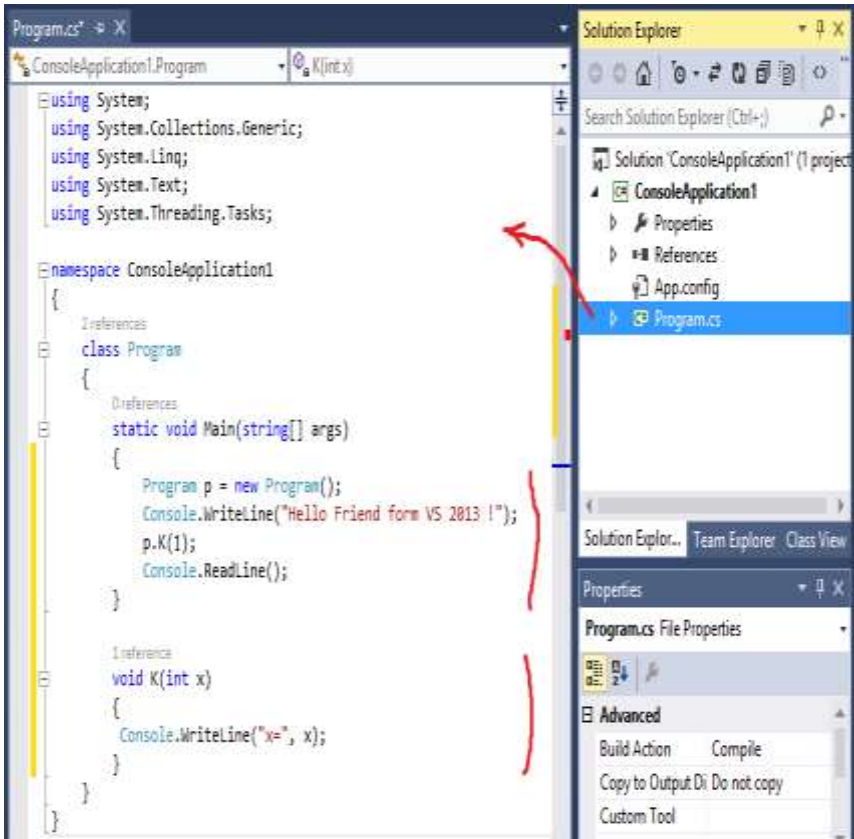
ამჯერად განვიხილოთ რეფაქტორინგის ამოცანა მარტივი კონსოლური აპლიკაციისათვის (ნახ.3.30).



ნახ.3.30. კონსოლური აპლიკაციის შექმნა

პროგრამული ინჟინერის საფუძვლები

მიღებული პროგრამის საწყის ტექსტს ჩავამატეთ რამდენიმე სტრიქონი, რომელსაც აქვს 3.31 ნახაზზე ნაჩვენები სახე.

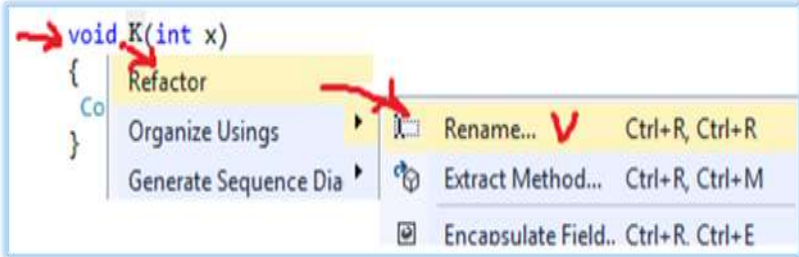


ნახ.3.31

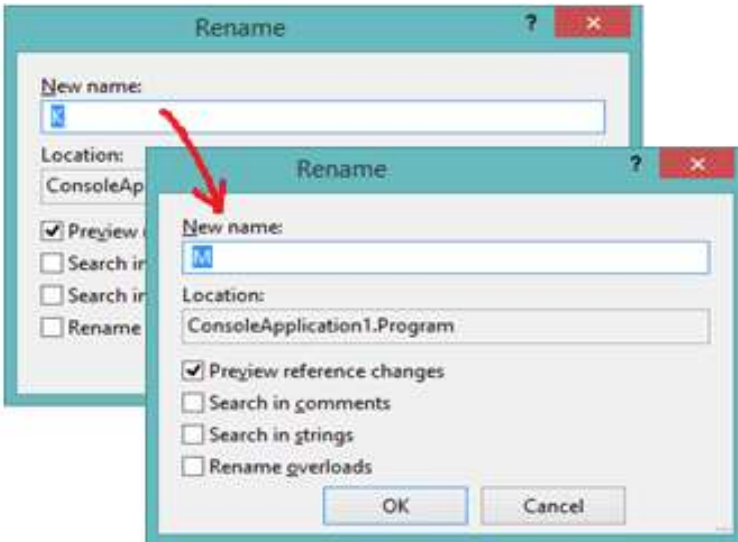
პროგრამაში განსაზღვრულია Program კლასი, სტატიკური მეთოდი Main და K- ეგზემპლარის მეთოდი x არგუმენტით.

დავუშვათ, რომ საჭიროა შეიცვალას K მეთოდის სახელი M-ით. ცვლილება უნდა განხორციელდეს არა მხოლოდ ამ მეთოდის განსაზღვრებაში, არამედ მისი გამოყენების ადგილებზეც!

კოდის რედაქტირების გარემოში მაუსის კურსორს ვაყენებთ K მეთოდის პირველ სტრიქონზე, მოვნიშნავთ შესაცვლელ სიტყვასა და კონტექსტური მენიუდან ვირჩევთ პუნქტს Refactor / Rename (ნახ.3.31).



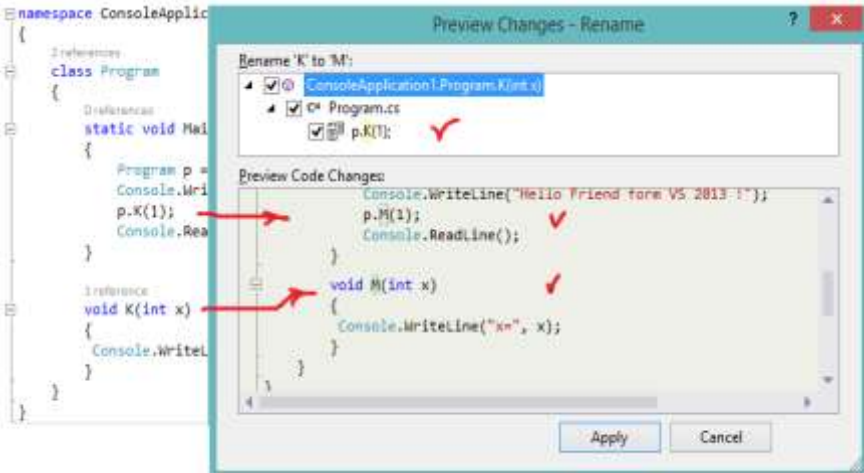
ნახ.3.31. K მეთოდისთვის რეფაქტორინგის ქმედების არჩევა



ნახ.3.32. K-ივლება M-ით

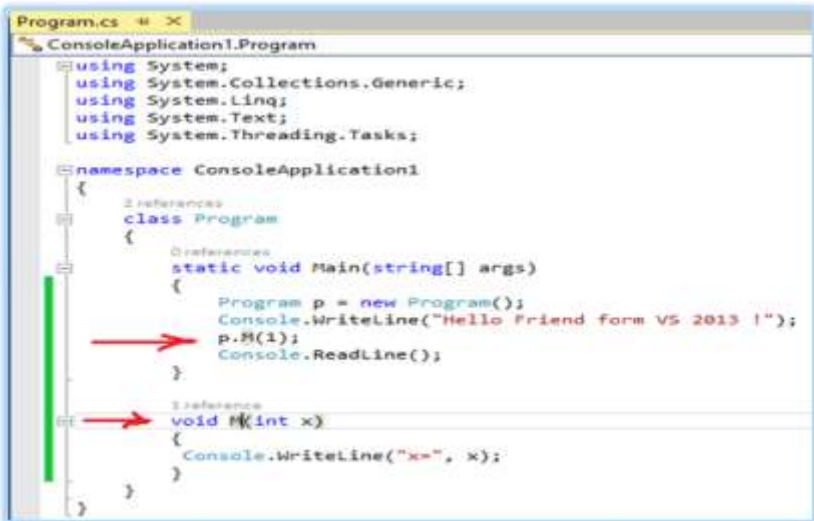
მიიღება 3.33 სურათი.

პროგრამული ინჟინერის საფუძვლები



ნახ.3.33. დაგეგმილი ცვლილებების წინასწარ ნახვა (Preview Changes)

თუ ყველაფერი წესრიგშია, მაშინ ვირჩევთ Apply-სა და ვიღებთ რეფაქტორინგის შედეგს (ნახ.3.34).



ნახ.3.34. რეფაქტორინგის შედეგი

VS.NET სამუშაო გარემოს აქვს რეფაქტორინგის შემდეგი შესაძლებლობები:

- Rename - სახელის შეცვლა;
- Extract Method - მეთოდის ამოღება: კოდის მონიშნული ფრაგმენტის მოდიფიკაცია მეთოდში მითითებული სახელით;
- Encapsulate Field - ველის ინკაფსულირება, მისი პრივატად გაკეთებით, ოღონდ Public თვისების დამატებით მისი წვდომის მიზნით;
- Extract Interface - ინტერფეისის ამღება: კლასის ტექსტის მონიშვნა მისთვის ავტომატურად შესაბამისი ინტერფეისის ფორმირება (თუ ეს შესაძლებელია);
- Remove parameters - მეთოდის პარამეტრების ნაწილის წაშლა;
- Reorder parameters - მეთოდის პარამეტრების რიგითობის შეცვლა.

დავალება - საკურსო პროექტში გამოსაყენებლად !

ააგეთ მარტივი კონსოლის აპლიკაციის პროექტი Visual Studio.NET პლატფორმის სამუშაო გარემოში (მაგალითად: უნივერსიტეტი, სუპერმარკეტი, აფთიაქი, პროდუქციის_წარმოება ან მიწოდება და ა.შ.).

ჩაატარეთ რეფაქტორინგის პროცედურები და გამოიკვლიეთ მისი ზემოქანმოთვლილი სახეების შესაძლებლობები.

3.5. პროგრამული აპლიკაციის მოდულური ტესტირება

Unit testing და Coded UI არის Microsoft-ის ტესტირების ინსტრუმენტები, რომლებიც სრულდება Visual Studio.NET-პლატფორმის გარემოში.

Unit testing - ანუ მოდულური ტესტირება დაპროგრამების პროცესია, რომლის საშუალებითაც მოწმდება საწყისი კოდის ცალკეული მოდულების კორექტულობა. ასეთი ტესტირების იდეა მდგომარეობს იმაში, რომ ყოველი არატრივიალური ფუნქციის ან მეთოდისათვის დაიწეროს ტესტი. ეს უზრუნველყოფს კოდის სწრაფად შემოწმებას, ხომ არ მიიყვანა კოდის ბოლო ცვლილებამ პროგრამა რეგრესიამდე ანუ შეცდომების გაჩენამდე პროგრამის უკვე ტესტირებულ ნაწილებში [39].

Coded UI ტესტი კი ავტომატურად იწერს, შესრულებაზე უშვებსა და ამოწმებს ტესტ - ქეისებს.

ასეთი ტესტების წერა შესაძლებელია C# ან Visual Basic-ზე Visual Studio გარემოში. Unit testing ტესტირების ტექნოლოგია განვიხილოთ ვირტუალური ობიექტის, მაგალითად, ფინანსური ობიექტის, ბანკის მაგალითზე.

- დასატესტი პროგრამის პროექტის შექმნა: Visual Studio-ში საჭიროა ავირჩიოთ:

File -> New -> Project

შედეგად გამოჩნდება დიალოგური ფანჯარა (ნახ.3.35). სადაც ავირჩევთ:

Visual C# => ClassLibrary

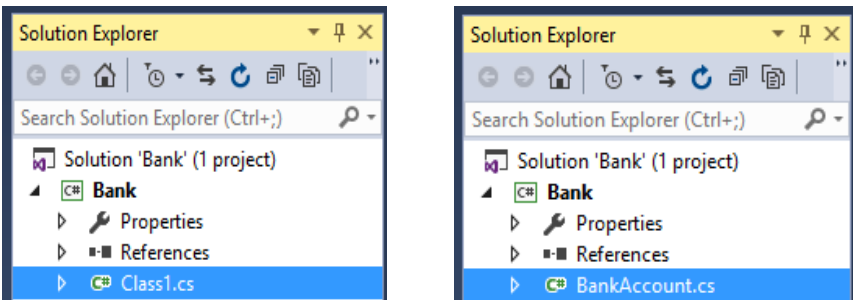
პროექტი სახელით Bank.

პროგრამული ინჟინერის საფუძვლები



ნახ.3.35. დასატესტი კლასის პროექტის შექმნა

მიიღება 3.36 ნახაზზე ნაჩვენები Solution Explorer ფანჯარა. აქ Class1.cs სახელი შეცვალეთ BankAccount.cs -ით.



ნახ.3.36. Class1 - > BankAccount

შემდეგ BankAccount.cs-ის ტექსტი რედაქტორის არეში შევცვალეთ ჩვენი დასატესტი პროგრამის კოდით.

ეს საწყისი ტექსტი, მაგალითად, მოცემულია 3.1 ლისტინგში.

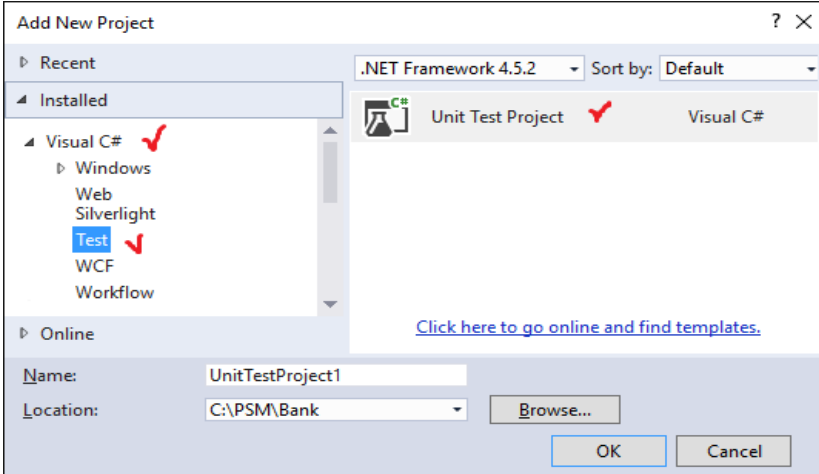
```
//-- ლისტინგი_3.1 --- BankAccount.cs -----
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace BankAccountNS
{
    public class BankAccount
    {
        private string m_customerName;
        private double m_balance;
        private bool m_frozen = false;
        private BankAccount()
        {
        }

        public BankAccount(string customerName, double balance)
        {
            m_customerName = customerName;
            m_balance = balance;
        }
        public string CustomerName
        {
            get { return m_customerName; }
        }
        public double Balance
        {
            get { return m_balance; }
        }
        public void Debit(double amount)
        {
            if (m_frozen)
            {
                throw new Exception("Account frozen");
            }
            if (amount > m_balance)
        }
    }
}
```

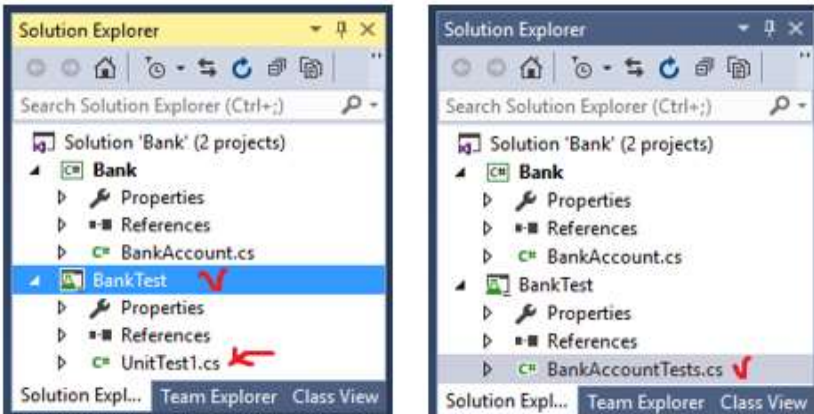
```
{
    throw new ArgumentOutOfRangeException("amount");
}
if (amount < 0)
{
    throw new ArgumentOutOfRangeException("amount");
}
m_balance += amount; // განზრახ არასწორი კოდი
// m_balance -= amount; // გასწორებული
}
public void Credit(double amount)
{
    if (m_frozen)
    {
        throw new Exception("Account frozen");
    }
    if (amount < 0)
    {
        throw new ArgumentOutOfRangeException("amount");
    }
    m_balance += amount;
}
private void FreezeAccount()
{
    m_frozen = true;
}
private void UnfreezeAccount()
{
    m_frozen = false;
}
public static void Main()
{
    BankAccount ba = new BankAccount("Mr.Bryan Walton",
                                     3.99);
    ba.Credit(5.77); ba.Debit(3.22);
    Console.WriteLine("Current balance is ${0}",
                     ba.Balance);
}
}
```

- ტესტ-ვაილის პროექტის აგება (Unit Test Project):



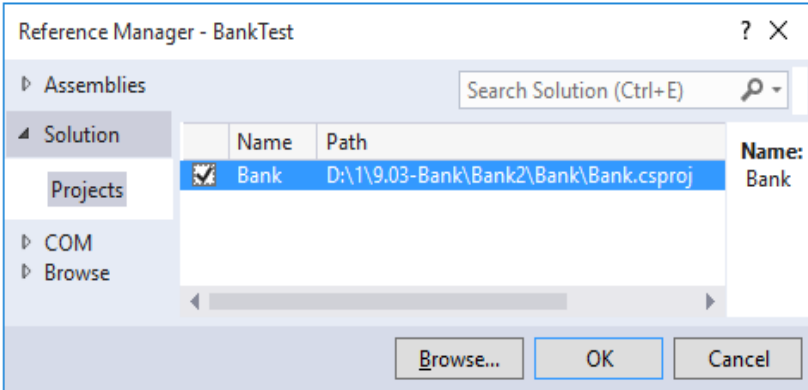
ნახ.3.37. Unit Test პროექტის შექმნა

მივიღებთ 3.38. ნახაზზე ნაჩვენებ სურათს BankTest პროექტით. მარჯვენა სურათზე შეცვლილია UnitTest კლასის სახელი BankAccountTests-ით.



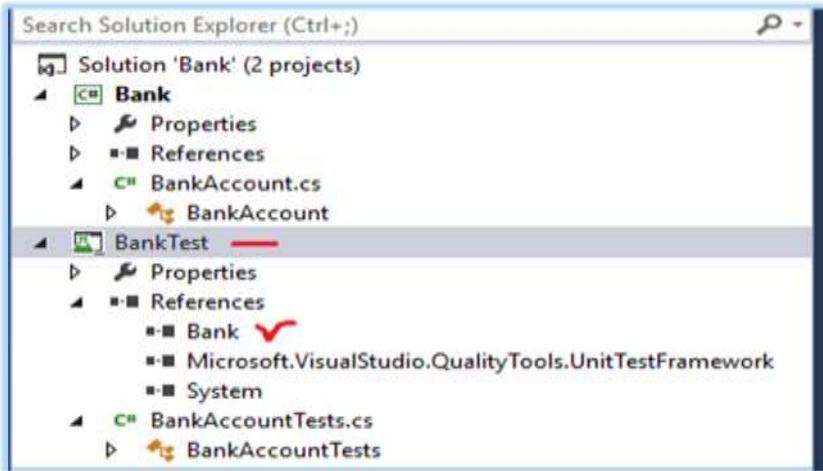
ნახ.3.38

BankTests პროექტში დავამატოთ reference **Bank** solution-იდან. ამისათვის **BankTests**-ზე მაუსის მარჯვენა ღილაკით ავირჩიოთ **Add Reference** და მივიღებთ 3.39 ნახაზზე ნაჩვენებ ფანჯარას. აქ Solution სტრიქონში ვირჩევთ Projects და Bank-ის ჩეკბოქსს მოვნიშნავთ.



ნახ.3.39

მივიღებთ 3.40 ნახაზზე ნაჩვენებ შედეგს.



ნახ.3.40

ჩავამატოთ BankAccountTests პროგრამაში სახელსივრცე Bank-ის პროექტიდან: using Bank;

ამგვარად, BankAccountTests.cs ფაილს ექნება 3.2 ლისტინგზე ნაჩვენები სახე.

```
//-- ლისტინგი_3.2 ----- BankAccountTests.cs -----
using System;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using Bank;

namespace UnitTestProject1
{
    [TestClass]
    public class BankAccountTests
    {
        [TestMethod]
        public void TestMethod1()
        {
        }
    }
}
```

ახლა შევქმნათ პირველი ტესტ - მეთოდი. ამ პროცედურაში, დაიწერება უნიტ - ტესტის მეთოდები BankAccount class-ის Debit მეთოდის ქცევის ვერიფიკაციისათვის. ეს მეთოდები ზემოთაა ჩამოთვლილი.

დასატესტი მეთოდების ანალიზის გზით გაირკვა, რომ საჭიროა მინიმუმ სამი ქცევის შემოწმება:

1. მეთოდი ქმნის ArgumentOutOfRangeException - გამონაკლისს, თუ კრედიტის ჯამი გადააჭარბებს ბალანსს;
2. იგი ქმნის ArgumentOutOfRangeException - გამონაკლისს მაშინაც, როცა კრედიტის ზომა უარყოფითია;
3. თუ 1 და 2 პუნქტები წარმატებით დასრულდა, მაშინ მეთოდი ითვლის ჯამს ბალანსის ანგარიშიდან.

პირველ ტესტში შევამოწმოთ, რომ კრედიტის დასაშვები მნიშვნელობისთვის (როცა დადებითი მნიშვნელობისაა და ბალანსის ანგარიშზე ნაკლებია) ანგარიშიდან მოიხსნება საჭირო თანხა.

1. დავამატოთ BankAccountTests კლასს შემდეგი მეთოდი:

```
// unit test code ----
[TestMethod]
public void Debit_WithValidAmount_UpdatesBalance()
{
    // arrange
    double beginningBalance = 3.99;
    double debitAmount = 4.55;
    double expected = 7.44;
    BankAccount account = new BankAccount("Mr. Dito",
        beginningBalance);
    // act
    account.Debit(debitAmount);
    // assert
    double actual = account.Balance;
    Assert.AreEqual(expected, actual, 0.001, "Account not debited
        correctly");
}
```

მეთოდი საკმაოდ მარტივია. ჩვენ ვქმნით ახალ BankAccount ობიექტს საწყისი ბალანსით და შემდეგ ვაკლებთ სწორ ოდენობას. ჩვენ ვიყენებთ Microsoft-ის unit-ტესტის ფრეიმვორკს მართვადი კოდის AreEqual მეთოდისათვის, რათა მოხდეს საბოლოო ბალანსის ვერიფიკაცია - არის ის, რასაც ჩვენ ველით.

ტესტ - მეთოდის მოთხოვნები ასეთია:

1. მეთოდი მონიშნული უნდა იყოს [TestMethod] ატრიბუტით;
2. მეთოდმა უნდა დააბრუნოს void;
3. მეთოდს არ შეიძლება ჰქონდეს პარამეტრები.

ტესტის აგება და ამუშავება:

მთლიანი ტესტის კოდი მოცემულია 3.3 ლისტინგში.

```
//-- ლისტინგი_3.3 ----- BankAccountTests.cs ----
using System;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using Bank;

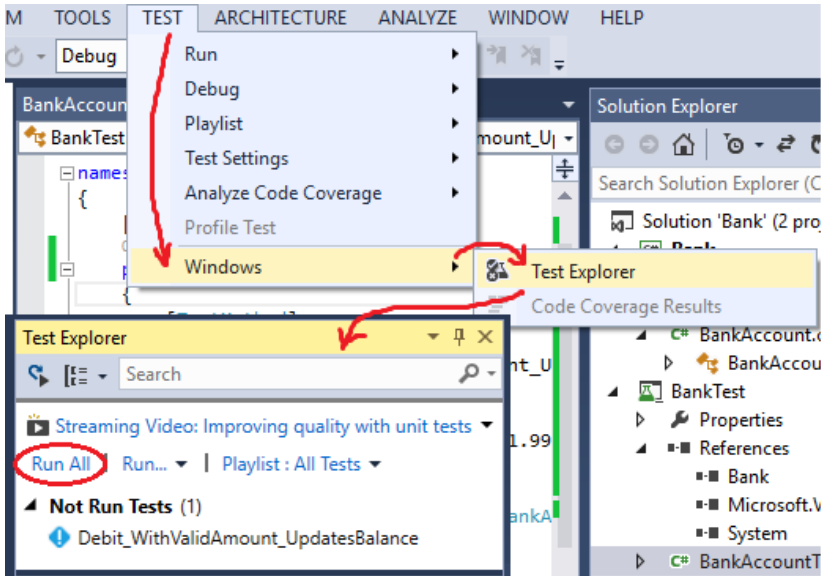
namespace UnitTestProject1
{
    [TestClass]
    public class BankAccountTests
    {
        [TestMethod]
        public void Debit_WithValidAmount_UpdatesBalance()
        {
            // arrange
            double beginningBalance = 3.99;
            double debitAmount = 4.55;
            double expected = 7.44;
            BankAccount account = new BankAccount("Mr. Dito",
                beginningBalance);

            // act
            account.Debit(debitAmount);

            // assert
            double actual = account.Balance;
            Assert.AreEqual(expected, actual, 0.001, "Account
                not debited correctly");
        }
    }
}
```

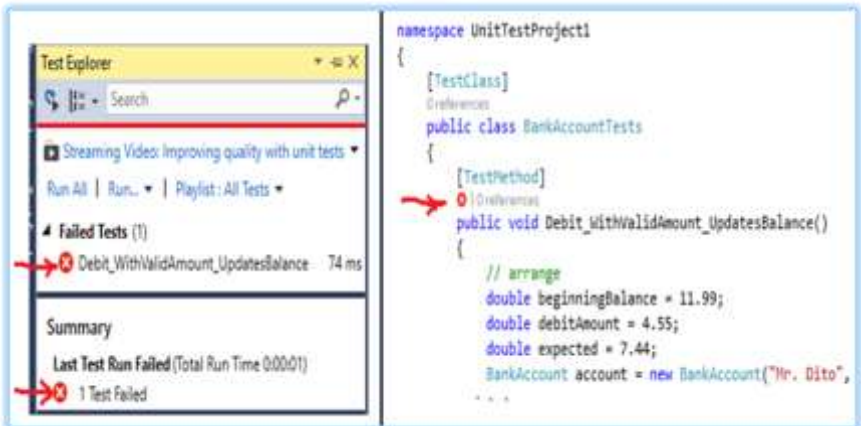
- BUILD მენიუდან ვირჩევთ Build Solution;
 - TEST მენიუდან ვირჩევთ Windows და Test Explorer პუნქტებს.
- ობსნება Test Explorer ფანჯარა (ნახ.3.41).

პროგრამული ინჟინერის საფუძვლები



ნახ.3.41.

აქ ვირჩევთ Run All - სა და ვიღებთ შედეგს (ნახ.3.42).



ნახ.3.42

პროგრამული ინჟინერის საფუძვლები

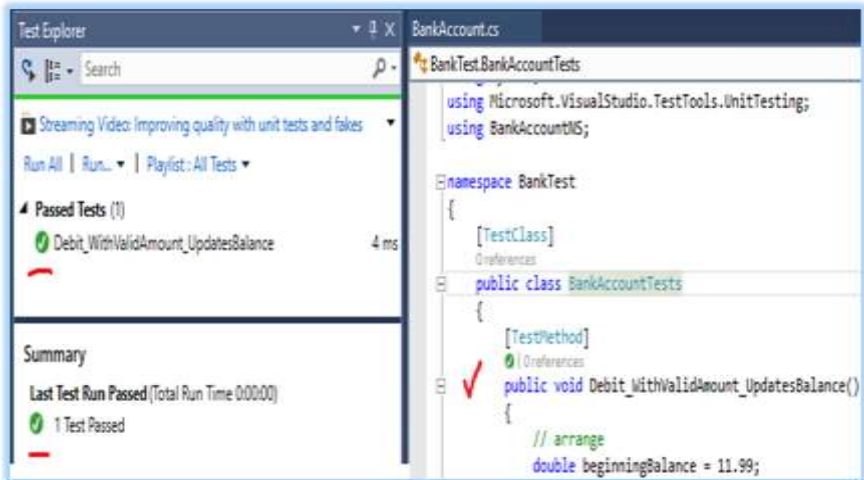
ნახაზზე მითითებული „x“-სიმბოლოები წითელ წრეშია მოთავსებული, ე.ი. ტესტირებამ აღმოაჩინა შეცდომები და კოდი წარმატებით ვერ შესრულდა.

თუ მეთოდი წარმატებით ჩაივლიდა, მაშინ მივიღებდით მწვანე ფერის x-სიმბოლოებს.

შემდეგი ეტაპი კოდის გასწორება და ხელახალი ტესტირებაა. დასატესტ პროგრამაში შევცვალოთ სტრიქონში „ + „ ნიშანი „ - „ -ით.

```
// m_balance += amount; // intentionally incorrect code  
m_balance -= amount; // intentionally correct code
```

ტესტის თავიდან ამუშავებით ვიღებთ წარმატებულ შედეგს ანუ მიიღება მწვანე ფერის სიმბოლოები (ნახ.3.43).

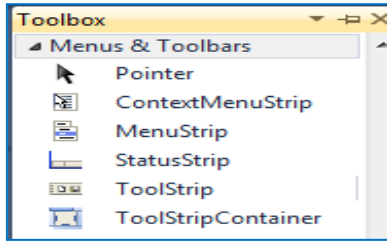


ნახ.3.43. სწორი შედეგი

IV თავი მომხმარებლის ინტერფეისის დამუშავება

4.1. მომხმარებლის ინტერფეისის მთავარი მენიუს აგება

C# ენის მენიუს აგების ვიზუალური ელემენტები, რომლებიც Visual Studio.NET-ის ინსტრუმენტების პანელზეა განთავსებული, ნაჩვენებია 4.1 ნახაზზე.



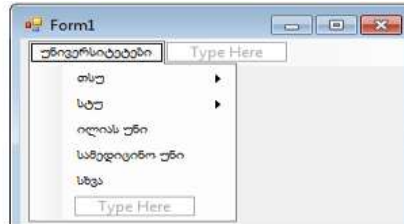
ნახ.4.1. Menus & Toolbars ელემენტები

➤ მთავარი მენიუს ვიზუალური დაპროგრამება

ამოცანა_1: Form1-ზე ავაგოთ მთავარი მენიუ „უნივერსიტე-ტები“ და ქვემენიუს პუნქტებით „ფაკულტეტები“ და „კათედრები“. ფორმაზე მთავარი მენიუს შესაქმნელად Toolbox-იდან გადმოვიტანოთ MenuStrip ელემენტი. ფორმაზე გაჩნდება 4.2-ა ნახაზზე ნაჩვენები გამოსახულება. შევიტანოთ მენიუს პუნქტებს (4.2-ბ).



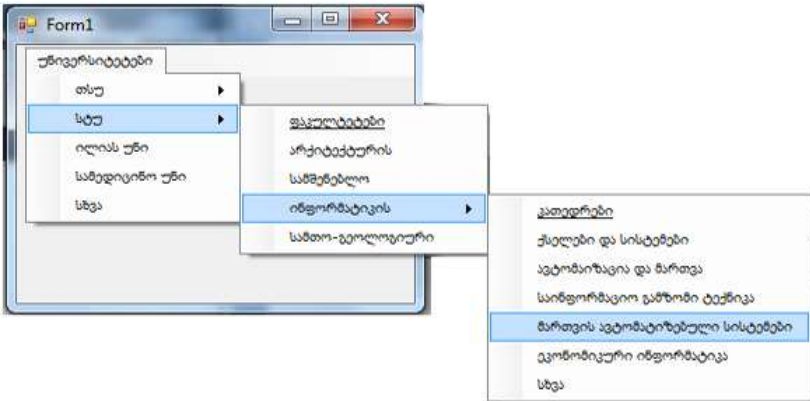
ნახ.4.2-ა



ნახ.4.2-ბ

პროგრამული ინჟინერიის საფუძვლები

ეს ხორციელდება ვერტიკალურად ან ჰორიზონტალურად. თითოეული სტრიქონისთვის შეიძლება ქვემენიუს შექმნა (ნახ.4.3).

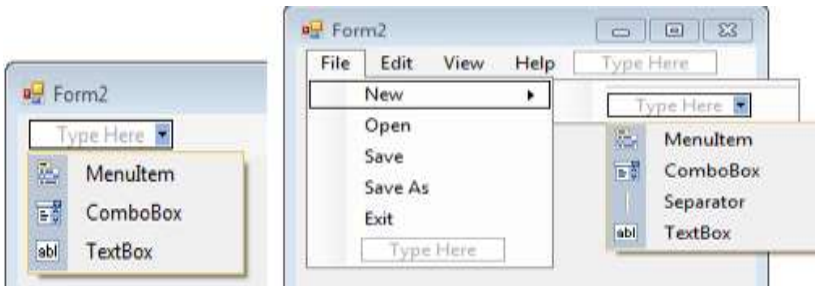


ნახ.4.3. სამდონიანი მენიუ

მენიუს პუნქტების (სტრიქონების) გადაადგილება შეიძლება Form1[Design] რეჟიმში მაუსის მარცხენა ღილაკით Drag&Drop (გადატანა) საშუალებით.

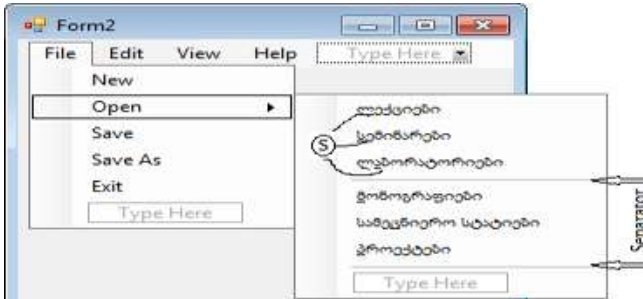
მთავარი მენიუს პუნქტები შიძლება სამი სახისა იყოს (ნახ.4.4):

- ჩვეულებრივი შესატანი სტრიქონი;
- კომბობოქსი, რომელშიც მოხდება ამორჩევა ან შეტანა;
- ტექსტოქსი.



ნახ.4.4

ქვემნიუსთვის დასაშვებია Separator პუნქტიც. რომლის დანიშნულებაა მენიუს პუნქტების ერთმანეთისაგან გამოყოფა ხაზით, რაც ვიზუალურ კომფორტს უქმნის მომხმარებელს (ნახ.4.5).



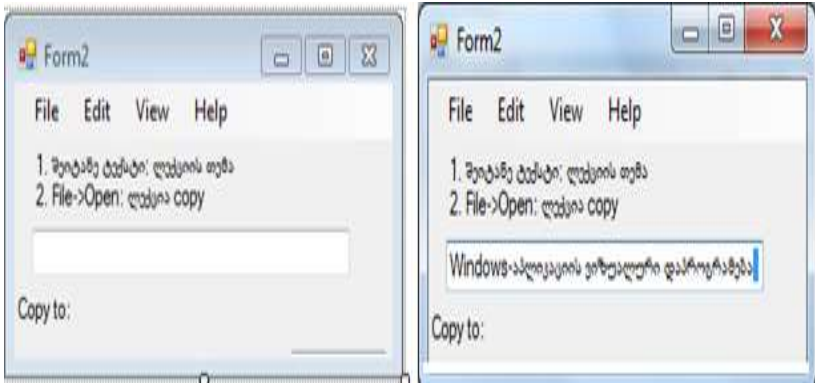
ნახ.4.5. Separator-ის გამოყენება

მენიუში ხშირად იყენებენ სტრიქონის ერთი ასოს გამოყოფას (ქვეშეგახაზვა), რომლითაც ამოქმედდება ეს პუნქტი. ნახაზზე იგი S სიმბოლოთია მითითებული.

მენიუს პუნქტებით უნდა მოხდეს გარკვეული დავალების შესრულება (საჭირო ინფორმაციისკენ გზის განსაზღვრა და ბოლოს ამორჩევა). ამიტომ ეს პუნქტები დაკავშირებულია მოვლენებითან და მეთოდებთან. მოვლენის მთავარი სახეა Click, რომელზეც მიბმულია შესასრულებელი პროცედურის კოდი. განვიხილოთ ეს საკითხი.

ამოცანა_2: ფორმაზე (ნახ.4.6) ავავოთ მთავარი მენიუ, რომლის პუნქტი „ლექციები“ გადააკოპირებს textBox1-ში ჩაწერილ ლექციის თემის დასახელებას label1-ში. მენიუს Exit პუნქტის არჩევისას კი პროგრამა დაასრულებს მუშაობას.

textBox1-ში ჩაწერილი სტრიქონი „Windows-აპლიკაციის ვიზუალური დაპროგრამება“ მთავარი მენიუს „File->Open->ლექციები“ პუნქტის არჩევით ამ სტრიქონს გადააკოპირებს label9 ველში, რომელზეც „Copy to“-ლებელის მარჯვნივაა მოთავსებული (ის არ ჩანს).



ნახ.4.6. ფორმის მაკეტი

ამ მოვლენის დამმუშავებელს აქვს ლისტინგში მოცემული კოდის ფორმა:

// ლისტინგი ---- მთავარი მენიუ: File -> Open -> ლექციები -----

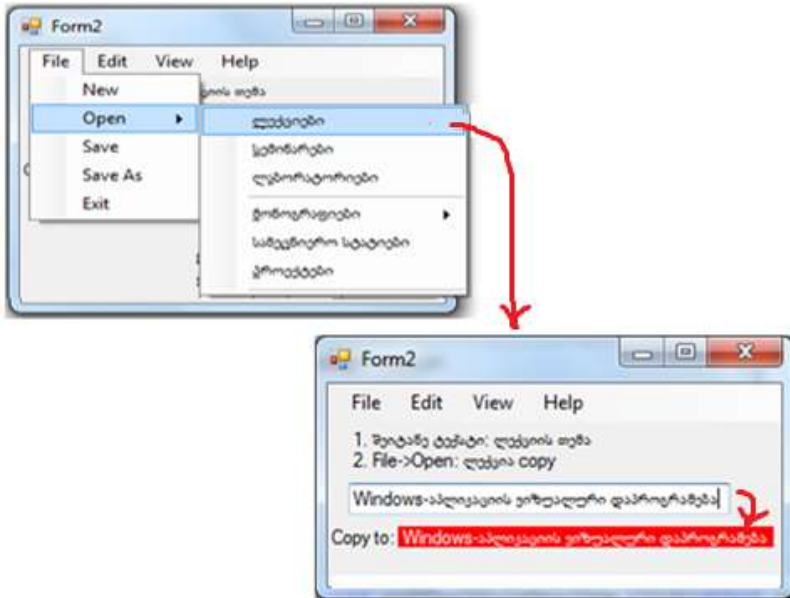
```
private void ToolStripMenuItem_Click(object sender, EventArgs e)
```

```
{
    label9.BackColor = Color.Red;
    label9.ForeColor = Color.White;
    label9.Text = textBox1.Text;
}
```

```
private void exitToolStripMenuItem_Click(object sender, EventArgs e)
```

```
{
    Close();
}
```

შედეგი მოცემულია 4.7 ნახაზზე, წითელი ფონით და თეთრი ტექსტით.



ნახ.4.7. Copy to -ს label-ში ჩაიწერა სტრიქონი textBox1-დან

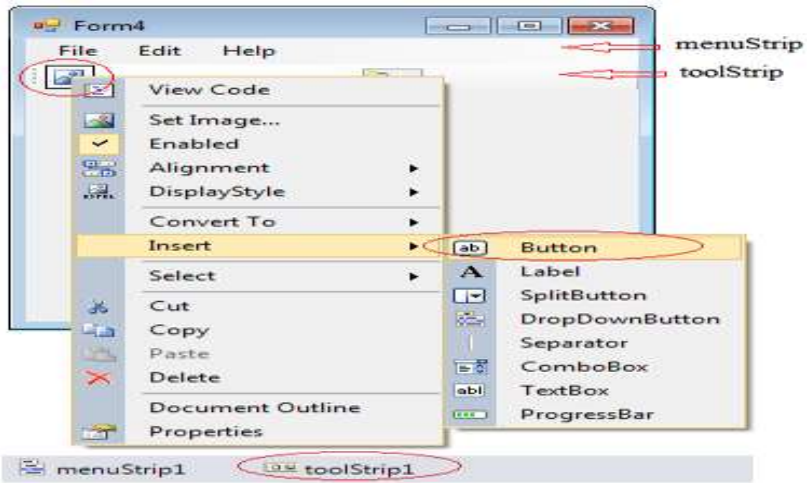
4.2. გრაფიკული მენიუს აგების ვიზუალური აგება

კომპიუტერული სისტემების აგების დროს ინტერფეისში მთავარი მენიუს გარდა ხშირად იყენებენ გრაფიკულ პიქტოგრამებს (icons), რაც უფრო ეფექტურს და მოქნილს ხდის მას.

C# ენაში ასეთი ვიზუალური ელემენტია Menus&Toolbars პანელის toolStrip ელემენტი. მისი გადატანით ფორმაზე მივიღებთ 4.8 ნახაზზე ნაჩვენებ სურათს. საჭიროა ავირჩიოთ სახე: button, Label, ComboBox და ა.შ.

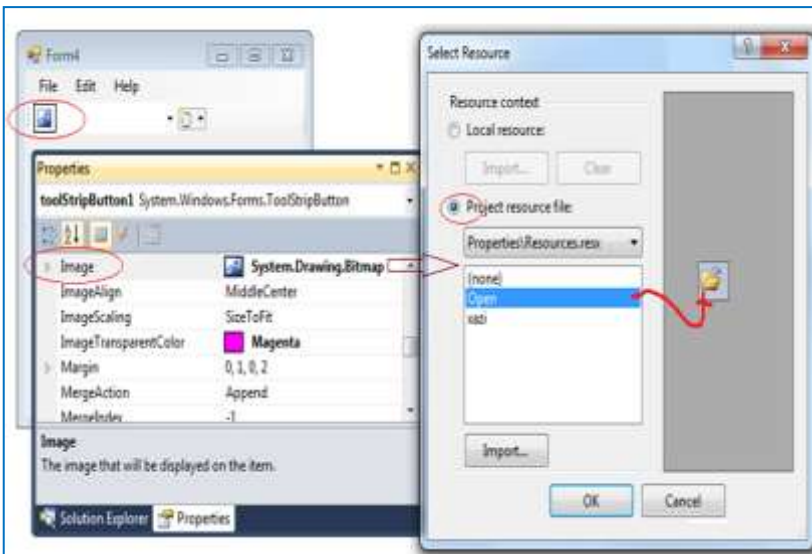
აქ შესაძლებელია მაუსის მარჯვენა ღილაკით სტანდარტულად მიღებული პიქტოგრამის შეცვლა.

პროგრამული ინჟინერის საფუძვლები



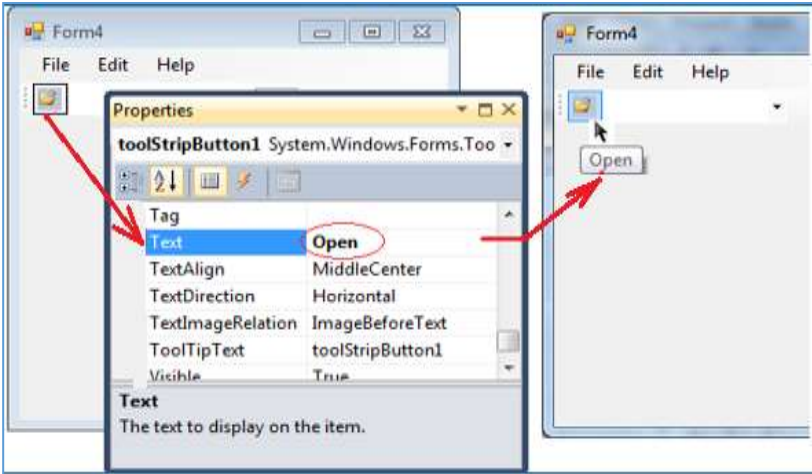
ნახ.4.8

ჯერ Properties-ში Image თვისების არჩევით და შემდეგ საჭირო პიქტოგრამის Import-ირებით დიალოგური ფანჯრიდან (ნახ.4.9).



ნახ.4.9

ახალ პიქტოგრამას თვისებაში Text ჩაუწერეთ მისი ფუნქციის შესაბამისი სიტყვა, მაგალითად, Open. 4.10 ნახაზზე ჩანს მიღებული შედეგი.



ნახ.4.10

შედეგები რამდენიმე ახალი პიქტოგრამის ჩასმის შემდეგ, რომელთაგანაც ერთ-ერთი comboBox ტიპისაა, მოცემულია 4.11 ნახაზზე.



ნახ.4.11

4.3. კონტექსტური მენიუს ვიზუალური აგება

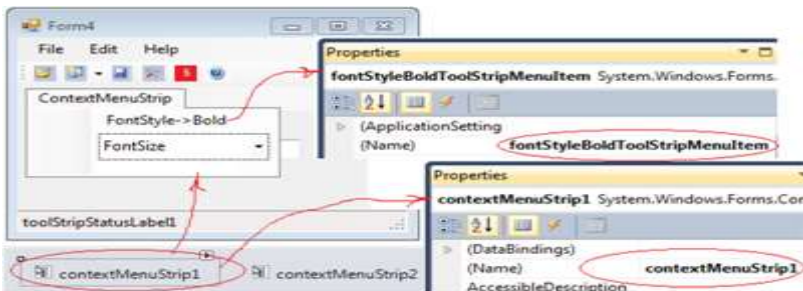
განვიხილოთ ToolBox პანელიდან ContextMenuStrip ელემენტის გამოყენების საკითხი. კონტექსტური მენიუს სტრიქონთა მნიშვნელობების შესაბამისობაში მოყვანა ფორმის ან ფორმაზე დადებული ელემენტებისთვის საჭირო ფუნქციებთან.

ამოცანა_3: ავაგოთ ფორმა, მაგალითად, 4.12 ნახაზზე მოცემულია სახით, რომელზეც label1 და textBox1 ელემენტებია დადებული. ამ ელემენტებისთვის უნდა შეიქმნას ორი კონტექსტური მენიუ. textBox1-ში ჩაწერილი სტრიქონი კონტექსტური მენიუთი უნდა გადაკოპირდეს label1-ში. შემდეგ label1-ში გადატანილი სტრიქონის ფორმა (სტილი, ზომა) უნდა შეიცვალოს კონტექსტური მენიუდან.



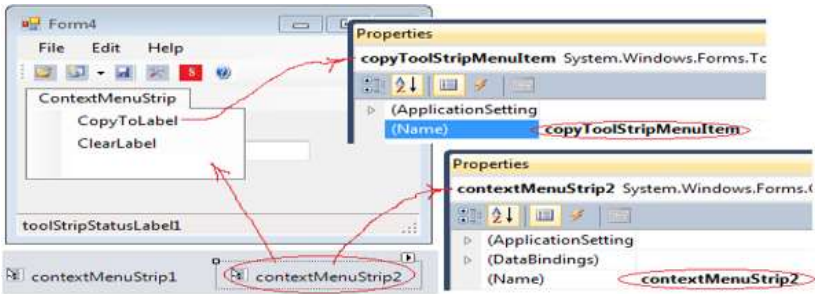
ნახ.4.12

4.13-ა,ბ ნახაზებზე ნაჩვენებია Form4-ზე მოთავსებული label1 და textBox1 ელემენტებისთვის ჩვენ მიერ შექმნილი კონტექსტური მენიუები.



ნახ.4.13-ა

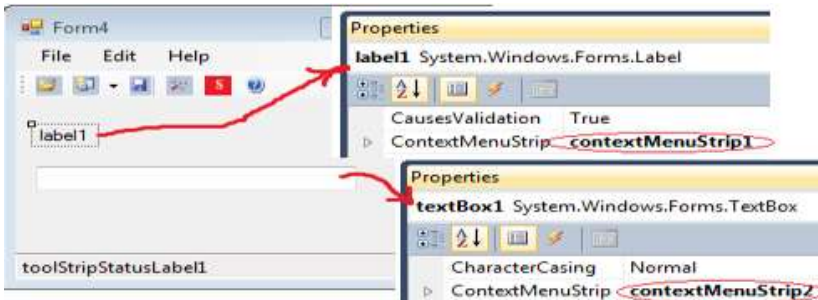
პროგრამული ინჟინერიის საფუძვლები



ნახ.4.13-ბ

შესაბამისად ContextMenuStrip1 და ContextMenuStrip2. კონტექსტური მენიუები აგებულია. ახლა ისინი უნდა „მივაბათ“ Form4-ის textBox1 და label1 ელემენტებს, რათა მაუსის მარჯვენა ღილაკმა იმუშაოს და ამ ელემენტებზე კურსორის მიტანისას გამოჩნდეს კონკრეტულად მისი შესაბამისი კონტექსტური მენიუ.

ამისათვის მოვნიშნოთ label1 და მის Properties-ის ContextMenuStrip თვისებაში ჩავწეროთ contextMenuStrip1 მნიშვნელობა (ნახ.4.14). ასევე textBox1-სთვის ჩავწეროთ contextMenuStrip2.



ნახ. 4.14

პროგრამული კოდი მოცემულია მომდევნო ლისტინგში, შედეგები კი 4.15-4.17 ნახაზებზე.

// ლისტინგი -- შრიფტის შეცვლა ---

```
using System;
```

```
using System.Drawing;
using System.Windows.Forms;
using System.Threading;

namespace WinMenus
{
    public partial class Form4 : Form
    {
        double DamtavrDro;
        public Form4()
        {
            InitializeComponent();
        }
        private void Form4_Load(object sender, EventArgs e)
        {
            toolStripStatusLabel1.Text = DateTime.Today.ToShortDateString();
        }
        private void Stop_Click(object sender, EventArgs e)
        {
            DamtavrDro = 0;
            timer1.Enabled = true;
        }
        private void timer_Tick(object sender, EventArgs e)
        {
            DamtavrDro += 0.1;
            if (DamtavrDro >= 5)
                Close();
            else
                toolStripProgressBar1.Value = (int)DamtavrDro;
            textBox1.Text = DamtavrDro.ToString();
        }
        private void copyToolStripMenuItem_Click(object sender, EventArgs e)
        {
            label1.Text = textBox1.Text;
        }
    }
}
```

```
        if (label1.Text == "")
            label1.Text = "(leer)";
    }
    private void clearLabelToolStripMenuItem_Click(object sender,
                                                    EventArgs e)
    {
        label1.Text = "";
    }
    private void fontStyleBoldToolStripMenuItem_Click(object sender,
                                                       EventArgs e)
    {
        label1.Font = new Font(label1.Font.FontFamily, label1.Font.Size,
                                label1.Font.Style ^ FontStyle.Bold);
        fontStyleBoldToolStripMenuItem.Checked =
            !fontStyleBoldToolStripMenuItem.Checked;
    }
    private void fontSize16ToolStripMenuItem_Click(object sender,
                                                    EventArgs e)
    {
        label1.Font = new Font(label1.Font.FontFamily, label1.Font.Size,
                                label1.Font.Style ^ FontStyle.Italic);
        fontStyleBoldToolStripMenuItem.Checked =
            !fontStyleBoldToolStripMenuItem.Checked;
    }
    private void toolStripComboBox1_TextChanged(object sender,
                                                EventArgs e)
    {
        double FontSize;

        try
        {
            FontSize = Convert.ToDouble(toolStripComboBox1.Text);
        }
        catch
```

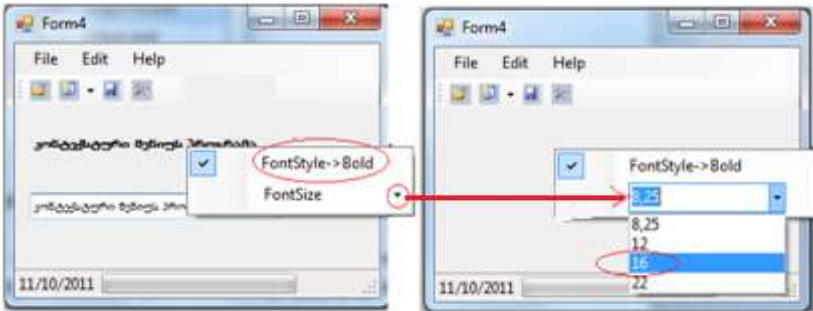
```
{
    FontSize = 8.25;
}

label1.Font = new Font(label1.Font.FontFamily, (float)FontSize,
    label1.Font.Style);
}
private void toolStripComboBox1_Click(object sender, EventArgs e)
{
    toolStripComboBox1.Items.Clear();
    toolStripComboBox1.Items.Add("8,25");
    toolStripComboBox1.Items.Add("12");
    toolStripComboBox1.Items.Add("16");
    toolStripComboBox1.Items.Add("22");
    toolStripComboBox1.SelectedIndex = 0;
}
}
}
```

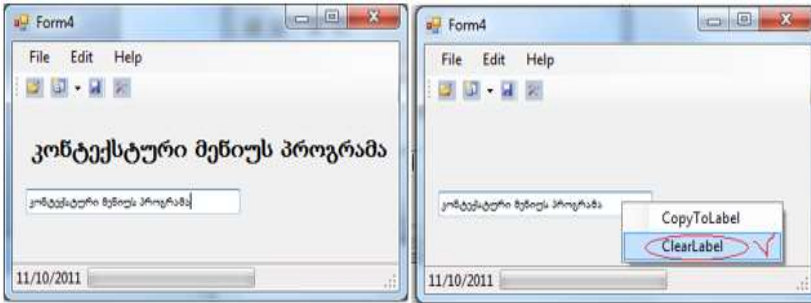
შედეგები:



ნახ.4.15. შედეგები



ნახ.4.16



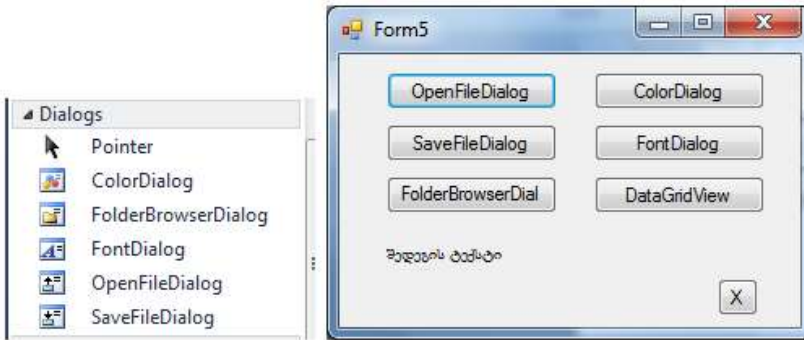
ნახ.4.17

4.4. C# ენის ვიზუალური სტანდარტული დიალოგური საშუალებები

დაპროგრამების ვიზუალურ C# ენაში არსებობს ხუთი სახის დიალოგური კლასი, რომლებიც ხშირად გამოიყენება პროგრამული პროექტების აგების პროცესში:

- OpenFileDialog
- SaveFileDialog
- FolderBrowserDialog
- ColorDialog და
- FontDialog.

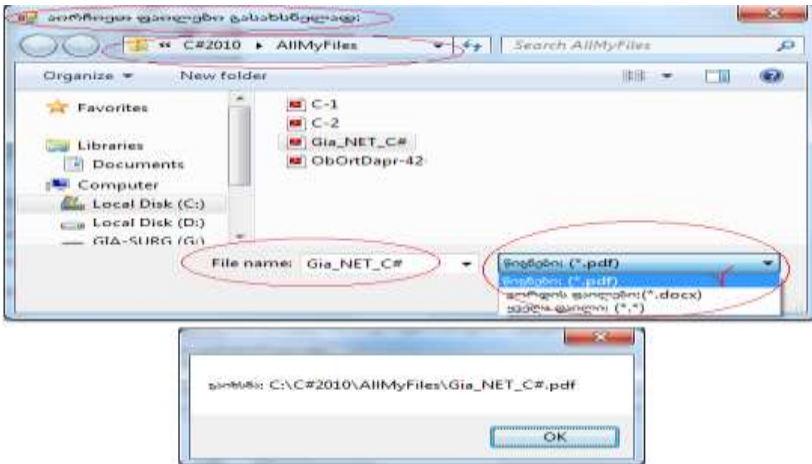
განვიხილოთ ეს დიალოგები დეტალურად (ნახ.4.18).



ნახ.4.18

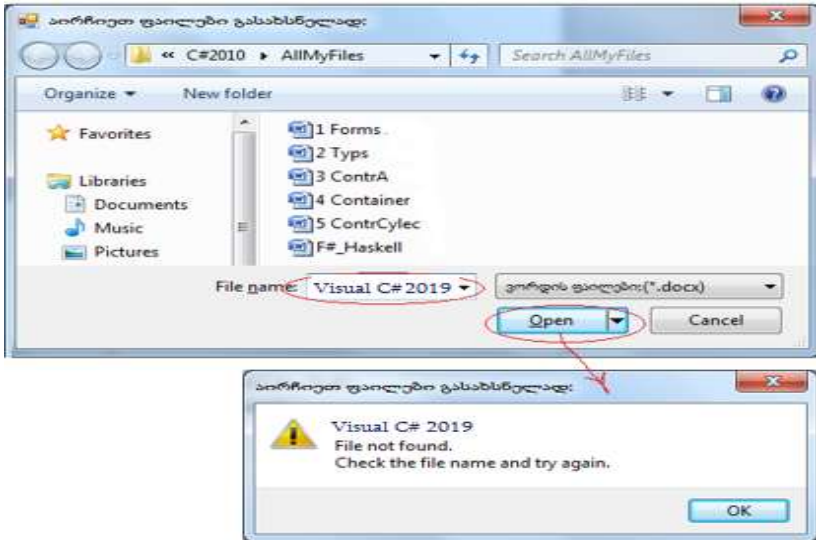
➤ **OpenFileDialog** კლასის ობიექტის დანიშნულებაა გასახსნელი ფაილის ამორჩევა, მითითებული ფოლდერის (InitialDirectory), ფილტრის (Filter - მაგალითად, ფაილის ტიპით და დიალოგური ველის სათაურის (Title) მიხედვით (ნახ.4.19 და 4.20).

დიალოგის შედეგი ფაილის სახელის თვისებისთვის იქნება - FileName.



ნახ.4.19. დადებითი შედეგით

პროგრამული ინჟინერიის საფუძვლები



ნახ.4.20. უშედეგოდ დასრულება

// ლისტინგი --- OpenFileDialog -----

```
private void button1_Click(object sender, EventArgs e)
{
    OpenFileDialog f = new OpenFileDialog();
    f.InitialDirectory = "c:\C#2010\AllMyFiles";
    f.Filter = "წიგნები: (*.pdf)|*.pdf" +
              " ვორდის ფაილები (*.docx)|*.docx|" +
              " ყველა ფაილი: (*.*)|*.*";
    f.Title = "აირჩიეთ ფაილები გასახსნელად:";
    if (f.ShowDialog() == DialogResult.OK)
        MessageBox.Show("გაიხსნა: " + f.FileName);
    else
        MessageBox.Show("უშედეგოდ დასასრული !");
}
```


➤ **SaveFileDialog** კლასის ობიექტის დანიშნულებაა იმ ფაილის შეტანა ან ამორჩევა, რომელიც შენახულ უნდა იქნას. შენახვის და დიალოგის განსახორციელებლად მითითებულ უნდა იქნას ფოლდერის (InitialDirectory), ფილტრის (Filter - მაგალითად, ფაილის ტიპით და დიალოგური ველის სათაური (Title).

```
// ლისტინგი --- SaveFileDialog -----
private void button2_Click(object sender, EventArgs e)
{
    SaveFileDialog fs = new SaveFileDialog();
    fs.InitialDirectory = "c:\C#2010\AllMyFiles";
    fs.Filter = "წიგნები: (*.pdf)|*.pdf" +
               " ვორდის ფაილები: (*.docx)|*.docx|" +
               " ყველა ფაილი: (*.*)|*.*";
    fs.Title = "ფაილების არჩევა შესანახად:";
    if (fs.ShowDialog() == DialogResult.OK)
        MessageBox.Show("შენახვა: " + fs.FileName);
    else
        MessageBox.Show("უშედეგო დასასრული !");
}
```

➤ **FolderBrowserDialog** კლასის ობიექტის დანიშნულებაა კატალოგის (ფოლდერის) ამორჩევა, რომელიც იქნება მომდევნო პროგრამული პროცედურების საბაზო წერტილი. შესაძლებელია ასევე ახალი კატალოგის შექმნა. დიალოგური ფორმის გახსნის წინ საჭიროა შემდეგი პარამეტრების მიწოდება: RootFolder: კატალოგი, რომელიც უნდა გამოჩნდეს დიალოგის ველში.

ShowNewFolderButton: ახალი კატალოგის შექმნისათვის საჭირო ბუტონის მითითება. Description: დიალოგური ველის სათაური.

```
// ლისტინგი --- FolderBrowserDialog -----
```

```
private void button3_Click(object sender, EventArgs e)
{
    FolderBrowserDialog fb = new FolderBrowserDialog();
    fb.RootFolder = Environment.SpecialFolder.MyDocuments;
    fb.ShowNewFolderButton = false;
    fb.Description = "კატალოგის არჩევა";

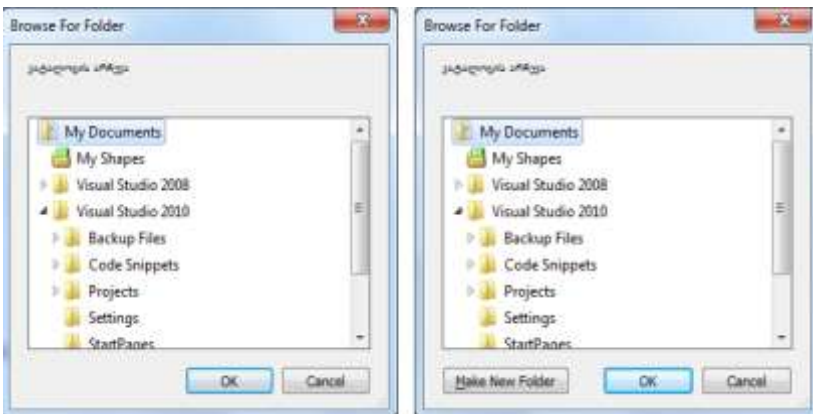
    if (fb.ShowDialog() == DialogResult.OK)
        MessageBox.Show("წვდომა კატალოგზე: " +
fb.SelectedPath);
    else
        MessageBox.Show("უშედეგო დასასრული!");
}
```

სტრიქონით:

fb.RootFolder = Environment.SpecialFolder.MyDocuments;

ფესვურ კატალოგად, ჩვენ შემთხვევაში, აიღება „MyDocuments“.

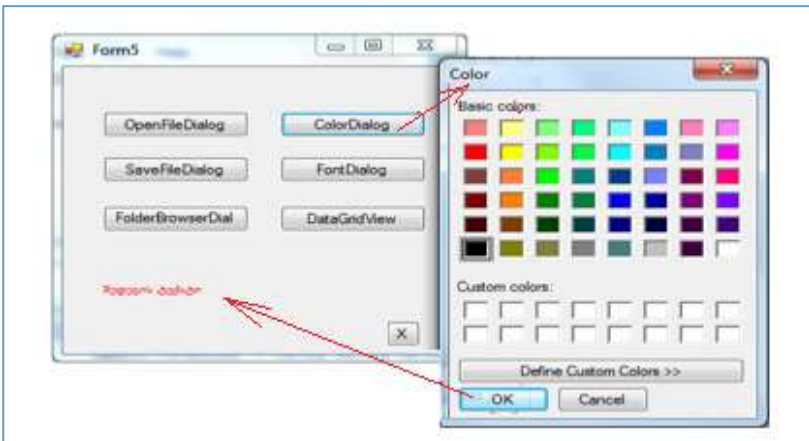
სტრიქონით: fb.ShowNewFolderButton = false; - ახალი კატალოგი არ იქმნება, ხოლო თუ არის “true” , მაშინ ფორმაზე ჩნდება ბუტონი “Make New Folder” (ნახ.4.21).



ნახ.4.21

➤ **ColorDialog** კლასის ობიექტის დანიშნულებაა ფერის არჩევა ფორმაზე მოთავსებული მონიშნული ელემენტისთვის (იხ. ლისტინგი და ნახ.4.22).

```
// ლისტინგი --- ColorDialog -----  
private void button4_Click(object sender, EventArgs e)  
{  
    ColorDialog cd = new ColorDialog();  
    if (cd.ShowDialog() == DialogResult.OK)  
        label1.ForeColor = cd.Color;  
    else  
        MessageBox.Show("უშედეგო დასასრული");  
}
```



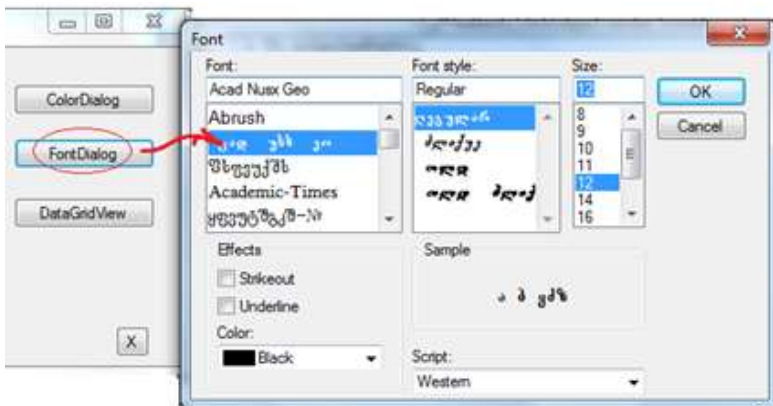
ნახ.4.22

➤ **FontDialog** კლასის ობიექტის დანიშნულებაა შრიფტის (ფონტის) არჩევა ფორმაზე მოთავსებული მონიშნული ელემენტისთვის (იხ. ლისტინგი და ნახ.4.23).

```
// ლისტინგი --- FontDalog -----
```

```
private void button5_Click(object sender, EventArgs e)
{
    FontDialog fd = new FontDialog();
    fd.ShowColor = true;
    fd.MaxSize = 22;
    fd.MinSize = 8;

    if (fd.ShowDialog() == DialogResult.OK)
    {
        label1.Font = fd.Font;
        label1.ForeColor = fd.Color;
    }
    else
        MessageBox.Show("უმედეგო დასასრული");
}
```



ნახ.4.23

V თავი

Web აპლიკაციების აგება ASP.NET ტექნოლოგიით

ASP.NET (Active Server Pages – აქტიური სერვერული გვერდები) არის NET პლატფორმის ნაწილი და ტექნოლოგია, რომელიც დინამიკურად ქმნის დოკუმენტებს Web სერვერზე, როცა ისინი მოითხოვება HTTP-ს საშუალებით.

ASP.NET ტექნოლოგია ანალოგიურია PHP, ColdFusion და სხვა ტექნოლოგიების, მაგრამ მათ შორის მნიშვნელოვანი განსხვავებაა. ASP.NET, როგორც მისი დასახელება გვიჩვენებს, შეიქმნა სპეციალურად NET პლატფორმასთან სრული ინტეგრაციის მიზნით, რომლის ნაწილი ითვალისწინებს C# ენის მხარდაჭერას.

როგორც ცნობილია, Web გვერდების დასაპროგრამებლად გამოიყენება ისეთი სცენარების ენები, როგორცაა VBScript ან JScript. ეს სკრიპტული ენები მუშაობდა, მაგრამ ხშირად გარკვეულ პრობლემებს უქმნიდა დაპროგრამების “ნამდვილი” ენების პროგრამისტებს სხვადასხვა ადმინისტრაციულ დავალებათა შესრულებისას, რაც საბოლოო ჯამში აისახება სისტემის მწარმოებლურობის დაქვეითებაში.

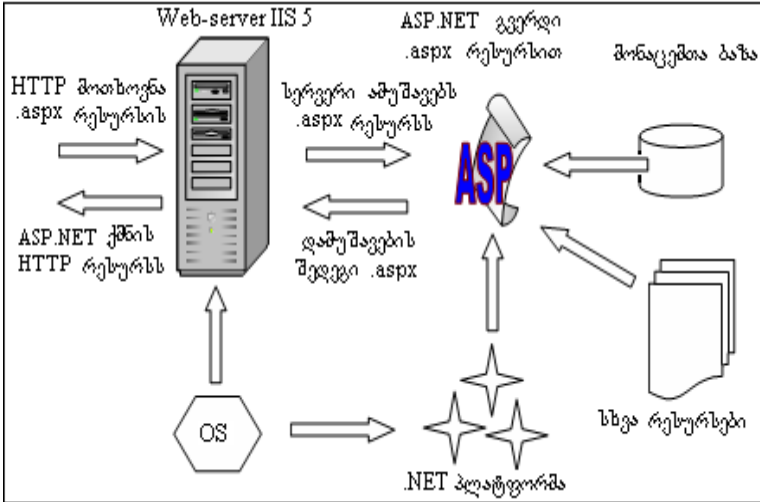
მაღალგანვითარებული ენების გამოყენების შემთხვევაში შესაძლებელია მუშაობის პროცესის უზრუნველყოფა სრული სერვერული ობიექტური მოდელით. ASP.NET ახორციელებს მიმართვას გვერდის ყველა მმართველ ელემენტთან, როგორც ზოგადად გარემოს ობიექტებთან. სერვერის მხარესაც კი ხორციელდება წვდომა .NET-ის ყველა საჭირო კლასთან.

გვერდების მართვის ელემენტები ფუნქციონალურია და ფაქტობრივად, შესაძლებელია ყველაფრის გაკეთება, რასაც Windows-ის ფორმის კლასებთან ვაკეთებდით, რაც უფრო მოქნილ ხდის სისტემას. ამის გამო, ASP.NET-ის გვერდებს, რომლებიც ქმნის HTML შედეგნილობას, ხშირად უწოდებენ Web ფორმებს.

ASP.NET გამოიყენებს ინტერნეტის ინფორმაციულ სერვერს (IIS – Internet Information Server) HTTP მოთხოვნებზე

პროგრამული ინჟინერიის საფუძვლები

საპასუხო შინაარსის მისაწოდებლად. ASP.NET გვერდები მოთავსებულია ფაილებში .aspx გაფართოებით. მისი საბაზო არქიტექტურა მოცემულია 5.1 ნახაზზე.



ნახ.5.1. საბაზო არქიტექტურა

ASP.NET-ის დამუშავებისას მისაწვდომია .NET-ის ყველა კლასი, სპეციალური კომპონენტები, შექმნილი C# ან სხვა ენებზე, მონაცემთა ბაზები და ა.შ. ფაქტობრივად, სახეზეა ყველა ის შესაძლებლობა, რომელსაც იყენებს C# დანართის აგებისას. ე.ი. C#-ის გამოყენება ASP.NET-ში ეფექტურს ხდის დანართის შესრულებას.

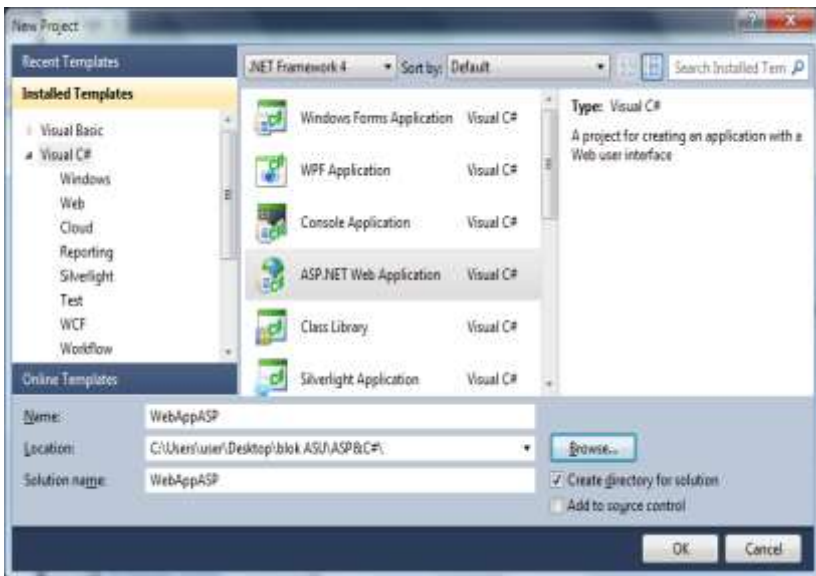
Web აპლიკაციების აგება ASP.NET-ში ხდება კლასების კონსტრუირებით, რომლებიც ურთიერთმოქმედებს სხვა კლასებთან. ზოგი კლასი იწარმოება პლატფორმის საბაზო კლასებიდან, ზოგს შეუძლია ინტერფეისების რეალიზება ამ პლატფორმაში, ზოგიც ურთიერთქმედებს პლატფორმის საბაზო კლასებთან მათი მეთოდების გამოძახების გზით.

5.1. ASP.NET აპლიკაციის შექმნის ეტაპები

ASP.NET აპლიკაციის შესაქმნელად საჭიროა შემდეგი საფეხურების შესრულება:

- პროგრამების პანელიდან ავირჩიოთ:
Start->Programs->Microsoft Visual Studio .NET 2010
- პროგრამის გაშვების შემდეგ აირჩიეთ მენიუს პუნქტი:
File -> New -> Project.

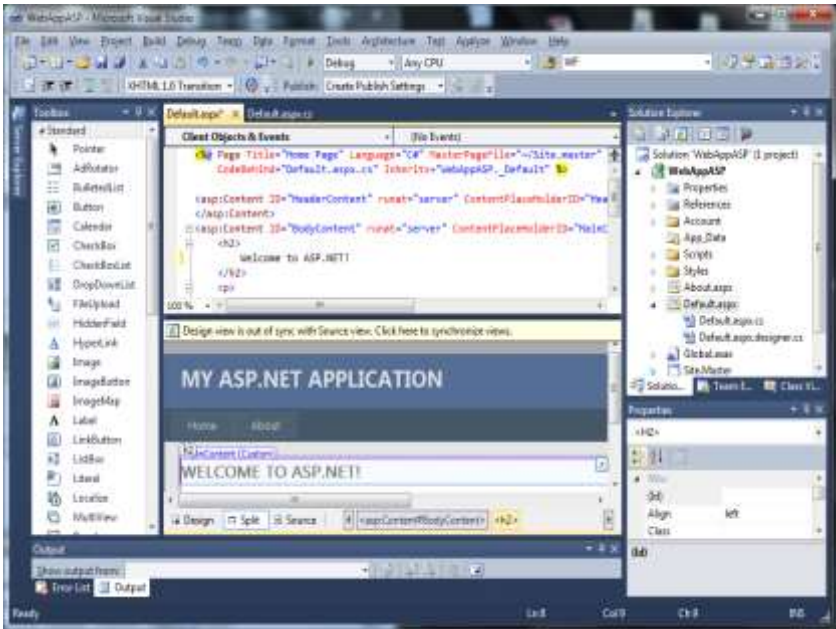
• გაიხსნება ახალი პროექტების ტიპის არჩევის ფანჯარა (ნახ.5.2). ავირჩიოთ Visual C# Project და ASP.NET Web Application. Location ველში მივუთითოთ პროექტის შესანახი ადგილი. შევიტანოთ Web აპლიკაციის სახელი, მაგალითად, WebAppASP და OK.



ნახ.5.2

- Visual Studio შექმნის აპლიკაციას და გახსნის Microsoft Visual C#.NET გარემოში (ნახ.5.3).

პროგრამული ინჟინერის საფუძვლები



ნახ.5.3

შექმნილი პროექტი შეიცავს რამდენიმე ფაილს. მაგალითად:

- Global.asax ფაილი ემსახურება აპლიკაციის დონის მოვლენების დამუშავებას, როგორცაა შეცდომების დაჭერა, ახალი სესიის შექმნა, სესიის დასრულება და სხვა;

- Web.config არის XML ფაილი, რომელიც შეიცავს აპლიკაციის კონფიგურაციის მონაცემებს: სესიის პარამეტრები, მონაცემთა ბაზასთან კავშირის პარამეტრები, მომხმარებლების ავტორიზაციისა და აუტენტიფიკაციის კონფიგურაციის პარამეტრები;

- Default.aspx და Default.aspx.cs ქმნის ერთ ვებ-გვერდს. Default.aspx ფაილი შეიცავს ვიზუალურ ელემენტებს. მისი სტანდარტული საწყისი XML კოდის ლისტინგი ასეთია:

```
<!-- ლისტინგი_5.1 ———>
```

```
<%@ Page Title="Home Page" Language="C#" MasterPageFile="~/Site.master"
AutoEventWireup="true"
CodeBehind="Default.aspx.cs" Inherits="WebAppASP._Default" %>
```



```
<asp:Content ID="HeaderContent" runat="server"
ContentPlaceHolderID="HeadContent">
</asp:Content>
<asp:Content ID="BodyContent" runat="server"
ContentPlaceHolderID="MainContent">
<h2> Welcome to ASP.NET! </h2>
<p>
To learn more about ASP.NET visit
<a href="http://www.asp.net" title="ASP.NET
Website"> www.asp.net</a>.
</p>
<p>
You can also find <a href="http://go.microsoft.com
/fwlink/?LinkId=152368&amp;clid=0x409"
title="MSDN ASP.NET Docs">documentation on ASP.NET at MSDN</a>.
</p>
</asp:Content>
```

- Default.aspx.cs – შეიცავს Web ფორმის კლასის მოვლენების დამუშავების მეთოდებს და ბიზნესლოგიკას. საწყისი კოდი მოცემულია 5.2 ლისტინგში.

```
// ——— ლისტინგი_5.2 ———
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
namespace WebAppASP
{
public partial class _Default : System.Web.UI.Page
{
protected void Page_Load(object sender, EventArgs e)
{
}
}
}
```

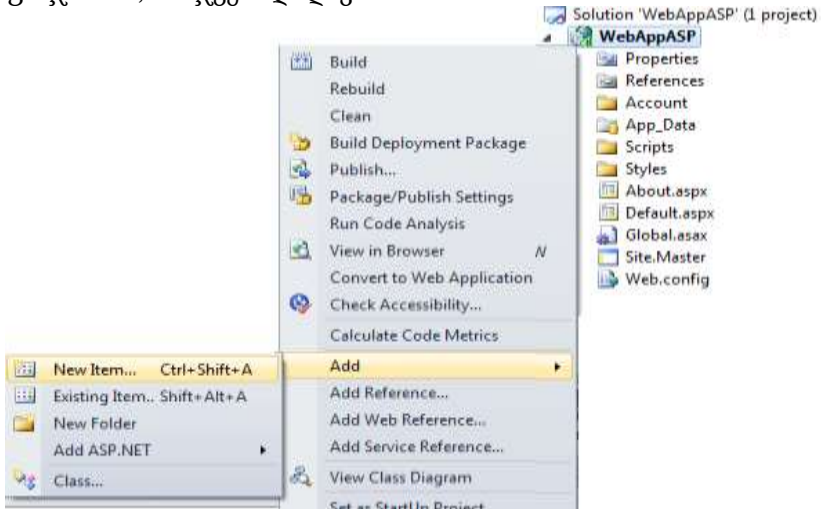
შექმნილი ვებ-გვერდის ნახვა შესაძლებელია Ctrl+F5 დაჭერით ან მენიუს პუნქტი Debug -> Start Without Debugging არჩევით (ნახ.5.4):



ნახ.5.3. შედეგი

5.2. პროექტში ახალი ვებ-გვერდის დამატება

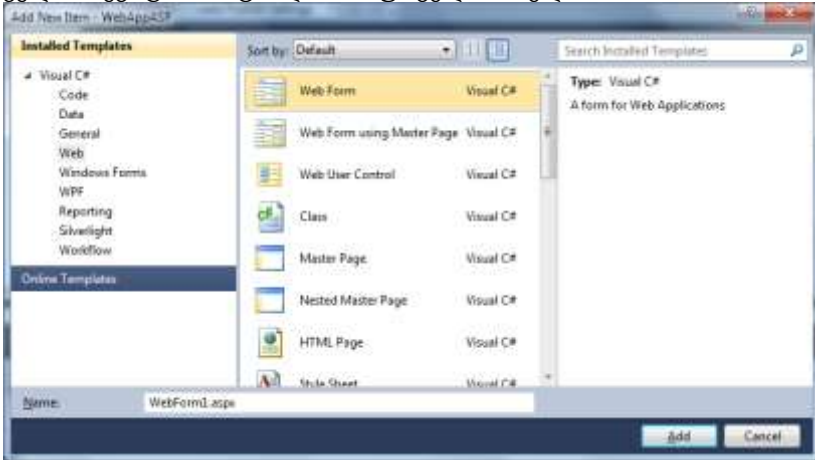
შექმნილ აპლიკაციას დავამატოთ ახალი ASPX ვებ-გვერდი. ახალი ფაილის დამატება შესაძლებელია Solution Explorer ფანჯარაში, მარჯვენა ღილაკით: Add->Add New Item:



ნახ.5.4

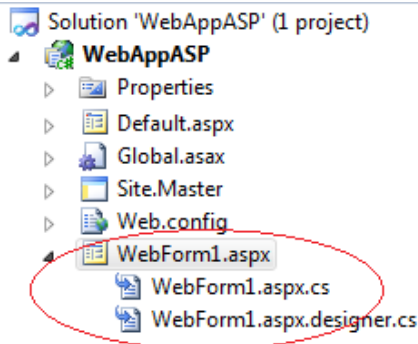
პროგრამული ინჟინერის საფუძვლები

გაიხსნება ახალი ფაილის არჩევის ფანჯარა (ნახ.5.5). ავირჩიოთ Web შაბლონების ფანჯარაში და Web Form. ხოლო Name ველში შევიტანოთ ფაილის სასურველი სახელი:



ნახ.5.5

Add დილაკზე დაჭერით პროექტს დამატება ახალი ფაილები: WebForm1.aspx და WebForm1.aspx.cs (ნახ.5.6). გვერდი ავტომატურად გაიხსნება დიზაინის რეჟიმში.



ნახ.5.6

5.3. ინტერაქტიული Web-გვერდის შექმნა

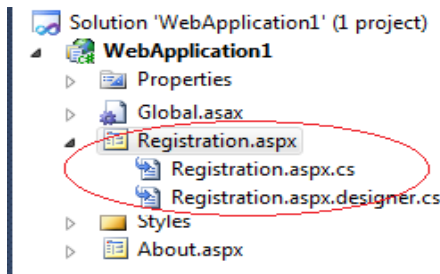
ამოცანა-5.1. ავაგოთ ახალი Web-გვერდი, რომელზეც მომხმარებელი შეიტანს საკუთარ მონაცემებს და გადააგზავნის სერვერზე.

შევქმნათ ახალი ASP.NET აპლიკაცია პროექტის სახელით WebApplication1 (ნახ.5.7).



ნახ.5.7

დავამატოთ ფაილები: Registration.aspx და Registration.aspx.cs.



ნახ.5.8

Web-გვერდის სარეგისტრაციო ფორმის მაკეტი ნაჩვენებია 5.9 ნახაზზე.

ნახ.5.9

ფორმაზე მოთავსებულია სერვერული მართვის ელემენტები form, asp:TextBox, asp:DropDownList, asp:CheckBoxList, asp:Button, asp:Label და ა.შ., რომლებიც ასახულია Registration.aspx ფაილის 5.3 ლისტინგში. გახსენით Registration.aspx და შეიტანეთ შემდეგი კოდი:

```
<!-- ლისტინგი_5.3 -->
<%@ 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>

```

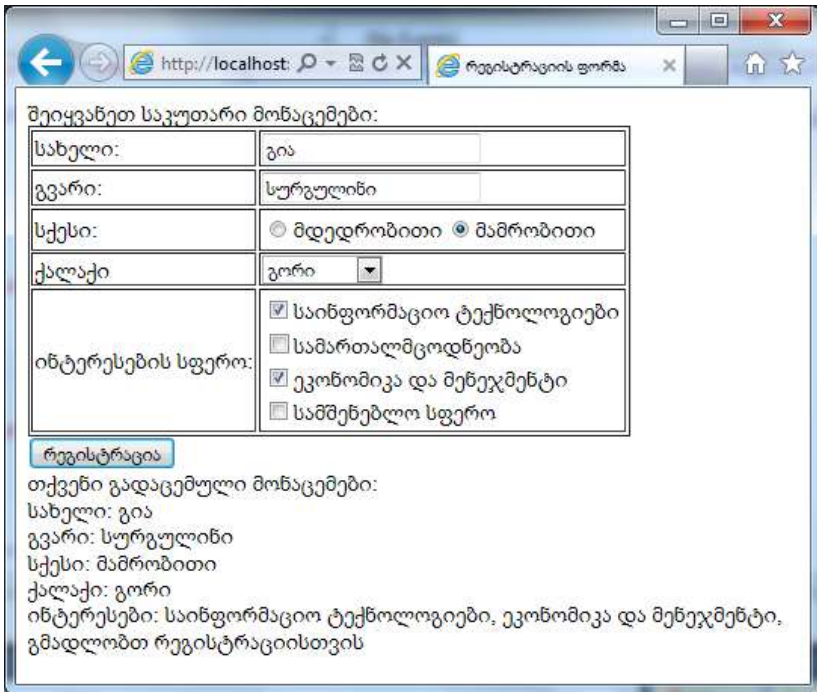
```
<asp:ListItem Value="სამშენებლო სფერო"></asp:ListItem>
    </asp:CheckBoxList></td>
</tr>
</table>
<asp:Button id="Register" runat="server" Text="რეგისტრაცია"
OnClick="Register_Click"></asp:Button>
<br />
<asp:Label id="Message" runat="server"></asp:Label>
</form>
</body>
</html>
```

Web-გვერდის ჩატვირთვის და მონაცემების შევსების შემდეგ „რეგისტრაცია“ Button-ის დაჭერისას გამოიძახება OnClick მოვლენაზე მიბმული მეთოდი Register_Click. ის აღიწერება C# კოდში, რომლის 5.4 ლისტინგი მოცემულია ქვემოთ. გავხსნათ Registration.aspx.cs ფაილი და შევიტანოთ შემდეგი კოდი:

```
// — ლისტინგი_5.4 —————
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 5.10 ნახაზზე.

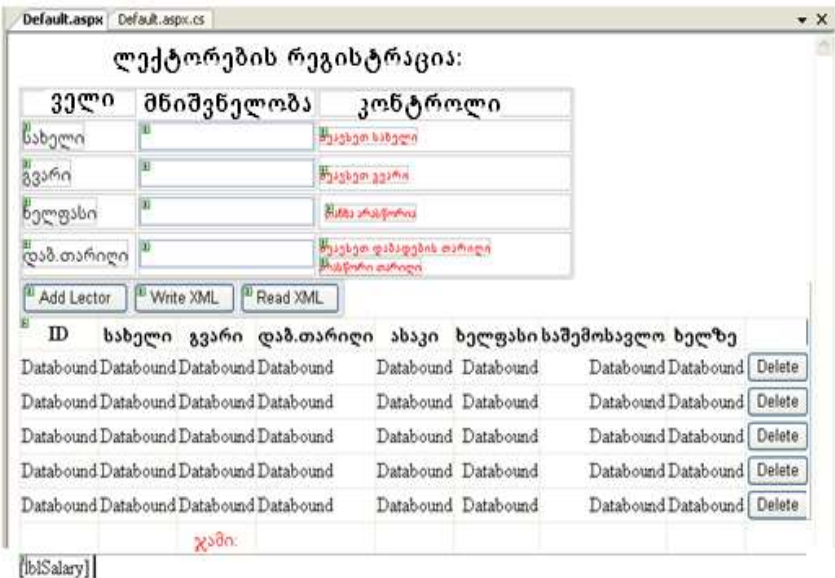


ნახ.5.10, შედეგი

5.4. DataSet / GridView ობიექტებთან მუშაობა და XML ფაილი

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.

ამოცანა-5.2. Visual Studio .NET გარემოში ავაგოთ Web-პროექტი ლექტორთა სარეგისტრაციო მონაცემების შესატანად. განვახორციელოთ მონაცემთა ვიზუალური და ავტომატური კონტროლის საშუალებების გამოყენება. Add Lector-ლილაკით შეტანილი მონაცემები აისახოს GridView ცხრილში უნიკალური ID-ს მქონე სტრიქონის სახით, მომზადდეს სარეგისტრაციო ცხრილი ახალი ინფორმაციის შესატანად (ნახ.5.11). ავტომატური კონტროლისთვის გამოყენებულ იქნეს ToolBox->Validation;



ნახ.5.11

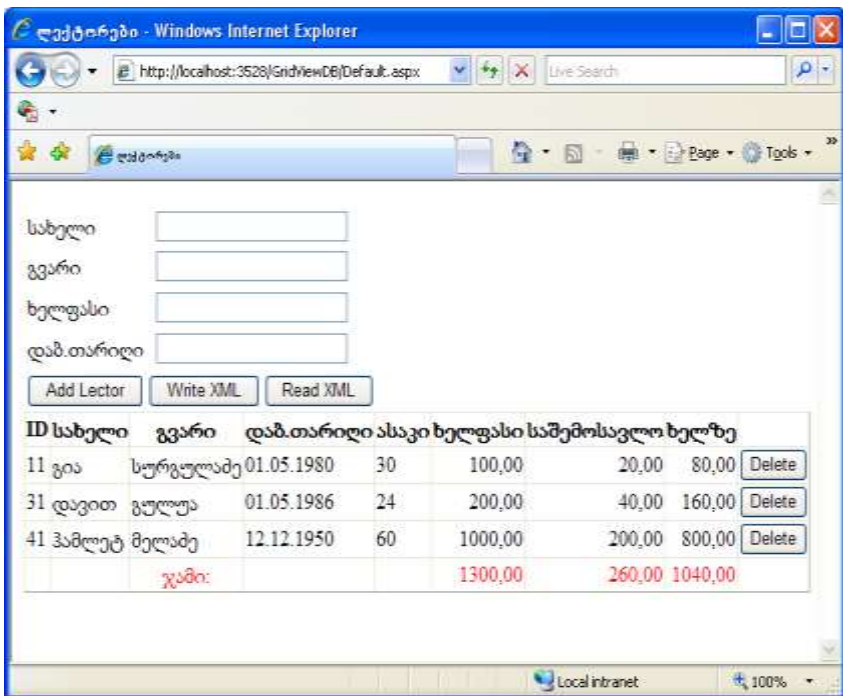
პროგრამული ინჟინერის საფუძვლები

- 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.

5.12 ნახაზზე ნაჩვენებია Web-გვერდის მაკეტი ბრაუზერში.



ნახ.5.12

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_5.5 — 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_5.6 — 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_5.7 — 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_5.8 — 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_5.9 — 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>
```

VI თავი. ინსტრუქციები სისტემის საპილოტო ვერსიისა და საკურსო პროექტის გასაფორმებლად

6.1. აპლიკაციის საპილოტო ვერსიის ტესტირება და სადემონსტრაციოდ მომზადება

პროგრამული აპლიკაციის საპილოტო ვერსია არის ასაგები რეალური სისტემის სადემონსტრაციოდ, „სათამაშო“ ვარიანტი, რომელშიც კარგად ჩანს საპრობლემო სფეროს ობიექტის ავტომატიზაციის ამოცანების სანიმუშო მაგალითები. აქ ალგორითმულად და პროგრამულად რეალიზებულია დეველოპერების მიერ კომპიუტერული სისტემის მოდელი, თავისი მონაცემებით და მეთოდებით (ფუნქციებით).

ტესტირება მოიცავს ჩვენ მიერ აწყობილი სისტემის ცალკეული ამოცანების შესრულებაზე გაშვებას. საჭიროა მონაცემთა ბაზაში, ცხრილებში ინფორმაციის შეტანა-გამოტანისა და მონაცემთა კორექტირების პროცედურების შემოწმება, აგრეთვე მენიუს, დილაგების 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.

გამოყენებული ლიტერატურა:

1. სურგულაძე გ., თურქია ე. პროგრამული სისტემების მენეჯმენტის საფუძვლები. ISBN 978-9941-20-651-1. სტუ. „ტექნიკ.უნივ.“, თბ., 2016. -350 გვ. https://gtu.ge/book/gia_sueguladze/GiaSurg1_ProgSysManag.pdf
2. ჩოგოვაძე გ., ფრანგიშვილი ა., სურგულაძე გ., მართვის საინფორმაციო სისტემების დაპროგრამების ჰიბრიდული ტექნოლოგიები და მონაცემთა მენეჯმენტი. ISBN 978-9941-20-790-7. სტუ. „ტექნიკ.უნივერსიტეტი“, თბ., 2017. -1001 გვ. ბიბლ.ინდექსი: 004.42/7. https://gtu.ge/book/monacemta_menejmenti.pdf
3. სურგულაძე გ., ქრისტესიაშვილი ხ., სურგულაძე გიორგი. საწარმოო რესურსების მენეჯმენტის ბიზნეს-პროცესების მოდელირება და კვლევა. ISBN 978-9941-20-557-6. სტუ. „ტექნიკ.უნივერსიტეტი“, თბ., 2015 -216 გვ. ბიბლ. ინდ.: 004.5 / 2. https://gtu.ge/book/gia_sueguladze/GiaSurgBPMN+ERP.pdf
4. სურგულაძე გ., პეტრიაშვილი ლ. ვიზუალური დაპროგრამება (C#.NET, Workflow Foundation.NET). ISBN 978-9941-20-880-5. სტუ. „ტექნიკ.უნივერსიტეტი“, თბ. 2017. 320 გვ. ბიბლ.ინდ. 004.42(02) / 8. https://gtu.ge/book/gia_sueguladze/GiaSurgC_andWorkflow.pdf
5. სურგულაძე გ., პეტრიაშვილი ლ. ვიზუალური დაპროგრამება C# ენის ბაზაზე ინფორმაციულ სისტემებისათვის (Visual StudioNET 2019 პლატფორმაზე). ISBN 978-9941-8-1708-3. სტუ. „IT-კონსალტინგ ცენტრი“. თბ., 2019. B5, -200 გვ. ბიბლ.ინდექსი 004.43(02)/14. https://gtu.ge/book/GiaSurg_Csh_IS_2019.pdf
6. Booch G., Jacobson I., rambaugh J. (1996). Unified Modeling Language for Object-Oriented Development. Rational Software Corporation, Santa Clara,
7. სურგულაძე გ. დაპროგრამების მეთოდები და ინსტრუმენტები (UML, MsVisio, C++). სტუ. თბილისი. 2007. <http://www.gtu.edu.ge/katedrebi/kat94/pdf/BC++B-22.pdf>
8. გოგიჩაიშვილი გ., სუხიაშვილი თ. სისტემების ობიექტ-ორიენტირებული ანალიზი და დაპროექტება. სტუ. თბ., 2012. http://gtu.ge/books/suxi_gogichaishvili.pdf

9. სამხარაძე რ., გაჩეჩილაძე ლ. SQL სერვერი. სტუ. „ტექნიკ. უნივერსიტეტი“, 2016. https://gtu.ge/book/SQL_Server1_1.pdf
10. სურგულაძე გ., პეტრიაშვილი ლ. მონაცემთა მენეჯმენტის თანამედროვე ტექნოლოგიები (Oracle, MySQL, MongoDB, Hadoop). ISBN 978-9941-27-176-2. სტუ. „IT-კონსალტინგ ცენტრი“. თბ., 2017. -202 გვ. https://gtu.ge/book/PetriSurgu_DataManagmTechn.pdf
11. სურგულაძე გ., თოფურია ნ., გავარდაშვილი ა. შავი ზღვის ეკოლოგიური მონიტორინგისა და კვლევის საინფორმაციო სისტემა. ISBN 978-9941-8-0624-7. სტუ. „IT-კონსალტინგ ცენტრი“. თბ., 2018. B5, -206 გვ. https://gtu.ge/book/Surgul_EcoBlackSea.pdf
12. Refactoring and Reengineering: Why Software Maintenance Is Important. <https://mintymint.net/blog/tech/reengineering-and-refactoring-why-software-maintenance-is-important/> (25.01.22)
13. სურგულაძე გ., ბულია ი., თურქია ე. Web-აპლიკაციების დამუშავება მონაცემთა ბაზების საფუძველზე (ADO.NET, ASP.NET, C#). სახელმძღვ., სტუ, თბ., 2009.
14. სურგულაძე გ., ბულია ი., თურქია ე. Web-აპლიკაციების აგება ASP.NET & C# პაკეტებით .NET პლატფორმაზე. დამხმ.სახ., ლაბ.პრაქტიკუმი. სტუ, თბ., 2009. http://www.gtu.ge/books/GiaSurg_ASP_Lab.pdf
15. სურგულაძე გ., ბულია ი. კორპორაციულ Web აპლიკაციათა ინტეგრაცია და დაპროექტება. მონოგრ., სტუ. თბ., 2012. ელ-ვერსია: http://www.gtu.ge/books/GiaSurg_Book_2012.pdf

/საკურსო პროექტის სატიტულო გვერდი/

საქართველოს ტექნიკური უნივერსიტეტი

ინფორმატიკისა და მართვის სისტემების ფაკულტეტი

პროგრამული ინჟინერიის დეპარტამენტი

საკურსო პროექტი

დისციპლინაში: „პროგრამული ინჟინერიის საფუძვლები“

ჯგ. N: 108 . . .

სტუდენტ(ებ):

თემა: „საკურსო პროექტის სათაური“

ხელმძღვანელი:

თარიღი

თბილისი - 2022/2023

საკურსო პროექტის ანგარიშის სარჩევი

1. საპრობლემო სფეროს ობიექტის შერჩევა; სამუშაო გუნდის ფორმირება (დასაშვებია 1 -:- 4 სტუდენტი);
2. ასაგები კომპიუტერული სისტემის ფუნქციონალური და არაფუნქციონალური მოთხოვნები: როლების და მათი ქმედებების UML დიაგრამები (UseCase, Activity) /VS .NET 2015 ან სხვ.);
3. ელექტრონული დოკუმენტების ფორმები: საწყისი, ნორმატიული, ოპერატიული და გამომავალი;
4. სისტემის მონაცემთა ბაზის სტრუქტურა ER დიაგრამა;
5. სისტემის მონაცემთა ბაზა (Ms SQL Server), ცხრილები (Tables) ჩანაწერებით;
6. მომხმარებელთა ინტერფეისები (სამუშაო ფორმები), MsVisual Studio.NET ინტეგრირებულ გარემოში (Windows Forms App ან ASP.NET Web). SQL Server ბაზის და C# -ენის საფუძველზე;
7. საპრობლემო სფეროს ფუნქციონალური ამოცანის (ჯგუფური პროექტის დროს – ამოცანების) გადაწყვეტის პროცესის ავტომატიზაცია (დაპროგრამება და ტესტირება);
8. მომხმარებელთა ინტერფეისებში MsSQL Server ბაზიდან მონაცემთა ძებნა და ამორჩევა, შესაბამისი C# კოდებით;
9. სისტემის დემოვერსია და საპრეზენტაციო სლაიდები.

**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: ელექტრონული 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: ელექტრონული შეკვეთა ბინაზე მიტანით
 - 4.4. ობიექტი: დღიური/თვიური/წლიური ეკონომიკური 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: ავტომატური 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.

საკურსო პროექტის ნიმუში



პროგრამული ინჟინერიის დეპარტამენტი

საკურსო პროექტი

აკად.კურსში: „პროგრამული ინჟინერიის საფუძვლები“
თემა:

„ვაკანსიების მოძიების ავტომატიზებული სისტემა
საგანმანათლებლო დაწესებულებაში“

ჯგ.: 108851

სტუდენტი: დავით კოსტავა,

ვახტანგ ფეზუაშვილი

ხელმძღვანელი:

პროფ. გ. სურგულაძე

თბილისი - 2021

რეზიუმე

წარმოდგენილია საგანმანათლებლო სფეროში მასწავლებელთა სამუშაო ადგილების (ვაკანსიების) მოძიების პროგრამული აპლიკაციის შექმნის ამოცანა. საპრობლემო ობიექტის ანალიზის საფუძველზე ჩამოყალიბებულია ასაგები პროგრამული სისტემის ფუნქციური მოთხოვნილებები და მისი მხარდამჭერი კომპიუტერული აპლიკაციის ინფრასტრუქტურა. გამოყენებულია უნიფიცირებული მოდელირების ენის მეთოდოლოგიის UseCase და Activity დიაგრამები. ისინი აგებულია Visual Studio.NET 2015 -ის პლატფორმაზე. განსაზღვრულია მონაცემთა ბაზის კონცეპტუალური და ლოგიკური მოდელები, რეალიზაცია კი განხორციელებულია Ms SQL Server-ის საფუძველზე.

Desktop და Web აპლიკაციები შექმნილია ვიზუალური დაპროგრამების C# ენის გამოყენებით Visual Studio. NET-ის გამოყენებით.

Windows Forms Application პროექტში მოცემულია მომხმარებელთა ინტერფეისის თერთმეტი ფორმა, მათ შორის მთავარი გვერდი და ვაკანსიების ჩამონათვალი. მთავარ გვერდზე ჩაშენებულია რამდენიმე ეტაპიანი საძიებო სისტემა.

ასევე მთავარ ფორმაზე განთავსებული Label-ების საშუალებით ნაწილდება შესაბამის ავტორიზაციის პანელზე შესვლა, ვაკანსიები გამოიცემა DataGridView საშუალებით SQL მონაცემთა ბაზიდან Select ბრძანების შედეგად.

1. საპრობლემო სფეროს ანალიზი და სისტემის მოთხოვნილების განსაზღვრა

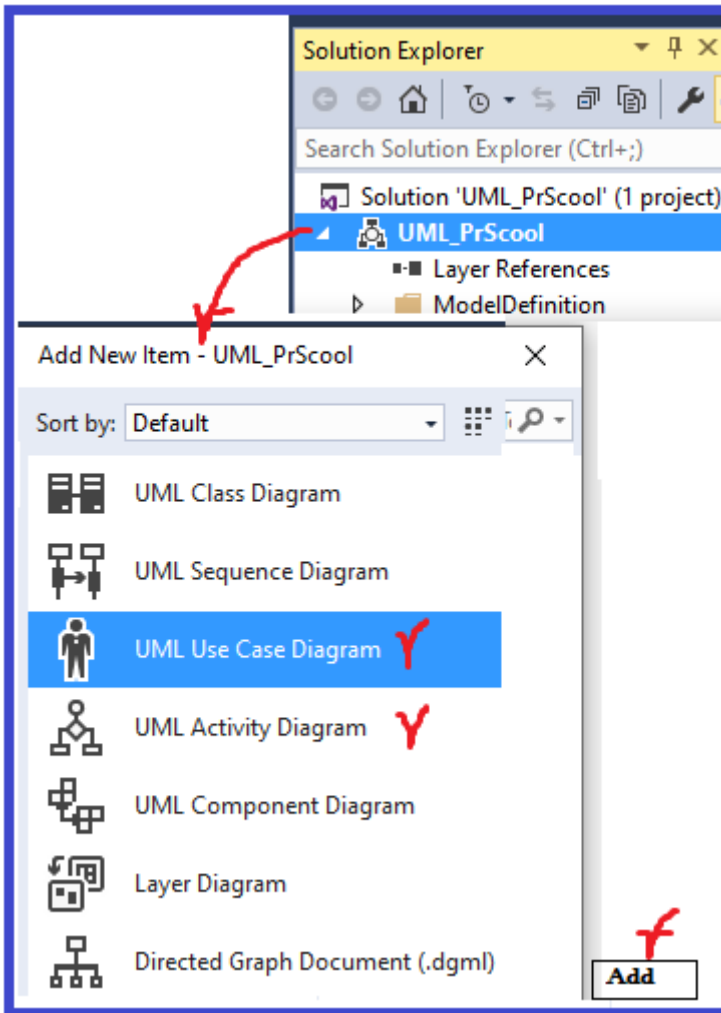
1.1. სისტემის მომხმარებელთა როლების და ფუნქციების დიაგრამების აგება

უნიფიცირებული მოდელირების ენა (UML) ეფუძნება სისტემების ობიექტ-ორიენტირებულ მიდგომას. ჩვენ ჩავატარეთ საპრობლემო სფეროს, ანუ საგანმანათლებლო დაწესებულების (მაგალითად, საჯარო სკოლა) ანალიზი და UML-ის UseCase დიაგრამის დახმარებით აღვწერეთ მომხმარებლის და სკოლის HR (კადრების განყოფილების) ბიზნეს ფუნქციები.

პროექტში ჩვენ ვამახვილებთ ყურადღებას ვაკანსიების ამოცანაზე, კერძოდ ვაკანსიების სიის (ჩამონათვლის) ნახვის შესაძლებლობის სარეალიზაციოდ, მისი დეტალურად გაცნობისა და გამოყენების მიზნით.

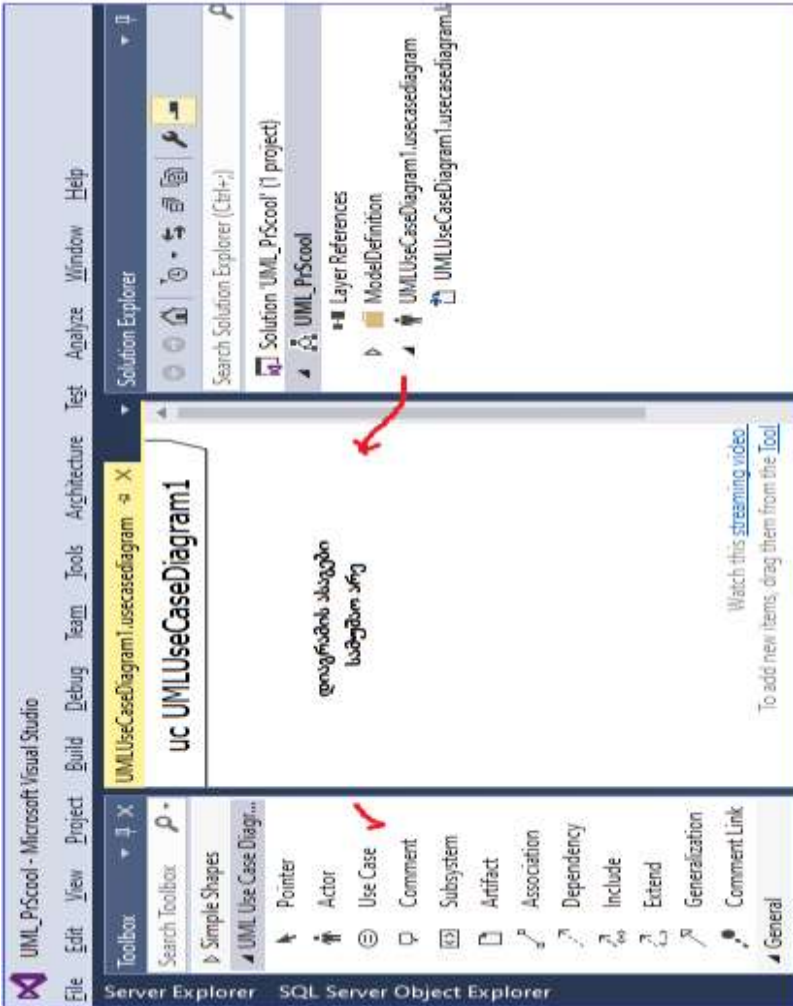
ამისათვის კი ეტაპობრივად, რამდენიმე ფაზის გავლა საჭირო: რეგისტრაცია, ავტორიზაცია, პირადი ინფორმაციის დამატება საკუთარ გვერდზე, სკოლის HR-სთვის კი ავტორიზაციის და ვაკანსიის დამატება, ინფორმაციის განახლების შესაძლებლობით. შემდეგ აპლიკანტების (მაძიებლების) ინფორმაციის ანალიზი და ა.შ.

1-ელ ნახაზზე მოცემულია Visual Studio.NET 2015 პლატფორმაზე UML-ის დიაგრამების აგების ინსტრუმენტი. Solution Explorer-ში ახალ პროექტს ვუმატებთ UseCase და Activity დიაგრამებს,



ნახ.1. Visual Studio.NET 2015-ის სამუშაო გარემო

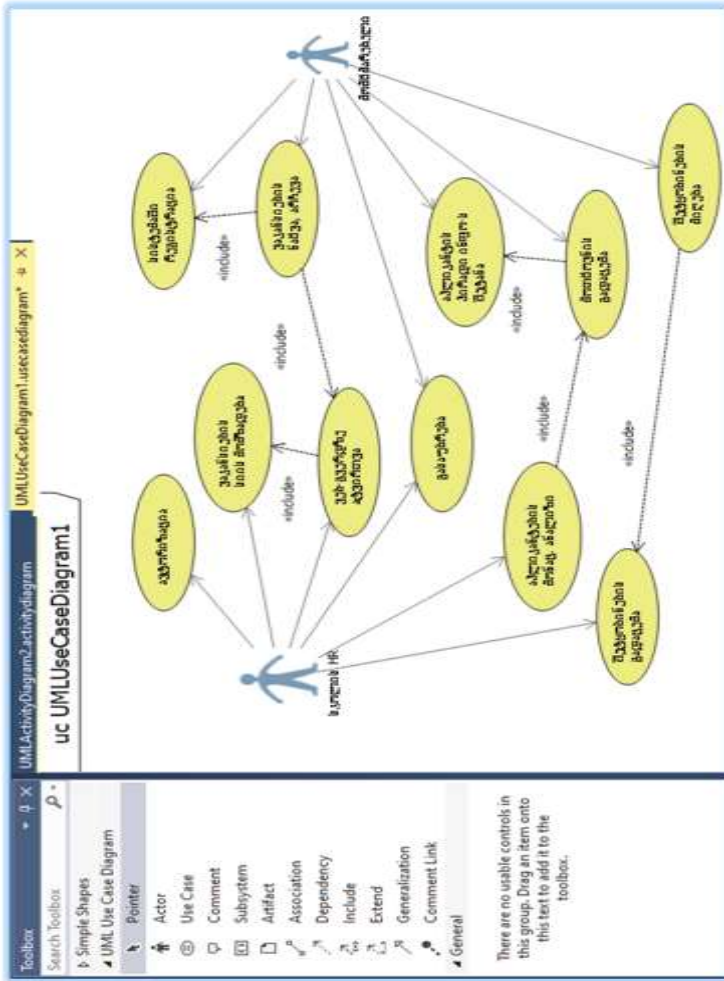
UML Use Case Diagram -ის არჩევის და Add ღილაკის ამოქმედებით მივიღეთ საწყისი სამუშაო სივრცე (ნახ.2).



ნახ.2. UseCase დიაგრამის აგების ინსტრუმენტი

პროგრამული ინჟინერიის საფუძვლები

ჩვენი საპრობლემო სფეროს გამოყენებით შემთხვევათა (პრეცედენტების) UseCase დიაგრამა „სკოლის-HR“ და „მომხმარებლით“ (აპლიკანტი) მოცემულია მე-3 ნახაზზე.



ნახ.3. საპრობლემო სფეროს (სკოლის) UseCase დიაგრამის ფრაგმენტი: „როლები და ფუნქციები“

ნახაზზე „კაცუნებს“ შეესაბამება მომავალი კომპიუტერული სისტემის მომხმარებლები ანუ როლები. პირობითად გვაქვს ორი როლი:

„სკოლის-HR“ (კადრების განყოფილების სპეციალისტი) და ვაკანსიის მაძიებელი „მომხმარებელი“ (ანუ აპლიკანტი).

აპლიკანტი შეიძლება იყოს 1-ზე მეტი, ამიტომ კადრების განყოფილებაში (სადაც შემოდის გარე დოკუმენტაცია სკოლის კანცელარიიდან) გროვდება აპლიკანტების მასალა (ძირითადად, ონლაინში მიღებული, რომლებიც განთავსდება სკოლის მონაცემთა ბაზაში და ხელმისაწვდომია სკოლის დირექტორის, კადრების განყოფილებს და სხვ. ხელმძღვანელებისთვის, რომლებიც მონაწილეობენ კადრების შერჩევის კომისიაში).

ნახაზზე „ოვალები“ შეესაბამება ფუნქციურ ამოცანებს, რომლებსაც ასრულებს ერთი როლი (ინდივიდუალურად) ან რამდენიმე როლი (მაგალითად, ოვალი „გასაუბრება“).

მე-3 ნახაზი არის სავარაუდო მთლიანი სისტემის მხოლოდ ფრაგმენტი (ჩვენი პროექტისათვის). მთლიან ვერსიაში დაემატება, მაგალითად, სკოლის დირექტორი, სასწავლო ნაწილი, ბუღალტერი, მასწავლებლები და სხვ. მათ მიემაგრება შესაბამისი ოვალები (ანუ ფუნქციები, რასაც ისინი ასრულებენ საჯარო სკოლის დამტკიცებული დებულების შესაბამისად).

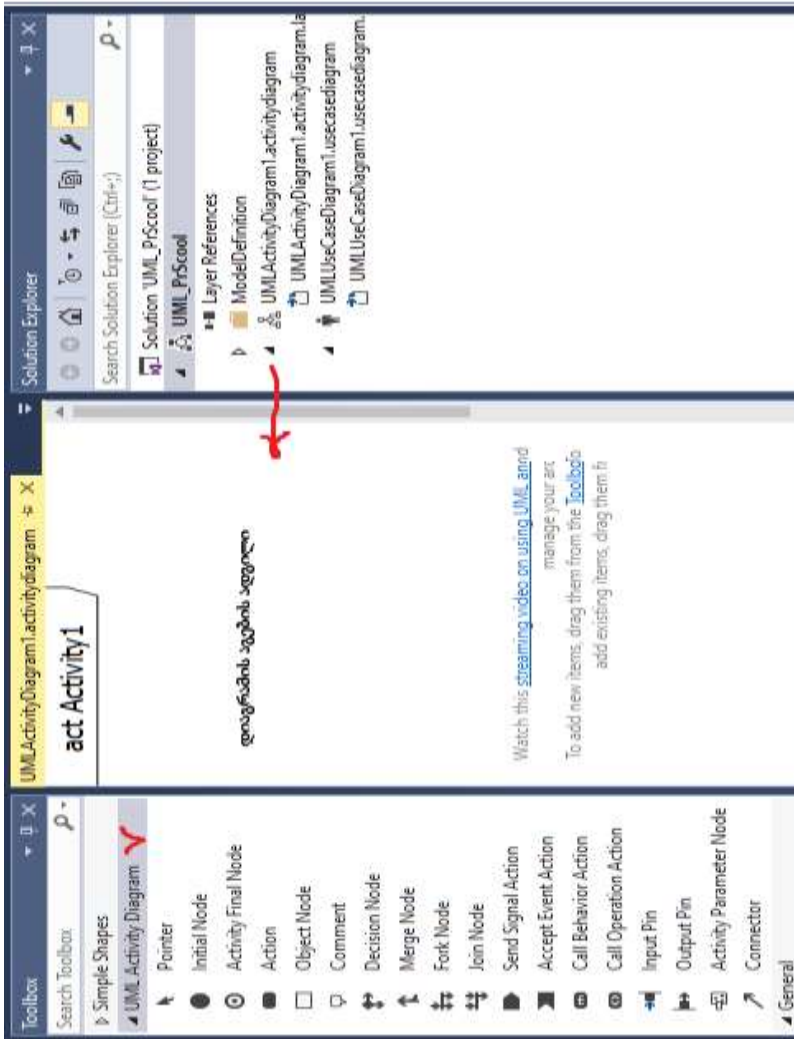
შემდეგი ეტაპია ოვალებში ჩაწერილი ფუნქციების „მეთოდებად“ წარმოდგენა (C#-ის ობიექტ-ორიენტირებული პროგრამირების ტერმინებით: კლასი, კლასის ატრიბუტი, კლასის მეთოდი და ა.შ.).

თითოეული მეთოდი არის „ბიზნეს-ფუნქცია“, რომელსაც როლი ასრულებს. ჩვენი მიზანია მისი დაპროგრამება და კომპიუტერულ სისტემაში ატვირთვა.

ამისათვის საჭიროა ამ ბიზნეს ფუნქციებში კარგად გარკვევა და შესაბამისი პროცედურების ალგორითმების აგება. ეს ხორციელდება UML-ის Activity დიაგრამით.

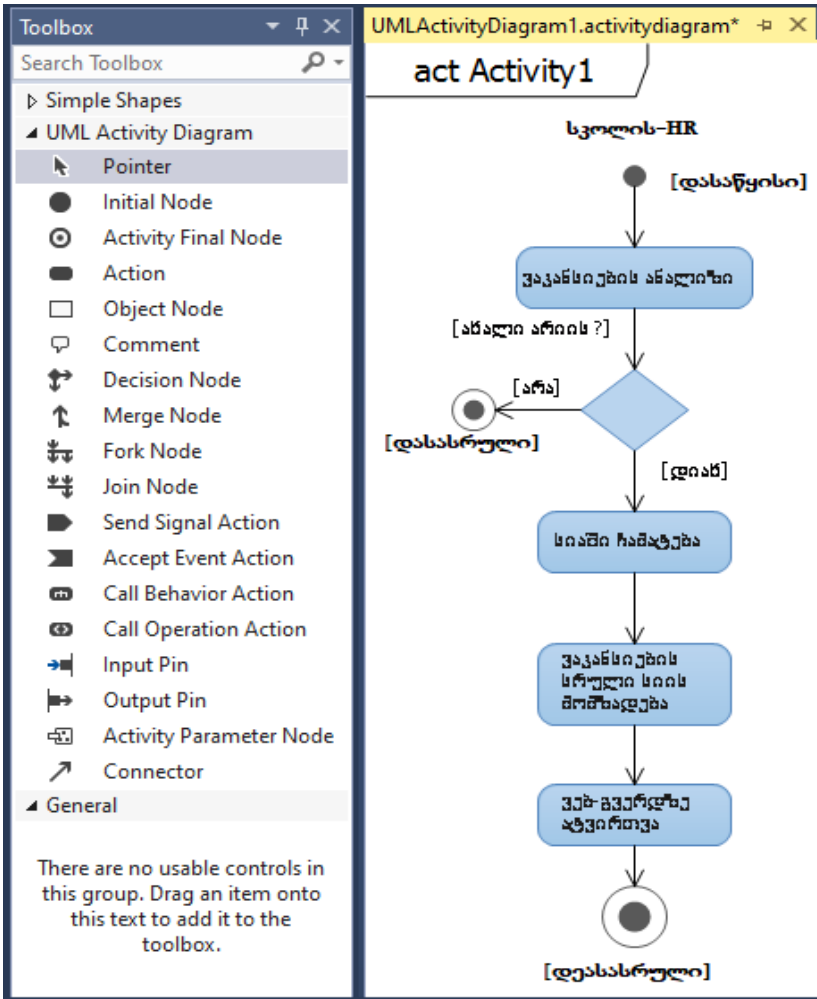
ჩვენი პროექტისთვის Visual Studio.NET 2015 -ში (ნახ.1) გადავდივართ UML Activity Diagram -ზე (ნახ.4).

აქ ჩანს Activity-D ინსტრუმენტები პანელი და სამუშაო არე.



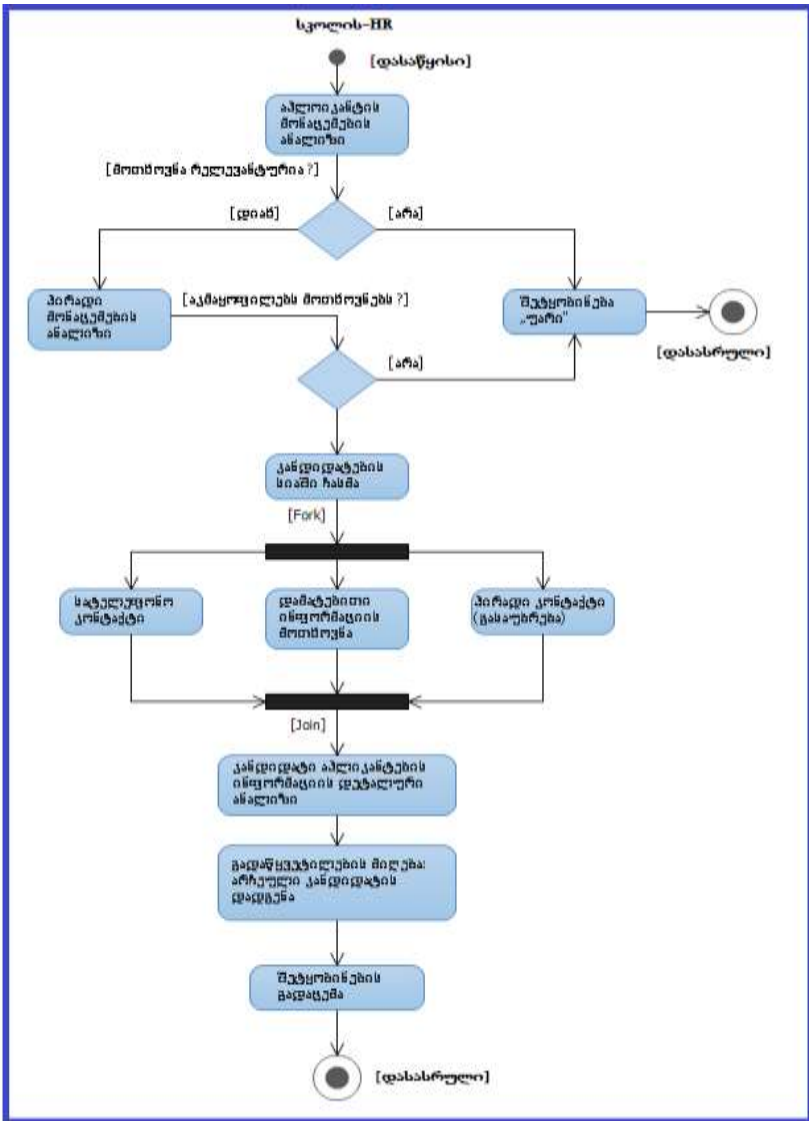
ნახ.4. UML Activity Diagram სამუშაო გარემო Visual Studio.NET 2015-ში

➤ „სკოლის -HR” სპეციალისტის ორი დიაგრამა მოცემულია მე-5 და მე-6 ნახაზებზე. სხვა ფუნქციებიც ამგვარად აღიწერება.



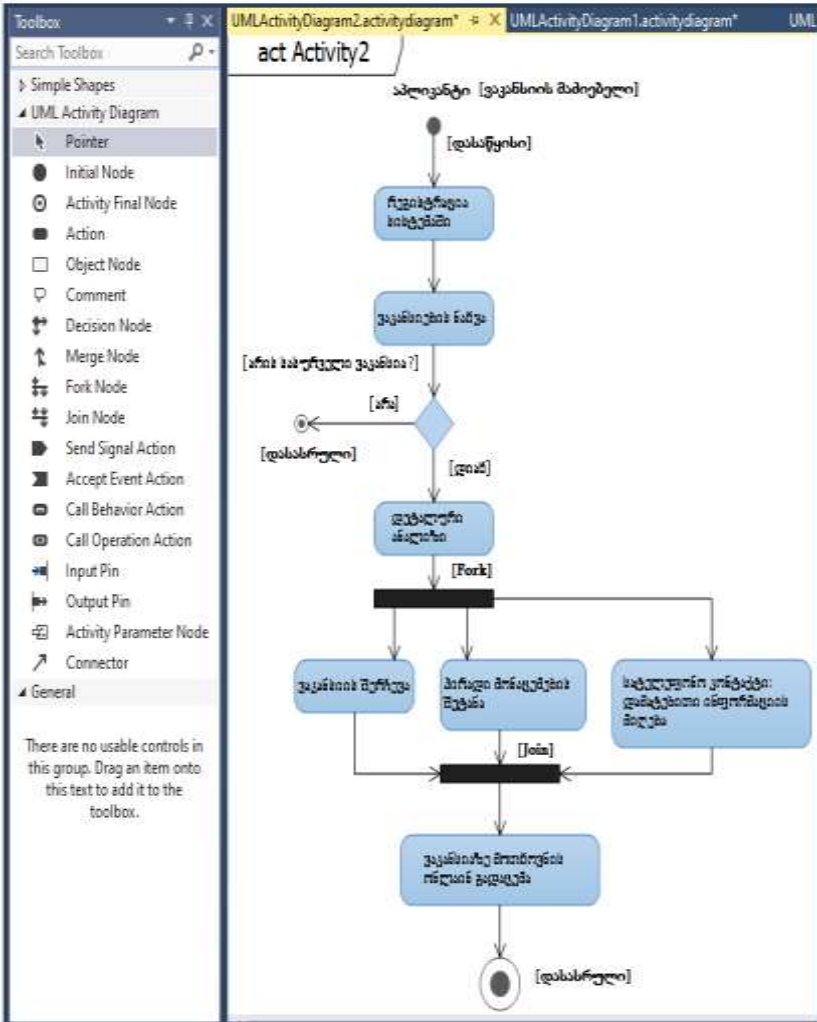
ნახ.5. ვაკანსიების სიის მოშვადება და ვებ-გვერდზე ატვირთვა

პროგრამული ინჟინერის საფუძვლები



ნახ.6. აპლიკანტის მონაცემების ანალიზი და გადაწყვეტილების მიღება

➤ სისტემის „მომხმარებლის“ (ანუ მაძიებელი აპლიკანტის) აქტიურობის დიაგრამა მოცემულია მე-7 ნახაზზე.



ნახ.7. ვაკანსიის მაძიებელი აპლიკანტის Activity დიაგრამა

1.2. მონაცემთა ბაზის დაპროექტება კადრების განყოფილებისთვის

დასაპროგრამებელი სისტემის ფუნქციონალური მოთხოვნების განსაზღვრა განხორციელდა UML-ის გამოყენებით შემთხვევათა (UseCase) და აქტიურობის (Activity) დიაგრამების აგების საფუძველზე.

ჩვენთვის ცხადი გახდა გადასაწყვეტი ამოცანების სია და მათი შესაბამისი ბიზნეს-ფუნქციები (ანუ ალგორითმები, რომლებისთვისაც დაიწერება პროგრამები).

ეს პროგრამები მოითხოვს მონაცემთა ბაზას, რომელსაც ამ პარაგრაფში წარმოვადგენთ.

ვიყენებთ Ms SQL Server მონაცემთა ბაზის მართვის სისტემას და შემდეგ მას მივამაგრებთ Visual Studio.NET Framework პლატფორმაზე დაწერილ მომხმარებლის ინტერფეისის პროგრამულ სისტემას.

ამ ეტაპზე ჩვენ გავანალიზეთ „სკოლის-HR“ სპეციალისტის და „მომხმარებელ-აპლიკანტების“ (პირადი მონაცემები) ამოცანებისთვის საჭირო ინფორმაცია, რომელიც ჩაიდება მონაცემთა ბაზაში ცხრილების სახით.

ავაგეთ სკოლ(ებ)ის საპრობლემო სფეროს კონცეპტუალური მოდელი (Entity Relationship დიაგრამა), რომელც მოცემულია მე-8 ნახაზზე.

ჩვენი პროექტის კონკრეტული ამოცანებისათვის შეიძლება ეს დიაგრამა დაიხვეწოს და შემდეგ მის საფუძველზე ავაგოთ მონაცემთა ბაზის შესაბამისი ლოგიკურ სტრუქტურა (ცხრილები ატრიბუტებით, მონაცემთა ტიპებით და ველის სიგრძეებით).

ნახაზზე ჩანს ცხრილები (Tables) თავისი ატრიბუტებით, აგრეთვე ცხრილთაშორისი კავშირები (1:n, m:n).

ჩვენი მონაცემთა ბაზის რამდენიმე ცხრილში (აქტიური ამოცანებისთვის) შევიტანეთ კონკრეტული სტრიქონები (2-3 ჩანაწერი თითოეულში). მისი სრული შევსება განხორციელდება მომხმარებლის ინტერფეისიდან (დამუშავების შემდეგ).

2. სისტემის დესკტოპ-აპლიკაცია

Visual Studio.NET Framework 2022 (ან სხვა ვერსიაში) უნდა ავაგოთ Windows Forms Application პროექტი C#, ADO,NET და SQL Server პაკეტების გამოყენებით.

პროგრამის მეშვეობით ხდება ვაკანსიის დათვალიერება (ნახ.9). ასევე შესაბამის ვაკანსიის დეტალურად ნახვა კონკრეტულ სვეტზე დაწკაპუნების შესაბამისად.

Combobox-ების საშუალებით გამოყენებულია 5-ეტაპიანი საძიებო სისტემა ეს გვეხმარება ჩვენთვის შესაბამისი ვაკანსიის მოძიებაში. ფორმაზე არის შემდეგი სახის ინფორმაცია:

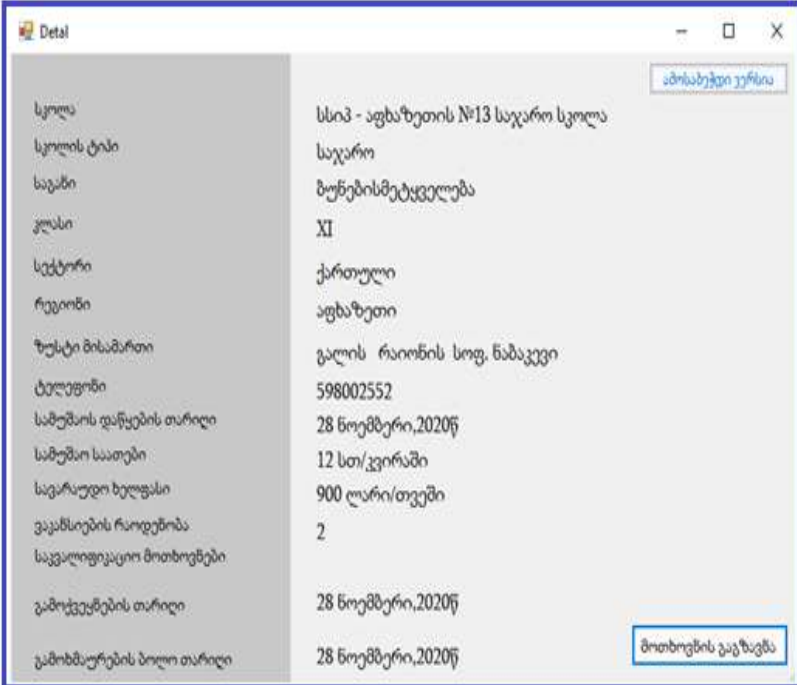
ID, თანამდებობა, საგანი, კლასი, სკოლა, მდებარეობა, გამოქვეყნების დრო და ჩვენების ბოლო ვადა.

ID	თანამდებობა	საგანი	კლასი	სკოლა	მდებარეობა	გამოქვეყნების დრო	ბოლო ვადა
1	მანქანის მძღველი	ფანჯილის მანქანა	V	სსი - აგშლის 303 საჯარო სკოლა	აგშლი	11/20/2020 5:15 PM	12/20/2020 5:15 PM
21	მანქანის მძღველი	მთელი	XII	სსი - აგშლის 304 საჯარო სკოლა	აგშლი	11/20/2020 5:22 PM	12/20/2020 5:22 PM
24	მანქანის მძღველი	ტბოლი და ლოტო	XII	სსი - ქალაქ ჩხეთის 300 საჯარო სკოლა	ჩხეთი	11/5/2020 3:02 PM	11/20/2020 3:02 PM
1007	მანქანის მძღველი	კონკრეტული	XI	სსი - აგშლის 303 საჯარო სკოლა	აგშლი	11/20/2020 2:16 PM	11/20/2020 2:16 PM
1008	მანქანის მძღველი	სტრუქტურული	XI	სსი - აგშლის 303 საჯარო სკოლა	აგშლი	11/20/2020 2:16 PM	11/20/2020 2:16 PM
1009	მანქანის მძღველი	სოლო	V	სსი - აგშლის 303 საჯარო სკოლა	აგშლი	11/20/2020 2:16 PM	11/20/2020 2:16 PM
1010	მანქანის მძღველი	ქიმი	VIII	სსი - აგშლის 303 საჯარო სკოლა	აგშლი	11/20/2020 2:17 PM	11/20/2020 2:16 PM
13	მანქანის მძღველი	სოლო	X	სსი - ქალაქ ჩხეთის 300 საჯარო სკოლა	ჩხეთი	11/4/2020 2:51 PM	12/1/2020 2:50 PM
22	მანქანის მძღველი	ქიმი	XII	სსი - აგშლის 304 საჯარო სკოლა	აგშლი	11/9/2020 1:19 PM	12/9/2020 1:18 PM

ნახ.9. ინტერფეისის ფორმა „მთავარი გვერდი“

პროგრამული ინჟინერიის საფუძვლები

მე-10 ნახაზზე ნაჩვენებია ვაკანსიის დეტალური ინფორმაციის ფორმა,



სკოლა	სსიპ - აფხაზეთის №13 საჯარო სკოლა
სკოლის ტიპი	საჯარო
საგანი	ზუნებისმეტყველება
კლასი	XI
სექტორი	ქართული
რეგიონი	აფხაზეთი
ზუსტი მისამართი	გალის რაიონის სოფ. ნაბაკევი
ტელეფონი	598002552
სამუშაოს დაწყების თარიღი	28 ნოემბერი, 2020წ
სამუშაო საათები	12 სთ/კვირაში
საეარაუღო ხელფასი	900 ლარი/თვეში
ვაკანსიების რაოდენობა	2
საკვალიფიკაციო მოთხოვნები	
გამოქვეყნების თარიღი	28 ნოემბერი, 2020წ
გამომხატურების ბოლო თარიღი	28 ნოემბერი, 2020წ

ნახ.10. ვაკანსიის დეტალური აღწერილობის ფორმა

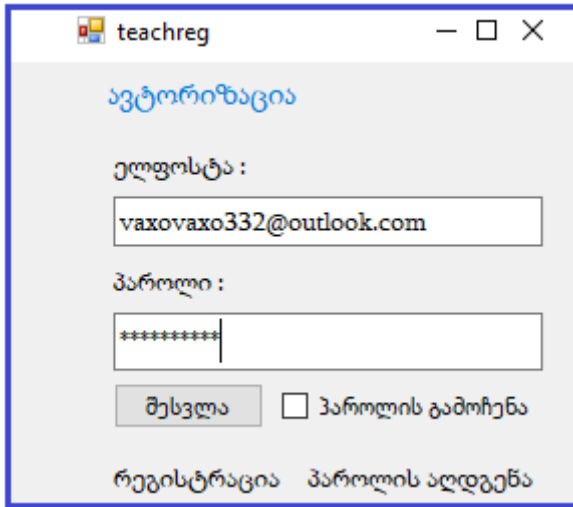
ფორმაზე გამოტანილია სკოლებში ვაკანსიის დეტალური ინფორმაციის აღწერა. ფორმაზე შესასვლელად უნდა ავირჩიოთ მთავარი ფორმიდან შესაბამისი ვაკანსია და მაუსის ერთი კლიკით შევდივართ ფორმა „Detail” -ზე.

ფორმაზე გამოყენებული გვაქვს ორი Button, პირველი „ამოსახულები უკრსია“- რის შედეგადაც პრინტერის საშუალებით ამოვბეჭდავთ ფორმაზე არსებულ ყველა ველს, მეორე „მოთხოვნის გაგზავნა“. ეს ღილაკი აქტიურდება მაშინ თუ მომხმარებელი ავტორიზაციას გადის, რის შედეგადაც დაწკაპუნების შემდეგ აგზავნის მოთხოვნას და მომხმარებლის პირად გვერდზე ემატება.

3. მასწავლებლის ავტორიზაცია/რეგისტრაცია და პაროლის აღდგენა

მასწავლებლების (როგორც მომხმარებლების) მუშაობა სისტემასთან იწყება მათი იდენტიფიკაციის საკითხის შემოწმებით. ანუ მათ სისტემაში შესავლისათვის უნდა გაიარონ ავტორიზაცია.

მთავარი გვერდიდან ირჩევენ პუნქტს „მასწავლებელი მაძიებელი“, რის შემდეგაც გადადიან ავტორიზაციის ფორმაზე (ნახ.11),



ნახ.11. ფორმა „მასწავლებლის ავტორიზაცია“

არსებული ველების სწორად შევსების შემთხვევაში სისტემა მიცემს მას უფლებას საკუთარ გვერდზე შესასვლელად.

მე-12 ნახაზზე ნაჩვენებია ონლაინ რეგისტრაციის ფორმა.

registration

რეგისტრაცია

პირადი ნომერი
13245678912 ატვირთეთ ფოტო

დაბადების თარიღი
Tuesday, February 1, 2000 თქვენი ფოტო

შემოწმება

სახელი
დავით

გვარი
კოსტავა

ელფოსტა
kostava@gmail.com

პაროლი
.....

გაიმეორეთ პაროლი
.....

პაროლის გამოჩენა

სქესი
 მამრობითი მდედრობითი

შენახვა

ნახ.12. ფორმა „რეგისტრაცია“

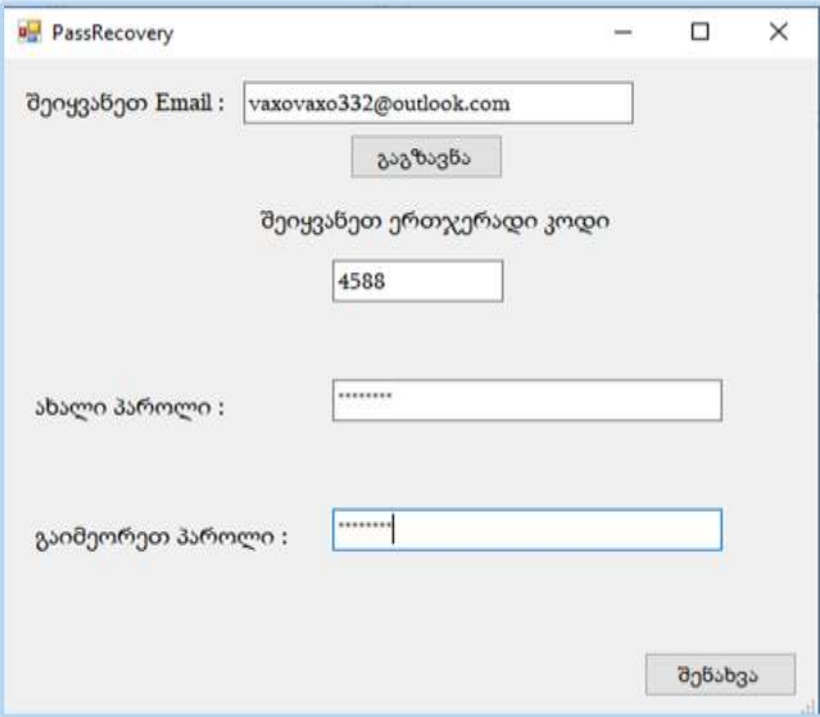
მასწავლებელმა რომ გაიაროს რეგისტრაცია, საჭიროა მან ჩაწეროს პირადი ნომერი და მიუთითოს დაბადების რიცხვი, თვე და წელი. ამის შემდეგ „შემოწმება“ - ლილაკის საშუალებით ხდება ბაზაში მონაცემის გადამოწმება, თუ ასეთი მონაცემი არსებობს, ბაზიდან ავტომატურად მოაქვს მომხმარებლის სახელი და გვარი.

შემდეგ კი გვაქვს სხვა ველების შევსების უფლება, ბოლოს მასწავლებელი რეგისტრირდება სისტემაში.

პროგრამული ინჟინერიის საფუძვლები

ერთ-ერთი მნიშვნელოვანი საკითხი პროგრამული აპლიკაციის აგებისას არის მომხმარებლისთვის დავიწყებული პაროლის აღდგენის ამოცანის დაპროგრამება და სერვისად მიწოდება (ნახ.13).

პირველ Textbox-ში შეგვყავს ის მეილი, რომლითაც რეგისტრირებული ვართ. Button „გაგზავნა“-ის შემდეგ კი მივიღებთ *ერთჯერად კოდს* არსებულ მეილზე და ვაგრძელებთ დანარჩენ პროცედურებს.



ნახ.13. პაროლის აღდგენა

მე-14 ნახაზზე მოცემულია მომხმარებლის პირადი გვერდის ფანჯრის საილუსტრაცია მაგალითი.



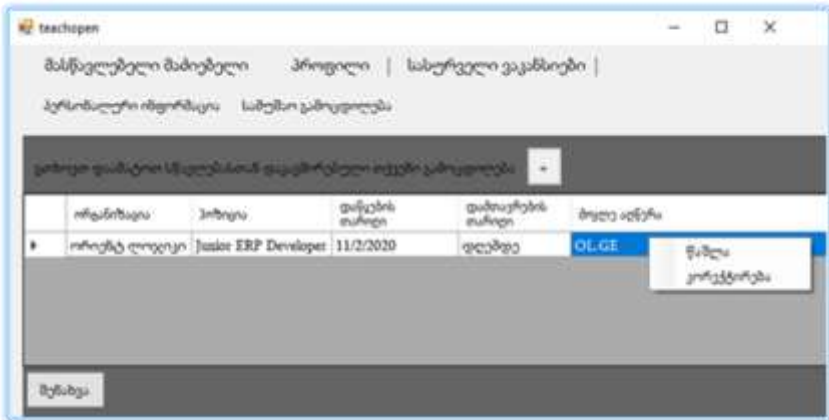
ნახ.14. მომხმარებლის გვერდი

მომხმარებლის გვერდზე გამოტანილია ყველა ის ველი, რომელიც მოიცავს მომხმარებლის პირად ინფორმაციას.

სასურველ ვაკანსიებში გადასვლის დროს DataGridView -ში გამოაქვს ყველა ის ვაკანსია რომელზეც გვაქვს მოთხოვნა გაგზავნილი,

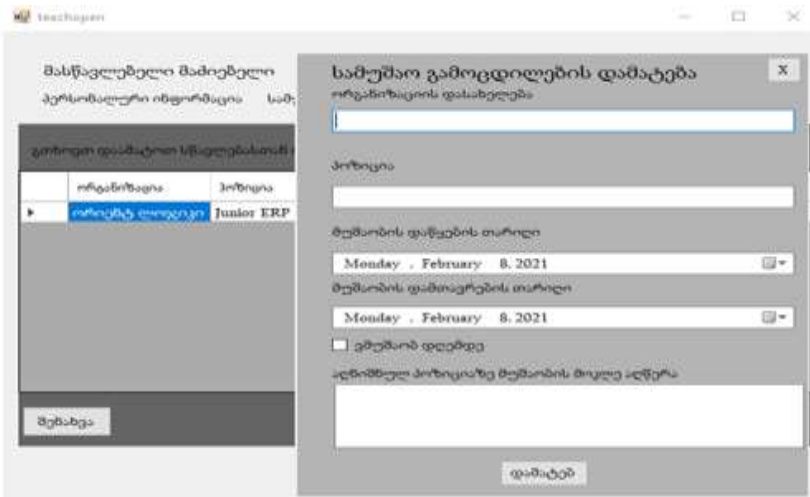
სამუშაო გამოცდილებაში გადასვლისას კი DatagridView -ში გამოიტანს სტრიქონებს, რომელიც ასახავს ჩვენი ყოფილი სამსახურების ჩამონათვალს (შრომის წიგნაკის მსგავსად) (ნახ.15).

პროგრამული ინჟინერიის საფუძვლები



ნახ.15. სამუშაო გამოცდილება

ფორმაზე გამოტანილია სამუშაო გამოცდილების ველები. აქ შესაძლებელია ახალი სამუშაო გამოცდილების სტრიქონების ჩამატებაც (ნახ.16).

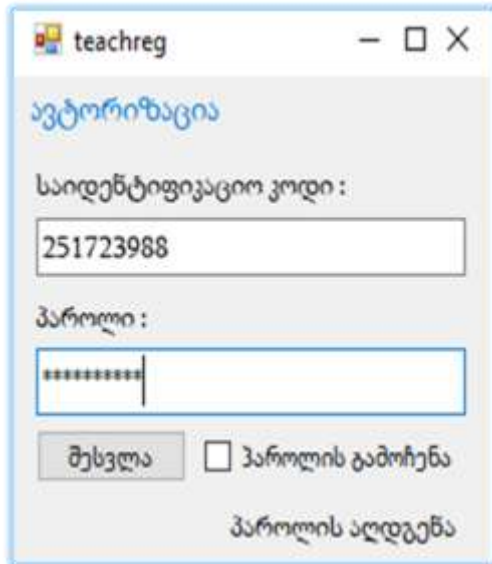


ნახ.16. სამუშაო გამოცდილების ახალი სტრიქონების დამატება

პროგრამული ინჟინერიის საფუძვლები

სისტემაში *ახალი ვაკანსიის* დასამატებლად საჭიროა ჯერ ავტორიზაციის გავლა (კადრების განყოფილების ინსპექტორის მიერ. სისტემა ამოწმებს აქვს თუ არა უფლება ამ პირს). წარმატებით გავლის შემთხვევაში კი ამატებს ვაკანსიას.

მე-17 ნახაზზე ნაჩვენებია ავტორიზაციის გავლის ფანჯარა,



ნახ.17. ვაკანსიის დასამატებელია
ავტორიზაცია

იმისთვის, რომ სკოლამ დაამატოს ვაკანსია, მთავარ გვერდზე Label „ვაკანსიის დამატება“ დაწკაპუნებით, უფლებამოსილი პირი (ავტორიზაციის წარმატებით გავლის შემდეგ) შეიყვანს კონკრეტული სტრიქონის მნიშვნელობებს (ნახ.18).

პროგრამული ინჟინერიის საფუძვლები

Director

სკოლა: სპი- აგზულის 313 საჯარო სკოლა

ბიზნისი: 251723968

სკოლის ტიპი: საჯარო

თამაშების :	მანქანების მართვა	საბუღალტრო თარიღი :	Monday , February 8, 2021
საბი :	მათემატიკა	საბუღალტრო საათები :	10
კლასი :	11	საბუღალტრო ზღვარი :	1000
სექტორი :	კარული	ვანების რაოდენობა :	1
ტელეფონი :	598002255	კომპიუტერის სილი თარიღი :	Friday , February 12, 2021

საგარეო საბუღალტრო
საგარეო საბუღალტრო ა
ჯგუფის მანქანების
მართვაზე

დატება

ნახ.18. ვაკანსიის დამატება

ავტორიზაციის გავლის შემდეგ შევდივართ ვაკანსიის დამატების ფორმაზე და ვავსებთ შესაბამის ველებს, „დამატება“ - ღილაკის ამოქმედების შემდეგ კი მთავარ გვერდზე დაემატება ახალი ვაკანსია.

3. C# კოდები Desktop აპლიკაციისთვის (სრული პაკეტი)

```
// - მთავარი ფორმა -----
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.Sql;
using System.Data.SqlClient;

namespace TeacherJob
{
    public partial class Form1 : Form
    {
        SqlConnection con = new SqlConnection(@"Data
Source=VPEZUASHVILI;Initial Catalog=teacherJob;Integrated Security=True");
        SqlCommand cmd;
        SqlDataAdapter adapter;
        DataTable dtb1;
        DataTable dtb2;
        DataTable dtb3;
        DataTable dtb4;
        DataTable dtb5;
        DataTable dtb6;
        DataSet set;

        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            TeacherJob();
            posst();
            local1();
            Clas1();
        }
    }
}
```

```
schol1();
Subj1();

Time();
comboBox1.DropDownWidth = DropDownWidth(comboBox1);
comboBox2.DropDownWidth = DropDownWidth(comboBox2);
ColumnEdit();
label10.Text = label10.Text + " " + dataGridView1.Rows.Count + " " +
"ვაკანსია";

}
public void local1()
{
    SqlCommand cmd = con.CreateCommand();
    cmd.CommandType = CommandType.Text;
    cmd.CommandText = "select * from Local1";
    dtb1 = new DataTable();
    adapter = new SqlDataAdapter();
    adapter.SelectCommand = cmd;
    set = new DataSet();
    adapter.Fill(dtb1);
    foreach (DataRow dr in dtb1.Rows)
    {
        comboBox5.Items.Add(dr["Location"]);
    }
}

}
public void Clas1()
{
    SqlCommand cmd = con.CreateCommand();
    cmd.CommandType = CommandType.Text;
    cmd.CommandText = "select * from Clas1";
    dtb2 = new DataTable();
    adapter = new SqlDataAdapter();
    adapter.SelectCommand = cmd;
    set = new DataSet();
    adapter.Fill(dtb2);
    foreach (DataRow dr in dtb2.Rows)
    {
        comboBox3.Items.Add(dr["Class"]);
    }
}
```

```
}  
public void posst()  
{  
    SqlCommand cmd = con.CreateCommand();  
    cmd.CommandType = CommandType.Text;  
    cmd.CommandText = "select * from Posst";  
    dtb3 = new DataTable();  
    adapter = new SqlDataAdapter();  
    adapter.SelectCommand = cmd;  
    set = new DataSet();  
    adapter.Fill(dtb3);  
    foreach (DataRow dr in dtb3.Rows)  
    {  
        comboBox1.Items.Add(dr["Possition"]);  
    }  
}  
  
public void schol1()  
{  
    SqlCommand cmd = con.CreateCommand();  
    cmd.CommandType = CommandType.Text;  
    cmd.CommandText = "select * from Schol1";  
    dtb4 = new DataTable();  
    adapter = new SqlDataAdapter();  
    adapter.SelectCommand = cmd;  
    set = new DataSet();  
    adapter.Fill(dtb4);  
    foreach (DataRow dr in dtb4.Rows)  
    {  
        comboBox4.Items.Add(dr["School"]);  
    }  
}  
  
public void Subj1()  
{  
    SqlCommand cmd = con.CreateCommand();  
    cmd.CommandType = CommandType.Text;  
    cmd.CommandText = "select * from Subj1";  
    dtb5 = new DataTable();  
    adapter = new SqlDataAdapter();  
    adapter.SelectCommand = cmd;  
    set = new DataSet();  
    adapter.Fill(dtb5);
```

```
foreach (DataRow dr in dtb5.Rows)
{
    comboBox2.Items.Add(dr["Subject"]);
}
}
public void TeacherJob()
{

    SqlCommand cmd = con.CreateCommand();
    cmd.CommandType = CommandType.Text;
    cmd.CommandText = "select * from TeacherJob1";
    dtb6 = new DataTable();
    adapter = new SqlDataAdapter();
    adapter.SelectCommand = cmd;
    set = new DataSet();
    adapter.Fill(dtb6);
    dataGridView1.DataSource = dtb6;

}
public void ColumnEdit()
{
    dataGridView1.Columns["id"].HeaderText = "ID";
    dataGridView1.Columns["id"].AutoSizeMode =
DataGridViewAutoSizeColumnMode.Fill;
    dataGridView1.Columns["Pos1"].HeaderText = "თანამდებობა";
    dataGridView1.Columns["Pos1"].AutoSizeMode =
DataGridViewAutoSizeColumnMode.Fill;
    dataGridView1.Columns["Sub1"].HeaderText = "საგანი";
    dataGridView1.Columns["Sub1"].AutoSizeMode =
DataGridViewAutoSizeColumnMode.Fill;
    dataGridView1.Columns["Clas1"].HeaderText = "კლასი";
    dataGridView1.Columns["Clas1"].AutoSizeMode =
DataGridViewAutoSizeColumnMode.Fill;
    dataGridView1.Columns["Scho1"].HeaderText = "სკოლა";
    dataGridView1.Columns["Scho1"].AutoSizeMode =
DataGridViewAutoSizeColumnMode.DisplayedCells;
    dataGridView1.Columns["Loc1"].HeaderText = "მდებარეობა";
    dataGridView1.Columns["Loc1"].AutoSizeMode =
DataGridViewAutoSizeColumnMode.Fill;
    dataGridView1.Columns["AdTime"].HeaderText = "გამოქვეყნდა";
```

```
        dataGridView1.Columns["AdTime"].AutoSizeMode =
DataGridViewAutoSizeColumnMode.Fill;
        dataGridView1.Columns["Retime"].HeaderText = "ბოლოვდა";
        dataGridView1.Columns["ReTime"].AutoSizeMode =
DataGridViewAutoSizeColumnMode.Fill;
        this.dataGridView1.DefaultCellStyle.Font = new Font("Sylfaen", 10);
        // dataGridView1.Columns["AdTime"].DefaultCellStyle.ForeColor =
Color.SpringGreen;
        comboBox1.SelectedIndex = 0;
        comboBox2.SelectedIndex = 0;
        comboBox3.SelectedIndex = 0;
        comboBox4.SelectedIndex = 0;
        comboBox5.SelectedIndex = 0;

    }

    private void dataGridView1_CellContentClick(object sender,
DataGridViewCellEventArgs e)
    {
        string i = dataGridView1.CurrentRow.Cells["id"].Value.ToString();
        string id = prof.SetId;
        int o = 0;
        Detal d1 = new Detal(con, i,id,o);
        d1.ShowDialog();

    }
    private int DropDownWidth(ComboBox myCombo)
    {
        int maxWidth = 0, temp = 0;
        foreach (var obj in myCombo.Items)
        {
            temp = TextRenderer.MeasureText(myCombo.GetItemText(obj),
myCombo.Font).Width;
            if (temp > maxWidth)
            {
                maxWidth = temp;
            }
        }
        return maxWidth + SystemInformation.VerticalScrollBarWidth;
    }

    public void Time()
    {
```



```
int i = 0;

foreach (DataRow r in dtb6.Rows)
{
    for (i = 0; i < dataGridView1.Rows.Count; i++)
    {
        // DateTime d1 = Convert.ToDateTime(r["AdTime"]);
        DateTime d1 = DateTime.Now;
        DateTime d2 = Convert.ToDateTime(r["ReTime"]);
        if (d1 > d2)
        {
            dataGridView1.Rows[i].Cells["Retime"].Style.ForeColor = Color.Red;
        }
        else
        {
            dataGridView1.Rows[i].Cells["Adtime"].Style.ForeColor =
Color.SpringGreen;
        }
    }
    break;
}
}

private void button1_Click(object sender, EventArgs e)
{

    string position = comboBox1.Text;
    string subject = comboBox2.Text;
    string clas = comboBox3.Text;
    string school = comboBox4.Text;
    string location = comboBox5.Text;

    DataView dv = dtb6.DefaultView;
    if (dv != null)
    {

        if (position != "ყველა თანამდებობა")
        {
            dv.RowFilter = "Pos1='" + position + "'";
        }
        if (subject != "ყველა საგანი")
        {
            dv.RowFilter = "Sub1='" + subject + "'";
        }
    }
}
```

```

if (clas!="ყველა კლასი")
{
    dv.RowFilter = "Clas1=" + clas + "";
}
if (school!="ყველა სკოლა")
{
    dv.RowFilter = "Schol1=" + school + "";
}
if (location != "ყველა რაიონი")
{
    dv.RowFilter = "Loc1=" + location + "";
}

dataGridView1.DataSource = dv;
}
if (position=="ყველა თანამდებობა" && subject=="ყველა საგანი" &&
clas=="ყველა კლასი" && school=="ყველა სკოლა" && location=="ყველა
რაიონი")
{
    TeacherJob();
}

Time();
}

private void label6_Click(object sender, EventArgs e)
{
    string s = prof.Setvalue1;
    string s1 = prof.Setvalue2;
    string zp = "mos";
    teachreg f1 = new teachreg(con,zp,s,s1);
    f1.ShowDialog();
}

private void label9_Click(object sender, EventArgs e)
{
    MessageBox.Show(" კოსტავას 148  ", "კონტაქტი");
}

private void label7_Click(object sender, EventArgs e)
{
    string s = "";

```

```
        string s1 = "";
        string zp = "mas";
        teachreg f1 = new teachreg(con,zp,s,s1);
        f1.ShowDialog();
    }

    private void comboBox4_SelectedIndexChanged(object sender, EventArgs e)
    {

    }

    private void label8_Click(object sender, EventArgs e)
    {

    }

    private void linkLabel1_LinkClicked(object sender,
    LinkLabelLinkClickedEventArgs e)
    {

    System.Diagnostics.Process.Start("https://www.facebook.com/vaxo.fezuashvili/");
    }

    private void linkLabel2_LinkClicked(object sender,
    LinkLabelLinkClickedEventArgs e)
    {

    System.Diagnostics.Process.Start("https://www.facebook.com/profile.php?id=10001
    4317060758");
    }
    }
}

// ---- მომხმარებლის რეგისტრაცია -----
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
```

```
using System.Windows.Forms;
using System.Data.Sql;
using System.Data.SqlClient;
using System.IO;

namespace TeacherJob
{
    public partial class registration : Form
    {
        SqlConnection con;
        SqlCommand cmd;
        SqlDataAdapter adapter;
        DataTable dtb1;
        DataSet set;
        string name;
        string lastname;
        string recive = "";
        public registration(SqlConnection c, string re)
        {
            con = c;
            recive = re;
            InitializeComponent();
        }

        private void registration_Load(object sender, EventArgs e)
        {
            label9.Visible = false;
            if (recive != "mas")
            {
                comboBox1.Visible = false;
            }
            else
            {
                schol1();
            }
        }

        private void button1_Click(object sender, EventArgs e)
        {
            SqlCommand cmd = con.CreateCommand();
            cmd.CommandType = CommandType.Text;
            cmd.CommandText = "select * from PerControl";
        }
    }
}
```

```
dtb1 = new DataTable();
adapter = new SqlDataAdapter();
adapter.SelectCommand = cmd;
set = new DataSet();
adapter.Fill(dtb1);
foreach (DataRow dr in dtb1.Rows)
{
    if (textBox1.Text == dr["PerID"].ToString())
    {
        string dt = dr["DT"].ToString();
        string[] tarigi = dt.Split(' ');
        string dt1 = dateTimePicker1.Value.ToString();
        string[] tarigi1 = dt1.Split(' ');
        string a = tarigi[0].ToString();
        string b = tarigi1[0].ToString();

        if (tarigi1[0].ToString() == tarigi[0].ToString())
        {
            textBox3.Text = dr["name"].ToString();
            textBox4.Text = dr["lastname"].ToString();
        }
        else
        {
            MessageBox.Show("ესეთი პიროვნება ბაზაში ვერ მოიძებნა",
"გაფრთხილება");
        }
    }
}

private void monthCalendar1_DateChanged(object sender,
DateRangeEventArgs e)
{
}

string Imagelocation = "";
private void label10_Click(object sender, EventArgs e)
{
    try
    {
```

```
OpenFileDialog dialog = new OpenFileDialog();
dialog.Filter = "jpg files(*.jpg)|*.jpg| PNG files(*.png)|*.png|ALL
files(*.*)|*. *|";
if (dialog.ShowDialog() == System.Windows.Forms.DialogResult.OK)
{
    Imagelocation = dialog.FileName;
    pictureBox1.ImageLocation = Imagelocation;
    label19.Visible = true;
}
}
catch (Exception)
{
    MessageBox.Show("შეცდომა", "გაფრთხილება",
    MessageBoxButtons.OK, MessageBoxIcon.Error);
}
}

private void button2_Click(object sender, EventArgs e)
{
    string sex = "";
    if (radioButton1.Checked == true)
    {
        sex = radioButton1.Text;
    }
    else
    {
        sex = radioButton2.Text;
    }

    string dt1 = dateTimePicker1.Value.ToString();
    string[] tarigi1 = dt1.Split(' ');
    string time = tarigi1[0].ToString();

    byte[] images = null;
    FileStream stream = new FileStream(Imagelocation, FileMode.Open,
    FileAccess.Read);
    BinaryReader brs = new BinaryReader(stream);
    images = brs.ReadBytes((int)stream.Length);
    con.Open();
    string insert = "Insert into
    Registration(PirID,DT,name,lastname,mail,password,Picture,sex)" +
    "Values(@PirID,@DT,@name,@lastname,@mail,@password,@Picture,@sex)";
```

```

cmd = con.CreateCommand();
cmd.CommandText = insert;
cmd.Parameters.Add("@PirID", SqlDbType.NVarChar, 15);
cmd.Parameters.Add("@DT", SqlDbType.Date, 15);
cmd.Parameters.Add("@name", SqlDbType.NVarChar, 20);
cmd.Parameters.Add("@lastname", SqlDbType.NVarChar, 20);
cmd.Parameters.Add("@mail", SqlDbType.NVarChar, 20);
cmd.Parameters.Add("@password", SqlDbType.NVarChar, 20);
cmd.Parameters.Add("@Picture", SqlDbType.Image);
cmd.Parameters.Add("@sex", SqlDbType.NVarChar);

cmd.Parameters["@PirID"].Value = textBox1.Text;
cmd.Parameters["@DT"].Value = Convert.ToDateTime(time);
cmd.Parameters["@name"].Value = textBox3.Text;
cmd.Parameters["@lastname"].Value = textBox4.Text;
cmd.Parameters["@mail"].Value = textBox2.Text;
cmd.Parameters["@password"].Value = textBox6.Text;
cmd.Parameters["@Picture"].Value = images;
cmd.Parameters["@sex"].Value = sex;
cmd.ExecuteNonQuery();
con.Close();
}
private int DropDownWidth(ComboBox myCombo)
{
    int maxWidth = 0, temp = 0;
    foreach (var obj in myCombo.Items)
    {
        temp = TextRenderer.MeasureText(myCombo.GetItemText(obj),
myCombo.Font).Width;
        if (temp > maxWidth)
        {
            maxWidth = temp;
        }
    }
    return maxWidth + SystemInformation.VerticalScrollBarWidth;
}
public void scholl()
{
    SqlCommand cmd = con.CreateCommand();
    cmd.CommandType = CommandType.Text;
    cmd.CommandText = "select * from Scholl";
    DataTable dtb4 = new DataTable();
    adapter = new SqlDataAdapter();

```

```
adapter.SelectCommand = cmd;
set = new DataSet();
adapter.Fill(dtb4);
foreach (DataRow dr in dtb4.Rows)
{
    comboBox1.Items.Add(dr["School"]);
}
}

private void checkBox1_CheckedChanged(object sender, EventArgs e)
{
    if (checkBox1.Checked == true)
    {
        textBox5.PasswordChar = char.Parse("\0");
        textBox6.PasswordChar = char.Parse("\0");
    }
    else
    {
        textBox5.PasswordChar = '*';
        textBox6.PasswordChar = '*';
    }
}
}
}

// --- მომხმარებლის ავტორიზაცია -----
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.Sql;
using System.Data.SqlClient;
namespace TeacherJob
{
    public partial class teachreg : Form
    {
        SqlConnection con;
```



```
DataTable dtb;
SqlDataAdapter adapter;
SqlDataAdapter adapter1;
int id = 0;
int recive = 0;
string recive1;
string mail0;
string password;

public teachreg(SqlConnection c,string z,string mail1,string pass)
{
    con = c;
    recive1 = z;
    try
    {
        mail0 = mail1;
        password = pass;
    }
    catch(NullReferenceException)
    {

    }

    InitializeComponent();

}

private void button1_Click(object sender, EventArgs e)
{
    if (recive1 == "mos")
    {
        fill();
    }
    if(recive1=="mas")
    {
        fill2();
    }

}

public void fill()
```

```

{
    int indicator = 0;
    if ((textBox1.Text != "") || (textBox2.Text != ""))
    {
        // con.Open();
        SqlCommand cmd = con.CreateCommand();
        cmd.CommandText = "select * from Registration";
        dtb = new DataTable();
        adapter = new SqlDataAdapter();
        adapter.SelectCommand = cmd;
        adapter.Fill(dtb);
        foreach (DataRow row in dtb.Rows)
        {
            string mail = row["mail"].ToString();
            if (textBox1.Text == mail)
            {
                string password = row["password"].ToString();
                if (textBox2.Text == password)
                {
                    indicator = 1;
                    id= Convert.ToInt32(row["id"].ToString());
                    this.Visible = false;
                    teachopen open = new teachopen(con, id);
                    open.ShowDialog();
                    recive = 1;

                }

            }

        }

        // con.Close();
    }
}
if (indicator == 0)
{
    MessageBox.Show("მომხმარებლის სახელი ან პაროლი არასწორია", "შეცდომა");
}

}
public void fill2()

```

```

{
    int indicator = 0;

    if (textBox1.Text!="" || textBox2.Text!="")
    {
        SqlCommand cmd = con.CreateCommand();
        cmd.CommandText = "select * from sajschol";
        dtb = new DataTable();
        adapter = new SqlDataAdapter();
        adapter.SelectCommand = cmd;
        adapter.Fill(dtb);
        foreach (DataRow row in dtb.Rows)
        {
            string Icode = row["Identificationcode"].ToString();
            if (textBox1.Text==Icode)
            {
                string pass = row["pass"].ToString();
                if (textBox2.Text==pass)
                {
                    indicator = 1;
                    string id = row["id"].ToString();
                    this.Visible = false;
                    Directors d1 = new Directors(con,id);
                    d1.ShowDialog();
                }
            }
        }
    }
    if(indicator==0)
    {
        MessageBox.Show("მომხმარებლის სახელი ან პაროლი არასწორია", "შეცდომა");
    }
}

public string mail
{
    get
    {
        return textBox1.Text;
    }
    set
    {

```

```
        mail = textBox1.Text;
    }
}
public string password1
{
    get
    {
        return textBox2.Text;
    }
    set
    {
        password1 = textBox2.Text;
    }
}
public int uk
{
    get
    {
        return recive;
    }
    set
    {
        uk = recive;
    }
}
private void label4_Click(object sender, EventArgs e)
{
    registration reg1 = new registration(con,recive1);
    reg1.ShowDialog();
}

private void teachreg_Load(object sender, EventArgs e)
{
    textBox1.Text = mail0;
    textBox2.Text = password;
    if(textBox1.Text != "" && textBox2.Text!="")
    {
        fill();
        this.Close();
    }

    if (recive1 == "mas")
```

```
{
    label2.Text = "საიდენტიფიკაციო კოდი :";
    label4.Visible = false;
}
}

private void checkBox1_CheckedChanged(object sender, EventArgs e)
{
    if (checkBox1.Checked==true)
    {
        textBox2.PasswordChar = char.Parse("\0");
    }
    else
    {
        textBox2.PasswordChar = '*';
    }
}

private void textBox2_TextChanged(object sender, EventArgs e)
{
    if (checkBox1.Checked == true)
    {
        textBox2.PasswordChar = char.Parse("\0");
    }
    else
    {
        textBox2.PasswordChar = '*';
    }
}

private void label5_Click(object sender, EventArgs e)
{
    PassRecovery re = new PassRecovery(con);
    re.ShowDialog();
}
}

}

//--- მომხმარებლის პაროლის აღდგენა -----
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
```

```
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Net;
using System.Net.Mail;
using System.Net.Mime;
using System.Data.Sql;
using System.Data.SqlClient;
```

```
namespace TeacherJob
{
    public partial class PassRecovery : Form
    {
        SqlConnection con;
        DataTable dtb;
        SqlDataAdapter adapter;
        SqlCommand cmd;
        int zt;
        public PassRecovery(SqlConnection co)
        {
            con = co;
            InitializeComponent();
        }

        private void PassRecovery_Load(object sender, EventArgs e)
        {
            label2.Visible = false;
            textBox2.Visible = false;
            label3.Visible = false;
            label4.Visible = false;
            textBox3.Visible = false;
            textBox4.Visible = false;
        }

        public int GenerateRandom()
        {
            int min = 1000;
            int max = 9999;
            Random rdm = new Random();
            return rdm.Next(min, max);
        }
    }
}
```

```

}
public void Mailsend()
{

}

private void button1_Click(object sender, EventArgs e)
{
    con.Open();
    SqlCommand cmd = con.CreateCommand();
    cmd.CommandType = CommandType.Text;
    cmd.CommandText = "Select * From registration where mail=" +
textBox1.Text+"";
    dtb = new DataTable();
    adapter = new SqlDataAdapter();
    adapter.SelectCommand = cmd;
    adapter.Fill(dtb);
    con.Close();
    if (dtb.Rows.Count>0)
    {

        zt = GenerateRandom();
        try
        {
            MailMessage mail = new MailMessage();
            mail.To.Add("teacherjobapp1@gmail.com");
            mail.To.Add(textBox1.Text);
            mail.From = new MailAddress("teacherjobapp1@gmail.com",
"TEACHERJOBAPP");
            mail.Subject = "პაროლის აღდგენა";
            mail.Body = "<p>შეიყვანეთ ერთჯერადი კოდი " + zt.ToString() +
"</p>"+ "<p>გისურვებთ წარმატებას.</p>"+ "<p>პატივისცემით :
TEACHERJOBAPP </p>";
            mail.IsBodyHtml = true;
            SmtplibClient smtp = new SmtplibClient();
            smtp.Host = "smtp.gmail.com";
            smtp.Credentials = new System.Net.NetworkCredential
("teacherjobapp1@gmail.com", "Password");
            smtp.Port = 465;
            smtp.EnableSsl = true;
            smtp.Send(mail);
            label2.Visible = true;
            textBox2.Visible = true;
        }
    }
}

```

```
        label2.ForeColor = Color.Red;
        label3.Visible = true;
        label4.Visible = true;
        textBox3.Visible = true;
        textBox4.Visible = true;
    }
    catch (Exception ex)
    {
        MessageBox.Show("Exception in sendEmail:" + ex.Message);
    }
}

private void button2_Click(object sender, EventArgs e)
{
    con.Open();
    if (textBox2.Text == zt.ToString())
    {
        if (textBox3.Text == textBox4.Text)
        {
            cmd = new SqlCommand("Update Registration set
password=@password where mail=" + textBox1.Text + "", con);
            cmd.Parameters.Add("@password", SqlDbType.NVarChar, 50);
            cmd.Parameters["@password"].Value = textBox4.Text;
            dtb.AcceptChanges();
            cmd.ExecuteNonQuery();
            con.Close();
            MessageBox.Show("პაროლი წარმატებით შეიცვალა", "პაროლის შეცვლა");
            this.Close();
        }
        else
        {
            MessageBox.Show("პაროლი არ ემთხვევა", "შეცდომა");
        }
    }
    else
    {
        MessageBox.Show("გამოგზავნილი კოდი არასწორია", "შეცდომა");
    }
}
}
```



```
}

//-- მომხმარებლის გვერდი -----
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.Sql;
using System.Data.SqlClient;
using System.IO;

namespace TeacherJob
{
    public partial class prof : Form
    {
        SqlConnection con;
        SqlCommand cmd;
        DataTable dtb;
        DataTable dtb1;
        string value = "1";
        DataTable dtb2;
        SqlDataAdapter adapter;
        string pass = "";
        string id;
        int ukan;
        int cur = 0;
        public prof(string id1, SqlConnection c)
        {
            id = id1;
            con = c;
            InitializeComponent();
        }

        private void prof_Load(object sender, EventArgs e)
        {
            Fill();
            combo1();
        }
    }
}
```

```

        PerReg();
    }
    public static string Setvalue1 = "";
    public static string Setvalue2 = "";
    public static string SetId = "";
    public void Fill()
    {
        SqlCommand cmd = con.CreateCommand();
        cmd.CommandType = CommandType.Text;
        cmd.CommandText= "Select * From registration where id=" + id;
        dtb = new DataTable();
        adapter = new SqlDataAdapter();
        adapter.SelectCommand = cmd;
        adapter.Fill(dtb);
        foreach (DataRow row in dtb.Rows)
        {
            pirlabel.Text = row["PirID"].ToString();
            namelabel.Text = row["name"].ToString();
            lastnamelabel.Text = row["lastname"].ToString();
            maillabel.Text = row["mail"].ToString();
            string DT = row["DT"].ToString();
            DT = DT.Replace("/", "-");
            string[] DT1 = DT.Split(' ');

            DTlabel.Text = DT1[0].ToString();
            sexlabel.Text = row["sex"].ToString();
            moqlabel.Text = "საქართველოს მოქალაქე";
            pass = row["password"].ToString();
            textBox2.Text = row["tel"].ToString();
            byte[] data = new byte[0];
            data = (byte[])(row["picture"]);
            MemoryStream mem = new MemoryStream(data);
            pictureBox1.Image = Image.FromStream(mem);
            Setvalue1 = row["mail"].ToString();
            Setvalue2 = row["password"].ToString();
            SetId = row["id"].ToString().ToString();
        }
        ukan = 1;
        xaxisxi();
        minichebuli();
    }
    public void combo1()

```

```

    {
        SqlCommand cmd = con.CreateCommand();
        cmd.CommandType = CommandType.Text;
        cmd.CommandText = "select * from TreeRegion where
pcode is null";
        dtb1 = new DataTable();
        adapter = new SqlDataAdapter();
        adapter.SelectCommand = cmd;
        DataSet set = new DataSet();
        adapter.Fill(dtb1);
        foreach (DataRow dr in dtb1.Rows)
        {
            comboBox1.Items.Add(dr["RegRion"]);
        }
        comboBox1.DataSource = new BindingSource(dtb1, null);
        comboBox1.DisplayMember = "RegRion";
        comboBox1.ValueMember = "id";
    }

    private void button1_Click(object sender, EventArgs e)
    {
        DataRowView r =
(DataRowView)comboBox1.SelectedItem;
        DataRow r1 = r.Row;
        value = r["id"].ToString();
        DataRowView r2 =
(DataRowView)comboBox2.SelectedItem;
        DataRow rr = r2.Row;
        string value2 = rr["pcode"].ToString();
        string value3 = rr["code"].ToString();

        con.Open();
        cmd = new SqlCommand("delete from PerReg where
perid=" + id, con);
        dtb1.AcceptChanges();
        cmd.ExecuteNonQuery();
        con.Close();
        int value4 =
Convert.ToInt32(comboBox3.SelectedIndex);
        if (comboBox1.Text != "" && comboBox2.Text != ""
&& textBox1.Text != "")
        {

```

```

        con.Open();
        string insert = "insert into
PerReg(perid,regtip,regpcode,adres,percode,regrion1,xarix)"
+
"VALUES(@perid,@regtip,@regpcode,@adres,@percode,@regrion1,@
xarix)";

        cmd = con.CreateCommand();
        cmd.CommandText = insert;
        cmd.Parameters.Add("@perid", SqlDbType.Int, 4);
        cmd.Parameters.Add("@regtip", SqlDbType.Int, 4);
        cmd.Parameters.Add("@regpcode", SqlDbType.Int, 4);
        cmd.Parameters.Add("@adres",
SqlDbType.NVarChar, 50);
        cmd.Parameters.Add("@percode", SqlDbType.Int, 4);
        cmd.Parameters.Add("@regrion1", SqlDbType.NVarChar, 50);
        cmd.Parameters.Add("@xarix", SqlDbType.Int, 4);
        cmd.Parameters["@perid"].Value = id;
        cmd.Parameters["@regtip"].Value = value;
        cmd.Parameters["@regpcode"].Value = value2;
        cmd.Parameters["@adres"].Value = textBox1.Text;
        cmd.Parameters["@percode"].Value = value3;
        cmd.Parameters["@regrion1"].Value = comboBox1.Text;
        cmd.Parameters["@xarix"].Value = value4;
        cmd.ExecuteNonQuery();
        con.Close();
    }

}
private void comboBox2_SelectedIndexChanged(object
sender, EventArgs e)
{
    comboBox2.DataSource = new BindingSource(dtb2, null);
    comboBox2.DisplayMember = "RegRion";
    comboBox2.ValueMember = "pcode";
}
public void PerReg()
{
    SqlCommand cmd = con.CreateCommand();
    cmd.CommandType = CommandType.Text;
    cmd.CommandText = "select A.id,
A.pcode,A.RegRion, B.perid, B.regtip,B.regpcode,
B.regrion1,B.xarix, B.adres,B.percode from TreeRegion A

```

```

inner join PerReg B on A.pcode=B.regpcode and
A.code=B.percode where B.perid=" + id;
    dtb1 = new DataTable();
    adapter = new SqlDataAdapter();
    adapter.SelectCommand = cmd;
    DataSet set = new DataSet();
    adapter.Fill(dtb1);
    foreach (DataRow row in dtb1.Rows)
    {
        mislabel.Text = row["regrion1"].ToString() +
        "," + " " + row["Regrion"].ToString() + ", " + " " +
        row["adres"].ToString();
        comboBox2.Text = row["Regrion"].ToString();
        textBox1.Text = row["adres"].ToString();
        comboBox1.Text = row["regrion1"].ToString();
        comboBox3.SelectedIndex =
Convert.ToInt32(row["xarisx"].ToString());
        break;
    }
}
public void xarisxi()
{
    using (StreamReader x = new
StreamReader(@"C:\Users\Vakhtang.Pezuashvili\Documents\Visual
Studio
2015\Projects\TeacherJob\TeacherJob\bin\Debug\xarisxi.txt"))
    {
        string s = "";
        while ((s = x.ReadLine()) != null)
        {
            string[] lines = s.Split(new[] { "\r\n",
"\r", "\n" }, StringSplitOptions.None);
            for (int i = 0; i < lines.Length; i++)
            {
                comboBox3.Items.Add(lines[i]);
            }
        }
    }
}
public void minichebuli()
{

```

```

        using (StreamReader x = new
StreamReader(@"C:\Users\Vakhtang.Pezuashvili\Documents\Visual
Studio
2015\Projects\TeacherJob\TeacherJob\bin\Debug\minichebuli.txt
"))
        {
            string s = "";
            while ((s = x.ReadLine()) != null)
            {
                string[] lines = s.Split(new[] { "\r\n",
"\r", "\n" }, StringSplitOptions.None);
                for (int i = 0; i < lines.Length; i++)
                {
                    comboBox4.Items.Add(lines[i]);
                }
            }
        }

private void button1_Click_1(object sender, EventArgs e)
{
}

private void comboBox1_SelectedIndexChanged_1(object
sender, EventArgs e)
{
    DataRowView r = (DataRowView)comboBox1.SelectedItem;
    DataRow r1 = r.Row;
    value = r["id"].ToString();
    comboBox2.Items.Clear();
    SqlCommand cmd = con.CreateCommand();
    cmd.CommandType = CommandType.Text;
    cmd.CommandText = "select * from TreeRegion where
pcode=" + value;
    dtb2 = new DataTable();
    adapter = new SqlDataAdapter();
    adapter.SelectCommand = cmd;
    DataSet set = new DataSet();
    adapter.Fill(dtb2);
    foreach (DataRow dr in dtb2.Rows)

```



```

        InitializeComponent();
    }

    private void button1_Click(object sender, EventArgs e)
    {
        if (gan == "add")
        {
            DialogResult dialogresult =
            MessageBox.Show("ნამდვილად გნებავთ დამატება", "მონაცემის
            დამატება", MessageBoxButtons.YesNo);
            if (dialogresult == DialogResult.Yes)
            {
                if (textBox1.Text != "" && textBox2.Text
                != "" && dateTimePicker1.Value != null)
                {
                    if (dateTimePicker2.Value != null ||
                    checkBox1.Checked == true)
                    {
                        con.Open();
                        string insert = "insert into
                        jobexper(perid,orgdasax,orgpos,dtb1,dtb2,litcoment)" +
                        "VALUES(@perid,@orgdasax,@orgpos,@dtb1,@dtb2,@litcoment)";
                        cmd = con.CreateCommand();
                        cmd.CommandText = insert;
                        cmd.Parameters.Add("@perid", SqlDbType.Int, 4);
                        cmd.Parameters.Add("@orgdasax", SqlDbType.NVarChar, 150);
                        cmd.Parameters.Add("@orgpos", SqlDbType.NVarChar, 150);
                        cmd.Parameters.Add("@dtb1", SqlDbType.Date, 12);
                        cmd.Parameters.Add("@dtb2", SqlDbType.NVarChar, 20);
                        cmd.Parameters.Add("@litcoment", SqlDbType.NVarChar, 150);
                        cmd.Parameters["@perid"].Value = id;
                        cmd.Parameters["@orgdasax"].Value = textBox1.Text;
                        cmd.Parameters["@orgpos"].Value = textBox2.Text;
                        cmd.Parameters["@dtb1"].Value =
                        Convert.ToDateTime(dateTimePicker1.Value);
                        if (checkBox1.Checked == true)
                        {
                            cmd.Parameters["@dtb2"].Value = "დღემდე";
                        }
                        else
                    {

```



```

        cmd.Parameters["@dtb2"].Value =
Convert.ToDateTime(dateTimePicker2.Value);
    }

cmd.Parameters["@litcomment"].Value = textBox3.Text;
cmd.ExecuteNonQuery();
con.Close();
this.Close();
    }
    else
    {
        MessageBox.Show("გთხოვთ შეავსოთ
აუცილებელი ველები", "გაფრთხილება");
    }
}
else
{
    MessageBox.Show("გთხოვთ შეავსოთ
აუცილებელი ველები", "გაფრთხილება");
}
}

con.Close();
}
else if(gan=="edit")
{
    DialogResult dialogresult =
MessageBox.Show("ნამდვილად გნებავთ მონაცემის კორექტირება",
"მონაცემის კორექტირება", MessageBoxButtons.YesNo);
    if (dialogresult == DialogResult.Yes)
    {
        if (gan == "edit")
        {
            row["orgdasax"] = textBox1.Text;
            row["orgpos"] = textBox2.Text;
            row["dtb1"] =
Convert.ToDateTime(dateTimePicker1.Value);
            string text3 = textBox3.Text;
            if (checkBox1.Checked==true)
            {
                row["dtb2"] = "დღემდე";
            }

```

```

else
{
    row["dtb2"] =
Convert.ToDateTime(dateTimePicker2.Value);
}
row["litcoment"] = text3;
string id1 = row["id"].ToString();
con.Open();
cmd = new SqlCommand("Update jobexper
set orgdasax=@orgdasax, orgpos=@orgpos, dtb1=@dtb1,
dtb2=@dtb2, litcoment=@litcoment where id=" + id1, con);
cmd.Parameters.Add("@perid", SqlDbType.Int, 4);
cmd.Parameters.Add("@orgdasax", SqlDbType.NVarChar, 150);
cmd.Parameters.Add("@orgpos", SqlDbType.NVarChar, 150);
cmd.Parameters.Add("@dtb1", SqlDbType.Date, 12);
cmd.Parameters.Add("@dtb2", SqlDbType.NVarChar, 20);
cmd.Parameters.Add("@litcoment", SqlDbType.NVarChar, 150);
cmd.Parameters["@perid"].Value = id;
cmd.Parameters["@orgdasax"].Value = textBox1.Text;
cmd.Parameters["@orgpos"].Value = textBox2.Text;
cmd.Parameters["@dtb1"].Value =
Convert.ToDateTime(dateTimePicker1.Value);
if (checkBox1.Checked == true)
{
    cmd.Parameters["@dtb2"].Value = "დღემდე";
}
else
{
    cmd.Parameters["@dtb2"].Value =
Convert.ToDateTime(dateTimePicker2.Value);
}
cmd.Parameters["@litcoment"].Value = text3;
dtb.AcceptChanges();
cmd.ExecuteNonQuery();
con.Close();
this.Close();
}
}
}
}

private void button2_Click(object sender, EventArgs e)

```

```

    {
        this.Close();
    }

private void experadd_Load(object sender, EventArgs e)
{
    if (gan=="edit")
    {
        textBox1.Text = row["orgdasax"].ToString();
        textBox2.Text = row["orgpos"].ToString();
        dateTimePicker1.Value
=Convert.ToDateTime(row["dtb1"].ToString());
        string dtb2 = row["dtb2"].ToString();
        if (dtb2 == "დღემდე")
        {
            checkBox1.Checked = true;
        }
        else
        {
            dateTimePicker2.Value =
Convert.ToDateTime(row["dtb2"].ToString());
        }
        textBox3.Text = row["litcoment"].ToString();
    }
}
}
}
}

```

//--- ვაკანსიის დამატება (სკოლის მიერ) -----

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.Sql;
using System.Data.SqlClient;
using System.IO;

```

```

namespace TeacherJob
{
    public partial class Directors : Form
    {
        string id;
        SqlConnection con;
        SqlCommand cmd;
        DataTable dtb;
        SqlDataAdapter adapter;
        DataSet set;
        DataTable dtb1;
        DataTable dtb2;
        DataTable dtb3;
        DataTable dtb4;
        int stime = 0;
        int vraod = 0;
        string scholid;
        public Directors(SqlConnection c, string idd)
        {
            con = c;
            id = idd;
            InitializeComponent();
        }

        private void Directors_Load(object sender, EventArgs e)
        {
            DirFill();
            posst();
            Subj1();
            Clas1();
            seqtor();
            dawbox.Text = "0";
            raodbox.Text = "0";
        }
        public void DirFill()
        {
            SqlCommand cmd = con.CreateCommand();
            cmd.CommandType = CommandType.Text;
            cmd.CommandText = "select * from sajschol A where id=" + id;
            dtb = new DataTable();
            adapter = new SqlDataAdapter();
            adapter.SelectCommand = cmd;
        }
    }
}

```

```
set = new DataSet();
adapter.Fill(dtb);

foreach (DataRow row in dtb.Rows)
{
    label4.Text = row["school"].ToString();
label4.BackColor = Color.LightGoldenrodYellow;
    label5.Text = row["tip"].ToString();
label5.BackColor = Color.LightGoldenrodYellow;
label6.Text = row["Identificationcode"].ToString();
    label6.BackColor = Color.LightGoldenrodYellow;
}
}
public void posst()
{
    SqlCommand cmd = con.CreateCommand();
    cmd.CommandType = CommandType.Text;
    cmd.CommandText = "select * from Posst where id>1";
    dtb1 = new DataTable();
    adapter = new SqlDataAdapter();
    adapter.SelectCommand = cmd;
    set = new DataSet();
    adapter.Fill(dtb1);
    foreach (DataRow dr in dtb1.Rows)
    {

        comboBox1.Items.Add(dr["Possition"]);
    }
}
public void Subj1()
{
    SqlCommand cmd = con.CreateCommand();
    cmd.CommandType = CommandType.Text;
    cmd.CommandText = "select * from Subj1 where id>1";
    dtb2 = new DataTable();
    adapter = new SqlDataAdapter();
    adapter.SelectCommand = cmd;
    set = new DataSet();
    adapter.Fill(dtb2);
    foreach (DataRow dr in dtb2.Rows)
    {
```

```

        comboBox2.Items.Add(dr["Subject"]);
    }
}
public void Clas1()
{
    SqlCommand cmd = con.CreateCommand();
    cmd.CommandType = CommandType.Text;
    cmd.CommandText = "select * from Clas1 where id>1";
    dtb3 = new DataTable();
    adapter = new SqlDataAdapter();
    adapter.SelectCommand = cmd;
    set = new DataSet();
    adapter.Fill(dtb3);
    foreach (DataRow dr in dtb3.Rows)
    {
        comboBox3.Items.Add(dr["Class"]);
    }
}
public void bolo()
{
    SqlCommand cmd = con.CreateCommand();
    cmd.CommandType = CommandType.Text;
    cmd.CommandText = "select* from TeacherJob1 where
id = (select MAX(id)from TeacherJob1)";
    dtb4 = new DataTable();
    adapter = new SqlDataAdapter();
    adapter.SelectCommand = cmd;
    set = new DataSet();
    adapter.Fill(dtb4);
    foreach (DataRow dr in dtb4.Rows)
    {
        scholid = dr["id"].ToString();
    }
}
public void seqtor()
{
    using (StreamReader x = new
StreamReader(@"C:\Users\Vakhtang.Pezuashvili\Documents\Visual
Studio
2015\Projects\TeacherJob\TeacherJob\bin\Debug\seqtori.txt"))
    {
        string s = "";
    }
}

```

```
        while((s=x.ReadLine())!=null)
        {
            string[] lines = s.Split(new[] { "\r\n",
"\r", "\n" }, StringSplitOptions.None);
            for (int i=0; i<lines.Length; i++)
            {
                comboBox4.Items.Add(lines[i]);
            }
        }
    }

private void skleba_Click(object sender, EventArgs e)
{
    int i = Convert.ToInt32(dawbox.Text);
    if (i>0)
    {
        stime = i - 1;
        dawbox.Text = stime.ToString();
    }
}

private void smateba_Click(object sender, EventArgs e)
{
    int i = Convert.ToInt32(dawbox.Text);
    if (i >= 0)
    {
        stime = i + 1;
        dawbox.Text = stime.ToString();
    }
}

private void vkleba_Click(object sender, EventArgs e)
{
    int i = Convert.ToInt32(raodbox.Text);
    if (i>0)
    {
        vraod = i - 1;
        raodbox.Text = vraod.ToString();
    }
}
```

```
    }  
  }  
  
  private void vmateba_Click(object sender, EventArgs e)  
  {  
    int i = Convert.ToInt32(raodbox.Text);  
    if (i>=0)  
    {  
      vraod = i + 1;  
      raodbox.Text = vraod.ToString();  
    }  
  }  
  
  private void button1_Click(object sender, EventArgs e)  
  {  
    string pos1 = comboBox1.SelectedItem.ToString();  
    string sub1 = comboBox2.SelectedItem.ToString();  
    string clas1 = comboBox3.SelectedItem.ToString();  
    string seqtor = comboBox4.SelectedItem.ToString();  
    string schol1 = "";  
    string loc1 = "";  
    string adres = "";  
    string tip = "";  
    foreach (DataRow r in dtb.Rows)  
    {  
      schol1 = r["school"].ToString();  
      loc1 = r["raioni"].ToString();  
      adres = r["adres"].ToString();  
      tip = r["tip"].ToString();  
    }  
  
    con.Open();  
    string insert = "insert into TeacherJob1(Pos1,  
Sub1, Clas1,Scho1,Loc1,AdTime,ReTime)" +  
"VALUES(@Pos1,@Sub1,@Clas1,@Scho1,@Loc1,@AdTime,@ReTime)";  
    cmd = con.CreateCommand();  
    cmd.CommandText = insert;  
    cmd.Parameters.Add("@Pos1", SqlDbType.NVarChar, 150);  
    cmd.Parameters.Add("@Sub1", SqlDbType.NVarChar, 150);  
    cmd.Parameters.Add("@Clas1", SqlDbType.NVarChar, 150);  
    cmd.Parameters.Add("@Scho1", SqlDbType.NVarChar, 200);
```



```

cmd.Parameters.Add("@Loc1", SqlDbType.NVarChar, 150);
cmd.Parameters.Add("@AdTime", SqlDbType.DateTime, 12);
cmd.Parameters.Add("@ReTime", SqlDbType.DateTime, 12);
    DateTime time1 = DateTime.Now;
    cmd.Parameters["@Pos1"].Value = pos1;
    cmd.Parameters["@Sub1"].Value = sub1;
    cmd.Parameters["@Clas1"].Value = clas1;
    cmd.Parameters["@Scho1"].Value = schol1;
    cmd.Parameters["@Loc1"].Value = loc1;
    cmd.Parameters["@Adtime"].Value = time1;
    cmd.Parameters["@Retime"].Value =
datePicker2.Value;
    cmd.ExecuteNonQuery();
    bolo();
    string insert1 = "insert into
Teach12(scholid,sektor,tel,dt1,samtime,salary,motxovna,misama
rti,tip,raodenoba)" +
"VALUES(@scholid,@sektor,@tel,@dt1,@samtime,@salary,@motxovna
,@misamarti,@tip,@raodenoba)";
    cmd = con.CreateCommand();
    cmd.CommandText = insert1;
    cmd.Parameters.Add("@scholid", SqlDbType.Int, 4);
    cmd.Parameters.Add("@sektor",
SqlDbType.NVarChar,100);
    cmd.Parameters.Add("@tel", SqlDbType.Int, 12);
    cmd.Parameters.Add("@dt1", SqlDbType.Date, 12);
    cmd.Parameters.Add("@samtime", SqlDbType.Int, 12);
    cmd.Parameters.Add("@salary", SqlDbType.Int, 12);
    cmd.Parameters.Add("@motxovna",
SqlDbType.NVarChar, 100);
    cmd.Parameters.Add("@misamarti",
SqlDbType.NVarChar, 100);
    cmd.Parameters.Add("@tip",
SqlDbType.NVarChar,50);
    cmd.Parameters.Add("@raodenoba", SqlDbType.Int, 4);
    cmd.Parameters["@scholid"].Value =
Convert.ToInt32(scholid);
    cmd.Parameters["@sektor"].Value =sektor;
    cmd.Parameters["@tel"].Value =
Convert.ToInt32(telbox.Text);
    cmd.Parameters["@dt1"].Value =
datePicker1.Value;

```

```

        cmd.Parameters["@samtime"].Value =
Convert.ToInt32(dawbox.Text);
        cmd.Parameters["@salary"].Value =
Convert.ToInt32(salarybox.Text);
        cmd.Parameters["@motxovna"].Value = coment.Text;
        cmd.Parameters["@misamarti"].Value = address;
        cmd.Parameters["@tip"].Value = tip;
        cmd.Parameters["@raodenoba"].Value =
Convert.ToInt32(raodbox.Text);
        cmd.ExecuteNonQuery();
        con.Close();
    }

    private void panel1_Paint(object sender,
PaintEventArgs e)
    {
    }

    private void comboBox4_SelectedIndexChanged(object sender,
EventArgs e)
    {
    }
}

//--- ვაკანსიის დეტალურად ნახვა -----
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.Sql;
using System.Data.SqlClient;
using System.Drawing.Printing;

namespace TeacherJob

```

```

{
public partial class Detal : Form
{
    SqlConnection con;
    SqlCommand cmd;
    DataTable dtb1;
    SqlDataAdapter adapter;
    DataSet set;
    string id;
    string tve;
    string[] DT1;
    string perid;
    int cur;
public Detal(SqlConnection c, string i,string id1,int o)
{
    con = c;
    id = i;
    perid = id1;
    cur = o;
    InitializeComponent();
}

private void Detal_Load(object sender, EventArgs e)
{
    mypanel.AutoScroll = true;
    mypanel.HorizontalScroll.Enabled = true;
    mypanel.HorizontalScroll.Visible = true;
    mypanel.HorizontalScroll.Maximum = 0;
    mypanel.AutoScroll = true;
    fil();
    if (perid=="")
    {
        button1.Visible = false;
    }
    if(cur==1)
    {
        button1.Visible = true;
    }
}
public void fil()

```

```

{
    SqlCommand cmd = con.CreateCommand();
    cmd.CommandType = CommandType.Text;
    cmd.CommandText = "select A.id,
A.Pos1,A.Sub1,A.Clas1,A.Scho1,A.loc1,A.Adtime,
A.Retime,B.seqtora,B.tel,B.dtl,B.samtime,B.raodenoba,B.tip,B.s
alary,B.motxovna,B.misamarti from TeacherJob1 A inner join
Teach12 B on A.id=B.scholid where B.scholid="+id;
    dtb1 = new DataTable();
    adapter = new SqlDataAdapter();
    adapter.SelectCommand = cmd;
    set = new DataSet();
    adapter.Fill(dtb1);
    foreach(DataRow r in dtb1.Rows)
    {
        skola.Text = r["scho1"].ToString();
        sagani.Text = r["Sub1"].ToString();
        tipi.Text = r["tip"].ToString();
        klasi.Text = r["Clas1"].ToString();
        seqtori.Text = r["seqtora"].ToString();
        regioni.Text = r["loc1"].ToString();
        misamarti.Text = r["misamarti"].ToString();
        tele.Text = r["tel"].ToString();
        DT1 = r["dtl"].ToString().Split(' ');
        dro();
        dawyeba.Text = tve;
        saatebi.Text = r["samtime"].ToString()+" "+ "სთ/კვირაში";
        salary.Text = r["salary"].ToString()+" "+ "ლარი/თვეში";
        raodenoba.Text = r["raodenoba"].ToString();
        motxovna.Text = r["motxovna"].ToString();
        DT1 = r["Adtime"].ToString().Split(' ');
        dro();
        gamoqveyneba.Text = tve;
        DT1 = r["Retime"].ToString().Split(' ');
        dro();
        boloDT.Text = tve;
    }
}

}
public void dro ()

```

```
{  
  
    string[] s = DT1[0].Split('/');  
    int number = Convert.ToInt32(s[0]);  
  
    switch(number)  
    {  
        case 1:  
            tve = "იანვარი"; break;  
        case 2:  
            tve = "თებერვალი"; break;  
        case 3:  
            tve = "მარტი"; break;  
        case 4:  
            tve = "აპრილი"; break;  
        case 5:  
            tve = "მაისი"; break;  
        case 6:  
            tve = "ივნისი"; break;  
        case 7:  
            tve = "ივლისი"; break;  
        case 8:  
            tve = "აგვისტო"; break;  
        case 9:  
            tve = "სექტემბერი"; break;  
        case 10:  
            tve = "ოქტომბერი"; break;  
        case 11:  
            tve = "ნოემბერი"; break;  
        case 12:  
            tve = "დეკემბერი"; break;  
    }  
    tve =s[1]+ " "+ tve+", "+ s[2].ToString()+"წ";  
}  
Bitmap bmp;  
  
private void button2_Click(object sender, EventArgs e)  
{  
    Panel panel = new Panel();  
    this.Controls.Add(panel);  
}
```

```
Graphics grp = panel.CreateGraphics();
Size formSize = this.ClientSize;
bmp = new Bitmap(formSize.Width, formSize.Height, grp);
grp = Graphics.FromImage(bmp);
Point panelLocation =
PointToScreen(panel.Location);
grp.CopyFromScreen(panelLocation.X,
panelLocation.Y, 0, 0, formSize);
printPreviewDialog1.Document = printDocument1;
printPreviewDialog1.PrintPreviewControl.Zoom = 1;
printPreviewDialog1.ShowDialog();
}

private void printDocument1_PrintPage(object sender,
System.Drawing.Printing.PrintPageEventArgs e)
{
    e.Graphics.DrawImage(bmp, 0, 0);
}
private void CaptureScreen()
{
    Graphics myGraphics = this.CreateGraphics();
    Size s = this.Size;
    bmp = new Bitmap(s.Width, s.Height, myGraphics);
    Graphics memoryGraphics =
Graphics.FromImage(bmp);
    memoryGraphics.CopyFromScreen(this.Location.X,
this.Location.Y, 0, 0, s);
}

private void button1_Click(object sender, EventArgs e)
{
    con.Open();
    string insert = "insert into
pervac(perid,vacancyid) + "VALUES(@perid,@vacancyid)";
    cmd = con.CreateCommand();
    cmd.CommandText = insert;
    cmd.Parameters.Add("@perid", SqlDbType.NVarChar, 150);
```

```
        cmd.Parameters.Add("@vacancyid",  
SqlDbType.NVarChar, 150);  
        cmd.Parameters["@perid"].Value  
=Convert.ToInt32(perid);  
        cmd.Parameters["@vacancyid"].Value =  
Convert.ToInt32(id);  
        cmd.ExecuteNonQuery();  
        con.Close();  
        MessageBox.Show("მოთხოვნა წარმატებით გაიზავნა",  
"მოთხოვნის გაგზავნა");  
    }  
}  
}
```

საკურსო პროექტში გამოყენებული ლიტერატურა:

1. სურგულაძე გ., თურქია ე. პროგრამული სისტემების მენეჯმენტის საფუძვლები. სახელმძღვ., სტუ, თბ., 2016. 350 გვ.
http://gtu.ge/book/gia_sueguladze/GiaSurg1_%20ProgSysManag.pdf

2. ჩოგოვაძე გ., ფრანგიშვილი გ., სურგულაძე გ. მართვის საინფორმაციო სისტემების დაპროგრამების ჰიბრიდული ტექნოლოგიები და მონაცემთა მენეჯმენტი. მონოგრაფია, თბ., „ტექნიკური უნივერსიტეტი“. 2017.

http://gtu.ge/book/monacemta_menejmenti.pdf

3. სურგულაძე გ. კომპიუტერული პროგრამირების მეთოდები და მეთოდოლოგიები (SP, OOP, VP, Agile, UML). სტუ-ს „IT კონსალტინგ ცენტრი“, თბ., 2019., -202 გვ.

https://gtu.ge/book/Surg_ProgMethod_2019.pdf

4. სურგულაძე გ., ურუშაძე ბ. საინფორმაციო სისტემების მენეჯმენტის საერთაშორისო გამოცდილება (BSI, ITIL, COBIT). სტუ, „ტექნიკური უნივერსიტეტი“. თბილისი, **2014**. -320 გვ.
http://gtu.ge/book/gia_sueguladze/sainfo_sistemebi_BSI_ITIL_COBIT.pdf

გადაეცა წარმოებას 10.02 2022. ოფსეტური ქაღალდის ზომა 60X84 1/16. პირობითი ნაბეჭდი თაბახი 11,5. ტირაჟი 50 ეგზ.



სტუ-ს „IT კონსალტინგის ცენტრი“,
თბილისი, მ.კოსტავას 77