UNIVERSITÀ DEGLI STUDI DI ROMA
TOR VERGATA
FACOLTÀ DI INGEGNERIA

CORSO DI LAUREA MAGISTRALE IN INGEGNERIA INFORMATICA

A.A. 2015/2016

Relazione sul progetto di Mobile Programming

GESTIONE DI EVENTI BASATI SU CODE

DOCENTE

Massimo Regoli

STUDENTE

Gabriele Belli
0229621

Indice

Capitolo 1

Introduzione

1.1 Panoramica

GestisciFacile è una piattaforma per la gestione di eventi basati su turno con lo scopo di migliorare la qualità dell'attesa per la ricezione di un servizio.
Utilizzando l'app *GestisciFacile* sul proprio smartphone sarà possibile staccare il numeretto per la fila direttamente da cellulare, conoscere in tempo reale il numero di persone davanti a sé e avere una stima del tempo di attesa.
A tutto questo va poi aggiunto un design molto intuitivo che segue le linee guida di un *elder application* (pulsanti e scritte molto grandi e applicazione simile ad un normale telecomando per TV), così da poter essere utilizzata da persone anziane e/o con problemi di vista.
Proprio a queste tipologie di persone che il *Voice Helper*, funzionalità dell'applicazione, viene incontro, dando loro informazioni leggibili e/o udibili sul servizio che stanno usufruendo.
Il design è inoltre completamente modificabile: dimensione testo, tema, colore testo potranno essere cambiate andando incontro così alle esigenze di particolari persone con un'alterata visione dei colori.

1.2 Obiettivi

Gli obiettivi che il sistema *GestisciFacile* vuole raggiungere sono molto ambiziosi e sopratutto non sono fini all'applicazione stessa.
La piattaforma vuole infatti:

- Rendere semplice meccanismi quotidiani come la coda per l'attesa di un servizio, necessaria a svolgere molti adempimenti pubblici o della nostra vita privata.
- Ridurre notevolmente l'impatto ambientale sostituendo il tradizionale metodo di erogazione dei numeretti.
- Poter essere utilizzato da tutti grazie alla sua semplicità e facilità di utilizzo. La semplicità non è sinonimo di superficialità in quanto qualcosa di semplice, se utile, può rendere piacevole l'utilizzo del sistema.
- Avvicinare al mobile e più in generale alle tecnologie digitali la fetta di pubblico ancora dubbiosa sui benefici che questa può portare anche a livello personale.
- Ottimizzare il servizio degli enti che decidono di usufruire di *GestisciFacile* fornendo loro un feedback del numero di persone che utilizzano quotidianamente il servizio stesso.

1.3 Contesti di utilizzo

L'idea di realizzazione dell'applicazione è nata in seguito ad un' esperienza personale: mi trovavo in un negozio dove il proprietario a causa della mancanza della carta nella macchinetta che distribuiva i numeretti e della grande mole di persone dovette scrivere e consegnare l'ordine di arrivo dei clienti su un foglio, provocando però un notevole disagio dovuto alla duplicazione dei numeri causati dall'errore umano ed al tempo perso.
Gestisci Facile è un sistema di ampio respiro, che può adeguarsi a molti contesti della realtà che ci circonda, così da poter entrare a far parte della nostra quotidianità, con vantaggi sia per chi l'utilizza sia per chi lo mette a disposizione come servizio. Rispetto ai concorrenti del mercato l'applicazione è semplice da utilizzare, intuitiva e non necessita di una figura per la gestione del servizio, chiunque potrà una volta letto il

manuale d'uso gestire questo servizio. Dal parrucchiere al fast-food passando per l'ufficio postale, l'applicazione potrà essere messa a disposizione da chiunque e dovunque.

La relazione è articolata come segue: il secondo capitolo introduce la tecnologia utilizzata; la descrizione dell'applicazione e come questa è stata realizzata occuperà tutto il terzo ed infine le conclusioni del lavoro svolto e gli sviluppi futuri sono riportati nel quarto capitolo, per finire verrà presentato un manuale d'uso. Il lavoro verrà concluso con la bibliografia.

Capitolo 2

Tecnologia utilizzata

Il QR–Code vuole rappresentare il punto di congiunzione tra il mondo reale (quello che tocchiamo con mano) ed il mondo virtuale rappresentato dalle strutture dati di un server. Ecco quindi che questa tecnologia può essere utilizzata per implementare la realtà aumentata per esempio associando un contenuto virtuale ad uno concreto, per abilitare alla geolocalizzazione dispositivi che ne sono sprovvisti (il QR–Code può rappresentare un luogo a partire dal quale erogare contenuti), per rendere interattivo ciò che non lo è (il QR–Code è associato ad una funzione).

Unire ciò che è virtuale e ciò che non lo è, ha come effetto il poter trasferire le caratteristiche proprie di un mondo nel suo duale: per cui un cartellone pubblicitario (elemento del mondo reale) può diventare clickabile (tipica azione del mondo virtuale).Allo stesso modo posso trasferire caratteristiche fisiche nel mondo virtuale.

2.1 QR-code

Un codice QR (in inglese QR Code, abbreviazione di Quick Response Code) è un codice a barre bidimensionale, ossia a matrice, composto da moduli neri disposti all'interno di uno schema di forma quadrata. Viene impiegato per memorizzare informazioni generalmente destinate a essere lette tramite un telefono cellulare o uno smartphone. In un solo crittogramma sono contenuti 7.089 caratteri numerici o 4.296 alfanumerici.
Il nome "QR" è l'abbreviazione dell'inglese "Quick Response" ("risposta rapida"), in virtù del fatto che il codice fu sviluppato per permettere una rapida decodifica del

suo contenuto. I codici QR possono contenere sia indirizzi internet, che testi, numeri di telefono, o sms. Sono leggibili da qualsiasi telefono cellulare e smartphone munito di un apposito programma di lettura (lettore di codici QR, o in inglese QR reader). Dato che Denso Wave ha reso pubblico l'uso della tecnologia QR con licenza libera, su internet è possibile trovare programmi gratuiti sia per la lettura (decodifica) che per la scrittura (codifica) dei codici QR.
Dalla fine degli anni 2000 i programmi di lettura dei codici QR sono spesso già installati nei telefonini dai relativi produttori. Esistono comunque molti siti web che offrono i lettori per cellulari, generalmente senza costi. Per leggere un codice QR è sufficiente inquadrarlo con la fotocamera del cellulare dopo aver aperto il lettore. Per quel che riguarda la scrittura, esistono diversi siti che consentono la libera produzione di codici QR.

2.1.1 Rilevazione e correzione d'errore

I codici QR possono memorizzare, come detto in precedenza, fino ad un massimo di 4.296 caratteri alfanumerici, 7.089 caratteri numerici. Nei codici QR è utilizzato il codice Reed-Solomon per la rilevazione e correzione d'errore: nel caso in cui il QR fosse in parte danneggiato, per esempio da macchie o graffi sul supporto cartaceo, l'applicazione Reed-Solomon permette di ricostruire i dati persi, ripristinando, durante la decodifica, fino al 30% delle informazioni codificate.

Capacità di memorizzazione dei dati:

- Solo numerico: max 7. 089 caratteri.
- Alfanumerico: max 4. 296 caratteri.
- Binario (8 bit): max 2. 953 byte.
- Kanji/Kana: max 1. 817 caratteri.

Capacità di correzione degli errori:

- Livello L: circa il 7 % delle parole in codice può essere ripristinato.
- Livello M: circa il 15 % può essere ripristinato.
- Livello Q: circa il 25 % può essere ripristinato.
- Livello H: circa il 30 % può essere ripristinato.

2.1.2 Esempi di utilizzo

Il QR–Code può essere applicato efficacemente nelle più disparate soluzioni dall'ambito consumer a quello enterprise.

- Disegnare il QR–Code sulle tovagliette di un bar per distribuire in qualsiasi momento il menù del giorno diminuendo l'abbattimento degli alberi.
- Utilizzare il Qr–Code nelle etichette dei prodotti, come nel caso della bottiglia di vino, per leggere la provenienza e le caratteristiche di questi
- Far viaggiare dati o documenti senza l'utilizzo di computer ma usufruendo semplicemente solo dello smartphone.
- Utilizzare il Qr–Code come biglietto da visita.

Capitolo 3

Scelte Implementative

In questo capitolo vengono descritte le scelte implementative che hanno contraddistinto il progetto e come queste sono state realizzate.

3.1 Client, Server e Service

La piattaforma *Gestisci Facile* basa il suo funzionamento su un sistema Client-Server:

- Il Server ha il compito di mostrare il numero attuale del cliente che deve essere servito e di interagire con i comandi inviati dai diversi client.
- Il Client invece deve inviare i messaggi al Server così da ricevere un numeretto o aggiornare quello del prossimo utente da servire.

Il Server è esposto al personale che vuole usufruire della piattaforma per questo motivo ha una grafica molto semplice ed intuitiva di facile lettura e confortevole alla vista. Alla base di questo vi è un Service.
I Service sono componenti Android messi a disposizione per lavori di durata lunga o addiritura indeterminata, sono classificabili in due tipologie, dipendentemente dal modo in cui vengono avviati. I Service Started sono da predilligere per operazioni con una loro finalità indipendente dallo stato delle altre applicazioni. Si potrebbero occupare di aggiornamenti dati in background, scaricamento di file o immagini, sincronizzazione remota verso server esterni, etc. Considerando che il service Started rimarrà in background a lungo la sua esistenza deve essere giustificata dalla finalità preposta. Per utilizzare un Service è necessario svolgere due operazioni:

- Creare una classe Java che estenda Service o un suo derivato.
- Registrare il service nel manifest con il nodo service.

Pertanto nel codice avremo una classe ServerService che estende Service:

```
public class ServerService extends Service {
public static AtomicBoolean running = new AtomicBoolean(false);
...
    @Override
    public void onCreate() {
        ...
        ServerOn = true;
    }


    @Override
    public int onStartCommand(Intent intent, int flags, int startId) {
       running.set(true);
       ...
    }


    @Override
    public IBinder onBind(Intent arg0) {
        return null;
    }

    @Override
    public void onDestroy() {

        ServerOn = false;
        running.set(false);
        ...
    }
}
```

Mentre nel AndroidManifest:

```
<service android:name=".ServerService" />
```

All'interno del Service troviamo il Server multithread, il quale, per come è stato creato e posizionato, anche in presenza di una chiusura anomala dell'activity, dovuta ad esempio al comportamento di un utente, continua a erogare il numeretto.
La variabile int che contiene il prossimo numero da distribuire è associata ad un lock poiché solo un thread alla volta può accedere a questa. Il Server una volta accettata una connessione, passa la socket ad un thread il quale legge il messaggio ricevuto ed esegue un'opportuna operazione. Se nel messaggio è contenuta la password del Server questo invierà attraverso un intent un messaggio di default alla ServerActivity la quale provvederà ad aggiornare il numero del prossimo cliente da servire. Se invece nel messaggio vi è contenuta una parola chiave, che di default è il nome dell'applicazione, allora il thread cercherà di accedere alla variabile protetta da lock e in seguito la modificherà per notificare ai prossimi thread che quel valore è cambiato.Una volta ottenuta il numero, il thread risponderà a colui che ha effettuato la richiesta inviando oltre al numero d'ordine la durata di validità di questo e un'informazione sul tempo medio per ciascuna persona.
Al termine di entrambe le operazioni la connessione viene chiusa.

```
    try {
        Socket clientSocket = myServerSocket.accept();
        ClientServiceThread cliThread =
        new ClientServiceThread(clientSocket);
        cliThread.start();
        }
        ...

        }

        ...


class ClientServiceThread extends Thread {

...

if (clientCommand.equals((prefs.getString(Util.key_ServerPassword,
    Util.default_password)))) {

    Intent i = new Intent(Util.BUS_ID);
```

```
    i.putExtra(Util.key_intent,Util.default_keyIntent);
    broadcaster.sendBroadcast(i);

     else {
     int numberToSend;
     synchronized (this) {
     numberToSend = numberClient;
     numberClient++;
                        }
     ...
     String send = ...
     dataOutputStream.writeUTF(send);
                                          }
...
```

Il comportamento delle due tipologie dei client è speculare a quello del Server.
Il thread del Worker dovrà semplicemente inviare la password al Server senza attendere una risposta da questo, mentre quello dell'utente dovrà eseguire un parser sul messaggio di risposta per adattare i dati ricevuti all'activity così da rendere comprensibili a chi si interfaccia con l'applicazione.
Nel Activity del Server è presente un altro thread che ha il compito di aggiornare l'interfaccia grafica con l'ora esatta.

```
            // @Override
            public void run() {
                while(!Thread.currentThread().isInterrupted()){
                    try {
                        doWork();
                        Thread.sleep(1000);
                            }
                     ...
                }
            }
        }


        public void doWork() {
            runOnUiThread(new Runnable() {
                public void run() {
                    try{
```

```
                        Calendar c = Calendar.getInstance();
                         SimpleDateFormat df = new
                             SimpleDateFormat(Util.format_date2);
                        String formattedDate =
                            df.format(c.getTime());
                        txtCurrentTime.setText(formattedDate);
                        ...
                    }
                });
            }
```

3.2 Utilizzo ZXing

Per costruire e interagire con il QR–code ho utilizzato le librerie ZXing.
E' stato pertanto necessario prima aggiungerle al progetto andando a modificare il build.grandle come segue:

```
    compile fileTree(dir: 'libs', include: ['*.jar'])
    compile 'com.android.support:appcompat-v7:23.1.1'
    compile 'com.journeyapps:zxing-android-embedded:3.2.0@aar'
    compile 'com.google.zxing:core:3.2.1'

}
```

Una volta aggiunte è stato possibile utilizzarle.L'activity del Server usufruisce di queste per disegnare il QR–code contenente l'informazioni sulla sua locazione e sulla sua porta.
Il metodo utilizzato è il seguente:

```
QRCodeWriter writer = new QRCodeWriter();
try {
BitMatrix bitMatrix = writer.encode(Util.app_name+"\n"+
Util.getIpAddress()+"\n"+
SP.getString(Util.key_ServerPort,Util.default_portS),
    BarcodeFormat.QR_CODE,200,512);
int width = bitMatrix.getWidth();
int height = bitMatrix.getHeight();
Bitmap bmp = Bitmap.createBitmap(width, height, Bitmap.Config.RGB_565);
```

```
for (int x = 0; x < width; x++) {
for (int y = 0; y < height; y++) {
bmp.setPixel(x, y, bitMatrix.get(x, y) ? Color.BLACK : Color.WHITE);
}
}
((ImageView) findViewById(R.id.QrCode)).setImageBitmap(bmp);

} catch (WriterException e) {
e.printStackTrace();
}
            }
```

Per quanto riguarda invece la lettura del Qr–code i passi essenziali sono l'utilizzo dell' IntentIntegrator per iniziare la scansione e Override del metodo OnActivityResult() che viene eseguita appena letto qualcosa dal codice.

```
public void onActivityResult(int requestCode, int resultCode, Intent
    intent) {
        IntentResult scanResult =
            IntentIntegrator.parseActivityResult(requestCode,
            resultCode, intent);
        if (scanResult != null) {

            if(scanResult.getContents()!=null) {
                String re = scanResult.getContents();
                 ....
                }
            }
        }


    }
```

3.3 Database

Nella piattaforma *Gestici Facile* è presente un database con il compito di memorizzare per ogni sessione il numero di biglietti staccati. È buona pratica creare una classe helper per semplificare le interazioni con il database ed introdurre così un livello di

astrazione che fornisca metodi intuitivi per la creazione e aggiornamento di questo.

```
    private static final String SESSIONS_TABLE_CREATE =
            "CREATE TABLE IF NOT EXISTS "
            + ProductsMetaData.SESSIONS_TABLE + " ("
            + ProductsMetaData.ID+ " integer primary key
                autoincrement, "
            + ProductsMetaData.DATATIME + " text not null, "
            + ProductsMetaData.VAL + " text not null);";
    .....

     private class DbHelper extends SQLiteOpenHelper {

        public DbHelper(Context context, String name,
            SQLiteDatabase.CursorFactory factory,int version) {
            super(context, name, factory, version);
        }

        @Override
        public void onCreate(SQLiteDatabase _db) {
            _db.execSQL(SESSIONS_TABLE_CREATE);
        }

        @Override
        public void onUpgrade(SQLiteDatabase _db, int oldVersion, int
            newVersion) {

        }

    }
```

Nella classe MyDb sono contenuti metodi per la scrittura di una sessione sul database e metodi per prelevare queste in base alla data o al numero di persone utilizzabili per un eventuale sviluppo futuro.

3.4 Preferences

La classe android.content.SharedPreferences permette agli sviluppatori di salvare dei settaggi applicativi in un file e condividerli nella applicazione stessa o tra tutte le

applicazioni. Il file dove vengono salvate queste applicazioni è presente nel path \data\data\"package". Android mette a disposizione un framework per gestire la persistenza dei dati e la presentazione dell'interfaccia per modificarli. Per memorizzare la password ho deciso invece prima di cifrarla così da aumentare la sicurezza; ho creato pertanto una classe che implementa quella su citata e utilizza i metodi "encrypt" e "decrypt" per criptare e decriptare il valore salvato o da salvare.

```
    protected String encrypt( String value ) {

        try {
            final byte[] bytes = value!=null ?
            value.getBytes(UTF8) : new byte[0];
            SecretKeyFactory keyFactory =
            SecretKeyFactory.getInstance(Util.key);
            SecretKey key = keyFactory.generateSecret(new
                PBEKeySpec(SEKRIT));
            Cipher pbeCipher = Cipher.getInstance(Util.key);
            pbeCipher.init(Cipher.ENCRYPT_MODE, key,
            new PBEParameterSpec(Settings.Secure.getString(
            context.getContentResolver(),
            Settings.System.ANDROID_ID).getBytes(UTF8), 20));
            return new String(Base64.encode(pbeCipher.doFinal(bytes),
                Base64.NO_WRAP),UTF8);

        } catch( Exception e ) {
            throw new RuntimeException(e);
        }

    }

    protected String decrypt(String value){
        try {
            final byte[] bytes = value!=null ?
            Base64.decode(value,Base64.DEFAULT) : new byte[0];
            SecretKeyFactory keyFactory =
                SecretKeyFactory.getInstance(Util.key);
            SecretKey key = keyFactory.generateSecret(new
                PBEKeySpec(SEKRIT));
            Cipher pbeCipher = Cipher.getInstance(Util.key);
            pbeCipher.init(Cipher.DECRYPT_MODE, key,
```

```
            new PBEParameterSpec(
            Settings.Secure.getString(context.getContentResolver(),
            Settings.System.ANDROID_ID).getBytes(UTF8), 20));
            return new String(pbeCipher.doFinal(bytes),UTF8);

        } catch( Exception e) {
            throw new RuntimeException(e);
        }
    }
```

A partire da API11 è consigliabile utilizzare i PreferenceFragment per costruire le schermate impostazioni;così infatti è quanto fatto nell'applicazione, il caricamento del layout viene effettuato esattamente effettuando Override del metodo OnCreate() ed utilizzando al suo interno il metodo addPreferencesFromResource().

```
        public void onCreate(final Bundle savedInstanceState)
        {
            super.onCreate(savedInstanceState);
            addPreferencesFromResource(R.xml.preferences_setting);
            ...
         }
```

dopodiché ho caricato il fragment all'interno della SettingActivity

```
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        ...
        getFragmentManager().beginTransaction().
        replace(android.R.id.content, new
            MyPreferenceFragment()).commit();
    }
```

Capitolo 4

Conclusioni

Gestisci Facile è un sistema che semplifica e velocizza operazioni quotidiane.
Peculiarità fondamentale della piattaforma è l'essere fruibile da tutti, grazie agli accorgimenti tecnologici in termini di *easy access*, senza distinzioni di età, predisposizione verso la tecnologia e, cosa molto importante a mio parere, può essere utilizzata anche da persone ipovedenti, poiché possiede una *user-experience* consona al loro stato fornendo la funzionalità del Voice Helper e un'interfaccia completamente modificabile.
Ho fatto della facilità di utilizzo la base per la mia app, avendo pensato che i maggiori utilizzatori di questa potrebbero essere persone che, anche per l'età, nonostante posseggano uno smartphone non sono consapevoli dei miglioramenti che questo può apportargli.
In conclusione sono molto soddisfatto del lavoro svolto, soprattutto in virtù dei progressi e dei miglioramenti apportati all'applicazione durante tutta la sua fase di analisi e sviluppo. La release attuale ha già tutti i requisiti per essere distribuibile per una prova sul "campo"; nonostante ciò sono molte gli *improvement* che sono sopraggiunti durante il lavoro e che a causa del poco tempo non ho potuto realizzare.

4.1 Sviluppi futuri

I possibili miglioramenti che potrebbero essere apportati alla piattaforma *Gestisci Facile* sono:

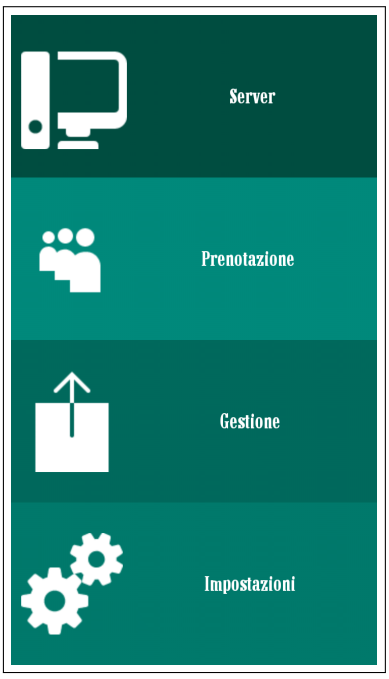
- Possibilità di visualizzare sotto forma di grafico la serie storica delle persone che hanno usufruito in ogni sessione del servizio.

- Gestire più file contemporaneamente distribuendo equamente il carico.
- Utilizzare le Google Cloud Messaging per notificare al cliente l'avvicinamento del suo turno.
- Possibilità qualora qualcuno rinunci al numero di poter scalare le posizioni di ogni utente in attesa.
- Inserire un form per la raccolta di feedback sul servizio.
- Inserire l'applicazione *Gestisci facile* all'interno di un contesto più ampio fondendola con la piattaforma *Prenota Facile*.
- Possibilità di inserimento di banner pubblicitari.
- Criptare la comunicazione tra client e server per evitare eventuali attacchi man-in-the-middle.
- Inserire nel client una serie storica dei servizi usufruiti.

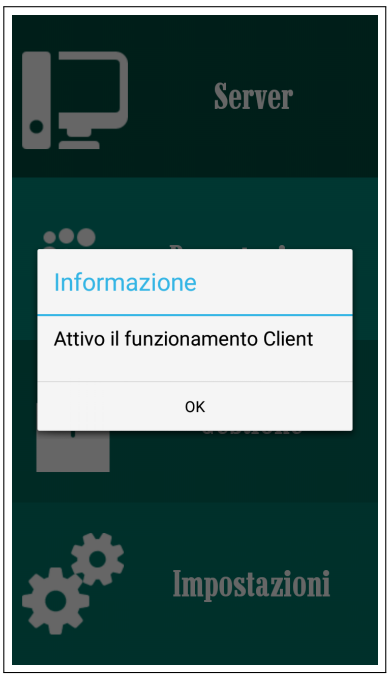
Capitolo 5

Manuale d'uso

L'applicazione appena installata si presenta all'utente con le impostazioni grafiche di default (dimensione del testo,colore e tema) e nella modalità di funzionamento client.



(a) Schermata iniziale



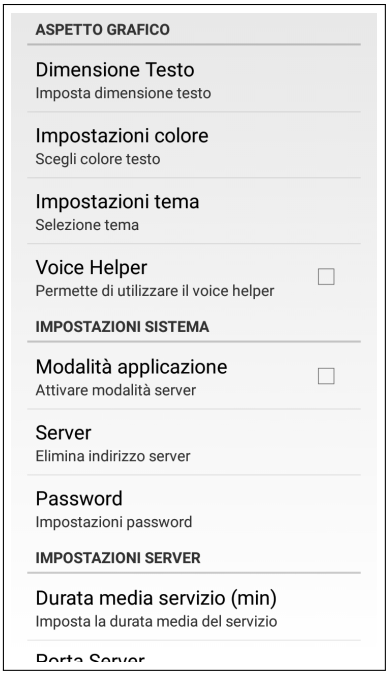
(b) Funzionamento client

Pertanto sarà possibile accedere solamente alle funzionalità raggiungibili tramite i pulsanti "Prenotazione", "Gestione" e "Impostazioni", qualora l'utente provasse però ad accedere alla funzionalità Server verrà mostrata un' AlertDialog che notificherà la modalità di utilizzo in corso (b) che può essere comunque cambiata dalle impostazioni.

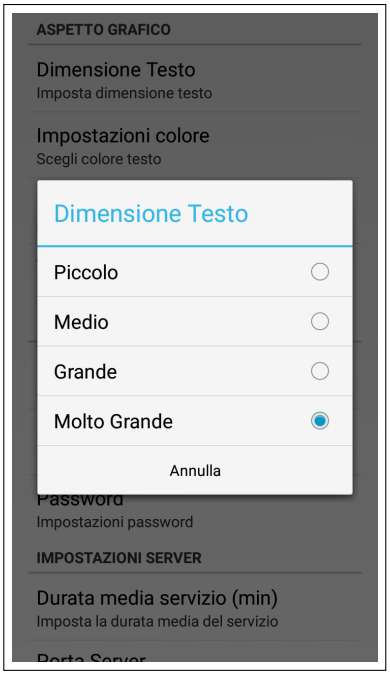
Tramite l'ultimo pulsante in basso è possibile accedere proprio a queste (c) le quali sono suddivise, in base al contesto, in 4 categorie:

- Aspetto grafico
- Impostazioni di sistema
- Impostazioni server
- Crediti

Con la prima categoria è possibile modificare, come è intuibile, la dimensione del testo (d), il colore di questo nella schermata iniziale, il tema dell'applicazione e l'attivazione o meno della funzionalità del Voice Helper. Quest'ultima funzionalità è quella tramite la quale premendo a lungo sui pulsanti è possibile (se connessi in internet) ascoltare a cosa si può accedere tramite loro.
Qualora invece, questa funzionalità non fosse attiva sarà possibile, tramite la stessa "gesture", prendere visione tramite una finestra di quanto detto sopra.



(c)



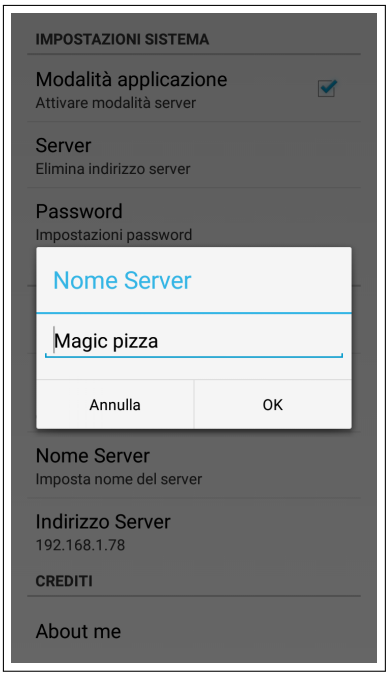
(d)

Apportando alcune modifiche alle impostazioni e tornando indietro nella schermata possiamo notare come questa sia cambiata (e)(f).

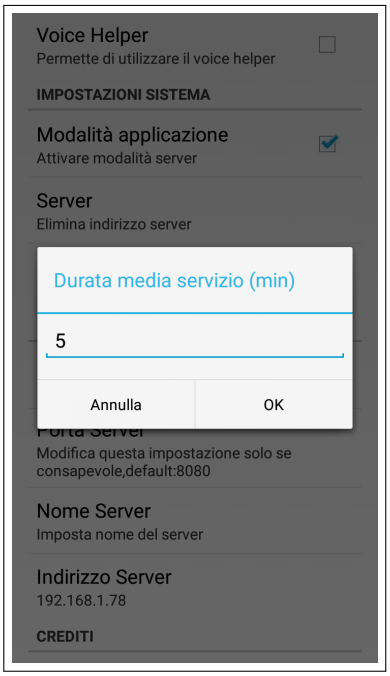
(e)

(f)

Con la seconda categoria invece è possibile modificare la modalità di funzionamento dell'applicazione. Queste modalità sono divise appunto in Server e Client, la prima è quella che permette di visualizzare il numero della persona che in questo momento è servita mentre la seconda permette di poter prendere il proprio numero così da poter mettersi in fila [tasto 2 (a)] o di aggiornare il numero della persona che dovrà essere servita [tasto 3 (a)].
La possibilità di aggiornare il numero della persona che dovrà essere servita, però, è possibile solo se si è in possesso della password del Server, la quale può essere impostata o modifica nell'apposito campo Password, dove è utilizzata come password di default la parola "123stella".
Come vedremo in seguito per poter connettersi ad un Server sarà prima necessario conoscere il suo indirizzo IP che possiamo ottenere scansionando semplicemente il QR-code che questo mette a disposizione dalla sua schermata che sarà memorizzato sul dispositivo fino a quando non decideremo di eliminarlo premendo sull'apposito pulsante "Server" di questa categoria (c).



(g)

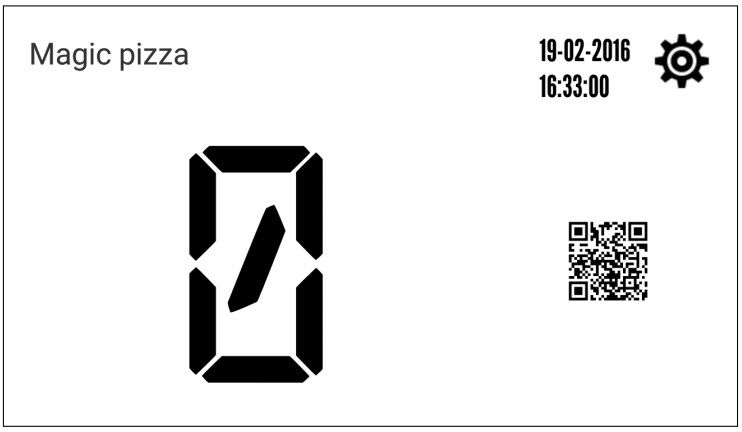


(h)

Tramite la categoria Impostazioni Server invece è possibile inserire il nome dell'esercizio (g) che utilizza il servizio e la durata media dell'erogazione di questo (h).
Premendo sul pulsante "About me" invece è possibile visualizzare le informazioni dello sviluppatore.

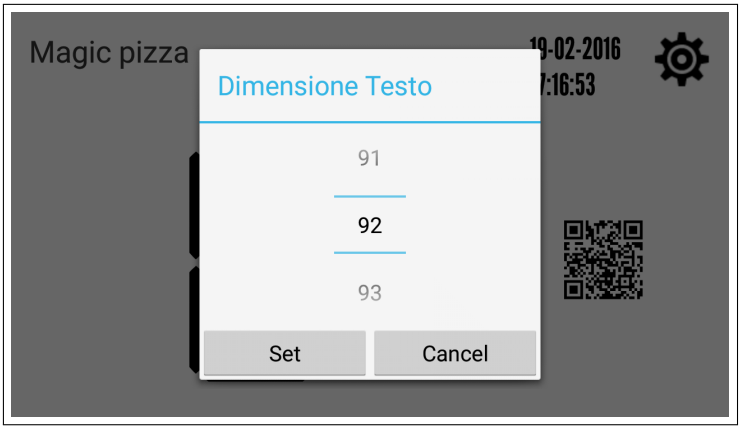
5.1 Modalità Server

Dopo una panoramica sulle impostazioni possiamo passare ad analizzare le vari parti della piattaforma con la consapevolezza di come è possibile accedervi e modificare le loro funzionalità.
Premendo il primo pulsante possiamo accedere alla schermata principale del Server; questa si presenterà come nella figura sottostante (i).



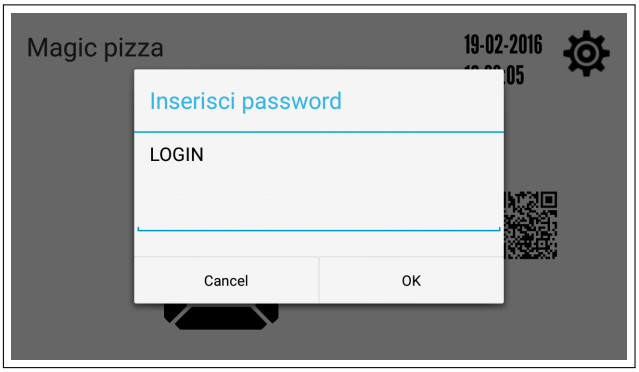
(i)

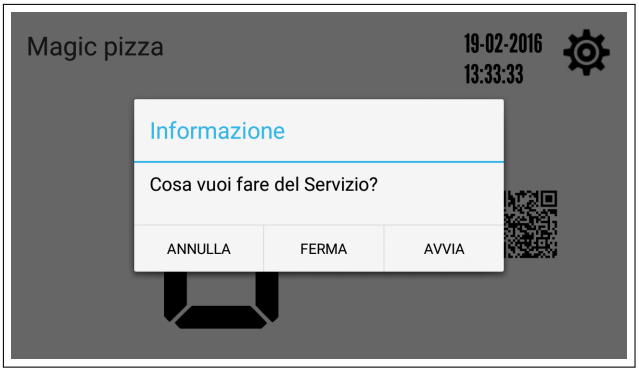
Analizzando l'immagine da sinistra a destra notiamo prima in alto il nome dell'esercizio che utilizza la piattaforma, aggiunto nelle impostazioni, mentre al di sotto il numero del prossimo cliente che dovrà essere servito, se il Server non è in funzione verrà visualizzato il numero 0; tenendo premuto questo è comunque possibile modificarne la dimensione (j).



(j)

In alto a destra è possibile invece visualizzare l'ora e il giorno corrente ed un pulsante tramite il quale dopo aver digitato la password è possibile avviare o fermare il Server.





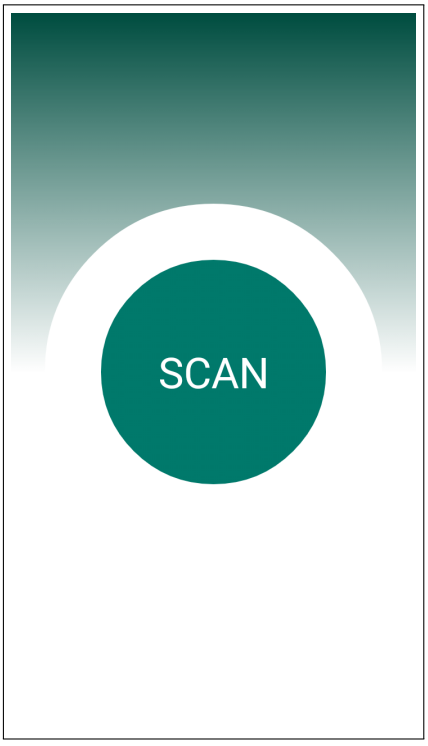
In basso a destra notiamo invece il QR–code che viene utilizzato sia dagli utenti per ottenere tramite scansione il proprio numero che dal personale dell'esercizio per aggiornare il numero del prossimo cliente da servire.

5.2 Modalità Client

La modalità client, di default e riattivabile come esposto in precedenza dalle impostazioni, si suddivide in modalità User e Worker la prima utilizzata dalle utente che vuole ricevere il numeretto, la seconda invece per modificare il numero da visualizzare dallo schermo del Server, ovvero quello del prossimo cliente che dovrà essere servito.

5.2.1 Modalità User

La procedura per ottenere il numeretto è molto semplice, basterà premere il pulsante "Prenotazione" per ritrovarsi la schermata della figura sottostante.



(o)

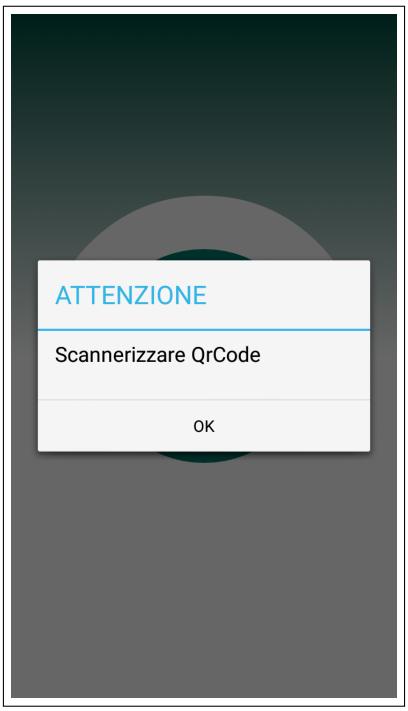
Da qui pigiando il pulsante "SCAN" si attiverà la fotocamera e avvicinando questa alla schermata del Server sull'apposito QR–code sarà possibile visualizzare il numero ottenuto, la data di validità del numero e il tempo medio di attesa per una persona (p).
In basso alla schermata è presente un pulsante il quale se premuto, dopo aver confermato la volontà, cancella la prenotazione appena eseguita e permette di eseguirne una nuova (q).

(p)

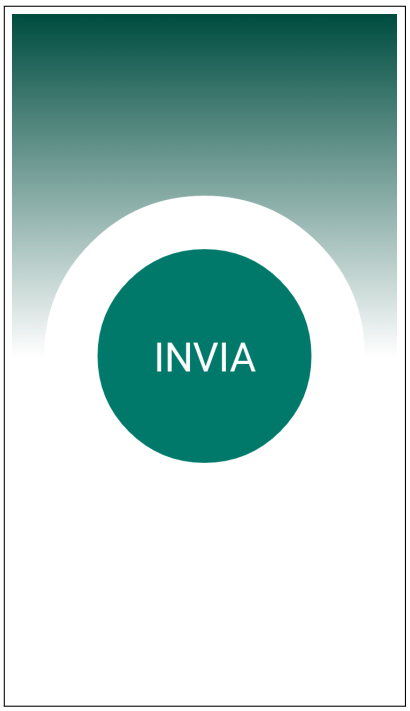
(q)

5.2.2 Modalità Worker

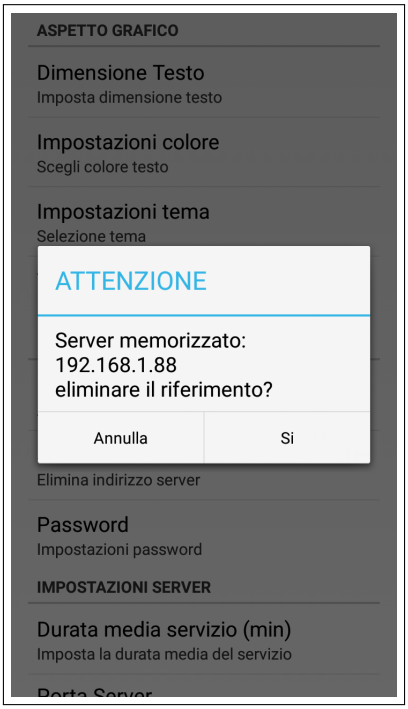
La procedura per ottenere i permessi per modificare il numeretto visualizzato nella schermata del Server richiede un passaggio in più rispetto a quanto visto in precedenza, ma resta comunque una procedura nel complesso molto semplice. Una volta recatisi in "Impostazioni" è modificato la password in modo che coincida con quella del Server, basterà premere il pulsante "Gestione" da qui ci troveremo di fronte ad una finestra che ci invita a scanerizzare il QR–code del Server (r)(s). Per semplificare il tutto la procedura di scansione, da effettuare per ottenere la locazione del Server, dovrà essere effettuata una sola volta per sessione, essendo questa salvata nel dispositivo. Qualora però volessimo eliminarla dovremmo recarci nelle impostazioni ed eliminare il riferimento, la quale procedura è stata descritta all'inizio di questo capitolo (t).


ATTENZIONE
Scannerizzare QrCode
OK

(r)


INVIA

(s)


ASPETTO GRAFICO
Dimensione Testo
Imposta dimensione testo
Impostazioni colore
Scegli colore testo
Impostazioni tema
Selezione tema
ATTENZIONE
Server memorizzato:
192.168.1.88
eliminare il riferimento?
Annulla
Si
Elimina indirizzo server
Password
Impostazioni password
IMPOSTAZIONI SERVER
Durata media servizio (min)
Imposta la durata media del servizio

(t)

Bibliografia

Cay Horstmann, Gary Cornell *"Core Java Volume I"* PEARSON Prentice-Hall.

Cay Horstmann, Gary Cornell *"Core Java Volume II"* PEARSON Prentice-Hall.

Massimo Carli *"Android Guida per lo sviluppatore"* APOGEO, 2010

"Android Programming" Edizioni Master, on line:
http://punto-informatico.it/PILibri/Dettaglio.aspx?id=238
ultimo accesso 18/02/2016

Rossi Pietro Alberto *"Android by Example"*, on line:
http://www.sprik.it/guida/Android4_2.pdf
ultimo accesso 18/02/2016

Tutorial Android, on line:
http://android.devapp.it/
ultimo accesso 18/02/2016

I Service, on line:
http://www.vogella.com/tutorials/AndroidServices/article.html7
ultimo accesso 18/02/2016

Cos'è un codice QR?, on line:
http://www.qrmode.it/it/
ultimo accesso 18/02/2015

Database e Android, on line:
http://www.html.it/articoli/la-gestione-dei-database-in-android-1/
ultimo accesso 18/02/2016

Official ZXing, on line:
https://github.com/zxing/zxing
ultimo accesso 18/02/2016

Socket, on line:
http://developer.android.com/reference/java/net/Socket.html
ultimo accesso 18/02/2016

Comunicazione Service e Activity
http://www.anddev.it/index.php?topic=8708.0
ultimo accesso 18/02/2016

Intent
http://developer.android.com/reference/android/content/Intent.html
ultimo accesso 18/02/2016

UI e thread
http://www.anddev.it/index.php?topic=29.0
ultimo accesso 18/02/2016

Obscured Shared Preferences
http://www.righthandedmonkey.com/2014/04/obscured-shared-preferences-for-android.html
ultimo accesso 18/02/2016

ZXing e Android Studio
http://stackoverflow.com/questions/27851512/how-to-integrate-zxing-library-to-android-studio-for-barcode-scanning
ultimo accesso 18/02/2016