UNIVERSITATEA TRANSILVANIA

BRAȘOV

FACULTATEA DE MATEMATICǍ ŞI INFORMATICǍ

SPECIALIZAREA INFORMATICĂ APLICATĂ

LUCRARE DE LICENŢĂ

Sport only

Conducător ştiinţific

Lect. Univ. Bocu Răzvan

Absolvent

Botez Bogdan

**2020**

|  |
| --- |
| **Universitatea TRANSILVANIA din Brașov**  **Facultatea de Matematică și Informatică**  **Programul de studiu: Informatică** |

Rezumat

În această lucrare este descrisă aplicația Android numită Sport Only împreună cu toate componentele folosite pentru realizarea acesteia. Sport Only este o aplicație destinată utilizatorilor de dispozitive cu sistemul de operare Android, reprezentând un magazin cu articole sportive. Cu ajutorul acestei aplicații, utilizatorii vor avea posibilitatea achiziționării articolelor sportive dorite la o atingere de ecran distanță. Clienții își vor crea un cont unic cu ajutorul numărului de telefon și o parolă. Odată autentificați, aceștia își vor putea seta detaliile personale cum ar fi: adresa, numele sau o poză de profil. Utilizatorii au posibilitatea de a căuta produsele dorite după anumite criterii, adăugarea acestora în coș cu cantitatea dorită, iar la final confirmarea comenzii. În cazul în care sunt nehotărâți, au la disponibilitate și o listă de dorințe, unde pot adăuga produsele dorite. Totodată, fiecare comandă confirmată va fi salvată în contul acestora, împreună cu detaliile aferente: data și ora plasării comenzii, preț total, informații client, produsele achiziționate împreună cu detaliile acestora.

Cuprins

[1.Introducere 5](#_Toc43336795)

[2.Tehnologii 6](#_Toc43336796)

[2.1 Java 6](#_Toc43336797)

[2.2 Android 7](#_Toc43336798)

[2.3 Android Studio 8](#_Toc43336799)

[2.4 Baze de date. Firebase Realtime 10](#_Toc43336800)

[2.5 Biblioteci 11](#_Toc43336801)

[2.5.1 Picasso 11](#_Toc43336802)

[2.5.2 Paper 12](#_Toc43336803)

[2.5.3 Android Image Cropper 12](#_Toc43336804)

[3. Sport Only – prezentarea sistemului 13](#_Toc43336805)

[3.1 Cerințe funcționale (administrator) 13](#_Toc43336806)

[3.2 Cerințe funcționale (utilizator) 13](#_Toc43336807)

[3.3 Modele 13](#_Toc43336808)

[3.3.1 Users 14](#_Toc43336809)

[3.3.2 Products 14](#_Toc43336810)

[3.3.3 Orders 15](#_Toc43336811)

[3.3.4 Cart 15](#_Toc43336812)

[3.3.5 Wish 15](#_Toc43336813)

[3.4 Activități 16](#_Toc43336814)

[3.4.1 MainActivity 16](#_Toc43336815)

[3.4.2 HomeActivity 18](#_Toc43336816)

[3.4.3 ProductDetailsActivity 19](#_Toc43336817)

# 1.Introducere

În prima parte a acestei lucrări voi oferi informații cu privire la conținutul aplicației și motivul alegerii acestei teme.

Obiectivul principal al aplicației mele este de a aduce în rândul utilizatorilor android posibilitatea achiziționării de articole cu caracter sportiv la o atingere de ecran distanță.

Suntem contemporani cu “era vitezei”, motiv pentru care dorim și totodată suntem nevoiți să

eficientizăm timpul petrecut făcând lucruri care sunt departe de o importanță critică. Deseori petrecem mai mult timp decât ne-am dori căutând articole de îmbrăcăminte sau echipamente sportive în magazine fizice, timp pe care l-am putea petrece antrenându-ne.

Foarte mulți oameni își desfășoară activitatea profesională pe un scaun în fața unui ecran, fapt care poate duce la scăderea sănătății fizice.

Întotdeauna am iubit sportul, motiv pentru care am încercat să expun avantajele practicării unui sport, atât în rândul copiilor cât și în rândul adulților. Pentru practicarea sportului în condiții optime și de siguranță avem nevoie de articole de calitate de la firme de prestigiu.

Motivele creării acestui tip de aplicație sunt: lipsa magazinelor dedicate sportului și totodată dorința de a cumpăra articolele dorite fără a mai fi nevoit să mergi într-un magazin fizic.

Alegerea dezvoltării aplicației pe platforma Android este datorată de faptul că la momentul actual în lume există foarte mulți utilizatori de Android, o foarte mare parte a populației globului având acces la un telefon.

În continuarea lucrării voi descrie tehnologiile și instrumentele utilizate pentru implementarea aplicației. Capitolul trei conține descrierea sistemului si componentele acestuia. În capitolul patru va fi detaliat manualul utilizatorului, cu imagini reprezentative si descrieri ale acestora. La final voi prezenta concluziile și planurile de viitor în legătură cu aplicația Sport Only.

# 2.Tehnologii

## 2.1 Java

Java este un limbaj de programare obiect orientat fiind conceput de către James Gosling, Ed Frank, Chris Warth, Patrick Naughton și Mike Sheridan la Sun Microsystems și lansată în anul 1991. Acest limbaj de programare unic a avut inițial denumirea „Oak”, fiind redenumit „Java” în anul 1995. Surprinzător este faptul că motivația principală pentru crearea Java nu a fost internetul. Motivul principal a fost dat de nevoia unui limbaj independent de platformă care să poată fi utilizat în dezvoltarea de sisteme software care să poată fi încorporate în diverse produse electronice, cum ar fi: toastere, cuptoarele cu microunde, telecomenzi etc. Aplicațiile Java sunt numite WORA (Write Once Run Anywhere), referindu-se la faptul că un programator poate dezvolta cod Java pe un sistem și este de așteptat ca acesta să ruleze pe oricare alt sistem fără a avea nevoie de ajustări.

În afară de portabilitate si obiect-orientare, obiectivele au fost simplitatea, ușurința de a scrie/compila cod, ușurință în depanare, efectul acestora fiind ușurința în învățare. Fiind un limbaj cu o dimensiune mică este și robust, fapt care duce la reducerea șanselor de erori greu de reparat de către programatori. Cu toate acestea, Java are o mare putere și flexibilitate.

Este modelată după C și C++, mare parte a sintaxei și a structurii obiect orientate fiind împrumutată din aceste limbaje, fapt îmbucurător pentru programatorii familiarizați cu C++, acestora fiindu-le mult mai ușor să învețe programare în Java. Chiar dacă Java seamănă cu cele două limbaje amintite anterior, o mare parte din părțile cu o complexitate ridicată regăsite în predecesori a fost exclusă. Câteva diferențe ar fi: în Java nu există pointeri, șirurile de caractere și tablourile sunt obiecte în Java, gestionarea memoriei este automată.

Internetul este o parte esențială a tehnologiei, însă a venit și cu o parte mai puțin bună, faptul că oricine are acces la acesta, astfel apărând și virușii. Dezvoltatorii Java au rezolvat această problemă într-un mod foarte elegant, fișierele rezultate de compilatorul Java, fiind numit „cod de biți” (bytecode) , nu reprezintă cod executabil. Codul de biți este format dintr-un set de instrucțiuni optimizat proiectat pentru a fi executat de Java Virtual Machine (JVM)(Figura 1), fiind un element al Java Runtime Environment (JRE). În esență prima JVM a fost creată ca un interpretor de bytecode. Translatarea programelor Java în bytecode crește ușurința rulării unui program în alte medii de programare deoarece este necesară numai implementarea JVM pentru oricare altă platformă, de exemplu, o aplicație Java scrisă sub Windows poate fi rulată și pe Linux, fără a fi necesară o recompilare.

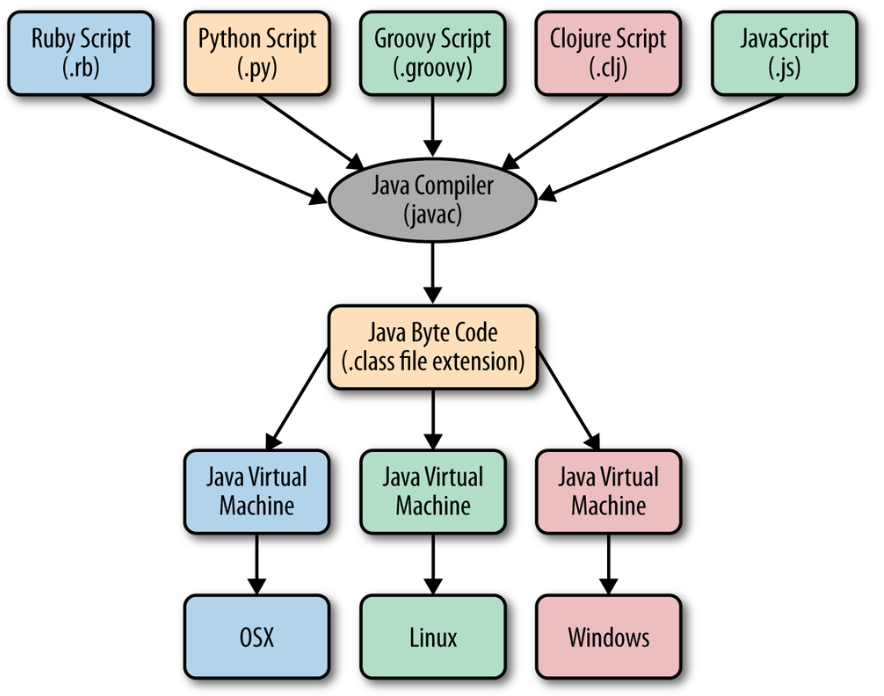


Figura 1. Proces de compilare JVM

2.2 Android

Android este un sistem de operare creat în principal pentru dispozitive touchscreen cum ar fi telefoanele mobile sau tabletele. Nucleul Android este bazat pe nucleul Linux, dezvoltatorii Google modificându-l pe cel din urmă la nivel arhitectural. A fost dezvoltat de Google, în prezent aparținând companiei Open Handset Alliance. Este un software gratis, de tip open source, codul sursă fiind cunoscut sub acronimul AOSP (Android Open Source Project) , acesta fiind în principal licențiat sub licența Apache.

Pentru programatori, Android furnizează toate instrumentele și framework-urile necesare dezvoltării de aplicații mobile ușor și rapid. SDK-ul Android este singurul lucru de care are nevoie un programator pentru a dezvolta aplicații Android.

Sistemul de operare Android este conceput din patru nivele. Fiecare nivel are propriile scopuri si caracteristici, însă acestea nu sunt total separate, având unele părți comune. Cel de la bază este reprezentat de nucleul Linux. Linux este un sistem de operare excelent din punct de vedere al portabilității, caracteristicilor și a securității. Dezvoltatorii Google au îmbunătățit nucleul Linux, eficientizând consumul bateriei.

Dalvik este o implementare specializată de mașină virtuală (VM) creată pentru utilizarea în Android. Programele Android sunt în principal scrise în limbajul Java și compilate în bytecode de către compilatorul Java, fiind mai apoi recompilate de compilatorul Dalvik în bytecod Dalvik, cel din urmă fiind executat de mașina virtuală Dalvik.

Intenția (eng. Intent) este o acțiune care are asociate o informații și este folosit pentru comunicarea dintre componentele Android. Din acest punct de vedere, Android poate fi privit ca o colecție de componente funcționale, interconectate și independente. În principal, o intenție este utilizată pentru a invoca alte activități Android sau pentru a transmite mesaje cu difuzare (eng. broadcast messages).

Majoritatea aplicațiilor Android prevăd și o interfață grafică cu utilizatorul, în acest caz conținând minim o activitate (eng. activity). O activitate poate fi descrisă ca o fereastră. De exemplu despre pagina principală a unei aplicații cum ar fi cea de autentificare, se poate spune că este o activitate. De cele mai multe ori o aplicație va conține mai multe activități.

Orice aplicație Android trebuie să conțină fișierul AndroidManifest.xml în directorul rădăcină. Acest fișier prevede informații esențiale despre aplicație cum ar fi: activități, servicii, permisiuni etc. Un layout bazat pe XML reprezintă o specificarea relațiilor dintre widget-uri. Layout-urile bazate pe XML sunt considerate resurse, fiind stocate în cadrul proiectelor Android în directoarele res sau layout. Atributele fișierelor XML descriu modul în care trebuie să arate un widget și totodată modul în care un container ar trebui să se comporte.

## 2.3 Android Studio

Android Studio este un mediul de dezvoltare (engl. integrated development environment - IDE) oficial folosit în dezvoltarea aplicațiilor Android. A fost anunțat pe 16 mai 2013 la conferința organizată anual numită Google I/O. A intrat în faza de „early access preview” începând cu versiunea 0.1 în luna prezentării, pentru ca mai apoi, cu puțin peste un an de la prezentare să apară varianta beta începând cu versiunea 0.8. Prima versiune stabilă, și anume 1.0, a fost lansată la finalul anului 2014, moment care a coincis cu încheierea suportului pentru pluginul Android Development Tools (ADT) pentru Eclipse. La conferința anuală din anul 2017 organizată de Google, limbajul de programare Kotlin a primit un loc în lista limbajelor oficiale pentru platforma Android, alături de Java si C++.

Odată cu apariția Android Studio, programatorii au parte de o interfață mai prietenoasă, și totodată de unelte mai ușor accesibile, comparativ cu predecesorul său. Cei de la Google au adăugat în Android Studio un „build automation system”, numit Gradle, folosit pentru automatizarea si gestionarea procesului de „build”([Figura 2](#Figura2)) . Acest proces este alcătuit din multe instrumente și procese care transformă proiectul într-un pachet de aplicații Android (APK). Gradle este folosit și pentru descărcarea automată de biblioteci specifice Android cu ajutorul unei singure linii de cod urmată de acționarea butonului „Sync Project with Gradle Files”.

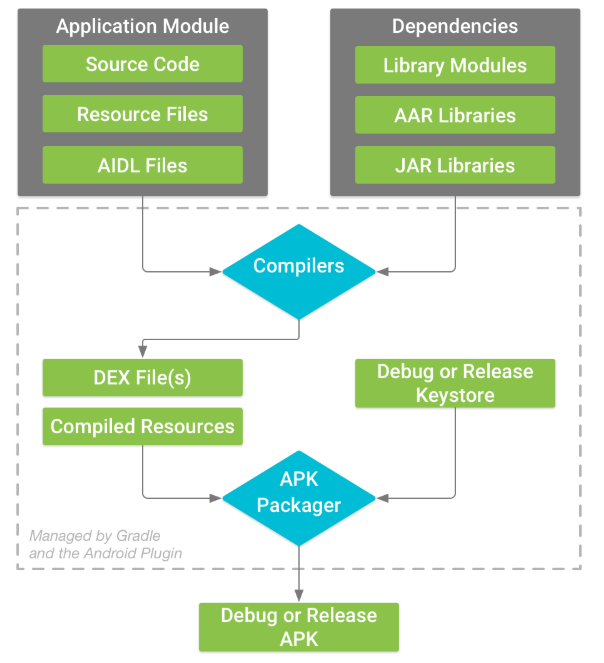


Figura 2 Procesul de „build” tipic unei aplicații Android

În mod implicit, Android studio afișează fișierele proiectului în Android view (Figura 3). Acest view nu reflectă cu exactitate ierarhia fișierelor din memoria calculatorului, fiind organizat pe module și tipuri de fișiere, cele mai puțin utilizate fiind ascunse pentru simplificarea navigării printre fișierele intens folosite. Acest view grupează fișierele în următoarele categorii:

* manifests - conține fișierul AndroidManifes.xml
* java - conține fișierele cu cod sursă Java grupate în pachete
* res – conține toate resursele non-cod cum ar fi layout-uri XML, imagini etc.

Pentru a vedea structura efectivă a proiectului vom folosi Project view(Figura 4), în acest mod putem vizualiza toate fișierele, inclusiv cele ascunse în Android view.

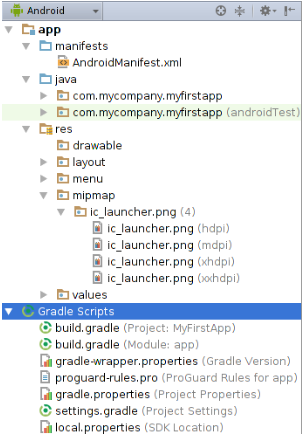
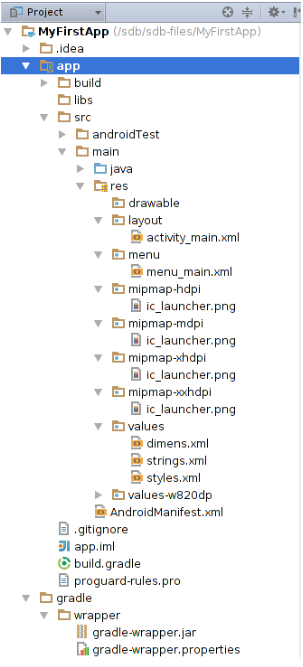
 

Figura 3. Android View

Figura 4. Project View

## 2.4 Baze de date. Firebase Realtime

Stocarea și preluarea informațiilor reprezintă de o perioadă lungă de timp două nevoi de bază, fie că vorbim despre datele personale ale cetățenilor sau de păstrarea arhivei naționale. Termenul de date poate fi descris ca un set de fapte brute izolate și fără legătură cu o semnificație implicită. De-a lungul istoriei, datele au fost salvate în depozite de mari dimensiuni sub forma unor cărți, documente sau registre. Având atât de multe informații din orice domeniu, stocarea acestora reprezenta un proces care consuma foarte mult timp și spațiu. Această problemă, a fost rezolvată odată cu dezvoltarea tehnologiei, îndeosebi a computerelor, deoarece stocarea informațiilor pe memoriile computerelor au dus la reducerea nevoii de a stoca informații pe hârtie. În prezent, computerele sunt principala unealtă pentru stocarea informațiilor importante.Creșterea numărului de informații de-a lungul timpului a dus la dezvoltarea bazelor de date. O bază de date ar putea fi descrisă ca o colecție de date prin intermediul căreia, utilizatorii pot obține eficient informația dorită.

Firebase Realtime este un model de bază de date NoSQL(baze de date nerelaționale). Acest tip de bază de date a apărut ca o alternativă pentru bazele de date relaționale. Modelele NoSQL pot avea o varietate de forme, acesta fiind și motivul pentru care au apărut, datorat de creșterea complexității aplicaților.

Firebase Realtime este o baza de date cloud-hosted. Datele sunt salvate sub forma de fișiere JSON și sunt sincronizate în timp real cu toți clienții conectați. În cazul unei aplicații Android, toți clienții împart o singură instață a bazei de date, primind în mod automat orice modificare a datelor din cadrul bazei de date.

Baza de date Firebase Realtime permite dezvoltarea de aplicații Android, permițând accesul securizat la aceasta direct din codul de client. Datele persistă la nivel local chiar dacă dispozitivul este offline, iar în momentul revenirii online, modificările locale sunt sincronizate cu actualizările apărute în timpul perioadei offline a dispozitivului, baza de date contopind conflictele eventual existente. Firebase Realtime conține un set de reguli bazate pe expresii denumit Database Security Rules, care definește modul în care sunt structurate datele, totodată putând fi stabilite drepturile pe care le au utilizatorii asupra datelor.

Cloud Storage este conceput pentru dezvoltatorii de aplicații care au nevoie de stocarea și preluarea datelor generate de utilizatori cum ar fi fotografiile sau clipurile video. Acest serviciu este unul foarte puternic, simplu și efectiv din punct de vedere al costurilor. SDK-urile Firebase pentru Cloud Storage adaugă securitate Google atât la încărcare cât și la descărcarea fișierelor pentru aplicații.

## 2.5 Biblioteci

### 2.5.1 Picasso

Picasso este o bibliotecă concepută pentru folosirea imaginilor în aplicații Android. Biblioteca permite încărcarea imaginilor foarte ușor în aplicații, deseori cu ajutorul unei singure linii de cod. Caracteristicile acestei biblioteci sunt:

* Adapter downloads – reutilizarea adaptorului este detectată automat, descărcarea anterioară fiind anulată
* Image transformations – se pot transforma imaginile pentru o potrivire mai bună în aplicație, totodată această caracteristică poate fi folosită pentru a reduce dimensiunea memoriei utilizate
* Resource loading – resursele, fișierele sunt acceptate ca imagini
* Debug indicators – se poate activa afișarea unei panglici colorate care indică sursa imaginii (Figura 5)

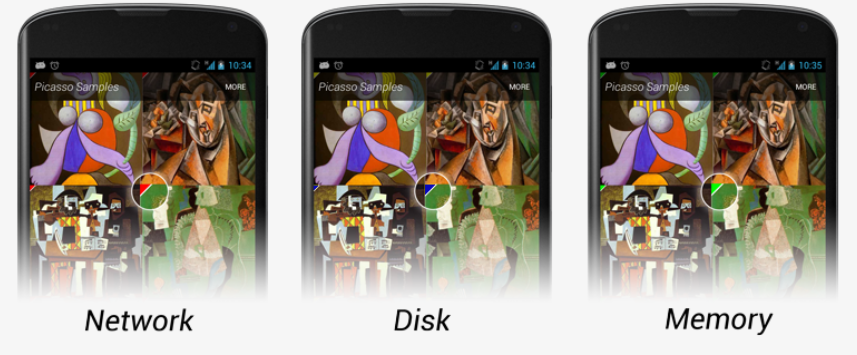


Figura 5. Picasso debug indicators

### 2.5.2 Paper

Scopul acestei biblioteci este de a oferi o alternativă simplă, dar rapidă pentru stocarea obiectelor în cadrul aplicațiilor Android. Permite utilizarea claselor Java/Kotlin în forma normală: fără adnotări, extensii obligatorii de clasă etc. Totodată, adăugarea sau eliminarea atributelor claselor nu reprezintă o problemă deoarece modificările structurilor de date sunt realizate automat. Caracteristici:

* Salvează – se poate salva orice tip de obiect (Map, List, HashMap etc.)
* Citește – clasa instanțiată este exact cea folosită pentru salvarea datelor
* Șterge – se pot șterge toate datele folosind metoda destroy sau putem alege varianta ștergerii după o anumită cheie folosind metoda delete având ca parametru cheia respectivă

### 2.5.3 Android Image Cropper

Android Image Cropper este o bibliotecă care ajută la transformarea imaginilor după bunul plac al utilizatorului. Cu ajutorul acestei biblioteci, utilizatorii pot selecta o imagine din memoria dispozitivului și o pot redimensiona sau roti pentru ca la final imaginea dorită să fie salvată la apăsarea butonului Crop.Caracteristici:

* CropImageActivity este incorporată în bibliotecă
* Imaginea rezultată poate fi setată ca bitmap, resursă sau Android URI
* Imaginea poate fi rotită în timpul decupării
* Se poate seta limita minimă sau maximă a pixelilor imaginii

# 3. Sport Only – prezentarea sistemului

În continuare voi prezenta sistemul aplicației Sport Only. Vor fi descrise componentele importante cu anumite pasaje din codul propriu zis după cum urmează.

## 3.1 Cerințe funcționale (administrator)

* Autentificare
* Ieșire din cont
* Adăugare produs nou
* Editare produs
* Vizualizare listă de produse
* Vizualizare comenzi finalizate

## 3.2 Cerințe funcționale (utilizator)

* Autentificare
* Ieșire din cont
* Vizualizare listă de produse
* Vizualizare detalii produs
* Adăugare produs în lista de dorințe
* Creare cont nou
* Adăugare produs în coșul de cumpărături
* Setare informații personale
* Căutare produse după criteriul dorit
* Vizualizare comenzi finalizate
* Confirmare comandă
* Ștergere produs din coșul de cumpărături sau lista de dorințe
* Modificare cantitate produs din coșul de cumpărături
* Vizualizare detalii comenzi finalizate
* Ștergere comenzi finalizate
* Recuperare parolă

## 3.3 Modele

Modelele din cadrul aplicației Sport Only sunt reprezentate de clasele: Cart, Orders, Users, Products, Wish. În continuare voi descrie fiecare model în parte.

### 3.3.1 Users

Modelul Users conține toate informațiile necesare despre utilizator și anume: numele acestuia, numărul de telefon, parola necesară pentru autentificare, adresa și un șir de caractere reprezentând url-ul către imaginea de profil. Pentru înregistrare, utilizatorii vor trebui să seteze numai numele, numărul de telefon care reprezintă și identificatorul unic al fiecărui client și parola. Utilizatorii se vor autentifica folosind numărul de telefon și parola aferentă. Clienții vor avea în momentul autentificării posibilitatea bifării checkbox-ului „Remember me” pentru a rămâne autentificați chiar dacă părăsesc aplicația. Acest lucru este posibil cu ajutorul bibliotecii Paper. În momentul autentificării, numărul de telefon și parola utilizatorului curent se vor salva folosind metoda write din clasa Book, această metodă returnează instanța curentă a clasei Book (Figura 6). În momentul în care închidem și redeschidem aplicația, fără a ieși din cont (moment în care se șterg toate cheile din interiorul instanței book), se preiau valorile cheilor date ca parametru în momentul salvării datelor, acest lucru fiind făcut în Activitatea principală, în continuare fiind apelată metoda verificării datelor pentru autentificare (Figura 7).

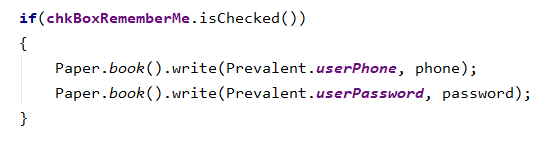


Figura 6. Salvarea datelor de conectare

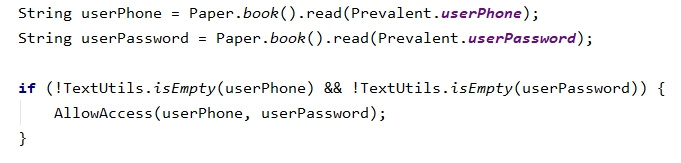


Figura 7. Preluarea datelor de conectare

### 3.3.2 Products

Modelul Products conține informațiile necesare despre fiecare produs în parte. Produsele pot fi adăugate de către administrator cu ajutorul activității AdminAddProduct. Atributele clasei Products sunt următoarele:

* pId (String) – identificatorul unic care este salvat ca data și ora la care este adăugat produsul
* name (String) – numele produsului
* description (String) – descrierea produsului
* image (String) – Url-ul imaginii
* category – categoria din care face parte produsul
* date (String) – data la care este adăugat produsul
* time (String) – ora la care este adăugat produsul
* size (String) – mărimea produsului
* gender (String) – genul pentru care este destinat produsul
* sport (String) – sportul pentru care este destinat produsul
* price (double) – prețul produsului
* quantity (int) – numărul de bucăți aflate în stoc

### 3.3.3 Orders

Modelul Orders este conceput pentru salvarea comenzilor fiecărui client în parte, alături de detaliile aferente. Pentru a realiza acest lucru clasa Orders necesită următoarele atribute:

* name – numele clientului care a finalizat comanda
* phone – numărul de telefon al clientului care a finalizat comanda
* address – adresa clientului
* date – data la care a fost finalizată comanda
* time – ora la care a fost finalizată comanda
* totalPrice – prețul total pentru comandă

### 3.3.4 Cart

Modelul Cart reprezintă coșul de cumpărături în care clienții vor adăuga produsele dorite. Pentru afișarea produselor în coșul de cumpărături nu este nevoie decât de atributele necesare, iar acestea sunt:

* pId – identificatorul unic al produsului
* name – numele produsului
* price – prețul/buc al produsului
* quantity – cantitatea de produse comandate

### 3.3.5 Wish

Modelul Wish reprezintă lista de dorințe în care clienții pot salva produsele dorite pentru o comandă ulterioară sau le pot elimina în cazul în care nu le mai doresc. Pentru afișarea produselor în lista de dorințe, clasa Wish are următoarele atribute:

* pId – identificatorul unic al produsului
* name – numele produsului
* sport – tipul sportului pentru care este destinat produsul
* category – categoria din care face parte produsul
* price – prețul produsului

## 3.4 Activități

O activitate este un singur lucru focalizat pe care utilizatorul îl poate face. În general, toate activitățile interacționează cu utilizatorul, deci clasele Activități creează o fereastră în care putem pune interfața cu utilizatorul. Activitățile sunt gestionate de sistem ca stive de activități. În momentul în care o activitate nouă este pornită, aceasta este pusă în vârful stivei și devine activă, iar cea anterioară rămâne sub aceasta în stivă. O activitate poate trece prin patru stări. Dacă o activitate apare în foreground-ul ecranului, se poate spune despre aceasta că este activă, reprezentând prima stare. A doua stare este cea în care despre activitate se poate spune că este vizibilă. De exemplu în momentul în care în urma unei acțiuni a utilizatorului apare pe ecran un dialog sau o activitate care nu acoperă în totalitate activitatea principală. Următoarea stare este cea oprită sau ascunsă. Acest lucru se poate întâmpla în momentul în care în foreground apare o altă activitate decât cea anterioară. Ultima stare, cea de-a patra, este distrugerea activității, lucru făcut de sistem sau la cerința dezvoltatorului.

Ciclul de viață al unei activități ([Figura 8](#Figura8)) începe fie la pornirea aplicației, fie când este creată de altă activitate. Prima metodă apelată în momentul creării activității este metoda *onCreate()*. Această metodă este urmată de metodele *onStart()* și *onResume()*, fiind mai apoi urmate de metodele *onPause()* și *onStop()* în cazul în care activitatea ajunge în starea a treia, descrisă anterior. În acest moment, activitatea poate fi distrusă prin apelarea metodei *onDestroy()* sau poate reapărea în foreground prin apelarea metodei *onRestart()*.

În continuare vor fi descrise implementările unor metode aparținând anumitor activități ale aplicației sau activitățile în sine.

### 3.4.1 Fereastra principală a aplicației

După cum aminteam, majoritatea aplicațiilor Android au o activitate principală. În momentul rulării aplicației *Sport Only*, este creată activitatea MainActivity. După setarea layout-ului aferent și legarea componentelor, se verifică posibilitatea autentificării clientului, dacă acesta a dorit să rămână autentificat. În caz contrar, există opțiunile de creare cont nou sau autentificare, ambele fiind făcute prin intermediul a două butoane, ambele apelând metoda *setOnClickListener()*, în cazul apăsării acestora (Figura 9).

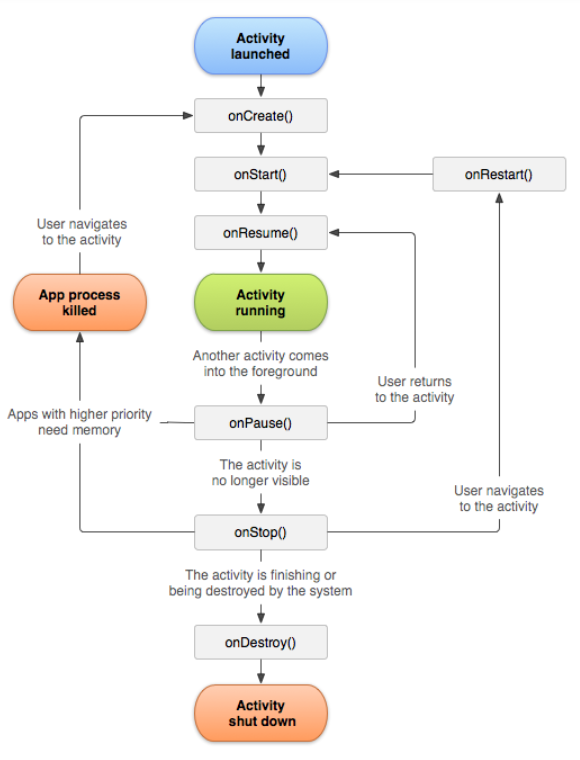


Figura 8. Ciclul de viață al activităților

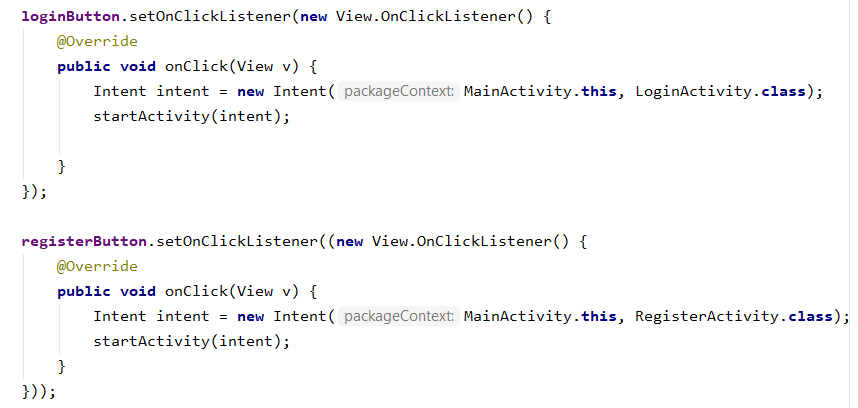


Figura 9. Butoanele Login și Register

3.4.2 Fereastra principală a utilizatorului

Dacă autentificarea este reușită, în foreground-ul aplicației va ajunge fereastra principală a utlizatorului (*HomeActivity*). În această activitate vor fi afișate inițial toate produsele existente în baza de date la momentul respectiv. Produsele vor fi preluate din baza de date cu ajutorul unei referințe la nodul din fișierul JSON în care sunt salvate datele în baza de date.



În metoda onStart() sunt preluate produsele folosind *FirebaseRecyclerAdapter*, acesta leagă un *query* de un *RecyclerView*.Întâi este necesară setarea corespunzătoare a unui obiect de tipul *FirebaseRecyclerOptions* (Figura 10), mai apoi se creează obiectul de tip *FirebaseRecyclerAdapter*, fiecare produs fiind afișat în final cu ajutorul clasei *ProductViewHolder*, fiecare atribut al produsului fiind setat în interiorul metodei *onBindViewHolder* (Figura 11). În continuare trebuie setat un *LayoutManager*, iar în final deoarece adaptorul folosește un *event listener* pentru a monitoriza schimbările din query, deci tot ce mai trebuie făcut este apelarea metodei *startListening()*. În momentul apăsării pe un anumit produs, se va crea o nouă activitate, mai exact *ProductDetailsActivity* , intenția transmițând totodată și identificatorul unic al produsului selectat (Figura 12).

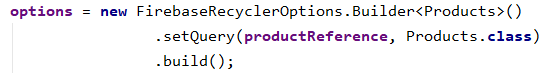


Figura 10. Query Product Reference

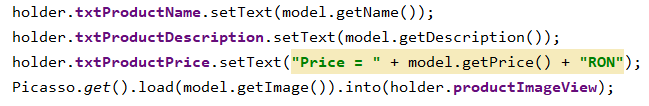


Figura 11. Setarea atributelor pentru afișare

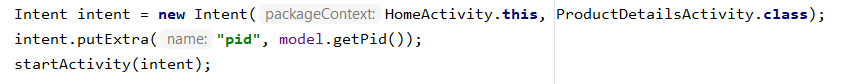
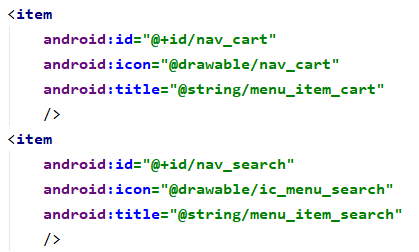


Figura 12. Crearea ProductDetailsActivity

Pentru această activitate am folosit template-ul Navigation Drawer Activity, oferit de Android Studio, în concluzie, utilizatorul va avea ca meniu al aplicației un widget de tip DrawerLayout care va conține două componente, bara de sus și un widget de tip NavigationView. NavigationView este un meniu standard pentru aplicațiile Android.

Includerea atributelor meniului este făcută cu folosind atributul app:menu cu o valoare care trimite către un fișier xml denumit activity\_home\_drawer, unde setăm fiecare item al meniului similar cu cele prezentate în continuare. În momentul apăsării pe o opțiune din meniu, se va crea o intenție nouă care va aduce în foreground-ul aplicației fereastra aferentă opțiunii din meniu(Figura 13). Variabila id reprezintă identificatorul opțiunii care a fost aleasă, aceasta fiind inițializată cu valoarea returnată metoda *getItemId*().



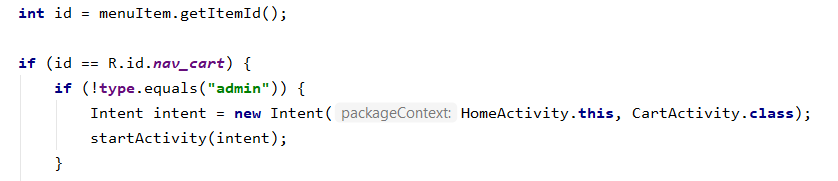


Figura 13. Acțiune în momentul alegerii opțiunii din meniu

### 

### 3.4.3 Fereastra „Detalii produs”

În momentul apăsării pe un produs, în foreground-ul aplicației va apărea fereastra detalii produs (ProductDetailsActivity). În momentul creării activității, se pasează identificatorul unic al produsului apăsat. Cu ajutorul metodei *getProductDetails* (Figura 14), având ca parametru identificatorul produsului, setăm detaliile produsului dorit. În primul rând se creează o referință către nodul produsului care se află în nodul *Products* din baza de date. Mai apoi se preiau cu ajutorul unui listener datele produsului într-un obiect de tip Products.

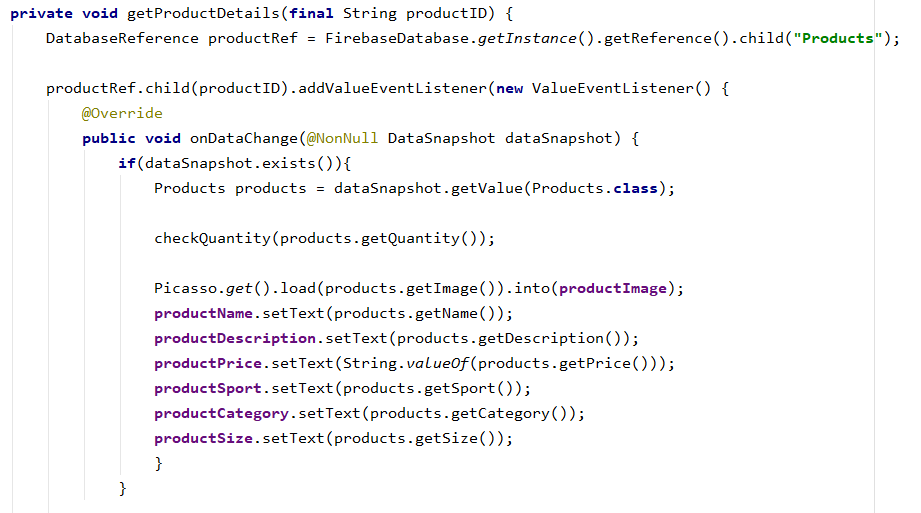
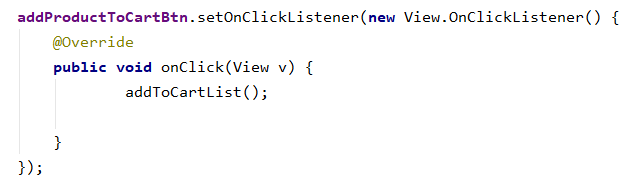


Figura 14. Setarea detaliilor produsului

După ce detaliile au fost setate, utilizatorul poate adăuga produsul în coșul de cumpărături sau în lista de dorințe. Pentru a adăuga produsul în coșul de cumpărături, utilizatorul trebuie să apese pe butonul de adăugare. Acestui buton îi este setat un listener de apăsare care va apela metoda addToCartList() în momentul în care butonul este acționat.



Metoda apelată va crea o referință către nodul „Cart List”, apoi într-un obiect de tip HashMap vor fi salvate atributele produsului. În continuare, se setează un listener către referința nodului unde vor fi salvate produsele, utilizatorul este înștiițat de faptul că produsul a fost adăugat cu succes în coșul de cumpărături și este adusă în foreground fereastra principală a utilizatorului (Figura 15).



Figura 15. Adăugarea produselor în coșul de cumpărături

### 3.4.4 Fereastra „Coș de cumpărături”

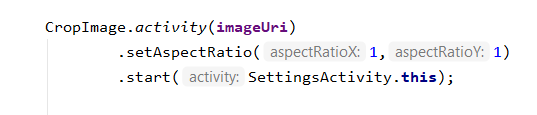
Această fereastră poate apărea în foreground în momentul apăsării opțiunii „Cart” din meniul paginii principale a utilizatorului. În momentul activării acestei activități (CartActivity), se afișează produsele într-un mod similar cu cel descris la fereastra principală a utilizatorului. Înainte de a se lega, însă, atributele produselor de viewHolder, este verificată prin apelarea metodei CheckProductQuantity cantitatea tuturor produselor din coș pentru a nu fi mai mare decât cantitatea din stock-ul magazinului. Se creează două referințe la baza de date, una pentru coșul de cumpărături și una pentru produse. Se trece apoi prin toate produsele din coșul de cumpărături și se caută, cu ajutorul identificatorului, pentru fiecare dintre acestea cantitatea totală din stocul magazinului. Dacă această cantitate totală este mai mică decât cantitatea adăugată în coș de client, se elimină din nodul coșului de cumpărături produsul respectiv. Mai apoi, în cazul în care fereastra din foreground este cea a coșului de cumpărături, se reîncarcă această fereastră, nu înainte de a fi șters vârful stivei activităților pentru a nu se putea reintra pe vechea fereastră a coșului. Această verificare a cantității este necesară deoarece stocul unui produs se poate termina sau poate scădea sub cantitatea dorită de un anumit client.

### 3.4.5 Fereastra „Confirmare comandă”

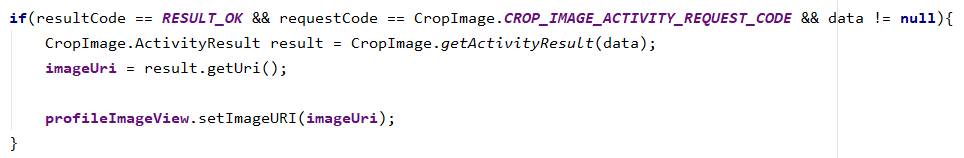
Această fereastră va apărea în foreground în momentul apăsării butonului „Next”, buton care aparține de CartActivity. În această fereastră vor fi afișate sub formă de EditText datele utilizatorului pentru care se face comanda. Inițial acestea vor fi setate ca cele ale utilizatorului curent. După apăsarea butonului de confirmare, care are setat un listener de click, se va adăuga comanda în lista de comenzi a utilizatorului curent, fiecare comandă având pe lângă datele comenzii și lista de produse care au fost achiziționate. După ce datele comenzii vor fi salvate în baza de date, cantitatea totală din stock-ul magazinului va fi updatată în funcție de cantitatea cumpărată.

### 3.4.6 Fereastra „Setări utilizator”

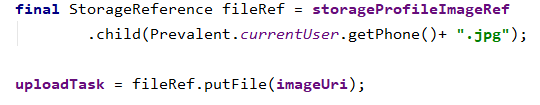
Această fereastră poate fi accesată din meniul ferestrei principale a utilizatorului, alegând opțiunea settings. SettingsActivity conține trei obiecte de tip EditText cu ajutorul cărora utilizatorii își pot schimba datele personale din baza de date, butoanele Update, Close și cel pentru setarea întrebării de securitate, un TextView care are setat un listener pentru apăsare, iar în momentul acționării acestuia se începe activitatea de redimensionare a imaginii, pentru o imagine salvată în prealabil pe dispozitiv.



Se suprascrie apoi metoda onActivityResult pentru a salva Uri-ul imaginii selectate.



În continuare se salvează imaginea în FirebaseStorage cu ajutorul unui obiect de tip StorageTask și referința dorită spre FirebaseStorage.



Detaliile utilizatorului vor fi setate la apăsarea butonului Update.

## 3.5 Baza de date

În continuare voi prezenta structura fișierului JSON, acesta reprezentând fișierul în care sunt stocate datele folosite în aplicație. Nodul Users conține datele clienților.Voi prezenta structura acestuia, folosind ca exemplu doi useri, unul dintre aceștia având toate datele completate, iar cel de-al doilea având completate datele necesare pentru crearea unui cont nou (Figura 16). Cel de-al doilea nod pe care doresc să îl prezint este cel al produselor (Figura 17).

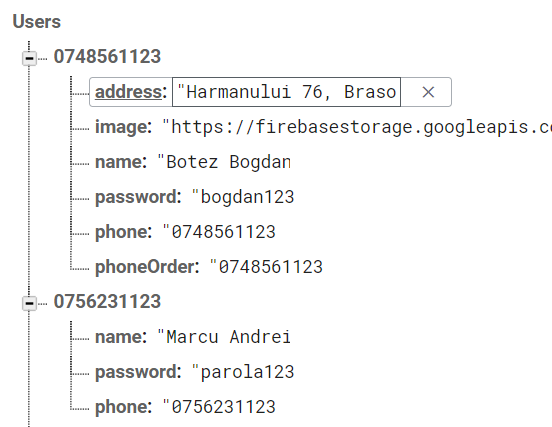


Figura 16. Structura nodului Users din baza de date

Bibliografie:

**Java Fundamentals: Programming Basics for Beginners** De Zohaib Nawaz Tarar

Teach yourself Java in 21 days – Laura lemay, Charles L. Perkins

<https://www.oreilly.com/library/view/client-server-web-apps/9781449369323/ch04.html>

<https://ocw.cs.pub.ro/courses/eim/laboratoare/laborator04>

<https://developer.android.com/studio/build>

<https://developer.android.com/studio/projects>

<https://square.github.io/picasso/>

https://developer.android.com/reference/android/app/Activity.html#ActivityLifecycle