**Tehnici de Programare in Internet**

**Proiect**

Aplicatie pentru licitatii on-line

Profesor coordonator: Laurentiu Boicescu

Student: Cristian Retea

Grupa: 434D

Cuprins:

1. Introducere

* Obiectivul lucrarii
* Continutul lucrarii
* Schema bloc

1. Teoria lucrarii

* Aspecte teoretice
* Aspecte tehnice
* Aspecte sociale

1. Dezvoltarea aplicatiei UPBid

* Functionalitatea aplicatiei
* Descrierea caracteristicilor cheie ale claselor folosite

1. Concluzii

* Concluziile proiectului
* Perspectiva de dezvoltare ulterioara
* Bibiografie

1. **Introducere**

**Obiectivul lucrarii**

Obiectivul lucrarii consta in familiarizarea cu tehnicile de programare folosite in crearea si dezvoltarea de aplicatii prin abordarea unei teme de proiect : *Licitatii on-line*. Pentru aceasta tema se va folosi mediul de programare Android Studio.

**Motivatia**

Am ales aceasta tema deoarece implica atat elemente de tehnica programarii, cat si elemente de intelegere a filozofiei market-ing-ului online, atat din punctul de vedere al vanzatorului cat si cel al cumparatorului. Pe langa aceste aspecte, aplicatia va imbina elemente de programare, grafica si imbinarea intre cele doua.

**Continutul lucrarii**

Lucrarea va contine aplicatia propriu-zisa, ulterior referita sub numele de UPBid. Lucrarea va vrea sa prezinte aspecte de natura teoretica, tehnica si sociala in legatura cu aplicatia dezvoltata, aspecte ce vor fi mai departe descries:

1. Aspecte teoretice:

* limbajul de programare folosit in Android Studio pentru functionalitate;
* limbajul de programare folosit in Android Studio pentru aspectul vizual al aplicatiei;
* diferite elemente specifice activitatilor mediului folosit.

1. Aspecte tehnice:

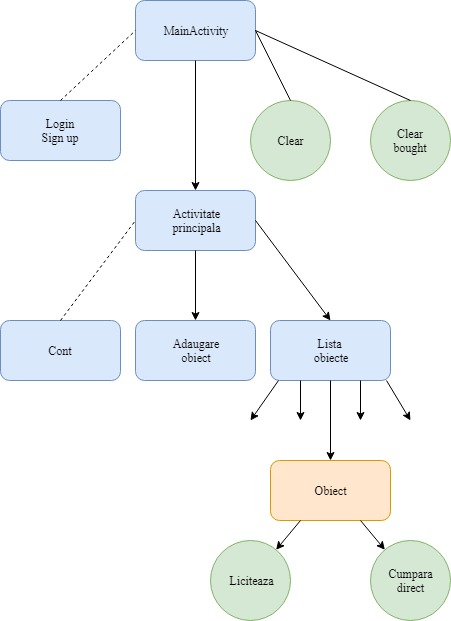
* biblioteci folosite in realizarea aplicatiei;
* functii predefinite folosite in realizarea aplicatiei;
* functii proprii ale aplicatiei;
* elemente de grafica;

1. Aspecte sociale:

* viziunea cumparatorului;
* viziunea vanzatorului;
* punerea la dispozitie a unui mediu usor de folosit si sigur pentru satisfacerea celor doua parti;

In continuare vom vrea sa prezentam notiunile teoretice precizat mai sus.

**Schema bloc**

****

1. **Teoria lucrarii**

**Programare orientate pe obiect**

Programarea orientate pe obiect(POO) este o paradigma de programare care foloseste “obiecte” – structure de date constand din campuri si metode de data – si interactiunile lor pentru a proiecta aplicatii si programe de calculator. Tehnicile de programare pot include caracteristici precum ascunderea informatiilor, abstractizarea datelor, incapsularea, modularitatea, polimorfismul si mostenirea.

Programarea orientate pe obiect este o paradigm dezvoltata pentru prima data in anii 1960, care are multe caracteristici unice. Nu a folost folosit in mod obisnuit in dezvoltarea aplicatiilr software de masa pana in inceputul anilor 1990. Multe limbi modern de programare suporta acum POO.

Una din primele caracteristici unice ale acestei paradigm ste utilizarea “obiectelor”. Obiectele sunt structure de date care contin “campuri” si “metode” pentru a descrie continutul si functionalitatea acestora. Obiectele in sine corespun tipurilor care adesea corespund datelor relevante pentru metodele pe care le contine. De exemplu, un obiect persoana ar putea contine un nume si o functie care ar putea tipari acest nume. Mai multe obiecte pot fi create dintr-un singur tip.

**XML**

Extensible Markup Language(XML) este un limbaj care definste un set de reguli pentru codificarea documentelor intron format care poate fi citit atat de om cat si de masina.

Scopurile de proiectare ale XML subliniaza simplitatea, generalitatea si utilitatea pe internet. Este un format de date textual cu suport puternic prin intermediul Unicode pentru diferite limbaje. Desi proiectarea XML se concentreaaz pe documente, limbajul este folosit pe scara larga pentru reprezentarea structurilor de date arbitrare, cum ar fi cele utilizate in web.

Mai multe sisteme de schema exista pentru a ajuta la definirea limbilor bazate pe XML, in timp ce programatorii au dezvoltat numeroase interfete de programare a aplicatiilor(API) pentru a ajuta la prelucrarea datelor XML.

**Activitati in Android Studio**

Daca de POO si XML putem vorbi la modul general, activitatile sunt specifice mediului de programare Android Studio.

Clasa “Activitate” este o component cruciala a unei aplicatii Android, iar modul in care sunt lansate si realizate activitatile este o parte fundamental a modelului de aplicatie al platformei. Spre deosebire de paradigmele de programare in care aplicatiile sunt lansate cu ajutorul metodei main(), sistemul Android initiaza codul intr-o instate a activitatii prin invocarea unor metode de apel invers specific care corespund anumitor etape ale ciclului sau de viata. Atunci cand o aplicatie invoca o alta aplicatie, apelul invoca o activitate in cealalta aplicatie, mai degraba decat aplicatia ca un intreg. In acest fel, activitatea serveste ca punct de intrare pentru interactiunea unei aplicatii cu utilizatorul. Se implementeaza o activitate ca o subclasa a clasei Activitate.

O activitate ofera o fereastra in care aplicatia isi deseneaza interfata folosita de utilizator. Aceasta fereastra, in mod obisnuit, umple ecranul, dar poate fi si mai mica decat ecranul, plutind deasupra altor ferestre. Cele mai multe aplicatii contin mai multe ferestre, ceea ce inseamna ca acestea cuprind mai multe activitati. In general, o activitate este numita activitate principala, care apare atunci cand utilizatorul lanseaza aplicatia. Fiecare activitate poate apoi sa inceapa o alta activitate pentru a efectua diferite actiuni.

Pentru a utiliza activitati intr-o aplicatie, acestea trebuie inregistrate in manifestul aplicatiei si trebuie gestionate in mode corespunzator.

**Aspecte tehnice**

S-au folosit o serie de biblioteci, functii predefinite, elemente de grafica si elemnte specific emediului de programare pentru realizarea aplicatiei.

Bibliotecile folosite ofera posibilitatea dezvoltatorului sa foloseasca o serie de elemente de grafica precum: butoane, TextView, scheme de text de intrare, scheme de aranjare in pagina a elementelor; cat si o serie de obiecte predefinite ale mediului cum ar fi: Intents, Bundles,Toast, Log, Context, Gson, ArrayList, SharedPreferences.

Pe langa functiile predefinite ale obiectelor de mai sus, in aplicatie se gasesc functii specific obiectelor create pentru functionalitatea aplicatiei.

**Aspecte sociale**

S-a vrut indeplinirea cerintelor cumparatorilor si vanzatorilor intr-o aplicatie usor de folosit care sa ofere celor doua parti oportunitatea de a interactionea intr-o schema de marketing clasica: vanzator-cumparator.

Vanzatorul cauta o metoda simpla prin care sa poata gasi persoanee carora sa le vanda obiectele pe care nu le mai doreste intr-un mod simplu, fara nevoia de intalnire cu cumparatorul, printr-un mediu online sigur: o aplicatie.

Cumparatorul cauta o metoda simpla prin care sa poata gasi obiecte de care are nevoie, fara nevoia de intalnire cu vanzatorul, printr-un medie online sigue: o aplicatie.

Cele doua viziuni se pot intalni foarte usor intr-o aplicatie de licitatii online prin care cei doi sa interactioneze. Orice utilizator poate folosi aplicatia atat din postura de vanzator cat si din cea de comparator.

1. **Dezvoltarea aplicatiei UPBid**

**Functionalitatea aplicatiei**

Fiind o aplicatie de licitatii online, UPBid urmareste sa-si indeplineasca scopul, implementand unor metode simple de vanzare si cumparare. Pentru asta, vom avea nevoie de un obiect java care va contine caracteristicile unui obiect ce va putea fi pus la licitatie. In acest scop, cream obiectul “Item” care va avea urmatoarele caracteristici: numele, descrierea, pretul de start al licitatiei, pretul curent al obiectului, pretul de cumparare directa si doua variabile de tip boolean “stare” si “isAuctioned” ce ne vor ajuta ulterior sa luam decizii. Acest obiect va avea metodele clasice de POO(getter,setter), metode de obtinere si schimbare a campurilor obiectului.

Se va folosi o lista de elemente de tip “item” care va ramane salvata chiar daca aplicatia va fi inchisa. Cele doua metode prin care se va salva si incarca lista se gasesc in clasa Utils sub numele de saveDataList si loadDataList. Aceste doua metode folosesc SharedPreferences, prin care se vor stoca date intr-o memorie a telefonului. Aceasta clasa ofera garantii puternice de consistenta.

Din pagina principala a aplicatiei putem alege una din cele doua functionalitati:

* Adaugarea unui obiect la licitatie;
* Accesarea listei de obiecte;

Activitatea de adaugare a obiectului va cere intrari pentru campurile necesare constructorului clasei Item(nume, descriere, pret de start, pret de cumparare directa).

Clasa care va contine lista se va completa cu lista de obiecte la fiecare intrare in activitate, urmand ca fiecare obiect sa aiba activitatea proprie unde gasim atat detalii despre acesta cat si metodele de licitare si cumparare.

**Descrierea caracteristicilor cheie ale claselor folosite**

1. **MainActivity.java si activity\_main.xml**

Clasa java va contine patru butoane. Primele doua butoane vor instantia alte activitati, in timp ce ultimele doua vor apela metode ale aceste clase:

* Pagina principala - duce la activitatea paginii principale a activitatii;
* Autentificare - duce la activitatea de creare cont/ autentificare;
* Clear – apeleaza metoda de stergere a tuturor elementelor din lista;
* Clear bought – apeleaza metoda de stergere a elementelor din lista care au fost deja cumparate;

1. **LoginActivity.java si activity\_login.xml – SignUpActivity.java si activity\_sign\_up.xml**

Aceste clase nu gestioneaza datele de intrare, urmarind doar ca textele primate sa fie formatul necesar si campurile sa nu fie lasate necompletate.

Clasa LoginActivity foloseste metode de validare a textului introdus de utilizator, urmarind ca aceste sa introduca un text de forma unui email. Parola este ascunsa.

Link-ul “Nu ai cont?Creeaza-ti unul!” duce catre SignUpActivity, unde se verifica datele introduse de utilizator: Numele, adresa, email-ul si parola.

In ambele cazuri, la introducerea unor date inadecvate se vor afisa erori.

1. **PaginaPrincipalaActivity.java si activity\_pagina\_principala.xml**

Aceasta clasa va contine trei butoane, dintre care doua sunt mai importante:

* Lista Items – duce la activitatea care va afisa elementele, unul cate unul;
* Adaugare Item – duce la activitatea care contine metoda de adaugare a unui obiect nou;

1. **AddItemActivity.java si activity\_add\_item.xml**

Clasa java va contine metoda de adaugare a obiectelor si metodele de validare a informatiei primate in timp ce fisierul de tip xml va contine codul pentru elementele grafice afisate la intrare in clasa.

Pentru fiecare camp necesar constructorului de obiect vom avea un element de tip TextInputLayout unde utilizatorul va adauga caracteristicile obiectului pe care vrea sa-l vanda(nume, descriere, pret de start, pret de cumparare directa). Metodele numite validateName, validatePretStart, validatePretBuyout si validateDesciere vor verifica datele introduse de utilizator:

* Se verifica daca au fost completate campurile;
* Se impune ca pretul de cumparare directa sa fie mai mare decat pretul de start;

Daca metodele de validare esueaza e vor afisa erori corespunzatoare. In cazul in care metodele de verificare reusesc se va trece la crearea obiectului si adaugarea acestuia intr-o lista globala. Dupa adaugarea obiectului in lista se va apela metoda Utils.saveDataList pentru salvarea listei la nivel global.

1. **ListaItemsActivity.java si activity\_lista\_items**

Aceasta clasa va contine doar cadrul in care vor fi afisata obiectele din lista. Pentru afisarea acestor obiecte, unul cate unul, vom folosi un adapter modificat.

1. **ItemAdapter.java**

Modelul adaptorului functioneaza ca o punte intre doua interfete incompatibile. Acest tip de model design intra sub model structural deoarece acest model combina capacitatea a doua interfete independente.

Clasa ItemAdapter, creata pentru afisarea obiectele de tip Item, unul cate unul, va extinde clasa RecyclerView.Adapter<ItemAdapter.ExampleViewHolder>. Prin aceaste metoda de POO, adapter-ul folosit de noi va mosteni metodele adapter-ului specific mediului de programare folosit.  
 Folosing aceasta utilitate vom putea afisa fiacare obiect din lista sub forma unei carti, intr-un CardView. Adapter-ul foloseste un exemplu(example\_item.xml) al design-ului obiectului si apoi adapteaza fiecare obiect la acesta, asezandu-l intr-o lista. Metodele clasei sunt de fapt @Overrider-uri asupra metodele adapter-ului original.

Tot cu ajutorul adapter-ului, folosind inca un exemplu(activity\_item\_detail.xml), vom putea lansa activitati in momentul in care se acceseaza un item(click).

1. **ItemDetail.java si activity\_item\_detail.xml**

In aceasta clasa vom avea metodele principale ce ofera functionalitatea aplicatiei:

* Licitare;
* Cumparare directa;

La accesul unui obiect din lista se va instantia o noua activitate specifica obiectului selectat. Aceasta activitate va avea detalii despre obiectul accesat:numele, descrierea, preturile si starea obiectului(nelicitat, vandut sau licitat in momentul acesta).

Activitatea va avea doua butoane, care vor apela doua metode diferite. Unul dintre ele va apela metodapentru licitarea sumei pe care utilizatorul o introduce, iar celalalt va apela metoda de cumparare directa a obiectului. In ambele cazuri, metodele verifica mai intai starea obiectului si in cazul licitarii, suma pe care utilizator doreste sa o liciteze.

1. **Utils.java**

In aceasta clasa vom avea cele doua metode de salvare si obtinere a listei pe care o folosim. Aceasta lista se obtine de fiecare data cand intram in aplicatie. Metoda de salvare a listei va fi apelata de fiecare data cand se vor face modificare asupra acesteia.

Aceste doua metode vor folosi SharedPreferences. Pentru stocarea unei liste de obiecte se va folosi biblioteca Gson.

1. **Concluzii**

**Concluziile proiectului**

Nivelul de cunostinte de programare necesar dezvoltarii unei aplicatii la nivelul de functionalitate nu este foarte ridicat, mai ales prin prisma faptului ca pe internet exista o multitudine de ghiduri si tutoriale, scrise si video care ajuta la implementarea viziunii celui care proiecteaza aplicatia.

Datorita faptului ca profesorul coordonator nu a specificat o cerinta clara a aplicatiei, studentul poate proiecta aplicatia exact cum o vede el.

**Perspective de dezvoltare ulterioara**

In acest proiect, am urmarit realizarea functionalitatii aplicatiei, dar si implementarea unor metode simplu de folosit. In continuare, pentru ca aplicatia sa poata fi folosita, aceasta ar avea nevoie de o baza de date la care sa se conecteze, in care sa fie stocati utilizatorii si/sau obiectele licitate.

In plus, metodele ar putea fi modificate astfel incat sa fie nevoie de un utilizator autentificat pentru apelarea lor.

**Bibliografie:**

<https://developer.android.com/guide/>

<https://codinginflow.com/>

<https://codingwithmitch.com/>

<https://en.wikipedia.org/wiki/Object-oriented_programming>

<https://en.wikipedia.org/wiki/XML>