ივანე ჯავახიშვილის სახელობის თბილისის სახელმწიფო უნივერსიტეტი

ლაშა დაუშვილი

უნივერსიტეტის შიდა სერვისების საფასურის გადასახდელი სტუდენტური პორტალი

კომპიუტერული მეცნიერება

ნაშრომი შესრულებულია კომპიუტერული მეცნიერების მაგისტრის აკადემიური ხარისხის მოსაპოვებლად

ხელმძღვანელი: მიხეილ თუთბერიძე, ფიზ.-მათ. მეცნ. კანდიდატი

თბილისი

2017

ანოტაცია

წინამდებარე ნაშრომის ფარგლებში შემუშავებულია ვებ აპლიკაცია, რომელიც წარმოადგენს პორტალს სტუდენტებისათვის და საშუალებას აძლევს სტუდენტებს, გადაიხადონ შიდასაუნივერსიტეტო სერვისების საფასური ვირტუალური ქულებით, რომელსაც გარკვეული წესით დაარიცხავს უნივერსიტეტი თავის სტუდენტებს სემესტრის დასაწყისში. პორტალი სტუდენტებს საშუალებას მისცემს, უფრო ეკონომიურად გამოიყენონ და ეფექტურად მართონ თავიანთი ფინანსური რესურსები, ხოლო უნივერსიტეტს კი საშუალებას მისცემს, რომ თანდათანობით ამოიღოს ნაღდი ფული შიდასაუნივერსიტეტო სერვისების ანაზღაურებიდან.

Annotation

Lasha Daushvili

Student Portal of Paying for University Internal Services

In the scope of the present work the web application is developed, which serves as portal for students and enables them to pay for university internal services using virtual scores, which will be transferred to students at the beginning of the term by the university using some rules. Portal will enable students to use financial resources optimally and effectively. Portal will enable university gradually remove cash as paying method for university internal services.

სარჩევი

შესავალი

მომსახურების საფასურის უნაღდოდ გადახდის მექანიზმი არსებითად დამკვიდრდა ყოველდღიურ ცხოვრებაში. ვიზა ბარათები, ინტერნეტბანკინგი, მობილურის ნომრიდან ჩარიცხვა ფართოდ გამოიყენება საზოგადოების სხვადასხვა ფენების მიერ ამა თუ იმ შესყიდვის განხორციელებისას. ამგვარი მიდგომა განსაკუთრებით ეფექტურია, როდესაც მომხმარებელი ახორციელებს დისტანციურ შესყიდვას.

ამ კუთხით განსაკუთრებული აღნიშვნის ღირსია სტუდენტები. სტუდენტები საზოგადოების ერთ-ერთ ღარიბ ფენას წარმოადგენენ. ამიტომ შესაძლოა, არ გააჩნდეთ ვიზა ბარათი და ინტერნეტბანკინგის ანგარიში, ან თუ გააჩნიათ, მაშინ არ აქვთ საკმარისი თანხა, რომელიც შეიძლება განათავსონ ვიზა ბარათზე ან საბანკო ანგარიშზე და შემდგომ გადაიხადონ მომსახურების საკომისიო. სწორედ ამიტომ, შესაძლებელია ასეთ სტუდენტებს უნივერსიტეტმა შესთავაზოს შიდა ანგარიშსწორების სისტემა, რომლითაც სტუდენტი შეძლებს, შეღავათიან ფასში გადაიხადოს უნივერსიტეტში შიდა სერვისების საფასური.

უნივერსიტეტს შეუძლია, სტუდენტებს გამოუყოს გარკვეული ლიმიტი ყოველ სემესტრში და ამ ლიმიტს მისცეს ვირტუალური ქულების სახე. ამ ვირტუალური ქულებით სტუდენტი შეძლებს, ბიბლიოთეკას გადააღებინოს წიგნის რამდენიმე გვერდის ასლი, უნივერსიტეტის პრინტერზე ამობეჭდოს რამდენიმე გვერდი,
შეიძინოს საუნივერსიტეტო ღონისძიებების ბილეთი, შეიძინოს აბონიმენტი შეღავათიანად უნივერსიტეტის სპორტ დარბაზში, უნივერსიტეტის ბუფეტში იყიდოს ყავა და ასე შემდეგ. ასევე სტუდენტს საშუალება უნდა ჰქონდეს, რომ ასესხოს სხვა სტუდენტს ქულები შემდეგ სემესტრამდე, რომელიც მას ავტომატურად დაუბრუნდება ახალი სემესტრის დაწყებისთანავე. ასევე შესაძლებელია თუ მსესხებელ სტუდენტს დარჩა ქულები, მთლიანად ან გარკვეული ნაწილი დაუბრუნოს მსესხებელს სემესტრის დამთავრებამდე.

წინამდებარე ნაშრომი მიზნად ისახავს, რომ შეიქმნას პორტალი ვებ აპლიკაციის სახით, რომელიც საშუალებას მისცემს სტუდენტს, განახორციელოს გადახდები სწორედ

ზემოხსენებული ვირტუალური ქულების საშუალებით. პორტალს ეყოლება ორი ტიპის მომხმარებელი: პირველი - სტუდენტი, რომელსაც უნდა ანგარიშსწორების განხორციელება, ხოლო მეორე - უნივერსიტეტის შიდა სერვისის მიმწოდებელი, მაგალითად - ბუფეტის მომსახურე პერსონალი და პორტალის ამოცანაა, ამ ორ სუბიექტს შორის ელექტრონული კომუნიკაციის გზით ვირტუალური ქულების გაცვლა.

ამოცანის დასმა

უნივერსიტეტში ჩარიცხულ სტუდენტებს უნივერსიტეტი აძლევს ვირტუალურ ქულებს, რომელთა მეშვეობითაც სტუდენტებს შეუძლიათ უნივერსიტეტში შეიძინონ ყავა (მაგალითად - სტარბაქსის), გადაიღონ ასლი, ამობეჭდონ პრინტერზე, შეიძინონ ბილეთი უნივერსიტეტის მიერ ორგანიზებულ საღამოებზე, შეიძინონ უნივერსიტეტის სპორტდარბაზის აბონემენტი და ა.შ.

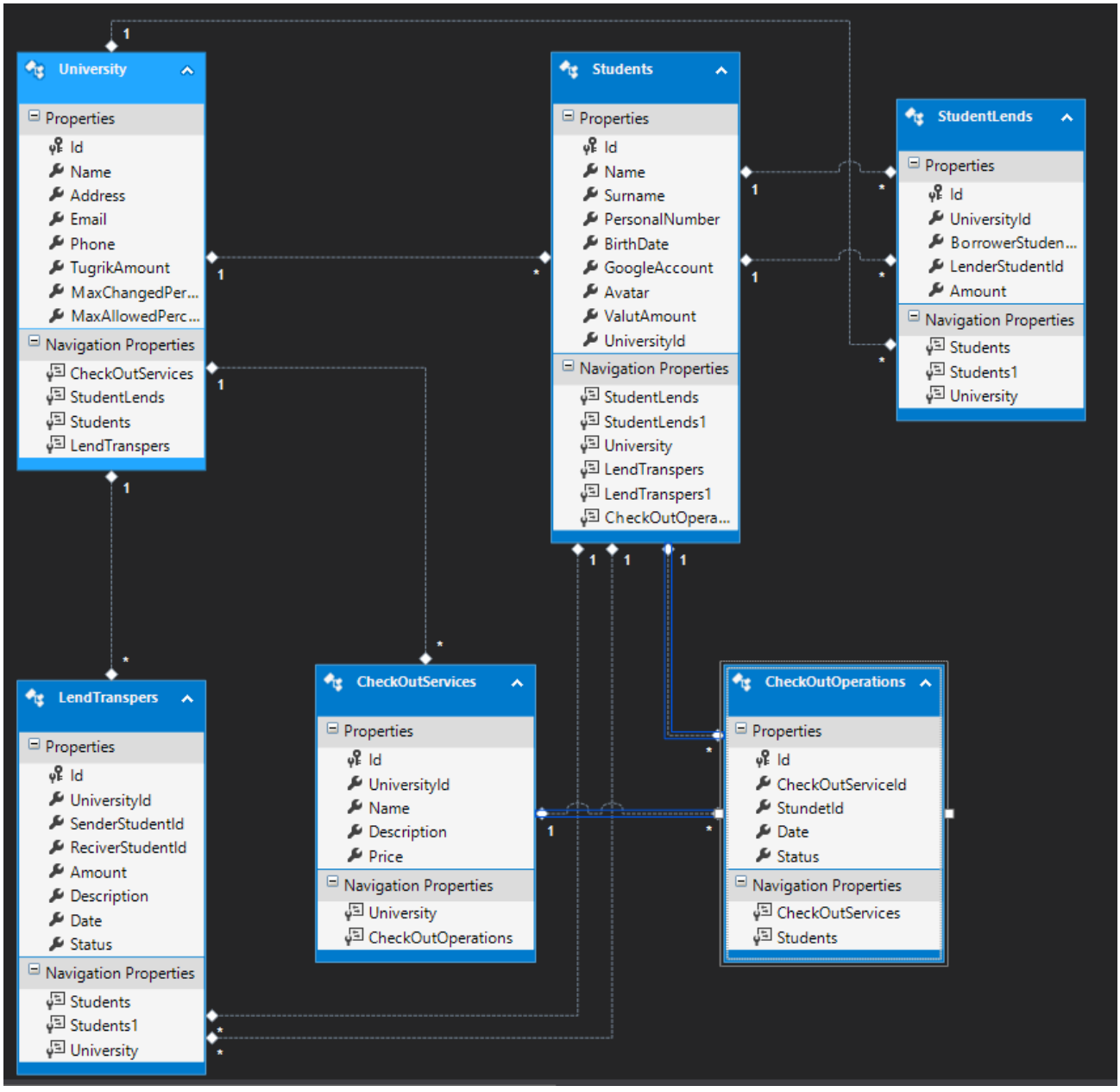
ქულების განაღდება დაუშვებელია. სტუდენტი უფლებამოსილია, თავისი ქულები ასესხოს სხვა სტუდენტს (სესხება უნდა იყოს ლიმიტირებული), იმ პირობით, რომ შემდეგ სემესტრში ქულები ავტომატურად ჩამოეჭრება მსესხებელს და დაუბრუნდება გამსესხებელს.

ქულებით ანგარიშსწორება ხდება ვებ პორტალით შემდეგი პრინციპით:

- სტუდენტი მოითხოვს, რომელიმე სერვისს;
- გამყიდველი მას დაურეგისტრირებს მოთხოვნას;
- სტუდენტი დაადასტურებს, რომ მართლა სურს გამოიყენოს აღნიშნული სერვისი;
- გამყიდველი დააფიქსირებს გადახდას.

მონაცემთა ბაზის მოდელი

ბაზის სტრუქტურა:



ცხრილები

ცხრილი University - უნივერსიტეტების სია, რომელშიც შეიძლება ამოქმედდეს ვირტუალური ქულების სისტემა. ეს ცხრილი შედგება შემდეგი ველებისაგან:

- Id (int) - უნივერსიტეტის იდენტიფიკატორი;

- Name (nvarchar) - უნივერსიტეტის სახელი;
- Address (nvarchar) - მისამართი;
- Email (varchar) - ელექტრონული ფოსტა;
- Phone (varchar) - ტელეფონი;
- TugrikAmount (money) - ყოველ სემესტრში დარიცხული ქულების რაოდენობა;
- MaxChangedPercent (int) - TugrikAmount-ის რამდენი პროცენტი შეიძლება გადარიცხო ერთ ჯერზე;
- MaxAllowedPercent (int) – TugrikAmount-ის რამდენი პროცენტი შეიძლება მიიღოს სტუდენტმა ჯამურად;

ცხრილი Students - სტუდენტების სია, რომლებიც ირიცხებიან უნივერსიტეტებში. ეს ცხრილი შედგება შემდეგი ველებისაგან:

- Id (int) - სტუდენტის იდენტიფიკატორი;
- Name (nvarchar) – სახელი;
- Surname (nvarchar) - გვარი;
- PersonalNumber (varchar) – პირადი ნომერი;
- Birthdate (Date) – დაბადების თარიღი
- GoogleAccount (varchar) – გუგლის ანგარიში (როგორც წესი, უნივერსიტეტებს გუგლი უფასოდ აძლევს ამ ანგარიშს);
- Avatar (varchar) – ფოტო;
- ValutAmount (money) – მიმდინარე თანხა;
- UniversityId (int) - უნივერსიტეტის იდენტიფიკატორი.

ცხრილი CheckOutServices - სერვისების სია, რომლებიც აქვს უნივერსიტეტს. ეს ცხრილი შედგება შემდეგი ველებისაგან:

- Id (int) - სერვისის იდენტიფიკატორი;
- UniversityId (int) – უნივერსიტეტის იდენტიფიკატორი;
- Name (nvarchar) – სერვისის დასახელება;
- Description (nvarchar) – სერვისის აღწერა;

- Price (money) - ფასი.

ცხრილი CheckOutOperations - სერვისების შესყიდვები. ეს ცხრილი შედგება შემდეგი ველებისაგან:

- Id (int) - სერვისის იდენტიფიკატორი;
- CheckOutServiceId(int) – სერვისის იდენტიფიკატორი;
- StundetId (int) – სტუდენტის იდენტიფიკატორი;
- Date (DateTime) – ოპერაციის თარიღი;
- Status (nvarchar) - სტატუსი (დარეგისტრირებული, გადახდილი, უარყოფილი).

ცხრილი StudentLends - სტუდენტების სესხები. ეს ცხრილი შედგება შემდეგი ველებისაგან:

- Id (int) - ჩანაწერის იდენტიფიკატორი;
- UniversityId(int) – უნივერსიტეტის იდენტიფიკატორი;
- BorrowerStudentId (int) – გამსესხებელი სტუდენტის იდენტიფიკატორი;
- LenderStudentId (int) – მსესხებელი სტუდენტის იდენტიფიკატორი;
- Amount (money) - გასესხებული თანხა.

ცხრილი LendTranspers - სტუდენტებს შორის გადარიცხვები. ეს ცხრილი შედგება შემდეგი ველებისაგან:

- Id (int) - ტრანსფერის იდენტიფიკატორი;
- UniversityId(int) – უნივერსიტეტის იდენტიფიკატორი;
- SenderStudentId (int) – გადამრიცხავი სტუდენტის იდენტიფიკატორი;
- ReciverStudentId (int) – მიმღები სტუდენტის იდენტიფიკატორი;
- Amount (money) - გასესხებული თანხა;
- Description(nvarchar) – აღწერა;
- Date (DateTime) - თარიღი;
- Status (nvarchar) - სტატუსი (გადარიცხული, მიღებული, უარყოფილი);

შენახული პროცედურები:

PayBackDebpts - უზრუნველყოფს სტუდენტებს შორის გადარიცხვების დაჯამებას, რაც გულისხმობს რომ სისტემამ კორექტულად დაადგინოს სტუდენტთა შორის დავალიანებები.

გამოყენებული ტექნოლოგიები

სერვერული აპლიკაციის შესამუშავებლად გამოყენებულ იქნა .NET პლატფორმა, კერძოდ - ASP.NET MVC ტექნოლოგია, რომელიც ერთერთი ყველაზე გამოყენებადი და ეფექტური ტექნოლოგიაა. იგი ეფუძნება პრინციპს - მოდელი/ხედი/კონტროლერი და იძლევა შრომის დანაწილების საშუალებას. განსაკუთრებული აღნიშვნის ღირსია ამ ტექნოლოგიის უსაფრთხოება და წარმადობა.

მონაცემთა ბაზა შემუშავებულ იქნა მონაცემთა ბაზის მართვის სისტემა Microsoft SQL Server 2012-ის გამოყენებით. Microsoft SQL Server 2012 არის სრულფასოვანი მონაცემთა ბაზის მართვის სისტემა რელაციური მონაცემთა ბაზის შესაქმნელად.

ვებ აპლიკაციაში ინტენსიურად არის გამოყენებული Web API მეთოდები, რომლებიც საშუალებას იძლევა, რომ გამოძახებულ იქნას არამარტო ბრაუზერიდან, არამედ მობილური აპლიკაციიდანაც, რაც აპლიკაციის განვითარების შესაძლებლობას იძლევა.

Google OAuth 2.0 protocol აუთენტიფიკაციისა და ავტორიზაციისათვის, რომელიც ნებისმიერ პროგრამას აძლევს საშუალებას, დაუკავშირეს Google-ს და გაიაროს ავტორიზაცია Google-ს ანგარიშით.

პროგრამული უზრუნველყოფის კოდი

პროგრამული უზრუნველყოფა შესრულდა ASP.NET MVC პლატფორმის გამოყენებით.

სერვერული აპლიკაციის კოდის იმ ძირითად ნაწილს, რომელიც პასუხისმგებელია პროგრამის სწორად ფუნქციონირებაზე აქვს შემდეგი სახე:

მთავარი მეთოდებია

RegistrationTransper (გადარიცხვის დარეგისტრირება, როდესაც ერთი სტუდენტი ურიცხავს მეორეს)

```
public int RegistrationTransper(Students sender, Students
reciever, decimal amount, string description, string statusDesc)
 {
     var lendTransper = new LendTranspers
     {
         SenderStudentId = sender.Id,
         ReciverStudentId = reciever.Id,
         Amount = amount,
         Description = description,
         Date = DateTime.Now,
         UniversityId = sender.UniversityId,
         Status = statusDesc
     };

     dbContext.LendTranspers.Add(lendTransper);

     dbContext.SaveChanges();

     return lendTransper.Id;
 }
```

თანხმობის შემთხვევაში TransperToFriend მეთოდი

```
public string TransperToFriend(Students sender, Students reciever,
decimal amount, string description, int rowId)
        {
            using (var dbContextTransaction =
dbContext.Database.BeginTransaction())
            {
                try
                {
                    var newStatus = "დადასტურებული";
                    var command = @"UPDATE LendTranspers SET Status = " +
"N'" + newStatus + "'" +
                            " WHERE Id=" + rowId;
                    dbContext.Database.ExecuteSqlCommand(command);

                    //Update dbo.Students
                    var newAmont = sender.ValutAmount - amount;
                    command = @"UPDATE Students SET ValutAmount = " +
newAmont.ToString().Replace(',', '.') +
                            " WHERE Id=" + sender.Id;
                    dbContext.Database.ExecuteSqlCommand(command);

                    //Update dbo.Students
                    newAmont = reciever.ValutAmount + amount;
                    command = @"UPDATE Students SET ValutAmount = " +
newAmont.ToString().Replace(',', '.') +
                            " WHERE Id=" + reciever.Id;
                    dbContext.Database.ExecuteSqlCommand(command);

                    dbContext.PayBackDebpts(sender.Id, reciever.Id,
amount);

                    dbContext.SaveChanges();
```

```
                    dbContextTransaction.Commit();
                }
                catch (Exception)
                {
                    dbContextTransaction.Rollback();
                    return "შეცდომა ოპერაციის დროს";
                }
            }

            return "done";
```

და ერთი მთავარი პროცედურა რომელიც პასუხიმგებელია დაადგინოს სტუდენტების სესხები.

```
ALTER PROCEDURE [andreauser].[PayBackDebpts]
     @sender int,
     @reciever int,
     @Amount decimal
AS
BEGIN
DECLARE @CurrLend decimal = 0
IF EXISTS (SELECT 1 FROM [andreauser].[StudentLends] WHERE
(BorrowerStudentId = @sender AND LenderStudentId = @reciever) OR
(BorrowerStudentId = @reciever AND LenderStudentId = @sender))
BEGIN
     IF EXISTS(SELECT 1 FROM [andreauser].[StudentLends] WHERE
(BorrowerStudentId = @sender AND LenderStudentId = @reciever))
     BEGIN
          SELECT @CurrLend = SUM(Amount) FROM
[andreauser].[StudentLends] WHERE (BorrowerStudentId = @sender AND
LenderStudentId = @reciever)
```

```
            UPDATE [andreauser].[StudentLends] SET Amount = @CurrLend +
@Amount
            WHERE BorrowerStudentId = @sender AND LenderStudentId =
@reciever
      END
      IF EXISTS(SELECT 1 FROM [andreauser].[StudentLends] WHERE
(BorrowerStudentId = @reciever AND LenderStudentId = @sender))
      BEGIN
            SELECT @CurrLend = SUM(Amount) FROM
[andreauser].[StudentLends] WHERE (BorrowerStudentId = @reciever AND
LenderStudentId = @sender)
            IF(@CurrLend <= @Amount)
            BEGIN
                  DELETE FROM [andreauser].[StudentLends]
                  WHERE BorrowerStudentId = @reciever AND LenderStudentId
= @sender

                  IF(@CurrLend < @Amount)
                  BEGIN
                        INSERT INTO [andreauser].[StudentLends]
                        VALUES(1, @sender, @reciever, @Amount-@CurrLend)
                  END
            END
            ELSE
            BEGIN
                        UPDATE [andreauser].[StudentLends] SET Amount =
@CurrLend - @Amount
                        WHERE BorrowerStudentId = @reciever AND
LenderStudentId = @sender
            END
      END
END
ELSE
```

```
BEGIN
      INSERT INTO [andreauser].[StudentLends]
      VALUES(1, @sender, @reciever, @Amount)
END
END
```

ასევე REST Service მეთოდი რომელიც პასუხიმგებელია გადახდის დარეგისტრირებაზე, რომელიც შეუძლია მიიერთოს ნებისმიერმა კლიენტმა და შესაბამისად არის თავსებადი ნებისმიერი ტექნოლოგიის პროგრამასთან

```
        [HttpGet]
        public IHttpActionResult Create()
        {
            var pairs = Request.GetQueryNameValuePairs();

            var studentId = pairs.Where(x => x.Key ==
"StudentId").FirstOrDefault().Value;
            var serviceId = pairs.Where(x => x.Key ==
"ServiceId").FirstOrDefault().Value;

            using (CheckOutServiceEntities dbContext = new
CheckOutServiceEntities())
            {
                var student = dbContext.Students.Where(x =>
x.GoogleAccount == studentId +
Settings.Default.MailSufix).FirstOrDefault();
                if (student == null)
                {
                    return BadRequest("სტუდენტი არ მოიძებნა");
                }
                var service = dbContext.CheckOutServices.Where(x =>
x.Id.ToString() == serviceId).FirstOrDefault();
                if (service == null)
```

```
                {
                    return BadRequest("სერვისი ვერ მოიძებნა");
                }

                AddCheckOutOperation(service, student);
            }

            return Ok();
        }

        private void AddCheckOutOperation(CheckOutServices service,
Students student)
        {
            using (CheckOutServiceEntities dbContext = new
CheckOutServiceEntities())
            {
                dbContext.CheckOutOperations.Add(new CheckOutOperations {
CheckOutServiceId = service.Id, StundetId = student.Id, Date =
DateTime.Now, Status = "დარეგისტრირებული" });
                dbContext.SaveChanges();
            }
        }
```

და გადახდაზე თანხმობა Payment

```
       public void Payment(int rowId, decimal price, Students user)
        {
            using (var dbContextTransaction =
dbContext.Database.BeginTransaction())
            {
                try
```

```
                {
                    var newStatus = "გადახდილი";
                    var command = @"UPDATE CheckOutOperations SET Status
= " + "N'" + newStatus + "'" +
                            " WHERE Id=" + rowId;
                    dbContext.Database.ExecuteSqlCommand(command);

                    //Update dbo.Students
                    var newAmont = user.ValutAmount - price;
                    command = @"UPDATE Students SET ValutAmount = " +
newAmont.ToString().Replace(',', '.') +
                            " WHERE Id=" + user.Id;
                    dbContext.Database.ExecuteSqlCommand(command);

                    dbContext.SaveChanges();

                    dbContextTransaction.Commit();
                }
                catch (Exception ex)
                {
                    dbContextTransaction.Rollback();
                    throw ex;
                }
            }
        }
```

Google Oauth2

აბსტრაქტული კლასი, რომელსაც დამჭირდა იმისათვის რომ საჭიროების შემთვევაში გადავრთო მაგალითად Facebook ავტორიზაციაზე.

```
public abstract class SocialMediaAuthenticationProviderBase
    {
```

```
        protected string _appId { get; set; }
        protected string _secretKey { get; set; }
        protected string _callbackUrl { get; set; }
        public SocialMediaAuthenticationProviderBase(string appId, string
secretKey, string callbackUrl)
        {
            _appId = appId;
            _secretKey = secretKey;
            _callbackUrl = callbackUrl;
        }
        public abstract string GetRedirectUrl();
        public abstract OAuthUserInfo GetUserInfo();
        protected string BuildQuery(Dictionary<string, string> queries)
        {
            if (queries == null || queries.Count == 0) { return
String.Empty; }
            StringBuilder sb = new StringBuilder();
            sb.Append("?");
            foreach (var q in queries)
            {
                sb.AppendFormat("{0}={1}&", q.Key, q.Value);
            }
            return sb.ToString().TrimEnd('&');
        }
        protected string MakeHttpGetRequest(string url)
        {
            using (var http = new HttpClient())
            {
                var response =
http.GetAsync(url).GetAwaiter().GetResult();
                return
response.Content.ReadAsStringAsync().GetAwaiter().GetResult();
            }
        }
        protected string MakeHttpPostRequest(string url,
IEnumerable<KeyValuePair<string, string>> body)
        {
            using (var http = new HttpClient())
            {
                var content = new FormUrlEncodedContent(body);
                var result = http.PostAsync(url,
content).GetAwaiter().GetResult();
                return
result.Content.ReadAsStringAsync().GetAwaiter().GetResult();
            }
        }
        protected bool IsAntiForgeryTokenValid(string key)
        {
```

```
            return HttpContext.Current.Request["state"] != null &&
HttpContext.Current.Session[key] != null
                                                                &&
HttpContext.Current.Request["state"].ToString() ==
HttpContext.Current.Session[key].ToString();
        }
    }
```

Google ავტორიზაციის კლასი.

```
public class GoogleAuthenticationProvider :
SocialMediaAuthenticationProviderBase
    {
        private const string _authUrl =
"https://accounts.google.com/o/oauth2/auth";
        private const string _profileUrl =
"https://www.googleapis.com/oauth2/v1/userinfo";
        private const string _tokenUrl =
"https://www.googleapis.com/oauth2/v3/token";
        private const string _stateKey = "googlestate";
        public GoogleAuthenticationProvider(string appId, string
secretKey, string callbackUrl) : base(appId, secretKey, callbackUrl) { }
        public override string GetRedirectUrl()
        {
            string googleState = new Guid().ToString();
            HttpContext.Current.Session[_stateKey] = googleState;
            Dictionary<string, string> queries = new Dictionary<string,
string>();
            queries.Add("client_id", _appId);
            queries.Add("response_type", "code");
            queries.Add("scope", "openid%20email");
            //აქ უნდა დავამატო: დომეინი
            queries.Add("redirect_uri", _callbackUrl);
            queries.Add("state", googleState);
            return _authUrl + base.BuildQuery(queries);
        }

        public override OAuthUserInfo GetUserInfo()
        {
            if (!base.IsAntiForgeryTokenValid(_stateKey)) { return null;
}

            string content = RequestUserInfo(GetToken());
            dynamic userInfo =
Newtonsoft.Json.JsonConvert.DeserializeObject(content);
            return new OAuthUserInfo()
            {
                FirstName = userInfo.given_name,
                LastName = userInfo.family_name,
```

```
                Email = userInfo.email,
                GoogleId = userInfo.id
            };
        }

        private string GetToken()
        {
            var body = new Dictionary<string, string>();
            body.Add("code", HttpContext.Current.Request["code"]);
            body.Add("client_id", _appId);
            body.Add("client_secret", _secretKey);
            body.Add("redirect_uri", _callbackUrl);
            body.Add("grant_type", "authorization_code");

            string content = base.MakeHttpPostRequest(_tokenUrl, body);

            dynamic tokenInfo =
Newtonsoft.Json.JsonConvert.DeserializeObject(content);
            return tokenInfo.access_token.ToString();
        }
        private string RequestUserInfo(string token)
        {
            var queries = new Dictionary<string, string>();
            queries.Add("alt", "json");
            queries.Add("access_token", token);

            return base.MakeHttpGetRequest(_profileUrl +
base.BuildQuery(queries));
        }
    }
```

დასკვნა

ამრიგად, წინამდებარე ნაშრომის ფარგლებში შექმნილია სტუდენტური პორტალი, რომლის მეშვეობით სტუდენტი ახორციელებს უნივერსიტეტის შიდა სერვისების საფასურის გადახდას იმგვარად, რომ არ აქვს შეხება ფულთან. პორტალი ასევე საშუალებას აძლევს სტუდენტებს, ერთმანეთს ასესხონ ვირტუალური ქულები, რომლებიც გამოიყენება გადახდებში. პორტალი არის საუნივერსიტეტო აპლიკაცია. შესაძლებელია მისი დანერგვა სხვადასხვა სიდიდის უნივერსიტეტებში. მას მიცემული აქვს საბოლოო სახე, რაც შესაძლებელს ხდის მის გამოყენებას კომერციული მიზნებისათვის.

გამოყენებული ლიტერატურა

1. JavaScript for Web Developers 3rd Edition by Nicholas C. Zakas, 2012.
2. Andrew Troelsen, Philip Japikse. C# 6.0 and the .NET 4.6 Framework. Seventh edition. Apress, 2015.
3. Itzik Ben-Gan, Dejan Sarka, Ron Talmage. Querying Microsoft® SQL Server® 2012. O'Reilly Media, Inc., 2012.