

Projekat OOAD

eCopy

dokumentacija

Grupa: 9

Tim: Intelligence

Tema: eCopy

Članovi:

Ajla Panjeta

Berina Omerašević

Nudžejma Zukorlić

Sadržaj

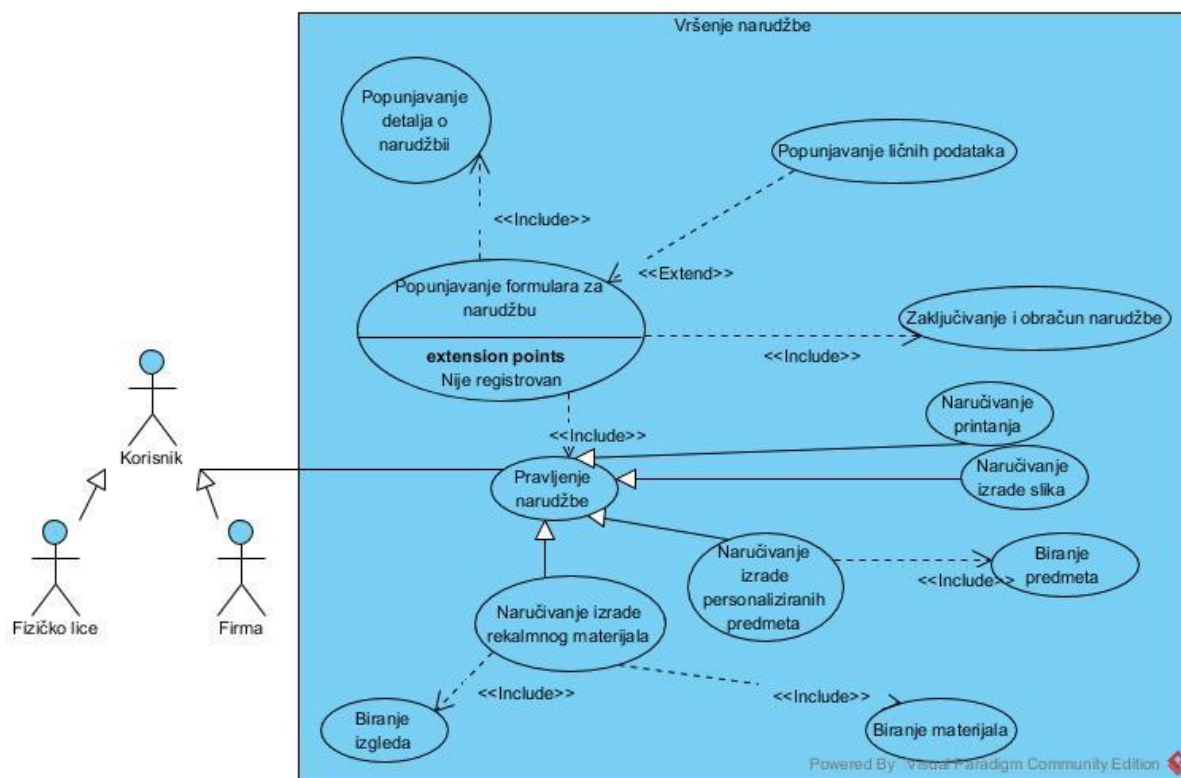
Opis	3
Dijagrami	4
Use Case dijagrami	4
Dijagram klasa (MVVM)	6
Dijagrami aktivnosti	7
Dijagrami sekvenci	10
Dijagrami komunikacija	11
Dijagram komponenti	13
UWP, ASP.NET i REST API	14
Izvještaj o aktivnostima	20
Tabela očekivanih i implementiranih funkcionalnosti	21
Refaktoring	22
Refaktoring pomoću design patterna	24

Opis

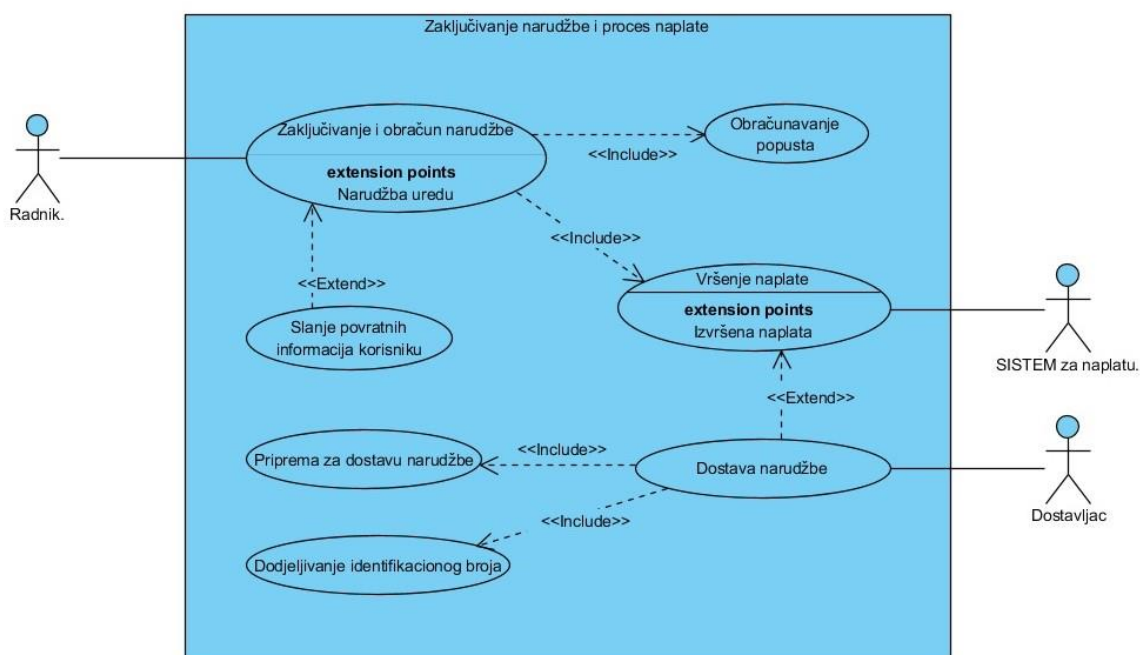
Aplikacija eCopy je aplikacija koja omogućava svojim korisnicima (kupcima) da koriste usluge kopirnice bez potrebe da je posjećuju, tj. online. Naime, zahvaljujući ovoj aplikaciji korisnik može putem svog uređaja (mobilnog telefona, laptopa i sl.), poslati dokumente za printanje, te odabrati način kopiranja koji želi (boja, format, uvez...). Pored usluga printanja na papiru, kopirnica nudi i druge usluge kao što je pravljenje personalizovanih šoljica, majica, termos boca, naljepnica te plakata, na korisniku je samo da putem ove aplikacije odabere željeni predmet, boju, natpis i/ili sliku), a kopirnica(odnosno radnik kopirnice) ima zadatak da obavi ono što je korisnik tražio i pošalje mu proizvod na kućnu adresu (ukoliko je on to odabrao). Kopirnica također nudi usluge izrade reklamnih materijala za firme u vidu printanja na raznim predmetima koje korisnik može odabrati.

Korisnici aplikacije imaju mogućnost da kreiraju vlastiti account kako bi ostvarili razne pogodnosti koje kopirnica nudi, ali i olakšali sebi način naručivanja jer ne bi morali svaki put navoditi svoje podatke i mogli bi izvršiti online plaćanje. Ukoliko ne žele kreirati nalog mogu se prijaviti i kao gost. Pored korisnika (kupaca) i radnici kopirnice imaju svoj account na aplikaciji kako bi mogli primati narudžbe, te slati povratne informacije klijentima.

Konačan projekat se nalazi na GitHub-u unutar foldera Projekat.

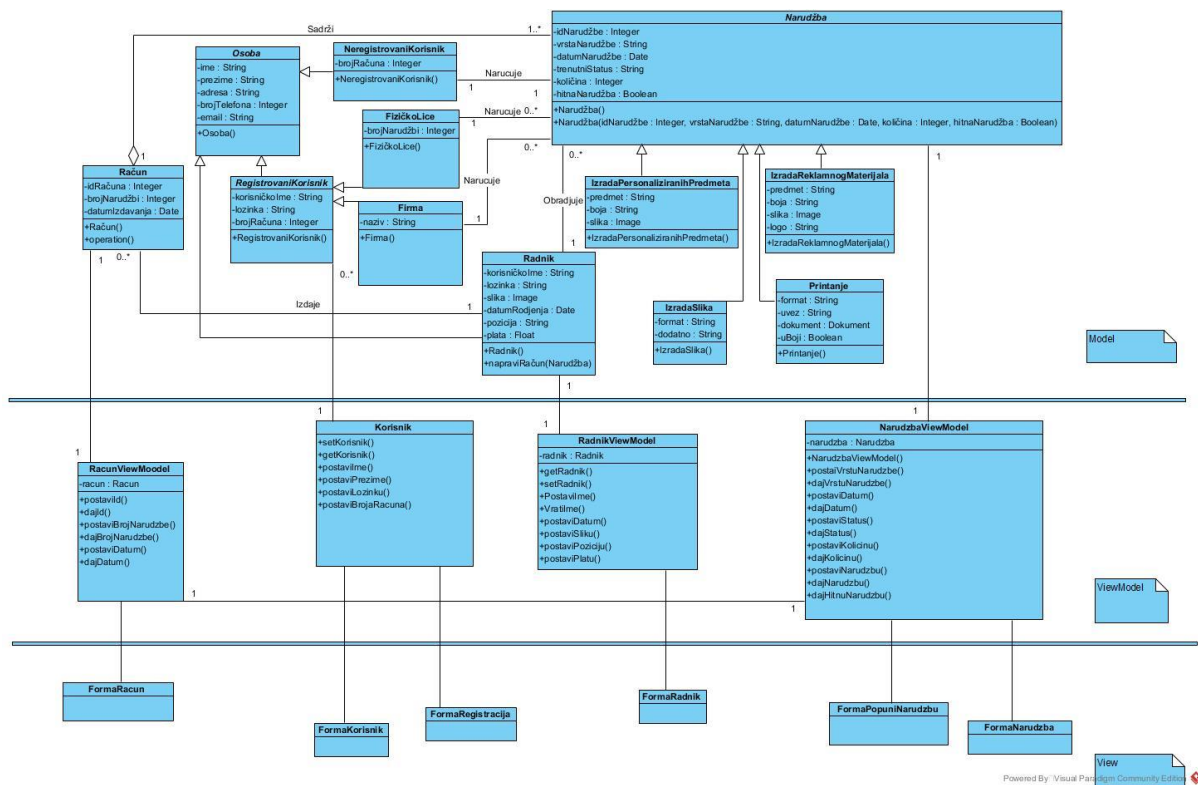


Slika 3. UseCase dijagram za obavljanje narudžbe



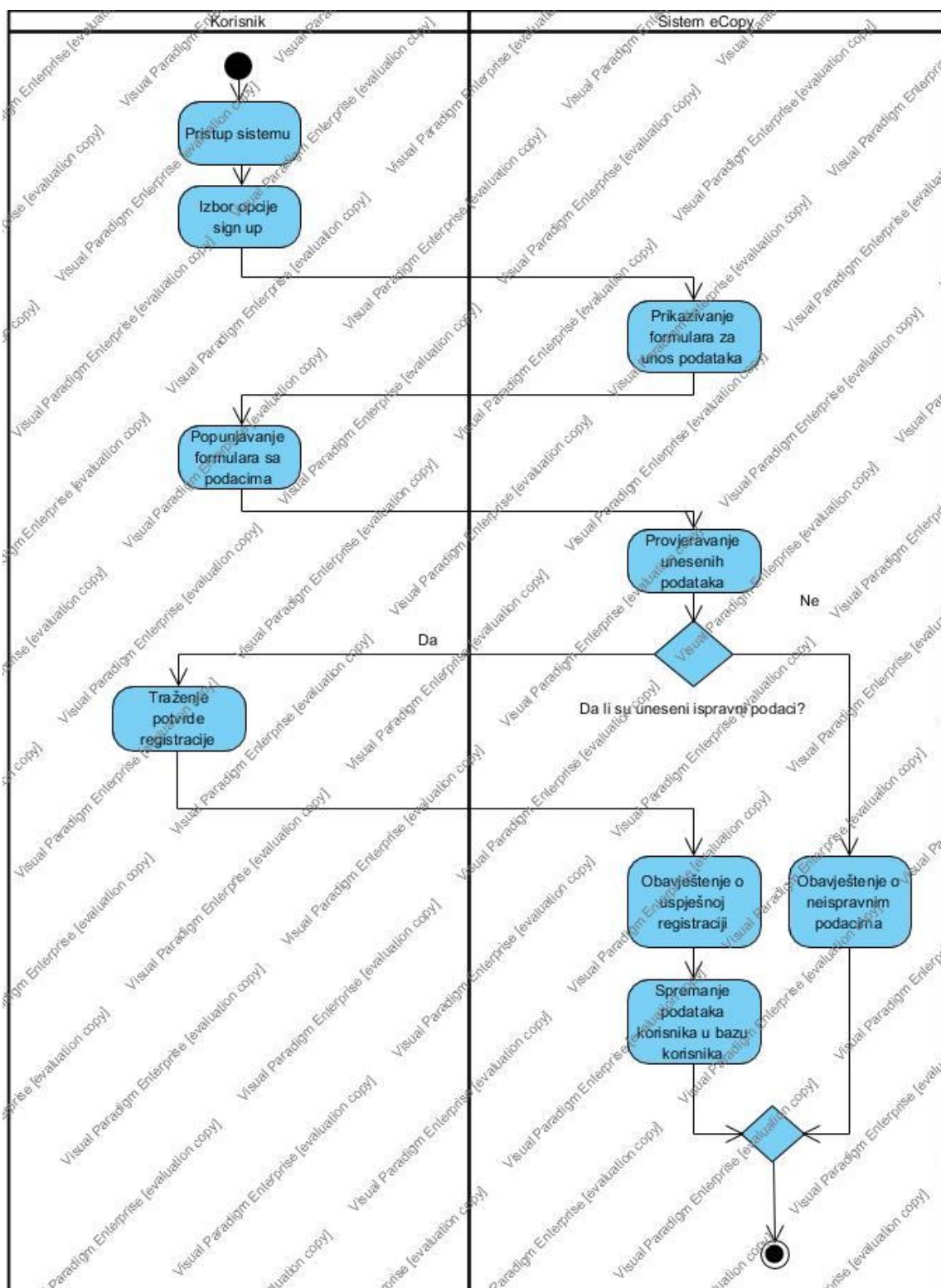
Slika 4. UseCase za obradu narudžbe

Dijagram klasa (MVVM)

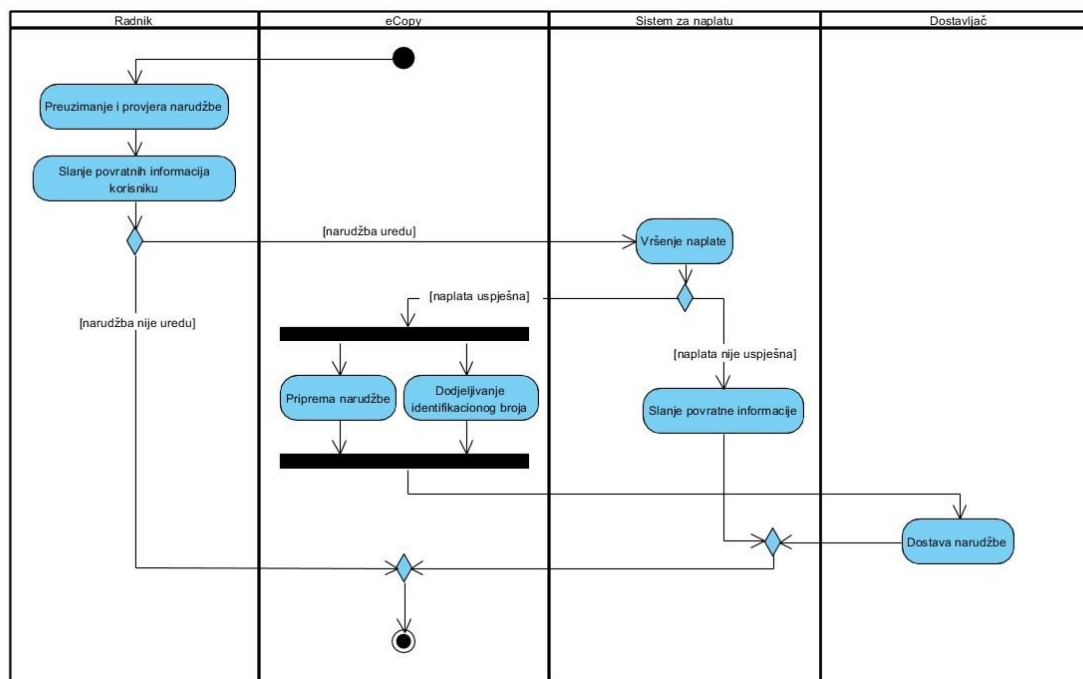


Slika 5. Dijagram klasa

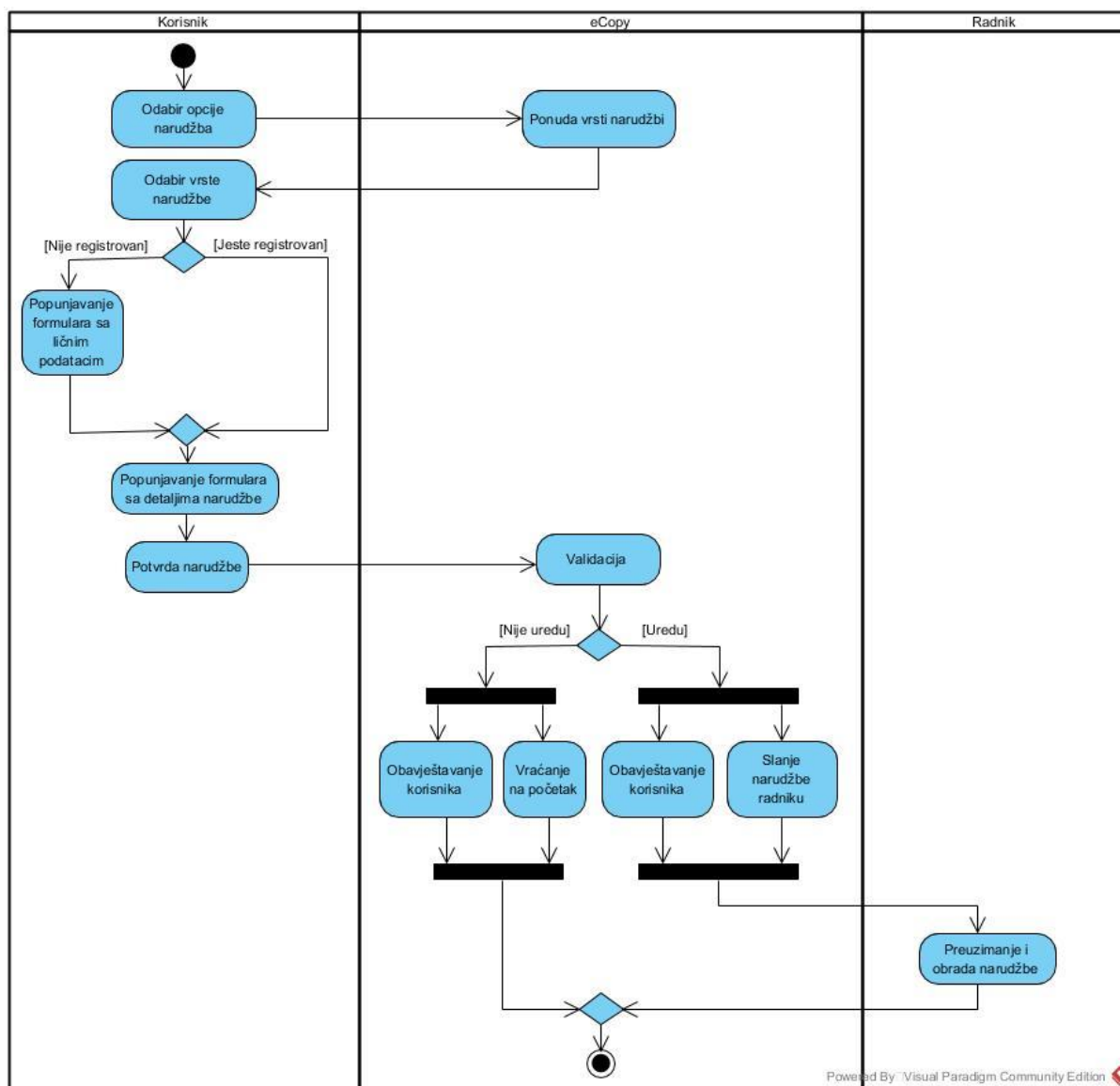
Dijagrami aktivnosti



Slika 6. Dijagram aktivnosti za registraciju

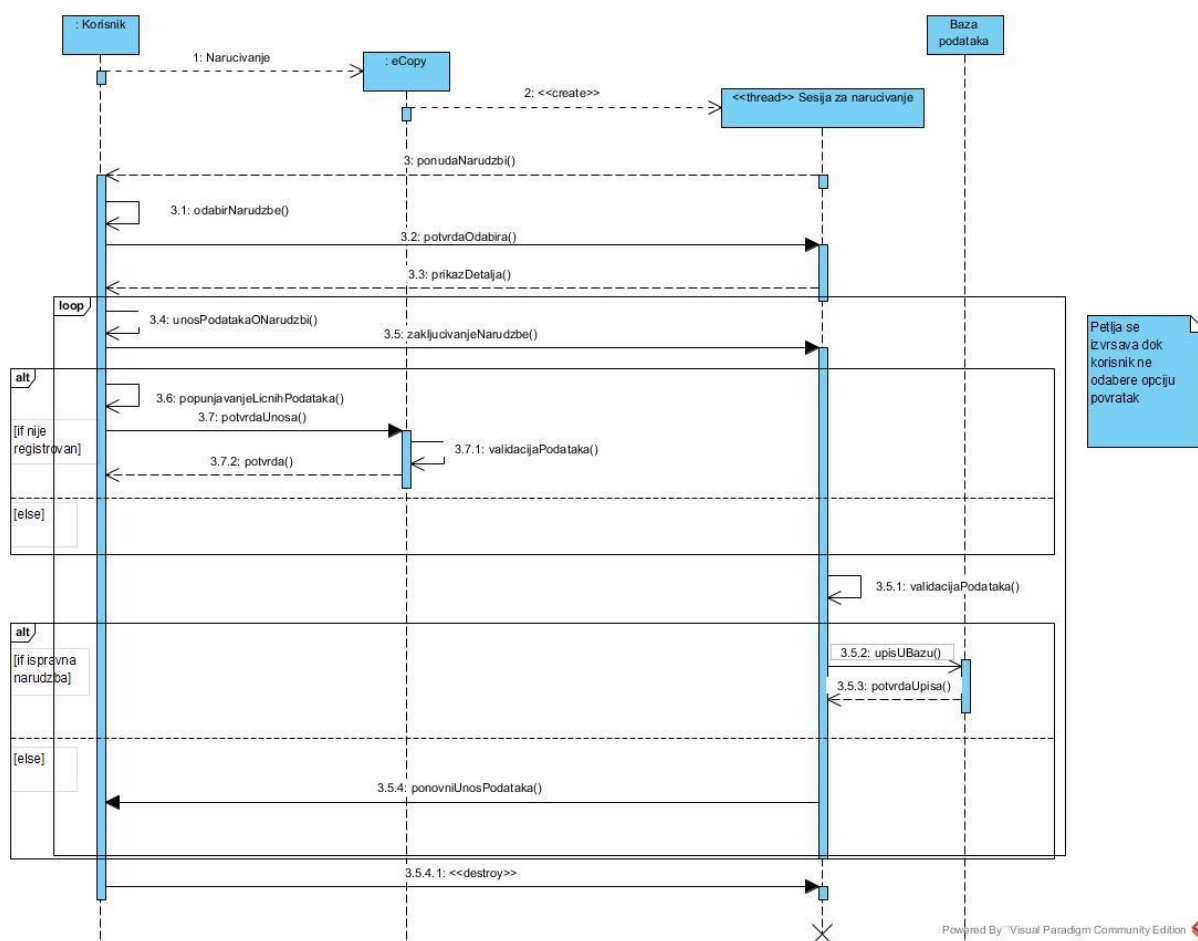


Slika 7. Dijagram aktivnosti za zaključivanje narudžbe

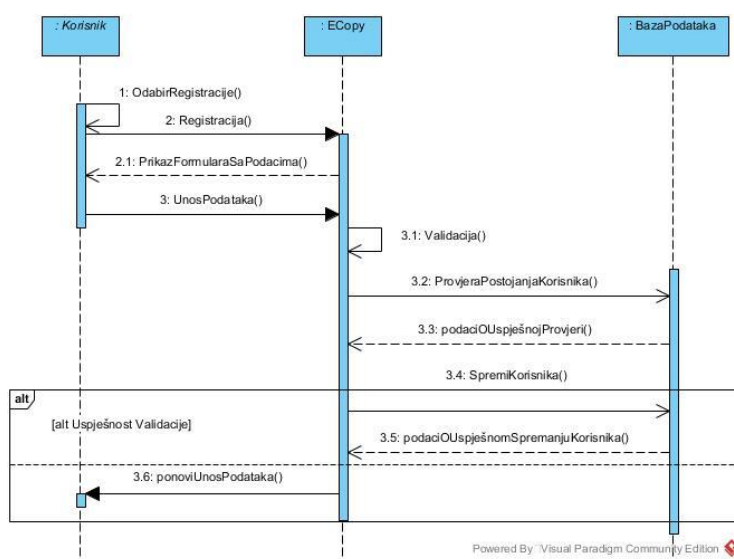


Slika 8. Dijagram aktivnosti za obavljanje narudžbe

Dijagrami sekvenci

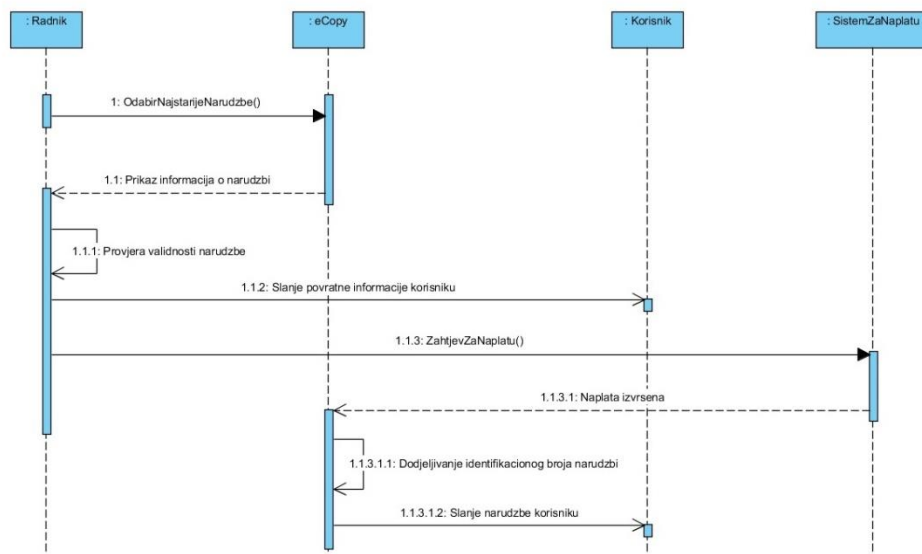


Slika 9. Dijagram sekvenci za obavljanje narudžbe



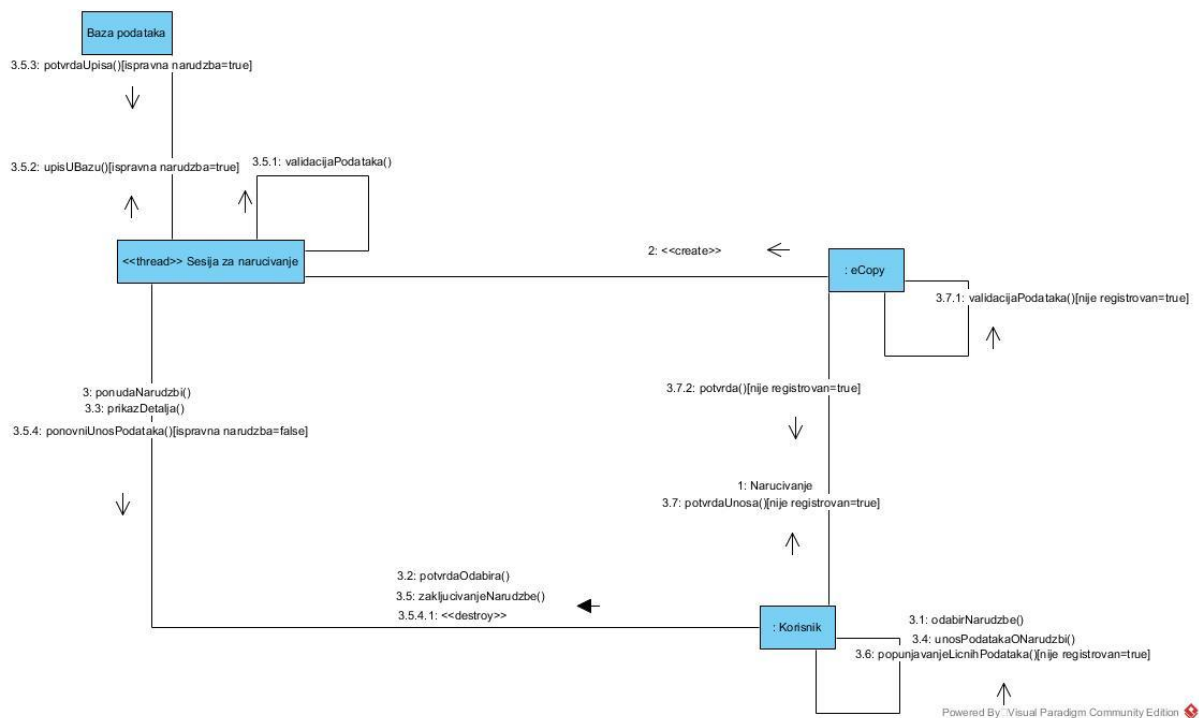
Slika 10. Dijagram sekvenci za registraciju korisnika

sd ZaključivanjeNarudžbeProcesNaplate

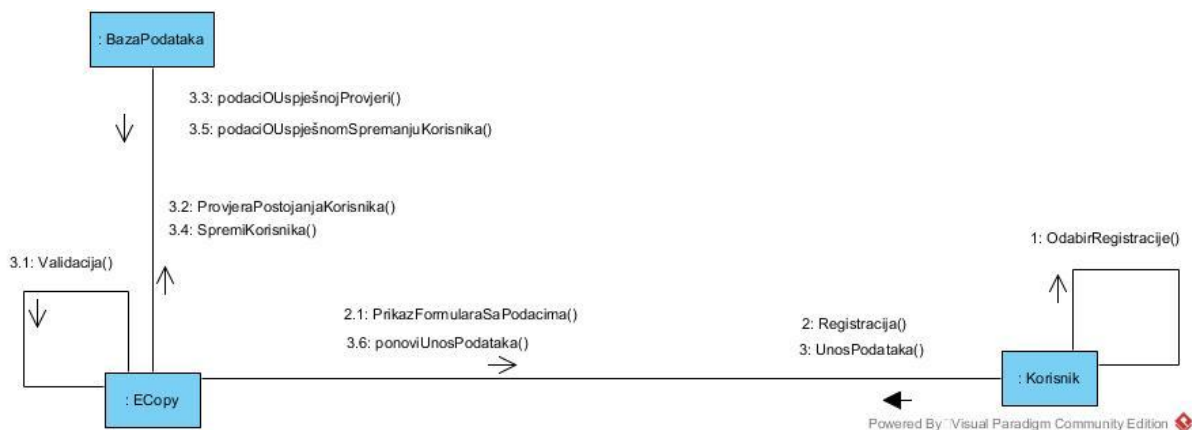


Slika 11. Dijagram sekvenci za proces naplate narudžbe

Dijagrami komunikacija

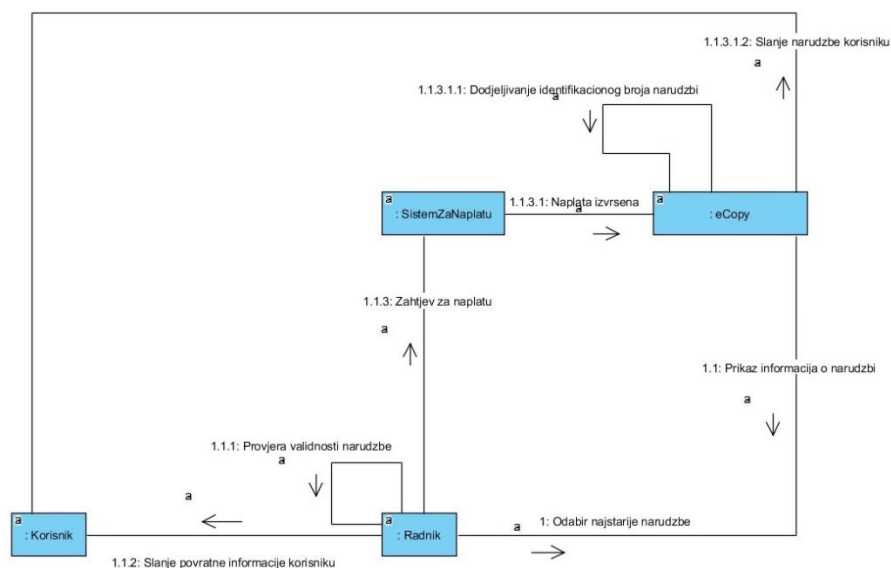


Slika 12. Dijagram sekvenci za obavljanje narudžbe



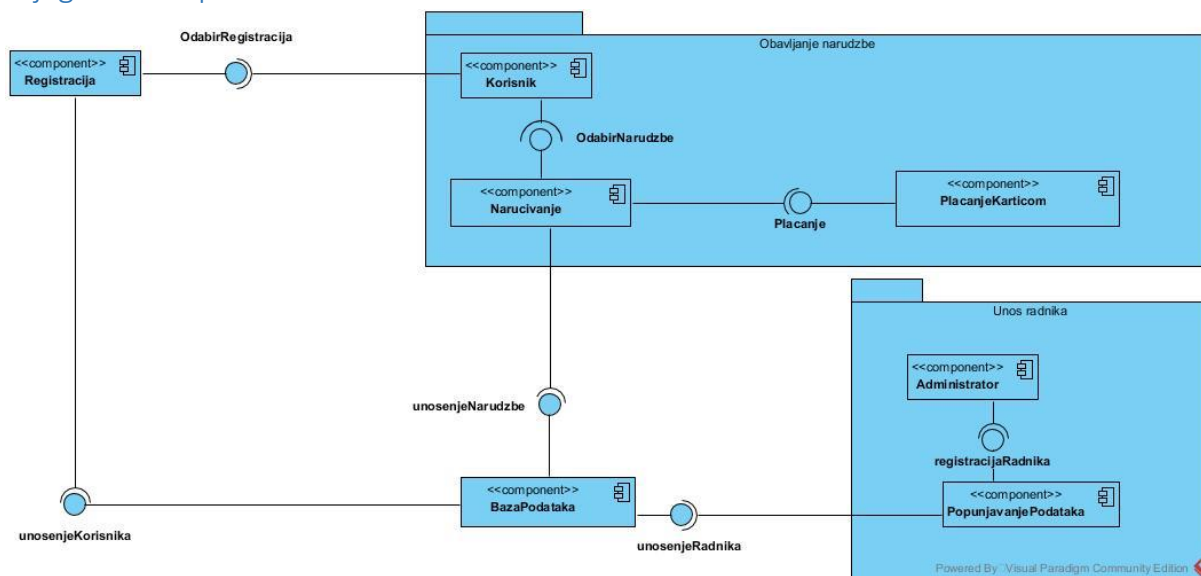
Slika 13. Dijagram komunikacija za registraciju korisnika

sd ZaključivanjeNarudžbeProcesNaplate - Communications

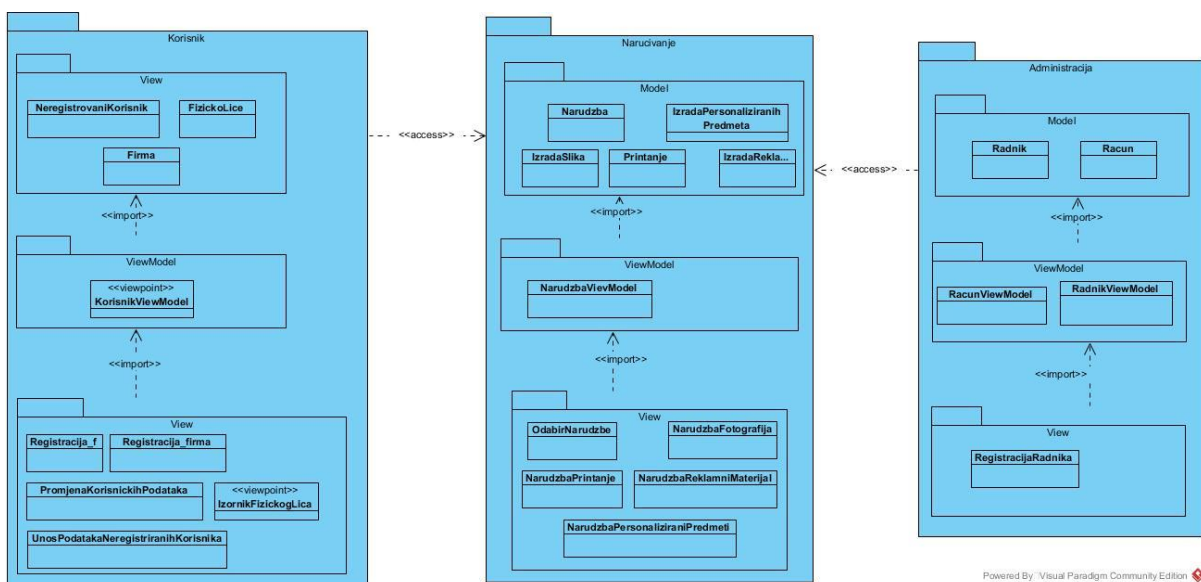


Slika 14. Dijagram komunikacija za proces naplate

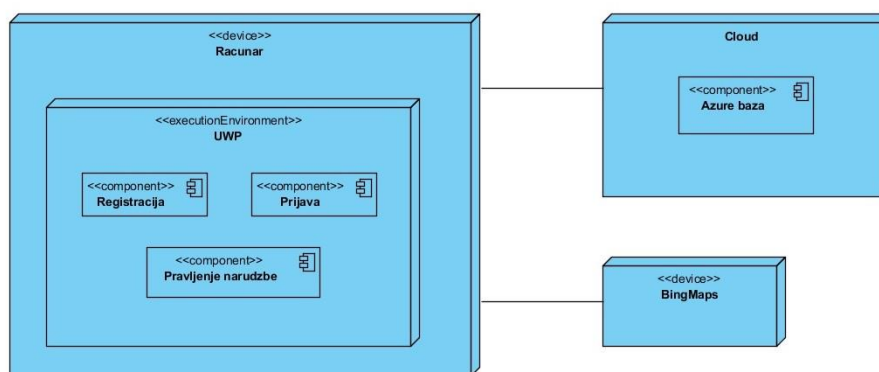
Dijagram komponenti



Slika 15. Dijagram komponenti



Slika 16. Dijagram paketa

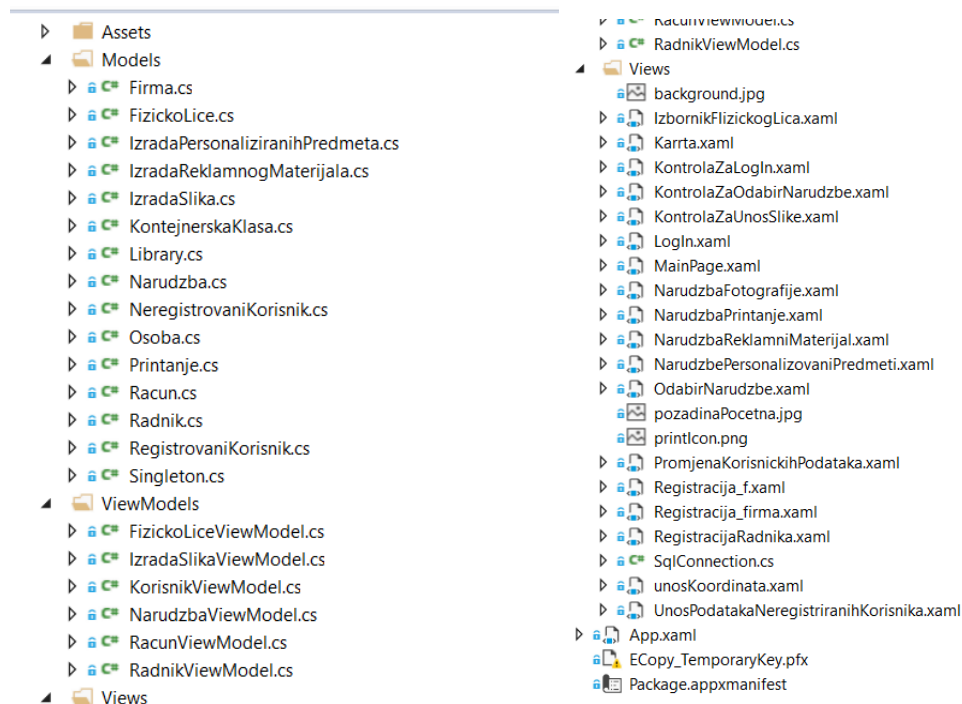


Slika 17. Dijagram raspoređivanja

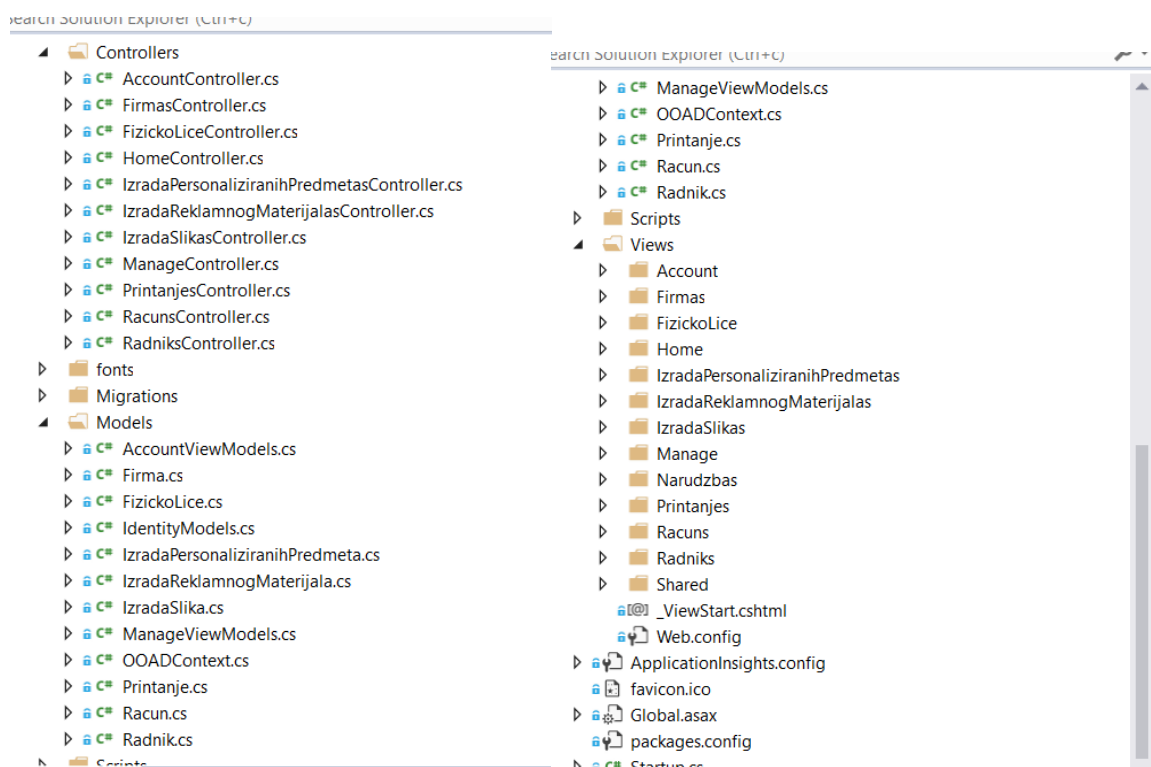
UWP, ASP.NET i REST API

Napravljene sve potrebne klase i view-ovi za rad aplikacije u uwp-u.

Implementiran je i adaptivni interfejs tako da se aplikacija može pokretati na raznim uređajima.

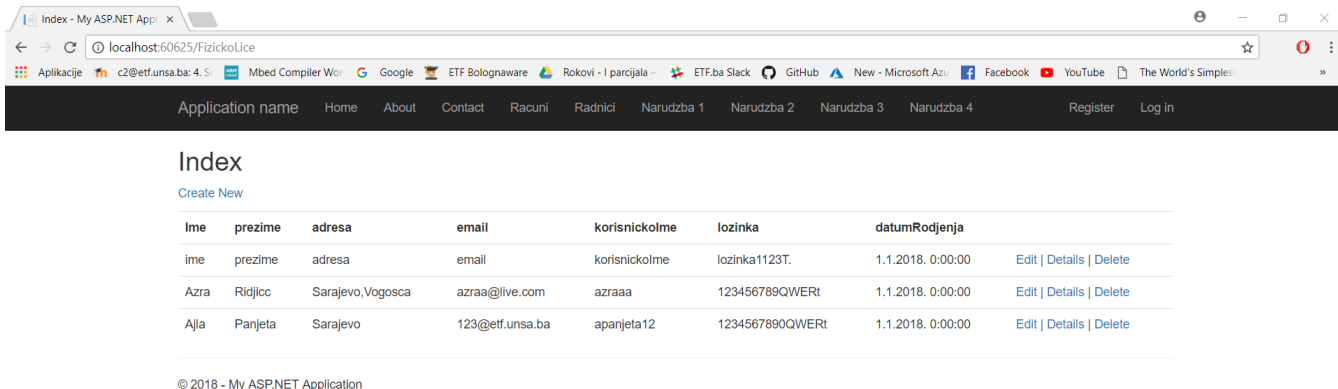


Napravljeni svi modeli, view-ovi i controleri u asp.net-u (postovana MVC arhitektura)



Povezano na bazu eCopyBaza na serveru [oad2018ecopy](#) (na Azuru)

Unos podataka u tu bazu može se vršiti pomoću UWP aplikacije korištenjem http requesta (Potrebno je da bude pokrenu ASP.NET u pozadini)




Index

[Create New](#)

Ime	prezime	adresa	email	korisnickolme	lozinka	datumRodjenja	
ime	prezime	adresa	email	korisnickolme	lozinka1123T.	1.1.2018. 0:00:00	Edit Details Delete
Azra	Ridjic	Sarajevo,Vogosca	azraa@live.com	azraaa	123456789QWERT	1.1.2018. 0:00:00	Edit Details Delete
Ajla	Panjeta	Sarajevo	123@etf.unsa.ba	apanjeta12	123456789QWERT	1.1.2018. 0:00:00	Edit Details Delete

© 2018 - My ASP.NET Application

Slika 18. Podaci u bazi prije unosa



Registracija firme

Datum rođenja: 8 June 2018

Ime: Nudzejma

Prezime: Zukorlic

Adresa: Sarajevo

Email: nzukorlic1@etf.unsa.ba

Korisnicko ime: nzukorlic

Lozinka:

Potvrda lozinke:

[Povratak](#) [Potvrdi](#)

Slika 19. Unos u UWP-u

Index

Create New

Ime	prezime	adresa	email	korisnickolme	lozinka	datumRodjenja	
ime	prezime	adresa	email	korisnickolme	lozinka1123T	1.1.2018. 0:00:00	Edit Details Delete
Azra	Ridjic	Sarajevo, Vogosca	azraa@live.com	azraaa	123456789QWERT	1.1.2018. 0:00:00	Edit Details Delete
Aja	Panjeta	Sarajevo	123@etf.unsa.ba	apanjeta12	123456789QWERT	1.1.2018. 0:00:00	Edit Details Delete
Nudzajma	Zukoric	Sarajevo	nzukoric1@etf.unsa.ba	nzukoric	asdfghjkl0	1.1.2018. 0:00:00	Edit Details Delete

© 2018 - My ASP.NET Application

Slika 20. Baza nakon unosa

Dio koda koji obavlja ovu funkcionalnost nalazi se u ViewModelu FizickoLiceViewModel u metodi registruj

```
namespace ECopy.ViewModels
{
    class FizickoLiceViewModel
    {
        private ECopy.Models.FizickoLice korisnik;
        private static readonly HttpClient client = new HttpClient();
        public static string httpRequest = "http://ecopyrestapi20180606065004.azurewebsites.net/";

        public async Task<bool> Registruj(string ime, string prezime, string adresa, string email,
            string korisnickoIme, string lozinka, DateTime datum)
        {
            Windows.Web.Http.HttpClient httpClient = new Windows.Web.Http.HttpClient();
            var headers = httpClient.DefaultRequestHeaders;
            string header = "ie";
            if (!headers.UserAgent.TryParseAdd(header))
            {
                throw new Exception("Invalid header value: " + header);
            }
            header = "Mozilla/5.0 (compatible; MSIE 10.0; Windows NT 6.2; WOW64; Trident/6.0)";
            if (!headers.UserAgent.TryParseAdd(header))
            {
                throw new Exception("Invalid header value: " + header);
            }
            //string stri = Convert.ToBase64String(Image);
            //MessageDialog showDialog = new MessageDialog(ime + prezime + email + adresa);
            //await showDialog.ShowAsync();
            korisnik = new Models.FizickoLice(ime, prezime, adresa, email, korisnickoIme, lozinka, datum);

            Uri requestUri = new Uri("http://localhost:60625/FizickoLice/Add/" + "?ime=" + korisnik.Ime +
                "&prezime=" + korisnik.prezime + "&adresa=" + korisnik.adresa + "&email=" + korisnik.email +
                "&korisnickoIme=" + korisnik.korisnickoIme + "&lozinka=" + korisnik.lozinka + "&datum=" +
                korisnik.datumRodjenja);

            Windows.Web.Http.HttpClient httpClient = new Windows.Web.Http.HttpClient();
            string httpResponseBody = "";

            try
            {
                //var success = await Windows.System.Launcher.LaunchUriAsync(requestUri);
                httpResponse = await httpClient.PostAsync(requestUri, null);
            }
        }
    }
}
```

Slika 21. Naznaceni dio sa http requestom

Na isti način je realizovano: unosenje Firme (druga vrsta korisnika), unosenje radnika, unosenje narudžbe (trenutno je implementirano samo za jednu vrst narudžbe, narudžbaSlika)

Implementiran ASP.NET web api service (napravljen je novi projekat eCopyRestAPI)

Tu su ponovo dodani kontroleri, modeli i view-ovi za sve klase, te za svaku klasu omogućene metode GET, POST, PUT i DELETE, pomoću kojih se može vršiti rad sa bazom (unošenje, čitanje, promjena...)

Omogućen i poziv WEB API servisa iz UWP aplikacije, za sve one klase (prethodno nabrojane) za koje je moguće unositi podatke, mogu se čitati podaci pomoću GET API-ja u UWP-u

Dio koda koji to pokazuje nalazi se u UWP-u unutar FizickoLiceViewModel klase (prikazano samo za fizičko lice u ovom slučaju iako je implementirano i za ostale):

```
,
public async Task<bool> nadjikorisnika(string username, string lozinka)
{
    List<FizickoLice> korisnici = new List<FizickoLice>();
    using (var client = new HttpClient())
    {
        client.BaseAddress = new Uri(httprequest);
        client.DefaultRequestHeaders.Clear();

        //definisanje formata koji želimo prihvatiti
        client.DefaultRequestHeaders.Accept.Add(new
            MediaTypeWithQualityHeaderValue("application/json"));
        //Asinhrono slanje zahtjeva za podacima o studentima

        HttpResponseMessage Res = await client.GetAsync("api/FizickoLice/");
        //Provjera da li je rezultat uspješan
        if (Res.IsSuccessStatusCode)
        {
            //spremanje podataka dobijenih iz responsa
            var response = Res.Content.ReadAsStringAsync().Result;

            //Deserijalizacija responsa dobijenog iz apija i pretvaranje u listustudenata
            korisnici = JsonConvert.DeserializeObject<List<FizickoLice>>(response);
        }
        foreach (FizickoLice f in korisnici)
        {
            if (f.korisnickoIme.Equals(username) && f.lozinka.Equals(lozinka)) return true;
        }
        return false;
    }
}
```

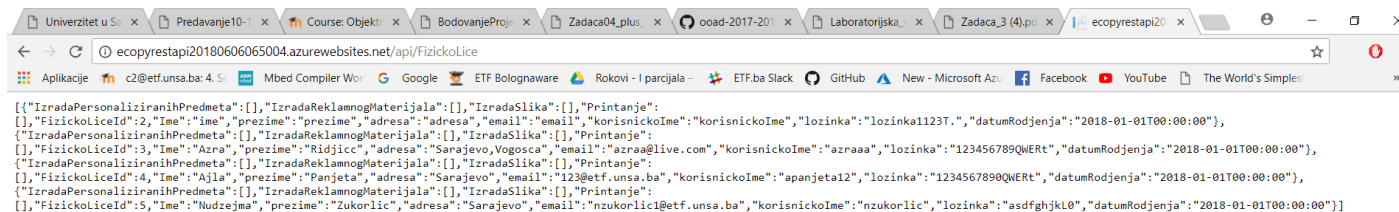
Slika 21.

String httprequest je u ovom slučaju

```
public static string httprequest=
"http://ecopyrestapi20180606065004.azurewebsites.net/";
```

Ovo je link koji se dobije nakon publish-a aplikacije na Azure.

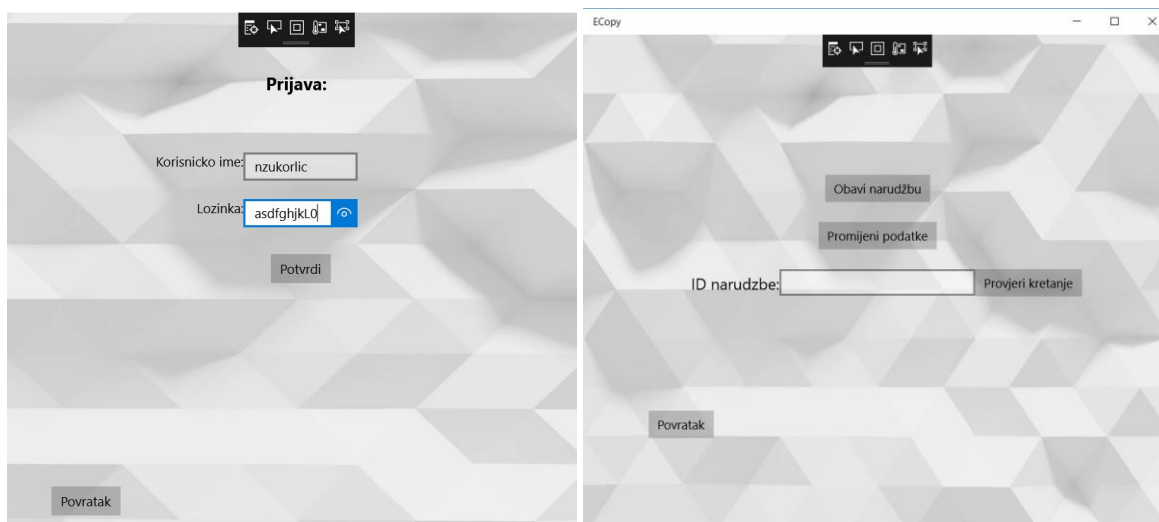
Možemo vidjeti da kucanjem ovog linka na web browser-u sa dodatkom api/FizickoLice dobijamo sve elemente te tabele u json formatu (Upravo se iz UWP-a vrši čitanje iz JSON formata)



Vidimo da se i uneseni podatak za FizičkoLice (koji smo maloprije unijeli) automatski dodao i ovdje.

Kada se pokušamo u uwp-u prijaviti kao korisnik sa ovim podacima, otvara nam se view za korisnika, što znači da aplikacija ispravno nalazi podatke i pazi pomoću API-ja

Pogledajmo na sljedećim slikama:



Slika 22. Prikaz login-a

Dakle aplikacija se sastoji iz UWP dijela, ASP.NET i RestApi (gdje je implementiran web service i uradjen publish aplikacije).

Sva tri dijela su povezana preko jedne baze, tako kad unesemo novog korisnika, radnika, narudžbu.. u UWP aplikaciji, te podatke možemo vidjeti u ASP.NETu i na linku

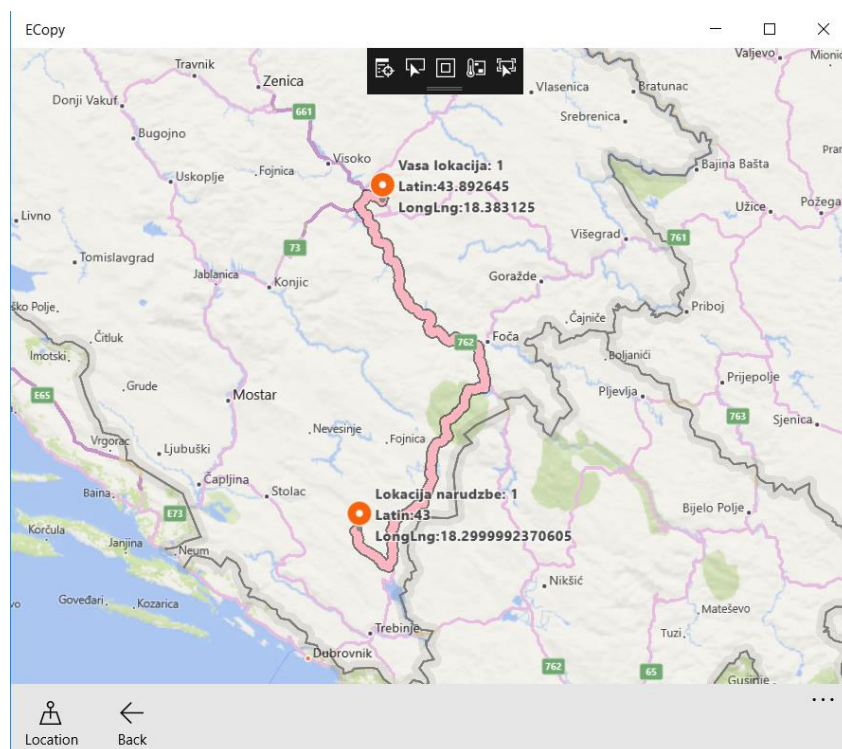
<http://ecopyrestapi20180606065004.azurewebsites.net/>

Takođe je moguće pristupiti tim podacima i mijenjati ih iz UWP aplikacije

Od specifičnih funkcionalnosti, implementirana je mapa koja prikazuje mjesto gdje se nalazi korisnik i njegova narudžba, te put između te dvije lokacije. Korisnik unosi id narudžbe i prikazuje mu se mapa, koordinate narudžbe mijenja radnik(dostavljač), ukoliko narudžba jos nije „krenula“ prikazuje se da se nalazi u kopirnici.

Slika 23. View pomoću kojeg radnik unosi koordinate određene narudžbe (on te koordinate treba da učita pomoću gps uređaja, ali to nije implementirano u aplikacij i- za potrebe testiranja unosimo koordinate ručno)

Radnik na svom profilu može unijeti narudžbu i kliknuti Provjeri kretanje button (Taj view vidimo na slici 22) nakon čega mu se otvara mapa kao na slici (na slici su učitane koordinate koje je unio radnik)



Izveštaj o aktivnostima

Redni br.	Aktivnost	Ucesnici	Vrijeme	Napomena
1.	Određivanje funkcionalnosti	Zajedno	14.03.2018.	
2.	Izrada svih klasa i view-ova u UWP-u	Zajedno	25.03.2018. – 04.04.2018.	
3.	Unos novog radnika u UWP	Berina	04.04.2018.	
4.	Unos novog korisnika (fizickoLice) u UWP	Ajla	07.04.2018.	
5.	Unos novog korisnika (firma) u UWP	Nudžejma	09.04.2018.	
6.	Unos nove narudžbe (za izradu slika) u UWP	Ajla	26.05.2018.	
7.	Log in kao administrator u UWP	Berina	09.04.2018.	
8.	Log in kao korisnik u UWP	Nudžejma	09.04.2018.	
9.	Unos koordinata narudžbe u UWP	Ajla	02.06.2018.	
10.	Prikaz „puta“ narudžbe na karti (unosjenjem id-a) i crtanje rute na karti (UWP) i očitavanje trenutne lokacije na karti (u UWP i ASP.NET)	Ajla	03.06.2018.	
11.	Implementirano sve u ASP.NET-u	Zajedno	26.05.2018.	
12.	Veza sa Azure bazom (iz ASP.NET-a, eCopyRestAPI i UWP-a)	Zajedno	26.05.2018.	
13.	Implementacija ASP.NET WEB API servisa (kreiran projekat eCopyRestApi gdje je to implementirano)	Ajla	05.06.2018.	
14.	Upis i čitanje svih podataka u istu bazu pomoću WEB API servisa	Ajla	05.06.2018.	
15.	Deployment ASP.NET WEB API servisa na AZURE	Ajla	06.06.2018.	
16.	Dokumentacija	Zajedno	08.06.2018.	

Uradjene su sve zahtjevane aktivnosti.

Tabela očekivanih i implementiranih funkcionalnosti

Akter	Funkcionalnost	Implementirano
Korisnik (fizičko lice)	Prijava	Da
Korisnik (fizičko lice)	Izmjena podataka	Da
Korisnik (fizičko lice)	Obavljanje narudžbe	Donekle (urađeno samo za jednu vrstu narudžbe, ostale se implementiraju slično)
Korisnik (sve vrste)	Pregled „puta“ narudžbe na mapi	Da
Korisnik(firma)	Prijava	Da
Korisnik(firma)	Izmjena podataka	Da
Korisnik(firma)	Obavljanje narudžbe	Ne
Gost	Obavljanje narudžbe	Donekle (ista funkcionalnost kao i za korisnika)
Gost	Registracija kao fizičko lice	Da
Gost	Registracija kao firma	Da
Radnik	Pregled i obrada narudžbe	Ne
Radnik	Unos koordinata za narudžbu	Da
Administrator	Unos novog radnika	Da
Administrator	Dodavanje novih pogodnosti	Ne

Refaktoring

- Zamjena magičnih brojeva sa konstantama

S obzirom da korisnik ima mogućnost da vidi na karti dokle je njegova narudžba stigla, potrebno je cuvati koordinate same kopirnice u varijablama, jer, ukoliko narudžba nije još poslana, korisnik treba vidjeti da je ona u kopirnici. Da bi kod bio čitljiviji, najbolje rješenje jeste korištenje magičnih brojeva, tj. čuvanje tih vrijednosti u nekim varijablama sa jasnim nazivom (a i ukoliko dođe u nekom trenutku do promjene lokacije, lakše ih je pronaći i zamijeniti novim).

```

1  /// </summary>
2  public sealed partial class Karrrta : Page
3  {
4
5      public const double lokacijaKopirniceLongitude = 18.3770994009617;
6      public const double lokacijaKopirniceLatitude = 43.8911123782028;
7      public Models.Library Library = new Models.Library();
8      public int indexPosition = 1;
9      public Karrrta()
10     {
11         this.InitializeComponent();
12     }
13     protected async override void OnNavigatedTo(NavigationEventArgs e)
14     {
15         double lo=lokacijaKopirniceLongitude;
16         double la=lokacijaKopirniceLatitude;
17         foreach (Narudzba n in KontejnerskaKlasa.narudzbe)
18         {
19             if (n.IdNarudzbe == KontejnerskaKlasa.i) { lo = n.pozicijaLongitude; la = n.pozicijaLatitude; }
20         }
21         Geopoint myPoint = await Library.Position();
22         eCopyMapa.ZoomLevel = 16;
23         eCopyMapa.Center = myPoint;
24         dodajIkonuNaPoziciju(myPoint, "Vasa lokacija: " + indexPosition);
25         BasicGeoposition pozicijaNarudzbe = new BasicGeoposition();
26         pozicijaNarudzbe.Longitude = lo;
27         pozicijaNarudzbe.Latitude = la;
28         //KopirniceLongitude = 18.3770994009617;

```

- Za komplikovan uslovni (if-then-else) iskaz izdvojiti metodu za uslov dio

Ovaj refaktoring je poželjan radi bolje čitljivosti koda i lakših izmjena u kodu.

Prije refaktoringa:

```

1  private async void Potvrdi_Click(object sender, RoutedEventArgs e)
2  {
3      var picker = new Windows.Storage.Pickers.FileOpenPicker();
4      picker.ViewMode = Windows.Storage.Pickers.PickerViewMode.Thumbnail;
5      picker.SuggestedStartLocation = Windows.Storage.Pickers.PickerLocationId.PicturesLibrary;
6      picker.FileTypeFilter.Add(".jpg");
7      picker.FileTypeFilter.Add(".jpeg");
8      picker.FileTypeFilter.Add(".png");
9
10     Windows.Storage.StorageFile file = await picker.PickSingleFileAsync();
11     if (file != null)
12     {
13         using (IRandomAccessStream fileStream = await file.OpenAsync(Windows.Storage.FileAccessMode.Read))
14         {
15             // Set the image source to the selected bitmap
16             BitmapImage bitmapImage = new BitmapImage();
17             bitmapImage.DecodePixelWidth = 600; //match the target Image.Width, not shown
18             await bitmapImage.SetSourceAsync(fileStream);
19             slika1.Source = bitmapImage;
20         }
21     }
22     else
23     {
24         var message = new MessageDialog("Ne moze se naci ruta");
25         await message.ShowAsync();
26     }
27 }

```

Poslije refaktoringa

```
private async void ucitajSliku(Windows.Storage.StorageFile file)
{
    using (IRandomAccessStream fileStream = await file.OpenAsync(Windows.Storage.FileAccessMode.Read))
    {
        // Set the image source to the selected bitmap
        BitmapImage bitmapImage = new BitmapImage();
        bitmapImage.DecodePixelWidth = 600; //match the target Image.Width, not shown
        await bitmapImage.SetSourceAsync(fileStream);
        slika1.Source = bitmapImage;
    }
}

private async void Potvrdi_Click(object sender, RoutedEventArgs e)
{
    var picker = new Windows.Storage.Pickers.FileOpenPicker();
    picker.ViewMode = Windows.Storage.Pickers.PickerViewMode.Thumbnail;
    picker.SuggestedStartLocation = Windows.Storage.Pickers.PickerLocationId.PicturesLibrary;
    picker.FileTypeFilter.Add(".jpg");
    picker.FileTypeFilter.Add(".jpeg");
    picker.FileTypeFilter.Add(".png");

    Windows.Storage.StorageFile file = await picker.PickSingleFileAsync();
    if (file != null) ucitajSliku(file);
    else
    {
        var message = new MessageDialog("Ne može se naći ruta");
        await message.ShowAsync();
    }
}
```

- Polimorfizam i nasljeđivanje kod narudžbi

Nasljeđivanje se koristi kako bi se izbjegli složeni if iskazi i switch case. Takođe, nasljeđivanjem se program otvara za nadogradnju (uveđenje novih funkcionalnosti i slično).

- Ponavljanje dijelova koda u različitim view-ovima

U svim view-ovima se koristi isti dio za validaciju gdje se koriste if i else if iskazi pa je taj dio odvojen u posebnu metodu koja se sad poziva više puta. Time je postignuta veća čitljivost koda te izbjegnuto ponavljanje istog dijela koda više puta. Takođe, sada je jednostavnije praviti promjene jer ih je potrebno vršiti samo na jednom mjestu.

Dio koda prije izvršenog refaktoringa:

```
string potvrda = potvrdasifrebox.Password.ToString();
//DateTime datum = (DateTime) date.Date;

greska1.Foreground = new SolidColorBrush(Colors.Red);

if (adresa.Length == 0 || email.Length == 0 || korisnicko.Length == 0)
{
    greska1.Text = "Morate popuniti sva polja!";
}
else if (lozinka.Length <= 3)
{
    greska1.Text = "Lozinka mora imati više od tri znaka!";
}
else if (korisnicko.Length <= 3)
{
    greska1.Text = "Korisničko ime mora imati više od tri znaka!";
}
else if (!email.Contains("@") || !email.Contains("."))
{
    greska1.Text = "Neispravan format email!";
}
else if (!lozinka.Equals(potvrda))
{
    greska1.Text = "Lozinke se ne podudaraju!";
}

else
{
    // ...
}
```

Dio koda nakon refaktoringa:

```
greska1.Foreground = new SolidColorBrush(Colors.Red);

if (MainPage.Validacija(adresa, email, korisnicko, lozinka, potvrda, greska1))
{
    greska1.Text = " ";
    Boolean b = await flvm.promjenaPassword(models.FizickoLice.trenutniUser, lozinka);
    if (b)
    {
        MessageDialog showDialog = new MessageDialog("Promjena šifre uspješno završena");
        await showDialog.ShowAsync();
    }
}
```

Refaktoring pomoću design patterna

Singleton patern - KREACIJSKI PATERNI

Uloga Singleton paterna je da osigura da se klasa može instancirati samo jednom i da osigura globalni pristup kreiranoj instanci klase. Singleton pattern koristimo kod klase FizickoLiceViewModel jer je za svako korištenje te klase potrebna samo jedna instanca.

Klasa FizickoLiceViewModel je Singleton klasa i sadrži mehanizam za jedinstveno instanciranje same sebe. Unutar klase je private static varijabla (uniqueInstance) koja čuva jednu/jedinstvenu instancu klase, static metoda (getInstance) preko koje se pristupa Singleton klasi.

```
namespace ECopy.ViewModels
{
    class FizickoLiceViewModel
    {
        private ECopy.Models.FizickoLice korisnik;
        private static readonly HttpClient client = new HttpClient();
        public static string httprequest = "http://ecopyrestapi20180606065004.azurewebsites.net/";

        // static varijabla koja čuva instancu Singleton klase
        private static FizickoLiceViewModel uniqueInstance;
        // privatni konstruktor klase-samo ova tj. Singleton klasa može instancirati ovaj objekat
        private FizickoLiceViewModel() { }
        public static FizickoLiceViewModel getInstance()
        {
            if (uniqueInstance == null)
            {
                // instanciranje objekta Singleton klase
                //ako nikad nije potrebna instanca neće biti nikad kreirana-lazy implementation
                uniqueInstance = new FizickoLiceViewModel();
            }
            return uniqueInstance;
        }

        public async Task<bool> Registruj(string ime, string prezime, string adresa, string email,
            string korisnickoIme, string lozinka, DateTime datum)
        {

```


	Refaktoring	Implementiro/la
1.	Zamjena magičnih brojeva sa konstantama	Zajedno
2.	Za komplikovan uslovni (if-then-else) iskaz izdvojiti metodu za uslov dio	Zajedno
3.	Polimorfizam i nasljeđivanje kod narudžbi	Zajedno
4.	Korištenje razumljivih i jasnih naziva klasa, atributa i metoda	Zajedno
5.	Ponavljjanje dijelova koda na različitim dijelovima zamjenjeno jednom metodom koja se poziva više puta	Ajla Panjeta
6.	Singleton patern	Ajla Panjeta