

ALMA MATER STUDIORUM - UNIVERSITA' DI BOLOGNA  
CAMPUS DI CESENA  
SCUOLA DI SCIENZE

CORSO DI LAUREA IN SCIENZE E TECNOLOGIE INFORMATICHE

**Progettazione e sviluppo  
di un'applicazione smartphone per acquisti  
usando l'nfc**

Relazione finale in  
**Mobile Web Design**

Relatore  
**Dott. Mirko Ravaioli**

Presentata da  
**Simone Gobbi**

Sessione III  
Anno Accademico 2013 - 2014



# Indice

---

<b>PRESENTAZIONE.....</b>	<b>4</b>
<b>INTRODUZIONE.....</b>	<b>5</b>
CHE COS'È "THE ANANKE"?	5
CHE COS'È L'NFC?	6
IL PROGETTO.....	9
<b>LA REALIZZAZIONE.....</b>	<b>11</b>
IL DATABASE .....	11
LE PAGINE PHP .....	12
<i>insert.php</i> .....	12
<i>select.php</i> .....	13
<i>update.php</i> .....	14
L'APP CLIENTI.....	15
<i>MainActivity.java</i> .....	17
<i>DettagliOfferta.java</i> .....	21
<i>AcquistoOfferta.java</i> .....	23
L'APP ESERCENTI .....	32
<i>MainActivity.java</i> .....	34
<b>POSSIBILI MIGLIORAMENTI E SVILUPPI FUTURI .....</b>	<b>41</b>
<b>CONCLUSIONI.....</b>	<b>43</b>

# Presentazione

---

Lo scopo della tesi in questione è di realizzare un prototipo di sistema che permetta agli utenti del social network “The Ananke” di acquistare determinate offerte sfruttando un chip NFC (*Near Field Communication*).

Il tag nfc è stato inserito in un braccialetto che dovrà essere associato a una persona e serve come autenticazione in fase di acquisto o di ritiro dell’offerta.

Per la realizzazione di questo progetto sono state sviluppate due applicazioni per Android (lato client), un piccolo database (lato server) e delle pagine in php per il dialogo tra i client e il server.

Nel primo capitolo saranno introdotti il social network “The Ananke”, la tecnologia nfc e si spiegherà meglio in cosa consiste il progetto.

Nel secondo capitolo sarà mostrato come sono stati sviluppati i vari elementi con i rispettivi linguaggi di programmazione.

Nel terzo capitolo saranno affrontati i possibili miglioramenti e gli sviluppi futuri del progetto.

Nel quarto e ultimo capitolo saranno illustrate le conclusioni.

# Introduzione

---

## *Che cos'è "The Ananke"?*

"The Ananke" è un social network aperto con l'obiettivo di superare gli eventuali preconcetti/abitudini delle persone derivanti da razza, religione, età o nazionalità focalizzandosi esclusivamente sugli interessi (e le community) dei singoli. Da ciò consegue lo slogan che lo contraddistingue "No borders, no age, no religion, no race".

Per questo motivo è nato "The Ananke" (Ananke nella mitologia greca è la personificazione del destino e del fato) che facilita l'interazione tra le persone sia nel mondo reale sia in quello virtuale, eliminando ogni barriera sociale.

I membri del social network utilizzano quale simbolo di riconoscimento un braccialetto che riporta un codice numerico (ID) stampato, identificando univocamente il suo possessore all'interno della community.

Il destino entra in gioco poiché lo stesso codice ID sarà stampato su due differenti bracciali. In seguito le coppie di bracciali con lo stesso ID saranno separate e rimescolate prima di essere destinate alla commercializzazione. Chi acquista un bracciale dispone quindi di un canale di comunicazione esclusivo e dedicato all'interno del social network con il proprio utente corrispettivo (con lo stesso codice ID), chiamato Ananke, con cui diversamente non avrebbe pensato di mettersi in contatto. In aggiunta, attraverso un algoritmo casuale, a ogni utente tutti i giorni è assegnato un "Ananke del giorno" (possessore di un bracciale con un altro ID), a tutti gli effetti simile all'Ananke corrispondente, ma attivo solo per una giornata.

Essendo ogni utente riconosciuto all'interno del social network esclusivamente con il codice ID del proprio bracciale (quindi anonimo per tutti tranne che per l'Ananke corrispondente e per l'Ananke del giorno per un periodo di tempo limitato) Ananke intende superare le affinità elettive e la ristretta cerchia di amici e conoscenti focalizzandosi esclusivamente sugli interessi e le community dei singoli rappresentate in sezioni ad hoc del social network.

## Che cos'è l'NFC?

NFC (*Near Field Communication*, “Comunicazione in prossimità”) è una tecnologia che fornisce connettività wireless bidirezionale a corto raggio (fino a un massimo di 10 cm).

Contrariamente ai più semplici dispositivi RFID (*Radio Frequency Identification* – Identificazione a Radio Frequenza), NFC permette una comunicazione bidirezionale: quando due apparecchi NFC (lo initiator e il target) vengono accostati entro un raggio di 4 cm, viene creata una rete peer-to-peer tra i due ed entrambi possono inviare e ricevere informazioni.

Lo NFC è realizzato direttamente tramite un chip integrato negli smartphone: il primo telefono ad averlo inserito fu il Nokia 6131 nel 2006, ma la sua diffusione è aumentata dopo il 2010 con il Samsung Nexus S (primo smartphone Android dotato di questa tecnologia) fino all'adozione nell'ottobre 2014 negli iPhone 6 di Apple.

Alcuni esempi di possibili applicazioni sono:

- Pagamento su dispositivi portatili NFC, attraverso computer o chioschi elettronici abilitati;
- Scaricamento da un PC o un chiosco elettronico su di un dispositivo portatile, della prenotazione o acquisto di una permanenza in albergo, ingressi a cinema, teatri, stadi, viaggio in treno o aereo, e accesso al servizio acquistato mediante il dispositivo stesso avvicinato o a contatto con il chiosco elettronico in albergo, al gate di ingresso o di partenza in stazioni/aeroporti;
- Trasferimento e visualizzazione di fotografie da una macchina fotografica o telefono cellulare NFC a un televisore, computer per la visione o la stampa;
- Trasferimento facilitato di file o messa in rete fra sistemi wireless;
- Uso della tecnologia NFC per i sistemi di bigliettazione elettronica;
- Operazioni rapide su smartphone come cambio di profilo, attivazione wi-fi, cambio di modalità della suoneria etc.

Gli utenti, vista la scarsa diffusione iniziale di questa tecnologia, si sono divertiti in sperimentazioni private di NFC, programmando ad hoc particolari chip chiamati TAG. Si tratta di piccoli adesivi programmati attraverso opportuni software per svolgere una particolare funzione.

Ad esempio, l'utente potrebbe programmare il tag per attivare automaticamente il GPS e il navigatore quando entra in auto, oppure per attivare l'accensione del wi-fi una volta rientrato a casa. Esistono diversi tipi di tag nfc, ognuna con dimensioni, capacità e scopi diversi.

La figura n.1 differenzia le categorie dei tag in base a ciascun tipo di utilizzo e ne descrive le proprietà.

## COME SCEGLIERE TAG NFC ADATTI ALLE PROPRIE ESIGENZE

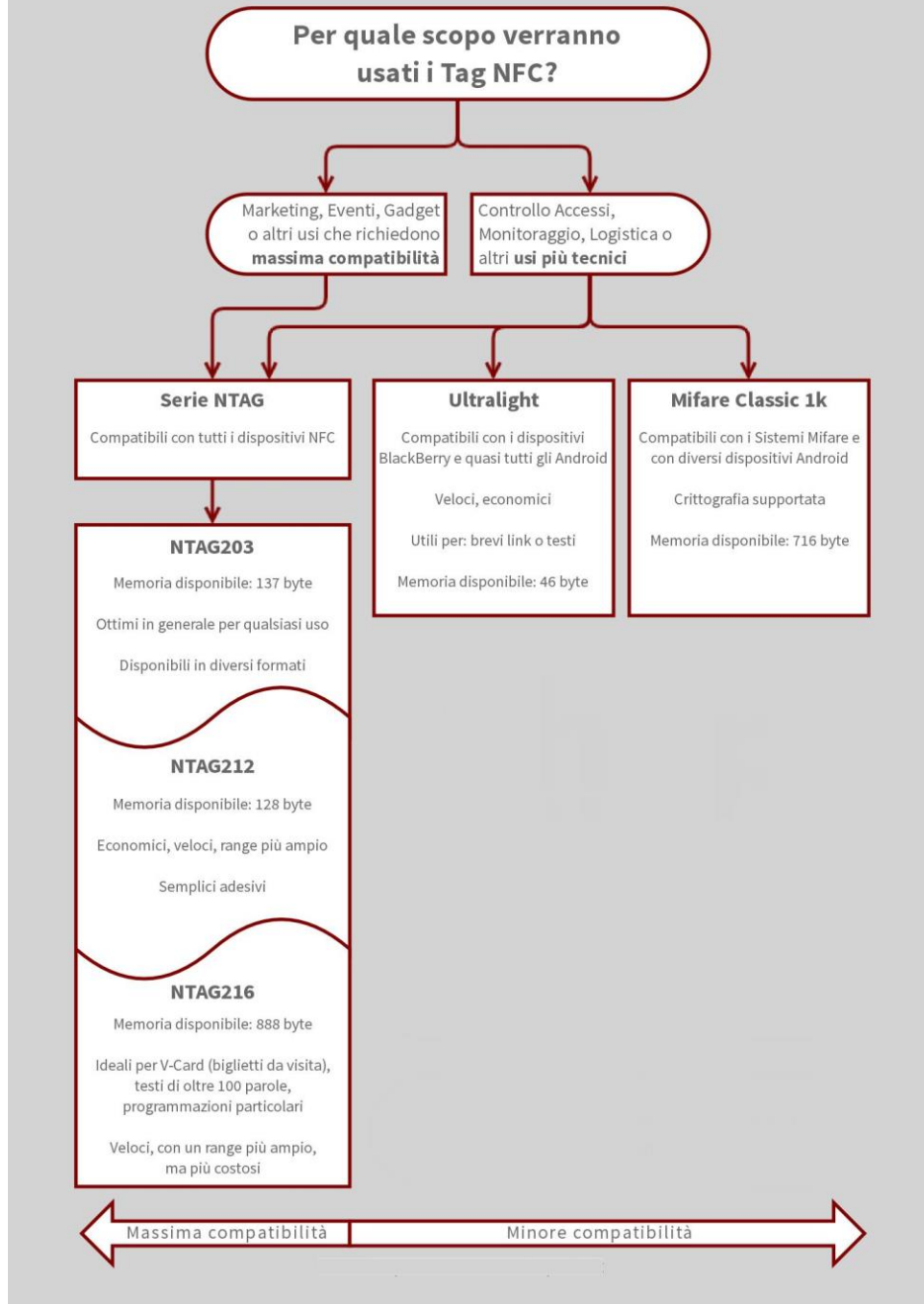


Figura 1: tipi di tag nfc



## Il progetto

L'idea alla base del progetto era costituita dalla ricerca dell'integrazione nei braccialetti di "The Ananke" di un chip nfc con lo scopo di facilitare e velocizzare l'acquisto di determinati beni e servizi all'interno di una catena di negozi.

Si pensi, ad esempio, alla possibilità di acquistare un biglietto per uno spettacolo al cinema attraverso lo smartphone e di evitare la coda alla biglietteria e all'ingresso in sala passando il proprio braccialetto su appositi sensori. Oppure alla possibilità di riservare determinati sconti nei locali semplicemente con un gesto. Un ultimo esempio può essere la possibilità di acquistare determinate offerte (come su Groupon) per ristoranti e, all'arrivo dall'esercente, avere già un tavolo riservato e la cucina con la lista degli ordini pronta. I contesti possono essere veramente svariati.

Questo servizio porterebbe vantaggi ad entrambe le parti: da un lato il cliente può avere un prezzo più vantaggioso ed evitare tempi di attesa, dall'altro l'esercente si procura un nuovo canale di vendita presso la community di "The Ananke", promosso anche dalla semplificazione dell'operazione.

Le due applicazioni per smartphone realizzate rispondono agli obiettivi descritti: con una l'iscritto al social network può scorrere le varie offerte e decidere di acquistarne una o più identificandosi con un gesto del polso, con la seconda l'esercente può controllare l'effettivo acquisto di una sua proposta e segnare che essa sia stata usufruita (e quindi non riutilizzabile).

Per fare ciò entrambe si collegano ad un server che mette a disposizione un database e delle pagine php che permettono la loro interazione.

Il progetto della tesi è un prototipo incentrato sullo sviluppo per ambiente Android, mentre sono stati toccati solo marginalmente la realizzazione del database, delle pagine php e di alcune funzionalità. Più nello specifico, l'app per la community permette:

- la visualizzazione della lista delle offerte;
- approfondire la conoscenza di un'offerta;

- l'acquisto di una determinata offerta.

Invece, l'app per gli esercenti permette:

- la visualizzazione delle offerte acquistate da un determinato possessore di un braccialetto;
- selezionare un offerta dalla lista e poter segnalarla come utilizzata.

Non esiste al momento alcuna interfaccia che permetta l'inserimento delle offerte da parte delle aziende convenzionate. Ne consegue, per ora, una possibilità di scelta limitata a sole tre offerte inserite nell'app clienti per lo sviluppo di questo progetto.

Il processo di acquisto è da considerarsi abbastanza sicuro:

1. il chip nfc contiene un codice univoco non modificabile che permette di riconoscere l'utente senza possibilità di errore;
2. essendo il codice non modificabile, non è possibile (o non è sicuramente facile) "clonare" un braccialetto per impossessarsi di beni o servizi acquistati da un altro utente;
3. anche qualora si riuscisse a clonare il braccialetto o a rubarlo, difficilmente si è a conoscenza degli acquisti effettuati dall'utente;
4. il perfezionamento dell'acquisto avverrà verosimilmente in un arco di tempo piuttosto breve, limitando di fatto il rischio di sottrazione del bene acquistato;
5. l'impiego di questa tecnologia è previsto, per ora, su transazioni di importo contenuto.

# La realizzazione

---

## Il database

Semplice ma necessaria è stata la realizzazione di un database per la memorizzazione dei dati.

Essendo uno degli aspetti marginali del progetto, si è sviluppato un database molto semplice e basilare per far sì che entrambe le applicazioni potessero leggere e scrivere i dati fondamentali delle ipotetiche transazioni.

Il database è stato creato in un server web con il vantaggio di essere disponibile facilmente tramite connessione internet dallo smartphone.

Le informazioni salvate sono inerenti all'offerta (nome, descrizione e prezzo), ai dati dell'utente (nome, cognome, carta di credito), ai dati del bracciale (codice univoco ed eventuali altri dati), alla data d'acquisto e alla fruizione o meno dell'offerta. Tutti questi campi sono stati inseriti in una stessa tabella con lo scopo di semplificare il prototipo iniziale.

Il codice SQL per la creazione di questa tabella è:

```
CREATE TABLE 'Acquisti' (  
    'Nome' varchar(32) NOT NULL COMMENT 'Nome acquirente',  
    'Cognome' varchar(32) NOT NULL COMMENT 'Cognome acquirente',  
    'Carta' varchar(32) NOT NULL COMMENT 'Numero carta di credito  
    dell'acquirente',  
    'Bracciale' varchar(10) NOT NULL COMMENT 'Codice bracciale',  
    'Usato' varchar(32) NOT NULL COMMENT 'Se è stato usato o meno',  
    'Data' varchar(20) NOT NULL COMMENT 'Data di acquisto',  
    'Offerta' varchar(40) NOT NULL,  
    'Prezzo' varchar(14) NOT NULL);
```

## Le pagine php

Come già detto, le informazioni necessarie per il funzionamento delle app sono nel database del server. Ma come andare ad interagire con esse?

Usando le pagine php è possibile eseguire direttamente da smartphone le operazioni di lettura e di scrittura sulla banca dati inviando al client una risposta.

Anche le pagine php sono state salvate sul server e, in particolare, si chiameranno insert.php, select.php e update.php.

Tutte le pagine php prevedono una connessione autenticata al database prima di eseguire le operazioni, ma i dati reali sono stati cambiati per motivi di privacy.

## *insert.php*

“insert.php” permette l’inserimento dell’acquisto e richiede come parametri i vari dati dell’utente, i dati dell’offerta acquistata, la data e le informazioni del tag nfc.

<?php

```
$my_user = 'root';
$my_password = 'root';
$my_db = 'Prova';

$connect = mysqli_connect("localhost", $my_user, $my_password, $my_db);

$nome = isset($_POST['sNome']) ? $_POST['sNome'] : "";
$cognome = isset($_POST['sCognome']) ? $_POST['sCognome'] : "";
$carta = isset($_POST['sCarta']) ? $_POST['sCarta'] : "";
$bracciale = isset($_POST['sBracciale']) ? $_POST['sBracciale'] : "";
$usato = isset($_POST['sUsato']) ? $_POST['sUsato'] : "";
$data = isset($_POST['sData']) ? $_POST['sData'] : "";
$offerta = isset($_POST['sOfferta']) ? $_POST['sOfferta'] : "";
$prezzo = isset($_POST['sPrezzo']) ? $_POST['sPrezzo'] : "";

$query = mysqli_query($connect,
    "INSERT INTO Acquisti (Nome, Cognome, Carta, Bracciale, Usato, Data,
    Offerta, Prezzo)
    VALUES ('$nome' , '$cognome', '$carta', '$bracciale', '$usato', '$data',
    '$offerta', '$prezzo' );
```

```
mysqli_close($connect);
?>
```

## *select.php*

“select.php” permette la lettura di tutti gli acquisti effettuati con un certo bracciale. Il codice del bracciale è il parametro.

```
<?php
    $my_server = 'localhost';
    $my_user = 'root';
    $my_password = 'root';
    $my_db = 'Prova';

    $connect = mysqli_connect($my_server, $my_user, $my_password, $my_db);

    if (mysqli_connect_errno($connect)) {
        echo "Failed to connect to MySQL: " . mysqli_connect_error();
    }

    $bracciale = $_GET['bracciale'];
    $res = mysqli_query($connect,
        "SELECT * FROM Acquisti
        WHERE Bracciale='$bracciale'");

    while($row = mysqli_fetch_array($res)) {
        $output[] = $row;
    }

    print(json_encode($output));

    mysqli_close($connect);
?>
```

## *update.php*

“update.php” permette la modifica di un determinato acquisto per segnalarne la sua fruizione. Riceve come parametri i dati dell’acquisto.

```
<?php
```

```
    $my_user = 'root';  
    $my_password = 'root';  
    $my_db = 'Prova';
```

```
    $connect = mysqli_connect("localhost", $my_user, $my_password, $my_db);
```

```
    $nome = isset($_POST['sNome']) ? $_POST['sNome'] : "";  
    $cognome = isset($_POST['sCognome']) ? $_POST['sCognome'] : "";  
    $carta = isset($_POST['sCarta']) ? $_POST['sCarta'] : "";  
    $bracciale = isset($_POST['sBracciale']) ? $_POST['sBracciale'] : "";  
    $usato = isset($_POST['sUsato']) ? $_POST['sUsato'] : "";  
    $data = isset($_POST['sData']) ? $_POST['sData'] : "";  
    $offerta = isset($_POST['sOfferta']) ? $_POST['sOfferta'] : "";  
    $prezzo = isset($_POST['sPrezzo']) ? $_POST['sPrezzo'] : "";
```

```
    $query = mysqli_query($connect,  
        "UPDATE Acquisti  
        SET Usato = '1'  
        WHERE Nome = '$nome'  
        AND Cognome = '$cognome'  
        AND Carta = '$carta'  
        AND Bracciale = '$bracciale'  
        AND Usato = '$usato'  
        AND Data = '$data'  
        AND Offerta = '$offerta'  
        AND Prezzo = '$prezzo' ") or die (mysqli_error());
```

```
    mysqli_close($connect);
```

```
?>
```

## L'app clienti

L'app clienti è l'applicazione che permette agli utenti della community di fare determinati acquisti ed è stata sviluppata attraverso Eclipse sfruttando i linguaggi Java e xml.

Molto importanti sono i permessi nel file AndroidManifest.xml per poter usare l'antenna nfc del telefono (per la lettura del codice del bracciale) e la connessione internet (per la comunicazione con il server). Inoltre va impostato il livello 10 come versione minima dell'SDK perché gli nfc sono supportati solo dalla versione Android 2.3.3

```
<uses-sdk
    android:minSdkVersion="14"
    android:targetSdkVersion="19" />

<uses-permission android:name="android.permission.NFC" />
    <uses-feature
        android:name="android.hardware.nfc"
        android:required="true" />
    <uses-permission android:name="android.permission.INTERNET"/>
    <uses-permission
        android:name="android.permission.ACCESS_NETWORK_STATE" />
```

E' stata creata una classe Offerta.java per rappresentare la singola offerta contenente il nome, la descrizione, il prezzo, un'immagine rappresentativa, il costruttore e i metodi di acquisizione dei dati.

```
public class Offerta implements Serializable{  
    private String nomeOfferta;  
    private String descrizioneOfferta;  
    private String prezzoOfferta;  
    private Bitmap immagineOfferta;  
  
    public Offerta(String nome, String descrizione, String prezzo, Bitmap immagine){  
        super();  
        nomeOfferta = nome;  
        descrizioneOfferta = descrizione;  
        prezzoOfferta = prezzo;  
        immagineOfferta = immagine;  
    }  
  
    public String getNome(){  
        return nomeOfferta;  
    }  
  
    public String getDescrizione(){  
        return descrizioneOfferta;  
    }  
  
    public String getPrezzo(){  
        return prezzoOfferta;  
    }  
  
    public Bitmap getImmagine(){  
        return immagineOfferta;  
    }  
}
```

Sono presenti tre activity che caricheranno ognuno il proprio layout (ovvero, la propria schermata):

1. MainActivity.java é la schermata iniziale. Permette la visualizzazione e la scelta di una offerta;
2. DettagliOfferta.java che espone l'offerta precedentemente selezionata focalizzandone le informazioni utili;
3. AcquistoOfferta.java che consente l'immissione dei propri dati, il riconoscimento del codice del proprio bracciale e permette l'acquisto tramite la pressione del tasto di conferma.



## MainActivity.java

La MainActivity si presenta con una semplice TextView per la scritta “Scegli la tua offerta!” e una ListView che andrà a contenere la lista di offerte selezionabili.



Figura 2: layout MainActivity.java (clienti)

La MainActivity conterrà le seguenti variabili globali:

```
ArrayList<Offerta> listaOfferte = new ArrayList<Offerta>(); //lista delle offerte che la  
listview visualizzerà
```

```
ListView listViewOfferte; //Variabile globale
```

```
Bitmap carne; //Variabile globale
```

```
Bitmap pesce; //Variabile globale
```

```
Bitmap bistecca; //Variabile globale
```

dove listaOfferte servirà per il passaggio dei dati e le varie bitmap serviranno a memorizzare le immagini delle offerte.

La ListView viene popolata da un adapter personalizzato e ottimizzato per poter visualizzare all'interno del singolo elemento tre componenti:

- Un'immagine raffigurativa a sinistra;
- Il nome dell'offerta al centro;
- Il prezzo in basso a destra.

```
public class CustomAdapterOptimize extends ArrayAdapter<Offerta> {  
  
    public CustomAdapterOptimize(Context context, int textViewResourceId,  
    List<Offerta> objects) {  
        super(context, textViewResourceId, objects);  
    }  
  
    @Override  
    public View getView(int position, View convertView, ViewGroup parent) {  
        return getViewOptimize(position, convertView, parent);  
    }  
  
    public View getViewOptimize(int position, View convertView, ViewGroup parent) {  
        ViewHolder viewHolder = null;  
        if (convertView == null) {  
            LayoutInflater inflater = (LayoutInflater) getContext()  
                .getSystemService(Context.LAYOUT_INFLATER_SERVICE);  
            convertView = inflater.inflate(R.layout.layout_offerta, null);  
            viewHolder = new ViewHolder();  
            viewHolder.name =  
(TextView)convertView.findViewById(R.id.textviewNomeOfferta);  
            viewHolder.cost =  
(TextView)convertView.findViewById(R.id.textViewPrezzoOfferta);  
            viewHolder.image = (ImageView)convertView.findViewById(R.id.imageOfferta);  
            convertView.setTag(viewHolder);  
        } else {  
            viewHolder = (ViewHolder) convertView.getTag();  
        }  
  
        Offerta c = getItem(position);  
        viewHolder.name.setText(c.getNome());  
        viewHolder.cost.setText(c.getPrezzo());  
        viewHolder.image.setImageBitmap(c.getImmagine());  
        return convertView;  
    }  
  
    private class ViewHolder {  
        public TextView name;  
        public TextView cost;  
        public ImageView image;  
    }  
}
```

Per poter mostrare all'interno di ogni singolo elemento della ListView più oggetti è stato utilizzato un layout apposito.

Il codice per popolare la ListView è il seguente:

```
Carne = BitmapFactory.decodeResource(getResources(),
R.drawable.grigliata_di_carne);
pesce = BitmapFactory.decodeResource(getResources(),
R.drawable.grigliata_di_pesce);
bistecca = BitmapFactory.decodeResource(getResources(),
R.drawable.bistecca_di_angus);

List list = new LinkedList();
list.add(new Offerta("Grigliata di carne", "Grigliata di carne per due persone", "€28,00",
carne));
list.add(new Offerta("Grigliata di pesce", "Grigliata di pesce per due persone", "€32,00",
pesce));
list.add(new Offerta("Bistecca di angus", "Bistecca di angus irlandese da 400 gr",
"€15,00", bistecca));

Offerta [] offerte={
    new Offerta("Grigliata di carne", "Grigliata di carne per due persone", "€28,00",
carne),
    new Offerta("Grigliata di pesce", "Grigliata di pesce per due persone",
"€32,00", pesce),
    new Offerta("Bistecca di angus", "Bistecca di angus irlandese da 400 gr",
"€15,00", bistecca)};

listaOfferte.add(offerte[0]);
listaOfferte.add(offerte[1]);
listaOfferte.add(offerte[2]);

listViewOfferte = (ListView) findViewById(R.id.listViewOfferte);
listViewOfferte.setAdapter(adapter);
listViewOfferte.setOnItemClickListener(new OnItemClickListener() {

    @Override
    public void onItemClick(AdapterView<?> parent, View view, int position, long
id) {
        dettagliOfferta(position);
    }
});
```

E' possibile notare come il tocco del singolo elemento di listViewOfferte chiami la funzione dettagliOfferta, passando anche la posizione dell'elemento della lista toccata.

La funzione `dettagliOfferta` serve a richiamare la seconda activity (`DettagliOfferta.java`) e, nel farlo, passerà nome, descrizione e prezzo dell'elemento scelto dalla lista.

```
private void dettagliOfferta(int index){  
    Intent intent = new Intent(MainActivity.this, DettagliOfferta.class);  
    String pkg = getPackageName();  
    Offerta elemento = listaOfferte.get(index);  
  
    String nome = elemento.getNome();  
    String descrizione = elemento.getDescrizione();  
    String prezzo = elemento.getPrezzo();  
  
    intent.putExtra(pkg+".Nome", nome);  
    intent.putExtra(pkg+".Descrizione", descrizione);  
    intent.putExtra(pkg+".Prezzo", prezzo);  
    startActivity(intent);  
}
```

## DettagliOfferta.java

Passiamo ora alla seconda activity. Anche DettagliOfferta.java ha un layout molto semplice e contiene il nome, l'immagine, la descrizione e il prezzo dell'offerta, più un tasto che servirà ad andare nella terza activity.

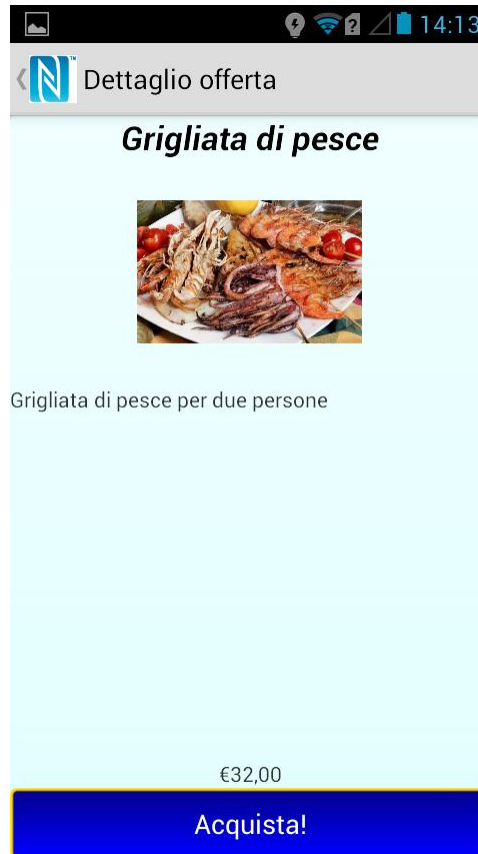


Figura 3: layout DettagliOfferta.java

In questa activity si fa uso di tre variabili globali: sono tutte e tre delle String che conterranno il nome, la descrizione e il prezzo dell'offerta.

Per ricavare le informazioni passate dall'activity precedente si ottiene l'intent (gli intent sono dei messaggi che il sistema manda a un'applicazione quando si aspetta che questa faccia qualcosa) da cui si possono estrarre i nostri dati. Il passo successivo è quello di valorizzare i nostri elementi del layout con i dati precedentemente memorizzati.

```
Intent intent = getIntent();  
String pkg = getPackageName();
```

```
nome = intent.getStringExtra(pkg+".Nome");  
descrizione = intent.getStringExtra(pkg+".Descrizione");
```

```
prezzo = intent.getStringExtra(pkg+".Prezzo");
```

```
TextView nomeOffertaDettaglio = (TextView)  
findViewById(R.id.textViewNomeOffertaDettaglio);  
TextView descrizioneOffertaDettaglio = (TextView)  
findViewById(R.id.textViewDescrizioneOffertaDettaglio);  
TextView prezzoOffertaDettaglio = (TextView)  
findViewById(R.id.textViewPrezzoDettaglio);  
nomeOffertaDettaglio.setText(nome);  
descrizioneOffertaDettaglio.setText(descrizione);  
prezzoOffertaDettaglio.setText(prezzo);
```

Infine, al tasto per l'acquisto è stata associata una funzione che permette l'avvio della terza activity passando ancora le informazioni necessarie dell'offerta.

```
Button buttonAcquista = (Button) findViewById(R.id.buttonAcquista);  
buttonAcquista.setOnClickListener(new View.OnClickListener() {  
    @Override  
    public void onClick(View v) {  
        Intent intent = new Intent(DettagliOfferta.this, AcquistoOfferta.class);  
        String pkg = getPackageName();  
        intent.putExtra(pkg+".Nome", nome);  
        intent.putExtra(pkg+".Prezzo", prezzo);  
        startActivity(intent);  
    }  
});
```

## AcquistoOfferta.java

La terza e ultima activity, chiamata AcquistoOfferta.java, si presenta con diverse TextView riportanti le informazioni dell'offerta selezionata e le etichette riguardanti i campi richiesti per l'acquisto con annesso suggerimento nelle EditText. Esattamente con lo stesso metodo con cui sono state prese le informazioni passate dalla prima activity alla seconda, vengono ricavati i dati per valorizzare le varie TextView.

In fondo al layout troviamo una TextView inizialmente non visibile che mostrerà se l'acquisto è stato effettuato e il tasto per la conferma dell'acquisto. Questi ultimi due elementi non faranno parte della ScrollView (che contiene gli altri oggetti e che consente di "scrollare" gli elementi nella schermata per visualizzare ciò che non sarebbe visibile negli schermi più piccoli).

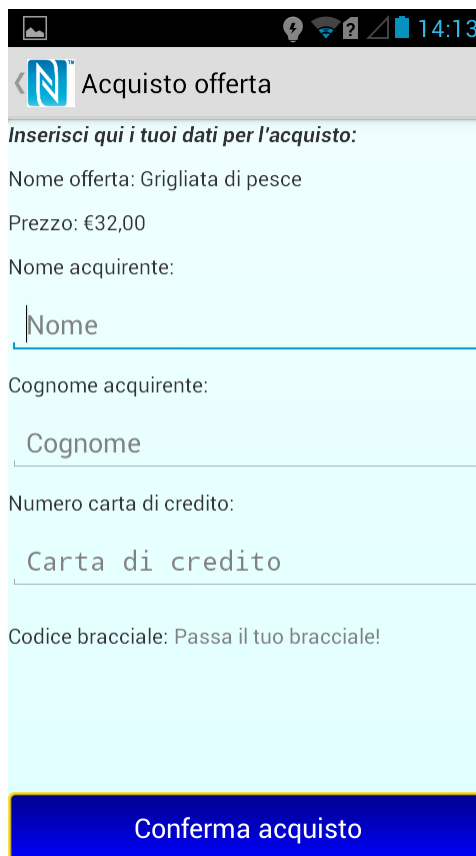


Figura 4: layout AcquistoOfferta.java

E' molto importante la TextView dov'è scritto "Passa il tuo bracciale!": questa, infatti, verrà valorizzata con il codice del bracciale a seguito della lettura da parte

dell'antenna dello smartphone.

Una delle prime operazioni una volta creata l'activity sarà controllare lo stato dell'nfc sul dispositivo per sapere se è presente o meno l'antenna e, subito dopo, se essa è attiva o meno.

```
private NfcAdapter mNfcAdapter; //questa variabile è globale
```

```
mNfcAdapter = NfcAdapter.getDefaultAdapter(this);  
if (mNfcAdapter == null) {  
    Toast.makeText(this, "Il dispositivo non supporta gli NFC.",  
    Toast.LENGTH_LONG).show();  
    finish();  
    return;  
}  
if (!mNfcAdapter.isEnabled()) {  
    codiceBracciale.setText("NFC é disabilitato");  
} else {  
}
```

E' ancora presto, però, per poter leggere i tag nfc e abbiamo bisogno di più funzioni per riuscirci.

```
@Override  
protected void onResume() {  
    super.onResume();  
    setupForegroundDispatch(this, mNfcAdapter);  
}
```

```
@Override  
protected void onPause() {  
    stopForegroundDispatch(this, mNfcAdapter);  
    super.onPause();  
}
```

```
@Override  
protected void onNewIntent(Intent intent) {  
    handleIntent(intent);  
}
```

```
public static void setupForegroundDispatch(final Activity activity, NfcAdapter  
adapter) {  
    final Intent intent = new Intent(activity.getApplicationContext(),  
    activity.getClass());  
    intent.setFlags(Intent.FLAG_ACTIVITY_SINGLE_TOP);  
    final PendingIntent pendingIntent =  
    PendingIntent.getActivity(activity.getApplicationContext(), 0, intent, 0);  
    IntentFilter[] filters = new IntentFilter[1];  
    String[][] techList = new String[][]{};  
    filters[0] = new IntentFilter();
```



```

filters[0].addAction(NfcAdapter.ACTION_NDEF_DISCOVERED);
filters[0].addCategory(Intent.CATEGORY_DEFAULT);
try {
    filters[0].addDataType(MIME_TEXT_PLAIN);
} catch (MalformedMimeTypeException e) {
    throw new RuntimeException("Check your mime type.");
}
adapter.enableForegroundDispatch(activity, pendingIntent, filters, techList);
}

public static void stopForegroundDispatch(final Activity activity, NfcAdapter adapter) {
    adapter.disableForegroundDispatch(activity);
}

```

Praticamente sono stati inseriti nel metodo `onResume` e `onPause` l'avvio e lo stop del `ForegroundDispatch` e il metodo `onNewIntent` viene lanciato quando si riavvicinerà il tag al dispositivo.

Nell'`handleIntent` inseriremo la lettura dei dati dal tag.

```

private void handleIntent(Intent intent) {
    String action = intent.getAction();
    if (NfcAdapter.ACTION_NDEF_DISCOVERED.equals(action)) {
        String type = intent.getType();
        if (MIME_TEXT_PLAIN.equals(type)) {
            Tag tag = intent.getParcelableExtra(NfcAdapter.EXTRA_TAG);
            UID = bin2hex(tag.getId());
            Log.i("Leggi QUA", "UID: " + UID);
            new NdefReaderTask().execute(tag);
        } else {
            Log.d(TAG, "Mime type errato: " + type);
        }
    } else if (NfcAdapter.ACTION_TECH_DISCOVERED.equals(action)) {
        Tag tag = intent.getParcelableExtra(NfcAdapter.EXTRA_TAG);
        UID = bin2hex(tag.getId());
        String[] techList = tag.getTechList();
        String searchedTech = Ndef.class.getName();
        for (String tech : techList) {
            if (searchedTech.equals(tech)) {
                new NdefReaderTask().execute(tag);
                break;
            }
        }
    }
}

static String bin2hex(byte[] data) {
    return String.format("%0" + (data.length * 2) + "X", new BigInteger(1, data));
}

```

Il metodo `NdefReaderTask` serve per avviare un task in background per leggere il contenuto del tag.

```

private class NdefReaderTask extends AsyncTask<Tag, Void, String> {
    @Override
    protected String doInBackground(Tag... params) {
        Tag tag = params[0];
        Ndef ndef = Ndef.get(tag);
        if (ndef == null) {
            return null;
        }
        NdefMessage ndefMessage = ndef.getCachedNdefMessage();
        NdefRecord[] records = ndefMessage.getRecords();
        for (NdefRecord ndefRecord : records) {
            if (ndefRecord.getTnf() == NdefRecord.TNF_WELL_KNOWN &&
Arrays.equals(ndefRecord.getType(), NdefRecord.RTD_TEXT)) {
                try {
                    return readText(ndefRecord);
                } catch (UnsupportedEncodingException e) {
                    Log.e(TAG, "Encoding non supportato", e);
                }
            }
        }
        return null;
    }
    private String readText(NdefRecord record) throws
UnsupportedEncodingException {
        byte[] payload = record.getPayload();
        String textEncoding = ((payload[0] & 128) == 0) ? "UTF-8" : "UTF-16";
        int languageCodeLength = payload[0] & 0063;
        // String languageCode = new String(payload, 1, languageCodeLength, "US-
ASCII");

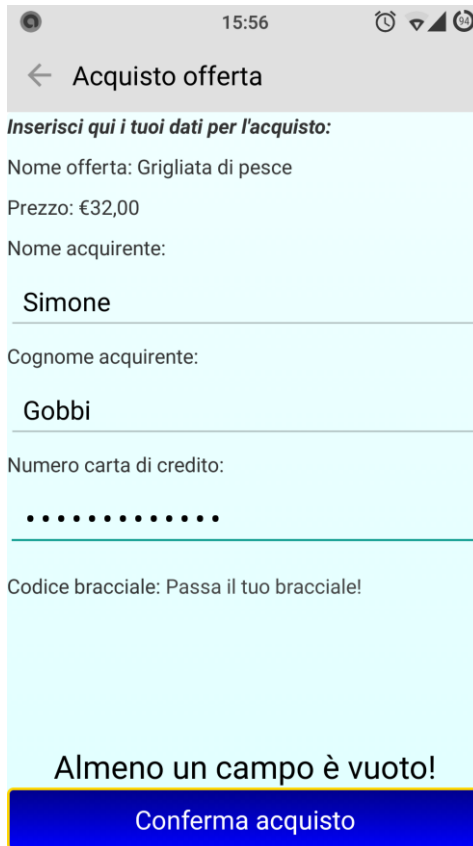
        return new String(payload, languageCodeLength + 1, payload.length -
languageCodeLength - 1, textEncoding);
    }
    @Override
    protected void onPostExecute(String result) {
        if (result != null) {
            codiceBracciale.setText(result);
        }
    }
}

```

Finita l'esecuzione del task per la lettura del chip NFC, scriverà nell'apposita TextView il codice appena letto.

Analizziamo ora l'acquisto dell'offerta: dopo aver inserito le proprie informazioni e passato il proprio bracciale, si può premere il tasto apposito. Da qui possono

aprirsi due scenari: l'acquisto va a buon fine o meno. Sarà l'applicazione stessa ad accorgersi se fosse stato omesso qualche dato. Durante la connessione verrà mostrata una progressDialog per mostrare all'utente che l'applicazione sta effettivamente lavorando.



The screenshot shows a mobile application interface for purchasing an offer. The title bar at the top is grey and contains a back arrow, the text "Acquisto offerta", and the time "15:56". Below the title bar, the text "Inserisci qui i tuoi dati per l'acquisto:" is displayed. The form contains the following fields and values:

- Nome offerta: Grigliata di pesce
- Prezzo: €32,00
- Nome acquirente: Simone
- Cognome acquirente: Gobbi
- Numero carta di credito: .....
- Codice bracciale: Passa il tuo bracciale!

At the bottom of the form, a red error message states "Almeno un campo è vuoto!". Below the error message is a blue button with the text "Conferma acquisto".

Figura 5: errore per campo vuoto

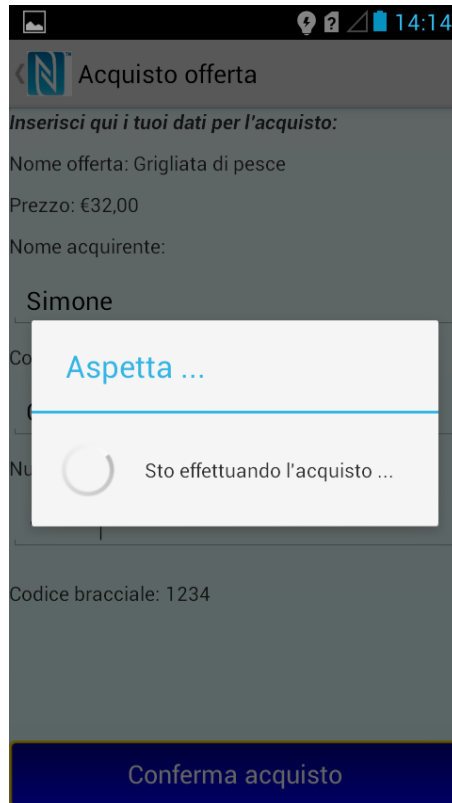


Figura 6: attesa durante l'acquisto



Figura 7: acquisto effettuato con successo

Il codice legato all'acquisto è il seguente:

```
Button acquista = (Button) findViewById(R.id.buttonConferma);
acquista.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        final ProgressDialog ringProgressDialogConnessione =
        ProgressDialog.show(AcquistoOfferta.this,
            "Aspetta ...",
            "Sto effettuando l'acquisto ...",
            true);
        new Thread(new Runnable() {
            @Override
            public void run() {
                try {
                    connect();
                } catch (Exception e) {
                }
                ringProgressDialogConnessione.dismiss();
            }
        }).start();
    }
});
```

Al tocco, viene mostrata prima la ProgressDialog e poi viene fatto partire un nuovo Thread che userà la funzione connect per aprire la connessione al server.

```
public void connect(){
    String sNome = nomeAcquirente.getText().toString();
    String sCognome = cognomeAcquirente.getText().toString();
    String sCarta = cartaCredito.getText().toString();
    String sBracciale = codiceBracciale.getText().toString();
    String sUsato = "0";
    Calendar date = Calendar.getInstance();
    String sData = date.getTime().toString();
    String sOfferta = nome;
    String sPrezzo = prezzo.substring(1);

    if(sNome.equals("") || sCognome.equals("") || sCarta.equals("") ||
    sBracciale.equals("")) {
        risultatoAcquisto.setText("Almeno un campo é vuoto!");
    }
    else {
        try
        {
            DefaultHttpClient httpClient = new DefaultHttpClient();
            HttpPost httpPost = new
            HttpPost("https://www.*****.org/services/nfc/insert.php");

            List<NameValuePair> postParameters = new
            ArrayList<NameValuePair>();
```

```

postParameters.add(new BasicNameValuePair("sNome", sNome));
postParameters.add(new BasicNameValuePair("sCognome",
sCognome));
postParameters.add(new BasicNameValuePair("sCarta", sCarta));
postParameters.add(new BasicNameValuePair("sBracciale",
sBracciale));
postParameters.add(new BasicNameValuePair("sUsato", sUsato));
postParameters.add(new BasicNameValuePair("sData", sData));
postParameters.add(new BasicNameValuePair("sOfferta", sOfferta));
postParameters.add(new BasicNameValuePair("sPrezzo", sPrezzo));
postParameters.add(new BasicNameValuePair("sUID", UID));
httppost.setEntity(new UrlEncodedFormEntity(postParameters));

HttpResponse response = httpClient.execute(httppost);
risultato = true;
dopoAcquisto();
}
catch(Exception e)
{
e.printStackTrace();
risultato = false;
dopoAcquisto();
}
}
}
}

```

E' possibile notare come inizialmente si ricavino tutte le informazioni per poter usare la pagina insert.php, mostrare un messaggio di errore nel caso ci fosse un campo vuoto e, altrimenti, creare la richiesta attraverso il metodo post con i parametri.

Successivamente verrà gestito l'esito con la funzione dopoAcquisto.

```

private void dopoAcquisto(){
    runOnUiThread(new Runnable() {
        @Override
        public void run() {
            String messaggio;
            String ris;
            if(risultato) {
                messaggio = "L'acquisto é stato effettuato!";
                ris = "ACQUISTO EFFETTUATO";
            } else {
                messaggio = "Errore! L'acquisto non é stato effettuato!";
                ris = "ACQUISTO NON EFFETTUATO";
            }
            AlertDialog.Builder alertDialogBuilder = new
            AlertDialog.Builder(AcquistoOfferta.this);
            alertDialogBuilder.setTitle("Acquisto");
            alertDialogBuilder.setMessage(messaggio);

```

```

// set positive button: Yes message
AlertDialogBuilder.setPositiveButton("Torna alla home", new
DialogInterface.OnClickListener() {
    public void onClick(DialogInterface dialog, int id) {
        Intent intent = new Intent(AcquistoOfferta.this,
        MainActivity.class);
        startActivity(intent);
    }
});
// set negative button: No message
AlertDialogBuilder.setNegativeButton("Cambia i dati", new
DialogInterface.OnClickListener() {
    public void onClick(DialogInterface dialog, int id) {
        dialog.cancel();
    }
});

AlertDialog alertDialog = AlertDialogBuilder.create();
alertDialog.show();

if(risultato) {
    risultatoAcquisto.setText("ACQUISTO EFFETTUATO");
    nomeAcquirente.setText("");
    cognomeAcquirente.setText("");
    cartaCredito.setText("");
} else {
    risultatoAcquisto.setText("ACQUISTO NON EFFETTUATO");
}
});
}

```

Quando verrà eseguita, scriverà sulla TextView precedentemente nascosta l'esito e mostrerà un alertDialog chiedendo se si vuole tornare alla home o se si vogliono cambiare i dati per permettere un secondo acquisto per un'altra persona.

## L'app esercenti

Come detto precedentemente, l'app per gli esercenti permette di visualizzare gli acquisti effettuati da un determinato bracciale e, eventualmente, selezionarne uno per cambiare il suo stato in usato.

Anche questa applicazione è stata realizzata tramite Eclipse utilizzando il linguaggio Java e xml e utilizza gli stessi permessi per l'nfc e per la connessione internet.

Per gestire meglio gli acquisti è stata creata una classe Acquisto.java da cui ottenere delle istanze utili a contenere tutte le informazioni usate anche nell'applicazione per i membri della community.

```
public class Acquisto {  
    private String Nome;  
    private String Cognome;  
    private String Carta;  
    private String Bracciale;  
    private String Usato;  
    private String Data;  
    private String Offerta;  
    private String Prezzo;  
  
    public Acquisto(String no, String co, String ca, String br, String us, String da,  
String of, String pr){  
        Nome = no;  
        Cognome = co;  
        Carta = ca;  
        Bracciale = br;  
        Usato = us;  
        Data = da;  
        Offerta = of;  
        Prezzo = pr;  
    }  
    public String getNome(){  
        return Nome;  
    }  
    public String getCognome(){  
        return Cognome;  
    }  
    public String getCarta(){  
        return Carta;  
    }  
    public String getBracciale(){  
        return Bracciale;  
    }  
}
```



```
    public String getUsato(){  
        return Usato;  
    }  
    public String getData(){  
        return Data;  
    }  
    public String getOfferta(){  
        return Offerta;  
    }  
    public String getPrezzo(){  
        return Prezzo;  
    }  
}
```

## *MainActivity.java*

E' l'unica activity di questa app che mostrerà tramite TextView il codice bracciale, un tasto che permette la ricerca sul server degli acquisti e una ListView personalizzata per mostrarne l'elenco.



**Figura 8: schermata iniziale app esercenti**

Il layout personalizzato mostrerà i dati non sensibili dell'acquisto (quindi non mostrerà il numero della carta di credito) e il prezzo pagato (che sarà comunque noto all'esercente).

La figura 9 mostra ciò che si vedrà una volta ricercata la lista degli acquisti.

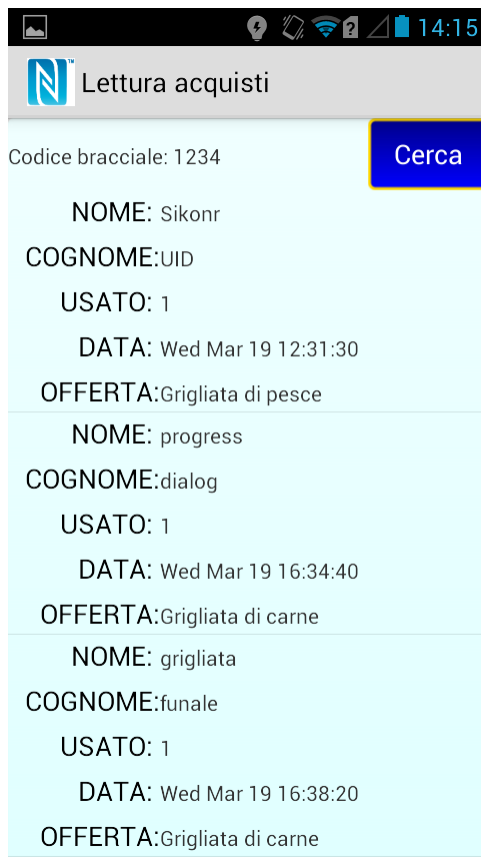


Figura 9: dopo la lettura

All'interno dell'activity, si usa praticamente lo stesso codice mostrato in precedenza per leggere il chip del bracciale e per questo motivo non verrà riproposto.

Vediamo invece il funzionamento del tasto cerca:

```

Button buttonCerca = (Button) findViewById(R.id.buttonSelect);
buttonCerca.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        final ProgressDialog ringProgressDialogConnessione =
            ProgressDialog.show(MainActivity.this,
                "Aspetta ...",
                "Scarico la lista degli acquisti ...",
                true);
        new Thread(new Runnable() {
            @Override
            public void run() {
                try {
                    connect();
                } catch (Exception e) {
                }
            }
        })
    }
}

```

```

        ringProgressDialogConnessione.dismiss();
    }
    }).start();
}
});

```

Anche in questo caso verrà mostrata una ProgressDialog in attesa della risposta del server e verrà fatta partire la funzione connect (che ovviamente non avrà le stesse operazioni della precedente).

```

public void connect() {
    //http post
    try{
        HttpClient httpClient = new DefaultHttpClient();
        HttpPost httppost = new
        HttpPost("http://www.****.org/services/nfc/select.php?bracciale=" +
        etBracciale.getText().toString() + "&UID=" + UID );
        nameValuePairs.add(new BasicNameValuePair("Bracciale",
        etBracciale.getText().toString()));
        nameValuePairs.add(new BasicNameValuePair("UID", UID));
        httppost.setEntity(new UrlEncodedFormEntity(nameValuePairs));
        HttpResponse response = httpClient.execute(httppost);
        HttpEntity entity = response.getEntity();
        is = entity.getContent();
    }catch(Exception e){
        Log.e("log_tag", "Error in http connection "+e.toString());
    }
    //convert response to string
    try{
        BufferedReader reader = new BufferedReader(new
        InputStreamReader(is, "iso-8859-1"),8);
        StringBuilder sb = new StringBuilder();
        String line = null;
        while ((line = reader.readLine()) != null) {
            sb.append(line + "\n");
        }
        is.close();
        result=sb.toString();
    }catch(Exception e){
        Log.e("log_tag", "Error converting result "+e.toString());
    }

    try{
        JSONArray jArray = new JSONArray(result);

        Acquisto acquistoApp;
        String nomeApp = "NOME";
        String cognomeApp = "COGNOME";
        String usatoApp = "USATO";
        String cartaApp = "CARTA";
    }

```

```

String braccialeApp = etBracciale.getText().toString();
String dataApp = "DATA";
String offertaApp = "OFFERTA";
String prezzoApp = "PREZZO";

for(int i = 0; i < jsonArray.length(); i++){
    JSONObject json_data = jsonArray.getJSONObject(i);
    nomeApp = json_data.getString("Nome");
    cognomeApp = json_data.getString("Cognome");
    cartaApp = json_data.getString("Carta");
    usatoApp = json_data.getString("Usato");
    dataApp = json_data.getString("Data");
    offertaApp = json_data.getString("Offerta");
    prezzoApp = json_data.getString("Prezzo");

    acquistoApp = new Acquisto(nomeApp, cognomeApp, cartaApp,
    braccialeApp, usatoApp, dataApp, offertaApp, prezzoApp);
    list.add(acquistoApp);
}
runOnUiThread(new Runnable() {
    @Override
    public void run() {
        CustomAdapterOptimize adapter = new
        CustomAdapterOptimize(MainActivity.this,
        R.layout.layout_acquisto, list);

        //utilizzo dell'adapter
        listaAcquisti.setAdapter(adapter);
        listaAcquisti.setOnItemClickListener(new OnItemClickListener() {

            @Override
            public void onItemClick(AdapterView<?> parent, View view, int
            position, long id) {
                posizione = position;
                openAlert(view);
            }
        });
    }
});
}catch(JSONException e){
    Log.e("log_tag", "Error parsing data "+e.toString());
}
}

```

In questo caso la funzione farà partire la pagina select.php con i dati del bracciale e leggerà la risposta del server convertendola dal formato json. Nel farlo creerà una lista di Acquisto che andrà a popolare la ListView per mostrare l'elenco degli acquisti.

Ora sarà possibile selezionarne uno per convalidarlo come usato.

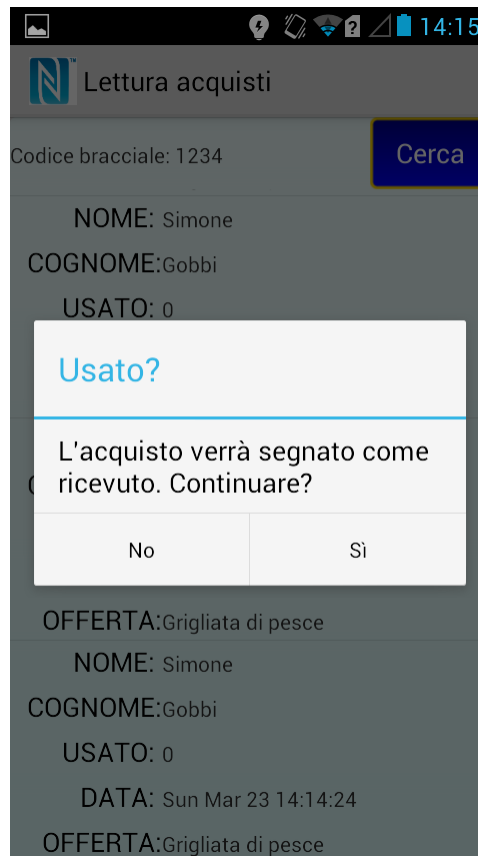


Figura 10: alert per segnare l'offerta

La figura 10 mostra come al tocco di un elemento verrà mostrato un AlertDialog che richiederà se continuare o meno la procedura. In caso affermativo, si farà partire la funzione `modificaAcquisto()`, si cancellerà il contenuto della lista attuale e si ricaricherà l'elenco.

```
private void openAlert(View view) {  
    AlertDialog.Builder alertDialogBuilder = new  
    AlertDialog.Builder(MainActivity.this);  
    alertDialogBuilder.setTitle("Usato?");  
    alertDialogBuilder.setMessage("L'acquisto verrà segnato come ricevuto.  
    Continuare?");  
    // set positive button: Yes message  
    alertDialogBuilder.setPositiveButton("Si",new  
    DialogInterface.OnClickListener() {  
        public void onClick(DialogInterface dialog,int id) {  
            final ProgressDialog ringProgressDialog =  
            ProgressDialog.show(MainActivity.this,  
                "Aspetta ...",  
                "Modifico l'acquisto ...",  
                true);  
            new Thread(new Runnable() {
```

```

        @Override
        public void run() {
            try {
                modificaAcquisto();
                list.clear();
                connect();
            } catch (Exception e) {
            }
            ringProgressDialog.dismiss();
        }
    }.start();
}
});
// set negative button: No message
alertDialogBuilder.setNegativeButton("No", new
DialogInterface.OnClickListener() {
    public void onClick(DialogInterface dialog, int id) {
        dialog.cancel();
    }
});

AlertDialog alertDialog = alertDialogBuilder.create();
alertDialog.show();
}

```

La funzione `modificaAcquisto()` richiamerà la pagina `update.php` con i parametri dell'acquisto da modificare.

```

private void modificaAcquisto(){
    Acquisto appoggio = (Acquisto) list.get(posizione);

    String sNome = appoggio.getNome();
    String sCognome = appoggio.getCognome();
    String sCarta = appoggio.getCarta();
    String sBracciale = appoggio.getBracciale();
    String sUsato = appoggio.getUsato();
    String sData = appoggio.getData();
    String sOfferta = appoggio.getOfferta();
    String sPrezzo = appoggio.getPrezzo();

    try
    {
        DefaultHttpClient httpClient = new DefaultHttpClient();
        HttpPost httpPost = new
        HttpPost("https://www.****.org/services/nfc/update.php");

        ArrayList<NameValuePair> nameValuePairs = new
        ArrayList<NameValuePair>();
        nameValuePairs.add(new BasicNameValuePair("sNome", sNome));
        nameValuePairs.add(new BasicNameValuePair("sCognome",
        sCognome));
    }
}

```

```
nameValuePairs.add(new BasicNameValuePair("sCarta", sCarta));
nameValuePairs.add(new BasicNameValuePair("sBracciale",
sBracciale));
nameValuePairs.add(new BasicNameValuePair("sUsato", sUsato));
nameValuePairs.add(new BasicNameValuePair("sData", sData));
nameValuePairs.add(new BasicNameValuePair("sOfferta", sOfferta));
nameValuePairs.add(new BasicNameValuePair("sPrezzo", sPrezzo));
nameValuePairs.add(new BasicNameValuePair("sUID", UID));

httppost.setEntity(new UrlEncodedFormEntity(nameValuePairs));
HttpResponse response = httpclient.execute(httppost);
}
catch(Exception e)
{
    e.printStackTrace();
}
}
```



# Possibili miglioramenti e sviluppi futuri

---

Il progetto così realizzato è già di per sé utilizzabile, ma certamente non privo di ottimizzazioni e, soprattutto, di incremento di altre funzionalità.

Il difetto maggiore è la mancanza di un metodo per inserire le offerte nel database. Due possibili soluzioni sono:

1. Creare un activity nell'app dedicata alla lettura degli acquisti facendo inserire i dati necessari in un form;
2. Creare un'interfaccia web con un modulo da compilare.

Le due soluzioni possono anche coesistere utilizzando una pagina php da creare appositamente.

Anche il database è per ora essenziale; è possibile, infatti, suddividere i vari dati in base alla loro correlazione piuttosto che tenerli tutti raggruppati. Una possibile soluzione può avere le seguenti tabelle:

- Utente → conterrà i dati anagrafici più i dati del bracciale;
- Esercente → conterrà i dati dell'azienda che proporrà le offerte;
- Offerta → avrà le informazioni dell'offerta e sarà collegato all'esercente che l'avrà creato;
- Acquisto → racchiuderà gli elementi dell'acquisto e collegherà di fatto l'offerta desiderata al compratore.

Questa suddivisione permetterebbe l'introduzione della registrazione nelle due app sia per gli utenti che per gli esercenti per evitare di reintrodurre i dati ad ogni acquisto e ad ogni accesso. Inoltre si avrebbe una maggiore privacy permettendo di limitare la consultazione dei commercianti alle sole offerte di propria competenza.

Un successivo sviluppo è possibile utilizzando adeguatamente l'orario dell'acquisto, dato già memorizzato dall'applicazione realizzata. Infatti ciò

permetterebbe di definire i limiti temporali di accesso a singole offerte e consentirebbe di limitare ulteriormente l'elenco mostrato nell'app degli esercenti alle sole informazioni valide nella data in corso.

Pensando ad una normale transazione in qualsiasi negozio, viene spontaneo immaginarsi l'operatore battere lo scontrino alla cassa. Quest'ultima spesso è collegata ad un computer per registrare immediatamente l'operazione ed è questo passaggio che dà un'idea per un possibile sviluppo del progetto: se collegassimo al computer un lettore nfc si potrebbe, tramite software implementato appositamente, avere le stesse funzioni dell'app esercenti sul pc e avere una maggiore praticità rispetto allo smartphone.

# Conclusioni

---

Con questa tesi è stato possibile creare un primo prototipo funzionante di applicazione per smartphone che possa permettere l'acquisto di offerte e che sfrutti i tag nfc che verranno introdotti prossimamente nei bracciali di "The Ananke". L'integrazione con l'applicazione del social network è ancora da sviluppare, ma certamente è stato possibile vedere la piena fattibilità del progetto.

Questa esperienza mi ha permesso di capire come creare e organizzare un intero progetto e come distribuire il carico di lavoro affrontando le inevitabili problematiche che si incontrano sviluppandolo. Le varie fasi di analisi, progettazione e sviluppo hanno richiesto numerose consultazioni di guide e soprattutto di esempi pratici e tutorial.