საქართველოს ტექნიკური უნივერსიტეტი

დავით სირბილაძე

მომხმარებელთა მართვის განვითარებული სისტემების პროექტირება ბაზისური მომხმარებელთა მართვის სქემების ბაზაზე

წარმოდგენილია დოქტორის აკადემიური ხარისხის მოსაპოვებლად

სადოქტორო პროგრამა ინფორმატიკა

შიფრი 0401

საქართველოს ტექნიკური უნივერსიტეტი

თბილისი, 0175, საქართველო

2019 წ

საქართველოს ტექნიკური უნივერსიტეტი

ინფორმატიკისა და მართვის სისტემების ფაკულტეტი

ჩვენ, ქვემეორე ხელისმომწერნი ვადასტურებთ, რომ გავეცანით **დავით სირბილაძის** მიერ შესრულებულ სადისერტაციო ნაშრომს დასახელებით: მომხმარებელთა მართვის განვითარებული სისტემების **პროექტირება ბაზისური მომხმარებელთა მართვის სქემების ბაზაზე** და ვაძლევთ რეკომენდაციას საქართველოს ტექნიკური უნივერსიტეტის _ საუნივერსიტეტო სადისერტაციო საბჭოში მის განხილვას დოქტორის აკადემიური ხარისხის მოსაპოვებლად.

_ _ _ _ _ , _ _ _ _ _ 2019 წელი

ხელმძღვანელი: პროფესორი თ. ბახტაძე

რეცენტენტები: ______________________________

____

საქართველოს ტექნიკური უნივერსიტეტი

2019 წ

ავტორი: დავით სირბილაძე.

დასახელება: მომხმარებელთა მართვის განვითარებული სისტემების პროექტირება ბაზისური მომხმარებელთა მართვის სქემების ბაზაზე.

სადოქტორო პროგრამა: ინფორმატიკა

ხარისხი: დოქტორი

სხდომა ჩატარდა:

რეზიუმე

დღესდღეობით ჩვენი ყოველდღიური ცხოვრება გარდამტეხი ცვლილებების პირასაა. სულ უფრო და უფრო აქტუალური ხდება მანქანური სწავლების სფერო, რაც გავლენას ახდენს ჩვენს ყოველდღიურ ცხოვრებაზე. თანამედროვე ტექნოლოგიები უკვე იძლევა ხელოვნური ინტელექტის შექმნის საშუალებას. პროგრამირების ენების მეშვეობით არაერთი მანქანა გარდაიქმნა მოაზროვნე მოწყობილობად. ახლა უკვე შესაძლებელია საკუთარი რობოტის ყოლა, რომელიც დაგვეხმარება სხვადასხვა ამოცანის მაქსიმალური ეფექტურობით შესრულებაში, რაც მათი მრავალფეროვნების ზრდასთან ერთად დღითიდღე ამატებს მათსახეობებს.

მოცემულ დისერტაციაში განხილულია მომხმარებელთა მართვის გამარტივებული სისტემის დაპროექტების რამდენიმე ძირითადი მეთოდი და დეტალურადაა აღწერილი მათი დადებითი და უარყოფითი მხარეები.

ძირითადი სამუშაო შევასრულე PHP ძრავზე, რომლითაც დავუკავშირდი Microsoft Active Directory მომხმარებელთა და ობიექტთა ქსელურ ბაზას და ანალიზისა დაკვირვების შედეგად ავაწყე სრულყოფილი პროექტი ვებ აპლიკაციის სახით, რომელიც ინტეგრაციისა და სინქრონის მეშვეობით ამუშავებს უფლებების მიხედვით შემოსულ აქტიურ თანამშრომლებს.

თემა აქტუალურია, რადგან თანამედროვე კომპანიებს სჭირდებათ პროგრამული უზრუნველყოფა, რომელიც მარტივად და საიმედოდ იმუშავებს და ხელს შეუწყობს მათ მზარდ განვითარებასა და პროგრესს როგორც ტექნოლოგიურად, აგრეთვე ტექნიკურად.

გამოყენების ერთ-ერთი ყველაზე ფართო მაგალითი არის ინტრანეტი, რომელიც გაუადვილებს კომპანიის სამმართველოს ადმინისტრირების პროცესს და დაუზოგავს დროს მათ ქსელურ და სისტემურ ადმინისტრატორებს, რომელთა ფუნქციის ნაწილობრივ შესრულებას შეძლებს ჩვენს მიერ შემოთავაზებული პროგრამა.

AD-სთან ინტეგრირების მეშვეობით მომაქვს ყველა აქტიური და არააქტიური მომხმარებლების დეტალური პირადი ინფორმაცია, როგორიცაა, მაგალითად, მათი სახელი, მეილი, პოზიცია, დეპარტამენტი, ფილიალი, მობილური და შემდგომ ვაანალიზებ ამ ყველა შეგროვებულ ინფორმაციას და ვიყენებ ჩემს აპლიკაციებში.

მომხმარებლების წამოღება შეიძლება როგორც თითოეულად, აგრეთვე ჯგუფურად, რაღაცა კატეგორიით გაერთიანებული, მაგალითად ოთახებით ან სართულებით - როგორც არის დალაგებული AD იერარქიის სტრუქტურის ხე.

აგრეთვე შეგვიძლია რამდენიმე კომპანია გვქონდეს ერთ ბაზაში და ვმართავდეთ მათ ცენტრალიზებულად, რის გამოყენებასაც შევძლებთ მუშაობის პროცესის გამარტივებისთვის და სრულყოფისთვის.

მომხმარებლების ინფორმაცია საშუალებას გვაძლევს, რომ გავარჩიოთ ისინი ერთმანეთისგან და დავახარისხოთ პროგრამული უზრონველყოფის გამოყენებისთვის სხვადასხვა უფლებებით და

კატეგორიებით, რისი მიღწევაც შესაძლებელია ამომცნობი ხელოვნური ინტელექტის შექმნით.

როცა გვჭირდება მაგალთად გაყიდვების რეპორტი, რომელიც უნდა გამოიყენოს ფილიალის მენეჯერმაც, სესხის ოფიცერმაც, გაყიდვების აგენტმაც და ანალიტიკოსმაც, მაგრამ უნდა ჰქონდეთ განსხვავებული ხედვები - ამის გასაკეთებლად გამოვიყენებდით 4 სხვადასხვა რეპორტს მსგავსი სტრუქტურით და დავტვირთავდით ზედმეტად როგორც პროგრამისტებს, აგრეთვე მანქანას.

AD-სთან ინტეგრირება იძლევა იმის საშუალებას, რომ გავარჩიოთ, თუ რომელი მომხმარებელი არის ამჟამად ავტორიზებული და მისთვის ისეთი რეპორტის შექმნა, რომელიც ავტომატურად დაუგენერირებს მისთვის განკუთვნილ სასურველ შედეგს.

AD-დან წამოღებული ინფორმაციის გარჩევა შესაძლებელია პროგრამული კოდით, მისი საკუთარ ბაზაში ჩაწერა და გამოყენება შემდგომი საჭიროებისამებრ და მომდევნო ანალიზი.

პროგრამული მხარისთვის გამოყენებული მაქვს PHP სკრიპტული ენა, რომელსაც აქვს ყველა საჭირო მხარდაჭერის ძრავი, რომ დაუკავშირდეს LDAP ტექსტური მისამართებით მომხმარებლურ ბაზას და აწარმოოს მასზე სინქრონი და სხვადასხვა ოპერაციები.

PHP AdeLdap არის ბიბლიოთეკა, რომელსაც შეუძლია Active Directory სერვერიდან ინფორმაციის წამოღება, განახლება, რედაქტირება, წაშლა, მოდიფიკაცია, შესწორება, დაგენერირება, შევსება.

იგი არის საკმაოდ მძლავრი პროგრამა, თუმცა ხარვეზები მაინც გააჩნია. მისი მეშვეობით სრული ძალაუფლება და წვდომა ირთვება პროგრამული კოდიდან მომხმარებლების ცენტრალიზებულ ბაზაზე, რაც კომპანიისთვის საფრთხეს წარმოადგენს.

არასწორი ან შემთხვევითი კოდის გაშვებისას შესაძლებელია წაიშალოს მთელი მომხმარებლური ბაზა, მათი ინფორმაცია, დაიკარგოს ყველა დანართი.

პროგრამის გამოყენება შესაძლებელია ნებისმიერი მიზნით. კარგი მენეჯმენტით და დაფიქრებით მივალთ იმ დასკვნამდე, რომ მას შეუძლია კომპანიის ფუნქციონირების უკეთესობისკენ შეცვლა.

მასზე იწერება ისეთი პროცესი, როგორიცაა თანამშრომლების მოსვლა-წასვლა, გასვლები, პროგრამებში შესვლები, ღილაკებზე დაჭერები, გვერდების დათვალიერებები, რეპორტების გენერაცია და მრავალი სხვა.

მისი ბაზის ერთხელ შევსება გვაძლევს საშუალებას რომ იგი გამოვიყენოთ აბსოლუტურად ყველგან - მეილზე, კალენდარში, კომუნიკატორში, რეპორტინგ სერვერზე, კომპიუტერში, ბაზებში და საკუთარ აპლიკაციებში.

ერთი ცვლილება მოქმედებს ყველა პროდუქტზე, რომელსაც იყენებს მომხმარებელი. არ არის საჭირო ყველას ცალ-ცალკე განახლება.

Resume

Nowadays our daily life is on the edge of change. Information processing currently is becoming more and more actual problem, what interacts with our daily lives. Moderntechnologies already provide the development of artificial intelligence. With the help of programming languages, many machines got the ability to think. Now it is possible to have your own robot, which will help us to perform various tasks with maximal efficiency, thenumber and types of which increases every day with a very high speed.

This dissertation discusses several key methods of designing a simplified system of customer management and describes their advantages and disadvantages.

Basic work on the PHP engine that I connected with Microsoft Active Directory is a network of users and objects, and as a result of analyzing the process, you can create a complete project as a web application that collects the active employees with their rules through integration and synchronization.

The theme is relevant because modern companies need software that is easy and reliable and will enhance their development and progress both technologically and technically.

One of the most widely used examples is the intranet that facilitates the management of the company's division and saves time for their network and system administrators whose part of the function is partially implemented by us.

By integrating with AD I have detailed personal information of all active and inactive users such as their name, email, position, department, branch, mobile and further analyzing all these information and use it in my applications.

Consumers can be collected either in groups or in groups, such as rooms or floors, as is the structure of the hierarchy structure of the AD hierarchy.

We can also have several companies in a single database and we use them centrally, which can be used to simplify the work process and to improve.

Users' information allows us to discern each other and draw a variety of rights and categories for the use of software accessibility, which can be achieved by creating artificial intelligence.

When we need a sales reporse, which should be used by a branch manager, a loan officer, a sales agent, and an analyst, but we should have different visions - to do that, we would use 4 different reports with a similar structure and loaded over as a programmer and a machine.

Integrating with AD allows you to find out which user is currently authorized and creating a report that automatically generates the desired result.

The information received from the AD can be distinguished with acode, recording and using it in its own database and further analysis and subsequent analysis.

For the software side I have a PHP script language that has all the supporting engine to communicate with the LDAP text addresses to the user base and to synchronize it with different operations.

PHP AdeLdap is a library that can fetch, update, edit, delete, modify, generate, and fill out the information from Active Directory Server.

It is a very powerful program, but still has drawbacks. With full power and access it processes the software code from the centralized base of users, which is a threat to the company.

When running incorrect or random code, it may be possible to remove all user base, their information, and all the attachments.

The program can be used for any purpose. With good management, you will come to the conclusion that it can change the functionality of the company.

It is written on a process such as the arrival of employees, departures, logging in programs, clicking on buttons, browsing pages, generating reports, and much more.

Once again, its base allows us to use it absolutely everywhere - mail, calendar, communicator, reporting server, computer, bases and in your own applications.

One change affects all products that the user uses. No need to update everyone separately.

შინაარსი

სურათების ნუსხა

კოდების ნუსხა

აბრევიატურების ნუსხა

AD - Active Directory.
MS - Microsoft.
BI - Business Intelligence.
SSRS - SQL Server Reporting Services.
Hr - Human Resources.
Admin - Administrator.
v - Version.
N - Number.
LDAP - Lightweight Directory Access Protocol.
P.O. - Post Office.
SQL - Structured Query Language.
MVC - Model, View, Controller.
JS - Javascript.
DB - Database.
OS - Operational System.
DNS - Domain Namespace.
DHCP - Dynamic Host Configuration Protocol.
WAN - Wide Area Network.
IT - ინფორმაციული ტექნოლოგიები.
VB - Visual Basic.
DOB - Day of Birth.
UCS - Univention Corporation Server.
IBM - International Business Machines.
e - electronic.
SASL - Simple Authentication and Security Layer.
DSML - Directory Services Markup Language.
SOAP - Simple Object Access Protocol.
SSO - Single Sign-On.
NTDS - NT Directory Services.

შესავალი

დღევანდელი თანამედროვე სამყარო ტექნოლოგიური განვითარების პიკზე დგას. პროგრამებისა და აპლიკაციების აწყობა უკვე მულტიმომხმარებლური მიდგომითაა საჭირო. ყველაფერი ეყრდნობა მსოფლიო ქსელსა და ინტერნეტს. შესაბამისად, აპლიკაცია, რომელსაც უშვებენ კომპანიები, უნდა იყოს უნივერსალური, უნდა იყოს ყველასთვის ხელმისაწვდომი და გასაგები. ხალხის ყოველი კატეგორიისთვის ცალკე აპლიკაციის დაწერა არის დიდი დროისა და რესურსის კარგვა, რისთვისაც ინერგება სისტემები, რომლების მიხედვითაც ხდება მომხმარებლების მახასიათებლების ამოცნობა და მათი გარჩევა ერთმანეთისგან.

ნაშრომშიგანხილულია მიდგომები მომხმარებლებისა და მათი მახასიათებლების ამოსაკითხად სწრაფად, მარტივად და საიმედოდ. თემაში შემოთავაზებულია მათი სუსტი და ძლიერი მხარეების გამიჯვნა ერთმანეთისგან და სწორი მიმართულება საკუთარ აპლიკაციაში დანერგვისთვის.

მაგალითად, მომხმარებელს, რომელსაც უყვარს ლურჯი ფერი და მომხმარებელს, რომელსაც უყვარს მწვანე ფერი, არ უნდა მივცეთ აპლიკაცია, რომელიც არის წითელი ფერის. ასევე გამორიცხულია მაგათთვის ცალ-ცალკე ფერის აპლიკაციის შექმნა. ყოველ მომხმარებელს უნდა ჰქონდეს საშუალება, რომ თვითონ აირჩიოს მისთვის სასურველი ფერი, რა ქმედებაც უნდა დაიმახსოვროს აპლიკაციამ. მოხმარებელი ერთხელ რომ აირჩევს პარამეტრს, მეორედ შესვლისას იმ არჩეული პარამეტრით უნდა ჩაიტვირთოს პროგრამა.

მომხმარებლების ერთმანეთისგან გამორჩევისთვის არსებობს რამდენიმე მიდგომა. ერთია საკუთარი მომხმარებლების ბაზის შექმნა და მასთან მუშაობა, მეორე კი არის რამე არსებული მომხმარებლური სისტემის ინტეგრირება საკუთარ აპლიკაციაში. ორივე მიდგომას აქვს თავისი პლიუსებიც და მინუსებიც.

თუ საკუთარი ბაზის აწყობა გვინდა, მაშინ ბევრი მუშაობა მოგვიწევს და ყველაფერი უნდა გავხადოთ დამოკიდებული მასზე. ყველა ნაწილი იქნება მიბმული მასზე და იქნება მუდმივი ჩაწერა-წაკითხვა და თუ რამეს არასწორად გავაკეთებთ, მაშინ ჩამოეკიდება მთელი აპლიკაცია ან როგორც მინიმუმ რამე ნაწილი იმუშავებს არასწორად.

შესარჩევია შესაფერისი და სათანადო ბაზა, სადაც გასათვალისწინებელია აპლიკაციის მოთხოვნები, მომხმარებლების მოცულობა, რაოდენობა, ლოკაცია, სეგმენტი და ა.შ.

თუ ავირჩევთ MySQL ბაზას - იგი სწრაფია, მაგრამ თუ არ გავმართავთ, არ არის სტაბილური, შეიძლება გაჭედოს, ვერ დაამუშავოს მონაცემები და შეჩერდეს მთელი პროგრამა მაგის გამო. თუ ავირჩევთ MSSQL ბაზას, მაგას სჭირდება დიდი რესურსი, მაგრამ საიმედოობაც მეტია. ასევე მინუსია, რომ წერა მეტი სჭირდება. გასათვალისწინებელია აპლიკაციის ძრავიც. მაგალითად თუ გვაქვს PHP სერვერი, მას უჭირს MS ბაზებთან მუშაობა და მაგის მოგვარებაზე ცალკე დრო დაიხარჯება.

თუ გვინდა რამე არსებული სისტემის გამოყენება და ჩასმა ჩვენთან, მანდ უნდა დავფიქრდეთ პლატფორმაზე. არის როგორც ფასიანი, ასევე უფასო ვერსიები. უფასო ვერსიები შეზღუდულია შესაძლებლობებში. თუ გაკმაყოფილებთ ეგ ლიმიტები, მაშინ შეგვიძლია დაპროგრამების დაწყება.

ასევე არსებობს შერეული სისტემები, რომლებშიც ინტეგრირებულია როგორც გარე სამომხმარებლო ბაზები, ასევე შიდა სერვისები, რომლებსაც იყენებენ ინფორმაციის გადამუშავებისთვის და დახარისხებისთვის. თუ არის შესაძლებლობა, მაშინ ეგ ყოველმხრივ უფრო მოქნილია და გამოსადეგი. წვდომები ნაწილდება. ზოგი იმართება გარე სისტემის მიერ, ზოგი კი - იწერება პერსონალურ ბაზებში. ამ ორის ერთად შერწყმით კი გვევლინება ერთი დიდი სრულყოფილი პროგრამა მძლავრი სქემის სახით.

ერთ-ერთი ყველაზე ფართოდ მოხმარებადი სისტემა არის ინტრანეტი, რომელსაც იყენებს ფაქტიურად ყველა კომპანია. იგი ძირითადად გამოიყენება და მოიხმარება შიდა კომუნიკაციისთვის და

ფაილების გაცვლისთვის. შესაბამისად, სისტემა უნდა მუშაობდეს ლოკალურად და არ უნდა იყოს მისი ქსელს გარეთ გასვლა და ინფორმაციის გაჟონვა.

უსაფრთხოება პრიორიტეტი არ არის ასეთ სისტემებში, მაგრამ როცა კომპანია დიდია, შესაძლებელია რომ ზოგიერთი თანამშრომელი აღმოჩნდეს ბოროტმოქმედი და პროგრამის სისუსტეები გამოიყენოს ორგანიზაციის საწინააღმდეგოდ.

თავი 1. ლიტერატურის მიმოხილვა

სანამ დავიწყებდი თემის წერას, ბევრი ვეძიე მსგავს ნაშრომებს, მაგრამ მიმსგავსებული პუბლიკაციები ინტერნეტ სივრცეში ფაქტიურად არ არსებობს და არ აქვს არავის ამ დონემდე სრულყოფილად მიყვანილი სისტემა.

არის რაღაც ფრაგმენტები, პატარა ბიბლიოთეკები, რომლებიც დამეხმარნენ ჩემი თემის დასრულებასა და დახვეწაში, მის ამუშავებაში და რეალიზაციაში.

ერთ-ერთი მთავარი პროგრამა, რაზე დაფუძნებითაც მაქვს გაკეთებული ნაშრომი არის PHP AdeLdap, რაც არის ძალიან მნიშვნელოვანი იარაღი პროგრამისტებისთვის. მისი მეშვეობით შესაძლებელია PHP ძრავიდან დაკავშირება Active Directory ბაზებთან და მომხმარებლებისა და ობიექტების შესახებ ინფორმაციის წამოღება და ჩაწერა. [2]

აგრეთვე გამოყენებული მაქვს Microsoft Active Directory ქსელური ადმინისტრირების ბაზა, რომელსაც აქვს მრავალ პროგრამასა და პროდუქტთან ინტეგრირების საშუალება მარტივი მიდგომებით. ვინაიდან Microsoft-ს აქვს მოწინავე ოპერაციული სისტემა, სწორედ მაგიტომ ავირჩიე მასზე დაფუძნებული სისტემის აწყობა და დანერგვა, რაზეც მაქვს დაწერილი ჩემი სადისერტაციო ნაშრომი.

მითითებული პროგრამული უზრუნველყოფის შექმნის თაობაზე ვარ გამოსული სტუდენტთა საერთაშორისო N86 ღია სამეცნიერო კონფერენციაზე, რომელზეც ახსნილი მაქვს, თუ როგორ არის შესაძლებელი, რომ უბრალო ბაზა ვაქციოთ ხელოვნურ ინტელექტად და როგორ ვასწავლოთ მას ობიექტების ამოცნობა და დახარისხება, რითიც იქმნება ერთგვარი ხელოვნური ნეირონული ქსელი, რისი ანალიზიც მაქვს ჩატარებული ამ თემაში. [11]

მაქვს გამოცემული რამდენიმე პუბლიკაცია, სტატია. ხელოვნური ნეირონული ქსელი ხომ დღესდღეობით ყველაზე აქტუალური თემაა, რაზეც მუშაობენ სხვადასხვა დარგის მეცნიერები და ატარებენ უამრავ

კვლევებს. ამის შესახებ დეტალურად მაქვს ჩამოყალიბებული საერთაშორისო სტატიაში, რომელიც გამოვაქვეყნე საქართველოს ტექნიკური უნივერსიტეტის არჩილ ელიაშვილის მართვის სისტემების ინსტიტუტის N21 ჟურნალში 2017 წელს. [12]

ხოლო 2018 წელს გამოვაქვეყნე საერთაშორისო სტატია უკვე კონკრეტულად მომხმარებლების ამომცნობი სისტემის ინტეგრირებაზე აპლიკაციაში, რაც მივიღე დიდი კვლევების შედეგად. ჩემი ეს პუბლიკაცია გამოვიდა საქართველოს ტექნიკური უნივერსიტეტის არჩილ ელიაშვილის სახელობის მართვის სისტემების ინსტიტუტის N22 ჟურნალში მორიგიშრომათა კრებულის სახით. [13]

ამავე 2018 წელს გამოვაქვეყნე ორი სტატია ჩემს კველვებსა და შედეგებზე დაყრდნობით მართვის ავტომატიზირებული სისტემების საერთაშორისო შრომათა კრებულში. ჩემი მისი პირველი თემა ეხებოდა მონაცემთა გრაფიკულ ვიზუალიზაციას Microsoft Power BI-ს მეშვეობით, რასაც აქტუალურად ვიყენებ ჩემს პროგრამებსა და პროექტებში, რადგან მისი დახმარებით მარტივად კეთდება დიდი მონაცემების გრაფიკულად გამოსახვა, რაც გვაძლევს იმის საშუალებას, რომ დავზოგოთ საკმაო დრო და რესურსი და გავაუმჯობესოთ ჩვენი კომპანიისა თუ ორგანიზაციის საქმიანობა. [14]

რაც შეეხება მეორე სტატიას, იგი გამოვაქვეყნე ინგლისურ ენაზე, რათა უცხოელ მკვლევარებსაც შეეძლოთ მისი გარჩევა და გამოყენება. თემა ეხებოდა მანქანური დასწავლის მეთოდების გამოყენებას სხვადასხვა სფეროებში, სადაც აღწერილი მაქვს ძირითადი პრინციპები და დებულებები, რაც უნდა ვიცოდეთ ხელოვნურ ინტელექტთან მუშაობისას და მისი დამუშავებისას, რაც გვეხმარება ეფექტურად და ეფექტიანად წარმოვაჩინოთ ჩვენი მექანიზმი ხელოვნურ ნეირონულ ქსელში. [14]

პროგრამის ვიზუალიზაციისთვის ვიყენებ Bootstrap ძრავს, რომელიც არის მსუბუქი, თანამედროვე და ყველა პლატფორმაზე მორგებული,

როგორც მობილურებზე, ასევე პლანშეტებზე, ლეპტოპებზე და პერსონალურ კომპიუტერებზე.

ამ ყველაფრის გაკეთებისთვის კი არ არის საჭირო უამრავი დროის დახარჯვა. არის მარტივი ფუნქციები, რომლების გამოყენებითაც ყველაფერი გაკეთდება ავტომატურად.

თავი 2. კვლევა, შედეგები და მათი განსჯა

დღესდღეობით ყველა კომპანია და ორგანიზაცია ფიქრობს განვითარებასა და პროგრესზე როგორც თავისი კომპანიის, აგრეთვე მისი შიდა ინფრასტრუქტურის.

ყველას სჭირდება შიდა საკომუნიკაციო და ფაილების გაცვლის საშუალება, რისთვისაც ზოგი ყიდულობს პროგრამებს, ზოგი - ლიცენზიებს, ზოგი კი საკუთარი რესურსით წერს მათ.

საკუთარი პროგრამის ქონა რა თქმა უნდა კარგია, მაგრამ მას სჭირდება ბევრი მოვლა და სანამ შედეგამდე მივალთ, ძალიან ბევრი პუნქტია გასავლელი. თუმცა ყველა მოდულის დაწერა აუცილებელი არ არის. როდესაც ვმუშაობთ Microsoft პროდუქტთან, მას აქვს ძალიან მძლავრი ცენტრალიზებული ბაზა Active Directory, რომლის გამოყენებაც შეგვიძლია სისტემებში ავტორიზაციისა და აუთენთიფიკაციისთვის. [1]

მე დავწერე აპლიკაცია, რომელიც PHP-ის მეშვეობით უკავშირდება AD ბაზას და ასინქრონებს მომხმარებლებთან ნებისმიერ სისტემას. ამ მიდგომის მეშვეობით შესაძლებელია ნებისმიერი პროექტის ავტორიზაციის გატარება Windows მომხმარებლის სახელისა და პაროლით ისევე, როგორც მისი მონაცემების მართვა და დაჯგუფება სხვა მომხმარებლებთან, რომლებიც არიან მსგავს პოზიციაზე, დეპარტამენტში, ფილიალსა ან ორგანიზაციულ სტრუქტურაში.

მიდგომა, რომელიც გამოვიყენე, ბევრ დაწესებულებას გაუმარტივებს შიდა მუშაობის პროცესს. მე უკვე დანერგილი მაქვს იგი საქართველოს რამდენიმე დიდ ორგანიზაციაში, რომლებიც წარმატებით ფუნქციონირებენ ბაზარზე.

მისი მეშვეობით უამრავი რამის გაკეთებაა შესაძლებელი, რეებსაც დეტალურად განვიხილავ მოცემულ სადისერტაციო ნაშრომში.

ძირითადი კვლევა, რომელიც ჩავატარე, ეყრდნობოდა იმ ტექნოლოგიების შესწავლას, რომელიც არსებობს ბაზარზე და დავასკვენი,

რომ მსგავსი სირთულისა და მასშტაბის პროგრამები ინტერნეტში არ დევს, რის გამოც გადავწყვიტე საკუთარი შედეგი მიმეღო.

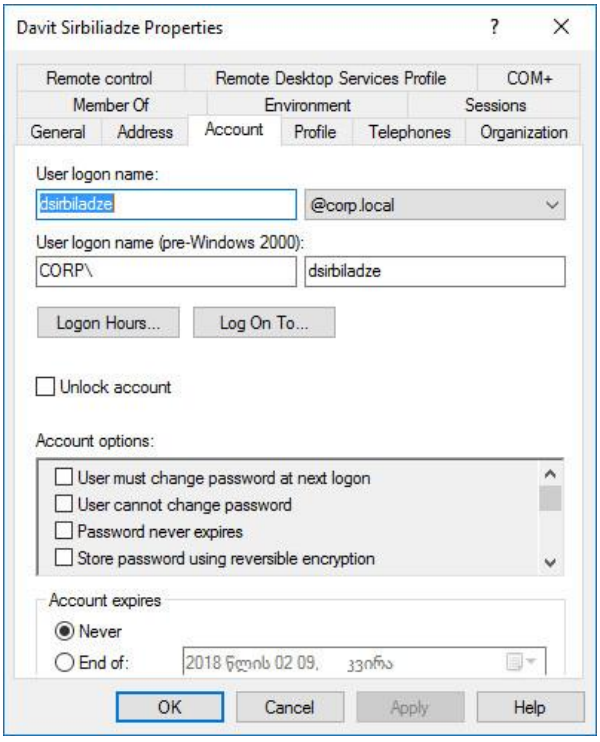
ვინაიდან ეს მიდგომა საჭიროებს ბევრ პროგრამულ ჩარევას, კომპანიები თვითონ ერიდებიან მსგავს პროცესში შესვლას, მაგრამ შედეგების გაუმჯობესებისთვის მაინც ღებულობენ გადაწყვეტილებებს, რომ დანერგონ ეს სისტემა.

მთავარი მოდელია - Microsoft Active Directory-ს სრულყოფა, შემდგომ სერვერიდან მასთან დაკავშირება, მონაცემების გადმოქაჩვა და ინტეგრაცია ყველა აპლიკაციაში. [3]

თავი 2.1. Microsoft Active Directory ბაზა

უპირველესია მოხმარებლების დახარისხება დეპარტამენტებისა და ფილიალების მიხედვით. ყველა პოზიციას თავისი წვდომა და უფლებები უნდა ჰქონდეს. ამის საკუთარი სახსრებით გათვალისწინება და დაწერა დიდ რესურსს საჭიროებს.

ძირითადად კომპანიები Windows პლატფორმებზე მუშაობენ და შესაბამისად სერვისიც უნდა ამოვირჩიოთ მასთან თავსებადი. საუკეთესო არჩევანია, როცა მოხმარებლები ცალკე კი არ შეიქმნება, არამედ გაივლიან ავტორიზაციას ავტომატურად - კომპიუტერის სახელისა და პაროლის მეშვეობით. ამის საშუალებას უკვე იძლევა თანამედროვე პროგრამები და სისტემები.



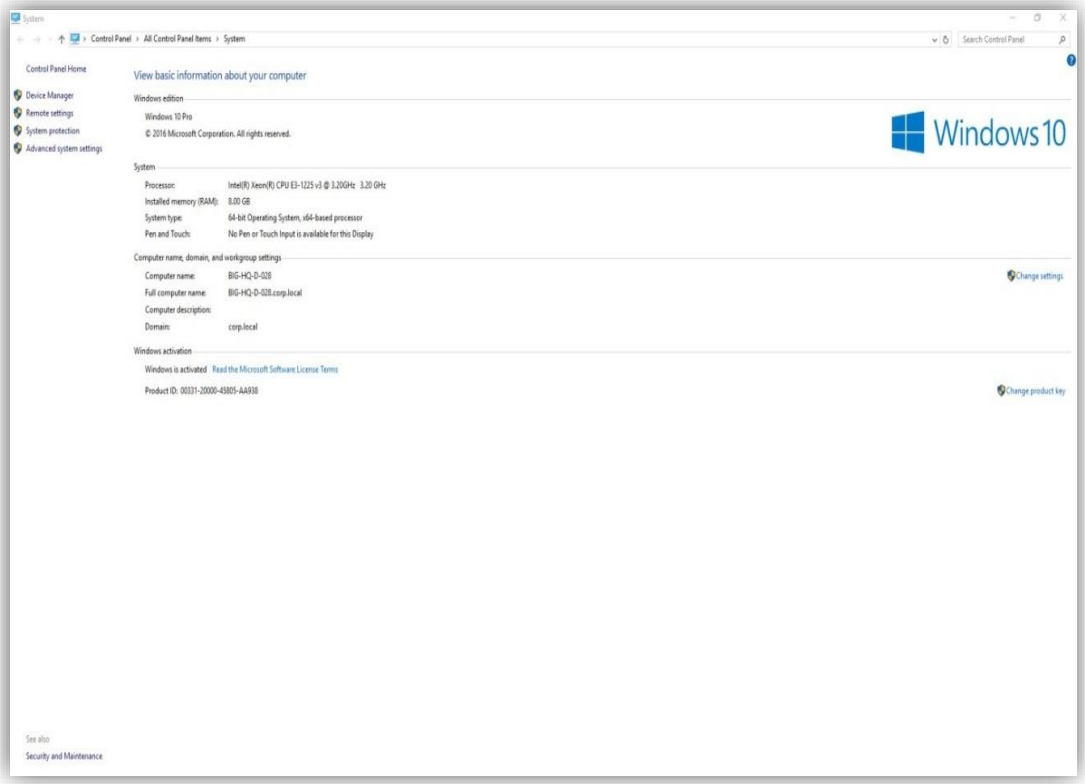
სურათი 1. Active Directory დომეინური ინტეგრაცია.

ყველა მომხმარებელი მოიაზრება ერთი დიდი

დომეინის ქვეშ, რომლებსაც მართავს ადმინისტრატორი. ეს შეგვიძლია გამოვიყენოთ ჩვენს სასიკეთოდ. სისტემური ადმინები მომხმარებლების მენეჯმენტისთვის იყენებენ Active Directory-ს (აბრევიატურა - AD), სადაც შესაძლებელია თანამშრომლებზე ბევრი დამატებითი ინფორმაციის შევსება, როგორიცაა მათი მისამართი, ტელეფონის ნომერი, ფოსტა, სახელი/გვარი და ა.შ.

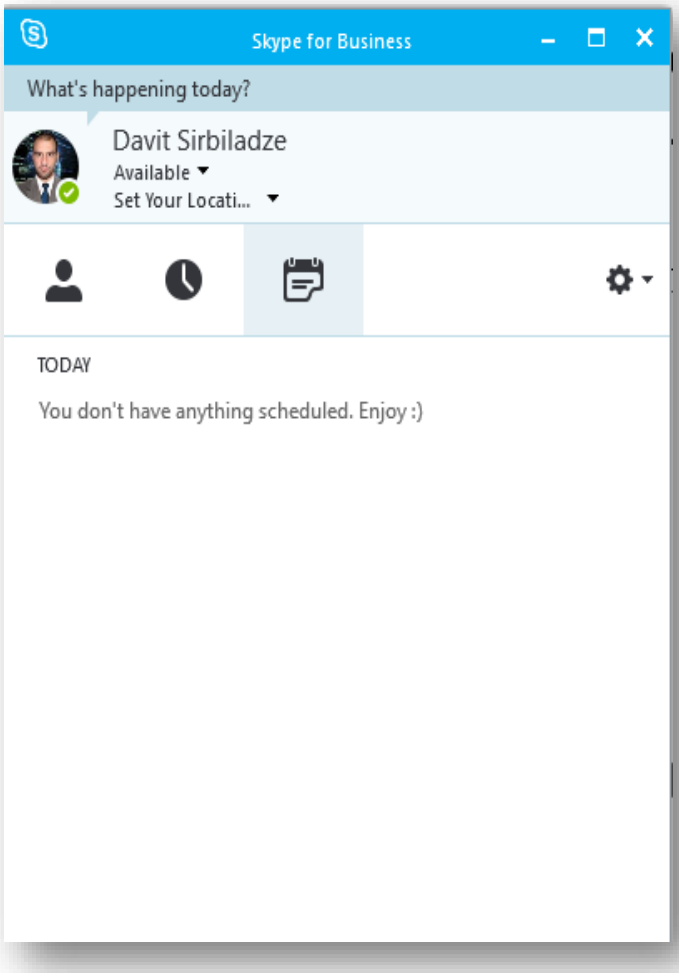
ეს ყველაფერი ინტეგრირებადია სპეციალური ბიბლიოთეკისა და დაპროგრამების სკრიპტული ენის მეშვეობით. ვასინქრონებთ ბაზას და გვაქვს ყველა ის მომხმარებელი, რომელიც რეგისტრირებულია და ვარჩევთ მარტივად თითოეულ მათგანს და შესაბამისად ვურთავთ უფლებებს.

იმისთვის, რომ შევძლოთ ქსელში მყოფი ყველა კომპიუტერის მართვა და დახარისხება, ანუ ადმინისტრირება, ისინი უნდა იყონ ერთი დომეინის ქვეშ, რომელსაც დავუკავშირდებით AD-თი.



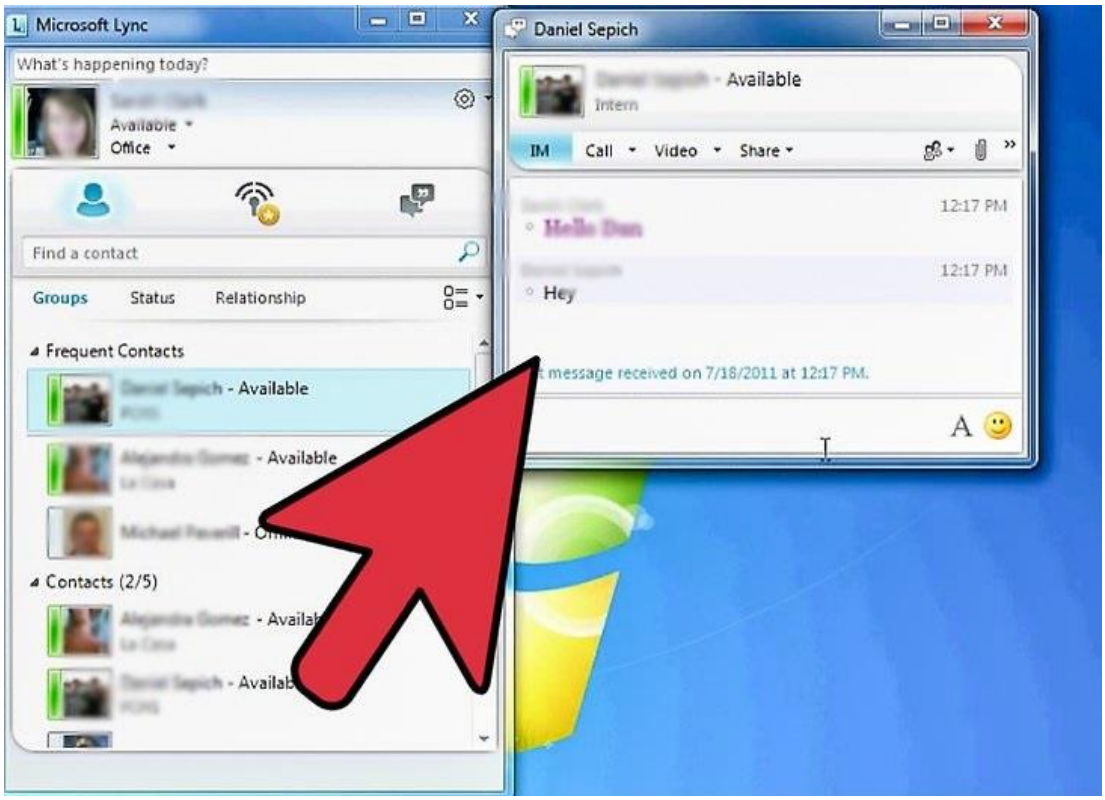
სურათი 2. Windows დომეინური ინფორმაცია.

AD მომხმარებლების ბაზის ინტეგრირება შესაძლებელია მრავალ პროგრამაში. იგი გამოიყენება როგორც Outlook ფოსტაში, ასევე კომუნიკატორში - Microsoft Lync, ასევე Microsoft-ის სხვა გარემოებშიც, როგორიცაა მაგალითად SSRS.



სურათი 3. Microsoft Lync.

თვითონ Microsoft Lync-იც შესაძლებელია დაინტეგრირდეს და გაეშვას Skype-დან სპეციალური კონფიგურაციის საფუძველზე, რომელსაც ფაქტიურად ყველა ის ხედვა აქვს, რაც Lync-ს.

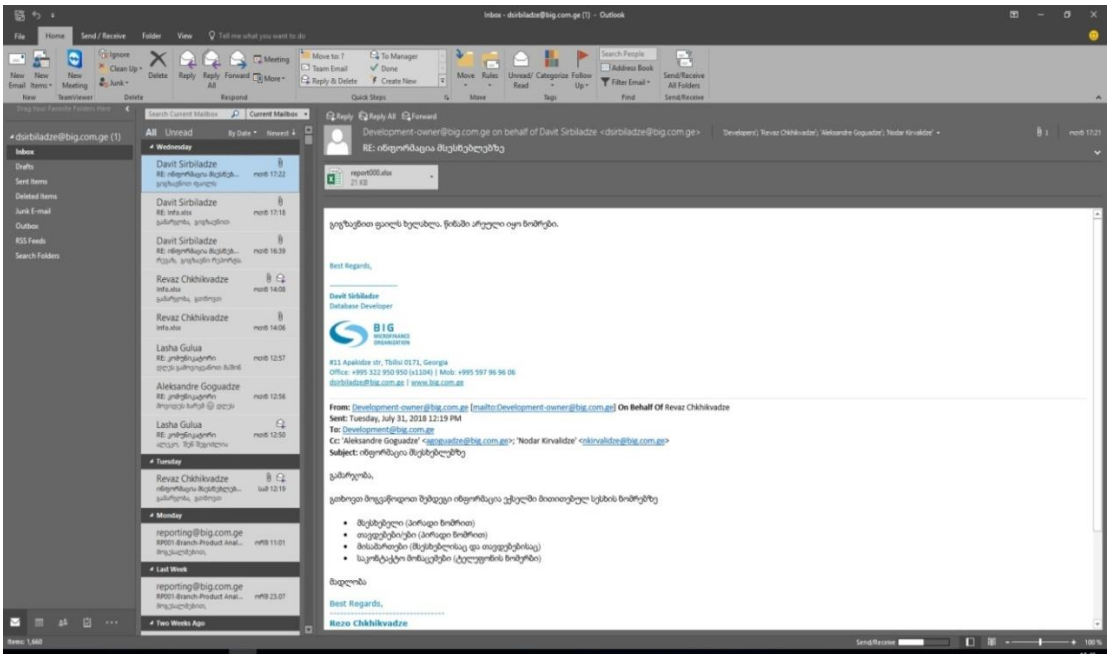


სურათი 4. Microsoft Lync ინტეგრირებული Skype-ში.

Microsoft Lync არის საკომუნიკაციო პროგრამა, რომელში ავტორიზების მერე ჩნდება საშუალება ყველა ოდესმე შემოსული კოლეგის მოძებნის, რომლებიც არ არიან გაუქმებული.

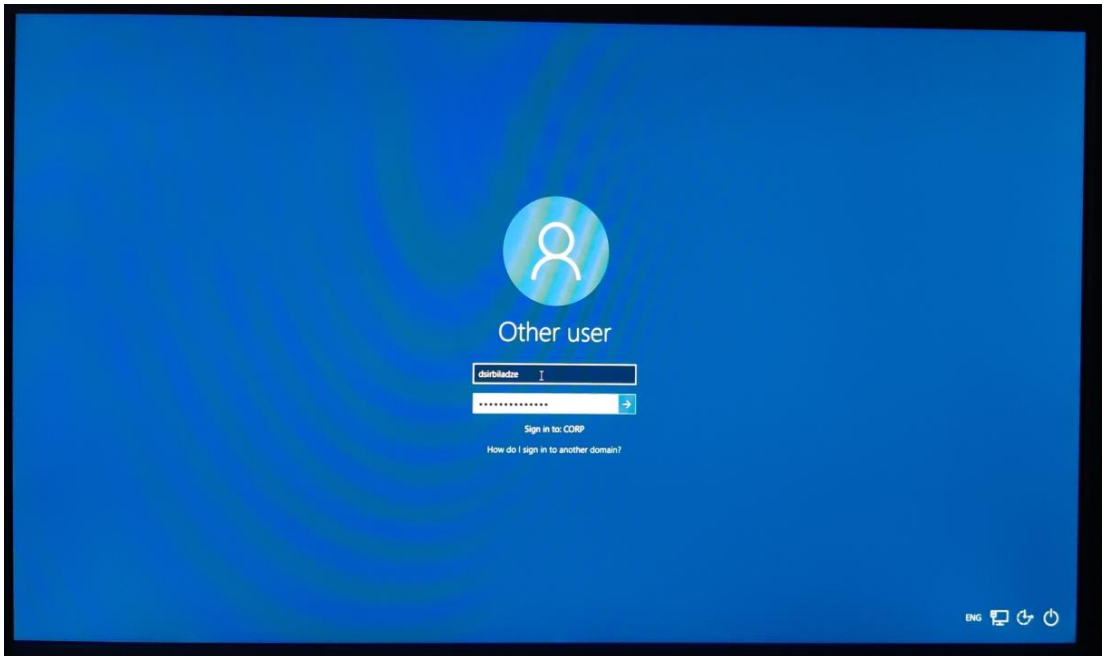
მას აქვს ბევრი ფუნქცია - როგორც ჯგუფური ზარების წამოწყების შესაძლებლობა, ასევე ვიდეო კონფერენციები, ფაილების გადაგზავნა, მიმოწერის ისტორიის ნახვა, რომელიც ინახება და სინქრონდება მთავარი სერვერიდან.

თუმცა AD-ზე წვდომა ყველას არ უნდა მიეცეს. მისი სტრუქტურის ნახვის და შეცვლის უფლება უნდა ჰქონდეს მხოლოდ სისტემურ ადმინისტრატორს და პროგრამისტს, რადგან იგი მოიცავს დიდ და კონფიდენციალურ ინფორმაციას.



სურათი 5. Microsoft Outlook ინტეგრირებული AD-სთან.

ამ ბოლო სისტემას - Microsoft SQL Server Reporting Services-ს სჭირდება ერთი ლინკის გაწერა დასაკავშირებლად, სადაც არის AD-ს სერვერის მისამართი, საიდანაც წამოიღებს შევსებულ ინფორმაციას და გამოიყენებს მას რეპორტების დახარისხებისთვის და შევსებისთვის.



სურათი 6. Windows ავტორიზაცია AD-დან.

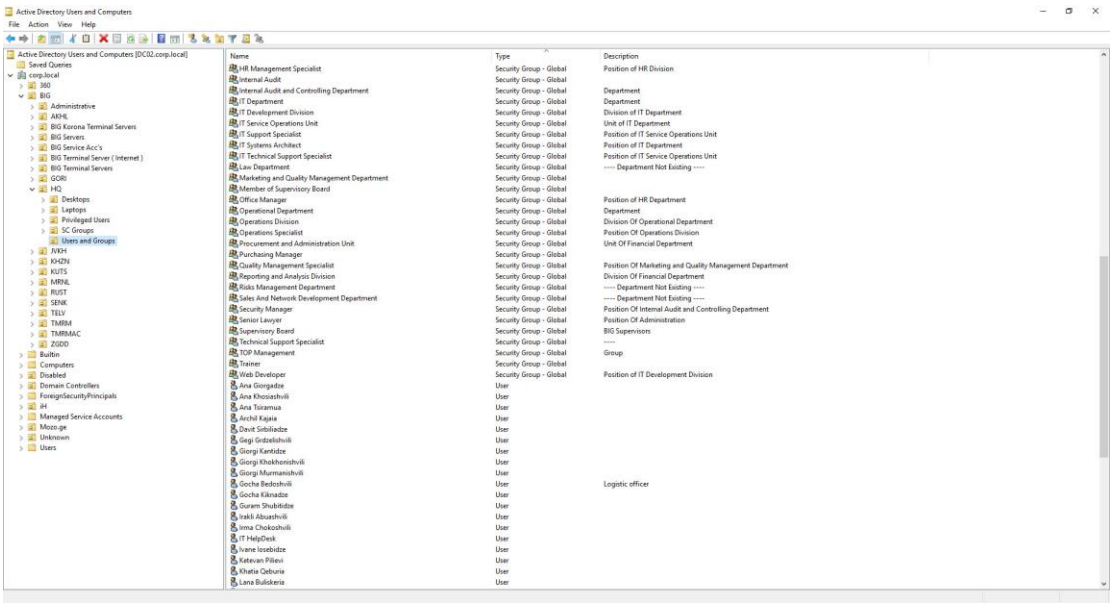
AD მართავს აბსოლუტურად ყველა კომპონენტს - როგორც მომხარებლის სურათს, ასევე მის გრაფიკს, სამუშაო დროებს, წასვლა-მოსვლებს, ტექნიკის ფლობებს, სტატუსს,..

სურათი 7. Microsoft Active Directory.

მისი მეშვეობით მარტივად ხდება ინფორმაციის გაცვლა სხვადასხვა აპლიკაციებს შორის, რადგან ყველა დაფუძნებული იქნება ერთ ძირითად ბაზაზე - AD-ზე. მაგალითად, მომხმარებელი თუ აღნიშნავს Windows Calendar-ში, რომ ოთხშაბათს ისვენებს, ეგ სინქრონდება მის საფოსტო ყუთთან და მის კოლეგებს უჩვენებს, რომ აღნიშნული პიროვნება ოთხშაბათს არის ოფისს გარეთ. მსგავსი პრინციპების დანერგვა შესაძლებელია მრავალ პროგრამას შორის.

იმისთვის, რომ შევძლოთ AD-სმართვა, საჭიროა მისიადმინისტრირების პროგრამის დაინსტალირება კომპიუტერში.

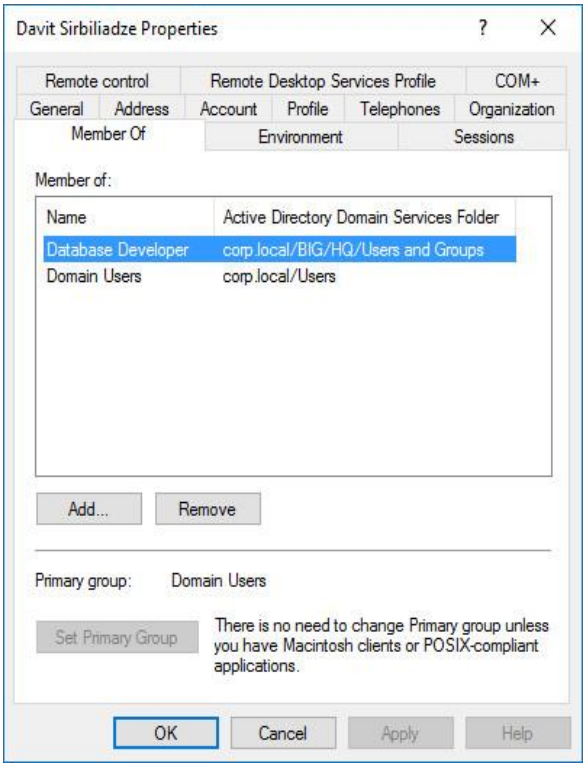
AD Browser გამოიყურება შემდეგნაირად:



სურათი 8. Microsoft Active Directory Users and Groups.

AD-ს აქვს ხის იერარქია. ყველაფერი იშლება მშობლიდან შვილისკენ. თითოეულ განშტოებას ჰყავს ერთი მშობელი და შეიძლება ჰყავდეს რამდენიმე შვილი. თითოეულ განშტოებაზე დაჭერით გამოდის მისი შიგთავსი ობიექტები, რომლებიც შეიძლება იყოს როგორც ჯგუფი, ასევე ელემენტი.

ჯგუფები არის რამდენიმე ელემენტის ნაკრები, რომელიც რაღაცა წინასწარ განსაზღვრული ლოგიკით აერთიანებთ მათ.

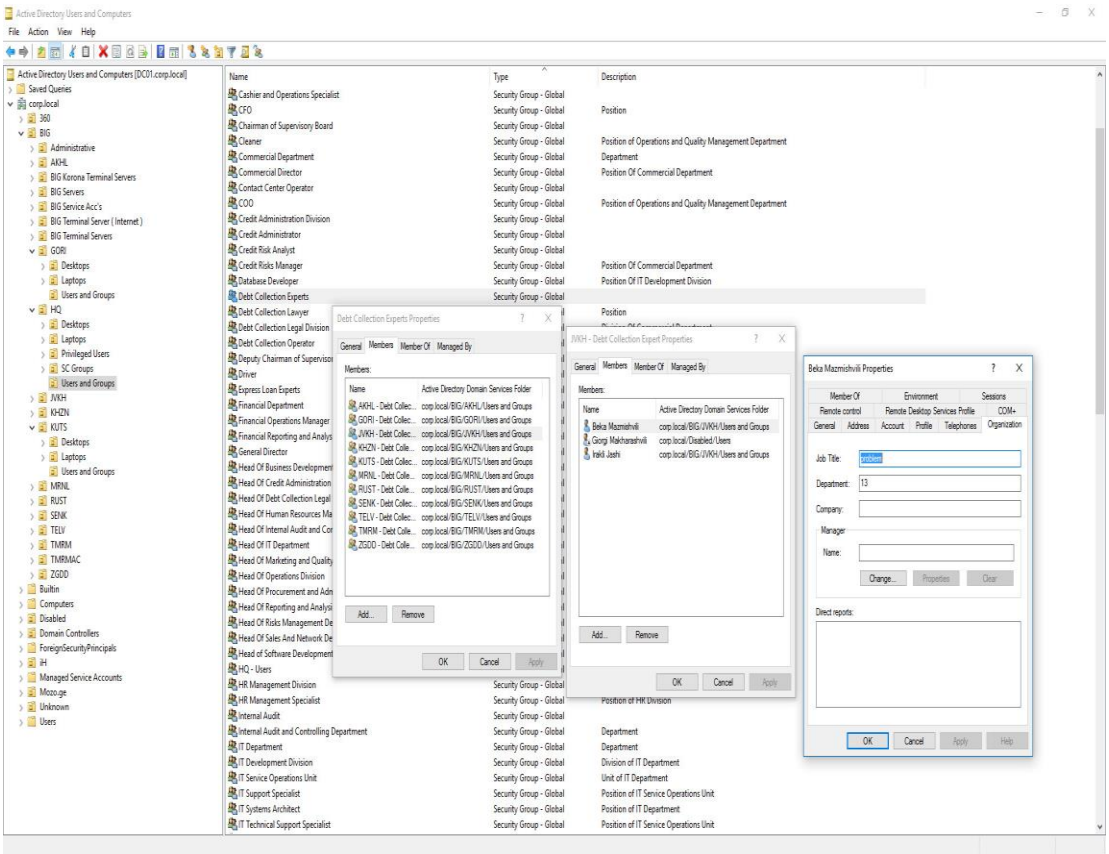


სურათი 9. MemberOf ჩანართი AD-ში.

"მშობელი" AD-ს ენაზე არის MemberOf, რომელიც მიუთითებს ელემენტის ორგანიზაციული სტრუქტურის ერთეულზე.

შეიძლება ერთდროულად იყო რამდენიმე ჯგუფის წევრი, როგორც ეგ სურათზეა ნაჩვენები - Domain Users და Database Developer. ყველა ელემენტი აუცილებელია იყოს რამე ჯგუფის წევრი.

ნებისმიერი ელემენტისთვის შესაძლებელია როგორც მისი მშობლის ნახვა, ასევე მისი შვილის ნახვა. მშობლიდან მერე გადავალთ მის მშობელზე და ა.შ. სანამ არ მივალთ მთავარ განშტოებამდე. ანუ არის ორი ტიპის ობიექტი - საქაღალდე ან User (მომხმარებელი).

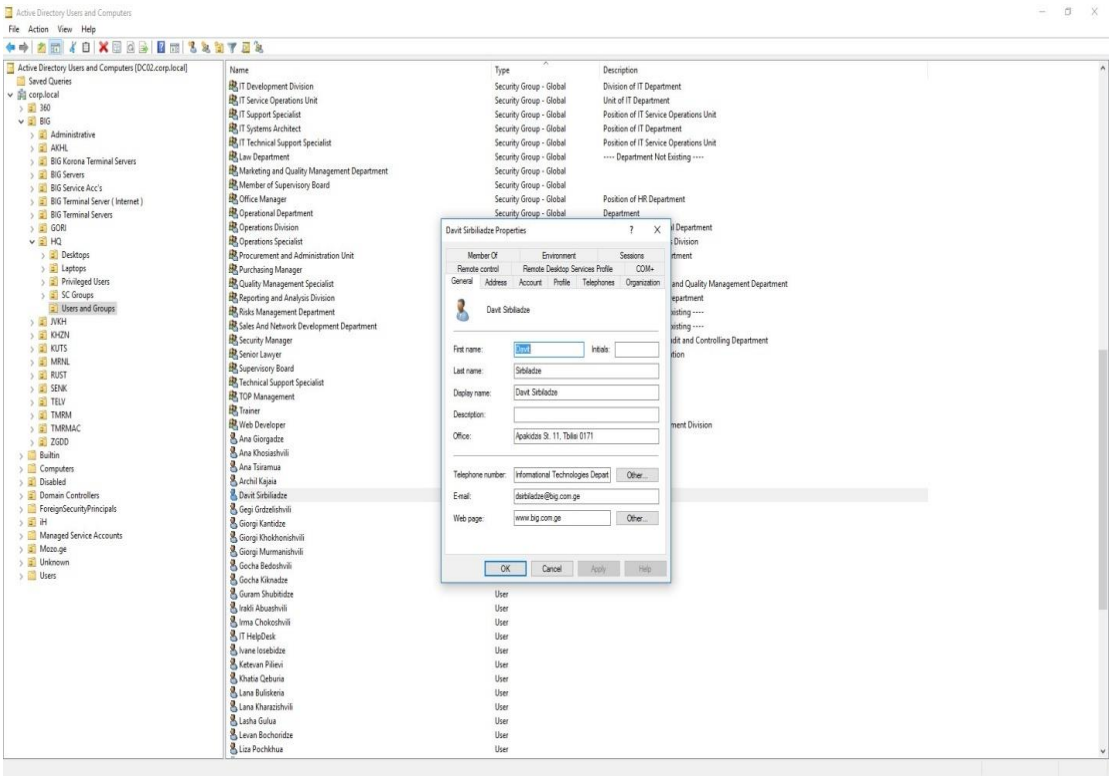


სურათი 10. AD იერარქია.

განვიხილოთ მათი ჩანართები. იგი შედგება რამდენიმე ფანჯრისგან (ტაბისგან), რაც ქვემოთ მოცემულ სქრინშია ნაჩვენები. თავიდან აქტიურია General ტაბი, სადაც წერია მთავარი ინფორმაცია:

- First name - სახელი,
- Last name - გვარი,

- Initials - ინიციალები,
- Display name - საჩვენებელი სახელი,
- Description - აღწერა,
- Office - ოფისი,
- Telephone number - ტელეფონის ნომერი,
- E-mail - ელექტრონული ფოსტა,
- Web page - ვებ გვერდი.

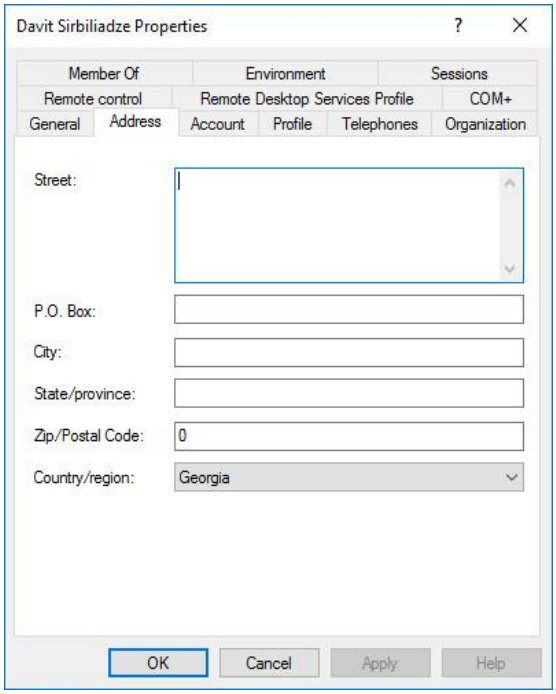


სურათი 11. AD ელემენტის დეტალური ინფორმაცია.

მეორე ფანჯარა არის Address (მისამართების), რომელიც შედგება შემდეგი ელემენტებისგან:

- Street - ქუჩა,
- P.O. Box - საფოსტო ყუთი,
- City - ქალაქი,

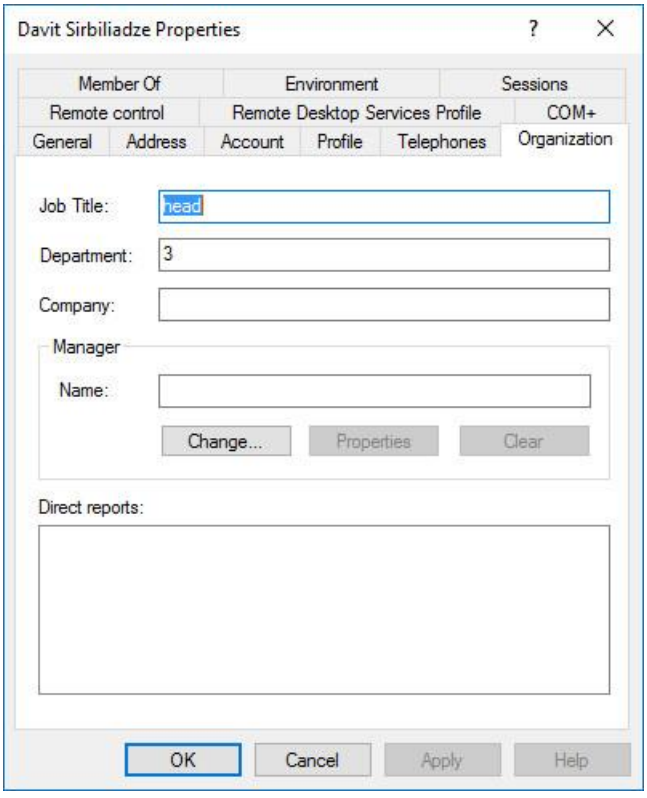
- State/province- შტატი/პროვინცია,
- Zip/Postal Code - საფოსტო ინდექსი,
- Country/region - ქვეყანა/რეგიონი.



სურათი 12. Address ჩანართი AD-ში.

შემდეგი ძირითადი ფანჯარა არის Organization, სადაც იწერება ორგანიზაციის ინფორმაცია:

- Job Title - პოზიციის დასახელება,
- Department - დეპარტამენტი,
- Company - კომპანია,
- Manager Name - მენეჯერის სახელი,
- Direct reports - მოვალეობები.



სურათი 13. Organization ჩანართი AD-ში.

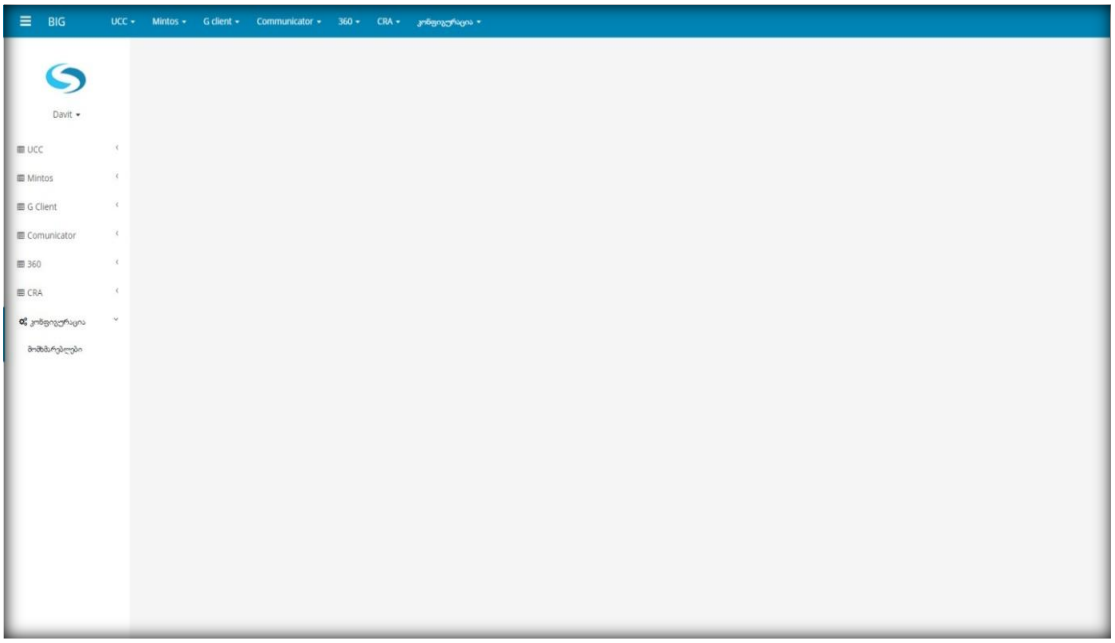
თავი 2.2. ინტრანეტი

AD-სთან ინტეგრირებით და მასზე პროგრამული მანიპულაციით შევიმუშავე უნივერსალური აპლიკაციის შექმნის მიდგომა - ინტრანეტი. ინტრანეტი კი აქვს რამდენიმე განვითარებულ კომპანიას კომპანიას, მაგრამ ისინი ძირითადად ყიდულობენ მზა პროდუქტებს, რომლებიც საკუთარ ძრავზე მუშაობენ - ეს კი პრობლემის გადაჭრის იოლი გზაა, მაგრამ ძალიან ძვირი.

ზოგს კი აქვს ასევე საკუთარი ინტრანეტიც, მაგრამ მასზე მუშაობას საკუთარი რესურსით ძალიან დიდი დრო უნდა. AD ბიბლიოთეკის მეშვეობით, პატარა სკრიპტის დაწერით და ძლიერი პროგრამისტით ეს პრობლემა მარტივად წყდება. ინტეგრირდება Windows-ის სამომხმარებლო ბაზა და ავტორიზაციის სისტემაც მარტივდება და ასევე სრული ინფორმაციის წყაროც ჩვენს ხელთაა.

სურათი 14. ბრაუზერული პროგრამის ავტორიზაცია AD-ს მეშვეობით.

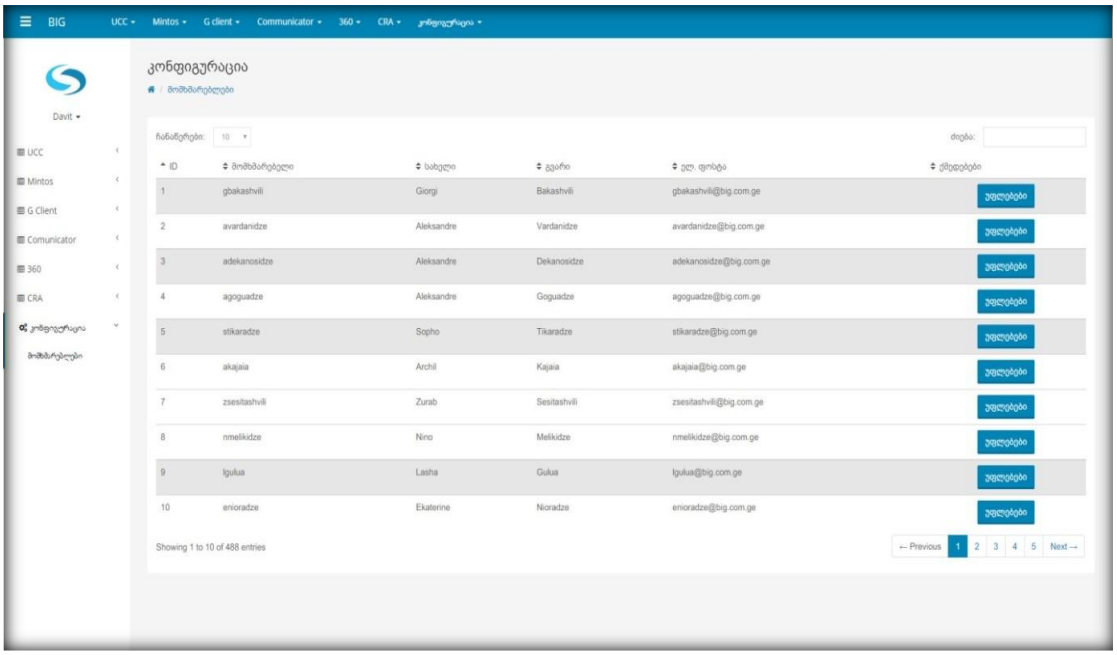
ერთხელ დაწერის შემდეგ კი მისი გამოყენება შეიძლება მრავალ მოდულში. მომხმარებლების უფლებები და ინფორმაცია ხდება წვდომადი პროგრამიდან და მისი დამუშავება და გამოყენება შეიძლება ნებისმიერ ადგილას.



სურათი 15. აპლიკაციის საწყისი გვერდი ავტორიზაციის შემდეგ.

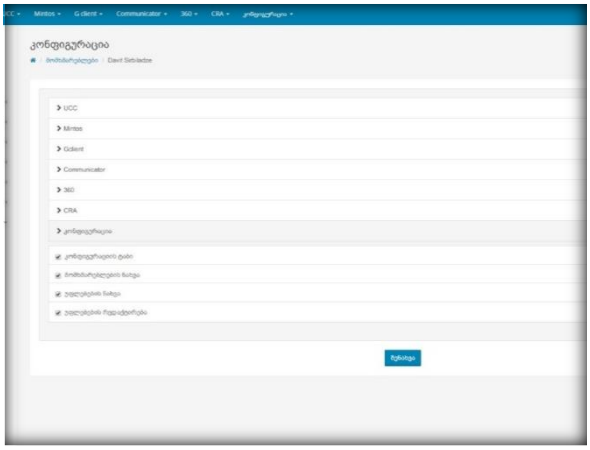
ავტორიზაციაზე იდება წვდომები და ყველა მომხმარებელს თავისი ჯგუფის მიხედვით ენიჭება ხედვები ველებზე და მათი ჩართვის შესაძლებლობა.

მაგალითად, ადმინებს ექნებათ კონფიგურაციის გაკეთების საშუალება, საიდანაც ასინქრონებენ სხვების წვდომებს.



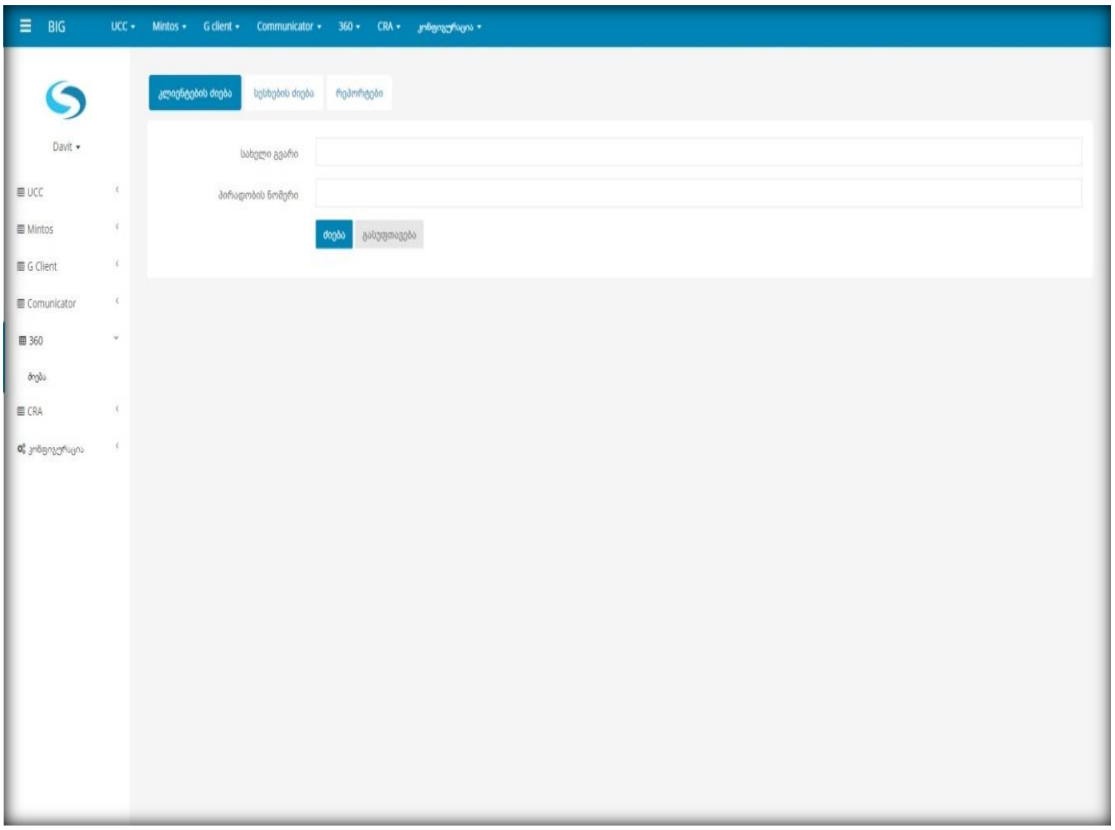
სურათი 16. აპლიკაციის მომხმარებელთა უფლებების გვერდი.

მომხმარებლების სიისთვის გამოვიყენე სტანდარტული DataTable ბიბლიოთეკა, რომელიც დავაკონფიგურირე ისე, რომ მოსახერხებელი იყოს მისი გამოყენება.



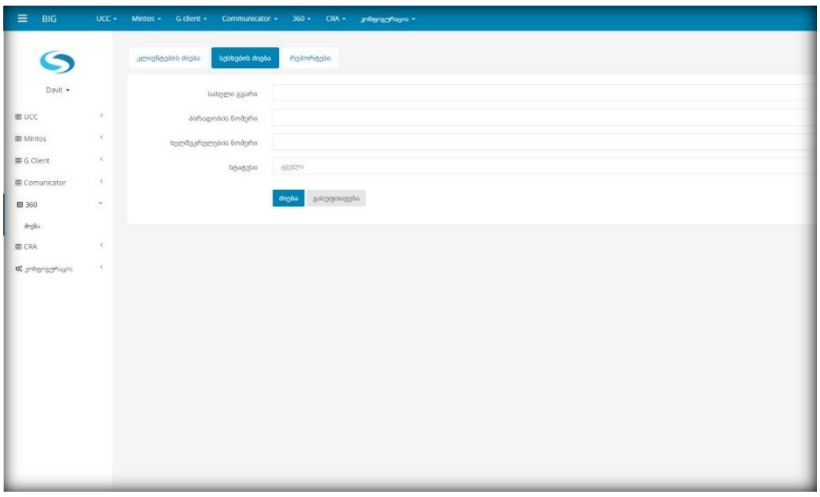
სურათი 17. აპლიკაციის კონფიგურაციის გვერდი.

თითოეული უფლების ჩართვა-გათიშვით კონტროლირდება მომხმარებლების უფლებები პროგრამის ნაწილებზე. მაგალითად, თუ ჩართული ექნებათ კლიენტების ძებნის ფუნქცია, მაშინ დაინახავენ მას, ხოლო თუ გავუთიშავთ შესაბამის უფლებას, მაშინ წვდომა აღარ ექნებათ მასზე.



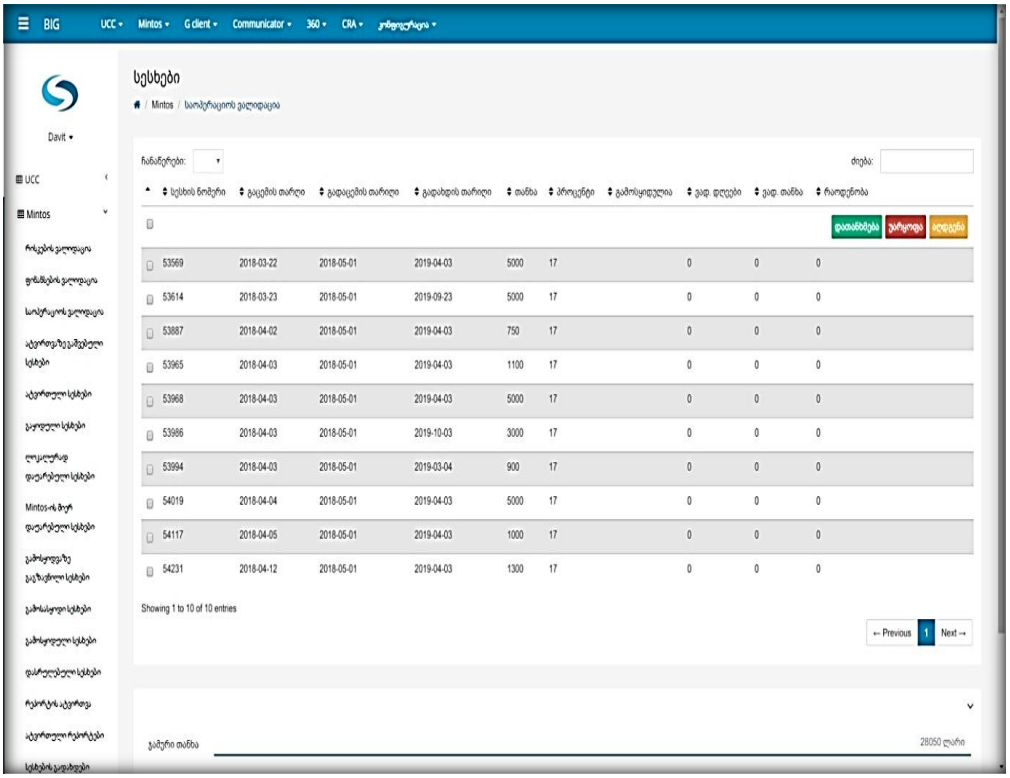
სურათი 18. აპლიკაციის ერთ-ერთი ჩვეულებრივი გვერდი.

ასე მაქვს ყველა ტაბზე უფლებები ჩაშლილი და გაწერილი, რომ მარტივად შეეძლოს ადმინისტრატორს როგორც მათი ცალ-ცალკე მართვა, ასევე გლობალურად.



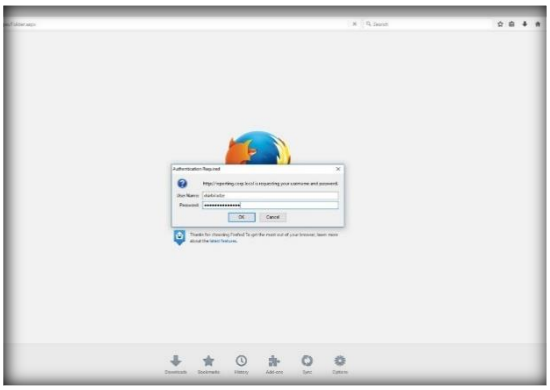
სურათი 19. სადემონსტაციო გვერდი.

უფლებების ჩაშლა ასევე ხდება პარამეტრების მიხედვით. ზოგს შეგვიძლია მხოლოდ ნახვის რეჟიმი ჩავურთოთ, ზოგს რედაქტირების, ზოგსაც მოდერირების თითოეული გვერდის ნაწილის ჭრილში.



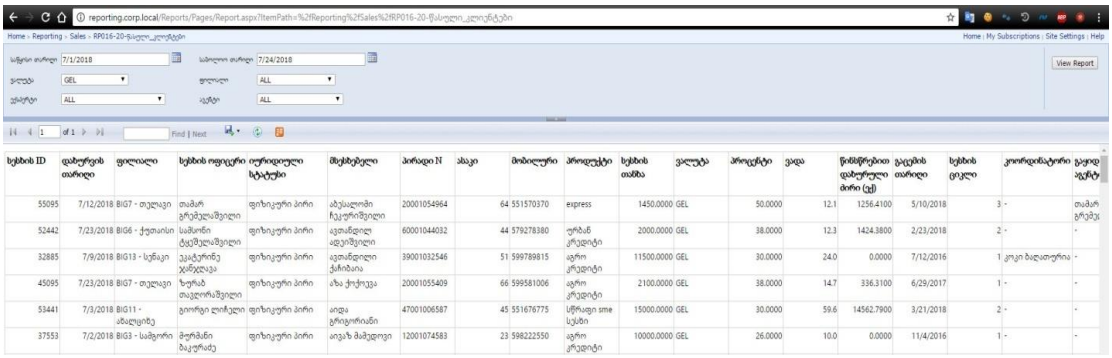
სურათი 20. გვერდი დაკონფიგებული უფლებებით.

რეპორტინგისთვის როდესაც ვიყენებთ Microsoft-ის პროდუქტს, მანდაც შესაძლებელი ხდება მარტივად ინტეგრაცია AD-ს სამომხმარებლო ბაზასთან და მისი აუთენთიფიკაციის გამოყენება.



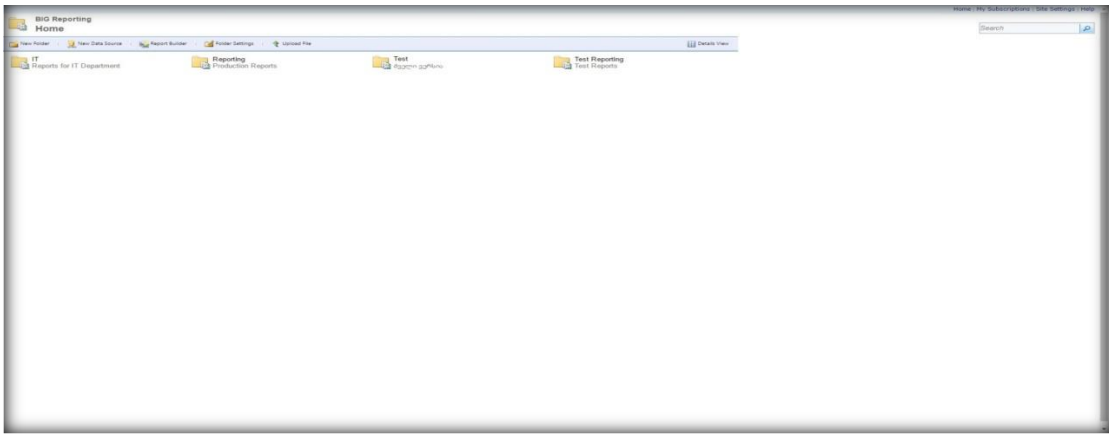
სურათი 21. რეპორტინგ სერვერის ავტორიზაცია.

დომეინური სახელისა და პაროლის შემდეგ იძლევა საშუალება სერვერის რეპორტების ნახვისა, რაც მოიცავს დიდ უსაფრთხოებას და საიმედოობას, ისევე როგორც დიდი დროის დაზოგვას. [9]



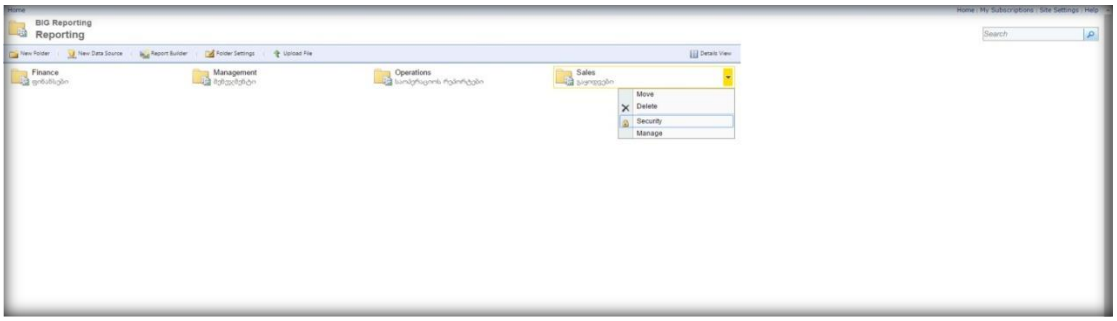
სურათი 22. რეპორტინგ სერვერის რეპორტი.

რეპორტინგ სერვერის საქაღალდეებზეც მოქნედებს იგივე უფლებები. წვდომების ჩართვა შეიძლება როგორც კონკრეტულ პირზე, ასევე ჯგუფზე.



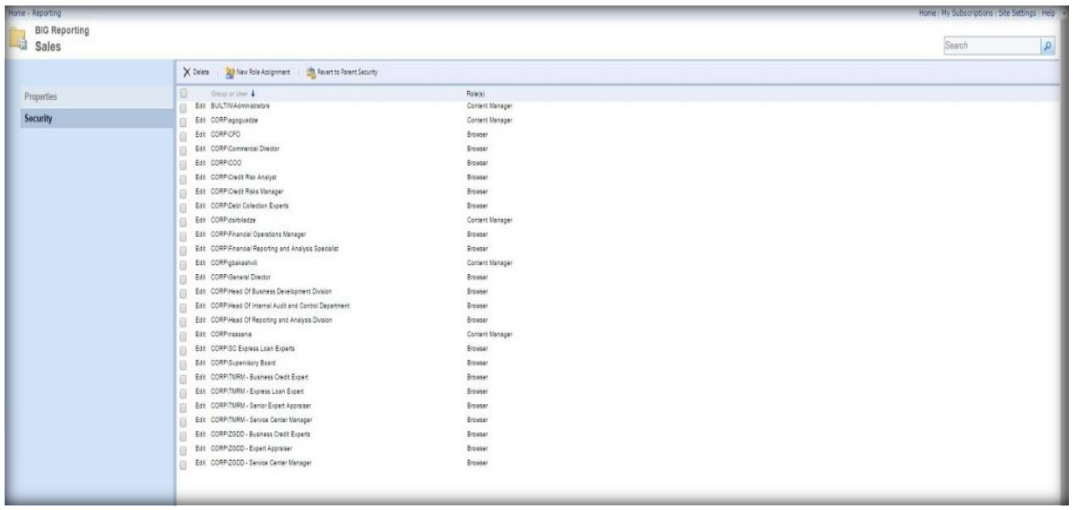
სურათი 23. რეპორტინგ სერვერის მთავარი გვერდი.

ჯგუფური უფლების გაწერა უფრო მიზანშეწონილია, რადგან მაგალითად თუ გვინდა, რომ რუსთავის სერვის ცენტრის მენეჯერს ჰქონდეს რაღაც რეპორტის გენერირების უფლება და 1 წელში იცვლის პოზიციას ან მიდის სამსახურიდან, მაშინ მისი წვდომა აირევა და იქნება თავიდან გასაწერი, ხოლო თუ უფლებას ჩავურთავთ მისი პოზიციის ჯგუფს, მაშინ მის შემცვლელს უკვე ექნება იგივე უფლებები, რაც ჰქონდა ძველ მენეჯერს, ყოველგვარი ხელახალი კონფიგურაციის გარეშე.



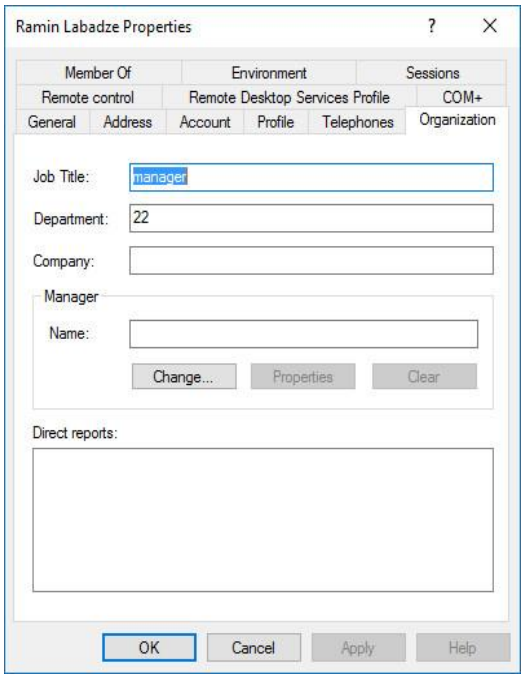
სურათი 24. რეპორტინგ სერვერის საქღალდე.

რეპორტინგ სერვერზე უფლებები შეიძლება ჩაერთოთ როგორც კონკრეტულ რეპორტზე, ასევე საქაღალდეზე. საქაღალდეზე წვდომის ჩართვით შეგვიძლია მის ყველა რეპორტზე მივცეთ დაშვება, ან შიგნითაც მანუალურად ვაკონტროლოთ და ზოგი ვაჩვენოთ, ზოგი კი - დავუმალოთ.

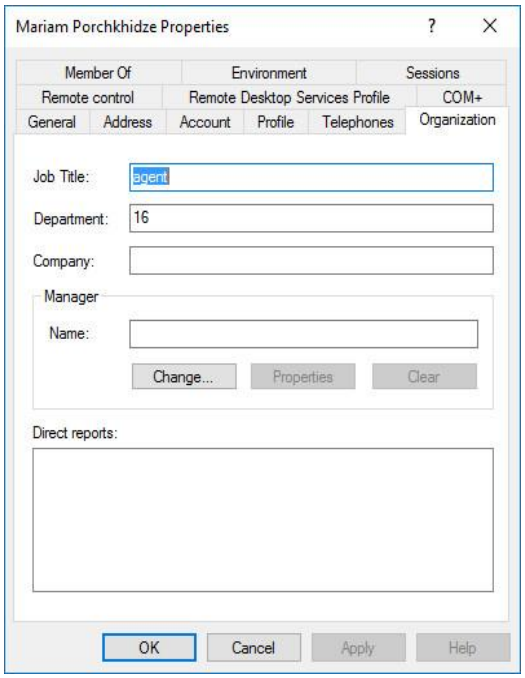


სურათი 25. რეპორტინგ სერვერის უფლებები.

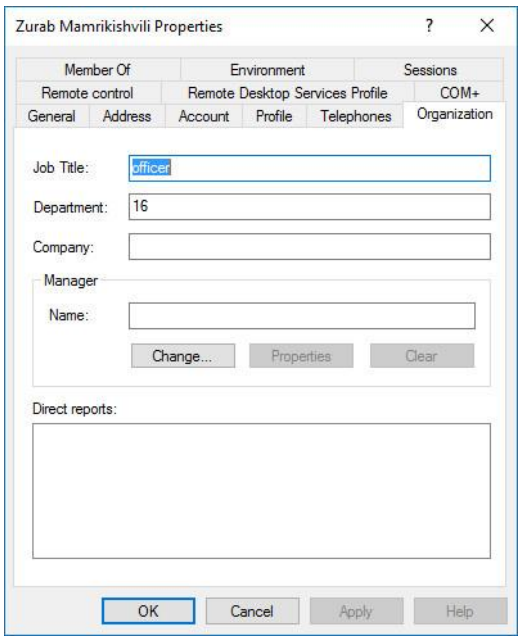
რეპორტინგ სერვერს შეუძლია დანახვა AD-ს თითოეული მოდულის, რომელიც ზევით აღვწერეთ. AD ხომ ჩვეულებრივი ბაზაა, რომელზეც შესაძლებელია სკრიპტი დაიწეროს და წამოვიღოთ ნებისმიერი ვარიაციის ინფორმაცია, რასაც შემდგომ SQL-ში გავარჩევთ ერთმანეთისგან და დავადებთ შესაბამის პირობას და ლოგიკას იერარქიის მიხედვით.



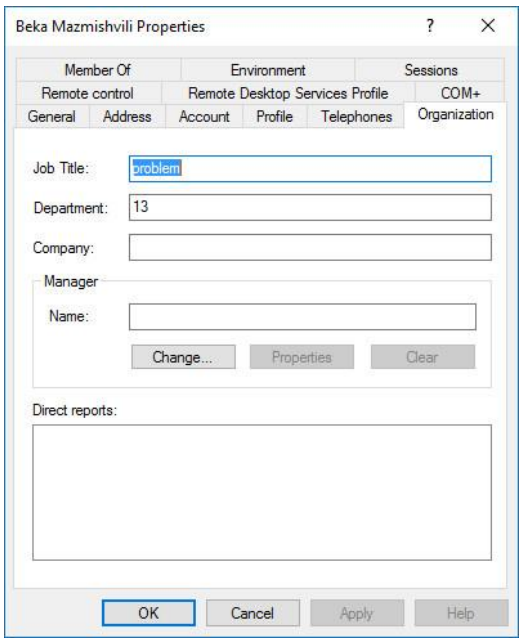
სურათი 26. უფლების ვარიანტი 1.



სურათი 27. უფლების ვარიანტი 2.



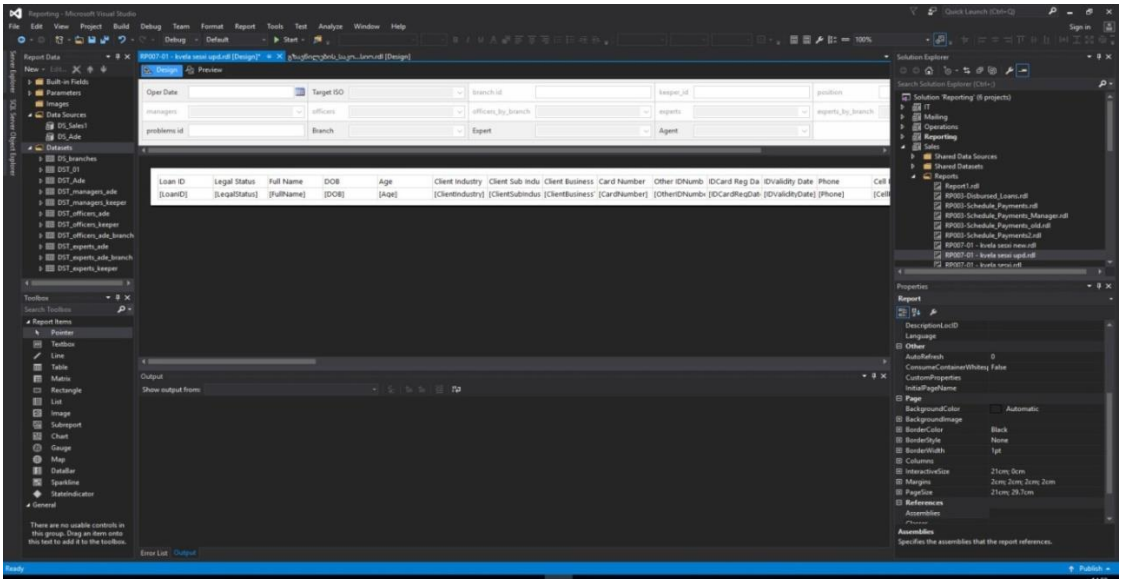
სურათი 28. უფლების ვარიანტი 3.



სურათი 29. უფლების ვარიანტი 4.

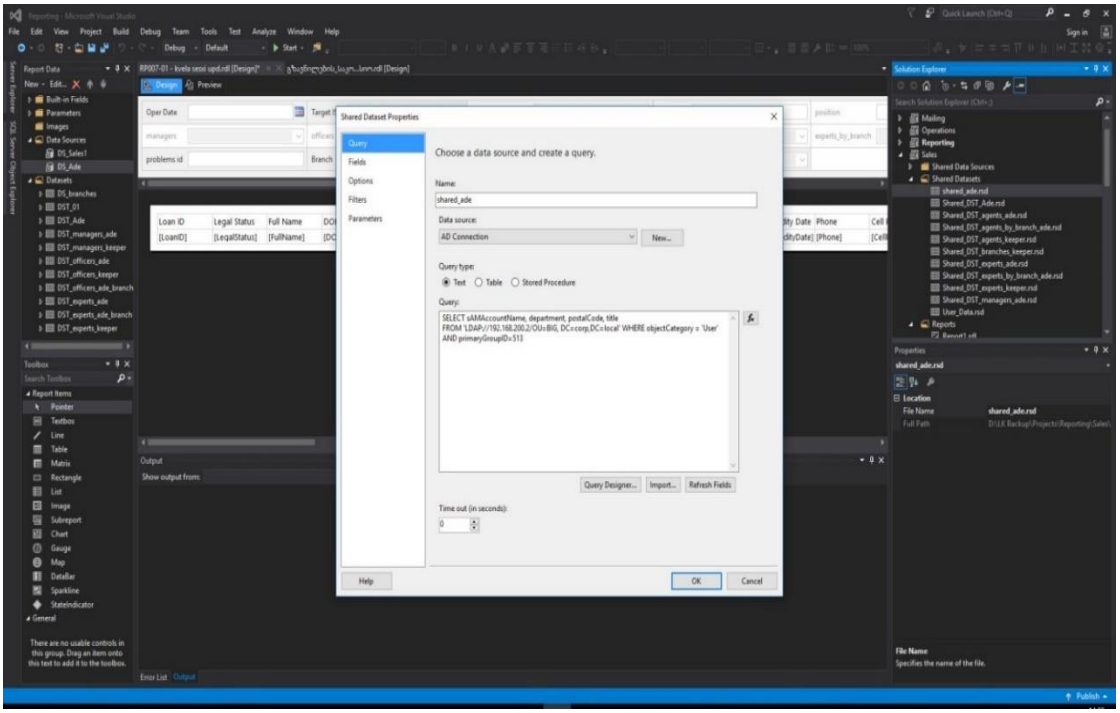
რ

ეპორტის აწყობისთვის საჭიროა Visual Studio და რეპორტინგის სერვისი, რომელიც ინტეგრირდება მასში.



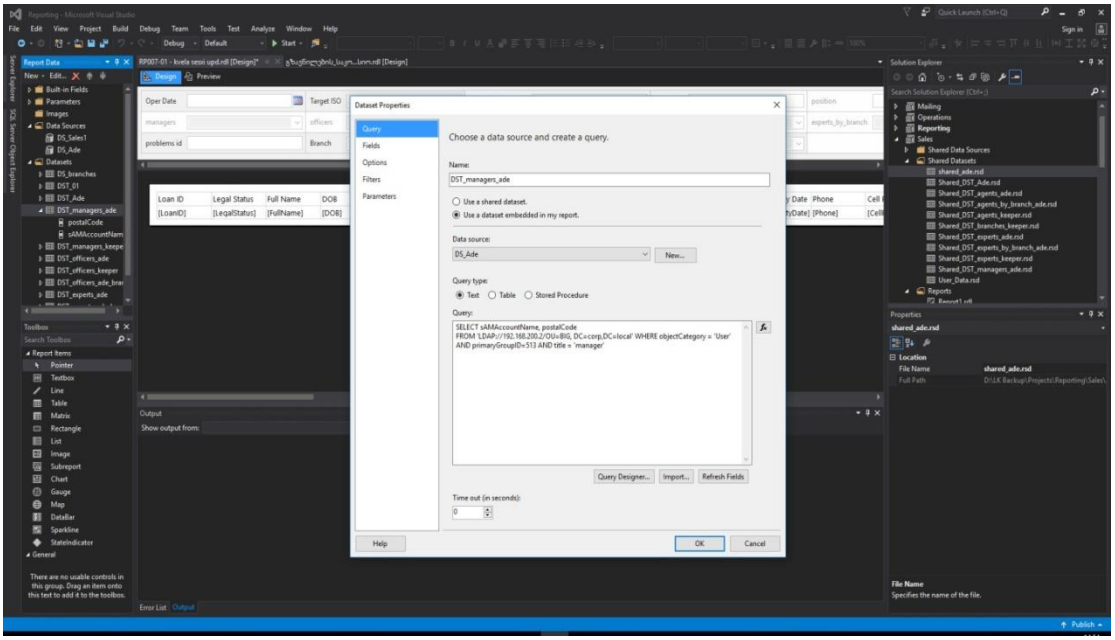
სურათი 30. Microsoft Visual Studio, რეპორტის აწყობა.

რეპორტები ეწყობა ვიზუალურად, ხოლო მათი სკრიპტისთვის გამოყოფილია სპეციალური სივრცე.

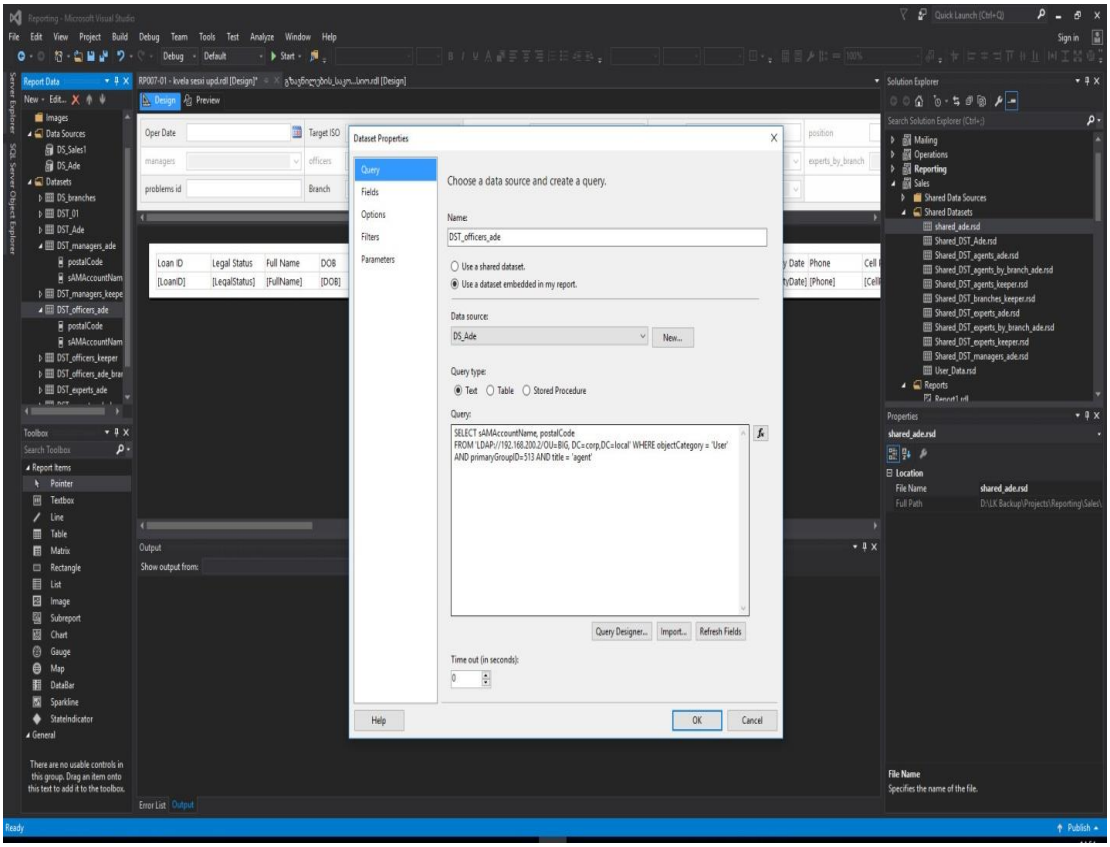


სურათი 31. Microsoft Visual Studio, კავშირი AD-სთან, სკრიპტი 1.

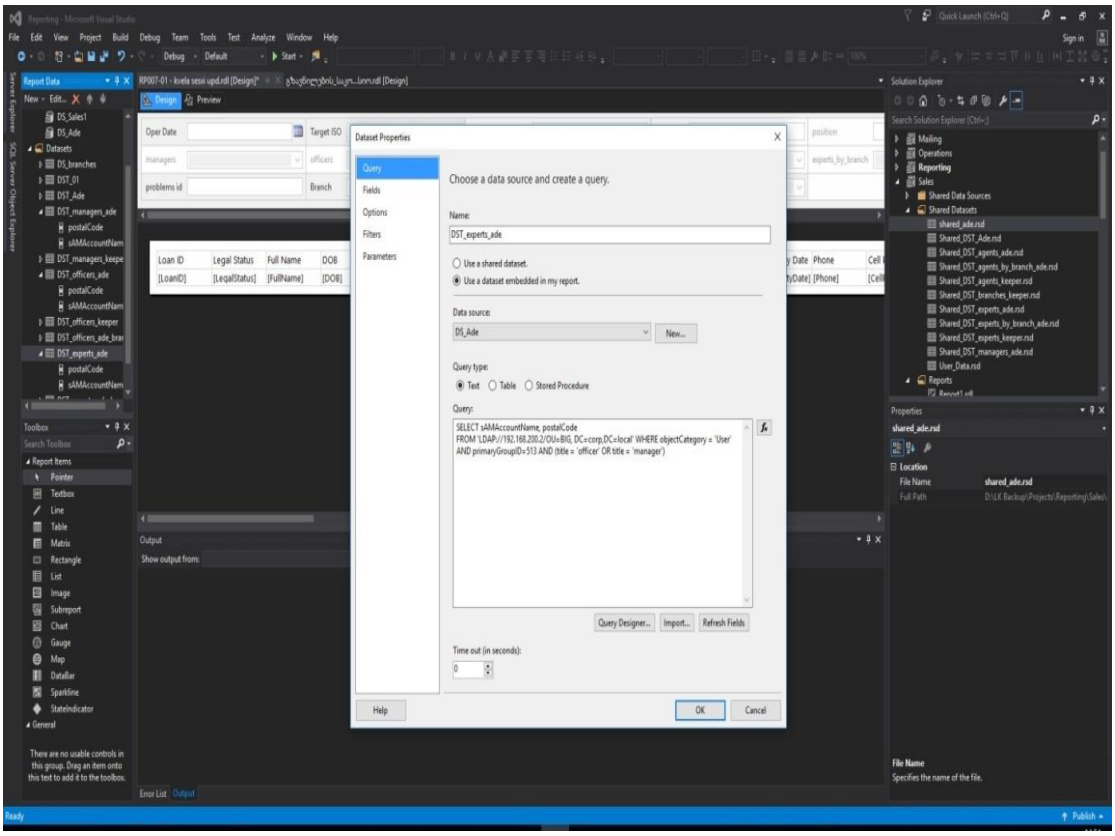
ყველა კატეგორიისთვის იწერება ცალკე სკრიპტი და ბოლოს რეპორტში ჯამდება და ამოდის საერთო დიდი ლოგიკის მქონე რეპორტი.



სურათი 32. Microsoft Visual Studio, კავშირი AD-სთან, სკრიპტი 2.



სურათი 33. Microsoft Visual Studio, კავშირი AD-სთან, სკრიპტი 3.

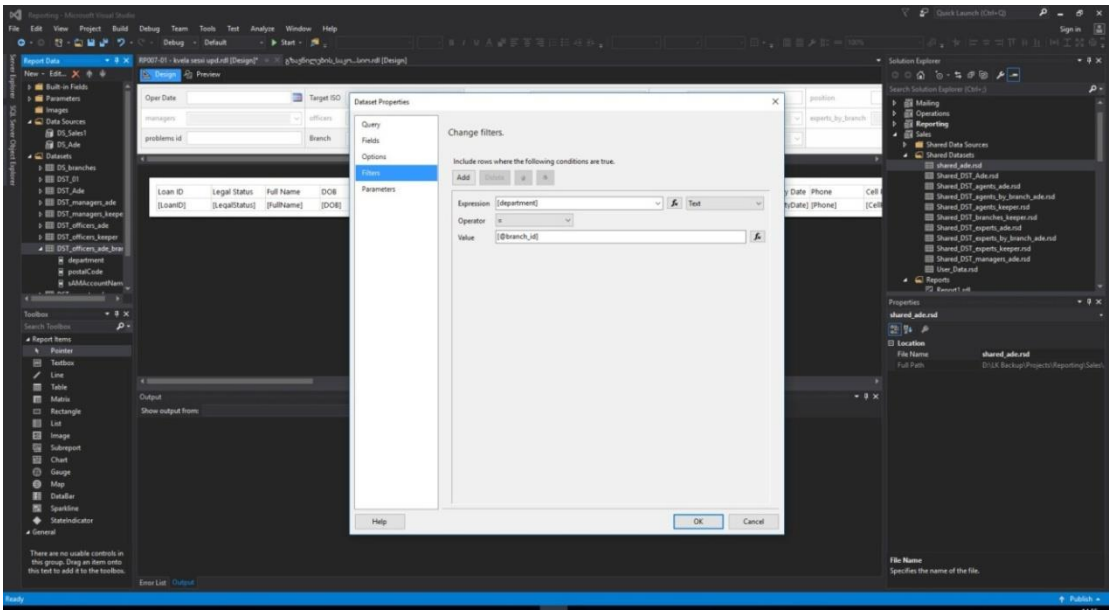


სურათი 34. Microsoft Visual Studio, კავშირი AD-სთან, სკრიპტი 4.

ყველა სკრიპტს ერქმევა უნიკალური დასახელება. მათი დაწერა შეიძლება როგორც ლოკალურად, ასევე გლობალურად, რომ პაკეტში მყოფმა სხვა რეპორტებმაც მიმართონ.

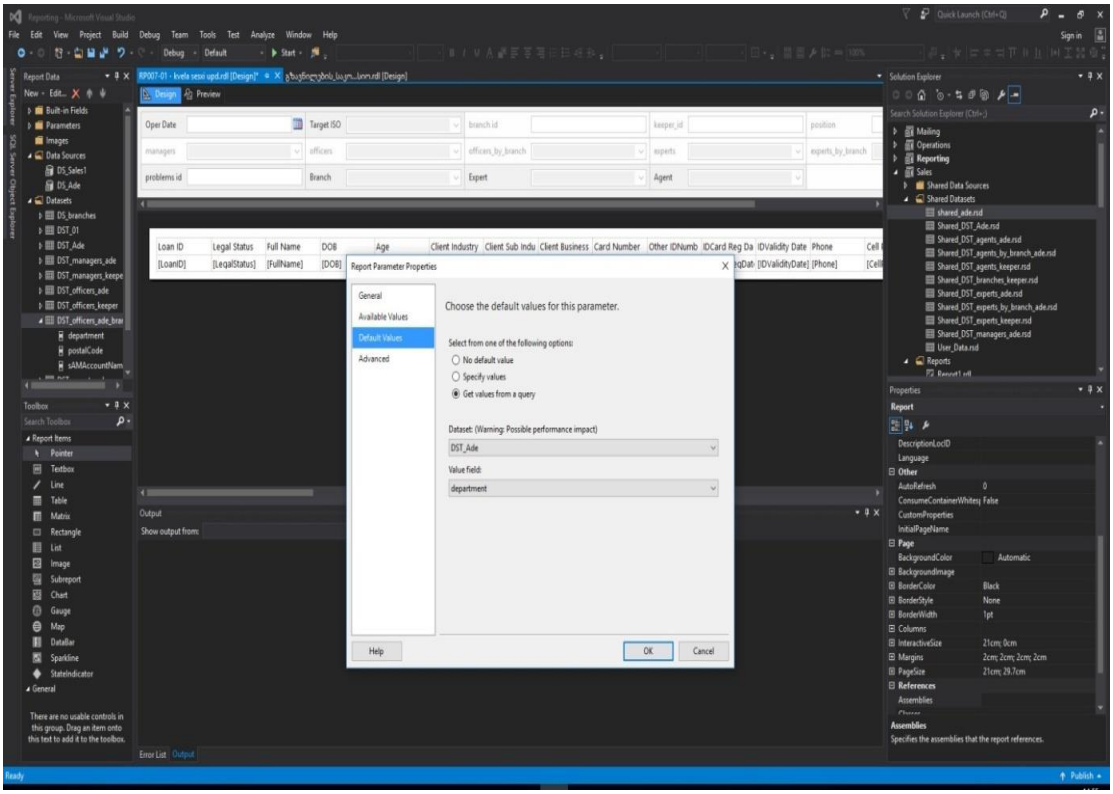
Visual Studio-ს ყოველი ახალი ვერსიის გამოსვლით უმჯობესდება რეპორტინგ სერვერის ინტერფეისი, ფუნქციონალი, ვიზუალი, ემატება ავტომატური რეპორტების გენერირების საშუალებები და იხვეწება გრაფიკები, ემატება სხვადასხვა სისტემებთან ინტეგრირების მხარდაჭერები.

წამოღებული ინფორმაციის გარჩევისთვის და რეპორტში გამოყენებისთვის არის ჩანართი დაპარამეტრებისთვის, საიდანაც SQL-ს გადაეცემა შევსებული ცვლადის მნიშვნელობა.



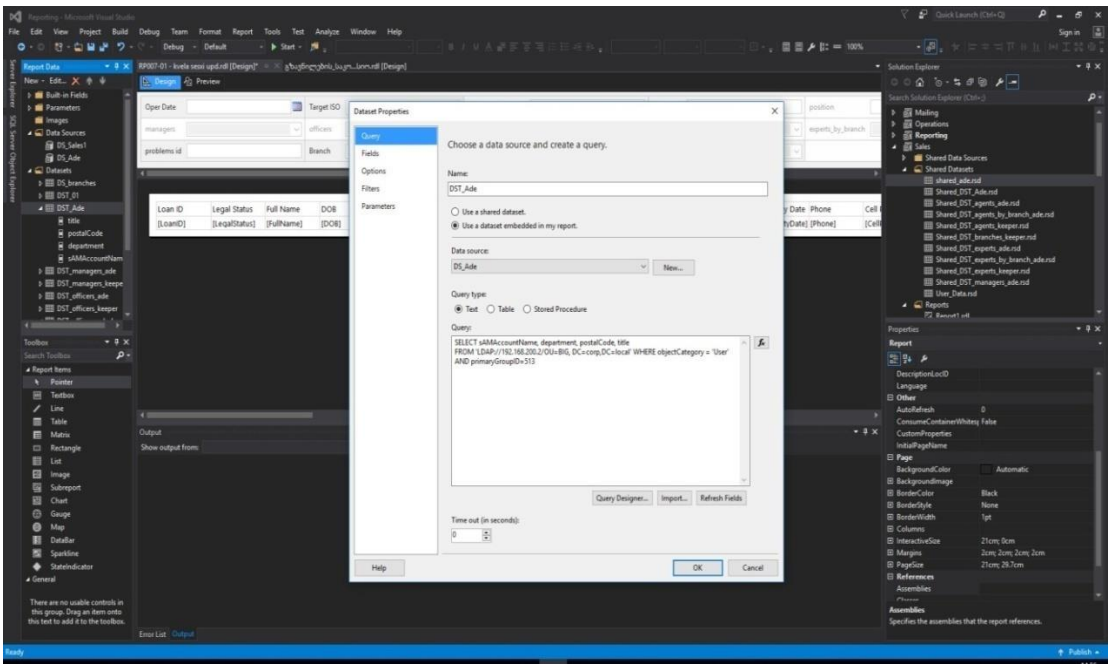
სურათი 35. Microsoft Visual Studio, პარამეტრების გაწერა.

მნიშვნელობა შეგვიძლია წამოვიღოთ როგორც ავტომატურად სერვერიდან, ასევე გადავცეთ სტატიკურად ხელით.



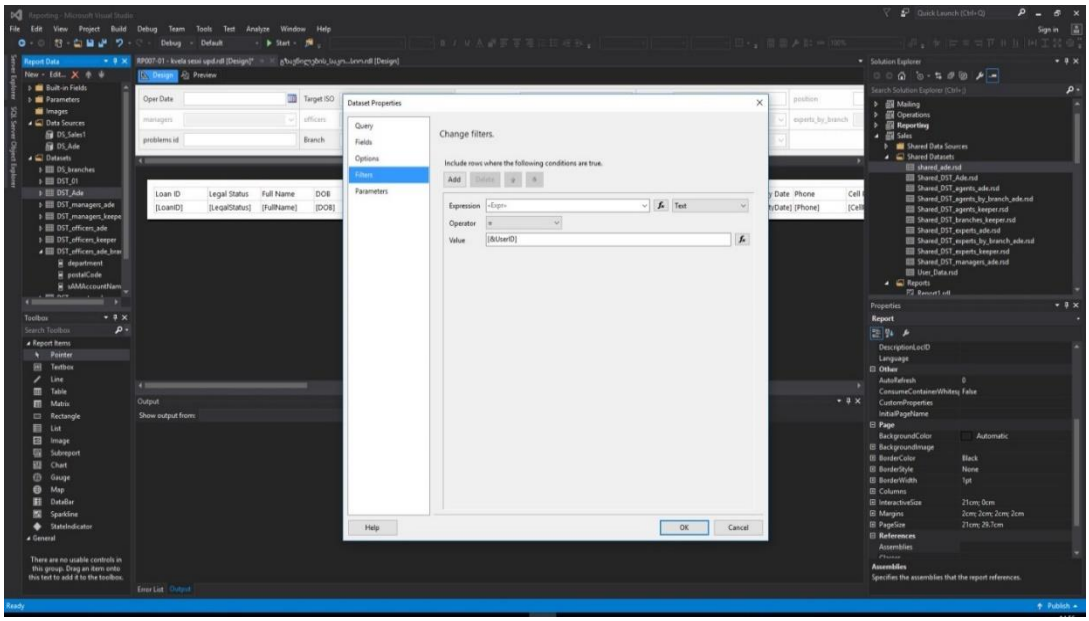
სურათი 36. Microsoft Visual Studio, Default პარამეტრები.

AD-დან შეგვიძლია როგორც სრული მონაცემები წამოვიღოთ, ასევე გაფილტრული.



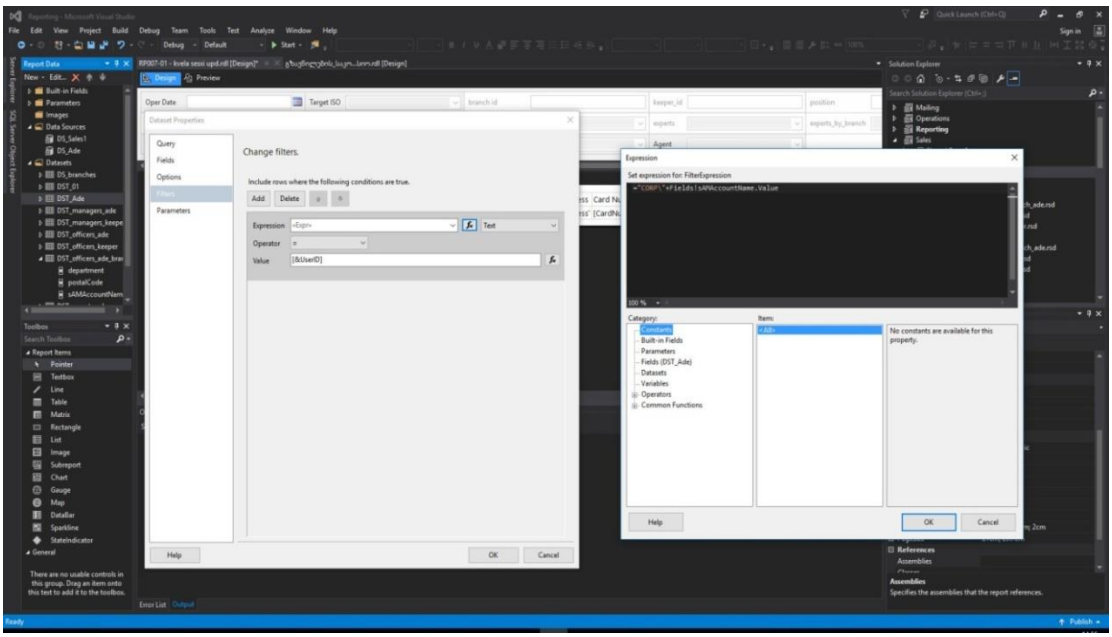
სურათი 37. Microsoft Visual Studio, AD სკრიპტი ფილტრაციით.

პროგრამას ჩაშენებული აქვს შემოსული მომხმარებლის ამომცნობი ფუნქცია, რომლის გამოძახებისთვის საჭიროა მანუალური პარამეტრის გამოძახება და მინიჭება.



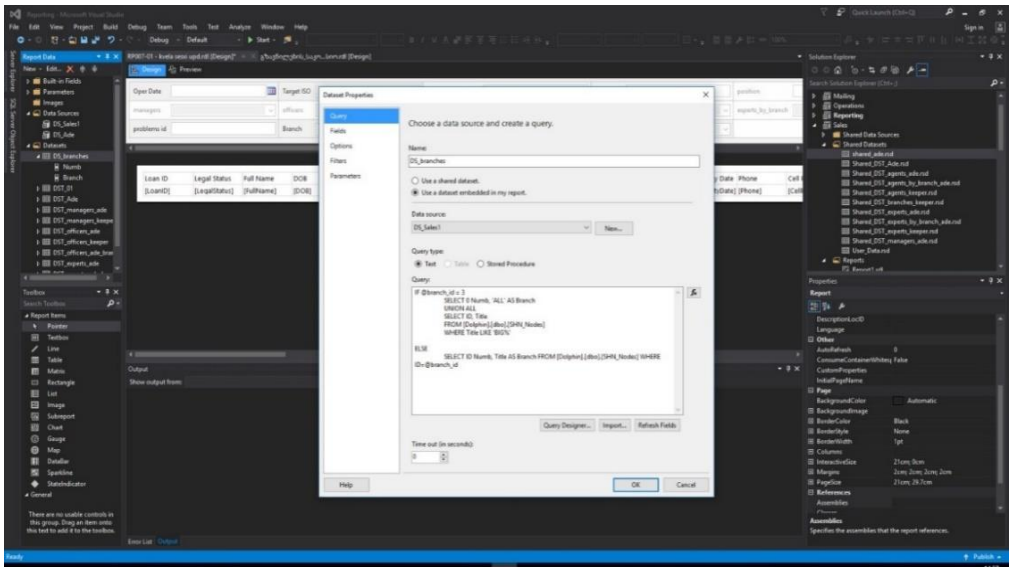
სურათი 38. მომხმარებლის ამოცნობა.

ასევე შესაძლებელია შემოსული მომხმარებლის დადარება SQL-ის ბაზიდან წამოსულ მონაცემებთან და გამიჯვნა ერთმანეთისგან.

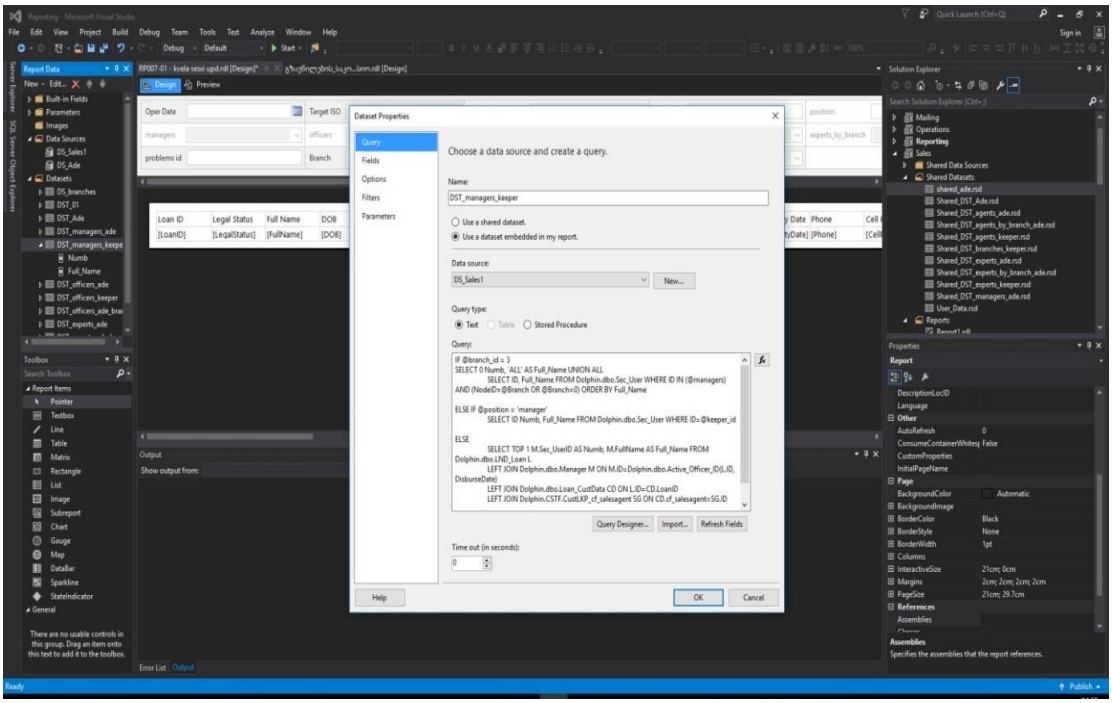


სურათი 39. მომხმარებლის ამოცნობა დეტალურად.

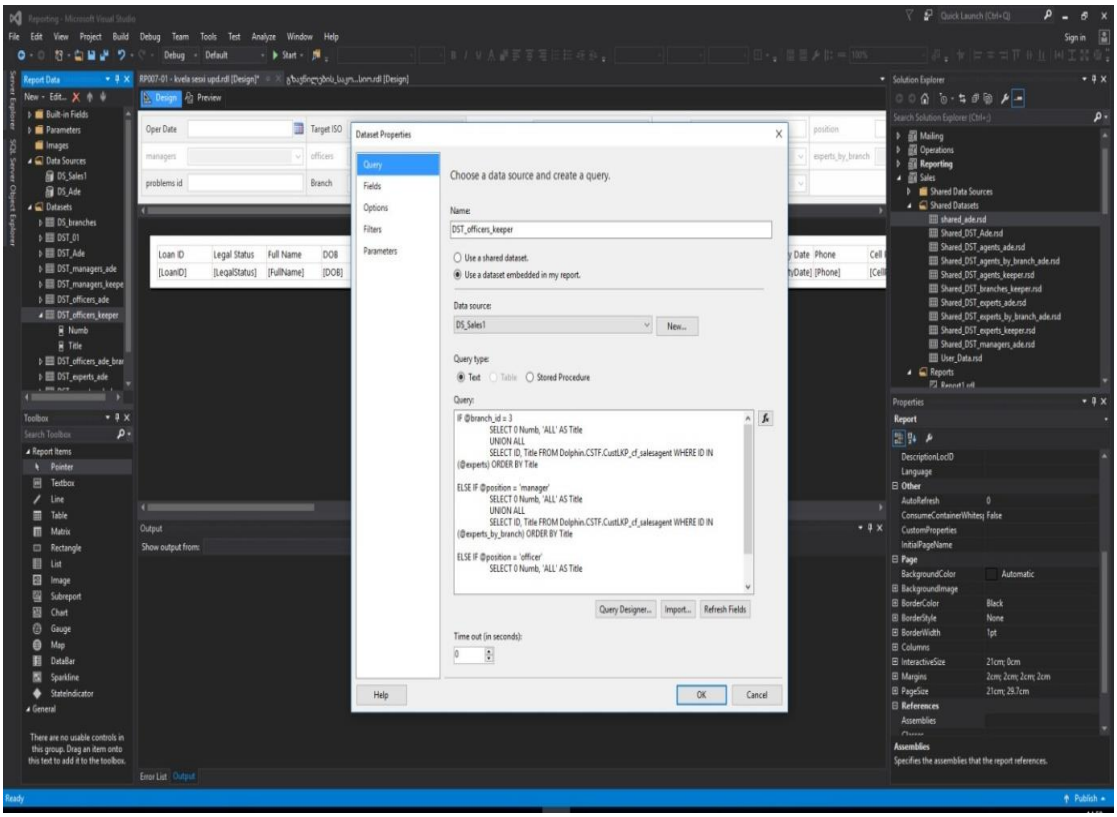
ამ მიდგომით შევძლებთ დინამიური რეპორტების შექმნას. თუ გვინდა რეპორტი, რომელიც სხვადასხვა ფილიალისთვის და პოზიციისთვის უნდა აგენერირებდეს სხვადასხვა ლოგიკით ერთი და იგივე მონაცემებს, მაგის გასაკეთებლად არაა ამ ყველა ვარიაციის ცალ-ცალკე რეპორტის შექმნა.



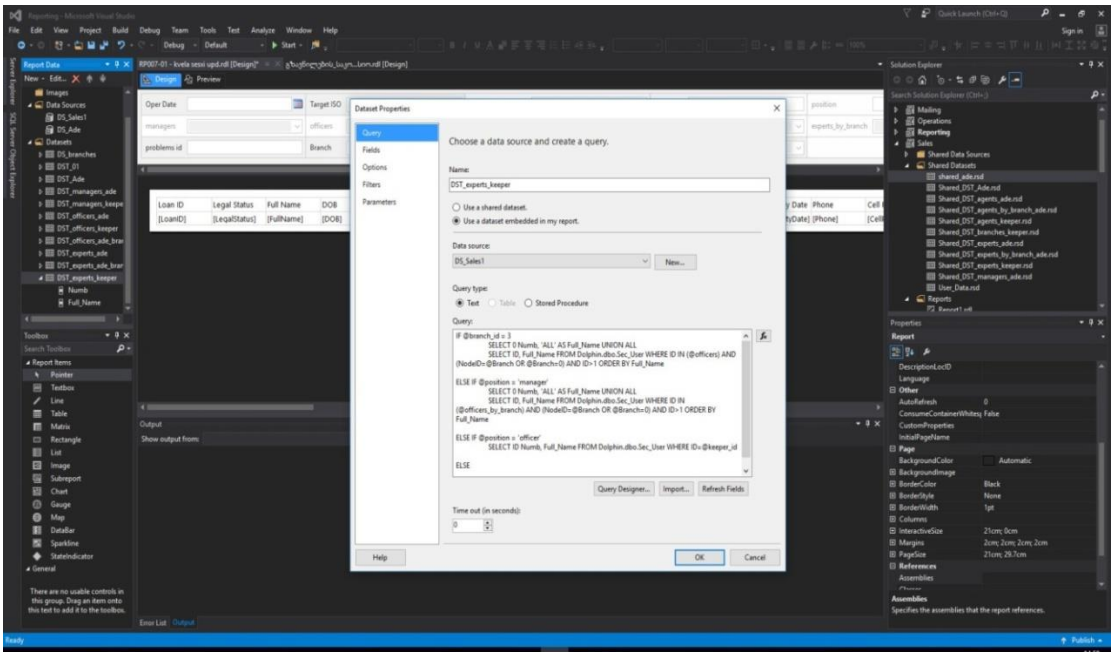
სურათი 40. პირობები პარამეტრებზე v1.



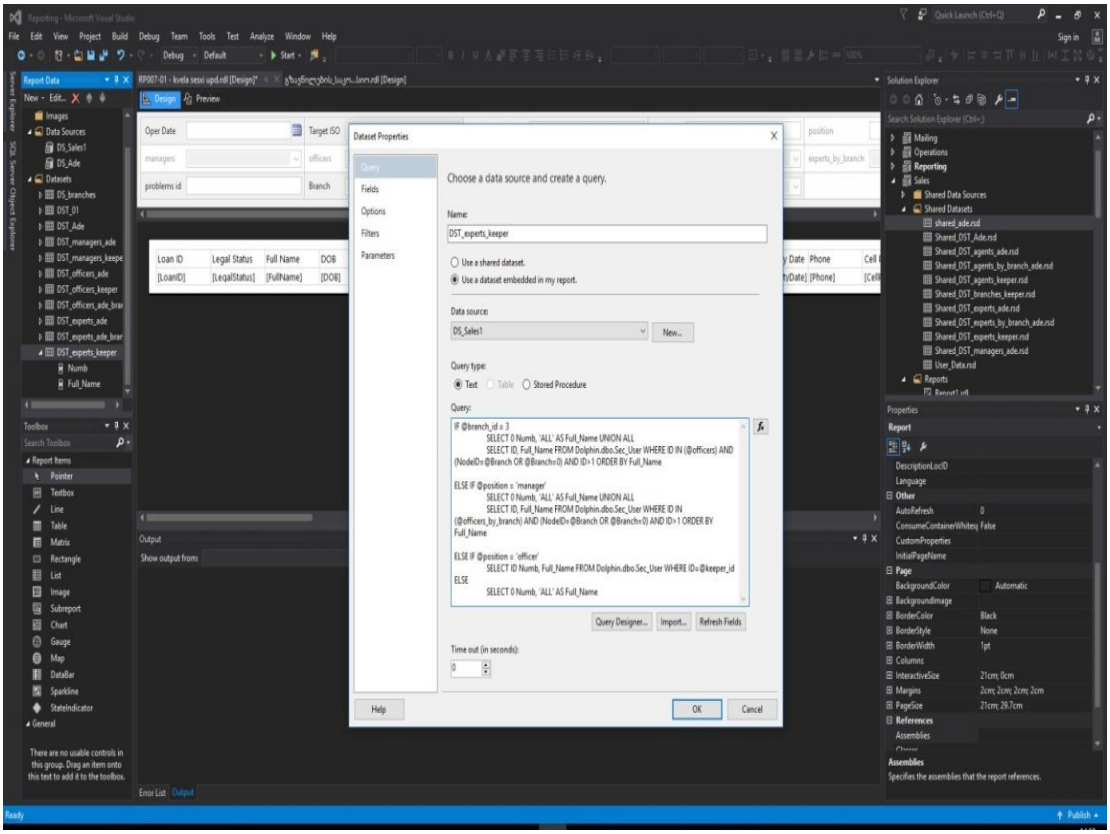
სურათი 41. პირობები პარამეტრებზე v2.



სურათი 42. პირობები პარამეტრებზე v3.

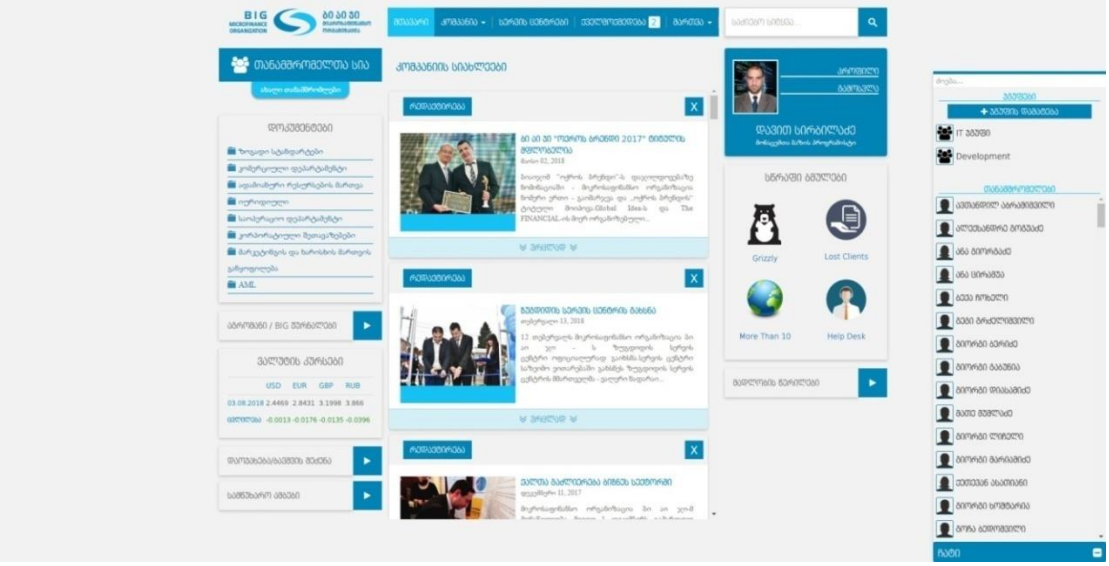


სურათი 43. პირობები პარამეტრებზე v4.



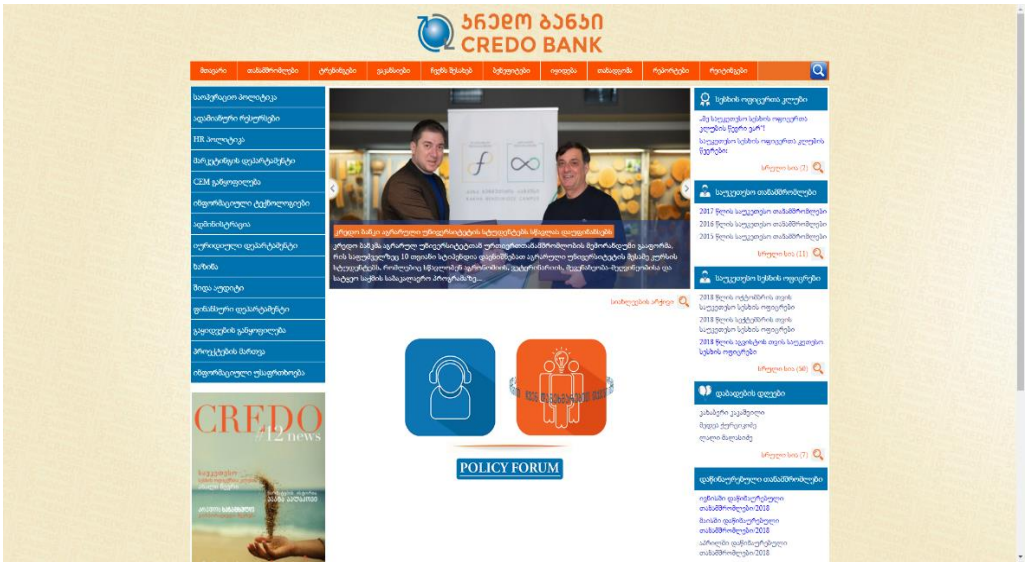
სურათი 44. პირობები პარამეტრებზე v5.

ინტრანეტი კომპანიას სჭირდება იმისთვის, რომ მისმა თანამშრომლება შეძლონ ერთმანეთთან კომუნიკაციის დამყარება და ფაილების გაცვლა გარე სერვისებისგან დამოუკიდებლად.



სურათი 45. ინტრანეტი v1.

ინტრანეტებისთვის შესაძლებელია მრავალი ინტერფეისის მიცემა და ორგანიზება, გაფორმება და გასტილვა ორგანიზაციის საქმიანობიდან გამომდინარე და მასზე დაყრდნობით მრავალი ფუნქციონალის მიმატება. იგი არის ჩვეულებრივი პროექტი, რომელიც განვითარებას საჭიროებს.



სურათი 46. ინტრანეტი v2.

იმ შემთხვევაში, თუ დაწესებულების პოლიტიკა არ იძლევა AD-ზე დაშვების უფლებას, შესაძლებელია ინტეგრირება და ავტორიზაციის სხვა სისტემებთანაც, ან პირადი სისტემის შექმნა, რომელიც აგრეთვე შესაძლებელია გახდეს ზოგადი მოდული სხვადასხვა სისტემებისთვის, სადაც იქნება პირადი ბაზა და შიფრაცია.

ინტეგრაციის დროს უნდა შედგეს Job-ი, რომელიც ავტომატურად გაეშვება დროის რაღაც პერიოდში, მაგალითად ყოველ ღამე და წამოიღებს თანამშრომლებს და ჩაწერს ბუფერში. მაგალითად HrAdmin-თან ინტეგრაცია გამოიყურება შემდეგნაირად:

```
DECLARE @Day DATETIME = GETDATE()
SELECT DISTINCT p.IDNo AS pid ,
        CredoBnk.dbo.Trim(p.FirstNameEn) AS FirstName ,
        CredoBnk.dbo.Trim(p.LastNameEn) AS LastName ,
        perstat.PersonStatusName AS PersonStatus ,
        LOWER(ISNULL(perstat.absc, '')) AS absc ,
        ISNULL(CredoBnk.dbo.Trim(rt.FirstNameEn), '') + ' ' +
ISNULL(CredoBnk.dbo.Trim(rt.LastNameEn), '') AS ReportsTo ,
        LOWER(ISNULL(CASE WHEN rp.InternalMail = 'Nina_Akhaladze@credo.ge' THEN
'Nino_Akhaladze@credo.ge' ELSE rp.InternalMail END, '')) AS ReportsToMail ,
        CAST(IIF(iwha.GradeName = 'N/A', '2', iwha.GradeName) AS NUMERIC(17,
2)) AS GradeNameSort ,
        LOWER(CASE WHEN ped.InternalMail = 'Nina_Akhaladze@credo.ge' THEN
'Nino_Akhaladze@credo.ge' ELSE ped.InternalMail END) AS mail ,
        ped.EmployeeCode ,
              IIF(ISNULL(TRY_CAST(u.ShortName AS INT), 0) = 0, 98,
TRY_CAST(u.ShortName AS INT)) AS BranchId ,
        iwha.JobName_S AS JobName ,
        CredoBnk.dbo.Trim(iwha.UnitTitle_S) AS UnitTitle ,
        iwha.LocationTitle AS Location ,
        ISNULL(( SELECT TOP 1
                        CredoBnk.dbo.Trim(p2.PhoneNo)
                 FROM   HR.HRP_SMART.Per.Phones AS p2
                 WHERE  p2.PersonID = p.ID
                        AND p2.PhoneTypeID = 1
                        AND p2.OpenDate <= @Day
                        AND ( p2.CloseDate >= @Day
                              OR p2.CloseDate IS NULL
```

```
                        )
                ), '') AS CorporateNumber ,
        ISNULL(( SELECT TOP 1
                        p2.PhoneNo
                 FROM   HR.HRP_SMART.Per.Phones AS p2
                 WHERE  p2.PersonID = p.ID
                        AND p2.PhoneTypeID = 2
                        AND p2.OpenDate <= @Day
                        AND ( p2.CloseDate >= @Day
                              OR p2.CloseDate IS NULL
                            )
                ), '') AS PrivateNumber ,
        ISNULL(( SELECT TOP 1
                        p2.PhoneNo
                 FROM   HR.HRP_SMART.Per.Phones AS p2
                 WHERE  p2.PersonID = p.ID
                        AND p2.PhoneTypeID = 3
                        AND p2.OpenDate <= @Day
                        AND ( p2.CloseDate >= @Day
                              OR p2.CloseDate IS NULL
                            )
                ), '') AS PhoneNumber ,
        ISNULL(pp.Name, '') AS PositionCategory
FROM    HR.HRP_SMART.Per.Persons AS p
        INNER JOIN HR.HRP_SMART.Per.Contracts AS c ON c.PersonID = p.ID
                                                      AND @Day >= c.StartDate
                                                      AND ( @Day <= c.EndDate
                                                            OR c.EndDate IS
NULL
                                                          )
        INNER JOIN HR.HRP_SMART.Per.vw_PersonEmpDetail_active AS ped ON
ped.PersonID = p.ID
        INNER JOIN HR.HRP_SMART.Per.vw_InternalWorkHistory_Active AS iwha ON
iwha.PersonID = p.ID
        INNER JOIN HR.HRP_SMART.Core.Units AS u ON u.ID = iwha.UnitID
        LEFT JOIN HR.HRP_SMART.Core.PositionCategory AS pp ON pp.ID =
iwha.PositionCategoryID
        LEFT JOIN HR.HRP_SMART.Core.vw_Positions_ReportsTo_Active AS RPTA ON
iwha.PositionID = RPTA.PositionID
        LEFT JOIN HR.HRP_SMART.Per.vw_InternalWorkHistory_Active AS IWA2 ON
RPTA.ParentPositionID = IWA2.PositionID
```

```
        LEFT JOIN HR.HRP_SMART.Per.Persons AS rt ON ISNULL(IWA2.PersonID,
iwha.ReportsToPersonID) = rt.ID
        LEFT JOIN HR.HRP_SMART.Per.vw_PersonEmpDetail_active AS rp ON
rp.PersonID = ISNULL(IWA2.PersonID, iwha.ReportsToPersonID)
        LEFT OUTER JOIN ( SELECT    PersonID ,
                                    ( SELECT IIF(MAX(InternalMail) =
'Nina_Akhaladze@credo.ge', 'Nino_Akhaladze@credo.ge', MAX(InternalMail)) FROM
HR.HRP_SMART.Per.vw_PersonEmpDetail_Active WHERE PersonID = ( SELECT
MAX(SubstitutePersonID) FROM HR.HRP_SMART.Per.Absences_View AS absc WHERE
absc.PersonID = Get_Persons_Statuses_1.PersonID AND @Day BETWEEN absc.StartDate
AND absc.EndDate ) ) AS absc ,
                                    PersonStatusID ,
                                    PersonStatusName ,
                                    StatusID ,
                                    abid ,
                                    bizid ,
                                    matid
                          FROM
Credo.dbo.Get_Persons_Statuses(Credo.dbo.TruncDate(@Day))
                                    AS Get_Persons_Statuses_1
                        ) AS perstat ON p.ID = perstat.PersonID
                            ORDER BY GradeNameSort DESC
```

სტანდარტულად AD-დან ავტორიზებისთვის ვიყენებ adLDAP ბიბლიოთეკას, რომელსაც აქვს ღია ძრავი და დაწერილია PHP ენაზე.იგი გვაძლევს საშუალებას, რომ კომპანია პროგრამულად ძალიან მაღალ დონეზე განვავითაროთ. შესაძლებელია ერთი პროგრამის დაწერა, რომელიც მართავს ყველა სტრუქტურას. ამ სისტემით შესაძლებელი იქნება როგორც მომხმარებლების რეგისტრაცია, ასევე რედაქტირება და გაუქმება, სინქრონი ყველა სისტემასთან, გამოყენება, დამუშავება, ანალიზი.

adLDAP-ის მეშვეობით შესაძლებელია როგორც მომხმარებლებისთვის ავტორიზების გაკეთება, აგრეთვე სესიის დახურვა, რაც შესაძლებელია, რომ ჩავიმახსოვროთ და შესაბამისად გვექნება თანამშრომელთა აღრიცხვა - შემოსვლა/გასვლები, რაც გვაძლევს ღრმა ანალიზის გაკეთების საშუალებას იმავე, ერთი უნივერსალური აპლიკაციიდან მარტივად. [8]

AD-ში ძიებისთვის არის შემდეგი სკრიპტი:

```
$ad = new Adldap($configuration);
$results = $ad->search()->all();
```

შეგვიძლია რომელიმე კონკრეტული თანამშრომლის ინფორმაციის ძიება:

```
$results = $ad->search()->where('cn', '=', 'John Doe')->get();
```

ან სახელის რომელიმე კონკრეტული ნაწილის მიხედვით ძიება:

```
$results = $ad->search()->where('cn', 'starts_with', 'John')->get();
$results = $ad->search()->where('cn', 'ends_with', 'Doe')->get();
$results = $ad->search()->where('cn', 'contains', 'John Doe')->get();
```

ასევე რამდენიმე მომხმარებლსი ერთად ძიება:

```
$results = $ad->search()
        ->where('cn', '=', 'John Doe')
        ->orWhere('cn' '=', 'Suzy Doe')
        ->get();
```

შეგვიძლია წამოვიღოთ ის მომხმარებლები, რომლებსაც შევსებული აქვთ რამე კონკრეტული ველი:

```
$results = $ad->search()->select(array('cn', 'displayname'))->all();
```

ასევე შეგვიძლია სკრიპტშივე დავალაგოთ მონაცემები და ისე წამოვიღოთ:

```
$results = $ad->search()->where('cn', '=', 'John*')->sortBy('cn', 'desc')-
>get();
```

შეგვიძლია პირიქით, წამოვიღოთ მომხმარებლები გარდა რომელიმე თანამშრომლისა:

```
$escapedValue = $ad->getLdapConnection()->escape('John Doe');
```

შეგვიძლია გავიგოთ მომხმარებლების ჯამური რაოდენობა:

```
$results->count();
```

ანუ ბრუნდება სტანდარტული მასივის ობიექტი, რომელზეც შესაძლებელია ნებისმიერი ოპერაციის ჩატარება და იტერაცია.

შეგვიძლია მოვფილტროთ დეტალურად მომხმარებლები, რომლებსაც არ აქვთ რამე კონკრეტული სახელი:

```
$results = $ad->search()
        ->where('objectClass', '=', $ad->getUserIdKey())
->where('cn', '!', 'John')
        ->get();
```

ან რამდენიმეს ერთად გამოკლებით:

```
$results = $ad->search()
            ->where('objectClass', '=', $ad->getUserIdKey())
            ->orWhere('cn', '!', 'John')
            ->orWhere('cn', '!', 'Suzy')
            ->get();
```

ან რამე ჯგუფში მყოფი ელემენტები:

```
$results = $ad->search()
        ->where('objectClass', '=', 'computer')
        ->get();
```

შეგვიძლია გავფილტროთ ოპერაციული სისტემის მიხედვითაც:

```
$results = $ad->search()
        ->where('objectClass', '=', 'computer')
        ->where('operatingSystem', 'starts_with', 'Windows 7')
        ->get();
```

შეგვიძლია წამოვიღოთ რომელიმე საქაღალდის ელემენტები:

```
$folderName = 'Accounting';

$results = $this->adldap->search()
->where('OU', '=', $folderName)
```

```
            ->first();
```

ყველა ფოლდერის წამოღება შეგვიძლია კოდით:

```
$folders = $ad->folder()->all();
```

ფოლდერის შექმნა ხდება შემდეგნაირად:

```
$attributes = [
    'ou_name' => 'Employees',
    'container' => [
        'Users'
    ]
];
```

ხოლო წაშლისთვის არის სკრიპტი:

```
$distinguishedName = 'OU=Accounting,OU=User
Accounts,DC=corp,DC=acme,DC=com';

$deleted = $ad->folder()->delete($distinguishedName);
```

შეგვიძლია ყველა საქაღალდის წაშლაც:

```
$folder = $ad->folder()->find('Accounting');

if(is_array($folder) && array_key_exists('dn', $folder)) {
    $ad->folder()->delete($folder['dn']);
}
```

ჯგუფების დეტალური ინფორმაცია მოგვდის შემდეგი კოდით:

```
// Get the users information
$user = $ad->user()->find('John Doe');

// Get their primary group ID
$primaryGroupId = $user['primarygroupid'];

// Get their object SID
$objectSid = $user['objectsid'];
```

```
/*
 * Get the primary groups DN by
 * passing in the users primary group ID
 * and SID
 */
$groupDn = $ad->group()->getPrimaryGroup($primaryGroupId, $objectSid);

echo $groupDn; // Returns 'CN=Domain Users,CN=Users,DC=corp,DC=acme,DC=org'
```

მომხმარებლებისთვის საძიებოდ არის შემდეგნაირად:

```
$users = $ad->user()->all();
```

ან სახელის მიხედვით:

```
$username = 'jdoe';

$user = $ad->user()->find($username);
```

მომხმარებლის ინფორმაციის წამოღება:

```
$username = 'jdoe';

$dn = $ad->user()->dn($username);
```

მომხმარებლის წაშლა:

```
$username = 'jdoe';

$deleted = $ad->user()->delete($username);
```

თუმცა მომხამრებლის წაშლა არ არის მიზანშეწონილი. უმჯობესია წაშლილთა ჯგუფის შექმნა და ყველა წასაშლელი მომხმარებლის გადატანა იქ და აღრიცხვა როგორც წაშლილების შესაბამისი უფლებებით.

adLDAP ბიბლიოთეკით შეგვიძლია მომხმარებლის პაროლის ცვლილება:

```
try
{
    $newPassword = 'newpassword123';
```

```
    $oldPassword = 'oldpassword123';

    $changed = $ad->user()->changePassword('jdoe', $newPassword,
$oldPassword);

} catch(Adldap\Exceptions\WrongPasswordException $e)
{
    return "Uh oh, you've entered the wrong old password!";
} catch(Adldap\Exceptions\PasswordPolicyException $e)
{
    return "Looks like your new password doesn't meet our requirements. Try
again."
}
```

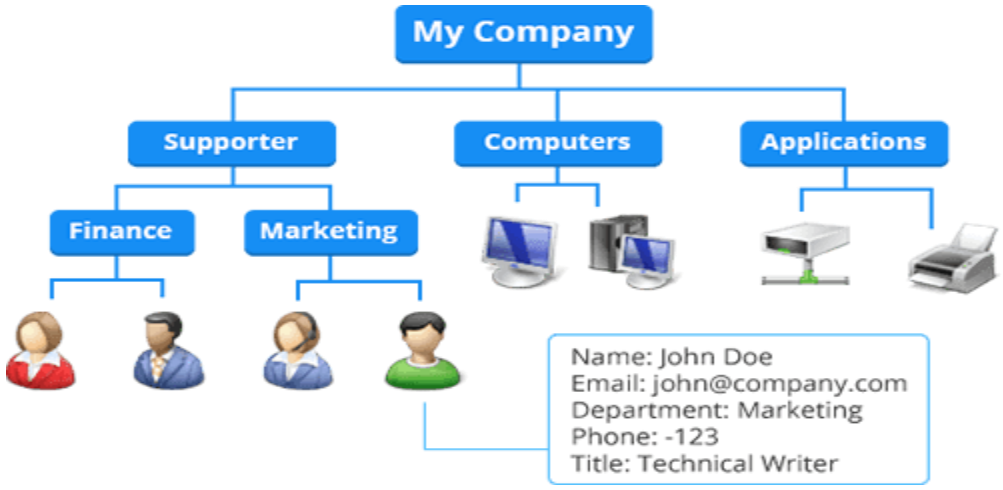
შეგვიძლია გაგება, თუ როდის გასდის მის პაროლს ვადა:

```
$results = $ad->user()->passwordExpiry('jdoe'); // Returns array|bool

$results['expires']; // Returns true / false if the users password expires
$results['has_expired']; // Returns true / false if the users password
**has** expired
$results['expiry_timestamp']; // Returns the users password expiry date in
unix time
$results['expiry_formatted']; // Returns the users password expiry date in
a formatted string ('YYYY-MM-DD HH:MM:SS')
```

შეგვიძლია გავიგოთ ბოლოს თუ როდის იყო შემოსული:

```
$time = $ad->user()->getLastLogon('jdoe'); // Returns in Unix time
$date = date('Y-m-d h:i:s', $time);
```

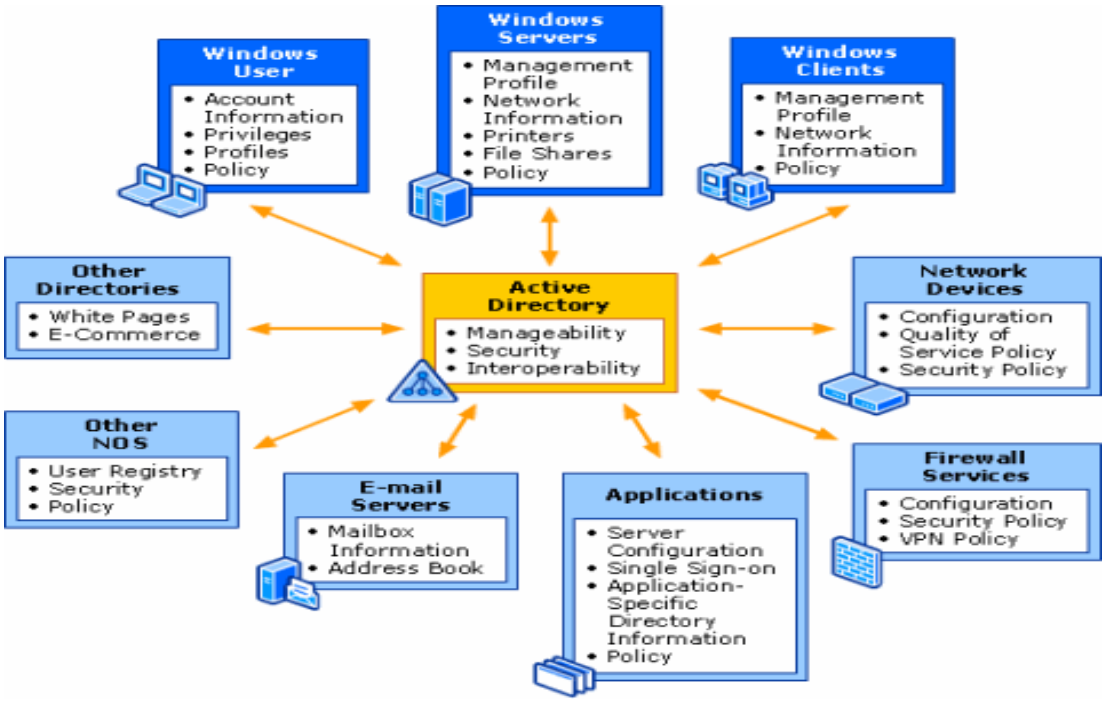
ვიზუალური სქემა, რა შედეგსაც ვღებულობთ, ნაჩვენებია შემდეგ სურათზე:



სურათი 47. AD ვიზუალიზაცია.

ანუ ActiveDirectory-ს მეშვეობით შესაძლებელია მარტივად აღიწეროს ქსელში მყოფი ნებისმიერი ელემენტი ადმინისტრირების გარეშე, კოდიდან და კერძო რეპორტების წარმოება მასზე.

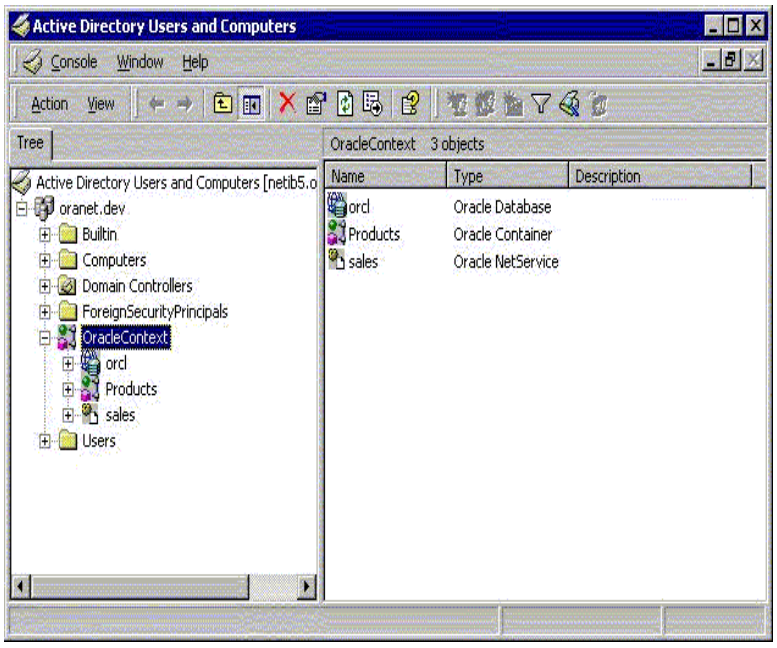
AD-ს შესაძლებლობების სტრუქტურის სრული ხე ნაჩვენებია შემდეგ გამოსახულებაზე:



სურათი 48. AD სრული შესაძლებლობების ხე.

თუმცა მხოლოდ ეს არ არის AD-ს სრული შესაძლებლობები. შუალედური ინტერპრეტატორის მეშვეობით შესაძლებელია მისი

ინტეგრირება სხვა სისტემებშიც, როგორიცაა თუნდაც Oracle-ს ბაზების ავტორიზაციის სისტემა, რის შემდეგაც ახალი ჩანართები დაემატება ჩვენს სტრუქტურას.



სურათი 49. AD ინტეგრირება Oracle ბაზებთან.

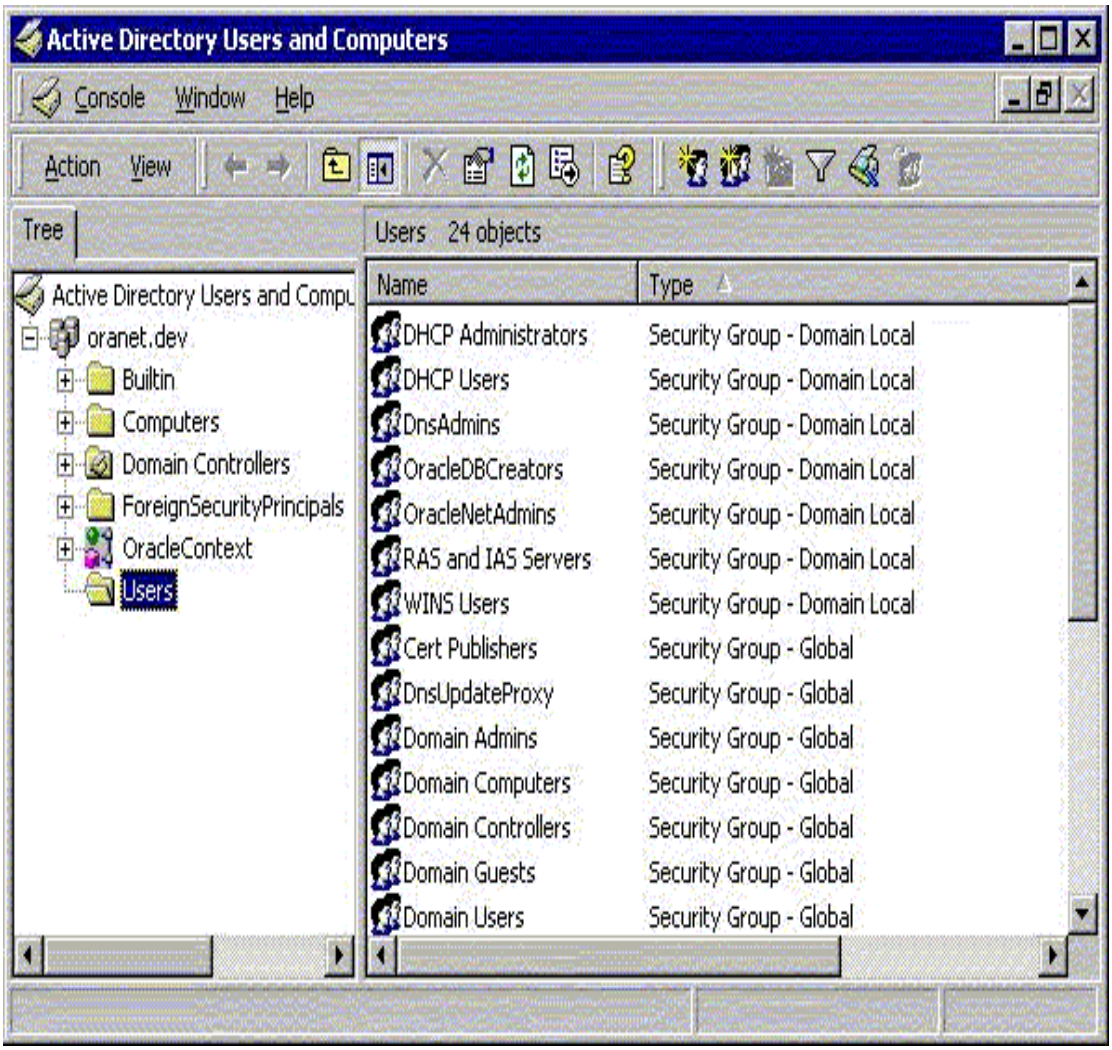
თვითონ სკრიპტის სინტაქსი კი იგივე რჩება, რაც იყო Microsoft-ის პროდუქტებთან. ერთადერთი, იცვლება დაკავშირების მისამართის ტექსტი, რომლებიც მაგალითად გამოიყურება შემდეგნაირად:

```
dsacls "CN=orcl,CN=OracleContext,OU=Example,O=Com" /G "anonymous
logon":GR

dsacls "CN=orcl,CN=OracleContext,OU=Example,O=Com" /G example\scott:GR

dsacls "CN=orcl,CN=OracleContext,OU=Example,O=Com" /R "anonymous logon"

dsacls "CN=orcl,CN=OracleContext,OU=Example,O=com" /R example\scott
```

სადაც პირველ შემთხვევაში ვცდილობთ დაკავშირებას როგორც ანონიმური მომხმარებელი, ხოლო სერვისად ვირჩევთ Generic კითხვის რეჟიმს.

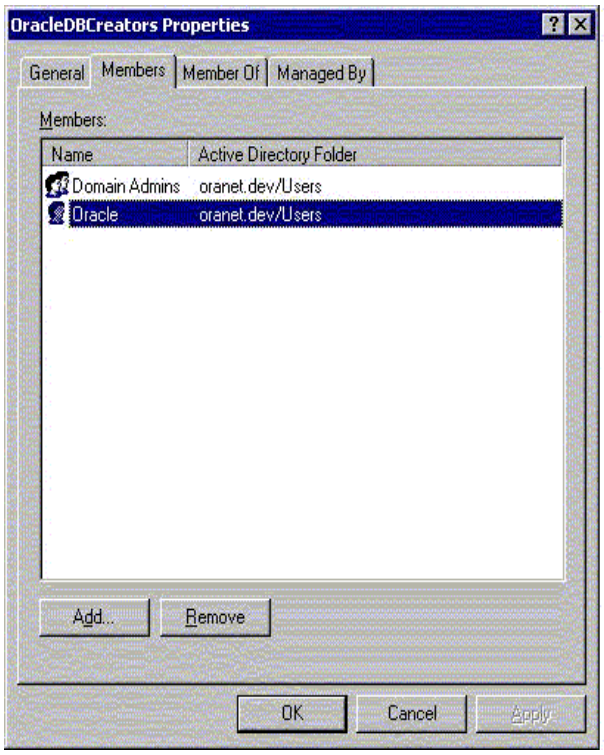
მეორე მაგალითში Generic მეთოდი ებმევა კონკრეტულ მომხმარებელს.

მესამე ხაზზე განხილულია ანონიმური ავტორიზაცია Generic მეთოდის გარეშე, ხოლო მეოთხე მაგალითი გვიჩვენებს კავშირს კონკრეტული მომხმარებლით Generic ბმის გარეშე, რის შემდგომაც მომხმარებლების ჩანართის სიაში ემატება ახალი მომხმარებლები სამართავად და Oracle ბაზებთან დასაკავშირებლად.



სურათი 50. AD + Oracle მომხმარებლები.

მათ კონკრეტულ მომხმარებელზე დაჭერით კი გამოდის მათი პირადი AD ინფორმაცია.



სურათი 51. AD Oracle მომხმარებლის ინფორმაცია.

თავი 2.3. ტექნიკური ნაწილი

ჩემს მიერ აწყობილი აპლიკაციები მუშაობენ PHP Laravel ძრავზე, რომელიც არის MVC (Model, View, Controller) სტრუქტურით, ხოლო ზოგიერთი ფუნქციის დანერგვისთვის საჭირო იყო NodeJS ფუნქციონალის დამატებაც, რომლებიც ერთმანეთთან იდეალურად მუშაობენ.

განვიხილოთ ჩემი რამდენიმე სოფტის ზოგადი ტექნიკური დოკუმენტაციების ფრანგემტები, რომლებიც მთლიანად აწყობილია და მუშაობენ AD-სთან სინქრონით:

გამშვები Commnad-ები აპლიკაციების

1. ***Adeldap*** *- სინქრონიზაცია თუ გვჭირდება, ხელით ვუშვებთ შემდეგ Command-ს:* ***php artisan adeldap:import***
    a. *მაგალითად, თუ გვინდა ინტრანეტზე უსერების სინქრონი adeldap-იდან*
        i. *Cd /var/www/intranet.ge*
        ii. *ვუშვებთ command-ს:* ***php artisan adeldap:import***

2. ***Intranet*** *- დარესტარტების ან Socket-ის ჩავარდნის შემთხვევაში ვუშვებთ შემდეგ* ***Command****-ს:****pm2 start echo.json***
3. ***Integration.corp.local*** *- დარესტარტების შემთხვევაში აუცილებლად გასაშვებია შემდეგი* ***Command: forever start socket.js***
    a. *თუ დარესტარტდა სერვერი*
        i. *cd /var/www/integration.corp.local*
        ii. *ვუშვებთ* ***putty-დან*** *ან ნებისმიერი სხვა კლიენტიდან, რომლითაც შეძლებთ სერვერზე შესვლას, Command-ს:* ***forever start socket.js***
4. ***Pizzaco*** *- ვიყენებთ შემდეგ Command, თუ სერვერი დარესტარტდა:****pm2 start echo.json***

integration.corp.local

1. ***Route***
    a. *როუტები მოთავსებულია app/Http/routes.php*
2. ***Controller***
    a. *Controller მოთავსებულია app/Http/Controllers*
        i. *შესაბამის აპლიკაციას აქვს შესაბამისი პაპკა Controller-ებში, მაგალითად: Communicator*
        ii. ***კომუნიკატორი***
            1. ***Communicator/HomeController.php***
                a. *getContractsList - გვიბრუნებს შედეგს: ყველა კონტრაქტს*
                b. *postGuarantorsList - გვიბრუნებს შედეგს: ყველა გარანტორს*
        iii. ***ტაბლო***
            1. *Tablo/HomeController.php*
                a. *GetRateVersion - გვიბრუნებს ვალუტის კურსის განახლების თარიღს*

b. *GetRate - გვიბრუნებს კურსს*

iv. ***მონაცემთა გაცვლის საგენტო***

1. *CRA/HomeController.php*

a. *postSearchApprobation - გვიბრუნებს შედეგს, თუ უკვე არის გაზაში მონაცემი*

b. *postCheckApprobation - აგზავნის პარამეტრებს შუამავალ სერვისთან რომელიც იღებს ინფორმაციას **მონაცემთა გაცვლის საგენტოდან** სერვერის მისამართი 192.168.[redacted] IIS პროექტი (CRA_Proxy)*

c. *postCheckLegally - აგზავნის პარამეტრებს შუამავალ სერვისთან, რომელიც იღებს ინფორმაციას **მონაცემთა გაცვლის საგენტოდან** სერვერის მისამართი 192.168.[redacted] IIS პროექტი (CRA_Proxy2)*

d. *postExportKeeperReport - Keeper-ის რეპორტის ამოღება*

e. *postLocalReport - ლოკალური რეპორტის ამოღება*

v. ***Mintos***

1. *Mintos/ApiController.php*

a. *sendPayments - გადახდების გაგზავნის მეთოდი*

b. *rebuyLoan - სესხის გამოსყიდვის მეთოდი*

2. *Mintos/HomeController.php*

a. *makePostRequest - სესხის გაგზავნა Mintos-ზე*

vi. ***UCC***

1. *UCC/ApiController.php*

a. *postAddTransaction - ლუმენიდან მიღებული ტრანზაქციის ინფორმაცის ჩაწერა Mintos DB-ში*

2. *UCC/HomeController.php*

a. *getListTransactions - ყველა ტრანზაქცია, მათ შორის დაარქივებული ტრანზაქციები*

b. *getProcessTransaction - დამუშავებული ტრანზაქციები*

c. *getRestoreTransaction - ტრანზაქციის აღდგენა*

d. *getLockTransaction - Locked ტრანზაქციები ოპერატორების მიერ*

e. *getUnlockTransaction - ტრანზაქციის Lock-ის მოხსნა*

3. ***Model***

a. *მოთავსებულია app პაპკაში შესაბამისი აპლიკაცის დასახელებით*

4. ***Jobs***

a. *App/console/Kernel.php - მოთავსებულია ყველა integration.corp.localSchedule*

Intranet.ge ჩათი

1. ***Controller***

a. *App/Http/Controllers/ChatController.php*

i. *postChatUsers - გვიბრუნებს შედეგს აქტიური იუზერების JSON სახით*

ii. *postViewMessages - გვიბრუნებს მესიჯებს*

iii. *postSendMessage - შეტყობინების გაგზავნა*

iv. *postContactsState - ჩათის მდგომარების ნახვა*

v. *postChatState - გახსნილი ფანჯრების მდგომარეობა*
vi. *postSeenMessages - წაკითხული მესიჯების ნახვა*
vii. *postUnseenMessages - გვაძლევს შედეგს წაუკითხავი მესიჯების*
viii. *postChatGroup - მოაქვს შექმნილი ჯგუფები*
ix. *postAddGroup - ამატებს ახალ ჯგუფებს*
x. *postEditGroup - ჯგუფის ედიტირების მეთოდი*
xi. *postLeaveGroup - ჯგუფის დატოვების მეთოდი*

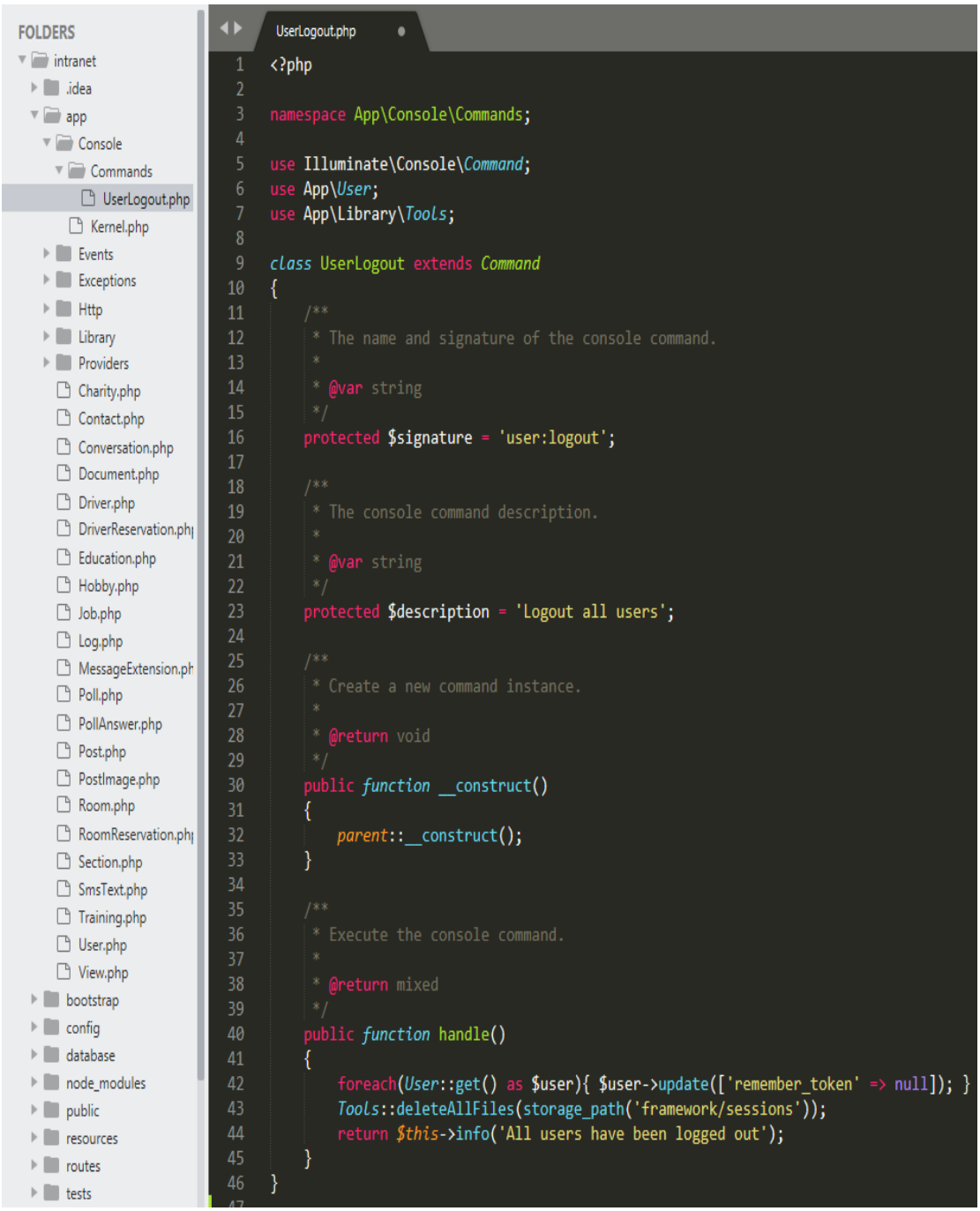
b. *App/Http/Controllers/HomeController.php - მოთავსებულია ყველა ინტრანეტის მეთოდი, მაგალითად ოთახის დაჯავშნა, თანამშრომლების სია და სხვა*

2. ***Intranet*** *- დარესტარტების ან Socket-ის ჩავარდნის შემთხვევაში ვუშვებთ შემდეგ* ***Command****-ს:****laravel-echo-server start***

Businessloan.ge

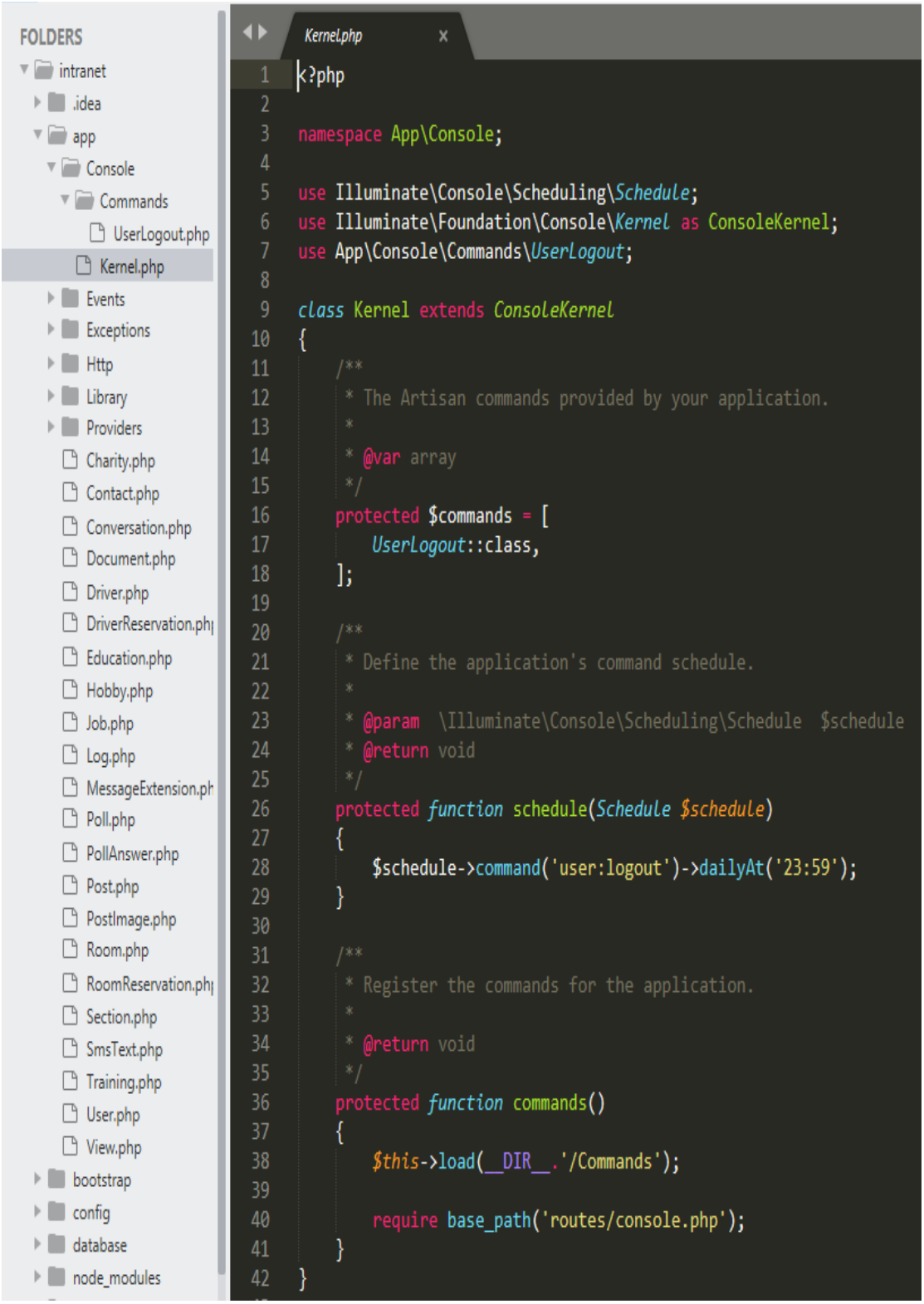
1. ***Controller***
   a. *App/Http/FrontContoller.php - მოთავსებულია ყველა მეთოდი ვებ გვერდის*
   b. *Methods*
      i. *GetInterestRate - მეთოდი გვიბრუნებს ეფექტურ საპროცენტო განაკვეთს დღეების მიხედვით*
      ii. *Step1 - მოცემულია განაცხადის ფორმის პირველი გვერდი*
      iii. *Step2 - მოცემულია განაცხადის მეორე გვერდი*
      iv. *Step3 - აგზავნის ინფორმაციას Qcash-ში სერვისით*
      v. *ContactUs - აგზავნის info@big.com.ge-ზე შეტყობინებას საკონტაქტო გვერდიდან*
2. ***View***
   a. *Resources/views/front-ში მოთავსებულია ყველა გვერდის Blade*

ერთ-ერთი დადებითი მხარე, რაც შეგვიძლია გავაკეთოთ PHP ADთი, არის მომხმარებლების სესიის გაუქმება, ანუ ნებისმიერ კლიენტს შეგვიძლია კოდიდან წავუშალოთ წვდომა რესურსებზე. [5] ფუნქცია, რომელიც ახდენს ამის რეალიზაციას გამოიყურება შემდეგნაირად:



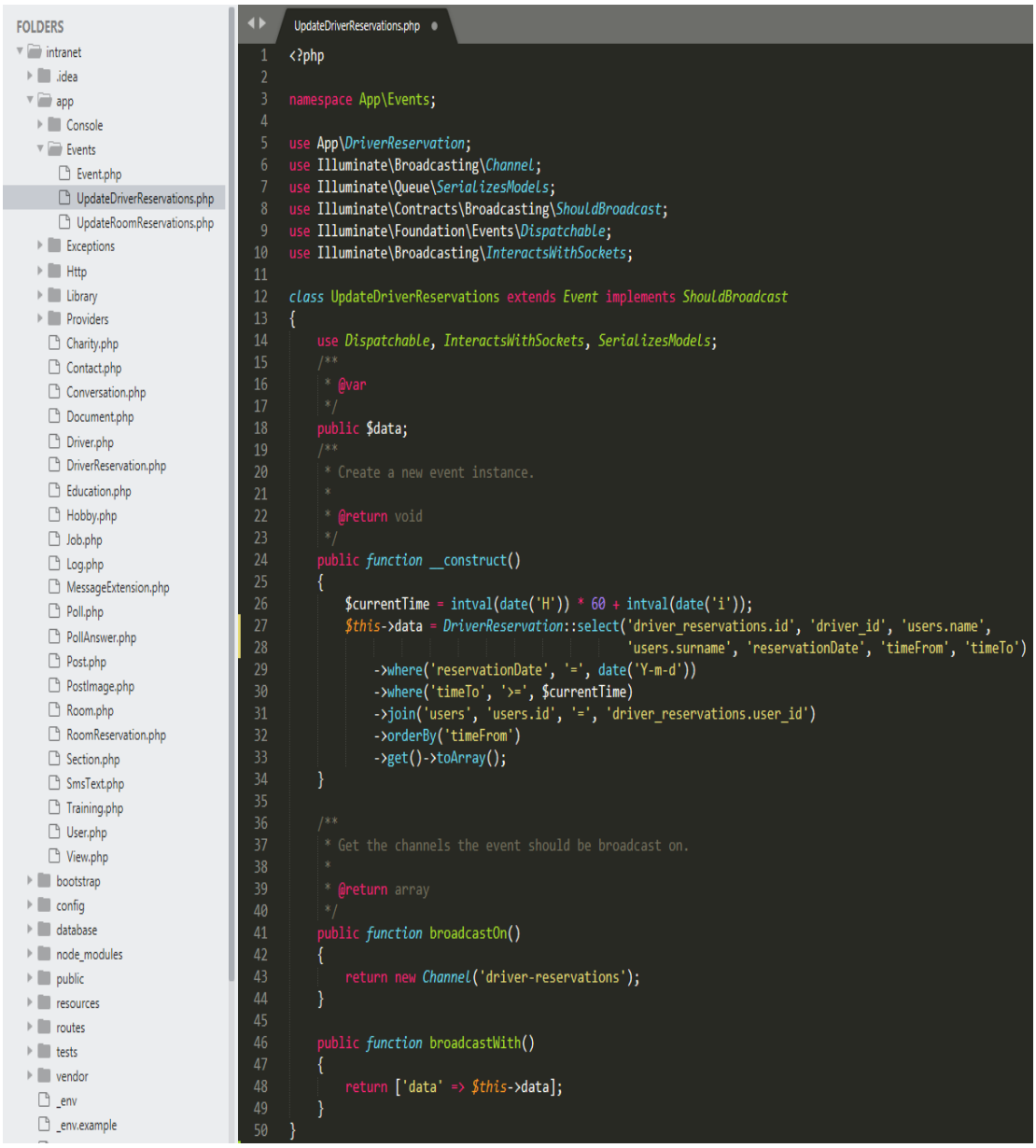
კოდი 1. UserLogout კლასი.

აგრეთვე ვახდენ ყველა მომხმარებლის გასვლის სიმულაციას ღამის 12 საათზე, რადგან შევძლო მათი აქტიურობის შედარების ანალიზის აგება. [6] ამის მარეგულირებელი PHP კოდი შედგება შემდეგი კომპონენტებისგან:



კოდი 2. Kernel კლასი.

ამ მიდგომაზე შემიძლია უამრავი გამოყენების მაგალითის მოყვანა. ერთ-ერთი მათგანი არის მძღოლების დაჯავშნის ფუნქციონალი, რომელიც ხდება AD-დან და მისი კოდის უკანა მხარე დაწერილია Laravel სრული ინტეგრაციებით.



კოდი 3. UpdateDriveReservations კლასი.

მოცემული მაგალითი კითხულობს ყველა მძღოლის მომხმარებელს ბაზიდან და მათი თავისუფლების სტატუსის მიხედვით გამოაქვს დასკვნა, შესაძლებელია მათი რეზერვაცია, თუ - არა.

შემდეგი საჭირო და მნიშვნელოვანი პარამეტრი, რისი გაკეთებას შეიძლება ადმინისტრატორის გარეშე, არის მომხმარებლის პაროლის დარესეტება. მისი კონტროლერის კლასი ნაჩვენები მაქვს შემდეგ სურათზე:

ForgotPasswordController.php

```php
<?php

namespace App\Http\Controllers\Auth;

use App\Http\Controllers\Controller;
use Illuminate\Foundation\Auth\SendsPasswordResetEmails;

class ForgotPasswordController extends Controller
{
    /*
    |--------------------------------------------------------------------------
    | Password Reset Controller
    |--------------------------------------------------------------------------
    |
    | This controller is responsible for handling password reset emails and
    | includes a trait which assists in sending these notifications from
    | your application to your users. Feel free to explore this trait.
    |
    */

    use SendsPasswordResetEmails;

    /**
     * Create a new controller instance.
     *
     * @return void
     */
    public function __construct()
    {
        $this->middleware('guest');
    }
}
```

კოდი 4. ForgotPasswordController კლასი.

მომხმარებლების ავტორიზაციის კოდი შედგება შემდეგი პუნქტებისგან:

- სახელის ამოკითხვა;
- ავტორიზაცია;
- სახელის შედარება;
- შეტყობინების გამოტანა;
- გასვლა;
- გაგდება;
- ლოგი.

კოდის ფრაგმენტი გამოიყურება შემდეგნაირად:

```php
LoginController.php
    protected $redirectTo = '/';

    /**
     * Create a new controller instance.
     *
     * @return void
     */
    public function __construct()
    {
        $this->middleware('guest')->except('logout');
    }

    public function username()
    {
        return 'username';
    }

    public function getLogin()
    {
        return $this->showLoginForm();
    }

    public function postLogin(Request $request)
    {
        if(User::where('username', '=', $request->only('username'))->where('username', '<>', '')->exists())
        {
            $this->login($request);
            if(Auth::check())
            {
                Logger::loginLog();
                return redirect('/');
            }
            else{ return back(); }
        }
        else{ return back()->withErrors(['username' => 'ასეთი მომხმარებელი ვერ მოიძებნა']); }
    }

    public function logout(Request $request)
    {
        $this->guard()->logout();

        $request->session()->invalidate();

        return redirect('/');
    }
}
```

კოდი 5. LoginController კლასი.

თუ ავტორიზაცია ვერ მოხერხდა, ან მომხმარებელი ვერ მოიძებნა, გამომაქვს შესაბამისი შეტყობინება, მაგალითად - "ასეთი მომხმარებელი ვერ მოიძებნა". [4]

ავტორიზაციისას ვქმნი სესია, ხოლო გასვლისას - ვხურავ მას. ამავდროულად ვახდენ ავტორიზაციების აღრიცხვას, რასაც შემდგომ ვაანალიზებ.

თითოეული მომხმარებლისთვის მაქვს მისი სახელის წამოღების ფუნქცია, რაც ნაჩვენები არის ზემოთ აღნიშნულ კოდში.

როგორც უკვე ავღნიშნე, PHP LDAP-ს აქვს ფაქტიურად ყველა ის შესაძლებლობა, [10] რაც Microsoft AD-ში შესვლისას. ერთ-ერთი ძალიან გამოსადეგი ფუნქცია არის ახალი მომხმარებლის დამატების შესაძლებლობა, რომელიც აღწერილი მაქვს შემდეგ კოდში:

RegisterController.php

```php
    /**
     * Where to redirect users after registration.
     *
     * @var string
     */
    protected $redirectTo = '/home';

    /**
     * Create a new controller instance.
     *
     * @return void
     */
    public function __construct()
    {
        $this->middleware('guest');
    }

    /**
     * Get a validator for an incoming registration request.
     *
     * @param  array  $data
     * @return \Illuminate\Contracts\Validation\Validator
     */
    protected function validator(array $data)
    {
        return Validator::make($data, [
            'name' => 'required|string|max:255',
            'email' => 'required|string|email|max:255|unique:users',
            'password' => 'required|string|min:6|confirmed',
        ]);
    }

    /**
     * Create a new user instance after a valid registration.
     *
     * @param  array  $data
     * @return \App\User
     */
    protected function create(array $data)
    {
        return User::create([
            'name' => $data['name'],
            'email' => $data['email'],
            'password' => bcrypt($data['password']),
        ]);
    }
}
```

კოდი 6. RegisterController კლასი.

ასევე გვაქვს პაროლების დარესეტების კონტროლერი:

ResetPasswordController.php

```php
<?php

namespace App\Http\Controllers\Auth;

use App\Http\Controllers\Controller;
use Illuminate\Foundation\Auth\ResetsPasswords;

class ResetPasswordController extends Controller
{
    /*
    |--------------------------------------------------------------------------
    | Password Reset Controller
    |--------------------------------------------------------------------------
    |
    | This controller is responsible for handling password reset requests
    | and uses a simple trait to include this behavior. You're free to
    | explore this trait and override any methods you wish to tweak.
    |
    */

    use ResetsPasswords;

    /**
     * Where to redirect users after resetting their password.
     *
     * @var string
     */
    protected $redirectTo = '/home';

    /**
     * Create a new controller instance.
     *
     * @return void
     */
    public function __construct()
    {
        $this->middleware('guest');
    }
}
```

კოდი 7. ResetPasswordController კლასი.

რა თქმა უნდა შეიძლებოდა ამ ყველაფრის გაკეთება ბარემ Microsoft-ის პროდუქტზე C#-ზე, მაგრამ მას ბევრი უარყოფითი მხარე აქვს, როგორიცაა მძიმე ფუნქციონალი და კოდი, დროის დიდი რესურსი, მძლავრი მანქანის საჭიროება, მეტი მოვლა.

PHP არის უნივერსალური ენა. მას შეუძლია მუშაობა როგორც დიდ სერვისებთან, ასევე მიკროსერვისებთან ძალიან სწრაფად და მარტივად, დიდი რესურსის საჭიროების გარეშე, რის გამოც საბოლოო არჩევანი გავაკეთე მასზე.

თავი 2.4. ჩათი

ინტრანეტის ერთ-ერთი ყველაზე მნიშვნელოვანი ნაწილი არის ჩათი. ჩათსაც ვუკეთებ ავტორიზაციას იგივე AD-დან PHP-ის მეშვეობით.განვიხილოთ რამდენიმე ძირითადი ფუნქციის კოდის ფრაგმენტი:

```php
// moaqvs aqtiuri userebi json shi
public function postChatUsers(Request $request)
{
    $data = [];
    foreach(User::active()->where('id', '!=', Auth::user()->id)->orderBy('name')->get() as $user) {
        $data[] = ['id' => $user->id, 'name' => $user->name, 'surname' => $user->surname];
    }
    return json_encode($data);
}
```

კოდი 8. postChatUsers ფუნქცია.

ეს ფუნქცია გამოიყენება ჩათში ტექსტის გადაგზავნისას, რომელიც ამოწმებს მომხმარებლის სახელს ავტორიზაციის ვალიდურობაზე.

ჯგუფების სამართავად და სამუშაოდ გამოიყენება შემდეგი ფუნქცია:

```php
// moaqvs sheqmnili jgufebi
public function postChatGroup(Request $request)
{
    $data = [];
    $group = GroupConversation::findOrfail($request->input('group_id'));
    $user = User::findOrFail($group->creator_id);
    $data['id'] = $group->id;
    $data['creator_id'] = $group->creator_id;
    $data['creator'] = $user->name.' '.$user->surname;
    $data['name'] = $group->name;
    $data['leave'] = $group->leave;
    $data['edit'] = $group->edit;
    foreach($group->users()->get() as $user)
    {
        $data['users'][] = ['id' => $user->id, 'name' => $user->name.' '.$user->surname];
    }
    return json_encode($data);
}
```

კოდი 9. postChatGoup ფუნქცია.

ზემოთ მითითებული ფუნქცია სინქრონირდება მომხმარებელთა ცენტრალიზებულ ბაზასთან და მოაქვს ჯგუფში აქტიურობა.

მომხმარებლებს თვითონაც აქვთ ჯგუფური ჩათების შექმნის შესაძლებლობა, რასაც მათ აძლევს შემდეგი ფუნქცია:

```php
// amatebs jgufs
public function postAddGroup(Request $request)
{
    $leave = ($request->input('leave') == 1 ? 1 : 0);
    $edit = ($request->input('edit') == 1 ? 1 : 0);
    if(ChatManager::addGroup($request->input('name'),
        $leave, $edit, $request->input('users'))){ return 'success'; }
    else{ return 'error'; }
}
```

კოდი 10. postAddGroup ფუნქცია.

ჯგუფი იქმნება კონკრეტული მომხმარებლის სახელზე, რომელიც ხდება ამ ჯგუფის მფლობელი. მხოლოდ მას აქვს ამ ჯგუფის დახურვის

საშუალება, ისევე როგორც მისი ძირითადი მმართველობის მოვალეობის შესრულება. რედაქტირებაზე პასუხისმგებელია შემდეგი ფუნქცია:

```
// aeditebs jgufs
public function postEditGroup(Request $request)
{
    $leave = ($request->input('leave') == 1 ? 1 : 0);
    $edit = ($request->input('edit') == 1 ? 1 : 0);
    if(ChatManager::editGroup($request->input('id'), $request->input('name'),
     $leave, $edit, $request->input('users'))){ return 'success'; }
    else{ return 'error'; }
}
```

კოდი 11. postEditGroup ფუნქცია.

მომხმარებლების ინფორმაციის წამოღებისთვის მაქვს შემდეგი ორი ფუნქცია, რომლებიც უკავშირდებიან ბაზას, რომელშიც უკვე წამოღებულია AD-ს ყველა აქტიური მომხმარებელი, რომელიც სინქრონირდება დღეში ერთხელ, ვიღებ ყოველი მათგანის:

- სურათს,
- სახელს,
- გვარს,
- თანამდებობას,
- დეპარტამენტს,
- ფილიალს,
- შიდა ნომერს,
- სამსახურის ტელეფონს,
- პირად ტელეფონს,
- სამსახურის ფოსტას,
- დაბადების თარიღს.

ასევე ვაერთიანებ რამდენიმე მნიშვნელოვან ცხრილთან საჭიროებისამებრ, მაგალითად - მიმდინარე სტატუსთან.

მეორე ფუნქციას გამოაქვს ის მომხმარებლები, რომელიც მოსულები არიან ბოლო თვეში, რათა ვაჩვენო, როგორც ახალი თანამშრომლები:

```php
public function listPersonnel()
{
    return Datatables::collection(User::active()->select('users.id',
        DB::raw("CONCAT(picture, '/', gender) as picture"), 'name', 'surname',
         'position', 'department', 'service_center', 'inner_number',
         'work_mobile', 'personal_mobile', 'work_email', 'DOB')
        ->leftJoin('jobs', 'users.id', '=', 'jobs.user_id')
        ->leftJoin('contacts', 'users.id', '=', 'contacts.user_id')
        ->where('job_status', '=', 'current')
        ->get())->make(true);
}

public function listNewPersonnel()
{
    return Datatables::collection(User::active()->select('users.id',
        DB::raw("CONCAT(picture, '/', gender) as picture"), 'name', 'surname',
         'position', 'department', 'service_center', 'inner_number', 'work_mobile',
         'personal_mobile', 'work_email', 'DOB')
        ->leftJoin('jobs', 'users.id', '=', 'jobs.user_id')
        ->leftJoin('contacts', 'users.id', '=', 'contacts.user_id')
        ->where('job_status', '=', 'current')
        ->orWhere(DB::raw('DATEDIFF(job_start, NOW())'), '<=', 30)
        ->orWhere(DB::raw('DATEDIFF(jobs.created_at, NOW())'), '<=', 30)
        ->get())->make(true);
}
```

კოდი 12. Personnel ფუნქციები.

თითოეულ მომხმარებელს ეძლევა საშუალება, რომ მიიღოს მონაწილეობა გამოკითხვებში, სადაც ფიქსირდება ერთხელ თავისი ნომრით და მეორედ აღარ ეძლევა იგივე გამოკითხვაში ხმის მიცემის საშუალება, რადგან ვიმახსოვრებთ მათ ყველა ქმედებებს.

ხმის მიცემის ფუნქცია გამოიყურება შემდეგნაირად:

```php
public function postPoll(PollRequest $request)
{
    if(Poll::where('id', '=', $request->input('poll_id'))->where('poll_status', '=', 1)->exists())
    {
        if(!PollAnswer::answered($request->input('poll_id')))
        {
            PollAnswer::create([
                'user_id' => Auth::user()->id,
                'poll_id' => $request->input('poll_id'),
                'answer' => $request->input('answer')
            ]);
            return back();
        }
        else{ return back()->withErrors('AlreadyAnswered', 'თქვენ უკვე უპასუხეთ ამ გამოკითხვას'); }
    }
    else{ return back()->withErrors('NotActivePoll', 'გამოკითხვა არ არის აქტიური'); }
}
```

კოდი 13. postPoll ფუნქცია.

ამის იდენთიფიცირებისთვის ვიყენებ დამატებით სტატუსს, რომელსაც ხმის მიცემამდე ვამოწმებ და თუ აღმოჩნდა, რომ მომხმარებელს უკვე აქვს მიცემული ხმა, მაშინ გამომაქვს მისთვის შესაბამისი შეტყობინება.

მომხმარებლებზე მაქვს მინიჭებული უფლებები. თუ მომხმარებელს აქვს გამოკითხვის შექმნის უფლება, მაშინ მას შეეძლება დაამატოს ახალი გამოკითხვა, რაც კეთდება შემდეგი ფუნქციით და ებმევა მას:

```php
public function postAddPoll(AddPollRequest $request)
{
    if(!Auth::user()->hasPermission('edit_polls')){ return response(view('errors.403'), 403); }
    Poll::create([
        'user_id' => Auth::user()->id,
        'poll_question' => $request->input('poll_question'),
        'answer_1' => $request->input('answer_1'),
        'answer_2' => $request->input('answer_2'),
        'answer_3' => $request->input('answer_3'),
        'answer_4' => $request->input('answer_4'),
        'poll_status' => $request->input('poll_status')
    ]);
    return back()->withErrors(['successNotice' => 'გამოკითხვის დამატება წარმატებულია']);
}
```

კოდი 14. postAddPoll ფუნქცია.

ასევე არის შესაძლებლობა გამოკითხვის რედაქტირების, თუ მომხმარებელს აქვს ამის შესაბამისი უფლება. ფუნქციის კოდი შედგება შემდეგი ნაწილებისგან:

```php
public function getEditPoll($id)
{
    if(!Auth::user()->hasPermission('edit_polls')){ return response(view('errors.403'), 403); }
    return view('editPoll')->with('tab', 'management')->with('poll', Poll::where('id', '=', $id)->first());
}
```

კოდი 15. getEditPoll ფუნქცია.

მომხმარებლებს აქვთ როგორც გამოკითხვის ტექსტის, აგრეთვე შეკითხვების რედაქტირება, თუ აქვთ შესაბამისი უფლება. კითხვების ტექსტებს არედაქტირებს შემდეგი ფუნქცია:

```php
public function postEditPoll(EditPollRequest $request, $id)
{
    if(!Auth::user()->hasPermission('edit_polls')){ return response(view('errors.403'), 403); }
    Poll::where('id', '=', $id)->update([
        'poll_question' => $request->input('poll_question'),
        'answer_1' => $request->input('answer_1'),
        'answer_2' => $request->input('answer_2'),
        'answer_3' => $request->input('answer_3'),
        'answer_4' => $request->input('answer_4'),
        'poll_status' => $request->input('poll_status')
    ]);
    return back()->withErrors(['successNotice' => 'გამოკითხვის რედაქტირება წარმატებულია']);
}
```

კოდი 16. postEditPoll ფუნქცია.

აგრეთვე შესაძლებელია შექმნილი გამოკითხვის წაშლა შემდეგნაირად:

```php
public function postRemovePoll(RemovePollRequest $request)
{
    if(!Auth::user()->hasPermission('edit_polls')){ return response(view('errors.403'), 403); }
    if(PollAnswer::where('poll_id', '=', $request->input('value'))->count() > 0)
    { return back()->withErrors(['errorNotice' => 'დაუშვებელია პასუხგაცემული გამოკითხვის წაშლა']); }
    Poll::where('id', '=', $request->input('value'))->delete();
    return back()->withErrors(['successNotice' => 'გამოკითხვა წარმატებით წაიშალა']);
}
```

კოდი 17. postRemovePoll ფუნქცია.

მომხმარებლის გადამოწმებისთვის მაქვს შემდეგი ფუნქცია:

```php
public function getAddUser()
{
    if(!Auth::user()->hasPermission('edit_users')){ return response(view('errors.403'), 403); }
    $managers = User::select('id', DB::raw("CONCAT(name, ' ',surname) as name"))->active()->get();
    $AD_users = Adldap::search()->users()->get();
    $intranet_users = User::get();
    $AD_usernames = [];
    $AD_emails = [];
    $intranet_usernames = [];
    $intranet_emails = [];
    foreach($AD_users as $AD_user)
    {
        if(strlen($AD_user->samaccountname[0]) > 0){ $AD_usernames[] = $AD_user->samaccountname[0]; }
        if(strlen($AD_user->mail[0]) > 0){ $AD_emails[] = $AD_user->mail[0]; }
    }
    foreach($intranet_users as $intranet_user)
    {
        if(strlen($intranet_user->username) > 0){ $intranet_usernames[] = $intranet_user->username; }
        if(strlen($intranet_user->AD_email) > 0){ $intranet_emails[] = $intranet_user->AD_email; }
    }
    $usernames = array_diff($AD_usernames, $intranet_usernames);
    $emails = array_diff($AD_emails, $intranet_emails);
    return view('addUser')
        ->with('tab', 'management')
        ->with('managers', $managers)
        ->with('usernames', $usernames)
        ->with('emails', $emails);
}
```

კოდი 18. getAddUser ფუნქცია.

მომხმარებლის შექმნის ღილაკზე დაჭერისას იგი ამოწმებს მისი შექმნის შესაძლებლობას და გამოაქვს შესაბამისი სტატუსის შეტყობინება.

მომხმარებლის შექმნის და მისი ბაზაში ჩაწერის ფუნქცია მთელი თავისი დამატებითი ინფორმაციით არის შემდეგი:

public function postAddUser(AddUserRequest $request)

{

```
        if(!Auth::user()->hasPermission('edit_users')){ return
response(view('errors.403'), 403); }
        $user = User::create([
                'username' => $request->input('username'),
                'AD_name' => '',
                'AD_email' => $request->input('AD_email'),
                'name' => $request->input('name'),
                'surname' => $request->input('surname'),
                'picture' => '',
                'gender' => $request->input('gender'),
                'DOB' => $request->input('year').'-'.$request->input('month').'-
'.$request->input('day'),
                'user_status' => $request->input('user_status')
        ]);
        Contact::create([
                'user_id' => $user->id,
                'work_email' => $request->input('work_email'),
                'personal_email' => $request->input('personal_email'),
                'home_phone' => $request->input('home_phone'),
                'work_mobile' => $request->input('work_mobile'),
                'personal_mobile' => $request->input('personal_mobile'),
                'inner_number' => $request->input('inner_number')
        ]);
        Job::create([
                'user_id' => $user->id,
                'position' => $request->input('position'),
                'department' => $request->input('department'),
                'service_center' => $request->input('service_center'),
```

```
                'job_start' => $request->input('job_start_year').'-'.$request->input('job_start_month').'-'.$request->input('job_start_day'),
                'job_finish' => '',
                'manager_id' => $request->input('manager_id'),
                'job_status' => 'current'
        ]);
        if(!empty($request->input('university_name')) && !empty($request->input('degree')) && intval($request->input('education_start_day')) > 0 && intval($request->input('education_start_month')) > 0 && intval($request->input('education_start_year')) > 0 && intval($request->input('education_finish_day')) > 0 && intval($request->input('education_finish_month')) > 0 && intval($request->input('education_finish_year')) > 0)
        {
                Education::create([
                        'user_id' => $user->id,
                        'university_name' => $request->input('university_name'),
                        'degree' => $request->input('degree'),
                        'education_start' => $request->input('education_start_year').'-'.$request->input('education_start_month').'-'.$request->input('education_start_day'),
                        'education_finish' => $request->input('education_finish_year').'-'.$request->input('education_finish_month').'-'.$request->input('education_finish_day'),
                ]);
        }
        if(!empty($request->input('training_name')) && intval($request->input('training_start_day')) > 0 && intval($request->input('training_start_month')) > 0 && intval($request->input('training_start_year')) > 0 && intval($request->input('training_finish_day')) > 0 && intval($request->input('training_finish_month')) > 0 && intval($request->input('training_finish_year')) > 0)
        {
                Training::create([
                        'user_id' => $user->id,
                        'training_name' => $request->input('training_name'),
                        'training_start' => $request->input('training_start_year').'-'.$request->input('training_start_month').'-'.$request->input('training_start_day'),
                        'training_finish' => $request->input('training_finish_year').'-'.$request->input('training_finish_month').'-'.$request->input('training_finish_day'),
                ]);
        }
        if(!empty($request->input('skill_name')))
        {
                Skill::create([
                        'user_id' => $user->id,
                        'skill_name' => $request->input('skill_name'),
                ]);
        }
        return redirect('/');
    }
}
```

```
>input('training_start_year')) > 0 && intval($request->input('training_finish_day'))
> 0 && intval($request->input('training_finish_month')) > 0 && intval($request-
>input('training_finish_year')) > 0 && !empty($request->input('training_place')))
        {
                Training::create([
                        'user_id' => $user->id,
                        'training_name' => $request->input('training_name'),
                        'training_start' => $request->input('training_start_year'). '-
'.$request->input('training_start_month'). '-'.$request->input('training_start_day'),
                        'training_finish' => $request->input('training_finish_year'). '-
'.$request->input('training_finish_month'). '-'.$request-
>input('training_finish_day'),
                        'training_place' => $request->input('training_place')
                ]);
        }
        if(!empty($request->input('hobby_name')))
        {
                Hobby::create([
                        'user_id' => $user->id,
                        'hobby_name' => $request->input('hobby_name'),
                        'hobby_description' => $request->input('hobby_description')
                ]);
        }
        if(Auth::user()->hasPermission('edit_permissions'))
        {
                if(count($request->input('permissions')) > 0)
                {
                        foreach($request->input('permissions') as $permission)
                        {
```

```
                    try{ $user->givePermissionTo($permission); }
                    catch (Exception $e)
                    {
                        if($e instanceof PermissionDoesNotExist)
                        {
                            Permission::create(['name' =>
$permission]);
                            $user->givePermissionTo($permission);
                        }
                    }
                }
            }
        }
        return back()->withErrors(['operation' => 'success']);
}
```

ხოლო მომხმარებლის პირადი ინფორმაციის რედაქტირებისთვის არის შემდეგი კოდი, რომელიც ამზადებს მომხმარებელს გაზებში განახლებისთვის:

```php
public function getEditUser($id)
{
    if(!Auth::user()->hasPermission('edit_users')){ return response(view('errors.403'), 403); }
    $managers = User::select('id', DB::raw("CONCAT(name, ' ',surname) as name"))->active()->get();
    $user = User::where('id', '=', $id)->firstOrFail();
    $AD_users = Adldap::search()->users()->get();
    $exists = false;
    foreach($AD_users as $AD_user){ if($user->username == $AD_user->samaccountname[0]){ $exists = true; } }
    if(!$exists){ $AD_users->push(Adldap::make()->user(['samaccountname' => $user->username, 'mail' => $user->AD_email])); }
    $contacts = Contact::where('user_id', '=', $id)->first();
    $jobs = $user->jobs()->first();
    $education = $user->educations()->first();
    $trainings = $user->trainings()->first();
    $hobbies = $user->hobbies()->first();
    return view('editUser')
        ->with('tab', 'management')
        ->with('managers', $managers)
        ->with('user', $user)
        ->with('AD_users', $AD_users)
        ->with('contacts', $contacts)
        ->with('jobs', $jobs)
        ->with('education', $education)
        ->with('trainings', $trainings)
        ->with('hobbies', $hobbies);
}
```

კოდი 19. getEditUser ფუნქცია.

თვითონ განახლების კოდი კი მუშაობს შემდეგ პრინციპებზე დაყრდნობით, რომელიც იყენებს ყველა იმ ცხრილსა და კლასის ობიექტს, რომელიც ზემო კოდში არის ნაჩვენები და ჩამოთვლილი, რომელიც ეყრდნობა მომხმარებლის კოდს, როგორც სხვა ყველა ფუნქციაში, [7] რომელიც მუშაობს AD-სთან სინქრონზე დაფუძნებით და მასთან სრული ინტეგრაციით:

```
public function postEditUser(EditUserRequest $request, $id)
{
    if(!Auth::user()->hasPermission('edit_users')){ return
response(view('errors.403'), 403); }
    User::where('id', '=', $id)->update([
        'username' => $request->input('username'),
        'AD_email' => $request->input('AD_email'),
        'name' => $request->input('name'),
        'surname' => $request->input('surname'),
        'gender' => $request->input('gender'),
        'DOB' => $request->input('year').'-'.$request->input('month').'-
'.$request->input('day'),
        'user_status' => $request->input('user_status')
    ]);
    Contact::where('user_id', '=', $id)->update([
        'work_email' => $request->input('work_email'),
        'personal_email' => $request->input('personal_email'),
        'home_phone' => $request->input('home_phone'),
        'work_mobile' => $request->input('work_mobile'),
        'personal_mobile' => $request->input('personal_mobile'),
        'inner_number' => $request->input('inner_number')
    ]);
    Job::where('user_id', '=', $id)->update([
```

```
                'position' => $request->input('position'),
                'department' => $request->input('department'),
                'service_center' => $request->input('service_center'),
                'job_start' => $request->input('job_start_year').'-'.$request->input('job_start_month').'-'.$request->input('job_start_day'),
                'job_finish' => $request->input('job_finish_year').'-'.$request->input('job_finish_month').'-'.$request->input('job_finish_day'),
                'manager_id' => $request->input('manager_id')
        ]);
        if(!empty($request->input('university_name')) && !empty($request->input('degree')) && intval($request->input('education_start_day')) > 0 && intval($request->input('education_start_month')) > 0 && intval($request->input('education_start_year')) > 0 && intval($request->input('education_finish_day')) > 0 && intval($request->input('education_finish_month')) > 0 && intval($request->input('education_finish_year')) > 0)
        {
                if(Education::where('user_id', '=', $id)->exists())
                {
                        Education::where('user_id', '=', $id)->update([
                                'university_name' => $request->input('university_name'),
                                'degree' => $request->input('degree'),
                                'education_start' => $request->input('education_start_year').'-'.$request->input('education_start_month').'-'.$request->input('education_start_day'),
                                'education_finish' => $request->input('education_finish_year').'-'.$request->input('education_finish_month').'-'.$request->input('education_finish_day'),
```

```
                ]);
            }
            else
            {
                Education::create([
                    'user_id' => $id,
                    'university_name' => $request->input('university_name'),
                    'degree' => $request->input('degree'),
                    'education_start' => $request->input('education_start_year').'-'.$request->input('education_start_month').'-'.$request->input('education_start_day'),
                    'education_finish' => $request->input('education_finish_year').'-'.$request->input('education_finish_month').'-'.$request->input('education_finish_day'),
                ]);
            }
        }
        if(!empty($request->input('training_name')) && intval($request->input('training_start_day')) > 0 && intval($request->input('training_start_month')) > 0 && intval($request->input('training_start_year')) > 0 && intval($request->input('training_finish_day')) > 0 && intval($request->input('training_finish_month')) > 0 && intval($request->input('training_finish_year')) > 0 && !empty($request->input('training_place')))
        {
            if(Training::where('user_id', '=', $id)->exists())
            {
                Training::where('user_id', '=', $id)->update([
                    'training_name' => $request->input('training_name'),
```

```
                    'training_start' => $request->input('training_start_year').'-'.$request->input('training_start_month').'-'.$request->input('training_start_day'),
                    'training_finish' => $request->input('training_finish_year').'-'.$request->input('training_finish_month').'-'.$request->input('training_finish_day'),
                    'training_place' => $request->input('training_place')
                ]);
            }
            else
            {
                Training::create([
                    'user_id' => $id,
                    'training_name' => $request->input('training_name'),
                    'training_start' => $request->input('training_start_year').'-'.$request->input('training_start_month').'-'.$request->input('training_start_day'),
                    'training_finish' => $request->input('training_finish_year').'-'.$request->input('training_finish_month').'-'.$request->input('training_finish_day'),
                    'training_place' => $request->input('training_place')
                ]);
            }
        }
        if(!empty($request->input('hobby_name')))
        {
            if(Hobby::where('user_id', '=', $id)->exists())
            {
                Hobby::where('user_id', '=', $id)->update([
```

```
                    'hobby_name' => $request->input('hobby_name'),
                    'hobby_description' => $request->input('hobby_description')
                ]);
            }
            else
            {
                Hobby::create([
                    'user_id' => $id,
                    'hobby_name' => $request->input('hobby_name'),
                    'hobby_description' => $request->input('hobby_description')
                ]);
            }
        }
        if(Auth::user()->hasPermission('edit_permissions'))
        {
            $user = User::where('id', '=', $id)->firstOrFail();
            DB::table('user_has_permissions')->where('user_id', $user->id)->delete();
            if(count($request->input('permissions')) > 0)
            {
                foreach($request->input('permissions') as $permission)
                {
                    try{ $user->givePermissionTo($permission); }
                    catch (Exception $e)
                    {
                        if($e instanceof PermissionDoesNotExist)
                        {
```

```
                                        Permission::create(['name' =>
        $permission]);
                                        $user->givePermissionTo($permission);
                                    }
                                }
                            }
                        }
        }
        return back()->withErrors(['operation' => 'success']);
}
```

ეს ფუნქცია ეშვება იმათთვის, ვისზეც გაწერილია ბაზაში შესაბამისი უფლება და მოვალეობა. წინააღმდეგ შემთხვევაში გამოდის შესაბამისი შეტყობინება.

ზემოთ ნახსენები ლოგის რეპორტის ამოღებისთვის და შემდგომ ვიზუალიზაციისთვის გამოყენებისთვის მაქვს შექმნილი შემდეგი ფუნქცია:

```
public function getListLogs()
{
    return Datatables::collection(Log::select('logs.id', 'service_center', 'name', 'surname', 'action_date')
        ->join('users', 'users.id', '=', 'logs.user_id')
        ->join('jobs', 'users.id', '=', 'jobs.user_id')
        ->where('user_status', '=', 1)
        ->where('action', '=', 'login')
        ->where('action_value', '=', 'first')
        ->groupBy('logs.id')
        ->get())->make(true);
}
```

კოდი 20. getListLogs ფუნქცია.

ხოლო დეტალური ლოგის სრული სახის მიღებისთვის მაქვს შემდეგი ფუნქცია:

```php
public function postExportLogs(ExportLogsRequest $request)
{
    if(!Auth::user()->hasPermission('view_logs')){ return response(view('errors.403'), 403); }
    DB::disableQueryLog();
    $timeLimit = ini_get('max_execution_time');
    $memoryLimit = ini_get('memory_limit');
    ini_set('max_execution_time', 256000);
    ini_set('memory_limit','256M');
    $from = Tools::validateDate($request->input('from')) ? date('Y-m-d 00:00:00', strtotime($request->input('from'))) : '';
    $to = Tools::validateDate($request->input('to')) ? date('Y-m-d 23:59:59', strtotime($request->input('to'))) : '';
    $user_id = $request->input('user_id');
    Excel::create('Logs_'.$user_id.'_'.date('Y-m-d'), function($excel) use($from, $to, $user_id)
    {
        $excel->sheet('Logs', function($sheet) use($from, $to, $user_id)
        {
            $logs = Log::select(
                "logs.id AS ID",
                "jobs.service_center AS ფილიალი",
                "users.name AS სახელი",
                "users.surname AS გვარი",
                "logs.action_date AS მოსვლა",
                DB::raw("GREATEST(TIMESTAMPDIFF(MINUTE, DATE_FORMAT(logs.action_date, '%Y-%m-%d 10:00:00'), logs.action_date), 0) AS
                    'დაგვიანებული წუთები'"))
                ->join('users', 'users.id', '=', 'logs.user_id')
                ->join('jobs', 'users.id', '=', 'jobs.user_id')
                ->where('user_status', '=', 1)
                ->where('logs.action', '=', 'login')
                ->where('logs.action_value', '=', 'first');
            if(Tools::validateTimestamp($from)){ $logs->where('logs.action_date', '>=', $from); }
            if(Tools::validateTimestamp($to)){ $logs->where('logs.action_date', '<=', $to); }
            if($user_id != 'All' && $user_id > 0){ $logs->where('logs.user_id', '=', $user_id); }
            $logs->groupBy('logs.id');
            $sheet->fromModel($logs->get());
        });
    })->download('xls');
    DB::enableQueryLog();
    ini_set('max_execution_time', $timeLimit);
    ini_set('memory_limit',$memoryLimit);
}
```

კოდი 21. postExportLogs ფუნქცია.

რომელზე წვდომაც აქვთ მხოლოდ შესაბამისი უფლების მქონე მომხმარებლებს, რადგან იგი შეიცავს სრულ დეტალურ ინფორმაციას მომხმარებლებზე მათი აქტიურობის შესახებ კონკრეტულ პერიოდსა და დროზე.

მომხმარებლების რეგისტრაციასა და რედაქტირებაზე მიდევს შეზღუდვები ველებზე, რათა თავი ავირიდო არავალიდური და არასწორი ინფორმაციის შეყვანისგან. ამისთვის მაქვს შექმნილი შემდეგი ფუნქცია:

```php
public function rules()
{
    return [
        'username' => 'required|min:2|max:255',
        'AD_email' => 'required|email|max:255|unique:users',
        'name' => 'required|min:2|max:255',
        'surname' => 'required|min:2|max:255',
        'gender' => 'required|in:male,female',
        'day' => 'required|in:00,01,02,03,04,05,06,07,08,09,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31',
        'month' => 'required|in:00,01,02,03,04,05,06,07,08,09,10,11,12',
        'year' => 'required',
        'user_status' => 'required|in:0,1',
        'work_email' => 'required|email|max:255',
        'personal_email' => 'sometimes|nullable|email|max:255',
        'home_phone' => '',
        'work_mobile' => '',
        'personal_mobile' => '',
        'inner_number' => '',
        'position' => 'required|min:2|max:255',
        'department' => 'required|min:2|max:255',
        'service_center' => 'required|min:2|max:255',
        'job_start_day' => 'in:00,01,02,03,04,05,06,07,08,09,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31',
        'job_start_month' => 'in:00,01,02,03,04,05,06,07,08,09,10,11,12',
        'job_start_year' => '',
        'manager_id' => '',
        'university_name' => '',
        'degree' => 'sometimes|nullable|in:bachelor,master,phd',
        'education_start_day' => 'in:00,01,02,03,04,05,06,07,08,09,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31',
        'education_start_month' => 'in:00,01,02,03,04,05,06,07,08,09,10,11,12',
        'education_start_year' => '',
        'education_finish_day' => 'in:00,01,02,03,04,05,06,07,08,09,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31',
        'education_finish_month' => 'in:00,01,02,03,04,05,06,07,08,09,10,11,12',
        'education_finish_year' => '',
        'training_name' => '',
        'training_start_day' => 'in:00,01,02,03,04,05,06,07,08,09,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31',
        'training_start_month' => 'in:00,01,02,03,04,05,06,07,08,09,10,11,12',
        'training_start_year' => '',
        'training_finish_day' => 'in:00,01,02,03,04,05,06,07,08,09,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31',
        'training_finish_month' => 'in:00,01,02,03,04,05,06,07,08,09,10,11,12',
        'training_finish_year' => '',
        'training_place' => '',
        'hobby_name' => '',
        'hobby_description' => ''
    ];
}
```

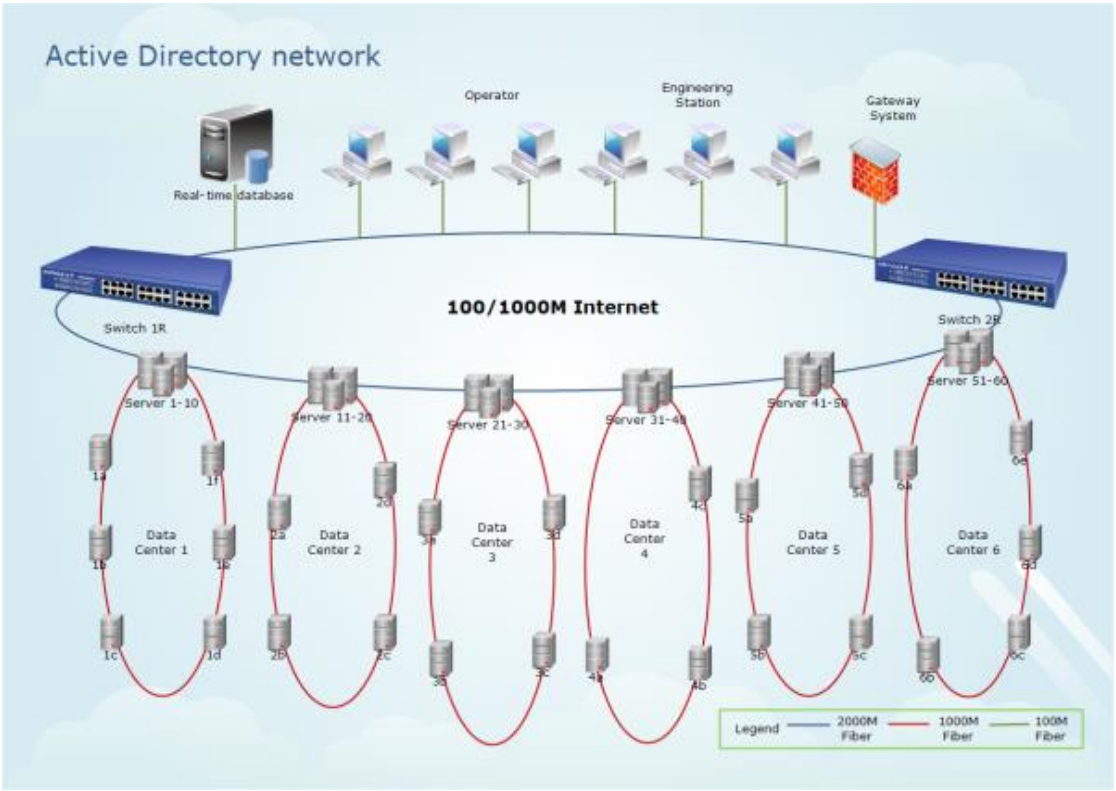
კოდი 22. rules ფუნქცია.

ჩათის შემოწმებისა და წამოწყების ვალიდურობისთვის მაქვს შემდეგი ორი ფუნქცია დამატებული:

```php
public static function checkConversation($user_id)
{
    return Conversation::whereIn('first_user_id', [Auth::user()->id, $user_id])
        ->whereIn('second_user_id',  [Auth::user()->id, $user_id])->exists();
}

public static function startConversation($user_id)
{
    if(self::checkConversation($user_id))
    {
        $conversation = Conversation::whereIn('first_user_id', [Auth::user()->id, $user_id])
            ->whereIn('second_user_id', [Auth::user()->id, $user_id])->firstOrFail();
        if($conversation->is_accepted == 0){ Chat::acceptMessageRequest($conversation->id); }
        return $conversation->id;
    }
    else
    {
        Chat::startConversationWith($user_id);
        $conversation = Conversation::whereIn('first_user_id', [Auth::user()->id, $user_id])
            ->whereIn('second_user_id', [Auth::user()->id, $user_id])->firstOrFail();
        Chat::acceptMessageRequest($conversation->id);
        return $conversation->id;
    }
}
```

კოდი 23. Conversation ფუნქციები.

ვირტუალურად რომ წარმოვიდგინოთ, გვაქვს ასეთი ნახაზი:



სურათ 52. Active Directoy ქსელი.

ასევე დამატებული მაქვს ჩათის შეტყობინების ფუნქციონალი, რომელიც გადავჭერი შემდეგნაირად:

```php
public static function sendMessage($type, $conversation_id, $message)
{
    if($type == 'user')
    {
        Chat::sendConversationMessage($conversation_id, $message);
        $conversation = Conversation::findOrFail($conversation_id);
        $receiver_id = ($conversation->first_user_id != Auth::user()->id ?
         $conversation->first_user_id : $conversation->second_user_id);
        MessageExtension::create([
            'type' => 'user',
            'conversation_id' => $conversation_id,
            'sender_id' => Auth::user()->id,
            'receiver_id' => $receiver_id,
            'data' => 'text',
            'seen' => 0
        ]);
        broadcast(new MessageNotification('user', Auth::user()
            ->id, 'message-notifications-'.$receiver_id));
        return true;
    }
    elseif($type == 'group')
    {
        Chat::sendGroupConversationMessage($conversation_id, $message);
        $conversation = GroupConversation::findOrFail($conversation_id);
        foreach($conversation->users()->get() as $groupUser)
        {
            if($groupUser->id != Auth::user()->id)
            {
                MessageExtension::create([
                    'type' => 'group',
                    'conversation_id' => $conversation->id,
                    'sender_id' => $conversation->id,
                    'receiver_id' => $groupUser->id,
                    'data' => 'text',
                    'seen' => 0
                ]);
                broadcast(new MessageNotification('group', $conversation->id,
                 'message-notifications-'.$groupUser->id));
            }
        }
        return true;
    }
    else{ return false; }
}
```

კოდი 24. sendMessage ფუნქცია.

ჩათში შეტყობინების გაგზავნისას თავდაპირველად არის სტატუსი "უნახავი", ხოლო ადრესატის მიერ შეტყობინების ნახვისას სტატუსი გარდაიქმნება "ნანახია" მნიშვნელობაზე, რასაც აკეთებს შემდეგი ფუნქცია:

```php
public static function seen($type, $id)
{
    if(!in_array($type, ['user', 'group'])){ return false; }
    MessageExtension::where('type', '=', $type)
        ->where('sender_id', '=', $id)
        ->where('receiver_id', '=', Auth::user()->id)
        ->update(['seen' => 1]);
    return true;
}
```

კოდი 25. seen ფუნქცია.

მთავარი კლასი, რომელიც ახდენს მომხმარებლების აქტივობებისა და მოვლენების დამახსოვრებას, ნაჩვენები მაქვს კოდის შემდეგ ფრაგმენტში:

Logger.php

```php
<?php

namespace App\Library;

use App\Log;
use Illuminate\Support\Facades\Auth;

class Logger
{
    public static function log($action, $actionValue = null)
    {
        Log::create([
            'user_id' => Auth::user()->id,
            'action' => $action,
            'action_value' => $actionValue,
            'action_date' => date('Y-m-d H:i:s')
        ]);
        return true;
    }

    public static function loginLog()
    {
        if(!Log::where('user_id', '=', Auth::user()->id)->where('action', '=', 'login')
            ->whereRaw('YEAR(action_date) = YEAR(NOW())')->whereRaw('MONTH(action_date) =
             MONTH(NOW())')->whereRaw('DAY(action_date) = DAY(NOW())')->exists())
        {
            Logger::log('login', 'first');
        }
        else{ Logger::log('login'); }
        return true;
    }
}
```

კოდი 26. Logger კლასი.

ბაზასთან დამაკავშირებელი და ინფორმაციის წამომღები მთავარი ფუნქცია გამოიყურება შემდეგნაირად:

```php
public static function getRandomDOB($type = 'array')
{
    $users = User::join('jobs', 'users.id', '=', 'jobs.user_id')
        ->active()
        ->select('AD_email', 'name', 'surname', 'gender', 'picture',
         'position', 'DOB', DB::raw('YEAR(NOW()) - YEAR(DOB) AS age'))
        ->where('job_status', '=', 'current')
        ->whereRaw('MONTH(DOB) = MONTH(NOW())')
        ->whereRaw('DAYOFMONTH(DOB) = DAYOFMONTH(NOW())')
        ->inRandomOrder()
        ->get();
    if($type == 'array'){ return $users; }
    elseif($type == 'json'){ return $users->toJson(); }
    else{ return false; }
}
```

კოდი 27. getRandomDOB ფუნქცია.

კონტაქტის მოდელი შედგება შემდეგი კომპონენტებისგან:

Contact.php

```php
<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class Contact extends Model
{
    protected $table = 'contacts';

    protected $fillable = ['user_id', 'work_email', 'personal_email',
     'home_phone', 'work_mobile', 'personal_mobile', 'inner_number'];

    public function user()
    {
        return $this->belongsTo('App\User', 'user_id');
    }
}
```

კოდი 28. Contact კლასი.

ჩათს ვამუშავებ როგორც ცალკე პროექტს - მისთვის იქმნება არხი, რომელზეც ხდება სხვადასხვა ოპერაციები მისამართების მიმოცვლის გზით, რაც ნაჩვენები მაქვს შემდეგ ფრაგმენტში:

channels.php

```php
<?php

/*
|--------------------------------------------------------------------------
| Broadcast Channels
|--------------------------------------------------------------------------
|
| Here you may register all of the event broadcasting channels that your
| application supports. The given channel authorization callbacks are
| used to check if an authenticated user can listen to the channel.
|
*/

Broadcast::channel('App.User.{id}', function ($user, $id) {
    return (int) $user->id === (int) $id;
});

Broadcast::channel('chat-room-*', function ($user, $id){
    return $user;
});

Broadcast::channel('chat', function ($user){
    return $user;
});

Broadcast::channel('message-notifications-*', function ($user){
    return $user;
});

Broadcast::channel('group-change-notifications-*', function ($user){
    return $user;
});

Broadcast::channel('room-reservations', function(){ return true; });

Broadcast::channel('driver-reservations', function(){ return true; });
```

კოდი 29. channels კლასი.

უფლებების ბაზის მიგრაციის კოდი შედგება შემდეგი ბიჯებისგან:

```
use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
class CreatePermissionTables extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        $tableNames = config('laravel-permission.table_names');
        $foreignKeys = config('laravel-permission.foreign_keys');
        Schema::create($tableNames['roles'], function (Blueprint
$table) {
            $table->increments('id');
            $table->string('name')->unique();
            $table->timestamps();
        });
        Schema::create($tableNames['permissions'], function
(Blueprint $table) {
            $table->increments('id');
            $table->string('name')->unique();
            $table->timestamps();
        });
        Schema::create($tableNames['user_has_permissions'],
function (Blueprint $table) use ($tableNames, $foreignKeys) {
            $table->integer($foreignKeys['users'])->unsigned();
```

```
            $table->integer('permission_id')->unsigned();
            $table->foreign($foreignKeys['users'])
                ->references('id')
                ->on($tableNames['users'])
                ->onDelete('cascade');
            $table->foreign('permission_id')
                ->references('id')
                ->on($tableNames['permissions'])
                ->onDelete('cascade');
            $table->primary([$foreignKeys['users'],
'permission_id']);
        });
        Schema::create($tableNames['user_has_roles'], function
(Blueprint $table) use ($tableNames, $foreignKeys) {
            $table->integer('role_id')->unsigned();
            $table->integer($foreignKeys['users'])->unsigned();
            $table->foreign('role_id')
                ->references('id')
                ->on($tableNames['roles'])
                ->onDelete('cascade');
            $table->foreign($foreignKeys['users'])
                ->references('id')
                ->on($tableNames['users'])
                ->onDelete('cascade');
            $table->primary(['role_id', $foreignKeys['users']]);
        });
        Schema::create($tableNames['role_has_permissions'],
function (Blueprint $table) use ($tableNames) {
            $table->integer('permission_id')->unsigned();
```

```
                $table->integer('role_id')->unsigned();
                $table->foreign('permission_id')
                    ->references('id')
                    ->on($tableNames['permissions'])
                    ->onDelete('cascade');
                $table->foreign('role_id')
                    ->references('id')
                    ->on($tableNames['roles'])
                    ->onDelete('cascade');
                $table->primary(['permission_id', 'role_id']);
            });
    }
    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        $tableNames = config('laravel-permission.table_names');
        Schema::drop($tableNames['role_has_permissions']);
        Schema::drop($tableNames['user_has_roles']);
        Schema::drop($tableNames['user_has_permissions']);
        Schema::drop($tableNames['roles']);
        Schema::drop($tableNames['permissions']);
    }
}
```

მიგრაციას ვაკეთებთ იმისთვის, რომ თუ იქნება საჭირო ბაზის რომელიმე ცხრილის სტრუქტურის ან ფორმატის ცვლილება, რომ ან მოგვიწიოს მაგისთვის ბაზაში შესწორებების შეტანა - პირდაპირ კოდიდან შევუცვლით სახელს ან ფორმატს და ავტომატურად გადაიყვანს ყველაფერს განახლებულ სისტემაზე.

ეს მიდგომა თანამედროვე სისტემებში ძალიან დიდი პოპულარობით სარგებლობს, რადგან ყველაფერი უკვე პროგრამირებასთან იკვეთება.

თავი 2.5. Microsoft Active Directory ალტერნატივები

მომხმარებლებთან სამუშაოდ Microsoft Active Directory არ არის ერთადერთი საშუალება. გვხვდება სხვა პოპულარული პროგრამული უზრუნველყოფებიც, რომლებიც მეტნაკლებად ჰგავან მას როგორც ფუნქციონალურად, ასევე ინტერფეისულადაც. მათრიცხვს განეკუთვნება:

- Apache Directory;
- Open LDAP;
- Univention Corporation Server (UCS);
- Lepide Auditor for Active Directory;
- JXplorer;
- FreeIPA;
- Samba;
- GoSa;
- eDirectory;
- Zentyal;
- 389 Directory Server;
- Red Hat Directory Servers;
- OpenSSO;
- SME Server;
- Resara Server;
- Sun Java System Directory Server;

- IBM Tivoli Directory Server;
- Windows NT Directory Services;
- Lotus Domino;
- SolarWinds Permissions Analyzer;
- ManageEngine AdManager Plus;
- Specops Command.

თითოეული მათგანის ფუნქციას განეკუთვნება თქვენი უსაფრთხოება უცხო პირების შემოღწევისგან და წვდომების შეზღუდვა მომხმარებლებისთვის როგორც მათი მართვის, ასევე მენეჯმენტისთვის.

თავი 2.5.1. Apache Directory

Apache-ს მიერ შექმნილი open source პროგრამა Apache Directory გვთავაზობს გადაწყვეტას Java დაპროგრამების ენაზე, რომელსაც მოყვება LDAP v3 სერთიფიცირებული მომხმარებლური სერვერი. მან სერთიფიცირება გაიარა 2006 წელს Open Group-ის მიერ, აგრეთვე Rclipse პროგრამული უზრუნველყოფის მიერ. LDAP-ისგან განსხვავებით მას აქვს Kerberos სერვერული მხარდაჭერა.

Apache Directory Studio უტილიტას დაემატა სქემატური საძიებო სისტემა, DSML რედაქტორი, LDAP მაძიებელი და რედაქტორი, LDIF რედაქტორი და მრავალი სხვა. იგი არის განახლებული ვერსია.

თავი 2.5.2. Open LDAP

Open Ldap ანუ იგივე LDAP Админ არის უფასო open-source მსუბუქი პროგრამა LDAP ტიპის ბაზებთან სამუშაოდ (Lightweight Directory Access Protocol - მსუბუქი დირექტორიის წვდომის პროტოკოლი). იგი მუშაობს Microsoft LDAP კლიენტზე და ადმინისტრირებას უწევს მას. იგი არის იდეალური ალტერნატივა. მას აქვს ფუნქციები - ძიება, ძებნა, ცვლილება, შექმნა და წაშლა სერვერიდან.

მას აგრეთვე აქვს რამდენიმე დამატებითი ფუნქცია - პაროლების ცვლილება, მხარდაჭერის ფუნქცია, ექსპორტი და იმპორტი და სხვ.

თავი 2.5.3. Univention Corporation Server (UCS)

Univention Corporation Server არის სერვერული პროგრამული უზრუნველყოფა, რომელიც გამოიყენება IT ოპერაციებსა და სერვერული აპლიკაციების კონტროლისთვის. სერვერის ოპერაციული სისტემა ადაპტირებულია Debian GNU/Linux-ზე, რომელიც კომბინირებულია მართვის სისტემებთან სერვერის მულტიპლატფორმული კონტროლისთვის, კლიენტებისთვის, კომპიუტერებისთვის, მომხმარებლებისთვის, სერვისებსა და სერვერულ სხვადასხვა ტექნიკისთვის.

ამ პროგრამულ უზრუნველყოფასაც დაემატა Microsoft AD-ს ფუნქციების მხარდაჭერა მრავალი კომპანიისთვის, მათი ადმინისტრირებისთვის და კონტროლისთვის Samba პროგრამასთან ურთიერთშეთანხმებით (აგრეთვე open source პროგრამა).

2.5.4. Lepide Auditor for Active Directory

Active Directory-ს კონტროლი იცვლება ისეთი მძლავრი პროგრამით, როგორიცაა Lepide Auditor. ეს სპეციალურად მაგისთვის იყო შექმნილი, რომ შესაძლებელი იყოს დირექტორიების ცვლილებების მენეჯმენტი. მისი მეშვეობით მარტივია გარკვევა, თუ ვინ, სად, რა და როდის შეცვალა.

იგი ინახავს ყველაფრის ლოგს, რომლებიც ინახება და სორტირდება ერთ სივრცეში, რომელიც ვიზუალურად ქმნის კომფორტულ ფანჯარას. მას ასევე აქვს ფუნქცია, რომელიც გატყობინებთ კრიტიკული ცვლილებების შესახებ.

თავი 2.5.5. JXplorer

JXplorer არის უფასო open source მულტი პლატფორმული LDAP საძიებო პროგრამა და რედაქტორი, რომელიც შემუშავებულია eTrust Directory პროგრამისტების ლაბორატორიის მიერ Computer Associate მხარდაჭერით. მას აქვს ძიების, წაკითხვის და რედაქტირების ფუნქციები ნებისმიერი სტანდარტული LDAP დირექტორიებისთცის ან X500 დირექტორიებისთვის DSML ინტერფეისით. იგი არის ყოველმხრივ მოქნილი, რადგან შესაძლებელია მისი მორგება მრავალი გზით.

მისი კოდი დაწერილია Java-ზე და ძრავი ატვირთულია და ყველასათვის მისაწვდომია svn-ზე, საიდანაც შესაძლებელია მისი შემდგომი განვითარება. მის პაკეტს მოყვება ჩაშენებული რეპორტინგის სისტემა და უსაფრთხოების საშუალებები.

თავი 2.5.6. FreeIPA

FreeIPA არის Red Hat კომპანიის მიერ შემუშავებული უფასო open source პროექტი, რომელის მიზანია Linux და Unix კომპიუტერული ქსელების პოლისების იდენთიფიცირება და აუდიტი. მას აქვს Active Directory-სთან სამუშაოდ საჭირო მრავალი მახასიათებელი.

მას აქვს გაძლიერებული უსაფრთხოება და ინტერფეისის მრავალი ნაირსახეობა, როგორიცაა კონსოლური და WEB მხარე.

თავი 2.5.7. Samba

Samba არის open source პროგრამა, რომელიც Unix და Windows პლატფორმებზე. იგი აგრეთვე მუშაობდა სხვადასხვაარა Unix ტიპის ოპერაციულ სისტემებზე, როგორებიცაა NetWare, AmigaOS და VMS, რაც მეტყველებს მის მრავალფუნქციურობაზე და თავსებადობაზე.

თავი 2.5.8. GoSa

GoSa არის ერთ-ერთი საუკეთესო ალტერნატივა Active Directory პროგრამისთვის. მას შეუძლია მომხმარებლების ადმინისტრირება, LDAP ძიება, აპლიკაციების მენეჯმენტი, საფოსტო გაზიარებები, ჯგუფებთან მუშაობა. იგი ერგება როგორც დიდ, ასევე მცირე კლიენტებს, ტელეფონიებსა და ფაქსებს.

იგი აგრეთვე გამოირჩევა მარტივი ინსტალაციის პრინციპით. მისი მეშვეობით შესაძლებელია როგორც დიდი და გლობალური პარამეტრების ცვლილება, აგრეთვე უმნიშვნელო ცვლილებების შეტანაც მარტივი გზით, რაც აძლევს მომხმარებლებსა და სისტემურ ადმინისტრატორს შესაბამისი მენეჯმენტის მიღების საშუალებას.

პროგრამა პოპულარულია საფრანგეთში, ესპანეთში, გერმანიაში, ბელგიაში და სხვა ამერიკისა და ევროპის ქვეყნებში. მას აქვს მრავალი ენის მხარდაჭერა, რაც უფრო მისაღებს ხდის სხვადასხვა ხალხისთვის.

თავი 2.5.9. eDirectory

Novell კომპანიის პროგრამა eDirectory არის თანამედროვე ალტერნატივა Microsoft პროდუქტისთვის. იგი თავიდან შეიქმნა კომპანიის ქსელური ოპერაციული სისტემის დირექტორიისთვის. იგი თავიდანვე გამოიკვეთა თავისი მძლავრი ფუნქციებით, წარმადობით და სერვისებით.

ახლა პროგრამას იყენებს 28 000 მომხმარებელი და 1000 ორგანიზაცია. მას ძირითადად მოიხმარენ დიდი ქსელებისთვის, რადგან მასთან მუშაობისას არანაირი პრობლემა არ წარმოიქმნება. მისი უსაფრთხოება ყველა სტანდარტს შეესაბამება და მუშაობს ნებისმიერ დირექტორიასთან - SASL, DSML, LDap, Soap.

2.5.10. Zentyal

Zentyal გვთავაზობს დირექტორიულ სერვერს, რომელიც თავსებადია MS AD სერვისებთან. მისი მეშვეობით მარტივია ქსელური ინფრასტრუქტურის მართვა მიუხედავად ოფისების რაოდენობისა და ადგილმდებარეობისა. მას აქვს ცენტრალიზებული მართვის სისტემა, გამარტივებული ავტორიზაციის სქემა, მულტი ორგანიზაციული სერვისები, პრინტერების მენეჯმენტი, კონტაქტები, მომხმარებლები, სიები, ჯგუფები, ანტივირუსული სისტემები, კარანტინის ფაილური სერვერი და მრავალი სხვა.

თავი 2.5.11. 389 Directory Server

389 Directory Server არის ბიზნეს კლასის open source უფასო LDAP სერვერი Linux პლატფორმისთვის, რომელიც ხდის მას ფასეულ ალტერნატივად MS პროდუქტისთვის. იგი მუშაობს მსოფლიოს უდიდეს LDAP დისტრიბუციასთან და რაც მთავარია, უპირატესია რომ ხელმისაწვდომია გადმოწერისთვის უფასოდ და მისი გამართვა შესაძლებელია ერთი საათის განმავლობაში მარტივი გრაფიკული ინტერფეისის მეშვეობით.

მისი ძლიერი მხარეა მომხმარებლებისა და მოთხოვნების დამუშავების სისწრაფე. მას შეუძლია უთვალავი ოპერაციის შესრულება წამში და ათობით ათასი პარალელური მომხმარებლის მომსახურება. მისი ერთადერთი შეზღუდვა არის ადგილის დიდი რაოდენობით საჭიროება, მაგრამ იგი ერგება აბსოლუტურად ყველა კომპანიას.

თავი 2.5.12. Red Hat Directory Servers

Red Hat Directory Server პროგრამით შესაძლებელია Unix გარემოში მომხმარებლების წვდომების მენეჯმენტი მრავალ სისტემაზე. იგი გვაძლევს საშუალებას, რომ შევინახოთ მომხმარებლების ინფორმაცია LDAP

იერარქიაზე დაფუძნებულ ბაზაში, რაც ხელმისაწვდომი იქნება დომეინში მყოფი ყველა მომხმარებლისთვის.

მისი მრავალფუნქციურობა გამოირჩევა ინფორმაციის შენახვის მაღალი უსაფრთხოებით, მასზე წვდომის შეზღუდვით, მომხმარებლების იდენთიფიცირებით, როლების ამოცნობით, დომეინური სახელების ამოკითხვით, ჯგუფის წევრების გარჩევით და სხვა.

მისი ვალიდაცია ეყრდნობა სერთიფიკატებს - წვდომა იხსნება დისტანციურადაც, ოღონდ ვალიდაციური პროცედურების გავლის შემდეგ.

თავი 2.5.13. OpenSSO

Open SSO (Single Sign-On) არის open source მულტი პლატფორმული წვდომების კონტროლის სერვერი. იგი ასევე უზრუნველყოფს სერვისების უსაფრთხოებას ავტონომიურ აპლიკაციებში.

მას აქვს ინტეგრირებული წვდომის მენეჯმენტი ბიზნესისთვის, დავალებებისა და სიტუაციების შესრულების სწრაფი ჩანაცვლება იმისთვის, რომ ყველა გარემო მოექცეს ერთი ინტერფეისის ქვეშ. იგი უზრუნველყოფს უსაფრთხოების უმაღლეს პარამეტრებს. პროგრამის დამფუძნებულია ორგანიზაცია Oracle.

თავი 2.5.14. SME Server

SME Server არის koozali ფირმის მიერ შექმნილი open source, უსაფრთხო,სოლიდური, ცირკულირებს Linux სერვერზე და განკუთვნილია მცირე და საშუალო ბიზნესებისთვის. იგი აწყობილია CentOS/Redhat კოდზე, რომელზეც მუშაობს აქტიური და ნიჭიერი გუნდი მრავალი პროგრამული უზრუნველყოფით 2007 წლიდან.

იგი არის უფასო პროგრამა, რომელიც გვთავაზობს რამდენიმე უნიკალურ ფუნქციას, რომელიც ქმნის მას უსაფრთხო, სტაბილურ და ხელმისაწვდომ მულტი პლატფორმულ განვრცობილ შემოთავაზებამდე მომავალი მოთხოვნების გათვალისწინებით.

მათი პროგრამა არის სანდო და ადვილად ოპერირებადი. მისი დაყენება შესაძლებელია სულ რაღაც 20 წუთში. ყველაზე ფასეული კი არის ოპერაციული სისტემა Linux, რაზეც მუშაობს იგი, რადგან სულ განახლებებში და უსაფრთხოების მაღალი დონის დახასიათებაში გამოსარჩევი იქნება.

თავი 2.5.15. Resara Server

Resara Server არის open source ტიპის პროგრამა განკუთვნილი მცირე და საშუალო ზომის ბიზნესებისთვის, რომელიც თავსებადია AD-სთან. იგი არის სრულიად უფასო და ასწყობილია Samba-ზე ინტეგრირებით. მისი მიზანია წინდახედულობა და ოპერაციების სისწრაფე. მისი სამართავი პანელით შესაძლებელია მომხმარებლების კონტროლი, ფაილების გაზიარება, DHCP და DNS გამართვა.

მას აქვს კომფორტული მომხმარებლური ინტერფეისი, რომელსაც მოყვება სრული დოკუმენტაცია. მას მოყვება მომხმარებლების მართვის ფუნქცია, ფაილების გაზიარება, Active Directory დომეინური კონტროლი, დისკების ავტომატური მეპინგი Windows მომხმარებლებისთვის და ქსელის ჰოსტების მენეჯმენტი.

თავი 2.5.16. Sun Java System Directory Server

ცნობილი როგორც Oracle Directory Server Enterprise Edition, პროგრამა Java System Directory Server იყო Sun მიკროსისტემის LDAP ბაზა და DSML სერვერი დაწერილი C-ზე. ორიგინალი კოდი ხელმისაწვდომია უფასო გადმოწერისთვის და კომერციული მოხმარებისთვის სამუდამოდ, სრული სერვისით და გამოკვლევებით ინდივიდუალური საჭიროებებისთვის, რომელიც დევს Oracle-ს საიტზე, რომელიც გახდა მათი ოფიციალური ახალი პროდუქტი ამჟამად.

გადმოსაწერად ატვირთულია მხოლოდ მისი ბოლო ვერსია. ვინაიდან პროგრამას ჰქონდა უამრავი bug C ენასთან უთანხმოების გამო, იგი

გადაიტანეს OpenDS პლატფორმაზე 2011 წელს. მას შემდეგ მისი კოდი არ განახლებულა.

თავი 2.5.17. IBM Tivoli Directory Server

Tivoli Directory Server არის IBM კორპორაციის ვერსია LDAP ბაზებთან სამუშაოდ. იგი არის მსუბუქი და მძლავრი. იგი მუშაობს როგორც ინტრანეტებთან, ასევე ინტერნეტშიც. იგი მოიცავს ყველა საჭიროებას - უსაფრთხოებას, მენეჯმენტს, კონტროლს.

სერვერი შედგება ვალიდაციის რამდენიმე მეთოდისგან, სტანდარტული სახელი/პაროლის ჩათვლით. მას აგრეთვე შეუძლია ციფრული სერთიფიკატით იდენთიფიცირება, შიფრაცია. მას ძირითადად იყენებენ განვითარებადი კომპანიები, რომლებიც ორიენტირებულები არიან შიდა პროგრესზე, რადგან პროგრამას აქვს იდენთიფიცირების სრული სისტემა.

თავი 2.5.18. Windows NT Directory Services

NTDS ანუ Windows NT Directory Services არის დირექტორიების სერვისების უტილიტები, რომლებიც ეძებენ ქსელში რესურსებს, აკონტროლებენ მათ და ახარისხებენ. იგი გადის დომეინში, ინტეგრირდება დირექტორიის სერვერის ლეველთან და გადის მარტივ სტანდარტულ ავტორიზაციას.

მისი ფუნქციებია ცენტრალური ადმინისტრირება და რესურსების წვდომების უფლებები. მას შეუძლია მრავალ მომხმარებელთან მუშაობა ეფექტურად და ვალიდურად, WAN ლინკების მეშვეობით.

თავი 2.5.19. Lotus Domino

ცნობილი როგორც Lotus Domino, IBM Domino არის კლიენტური სერვერის კოლექციის პროგრამული პლატფორმა. მას იყენებდნენ მესიჯების ავტომატიზაციისთვის. ახლა იგი არის სერვერული აპლიკაციის

პლატფორმა მესიჯინგისთვის, ფოსტისთვის, კოლაბორაციისთვის და განრიგისთვის.

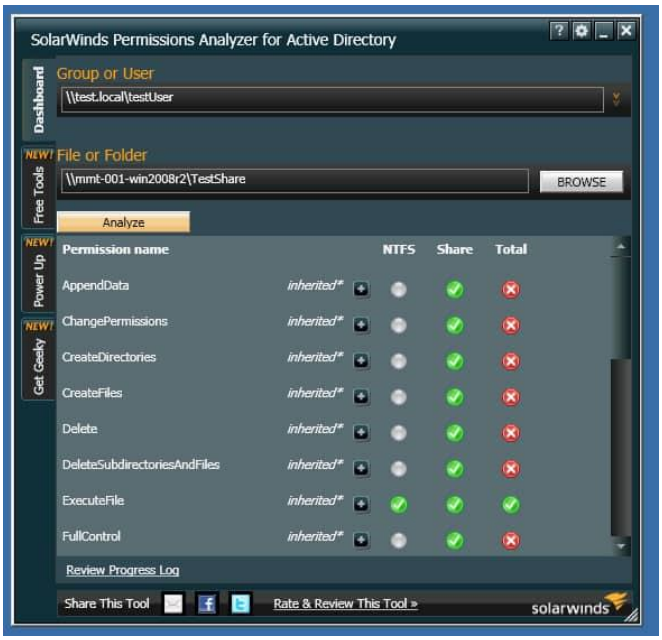
მას აგრეთვე აქვს კალენდრის ფუნქცია, გასაკეთებელი სიების ფუნქცია, ფაილების გაზიარება, ბლოგი, დისკუსიების ფორუმები, მიკრო ბლოგინგი, მომხმარებელთა ლექსიკონები და სხვ.

იგი არის კომპიუტერული სისტემის აპლიკაცია, რომელიც ძირითადად გამოიყენება კორპორაციული მიზნებისთვის, მეილებისთვის, აგრეთვე დირექტორიების წვდომებისთვის, დოკუმენტების შენახვისთვის და პირადი მიზნებისთვის.

თავი 2.5.20. SolarWinds Permissions Analyzer

SolarWinds Permissions Analyzer არის პროგრამა AD-სთან სამუშაოდ, რომელიც უწევს დიდ კონკურენციას Microsoft-ს, რადგან მასთან შედარებით მუშაობს ბევრად სწრაფად და აქვს ავტომატიზაციის შესაძლებლობა. მისი ნაკლოვანებაა - მცირე უფლებების მენეჯმენტი მომხმარებლებისთვის. იგი ძებნის ქსელში მყოფ მომხმარებლებს და ნახულობს თუ ვის რა მონაცემებზე აქვს წვდომა, რაც იძლევა იდენთიფიცირების მარტივ საშუალებას და გარემოს - რომელო წევრი რომელ ჯგუფშია და რა პრივილეგიები აქვს კონკრეტულ მონაცემებზე.

მათი ნახვა შესაძლებელია როგორც ინდივიდუალურად, აგრეთვე ჯგუფურად. აგრეთვე შესაძლებელია ნახვა თუ საიდან აქვთ მომხმარებლებს მინიჭებული უფლებები.



სურათი 53. AD ალტერნატივა: SolarWinds Permissions Analyzer.

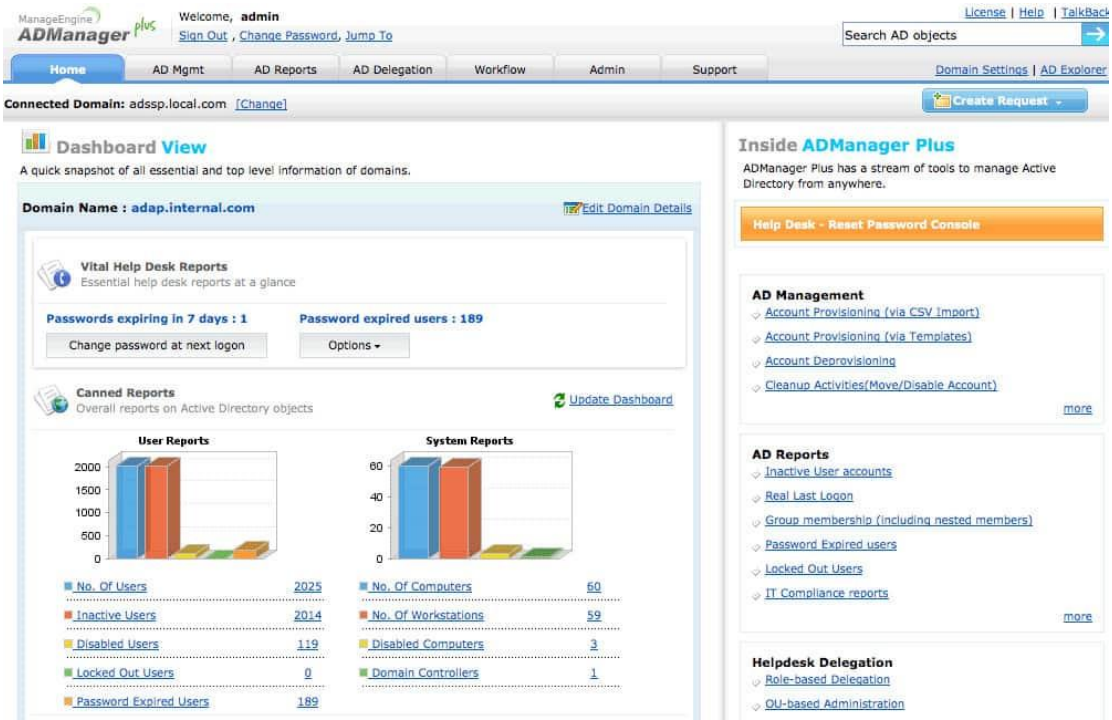
მთავარი უპირატესობა კი არის ის, რომ პროგრამ არის სრულიად უფასო და ხელმისაწვდომი გადმოწერისთვის. მისი მეშვეობით შესაძლებელია სწრაფი მონიტორინგი ჩვენი ქსელის ობიექტების მდგომარეობის.

თავი 2.5.21. ManageEngine AdManager Plus

AdManager Plus არის პროგრამა AD-სთან სამუშაოდ და მისი რეპორტების დაგენერირებისთვის. მისი მეშვეობით შესაძლებელია დომეინური ობიექტების ნახვა, ჯგუფების, მომხმარებლების და ეს ყველაფერი ერთი გრაფიკული ინტერფეისიდან.

რაც შეეხება რეპორტებს, იგი იძლევა რეპორტების გენერაციის ავტომატიზაციის საშუალებას, რაც იმას ნიშნავს, რომ რეპორტების მიღებისთვის არ იქნება საჭირო მანუალური ჩარევა, რაც კიდევ უფრო მეტად აადვილებს მუშაობის პროცესს.

პროგრამა რეკომენდებულია მათთვის, ვისაც უნდა რეპორტების მიღება AD-დან. პროგრამა არის სრულიად უფასო და ხელმისაწვდომი გადმოწერისთვის.



სურათი 54. AD ალტერნატივა: AdManager Plus.

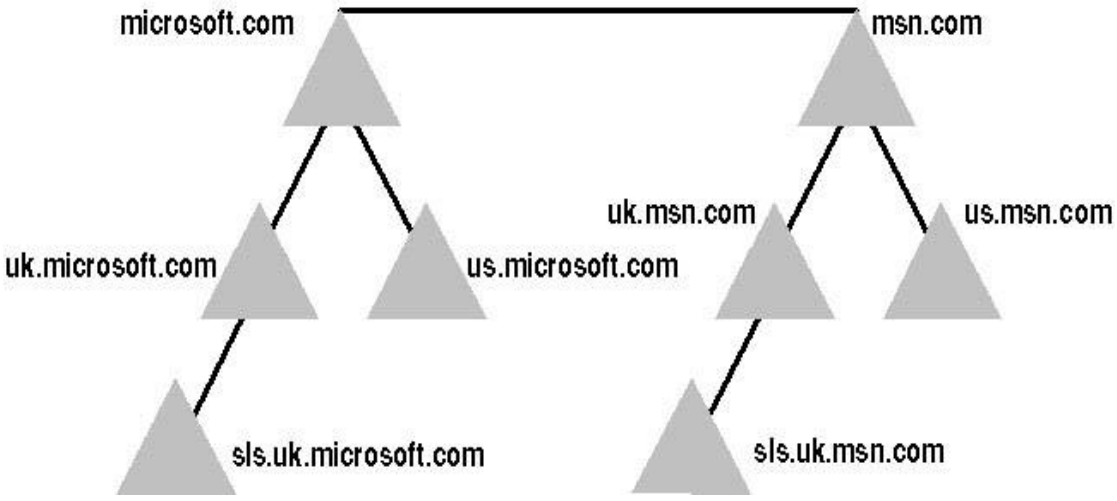
თავი 2.5.22. Specops Command

Specops Command პროგრამა გამოიყენება AD-ს მენეჯმენტისთვის. მისი სკრიპტების მეშვეობით შესაძლებელია ქსელში მდებარე ობიექტების მდგომარეობის მართვა. მას აქვს PowerShell და VB კონსოლური სკრიპტების მხარდაჭერა, რომლების მეშვეობითაც ვაკონტროლებთ ქსელურ მოწყობილობებს. ბრძანებების გაშვება აგრეთვე შესაძლებელია მომხმარებლების მხრიდანაც.

პროგრამას შეუძლია არა მხოლოდ საკუთარი სკრიპტების გაშვება, არამედ უკვე მზა სკრიპტების იმპორტი და გაშვება ფაილებიდან. ამ პროცესის შესრულება შესაძლებელია ავტომატურადაც, რაღაც ფიქსირებულ დროზე დაყენებით, როდესაც სკრიპტი ავტომატურად გაეშვება, რაც ავტომატიზაციის პირველი საფეხურია.

ერთ-ერთი მთავარი მახასიათებელი კი არის რეპორტინგის მხარდაჭერა, ვებ ბრაუზერული ტიპის, რომელსაც აქვს გამოხმაურების დატოვების შესაძლებლობა და მათი შემდგომი ანალიზი. პროგრამა არის აბსოლუტურად უფასო და ხელმისაწვდომი გადმოწერისთვის.

მაგალითად, თვითონ Microsoft-ს თავისი კომპანიის გარე AD ხე აქვს წარმოდგენილი შემდეგნაირად:



სურათი 55. Microsoft ხე.

სადაც სათავოში არის Microsoft-ის საიტი და მერე იშლება შიდა სტრუქტურებად, როგორიცაა დამატებითი გვერდები სხვადასხვა ქვეყნებისთვის:

- uk.microsoft.com
- us.microsoft.com
- sls.uk.microsoft.com
- msn.com
- uk.msn.com
- us.msn.com
- sks.msn.com

დასკვნა

Microsoft Windows Active Diretory არის მძლავრი იარაღი მომხმარებლების სამართავად, მათი უფლებების გასაწერად და ამ ყველაფრის გასაზიარებლად სხვადასხვა სისტემებში - ინტრანეტში, საფოსტო ყუთში, კალენდარში, კომუნიკატორსა და Microsoft-ისა თუ პირად პროგრამებში და აგრეთვე საბრაუზერო აპლიკაციებში. იგი საშუალებას გვაძლევს მონაცემების შესავსებად და დასაყოფად. იმახსოვრებს როგორც ჩაშენებულ ველებს, ასევე შეუძლია პერსონალური ველების შექმნა და შევსება. მისი მეშვეობით შესაძლებელია ერთი ბაზის შევსება და მრავალ აპლიკაციაში გამოყენება, რადგან მისი ინტეგრირების მხარდაჭერა თითქმის ყველა თანამედროვე პროგრამირების ენას აქვს.

გამოყენებული ლიტერატურა

1. Hynes, Byron (November 2006). "The Future Of Windows: Directory Services in Windows Server "Longhorn".
2. "The LDAP Application Program Interface". Retrieved 2013-11-26.
3. "An Approach for Using LDAP as a Network Information Service". Retrieved 2013-11-26.
4. "LDAP Password Modify Extended Operation". Retrieved 2013-11-26.
5. "The Lightweight Directory Access Protocol (LDAP) Content Synchronization Operation". Retrieved 2013-11-26.
6. Andreas Luther. "Active Directory Replication Traffic". Microsoft Corporation. Retrieved 26 May 2010.
7. "Using Scripts to Search Active Directory". Microsoft. Retrieved 22 May 2012.
8. Carter, G. LDAP System Administration. O'Reilly Media. ISBN 978-1-56592-491-8.
9. Donley, C. LDAP Programming, Management, and Integration. Manning Publications. ISBN 978-1-930110-40-3.
10. "LDAP - Lightweight Directory Access Protocol". Webopedia.com. Retrieved 2014-04-05.
11. სტუ-ს სტუდენტთა 86-ე ღია საერთაშორისო სამეცნიერო კონფერენცია, თეზისების კრებული. თემის დასახელება: "ხელოვნური ნეირონული ქსელების ანალიზი", გვ. 248. ავტორები: დ. სირბილაძე, თ. ბახტაძე.
12. საქართველოს ტექნიკური უნივერსიტეტის არჩილ ელიაშვილის მართვის სისტემების ინსტიტუტი, შრომათაკრებული N21, 2017. სტატიის დასახელება: "ხელოვნური ნეირონული ქსელების (ინტელექტის) ანალიზი", გვ. 168-172. ავტორები: დ. სირბილაძე, თ. ბახტაძე.
13. საქართველოს ტექნიკური უნივერსიტეტის არჩილ ელიაშვილის სახელობის მართვის სისტემების ინსტიტუტი, შრომათა კრებული N22, 2018. სტატიის დასახელება: "მომხმარებლების ამომცნობი სისტემის ინტეგრირება აპლიკაციაში", გვ. 180-182. ავტორები: დ. სირბილაძე, თ. ბახტაძე.
14. შრომები - მართვის ავტომატიზირებული სისტემები N3 (27), 2018. სტატიების დასახელებები: "მონაცემთა გრაფიკული ვიზუალიზაცია Microsoft Power BI-ს მეშვეობით", გვ. 87-94 და "Machine learning usage in different fields", გვ. 163-166. ავტორი: დ. სირბილაძე.