Razvoj Windows aplikacije za iznajmljivače

Košmrl, Simon

Sveučilište u Rijeci – Odjel za informatiku

Jednopredmetna informatika

Simon Košmrl

Razvoj Windows aplikacije za iznajmljivače

Završni rad

Mentor: Doc. dr. sc. Marija Brkić Bakarić

Rijeka, 13.7.2018

Sadržaj

Sažetak

Kao praktični dio završnog rada izradio sam Windows aplikaciju za iznajmljivače. Za pristup aplikaciji potrebna je registracija i prijava korisnika. Na glavnom zaslonu korisnik odabire kategoriju stvari koje želi iznajmiti te ga aplikacija usmjerava na zaslon gdje se prikazuju svi oglasi te kategorije koji nisu iznajmljeni. Odabirom oglasa korisnik ima opciju poslati zahtjev za iznajmljivanje. Vlasnik oglasa odlučuje kojem korisniku želi iznajmiti stvar. Oglasi koje trenutni korisnik iznajmljuje prikazuju se samo u drugoj kartici glavnog prozora. Stvari koje je trenutni korisnik unajmio nalaze se u trećoj karti glavnog prozora. Aplikacija je napravljena sa Xamarin-om koji koristi XAML za prikaz UI (engl. *User Interface*) i C# klase za kontroliranje događaja koji se odvijaju u aplikaciji. Baza aplikacije nalazi se u Firebase projektu kao i autentifikacija korisnika. Aplikaciju sam nazvao RentApp.

Ključne riječi

Windows univerzalna aplikacija, Xamarin, Xamarin.Forms, Firebase, mobilna aplikacija, XAML, C#

Uvod

Posljednjih godina pametni telefoni preplavili su svijet te je paralelno tome i razvojna industrija mobilnih aplikacija brzo rastuća. Dodatno tome, Microsoft sa izdanjem Windows 10 napravio je novi korak u povezivanju univerzalnosti Windows platforme te time omogućio pokretanje univerzalnih Windows aplikacija na svim njihovim uređajima kao što su Xbox, Surfice Hub, IoT (engl. *Internet of things*) i ostali. Zbog toga tema ovog završnog rada je izrada Windows Phone / Univerzalne Windows Aplikacije za iznajmljivače. Pošto sam do sada imao dosta iskustva u izradi Android i iOs aplikacija htio sam naučiti nešto novo te se okušati u .NET okruženju i izradi aplikacije za Windows platformu.

Završni rad podijelio sam u dva dijela. U prvom dijelu rada opisati ću tehnologije i programe kojima sam se koristio prilikom izrade aplikacije, dok ću u drugom dijelu opisati sam proces izrade aplikacije, pravila dobrog programiranja te razmišljanja i logiku po kojoj se program izvršava. Poslije glavnih cjelina iznijeti ću nedostatke i moguća poboljšanja aplikacije kako bi korisnici lakše upravljali njome. Na kraju rada iznijeti ću zaključak te svoja mišljenja o aplikaciji i tehnologijama koje sam koristio.

Korištene tehnologije

Visual Studio 2017

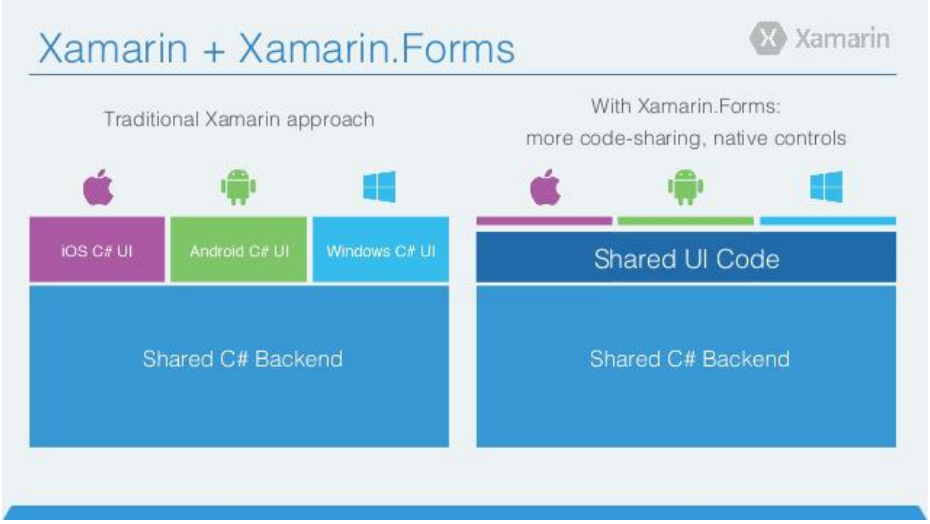
Visual Studio je Microsoft-ovo integrirano razvojno okruženje (engl. *IDE*). Programeri ga primarno koriste za razvoj Windows aplikacija, no koristi se i pri izradi web stranica, web servisa te u posljednje vrijeme za izradu mobilnih aplikacija. Unutar programa integriran je program za pronalaženje i otklanjanje grešaka (engl. *debugger*) koji radi na razini izvornog koda (engl. *source-level debugger*) i razini mašinskog koda (engl. *machine-level debugger*) te time programerima olakšava rad. Podržava 36 različitih programskih jezika od kojih su najpopularniji C, C#, C++, JavaScript, XML, XAML, HTML i CSS. U Visual Studio moguće je dodati i podršku za verzioniranje koda/softvera kao i druge alate za razvoj kao što je Xamarin.

Firebase

Kod izrade mobilnih aplikacija često se pojavljuje problem u izradi baze podataka. Komunikacija između uređaja i servera na kojem se nalazi baza podataka vrlo je važna,ali često nije dobro optimizirana. Firebase je platforma, koju je 2014. godine preuzeo Google, koja služi kod izrade mobilnih aplikacija. Ta platforma nam omogućava dobro optimiziranu bazu podataka i jedan od najbržih prijenosa podataka između uređaja i servera. Real Time Database, kako je Firebase naziva, u stvarnom vremenu prenosi podatke sa uređaja u bazu, odnosno istog trenutka kada korisnik pritisne gumb, što tu bazu čini jednom od najbržih na svijetu. Osim baze podataka teško je osmisliti i napraviti autentifikacijski sustav. Firebase-om rješavamo i taj problem. Postoje više načina kojima Firebase autentificira korisnike, a to su autentifikacija pomoću društvenih mreža kao što su Facebook, Twitter, Goolgle autentifikacija, autentifikacija pomoću broja mobitela te najjednostavniji oblik - e-mail i password. Osim tih funkcionalnosti Firebase nam nudi i mogućnost Push notifikacija, Cloud Messaging-a i mnoge druge. Ukratko rečeno Firebase je platforma koja služi za razvoj serverskog dijela mobilnih aplikacija bez pisanja koda.

Xamarin

Xamarin.Forms je programska biblioteka koji omogućava izradu izvornih mobilnih aplikacija za iOS, Android i Windows Phone koristeći C#. Razlog zbog kojeg me zainteresirao Xamarin je njegov UI toolkit koji služi za izradu izgleda korisničkog sučelja koji je zajednički za sve platforme (Slika 1). To znači da pisanjem jednog koda radimo istu mobilnu aplikaciju istovremeno za 3 navedene platforme što programerima skraćuje vrijeme izrade.



Slika 1 - Xamarin UI poboljšanja sa Xamarin.Forms

Za izradu korisničkog sučelja Xamarin Forms koristi XAML (engl. *Extensible Application Markup Language*). XAML je jezik sličan XML-u kojeg je kreirao Microsoft kao zamjena za kodne inicijalizacije objekta (elemenata korisničkog sučelja). U XAML datoteci definiramo koje elemente korisničkog sučelja želimo imati, kako će ti elementi izgledati te gdje će se nalaziti na stranici. Iza svake XAML datoteke imamo nešto što nazivamo "kod iza". To je klasa u C# koja implementira ponašanja za tu stranicu, npr. što se desi kada korisnik pritisne na gumb ili odabere neku vrijednost iz padajuće liste.

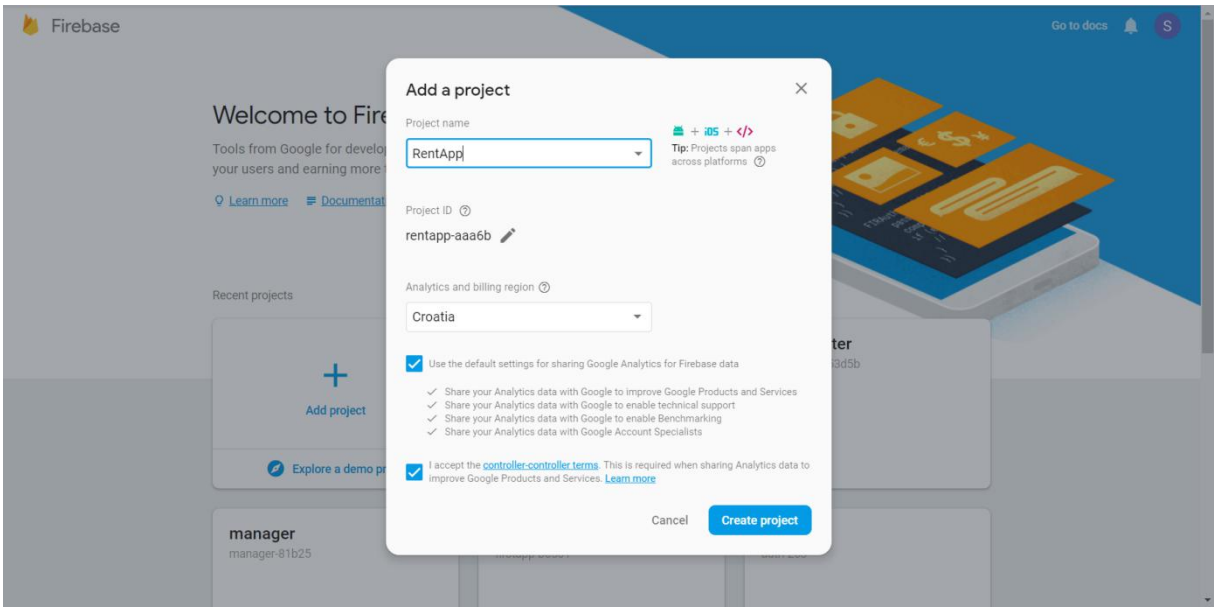
Gorilla Player

Da bi vidjeli promjene elemenata korisničkog sučelja sa Xamarin-om potrebno je ponovno pokretati aplikaciju, što je jedini nedostatak tog softverskog okvira. Gorilla Player je desktop program i dodatak Visual Studio-u koji služi za uživo prikazivanje XAML promjena te nam skraćuje vrijeme kod dizajniranja aplikacije. Za prikaz promjena potrebno je skinuti aplikaciju na mobilnom uređaju te je spojiti sa Visual Studio-om.

Izrada Aplikacije

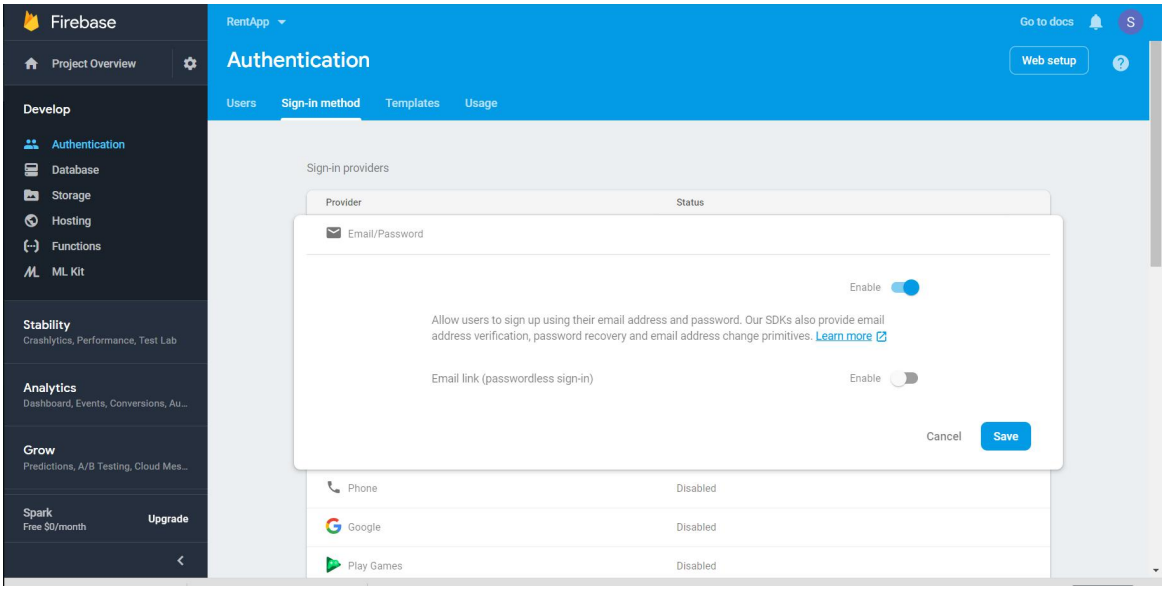
Konfiguracija servera - Firebase

Izradu aplikacije započet ćemo stvaranjem novog projekta u Firebase-u. U pregledniku napišemo adresu https://firebase.google.com/, a zatim se prijavimo pomoću Google računa. Nakon prijave dobivamo pristup konzoli. Iz konzole možemo vidjeti projekte (ukoliko postoje) ili izraditi novi projekt. Stvaranjem novog projekta otvara se novi prozor u kojemu je potrebno napisati ime projekta te odabrati regiju u kojoj se nalazimo (Slika 2). Pritiskom gumba *Create project* stvaramo novi projekt unutar Firebase-a.



Slika 2 - Stvaranje novog projeka u Firebase-u

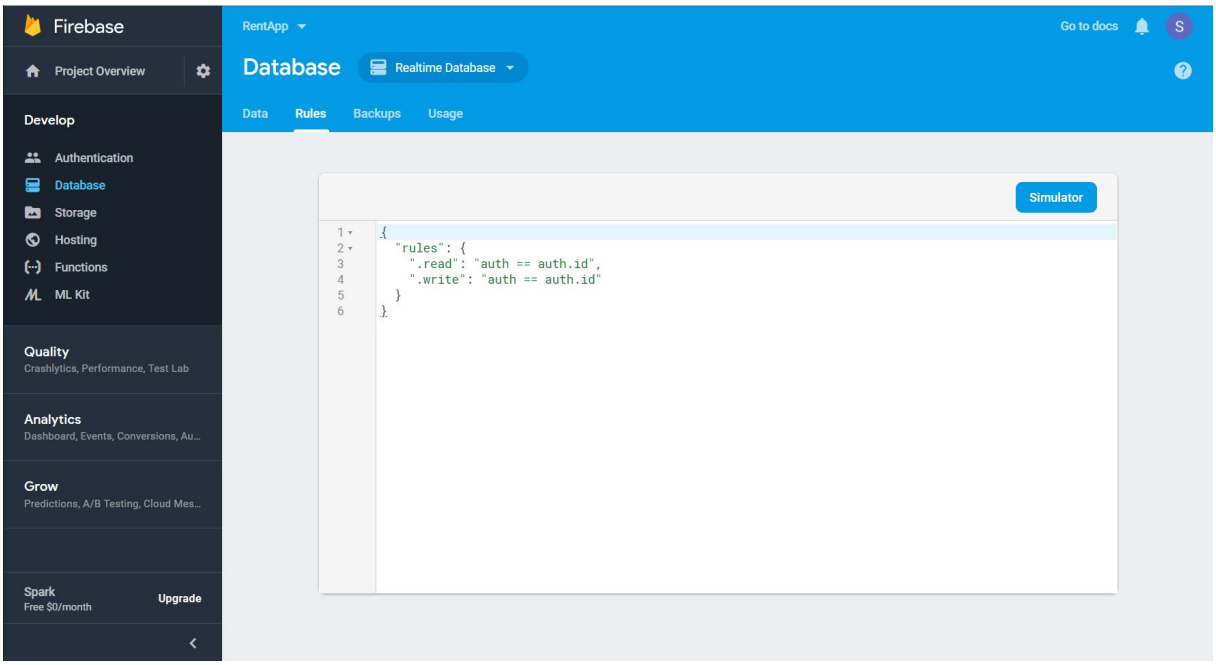
Kod izrade ove aplikacije od Firebase-a koristiti ćemo autentifikaciju i bazu podataka koje trebamo namjestiti. Unutar projekta u *Project Overview-u* sa desne strane klikom na *Authentication* otvaraju se postavke autentifikacije. U *Sign-in method* kartici omogučiti ćemo *Email/Password Sign-in provider*.



Slika 3 - Podešavanje metode prijave korisnika

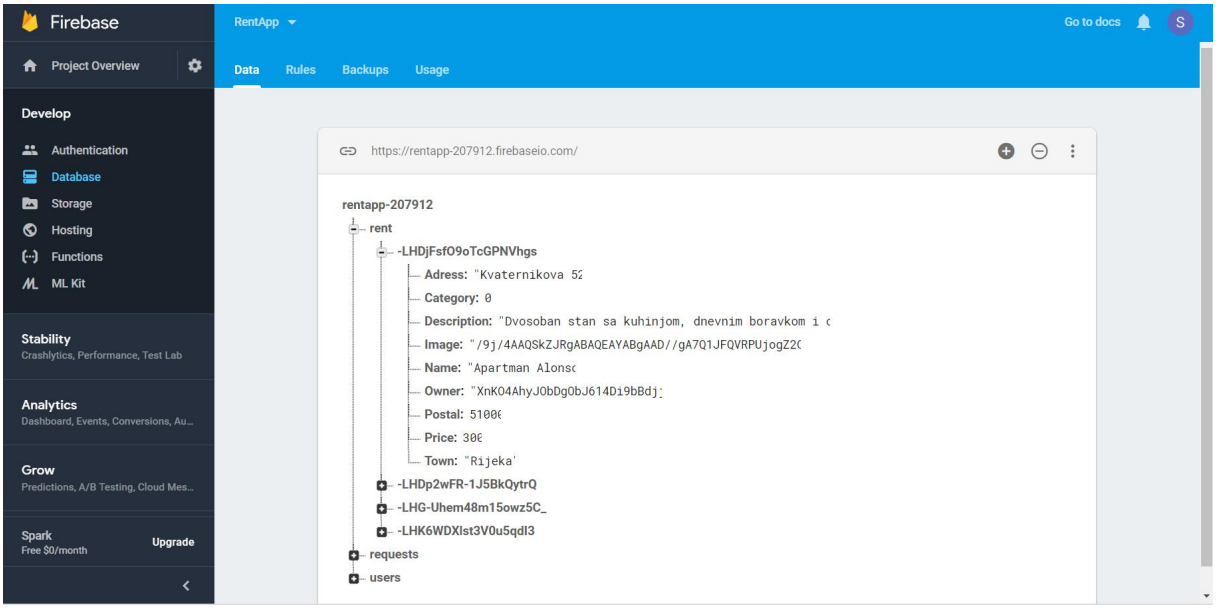
Klikom na *Save* podesili smo autentifikaciju. Kod podešavanja baze podataka (Slika 5) jedina stvar koju je potrebno namjestiti je dozvola čitanja i pisanja baze podataka. Želimo namjestiti da našu bazu podataka mogu čitati i pisati korisnici koji su registrirani u aplikaciji. Firebase nam daje opciju postavljanja tih pravila. *Project Overview -> Database -> Rules* unosimo pravila dozvole čitanja i pisanja podataka za prijavljene korisnike tako da aplikacija Firebase-u pošalje korisnikov id te se provjerava postojanost korisnika (Slika 4).

“.read” : “auth == auth.id”
“.write” : “auth == auth.id”



Slika 4 - Postavljanje pravila sigurnosti čitanja i pisanja podataka

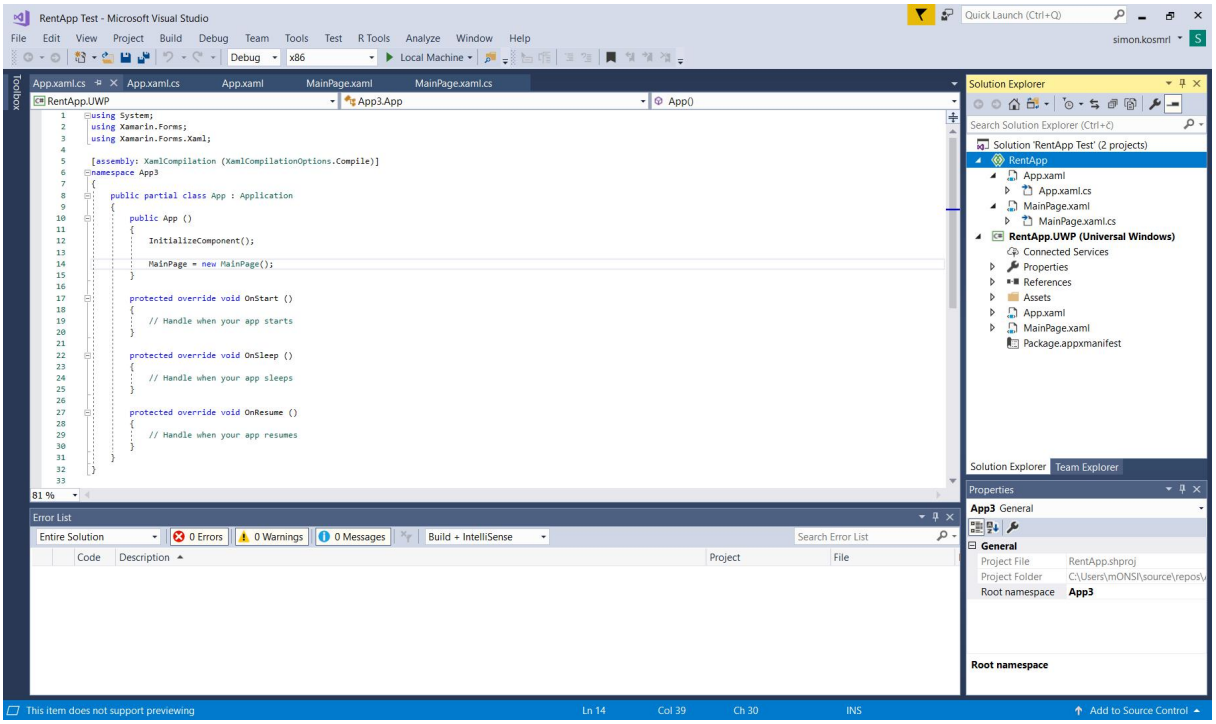
Ako je Firebase dozvolio promijene naše podešavanje je gotovo te možemo početi sa izradom aplikacije.



Slika 5 - Izgled popunjene baze u Firebase-u

Izrada projekta

Prvi korak pri izradi aplikacije je stvaranje novog projekta u Visual Studio-u a to postižemo sa *File -> New -> Project*. U prozoru za izradu projekta sa desne strane odaberemo vrstu projekta. Znamo da Xamarin.Forms koristi C# pa se naš template nalazi pod *Visual C# -> Cross Platform -> Mobile App (Xamarin.Forms)*. Ako ta opcija ne postoji nismo instalirali Xamarin proširenje u Visual Studio-u te ga je potrebno dohvatiti (*Tools -> Get Tools and Features*). Kada smo odabrali predložak dolazimo do novog prozora u kojem se odabiru platforme za koje se razvija mobilna aplikacija. U ovom projektu (Slika 6) izradit ću aplikaciju samo za Windows platformu.



Slika 6 - Izgled početnog projekta

Visual Studio je za nas stvorio novu soluciju sa dva glavna pod projekta (RentApp i RentApp.UWP). RentApp je generalni dio našeg projekta, sve datoteke ovdje su zajedničke datoteke, tj. kada bi imali više platformi (npr. RentApp.Android i RentApp.iOS) sve platforme bi koristile kod tih datoteka. RentApp.UWP služi nam za dodavanje NuGet paketa i programskih biblioteka za Univerzalnu Windows platformu te za namještanje postavka te platforme. App.xaml i App.xaml.cs su datoteke kojima Xamarin kompajlira aplikaciju. U toj klasi nalaze se metode životnog ciklusa (engl. *life cycle methods*) aplikacije te možemo postaviti početnu stranicu aplikacije.

```
MainPage = new MainPage();
```

Registracija i prijava korisnika

Za registraciju korisnika dodati ćemo novi *Content Page* (Slika 8). Desnim klikom na generalni dio solucije *RentApp -> Add -> New item* otvara se prozor za dodavanje nove datoteke. Imenujemo je *RegistrationPage* i odaberemo tip *Content Page*-a koji se nalazi pod *Xamarin.Forms*. Ukratko rečeno *Content Page* je tip stranice koja prikazuje sadržaj koristeći XAML. Ta stranica prikazivat će referencirana polja za upis (engl. *Entry*) i jednu tipku za registraciju (engl. *Button*).

```
<Entry
    Placeholder="Email"
    x:Name="email" />
<Entry
    Placeholder="Name"
    x:Name="name" />
<Entry
    Placeholder="Surname"
    x:Name="surname" />
<Entry
    Placeholder="Phone number"
    x:Name="phone" />
<Entry
    IsPassword="true"
    Placeholder="Password"
    x:Name="password" />
<Entry
    IsPassword="true"
    Placeholder="Confirm password"
    x:Name="passwordC" />
<Button
    Text="Register"
    Clicked="RegisterClicked"></Button>
```

Ti elementi postavljeni su unutar *StackLayout*-a kako bi bili redom prikazani. Kod svakog *Entry*-a, x:Name označava referencu na taj element kako bi laške kontrolirali njihovim svojstvima unutar klase te stranice. *Button* element sadrži *Clicked* metodu koja poziva funkciju *RegisterClicked* koju ćemo stvoriti u klasi stranice. U njoj će se odvijati procesi registracije korisnika.

```
private async void RegisterClicked(object sender, EventArgs e) {
    ...
}
```

Ta funkcija je asinkrona (*async*) jer će spremati podatke u Firebase bazu.

Da bi pristupili upisanom tekstu korisnika postaviti ćemo varijable koje poprimaju vrijednosti svojstva *Text* referenciranih elemenata.

```
var email_text = email.Text;
var password_text = password.Text;
var name_text = name.Text;
var surname_text = surname.Text;
var phone_number = phone.Text;
var password_c = passwordC.Text;
```

Želimo da korisnik upiše sve informacije pa prije registracije postavljamo provjeru da li je sve upisano te se korisnik upozorava u slučaju da nije sve upisano.

```
await DisplayAlert("Registration failed", "To complete registration provide all
    required information!", "OK");
```

Za lakše spajanje sa Firebase-om te kontrolu registracije i prijave korisnika preuzeti ćemo programsku biblioteku *FirebaseAuthentication.net* koja je dostupna kao NuGet paket u Visual Studio-u. Nije sigurno da će se korisnik uspijeti registrirati pa koristimo sintaksu *try - catch* (gdje *catch* vraća informacije greške registracije). Ako registracija sa *Firebase.Auth* metodom *CreateUserWithEmailAndPasswordAsync* nije uspjela odvija se *catch* te korisnika obavještavamo zašto je odbijen zahtjev (npr. email adresa već postoji).

```
using Firebase.Auth;
...
try {
    var auth = new FirebaseAuthProvider(new FirebaseConfig(firebaseApiKey));
    var user = await auth.CreateUserWithEmailAndPasswordAsync(email_text, password_text);
     ...
}
catch (FirebaseAuthException error) {
    ...
}
```

Nakon odobrenog zahtijeva spremamo unesene informacije u bazu podataka kako bi ih kasnije mogli iskoristiti. Iz NuGet paketa preuzeti ćemo *FirebaseDatabase.net* koji je kao i *FirebaseAuthentication.net* programska biblioteka za .NET koja nam pomaže pri radu sa Firebase-om. Bazi šaljemo objekt i spremamo ga u "*users*" sa ključem korisnikovog *id*-a. Napraviti ćemo novu klasu *UserInfo* koja će nam služiti i za dohvaćanje informacija korisnika. Klasa *UserInfo* sadrži atribute *name,surname,email,phone,id* te sadrži konstruktor za njih.

```
using Firebase.Database;
...
var firebase = new FirebaseClient(firebaseUrl);
await firebase
    .Child("users")
    .Child(userId)
    .PutAsync(new UserInfo(email_text, name_text, surname_text, phone_number,null));
```

Za prijavu korisnika napraviti ćemo novi *Content Page* (Slika 7). Na toj stranici prikazani su 2 *Entry*-a (*email* i *password*), *Button* za prijavu i *Button* za prijelaz na registracijsku stranicu. U klasi potrebno je napraviti funkcije koje su "prikvačene" na pojedine tipke. Pritiskom na tipku "*Create account*" pomoću *stack* navigacije otvara se registracijska stranica te se na njoj sada prikazuje i strelica unatrag koja vraća aplikaciju nazad na stranicu za prijavu.

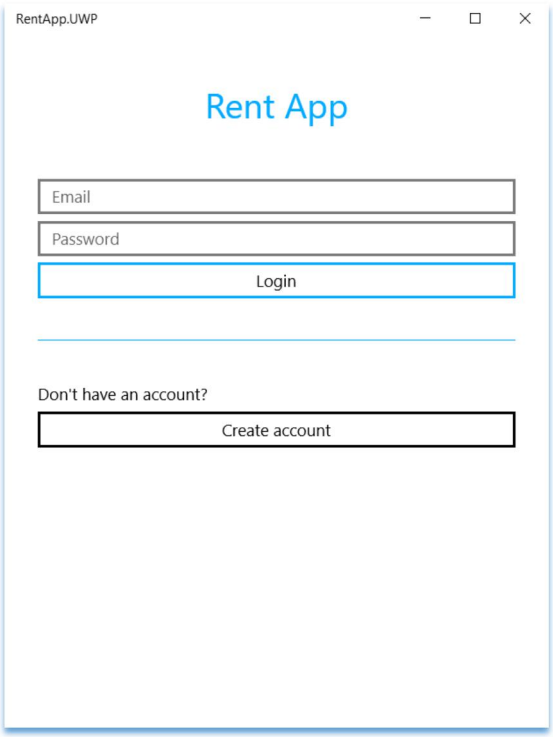
```
    await Navigation.PushAsync(new RegistrationPage());
```

Pritiskom na "*Login*" pomoću *try - catch* sintakse koja je slična registracijskoj prijavljujemo korisnika i obavještavamo ga ukoliko je došlo do pogreške. Ako se prijava izvrši bez greške preusmjeravamo korisnika na glavni prozor naše aplikacije i spremamo dobiveni *token* i *id* korisnika pomoću još jednog NuGet paketa *SecureStorage*.

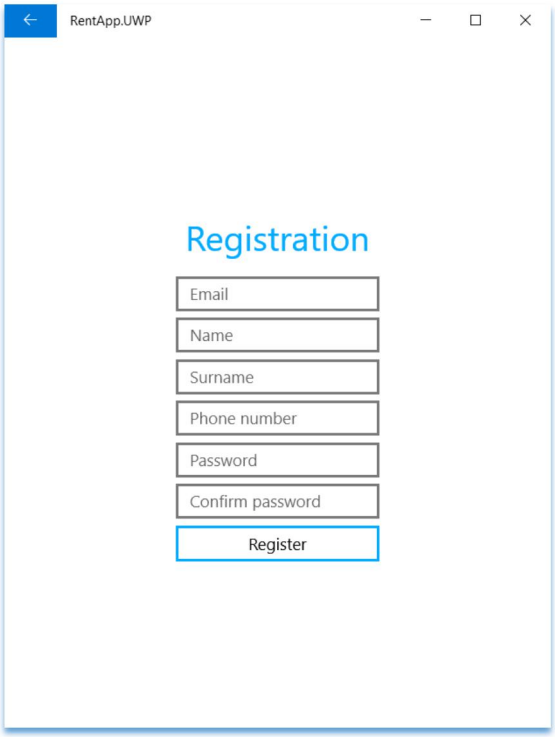
```
using Firebase.Auth;
using Plugin.SecureStorage;
...
    var token = await auth.SignInWithEmailAndPasswordAsync(email_text, password_text);
    CrossSecureStorage.Current.SetValue("user_token", token.FirebaseToken);
    CrossSecureStorage.Current.SetValue("user_id", token.User.LocalId);
    await Navigation.PushModalAsync(new NavigationPage(new MainPage()));
```

Kako bi korisniku olakšali rad sa aplikacijom i riješili ga muke kao što je nepotrebna prijava svaki puta kada se aplikacija pokrene, u App.xaml.cs mijenjamo početne stranice naše aplikacije ovisno o tome da li je spremljen "*user_token*". Taj token nam kasnije u aplikaciji po potrebi služi za ponovnu prijavu korisnika ukoliko je sesija sa Firebase-om istekla.

```
if (CrossSecureStorage.Current.HasKey("user_token")) {
    MainPage = new NavigationPage(new MainPage());
} else {
    MainPage = new NavigationPage(new LoginPage());
 }
```



Slika 7 - Izgled stranice za prijavu korisnika



Slika 8 - Izgled stranice za registraciju korisnika

Glavni prozor aplikacije

Za glavnu stranicu aplikacije izraditi ćemo novi *Tabbed Page* kojeg nazivamo *MainPage*. Prilikom prijave korisnika u *LoginPage*-u preusmjeriti ćemo ga na novo kreirani *MainPage*. Unutar *Tabbed Page*-a Xamarin nam daje opciju da ubacimo onoliko kartica koliko imamo *Content Page-ova* koji prikazuju sadržaje pojedine kartice. Prije nego što prikažemo sadržaj korisniku prikazujemo indikator aktivnosti koji korisniku upućuje da se nešto u programu izvodi.

```
...
<ActivityIndicator  Color="Black"
                    x:Name="loader"
                    IsRunning="True"
                    VerticalOptions="Center" HorizontalOptions="Center"
                    WidthRequest="50" HeightRequest="50" />
...
```

Dohvaćanje podataka iz baze i izrada kolekcije objekata

Za dohvaćanje svih podataka iz baze napraviti ćemo novu klasu *DBFirebase* koja će sadržavati metode za kompletan rad koji će se odvijati sa bazom podataka. U toj klasi svaki puta kada se napravi njezin objekt povezati ćemo aplikaciju sa Firebase-om.

```
namespace RentApp.Model
{
    public class DBFirebase
    {
        FirebaseClient fclient;
        public DBFirebase() {
        var firebaseUrl = "https://rentapp-207912.firebaseio.com";
        fclient = new FirebaseClient(firebaseUrl);
     }
     ....
```

Napraviti ćemo novu klasu *RentItem* koja će sadržavati atribute naših stvari za iznajmljivanje i klasu *Request* koja sadrži atribute *id,itemId* i *Rentee*. Klase također trebaju sadržavati konstruktore za te atribute. Unutar klase *DBFirebase* napravimo metodu za dohvaćanje stvari za iznajmljivanje, metodu za dohvaćanje zahtjeva za unajmljivanje i metodu za dohvaćanje informacija korisnika.

```
public async Task<List<RentItem>> GetItems()
        {
            var list = (await fclient
                .Child("rent")
                .OnceAsync<RentItem>())
                .Select(item =>
                   new RentItem(
                       item.Object.Name,
                       item.Object.Price,
                       item.Object.Adress,
                       item.Object.Town,
                       item.Object.Postal,
                       item.Object.Description,
                       item.Object.Image,
                       item.Object.Category,
                       item.Object.Rentee,
                       item.Object.Owner,
                       item.Key)).ToList();
            return list;
        }
```

Sve metode za dohvaćanje odrađuju se na isti način jedina razlika je mjesto čitanja podataka iz baze (*.Child*(“mjesto_čitanja”)) i objekti koji se popunjavaju. Te funkcije su *async Task*-ovi koji vraćaju listu popunjenih objekta *RentItem* - za stvari koje se iznajmljuju, *UserInfo* - za informacije korisnika te *Request* - za zahtjeve iznajmljivanja. U klasi *MainPage*-a prilikom stvaranja njegovog objekta pozivamo funkciju *GetData*. U toj funkciji napravimo novi objekt klase *DBFirebase* kao i varijable u koju se spremaju dohvaćeni podaci.

```
            var db = new DBFirebase();
            var listAsync = await db.GetItems();
            var userListAsync = await db.GetUserInfo();
            var requestListAsync = await db.GetRequests();
```

Za prikaz slika u Xamarinu potrebno je stvoriti strujanje (engl. *stream*) bitova slike. Stoga ćemo za prikazivanje stvari za iznajmljivanje stvoriti još jednu klasu *DisplayItem* koja je podklasa klase *RentItem*.

```
public class DisplayItem : RentItem
    {
        public ImageSource ImgSrc { get; set; }
        public UserInfo OwnerInfo { get; set; }
        public UserInfo RenteeInfo { get; set; }
        public Color Status { get; set; }

        public DisplayItem (RentItem Item, MemoryStream imgStream, UserInfo ownerInfo,
UserInfo RenteeInfo, Color status)

         ...
```

DisplayItem klasa, za razliku od *RentItem* klase, sadrži atribut *ImgSrc* tipa *ImageSource* koji nam služi za prikazivanje slika *Image* XAML elemenata, atribute *OwnerInfo* i *RenteeInfo* koji sadrže podatke o vlasniku iznajmljene stvari te ako je iznajmljena, podatke o osobi koja je unajmila tu stvar. Atribut *Status* služi nam za prikazivanje boje u slučaju ako je stvar iznajmljena ili ako postoji barem jedan zahtjev za iznajmljivanje.

U klasi *MainPage*-a stvoriti ćemo kolekcije objekata koje ćemo prosljeđivati ostalim stranicama aplikacije.

```
static ObservableCollection<RentItem> list = new ObservableCollection<RentItem>();
static ObservableCollection<UserInfo> users = new ObservableCollection<UserInfo>();
static ObservableCollection<Request> requests = new ObservableCollection<Request>();

static ObservableCollection<DisplayItem> myList = new
ObservableCollection<DisplayItem>();

static ObservableCollection<DisplayItem> myRent = new
ObservableCollection<DisplayItem>();
```

U fukciji *GetData* popuniti ćemo te kolekcije podacima koje smo dohvatili u varijable *listAsync*, *userListAsync*, *requestListAsync*.

```
foreach (var user in userListAsync) {
    users.Add(user);
}
```

Kolekciju request i list popunjavamo na isti način, dok u *myList* popunjavamo podatke stvari za iznajmljivanje kojima je vlasnik (*Owner*) korisnik koji je trenutno prijavljen u aplikaciji, a u kolekciju *myRent* stvari koje je trenutrni korisnik unajmio (gdje je trenutni korisnik *Rentee*).

Kod popunjavanja kolekcije *myList* koja sadrži objekte tipa *DisplayItem* dodajemo i boju. Ako je stvar iznajmljena prikazati će se zelena boja. Prolaskom kroz listu zahtjeva možemo dodati žutu boju ako postoji barem jedan zahtjev za tu stvar ili bijelu boju ako ne postoji.

```
bool inRequest = false;
foreach (var request in requestListAsync)
{
    if (request.ItemId == item.Id)
     {
     inRequest = true;
     }
}
if (inRequest)
    myList.Add(new DisplayItem(item, imgStream, null, null, Color.FromHex("#ffd503")));
else
    myList.Add(new DisplayItem(item, imgStream, null, null, Color.White));
```

Izgled glavnog prozora

Kada smo završili popunjavanje kolekcija, sakrivamo indikator aktivnosti i prikazujemo glavni prozor. Glavni prozor sastoji se od 3 kartice (*Home* , *My renting items*, *Profile*) stoga u XAML datoteci *MainPage*-a dodamo 3 *Content Page*-a.

Kartica profila

Kartica profila prikazuje informacije o prijavljenom korisniku te tipku za odjavu i tipku za promjenu informacija korisnika (Slika 9). Ispod toga prikazuje se lista stvari koje je korisnik unajmio. Listu u Xamarinu prokazujemo elementom *<ListView>*.

```
<ListView ItemsSource="{Binding .}" x:Name="_mRntLst" ItemTapped="OnMyItemClick">
    <ListView.ItemTemplate>
        <DataTemplate>
            <ViewCell>
                <ViewCell.View>
                    ...
                </ViewCell.View>
            </ViewCell>
        </DataTemplate>
    </ListView.ItemTemplate>
</ListView>
```

Xamarin nam nudi opciju vezivanja podataka (engl. *Data Binding*) što nam omogućuje prikazivanje podataka sa liste kolekcije objekata *myRent*. ListView za svaki objekt u listi napravi novi red te ga prikaže na ekranu. Ako se u listi ne nalazi niti jedan objekt prikazati će se obavijest korisniku da nema unajmljenih stvari. Jedina stvar koja je preostala za napraviti je povezati listu kolekcije i element *ListView*.

```
_mRntLst.BindingContext = myRent;
```

Pritiskom tipke Logout pokreće se funkcija koja briše spremljene podatke "*user_token*" i "*user_id*", kako bi pri ponovnom pokretanju aplikacije početna stranica bila *LoginPage*, te usmjerava korisnika na taj prozor.

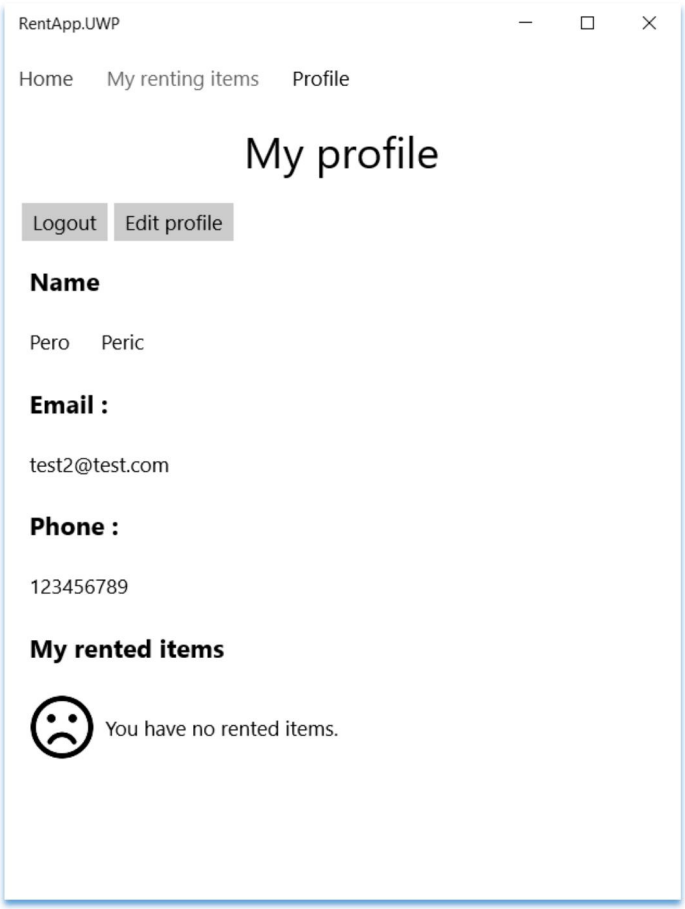
Pritiskom gumba *Edit profile* otvara se nova stranica *EditUser*.

```
private async void OnEditProfileClick(object sender, EventArgs e)
{
    await Navigation.PushAsync(new EditUser(currentUserInfo));
}
```

Promijena podataka korisnika

Napraviti ćemo novi *Content Page* kojeg ćemo nazvati *EditUser*. Ta stranica prikazivati će 4 *Entry*-a (jedan za svaki podatak korisnika) te jedan *Button* za prihvaćanje promijena. Klasi novo stvorene datoteke poslati ćemo informacije korisnika te ćemo popuniti tekst polja *Entry*-a tim podacima.

```
public EditUser (UserInfo userInfo)
        {
           userEmail.Text = userInfo.Email;
           userName.Text = userInfo.Name;
           userSurname.Text = userInfo.Surname;
           userPhone.Text = userInfo.Phone;
         }
```



Slika 9 - Izgled kartice profila

U klasi *DBFirebase* potrebno je stvoriti novu metodu koja će promijenjene podatke zapisati u bazu podataka koje se nalaze pod ključem trenutnog korisnika. Pritiskom na gumb "*Change profile information*" pozvati će se novostvorena metoda.

Kartica stvari za iznajmljivanje

U ovoj kartici glavnog prozora, prikazuje se lista stvari koje iznajmljuje trenutni korisnik. Kao i kod liste iznajmljenih stvari (u kartici profila) povezujemo listu sa kolekcijom objekta *myList*. Jedina razlika u ovim listama je ta što se ovdje prikazuju i boje, ovisno o tome da li postoji neki zahtjev za unajmljivanje. Ispod liste nalazi se *Button*. Na njegov pritisak otvara se novi *Content Page - AddItemPage*.

Dodavanje stvari za iznajmljivanje

Za dodavanje stvari u bazu također ćemo stvoriti novu stranicu (Slika 10). Na njoj ćemo postaviti sliku te kao i skoro sve elemente referencirati sa *x:Name = "aadRentImage"*.

```
<Image
    x:Name="addRentImage”>
```

Na njoj nalaziti će se dvije tipke. pritiskom na tipku *Add photo* otvoriti će se prozor za odabir slike koristeći alat za odabir datoteke. Ako se slika odabere napraviti ćemo dva strujanja slike, jedan *stream* ćemo pretvoriti u *Base64String* format kako bi sliku mogli pohraniti u bazu podataka. Drugi *stream* ćemo iskoristiti za prikaz slike. Windows-ov alat za odabir datoteka daje nam mogućnost i postavljanja filtera za pretraživanje po tipu datoteke.

```
using Windows.Storage;
using Windows.Storage.Pickers;
using Windows.Storage.Streams;
FileOpenPicker picker = new FileOpenPicker {
              ViewMode = PickerViewMode.Thumbnail,
              SuggestedStartLocation = PickerLocationId.PicturesLibrary,
           };
           picker.FileTypeFilter.Add(".jpg");
           picker.FileTypeFilter.Add(".jpeg");
           picker.FileTypeFilter.Add(".png");
           StorageFile file = await picker.PickSingleFileAsync();
           if (file != null)
           {
                Stream stream = await file.OpenStreamForReadAsync();
                ...
                Byte[] bytes = ReadFully(stream);
                String imageBase64 = Convert.ToBase64String(bytes);
                ...
            }
```

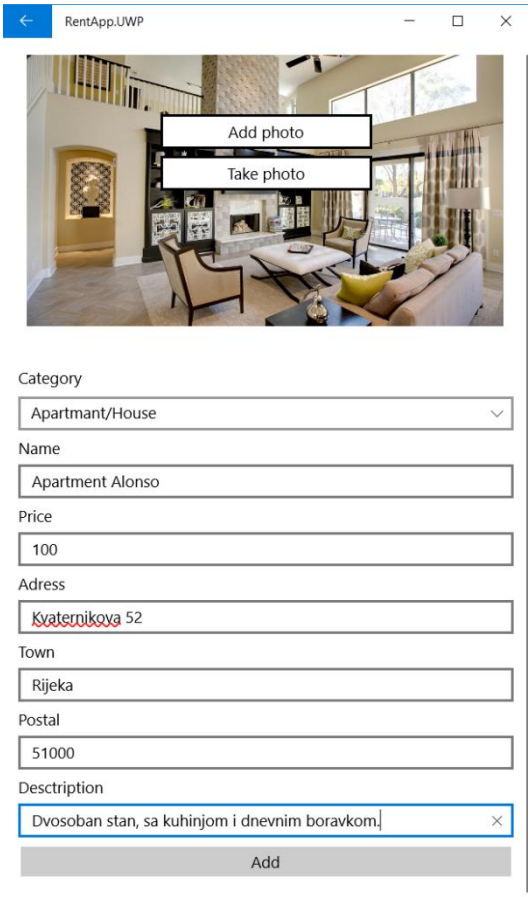
Druga tipka otvoriti će komponentu kamere kako bi korisnik mogao slikati stvar koju želi iznajmiti.

```
var cameraComponent = new CameraCaptureUI();
var file = await cameraComponent.CaptureFileAsync(CameraCaptureUIMode.Photo);
```

Uslikanu fotografiju spremamo u varijablu *file* koju kasnije pretvaramo ponovno u dva strujanja bitova. Ispod slike dodamo *Picker* kategorija prema kojima će se prikazivati stvari.

```
<Picker Title="Category" x:Name="addCategory">
    <Picker.Items>
        <x:String>Apartmant/House</x:String>
        <x:String>Vehicle</x:String>
        <x:String>Sport</x:String>
        <x:String>Tehnology</x:String>
        <x:String>Other</x:String>
    </Picker.Items>
</Picker>
```

Dodamo još polja sa unos (*Entry*) ostalih informacija kao i tipku "*Add*" na čiji pritisak se poziva funkcija iz klase *DBFirebase* koja sprema podatke u bazu pod novim ključem koji se napravi. Sve elemente korisničkog sučelja ove stranice "zamotati" ćemo u *ScrollView* element zbog toga što ima puno elemenata pa se korisniku u suprotnome ne bi prikazali svi elementi.

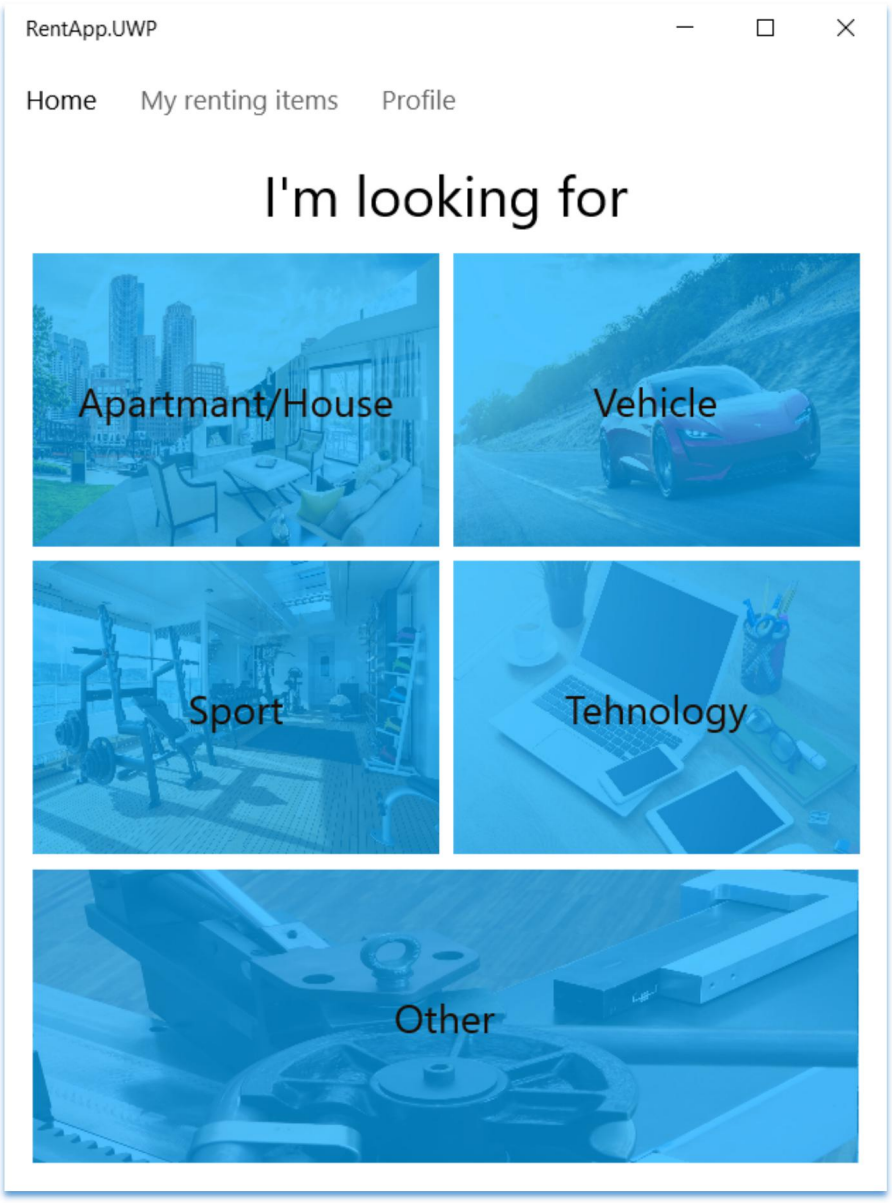


Slika 10 - Pupunjena stranica za dodavanje nove stvari

Početna kartica

Home kartica prikazuje *Label* element sa tekstom "*I'm looking for*", a zatim 5 *Button*-a za svaku kategoriju (Slika 11). Svaka tipka ima svoju *Clicked* metodu koja ovisno o kategoriji koju je korisnik pritisnuo otvara novi *Content Page* - *ItemsPage* te klasi te stranice šalje broj kategorije, listu kolekcija svih oglasa - *list* i informacije o svim korisnicima - *users*.

```
private async void SportClicked(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new ItemsPage(2, list, users));
        }
```



Slika 11 - Izgled Home kartice

Prikaz stvari za iznajmljivanje i prikaz detalja

U izradi mobilne aplikacije dobra je praksa iskorištavati isti kod. Stoga ćemo napraviti novi *Content Page* te ga imenovati *ItemsPage.* Ovdje će se prikazivati oglasi ovisno o kategoriji koja je izabrana u prošlom prozoru (*Home* kartica). Ovdje ćemo prikazivati sve oglase koji nisu vlasništvo trenutnog korisnika, nego nekog drugog te oglase koji nisu iznajmljeni i staviti ih u novu listu kolekcija objekta. Potrebno je pretvoriti i sliku (koja se trenutno nalazi u *Base64* obliku (*string*)) u strujanje bitova kako bi se prikazala slika.

```
ObservableCollection<DisplayItem> itemsList = new ObservableCollection<DisplayItem>();

public ItemsPage(int category, ObservableCollection<RentItem> list,
ObservableCollection<UserInfo> users) {

_iLst.BindingContext = itemsList;

...

foreach (var item in list)
{
     if (item.Category == category && item.Owner != Owner && item.Rentee == null)
     {
     byte[] byteresponse = Convert.FromBase64String(item.Image);
     MemoryStream imgStream = new MemoryStream(byteresponse);
     foreach (var user in users)
     {
         if(item.Owner == user.Id)
         {
             itemsList.Add(new DisplayItem(item, imgStream, user, null, Color.White));
         }
    }
    }
}
```

Kada pritisnemo na neki oglas želimo prikazati njegove detalje i poslati potrebne informacije na slijedeći prozor.

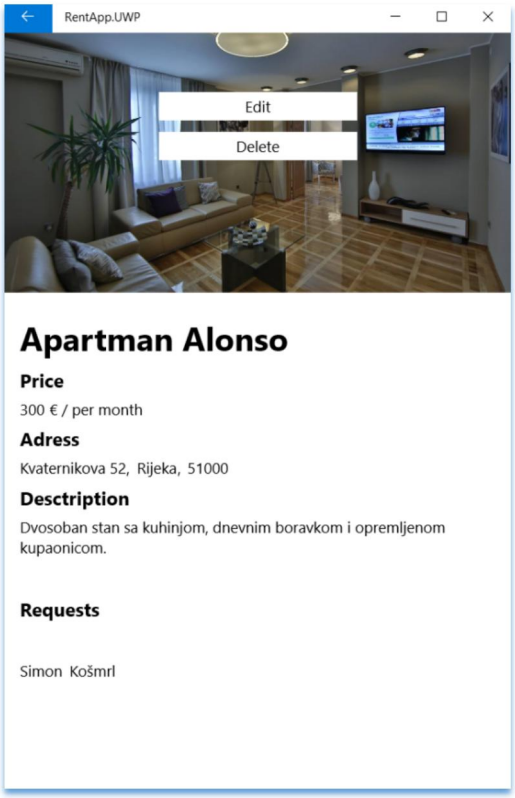
```
        private async void OnItemClick(object sender, ItemTappedEventArgs e)
        {
            var itemClicked = e.Item as DisplayItem;
            await Navigation.PushAsync(new ItemPage(itemClicked, null, null));
        }
```

Detalje prikaza prikazivat ćemo u novome *Content Page*-u imena *ItemPage* kojem šaljemo sve informacije o odabranom oglasu oglasu.

ItemPage sastavljen je sastavljen od slike oglasa, njegovih informacija, te informacija o vlasniku. Ispod informacija o vlasniku, postavljene su tipke koje za korisnika sastavljaju poruku, poziv ili napišu primjer elektroničke pošte kako bi se korisnici, ukoliko su zainteresirani, mogli javiti vlasniku. Na kraju prozora nalazi se *Request* tipka koja pokreće metodu iz klase *DBFirebase*-a koja dodaje novi zahtjev u bazu podataka. Za funkcionalnosti tih tipki možemo iskoristiti *Xam.Plugins.Messaging*, koji se može preuzeti kao NuGet paket.

```
private void RCallClicked(object sender, EventArgs e)
        {
            var phoneDialer = CrossMessaging.Current.PhoneDialer;
            if (phoneDialer.CanMakePhoneCall)
                phoneDialer.MakePhoneCall(itm.RenteeInfo.Phone);
        }
```

Da bi iskoristili *ItemPage* kao prikaz detalja oglasa za liste korisnikovih oglasa, možemo kontrolirati koje elemente želimo prikazivati na toj stranici ovisno o tome koje podatke joj pošaljemo, npr. ako postoje zahtjevi za taj oglas možemo umjesto informacija o vlasniku prikazati listu korisnika koji su se prijavili na taj oglas (Slika 12). Na toj listi možemo dodati kontrolu odabira kojem korisniku se iznajmljuje oglas, pa je potrebno dodati i novu metodu u klasi *DBFirebase* koja samo popunjuje polje *Rentee* u bazi podataka. Ako je oglas iznajmljen vlasniku se na stranici *ItemPage* ne prikazuje više lista, nego se prikazuju informacije o osobi koja je unajmila tu stvar kao i tipke za poruku,poziv ili e-mail toj osobi.



Slika 12 - Izgled detalja oglasa sa listom korisnika koji su poslali zahtjev

Zaključak

Osobno bilo mi je zanimljivo učiti i raditi sa Xamarin.Forms-om jer nudi izradu elegantnog korisničkog sučelja koji je isti za sve platforme koje podržava. Xamarin.Forms nudi nam bogatu kolekciju elemenata korisničkog sučelja koji su nam potrebi pri izradi profesionalnih mobilnih aplikacija. Objektni pristup izrade mobilnih aplikacija u C# sa Xamarin-om nudi nam bolju kontrolu nad objektima stoga ima prednost nad nekim drugim programskim okvirima. Firebase sa svojim *Real-time Database*-om nudi nam brzi zapis u bazu, a pošto se podaci dohvaćaju kao *JSON* datoteke, koje se unutar aplikacije pretvore u objekte te se sa njima lako upravlja. Windows-ovi alati za razvoj mnogo su bogatiji od ostalih platformi. Ti alati sadrže elemente kao što su kontroliranje kamere i odabir datoteke te je tim elementima lakše upravljati. .NET također nam nudi brzo pretvaranje niza bitova u strujanje koje koristimo za prikazivanje slika u aplikaciji. Ova aplikacija i projekt sadrži osnovne funkcionalnosti koje bi aplikacija za iznajmljivače trebala posjedovati, no ipak ima nekih nedostataka koji se lako dodaju.

Kod mobilnih aplikacija, korisnici najviše cijene jednostavnost korištenja. Da bi ova mobilna aplikacija bila jednostavnija za korištenje, *email* - *password* sistem prijave i registracije trebalo bi zamijeniti sa prijavom putem društvenih mreža kao što su Facebook i Twitter. Time bi se uklonila registracija zbog toga što preko tih društvenih mreža programeri mogu doći do korisnikovih informacija. Da bi aplikacija izgledala profesionalnije i još više se olakšala jednostavnost korištenja potrebna je suradnja sa dizajnerima mobilnih aplikacija. Dobar dizajn prva je stvar koju korisnik uoči unutar aplikacije. Da bi aplikacija bila praktična za korištenje trebao bi se dodati prozor u kojem se nalazi mapa te prikazuje oglase koji su u blizini korisnikove trenutne lokacije mobilnog uređaja. Valjalo bi dodati i neku vrstu filtera, npr. prikazivanje oglasa prema cijeni ili prema odabranom mjestu. Prilikom slanja zahtjeva na oglas, za vlasnika oglasa poželjno bi bilo da mu mobilnim notifikacijama javimo kada se netko prijavi na pojedini oglas.

Popis slika

Popis literature

An Introduction to Xamarin.Forms | Microsoft Docs - 12/02/2016 - https://docs.microsoft.com/en-us/xamarin/xamarin-forms/get-started/introduction-to-xamarin-forms

Xamarin.Forms XAML Basics | Microsoft Docs - 10/25/2017 - https://docs.microsoft.com/en-us/xamarin/xamarin-forms/xaml/xaml-basics/index

Microsoft Visual Studio - Wikipedia - 12 July 2018, at 21:34 (UTC) - https://en.wikipedia.org/wiki/Microsoft_Visual_Studio

Firebase Realtime Database | Firebase - May 8, 2018 - https://firebase.google.com/docs/database/

Github - step-up-labs/firebase-database-dotnet: C# library for Firebase Realtime Database - Apr 6, 2018 - https://github.com/step-up-labs/firebase-database-dotnet

Github - step-up-labs/firebase-authentication-dotnet: C# library for Firebase Authentication - Apr 4, 2018 - https://github.com/step-up-labs/firebase-authentication-dotnet

Github - sameerkapps/SecureStorage - May 21 , 2018 - https://github.com/sameerkapps/SecureStorage

Dokumnetacija projekta

DBFirebase.cs

```
using Firebase.Database;
using System;
using System.Linq;
using System.Collections.Generic;
using System.Text;
using System.Threading.Tasks;
using Firebase.Database.Query;
using System.Collections.ObjectModel;

namespace RentApp.Model
{
    public class DBFirebase
    {
        FirebaseClient fclient;
        public DBFirebase() {
        var firebaseUrl = "https://rentapp-207912.firebaseio.com";
        fclient = new FirebaseClient(firebaseUrl);

        }

        public async Task NewRentItem(string name, int price, string adress, string town,
int postal, string description, int category, string image, string owner)
        {
            await fclient
                .Child("rent")
                .PostAsync(new
RentItem(name,price,adress,town,postal,description,image,category,null, owner));
        }

        public async Task<List<RentItem>> GetItems()
        {

            var list = (await fclient
                .Child("rent")
                .OnceAsync<RentItem>())
                .Select(item =>
                   new RentItem(
                       item.Object.Name,
                       item.Object.Price,
                       item.Object.Adress,
                       item.Object.Town,
                       item.Object.Postal,
                       item.Object.Description,
                       item.Object.Image,
                       item.Object.Category,
                       item.Object.Rentee,
```

```
                item.Object.Owner,
                item.Key)).ToList();

            return list;
        }

        public async Task EditUser(string id, string email, string name, string surname,
string phone)
        {
            await fclient
                    .Child("users")
                    .Child(id)
                    .PutAsync(new UserInfo(name, surname, phone, email, null));
        }

        public async Task<List<UserInfo>> GetUserInfo()
        {
            var users = (await fclient
                .Child("users")
                .OnceAsync<UserInfo>())
                .Select(item =>
                   new UserInfo(
                       item.Object.Name,
                       item.Object.Surname,
                       item.Object.Phone,
                       item.Object.Email,
                       item.Key
                       )).ToList();

            return users;
        }

        public async Task<List<Request>> GetRequests()
        {
            var requests = (await fclient
                .Child("requests")
                .OnceAsync<Request>())
                .Select(item =>
                   new Request(
                       item.Object.ItemId,
                       item.Object.Rentee,
                       item.Key
                       )).ToList();

            return requests;
        }

        public async Task RequestForRenting(string ItemId, string Rentee)
        {
            await fclient
                .Child("requests")
                .PostAsync(new Request(ItemId, Rentee));
        }
```

```
        public async Task RentItem(string itemId, string Rentee,
ObservableCollection<Request> rqst, DisplayItem itm)
        {
            foreach (var request in rqst) {

                if (request.ItemId == itm.Id) {
                await fclient
                    .Child("requests")
                    .Child(request.Id)
                    .DeleteAsync();
                }
            }
            await fclient
               .Child("rent")
               .Child(itm.Id)
               .PutAsync(new RentItem(itm.Name, itm.Price, itm.Adress, itm.Town,
itm.Postal, itm.Description, itm.Image, itm.Category, Rentee, itm.Owner));
            MainPage.RefreshList();
        }

        public async Task RemoveRentee(DisplayItem itm)
        {
            await fclient
               .Child("rent")
               .Child(itm.Id)
               .PutAsync(new RentItem(itm.Name, itm.Price, itm.Adress, itm.Town,
itm.Postal, itm.Description, itm.Image, itm.Category, null, itm.Owner));
            MainPage.RefreshList();
        }

        public async Task ChangeRentItem(string name, int price, string adress, string town,
int postal, string description, int category, string image, string owner, string id)
        {
            await fclient
                .Child("rent")
                .Child(id)
                .PutAsync(new RentItem(name, price, adress, town, postal, description, image,
category, null, owner));
        }

        public async Task RemoveRentItem(DisplayItem itm)
        {
            await fclient
                    .Child("rent")
                    .Child(itm.Id)
                    .DeleteAsync();
            MainPage.RefreshList();
        }
    }
}
```

DisplayItem.cs

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

namespace RentApp.Model
{
    public class DisplayItem : RentItem
    {
        public ImageSource ImgSrc { get; set; }
        public UserInfo OwnerInfo { get; set; }
        public UserInfo RenteeInfo { get; set; }
        public Color Status { get; set; }
        public DisplayItem (RentItem Item, MemoryStream imgStream, UserInfo ownerInfo,
UserInfo RenteeInfo, Color status)
        {
            this.Name = Item.Name;
            this.Price = Item.Price;
            this.Adress = Item.Adress;
            this.Town = Item.Town;
            this.Id = Item.Id;
            this.Image = Item.Image;
            this.Postal = Item.Postal;
            this.Category = Item.Category;
            this.Description = Item.Description;
            this.Owner = Item.Owner;
            this.OwnerInfo = ownerInfo;
            this.RenteeInfo = RenteeInfo;
            this.Status = status;
            ImgSrc = ImageSource.FromStream(() => imgStream);
        }
        public DisplayItem(RentItem Item, MemoryStream imgStream)
        {
            this.Name = Item.Name;
            this.Price = Item.Price;
            this.Adress = Item.Adress;
            this.Town = Item.Town;
            this.Id = Item.Id;
            this.Image = Item.Image;
            this.Postal = Item.Postal;
            this.Description = Item.Description;
            this.Owner = Item.Owner;
            ImgSrc = ImageSource.FromStream(() => imgStream);
        }

        public DisplayItem()
        {
        }
    }
}
```

RentItem.cs

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

namespace RentApp.Model
```

```
{
    public class RentItem
    {
        public string Name { get; set; }
        public int Price { get; set; }
        public string Adress { get; set; }
        public string Town { get; set; }
        public int Postal { get; set; }
        public string Description { get; set; }
        public string Image { get; set; }
        public int Category { get; set; }
        public string Rentee { get; set; }
        public string Owner { get; set; }
        public string Id { get; set; }

        public RentItem(string name, int price, string adress, string town, int postal,
string description, string image, int category, string Rentee, string owner)
        {
            this.Name = name;
            this.Price = price;
            this.Adress = adress;
            this.Town = town;
            this.Postal = postal;
            this.Description = description;
            this.Image = image;
            this.Category = category;
            this.Rentee = Rentee;
            this.Owner = owner;
        }

        public RentItem(string name, int price, string adress, string town, int postal,
string description, string image, int category, string Rentee, string owner, string id)
        {
            this.Name = name;
            this.Price = price;
            this.Adress = adress;
            this.Town = town;
            this.Postal = postal;
            this.Description = description;
            this.Image = image;
            this.Category = category;
            this.Rentee = Rentee;
            this.Owner = owner;
            this.Id = id;
        }

        public RentItem() { }
    }
}
```

Request.cs

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

```
namespace RentApp.Model
{
    public class Request
    {

        public string Id { get; set; }
        public string ItemId { get; set; }
        public string Rentee { get; set; }

        public Request(string itemId, string Rentee, string id)
        {
            this.ItemId = itemId;
            this.Rentee = Rentee;
            this.Id = id;
        }

        public Request(string itemId, string rentee)
        {
            ItemId = itemId;
            Rentee = rentee;
        }

        public Request() { }
    }
}
```

UserInfo.cs

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

namespace RentApp.Model
{
    public class UserInfo
    {

        public UserInfo(string name, string surname, string phone, string email, string id)
        {
            Name = name;
            Surname = surname;
            Phone = phone;
            Email = email;
            Id = id;
        }

        public string Id { get; set; }
        public string Email { get; set; }
        public string Name { get; set; }
        public string Phone { get; set; }
        public string Surname { get; set; }
    }
}
```

AddItem.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="RentApp.AddItemPage">
    <ContentPage.Content>
        <ScrollView>
            <StackLayout Padding="16">
                <AbsoluteLayout VerticalOptions="CenterAndExpand" HeightRequest="256">
                    <Image
                        x:Name="addRentImage"
                        Aspect="AspectFill"
                        VerticalOptions="CenterAndExpand"
                        HorizontalOptions="CenterAndExpand"
                        HeightRequest="256"
                        AbsoluteLayout.LayoutFlags="All" AbsoluteLayout.LayoutBounds="1, 1, 1, 1"></Image>
                    <StackLayout AbsoluteLayout.LayoutFlags="XProportional,YProportional"
AbsoluteLayout.LayoutBounds="0.5, 1, 200, 200">
                        <Button BorderWidth="2" BorderColor="Black" BackgroundColor="White"
Clicked="PhotoPicker" Text="Add photo"></Button>
                        <Button BorderWidth="2" BorderColor="Black" BackgroundColor="White"
Clicked="TakePhoto" Text="Take photo"></Button>
                    </StackLayout>
                </AbsoluteLayout>
                <StackLayout Padding="0, 32">
                    <Picker Title="Category" x:Name="addCategory" HorizontalOptions="FillAndExpand"
SelectedIndexChanged="CategoryPicked">
                        <Picker.Items>
                            <x:String>Apartmant/House</x:String>
                            <x:String>Vehicle</x:String>
                            <x:String>Sport</x:String>
                            <x:String>Tehnology</x:String>
                            <x:String>Other</x:String>
                        </Picker.Items>
                    </Picker>
                    <Label Text="Name" />
                    <Entry Placeholder="Apartment, House..." x:Name="addName"></Entry>
                    <Label Text="Price" />
                    <Entry Keyboard="Numeric" Placeholder="Price per month in €"
x:Name="addPrice"></Entry>
                    <Label Text="Adress" />
                    <Entry x:Name="addAdress"></Entry>
                    <Label Text="Town" />
                    <Entry x:Name="addTown"></Entry>
                    <Label Text="Postal" />
                    <Entry Keyboard="Numeric" x:Name="addPostal" Placeholder="Postal"></Entry>
                    <Label Text="Desctription" />
                    <Entry x:Name="addDesctription"></Entry>
                    <Button x:Name="uploadBtn" Clicked="UploadData" Text="Add"></Button>
                    <Button x:Name="changeBtn" IsVisible="False" Text="Change"></Button>
                </StackLayout>
            </StackLayout>
        </ScrollView>
    </ContentPage.Content>
</ContentPage>
```

AddItem.xaml.cs

```
using Plugin.SecureStorage;
using RentApp.Model;
using System;
using System.IO;
using Windows.Media.Capture;
using Windows.Storage;
using Windows.Storage.Pickers;
using Windows.Storage.Streams;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace RentApp
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class AddItemPage : ContentPage
    {
        public string image = null;
        public string owner = CrossSecureStorage.Current.GetValue("user_id");
        public AddItemPage(RentItem item)
         {
             InitializeComponent ();

            if (item != null)
            {
                byte[] byteresponse = Convert.FromBase64String(item.Image);

                MemoryStream imgStream = new MemoryStream(byteresponse);
                image= item.Image;
                addRentImage.Source = ImageSource.FromStream(() => imgStream);
                addName.Text = item.Name;
                addPrice.Text = item.Price.ToString();
                addTown.Text = item.Town;
                addAdress.Text = item.Adress;
                addCategory.SelectedIndex = item.Category;
                addPostal.Text = item.Postal.ToString();
                addDesctription.Text = item.Description;
                uploadBtn.IsVisible = false;
                changeBtn.IsVisible = true;
                changeBtn.Clicked += delegate
                {
                    this.MakeChange(item as RentItem);
                };
            }
            NavigationPage.SetHasNavigationBar(this, false);
        }

        private async void PhotoPicker(object sender, EventArgs e)
        {
            FileOpenPicker picker = new FileOpenPicker
            {
                ViewMode = PickerViewMode.Thumbnail,
                SuggestedStartLocation = PickerLocationId.PicturesLibrary,
```

```
            };

            picker.FileTypeFilter.Add(".jpg");
            picker.FileTypeFilter.Add(".jpeg");
            picker.FileTypeFilter.Add(".png");

            StorageFile file = await picker.PickSingleFileAsync();

            if (file != null)
            {
                Stream stream = await file.OpenStreamForReadAsync();
                IRandomAccessStreamWithContentType iStream = await file.OpenReadAsync();
                Byte[] bytes = ReadFully(stream);
                String imageBase64 = Convert.ToBase64String(bytes);
                image = imageBase64;
                addRentImage.Source = ImageSource.FromStream(() =>
iStream.AsStreamForRead());
            }
        }

        private async void TakePhoto(object sender, EventArgs e)
        {
            var cameraComponent = new CameraCaptureUI();
            var file = await cameraComponent.CaptureFileAsync(CameraCaptureUIMode.Photo);

            if (file != null)
            {
                Byte[] bytes = File.ReadAllBytes(file.Path);
                String imageBase64 = Convert.ToBase64String(bytes);
                image = imageBase64;
                addRentImage.Source = file.Path;
            }
        }

        private async void UploadData(object sender, EventArgs e)
        {
            if (addName.Text != null && addPrice.Text != null && addTown.Text != null &&
addPostal.Text != null && addAdress.Text != null && addDesctription.Text != null && image !=
null) {
            var db = new DBFirebase();
            await db.NewRentItem(addName.Text, int.Parse(addPrice.Text), addAdress.Text,
addTown.Text, int.Parse(addPostal.Text), addDesctription.Text, addCategory.SelectedIndex,
image, owner);
            MainPage.RefreshList();
            await DisplayAlert("Item added", "Item successfully added to application!",
"Ok");
            await Navigation.PopAsync();
            } else {
                await DisplayAlert("Failed", "Please enter all information!", "Ok");
            }
        }

        private async void MakeChange(RentItem item)
        {
```

```
            if (addName.Text != null && addPrice.Text != null && addTown.Text != null &&
addPostal.Text != null && addAdress.Text != null && addDesctription.Text != null && image !=
null)
            {
                var db = new DBFirebase();
                await db.ChangeRentItem(addName.Text, int.Parse(addPrice.Text),
addAdress.Text, addTown.Text, int.Parse(addPostal.Text), addDesctription.Text,
addCategory.SelectedIndex, image, owner, item.Id);
                MainPage.RefreshList();
                await DisplayAlert("Item changed", "You successfully changed item
information!", "Ok");

this.Navigation.RemovePage(this.Navigation.NavigationStack[this.Navigation.NavigationS
tack.Count - 2]);
                await Navigation.PopAsync();
            }
            else
            {
                await DisplayAlert("Failed", "Please enter all information!", "Ok");
            }
        }

        public void CategoryPicked(object sender, EventArgs e)
        {
            var picker = sender as Picker;
            switch (picker.SelectedIndex)
            {
                case 0:
                    addPrice.Placeholder = "Price per month in €";
                    addName.Placeholder = "Apartment, House...";
                    break;
                default:
                    addName.Placeholder = "Name of the renting item";
                    addPrice.Placeholder = "Price per day in €";
                    break;
            }
        }

        public static byte[] ReadFully(Stream input)
        {
            byte[] buffer = new byte[16 * 1024];
            using (MemoryStream ms = new MemoryStream())
            {
                int read;
                while ((read = input.Read(buffer, 0, buffer.Length)) > 0)
                {
                    ms.Write(buffer, 0, read);
                }
                return ms.ToArray();
            }
        }

    }
}
```

App.xaml

```
<?xml version="1.0" encoding="utf-8" ?>
<Application xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="RentApp.App">
    <Application.Resources>

    </Application.Resources>
</Application>
```

App.xaml.cs

```
using Xamarin.Forms;
using Xamarin.Forms.Xaml;
using Plugin.SecureStorage;

[assembly: XamlCompilation(XamlCompilationOptions.Compile)]
namespace RentApp
{
    public partial class App : Application
    {
        public App ()
        {
           InitializeComponent();

           if (CrossSecureStorage.Current.HasKey("user_token"))
           {
               MainPage = new NavigationPage(new MainPage());
           } else
           {
               MainPage = new NavigationPage(new LoginPage());
           }
       }

       protected override void OnStart ()
        {
           // Handle when your app starts
        }

        protected override void OnSleep ()
        {
            // Handle when your app sleeps
        }

        protected override void OnResume ()
        {
            // Handle when your app resumes
        }
    }
}
```

EditUser.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="RentApp.EditUser">
    <ContentPage.Content>
        <ScrollView>
            <StackLayout Padding="16">
                <Label Margin="8" FontSize="Large" HorizontalTextAlignment="Center"
VerticalTextAlignment="Center" FlexLayout.AlignSelf="Center" Text="Edit profile"/>
                <Label Text="Name" />
            <Entry x:Name="userName" />
            <Label Text="Surname" />
            <Entry x:Name="userSurname"/>
            <Label Text="Email" />
            <Entry x:Name="userEmail"/>
            <Label Text="Phone" />
            <Entry x:Name="userPhone"/>
            <Button Clicked="EditClicked" Text="Change profile information"></Button>
            </StackLayout>
        </ScrollView>
    </ContentPage.Content>
</ContentPage>
```

EditUser.xaml.cs

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

using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace RentApp
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class EditUser : ContentPage
    {
        UserInfo usr;
         public EditUser (UserInfo userInfo)
         {
             InitializeComponent ();
            usr = userInfo;
            userEmail.Text = userInfo.Email;
            userName.Text = userInfo.Name;
            userSurname.Text = userInfo.Surname;
            userPhone.Text = userInfo.Phone;
            NavigationPage.SetHasNavigationBar(this, false);
        }

        private async void EditClicked(object sender, EventArgs e)
        {
            var db = new DBFirebase();
```

```
            if (userEmail.Text != "" && userName.Text != "" && userSurname.Text != "" &&
userPhone.Text != "") {
                await db.EditUser(usr.Id, userEmail.Text, userName.Text, userSurname.Text,
userPhone.Text);
                MainPage.RefreshList();
                await DisplayAlert("Success", "You have successfully change you profile
information.", "Ok");
                await Navigation.PopAsync();
            }
            else
                await DisplayAlert("Failed", "Please enter all information", "Ok");
        }
    }
}
```

ItemPage.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="RentApp.ItemPage">
    <ContentPage.Content>
        <ScrollView>
        <StackLayout>
            <AbsoluteLayout HeightRequest="256">
                <Image
                       Aspect="AspectFill" x:Name="itemImage"
                        HeightRequest="256"
                        AbsoluteLayout.LayoutFlags="All" AbsoluteLayout.LayoutBounds="1, 1,
1, 1"></Image>
                <StackLayout AbsoluteLayout.LayoutFlags="XProportional,YProportional"
AbsoluteLayout.LayoutBounds="0.5, 1, 200, 200" x:Name="editDeleteContainer">
                    <Button TextColor="Black" BackgroundColor="White"
Clicked="EditItemClicked" Text="Edit"></Button>
                    <Button TextColor="Black" BackgroundColor="White"
Clicked="DeleteItemClicked" Text="Delete"></Button>
                </StackLayout>
            </AbsoluteLayout>

            <StackLayout Padding="16">
                <Label FontSize="Large" FontAttributes="Bold" x:Name="itemName"/>

                <Label FontSize="Small" FontAttributes="Bold" Text="Price"/>
                <FlexLayout Direction="Row">
                    <Label x:Name="itemPrice"/>
                    <Label x:Name="price"/>
                </FlexLayout>

                <Label FontSize="Small" FontAttributes="Bold" Text="Adress"/>
                <FlexLayout Direction="Row">
                    <Label x:Name="itemAdress"/>
                        <Label Text=", " />
                        <Label Margin="6, 0, 0, 0"  x:Name="itemTown" />
                        <Label Text=", "/>
                        <Label Margin="6, 0, 0, 0"  x:Name="itemPostal"/>
```

```
            </FlexLayout>
                <Label FontSize="Small" FontAttributes="Bold" Text="Desctription"/>
            <Label x:Name="itemDescription"/>
        </StackLayout>
        <StackLayout x:Name="ownerContainer" Padding="16">
            <Label FontSize="Small" FontAttributes="Bold" Text="Owned by"/>
            <FlexLayout>
                    <Label x:Name="ownerName"/>
                    <Label Margin="6, 0, 0, 0" x:Name="ownerSurname"/>
            </FlexLayout>
            <FlexLayout>
                <Button Clicked="OSmsClicked" Text="SMS"></Button>
                <Button Clicked="OCallClicked" Text="Call"></Button>
                <Button Clicked="OEmailClicked" Text="Email"></Button>
            </FlexLayout>
            <Button Margin="16" Text="Request rent" x:Name="btnRequest"
Clicked="RequestForRent"/>
        </StackLayout>
        <StackLayout x:Name="RenteeContainer" Padding="16">
            <Label FontSize="Small" FontAttributes="Bold" Text="Rented by"/>
            <FlexLayout Direction="Row">
                    <Label x:Name="RenteeName"/>
                    <Label Margin="6, 0, 0, 0"  x:Name="RenteeSurname"/>
            </FlexLayout>
            <FlexLayout>
                <Button Clicked="RSmsClicked" Text="SMS"></Button>
                <Button Clicked="RCallClicked" Text="Call"></Button>
                <Button Clicked="REmailClicked" Text="Email"></Button>
            </FlexLayout>
            <Button  Text="Delete Rentee" x:Name="deleteRentee"
Clicked="OnRemoveRentee"/>

        </StackLayout>
            <Label FontSize="Small" FontAttributes="Bold" Margin="16" Text="Requests"
x:Name="reqst" IsVisible="False"/>
            <ListView ItemsSource="{Binding .}" x:Name="_rLst" HeightRequest="128">
            <ListView.ItemTemplate>
                <DataTemplate>
                    <ViewCell>
                        <ViewCell.ContextActions>
                            <MenuItem Clicked="OnRent" CommandParameter="{Binding .}"
Text="Rent" />
                        </ViewCell.ContextActions>
                        <ViewCell.View>
                            <StackLayout Orientation="Horizontal" Padding="16"
BackgroundColor="Transparent">
                                    <Label Text="{Binding Name}"></Label>
                                    <Label Text="{Binding Surname}"/>
                            </StackLayout>
                        </ViewCell.View>
                    </ViewCell>
                </DataTemplate>
            </ListView.ItemTemplate>
        </ListView>
```

```
        </StackLayout>
        </ScrollView>
    </ContentPage.Content>
</ContentPage>
```

ItemPage.xaml.cs

```
using Plugin.Messaging;
using Plugin.SecureStorage;
using RentApp.Model;
using System;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace RentApp
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class ItemPage : ContentPage
    {
        string ItemId;
        string Rentee = CrossSecureStorage.Current.GetValue("user_id");
        string Owner = CrossSecureStorage.Current.GetValue("user_id");
        static ObservableCollection<UserInfo> Rentees = new
ObservableCollection<UserInfo>();
        static ObservableCollection<Request> rqst = new ObservableCollection<Request>();
        static DisplayItem itm = new DisplayItem();


        public ItemPage(DisplayItem item, ObservableCollection<Request> requests,
ObservableCollection<UserInfo> users)
        {
            InitializeComponent();
            itemName.Text = item.Name;
            itemPrice.Text = item.Price.ToString();
            itemAdress.Text = item.Adress;
            itemTown.Text = item.Town;
            itemPostal.Text = item.Postal.ToString();
            itemDescription.Text = item.Description;
            editDeleteContainer.IsVisible = false;
            Rentees.Clear();
            rqst = requests;
            itm = item;

            if (item.Category == 0)
                    price.Text = " € / per month";
            else
                    price.Text = " € / per day";
```

```
if (requests != null)
{
    _rLst.IsVisible = true;
    _rLst.BindingContext = Rentees;
    ownerContainer.IsVisible = false;
    RenteeContainer.IsVisible = false;
    btnRequest.IsVisible = false;
    deleteRentee.IsVisible = false;
    editDeleteContainer.IsVisible = true;
    foreach (var request in requests)
    {
        if (item.Id == request.ItemId)
        {
            reqst.IsVisible = true;

            foreach (var user in users)
            {
                if (user.Id == request.Rentee) {
                    Rentees.Add(user);
                }
            }
        }
    }
}
if (item.RenteeInfo != null)
{
    editDeleteContainer.IsVisible = false;
    _rLst.IsVisible = false;
    reqst.IsVisible = false;
    RenteeContainer.IsVisible = true;
    ownerContainer.IsVisible = false;
    ownerContainer.HeightRequest = 0;
    btnRequest.IsVisible = false;
    deleteRentee.IsVisible = true;
    RenteeName.Text = item.RenteeInfo.Name;
    RenteeSurname.Text = item.RenteeInfo.Surname;
}
else if (item.OwnerInfo != null && requests != null)
{
    RenteeContainer.IsVisible = false;
    RenteeContainer.HeightRequest = 0;
    deleteRentee.IsVisible = false;
    ownerContainer.IsVisible = true;
    reqst.IsVisible = false;
    editDeleteContainer.IsVisible = false;
    _rLst.IsVisible = false;
    btnRequest.IsVisible = false;
    ownerName.Text = item.OwnerInfo.Name;
    ownerSurname.Text = item.OwnerInfo.Surname;
}
else if (requests == null)
{
    RenteeContainer.IsVisible = false;
```

```
                RenteeContainer.HeightRequest = 0;
                btnRequest.IsVisible = true;
                deleteRentee.IsVisible = false;
                ownerContainer.IsVisible = true;
                editDeleteContainer.IsVisible = false;
                _rLst.IsVisible = false;
                ownerName.Text = item.OwnerInfo.Name;
                ownerSurname.Text = item.OwnerInfo.Surname;
            }
            ItemId = item.Id;
            byte[] byteresponse = Convert.FromBase64String(item.Image);

            MemoryStream imgStream = new MemoryStream(byteresponse);

            itemImage.Source = ImageSource.FromStream(() => imgStream);

            NavigationPage.SetHasNavigationBar(this, false);
        }

        private async void RequestForRent(object sender, EventArgs e)
        {
            var db = new DBFirebase();
            bool requestMade = false;
            var requestList = await db.GetRequests();
            await Task.Run(() =>
            {
                foreach (var request in requestList)
                {
                    if (request.Rentee == Rentee && request.ItemId == itm.Id)
                    {
                        requestMade = true;
                    }
                }
            });

            Task.WaitAll();
            if (!requestMade)
            {
                await db.RequestForRenting(ItemId, Rentee);
                await DisplayAlert("Success", "You successfuly sent rent request to " +
itm.OwnerInfo.Name + " " + itm.OwnerInfo.Surname , "Ok");
            } else
            {
                await DisplayAlert("Request", "You have already made a rent reqest for " +
itm.Name, "Ok");
            }
        }
        private async void OnRent(object sender, EventArgs e)
        {
            var Rentee = (sender as MenuItem).BindingContext as UserInfo;

            var db = new DBFirebase();
            await db.RentItem(ItemId, Rentee.Id, rqst, itm);
```

```
            await DisplayAlert("Success", "You have successfully rented " + Rentee.Name +
" " + Rentee.Surname + " " + itm.Name + " - " + itm.Adress + ", " + itm.Town + "." , "Ok");
            await Navigation.PopAsync();

        }
        private async void OnRemoveRentee(object sender, EventArgs e)
        {
            var db = new DBFirebase();
            await db.RemoveRentee(itm);
            await DisplayAlert("Success", "You have successfully removed rentee!", "Ok");
            await Navigation.PopAsync();
        }
        private async void DeleteItemClicked(object sender, EventArgs e)
        {
            var db = new DBFirebase();
            await db.RemoveRentItem(itm);
            await DisplayAlert("Success", "You have successfully removed item!", "Ok");
            await Navigation.PopAsync();
        }
        private void OSmsClicked(object sender, EventArgs e)
        {
            var smsMessenger = CrossMessaging.Current.SmsMessenger;
            if (smsMessenger.CanSendSms)
                smsMessenger.SendSms(itm.OwnerInfo.Phone, "");
        }
        private void OCallClicked(object sender, EventArgs e)
        {
            var phoneDialer = CrossMessaging.Current.PhoneDialer;
            if (phoneDialer.CanMakePhoneCall)
                phoneDialer.MakePhoneCall(itm.OwnerInfo.Phone);
        }
        private void OEmailClicked(object sender, EventArgs e)
        {
            var emailMessenger = CrossMessaging.Current.EmailMessenger;
            if (emailMessenger.CanSendEmail)
            {
                emailMessenger.SendEmail(itm.OwnerInfo.Email, itm.Name + "-" + itm.Town +
", " + itm.Adress, "");
            }
        }

        private void RSmsClicked(object sender, EventArgs e)
        {
            var smsMessenger = CrossMessaging.Current.SmsMessenger;
            if (smsMessenger.CanSendSms)
                smsMessenger.SendSms(itm.RenteeInfo.Phone, "");
        }
        private void RCallClicked(object sender, EventArgs e)
        {
            var phoneDialer = CrossMessaging.Current.PhoneDialer;
            if (phoneDialer.CanMakePhoneCall)
                phoneDialer.MakePhoneCall(itm.RenteeInfo.Phone);
        }
        private void REmailClicked(object sender, EventArgs e)
```

```
        {
            var emailMessenger = CrossMessaging.Current.EmailMessenger;
            if (emailMessenger.CanSendEmail)
            {
                emailMessenger.SendEmail(itm.RenteeInfo.Email, itm.Name + "-" + itm.Town +
", " + itm.Adress, "");
            }
        }

        private async void EditItemClicked(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new AddItemPage(itm as RentItem));
        }
    }
}
```

ItemsPage.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:local="clr-namespace:RentApp.Model"
                x:Class="RentApp.ItemsPage">
    <ContentPage.Content>
        <StackLayout>
            <Label Margin="8" FontSize="Large" HorizontalTextAlignment="Center"
VerticalTextAlignment="Center" FlexLayout.AlignSelf="Center" x:Name="categoryTitle"/>
            <FlexLayout FlexLayout.Grow="1" AlignItems="Center"
JustifyContent="SpaceAround" x:Name="noItems" IsVisible="False">
                <Image Source="sad.png" WidthRequest="128" HeightRequest="128"></Image>
                <Label Text="No items currently available for rent!"></Label>
            </FlexLayout>
            <ListView ItemsSource="{Binding .}" x:Name="_iLst" ItemTapped="OnItemClick">
                <ListView.ItemTemplate>
                    <DataTemplate>
                        <ViewCell>
                            <ViewCell.View>
                                <FlexLayout Margin="0, 8" HeightRequest="86"
AlignItems="Center" Direction="Row" FlexLayout.Grow="1">
                                    <FlexLayout FlexLayout.Grow="1">
                                        <Image Aspect="AspectFill" Source="{Binding
ImgSrc}"></Image>
                                    </FlexLayout>
                                    <FlexLayout Direction="Column" Padding="16, 0"
FlexLayout.Grow="3" AlignItems="Center">
                                        <FlexLayout FlexLayout.Grow="1" AlignItems="Center"
Direction="Row">
                                            <Label FontAttributes="Bold" Text="{Binding
Name}"/>
                                            <Label FontAttributes="Bold" Text=" - "/>
                                            <Label FontAttributes="Bold" Text="{Binding
Adress}" LineBreakMode="TailTruncation"/>
                                            <Label FontAttributes="Bold" Text=", "/>
                                            <Label FontAttributes="Bold" Text="{Binding
Town}" LineBreakMode="TailTruncation"/>
```

```
                                    </FlexLayout>
                                    <FlexLayout FlexLayout.Grow="1" AlignItems="Center"
Direction="Row">
                                        <Label FontSize="Micro" Text="Price :  "/>
                                        <Label FontSize="Micro" Text="{Binding Price}"/>
                                        <Label FontSize="Micro" Text=" €"/>
                                    </FlexLayout>
                                </FlexLayout>
                            </FlexLayout>
                        </ViewCell.View>
                    </ViewCell>
                </DataTemplate>
            </ListView.ItemTemplate>
        </ListView>
    </StackLayout>
</ContentPage.Content>
</ContentPage>
```

ItemsPage.xaml.cs

```
using Plugin.SecureStorage;
using RentApp.Model;
using System;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace RentApp
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class ItemsPage : ContentPage
    {

        ObservableCollection<DisplayItem> itemsList = new
ObservableCollection<DisplayItem>();
        string Owner = CrossSecureStorage.Current.GetValue("user_id");
        public ItemsPage(int category, ObservableCollection<RentItem> list,
ObservableCollection<UserInfo> users)
        {
            InitializeComponent();
            _iLst.BindingContext = itemsList;
            this.SetItemsList(category, list, users);
            switch (category)
            {
                case 1:
                    categoryTitle.Text = "Vehicle";
                    break;
                case 2:
                    categoryTitle.Text = "Sport";
```

```
                break;
            case 3:
                categoryTitle.Text = "Tehnology";
                break;
            case 4:
                categoryTitle.Text = "Other";
                break;
            default:
                categoryTitle.Text = "Apartment/House";
                break;
        }
        NavigationPage.SetHasNavigationBar(this, false);
    }

    private void SetItemsList(int category, ObservableCollection<RentItem> list,
ObservableCollection<UserInfo> users)
    {

        if (itemsList != null)
        {
            foreach (var item in list)
            {

                if (item.Category == category && item.Owner != Owner && item.Rentee ==
null)
                {

                    byte[] byteresponse = Convert.FromBase64String(item.Image);

                    MemoryStream imgStream = new MemoryStream(byteresponse);

                    foreach (var user in users)
                    {
                        if(item.Owner == user.Id)
                        {
                            itemsList.Add(new DisplayItem(item, imgStream, user, null,
Color.White));
                        }
                    }


                }
            }
        }
        if (itemsList.Count == 0)
        {
            noItems.IsVisible = true;
        }
    }

    private async void OnItemClick(object sender, ItemTappedEventArgs e)
    {
        var itemClicked = e.Item as DisplayItem;
        await Navigation.PushAsync(new ItemPage(itemClicked, null, null));
```

```
        }

    }
}
```

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"
             x:Class="RentApp.LoginPage"
             Title="RentApp">
    <StackLayout Margin="32">
        <Label Margin="8" FontSize="Large" TextColor="#03abff"
HorizontalTextAlignment="Center" VerticalTextAlignment="Center"
FlexLayout.AlignSelf="Center" Text="Rent App"/>
        <StackLayout Padding="0, 32">
        <Entry
                Placeholder="Email"
                x:Name="email" />
        <Entry
                IsPassword="true"
                Placeholder="Password"
                x:Name="password" />
            <Button BorderWidth="2" BorderColor="#03abff" BackgroundColor="White"
Text="Login" Clicked="LoginClicked"></Button>
        </StackLayout>
        <BoxView
    HorizontalOptions="Fill"
    HeightRequest="1"
    Color="#03abff"/>
        <StackLayout Padding="0, 32">
            <Label Text="Don't have an account?"/>
            <Button BorderWidth="2" BorderColor="Black" BackgroundColor="White"
Text="Create account" Clicked="CreateClicked"></Button>
        </StackLayout>
    </StackLayout>
</ContentPage>
```

LoginPage.xaml.cs

```
using System;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;
using Firebase.Auth;
using Plugin.SecureStorage;
using Newtonsoft.Json.Linq;

namespace RentApp
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class LoginPage : ContentPage
    {
        public LoginPage()
        {
            InitializeComponent();
```

```
            NavigationPage.SetHasNavigationBar(this, false);
        }

        public class FirebaseErrorType
        {
            public FirebaseErrorType(string json)
            {
                JObject jObject = JObject.Parse(json);
                JToken jError = jObject["error"];
                Message = (string)jError["message"];
            }

            public string Message { get; set; }
        }

        private async void LoginClicked(object sender, EventArgs e)
        {
            var email_text = email.Text;
            var password_text = password.Text;

            var firebaseApiKey = "AIzaSyDJzuwFs4sSleR3N-DbnKk2snFCrBfLyM4";
            var auth = new FirebaseAuthProvider(new FirebaseConfig(firebaseApiKey));
            try
            {
                var token = await auth.SignInWithEmailAndPasswordAsync(email_text,
password_text);
                CrossSecureStorage.Current.SetValue("user_token", token.FirebaseToken);
                CrossSecureStorage.Current.SetValue("user_id", token.User.LocalId);
                await Navigation.PushModalAsync(new NavigationPage(new MainPage()));
            }
            catch (FirebaseAuthException error)
            {
                FirebaseErrorType errorType = new FirebaseErrorType(error.ResponseData);
                switch (errorType.Message)
                {
                    case "INVALID_EMAIL":
                        {
                            await DisplayAlert("Login failed", "You have entered invalid
email, please try again.", "OK");
                            password.Text = "";
                            break;
                        }
                    case "EMAIL_NOT_FOUND":
                        {
                            await DisplayAlert("Login failed", "There is no user registered
with this email.", "OK");
                            password.Text = "";
                            break;
                        }
                    case "MISSING_PASSWORD":
                        {
                            await DisplayAlert("Login failed", "You forgot to enter
password.", "OK");
                            break;
```

```
                    }
                case "INVALID_PASSWORD":
                    {
                        await DisplayAlert("Login failed", "You enter invalid password.
Please enter password again.", "OK");
                        password.Text = "";
                        break;
                    }
                default:
                    {
                        await DisplayAlert("Login failed", "There was an error. Please
check you internet connection.", "OK");
                        email.Text = "";
                        password.Text = "";
                        break;
                    }
            }
        }
    }

    private async void CreateClicked(object sender, EventArgs e)
    {
        await Navigation.PushAsync(new RegistrationPage());
    }
}
}
```

MainPage.xaml

```
<?xml version="1.0" encoding="utf-8" ?>
<TabbedPage xmlns="http://xamarin.com/schemas/2014/forms"
            xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
            x:Class="RentApp.MainPage">

    <!--Pages can be added as references or inline-->
    <ContentPage Title="Home">
        <ContentPage.Content>
            <FlexLayout JustifyContent="Center" Padding="0, 0, 0, 12">
                <StackLayout x:Name="loaderContainer" IsVisible="True" FlexLayout.AlignSelf="Center">
                <ActivityIndicator Color="Black"
                    x:Name="loader"
                    IsRunning="True" VerticalOptions="Center" HorizontalOptions="Center"
WidthRequest="50" HeightRequest="50" />
            </StackLayout>
                <FlexLayout FlexLayout.Grow="1" Direction="Column" x:Name="contentContainer"
IsVisible="False">
                    <Label Margin="8" FontSize="Large" FlexLayout.AlignSelf="Center" Text="I'm looking for"/>

                    <FlexLayout FlexLayout.Grow="1" JustifyContent="SpaceBetween" Direction="Row">
                        <Grid FlexLayout.Grow="1" Padding="4">
                            <Image Aspect="Fill" Source="h_a.png"/>
                            <Button BorderWidth="0" FontSize="Medium" TextColor="#1c0a00"
BackgroundColor="#B303abff" Text="Apartmant/House" Focused="BtnPressed" Unfocused="UnF"
Clicked="ApartmantHouseClicked"/>
```

```
                    </Grid>
                    <Grid FlexLayout.Grow="1" Padding="4">
                        <Image Aspect="Fill" Source="car.jpg"/>
                        <Button BorderWidth="0" FontSize="Medium" TextColor="#1c0a00"
BackgroundColor="#B303abff" Text="Vehicle" Focused="BtnPressed" Unfocused="UnF"
Clicked="VehicleClicked"></Button>
                    </Grid>
                </FlexLayout>

                <FlexLayout FlexLayout.Grow="1" JustifyContent="SpaceBetween" Direction="Row">
                    <Grid FlexLayout.Grow="1" Padding="4">
                        <Image Aspect="Fill" Source="sport.jpg"/>
                        <Button BorderWidth="0" FontSize="Medium" TextColor="#1c0a00"
BackgroundColor="#B303abff" Text="Sport" Focused="BtnPressed" Unfocused="UnF"
Clicked="SportClicked"></Button>
                    </Grid>
                    <Grid FlexLayout.Grow="1" Padding="4">
                        <Image Aspect="Fill" Source="tehnology.jpg"/>
                        <Button BorderWidth="0" FontSize="Medium" TextColor="#1c0a00"
BackgroundColor="#B303abff" Text="Tehnology" Focused="BtnPressed" Unfocused="UnF"
Clicked="TehnologyClicked"></Button>
                    </Grid>
                </FlexLayout>

                <FlexLayout FlexLayout.Grow="1">
                    <Grid FlexLayout.Grow="1" Padding="4">
                        <Image Aspect="Fill" Source="other.jpg"/>
                        <Button BorderWidth="0" FontSize="Medium" TextColor="#1c0a00"
BackgroundColor="#B303abff" Text="Other" Focused="BtnPressed" Unfocused="UnF" Clicked="OtherClicked"/>
                    </Grid>
                </FlexLayout>

            </FlexLayout>
        </FlexLayout>
    </ContentPage.Content>
</ContentPage>


<ContentPage Title="My renting items" x:Name="myItems">
    <ContentPage.Content>
        <StackLayout Padding="0, 16">
            <Label Margin="8" FontSize="Large" HorizontalTextAlignment="Center"
VerticalTextAlignment="Center" FlexLayout.AlignSelf="Center" Text="My items for rent"/>
            <FlexLayout x:Name="noItems" IsVisible="False" FlexLayout.Grow="1" AlignItems="Center"
Direction="Row">
                    <Image WidthRequest="128" HeightRequest="128" Source="sad.png"></Image>
                <StackLayout FlexLayout.Grow="1">
                    <Label Text="You have no items for renting."></Label>
                    <Button x:Name="noItemBtn" Text="Add an item for renting"
Clicked="AddItemClicked"></Button>
                </StackLayout>
            </FlexLayout>
```

```
                <ListView ItemsSource="{Binding .}" x:Name="_lst" ItemTapped="OnMyItemClick">
                    <ListView.ItemTemplate>
                        <DataTemplate>
                            <ViewCell>
                                <ViewCell.View>
                                    <FlexLayout Margin="0, 8" AlignItems="Center" HeightRequest="86"
Direction="Row" FlexLayout.Grow="1">

                                        <FlexLayout FlexLayout.Grow="1" AlignItems="End">
                                            <Image Aspect="AspectFill" Source="{Binding
ImgSrc}"></Image>
                                        </FlexLayout>

                                        <FlexLayout Direction="Column" Padding="16, 0"
FlexLayout.Grow="3" AlignItems="Center">
                                            <FlexLayout FlexLayout.Grow="1" AlignItems="Center"
Direction="Row">
                                                <Label FontAttributes="Bold" Text="{Binding Name}"/>
                                            </FlexLayout>
                                            <FlexLayout FlexLayout.Grow="1" AlignItems="Center"
Direction="Row">
                                                <Label Text="{Binding Adress}"/>
                                                <Label Text=", "/>
                                                <Label Text="{Binding Town}"/>
                                            </FlexLayout>
                                        </FlexLayout>

                                        <FlexLayout FlexLayout.Grow="1" AlignItems="Center"
JustifyContent="Center">
                                            <BoxView    BackgroundColor="{Binding Status}"
WidthRequest="32" HeightRequest="32"></BoxView>
                                        </FlexLayout>

                                    </FlexLayout>
                                </ViewCell.View>
                            </ViewCell>
                        </DataTemplate>
                    </ListView.ItemTemplate>
                </ListView>
                <Button Text="Add new item" Clicked="AddItemClicked"></Button>
            </StackLayout>
        </ContentPage.Content>
    </ContentPage>

    <ContentPage Title="Profile">
        <ContentPage.Content>
            <ScrollView>
            <StackLayout>
                <Label Margin="8" FontSize="Large" HorizontalTextAlignment="Center"
VerticalTextAlignment="Center" FlexLayout.AlignSelf="Center" Text="My profile"/>
                <FlexLayout>
                    <Button Clicked="OnLogoutClick" Text="Logout"></Button>
```

```
                    <Button Clicked="OnEditProfileClick" Text="Edit profile"></Button>
                </FlexLayout>
                <Label Text="Name" Margin="8" FontSize="Small" FontAttributes="Bold"/>
                <StackLayout Orientation="Horizontal">
                <Label Margin="8" x:Name="userName"></Label>
                <Label Margin="8" x:Name="userSurname"></Label>
                </StackLayout>
                <Label Text="Email : " Margin="8" FontSize="Small" FontAttributes="Bold"/>
                <Label Margin="8" x:Name="userEmail"></Label>
                    <Label Text="Phone : " Margin="8" FontSize="Small" FontAttributes="Bold"/>
                <Label Margin="8" x:Name="userPhone"></Label>
                    <Label Text="My rented items" Margin="8" FontSize="Small" FontAttributes="Bold"/>
                    <FlexLayout x:Name="noRentedItems" IsVisible="False" FlexLayout.Grow="1"
AlignItems="Center" Direction="Row">
                        <Image WidthRequest="64" HeightRequest="64" Source="sad.png"></Image>
                        <Label Text="You have no rented items."></Label>
                    </FlexLayout>
                    <ListView ItemsSource="{Binding .}" x:Name="_mRntLst" ItemTapped="OnMyItemClick"
HeightRequest="256">
                    <ListView.ItemTemplate>
                        <DataTemplate>
                            <ViewCell>
                                <ViewCell.View>
                                        <FlexLayout Margin="0, 8" HeightRequest="86"
AlignItems="Center" Direction="Row" FlexLayout.Grow="1">
                                            <FlexLayout FlexLayout.Grow="1">
                                                <Image Aspect="AspectFill" Source="{Binding
ImgSrc}"></Image>
                                            </FlexLayout>
                                            <FlexLayout Direction="Column" Padding="16, 0"
FlexLayout.Grow="3" AlignItems="Center">
                                                <FlexLayout FlexLayout.Grow="1" AlignItems="Center"
Direction="Row">
                                                    <Label FontAttributes="Bold" Text="{Binding
Name}"/>
                                                    <Label FontAttributes="Bold" Text=" - "/>
                                                    <Label FontAttributes="Bold" Text="{Binding
Adress}" LineBreakMode="TailTruncation"/>
                                                    <Label FontAttributes="Bold" Text=", "/>
                                                    <Label FontAttributes="Bold" Text="{Binding Town}"
LineBreakMode="TailTruncation"/>
                                                </FlexLayout>
                                                <FlexLayout FlexLayout.Grow="1" AlignItems="Center"
Direction="Row">
                                                    <Label FontSize="Micro" Text="Price :    "/>
                                                    <Label FontSize="Micro" Text="{Binding Price}"/>
                                                    <Label FontSize="Micro" Text=" €"/>
                                                </FlexLayout>
                                            </FlexLayout>
                                        </FlexLayout>
                                    </ViewCell.View>
                                </ViewCell>
```

```
                            </DataTemplate>
                        </ListView.ItemTemplate>
                    </ListView>
                </StackLayout>
                </ScrollView>
            </ContentPage.Content>
        </ContentPage>

</TabbedPage>
```

MainPage.xaml.cs

```
using Plugin.SecureStorage;
using RentApp.Model;
using System;
using System.Collections.ObjectModel;
using System.IO;
using System.Linq;
using System.Threading.Tasks;
using Windows.UI.Core;
using Windows.UI.Xaml;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace RentApp
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class MainPage : TabbedPage
    {
        static ObservableCollection<RentItem> list = new ObservableCollection<RentItem>();
        static ObservableCollection<UserInfo> users = new ObservableCollection<UserInfo>();
        static ObservableCollection<Request> requests = new ObservableCollection<Request>();
        static ObservableCollection<DisplayItem> myList = new ObservableCollection<DisplayItem>();
        static ObservableCollection<DisplayItem> myRent = new ObservableCollection<DisplayItem>();
        static VisualElement noItm;
        static VisualElement noRentItm;
        static Button noItmBtn;
        string currentUser = CrossSecureStorage.Current.GetValue("user_id");
        UserInfo currentUserInfo;
        public MainPage()
        {
            InitializeComponent();
            _lst.BindingContext = myList;
            noItm = noItems as FlexLayout;
            noItmBtn = noItemBtn;
            noRentItm = noRentedItems;
            _mRntLst.BindingContext = myRent;
            this.GetData();
            if (currentUserInfo != null)
            {
                userName.Text = currentUserInfo.Name;
                userSurname.Text = currentUserInfo.Surname;
                userEmail.Text = currentUserInfo.Email;
```

```
            userPhone.Text = currentUserInfo.Phone.ToString();
        }
        NavigationPage.SetHasNavigationBar(this, false);
}

public void BtnPressed(object sender, EventArgs e)
{
        var x = e as FocusEventArgs;
        x.VisualElement.BackgroundColor = Color.Transparent;
        x.VisualElement.Opacity = 0;
}

public void UnF(object sender, EventArgs e)
{
        var x = e as FocusEventArgs;
        x.VisualElement.BackgroundColor = Color.FromHex("#B303abff");
        x.VisualElement.Opacity = 100;
}

public async void GetData()
{
        contentContainer.IsVisible = false;
        loaderContainer.IsVisible = true;
        list.Clear();
        users.Clear();
        requests.Clear();
        myList.Clear();
        myRent.Clear();
        var db = new DBFirebase();
        var listAsync = await db.GetItems();
        var userListAsync = await db.GetUserInfo();
        var requestListAsync = await db.GetRequests();
        int listCounter = 0;
        int requestCounter = 0;
        int usersCounter = 0;
        if (list != null)
        {
            await Task.Run(() =>
            {
                foreach (var item in listAsync)
                {
                    list.Add(item);
                    listCounter++;
                    if (item.Owner == currentUser)
                    {
                        byte[] byteresponse = Convert.FromBase64String(item.Image);

                        MemoryStream imgStream = new MemoryStream(byteresponse);
                        if (item.Rentee != null)
                        {
                            foreach (var user in userListAsync)
                            {
```

```
                                        if (item.Rentee == user.Id)
                                        {
                                            myList.Add(new DisplayItem(item, imgStream, null, user,
Color.FromHex("#57ff03")));
                                        }
                                    }
                                }
                                else
                                {
                                    bool inRequest = false;
                                    foreach (var request in requestListAsync)
                                    {
                                        if (request.ItemId == item.Id)
                                        {
                                            inRequest = true;
                                        }
                                    }
                                    if (inRequest)
                                        myList.Add(new DisplayItem(item, imgStream, null, null,
Color.FromHex("#ffd503")));
                                    else
                                        myList.Add(new DisplayItem(item, imgStream, null, null, Color.White));
                                }
                            }
                            if (item.Rentee == currentUser)
                            {
                                byte[] byteresponse = Convert.FromBase64String(item.Image);
                                MemoryStream imgStream = new MemoryStream(byteresponse);
                                foreach (var user in userListAsync)
                                {
                                    if (item.Owner == user.Id)
                                    {
                                        myRent.Add(new DisplayItem(item, imgStream, user, null, Color.White));
                                    }
                                }
                            }
                        }
                        foreach (var user in userListAsync)
                        {
                            users.Add(user);
                            usersCounter++;
                            if (user.Id == currentUser)
                            {
                                currentUserInfo = user;
                                userName.Text = user.Name;
                                userSurname.Text = user.Surname;
                                userEmail.Text = user.Email;
                                userPhone.Text = user.Phone.ToString();
                            }
                        }
                        foreach (var request in requestListAsync)
                        {
```

```
                    requests.Add(request);
                    requestCounter++;
                }
            });
            if (listCounter == listAsync.Count() && usersCounter == userListAsync.Count() && requestCounter
== requestListAsync.Count())
            {
                loaderContainer.IsVisible = false;
                contentContainer.IsVisible = true;
            }

            if (myList.Count == 0)
                noItems.IsVisible = true;
            else
                noItems.IsVisible = false;

            if (myRent.Count == 0)
                noRentedItems.IsVisible = true;
            else
                noRentedItems.IsVisible = false;
        }
    }

    private async void ApartmantHouseClicked(object sender, EventArgs e)
    {
        await Navigation.PushAsync(new ItemsPage(0, list, users));
    }

    private async void VehicleClicked(object sender, EventArgs e)
    {
        await Navigation.PushAsync(new ItemsPage(1, list, users));
    }

    private async void SportClicked(object sender, EventArgs e)
    {
        await Navigation.PushAsync(new ItemsPage(2, list, users));
    }

    private async void TehnologyClicked(object sender, EventArgs e)
    {
        await Navigation.PushAsync(new ItemsPage(3, list, users));
    }

    private async void OtherClicked(object sender, EventArgs e)
    {
        await Navigation.PushAsync(new ItemsPage(4, list, users));
    }

    private async void AddItemClicked(object sender, EventArgs e)
    {
        await Navigation.PushAsync(new AddItemPage(null));
    }
```

```
private async void OnMyItemClick(object sender, ItemTappedEventArgs e)
{
	var itemClicked = e.Item as DisplayItem;
	await Navigation.PushAsync(new ItemPage(itemClicked, requests, users));
}

private async void OnEditProfileClick(object sender, EventArgs e)
{
	await Navigation.PushAsync(new EditUser(currentUserInfo));
}

private async void OnLogoutClick(object sender, EventArgs e)
{
	CrossSecureStorage.Current.DeleteKey("user_id");
	CrossSecureStorage.Current.DeleteKey("user_token");
	await Navigation.PushModalAsync(new NavigationPage(new LoginPage()));
}

public async static void RefreshList()
{
	list.Clear();
	users.Clear();
	requests.Clear();
	myList.Clear();
	myRent.Clear();
	var db = new DBFirebase();
	var listAsync = await db.GetItems();
	var userListAsync = await db.GetUserInfo();
	var requestListAsync = await db.GetRequests();
	if (list != null)
	{
		await Task.Run(() =>
		{
			foreach (var item in listAsync)
			{
				list.Add(item);
				if (item.Owner == CrossSecureStorage.Current.GetValue("user_id"))
				{
					byte[] byteresponse = Convert.FromBase64String(item.Image);

					MemoryStream imgStream = new MemoryStream(byteresponse);

					if (item.Rentee != null)
					{
						foreach (var user in userListAsync)
						{
							if (item.Rentee == user.Id)
							{
								myList.Add(new DisplayItem(item, imgStream, null, user,
Color.FromHex("#57ff03")));
							}
```

```
                    }
                }
                else
                {
                    bool inRequest = false;
                    foreach (var request in requestListAsync)
                    {
                        if (request.ItemId == item.Id)
                        {
                            inRequest = true;
                        }
                    }
                    if (inRequest)
                        myList.Add(new DisplayItem(item, imgStream, null, null,
Color.FromHex("#ffd503")));
                    else
                        myList.Add(new DisplayItem(item, imgStream, null, null, Color.White));
                }
            }
            if (item.Rentee == CrossSecureStorage.Current.GetValue("user_id"))
            {
                byte[] byteresponse = Convert.FromBase64String(item.Image);
                MemoryStream imgStream = new MemoryStream(byteresponse);
                foreach (var user in userListAsync)
                {
                    if (item.Owner == user.Id)
                    {
                        myRent.Add(new DisplayItem(item, imgStream, user, null, Color.White));
                    }
                }
            }
        }
        foreach (var user in userListAsync)
        {
            users.Add(user);
        }

        foreach (var request in requestListAsync)
        {
            requests.Add(request);
        }

Windows.ApplicationModel.Core.CoreApplication.MainView.CoreWindow.Dispatcher.RunAsync(CoreDispatcherPrior
ity.Normal, () =>
        {
            if (myList.Count == 0) {
                noItm.IsVisible = true;
                noItmBtn.Text = "Add an item for renting";
            }
            else
                noItm.IsVisible = false;
```

```
                        if (myRent.Count == 0)
                            noRentItm.IsVisible = true;
                        else
                            noRentItm.IsVisible = false;
                    });

                });
            }
        }
    }
}
```

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="RentApp.RegistrationPage"
             Title="Registration" NavigationPage.HasNavigationBar="False">
    <StackLayout HorizontalOptions="Center" VerticalOptions="Center">
        <Label Margin="8" FontSize="Large" TextColor="#03abff"
HorizontalTextAlignment="Center" VerticalTextAlignment="Center"
FlexLayout.AlignSelf="Center" Text="Registration"/>
        <Entry
            Placeholder="Email"
            x:Name="email" />
        <Entry
            Placeholder="Name"
            x:Name="name" />
        <Entry
            Placeholder="Surname"
            x:Name="surname" />
        <Entry
            Placeholder="Phone number"
            x:Name="phone" />
        <Entry
            IsPassword="true"
            Placeholder="Password"
            x:Name="password" />
        <Entry
            IsPassword="true"
            Placeholder="Confirm password"
            x:Name="passwordC" />
        <Button BorderWidth="2" BorderColor="#03abff" BackgroundColor="White"
Text="Register" Clicked="RegisterClicked"></Button>
    </StackLayout>
</ContentPage>
```

RegistrationPage.xaml.cs

```
using System;
using System.Dynamic;
using System.Linq;
using System.Threading;
using System.Threading.Tasks;
using System.Windows;
```

```
using Xamarin.Forms;
using Xamarin.Forms.Xaml;
using Firebase.Auth;
using Newtonsoft.Json.Linq;

using Firebase.Database;
using Firebase.Database.Query;
using RentApp.Model;

namespace RentApp
{
    public class FirebaseErrorType
    {
        public FirebaseErrorType(string json)
        {
            JObject jObject = JObject.Parse(json);
            JToken jError = jObject["error"];
            Message = (string)jError["message"];
        }

        public string Message { get; set; }
    }

    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class RegistrationPage : ContentPage
    {

        public RegistrationPage()
        {
            InitializeComponent();
        }

        private async void RegisterClicked(object sender, EventArgs e)
        {
            var email_text = email.Text;
            var password_text = password.Text;
            var name_text = name.Text;
            var surname_text = surname.Text;
            var phone_number = phone.Text;
            var password_c = passwordC.Text;

            var firebaseUrl = "https://rentapp-207912.firebaseio.com";

            var firebaseApiKey = "AIzaSyDJzuwFs4sSleR3N-DbnKk2snFCrBfLyM4";

            var auth = new FirebaseAuthProvider(new FirebaseConfig(firebaseApiKey));

            if (email_text != null && password_text != null && password_c != null &&
name_text != null && surname_text != null && phone_number != null)
            {
                if (password_c == password_text)
                {
                    try
```

```
                {
                    var user = await
auth.CreateUserWithEmailAndPasswordAsync(email_text, password_text);
                    var userId = user.User.LocalId;
                    var firebase = new FirebaseClient(firebaseUrl);
                    await firebase
                            .Child("users")
                            .Child(userId)
                            .PutAsync(new UserInfo(name_text, surname_text,
phone_number, email_text, null));
                    await DisplayAlert("Registration successful", "Thank you for
registering " + name_text + ". You can login now.", "OK");
                    await Navigation.PopAsync();
                }
                catch (FirebaseAuthException error)
                {
                    FirebaseErrorType errorType = new
FirebaseErrorType(error.ResponseData);
                    switch (errorType.Message)
                    {
                        case "INVALID_EMAIL":
                            {
                                await DisplayAlert("Registration failed", "You have
entered invalid email, please try again.", "OK");
                                email.Text = "";
                                break;
                            }
                        case "EMAIL_EXISTS":
                            {
                                await DisplayAlert("Registration failed", "Email already
in use, please try again.", "OK");
                                email.Text = "";
                                break;
                            }
                        case "MISSING_PASSWORD":
                            {
                                await DisplayAlert("Registration failed", "You forgot to
enter password.", "OK");
                                break;
                            }
                        case "WEAK_PASSWORD":
                            {
                                await DisplayAlert("Registration failed", "For security
reasons password must contain 6 or more characters.", "OK");
                                password.Text = "";
                                break;
                            }
                        default:
                            {
                                await DisplayAlert("Registration failed", "There was an
error. Please check you internet connection", "OK");
                                email.Text = "";
                                password.Text = "";
                                break;
```

```
                            }
                        }
                    }
                }
                else
                {
                    await DisplayAlert("Registration failed", "Passwords do not match!",
"OK");
                    password_c = "";
                    password_text = "";
                }
            }
            else
            {
                await DisplayAlert("Registration failed", "To complete registration provide
all required information!", "OK");
            }
        }

    }
}
```