

**SVEUČILIŠTE U ZAGREBU
FAKULTET ORGANIZACIJE I INFORMATIKE
V A R A Ž D I N**

Tim AIR1916

FOI Knjižnica

PROJEKT IZ KOLEGIJA „ANALIZA I RAZVOJ PROGRAMA“

Varaždin, 2020.

SVEUČILIŠTE U ZAGREBU
FAKULTET ORGANIZACIJE I INFORMATIKE
V A R A Ž D I N

Antonio Kudelić
Marin Mačinković
Matej Lipovača
Matija Ivanić
Stiven Drvoderić

GitHub repozitorij: <https://github.com/AIR-FOI-HR/AIR1916>

ZenHub: <https://app.zenhub.com/workspaces/air1916-5db63d487e2344000155eec5/board?repos=217938608>

Vsdocman : Nalazi se u folderu „Dokumentacija“ na githubu ([VSDoc -> index.html](#))

FOI Knjižnica – Tehnička dokumentacija

PROJEKT IZ KOLEGIJA „ANALIZA I RAZVOJ PROGRAMA“

Mentor:

Doc. dr. sc. Boris Tomaš

Sadržaj


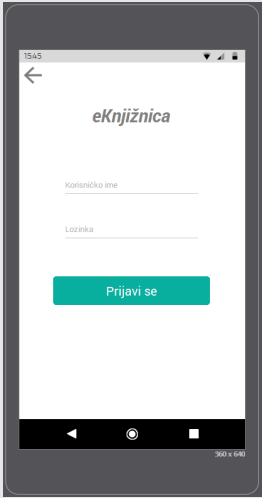
1. Uvod.....	1
2. Popis funkcionalnosti.....	2
Karakteristike korisnika	8
3. WireFrame	9
4. UseCase dijagram	13
5. Arhitektura aplikacije.....	14
FOI CAS server.....	14
Azure – API	14
Azure – Baza podataka	14
Azure – Function App.....	15
Azure – Notification Hub.....	15
Aplikacija.....	15
Ograničenja aplikacije	15
6. Baza podataka	16
7. UML dijagram klasa	18
8. Funkcionalnosti sustava	19
Prijava	19
FOI Prijava.....	19
Modularna prijava (brzi načini)	20
Pregled literature	22
Sortiranje literature	23
Filtriranje literature	23
Pretraga	24
Detaljne informacije.....	24
Pregled sadržaja	24
Rezervirane knjige	24
Posudba knjige koristeći QR ili NFC.....	25
Korisnički račun.....	25
Notifikacije	26
Podsjetnik o slobodnoj knjizi.....	26
Notifikacija o isteku rezervacije ili roka vraćanja knjige	27
9. Xamarin.....	28
Instaliranje Xamarina.....	28
Rad u Visual Studio Xamarin	30


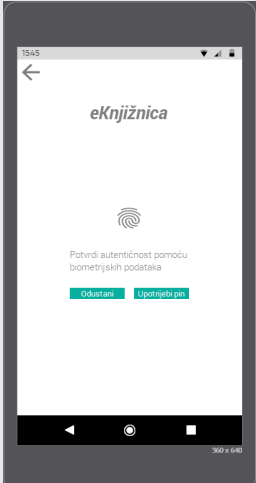

Način izrade programskog rješenja u Xamarinu.....	31
Karakteristika pokretanja aplikacije (FOI Knjižnica).....	31
10. BurnDown grafovi	33
Radni satovi za izradu projekta.....	33
Prvi sprint.....	34
Drugi sprint	34
Treći sprint	35
Četvrti sprint	36

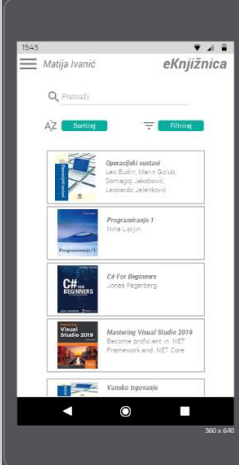
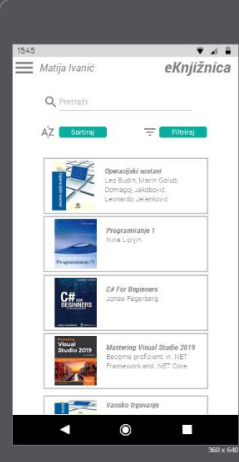
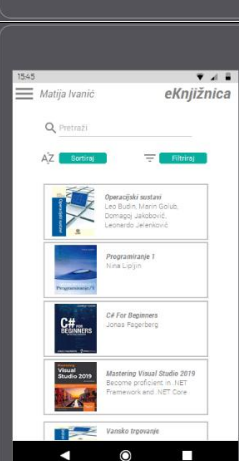
1. Uvod

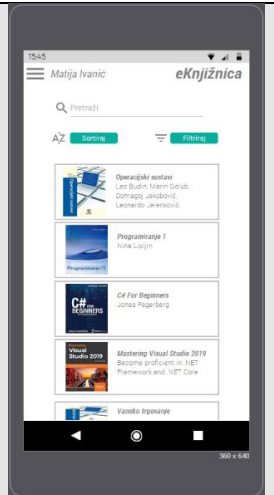


Kroz ovaj uvod ujedno će biti opisana sama domena aplikacije kako bi pobliže objasnili projekt koji smo zamislili. Aplikacija je namijenjena svima onima koji imaju pravo autentifikacije putem AAI@EduHr sustava te bi željeli posuditi knjigu iz FOI knjižnice. Vrsta aplikacije će biti Android te će se sastojati od nekoliko ključnih funkcionalnosti koje bi olakšale svakodnevno rezerviranje i posudbe unutar knjižnice. Valja naglasiti da je ovo neovisan projekt te nemamo pristup stvarnim podacima iz FOI knjižnice te će podaci unutar aplikacije biti testne naravi. Preko korisničkog sučelja bila bi omogućena inicijalna prijava putem AAI@EduHr sustava nakon čega bi se korisnicima ponudila opcija da se ubuduće prijavljuju koristeći pin, uzorak ili otisak prsta. Nakon prijave korisnici bi bili u mogućnosti pregledavati katalog knjižnice (sortiranje, filtriranje, pretraga i sl.) te prema želji rezervirati knjigu ako je ona slobodna. Rezervacija bi se slala putem e-maila knjižničarima te bi ista vrijedila određeni vremenski period. Ukoliko bi publikacija bila podignuta u tom periodu novo stanje dostupnosti bi bilo vidljivo svim korisnicima na korisničkom sučelju, u suprotnom rezervacija bi bila otkazana. Ukoliko bi korisnici prikazali određeni interes za neku publikaciju te ju stavili u vlastite favorite, dobili bi notifikaciju o raspoloživosti iste kako bi ju mogli rezervirati i posuditi. Razlog radi kojeg bi izradili ovakvu vrstu aplikacije je upravo taj što ovakva mobilna aplikacija još ne postoji za FOI knjižnicu (osim web stranice koja ne posjeduje sve mogućnosti koje bi bile implementirane unutar ove aplikacije) te smatramo da bi uvelike mogla olakšati proces posudbe knjižničarima, ali i korisnicima.


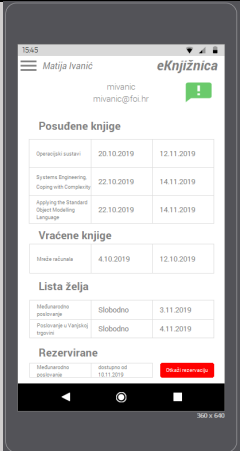
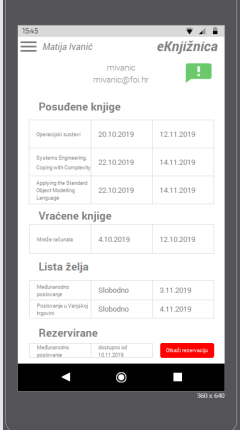
2. Popis funkcionalnosti

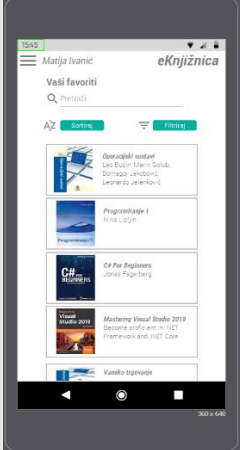
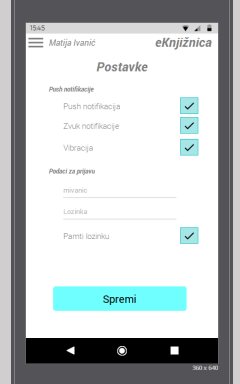


Broj funkc.	Naziv funkcionalnosti	Opis funkcionalnosti	Slika ekrana
1.	Prijava (modularno)	Mogućnost prijave korisnika u sustav sa svojim podacima za prijavu. Prijava na svoj račun putem tuđeg mobilnog uređaja/uređaja na kojim nisu napravljeni brzi načini prijave bit će moguća samo uz korisničko ime i lozinku, a prijava brzim načinom samo na svojem mobilnom uređaju/uređaju na kojem su napravljeni brzi načini prijave.	
1.1	Prijava (korisničko ime i lozinka)	Prijava putem korisničkog imena i lozinke je osnovna vrsta prijave koju korisnik uvijek može koristiti u slučaju da ostale vrste ne rade ili ako korisnik to preferira. Kada se korisnik prvi puta prijavi sa svojim <u>AAI@edu.hr</u> podacima u našu bazu će se povući njegovi podaci i kreirati njegov korisnički profil u našoj bazi te će se korisniku u budućnosti omogućiti prijava sa tim podacima unutar same aplikacije.	


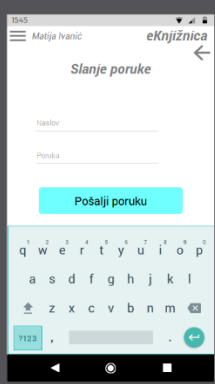
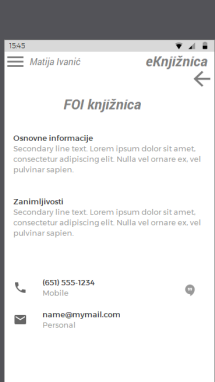
1.2	Prijava (PIN)(modularno)	Prijava putem PIN koda bit će jedna od opcija za bržu prijavu u sustav. Kako bi korisnik postavio PIN za prijavu mora se najprije prijaviti u aplikaciju sa svojim korisničkim imenom i lozinkom te jedanput kada je prijavljen može unutar svojeg profila aktivirati mogućnost prijave putem PIN-a .	
1.3	Prijava (Otisak prsta)(modularno)	Prijava putem otiska prsta bit će opcija brze prijave za korisnike koji imaju čitač otiska prsta na svojem mobilnom uređaju. Kada se korisnik prijavi u sustav može u unutar svojeg profila aktivirati mogućnost korištenja otiska prsta te će mu se u budućnosti nuditi ta opcija za prijavu.	
1.4	Prijava (Uzorak)(modularno)	Prijava putem uzorka je opcija brze prijave korisnika u sustav na način da u polju točaka 3x3 napravi svoj uzorak i ako je on točan korisnik se prijavljuje u sustav. Korisnik svoj uzorak aktivira u svojem računu i specificira izgled uzorka kojeg će koristiti.	

2.	Pregled literature	Osnovni pregled literature u obliku popisa sa slikom naslovnice knjige, informacijama o autorima, godini izdanja, izdavačkoj kući te mogućnošću otvaranja ekrana sa detaljima o knjizi.	
2.1	Filtriranje literature	Mogućnost filtriranja literature prema prvom slovu naslova, autoru, razdoblju izdavanja, izdavaču itd. Za opcije filtriranja otvara se novi ekran gdje se postavljaju željeni parametri te se nakon toga otvara opet pregled literature.	
2.2	Sortiranje literature	Mogućnost sortiranja literature po abecednom redu, godini izdavanja, autoru itd.	

2.3	Pretraga	Mogućnost pretrage baze podataka literature prema unesenoj ključnoj riječi bio to dio naziva knjige, autoru, izdavačkoj kući itd.	
2.4	Detaljne informacije	Poseban ekran koji se otvara pritiskom na željenu literaturu kod pregleda literature. Pregled detaljnih informacija sastoji se od kratkog opisa sadržaja knjige, osnovnih informacija o knjizi (autori, broj stranica, godina izdavanja...), informacija o dostupnosti pojedine instance knjige itd.	
2.5	Pregled sadržaja	Pritiskom na gumb korisniku se otvara novi ekran koji sadrži listu sadržaja kako bi se korisniku bolje predložila knjiga.	

2.6	Rezerviranje knjige	Aplikacije daje korisniku mogućnost rezerviranja dostupne knjige pritiskom na gumb te se stanje knjige mijenja u rezervirano, nakon čega korisnik podiže knjigu u knjižnici u zadanome roku te se time mijenja stanje knjige u zauzeto.	
3.	Korisnički račun	Ekran korisničkog računa sadrži informacije o trenutno posuđenim, rezerviranim i prethodno posuđenim knjigama. Također daje mogućnost otvaranja detaljnih informacija.	
3.1	Prekidanje rezervacije	Mogućnost prekidanja trenutno rezervirane knjige.	

3.2	Favoriti	Mogućnost da korisnik doda neku knjigu kao favorita te primi notifikacije o dostupnosti knjige	
4.	Postavke/Notifikacije	Ekran koji nudi mogućnost odabira notifikacija koje korisnik želi primati.	
4.1	Podsjetnik o slobodnoj knjizi	Notifikacija koja korisniku daje obavijest o tome da se oslobodila neka knjiga koju je on odredio da ga zanima.	
4.2	Notifikacija o isteku rezervacije ili roka vraćanja knjige	Notifikacija koja daje korisniku informaciju da će za neko vrijeme isteći njegova rezervacija ili da će isteći rok posudbe.	

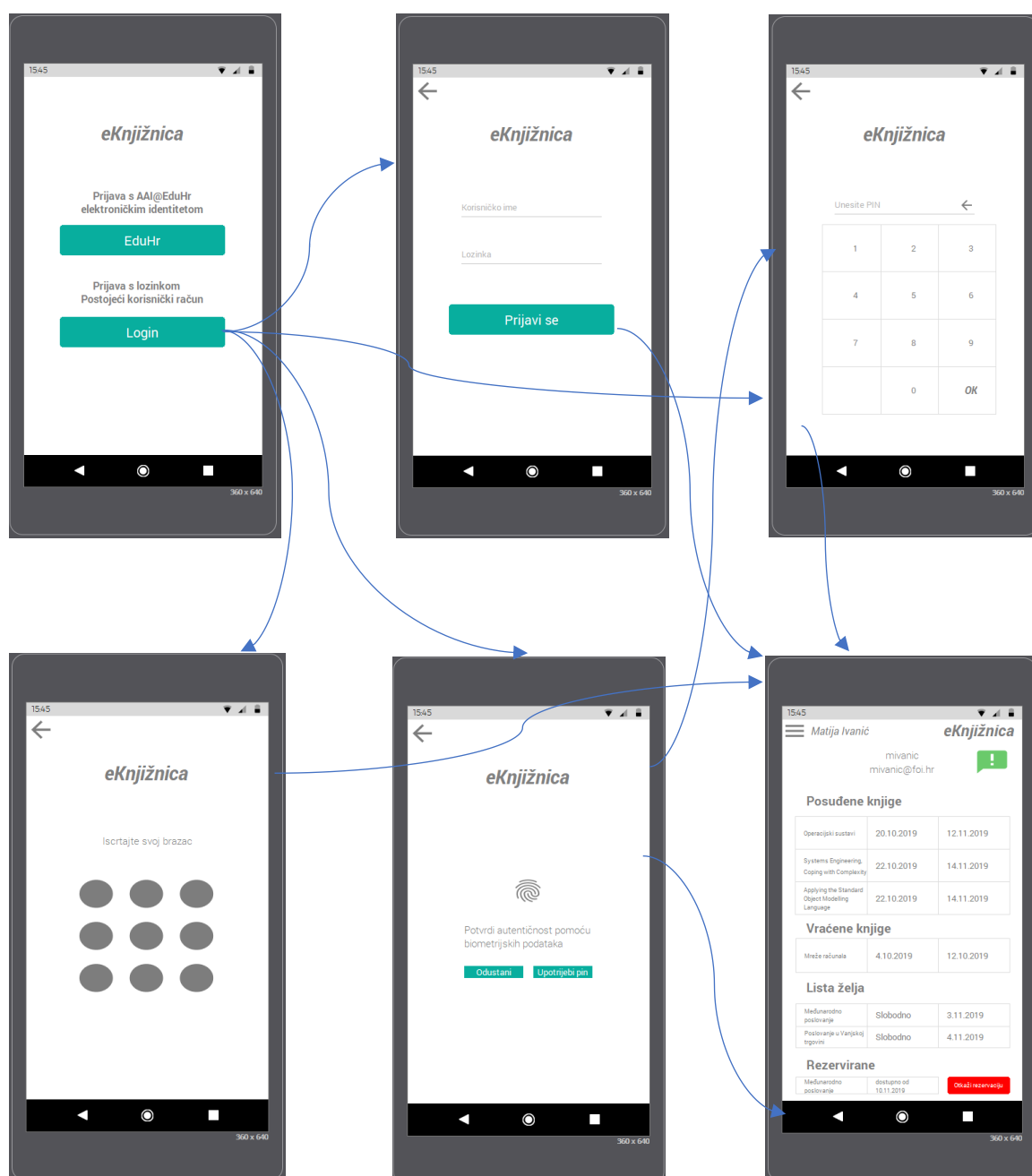
5.	Pomoć u sustavu	Pomoć korisniku pritiskom na upitnik u gornjem kutu ekrana.	
6.	Slanje poruke	Slanje poruke na e-mail adresu knjižnice.	
7.	Osnovne informacije o knjižnici	Ekran sa radnim vremenom te kontakt informacijama knjižnice.	

Karakteristike korisnika

Aplikacija je namijenjena svima koji imaju pristup FOI autentikaciji i željeli bi rezervirati/posuditi neku publikaciju iz knjižnice. Bitno je naglasiti da je korisnik samo onaj koji posuđuje, ali ne i knjižničar/knjižničarka.

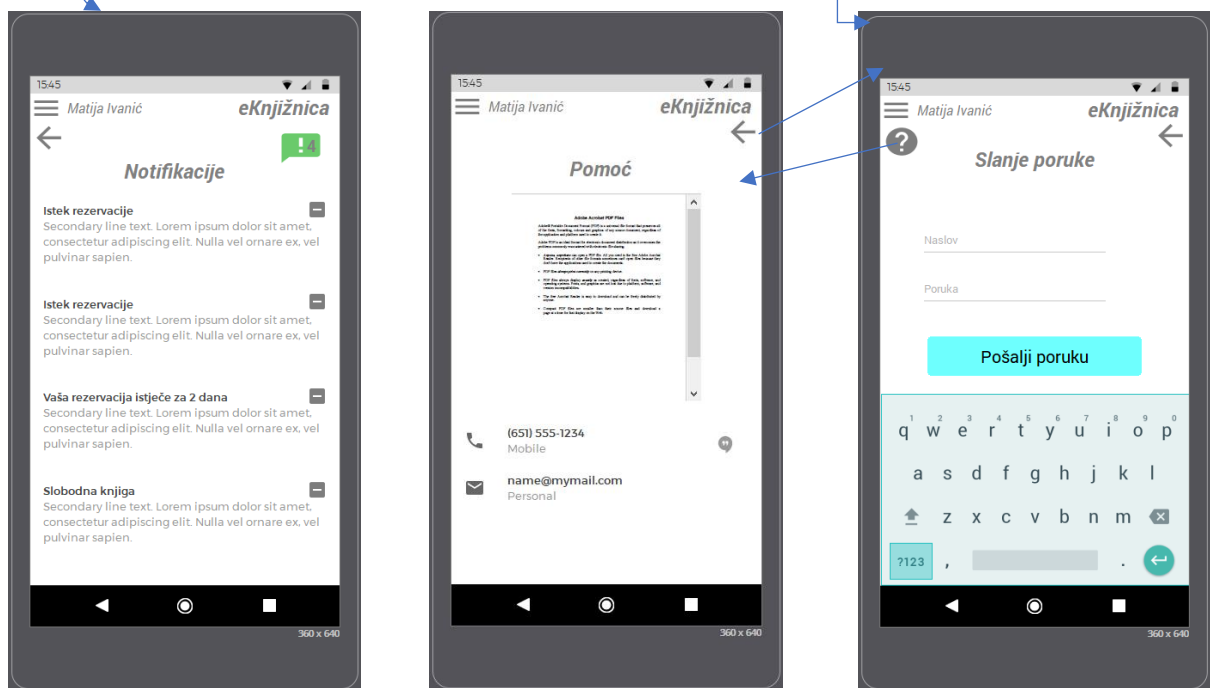
3. WireFrame

Početni prikaz omogućava nam da odaberemo mogućnost prijave. Ukoliko se prijavljujemo prvi puta, onda ćemo za odabir odabrati gumb „EduHr“ s kojim ćemo se prijaviti na eduHr, te potvrditi naš status kao studenta FOI, te nam omogućiti prijavu u aplikaciju. Ukoliko smo se već prijavili, te je aplikacija spremila našu prijavu i naš korisnički račun, onda za lakšu i bržu prijavu možemo odabrati drugi gumb, „Login“. S ovim gumbom odlazimo do ekrana s kojim se prijavljujemo u aplikaciju. Ovisno o našem izboru, to može biti preko korisničkog imena i lozinke, otiska prsta, pin zaporka ili crtanje uzorka, odnosno obrazac.

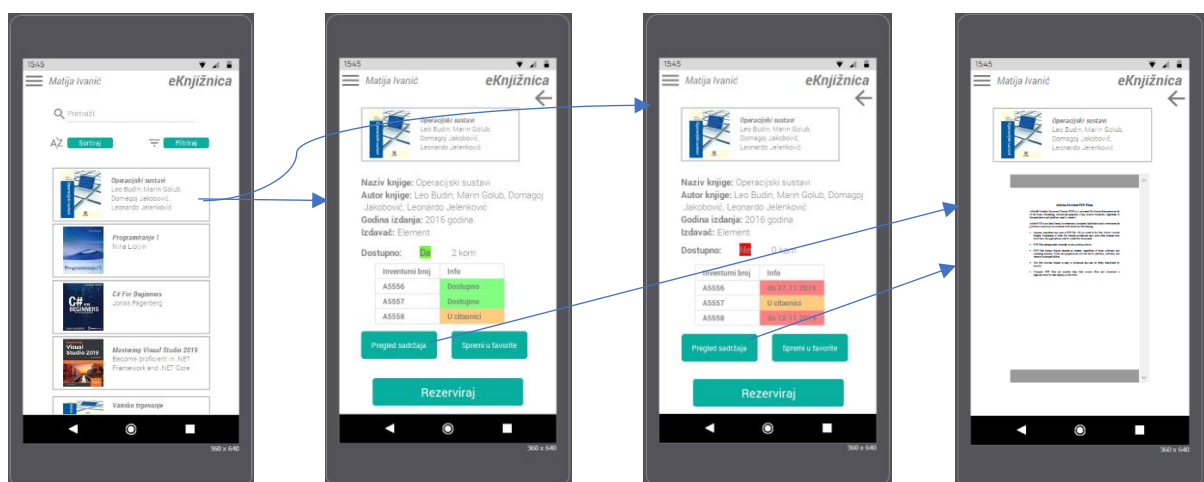


Ukoliko smo se uspješno prijavili, otvara nam se početni zaslon, odnosno naš korisnički račun, u kojem možemo vidjeti detaljnije informacije o našim posuđenim knjigama. Isto tako u listi prijavljenih rezervacija, možemo otkazati našu rezervaciju, odabirom gumba „Otkazi rezervaciju“. U gornjem lijevom kutu nalazi se ikonica izbornika. Pritiskom na ikoncu otvara nam se izbornik. Odabirom određenog gumba, odlazimo do sljedećeg ekrana.

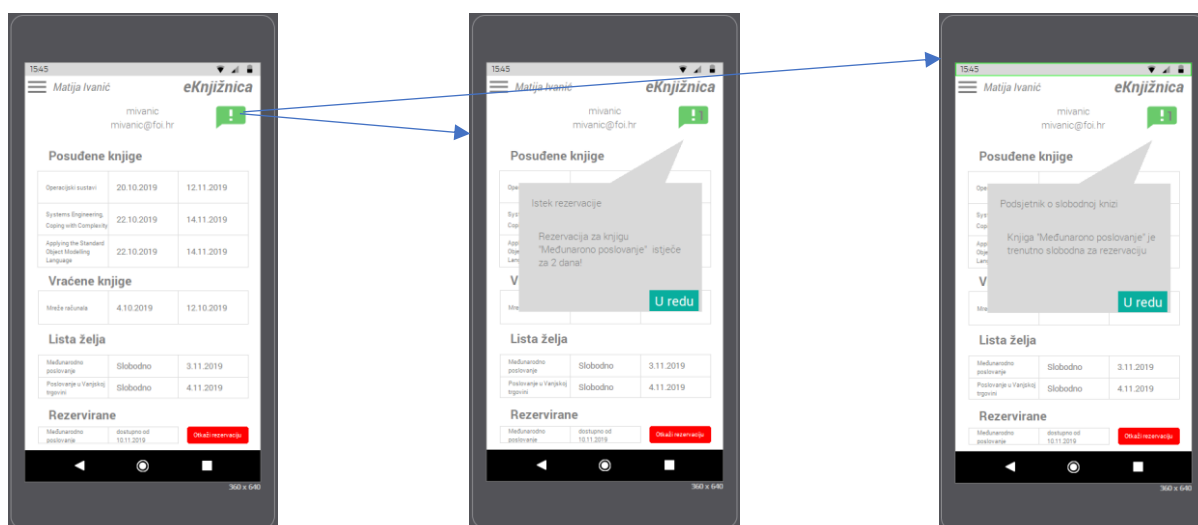




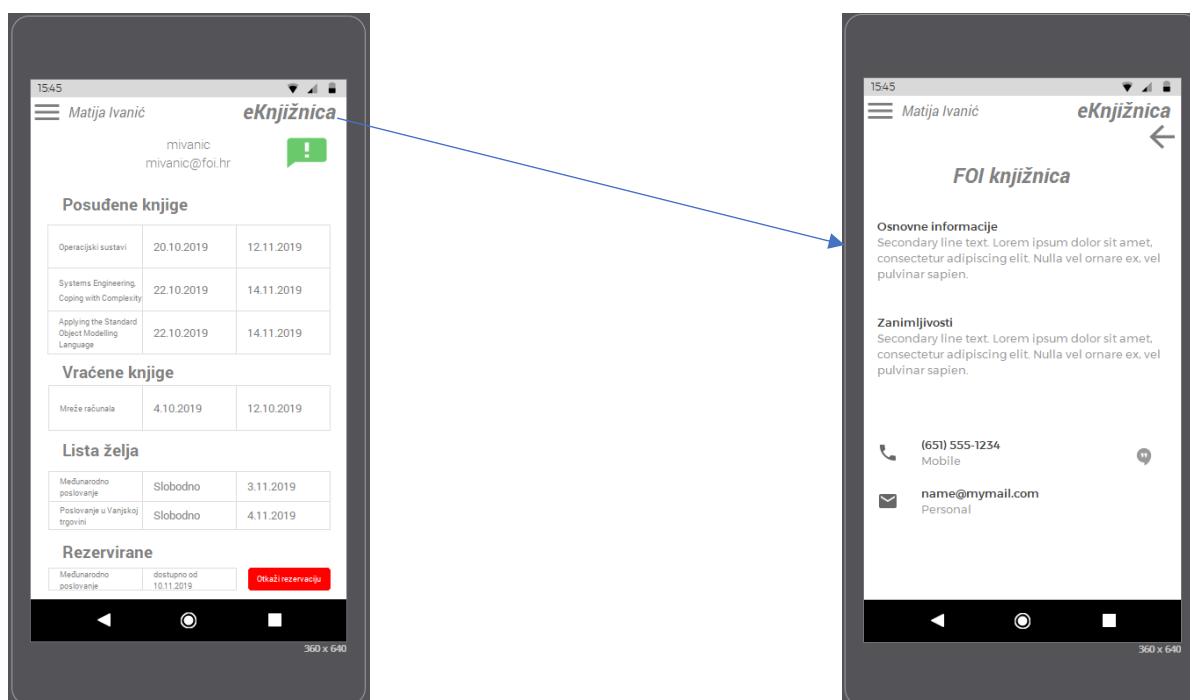
U ekranu pretraživanje, možemo prelistati cijeli katalog knjižnice, ali isto tako unosom određenog pojma možemo pretražiti, sortirati, ali i filtrirati. Klikom na odabranu knjigu, otvara nam se detaljni prikaz knjige, gdje možemo vidjeti sve pojedinosti o toj knjizi, te odabirom gumba „Prikaz sadržaja“, možemo vidjeti nekoliko stranica iz knjige. Ovdje imamo mogućnost rezervacije knjige ukoliko je slobodna, te spremiti u favorite. Pomoć u sustavu osmišljena je na način da nam se u svakom prozoru pojavi znak „?“, upitnika. Klikom na upitnik otvara nam se prozor sa pomoć u sustavu za određeni problem. Tako da korisniku olakšamo snalaženje, te mu omogućimo jednostavnost. Primjer ovoga možemo vidjeti na slici prozora „Slanje poruke“, gdje imamo upitnik, te klikom na njega otvaramo pomoć. Za vraćanje nazad, samo pritisnemo znak strelice unatrag.



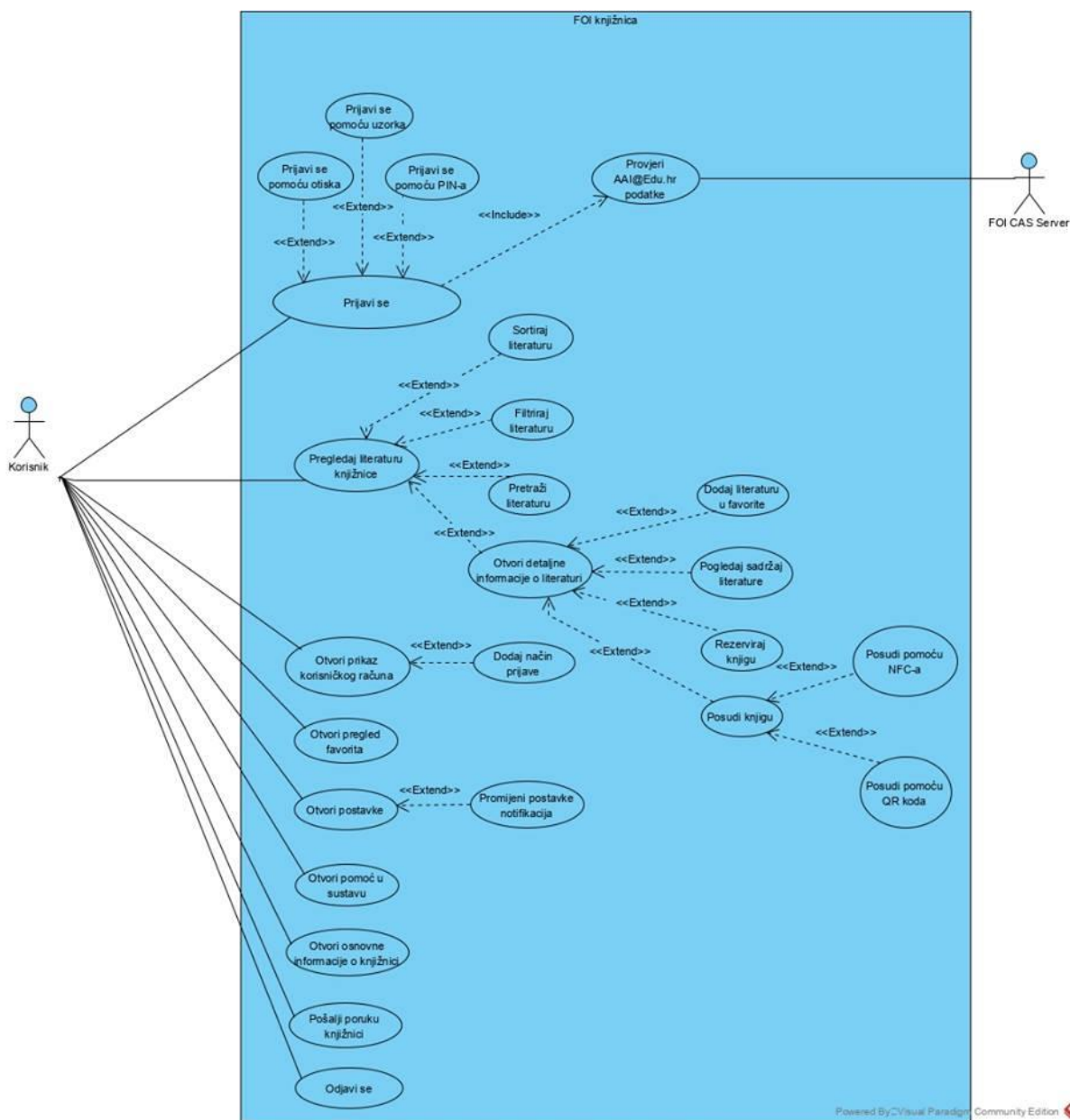
Ukoliko dobijemo notifikaciju, nju možemo vidjeti u notifikacijama, ali isto tako možemo vidjeti novu nepročitanu notifikaciju na našem profilu klikom na ikonicu notifikacije.



Ukoliko želimo vidjeti osnovne informacije o knjižnici, to možemo vidjeti klikom na logo „eKnjižnica“, koja nam otvara poseban ekran sa svim detaljnim informacijama o FOI knjižnici, te kontakt. To možemo učiniti na bilo kojem ekranu, gdje je prikazan logo „eKnjižnica“.



4. UseCase dijagram



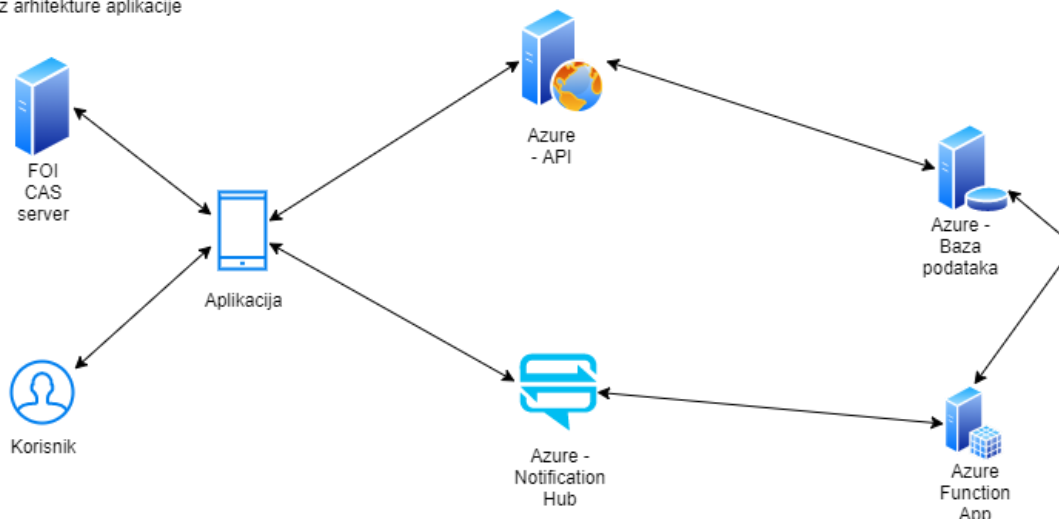
Na slici gore prikazan je UseCase dijagram za aplikaciju Foi knjižnica na kojoj su prikazani svi mogući slučajevi korištenja aplikacije . Potrebno je napomenuti da je slučaj prijave u aplikaciju potreban za izvađanje ostalih slučajeva, ali nije prikazano na slici zbog preglednosti.

5. Arhitektura aplikacije

Na slici ispod vidljiva je arhitektura „FOI Knjižnica“ aplikacije. Vidljivo je da se aplikacija temelji na funkcionalnosti nekoliko različitih funkcionalnih cjelina koji će biti objašnjeni u nastavku.

Arhitektura Aplikacije FOI Knjižnica

Prikaz arhitekture aplikacije



FOI CAS server

Riječ je o FOI serveru koji nam omogućuje našu inicijalnu prijavu u aplikaciju. Temelji se na svojoj vezi sa *AAI@EduHr* sustavom koji predstavlja autentikacijsku i autorizacijsku infrastrukturu znanosti i visokog obrazovanja u Republici Hrvatskoj. Nakon uspješne prijave našeg korisnika u sustav, odnosno inicijalne registracije, aplikacija od FOI CAS servera, koristeći **DotNetCasClient** dobije osnovni podatak o prijavljenom korisniku koji se potom zapisuje u bazu podataka na našoj strani.

Azure – API

Sva komunikacija između baze podataka i aplikacije, koja je vezana uz dohvaćanje i ažuriranje zapisa (u našem slučaju – Publikacija) obavlja se preko Web API-a implementiranog u ASP.NET-u koristeći „*Api Controller*“ klasu.

Azure – Baza podataka

Naša baza podataka na Azure servisu koja nam služi za pohranu svih potrebnih podataka.

Azure – Function App

Azure Function App u projektu nam služi, zajedno s *Notification Hubom*, za slanje obavijesti korisniku o isteku rezervacije/posudbe određene publikacije. Također, koristi se još i za automatsko ukidanje isteklih rezervacija.

Azure – Notification Hub

Azure Notification Hub nam omogućuje slanje obavijesti na mobitel korisnika, a temelji se na izvršenju *time trigger-a* sa *Function App-a*.

Aplikacija

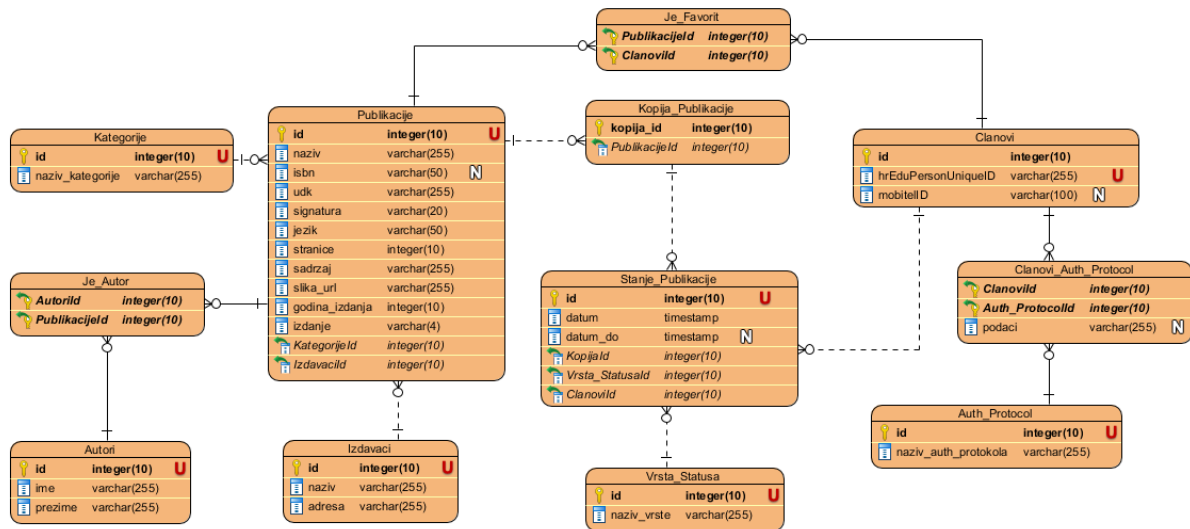
Sama aplikacija se sastoji od nekoliko projekata: **FOIKnjiznica**, **FOIKnjiznica.Android**, **FOIKnjiznicaWebServis** i modula za prijavu.

- *FOIKnjiznica* – Sadrži sve ekrane i potrebne klase za rad aplikacije
- *FOIKnjiznica.Android* – Sadrži *activitye* koji se pokreću prilikom pokretanja aplikacije
- *FOIKnjiznicaWebServis* – Sadrži kontrolere preko kojih komuniciramo s bazom podataka te nam omogućuje prijavu koristeći *FOI login*.

Ograničenja aplikacije

Aplikacije je ograničena na korisnike mobilnih uređaja s Android operativnim sustavom koji moraju imati nekakav oblik konekcije na Internet kako bi pristupili našoj aplikaciji – potrebno se spojiti na bazu podataka, tj. WebAPI-e. Također, kako je navedeno i u potpoglavlju 2.1., aplikacija je namijenjena isključivo osobama s pristupom FOI autentikaciji.

6. Baza podataka

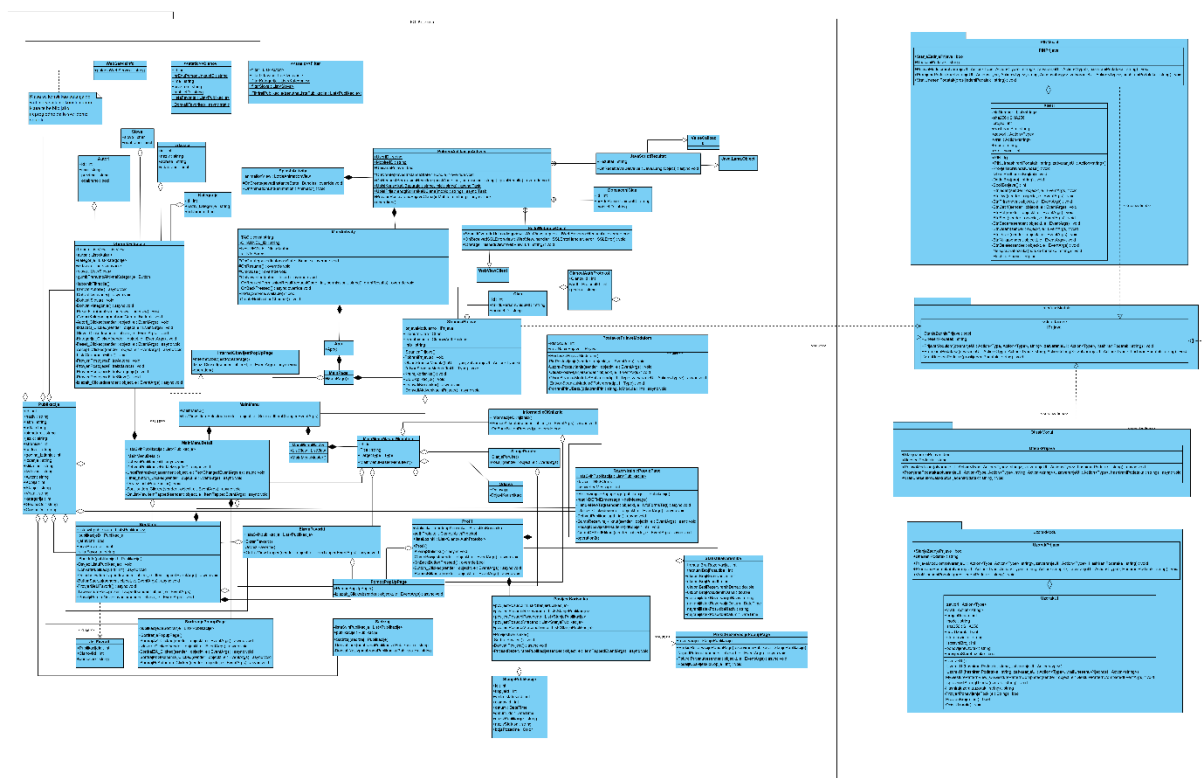


Na slici iznad nalazi se osnovni ERA model naše aplikacije. U nastavku će biti opisane uloge svake od tablica kako bi model bio jasniji.

1. **Publikacije** – Tablica koja se odnosi na sve publikacije koje se u knjižnici nalaze te sadrži sve osnovne informacije o njima.
2. **Kategorije** – Kategorije predstavljaju vrstu publikacije jer publikacija može biti knjiga ili časopis npr.
3. **Autori** – Tablica sadrži imena svih autora
4. **Je_Autor** – Tablica koja sadrži podatke o tome tko je autor koje publikacije (međutablica)
5. **Izdavači** – Tablica sadrži popis izdavača
6. **Članovi** - Tablica koja sadrži osnovne podatke o članovima koji imaju pristup posuđivanju knjiga. Tablica je napravljena na temelju imeničke sheme AAI@EduHr sustava koja se može pronaći na: https://www.aaiedu.hr/o-sustavu/imenicke-sheme/shema?fbclid=IwAR01PQ5iwTHi1ALza5MqguzVi1hE9_t5dOCQUkLB_NZHGQBiZbBBMoskmwWo. Riječ je o osnovnim podacima koji bi se pohranili prilikom inicijalne prijave u sustav koje ćemo onda koristiti kako bi omogućili nesmetan rad korisnika u aplikaciji.
7. **Je_Favorit** – Omogućuje članovima odabir publikacija koje favorizira
8. **Kopija_Publikacije** – Tablica koja služi za evidenciju svih kopija neke publikacije jer svaka ima određeni broj kopija koji se nalazi u knjižnici.

9. **Stanje_Publikacije** – Tablica koja nam služi za dohvaćanje podatka o određenoj kopiji publikacije. Odnosno, omogućuje nam evidenciju rezervacija, posudbi, vraćenih publikacija i sl. Također, stupac te tablice „datum_do“ bi se popunjavao isključivo prilikom posudbe publikacije kako bi se mogla voditi evidencija do kad je neka publikacija posuđena.
10. **Vrsta_Statusa** – Tablica koja služi za pohranu mogućih statusa neke publikacije (slobodna, rezervirana, posuđena).
11. **Auth_Protocol** – Tablica koja je vezana uz modularni dio aplikacije te koristi za evidenciju mogućih načina prijave korisnika u aplikaciju (pin, otisak prsta, uzorak).
12. **Članovi_Auth_Protocol** – Međutablica između Auth_Protocol i Članovi tablica putem koje vodimo evidenciju o tome što je pojedini član odabrao za način prijave u sustav.

7. UML dijagram klase



Na slici iznad može se vidjeti UML dijagram klase za aplikaciju FOI Knjižnica. Slika predstavlja trenutni izgled dijagrama klase te će se dijagram mijenjati kroz vrijeme dodavanjem i mijenjanjem klasa u samoj aplikaciji. Na trenutnome dijagramu mogu se vidjeti klase potrebne za prvi sprint, a to su klase vezane uz same publikacije. Klase vezane uz publikacije su Kategorija, Autor, Izdavac, Publikacija, KopijaPublikacije, StanjePublikacije te vrsta statusa. Te klase poslužit će za izradu glavnog ekrana na kojem će se prikazivati rezultati nekog filtriranja, pretrage i sortiranja. Osim tih klasa za prvi sprint također je prikazana klasa BazaPodataka koja će služiti za povezivanje aplikacije sa bazom podataka. Klasa će sadržavati neke preddefinirane SQL upite kao funkcije klase te će korisniku omogućiti i unos vlastitog SQL upita. Zadnja stvar koja je prikazana na UML dijagramu je modularni dio aplikacije, a to je sučelje kojeg implementiraju prethodno navedeni moduli.

8. Funkcionalnosti sustava

U nastavku će biti opisane neke istaknute funkcionalnosti sustava koje su ključne za funkcioniranje aplikacije. Uz navedene ispod, postoje i još neke poput slanja emaila (upita) knjižnici, provjera informacija o knjižnici, odjava i slično što nije toliko komplicirano samo po sebi osim specifičnosti odjave da se obriše *id mobitela* iz baze kako bi se korisnik, ako želi, mogao prijaviti s nekog drugog mobitela (FOI Prijava).

Prijava

Da bi korisnik mogao koristiti aplikaciju, nužno je da izvrši prijavu. Prvi puta kad se korisnik prijavi, obavezno mora prijavu obaviti preko FOI EDU verifikacije, jer je ova aplikacija namijenjena samo korisnicima Fakulteta organizacije i informatike. Nakon što se korisnik prijavio u aplikaciju tada može birati jednu od tri ponuđene mogućnosti prijave. A to su preko PIN-a, Uzorka i Otiska prsta.

FOI Prijava

Na slici ispod možemo vidjeti dijagram aktivnosti FOI prijave. Naime, sama prijava ovisi o FOI CAS (*Client Access Server*) serveru prema kojem mi igramo ulogu CAS klijenta. Pošto su potrebna određena dopuštenja za korištenje takvog načina prijave isto smo zatražili od **Centra za razvoj programskih proizvoda** od kojih smo dobili upute kako se isti može implementirati. Međutim, kako bi sustav radio za vrijeme izrade programskog proizvoda potrebno je raditi na *localhostu* pa je i sama uspostava takve komunikacije bila dosta kompleksna (opisano u odjeljku 8). Sama prijava se nalazi u *PokreniAplikacijuActivity* unutar kojeg se na temelju ID-a mobitela provjerava postojanje korisnika u bazi, ako isti ne postoji otvara se *webview* unutar kojeg se poziva FOI Prijava. Ukoliko isti postoji, a nije postavljen brzi način prijave također se otvara FOI Prijava. Ukoliko korisnik postoji i postavio je brzi način prijave obavlja se prijava putem odabranog načina. Razlog zašto je ova prijava unutar .Android dijela je upravo radi specifičnosti **webviewa**. Naime, problem nastaje sa **Xamarin.Forms** webviewom koji se ponaša drukčije od Androidovog te se FOI prijava jednostavno nije mogla otvoriti u Xamarin.Forms webviewu. Pošto je riječ o androidovom webviewu bilo nam je omogućeno koristiti mogućnosti *overridea* funkcija koje koriste **webviewclienti** te se s pomoću istih dohvaćaju potrebni podaci i otvara stranica FOI prijave unutar webviewa. Tako koristimo klasu *HelloWebViewClient* koja nasljeđuje *WebViewClient*

```
graph TD
    subgraph Korisnik
        Start(( )) --> Pokreni[Pokreni aplikaciju]
        Unesi[Unesi podatke za prijavu]
        Ispisi[Ispisi poruku o neuspješnoj prijavi]
        Postavi[Postavi atribute prijavljenog člana]
        End(( ))
    end

    subgraph Aplikacija
        Otvori[Otvori WebView FOI prijava]
    end

    subgraph WebServis
        Provjeri1[Provjeri postojanje korisnika u bazi na temelju id mobitela]
        Provjeri2[Provjeri unesene podatke]
        Prijava[Prijava putem postavljenog načina brze prijave]
        Vrti[Vrati token prijavljenog korisnika]
        Generiraj[Generiraj mail korisnika na temelju tokena]
        Pohrani[Pohrani mail i id mobitela u bazu podataka]
    end

    Start --> Pokreni
    Pokreni --> Provjeri1
    Provjeri1 --> D1{ }
    D1 -- "Korisnik postoji?" --> Otvori
    D1 -- "postoji i ima postavljen način prijave" --> Prijava
    Otvori -- "ne postoji" --> Unesi
    Prijava --> Unesi
    Unesi --> Provjeri2
    Provjeri2 --> D2{ }
    D2 -- "da" --> Vrti
    D2 -- "ne" --> Ispisi
    Vrti --> Generiraj
    Generiraj --> Pohrani
    Pohrani --> Postavi
    Postavi --> End
    Ispisi --> Unesi
```

[illegible]

Ključan način prijave u naš sustav predstavlja modularna implementacija prijave „brzih“ načina, tj. pin, otisak i uzorak. Na slici ispod možemo vidjeti dijagram klase funkcioniranja cijelog modula. Ključ implementacije nalazi se u *InterfaceModule* unutar kojeg se nalazi interface *IPrijava* sa svojim atributima (*StanjeZadnjePrijave* i *UneseniPodatak*) i metodama (*PrijavaModulom*, *PromjenaPodataka* i *VratiUneseniPodatak*). Svaka metoda ima specifične parametre tipa *Action*. Action tip nam služi kako bi funkcije proslijedili kao parametre odnosno za **callback** funkcionalnost modula.

Atribut *StanjeZadnjePrijave* služi kako bi se nakon izvršenja metode *PrijavaModulom* u atribut spremilo je li prijava uspješna ili nije. Atribut *UnesenPodatak* služi za drugi slučaj, a to je kada se izvršava metoda *PromjenaPodataka* koja služi za promjenu podataka brze prijave unutar aplikacije.

Metoda *PrijavaModulom* prima dvije funkcije te *string*. Funkcije su potrebne jer moduli sadržavaju svoju stranicu koju je potrebno staviti na vrh stoga navigacije u aplikaciji što nije moguće unutar modula pa se to mora izvesti u aplikaciji. Zato je potrebno prije korištenja modula napraviti metodu koja prima sve parametre koji su potrebni za rad stranice modula, te jedan parametar za tip stranice. Metoda mora imati samo jednu liniju a to je stavljanje na vrh stoga stranice koja se dobije u parametru te je potrebno proslijediti ostale parametre stranici. Druga metoda služi za zatvaranje stranice, to jest služi za uzimanje stranice sa stoga. Treći parametar potreban funkciji je hashirani podatak kojeg korisnik mora unesti kako bi potvrdio svoju prijavu.

Metoda *PromjenaPodataka* je ista kao i prethodna metoda samo je još potrebno da metoda koja se koristi kao prvi parametar funkcije *PromjenaPodataka* prima jednu funkciju kao metodu, a to je treća funkcija sučelja.

Ta treća funkcija sučelja koristi se kako bi se iz stranice za unos podataka do klase modula prenio izmijenjeni podatak o prijavi.

Ispod se mogu vidjeti implementirane metode koja se proslijeđuju kod korištenja prijave putem sučelja.

```

public async void OtvoriStranicuModula(Type tipUI, Action<Type> zatvaranjeUI, string hashiraniPodatak)
{
    var args = new object[] { hashiraniPodatak, zatvaranjeUI };
    await Navigation.PushAsync((Page)Activator.CreateInstance(tipUI, args));
}

1 reference
public void ZatvoriStranicuModula(Type tipUI)
{
    if (tipUI == null)
    {
        KreirajKorisnika();
        UdiUAplikaciju();
    }
    else
    {
        KreirajKorisnika();
        Navigation.PopAsync();
        UdiUAplikaciju();
    }
}

```

Ispod se mogu vidjeti metode koje se proslijeđuju kod korištenja izmjene podataka prijave.

```

3 references
public async void OtvoriStranicuModulaSPotvrdom(Type tipUI, Action<Type> zatvaranjeUI, string hashiraniPodatak, Action<string> vratiPodatke)
{
    var args = new object[] { hashiraniPodatak, zatvaranjeUI, vratiPodatke };
    await Navigation.PushAsync((Page)Activator.CreateInstance(tipUI, args));
}

3 references
public void ZatvoriStranicuModulaSPotvrdom(Type tipUI)
{
    if (tipUI == null)
    {
        PohraniPinUBazu(noviNacinPrijave.UneseniPodatak, idModula);
    }
    else
    {
        PohraniPinUBazu(noviNacinPrijave.UneseniPodatak, idModula);
        Navigation.PopAsync();
    }
}

```

Te na kraju još preporučeni izgled metode klase modula koja se proslijeđuje stranici modula.

```

5 references
public void VратиUneseniPodatak(string proslijedeniPodatak)
{
    this.UneseniPodatak = proslijedeniPodatak;
}

```

Pregled literature

Za pregled literature koristimo stranicu *MainMenuDetail* u kojoj prikazujemo popis svih literature. Za dohvaćanje literature i prikaz na zaslonu koristimo metodu *DohvatiPublikacije* u kojoj preko kontrolera *Publikacije* dohvaćamo sve publikacije iz baze podataka, te ih spremamo u *ListView listaPublikacije*, te u statičku listu *listasvihPublikacija*. Ovisno o broju publikacije, dinamički generiramo broj prikaza na ekranu. U slučaju da korisnik nema interneta u trenutku prikaza ekrana za pregled

literature, koristimo metodu *DohvatiPublikacijeKadJeMoguće*. Tom metodom postigli smo da dohvatimo publikacije čim se korisniku vrati Internet.

Sortiranje literature

Na glavnom zaslonu za pregled literature nalazi se gumb „Sortiraj“. Pritiskom na taj gumb, otvara nam se „Pop up „ prozor sa mogućnostima sortiranja. Ukoliko korisnik odabere sortiranje od A do Z, to će učiti preko metode *SortirajAZ*, gdje ćemo sortirati sve publikacije preko metode *OrderBy*. Kako ne bismo ponovno povlačili podatke iz baze, koristimo listu publikacijeZaSortiranje u koju smo spremili podatke iz statičke liste s stranice *MainMenuDeailt*, u koju smo zapisali sve publikacije. Nakon što smo izvršili metodu sortiranja, vraćamo sortirane podatke nazad u statičku klasu, te zatvaramo pop up prozor i šaljemo poruku stranici *MainMenuDetail*, gdje se preko konstruktora poziva metoda *OsvjeziListuPublikacija*, s kojom prikazujemo željeno sortiranje. Ovaj postupak primjenjujemo kod svih sortiranja, jedino se razlikuje metoda sortiranja sadržaja u listi.

Filtriranje literature

Mogućnost filtriranja korisnik ima odabirom na gumb *Filtriraj*, na zaslonu za pregled literature. Klikom na navedeni gumb, otvara se pop up zaslon, gdje korisniku nudimo mogućnost filtriranja po autorima, izdavačima, prvo slovo publikacije, te kategorije. Ovisno o odabiru filtracije izvodi se metoda koja filtrira literaturu. Pošto sve filtracije rade na isti princip, objasniti ću jednu. Da bi filtrirali literaturu po autoru, aplikacija će preko metode *DohvatiAutore* dohvatiti sve autore iz baze podataka, te ih vratiti u *ListView* u xaml datoteci. Time će se generirati na zaslonu svi autori koji postoje u bazi podataka kao lista autora poredanih u jedan stupac s mogućnosti odabira preko *checkbox*-a. Klikom, odnosno označavanjem određenog *checkbox*-a, označili smo autora prema kojemu želimo filtrirati. Možemo istovremeno odabrati samo jednog ili više autora. Kako bismo do kraja izvršili filtriranje, potrebno je kliknuti na gumb „ODABERI“ omogućujemo ispis samo literature po željenom autoru. Klikom na navedeni gumb, pokrećemo metodu *ListaOdabranihFiltra* preko koje stavljamo autore u statičku listu *filterAutori* u klasi *Filteri*. Nakon što je metoda dodala autore u listu iz klase *Filteri*, šaljemo poruku, preko funkcije *MessagingCenter*, konstrukturu stranice *MainMenuDetailt* da pokrene metodu *OsvjeziListuPublikacije*, koja izvodi statičku metodu *FiltrirajPublikacije* iz klase *Filtri*, i dohvaća sve publikacije ovisno o odabranim

autorima, te vraća listu sa filtriranim publikacijama u metodu *OsvjeziListuPublikacije*, koja zatim tu listu sprema i *ListView* na stranici *MainMenuDetail*, te prikazuje na zaslonu filtriranu literaturu. Nakon što su sve metode završile, pop up prozor se gasi, i vraćamo se na stanicu pregled literature, gdje možemo vidjeti filtriranu literaturu.

Pretraga

Pretraga se nalazi na stranici pregled literature u obliku polja za unos teksta. Ukoliko korisnik unese tekst, aktivira se metoda *UnosPretraživanje* za svaki uneseni string. Uneseni string se pomoću kotrolera *PretragaPublikacije* proslijeđuje bazi podataka, gdje se vrši upit, te se vraćeni podaci iz baze podataka zapisuju i *ListView*. Time smo zapisali nove podatke, te tako dobili na zaslonu nove literature, ovisno o upisanom sadržaju. Pretraga se može vršiti preko naslova, autora ili publikacije.

Detaljne informacije

Odabirom određene literature, pritiskom unutar okvira literature, otvara nam se stranica *BookInfo*. U konstruktoru se poziva metoda *DohvatiPublikacije* te joj proslijedimo id publikacije, koju smo proslijedili kroz konstruktor. Ova metoda dohvaća publikaciju iz baze te ju sprema u statički listu *listaSvihPublikacija*. Nakon toga se kroz konstruktor stvara sadržaj na zaslonu. Na ovom zaslonu možemo vidjeti sve informacije o knjizi, autor, publikacija, broj stranica, izdanje, izdavač, ISBN. Isto tako možemo vidjeti koliki je broj knjiga slobodan odnosno rezerviran.

Pregled sadržaja

Da bismo pregledali sadržaj određene publikacije, potrebno je odabrati gumb „SADRŽAJ“ na zaslonu detaljne informacije. Kroz metodu *ButtonSadržaj*, šaljemo publikaciju u konstruktor stranice *Sadržaj*, koji poziva dvije metode. Prva metoda je *DohvatiNaziv*, kojom dohvaćamo naziv publikacije, te ga ispisujemo kao naslov na zaslonu, te druga metoda *DohvatiLink*, kojom dohvaćamo link adresu sadržaja na web stranici foj knjižnice, te ga prikazujemo pomoću *fukncije webView*.

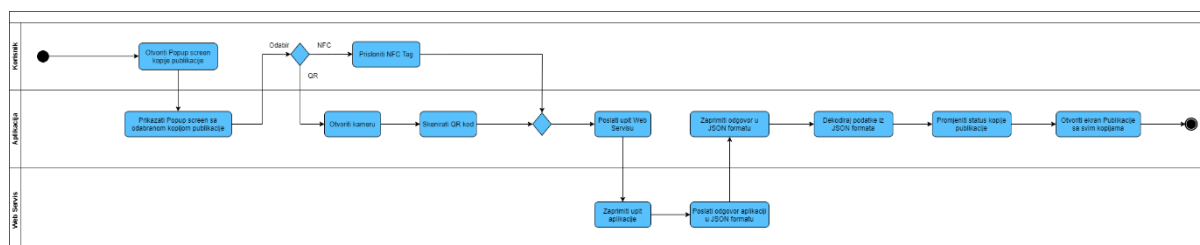
Rezervirane knjige

Na zaslonu detaljne informacije o publikaciji možemo klikom na traku slobodno, ukoliko postoji slobodan broj kopija knjiga, rezervirati. Otvara se pop up zaslon *RezerviranjePopupPage* te proslijeđujemo do konstruktora publikaciju. Na zaslonu

možemo vidjeti tri načina rezervacije, pritiskom na gumb rezerviraj, gdje rezerviramo knjigu, te nakon toga odlazimo do knjižnice osobno po nju, ili možemo direktno posuditi preko NFC i QR opcije (slika ispod). O tim opcijama nešto kasnije. U konstruktoru se prvo provjerava da li je publikacija slobodna za posudbu ili su sve kopije posuđene. U provjeru provjeravamo kroz if funkciju. Kroz metodu *GumbRezervirajKliknut* updatamo status u bazi podataka iz slobodno u posuđeno, za odabranog korisnika kroz kontroler *GumpRezerviraj*.

Posudba knjige koristeći QR ili NFC

Da bi došli do mogućnosti do posudbe publikacije putem **QR** skeniranja koda ili prislanjanjem puta **NFC** Taga, korisnik odabire kopiju publikacije sa statusom “Slobodno” jer za kopije publikacije kojemu su stanja “Rezervirano” i “Posudeno” korisnik nije u mogućnosti posuditi publikaciju putem QR ili NFC-a. Korisniku se otvara Popup stranica odabrane kopije publikacije pod nazivom *RezerviranjePopupPage*. Nakon otvaranja, korisnik može posuditi publikaciju putem QR koda ili NFC-a. Ukoliko korisnik odabere posudbu putem QR koda, pritišće na ikonicu od QR koda gdje mu se nadalje otvara kamera za skeniranje koda. Nakon skeniranja koda dohvaća se *id kopije* kojima se šalje upit prema Web Servisu za tu kopiju publikacije u kojemu se stanje te kopije mjenja u “Posudeno” te vraća rezultat prema aplikaciji gdje se refresha stranica *BookInfo* u kojoj se nalazi odabrana knjiga sa svim kopijama i korisniku se ispisuje datum posudbe i datum isteka iste. Ukoliko korisnik odluči posuditi putem NFC-a, korisnik nadalje prislanja NFC Tag u kojemu se dohvaća *id kopije* te isto vrijedi kao i za QR gdje se šalje upit prema Web Servisu i mjenja stanje te kopije publikacije u “Posudeno”. Na slici ispod možemo vidjeti dijagram aktivnosti posudbe putem QR i NFC.



Korisnički račun

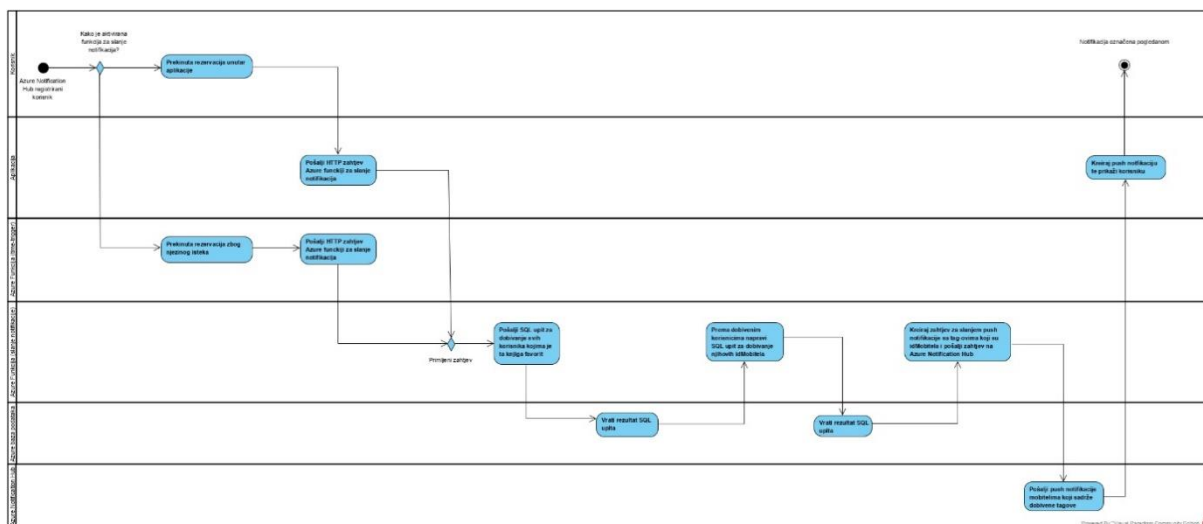
Da bi smo došli do profila, moramo odabrati u izborniku „Moj profil“. U profilu korisniku se kroz metodu *KreirajStatistiku* ispisuje njegova statistika rezerviranih,

posuđenih knjiga, te istek rezervacije odnosno posudbe ukoliko ima. Svu njegovu statistiku dohvaćamo preko kontrolera Statistika kojoj prosljeđujemo id korisnika kojeg dohvaćamo preko statičke klase *Clanovi*. Na profilu još korisnik može vidjeti povijest svojih rezervacije i posudbe, kroz gumb „Povijest“ koja nam otvara novu stranicu *PovijestKorisnika*. Kroz gumb ButtonPostavkePrijave_Clicked možemo dodati ili promijeniti naš način prijave u aplikaciju.

Notifikacije

U aplikaciji postoje dvije upotrebe notifikacija. Prva upotreba je za obavještanje korisnika da se oslobodila kopija knjige koja je dodana u njegove favorite, a druga se koristi za obavještanje korisnika da mu je danas zadnji dan da vrijedi neka posudba/rezervacija. Ispod će biti prikazani dijagrami aktivnosti te će biti opisane neke od samih aktivnosti koje smatramo da je potrebno detaljnije objasniti kako funkcioniraju.

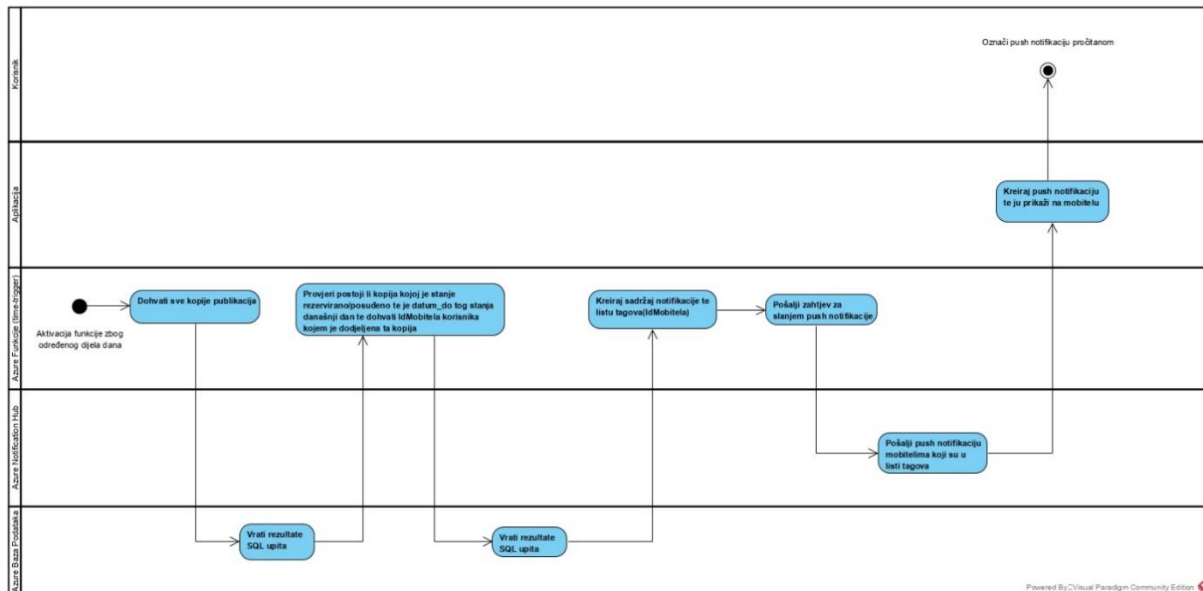
Podsjetnik o slobodnoj knjizi



Slanje „push“ notifikacija je ostvareno prilično komplicirano. Slanje notifikacije o oslobođenju knjige može se aktivirati na dva načina, a to su taj da korisnik sam prekine rezervaciju unutar aplikacije te prekidanje rezervacije koja je istekla što se radi pomoću azure funkcije koja se ponaša kao „time trigger“ te svaki dan u 8:00 provjerava postoje li rezervacije koje su istekle te ih automatski prekida. Ako se ostvari jedan od uvjeta šalje se HTTP zahtjev na azure funkciju koja najprije dohvaća potrebne idMobitela. IdMobitela se dohvaća jer se kod registracije svakog korisnika njemu dodaje „tag“ čija je vrijednost IdMobitela te se onda kasnije može prema tome tag-u

poslati ciljana notifikacija. Kada je funkcija napravila sadržaj poruke i listu tagova ona šalje zahtjev Azure Notification Hub-u koji se onda pobrine za slanje notifikacije prema mobitelima. Tu notifikaciju dobiva mobitel te on u sebi sadrži klasu u kojoj je definirano što mora napraviti s tom notifikacijom te ju on onda kreira i prikaže na mobitelu.

Notifikacija o isteku rezervacije ili roka vraćanja knjige



Notifikacija o isteku rezervacije ili roka za vraćanje knjige započinje tako da se svaki dan u 8:00h aktivira azure funkcija koja najprije dohvaća sve kopije publikacija te zatim za svaku kopiju gleda je li ona rezervirana ili posuđena te ako je gleda je li danas zadnji dan do kad vrijedi ta rezervacija/posudba i ako je vraća IdMobitela od tog člana koji ima posudbu/rezervaciju. Kada se dohvate potrebni IdMobitela kreira se sadržaj notifikacije te se kreira lista tagova kako bi Azure Notification Hub znao kamo mora poslati push notifikaciju. Kada se pošalje push notifikacija prema mobitelu, aplikacija kreira tu notifikaciju jer ima klasu u kojoj je definirano kako će prikazivati dobivenu notifikaciju te ju prikaže korisniku.

9. Xamarin

Xamarin se integrira s Visual Studio, Microsoftovim IDE-om za .NET Framework, a kasnije je dostupan za upotrebu macOS korisnicima putem Visual Studio za Mac. Koristi C# kod, te razne alate i biblioteke za kreiranje aplikacija na platformama kao što su iOS, Android, macOS, Windows apps, te za razne druge.

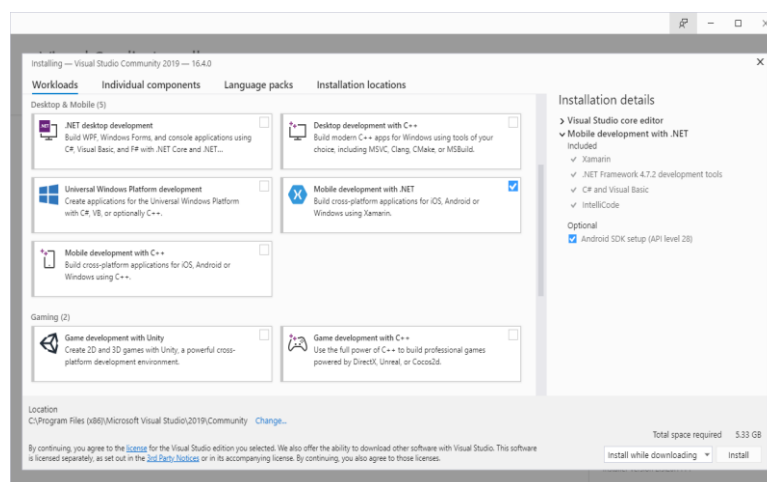
Microsoft **Visual Studio** je integrirano razvojno okruženje (IDE) koje je razvio Microsoft. Koristi se za razvoj računalnih programa, kao što su web-stranice, web aplikacije, servisi, te mobilne aplikacije.

Pošto je Xamarin dio Visual Studio paketa, zasniva se na .NET platformi. .NET je razvojna platforma napravljena od raznih alata, programskog jezika i biblioteka za kreiranje raznih različitih vrsta aplikacija. Baza platforme pruža komponente svim različitim vrstama aplikacija. .NET je besplatna cross-platform platforma, otvorenog koda. Podržava različite programske jezike, editore i biblioteke za kreiranje web, mobilnih, desktop, gaming, i IoT aplikacija.

Instaliranje Xamarina

Prema uputama na moodle-u u nastavku će biti opisana instalacija i neke od osnova rada u Xamarinu. Pošto smo mi koristili Xamarin za Windows, slijedi opis instalacije istoga na računalu s operacijskim sustavom Windows.

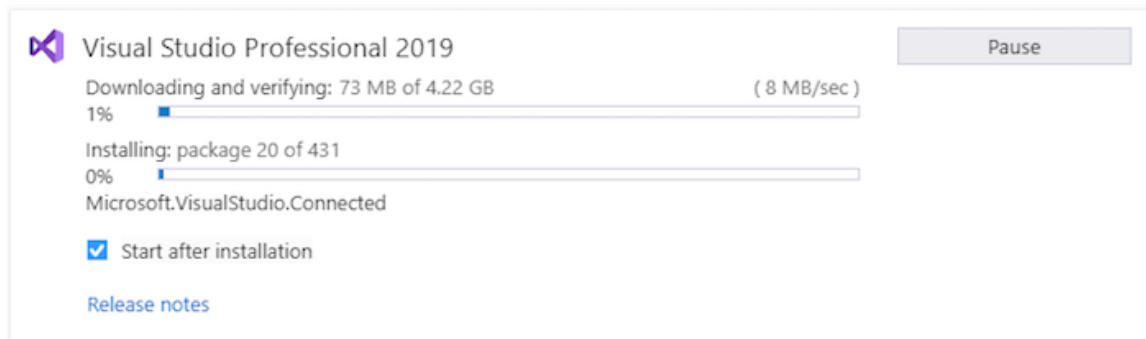
Prvo što trebamo je skinuti Visual Studio, najbolje zadnju verziju (2019). Kako mi koristimo besplatnu verziju, onda ćemo svakako odabrati Community verziju. Verzija koju mi koristimo je Visual Studio 2019 Community.



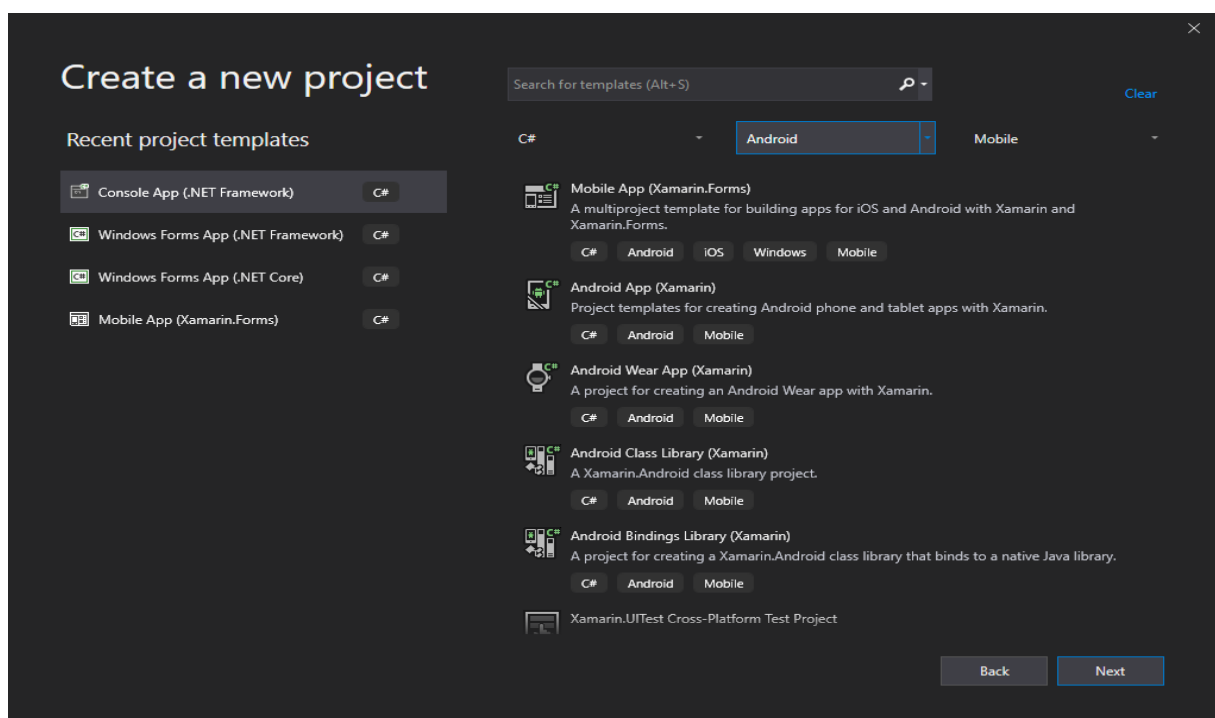
Posjetimo VisualStudio stranicu , te odaberemo navedenu verziju, te nakon završenog skidanja, dvaput kliknemo na start instalacija. Otvara nam se ekran

instalacije (slika) , gdje odabiremo željene instalacijske pakete. Paket koji mi trebamo je „Mobile development with .NET“.

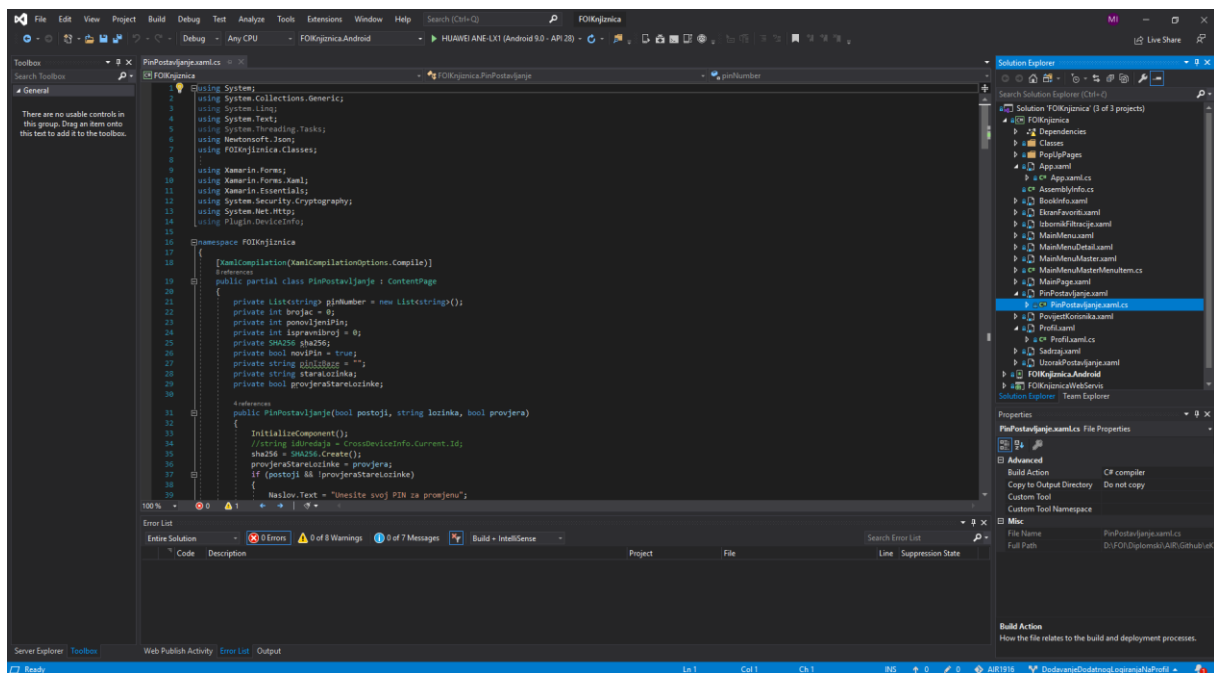
Nakon što smo odabrali, kliknemo na „Install“ u donjem desnom kutu. Za ovu instalaciju potrebno je oko 19 GB slobodno prostora na disku.



Nakon što je instalacija gotova, kliknemo na „Launch button“ kako bi pokrenuli Visual studio. Nakon što smo pokrenuli Visual studio možemo odabrati kreiranje novog projekta za mobline aplikacije (slika ispod).



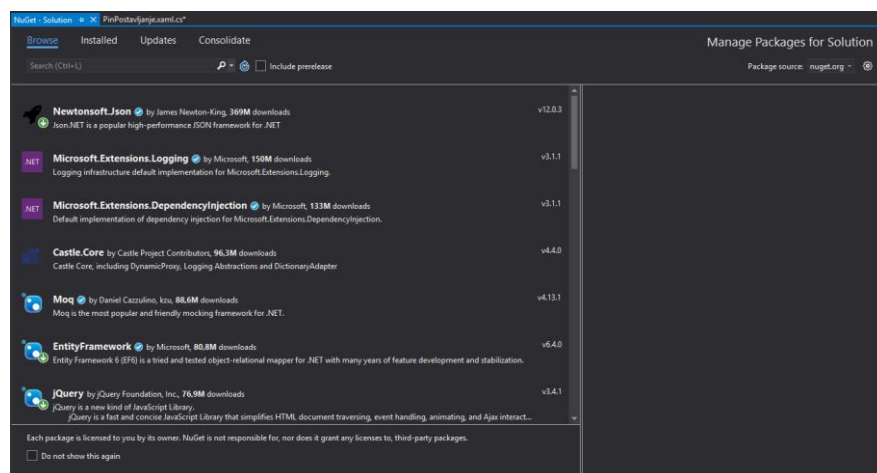
Odabirom na projekt, u našem slučaju to je **Android App (Xamarin)**, dolazimo do sučelja (slika ispod) na kojem radimo dalje naš projekt.



Rad u Visual Studio Xamarin

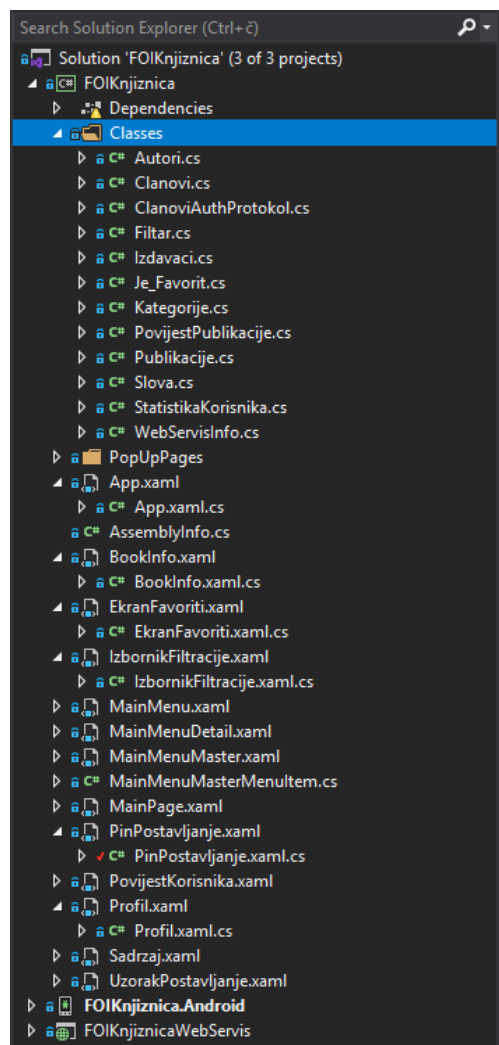
Vrlo važna stavka u radu unutar Xamarin-a su svakako „Nuguet „ paketi. Nuguet paketi su moduli, odnosno gotova rješenja za neku vrstu funkcionalnosti. Time se postiže jednostavnost, te se značajno ubrzava rješenje, odnosno izrada aplikacije.

Kako bismo dodali Nuguet paket, potrebno je otići *Tools -> Nuguet packages Manager -> Manage Nuguet Packages for solutions*. Nakon odabira, otvara nam se Nuguet prozor (slika). Nuguet browser se sastoji od 4 izbornika. *Browse* u kojem tražimo, odnosno odabiremo paket kojeg želimo uključiti u naše rješenje. Nakon odabira, potrebno je odabrati u koje projekte našeg rješenja želimo uključiti paket. Neki paketi trebaju se uključiti u cijelo rješenje, dok drugi u samo određene projekte. I to je vrlo važno da odaberemo pravi projekt, kako bi paket mogao normalno raditi.



Sljedeći izbornik je *Installed*, gdje možemo vidjeti sve Nuget pakete koje smo instalirali u naše rješenje, te možemo vidjeti točno u koji dio projekta smo ga uključili. Treći dio izbornika je *Updates*, koji nam javlja potrebna ažuriranja. Zadnji izbornik je *Consolidate*.

Način izrade programskog rješenja u Xamarinu



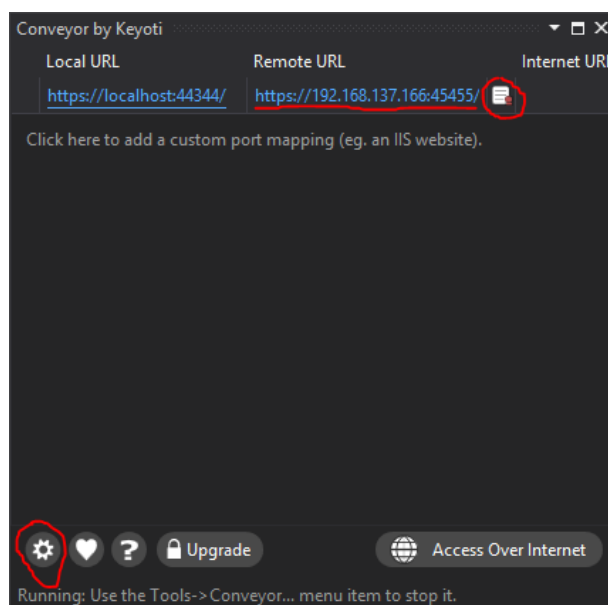
Rad u Xamarinu je veoma sličan kao i radi u "običnom" Visual Studio-u. Solution explorer sastoji se od projekata. U našem slučaju to je *FOIKnjiznica*, *FOIKnjiznica.Android* i *FOIKnjiznica.WebServis*. *FOIKnjiznica.Android* je projekt koji služi kao komunikaciju s android uređajima, te služi za komunikaciju androida i našeg projekta. *FOIKnjiznica* služi za izgradnju našeg projekta. U njemu se nalaze sve potrebne klase i stranice koje koristimo u aplikaciji. Stranice u xamarinu su "*Content Page*" koje su oblika XAML koda s c# kodom u pozadini (slika). Treći dio našeg rješenja je projekt koji se zove *FOIKnjiznica.WebServis* te služi za komunikaciju sa bazom podataka, točnije, služi nam kao WebAPI putem kojeg se ostvaruje komunikacija aplikacije i baze.

Karakteristika pokretanja aplikacije (FOI Knjižnica)

Trenutačno, aplikacija radi na **localhostu** jer nemamo potpuni pristup CAS serveru jer je aplikacija, realno, i dalje u razvoju. Točnije, naša domena koja se koristi nije uvrštena u pristup CAS serveru. Stoga, u Visual Studiu koristimo jedan dodatan alat **Conveyor by Keyoti** koji nam omogućuje sigurnu vezu sa FOI CAS serverom sa mobitela preko localhosta (pošto je https konekcija).

Kako bi ga instalirali potrebno je pratiti korake navedene na <https://marketplace.visualstudio.com/items?itemName=vs-publisher-1448185.ConveyorbyKeyoti>.

Nakon instalacije potrebno ga je uključiti tako da odemo u *Tools* i samo kliknemo na njega. Tada je potrebno pokrenuti naš projekt FOIKnjiznicaWebServis u IIS exploreru te ćemo dobiti prozor u kojem je prikazan „**Remote URL**“ putem kojeg možemo pristupiti localhostu. Naravno, kako bi uspjeli pristupiti istom sa mobilnog uređaja biti će potrebno instalirati conveyor certifikat, a postupku te instalacije pristupamo kada kliknemo na ikonicu pokraj Remote URL-a.



Nakon uspješne instalacije certifikata na mobilni uređaj, potrebno je izvršiti malu konfiguraciju *web.config* kako bi aplikacija mogla raditi na localhostu nekog računala, a koraci su sljedeći:

1. Otvoriti *command prompt* i izvršiti naredbu *ipconfig* te pronaći našu IPv4 adresu.
2. U *Conveyor by Keyoti* prozoru postaviti u opcijama našu IPv4 adresu (slika gore, donji lijevi kut).
3. Pokrenuti naš FOIKnjiznicaWebServis projekt te pogledati **Remote URL** koji dobijemo (Port je jako bitan!)
4. Sada u *web.config* je potrebno promijeniti atribut **serverName** te obavezno staviti i broj porta.

5. Kako bi se aplikacija uspješno izvršila na *webviewu* unutar aplikacije potrebno je još u `PokreniAplikacijuActivity.cs` promijeniti `webView.LoadUrl()` u istu vrijednost kao i što je `serverName`.

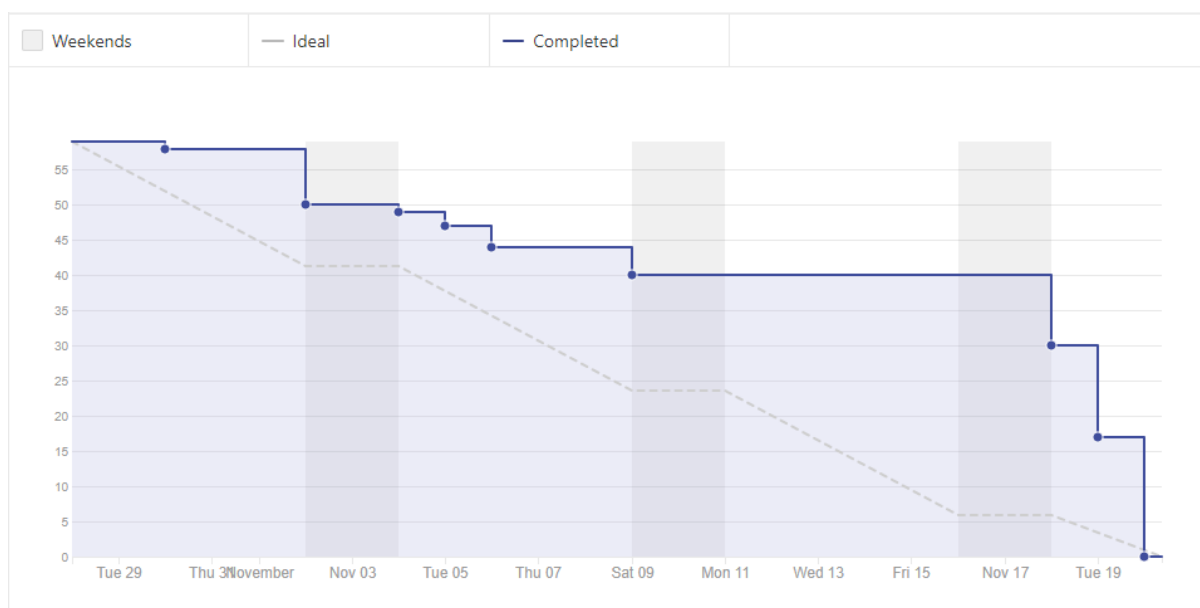
10. BurnDown grafovi

Radni satovi za izradu projekta

Da bismo mogli dobiti prosječan broj sati koji će nama biti potreban za uložiti, da bismo izradili aplikaciju, potrebno je odrediti ukupni potreban broj sati za kolegij. Kako ovaj kolegij nosi 6 ECTS bodova, to znači da student treba odraditi 180 sati rada. Ako stavimo da prosječan broj tjedana iznosi 15, dobijemo da student treba raditi 12 sati tjedno. Tjedno kolegij sadržava 3 sata predavanja, te još tri sata labos vježbi. Ukupno 6 sati tjedno. Kako se nastava izvodila 10 tjedana, dobijemo da student potroši 60 sati za potrebe nastave. Kako je 60 sati potrebno raspodijeliti na 15 tjedana, dobijemo da student prosječno „potroši“ 4 sata za potrebe nastave. Stoga za na aplikaciji ostalo 8 sati tjedno. Ako 8 sati podijelimo na 7 dana, jer smo radili i kroz vikende, stoga dolazimo da dnevno možemo raditi prosječno sat vremena. Kako nas je 5 u timu, dolazimo do toga da dnevno možemo raditi 5 sati. Prosječno nam sprint traje 21 dan. Stoga smo za sprintove uzeli 105 sati rada.

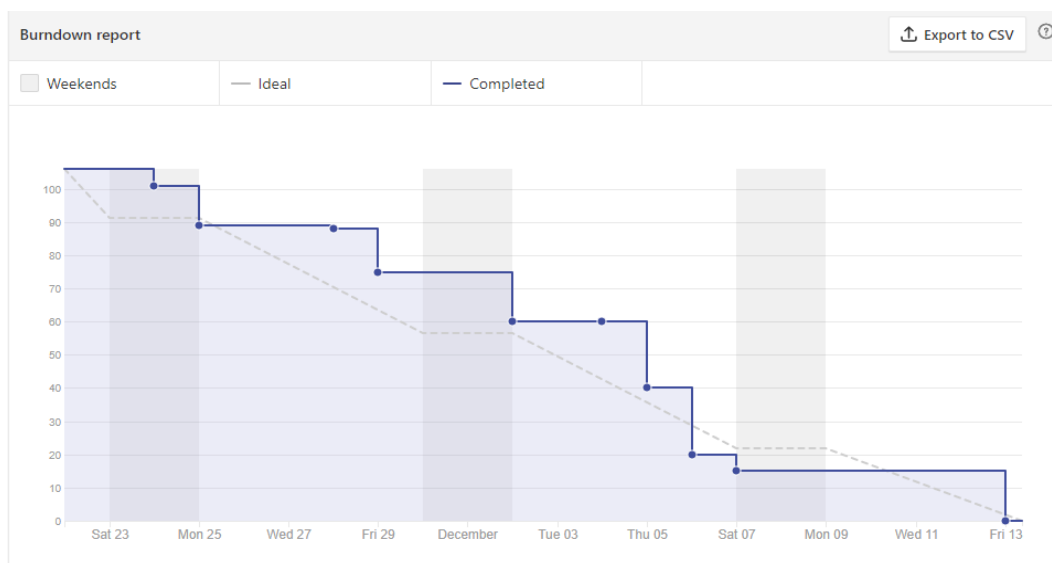
Ovo je subjektivan broj sati, neki prosjek koji smo uzeli da bismo odredili trajanje sprintova, te želimo napomenuti kako stvarni broj sati koji smo odradili u svakom sprintu je puno već, jer nismo uzimali vrijeme koje nam je bilo potrebno da uklonimo određene probleme, koje smo imali što zbog nove tehnologije, što zbog manja znanja., što zbog nedostatka dostupnih materijala.

Prvi sprint



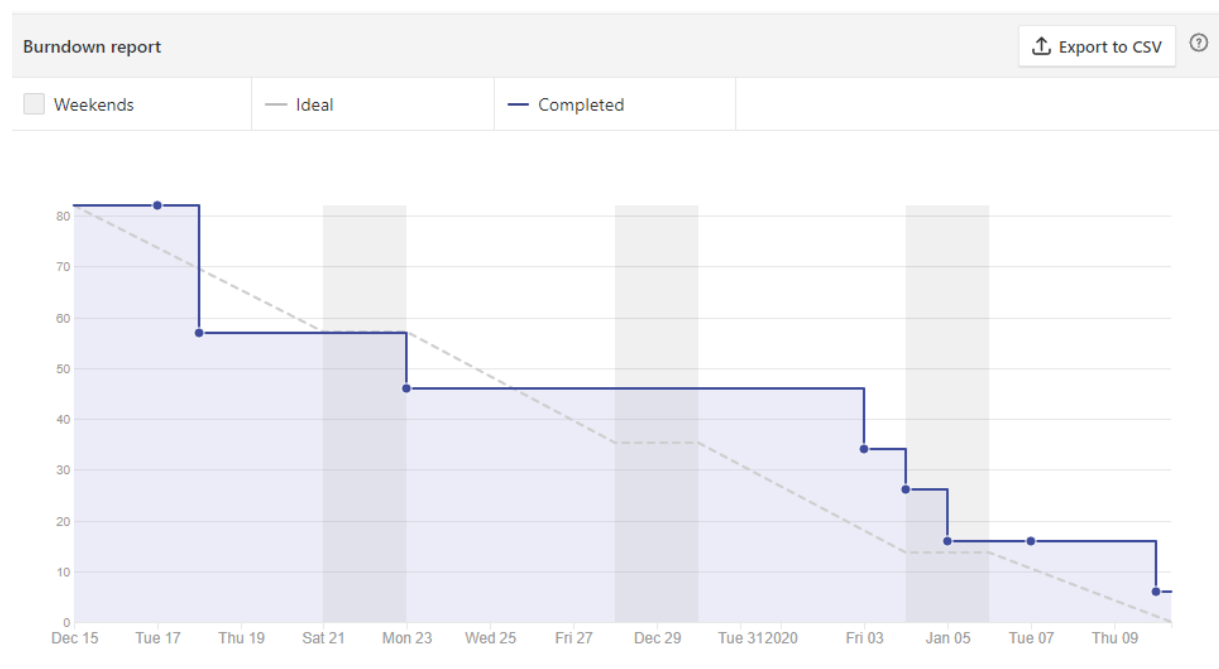
Za prvi sprint odredili smo trajanje od 21 dana., te smo ga u roku , kako smo i isplanirali, izvršili. Odradili smo ga veoma dobro. U prvi sprint stavili smo tehničku dokumentaciju, kreiranje baze podataka, spajanje sa našom aplikacijom, većinom početne funkcionalnosti projekta . Uspjeli smo još odraditi i funkcionalnost prikaza podataka. U ovom sprintu obavili smo konzultacije sa mentorom našeg projekta, timski sastanak u kojem smo napravili reviziju našeg rada, te da odredimo smjernice za sljedeći sprint. Broj sati koji smo si odredili bio je idealno za izvršavanje svih zadanih zadataka. Nije bilo potrebe za dodavanjem dodatnih satova.

Drugi sprint



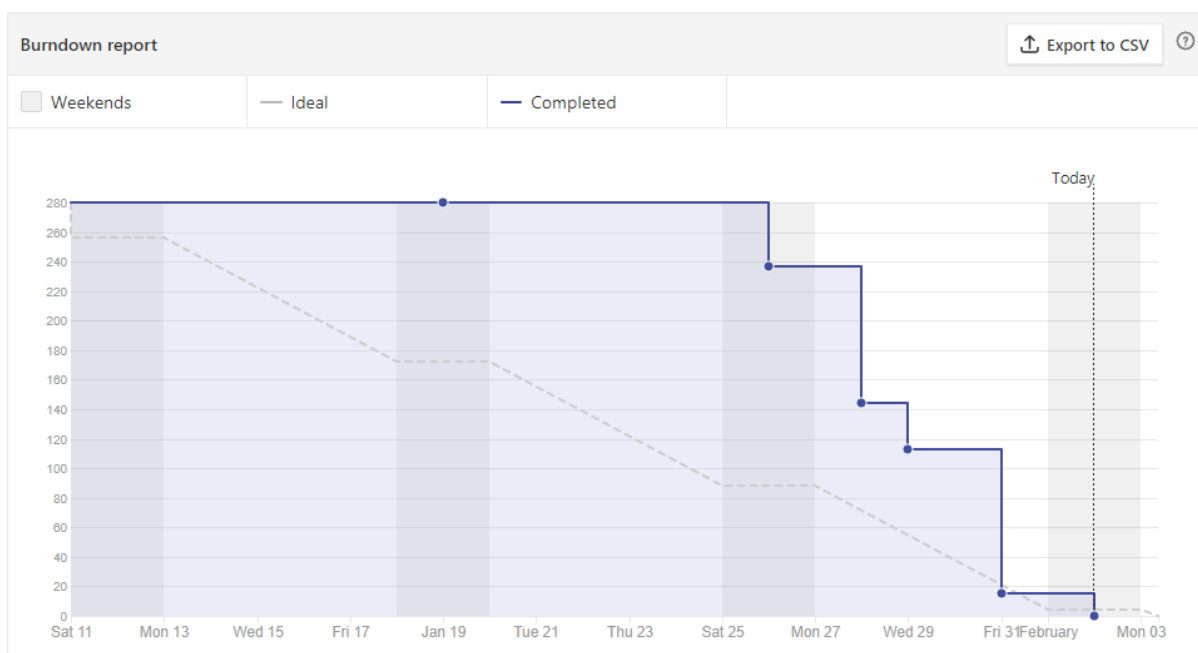
Kao i prvom sprintu, tako smo i u drugom sprintu uspjeli izvršiti sve zadane zadatke. Ovaj sprint bio je puno teži i vremenski zahtjevniji od prvog. Razlog težine ovog sprinta sigurno je što su zadaci bili puno kompleksniji, te su bili zadani za izradu funkcionalnosti aplikacije. U ovom sprintu upoznali smo se sa novom tehnologijom. Naučili smo način funkcioniranja programskog okruženja Xamarin. Po prvi puta upoznali smo se s radom WebAPI-em. Sa radom fokusirali smo se dohvaćanje publikacije, te njihovo sortiranje i filtriranje. Svakako najteži dio ovog sprinta bilo je dohvaćanje potrebnih informacija o tome kako se implementira FOI login.

Treći sprint



Nažalost ovaj sprint nismo uspjeli završiti na vrijeme, iako smo stavili veći broj sati, odnosno trajao je 4 dana duže od prethodnih. Razlog što se ovaj sprint izvodio za vrijeme zimskih praznika svakako nije razlog kašnjenja, štoviše radili smo i više, već je razlog što smo naišli na neke probleme za koje nismo mogli naći pravo rješenje, te smo gubili puno sati na dobivanje odgovora za rješavanje problema. Jedan od najvećih problema u ovom sprintu svakako je bio EDU FOI prijava.

Četvrti sprint



Četvrti sprint, odnosno zadnji sprint trajao je jednako kao i prethodni, ali za razliku od njega, ovoga smo uspješno završili i to prije zadanoga roka. Ostaviti smo si još dva dana da sve funkcionalnosti aplikacije provjerimo i testiramo, da nema grešaka, te da doradimo dokumentaciju. Ovaj sprint je bio veoma težak i zahtjevan. Veliku ulogu i težini doprinijeli su i kolokviji iz drugi predmeta, te nismo mogli svo vrijeme posvetiti samo izradi ove aplikacije. Kao što se može vidjeti iz grafa, početak sprinta je bio u stagnaciji, jer je to bilo vrijeme kolokvija. Kroz prethodne sprintove dobro smo se upoznali sa radom u Xamarin-u, te smo pretpostavili da ne bi trebalo biti problema i da bi sve trebalo ići bolje i brže. No kako smo si ostavili neke dosta zahtjevne zadatke, ispostavilo se da sprint neće nikako biti brži i jednostavniji. Puno vremena smo potrošili na izradu prijave, odnosno na modularnost, te nam je to predstavljalo najveći problem u ovom sprintu. Dodao bih ovdje da smo uspješno riješili i prijavu putem FOI CAS verifikacije, koju nismo uspjeli završiti u prethodnom sprintu i s čijom realizacijom smo imali naviše problema.