



УНИВЕРЗИТЕТ У НОВОМ САДУ  
ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА У  
НОВОМ САДУ



Магдалена Шљивић, ПР142/2019

## **ВЕБ ПРОДАВНИЦА**

### **ПРОЈЕКАТ**

- Примењено софтверско инжењерство (ОАС) -

Нови Сад, 15.09.2023.

## 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

Веб апликација је рачунарски програм који користи веб прегледнике и веб технологију за извршавање задатака путем интернета.[1] Веб апликација користи комбинацију скрипти на *frontend*-у и скрипти на *backend*-у. *Backend* управља захтевима и упитима самог кода апликације и заслужан је за прикупљање и складиштење података, док *frontend* омогућава корисницима веб апликације приказ података, која уз помоћ дизајна постаје корисницима интуитивна за употребу. Често је повезана са базом података како би се корисницима апликације пружило интерактивно искуство.

Микросервиси су аутономни сервиси, који раде сви заједно. Погодно је да буду што једноставнији приликом извршавања, да имплементирају мали сегмент пословног процеса и да може брзо да обради захтев и врати одговор на захтев.

Микросервисна архитектура је еволуирала на два главна извора: развој монолитних апликација користећи образац за слојевиту архитектуру и развој дистрибуираних апликација кроз образац за сервисно-оријентисану архитектуру. Микросервисна архитектура базирана на *API REST* топологији се користи за веб сајтове, који излажу мали индивидуални сервис преко неког *API*-ја. Пример коришћења ове топологије обухватају неке од заједничких *cloud REST* веб сервиса као што су *Yahoo*, *Google* и *Amazon*.

Карактеристике микросервиса:

- Сервиси као компоненте
- Организација пословних јединица
- Развој производа
- Микросервисна комуникација
- Децентрализација
- Аутоматизација
- Отпорност на грешке

Предности микросервиса:

- Подела сервиса
- Постављање апликације
- Разноликост технологије
- Изолација
- Капсулација
- Скалабилност
- Мониторинг
- Замењивост

Мане микросервиса се огледају у праћењу перформанси, асинхроној комуникацији, децентрализацији и расту броја микросервиса.[2]

За имплементацију комплексних и обимних софтверских решења, препоручује се употреба микросервисне архитектуре.

Циљ пројекта је унапређење основног пројекта из предмета Програмирање у Смарт Грид системима, у микросервисни систем са барем два микросервиса. Испред микросервисног система је постављен *Ocelot API gateway*, који захтеве са предње стране система рерутира ка индивидуалним микросервисима. Активности *API gateway*-а се логују у текстуалне фајлове и на конзолу. Сваки микросервис има своју базу података.

## ОПИС КОРИШЋЕНИХ ТЕХНОЛОГИЈА И АЛАТА

**Visual Studio 2022 Community[3]** – бесплатно интегрисано развојно окружење (*IDE*) које пружа алате за програмирање и развој софтвера. Омогућава програмерима да креирају различите врсте апликација. Има велик избор функционалности као што су преглед кода, дебаговање, управљање верзијама, интеграција са различитим програмским језицима и платформама. У овој апликацији, коришћен је за развој *backend* решења.

**ASP .NET Core 6.0 WEB API C#[4]** - *framework* за изградњу веб апликација у *C#* језику. Пружа ефикасан начин за развој *API*-ја за модерне апликације. Са богатим сетом алата и подршком за аутентификацију, ауторизацију и управљање токовима података, олакшава имплементацију сигурности. Пример коришћених пакета у апликацији су:

- *Automapper* - врши мапирање својстава између објеката класа
- *Google.Apis.Auth* - обезбеђује аутентификацију и регистрацију преко друштвене мреже Google mail
- *Ocelot* – пружа јединствену тачку уласка у систем који користи микросервисну архитектуру
- *Microsoft.AspNetCore.EntityFrameworkCore* – добавља ресурсе из базе података
- *Microsoft.AspNetCore.Authentication.JwtBearer* – креира и врши валидацију токена за препознавање корисника

**SQL Management Studio 19[5]** – алат који омогућава управљање и администрацију *SQL Server* база података. Пружа напредне функционалности за креирање, уређивање и извршавање административних задатака. Има кориснички интерфејс који омогућава лак приступ и управљање подацима у бази. У овој апликацији, коришћен је за развој решења базе података.

**Visual Studio Code[6]** – бесплатни интерфејс за развој који је лак, брз и направљен за програмирање различитих врста апликација. Нуди богат скуп функција, укључујући кодни преглед, интелигентно навођење, дебагирање, интеграцију са системима контроле верзија и широку подршку за екстензије. У овој апликацији, коришћен је за развој *frontend* решења.

**React[7]** – *JavaScript* библиотека за развој корисничког интерфејса у веб апликацијама. Користи архитектуру базирану на компонентама и тако омогућава лако коришћење и модуларност кода. Са својим виртуелним *DOM (Document Object Model)* приступом, *React* обезбеђује ефикасно ажурирање само промењених делова корисничког интерфејса, што доприноси брзини и перформансама апликације.

## ОПИС РЕШЕЊА ПРОБЛЕМА

Циљ ове апликације јесте омогућавање корисницима онлине куповину производа. Систем се састоји од корисника, производа и поруџбина. Корисник овог система може да буде: администратор, продавац или купац.

### Микросервисни систем

Бекенд страну апликације чине два микросервиса и један *API gateway*, дакле, постоји микросервис с фокусом на пословне процесе корисника система и микросервис с фокусом на пословне процесе артикала и поруџбина. Унутар *appsettings.json* фајла сваког микросервиса, дефинисан је конекциони стринг ка бази података.

У овом пројекту се користи *Ocelot* пакет за имплементацију *API gateway*-а. *API gateway* омогућава рерутирање захтева са предње стране апликације ка индивидуалним микросервисима. Кључна ствар при имплементацији је *ocelot.json* фајл, који је сачињен од две конфигурационе секције, а то су *Routes* и *GlobalConfiguration*. *Routes* представља низ објеката, на основу којих *Ocelot* зна како да третира *upstream* захтеве. Објекат унутар *Routes* садржи опис *host*-а, портова и рута микросервиса, као и *HTTP(S)* методе.

Затим, *GlobalConfiguration* све специфициране руте обједињује у једну глобалну руту, која ће комуницирати са предњом страном апликације.

```
{
  "GlobalConfiguration": {
    "BaseUrl": "https://localhost:5000",
    "ServiceDiscoveryProvider": {
      "Host": "localhost",
      "Port": 5000
    }
  },
  "Routes": [
    {
      "UpstreamPathTemplate": "/api/user/register",
      "UpstreamHttpMethod": [ "Post" ],
      "DownstreamPathTemplate": "/api/user/register",
      "DownstreamScheme": "https",
      "DownstreamHostAndPorts": [
        {
          "Host": "localhost",
          "Port": 5001
        }
      ]
    }
  ]
}
```

Слика 1. Имплементација *ocelot.json* фајла

Овако имплементиран *API gateway*, задовољава захтев да предња страна не комуницира директно са микросервисима.

Имплементација логовања рада *API gateway*-а у текстуалне фајлове, приказана је на слици испод.

```
builder.Configuration.SetBasePath(builder.Environment.ContentRootPath)
    .AddJsonFile("ocelot.json", optional: false, reloadOnChange: true)
    .AddEnvironmentVariables();
builder.Services.AddOcelot(builder.Configuration);
builder.Logging.ClearProviders();
builder.Logging.AddConsole();
builder.Logging.SetMinimumLevel(LogLevel.Information);
```

Слика 2. Имплементација логовања у текстуалне фајлове

### Непријављен корисник

Уколико корисник није пријављен или није регистрован, он нема могућност употребе апликације. Корисник се на систем региструје уносом својих личних података, одабиром типа корисника (купац или продавац). Корисник типа купац може да се региструје на систем на два начина: путем форме за регистрацију на апликацији и путем свог *Google mail* налога. Корисник типа продавац се региструје на систем путем форме за регистрацију, те нема приступ налогу све док верификацију налога не одобри

корисник типа администратор. Верификација налога се шаље у виду имејл поруке. Корисник типа администратор је унет директно у базу и не постоји регистрација на систем за овај тип корисника. Подаци се валидирају и на серверској и на клијентској страни. Лозинка мора да буде унесена као комбинација великих слова, малих слова и бројева и да њена дужина буде бар осам карактера. У супротном, корисник стиче увид у лоше осмишљеној лозинци тако што лабеле за унос и потврду шифре постану црвене боје.



The image shows a web form with two input fields. The first field is labeled 'Password' and the second is labeled 'Repeat Password'. Both fields have a red border, indicating that the entered passwords are invalid according to the system's requirements.

Слика 3. Приказ уноса недовољно сигурне шифре

Уколико је лозинка испунила све валидације, криптује се са *SHA526* алгоритмом и касније се, приликом пријаве на систем, пореди са сачуваном криптованом вредношћу у бази података за датог корисника. Имплементација криптовања лозинке са *SHA526* алгоритмом се налази на слици испод.

```
4 references
public static string GenerateSaltedHash(byte[] password, byte[] salt)
{
    HashAlgorithm algorithm = new SHA256Managed();

    byte[] passwordWithSaltBytes = new byte[password.Length + salt.Length];

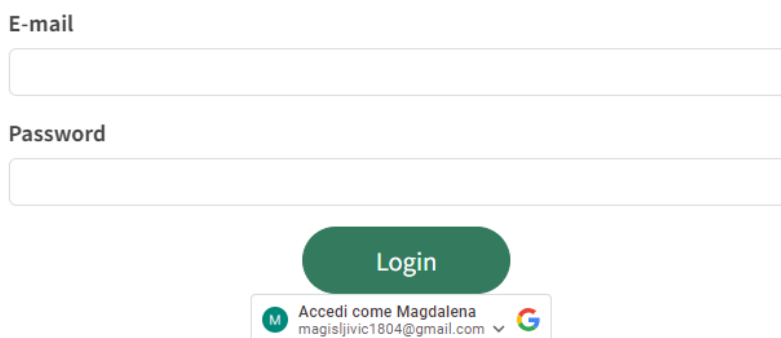
    for (int i = 0; i < password.Length; i++)
    {
        passwordWithSaltBytes[i] = password[i];
    }

    for (int i = 0; i < salt.Length; i++)
    {
        passwordWithSaltBytes[password.Length + i] = salt[i];
    }

    return algorithm.ComputeHash(passwordWithSaltBytes).ToString();
}
```

Слика 4. Криптовање шифре помоћу *SHA526* алгоритма

Пријава на систем се врши попуњавањем форме у коју корисник уноси важећи имејл и лозинку. Такође, корисник може да се пријави и преко Google mail налога. Корисник се пријављује на систем путем форме, која је приказана на слици испод.



The image shows a login interface. It has two text input fields labeled 'E-mail' and 'Password'. Below these fields is a green 'Login' button. At the bottom, there is a Google login option showing a profile picture, the name 'Accedi come Magdalena', and the email 'magisljivic1804@gmail.com'.

Слика 5. Приказ форме преко које се корисник пријављује на систем

Пријавом на систем, корисник је редиректован на страницу *dashboard*-а која садржи све опције одређеног типа корисника.

Корисник типа купац може да:

- Креира поруџбину
- Откаже поруџбину
- Види списак свих доступних артикала
- Види списак сопствених поруџбина
- Прати очекивано време када ће поруџбина бити достављена

Корисник типа продавац може да:

- Креира нови артикал
- Види списак својих артикала
- Види поруџбине које садрже његов артикал

Корисник типа администратор може да:

- Види све поруџбине
- Има увид у списак свих купаца и продавача
- Врши верификацију налога продавача (прихвата или одбија одређени налог)

Уколико је пријава успешна, метода враћа токен генерисан са *JwtBearer* и садржи *Claim*-ове, који се користе за ауторизацију. Дакле, систем добавља корисника са датом имејл адресом и проверава да ли је унесена тачна лозинка. Затим, проверава тип корисника и помоћу *JwtBearer* генерише токен за корисника, који се пријавио на систем. Имплементација пријаве корисника је приказана на слици испод.

```
User u = _uow.User.GetFirstOrDefault(u => u.Email == loginDTO.Email);

if (u != null)
{
    if (u.Password.SequenceEqual(PasswordHasher.GenerateSaltedHash(
        Encoding.ASCII.GetBytes(loginDTO.Password), u.Salt)))
    {
        List<Claim> claims = new List<Claim>();
        if (u.Role == SD.Roles.Buyer)
            claims.Add(new Claim(ClaimTypes.Role, "Buyer"));
        if (u.Role == SD.Roles.Seller)
            claims.Add(new Claim(ClaimTypes.Role, "Seller"));
        if (u.Role == SD.Roles.Admin)
            claims.Add(new Claim(ClaimTypes.Role, "Admin"));

        SymmetricSecurityKey secretKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(_secretKey.Value));
        var signinCredentials = new SigningCredentials(secretKey, SecurityAlgorithms.HmacSha256);
        var tokenOptions = new JwtSecurityToken(
            issuer: "https://localhost:5001",
            claims: claims,
            expires: DateTime.Now.AddMinutes(20),
            signingCredentials: signinCredentials
        );
        string tokenString = new JwtSecurityTokenHandler().WriteToken(tokenOptions);

        ProfileDTO p = _mapper.Map<ProfileDTO>(u);
        p.Token = tokenString;
    }
}
```

Слика 6. Имплементација пријаве корисника на систем

## Пријављени корисник

Пријављени корисник може да мења своје личне податке на свом налогу, али не може да мења који је тип корисника.

## Купац

Купац може да убацује артикле, које су креирали продавци, у своју корпу и затим изврши поруџбину артикала. Уколико је количина унетих производа мања од доступне количине производа, креира се поруџбина тако што корисник види списак артикала и кликом на дугме *Add* додаје артикал у своју корпу.

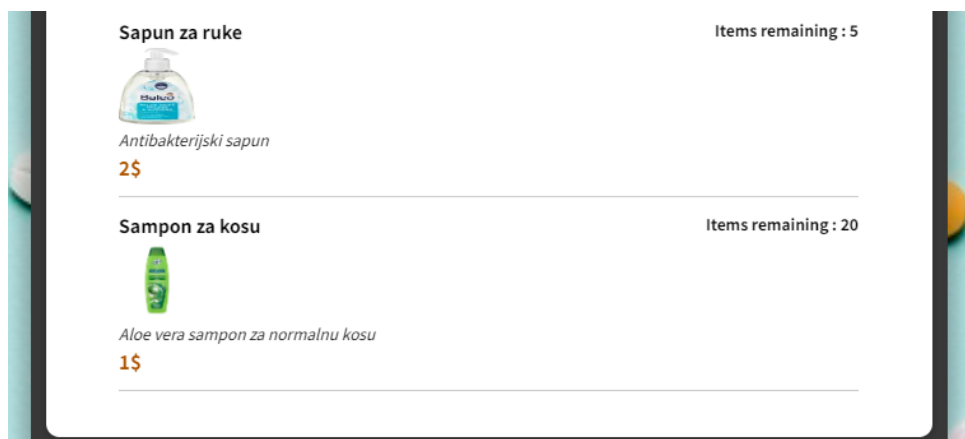


Слика 7. Додавање артикала у корпу

Поруџбину плаћа поузећем, а може и да откаже поруџбину. Купац види листу својих претходних поруџбина, које нису отказане.

## Продавац

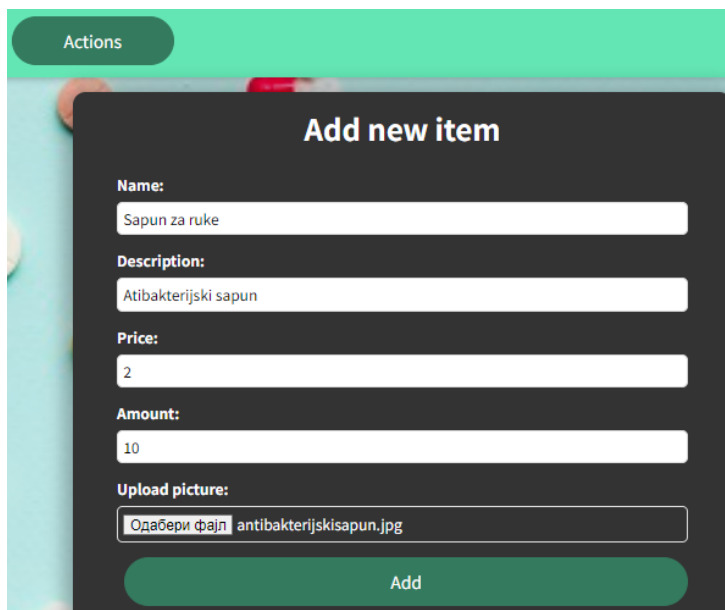
Основна функционалност продавца је да може креирати нове артикле. Продавац има увид у листу својих артикала.



Слика 8. Приказ артикала продавца



Кликом на дугме *Actions*, продавац прелази на прозор за креирање новог артикла. Продавац затим уноси податке о артиклу и додаје слику артикла. Кликом на дугме *Add* артикал се креира.

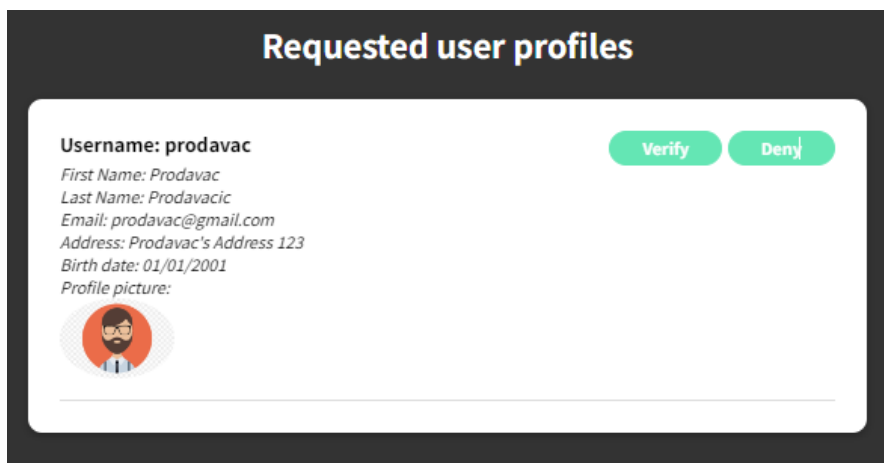


Слика 9. Форма за додавање артикла

Продавац може да види листу свих поруџбина, које садрже бар један његов артикал. Уколико купац откаже поруџбину продавац нема увид у отказану поруџбину.

### Администратор

Администратор је директно убачен у базу података и не постоји опција да се у систем региструје корисник типа администратор. Администратор се пријављује на систем уносом валидне имејл адресе и лозинке. Након извршене пријаве, администратор може да прегледа свој профил и да изврши измену сопствених података уколико жели. Такође, администратор има увид у списак свих поруџбина и списак невалидованих продаваца одакле кликом на дугме *Verify* или *Deny* може да изврши верификацију налога продавца.



Слика 10. Верификација продавца

Процес верификације налога продавца се огледа у томе да администратор може да прихвати или одбије продавца, а продавац добија информацију о верификацији на свој имејл.

## ПРЕДЛОЗИ ЗА ДАЉА УСАВРШАВАЊА

Предност оваквог система, односно веб продавнице, се огледа у томе што је понуда производа и могућност поручивања увек доступна и могућа. Такође, за продавце овакав начин пословања представља и вид уштеде, јер немају трошкове за одржавање пословних простора.

Једна од мана оваквог система јесте што корисник није сигуран у квалитет производа које наручује, као и да ли ће на његову адресу стићи баш ти производи које је наручио.

Правци даљег истраживања и унапређења апликације могу бити:

- Могућност дво-факторске аутентификације, како би пријава корисника на систем била безбеднија
- Могућност корисника да може видети на мапи где се његова поруџбина налази у том тренутку
- Контејнеризација апликације применом нпр. *Docker* алата, који у контејнерима смешта апликацију са свим пакетима од којих она зависи и значајно смањује величину апликације
- Могућност корисника да може означити жељене артикле као своје фаворите, како би при прављењу следеће поруџбине брже пронашао своје омиљене артикле
- Могућност да корисници оцене артикле, као и да дају коментар на сам артикал, како би остали корисници имали увид у квалитет артикла и поузданост продавца
- Могућност плаћања поруџбина банковном картицом, како би апликација била модернија
- Омогућити администратору да купцима, који не преузму поруџбину, додели казнене накнаде при наредној поруџбини
- Обезбедити респонзивни дизајн апликације, како би корисници несметано могли користити и прегледати веб апликацију, без обзира да ли користе десктоп, таблет или паметни телефон.

## ЛИТЕРАТУРА

- [1] <https://tehnoklik.ba/sta-su-web-aplikacije-i-cemu-sluzi/>
- [2] *Katedra za elektroenergetiku i primenjeno softversko inženjerstvo, Programiranje u Smart Grid sistemima, Predavanja - termin 11 i 12 (Mikroservisi), <https://www.eepsi.ftn.uns.ac.rs/group/programiranje-u-smart-grid-sistemima/custom>*
- [3] <https://visualstudio.microsoft.com/vs/getting-started/>
- [4] <https://learn.microsoft.com/en-us/aspnet/core/web-api>
- [5] <https://learn.microsoft.com/en-us/sql/ssms/sql-server-management-studio-ssms?view=sqlserver-ver16>
- [6] <https://code.visualstudio.com/docs>
- [7] *Alex Banks. Schmidt, Learning React: Functional Web Development with React and Redux, 2017*