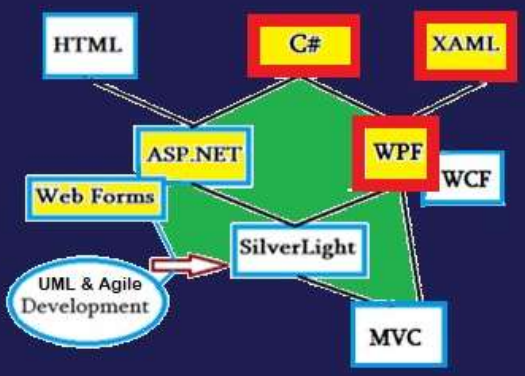
საქართველოს ტექნიკური უნივერსიტეტი
1922 წლიდან

გია სურგულაძე, ნინო თოფურია,
ანა ბერულავა

პროგრამული პროდუქტების დეველოპმენტი



სტუ-ს IT-კონსალტინგის ცენტრი

საქართველოს ტექნიკური უნივერსიტეტი

გია სურგულაძე, ნინო თოფურია,
ანა ბერულავა

პროგრამული პროდუქტების დეველოპმენტი

(WPF, C#.NET, XAML, Azure SQL)

(საკურსო პროექტის დამხმარე სახელმძღვანელო)

რეკომენდებულია:

სტუ-ს „IT კონსალტინგის სამეცნიერო ცენტრის“ სარედაქციო კოლეგიის მიერ
21.12.2021, ოქმი N 4

თბილისი - 2022

უაკ 004.5

გადმოცემულია პროგრამული პროდუქტების დეველოპმენტის (აპლიკაციების განვითარების) პროცესი მაიკროსოფთის თანამედროვე ტექნოლოგიების ბაზაზე. განხილულია პროგრამული სისტემების მენეჯმენტის სასიცოცხლო ციკლის ეტაპები, კერძოდ, დესკტოპ-, ვებ- და მობილური აპლიკაციების ინტეგრირებული პროგრამული რეალიზაციიის მიზნით. შემოთავაზებულია აღნიშნულ დისციპლინაში საკურსო პროექტის შესრულების მეთოდოლოგია და კონკრეტული საილუსტრაციო მაგალითები. ინსტრუმენტული საშუალებების სახით განიხილება MsVisual Studio.NET Framework პლატფორმა, ჰიბრიდული WPF ტექნოლოგია C# და XAML ენებით, NuGet პაკეტისა და ADO.NET დრაივერის, MsSQL Server და MongoDB Compass მონაცემთა ბაზების, აგრეთვე ASP.NET Web, Office365-ის პაკეტების Azure SQL და Power App-ის, Android-ის გამოყენებით. განკუთვნილია ინფორმატიკის ბაკალავრიატის საგანმანათლებლო პროგრამის („პროგრამული ინჟინერიის“ კონცენტრაციის) სპეციალობის მაღალი კურსის სტუდენტებისათვის,

რეცენზენტები:

პროფ. ე. თურქია, ინფორმატიკის ტ.მ.კ., (საქ. ეროვნული ბანკი)
პროფ. გ. ღვინეფაძე, ინფორმატიკის ტ.მ.კ., (სტუ)

რედკოლეგია:

ა. ფრანგიშვილი (თავმჯდომარე), ზ. აზმაიფარაშვილი, ნ. ამილახვარი, მ. ახობაძე, ზ. ბოსიკაშვილი, ე. თურქია, თ. კაიშაური, რ. კაკუბავა, ვ. კვარაცხელია, თ. ლომინაძე, ნ. ლომინაძე, ჰ. მელაძე, ლ. პეტრიაშვილი, ი. ქართველიშვილი, მ. ჩხაიძე, ა. ცინცაძე, ზ. წვერაიძე, გ. სურგულაძე (რეადაქტორი)

© სტუ-ს „ IT კონსალტინგის სამეცნიერო ცენტრი”, 2022
ISBN 978-9941-8-3809-5

ყველა უფლება დაცულია, ამ წიგნის არც ერთი ნაწილი (იქნება ეს ტექსტი, ფოტო, ილუსტრაცია თუ სხვა) არაბაირი ფორმით და საშუალებით (იქნება ეს ელექტრონული თუ მექანიკური), არ შეიძლება გამოყენებულ იქნას გამომცემლის წერილობითიი ნებართვის გარეშე.

სარჩევი

შესავალი:

საკურსო პროექტის მიზანი, ამოცანის დასმა და შესრულების გეგმა

აკადემიური კურსი „პროგრამული პროდუქტების დეველოპმენტი“ ისწავლება საქართველოს ტექნიკური უნივერსიტეტის „ინფორმატიკის“ საგანმანათლებლო პროგრამის („ინფორმატიკისა და მართვის სისტემების“ ფაკულტეტი) მე-7 სემესტრში („პროგრამული ინჟინერიის“ კონცენტრაციით).

სასწავლო კურსის მიზანია შეასწავლოს სტუდენტებს კომპიუტერული სისტემების განვითარების (დეველოპმენტის) მეთოდოლოგიები და მეთოდები საერთაშორისო სტანდარტებისა და საუკეთესო პრაქტიკების საფუძველზე. კერძოდ: კორპორაციული მენეჯმენტის საინფორმაციო სისტემების პროგრამული უზრუნველყოფის (გამოყენებითი პროგრამული აპლიკაციების) პაკეტების შექმნის გუნდური მეთოდები და პროგრამირების ჰიბრიდული (ვინდოუს- და ვებ-) ტექნოლოგიები; პრაქტიკული გამოყენების მიზნით ობიექტ-ორიენტირებული მეთოდი, ენა და შესაბამისი WPF-ინსტრუმენტული საშუალები (Ms Windows Presentation Foundation); ASP.NET MVC (Model-View-Controller) Visual Studio.NET Framevork პლატფორმაზე. აგრეთვე Office365-ის პაკეტების Azure SQL და Power App-ის გამოყენება.

კურსი 6-კრედიტიანია (1 სემესტრი - 150 სთ). იგი ითვალისწინებს ლექციებს, ლაბორატორიებს და საკურსო პროექტს, სადაც სტუდენტებს ეძლევათ საშუალება სპეციალობის საგნის ცოდნასთან ერთად მიიღონ გუნდური მუშაობის უნარების გამოცდილება.

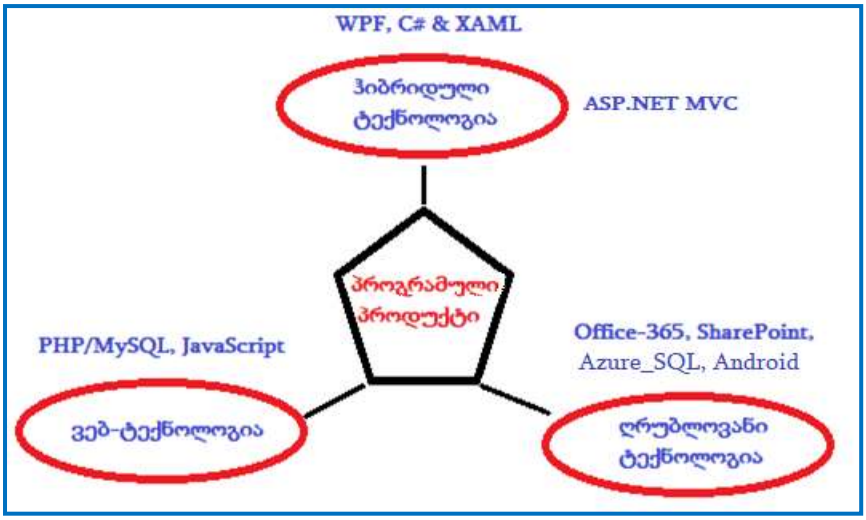
„პროგრამული პროდუქტების დეველოპმენტი“-ს საკურსო პროექტი კომპლექსური, ინტერდისციპლინური ხასიათისაა. იგი სტუდენტისაგან მოითხოვს საპრობლემო სფეროს ობიექტ-ორიენტირებული მოდელირების, ანალიზისა და დაპროექტების მეთოდოლოგიის ცოდნას საერთაშორისო სტანდარტების საფუძველზე. კორპორაციის მონაცემთა განაწილებული ბაზების აგებისა და ვიზუალური დაპროგრამების მეთოდების და ინსტრუმენტული საშუალებების გამოყენების უნარ-ჩვევებს [1,2].

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

საკურსო პროექტის საპრობლემო სფერო შეიძლება იყოს ნებისმიერი დარგის ობიექტის ორგანიზაციული მართვის (მენეჯმენტის) მხარდამჭერი სისტემა. მაგალითად, განათლების სფერო: უნივერსიტეტი, ფაკულტეტი, კათედრა, სკოლა და ა.შ.; დარგობრივი სამინისტროების ნებისმიერი დეპარტამენტი, სოფლის მეურნეობის სფერო: სასოფლო სამეურნეო პროდუქციის წარმოება, ფერმა და სხვ.; საბანკო სისტემა: კლიენტთა ანაბრების მართვა, კრედიტების დეპარტამენტი, რისკების ანალიზი და ა.შ., ბიზნესის და კომერციის სფერო: მარკეტინგის დეპარტამენტი, სასაწყობო მეურნეობა, სუპერ-მარკეტი. აგრეთვე სხვა სფეროები: აეროპორტი, ტრანსპორტი, აფთიაქები, ბიბლიოთეკა, მუზეუმები, კინო-თეატრები, სპორტი, სხვადასხვა სახის ელექტრონული ცნობარი და ა.შ.

საილუსტრაციო მაგალითების სახით წიგნში განხილულია მარტივი, საშუალო სირთულის და შედარებით რთული ამოცანებიც, რომელთა გარჩევა და ათვისება არ გაუჭირდებათ სტუდენტებს (გათვალისწინებულია მათი ცოდნის და უნარ-ჩვევების განსხვავებული დონე). კურსის წინაპირობა ითვალისწინებს, რომ მათ გავლილი აქვთ მონაცემთა ბაზების მართვის სისტემების (მაგალითად, SQL Server, NoSQL) თეორიული და პრაქტიკული საკითხები, აგრეთვე სტრუქტურული და ობიექტ-ორიენტირებული დაპროგრამების საკითხები (მაგალითად, C++/C#, Java, Android, Python, HTML/CSS, JavaScript, PHP/MySQL, XML, ASP.NET და სხვ. ენები და პროგრამული პლატფორმები).

სიახლე, რომელიც დამატებულია Desktop- და Web-აპლიკაციების პარაქტიკული ამოცანების და საკურსო პროექტის შესასრულებლად, არის ახალი ტექნოლოგიების, კერძოდ, WPF/WCF.NET ტექნოლოგიების, Office365-ის, ღრუბლოვანი და მობილური პლატფორმების და Azure SQL-ის გამოყენება პროგრამული პროდუქტების ასაგებად (ნახ.1). სტუდენტები პროგრამული ინჟინერიის კონცენტრაციის სპეციალობით ასრულებენ საკურსო პროექტებს Web-დეველოპმენტსა და სხვა ძირითად საგნებშიც. ამიტომ წინამდებარე ნაშრომში ჩვენ შემოვიფარგლებით ინტეგრირებული ჰიბრიდული და ღრუბლო-ვანი ტექნოლოგიებით, თუმცა საბოლოო არჩევანს თვით სტუდენტი აკეთებს თავისი სურვილისა და ინტერესების შესაბამისად. სამუშაო ჯგუფებიც (3-5 სტუდენტი) მათი ინიციატივით იქმნება (პროექტის მოცულობა და სირთულე განისაზღვრება ინდივიდუალურად, ჯგუფის წევრების მომზადების დონის მონაცემებით).



ნახ.1. პროგრამული პროდუქტების სახეები

განვიხილოთ კონკრეტული საილუსტრაციო მაგალითი ორგანიზაციაში ადამიანური რესურსების მართვის ერთი ამოცანის გადასაწყვეტად, შესაბამისი მხარდამჭერი პროგრამული სისტემის (Software) დეველოპმენტის მიზნით. საჭიროა რამდენიმე ეტაპის (კლასიკური სასიცოცხლო ციკლის - LifeCycle) შესრულება [1]:

1) ჩამოვაყალიბოთ შინაარსობრივად (კონტექსტ-ფორმით) ორგანიზაციაში ახალი თანამშრომლის მიღების არსებული ბიზნეს–პროცესი და ბიზნესწესები (ფირმის დებულების შესაბამისად). შედეგად უნდა მივიღოთ ასაგები სისტემის ფუნქციური მოთხოვნილებები: როლები და მათი ფუნქციები (ანუ რომელი თანამდებობრივი როლი მონაწილეობს ახალი თანამშრომლის მიღების პროცესში და რა მოვალეობებს ასრულებენ აქ ისინი);

2) ობიექტ-ორიენტირებული მოდელირების მიზნით გამოიყენება უნიფიცირებული მოდელირების ენის (UML ტექნოლოგიის) UseCase და Activity დიაგრამები [2,3]. მოდელების ასაგებად გრაფიკული სახით ჩვენ ვიყენებთ Microsoft Visual Studio.NET (vers.4.5) ინტეგრირებულ გარემოს Modeling Projects-ის საფუძველზე ან Sparx Systems Enterprise Architect (v.18.2) [4-6];

3) ბიზნესპროცესების და ბიზნესწესების საფუძველზე აგებული მოდელის (Activity-D) შესაბამისად ამოცანისათვის: „ორგანიზაციაში ახალი თანამშრომლის მიღება და ხელფასის დანიშვნა“ უნდა აიგოს კონკრეტული როლ(ებ)ის კომპიუტერთან მუშაობის „სცენარი“ UML-ის მიმდევრობითობის დიაგრამის სახით (Sequence-D);

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

5) მონაცემთა ბაზის დაპროექტება და აგება დასმული ამოცანისათვის. საჭიროა გამოვლინდეს დოკუმენტთა ფორმები და ინფორმაცია: *საწყისი* (ნორმატიული: თანამდებობები, ხელფასები, დაქვითვების სკალა; *ოპერატიული:* ყოველდღიური ან თვიური აღრიცხვის ტაბელები, შვებულების განრიგი, საავადმყოფო ბიულეტენი ან ფორმა-100); *გამომავალი* (უწყისი თანამშრომელთა ანგარიშებზე ხელფასების გადასარიცხად);

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

ან 6-ბ) განისაზღვროს NoSQL ტიპის მონაცემთა ბაზა, პროგრამულად აიგოს მისი კოლექციები და შეივსოს დოკუმენტებით. აიგოს კოლექციებსშორისი კავშირები MongoDB Compass-ს (ან Shell) პაკეტით;

7) შემუშავდეს მომხმარებელთა ინტერფეისები (სამუშაო ფორმები), MsVisual_Studio.NET Framework ინტეგრირებულ გარემოში Visual C# და WPF Application პროექტის სახით (ან სხვა პროგრამულ გარემოში), შემუშავდეს ხელფასის დარიცხვის ამოცანის პროცესების შესრულების მეთოდები;

8) შემუშავდეს მომხმარებელთა ინტერფეისებისთვის ხელფასების MsSQL Server (ან MongoDB) ბაზასთან კავშირის და CRUD ოპერაციები (მონაცემთა ამორჩევის, ჩამატების, წაშლის და მოდიფიკაციის მეთოდები), შესაბამისი პროგრამული კოდებით;

9) შეიქმნას საილუსტრაციო სტანდარტული მოთხოვნები, რომლებიც განთავსდება Button ღილაკების პანელზე (SQL- ან NoSQL ბაზასთან სამუშაოდ);

10) ჩატარდეს აგებული სისტემის ტესტირება კონკრეტულ მონაცემებზე;

11) შეიქმნას პროგრამული პროდუქტის დისტრიბუციული ვერსია (საინსტალაციო პაკეტი - გასაყიდად);

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

შენიშვნა:

საკურსო პროექტის მოცემულობა და მისი დეტალების დაზუსტება ხდება ინდივიდუალურად ან მუშა ქვეჯგუფში პედაგოგის დახმარებით. დისკუსიაში მონაწილეობა სასურველია ჯგუფის სხვა სტუდენტებისთვისაც. ასეთ პროცესში ხშირად ვლინდება სტუდენტთა აქტიურობა ინოვაციური წინადადებების ფორმირებით,

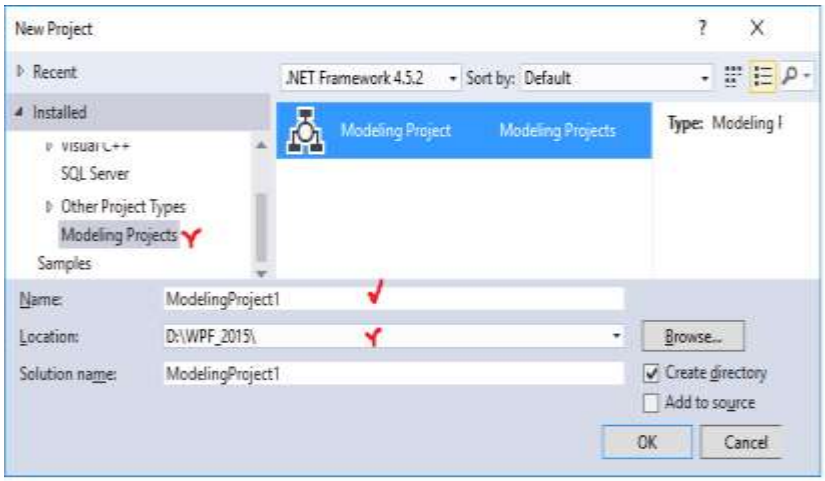
I თავი

საპროექტო ობიექტის ბიზნესპროცესების აღწერა და პროგრამული სისტემის მუშაობის სცენარი

1.1. Visual Studio.NET 4.5-ის სამუშაო გარემო

განვიხილოთ ბიზნესპროცესების ობიექტ-ორიენტირებული ანალიზისა და დაპროექტების კონკრეტული ამოცანის მოდელირების საკითხები MsVisual Studio.NET 2015 პლატფორმაზე [4]. დიაგრამების ასაგებად გამოიყენება ModelingProject შაბლონი. (შეიძლება Ms Visio პაკეტის გამოყენებაც).

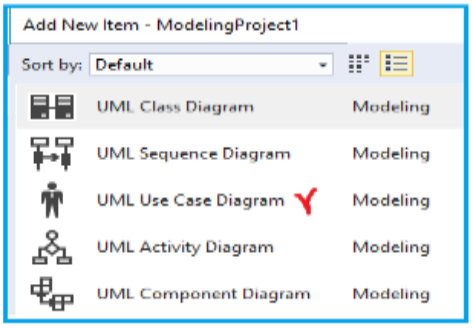
MsVisual Studio.NET 2015 პლატფორმის სამუშაო გარემოში ვირჩევთ სტრიქონს – Modeling Projects (ნახ.1.1), აგრეთვე ვუთითებთ Name-ს და Location-ს. ბოლოს OK.



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

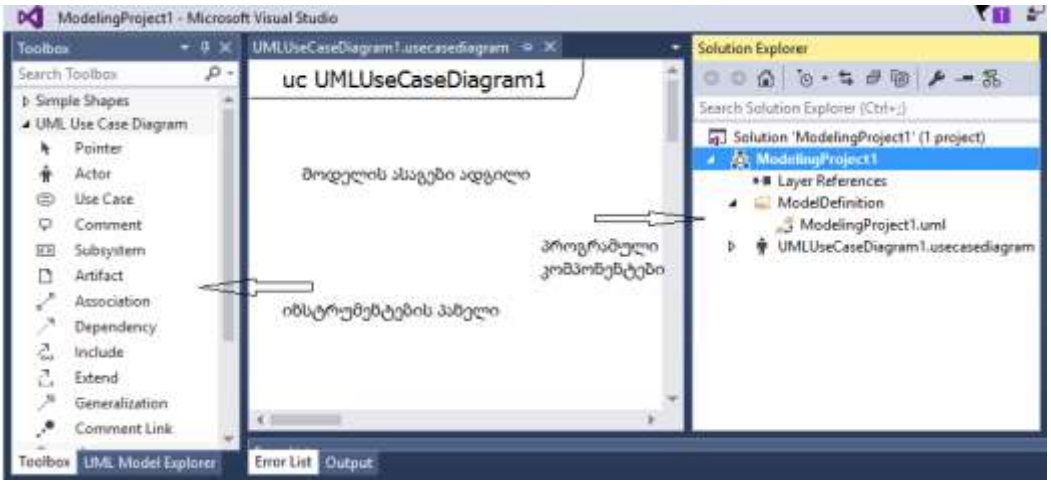
თუ სახელი არ მიეთითება, მაშინ სისტემა თვითონ ქმნის პროექტს ModelingProject1 სახელით.

Sollution Explorer-ში ჩვენ ვირჩევთ Add new Item-ით ჩვენთვის საჭირო ფუნქციას, ამ შემთხვევაში Use Case Diagram (ნახ.1.2).



ნახ.1.2

მიიღება 1.3 ნახაზზე ნაჩვენები საწყისი მდგომარეობა.

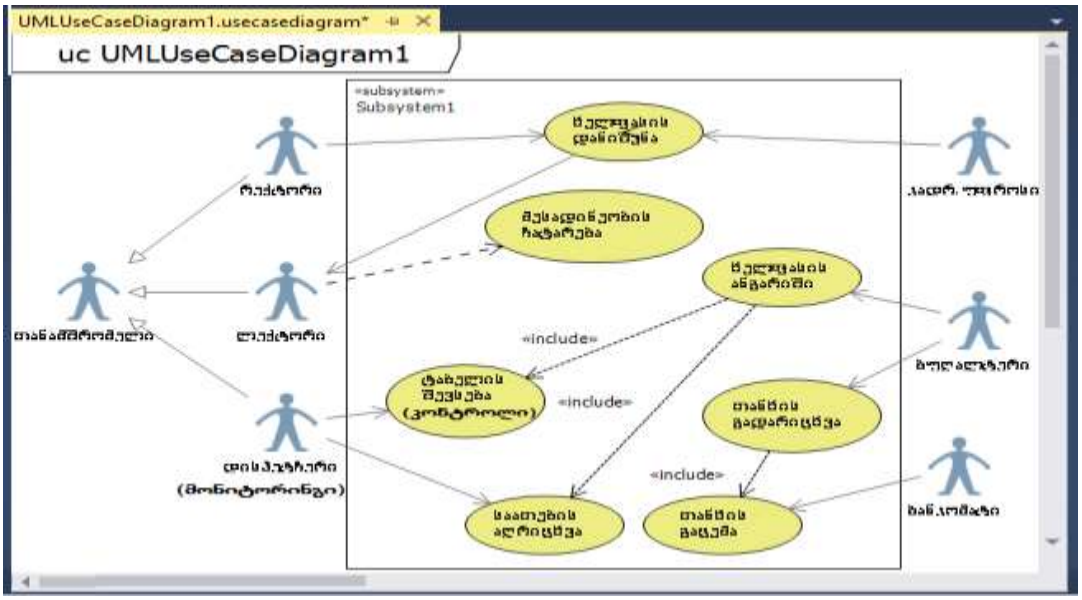


ნახ.1.3. Use Case მოდელის აგების ინტერფეისი
(MsVisualStudio .NET 2015)

1.2. პროგრამული სისტემის მოთხოვნილების განსაზღვრა UML მეთოდოლოგიით

1.2.1. სისტემის „როლები/ფუნქციების“ (UseCase) მოდელის დაპროექტება

1.4 ნახაზზე მოცემულია ხელფასის დარიცხვის ამოცანის ბიზნესპროცესში მონაწილე მომხმარებელთა როლები (Actors) და ფუნქციები (Actions). აგებულია მისი გამოყენებით-შემთხვევათა (UseCase-D) დიაგრამა.



ნახ.1.4. UseCase დიაგრამის ფრაგმენტი

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

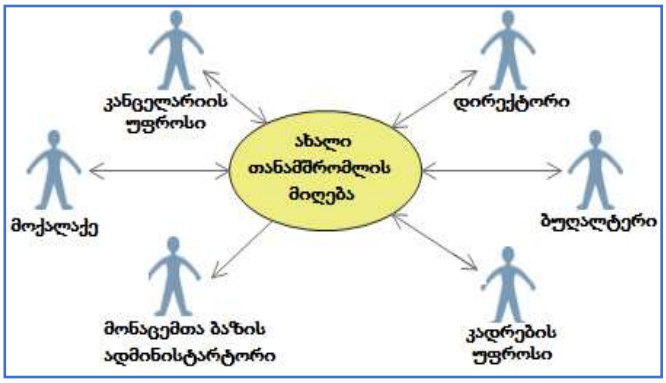
ფუნქცია (ოვალი) არის ქმედება, რომელსაც როლი ასრულებს. მაგალითად, „ხელფასის დანიშვნა”: პიროვნება გარკვეული საკონკურსო წესების საფუძველზე და პირადი განცხადებით საბუთებს წარუდგენს კადრების განყოფილებას (ან საკონკურსო კომისიას), სადაც გარკვეული ეტაპების გავლის შემდეგ, თუ საკითხი დადებითად გადაწყდა, კადრების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

ფუნქცია „მეცადინეობის ჩატარება“ ევალება ლექტორს და იგი აუდიტორიაში (ან დისტანციაზე) საკუთარი კომპიუტერით „რეგისტრირდება“ (სასწავლო პროცესის ჰიბრიდული რეჟიმის პერიოდი). .

ფუნქციები „მეცადინეობის ჩატარების აღრიცხვა“ და „ელ-ტაბელის შევსება“ ევალება კომპიუტერ-დისპეტჩერს (ან პროცესს აკონტროლებს მონიტორინგის სამსახურის სპეციალისტი - აფიქსირებს „მეცადინეობის ჩატარება-გაცდენას“). დისპეტჩერი ლექტორთა ელ-აღრიცხვის ჟურნალიდან მონაცემებს გადაიტანს ელ-ტაბელში, ესაა ყოველთვიური ელ-დოკუმენტი (ფაილი), რომელიც გადაეცემა ბუღალტერიას.

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

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



ნახ.1.5

1.2.2. სისტემის ბიზნესპროცესების და ბიზნესწესების (Activity-D) მოდელის დაპროექტება

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

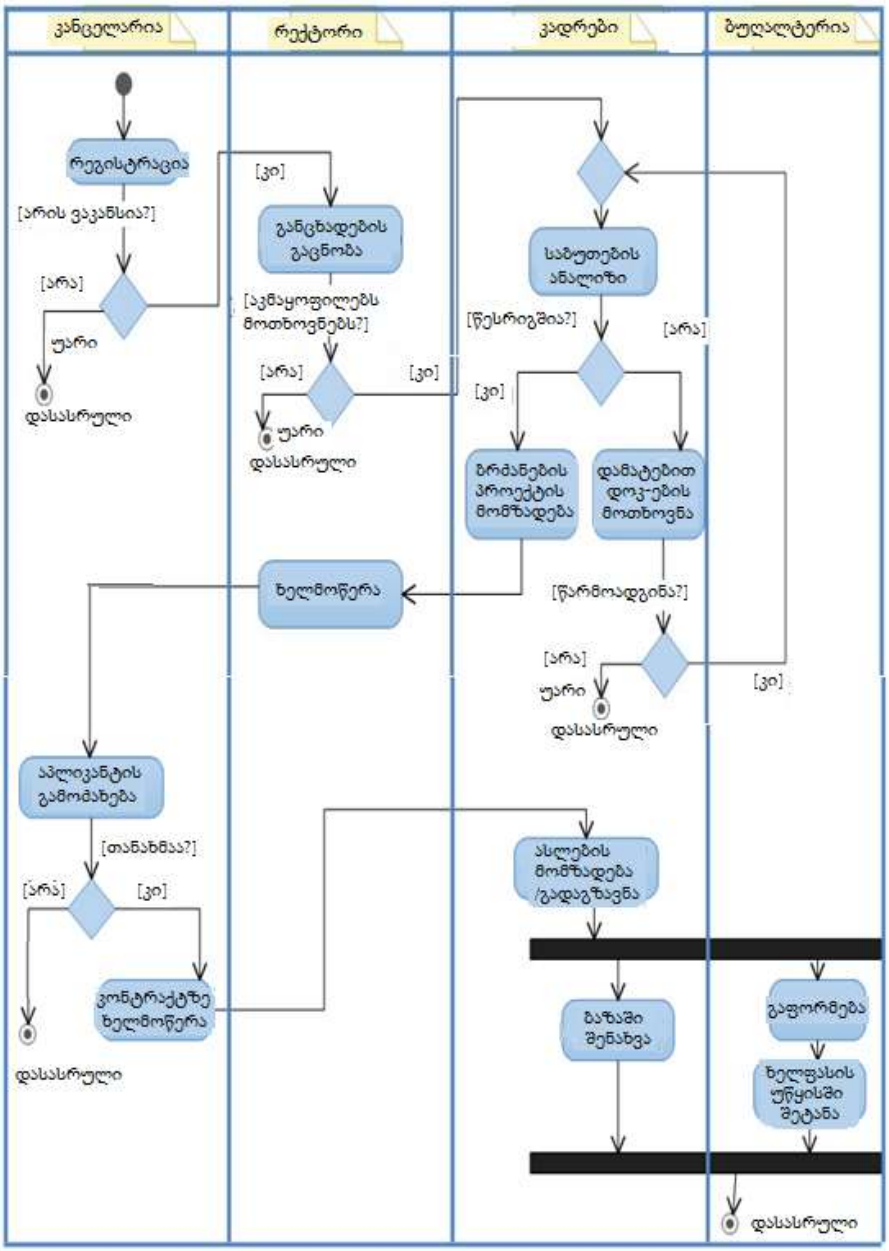
აქტიურობის დიაგრამაში ბილიკები ასახავს როლების მართვის სფეროებს. მაგალითად, პიროვნების განცხადება ამა თუ იმ ვაკანსიის დაკავების შესახებ შედის კანცელარიაში. იგი აქ გადის რეგისტრაციას.

თუ ორგანიზაციაში არ არის ვაკანტური ადგილი, მაშინ განმცხადებელი ღებულობს უარს. თუ ვაკანსია არსებობს, განცხადება გადაეცემა დირექტორს, რომელიც გადახედავს რა კანდიდატების მონაცემებს, პირადი მოსაზრებით აარჩევს საუკეთესოს, დაადებს რეზოლუციას და გადასცემს კადრების განყოფილებას. კადრების ინსპექტორი გადაამოწმებს ვაკანსიის არსებობას და პიროვნების შრომითი საქმიანობის დოკუმენტაციას.

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

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

Activity-D: ამოცანა - „ახალი თანამშრომლის მიღება“



ნახ.1.6. Activity დიაგრამის ფრაგმენტი

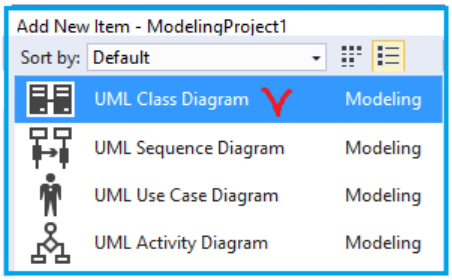
1.3. კლასების და კლასთაშორისი ასოციაციის (Class-D) მოდელის დაპროექტება და C#-კოდების გენერაცია

კლასი არაერთგვაროვან მონაცემთა სტრუქტურაა (int, string, double და სხვ.), რომელსაც ქმნის მომხმარებელი და მასში შესაძლებელია ზოგიერთს ხილვადობის private მოდიფიკატორი ჰქონდეს (ანუ იყოს დამალული „სხვებისთვის“).

ზოგადად, კლასი არის „დასახელების“, „კლასის მონაცემების“ და „კლასის მეთოდების“ ინკაფსულაცია. Ms Visual Studio.NET-ში კლასთა მოდელის ასაგებად Solution Explorer-ში ვიყენებთ Add New Item და მის ტიპს (ნახ.1.7).

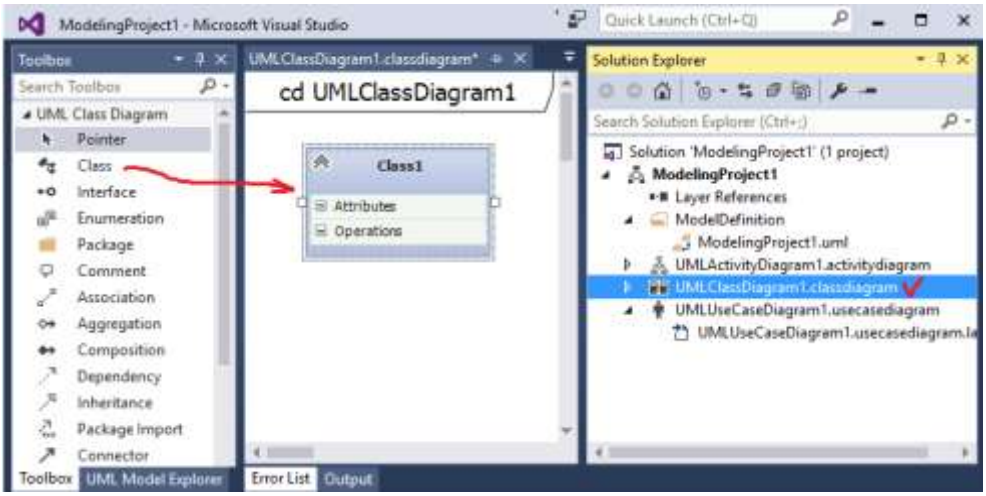
აქ შევარჩევთ სტრიქონს:

UML Class Diagram



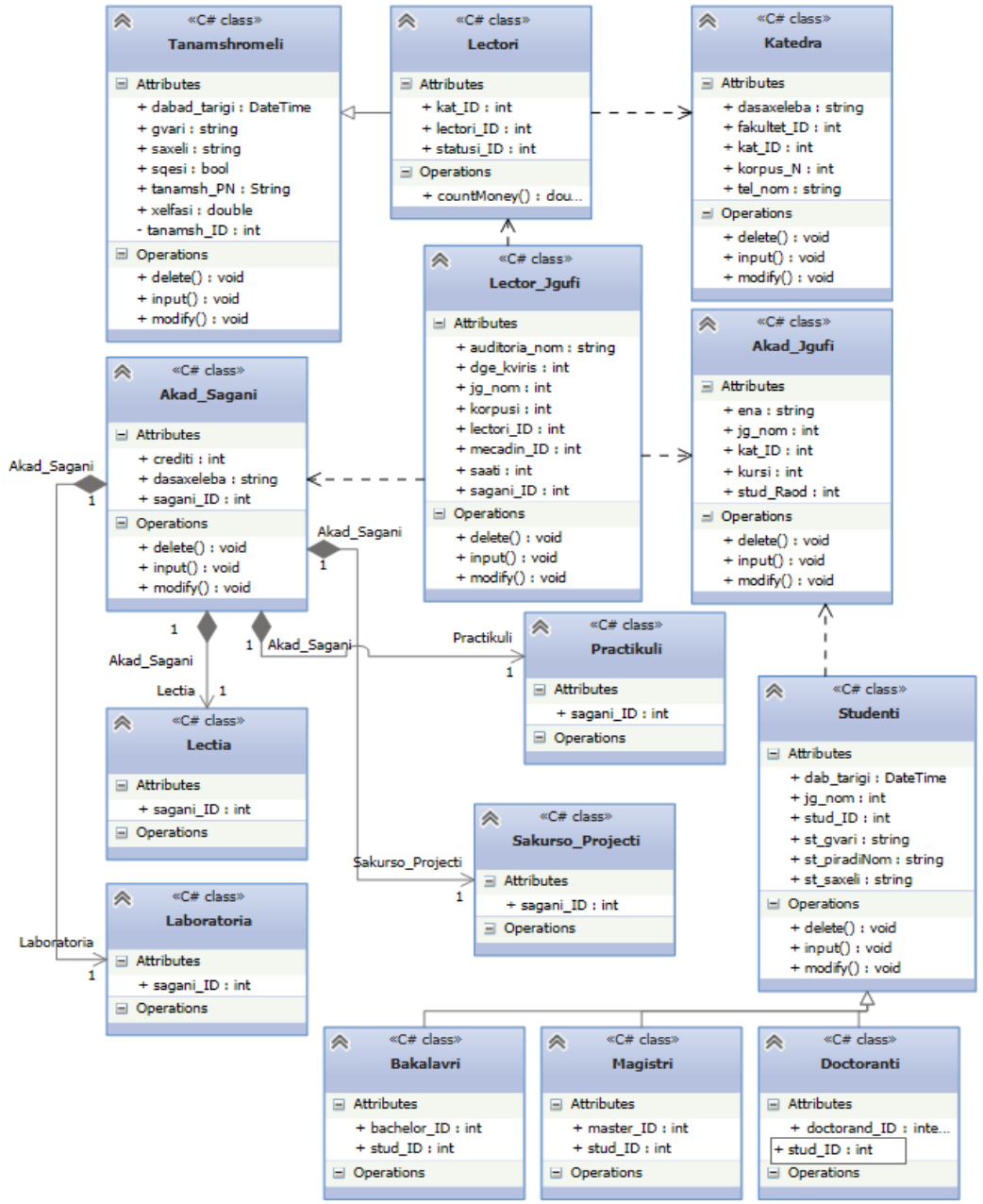
ნახ.1.7. კლასების დიაგრამის დამატება პროექტში

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



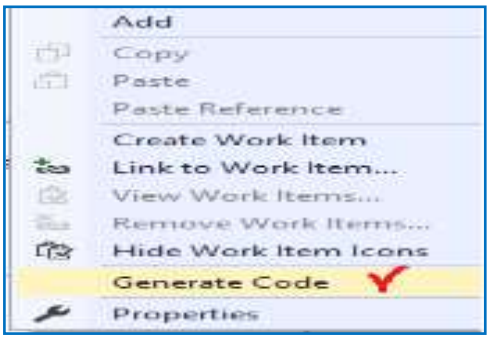
ნახ.1.8. კლასების დიაგრამის ასაგები ინტერფეისი

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



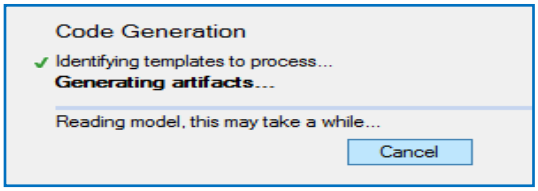
ნახ.1.9. სასწავლო პროცესის პროგრამული სისტემის კლასები
და კლასთაშორისი კავშირები (VisualStudio .NET 2015)

მიღებული კლასთა-ასოციაციის დიაგრამის საფუძველზე ვახდენთ C# კოდის გენერაციას კონტექსტური მენიუთი (ნახ.1.10).



ნახ.1.10

გამოდის შეტყობინება კოდის გენერაციის პროცესის შესახებ (ნახ.1.11),



ნახ.1.11

კოდის გენერაციის პროცესი გრძელდება რამდენიმე წამი/წუთი (ეს დამოკიდებულია კლასების რაოდენობასა და მათ ზომაზე). ჩვენ შემთხვევაში (ნახ.1.9) Solution Explorer-ში მიიღება ასეთი საილუსტრაციო შედეგი (ნახ.1.12). ქვედა ნაწილში ჩანს C# პროგრამების სახელები. ისინი სისტემამ შექმნა ავტომატურად. განვიხილოთ რამდენიმე C# კოდი Solution Explorer-იდან. მაგალითად, Lectori.cs (ლისტინგი 1.1) და Lectia.cs (ლისტინგი 1.2).

```
//--ლისტინგი_1.1 ----Lectori ---------------------------
// <auto-generated>
//------------------------------------------------------
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
public class Lectori : Tanamshromeli  // მემკვიდრეობითობა!!!
 {
    public virtual int lectori_ID { get; set; }
    public virtual int statusi_ID { get;      set; }
    public virtual int kat_ID { get; set; }
    public virtual double countMoney()
      { throw new System.NotImplementedException(); }
 }
```

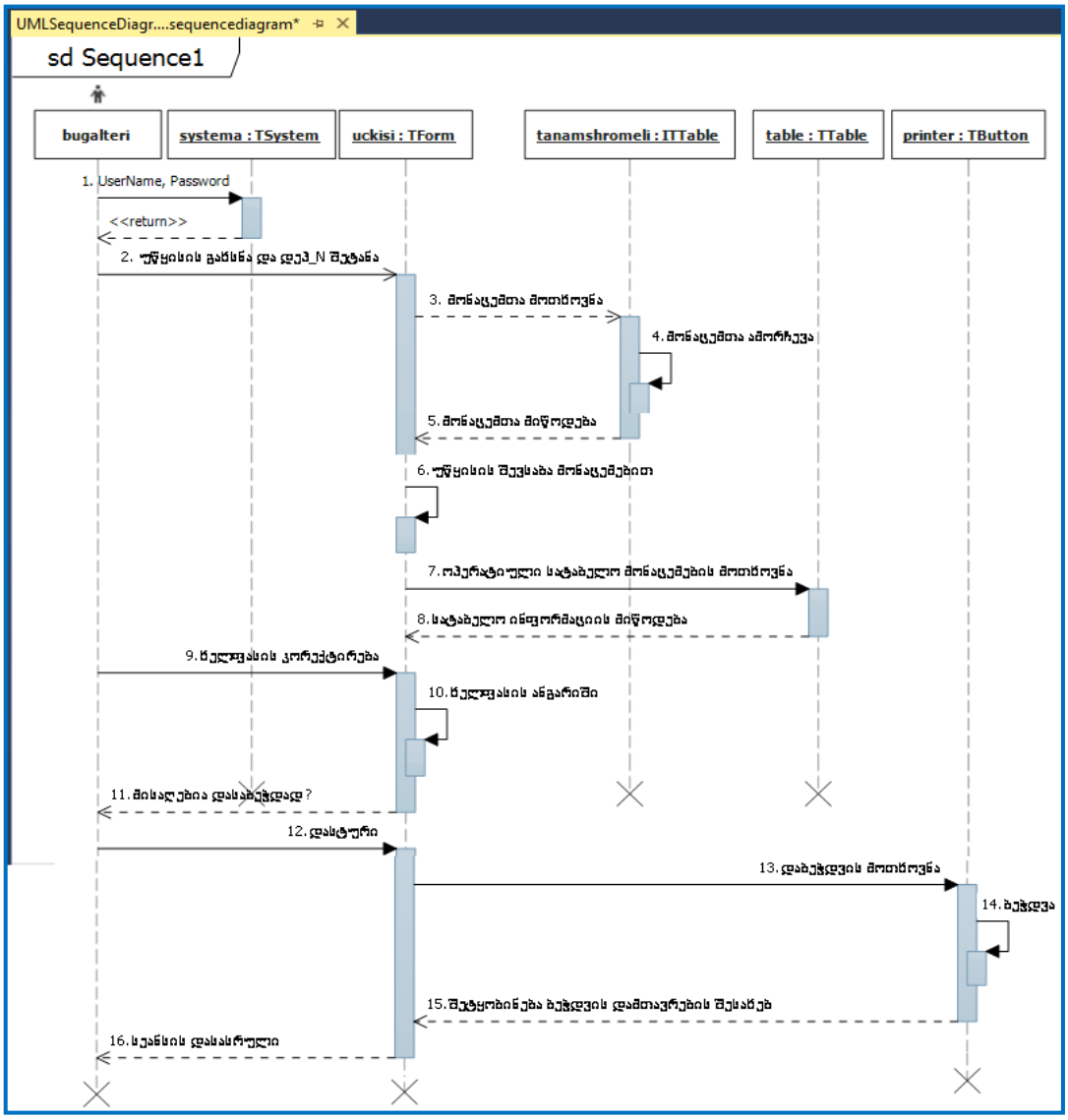
```
//-- ლისტინგი_1.2  ----Akad_Sagnebi --------------------
// <auto-generated>
//------------------------------------------------------
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

public class Akad_Sagani
{
  public virtual int sagani_ID { get; set; }
  public virtual string dasaxeleba { get; set; }
  public virtual int crediti { get; set; }
  public virtual Sakurso_Projecti Sakurso_Projecti // !!! კომპოზიცია
       { get; set; }
  public virtual Laboratoria Laboratoria // !!! კომპოზიცია
       { get; set; }
  public virtual Practikuli Practikuli // !!! კომპოზიცია
       { get; set; }
  public virtual Lectia Lectia   // !!! კომპოზიცია
       { get; set; }
  public virtual void input()
       {
         throw new System.NotImplementedException();
       }
  public virtual void delete()
        {
         throw new System.NotImplementedException();
       }
  public virtual void modify()
       {
          throw new System.NotImplementedException();
       }
 }
```

1.4. მომხმარებლის ინტერფეისი და პროგრამული სიტემის მუშაობის სცენარის (Sequence-D) დაპროექტება

სცენარი არის რომელიმე როლის (Actor) კლასებსა და მის ობიექტებთან მუშაობის პროცესის თანამიმდევრობის აღწერა კონკრეტული ამოცანის (Action) გადასაჭრელად კომპიუტერზე.

ამგვარად, ჯერ უნდა აიგოს სცენარი, თუ როგორ იმუშავებს მომხმარებელი კომპიუტერთან და შემდეგ მოხდეს მისი პროგრამული რეალიზაცია (მაგალითად, WPF ტექნოლოგიით, ან სხვა ინსტრუმენტის გამოყენებით). სცენარის ასაგებად ჩვენ გამოვიყენებთ მიმდევრობითობის მოდელის დაპროექტებას (Sequence-D დიაგრამას) (ნახ.1.13).

ამოცანა: „ხელფასის დარიცხვის უწყისის მომზადება და ბეჭდვა“



ნახ.1.1.3. მიმდევრობითობის დიაგრამა (Sequence–D)

მიმდევრობითობის დიაგრამა ინტერაქციული მოდელის ტიპია. იგი აღწერს როგორ და რა თანმიმდევრობით მუშაობს კლასის ობიექტების ჯგუფი ერთად. ამ დიაგრამებს იყენებენ პროგრამული უზრუნველყოფის დეველოპერები და ბიზნეს ანალიტიკოსები ახალი სისტემის მოთხოვნების დასადგენად. დიაგრამაზე შეტყობინებები (Messages) და ოპერაციები დალაგებულია ვერტიკალურად მათი შესრულების მიმდევრობით დროში.

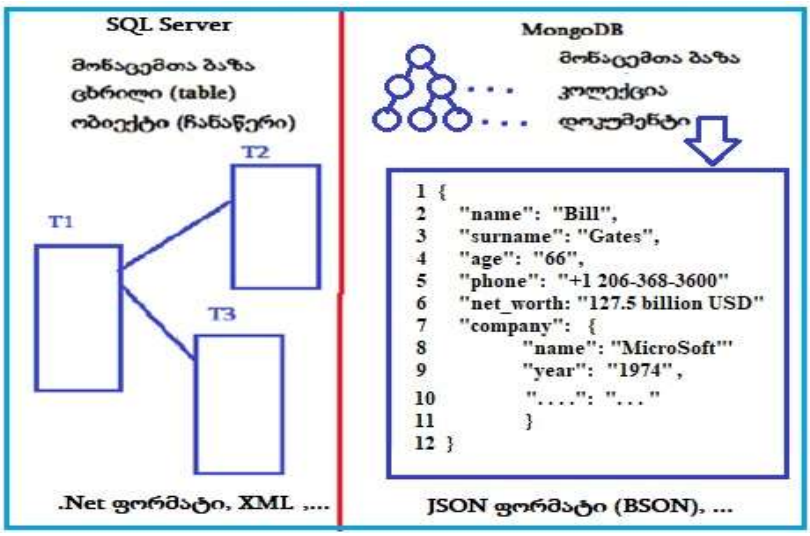
II თავი

პროგრამულ პროდუქტებში SQL | NoSQL ტიპის მონაცემთა ბაზების გამოყენება

პროგრამული პროდუქტების დეველოპმენტი ხშირად მოიცავს საკითხებს მონაცემთა ბაზების მენეჯმენტის ან მათი გამოყენების შესახებ.

დღეისათვის გამოყენებაშია მონაცემთა ბაზების სხვადასხვა ტიპები და პაკეტები, როგორიცაა კლასიკური რელაციური (SQL-ტიპის) და არარელაციური (NoSQL-ტიპის). მათ შორის არის საკმაო განსხვავება ტერმინოლოგიის და გამოყენების თვალსაზრისით [9].

2.1 ნახაზზე ნაჩვენებია ასეთი ბაზების სტრუქტურული განსხვავება (ცხრილური და იერარქიული მოდელების სახით).



ნახ. 2.1. SQL და NoSQL ბაზების შედარება

განვითარების დინამიკის, Big-Data და Cloud-App სისტემების თვალსაზრისით უპირატესობა NoSQL-ის მხარეზეა, თუმცა რელაციური ბაზები ინარჩუნებენ პრივილეგიებს კორპორაციულ სისტემებში. ხშირად ორივე ტიპი ერთდროულად გამოიყენება.

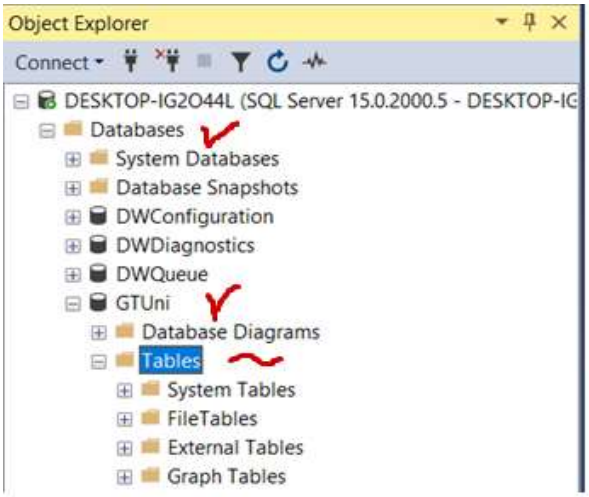
2.1. SQL Server მონაცემთა ბაზა

2.1.1. მონაქცემთა ბაზის და ცხრილების (Tables) შექმნა

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

დავალება_1: *აამუშავეთ Ms SQL Server Management Studio პაკეტი და შექმენით თქვენი პროგრამული აპლიკაციისთვის საჭირო მონაცემთა ბაზა და ცხრილთა (Tables) სტრუქტურები.*

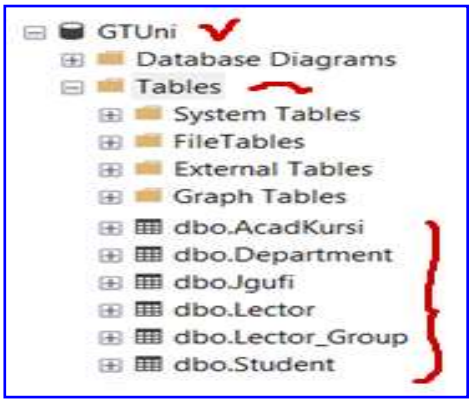
მაგალითად, 2.1 ნახაზზე ჩვენ შევქმენით უნივერსიტეტის მონაცემთა ბაზა (GTUni).



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

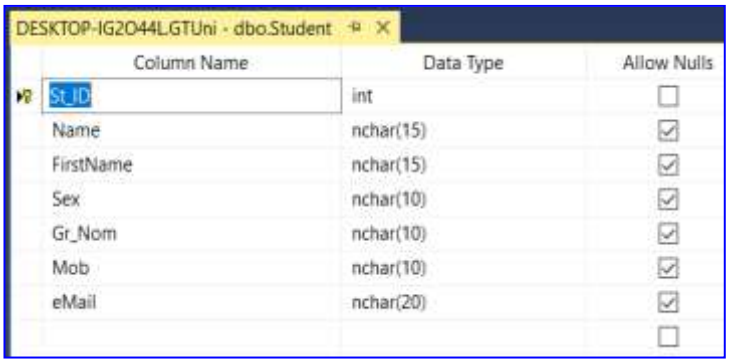
ჩვენი პროექტისთვის შევქმნათ ექვსი ცხრილი, რომლებშიც განთავსებული იქნება ინფორმაცია სტუდენტების, ლექტორების, ჯგუფების, აკადემიური კურსების (საგნების) და დეპარტამენტების შესახებ.

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



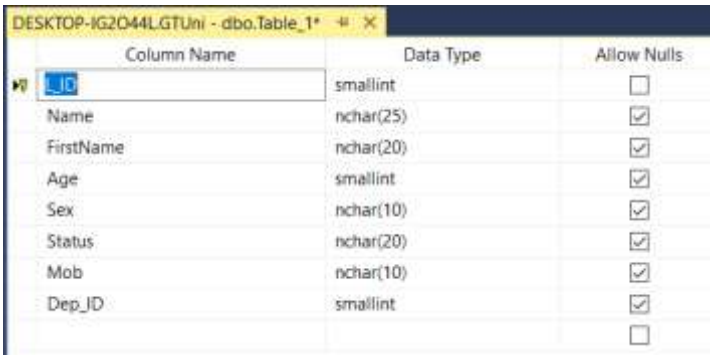
ნახ.2.2

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 ბიზნესწესს (არსებულ კანონზომიერებას), რომ ერთი სტუდენტი შეიძლება იყოს მხოლოდ ერთ ჯგუფში და ერთ ჯგუფში შეიძლება იყოს რამდენიმე (N) სტუდენტი. ასევე, ***ლექტორი*** (Lector) ასწავლის რამდენიმე (N) ჯგუფს, მაგრამ ჯგუფსაც ჰყავს რამდენიმე (M) ლექტორი. ესაა M:N კავშირი. მისი რეალიზაცია არაა შესაძლებელი ***ლექტორი***-ს და ***ჯგუფი***-ს პირდაპირი კავშირით *(განმეორებადი ველების პრობლემა !)*. ამისათვის შემოტანილია დამატებითი ცხრილი (რელაცია) ***ლექტორი_ჯგუფი*** (Lector_Group). მასში შედგენილი ინდექსი იქნება L_JG, რომელიც უკავშირდება ცალ-ცალკე ***ლექტორს*** (L_ID) და ***ჯგუფს*** (Gr_Nom). 2.3 ა–ვ ნახაზებზე მოცემულია ეს სტრუქტურები.



ნახ.2.3-ა.

„სტუდენტი“ ცხრილის სტრუქტურა



ნახ.2.3-ბ. „ლექტორი“ცხრილის სტრუქტურა

DESKTOP-IG2O44LG...- dbo.Department

Column Name	Data Type	Allow Nulls
DepID	smallint	☐
Name	nchar(30)	☑
Head_ID	smallint	☑
Adress	nchar(35)	☑
Tel	nchar(10)	☑

ნახ.2.3-გ. „დეპარტამენტი“ ცხრილის სტრუქტურა

DESKTOP-IG2O44LG....GTUni - dbo.Jgufi

Column Name	Data Type	Allow Nulls
Gr_Nom	nchar(10)	☐
Kursi	smallint	☑
ena	nchar(10)	☑
stud_raod	smallint	☑

ნახ.2.3-დ. „ჯგუფი“ ცხრილის სტრუქტურა

DESKTOP-IG2O44LG...i - dbo.AcadKursi

Column Name	Data Type	Allow Nulls
akadKurs_ID	smallint	☐
Name	nchar(50)	☑
crediti	smallint	☑
lectia	smallint	☑
semin_praqt	smallint	☑
laboratoria	smallint	☑
sakurso_pr	smallint	☑

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

DESKTOP-IG2O44LG...dbo.Lector_Group

Column Name	Data Type	Allow Nulls
L_JG	smallint	☐
L_ID	smallint	☑
Gr_Nom	nchar(10)	☑
akadKurs_ID	smallint	☑
semestri	smallint	☑
RoomNom	nchar(10)	☑
dataTime	datetime	☑

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

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

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.

დავალება_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 შესაბამისი საწყისი შევსებული ცხრილები.

DESKTOP-IG2O44L.GTUni - dbo.Student

St_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	ბახია	გიორგი ...	მამრ	108852	599001122	gbakhi@gmail.com
5	თურქია	ქეთი	მდედრ	108852	555334455	kturkia@gtu.ge
6	კოსტავა	დავით	მამრ	108851	577454545	dkosta@gtu.ge
7	მაისურაძე	რატი	მამრ	108851	577131313	rmaisu@gmail.com
8	მესხური	ცოტნე	მამრ	108853	577252525	tsotne7@gmail.com
9	ნებულიშვილი	ამიკო	მამრ	108850	599112112	anebuli@gtu.ge
10	ულრელიძე	ვახო	მამრ	108850	593669977	vughrel@gtu.ge
11	ხრიკული	მაია	მდედრ	108853	591222324	mkhriku@gtu.ge
21	ვაჩნაძე	საბა	მამრ	108950	577303030	svachna@gtu.ge
22	თოფურია	გიორგი ...	მამრ	108950	593757575	gtopuri@gmail.com
23	მამედოვა	სამირა	მდედრ	108950	577008899	smamed@gtu.ge
31	დუმბაძე	ნანა	მდედრ	108951	555777999	ndumba@gtu.ge

ნახ.2.4-ა. ცხრილი „სტუდენტი“

DESKTOP-IG2O44LG...- dbo.Department | DESKTOP-IG2O44LG...- dbo.Department

DepID	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-8; 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	ბაბტაძე	თენგიზ	53	მამრ	პროფ ...	577535353	1
3	თოფურია	ნინო	30	მდედრ	ასოც.პროფ...	555000200	1
4	ღვინეფაძე	გელა	57	მამრ	პროფ ...	577666777	1
5	თურქია	ეკა	31	მდედრ	მოწვ.პრო...	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	კიკნაძე	მზია	37	მდედრ	პროფ ...	591377373	2
52	კაკუბავა	ივერი	45	მამრ	პროფ ...	591515151	2
60	ახობაძე	მერაბ	55	მამრ	პროფ ...	577111111	4
61	კაშიბაძე	მარინა	34	მდედრ	ასოც.პროფ...	599775577	4
62	ოხანაშვილი	მაია	31	მდედრ	ასოც.პროფ...	599557755	4
80	ჩხაიძე	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.2.2-დ. ცხრილი „ჯგუფი“

DESKTOP-IG2O44LG...i - dbo.AcadKursi | DESKTOP-IG2O44LG...i - dbo.AcadKursi

akadKurs_ID	Name	crediti	lectia	semin_praqt	laboratoria	sakurso_pr
1	დაპროგრამების საფუძვლები	6	15	0	45	0
2	შესავალი ქსელურ 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 დაპროგრამება 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 დაპროექტება	5	145	15	0	15
9	ხელოვნური ინტელექტის საფუძვლები	5	15	30	0	0
10	კიბერუსაფრთხოების საფუძვლები	5	15	30	0	0

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

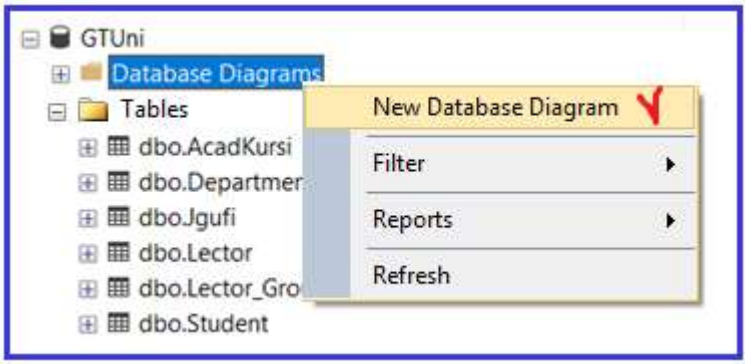
DESKTOP-IG2O44LG...dbo.Lector_Group | DESKTOP-IG2O44LG...dbo.Lector_Group

L_JG	L_ID	Gr_Nom	akadKurs_ID	semestri	RoomNom	dataTime
1	1	108850	6	1	06-207ა	2022-09-03 09:00:0...
2	1	108851	6	1	06-207ა	2022-09-03 11:00:0...
3	3	108852	6	1	06-204ა	2022-09-04 09:00:0...
4	3	108853	6	1	06-204ა	2022-09-04 13:00:0...
5	3	108850	7	1	06-230დ	2022-09-05 14:00:0...
6	3	108851	7	1	06-230დ	2022-09-05 14:00:0...
7	6	108952	5	2	06-207ა	2022-09-05 10:00:0...
8	6	108953	5	2	06-207ა	2022-09-04 10:00:0...
9	33	108850	8	2	06-405ბ	2022-09-06 09:00:0...
10	33	108851	8	2	06-405ბ	2022-09-06 09:00:0...
11	32	108058	10	2	09-208	2022-09-03 14:00:0...
12	51	108039	2	1	06-324	2022-09-03 10:00:0...

ნახ.2.4-ვ. ცხრილი „ლექტორის_ჯგუფი“
საგნების მიხედვით

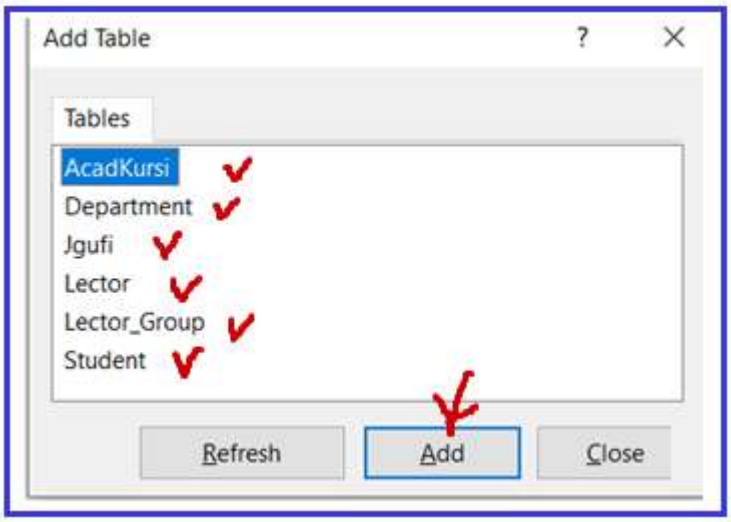
2.1.3. მონაცემთა ბაზის დიაგრამის აგება (Relationships in SQL)

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



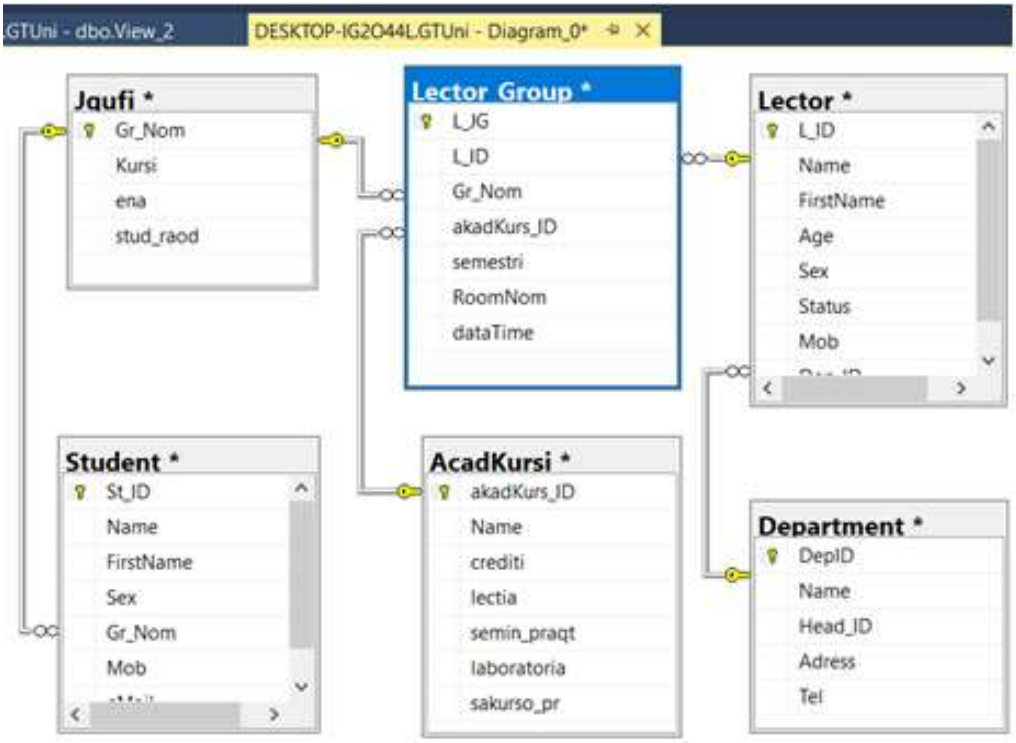
ნახ.2.5 ცხრილთაშორის კავშირების აგების სტრიქონი

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



ნახ.2.6 -ა

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



ნახ.2.6-ბ. 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 WPF აპლიკაციიდან მონაცემების ამორჩევა და მონაცემთა ბაზის განახლება C#-ის Insert, Update და Delete მეთოდებით. ეს საკითხები განხილულია მე-3 თავში.

NoSQL ტიპის მონაცემთა ბაზის MongoDB შექმნის და გამოყენების საკითხი განხილულია [10]-ში.

2.2. MongoDB Compass მონაცემთა ბაზა

ამჯერად განვიხილოთ NoSql ტიპის დოკუმენტური მონაცემთა ბაზის მართვის სისტემა MongoDB Compass, რომელიც ხშირად გამოიყენება დიდ მონაცემთა მენეჯმენტის სისტემებში, აქვს მეგობრული გრაფიკული ინტერფეისი და მონაცემთა დამუშავების მოქნილი სერვისები.

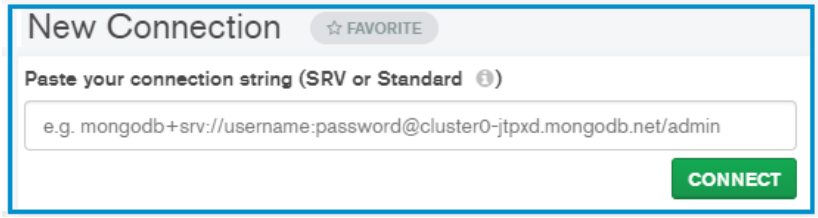
MongoDB Compass-ის ჩამოწერა და ინსტალირება, შესაბამისი ინსტრუქციებით, შესაძლებელია ამ მისამართზე:

https://www.mongodb.com/docs/compass/current/install/

ჩვენ განვიხილავთ უნივერსიტეტის მონაცემთა ბაზას. წინა პარაგრაფში ვიმუშავეთ SQL Server რელაციური მონაცემთა ბაზის მართვის სისტემაში, Management Studio გარემოში.

ახლა გამოვიყენებთ არარელაციური ტიპის MongoDB Compass პაკეტს და შევქმნით GTUni_Mongo მონაცემთა ბაზას, მის რამდენიმე კოლექციას (SQL-ში Tables) და კოლექციის დოკუმენტებს (SQL-ში ობიექტებს ან ცხრილის ჩანაწერებს, სტრიქონებს).

MongoDB Compass პიქტოგრამის ამუშავებით სისტემაში მომზადდება სპეციალური პლაგინები და საჭირო იქნება სერვერთან დაკავშირება Connection ღილაკით (ნახ.2.7).



ნახ.2.7. MongoDb Compass ბაზასთან მიერთება

2.8 ნახაზზე ნაჩვენებია სისტემაში არსებული ბაზების ნუსხა, მათ შორის ჩვენ მიერ ახლახან შექმნილი (ჯერ ცარიელი) GTUni_Mongo მონაცემთა ბაზა.

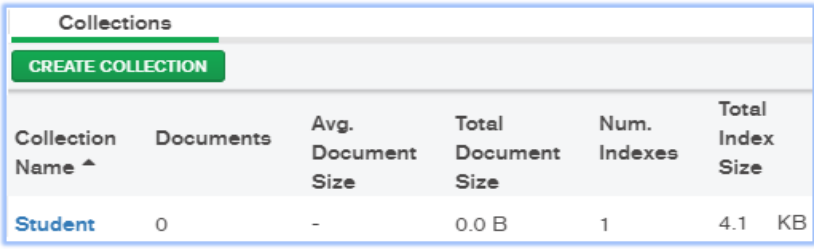
ნახ.2.8. საწყისი მდგომარეობა

დავიწყოთ მისი შევსება მონაცემებით.

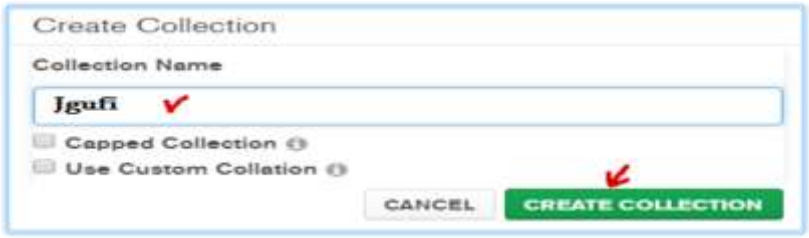
2.2.1. მონაცემთა ბაზის კოლექციების შექმნა

როგორეც აღვნიშნეთ, MongoDb ბაზა შედგება კოლექციებისგან (ანალოგი - Tables), რომლებშიც განთავსებულია დოკუმენტები (ანალოგი ცხრილის ობიექტები, ან სტრიქონები). უნივერსიტეტის მონაცემთა ბაზის ფრაგმენტი 6 კოლექციით უნდა ავაგოთ (სტუდენტები, ლექტორები, დეპარტამენტები, აკადემიური კურსები, ჯგუფები და მეცადინეობები. ეს უკანასკნელი „ჯგუფი-ლექტორის“ დამოკიდებულებაა).

გავააქტიუროთ GTUni_Mongo ბაზა და ეკრანზე გამოჩნდება Collections - ახალი კოლექციის შექმნის საშუალება (ნახ.2.9-ა,ბ).

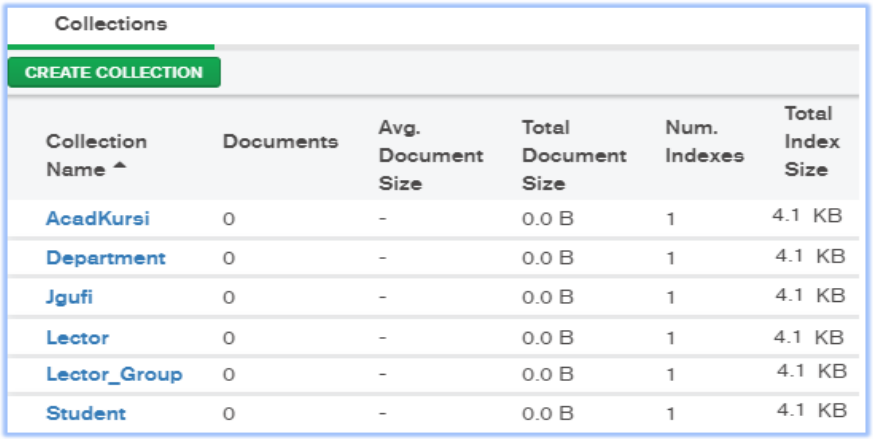


ნახ.2.9-ა. კოლექციის შექმნის დასაწყისი



ნახ.2.9-ბ. Jgufi - კოლექციის შექმნა

ამავე სტილში იქმნება დანარჩენი კოლექციებიც (ნახ.2.10).

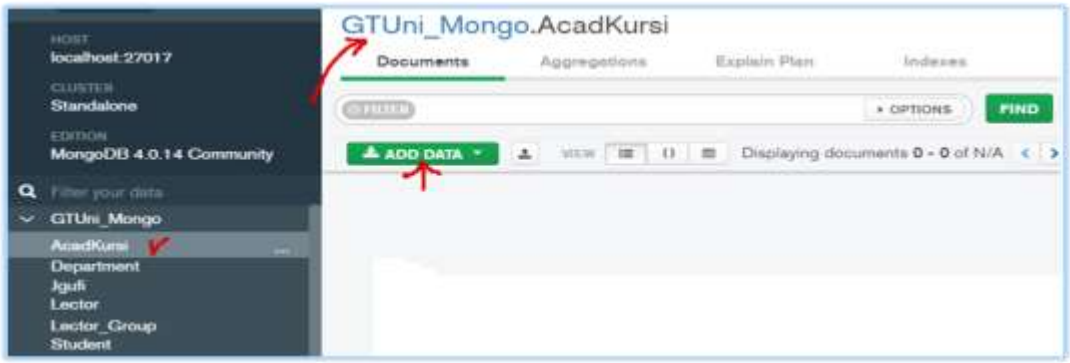


ნახ.2.10. ბაზის კოლექციები (ცარიელი)

შემდეგი ეტაპი კოლექციების შევსებაა, რომელიც Json ფორმატით ხორციელდება და მისი სტრუქტურაა: „ველი“: „მნიშვნელობა“.

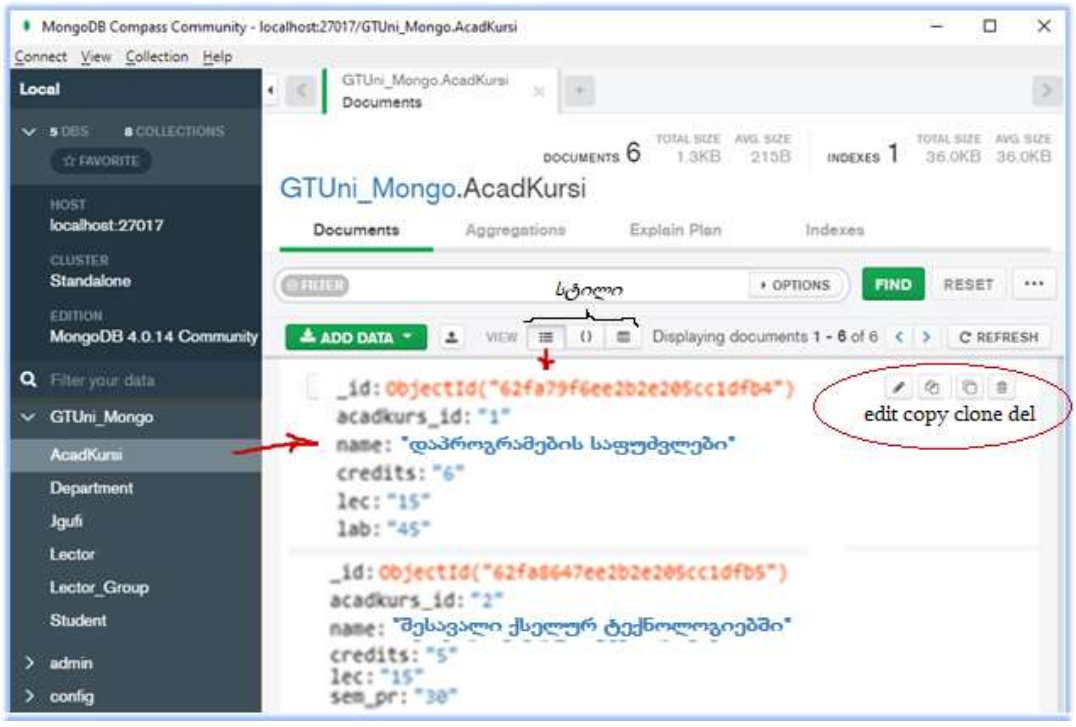
2.2.2. მონაცემთა ბაზის დოკუმენტებთან მუშაობა

2.11 ნახაზზე ნაჩვენებია GTUni_Mongo ბაზის 6 კოლექცია (მარცხნივ) და AcadKursi კოლექციისთვის დოკუმენტების (ანუ მონაცემების) შევსების პროცედურა.



ნახ.2.11. ADD DATA პროცედურა

შესაძლებელია მიმდევრობით რამდენიმე დოკუმენტის შეტანა (ნახ.2.12).



ნახ.2.12. შეტანილია 2 დოკუმენტი

მონაცემთა შეტანის სტილი სამგვარია: 1) „ველები“ და „მნიშვნელობები“ თავსდება ვერტიკალურად; 2) იწერება Json -ში და 3) ცხრილის სახით. მაგალითად, 2.13-ა ნახაზი 1-ელი სტილია, ხოლო 2.13-ბ კი მე-3, ანუ ცხრილური.


```
_id: ObjectId("62fa87e200fab3205c0692c8")
acadkurs_id: "3"
name: "ინფორმაციული ტექნოლოგიები"
credits: "5"
lec: "15"
sem_pr: "0"
lab: "30"
curs_project: "0"

_id: ObjectId("62fa888b00fab3205c0692c9")
acadkurs_id: "4"
name: "პროგრამული ინჟინერიის საფუძვლები"
credits: "6"
lec: "15"
sem_pr: "0"
lab: "30"
curs_project: "15"

_id: ObjectId("62fa88f400fab3205c0692ca")
acadkurs_id: "5"
name: "აპლიკაციების დაპროგრამება და მონაცემთა მენეჯმენტი"
credits: "5"
lec: "15"
sem_pr: "30"
lab: "0"
curs_project: "0"

_id: ObjectId("62fa897c00fab3205c0692cb")
acadkurs_id: "6"
name: "პროგრამული პროდუქტების დეველოპმენტი"
credits: "6"
lec: "15"
sem_pr: "0"
lab: "30"
curs_project: "15"
```


ნახ.2.13-ა. კოლექცია 4 დოკუმენტით

ADD DATA ▾ VIEW

AcadKursi

	_id ObjectId	acadkurs id String	name String	credits String	lec String	sem_pr String	lab String	project String
1	62fa79f6ee2b2e205cc1dfb4	"1"	"დაპროგრამების საფუძვლები"	"6"	"15"	"0"	"45"	"0"
2	62fa8647ee2b2e205cc1dfb5	"2"	"შესავალი ქსელურ ტექნოლოგიებში"	"5"	"15"	"30"	"0"	"0"
3	62fa87e200fab3205c0692c8	"3"	"ინფორმაციული ტექნოლოგიები"	"5"	"15"	"0"	"30"	"0"
4	62fa888beefab3205c0692c9	"4"	"პროგრამული ინჟინერიის საფუძვლები"	"6"	"15"	"0"	"30"	"15"
5	62fa88f400fab3205c0692ca	"5"	"აპლიკაციების დაპროგრამება და მონაცემთა მენეჯმენტი"	"5"	"15"	"30"	"0"	"0"
6	62fa897c00fab3205c0692cb	"6"	"პროგრამული პროდუქტების დეველოპმენტი"	"6"	"15"	"0"	"30"	"15"

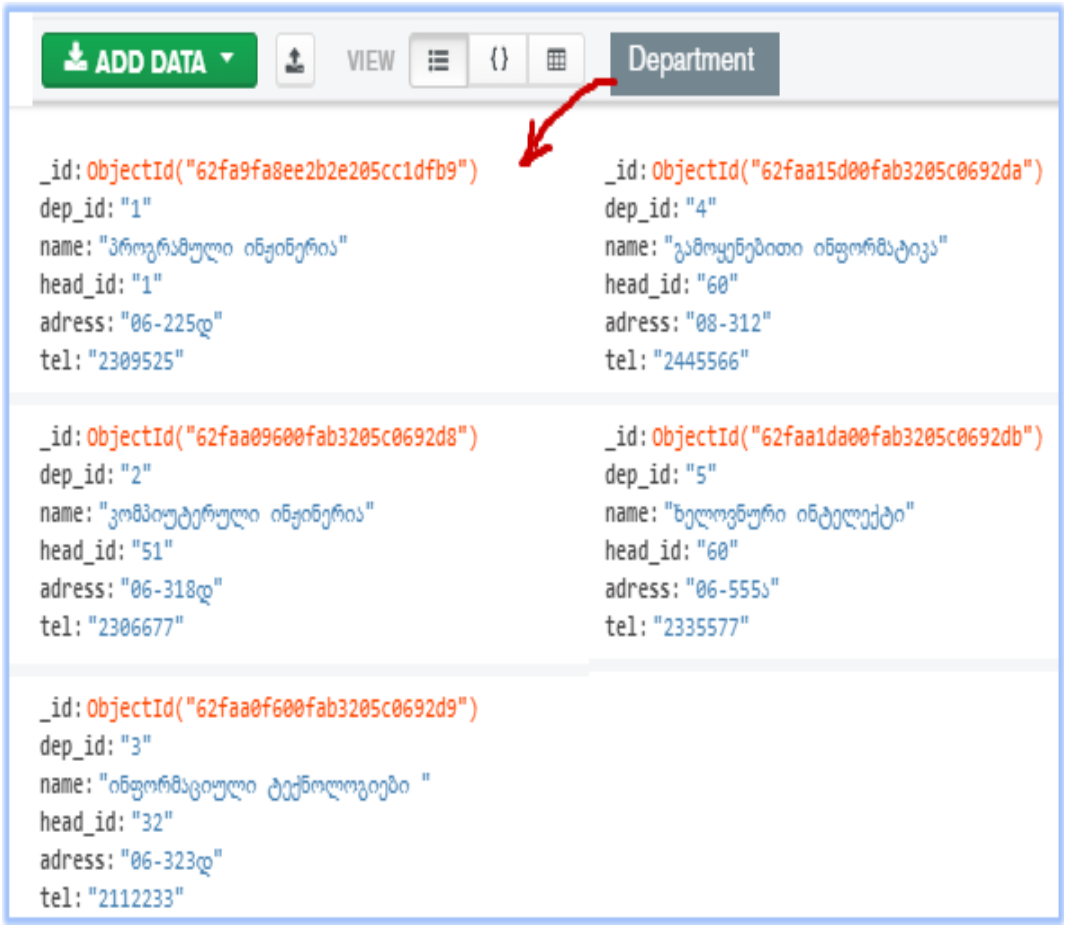
ნახ.2.13-ბ. Json ფორმატით დოკუმენტების ცხრილური წარმოდგენა

ყველა დოკუმენტს სისტემის მიერ ენიჭება _id - უნიკალური იდენტიფიკატორი.

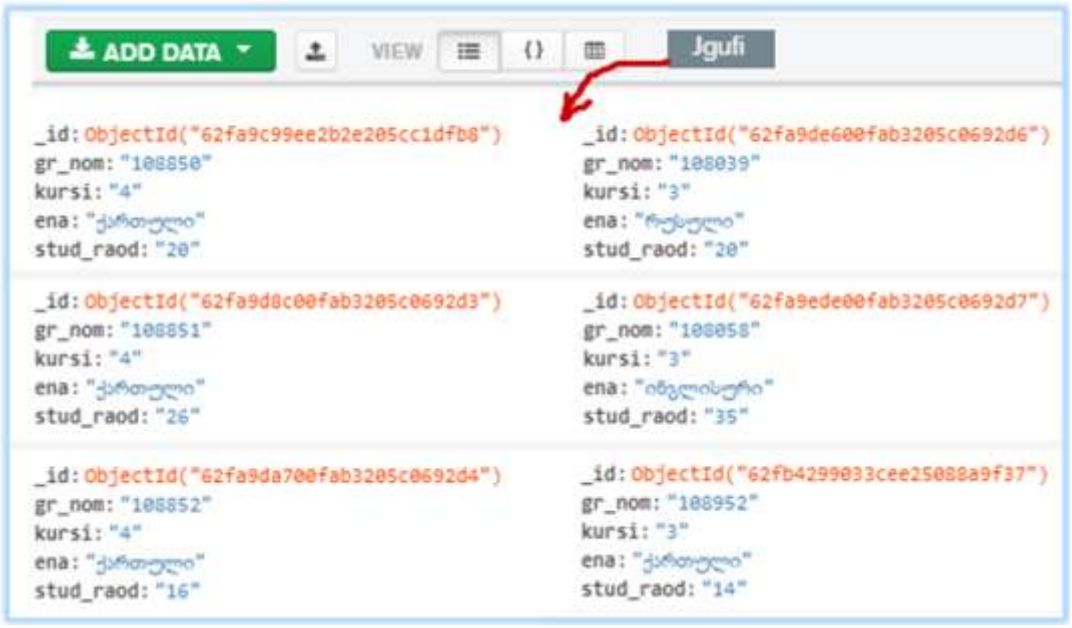
თუ შეტანილ მონაცემებში შეცდომაა და საჭიროა ცვლილებების განხორციელება, მაშინ უნდა ჩაირთოს edit-რედაქტირების რეჟიმი (ნახ.2.12-ზე ოვალის შიგნით). აქვეა copy, clone და delete ფუნქციათა ღილეკები (დოკუმენტებთან სამუშაოდ).

მაგალითად, თუ საჭიროა კოლექციაში ახალი დოკუმენტის დამატება, შეიძლება clone-ს ამოქმებება რომელიმე მსგავს არსებულ დოკუმენტზე. ეს დოკუმენტი ჩაიწერება ბოლოში და ჩვენ ხელით შევცვლით ზოგიერთი ველის მნიშვნელობას. დანარჩენი ველები უცვლელად ფიქსირდება.

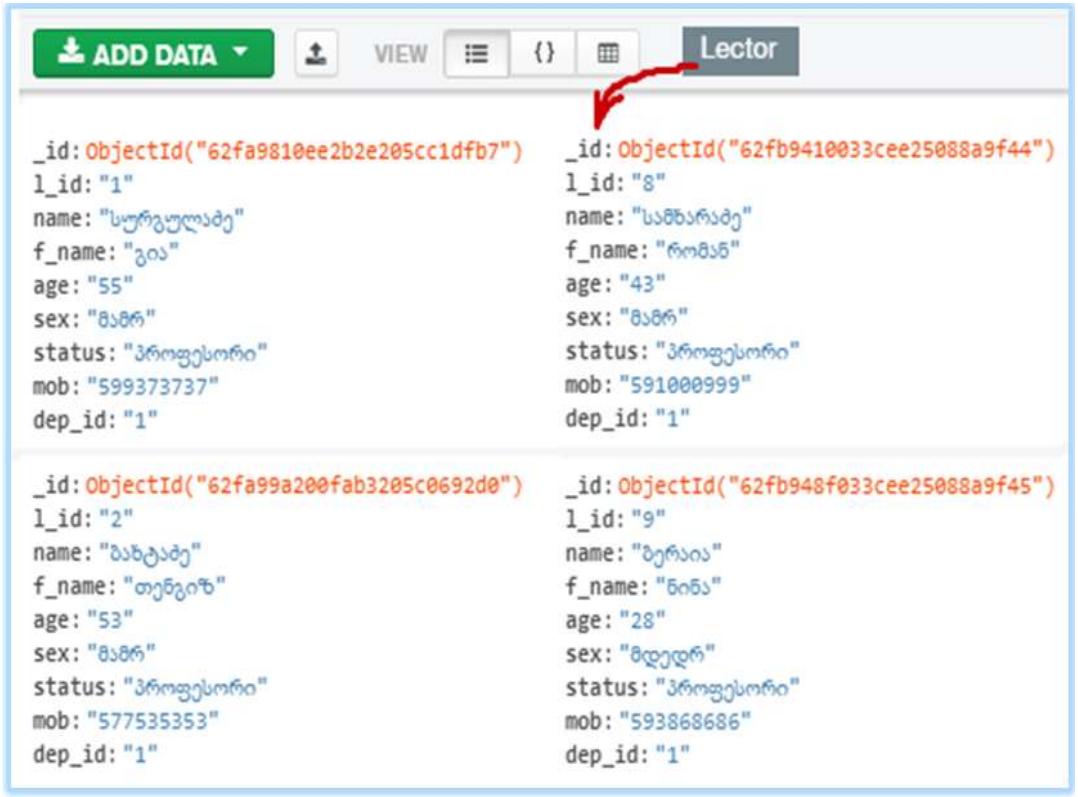
2.14 – 2.18 ნახაზებზე მოცემულია GTUni მონაცემთა ბაზის სხვა კოლექციათა დოკუმენტების ერთობლიობის ფრაგმენტები.



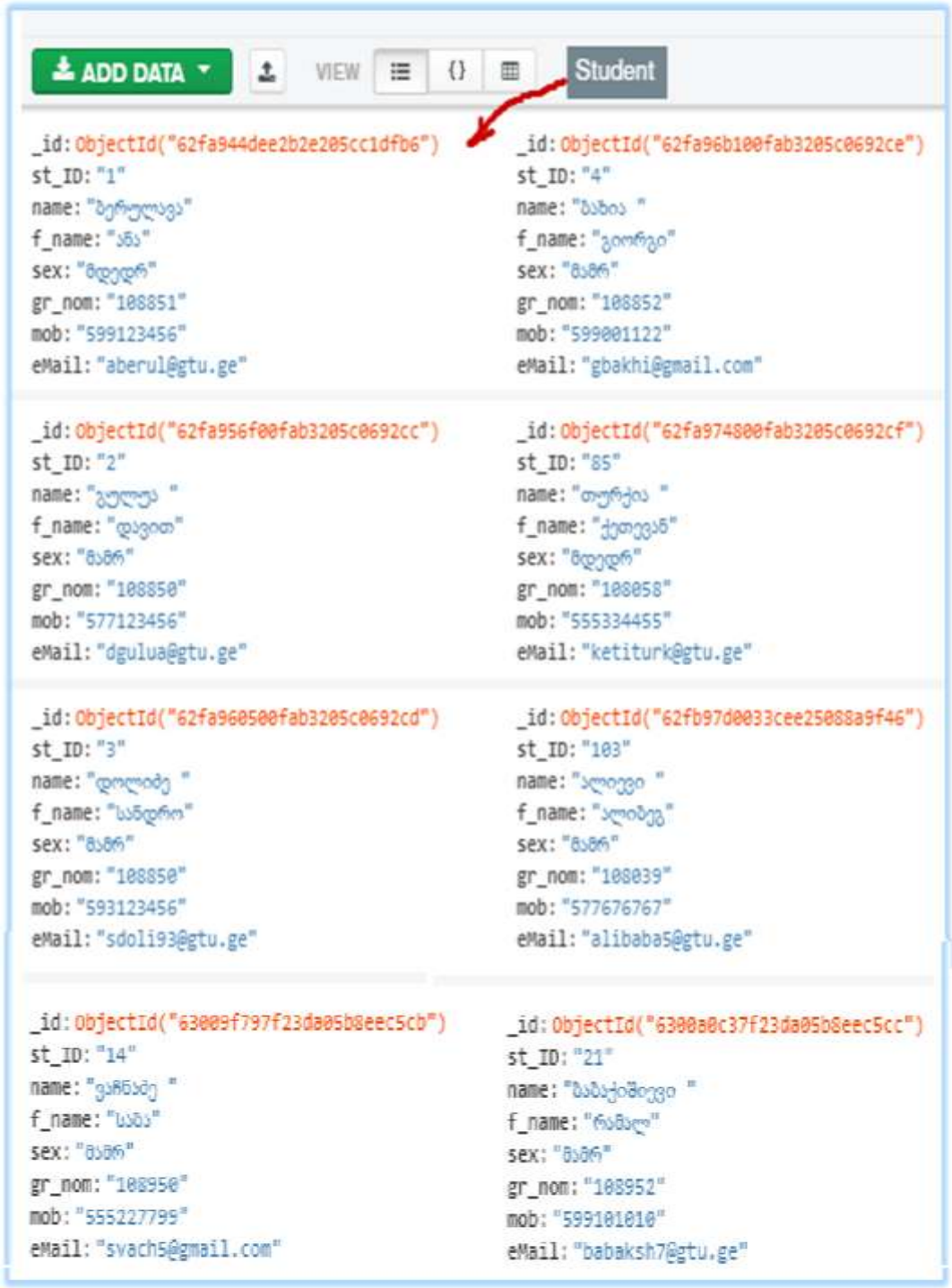
ნახ.2.14. Department - კოლექციის დოკუმენტები



ნახ.2.18. Jgufi - კოლექციის დოკუმენტები



ნახ.2.16. Lector - კოლექციის დოკუმენტები



ნახ.2.17. Student - კოლექციის დოკუმენტები

ADD DATA ▾ VIEW Lector_Group documents 12 REFRESH

	_id ObjectId	l_jg String	l id String	gr_nom String	acadkurs_id String	semestr String	roomNom String	dataTime String
1	62fb434f3d228025080e0099	"1"	"1"	"108850"	"6"	"1"	"06-207ა"	"2022-09-03 09:00:
2	62fb4690033cee25088a9f39	"2"	"1"	"108851"	"6"	"1"	"06-207ა"	"2022-09-03 11:00:
3	62fb46e3033cee25088a9f3a	"3"	"3"	"108852"	"6"	"1"	"06-204ა"	"2022-09-04 09:00:
4	62fb4752033cee25088a9f3b	"4"	"3"	"108853"	"6"	"1"	"06-204ა"	"2022-09-04 13:00:
5	62fb47bb033cee25088a9f3c	"5"	"3"	"108850"	"7"	"1"	"06-230დ"	"2022-09-05 14:00:
6	62fb4861033cee25088a9f3d	"6"	"3"	"108851"	"7"	"1"	"06-230დ"	"2022-09-05 14:00:
7	62fb48d2033cee25088a9f3e	"7"	"6"	"108952"	"5"	"2"	"06-207ა"	"2023-03-06 10:00:
8	62fb4de8033cee25088a9f3f	"8"	"6"	"108953"	"5"	"2"	"06-207ა"	"2023-03-06 10:00:
9	62fb4e34033cee25088a9f40	"9"	"33"	"108850"	"8"	"2"	"06-405ბ"	"2022-03-07 09:00:
10	62fb4ea2033cee25088a9f41	"10"	"33"	"108851"	"8"	"2"	"06-405ბ"	"2022-03-07 09:00:
11	62fb4edd033cee25088a9f42	"11"	"8"	"108858"	"10"	"2"	"09-208"	"2022-03-08 14:00:
12	62fb703a033cee25088a9f43	"12"	"9"	"108839"	"2"	"1"	"09-324დ"	"2022-09-08 10:00:

ნახ.2.18. Lector_Group - კოლექციის დოკუმენტები
(ცხრილური ფორმით)

2.2.3. მონაცემთა ბაზის კოლექციებსშორისი კავშირების აგება (Relationships in MongoDB)

MongoDB ბაზის შექმნისას, როგორც ვიცით, არაა საჭირო წინასწარ მონაცემთა მოდელის (ცხრილების - Tables) და მათ შორის კავშირების (Database Diagrams) აგება. MongoDB Compass -ის მაგალითზე ჩვენ ვნახეთ, რომ ბაზის შექმნა იწყება უშუალოდ კონკრეტული კოლექციის კონკრეტული დოკუმენტის (მაგალითად, სტუდენტის მონაცემები) ჩაწერით Json-ის ფორმატით „key“: „value“ (ველი - მნიშვნელობა).

MongoDB Compass -ის მართვის სისტემა თვითონ ქმნის ამის მიხედვით ბაზასაც, კოლექციასაც და დოკუმენტებსაც.

რეალურ მონაცემთა ბაზაში n-კოლექციაა. მაშინ ისმება კითხვა - როგორი კავშირები არსებობს კოლექციებს შორის, კოლექციის დოკუმენტებს შორის ? რეალურად ხომ ეს კავშირები არსებობს მონაცემებს შორის, ბუნებრივად: სემანტიკური, სინტაქსური და ა.შ.

დიახ, ზოგადად არსებობს, ესაა 1:1, 1:N, M:N კავშირები. ის, რაც ცხადად აღიწერება არსთა-დამოკიდებულების მოდელში (Entity-Relationship Model) SQL ბაზების შემთხვევაში და მას Database Diagrams ვუწოდებთ (ნახ.2.6).

წინამდებარე პარაგრაფში განვიხილავთ MongoDB-ს კოლექციების რელაციური კავშირების პროგრამული რეალიზაციის მეთოდებს და ინსტრუმენტულ საშუალებებს. არსებობს ორი ხერხი:

- კოლექციაში ჩაშენებული (embedded) დოკუმენტების გამოყენება;
- კოლექციაში მიმთითებლის (reference, ლინკი) გამოყენება სხვა კოლექციის დოკუმენტზე.

ჩვენ ზემოგანხილულ მაგალითებში უფრო ხშირად მეორე ხერხს ვიყენებდით, მაგალითად, Jgufi – Student (ლინკი gr_nom).

განვიხილოთ უფრო დეტალურად აღნიშნულ ორ ხერხს შორის შედარების და მათი გამოყენების ეფექტურობის შემთხვევები, ანუ რომელი რა დროს ჯობია გამოვიყენოთ მონაცემთა ბაზის კოლექციებს შორის 1:1, 1:N და M:N დამოკიდებულებათა არსებობისას.

მაგალითი_1,2: ჩაშენებული დოკუმენტები 1:1 და 1:N კავშირებით. ვიხილავთ სამ კოლექციას: Lector – Address – Department (ნახ.2.19).

აქ Lector – Address კოლექციებს შორის არსებობს 1:1 დამოკიდებულება. (ყოველ ლექტორს აქვს უნიკალური მისამართი. ესაა შემთხვევა, როდესაც ერთად მცხოვრები მეუღლეები ან მამა-შვილი არ მუშაობენ ერთ დეპ-ში).

Department – Lector კოლექციებს შორის კი არის 1: N კავშირი (ანუ 1 დეპარტამენტში რამდენიმე ლექტორია და ყოველ მათგანს აქვს უფლება იყოს მხოლოდ 1 დეპარტამენტში მთლიან ან ნახევარ შტატზე).

ნახ.2.19. Lector-კოლექცია ჩაშენებული მისამართის (1:1) და ჩაშენებული დეპარტამენტის (1:N) დოკუმენტებით

შენიშვნა: სამი კოლექციის ასახვა (პროგრამული რეალიზაცია) ერთში თეორიულად, მონაცემთა სტრუქტურების ოპტიმიზაციის თვალსაზრისით, ნიშნავს *დენორმალიზაციის* პროცესს.

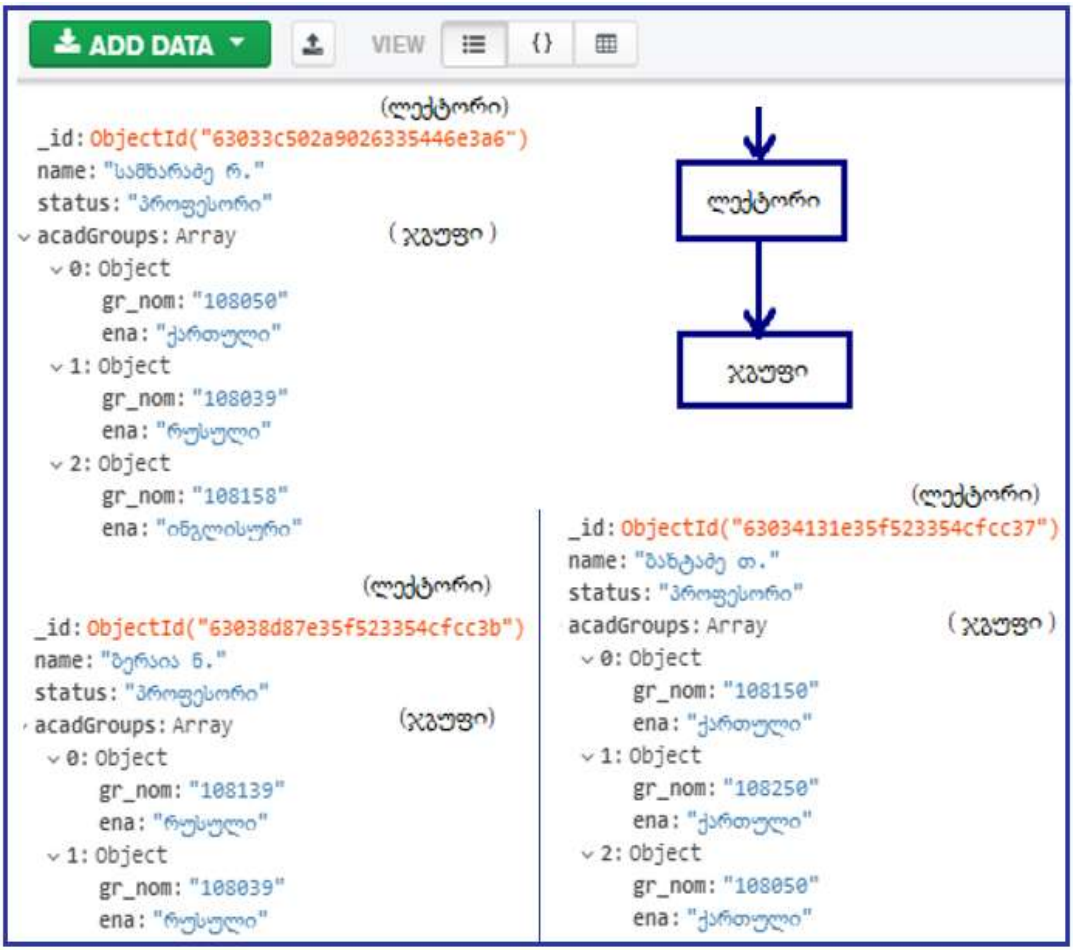
მონაცემთა სტრუქტურების *ნორმალიზაცია* (SQL ბაზაში) ნიშნავდა ერთი ცხრილის (table, მაგალითად, 1-ელ ნორმალურ ფორმაში) დაშლას ფუნქციონალური და ტრანზიტული კავშირების ანალიზის საფუძველზე, რამდენიმე მე-3 ნორმალური ფორმის ცხრილებად (მაგალითად 3 - დამოკიდებულებად).

***მაგალითი_3*:** ახლა განვიხილოთ M:N კავშირის მაგალითები ჩაშენებული კოლექციების და დოკუმენტების პროგრამული რეალიზაციის შემთხვევაში.

MongoDB Compass ბაზის ჩვენი მაგალითიდან გამოვყოთ Lector-ის და Jgufi-ის კოლექციები.

M:N დამოკიდებულების საილუსტრაციოდ განვიხილავთ ორ იერარქიას:

1. *ლექტორი -> ჯგუფი* (ნახ.2.20) და
2. *ჯგუფი -> ლექტორი* (ნახ.2.21).



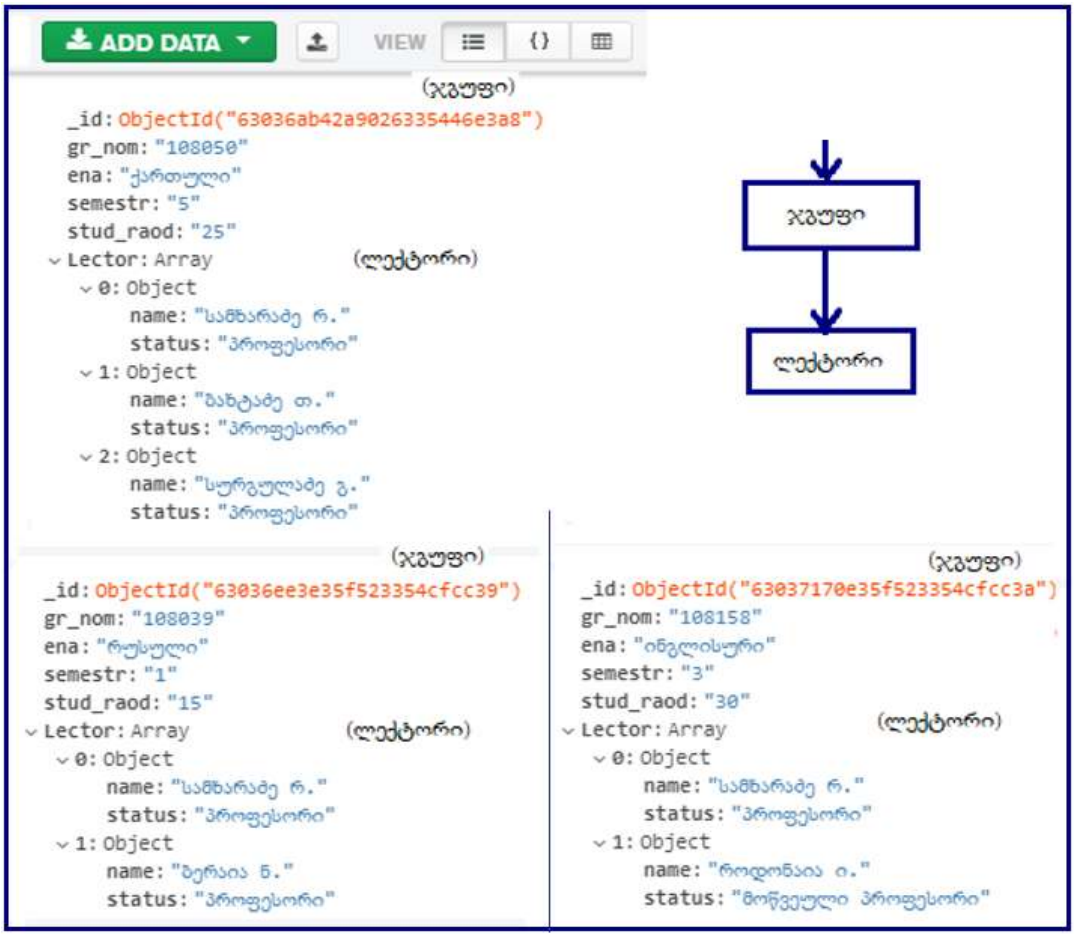
ნახ.2.20. კოლექცია Lector და მასში ჩაშენებული Jgufi-ს მონაცემები

როგორც ნახაზიდან ჩანს, ერთ ლექტორს ჰყავს სტუდენტთა რამდენიმე ჯგუფი და ეს ჯგუფები განსხვავებული მახასიათებლებით ხასიათდება. მაგალითად, გარდა ჯგუფის ნომრისა, აქ მნიშვნელოვანია „სწავლების ენა“, „სემესტრი“ და ა.შ.

მონაცემთა ძებნის პროცესი სწრაფია, თუ კონკრეტული ლექტორით ვეძებთ ჯგუფებს და ამ ჯგუფის სუდენტებს.

თუ მოთხოვნაში საჭიროა ჯგუფის_ნომრით (ან სტუდენტის_გვარით) ვიპოვოთ ლექტორი, მაშინ ძებნის პროცესის დრო იზრდება (იერარქიული ასიმეტრიულობის გამო) და საჭიროა დამატებითი მექანიზმების გამოყენება.

ერთ-ერთი ვარიანტია 2.21 ნახაზზე ნაჩვენები მონაცემთა სტრუქტურა, სადაც ბაზაში შესვლა ხორციელდება Jgufi - კოლექციის დოკუმენტის gr_nom ატრიბუტის მნიშვნელობით. ამ კოლექციის დოკუმენტის შიგნით ჩაშენებულია ლექტორთა კოლექციის დოკუმენტის შესაბამისი ველების მნიშვნელობები.



ნახ.2.21. კოლექცია Jgufi და მასში ჩაშენებული Lector-ის მონაცემები

ასეთ შემთხვევაში მონაცემთა ძებნის დრო იქნება სწრაფი. მაგრამ ჩვენ საბოლოო შედეგში გვაქვს ორი მოდელის (Lector -> Jgufi და Jgufi -> Lector), ანუ ორი სტრუქტურის რეალიზაცია (მონაცემთა დუბლირებით).

გადასაწყვეტი პრობლემის ოპტიმიზაციის თვალსაზრისით, მონაცემთა ძებნის დროისა ($T_{ძ}$) და მონაცემთა მოცულობის ($V_{ძ}$) კომპრომისული მნიშვნელობის განსაზღვრის მიზნით, იყენებენ *ინდექსირებული* ფაილების შექმნის ხერხს [10].

ზოგადად, ასეთი კლასის ოპტიმიზაციის ამოცანების გადაწყვეტა დამოკიდებულია სისტემის მომხმარებელთა მოთხოვნების ანალიზის შედეგებზე. კერძოდ, სამი (ან მეტი) დამოუკიდებელი ცხრილი, მიმთითებლიანი კავშირებით (reference) ერთმანეთთან, იკავებს მონაცემთა მოცულობის ($V_{მონაც.}$) თვალსაზრისით მინიმალურ ადგილს მეხსიერებაში (ნაკლებია მონაცემთა დუბლირება, სიჭარბე). მაგრამ მოთხოვნისათვის მონაცემთა ძებნის დრო არაა მინიმალური (ცხრილთაშორისი კავშირების /გადართვების/ მეტი რაოდენობის გამო). აღნიშნული საკითხის გადაწყვეტის მათემატიკური მოდელი და ალგორითმი მოცემულია მომდევნო პარაგფრაფში.

2.3. მონაცემთა სტრუქტურის ოპტიმიზაცია: სქემის ნორმალიზაცია / დენორმალიზაციის ალგორითმი

მონაცემთა ბაზის სქემის დაპროექტების პროცესში ცხრილების (ან კოლექციების) ნორმალიზაცია არის მათი ოპტიმიზაციის მექანიზმი. ნორმალიზაციას საფუძვლად უდევს რელაციურ დამოკიდებულებათა თეორია, რომელიც განიხილავს მათ ისეთ კლასებს (ოჯახებს) როგორიცაა: ფუნქციონალური, სრული ფუნქციონალური, ტრანზიტული, ფსევდოტრანზიტული, მრავალსახა და ზოგადი არაფუნქციონალური დამოკიდებულებანი [13]. მოკლედ განვიხილოთ ისინი.

დავუშვათ, მოცემულია საპრობლემო სფეროს მონაცემთა ბაზის ატრიბუტთა დასახელების სიმრავლე, U={A1,A2,...Aნ}, რომელზეც განსაზღვრულია R(U) რელაცია. იგი არის დომენთა დეკარტული ნამრავლის ქვესიმრავლე:

$$R(U) \subseteq dom(D_1) \times dom(D_2) \times \ldots dom(D_n)$$

A ატრიბუტების მნიშვნელობები (a) განისაზღვრება შესაბამის დომენთა სიმრავლეებზე $a_1 \in dom(D_1)$, $a_2 \in dom(D_2)$... და ა.შ. სტრიქონი რელაციის ელემენტია და მას კორტეჟი (ამონარჩევი) ეწოდება. R(U) ცხრილისათვის

(ატრიბუტთა დამოკიდებულებებისთვის) მოიცემა აგრეთვე P(U) პრედიკატი ანუ მთლიანობის შეზღუდვები.

დასაშვებია უნივერსალური რელაციის არსებობა, რომლის პროექციებიც არის ყველა დანარჩენი რელაცია $R_i(U_i)$. იგი განსაზღვრულია $U = \bigcup U_i$ ატრიბუტთა სიმრავლეზე.

მონაცემთა ბაზის საწყისი სქემა მოიცემა უნივერსალური რელაციის სახით:

$$\overline{S}^0 = \{\overline{R} = <U, P>\}$$

საჭიროა მოიძებნოს სქემა:

$$\overline{S} = \{R_i = <U_i, P_i> i = \overline{1,k} \,|\, R_i(U_i) = R[U_i]\}$$

რომელიც საწყისი სქემის ეკვივალენტურია და გარკვეული მოსაზრებით უკეთესი.

დასმული ამოცანის გადაწყვეტა წარმოებს ნორმალურ ფორმათა თეორიის გამოყენებით [10,13]. წინასწარ განვიხილოთ რელაციურ დამოკიდებულებათა კლასები, რომლებიც განისაზღვრება P მთლიანობის პრედიკატის საფუძველზე.

ფუნქციონალური დამოკიდებულება (ფდ). ამ კლასის დამოკიდებულების ცნებას აქვს პირველხარისხოვანი მნიშვნელობა. ვთქვათ, მოცემულია R(U) რელაცია და მისი X და Y ატრიბუტები. თუ R(U) რელაციაში X ატრიბუტის ნებისმიერ მნიშვნელობას შეესაბამება Y ატრიბუტის ერთადერთი მნიშვნელობა, ამბობენ, რომ R(U) დამოკიდებულება აკმაყოფილებს მთლიანობის შეზღუდვას - X -> Y ფუნქციონალურ დამოკიდებულებას.

სრული ფუნქციონალური დამოკიდებულება (სფდ). Y ატრიბუტი სრულ ფუნქციონალურ დამოკიდებულებაშია X ატრიბუტთან, თუ ის ფუნქციონალურადაა დამოკიდებული X ატრიბუტზე და ამასთანავე, არაა ფუნქციონალურად დამოკიდებული X-ის ნებისმიერ ქვესიმრავლესთან (X უნდა იყოს შედგენილი ატრიბუტი). სდფ აღინიშნება ორმაგი ისრით X => Y. ე.ი. ფორმალიზებულ ჩანაწერს ექნება შემდეგი სახე:

$$X \Rightarrow Y,\ if\ X \to Y \,\&\, (\overline{\exists} X' \subset X) : X' \to Y.$$

ტრანზიტული ფუნქციონალური დამოკიდებულება (ტფდ). X -> Y ფუნქციონალური დამოკიდებულება არის ტრანზიტული, თუ არსებობს ისეთი Z ატრიბუტი, რომ მართებულია (X -> Z & Z -> Y) დამოკიდებულების არსებობა.

სხვა დამოკიდებულებებს ამჯერად აღარ ვიხილავთ [10]. მხოლოდ შემოვიტანთ სამ განსაზღვრებას: 1ნფ, 2ნფ და 3ნფ -ის შესახებ.

განსაზღვრება 2.1: რელაცია იმყოფება პირველ ნორმალურ ფორმაში (1ნფ), თუ იგი მხოლოდ მარტივი დომენებისაგან შედგება, ანუ

$$\{\overline{\exists x} \subseteq U_R : x \equiv R_i \subseteq D_1 \times D_2 \times D_3 \times \ldots \times D_n, (D_1, D_2, D_3, \ldots, D_n) \subset U\}$$

დომენი არის მარტივი, თუ *არ არსებობს რელაცია,* რომელიც ამ დომენის ნაწილს წარმოადგენს, ანუ

$$\forall_i \overline{\exists R_i} : R_i \subseteq D,\ i \in M.$$

შენიშვნა: 1ნფ-ით გამოტანილი რელაციები განიხილება მარტივი ორგანზომილებიანი ცხრილების სახით, ატომარული სვეტებით.

განსაზღვრება 2.2: რელაცია იმყოფება მეორე ნორმალურ ფორმაში (2ნფ), თუ ის იმყოფება 1ნფ-ში და მისი ყოველი ატრიბუტი სრულ ფუნქციონალურ დამოკიდებულებაშია რელაციის გასაღებური ატრიბუტისაგან, ანუ

$$X \to Y,\ \ if\ \bar{\exists} Z \subset X : Z \to Y$$

განსაზღვრება 2.3: რელაცია იმყოფება მესამე ნორმალურ ფორმაში (3ნფ), თუ ის იმყოფება 2ნფ-ში და არც ერთი მისი ატრიბუტი არაა ტრანზიტულად დამოკიდებული არც ერთ შესაძლო გასაღებური ატრიბუტისაგან, ანუ

$$X \to Y,\ \ if\ \bar{\exists} Z : X \subset Z \,\&\, Z \to Y$$

საწყისი ცნებების შემოტანის შემდეგ შეიძლება განვიხილოთ მონაცემთა ბაზის სქემის ოპტიმიზაციის ამოცანა [10].

ზოგადად, საპრობლემო სფეროს კონცეპტუალლური მოდელის გათვალისწინებით მონაცემთა ბაზის ლოგიკური სტრუქტურის (სქემის) ოპტიმიზაციის ამოცანის გადაწყვეტა დამოკიდებულია სისტემის მომხმარებელთა მოთხოვნების ანალიზის შედეგებზე. რის საფუძველზეც წყდება ბაზის ცხრილების (კოლექციების) ასახვის და რეალიზაციის ნორმალიზაცია-დენორმალიზაციის ოპტიმალურობის საკითხი.

კერძოდ, სამი (ან მეტი) დამოუკიდებელი ცხრილი, მიმთითებლიანი კავშირებით (reference) ერთმანეთთან, იკავებს მონაცემთა მოცულობის ($V_{მონაც.}$) თვალსაზრისით მინიმალურ ადგილს მეხსიერებაში (ნაკლებია მონაცემთა დუბლირება, სიჭარბე). მაგრამ მოთხოვნისათვის მონაცემთა ძებნის დრო არაა მინიმალური (ცხრილთაშორისი კავშირების /გადართვების/ მეტი რაოდენობის გამო) [11, 12].

ანფ -> 1ნფ -> 2ნფ -> 3ნფ -> 4ნფ -> 5ნფ -> . ? . -> ბნფ

სადაც **ანფ** არანორმალიზებული ფორმაა (იერარქიული სტრუქტურა), **1ნფ** - პირველი ნფ (ერთი უნივერსალური რელაცია), ... , **ბნფ** - ბინარული ნფ (ორი ატრიბუტისგან შემდგარი. შეიძლება პირველი ატრიბუტი იყოს ნატურალურ რცხვთა დომენი). ნიშანი "?" მიუთითებს იმაზე, რომ მონაცემთა დამოკიდებულებათა თვისებების კვლევა შეიძლება გაგრძელდეს მათი შემდგომი ოპტიმიზაციის მიზნით. დღეისათვის მრავლად არსებობს სხვა სახის ნფ-ებიც, მაგრამ ნორმალიზაციის კლასიკურ თეორიაში ისინი ნაკლებადაა ასახული [13].

მონაცემთა ცხრილების დეკომპოზიციის დროს ბნფ-ები მიიღება სქემის დაპროექტების ცალკეულ ეტაპზე. არადეკომპონირებადი ცხრილისთვის კი საჭიროა ხელოვნურად ფიქტიური ატრიბუტის (ნატურალურ რიცხვთა სასრული სიმრავლე) შემოტანა. ინფორმაციის სემანტიკური მთლიანობის უზრუნველყოფა დეკომპონირებულ დამოკიდებულებათა შორის ხორციელდება ინდექსური კავშირების დუბლირების საშუალებით, ე.ი. შემოტანილია გარკვეული სიჭარბე. რაც მაღალია ნფ-ის რიგი, მით ნაკლებია ინფორმაციული და მეტია ინდექსური სიჭარბე:

1ნფ -> 2ნფ ->. . . -> ბნფ, **1ნფ -> 2ნფ-> . . . ->ბნფ,**

V_1-ინფ >= V_2-ინფ >=..>= $V_ბ$-ინფ, **V_1-ინდ <= V_2-ინდ <=...<= $V_ბ$-ინდ**

მონაცემთა სტრუქტურის *ოპტიმალური სიჭარბის* განსაზღვრის ამოცანა *განახლების დროის მინიმიზაციის* მოთხოვნით – არის გადასაწყვეტი (მიზანი). მისი ამოხსნის შედეგად შესაძლებელია მონაცემთა ბაზის სქემის ცხრილების ოპტიმალური ნფ-ების დადგენა, ანუ სად გადის „ნორმალიზაცია / დენორმალიზაციის“ ზღვარი.

საჭიროა შესაბამისი ალგორითმის აგება და პროგრამული რეალიზაცია, რის საფუძველზეც ავტომატიზებულად დაპროექტდება საპრობლემო სფეროს კონცეპტუალური სქემა (ER-მოდელი).

საწყის ცხრილთა სტრუქტურა შეიძლება ასე ჩავწეროთ:

$$\begin{cases} R_1(k_1, k_2, \ldots k_{n1}, A_1, A_2, \ldots, A_{a1}) \\ R_2(k_1, k_2, \ldots k_{n2}, B_1, B_2, \ldots, B_{a2}) \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ R_l(k_1, k_2, \ldots k_{nl}, Z_1, Z_2, \ldots, Z_{al}) \end{cases} \quad (1)$$

სადაც *k* ატრიბუტების გასაღებური, ხოლო *A-:-Z* არაგასაღებური ნაწილებია.

სემანტიკური მთლიანობა შედეგია იმ ფაქტისა, რომ სიმრავლე მიღებულია ერთი უნივერსუმის დეკომპოზიციით. თუ ჩავთვლით, რომ

$$k_1, k_2, \ldots k_{n1} \supseteq k_1, k_2, \ldots k_{n2} \supseteq k_1, k_2, \ldots k_{nl}$$

(1) სისტემის კომპოზიციით (*დენორმალიზაციით*), მაშინ შესაძლებელია ერთი შედარებით დაბალი ნფ-ის მიღება (რაც NoSQL-ის დროსაც გვჭირდება):

$$R(k_1 \ldots k_{n1}, A_1 \ldots A_{a1}, B_1 \ldots B_{a2}, \ldots Z_1 \ldots Z_{al}) \quad (2)$$

დავუშვათ აგრეთვე, რომ წინასწარ ცნობილია R_i -ის ცვლილების რაოდენობა μ_i დროის განსაზღვრულ ინტერვალში და მართებულია შემდეგი მოწესრიგება:

$$\mu_1 \geq \mu_2 \geq \ldots \geq \mu_l$$

(1) და (2) -სთვის მონაცემთა განახლების მოცულობები შესაბამისად გამოითვლება ასე:

$$Q_{dec} = \sum_{i=1}^{l} \mu_i * (n_i + a_i) \text{ და } Q_{com} = \mu_1 * (n_1 + \sum_{j=1}^{l} (a_j - r))$$

სადაც r – ატრიბუტების ის რაოდენობაა, რომლითაც სრულდება შეერთების (“Join”) ოპერაცია. შემდგომში შეიძლება იგნორირება.

თუ დავუშვებთ, რომ (1) და (2) ნფ-ებს შორის არსებობს შუალედური ნფ, მაშინ მისთვის განახლებათა მოცულობა შეადგენს:

$$Q = \sum_{j=1}^{s} \mu_j * (n_j + \sum_{k=1}^{l} a_k)$$

სადაც S ფდ-ების რაოდენობაა შუალედურ ნფ-ში. მაშინ, მართებულია გამოსახულება:

$$\mu_1 * (n_1 + \sum_{k=1}^{l} a_k) \geq \ldots \geq \sum_{j=1}^{s} \mu_j * (n_j + \sum_{k=1}^{l} a_k) \geq \ldots \geq \sum_{i=1}^{s} \mu_i * (n_i + a_i) \quad (3)$$

სადაც უტოლობის მარცხენა მხარე ეთანადება *(i-1)*ნფ-ს, ხოლო მარჯვენა მხარე - *(i+1)*ნფ-ს, ცენტრალური კი – *i* ნფ-ს.

გავაანალიზოთ დეტალურად ორი მოსაზღვრე ნფ, მაგალითად, i და i+1. (3)-დან შეიძლება მივიღოთ:

$$\sum_{j=1}^{s} \mu_j n_j + \sum_{j=1}^{s} \sum_{k=1}^{l} \mu_j a_k \geq \sum_{i=1}^{l} \mu_i n_i + \sum_{i=1}^{l} \mu_i a_i \quad (4)$$

ამის საფუძველზე მართებულია შემდეგი გამოსახულებები:

$$\sum_{i=1}^{l}\mu_i n_i - \sum_{j=1}^{s}\mu_j n_j = \sum_{i=s+1}^{l}\mu_i n_i \tag{5}$$

$$\sum_{j=1}^{s}\sum_{k=1}^{l}\mu_j a_k - \sum_{i=1}^{l}\mu_i a_i = \sum_{j=1}^{s}\sum_{k=1,\, j\neq k}^{l}\mu_j a_k - \sum_{i=s+1}^{l}\mu_i a_i \tag{6}$$

თუ (5) და (6) გამოსახულებატა მარჯვენა ნაწილებს ჩავსვამთ (4)-ში, მივიღებთ:

$$\sum_{j=1}^{s}\sum_{k=1,\, j\neq k}^{l}\mu_j a_k \geq \sum_{i=s+1}^{l}\mu_i n_i + \sum_{i=s+1}^{l}\mu_i a_i \tag{7}$$

უტოლობის ორივე მხარე გავყოთ $\sum_{i=s+1}^{l}\mu_i a_i$ -ზე:

$$\frac{\sum_{j=1,k=1}^{s}\sum_{j\neq k}^{l}\mu_j a_k}{\sum_{i=s+1}^{l}\mu_i a_i} \geq \frac{\sum_{i=s+1}^{l}\mu_i n_i}{\sum_{i=s+1}^{l}\mu_i a_i} + \frac{\sum_{i=s+1}^{l}\mu_i a_i}{\sum_{i=s+1}^{l}\mu_i a_i} \tag{8}$$

ვინაიდან $[1:l]=[1:s]\cup[s+1:l]$ ამიტომ:

$$\frac{\sum_{j=1}^{s}\sum_{k=1,\, j\neq k}^{l}\mu_j a_k}{\sum_{i=s+1}^{l}\mu_i a_i} = \frac{\sum_{j=1}^{s}\sum_{k=1,\, j\neq k}^{s}\mu_j a_k}{\sum_{i=s+1}^{l}\mu_i a_i} + \frac{\sum_{j=1}^{s}\mu_j \sum_{k=s+1}^{l} a_k}{\sum_{i=s+1}^{l}\mu_i \sum_{i=s+1}^{l} a_i}$$

საბოლოოდ მივიღებთ:

$$\frac{\sum_{j=1}^{s}\sum_{k=1,\, j\neq k}^{l}\mu_j a_k}{\sum_{i=s+1}^{l}\mu_i a_i} + \frac{\sum_{j=1}^{s}\mu_j}{\sum_{i=s+1}^{l}\mu_i} \geq \sum_{i=s+1}^{l}\frac{n_i}{a_i} + 1 \tag{9}$$

სადაც $l \geq 2, s \geq 1$ და $l > s$.

პრაქტიკაში ხშირად გამოიყენება შემთხვევა, როცა *l*=2 და *s*=1, მაშინ (9)-ს ექნება ასეთი სახე:

$$\frac{\mu_1}{\mu_2} \geq \frac{n_2}{a_2} + 1$$

ესაა ვონგ-ვედეკინდის მოდელი (1,2 და 3 ნფ განხილვისას) [11]. იგი არის (9) გამოსახულების (ვედეკინდ-სურგულაძის მოდელის) კერძო შემთხვევა, როცა გვაქვს *n*ნფ და n>=3 [11,12].

გამოკვლეულია აგრეთვე შემთხვევა, როდესაც კორტეჟის არაგასაღებური ატრიბუტების მნიშვნელობათა ცვლილების სიხშირე მაღალია, ხოლო გასაღებურისა - დაბალი [11]. განვიხილოთ ეს შემთხვევა, როდესაც j=2, s=1 და მოცემულია რელაციათა სქემები:

$$\begin{cases} R_1(k_1, k_2, \ldots k_{n1}, A_1, A_2, \ldots, A_{a1}) \\ R_2(k_1, k_2, \ldots k_{n2}, B_1, B_2, \ldots, B_{a2}) \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ R_l(k_1, k_2, \ldots k_{nl}, Z_1, Z_2, \ldots, Z_{al}) \end{cases}$$

დავუშვათ, რომ მართებულია პირობები:

$$k_1, \ldots k_n \supseteq k_1, \ldots k_{n2} \quad \text{და} \quad \mu_1 > \mu_2 \qquad (10)$$

(9)-დან გამომდინარე, მოცემული R_1, R_1 და R_{12} სქემებისათვის გასაღებურ ატრიბუტთა მნიშვნელობების ცვლილების მაღალი სიხშირის დროს მიზანშეწონილია R_1 და R_2 დამოკიდებულებათა კომპოზიცია R_{12}-ში, თუ სრულდება პირობა:

$$\mu_1(n_1 + a_1) + \mu_2(n_2 + a_2) > \mu_1(n_1 + a_1 + a_2)$$

აქედან გამომდინარეობს, რომ:

$$\frac{n_2}{a_2} > \frac{\mu_1}{\mu_2} - 1.$$

თუ განიხილება არაგასაღებური ატრიბუტების მნიშვნელობათა ცვლილება გასაღებური ატრიბუტების ცვლილების გარეშე, მაშინ მართებულია შემდეგი გამოსახულება:

$$\mu_1 a_1 + \mu_2 a_2 > \mu_1(a_1 + a_2)$$

აქედან გამომდინარეობს, რომ .

$$\mu_1 > \mu_2$$

რაც ეწინააღმდეგება (10)-ს.

ამგვარად, დასკვნითი ნაწილი შეიძლება ასე ჩამოვაყალიბოთ:

- მონაცემთა ბაზის დამოკიდებულებანი უნდა წარმოდგენილი იქნეს სხვადასხვა რიგის ნორმალური ფორმებით (3ნფ-:-ბნფ), განახლების სიხშირესა და კავშირების ტიპებზე დამოკიდებულებით მოცემულ კონტექსტში;

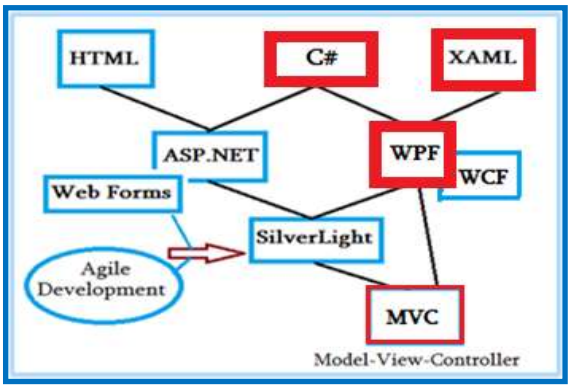
- მოცემული μ-თვის შეიძლება (2.9) გამოსახულებით დადგინდეს მოსახერხებელი (ოპტიმალური) ნფ-ები;

- თუ დამოკიდებულებათა გასაღებური ატრიბუტების ცვლილებების სიხშირე მაღალია, მაშინ მათთვის სასურველია დაბალი რიგის ნფ-ების გამოყენება (დენორმალიზაცია), ხოლო თუ არაგასაღებურ ატრიბუტთა ცვლილების სიხშირე დომინირებადია, მაშინ - შედარებით მაღალი რიგის ნფ-ებისა (ნორმალიზაცია).

III თავი

მომხმარებელთა ინტერფეისის აგება WPF ტექნოლოგიით

მაიკროსოფთმა გამოყენებითი პროგრამული სისტემების პროდუქციის საწარმოებლად, რომელიც მომხმარებლებზე (კლიენტებზე) არის ორიენტირებული ხარისხისა და დიზაინის თვალსაზრისით, შექმნა დეველოპმენტის და ტესტირების არაერთი ინსტრუმენტი [14,15], ასევე ახალი ჰიბრიდული ტექნოლოგია WPF (Windows Presentation Foundation)-ის სახით [1] (ნახ.3.0). ამჯერად განვიხილავთ ინტერფეისის აგების რამდენიმე ამოცანას C# და XAML ენების გამოყენებით.

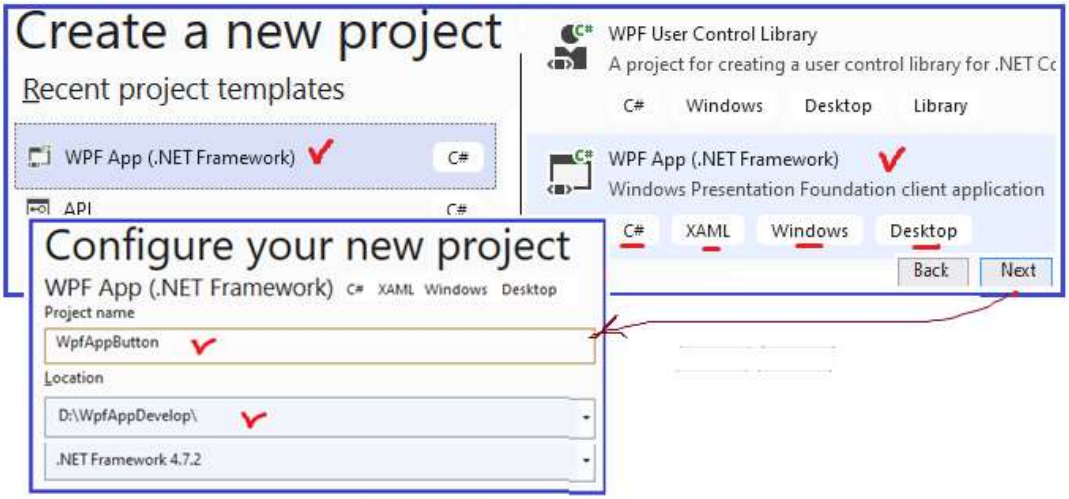


ნახ. 3.0.

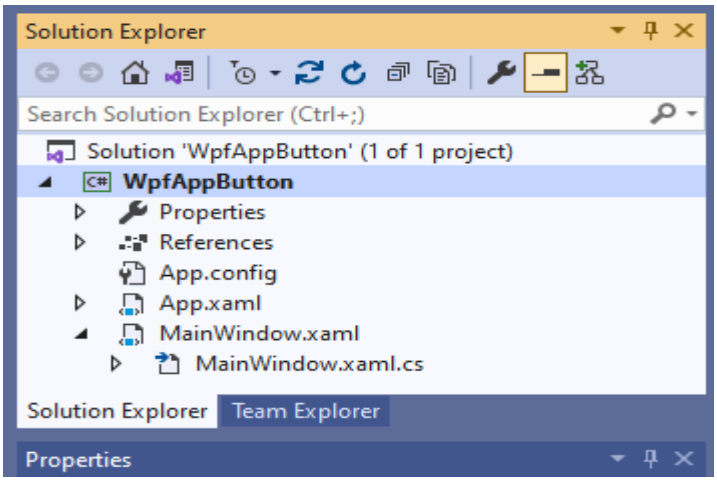
3.1. ანიმაციური ღილაკის დაპროგრამება

წინამდებარე პარაგრაფის მიზანია WPF ტექნოლოგიის მულტიმედიალური შესაძლებლობების დემონსტრირება ანიმაციური თვისებების მქონე ღილაკის (Button) დაპროგრამების საილუსტრაციო მაგალითზე. განვიხილოთ კონკრეტული პროგრამული პროდუქტის აგების მაგალითი დეტალურად.

1) შევქმნათ ახალი პროგრამის პროექტი WpfAppButton სახელით Solution Explorer-ში (ნახ.3.1 და 3.2)



ნახ.3.1. ახალი პროექტის შექმნა



ნახ.3.2. ახალი პროექტის სასტარტო ფაილები

2) წავშალოთ ავტომატურად შექმნილი MainWindow.xaml ფაილი Solution-Explorer-ში და ჩავამატოთ მის ნაცვლად ახალი ფაილი AnimButton.xaml სახელით;

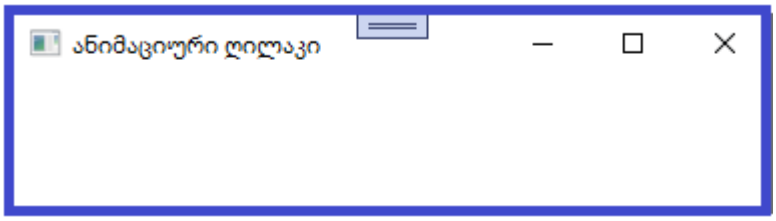
3) გავხსნათ App.xaml ფაილი და შევცვალოთ მასში ატრიბუტი

StartupUri = "AnimButton.xaml"

4) განვსაზღვროთ ფანჯრის სათაური და ზომები (სიმაღლე და სიგანე)

```
<Window x:Class="WpfAppButton.AnimButton"
      xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
      xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
      xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
      xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
      xmlns:local="clr-namespace:WpfAppButton"
      mc:Ignorable="d"
      Title="ანიმაციური ღილაკი" Height="350" Width="500" >
   <Grid>

   </Grid>
</Window>
```

5) ავამუშავოთ პროგრამა და შევამოწმოთ საწყის ეტაპზე შეცდომების არარსებობა (ნახ.3.3).



ნახ.3.3. შეცდომები არაა, ფანჯარა ცარიელია

6) შევავსოთ XAML კოდი საჭირო სტრიქონებით. აქ მწვანე ფერით და სიმბოლიკით „ <!-- ტექსტი ---> “ მოცემულია თითოეული სტრიქონის კომენტარი (მკითხველის დასახმარებლად). დაპროგრამების პროცესში მათი შეტანა კომპიუტერში არაა აუცილებელი. უმჯობესია xaml-ტექსტი გამოვიყენოთ პროტოტიპის სახით და გავარჩიოთ კომენტარები. პროგრამის ამუშავების შემდეგ შესაძლებელია მისი ცვლილება საკუთარი მიზნებისთვის.

```
<!-- მთლიანად ფანჯრის აწყობა: მომხმარებლის ინტერფეისი -->
<Window x:Class="WpfAppButton.AnimButton"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    xmlns:local="clr-namespace:WpfAppButton"
    mc:Ignorable="d"
    Title="ანიმაციური ღილაკი" Height="350" Width="500" >
        x:Name="Window"
        Background="Aqua">

    <!-- აპლიკაციის ფანჯრის რესურსების შესანახი სექცია: -->
    <Window.Resources>
      <!-- იქმნება შაბლონი დასახელებით ClassButton, ღილაკის ტიპის
ელემენტებისთვის-->
      <ControlTemplate x:Key="ClassButton" TargetType="{x:Type Button}">
        <!-- ფანჯრის რესურსების შესანახი სექცია ღილაკისთვის -->
         <ControlTemplate.Resources>
           <!-- სექციაში Storyboard აღიწერება ანიმაციური ეფექტი, მაგ., ღილაკის
დაჭერა-->
           <Storyboard x:Key="Timeline1">
    <!-- მითითებულ დროში, მაგ., 0.3 წამი, ღილაკი ხდება გაუმჭვირვალე -->
             <DoubleAnimationUsingKeyFrames BeginTime="00:00:00"
       Storyboard.TargetName="glow"
Storyboard.TargetProperty="(UIElement.Opacity)">
```

```
                <!-- ანიმაციის ბოლო წერტილი -->
                    <SplineDoubleKeyFrame KeyTime="00:00:0.3" Value="1" />
                </DoubleAnimationUsingKeyFrames>
            </Storyboard>

                <!-- სექცია Storyboard ღილაკის ჩასაქრობად -->
            <Storyboard x:Key="Timeline2">
          <!-- მითითებულ დროში, მაგ., 0.3 წამში ღილაკი ხდება გამჭვირვალე -->
                <DoubleAnimationUsingKeyFrames BeginTime="00:00:00"
        Storyboard.TargetName="glow"
Storyboard.TargetProperty="(UIElement.Opacity)">
            <!-- ანიაციის ბოლო წერტილი -->
             <SplineDoubleKeyFrame KeyTime="00:00:0.3" Value="0" />
                </DoubleAnimationUsingKeyFrames>
            </Storyboard>
    </ControlTemplate.Resources>
      <!-- ღილაკის აღწერის სექცია -->
        <!-- გარე საზღვარი - თეთრი -->
    <Border BorderBrush="#FFFFFFFF" BorderThickness="1,1,1,1"
CornerRadius="4,4,4,4">
        <!-- შიგა საზღვარი - შავი -->
    <Border x:Name="border" Background="#7F000000"
BorderBrush="#FF000000"
                                    BorderThickness="1,1,1,1"
CornerRadius="4,4,4,4">
     <Grid>
       <Grid.RowDefinitions>
         <!-- ღილაკის ზედა ნახევარი -->
          <RowDefinition Height="0.5*" />
            <!-- ღილაკის ქვედა ნახევარი -->
              <RowDefinition Height="0.5*" />
       </Grid.RowDefinitions>

       <!-- იხატება ღილაკის განათება -->
      <Border Opacity="0" HorizontalAlignment="Stretch" x:Name="glow"
Width="Auto"
              Grid.RowSpan="2" CornerRadius="4,4,4,4">
          <Border.Background>
            <!-- მიეწოდება რადიალური გრადიენტი წანაცვლებით -->
             <RadialGradientBrush>
               <RadialGradientBrush.RelativeTransform>
                  <TransformGroup>
```

```
                    <ScaleTransform ScaleX="1.702" ScaleY="2.243" />
                    <SkewTransform AngleX="0" AngleY="0" />
                    <RotateTransform Angle="0" />
                    <TranslateTransform X="-0.368" Y="-0.152" />
                  </TransformGroup>
                </RadialGradientBrush.RelativeTransform>
                <!-- გრადიენტის ფერები ARGB ფორმატში-->
                  <GradientStop Color="#B28DBDFF" Offset="0" />
                  <GradientStop Color="#008DBDFF" Offset="1" />
                </RadialGradientBrush>
              </Border.Background>
            </Border>

            <!-- ათინათის დახატვა -->
            <ContentPresenter HorizontalAlignment="Center"
VerticalAlignment="Center"  Width="Auto" Grid.RowSpan="2" />
          <Border HorizontalAlignment="Stretch" x:Name="shine" Width="Auto"
                    CornerRadius="4,4,0,0">
          <Border.Background>
            <LinearGradientBrush StartPoint="0.494,0.028" EndPoint
="0.494,0.889">
                  <GradientStop Color="#99FFFFFF" Offset="0" />
                  <GradientStop Color="#33FFFFFF" Offset="1" />
                </LinearGradientBrush>
              </Border.Background>
            </Border>
          </Grid>
        </Border>
      </Border>

      <!-- ტრიგერული მოვლენების დაყენება, მაუსის დაჭერის რეაქციაზე  -->
        <ControlTemplate.Triggers>
          <!-- მაუსის ღილაკი დაჭერილია -->
          <Trigger Property="IsPressed" Value="True">
            <Setter Property="Opacity" TargetName="shine" Value="0.4" />
            <Setter Property="Background" TargetName="border" Value =
"#CC000000" />
          <Setter Property="Visibility" TargetName="glow" Value="Hidden" />
          </Trigger>

      <!-- მაუსის კურსორი ობიექტზეა  -->
          <Trigger Property="IsMouseOver" Value="True">
```

```
            <!-- ობიექტზე შესვლა - ანიმაციის გამოძახება Timeline1 -->
            <Trigger.EnterActions>
              <BeginStoryboard Storyboard="{StaticResource Timeline1}" />
            </Trigger.EnterActions>
             <!-- ობიექტიდან გამოსვლა - ანიმაციის გამოძახება Timeline2 -->
            <Trigger.ExitActions>
              <BeginStoryboard Storyboard="{StaticResource Timeline2}" />
            </Trigger.ExitActions>
          </Trigger>
            </ControlTemplate.Triggers>
        </ControlTemplate>
    </Window.Resources>
    <Grid>
        <!-- იქმნება ღილაკის ეგზემპლარი ფანჯრის შუაში -->
    <Button x:Name="Btn1" HorizontalAlignment="Center" VerticalAlignment
="Center"  Width="176" Height="34" Content="მულტიმედიალური ღილაკი"
Foreground="#FFFFFFFF" Template="{DynamicResource ClassButton}" />
    </Grid>
</Window>
```

7) ავამუშავოთ პროგრამა, მიიღება ფანჯარა შაბიამნისფერი ფონით და ანიმაციური ღილაკით ცენტრში, წარწერით (ნახ.3.4)

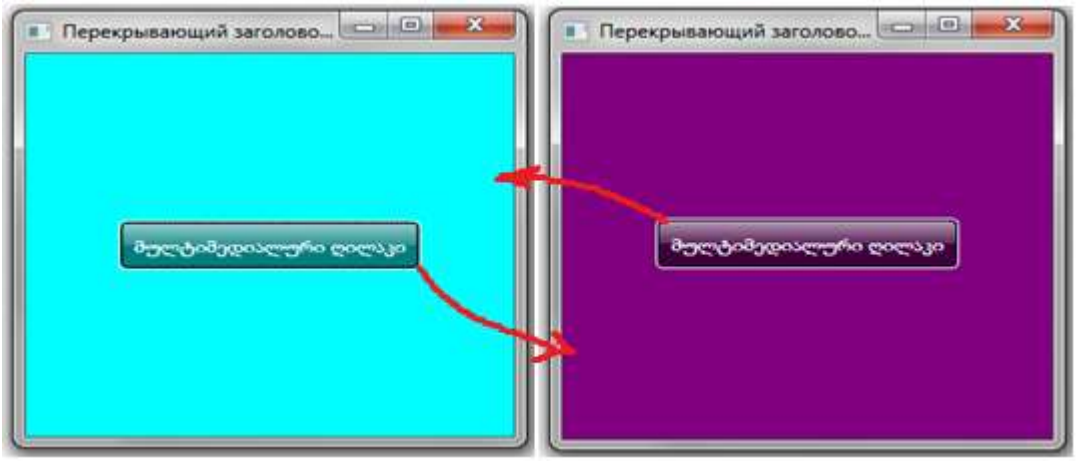
ნახ.3.4

8) დავაპროგრამოთ ლოგიკა C#-ენაზე და შევამოწმოთ ღილაკის მოქმედება.

```
using System;
using System.Collections.Generic;
using System.Text;
```

```
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
namespace WpfAppButton
{
    /// <summary>
    /// Interaction logic for MainWindow.xaml
    /// </summary>
    public partial class AnimButton : Window
    {
        System.Windows.Media.Brush color;
        bool colorFlag = true;

       public AnimButton()
        {
            //InitializeComponent();
             Application.LoadComponent(this, new
                         Uri("AnimButton.xaml.xaml", UriKind.Relative));

              Btn1.Click += new RoutedEventHandler(Btn1_Click);
              color = this.Window.Background;
        }

         void Btn1_Click(object sender, RoutedEventArgs e)
         {
            // ფინის ფერის შეცვლა
          if (colorFlag)
             this.Window.Background = System.Windows.Media.Brushes.Purple;
            else
              this.Window.Background = color;

              colorFlag = !colorFlag;
        }
    }
}
```

ავამუშავოთ აპლიკაცია, შედეგი მოცემულია 3.4 ნახაზზე



ნახ.3.4. შედეგი

9) განმარტებისათვის: Application – ობიექტი

ნებისმიერი დანართი (აპლიკაცია) იყენებს Application კლასს, რომელიც Run() მეთოდის საშუალებით ამ აპლიკაციას მიუერთებს ოპერაციული სისტემის მოვლენების მოდელს – ასამუშავებლად.

Application - ობიექტი მართავს დანართის სიცოცხლის დროს, აკვირდება ხილვად ფანჯრებს, ათავისუფლებს რესურსებს და აკონტროლებს აპლიკაციის გლობალურ მდგომარეობას. Run() მეთოდი აამუშავებს შესრულების გარემოს დისპეტჩერს, რომელიც დაიწყებს მოვლენების და შეტყობინებების გადაგზავნას დანართის კომპონენტებთან.

დროის მოცემულ მომენტში აქტიური შეიძლება იყოს მხოლოდ ერთი Application – ობიექტი, და ის მუშაობს მანამ, სანამ დანართი არ დასრულდება. შესრულებად Application-ობიექტზე მიმართვა შეიძლება დანართის ნებისმიერი ადგილიდან Application.Current სტატიკური თვისების საშუალებით.

WPF დანართის (და Application ობიექტის) სასიცოცხლო დრო შედგება შემდეგი ეტაპებისგან:

1. კონსტრუირდება Application ობიექტი;

2. გამოიძახება მისი Run() მეთოდი;

3. ინიცირდება მოვლენა Application.Startup;

4. მომხმარებლის კოდი აკონსტრუირებს ერთ ან რამდენიმე Window (ან Page) ობიექტს და დანართი მუშაობს;

8. გამოიძახება მეთოდი Application.Shutdown();

6. გამოიძახება მეთოდი Application.Exit().

ჩვენი WPF-დანართის აგებისას სისტემამ თვითონ შექმნა Solution Explorer-ში ორი ფაილი Application-ობიექტთან დაკავშირებული ტიპური სახელებით: App.xaml და App.xaml.cs. მათი ნახვა შესაძლებელია. აქ არ ჩანს ცხადად არც Application და Window ობიექტების შექმნა და არც Run() მეთოდის გამოძახება (!)

WPF-პლატფორმის შემქმნელებმა გადაწყვიტეს, რომ, ვინაიდან ეს სტანდარტული ოპერაციები გამოიყენება ყველა დანართისათვის, არაა საჭირო მისი მომხმარებელზე გადაცემა და თვით კომპილატორი (სისტემა) ავტომატურად გამოიყენებს მათ (!), თვითონ შექმნის Application ობიექტს WPF-პროექტის კომპილაციის დროს, შექმნის ასევე Window (ან Page) ობიექტებს და გადასცემს მათ Application.Run() მეთოდს.

სხვა სიტყვებით რომ ვთქვათ, ჩვენ „ვერ ვხედავთ ცხადად“ კომპილატორის მიერ ასეთი კოდის შესრულების ტექსტს:

```
// ეს ტექსტი მოტანილია საილუსტრაციოდ ---------
public partial class App : System.Windows.Application
{
  public App()
    {
        System.Windows.Window win = new System.Windows.Window();
        win.Title = "Hello World";
        win.Show();
    }
    // Application entry point
    [System.STAThreadAttribute()]
    [System.Diagnostics.DebuggerNonUserCodeAttribute()]
    public static void Main()
    {
        App app = new App();
        app.Run();
    }
}
```

- ჩვენი დანართის მაგალითისათვის App.xaml და App.xaml.cs ფაილები ასე გამოიყურება:

```
<!—App.xaml - ფაილი ------>
<Application x:Class="WpfAppButton.App"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:local="clr-namespace:WpfAppButton"
        StartupUri="AnimButton.xaml">
  <Application.Resources>
```

```
  </Application.Resources>
</Application>
```

```
//--- App.xaml.cs - ფაილი ----
  using System;
  using System.Collections.Generic;
  using System.Configuration;
  using System.Data;
  using System.Linq;
  using System.Threading.Tasks;
  using System.Windows;

  namespace WpfAppButton
  {
    /// <summary>
    /// Interaction logic for App.xaml
    /// </summary>
    public partial class App : Application
    {
    }
  }
```

➢ გადავაკეთოთ App.xaml.cs ფაილი ასე:

```
// ----- App.xaml.cs ------------------------
using System;
using System.Collections.Generic;
using System.Configuration;
using System.Data;
using System.Windows;

namespace WpfAppButton
{
  public partial class App : Application
    {
      public App()
        {
         // გაშვებულია Application ობიექტი
         this.Startup += new StartupEventHandler(App_Startup);
         // მოვლენა, დაუმუშავებელი გამოსარიცხი სიტუაციისთვის
 this.DispatcherUnhandledException +=
                    App_DispatcherUnhandledException;
```

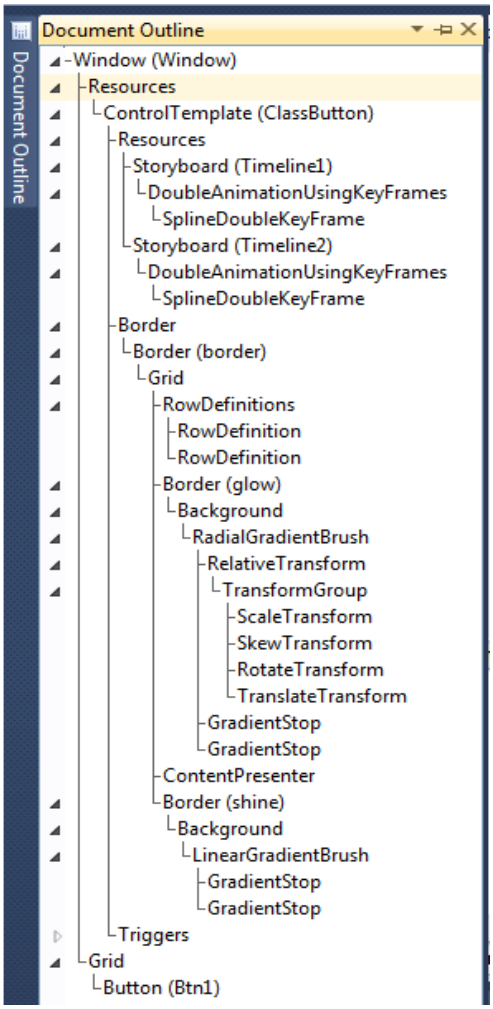
```
        }
    void App_DispatcherUnhandledException(object sender,

System.Windows.Threading.DispatcherUnhandledExceptionEventArgs e)
        {
            e.Handled = true;// შესრულების გაგრძელება
        }

        void App_Startup(object sender, StartupEventArgs e)
        {
            // იქმნება სასტარტო ფანჯრის ობიექტი -----
            AnimButton win = new AnimButton ();
            // ფანჯრის ობიექტის აწყობა -----
            win.Title = "ფანჯრის  ახალი სათაური ";
            // ფანჯრის ნახვა ---------
            win.Show();
        }
    }
}
```

ავამუშავოთ დანართი და ვნახოთ შედეგი.

10) XAML-პროგრამის სტრუქტურის სანახავად Document Outline პანელის გამოტანა (ნახ.3.6).

View->Other Windows->Document Outline

მიიღება შემდეგი ეს იერარქიული სურათი.

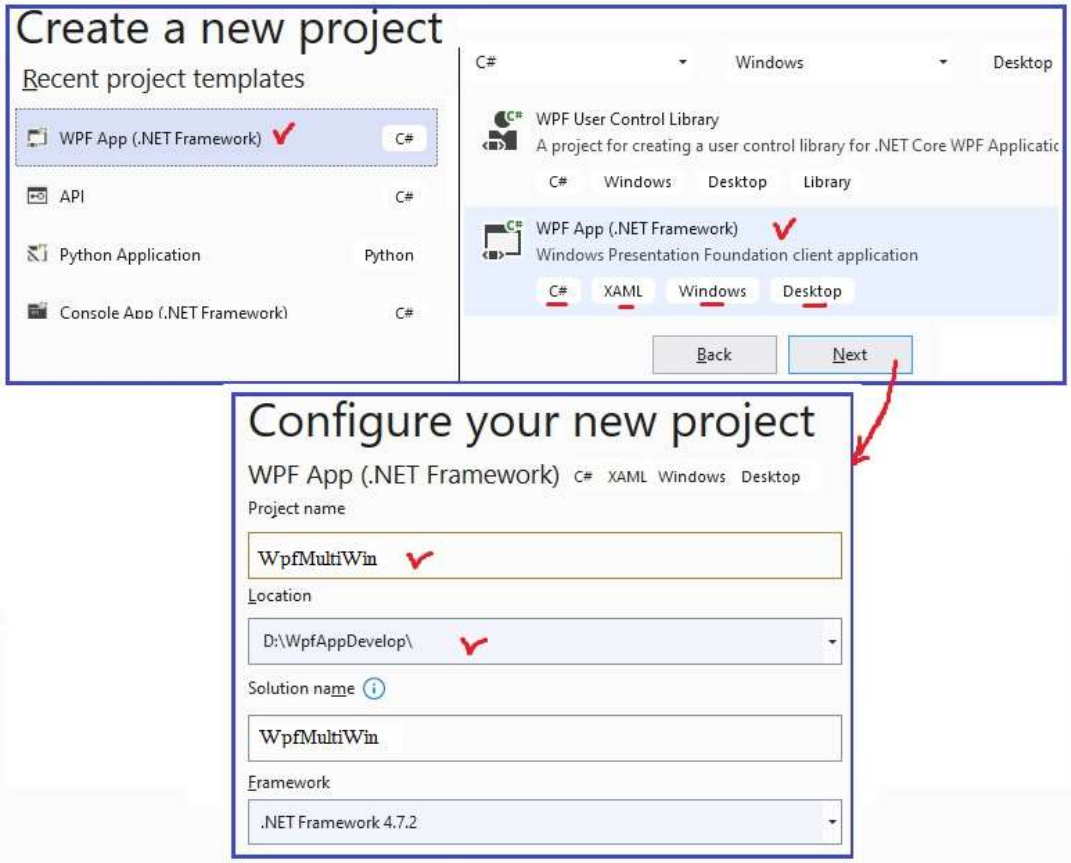


ნახ.3.6

3.2. მრავალფანჯრიანი ინტერფეისის პროგრამული პროექტის აგება

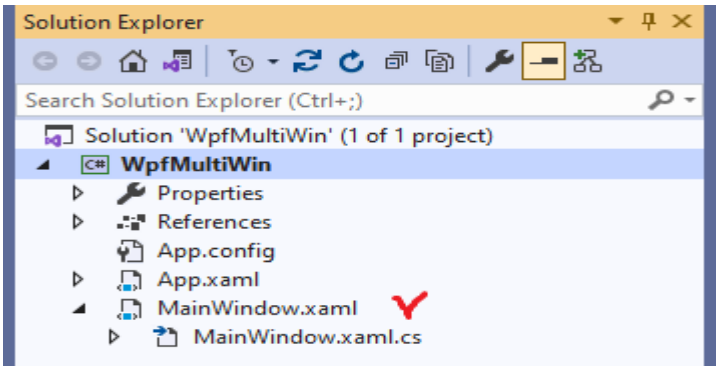
ერთ აპლიკაციაში შესაძლებელია რამდენიმე ფანჯრის შექმნა, რომლებიც ერთდროულად (მიმდევრობით) შეიქმნება და გაიხსნება. მათი დახურვის პროცესი მოითხოვს გარკვეული მართვის განხორციელებას. მაგალითად, თუ მომხმარებელმა დახურა ერთი რომელიმე, სხვა რჩება გახსნილი. შესაძლებელია აგრეთვე ინფორმაციის (შეტყობინების) მიღება არსებული მდგომარეობის შესახებ. საჭიროა ისეთი მოვლენის რეალიზაციაც (პროცესის აგება), როდესაც ყველა ფანჯრის დახურვა იქნება შესაძლებელი ერთდროულად და ა.შ. განვიხილოთ ასეთი პროგრამული პროექტის შექმნა დეტალურად Visual Studio .NET Framework პლატფორმაზე.

1) შევქმნათ ახალი C# პროექტი WpfMultiWin სახელით, Solution Explorer-ში (ნახ.3.7 და 3.8);



ნახ.3.7

შედეგი - Solution Explorer-ის საწყისი მდგომარეობა (ნახ.3.8).



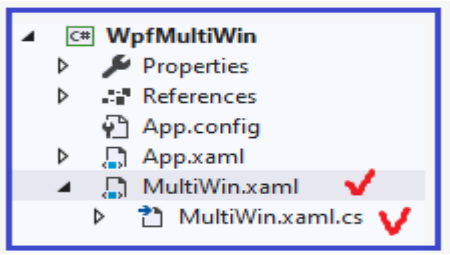
ნახ.3.8

MultiWin.xaml და MultiWin.xaml.cs კოდების საწყისი ლისტინგები:

```
<!--- xml კოდი ---- იქმნება სისტემის მიერ ------- >
<Window x:Class="WpfMultiWin.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:WpfMultiWin"
        mc:Ignorable="d"
        Title="MainWindow" Height="300" Width="400">
    <Grid>

    </Grid>
</Window>

//-- C# კოდი -------- იქმნება სისტემის მიერ ---
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
// -- using-ებიდან შეიძლება წაიშალოს ის , რაც არ გვჭირდება -----
```

```
namespace WpfMultiWin
{
   /// <summary>
   /// Interaction logic for MainWindow.xaml
   /// </summary>
   public partial class MainWindow : Window
   {
      public MainWindow()
      {
         InitializeComponent();
      }
   }
}
```

2) Solution Explorer-ში შევცვალოთ (Rename-ით) MainWindow.xaml ახალი სახელით MultiWin-ით (ეს შეცვლის ავტომატურად cs-ის სახელსაც (ნახ.3.9);



ნახ.3.9

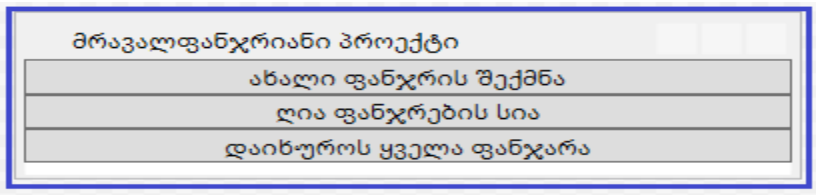
3) გახსენით `App.xaml` ფაილი და შეცვალეთ `Application`-ობიექტის პარამეტრი `StartupUri` ახალი სასტარტო ფანჯრის MultiWin.xaml სახელით (ნახ.3.10).

ნახ.3.10. Solution Explorer-ის App.xml ფაილი

4) შევავსოთ MultiWin.xaml კოდი ჩვენთვის საჭირო ფანჯრის ელემენტებით, კერძოდ ღილაკებით და მათი ფუნქციების წარწერებით:

```
<Window x:Class="WpfMultiWin.MultiWin"
    ...
    Title="მრავალფანჯრიანი პროექტი"
    SizeToContent="WidthAndHeight"  Height="140" Width="350">
  <StackPanel>
        <Button Click="NewWindowClicked">ახალი ფანჯრის შექმნა</Button>
        <Button Click="ListOpenWindows">ღია ფანჯრების სია</Button>
        <Button Click="AllCloseWindows">დაიხუროს ყველა ფანჯარა</Button>
  </StackPanel>
</Window>
```

შედეგი (ნახ.3.11).



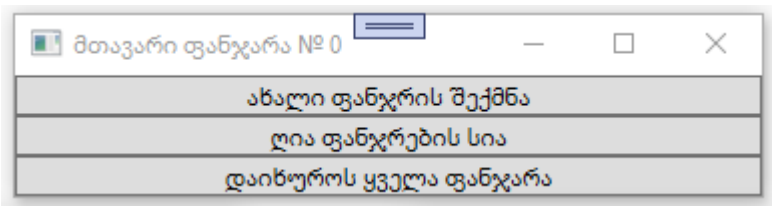
ნახ.3.11

5) შეცვალეთ MultiWin.xaml.cs კოდი, ჩავამატონ სამი ღილაკეს ფუნქციები (Events - მოვლენები, რომლებმაც უნდა გამოიძახოს შესაბამისი მეთოდები).

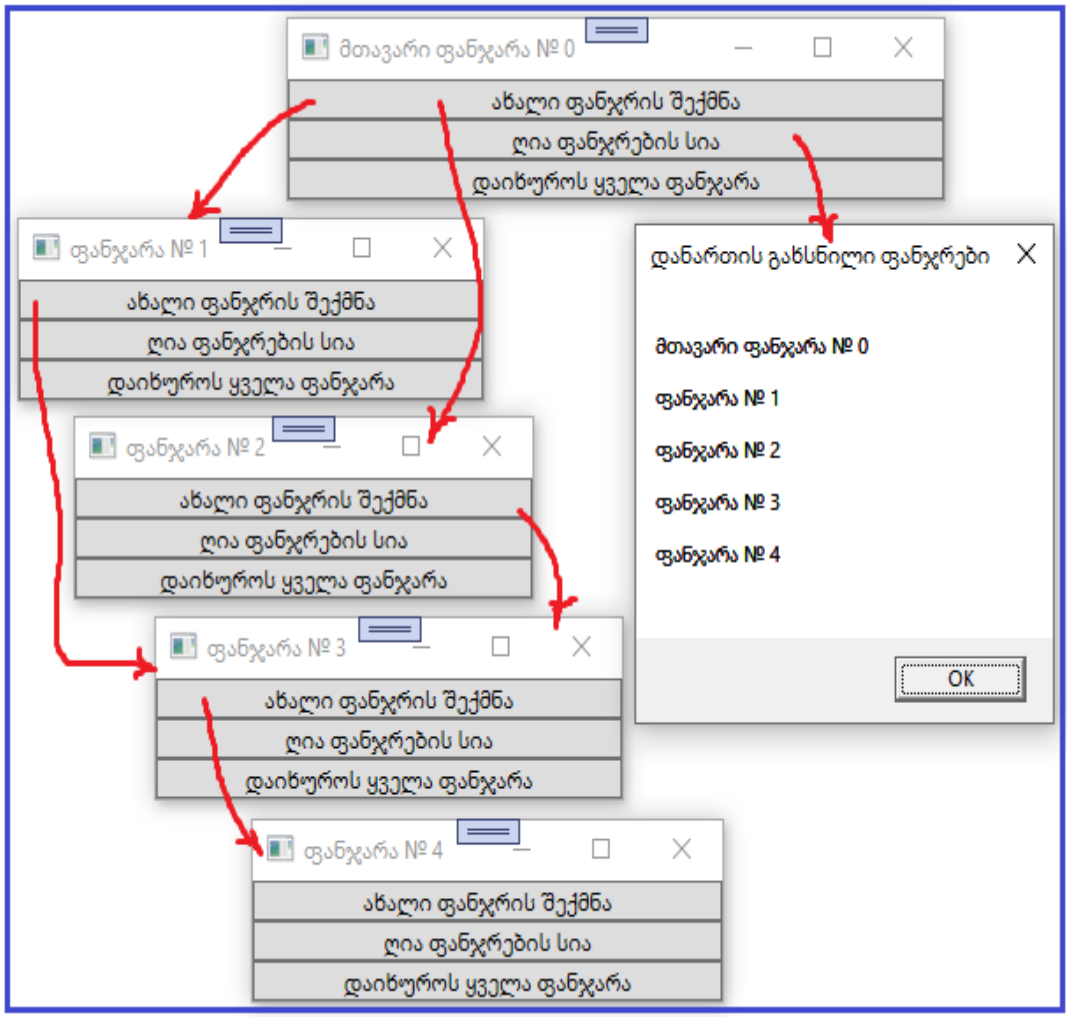
```
using System;
using System.Collections.Generic;
using System.Text;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Shapes;

namespace WpfApp3
{
    public partial class WindowList : Window
    {
        static int createCount;
        public WindowList()
        {
            InitializeComponent();

            // ფანჯრის სახელის განსაზღვრა და მთავარი ფანჯრის არჩევა
            if (createCount == 0)
```

```
            {
                Application.Current.ShutdownMode =
ShutdownMode.OnMainWindowClose;
                Application.Current.MainWindow = this;
                this.Title = "მთავარი ფანჯარა № " +
(createCount++).ToString();
            }
            else
                this.Title = "ფანჯარა № " + (createCount++).ToString();
        }

        private void NewWindowClicked(object sender, RoutedEventArgs e)
        {
            new WindowList().Show();
        }

        private void ListOpenWindows(object sender, RoutedEventArgs e)
        {
            StringBuilder sb = new StringBuilder();
            foreach (Window openWindow in Application.Current.Windows)
                sb.AppendLine(openWindow.Title);

            MessageBox.Show(sb.ToString(), "დანართის გახსნილი ფანჯრები");
        }

        private void AllCloseWindows(object sender, RoutedEventArgs e)
        {
            //Application.Current.ShutdownMode =
ShutdownMode.OnExplicitShutdown;
            Application.Current.Shutdown();// ხურავს ფანჯრებს
        }
    }
}
```

6) ავამუშავოთ პროგრამა. შედეგები ასახულია 3.11-3.14 ნახაზებზე.



ნახ.3.11. მთავარი ფანჯარა N0 (შეიძლება სხვა გავხადოთ მთავარი)



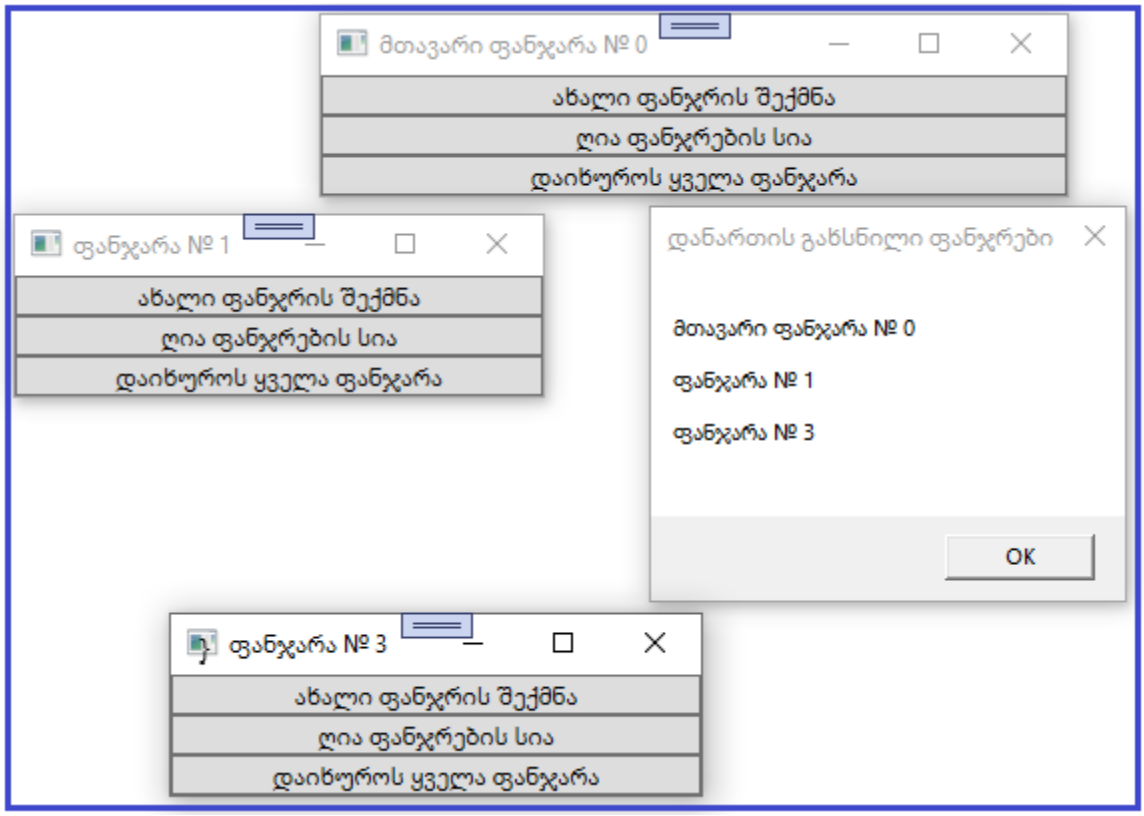
ნახ.3.12. ახალი ფანჯრების შექმნა და ღია ფანჯრების სიის გამოტანა

„ახალი ფანჯრის შექმნა“ ღილაკით (მიმდევრობით) ჩნდება ეკრანზე ნომერმინიჭებული ფანჯრები. „ღია ფანჯრების სიის“ ღილაკით.

მთავარი ფანჯრის სტატუსი და წარწერა განისაზღვრება C# კოდში:

```
// ფანჯრის სახელის განსაზღვრა და მთავარი ფანჯრის არჩევა
   if (createCount == 0) // 0-ის მაგივრად ჩავწერთ სასურველ ნომერს ------
     { . . . }
```

მაგალითად, დავხუროთ მე-2 და მე-4 ფანჯრები. შედეგი (ნახ.3.13).



ნახ.3.13. შეიცვალა ღია ფანჯრების სია

7) ღილაკით „დაიხუროს ყველა ფანჯარა“ - იხურება ფანჯრებიც და პროგრამა ამთავარებს მუშაობას.

შეიზლება ითქვას, რომ ერთფანჯრიანი ან მრავალფანჯრიანი ინტერფეისების შექმნა განისაზღვრება ასაგები მიზნობრივი პროგრამული სისტემის ფუნქციური ამოცანების საფუძველზე.

საჭიროა სისტემური ანალიზის საფუძველზე დადგინდეს სისტემისა და სავარაუდო ფუნქციონალური ქვესისტემების არქიტექტურა, მათში შემავალი ფუნქციური ამოცანების ნუსხა და ა.შ. შესაბამისად, დასაპროექტებელი ფანჯრების შიგთავსიც (მაგალითად, ღილაკების წარწერები) წარმოდგენილ უნდა იქნას აღნიშნული ფუნქციური ამოცანების სათაურებით.

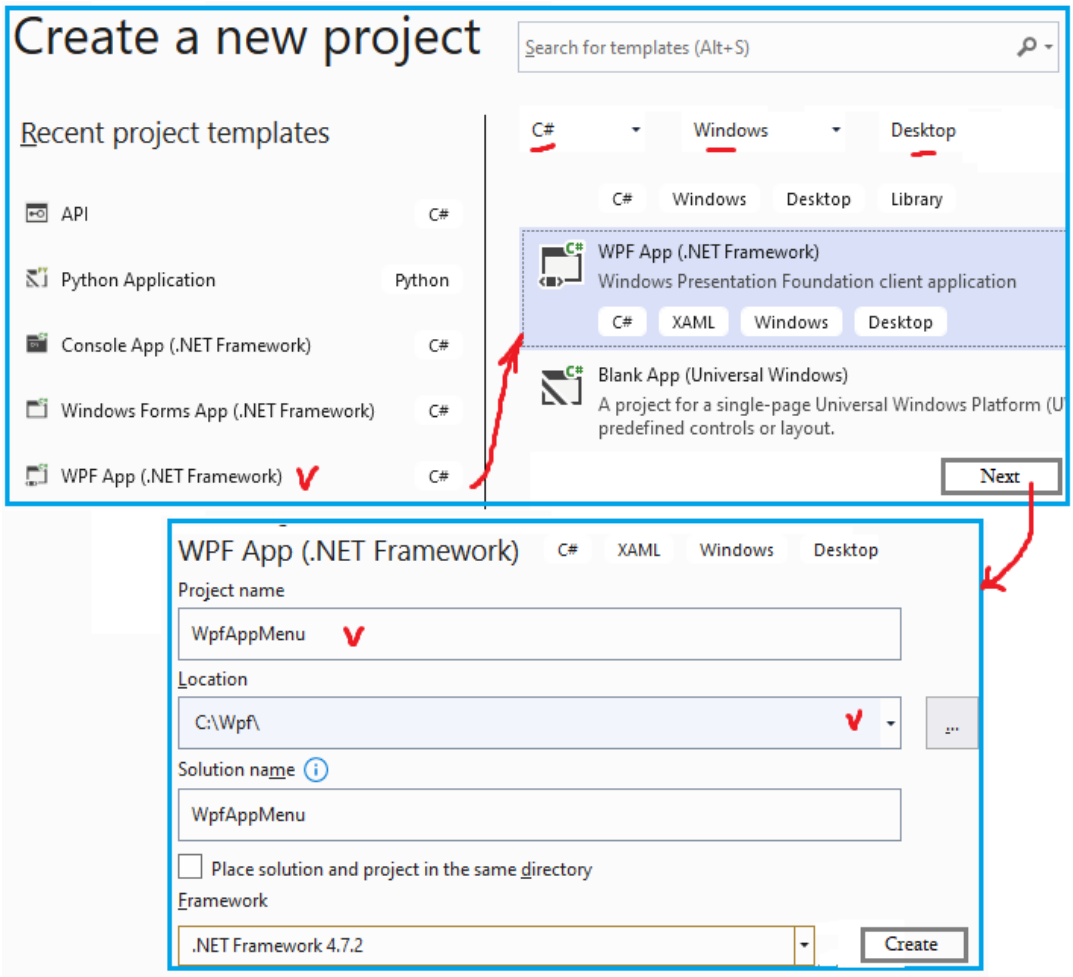
ინტერფეისებში დიალოგური პროცედურების პროგრამული რეალიზაცია სისტემის დაპროექტების პირველივე ეტაპებზე შეთანხმებული ან დაზუსტებული უნდა იყოს სავარაუდო მომხმარებელთან (დამკვეთთან), მათი სურვილების, დიზაინის და სხვა პარამეტრების გათვალისწინებით.

მომდევნო პარაგრაფში გავაგრძელებთ მომხმარებლის ინტერფეისების სხვა ხერხების და მეთოდების გაცნობას.

3.3. მომხმარებლის ინტერფეისის ელემენტები WPF-ში: Menu, ToolBar და TabControl

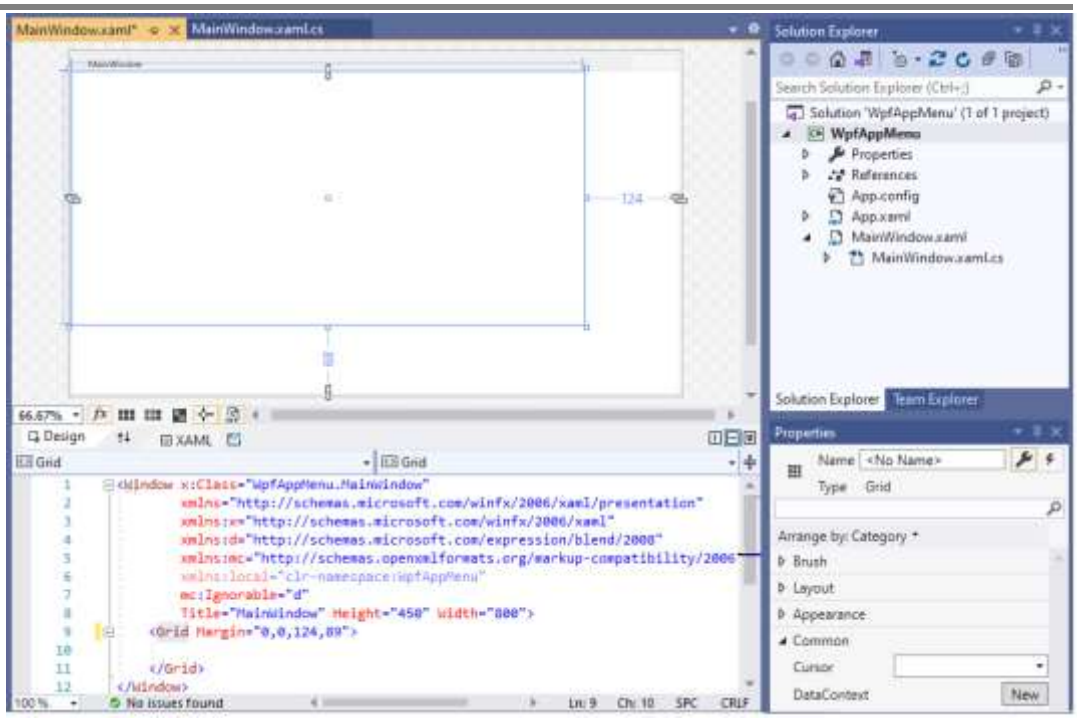
განვიხილოთ WPF-აპლიკაციების ასაგებად Menu, ToolBar, TabControl და ToolTip მართვის ელემენტების გამოყენება. მომხმარებელთა ინტერფეისების ასაგებად WPF-ის ვიზუალური მართვის ელემენტების უფრო დეტალური გაცნობა შესაძლებელია მაიკროსოფთის ბიბლიოთეკის ელექტრონული ცნობარიდან: http://msdn.microsoft.com/en-en/library/ms752324.aspx

შევქმნათ ახალი პროექტი Visual Studio-ს Solution Explorer-ში WpfAppMenu სახელით (ნახ.3.14) :



ნახ.3.14. ახალი პროექტის WpfAppMenu შექმნა

მიიღება პროექტის საწყისი მდგომარეობა (ნახ.3.15)



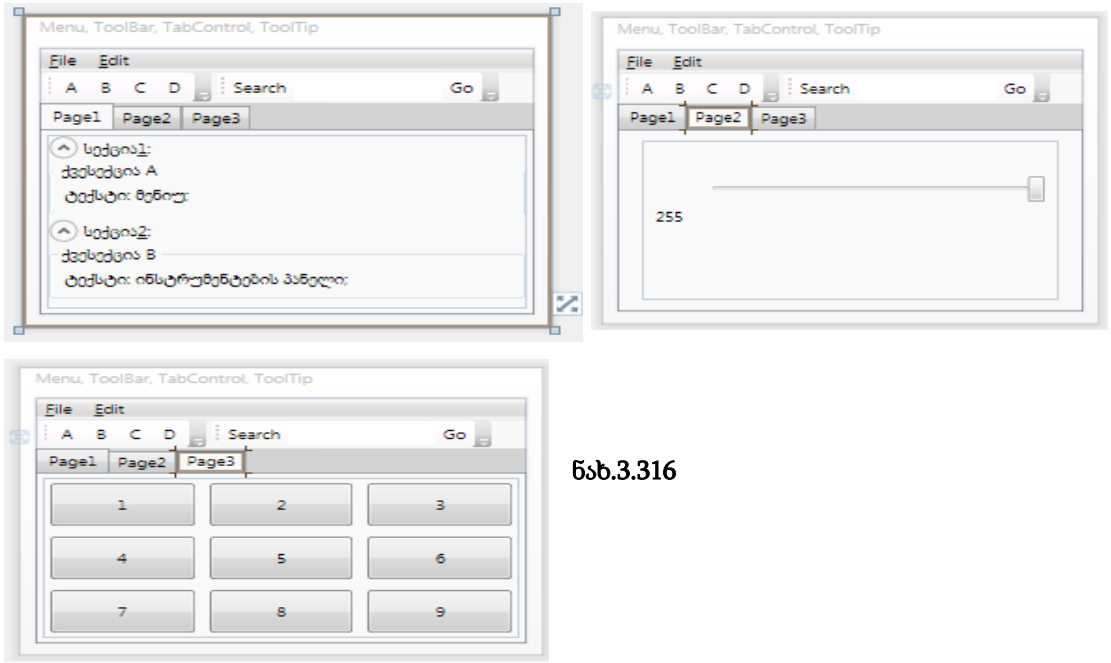
ნახ.3.18.

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

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

     <Window.ToolTip>
         <ToolTip x:Name="toolTip"
             Placement="RelativePoint"
             VerticalOffset="10"
             />
    </Window.ToolTip>
     <DockPanel Background="LightGray">
         <Menu DockPanel.Dock="Top">
```

```
    <MenuItem Header="_File">
    <MenuItem Header="E_xit" Click="ExitClicked" />
    </MenuItem>
    <MenuItem Header="_Edit">
        <MenuItem Header="_Cut" />
        <MenuItem Header="C_opy" />
        <MenuItem Header="_Paste" />
    </MenuItem>
</Menu>

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

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

        </StackPanel>
    </TabItem>
    <TabItem Header="Page2">
       <StackPanel Orientation="Horizontal"
                        Margin="5" Width="297">
```

```
            <TextBlock Name="value" Margin="10"
                 Text="255" Width="25" Height="23" />
              <Slider Name="slider" Width="241"
              Minimum="0" Maximum="255" Value="255"
      ValueChanged="slider_ValueChanged" Height="77" />
            </StackPanel>
          </TabItem>
          <TabItem Header="Page3">
              <UniformGrid Rows="3" Columns="3">
        <Button ToolTip="ღილაკი 1" Margin="5">1</Button>
        <Button ToolTip="ღილაკი 2"  Margin="5">2</Button>
        <Button ToolTip="ღილაკი 3" Margin="5">3</Button>
        <Button ToolTip="ღილაკი 4" Margin="5">4</Button>
        <Button ToolTip="ღილაკი 5" Margin="5">5</Button>
        <Button ToolTip="ღილაკი 6" Margin="5">6</Button>
        <Button ToolTip="ღილაკი 7" Margin="5">7</Button>
        <Button ToolTip="ღილაკი 8" Margin="5">8</Button>
        <Button ToolTip="ღილაკი 9" Margin="5">9</Button>
         </UniformGrid>
       </TabItem>
     </TabControl>
   </DockPanel>
</Window> // შედეგში მიიღება სამი გვერდი Page1, Page2 და Page3 (ნახ.3.16).
```

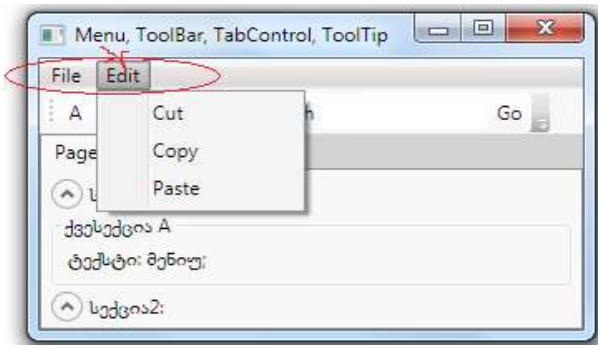


ნახ.3.316

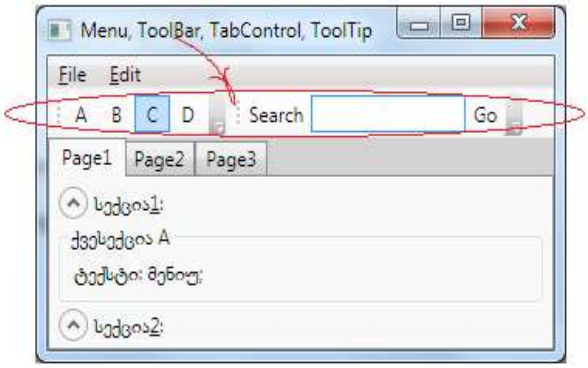
ახლა MainWindow.xaml.cs კოდისათვის შევიტანოთ ტექსტი (იხ. ლისტინგი).

```
//-- ლისტინგი_ MainWindow.xaml.cs ----------------------------
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;

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

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

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

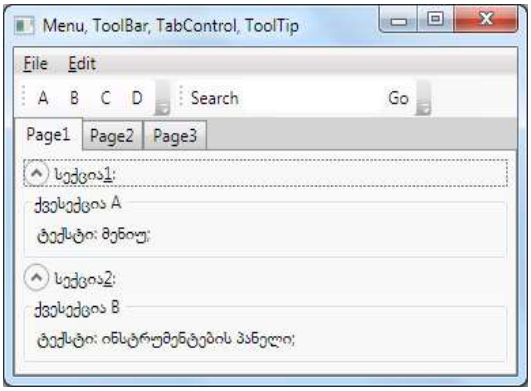
შედეგები ასახულია 3.17 (ა-ე) ნახაზებზე.



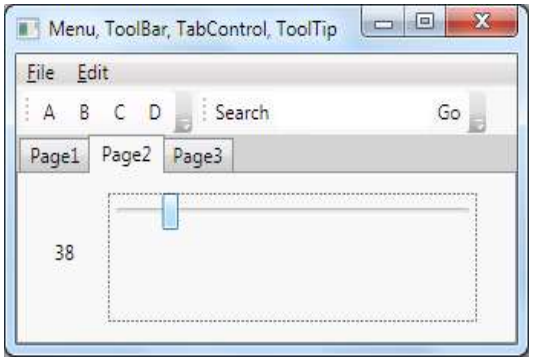
ნახ.3.17–ა



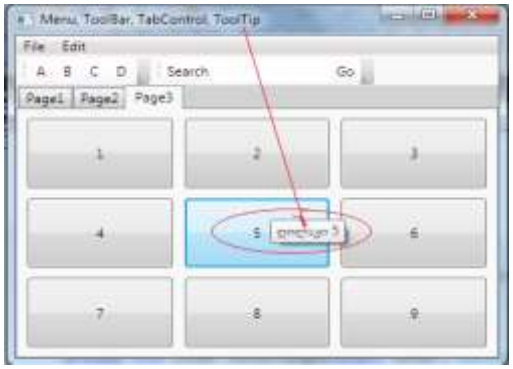
ნახ.3.17–ბ



ნახ.3.17–გ



ნახ.3.17–დ



ნახ.3.17-ე

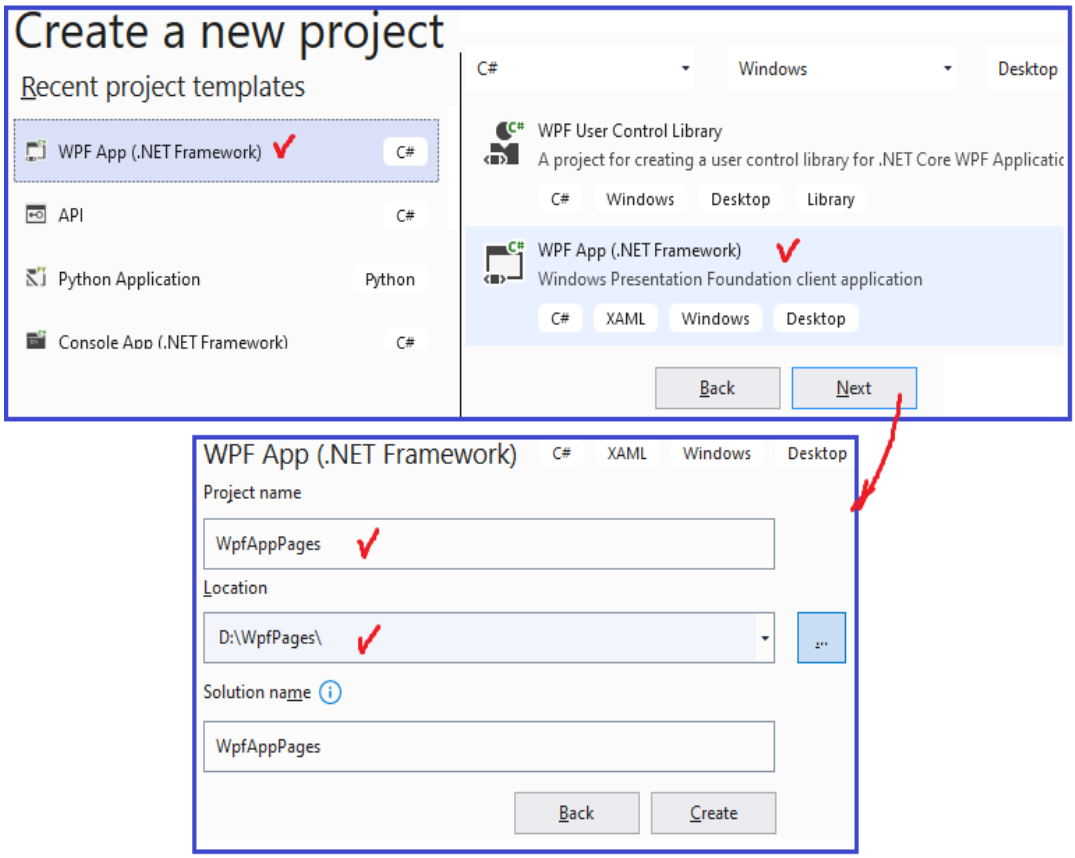
3.4. WPF-ში ვებგვერდების (Pages) აპლიკაციის შექმნა

WPF-ტექნოლოგიას აქვს ვებ-გვერდების დეველოპმენტის საშუალებები. ასეთი გვერდების გამოყენება ხშირად მომხმარებლისთვის უფრო მოსახერხებელია, ვიდრე ფანჯრული (Windows) ორგანიზაცია.

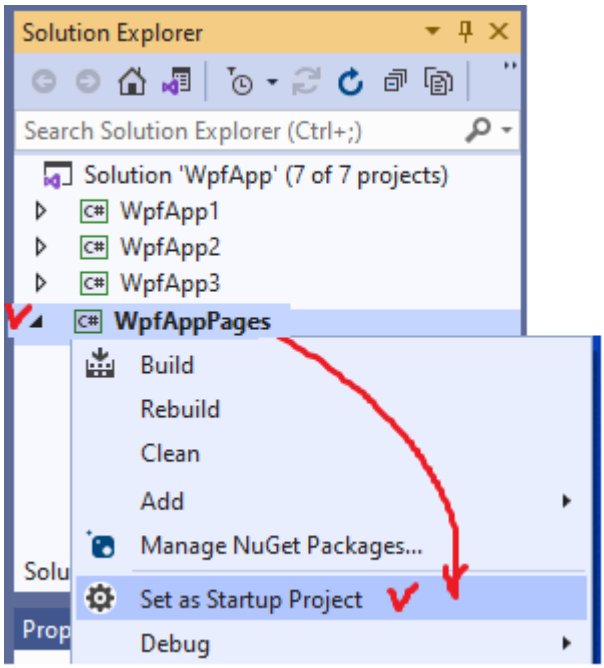
სამუშაოს ძირითადი პრინციპი ისაა, რომ გვერდული დანართის (აპლიკაციის) შიგთავსი ჩაშენდება სპეციალურ ნავიგაციურ კარკასში, რომელსაც აქვს ნავიგაციური კავშირები და ნავიგაციური ჟურნალი [17,18].

მისი ფესვური კლასია NavigationWindow, რომელიც ამატებს აპლიკაციისთვის *ნავიგაციის სტანდარტულ ინტერფეისს* და მისთვის საჭირო ინფრასტრუქტურას. NavigationWindow კლასი წარმოებულია Window-კლასიდან და აქვს წვდომა დანართის იგივე საშუალებებზე.

1) შევქმნათ ახალი პროექტი WpfAppPages3 სახელით Solution Explorer-ში (ნახ.3.18). გავხადოთ იგი „სასტარტო“, თუ აქ სხვა პროექტებიცაა (ნახ.3.19);



ნახ.3.18



ნახ.3.19

2) შევცვალოთ ავტომატურად შექმნილი MainWindow.xaml ფაილის სახელი SolutionExplorer-ში და ჩავწეროთ NavExample.xaml -ით;

3) გავხსნათ App.xaml ფაილი და შევცვალოთ მასში ატრიბუტი

StartupUri = "NavExample.xaml"

4) ავამუშავოთ პროექტი WpfAppPages3, დავრწმუნდეთ, რომ არაა შეცდომები;

5) გავხსნათ ფაილი NavExample.xaml და შევასწოროთ მასში დესკრიპტორული კოდი:

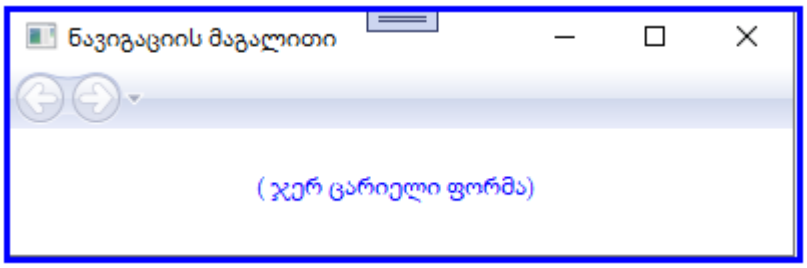
```
<NavigationWindow x:Class="WpfAppPages3.NavExample"
      xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
      xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
         Title="ნავიგაციის მაგალითი"
   Height="300"      Width="400"
   WindowStartupLocation="CenterScreen"
           >
</NavigationWindow>
```

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

```
using System;
using System.Collections.Generic;
using System.Text;
```

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

using System.Windows.Navigation;

namespace WpfAppPages3
{
   public partial class NavExample : NavigationWindow
   {
      public NavExample()
      {
         InitializeComponent();

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

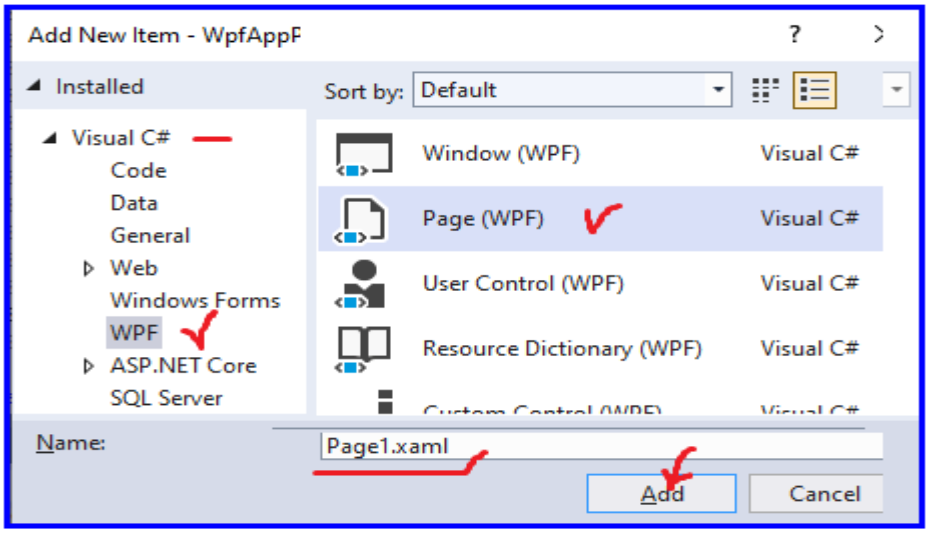
ავამუშავოთ პროექტი, შედეგი (ნახ.3.20).



ნახ.3.20. უშეცდომო საწყისი შედეგი

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

- დავამატოთ პროექტს ახალი Page ელემენტი Page1.xaml სახელით (ნახ.3.21).



ნახ.3.21

8) შევასწოროთ Page1.xaml ფაილის ტექსტი:

```
<Page x:Class="WpfAppPages3.Page1"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:local="clr-namespace:WpfPages"
    mc:Ignorable="d"
    d:DesignHeight="400" d:DesignWidth="600"
    Title="Page1">

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

9) შევასწოროთ Page1.xaml.cs ფაილის კოდი:

```
using System;
using System.Collections.Generic;
using System.Text;
```

```
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;

namespace WpfAppPages
{
   /// <summary>
   /// Interaction logic for Page1.xaml
   /// </summary>
   public partial class Page1 : Page
   {
      public Page1()
      {
         InitializeComponent();
      }
      private void LinkClicked(object sender, RoutedEventArgs e)
      {
         this.NavigationService.Navigate("Page2.xaml");
      }
   }
}
```

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

```
<Page x:Class="WpfAppPages.Page2"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:local="clr-namespace:WpfPages"
    mc:Ignorable="d"
    d:DesignHeight="400" d:DesignWidth="700"
    Title="Page2"
    >
<StackPanel>
```

```
   <TextBlock TextAlignment="Center" FontSize="24">გვერდი 2</TextBlock>
</StackPanel>
</Page>
```

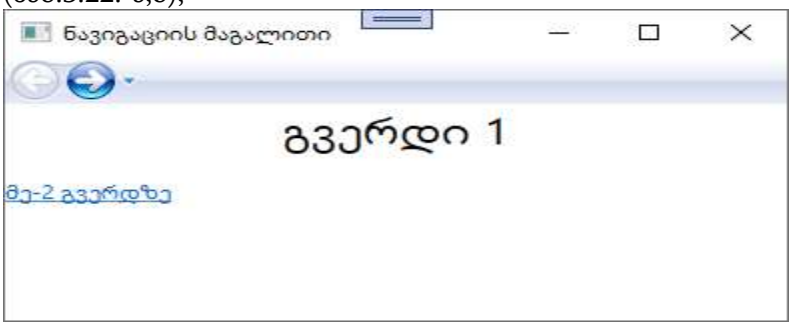
11) Page2.xaml.cs ფაილის კოდი ასეთი შედგენილობისაა:

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

namespace WpfAppPages3
{
   /// <summary>
   /// Interaction logic for Page2.xaml
   /// </summary>
   public partial class Page2 : Page
   {
      public Page2()
      {
         InitializeComponent();
      }
   }
}
```

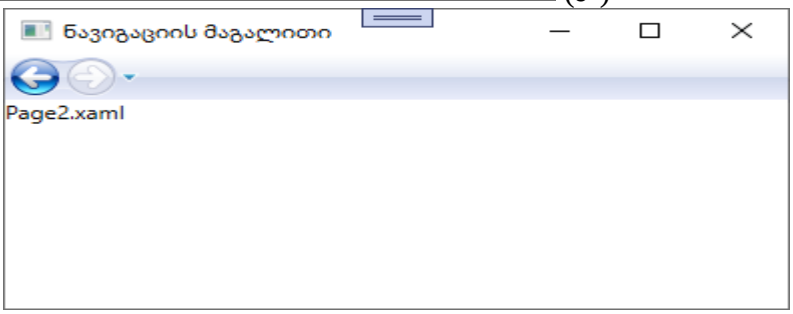
12) NavExample.xaml ფაილში დავამატოთ Source-ატრიბუტი, რომელიც მიუერთდება კარკასის გაშვებისას Page1.xaml საწყისი გვერდის შიგთავსს.

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

13) ავამუშავოთ პროექტი. შევამოწმოთ ღილაკების მუშაობისუნარიანობა (ნახ.3.22.-ა,ბ);



(ა)



(ბ)

ნახ.3.22 - (ა) Page1 და (ბ) Page2

დასკვნა:

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

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

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

[ნაწილი_2: მაგალითის გაგრძელება]

14) Page1-ზე დავამატოთ სახელმინიჭებული ტექსტური ველი შემდეგი სახით:

```
<Page x:Class="WpfAppPages3.Page1"
   xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
   xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
   Title="Page1"
      >
```

```
  <StackPanel>
    <TextBlock TextAlignment="Center" FontSize="24">გვერდი 1</TextBlock>
    <TextBlock></TextBlock>
    <Label> შეიტანეთ თქვენი სახელი: </Label>
    <TextBox Name="nameBox" Width="200"></TextBox>
    <TextBlock></TextBlock>
    <TextBlock>
      <Hyperlink Click="LinkClicked">მე-2 გვერდზე</Hyperlink>
    </TextBlock>
  </StackPanel>
</Page>
```

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

15) შევცვალოთ Page1 გვერდის კოდი შემდეგი სახით;

```
using System;
. . .
namespace WpfAppPages3
{
  public partial class Page1 : Page
  {
    public Page1()
    {
      InitializeComponent();
    }
    private void LinkClicked(object sender, RoutedEventArgs e)
    {
      Page2 page2 = new Page2();
      page2.Message = nameBox.Text + " !!!"; // ინფორმაციის „ჩაკერვა“ ობიექტში
      this.NavigationService.Navigate(page2);
    }
  }
}
```

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

```
<Page x:Class="WpfAppPages3.Page2"
```

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

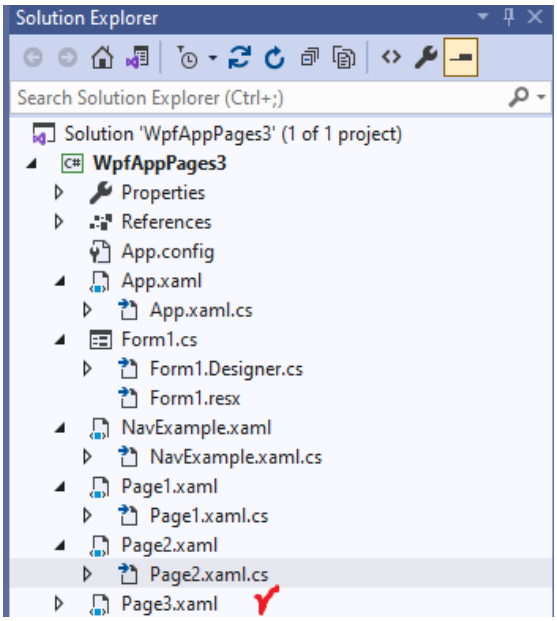
Label ობიექტს მივანიჭეთ სახელი, რათა კოდიდან შეიძლებოდეს მასზე მიმართვა.

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

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;

namespace WpfAppPages3
{
    public partial class Page2 : Page
    {
        public Page2()
        {
```

```
            InitializeComponent();
        }
        string message;
        public string Message
        {
            set { message = value; }
        }

        private void Page_Loaded(object sender, RoutedEventArgs e)
        {
            nameLabel.Content = message;
        }

        private void LinkClicked(object sender, RoutedEventArgs e)
        {
            Page3 page3 = new Page3();
            this.NavigationService.Navigate(page3);
        }
    }
}
```

18) დავამატოთ პროექტს მესამე გვერდი - Page3. შედეგი (ნახ.3.23);



ნახ.3.23

19) Page3 გვერდისათვის შევავსოთ xaml-კოდი:

```
<Page x:Class="WpfAppPages3.Page3"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Title="Page3">
```

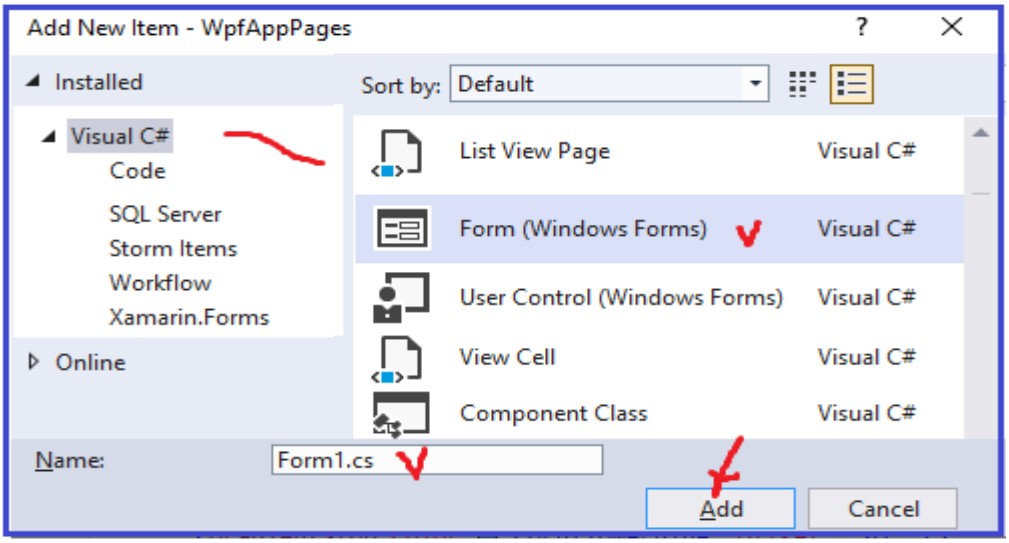
```
    <StackPanel>
        <!--გვერდის კონტექსტის სათაური -->
        <TextBlock TextAlignment="Center" FontSize="24">გვერდი 3
                <TextBlock.Margin>0,0,0,10</TextBlock.Margin>
        </TextBlock>

        <!--პირველი ღილაკი-->
      <Button Content="Push Me!" FontSize="22" Width="175" Height="50"
Click="Button_Click">
        <Button.Effect>
          <DropShadowEffect />
        </Button.Effect>
      </Button>

      <!--მეორე ღილაკი-->
      <Button FontSize="22" Height="50" Width="175" Margin="0,10" Click="Button_Click">
        "დამკლიკე"
        <Button.Effect>
          <DropShadowEffect />
        </Button.Effect>
        <Button.Foreground>
          <LinearGradientBrush StartPoint="1,0" EndPoint="0,0">
            <GradientStop Color="Red" Offset="0" />
            <GradientStop Color="Orange" Offset=".17" />
            <GradientStop Color="Yellow" Offset=".33" />
            <GradientStop Color="Green" Offset=".5" />
            <GradientStop Color="CornflowerBlue" Offset=".67" />
            <GradientStop Color="Blue" Offset=".84" />
            <GradientStop Color="BlueViolet" Offset="1" />
          </LinearGradientBrush>
        </Button.Foreground>
        <Button.Background>
          <LinearGradientBrush StartPoint="0,0" EndPoint="1,0">
            <GradientStop Color="Red" Offset="0" />
            <GradientStop Color="Orange" Offset=".17" />
            <GradientStop Color="Yellow" Offset=".33" />
            <GradientStop Color="Green" Offset=".5" />
            <GradientStop Color="CornflowerBlue" Offset=".67" />
```

```
            <GradientStop Color="Blue" Offset=".84" />
            <GradientStop Color="BlueViolet" Offset="1" />
          </LinearGradientBrush>
        </Button.Background>
      </Button>
    </StackPanel>
</Page>
```

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

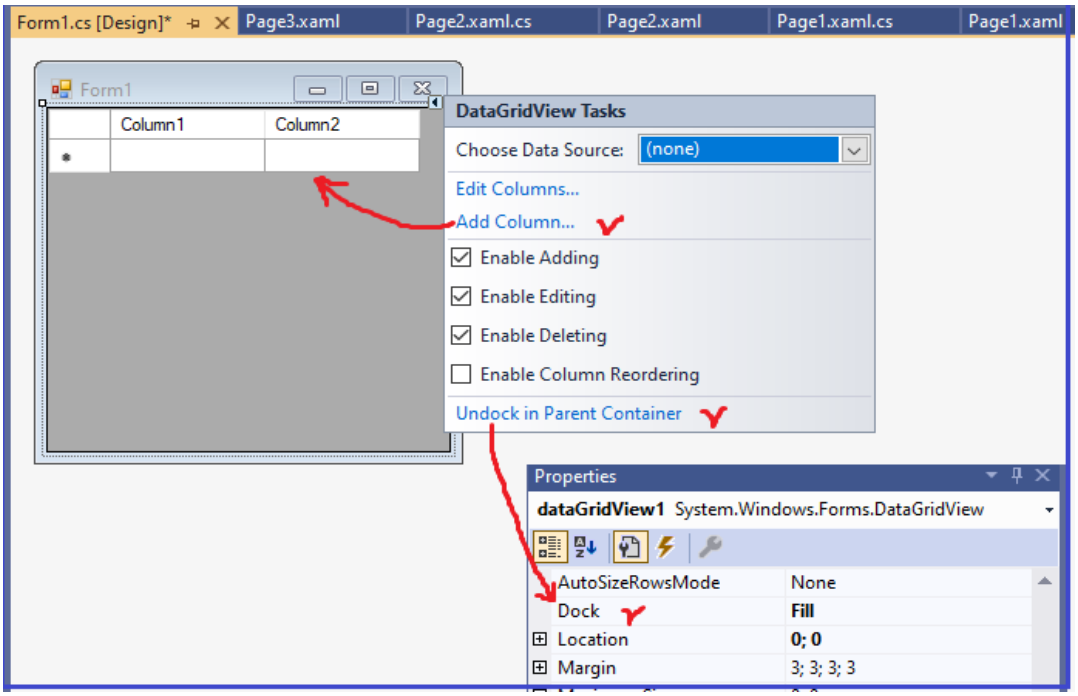
სანამ შევავსებთ Page3 გვერდის კოდის ნაწილს, საჭიროა პროექტს დაემატოს ახალი ფორმა. ამით შესაძლებელია WPF და Windows Forms ტექნოლოგიების ერთობლივად მუშაობის დემონსტრირება. ღილაკებზე დავდოთ ფორმის ამუშავების კოდი.

20) დავამატოთ WpfApPages3-ში ახალი ფორმა Form1.cs (ნახ.3.24).



ნახ.3.24

21) Form1 ფორმაზე ტულბოქსიდან დავდოთ DataGridView ელემენტი. დავაყენოთ მისი თვისება Dock=Fill და დავამატოთ ინტელექტუალური დესკრიპტორიდან (SmartTag) ორი სვეტი (ნახ.3.25).



ნახ.3.25

22) ღილაკების საერთო დამმუშავებელი Page3.xaml.cs კოდის ნაწილში შევავსოთ ასე:

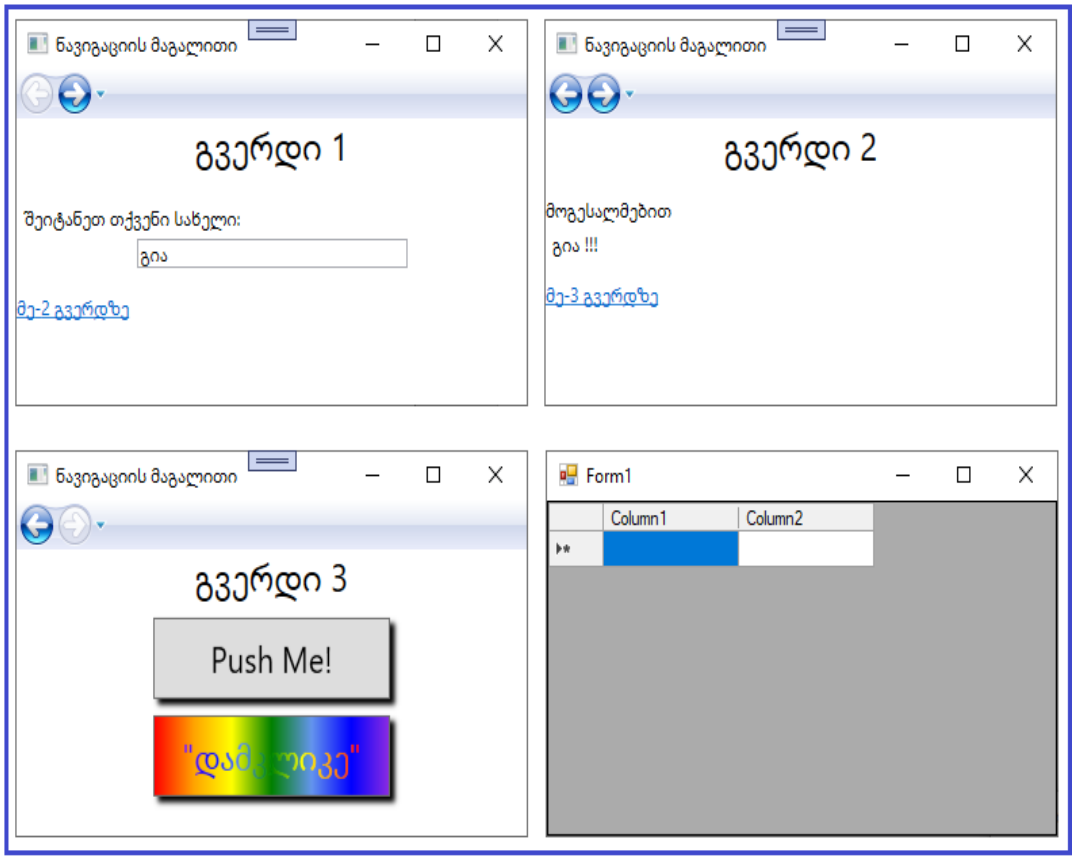
```
using System;
using System.Collections.Generic;
using System.Text;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;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

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

23) ავამუშავოთ პროექტი. დავაკვირდეთ ახ

ალი გვერდის დიზაინს. იხსნება Form1 -იც, რაც ადასტურებს WPF და Windows Form ტექნოლოგიების ერთობლივი მუშაობის შესაძლებლობას.

შედეგები, სამი გვერდი (Page1, Page2 და Page3) და ერთი ვინდოუსის ფორმა (Form1 - Windows Forms ტიპის) მოცემულია 3.26 ნახაზზე.



ნახ.3.26. გვერდების ნავიგაციის მაგალითის შედეგები
WPF აპლიკაციისთვის (3 გვერდი და ერთი ვინფორმა DataGridViev-ით)

3.3. ინტერფეისის პროგრამის დაკავშირება მონაცემთა ბაზასთან

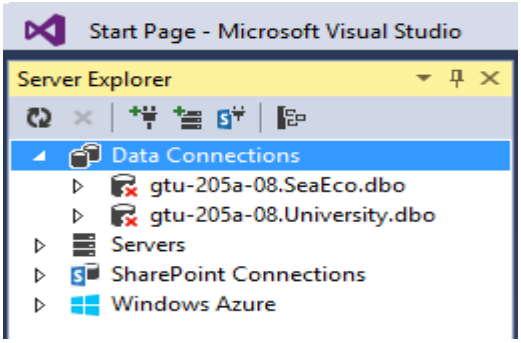
Visual Studio .NET სისტემას აქვს სტანდარტული ოსტატი პროგრამებისა და დიზაინერების სიმრავლე, რომელთა საშუალებითაც ადვილად და ეფექტურად ხორციელდება მონაცემებთან წვდომა აპლიკაციების დამუშავების პროცესში.

ამასთან, ADO.NET (ActiveX Data Object) ობიექტური მოდელის ყველა შესაძლებლობა მისაწვდომია პროგრამულად, რაც უზრუნველყოფს არასტანდარტული ფუნქციების რეალიზაციის ან დანართების აგების შესაძლებლობას, რომელიც მომხმარებლის მოთხოვნილებებზეა ორიენტირებული [1,19].

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

C# პროგრამულ აპლიკაციაში არსებობს მონაცემთა ბაზასთან მიერთების რამდენიმე ხერხი. ამის განხორციელება ყველაზე მარტივია Visual Studio .NET-ის გრაფიკული ინსტრუმენტით. მონაცემთა წყაროსთან (DataSource) მიერთებისა და მართვისათვის გამოიყენება ფანჯარა Server Explorer.

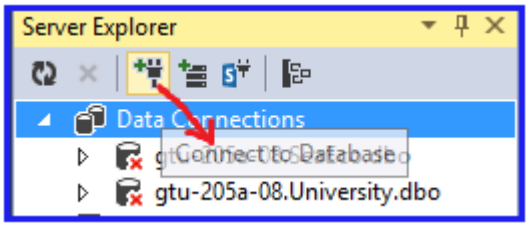
ძირითადი ამოცანა, რომელსაც აქ განვიხილავთ, არის ADO.NET პროგრამული პაკეტის (დრაივერის) გამოყენებით მომხმარებელთა სამუშაო ინტერფეისის დამუშავების სადემონსტრაციო მაგალითის აგება. ამასთანავე, მონაცემთა ბაზების სახით უნდა გამოვიყენოთ Ms SQL Server-ის management Studio-ში აგებული ცხრილები (Tables). .NET სამუშაო გარემოს ჩატვირთვის შემდეგ საჭიროა Server Explorer-ის გახსნა და ბაზებთან კავშირის შემოწმება (მაგალითად, ნახ.3.27).



ნახ.3.27. არაა კავშირი ბაზასთან

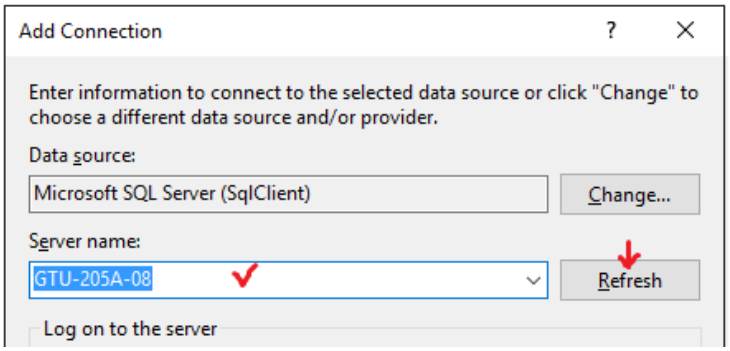
ახლა განვიხილოთ MsSQL Server ბაზასთან მუშაობის საკითხები.

Server Explorer-ის გრაფიკული მენიუდან ავირჩიოთ Connect to Database (ნახ.3.28).



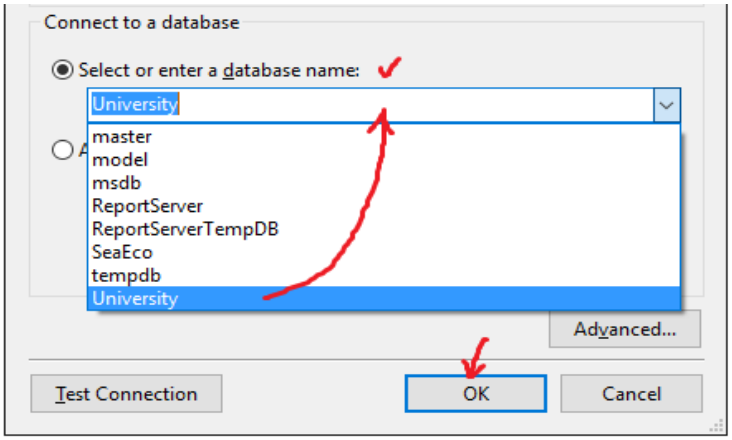
ნახ.3.28. დაკავშირების პროცესი

3.29 ფანჯრის Server name ველში ჩავწეროთ GTU-205A-08 *(თქვენ კომპიუტერზე იქნება სხვა მნიშვნელობა)*. შემდეგ - Refresh ღილაკი.



ნახ.3.29

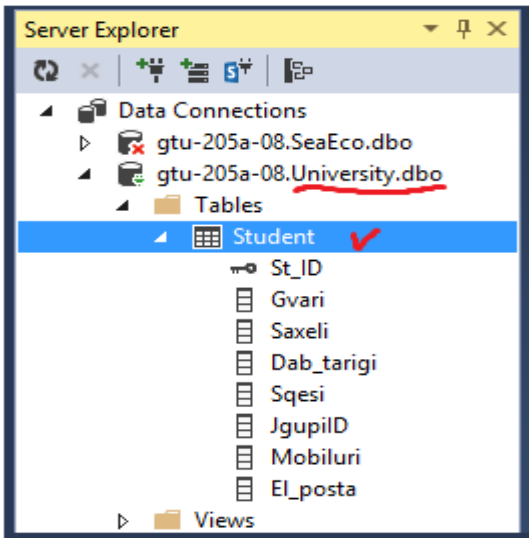
ამის შემდეგ Add Connection-ის ქვედა ნაწილში, კერძოდ, Select or enter Database name ველში ავირჩევთ სერვერზე არსებული ბაზებიდან ჩვენთვის საჭიროს (ნახ.3.30).



ნახ.3.30

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

შედეგად Server Explorer-ში გამოჩნდება 3.31 ნახაზზე მოცემული ფანჯარა.



ნახ.3.31. კავშირი შედგა University ბაზასთან

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

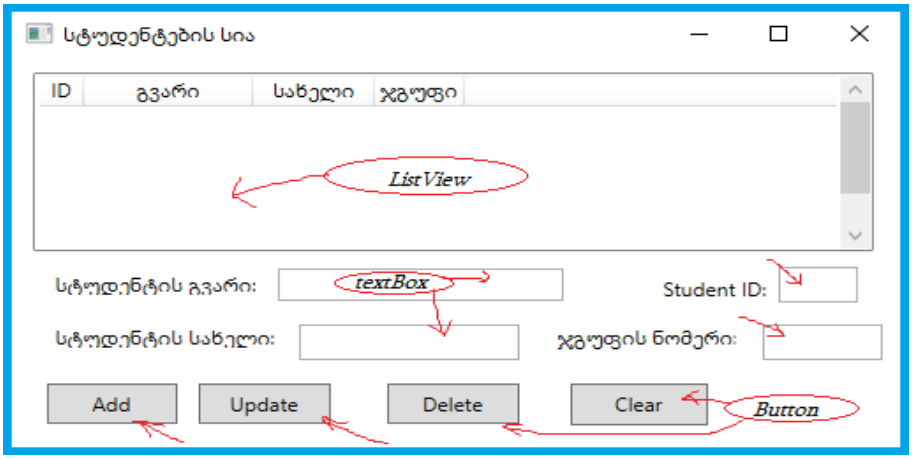
ამგვარად, University მონაცემთა ბაზა მზადაა პროგრამულ აპლიკაციასთან სამუშაოდ.

3.6. WpfApp კოდიდან SQL Server ბაზის წვდომა, CRUD ოპერაციები და და მოთხოვნების დამუშავება

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

პროცესისა და შედეგების საილუსტრაციოდ დავსვათ კონკრეტული ამოცანა: „საჭიროა მომხმარებლის ინტერფეისის აგება ListView მართვის ელემენტის გამოყენებით, რომელშიც აისახება SQL Server-ის University.dbo მონაცემთა ბაზის განსაზღვრული ატრიბუტების (ველების) მნიშვნელობები

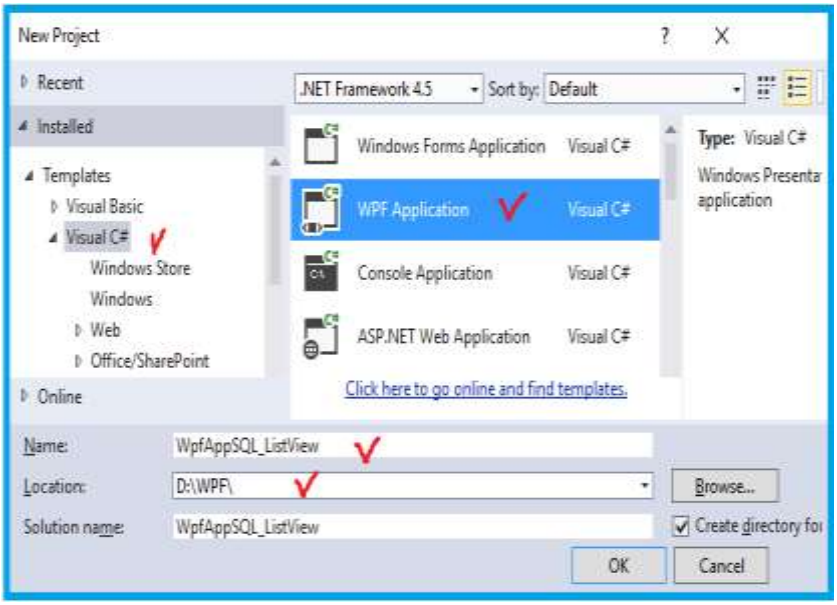
(ბაზის ჩანაწერები) და შესაძლებელი იქნება ინტერფეისის Add, Update და Delete ღილაკებით მონაცემთა მანიპულირება (ბაზის განახლების ოპერაციების შესრულება). 3.32 ნახაზზე მოცემულია ინტერფეისის მოდელი Visual Studio .NET სამუშაო გარემოში, Visual C# და WPF Application-ის საფუძველზე.



ნახ.3.32. მომხმარებლის ინტერფეისის მოდელი

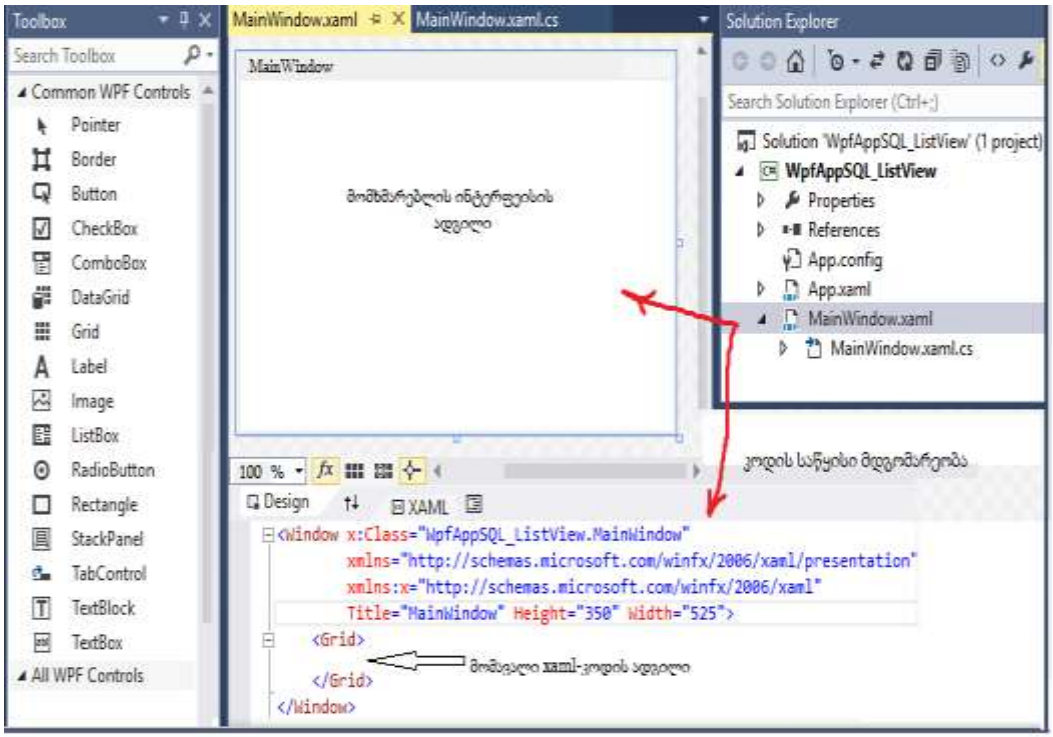
- **პროექტის აგება.**

ავამუშავოთ Visual Studio .NET და შევქმნათ ახალი პროექტი სახელით WpfAppSQL_ListView, რომელსაც მოვათავსებთ D:\\ დისკოს WPF ფოლდერში (ნახ.3.33).



ნახ.3.33. პროექტის შექმნა

შედეგად ვღებულობთ 3.34 ნახაზზე მოცემულ მდგომარეობას.



ნახ.3.34. WpfAppSQL_LisView პროექტის საწყისი მდგომარეობა

Toolbox პანელიდან ინტერფეისზე უნდა გადმოვიტანოთ მართვის ელემენტები და განვათავსოთ ისე, როგორც 3.32 ნახაზზეა ნაჩვენები.

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

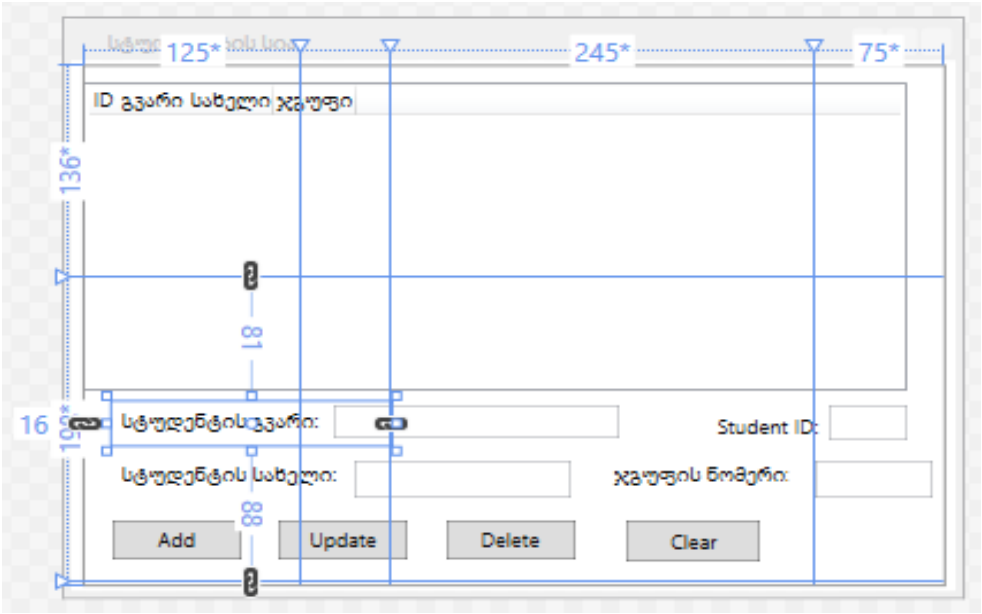
ღილაკების მოვლენების შესაბამისი მეთოდები (კოდები) ჯერ არაა დაწერილი C# ენაზე.

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

```
  <Grid.ColumnDefinitions>
    <ColumnDefinition Width="125*" />
    <ColumnDefinition Width="52*" />
    <ColumnDefinition Width="245*" />
    <ColumnDefinition Width="75*" />
  </Grid.ColumnDefinitions>
  <Grid.RowDefinitions>
    <RowDefinition Height="136*" />
    <RowDefinition Height="198*" />
    <RowDefinition Height="2*" />
 </Grid.RowDefinitions>
 <ListView Margin="-1,10,21,124" Name="listView1"
       ItemsSource="{Binding}" MinWidth="250" MinHeight="100"
                Grid.ColumnSpan="4" Grid.RowSpan="2">
  <ListView.View>
   <GridView>
         <GridViewColumn Header="ID" DisplayMemberBinding=
                "{Binding Path=St_ID}"></GridViewColumn>
    <GridViewColumn Header="გვარი" DisplayMember
              Binding="{Binding Path=Gvari}"></GridViewColumn>
    <GridViewColumn Header="სახელი" DisplayMember
             Binding="{Binding Path=Saxeli}"></GridViewColumn>
    <GridViewColumn Header="ჯგუფი" DisplayMemberBin
              ding="{Binding Path=JgupiID}"></GridViewColumn>
   </GridView>
   </ListView.View>
  </ListView>
<TextBox Margin="19,84,113,93" Name="textBox1" Grid.Row="1"
          Grid.Column="1" Grid.ColumnSpan="2" />
<TextBox Height="23" Margin="0,0,7,55" Name="textBox2"
       VerticalAlignment="Bottom" Grid.Row="1" Horizontal
          Alignment="Right" Width="68" Grid.Column="3" />
<Label Margin="16,81,0,88" Name="label1" Grid.Row="1"
     Content="სტუდენტის გვარი:" Grid.ColumnSpan="2"> </Label>
<Label Height="29" Margin="124,0,10,54" Name="label2"
     VerticalAlignment="Bottom" Grid.Row="1" Content="ჯგუფის
        ნომერი:" Grid.Column="2"/>
```

```
<Button Height="26" Margin="16,0,0,14" Name="btnAdd"
         VerticalAlignment="Bottom" Click="btnAdd_Click"
         Grid.Row="1" HorizontalAlignment="Left"
         Width="74">Add</Button>
<Button Height="26" Margin="112,0,0,14" Name="btnUpdate"
        VerticalAlignment="Bottom" Click="btnUpdate_Click"
        HorizontalAlignment="Left" Width="75"
        Grid.ColumnSpan="3" Grid.Row="1">Update</Button>
<Button Height="26" Margin="32,0,0,14" Name="btnDelete"
         VerticalAlignment="Bottom" Click="btnDelete_Click"
       Grid.Column="2" HorizontalAlignment="Left" Width="75"
       Grid.Row="1">Delete</Button>
<Button Height="27" Margin="136,0,31,13" Name="btnClear"
       VerticalAlignment="Bottom" Click="btnClear_Click"
       Grid.Column="2" Grid.Row="1">Clear</Button>
<Label Content="Student ID:" Grid.Column="2" Grid.Row="1"
       Height="26" HorizontalAlignment="Left"
       Margin="184,84,0,0" Name="label3"
       VerticalAlignment="Top" Width="70"
       Grid.ColumnSpan="2" />
<TextBox Grid.Column="3" Grid.Row="1" Height="23"
       HorizontalAlignment="Left" Margin="9,83,0,0"
       Name="textBox3" VerticalAlignment="Top" Width="45" />
<Label Margin="16,115,23,55" x:Name="label1_Copy"
       Grid.Row="1" Content="სტუდენტის სახელი:"
       Grid.ColumnSpan="2"/>
<TextBox x:Name="textBox4" Grid.Column="1"
      HorizontalAlignment="Left" Height="23"
      Margin="31,120,0,0" Grid.Row="1" TextWrapping="Wrap"
      VerticalAlignment="Top" Width="125"
      Grid.ColumnSpan="2"/>
 </Grid>
</Window>
```

3.35 ნახაზზე მოცემულია ლისტინგის შესაბამისი ინტერფეისის დიზაინის სურათი.



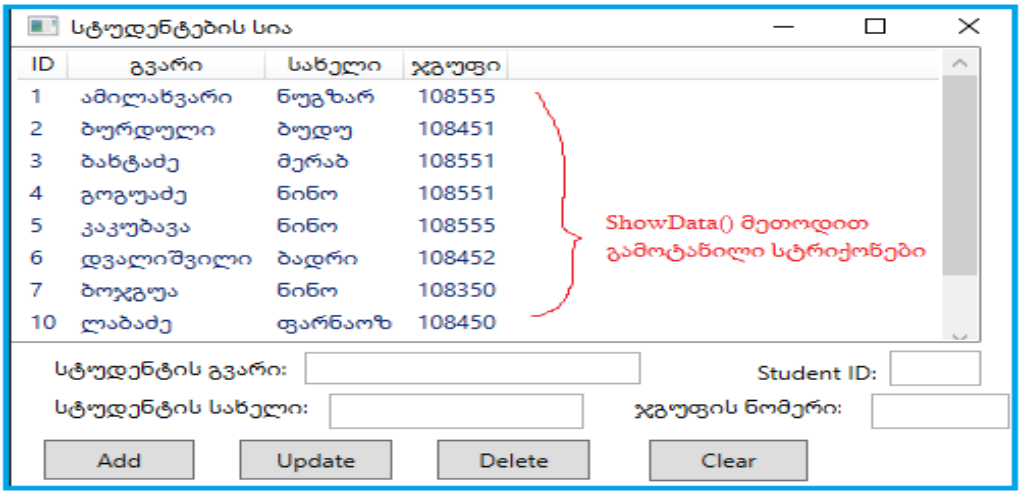
ნახ.3.38. ინტერფეისის სტრუქტურა

MsSQL_Server ბაზის Student ცხრილის მონაცემთა სტრიქონების გამოსატანად ListView-ში „ **studID გვარი სახელი ჯგუფი** “ ფორმატით, დავწეროთ ShowData() მეთოდი, რომელსაც შემდგომ Add, Update და Delete ღილაკებიც გამოიყენებს მონაცემთა ასახვისათვის ეკრანზე (ლისტინგი_ShowData() და ნახ.3.36).

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

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

```
// -- ლისტინგი_Loaded მეთოდი -------
private void Window_Loaded(object sender,RoutedEventArgs e)
   {
       ShowData();
   }
```

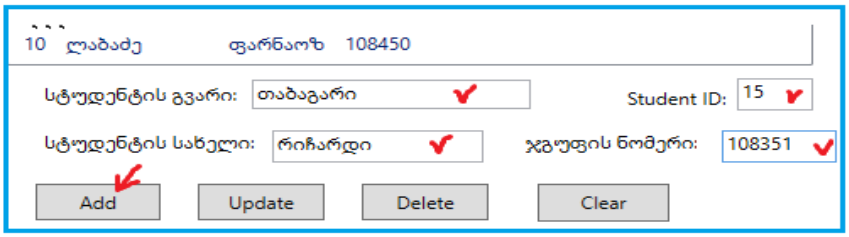


ნახ.3.36. მონაცემთა ბაზის ცხრილის სტრიქონები, გამოტანილი Window_Loaded მეთოდით

- **Add:** ბაზის ცხრილში ახალი სტუდენტის დასამატებლად უნდა შეივსოს *Student ID, გვარი, სახელ და ჯგუფის ნომერი,* შესაბამის textBox-ებში და Add ღილაკით ავამოქმედოთ შენახვის პროცედურა. იგი არ მუშაობს, რადგან Add-ის შესაბამისი მეთოდის C# კოდი ჯერ არ არსებობს. შევქმნათ იგი (იხ. .ლისტინგი_Add, ნახ.3.37).

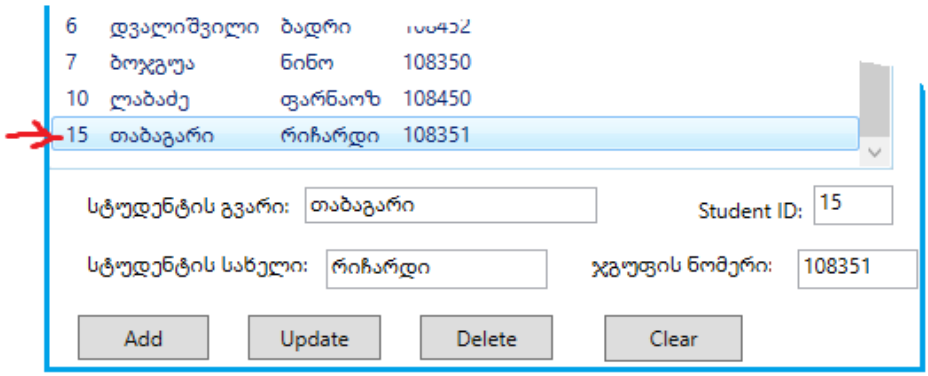
```
//--- ლისტინგი_Add() ღილაკი ----------------
private void btnAdd_Click(object sender, RoutedEventArgs e)
   {
      string Gvari = textBox1.Text;
      string Saxeli = textBox4.Text;
      string JgupiID = textBox2.Text;
      string St_ID = textBox3.Text;
      SqlConnection con = new SqlConnection(@"Data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

```
    comm.Parameters.AddWithValue("@Saxeli", textBox4.Text);
    comm.Parameters.AddWithValue("@JgupiID", textBox2.Text);
    comm.Parameters.AddWithValue("@St_ID", textBox3.Text);
    comm.ExecuteNonQuery();
    con.Close();
    ShowData();
}
```



ნახ.3.37. ახალი სტუდენტის მონაცემების მომზადება

Add ღილაკის ამოქმედების შედეგი მოცემულია 3.38 ნახაზზე.



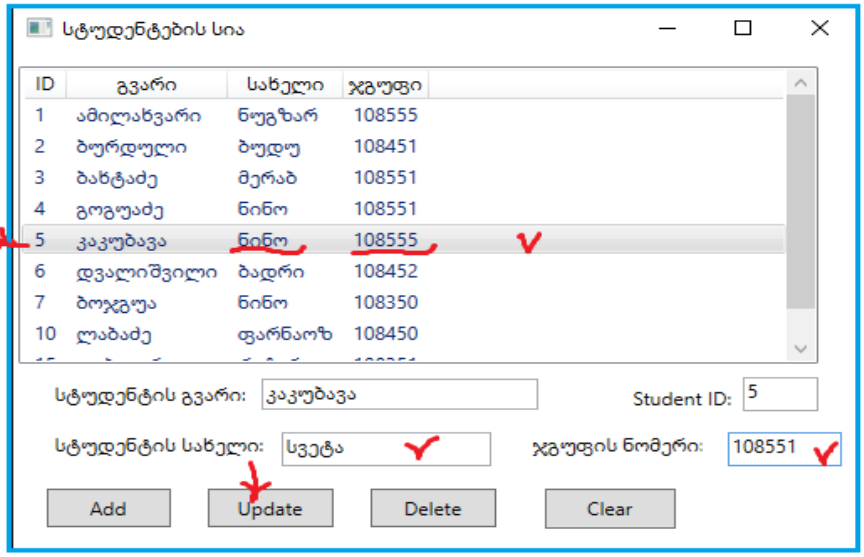
ნახ.3.38. StudentID=15 ჩაემატა ბაზაში

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

```
// -- ლისტინგი_Clear() ღილაკი ----------

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

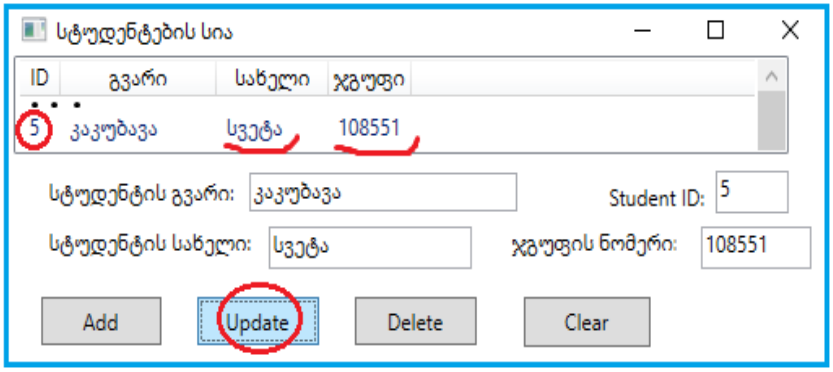
- **Update:** ბაზის ცხრილში მონაცემთა მნიშვნელობების შესაცვლელად დავწეროთ შემდეგი მეთოდი (ლისტინგი_Update, ნახ.3.39).

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



ნახ.3.39. სტუდენტის მონაცემების: სახელის და ჯგუფის შეცვლა

Update ღილაკის ამოქმედებით Student_ID=5 სტრიქონში მოხდება სახელის და ჯგუფის ნომრის შეცვლა ახალი მნიშვნელობებით. შედეგის სტრიქონი მოცემულია 3.40 ნახაზზე.

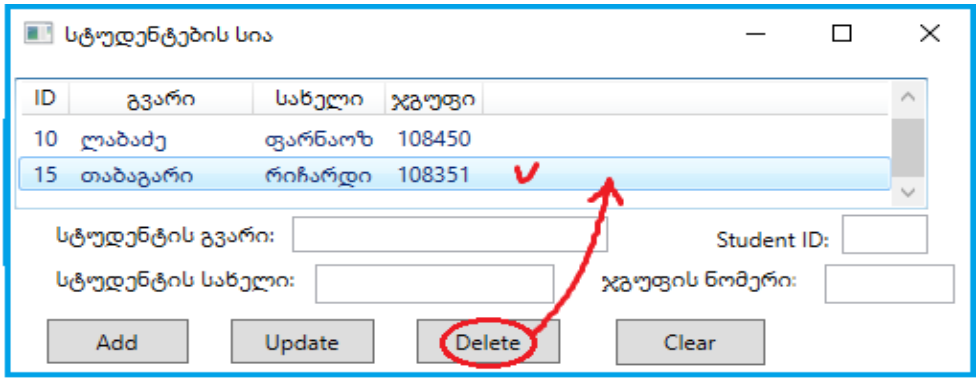


ნახ.3.40. Update() მეთოდით შეცვლილი სტრიქონი

- **Delete:** ბაზის ცხრილში სტრიქონის წასაშლელად დავწეროთ შემდეგი Delete მეთოდი (ლისტინგი_Delete, ნახ.3.41).

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

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



ნახ.3.41. Delete() მეთოდით წაიშლება წინასწარ მონიშნული სტრიქონი (St_ID=15)

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

ქვემოთ, ლისტინგში მოცემულია C# პროგრამის მთლიანი კოდი.

```
//--- ლისტინგი_-----  მთლიანი კოდი ---------------------------
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using System.Data.SqlClient;   // !!!
using System.Data;          // !!!
namespace WpfAppSQL_ListView
{
  public partial class MainWindow : Window
   {
    public MainWindow()
     {
       InitializeComponent();
     }
```

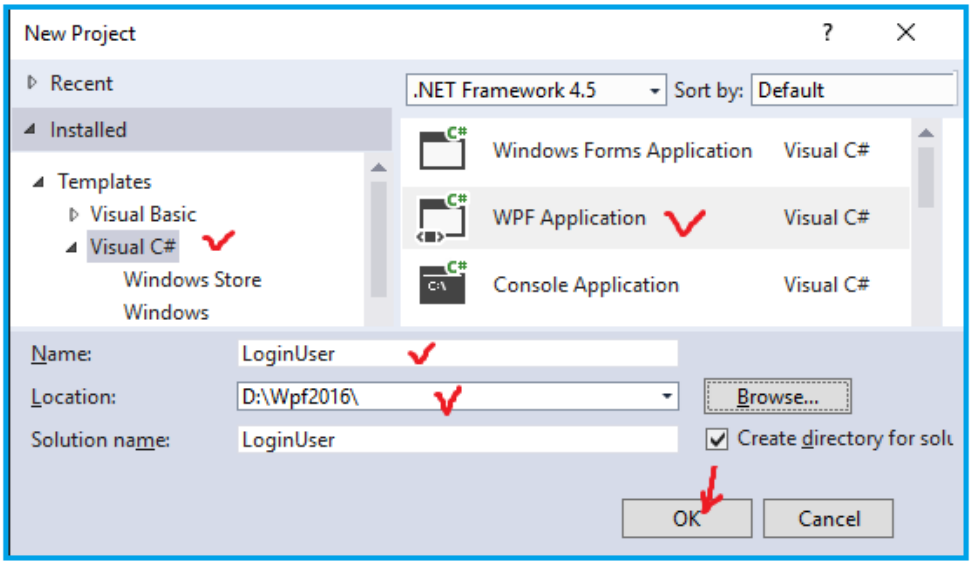
```
private void Window_Loaded(object sender,RoutedEventArgs e)
      {
         ShowData();
      }
     public void ShowData()
      {
        SqlConnection con = new SqlConnection(@"Data Source=GTU-205A-08;Initial
Catalog=University;Integrated Security=True");
        con.Open();
  SqlCommand comm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

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

private void btnUpdate_Click(object sender,RoutedEventArgs e)
 {
  if (listView1.SelectedItems.Count > 0)
   {
     DataRowView drv = (DataRowView)listView1.SelectedItem;
```

```
    string id = drv.Row[0].ToString();
    SqlConnection con = new SqlConnection(@"Data
            Source=GTU-205A-08;Initial
            Catalog=University;Integrated Security=True");
    con.Open();
    SqlCommand comm = new SqlCommand("update Student  set
             Gvari=@Gvari, Saxeli=@Saxeli,JgupiID=@JgupiID where
              St_ID=@St_ID", con);
    comm.Parameters.AddWithValue("@St_ID", id);
   comm.Parameters.AddWithValue("@Gvari", textBox1.Text);
   comm.Parameters.AddWithValue("@Saxeli", textBox4.Text);
   comm.Parameters.AddWithValue("@JgupiID", textBox2.Text);
   comm.ExecuteNonQuery();
   con.Close();
   ShowData();
  }
 }
private void btnDelete_Click(object sender,RoutedEventArgs e)
 {
  if (listView1.SelectedItems.Count > 0)
  {
    DataRowView drv = (DataRowView)listView1.SelectedItem;
    string id = drv.Row[0].ToString();
    SqlConnection con = new SqlConnection(@"Data
                Source=GTU-205A-08;Initial
                Catalog=University;Integrated Security=True");
   con.Open();
   SqlCommand comm = new SqlCommand("delete from
                 Student where St_ID=@St_ID", con);
   comm.Parameters.AddWithValue("@St_ID", id);
   comm.ExecuteNonQuery();
   ShowData();
  }
 }
private void btnClear_Click(object sender, RoutedEventArgs e)
    {
       textBox1.Text = "";
       textBox2.Text = "";
       textBox3.Text = "";
       textBox4.Text = "";
    } } }
```

3.7. WPF-აპლიკაცია – მუშაობის უსაფრთხოების კონტროლი

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

შევქმნათ ახალი Wpf Application პროექტი Login2test სახელით და მოვათავსოთ ახლადშექმნილ ფოლდერში (ნახ.3.42).



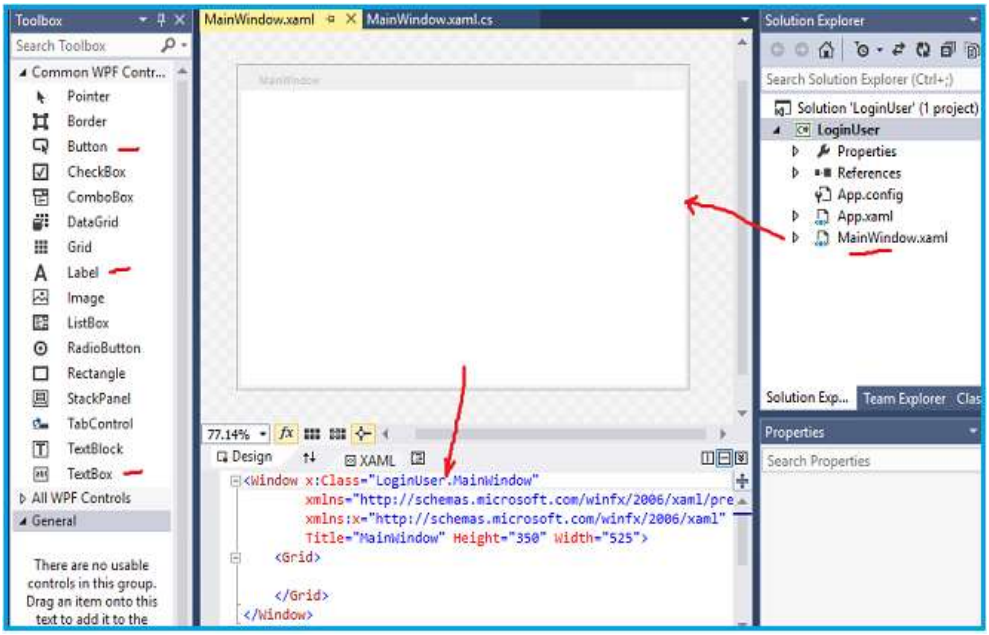
ნახ.3.42. პროექტის შექმნა

მიიღება სამუშაო გარემო (ნახ.3.43).

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

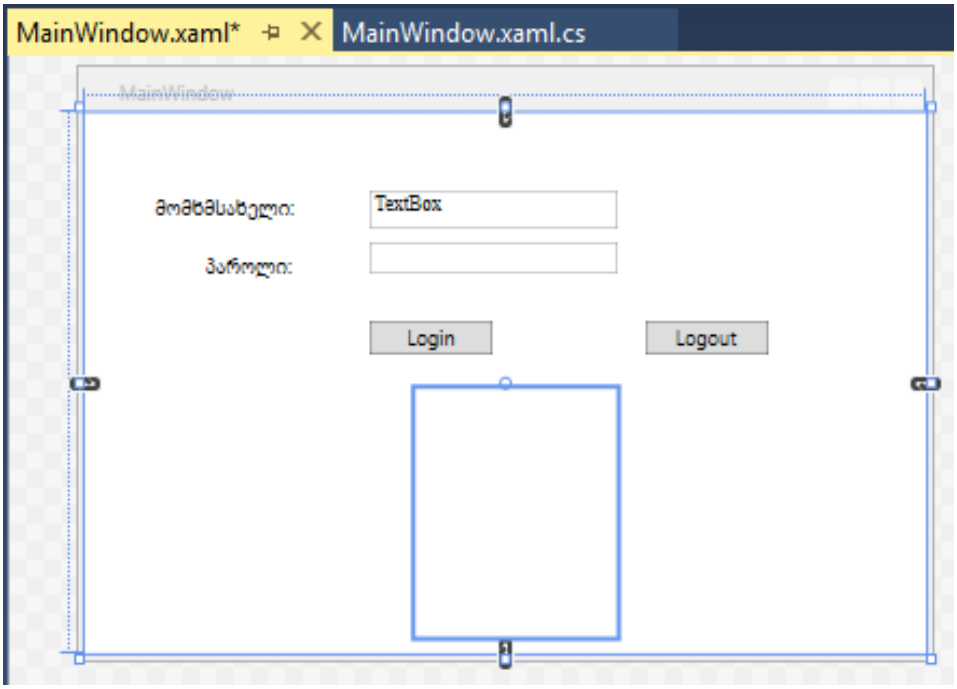
ფორმაზე დავდოთ აგრეთვე ორი Button, ერთი Login და მეორე Logout.

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



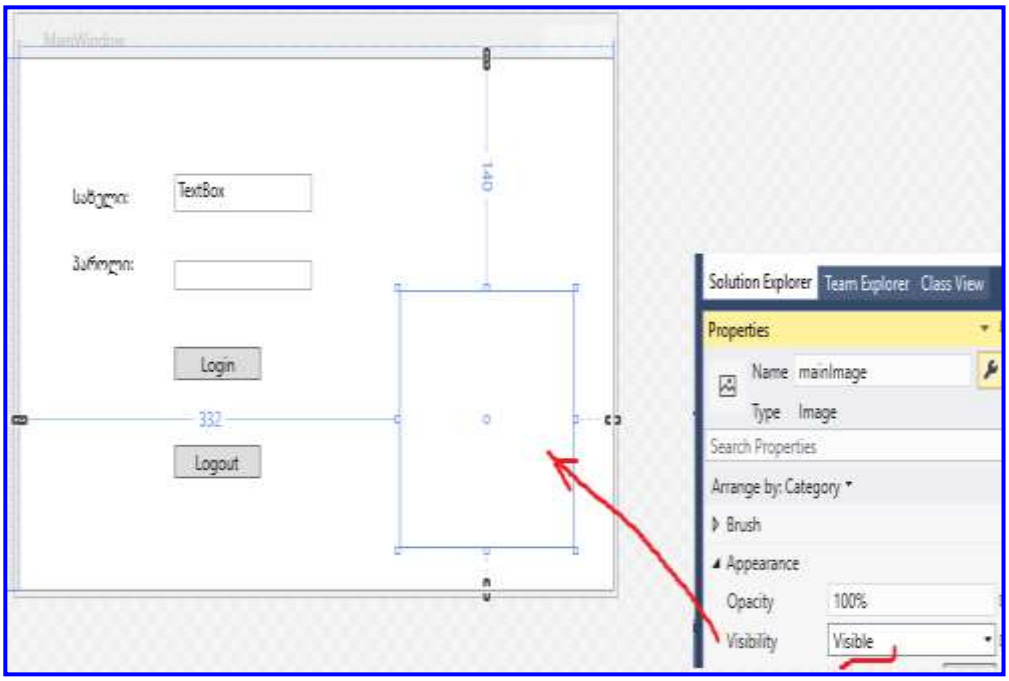
ნახ.3.43. პროექტის შექმნა

თუ სახელი ან/და პაროლი შეცდომითაა, მაშინ MessageBox-ში. გამოვა შეტყობინება არასწორი პასუხის შესახებ. დიზაინის ფორმა ნაჩვენებია 3.44 ნახაზზე.

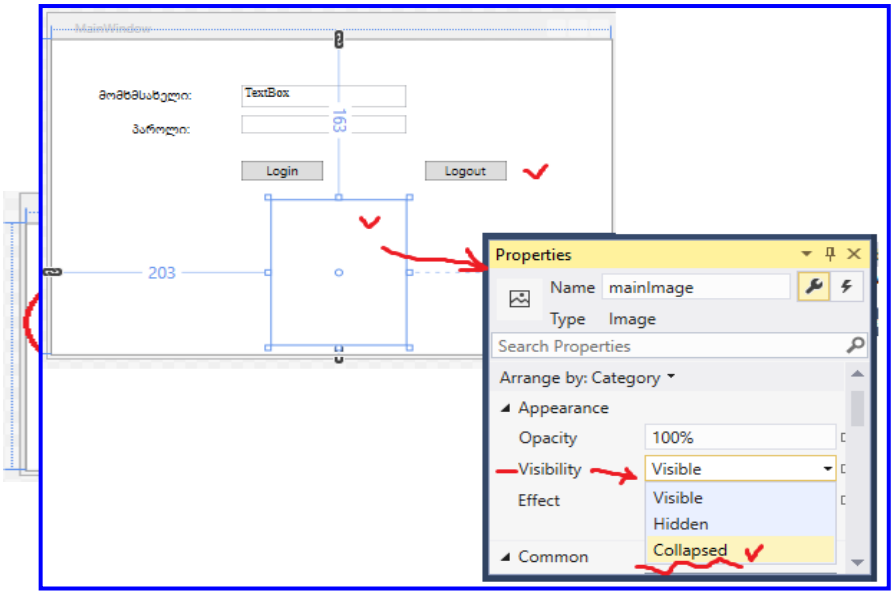


ნახ.3.44. ინტერფეისი

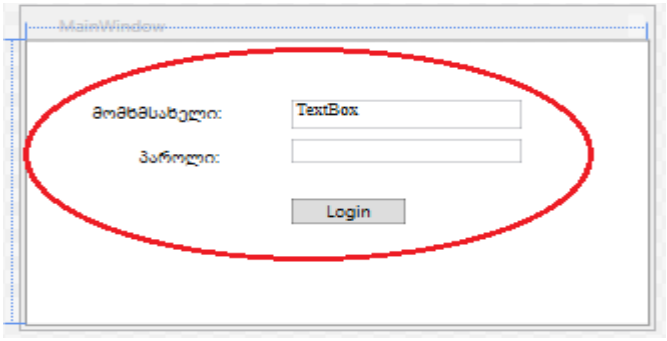
ელემენტების მდებარეობა შეიძლება შეიცვალოს დამპროექტებლის სურვილით. Properties-ში Image და Logout Button-ისთვის ხილვადობა დავაყენოთ ***Visibility="Collapsed"*** (ნახ.3.45).



ნახ.3.48. თვისება თავიდან Visible-შია



ნახ.3.46-ა. ბოლოს Collapsed-ში დაიმალა სურათი და Logout (ბ).



ნახ.3.46-ბ.

xaml - კოდი ჩავწეროთ შემდეგი ლისტინგის მიხედვით:

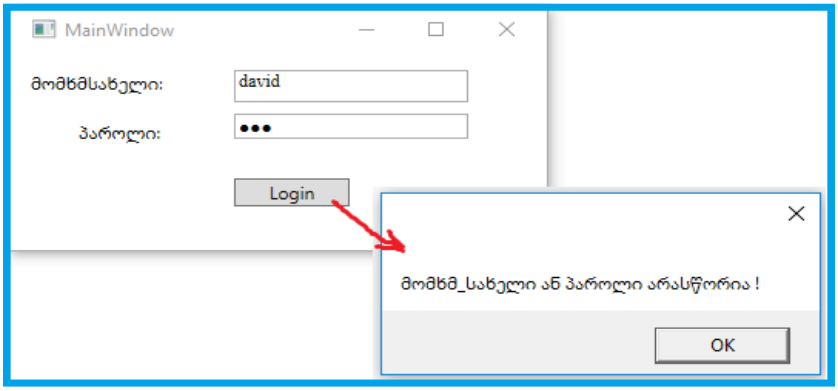
```
<!-- ლისტინგი_3.9 ----------------------------------------
<Window x:Class="test2Log.MainWindow"
    mlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Title="MainWindow" Height="350" Width="525">
        <Grid>
       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

C# კოდის ტექსტი მოცემულია ქვემოთ, ლისტინგში.

▪ Double click ღილაკზე Login და ჩავწეროთ კოდი:

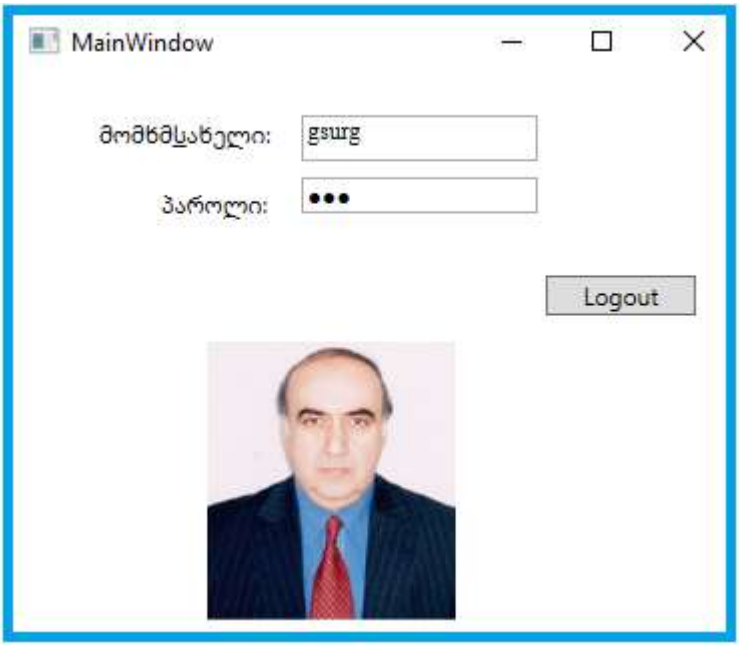
```
//-- ლისტინგი_ - Double click ----------------------------------
private void LoginBTN_Click(object sender, RoutedEventArgs e)
{
  if (!usernameTB.Text.Equals(“”) &&
                !PasswordBoxPB.Password.Equals(“”))
    {
      if (usernameTB.Text.Equals(“1″) &&
                    PasswordBoxPB.Password.Equals(“1″))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

ავამუშავოთ პროგარმა და შევიტანოთ არასწორი პაროლი (ნახ.3.47):



ნახ.3.47. არასწორი პაროლის შეტანის შემთხვევა

სწორი სახელის და პაროლის შემთხვევაში უნდა მივიღოთ 3.48 ნახაზზე მოცემული სურათი.



ნახ.3.48. სწორი პაროლის შეტანის შემთხვევა

Logout-ზე დაკლიკვით შევიტანოთ C# კოდი:

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

მთლიანი C# კოდი მოცემულია 3.12 ლისტინგში.

```
//--- ლისტინგი_3.12 -------------------------------------
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
```

```
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
namespace test2Log
{
  public partial class MainWindow : Window
  {
     public MainWindow()
     {  InitializeComponent();
     }
 private void LoginBTN_Click(object sender, RoutedEventArgs e)
  {
    if (!UserName.Text.Equals("") &&
                !PasswordBox.Password.Equals(""))
       {
         if (UserName.Text.Equals("gsurg") &&
                   PasswordBox.Password.Equals("123"))
          {
             mainImage.Visibility = Visibility.Visible;
             LoginBTN.Visibility = Visibility.Collapsed;
             LogoutBTN.Visibility = Visibility.Visible;
          }
          else
                MessageBox.Show("მომხმ_სახელი ან პაროლი
                                                    არასწორია !");
       }
       else
          MessageBox.Show("ინფორმაცია არაა !");
     }
 private void LogoutBTN_Click(object sender, RoutedEventArgs e)
     {  mainImage.Visibility = Visibility.Collapsed;
        LoginBTN.Visibility = Visibility.Visible;
        LogoutBTN.Visibility = Visibility.Collapsed;
     }
  }
}
```

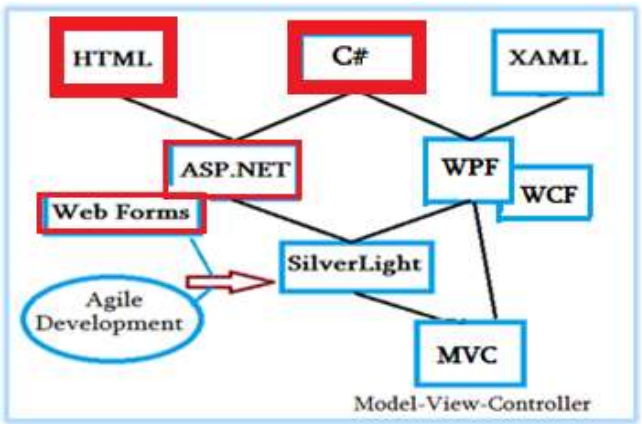
IV თავი

Web აპლიკაციის აგება ASP.NET ტექნოლოგიით

4.1. ASP ენა და ASP.NET – დეველოპმენტის ფრეიმვორკი

ASP (Active Server Pages) – ვებგვერდების აგების ინტერპრეტატორული ენაა, რომელიც შეიქმნა მაიკროსოფთში 1998 წელს, დაიწერა VBScript-ზე. მის ფაილებს აქვს .asp გაფართოება [7, 20].

ASP.NET - კომპილატორია, მაიკროსოფთის უფასო ვებ-ფრეიმვორკი (Ms Visual Studio .Net პლატფორმაზე), შეიქმნა 2002 წელს, დაიწერა C# -ზე. იგი მრავალფეროვანი ვებსაიტების და ვებაპლიკაციების შესაქმნელად იყენებს HTML, CSS და JavaScript-ს (ნახ.4.0). მაგალითად, HTML და C# ენებს (ან Visual Basic-ს). ფაილების გაფართოებაა - .aspx.



ნა.4.0

შეიძლება ითქვას, რომ ASP და ASP.NET სერვერის მხარის (Back-End) ტექნოლოგიებია. მათი საშუალებით ხორციელდება სხვადასხვა სახის მოდელე-ბის დეველოპმენტი, რომლებიც ძალზე აქტუალურია დღეს, კერძოდ:

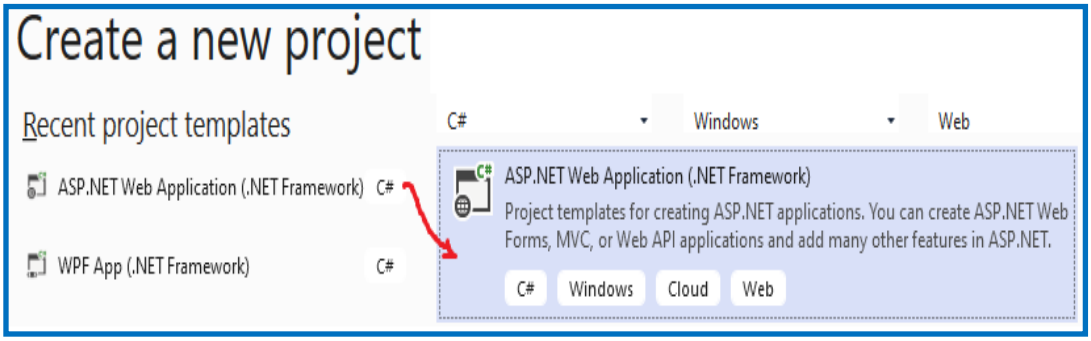
- ASP.NET Web Forms;
- ASP.NET MVC (Model – View - Controler);
- ASP.NET Web Pages;
- ASP.NET API;
- ASP.NET Core.

ეს მოდელები პროგრამულად რეალიზაბადია Ms Visual Studio .NET-ში.

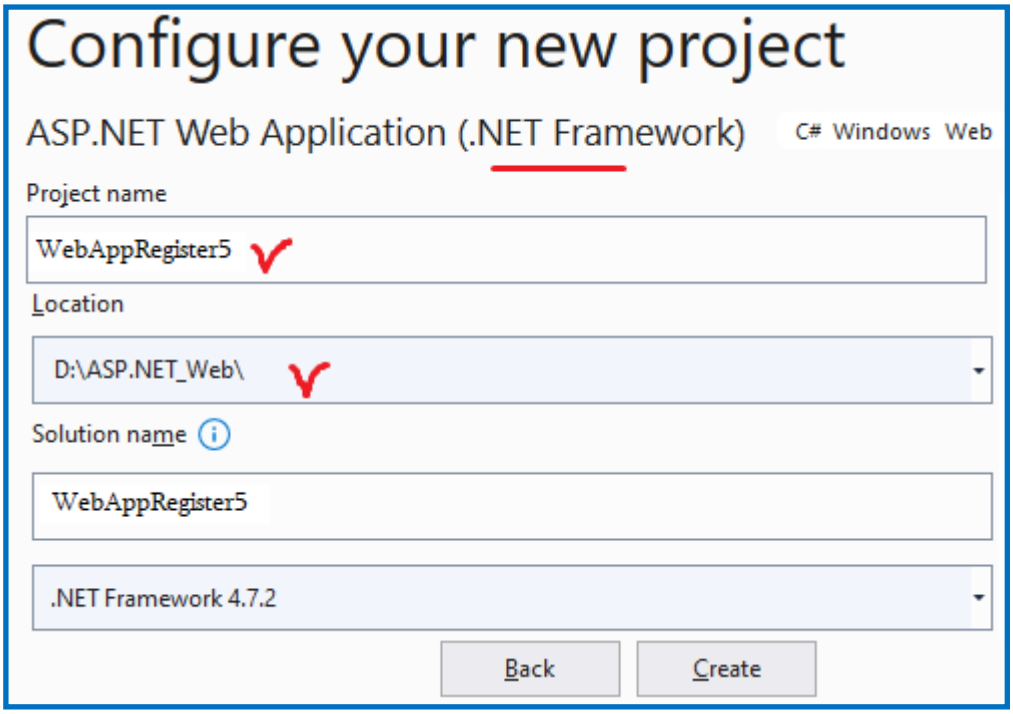
4.2. ASP.NET: ინტერაქტიული Web-გვერდის შექმნა

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

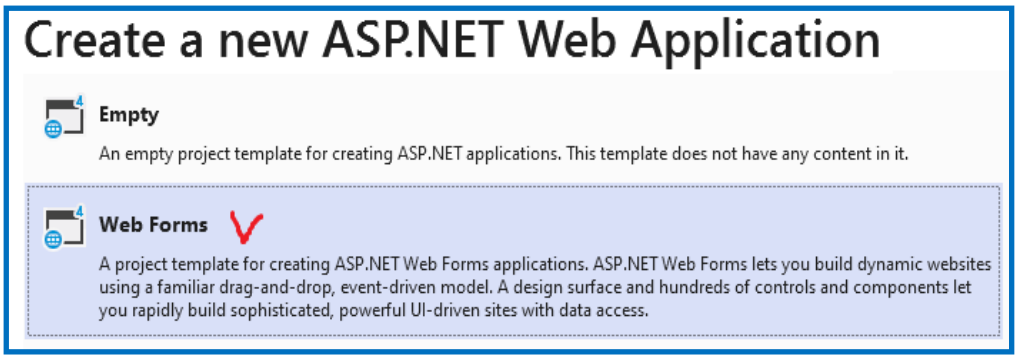
– შევქმნათ ახალი ASP.NET აპლიკაცია პროექტის სახელით WebAppRegister5 (ნახ.4.1, 4.2-ა,ბ);



ნახ.4.1. ახალი ASP.NET პროექტის შექმნა .NET Framework პლატფორმაზე

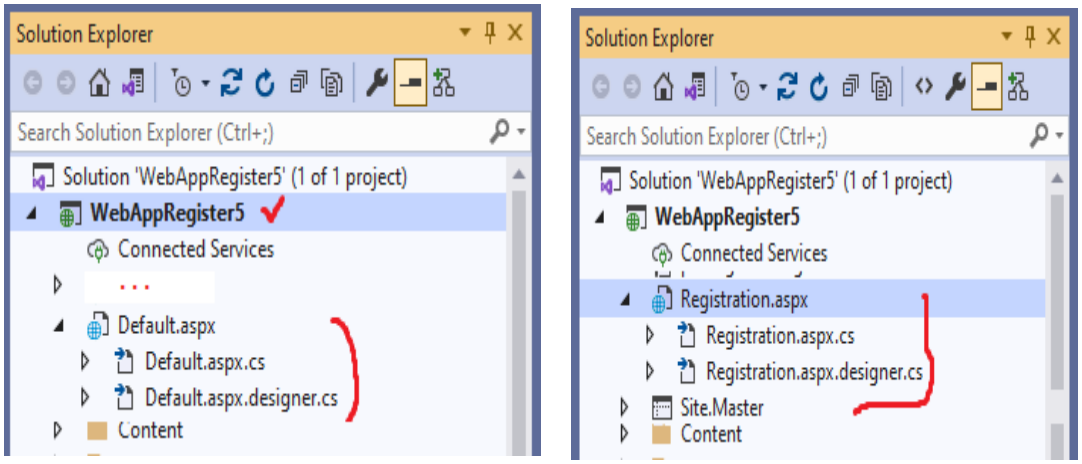


ნახ.4.2-ა



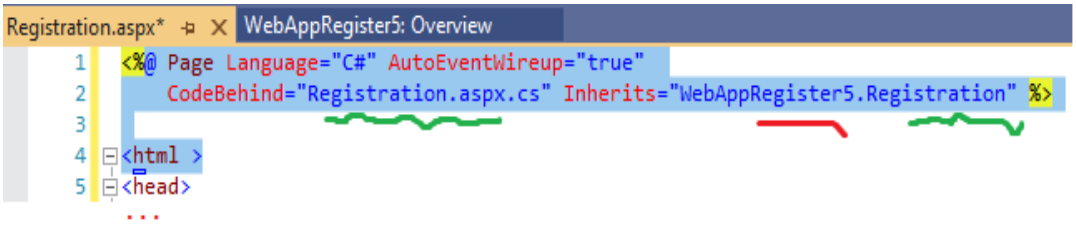
ნახ.4.2-ბ

– მიიღება Solution Explorer (ნახ.4.3 - მარცხენა). შევცვალოთ Default სახელი ახლით, კერძოთ, Registration-ით (ნახ.4.3 - მარჯვენა);



ნახ.4.3

– Registration.aspx ფაილის 1-ელ სტრიქონს ექნება 4.3 ნახაზზე ნაჩვენები სახე, სადაც ასახულია პროექტის და .aspx და .aspx.cs პროგრამების სახელები;



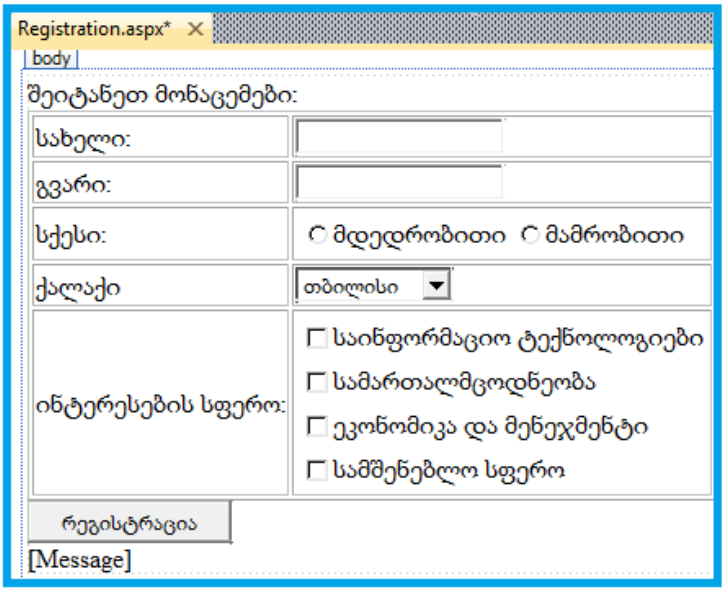
ნახ.4.4

– Registration.aspx.cs ფაილის საწყისი ტექსტის ლისტინგი ასე გამოიყურება;

```
// ----- ლისტინგი_4.1  Registration.aspx.cs.------
using System;
```

```
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
namespace WebAppRegister5
{
   public partial class Registration : Page
   {
      protected void Page_Load(object sender, EventArgs e)
      {

      }
   }
}
```

– Web-გვერდის სარეგისტრაციო ფორმის მაკეტი ნაჩვენებია 4.5 ნახაზზე.



ნახ.4.5

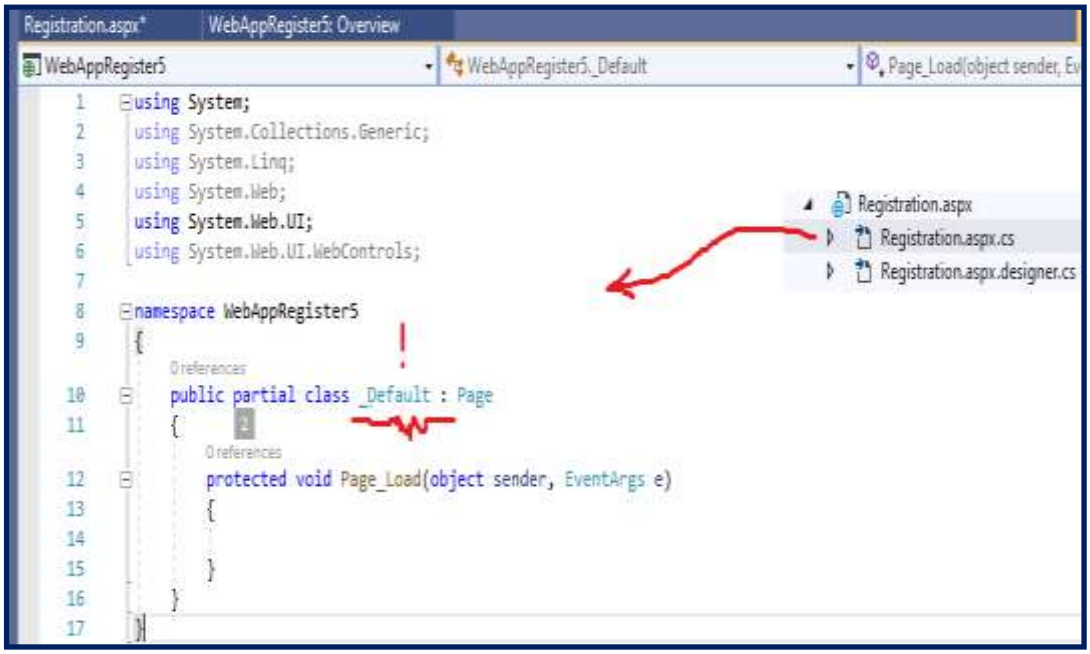
ფორმაზე მოთავსებულია სერვერული მართვის ელემენტები form, asp:TextBox, asp:DropDownList, asp:CheckBoxList, asp:Button, asp:Label და ა.შ., რომლებიც ასახულია Registration.aspx ფაილის 4.2 ლისტინგში. გავხსნათ Registration.aspx და შევიტანოთ შემდეგი კოდი (ან გამოვიყენოთ წიგნიდან მისი შესაბამისი პროტოტიპი):

```
<!— ლისტინგი_4.2 ———— –>
    <%@ Page Language="C#" AutoEventWireup="true"
      CodeBehind="Registration.aspx.cs" Inherits="WebAppRegister8.Registration" %>
    <html >
    <head>
     <title>რეგისტრაციის ფორმა</title>
      <style type="text/css">
        .auto-style1 {
           width: 253px;
        }
      </style>
    </head>
     <body>
      <form method="post" runat="server" id="registration">
       შეიტანეთ მონაცემები:
       <table border="1">
        <tr>
         <td>სახელი:</td>
         <td class="auto-style1">
          <asp:TextBox id="FirstName" runat="server"></asp:TextBox></td>
        </tr>
        <tr>
         <td>გვარი:</td>
         <td class="auto-style1">
          <asp:TextBox id="LastName" runat="server"></asp:TextBox></td>
        </tr>
        <tr>
         <td>სქესი:</td>
         <td  class="auto-style1"><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 class="auto-style1"><asp:DropDownList id="City" runat="server">
          <asp:ListItem Value="თბილისი"></asp:ListItem>
          <asp:ListItem Value="ქუთაისი"></asp:ListItem>
```

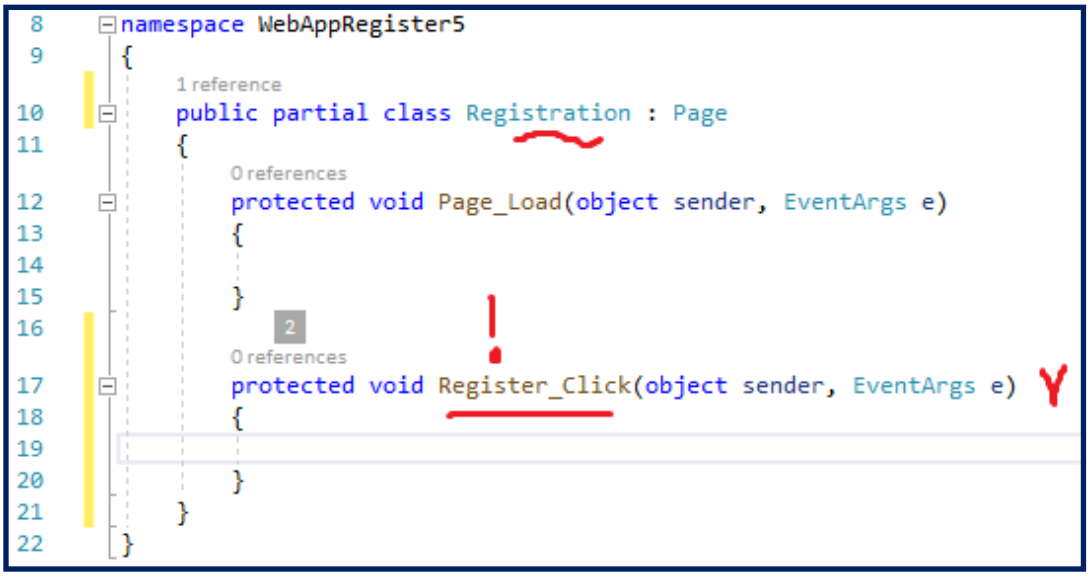
```
          <asp:ListItem Value="რუსთავი"></asp:ListItem>
          <asp:ListItem Value="გორი"></asp:ListItem>
          <asp:ListItem Value="ზათუმი"></asp:ListItem>
          <asp:ListItem Value="თელავი"></asp:ListItem>
         </asp:DropDownList></td>
        </tr>
        <tr>
            <td>ინტერესების სფერო:</td>
         <td class="auto-style1">
          <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-გვერდზე „რეგისტრაცია“ Button-ის დაკლიკვით უნდა გადავიდეთ C# კოდში... მაგრამ შეიძლება ეს ვერ მოხერხდეს... რადგანაც .aspx და .aspx.cs ფაილებს შორის კავშირი არაა გასწორებული... კერძოდ Registration.aspx.cs გადასვლით (ნახ. 4.6-ა,ბ) ჩანს რომ მე-10 სტრიქონში დარჩენილია საწყისი კლასის სახელი _Default : Page.

Web-გვერდის ჩატვირთვის და მონაცემების შევსების შემდეგ „რეგისტრაცია“ Button-ის დაჭერისას გამოიძახება OnClick მოვლენაზე მიბმული მეთოდი Register_Click. ის აღიწერება C# კოდში, რომლის 4.3 ლისტინგი მოცემულია ქვემოთ. გავხსნათ Registration.aspx.cs ფაილი და შევიტანოთ შემდეგი კოდი:

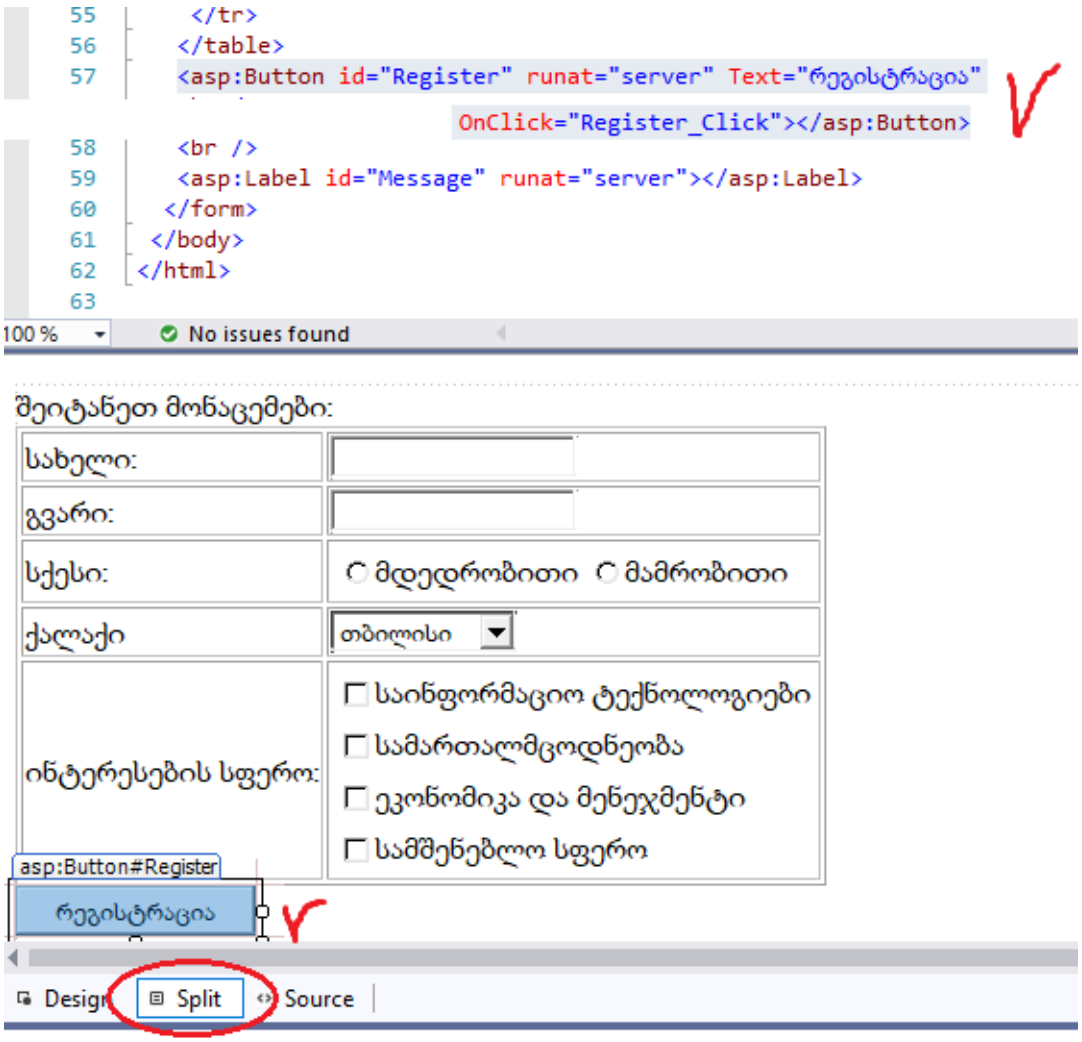


ნახ.4.6-ა. **_Default -ის შეცვლა Registration-ით**



ნახ.4.6-ბ. შეცვლის შედეგი

ამის შემდეგ Registration.aspx დიზაინის „რეგისტრაცია“ ღილაკი ამუშავდება. Split რეჟიმში ჩანს ფორმის დიზაინიც და შესაბამისი .aspx პროგრამის ტექსტის ფრაგმენტიც (ნახ.4.7). აქ გამუქებულია <asp:Button id …>.



ნახ.4.7.

– „რეგისტრაცია“ ღილაკის ამოქმედებით სისტემა გადაგვიყვანს C# -ის კოდში, Registracion.aspx.cs-ის „Register_Click“ - მოვლენაზე. აქ უნდა ჩავამატოთ ლოგიკის ტექსტი, რომელიც 4.3 ლისტინგშია აღწერილი.

```
// ── ლისტინგი_4.3 ──────────
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
```

```
namespace WebAppRegister5
{
   public partial class Registration : 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();
      }
   }
}
```

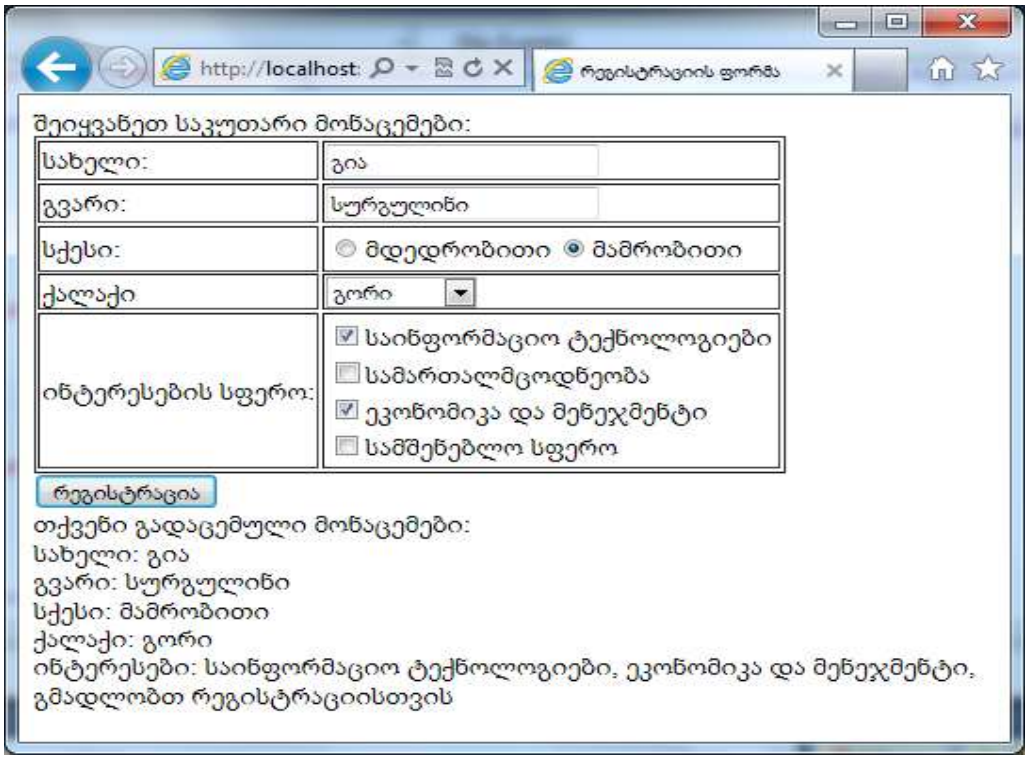
– ავამუშავოთ პროგრამა შესრულებაზე;

– Web-გვერდი, მომხმარებლის მიერ შეტანილი მონაცემებით, „რეგისტრაცია“ ღილაკზე დაკლიკვის შემდეგ, შეასრულებს C# კოდის Register_Click -ში ჩაწერილ ოპერაციებს.

სინტაქსური და სისტემური შეცდომების არარსებობის შემთხვევაში შედეგებს ექნება 4.8 ნახაზზე მოცემული სახე.

სინტაქსური შეცდომების (Errors) არსებობის დროს ისინი უნდა გავასწოროთ და შევამოწმოთ მუშაობისუნარიანობა.

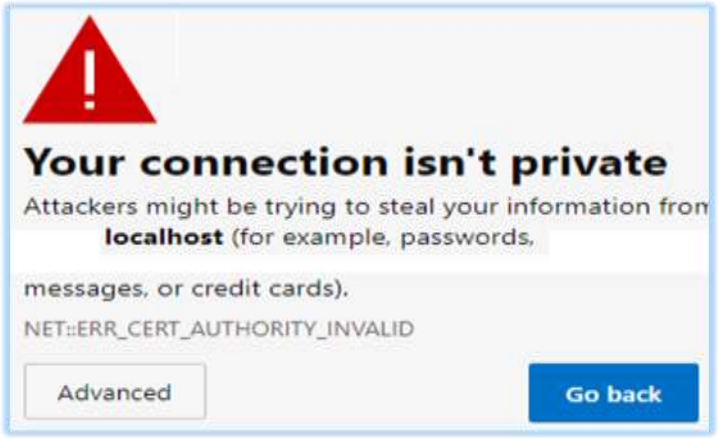
ზოგჯერ შეიძლება იყოს ლოგიკური ან სისტემური შეცდომები, რაც პროგრამის შინაარსში ან სისტემური ცდომილების გარკვევას მოითხოვს.



ნახ.4.8. სწორი შედეგით დასრულება

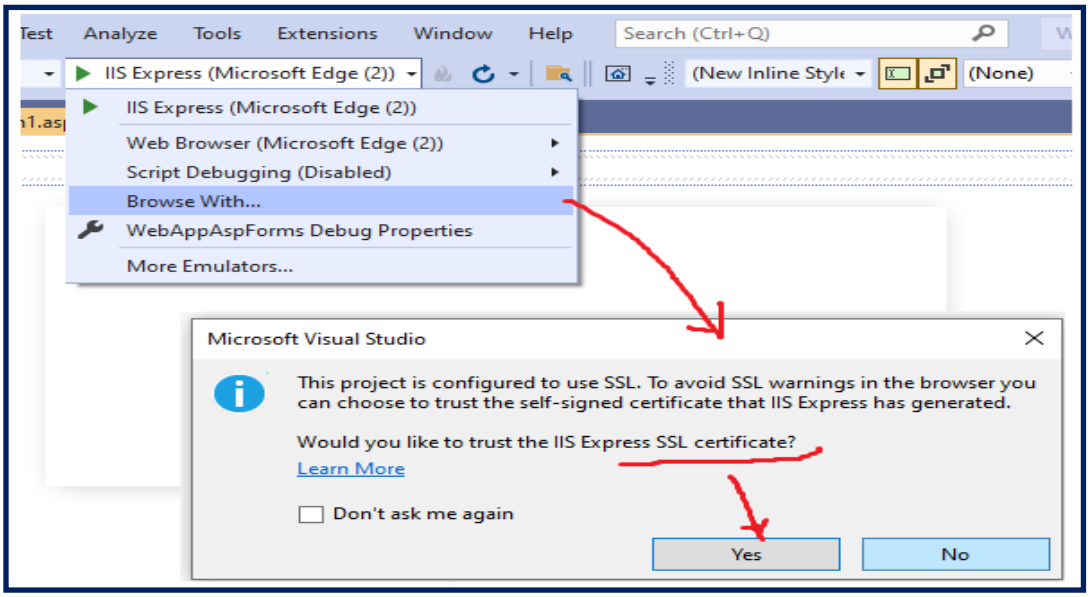
- სისტემური შეცდომის მაგალითი.

დავუშვათ, პროგრამის ამუშავების შემდეგ ვერ მივიღეთ სწორი შედეგი (ნახ.4.8). სისტემამ მოგვცა 4.9 ნახაზზე გამოტანილი შეტყობინება.



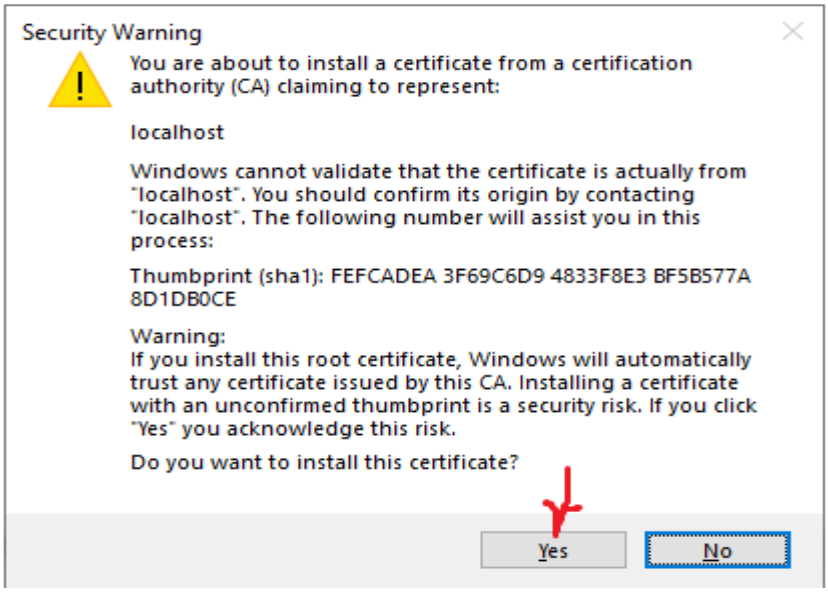
ნახ.4.9

ასეთი შეცდომის არსებობა ხშირ შემთხვევაში, დაკავშირებულია localhost-ის პრობლემასთან და კავშირშია ბრაუზერის ტიპთან. შესაძლებელია სხვა მსგავსი ვერსიაც [21]. მაგალითად, 4,10 ნახაზზე ნაჩვენებია ჩვენი მაგალითისთვის “Microsoft Edge” -ს არსებობა.



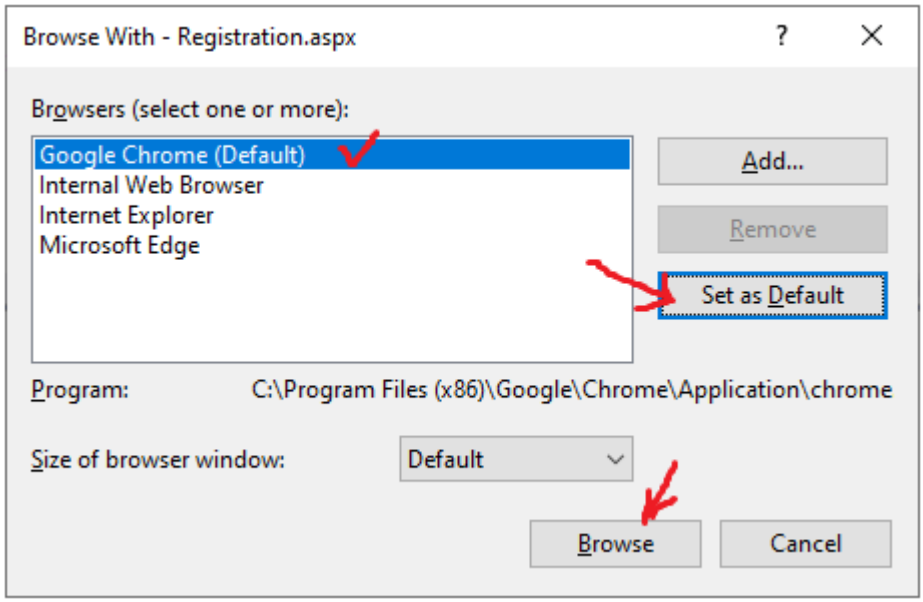
ნახ.4.10. ვირჩევთ Browse With… და Yes

გამოიტანება შეტყობინება (ნახ.4.11) - ვირჩევთ “Yes” ღილაკს.



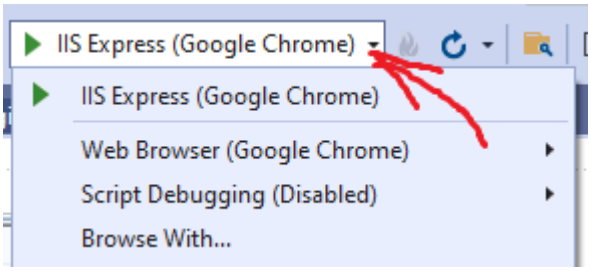
ნახ.4.11.

Browse With… ფანჯარაში ვირჩევთ Google Chrome (Default) -ს (ნახ.4.12).



ნახ.4.12. "Microsoft Edge" ბრაუზერის შეცვლა „Google Chrome" -ით

შედეგი ნაჩვენებია 4.13 ნახაზზე.



ნახ.4.13.

ამჯერად პროგრამის ამუშავებით ვღებულობთ სწორ შედეგს, „რეგისტრაციის" ღილაკი ფუნქციონირებს სწორად.

სისტემა ახორციელებს მონაცემების შეტანას, კონტროლს და გადაცემას.

4.3. ADO.NET დრაივერი და მისი DataSet ობიექტი

➢ ADO.NET (ActiveX Data Object) – მაიკროსოფთის ტექნოლოგიაა .NET Framework პლატფორმაზე, რომელიც უზრუნველყოფს მონაცემებთან წვდომას რელაციური და არარელაციური ბაზებისათვის. ეს ტექნოლოგია ოპტიმიზირებულია გამოყოფილი (disconnected) სისტემების (მონაცემთა ბაზის ნაწილის) მიწოდებისათვის კლიენტის მოთხოვნების გათვალისწინებით. ტექნოლოგია ორიენტირებულია მრავალდონიანი დანართების არქიტექტურაზე და წარმოადგენს განაწილებული სისტემების შექმნის ფაქტობრივ სტანდარტს [22].

- ADO აფართოებს მონაცემთა ბაზებს ახალი ტიპით DataSet, რომელიც უზრუნველყოფს ბაზიდან დაკავშირებული ცხრილების ლოკალური ასლის მიღებას. DataSet-ის ობიექტის დახმარებით მომხმარებელს ლოკალურად შეუძლია სხვადასხვა ოპერაციების ჩატარება ბაზის მონაცემებზე, რომლებიც ფიზიკურად გამოყოფილია ძირითად ბაზისგან. ამ ოპერაციების დასრულების შემდეგ ცვლილებები შესაძლებელია აისახოს ბაზაში შესაბამისი "მონაცემთა ადაპტერის" (data adapter) საშუალებით;
- ADO.NET-ში რეალიზებულია XML ფორმატში მონაცემების ასახვის მხარდაჭერა და მათი გაცვლა სერვერთან;
- ADO.NET არის მართვადი კოდის ბიბლიოთეკა და მასთან ურთიერთობა ხორციელდება ჩვეულებრივი .NET-ნაკრებით. ADO.NET ტიპები იყენებს მეხსიერების CLR შესაძლებლობებს.

ADO.NET-ის ყველა ტიპი გამოიყენება ერთიანი ამოცანების შესასრულებლად:

- დაამყაროს კავშირი მონაცემთა საცავთან;
- შექმნას DataSet ობიექტი და შეავსოს მონაცემებით;
- გამოყოფილ იქნას მონაცემთა საცავიდან და DataSet ობიექტში განხორციელებული ცვლილებები დააბრუნოს ბაზაში.

➢ **DataSet ობიექტი** – მონაცემთა ტიპია, რომელიც ასახავს ცხრილების ლოკალურ ერთობლიობას და ინფორმაციას მათი კავშირების შესახებ.

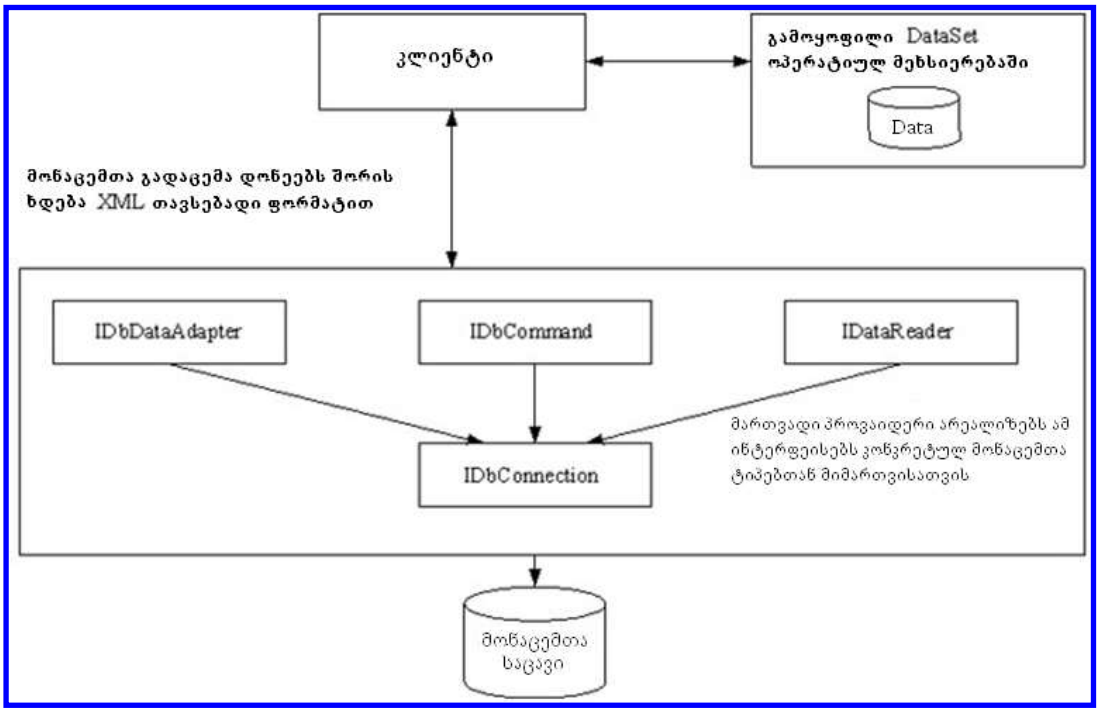
პრაქტიკულად, შესაძლებელია კლიენტის მხარეს შეიქმნას DataSet ობიექტი, რომელიც იქნება გამოყოფილი, მონაცემთა ბაზის სრული ასლი.

DataSet ობიექტის შექმნისა და მისი მონაცემებით შევსების შემდეგ შესაძლებელია პროგრამული საშუალებებით მასთან მოთხოვნების შესრულება, ცხრილებში მოძრაობა, ოპერაციების შესრულება ისე, როგორც რეალურ მონაცემთა ბაზასთან: ცხრილში ახალი ჩანაწერების ჩამატება, არსებულის წაშლა

ან შეცვლა, ფილტრების გამოყენება და ა.შ. როცა კლიენტი დაასრულებს მონაცემთა ცვლილებებს, ინფორმაცია გადასამუშავებლად გადაეცემა მონაცემთა საცავს.

DataSet-ის შექმნა ხორციელდება მართვადი პროვაიდერის (managed provider) დახმარებით. ესაა *კლასების ერთობლიობა*, რომლითაც რეალიზდება *ინტერფეისები*, განსაზღვრული System.Data სახელსივრცეში (namespace).

საქმე ეხება ინტერფეისებს: IDbCommand, IDbDataAdapter, IDbConnection და IdataReader (ნახ.4.14).



ნახ.4.14. კლიენტის ურთიერთქმედება მართვად პროვაიდერებთან

ADO.NET-ში გამოიყენება ორი მართვადი პროვაიდერი: SQL და OleDb. პირველი Microsoft SQL Server -თან სამუშაოდ. მონაცემთა სხვა წყაროებისთვის გამოიყენება OleDb პროვაიდერი (მაგალითად, Ms Access-თან).

System.Data სახელსივრცის ტიპები გამოიყენება მონაცემების ასახვისათვის, რომლებიც მიიღება მონაცემთა წყაროდან (მაგრამ არა კავშირის დასამყარებლად ამ წყაროსთან).

ძირითადად ეს ტიპები გამოიყენება მონაცემთა ბაზასთან სამუშაოდ: ცხრილებთან, სტრიქონებთან, სვეტებთან, შეზღუდვებთან და ა.შ. (ცხრ.4.1).

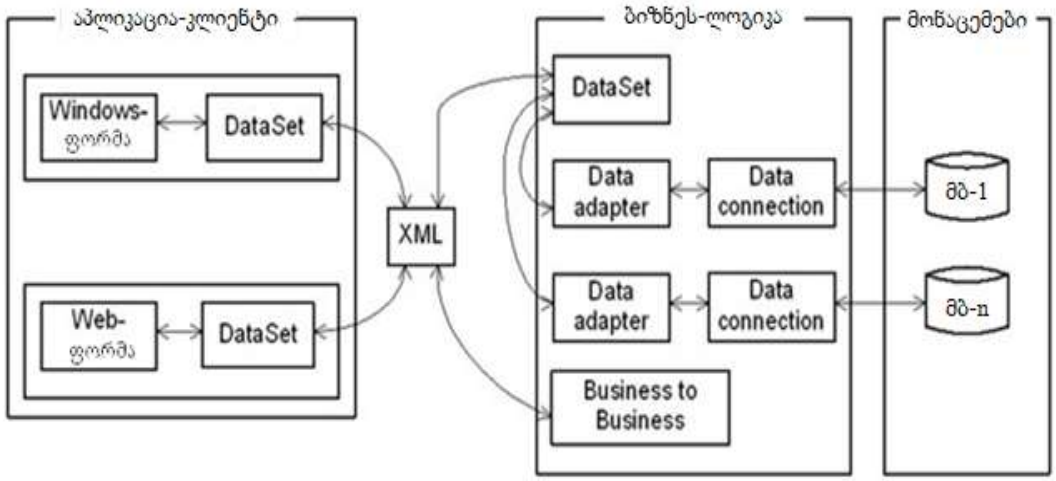
System.Data სახელსივრცის ტიპები	
ტიპი	დანიშნულება
DataColumn DataColumnCollection	– ერთი სვეტია DataTable ობიექტში, – ყველა სვეტი
Constraint ConstraintCollection	– ობიექტ-ორიენტირებული გარსი შეზღუდვაზე (მაგ., გარე გასაღებზე ან უნიკალურობაზე), დადებული ერთ ან რამდენიმე DataColumn -ზე, – ყველა შეზღუდვა DataTable ობიექტში
Data Row DataRowCollection	– ასახავს ერთადერთ სტრიქონს DataTable =-ში, – ყველა სტრიქონი DataTable -ში
DataRowView Data View	– ქმნის წარმოდგენას (view) ერთი სტრიქონით. – პროგრამულად შექმნილი წარმოდგენა DataTable ობიექტისთვის, რომელიც შეიძლება გამოყენებულ იქნას მოწესრიგების, ფილტრაციის, ძებნის, რედაქტირების და სხვ.
DataSet	– ობიექტია, რომელიც იქმნება კლიენტის ოპერატიულ მეხსიერებაში. DataSet შედგება DataTable ობიექტების სიმრავლისა და ინფორმაციისგან მათი ურთიერთკავშირების შესახებ.
ForeignKeyConstraint UniqueConstraint	– ასახავს შეზღუდვას ცხრილის სვეტებზე, პირველადი და გარე გასაღებების დამოკიდებულებაზე. – უნიკალურობის შეზღუდვა
DataRelationCollection DataRelation DataTableCollection DataTable	– ასახავს დამოკიდებულებათა ყველა ტიპს (ანუ DataRelation ობიექტებს) ცხრილებს შორის DataSet-ში. – ასახავს ყველა ცრილის ერთობლიობას (DataTable ობიექტები) DataSet ში.

ADO.NET-ში გამოიყენება მომხმარებლის მუშაობა მონაცემთა ბაზასთან, რომელიც ფიზიკურად გამოყოფილია მონაცემთა წყაროსგან.

აპლიკაცია მიუერთდება მონაცემთა ბაზას მხოლოდ მცირე დროის გამნავლობაში, გადმოიწერს შერჩეულ მონაცემებს და გაწყვეტს კავშირს.

ამით თავისუფლდება სერვერის რესურსები სხვა მომხმარებლისთვის.

მონაცემებთან მიმართვის აღნიშნული მოდელი ნაჩვენებია 4.15 ნახაზზე.



ნახ.4.15. მონაცემებთან მიმართვის მოდელი ADO.NET -ში

- ADO.NET ობიექტურ მოდელში შეიძლება გამოვყოთ რამდენიმე დონე:

• *მონაცემთა დონე:* ესაა საბაზო დონე, სადაც ფიზიკურად განთავსებულია მონაცემები (მაგალითად, Ms SQL Server ცხრილები);

• *ბიზნეს-ლოგიკის დონე:* ესაა ობიექტების ერთობლიობა, რომელიც განსაზღვრავს, თუ რომელ ბაზასთანაა კავშირი დასამყარებელი და რა ქმედებებია ჩასატარებელი. ბაზებთან კავშირის დასამყარებლად გამოიყენება DataConnection ობიექტი. მონაცემებზე ქმედებათა ბრძანებების შესანახად გამოიყენება DataAdapter ობიექტი. და ბოლოს, თუ განხორციელდა მონაცემთა ამორჩევა ბაზიდან, მაშინ მათ შესანახად გამოიყენება DataSet ობიექტი;

• *აპლიკაციის დონე:* ესაა ობიექტების ერთობლიობა, რომელიც უზრუნველყოფს მონაცემთა შენახვას და ასახვას მომხმარებლის კომპიუტერზე.

მონაცემთა შესანახად გამოიყენება DataSet ობიექტი, ხოლო ასახვისათვის - მართვის ელემენტების ერთობლიობა: DataGrid, TextBox, ComboBox, Label და ა.შ. Visual Studio .Net -ში შეიძლება ორი ტიპის აპლიკაციის აგება: Windows- (exe-ფაილების სახით), და Web-აპლიკაცია, რომელიც მუშაობს ბრაუზერის გარსში. ორივე შემთხვევაში გამოიყენება DataSet ობიექტი.

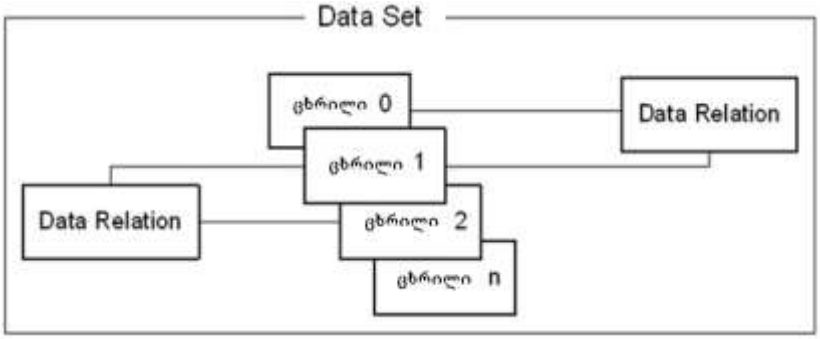
მონაცემების გაცვლა აპლიკაციებსა და ბიზნეს-ლოგიკის დონეს შორის ხდება XML ფორმატით, მონაცემთა გადაცემის გარემო კი შეიძლება იყოს ინტრანეტი ან ინტერნეტი.

ADO.NET-ში მონაცემების მანიპულირება ხდება SQL-მოთხოვნებით ან შენახვადი პროცედურებით (DataCommand).

მონაცემთა ბაზიდან ჩანაწერების ამოსაღბად საჭიროა შემდეგი პროცედურების ჩატარება:

- მბ-თან კავშირის გახსნა (შეერთება -connection);
- მეთოდის ან ბრძანების ან შენახვადი პროცედურის გამოძახება;
- მბ-თან კავშირის დახურვა.

ამგვარად, შეიძლება ვთქვათ, რომ DataSet არის თავისებური კეშ-მეხსიერება მონაცემთა ბაზიდან ამოღებული ჩანაწერებისთვის. იგი შედგება ერთი ან მეტი ცხრილის ასლისგან. გარდა ამისა აქ იქნება ცხრილებსშორისი კავშირებიც. DataSet ობიექტის სტრუქტურა ნაჩვენებია 4.16 ნახაზზე.



ნახ. 4.16. DataSet ობიექტის სტრუქტურა

DataSet – მონაცემების პასიური კონტეინერია, რომელიც ახორციელებს მხოლოდ მათ შენახვას.

რა უნდა მოთავსდეს ამ კონტეინერში, ამას განსაზღვრავს სხვა ობიექტი, როგორიცაა DataAdapter. მონაცემთა ადაპტერი შეიცავს ძირითადად ოთხ ბრძანებას: SELECT, INSERT, UPDATE, DELETE, ჩანაწერების ამოსარჩევად, დასამატებლად, შესაცვლელად და წასაშლელად.

მაგალითად, DataAdapter ობიექტის Fill-მეთოდი, რომელიც ავსებს DataSet კონტეინერს მონაცემებით, შეიძლება გამოიყენოს SelectCommand -ში შემდეგი მოთხოვნა:

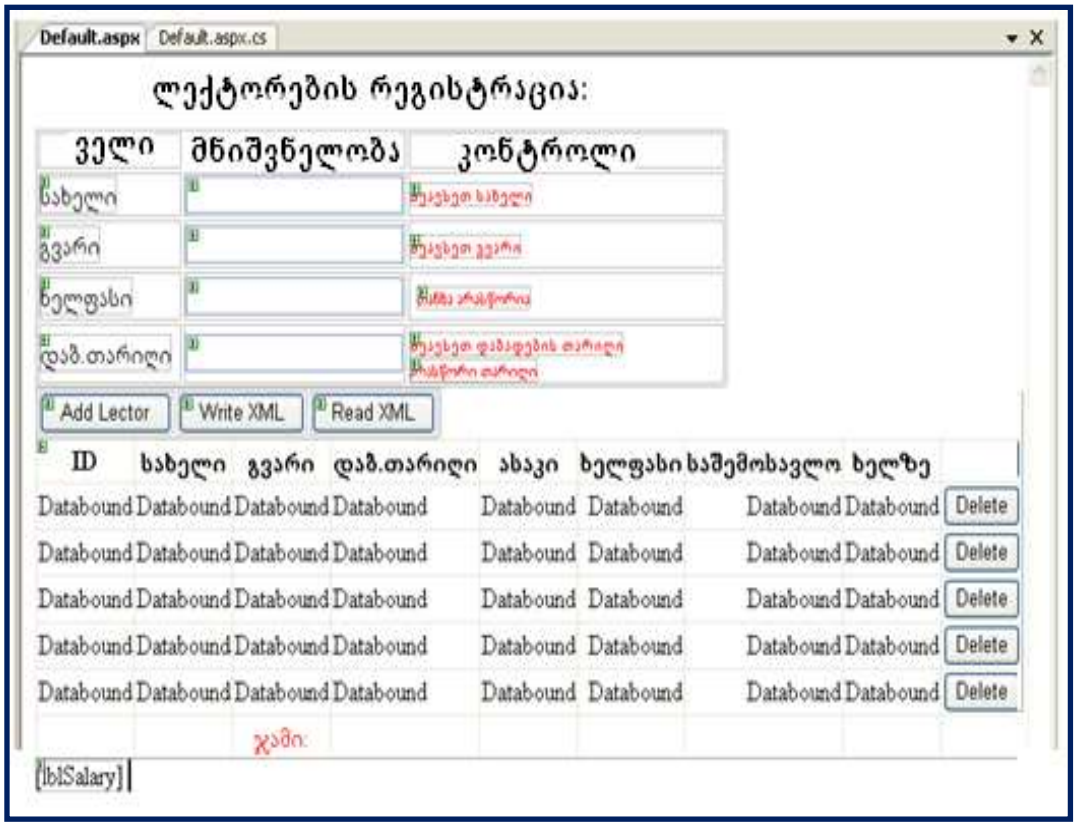
SELECT au_id, au_lname, au_fname FROM authors

DataSet მონაცემები – „დამოუკიდებელი" ასლია მონაცემთა ბაზის ფრაგმენტის, რომელიც მოთავსებულია კლიენტის კომპიუტერზე. მასში არაა ასახული რეალური ბაზის ბოლო ცვლილებები, რომლებიც სხვა მომხმარებლებმა გააკეთეს.

4.4. ASP.NET Web პროექტის აგება GridView –ს გამოყენებით და შედეგების XML ფაილში ჩაწერა

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

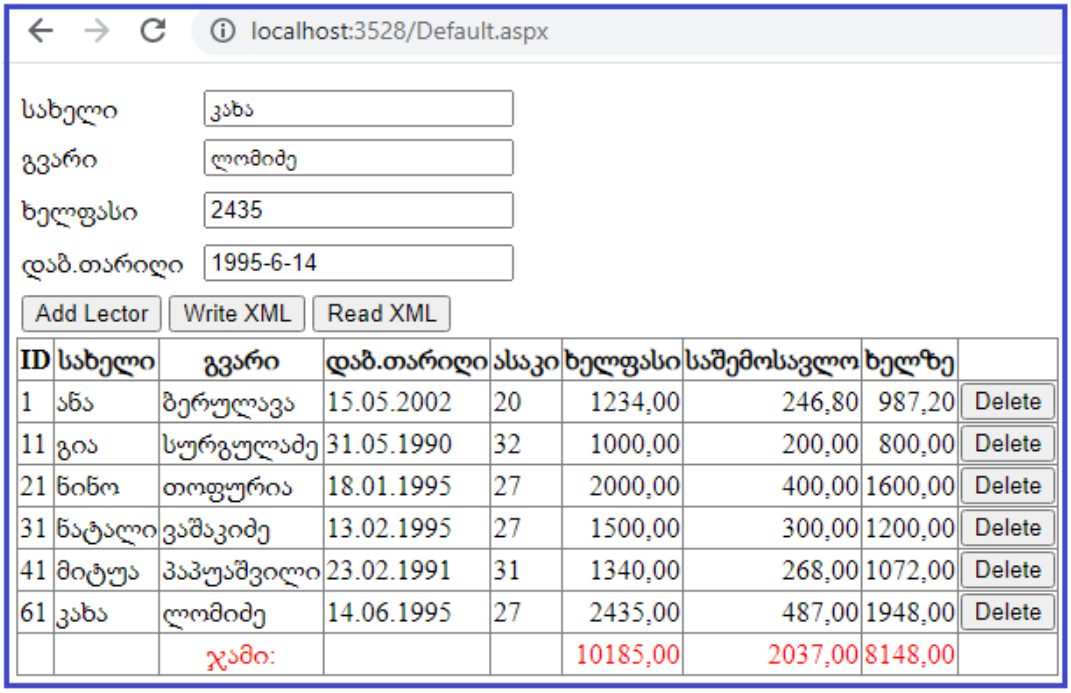
განვიხილოთ ასეთი ამოცანა: Visual Studio .NET გარემოში ავაგოთ ***Web-პროექტი*** ლექტორთა სარეგისტრაციო მონაცემების შესატანად. განვახორციელოთ მონაცემთა ვიზუალური და ავტომატური კონტროლის საშუალებების გამოყენება. Add Lector-ღილაკით შეტანილი მონაცემები აისახოს GridView ცხრილში უნიკალური ID-ს მქონე სტრიქონის სახით, მომზადდეს სარეგისტრაციო ცხრილი ახალი ინფორმაციის შესატანად (ნახ.4.17). ავტომატური კონტროლისთვის გამოიყენება ToolBox->Validation.



ნახ.4.17

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

4.18 ნახაზზე ნაჩვენებია Web-გვერდის მაკეტი ბრაუზერში (პირობითი მნიშვნელობებით).



ID	სახელი	გვარი	დაბ.თარიღი	ასაკი	ხელფასი	საშემოსავლო	ხელზე	
1	ანა	ბერულავა	15.05.2002	20	1234,00	246,80	987,20	Delete
11	გია	სურგულაძე	31.05.1990	32	1000,00	200,00	800,00	Delete
21	ნინო	თოფურია	18.01.1995	27	2000,00	400,00	1600,00	Delete
31	ნატალი	ვაშაკიძე	13.02.1995	27	1500,00	300,00	1200,00	Delete
41	მიტუა	პაპუაშვილი	23.02.1991	31	1340,00	268,00	1072,00	Delete
61	კახა	ლომიძე	14.06.1995	27	2435,00	487,00	1948,00	Delete
		ჯამი:			10185,00	2037,00	8148,00	

ნახ.4.18

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

```
// ── ლისტინგი_4.4── 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 ღილაკის კოდები: Write_XML (სტრიქონების შესანახად XML ფაილში) და Read_XML (სტრიქონების ამოსაღებად XML ფაილიდან).

```
// ── ლისტინგი_4.5 ──  Write_XML ----------------
protected void Button2_Click(object sender, EventArgs e)
    {
        DataSet ds = Session["MyDataSet"] as DataSet;
        ds.WriteXml(Request.PhysicalApplicationPath + "\\lectors.xml");
        //Response.Redirect("~/lectors.xml");
    }

// ── ლისტინგი_4.6 ──  Read_XML ---------------
protected void Button3_Click(object sender, EventArgs e)
    {
```

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

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

```
// —— ლისტინგი_4.7 —— DataSet ——
private DataSet GetDataSet()
  {
    DataTable lectors = new DataTable("Lectors");
    //Add the DataColumn using all properties
    DataColumn id = new DataColumn("ID");
    id.DataType = typeof(int);
    id.Unique = true;
    id.AutoIncrement = true;
    id.AutoIncrementSeed = 1;
    id.AutoIncrementStep = 10;
    id.AllowDBNull = false;
    id.Caption = "ID";
    lectors.Columns.Add(id);

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

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

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

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

DataColumn age = new DataColumn("Age", typeof(DateTime));
age.ColumnMapping = MappingType.Hidden;
age.Expression = "BirthDate";
lectors.Columns.Add(age);

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

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

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

```
    DataSet ds = new DataSet();
    ds.Tables.Add(lectors);
    return ds;
}
```

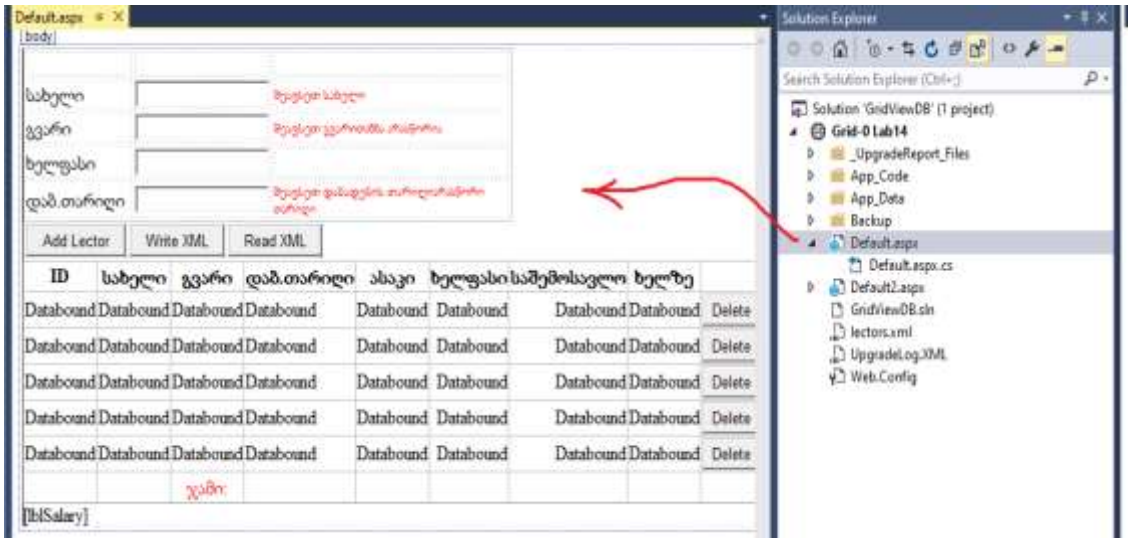
- XML ფაილს, მასში სტრიქონების (ობიექტების) ჩაწერის შემდეგ ექნება ასეთი სახე:

```
<!—— ლისტინგი_4.8 — XML ჩანაწერების შენახვის სტრუქტურა –>
<?xml version="1.0" standalone="yes"?>
<NewDataSet>
  <Lectors>
    <ID>1</ID>
    <FirstName>ანა</FirstName>
    <LastName>ბერულავა</LastName>
    <Salary>1234</Salary>
    <BirthDate>2002-05-15T00:00:00+04:00</BirthDate>
  </Lectors>
  <Lectors>
    <ID>11</ID>
    <FirstName>გია</FirstName>
    <LastName>სურგულაძე</LastName>
    <Salary>1000</Salary>
    <BirthDate>1990-05-31T00:00:00+04:00</BirthDate>
  </Lectors>
  <Lectors>
    <ID>21</ID>
    <FirstName>ნინო</FirstName>
    <LastName>თოფურია</LastName>
    <Salary>2000</Salary>
    <BirthDate>1995-01-18T00:00:00+04:00</BirthDate>
  </Lectors>
  <Lectors>
    <ID>31</ID>
    <FirstName>ნატალი</FirstName>
    <LastName>ვამაკიძე</LastName>
    <Salary>1500</Salary>
    <BirthDate>1995-02-13T00:00:00+04:00</BirthDate>
  </Lectors>
  <Lectors>
    <ID>41</ID>
    <FirstName>მიტუა</FirstName>
    <LastName>პაპუაშვილი</LastName>
    <Salary>1340</Salary>
    <BirthDate>1991-02-23T00:00:00+04:00</BirthDate>
  </Lectors>
  <Lectors>
    <ID>61</ID>
    <FirstName>კახა</FirstName>
```

```
    <LastName>ლომიძე</LastName>
    <Salary>2435</Salary>
    <BirthDate>1995-06-14T00:00:00+04:00</BirthDate>
  </Lectors>
 </NewDataSet>
```

პროგრამის ამუშავების შემდეგ, როდესაც მასში აღარ იქნება სინტაქსური და ლოგიკური შეცდომები, მივიღებთ შედეგებს, რომლებსაც ამ პარაგრაფის ბოლოში წარმოვადგენთ.

ახლა კი ვაჩვენოთ პარაგრაფის ძირითადი კომპონენტები (დიზაინისა და ლოგიკის დონეზე). 4.16 ნახაზზე მოცემულია Web პროექტის ფორმა (დიზაინი) მისი შესაბამისი Default.aspx და Default.aspx.cs –ის ლისტინგები 4.9 -4.10.



ნახ.4.16. Web აპლიკაციის ინტერფეისი Visual Studio.NET გარემოში

```
<!--- ლისტინგი 4.9. --- Dafault.aspx – Source - კოდი ------>
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default.aspx.cs"
Inherits="_Default" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
   <title>Untitled Page</title>
</head>
<body>
```

```
<form id="form1" runat="server">
    <div>
        <table style="position: relative">
            <tr>
                <td style="width: 100px">
                </td>
                <td style="width: 100px">
                </td>
                <td style="width: 223px">
                </td>
            </tr>
            <tr>
                <td style="width: 100px">
                    <asp:Label ID="Label1" runat="server" Style="position: relative"
Text="სახელი"></asp:Label></td>
                <td style="width: 100px">
                    <asp:TextBox ID="txtFirstName" runat="server" Style="position:
relative"></asp:TextBox></td>
                <td style="width: 223px">
                    <asp:RequiredFieldValidator ID="RequiredFieldValidator1" runat="server"
ControlToValidate="txtFirstName"
                        Display="Dynamic" ErrorMessage="შეავსეთ სახელი" Font-Size="8pt"
Style="position: relative"></asp:RequiredFieldValidator></td>
            </tr>
            <tr>
                <td style="width: 100px; height: 26px">
                    <asp:Label ID="Label2" runat="server" Style="position: relative"
Text="გვარი"></asp:Label></td>
                <td style="width: 100px; height: 26px">
                    <asp:TextBox ID="txtLastName" runat="server" Style="position:
relative"></asp:TextBox></td>
                <td style="width: 223px; height: 26px">
                    <asp:RequiredFieldValidator ID="RequiredFieldValidator2" runat="server"
ControlToValidate="txtLastName"
                        ErrorMessage="შეავსეთ გვარი" Font-Size="8pt" Style="position:
relative"></asp:RequiredFieldValidator>
                    <asp:RangeValidator ID="RangeValidator1" runat="server"
ControlToValidate="txtSalary"
```

```
                Display="Dynamic" ErrorMessage="თანხა არასწორია"
MaximumValue="10000" MinimumValue="0"
                Style="left: -78px; position: relative; top: 30px" Type="Currency" Font-
Italic="False"
                Font-Size="X-Small"></asp:RangeValidator></td>
            </tr>
            <tr>
              <td style="width: 100px; height: 26px">
                <asp:Label ID="Label3" runat="server" Style="position: relative"
Text="ხელფასი"></asp:Label></td>
              <td style="width: 100px; height: 26px">
                <asp:TextBox ID="txtSalary" runat="server" Style="position:
relative"></asp:TextBox></td>
              <td style="width: 223px; height: 26px; text-align: left;">
              </td>
            </tr>
            <tr>
              <td style="width: 100px; height: 26px">
                <asp:Label ID="Label4" runat="server" Style="position: relative"
Text="დაბ.თარიღი"></asp:Label></td>
              <td style="width: 100px; height: 26px">
                <asp:TextBox ID="txtBirthDate" runat="server" Style="position:
relative"></asp:TextBox></td>
              <td style="width: 223px; height: 26px">
                <asp:RequiredFieldValidator ID="RequiredFieldValidator3" runat="server"
ControlToValidate="txtBirthDate"
                Display="Dynamic" ErrorMessage="შეავსეთ დაბადების თარიღი" Font-
Size="8pt" Style="position: relative"></asp:RequiredFieldValidator>
                <asp:RangeValidator ID="RangeValidator2" runat="server"
ControlToValidate="txtBirthDate"
                Display="Dynamic" ErrorMessage="არასწორი თარიღი" Font-Size="X-Small"
MaximumValue="01/01/2079"
                MinimumValue="01/01/1900" Style="position: relative"
Type="Date"></asp:RangeValidator></td>
            </tr>
          </table>
        </div>
        <table>
```

```
        <tr>
         <td>
          <asp:Button ID="Button1" runat="server" Text="Add Lector" OnClick="Button1_Click" />
         </td>
        <td>
          <asp:Button ID="Button2" runat="server" OnClick="Button2_Click" Text="Write XML"
CausesValidation="False" />
        </td>
         <td>
          <asp:Button ID="Button3" runat="server" OnClick="Button3_Click" Text="Read XML"
CausesValidation="False" />
            </td>
         </tr>
      </table>
      <asp:GridView ID="GridView1" runat="server" Style="position: relative; top: 0px"
        AutoGenerateColumns="False" ShowFooter="True" FooterStyle-ForeColor="red"
OnRowDataBound="GridView1_RowDataBound"
         OnRowDeleting="GridView1_RowDeleting">
         <Columns>
           <asp:BoundField DataField="ID" HeaderText="ID" />
           <asp:BoundField DataField="FirstName" HeaderText="სახელი" />
           <asp:BoundField DataField="LastName" HeaderText="გვარი" FooterText="ჯამი:">
             <FooterStyle HorizontalAlign="Center" />
           </asp:BoundField>
        <asp:BoundField DataField="BirthDate" HeaderText="დაბ.თარიღი" HtmlEncode="False"
             DataFormatString="{0:dd/MM/yyyy}" />
        <asp:BoundField DataField="Age" HeaderText="ასაკი" HtmlEncode="False"
DataFormatString="{0:dd/MM/yyyy}" />
        <asp:BoundField DataField="Salary" HeaderText="ხელფასი" HtmlEncode="False"
DataFormatString="{0:F2}">
             <ItemStyle HorizontalAlign="Right" /> <FooterStyle HorizontalAlign="Right" />
        </asp:BoundField>
        <asp:BoundField DataField="Tax" HeaderText="საშემოსავლო" HtmlEncode="False"
DataFormatString="{0:F2}">
             <ItemStyle HorizontalAlign="Right" /> <FooterStyle HorizontalAlign="Right" />
        </asp:BoundField>
        <asp:BoundField DataField="NettoSalary" HeaderText="ხელზე" HtmlEncode="False"
DataFormatString="{0:F2}">
```

```
            <ItemStyle HorizontalAlign="Right" /> <FooterStyle HorizontalAlign="Right" />
        </asp:BoundField>
            <asp:ButtonField ButtonType="Button" CommandName="Delete" Text="Delete" />
        </Columns>
        <FooterStyle ForeColor="Red" />
    </asp:GridView>
    <asp:Label runat="server" ID="lblSalary"></asp:Label>
  </form>
</body>
</html>


// - - - ლისტინგი 4.10. --- Dafault.aspx.cs - - - - - - -
using System;
using System.Data;
using System.Configuration;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;

public partial class _Default : System.Web.UI.Page
{
  protected void Page_Load(object sender, EventArgs e)
  {

    if (!IsPostBack)
    {
      DataSet lectors = GetDataSet();
      Session["MyDataSet"] = lectors;
    }
    DataSet dsLectors = Session["MyDataSet"] as DataSet;
    GridView1.DataSource = dsLectors;
    GridView1.DataBind();
  }

  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;

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

```
    }
    protected void GridView1_RowDataBound(object sender, GridViewRowEventArgs e)
    {
        if (e.Row.RowType == DataControlRowType.DataRow)
        {
            DateTime dt = (DateTime)DataBinder.Eval(e.Row.DataItem, "BirthDate");
            int years = DateTime.Now.Year - dt.Year;
            e.Row.Cells[4].Text = years.ToString();

            if (years>65) e.Row.BackColor = System.Drawing.Color.Gray;
        }
        //e.Row.Cells[4].Text = ((DateTime)e.Row.DataItem).ToShortDateString();
    }

    protected void GridView1_RowDeleting(object sender, GridViewDeleteEventArgs e)
    {
        DataSet ds = GridView1.DataSource as DataSet;
        DataTable dtLectors = ds.Tables["Lectors"];
        dtLectors.Rows[e.RowIndex].Delete();

        //GridView1.DataSource = ds;
        GridView1.DataBind();
    }
}
```

V თავი
კორპორაციული ინტრანეტ პორტალის აგება Office-365 ბაზაზე

აპლიკაციის დასაპროექტებლად საჭიროა გვქონდეს front end და back end. front end შეიძლება იყოს ვებ-საიტი ან მობილური აპლიკაცია, ხოლო backend-ი Azure SQL.

დააპროექტეთ კორპორაციული ვებ-პორტალი Sharepoint Online-ის საშუალებით, საჭიროა Office365-ის ექაუნთი. ნებისმიერ ვებ-ბრაუზერში აკრიფეთ portal.office.com და შეიტანეთ თქვენი მონაცემები.

შექმენით სხვადასხვა ვებ-გვერდი, ორგანიზაციული სტრუქტურის გათვალისწინებით.

➢ *კორპორაციული ვებ-პორტალის დასაპროექტებლად აუცილებელია წინასწარ მკაფიოდ იყოს განსაზღვრული ვირტუალური ორგანიზაციის საქმიანობის მიზანი. კერძოდ:*

- ვინ არიან საიტის მომხმარებლები ?
- რა არის მათი მიზნები ?
- როგორ ასრულებენ მომხმარებლები ყოველდღიურ სამუშაოს ?
- რომელ აპლიკაციებს იყენებენ ისინი ?
- რომელი სამუშაო პროცესების ავტომატიზაციაა აუცილებელი ?
- რა სახის სიების შექმნაა აუცილებელი ?
- რომელი დოკუმენტების ატვირთვაა აუცილებელი დოკ-ბიბლიოთეკაში ?
- რა ინფორმაცია უნდა განთავსდეს მთავარ გვერდზე ?
- რა ინფორმაცია უნდა განთავსდეს დანარჩენ გვერდებზე ?
- რომელი გვერდების, სიების და აპებისთვის შეიქმნას ნავიგაციის ლინკები და სად ?

➢ *დასრულებული პროექტი (წლის ბოლოს) შეიცავს:*

1. *მთავარს:* ვიზუალურ საწყის გვერდს;
2. *საკომუნიკაციო საიტს*: ახალი ამბებისა და ინფორმაციის გაზიარება შაბლონებისა და ვებ ელემენტების მეშვეობით;
3. *სიახლეებს:* ვებ-ელემენტი ისტორიების, განცხადებების, სიახლეებისა და განახლებების გამოსაქვეყნებლად;
4. *სამუაო პროცესების ავტომატიზაციას;*
5. *ნავიგაციის პანელს;*
6. მობილურ აპლიკაციას.

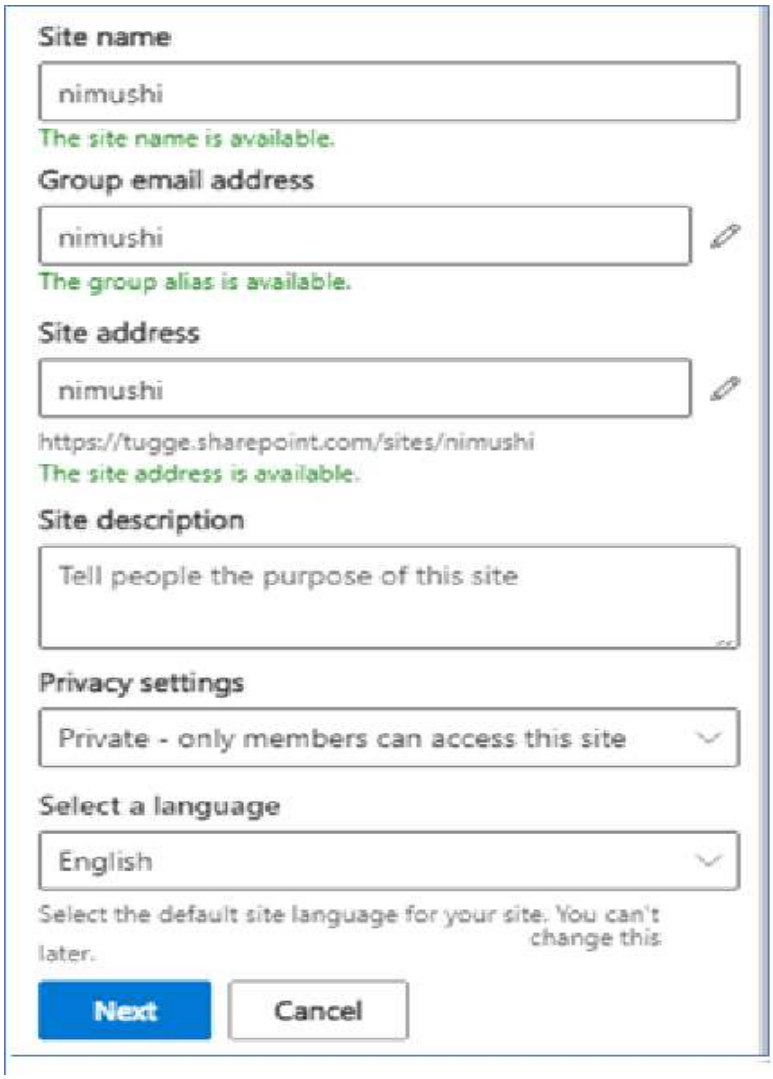
➢ *სამუშაოს მსვლელობა:*

ნებისმიერ ბრაუზერში აკრიფეთ portal.office.com, გაიარეთ ავტორიზაცია, აირჩიეთ sharepoint.

5.1. Web-საიტის დაპროექტება Sharepoint Online-ის ბაზაზე

➢ *საიტის შექმნა*

შევქმნათ ვებსაიტი ბრძანებით: Create site → Team site, დავარქვათ სახელი. მივუთითოთ ჯგუფის წევრების ექაუნთები (tug.ge) – ნახ.5.1.

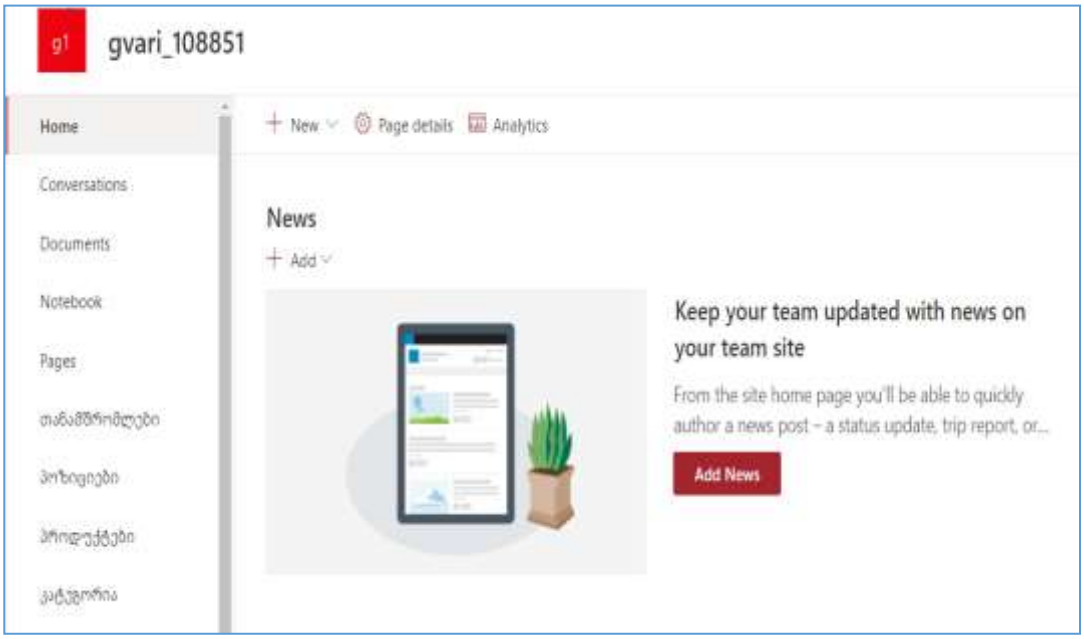


ნახ.5.1

შეგიძლიათ პირველ ეტაპზე შეუვსებელი დატოვოთ ეს ველები, და ჯგუფის წევრები ჩაამატოთ მოგვიანებით. აირჩიეთ ღილაკი Finish (ნახ.5.2).

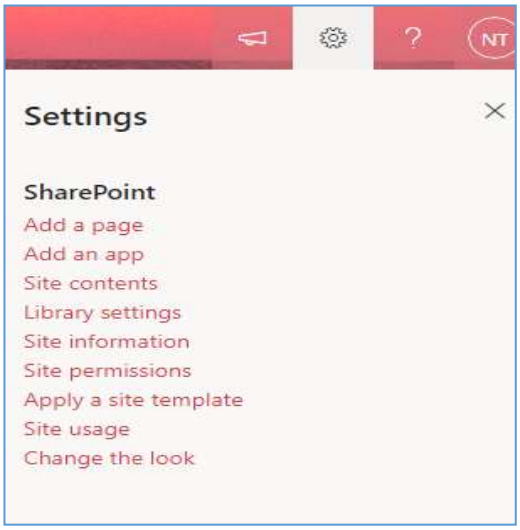
ნახ.5.2

საიტის Home Page მოცემულია 5.3 ნახაზზე.



ნახ. 5.3.

ავირჩიოთ დიზაინი, ბრძანებით Change the look, სადაც შეიძლება თემის და თავსართის შეცვლა (ნახ.5.4.).



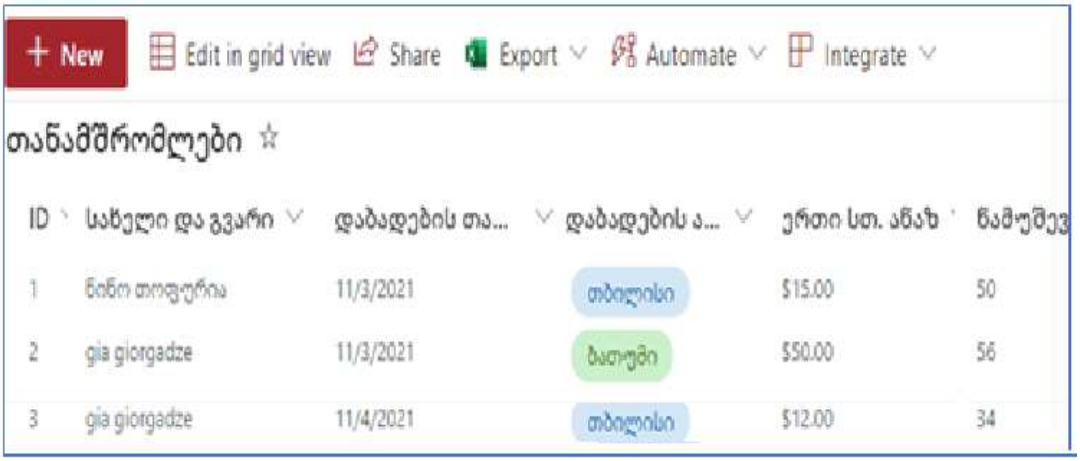
ნახ.5.4

5.2. სიებისა და ველების შექმნა

ავირჩიოთ ბრძანება

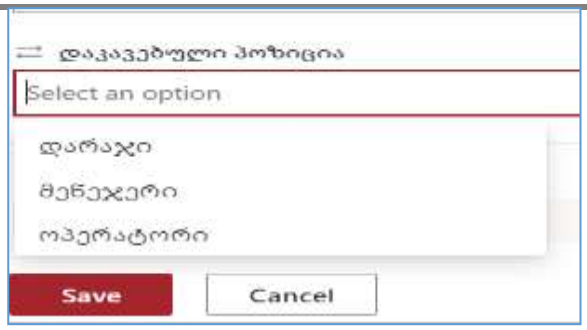
Home→ New→List

შევქმნათ თანამშრომლების სია (იგი მოგვაგონებს მონაცემთა ბაზის ცხრილს). მისი ფრაგმენტი მოცემულია 5.5 ნახაზზე.



ნახ.5.5

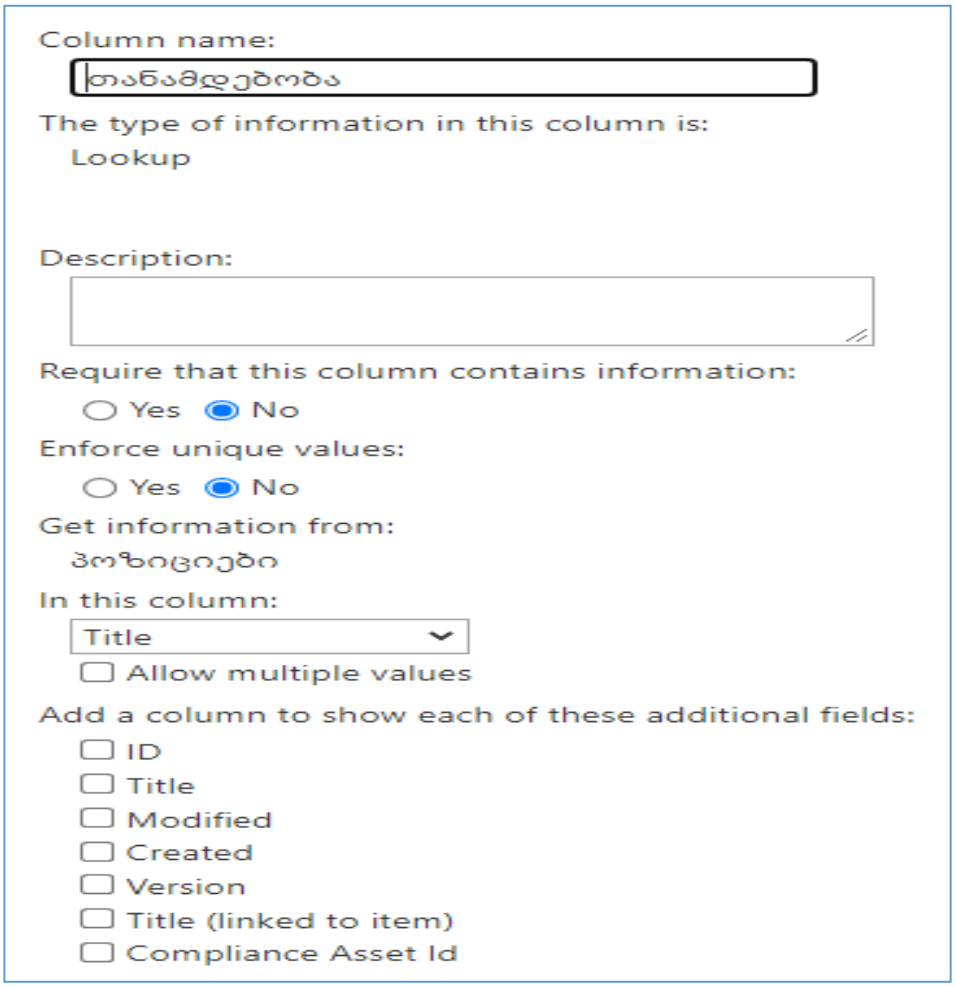
ახლა შევქმნათ პოზიციების (საშტატო თანამდებობების) სია (ნახ.5.6).



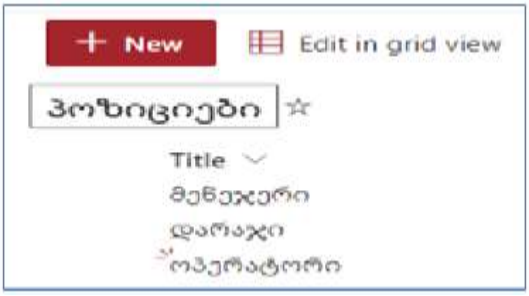
ნახ.5.6

- *Lookup -ველის შექმნა*

კერძოდ, თანამდებობის მითითებისას შესაძლებელია პოზიციის სიიდან ამორჩევა (ნახ.5.7-ა,ბ).



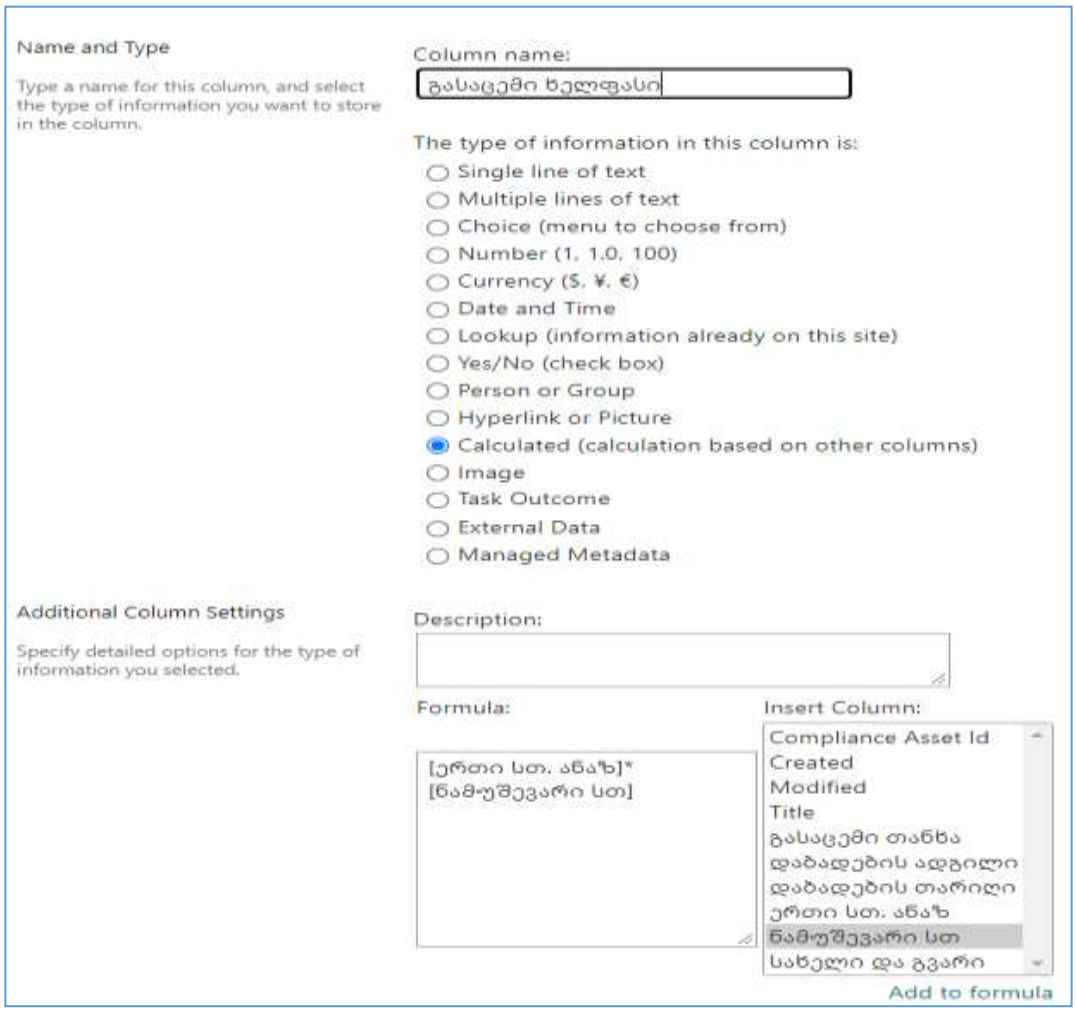
ნახ.5.7-ა



ნახ.5.7-ბ

- **Calculated ველის შექმნა**

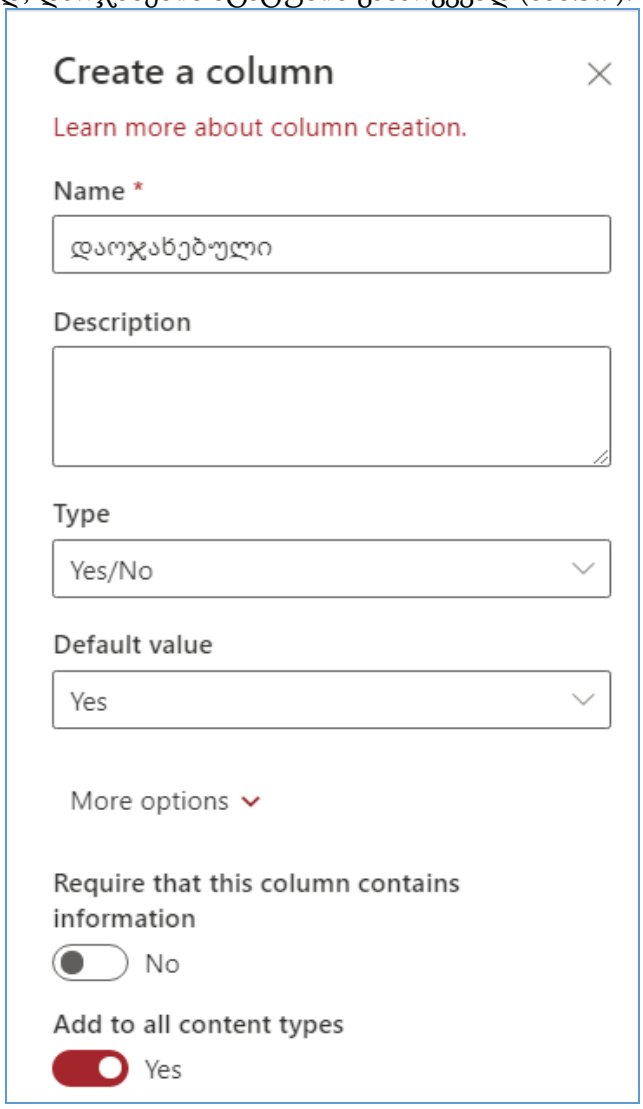
მაგალითად, ხელზე გასაცემი ხელფასი გამოითვლება ნამუშევარი საათების და ერთი საათის ანაზღაურების მიხედვით.



ნახ.5.8

- **yes/no ველის შექმნა**

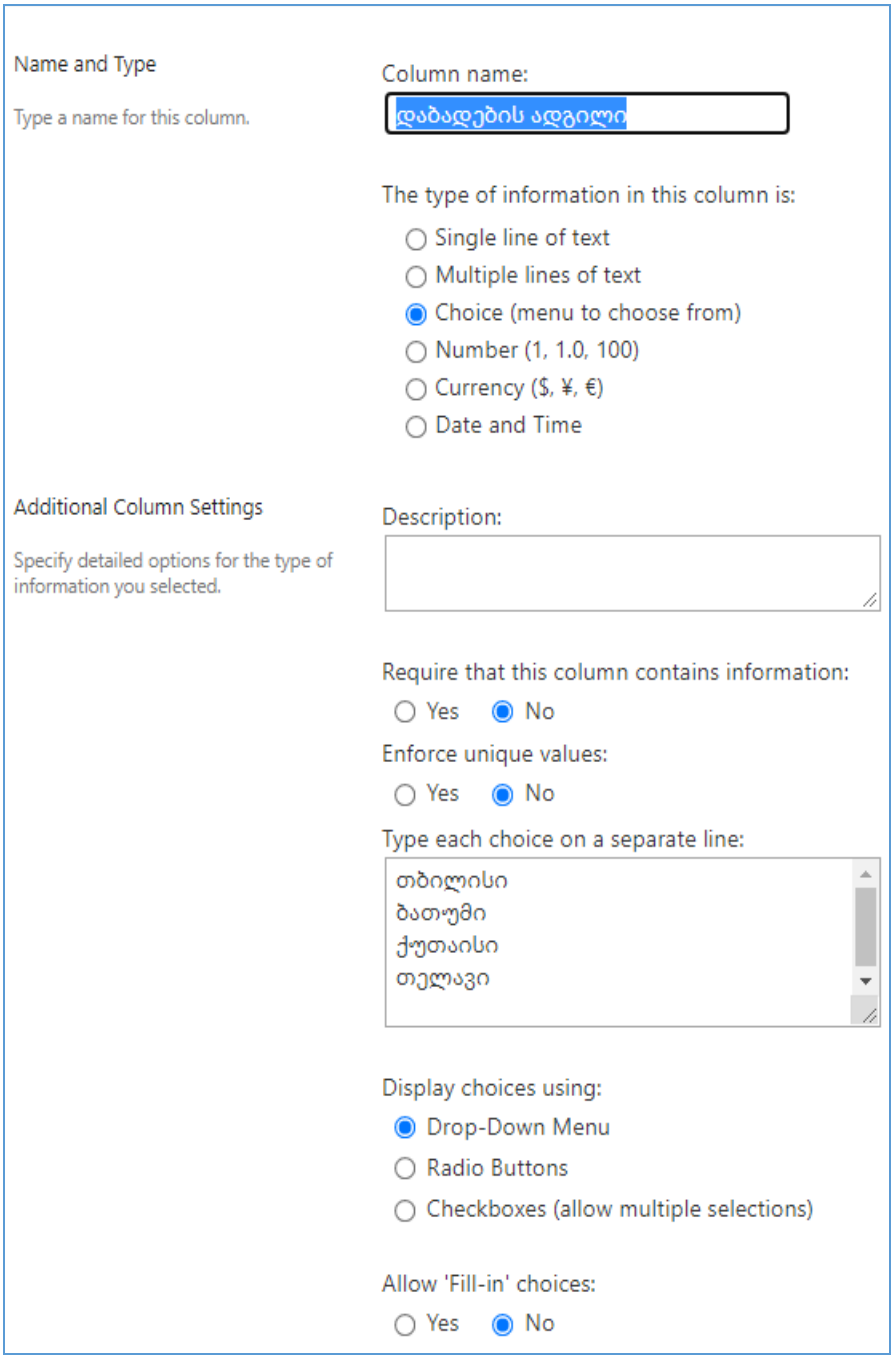
მაგალითად, დაოჯახების სტატუსის გასარკვევად (ნახ.5.9).



ნახ.5.9

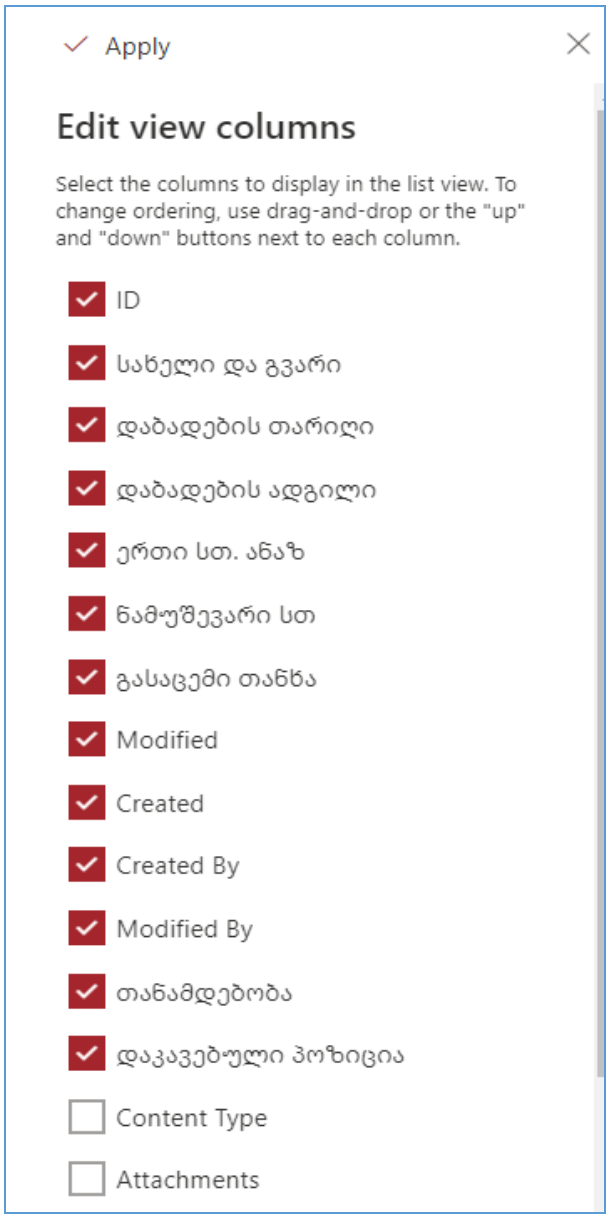
- **Choice ველის შექმნა**

მაგალითად, დაბადების ადგილის ასარჩევად (ნახ.5.10).



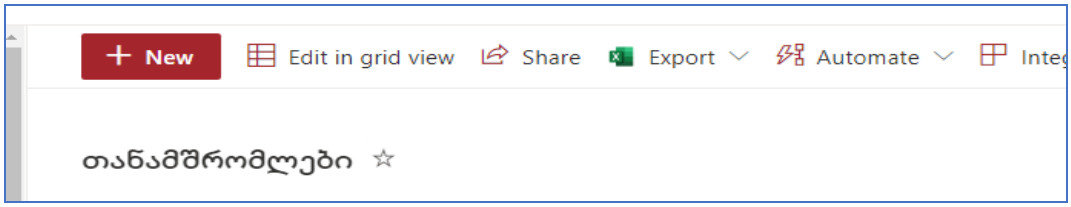
ნახ.5.10

სიებში აუცილებლად უნდა გამოვიყენოთ Created by, Modified by, Created, Modified და ID ველები. აღნიშნული ველების გააქტიურება ხდება Show/hide columns ბრძანებით (ნახ.5.11).

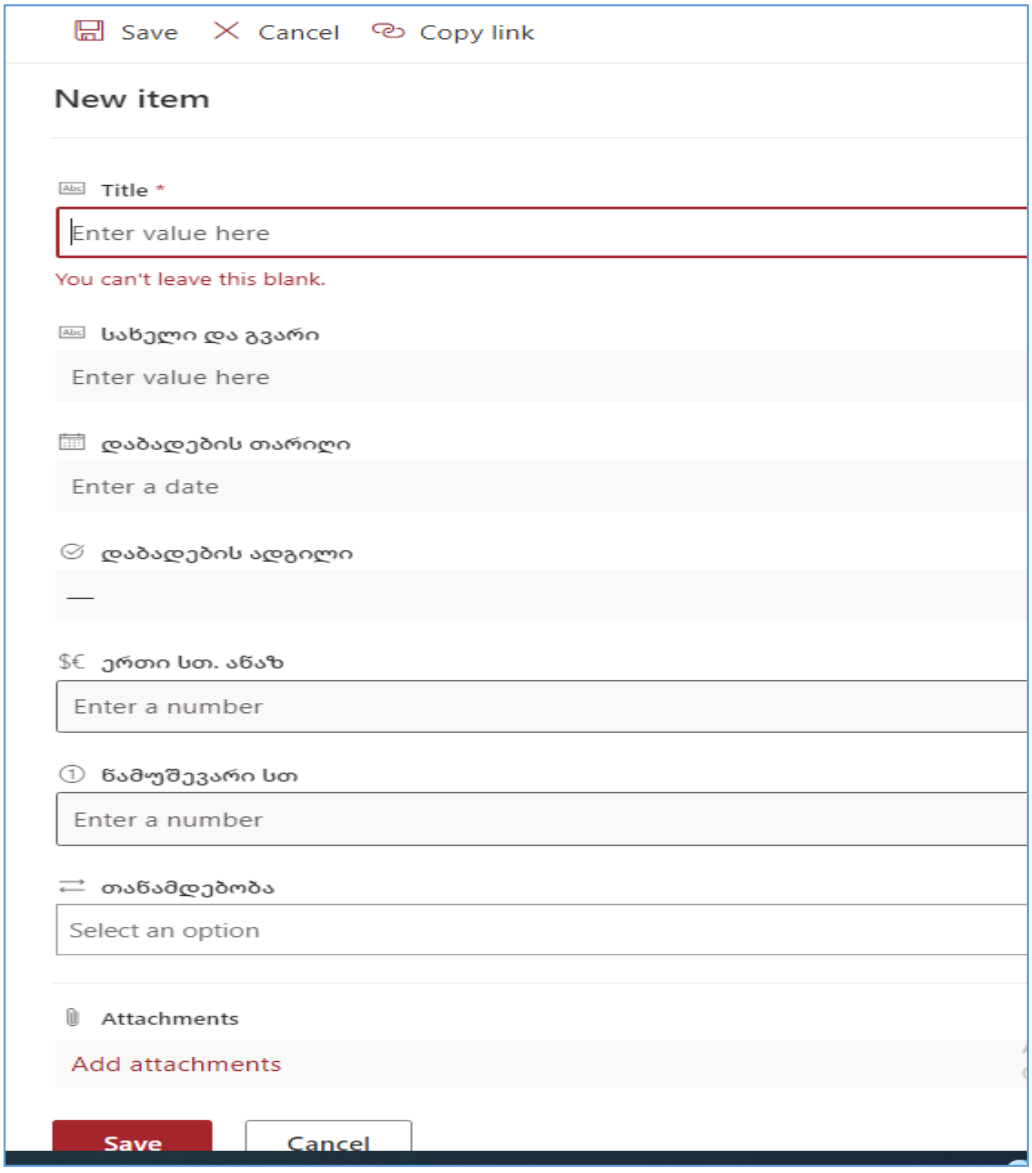


ნახ.5.11

სიის შევსება ხდება New ღილაკით (ნახ.5.12-ა,ბ).

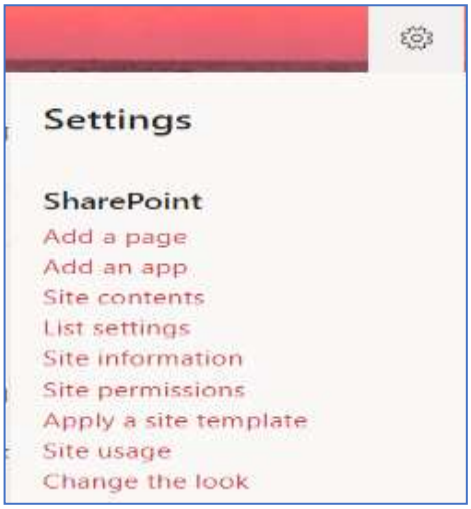


ნახ.5.12-ა



ნახ.5.12-ბ

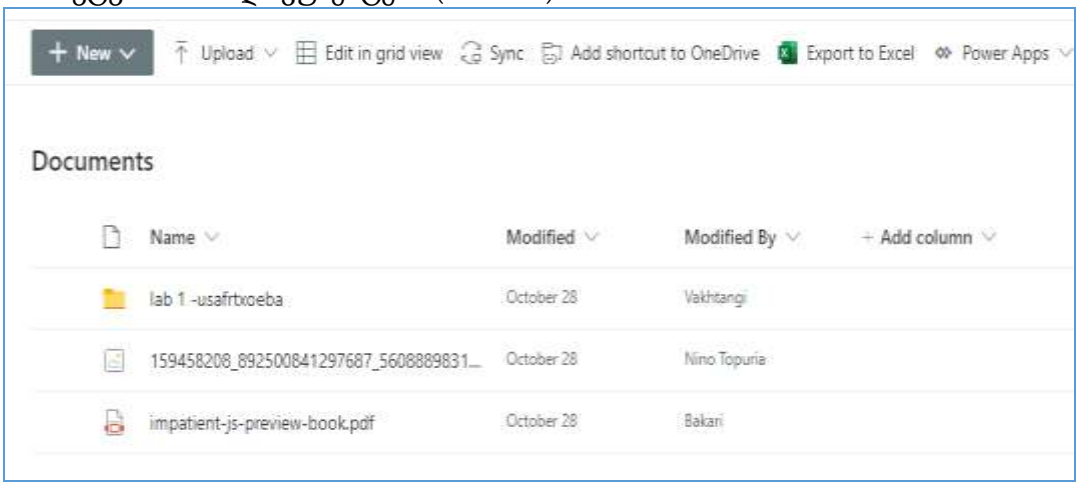
სიის სვეტების რედაქტირება, ჩამატება/წაშლა - ხდება List Settings ბრძანებით (ნახ.5.13).



ნახ.5.13

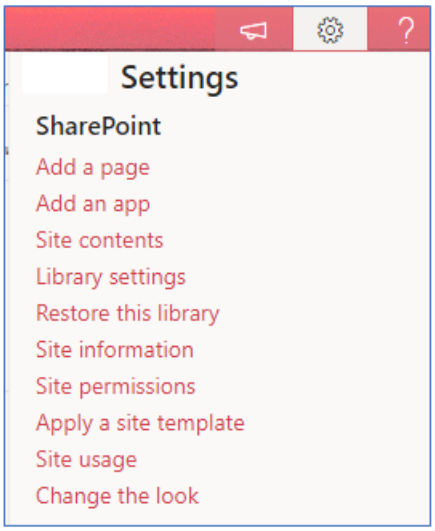
5.3. დოკუმენტებთან მუშაობა (Document Library)

ავტვირთოთ დოკუმენტები (ნახ.5.14).

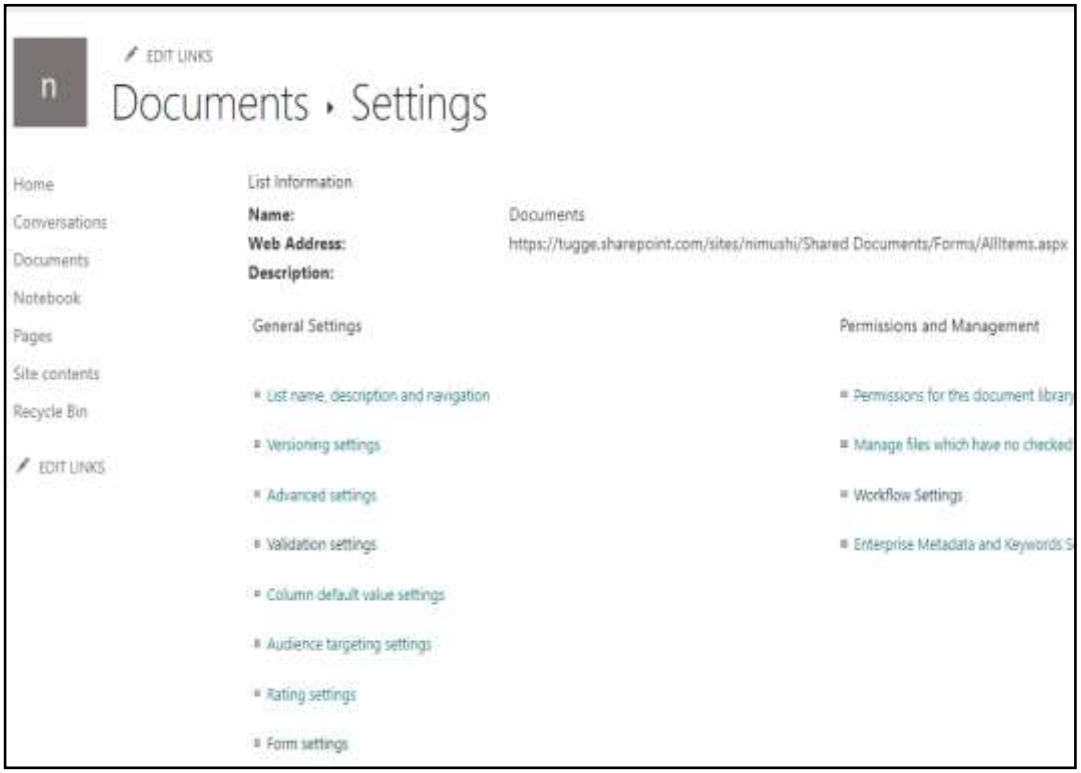


ნახ.5.14

დოკუმენტების ბიბლიოთეკასთან სამუშაო ბრძანებები მოცემულია Library settings მენიუში (ნახ.5.15-ა,ბ).



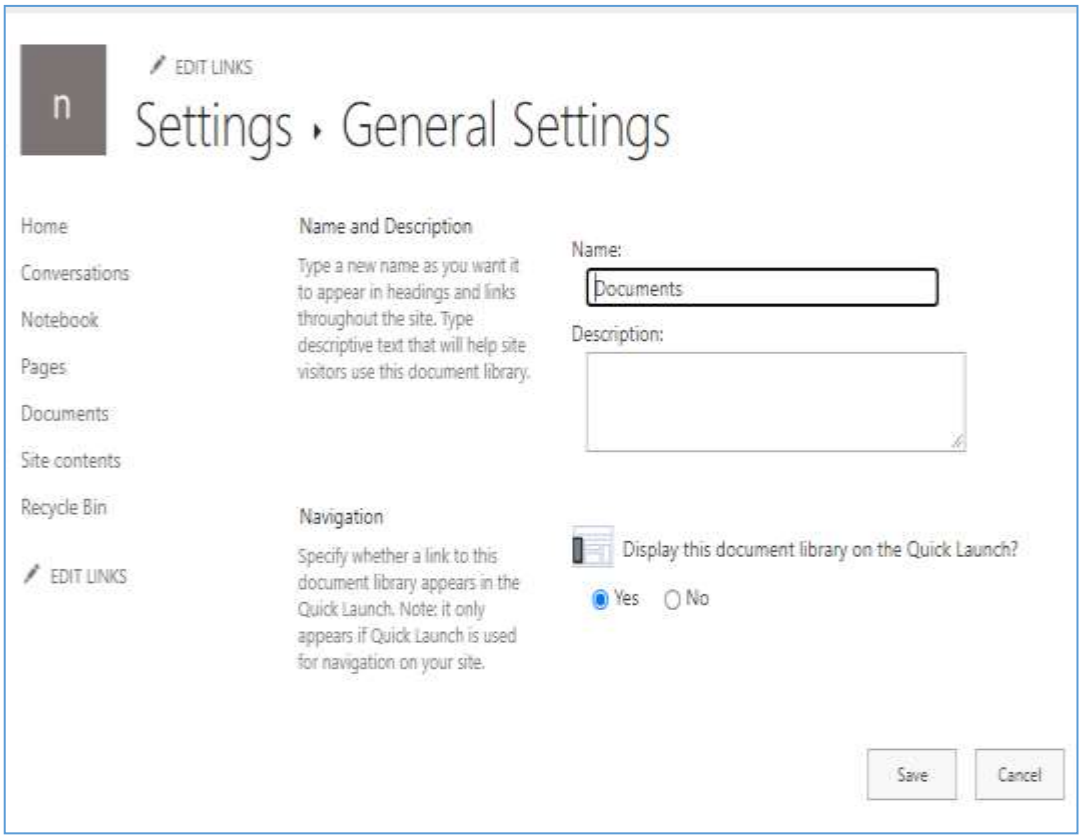
ნახ.5.15-ა



ნახ.5.15-ბ

Permissions for this document library - შესაძლებელია თითოეული წევრისთვის დოკუმენტების ბიბლიოთეკასთან წვდომის უფლებების განსაზღვრა.

Versioning settings - დოკუმენტებისთვის ვერსიების რეჟიმის ჩართვა.
List name, description and navigation - ნავიგაციის ჩართვა/გამორთვა (ნახ.5.16).



ნახ.5.16

5.4. Web-გვერდის შექმნა და მოდიფიკაცია, გვერდის სახეები

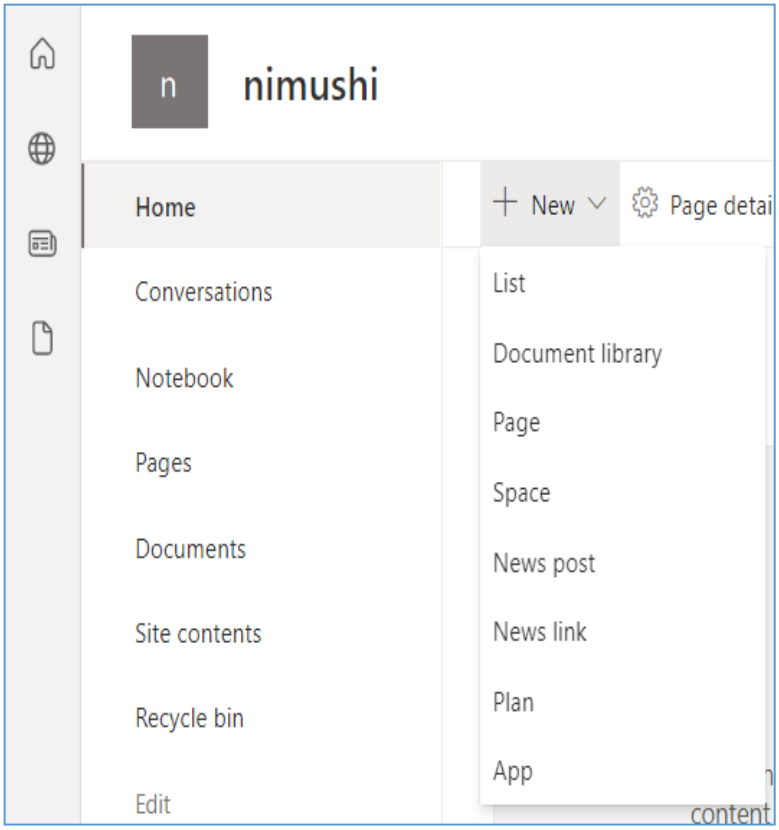
- *გვერდის შექმნა და მოდიფიკაცია*

გვერდის შექმნა ხდება Home→ Page → Create Page ბრძანებით (ნახ.5.17).

Page details ბრძანებით შესაძლებელია სურათის, სახელწოდების და ა.შ შეცვლა,

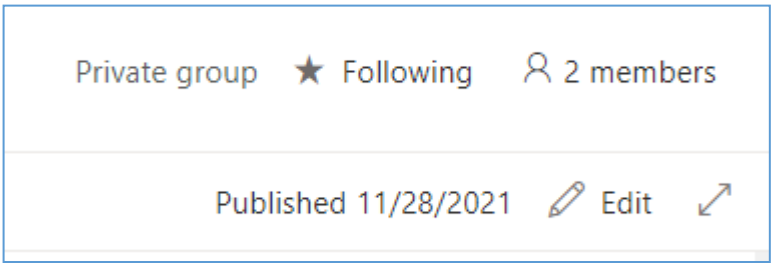
Edit ბრძანებით შესაძლებელია ვებ ნაწილებთან მუშაობა. მაგალითად, რუკის, ამინდის და ა.შ ჩანერგვა.

ვირტუალური ფირმის მისამართის ჩვენება Bing-ის საშალებით.

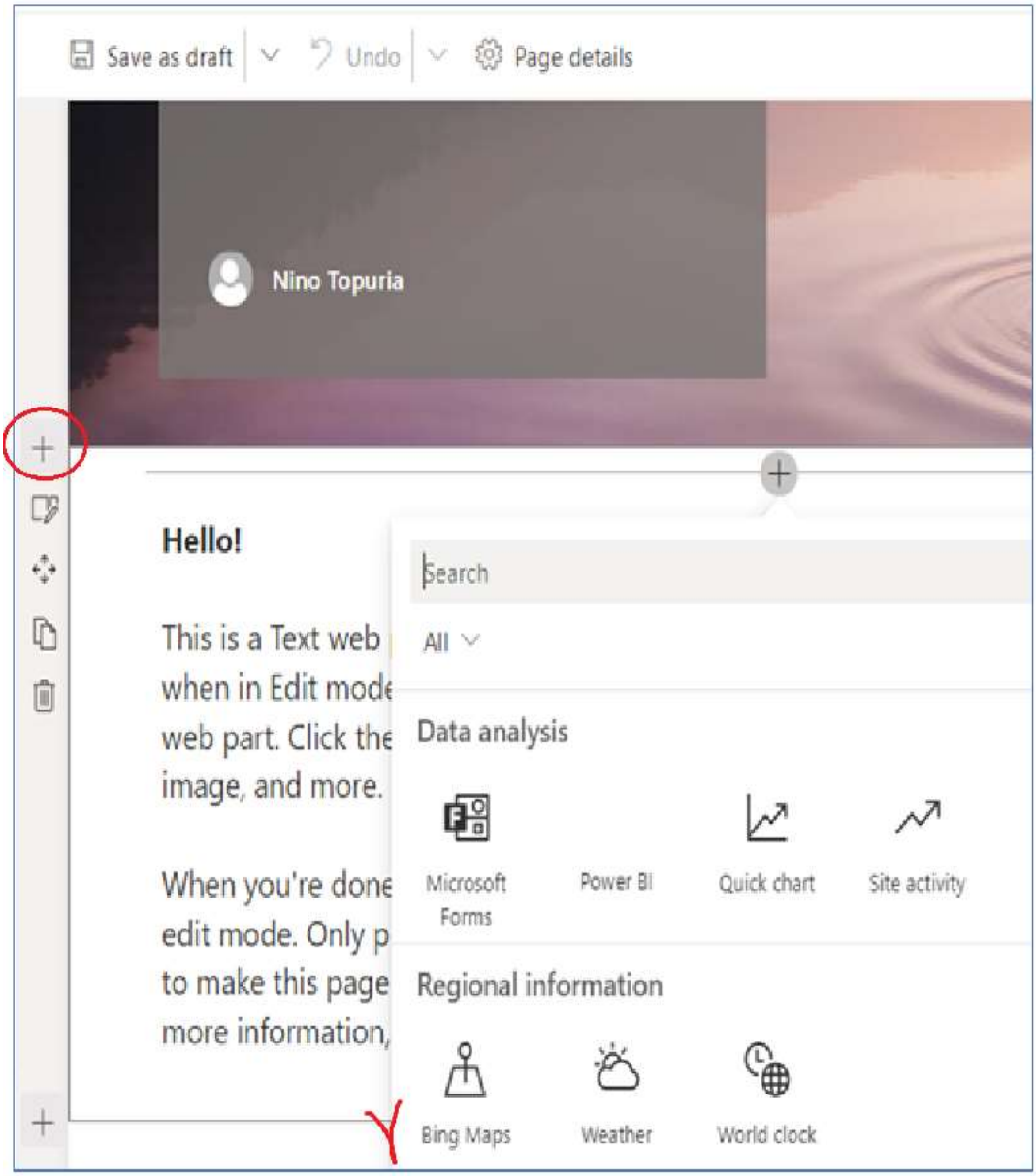


ნახ.5.17

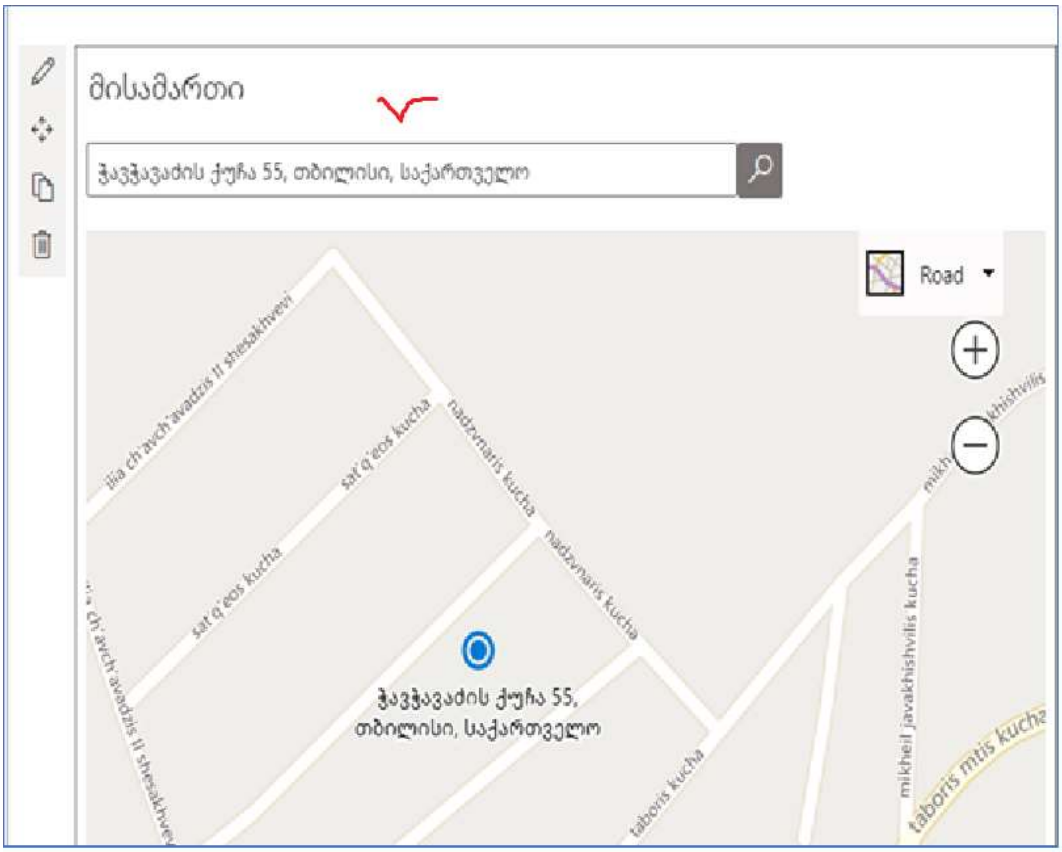
აირჩიეთ edit ღილაკი (ნახ.5.18-ა), შემდეგ „ +“, შემდეგ Bing (ნახ.5.18-ბ), ჩაწერეთ მისამართი (ნახ.5.18-გ)



ნახ.5.18-ა


Save as draft
Undo
Page details
Nino Topuria
Hello!
This is a Text web
when in Edit mode
web part. Click the
image, and more.
When you're done
edit mode. Only p
to make this page
more information,
Search
All
Data analysis
Microsoft Forms
Power BI
Quick chart
Site activity
Regional information
Bing Maps
Weather
World clock

ნახ.5.18-ბ



ნახ.5.18-გ

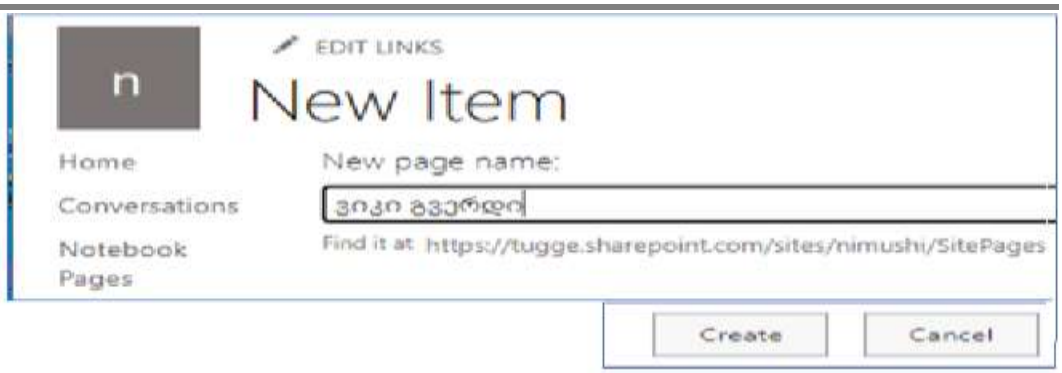
- *Web-გვერდის სახეები:*

არსებობს *სამი სახის გვერდის* შექმნის საშუალება.
სხადასხვა ტიპის გვერდების შექმნა შესაძლებელია ბრძანებით:

Pages→New→ Wiki Page, Web part Page, Site Page

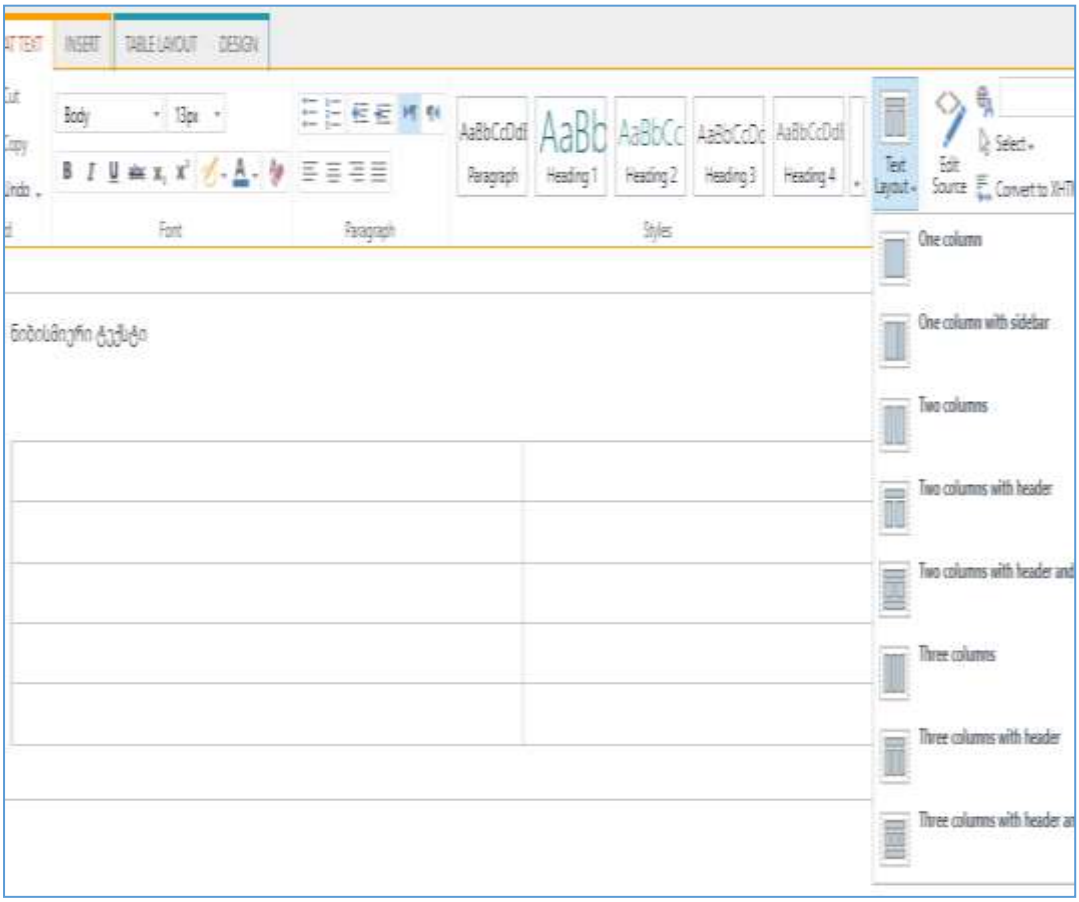
1) Wiki Page გვერდის შექმნა

განვიხილოთ Wiki Page. ვიკი-გვერდები შეიცავს ინფორმაციას რაიმე ფაქტების შესახებ ან კონკრეტულ რჩევებს. ვიკი-ბიბლიოთეკის ახალი გვერდის შექმნა შეიძლება შემდეგი თანმიმდევრობით (ნახ.5.19).



ნახ.5.19

ავირჩიოთ გვერდის მაკეტი, ჩავსვათ ტექსტი, ცხრილი და ა.შ. (ნახ.5.20).

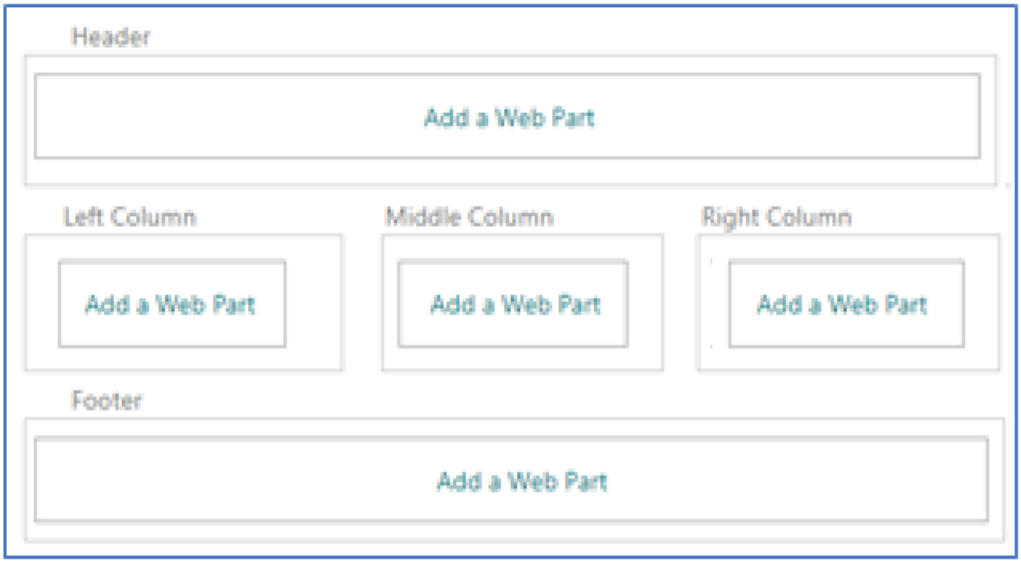


ნახ.5.20

დავიმახსოვროთ Save ღილაკით.

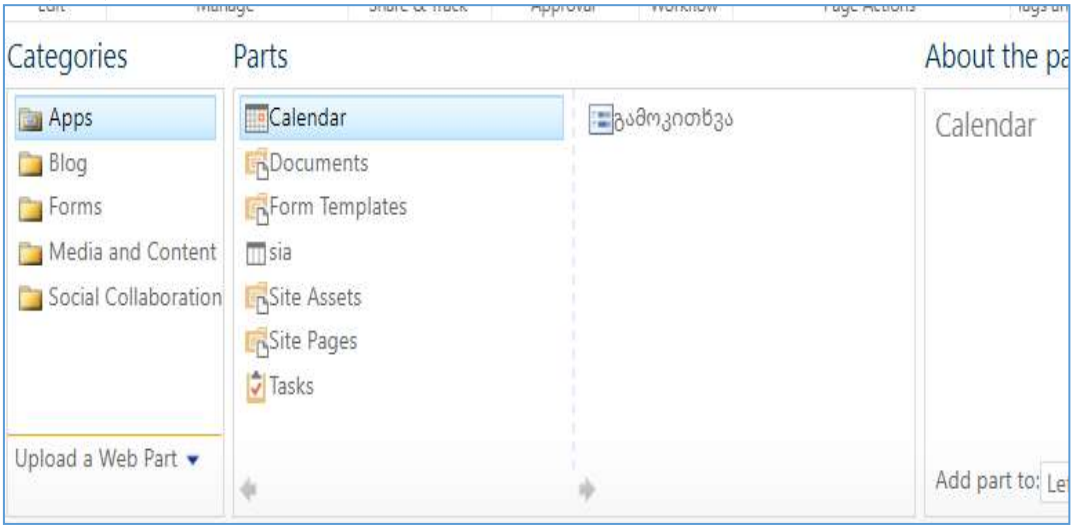
2) Web Part Page გვერდის შექმნა

ავირჩიოთ ბრძანება Pages -> New -> Web Part Page (ნახ.5.21)

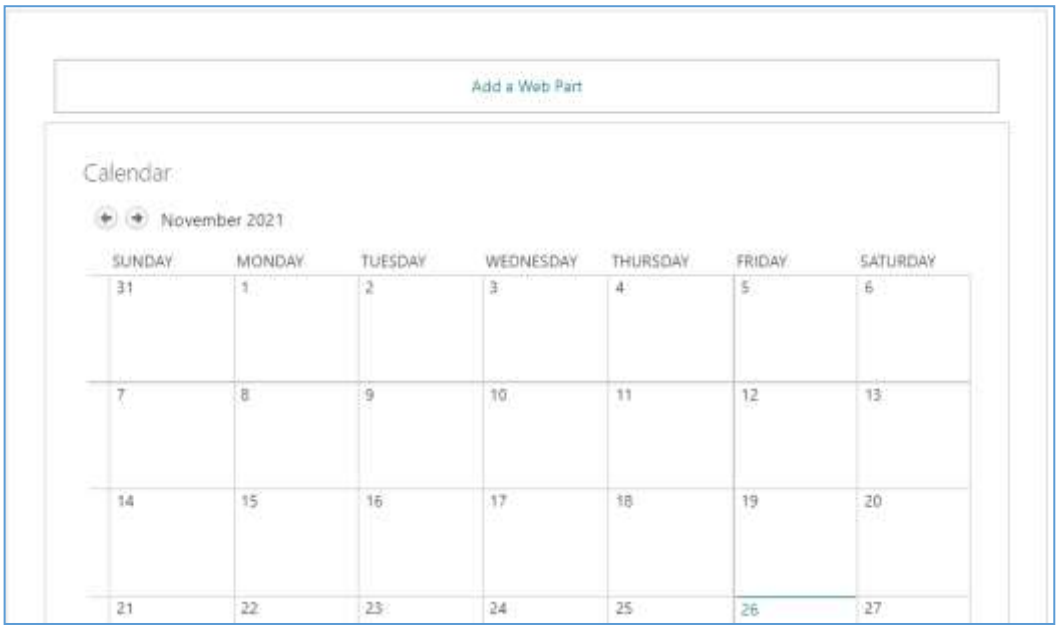


ნახ.5.21

ავირჩიოთ ვებ ნაწილი, რომლის ჩაშენებაც გვინდა. მაგალითად ჩავაშენოთ კალენდარი (ნახ.5.22-ა,ბ)

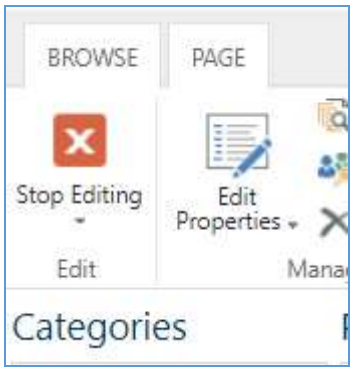


ნახ.5.22-ა



ნახ.5.22-ბ

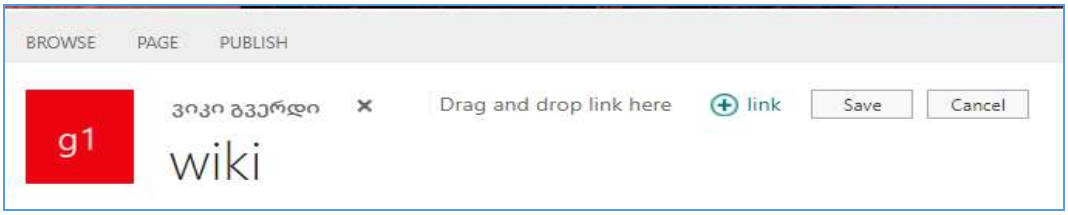
ავირჩიოთ Stop Editing (ნახ.5.23).



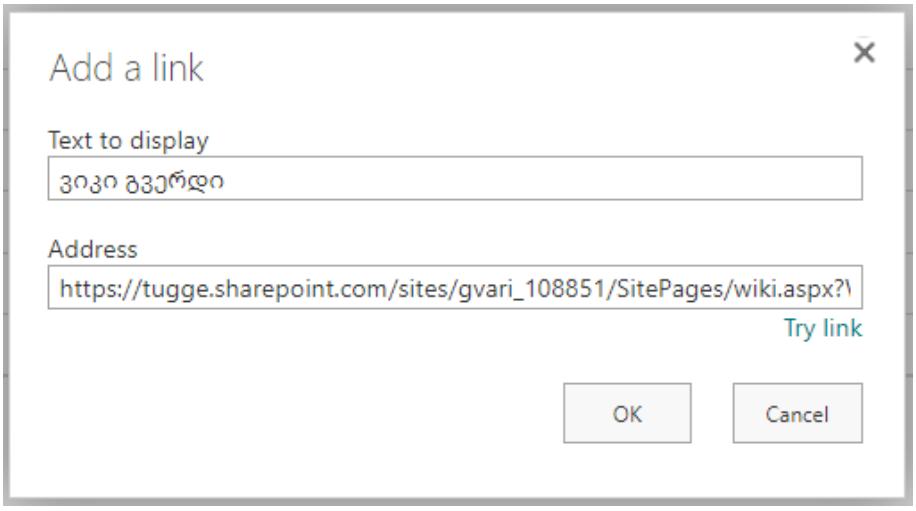
ნახ.5.23

5.5. ნავიგაციის პანელის შექმნა

გვერდის Settings მენიუდან ავირჩიოთ ბრძანება
Site settings > Edit Links
განვიხილოთ 5.24-ა,ბ,გ ნახაზები.

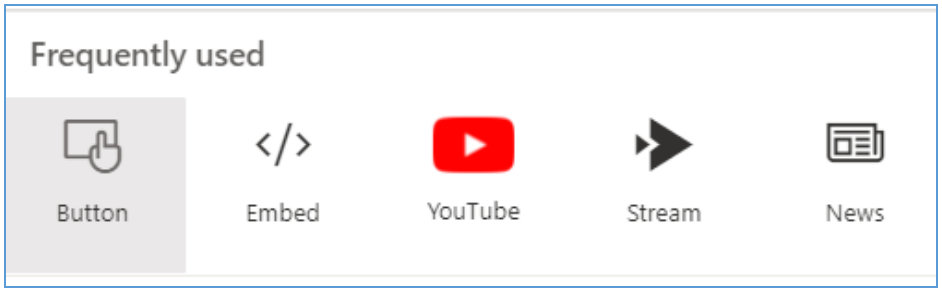


ნახ.5.24-ა



ნახ.5.24-ბ

Button-ით შესაძლებელია რომელიმე საიტზე, App-ზე ან გვერდზე გადასვლა(ნახ.5.24-გ).

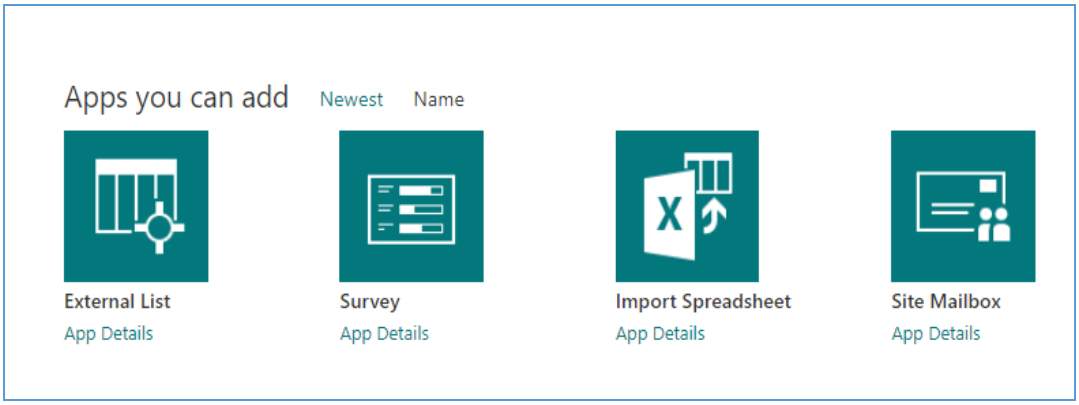


ნახ.5.24-გ

5.6. App-ის ჩაშენება

კოპროპატიული პორტლალის ვებ გვერდზე შესაძლებელია სხადასხვა აპლიკაციების ჩაშენება (Asanam Bitucket, GitHub, JIRA, Trello და სხვ.).

საჭირო აპლიკაციის არჩევა შესაძლებელია ვებ-გვერდის რედაქტირების რეჟიმიდან. მაგალითისთვის განვიხილოთ Survay აპლიკაცია, რომელიც ფირმაში სხადასხვა სახის გამოკითხვის ჩატარების საშუალებას იძლევა.



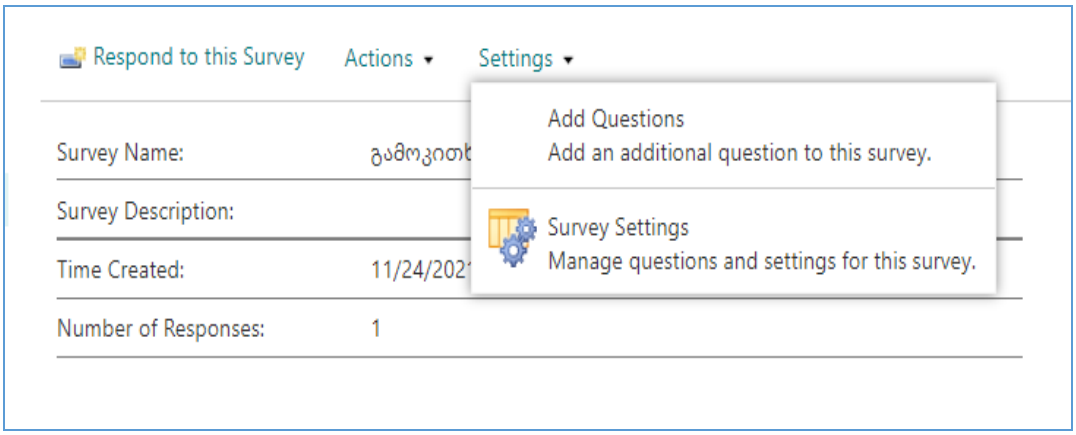
ნახ.5.25

აპლიკაცია Survay-ის ჩასაშენებლად აირჩიეთ ბრძანება

Home -> New apps -> Classic Experience -> Survay

ნახ.5.26

ჩავწეროთ გამოკითხვისთვის საჭირო შეკითხვები, რომელიც ამა თუ იმ სალითხთან დაკავშირებით თანამშრომლების აზრის გარკვევაში დაეხმარება ორგანიზაციის ხელმძღვანელობას (ნახ.5.27).



ნახ.5.27

. .

5.7. სამუშაო პროცესების ავტომატიზაცია Power Automate-ით

Power Automate დაფუძნებულია ტრიგერებზე (triggers) და მოქმედებებზე (actions). ტრიგერის საშუალებით იწყება ნაკადი (flow) ანუ ვირჩევთ მოვლენას, რომლის მიხედვითაც იგეგმება მოქმედება. მოქმედება არის ის, რაც ხდება ნაკადის გააქტიურების შემდეგ. ეს შეიძლება იყოს დავალების შექმნა ან ერთი ან მეტი მოქმედების შესრულება.

დავსვათ ამოცანა.

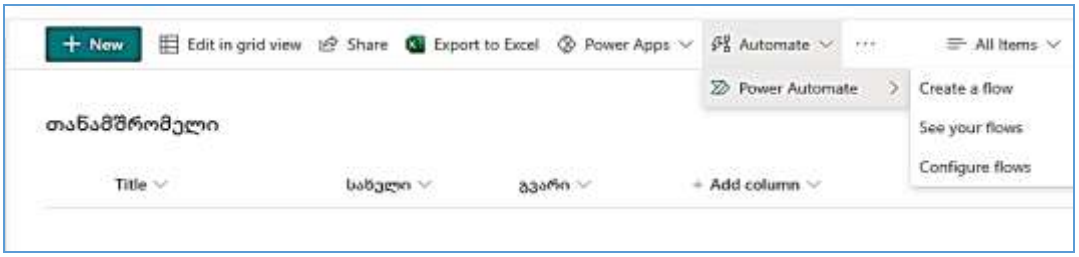
გვინდა შევქმნათ ნაკადი, რომელიც გააქტიურდება თანამშრომლების სიაში ჩანაწერის შექმნისთანავე, და მიუვა შეტყობინება მაგალითად, მენეჯერს, იმის შესახებ, რომ მოხდა ახალი თანამშრომლის მიღება.

- **Power Automate გააქტიურება**

ნებისმიერ ინტერნეტ-ბრაუზერში აკრიფეთ **poratl.office.com**, შეიტანეთ მომხმარებლის სახელი და პაროლი. გააქტიურდება Office 365 მთავარი გვერდი, აირჩიეთ **Power Automate.**

აირჩიეთ შაბლონი - **Start approval when a new item is added**

Power Automate-ს გააქტიურება ასევე შესაძლებელია უშუალოდ კორპორაციის sharepoint online-ის საშუალებით დაპროექტებული ვებ-პორტალიდან. 5.28 ნახაზზე მოცემულია Power Automate-ის გააქტიურება თანამშრომლის სიიდან.



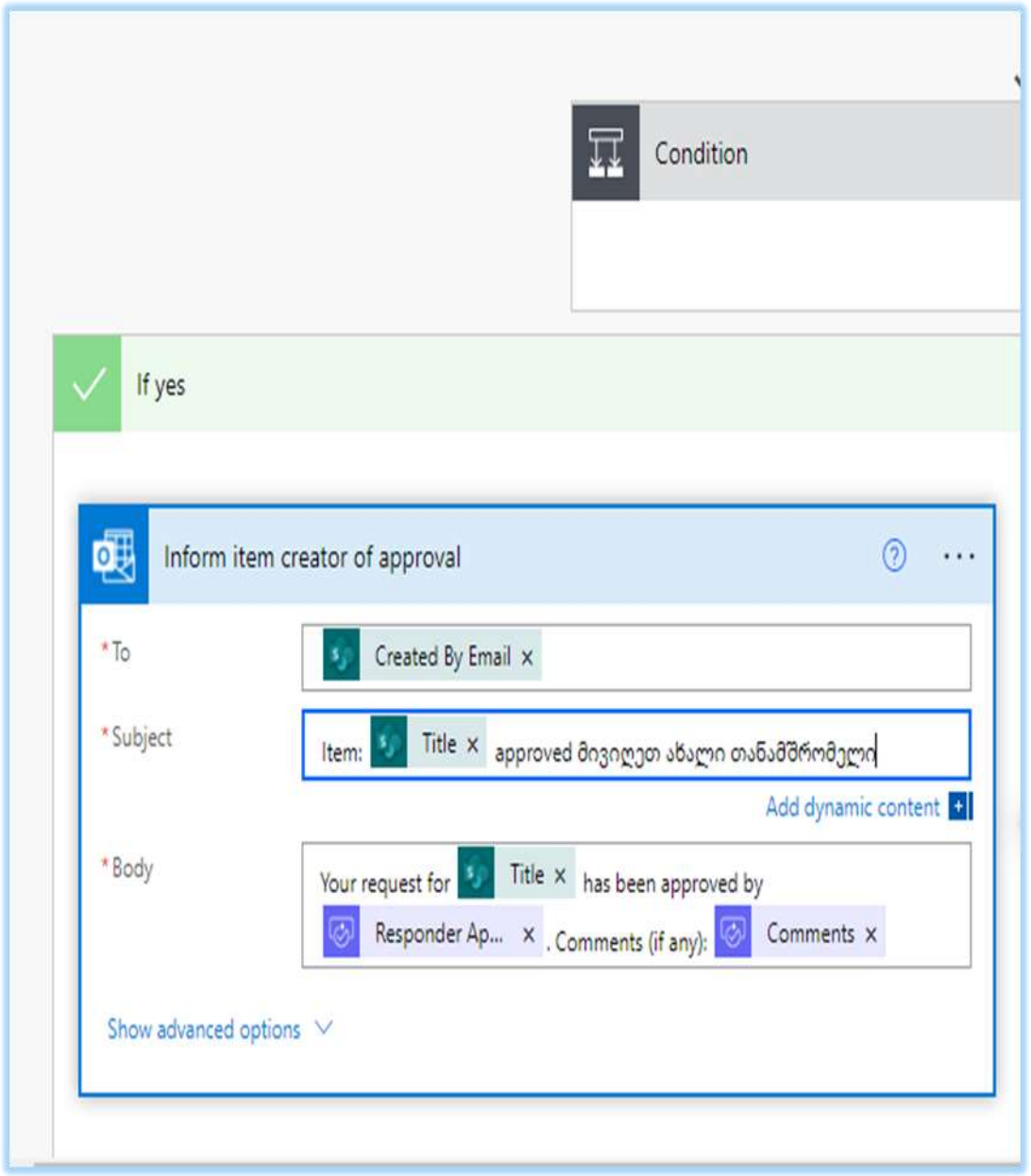
ნახ.5.28

Create a flow ბრძანების არჩევის შედეგად გაიხსნება ფანჯარა, საიდანაც შესაძლებელია საჭირო შაბლონის არჩევა.

ავირჩიოთ Start approval when a new item is added შაბლონი, რომელიც Outlook-ის, Sharepoint-ის და Power Automate-ის Approvals მოქმედების ინტეგრირებული მუშაობის შედეგია (ნახ.5.29).

ნახ.5.29

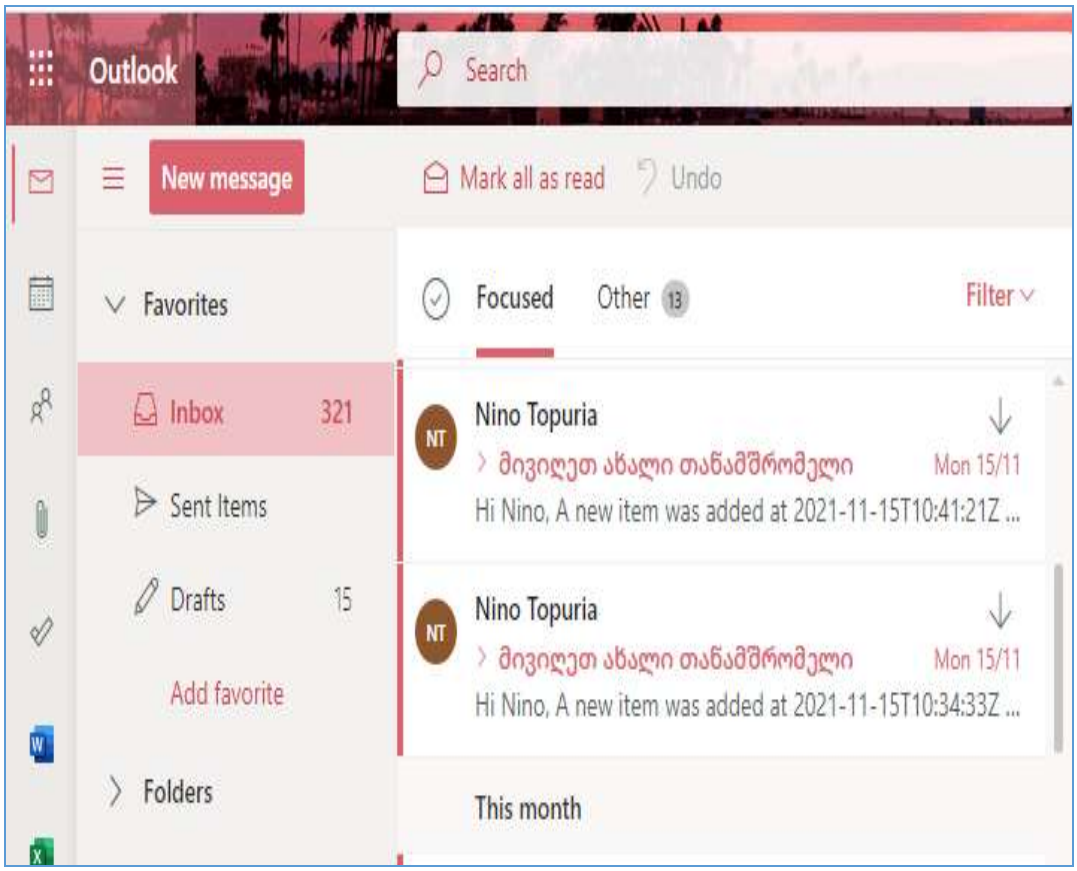
ველში Assigned To მივუთითოთ იმ პიროვნების მეილი, რომელმაც უნდა დაადასტუროს აღნიშნულ სიაში ჩანაწერის დამატება (ნახ.5.30).



ნახ.5.30

SharePoint online ვებ-პორტალზე თანამშრომლის სიაში ახალი ჩანაწერის ჩაწერისთანავე, გაეშვება Power Automate-ში შექმნილი flow და მოხდება შეტყობინების გაგზავნა Assigned To ველში მითითებულ მეილზე, რითაც მოითხოვება აღნიშნული მოქმედების დადასტურება ან უარყოფა. Assigned To ველში მითითებული მომხმარებელი მარტივად ნახავს ამ შეტობინებას Outlook-

ის საშუალებით, სადაც Approve ღილაკით დაადასტურებს ან Reject ღილაკით უარყოფს აღნიშნულ მოქმედებას (ნახ.5.31).



ნახ.5.31

VI თავი

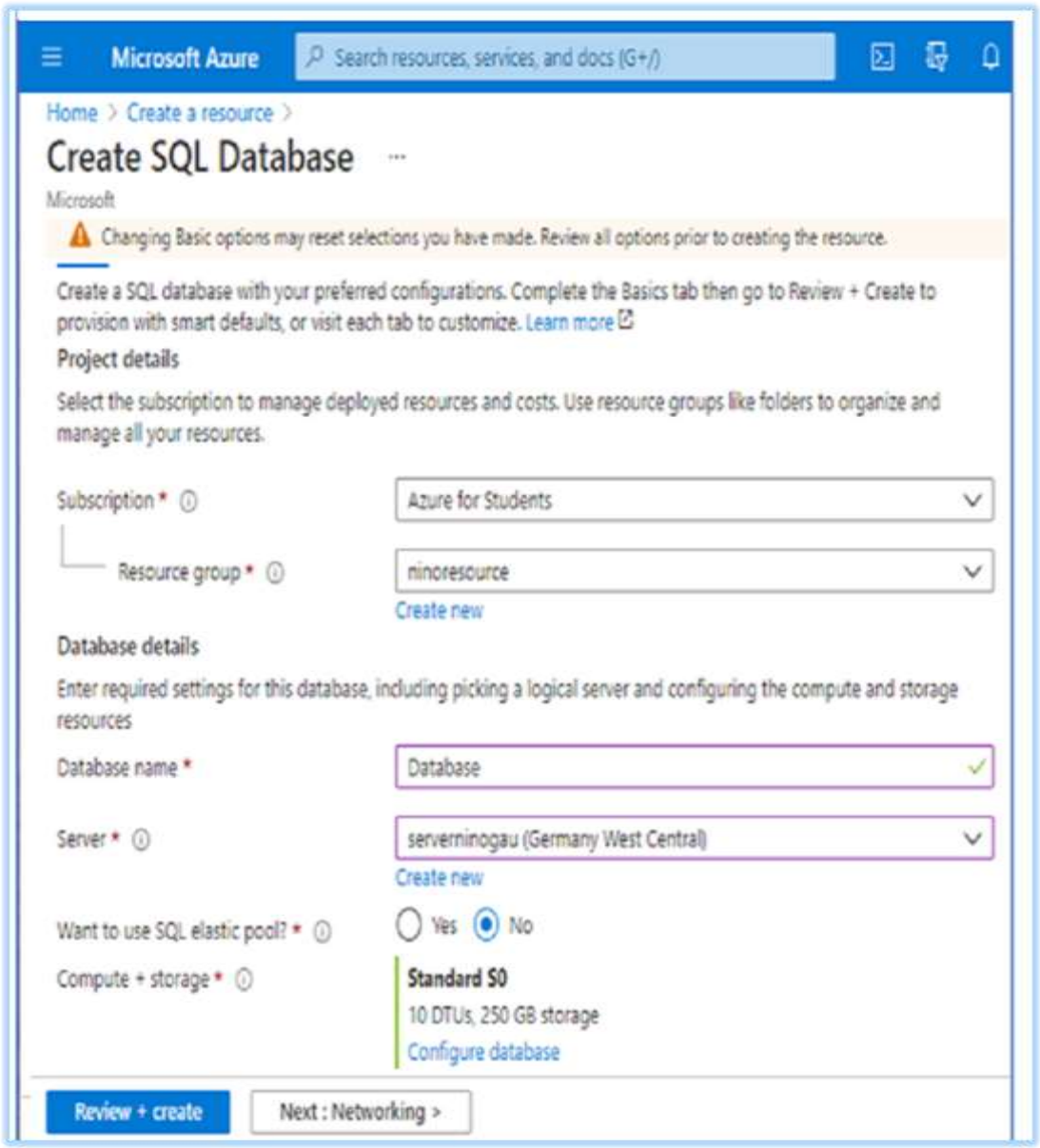
Azure SQL ბაზის დაპროექტება

6.1. Azure პორტალზე SQL მონაცემთა ბაზის შექმნა

Azure პორტალზე, portal.azure.com, ავირჩიოთ ბრძანება

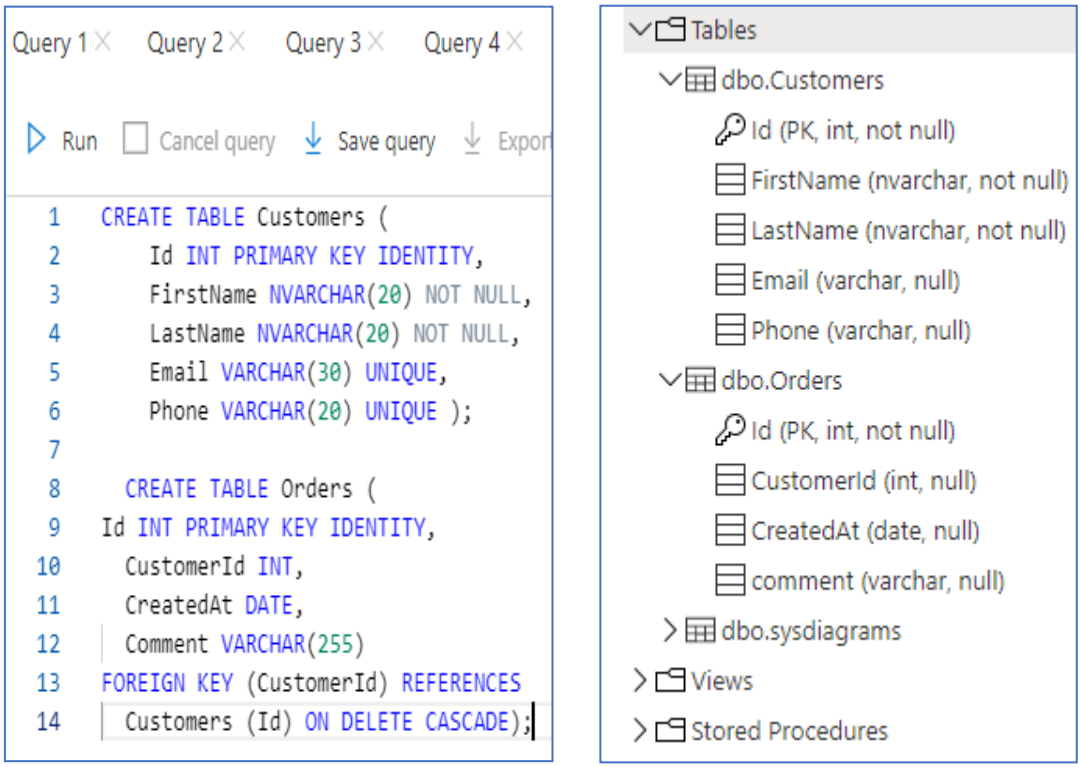
SQL Database →Create

და შევავსოთ დიალოგური ფანჯარა (ნახ.6.1).



ნახ.6.1. მონაცემთა ბაზის შექმნა Azure SQL-ში

მაგალითისთვის, შევქმნათ ცხრილები Customers და Orders (ნახ.6.2).

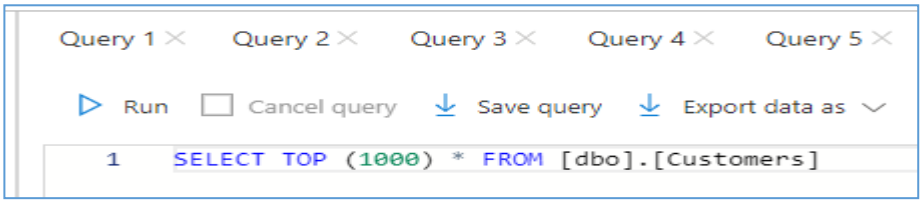


ნახ.6.2. ცხრილების დაპროექტება Azure SQL-ში

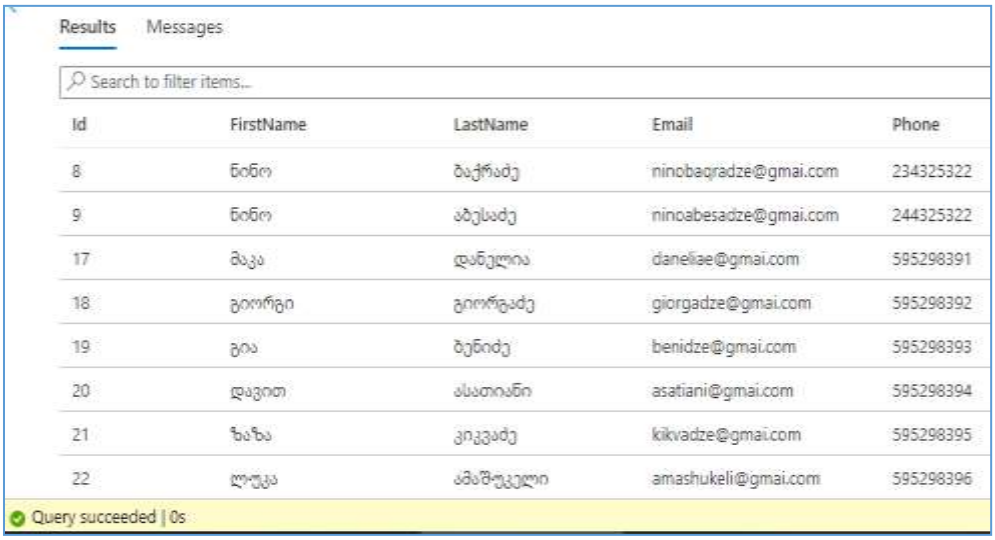
შევავსოთ ჩანაწერებით:

```
insert into [dbo].[Customers] values (N'მაკა', N'დანელია', 'daneliae@gmail.com', 595298391)
insert into [dbo].[Customers] values (N'გიორგი', N'გიორგაძე', 'giorgadze@gmail.com', 595298392)
insert into [dbo].[Customers] values (N'გია', N'ბენიძე', 'benidze@gmail.com', 595298393)
insert into [dbo].[Customers] values (N'დავით', N'ასათიანი', 'asatiani@gmail.com', 595298394)
insert into [dbo].[Customers] values (N'ზაზა', N'კიკვაძე', 'kikvadze@gmail.com', 595298395)
insert into [dbo].[Customers] values (N'ლუკა', N'ამაშუკელი', 'amashukeli@gmail.com', 595298396)
```

შევამოწმოთ როგორ შეივსო ჩანაწერებით ჩვენი ბაზა (ნახ.6.3, 6.4).



ნახ.6.3



ნახ.6.1. Customers-ცხრილის შევსება ჩანაწერებით

ნახაზიდან ჩანს, რომ ჩანაწერებით შევსების პროცესი წარმატებით შესრულდა.

6.2. Sharepoint Online სიების დაკავშირება Azure SQL -თან

დააკავშირეთ sharepoint online სიები azure sql server--თან Power Automate-ის საშუალებით.

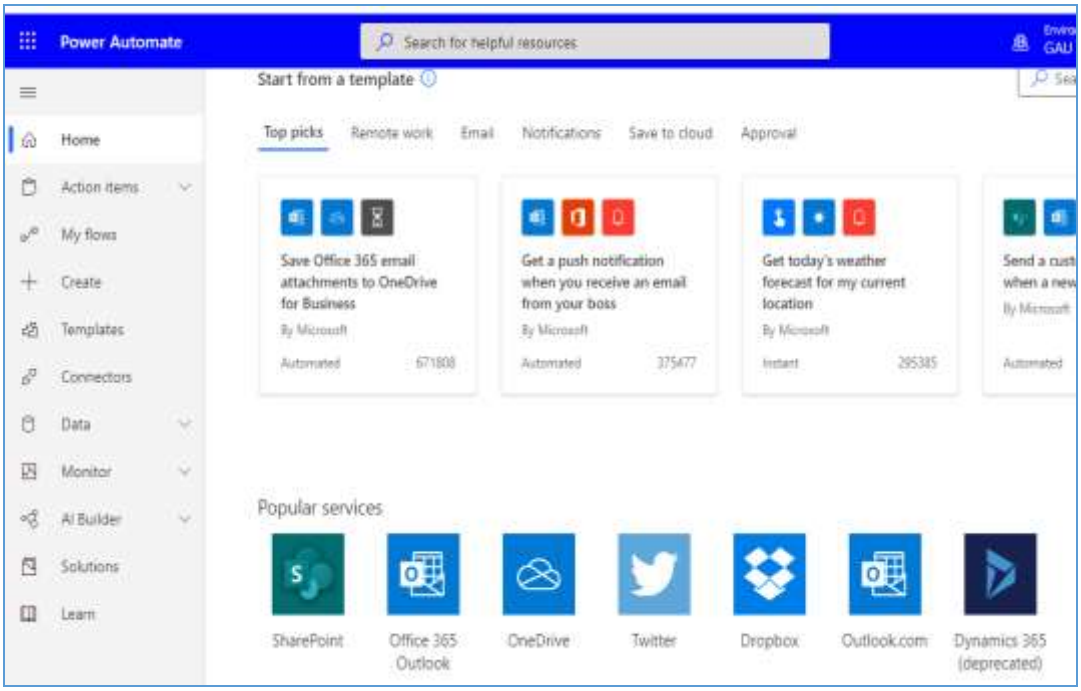
Power Automate - Office 365-ის ერთ-ერთი შემადგენელი აპლიკაციაა. იგი წარმოადგენს ღრუბელზე დაფუძნებულ სისტემას, რომლითაც შეგიძლიათ შექმნათ სამუშაო პროცესების ავტომატიზაცია, გაამარტივოთ ბიზნეს პროცესები და უფრო ეფექტურად მართოთ ისინი. Power Automate დაფუძნებულია ტრიგერებზე (triggers) და მოქმედებებზე (actions). ტრიგერის საშუალებით იწყება ნაკადი ანუ ვირჩევთ მოვლენას რომლის მიხედვითაც იგეგმება მოქმედება. მოქმედება არის ის, რაც ხდება ნაკადის გააქტიურების შემდეგ. ეს შეიძლება იყოს დავალების შექმნა ან ერთი ან მეტი მოქმედების შესრულება.

არსებობს *ხუთი ტიპის ნაკადის (flow)* შექმნის საშუალება:

- *ავტომატიზებული* – მოვლენის შედეგად გამოწვეული ნაკადი, მაგალითად, გაიგზავნოს მეილი, თუ SharePoint სიის ელემენტი შეიცვალა;
- *მყისიერი* – მომხმარებლებს საშუალებას აძლევს ხელით იმოქმედონ მობილურიდან ან აპლიკაციიდან ღილაკის დაჭერით. მაგალითისთვის, მარტივად გაგზავნონ შეხსენების მეილი ორგანიზაციის წევრებთან შეხვედრის დაწყებამდე;
- *დაგეგმილი* – იწყებს მუშაობას გარკვეულ დროს;
- *ბიზნეს პროცესების ნაკადები,* რომელიც ემყარება განსაზღვრულ მოქმედებათა ერთობლიობას;
- *UI (user interface) ნაკადები,* რომლებიც გამოიყენება Windows და Web პროგრამებში განმეორებადი ამოცანების ავტომატიზაციისათვის.

➢ **ბიჯი 1:** *Power Automate გააქტიურება*

ნებისმიერ ინტერნეტ-ბრაუზერში აკრიფეთ **poratal.office.com**, შეიტანეთ მომხმარებლის სახელი და პაროლი. გააქტიურდება Office 365 მთავარი გვერდი, აირჩიეთ **Power Automate.**

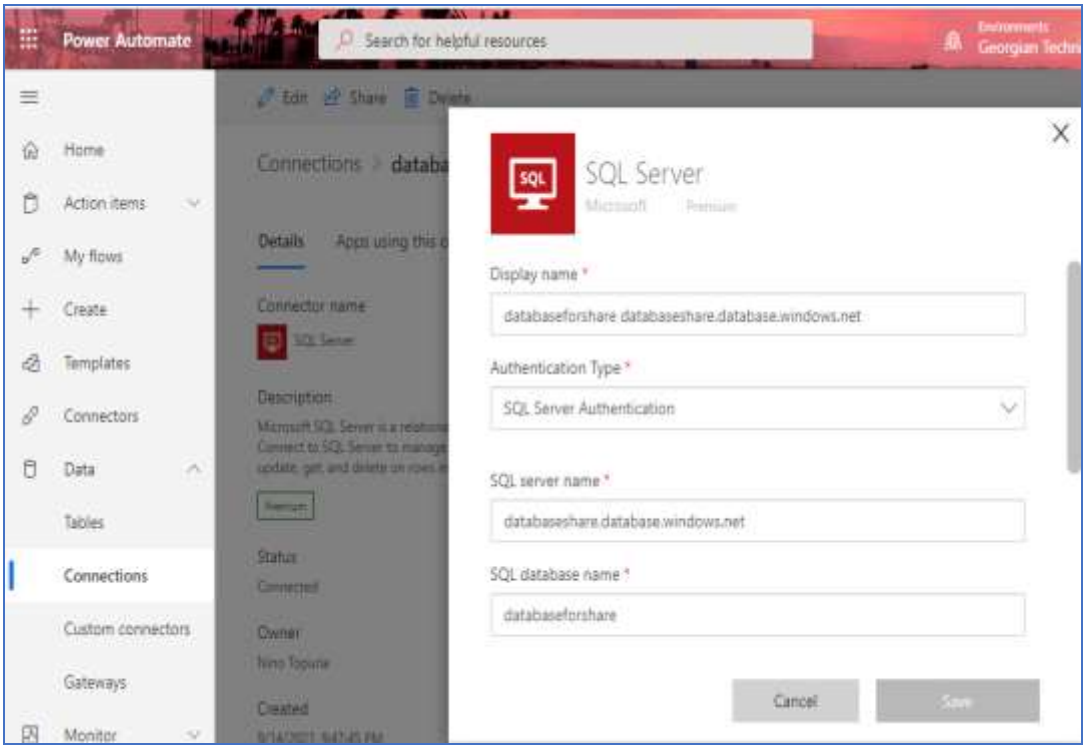


ნახ.6.5

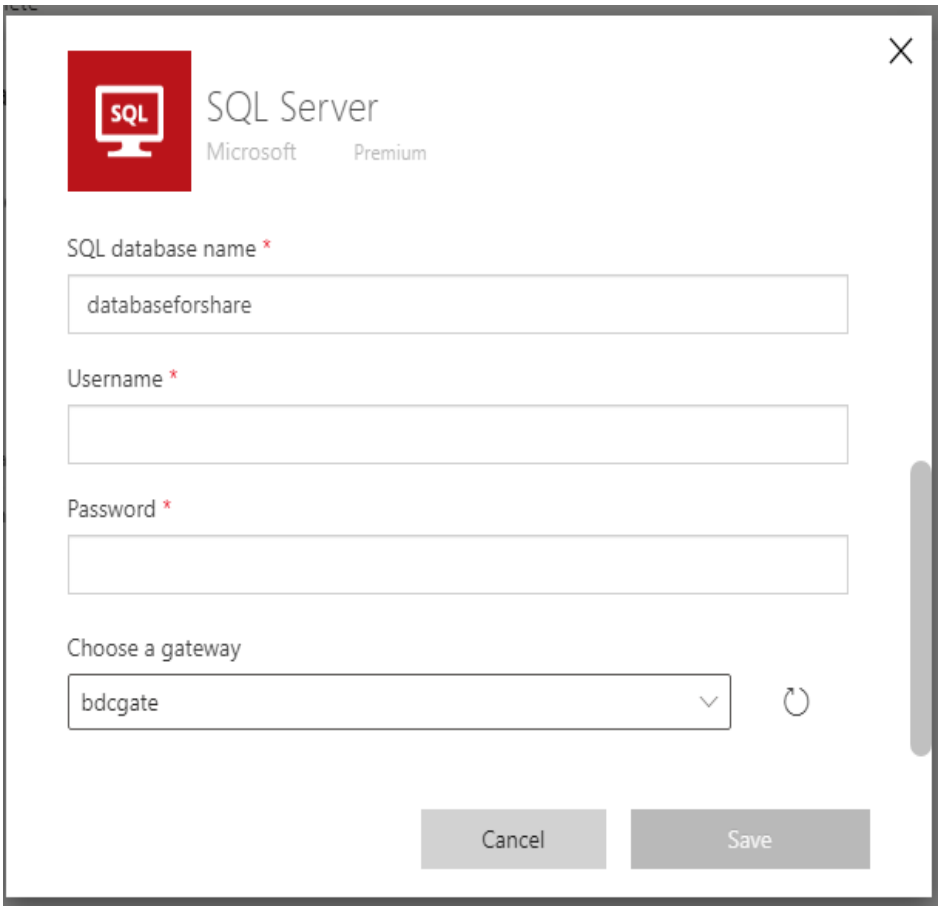
- ***ბიჯი 2 :*** *ეკრანზე გამოჩნდება Power Automate-ის შაბლონები. ძებნის ველში აკრიფეთ Sharepoint. გამოჩნდება* (ნახ.6.6).

ნახ.6.6

შეავსეთ საჭირო ველები, ნიმუში მოცემულია ნახ.6.7 -ზე



ნახ.6.7



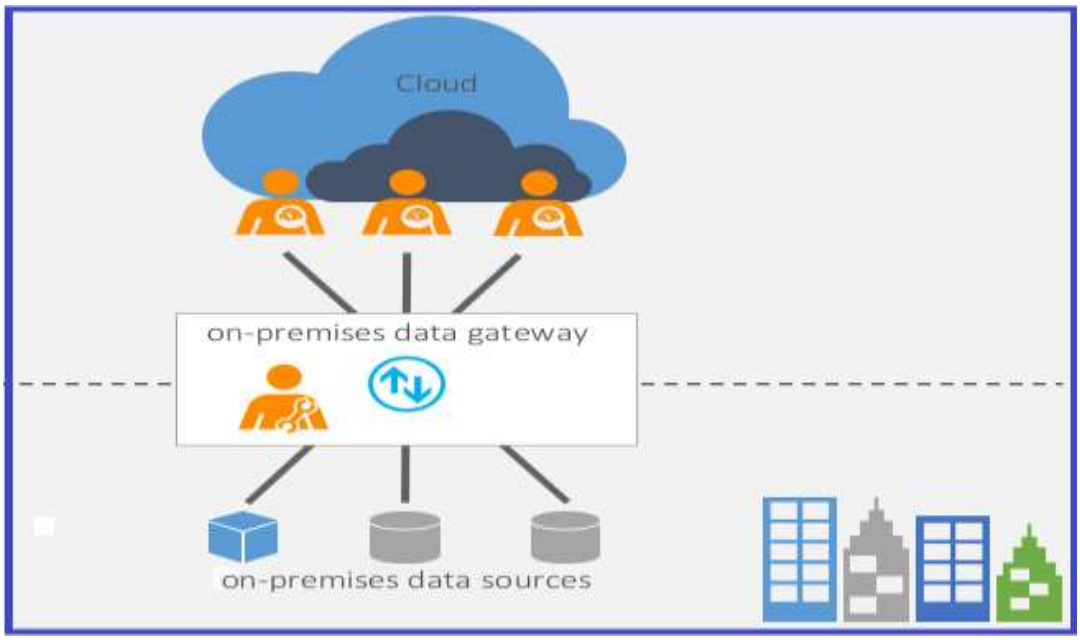
ნახ.6.8

ჩამოტვირთეთ და დააინსტალირეთ Gateway მაიკროსოფტის საიტიდან.

on-premises data gateway მოქმედებს როგორც ხიდი, რათა უზრუნველყოს მონაცემთა სწრაფი და უსაფრთხო გადაცემა შიდა მონაცემებს შორის (მონაცემები, რომლებიც არ არის ღრუბელში) და Microsoft-ის სხადასხვა ღრუბლოვან სერვისს შორის.

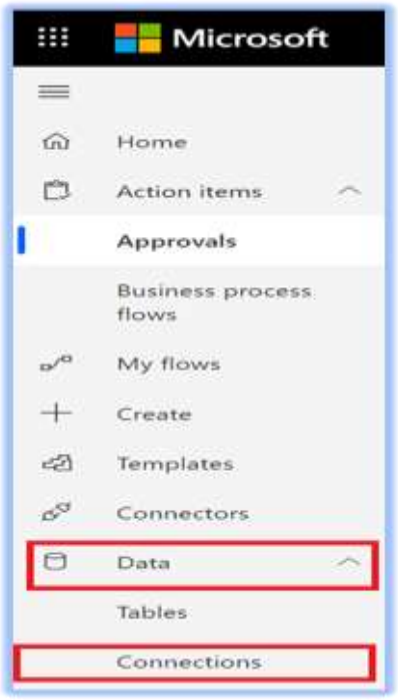
ეს ღრუბლოვანი სერვისები მოიცავს:

- Power BI, Power Apps;
- Power Automate;
- Azure Analysis Services და
- Azure Logic Apps.



ნახ.6.9

გამოვიყენოთ Power Automate სამუშაო პროცესების ავტომატიზაციისთვის. კერძოდ, მოვახდინოთ Sharepoint-ის სიაში შეტანილი მონაცემების სინქრონულად გადატანა Azure SQL-ში.

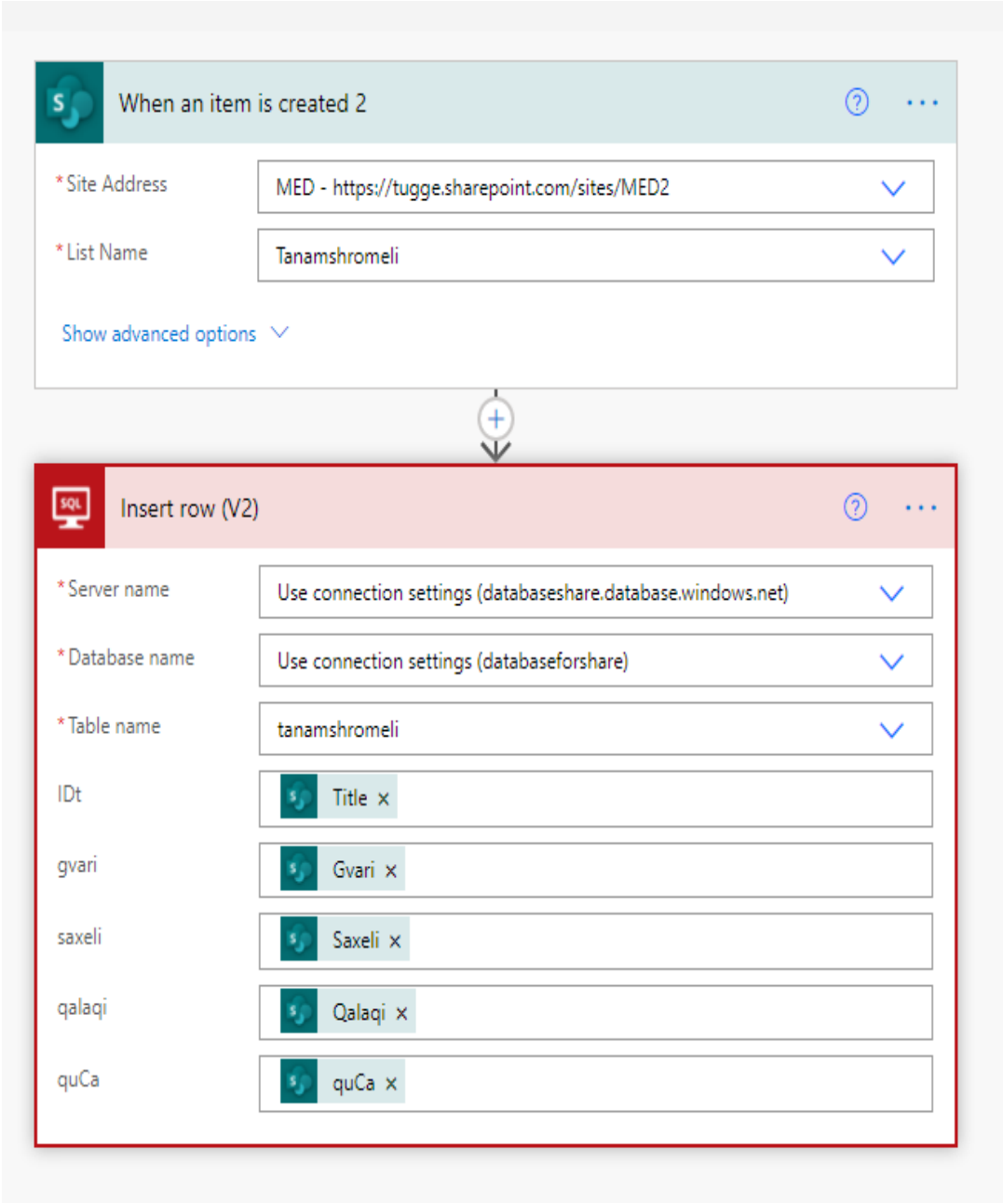


დავამატოთ ახალი კავშირი (ნახ.6.10)

1) Power Automate-ის საშუალებით გავიაროთ ავტორიზაცია;
2) ავირჩიოთ
   Data > **Connections** >**New connection;**
3) ავირჩიოთ **SharePoint**

მოვახდინოთ სამუშაო ნაკადის კონფიგურაცია. 6.11 ნახაზზე ასახული ნაკადი დეტალურად აღწერს Azure SQL-ის tanamshromeli-ის ცხრილის ველების შევსების პროცესს.

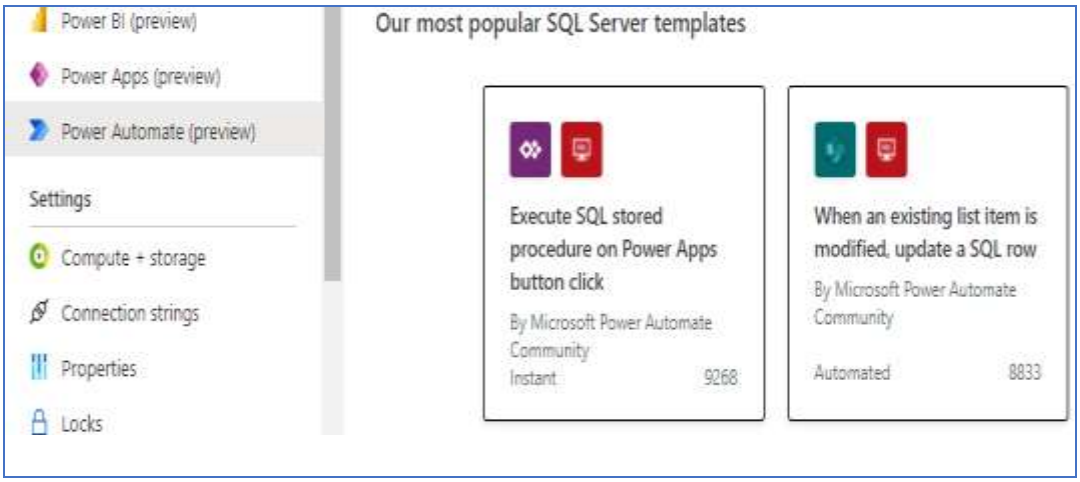
ნახ.6.10


When an item is created 2
*Site Address
MED - https://tugge.sharepoint.com/sites/MED2
*List Name
Tanamshromeli
Show advanced options
Insert row (V2)
*Server name
Use connection settings (databaseshare.database.windows.net)
*Database name
Use connection settings (databaseforshare)
*Table name
tanamshromeli
IDt
Title
gvari
Gvari
saxeli
Saxeli
qalaqi
Qalaqi
quCa
quCa

ნახ.6.11

6.3. მოვლენის მონიტორინგი Power Automate-ის საშუალებით

Power Automate-ში ხელმისაწვდომია მრავალი ჩაშენებული შაბლონი, ავირჩიოთ Send an email when an item is created in SQL Server შაბლონი (ნახ.6.12).



ნახ.6.12. Azure SQL-ის დაკავშირება Power Automate -თან

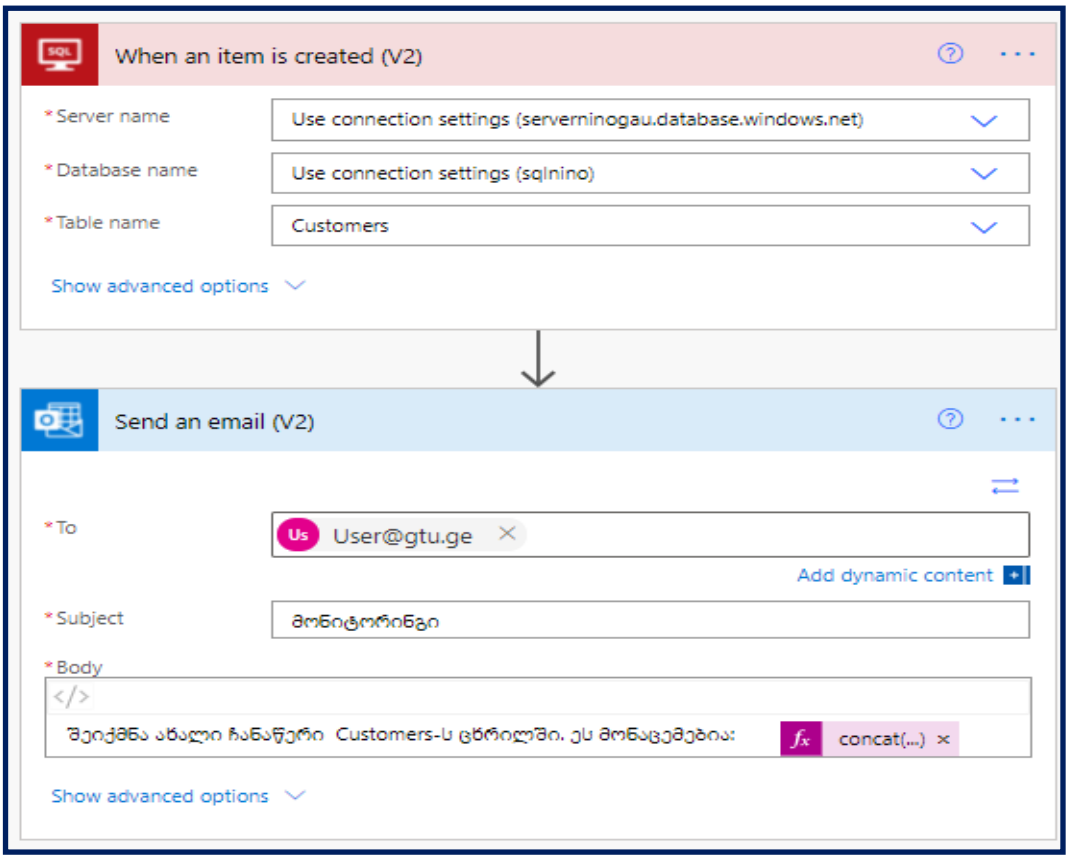
შევავსოთ აღნიშნული ბლოკები საჭირო ინფორმაციით, კერძოდ მივუთითოთ სერვერის, მონაცემთა ბაზის და ცხრილის სახელწოდება (ნახ.6.13).

შევამოწმოთ მეილი, Outlook-ის საშუალებით. მივიღეთ (ნახ.6.14).

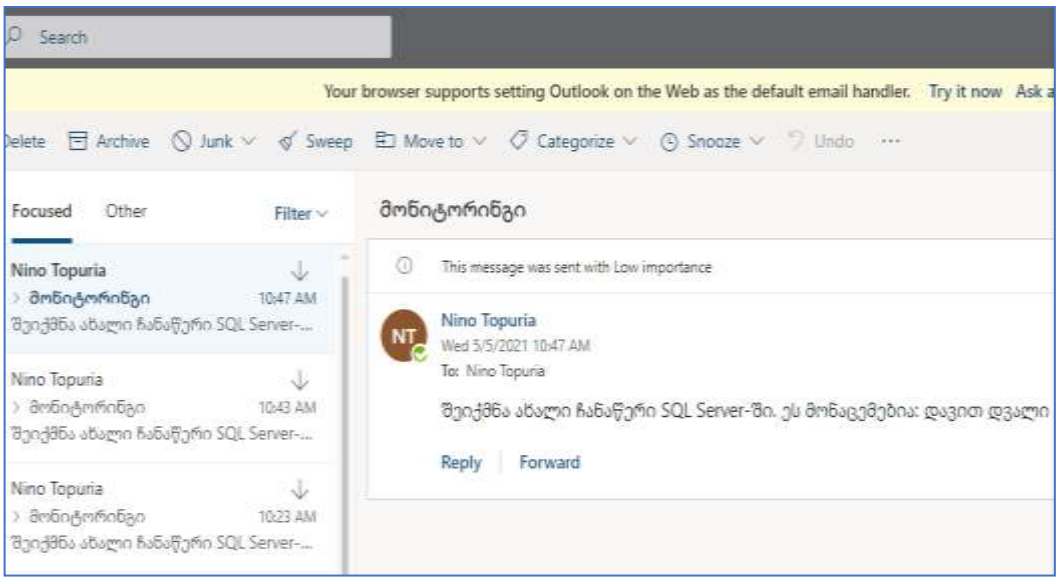
შემდეგი მაგალითის სახით განვიხილოთ, როგორ შეიძლება ვაკონტროლოთ მონაცემთა ბაზის ცხრილში კონკრეტული მონაცემი, მაგალითად, გარკვეული ტელეფონის ნომერი.

აღნიშნული ამოცანის გადასაწყვეტად გამოვიყენოთ Notify about rows in a SQL DB შაბლონი.

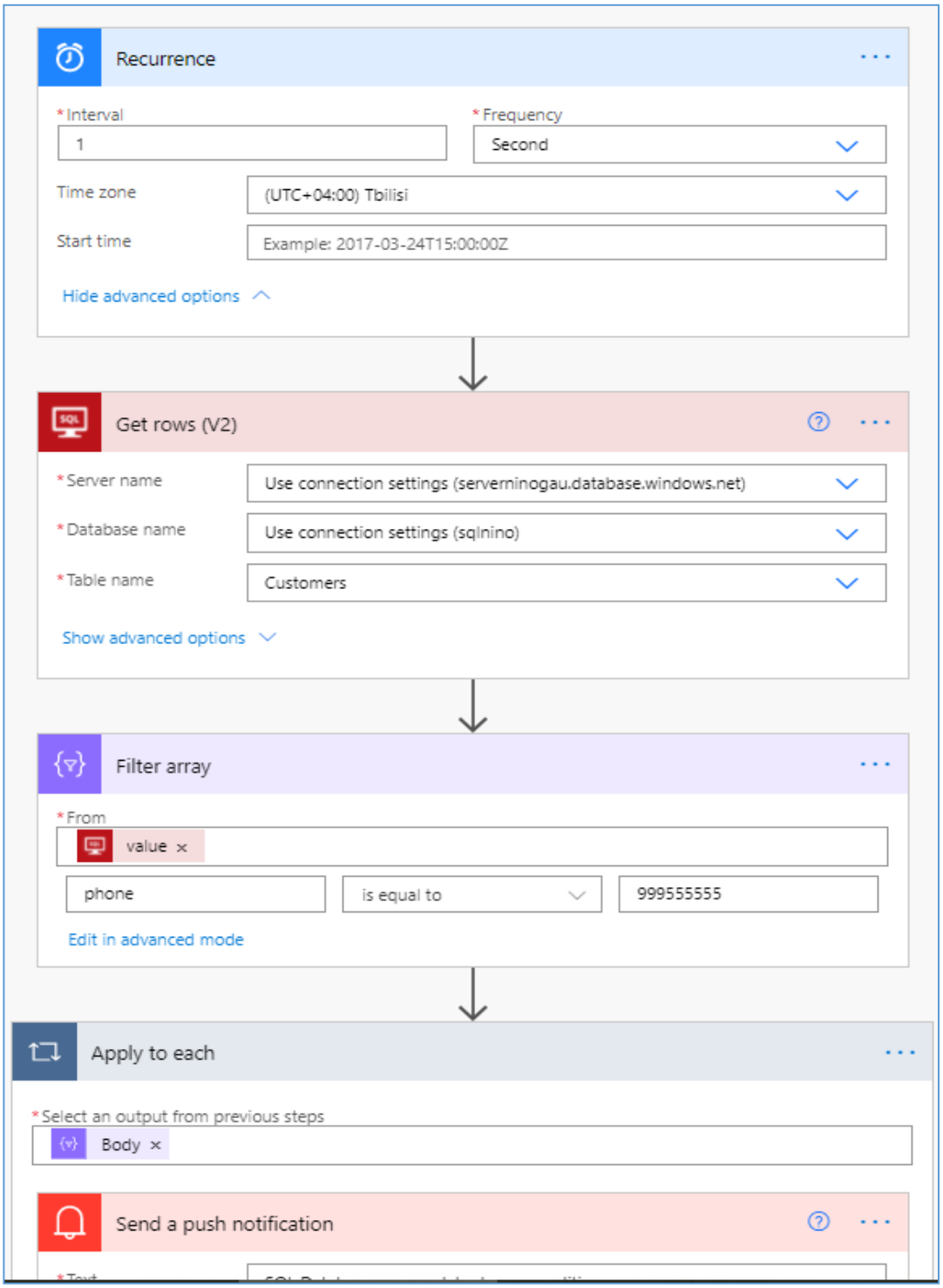
მოცემულ მაგალითში მონიტორინგისთვის არჩეულია ტელეფონის ნომერი (ნახ.6.15).



ნახ.6.13. სამუშაო ნაკადი Azure SQL-ში ჩანაწერების მონიტორინგი



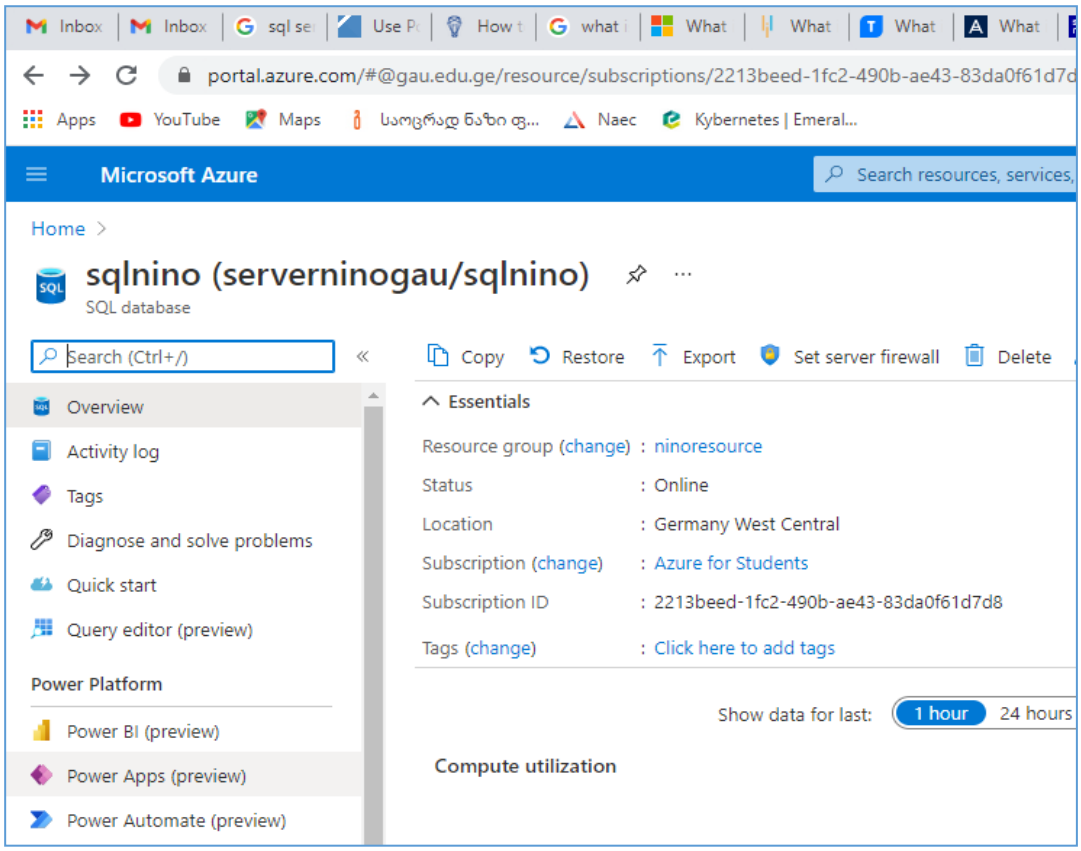
ნახ.6.14. შეტყობინების მიღება Outlook-ში



ნახ.6.15. რეკურენტული სამუშაო პროცესი კონკრეტული მონაცემის კონტროლი

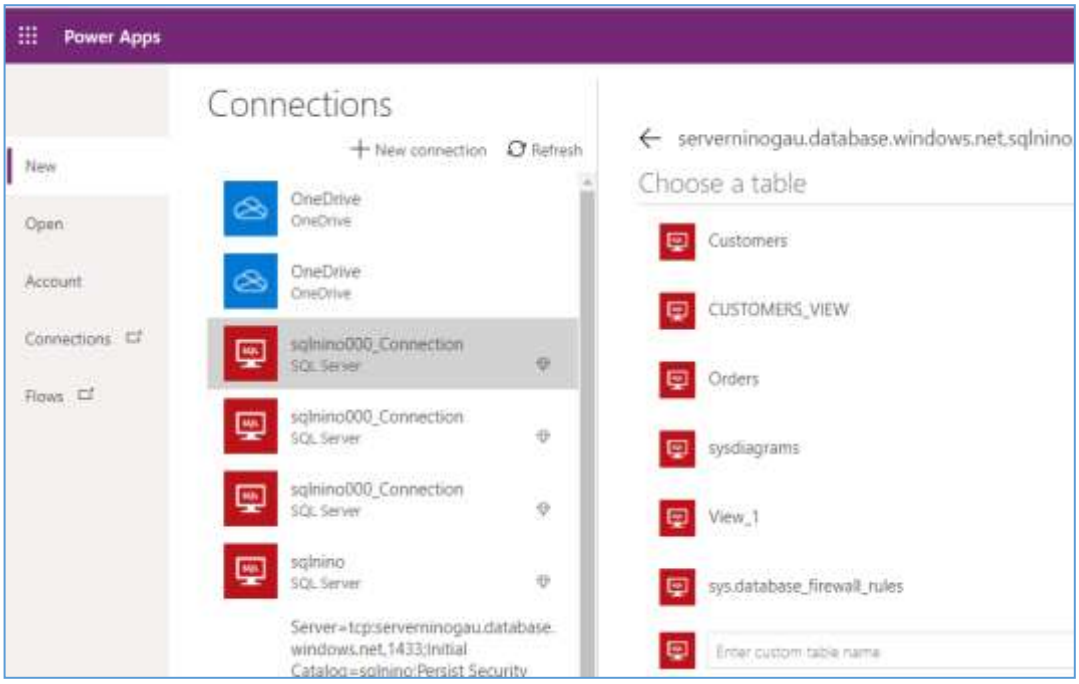
6.4. მობილური აპლიკაციის შექმნა მონაცემთა შესატანად AzureSQL-ში

შევქნათ მობილური აპლიკაცია, რომლის საშუალებითაც შესაძლებელი იქნება ჩანაწერის ჩამატება Azure Sql-ში დაპროექტებულ მონაცემთა ბაზის ერთ-ერთ ცხრილში, მაგალითად, ავირჩიოთ ცხრილი Customers. აღნიშნული დავალების შესრულება შესაძლებელია Power Platform-ის ერთ-ერთ შემადგენელი აპლიკაციით Power App, რომელიც შედის Office 365-ის შემადგენლობაში და ასევე ჩაშენებულია Azure Sql-ში (ნახ.6.16).



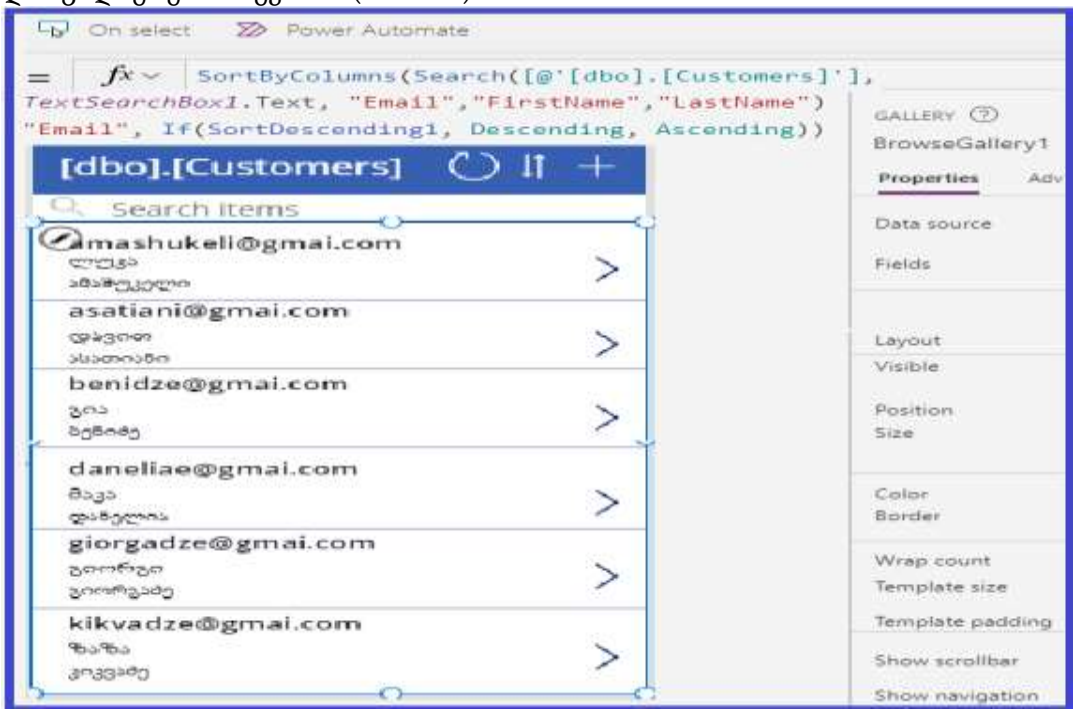
ნახ.6.16. Power App-სერვისის გააქტიურება

Power Apps-ის დიალოგურ ფანჯარაში ავირჩიოთ See your apps ღილაკი. შემდეგ ავირჩიოთ ბრძანება Apps→ New app → Canvas და შევარჩიოთ SQL Server Phone layout შაბლონი, ხოლო Power Apps Connections ფანჯარაში ავირჩიოთ Customers ცხრილი (ნახ.6.17).



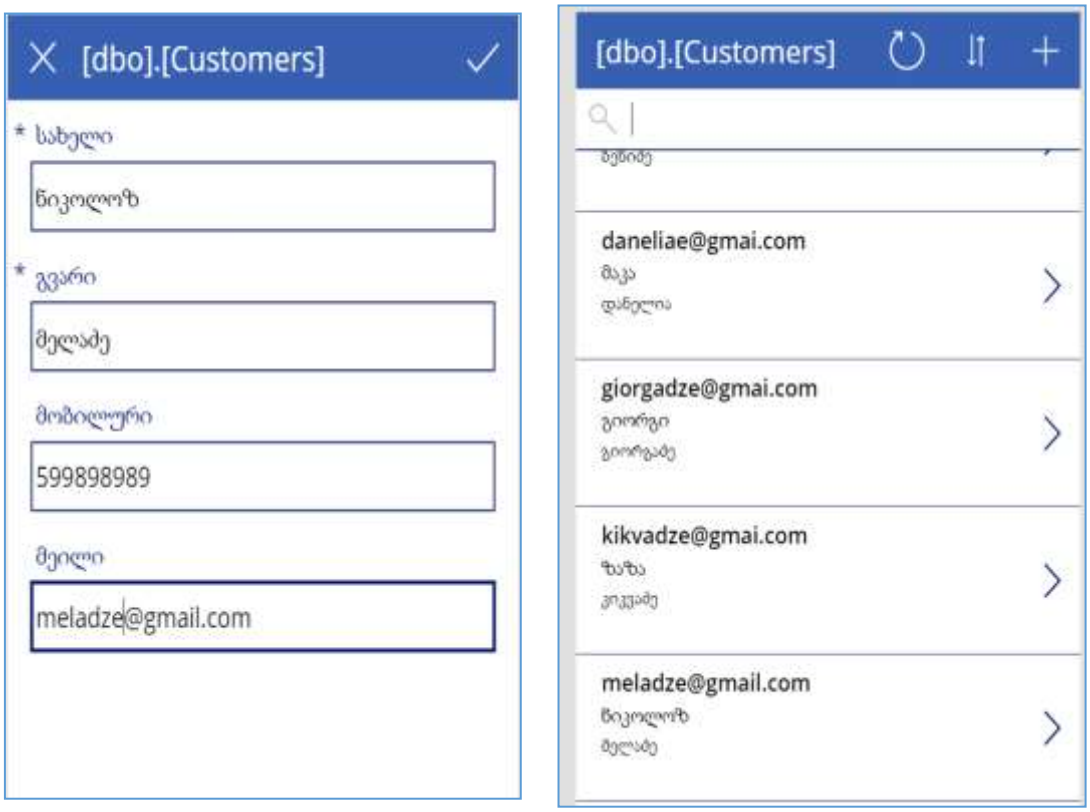
ნახ.6.17. Power Apps -სა და Azure SQL-ს შორის კავშირის შექმნა

ვნახოთ როგორი იქნება ჩვენი მობილური აპლიკაცია წინასწარი დათვალიერების რეჟიმში (ნახ.6.18).



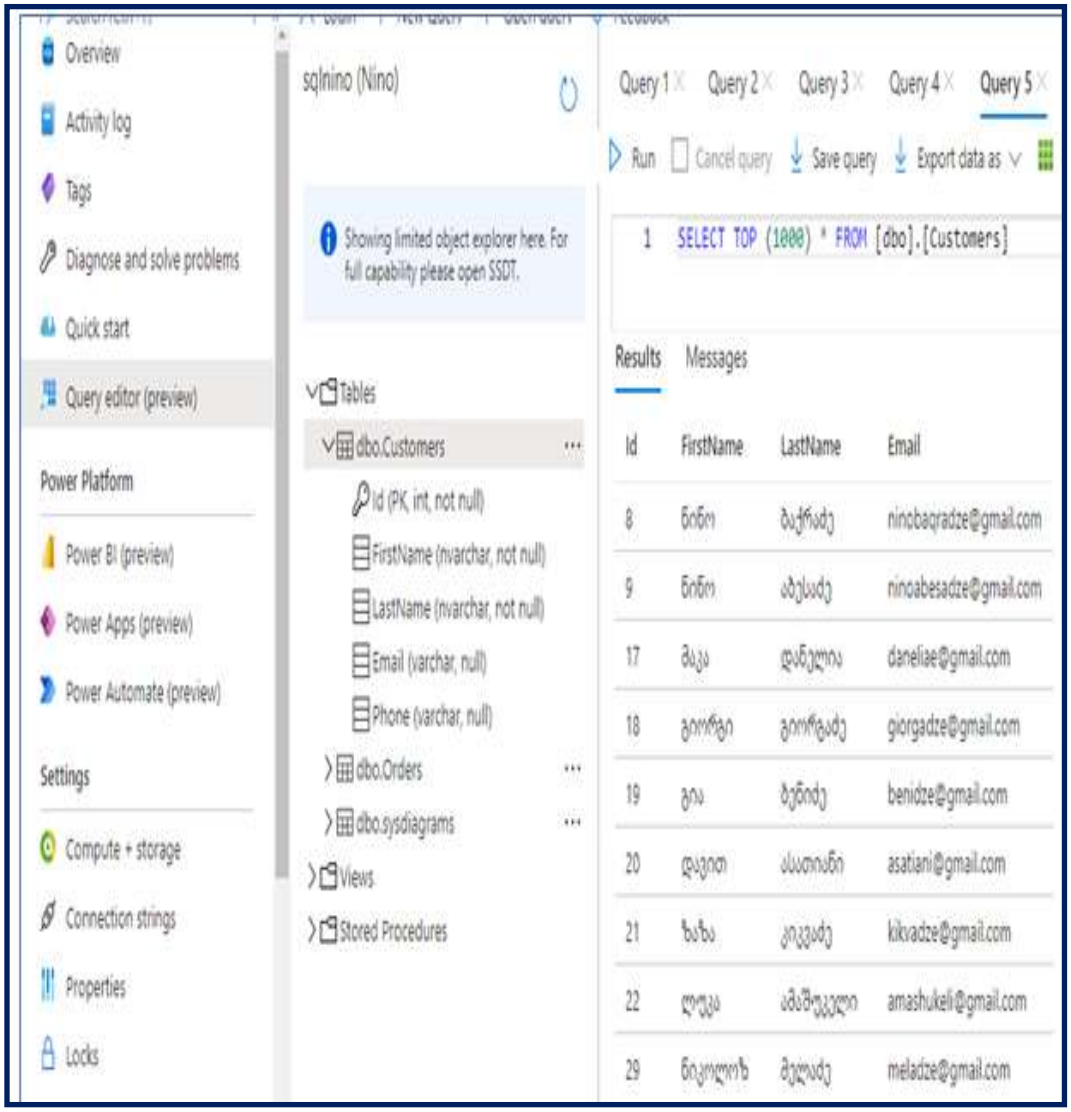
ნახ.6.18. მობილური აპლიკაცია წინასწარი დათვალიერების რეჟიმში

აქვე შეგვიძლია შევიტანოთ გარკვეული ცვლილებები, შევცვალოთ ფონი, ასოების ფერი, დავადოთ ხმა და ა.შ. სატესტოდ შევიტანოთ ინფორმაცია ნიკოლოზ მელაძის შესახებ (ნახ.6.19).



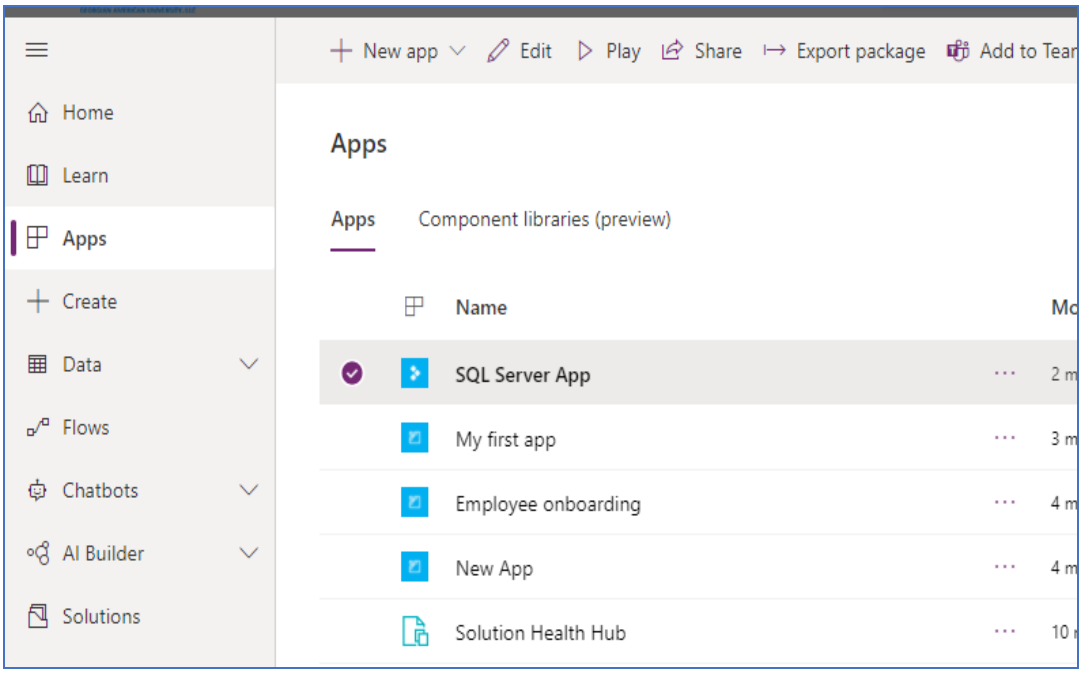
ნახ.6.19. მობილური აპლიკაციის მუშაობის შემოწმების პროცესი

შევამოწმოთ, იმუშავა თუ არა მობილურმა აპლიკაციამ. კვლავ გავააქტიუროთ Azure SQL. ნახაზზე ჩანაწერი (ID=29) ჩამატებულია (ნახ.6.20).



ნახ.6.20. ცხრილში ბოლოს ჩამატებული ჩანაწერის ასახვა

მობილური აპლიკაცია წარმატებით მუშაობს. დავიმახსოვროთ იგი, Save ბრძანებით (ნახ.6.21).



ნახ.6.21. SQL Server App - მზა მობილური აპლიკაცია

ნახაზზე ჩანს, რომ SQL Server App, მზადაა გამოსაყენებლად და შეგვიძლია ჩამოვტვირთოთ სასურველი ვერსია (ნახ.6.22).

ნახ.6.22. სხადასხვა მობილური მოწყობილობისთვის მობილური აპლიკაციის ჩამოტვირთვის ფანჯარა

6.5. ხელოვნური ინტელექტის მოდელების ჩანერგვა მობილურ აპლიკაციაში

AI Builder არის Microsoft Power Platform-ის შესაძლებლობა, საიდანაც ხდება სხადასხვა ტიპის ხელოვნური ინტელექტის მზა მოდელების „აღება“ და მათი ჩანერგა Power Apps-სა და Power Automate-ში.

AI Builder საშუალებას იძლევა გამოივიყენოთ ხელოვნური ინტელექტი ბიზნეს პროცესების ავტომატიზაციისთვის. შესაძლებელია შევქმნათ ორგანიზაციის საჭიროებებზე მორგებული მოდელები ან ავირჩიოთ წინასწარ აშენებული მოდელი, რომელიც მზად არის გამოსაყენებლად მრავალი ჩვეულებრივი ბიზნეს სცენარისთვის.

განვიხილოთ Business card reader-ის გამოყენების მაგალითი მობილურ აპლიკაციაში. დავსახოთ ამოცანა. საქმიან შეხვედრაზე ხშირად მონაწილეები ცვლიან თავიანთ სავიზიტო ბარათებს. უფრო მოხერხებულია გადავუღოთ სურათი სავიზიტო ბარათს, საიდანაც ამოკითხული ინფორმაცია ავტომატურად აისახება SharePoint-ის სიაში ან Dataverse მონაცემთა ბაზაში ან Outlook-ის კონქატებში..

Dataverse შედის Office 365-ის შემადგენლობაში. იგი არსებითად არის ონლაინ მონაცემთა ბაზა ან მონაცემთა საწყობი (data warehouse), რომელიც ინახავს მონაცემებს მრავალი წყაროდან. შემდეგ ამ მონაცემებზე წვდომა ან სხვა აპლიკაციებში გადატანა შესაძლებელია ანალიზისა და შემდგომი გამოყენებისთვის. Dataverse არის ახალი ტიპის რელაციური მონაცემთა ბაზა, რომელიც ინახავს თავის მონაცემებს ცხრილებში (table) ან არსებში (entities).

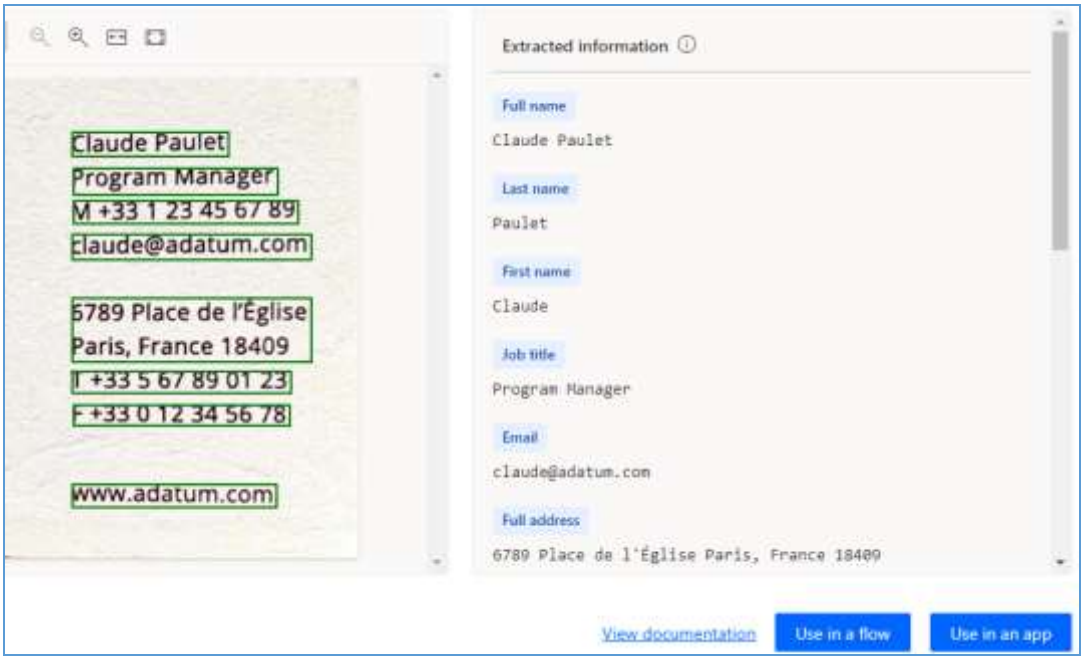
მობილური აპლიკაციის დასაპროექტებლად ავირჩიოთ Power App აპლიკაცია, Insert ბრძანებით ავირჩიოთ Business card reader-ის მოდელი.

6.23 ნახაზზე ჩანს რომ შესაძლებელია ავირჩიოთ აღნიშნული მოდელის გამოყენების ორი ვარაინტი:

Use in a flow - გამოვიყენოთ როგორც ნაკადი

Use in an app - გამოვიყენოთ მობილური აპლიკაციაში.

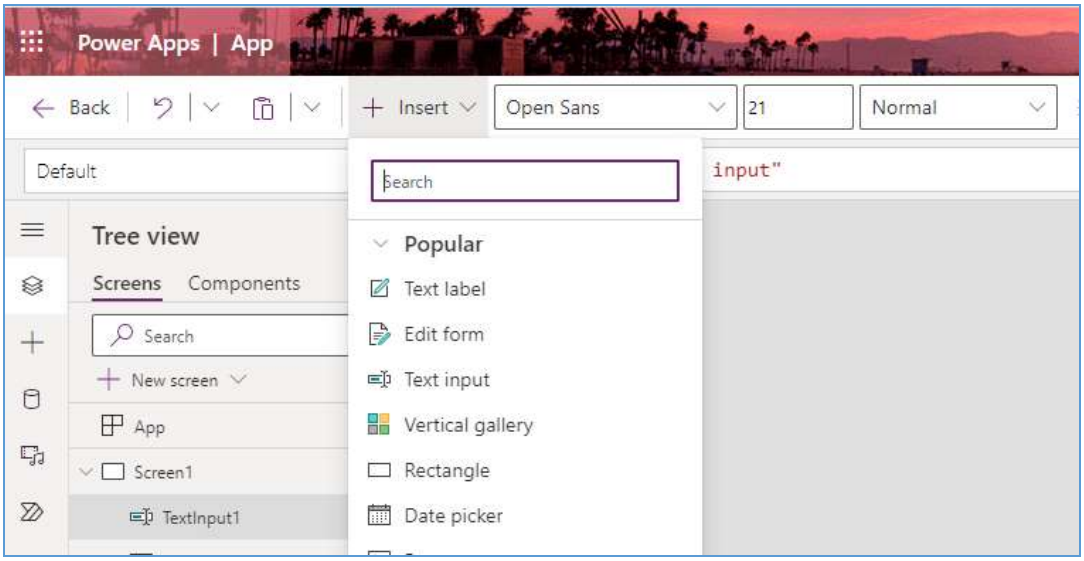
ავირჩიოთ მობილურ აპლიკაციაში გამოყენების შესაძლებლობა.



ნახ.6.23

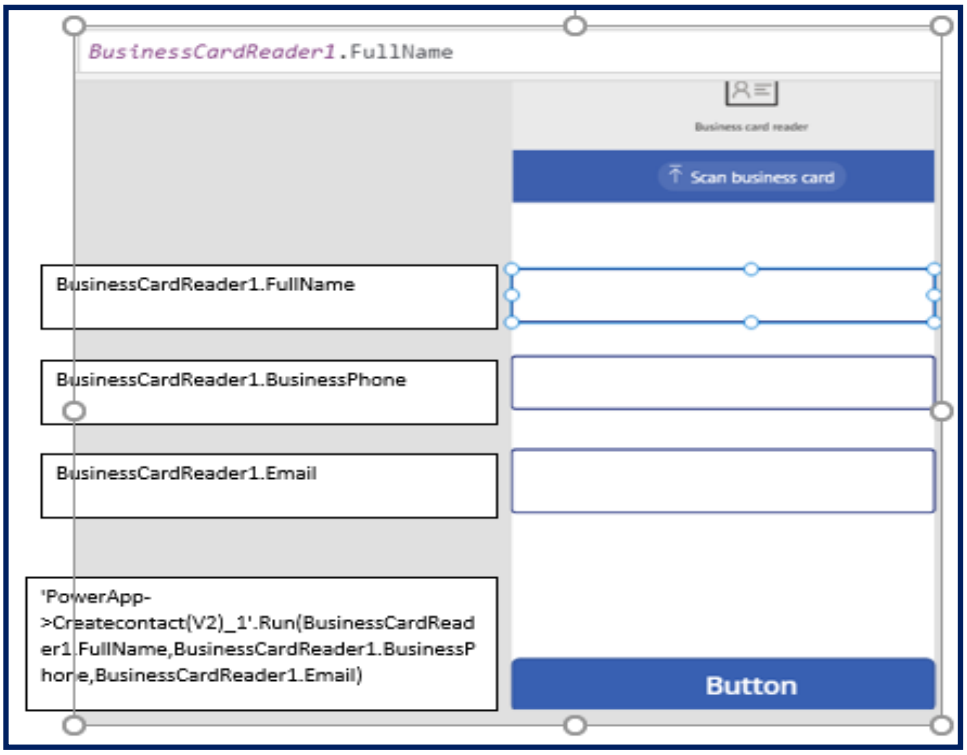
ჩავსვათ მობილურ აპლიკაციაში, ველები რომელთა დამახსოვრებასაც ვაპირებთ. ავირჩიოთ ბრძანება (ნახ.6.24):

Insert → Text Label.



ნახ.6.24

თითოეული ასეთი ველისთვის უნდა მოხდეს ხელოვნური ინტელექტის მოდელის გამოძახება (ნახ.6.25).

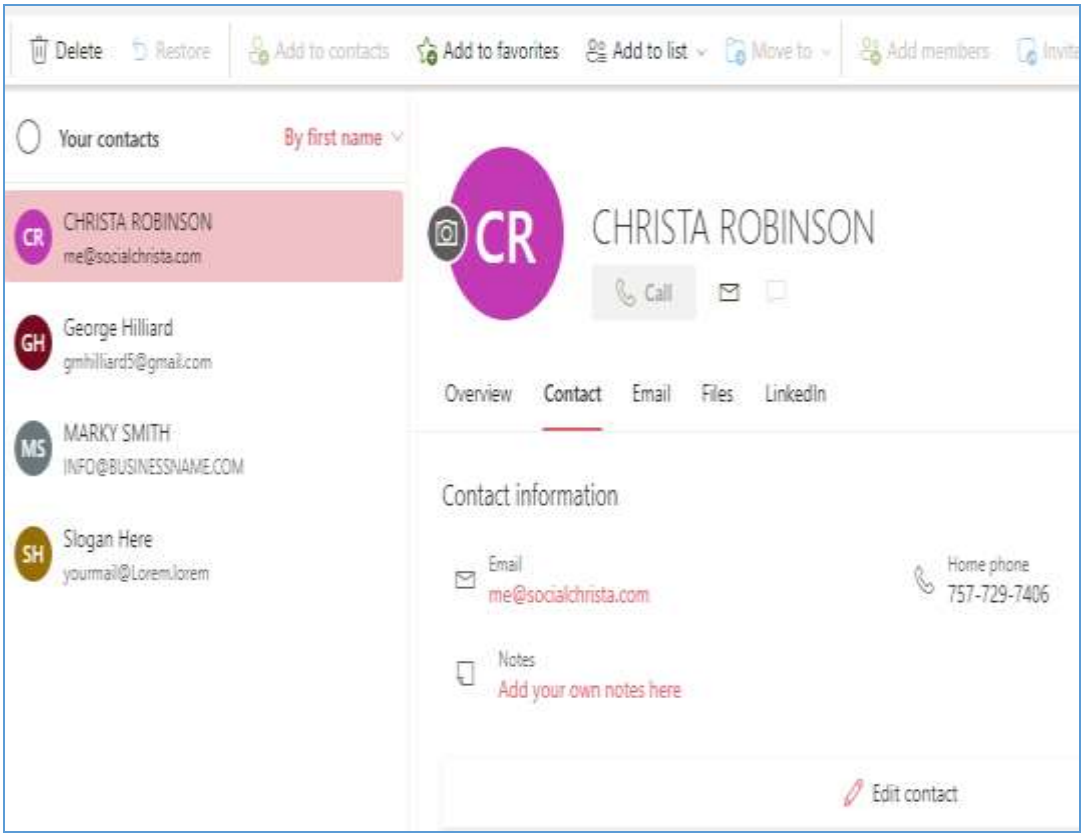


ნახ.6.25

მობილური აპლიკაციის მუშაობის სატესტო რეჟიმში გაშვება შესაძლებელია Preview the App ბრძანებით ან F5 ღილაკით. მზა აპლიკაციას აქვს 6.26 ნახაზზე ნაჩვენები სახე.

ნახ.6.26

აღნიშნული აპლიკაციაში Button ღილაკზე დაჭერის შემდეგ მოხდება სახელის, გვარის, ტელეფონის ნომრის და მეილის ავტომატურად ჩაწერა Outlook-ის კონტაქტებში (ნახ.6.27).

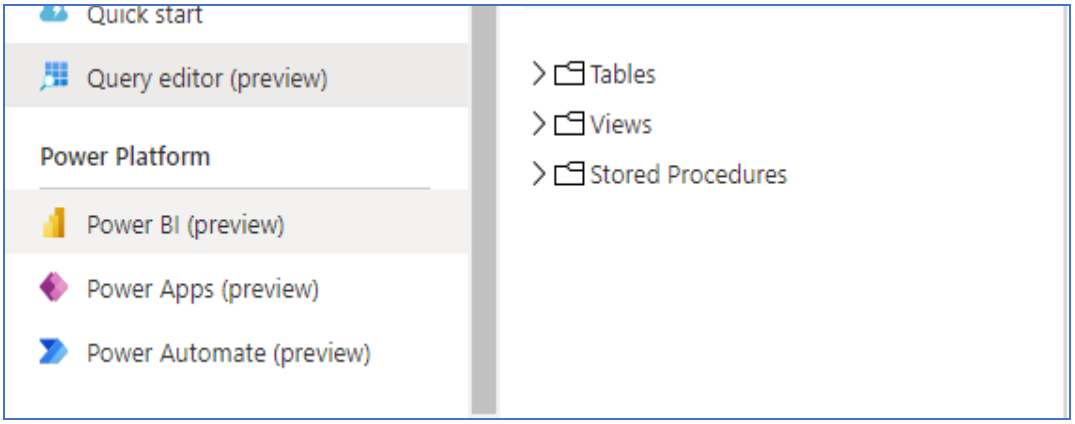


ნახ.6.27

6.6. Azure SQL-ის ბიზნეს ანალიტიკის სერვისი – Power BI Services

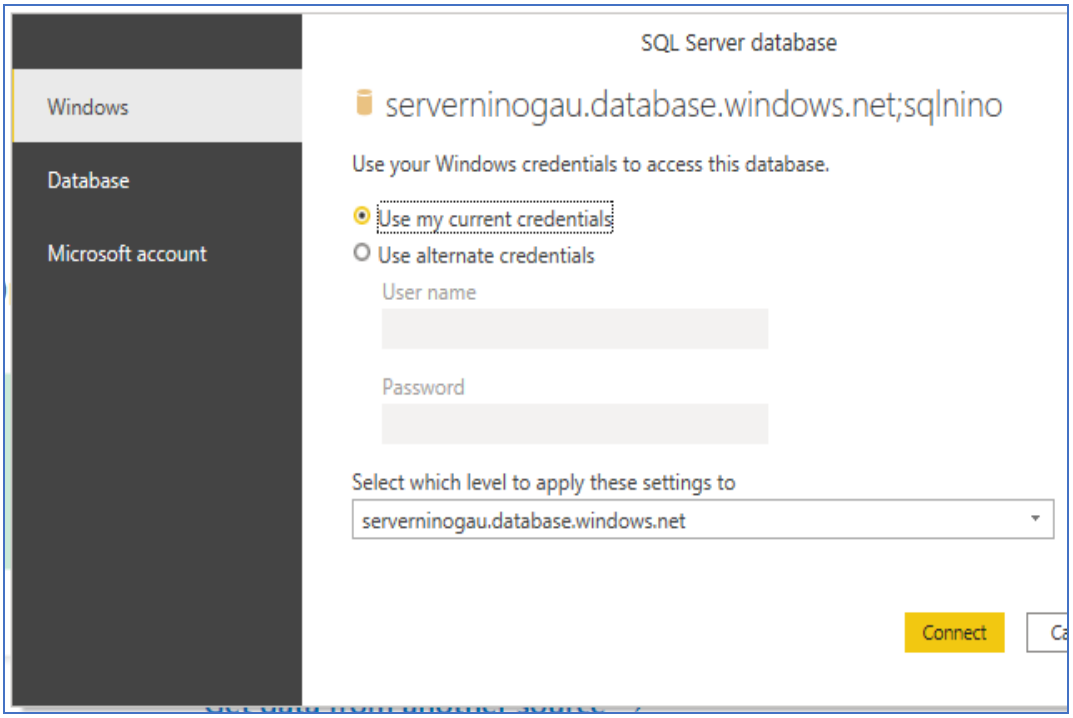
Power BI წარმოადგენს მაიკროსოფტის ღრუბლოვან ბიზნეს ანალიტიკის სერვისს, რომელის საშუალებითაც შესაძლებელია მონაცემების ანალიზი და ვიზუალიზაცია სწრაფად და მაღალი ეფექტურობით. აღსანიშნავია, რომ Power BI Services შეგვიძლია გამოვიყენოთ როგორც Ms SQL Server-ის reporting-ის საშუალება. ამგვარად, ჩვენ შეგვიძლია მარტივად შევქმნათ დიაგრამები, ანგარიშები და დაშბორდები, დიდი მოცულობის მონაცემებთან პირდაპირი დაკავშირების საფუძველზე.

როგორც აღვნიშნეთ, Power BI უშუალოდ ჩაშენებულია Azure SQL-ის სამუშაო გარემოში (ნახ.6.28).

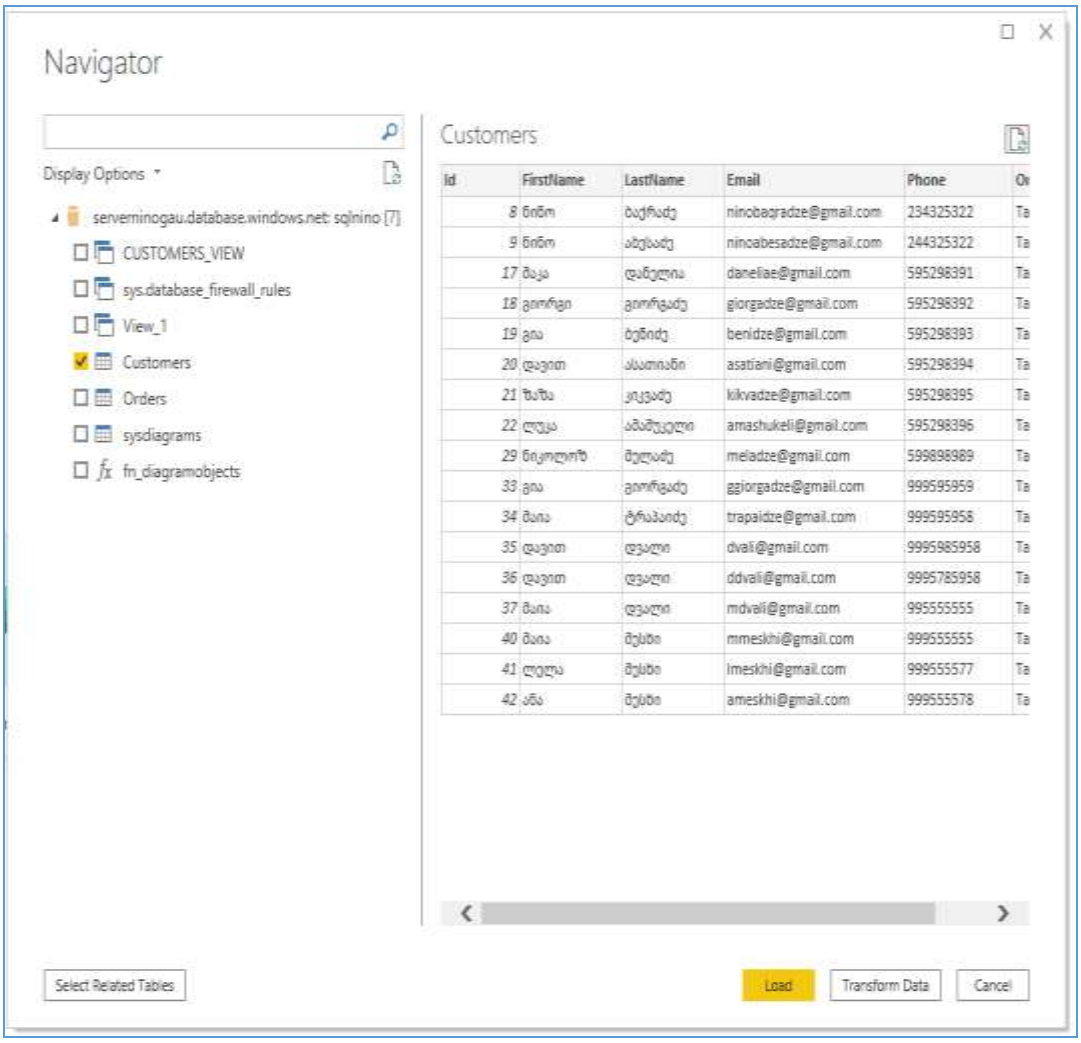


ნახ.6.28. Azure SQL-ის დაკავშირება Power BI-სთან

როგორც კი ავირჩევთ Power BI-ს, ხოლო ეკრანზე გამოსულ დიალოგურ ფანჯარაში Get started ღილაკს, შევძლებთ Azure SQL-ის მონაცემთა ბაზასთან დაკაშირებას (ნახ.6.29, 6.30).

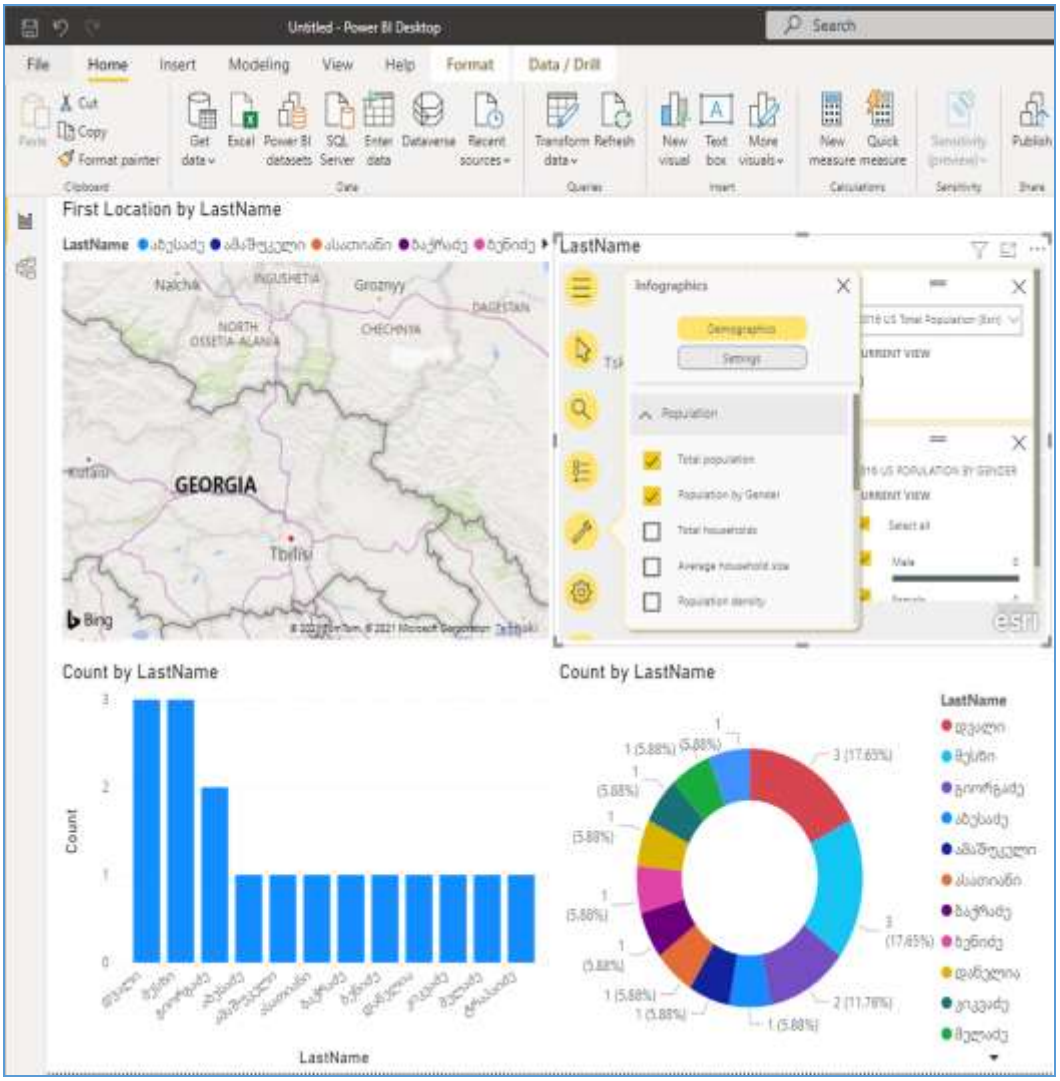


ნახ.6.29. Azure SQL-ის დაკავშირება Power BI-სთან



ნახ.6.30. მონაცემების ჩატვირთვა Power BI-ს გარემოში

ავირჩიოთ Load ღილაკი, გაიხსნება Power BI Desktop-ის სამუშაო გარემო, სადაც შეგვიძლია სხადასხვა ვიზუალების შექმნა და შემდგომ მათი გაზიარება (ნახ.6.31).



ნახ.6.31. Power BI-ის საშუალებით შექმნილის დაშბორდი

ნახაზზე ჩანს ჩაშენებული ArcGIS რუკა, სადაც მონიშნულია დემოგრაფიული მონაცემები. ArcGIS Online არის ღრუბელზე დაფუძნებული რუკების და მონაცემთა ანალიზის გადაწყვეტილება, რომლის საშუალებითაც შესაძლებელია სპეციფიკური რუკების შექმნა მთელი მსოფლიოს მაშტაბით და შემდგომ მათი გაზიარება.

VII თავი
პროგრამული აპლიკაციის ტესტირება და დისტრიბუციული ფაილის შექმნა

7.1. პროგრამული აპლიკაციის ტესტირება

მოდულური ტესტირება (Unit testing) 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 შეცდომების გაჩენამდე პროგრამის უკვე ტესტირებულ ნაწილებში [16,19].

Unit testing ტესტირების ტექნოლოგია განვიხილოთ, მაგალითად, ფინანსური ობიექტის – ბანკის მაგალითზე.

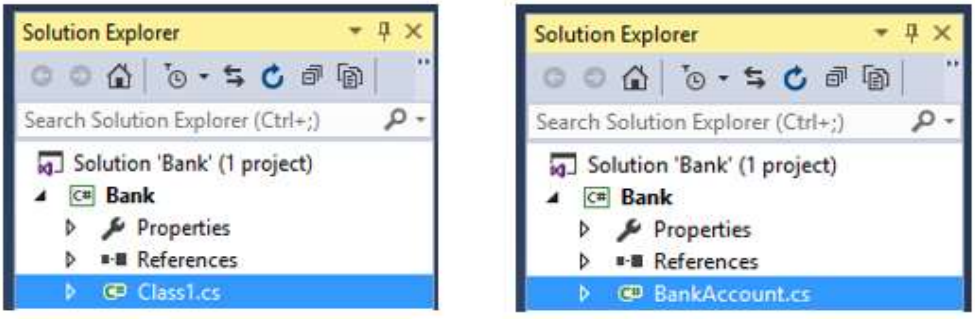
პროცესი რამდენიმე ეტაპს მოიცავს, კერძოდ, იწერება დასატესტი პროგრამა (რაც უნდა შემოწმდეს), შემდეგ სატესტო პროგრამა (რითიც უნდა შემოწმდეს). ხორციელდება ტესტირება და შედეგის მიხედვით გრძელდება პროცესი: იგი სრულდება წარმატებით (ე.ი. მიზნობრივი პროგრამა მზადაა დასანერგად წარმოებაში) ან სრულდება შეცდომებით - ე.ი. საჭიროა პროგრამის შემდგომი დამუშავება.

- დასატესტი პროგრამის პროექტის შექმნა: Visual Studio–ში საჭიროა ავირჩიოთ: **File -> New -> Project.** შედეგად გამოჩნდება დიალოგური ფანჯარა, სადაც ავირჩევთ:

Visual C# => ClassLibrary

პროექტი სახელით Bank.

მიიღება 7.1 ნახაზზე ნაჩვენები Solution Ex[plorer ფანჯარა. აქ Class1.cs სახელი შევცვალოთ BankAccount.cs -ით.



ნახ.7.1. Class1 - > BankAccount

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

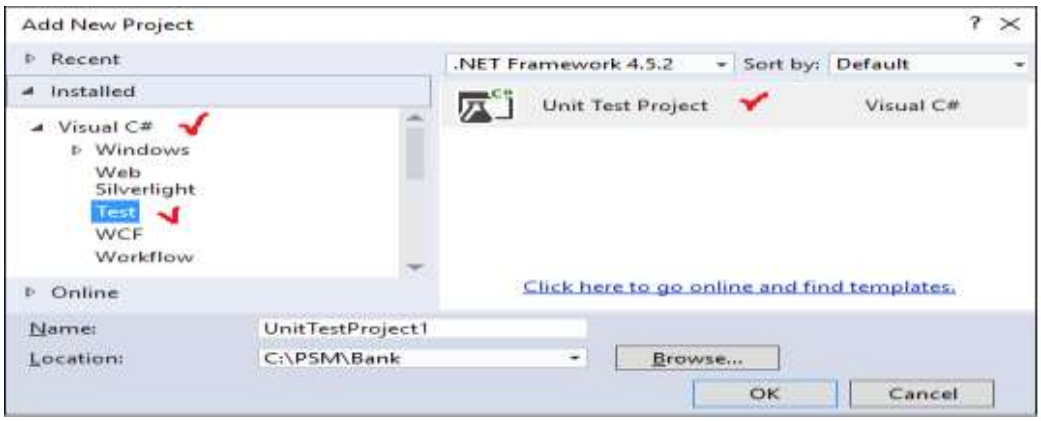
ეს საწყისი ტექსტი, მაგალითად, მოცემულია 7.1 ლისტინგში.

//-- ლისტინგი_7.1 --- 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");
```

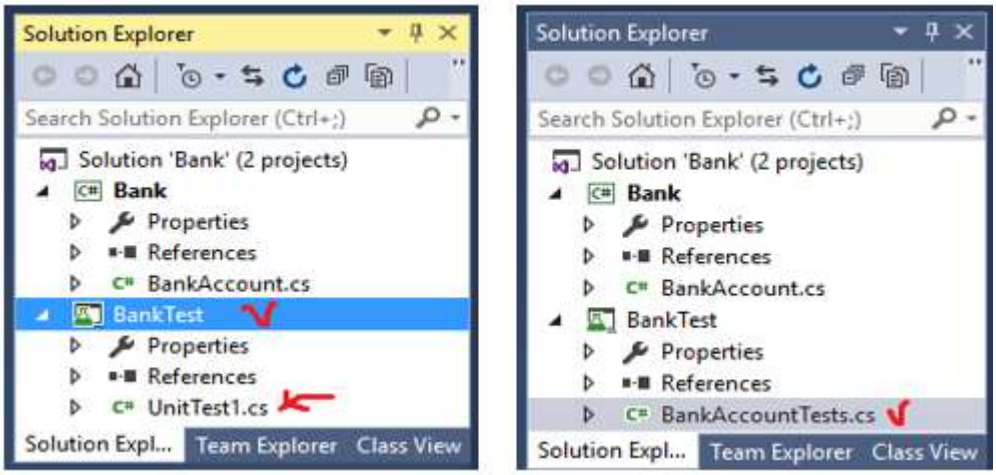
```
        }
        m_balance += amount; // განზრახ არასწორი კოდი
        // m_balance -= amount;   // გასწორებული
     }
    public void Credit(double amount)
     {
       if (m_frozen)
         {
          throw new Exception("Account frozen");
         }
       if (amount < 0)
        {
          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", 11.99);
      ba.Credit(5.77); ba.Debit(11.22);
      Console.WriteLine("Current balance is ${0}", ba.Balance);
     }
  }
}
```

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



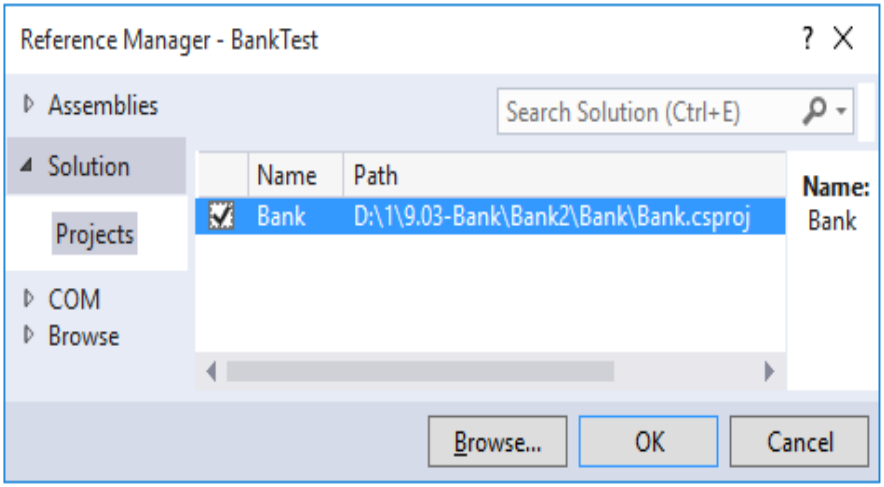
ნახ.7.2. Unit Test პროექტის შექმნა

მივიღებთ 7.2. ნახაზზე ნაჩვენებ სურათს BankTest პროექტით. მარჯვენა სურათზე შეცვლილია UnitTest კლასის სახელი BankAccountTests-ით.



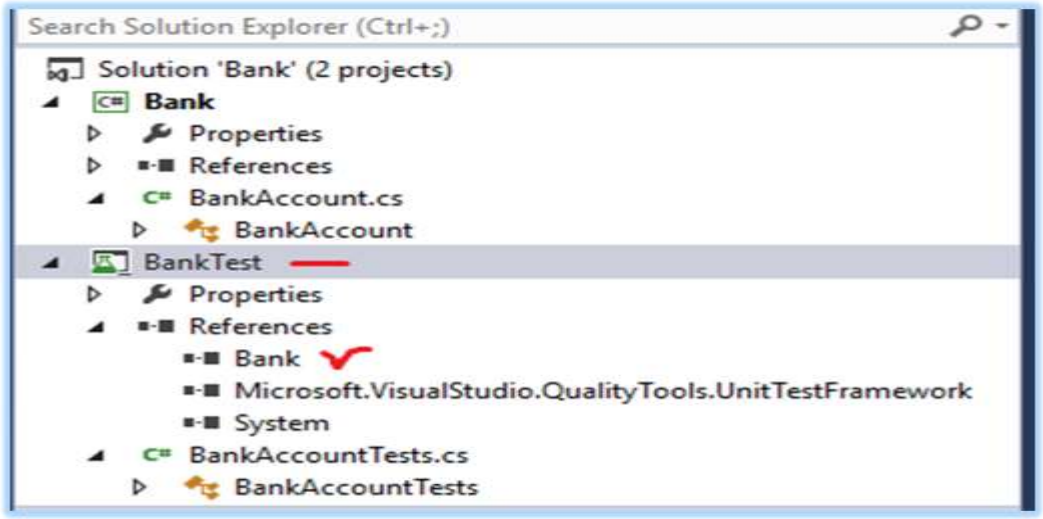
ნახ.7.2

BankTests პროექტში დავამატოთ reference **Bank** solution-იდან. ამისათვის **BankTests**-ზე მაუსის მარჯვენა ღილაკით ავირჩიოთ **Add Reference** და მივიღებთ 7.3 ნახაზზე ნაჩვენებ ფანჯარას. აქ Solution სტრიქონში ვირჩევთ Projects და Bank-ის ჩეკბოქსს მოვნიშნავთ.



ნახ.7.3

მივიღებთ 7.4 ნახაზზე ნაჩვენებ შედეგს.



ნახ.7.4

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

ამგვარად, BankAccountTests.cs ფაილს ექნება 7.2 ლისტინგზე ნაჩვენები სახე.

```
//-- ლისტინგი_7.2 -----  BankAccountTests.cs -----
using System;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using BankAccountNS;

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 მნიშვნელობისათვის (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 = 11.99;
   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.

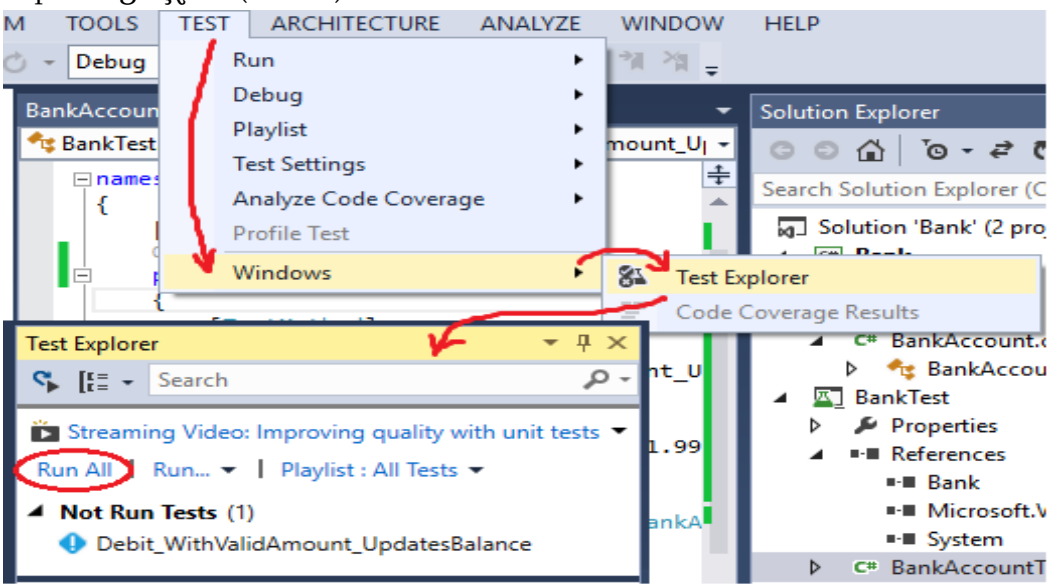
ტესტის აგება და ამუშავება:
მთლიანი ტესტის კოდი მოცემულია 7.3 ლისტინგში.

//-- ლისტინგი_7.3 ----- BankAccountyTests.cs ----

```
using System;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using BankAccountNS;
namespace UnitTestProject1
{ [TestClass]
  public class BankAccountTests
  {  [TestMethod]
     public void Debit_WithValidAmount_UpdatesBalance()
     {
       // arrange
       double beginningBalance = 11.99;
       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");
     }
  }
}
```

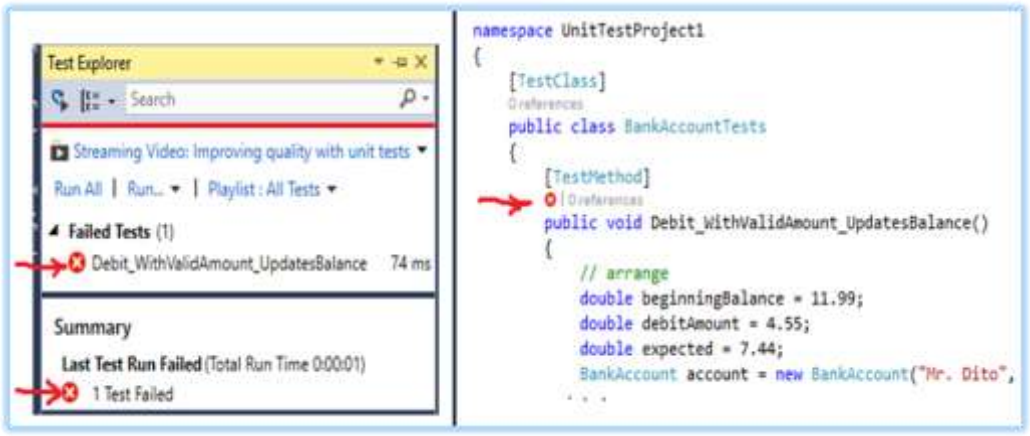
- BUILD მენიუდან ვირჩევთ Build Solution;

- TEST მენიუდან ვირჩევთ Windows და Test Explorer პუნქტებს. იხსნება Test Explorer ფანჯარა (ნახ.7.5).



ნახ.7.5

აქ ვირჩევთ Run All - ს და ვიღებთ შედეგს (ნახ.7.6).



ნახ.7.6

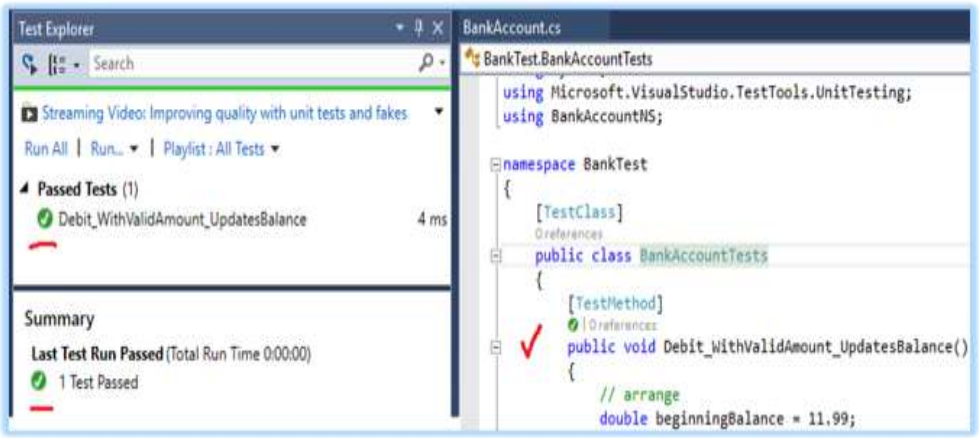
ნახაზზე მითითებული „x“-სიმბოლოები წითელ წრეშია მოთავსებული, ე.ი. ტესტირებამ აღმოაჩინა შეცდომები და კოდი წარმატებით ვერ შესრულდა.

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.7.7).



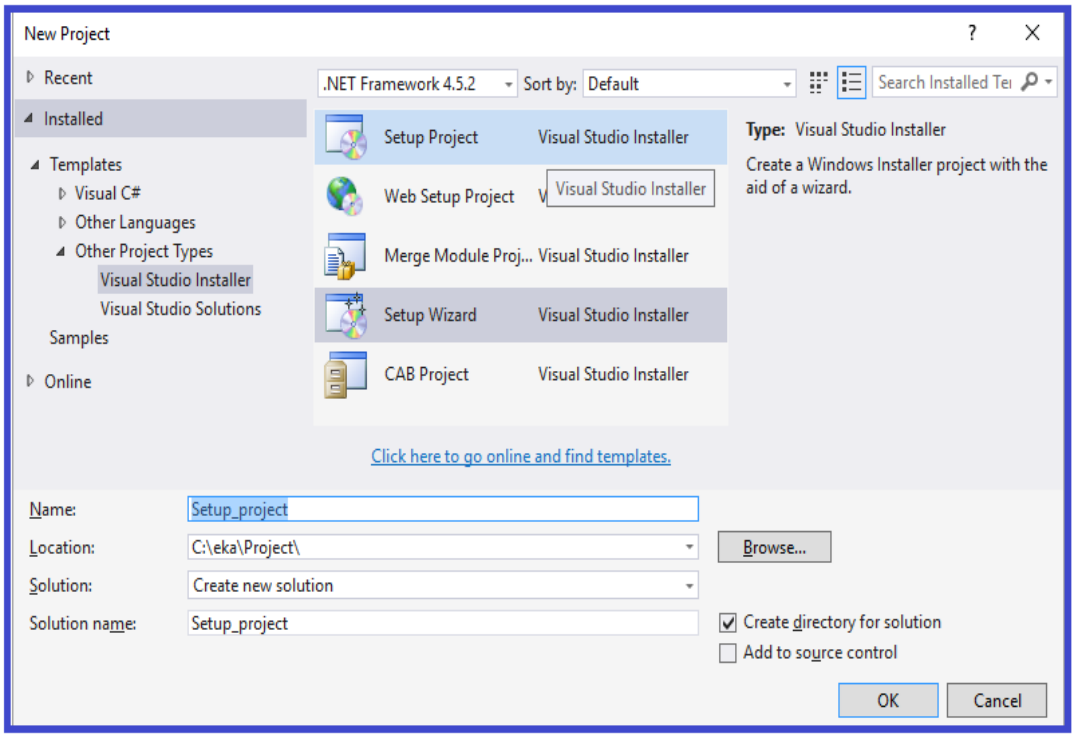
ნახ.7.7. სწორი შედეგი

7.2. პროგრამული აპლიკაციის საინსტალაციო ფაილის შექმნა

პროგრამული აპლიკაციების პროდუქტად ქცევა და მისი ბაზარზე გატანა მოითხოვს საერთაშორისო სტანდარტებით გაფორმებას. ამჯერად ჩვენი მიზანია Setup და Deployment პროექტში რეალიზებული პროგრამული პაკეტის ინტეგრირება. რეალიზებული პროგრამული პაკეტის გამშვები (exe) ფაილის შექმნა.

ამოცანა: პროგრამული აპლიკაციის გამშვები ფაილის მომზადება

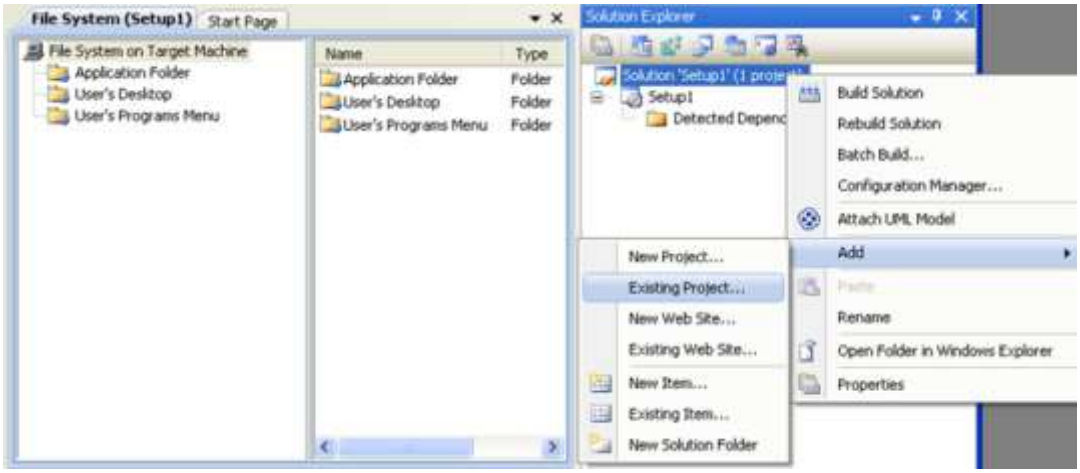
Visual Studio სისტემაში ახალი პროექტის შექმნის მენიუში Other Project Type-Visual Studio Installer პროგრამული პაკეტის გამშვები (exe) ფაილის შექმნისათვის (ნახ.7.8) არის ორი არჩევანი Setup Project და Setup Wizard. მიუხედავად ამისა, მათი ფუნქციონირება მცირედით განსხვავდება.



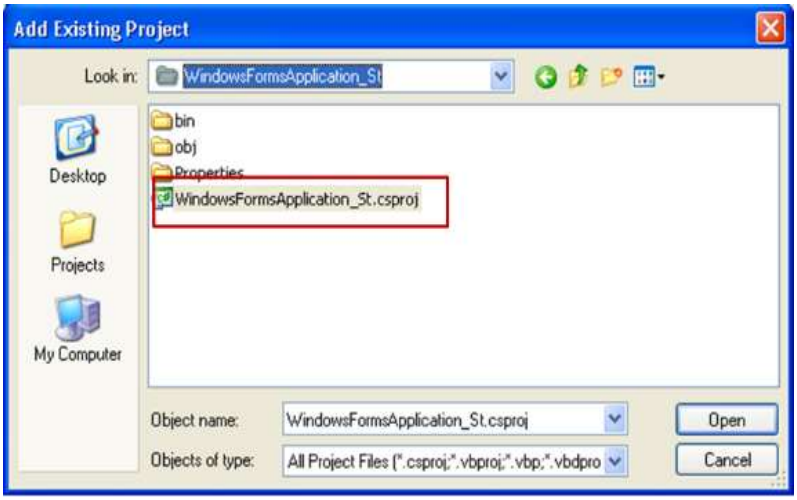
ნახ.7.8

Setup Wizard საშუალებას იძლევა ბიჯური რეჟიმით შეიქმნას Web ან Windows ფორმის საინსტალაციო გარემო და ინტეგრირდეს სისტემისათვის თანამდევი ფაილებით. თუმცა გამშვები ფაილის მომზადების ბირთვის ავტომატიზებულ მექანიზმებს Setup Wizard საშუალება არ შეიცავს.

Setup Project ან Setup Wizard საშუალებებით პროექტის შექმნის შემდეგ, პროექტში უნდა ინტეგრირდეს რეალიზებული პროგრამული პაკეტი ფუნქციით Add – Existing Project (მაუსის მარჯვენა ღილაკი Solution Explorer ფანჯარაში. მიეთითება ფაილი გაფართოებით - .csproj (ნახ. 7.9, 7.10).

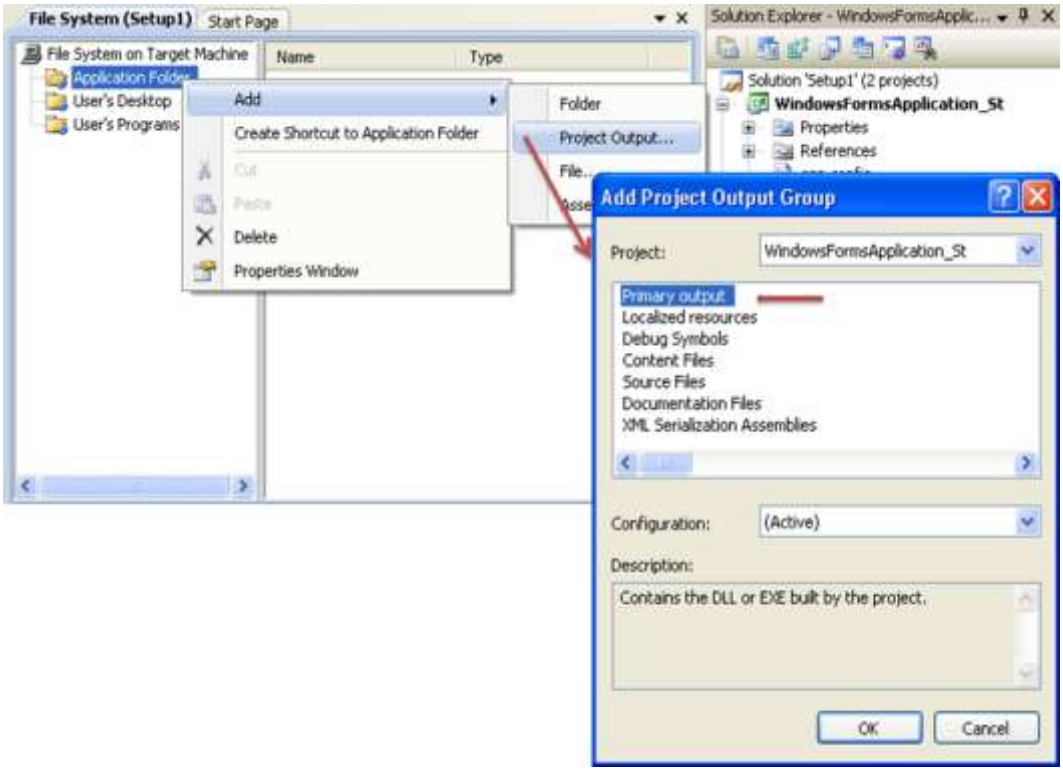


ნახ.7.9

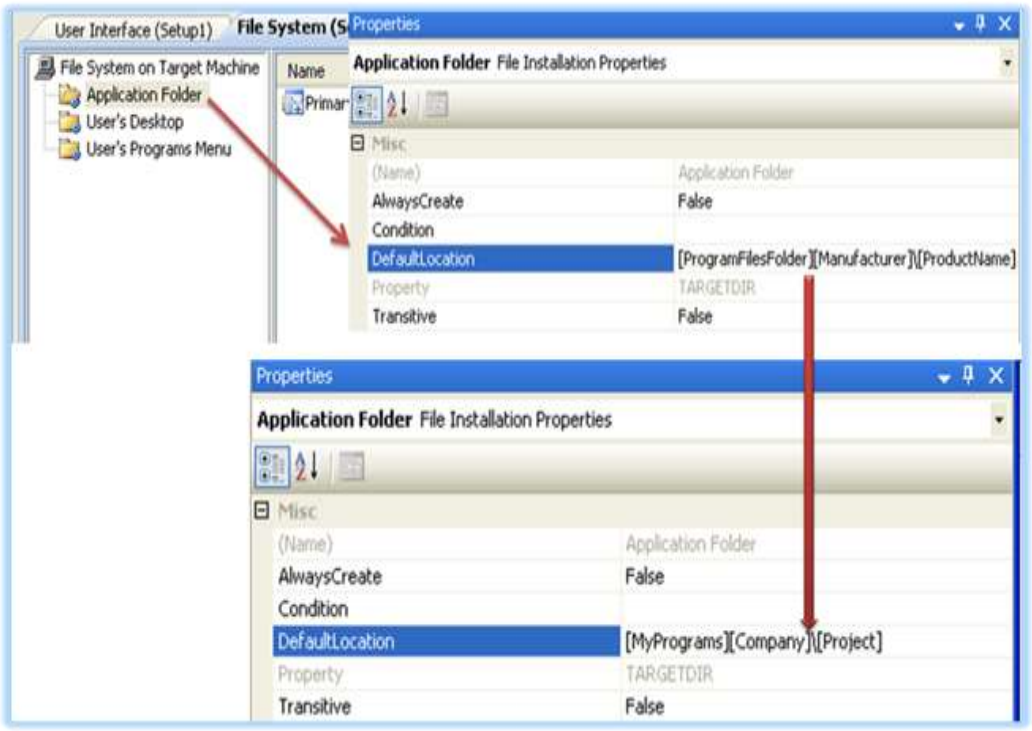


ნახ.7.10

პროექტის ნაწილში File System საქაღალდე Application Folder განკუთვნილია საინსტალაციო პაკეტისადთვის. ამ საქაღალდეზე ფუნქციით Add (მაუსის მარჯვენა ღილაკი) იხსნება პროექტის შედეგების დასაპროექტებელი ფანჯარა, სადაც რეკომენდებულია ბაზური შედეგის - Primer output მითითება (ნახ.7.11). Application Folder საქაღალდეს თვისებების ფანჯარაში შესაძლებელია დასაინსტალირებელი პაკეტის სახელის მითითებაც (ნახ.7.12).



ნახ.7.11



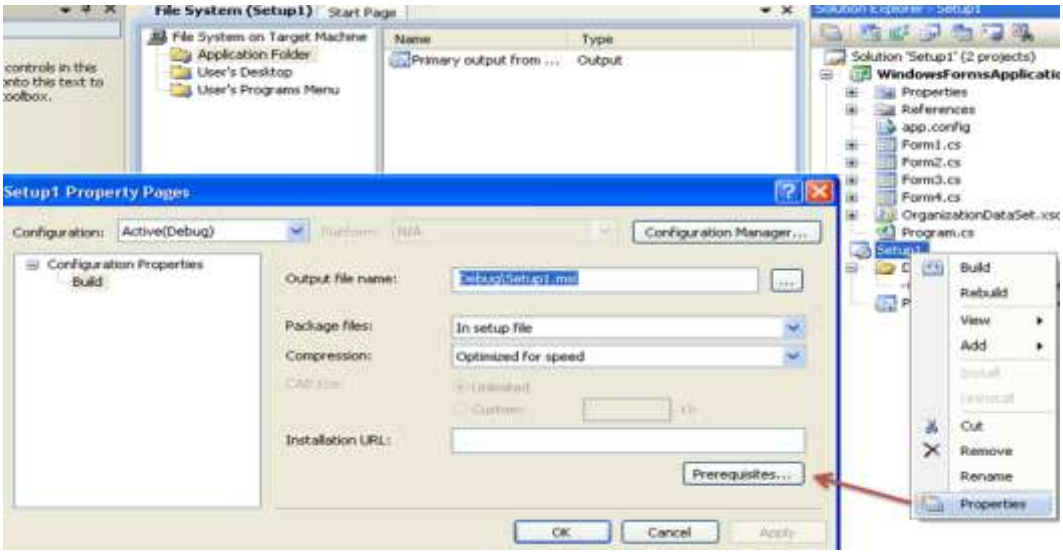
ნახ.7.12

სისტემის ინსტალირების დროს, ხშირად საჭიროა დამხმარე პროგრამების, კომპონენტების ან ბიბლიოთეკების დაყენებაც (მაგალითად, Windows Installer, .NET Framework, SQL Server Express და სხვ.).

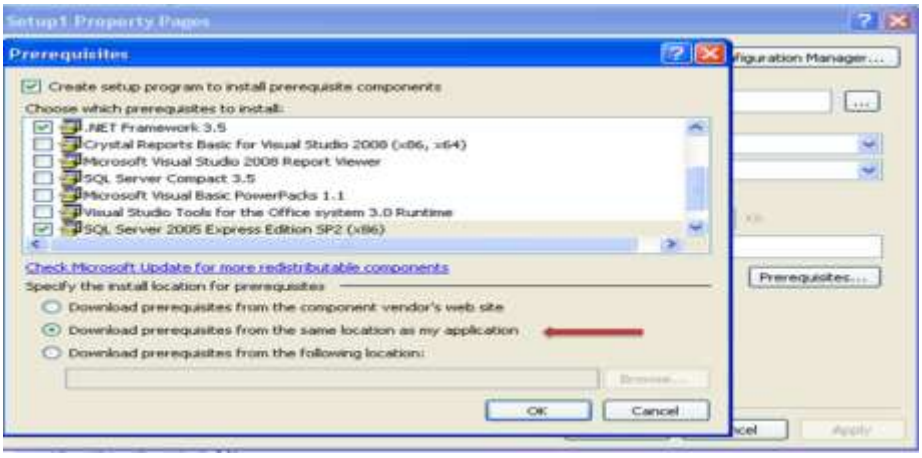
პროექტის Solution Expoler-ის დიალოგურ ფანჯარაში რეალიზებული პროგრამული პაკეტის დამატების შემდეგ ჩნდება ორი პროექტი:

1. მიმდინარე ანუ Setup პროექტი და
2. მიერთებული ანუ რაც უნდა დაინსტალირდეს.

Setup პროექტზე მაუსის მარჯვენა ღილაკის მეშვეობით ფუნქციაზე Property ღილაკით ხდება თვისებების დიალოგური ფორმის გამოძახება, სადაც ღილაკით Prerequisites იხსნება წინაპირობების მისათითებელი დიალოგური ფანჯარა (ნახ.7.13 და 7.14).



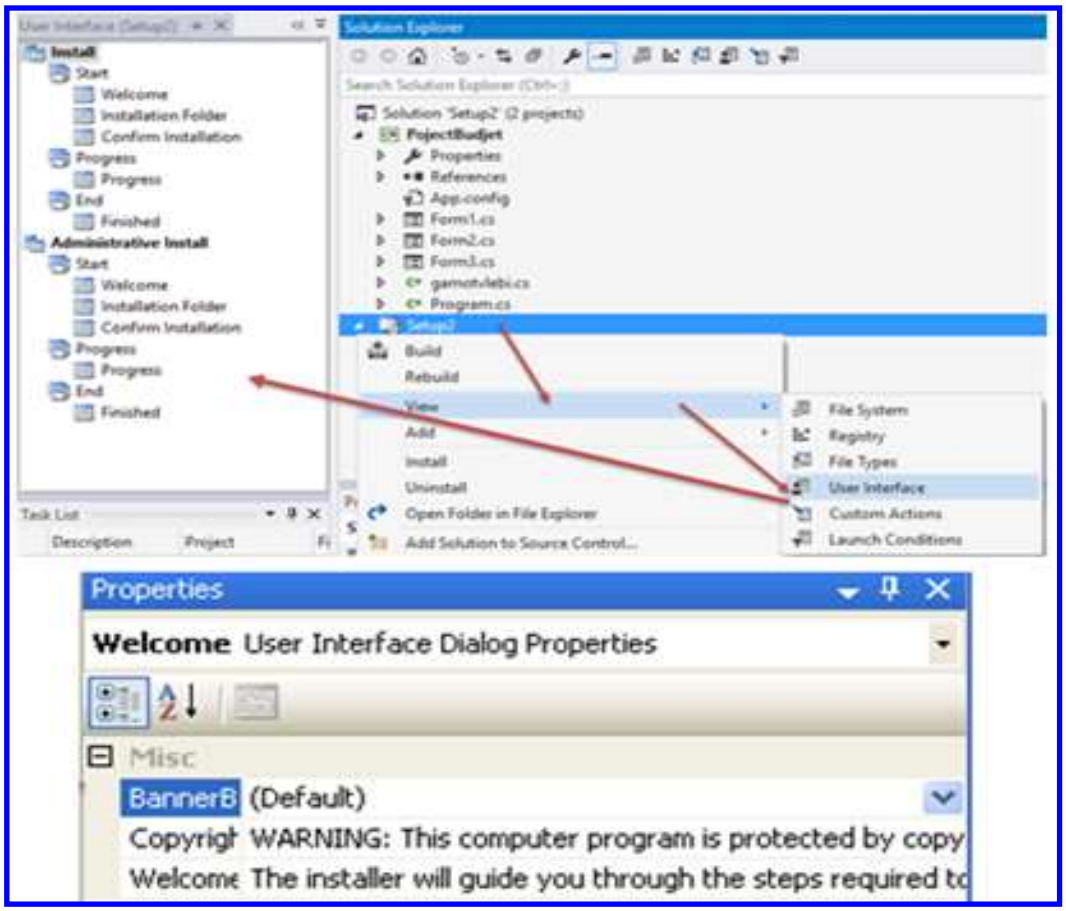
ნახ.7.13



ნახ.7.14

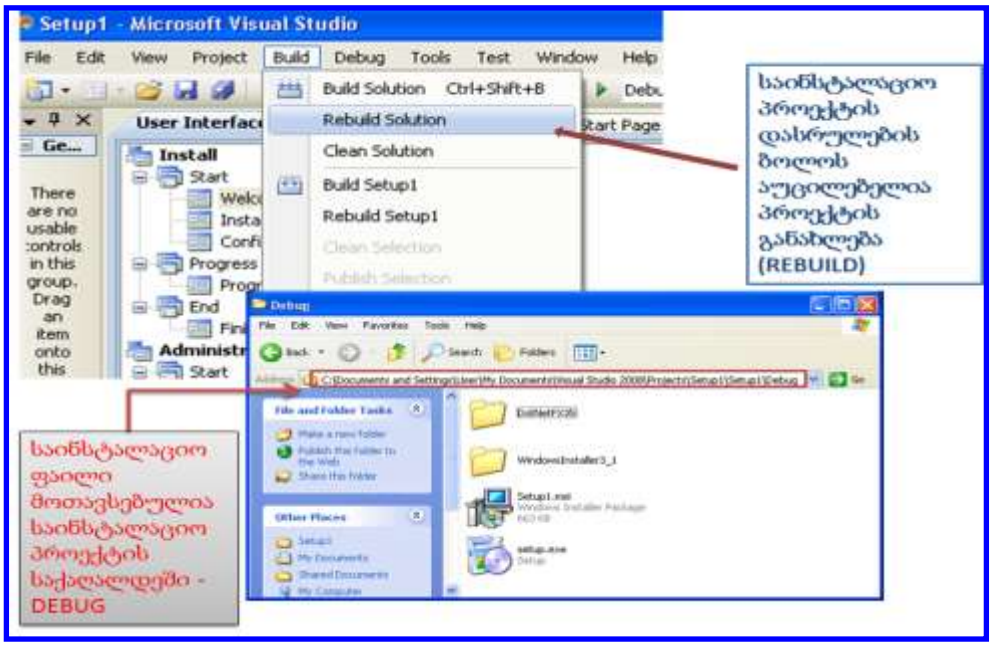
პროექტის ნაწილში File System მოთავსებულია ასევე ორი დამატებითი საქაღალდე User's Desktop (რა განთავსდეს მომხმარებლის სამუშაო მაგიდის საქაღლდეში) და User's Programs Menu (რა განთავსდეს პროგრამული ჩამონათვალის საქაღალდეში).

შესაძლებელია ინსტალაციის პროგრამის დიზაინის ცვლილება (ძირითადად, სურათებით/იკონებით გაფორმება) Setup პროექტზე თაგუს მარჯვენა ღილაკის მეშვეობით View-User Interface ფუნქციის გამოძახება (ნახ.7.15).



ნახ.7.15

საბოლოოდ, აუცილებელია Setup პროექტის განახლება ფუნქციით Rebuild (ნახ.7.16). Setup.exe გამშვები ფაილი მოთავსებულია Setup პროექტის საქაღალდის ქვესაქაღალდეში - Debug.


Setup1 - Microsoft Visual Studio
File Edit View Project Build Debug Tools Test Window Help
Build Solution Ctrl+Shift+B
Rebuild Solution
Clean Solution
Build Setup1
Rebuild Setup1
Clean Selection
Publish Selection
Start Page
User Interface
Install
Start
Progress
End
Administrative Install
There are no usable controls in this group. Drag an item onto this
Debug
File Edit View Favorites Tools Help
File and Folder Tasks
Other Places
საინსტალაციო პროექტის დასრულების ბოლოს აუცილებელია პროექტის განახლება (REBUILD)
საინსტალაციო ფაილი მოთავსებულია საინსტალაციო პროექტის საქაღალდეში - DEBUG

ნახ.7.16

VIII თავი
საკურსო პროექტის გაფორმების ინსტრუქციები

➢ აპლიკაციის საპილოტო ვერსიის ტესტირება და სადემონსტრაციოდ მომზადება

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

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

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

➢ 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 გვერდი (გუნდური პროექტი 25+n*10 გვ., სტუდენების რაოდენობა გუნდში n=2, 3 ან 4).

პროექტის დაცვა ხდება გამიცდების წინა კვირაში.

გამოყენებული ლიტერატურა

1. ჩოგოვაძე გ., ფრანგიშვილი ა., სურგულაძე გ. მართვის საინფორმაციო სისტემების დაპროგრამების ჰიბრიდული ტექნოლოგიები და მონაცემთა მენეჯმენტი. სტუ, „ტექნიკური უნივერსიტეტი“, თბილისი, 2017, 1001 გვ. https://gtu.ge/book/monacemta _menejmenti.pdf

2. გოგიჩაიშვილი გ., ბოლხი გ., სურგულაძე გ., პეტრიაშვილი ლ. მართვის ავტომატიზებული სისტემების ობიექტ-ორიენტირებული დაპროექტების და მოდელირების ინსტრუმენტები (MsVisio, WinPepsy, PetNet, CPN). სტუ. „ტექნიკური უნივერსიტეტი”. თბ., 2013. -232 გვ. http://gtu.ge/book/ims/Gogichai-Surgul.pdf

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

4. სურგულაძე გ. კომპიუტერული პროგრამირების მეთოდები და მეთოდოლოგიები (SP, OOP, VP, Agile, UML). ISBN 978-9941-1900-1. სტუ. „IT-კონსალტინგ ცენტრი“. თბ., 2019. -200 გვ. https://gtu.ge/book/Surg_ProgMethod _2019.pdf

5. Make Your Vision a Reality. Enterprise Architect. Fast Intuitive Modeling & Design. Internet resource: https://sparxsystems.com/ (20.12.21)

6. სურგულაძე გ., ბულია ი. კორპორაციულ Web–აპლიკაციათა ინტეგრაცია და დაპროექტება. მონოგრ., სტუ. „ტექნიკური უნივერსიტეტი“. თბ., 2012. -200 გვ. ელ–ვერსია: http://www.gtu.ge/books/ GiaSurg_Book_2012.pdf

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

8. სურგულაძე გ. ვიზუალური დაპროგრამება C#_2010 ენის ბაზაზე. სტუ. „ტექნიკური უნივერსიტეტი“. თბ., 2011. -445 გვ.

9. სურგულაძე გ., კივილაძე გ. შესავალი NoSQL მონაცემთა ბაზებში. ISBN978-9941-0-9642-6. სტუ. „IT-კონსალტინგ ცენტრი“. თბ., 2017. -152 გვ. https://gtu.ge/book/ NoSQL_Surgul.pdf

10. ჩოგოვაძე გ., , გოგიჩაიშვილი გ., სურგულაძე გ., შეროზია თ., შონია ო. მართვის ავტომატიზებული სისტემების დაპროექტება და აგება (თეორიული და პრაქტიკული ინფორმატიკა). ISBN 99928-882-7-X. სტუ, თბ., 2001, 740 გვ.

11. Сургуладзе Г., Качибая В., Кортуа Т. О выборе приемлемых нормальных форм логической схемы реляционных БД. Сб. трудов ГПИ, „Техн.Кибернетика", №10 (267), 1983. стр. 47-51

12. Surguladze G., Topuria N., Chikovani D. Construction of an Optimal Conceptual Schema RDB using Object-Role Modeling Notation. ISBN 2298-0032,, Journal of Business, IBSU, vol.1, Black Sea Univ., Tbilisi, 2014, pp.15-18

13. Чоговадзе Г., Качибая В., Сургуладзе Г. Теория реляционных зависимостей и проектирование логической схемы баз данных. ISBN 5-511-00072-8. ТГУ, „Тбилисский Гос. Университет", 1988, - 230 ст.

14. სურგულაძე გ., დოლიძე ს. მომხმარებლის ინტერფეისის დაპროგრამება (AngularJS, ReactJS). ISBN 978-9941-8-0625-4. სტუ. „IT-კონსალტინგ ცენტრი“. თბ., 2019. -106 გვ. https://gtu.ge/book/SurguDoliReact.pdf

15. ჩოგოვაძე გ., სურგულაძე გ., გულიტაშვილი მ., დოლიძე ს. პროგრამული აპლიკაციების ხარისხის მართვა: ტესტირება და ოპტიმიზაცია. ISBN 978-9941-20-629-2. სტუ. „IT-კონსალტინგ ცენტრი“. თბილისი, 2020. -363 გვ. https://gtu.ge/book/Surgu_SoftwareQuality.pdf

16. სურგულაძე გ., გულიტაშვილი მ., კივილაძე ნ. Web-სისტემების ტესტირება, ვალიდაცია და ვერიფიკაცია. მონოგრ. ISBN 9789941-0-7682-4. სტუ. „IT-კონსალტინგის ცენტრი“. თბილისი. 2015. -205 გვ.

17. WPF this.NavigationService.Navigate. Internrt resource: https://stackoverflow.com/questions/54750564/wpf-this-navigationservice-navigate (20.04.22)

18. NavigationService.Navigate Method. Internrt resource: https://learn.microsoft.com/en-us/dotnet/api/system.windows.navigation.navigationservice.navigate?view=netframework-4.8 (20.04.22)

19. სურგულაძე გ., თურქია ე.პროგრამული სისტემების მენეჯმენტის საფუძვლები. ISBN 978-9941-20-651-1. სტუ. „ტექნიკ.უნივ.“, თბ., 2016. -350 გვ.

20. ASP and ASP.NET Tutorials. Internet resource: https://www.w3schools.com /asp/ default.asp (18.01.22)

21. Hardin J. How do I get Visual Studio 2019 to connect to IIS Express? 2021. Internet resource: https://learn.microsoft.com/en-us/answers/questions/648737/how-do-i-get-visual-studio-2019-to-connect-to-iis.html (20.01.22)

22. ADO.NET Overview. Internet resource: https://learn.microsoft.com/en-us/dotnet/framework/data/adonet/ado-net-overview (22.02.22)

დანართები

დანართი N1

/საკურსო პროექტის სატიტულო გვერდი/

ინფორმატიკისა და მართვის სისტემების ფაკულტეტი

პროგრამული ინჟინერიის დეპარტამენტი (№805)

საკურსო პროექტი

დისციპლინაში: „პროგრამული პროდუქტების დეველოპმენტი“

ჯგ.N: 108 . . .
სტუდენტ(ებ)ი :

თემა: “ საკურსო პროექტის სათაური “

ხელმძღვანელი: პროფ. გ სურგულაძე
ასოც.პროფ. ნ. თოფურია

თბილისი 2022/23

დანართი N2

საკურსო პროექტის ინდივიდუალური და ჯგუფური სავარაუდო თემები

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

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

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

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

5. საპრობლემო სფერო: **საწარმოო ფირმა**
    - 8.1. ობიექტი: პროდუქტი, ფასი, კატეგორია
    - 8.2. ობიექტი: პროდუქტი, თვითღირებულება, ფასი, მოგება, რენტაბელობა
    - 8.3. ობიექტი: პროდუქტი, ნედლეული, მიმწოდებელი, ნედლეულის_ფასი
    - 8.4. ობიექტი: თვიური/წლიური შემოსავალი, ხარჯი, მოგება
    - 8.5. ობიექტი: ნედლეულის და მზა პროდუქციის საწყობები

6. საპრობლემო სფერო: **კლინიკა**
    - 6.1. ობიექტი: პაციენტი, დაავადება, მკურნალი_ექიმი
    - 6.2. ობიექტი: ექიმი, ნოზოლოგიური_განყოფილება, ოთახი, ტელეფონი
    - 6.3. ობიექტი: პაციენტი, დაავადება, მკურნალობის_ფასი

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

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

7. საპრობლემო სფერო: **მარკეტინგი**

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

3.2. ობიექტი: ავტოპროფილაქტიკა, მომსახურების აღრიცხვა

3.3. ობიექტი: კომპიუტერების მაღაზია

3.4. ობიექტი: წიგნების მაღაზია სექციების მიხედვით

3.5. ობიექტი: პარფიუმერიის მაღაზია

8. საპრობლემო სფერო: **რესტორანი**

8.1. ობიექტი: მენიუ, კერძები, ფასები

8.2. ობიექტი: წინასწარი დაჯავშნის ამოცანა

8.3. ობიექტი: კლიენტთა მაგიდის მომსახურება

8.4. ობიექტი: შეკვეთების მიღება კერძების ბინაზე მიტანით

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

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

დანართი N3

საკურსო პროექტის რეპორტის სარჩევი

1. საპრობლემო სფეროს ობიექტისთვის ამოცანის დასმა;

2. გადასაწყვეტი ამოცანის „როლების და ქმედებების“ UML დიაგრამები (UseCase, Activity) და სცენარი (Sequence D);

3. პროექტისთვის მონაცემთა ბაზის სტრუქტურის შემუშავება (ER მოდელი) და SQL Server ბაზის შექმნა 3-6 ცხრილით (დასაშვებია SQL Server გამოყენება);

4. მომხმარებელთა ინტერფეისები (სამუშაო ფორმები), MsVisual Studio.NET Entity Framework ინტეგრირებულ გარემოში (WPF Application),

5. აპლიკაციის აგება NuGet-პაკეტის, C# და XAML ენების საფუძველზე;

6. მომხმარებელთა ინტერფეისების დაკავშირება მონაცემთა ბაზასთან და შედეგების ასახვა Windows-ან Web- ფორმებზე;

7. დამუშავებული პროგრამის გამართვა და ტესტირება

8. საბოლოო პროდუქტის მომზადება;

9. სისტემის დემოვერსია და საპრეზენტაციო სლაიდები.

საკურსო პროექტის ნიმუშები
(შესრულებული სტუდენტების მიერ 2021-2022 წწ.)

დანართი N4

ინფორმატიკისა და მართვის სისტემების ფაკულტეტი

პროგრამული ინჟინერიის დეპარტამენტი

საკურსო პროეატი საგანში
„პროგრამული პროდუქტების დეველოპმენტი“

თემა:

„მართვის საინფორმაციო სისტემის აგება სააფთიაქო პროცესების ავტომატიზაციისათვის“

ჯგუფი: **108753**
სტუდენტი: **გიორგი მაცაბერიძე**
ხელმძღანელი **:**
პროფ. გ. სურგულაძე

თბილისი - 2022

პროექტის აღწერა

საკურსოს თემა ეხება აფთიაქის პროგრამულ უზრუნველყოფასა და ამ პროგრამულ უზრუნველყოფაში გათვალისწინებულ უსაფრთხოების საკითხებს. პროგრამა საშუალებას იძლევა აფთიაქის თანამშრომელთა მიერ პროგრამის დახმარებით მონაცემთა ბაზაში დამატებული, წაშლილი ან რედაქტურებული იქნას მედიკამენტთა თუ თანამშრომელთა მონაცემები.

პროგრამის რეალიზება ხდება Microsoft Visual Studio 2019, Microsoft SQL Server 2019 და Microsoft Visio 2013 დახმარებით. პროგრამა აგებულია WPF ტექნოლოგიის დახმარებით, რომელიც წარმოდგენილია სამომხმარებლო ინტერფეისი კონკრეტული მართვის ელემენტებით. მათთვის შემუშავებულია ფუნქციური ალგორითმები და პროგრამები ობიექტ-ორიენტირებული დაპროგრამებისა და სერვის-ორიენტირებული არქიტექტურით.

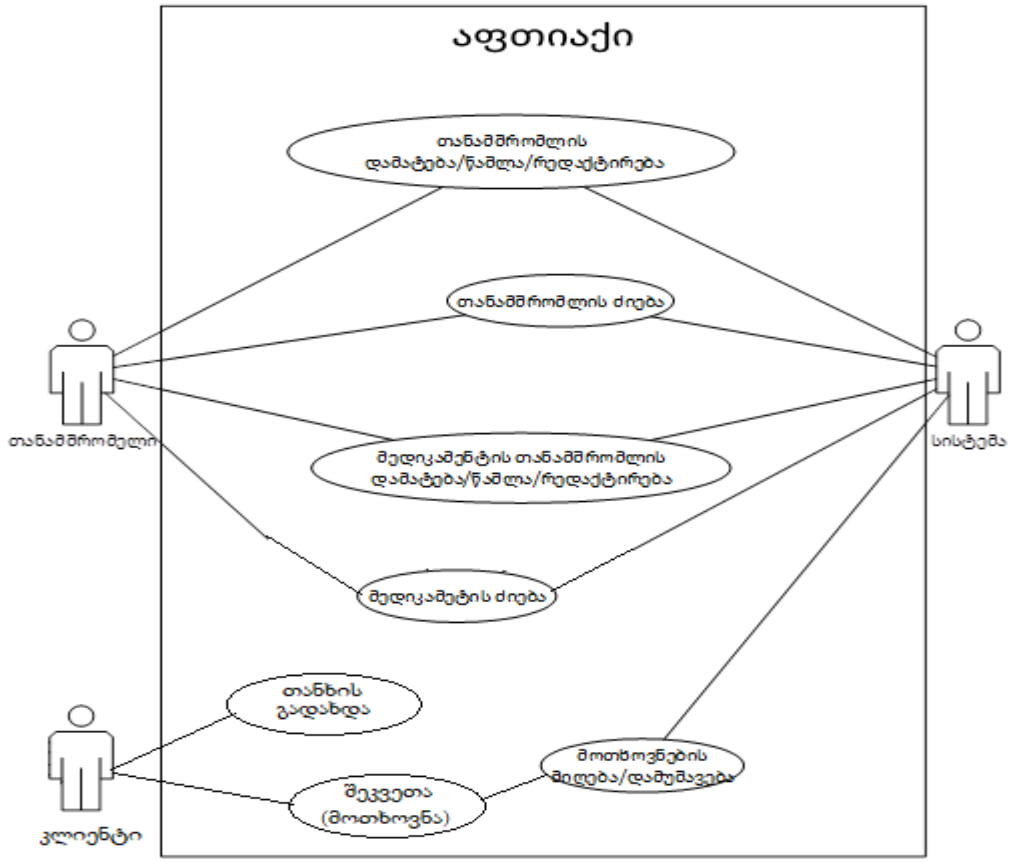
პროექტში წარმოდგენილია პროგრამული უზრუნველყოფის სასიცოცხლო ციკლის ეტაპების ამოცანების გადაწყვეტა. უნიფიცირებული მოდელირების ენის (UML ტექნოლოგია) საფუძველზე აგებული UseCase, Activity დიაგრამები და Sequence ინტერაქტიური სცენარები. მომხმარებლის ინტერფეისები დაკავშირებულია მონაცემთა ბაზის ცხრილებთან და ხორციელდება მონაცემთა ავტომატიზებული დამუშავების პროცედურები. პროგრამული უზრუნველყოფა რეალიზებულია WPF (C# და XAML) ტექნოლოგიით და Ms SQL Server ბაზით.

პროექტში შემუშავებულია ინფორმაციის დაცვის (აუთენტიფიკაციის თვალსაზრისით) პროგრამა მომხმარებელთა არასანქცირებული წვდომის აღმოსაფრხვრელად.

1. დასაპროექტებელი სისტემის ბიზნეს-მოთხოვნილებების განსაზღვრა

1.1. UseCase დიაგრამა

საპრობლემო სფეროს (აფთიაქი) უნიფიცირებული მოდელირების ენის მეთოდოლოგიის საფუძველზე ავაგეთ როლებისა და ფუნქციების UseCase დიაგრამა (ნახ.1). ძირითადი როლებია (კაცუნები) თანამშრომელი, კლიენტი და სისტემა. თანამშრომელი არის: დირექტორი, ბუღალტერი, მენეჯერი, მოლარე-ოპერატორი და სხვ. (მემკვიდრეობითი კავშირები). სისტემა არის კომპიუტერზე გადატანილი ფუნქციების ნაკრები, რომელსაც იყენებენ თანამშრომლები. კლიენტი არის მედიკამენტების ან სხვ. პროდუქტის პოტენციური შემძენი ფიზიკური პირი.

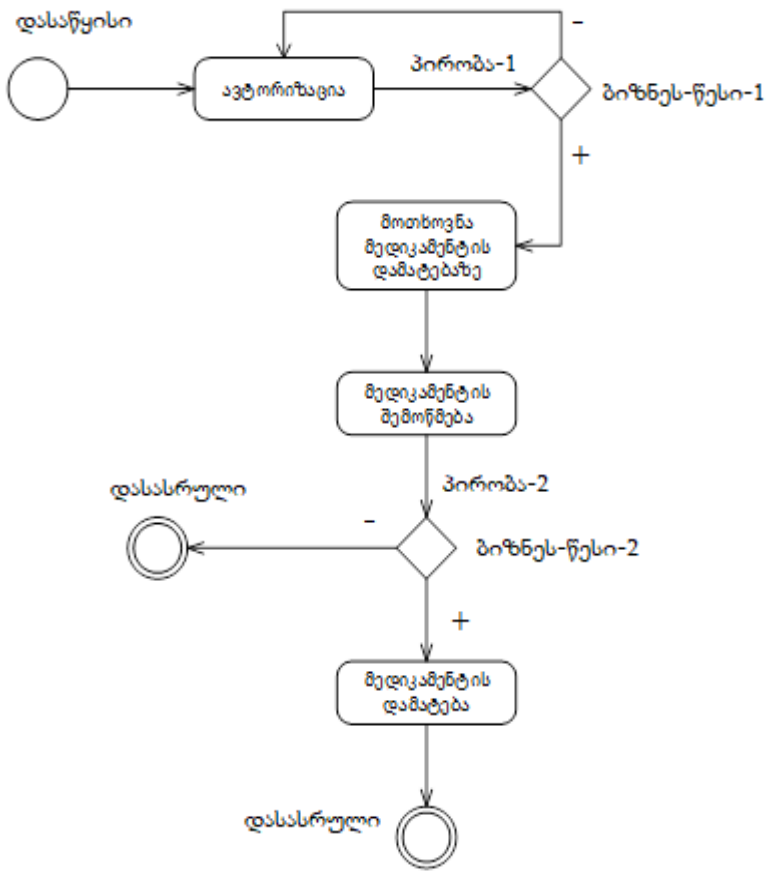


ნახ.1. როლები და ფუნქციები (მოდელი)

1.2. Activity დიაგრამა

აქტიურობის ანუ ქმედებათა დიაგრამა აღწერს UseCase დიაგრამის (ნახ.1) ოვალებში მოთავსებულ ფუნქციებს - აქტივობებს. აქტიურობის დიაგრამა მოიცავს კონკრეტული ბიზნეს-ამოცანის ბიზნეს-პროცესებს და ბიზნეს-წესებს.

მე-2 ნახაზზე განხილული გვაქვს „მედიკამენტების დამატების და გამოკლების“ პროცესების სცენარი აფთიაქში.

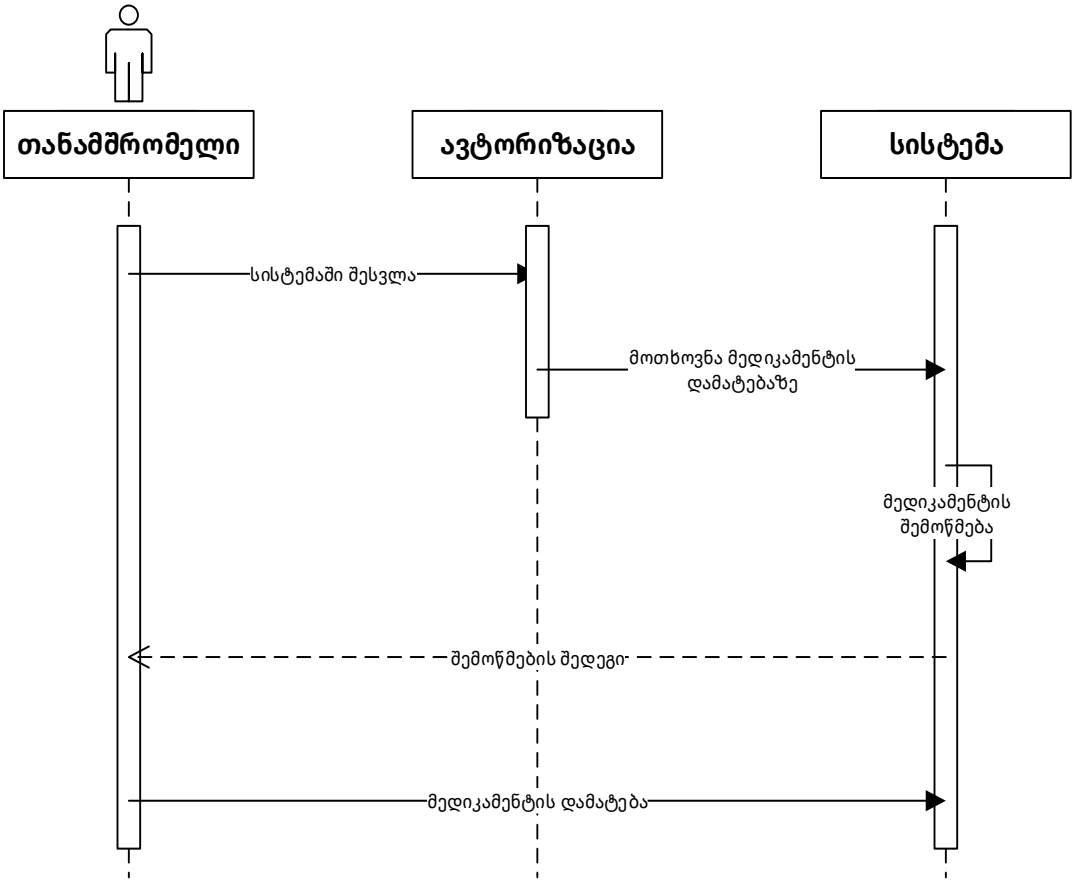


ნახ.2. ბიზნეს-პროცესის მაგალითი

1.3. Sequence დიაგრამა

მიმდევრობითობის დიაგრამა აღწერს სცენარს, თუ როგორ უნდა იმუშაოს აფთიაქის კონკრეტულმა თანამშრომელმა (სისტემის ფუნქციურმა მომხმარებელმა) მის შესაბამის პროგრამულ ინტერფეისთან.

მე-3 ნახაზზე ნაჩვენებია მაგალითი თანამშრომელი (ამ შემთხვევაში მენეჯერი, ან მედიკამენტების საწყობის გამგე) გადის ავტორიზაციას სისტემაში. წარმატებული შესვლის შემთხვევაში იგი სისტემაში იღებს ინფორმაციას მედიკამენტების რაოდენობრივ მდგომარეობაზე. გამოაქვს გადაწყვეტილება მათი დამატების შესახებ შეკვეთის მომზადებაზე და გადაცემაზე. მიმწოდებლიდან შემოსული მედიკამენტების შემთხვევაში, ახალი მედიკამენტები უნდა დაემატოს საწყობს შესაბამისად.



ნახ.3.

2. მონაცემთა ბაზის და ცხრილების აგება SQL Server -ში

2.1. მონაცემთა ბაზის შექმნა და ცხრილების მომზადება

ჩვენი საპრობლემო სფერო – აფთიაქი, ბიზნეს-მიზნებიდან გამომდინარე, მოითხოვს მონაცემთა ბაზის აგებას შემდეგი ცხრილებით: მომხმარებლები (users), თანამშრომლები (employees), პრივილეგიები (privilege), მედიკამენტები (medication) და სხვ. შესაძლებელი უნდა იყოს ახალი როლის დამატება.

users

Column Name	Data Type	Allow Nulls
usr_ID	int	☐
usrname	nvarchar(15)	☐
passwd	nvarchar(33)	☐
privilege_ID	int	☐
emp_ID	int	☐
		☐

employees

Column Name	Data Type	Allow Nulls
emp_ID	int	☐
emp_name	nvarchar(15)	☐
emp_surname	nvarchar(25)	☐
emp_birthday	date	☐
emp_MF	nvarchar(5)	☐
emp_pn	nvarchar(11)	☐
emp_phone	nvarchar(9)	☐
emp_email	nvarchar(255)	☑
		☐

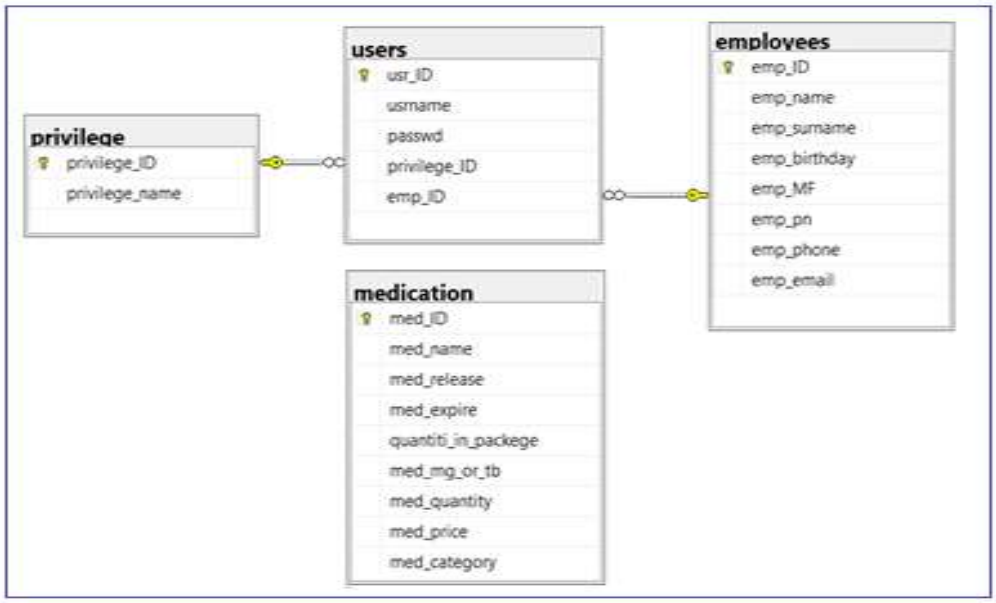
privilege

Column Name	Data Type	Allow Nulls
privilege_ID	int	☐
privilege_name	nvarchar(15)	☐
		☐

medication

Column Name	Data Type	Allow Nulls
med_ID	int	☐
med_name	nvarchar(50)	☐
med_release	date	☐
med_expire	date	☐
quantiti_in_packege	int	☐
med_mg_or_tb	nvarchar(8)	☐
med_quantity	int	☐
med_price	float	☐
med_category	nvarchar(20)	☐
		☐

2.2. ცხრილთა კავშირები

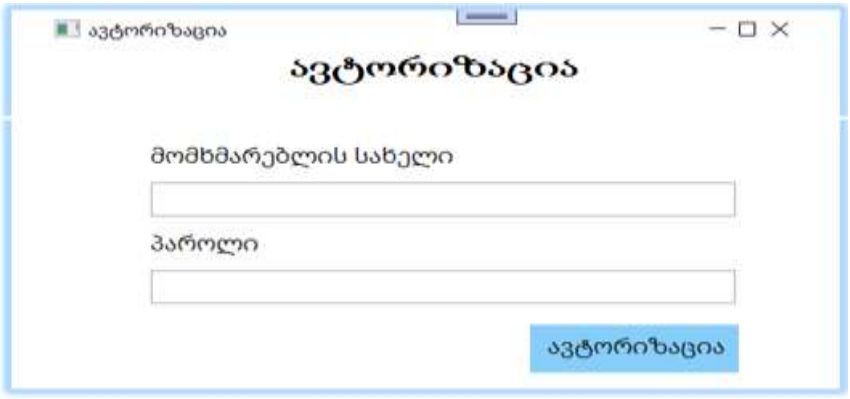


ნახ.4. აფთიაქის ცხრილთაშორისი კავშირების ფრაგმენტი

3. პროგრამული უზრუნველყოფა

3.1. ავტორიზაცია

პროგრამულ უზრუნველყოფაზე წვდომისთვის სავალდებულოა ავტორიზაციის გავლა. წინასწარ განაწილებული მომხმარებელთა როლები საშუალებას იძლევა სხვადასხვა მომხმარებელს სხვადასხვა წვდომის უფლებები მივანიჭოთ. ამ შემთხვევაში პროგრამაზე წვდომისთვის განკუთვნილია 3 როლი: დირექტორი, მენეჯერი და ფარმაცევტი (თითოეულ მათგანს გააჩნია თავიანთი უფლებები).



ნახ.5.

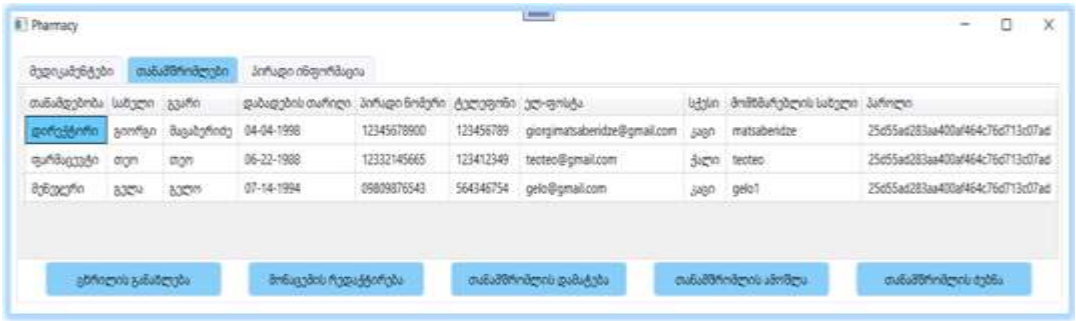
➢ დირექტორი

მომხმარებლის ამ როლს პროგრამაში ყველაფრის უფლება აქვს, თანამშრომლის და მედიკამენტების დამატება, წაშლა, რედაქტირება.

მთავარი გვერდი დირექტორის

მედიკამენტებისთვის

თანამშრომლებისთვის

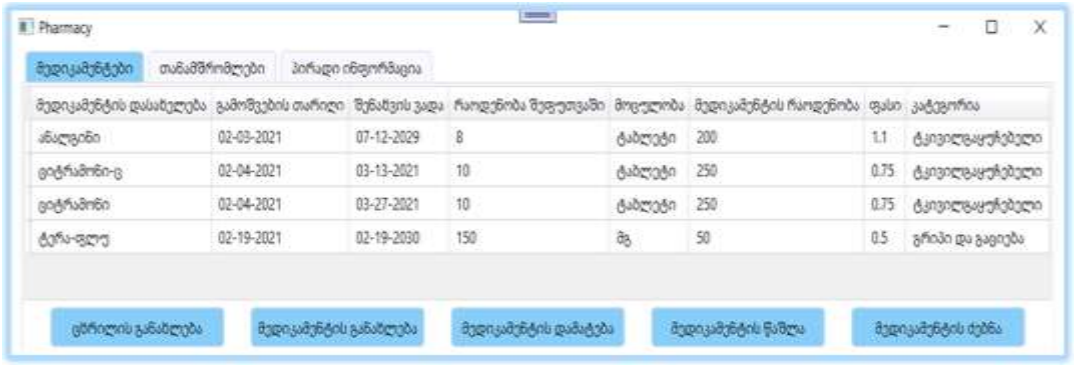


➢ მენეჯერი

მომხმარებელთა ამ როლს მხოლოდ მედიკამენტების დამატება, წაშლა, რედაქტირება და თანამშრომელთა სიის ნახვა შეუძლია

მთავარი გვერდი მენეჯერისთვის

მედიკამენტებისთვის

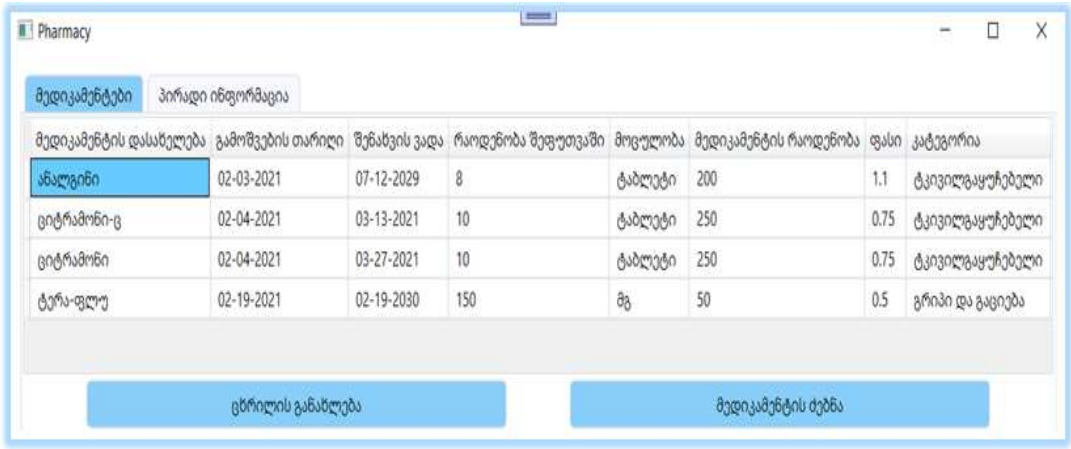


თანამშრომლებისთვის

- ფარმაცევტი

ამ როლს მხოლოდ მედიკამენტების სიის ნახვის უფლება აქვს.

მთავარი გვერდი ფარმაცევტისთვის

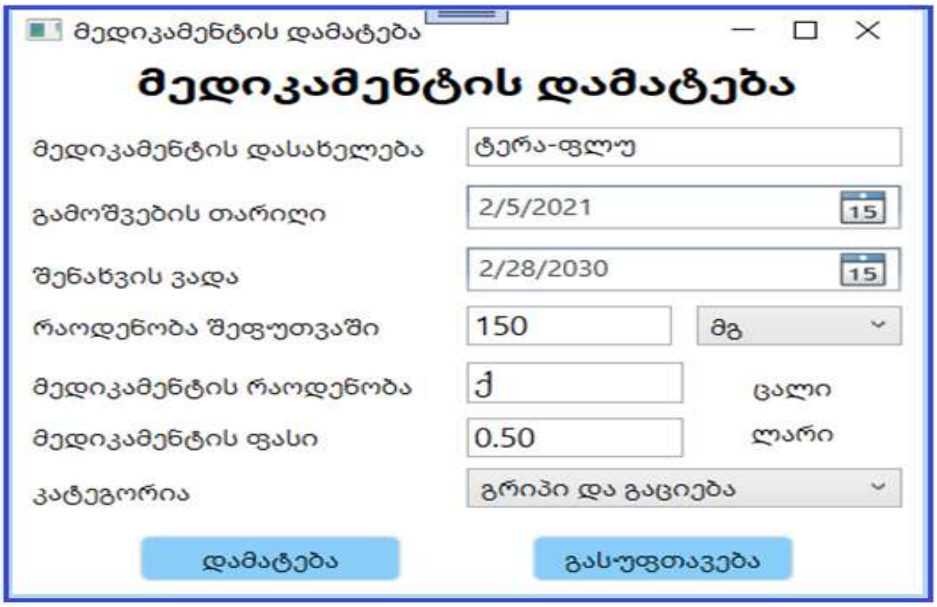


3.2. შესავსები ველების კონტროლი

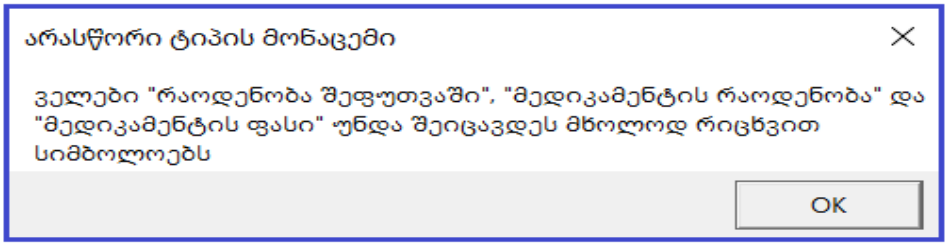
პოგრამაში არასწორი ტიპის ინფორმაციის შემოტანის, ცარიელი ველის დატოვებისა და სიმბოლოების არასაკმარისი რაოდენობის შემთხვევაში პროგრამა მომხმარებელს ატყობინებს ამის თაობაზე.

- არასწორი ტიპის ინფორმაცია

არასწორი ტიპის ინფორმაციის შემთხვევაში პროგრამას შემდეგი სახის შეტყობინება გამოაქვს

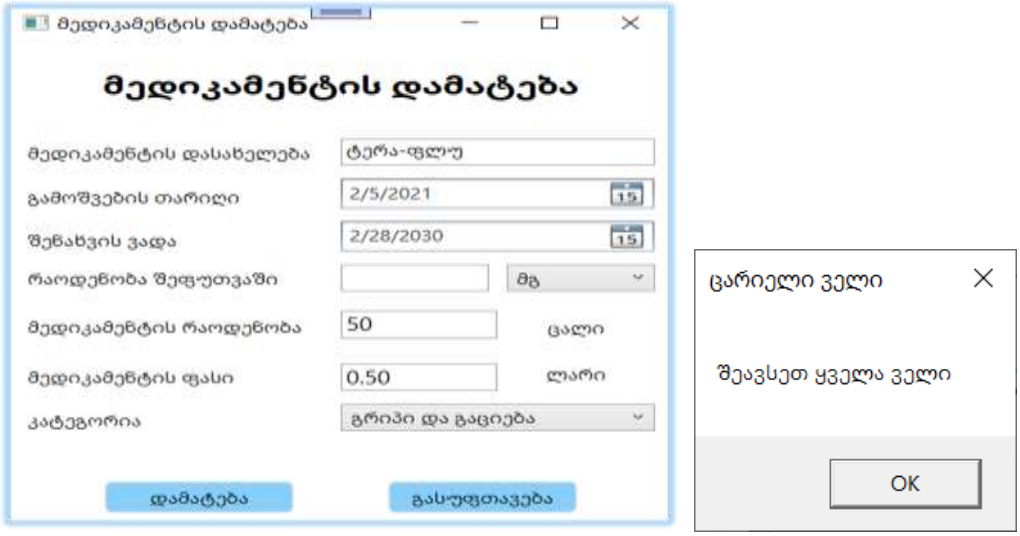


ნახ.6



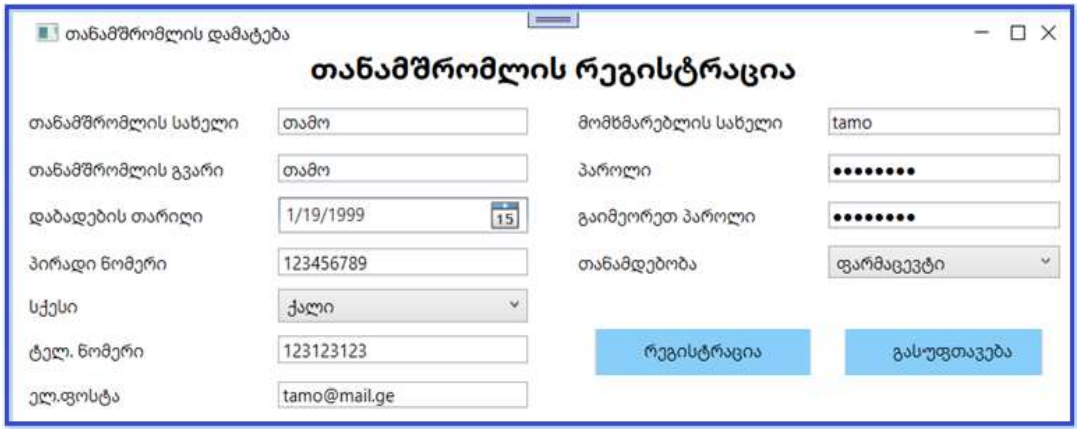
ნახ.7

- ცარიელი ველი

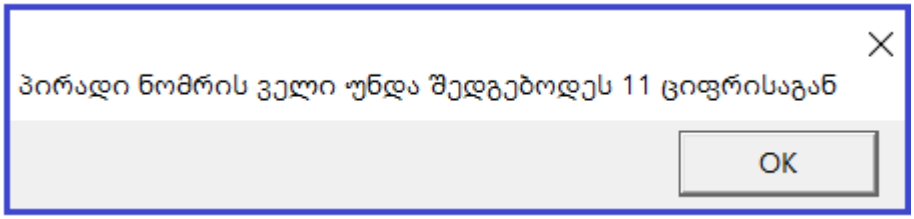


ნახ.8

- არასაკმარისი სიმბოლოები



ნახ.9

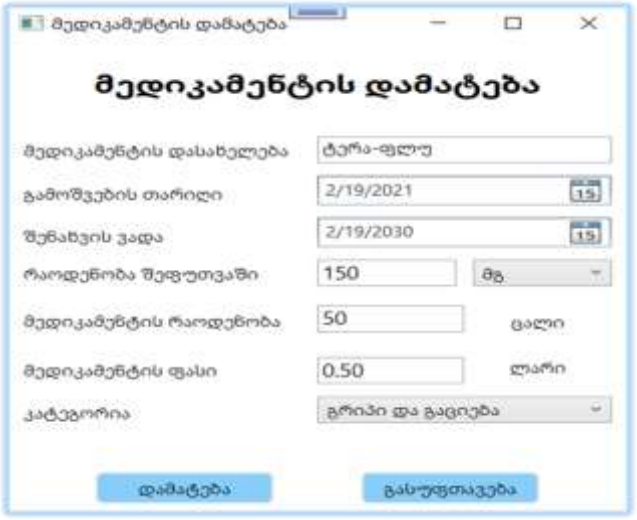


ნახ.10

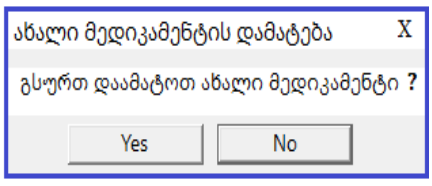
3.3. მონაცემთა დამატება/წაშლა/რედაქტირება

პროგრამაში შესაძლებელია მონაცემთა დამატება/წაშლა/რედაქტირება. არასასურველი მონაცემის წაშლის, დამატების ან რედაქტირების თავიდან აცილების მიზნით. მონაცემთა ბაზაში ნებისმიერი ცვლილების შემთხვევაში პროგრამას ამის თაობაზე გამოაქვს შესაბამისი გამაფრთხილებელი შეტყობინება.

- მონაცემთა დამატება

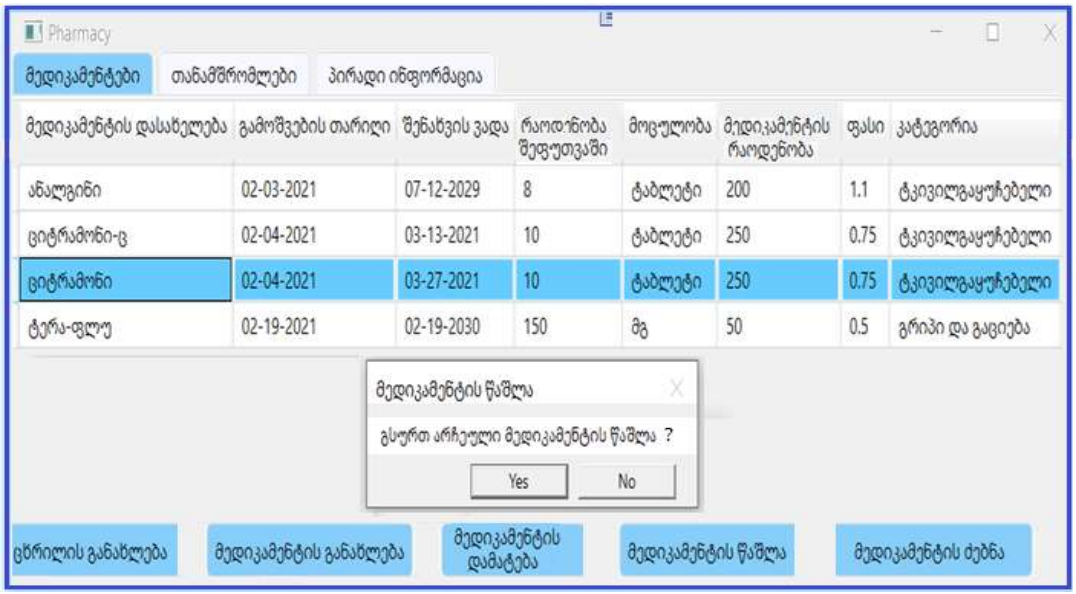


ნახ.11



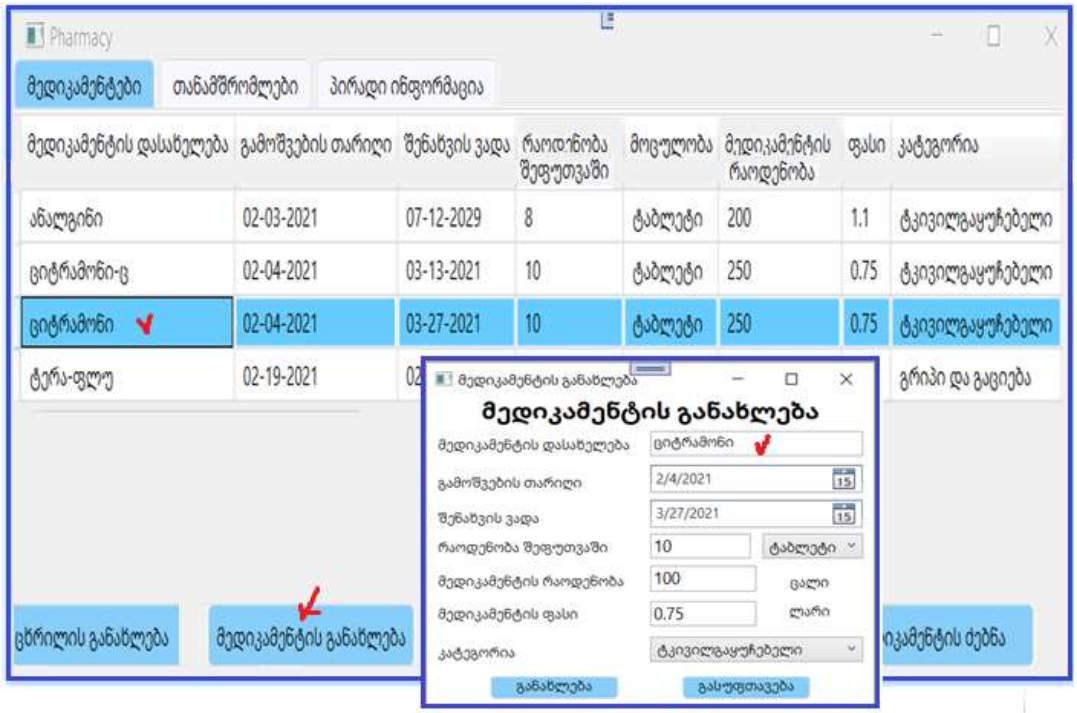
ნახ.12

- მონაცემების წაშლა

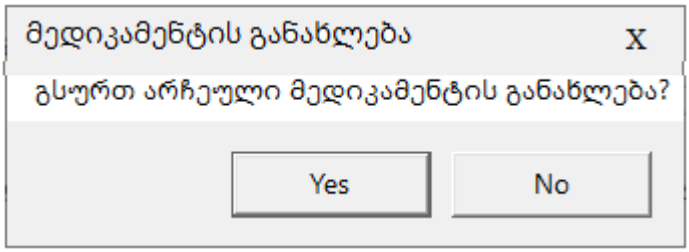


ნახ.13

- მონაცემთა რედაქტირება



ნახ.14



ნახ.15

4. უსაფრთხოება

პროგრამაში უსაფრთხოებისთვის გათვალისწინებულია რამდენიმე უსაფრთხოების ზომა:

- პაროლების დაშიფვრა (კრიპტოგრაფიული ალგორითმი)

მეტი უსაფრთხოებისთვის ახალი მომხმარებლის დამატების შემთხვევაში მომხმარებლის პაროლი იშიფრება MD5 შიფრაციის ალგორითმით და ისე ეხდება მონაცემთა ბაზაში შენახვა

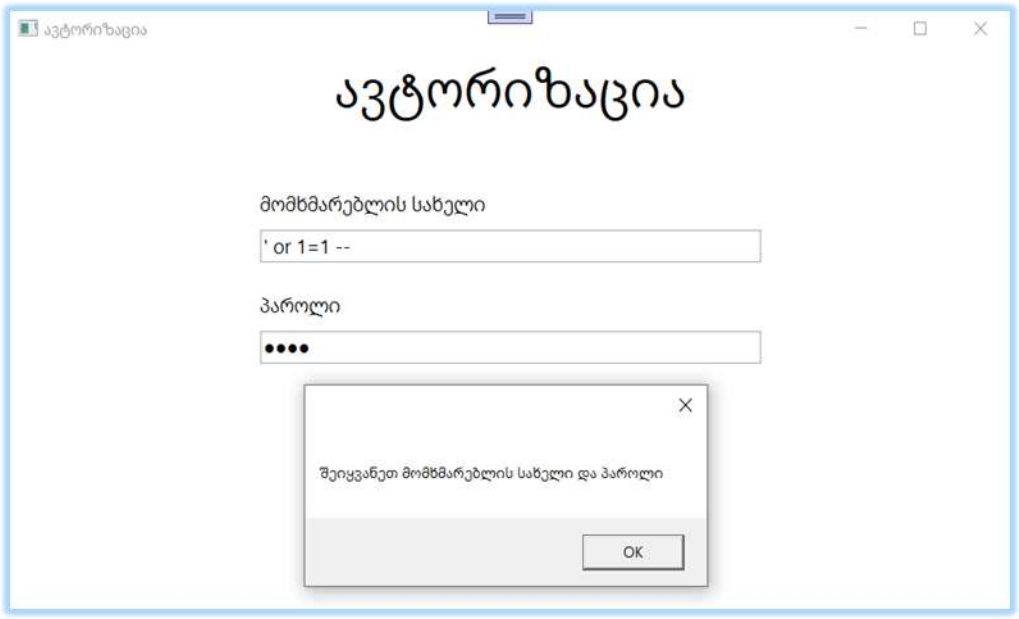
```
private string MD5Hash(string psswd)
{
    MD5 md5 = new MD5CryptoServiceProvider();
    md5.ComputeHash(ASCIIEncoding.ASCII.GetBytes(psswd));
    byte[] result = md5.Hash;
    StringBuilder stringBuilder = new StringBuilder();
    for (int i = 0; i < result.Length; i++)
    {
        stringBuilder.Append(result[i].ToString("x2"));
    }
    return stringBuilder.ToString();
}
```

პაროლი
25d55ad283aa400af464c76d713c07ad
25d55ad283aa400af464c76d713c07ad
25d55ad283aa400af464c76d713c07ad

➢ SQL Injection-ის თავიდან აცილება

SQL Injection-ის თავიდან აცილებისთვის პროგრამა ავტორიზაციის დროს ამოწმებს ხომ არ აქვს ადგილი მსგავს ქმედებას

```
private bool IsUsrnameContain()
{
    if (usrname.Text != "" && passwd.Password != "")
    {
        foreach (char symbol in usrname.Text)
        {
            if (symbol == '-' || symbol == '=' || symbol == ' ')
                return true;
        }
        return false;
    }
    else
        return true;
}
```



ნახ.16.

შენიშვნა: პროგრამა - „ფარმაცევტი“

პრგრამაზე წვდომისთვის შექმნილია 3 მომხმარებელი:
დირექტორი - მომხმარებლის სახელი: matsaberidze პაროლი: 12345678
მენეჯერი - მომხმარებლის სახელი: gelo1 პაროლი: 12345678
ფარმაცევტი - მომხმარებლის სახელი: teoteo პაროლი: 12345678

გამოყენებული ლიტერატურა:

1. ჩოგოვაძე გ., ფრანგიშვილი გ., სურგულაძე გ. მართვის საინფორმაციო სისტემების დაპროგრამების ჰიბრიდული ტექნოლოგიები და მონაცემთა მენეჯმენტი. თბ., „ტექნიკური უნივერსიტეტი“. 2017. -1001 გვ., http://gtu.ge/book/monacemta_menejmenti.pdf

2. 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

3. სურგულაძე გ., თურქია ე. პროგრამული სისტემების მენეჯმენტის საფუძვლები. სახელმძღვ., სტუ, თბილისი, 2016. 350 გვ. სახელმძღვ., სტუ, თბილისი, 2016. 350გვ. http://gtu.ge/book/gia_sueguladze/GiaSurg1_%20Prog-SysManag.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_sisteme-bi_BSI_ITIL_COBIT.pdf

5. სურგულაძე გ., პეტრიაშვილი ლ., ბიტარაშვილი მ. კორპორაციული მენეჯმენტის სისტემების დაპროგრამე¬ბის ტექნოლოგია (საკურსო პროექტე-ბისთვის). სტუ. თბ., დამხმ.სახელმძღ., 2017, -232 გვ. http://gtu.ge/book/Sakurso-Project_WPF.pdf

დანართი N5

საგანმანათლებლო პროგრამა: ინფორმატიკა
(პროგრამული ინჟინერია)

საკურსო პროექტი

თემა:

საბანკო მომსახურების პროგრამული აპლიკაცია ანდროიდის ბაზაზე

ჯგ.: 108851
სტუდენტი: **ანა ბერულავა**

ხელმძღვანელი:
პროფ. გ. სურგულაძე,
ასოც.პროფ. ნ. თოფურია

თბილისი - 2022

სარჩევი

შესავალი

ადამიანებს, უხსოვარი დროიდან აქვთ ფულთან კავშირი - ყიდულობენ, ყიდიან, ინახავენ, თუმცა ფულთან კომუნიკაციის გზები და მიდგომები დროთა განმავლობაში შეიცვალა, დაიხვეწა და თანამედროვე საზოგადოების მოთხოვნებზე მორგებული გახდა.

გასული საუკუნის პირველ ნახევარსა და მეორე ნახევრის დასაწყისში, ადამიანებს, საბანკო მომსახურების მისაღებად, აუცილებლად უწევდათ ბანკში მისვლა და ზოგჯერ, საათობით, დიდ რიგებში დგომა; თუმცა გასული საუკუნის 90-იანი წლებიდან დაიწყო ცვლილებების ხანა ინტერნეტისა და კომპიუტერული ტექნოლოგიების განვითარებასთან ერთად. 1994 წელს, „Stanford Credit Union" იყო პირველი საფინანსო დაწესებულება ამერიკის შეერთებულ შტატებში, რომელმაც თავის მომხმარებლებს შესთავაზა ონლაინ საბანკო მომსახურება. ამის შემდეგ, სხვა საბანკო დაწესებულებებმაც დაიწყეს ონალინ მომსახურების გამოყენება, რადგან ინტერნეტი სულ უფრო ხელმისაწვდომი ხდებოდა სახლებში.

კიდევ ერთი გარდამტეხი მომენტი თანამედროვე საზოგადოებისთვის იყო ე.წ. „სმართფონების" განვითარების ახალი ეტაპი (პირველი სმართფონი ისეთი სახით, როგორაც ჩვენ ვიცნობთ, ჯერ კიდევ 1994 წელს გამოუშვა IBM-მა). „ჭკვიანი ტელეფონები" სულ უფრო და უფრო ხელმისაწვდომი ხდებოდა ადამიანებისთვის. ისეთმა გიგანტმა კომპანიებმა, როგორიც არის Google და Apple, შექმნეს ოპერაციული სისტემები მობილურებისთვის: Android და iOS. შეიქმნა ე.წ. „აპლიკაციის მაღაზიები", საიდანაც მომხმარებლებს შეეძლოთ უამრავი აპლიკაციის გადმოწერა და მისი ნებისმიერ ადგილას გამოყენება, თავიანთ კომპიუტერთან კომუნიკაციის გარეშე. სმარტფონების სწრაფად გავრცელებამ (2,5 მილიარდი აქტიური Android-ის მომხმარებელი და 1 მლიარდი iOS მომხმარებელი) საბანკო დაწესებულებები დააყენა იმ ამოცანის წინაშე, რომ შეეთავაზებინათ თანამედროვე საბანკო მომსახურებები, როგორც ვებ სივრცეში, ასევე მობილურის მომხმარებლებისთვისაც.

ჩვენ მიერ შექმნილი მობილური აპლიკაცია, მომხმარებლებს უადვილებს სხვადასხვა, ბანკთან დაკავშირებული პროცდურების შესრულებას. აპლიკაციას აქვს მომხმარებელზე მორგებული, მეგობრული ინტერფეისი და მარტივია მასთან კომუნიკაცია. აპლიკაცია აგებულია ანდროიდის ბაზაზე - მომხმარებლის ინტერფეისის გასამართად ვიყენებთ XML-ს, ხოლო გამოყენებული დაპროგრამების ენა არის Kotlin-ი; Back end სერვისებისთვის - Google Firebase (მომხმარებლის ავტორიზაცია, რეგისტრაცია), ხოლო, მომხმარებლის ინფორმაციის შესანახად და მასზე შემდგომი წვდომისთვის - FireBase Real-time database. აპლიკაცია ეყრდნობა OOP (ობიექტ-ორიენტირებულ) და SOLID პრინციპებს და იყენებს MVVM არქიტექტურას.

1. საპრობლემო სფერო: პროექტირების ობიექტი და პროგრამირების ინსტრუმენტი

ბანკი არის ფინანსური დაწესებულება, რომლის მთავარი ფუნქციაა დეპოზიტების აღება და სესხების გაცემა, თუმცა ბანკმა მომხმარებლებს შეიძლება შესთავაზოს სხვადასხვა სერვისები, როგორიცაა: სადეპოზიტო ანგარიშების შექმნა, სესხების გაცემა, მათ შორის იპოთეკური სესხების, საკრედიტო ბარათები, ჩეკის განაღდების მომსახურენა, ქონების მართვა და დაზღვევა.

საბანკო დაწესებულების მთავარი მიზანი არის ფულის ნაკადის მართვა ადამიანებსა და ბიზნესს შორის. უფრო კონკრეტულად, ბანკები, ჩვენ მომხმარებლებს, გვთავაზობენ სადეპოზიტო ანგარიშებს, რომლებიც უსაფრთხო ადგილია ფულის შესანახად. შემდგომში, ბანკები, სადეპოზიტო ანგარიშებზე არსებულ ფულს აძლევენ სხვა ადამიანებს ან ბიზნესს სესხის სახით.

დღესდღეობით, ბანკები, თანამედროვე ეკონომიკის მნიშვნელოვან ნაწილს წარმოადგენს. ბანკების არსებობის აუცილებლობისა და მათ მომსახურებაზე მოთხოვნის გაზრდის გამო, ისინი, არ შემოიფარგლებიან მხოლოდ ერთი დაწესებულებით და მათი ფილიალები მრავლადაა ქვეყნის მასშტაბით. სწორედ ამიტომ, ბანკში დასაქმებულ ადამიანთა რაოდენობა, დასაქმებულთა საერთო რაოდენობის დიდ ნაწილს შეადგენს.

ბანკები უზრუნველყოფს მომსახურებას, როგორც ფიზიკურად (ოფისებში), ასევე, ინტერნეტის საშუალებით. იმისთვის, რომ მომხმარებელთა მომსახურება მოხდეს შეუფერხებლად, თანამედროვე საბანკო დაწესებულებებს ესაჭიროებათ თანამედროვე ტექნოლოგიები; მნიშვნელოვანია, ასევე, ფილიალებისა და თანამშრომლების მართვა და მათი ერთმანეთთან შეუფერხებელი კომუნიკაცია.

ბანკი, საკმაოდ რთული ფინანსური დაწესებულებაა, რომელიც სწორ მენეჯმენტსა და შეუფერხებელ მუშაობას მოითხოვს. დღესდღეობით, მრავლადაა საბანკო სისტემები, რომლებიც ბანკებს თავიანთი ფუნქციის სწორად შესრულებაში ეხმარება.

ჩვენი პროექტის ***მიზანია*** ის, თუ როგორ შეიძლება გამარტივდეს საბანკო მომსახურების მიწოდება მომხმარებლებისთვის მობილური აპლიკაციის საშუალებით და რა მნიშვნელოვანი სარგებელი შეიძლება ჰქონდეს საბანკო მომსახურების ასეთი ფორმით მიღებას, როგორც კლიენტებისთვის, ასევე, თავად ბანკისთვის.

2. თანამედროვე მობილური ბანკი (mobile banking) და მისი ფუნქციები

როგორც ზემოთ აღვნიშნეთ, ტექნოლოგიების განვითარებამ, ინტერნეტისა და სმარტფონების პოპულარულობის ზრდამ, თანამედროვე საბანკო დაწესებულებები დააყენა გამოწვევის წინაშე, შეექმნათ ისეთი პროდუქტი, რომელიც მომხმარებელს შესთავაზებდა მობილურ, მოქნილ და კომფორტულ გადაწყვეტას საბანკო მომსახურების მისაღებად და სწორედ ასეთი გადაწყვეტაა, სპეციალური აპლიკაცია, შემუშავებული ბანკის მიერ, რომლიც გადმოწერაც და რომლით სარგებლობაც მარტივად შეუძლიათ ბანკის მომხმარებლებს.

მობილური ბანკით სარგებლობა გულისხმობს, მობილურ მოწყობილობაზე (მობილური ტელეფონი, ტაბლეტი) ტრანზაქციების განხორციელების აქტს. ეს აქტივობა შეიძლება იყოს: თანხის გადარიცხვა როგორც ქვეყნის შიგნით, ასევე, მის ფარგლებს გარეთ, გადასახადების გადახდა და სხვ. ტანზაქციების განხორციელების დროს, მომხმარებელს არ უწევს იმაზე ფიქრი, თუ რამდენად დაცულია მისი ფული. MFA აუტენთიფიკაციის საშუალებით, შესაძლებელია თქვენი ანგარიშის დაკავშირება თქვენს მობილურ მოწყობილობასთან, რაც ავტომატურად მოახდენს თქვენს გაფრთხილებას, თუ ვინმე არასანქცირებულად შეეცდება, თვენს ანგარიშზე შეღწევას. ზემოთ ჩამოთვლილი უპირატესობებიდან, აშკარაა, რომ მობილური ბანკი მოსახერხებელი და უსაფრთხოა გამოსაყენებლად. მომხმარებელს შეუძლია შევიდეს თავის ანგარიშზე, მობილური აპლიკაციის საშუალებით ნებისმიერ ადგილას და ნებისმიერ დროს.

3. მობილური ბანკის (mobile banking) აპლიკაციები თანამედროვე ბაზარზე

ტექნოლოგიების განვითარებასთან ერთად, მრავალმა ბანკმა მსოფლიოს მასშტაბით შეიუმშავა მობილური აპლიკაცია, რომელიც მათ მომხმარებლებს უფრო გაუადვილებდა ბანკთან ურთიერთობას და საბანკო სერვისებით სარგებლობას, მათ შორის არის რამდენიმე ქართული ბანკი: საქართველოს ბანკი, თიბისი ბანკი და ლიბერთი ბანკი.

ერთერთი გამორჩეული მობილური აპლიკაცია, საქართველოს ბაზარზე არის თი-ბი-სი ბანკის მობილური აპლიკაცია, რომელიც ხელმისაწვდომია Android და iOS პლათფორმებზე და მომხმარებელს აძლევს საშუალებას ისარგებლოს შემდეგი საბანკო მომსახურებით:

- ერთი აპლიკაციის გამოყენებით შეხვიდეთ როგორც იურიდიული პირის ასევე ფიზიკური პირის მობაილ ბანკში;
- ხელმისაწვდომი და ბლოკირებული თანხების ნახვა და ამონაწერის გაკეთება;
- თანხის გადარიცხვა თქვენს ანგარიშებზე და სხვასთან;
- გადარიცხვა კომპანიის ანგარიშებზე, სხვა პირის ან კომპანიის ანგარიშსა და ბიუჯეტში;
- მობილურის ბალანსის შევსება, კომუნალური, საბიუჯეტო გადასახადებისა და ჯარიმების გადახდა;
- გადარიცხვის შაბლონების შექმნა;
- დეპოზიტებისა და სესხების მართვა;
- დახარისხებული შემოსავლების და ხარჯების ნახვა;
- ვალუტის კურსების შემოწმება და კონვერტაცია;
- ბარათის დაბლოკვა და განბლოკვა, განახლება, ახალი ბარათის შეკვეთა;
- თიბისი ბანკის უახლოესი ფილიალის ან/და ბანკომატის პოვნა.

4. პროგრამული სისტემის დამუშავება

ჩვენი მიერ შექმნილი მობილური აპლიკაცია მომხმარებლებს სთავაზობს შემდეგ შესაძლებლობებს :

- რეგისტრაცია (ანგარიშის არარსებობის შემთხვევაში);
- სისტემაში შესვლა;
- ბალანსის შემოწმება (თანხის შეტანა/გამოტანა);
- ვალუტის კურსის შესახებ ინფორმაციის მიღება;
- ვალუტის კონვერტაცია;
- ტოპ-50 კრიპტოვალუტის შესახებ ძირითადი ინფორმაციის მიღება.

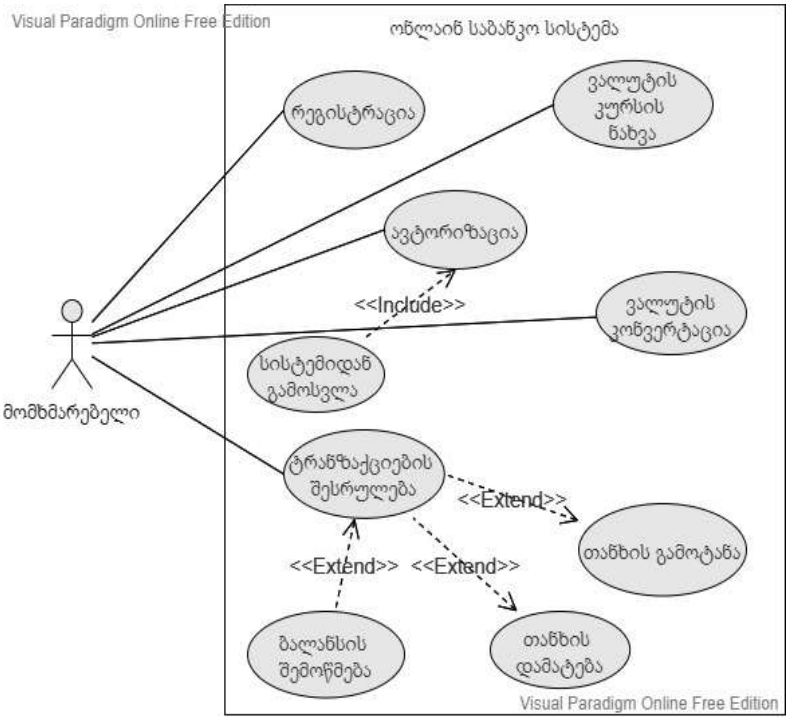
აპლიკაცია „Baad Bank”-ი, შექმნილია Android ოპერაციული სისტემის პლათფორმაზე. თავდაპირველად, კვლევის ობიექტის ანალიზის საფუძველზე დავადგინეთ სისტემის ბიზნეს-მოთხოვნები, ავაგეთ როლებისა და ფუნქციების მოდელი (UseCase Diagram)[6]. შემდეგ ეტაპზე, შევიმუშავეთ აპლიკაციის დიზაინი, რომელიც აგებულია Adobe XD-ში და სწორედ ამის შემდეგ გადავედით, უშუალოდ აპლიკაციის დაპროექტებაზე. მომხმარებლის ინტერფეისის ასაგებად გამოვიყენეთ XML-ი (Extensible Markup Language), ხოლო ბიზნეს ლოგიკა მთლიანად შემუშავებულია Kotlin დაპროგრამების ენაზე.

Back end სერვისებისთვის (რეგისტრაცია, ავტორიზაცია, პაროლის აღდგენა, პაროლის შეცვლა) გამოვიყენეთ Google Firebase, ხოლო მომხმარებლის მიმდინარე ანგარიშისა და სხვა პირადი ინფორმაციის შესანახად, ასევე,

ანგარიშზე განხორციელებული ცვლილებების რეალურ დროში მონიტორინგი-სათვის გამოვიყენეთ Firebase Real-time database მონაცემთა ბაზა. ვალუტის კურსისა და ვალუტის კონვერტაციისთვის გამოვიყენეთ developer.tbcbank.ge-ზე შემოთავაზებული Api-ები (Get commercial rates და Convert with commercial rates) ,ხოლო, კრიპტოვალუტაზე ინფორმაციის მისაღებად გამოვიყენეთ Coinpaprika Api.

4.1. მოთხოვნილების განსაზღვრა

უნიფიცირებული მოდელირების ენის (Unified Modeling Language) მეთოდოლოგიის საფუძველზე ავაგეთ სისტემის მომხმარებლებისა და მათი ფუნქციების UseCase-diagrama (ნახ.1) [5]

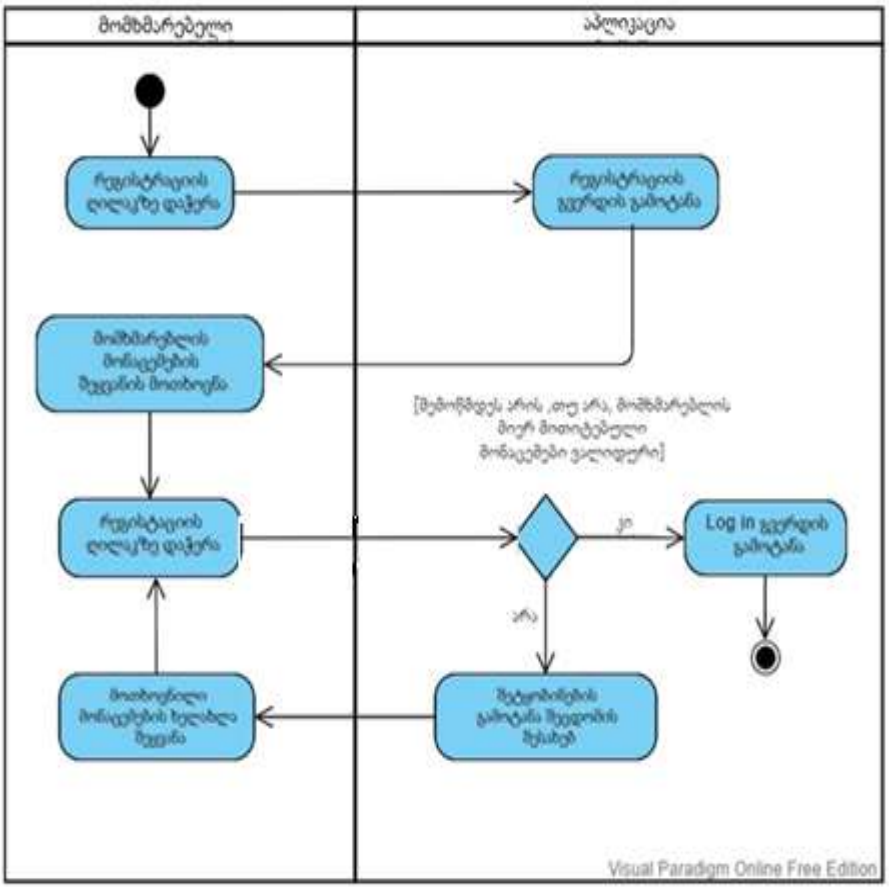


ნახ.1. UseCase - დიაგრამა (როლები და ფუნქციები)

ნახაზზე გამოსახულია მომხმარებელი და მისი ფუნქციები. ჩვენ ამ მომხმარებლისთვის უნდა ავაგოთ ინტერფეისი, რომლებზეც მოთავსდება ოვალებში ნაჩვენები ფუნქციები.

მაგალითად, მე-2 ნახაზზე ნაჩვენებია მომხმარებლის აპლიკაციაში რეგისტრაციის აქტიუროზათა დიაგრამა. მომხმარებელი, რომელიც არ არის რეგისტრირებული, ავტორიზაციის გვერდიდან გადადის რეგისტრაციის გვერდზე და შეჰყავს მოთხოვნილი მონაცემები; თუ მის მისერ შეყვანილი

მონაცემები ვალიდურია, ის გადადის ავტორიზაციის გვერდზე, წინააღმდეგ შემთხვევში , სისტემა გასცემს შეტყობინებას შეცდომაზე და მომხმარებელს თავიდან მოუწევს მონაცემების შეყვანა [5].



ნახ.2. Activity-დიაგრამა (მომხმარებლის აპლიკაციაში რეგისტრაციის აღწერა)

4.2. მობილური ბანკის მონაცემთა ბაზა

აპლიკაციის მონაცემთა ბაზა აგებულია Firebase Real time dabatse-ის საშუალებით.

Firebase Realtime database არის NoSQL ღრუბლოვანი მონაცემთა ბაზა, რომელიც ინახავს მონაცემებს JSON ფორმატის საშუალებით და ის ახდენს მონაცემების სინქრონიზაციას რეალურ დროში ყველა დაკავშირებულ კლიენტთან.

როგორც მე-3 ნახაზზე ჩანს, მოცემულ მონაცემთა ბაზაში არის ერთი Json ობიექტი profile :

ნახ.3

Profile ობიექტი ინახავს ინფორმაციას მომხმარებლის შესახებ (იმეილი, სახელი, ტელეფონის ნომერი და თანხა ანგარიშზე), რომელიც წარმოდგენილია სახელი-მნიშვნელობა (name:value) წყვილის სახით. თითოეულ ჩანაწერს აქვს თავისი უნიკალური იდენტიფიკატორი. სურათზე მოცემულია ერთ-ერთი დარეგისტრირებული მომხმარებლის მონაცემები (საგულიხმოა, რომ ბაზაში მონაცემების ჩაწერა ხდება მას შემდეგ, რაც მომხმარებელი გაივლის რეგისტრაციას) (ნახ.4).

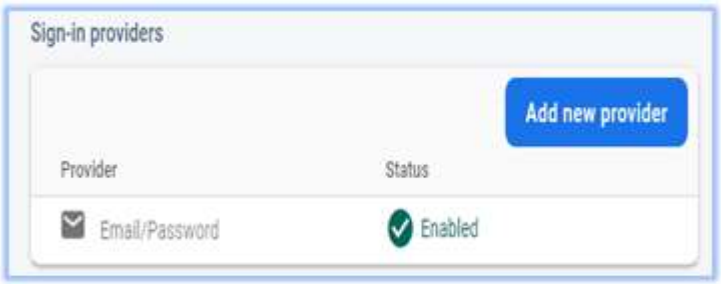
ნახ. 4

4.3. მობილურ ბანკში გამოყენებული Back-end სერვისები

ჩვენი აპლიკაცია მომხმარებელთა რეგისტრაციისა და ავტორიზაციისთვის იყენებს Google Firebase Authentication სერვისს.

Firebase Authentication გვთავაზობს back-end სერვისებს, ადვილად გამოსაყენებელ SDK-ებს და მზა UI ბიბლიოთეკებს, რათა მოხდეს მომხარებლის

რეგისტრაცია და ავტორიზაცია ჩვენს აპლიკაციაში. ის მხარს უჭერს ავტორიზაციას პაროლების, ტელეფონის ნომრებისა და სხვადასხვა პოპულარული პროვაიდერების გამოყენებით, როგორიცაა: Google,Facebook,Twitter და სხვ. ჩვენ აპლიკაციაში, მომხარებელს ავტორიზაცია შეუძლია მეილისა და პაროლის გამოყენებით. როგორც მე-5 ნახაზზე ჩანს, პროვაიდერად სწორედ ეს მეთოდია არჩეული.



ნახ. 5

Firebase პანელი საშუალებას გვაძლევს ვნახოთ ინფორმაცია დარეგისტრირებული მომხმარებლის შესახებ, მაგალითად, მისი იდენტიფიკატორი (ჩვენ შემთხვევაში მეილი), პროვაიდერი, ანგარიშის შექმნის დრო, ბოლო ვიზიტის თარიღი და უნიკალური Id (ნახ.6).

ნახ. 6

ჩვენს მობილურ ბანკში გამოყენებულია რამდენიმე API სერვერთან საკომუნიკაციოდ და საჭირო ინფორმაციის მისაღებად. ერთ-ერთი მათგანი არის api კომერციული ვალუტის კურსის მისაღებად. სერვერთან წარმატებული კომუნიკაციის შემთხვევაში გვიბრუნდება 200 Ok კოდი და შედეგი.

სერვერთან კომუნიკაციის ინიცირება ხდება GET მოთხოვნის საშუალებით, როგორც ეს მე-7 ნახაზზე ჩანს.

```
interface CommercialApi {

    @GET( value: "exchange-rates/commercial")
    suspend fun getCommercialRates():Response<CommercialRates>
}
```

ნახ. 7

შედეგი კი გვიბრუნდება მასივის სახით, რომელიც შეიცავს Json ობიექტებს.

RESPONSE 200 LOG


```
[
  {
    "currency": "EUR",
    "value": 4.0233
  },
  {
    "currency": "USD",
    "value": 3.2766
  }
]
```


ნახ. 8

4.4 მობილური ბანკის ინტერფეისის დაპროგრამება

აპლიკაციის გახსნის შემდეგ, პირველი, რასაც მომხმარებელი ხედავს არის ე.წ. Splash Screen-ი (ნახ.9).

Splash Screen-ზე ჩანს აპლიკაციის ლოგო და ის, აპლიკაციის ჩატვირთვის პროცესს ნაკლებად დამღლელს ხდის მომხმარებლისთვის. Splash Screen-ის შიეძლება შეიცავდეს სხვდასხვა სახის ანიმაციასაც.

Splash Screen-ის ჩატვირთვის შემდეგ მომხმარებლისთვის ხელმისაწვდომი ხდება რეგისტრაციის ფანჯარა (ნახ.10). მომხარებელს შეუძლია გაიაროს ავტორიზაცია, ანგარიშის არქონის შემთხვევაში გადავიდეს რეგისტრაციის გვერდზე, დაიმახსოვროს არსებული ანგარიში, აღადგინოს პაროლი ან ისარგებლოს მხოლოდ სტუმრის პრივილეგიებით.

ნახ. 9

ნახ.10. აპლიკაციის საწყისი გვერდი

თუ მომხმარებელი მიუთითებს არასწორად დაფორმატებულ მეილს (ნახ.11-ა), არასწორ პაროლს (ნახ.11-ბ) ან საერთოდ არ შეავსებს შესაბამის ველებს, გამოჩნდება შეტყობინება შესაბამისი შეცდომის შესახებ ფანჯრის ქვედა ნაწილში Snackbar-ის საშუალებით.

ნახ.11-ა. მომხმარებელმა შეიყვანა არასწორად დაფორმატებული მეილი

ნახ.11-ბ. მომხმარებელმა მიუთითა არასწორი პაროლი

თუ მომხმარებელი პირველად სარგებლობს აპლიკაციით და მხოლოდ სტუმრის პრივილეგიები მისთვის საკმარისი არ არის, შეუძლია გადავიდეს რეგისტრაციის გვერდზე და გაიაროს რეგისტრაცია (ნახ.12).

ახალი მომხმარებლის რეგისტრაციის პროცესი: მომხარებელმა შეავსო ყველა საჭირო ველი (სრული სახელი, ტელეფონის ნომერი, მეილი და პაროლი) (ნახ.13-ა).

ნახ. 12. რეგისტრაციის გვერდი

ნახ. 13-ა

მომხმარებელმა წარმატებით გაიარა რეგისტრაცია, შესაბამისად მისი მონაცემები აისახება, როგორც Firebase-ის ავტორიზაციის გვერდზე (ნახ.13-ბ), ასევე მონაცემთა ბაზაში (ნახ.13-გ).

ნახ.13-ბ

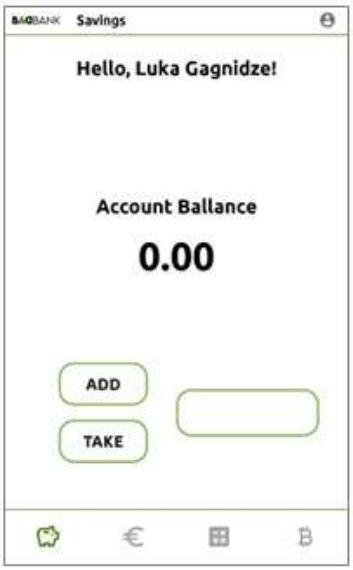
ნახ. 13-გ

წარმატებული რეგისტრაციის შემდეგ, მომხმარებელი გადამისამართდება ავტორიზაციის გვერდზე, სადაც შეიყვანს თავის მონაცემებს და ისარგებლებს თავისი ანგარიშით.

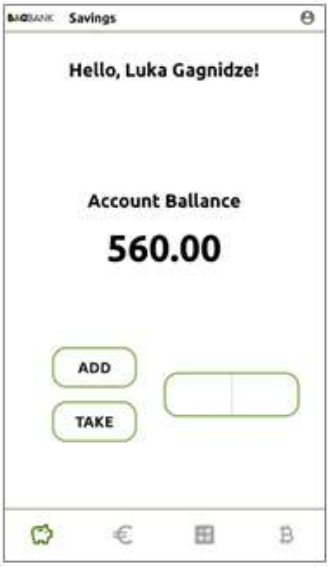
ჩვენ მიერ, ზემოთ დარეგისტრირებული მომხმარებლის ე.წ პირადი კაბინეტი (პირადი გვერდი) შემდეგნაირად გამოიყურება (ნახ.14-ა).

როგორც ვხედავთ, თავდაპირველად მის ანგარიშზე თანხის ოდენობა 0 ლარის ტოლია, თუმცა Add ღილაკის საშუალებით შეგვიძლია თანხის დამატების სიმულაცია (ნახ.14-ბ), ისევე, როგორც Take ღილაკით ანგარიშიდან თანხის გადარიცხვის სიმულაცია (ნახ.14-გ).

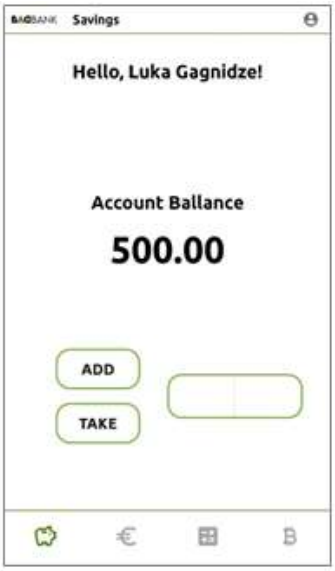
თუ მომხმარებელი დააპირებს იმაზე მეტი თანხის გატანას ვიდრე მის ანგარიშზეა, გამოჩნდება შესაბამისი შეტყობინება შეცდომაზე.



ნახ.14-ა



ნახ.14-ბ. თანხის დამატება ანგარიშზე



ნახ.14-გ. თანხის გატანა ანგარიშიდან.

რაც მთავარია, ყველა განხორციელებული ცვლილება, რეალურ დროში აისახება მონაცემთა ბაზაში. როგორც ინტერფეისზე ჩანს, არსებული მომხარებლის ანგარიშზე 500 ლარის ოდენობის თანხაა, შესაბამისად, იგივე რაოდენობის თანხაა მონაცემთა ბაზაშიც (ნახ.14-დ).

ნახ.14-დ.

ანგარიშზე თანხის შეტანისა და გადარიცხვის გარდა, მომხმარებელს შეუძლია გაეცნოს ვალუტის ოფიციალურ და კომერციულ კურსს (ყიდვა გაყიდვა, ნახ.15), მოახდინოს კონვერტაცია ვალუტებს შორის (ნახ.16-ა,16-ბ) და გაეცნოს ინფორმაციას პირველი ტოპ 100 კრიპტოვალუტის შესახებ (ნახ.17).

ნახ.15. ნახ.16-ა. ნახ.16-ბ. ნახ.17.

ზედა მარჯვენ კუთხეში, პროფილის ფოტოზე დაჭერის შედეგად მომხმარებელი გადავა საინფორმაციო გვერდზე (ნახ.18), სადაც მას შეუძლია ნახოს თავისი პირადი ინფორმაცია (სრული სახელი, ტელეფონის ნომერი და

მეილი), შეცვალოს პაროლი, შეცვალოს პირადი ინფორმაცია (სახელი და ტელფონის ნომერი) (ნახ.19-ა,ნახ.19-ბ) და გამოვიდეს სისტემიდან. სისტემიდან გამოსვლის შემდეგ მომხმარებელი უბრუნდება საწყის, ავტორიზაციის გვერდს.

ნახ.18.

ნახ.19-ა.

ნახ.19-ბ.

განხორციელებული ცვლილება აისახება როგორც ინტერფეისზე, ასევე, მონაცემთა ბაზაში.

დასკვნა

1. საბანკო სფერო ძალზე კონკურენტუნარიანი და სწრაფად განვითარებადია, სწორედ ამიტომ, როგორც ბიზნესს, ასევე მის მომხმარებლებს ესაჭიროებათ სწრაფი და მათ მოთხოვნებზე მორგებული სისტემების შემუშავება;

2. ნებისმიერი პროგრამული უზრუნველყოფა, რომელიც იქმნება ბაზარზე, უნდა ემსახურებოდეს და აკმაყოფილებდეს როგორც ბიზნესის, ასევე, მისი მომხმარებლების ინტერესებს, სწორედ ასეთ პროგრამული პროდუქტია მობილური ბანკი, რომელიც სწორად და მაქსიმალურად ორივე მხარის ინტერესებზეა მორგებული;

3. მობილური ბანკი საშუალებას აძლევს მომხმარებლებს ნებისმიერ დროს, ნებისმიერ ადგილას, ცოცხალ რიგებში დგომისა და დისკომფორტის გარეშე, იტერნეტთან წვოდმის საშუალებით განახორციელოს თითქმის ყველა საბანკო მომსახურება;

4. დღესდღეობით, მობილური ბანკი სულ უფრო და უფრო იხვეწება და მრავალ ახალ ფუნქციას იძენს, რაც კიდევ უფრო მოთხოვნადს და გამოყენებადს ხდის მას.

მომხმარებლის ინტერფეისისა და ბიზნეს ლოგიკის დაპროგრამება :

XML კოდი (მომხმარებლის ინტერფეისი) :

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".ui.login.LoginFragment">


    <ProgressBar
        android:id="@+id/progressbar"
        android:elevation="@dimen/progress_bar_elevation"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:visibility="invisible"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <ImageView
        android:id="@+id/iv_main_background"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:src="@drawable/main_background"
        android:contentDescription="@string/app_s_main_background"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />


    <ImageView
        android:id="@+id/iv_logo"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="16dp"
        android:contentDescription="@string/app_s_secondary_logo"
        android:src="@drawable/ic___logo__a"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        tools:ignore="ImageContrastCheck" />
```

```
<ImageView
    android:id="@+id/iv_name_logo"
    android:layout_width="225dp"
    android:layout_height="37dp"
    android:layout_marginTop="16dp"
    android:contentDescription="@string/app_s_name"
    android:src="@drawable/ic_baadbanklogo"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/iv_logo" />

<com.google.android.material.textfield.TextInputLayout
    android:id="@+id/et_login_email"
    style="@style/TextInputLayoutStyle"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginTop="64dp"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/iv_name_logo"
    app:startIconDrawable="@drawable/ic_login_email">

    <com.google.android.material.textfield.TextInputEditText
        android:id="@+id/etEmail"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:fontFamily="@font/ubuntu_light"
        android:hint="@string/login_emil_hint"
        android:imeOptions="actionDone"
        android:inputType="textEmailAddress"
        android:minHeight="48dp"
        tools:ignore="TextContrastCheck" />

</com.google.android.material.textfield.TextInputLayout>

<com.google.android.material.textfield.TextInputLayout
    android:id="@+id/et_login_password"
    style="@style/TextInputLayoutStyle"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginTop="16dp"
    app:endIconMode="password_toggle"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/et_login_email"
    app:startIconDrawable="@drawable/ic_login_key">


    <com.google.android.material.textfield.TextInputEditText
```

```
            android:id="@+id/etPassword"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:fontFamily="@font/ubuntu_light"
            android:hint="@string/login_password_hint"
            android:minHeight="48dp"
            android:imeOptions="actionDone"
            android:inputType="textPassword" />

    </com.google.android.material.textfield.TextInputLayout>

    <CheckBox
        android:id="@+id/cb_remember_me"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="48dp"
        android:buttonTint="@color/bb_green_01"
        android:fontFamily="@font/ubuntu_light"
        android:minHeight="48dp"
        android:text="@string/remember_me"
        android:textColor="@color/black"
        app:layout_constraintStart_toStartOf="parent"

app:layout_constraintTop_toBottomOf="@+id/et_login_password" />


    <androidx.appcompat.widget.AppCompatTextView
        android:id="@+id/tv_resetPassword"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginEnd="48dp"
        android:fontFamily="@font/ubuntu_light"
        android:text="@string/forgot_password"
        android:textColor="@color/bb_black"

app:layout_constraintBottom_toBottomOf="@+id/cb_remember_me"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintTop_toTopOf="@+id/cb_remember_me" />

    <androidx.appcompat.widget.AppCompatButton
        android:id="@+id/btn_login"
        android:layout_width="0dp"
        android:layout_height="56dp"
        android:layout_marginStart="32dp"
        android:layout_marginTop="80dp"
        android:layout_marginEnd="32dp"

android:background="@drawable/login_register_button_background"
        android:fontFamily="@font/ubuntu_bold"
        android:text="@string/log_in"
        android:textAllCaps="false"
```

```
        android:textColor="@color/bb_white"
        android:textSize="20sp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"

app:layout_constraintTop_toBottomOf="@+id/et_login_password" />


    <androidx.appcompat.widget.AppCompatTextView
        android:id="@+id/tv_no_account"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="20dp"
        android:fontFamily="@font/ubuntu_regular"
        android:text="@string/no_account"
        android:textColor="@color/bb_grey"
        android:textSize="16sp"
        app:layout_constraintStart_toStartOf="@+id/btn_login"
        app:layout_constraintTop_toBottomOf="@+id/btn_login" />

    <androidx.appcompat.widget.AppCompatTextView
        android:id="@+id/tvdontWantToRegister"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginEnd="8dp"
        android:fontFamily="@font/ubuntu_regular"
        android:text="@string/don_t_want_to_register"
        android:textColor="@color/bb_grey"
        android:textSize="16sp"
        app:layout_constraintEnd_toStartOf="@+id/tv_guest"
        app:layout_constraintStart_toStartOf="@+id/btn_login"
        app:layout_constraintTop_toTopOf="@+id/tv_guest" />


    <androidx.appcompat.widget.AppCompatTextView
        android:id="@+id/tv_register"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="20dp"
        android:fontFamily="@font/ubuntu_bold"
        android:text="@string/register"
        android:textColor="@color/bb_black"
        android:textSize="16sp"
        app:layout_constraintEnd_toEndOf="@+id/btn_login"
        app:layout_constraintTop_toBottomOf="@+id/btn_login" />


    <androidx.appcompat.widget.AppCompatTextView
        android:id="@+id/tv_guest"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
```

```
        android:layout_marginTop="16dp"
        android:fontFamily="@font/ubuntu_bold"
        android:text="@string/continue_as_guest"
        android:textColor="@color/bb_black"
        android:textSize="16sp"

        app:layout_constraintEnd_toEndOf="@+id/tv_register"
        app:layout_constraintTop_toBottomOf="@+id/tv_no_account" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

მომხარებლის ავტორიზაცია (ბიზნეს ლოგიკა, Kotlin)

FireBaseRepository class :

```
class FireBaseRepository @Inject constructor() {

 suspend fun loginUser(email: String, password: String):
Flow<Resource<AuthResult>> {
    return flow {
        try {
            emit(Resource.Loading())
            val result =
                auth.signInWithEmailAndPassword(email,
password).await()
            emit(Resource.Success(result))
        } catch (e: Exception) {
            emit(Resource.Error(e.message ?: "unknown error"))

        }
    }.flowOn(IO)

   }

}
```

LoginViewModel class :

```
@HiltViewModel
class LoginViewModel @Inject constructor(
    private val repository: FireBaseRepository, private val
userPreferencesRepository: UserDataStoreRepository
) : ViewModel() {


    private val _userLogInStatus:
MutableSharedFlow<Resource<AuthResult>> = MutableSharedFlow()
    val userLogInStatus: SharedFlow<Resource<AuthResult>> =
_userLogInStatus
```

```
    fun logIn(email: String, password: String) {
        viewModelScope.launch {
            repository.loginUser(email, password).collect {
                _userLogInStatus.emit(it)
            }
        }
    }
```

LoginFragment class :

```
@AndroidEntryPoint
class LoginFragment :
BaseFragment<FragmentLoginBinding>(FragmentLoginBinding::inflate) {

    private var rememberCredential: Boolean = false
    lateinit var password: String
    lateinit var email: String

    private val viewModel: LoginViewModel by activityViewModels()


    override fun start() {

        auth.signOut() //droebit
        val userLogged = auth.currentUser
        userLogged?.let {
            binding.btnLogin.text = userLogged.email
        }


        setListeners()
        observer()
        setLoginCredentials()

    }



    private fun loginUser() {
        val email = binding.etEmail.text.toString()
        val password = binding.etPassword.text.toString()
        viewModel.logIn(email, password)

        viewLifecycleOwner.lifecycleScope.launch {
```

```
            repeatOnLifecycle(Lifecycle.State.STARTED) {
                viewModel.userLogInStatus.collect() {
                    when (it) {
                        is Resource.Loading -> {
                            showLoading()
                        }
                        is Resource.Success -> {
                            hideLoading()
                            saveToUserDatastore()

findNavController().navigate(LoginFragmentDirections.actionLoginFra
gmentToNavHomeFragment())

                        }
                        is Resource.Error -> {
                            hideLoading()
                            view?.makeSnackbar("${it.message}")

                        }
                    }

                }
            }
        }
    }



    private fun setListeners() {


        binding.apply {

            btnLogin.setOnClickListener {
                loginUser()

            }
            tvRegister.setOnClickListener {

findNavController().navigate(LoginFragmentDirections.actionLoginFra
gmentToRegisterFragment())
            }
            tvGuest.setOnClickListener {

findNavController().navigate(LoginFragmentDirections.actionLoginFra
gmentToNavHomeFragmentGuest())
            }

            tvResetPassword.setOnClickListener {
```

```
findNavController().navigate(LoginFragmentDirections.actionLoginFra
gmentToResetPasswordFragment())
            }
        }
    }

    private fun observer() {

        viewModel.userPreferences.observe(viewLifecycleOwner, {
            rememberCredential = it.rememberCredentials
            email = it.password
            password = it.password
        })
    }

    private fun saveToUserDatastore() {

        if (binding.cbRememberMe.isChecked) {
            val email = binding.etEmail.text.toString()
            val password = binding.etPassword.text.toString()
            viewModel.saveUserPreferences(email, password, true)
        }
    }

    private fun setLoginCredentials(){

        viewModel.userPreferences.observe(viewLifecycleOwner, {
userPreferences ->
            val email = userPreferences.email
            val password = userPreferences.password
            if (email.isNotEmpty() && password.isNotEmpty()) {
                binding.etEmail.setText(email)
                binding.etPassword.setText(password)
            }
        }
      )
   }
}
```

გამოყენებული ლიტერატურა:

1. ჩოგოვაძე გ., ფრანგიშვილი ა., სურგულაძე გ., მართვის საინფორმაციო სისტემების დაპროგრამების ჰიბრიდული ტექნოლოგიები და მონაცემთა მენეჯმენტი. სტუ. „ტექნიკ.უნივერსიტეტი“, თბ., 1001 გვ. ინტერნეტ რესურსი: https://gtu.ge/book/monacemta_menejmenti.pdf (5.07.22)

2. Rafols, Ismael, Alan L. Porter, and Loet Leydesdorff. "Science overlay maps: A new tool for research policy and library management." *Journal of the American Society for information Science and Technology* 61.9 (2010): 1871-1887.

3. Li, Shuqing, et al. "Research on the application of information technology of Big Data in Chinese digital library." *Library Management* (2019).

4. სურგულაძე გ., თურქია ე.. ინფორმატიკა - „პროგრამული ინჟინერია" (საბაკალავრო ნაშრომის მეთოდური მითითებანი). ISBN 978-9941-8-2927-7. სტუ. თბ., 42 გვ.

5. სურგულაძე გ. კომპიუტერული პროგრამირების მეთოდები და მეთოდოლოგიები (SP, OOP, VP, Agile, UML). ISBN 978-9941-1900-1. სტუ. „ITკონსალტინგ ცენტრი“. თბ., 2019. -200 გვ.

6. 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 (1.02.22)

იბეჭდება ავტორთა მიერ
წარმოდგენილი სახით

გადაეცა წარმოებას 1.02.2022 წ. ხელმოწერილია დასაბეჭდად 18.02.2022 წ. ოფსეტური ქაღალდის ზომა 60X84 1/16. პირობითი ნაბეჭდი თაბახი 15.75. ტირაჟი 50 ეგზ.

Verba volant, scripta manent

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

ISBN 978-9941-8-3809-5