PAMETNA SELA

Sertić, Antonio

VELEUČILIŠTE U VIROVITICI
Preddiplomski stručni studij Računarstva

ANTONIO SERTIĆ

PAMETNA SELA
ZAVRŠNI RAD

VIROVITICA, 2021.

VELEUČILIŠTE U VIROVITICI
Preddiplomski stručni studij Računarstva

PAMETNA SELA
ZAVRŠNI RAD

Predmet: Projektiranje informacijskih sustava

Mentor:
dr. sc. Oliver Jukić, prof. v. š.

Student:
Antonio Sertić

VIROVITICA, 2021.

Veleučilište u Virovitici

Preddiplomski stručni studij Računarstva - Smjer Programsko inženjerstvo

OBRAZAC 1b

ZADATAK ZAVRŠNOG RADA

Student/ica: **ANTONIO SERTIĆ** **JMBAG:** **0307015533**

Imenovani mentor: dr. sc. Oliver Jukić, prof. v. š.

Imenovani komentor: -

Naslov rada:

Pametna sela

Puni tekst zadatka završnog rada:

Tema zadatka je aplikacija razvijena u Blazor tehnologiji, a koje je inspirirana konceptom "Pametna sela" (http://www.obz.hr/index.php/pametna-sela).

Aplikacija bi imala dvije vrste korisnika: Poljoprivrednik i Kupac.

Namjena aplikacije bi bila da Poljoprivredniku omogući bolju suradnju i lakši dolazak do svojih kupaca, što bi uključivalo izradu "Marketplace-a" na kojem bi poljoprivrednik mogao objaviti što prodaje i sve detalje vezane uz taj proizvod.

Potom dolazi kupac te on naručuje taj proizvod. Kupac ima mogućnost ocjenjivanja proizvoda i javljanja poljoprivredniku putem poruka za više upita.

Aplikacija bi se dala proširiti – osim grana za prodaju i kupnju tu bi bila i treća s poljoprivrednikove strane koja bi služila isključivo njemu za uzgoj. U ovom dijelu aplikacije bi on mogao voditi evidenciju, lakšu i kvalitetniju brigu o svojim proizvodima – od vođenja evidencije potrošnje goriva za mehanizaciju (veliki poljoprivrednici) do evidencije potrošnje vode u manjim staklenicima. Razmotriti i rad aplikacije s podacima u stvarnom vremenu.

Primjer korištenja: Poljoprivrednik ima zasađen voćnjak jabuka (ili planira zasaditi) te sve savjete i pomoć može pronaći na stranici tako da se savjetuje s ostalim poljoprivrednicima iz svojeg kraja i šire. Također se nudi mogućnost da odabere npr. sortu jabuka nakon čega mu se automatski prikažu savjeti za gnojenje, obrezivanje itd..

Prije izrade same aplikacije, potrebno je kreirati bazu podataka (odaberite vrstu), koja će biti osnova za izradu aplikacije. Potrebno je kreirati osnovne tablice, za pojedina polja koja imaju predefinirane vrijednosti potrebno je kreirati i šifrarnike. U osnovnim tablicama potrebno je definirati primarne ključeve i indekse. Bazu je potrebno napuniti testnim podacima.

Nakon kreiranja baze podataka potrebno je nekim od formalnih jezika unutar jezičnog prostora napraviti specifikaciju aplikacije koja će biti zatim implementirana kroz završni rad.

Datum uručenja zadatka studentu/ici: 28.07.2021.
Rok za predaju gotovog rada: 06.09.2021.

Mentor:

dr. sc. Oliver Jukić, prof. v. š.

Dostaviti:
1. Studentu/ici
2. Povjerenstvu za završni rad - tajniku

PAMETNA SELA

SMART VILLAGES

SAŽETAK – *Cilj ovog rada bio je osmisliti i kreirati programsko rješenje, tj. web aplikaciju koja bi omogućila poljoprivrednicima bolju suradnju i lakši dolazak do svojih kupaca. Aplikacija je podijeljena u dva modula. Prvi modul je tržnica u koju poljoprivrednik može dodavati svoje proizvode postavljajući bitne informacije o njima. Proizvod se potom može pronaći u tržnici pretraživanjem ili filtriranjem. Klikom na proizvod dobivaju se detaljnije informacije i odakle se kupac može direktno javiti poljoprivredniku putem poruka. Kupac može dodati proizvod u košaricu te odlaskom na samu košaricu naručiti željene proizvode. Poljoprivrednici dobivaju narudžbu te kada ona bude izvršena označavaju je kao izvršenu gdje potom kupac može vidjeti narudžbu kao dovršenu i dati posebno svakom proizvodu ocjenu i komentar. Drugi modul su poruke gdje se vode razgovori kupaca i poljoprivrednika u stvarnom vremenu. Odabrane tehnologije za izradu sustava su: Blazor koji koristi C# i Web Assembly, SignalR, MudBlazor, te Entity Framework Core i Microsoft SQL Server za kreiranje i rad s bazom podataka. U prvom dijelu rada opisane su izabrane tehnologije koje su se koristile za izradu sustava, u drugom dijelu arhitektura samog sustava i baze podataka, a u trećem dijelu glavne funkcionalnosti sustava.*

Ključne riječi: *Web aplikacija, tržnica, poljoprivrednik, kupac, Blazor*

SUMMARY – *The purpose of this final work is to design and create software solution i.e. web application that would enable farmers to better cooperate and easier access to its customers. The application is divided into two modules. The first module is a marketplace in which the farmer can add his products by posting essential information about them. The product can then be found in the marketplace by searching or filtering. By clicking on the product, more detailed information is obtained and from where the buyer can contact the farmer directly via messages. The customer can add the product to the cart and order the desired products by going to the cart itself. Farmers receive the order and when it is executed, they mark it as ordered, then the customer can then see the order as completed and give each product a rating and comment. The second module is messages where customer and farmer can have conversations in real time. Selected technologies for creating the system are: Blazor that uses C # and Web Assembly, SignalR, MudBlazor, and Entity Framework Core with Microsoft SQL Server to create and work with the database. The first part of this final work describes the selected technologies that were used to create the system, the second part describes the architecture of the system and the database, and the third part describes the main functionality of the system.*

Keywords: *Web application, marketplace, farmer, customer, Blazor*

SADRŽAJ

UVOD

Ideja za izradu web aplikacije Pametna sela nastala je rješavanjem problematike s kojom se suočavaju mali lokalni poljoprivrednici. Problem im stvara dolazak do svojih kupaca koji radije biraju svježe namirnice od lokalnog poljoprivrednika nego odlazak u dućan po uvezene namirnice. Dakako već postoje lokalne tržnice koje rješavaju taj problem, ali stvaraju nove kako za poljoprivrednika tako i za kupca. Novi problemi su povezani s tržnicom i samim odlaskom na nju. Kod tržnica otvorene gradnje problem mogu stvarati vremenske nepogode (npr. kiša, snijeg, vrućine…), te potencijalni nedostatak ljudi. Sami odlazak na tržnicu stvara potencijalne probleme poljoprivredniku kod transporta namirnica na tržnicu te kupcu sami odlazak može biti napor ako se radi o starijim ljudima ili nedostatak vremena kod mlađih. Današnja vremena su stvorila i nove nepogode kao što je pandemija virusa COVID-19 koja otežava odlazak na tržnicu, ali i prilike kao što su digitalizacija koja omogućava lakši i brži dolazak kupcu do namirnica.

Svrha izrade ovog informacijskog sustava je upravo digitalizacija manjih lokalnih poljoprivrednika koja im donosi razne prednosti i pomaže im u razvoju i samoj prodaji svojih proizvoda. Poljoprivredniku se kupci sami javljaju te se oni mogu međusobno dogovoriti oko transporta putem poruka. Kupac može od doma naručiti njemu potrebne zdravije i kvalitetnije namirnice i znati kada ih može očekivati bez potrebe za odlaskom na tržnicu.

Za izradu web aplikacije odabrana je trenutno nova, moderna i sve popularnija tehnologija zvana Blazor. Zato što je jako intuitivna za trenutne i potencijalno nove developere[1] sustava.

Teorijski dio završnog rada podijeljen je u tri glavne sadržajne komponente. U prvoj komponenti su detaljnije opisane sve korištene tehnologije. U drugoj komponenti je opisana arhitektura aplikacije. U trećoj komponenti su opisane glavne funkcionalnosti odnosno samo korištenje aplikacije.

[1] Developer - razvojni inženjer

1. KORIŠTENE TEHNOLOGIJE

Prije opisivanja arhitekture aplikacije opisat će se tehnologije kojom je ona građena. Tehnologije korištene u svrhu izrade su: Blazor koji koristi C# i Web Assembly, SignalR, MudBlazor, te Entity Framework Core i Microsoft SQL Server za kreiranje i rad s bazom podataka. Kako bi počeli priču o Blazoru prvo se trebaju razjasniti neke stvari i pojmovi kao što su C# programski jezik, .NET Framework uz ASP.NET te WebAssembly.

1.1. C# programski jezik

C# je programski jezik kojeg je razvio Microsoft 2000. godine. Namijenjen je za izradu aplikacija u .NET okruženju o kojem će biti više riječi u nastavku. Kasnije su ga za standard odobrili Ecma (ECMA-334) i ISO (ISO/IEC 23270:2006) te je jedan od programskih jezika dizajniran za CLI[2]. Poznat je zbog povezanosti s objektno orijentiranim jezicima Java i C++ na temelju kojih je i nastao te je stoga i on sam objektno orijentiran. Sintaksu je većinski također preuzeo od Jave. Najnovija verzija je C# 9.0, koja je objavljena u rujnu 2020.

1.2. .NET Framework

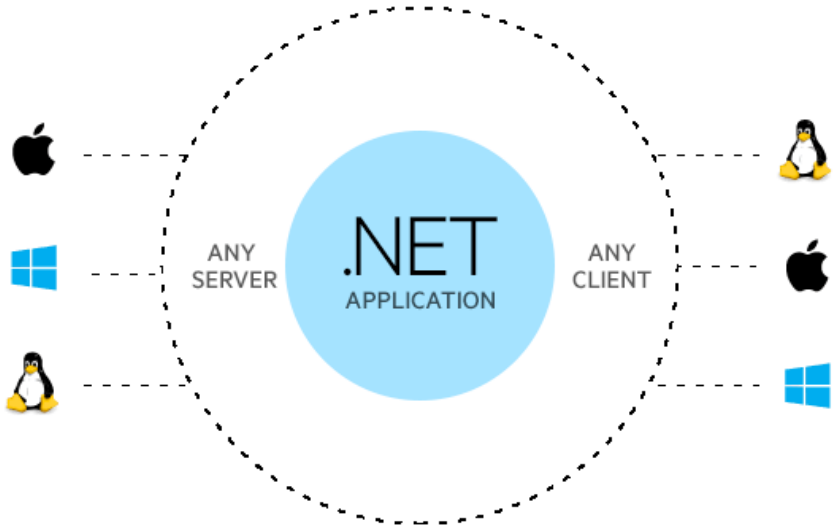
.NET Framework je softverski okvir kojeg razvija Microsoft, a koji prvenstveno radi na Microsoft Windows-u. Uključuje veliku biblioteku klasa poznatu kao Framework Class Library (FCL) i pruža jezičku interoperabilnost[3] u nekoliko programskih jezika. Programi napisani za .NET Framework se izvršavaju u softverskom okruženju, poznatom kao Common Language Runtime (CLR), aplikacijska virtualna mašina koja pruža usluge kao što su sigurnost, upravljanje memorijom, i rukovanje izuzecima. FCL i CLR zajedno čine .NET Framework.

FCL pruža korisničko sučelje, pristup podacima, povezanost baze podataka, kriptografiju, razvoj web-aplikacija, numeričke algoritme, i mrežne komunikacije. Programeri proizvode softver kombinacijom vlastitih izvornih kodova sa .NET Framework-om i ostalim bibliotekama. .NET Framework je namijenjen za upotrebu većine novih aplikacija kreiranih za Windows platforme. Microsoft također proizvodi integrirano razvojno okruženje uglavnom za .NET softver nazvan Microsoft Visual Studio.

[2] CLI - infrastruktura zajedničkog jezika
[3] Interoperabilnost – svaki jezik može koristiti kodove napisane na drugim jezicima

.Net Framework je baziran isključivo za Windows platformu što je navelo Microsoft da napravi verziju .Net-a koja se može pokretati na svim platformama - .Net Core(Cross platform .Net Framework).

Slika 1. Cross platform mogućnosti .NET-a



Izvor: izrada autora prema WebITGurus, https://webitgurus.com/how-net-core-is-beneficial-for-developing-an-enterprise-application (17.08.2021.)

Aplikacije za .NET platformu mogu se pisati u raznim programskim jezicima, gotovo svim poznatijim. CLR, međutim, ne poznaje niti jedan taj jezik - on dobiva naredbe isključivo u jeziku nazvanom Microsoft Intermediate Language (skraćeno MSIL), temeljen na pravilima koja se nazivaju Common Language Specifications (CLS). Stoga je jasno da mora postojati kompajler[4] koji će programski jezik u kojem programer piše kod, prevesti u MSIL kako bi ga CLR razumio. Ovi kompajleri nazivaju se IL-kompajleri te su dostupni za velik broj programskih jezika. Microsoft je izdao kompajlere za pet jezika: C#, J#, C++, Visual Basic i JScript, dok su se ostali proizvođači softvera potrudili oko brojnih drugih kao što su: Perl, Python, Cobol, Eiffel…

1.2.1. ASP.NET

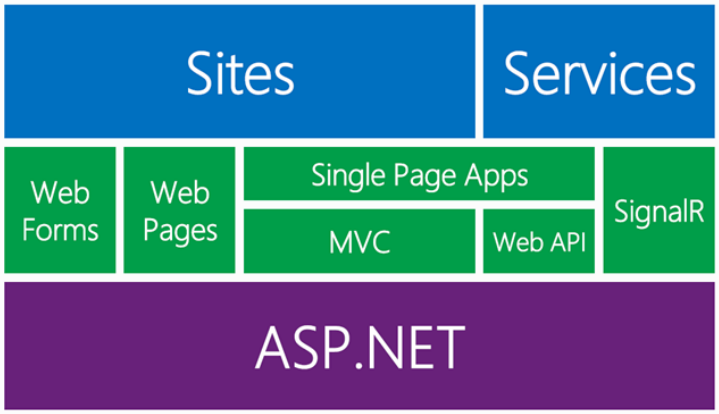
[4] Kompajler – računalni program koji čita program napisan u izvornom jeziku, te ga prevodi u ciljni jezik

Microsoft 1997. godine započinje razvoj programskih okvira za izradu web aplikacija, izdavanjem Option Packa za Windows NT Server, u kojem unutar IIS-a 4.0 (Internet Information Services) uvodi značajnu novost - prvu verziju svog Web programskog jezika nazvanog Active Server Pages ili skraćeno ASP. Dvije godine kasnije, izlaskom Windowsa 2000, izlazi i IIS 5.0 s ASP-om 3.0 što će biti ujedno i posljednja inačica „klasičnog“ ASP-a.

Početkom 2001. godine, Microsoft objavljuje osnovnu arhitekturu svoje nove tehnologije nazvanu .NET. Sredinom 2002. godine finaliziran je .NET Framework 1.0 i MS Visual Studio 2002. Od tog vremena potječe sveopća opčinjenost .NET-om koja traje i danas. ASP.NET je objavljen 2002. godine uz .NET 1.0 i bio je open-source programski okvir namijenjen za izradu server-side web aplikacija i izradu dinamičnih web stranica ali i aplikacija i servisa.

Nasljednik ASP.NET-a je ASP.NET Core koji je re-implementacija ASP.NET-a kao modularnog programskog okvira zajedno s ostalim programskim okvirima poput Entity-ja. Novi ASP.NET koristi novu open source .NET platformu(„Roslyn“) i dostupan je na svim platformama. ASP.NET MVC, ASP.NET Web API, i ASP.NET Web Pages (platforma koja koristi samo Razor stranice – HTML + C#) spojeni su u unificiran MVC 6.

Slika 2 . ASP.Net tehnologije

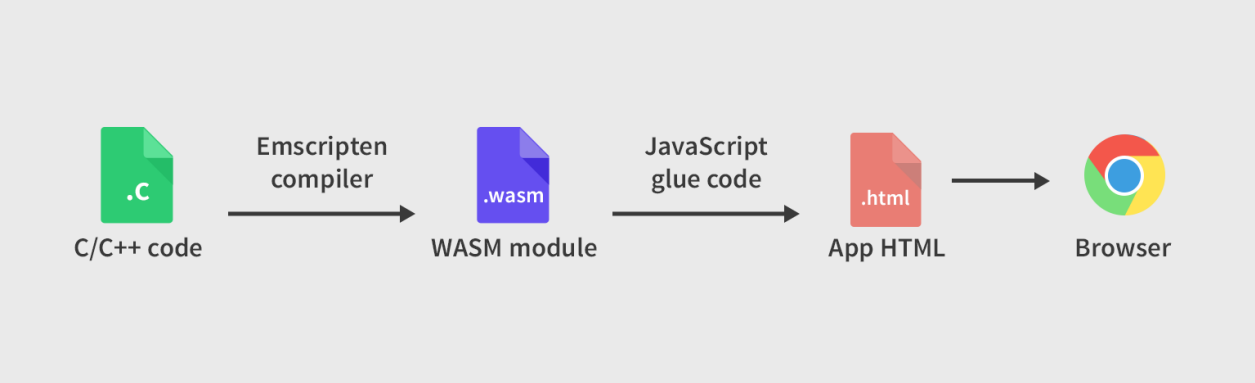


Izvor: izrada autora prema C# Corner, https://www.c-sharpcorner.com/article/asp-net-overview-and-comparison-of-asp-net-technologies/ (17.08.2021.)

1.3. WebAssembly

WebAssembly(Wasm) je najavljen 2015. godine i prva demonstracija Wasm-a bila je igrica Angry Bots u Unity game engine-u koja se pokretala u Firefox-u, Google Chrome-u i MS Edge-u. Prethodnik je bila tehnologija od Mozille i Google Native Clienta - asm.js. Ta tehnologija pružila je podlogu za kreiranje samog WebAssembly-ja i danas valjanu alternativu za preglednike koji ne podržavaju Wasm ili za preglednike koji su ga isključili iz sigurnosnih razloga. U ožujku 2017. godine završena je pregledna faza i dizajn MVP-a(Minimum viable product). Poslije završene MVP verzije nastavljen je rad na garbage collection-u i multithreading-u što je omogućilo kompajliranje koda u jezicima koji zahtijevaju te funkcionalnosti. Neki od tih jezika su C#(Blazor), F#(Bolero), Python, Java, Julia, Ruby, Go...

Slika 3. Proces prevođenja koda za WASM



Izvor: izrada autora prema Tutorialzine, https://tutorialzine.com/2017/06/getting-started-with-web-assembly (17.08.2021.)

WebAssembly je otvoreni standard koji definira prijenosni format binarnog koda za izvršne programe i odgovarajući tekstualni asemblerski jezik, kao i sučelja za olakšavanje interakcije između takvih programa i njihovih poslužiteljskih okolina. Dizajniran je kao portabilni kompajler za jezike kao što su C#, C, C++, Rust koji omogućava njihov razvoj na webu za klijentske i serverske aplikacije. Njegov osnovni cilj je omogućavanje rada visoko-performansnih aplikacija na web stranicama, ali format je prilagođen i kreiran da se pokreće i integrira u druge okoline.

Trenutno oko 40 programskih jezika podržava Wasm kao ciljni kompajler. WebAssembly postao je W3C(World Wide Web Consortium) standard 5.12.2019. uz HTML, CSS i JavaScript četvrti je jezik koji se pokreće izvorno(natively) u pregledniku.

U studenome 2017. godine Mozilla je najavila podršku u skoro svim preglednicima, te je danas Wasm omogućen skoro svim korisnicima interneta (91.9% desktop preglednici, 91.78% mobilni preglednici), a onim koji nemaju mogućnost korištenja Wasm-a, za starije preglednike, stvoren je JavaScript polyfill[5] koji kompajlira Wasm kod u asm.js. Prvobitni cilj bio je podrška za kompajliranje koda u C i C++ te je kasnije omogućena i za Rust i .Net jezike(C#, F#, ...)

1.4. Blazor

Blazor je programski okvir za stvaranje web aplikacija na strani klijenta, te se najviše izdvaja od drugih jer su njegove aplikacije pisane C# programskim jezikom umjesto svuda prisutnog JavaScript-a. Odlikuje ga mogućnost pisanja klijentske i poslužiteljske strane za istu aplikaciju u istom programskom jeziku što smanjuje vrijeme izrade web aplikacija, olakšava programeru snalaženje u pisanju koda i omogućava dijeljenje koda i biblioteka.

Blazor je dio ASP.NET-a koji je dio .NET razvojne platforme otvorenog koda koja posjeduje moćne i raskošne alate za kreiranje web aplikacija, te se odlikuje snažnom zajednicom suradnika iz više od 3.700 tvrtki. .NET je besplatan, a to uključuje i Blazor. Ne postoje naknade niti troškovi licenciranja, uključujući i za komercijalnu uporabu.

Blazor aplikacije temelje se na komponentama, gdje svaka komponenta sadrži svoj HTML, CSS i C# kod. Blazor komponente imaju višekratnu upotrebu što znači da možemo imati jednu komponentu i koristiti je na više mjesta u aplikaciji. Komponenta je klasa koja se piše u formi Razor markup stranice sa .razor nastavkom za datoteku. Komponente u Blazoru formalno se nazivaju Razor komponentama. Razor je sintaksa koja kombinira HTML i C# u istom fajlu s IntelliSense podrškom. Razor stranice i MVC također koriste Razor. Za razliku od Razor stranica i MVC, koji su građeni na principu request/response modela, komponente su korištene specifično za klijentsku stranu UI logike i kompozicije.

[5] Polyfill - kod koji implementira funkcije u pregledniku koje izvorno nisu kompatibilne ili podržane od strane tog preglednika

Slika 4. Razor markup

```
@layout EmptyLayout
@page "/notfound"

<div class="div_container">
    <img src="./img/404error.png" />
    <div class="jumbotron">
        <h1 class="display-4">404 Error - Page Not Found</h1>
        <p class="lead">The page you requested could not be found..</p><hr class="my-4">
        <p>We suggest you go to the home page.</p>
        <button class="btn btn-success btn-lg" @onclick="GoHome" role="button">Home</button>
    </div>
</div>

@code {
    @inject NavigationManager NavigationManager

    public void GoHome()
    {
        NavigationManager.NavigateTo("/");
    }
}
```

Izvor: izrada autora (18.08.2021.)

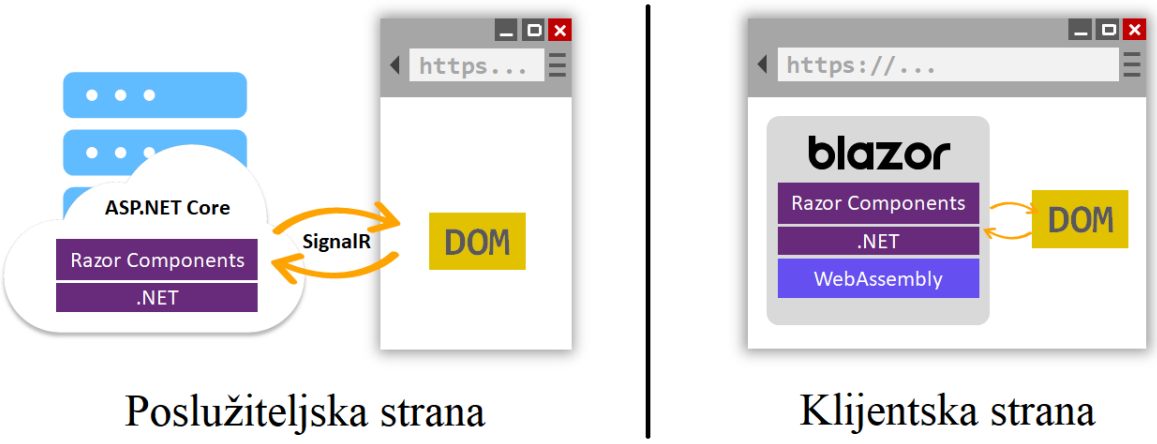
Prethodna slika prikazuje komponentu NotFound.razor koja se poziva kada pristupimo nepostojećoj ruti na stranici. Blazor koristi HTML tagove za UI kompoziciju. GoHome je C# metoda koja se pokreće kada se pokrene onclick event prilikom klika na gumb. Metoda je definirana unutar @code{} te se izvršava naredba koja nas vraća na početnu stranicu.

Blazor može pokretati C# kod na strani klijenta izravno u pregledniku, koristeći WebAssembly. Budući da je pravi .NET pokrenut na WebAssembly-u, može se ponovno koristiti kod i biblioteke iz dijelova aplikacije na strani poslužitelja. Alternativno, Blazor može pokrenuti logiku klijenta na poslužitelju. Klijentski UI događaji se šalju natrag na poslužitelj koristeći SignalR (a-real-time messaging framework). Nakon završetka izvođenja, potrebne UI promjene se šalju na klijenta i spajaju u DOM[6].

Blazor je podržan na Windows-u, MacOS-u i Linux-u te radi u svim modernim web preglednicima, uključujući i mobilne preglednike. Kod pokrenut u pregledniku izvršava se u istom sigurnosnom okruženju kao i JavaScript programski okviri. Blazor kod koji se izvršava na poslužitelju ima fleksibilnost da učini sve što bi se inače radilo na poslužitelju, kao što je izravno povezivanje s bazom podataka. C# kod može lako pozvati JavaScript API[7]-je i biblioteke te se može nastaviti koristiti veliki ekosustav JavaScript biblioteka koje postoje za UI na strani klijenta dok se logika piše u C#.

[6] DOM – eng. Document Object Model
[7] API – Application Programming Interface

Slika 5. Rad Blazora na poslužiteljskoj i klijentskoj strani

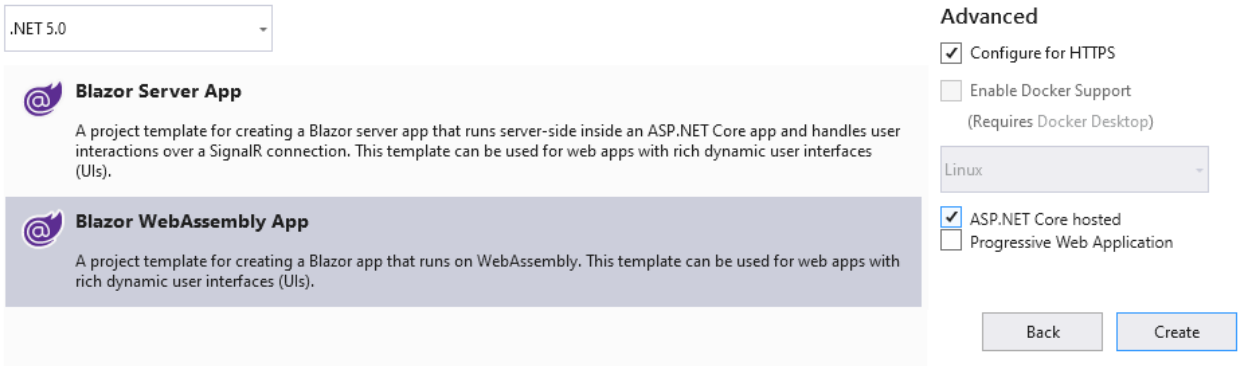


Izvor: izrada autora (18.08.2021.)

Blazor se može integrirati s modernim hosting platformama kao što je Docker, otvorena platforma za razvoj, isporuku i pokretanje aplikacija.

Pri kreiranju Blazor projekta postoje dvije opcije: Blazor Server App i Blazor WebAssembly App. Neposredno su u nastavku opisane njihove karakteristike, razlike i prednosti.

Slika 6. Kreiranje Blazor projekta



Izvor: izrada autora (18.08.2021.)

1.4.1. Blazor WebAssembly

Blazor WebAssembly je signle-page aplikacijski programski okvir za pravljenje interaktivnih web aplikacija u .NET-u. Koristi open web standarde bez plugin-ova, te radi u svim modernim preglednicima, uključujući i preglednike za pametne mobitele.

Pokretanje .NET koda unutar preglednika omogućio je WASM. WASM kod može pristupiti punoj funkcionalnosti preglednika putem JavaScript-a(JavaScript interoperability / JavaScriptinterop). .NET kod izvršen preko WASM-a u pregledniku pokreće se u preglednikovom JavaScript sandbox-u sa zaštitom koju taj sandobx pruža protiv zlonamjernih akcija na klijentskom računalu.

Blazor WebAssembly koristi .NET runtime i konfigurira ga da učita assemblije za aplikaciju. Koristi JavaScript interop da upravlja DOM-om i API pozivima

Veličina objavljene aplikacije(payload size) je ključna za performanse aplikacije. Velikoj aplikaciji treba relativno puno vremena da se učita, što pogoršava korisnički doživljaj i iskustvo. Blazor WebAssembly optimizira payload size da smanji vrijeme skidanja.

1.4.2. Blazor Server

Blazor odvaja logiku renderiranja komponenti od načina kako se UI ažuriranja apliciraju. Blazor Server pruža podršku za hosting Razor komponenti u ASP.NET Core aplikaciji. UI ažuriranjima se upravlja kroz SignalR konekciju.

Runtime upravlja slanjem UI eventa od preglednika do servera i primjenjuje UI ažuriranja poslana od strane servera nazad do preglednika nakon pokretanja komponenti.

Konekcija koja se koristi u Blazor Server-u za komunikaciju s preglednikom se također koristi za upravljanje JavaScript interop pozivima.

1.5. SignalR

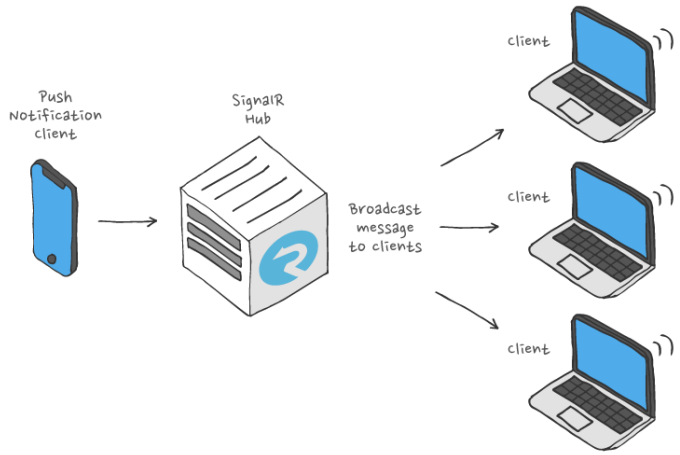
SignalR jedna je od najčešće korištenih biblioteka za razvoj Web aplikacija u stvarnom vremenu otvorenog koda. Izrađena je 2011. godine od strane Davida Fowlera i Damiana

Edwardsa, no potencijal ove biblioteke ubrzo je prepoznat sa strane Microsoft-ovih razvojnih inženjera koji kasnije preuzimaju njegov razvoj.

Dobar izbor za korištenje SignalR-a su aplikacije koje zahtijevaju česte komunikacije s poslužiteljem, nadzorne ploče ili aplikacije za nadzor, zajedničke aplikacije, aplikacije koje zahtijevaju obavijesti. Primjeri su igre, društvene mreže, GPS aplikacije, trenutna ažuriranja prodaje itd.

SignalR koristi hub-ove za komunikaciju između klijenata i poslužitelja. Omogućuje klijentu i poslužitelju da međusobno pozivaju metode. Hub-ovi pozivaju kod na strani klijenta slanjem poruka koje sadrže naziv i parametre metode na strani klijenta. Objekti poslani kao parametri metode deserijaliziraju se pomoću konfiguriranog protokola. Klijent pokušava uskladiti ime s metodom u kodu na strani klijenta. Kada klijent pronađe podudaranje, on poziva metodu i prosljeđuje mu deserijalizirane podatke parametara.

Slika 7. SignalR komunikacija



Izvor: izrada autora prema Ably, https://ably.com/periodic-table-of-realtime/signalr *(18.08.2021.)*

1.6. MudBlazor

Dodatno povećanje produktivnosti se može postići korištenjem UI komponenti koje su već napravljene za korištenje u Blazor-u. Neki od kreatora gotovih UI komponenti su: Telerik, DevExpress, Syncfusion, Radzen, MudBlazor... Neki se plaćaju dok postoje i besplatni. U projektu je korišten MudBlazor koji je besplatan.

MudBlazor je ambiciozni programski okvir za materijalni dizajn za Blazor s naglaskom na jednostavnosti korištenja i jasnoj strukturi. Savršen je za .NET programere koji žele brzo graditi web aplikacije bez potrebe za borbom s CSS-om i Javascript-om. U potpunosti je napisan na C#-u, omogućuje prilagodbu, popravljanje ili proširenje okvira.

1.7. Entity Framework Core

Entity Framework Core je lagana, proširiva i za više platformi verzija Microsoftovog Object-relational mapper-a (ORM[8]), te je službena Microsoftova platforma za rad s bazom podataka. Olakšava programerima pisanje jer eliminira potrebu za pisanjem većine koda.

Entity Framework Core je prvi put objavljen krajem lipnja 2016. nakon više od 2 godine rada. Prvu verziju prati velika promjena s EF Core 2, koja je objavljena godinu dana kasnije zajedno s .NET Core i ASP.NET Core. Posljednja verzija je EF Core 5.0 izašla 12. Siječnja 2021.

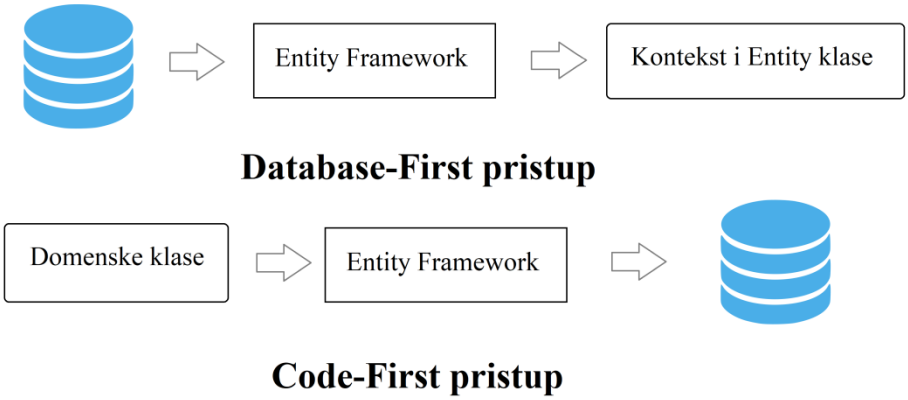
Entity Framework Core podržava mnoge pružatelje baza podataka kao što su: SQL Server, MySQL, PostgreSQL, SQLite, SQL Compact, In-memory.

Postoje dvije vrste razvojnog pristupa modeliranju. Prvi pristup je Code-First čiji tijek rada započinje klasama koje opisuju konceptualni model, a zatim Entity Framework automatski generira bazu podataka iz tog modela. Dok je drugi pristup Database-First koji je suprotan prvome gdje se prvo kreira baza podataka te po njoj nastaju modeli klasa.

Code-First nudi najveću kontrolu nad konačnim izgledom aplikacijskog koda i rezultirajućom bazom podataka, te je zato upravo on izabran kao pristup u ovom projektu.

[8] ORM – Objektno-relacijski preslikač

Slika 8. Razvojni pristupi modeliranju u EF Core-u



Izvor: izrada autora (19.08.2021.)

1.8. Microsoft SQL Server

SQL Server je sustav za upravljanje relacijskim bazama podataka (RDBMS) kojeg je predstavio Microsoft. Prva verzija objavljena je 1989. godine pod imenom „SQL SERVER for OS/2 1.0“. Pruža okruženje za stvaranje i upravljanje bazama podataka. Omogućuje sigurno i učinkovito skladištenje te pruža druge komponente i usluge koje podržavaju platformu poslovne inteligencije za generiranje izvješća i pomoć pri analizi podataka.

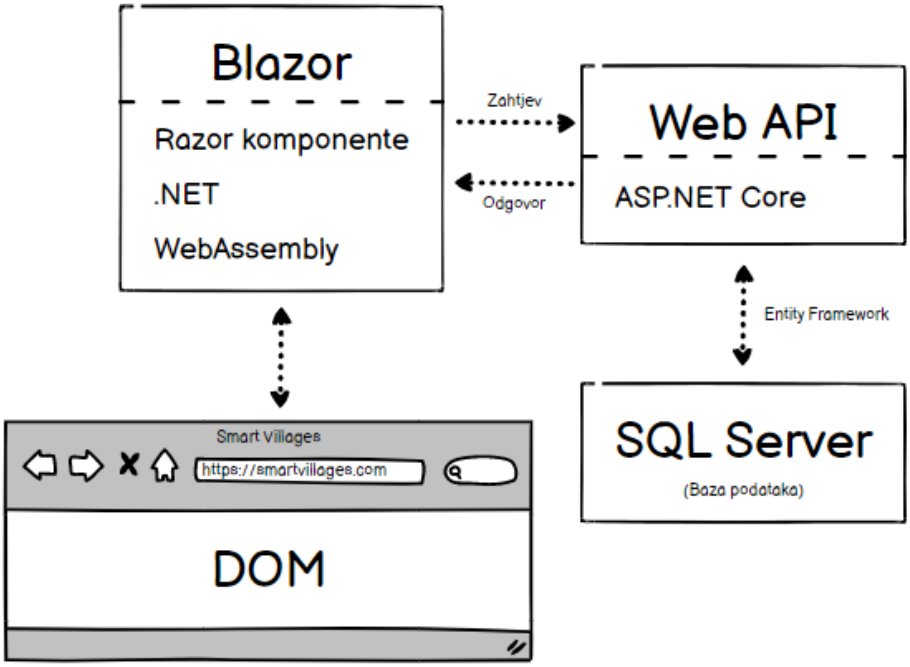
SQL Server podržava ANSI SQL, koji je standardni jezik SQL (jezik strukturiranih upita). Međutim, SQL Server dolazi s vlastitom implementacijom SQL jezika, T-SQL (Transact-SQL). Pruža daljnje mogućnosti deklariranja varijable, rukovanja iznimkama, pohranjene procedure itd.

SQL Server Management Studio (SSMS) glavni je alat sučelja za SQL Server i podržava 32-bitna i 64-bitna okruženja.

2. ARHITEKTURA APLIKACIJE

Nakon što smo opisali najvažnije tehnologije, opisat ćemo arhitekturu aplikacije pomoću kojih je ona napravljena. Arhitektura aplikacije je podijeljena na dva dijela. Prvi dio je arhitektura sustava gdje je opisana sama programska arhitektura u kojoj se nalazi klijentski, poslužiteljski i dijeljeni kod. Drugi dio je arhitektura baze podataka koja opisuje kako i po kojem principu su rađene klase unutar sustavne arhitekture i tablice u bazi podataka.

Slika 9. Arhitektura aplikacije



Izvor: izrada autora (30.08.2021.)

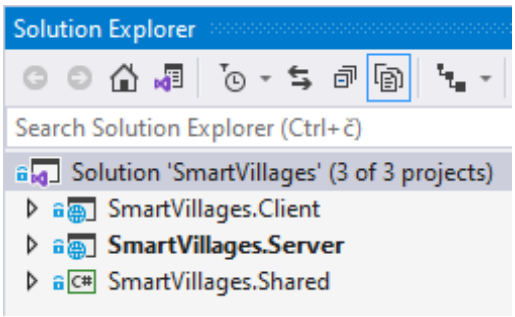
Pozadina rada aplikacije je skicirana na slici 9. gdje se može bolje predočiti sami tok radnji u pozadini. Pokretanjem radnje u aplikaciji poziva se određena funkcija koja je definirana u Blazor projektu te se izvrše napisane funkcije i šalju promjene natrag na web preglednik. Ako učinjene radnje zahtijevaju podatke, kreira se zahtjev prema poslužitelju (Web API-ju) te on putem Entity Framework-a dohvaća ili radi promjene nad podacima u bazi podataka i šalje ih natrag na klijenta.

2.1. Arhitektura sustava

Pri kreiranju sustava odabrana vrsta projekta je Blazor WebAssembly te kao dodatna opcija označen je checkbox „ASP.NET Core hosted“ koji se nalazi s desne strane pod stupcem Advanced što se može vidjeti na slici 6. Ta dodatna opcija je odobrena zato što nudi mogućnost korištenja mnogih stvari s poslužitelja, ali se ipak te stvari moraju dohvatiti putem API-ja. Stoga ne zahtijeva stalnu konekciju na Internet i ne komunicira sa serverom radi svake funkcionalnosti u aplikaciji kao što bi u Blazor Server tipu projekta bio slučaj nego se mogu i u internet pregledniku izvršiti.

Hosted opcija je tu jer u isto vrijeme kada se kreira klijentski dio aplikacije možemo automatski kreirati i ASP.NET Core projekt koji služi kao API, tada se može ponovno upotrijebiti to isto mjesto za hostiranje Blazor klijenta. Stvorit će tri projekta: [ImeAplikacije].Client, [ImeAplikacije].Server, [ImeAplikacije].Shared opisanih u nastavku.

Slika 10. Blazor WebAssembly hosted arhitektura

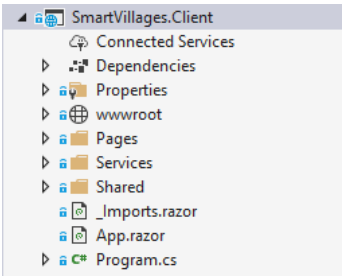


Izvor: izrada autora (19.08.2021.)

2.1.1. Client

U Client projektu se kako i samo ime kaže nalazi sve za klijentsku stranu. Mapa Pages sadrži sve razor komponente do kojih se može doći. Oni će unutar sebe sadržavati direktivu @page. Mapa Shared sadrži razor komponente koje se mogu koristiti bilo gdje u aplikaciji kao što su dijalozi i layout-i. Datoteka App.razor sadrži komponentu za usmjeravanje koja određuje što učiniti kada se pronađe ili ne pronađe ruta. Program.cs sadrži kod koji pokreće Blazor WebAssembly aplikaciju i gradi dependency injection kontejner.

Slika 11. Arhitektura Client projekta



Izvor: izrada autora (20.08.2021.)

2.1.2. Server

Server je projekt tipa ASP.NET Core koji se koristi kao Web API za rad s bazom podataka preko Entity Framework Core-a i SQL Servera. Unutar njega se nalazi datoteka Controllers u kojoj su kontroleri za svaki model podataka kreiranih u Shared projektu. Svaki kontroler samim time i model ima osnovne funkcije GET (dohvaćanje), POST (dodavanje), PUT (ažuriranje) i DELETE (brisanje).

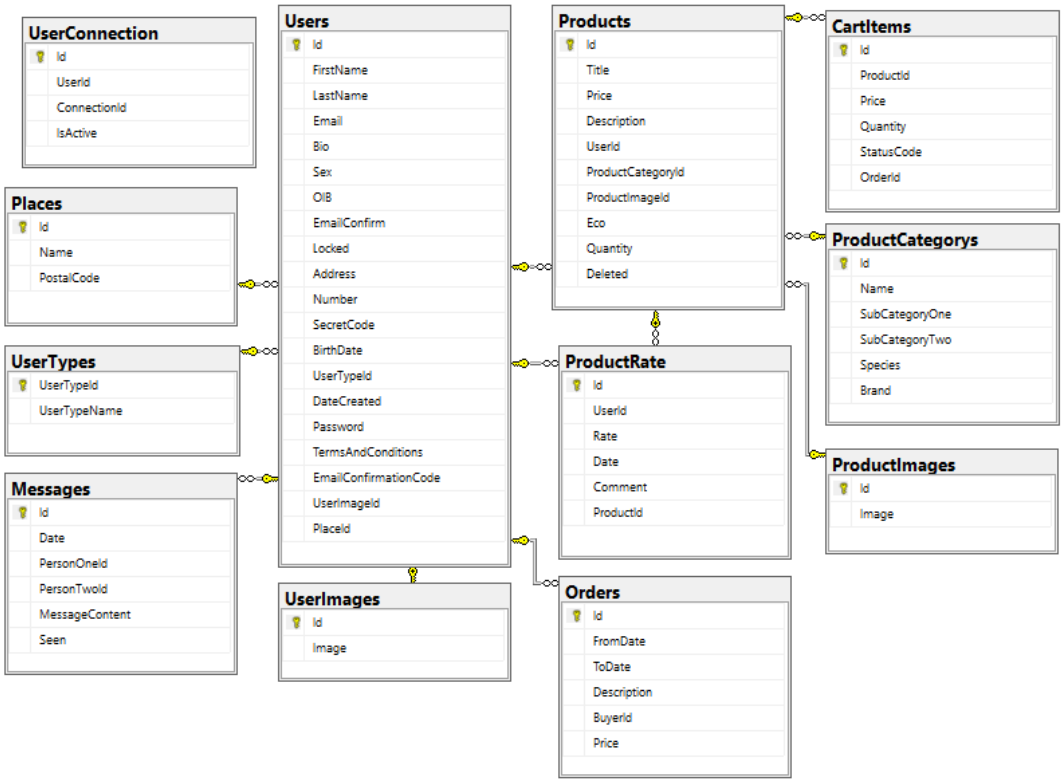
2.1.3. Shared

Unutar Shared projekta kao što smo već spomenuli nalaze se modeli odnosno klase s kojima se radi u Client i Server projektima kako bi se izbjeglo višestruko pisanje istih klasa i njihovog lakšeg ažuriranja. Neki od modela kao što su User za registraciju sadrže Data Annotations koji olakšava pisanje i smanjuje veličinu koda pri validaciji podataka.

2.2. Arhitektura baze podataka

U ovome dijelu će se spominjati baza podataka i na koji je način ona kreirana. Na slici 11. se može vidjeti dijagram baze podataka sa svim tablicama i njihovim relacijama.

Slika 12. Baza podataka



Izvor: izrada autora (20.08.2021.)

2.2.1. Kreiranje tablica

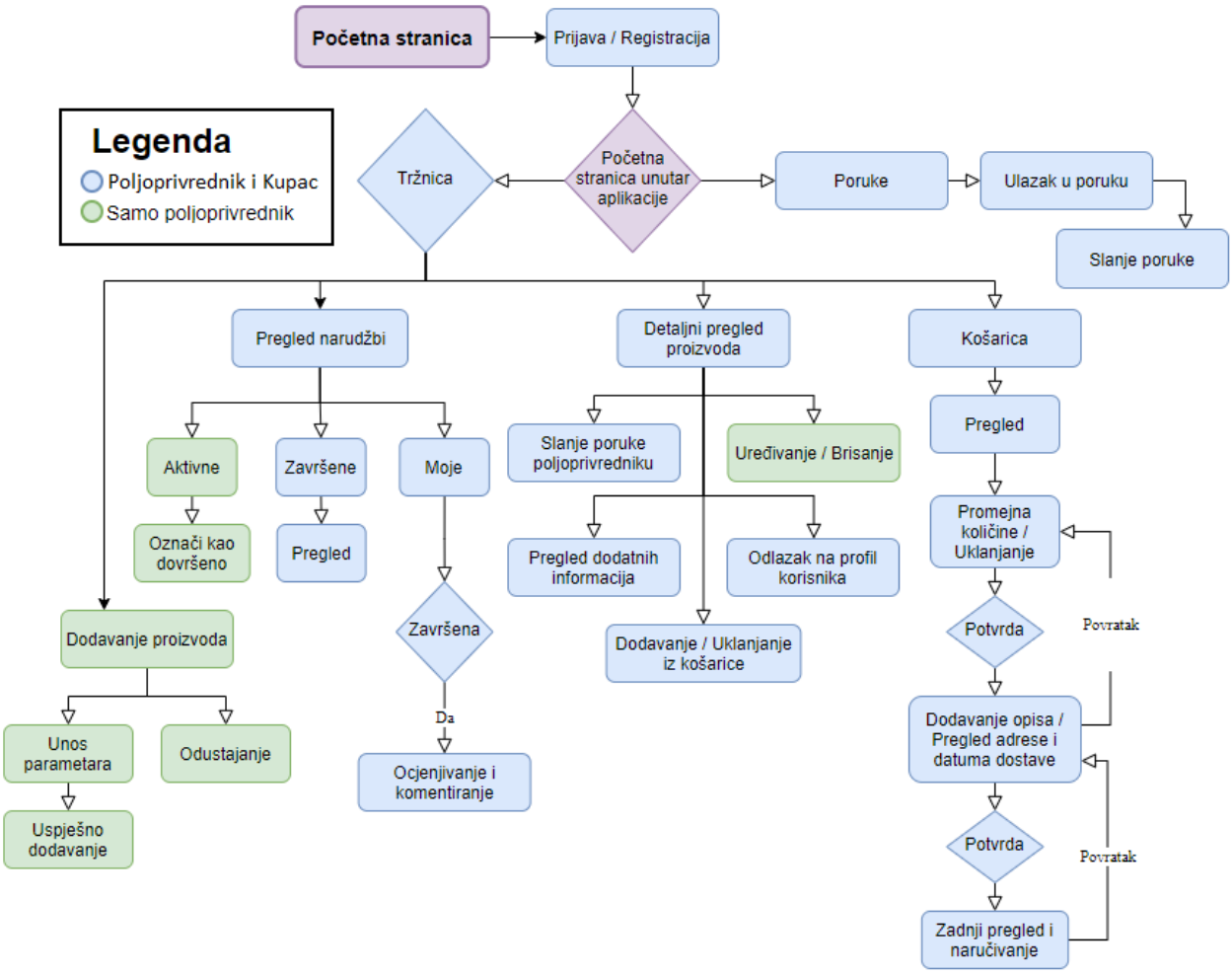
Pri izradi tablica se koristio Entity Framework Core točnije Code First pristup izradi. Za početak je bilo potrebno kreirati klase i dodijeliti svakom atributu pripadajuće tipove unutar Shared projekta. Zatim se unutar Server projekta kreirala DataContext.cs datoteka i u njoj navelo sve klase koje želimo kreirati kao tablice te uz samo dvije komande kreirali bi bazu. Prva komanda je „add-migration [ImeMigracije]“ koja kreira migraciju u kojoj se može vidjeti koje će se tablice kreirati s kojim kolonama i pripadajućih im tipova. Za kraj pokreće se i druga komanda „update-database“ koja će izvršiti tu migraciju.

3. PREGLED FUNKCIONALNOSTI APLIKACIJE

U ovome dijelu rada ćemo detaljno opisati funkcionalnost aplikacije te ju prikazati uz pomoć dijagrama toka. Funkcionalnost je podijeljena na tri dijela po svojoj namijeni: Prijava / Registracija, Tržnica, Poruke. Tržnica je još dodatno podijeljena na dva dijela zbog svoje razlike u namijeni prema različitim vrstama korisnika.

Na slici 12. u nastavku se mogu uočiti najbitnije funkcionalnosti aplikacije. Funkcionalnosti su dodatno obojene kako bi bilo jasnije koja vrsta korisnika ima prava na koju funkcionalnost.

Slika 13. Dijagram toka aplikacije

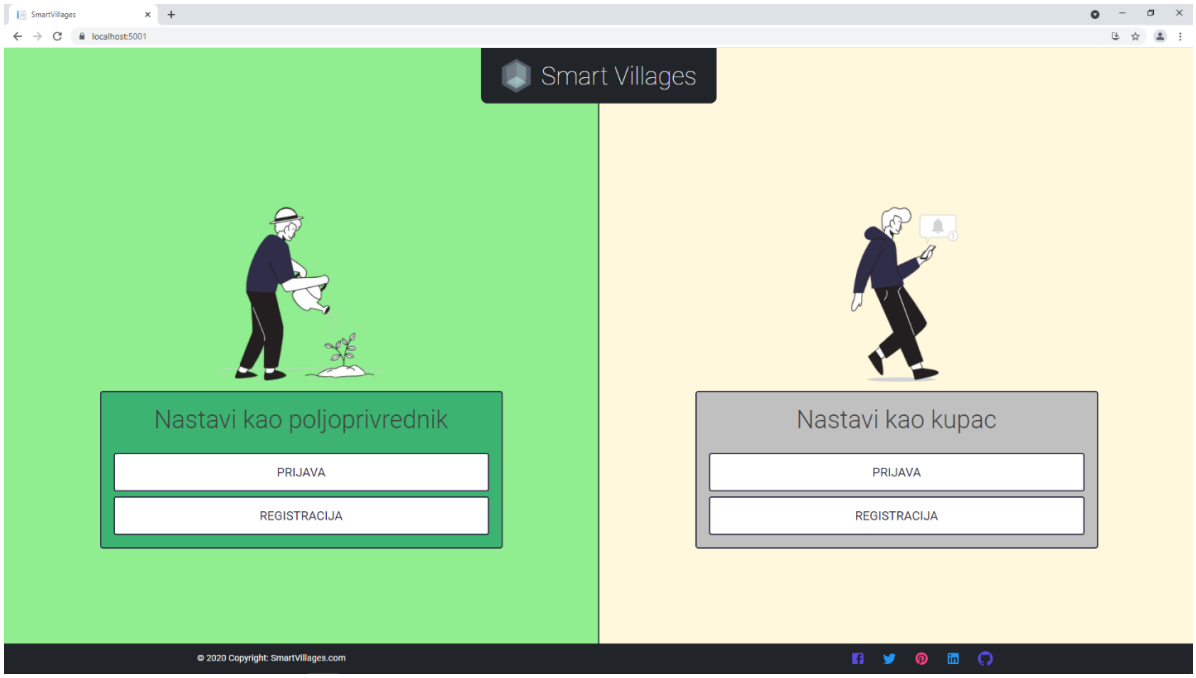


Izvor: izrada autora (20.08.2021.)

3.1. Prijava / Registracija

Prilikom izrade aplikacije realizirane su dvije vrste korisnika Poljoprivrednik i Kupac. Poljoprivrednik je osoba koja će u ovoj aplikaciji imati ulogu prodavatelja svojih proizvoda dok će kupac imati ulogu kupnje upravo poljoprivrednikovih proizvoda. Ako poljoprivrednik želi naručiti proizvod od drugog poljoprivrednika tada on izvršava ulogu kupca. Također prilikom realizacije uzeto je u obzir da će se ta dva korisnika razlikovati i postoji mogućnost promjene načina prijave te je stoga na početnom ekranu aplikacije napravljena posebna forma za svakoga od njih.

Slika 14. Početna stranica



Izvor: izrada autora (22.08.2021.)

Prilikom registracije u sustav korisnik unosi tražene informacije te potom potvrdom pokreće proces registracije. Na uneseni mail korisniku stiže poveznica(link) za potvrdu te kada se on potvrdi stiže mu još jedan mail ovoga puta s tajnim kodom s kojim se potom prijavljuje u sustav uz mail i lozinku. Ako je korisnik unio krive podatke dolazi mu obavijest u pop-up obliku na ekran s porukom.

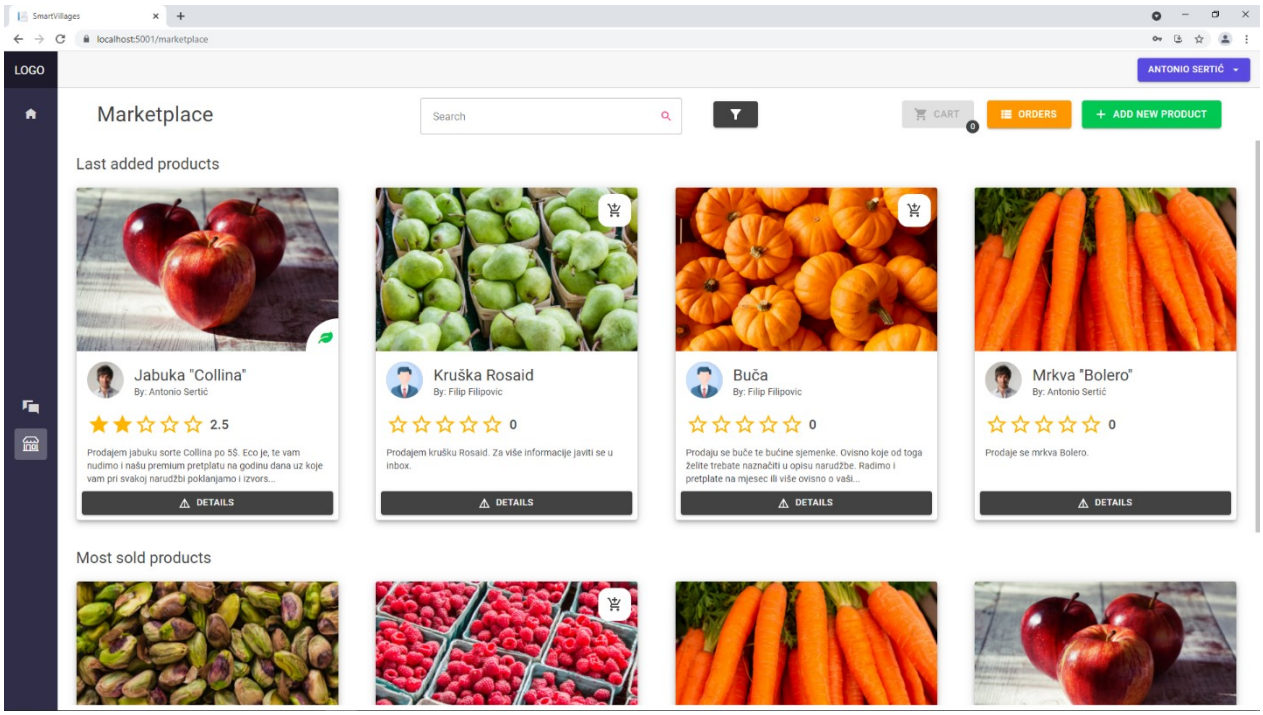
3.2. Tržnica

Obje vrste korisnika nakon prijave u sustav imaju pravo posjetiti tržnicu gdje će ih dočekati kraći popis zadnje dodanih i najprodavanijih proizvoda u tržnici. Kao što je

prikazano na dijagramu sa slike 13. korisnici ovdje imaju opcije: pregleda narudžbi, pregleda košarice, odabir proizvoda kako bi vidjeli više informacija o njemu, filtriranja i pretraživanja te samo poljoprivrednik ima dodatnu opciju dodavanja proizvoda.

Prikaz proizvoda je u kartičnom obliku popraćen modernim dizajnom na kojima se vide osnovne informacije. Uz naziv tu su ocjena, kratki opis, poljoprivrednikovo ime, slika proizvoda, mogućnost dodavanja u košaricu. Dodatna opcija ako je proizvod eko to se može vidjeti zelenom ikonicom lista u donjem desnom dijelu slike proizvoda što je prikazano na prvom proizvodu sa slike 15.

Slika 15. Tržnica



Izvor: izrada autora (22.08.2021.)

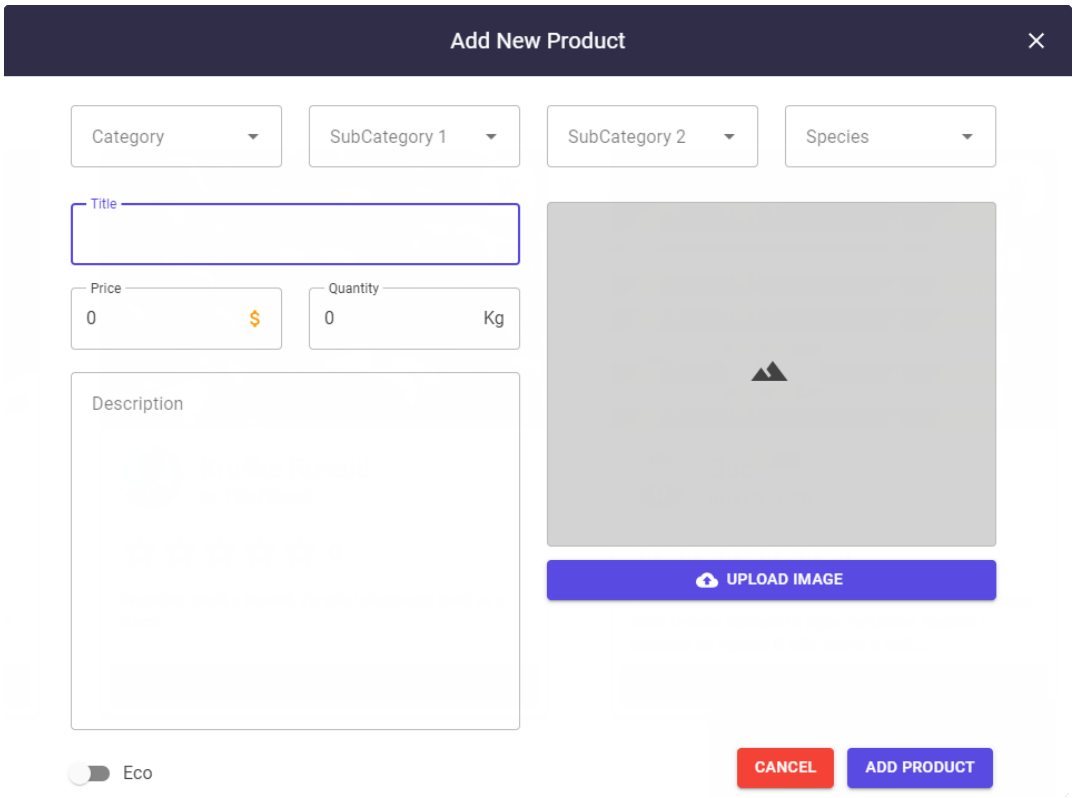
3.2.1. Poljoprivrednikovo dodavanje proizvoda

Poljoprivredniku je u tržnici omogućena dodatna opcija dodavanja proizvoda te nakon njenog odabira otvara se modal s formom za dodavanje prikazan na slici 15. Unutar forme korisnik unosi osnovne ujedno i obvezne informacije o proizvodu. Ako je neko polje ostavljeno prazno korisniku dolazi poruka obavijesti na ekran.

Korisnik u bilo kojem trenutku može odustati od namjere dodavanja proizvoda klikom na „X“ u gorenjem desnom kutu, na gumb na dnu stranice „Odustani“ ili jednim klikom miša bilo gdje izvan okvira modala.

U slučaju da je za neko svojstvo postavljena pogrešna vrijednost korisnik ju može u bilo kojem trenutku ispraviti. To će učiniti odlaskom na detaljni prikaz proizvoda i klikom na gumb Uredi gdje će se otvoriti ista forma kao prilikom dodavanja, ovoga puta popunjena.

Slika 16. Dodavanje proizvoda

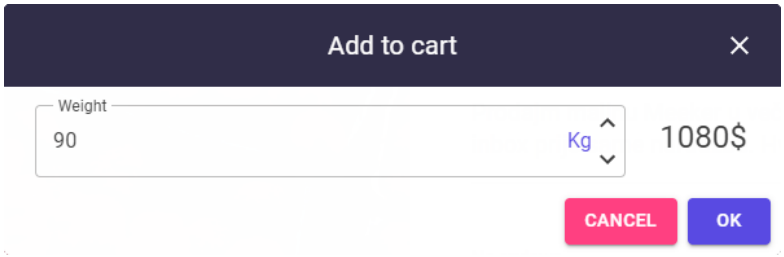


Izvor: izrada autora (22.08.2021.)

3.2.2. Kupčevo naručivanje proizvoda

Korisnik odabirom opcije za dodavanje proizvoda u košaricu započinje proces naručivanja proizvoda te mu se otvara modal za unos količine odabranog proizvoda. Korisnik unosi željenu kilažu te mu se automatski izračuna ukupna cijena za tu kilažu. U slučaju da korisnik unese veću kilažu od one dostupne na proizvodu automatski će mu prepraviti na maksimalnu moguću.

Slika 17. Odabir kilaže

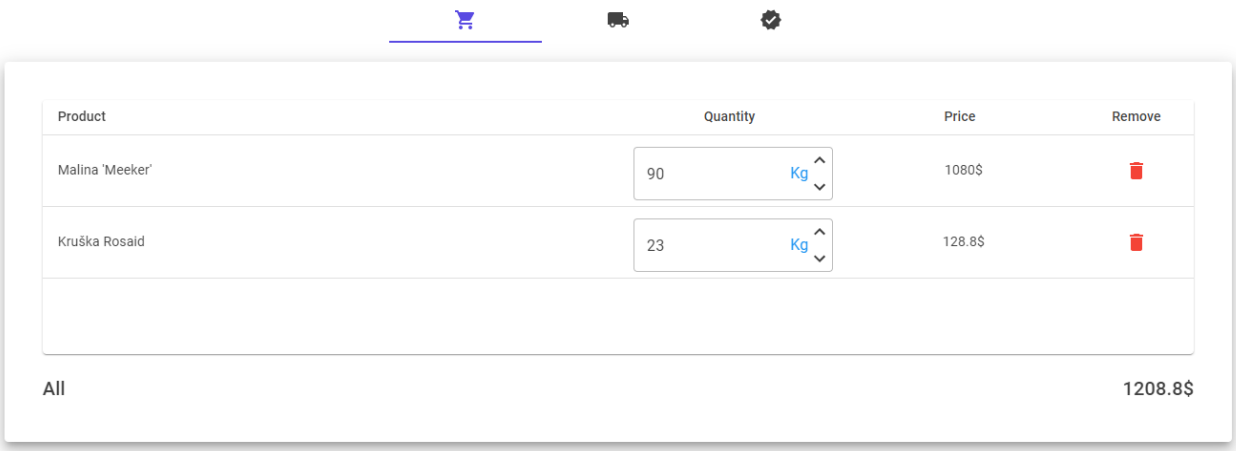


Izvor: izrada autora (22.08.2021.)

Nakon odabira željene kilaže i potvrde pritiskom gumba „OK“, u košaricu se dodaje proizvod te se ona automatski ažurira. Svi proizvodi koji se spremaju u košaricu se spremaju tako da se njihovi objekti dodaju u local storage[9] gdje ostaju sve dok korisnik ne ukloni proizvod iz košarice ili se obrišu svi podaci spremljeni u pregledniku. To je učinjeno iz razloga ako korisnik slučajno izađe sa stranice ili zatvori preglednik da kada se vrati ima i dalje odabrane proizvode.

Odlaskom u samu košaricu korisnik ima pregled svih odabranih proizvoda gdje ih može uklanjati ili im mijenjati kilažu. Promjena kilaže je dodana iz razloga da ako korisnik promjeni mišljenje o željenoj kilaži ne mora uklanjati proizvod i dodavati ga ponovno. Ovdje se može vidjeti i ukupna cijena za sve proizvode koja se automatski ažurira kada nastane promjena.

Slika 18. Košarica

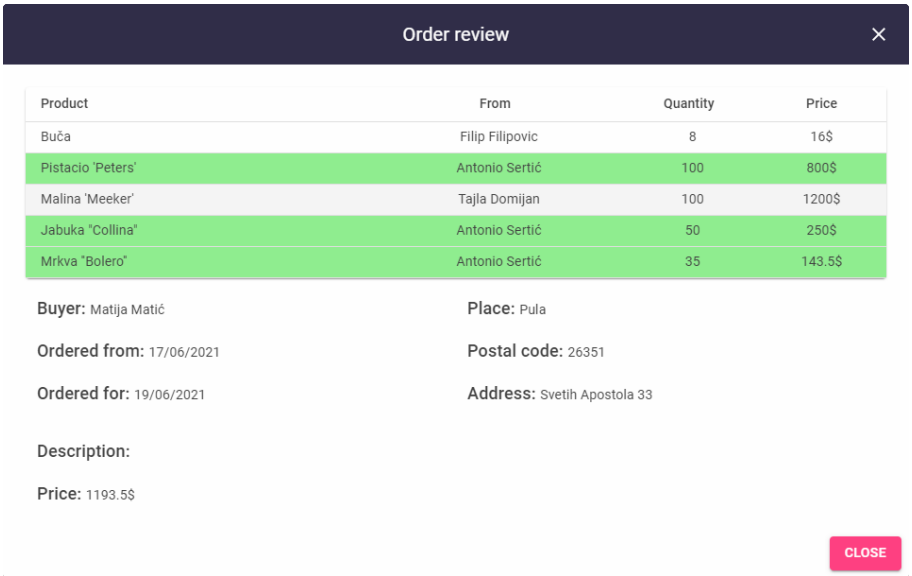


[9] Local storage – eng. lokalna pohrana

Izvor: izrada autora (22.08.2021.)

Nastavkom korisnik pregledava i postavlja detalje dostave kao što su adresa, vrijeme dostave i ukupna cijena te ima opciju dodavanja opisa. Kada korisnik potvrdi narudžbu klikom na zelenu kvačicu kreira se zapis u bazi te svi poljoprivrednici čiji su proizvodi naručeni imaju pregled te narudžbe. Završetkom dostave proizvoda svaki poljoprivrednik ima opciju označiti da su njegovi proizvodi dostavljeni.

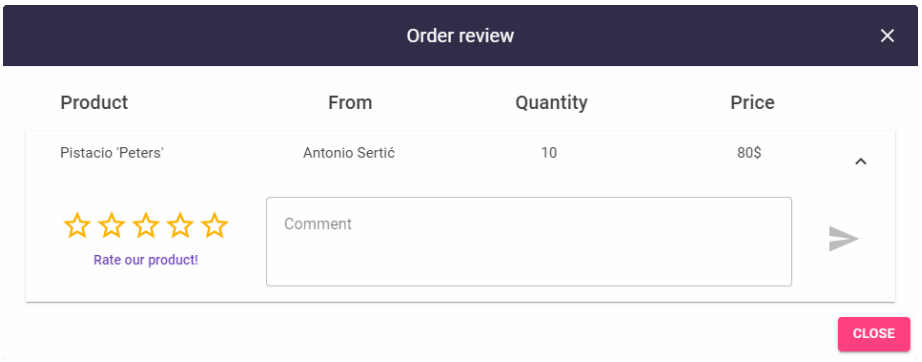
Slika 19. Pregled završene dostave



Izvor: izrada autora (22.08.2021.)

Oznakom svih proizvoda u narudžbi kao dovršenih, proces naručivanja proizvoda je završio te se ostavlja kupcu na odabir želi li ocijeniti i komentirati naručeni proizvod. To može učiniti tako da u tabu Završene narudžbe pregleda narudžbu i klikom na proizvod mu se proširi forma za unos ocjene i komentara. Nakon uspješne potvrde unesenih parametara komentar se prikazuje na stranici detaljnog prikaza proizvoda. Korisnik u bilo kojem trenutku može obrisati ili ažurirati ocjenu i/ili komentar ponovnim odlaskom na narudžbu i odabirom tog proizvoda. Time je korisniku dana mogućnost ocjenjivanja proizvoda po narudžbama jer se svaka može razlikovati.

Slika 20. Ocjenjivanje i komentiranje proizvoda



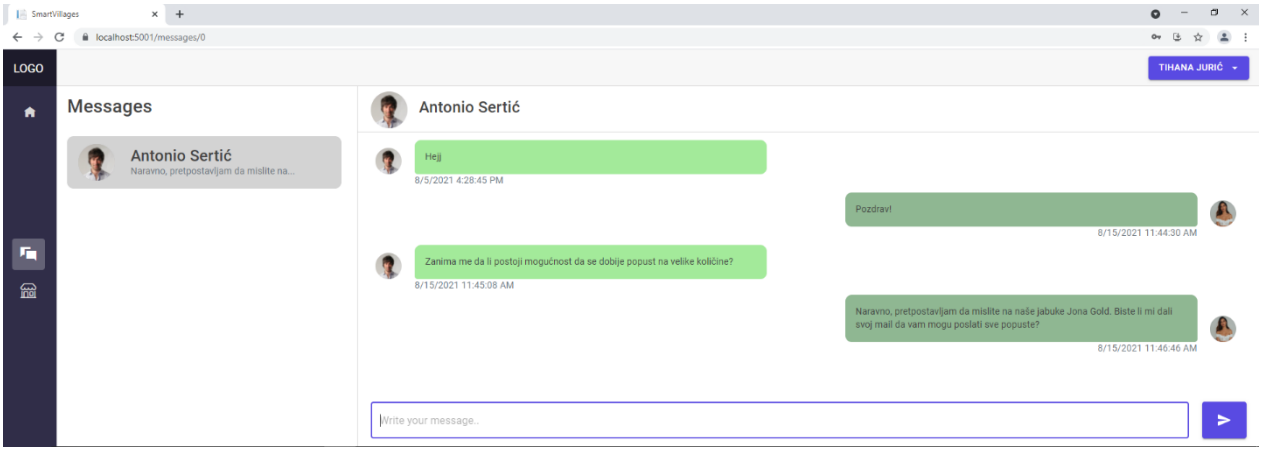
Izvor: izrada autora (22.08.2021.)

3.3. Poruke

Pored tržnice tu se nalazi i stranica za razmjenu poruka između korisnika aplikacije. U bilo kojem trenutku kupac se može javiti poljoprivredniku porukom ako ima neko pitanje ili dodatnih zahtjeva u vezi proizvoda ili narudžbe.

Do stranice poruka se dolazi klikom na ikonicu poruka s bočne navigacijske trake. U lijevom dijelu stranice se nalaze zadnje poruke od svih osoba s kojima se ikad razmijenila poruka u kartičnom obliku te stupčanom poretku. Nepročitane poruke će biti obojene tamnijom bojom. Odabirom jedne od kartica, s desne strane se otvaraju sve poruke s tom osobom.

Slika 21. Poruke



Izvor: izrada autora (22.08.2021.)

Poruke krasi također moderan dizajn koji olakšava snalaženje na stranici. Za slanje nove poruke dovoljno je samo upisati poruku u polje za unos teksta s donje strane ekrana te pritisnuti gumb za slanje koje se nalazi odmah pored njega.

3.3.1. Komunikacija putem poruka u stvarnom vremenu

Uz moderan dizajn poruka, poruke krasi i moderan način rada gdje se poruke šalju i primaju u stvarnom vremenu. Upravo ti događaji su omogućeni u Blazoru putem SignalR-a. Prilikom same prijave u aplikaciju korisnik dobije svoj ConnectionId pomoću kojega se šalju poruke direktno tom ili od tog korisnika. ConnectionId se pohranjuje u bazu zajedno s id-jem korisnika te se tako on dohvaća kada se šalje poruka.

U scenariju kada osoba Jedan šalje poruku osobi Dva s uvjetom da su oboje prijavljeni, dohvaća se ConnectionId osobe Dva iz baze te se prosljeđuje SignalR naredbi[10] uz sadržaj poruke kao parametar. Osoba Dva prima obavijest na ekranu da je zaprimljena nova poruka te ako se nalazi na stranici poruka automatski joj se i ažuriraju poruke s osobom Jedan.

[10] Naredba - `_hubContext.Clients.Client(connectionId).SendAsync("Message");`

4. ZAKLJUČAK

Prije početka razvoja sustava bilo je potrebno pobliže razjasniti problematiku te osmisliti i skicirati potencijalno rješenje. Rješenje osmišljeno na početku nije ostalo isto do kraja, dolazilo je do promjena u načinu rada i dizajnu, ali s tim se načinom razmišljanja i krenulo u realizaciju. Stoga je sama arhitektura sustava i dizajn rađen tako da bude podložan promjenama i nadogradnji.

Poljoprivreda je u Hrvatskoj slabo razvijena i ima dosta problema te dosta kasni za drugim državama članica EU. Trenutno se svuda u svijetu pokušava digitalizirati sve što se može zbog velike nagrade na kraju provedbe koja predstavlja novi način poslovanja u ovom slučaju trgovanja.

U prvom planu rada je bilo osmišljeno puno više funkcionalnosti no zbog pretjeranosti na razini završnog rada su izašle iz plana. Aplikacija bi se svakako dala dodatno nadograditi s jako puno stvari, što zbog same teme tako i kreativnosti autora. Izbor tehnologija, razrada strukture baze podataka, upoznavanje s novom arhitekturom je također doprinijelo razvoju i obogatilo autora novim znanjima.

5. LITERATURA

Knjige:

1. https://aka.ms/blazor-ebook – Daniel Roth, Jeff Fritz, Taylor Southwick, ScottAddie; „Blazor For ASP.NET Web Forms Developers“; Verzija v1.0.2 (13.07.2020.)

Internetski izvori:

1. https://docs.microsoft.com/en-us/dotnet/csharp (17.08.2021.)
2. https://docs.microsoft.com/en-us/dotnet/ (17.08.2021.)
3. https://docs.microsoft.com/en-us/aspnet/core/?view=aspnetcore-5.0 (17.08.2021.)
4. https://webassembly.org (17.08.2021.)
5. https://developer.mozilla.org/en-US/docs/WebAssembly (17.08.2021.)
6. https://docs.microsoft.com/en-us/aspnet/core/blazor/?view=aspnetcore-5.0 (18.08.2021.)
7. https://stackoverflow.blog/2020/02/26/whats-behind-the-hype-about-blazor/ (18.08.2021.)
8. https://blazor-university.com/overview/what-is-blazor/ (18.08.2021.)
9. https://docs.microsoft.com/en-us/aspnet/core/blazor/host-and-deploy/webassembly?view=aspnetcore-3.1#hosted-deployment-with-aspnet-core-2 (18.08.2021.)
10. https://stackoverflow.com/questions/58093386/whats-the-difference-between-asp-net-core-hosted-and-server-side-blazor-really (18.08.2021.)
11. https://docs.microsoft.com/en-us/aspnet/core/signalr/introduction?view=aspnetcore-5.0 (18.08.2021.)
12. https://docs.microsoft.com/en-us/ef/core/ (19.08.2021.)
13. https://docs.microsoft.com/en-us/sql/sql-server/?view=sql-server-ver15 (19.08.2021.)

6. POPIS ILUSTRACIJA

Veleučilište u Virovitici

OBRAZAC 5

IZJAVA O AUTORSTVU

Ja, Antonio Sertić

izjavljujem da sam autor/ica završnog/diplomskog rada pod nazivom

Pametna sela - Smart villages

Svojim vlastoručnim potpisom jamčim sljedeće:
- da je predani završni/diplomski rad isključivo rezultat mog vlastitog rada koji se temelji na mojim istraživanjima i oslanja se na objavljenu literaturu, a što pokazuju korištene bilješke i bibliografija,
- da su radovi i mišljenja drugih autora/ica, koje sam u svom radu koristio/la, jasno navedeni i označeni u tekstu te u popisu literature,
- da sam u radu poštivao/la pravila znanstvenog i akademskog rada.

Potpis studenta/ice

Antonio Sertić

Veleučilište u Virovitici

OBRAZAC 6

ODOBRENJE ZA POHRANU I OBJAVU ZAVRŠNOG/DIPLOMSKOG RADA

Ja Antonio Sertić

dajem odobrenje za objavljivanje mog autorskog završnog/diplomskog rada u javno dostupnom digitalnom repozitoriju Veleučilišta u Virovitici te u javnoj internetskoj bazi završnih radova Nacionalne i sveučilišne knjižnice bez vremenskog ograničenja i novčane nadoknade, a u skladu s odredbama članka 83. stavka 11. Zakona o znanstvenoj djelatnosti i visokom obrazovanju (NN 123/03, 198/03, 105/04, 174/04, 02/07, 46/07, 45/09, 63/11, 94/13, 139/13, 101/14, 60/15, 131/17).

Potvrđujem da je za pohranu dostavljena završna verzija obranjenog i dovršenog završnog/diplomskog rada. Ovom izjavom, kao autor navedenog rada dajem odobrenje i da se moj rad, bez naknade, trajno javno objavi i besplatno učini dostupnim:

a) široj javnosti
b) studentima i djelatnicima ustanove
c) široj javnosti, ali nakon proteka 6 / 12 / 24 mjeseci (zaokružite odgovarajući broj mjeseci).

Potpis studenta/ice

Antonio Sertić

U Virovitici, 23.8.2021.

**U slučaju potrebe dodatnog ograničavanja pristupa Vašem završnom/diplomskom radu, podnosi se pisani obrazloženi zahtjev.*