ELEKTROTEHNIČKI FAKULTET UNIVERZITETA U BEOGRADU

APLIKACIJA ZA PRIJAVLJIVANJE PRISUTNOSTI PUTEM BLUETOOTH BEŽIČNE TEHNOLOGIJE

– Master rad –

Kandidat:

Marko Đorđević 2012/3179

Mentor:

Prof. dr ZoranČiča

Beograd, Septembar 2015.

SADRŽAJ

1. UVOD

Sa razvojem telekomunikacija i Internet tehnologija sve se više napreduje ka kreiranju multifunkcionalnih uređaja. S obzirom da je mobilni uređaj jedan od uređaja koji većina ljudi ima uvek sa sobom, postoji tendencija stalnog rasta broja dostupnih aplikacija, odnosno dodatnih usluga koje se mogu izvršiti sa mobilnog uređaja.

Prisutna je tendencija da se sve više funkcija i poslova izvršava preko pametnih telefona, počevši od fotografisanja, plaćanja umesto kreditnih kartica, korišćenja telefona u funkciji ključeva, zatim za iščitavanje identifikacionih podataka i još mnogo toga. Shodno tome, ideja ove teze je da se kreira aplikacija koja će korisniku omogućiti prijavljivanje da je prisutan na određenom događaju. Jedini potreban uslov za rad aplikacije je postojanje Arduino servera na tom događaju nakon čije identifikacije će prijavljivanje biti omogućeno.

U drugom poglavlju biće reči o android operativnom sistemu uopšte, o android programiranju, zatim programskom jeziku koji se koristi kao i radnom okruženju.

Treće poglavlje će dati uputstvo za upotrebu aplilkacije korisniku. Biće detaljno opisan korisnički interfejs i data uputstva šta i kako korisnik mora da uradi ukoliko želi da uspešno pokrene aplikaciju. Takođe, biće prikazan izgled svih ekrana koje će korisnik videti u odgovarajućem delu aplikacije.

U četvrtom poglavlju najviše pažnje će se posvetiti samom programskom kodu. Kako klasama i metodama koje se koriste, tako i samom funkcijom pojedinih delova aplikacije, tačnije objašnjenja šta i kako radi. Takođe, ukratko će biti opisan koncept jednog android projekta. Koje sve foldere sadrži kao i koja je namena svakog od tih foldera.

U zaključku ćemo istaći prednosti i nedostatke razvijene aplikacije. Pokušaćemo da ukažemo na primene koje bi aplikacija mogla da ima. Takođe ćemo izneti smernice budućih mogućih modifikacija na osnovu informacija stečenih prilikom izrade aplikacije.

2.KORIŠĆENI ALATI

2.1Androidprogramiranje

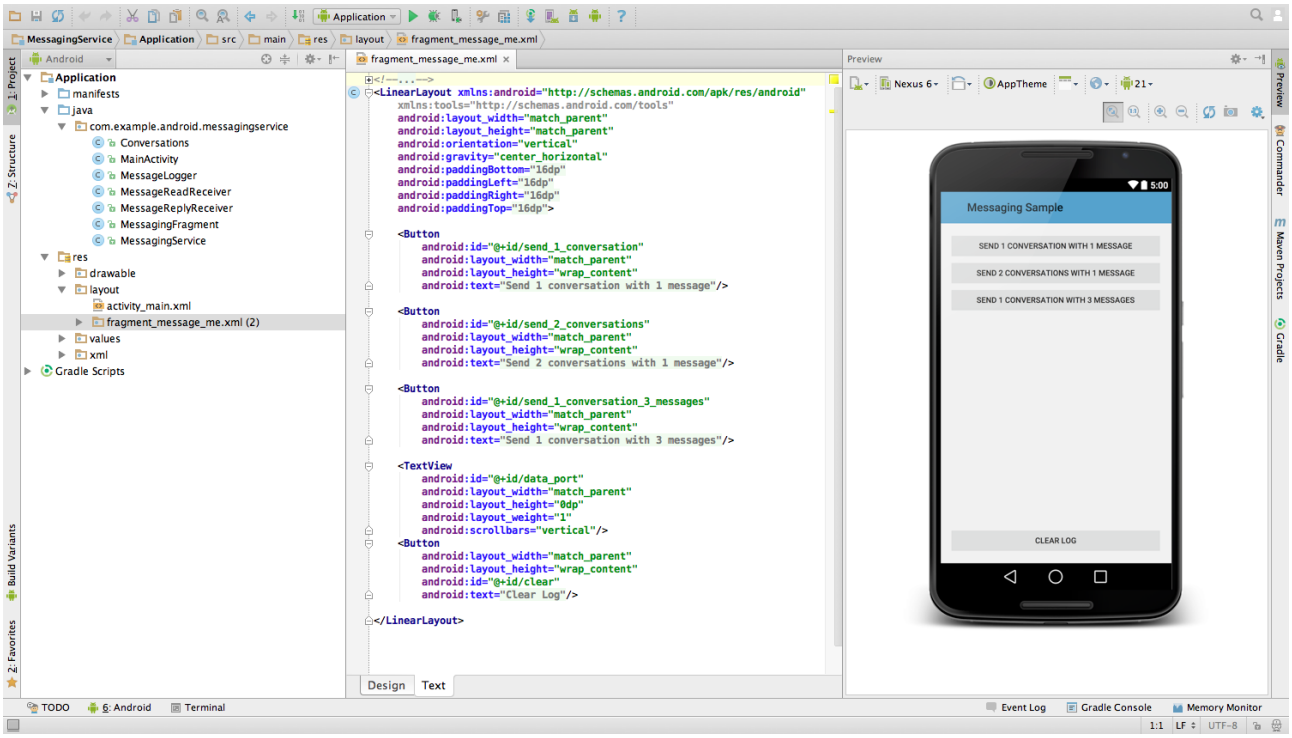
Android je operativni sistem za različite mobilne platforme – pre svega za mobilne telefone i tablet uređaje. Ovaj operativni sistem kreiran je 2005. godine i napravila ga je kompanija Android Inc. Google je preuzeo ovu kompaniju i nastavio razvoj operativnog sistema Android, pa je operativni sistem Android u ovom trenutku vlasništvo Googla. Kod Androida je otvoren (*Open Source*). To znači da bilo ko može preuzeti kompletan kod, koristiti ga i modifikovati prema sopstvenim potrebama. Na ovaj način, različite kompanije za izradu mobilnih uređaja (Sony Ericsson, Motorola, Samsung...) su upravo u Androidu pronašle rešenje za svoju platformu. U odnosu na uređaj na kome se nalazi, svaki od ovih sistema ima sopstvene osobenosti (sadržane pre svega u aplikacijama), ali je osnova na svima ista. Sa stanovišta izgradnje aplikacija, ovo je olakšavajuća okolnost, jer programeri ne moraju pisati različit softver za različite telefone, već jednostavno mogu pisati softver za Android i sve dok se verzija Androida za koju pišu poklapa sa verzijom koju koristi određeni uređaj – softver će funkcionisati.

Jezik koji se koristi za kreiranje programa na Androidu je Java. Java je objektno orijentisan programski jezik, nastao po uzoru na C++. Za razliku od C++ koji se donekle smatra hibridnim jezikom, Java je potpuno objektno orijentisan programski jezik. Sintaksa je slična programskom jeziku C, od koga su nastali i C++ i Java. Java pokušava da reši neke probleme koji se javljaju pri korišćenju programskih jezika C ili C++. Tu se pre svega misli na sigurnost i portabilnost, zbog koje je Java danas jedan od najčešće korišćenih programskih jezika. Način na koji je obezbeđena u isto vreme i portabilnost i sigurnost koda je prevođenje Java koda pomoću Java kompajlera u bajtkod, umesto u izvršni. Bajtkod se zatim pokreće, odnosno interpretira pomoću JVM (*Java Virtual Machine*). S obzirom da JVM kontroliše tok izvršavanja programa, postiže se određena sigurnost od širenja malicioznih programa izvan sistema. Takođe, Java program se može izvršiti na svakom uređaju koji ima JVM.JDK (*Java Development Kit*) je set alata za razvoj, debagovanje i nadgledanje Java aplikacija. U okviru JDK se nalazi JRE (*Java Runtime Environment*), koje opet sadrži implementaciju JVM, zajedno sa standardnim Java bibliotekama.

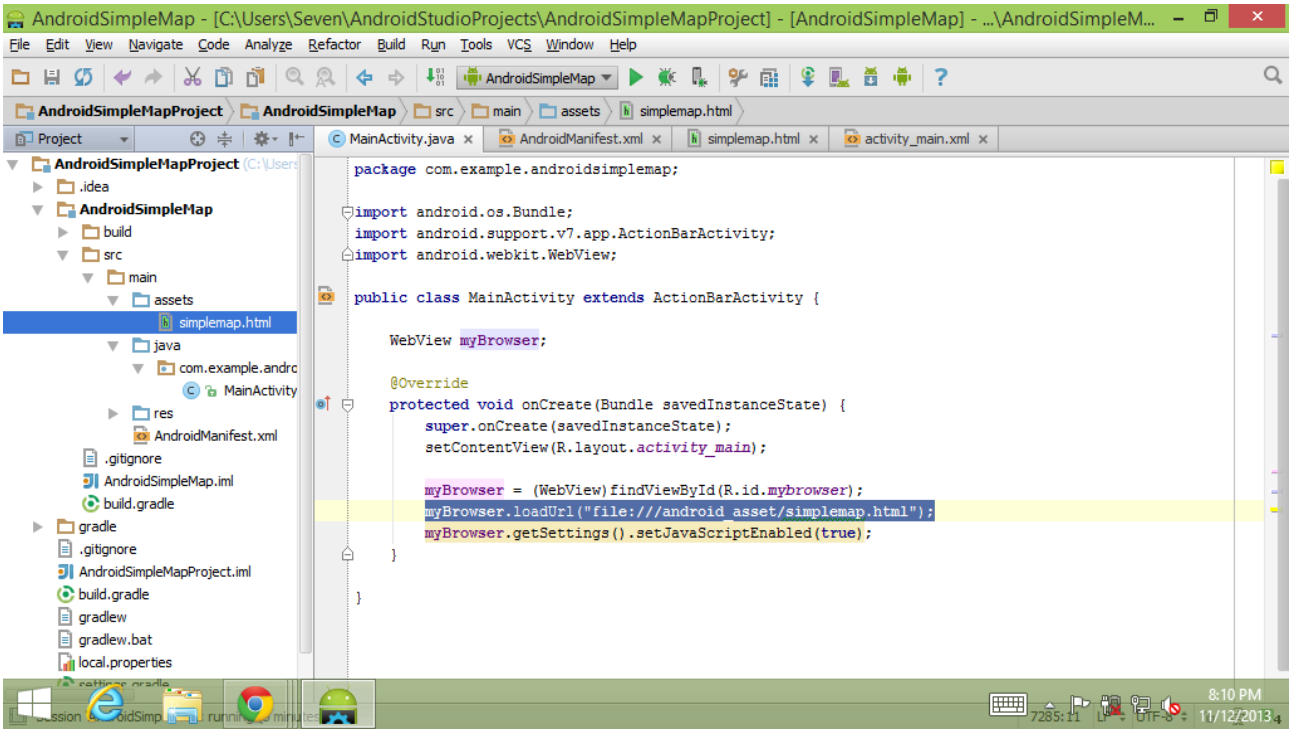
Ipak, platforma pod kojom se izvršavaju Android programi napisani u Javi nije Java virtuelna mašina, što bi se očekivalo – već Dalvik virtuelna mašina. Iako se ove dve virtuelne mašine razlikuju u nekim aspektima, imaju sličnu ulogu i krajnji rezultat: interpretaciju/startovanje bajtkoda. Da bi program mogao da se izvršava u Dalvik virtuelnoj mašini, mora biti preveden u Dalvik bajtkod. Ovo prevođenje nema naročit uticaj na programera, jer je transparentno – okruženje će ga izvršiti automatski. Programer sve vreme radi u Javi, kao što bi radio za bilo koju drugu platformu.

Za kreiranje i izgradnju Android aplikacija, od 2013. godine koristi se Android Studio radno okruženje umesto dotadašnjeg Eclipse radnog okruženja. Pored Java biblioteka koje su ugrađene u ovo radno okruženje, neophodno je instalirati i dodatak pod nazivom Android SDK (*Software*

Development Kit). Android SDK sadrži razne dodatne alate za modelovanje, testiranje, debagovanje itd. Android Studio donosi novi izgradni sistem Gradle. Zatim poseduje bolji editor koda koji programeru značajno olakšava posao. Takođe, kreiranje novog projekta je mnogo lakše kao i razvoj aplikacija koje podrazumevaju više ekrana, grafički korisnički interfejs je dosta bogatiji i lakši za upotrebu. Moguće je kreiranje virtuelnih mašina u više različitih veličina i oblika, pristup Gugl servisima je značajno olakšan kao i niz drugih poboljšanja [1][5].



Slika 1–*Korisnički interfejs u razvojnom okruženju Android Studio*



Slika 2–*Izgled Java fajla u razvojnom okruženju Android Studio*

2.2 PHP i MySQL

Sledeći alati neophodni za realizaciju aplikacije jesu PHP (*PHP: Hypertext PrepProcessor*) i MySQL.

PHP je specijalizovani skriptni programski jezik koji se koristi za izradu dinamičkih Internet stranica, tj. za dinamičko generisanje HTML (*HyperText Mark-up Language*) koda. Pomoću PHP programskog jezika moguće je kreirati HTML stranicu popunjenu dinamičkim sadržajem na serveru pre njenog slanja na lokalni računar klijentu. Na ovaj način, PHP kod koji je generisao stranicu se ne može videti. Po sintaksi je sličan jeziku C, čak ima i iste funkcije. To znači da je jednu radnju moguće izvesti korišćenjem više različitih funkcija.

U ovoj aplikaciji PHP igra ulogu posrednika u komunikaciji između adnroid aplikacije i servera tj. baze podataka. PHP skripte se pozivaju iz android aplikacije svaki put kada je neophodno uneti određene podatke na bazu ili dobiti neku povratnu informaciju sa baze podataka, a koja je bitna za izvršavanje aplikacije. Korišćene su tri različite PHP skripte. Dve su imale ulogu posrednika kada su određeni podaci trebali da budu poslati na bazu, što se u aplikaciji dešava u trenucima kada se korisnik registruje i kada želi da se prijavi da je prisutan na datom događaju. Treći skript ima ulogu posrednika i u suprotnom smeru. Kada korisnik želi da se uloguje, podaci koje je uneo šalju se u bazu i odmah zatim se proverava da li dati podaci postoje u bazi. Nakon provere PHP skript vraća odgovor u android aplikaciju, i u zavisnosti od toga da li su podaci nađeni u bazi ili ne, korisnik dobija informaciju da li je logovanje uspešno obavljeno [2].

MySQL je najpopularniji sistem otvorenog koda za upravljanje bazama podataka. Baza podataka je strukturirana kolekcija podataka. Može biti sve od najjednostavnije liste za kupovinu do kolekcije ogromne količine podataka neke korporacije. Za dodavanje, pristup i obradu podataka koji su smešteni u bazi podataka potreban je sistem za upravljanje bazama podataka kao što je MySQL server. MySQL je sistem za upravljanje relacionim bazama podataka. U relacionoj bazi podataka se podaci smeštaju u više međusobno povezanih tabela. Ovim se dobija na brzini i fleksibilnosti. SQL deo naziva "MySQL" potiče od "Structured Query Language" (strukturirani jezik za upite). SQL je najrasprostranjeniji standardizovani jezik koji se koristi za pristup bazama podataka. MySQL server je veoma brz, pouzdan i jednostavan za korišćenje. MySQL server je prvobitno bio razvijen radi mnogo bržeg upravljanja velikim bazama podataka u odnosu na postojeća rešenja i uspešno se koristi u visokozahtevnim proizvodnim okruženjima veći broj godina. Iako pod stalnim razvojem, MySQL server danas nudi bogat i koristan skup funkcija. Konektivnost, brzina i sigurnost čine MySQL server jako pogodnim za pristup bazama podataka na internetu. MySQL server radi u klijent/server sistemima.

U našoj aplikaciji MySQL baza podataka se sastoji iz dve tabele. U prvu tabelu upisuju se podaci korisnika pri registraciji, i to, ime i prezime, e-mail adresa i šifra. U drugoj tabeli se upisuju podaci korisnika kada korisnik pritisne dugme za prijavljivanje da je prisutan na određenom događaju. Podaci koji se tu nalaze su ime i prezime, e-mail adresa i tačno vreme prijave. Na taj način omogućena je jednostavna evidencija o broju prisutnih na događaju, kao i o njihovim podacima [3].

2.3 Arduino

Arduino je fizičko-računarska platforma (razvojni sistem) otvorenog koda. Hardver se sastoji od jednostavnog otvorenog hardverskog dizajna Arduino ploče sa Atmel AVR procesorom i pratećim ulazno-izlaznim elementima. Softver se sastoji od razvojnog okruženja koje čine standardni kompajler i bootloader koji se nalazi na samoj ploči. Arduino hardver se programira koristeći programski jezik zasnovan na Wiring jeziku (sintaksa i biblioteke). U osnovi je sličan C++ programskom jeziku sa izvesnim pojednostavljenjima i izmenama.

Arduino ploču čine 8-bitni Atmel AVR mikrokontroler sa pripadajućim komponentama koje omogućavaju programiranje i povezivanje sa drugom elektronikom. Bitan aspekt Arduino projekta je standardizovan raspored konektora koji omogućava lako povezivanje sa dodatnim modulima, poznatijim kao štitovi. Arduino mikrokontroleri se isporučuju sa programiranom bootloader komponentom koja pojednostavljuje postupak prebacivanja prevedenog koda u fleš memoriju na čipu. Drugi mikrokontroleri obično zahtevaju zaseban programator. Arduino integrisano razvojno okruženje je aplikacija napisana u Java programskom jeziku. Kreirano je tako da uvede u programiranje učenike, studente i ostale početnike koji nisu upoznati sa načinom razvoja softvera. Sastoji se od uređivača koda sa mogućnostima kao što su označavanje koda, uparivanje zagrada, automatsko uvlačenje linija. Ovaj uređivač može da prevede kod a zatim ga i prebaci u čip jednom komandom. U ovom slučaju nije potrebno podašavati parametre prevođenja koda ili pokretati programe iz komandne linije. Arduino integrisano razvojno okruženje dolazi sa C/C++ bibliotekom zvanom "Wiring" koja čini uobičajene ulazno-izlazne operacije veoma jednostavnim. Arduino programi se pišu u C/C++ programskom jeziku, mada korisnici moraju da definišu samo dve funkcije kako bi napravili izvršni program. Te funkcije su *setup()* , funkcija koja se izvršava jednom

na početku i služi za početna podešavanja i *loop()* funkcija koja se izvršava u petlji sve vreme dok se ne isključi ploča [4].

3. PRIMENA I KORIŠĆENJE APLIKACIJE

Aplikacija je zamišljena tako da korisnik može preko svog telefona da se prijavi da je prisutan na određenom događaju. U svrhu toga potreban je server na kome će biti skladišteni podaci korisnika koji žele da prijave prisutnost. Međutim, aplikacija ne može tek tako da komunicira sa serverom, već joj je potreban posrednik u toj komunikaciji. Da bi se razumelo kako radi aplikacija treba je sagledati kao celinu koja se sastoji iz tri glavna dela. Prvi deo je sama android aplikacija i to predstavlja sve ono što korisnik vidi i uradi na svom mobilnom telefonu. Drugi deo ostvaruje komunikaciju između android aplikacije i servera i to je nešto što se obavlja u pozadini nakon što korisnik izabere određenu opciju u aplikaciji. Treći deo predstavlja server, na kome se smeštaju podaci koje korisnik unosi pri registraciji i prijavljivanju prisutnosti.

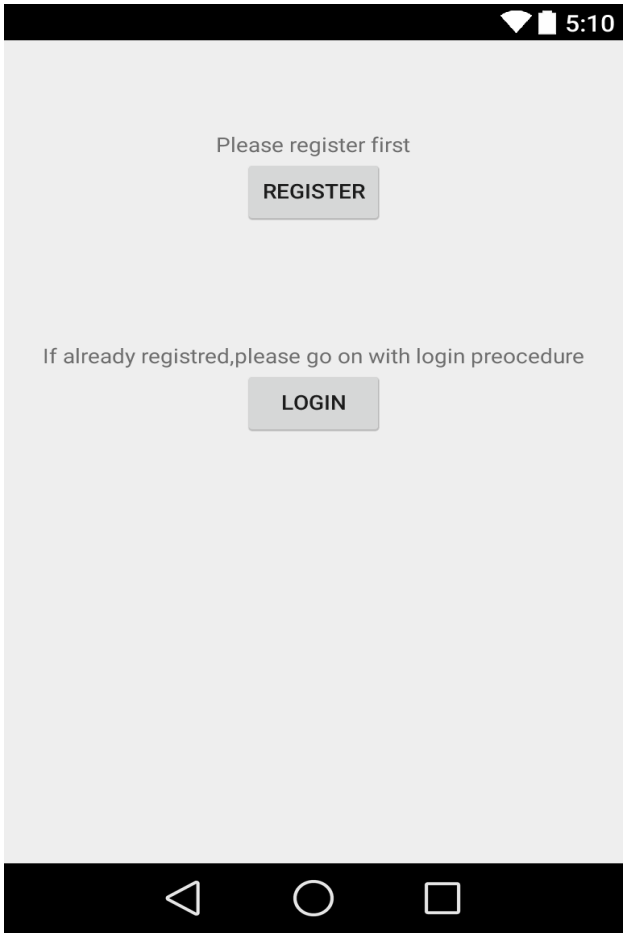
U okviru aplikacije korisnik ima nekoliko radnji koje treba da obavi ukoliko želi da se prijavi da je prisutan na određenom događaju. Prvi i neophodan uslov za korišćenje aplikacije jeste to da korisnik mora da se registruje. Klikom na dugme za registraciju korisniku se otvara ekran gde se nalaze polja u koja mora da upiše svoje ime i prezime, e-mail adresu kao i šifru. Nakon što je uneo svoje podatke, dužan je da pritisne dugme koje će potvrditi tu njegovu registraciju i nakon čega će njegovi podaci biti uneseni u bazu podataka. Korisnik kada je jednom registrovan svaki sledeći put kada koristi aplikaciju, dovoljno je da se uloguje. Logovanje vrši unosom svoje e-mail adrese i šifre u polja predviđena za to. Ukoliko su podaci ispravno uneseni, korisnik dolazi do mogućnosti da uradi ono zbog čega je aplikacija i napravljena tj. da se prijavi da je prisutan na određenom događaju. Još jedan uslov koji mora biti ispunjen za to jeste postojanje prethodno definisanog Arduino servera (pod time se podrazumeva ime Arduino uređaja tj. ime koje će se pojaviti na spisku uređaja sa kojima je moguće upariti korisnikov telefon putem *bluetooth* veze) u prostoriji gde se događaj odvija. Pretraga za odgovarajućim Arduino serverom vrši se putem *bluetooth* tehnologije, koja se aktivira na korisnikovom telefonu. Ukoliko je klikom na dugme korisnik dobio potvrdnu informaciju o postojanju odgovarajućeg Arduino servera, na ekranu mu postaje vidljivo dugme za prijavu. Pritiskom na to dugme šalju se korisnikovi podaci na server i to njegovo ime i prezime, e-mail adresa i tačno vreme kada se prijavio. Time je praktično završeno uspešno korišćenje aplikacije.

Drugi deo koji sam naveo kao jednu od celina od koje se sastoji cela aplikacija jeste deo koji omogućava komunikaciju između android aplikacije i servera. Da nije ovog dela ne bi ni bilo moguće da se podaci koje korisnik unese pri registraciji ili pri prijavljivanju da je prisutan, uskladište u bazi podataka. To se ostvaruje tako što se u PHP skriptu napiše kod koji prvo preuzme unesene podatke iz aplikacije, a zatim izvrši upit nad bazom podataka i u odgovarajuću tabelu upiše te podatke. Takođe, komunikacija između aplikacije i servera vrši se i prilikom logovanja. Tada se komunikacija odvija u oba smera. Prvo se iz aplikacije preuzimaju podaci koje je korisnik uneo pri logovanju, zatim se upitom nad bazom proverava da li su uneti podaci validni tj. da li se podudaraju sa podacima nekog od registrovanih korisnika. Ukoliko je odgovor potvrdan, šalje se povratna informacija u aplikaciju i korisniku se na ekranu prikazuje poruka da je logovanje uspešno obavljeno.

Sve ovo prethodno ne bi imalo nikakvog smisla ukoliko ne bi postojao server na kome će biti skladišteni podaci koje korisnik unosi pri korišćenju aplikacije. Ulogu servera igra MySQL baza podataka koja se nalazi na tačno određenoj IP adresi. U toj bazi se nalaze dve tabele. Jedna se naziva Users, a druga Presence. U tabeli Users nalaze se podaci o korisniku koje on unosi pri registraciji. Znači, ime i prezime, e-mail adresa i šifra. Primarni ključ ove tabele je kolona pod nazivom id, čija se vrednost automatski povećava za 1 nakon svakog novog unosa. Znači prvi korisnik koji se registrovao ima id jednak 1, sledeći jednak 2 itd. Druga tabela sadrži podatke onih korisnika koji su se prijavili da su prisutni na određenom događaju. Ona sadrži kolone za ime i prezime, e-mail adresu, koja je ujedno i primarni ključ i kolonu u koju se upisuje tačno vreme i datum kada se korisnik prijavio. U trenutku kada se korisnik prijavljuje da je prisutan vrši se komunikacija između ove dve tabele. Aplikacija je napravljena tako da se pritiskom na dugme za prijavu, iz nje preuzima samo e-mail adresa korisnika. Pošto nije dovoljno da u tabeli Presence postoji samo podatak o e-mail adresi korisnika koji je prisutan, vrši se upit nad tabelom Users i na osnovu poklapanja e-mail adrese, saznaje se ime i prezime datog korisnika. Na taj način u tabeli Presence imamo i e-mail adresu i puno ime prijavljenog. Takođe, automatski se u tabeli učitava i tačno vreme i datum kada je prijava izvršena. Na ovaj način, vrlo je jednostavna evidencija o tome ko je i kada prisustvovao određenom događaju.

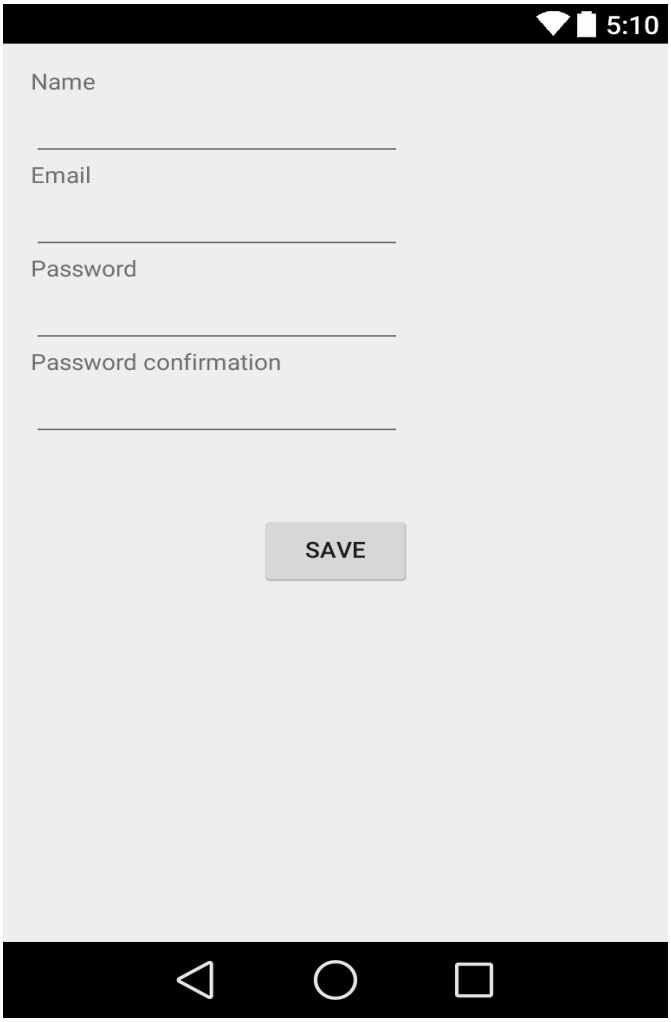
U nastavku ovog poglavlja objasniću ceo ovaj proces postepeno i sa odgovarajućim slikama koje predstavljaju ono što korisnik vidi na ekranu svog telefona u svakom trenutku korišćenja aplikacije.

Prvi ekran koji se prikaže korisniku pri pokretanju aplikacije jeste prozor sa mogućnošću da bira da li želi da se registruje, što je i neophodan uslov ukoliko korisnik uopšte želi da koristi aplikaciju, ili ako je već ranije registrovan, onda pritiska dugme za logovanje.



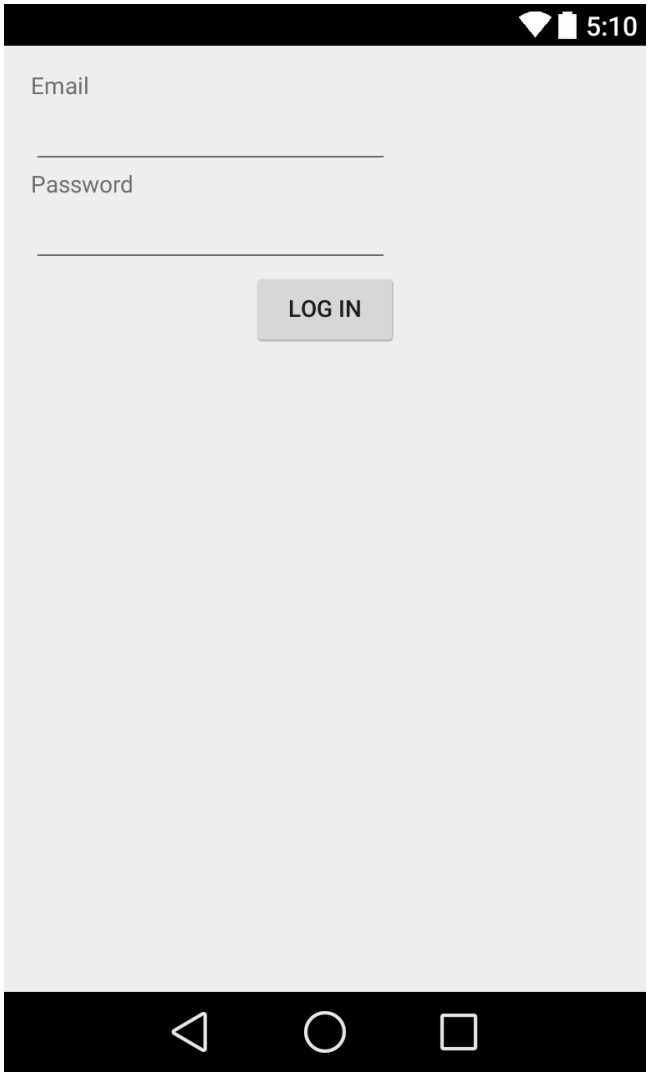
Slika 3 –*Prikaz prvog prozora gde korisnik mora da bira da li će da se registruje ili uloguje*

Ukoliko korisnik prvi put koristi aplikaciju, moraće da se registruje da bi nastavio dalje. Nakon što je pritisnuo dugme **register** na prvom prozoru otvara mu se drugi na kome je potrebno da unese svoje podatke. Zahtevani podaci su ime i prezime u prvom polju, e-mail adresa u drugom, kao i šifra i potvrda te šifre u trećem,odnosno četvrtom polju. Nakon unetih podataka korisnik treba da pritisne dugme **save** kako bi ti podaci bili poslati na MySQL bazu i to u tabelu pod imenom Users.



Slika 4 – *Prozor gde korisnik unosi svoje podatke za registraciju i šalje ih na bazu*

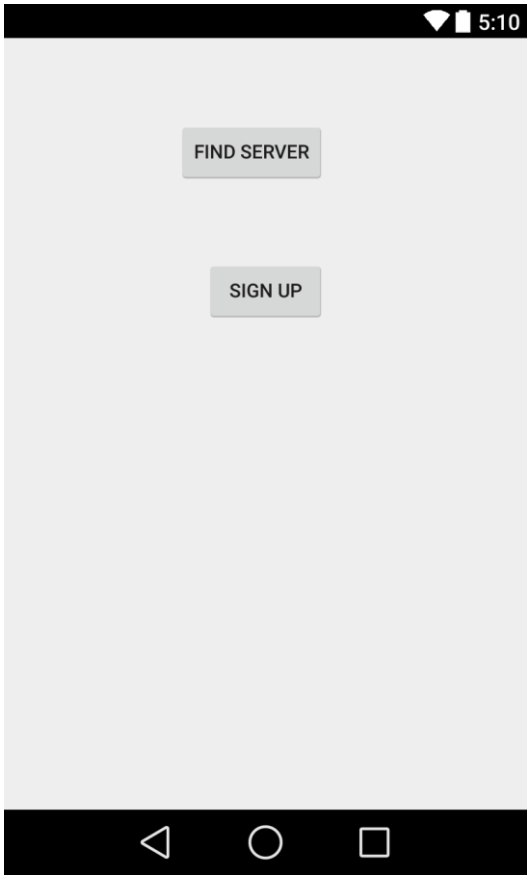
Nakon što je unos podataka u bazu pri registraciji uspešno obavljen, na ekranu će se ispisati poruka o tome i odmah će se otvoriti sledeći ekran, gde korisnik mora da se uloguje kako bi nastavio dalje sa aplikacijom. Korisnik logovanje izvršava tako što u prvo polje upisuje svoju e-mail adresu, a u drugo šifru i pritiskom na dugme **log in**, uneti podaci se proveravaju u bazi. Ukoliko su podaci tačno uneseni,što znači da odgovaraju podacima koje je korisnik uneo pri registraciji, na ekranu će se prikazati poruka o uspešnom logovanju i korisniku će se otvoriti sledeći prozor. U suprotnom, pojaviće se poruka o pogrešno unetim podacima i korisnik će morati da ih ispravno unese u data polja ukoliko želi da nastavi sa aplikacijom.



Slika 5 – *Prozor gde korisnik unosi svoje podatke kako bi se uspešno ulogovao*

Napokon stižemo i do četvrtog, ujedno i poslednjeg i najvažnijeg prozora koji se pojavljuje korisniku pri korišćenju ove aplikacije. Nakon što su uspešno obavljeni procesi registracije i logovanja, korisnik je u mogućnosti da u ovom prozoru izabere opciju da se prijavi da je prisutan na datom događaju. Na ekranu su prikazana dva dugmeta, dugme **find server** i dugme **sign up**. Na početku dugme **sign up** je u takozvanom nevidljivom modu, što znači da na njega nije moguće pritisnuti sve dok se ne ispuni određeni uslov. Taj uslov se ostvaruje na sledeći način: korisnik prvo mora da pritisne dugme **find server**, pri čemu se aktivira *bluetooth* na njegovom telefonu i traže se uređaji u neposrednom okruženju sa kojima može da se upari putem *bluetooth* veze. Ukoliko se ime jednog od prikazanih uređaja poklapa sa jedinstvenim imenom Arduino servera koji se nalazi u prostoriji gde se održava događaj na koji korisnik želi da se prijavi i koji je tu postavljen sa namenom da bude server, korisnikov telefon će se upariti sa spomenutim Arduinom. Nakon što je *bluetooth* veza uspešno uspostavljena dugme **sign up** će postati vidljivo i korisnik će biti u mogućnosti da na njega klikne. Klikom na dugme **sign up** korisnik ostvaruje glavnu svrhu ove aplikacije, a to je prijavljuje se da je prisutan na datom događaju. Prijava se ostvaruje tako što se preuzima korisnikova e-mail adresa koju je uneo u prethodnom delu aplikacije pri procesu

logovanja i šalje u istu bazu podataka kao i ranije, ali ovoga puta u drugu tabelu pod nazivom Presence. Na osnovu e-mail adrese izvlači se korisnikovo ime iz tabele Users, gde su uneti korisnikovi podaci pri registrovanju, a takođe automatski se u bazu podataka unosi i tačno vreme i datum korisnikove prijave za dati događaj. Znači, nakon uspešno odrađenih prethodno opisanih koraka, u tabeli u okviru baze podataka biće prikazani korisnikovo ime, e-mail adresa i tačno vreme i datum kada se prijavio.



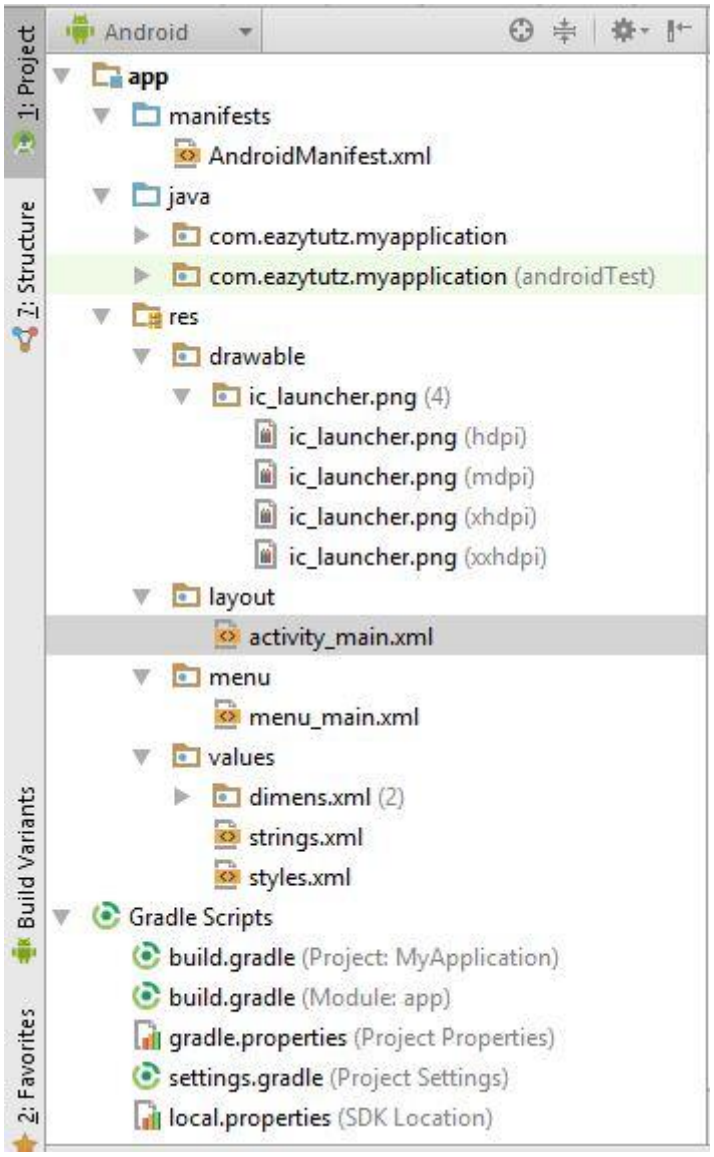
Slika 6 – *Prozor gde korisnik prijavljuje svoje prisustvo*

Aplikaciju je jedino moguće testirati na pravom uređaju tj. mobilnom telefonu. Iako razvojno okruženje Android Studio sadrži emulator koji simulira ulogu pravog uređaja, u ovom slučaju to nije bilo moguće, jer emulator ne podržava *bluetooth* opciju, bez koje ova aplikacija ne bi ni mogla da radi. Takođe, u trenutku testiranja mobilni telefon mora biti povezan sa Internetom kako bi se uspešno uspostavila veza između aplikacije i servera. Bez toga ne bi bila moguća ni registracija, ni logovanje korisnika, a samim tim ni prijava prisutnosti na datom događaju. Naravno, korisnik je dužan da ispravno unese sve podatke koji se od njega traže, u suprotnom dalje izvršavanje aplikacije se zaustavlja i korisniku se prikazuje poruka šta je pogrešio. Na primer, ukoliko je pogrešno uneo e-mail adresu ili šifru pri procesu logovanja, na ekranu će se pojaviti poruka koja ukazuje na postojanje greške i korisnik će ponovo, ovoga puta uspešno, morati da ukuca tražene podatke kako bi aplikacija nastavila sa izvršavanjem. Zatim, ukoliko aplikacija ne prepoznaje odgovarajući Arduino server, korisnik neće biti u mogućnosti da se prijavi da je prisutan tj. dugme

sign up na slici 6 neće biti vidljivo na korisnikovom ekranu. Ukoliko je unošenje podataka pri registraciji i logovanju izvršeno uspešno, poslednji uslov da korisnik može da se prijavi da je prisutan, jeste da putem *bluetooth* tehnologije njegov telefon prepozna prethodno definisani Arduino server. Ako je i to ispunjeno korisnik pritiska dugme kojim se prijavljuje da je prisutan i aplikacija je uspešno izvršena.

4. STRUKTURA I OPIS KODA

Pre opisa strukture koda i realizovanih funkcija i korišćenih biblioteka, prikazaćemo organizaciju jednog android projekta.



Slika 7 – *Prikaz koncepta android projekta*

Kao što se može videti na slici 7, izdvajaju se tri glavna foldera, pod imenima **manifest**, **java** i **res**, a u okviru kojih se nalaze razni drugi podfolderi neophodni za izradu aplikacije.

4.1 AndroidManifest.xml

U okviru **manifest** foldera nalazi se AndroidManifest.xml. To je najvažniji fajl u strukturi android projekta i on sadrži sve informacije o aplikaciji. U trenutku kada napravimo projekat za aplikaciju, ovaj fajl se automatski generiše. Kako dodajemo nove komponente našoj aplikaciji, sve standardne informacije će biti automatski upisane u ovaj fajl. Ukoliko ipak želimo da aplikaciji dodelimo neke posebne dozvole, potrebno je ručno ih dodati u fajl. U slučaju naše aplikacije tri su dozvole dodate.

```
<uses-permission android:name="android.permission.BLUETOOTH"/>
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN"/>
<uses-permission android:name="android.permission.INTERNET"/>
```

Prve dve dozvole se odnose na mogućnost aplikacije da koristi *bluetooth* opciju, dok se treća odnosi na njenu mogućnost pristupa Internetu [5].

4.2 Res folder

U okviru **res** foldera se smeštaju eksterni resursi potrebni za realizovanu aplikaciju, kao što su slike, fajl gde je definisan korisnički interfejs, stringovi, animacije, meniji, audio fajlovi i razni drugi. Podfolderi u okviru **res** foldera jesu sledeći.

Prvi je pod nazivom **layout** i tu se nalazi xml fajl koji definiše korisnički interfejs. Pošto sam u poglavlju o korisničkim primenama i testiranju aplikacije detaljno opisao izgled korisničkog interfejsa u različitim situacijama ovde se neću dalje zadržavati na ovu temu.

Drugi se naziva **values** i tu se definišu stringovi, nizovi, dimenzije, boje, stilovi i slično.

Treći je **menu** u kome se definišu meniji koji se koriste u aplikaciji.

I četvrti se naziva **drawable**. On sadrži tri različita podfoldera, **drawable-ldpi**, **drawable-mdpi** i **drawable-hdpi**. Izvor slika za svaku rezoluciju skladište se u odgovarajući folder i onda sistem sam bira koji će izabrati u zavisnosti od gustine piksela ekrana [5].

4.3 Java folder

U java folderu su smešteni, kao što i sama reč kaže, java folderi. Znači, ceo programski kod koji se tiče funkcionalnosti aplikacije smešten je unutar ovog foldera. Kao što je objašnjeno u prethodnom poglavlju o primeni i korišćenju aplikacije, aplikacija sadrži četiri ekrana sa odgovarajućim funkcionalnostima. U projektu sam ih nazvao **MainActivity**, **RegisterActivity, LoginActivity** i **ConnectActivity**, respektivno. **MainActivity** je početni ekran na kome korisnik bira da li želi da se registruje, što je neophodno ukoliko prvi put koristi aplikaciju, ili želi da se

uloguje ukoliko je ranije već registrovan. **RegisterActivity** je ekran gde korisnik unosi svoje podatke i pritiskom na dugme izvršava registraciju tj. aplikacija šalje podatke na bazu. **LoginActivity** je ekran gde korinsik unosi samo svoju e-mail adresu i šifru i dobija poruku sa servera da li je uspešno ulogovan tj. da li se uneti podaci poklapaju sa podacima koji postoje u bazi. **ConnectActivity** je ekran gde korisnik pritiskom na dugme traži, putem *bluetooth* tehnologije sa svog telefona, odgovarajući Arduino server u blizini i ukoliko ga pronađe, pojavljuje mu se dugme na koje može da se prijavi da je prisutan na događaju.

Svaka od ove četiri klase se sastoji od metoda koje opisuju njen životni ciklus. Ciklus počinje pozivanjem metode *OnCreate()* i završava se metodom *onDestroy()*. Mogu se implementirati i dodatne metode kao što su *onStart()*, *onStop()*, *onPause()*, *onRestart()* itd. S obzirom da je akcenat pri razvoju aplikacije stavljen na implementaciji sigurne komunikacije, koristićemo samo prve dve metode. Ovde treba naglasiti da se *onCreate()* metoda poziva pri pokretanju aplikacije, dok metodu *onDestroy()* treba eksplicitno pozvati iz same aplikacije [5].

4.3.1 MainActivity klasa

Objasnićemo malo detaljnije kako je realizovana **MainActivity.java** klasa aplikacije koju razvijamo, a na istom principu realizovane su i ostale tri klase. Prva stvar koju treba da odradimo je da povežemo elemente definisane u okviru korisničkog interfejsa sa promenljivim u našoj aplikaciji da bi mogli da manipulišemo unesenim podacima i vraćamo rezultat.

```
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
bReg = (Button) findViewById(R.id.btnRegister);
bLog = (Button) findViewById(R.id.btnLogin);
tvReg = (TextView) findViewById(R.id.tvReg);
tvLog = (TextView) findViewById(R.id.tvLog);
```

Postavljanje sadržaja stranice će takođe biti smešteno u *onCreate()* metodi, pošto se ona poziva odmah po pokretanju aplikacije. Dakle, prvo ćemo metodom *setContentView* postaviti sadržaj stranice.

Zatim se metodom *findViewByID* vezuje element definisan u korisničkom interfejsu sa promenljivom u okviru klase. Element se identifikuje pomoću id vrednosti, koju programer sam dodeljuje po sopstvenom nahođenju. Primera radi, za id dugmeta **register**, dodelio sam naziv *btnRegister* jer mi taj naziv pruža najbolju sliku o tome koja je funkcija datog dugmeta pa samim tim je mogućnost greške dosta smanjena. Isto mora da se uradi i za ostale elemente definisane u korisničkom interfejsu, nebitno da li su u pitanju dugmići (*Button*), polja za unos teksta (*EditText*), polja koja imaju funkciju labele (*TextView*) ili nešto od brojnih drugih mogućih opcija.

Pošto smo definisali elemente, između kojih i *Button*, treba da implementiramo ponašanje aplikacije. To ćemo uraditi tako da se pritiskom na *Button* (dugme) pokreću određeni procesi.

```
bReg.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View v) {
        Intent i = new Intent(getApplicationContext(), RegisterActivity.class);
        startActivity(i);
    }
});
```

Prvo ćemo pozvati interfejs iz Android klase *View*, čija je uloga da osluškuje da li je pokrenut odgovarajući događaj. U ovom slučaju to će biti *onClick*, odnosno ukoliko se pritisne odgovarajući osluškivani element, dolazi do izvršavanja naredbi unutar metode. Pri pritisku dugmeta register u okviru **MainActivity** klase jedino što se dešava jeste da se umesto dotadašnjeg ekrana prikazuje sledeći. To se ostvaruje korišćenjem klase **Intent** u okviru koje mora da se definiše u kojoj sledećoj klasi će se odvijati dalja radnja. U ovom slučaju to je **RegisterActivity** što se i dodeljuje kao drugi argument pri kreiranju instance klase *i = new Intent()*. Konačno *startActivity(i)* izvršava prethodno opisanu operaciju [5][6].

4.3.2RegisterActivity klasa

U okviru ove klase uvodi se novi element korisničkog interfejsa. U pitanju je polje za unos podataka koje obavlja korisnik. To se naziva *EditText*. Unete podatke dodeljujemo kao vrednost promenljivoj u okviru java klase korišćenjem *etName.getText().toString()* gde je *etName* ime promenljive u koju želimo da smestimo uneti podatak. Nakon što je korisnik uneo sve podatke koji su zahtevani, pritiskom na dugme **save** aplikacija pomoću PHP skripta šalje podatke na bazu podataka gde se podaci upisuju u okviru tebele Users.

Za realizaciju prethodno opisanog postupka neophodno je koristiti sistemsku klasu **AsyncTask**. Njena funkcija je da omogući obavljanje operacija u pozadini čiji će rezultati biti prikazani na glavnoj niti, a sve u cilju kako bi se glavna nit rasteretila. Klasa **AsyncTask** u ovoj aplikaciji sadrži tri metode, *onPreExecute*, *doInBackground* i *onPostExecute*.

```
@Override
protected void onPreExecute() {
super.onPreExecute();
dialog.setMessage("Sending data...");
dialog.show();
}
```

U okviru *onPreExecute* definiše se šta će se desiti neposredno pre izvršavanja glavne operacije koja se obavlja u unutar *doInBackgrund* metode, a koja u našoj aplikaciji predstavlja slanje korisnikovih podataka u bazu podataka u cilju uspešne registracije. Kao što se vidi iz priloženog koda korisniku se prikaže privremeni prozor sa natpisom da je u toku slanje podataka.

```
@Override
protected Boolean doInBackground(String... params) {
for(String url1 : params){

try {
            ArrayList<NameValuePair> pairs = new ArrayList<>();
            pairs.add(new BasicNameValuePair("txtName",etName.getText().toString()));
            pairs.add(new BasicNameValuePair("txtEmail",etEmail.getText().toString()));
            pairs.add(new BasicNameValuePair("txtPass",etPassConf.getText().toString()));
            HttpClient client = new DefaultHttpClient();
            HttpPost post = new HttpPost(url1);
            post.setEntity(new UrlEncodedFormEntity(pairs));
            HttpResponse response = client.execute(post);
        }   catch (ClientProtocolException e) {
            e.printStackTrace();
```

```
return false;
        }
catch (IOException e) {
            e.printStackTrace();
return  false;
        }
    }
return true;
}
```

Kao što je već rečeno, glavni deo se obavlja unutar *doInBackground* metode. Prvo se u *ArrayList* dodaju promenljive čija vrednost odgovara vrednosti koje korisnik unese i onda se ti podaci, nakon što aplikacija ostvari vezu sa serverom, šalju upravo na server. Ukoliko je sve prošlo kako treba vrednost koju vraća metoda biće true, a u supotnom false.

```
@Override
protected void onPostExecute(Boolean aBoolean) {
super.onPostExecute(aBoolean);
if(aBoolean == true){
        Toast.makeText(RegisterActivity.this,"You are successfully
registered",Toast.LENGTH_LONG).show();
        Intent i = new Intent(getApplicationContext(), LoginActivity.class);
        startActivity(i);
    }
else {
        Toast.makeText(RegisterActivity.this,"Error",Toast.LENGTH_LONG).show();
    }
 }
```

U okviru *onPostExecute* odlučuje se šta će se desiti nakon što je glavna operacija izvršena. Ukoliko je povratna vrednost metode *doInBackground* true, što znači ukoliko je unos podataka u bazu uspešno izvršen, korisniku se na ekranu prikazuje poruka da je uspešno registrovan i prelazi se na sledeći ekran pomoću klase **Intent**, kao što sam opisao u poglavlju o **MainActivity** klasi. Ukoliko je povratna vrednost metode *doInBackground* false, tj. ukoliko je unos podataka u bazu neuspešno izvršen korisniku će biti prikazana poruka o grešci i ostaće na istom ekranu da pokuša ponovo proces registracije

```
btnSave.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View v) {
        InsertData insert = new InsertData();
        insert.execute("http://45.63.104.97/registerApp.php");
    }
});
```

Poslednji korak u programiranju **RegisterActivity** klase jeste pozivanje *AsyncTask* klase u okviru glavne niti.

```
btnSave.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View v) {
        InsertData insert = new InsertData();
        insert.execute("http://45.63.104.97/registerApp.php");
    }
});
```

To se obavlja onog trenutka kada korisnik pritisne dugme **save**. Adresa napisana kao argument u okviru *insert.execute* sastoji se od IP adrese i PHP fajla. Napisana IP adresa u kodu

predstavlja IP adresu servera na kojem se nalazi baza podataka u koju se upisuju korisnički podaci, dok je dati PHP fajl aploudovan na taj server i napisan je tako da se pomoću njega ostvaruje veza između aplikacije i servera [5][6].

4.3.3LoginActivity klasa

U okviru ove klase korisnik je dužan da unese evoju e-mail adresu i šifru i pritiskom na dugme **login** aplikacija putem PHP skripta proverava da li je dati korisnik već registrovan u bazi podataka. Ukoliko jeste, korisniku će se prikazati poruka o uspešnom logovanju kao i sledeći ekran. Što se tiče samog koda, većina stvari je već korišćena i objašnjena u poglavlju o **RegisterActivity** klasi. Jedina novina se nalazi unutar *doInBackground* metode. S obzirom da se u ovom slučaju očekuje i povratna informacija sa servera, na kod iz pređašnje klase potrebno je dodati još jedan deo.

```
   InputStream is1 = response.getEntity().getContent()
        BufferedReader reader = null;
reader = new BufferedReader(new InputStreamReader(is1, "UTF-8"), 8);
        String line = null;
while ((line=reader.readLine())!=null) {
text += line + "\n";
         }
         is1.close();
return text;
```

Pojavljuju se dve nove klase, **InputStream** i **BufferedReader**. Da bi se pročitali podaci koje nam šalje server neophodno je koristiti klasu **InputStream**, a da bi se u okviru aplikacije pročitao sadržaj input strima neophodno je koristiti **BufferedReader** klasu. Na kraju, metoda **doInBackground** vraća promenljivu pod nazivom text, tipa string kao rezultat. U cilju boljeg razumevanja prikazaću deo koda PHP skripte neophodan za raumevanje izvršavanja *doInBackground* i *onPostExecute* metode.

```
<?php
        $con = mysql_connect(database_name,'root','password') or die('cannot connect to db');
        mysql_select_db('master1') or die('cannot select db');
        if(isset($_POST['email']) && isset($_POST['pass'])) {
        $email = $_POST['email'];
        $pass = $_POST['pass'];
        $query = "SELECT * FROM Users WHERE user_email='$email' AND
user_password='$pass'";
        $res=mysql_query($query);
        }
   if(mysql_num_rows($res) > 0)
   echo "You are successfully logged in";
   else
   echo "Incorrect email or password";
        mysql_close();
?>
```

Znači u PHP skripti izvršava se upit nad bazom podataka gde se proverava da li postoji korisnik sa upisanom e-mail adresom i datom šifrom. Ukoliko postoji, izbacuje se poruka da je logovanje uspešno obavljeno, što je predstavljeno stringom "You are successfully logged in", u suprotnom ispisuje se da je došlo do greške i da su šifra ili e-mail adresa pogrešni.

Vraćamo se na našu promenljivu pod nazivom text, koju metoda *doInBackground* vraća kao rezultat, i na osnovu prikazanog PHP skripta zaključujemo da ona može da ima dve vrednosti, "You are successfully logged in" za slučaj kada je logovanje uspešno obavljeno i "Incorrect email or password" kada to nije slučaj. Nakon ovoga, kod unutar *onPostExecute* metode postaje jasan.

```
@Override
protected void onPostExecute(String s) {
super.onPostExecute(s);
    Toast.makeText(LoginActivity.this, s, Toast.LENGTH_SHORT).show();
if (s.contains("You are successfully logged in")){
        Intent i = new Intent(LoginActivity.this,ConnectActivity.class);
        i.putExtra("user_email",etEmail.getText().toString());
        startActivity(i);
    }
}
```

Ukoliko je logovanje uspešno obavljeno, ispisuje se odgovarajuća poruka i pomoću **Intent** klase prelazi se na sledeći ekran.

Još jedna bitna stvar u ovom delu koda jeste sledeća linija:
i.putExtra("user_email",etEmail.getText().toString());
Ona nam omogućava da vrednost promenljive u koju je upisan korisnikov e-mail prenesemo u sledeću klasu, što nam je, videćemo u sledećem poglavlju, neophodno za uspešan rad aplikacije [5][6].

4.3.4ConnectActivity klasa

Četvrta i ujedno poslednja klasa u ovoj aplikaciji jeste **ConnectActivity** klasa. U okviru nje obavlja se glavni deo aplikacije, a to je prijavljivanje korinisnika na određeni događaj. Klikom na dugme **find** server, pali se *bluetooth* opcija na telefonu i kreće pretraga za drugim uređajima u blizini sa kojima bi mogla da se ostvari *bluetooth* veza.

```
mBluetoothAdapter = BluetoothAdapter.getDefaultAdapter();
if(mBluetoothAdapter == null) {
    Toast.makeText(ConnectActivity.this,"No bluetooth adapter
available",Toast.LENGTH_SHORT);
}

if(!mBluetoothAdapter.isEnabled()) {
    Intent enableBluetooth = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
    startActivityForResult(enableBluetooth, 0);
}
```

Kao što vidimo, koristi se klasa **BluetoothAdapter**, koja u stvari predstavlja radio hardver uređaja na kome se pokreće *bluetooth* opcija i mora se pozvati metoda ove klase pod nazivom *getDefaultAdapter()*. Ukoliko je prethodna stavka urađena kako treba, preko klase **Intent** se automatski korisniku pojavljuje mali prozor u kome ga pita da potvrdi da li želi da *bluetooth* opcija

bude uključena na njegovom uređaju. To se postiže servisnom metodom *action_request_enable* koja je u okviru **BluetoothAdapter** klase.

```
Set<BluetoothDevice> pairedDevices = mBluetoothAdapter.getBondedDevices();
if(pairedDevices.size() >0) {
for(BluetoothDevice device : pairedDevices) {
if(device.getName().equals("ARD_SPP")) {
mmDevice = device;
sendButton.setVisibility(View.VISIBLE);
            Toast.makeText(ConnectActivity.this, "Arduino Server Found",
Toast.LENGTH_SHORT);
openButton.setVisibility(View.GONE);
break;
        }
else
Toast.makeText(ConnectActivity.this, "Server not found",Toast.LENGTH_SHORT).show();
```

Sledeći korak jeste sagledavanje svih uređaja u okolini sa kojima je moguće ostvari *bluetooth* vezu. Ključni deo je linija u kojoj se proverava da li ime nekog od uređaja odgovara imenu Arduino servera koji se nalazi u prostoriji i čije je ime ranije definisano. Ime Arduino servera definiše onaj čiji je Arduino i u trenutnoj verziji aplikacije nije moguće menjati to ime, a da se aplikacija ne rekompajlira. Kada je dati uslov ispunjen, što znači uređaj je prepoznao odgovarajući Arduino server, korisniku na ekranu postaje vidljivo i drugo dugme na kome piše **sign up**.

Klikom na to dugme korisnik obavlja krajnji cilj aplikacije, a to je prijavljivanje da je prisutan na datom događaju. Ponovo se koristi klasa **AsyncTask** i na isti način kao u klasi **RegisterActivity**, podaci se šalju na bazu podataka. Ovoga puta je druga tabela u pitanju, u odnosu na onu gde su podaci kada se korisnik registrovao, i ona sadrži polja koja beleže korisnikovo ime, e-mail adresu i tačno vreme kada se prijavio.

S obzirom da na ovom ekranu korisnik nema nijedno polje u kome bi mogao da upiše neki od svojih podataka, klasa pomoću sledećeg koda povlači e-mail adresu koju je korisnik upisao kada se ulogovao.

```
Bundle extras = getIntent().getExtras();
if (extras != null) {
user_email = extras.getString("user_email");
}
```

Na ovaj način u mogućnosti smo da na server pošaljemo korisnikovu e-mail adresu iz date klase iako u okviru nje nije zahtevano unošenje tog podatka od strane korisnika. Zatim, pomoću PHP skripta vrši se upit nad tabelom Users u kojoj su smešteni podaci o registrovanim korisnicima. Pomoću tog upita iz te tabele sena osnovu jedinstvene e-mail adrese izvlači ime korisnika i sve to smešta u drugu tabelu gde se nalaze podaci o korisnicima koji su se prijavili na dati događaj [5][6].

5. ZAKLJUČAK

U ovom radu prikazao sam rad aplikacije koja bi uz određene modifikacije i unapređenja mogla da nađe svoje mesto među brojnim popularnim aplikacijama. S obzirom da se stalno teži ka tome da mobilni telefoni obavljaju što više radnji za koje je do skora bilo nezamislivo da će moći da se odrade na taj način, nije uopšte neobično što se došlo na ideju izrade ovakve aplikacije.

Aplikacija zamenjuje staromodni način prijavljivanja prisustva na primer, na predavanju, putem potpisivanja na papiru. Time se dosta smanjuje mogućnost za lažno prijavljivanje. Dok je na papiru moguće da jedna osoba praktično potpiše neograničen broj ljudi, ovde bi za tako nešto bilo potrebno da osoba poseduje isto toliko telefona koliko ljudi želi lažno da prijavi i pri tom da se na svakom registruje drugačije.

Olakšava se posao i onome ko drži predavanje, jer će sa baze podataka moći vrlo jednostavno da pročita ko je bio prisutan i kada na predavanju.

Naravno, aplikacija ima i svoje nedostatke. U ovom slučaju moguće je prijaviti prisustvo ako je u prostoriji samo jedan konkretan Arduino server. Moguće je dodati u aplikaciju i spisak drugih Arduino servera, ali ni to nije idealan slučaj, jer u krajnjoj liniji neko uvek može, makar slučajno, da promeni ime servera na osnovu kojeg ga aplikacija prepoznaje.

Prvobitno je bila ideja da se ostvari komunikacija na relaciji android telefon – Arduino server, gde uopšte ne bi bilo bitno koje je ime tog Arduino servera. Međutim, da bi se ovo realizovalo neophodno je dobro znanje i snalaženje sa programiranjem u Arduinu, tako da to ostavljam za poduhvat u skorijoj ili daljoj budućnosti. Sa tim izmenama ova aplikacija bi povećala svoju šansu za komercijalnu primenu.

LITERATURA

[1] https://en.wikipedia.org/wiki/Android_software_development

[2] PHP zvanična dokumentacija (https://secure.php.net/docs.php)

[3] MySQL zvanična dokumentacija (http://dev.mysql.com/doc/)

[4] Arduino zvanični tutorijali (https://www.arduino.cc/en/Tutorial/HomePage/)

[5] http://www.developer.android.com

[6] http://www.stackoverflow.com

A. KOD PROGRAMA NA KLIJENT STRANI

A.1. Fajl MainActivity.java

```
import android.content.Intent;
import android.support.v7.app.ActionBarActivity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;


public class MainActivity extends ActionBarActivity {
    public Button bReg,bLog;
    public TextView tvReg,tvLog;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        bReg = (Button) findViewById(R.id.btnRegister);
        bLog = (Button) findViewById(R.id.btnLogin);
        tvReg = (TextView) findViewById(R.id.tvReg);
        tvLog = (TextView) findViewById(R.id.tvLog);

        bLog.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent i = new Intent(getApplicationContext(), LoginActivity.class);
                startActivity(i);
            }
        });

        bReg.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent i = new Intent(getApplicationContext(), RegisterActivity.class);
                startActivity(i);
            }
        });
    }

}
```

A.2. Fajl LoginActivity.java

```
import android.app.ProgressDialog;
import android.content.Intent;
import android.os.AsyncTask;
import android.support.v7.app.ActionBarActivity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
```

```java
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;

import org.apache.http.HttpResponse;
import org.apache.http.NameValuePair;
import org.apache.http.client.ClientProtocolException;
import org.apache.http.client.HttpClient;
import org.apache.http.client.entity.UrlEncodedFormEntity;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.impl.client.DefaultHttpClient;
import org.apache.http.message.BasicNameValuePair;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.UnsupportedEncodingException;
import java.util.ArrayList;
import java.util.concurrent.ExecutionException;


public class LoginActivity extends ActionBarActivity {
    TextView tvEmail,tvPass,tv;
    EditText etEmail,etPass;
    Button btnLogin;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_login);
        tvEmail = (TextView) findViewById(R.id.tvEmail);
        etEmail = (EditText) findViewById(R.id.etEmail);
        tvPass = (TextView) findViewById(R.id.tvPass);
        etPass = (EditText) findViewById(R.id.etPass);
        btnLogin = (Button) findViewById(R.id.btnLogin);

        btnLogin.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                AttemptLogin login = new AttemptLogin();
                    login.execute("http://45.63.104.97/loginApp.php");
            }
        });
    }

    private class AttemptLogin extends AsyncTask<String, Void, String>{
        String text="";
        ProgressDialog dialog = new ProgressDialog(LoginActivity.this);
        @Override
        protected void onPreExecute() {
            super.onPreExecute();
            dialog.setMessage("Attempting to log in...");
            dialog.show();
        }

        @Override
        protected String doInBackground(String... params) {
            InputStream is1 = null;
            for(String url1:params) {
                try {
                    ArrayList<NameValuePair> pairs = new ArrayList<>();
                    pairs.add(new BasicNameValuePair("email",
etEmail.getText().toString()));
```

```
                    pairs.add(new BasicNameValuePair("pass",
etPass.getText().toString()));
                    HttpClient client = new DefaultHttpClient();
                    HttpPost post = new HttpPost(url1);
                    post.setEntity(new UrlEncodedFormEntity(pairs));
                    HttpResponse response = client.execute(post);
                    is1 = response.getEntity().getContent();
                } catch (ClientProtocolException e) {
                    e.printStackTrace();
                } catch (UnsupportedEncodingException e) {
                    e.printStackTrace();

Toast.makeText(LoginActivity.this,e.toString(),Toast.LENGTH_SHORT).show();
                } catch (IOException e) {
                    e.printStackTrace();

Toast.makeText(LoginActivity.this,e.toString(),Toast.LENGTH_SHORT).show();
                }
                BufferedReader reader = null;
                try {
                    reader = new BufferedReader(new InputStreamReader(is1, "UTF-8"), 8);
                } catch (UnsupportedEncodingException e) {
                    e.printStackTrace();

Toast.makeText(LoginActivity.this,e.toString(),Toast.LENGTH_SHORT).show();
                }
                String line = null;
                try {
                    while ((line=reader.readLine())!=null) {
                        text += line + "\n";
                    }
                } catch (IOException e) {
                    e.printStackTrace();

Toast.makeText(LoginActivity.this,e.toString(),Toast.LENGTH_SHORT).show();
                }
                try {
                    is1.close();
                } catch (IOException e) {
                    e.printStackTrace();

Toast.makeText(LoginActivity.this,e.toString(),Toast.LENGTH_SHORT).show();
                }
            }
                return text;
        }

        @Override
        protected void onPostExecute(String s) {
            super.onPostExecute(s);
            Toast.makeText(LoginActivity.this, s, Toast.LENGTH_SHORT).show();
            if (s.contains("You are successfully logged in")){
                Intent i = new Intent(LoginActivity.this,ConnectActivity.class);
                i.putExtra("user_email",etEmail.getText().toString());
                startActivity(i);
            }
            dialog.dismiss();
        }
    }
}
```

A.3. Fajl RegisterActivity.java

```
import android.app.ProgressDialog;
import android.content.ContentValues;
import android.content.Intent;
import android.os.AsyncTask;
import android.os.StrictMode;
import android.support.v7.app.ActionBarActivity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;

import org.apache.http.HttpEntity;
import org.apache.http.HttpResponse;
import org.apache.http.NameValuePair;
import org.apache.http.client.ClientProtocolException;
import org.apache.http.client.HttpClient;
import org.apache.http.client.entity.UrlEncodedFormEntity;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.impl.client.DefaultHttpClient;
import org.apache.http.message.BasicNameValuePair;

import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.UnsupportedEncodingException;
import java.util.ArrayList;
import java.util.List;


public class RegisterActivity extends ActionBarActivity {
    TextView tvName,tvEmail,tvPass,tvPassConf;
    EditText etName,etEmail,etPass,etPassConf;
    Button btnSave;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_register);
        tvName = (TextView) findViewById(R.id.tvName);
        etName = (EditText) findViewById(R.id.etName);
        tvEmail = (TextView) findViewById(R.id.tvEmail);
        etEmail = (EditText) findViewById(R.id.etEmail);
        tvPass = (TextView) findViewById(R.id.tvPass);
        etPass = (EditText) findViewById(R.id.etPass);
        tvPassConf = (TextView) findViewById(R.id.tvPassConf);
        etPassConf = (EditText) findViewById(R.id.etPassConf);
        btnSave = (Button) findViewById(R.id.btnSave);

        btnSave.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                InsertData insert = new InsertData();
                insert.execute("http://45.63.104.97/registerApp.php");
            }
        });
    }

    private class InsertData extends AsyncTask<String, Void, Boolean>{
        ProgressDialog dialog = new ProgressDialog(RegisterActivity.this);
```

```
        @Override
        protected void onPreExecute() {
            super.onPreExecute();
            dialog.setMessage("Sending data...");
            dialog.show();
        }

        @Override
        protected Boolean doInBackground(String... params) {
            for(String url1 : params){

                try {
                    ArrayList<NameValuePair> pairs = new ArrayList<>();
                    pairs.add(new
BasicNameValuePair("txtName",etName.getText().toString()));
                    pairs.add(new
BasicNameValuePair("txtEmail",etEmail.getText().toString()));
                    pairs.add(new
BasicNameValuePair("txtPass",etPassConf.getText().toString()));
                    HttpClient client = new DefaultHttpClient();
                    HttpPost post = new HttpPost(url1);
                    post.setEntity(new UrlEncodedFormEntity(pairs));
                    HttpResponse response = client.execute(post);
                }   catch (ClientProtocolException e) {
                    e.printStackTrace();
                    return false;
                }
                catch (IOException e) {
                    e.printStackTrace();
                    return  false;
                }
            }
            return true;
        }

        @Override
        protected void onPostExecute(Boolean aBoolean) {
            super.onPostExecute(aBoolean);
            if(aBoolean == true){
                Toast.makeText(RegisterActivity.this,"You are successfully
registered",Toast.LENGTH_LONG).show();
                Intent i = new Intent(getApplicationContext(), LoginActivity.class);
                startActivity(i);
            }
            else {
                Toast.makeText(RegisterActivity.this,"Error",Toast.LENGTH_LONG).show();
            }
            dialog.dismiss();
        }
    }
}
```

A.4. Fajl ConnectActivity.java

```
import android.app.Activity;
import android.app.ProgressDialog;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.bluetooth.BluetoothSocket;
import android.content.Intent;
import android.os.AsyncTask;
import android.os.Bundle;
import android.os.Handler;
import android.support.v7.app.ActionBarActivity;
```

```java
import android.view.View;
import android.widget.TextView;
import android.widget.EditText;
import android.widget.Button;
import android.widget.Toast;

import org.apache.http.HttpResponse;
import org.apache.http.NameValuePair;
import org.apache.http.client.ClientProtocolException;
import org.apache.http.client.HttpClient;
import org.apache.http.client.entity.UrlEncodedFormEntity;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.impl.client.DefaultHttpClient;
import org.apache.http.message.BasicNameValuePair;

import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.util.ArrayList;
import java.util.Set;
import java.util.UUID;

public class ConnectActivity extends ActionBarActivity {
    BluetoothAdapter mBluetoothAdapter;
    Button openButton;
    Button sendButton;
    BluetoothDevice mmDevice;
    String user_email;
    Boolean b = false;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_connect);

        openButton = (Button)findViewById(R.id.open);
        sendButton = (Button)findViewById(R.id.send);
        sendButton.setVisibility(View.GONE);

        Bundle extras = getIntent().getExtras();
        if (extras != null) {
            user_email = extras.getString("user_email");
        }

        mBluetoothAdapter = BluetoothAdapter.getDefaultAdapter();
        if(mBluetoothAdapter == null) {
            Toast.makeText(ConnectActivity.this,"No bluetooth adapter
available",Toast.LENGTH_SHORT);
        }

        if(!mBluetoothAdapter.isEnabled()) {
            Intent enableBluetooth = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
            startActivityForResult(enableBluetooth, 0);
        }

        openButton.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {

                Set<BluetoothDevice> pairedDevices =
mBluetoothAdapter.getBondedDevices();
                if(pairedDevices.size() > 0) {
                    for(BluetoothDevice device : pairedDevices) {
                        if(device.getName().equals("ARD_SPP")) {
                            mmDevice = device;
                            sendButton.setVisibility(View.VISIBLE);
```

```
                            Toast.makeText(ConnectActivity.this, "Arduino server found,
please sign up", Toast.LENGTH_SHORT);
                            //openButton.setVisibility(View.GONE);
                            b = true;
                            break;
                        }
                    }
                }
                else
                    Toast.makeText(ConnectActivity.this, "Server not
found",Toast.LENGTH_SHORT).show();

                if (b==true)
                        sendButton.setVisibility(View.VISIBLE);
                else
                    sendButton.setVisibility(View.GONE);
            }
        });

        sendButton.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                InsertData insert = new InsertData();
                insert.execute("http://45.63.104.97/arduinoCheckInApp.php");
            }
        });

    }

    private class InsertData extends AsyncTask<String, Void, Boolean> {
        ProgressDialog dialog = new ProgressDialog(ConnectActivity.this);

        @Override
        protected void onPreExecute() {
            super.onPreExecute();
            dialog.setMessage("Signing up...");
            dialog.show();
        }

        @Override
        protected Boolean doInBackground(String... params) {
            for(String url1 : params){

                try {
                    ArrayList<NameValuePair> pairs = new ArrayList<>();
                    pairs.add(new BasicNameValuePair("user_email",user_email ));
                    pairs.add(new BasicNameValuePair("server_name", "ARD_SPP"));
                    HttpClient client = new DefaultHttpClient();
                    HttpPost post = new HttpPost(url1);
                    post.setEntity(new UrlEncodedFormEntity(pairs));
                    HttpResponse response = client.execute(post);
                }   catch (ClientProtocolException e) {
                    e.printStackTrace();
                    return false;
                }
                catch (IOException e) {
                    e.printStackTrace();
                    return  false;
                }
            }
            return true;
        }

        @Override
        protected void onPostExecute(Boolean aBoolean) {
            super.onPostExecute(aBoolean);
```

```
            if(aBoolean == true){
                Toast.makeText(ConnectActivity.this,"You are successfully signed
up",Toast.LENGTH_LONG).show();
            }
            else {
                Toast.makeText(ConnectActivity.this,"Error",Toast.LENGTH_LONG).show();
            }
            dialog.dismiss();
        }
    }


}
```

B. KOD PROGRAMA NA SERVER STRANI

B.1. Fajl registerApp.php

```
<?PHP
      $con = mysql_connect('45.63.104.97','master1','djordjevic89') or
die('cannot connect to db');
      mysql_select_db('master1') or die('cannot select db');
       if(isset($_POST['txtName']) && isset($_POST['txtEmail']) &&
isset($_POST['txtPass'])){
       $txtName = $_POST['txtName'];
     $txtEmail = $_POST['txtEmail'];
       $txtPass = $_POST['txtPass'];
     mysql_query("insert into Users(user_name,user_email,user_password)
values('$txtName','$txtEmail','$txtPass')");
       }
?>
```

B.2. Fajl loginApp.php

```
<?php
      $con = mysql_connect('45.63.104.97','master1','djordjevic89') or
die('cannot connect to db');
      mysql_select_db('master1') or die('cannot select db');
      if(isset($_POST['email']) && isset($_POST['pass'])) {
      $email = $_POST['email'];
      $pass = $_POST['pass'];
      $query = "SELECT * FROM Users WHERE user_email='$email' AND
user_password='$pass'";
      $res=mysql_query($query);
      }
    if(mysql_num_rows($res) > 0)
    echo "You are successfully logged in";
    else
    echo "Incorrect email or password";
      mysql_close();
?>
```

B.3. Fajl arduinoCheckInApp.php

```
<?PHP
      $con = mysql_connect('45.63.104.97','master1','djordjevic89') or
die('cannot connect to db');
     mysql_select_db('master1') or die('cannot select db');
       if(isset($_POST['user_email']) && isset($_POST['server_name'])){
       $user_email = $_POST['user_email'];
     $server_name = $_POST['server_name'];
```

```
        $query = "select user_name from Users where user_email = '$user_email'";
        $result = mysql_query($query);
        $row = mysql_fetch_assoc($result);

        $user_name = $row['user_name'];

     mysql_query("insert into
Presence(user_name,user_email,server_name,time_presence)
values('$user_name','$user_email','$server_name','".date('Y-m-d H:i:s')."')");
        }
?>
```