Mobilna aplikacija za upravljanje vatrogasnim aparatima

Peršić, Matej

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Sveučilišni studij

MOBILNA APLIKACIJA ZA UPRAVLJANJE VATROGASNIM APARATIMA

Diplomski rad

Matej Peršić

Osijek, 2023.

SADRŽAJ

1. UVOD

Zakonodavac propisuje da svaka javna ustanova i industrijski prostor (te prodajni, trgovački, skladišni, uredski, smještajni, uslužni, ugostiteljski, kulturno – zabavni i obrtni prostori) mora imati određenu količinu vatrogasnih aparata kako bi se oni mogli upotrijebiti u slučaju požara. Vatrogasni aparati moraju biti redovito servisirani kako bi se njima moglo koristiti. Kao i sva oprema, mogu se pokvariti i imaju svoj rok trajanja. Od stajanja može im se spustiti tlak, otpustiti ventil i tada nisu pouzdani za gašenje požara. Upravljanje vatrogasnim aparatima je vrlo odgovoran zadatak jer svaki vatrogasni aparat mora biti ispravan. Za njihov pregled važno je tehničarima omogućiti što jednostavnije i intuitivnije sučelje, kako bi taj pregled mogli uspješno izvršiti.

Cilj ovog diplomskog rada je izraditi mobilnu aplikaciju za upravljanje vatrogasnim aparatima korištenjem tehnologije Xamarin.Forms. Aplikacija će tehničarima na njihovim mobilnim uređajima omogućiti preglede vatrogasnih aparata, izradu izvješća, izradu redovnog pregleda, pregled servisa za uređaje i jednostavan kontakt putem telefona ili elektroničke pošte. Svi korisnici moraju odraditi validaciju svojih podataka prije pristupa sustavu za upravljanje aparatima. Osnovna funkcija aplikacije je omogućiti tehničaru izvršavanje pregleda uređaja.

U drugom poglavlju su opisani izazovi koji će morati biti savladani prilikom izrade aplikacije. U trećem poglavlju će biti opisana arhitektura i tehnologija korišteni za izradu programskog rješenja. U četvrtom poglavlju će biti prikazan razvoj i tok aplikacije koji omogućuju tehničaru odrađivanje potrebnih zadataka. Zadnje poglavlje pojašnjava korištenje mobilne aplikacije uz prikaz pripadajućih sučelja.

2. IZAZOVI U IZRADI APLIKACIJE I STANJE U PODRUČJU

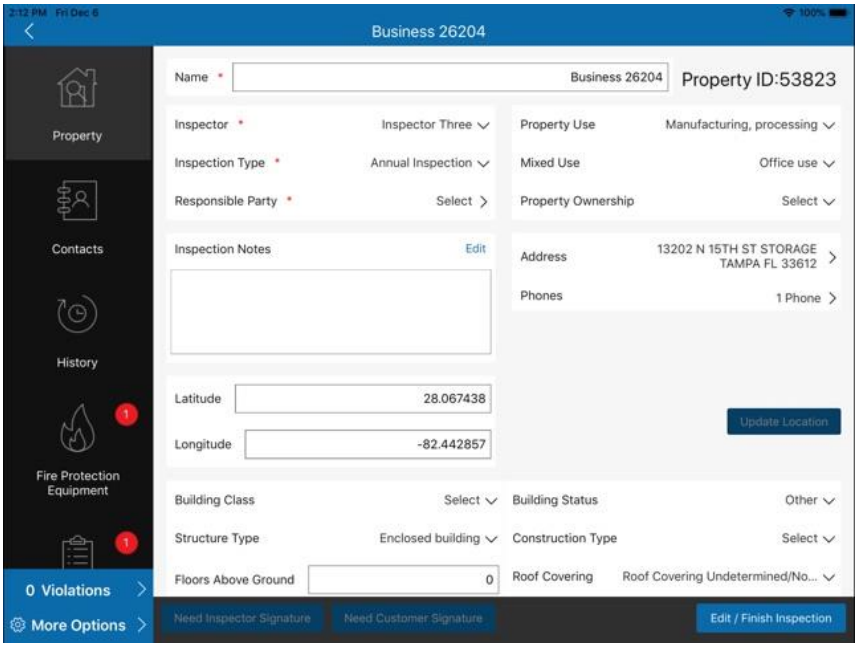
Izazovi prilikom izrade aplikacije su podijeljeni su u dvije velike cjeline: registracijski dio i glavni dio aplikacije. One obuhvaćaju sve funkcionalne mogućnosti aplikacije kao što su prijava, autentifikacija, pregled vatrogasnih aparata na određenoj lokaciji, pregled dostupnih servisa na Google kartama, izrada i pisanje izvješća. Između ostalog, tehničar može dobiti i notifikaciju za vatrogasni aparat kojega primjerice treba pregledati za idućih 30 dana i može izgenerirati PDF dokument koji će koristiti u daljnjem procesu kontrole uređaja.

2.1. Stanje u području aplikacija za upravljanje vatrogasnim aparatima

Postoje različita rješenja dostupna za upravljanje vatrogasnim aparatima, a ona mogu biti ili dio web stranice ili mobilna aplikacija. Ovisno o samoj problematici koju aplikacija treba rješavati, ima više različitih tipova aplikacija. Postoje aplikacije čiji je jedini zadatak omogućiti inspektorima vatrogasnih aparata sučelje za izvršavanje inspekcije i izradu izvješća za odrađeni posao, a postoje i aplikacije koje uz takvu funkcionalnost omogućuju lokalnim vatrogasnim postrojbama pregled problematičnih područja i terena na kojemu se događa požar.

2.1.1. Sustav Streamline Inspections

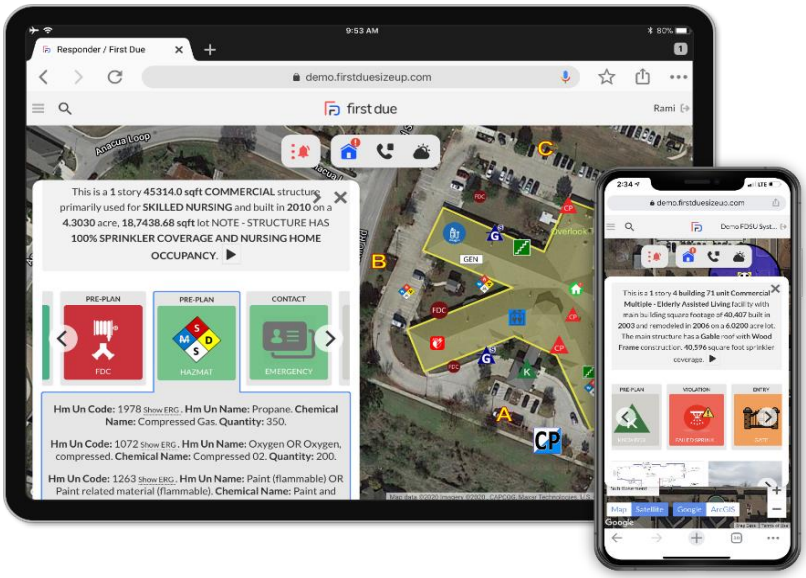
Streamline Inspections je mobilna i cloud-based platforma za inspekciju i dizajnirana je za državne i lokalne vlasti kao i poslovne subjekte koji obavljaju inspekcijske usluge. Dizajnirana je kako bi olakšala i učinila nadzor učinkovitijim. Platforma uključuje više programa za inspektore, između ostalog, omogućuje i inspekciju protupožarne zaštite. Dostupna je kao desktop aplikacija i kao mobilna aplikacija za iOS platformu. StreamLine Inspections sustav omogućuje sinkroniziranje posla na Windows uređajima i tabletima, što omogućuje korisnicima jednostavan nastavak posla na drugoj platformi. Uz pružanje funkcionalnosti poput redovnih pregleda za vatrogasne aparate, promjene lokacije preko geografske širine i geografske dužine, omogućuje korisnicima i prikaz lokacija na karti. Dodatno, sustav omogućava izračun procjene rizika imovine, upravljanje sustavom hidranta, edukaciju zajednice uz inspekciju sigurnosti požara u domu, postavljanje detektora dima i ostale preglede za osiguravanje kvalitete života i sigurnost u domu [1]. Korisničko sučelje StreamLine Inspections aplikacije se može vidjeti na slici Sl. 2.1.



Sl. 2.1. Streamline Inspections korisničko sučelje [2]

2.1.2. Sustav First Due Mobile

First Due je sustav u oblaku te kao takav dostupan putem web i mobilnih aplikacija koji omogućuje praćenje i obavještavanje o incidentima, rutu za hitne slučajeve, chat komunikaciju s vatrogasnim timom, praćenje incidenta i održavanje vatrogasnih aparata. Dodatno, First Due pruža usluge treniranja, planiranje događaja i aktivnosti, dijeljenja informacija i uređaja, povezivanja zajednice, prevencije požara, upravljanje hidrantima i usluge mobilnog operater preko mobilne aplikacije [3]. Korisničko sučelje First Due Mobile aplikacije može se vidjeti na slici Sl. 2.2.



Sl. 2.2. First Due Mobile aplikacija [4]

2.1.3. FireMate

FireMate je australska tvrtka specijalizirana za upravljanje uslugama održavanja sustava zaštite od požara. Njihov softver se koristi za upravljanje uslugama za održavanje sustava zaštite od požara u stotinama tvrtki diljem Australije. Navedeni softver transformirao je industriju zaštite od požara tako što ubrzava svakodnevne postupke terenskih tehničara te je automatiziran. Na taj način je FireMate aplikacija postala neizostavan alat zanata za tisuće stručnjaka za zaštitu od požara. FireMate je osvojio nekoliko nagrada za inovaciju. Kontinuirano rade na razvoju novih funkcija i poboljšanju svojih proizvoda. Ova aplikacija član je globalne grupe tehnoloških tvrtki Halma koje spašavaju živote i grade sigurniju, čišću i zdraviju budućnost. [5]. Korisničko sučelje aplikacije FireMate se može vidjeti na slici Sl. 2.3.

Sl. 2.3. FireMate aplikacija [6]

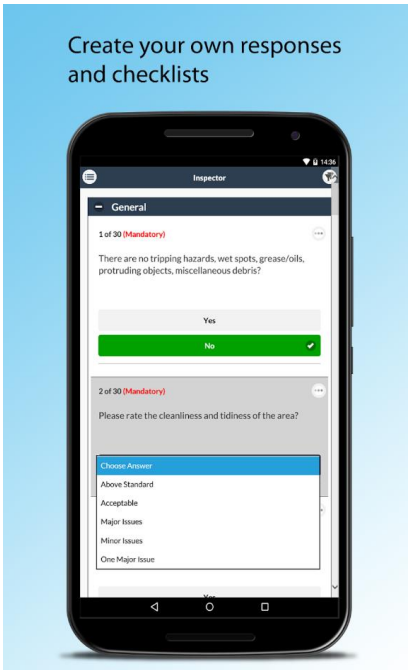
2.1.4. Fire Extinguisher Inspection

Fire Extinguisher Inspection aplikacije je aplikacija na Google Play trgovini koja se može koristiti za obavljanje inspekcija vatrogasnih aparata, kao i pregled njihovih lokacija i lokalnih smjernica o požarima. Za brzo i točno identificiranje stavke koja se pregledava, služe barkodovi i QR kodovi. Korisnici mogu sakupljati komentare, fotografije s napomenama, GPS koordinate i potpise, te stvarati vlastite obrasce. Nakon završetka pregleda rezultati obrazaca se prenose na web portal [7]. Korisničko sučelje Fire Extinguisher Inspection aplikacije može se vidjeti na slici Sl. 2.4.

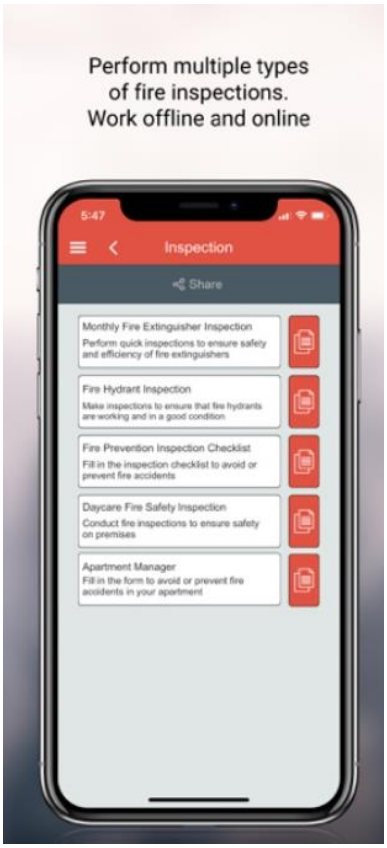
2.1.5. Fire Inspection App

Fire Inspection App je mobilna aplikacija dostupna na Apple trgovini i dizajnirana je kako bi pomogla tvrtkama da osiguraju svoje radno mjesto i osiguraju da objekti imaju sve potrebne opreme za gašenje požara, kao i inspekciju vatrogasnih aparatima. Aplikacija pokriva vatrogasne aparate, hidrante, prevenciju požara, dnevni vatrogasni pregled i ostalo. Aplikacija Fire Inspection

App dizajnirana je za inspektore vatrogasnih aparata i izvrstan je alat za brz pregled i lako prikupljanje potrebnih podataka [8] . Korisničko sučelje aplikacije Fire Inspection App se može vidjeti na slici Sl. 2.5.



Sl. 2.4. Fire Extinguisher Inspection aplikacija [9]



Sl. 2.5. Fire Inspection App aplikacija [10]

2.2. Izazovi kod upravljanja vatrogasnim aparatima

Iako već postoje sustavi za upravljanje vatrogasnim aparatima, s obzirom na brojne različite proizvođače i razlike među tržištima, potreban je velik broj sustava koji će raditi s njima. Svaka država ima svoje propise i regulacije oko vatrogasnih aparata, stoga tehničarima treba pružiti drugačija sučelja za rad s njima.

2.3. Stanje u području aplikacija za upravljanje vatrogasnim aparatima

Upravljanje vatrogasnim aparatima danas ne predstavlja problem obzirom na brojne provjerene servise koji se desetljećima bave izradom i servisiranjem vatrogasnih aparata. Pojavom interneta i pametnih uređaja, svoju tradiciju upravljanja vatrogasnim aparatima su modernizirali izradom brojnih različitih sustava za upravljanje vatrogasnih aparatima. Takvi sustavi omogućuju tehničarima brz pregled vatrogasnih uređaja na više različitih platformi te im tako olakšavaju odrađivanje potrebnih pregleda s manje papirologije. Neki sustavi za upravljanje vatrogasnim aparatima su povezani i s lokalnim vatrogasnim zajednicama te tako uz funkcionalnost pregleda omogućuju komunikaciju između vatrogasaca i dojavu požara, ali i dodatno olakšavaju problematiku gašenja. Pojava novih proizvođača vatrogasnih aparata i novih sustava za upravljanje vatrogasnim aparatima samo ojačava već brojni izbor gotovih rješenja za protupožarne sustave.

2.4. Idejno rješenje mobilne aplikacije za upravljanje vatrogasnim aparatima

Idejno rješenje mobilne aplikacije definira programsku podršku kojoj je zadaća omogućiti tehničarima pregled vatrogasnih aparata. U izvedbi rješenja važno je spomenuti višeslojnu arhitekturu koja omogućuje lakše održavanje i testiranje sustava. Takav sustav omogućava programeru lakše dodavanje novih funkcionalnosti i rješavanje postojećih problema. Cilj je zadovoljiti sve zahtjeve na sustav. Nakon izvedbe rješenja aplikacije, prikazane su korištene tehnologije i pojašnjeni su ključni dijelovi implementacije kod korištenja aplikacije. Nakon pregleda implementacije, prikazano je rješenje u načinu korištenja tehničara.

3. ARHITEKTURA I PREGLED KORIŠTENIH TEHNOLOGIJA

3.1. Tehnologija Xamarin

U svrhu izrade mobilne aplikacije pomoću koje se obavljaju akcije vezane uz nadgledanje vatrogasnih aparata korišten je Xamarin.Forms. To je više-platformski okvir otvorenog koda za izradu aplikacija predstavljan u svibnju 2014. godine. Tvrtku Xamarin su osnovali inženjeri koji su napravili okvir Mono za .NET sustave. Budući da su Xamarin alati više-platformski, moguće je dizajnirati aplikacije koje se mogu pokretati na uređajima s Androidom, iOS-om i UWP uređajima. Programski jezik koji se koristi sa Xamarinom je C# te omogućuje stvaranje brzih i učinkovitih aplikacija. [11]

Xamarin.Forms stranice, rasporedi i kontrole omogućuju izrađivanje i dizajniranje mobilnih aplikacija iz jednog API-ja koji ima brojne mogućnosti. Nasljeđivanje bilo koje kontrole i nadogradnja njenih ponašanja omogućuju izradu aplikacija savršeno vjernih izvornom dizajnu. [12] Xamarin.Forms će biti službeno podržan sve do svibnja 2024. godine, kada će ga potpuno zamijeniti .NET MAUI, u kojem će Xamarin.Forms biti spojen u .NET 6. Poput Xamarin.Formsa, .NET MAUI koristi C# i omogućuje stvaranje aplikacija za više različitih platformi iz jednog koda. Za razliku od Xamarina koji koristi primarno za razvoj mobilnih aplikacija, .NET MAUI uključuje i potporu za razvoj desktop aplikacija za macOS-a i Windows platforme. .NET MAUI ima moderniju arhitekturu od platformskog okvira Xamarin.Forms i novi *hot reload*, koji omogućuje programerima da istovremeno vide promjene u kodu bez ponovnog pokretanja aplikacije. [13]

3.2. NuGet paketi

NuGet je upravitelj paketa za .NET tehnologije. NuGet Gallery je centralni repozitorij za pakete koji koriste svi autori i korisnici paketa. NuGet paketi omogućuju jednostavno proširivanje određenih funkcionalnosti unutar C# aplikacija [14]. Unutar ovog projekta koristi se nekoliko takvih paketa. NuGet paketi koji su korišteni za ostvarivanje funkcionalnosti poput *popup-a*, notifikacija i generiranja PDF dokumenata biti će dodatno pojašnjeni.

3.2.1. Generiranje PDF-a

PdfSharp.Xamarin.Forms je Xamarin.Forms *library* za pretvaranje bilo kojeg Xamarin.Forms korisničkog sučelja u PDF dokument. To je NuGet paket objavljen u 2019. godini i ima 12 tisuća preuzimanja. Koristi PDFSharp koji je djelomičan port PDFSharpCore-a. Podržava platforme UWP, Android, iOS i WPF [15]. Primjeri generiranih PDF-ova se mogu vidjeti na slici Sl. 3.1.

Njegovo korištenje se ostvaruje u tri koraka:

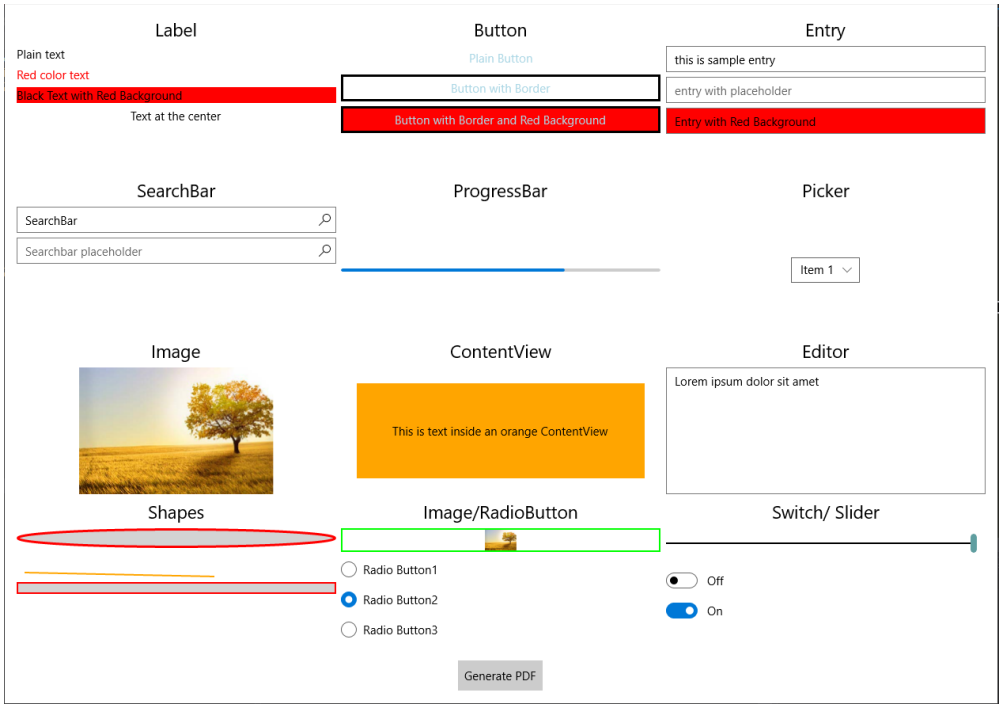
1. Init – za svaku platformu potrebno je posebno inicijalizirati NuGet
2. Generiranje – generiranje PDF dokumenta pomoću PDFManagera
3. Spremanje – spremanje izgeneriranog dokumenta na svakoj platformi pomoću metoda *DependencyService*

Funkcionalnosti:

- fontovi po želji
- renderiranje slika
- custom renderer
- veličina papira i postavljanje orijentacije

Ograničenja:

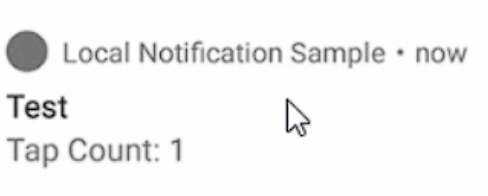
- podržava samo Jpeg format slike
- za korištenje komponente *ListView* potrebno je napisati vlastiti *renderer*
- podržava samo generiranje PDF dokumenata i ograničenu manipulaciju njima
- generiranje PDF dokumenta kroz PdfSharp.Xamarin.Forms paket može biti sporije od ostalih rješenja za generiranje PDF dokumenta
- ne postoji službena podrška Xamarin tima



Sl. 3.1. Primjer izgeneriranog PDF dokumenta iz aplikacije [16]

3.2.2. Slanje notifikacija

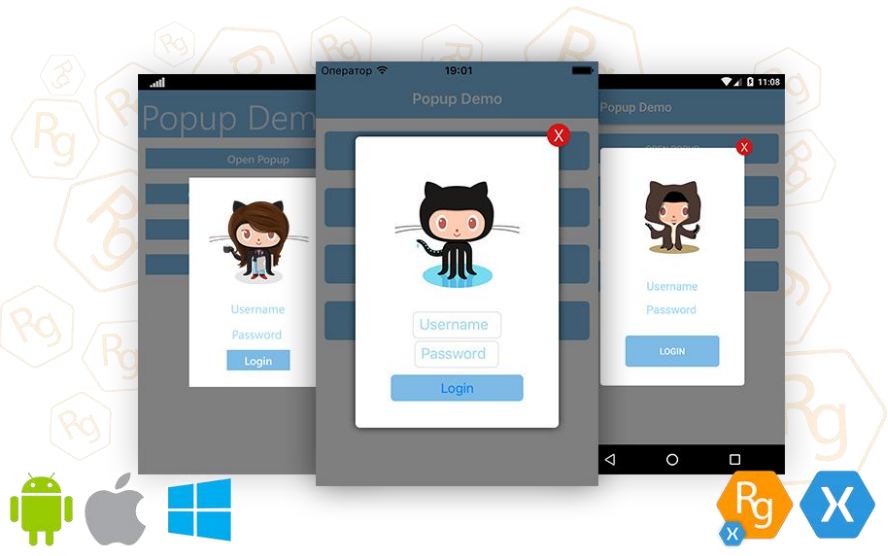
Plugin.LocalNotification je NuGet paket koji omogućuje korištenje lokalnih notifikacija unutar Xamarin.Forms i .Net MAUI aplikacija. Objavljen je krajem 2022. godine i ima MIT licencu, što daje NuGetu svojevrsni certifikat pouzdanosti. Za njegovo korištenje unutar projekta, treba ga inicijalizirati na svakoj platformi na kojoj će se njegova funkcionalnost pozivati. Neke od funkcionalnosti Plugin.LocalNotification NuGet paketa su ponavljajuća notifikacija, postavljanje posebnih zvukova i slika na notifikaciju, postavljanje posebne akcije na klik notifikacije, notifikacija s fiksnim vremenom izvođenja i mnoge ostale. Trenutno podržava samo Android i iOS uređaje. Testna notifikacija se može vidjeti na slici Sl. 3.2.



Sl. 3.2. Testna notifikacija

3.2.3. Prikazivanje popup-ova

Rg.Plugins.Popup je više-platformski Nuget paket za Xamarin.Forms koji omogućuje otvaranje Xamarin.Forms stranica kao *popup* koji se može koristiti u iOS, Android, UWP, WPF, Tizen i macOS aplikacijama. Ovaj paket objavljen je u 2022. godini pod MIT licencom, a ima i mogućnost stvaranje jednostavnih i fleksibilnih animacija za prikazivanje *popup-a*. Za njegovo korištenje, Rg.Plugins.Popup paket treba inicijalizirati unutar projekta za platformu na kojoj se želi koristiti. Testni *popup-ovi* se mogu vidjeti na slici Sl. 3.3.



Sl. 3.3. Primjer Rg.Plugins.Popup-a s Github-a [17]

3.2.4. Kolekcija Xamarin alata zajednice

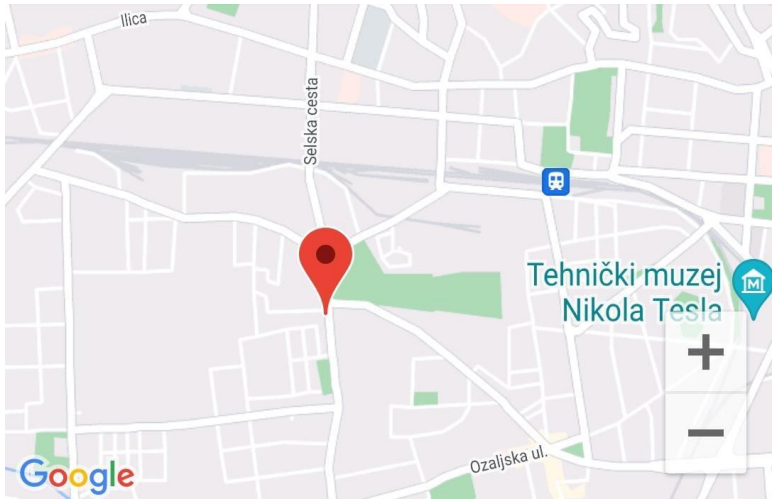
Xamarin.CommunityToolkit je kolekcija animacija, ponašanja, pretvarača i efekata za razvoj mobilnih aplikacija u Xamarin.Forms-u. On je dio .Net Foundationa, koji je neovisna, neprofitna organizacija za potporu inovativnih, komercijalno dostupnih ekosustava otvorenog koda na .NET platformi. Xamarin.CommunityToolkit ima 2.8 milijuna preuzimanja i MIT licencu, a njegovi vlasnici su Xamarin i Microsoft. [18] On je vrlo popularan i pouzdan NuGet paket, a samo jedan od mnogih primjera njegovog korištenja je *AsyncCommand*, koji blokira višestruko izvršavanje jedne naredbe na korisničkom sučelju. Primjer korištenja naredbe *AsyncCommand* se može vidjeti kao programski kod 3.1.

```
CreateRegularExamCommand = new AsyncCommand(CreateRegularExamAsync, allowsMultipleExecutions: false);
        ChangeLocationCommand = new AsyncCommand(ChangeLocationAsync, allowsMultipleExecutions:
false);
        DeviceHistoryCommand = new AsyncCommand(DeviceHistoryAsync, allowsMultipleExecutions: false);.
```

Programski kod 3.1. Primjer korištenja *AsyncCommand-a* unutar aplikacije

3.2.5. Korištenje karti

Xamarin.Forms.Maps je Nuget paket s MIT licencom i 8.4 milijuna preuzimanja te se koristi za ostvarivanje funkcionalnosti karata unutar aplikacije. Za pristup korisnikovoj lokaciji aplikacija mora imati posebna prava pristupa. Kontrola Mapa je višeplatformski pogled za prikazivanje i anotaciju karata. Koristi nativnu kontrolu za karte na svakoj platformi te tako pruža brzo i poznato iskustvo s kartama svim korisnicima. Funkcionalnost karata se unutar ovog diplomskog rada koristi za prikaz lokacija servisa vatrogasnih aparata. Primjer korištenja karte može se vidjeti na slici Sl. 3.4.



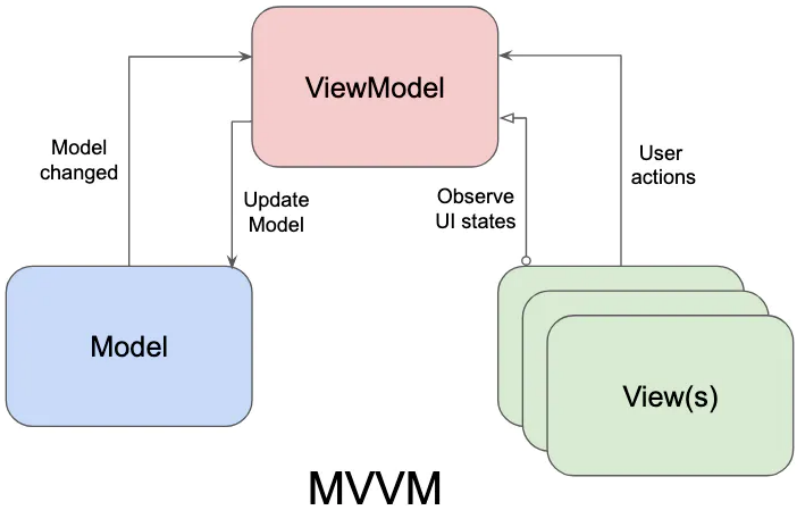
Sl. 3.4. Xamarin.Forms.Maps karte

3.2.6. Skeniranje barkoda

ZXing.Net.Mobile je C#/.NET *library* temeljen na barkod *libraryju* otvorenog koda ZXing. Podržava iOS, Android, Tizen i UWP platforme. Zxing.Net.Mobile omogućuje jednostavno skeniranje barkodova unutar mobilnih aplikacija. Za njegov rad, treba ga inicijalizirati unutar projekata na svakoj platformi na kojoj će se koristiti. Unutar ovog diplomskog rada funkcionalnost skeniranja barkoda koristi se za pretraživanja vatrogasnih aparata prema njihovom barkodu.

3.3. MVVM arhitektura

Obrazac korišten za izradu aplikacije je MVVM (*Model-View-ViewModel*.) To je obrazac strukturiran tako da odvoji grafičko sučelje (*view*) od razvoja poslovne logike ili *back-end* logike (modela,) tako da *view* nije ovisan ni o jednom specifičnom modelu platforme. *Viewmodel* je pretvarač vrijednosti, što znači da je *viewmodel* odgovoran za pretvaranje podatkovnih objekata od modela tako da se njima lakše upravlja i lakše ih je prezentirati. MVVM ima i *binder*, koji automatizira komunikaciju između *viewa* i *propertyja* vezanim za *viewmodel*. Deklarativni podaci i vezanje naredbi su implicitni u MVVM arhitekturi. U Microsoftovom rješenju, *binder* je *markup* jezik koji se zove XAML. *Binder* oslobođa programera od obveze pisanja predloška logike za sinkronizaciju *viewmodela* i *viewa*. Bez *bindera*, prije bi se koristili MVP ili MVC arhitektura [19]. Skica prikaza i komunikacije između modula u MVVM arhitekturi se može vidjeti na slici Sl. 3.5.



Sl. 3.5. MVVM arhitektura [20]

4. RAZVOJ APLIKACIJE

Aplikacija je funkcionalno podijeljena u nekoliko mapa koje olakšavaju snalaženje po aplikaciji. Početna točka aplikacije je *App.xaml* datoteka. Unutar nje, postavlja se *MainPage* aplikacije, koji je tipa *AppShell*. Unutar klase *AppShell* se registriraju sve stranice *ContentPage* koji se koriste u aplikaciji te s njihovom registracijom omogućujemo navigaciju po rutama unutar *AppShella*. Za provjeru postoji li prijavljen korisnik s postavljenim *CheckBox-om* Zapamti me prilikom prijave, koristi se *InitializeApp* metoda. Konstruktor klase App prikazan je programskim

```
public App()
{
    InitializeComponent();
    InitializeApp();
      }
```

Programski kod 4.1. App konstruktor i poziv InitializeApp metode

kodomProgramski kod 4.1.

4.1. Postavljanje ViewModela

Za prikaz različitih ekrana i navigaciju po aplikaciji koristi se Xamarin *ContentPage*. Za korištenje MVVM arhitekture, svakom *ContentPage-u* treba postaviti *ViewModel*, točnije, njegov *BindingContext*. Za dohvaćanje *ViewModela* i servis klasa unutar aplikacije, služi *DIContainer* klasa koja koristi *Dependency injection* (DI). DI je koncept u programiranju koji omogućuje izoliranje ovisnosti između komponenti objektno orijentiranog sustava i njihovo ubrizgavanje izvana. To znači da se objekti ne brinu o tome kako će dobiti svoje ovisnosti, nego se one ubrizgavaju u objekte izvan njih. Takva praksa pisanja koda olakšava testiranje, održavanje i ponovno korištenje komponenti u sustavu [21]. U aplikaciji *Dependency injection* pretežito služi

```
public partial class InitialPage : ContentPage
{
    #region Constructors

        public InitialPage()
    {
                InitializeComponent();
        BindingContext = DIContainer.Instance.Resolve<InitialViewModel>();
    }

    #endregion Constructors
}
```

za dodavanje servisa koji obavljaju HTTP pozive. Klasa *InitialPage* i postavljanje njenog *BindingContext-a* se može vidjeti kao programski kodProgramski kod 4.2.

4.2. Autentikacija korisnika

Kada korisnik prvi puta upali aplikaciju, pronalazi se na ekranu *InitialPage*. *InitialPage* omogućuje korisniku odlazak na ekrane s funkcionalnostima prijave postojećih korisnika,

Sl. 4.1. *InitialPage*

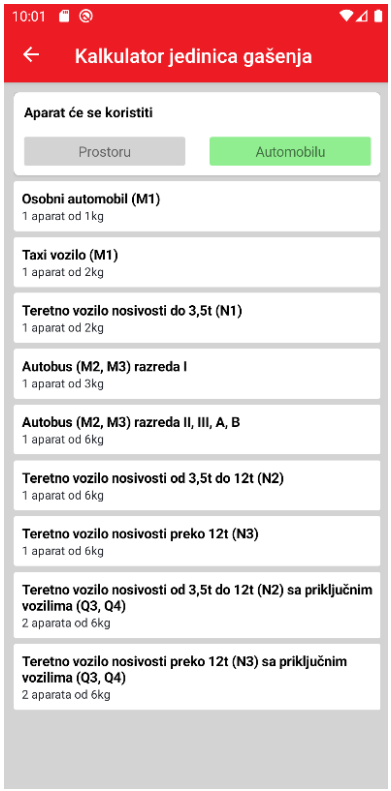
registracije novih korisnika, kalkulatora jedinica gašenja, kao i kontaktu tvrtke putem elektroničke pošte, mobitela i pristupu njihovoj web stranici. Izgled ekrana *InitialPage* se može vidjeti na slici Sl. 4.1.

4.2.1. Kalkulator jedinica gašenja

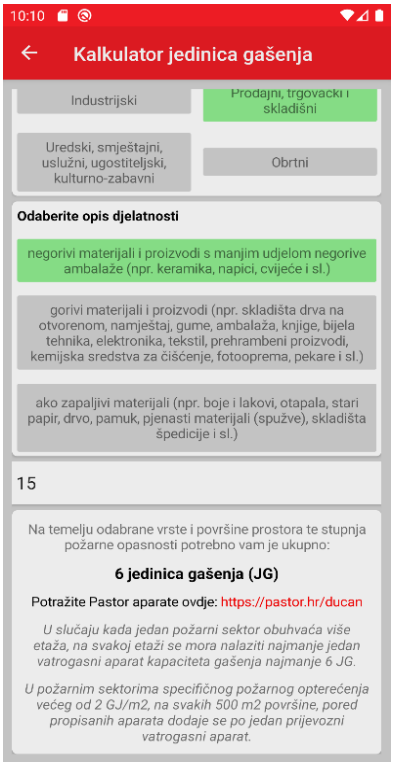
Kalkulator jedinica gašenja omogućuje korisniku izračun koliko mu je i kakvih vatrogasnih aparata potrebno za određeni prostor. Velika podjela u računanju se vrši odabirom na okruženje u kojem će se aparat koristiti, a mogući odabiri u izborniku su prostor i automobil. Klikom na vrstu vozila za koje korisnik želi izvršiti izračun, odveden je na stranicu proizvođača na kojoj može vidjeti potreban aparat. Padajući izbornik za odabir vozila može se vidjeti na slici Sl. 4.2. Ako korisnik želi

izračunati kakav mu je tip aparata potreban za prostor, a ne za automobil, ima drugačiji izbornik. U njemu može izabrati o kakvoj je vrsti prostora riječ te odabrati opis djelatnosti za svaki od odabranih prostora. Naposlijetku, korisnik treba unijeti kvadraturu odabranog prostora kako bi se mogao odraditi potreban izračun za jedinice gašenja. Kada korisnik unese valjani podatak, dobit će broj potrebnih jedinica gašenja, kao i neke dodatne informacije o požarnim sektorima i

poveznicu na stranicu proizvođača vatrogasnih aparata. Kompletan unos informacija o kalkulatoru jedinica gašenja za prostor može se vidjeti na slici Sl. 4.3.



Sl. 4.2. Odabir vozila u kalkulatoru jedinica gašenja



Sl. 4.3. Odabir prostora u kalkulatoru jedinica gašenja

4.2.2. Registracija novih korisnika

Klikom na tipku registracija na *InitialPage-u*, korisnik odlazi na *RegisterPage*. Na njemu može unijeti svoje podatke koji su korišteni za njegovu validaciju u sustav. Dok god korisnik nema valjan unos u svakom inputu za tekst, tipka Pošalji zahtjev će biti onemogućena. Kada korisnik unese podatke i kada pritisne tipku Pošalji zahtjev, poslat će se zahtjev na *back-end* za kreiranje novog korisnika. Po uspješnom kreiranju novog korisničkog računa, korisnik dobiva kod za prijavu u sustav na elektroničku poštu koju je naveo u formi za registraciju. Ukoliko korisnik ne unese valjane podatke, prikazano mu je upozorenje te bi trebao provjeriti unesene podatke. Metoda *SendRegistrationRequestAsync* koja se poziva klikom na tipku Pošalji zahtjev se može vidjeti kao programski kodProgramski kod 4.3.

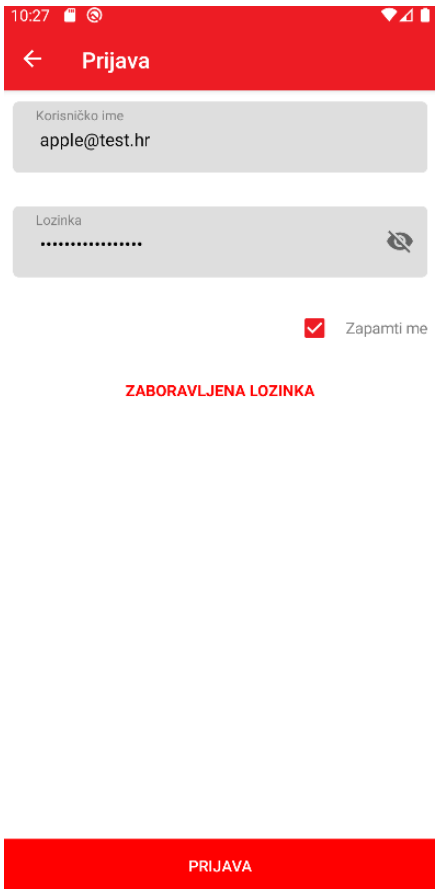
```
        private async Task SendRegistrationRequestAsync()
{
    try
    {
        if (!InputIsValid)
        {
            return;
        }

        IsBusy = true;
        var newUser = new NewUser
        {
            Name = Name,
            Surname = Surname,
            Email = Email,
            PostalCode = PostalCode,
            City = City,
            Address = Address,
            PhoneNumber = PhoneNumber
        };
        var response = await _userService.RegisterNewUser(newUser);
        if (response != null)
        {
            await Shell.Current.DisplayAlert("Zahtjev poslan", "Provjerite svoj email", "OK");
            return;
        }
        await Shell.Current.DisplayAlert("Neuspješna registracija", "Provjerite unešene podatke.", "OK");
        IsBusy = false;
    }
    catch { }
}
```

4.2.3. Prijava korisnika

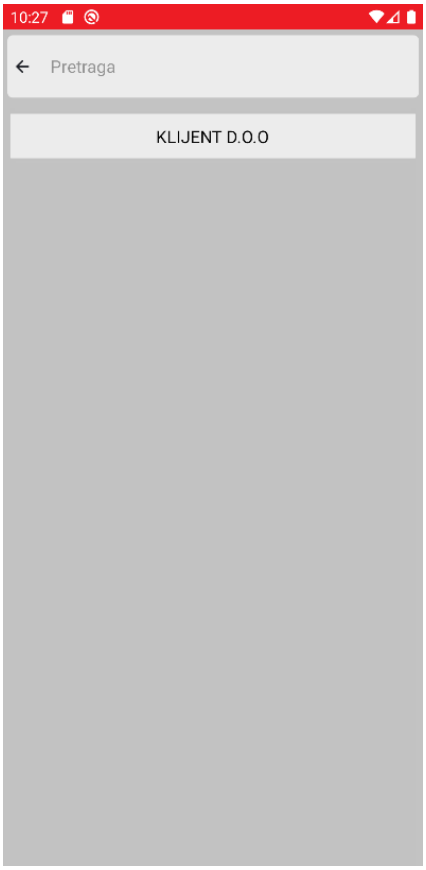
Klikom na tipku Prijava, korisnik odlazi na *SignInPage*. Na njemu korisnik može obnoviti zaboravljenu lozinku i prijaviti se u sustav. Ukoliko korisnikova prijava bude uspješna i označio je *CheckBox* Zapamti me, prilikom idućeg pokretanja aplikacije koriste se njegovi podatci

spremljeni na lokalnu pohranu za autentikaciju i puštaju ga direktno u aplikaciju. Ukoliko korisnik nije označio *CheckBox* Zapamti me, prilikom svakog novog pokretanja aplikacije prikazan mu je ekran *InitialPage*. Kada korisnik unese korisničko ime i lozinku, omogućena je tipka za prijavu korisnika. Ekran *SignInPage* se može vidjeti na slici Sl. 4.4.



Sl. 4.4. *SignInPage* za prijavu postojećih korisnika u sustav

Nakon uspješne prijave, korisniku se otvara *ClientPage*. Na njemu su mu prikazani svi klijenti kojima korisnik ima pristup. Odabirom odgovarajućeg klijenta, otvara mu se *LocationsPage*, na kojem može odabrati željenu lokaciju za pregled vatrogasnih aparata. *ClientPage* se može vidjeti na slici Sl. 4.5. *ClientPage* i *LocationsPage* su slični po dizajnu i koriste istu logiku filtriranja pomoću unosa iznad liste objekata koje korisnik može odabrati. Metoda *FilterLocations* za filtriranje lokacija se može vidjeti kao programski kodProgramski kod 4.4. Odabirom lokacije, korisniku se postavlja novi *MainPage* i ulazi u glavni dio aplikacije, *RootPage*.



Sl. 4.5. *ClientPage* za odabir klijenta korisnika

```
    private void FilterLocations()
    {
        if (_searchInput.Equals(string.Empty))
        {
            Items = new ObservableCollection<BindableLocation>();
            UnfilteredLocations.ForEach(c => Items.Add(c));
            OnPropertyChanged(nameof(Items));
        }
        else
        {
            var filteredItems = UnfilteredLocations.Where(x =>
x.Name.ToLower().Contains(_searchInput.ToLower())).ToList();
            Items = new ObservableCollection<BindableLocation>();
            filteredItems.ForEach(c => Items.Add(c));
            OnPropertyChanged(nameof(Items));
        }
    }
```

4.3. Glavni dio aplikacije

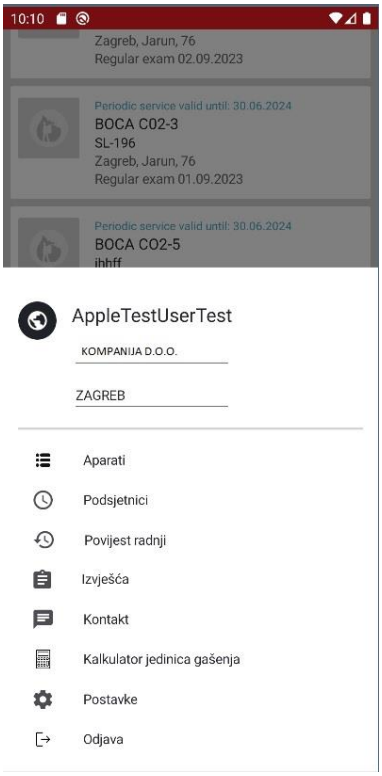
Nakon odabira klijenta, lokacije i uspješne autentikacije, korisniku je prikazan *RootPage*. On predstavlja središnji dio aplikacije u kojem korisnik obavlja sve aktivnosti. Na ekranu *RootPage* korisnik može odraditi pregled, dodati novi aparat, prijaviti kvar, kreirati izvješće, upravljati

profilom, odabranom lokacijom, klijentom i pristupiti kalkulatoru jedinica gašenja. Izgled ekrana *RootPage* se može vidjeti na slici Sl. 4.5.

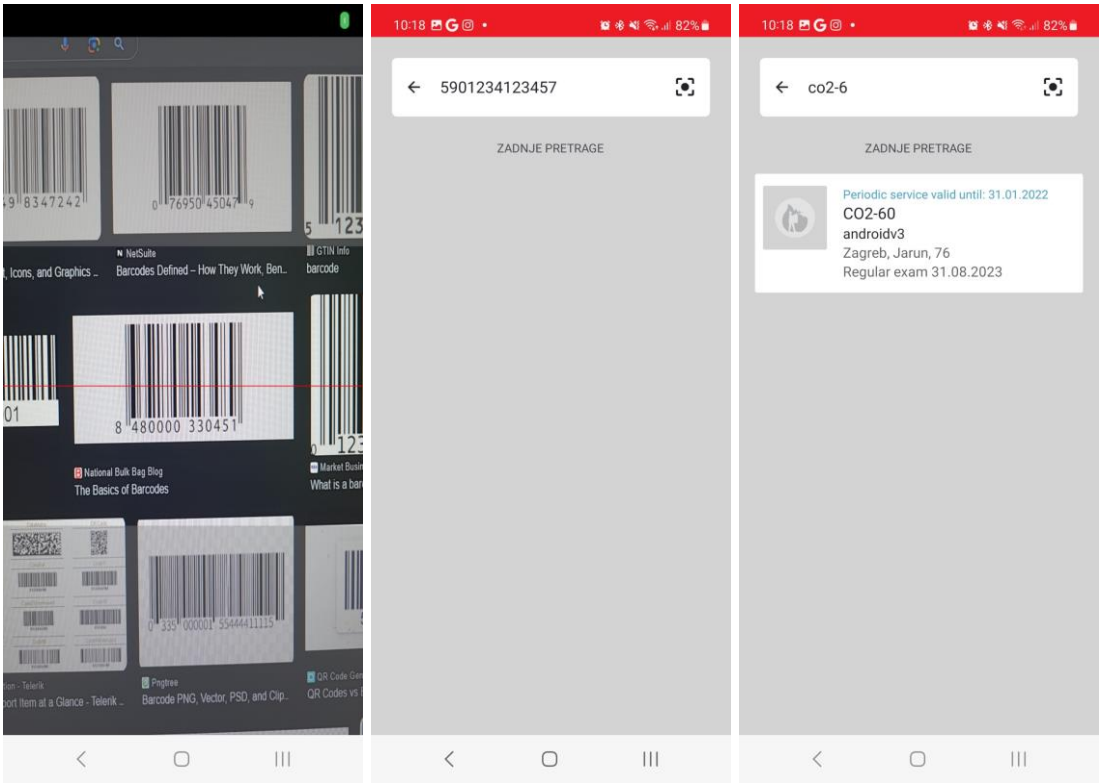
Sl. 4.5. *RootPage* – glavni dio aplikacije

Klikom na kontrolu na dnu ekrana, korisniku se prikazuju *popup-ovi* koji mu omogućuju promjenu aktivnog pogleda na ekranu *RootPage*, promjenu sortiranja prikazanih vatrogasnih uređaja, otvaranje izbornika za pretraživanje vatrogasnih uređaja i aktivnosti vezanje uz upite, dodavanje novih uređaja i kreiranje izvješća. Klikom na *Hamburger menu* ikonu na kontroli na dnu ekrana *RootPage*, korisniku se otvara *MenuPopupPage*. Na njemu korisnik može pristupiti ostalim funkcionalnostima aplikacije, koje su opisane u nastavku rada. Izgled ekrana *MenuPopupPage* može se vidjeti na slici Sl. 4.6. Klikom na ikonu povećala na kontroli na dnu ekrana *RootPage*, korisniku će se otvoriti *DashboardSearchPage*, na kojem će moći obaviti pretragu uređaja prema njihovom tipu i barkodu pomoću kontrole za pretragu koja se nalazi na vrhu ekrana. Prikaz korištenja barkod skenera i pretrage uređaja po barkodu i njihovom tipu može se vidjeti na slici Sl. 4.7. Kod za pretragu uređaja prema barkodu i tipu uređaja je sličan kodu za pretragu lokacija koji se može vidjeti kao programski kodProgramski kod 4.4. Klikom na ikonu desnu ikonu na kontroli na dnu ekrana *RootPage*, korisniku se otvara *SortPopupPage* na kojem može odabrati način sortiranja uređaja na početnoj stranici ekrana *RootPage*. Vrste sortiranja su po nadolazećem pregledu, zadnjem datumu dodavanja i po lokaciji. Metoda za sortiranje uređaja po lokaciji se može vidjeti

kao programski kod Programski kod 4.5. Klikom na neki od aparata korisniku će se otvoriti novi ekran *DeviceDetailsPage*, gdje su mu prikazani detalji o odabranom uređaju.



Sl. 4.6. *MenuPopupPage* – pristup dodatnim funkcionalnostima u aplikaciji



Sl. 4.7. Korištenje barkod skenera i pretraga uređaja na *DashboardSearchPage-u*

```
        private async Task SortByLocation()
    {
        AllDevices = new ObservableCollection<Device>(AllDevices.OrderByDescending(x => x.SifraLokacije));
        ScheduledDevices = new ObservableCollection<Device>(ScheduledDevices.OrderByDescending(x =>
x.SifraLokacije));
        UpdatedDevices = new ObservableCollection<Device>(UpdatedDevices.OrderByDescending(x =>
x.SifraLokacije));

        OnPropertyChanged(nameof(UpdatedDevices));
        OnPropertyChanged(nameof(ScheduledDevices));
        OnPropertyChanged(nameof(AllDevices));
        await SafeClearPopups();
    }
```

Programski kod 4.5. Sortiranje uređaja na *RootPage-u* prema lokaciji

Klikom na plus na kontroli na dnu ekrana *RootPage* korisniku se otvara *DashboardAddPopupPage*, na kojem korisnik može odabrati redovni pregled uređaja, dodati novi uređaj, prijaviti kvar, poslati upit iz aplikacije i kreirati izvješće.

4.3.1. Detalji o odabranom uređaju

Za prikazivanje detalja o odabranom uređaju, koristi se *DeviceDetailsPage*. Na njemu korisnik može dalje upravljati s odabranim uređajem, pregledati povijest radnji na njemu, promijeniti mu lokaciju, odraditi redovni pregled i vratiti se na prethodni page. Ekran *DeviceDetailsPage* se može vidjeti na slici **Error! Reference source not found.**

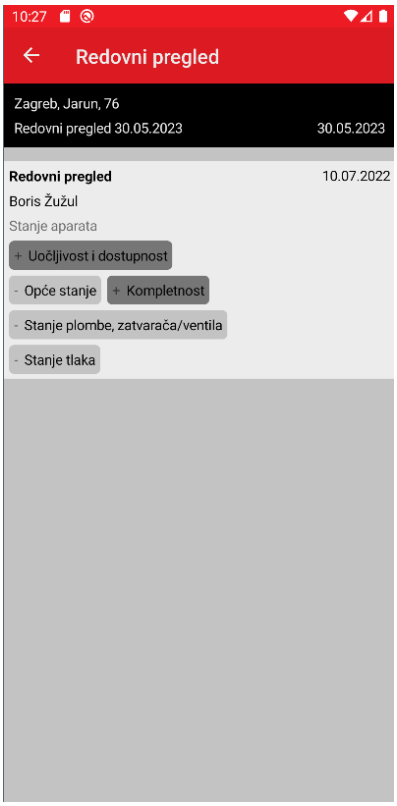
Sl. 4.8. *DeviceDetailsPage* – prikaz detalja o odabranom uređaju

Učitavanje podataka za *DeviceDetailsPage* izvršava se pozivanjem metode *LoadData*. U njoj se vrši dodatna provjera za prikaz ažuriranih podataka ako je za uređaj napravljen redovan pregled. Metoda *LoadData* se može vidjeti kao programski kod Programski kod 4.6

```
private async Task LoadData()
{
    SelectedDevice = _deviceService.SelectedDevice;
    var UserId = Preferences.Get("userId", string.Empty);
    var ClientId = Preferences.Get("clientId", string.Empty);
    var locations = await _deviceService.GetLocation(ClientId, UserId);
    var LocationId = Preferences.Get("locationId", string.Empty);
    var location = locations.Find(l => l.LocationId == LocationId);
    SelectedDeviceLocation = location;
    var locationName = Preferences.Get("locationName", string.Empty);
    LocationName = CultureInfo.CurrentCulture.TextInfo.ToTitleCase(locationName.ToLower());
    OnPropertyChanged(nameof(SelectedDevice));
    ProcessUpdatedDevice();
}
```

Programski kod 4.6 Metoda *LoadData DeviceDetailsPage-a* za učitavanje podataka o odabranom aparatu

Klikom na lijevu tipku na kontroli na dnu ekrana, otvara se *DeviceHistoryPage* koji prikazuje pregled povijesti radnji za odabrani uređaj. Na njemu korisnik može vidjeti informacije o periodočnim servisima, redovnim pregledima i unutarnjim pregledima uređaja. Ovisno o tome kakav tip povijesti radnji korisnik odabere, na idućem ekranu *DeviceHistoryExamSelectionPage* učitani su podatci o takvom tipu pregleda. Kada korisnik odabere željeni datum, otvara se *DeviceHistoryExamDetailsPage* gdje su prikazani detalji o odabranom pregledu. Primjer prikaza podataka ekrana *DeviceHistoryExamDetailsPage* za odabrani redovni pregled može se vidjeti na slici Sl. 4.9. Klikom na desnu tipku na kontroli na dnu ekrana *DeviceDetailsPage* korisniku se otvara *DeviceRegularExamPage* koji pruža izbornik za redovni pregled odabranog uređaja. Na njemu može promijeniti stanje uporabe aparata, njegovu uočljivost i dostupnost, njegovo opće stanje, kompletnost i stanje plombe. Korisnik također može napisati dodatnu primjedbu u slučaju da je primijetio nešto na aparatu te upisati svoje ime i prezime kao voditelja upisnika. Klikom na tipku Ažuriraj stanje, aparat se dodaje u listu ažuriranih uređaja koji kasnije mogu biti korišteni za izradu izvješća. Metoda *UpdateDeviceAsync* korištena za ažuriranje odabranog uređaja se može vidjeti kao programski kod **Error! Reference source not found.**U njoj se prvo radi provjera je li unesen dodatni prigovor prilikom izrade izvještaja, nakon toga poziva se metoda *AddDevice* servisa *DeviceService*, čijim se pozivom za postojeći uređaj unutar sustava odrađuje redovni



Sl. 4.9. *DeviceHistoryExamDetailsPage* – prikaz detalja redovnog pregleda odabranog uređaja

pregled uređaja. Ukoliko je došlo do greške prilikom ažuriranja prigovora ili izrade redovnog pregleda za uređaj, korisnik je o tome obavješten u formatu *Toast* poruke na ekranu. U suprotnom, dobiva povratnu informaciju da je željeni zahtjev uspješno izvršen. Nakon uspješnog zahtjeva za ažuriranje uređaja, odabrani uređaj je spremljen na lokalnu pohranu uređaja s *AddToUpdatedDevices* metodom zbog izrade PDF izvješća o ažuriranim uređajima. Klikom na ikonu olovke na dnu ekrana *DeviceDetailsPage*, korisniku se otvara *ChangeDeviceLocationPage*. Na njemu korisnik može promijeniti informacije o odabranoj lokaciji odabranog uređaja, može dodati novu lokaciju i odabrati novu lokaciju za uređaj iz postojećih lokacija. *ChangeDeviceLocationPage* se može vidjeti na slici Sl. 4.10. Klikom na ikonu kućice na dnu ekrana *DeviceDetailsPage*, korisnik se vraća na prethodni ekran *RootPage*.

4.3.2. Dodavanje novog vatrogasnog aparata

Klikom na Dodaj aparat na ekranu *DashboardAddPopupPage*, korisniku se otvara novi ekran *AddDevicePage*, na kojem može dodati novi vatrogasni aparat. Na njemu treba unijeti tvornički broj novog uređaja, odabrati mu godinu proizvodnje, odabrati tip novog aparata iz postojećih tipova aparata, proizvođača aparata i lokaciju aparata, odabrati datum kupnje aparata te mu odrediti

specifikacije dostupnosti, općeg stanja, kompletnosti, stanja tlaka i stanja plombe, zatvarača i ventila. Kada je korisnik odabrao sve navedene parametre, tipka na dnu

```
private async Task UpdateDeviceAsync()
{
    if (!String.IsNullOrEmpty(Complaint))
    {
        var remark = new RemarkModel
        {
            ClientId = _clientId,
            DeviceType = SelectedDevice.DeviceType,
            FactoryNumber = SelectedDevice.FactoryNumber
            ManufacturerCode = SelectedDevice.ProducerId,
            ProductionYear = SelectedDevice.YearManufactured,
            Remark = Complaint
        };
        var remarkResponse = await _deviceService.ChangeRemark(remark);
        if (remarkResponse.Msg.Length > 1)
        {
            await Shell.Current.DisplayToastAsync(remarkResponse.Msg);
        }
    }

    var device = new Device
    {
        SerijskiBroj = SelectedDevice.Id,
        GodinaProizvodnje = SelectedDevice.YearManufactured,
        TipAparata = SelectedDevice.DeviceType,
        SifraProizvodjaca = SelectedDevice.ProducerId,
        SifraKlijenta = SelectedDevice.ClientId,
        Aktivan = SelectedDevice.InUse ? "0" : "1",
        DatumKupnje = SelectedDevice.PurchaseDateTime,
        Dostupnost = IsAvailable ? "0" : "1",
        OpceStanje = IsGoodGeneralState ? "0" : "1",
        Kompletnost = IsComplete ? "0" : "1",
        StanjePlombe = IsGoodSeal ? "0" : "1",
        StanjeTlaka = SelectedDevice.PreassureCondition,
        SifraKorisnika = _userId,
        TvornickiBroj = SelectedDevice.FactoryNumber,
        SifraLokacije = SelectedDevice.SifraLokacije.ToString(),
        DatumRedovnogPregleda = DateTime.UtcNow,
        Barcode = SelectedDevice.Barcode,
        DatumPerServ = SelectedDevice.PeriodicExamDateTime,
        DatumUnPregled = SelectedDevice.InnerExamDateTime,
        PodTlakom = SelectedDevice.PreassureCondition,
        RazlogPerServ = string.Empty,
        RazlogUnPregled = string.Empty
    };
    var response = await _deviceService.AddDevice(device);
    if (response.Msg.Length > 1)
    {
        await Shell.Current.DisplayToastAsync(response.Msg);
        return;
    }
    else
```

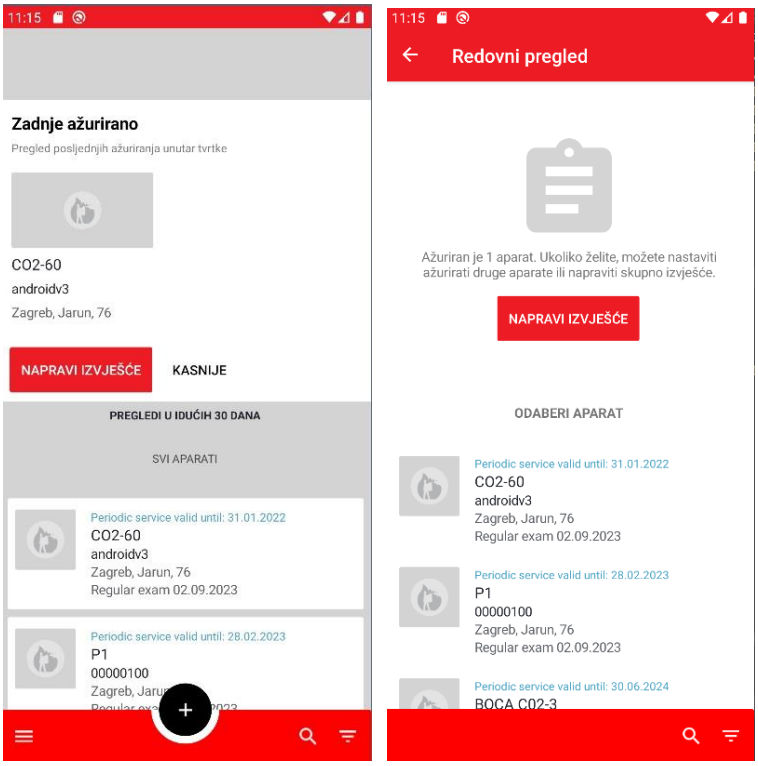
Sl. 4.10. *ChangeDeviceLocationPage* – promjena detalja o lokaciji uređaja

ekrana *DashboardAddPopupPage* Dodaj aparat je omogućena. Na njen klik, izvršava se logika za dodavanje novog vatrogasnog aparata pozivanjem metode *AddDeviceAsync* koja se može vidjeti kao programski kod**Error! Reference source not found.**Nakon pozivanja metode *AddDevice* u servisu *DeviceService*, korisnik dobiva odgovarajuću *Toast* poruku o uspješnosti izvršavanja njegovog zahtjeva za dodavanje novog uređaja.

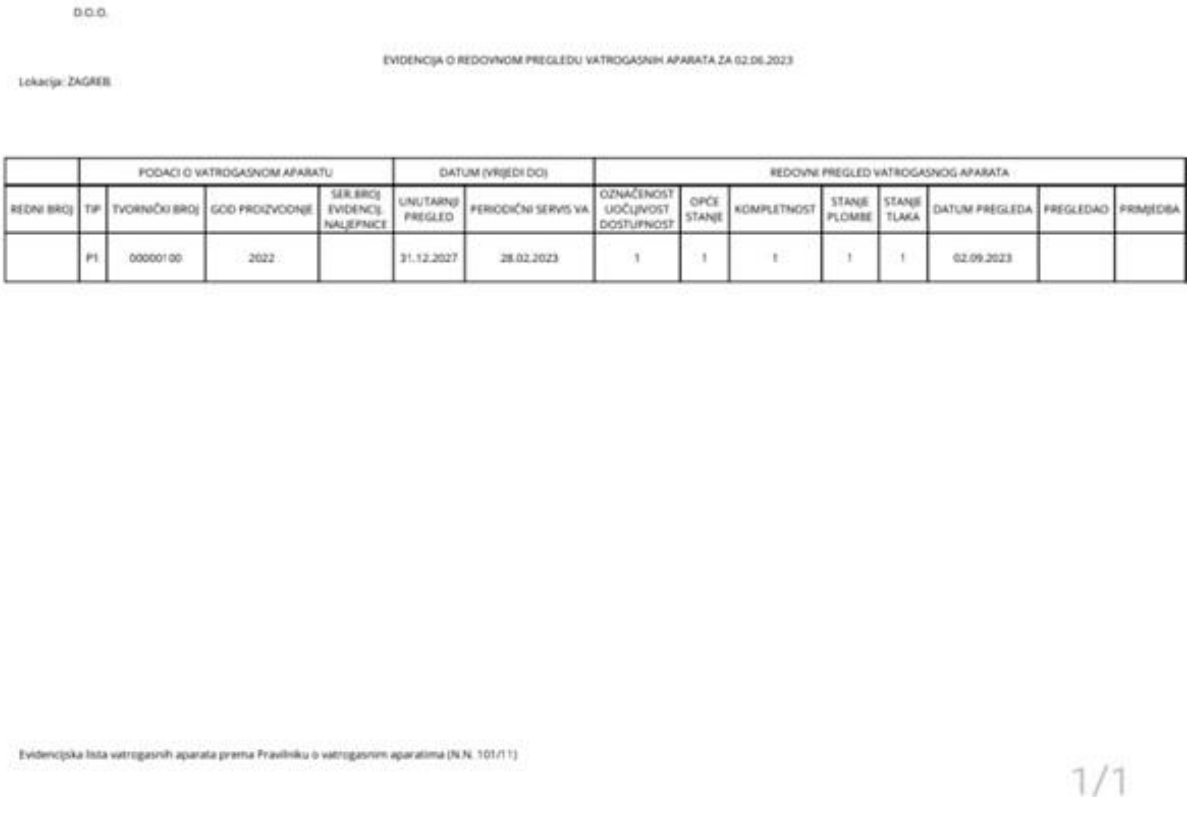
4.3.3. Kreiranje izvješća

Kada je napravio izmjene, korisnik na ekranima *RootPage* i na *RegularReportPage* vidi mogućnost za izradu izvješća o uređajima na kojima je od zadnje izrade izvješća napravio redovni pregled. Izgled ekrana *RootPage* i *RegularReportPage* kada postoje uređaji na kojima je odrađen redovni pregled se mogu vidjeti na slikama **Error! Reference source not found.**Klikom na tipku Napravi izvješće, otvara se ekran *CreateReportPage* na kojem korisnik može odabrati vremenski raspon ažuriranih uređaja u kojem želi da mu se izvješće izgenerira. Prilikom generiranja izvješća, poziva se *PdfGeneratorService* koji koristi PdfSharpCore za crtanje tablice i generiranje bajtova koji će biti korišteni unutar servisa na mobilnim platformama za prikazivanje stvorenog PDF dokumenta. Izvješće se može izgenerirati i s ekrana Izvješća kada se odabere datum pregleda jednog od uređaja. Primjer izgeneriranog PDF dokumenta se može vidjeti na slici **Error! Reference source not found.**

```
private async Task AddDeviceAsync()
{
    try
    {
        IsBusy = true;
        var clientId = Preferences.Get("clientId", string.Empty);
        var userId = Preferences.Get("userId", string.Empty);
        var newDevice = new Device
        {
            TvornickiBroj = SerialNumber,
            SifraKlijenta = clientId,
            SifraKorisnika = userId,
            GodinaProizvodnje = ManufacturerYear,
            TipAparata = DeviceType,
            SifraProizvodjaca = ManufacturerId,
            SifraLokacije = SelectedLocation.LocationId,
            DatumKupnje = YearBoughtDateTime,
            Dostupnost = Availability.Equals("DA") ? "1" : "0",
            OpceStanje = GeneralState.Equals("Dobro") ? "1" : "0",
            StanjePlombe = SealState.Equals("Uredno") ? "1" : "0",
            StanjeTlaka = PreassureState.Equals("Dobro") ? "1" : "0",
            Kompletnost = Completeness.Equals("DA") ? "1" : "0"
        };
        var response = await _deviceService.AddDevice(newDevice);
        if (response.Msg.Length > 1)
        {
            await Shell.Current.DisplayToastAsync(response.Msg);
            return;
        }
        await Shell.Current.DisplayToastAsync("Uspješno dodan uređaj.");
        IsBusy = false;
    }
```



Sl. 4.11. *RootPage* i *RegularReportPage* u slučaju odrađenog redovnog pregleda



D.O.O.

EVIDENCIJA O REDOVNOM PREGLEDU VATROGASNIH APARATA ZA 02.06.2023

Lokacija: ZAGREB

	PODACI O VATROGASNOM APARATU				DATUM (VRIJEDI DO)		REDOVNI PREGLED VATROGASNOG APARATA							
REDNI BROJ	TIP	TVORNIČKI BROJ	GOD PROIZVODNJE	SER.BROJ EVIDENCIJ. NALJEPNICE	UNUTARNJI PREGLED	PERIODIČNI SERVIS VA	OZNAČENOST UOČLJIVOST DOSTUPNOST	OPĆE STANJE	KOMPLETNOST	STANJE PLOMBE	STANJE TLAKA	DATUM PREGLEDA	PREGLEDAO	PRIMJEDBA
	P1	00000100	2022		31.12.2027	28.02.2023	1	1	1	1	1	02.09.2023		

Evidencijska lista vatrogasnih aparata prema Pravilniku o vatrogasnim aparatima (N.N. 101/11)

1/1

Sl. 4.12. Izvješće izgenerirano od uređaja na redovnom pregledu

Za generiranje dokumenta u PdfSharpCore-u potrebno je odrediti nekoliko parametara kako bi dokument odgovarao specifikacijama. Generirani dokument treba sadržavati tablicu koja prikazuje informacije o uređajima kojima je odrađen redovni pregled, a u podnožju i zaglavlju dokumenta treba prikazati informacije o datumu pregleda, informacije o odabranom klijentu i lokaciji uređaja, kao i informaciju o korištenom pravilniku o vatrogasnim aparatima.

U slučaju kada postoji više vatrogasnih aparata, važno je definirati koliko aparata treba prikazati korisniku na jednoj stranici. Kada se odredi veličina stranice PDF dokumenta na veličinu papira A4, dođe se do optimalne brojke od 5 prikazanih aparata po stranici dokumenta. Tablica se treba sastojati od 15 stupaca, a stupci se mogu promatrati kroz dvije velike skupine, najveća od njih je *PODACI O VATROGASNOM APARATU, DATUM (VRIJEDI DO) i REDOVNI PREGLED VATROGASNOG APARATA* koji funkcionalno grupiraju manje skupine koje im pripadaju.

Metode koje je važno istaknuti kod crtanja tablice u *PDFGeneratorService-u* su metode *DrawCell* i *DrawLabels*. Metoda *DrawLabels* služi za crtanje teksta u podnožje i zaglavlje dokumenta. Prema vrijednostima margine koje su postavljene za tekst labele, prilagođava se crtanje tablice kako se tekst ne bi preklapao. Crtanje labela se izvodi pomoću podataka dohvaćenim iz servisa

UserService i *DeviceService*, a izgled *DrawLabels* metode se može vidjeti kao programski kod Programski kod 4.9. Nakon crtanja labela, prvo treba nacrtati prva dva reda tablice. Kako bi se tablica prilagodila potrebnim širinama teksta, bez da postoji nepotreban prevelik razmak u nekim od stupaca, računa se najveća širina ćelije drugog reda tablice, pa se prvi red tablice crta na osnovu dobivenih vrijednosti njegovih povezanih elemenata iz drugog reda. Metoda za računanje ćelije koja ima najveću širinu se može vidjeti kao programski kodProgramski kod 4.12. Metoda *DrawCell* služi za crtanje ćelija tablice. Prije svega treba centrirati tekst unutar dostupnog prostora, zatim provjeriti postoji li tekst koji treba nacrtati i u tom slučaju nacrtati praznu ćeliju. Za tekst koji u sebi sadrži znak za novi red \n, tekst treba podijeliti na više manjih dijelova kako bi PdfSharpCore znao da taj tekst treba nacrtati u više dostupnih linija. Metoda *DrawCell* se može vidjeti kao

```
private async Task DrawLabels(XGraphics graph, XFont labelFont, XBrush labelBrush, double pageWidth, double
pageHeight, BindableReportDevice selectedDevice = null)
      {
          var clientInfo = await _userService.GetClientInfo(_clientId);
          XStringFormat format = new XStringFormat();
          format.Alignment = XStringAlignment.Near;

          // Draw label in the top right corner
          XRect clientNameBounds = new XRect(20, 20, 80, 20);
          graph.DrawString(clientInfo?.Name, labelFont, labelBrush, clientNameBounds, format);

          // Draw label in the middle top
          XRect examBounds = new XRect((pageWidth - 200) / 2, 50, 200, 20);
          var date = selectedDevice != null ? DateTime.Parse(selectedDevice.RegularReviewDate) : DateTime.Now;
          var formattedDate = date.ToString("dd.MM.yyyy");
          // Calculate the X and Y positions to center the label within the rectangle
          double labelX = examBounds.X + (examBounds.Width - graph.MeasureString($"EVIDENCIJA O
REDOVNOM PREGLEDU VATROGASNIH APARATA ZA {formattedDate}", labelFont).Width) / 2;
          double labelY = examBounds.Y + (examBounds.Height - labelFont.Height) / 2;
          graph.DrawString($"EVIDENCIJA O REDOVNOM PREGLEDU VATROGASNIH APARATA ZA
{formattedDate}", labelFont, labelBrush, labelX, labelY, format);

          var locations = await _deviceService.GetLocation(_clientId, _userId);
          var location = locations.FirstOrDefault(l => l.LocationId == _locationId);
          XRect locationBounds = new XRect(20, 70, 80, 20);
          graph.DrawString($"Lokacija: {location?.Place}, {location?.Street}, {location?.StreetNumber}", labelFont,
labelBrush, locationBounds, format);

          // Draw label in the bottom left corner
          XRect bottomLeftBounds = new XRect(20, pageHeight - 40, 80, 20);
          graph.DrawString("Evidencijska lista vatrogasnih aparata prema Pravilniku o vatrogasnim aparatima (N.N.
101/11)", labelFont, labelBrush, bottomLeftBounds, format);
}
```

Programski kod 4.9. Metoda DrawLabels za crtanje teksta u zaglavlju i podnožju dokumenta

programski kod Programski kod 4.11.

```
private void DrawCell(XGraphics graph, string text, XFont font, XBrush textBrush, XBrush backBrush, XPen
borderPen, double x, double y, double width, double height)
    {
        XStringFormat format = new XStringFormat();
        format.Alignment = XStringAlignment.Center;
        format.LineAlignment = XLineAlignment.Center;
        graph.DrawRectangle(borderPen, x, y, width, height);
        if (text != null)
        {
            XRect cellRect = new XRect(x, y, width, height);
            string[] lines = text.Split('\n');
            List<string> wrappedLines = new List<string>();
            foreach (string line in lines)
            {
                string[] words = line.Split(' ');

                string currentLine = string.Empty;

                foreach (string word in words)
                {
                    string tempLine = currentLine + word + " ";
                    double tempWidth = graph.MeasureString(tempLine, font).Width;

                    if (tempWidth > width)
                    {
                        wrappedLines.Add(currentLine.Trim());
                        currentLine = word + " ";
                    }
                    else
                    {
                        currentLine = tempLine;
                    }
                }

                wrappedLines.Add(currentLine.Trim());
            }
            double totalLinesHeight = wrappedLines.Count * font.Height;
            double startY = cellRect.Top + ((cellRect.Height - totalLinesHeight) / 2);
            for (int i = 0; i < wrappedLines.Count; i++)
            {
                XRect lineRect = new XRect(cellRect.Left, startY + (i * font.Height), width, font.Height);
                graph.DrawString(wrappedLines[i], font, textBrush, lineRect, format);
            }
        }
        else
        {
            XRect cellRect = new XRect(x, y, width, height);
            graph.DrawString("", font, textBrush, cellRect, format);
        }
    }
```

```
    private double CalculateMaxCellDataWidth(XGraphics graph, string[] headerTexts, int columnIndex, XFont
font)
    {
        string cellData = headerTexts[columnIndex];
        if (cellData == null)
            return 0;

        string[] lines = cellData.Split('\n');

        double maxWidth = 0;
        foreach (string line in lines)
        {
            double lineWidth = graph.MeasureString(line, font).Width;
            if (lineWidth > maxWidth)
                maxWidth = lineWidth;
        }

        return maxWidth;
    }
```

4.3.4. Lokalne notifikacije

Kako bi korisnik bio informiraniji o uređajima na kojima treba napraviti pregled, treba dobiti notifikaciju na svom uređaju. Za to je korišten Plugin.LocalNotification NuGet paket. Prilikom učitavanja uređaja na ekranu *RootPage*, izvršava se dodatna logika koja provjerava postoji li uređaj koji ima datum pregleda u idućih 30 dana. Ako takav uređaj postoji, treba ga dodati u kolekciju *ScheduledDevices*. Kod za provjeru i dodavanje uređaja u kolekciju uređaja koji imaju pregled u idućih 30 dana se može vidjeti kao programski kod Programski kod 4.13.

```
var devicesToAdd = devices.Where(device => (device.DatumRedovnogPregleda != null)
&&(device.DatumRedovnogPregleda.Value - DateTime.Now).TotalDays <= 30).ToList();
devicesToAdd.ForEach(device =>
        {
                ScheduledDevices.Add(new Device(device));
        ProcessDeviceNotification(device);
        });
```

Programski kod 4.13. Kod za provjeru redovnog pregleda i dodavanje uređaja u kolekciju *ScheduledDevices*

Postoje tri vrste pregleda na uređajima, unutarnji pregled, periodički pregled i redovan pregled. Svaki od njih ima pripadajuću *bool* vrijednost u Xamarin.Preferences i ta vrijednost se poziva u klasi *RootViewModel* ekrana *RootPage* i u postavkama unutar aplikacije na ekranu *NotificationsPage*, gdje se učitava postojeća vrijednost pripadajuće *Toggle*, a može se postaviti i nova vrijednost za *boolean Toggle* svih notifikacija i tako ih korisnik može isključiti ili uključiti. Primjer dohvaćanja vrijednosti notifikacijskih *Toggle-ova* i kreiranje notifikacije za redovni pregled unutar *ProcessDeviceNotification* metode mogu se vidjeti kao programski kod Programski kod 4.14.

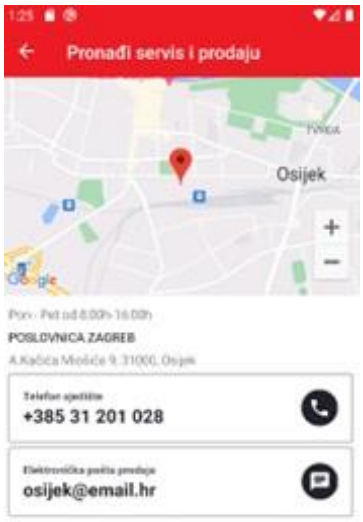
```
var isInnerExamToggled = Preferences.Get("isInnerExamToggled", true);
var isRegularExamToggled = Preferences.Get("isRegularExamToggled", true);
var isPeriodicExamToggled = Preferences.Get("isPeriodicExamToggled", true);
Random rand = new Random();
int randomInt = rand.Next();
if (isRegularExamToggled)
          {
     var notification = new NotificationRequest
                {
                Title = "Notifikacija iz aplikacije",
                Description = $"Redovni pregled za uređaj {device.TvornickiBroj}
{device.DatumRedovnogPregleda.Value.ToString("dd'.'MM'.'yyyy")}",
                NotificationId = randomInt,
                Schedule = new NotificationRequestSchedule
                {
                   NotifyTime = DateTime.Now.Date.AddDays(1).AddHours(8)
                }
           };
           LocalNotificationCenter.Current.Show(notification);
                    }
           }
```

4.3.5. Korištenje karti

Kada korisnik otvori *MenuPopupPage* i odabere opciju kontakt, kao aktivan pogled na ekranu *RootPage* postavlja se *ContactView*. Na njemu korisnik može odabrati različite opcije poput kontakta proizvođača aparata putem elektroničke pošte ili kontakt telefona, a može pregledati i lokacije proizvođača ako korisnik odabere tipku Pronađi servis i prodaju. Tada mu se otvara ekran s različitim lokacijama proizvođača aparata, a po njihovom odabiru se otvara karta s pripadajućim informacijama za odabranu lokaciju. Za prikaz karte na kojoj je prikazana lokacija proizvođača aparata, koristi se Xamarin.Forms.Maps karta. Pri inicijaliziaciji karte treba postaviti *Pin* na kartu kako bi se prikazala lokacija odabranog mjesta i to se ostvaruje s metodom *GoToPin*. *GoToPin* metoda se može vidjeti kao programski kod **Error! Reference source not found.**Korištenje karte se može vidjeti na kao slika Sl. Sl. 4.13.

```
private void GoToPin(object sender, EventArgs e)
{
           Pin selectedLocationPin = new Pin()
     {
                    Type = PinType.Place,
           Label = _viewModel.MapsLocationData.Name,
           Address = _viewModel.MapsLocationData.Address,
           Position = new Position(_viewModel.MapsLocationData.Latitude,
_viewModel.MapsLocationData.Longitude),
       };

           map.Pins.Add(selectedLocationPin);
           map.MoveToRegion(MapSpan.FromCenterAndRadius(selectedLocationPin.Position,
Distance.FromMeters(1000)));
```



Sl. 4.13. Korištenje karte u aplikaciji

4.3.6. Ostale funkcionalnosti

S obzirom na to da aplikacija koristi sustav autentikacije i provjere korisničkih podataka prilikom prijave, treba korisniku omogućiti prijavu i promjenu podataka u slučaju da ih je zaboravio. Ako korisnik ne može ući u sustav jer je zaboravio svoje korisničke podatke, može napraviti zahtjev za resetiranje lozinke prilikom neuspjele prijave na ekranu *SignInPage*. Također može promijeniti svoje korisničko ime u postavkama kada je ulogiran u sustav. Korisnik na izborniku ekrana *RootPage* može ostvariti funkcionalnosti promjene aktivnog klijenta, aktivne lokacije, pregledati podsjetnike za uređaje kojima je pregled u idućih trideset dana, povijest radnji za pojedini uređaj (koje uključuju prediodičke servise, redovni i unutarnji pregled), pregled izvješća za odrađene preglede unutar određenog vremenskog raspona, pristup kalkulatoru za jedinice gašenje i promjeni postavki aplikacije, kontakt informacijama proizvođača vatrogasnih aparata i odjavi iz sustava. Pregled odrađenih vrsta izvješća iz aplikacije se može vidjeti na slici Sl. 4.14. Za odabrano izvješće korisnik može vidjeti dodatne detalje pregleda i o njemu napraviti PDF izvještaj pregleda. Uz prethodno navedeno, korisnik iz aplikacije može poslati upit putem elektroničke pošte, kao i odraditi prijavu kvara za neki od uređaja.

Sl. 4.14. Pregled odrađenih izvješća u aplikaciji

5. KORIŠTENJE APLIKACIJE

U nastavku je pojašnjeno korištenje aplikacije uz dijelove korisničkog sučelja. Kako bi koristio aplikaciju, korisniku je potreban korisnički račun. Prvo je objašnjena registracija i prijava. Nakon toga, korisnik odabire klijenta i lokaciju koju koristi u aplikaciji. Prikazan je i pojašnjen proces kreiranja izvješća i pregled vatrogasnog aparata i ostalih korisnikovih aktivnosti po aplikaciji. Korisnik dobiva obavijest o uređajima koje treba pregledati u tipu lokalne notifikacije na svom uređaju.

5.1. Registracija i prijava korisnika

Slike Sl. 4.4. i Sl. 5.1. prikazuju forme za registraciju i prijavu korisnika. Forma za prijavu sadrži polje za unos korisničkog imena i lozinke, a forma za registraciju sadrži polje za unos imena, prezimena, elektroničke pošte, poštanskog broja, grada, adrese i telefona novog korisnika. Sva navedena polja moraju biti ispunjena kako bi se omogućile tipku za prijavu ili stvaranje zahtjeva za registraciju novog korisnika.

Sl. 5.1. *RegisterPage* – forma za registraciju novih korisnika

Nakon registracije, na navedenu adresu elektroničke pošte se pošalje poveznica s lozinkom koju

novi korisnik može koristiti za prijavu. Ukoliko je korisnik zaboravio lozinku, može zatražiti novu na formi za prijavu, a može i promijeniti lozinku u izborniku na ekranu s postavkama.

5.2. Uporaba aplikacije u ulozi kontrolera vatrogasnih aparata

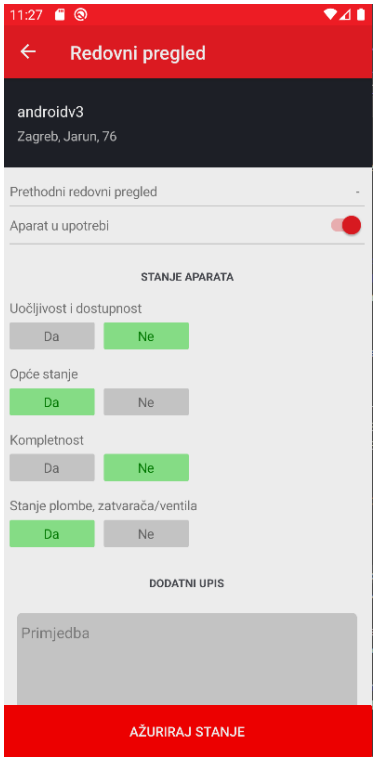
Nakon što se korisnik prijavi u sustav i odabere željenog klijenta i lokaciju s kojom želi raditi u aplikaciji, pronalazi se na glavnom ekranu u aplikaciji *RootPage* koji se može vidjeti na slici Sl. 4.5. Na vrhu stranice nalazi se informacija o aktivnom klijentu i vatrogasni aparati koji imaju zakazan pregled u idućih 30 dana, a ispod njih se nalazi lista svih vatrogasnih aparata na odabranoj lokaciji. Ukoliko postoje aparati na uređaju kojima je odrađen redovni pregled, prikazuje se posebna forma *RootPage-a* koja se može vidjeti na slici **Error! Reference source not found.**Klikom na bilo koji od vatrogasnih aparata na *RootPage-u* otvara se ekran *DeviceDetailsPage* na kojem korisnik odrađuje promjenu lokacije vatrogasnog aparata, redovni pregled uređaja i ima pregled prethodno odrađenih promjena na odabranom uređaju. *DeviceDetailsPage* se može vidjeti na slici **Error! Reference source not found.**Prilikom učitavanja uređaja za prikaz na ekranu *RootPage* okida se logika za postavljanje podsjetnika za uređaje koji imaju zakazan pregled u idućih 30 dana. Ako postoji takav uređaj, za njega će se sutradan u osam ujutro poslati lokalna notifikacija na njegovom uređaju za uređaj kojemu treba napraviti pregled i tip zakazanog pregleda. Primjer notifikacije za pregled uređaja može se vidjeti

Sl. 5.2. Primjer lokalne notifikacije za unutarnji pregled vatrogasnog aparata

na slici Sl. 5.2.

Kada korisnik otvori ekran *DeviceDetailsPage* i želi odraditi redovni pregled uređaja, klika na crnu tipku na izborniku tog *ContentPage-a* i otvara mu se *DeviceRegularExamPage*. Na njemu treba odabrati podatke o upotrebi aparata i odabrati različite parametre kojima se određuje stanje aparata. To su uočljivost i dostupnost, opće stanje, kompletnost i stanje plombe, zatvarača/ventila. Također može upisati primjedbu u redovnom pregledu ako je primijetio dodatan problem s aparatom. Forma za redovni pregled vatrogasnih *DeviceRegularExamPage* se može vidjeti na slici

Sl. 5.3. Dodatno, na ekranu *DeviceDetailsPage* korisnik može promijeniti podatke vezane uz lokaciju aparata i pregledati povijest radnji na njemu. Forma za promjenu lokacije se može vidjeti



Sl. 5.3. *DeviceRegularExamPage* – forma za redovni pregled vatrogasnih aparata

na slici 4.10., a forma za povijest radnji na aparatu na slici 4.9.

S ekrana *RootPage* koji predstavlja glavni dio aplikacije, korisnik može pristupiti gotovo svim funkcionalnostima u aplikacije poput pretrage aparata, povijesti radnji na aparatima, kalkulatoru jedinica gašenja, postavkama, kontakt informacijama proizvođača aparata i izradi izvješća za odrađene preglede na uređajima. Na slici **Error! Reference source not found.**se može vidjeti izgled aplikacije kada je korisnik odradio pregled na nekom od uređaja. Kada želi napraviti izvješće, treba odabrati tipku izradi izvješće, nakon čega mu se otvara izgeneriran PDF dokument s aparatima na kojima je napravio redovan pregled.

5.3. Osvrt na rad aplikacije

Predstavljeno rješenje mobilne aplikacije zadovoljava sve definirane zahtjeve za upravljanje sustavom vatrogasnih aparata. Budući da aplikacija koristi arhitekturu *Model-View-ViewModel* (MVVM), lako je proširiva te je jednostavno prilagoditi postojeće funkcionalnosti. U kontekstu upravljanja vatrogasnim aparatima, predstavljeno rješenje zadovoljava sve potrebe korisnika koji se bavi pregledom vatrogasnih aparata. Važna stavka upravljanja vatrogasnim aparatima je

sigurnost, valjanost i pravovremenost dostupnih informacija. Postoji mnogo prethodno dostupnih rješenja te problematike. Ukoliko se kreirano rješenje mobilne aplikacije želi koristiti uz dodatne funkcionalnosti i proširenje sigurnosnih zahtjeva, moguća je laka proširivost njenih funkcionalnosti i nadogradnju sigurnosti.

6. ZAKLJUČAK

U ovome diplomskom radu razvijeno je programsko rješenje mobilne aplikacije za upravljanje vatrogasnim aparatima. Aplikacija predstavlja sustav za pregled vatrogasnih aparata i različitim aktivnosti vezane za njihovo upravljanje. Prilikom istraživanje teme, analizirana su postojeća rješenja slične tematike. Uz pregled postojećih rješenja, istraženi su aktualni trendovi u području upravljanja vatrogasnim aparatima. Analizirana su novija rješenja koja se bave tom tematikom kao i funkcionalnosti koje one pružaju korisnicima.

Mobilna aplikacija koristi *Model-View-ViewModel* arhitekturu. Aplikacija je izrađena u tehnologiji Xamarin.Forms. Za komuniciranje s postojećim API-jem proizvođača vatrogasnih aparata, koriste se različiti HTTP servisi koji se ubrizgavaju u *ViewModele* i koriste *Dependency injection*. Prikazan je tijek korištenja aplikacije kroz različite faze korisnikovih aktivnosti. Nakon prolaska kroz implementacijski dio, prikazano je korištenje aplikacije od strane korisnika. Objašnjene su njihove mogućnosti koje imaju u različitim fazama korištenja aplikacije. Analizirana je okolina u slučaju korištenja valjane autentifikacije korisnika u sustav.

LITERATURA

[1] Streamline Inspections, Informacije o aplikaciji, dostupno na: https://streamlineas.com/ [3.6.2023.]

[2]Streamline Inspections, Korisničko sučelje, dostupno na: https://apps.apple.com/us/app/streamline-inspections/id964653980 [27.6.2023]

[3]First Due, Informacije o aplikaciji, dostupno na: https://www.firstdue.com/ [3.6.2023.]

[4]First Due, Korisničko sučelje, dostupno na: https://www.firstdue.com/products/mobileresponder [27.6.2023]

[5]Fire Mate, Informacije o aplikaciji, dostupno na: https://firemate.com/ [3.6.2023.]

[6]Fire Mate, Korisničko sučelje, dostupno na: https://firemate.com/ [27.6.2023]

[7]Fire Extinguisher Inspection, Informacije o aplikaciji, dostupno na: https://play.google.com/store/apps/details?id=ie.imec.technology [3.6.2023.]

[8]Fire Inspection App, Informacije o aplikaciji, dostupno na: https://apps.apple.com/us/app/fire-inspection-app/id860747212 [3.6.2023.]

[9]Fire Extinguisher Inspection, Korisničko sučelje, dostupno na: https://play.google.com/store/apps/details?id=ie.imec.technology [27.6.2023.]

[10]Fire Inspection App, Korisničko sučelje, dostupno na: https://apps.apple.com/us/app/fire-inspection-app/id860747212 [27.6.2023.]

[11]C. Petzold, Creating Mobile Apps with Xamarin.Forms, Microsoft Press, Washington, 2016.

[12]Microsoft, Osnove o okviru Xamarin.Forms, dostupno na: https://learn.microsoft.com/en-us/xamarin/get-started/what-is-xamarin-forms [3.6.2023.]

[13]Microsoft, Dokumentacija okvira Xamarin.Forms, dostupno na: https://dotnet.microsoft.com/en-us/apps/xamarin/xamarin-forms [3.6.2023.]

[14]Microsoft, Informacije o NuGet paketima, dostupno na: https://learn.microsoft.com/en-us/nuget/what-is-nuget [3.6.2023.]

[15]PdFSharp.Xamarin.Forms, Informacije o NuGet paketu, dostupno na: https://github.com/akgulebubekir/PDFSharp.Xamarin.Forms [3.6.2023.]

[16]PdFSharp.Xamarin.Forms, Primjer generiranja PDF dokumenta s Githuba, dostupno na: https://github.com/akgulebubekir/PDFSharp.Xamarin.Forms [27.6.2023.]

[17]Rg.Plugins.Popup, Primjer *popup* stranica u Xamarin.Forms aplikaciji, dostupno na: https://github.com/rotorgames/Rg.Plugins.Popup [27.6.2023.]

[18]Microsoft, Dokumentacija alata Xamarin.Community.Toolkit, dostupno na: https://learn.microsoft.com/en-us/xamarin/community-toolkit/ [3.6.2023.]

[19]E. Snider, Mastering Xamarin.Forms, Packt Publishing, siječanj, 2016.

[20]Dashlane, Migracija Android arhitekture na *MVVM*, dostupno na: https://www.dashlane.com/blog/android-ui-architecture-mvvm [3.6.2023.]

[21]S. van Deursen, M. Seemann, Dependency Injection Principles, Practices, and Patterns, Manning Publications, ožujak, 2019.

SAŽETAK

Teorijski dio rada obuhvaća izazove u upravljanju vatrogasnim aparatima. Prikazuju se i aktualna programska rješenja za navedenu problematiku, sukladno čemu je razvijena mobilna aplikacija za upravljanje vatrogasnim aparatima kao zadatak ovog diplomskog rada. Aplikacija je implementirana korištenjem tehnologije Xamarin.Forms. Na primjeru korištenja objašnjeni su ključni dijelovi implementacije. U programskom jeziku C# i korištenjem dostupnih NuGet paketa, dodane su funkcionalnosti generiranja PDF dokumenta, skeniranja barkoda i pristupu kartama. Nakon analize implementacije, prikazan je i analiziran rad aplikacije od strane njenog korisnika.

Ključne riječi: mobilna aplikacija, NuGet paketi, Xamarin.Forms

ABSTRACT

Mobile aplication for managing fire extinguishers

The theoretical part of the thesis covers the challenges in managing fire extinguishers. Current software solutions for the mentioned issues are presented, based on which a mobile application for fire extinguisher management was developed as the task of this master's thesis. The application is implemented using Xamarin.Forms technology. The key implementation parts are explained using available examples. Functionalities such as generating PDF documents, barcode scanning, and map access were added using the C# programming language and available NuGet packages. After analyzing the implementation, the application's performance was demonstrated and evaluated by its user.

Keywords: mobile application, NuGet packages, Xamarin.Forms

ŽIVOTOPIS

Matej Peršić rođen je 20.8.1999. godine u Našicama. Završio je Osnovnu školu Josipa Kozarca Slatina. Godine 2018. završio je Srednju školu Marka Marulića Slatina, smjer opća gimnazija. Iste godine upisao je Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek. Godine 2021. upisuje sveučilišni diplomski studij računarstva, smjer informacijske i podatkovne znanosti, na istom fakultetu, kojeg završava 2023. godine. Stručnu praksu odradio je u IT tvrtki COBE te je tijekom studiranja na diplomskom studiju bio zaposlen kao programer mobilnih aplikacija u osječkoj tvrtki MONO.

_____

Potpis autora