Višeplatformska mobilna aplikacija za evidenciju dolaska zaposlenika

Gluhaković, Mario

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

Sveučilišni preddiplomski studij računarstva

VIŠEPLATFORMSKA MOBILNA APLIKACIJA U XAMARINU

Završni rad

Mario Gluhaković

Osijek, 2017.

Sadržaj:

1. UVOD 1
1.1. Zadatak završnog rad 2
2. PRIJENOSNI OPERACIJSKI SUSTAVI 3
2.1. iOS 3
2.2. Android 4
2.3. Windows 10 Mobile 6
3. TEHNOLOGIJE ZA RAZVOJ VIŠEPLATFORMSKIH MOBILNIH APLIKACIJA 7
3.1. Xamarin 7
3.2. Phone Gap 8
3.3. Appcelerator Titanium 9
4. XAMARIN TEHNOLOGIJA 11
4.1. Problemi višeplatformskog razvoja aplikacija 11
4.2. C#, .NET i djeljivost koda 12
4.3. Xamarin Forme 15
4.4. Razvojna okruženja i emulatori 17
4.5. Struktura aplikacije 18
5. IZRADA APLIKACIJE 20
5.1. Unos podataka i provjera povezanosti na bežičnu mrežu 20
5.2. Stranica za skeniranje NFC 24
5.3. Implementacija baze podataka 26
5.4. Implementacija NFC skenera 29
6. ZAKLJUČAK 32
LITERATURA 33
SAŽETAK 35
ABSTRACT 36
ŽIVOTOPIS 37
PRILOZI 38

1. UVOD

Tema ovog završnog rada je izrada višeplatformske mobilne aplikacije koja će omogućiti zaposlenicima prijavu dolaska i odlaska s posla. U nekim tvrtkama zaposlenici se prijavljuju i odjavljuju s posla pomoću magnetnih kartica. Čest je slučaj da zaposlenik svoju karticu zaboravi doma ili da je izgubi. Ako se uzme u obzir da se veliki broj osoba ne odvaja od svoga mobilnog telefona i jako je mala vjerojatnost da će ga zaboraviti, onda se također može zaključiti da ga zaposlenici neće zaboraviti prilikom polaska na posao. S obzirom na veoma jednostavno grafičko sučelje i korištenje NFC tehnologije ova aplikacija pruža jednostavno i brzo prijavljivanje i odjavljivanje s posla. Pametni telefon će spremati podatke zaposlenika i prilikom skeniranja NFC oznake podatke tog korisnika slati u bazu podataka i tako vršiti prijavu i odjavu. U bazi će se nalaziti vrijeme prijave odnosno odjave zaposlenika.

Kao što se može vidjeti temelj ove aplikacije je korištenje mobilnih uređaja i njihovih značajki poput NFC u svrhu olakšavanja prijave i odjave zaposlenika s posla. Pošto je smisao ove aplikacije korištenje iste na raznovrsnim mobilnim platformama za izradu aplikacije je korištena Xamarin tehnologija koja omogućava izradu višeplatformskih mobilnih aplikacija. Pri izradi aplikacije su korišteni besplatni Internet tečajevi koje je omogućio Microsoft u svrhu proširenja zajednice Xamarin programera.

Prvi dio završnog rada sadrži opis tri najrasprostranjenije mobilne platforme, te opis nekoliko poznatih tehnologija za razvoj višeplatformskih aplikacija. U daljnjem dijelu teorijskog dijela je detaljno opisana Xamarin tehnologija, programski jezici koje ona koristi, razvojna okruženja, načini testiranja aplikacije i razine kompatibilnosti s uređajima. Nakon toga slijedi opis nastanka aplikacije sa svim elementima i značajkama. Vidljiv je izgled zaslona, pozadinski logički kod koji upravlja samom aplikacijom i opis funkcionalnosti samoga koda.

1.1. Zadatak završnog rad

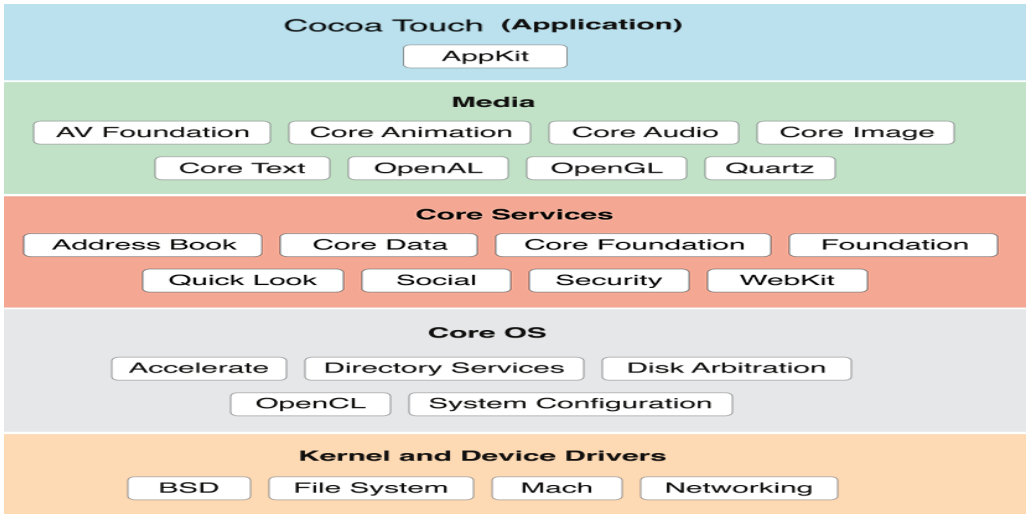
Zadatak završnog rada je opisati nekoliko odabranih tehnologija za razvoj višeplatformskih mobilnih aplikacija i po čemu se razlikuju. Detaljno opisati Xamarin tehnologiju i njene prednosti i nedostatke. U praktičnom dijelu potrebno je izraditi mobilnu aplikaciju koja će omogućiti zaposlenicima prijavu dolaska i odlaska s posla koristeći metode specifične za mobilne uređaje (skeniranje NFC oznake te detekcija spojenosti na bežičnu mrežu tvrtke).

2. PRIJENOSNI OPERACIJSKI SUSTAVI

Prijenosni operacijski sustavi su operacijski sustavi za tablete, pametne telefone i druge prijenosne uređaje. Prijenosni operacijski sustavi kombiniraju značajke operacijskih sustava stolnih računala sa značajkama pametnih telefona i ručnih uređaja kao što su ekran na dodir, Wi-Fi, GPS (engl. *global positioning system),* kamera, video kamera, NFC (eng. *near field communication*), Bluetooth i dr. Na kraju 2016. godine prodano je preko 430 milijuna pametnih telefona od kojih 86.2% pokreće Android, 12.9% iOS i 0.6% Windows Mobile. [1]

2.1. iOS

iOS je mobilni operacijski sustav kojeg je razvila američka tvrtka Apple Inc. za iPhone no danas se ovaj operacijski sustav koristi i za druge Apple proizvode kao: iPad, Apple TV i iPod Touch. Predstavljen je 9. siječnja 2007. godine pod nazivom iPhone SDK istovremeno s iPhoneom, a pušten u javnost u lipnju iste godine. U lipnju 2011. godine preimenovan u iOS. Ovaj operacijski sustav je zasnovan na projektu Darwin koja potječe od BSD i NeXTSTEP operacijskog sustava. Darwin je napravljen oko jezgre XNU koja se sastoji od mikrojezgre Mach3, različitih elemenata BSD-a i OO biblioteka za pogonske programe nazvane i/O Kit. [2]



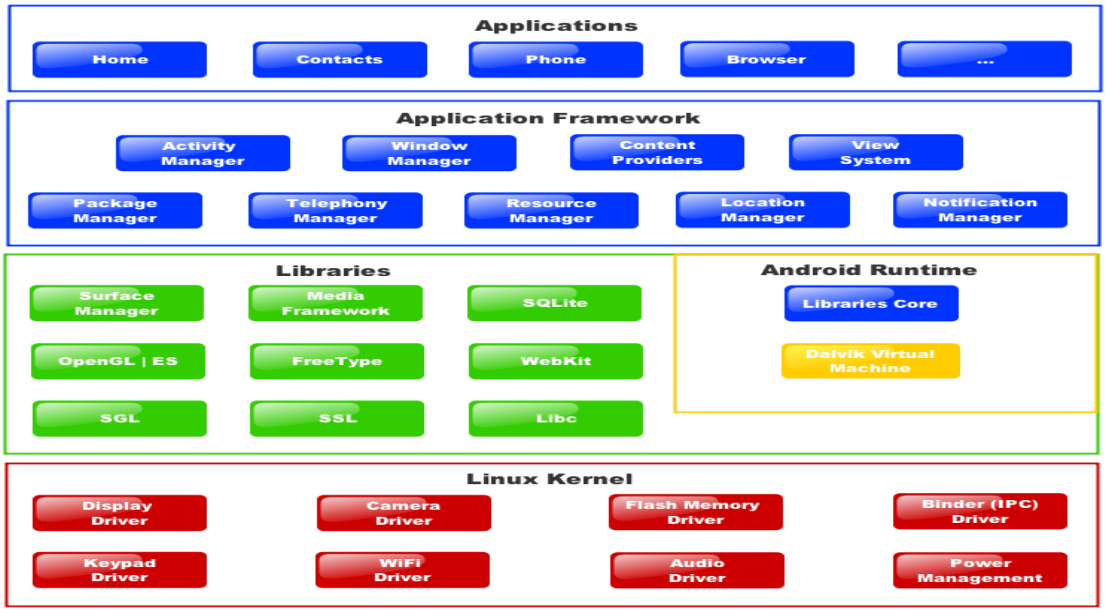
Sl. 2.1. iOS arhitektura [3]

Core Services je osnovni sloj koji koriste sve aplikacije, često samo indirektno preko biblioteka viših slojeva. Ovaj sloj upravlja bazama podataka, adresarom, korištenje uslugama oblak računala i slično. Core OS pruža sučelje za kontroliranje procesa. On podrazumijeva rada s poslužiteljem,

implementaciju sigurnosnih protokola i sučelje jezgri OS-a. Core OS sadrži sučelje za operacije s brojevima, sučelje za zaštitu podataka, sučelje za komunikaciju s vanjskim uređajima i sučelja na razini operacijskog sustava. Media sadrži tehnologije za upravljanje grafikom, audio i video datotekama u cilju stvaranja multimedijski bogatih aplikacija. Cocoa Touch sadrži ključna programska okruženja za izgradnju aplikacija za iOS. Ovaj sloj definira osnovnu infrastrukturu aplikacije i daje podršku za višenitnost, unos preko zaslona i dr.

2.2. Android

Android Inc. su osnovali Nick Sears, Rich Miner, Andy Rubin i Chris White 2003. godine kako bi razvijali programe za pametne mobilne uređaje koji bi uzimali u obzir korisničke postavke. Google je 2005. godine odlučio kupiti Android te se osnivačima i ključnim programerima pridružuju i Googleovi programeri. Open Handset Alliance (OHA) je osnovan s ciljem stvaranja javnog standarda za mobilne uređaje. Inicijator je bio Google koji je okupio 34 tvrtke iz različitih domena mobilne industrije poput proizvođača mobilnih telefona, mobilnih operatera, programera aplikacija i sličnih. U studenom iste godine OHA otkriva mobilnu platformu otvorenog koda baziranu na Linux jezgri koja se naziva Android. Android platforma je prilagodljiva platforma pa se koristi na velikom broju uređaja kao što su pametni telefoni, satovi, televizori, pa čak i automobili. Android je zasnovan na Linux jezgri 2.6, te je napisan u C/C++ programskom jeziku. Iako je Android napisan u C/C++ programskom jeziku, ali većina aplikacija pisana je u Java programskom jeziku koji koristi Android Software Development Kit (SDK). [4]

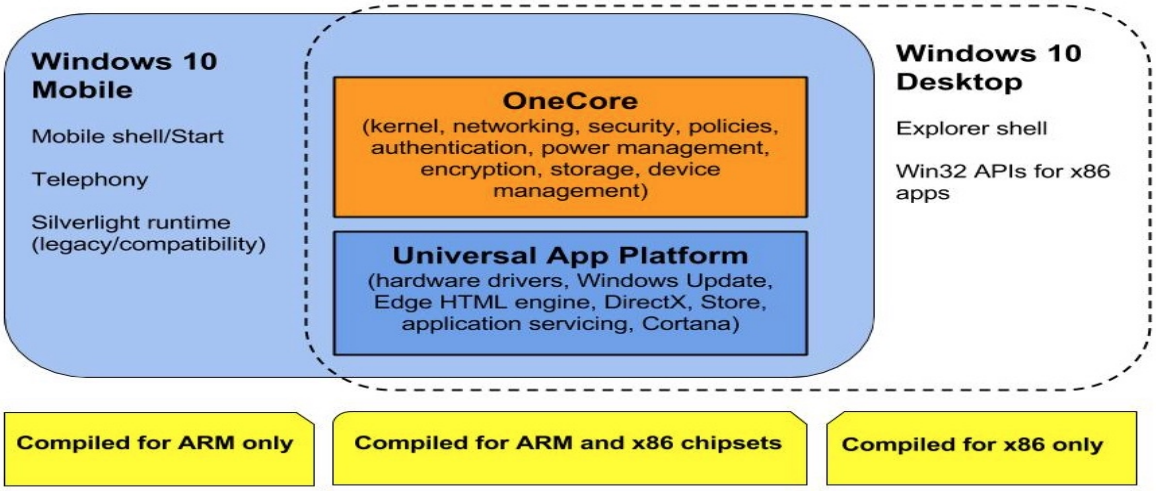


Sl. 2.2. Android arhitektura [5]

Kao što se može vidjeti na prethodnoj slici prvi sloj Android arhitekture čini Linux 2.6 jezgra. Arhitektura operacijskog sustava sadrži biblioteke koje su pisane u C/C++ programskom jeziku, kao što su SQLite, WebKit, libc, OpenGL, Surface Manager i druge. Aplikacije koriste ove biblioteke da bi izvršavali određene zadaće kao što su upravljanje bazama podataka, snimanje i reproduciranje audio i video formata, nadziranje iscrtavanja grafičkog sučelja i mnoge druge funkcije. Slijedi Android Runtime, ovaj sloj služi za pokretanje aplikacija. On se sastoji od dvije komponente, a to su Core libraries i Dalvik Virtual Machine (DVM). Core libraries sadrži većinu izvornih biblioteka programskog jezika Java. Dalvik Virtual Machine pokreće aplikacije kao zaseban proces i pretvara Java class datoteke u svoj vlastiti format (.dex). Sljedeći sloj ove arhitekture je Application Framework koji sadrži mehanizme koji omogućavaju pisanje aplikacija i koji dozvoljava upotrebu svih API-ja (eng. *Application Programming Interface*). Ovaj sloj upravlja prozorima, programskim paketima, vrši dohvaćanje i upotrebu trenutne lokacije korisnika, upravlja resursima, itd. Najviši sloj Android arhitekture je Applictions, to je sloj vidljiv krajnjem korisniku. Sastoji se od osnovnih, ugrađenih aplikacija poput SMS programa, kalendara, web preglednika, kao i aplikacija s Android trgovine.

2.3. Windows 10 Mobile

Proces ujedinjavanja Microsoftovih operacijskih sustava započeo je već 2012. izlaskom Windows Phone 8 koji je napustio Windows CE arhitekturu i okrenuo se Windows NT kojeg koristi i Windows 8. Microsoft je prvi put pokazao Windows 10 javnosti 30. rujna 2014. Iz Microsofta je najavljeno kako će Windows 10 biti ujedinjen operacijski sustav za pametne telefone, tablete, računala, Xbox i ostale uređaje. Windows 10 Mobile prvi put je predstavljen 21. siječnja 2015. i za razliku od prethodnih inačica Windows 10 Mobile će biti dostupan i za male tablete. U operacijski sustav će biti ugrađeno runtime okruženje kodnog imena Astoria koje će implementirati većinu API-ja Androida 4.4 KitKat i tako omogućiti i olakšati pokretanje Android aplikacija na Windows 10 Mobile operacijskom sustavu. Windows 10 i Windows 10 Mobile dijele istu arhitekturu sa sitnim razlikama s obzirom na različite značajke pametnih telefona i drugih uređaja koje koriste Windows 10 operacijski sustav. [6]

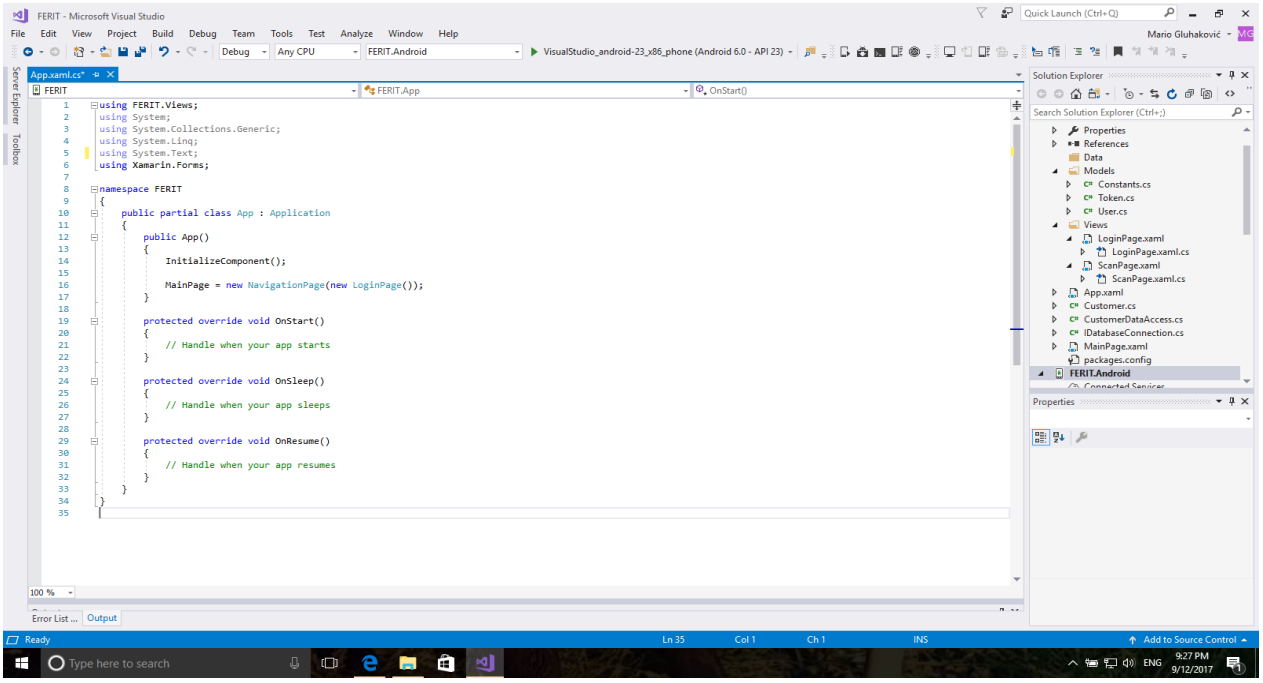


Sl. 2.3. Windows Universal Platform arhitektura [7]

3. TEHNOLOGIJE ZA RAZVOJ VIŠEPLATFORMSKIH MOBILNIH APLIKACIJA

3.1. Xamarin

Uz pomoć ovog alata, mogu se napisati gotovo izvorne aplikacije za Android, iOS i Windowse 10, koristeći programski jezik C#. Riječ je o Microsoftovom jeziku koji donosi hrpu stvari i možemo ga usporediti s Javom s kojom dijeli dobar dio mogućnosti, no C# je uvijek korak ili dva ispred Jave. Razvijanje aplikacija u Xamarinu se vrši uz pomoć spomenutog jezika, no odvija se praktički u dva dijela – prvi dio tiče se isključivo pozadinskog nevidljivog dijela, odnosno onog nevidljivog dijela koji upravlja aplikacijom i poslovnom logikom, a drugi dio tiče se korisničkog sučelja. Stvar je tehničke prirode jer još uvijek ne postoji način da se 'nacrtaju' forme i da ih se iskoristi za sve platforme, no kao što je već rečeno, 90 posto koda može se iskoristiti na sve tri platforme. Uz to, Xamarin nudi i izvorne SDK uz pomoć kojih se mogu iskoristiti specifične mogućnosti vezane uz svaku platformu. To znači da će se moći stvoriti izvornu aplikaciju koja se praktički neće razlikovati od one koja je napravljena, recimo, u Javi i isključivo za Android. Xamarin omogućuje da se paralelno razvija aplikaciju za više platformi. To naravno znači i više posla oko određenih funkcionalnosti. Performanse ovakvih aplikacija su odlične, jer Xamarin daje izvorne aplikacije koje u potpunosti iskorištavaju hardver i softver uređaja na kojem se pokreću tako da u praksi nije vidljiva razlika u odnosu na izvorne aplikacije. Microsoft je odlučio da Xamarin SDK bude otvorenog koda, tako da će se i tu moći iskoristiti puni potencijal alata koji se koristi.



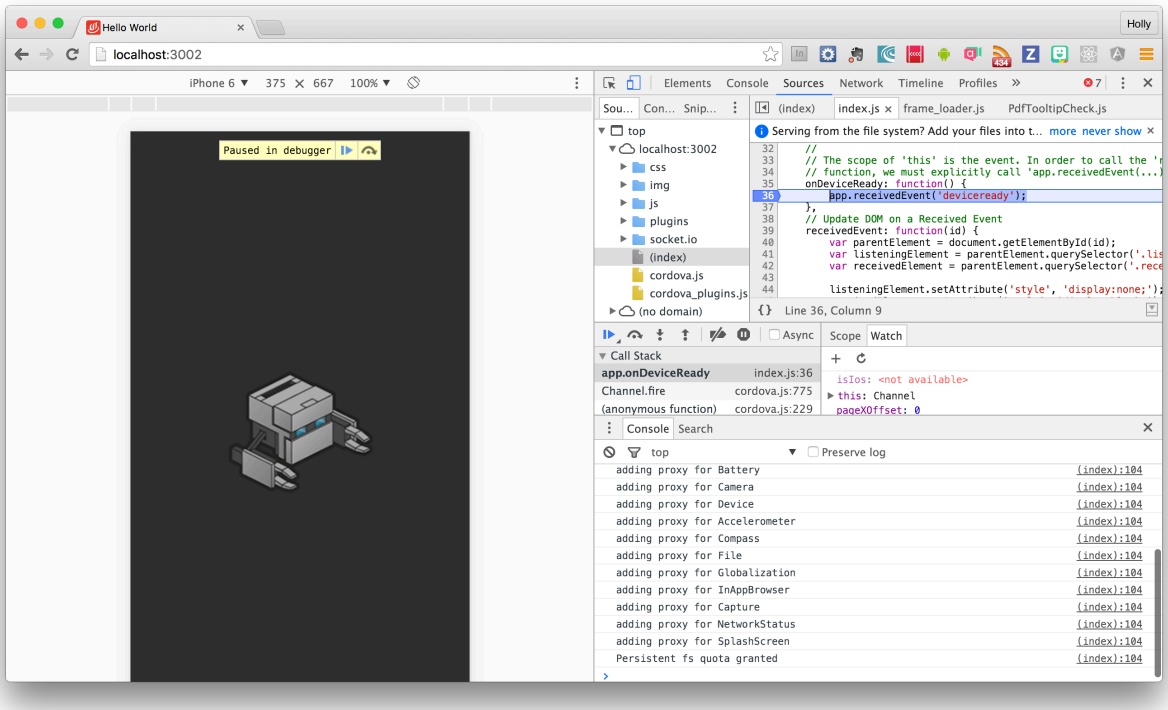
Sl. 3.1. *Xamarin* razvojno okruženje

3.2. Phone Gap

Druga aplikacija vrijedna spomena je PhoneGap. PhoneGap je, kao i Xamarin, projekt otvorenog koda, točnije okruženje otvorenog koda koji služi za brzu izgradnju mobilnih aplikacija. On podupire veliki broj mobilnih platformi – iOS, Android, Blackberry, Windows phone, Symbian i Tizen. PhoneGap je namijenjen isključivo onim programerima koji žele probati napraviti mobilne aplikacije koristeći HTML5, CSS3 i JavaScript. To su većinom web programeri koji nisu vjerojatno imali doticaja s drugim programskim jezicima ili se u profesionalnom smislu ne bave razvojem mobilnih aplikacija. HTML5 i CSS3 su odlični za izradu grafičkog sučelja, a logika se rješava uz pomoć JavaScripta. Naravno, JavaScript služi za pristupanje izvornim funkcionalnostima mobilnog telefona, tako da treba biti svjestan činjenice da u određenim segmentima ipak postoje ograničenja. S druge strane, naravno da se s time može napraviti vrhunsku aplikaciju i imati odlične performanse bez obzira da li se aplikaciju vrti na Androidu, iPhoneu, Windowsima ili drugim mobilnim platformama.

Prednosti PhoneGapa su kodiranje aplikacija koristeći HTML, CSS, JS, umjesto Objective-C i Jave, mnogo je lakše za održavanje i izgradnju koda te sadrži manje koda za određene funkcionalnosti. Grafičko sučelje je čišće, jednostavnije i podržava veliki broj mobilnih platformi.

Međutim PhoneGap ima i svojih loših strana. PhoneGap ne podržava sve mogućnosti koje bi inače telefoni mogli imati da je aplikacija razvijena u nativnom okruženju. Također, PhoneGap se dosta rijetko ažurira pa ponekad ne koristi neke specifične funkcije koje su kasnije predstavljene, te ponekad aplikacije za određene platforme trebaju dodatnu prilagodbu. [8]



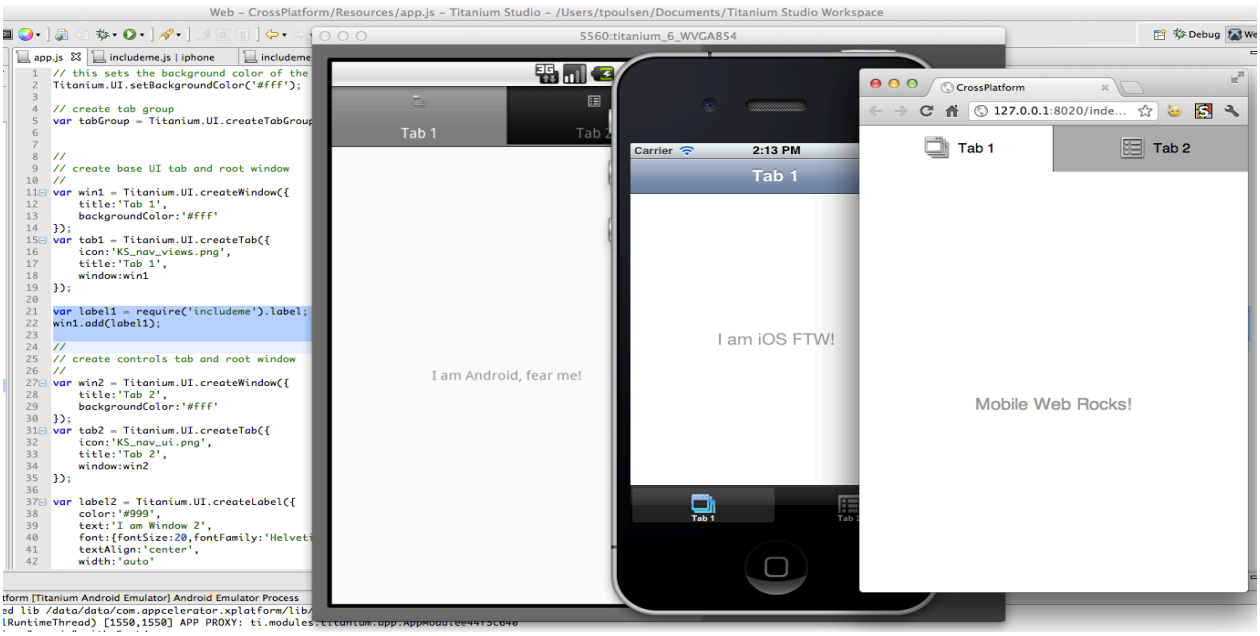
Sl. 3.2. Phone Gap razvojno okruženje [9]

3.3. Appcelerator Titanium

Treće višeplatformsko rješenje koje je potrebno spomenuti je Titanium. Proizvod iza kojeg stoji velika kompanija Appcelerator koja se specijalizirala u izradi alata za izradu mobilnih aplikacija. Prvo su pružali podršku samo za razvoj Android i iPhone aplikacija, pa su tijekom 2009. godine dodali i podršku za iPad, a godinu dana kasnije i za Blackberry uređaje. Najnovija podrška je za Tizen. Titanium nije samo rješenje za izradu aplikacija, nego punokrvna platforma koja pruža jako puno korisnicima. Appcelerator platforma ima tri segmenta koji omogućuju brzi razvoj aplikacija. To su alati za izgradnju aplikacije, API-ji i alati za analitiku koji vam pomoći u optimiziranju aplikacije.

Appcelerator ima svoj studio koji se skine (zajedno sa SDK-om) i u kojem se nalaze svi potrebni alati da bi se razvila aplikacija. Nije potrebno ništa dodatno skidati, sve će studio sam

skinuti i implementirati. Jezik koji se koristi je javaScript u kombinaciji s HTML5 i CSS3, kao i kod PhoneGapa. Sa Titaniumom je moguće pisati i desktop aplikacije u programskim jezicima Pythonu, Rubyju i PHP programskom jeziku. Ugrađenih aplikacijsko programskih sučelja ima mnogo. Ovdje se mogu naći aplikacijsko programska sučelja za spajanje na društvene mreže (Google+, Twitter, Facebook, LinkedIn …), servise za online plaćanje, spajanje na SAP sustave, spajanje na Oracle sustave i druge razne API-je koji mogu dobro doći kod razvijanja aplikacija. Dobra aplikacija nije samo aplikacija koja ima lijepo sučelje i koja dobro radi. Dobra aplikacija je optimizirana, nema memorijskih rupa, koristi najsigurnije mehanizme za pohranu i pristupanje podacima. Titanium je platforma u oblaku, pa se tako mogu prikupljati podatci o svojim korisnicima i analizirati ih. Naravno, neće se moći poput Googla vidjeti sadržaj koji se nalazi unutar aplikacije koja je napravljena, no zato će se moći dobiti svi drugi relevantni podaci. [8]



Sl. 3.3. Titanium razvojno okruženje [10]

4. XAMARIN TEHNOLOGIJA

4.1. Problemi višeplatformskog razvoja aplikacija

Industrija osobnih računala je posljednjih godina doživjela velike promjene. Naravno stolna računala će uvijek biti prisutna za zadatke koji imaju potrebu za tipkovnicama i velikim ekranima, ali osobna računala sada doživljavaju promjenu u vidu manjih uređaja za široke narodne mase. Tableti i pametni telefoni su postavili nove osnove interakcije korisnika i računala primarno preko ekrana na dodir i tipkovnica koje se pojavljuju samo kada je to potrebno dok se ostala interakcija izvodi dodirivanjem na određena područja na ekranu.

Iako se tržište mobilnih uređaja sve više širi trenutno su najrasprostranjenije dvije platforme na pametnim telefonima, a to su Apple proizvodi koje pokreće iOS operacijski sustav i drugi uređaji pokretani pomoću Android platforme koja pokreće razne tipove pametnih telefona. Naravno, tu postoji i treća platforma koja nije toliko popularna kao iOS i Android ali uključuje tvrtku s dugom povijesti razvoja računala, a to su Microsoft Windows Phone i Windows 10 Mobile platforme.

Iako sve tri platforme uključuju slične načine prikaza grafičkog sučelja i interakcije s uređajem kroz zaslon na dodir postoje mnoge razlike među njima. Svaka od ovih platformi ima različit način kretanja kroz aplikaciju i stranice aplikacije, različite načine prikazivanja podataka i prikaza na zaslonu, te različitog pristupa dodiru zaslona. Jedan od problema s kojim se programeri susreću je korištenje sofisticiranih integriranih razvojnih okruženja (IDE) za svaku pojedinu platformu. Za razvoj iOS aplikacija se koristi Xcode na Mac računalima, za razvoj Android aplikacija se koristi Android studio na raznim operacijskim sustavima, dok za razvoj Windows aplikacija koristi *Visual studio*. Sve tri platforme su bazirane na različitim operacijskim sustavima s različitima aplikacijsko programskim sučeljima. U većini slučajeva ove platforme koriste sličan tip korisničkih sučelja ali pod različitim imenima. Pa tako, na primjer, svaka platforma omogućava promjenu Boolean vrijednosti na sličan način ali kontrole koje se koriste imaju različite nazive. Android koristi Switch, iOS koristi UISwitch, a Windows koristi ToggleSwitch. Naravno postoje i veće razlike od samoga imena u korisničkom sučelju. Jedan od problema je također što je svaka platforma usko vezana za pojedini programski jezik. Objective-C se koristi za razvoj iOS aplikacija, Java za Android aplikacije, a C# za Windows aplikacije. Sva tri navedena programska jezika su objektno orijentirani programski jezici i svi su nastali iz C programskog jezika. Međutim, ovi programski

jezici su s vremenom postali jako udaljeni. Zbog toga tvrtke koje žele razvijati višeplatformske aplikacije zapošljavaju veći broj timova od kojih je svaki specijaliziran za određeni programski jezik i aplikacijsko programsko sučelje.

4.2. C#, .NET i djeljivost koda

Microsoft je 2000. godine predstavio C# te je za razliku od Jave i Objective-C poprilično nov programski jezik. C# je prilično jednostavan, snažno tipiziran, objektno orijentiran programski jezik, zasigurno pod utjecajem C++ i Jave, ali s mnogo čišćom sintaksom od C++ i Jave. Prva verzija C# je imala podršku na jezičnoj razini za svojstva i događaje koji su posebno pogodni za programiranje grafičkih korisničkih sučelja. C# se tijekom godina mijenjao i proširivao tako da sada podržava veliki broj funkcija i asinkronih operacija. Od svog nastanka C# je usko povezan s Microsoft .NET *Frameworkom*. Na najnižoj razini .NET pruža infrastrukturu za C# osnovne tipove podataka, ali proširiva .NET biblioteka omogućava podršku za veliki broj tipova podataka i funkcija za matematičke operacije, sigurnost, web usluge i druge.

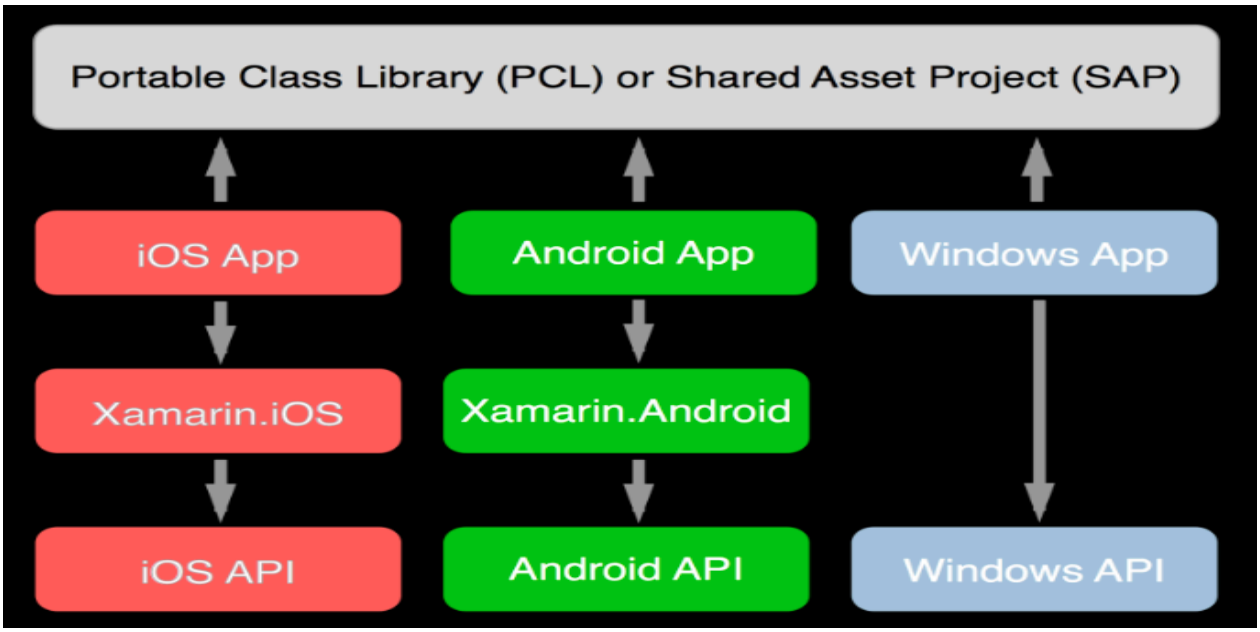
Tijekom godina svoga postojanja Xamarin se fokusirao na razvoj prevoditeljskih tehnologija s tri osnovne grupe .NET biblioteka. Ova grupa biblioteka je poznata kao Xamarin platforma. Ove biblioteke sadrže .NET verzije izvornih Mac, iOS i Android aplikacijsko programskih sučelja. One omogućavaju programerima da u C# programskom jeziku pišu izvorne aplikacije za ove tri platforme. Za razvoj Xamarin aplikacija moguće je koristiti dva ponuđena integrirana razvojna okruženja, a to su *Visual studio* i *Xamarin studio*. O razvojnom okruženju i operacijskom sustavu na računalu će ovisiti opseg razvoja aplikacija, jer su neki od operacijskih sustava i razvojnih okruženja ograničeni na samo određene mobilne platforme.

	MACOS	WINDOWS	
Development Environment	XAMARIN STUDIO	VISUAL STUDIO	XAMARIN STUDIO
Xamarin.iOS	Yes	Yes (with Mac computer)	No
Xamarin.Android	Yes	Yes	Yes
Xamarin.Forms	iOS & Android only	Android, Windows, Windows Phone (iOS with Mac computer)	Android only
Xamarin.Mac	Yes	Open project & compile only ^	No

Sl. 4.1. Razvojna okruženja [11]

Nedavno je Mac operacijski sustav dobio podršku za Visual studio razvojno okruženje, ali se u odnosu na Xamarin studio samo razvojno okruženje promijenilo, a ograničenja su ostala ista. Visual studio je postao dostupan za Mac operacijski sustav da bi Microsoft ujedinio sve Xamarin programere na jednom razvojnom okruženju jer je sredinom 2016. godine Microsoft otkupio sva prava na Xamarin.

Prednost fokusiranja na više platformi je jedan programski jezik koji proizlazi iz mogućnosti dijeljenja koda među aplikacijama. Prije nego što kod bude djeljiv aplikacije moraju biti strukturirane tako da podržavaju takvo dijeljenje. Praktički od kad se raširilo korištenje grafičkog korisničkog sučelja programeri su uvidjeli važnost razdvajanja aplikacijskog koda u funkcionalne slojeve. Jedan od najkorisnijih podjela je između koda korisničkog sučelja i pozadinskog datotečnog modela i algoritama. Popularni MVC (engl. *Model-View-Controller*) model aplikacijske arhitekture dijeli kod na tri dijela, a to su pozadinski podatci, vizualni prikaz i kontrole. Noviji model koji je nastao na osnovu MVC modela je MVVM (engl. *Model-View-ViewModel*) model koji također dijeli kod na tri dijela, a to su pozadinski podatci, korisničko sučelje (ulazi i izlazi) i podatci koji se prosljeđuju između prva dva dijela. Kada programer ima za cilj razvoj aplikacija za više platformi, MVVM arhitektura pomaže u razdvajanju koda u platformi specifičan kod koji zahtjeva interakciju s aplikacijsko programskim sučeljem i u drugi dio koda koji je neovisan o platformi. Dio aplikacije koji je neovisan o platformi moguće je izolirati i spremiti u zaseban dio projekta koji je moguće dijeliti između platformi. Može se kreirati jedan *Visual studio* solution koji sadržava četiri C# projekta da bi se mogli usredotočiti na razvoj aplikacija za sve tri glavne platforme, dok u Xamarin studiu je moguće razvijati samo iOS i Android aplikacije. Sljedeći dijagram prikazuje ilustraciju veza između *Visual studio* odnosno *Xamarin studio* projekta, Xamarin biblioteka i aplikacijsko programskog sučelja.



Sl. 4.2. Veze razvojnog okruženja i Xamarin biblioteka [12]

Treći redak se odnosi na sve .NET bazirane Windows platforme neovisno o uređaju. Drugi redak su zapravo platformski specifične aplikacije. Ove aplikacije vrše pozive u uobičajenim projektima i Xamarin bibliotekama koje implementiraju izvorna aplikacijsko programska sučelja. Međutim ovaj dijagram ne pokazuje SAP ili PCL pozive prema .NET biblioteci.

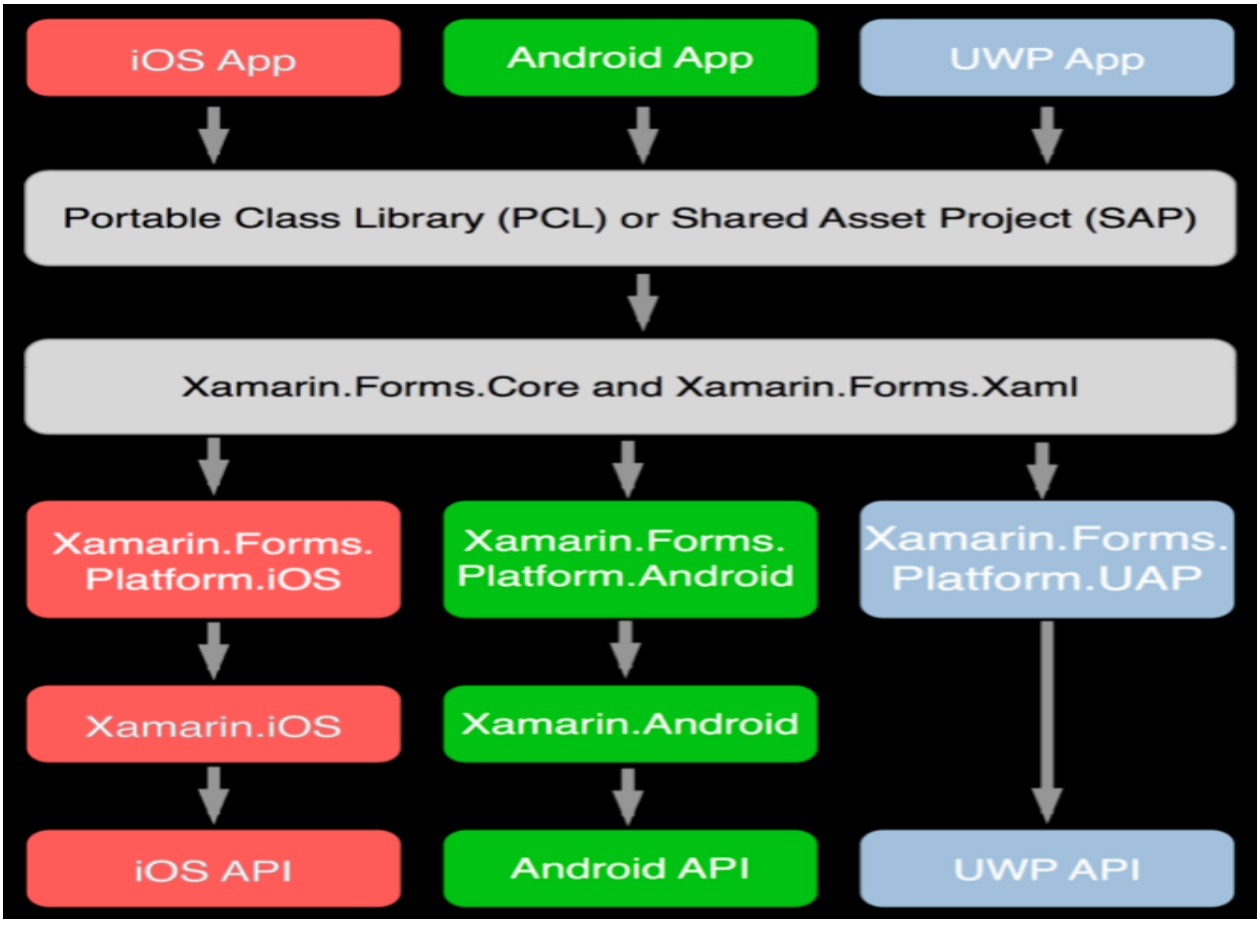
Kada se iOS aplikacija napiše, Xamarin C# prevoditelj stvara C# Intermediate Language (IL) kao i obično, ali onda koristi Apple prevoditelj na Mac-u da bi stvorio izvorni iOS strojni kod isto kao što bi to uradio i Objective-C prevoditelj. Kod koji je napisan će se izvršavati identično kao da je napisan u Objective-C jeziku. Za Android aplikacije Xamarin C# prevoditelj stvara IL, koji se kasnije pokreće na verziji Mono na uređaju uz Java engine, ali su API pozivi identični kao da je program pisan u Java programskom jeziku. Naravno da postoje neke funkcionalnosti pojedinih operacijskih sustava koje nisu djeljive između platformi, ali postoji veliki broj funkcionalnosti koje je moguće podijeliti između platformi tako da će Xamarin u velike olakšati razvoj višeplatformskih mobilnih aplikacija.

Xamarin.Forms također podržavaju XAML (engl. *Extensible Application Markup Language*) kojeg je Microsoft razvio s ciljem brzog oblikovanja i stvaranja objekata. XAML omogućava pisanje korisničkih sučelja za sve mobilne operacijske sustave koji podržavaju dijeljeni kod. Ovaj označni jezik se koristi za prikaz objekata i njihovo oblikovanje nakon čega se u C# vrši implementiranje tih objekata i njihovih funkcija. Premda je XAML kod djeljiv među platformama

ipak postoje neke značajke ovisne o platformi te se one spremaju u različite klase i koriste se samo za određene platforme.

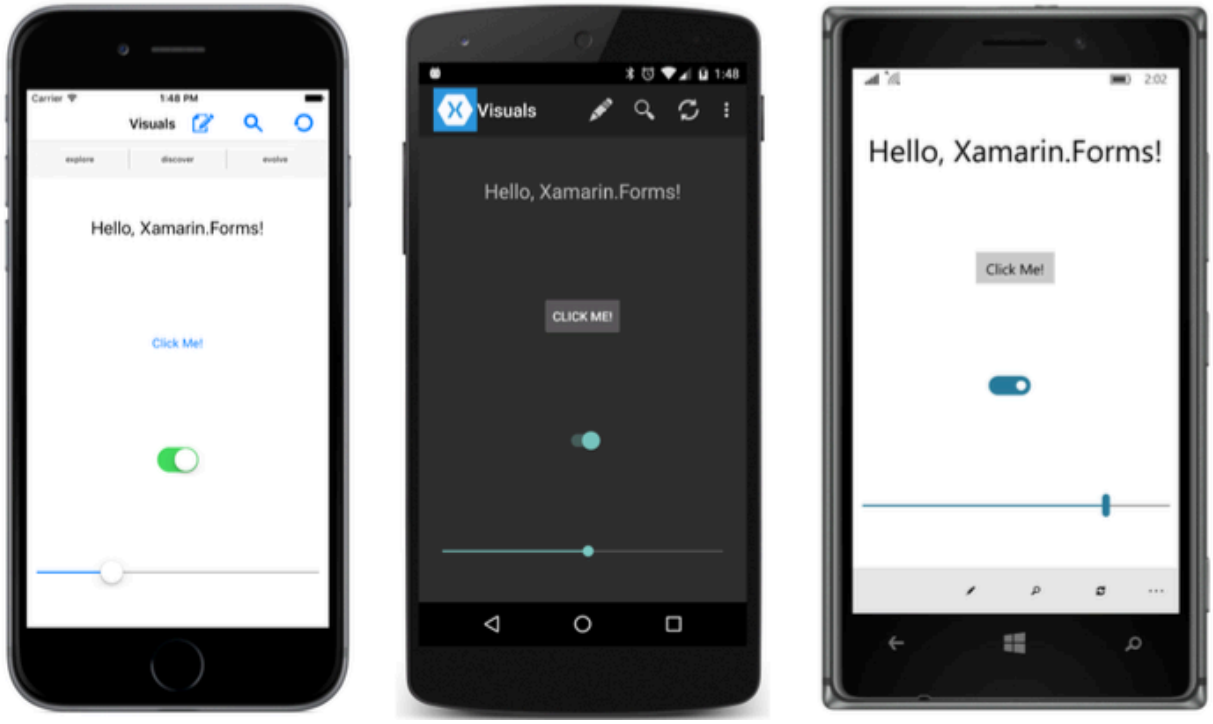
4.3. Xamarin Forme

28. svibnja 2014. Xamarin je predstavio Xamarin.Forms koji je omogućio pisanje korisničkog sučelja kojega je moguće prevesti na iOS, Android i Windows uređajima. Xamarin.Forms podržava pet aplikacijskih platformi: iOS, Android, UWP (engl. *Universal Windows Platform*), Windows 8.1. i Windows Phone 8.1. U *Visual studiju* će postojati šest odvojenih projekata za svih pet platformi i šesti projekt za zajednički djeljivi kod. Pet platformskih projekata u Xamarinu su veoma mali i često sadrže mali dio koda koji se odnosi na pokretanje koda kojeg treba izvršiti. Sljedeći dijagram nam pokazuje iOS, Android i UWP (engl. *Universal Windows Platform*) koja je identična kao druge dvije Windows platforme.



Sl. 4.3. Implementacija Xamarin biblioteka [12]

Xamarin.Forms.Core i Xamarin.Froms.Xaml biblioteke implementiraju Xamarin.Forms aplikacijsko programsko sučelje. Ovisno o platformi Xamarin.Forms.Core biblioteke koriste Xamarin.Forms.Platform biblioteke da bi izvršavao specifične zadatke ovisne o platformi. Ove biblioteke vrše prevođenje određenih funkcija koje se na različitim platformama nalaze pod različitim imenima kao što je funkcija za mijenjanje logičke (engl. *Boolean*) vrijednosti koja se na Androidu naziva Switch, dok se na Windowsu naziva ToggleSwitch. To znači da kada se u Xamarin.Forms-u navede jedna od funkcija koja imaju različita imena ali iste funkcije ta će se funkcija prenijeti na sve platforme i imati istu funkcionalnost na svim platformama. Na sljedećoj slici je prikaz zaslona za sve tri platforme, te one sadrže iste elemente i funkcionalnosti.

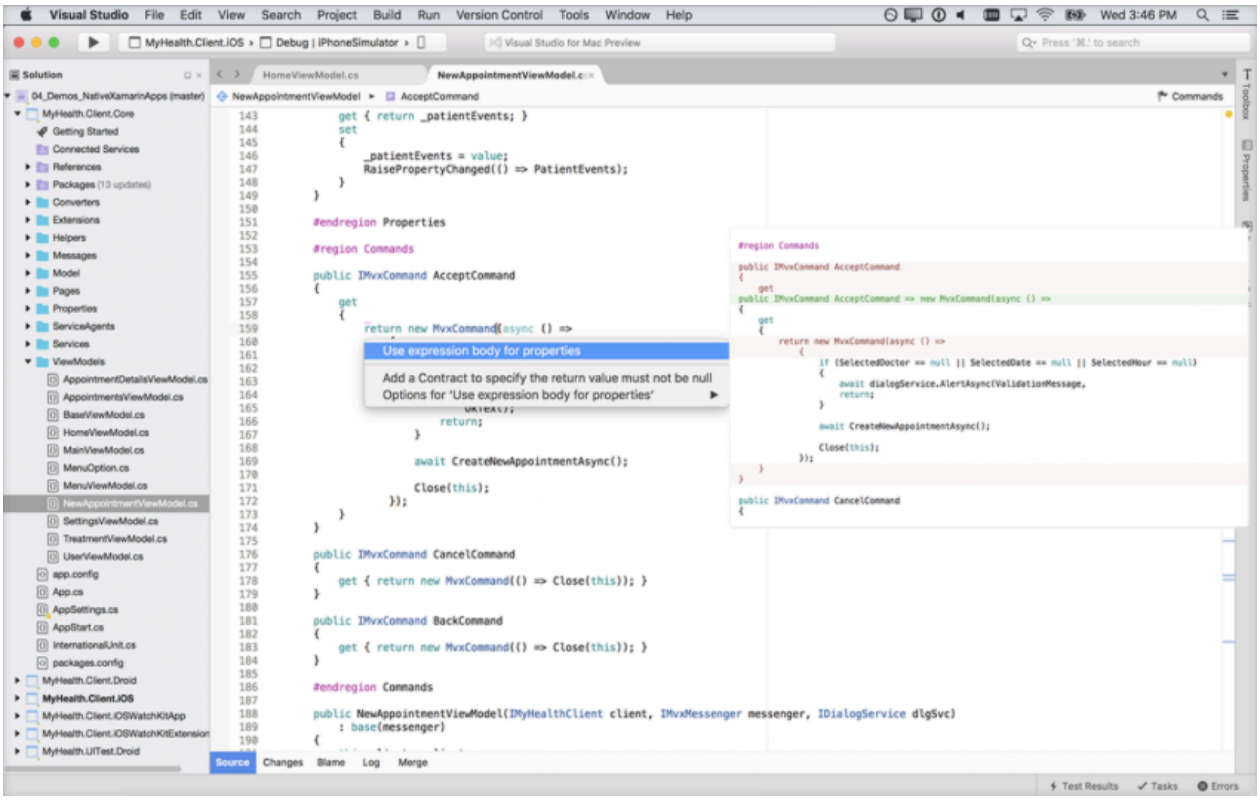


Sl. 4.3. Emulatori za iOS, Android i Windows [12]

Na slici je prikazan iPhone 6 simulator koji pokreće iOS 9.2, LG Nexus 5 simulator koji pokreće Android 6 verziju te Windows 10 koji je pokrenut na Nokia Lumia 935 simulatoru. Kao što se može vidjeti sve verzije aplikacije sadrže iste funkcionalnosti, a to su Button, Switch i Slider, međutim imaju drugačiji prikaz ovisno o platformi. Na slici se nalaze prikazi u vertikalnom položaju, međutim aplikacija zadržava jednaka svojstva i izgled u horizontalnom načinu prikaza.

4.4. Razvojna okruženja i emulatori

Razvoj određenih aplikacija će ovisiti o razvojnom okruženju. Ako se cilja na razvoj aplikacija za iPhone onda će biti potrebno Mac računalo. Apple zahtijeva korištenje Mac računala za razvoj iOS aplikacija. Potrebno je instalirati Xcode na svoje Mac računalo i naravno Xamarin platformu koja sadrži sve potrebne biblioteke. Na Mac računalima je od prije moguće korištenje Xamarin Studija, ali od nedavno je Microsoft pružio mogućnost podrške *Visual Studija* za Mac računala. Na Mac računalu sa Xamarin platformom moguće je razvijati i Android aplikacije. Međutim, ako se želi razvijati aplikacije za Windows platforme potrebno je računalo s Windows platformom i *Visual Studio* 2015 ili noviji. Na takvoj platformi je uz Windows aplikacije moguće razvijati Android aplikacije. Kao što se može uočiti Android aplikacije ne ovise o platformi na kojoj se razvija aplikacija. Aplikacije se mogu testirati na realnim uređajima putem USB kabela ili pomoću emulatora. Postoje prednosti i nedostatci kod oba pristupa testiranju aplikacija. Stvarni uređaji pokazuju kako aplikacija reagira na složenu interakciju i dodir te na brzinu odziva aplikacije. Međutim, emulatori omogućavaju testiranje aplikacije na većem broju i obliku ekrana i ponašanje aplikacije u ovisnosti o njima. Također, emulatori imaju svojih nedostataka u vidu brzine rada. Nekada emulatori zbog veličine i kompleksnosti aplikacije mogu biti spori i troše veliku količinu memorije računala. Tako da se preporučuje kombinacija testiranja na stvarnim uređajima i emulatorima.



Sl. 4.4. *Visual Studio* za Mac računala [13]

4.5. Struktura aplikacije

Moderno korisničko sučelje je sačinjeno od vizualnih objekata različitih tipova. Ovisno o operacijskom sustavu postoje razlike u nazivima objekata. U Xamarin.Forms objekti koji se prikazuju na zaslonu se nazivaju vizualni elementi. Oni dolaze u tri glavne kategorije, a to su *page*, *layout* i *view*. Ovo nisu apstraktni koncepti. Aplikacijsko programsko sučelje definira razrede *VisualElement*, *Page*, *Layout* i *View*. Ove klase sačinjavaju Xamarin.Forms korisničko sučelje, a *VisualElement* objekti sačinjavaju prikaz na zaslonu. Aplikacija u Xamarin.Forms se sastoji od jedne ili više stranica. Stranica uobičajeno zauzima cijeli ekran. Neke aplikacije se sastoje od jedne, a neke sadrže više stranica kroz koje je moguće prelaziti sa stranice na stranicu. Izraz View u Xamarin.Forms se odnosi na porodicu tipova elemenata koji se koriste za ispis sadržaja i interakciju s aplikacijom. Tu spadaju *text*, *bitmaps*, *buttons*, *sliders*, *switches*, *progress bars*, *date and time picker* i dr.

Neovisno da li se koristi *Microsoft Visual Studio* ili *Xamarin Studio*, prilikom kreiranja nove aplikacije koristi se uobičajeni template i stvara se *Solution* koji sadrži pet projekata za specifične platforme i šesti koji sačinjava djeljivi aplikacijski kod. Prilikom kreiranja nove Cross-Platform

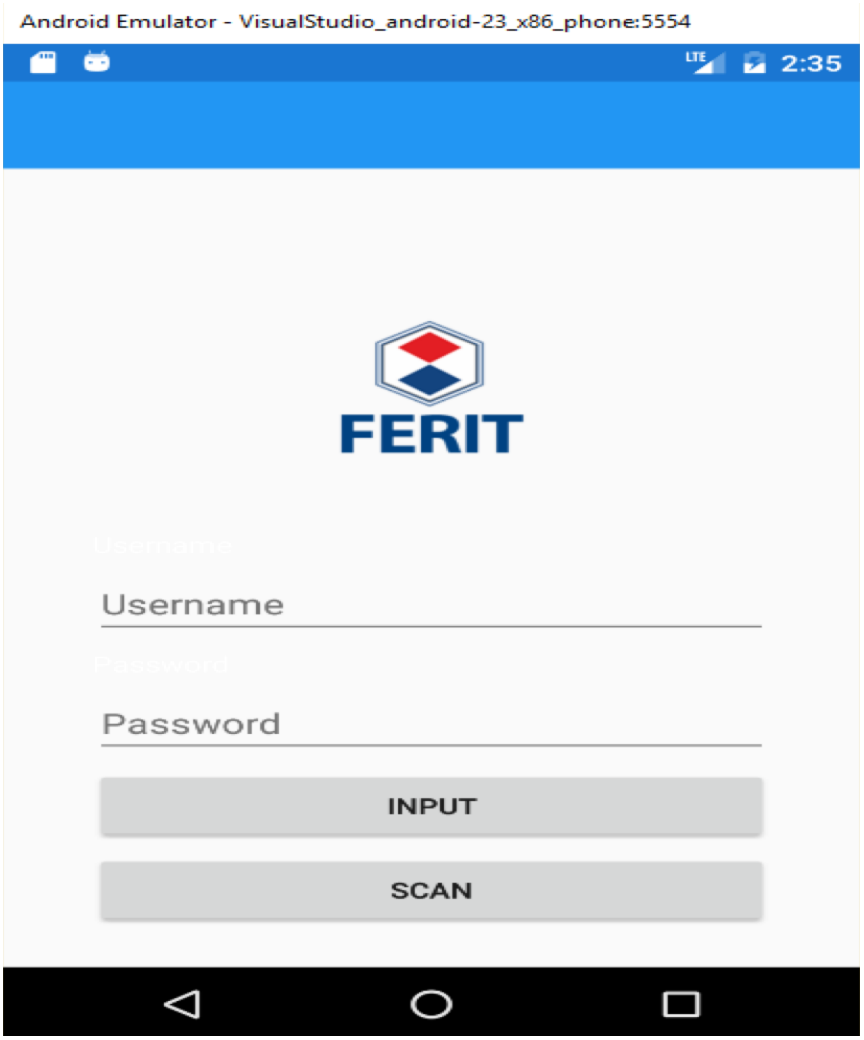
aplikacije Xamarin pruža više dostupnih predložaka: Blank App (Xamarin.Forms Portable), Blank App (Xamarin.Forms Shared) i Class Library (Xamarin.Forms). Izraz Portable se odnosi na Portable Class Library (PCL). Sav sličan aplikacijski kod postaje dynamic-link library (DLL) i referencira se na individualnu platformu. Izraz Shared se odnosi na Shared Asset Project (SAP) i sadrži sav kod koji se dijeli među platformama i postaje značajan dio svake platforme. Ako se odabere Blank App (Xamarin.Forms Portable) razvojno okruženje će stvoriti šest projekata. Prvi projekt će biti Portable Class Library s nazivom projekta koju programer zada. Ostalih pet projekata se odnosi na određene mobilne platforme, pa tako imamo Name.Droid za Android, Name.iOS za iOS, Name.UWP za univerzalnu Windows platformu, Name.Windows za Windows 8.1 i Name.WinPhone za Windows Phone 8.1. Ako koristite Xamarin Studio na Mac računalima neće biti stvoreni Windows projekti jer nisu kompatibilni s razvojnim okruženjem. Prilikom kreiranja projekta u *Visual Studiu* automatski se spremaju i povlače s interneta Xamarin.Forms biblioteke pomoću NuGet package managera. *Visual Studio* sprema te biblioteke u direktorij pod nazivom packages u izvornom direktoriju. Ako se kreira jedan projekat sa SAP, također se može dodati novi projekt s PCL tako da program može imati pristup i jednom i drugom projektu.

5. IZRADA APLIKACIJE

Aplikacija se sastoji od 2 stranice. Prva je log in stranica u kojoj će korisniku biti omogućen upis njegovog ID i lozinke. Pomoću log in stranice se provjerava je li korisnik zaposlenik odgovarajuće tvrtke. Prilikom pokretanja aplikacije provjerava se je li uređaj spojen na bežičnu internet mrežu tvrtke. Ako uređaj nije spojen na bežičnu internet mrežu tvrtke ispisuje se poruka koja korisniku govori da je prije pokretanja aplikacije potrebno uređaj spojiti na bežičnu mrežu. Ako je uređaj spojen na bežičnu mrežu tvrtke prikazuje se log in stranica. Nakon što korisnik unese podatke, vrši se provjera u bazi je li korisnik zaposlenik tvrtke ako nije, ispisuje se poruka o pogrešnom unosu lozinke ili ID oznake. Ako je korisnik zaposlenik tvrtke pokreće se druga stranica aplikacije koja omogućava skeniranje putem NFC. Ako NFC skener na mobilnom uređaju nije uključen ispisuje se poruka korisniku da je potrebno uključiti NFC skener. Neki uređaji imaju opciju uključivanja i isključivanja NFC, dok neki uređaji imaju NFC skener u *stand by* modu rada. Nakon što korisnik skenira NFC oznaku aplikacija provjerava u bazi da li je zaposlenik već prijavljen na posao. Ako je onda ga odjavljuje s posla. Ako nije prijavljen onda ga aplikacija prijavljuje na posao. U bazu se sprema vrijeme prijave, vrijeme odjave zaposlenika i ID zaposlenika.

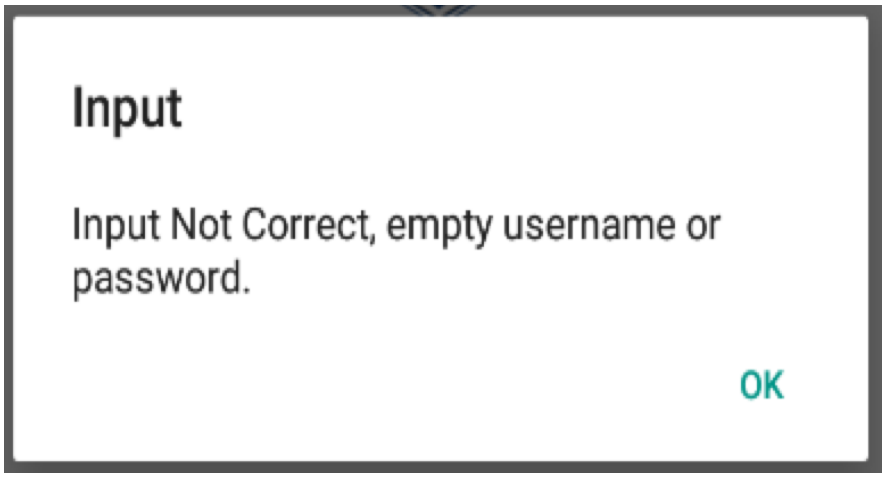
5.1. Unos podataka i provjera povezanosti na bežičnu mrežu

Unos podataka se vrši na glavnoj stranici aplikacije. Od korisnika se traži unos korisničkog imena i lozinke. Nakon unosa korisničkog imena i lozinke potrebno je kliknuti tipku Input da bi se podatci koje je korisnik unio učitali u aplikaciju i da bi se provjerilo je li korisnik unio oba podatka pravilno. Tipku *Scan* nije moguće kliknuti sve dok se ne izvrši pravilan unos podataka. Nakon što se unesu pravilni podatci i klikne se tipka Input program vraća povratnu poruku o točnosti podataka. Ako su podatci točni šalje se poruka o uspješnom unosu, a ako podatci nisu točni šalje se poruka o pogrešnom unosu. Nakon što je unos pravilan i potvrđen klikom na tipku *Scan* se prelazi na stranicu koja omogućava skeniranje NFC.



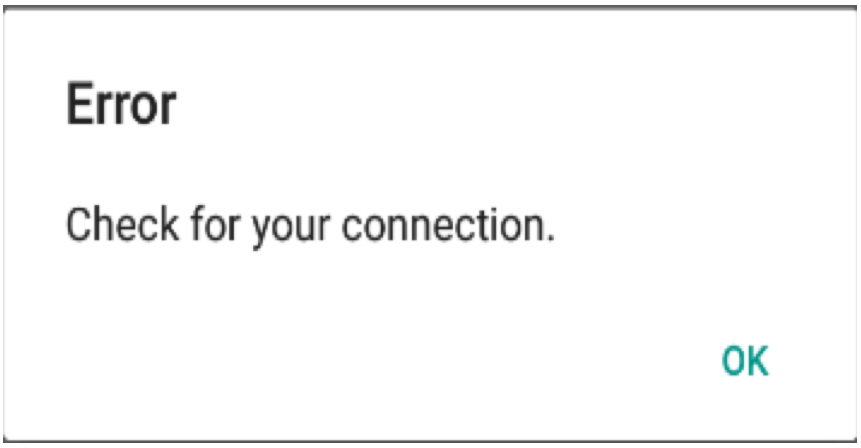
Sl. 5.1. Glavna stranica za unos podatka

Sl. 5.2. Obavijest o točnom unosu podataka



Sl. 5.3. Obavijest o pogrešnom unosu podataka

Na početnoj se stranici još obavlja provjera da li je uređaj spojen s internetskom mrežom, odnosnom bežičnom mrežom tvrtke. Ako je uređaj nije spojen s internetom program ispisuje poruku upozorenja, a ako je spojen program normalno nastavlja s radom.



Sl. 5.4. Obavijest da uređaj nije povezan s internetom

Vizualni dio stranice za unos podataka je pisan u XAML programskom jeziku dok su sve funkcionalnosti implementirane pomoću C# programskog jezika.

```xml
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="FERIT.Views.LoginPage">
    <ContentPage.Content>
        <StackLayout x:Name="MsterLayout">

            <StackLayout x:Name="LogoStack" VerticalOptions="FillAndExpand">

                <Image x:Name="LoginIcon" Source="LoginIcon.png" Margin="0,80,0,0"/>


            </StackLayout>

            <StackLayout x:Name="LoginEntriesStack" VerticalOptions="StartAndExpand">

                <StackLayout.Padding>
                    <OnIdiom x:TypeArguments="Thickness">
                        <OnIdiom.Phone>40,0,40,0</OnIdiom.Phone>
                        <OnIdiom.Tablet>140,150,140,0</OnIdiom.Tablet>
                    </OnIdiom>
                </StackLayout.Padding>

                <ActivityIndicator x:Name="ActivitySpinner" Color="Red" IsRunning="true"/>


                <Label x:Name="Lbl_Username" Text="Username"/>
                <Entry x:Name="Entry_Username" Placeholder="Username"/>
                <Label x:Name="Lbl_Password" Text="Password"/>
                <Entry x:Name="Entry_Password" Placeholder="Password"/>
                <Button x:Name="Btn_Input" Text="Input" Clicked="SignInProcedure"/>
                <Button x:Name="Btn_Scan" Text="Scan" Clicked="ScanProcedure"/>

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

Sl. 5.5. XAML kod stranice za unos podataka

```csharp
private async void Current_ConnectivityChanged(object sender, Plugin.Connectivity.Abstractions.ConnectivityChangedEventArgs e)
{
    if(!e.IsConnected)
    {
        await DisplayAlert("Error", "Check for your internet connection.", "OK");
    }
}

protected async override void OnAppearing()
{
    base.OnAppearing();

    if(!CrossConnectivity.Current.IsConnected)
    {
        await DisplayAlert("Error", "Check for your internet connection.", "OK");
    }
}
```

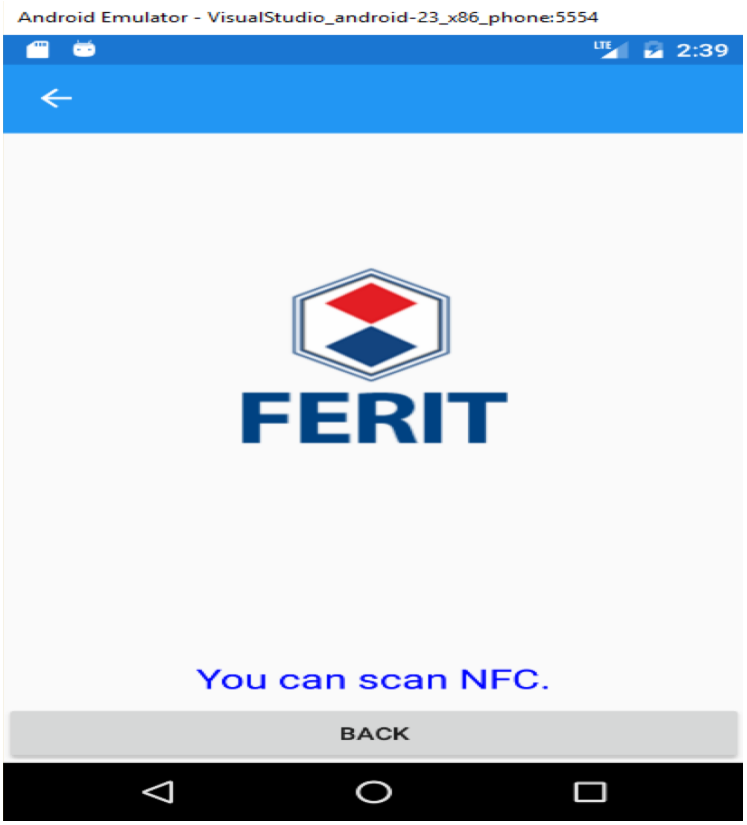
Sl. 5.6. C# kod za provjeru povezanosti s internetom

```
void Init()
{
    BackgroundColor = Constants.BackgroundColor;
    Lbl_Username.TextColor = Constants.MainTextColor;
    Lbl_Password.TextColor = Constants.MainTextColor;
    ActivitySpinner.IsVisible = false;
    LoginIcon.HeightRequest = Constants.LoginIconHeight;

    Entry_Username.Completed += (s, e) => Entry_Password.Focus();
    Entry_Password.Completed += (s, e) => SignInProcedure(s, e);

}
public int a = 0;
protected async void ScanProcedure(object sender,EventArgs e)
{
    if(a==1)
    await Navigation.PushAsync(new ScanPage());
}


void SignInProcedure(object sender, EventArgs e)
{
    User user = new User(Entry_Username.Text, Entry_Password.Text);

    if (user.CheckInformation())
    {
        a = 1;
        DisplayAlert("Input", "Input successful", "OK");
    }
    else
    {
        a = 0;
        DisplayAlert("Input", "Input Not Correct, empty username or password.", "OK");
    }
}
```

Sl. 5.7. C# kod za Input i Scan tipke

5.2. Stranica za skeniranje NFC

Stranica za skeniranje NFC sadrži obavijest da je skeniranje dozvoljeno, te tipku za povratak na stranicu za unos podataka ako je potrebno prijaviti nekog drugog korisnika.



Sl. 5.8. Stranica za skeniranje NFC

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="FERIT.Views.ScanPage">
    <ContentPage.Content>
        <StackLayout>
            <StackLayout x:Name="LogoStack" VerticalOptions="FillAndExpand">

                <Image x:Name="LoginIcon" Source="LoginIcon.png" Margin="0,80,0,0"/>
            </StackLayout>
            <Label x:Name="Lbl_Scan" Text="You can scan NFC." TextColor="Blue"
                   HorizontalTextAlignment="Center"
                   FontSize="Large" VerticalTextAlignment="Center"/>
            <Button x:Name="Btn_Back" Text="Back" Clicked="GoBack"/>
        </StackLayout>
    </ContentPage.Content>
</ContentPage>
```

Sl. 5.9. XAML kod stranice za skeniranje NFC

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

namespace FERIT.Views
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class ScanPage : ContentPage
    {
        public ScanPage ()
        {
            InitializeComponent();
        }

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

Sl. 5.10. C# kod za učitavanje stranice i tipke za povratak na stranicu za unos

5.3. Implementacija baze podataka

Kako bi se obavilo povezivanje i rad s bazama podataka potrebno je u projekt dodati određene biblioteke za rad s bazama. Glavna biblioteka za korištenje baza podataka je SQL biblioteka koja omogućava rad sa SQLite bazama podataka.

Prvo je potrebno napisati SQL kod za generiranje baze i generiranje putanje do nje. Nakon toga se obavlja SQL kod koji stvara tablicu za unos podataka. Za evidenciju dolaska zaposlenika potrebna je jedna tablica koja se sastoji od id unosa, korisničkog imena, lozinke i vremena prijave odnosno odjave. Id unosa je cjelobrojnog tipa i povećava se svakim unosom u bazu za jedan.

Korisničko ime određuje korisnika tj. zaposlenika koji se prijavljuje i njegovu lozinku. Vrijeme prijave i odjave je vrijeme skeniranja NFC.

```
namespace FERIT
{
    [Table("Customers")]
    public class Customer: INotifyPropertyChanged
    {
        private int _id;
        [PrimaryKey, AutoIncrement]

        public int Id
        {
            get
            { return _id;}
            set
            {this._id = value;
             OnPropertyChanged(nameof(Id));}
        }
        private string _username;
        [NotNull]
        public string Username
        {
            get{return _username; }
             set{
                this._username = value;
                OnPropertyChanged(nameof(Username));}
        }
        private string _password;
        [NotNull]
        public string Password
        {
            get {return _password;}
            set{
                this._password = value;
                OnPropertyChanged(nameof(Password));}
        }
```

Sl. 5.11. SQL kod za implementaciju tablice

```
private DateTime _logtime;
[NotNull]
public DateTime LogTime
{
    get { return _logtime; }
    set {
        this._logtime = value;
        OnPropertyChanged(nameof(LogTime)); }
}
public event PropertyChangedEventHandler PropertyChanged;
private void OnPropertyChanged(string propertyName)
{
    this.PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(propertyName));
}
```

Sl. 5.12. SQL kod za implementaciju tablice

Da bi se mogao izvršiti unos prijave i odjave, program koristi procedure za spremanje podataka u bazu. Procedure su napisane tako da omogućuju unos nove vrijednosti i promjenu već postojeće, međutim program koristi samo unos nove vrijednosti. Korištene su proširene procedure radi lakšeg daljnjeg razvoja aplikacije.

```
public int SaveCustomer(Customer customerInstance)
{
    lock (collisionLock)
    {
        if (customerInstance.Id != 0)
        {
            database.Update(customerInstance);
            return customerInstance.Id;
        }
        else
        {
            database.Insert(customerInstance);
            return customerInstance.Id;
        }
    }
}
public void SaveAllCustomers()
{
    lock(collisionLock)
    {
        foreach(var customerInstance in this.Customers)
        {
            if (customerInstance.Id!=0)
            {
                database.Update(customerInstance);
            }
            else
            {
                database.Insert(customerInstance);
            }
```

Sl. 5.13. SQL kod za spremanje podataka u bazu nakon skeniranja NFC

5.4. Implementacija NFC skenera

Kod implementacije procedura za skeniranje NFC se javlja problem kompatibilnosti. Tvrtka Apple, vlasnik licence za iOS operacijski sustav, je zabranila bilo kakvu manipulaciju s NFC značajkom njihovih uređaja radi sigurnosti pri plaćanju preko njihovih aplikacija skeniraj i plati. Ostale platforme dopuštaju manipulaciju NFC značajkom. Zbog ograničenja na Apple uređajima nije moguća implementacija NFC skeniranja kroz dijeljeni kod već je potrebno implementirati

proceduru za skeniranje samo na određenoj platformi kroz razvoj za samo određenu platformu i pisanja čitavog programa na njoj jer Xamarin.Forms ne podržava implementaciju NFC značajki preko neke od svojih biblioteka. S obzirom na to da je tema ovog rada implementacija višeplatformske aplikacije u Xamarinu, a implementacija NFC adaptera kroz dijeljeni kod u Xamarinu nije ostvariva u praktičnom dijelu rada nije implementirana NFC značajka za skeniranje.

Operacijski sustav Android posjeduje posebne biblioteke za rad s NFC značajkama, ali je moguća implementacija striktno na uređajima pokretanim Android operacijskim sustavom. Na sljedećoj slici je prikazana implementacija procedura koje se koriste za manipuliranje NFC adapterom na Android operacijskom sustavu. Ove procedure i funkcije nije moguće implementirati u djeljivi kod jer se koriste samo Android biblioteke i značajke operacijskog sustava. Da bi te funkcije i procedure radile potrebno ih je implementirati pomoću aplikacije pisane za Android operacijski sustav. Za pisanje takve aplikacije potrebno je generirati novi projekt i odabrati template Android App. Nakon kreiranja projekta potrebno je napisati korisničko sučelje u XAML označnom jeziku, te implementirati logički kod koristeći Android biblioteke za manipulaciju NFC značajkom. [14]

```
namespace FERIT.Droid
{

    public class XamarinNFC
    {
        private NfcAdapter _nfcAdapter;

        public void NFCScand(Intent intent)
        {
            if (intent.Action == NfcAdapter.ActionTagDiscovered)
            {
                var tag = intent.GetParcelableExtra(NfcAdapter.ExtraTag) as Tag;
                if (tag != null)
                {
                    var rawMessages = intent.GetParcelableArrayExtra(NfcAdapter.ExtraNdefMessages);
                    if (rawMessages != null)
                    {
                        var msg = (NdefMessage)rawMessages[0];

                        var record = msg.GetRecords()[0];
                        if (record != null)
                        {
                            if (record.Tnf == NdefRecord.TnfWellKnown)
                            {
                                var data = Encoding.ASCII.GetString(record.GetPayload());
```

Sl. 5.12. Klasa s procedurama za rad s NFC adapterom

Za implementaciju NFC značajke na Windows platformama potrebno je kreirati novi projekt i odabrati template Universal Windows Blank app. Nakon odabira vrste projekta potrebno je implementirati korisničko sučelje u XAML označnom jeziku i logički kod u C# programskom jeziku. Da bi se mogla implementirati NFC značajka na Windows platformi potrebno je koristiti posebne NuGet pakete koji omogućavaju manipulaciju NFC značajkom. [15]

6. ZAKLJUČAK

Već dugi niz godina mobilni uređaji kao što su pametni telefoni, tableti i pametni satovi su najprodavaniji elektronički uređaji. Velika većina ljudi nigdje ne ide bez svog mobilnog telefona. Za prijavu zaposlenika na posao veliki broj tvrtki i državnih ustanova koristi magnetne kartice. Velika je vjerojatnost da će zaposlenik takvu karticu zaboraviti ili izgubiti, ali poznato je da mobilni telefon rijetko tko zaboravlja i da su ljudi jako ovisni o njima. Iz tog razloga odlučeno je napraviti aplikaciju koja će pomoću NFC na mobilnim telefonima omogućavati prijavu i odjavu zaposlenika s posla.

Prilikom razvoja aplikacije javio se problem kompatibilnosti NFC značajke s određenim operacijskim sustavima. Sve funkcionalnosti koje je moguće implementirati na dijeljenoj platformi su u potpunosti funkcionalne spremne za korištenje, međutim to nije slučaj s funkcionalnosti NFC adaptera koji se može koristiti isključivo na određenim operacijskim sustavima kao što je Android koji posjeduje svoju biblioteku za rad s NFC adapterom, a koja nije kompatibilna s dijeljenim kodom.

U budućnosti će se vjerojatno napisati zajednička biblioteka za dijeljeni kod koja će omogućiti pisanje aplikacije na dijeljenom kodu i koja će ponuditi funkcionalnost NFC biblioteke za sve platforme. Do tada je potrebno razvijati aplikacije koje koriste NFC značajke na zasebnim platformama koje podržavaju NFC funkcionalnost i koje posjeduju biblioteke za manipulaciju NFC adapterom.

LITERATURA

[1] „Tržišni udio mobilnih platformi“, http://www.gartner.com/newsroom/id/3415117, kolovoz 2016., pristup ostvaren 15. rujna 2017.

[2] „iOS tehnologija“,https://developer.apple.com/library/content/documentation/, rujan 2014., pristup ostvaren 15. rujna 2017.

[3] „iOS arhitektura“, http://www.dotnettricks.com/img/xamarin/ios-architecture.png, pristup ostvaren 25. lipnja 2017.

[4] „Android tehnologija“, https://en.wikipedia.org/wiki/Android_(operating_system), pristup ostvaren 25. lipnja 2017.

[5] „Android arhitektura“, http://allaboutwindowsphone.com/images/features/misc/3windows10architecture.jpg, pristup ostvaren 25. lipnja 2017.

[6] „Windows 10 mobile tehnologija“, https://en.wikipedia.org/wiki/Windows_10_Mobile, pristup ostvaren 25. lipnja 2017.

[7] „UWP arhitektura“, http://allaboutwindowsphone.com/images/features/misc/3windows10architecture.jpg

[8] Boris Plavljanić, „Razvoj višeplatformskih mobilnih aplikacija“, http://pcchip.hr/helpdesk/razvoj-viseplatformskih-mobilnih-aplikacija/, kolovoz 2016., pristup ostvaren 25. lipnja 2017.

[9] „Phone Gap razvojno okruženje“, http://docs.phonegap.com/images/browser-support/chrome-debug-deviceready.png, pristup ostvaren 25. lipnja 2017.

[10] „Titanium razvojno okruženje“, http://docs.appcelerator.com/platform/latest/images/download/attachments/29004890/crossplatform.png, pristup ostvaren 25. lipnja 2017.

[11] „Xamarin razvojna okruženja“, https://developer.xamarin.com/guides/cross-platform/getting_started/requirements/, pristup ostvaren 25. lipnja 2017.

[12] „Xamarin“, https://developer.xamarin.com/guides/xamarin-forms/creating-mobile-apps-xamarin-forms/, Xamarin Inc., pristup ostvaren 25. lipnja 2017.

[13] „Visual studio za Mac računala“, https://www.visualstudio.com/vs/?os=mac, Microsoft Corp., 2017., pristup ostvaren 25. lipnja 2017.

[14] „Implementacija NFC adaptera za android“, https://developer.xamarin.com/, Xamarin Inc., 2015., pristup ostvaren 21. kolovoza 2017.

[15] Andreas Jakl, „Implementacija NFC adaptera za windows“, https://www.nfcinteractor.com/nfc-ndef-library-now-supports-windows-10-uwp-apps/, 2. veljače 2016., pristup ostvaren 21. kolovoza 2017.

SAŽETAK

Cilj ovog rada je izrada višeplatformske mobilne aplikacije u Xamarinu koja će omogućavati zaposlenicima prijavi i odjavu s posla primjenom NFC tehnologije. Aplikacija omogućava brzu i jednostavnu prijavu na posao, te od njih traži unos ID i lozinke.

Teorijski dio rada obuhvaća opis tri najzastupljenije mobilne platforme Android, iOS i Windows Universal Platform i razvojnih okruženja za višeplatformske mobilne aplikacije od kojih je najvažniji Xamarin. Nadalje, sadrži opis programskog jezika C# u kojemu je pisan logički dio aplikacije i XAML označnog jezika u kojem je pisano grafičko sučelje.

Aplikacija se sastoji od dva glavna zaslona. Početni zaslon koji od korisnika traži unos ID i lozinke, a drugi zaslon omogućava skeniranje NFC čipa. Aplikacija sama provjerava da li se vrši prijava na posao ili odjava s posla pomoću baze podataka.

Ključne riječi: iOS, Android, UWP, Xamarin studio, Xamarin, C#, XAML

ABSTRACT

Multiplatform Mobile Application for Recording Employee Access

The goal of this project was to develop a multiplatform mobile application in Xamarin that will allow employees to sign up and sign out of work using NFC technology. The application enables quick and easy login to the job, and requires ID and password entry.

The theoretical part of the paper includes a description of the three most widely used mobile platforms in Android, iOS and Windows Universal Platform, and the development environment for multiplatform mobile applications, most important of which are Xamarin. Further, it contains a description of the C # programming language in which there is a written logical part of the application and a XAML language with a written graphical interface.

The application consists of two main screens. A home screen that requires a user ID and password entry, and the other screen allows you to scan an NFC chip. The application itself verifies whether employee is made to work or to leave a job using a database.

Keywords: iOS, Android, UWP, Xamarin studio, Xamarin, C#, XAML

ŽIVOTOPIS

Mario Gluhaković rođen je 29.09.1995. godine u Tuzli, Bosna i Hercegovina. Nakon završene osnovne škole, 2010. godine upisuje JU Tehničku školu u Brčkom, Bosna i Hercegovina.

U srednjoj školi se iskazuje na mnogobrojnim natjecanjima iz elektronike i elektrotehnike, te s odličnim uspjehom tijekom školovanja i položenom državnom maturom završava dotadašnje obrazovanje i stječe stručno zvanje Elektrotehničar računarstva. U Osijeku 2014. godine upisuje preddiplomski sveučilišni studij računarstva na Elektrotehničkom fakultetu. Od ostalih znanja i vještina posjeduje određeno znanje engleskog jezika, izrade baza podataka i raznih programskih rješenja, te ima vozačku dozvolu B kategorije.

PRILOZI

CD

-Xamarin projekat u *Visual Studiu*
-Rad u .docx i .pdf formatu