INTEGRATED SYSTEMS CHECKPOINTS

Лабораториска вежба 5 (Група Б) / Laboratory exercise 5 (Group B)

Дадени ви се две апликации за нарачување на карти за филмови и Админ апликација. Симнете го кодот поставен на курсот и дополнете ја апликацијата со следниве функционалности:

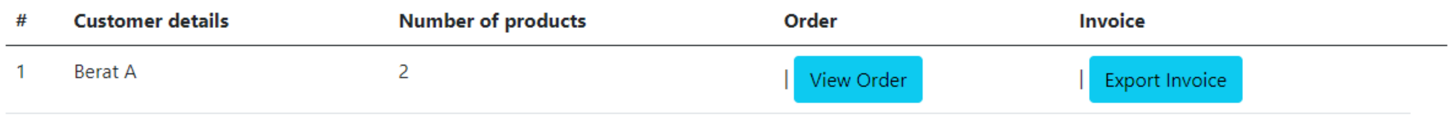
- Импорт на Филмови
- Експорт на сите нарачки од Admin апликација во PDF
- Експорт на секоја нарачка посебно и продукти во нарачка од Admin апликација во PDF

You are given two applications for ordering movie tickets and an Admin application. Download the code posted on the course and complete the application with the following functionalities:

- Import of Movies
- Export of all orders from Admin application to PDF
- Export of every order and products in order from Admin application to PDF.

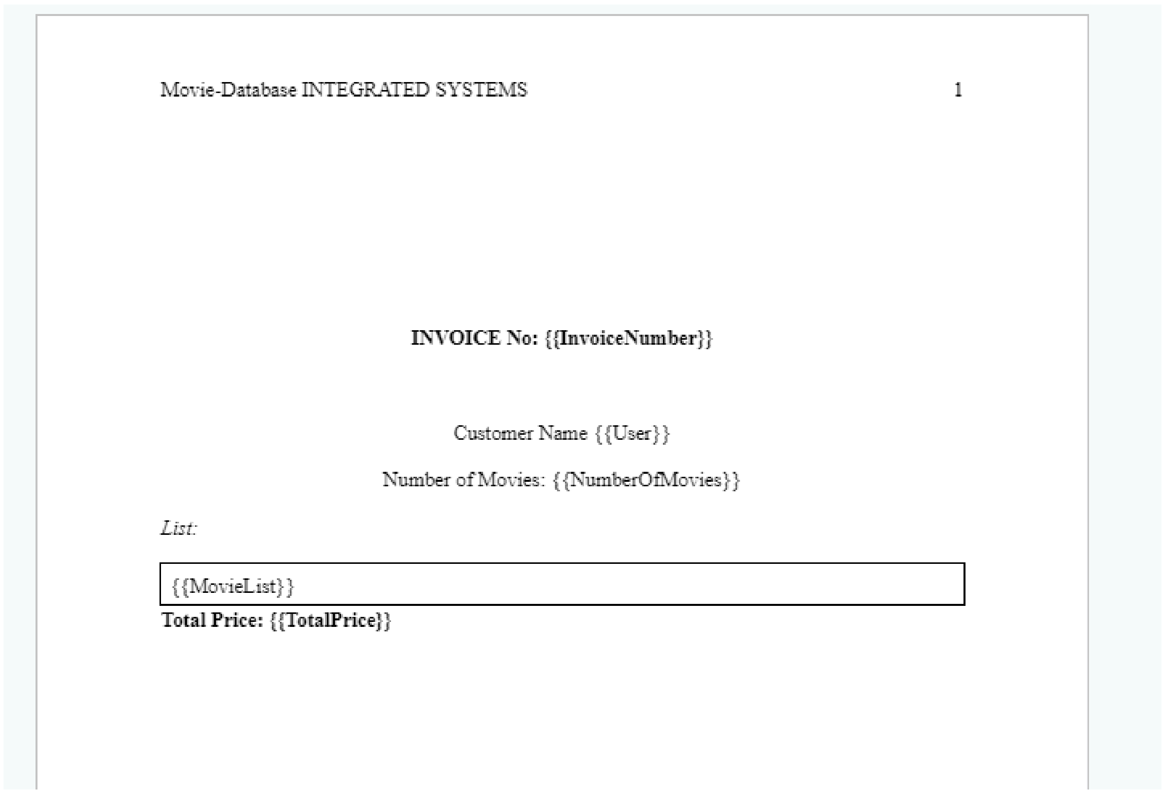
Step 1 : Create Button for Invoice

```
<td>
    |
    <a asp-action="ExportInvoice" asp-route-id="@item.Id" class="btn btn-info">Export Invoice</a>
</td>
```



#	Customer details	Number of products	Order	Invoice
1	Berat A	2	| View Order	| Export Invoice

Step 2 : Create a word file template



Movie-Database INTEGRATED SYSTEMS 1

INVOICE No: {{InvoiceNumber}}

Customer Name {{User}}

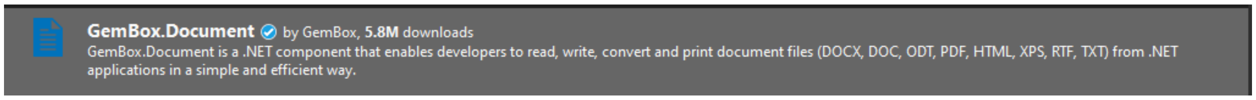
Number of Movies: {{NumberOfMovies}}

List:

{{MovieList}}

Total Price: {{TotalPrice}}

Download Gembox Document package to be able to read documents from VS



GemBox.Document by GemBox, **5.8M** downloads
GemBox.Document is a .NET component that enables developers to read, write, convert and print document files (DOCX, DOC, ODT, PDF, HTML, XPS, RTF, TXT) from .NET applications in a simple and efficient way.

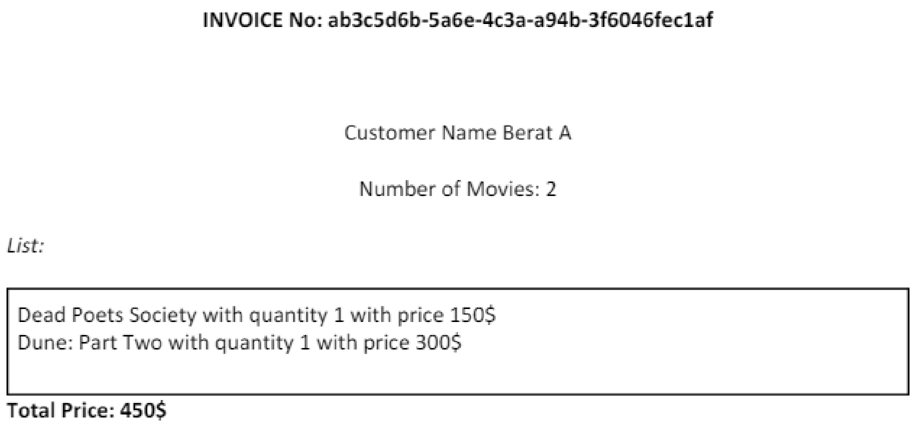
And to be able to use it import the license in the OrderController **Constructor**

```
public OrderController()
{
    ComponentInfo.SetLicense("FREE-LIMITED-KEY");
}
```

Step 3: Write The Export Code

```
public IActionResult ExportInvoice(Guid id)
{
    //COPIED FROM DETAILS
    HttpClient client = new HttpClient();
    string URL = "http://localhost:5054/api/Admin/GetDetailsForOrder";
    var model = new
    {
        Id = id
    };
    HttpContent content = new StringContent(JsonConvert.SerializeObject(model), Encoding.UTF8,
"application/json");

    HttpResponseMessage response = client.PostAsync(URL, content).Result;

    var data = response.Content.ReadAsAsync<Order>().Result;

    if (data == null)
    {
        // Handle the error appropriately, e.g., log the error and return
        // You might want to throw an exception or return a default value here
        throw new Exception("Data is null");
    }
    //COPIED FROM DETAILS

    var templatePath = Path.Combine(Directory.GetCurrentDirectory(), "Invoice.docx");

    var document = DocumentModel.Load(templatePath);
    document.Content.Replace("{{InvoiceNumber}}", data.Id.ToString());
    document.Content.Replace("{{User}}", data.Owner.FirstName.ToString() +" "
+data.Owner.LastName.ToString());
    document.Content.Replace("{{NumberOfMovies}}", data.ProductInOrders.Count.ToString());


    StringBuilder sb = new StringBuilder();
    var totalPrice = 0;
    foreach (var item in data.ProductInOrders)
    {
        sb.Append(item.OrderedProduct.Movie.MovieName + " with quantity " + item.Quantity + " with price " +
item.OrderedProduct.Price + "$");
        //New Line
        sb.Append(Environment.NewLine);
        totalPrice += item.Quantity * (int)item.OrderedProduct.Price;
    }
    document.Content.Replace("{{MovieList}}", sb.ToString());
    document.Content.Replace("{{TotalPrice}}", totalPrice.ToString() + "$");

    var stream = new MemoryStream();
    document.Save(stream, new PdfSaveOptions());
    return File(stream.ToArray(), new PdfSaveOptions().ContentType, "ExportedInvoice.pdf");
}
```

Export now works, example:



INVOICE No: ab3c5d6b-5a6e-4c3a-a94b-3f6046fec1af

Customer Name Berat A

Number of Movies: 2

List:

Dead Poets Society with quantity 1 with price 150$
Dune: Part Two with quantity 1 with price 300$

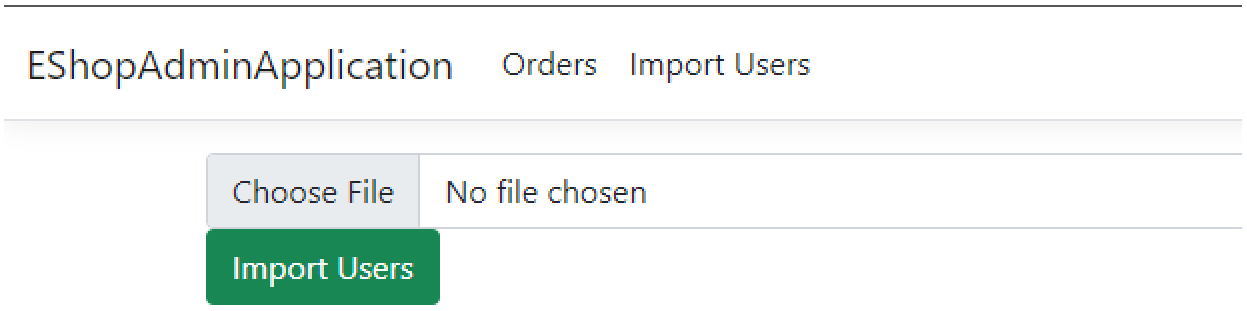
Total Price: 450$

Step 4: Import Users

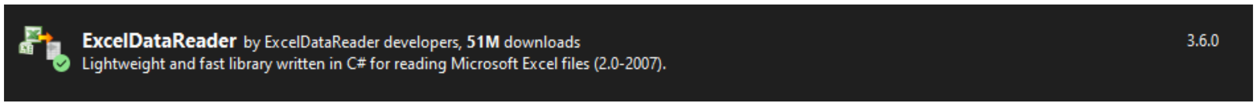
Make View From Importing Users with a form that accepts file

```
<form asp-controller="User" asp-action="ImportUsers" method="post" enctype="multipart/form-data">
    <div class="form-group">
        <input type="file" name="file" class="form-control" />
    </div>

    </hr>

    <button type="submit" class="btn btn-success">Import Users</button>

</form>
```

Result:



Package Download: ExcelReader



ExcelDataReader by ExcelDataReader developers, **51M** downloads
Lightweight and fast library written in C# for reading Microsoft Excel files (2.0-2007).
3.6.0

Code (Get All Users From File & Import Them To Database):

```
namespace MVCAdminApplication.Controllers
{
    public class UserController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }

        public IActionResult ImportUsers(IFormFile file)
        {
            string pathToUpload = $"{Directory.GetCurrentDirectory()}\\files\\{file.FileName}";

            using (FileStream fileStream = System.IO.File.Create(pathToUpload))
            {
                file.CopyTo(fileStream); ;
                fileStream.Flush();
            }

            List<EShopApplicationUser> users = getAllUsersFromFile(file.FileName);

            HttpClient client = new HttpClient();
            string URL = "http://localhost:5180/api/Admin/ImportAllUsers";
            HttpContent content = new StringContent(JsonConvert.SerializeObject(users), Encoding.UTF8,
"application/json");

            HttpResponseMessage response = client.PostAsync(URL, content).Result;

            var data = response.Content.ReadAsAsync<bool>().Result;
            return RedirectToAction("Index", "Order");

        }

        private List<EShopApplicationUser> getAllUsersFromFile(string fileName)
        {
            List<EShopApplicationUser> users = new List<EShopApplicationUser>();
            string filePath = $"{Directory.GetCurrentDirectory()}\\files\\{fileName}";
            System.Text.Encoding.RegisterProvider(System.Text.CodePagesEncodingProvider.Instance);

            using (var stream = System.IO.File.Open(filePath, FileMode.Open, FileAccess.Read))
            {


                using (var reader = ExcelReaderFactory.CreateReader(stream))
                {
                    while (reader.Read())
                    {
                        users.Add(new EShopApplicationUser
                        {
                            Email = reader.GetValue(0).ToString(),
                            Password = reader.GetValue(1).ToString(),
                            ConfirmPassword = reader.GetValue(2).ToString()
                        });

                    }
                }
            }
            return users;
        }
    }
}
```

Important:

You may need to modify the EshopApplicationUser class so that you can save Email,Password & ConfirmPassword

An Alternative is to save all of these in a new class, call it User and have only the Email,Password and ConfirmPassword, both ways work

```
    public class EShopApplicationUser
    {
        public string? FirstName { get; set; }
        public string? LastName { get; set; }
        public string? Address { get; set; }

        public string? Email { get; set; }

        public string? UserName { get; set; }

        public string? Password { get; set; }

        public string? ConfirmPassword { get; set;}
    }
}
```