UNIVERZITET U SARAJEVU

FAKULTET ZA SAOBRAĆAJ I KOMUNIKACIJE

PROJEKTNI RAD IZ PREDMETA:

Razvoj aplikacija mobilnih uređaja

Tema rada:	Mobilna aplikacija za naručivanje u restoranu

Predmetni nastavnik:	Doc. dr. Asmir Butković dipl.ing.el.
Student:	Durmo Merima, Agić Selma
Broj indeksa:	7816, 7924
Usmjerenje:	KIT
Godina studija:	Treća(III)
Rezultat rada:	

Datum: 25.6.2021

Sadržaj

Uvod

Svakome se nekad desilo da dođe u restoran u vrijeme kad je najveća gužva i da dugo čeka čak i da konobar dođe da uzme narudžbu a još više vremena da narudžba dođe do stola zbog iste te gužve.

Takve situacije mogu odvratiti goste od dolaska u restoran i navesti ih da izaberu neki drugi restoran. Zbog toga bi aplikacija za mobilno narucivanje u restoranu bila jako korisna.

Mogla bi poboljšati poslovanje restorana. Mobilna aplikacija za narucivanje u restoranu kreirana je uz koristenje .NET razvojne platforme Xamarin u Microsoftovom razvojnom okruženju Visual Studio 2019.

Za izradu je korištena aplikacijska arhitektura Model-View-ViewModel.

U nastavku će biti prikazana aplikacija, objašnjen način korištenja i kod.

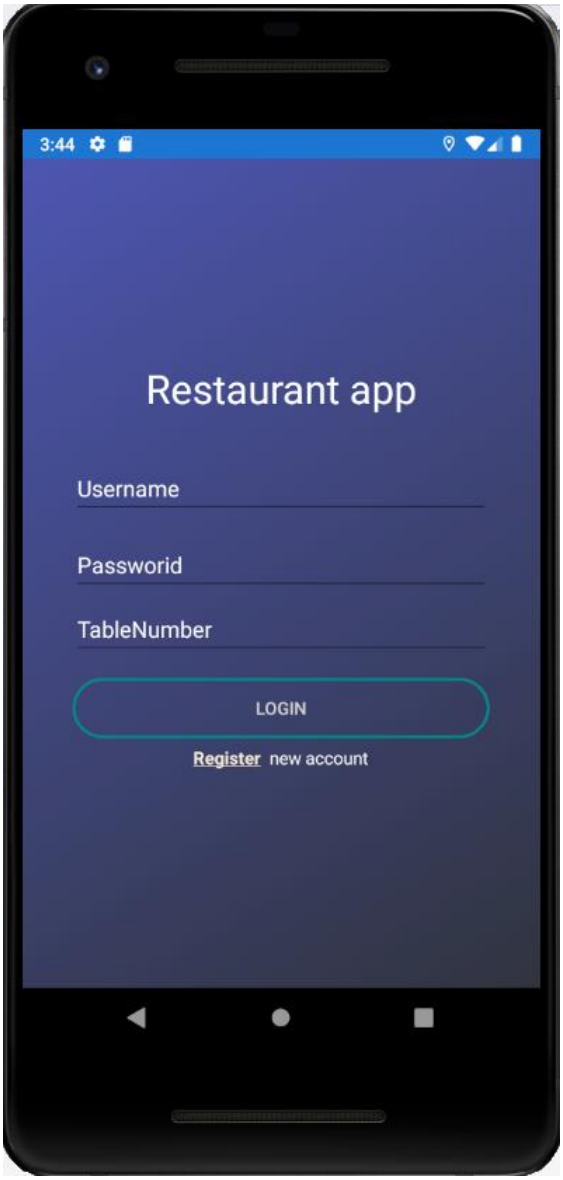
Login Page

Backend kod za Login Page

Prilkom pokretanja aplikacije u emulatoru otvara se Login page gdje registrovani korisnik unosi Username,Password i Table Number . Broj stola Gost može vidjeti na svom stolu.

Korisnik koji nije registrovan ide na opciju Register New Account.

Kada Korisnik izvrši Login otvara mu se stranica na kojoj se nalazi Meni.



Slika 1 Login Page

LoginPage.xaml.cs

```
using RestaurantApp.ViewModels;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace RestaurantApp.Views
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class LoginPage : ContentPage
    {
        public LoginPage()
        {
            InitializeComponent();
            this.BindingContext = new LoginViewModel();
        }

        private void TapGestureRecognizer_Tapped(object sender, EventArgs e)
        {
            Console.WriteLine(sender);
            Console.WriteLine(e);
        }



    }
}
```

U LoginPage.xaml.cs izvršava se inicijalizaciju nove instance TapGestureRecognizer objekta događaja koji se pokreće kada korisnik koristi Tapped.

LoginViewModel.cs

```
using RestaurantApp.Models;
using RestaurantApp.Views;
using System;
using System.Linq;
using Xamarin.Forms;

namespace RestaurantApp.ViewModels
{
    public class LoginViewModel : BaseViewModel
    {
        public Command LoginCommand { get; }
        private string username;
        private string password;

        public LoginViewModel()
        {
            LoginCommand = new Command(OnLoginClicked);
            SaveCommand = new Command(OnSave, ValidateSave);
            RegistrationCommand = new Command(GoToRegistration);
```

```
            this.PropertyChanged +=
                (_, __) => SaveCommand.ChangeCanExecute();
        }

        private async void GoToRegistration(object obj)
        {
            await Shell.Current.GoToAsync($"//{nameof(RegistrationPage)}");
        }

        public string Username
        {
            get => username;
            set => SetProperty(ref username, value);
        }

        public string Password
        {
            get => password;
            set => SetProperty(ref password, value);
        }


        private async void OnLoginClicked(object obj)
        {

            var users = DataStore.GetItemsAsync().Result;
            User ifexisits = users.ToList().Find(x => x.Username == Username &&
x.Password == Password);
            if(ifexisits != null)
            {
                Console.WriteLine("succesfully passed further");

                await Shell.Current.GoToAsync($"//{nameof(AboutPage)}");
                Username = null;
                Password = null;

            }

        }


        private bool ValidateSave()
        {
            return !String.IsNullOrWhiteSpace(username)
                && !String.IsNullOrWhiteSpace(password);
        }


        public Command SaveCommand { get; }
        public Command CancelCommand { get; }
        public Command RegistrationCommand { get; }

        private async void OnCancel()
        {
            // This will pop the current page off the navigation stack
            await Shell.Current.GoToAsync("..");
        }

        private async void OnSave()
        {
            User newItem = new User()
            {
```

```
                Id = Guid.NewGuid().ToString(),
                Username = Username,
                Password = Password
            };

            await DataStore.AddItemAsync(newItem);

            // This will pop the current page off the navigation stack
            await Shell.Current.GoToAsync("..");
        }


        }
}
```

U ovom kodu izvršava se preusmjeravanje korisnika na screen za registraciju računa.

Zatim se obavlja setovanje vrijednosti koje su unesene kroz inpute.

Izvršavanje komande prilikom klika na login button.

Provjerava se da li korisnik postoji u 'MockDataStore' nizu u slučaju da postoji sa username i pripadajućom šifrom.

Proces se prosljeđuje na AboutPage koji je u našem slučaju stranica na kojoj se nalazi Meni.

Provjera se da li su inputi prazni prilikom klika na Login button.

User.Cs

```
using System;
using System.Collections.Generic;
using System.Text;

namespace RestaurantApp.Models

{
    //Atributi modela
    public class User
    {
        public string Id { get; set; }
        public string FirstName { get; set; }
        public string LastName { get; set; }
        public string Username { get; set; }
        public string Password { get; set; }
        public string Text { get; set; }
        public string Description { get; set; }
    }
}
```

Ovdje se nalaze tipovi podataka koji su napravljeni za korisnike.Odnosno atributi modela koji su kreirani za unificiranje podataka o gostu.

To su :Id,FirstName,Username,Password,Text i Description.

Navedeni podaci su smješteni u MockDataStore.cs

Services

IDataStore.cs

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;

namespace RestaurantApp.Services
{

      // Genericki CRUD interface
    public interface IDataStore<T>
    {
        Task<bool> AddItemAsync(T item);
        Task<bool> UpdateItemAsync(T item);
        Task<bool> DeleteItemAsync(string id);
        Task<T> GetItemAsync(string id);
        Task<IEnumerable<T>> GetItemsAsync(bool forceRefresh = false);
    }
}
```

U **IDataSore.cs** se nalazi generički CRUD interferjs koji služi da implementiraju klase '**MockDataStore**'i '**MockDataProducts**' zbog jednostavije manipulacije podacima .

MockDataStore.cs

```
using RestaurantApp.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace RestaurantApp.Services
{
    public class MockDataStore : IDataStore<User>
    {
        readonly List<User> items;

        public MockDataStore()
        {
            items = new List<User>()
            {
                new User { Id = Guid.NewGuid().ToString(), FirstName = "Merima",
LastName="Durmo", Username ="merima", Password="string" },
                new User { Id = Guid.NewGuid().ToString(), FirstName = "Selma",
LastName="Agic", Username ="selma", Password="string"},
                new User { Id = Guid.NewGuid().ToString(), FirstName = "AbcMerima",
LastName="Durmo", Username ="abcmerima", Password="string" },
                new User { Id = Guid.NewGuid().ToString(), FirstName = "AbcSelma",
LastName="Agicd", Username ="abcdselma", Password="string" }

            };
        }

        public async Task<bool> AddItemAsync(User item)
        {
            items.Add(item);
```

```
            return await Task.FromResult(true);
        }

        public async Task<bool> UpdateItemAsync(User item)
        {
            var oldItem = items.Where((User arg) => arg.Id ==
item.Id).FirstOrDefault();
            items.Remove(oldItem);
            items.Add(item);

            return await Task.FromResult(true);
        }

        public async Task<bool> DeleteItemAsync(string id)
        {
            var oldItem = items.Where((User arg) => arg.Id == id).FirstOrDefault();
            items.Remove(oldItem);

            return await Task.FromResult(true);
        }

        public async Task<User> GetItemAsync(string id)
        {
            return await Task.FromResult(items.FirstOrDefault(s => s.Id == id));
        }

        public async Task<IEnumerable<User>> GetItemsAsync(bool forceRefresh = false)
        {
            return await Task.FromResult(items);
        }
    }
}
```

U **MockDataStore.cs** spremljeni su korisnički podaci FirstName,LastName ,UserName i Password.

Postoji klasa koja sadrži primjer podataka koje bi imao gost u situaciji kada bi se gost spremao u bazu podataka. Pored primjera, klasa sadrzi implementaciju svih metoda iz interfacea.Navedeni su primjeri podataka pomoću koji Korisnik može izvršiti Login bez prethodne Registracije.To je navedeno u `public MockDataStore()`

Frontend kod za Login Page

LoginPage.xaml

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             xmlns:d="http://xamarin.com/schemas/2014/forms/design"
             xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
             xmlns:ios="clr-
namespace:Xamarin.Forms.PlatformConfiguration.iOSSpecific;assembly=Xamarin.Forms.Core"
             ios:Page.UseSafeArea="true"
             mc:Ignorable="d"
             x:Class="RestaurantApp.Views.LoginPage"
             Shell.NavBarIsVisible="False">

    <Grid  RowDefinitions="*, 0*, Auto">
```

```
        <BoxView Grid.RowSpan="3">
            <BoxView.Background>
                <LinearGradientBrush>
                    <GradientStopCollection>
                        <GradientStop Color="#4d56b2" Offset="0"/>
                        <GradientStop Color="#30343F" Offset="1"/>
                    </GradientStopCollection>
                </LinearGradientBrush>
            </BoxView.Background>
        </BoxView>
        <StackLayout Grid.Row="0" Padding="40" VerticalOptions="Center">
            <Label HorizontalOptions="Center" Margin="0,0,0,30" Text="Restaurant app"
FontSize="33" TextColor="White"/>
            <Entry Text="{Binding Username, Mode=TwoWay}" TextColor="White"
FontSize="Medium" Placeholder="Username" PlaceholderColor="White"/>
            <Entry Text="{Binding Password, Mode=TwoWay}" TextColor="White"
FontSize="Medium" IsPassword="True" Margin="0,10,0,0" Placeholder="Password"
PlaceholderColor="White" />
            <Entry Text="{Binding TableNumber, Mode=TwoWay}" TextColor="White"
FontSize="Medium" Keyboard="Numeric" Placeholder="TableNumber"
PlaceholderColor="White"/>
            <Button VerticalOptions="Center" Text="Login" Command="{Binding
LoginCommand}"
                    BackgroundColor="Transparent" BorderColor="Green" BorderWidth="2"
Margin="0,10,0,0"
                    CornerRadius="50"/>
            <Label HorizontalOptions="Center">
                <Label.FormattedText>
                    <FormattedString>
                        <Span Text="Register" TextColor="Gray" FontAttributes="Bold"
TextDecorations="Underline" />
                        <Span Text="  new account" TextColor="White" />
                    </FormattedString>
                </Label.FormattedText>
                <Label.GestureRecognizers>
                    <TapGestureRecognizer Tapped="TapGestureRecognizer_Tapped"
Command="{Binding RegistrationCommand}" />
                </Label.GestureRecognizers>
            </Label>
        </StackLayout>

    </Grid>
</ContentPage>
```

U LoginPage.xaml definirane su boje pozadine,određuje se pozicija teksta koji će biti prikazan na stranici (RestaurantApp ,UserName,Password,TableNumber) i njegova boja.

Button je Login povezan pomoću svojstva Binding sa Login komandom.

Ispod forme za Login nalazi se opcija za registraciju novog korisnika “Register new account” .

Registration Page

Backend kod za Registration Page

Korisnik koji nije prije izvršio prijavu ima mogućnost Registracije, unosi svoje podatke i dobije obavijest da je registracija uspješno završena.

RegistrationPage.xaml.cs

```
using RestaurantApp.ViewModels;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Xamarin.CommunityToolkit.Extensions;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace RestaurantApp.Views
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class RegistrationPage : ContentPage
    {
        public RegistrationPage()
        {
            InitializeComponent();
            this.BindingContext = new RegisterViewModel();
        }

        public async void RegisteredCommand(System.Object sender, System.EventArgs e)
        {
            await this.DisplayToastAsync("Registration completed", 1000);
            await Shell.Current.GoToAsync($"//{nameof(LoginPage)}");
        }
    }
}
```

Klikom na Register pojavljuje se Toast -DisplayToastAsync odnosno prikaz koji sadrži kratku malu poruku “Registation completed” za korisnika i zatim se vraća na Login page.

RegisterViewModel.cs

```
using RestaurantApp.Models;
using RestaurantApp.Views;
using System;
using System.Collections.Generic;
using System.Text;
using Xamarin.Forms;

namespace RestaurantApp.ViewModels
{
    public class RegisterViewModel : BaseViewModel
    {
        string username, firstname, lastname, password;
        public RegisterViewModel()
```

```
{
    Title = "Register Page";
    OnSaveCommand = new Command(OnSave, Validate);
    BackCommand = new Command(GetBack);
    this.PropertyChanged +=
        (_, __) => OnSaveCommand.ChangeCanExecute();
}

private async void GetBack(object obj)
{
    Console.WriteLine(obj);
    await Shell.Current.GoToAsync($"//{nameof(LoginPage)}");
}

private bool Validate(object arg)
{
    return !String.IsNullOrWhiteSpace(firstname)
        && !String.IsNullOrWhiteSpace(lastname)
        && !String.IsNullOrWhiteSpace(username)
        && !String.IsNullOrWhiteSpace(password);
}

private async void OnSave(object obj)
{
    User newItem = new User()
    {
        Id = Guid.NewGuid().ToString(),
        Username = Username,
        Password = Password,
        FirstName = FirstName,
        LastName = LastName
    };

    await DataStore.AddItemAsync(newItem);

    Username = null;
    Password = null;
    FirstName = null;
    LastName = null;

    // This will pop the current page off the navigation stack

}

public string Username
{
    get => username;
    set => SetProperty(ref username, value);
}
public string FirstName
{
    get => firstname;
    set => SetProperty(ref firstname, value);
}
public string LastName
{
    get => lastname;
    set => SetProperty(ref lastname, value);
}
public string Password
{
    get => password;
```

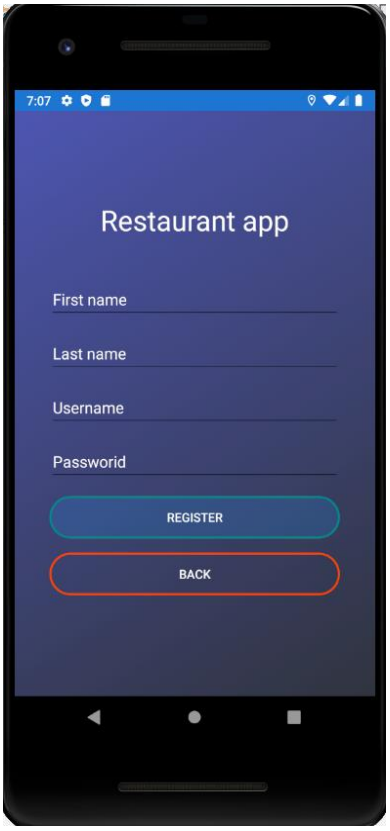
```
            set => SetProperty(ref password, value);
        }
        public Command OnSaveCommand { get; }
        public Command BackCommand { get; }
    }
}
```

U RegisterViewModel.cs nalazi se prikaz stranice na kojoj se nalazi forma za prijavu Gosta.

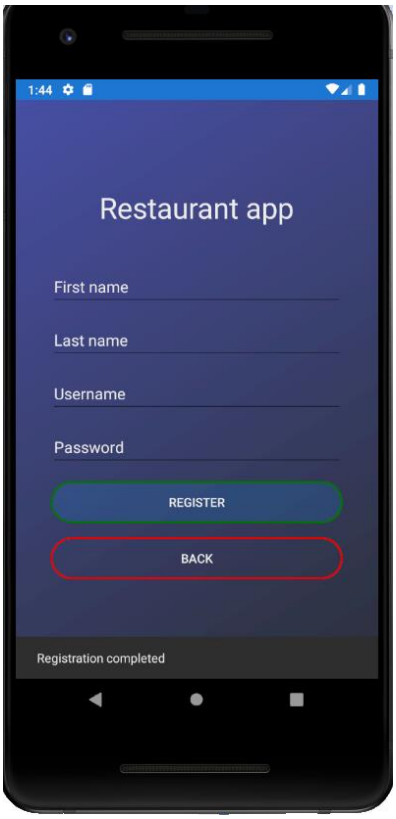
Kada se ispune potrebni podaci vraća se nazad na Login Page.

Izvršava se validacija svih inputa koje se unosi na Registration Page.

Novi gost se sprema u array usera i izvršava se resetovanje inputa,i nakon toga se setuju vrijednosti inputa.



Slika 2 Registration Page



Slika 3 Registration Completed

Frontend kod za Registration Page

RegistrationPage.xaml

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="RestaurantApp.Views.RegistrationPage"
             Shell.NavBarIsVisible="False">
    <Grid  RowDefinitions="*, 0*, Auto">

        <BoxView Grid.RowSpan="3">
            <BoxView.Background>
                <LinearGradientBrush>
                    <GradientStopCollection>
                        <GradientStop Color="#4d56b2" Offset="0"/>
                        <GradientStop Color="#30343F" Offset="1"/>
                    </GradientStopCollection>
                </LinearGradientBrush>
            </BoxView.Background>
        </BoxView>
        <StackLayout Grid.Row="0" Padding="40" VerticalOptions="Center">
            <Label HorizontalOptions="Center" Margin="0,0,0,30" Text="Restaurant app"
FontSize="33" TextColor="White"/>
            <Entry Text="{Binding FirstName, Mode=TwoWay}" TextColor="White"
FontSize="Medium" Placeholder="First name" Margin="0,10,0,0"
PlaceholderColor="White"/>
```

```
            <Entry Text="{Binding LastName, Mode=TwoWay}" TextColor="White"
FontSize="Medium" Placeholder="Last name" Margin="0,10,0,0"  PlaceholderColor="White"
/>
            <Entry Text="{Binding Username, Mode=TwoWay}" TextColor="White"
FontSize="Medium" Placeholder="Username" Margin="0,10,0,0"  PlaceholderColor="White"/>
            <Entry Text="{Binding Password, Mode=TwoWay}" TextColor="White"
FontSize="Medium" IsPassword="True" Margin="0,10,0,0" Placeholder="Password"
PlaceholderColor="White" />
            <Button VerticalOptions="Center" Text="Register" Command="{Binding
OnSaveCommand}" Clicked="RegisteredCommand"
                    BackgroundColor="Transparent" BorderColor="Green" BorderWidth="2"
Margin="0,10,0,0"
                    CornerRadius="50"/>
            <Button VerticalOptions="Center" Text="Back" Command="{Binding
BackCommand}"
                    BackgroundColor="Transparent" BorderColor="Red" BorderWidth="2"
Margin="0,10,0,0"
                    CornerRadius="50"/>


        </StackLayout>

    </Grid>
</ContentPage>
```

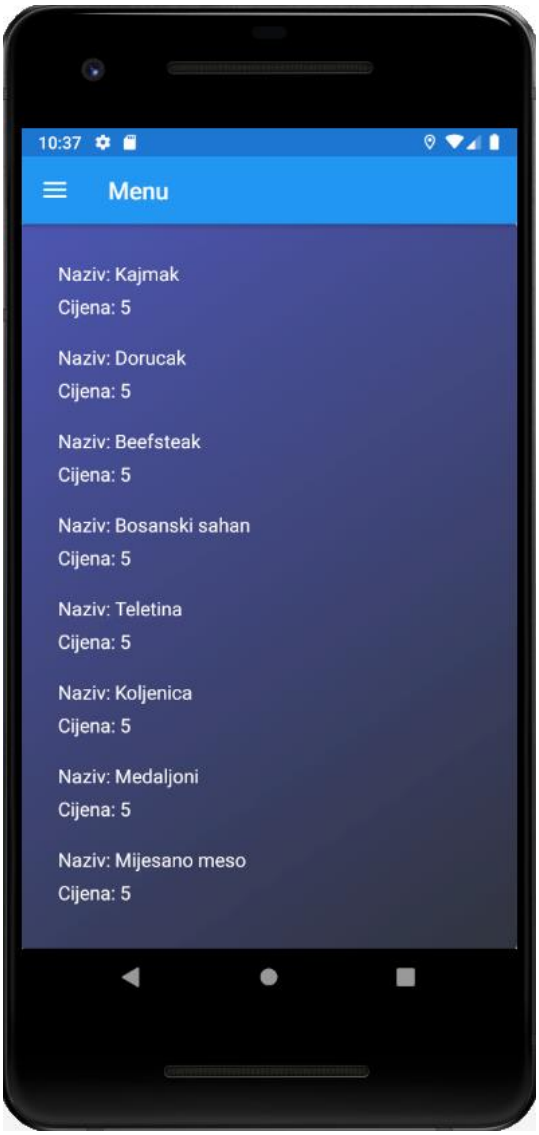
U RegistrationPage.xaml nalazi se forma za registraciju novog korisnika.

Boje za pozadinu iste kao i na ostalim .xaml stranicama. Buttoni koji imaju su Register i Back.

Podaci koji se upisuju su First Name,Last Name,Username i Password .

Menu Page

Nakon registracije i login-a korisniku se otvara stranica na kojoj se nalazi menu. Korisnik može vidjeti listu svih jela koja su u ponudi i njihovu cijenu.



Slika 4 Menu Page

Backend kod za Menu Page

AboutPage.xaml.cs

```
using RestaurantApp.Models;
using RestaurantApp.ViewModels;
using RestaurantApp.Views;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Linq;
```

```
using System.Text;
using System.Threading.Tasks;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace RestaurantApp.Views
{
    public partial class ItemsPage : ContentPage
    {
        ItemsViewModel _viewModel;

        public ItemsPage()
        {
            InitializeComponent();

            BindingContext = _viewModel = new ItemsViewModel();
        }

        protected override void OnAppearing()
        {
            base.OnAppearing();
            _viewModel.OnAppearing();
        }
    }
}
```

AboutPage.xaml.cs: Ovaj kod se automatski generiše prilikom kreiranja Flyout template-a

AboutViewsModel.cs

```
using RestaurantApp.Models;
using RestaurantApp.Views;
using System;
using System.Collections.ObjectModel;
using System.Diagnostics;
using System.Threading.Tasks;
using System.Windows.Input;
using Xamarin.Essentials;
using Xamarin.Forms;

namespace RestaurantApp.ViewModels
{
    public class AboutViewModel : BaseViewModel
    {
        private Item _selectedItem;

        public ObservableCollection<Item> Items { get; }
        public Command LoadItemsCommand { get; }
        public Command AddItemCommand { get; }
        public Command<Item> ItemTapped { get; }
        public AboutViewModel()
        {
            Console.WriteLine("loadaed");
            Title = "Menu";
            Items = new ObservableCollection<Item>();
            LoadItemsCommand = new Command(async () => await
ExecuteLoadItemsCommand());

            ItemTapped = new Command<Item>(OnItemSelected);
```

```
            AddItemCommand = new Command(OnAddItem);
        }
        async Task ExecuteLoadItemsCommand()
        {
            IsBusy = true;

            try
            {
                Items.Clear();
                var items = await ItemDataStore.GetItemsAsync();
                foreach (var item in items)
                {
                    Items.Add(item);
                }
            }
            catch (Exception ex)
            {
                Debug.WriteLine(ex);
            }
            finally
            {
                IsBusy = false;
            }
        }
        public void OnAppearing()
        {
            IsBusy = true;
            SelectedItem = null;
        }

        public Item SelectedItem
        {
            get => _selectedItem;
            set
            {
                SetProperty(ref _selectedItem, value);
                OnItemSelected(value);
            }
        }


        // Preusmjeravanje na screen za dodavanje novih jela
        private async void OnAddItem(object obj)
        {
            await Shell.Current.GoToAsync(nameof(NewItemPage));
        }

        async void OnItemSelected(Item item)
        {
            if (item == null)
                return;
            await
Shell.Current.GoToAsync($"{nameof(ItemDetailPage)}?{nameof(ItemDetailViewModel.ItemId)
}={item.Id}");
        }
    }
}
```

Kod za AboutViewsModel.cs sadrži: funkcije za učitavanje svih jela iz MockDataStore.cs, postavljanje parametara prilikom učitavanja screen-a, dodjeljivanje vrijednosti koje se nalaze u modelu jela koje je

korisnik kliknuo te funkciju za preusmjeravanje korisnika na screen sa detaljima jela koje je gost izabrao. Kod također sadrži funkciju koja nije implementirana a služi za preusmjeravanje na screen za dodavanje novih jela, s obzirom da je aplikacija napravljena za gosta ali mogla bi se iskoristiti za administratora.

Item.cs

```
using System;

namespace RestaurantApp.Models
{
    public class Item
    {
        public string Id { get; set; }
        public string Text { get; set; }
        public string Description { get; set; }
        public string Title { get; set; }
        public string Picture { get; set; }
        public string Price { get; set; }
    }
}
```

Model **Item.cs** je kreiran za unificiranje podataka o jelu i dodani su mu sljedeći atributi: Id, Text, Description, Title, Picture, Price.

IDataStore.cs

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;

namespace RestaurantApp.Services
{
    public interface IDataStore<T>
    {
        Task<bool> AddItemAsync(T item);
        Task<bool> UpdateItemAsync(T item);
        Task<bool> DeleteItemAsync(string id);
        Task<T> GetItemAsync(string id);
        Task<IEnumerable<T>> GetItemsAsync(bool forceRefresh = false);
    }
}
```

IDataStore.cs je kao što je već navedeno generički CRUD interface koji implementira klasa MockDataStoreProduct zbog jednostavnijeg manipulisanja podacima. Sadrži metode za dodavanje, update-ovanje, brisanje i dohvaćanje podataka.

MockDataStoreProducts.cs

```
using RestaurantApp.Models;
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace RestaurantApp.Services
{
    public class MockDataStoreProducts : IDataStore<Item>
    {
        readonly List<Item> items;
        public MockDataStoreProducts()
        {
            items = new List<Item>()
            {
                new Item {Id = Guid.NewGuid().ToString(), Description = "Lepina sa
kajmakom, 100g", Title = "Kajmak", Picture = "kajmak.png", Price = "5"},
                new Item {Id = Guid.NewGuid().ToString(), Description = "Jaja, šunka,
kajmak, hrenovke, sudžuk", Title = "Dorucak", Picture = "dorucak.png", Price = "5"},
                new Item {Id = Guid.NewGuid().ToString(), Description = "Teleći
steak", Title = "Beefsteak", Picture = "beef.png", Price = "5"},
                new Item {Id = Guid.NewGuid().ToString(), Description = "Čorba sa
povrćem i teletinom", Title = "Bosanski sahan", Picture = "bosanski.png", Price =
"5"},
                new Item {Id = Guid.NewGuid().ToString(), Description = "Teletina u
lepini", Title = "Teletina ", Picture = "cudesa.png", Price = "5"},
                new Item {Id = Guid.NewGuid().ToString(), Description = "Lešo teleća
koljenica", Title = "Koljenica", Picture = "koljenica.png", Price = "5"},
                new Item {Id = Guid.NewGuid().ToString(), Description = "Teleći
medaljoni", Title = "Medaljoni", Picture = "medaljoni.png", Price = "5"},
                new Item {Id = Guid.NewGuid().ToString(), Description = "Mijesano
meso", Title = "Mijesano meso", Picture = "mijesano.png", Price = "5"},
                new Item {Id = Guid.NewGuid().ToString(), Description = "Suho meso,
sudžuk, sir, kajmak", Title = "Hladna plata", Picture = "plata.png", Price = "5"},
                new Item {Id = Guid.NewGuid().ToString(), Description = "Teleća
pljeskavica", Title = "Pljeskavica", Picture = "pljeska.png", Price = "5"},
                new Item {Id = Guid.NewGuid().ToString(), Description = "Sarma", Title
= "Sarma", Picture = "sarma.png", Price = "5"},
                new Item {Id = Guid.NewGuid().ToString(), Description = "Teletina sa
rostilja s prilozima", Title = "Teletina sa rostilja", Picture = "slatkis.png", Price
= "5"},
                new Item {Id = Guid.NewGuid().ToString(), Description = "Tomahawk",
Title = "Tomahawk", Picture = "tomahawk.png", Price = "100"},
                new Item {Id = Guid.NewGuid().ToString(), Description = "Tuna steak",
Title = "Tuna steak", Picture = "tuna.png", Price = "5"},
            };

        }

        public async Task<bool> AddItemAsync(Item item)
        {
            items.Add(item);

            return await Task.FromResult(true);
        }

        public async Task<bool> DeleteItemAsync(string id)
        {
            var oldItem = items.Where((Item arg) => arg.Id == id).FirstOrDefault();
            items.Remove(oldItem);

            return await Task.FromResult(true);
```

```
        }

        public async Task<Item> GetItemAsync(string id)
        {
            return await Task.FromResult(items.FirstOrDefault(s => s.Id == id));
        }

        public async Task<IEnumerable<Item>> GetItemsAsync(bool forceRefresh = false)
        {
            return await Task.FromResult(items);
        }

        public async Task<bool> UpdateItemAsync(Item item)
        {
            var oldItem = items.Where((Item arg) => arg.Id ==
item.Id).FirstOrDefault();
            items.Remove(oldItem);
            items.Add(item);

            return await Task.FromResult(true);
        }
    }
}
```

Kod za MockDataStoreProducts.cs: Kreirana je klasa koja sadrži primjer podataka koje bi imalo svako jelo kad bi se spremalo u bazu podataka. Klasa također sadrzi implementaciju svih metoda iz interface-a IDataStore.cs.

BaseViewModel.cs

```
using RestaurantApp.Models;
using RestaurantApp.Services;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Runtime.CompilerServices;
using Xamarin.Forms;

namespace RestaurantApp.ViewModels
{
    public class BaseViewModel : INotifyPropertyChanged
    {
        public IDataStore<User> DataStore =>
DependencyService.Get<IDataStore<User>>();
        public IDataStore<Item> ItemDataStore =>
DependencyService.Get<IDataStore<Item>>();

        bool isBusy = false;
        public bool IsBusy
        {
            get { return isBusy; }
            set { SetProperty(ref isBusy, value); }
        }

        string title = string.Empty;
        public string Title
        {
            get { return title; }
            set { SetProperty(ref title, value); }
        }
```

```
        protected bool SetProperty<T>(ref T backingStore, T value,
            [CallerMemberName] string propertyName = "",
            Action onChanged = null)
        {
            if (EqualityComparer<T>.Default.Equals(backingStore, value))
                return false;

            backingStore = value;
            onChanged?.Invoke();
            OnPropertyChanged(propertyName);
            return true;
        }

        #region INotifyPropertyChanged
        public event PropertyChangedEventHandler PropertyChanged;
        protected void OnPropertyChanged([CallerMemberName] string propertyName = "")
        {
            var changed = PropertyChanged;
            if (changed == null)
                return;

            changed.Invoke(this, new PropertyChangedEventArgs(propertyName));
        }
        #endregion
    }
}
```

BaseViewModel.cs je generička klasa koja predstavlja bazni model screen-a i kroz nju se pristupa svim podacima. `DependencyService.Get<IDataStore<Item>>()` je komanda kojom se dohvacaju podaci iz Item.cs

Frontend kod za Menu Page

AboutPage.xaml

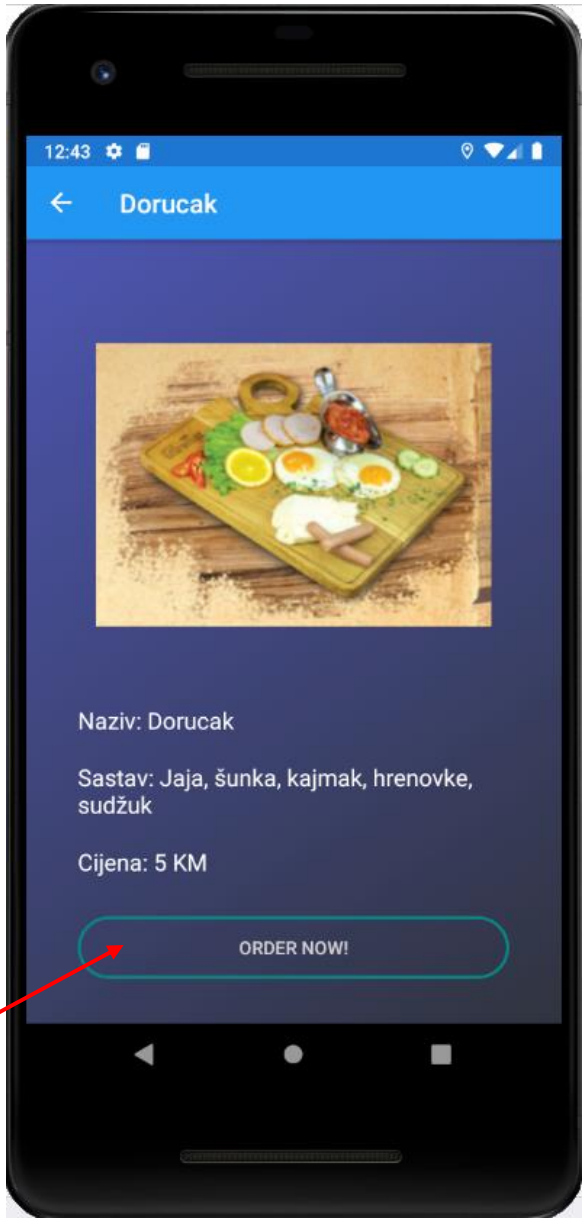
```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="RestaurantApp.Views.AboutPage"
             xmlns:local="clr-namespace:RestaurantApp.ViewModels"
             xmlns:model="clr-namespace:RestaurantApp.Models"
             Title="{Binding Title}">


    <RefreshView x:DataType="local:AboutViewModel" Command="{Binding
LoadItemsCommand}" IsRefreshing="{Binding IsBusy, Mode=TwoWay}">
        <Frame >
            <Frame.Background>
                <LinearGradientBrush>
                    <GradientStopCollection>
                        <GradientStop Color="#161616" Offset="0"/>
                        <GradientStop Color="#30343F" Offset="1"/>
                    </GradientStopCollection>
                </LinearGradientBrush>
            </Frame.Background>
            <StackLayout AbsoluteLayout.LayoutFlags="All"
AbsoluteLayout.LayoutBounds="0,0,1,1" >
                <CollectionView x:Name="ItemsListView"
                ItemsSource="{Binding Items}"
                SelectionMode="None">
                    <CollectionView.ItemTemplate>
```

```
                        <DataTemplate >
                            <StackLayout Padding="10" x:DataType="model:Item" >

                                <Label  Text="Text: " TextColor="White"
Style="{DynamicResource ListItemTextStyle}"
                                FontSize="16">
                                        <Label.FormattedText >
                                            <FormattedString >
                                                <Span Text="Naziv: " FontSize="Small"/>
                                                <Span Text="{Binding Title}"
FontSize="Small"/>
                                            </FormattedString>
                                        </Label.FormattedText>
                                    </Label >
                                    <Label Style="{DynamicResource ListItemTextStyle}"
                                FontSize="16"  Text="Text: " TextColor="White"  >
                                        <Label.FormattedText >
                                            <FormattedString >
                                                <Span Text="Cijena: " FontSize="Small"/>
                                                <Span Text="{Binding Price}"
FontSize="Small"/>
                                            </FormattedString>
                                        </Label.FormattedText>
                                    </Label>

                                    <StackLayout.GestureRecognizers>
                                        <TapGestureRecognizer
                                    NumberOfTapsRequired="1"
                                    Command="{Binding Source={RelativeSource
AncestorType={x:Type local:AboutViewModel}}, Path=ItemTapped}"
                                    CommandParameter="{Binding .}">
                                        </TapGestureRecognizer>
                                    </StackLayout.GestureRecognizers>
                                </StackLayout>
                            </DataTemplate>
                        </CollectionView.ItemTemplate>
                    </CollectionView>
                </StackLayout>
            </Frame>
        </RefreshView>
</ContentPage>
```

AboutPage.xaml: frontend kod određuje kako će aplikacija izgledati. Određuje boju pozadine i teksta, veličinu fonta i sl.

Item Detail Page

Korisnik moze kliknuti na svako jelo koje izabere da dobije informacije o njemu. Nakon što se klikne na određeno jelo otvori se nova stranica sa slikom, cijenom i sastavom jela. Gost moze naručiti jelo klikom na button Order now! nakon čega dobije obavijest da je narudžba primljena.



Slika 5 Menu Detail

Slika 6 Order now!

Backend kod za Menu Detail Page

ItemDetailPage.xaml.cs

```
using RestaurantApp.ViewModels;
using System.ComponentModel;
using Xamarin.Forms;
using Xamarin.CommunityToolkit.Extensions;

namespace RestaurantApp.Views
{
    public partial class ItemDetailPage : ContentPage
    {
        public ItemDetailPage()
        {
            InitializeComponent();
            BindingContext = new ItemDetailViewModel();
        }

        private async void OrderCommand(System.Object sender, System.EventArgs e)
        {
            await this.DisplayToastAsync("Your order has been recieved!", 1000);
        }
    }
}
```

Kod za ItemDetailPage.xaml.cs: Klikom na button 'Order now' poziva se OrderCommand koja inicijalizira toast obavijest o primljenoj narudžbi.

ItemDetailViewModel.cs

```
using RestaurantApp.Models;
using System;
using System.Diagnostics;
using System.Threading.Tasks;
using Xamarin.Forms;
using Xamarin.CommunityToolkit.Extensions;

namespace RestaurantApp.ViewModels
{
    [QueryProperty(nameof(ItemId), nameof(ItemId))]
    public class ItemDetailViewModel : BaseViewModel
    {
        private string itemId;
        private string text;
        private string description;
        private string picture;
        private string price;
        public Command OrderCommands;

        public ItemDetailViewModel()
        {

        }

        public string Id { get; set; }
```

```
        public string Text
        {
            get => text;
            set => SetProperty(ref text, value);
        }
        public string Price
        {
            get => price;
            set => SetProperty(ref price, value);
        }
        public string Picture
        {
            get => picture;
            set => SetProperty(ref picture, value);
        }


        public string Description
        {
            get => description;
            set => SetProperty(ref description, value);
        }

        public string ItemId
        {
            get
            {
                return itemId;
            }
            set
            {
                itemId = value;
                LoadItemId(value);
            }
        }

        public async void LoadItemId(string itemId)
        {
            try
            {
                var item = await ItemDataStore.GetItemAsync(itemId);
                Id = item.Id;
                Text = item.Title;
                Description = item.Description;
                Price = item.Price;
                Picture = item.Picture;
            }
            catch (Exception)
            {
                Debug.WriteLine("Failed to Load Item");
            }
        }
    }
}
```

Kod za ItemDetailViewModel.cs sadrži naredbe za setovanje varijabli iz modela i za učitavanje jela na koje korisnik klikne. `public async void LoadItemId(string itemId)` -učitavanje jela koje je korisnik izabrao. `var item = await ItemDataStore.GetItemAsync(itemId);` - traži ga da li postoji u IdataStore.c

Frontend kod za Menu Detail Page

ItemDetailPage.xaml

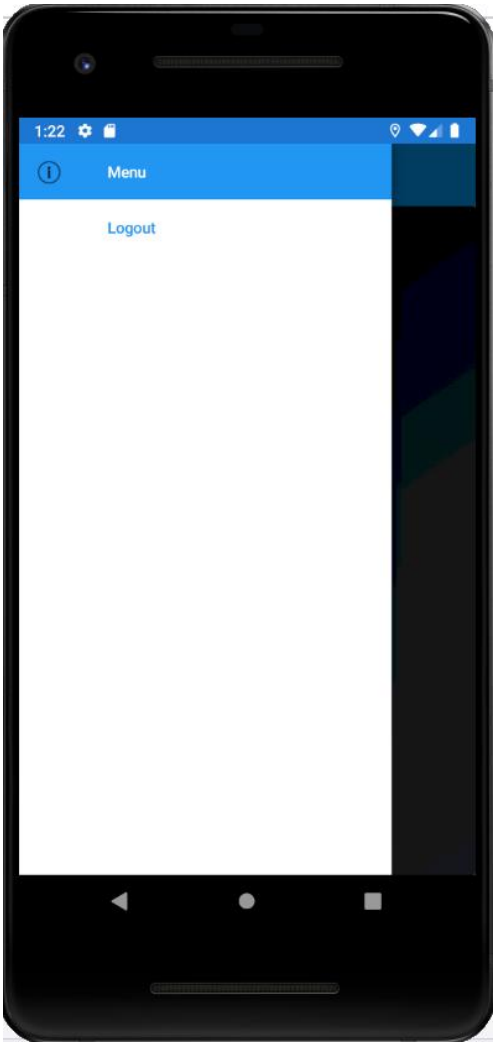
```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="RestaurantApp.Views.ItemDetailPage"
             Title="{Binding Text}">

    <Grid RowDefinitions="*, 0*, Auto">
        <BoxView Grid.RowSpan="3">
            <BoxView.Background>
                <LinearGradientBrush>
                    <GradientStopCollection>
                        <GradientStop Color="#4d56b2" Offset="0"/>
                        <GradientStop Color="#30343F" Offset="1"/>
                    </GradientStopCollection>
                </LinearGradientBrush>
            </BoxView.Background>
        </BoxView>
        <StackLayout Grid.Row="0" Spacing="20" Padding="40" VerticalOptions="Center">
            <ContentView Padding="0,40,0,40" VerticalOptions="FillAndExpand">
                <Image Source="{Binding Picture}" VerticalOptions="Center"  />
            </ContentView>
            <Label  Text="Text: " TextColor="White" FontSize="Medium" >
                <Label.FormattedText >
                    <FormattedString >
                        <Span Text="Naziv: " FontSize="Small"/>
                        <Span Text="{Binding Text}" FontSize="Small"/>
                    </FormattedString>
                </Label.FormattedText>
            </Label>

            <Label TextColor="White"  FontSize="Medium" >
                <Label.FormattedText>
                    <FormattedString>
                        <Span Text="Sastav: " FontSize="Small"/>
                        <Span Text="{Binding Description}" FontSize="Small"/>
                    </FormattedString>
                </Label.FormattedText>
            </Label>
            <Label TextColor="White" Text="Text: " FontSize="Medium" >
                <Label.FormattedText>
                    <FormattedString>
                        <Span Text="Cijena: " FontSize="Small"/>
                        <Span Text="{Binding Price}" FontSize="Small"/>
                        <Span Text=" KM" FontSize="Small"/>
                    </FormattedString>
                </Label.FormattedText>
            </Label>
            <Button VerticalOptions="Center" Text="Order now!" Clicked="OrderCommand"
                    BackgroundColor="Transparent" BorderColor="DarkCyan"
BorderWidth="2" Margin="0,10,0,0"
                    CornerRadius="50"/>

        </StackLayout>
    </Grid>
</ContentPage>
```

U ItemDetailPage.xaml definirane su boje pozadine,određuje se pozicija teksta koji će biti prikazan na stranici, velicina teksta i njegova boja. Button Order now povezan pomoću svojstva Binding sa Order komandom.

ItemDetailPage.xaml

Side Menu

Side Menu sadrži Logout opciju. Klikom na nju korisnik se odjavljuje sa svog account-a.



Slika 7 Side Menu

Backend kod za side menu

App.xaml.cs

```
using RestaurantApp.Services;
using RestaurantApp.Views;
using System;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;
```

```
namespace RestaurantApp
{
    public partial class App : Application
    {

        public App()
        {
            InitializeComponent();

            DependencyService.Register<MockDataStore>();
            DependencyService.Register<MockDataStoreProducts>();
            MainPage = new AppShell();
        }

        protected override void OnStart()
        {
        }

        protected override void OnSleep()
        {
        }

        protected override void OnResume()
        {
        }
    }
}
```

App.xaml.cs: Backend kod za side menu se automatski generiše prilikom kreiranja Flyout template-a.

`MainPage = new AppShell();` - AppShell je dodata kao root stranica.

AppShell.xaml.cs

```
using System.Collections.Generic;
using Xamarin.Forms;

namespace RestaurantApp
{
    public partial class AppShell : Xamarin.Forms.Shell
    {
        public AppShell()
        {
            InitializeComponent();
            Routing.RegisterRoute(nameof(ItemDetailPage), typeof(ItemDetailPage));
            Routing.RegisterRoute(nameof(NewItemPage), typeof(NewItemPage));
            Routing.RegisterRoute(nameof(RegistrationPage), typeof(RegistrationPage));
        }

        private async void OnMenuItemClicked(object sender, EventArgs e)
        {
            await Shell.Current.GoToAsync("//LoginPage");
        }
    }
}
```

Kod za AppShell.xaml.cs:

```
        private async void OnMenuItemClicked(object sender, EventArgs e)
        {
            await Shell.Current.GoToAsync("//LoginPage");
        }
```

Kada se aplikacija otvori prikazuje se Login Page. Deklarisane su i ostale stranice koje postoje.

Frontend kod za side menu

AppShel.xaml

```
<?xml version="1.0" encoding="UTF-8"?>
<Shell xmlns="http://xamarin.com/schemas/2014/forms"
       xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
       xmlns:local="clr-namespace:RestaurantApp.Views"
       Title="RestaurantApp"
       x:Class="RestaurantApp.AppShell">

    <Shell.Resources>
        <ResourceDictionary>
            <Style x:Key="BaseStyle" TargetType="Element">
                <Setter Property="Shell.BackgroundColor" Value="{StaticResource
Primary}" />
                <Setter Property="Shell.ForegroundColor" Value="#1f0a69" />
                <Setter Property="Shell.TitleColor" Value="White" />
                <Setter Property="Shell.DisabledColor" Value="#1f0a69" />
                <Setter Property="Shell.UnselectedColor" Value="#1f0a69" />
                <Setter Property="Shell.TabBarBackgroundColor" Value="{StaticResource
Primary}" />
                <Setter Property="Shell.TabBarForegroundColor" Value="#1f0a69"/>
                <Setter Property="Shell.TabBarUnselectedColor" Value="#1f0a69"/>
                <Setter Property="Shell.TabBarTitleColor" Value="White"/>
            </Style>
            <Style TargetType="TabBar" BasedOn="{StaticResource BaseStyle}" />
            <Style TargetType="FlyoutItem" BasedOn="{StaticResource BaseStyle}" />

            <Style Class="FlyoutItemLabelStyle" TargetType="Label">
                <Setter Property="TextColor" Value="White"></Setter>
            </Style>
            <Style Class="FlyoutItemLayoutStyle" TargetType="Layout"
ApplyToDerivedTypes="True">
                <Setter Property="VisualStateManager.VisualStateGroups">
                    <VisualStateGroupList>
                        <VisualStateGroup x:Name="CommonStates">
                            <VisualState x:Name="Normal">
                                <VisualState.Setters>
                                    <Setter Property="BackgroundColor"
Value="{x:OnPlatform UWP=Transparent, iOS=White}" />
                                    <Setter TargetName="FlyoutItemLabel"
Property="Label.TextColor" Value="{StaticResource Primary}" />
                                </VisualState.Setters>
                            </VisualState>
                            <VisualState x:Name="Selected">
                                <VisualState.Setters>
                                    <Setter Property="BackgroundColor"
Value="{StaticResource Primary}" />
                                </VisualState.Setters>
                            </VisualState>
                        </VisualStateGroup>
                    </VisualStateGroupList>
                </Setter>
            </Style>

            -->
```

```
            <Style Class="MenuItemLayoutStyle" TargetType="Layout"
ApplyToDerivedTypes="True">
                <Setter Property="VisualStateManager.VisualStateGroups">
                    <VisualStateGroupList>
                        <VisualStateGroup x:Name="CommonStates">
                            <VisualState x:Name="Normal">
                                <VisualState.Setters>
                                    <Setter TargetName="FlyoutItemLabel"
Property="Label.TextColor" Value="{StaticResource Primary}" />
                                </VisualState.Setters>
                            </VisualState>
                        </VisualStateGroup>
                    </VisualStateGroupList>
                </Setter>
            </Style>
        </ResourceDictionary>
    </Shell.Resources>

    <TabBar>
        <ShellContent Route="LoginPage" ContentTemplate="{DataTemplate
local:LoginPage}" />
    </TabBar>

    <TabBar >
        <ShellContent Route="RegistrationPage" ContentTemplate="{DataTemplate
local:RegistrationPage}" />
    </TabBar>
    <FlyoutItem Title="Menu" Icon="icon_about.png">
        <ShellContent Route="AboutPage" ContentTemplate="{DataTemplate
local:AboutPage}" />
    </FlyoutItem>

    <MenuItem Text="Logout" StyleClass="MenuItemLayoutStyle"
Clicked="OnMenuItemClicked">
    </MenuItem>



</Shell>
```

AppShel.xaml kod odredjuje izgled i nacin prikaza side menija, velicinu teksta, boju pozadine, boju teksta i sl. Takodjer su deklarisane rute do stranica.

Zaključak

U ovom radu je prikazan način rada Mobilne aplikacije za naručivanje u restoranu.

Mobilna aplikacija služi da se olakša rad u restoranu.Kako korisnicima usluge -gostima tako i radnicima-konobarima. Omogućava se efikasniji rad ,bez zadržavanja gostiju i bez bespotrebnog čekanja za dostavljanje Menija , Računa i uzimanje narudžbe .

Ova aplikacija bi se mogla dodatno nadograđivati gdje bi se povezalo sa bazom podataka.

Za realizaciju ove aplikacije primjenjeno je što više znanja naučenog na vježbama sa predmeta Razvoj aplikacija mobilnih uređaja.Sprovedena su mnoga dodatna istraživanja da bi aplikacija na što bolji način bila predstavljena.

Tablica slika