



УНИВЕРЗИТЕТ У НОВОМ САДУ  
ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА У  
НОВОМ САДУ



Андрија Мусић, PR85/2019

**Онлине продавница**

ПРОЈЕКАТ

- Примењено софтверско инжењерство (ОАС) -

Нови Сад, септембар, 2023.

## **САДРЖАЈ**

1. ОПИС РЕШАВАНОГ ПРОБЛЕМА
2. ОПИС КОРИШЋЕНИХ ТЕХНОЛОГИЈА И АЛАТА
3. ОПИС РЕШЕЊА ПРОБЛЕМА
4. ПРЕДЛОЗИ ЗА ДАЉА УСАВРШАВАЊА
5. ЛИТЕРАТУРА

## ОПИС РЕШАВАНОГ ПРОБЛЕМА

Главни циљ овог пројекта је да обезбеди корисницима могућност за куповину производа. Продавница се састоји од корисника, поруџбина и артикала.

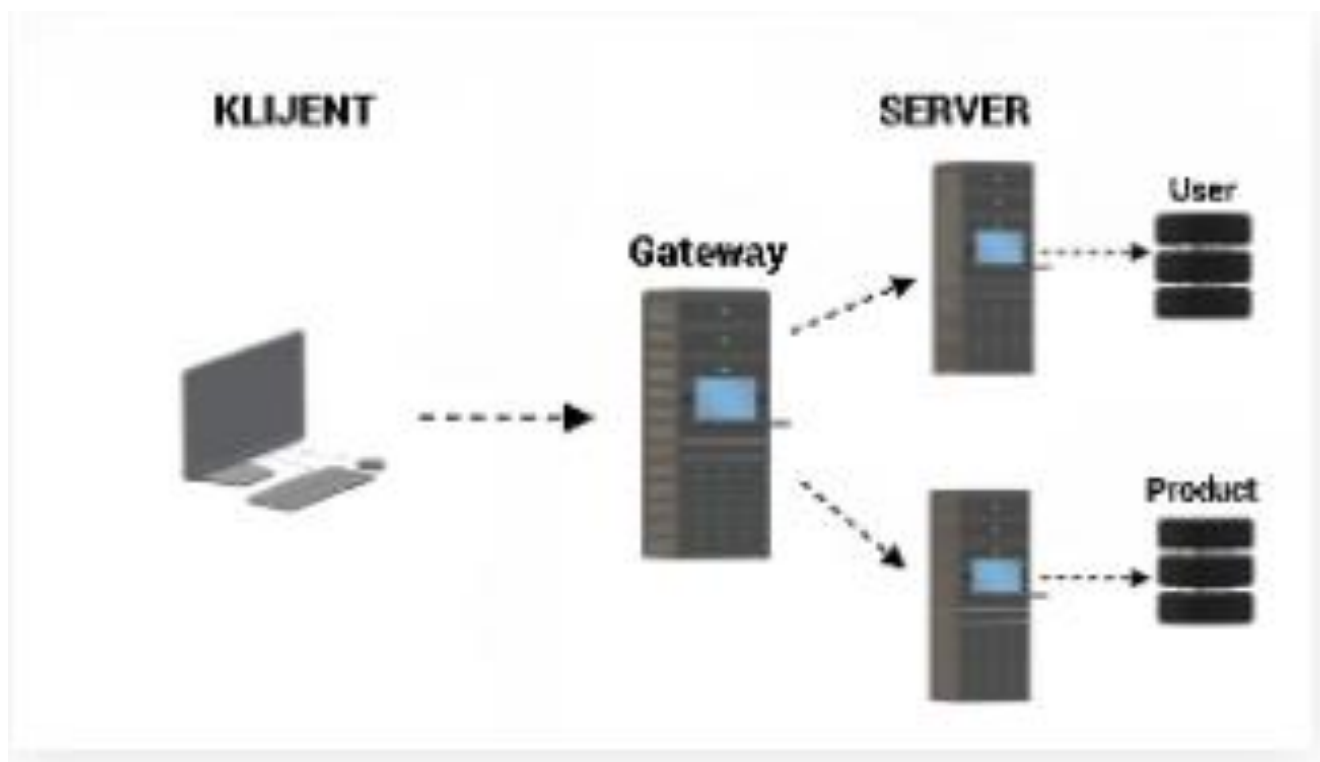
У систему постоје три врсте корисника:

- Купац
- Продавац
- Администратор

Корисници који нису пријављени и немају регистрован налог не могу користити апликацију. Купци могу извршити регистрацију путем друштвене мреже или путем апликације. Продавци се региструју преко апликације. Администраторски корисници су унети директно у базу и не захтевају регистрацију.

Апликација се базира на клијентској страни, која је развијена користећи React библиотеку, и серверској страни, која је подељена на два микросервиса са независним базама података.

Сви захтеви са клијентске стране се упућују ка API Gateway-у, који прихвата пристигле захтеве и прослеђује их у складу са конфигурацијом (Слика 1).



Слика 1 – изглед система

Сва бизнис логика, укључујући ауторизацију и аутентификацију корисника, интегрисана је у микросервисне апликације, док API Gateway функционише као примач за захтеве од стране клијентске апликације и затим њихово редиректовање ка одговарајућим микросервисима.. Основна функција API Gateway-а је да омогући комуникацију између клијентске стране и микросервиса. Микросервиси су организовани према типу ентитета са којима раде.

## ОПИС КОРИШЋЕНИХ ТЕХНОЛОГИЈА И АЛАТА

**Visual Studio Code** [1] представља бесплатни развојни интерфејс који је дизајниран да омогући програмирање различитих врста апликација. Овај алат је познат по својој лакоћи у коришћењу, брзини и обиљу функција. Нуди многе напредне могућности као што су преглед кода, интелигентно навођење, дебаговање, интеграцију са системима контроле верзија и подршку за многе екстензије. **Visual Studio Code** је крос-платформски алат, доступан на Windows, macOS и Linux оперативним системима. Због своје приступачности, проширивости и брзине, ова апликација је изузетно популарна међу програмерима. У конкретном случају, **Visual Studio Code** је коришћен за развој фронтенд решења.

**Visual Studio 2022** [2] представља бесплатно интегрисано развојно окружење (IDE) које обезбеђује различите алатке за програмирање и развој софтвера. Ово интегрисано окружење омогућава програмерима да креирају разноврсне апликације, и нуди широк спектар функционалности као што су преглед кода, дебаговање, управљање верзијама, и интеграција са различитим програмским језицима и платформама. У контексту ове апликације, **Visual Studio 2022** је био коришћен за развој бекенд решења.

**React** [3] је JavaScript библиотека за развој корисничког интерфејса у веб апликацијама. Ова библиотека примењује архитектуру базирану на компонентама, што олакшава модуларност и управљање кодом. Има виртуелни DOM (Document Object Model) приступ, што доприноси ефикасном ажурирању само промењених делова корисничког интерфејса, што у суштини повећава брзину и перформансе апликације."

**SQL Management Studio** [4] представља алатку која омогућава администрацију и управљање SQL Server базама података. Ова алатка обезбеђује широк спектар напредних функционалности, укључујући креирање, уређивање и извршавање SQL упита, дебаговање и "профајлинг" базе података, управљање безбедношћу и извршавање административних задатака. Има кориснички интерфејс који омогућава лак приступ и управљање подацима у бази. У контексту ове апликације, **SQL Management Studio** је коришћен за развој решења базе података.

**ASP .NET Core 6.0 Web API C#** [5] - framework за изградњу веб апликација у C# језику. Пружа ефикасан начин за развој API -ја за модерне апликације. Платформа се састоји од три независне и међусобно повезане апликације, које су развијене користећи **ASP.NET CORE 6.0**. Те апликације укључују API Gateway, Product и User.

**API Gateway** користи библиотеку **Ocelot** [6] за управљање захтевима и усмеравање у микросервисним архитектурама. Ова библиотека омогућава клијентским апликацијама да комуницирају са више микросервиса користећи јединствену приступну тачку. **Ocelot** се користи са циљем поједностављења клијентских апликација и обезбеђује једноставан и конзистентан интерфејс према спољашњим сервисима. Његова конфигурација је једноставна и извршава се путем JSON формата. Поред усмеравања, **Ocelot** пружа и функционалности као што су аутентификација, ауторизација, надзор и контрола оптерећења.

## ОПИС РЕШЕЊА ПРОБЛЕМА

Системски дизајн обухвата четири апликације:

- API Gateway
- User
- Product
- Клијентска апликација


Слика 3.1

Прво што се приказује кориснику је логин страница (Слика 3.1) са могућношћу регистравања ако нема већ нема направљен налог. Корисник то може урадити преко популарне друштвене мреже Фејсбук или да ручно попуни форму за регистрацију (Слика 3.2).


```
2 references
public async Task<ResponsePackage<string>> Save(RegisterUserDataIn dataIn)
{
    dataIn.Email = dataIn.Email.ToLower().Trim();

    if(dataIn.Id == null) //create new
    {
        //generate user
        var userForDb = new User()
        {
            Address = dataIn.Address,
            Email = dataIn.Email,
            FirstName = dataIn.FirstName,
            UserName = dataIn.Username,
            BirthDate = dataIn.BirthDate.GetValueOrDefault(),
            LastName = dataIn.LastName,
            Password = _passwordMD5Service.GetMd5Hash(dataIn.Password),
            Role = (Role)Int32.Parse(dataIn.RoleId),
            Status = ((Role)Int32.Parse(dataIn.RoleId)) == Role.Customer ? UserStatus.Approved : UserStatus.Pending,
            ActivateKey = System.Guid.NewGuid().ToString(),
            LastUpdateTime = DateTime.Now
        };
    }
}
```

Слика 3.3



## REGISTER

<b>First Name *</b> <input type="text"/>	<b>Last Name *</b> <input type="text"/>
<b>UserName *</b> <input type="text"/>	<b>Address *</b> <input type="text"/>
<b>Birth Date *</b> <input type="text" value="ДД - ММ - ГГГГ -"/> 	<b>Email *</b> <input type="text"/>
<b>Role *</b> <div>Customer ▼</div>	<b>Password *</b> <input type="password"/>
<b>Profile Picture *</b> <div>Одабери фајл <span>Није одабрано</span></div>	<b>Confirm Password *</b> <input type="password"/>

[Log in](#)

Слика 3.2

Сви подаци морају бити валидно унети. Лозинка се у бази чува у хешираном формату. Имплементација за регистрацију новог корисника је приказана на слици (Слика 3.3).

## USER

Ова .NET WEB API апликација користи трослојну архитектуру. Три слоја архитектуре су:

- Слој контролера, који излаже тачке приступа апликацији.
- Слој сервиса, одговоран за извршавање пословне логике и операција као што су листање, рачунање, слање порука и други задаци.
- Слој података, који се бави задацима додавања, чувања и брисања података у бази података.

Ова апликација има одговорност за све задатке који се односе на кориснике у самом систему.

У овом сервису, основни модел који је примарно коришћен носи име "User" и наслеђује "Entity" модел. "Entity" модел је основни модел који наслеђују све класе које желимо да користимо у табелама.

```

31 references
public class User : Entity
{
    [Required]
    8 references
    public string? Address { get; set; }
    [Required]
    12 references
    public string? FirstName { get; set; }
    [Required]
    12 references
    public string? LastName { get; set; }
    [Required]
    15 references
    public string? Email { get; set; }
    5 references
    public string? Password { get; set; }
    6 references
    public string? UserName { get; set; }
    7 references
    public string? Image { get; set; }
    5 references
    public DateTime? BirthDate { get; set; }
    10 references
    public Role Role { get; set; }
    9 references
    public UserStatus Status { get; set; }
    4 references
    public string? ActivateKey { get; set; }
}

```

Слика 3.4

У моделу за кориснике, поред поља која корисник уноси при регистрацији, имамо и следећа поља:

- Улога (Role): Ово поље чува информацију о групи којој корисник припада и на основу тога га ауторизујемо за приступ одређеним деловима апликације.
- Кључ за активацију (ActivateKey): Ово је насумично генерисан кључ који користимо при активацији корисничког налога путем е-поруке.
- Статус (Status): Ово поље чува информацију о тренутном стању корисничког налога, на пример, да ли је налог активан или неактиван.
- Слика (Image): Овде чувамо слику корисника.

Ова поља су део модела за кориснике и користе се за различите сврхе, укључујући аутентификацију, ауторизацију и персонализацију корисничког искуства (Слика 3.4).

Поред овог модела за складиштење, такође користимо и други DTO (Data Transfer Object) модел за пренос података између базе и клијентске апликације. Овај модел користимо како бисмо избегли кружно референцирање приликом серијализације у JSON формат за слање података (Слика 3.5).

```

namespace ECommerce.DAL.DTO.User.DataIn
{
    3 references
    public class RegisterUserDataIn
    {
        2 references
        public int? Id { get; set; }
        4 references
        public string? FirstName { get; set; }
        4 references
        public string? LastName { get; set; }
        1 reference
        public string? Username { get; set; }
        4 references
        public string? Address { get; set; }
        9 references
        public string? Email { get; set; }
        2 references
        public string? RoleId { get; set; }
        4 references
        public string? Password { get; set; }
        4 references
        public DateTime? BirthDate { get; set; }
        4 references
        public IFormFile? Image { get; set; }
    }
}

```

Слика 3.5

Функционалности овог сервиса укључују још и:

1. Регистрација корисника: Корисници могу да се региструју и при томе додају слике на сервер. Приликом додавања слика, слике се конвертују у формат преко FileStream и копирају на сервер. Сервер је конфигурисан у стартном фајлу тако да се понаша као сервер за рад са фајловима.
2. Измена података о кориснику: Корисници могу да измене своје информације.
3. Приказ свих продаваца: Систем омогућава приказ свих продаваца.
4. Верификација од стране админа: Администратор има могућност верификације корисника.
5. Пријава корисника: Корисници се могу пријавити на систем тако што унесу своје креденцијале. Креденцијали се шаљу на сервер као два стринга и врши се претрага базе података. Ако корисник са датим креденцијалима не постоји, систем враћа одговарајућу грешку. Пре провере лозинке, шифра се криптује ради безбедности и затим се упоређује са криптованом шифром из базе. Након успешне валидације, генерише се токен који се користи за аутентификацију на клијентској апликацији.
6. Верификација корисника путем линка са маила: Корисници могу верификовати своје налоге кликом на линк који се шаље на њихову мејл адресу. Ово се постиже упоређивањем мејл адресе и активационог кључа са вредностима у бази података. Уколико се пар вредности подударају, налог се активира.



## PRODUCT

Ова .NET WEB API апликација представља систем који функционише користећи трослојну архитектуру. Три слоја архитектуре су идентични као и на корисничком микросервису и врше операције над свим операцијама у вези са производима и поруџбинама.

Модел за производ (Слика 3.6) је основни модел у овом сервису и садржи сва неопходна поља која описују производ. То укључује информације које корисник уноси, референцу на категорију којој производ припада и идентификатор корисника који је додао тај производ. За референцирање корисника, користи се поље `int? CustomerId` унутар модела производа због тога што постоји одвојена табела за кориснике. У случају када је потребно добити информације о продавцу, извршава се HTTP захтев ка другом микросервису користећи методу `PostDataToApi` (користећи HTTP Client) и информације се претварају у модел који је намењен за враћање клијентској апликацији (`ProductDataOut`).

```
18 references
public class Product : Entity
{
    10 references
    public string Name { get; set; }
    7 references
    public double Price { get; set; }
    11 references
    public int Stock { get; set; }
    8 references
    public int? CustomerId { get; set; }
    5 references
    public string Description { get; set; }
    5 references
    public int? CategoryId { get; set; }
    4 references
    public Category Category { get; set; }
    5 references
    public string? Images { get; set; }
}
```

Слика 3.6

Поруџбина такође има свој модел (Слика 3.7) који садржи информације о достави и листу ставки које су наручене. Додатно, у моделу поруџбине постоји поље статуса које означава у којем стању се налази поруџбина.

```
13 references
public class Order : Entity
{
    public int? CustomerId { get; set; }
    public string? Name { get; set; }
    public string? Address { get; set; }
    public string? Phone { get; set; }
    public string? Comment { get; set; }
    public DateTime ShippingTime { get; set; }
    public double? Shipping { get; set; }
    public double? Total { get; set; }
    public DateTime OrderDate { get; set; }
    public ICollection<OrderItem> OrderItems { get; set; }
    public OrderStatus Status { get; set; }
}
```

Слика 3.7

Функционалности у овом сервису су подељене на једноставније и комплексније операције, и укључују:

1. Операције додавања, брисања и модификације производа: Корисници могу извршавати операције додавања нових производа, брисања постојећих и измене информација о производима.
2. Провера важности корпе: Систем прима корпу са корисничке апликације и проверава да ли је дошло до измена у међувремену у вези са производима у корпи. Ова провера је важна како би се осигурало да корпа садржи актуелне информације о производима.
3. Приказ поруџбина у односу на улогу корисника и статус поруџбине: Поруџбине се приказују корисницима у складу са њиховом улогом и статусом поруџбине. На пример, администратори могу видети све поруџбине, док обични корисници могу видети само своје поруџбине. Статус поруџбине означава у ком стању се налази поруџбина, на пример, да ли је поруџбина обрађена или је још у обради.
4. Креирање поруџбине: Корисници могу креирати нове поруџбине, што укључује додавање ставки у корпу, избор информација о достави и друге детаље поруџбине.

## API GATEWAY

API Gateway сервис послушкује захтеве од клијентске стране и обавља редирекцију тих захтева према микросервисним апликацијама (Слика 3.8). Ова апликација базирана је на Ocelot програмској библиотеци. Она представља лаку апликацију која садржи конфигурацију у виду JSON фајла, где су дефинисана правила за рутирање. Сва методе које се налазе у микросервисима доступне су преко јединствених адреса. Појмови "Downstream" и "Upstream" се користе да означе адресу за методе микросервиса и адресу на којој је тај микросервис доступан када се позива преко главне апликације.

Поред прослеђивања захтева, API Gateway сервис такође прослеђује токене и врши логовање свих захтева у текстуални фајл. Ово омогућава практичну аутентикацију и следеће активности на систему.

```
{
  "DownstreamPathTemplate": "/api/product/save",
  "DownstreamScheme": "https",
  "DownstreamHostAndPorts": [
    {
      "Host": "localhost",
      "Port": 7220
    }
  ],
  "UpstreamPathTemplate": "/api/product/save",
  "UpstreamHttpMethod": [ "Post" ],
  "AuthenticationOptions": {
    "AuthenticationProviderKey": "Bearer",
    "AllowedScopes": []
  }
}
```

Слика 3.8

## КОРИСНИЧКА АПЛИКАЦИЈА

Клијентска апликација је организована у више компоненти које су груписане по функционалности и типу ентитета којим раде. Групе компоненти обухватају:

1. Производи: Ова група компоненти је одговорна за приказ и манипулацију подацима о производима у апликацији. Корисници могу прегледати производе, додавати нове или вршити измене на већ постојећим производима.
2. Корисници: Ова група компоненти обухвата функционалности за рад са корисницима. Овде корисници могу видети информације о својем налогу, изменити своје податке и обављати сличне операције.
3. Почетна страница: Ова компонента представља почетну страницу апликације и може садржавати информације и функционалности које се приказују при почетку коришћења апликације (Слика 3.9).
4. Поручбине: Група компоненти за поручбине омогућава корисницима да виде своје поручбине, прегледају детаље поручбина и управљају њима, као и да обављају операције попут додавања ставки у корпу и обрађивања поручбина (Слика 4.0, 4.1).

Свака од ових група компоненти има своју специфичну функционалност у складу са ентитетима са којима раде.



Слика 3.9

ORDERS

NEW ORDERS

Id	Customer	Shipping Data	Comment	Total	Shipping	Date	Status	No orders	
1	Strahinja Gojkovic (3)	Andrija Musić Bulevar Oslobođenja 65 0648375702	dgfdgdfgfdg	2323 RSD	250 RSD	<div>Order Date 2023-09-15T13:17:08.4841853</div> <div>Shipping Date 2023-09-15T21:17:08.4840184</div> <div>Shipping has already occurred.</div>	SHIPPED	1	<div></div>

1

SHIPPING DETAILS

First Name

Marko

Last Name

Samardzija

Mobile No

Address

Bulevar Oslobođenja 65

Comment

SUBMIT

ORDER DETAILS

Andrija

1 x 2323 RSD

PRODUCTS	SHIPPING	TOTAL
2323 RSD	250 RSD	2573 RSD

Обраћање серверу (Слика 4.2) је одвојено у сервисне структуре, где сваки сервис поседује методе које позивају методе на серверској страни. У случају грешке, систем врши обраду исте. Важно је да систем обавештава клијента о стању захтева како би корисник знао у ком стању се налази. Обавештења се често приказују кориснику у виду кружића за читавање и исписа поруке која се појављује у горњем левом углу екрана.

За комуникацију са сервером користи се Axios библиотека, која омогућава обављање HTTP захтева. С друге стране, сервиси враћају Observable<T>, где "T" представља очекивани тип повратне вредности. Овај начин комуникације омогућава асинхрону и ефикасну обраду захтева и одговора између клијента и сервера.

```
const createProduct = async (productData) => {
  try {
    const response = await axiosInstance.post('/product/save', productData);
    if (response.status === 200 || response.status === 'OK') {
      if (response.data.message !== '') {
        toast.success(response.data.message);
      }
    } else if (response.status === 402) {
      toast.error(response.data.message);
    } else {
      toast.error('An error occurred.');
```

Слика 4.2

## ПРЕДЛОЗИ ЗА ДАЉА УСАВРШАВАЉА

Апликација Онлине Продавница, поседује многе функционалности које и најбоље веб проданице поседују. Поред тога, доста ствари се може додати и допунити као би апликација била још боља и успешнија. Ово су само неке од њих:

- Дозволити корисницима да додатно оцењују производе исказивањем мишљења и додавањем оцена.
- Било би корисно имплементирати могућност превода странице на различите језике које корисници могу да изаберу.
- Може се увести одговор на случај заборављене лозинке.
- Оптимизација кода да би апликација била бржа и ефикаснија у раду.
- Додати тестове за све делове апликације
- Увести слање порука тј. комуникацију између корисника, продавца и администратора.
- Увођење различите погодности за купце, на пример попусти и купони.
- Комуникацију између микросервиса с HTTP Rest API би требало пребацити на GRPC[7], што је напреднија технологија прилагођена за комуникацију између веб апликација и често бржа. Међутим, ово није применљиво за комуникацију са корисничком апликацијом јер интернет претраживачи за сада не подржавају ову технологију.

## ЛИТЕРАТУРА

- [1] <https://visualstudio.microsoft.com/vs/getting-started/>
- [2] <https://code.visualstudio.com/docs>
- [3] <https://learn.microsoft.com/en-us/sql/ssms/sql-server-management-studio-ssms?view=sqlserver-ver16>
- [4] Alex Banks. Schmidt, Learning React: Functional Web Development with React and Redux, 2017
- [5] <https://learn.microsoft.com/en-us/aspnet/core/web-api>
- [6] <https://docs.microsoft.com/en-us/dotnet/architecture>, Ocelot API Gateway
- [7] <https://unnieayilliath.com/2023/02/19/using-http-3-in-grpc> , GRPC