



УНИВЕРЗИТЕТ У НОВОМ САДУ
ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА У
НОВОМ САДУ



Марко Миленковић, PR 2/2019

**Имплементација интернет продавнице
коришћењем микросервисне архитектуре**

ПРОЈЕКАТ

- Примењено софтверско инжењерство (ОАС) -

Нови Сад, 29.6.2023

Садржај

ОПИС РЕШАВАНОГ ПРОБЛЕМА	3
ОПИС КОРИШЋЕНИХ ТЕХНОЛОГИЈА И АЛАТА	4
ОПИС РЕШЕЊА ПРОБЛЕМА	5
Увод	5
Пријава и регистрација корисника	5
Почетна страна и профил корисника	6
Администратори	7
Продавци	8
Купци	9
Задња страна апликације и микросервисна архитектура	11
Ocelot API gateway	11
Микросервиси	12
ПРЕДЛОЗИ ЗА ДАЉА УСАВРШАВАЊА	15
ЛИТЕРАТУРА	16

ОПИС РЕШАВАНОГ ПРОБЛЕМА

Имплементирани систем представља апликацију за куповину на интернету, са акцентом на употреби микросервисне архитектуре на задњој страни.

Постоје три врсте корисника система, и то су Администратор, Продавац и Купац. Сходно томе, у апликацији постоје имплементиране функционалности аутентификације и ауторизације.

При покретању апликације, неаутентификован корисник види почетну страницу на којој је могуће улоговати се на систем или, уколико је потребно, направити нови налог. Регистрација се обавља попуњавањем личних података, и одабиром врсте корисника система.

Постоји могућност аутентификације и преко друштвене мреже, односно *Google*-а. Након пријаве, корисник може прегледати и по жељи ажурирати личне податке профила.

Новорегистровани продавац на систему прво чека верификацију профила од стране администратора, и до тог тренутка може само да прегледа информације о свом профилу. Уколико је успешно верификован, може да додаје нове производе у апликацију, уз могућност измене и брисања истих. Продавац има увид у новопристигле наруџбине, као и у историју наруџбина генерално.

Купац у систему може креирати нову наруџбину, додавањем жељених производа у корпу. Такође, може брисати или мењати количину производа у корпи, као и прегледати историју својих поруџбина у систему.

Није могуће регистровати се као администратор, његов налог већ постоји у систему. Он има могућност прегледања личних података продавца, односно може да прихвати или одбије пристигли захтев за регистрацију од стране продавца. Такође, администратор има увид у историју креираних поруџбина и може детаљно прегледати податке о свакој.

Задња страна имплементирана је поделом почетне монолитне апликације на два микросервиса: *ProductOrder* и *User*. Сваки микросервис има уобичајену слојевиту структуру *WebApi* апликације која обухвата слој контролера, интерфејсе односно сервисе за бизнис логику и репозиторијум за приступ бази података.[1]

Микросервиси не деле исту базу података. *ProductOrder* микросервис садржи функционалности везане за креирање и обраду производа и наруџбина, док се *User* микросервис бави корисничким профилима.

С обзиром да није дозвољено да се предња страна директно обрађа микросервисима, испред микросервисног система постављен је *Ocelot API gateway* који преусмерава пристигле захтеве са предње стране ка оном микросервису ка коме је то потребно. Преусмеравање се врши на основу дефинисаног *JSON* конфигурационог фајла који се читава у оквиру *gateway* пројекта.

Имплементиран је и механизам логовања, који прати и бележи активности *API gateway*-а у текстуалне фајлове и на конзолу. Исписују се и информације о пристиглом захтеву, односно пристиглој методи и путањи, а након извршеног захтева, исписује се и статусни код операције као одговор.

Целокупна комуникација у систему се одвија асинхронно, употребом *HTTP* протокола, а конкретни микросервиси могу и између себе да комуницирају онда када је то неопходно.

ОПИС КОРИШЋЕНИХ ТЕХНОЛОГИЈА И АЛАТА

- **Microsoft Visual Studio 2019** – платформа развијена од стране *Microsoft-a*, тзв. интегрисано програмско окружење, намењено развоју апликација. [2] Коришћена за развој задње стране.
- **Microsoft Visual Studio Code** – софтвер намењен уређивању изворног кода. Подржава многе корисне екстензије које додатно олакшавају кодирање. [3] Коришћен за развој предње стране.
- **React** – представља *JavaScript* библиотеку за развој корисничких интерфејса. [4] Акцент је на коришћењу тзв. компоненти, како би се спречило понављање истог кода, односно оствариле боље перформансе.
- **.NET** – framework коришћен уз програмски језик *C#*. Коришћен је на задњој страни за креирање *ASP.NET Core Web API* апликације уз *RESTful* сервисе.
- **Microsoft SQL Server Management Studio** – интегрисано окружење за руковање *SQL* инфраструктурама. Користи се у контексту приступа, уређења, администрације базе података. [5]
- **Ocelot** - *.NET API Gateway*, дизајниран за коришћење уз микросервисну архитектуру, где је потребно постојање јединственог улаза у систем. Представља *middleware* који манипулише *HTTP* захтевима. [6]
- **AutoMapper** – библиотека коришћена за упрошћавање процеса мапирања једног објекта у други на задњој страни апликације. Ради тако што аутоматски трансформише податке објекта једног типа, у објекат другог типа. Најбоље ради када су називи *property-ja* улазног и излазног објекта исти. [7]
- **Entity Framework Core** – лака, проширива, open source верзија *Entity Framework* технологије за приступ подацима. Служи за објектно-релационо мапирање које помаже и олакшава програмерима у раду са базама података. [8]
- **Bcrypt** – *NuGet* пакет који се користи за чување лозинки користећи компликоване криптографске алгоритме. Користи варијанту *Blowfish* енкрипционог алгоритма, што знатно смањује шансу хакерских упада. [9]
- **Google.Apis.Auth** – *NuGet* пакет у облику *Google API* клијента за рад са *Google* сервисима. Састоји се од компоненти који поред многих функционалности пружају и могућност аутентификације преко *Google-a*. [10]
- **Serilog** – *.NET* библиотека која пружа флексибилне могућности логовања у фајлове или на конзолу. [11] Коришћена за функционалности логовања активности *Ocelot gateway-a*.
- **JsonWebToken** – протокол коришћен за функционалности аутентификације у систему. Дефинише безбедни стандард за размену информација између две стране као *JSON* објекат. Информација се може верификовати, и потврдити, зато што је дигитално потписана. [12]
- **React-Toastify** – библиотека коришћена за форматирани приказ обавештења на корисничком интерфејсу на предњој страни. [13]
- **Axios** – *promise-based HTTP Client*, коришћен на предњој страни, за креирање и слање асинхроних захтева ка задњој страни. [14]

ОПИС РЕШЕЊА ПРОБЛЕМА

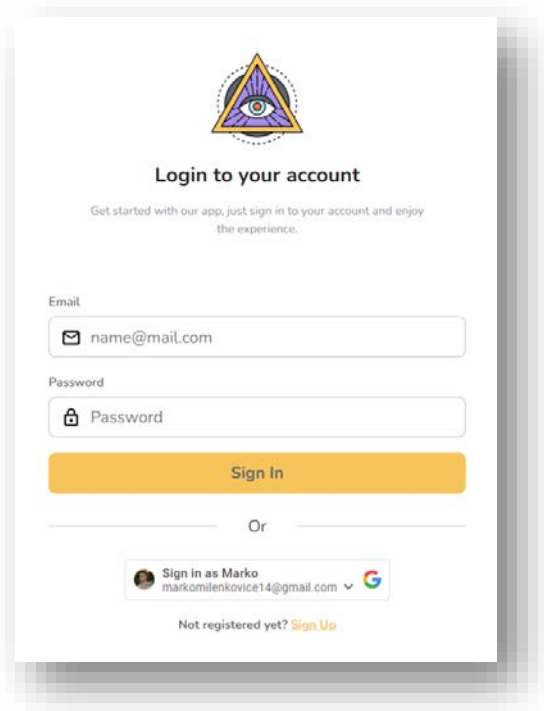
Увод

Имплементирана веб апликација развијена је као клијент-сервер архитектура, односно са постојањем предњег и задњег дела система. Предњи део је имплементиран у *React*-у, а задњи у *.NET* C#.

React, односно клијент, апликација излаже кориснички интерфејс преко ког корисници система користе интернет продавницу. Код је подељен у компоненте, што пружа додатну ефикасност рада апликације. Компоненте су организоване по фолдерима, сходно својој функционалности односно намени. Поред основних механизма аутентификације и ауторизације корисника, постоји и *Guarded Routes* компонента која за сваку путању у оквиру апликације проверава да ли јој се може приступити. Сервиси су такође организовани у посебним компонентама, а задужени су за комуникацију са серверским делом апликације коришћењем *Axios*-а. У основној *App.js* компоненти, коришћен је *RouterBrowser*, помоћу ког је омогућено рутирање, односно кретање кроз апликацију.[15] Такође, коришћена је *react-toastify* библиотека, како би се кориснику на уредан и форматиран начин приказала сва обавештења система. Модели у апликацији постоје у облику класа, како би се све пристигле информације са сервера правилно мапирале на одговарајуће објекте.

Пријава и регистрација корисника

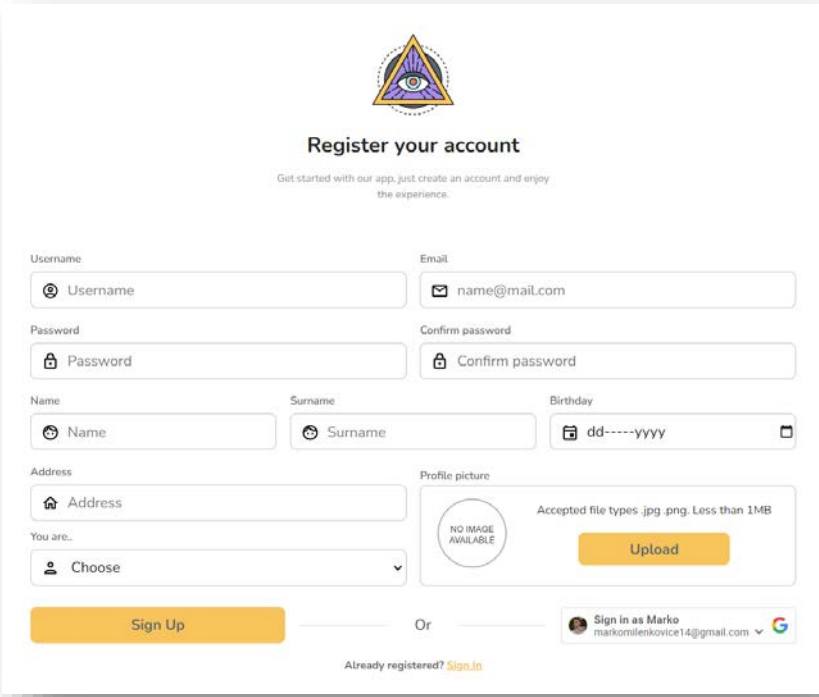
При покретању апликације, отвара се страница за пријаву на систем. (Слика 1) Корисник се може пријавити уношењем своје имејл адресе и лозинке, или одабиром опције за пријаву помоћу *Google*-а. Кликом на дугме „*Sign In*”, подаци се шаљу на сервер, где се одвија аутентификација унетих креденцијала. Уколико је аутентификација успешна, формира се *JWT* токен који се враћа назад клијенту као одговор за успешну пријаву и који се користи за приступ свим страницама. Након тога, корисник бива аутоматски преусмерен на почетну страну апликације – *Dashboard*. (Слика 3). Пријава преко *Google*-а функционише слично, разлика је у томе што проверу креденцијала обављају *Google* сервиси.



Слика 1. Форма за пријаву на систем

Уколико корисник нема налог на систему, може да направи исти кликом на опцију „*Sign Up*” са Слика 1. У том случају, отвара се заглавље за регистрацију (Слика 2) где се корисник попуњавањем личних података може регистровати на систем.

Лични подаци укључују корисничко име, имејл адресу, лозинку, име, презиме, датум рођења, адресу и слику. Такође, бира се и врста корисника система, односно купац или продавац. Сви подаци морају бити попуњени и тачни, за шта је задужена верификација форме система. Уколико то жели, корисник се може регистровати и преко друштвене мреже *Google*-а. Аутентификацијом *Google* креденцијала, форма бива аутоматски попуњена познатим информацијама корисниковог профила. Како би се довршила регистрација, неопходно је попунити форму до краја подацима који недостају. На овај начин корисник се може регистровати само као купац.



The registration form is titled "Register your account" and includes a subtext: "Get started with our app, just create an account and enjoy the experience." The form contains the following fields and options:

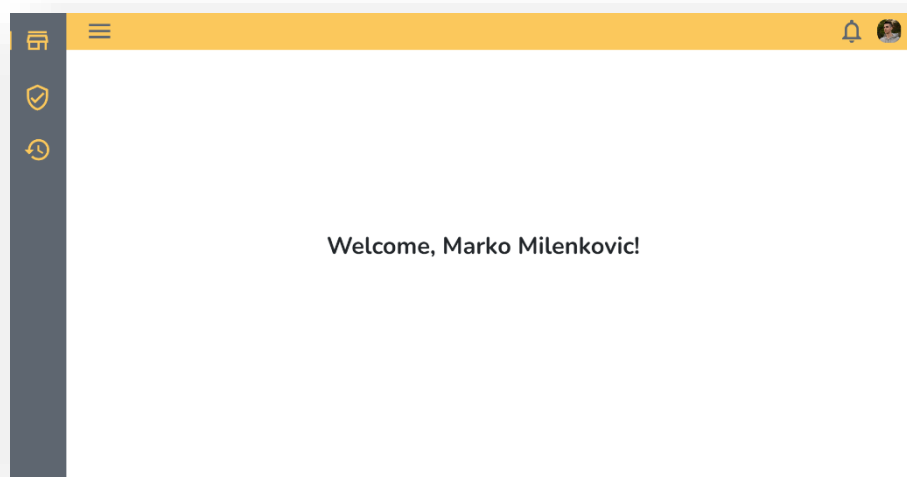
- Username:** Input field with a user icon.
- Email:** Input field with an email icon, containing "name@mail.com".
- Password:** Input field with a lock icon.
- Confirm password:** Input field with a lock icon, containing "Confirm password".
- Name:** Input field with a user icon, containing "Name".
- Surname:** Input field with a user icon, containing "Surname".
- Birthday:** Date picker field containing "dd-----yyyy".
- Address:** Input field with a house icon, containing "Address".
- Profile picture:** A circular placeholder with "NO IMAGE AVAILABLE" and an "Upload" button. Text below indicates "Accepted file types: .jpg .png. Less than 1MB".
- You are..:** A dropdown menu with a person icon and the text "Choose".

At the bottom, there is a large orange "Sign Up" button, an "Or" separator, and a "Sign in as Marko" button with a profile picture and email "markomilenkovic14@gmail.com". A link "Already registered? Sign in" is located at the very bottom.

Слика 2. Форма за регистрацију на систем

Почетна страна и профил корисника

Dashboard, као почетну страну апликације, виде сви успешно пријављени корисници. (Слика 3) Састоји се од навигационог менија, и главног простора у средини. На левој страни менија, приказане су могуће акције једног корисника на систему, и оне зависе од конкретне врсте корисника. На конкретној слици приказан је изглед почетне стране администратора. Сви корисници имају могућност прегледања и измене података свог профила. Одабиром такве опције, у горњем десном углу, отвара се форма попуњена подацима једног профила. (Слика 4)



Слика 3. Изглед почетне стране - *Dashboard*

Account settings

Username: Email:

New Password (optional): Confirm new password:

Name: Surname: Birthday:

Address: Profile picture:

To save changes, enter current password:

Слика 4. Изглед форме за преглед и измену корисничког профила

Могуће је променити све податке профила осим врсте корисника. Како би се сачувале промене, потребно је унети тренутну лозинку као верификацију идентитета.

Администратори

У конкретном решењу, администратор је предефинисани профил у бази података, односно није могуће регистровати се као исти. Он има могућност прегледања профила регистрованих продаваца, као и функционалност верификације њихових профила. Када се региструје на систем, да би користио своје функционалности, продавац прво мора да чека верификацију профила од стране администратора. Захтев, односно информације о профилу, приказане су у облику картица (Слика 5), са опцијама да се верификација прихвати или одбије. Након окинуте акције, продавцу се шаље мејл са информацијом о статусу верификације његовог профила.

Marko Milenkovic

Слика 5. Картица за верификацију профила продавца

Администратори такође имају увид у целокупну историју наруџбина у систему, уз могућност детаљног прегледа сваке. Листа наруџбина је форматирана у облику табеле (Слика 6), где се кликом на дугме на крају реда може видети детаљни приказ конкретне наруџбине.

ID	Buyer	Address	Comment	Order Placed On	Delivery On	Price	Status	
1	Bojan Bojanovic	Cara Dusana 50, 21000 Novi Sad	Komentar narudzbine.	25 June 2023 23:10:24	29 June 2023 1:10:24	8700	Cancelled	Show details
7	Bojan Bojanovic	Cara Dusana 50, 21000 Novi Sad	Komentar narudzbine.	26 June 2023 0:6:21	28 June 2023 4:6:21	5200	Completed	Show details
8	Bojan Bojanovic	Cara Dusana 50, 21000 Novi Sad	Komentar narudzbine.	28 June 2023 23:53:19	30 June 2023 6:53:19	3900	Completed	Show details
9	Bojan Bojanovic	Cara Dusana 50, 21000 Novi Sad	Komentar narudzbine.	29 June 2023 7:29:10	1 July 2023 2:29:10	5000	Completed	Show details
10	Andjela Milenkovic	Cara Dusana 50, 21000 Novi Sad	Komentar narudzbine.	2 July 2023 18:13:41	5 July 2023 5:13:41	4400	Completed	Show details
11	Bojan Bojanovic	Cara Dusana 50, 21000 Novi Sad	Komentar narudzbine.	6 July 2023 18:36:12	10 July 2023 6:36:12	2800	Completed	Show details

Слика 6. Табела нарудбина

Детаљан приказ поруџбине форматиран је тако да се виде информације као што су подаци о купцу, листа купљених производа, укупна цена, адреса, коментар, статус, и датум поруџбине односно датум доставе. Такође, кликом на дугме „Seller Info” могу се видети подаци о продавцу тог конкретног производа у горњем десном углу. (Слика 7)

Thanks for your order, Bojan!

Bojan Bojanovic

Cara Dusana 50, Novi Sad 21000

bojan@lx.com

@bojanb

Milos Milosevic

Cara Dusana 50, Novi Sad 21000

milos@mx.com

@milosm

Sunglasses

Introducing the "Radiance Shades" - sleek sunglasses that combine style and sun protection. With a foldable frame and premium lenses, they offer both fashion and functionality in one must-have accessory.

Seller info

Price : 200rsd

Quantity : 2

Total : 400rsd

LED portable light

Introducing the "LuminaLite" - a sleek and portable LED light that delivers powerful illumination on the go. Illuminate your world with ease and brilliance using this compact lighting solution.

Seller info

Price : 500rsd

Quantity : 5

Total : 2500rsd

iPhone 14 case

Introducing the "SleekShield" iPhone case - a perfect blend of style and protection. With its slim design and durable construction, this case keeps your iPhone safe from everyday bumps and scratches while adding a touch of elegance to your device.

Seller info

Price : 200rsd

Quantity : 5

Total : 1000rsd

Black bag

Introducing the "Edge Noir" bag - a striking accessory that combines timeless elegance with a vibrant pop of color. With its sleek black design and eye-catching orange edges, this bag is a perfect blend of sophistication and contemporary style.

Seller info

Price : 700rsd

Quantity : 6

Total : 4200rsd

Address: Cara Dusana 50, 21000 Novi Sad

Comment: Komentar narudzbine.

Status: Completed

Order placed on: 25 June 2023 23:10:24

Delivery expected on: 29 June 2023 1:10:24

Total: 8100rsd

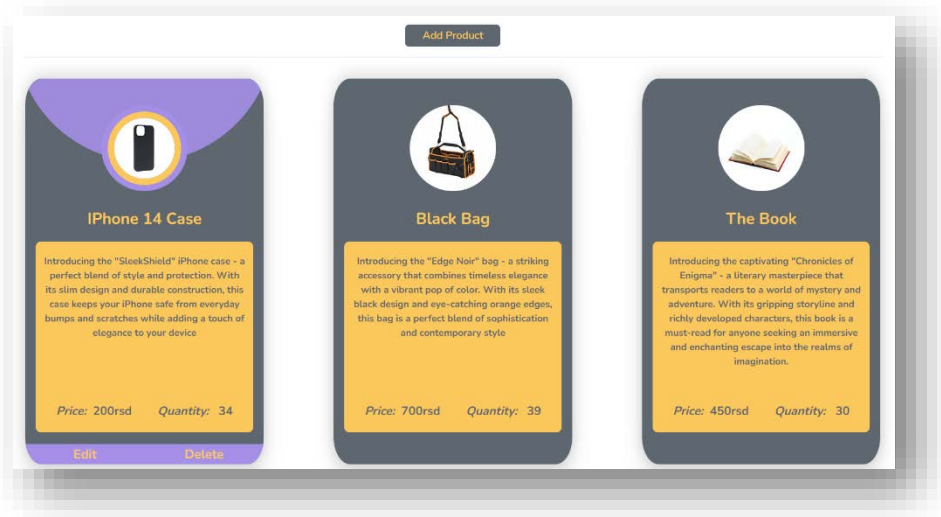
Delivery: 600rsd

Grand total: 8700rsd

Слика 7. Детаљан приказ једне наруџбине

Продавци

Као што је претходно речено у тексту, новорегистровани продавац на систему прво чека верификацију профила од стране администратора, и до тог тренутка може само да прегледа или мења информације о свом профилу. Након успешне верификације, продавцу се са леве стране навигационог менија (Слика 3) појављују функционалности које су до тада биле сакривене (онемогућене). Он има могућност прегледања свих својих производа у апликацији, односно измене или брисања истих. Такође, може да додаје нове производе у систем. Производи су на корисничком интерфејсу приказани кроз картице (Слика 8). На картицама се могу видети информације о производу као што су име производа, опис, цена, количина и слика производа. Кликком на дугме „Edit” отвара се форма за измену (Слика 10), попуњена подацима тог конкретног производа. Кликком на дугме „Delete”, активира се акција брисања.



Слика 8. Приказ производа једног продавца

The 'Add product' form contains the following fields and buttons:

- Product name:** A text input field with a placeholder icon.
- Price:** A text input field with a currency icon.
- Quantity:** A text input field with a unit icon.
- Description:** A text area with a placeholder icon.
- Profile picture:** A circular placeholder with 'NO IMAGE AVAILABLE' text, an 'Upload' button, and a note: 'Accepted file types: .jpg, .png, Less than 1MB'.
- Add product:** A large orange button at the bottom.

Слика 9. Форма за додавање производа

The 'Edit product' form contains the following fields and buttons:

- Product name:** A text input field with a placeholder icon.
- Price:** A text input field with a currency icon, containing the value '450'.
- Quantity:** A text input field with a unit icon, containing the value '30'.
- Description:** A text area with a placeholder icon, containing the text: 'Introducing the captivating "Chronicles of Enigma" - a literary r'.
- Profile picture:** A circular placeholder with a book icon, an 'Upload' button, and a note: 'Accepted file types: .jpg, .png, Less than 1MB'.
- Edit product:** A large orange button at the bottom.

Слика 10. Форма за измену производа

Кликом на дугме „Add Product” отвара се форма за додавање новог производа (Слика 9). При додавању сви подаци морају бити попуњени и тачни. Продавац има могућност увида у новопристигле наруџбине, као и у историју наруџбина генерално. Новопристигле наруџбине представљају оне које још увек нису достављене. Историја наруџбина обухвата претходне наруџбине односно оне које су достављене или отказане. Приказ ове две функционалности омогућен је табеларно слично као и код врсте корисника администратор (Слика 6 и 7). Разлика је у томе што се продавцу приказују оне наруџбине које садрже бар један његов производ, и при томе у листи наручених производа (Слика 7) види само те своје производе.

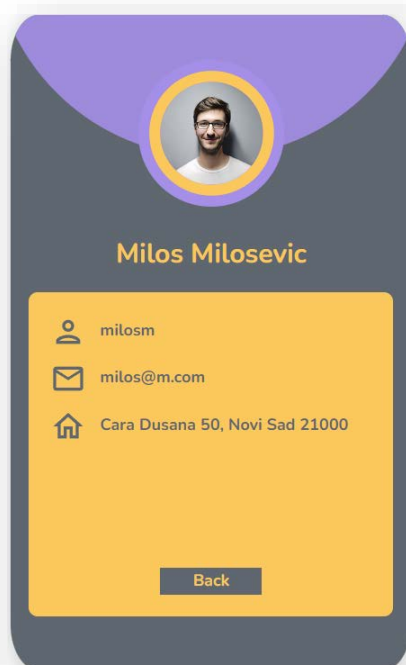
Купци

Врста корисника купац има две главне функционалности у систему, а то су креирање поруџбина и увид у историју истих. Купцу су производи поређани по картицама слично продавцима (Слика 11), где се за сваки производ може видети његов назив, опис, цена и слика. Кликом на дугме „Add to cart” производ се према постављеној количини, са леве стране од дугмета, додаје у корпу. Постоји и дугме „Seller Info” које се понаша као toggle, и које исту картицу попуњава подацима продавца тог производа (Слика 12). Купац може прегледати корпу, и евентуално изменити количину или обрисати неки додати производ. Кликом на „Checkout” отвара се заглавље које приказује тренутно стање поруџбине (Слика 13). Цена поруџбине израчуната је на основу цене производа и његове количине, као и цене доставе која је фиксна за једног продавца, а повећава се за више њих.

Додавањем коментара и адресе за доставу, поруџбина се може креирати кликом на дугме „Place order”. Генерисано време за доставу је насумично, и корисник ће као одговор од сервера добити информацију о тачном датуму и времену доставе. Такође, купац има увид у историју својих наруџбина које су организоване у оквиру табеле слично администраторима и продавцима (Слика 6). Разлика је у томе што купац има могућност да откаже наруџбину сат времена након њеног креирања. Уколико је откаже, та наруџбина се више неће приказивати у табели, а количина наручених производа ће бити враћена на старо стање у бази.




Слика 11. Картица производа



Слика 12. Приказ информација о продавцу производа

Checkout



Sunglasses

Introducing the "Radiance Shades" - sleek sunglasses that combine style and sun protection. With a bold black frame and premium lenses, they offer both fashion and functionality in one must-have accessory.

Price : 200rsd
Quantity : 3
Total : 600rsd

Change quantity
Remove

Comment

Komentar narudzbine.

Address

Adresa narudzbine.

Place order

Total: 600rsd
Delivery: 300rsd
Grand total: 900rsd

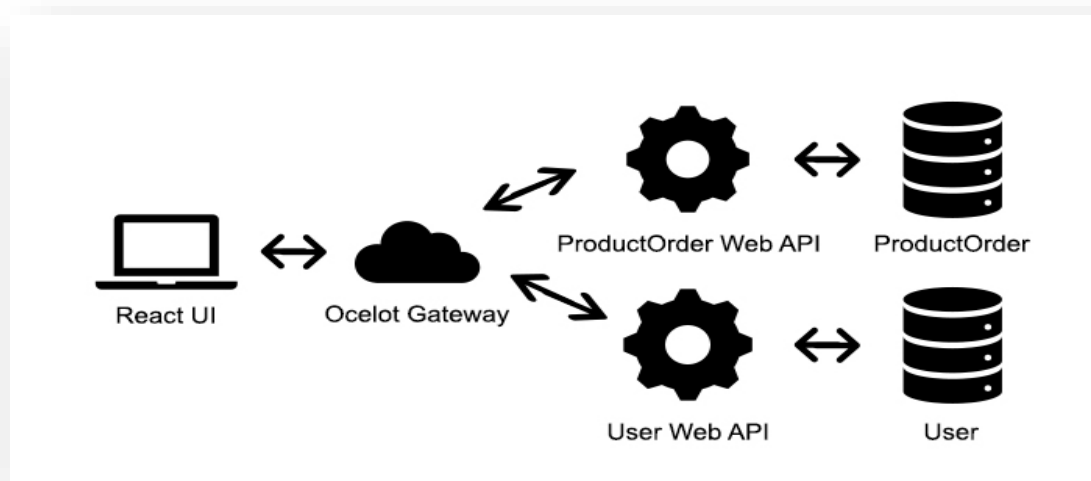
Слика 13. Заглавље за потврду поруџбине

Задња страна апликације и микросервисна архитектура

У контексту развоја апликација, монолитна архитектура подразумева апликацију која је изграђена као једна целина. У њој се налазе све функционалности система, као што су обрада *HTTP* захтева, извршавање пословне логике, читање и ажурирање података из базе и генерисање одговора клијенту. Такав начин развоја је природан, једноставан и лако разумљив, али са собом повлачи и одређене мане. Уколико дође до потребе за скалирањем, врши се скалирање целе апликације уместо само делова који захтевају више ресурса. Такође, са порастом монолитне апликације, управљање односно одржавање може постати компликовано јер све функционалности деле исти код и податке.[16]

Наведене мане монолитне архитектуре могу се отклонити преласком на микросервисну архитектуру. Ради бољег управљања ресурсима, делови апликација се издвајају и постају мање независне компоненте. Свака компонента има јединствену функционалност система и ради као засебна апликација коју називамо микросервисом.

У конкретном решењу, апликација је подељена на два микросервиса (*ProductOrder* и *User*), од којих сваки има своју базу података, док се комуникација између предње стране и микросервиса одвија посредством *Ocelot API Gateway*-а (Слика 14).



Слика 14. Приказ архитектуре система решаваног проблема

Ocelot API gateway

На улазу система задње стране, постављен је *Ocelot API* који све пристигле захтеве са предње стране, прослеђује ка одређеном микросервису. Функционалност прослеђивања захтева одређена је конфигурационим *JSON* фајлом (Слика 15), који се учитава у оквиру самог пројекта.

```
{
  "UpstreamPathTemplate": "/api/auth/register",
  "UpstreamHttpMethod": [ "POST" ],
  "DownstreamScheme": "http",
  "DownstreamHostAndPorts": [
    {
      "Host": "localhost",
      "Port": 2554
    }
  ],
  "DownstreamPathTemplate": "/api/auth/register"
},
```

Слика 15. Пример изгледа конфигурационог фајла *Ocelot API gateway*-а

У оквиру конфигурационог фајла налазе се блокови кода испуњени одређеним кључ-вредност паровима. У конкретном решењу, сваки блок се односи на једну трансакцију, односно препознавање и прослеђивање једног *HTTP* захтева. Међу особинама уочавају се два битна појма: *Upstream* и *Downstream*.

Upstream вредности се односе на препознавање долазног захтева. *UpstreamPathTemplate* наводи шаблон путање, док *UpstreamHttpMethod* означава врсту *HTTP* захтева. Обе особине се морају поклапати са пристиглим захтевом, како би се исти проследио.

Downstream вредности одређују параметре по којима се врши прослеђивање једног захтева. *DownstreamScheme* представља шему коју ће *API* користити при прослеђивању, *HTTP* или *HTTPS*. *DownstreamPathTemplate* наводи путању којој ће захтев бити прослеђен. *DownstreamHostAndPorts* обухвата име хоста (или *IP* адресу) и порт на основу којих ће се прослеђивање извршити.

Овај *API* такође поседује функционалност логовања односно праћења активности. Прате се информације о времену пристиглог захтева, путањи, као и о узвраћеном одговору клијенту. Подаци се бележе у оквиру посебних фајлова на дневном нивоу, као и на конзолу, на основу дефинисане *LoggingMiddleware* класе у пројекту.

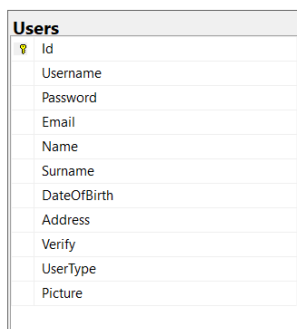
Микросервиси

Микросервиси су креирани као *ASP.NET Core Web API* пројекти, који функционишу као *RESTful* сервиси и који се извршавају на *localhost* машини. У сваком је присутна трослојна архитектура која подразумева постојање слоја контролера, слоја сервиса и слоја података. Слојеви међусобно комуницирају преко интерфејса, и то тако да се међусобно не прескачу, и да нижи слој не види виши. Комуникација се одвија асинхроно.

За пристигле захтеве, од стране *API gateway*-а, слој контролера представља улазну тачку. Све функционалности које су логички повезане, постављене су у оквиру истог контролера. Сваки контролер, користећи интерфејс, комуницира са одговарајућим слојем сервиса. Овакав механизам назива се инјекција зависности. Животни циклуси зависности дефинисани су као *Scoped*, што значи да ће све зависности бити инјектоване са истом референцом у оквиру једног *HTTP* захтева.

Слој сервиса је задужен за бизнис логику апликације. Када је потребно приступити бази он се може обратити слоју података (репозиторијума). Такође, у овом слоју, микросервис је у могућности да комуницира са другим микросервисом онда када је то потребно. Таква функционалност имплементирана је употребом уграђене *HttpClient* класе која служи за слање *HTTP* захтева односно примање одговора.

Слој репозиторијума излаже методе за директан приступ бази података у систему. База података имплементирана је употребом *Entity framework*-а, и коришћењем *code-first* технике. На основу дефинисаних модела (Слика 16), и конфигурација истих (Слика 17), генерисане су базе *ProductOrder* (Слика 18) и *User* (Слика 19).



Слика 18. Релациони модел базе *User*



Слика 19. Релациони модел базе *ProductOrder*

```

19 references
public class OrderedProduct
{
    1 reference
    public long Id { get; set; }
    13 references
    public long OrderId { get; set; }
    2 references
    public Order Order { get; set; }
    11 references
    public long ProductId { get; set; }
    2 references
    public Product Product { get; set; }
    6 references
    public int OrderedQuantity { get; set; }
}

```

Слика 16. Пример једног модела

```

0 references
public class OrderedProductConfiguration : IEntityTypeConfiguration<OrderedProduct>
{
    0 references
    public void Configure(EntityTypeBuilder<OrderedProduct> builder)
    {
        builder.HasKey(op => new { op.OrderId, op.ProductId });

        builder.Property(x => x.Id).ValueGeneratedOnAdd();

        builder.HasOne(op => op.Order)
            .WithMany(o => o.OrderedProducts)
            .HasForeignKey(op => op.OrderId)
            .OnDelete(DeleteBehavior.Restrict);

        builder.HasOne(op => op.Product)
            .WithMany(o => o.OrderedProducts)
            .HasForeignKey(o => o.ProductId)
            .OnDelete(DeleteBehavior.Cascade);
    }
}

```

Слика 17. Пример једне конфигурације

Конкретан пример модела са слике 16, представља табелу која повезује табеле *Product* и *Order*, а које се налазе у оквиру микросервиса *ProductOrder*. Он обухвата сву логику и функционалности које се тичу наруџбина и производа. Неке од њих јесу добављање историје наруџбина, креирање наруџбина, додавање производа, измена и брисање производа итд.

Микросервис *User* комуницира са истоименом базом, бави се корисничким профилима као и аутентификационим функционалностима апликације. Подржава пријаву на систем, регистрацију, добављање и измену корисничких профила али и верификацију корисника.

Генерисањем базе, у апликацији се појављује *DbContext* који садржи намапиране табеле у оквиру *DbSet*-ова, а који се користи за директан приступ бази у репозиторијумима (Слика 18).

```

14 references
public class ProductOrderDbContext : DbContext
{
    3 references
    public DbSet<Product> Products { get; set; }
    3 references
    public DbSet<Order> Orders { get; set; }
    2 references
    public DbSet<OrderedProduct> OrderedProducts { get; set; }
    0 references
    public ProductOrderDbContext(DbContextOptions options) : base(options)
    {
    }

    0 references
    protected override void OnModelCreating(ModelBuilder modelBuilder)
    {
        base.OnModelCreating(modelBuilder);
        modelBuilder.ApplyConfigurationsFromAssembly(typeof(ProductOrderDbContext).Assembly);
    }
}

```

Слика 18. Креирани DbContext базе података ProductOrder

Сваки микросервис има *Data Transfer* објекте (класе) помоћу којих се врши размена података (Слика 19). Користећи библиотеку *Automapper*, подржано је ефективније мапирање модела на *DTO* објекте и обрнуто.

```

4 references
public class CreateOrderDto
{
    3 references
    public List<ProductItem> ProductList { get; set; }
    0 references
    public string Comment { get; set; }
    0 references
    public string Address { get; set; }
    0 references
    public float ProductsPrice { get; set; }
    0 references
    public float DeliveryPrice { get; set; }
    0 references
    public float TotalPrice { get; set; }
    1 reference
    public long BuyerId { get; set; }
}

3 references
public class ProductItem
{
    2 references
    public long ProductId { get; set; }
    6 references
    public int OrderedQuantity { get; set; }
}

```

Слика 19. Пример коришћене DTO класе за обраду пристиглог захтева за креирање наруџбине

ПРЕДЛОЗИ ЗА ДАЉА УСАВРШАВАЊА

Имплементирани систем у тренутној верзији представља потпуно исправну и функционалну апликацију. Наравно, увек постоји простор за даља истраживања и усавршавања исте.

У контексту предње стране, било би добро обратити пажњу на прилагодљивост дизајна апликације, односно исправити све могуће проблеме при приказивању на свим могућим уређајима. Одређене ствари је потребно скалирати у односу на величину екрана, док би у неким ситуацијама вероватно било погодније направити и нешто другачији дизајн.

Приказ производа на страници није регулисан пагинацијом, на тај начин би садржај био прегледнији за кориснике. Не постоји опција за претрагу, филтрирање или сортирање производа по неким критеријумима, што корисницима система отежава коришћење продавнице. Постојање категоризације производа би употребу система учинило пуно интуитивнијом.

Информације о достави би требало имплементирати другачије, јер само податак о тачном броју просталих сати одступа од реалности. Било би добро да постоји посебно заглавље на коме би корисник могао да прати стање наруџбине, нпр: наруџбина примљена, потврђена, послата, и томе слично.

Начин плаћања наруџбине није дефинисан у систему, што би представљало још једну ману. Корисник би требао да има могућност да изабере између нпр. плаћања поузећем или платном картицом. Још неки од начина плаћања обухватали би плаћање чековима или плаћање на рате.

Постојање одређеног система оцењивања би значајно допринело апликацији. Оцењивање се може посматрати као оцењивање продавца, позитивним или негативним оценама, или оцењивање индивидуалних производа у виду искуства и мишљења.

Побољшање корисничког искуства представља важну ствар у развоју апликација, па би самим тим требао да постоји механизам за прикупљање и анализу повратних информација од стране корисника. Корисници би имали посебну функционалност у оквиру које би могли да пријављују сваки потенцијални баг или грешку на систему.

У контексту задње стране и микросервисне архитектуре, могућа је подела микросервиса *ProductOrder*, на два микросервиса од којих би се један бавио посебно производима, а други посебно наруџбинама. Таква подела би допринела скалабилности система, али би додатно оптеретила систем. Пошто је постојање страних кључева између табела уклоњено, врше се додатне валидације и провере података на апликативном нивоу. Такође, треба обратити пажњу на то да ли је подела заправо неопходна јер са собом доноси и компликованији поглед на целокупни систем.

Не постоји никакав механизам који би био задужен за отпорност на испаде у систему. Било би пожељно размислити о таквим опцијама које би знатно допринеле поузданости система. Имплементација детаљнијег система логовања би утицала на лакши увид у активности целог система, па самим тим и лакше решавање проблема и анализу грешака.

Предности микросервисне архитектуре би се огледале у скалабилности апликације, бржем развоју и испоруци софтвера, лакшем одржавању индивидуалних функционалности, као и технолошкој разноликости. Различити микросервиси могу користити различите технологије, што омогућава избор најбољег алата за сваку специфичну употребу.

Мане микросервисне архитектуре обухватају сложеност апликације, координацију и комуникацију која може бити изазовна, оперативне трошкове јер може захтевати више ресурса у контексту управљања и одржавања, и складиште података јер може постати компликовано посебно јер постоји потреба за конзистентношћу.

ЛИТЕРАТУРА

- [1] <https://dev.to/scorpio69/how-to-web-api-net-core-basics-to-advanced-part-4-service-layer-31gk>, структура WebApi апликације
- [2] <https://visualstudio.microsoft.com/>, Microsoft Visual Studio
- [3] <https://code.visualstudio.com/docs>, Microsoft Visual Studio Code
- [4] <https://legacy.reactjs.org/docs/getting-started.html>, React
- [5] <https://learn.microsoft.com/en-us/sql/ssms/sql-server-management-studio-ssms?view=sql-server-ver16>, SQL Server Management Studio
- [6] <https://ocelot.readthedocs.io/en/latest/introduction/bigpicture.html>, Ocelot API Gateway
- [7] <https://automapper.org/>, NuGet package AutoMapper
- [8] <https://learn.microsoft.com/en-us/ef/core/>, Entity Framework Core
- [9] <https://github.com/BcryptNet/bcrypt.net>, BCRYPT
- [10] <https://www.nuget.org/packages/Google.Apis.Auth/>, Google аутентификација
- [11] <https://serilog.net/>, логовање коришћењем Serilog
- [12] <https://jwt.io/introduction>, JWT токен аутентификација
- [13] <https://www.npmjs.com/package/react-toastify>, React-toastify библиотека
- [14] <https://axios-http.com/docs/intro>, Axios
- [15] <https://reactrouter.com/en/main/router-components/browser-router>, RouterBrowser у React
- [16] <https://blog.imi.pmf.kg.ac.rs/spring-cloud-mikroservis-arhitektura/>, микросервисна архитектура