

Fakultet: Politehnički fakultet u Zenici  
Predmet: Razvoj mobilnih aplikacija  
Profesor: Asmir Butković  
Studenti: Tarik Mujkić, 133 i Tarik Valjevac, 100

# DOKUMENTACIJA ZAVRŠNOG PROJEKTA *“Tabbled”*

Zenica, 2022. Godine

# Sadržaj

<b>Opis projekta</b>	<b><a href="#">3</a></b>
<b>Dizajn projekta</b>	<b>4</b>
3.1. Kreiranje strukture	5
3.2 Kreiranje Modela	6
3.3 Kreiranje ViewModela	8
3.4 Views	11
<b>Zaključak</b>	<b>22</b>

## 1. Opis projekta

Projekat pod nazivom “Tabbled” radili su studenti Tarik Mujkić i Tarik Valjevac u kojem su primijenjene sve tehnike razvoja mobilnih aplikacija koje su obrađene na predmetu Razvoj mobilnih aplikacija. Tehnologije koje su korištene su Xamarin.Forms i Firebase Realtime database.

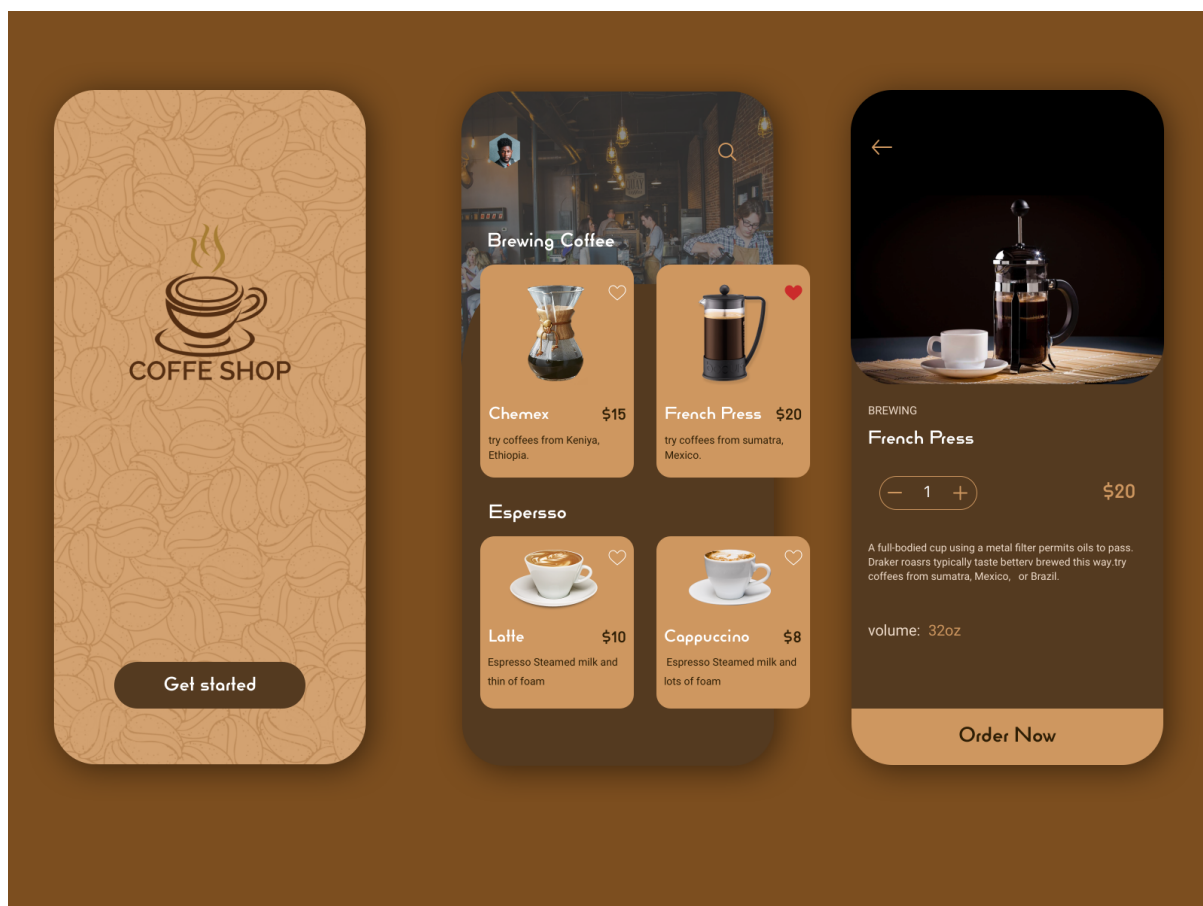
Projekat je nastao iz veoma primjetnog problema rezervacije stolova u nekom od ugostiteljskih objekata koji se petkom i subotom veoma brzo popune. S obzirom na samu psihologiju ljudi, ovom aplikacijom se omogućuje svima da sa par klikova rezervišu stol za sebe i svoje prijatelje. Aplikacija se svodi na klasični login i registration flow, a nakon toga u Main screenu nam se prikazuju svi kafići koji koriste ovu aplikaciju te klikom na njih možemo rezervirati stol.

Bitno je napomenuti da je aplikacija prototip te da nije u potpunosti finalizirana ali sadrži osnovne koncepte i tehnike razvijanja jedne mobilne aplikacije.

## 2. Dizajn projekta

U ovoj aplikaciji korišteni su svi osnovni XAML elementi koji se koriste u Xamarin.Forms platformi za razvoj mobilnih aplikacija.

Sve slike, stickeri i ikonice su potpuno free i korištene su u svrhu uljepšavanja aplikacije a ne u svrhu sponzorisanja istih. Stickeri i ikone su korištene sa web stranice Flaticon koja pruža uslugu preuzimanja besplatnih svg fileova.



Okvirni dizajn koji je bio korišten je prikazan na slici iznad, a neke od elemenata ćemo detaljno objasniti u nastavku dokumentacije.

### 3. Aplikacija

Aplikacija sadrži sljedeće elemente koji čine osnovu funkcionalnosti i dizajna:

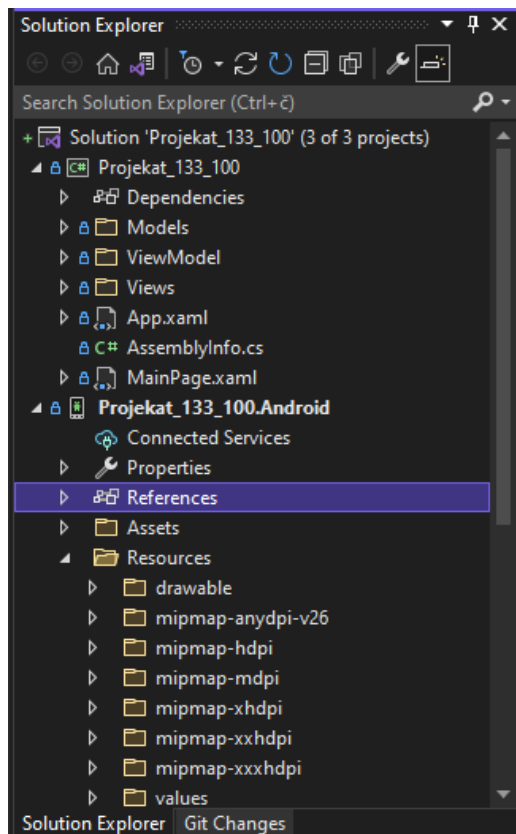
- Entry Screen
- Registration & Login
- Pregled svih kafića
- Detalji kafića
- Rezervacija stola

Pri samom kreiranju aplikacije koristili smo Blank project app, zbog lakše manipulacije elementima i razumijevanja flow-a aplikacije, iako je zastupljen Tabbed View unutar MainScreena.

Također, prije samog developmenta kreirana je Real Time baza na cloud platformi Firebase koja je napravljena od strane Google. Baza je kreirana na privatnom kontejneru.

#### 3.1. Kreiranje strukture

Najprije je kreirana struktura projekta koja je priložena na slici broj 3.



*Slika 3. Struktura projekta*

Projekat se sastoji od 4 osnovna paketa a to su: Models, ViewModel, Views i Main komponente. Ovakva struktura predstavlja MVVM pattern koji je jako popularan i korišten pattern u Xamarin.Forms-u.

### 3.2 Kreiranje Modela

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

namespace Projekat_133_100.Models
{
    public class User
    {
        public int Id { get; set; }
        public string Username { get; set; }
        public string Email { get; set; }
        public string Password { get; set; }
        public string ConfirmPassword { get; set; }
    }
}
```

```

        public string ReservedTable { get; set; }

        public User() { }

        public User(string Username, string Email, string
Password, string ConfirmPassword)
        {
            this.Username = Username;
            this.Email = Email;
            this.Password = Password;
            this.ConfirmPassword = ConfirmPassword;
        }
    }
}

```

*Code snippet 1*

U code snippet-u iznad je prikazano kreiranje Modela User. Zajedno sa deklarisanjem varijabli, napravljena su i dva konstruktura koji se koriste pri pozivanju instance klase User i pri spremanju usera u bazu.

```

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

namespace Projekat_133_100.Models
{
    public class Table
    {
        public int Id { get; set; }
        public Boolean Reserved { get; set; }
        public string Caffee { get; set; }

        public Table()
        {
            Reserved = false;
        }
    }
}

```

```
}
```

*Code snippet 2*

U code snippet-u 2 je prikazan Model Table koji predstavlja stol kafića koji ima defaultni konstruktor sa postavljenom rezervacijom na false.

Ova varijabla će da poprimi vrijednost true nakon što user rezerviše ovaj stol.

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

namespace Projekat_133_100.Models
{
    public class Caffee
    {
        public int Id { get; set; }
        public string Name { get; set; }
        public int Capacity { get; set; }
        public string Description { get; set; }
        public string Image { get; set; }

        public Caffee()
        {
        }
    }
}
```

*Code snippet 3*

U code snippet-u 3 se prikazuje Model Caffee. Ovo je osnovni model svakog kafića koji se nalazi u aplikaciji i koji ima jedan defaultni konstruktor za njegovo pozivanje u View paketima.

### 3.3 Kreiranje ViewModela



ViewModel u ovoj aplikaciji je korišten da bi se uspješno spojila baza podataka na samu aplikaciju. ViewModel koji se koristio se naziva FirebaseHelper, te kao što i sam naziv govori, omogućava nam korištenje Firebase baze podataka. Prikaz osnovnih modela i njihovo objašnjenje se nalazi u nastavku.

```
public static FirebaseClient firebaseClient = new
FirebaseClient("https://xamarinDATABASEproject-6453f-default-rtdb.eu
rope-west1.firebaseioapp/");
```

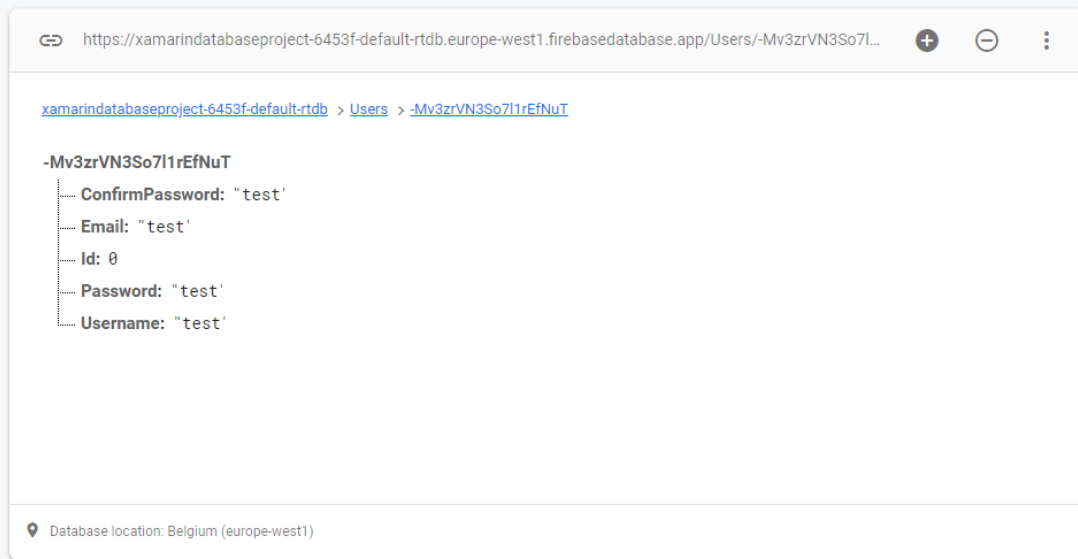
Varijabla firebaseClient nam omogućava povezivanje sa bazom.

Sada kada smo uspješno povezali aplikaciju sa bazom, možemo manipulirati sa podacima. Funkcionalnost koja je implementirana jeste spremanje usera u bazu te njegova autentifikacija kroz bazu.

```
public static async Task<bool> AddUser(string username, string
email, string password, string confirm_password)
{
    try
    {
        await firebaseClient
            .Child("Users")
            .PostAsync(new User() { Username = username, Email
= email, Password = password, ConfirmPassword = confirm_password
});
        return true;
    }
    catch (Exception e)
    {
        Debug.WriteLine($"Error:{e}");
        return false;
    }
}
```

```
}
```

Metoda `AddUser` nam sprema usera u bazu kao klasični POST request u RESTful API sistem. Konstruktor koji je napravljen u Modelu se koristi u ovoj metodi da bi inicijalizirao usera koji treba da se registruje. Nakon što smo registrovali usera, dobit ćemo strukturu u bazi.



U ovom slučaju, kao primjer, trenutni user jeste testni user sa test podacima.

Za autentifikaciju se koristila metoda `GetUser` koja uzima uneseni username i pretražuje isti u bazi, te pomoću username-a provjerava da li User postoji i da li su uneseni ispravni podaci. Code snippet se nalazi u nastavku.

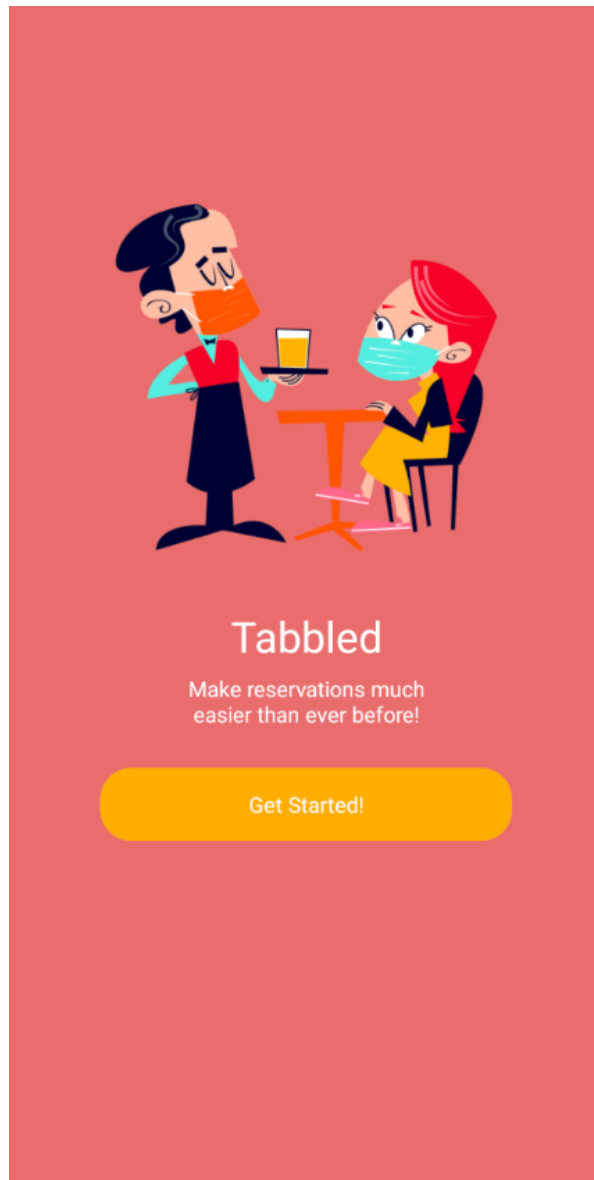
```
public static async Task<User> GetUser(string username)
{
    try
    {
        var allUsers = await GetAllUser();
        await firebaseClient
            .Child("Users")
            .OnceAsync<User>();
        return allUsers.Where(a => a.Username ==
username).FirstOrDefault();
    }
}
```

```
        catch (Exception e)
        {
            Debug.WriteLine($"Error:{e}");
            return null;
        }
    }
```

Ove dvije metode su temelj manipulisanjem podacima putem interneta i ispisivanje istih na mobilnu aplikaciju. U nastavku će se prikazati dijelovi koda u kojima se vrši registracija i login usera, odnosno flow aplikacije.


### 3.4 Views

U paketu Views se nalaze svi prikazi i većinska logika



*EntryScreen*

Na slici iznad se nalazi EntryScreen koji se pali pri pokretanju aplikacije i koji nas klikom na dugme vodi na registraciju korisnika.



Enter username:

Username

Enter email:

Email

Enter password:

Password

Confirm password:

Confirm password

Register

Already a user? Sign In.

Sign In

*RegisterScreen*

XAML elementi koji su korišteni prilikom izrade Register screena su: Entry, Label, StackLayout i Button, kao i slika iznad.

Nakon što korisnik unese podatke, naravno ispravno, nastavlja dalje klikom na Register dugme. Ukoliko već posjeduje račun, klikom na Sign In automatski se prebacuje na screen za Login. U nastavku će se prikazati code snippet u kojem se prikazuje proces registracije korisnika.

```

public async void SignIn_Clicked(object sender, EventArgs e)
{
    if (string.IsNullOrEmpty(EntryUsername.Text) ||
string.IsNullOrEmpty(EntryPassword.Text) ||
string.IsNullOrEmpty(EntryEmail.Text) ||
string.IsNullOrEmpty(EntryConfirmPassword.Text))
        await DisplayAlert("Empty Values", "Please enter
all of the values", "OK");
    else
    {
        if (EntryPassword.Text ==
EntryConfirmPassword.Text)
        {
            var users = await
FirebaseHelper.AddUser(EntryUsername.Text, EntryEmail.Text,
EntryPassword.Text, EntryConfirmPassword.Text);


            if (users)
            {
                await DisplayAlert("SignUp Success", "",
"OK");

                await Navigation.PushModalAsync(new
LoginPage());
            }
            else
            {
                await DisplayAlert("Error", "SignUp Fail",
"OK");
            }
        }
        else
        {
            await DisplayAlert("Error", "Passwords do not
match!", "Continue");
        }
    }
}

```

```
}  
  
}
```

Metoda `SignIn_Clicked` se poziva na klik za registraciju te najprije provjerava da li su svi podaci uneseni i dali postoji neki field koji nije napisan. Nakon toga slijedi provjera da li je uneseni password ispravna potvrda istog, ukoliko nije izbacuje error. Ukoliko su sve provjere prošle, deklariše se nova varijabla `var users` pomoću koje se korisnik smješta u bazu te nakon toga služi za provjeru ispravnosti procesa registracije. Nakon što se korisnik registrovao, slijedi Login.



Enter username:

Username

Enter password:

Password

Sign In

*LoginScreen*

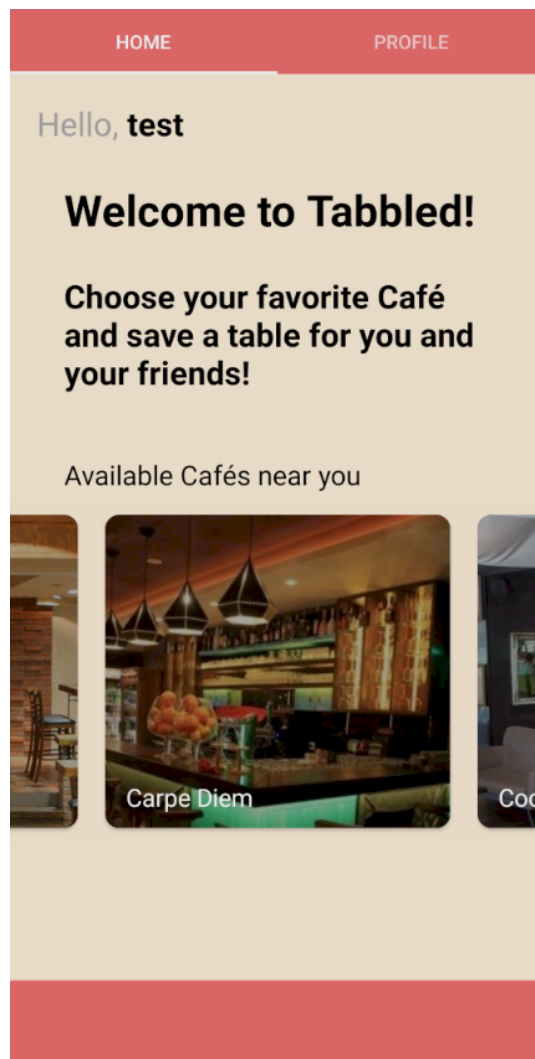
Login screen koristi iste XAML elemente kao i register page. U nastavku će se navesti dio koda koji opisuje provjeru korisnika u bazi.

```
public async void MainScreen_Clicked(object sender, EventArgs e)
{
    if (string.IsNullOrEmpty(LoginUsername.Text) ||
        string.IsNullOrEmpty(LoginPassword.Text))
        await DisplayAlert("Empty Values", "Please enter
Username and Password", "OK");
    else
    {
        //call GetUser function which we define in
        Firebase helper class
        var user = await
        FirebaseHelper.GetUser(LoginUsername.Text);
        //firebase return null valuse if user data not
        found in database
        if (user != null)
            if (LoginUsername.Text == user.Username &&
                LoginPassword.Text == user.Password)
            {
                await DisplayAlert("Login Success", "",
                "Ok");

                await Navigation.PushModalAsync(new
                MainScreen(LoginUsername.Text));
            }
            else
                await DisplayAlert("Login Fail", "Please
                enter correct Email and Password", "OK");
            else
                await DisplayAlert("Login Fail", "User not
                found", "OK");
        }
    }
}
```

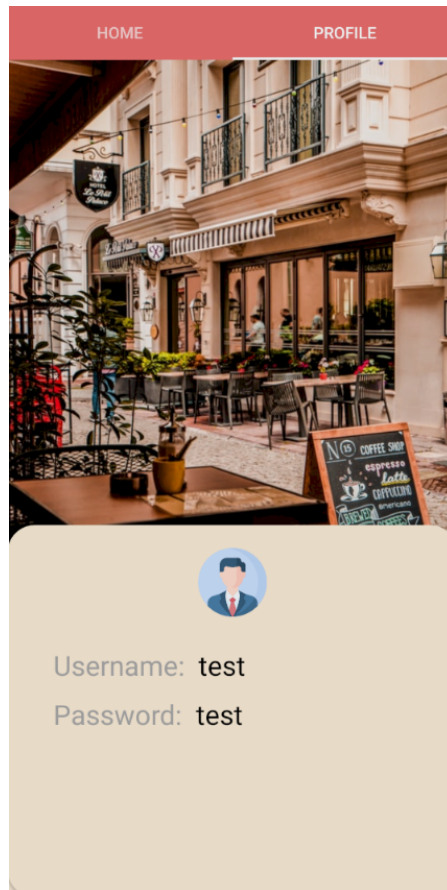


U funkciji `MainScreen_Clicked` za razliku od one pri registraciji, koristi se metoda `getUser` koja se poziva iz prije navedenog `FirebaseHelper`a. Ali prije samog pozivanja metode `getUser`, izvršava se provjera da li su sva polja unesena, a zatim, varijabla `user`, koja prima metodu `getUser`, provjerava da li `user` uopste postoji. Ukoliko je sve uredu, koristi se `await Navigation.PushModalAsync(new MainScreen(LoginUsername.Text));` dio koda koji nas prebacuje na drugu stranicu, pri čemu se u `MainScreen` proslijeđuje parametar `username`-a koji će nam služiti za ispis usera na glavnoj stranici.



*MainScreen*

Na slici iznad se nalazi glavni prikaz aplikacije, gdje su izlistani svi kafići na koje korisnik može kliknuti i rezervirati stol. Treba također napomenuti da ovaj prikaz sadrži i tabove. Na drugom tabu pod nazivom "Profile" se nalaze podaci o korisniku kao što je prikazano na slici ispod.



*ProfileScreen*

S obzirom da se ovdje radi o testnom korisniku, podaci koji su navedeni iznad su prototipnog tipa.

U samom mainScreen-u, bitno je istaknuti komponentu XAML-a koja se koristila pri izradi ovog prikaza a to je CarouselView, koji po mišljenju mnogih predstavlja nešto moderniji dizajn i funkcionalnost. Implementacija CarouselView-a je prikazana na code snippetu ispod.

```
<CarouselView x:Name="Caffes" PeekAreaInsets="60"
HeightRequest="250" Margin="0, 0, 0, 0">
    <CarouselView.ItemTemplate>
        <DataTemplate>
            <StackLayout>

<StackLayout.GestureRecognizers>
                                <TapGestureRecognizer
Tapped="DetailPage_Clicked"/>
        </DataTemplate>
    </CarouselView.ItemTemplate>
</CarouselView>
```

```

</StackLayout.GestureRecognizers>
                                <Frame HeightRequest="300"
WidthRequest="400"
                                BackgroundColor="White"
                                Padding="0"
                                HasShadow="True"
Margin="10" CornerRadius="10"
                                HorizontalOptions="CenterAndExpand">
                                <Grid>
                                    <StackLayout
                                        <Image
                                            Opacity="0.7"
                                        />
                                    </StackLayout>
                                    <StackLayout
                                        <Label
                                            Text="{Binding Name}"
                                            TextColor="White"
                                            FontSize="18"
                                            FontAttributes="None" Margin="15,0"
                                        />
                                    </StackLayout>
                                </Grid>
                            </Frame>
                        </StackLayout>
                    </DataTemplate>
                </CarouselView.ItemTemplate>
            </CarouselView>

```

Razlika između CarouselView-a i klasičnog ListView-a jeste ta što CarouselView, na uštrb estetike, ne posjeduje neke ugrađene funkcije za klikanje na dodane “item-e”, nego se ona ručno ugrađivala pomoću <StackLayout.GestureRecognizers> koji poziva funkciju nazad u logičkom dijelu koda.

```
CaffeList.Add(new Caffee { Id = 1, Capacity = 150, Name = "Carpe
Diem", Description = "The title, Seize the Day, describes an
endless possibility of relaxing the entire day.", Image =
"carpediem.jpg" });

CaffeList.Add(new Caffee { Id = 2, Capacity = 100, Name =
"Cooltura", Description = "A terrific location (City Arena), an
excellent location for the whole family.", Image = "cooltura.jpg"
});

CaffeList.Add(new Caffee { Id = 3, Capacity = 80, Name = "Da
Vinci's Pub", Description = "A place where the youth goes to have
an amazing time.", Image = "davincipub.jpg" });

CaffeList.Add(new Caffee { Id = 4, Capacity = 110, Name =
"Libertas Pub", Description = "As a part of a hotel, the Libertas
Pub is a frequent place of marvelous joy.", Image = "libertas.jpg"
});
```

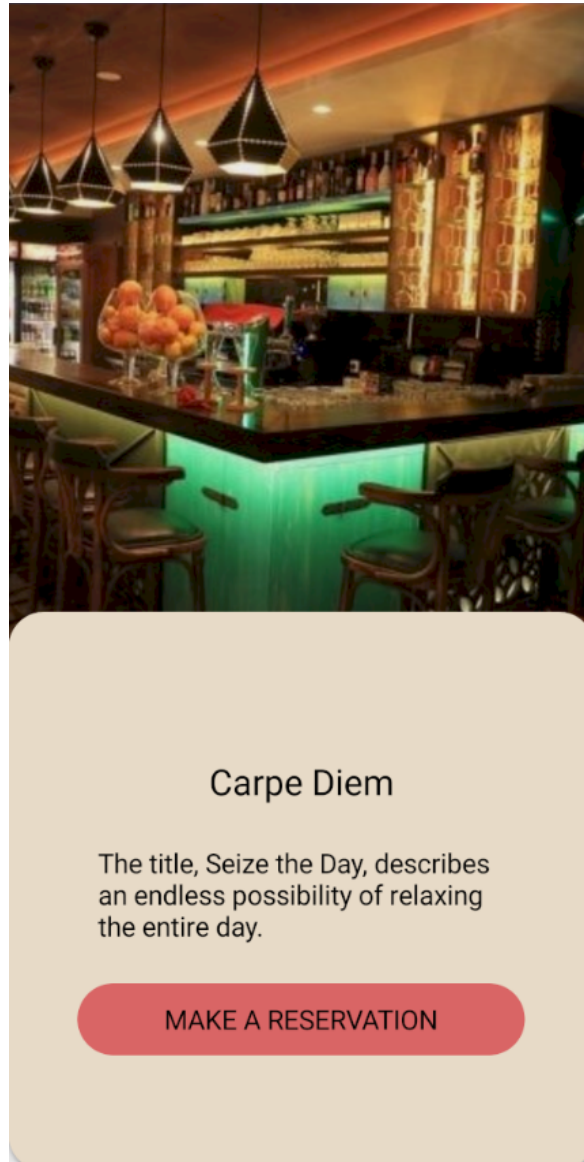
Prikaz unosa različitih kafića u aplikaciju.

```
public async void DetailPage_Clicked(object sender, EventArgs e)
{
    Caffee selectedCaffe = new Caffee();
    var stack = sender as StackLayout;
    var caffe = stack.BindingContext as Caffee;

    selectedCaffe = CaffeList.Where(x => x.Id ==
caffe.Id).FirstOrDefault();

    await Navigation.PushModalAsync(new
CaffeDetail(selectedCaffe, userUsername));
}
```

Funkcija koja se poziva je `DetailPage_Clicked`, koja ima zadatak da nas proslijedi na “Detail Page” kafića koji je izabran kako bi mogli vidjeti više informacija o samom kafiću i kako bi korisnik mogao rezervirati stol.



*DetailPageScreen*

Detail page screen nam pokazuje detaljnije informacije o odabranom kafiću i pruža nam funkciju rezervacije stola. Nakon što korisnik klikne na dugme “Make a reservation”, prije spomenuta varijabla `Reserved` se postavlja na `true` te tako se korisniku ostavlja stol u nekom od odabranih objekata.

## 4. Zaključak

Ova aplikacija kao glavnu svrhu ima olakšavanje rezervacija nekog stola u kafiću, ali ako pogledamo malo širu sliku, ovo je aplikacija koja ubrzava dosta sfera života modernog čovjeka, a pogotovo mladih ljudi.

Ljudi u gradovima kao što su New York City ili London, rade naporno svaki dan, ali naravno nakon radne sedmice vole da uživaju sa svojim prijateljima u nekom od najdražih mjesta. Ubrzan ritam takvih gradova ne pruža mladim ljudima puno slobodnog vremena tokom radnih sati, a upotrebom “Tabbled” aplikacije, izbjegava se pozivanje kafića, što nekad može potrajati i po 15 do 20 minuta s obzirom na gužvu ili zauzeće kafića.

Projekat koji je predstavljen jeste aplikacija koja je prototip jedne aplikacije koja bi mogla da uvede dosta promjena u živote modernog čovjeka i mislim da ima veoma bogat prostor za dodatne ideje koje bi mogle nadopuniti ovaj projekat.

U ovom dokumentu su navedeni samo osnovni i glavni dijelovi koda, ali ne i čitav kod.