Universitatea Tehnică din Cluj-Napoca
Facultatea de Electronică,Telecomunicații și Tehnologia Informației

Gestionarea cheltuielilor la cumpărături folosind scanarea codurilor QR

Îndrumător:
Dr.ing. Adriana STAN

Proiect realizat de:
Ionela Mariana Bălțat
TC master an II

2017

1 Descrierea aplicaţiei

Aplicaţia poate fi folosită pentru gestionarea cheltuielilor la cumpărături într-un magazin mare super/hyper-market) folosind scanarea etichetelor cu cod QR. Aplicaţia permite scanarea de coduri QR generate după un format specificat şi afişarea detaliilor produsului scanat. După scanarea unui produs se vor afişa detaliile despre acel produs precum: denumire produs, descriere, preţ şi se oferă posibilitatea de adăugare în coşul de cumpărături. Aplicaţia creată pentru gestionarea cheltuielilor la cumpărături permite adăugarea, vizualizarea, modificarea cantităţii şi ştergerea produselor din coş. De asemenea aplicaţia calculează valoarea produselor din coş pentru a uşura munca utilizatorului şi a-l ajuta la gestionarea cumpărăturilor.

Aplicaţia se numeşte GestionareCheltuieli şi este compusă din două activităţi (view-uri). Prima activitate, reprezentată de pagina de start a aplicaţiei care are ca titlu „Produse scanate” este compusă dintr-un toolbar ilustrat în Figura 1. Acesta conţine patru elemente: o imagine cu un coş de cumpărături care permite navigarea către pagina „Cart” , un contor al produselor din coş în colţul din dreapta al imaginii descrise anterior, titlul paginii şi un buton de închidere al aplicaţiei.

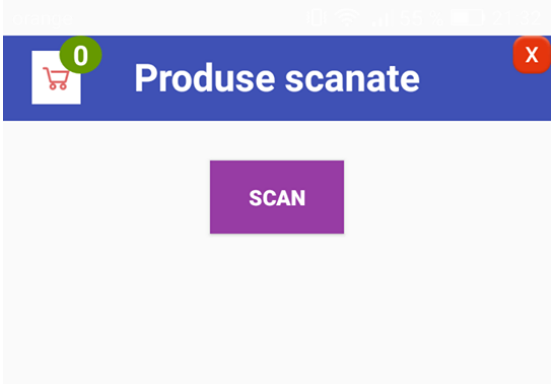

Figura 1. Prezentare toolbar din pagina de start

În Figura 1 se poate obeserva şi butonul „SCAN”. Acest buton permite scanarea codurilor QR. La click pe acest buton se va deschide camera principală a telefonului pentru identificarea unui cod QR sau BarCode. Această aplicaţie poate citi şi coduri de tip BarCode, însă nu este implementată interpretarea acestora, aplicaţia fiind axată pe interpretarea de coduri QR. În Figura 2 este ilustartă o imagine a ecranului după declanşarea procesului de scanare prin apăsarea butonuului „SCAN”.

Figura 2. Pornirea scanerului

După detectarea unui cod, acesta este interpretat şi dacă nu respectă formatul impus atunci se va afişa pe ecran mesajul „Codul scanat nu este valid” . Dacă codul scanat respectă formatul predefinit atunci se va crea un container cu detaliile produsului care se va adăuga în lista de produse scanate afişată sub butonul de „SCAN”. Modelul listei de produse scanate este ilustrat în Figura 3.

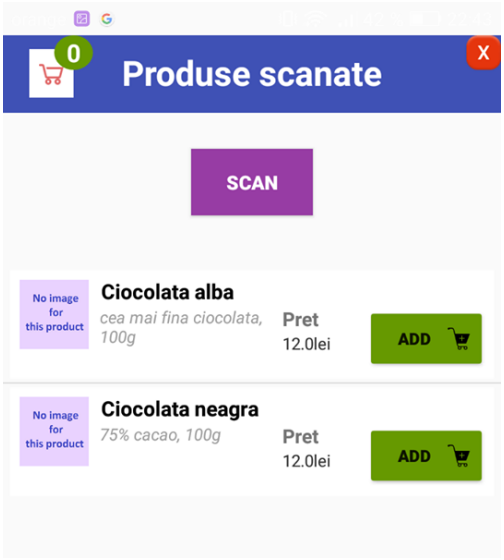

Figura 3. Lista produse scanate

Din Figura 3 se poate observa că pentru fiecare produs scanat avem o imagine, denumirea produsului, o scurtă descriere, preţul şi un buton „ADD” care permite adăugarea acelui produs în coşul de cumpărături. Dacă imaginea produsului nu este găsită în aplicaţie, se va afişa imaginea default cu textul „No image for this product”, care apare şi în Figura 3. La apăsarea butonului ADD, produsul va fi adăugat în coş şi contorul produselor din coş va fi incrementat. Pentru a vizuliza şi gestiona produsele din coş se va apăsa butonul din staga sus din toolbarul acestei activităţi, care ne va deschide view-ul cu lista produselor adăugate în coş. Pagina coşului de cumpărături este ilustartă în Figura 4.

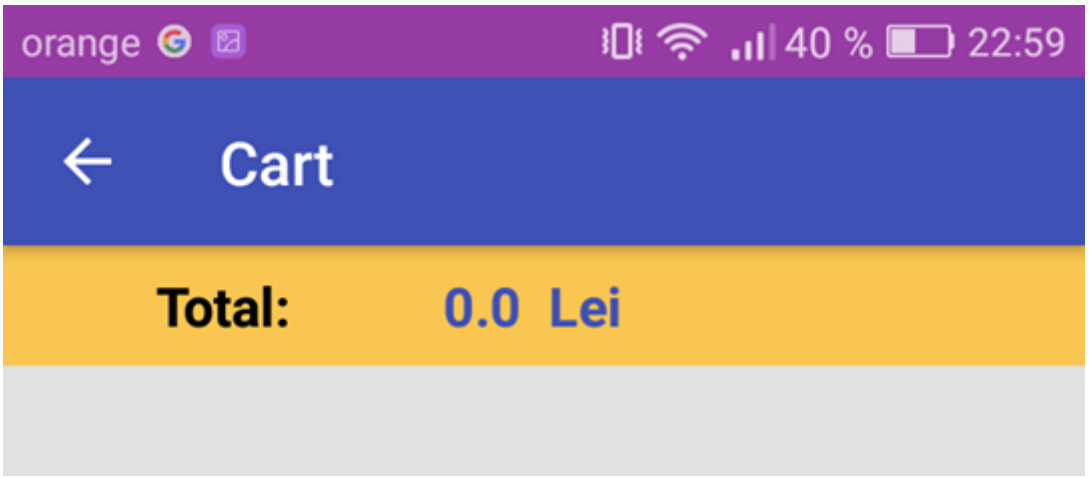

Figura 4. Pagina cosului de cumparaturi

În Figura 4 se pot observa câteva elemente ale activităţii „Cart” (pagina coşului de cumpărături). În partea stângă sus avem o săgeată „Back” care ne permite întoarcerea la pagina principală „Produse scanate”, apoi avem titlul care descrie acestă activitate „Cart” şi un

container galben care afişează valoarea totală a produselor din coş (valoarea coşului). Această valoare este recalculată la fiecare modificare a produselor din coş (adăugare, ştergere, creşterea cantităţii sau scăderea cantităţii unui produs). Produsele existente în coş vor fi afişate într-o listă care conţine imaginea, denumirea, descrierea, preţul şi cantitatea fiecărui produs. Cantitatea unui produs poate fi modificată cu ajutorul butoanelor „+" şi „-", ilustarte în Figura 5. Dacă cantitatea unui tip de produs ajunge la valoarea 0, acel produs va fi şters din coş. De asemenea dacă descrierea unui produs este mai mare de 50 de caractere atunci aceasta va fi limitată la 50 de caractere, iar restul descrierii va fi ascunsă şi înlocuită de (...). Pentru a afişa întreaga descriere se va da click pe textul deja afişat al descrierii.

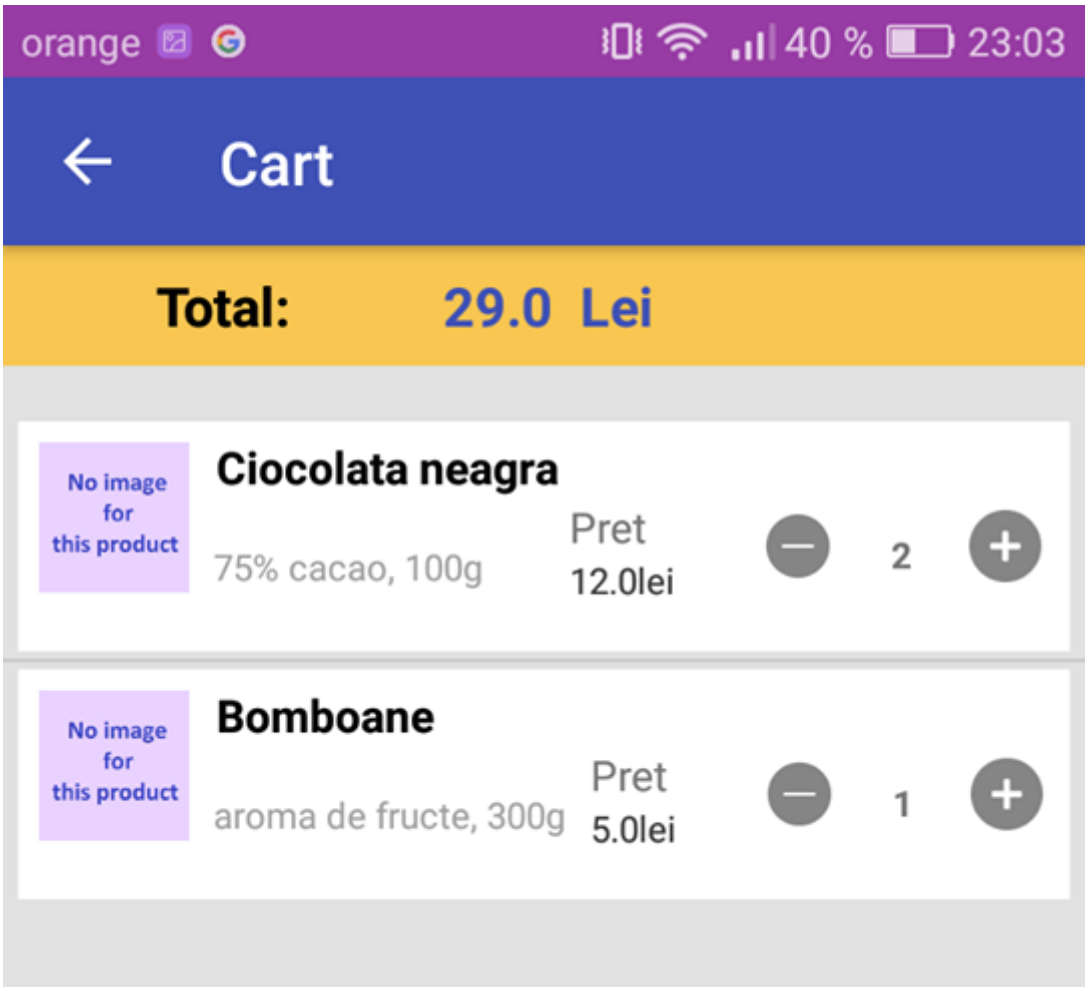

Figura 5. Lista produse din cosul de cumparaturi

Din Figura 5 se poate obesrva că în coş sunt 3 produse (2 produse „Ciocolată neagră" şi un produs „Bomboane") şi valoarea toatală a coşului este de 29 de Lei. Numărul de produse existente în coş este afişat şi în pagina principală „Produse scanate" în counter-ul pentru produsele din coş afişat în partea stângă sus, într-un cerc verde ce poate fi observat şi în Figura 6.

Figura 6. Counter produse cos

2 Implementarea aplicației

Aplicația este compusă din două activități: „activity_main.xml” și „activity_cart.xml”. În layout-ul „activity_main” sunt patru butoane, fiecare având atașat câte un eveniment de tipul „OnClick”.

```
public void onClick(View v) {
   // implement the OnClickListener callback
    //Close button (close application)
    if(v.getId()==R.id.close_btn){
        finish();
        System.exit(0);
    }
    //Scan button (Open camera to scan)
    if(v.getId()==R.id.scan_button){
        IntentIntegrator scanIntegrator = new IntentIntegrator(this);
        scanIntegrator.initiateScan();
    }
    //Cart button (change to Cart activity)
    if(v.getId()==R.id.cartButton){
        Intent intent = new Intent(this,Cart.class);
        startActivity(intent);
    }
}
```

La click pe butonul de închidere a aplicaţiei (id = close_btn) este apelată funcţia *finish()* care închide activitatea curentă, iar *System.exit(0)* opreşte execuţia programului şi aplicaţia se va închide. La declanşarea evenimentului click pe butonul „*cartButton*” este apelat constructorul *Intent* care are doi parametri. Primul parametru reprezintă contextul(se foloseşte this deoarece *Activity* este o subclasă a *clasei Context*), iar al doilea parametru reprezintă clasa componentei aplicaţiei către care se livrează conținultul, în cazul de față activitatea care va fi pornită.

Un aspect important al aplicaţiei îl reprezintă integrarea scanerului de coduri de bare şi QR în aplicaţie. Acest lucru este realizat folosind pachetul *zxing*. Pentru implementarea funcționalităților acestei aplicaţii am importat clasele *IntentIntegrator* şi *IntentResult* ale acestui pachet.

```
import com.google.zxing.BarcodeFormat;
import com.google.zxing.integration.android.IntentIntegrator;
import com.google.zxing.integration.android.IntentResult;
```

Începerea procesului de scanare este declanșat de apelarea funcției *initiateScan()* la evenimentul click al butonului cu id = ***scan_button .***

```
IntentIntegrator scanIntegrator = new IntentIntegrator(this);
scanIntegrator.initiateScan();
```

Pentru a obține și rezultatul scanării este folosită *clasa onActivityResult(**int** requestCode, **int** resultCode, Intent intent)* care parsează răspunsul scanerului astfel:

```
IntentResult scanningResult =
IntentIntegrator.parseActivityResult(requestCode, resultCode, intent);
```

- se verifică ca *scanningResult* să nu fie nul
- dacă este diferit de nul se prelucrează datele scanate și se adaugă într-un ArrayAdapter

- dacă răspunsul este nul atunci se va afișa un mesaj pe ecran "Nu s-au primit date de la scaner" folosind instanțierea unui obiect *Toast* cu metoda *makeText ()*. Această metodă are trei parametri: contextul aplicației, mesajul text si durata de afișare a mesajului. Ea returnează un obiect *Toast* corect inițializat și se va afișa notificarea prin apelarea metodei *show ()*.

Datele scanate sunt validate și adăugate în șirul *scannedProducts* de tipul *ArrayList<Property>*.

```
MystaticVar.scanedProducts.add(new Property(tokens[0], tokens[1], price, 1,
img));
adapter.notifyDataSetChanged();
```

Clasa *Property* este o clasa custom care conține proprietățile unui produs. Constructorul acestei clase are următorii parametrii : productName, description, price, product_count, image. Clasa mai conține următoarele metode de tip getters : *getProductName(), getDescription(), getPrice(), getImage(), getProductCount()* care returnează proprietățile unui produs și o metodă de tip setter : *setProductQuatity()* care setează cantitatea unui produs.

```
public Property(String productName, String description,  Double price,int
product_count, String image)
```

Popularea listei cu detaliile produselor se face cu ajutorul unui *ArrayAdapter* customizat, *CustomListAdapter*. Acest adapter este creat in functia *onCreate* si are trei parametrii: primul este un obiect de tip *Context context*, al doilea un întreg resource, si al treilea este șirul de obiecte care vor fi afișate în lista(ArrayList<Property> objects).

```
//create a custom array adapter
adapter = new CustomListAdapter(this, 0, MystaticVar.scanedProducts);
```

Atașarea ArrayAdapter-ului customizat la lista creată în view se face prin următoarele linii de cod :

```
  //Find list view and bind it with the custom adapter
ListView listView = (ListView) findViewById(R.id.list);
listView.setAdapter(adapter);
adapter.notifyDataSetChanged();
```

Funcţia *notifyDataSetChanged()* se apelează la orice modificare a adapter-ului. Clasa *CustomListAdapter* creată moşteneşte clasa *ArrayList<Property>* şi la randarea unei liste care are un Adapter ataşat se apelează funcţia *getView(int position, View convertView, ViewGroup parent)* în interiorul căreia se formatează şi pupuleaza elementele unui rând din listă. Există un template creat pentru elementele listei nmit *property_layout.xml*. De asemenea pentru activitatea *activity_cart* este definit un alt adapter asemenator cu cel descris mai sus, numit *CartListAdapter*. Deoarece am avut nevoie de un templete diferit pentru lista produseleor din activitatea *Cart* am definit un alt template, *property_cart_layout.xml*, pentru ficeare rând din listă, folosit în adapter-ul *CartListAdapter*. Acest template este populat cu elementele din ArrayList-ul *cartProd*, definit în clasa publică *MystaticVar*. Motivul definirii celor trei

variabile de tipul public static în clasa *MystaticVar* este de a se păstra elementele adăugate în aceste array-uri la redirecţionarea dintr-o activitate în alta.

Codurile QR generate pentru aceasta aplicație trebuie să aibă următorul format :

```
denumire_produs;descriere_produs;pret;denumire_imagine_produs[/ -]
```

Denumire_produs, descriere_produs și denumire_imagine_produs sunt de tipul string, iar pretul poate lua numai valori numerice (int, float), altfel codul scanat va fi invalid. În imaginea de mai jos sunt atașate 3 coduri QR pentru produsele: Poiana cu visine (a), Ciocolata cu rom (b) și Arahide(c). Exemplul prin care s-a generat primul cod (a) este următorul:

```
Poiana cu visine;ciocolata fina cu crema de visine;5;poiana_cu_visine
```

Figura 7 . Exemple de coduri QR valide

Dificultăţile întâmpinate la crearea acestui proiect au fost întâmpinate la crearea unui ArrayAdapter customizat, menţinerea datelor la trecerea dintr-o activitate în alta şi actualizarea datelor dintr-o activitate în alta. De exemplu modificarea counter-ului produselor din coş când se modifică cantitatea din pagina coşului de cumpărături, menţinerea listei cu produsele adăugate în coş dacă se revenea la prima pagină şi apoi din nou înapoi la pagina coşului.

3 Concluzii

Acestă aplicaţie este utilă pentru a putea scana produsele pe care dorim să le adăugăm în coşul de cumparaturi intr-un supermarket care are produse cu astfel de coduri şi a menţine uşor o evidenţă a produselor deja adaugte în coş şi valoarea acestora. De asemenea aplicaţia permite doar scanarea informativă a produselor fără ca acestea să fie adăugate automat în coş. Ştergerea produselor din coş, creşterea cantităţii sau scăderea acesteia precum şi vizuoalizarea numărului de produse deja adăugate în coş şi actualizarea valorii totale la orice modificare a produselor sunt alte avantaje ale folosirii acestei aplicaţii.

Îmbunătățirile care pot fi aduse aplicaţiei sunt: memorarea în baza de date a sesiunii unui utilizator pentru a păstra produsele scanate sau adăugate în coş şi în cazul în care utilizatorul închide aplicaţia din gresala, îmbunătăţirea design-ului, generalizarea formatului de coduri QR valide şi de asemenea interpretarea codurilor de tip BarCode.