**UNIVERSITATEA BABEȘ-BOLYAI**

**Facultatea de Științe Economice și Gestiunea Afacerilor**

**Informatică Economică**

**Lucrare de licență**

Absolvent,

Paula - Iuliana **Iscu**

Coordonator științific,

Asist.univ.dr.Paula-Georgiana **ZĂLHAN**

**2021**

**UNIVERSITATEA BABEȘ-BOLYAI**

**Facultatea de Științe Economice și Gestiunea Afacerilor**

**Informatică Economică**

**Lucrare de licență**

**CC - Store**

Absolvent,

Paula - Iuliana **Iscu**

Coordonator științific,

Asist.univ.dr.Paula-Georgiana **ZĂLHAN**

**2021**

**Cuprins**

[**1.Abrevieri** 4](#_Toc74932304)

[**2.Lista tabelelor, figurilor și a secvețelor de cod** 5](#_Toc74932305)

[**1.** **Analiza cerințelor sistemului informatic** 7](#_Toc74932306)

[**1.1** **Identificarea și descrierea problemei** 7](#_Toc74932307)

[1.2 Cerințe de sistem 10](#_Toc74932308)

[*1.2.1 Surse de cerințe* 10](#_Toc74932309)

[*1.2.2 Elicitația cerințelor* 11](#_Toc74932310)

[*1.2.3 Descrierea stakeholderilor* 13](#_Toc74932311)

[*1.2.4 Documentarea cerințelor* 19](#_Toc74932312)

[*1.2.5 Procese și activități* 19](#_Toc74932313)

[1.3 Modelul de dezvoltare 22](#_Toc74932314)

[**2.** **Proiectarea sistemului informatic** 23](#_Toc74932315)

[2.1 Proiectarea logică 23](#_Toc74932316)

[*2.1.1 Arhitectura sistemului* 23](#_Toc74932317)

[*2.1.2 Baza informațională* 26](#_Toc74932318)

[2.2 Proiectarea tehnică 26](#_Toc74932319)

[*2.2.1* *Structura fizica a datelor* 28](#_Toc74932320)

[*2.2.2 Tehnologii specifice* 28](#_Toc74932321)

[**3.** **Implementarea sistemului informatic** 32](#_Toc74932322)

[**4.** **Testarea sistemului informatic** 42](#_Toc74932323)

[**Concluzii** 45](#_Toc74932324)

[**Bibliografie** 46](#_Toc74932325)

# **1.Abrevieri**

*CRUD* Create, Read, Update, Delete

*CSS* Cascading Style Sheets

*GUI* Graphical User Interface

*HTML* HyperText Markup Language

*JS* JavaScript

*PHP* HyperText Preprocessor

*RSS* Really Simple Syndication

*SMTP*  Simple Mail Transfer Protocol

*DFD* Data Flow Diagram

# **2.Lista tabelelor, figurilor și a secvețelor de cod**

**Tabele**

[Tabel 1 Scenariul principal 18](#_Toc74832308)

[Tabel 2 Scenariul de interacțiune 19](#_Toc74832309)

[Tabel 3 Rularea aplicației 40](#_Toc74832310)

[Tabel 4 Accesul în aplicație 41](#_Toc74832311)

[Tabel 5 Administrator 42](#_Toc74832312)

**Figuri**

[Figură 1 Diagrama Fish-bone (Sursa: Proiecția autoarei) 8](#_Toc74931882)

[Figură 2 Diagrama de descompunere a obiectivelor (Sursa: Proiecția autoarei) 9](#_Toc74931883)

[Figură 3 Diagrama Pareto (Sursa: Proiecția autoarei) 12](#_Toc74931884)

[Figură 4 Diagrama Use-Case Vizitator (Sursa: Proiecția autoarei) 14](#_Toc74931885)

[Figură 5 Diagrama Use-Case Client (Sursa:Proiecția autoarei) 15](#_Toc74931886)

[Figură 6 Diagrama Use case - Gestionarea coșului de cumpărături (Sursa: Proiecția autoarei) 16](#_Toc74931887)

[Figură 7 Diagrama de activități (Sursa: Proiecția autoarei) 20](#_Toc74931888)

[Figură 8 Diagrama de secvență (Sursa: Proiecția autoarei) 21](#_Toc74931889)

[Figură 9 Diagrama de stare (Sursa: Proiecția autoarei) 21](#_Toc74931890)

[Figură 10 Diagrama Agile-Scrum (Sursa: Proiecția autoarei) 22](#_Toc74931891)

[Figură 11 Diagrama Fluxului de Date (Sursa: Proiecția autoarei) 23](#_Toc74931892)

[Figură 12 Arhitectura Client-Server (Sursa: Proiecția autoarei) 24](#_Toc74931893)

[Figură 13 Paradigma Model-View-Controller (Sursa: Proiecția autoarei) 25](#_Toc74931894)

[Figură 14 Diagrama bazei de date (Sursa:Proiecția autoarei) 28](#_Toc74931895)

[Figură 15 Administrator (Sursa: Proiecția autoarei) 32](#_Toc74931896)

[Figură 17 Autentificarea clientului (Sursa: Proiecția autoarei) 33](#_Toc74931897)

[Figură 18 Înregistrarea vizitatorului (Sursa: Proiecția autoarei) 34](#_Toc74931898)

[Figură 19 Pagina principală (Sursa:Proiecția autoarei) 35](#_Toc74931899)

[Figură 20 Istoric (Sursa: Proiecția autoarei) 35](#_Toc74931900)

[Figură 22 Subsolul paginii Home a aplicației (Sursa: Proiecția autoare) 36](#_Toc74931901)

[Figură 23 Adăugare produs nou de către utilizator (Sursa: Proiecția autoarei) 37](#_Toc74931902)

[Figură 25 Produse (Sursa: Proiecția autoarei) 38](#_Toc74931903)

[Figură 27 Coș de cumpărături (Sursa: Proiecția autoarei) 39](#_Toc74931904)

[Figură 28 Finalizare comandă (Sursa: Proiecția autoarei) 40](#_Toc74931905)

[Figură 30 Răspunsul automat trimis prin email (Sursa: Proiecția autoarei) 41](#_Toc74931906)

**Secvențe de cod**

[Listare 1 Funția de inserarea utilizatorilor 33](#_Toc74931745)

[Listare 2 Funcția de traducere 36](#_Toc74931746)

[Listare 3 Funcția foreach 38](#_Toc74931747)

[Listare 4 Filtrare în funție de preț 38](#_Toc74931748)

[Listare 5 Filtrare în funcție de nume 39](#_Toc74931749)

[Listare 6 Funcția de creare a comenzii 40](#_Toc74931750)

[Listare 7 Confirmare prin email 41](#_Toc74931751)

[Listare 8 Șir array pentru istoric 41](#_Toc74931752)

**Introducere**

În cadrul lucrării de licență am hotărât sa dezvolt o aplicație web care permite utilizatorilor să achiziționeze și să vândă haine deja folosite. Am ales această tema, deoarece hainele sunt elemente indispensabile tuturor oamenilor și au rol semnificativ in personalitatea noastră. Astfel, îmi propun ca prin acest magazin online să ofer șansa tuturor de a achiziționa haine la un preț cât mai bun, dar intr-o formă cât mai bună. Mai mult de atât, cei care nu își mai folosesc anumite articole, le pot posta pe site spre vânzare. Utilizatorii care o să își pună la vânzare hainele, sunt supuși unor condiții. Articolele nu trebuie sa fie rupte sau să aibă pete ce nu pot fi scoase. De asemenea, aceastea trebuie sa fie spălate si bine întreținute.

Motivul pentru care îmi doresc să promovez această idee, este reprezentat de faptul că de cele mai multe ori, mai cu seamă în calitate de student, nu ne permitem să cumpăram anumite haine, deoarece prețul este prea ridicat. Chiar dacă hainele nu sunt noi, prețul fiind mai mic decât cel din magazin, tind să cred ca oamenii vor fi deschiși cu ideea ce urmează să o implementez. Am considerat ca grupul țintă să fie reprezentat de elevi, studenți, persoane pe care situația financiară le limitează, dar și oamenilor cărora nu le place să investească foarte mult în vestimentație. Alternativele mai ieftine sunt bine venite mereu, iar hainele mereu vor reprezenta o necesitate a fiecărui om. Astfel, ceea ce ar putea fi considerată o pierdere pentru unii, devine un câștig pentru alții.

Majoritatea clienților pun accent pe impactul vizual, mai mult decât pe conținut. Cu ajutorul unor strategii de marketing, am hotărât ca aspectul magazinului să fie unul prietenos, să conțină informații clare și câteva imagini cu produsele disponibile. Prefer culori cât mai deschise pentru a nu fi obositor, iar navigarea să fie una simplă și rapidă. Consider că accesul nelimitat al clientului va aduce un plus aplicației.

Ca metodologie am ales să folosesc Agile, deoarece pune accent pe organizarea bine gândită. Însă ca și metolodogie am hotărât să folosesc Scrum, deoarece este procesul ce îmbunătățește comunicarea și cooperarea în echipă. Metoda Scrum se bazează pe dezvoltarea și livrarea la timp a proiectelor. Obiectivele pot fi îndeplinite în anumite intervale de timp și asta datorită așa numitelor sprinturi. La sfârșitul fiecărui sprint se oferă feedback pentru muncă depusă până în momentul de față.

Lucrarea este structurată pe capitole pentru o bună desfașurare după cum urmeaza: Capitolul 1 prezintă principalele motive pentru care am ales această temă, dar si metodele de dezvoltare. În capitolul 2 este descrisă implementarea sistemului informatic, ce limbaje am utilizat și am explicat conținutul aplicației, iar capitolul 3 este despre testarea aplicației.

# **Analiza cerințelor sistemului informatic**

În cadrul primului capitol am hotărât să vorbesc despre motivația alegerii acestei teme, dar și despre metodele de dezvoltare. Utilizarea diagramelor și a tabelelor mă va ajuta să descriu în detaliu ceea ce doresc să creez.

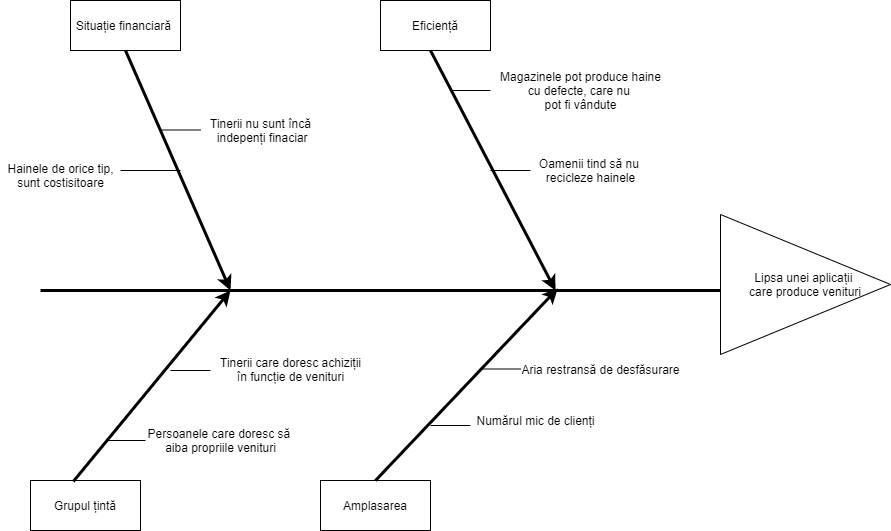
## **Identificarea și descrierea problemei**

Tehnologia a atins pragul cel mai de sus al dezvoltării și astfel există posibilitatea de a achiziționa orice, oricând, de oriunde. Trăim în secolul vitezei, dar totuși niciodată nu avem timp. Suntem ocupați mereu, iar când vine vorba de achiziții, preferăm să stăm acasă și să așteptăm curierul. Situația financiară instabilă este foarte bine cunoscuta în rândul studenților. Datorită achiziționării dintr-un magazin second-hand, prețul hainelor este scăzut, iar problema este rezolvată.

La baza unei societăți echilibrate, stă o educație echilibrată. În ultimii ani, tehnologia s-a dezvoltat iar oameni au devenit din ce în ce mai familiarizați cu ideea de online. Nevoile tinerilor stau la baza motivației pentru realizarea acestei aplicații, deoarece, ei reprezintă viitorul societății. Scopul aplicației este de a fi incurajați tinerii spre duce la bun sfârșit dorințele lor. Nevoia puternică de dezvoltare în mediul online stă la baza principalei motivații.

Sunt numeroase beneficii ale întreprinderii atunci când este transpusă în mediul online. Magazinul poate fi accesat de oriunde în lume, platforma poate fi accesată de clienți la orice oră, relația cu clienții este mult mai bine gestionată, iar informațiile sunt mult mai ușor de gestionat. Unul dintre importantele beneficii alte unui magazin online este costul scăzut al deschiderii. Cheltuielile cu achiziția sau închirierea unui spațiul comercial, întreținerea si salariul angajaților sunt scazute considerabil. Clienții pot avea produsele la un click distață, evitând deplasarea pentru achiziție. Desfășurarea comerțului poate să se extindă odată cu creearea magazinul virtual. Numarul clienților va fi ridicat, magazinul va deveni cunoscut, iar impactul va fi unul crescut.

Figura 1 (Diagrama Fish-bone) mai este denumită și diagrama cauză-efect, deoarece este utilizată pentru clarificarea obiectivelor si identificarea cauzelor ce stau la bază creării unei aplicații. Această diagrama este de ce mai multe ori folosită pentru a fi analizate procese industriale, mai cu seamă procesul pentru proiectarea produselor și în îndepărtarea greșelilor.

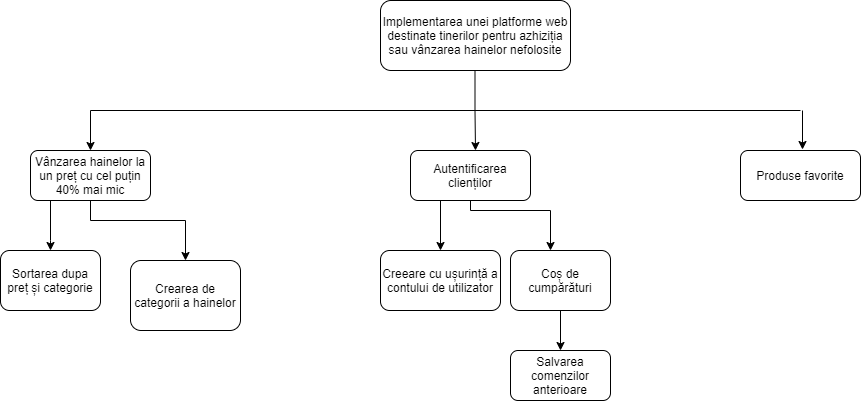
După cum se poate observa în figura 1, problema principală este ramificată în probleme secundare care au legătură între ele. Situația financiară este una dintre problemele des întâlnite, deoarece majoritatea tinerilor sunt de cele mai multe ori respinși, din pacate, doar pentru că nu au o vestimentație potrivită, care costă. Acesta a fost primul aspect la care m-am gândit în momentul în care am ales tema.

Figură 1 Diagrama Fish-bone (Sursa: Proiecția autoarei)

De asemenea, m-am axat pe un anumit grup țintă, iar acesta este reprezentat de tineri. Acești sunt intr-o continuă dezvoltare și schimbarea, dorindu-și mereu să fie în pas cu moda. Așadar, sper cu acest magazin online sa vin în ajutorul lor.

Amplasarea magazinului este o alta problemă, deoarece oamenilor le este mult mai convenabil să comande online, decât să se deplaseze fizic pentru cumpărături, iar impactul clienților poate fi surprinzător. Având în vedere că multe articole vestimentare rămân de pe un sezon pe altul, iar designerii produc modele noi, magazinele rămân cu hainele nevândute, pot veni în ajutorul lor spre a le cumpăra și a le vinde la un preț mai bun.

Acest magazin poate fi unul eficient din punct de vedere al timpului petrecut pentru cumpărături. M-am gândit, pe viitor, să pot deschide parteneriate cu desgineri vestimentari. Acest lucru constă în cumpărarea hainelor nefolosite sau rămase de pe un sezon pe altul, iar în urma unor beneficii, eu să le pot prezenta pe aplicația mea spre vânzare.

O alta metodă prin care am ales ca îmi îndeplinesc toate obiectivele este reprezentată de figura 2, mai exact metoda de descompunere a obiectivelor. Această oferă posibilitatea de a defini clar pașii ce urmează să fie parcurși pentru realizarea aplicației dorite.

Figură 2 Diagrama de descompunere a obiectivelor (Sursa: Proiecția autoarei)

Contextul soluției alese este evidențiat prin intermediul celor 4 fațete:

**Fațeta subiect** Aplicația de tip e-commerce din cadrul acestei lucrări de licență are ca scop final furnizarea unui mediu online ușor accesibil, punând la dispoziție haine la un preț mic. Scopul aplicației este de a minimiza timpul efectiv petrecut în magazine. Principalii beneficiari ai acestui magazin sunt persoanele care utilizează aplicația si ale căror interese se aliniază cu ce oferă site-ul. Grupul țintă fiind tinerii. De asemenea, este posibil pe viitor ca magazinul să pună la dispoziție haine care poate nu se mai gasesc pe piață.

**Fațeta utilizare**

Utilizatorii se impart in 3 categorii distincte:

1. Utilizatorul de tip vizitator care poate naviga pe pagina web și să vizualizeze produsele, însă nu poate plasa comenzi sau adăuga un articol nou.
2. Utilizatorul cu cont de autentificare poate face cumpărături și toate cumpărăturile îi sunt salvate în contul său, de asemenea acesta poate posta un articol nou spre vânzare.
3. Administratorul are acees la părțile aplicației care nu sunt prezente pentru utilizatorul vizitator sau cel cu cont de autentificare. Administratorul are dreptul de a modifica orice dorește cu ajutorul operațiilor CRUD. De asemenea, administratorul gestioneaza conturile în baza de date și stochează comenzile.

**Fațeta IT**

Partea de stilizare a site-ului a fost realizată cu ajutorul limbajelor CSS, HTML și JS, iar functionalitățile aplicației au fost implementate prin limbajul PHP. Pentru dezvoltare este folosit framework-ul Codeigniter și nu în ultimul rând am folosit Xampp și Apache pentru furnizarea unui server și creearea baze de date MySql, unde sunt stocate datele.

**Fațeta dezvoltare**

Dezolvarea aplicației din această lucrare de licență s-a bazat pe metodologia SCRUM, aceasta fiind una din metodele Agile de dezvoltare a proiectelor. Caracteristicile specifice acestei metode se pliază cel mai bine astfel de aplicații. Metoda scrum implică o echipă formată din persoane specializate pe diferite domenii ce au ca obiectiv realizarea aplicației. Pentru a fi îndeplinite task-urile specifice, toate persoanele lucrează în echipă. Sarcinile sunt împărțite si rezolvate pe rând.

## 1.2 Cerințe de sistem

### *1.2.1 Surse de cerințe*

În contextul acestui magazinul online, sursa de cerințe este reprezentată de tineri. Având în vedere situația financiară instabilă pe care aceștia o au și necesitatea de a fi tot timpul în ritm cu ceilalți. Principalele cerințe luate în considerare vin din partea stakeholderilor esențiali și clienții din mediul online. Se ia în considerare realizarea unei aplicații ușor de utilizat, diversificată și actualizată la cerințele clienților care ofera haine la mâna a doua. Se dorește și crearea unui sistem de administrare a utilizatorilor ceea ce va eficientiza munca depusă pentru comenzi.

Cerințele funcționale se pot împărți astfel: organizarea cât mai clară a categoriilor, navigarea ușoară și eficientă, creeare unui cont persoanalizat, adăugarea produselor în coșul de cumpărături, pagina specială pentru a revedea comenzile anterioare, mecanism de căutare a produselor dupa preț și categorie, vizualizarea descrierii si prețului produselor. Magazinele online sunt în general căutate de către clienți trebuie să fie simple, ușor de utilizat și să conțină toate informațiile referitoare la produs. Clienții aleg magazinul online , deoarece este mult mai eficient.

O altă cerință în realizarea aplicației este reprezentată de designul bine structurat si aerisit. În ceea ce privește piața concurentă, ca și creatori ai aplicației trebuie să studiem magazinele deja existente și să vedem ce oferă. “Pentru a depăși concurența trebuie să atingem targetul deja existent și să venim mereu cu ceva nou.”

### *1.2.2 Elicitația cerințelor*

Prin metoda de elicitație se dorește extragerea și înțelegerea necestăților părților implicate (stakeholderi). Astfel prin metoda interviului si brainstorming, intenționez să aflu preferințele tinerilor, nevoile acestora și observațiile cu privire la aplicațiile de tip e-commerce. Analiza finală a rezultatelor celor două metode determină maniera în care o sa dezvolt aplicația. Stakeholderii implicați sunt vizitatorii, clienții și administratorul.

Metoda de elicitație: Brainstorming

Prima metoda de elicitație folosită este brainstorming-ul. Metoda aceasta se folosește într-un grup, fiecare persoană își folosește creativitatea, iar la final vor fi generate idei diferite pentru realizarea aplicației. Problemele principale au fost: venitul instabil al tinerilor, diversitatea hainelor și necesitatea de a satisface cât mai multe persoane, informațiile oferite trebuie să fie transparente, aplicația să fie ușor de folosit cu un design plăcut și crearea contului de utilizator sa fie simplă, fara prea multe informații personale. Regulile ce au fost urmărite în procesul de brainstorming: stabilirea obiectivlor, legarea ideilor, stimularea creativității și obținerea unui număr mare de idei. Fazele de realizare a procesului au fost urmatoarele:

1. Pregătire: mi-am împărțit sarcinile pe zile pentru a nu pierde mult timp, iar în momentul în care m-am apucat de implementat am știut exact ce fac. Obiectivele au fost descrise clar si afișate pe o foaie pentru a putea urmări cu atenție toți pașii.

2. Colectarea ideilor : am notat fiecare idee pe hârtie. Le-am citit cu atenție și m-am gândit dacă pot sau nu să fac asta. Am ales în final ideile pe care urma să le implementez: aspect simplist, dar atractiv, crearea paginilor de logare/înregistrare, vizualizare produse, coș de cumpărături, istoric comenzi.

3. Ajustarea ideilor: culorile am decis sa fie unele calde, care sa dea senzația de confort, ordine, siguranță și diversitate. De asemenea s-a dorit un aspect aerisit în pagina, fără prea multe detalii ce ar putea distrage atenția. La un articol vestimentar vor fi afisate imagini, prețul și descriere, adăugarea în coș fiind posibilă direct din pagina respectivă.

4. Organizarea idelor: aplicația va conține pagini esențiale precum Home, unde sunt prezentate câteva din produse, Register unde se poate crea cont sau se pot conecta utilizatorii dacă deja există contul, Produse unde se pot vedea toate produsele disponibile, Coș unde clientul își gestionează produsele alese și Istoric Comenzi unde vor fi disponibile toate comenzile plasate, alături de suma, produse și adresă. Pe baza concluziilor obținute în urma brainstorming-ului, s-a realizat diagrama Pareto unde sunt evidențiate defectele ce apar in contextual unui site de tip e-commerce.

Figura 3 este tot de o diagramă de analizare a obiectivelor ce pune accent pe ierarhizare, în ordinea importanței. Diagrama conține atât coloane sortate cât și o linie care reprezintă procentul total cumulat. Figura 3 reprezintă instrumentul grafic care este folosit pentru a constata prioritățile, pe baza efectelor unor cauze diferite, ierarhizrea se face de la cea mai des întâlnită, până la cea mai rar întâlnită.

Figură 3 Diagrama Pareto (Sursa: Proiecția autoarei)

Metoda de elicițație – Interviul

A doua metoda utilizată pentru extragerea datelor este interviul. Astfel, am realizat câte un dialog de tip întrebare-răspuns. Au participat 3 tineri de liceu și de facultate deschiși propunerii mele.

Notă : “Î” = întrebare “R”= răspuns

1. Î: “Ați cumpărat vreodată haine din mediul online? De ce?”

R: “Sigur, deoarece magazinele online oferă diversitate în ceea ce privește tipurile de haine. Magazinul online îmi oferă timpul necesar să aleg articolul potrivit și îl pot analiza timp îndelungat.”

1. Î: “Ați fi dispus să cumpărați haine la mâna a doua sau cu posibile defecte din fabricație,dar la un preț redus?”

R: “Da, aș fi dispusă să achiziționez haine la mâna a doua datorită prețului redus. Sunt pasionată de haine noi și mi-ar plăcea să știu că le pot găsi și în funcție de bugetul meu.”

1. Î: “Care este tipul de articol vestimentar cel mai des întâlnit în garderoba?”

R: “Prefer să port blugi pentru că se potrivesc oriunde și la orice tip de papuci. Însă cel mai comod mă simt în tricou și blugi, iar in zilele reci adaug un hanorac.”

1. Î: “Ați întâlnit vreodata aplicatii de tip magazine care nu v-au mulțumit?”

R: “Am întâlnit site-uri care au prelungit timpul de livrare. Am fost nevoită să plasez mai multe comenzi, deoarece site-ul nu îmi permitea adăugarea a mai mult de 3 produse în coș.

1. Î: “Cum ați prefera să arate o aplicație web destinată hainelor?”

R: “Mi-aș dori să fie ușor de utilizat. Să aiba un design minimalist, navigarea și plasarea comenzilor sa fie simplă, iar la crearea contului sa nu fiu nevoită să acord foarte multe date personale.”

1. Î: “Ați cumpărat vreodată haine din mediul online? De ce?”

R: “Sigur, deoarece magazinele online oferă diversitate în ceea ce privește tipurile de haine. Magazinul online îmi oferă timpul necesar să aleg articolul potrivit și îl pot analiza timp îndelungat.”

**Modelul use – case**

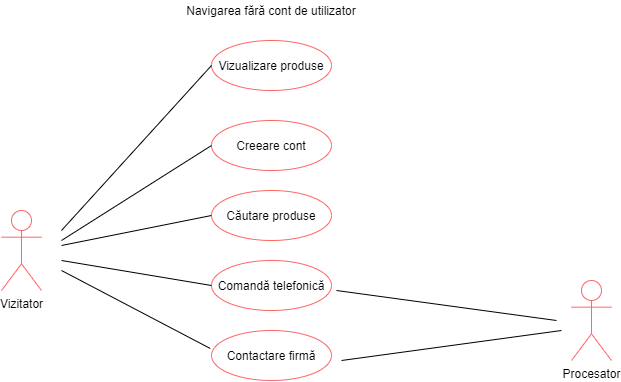
### *1.2.3 Descrierea stakeholderilor*

Vizitatorii sunt cei care pot accesa aplicația și pot vizualiza produsele disponibile, dar din păcate acesta nu poate să realizeze comenzi online. De asemenea acesta nu poate să adauge produse sau să aibă acces la istoric comenzi.

Clienții sunt utilizatorii care și-au create cont personal cu ajutorul căruia pot beneficia de toate caracteristice magazinului. Pot realiza comenzi online, iar datele vor fi stocate în baza de date ceea ce îl va ușura la urmatoare comanda, acesta având deja datele salvate (adresa de livrare , nume, prenume). De asemenea aceștia își pot expune produsul spre vânzare.

Administratorul este cel care controlează întreaga structură a aplicației și funcionalitățile bazei de date. Acesta poate șterge produsele nepotrivite, poate sa schimbe aspectul site-ului . Mai pe scurt, acesta utilizeaza operațiuni de tip CRUD.

Figura 4 a fost creată pentru a observa clar care sunt acțiunile pe care un vizitator le poate face.



Figură 4 Diagrama Use-Case Vizitator (Sursa: Proiecția autoarei)

*Proprietățile cazurilor de utilizare:*

*Vizualizare produse*: cazul acesta presupune navigarea pe site și obținerea infomațiilor despre oricare produs. Informațiile constau în preț, descriere, imagini și posibilitatea de a adăuga produsul în coș. Toți utilizatorii pot vizualiza produsele.

*Căutare produse*: orice utilizator poate să caute produse doar prin scrierea numelui articolului în bara de căutare. Existe filtre pentru o căutare mai rapidă preț sau nume.

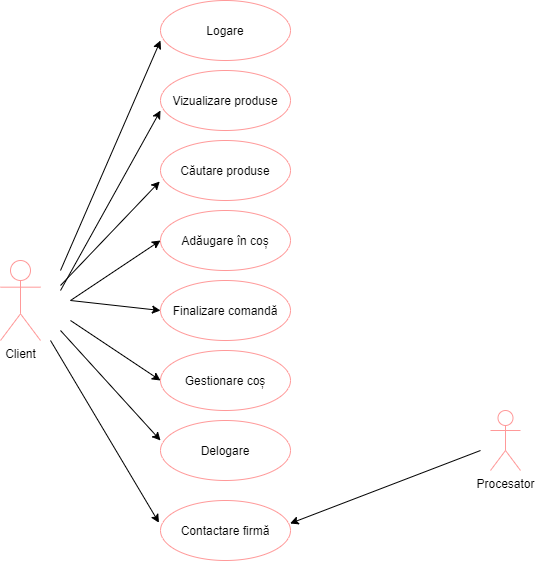
*Creare cont*: acest caz se poate realiza de către orice vizitator care dorește să devina client. Contul se creeaza prin introducerea de nume, prenume, email, username și parola care trebuie sa îndeplinească anumite reguli.

*Autentificare*: aceasta se realizează prin introducerea usernameului si parolei adaugate in pagina de creeare cont.

*Coș de cumpărături*: adăugarea unui produs în coșul de cumpărături se produce prin apăsarea butonului adăugare în coș de sub produs. Coșul de cumpărături poate conține câte produse dorește utilizatorul inclusiv acesta poate să schimbe cantitatea. Acesta are opțiunea de a finaliza comanda sau de a reveni la produse. Coșul își va actualiza suma după fiecare acțiune a utilizatorului. Efectuare comandă: acțiunea este disponibilă doar clienților deja înregistrați în baza de date.

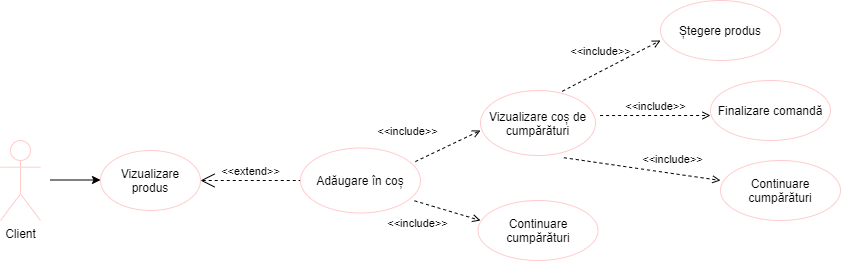
*Istoric comenzi*: acesta pune la dispoziția clientului să își revadă comenzile pe care le-a plasat.

Conform figurii 5, putem observa accesul pe care îl are un client la aplicație, dar și rolul pe care îl are procesatorul. Clienții sunt persoanele care au cont de utilizator completând formularul specific. După crearea contului aceștia pot folosi toate paginile aplicației, iar procesatorul este responsabil în ceea ce privește interacțiunea cu clienții, se ocupă de comenzile atât procesate prin aplicație cât și cele telefonice.



Figură 5 Diagrama Use-Case Client (Sursa:Proiecția autoarei)

Figura 6 reprezintă gestionarea coșului de cumpărături. Acestă pagină poate fi utilizată doar de clienți care au cont de autentificare, iar din reiese din figura de mai jos, aceștia au posibilitatea de a viziona produsele. După ce clientul este sigur pe articolul dorit, acesta îl poate adăuga în coș. Odată ajuns în pagina specifică pentru coșul de cumpărături, aceasta are două opțiuni: continuare cumpărături și finalizare comandă. Imediat după plasare comenzii, acestuia i se va crea automat istoricul și va fi atenționat printr-un email despre starea comenzii lui.



Figură 6 Diagrama Use case - Gestionarea coșului de cumpărături (Sursa: Proiecția autoarei)

Tabel 1 Scenariul principal

|  |  |  |
| --- | --- | --- |
| **NR** | **SECȚIUNE** | **EXPLICAȚIE** |
| 1 | Identificator | Scenariul 1 |
| 2 | Nume | Adaugă un produs nou |
| 3 | Autor | Iscu Paula Iuliana |
| 4 | Versiune | 1 |
| 5 | Istoricul schimbărilor | Fără modificări |
| 6 | Prioritate | Ridicată |
| 7 | Criticitate | Ridicată |
| 8 | Sursă | Analiza cerințelor |
| 9 | Stakholderi | Client |
| 10 | Descriere | După autentificare, utilizatorul are dreptul să își posteze un articol vestimentar pe pagina specifică acestei cerințe. Produsul va fi adăugat dupa apăsarea butonului Add new |
| 11 | Tipul scenariului | Acțiune din partea utilizatorului. |
| 12 | Scopuri | Adăugarea cu ușurință a unui nou produs. |
| 13 | Actori | Utilizatorul cu cont de autentificare |
| 14 | Precondiții | Pentru a putea adăuga un produs nou, utilizatorul trebuie să fie obligatoriu autentificat. |
| 15 | Postcondiții | După ce acesta și-a introdus credențialele, poate să posteze produsul în secțiunea destinată. |
| 16 | Rezultat | Produsul vestimentar devine public, iar restul utilizatorilor pot să-l adauge în coșul de cumpărături. |
| 17 | Pașii scenariului | 1. Utilizatorul se loghează 2. Dacă datele sunt corecte, acesta va fi redirecționat către pagina principală 3. Clientul are posibilitatea de a vedea produsele disponibile și de asemenea poate adăuga unul nou 4. Odată cu apăsarea butonului de adăugare, va fi redirecționat către pagina destinată. 5. După ce este adăugat numele, prețul, o scurtă descriere și o imagina, prin apăsarea butonului add new o să i se deschidă pagina de produse. 6. Sistemul salvează produsul în baza de date. 7. În dreptul produsului apar opțiunile de stergere, editeaza sau adaugă în coș. |
| 18 | Calități | Procesul trebuie să fie unul scurt, iar operațiunile CRUD trebuie să fie disponibile imediat. |
| 19 | Relații cu alte scenarii | Relația dintre scenarii trebuie sa fie eficientă. |

Tabel 2 Scenariul de interacțiune

|  |  |  |
| --- | --- | --- |
| NR | SECȚIUNE | EXPLICAȚIE |
| 1 | Identificator | Scenariul 2 |
| 2 | Nume | Adăugare articol în coșul de cumpărături |
| 3 | Autor | Iscu Paula Iuliana |
| 4 | Versiune | 1 |
| 5 | Schimbări | Fără modificări |
| 6 | Prioritate | Ridicată |
| 7 | Criticitate | Ridicată |
| 8 | Sursă | Analiza cerințelor |
| 9 | Stakeholderi | Client B |
| 10 | Descriere | Utilizatorul alege un articol pe care dorește să îl achiziționeze |
| 11 | Tipul scenariului | Legătura dintre utilizator și baza de date |
| 12 | Scopuri | Adăugarea rapidă în coșul de cumpărături |
| 13 | Actori | Clientul cu cont de autentificare |
| 14 | Precondiții | Pentru a putea adăuga un produs nou, utilizatorul trebuie să fie obligatoriu autentificat. |
| 15 | Postcondiții | După autentificare, utilizatorul poate să adauge în coș produsul droit |
| 16 | Rezultat | Va fi disponibilă o pictogramă ce este reprezentată de coșul de cumpărături pe care va apărea numarul corespunzător articolelor din coș. În coșul de cumpărături vor apărea opțiunile de stergere si modificarea cantității. Coșul va fi actualizat mereu. |
| 17 | Pașii scenariului | 1. Clientul se loghează 2. După logare va fi redirecționat către pagina principală 3. Clientul accesează produsul dorit. 4. În urma apăsării pe articol, se va deschide o pagina specială cu informațiile articolului. 5. Utilizatorul va apăsa pe butonul de adăugare în coș. 6. Sistemul verifica baza de date,iar dacă acesta există, va fi adăugat cu success 7. După acest proces, pe pagina destinată coșului de cumpărături vor fi disponibile acțiunile de continuare cumpărături si finalizare comandă. |
| 18 | Calități | Procesul este unul rapid |
| 19 | Relații cu alte scenarii | Eliminarea produsului din coș, modificarea cantității, finalizarea comenzii și continuare cumpărături. |

### 

### *1.2.4 Documentarea cerințelor*

Pentru realizarea aplicației aferentă acestei lucrări de licență, am găsit anumite cerințe, astfel magazinul să fie cât mai eficient. Aceste sunt clasificate în cerințe funcționale si calitative

1.Cerințele funcționale :

* Aplicația permite clientului să vizualizeze toate produsele, după ce și-a creat cont
* Utilizatorul va avea parte de diversitate
* Aplicația web conține și un coș de cumpărături ceea ce îi oferă posibilitatea clientului de a comanda produsul dorit.
* De asemenea, clientul poate să își adauge propriul său articol destinate vânzării.
* Administratorul va gestiona aplicația având access și la baza de date.

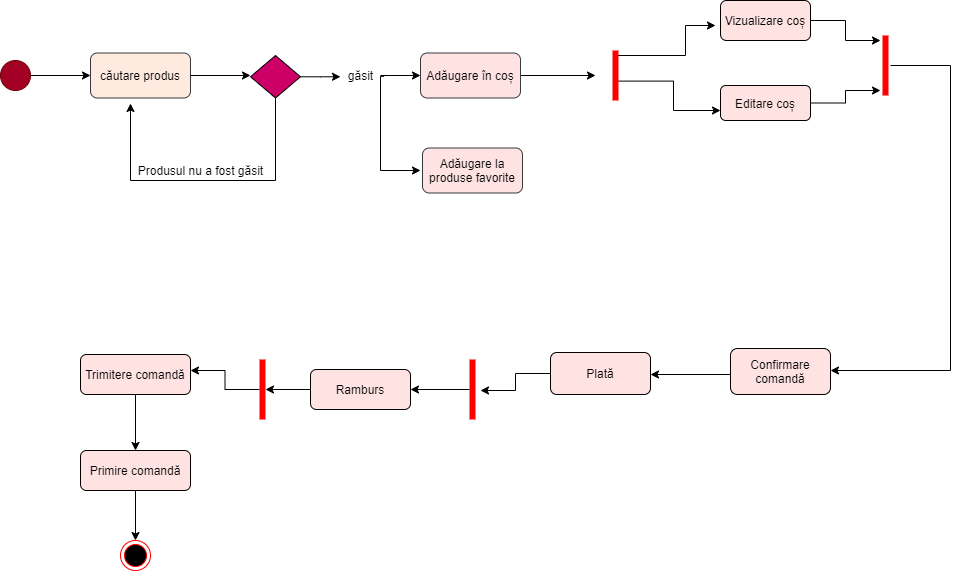
2. Cerițele calitative:

* Aplicația trebuie să fie rapidă pentru a permite conectarea mai multor utilizatori simultan.
* Aplicația va fi ușor de folosit atât pentru utilizatori, cât și pentru vizitatori.
* Aceasta trebuie să conțină un sistem de Securitate unde parolele să fie criptate.
* Arhitectura sistemului trebuie sa permită tuturor operațiunilor să se desfășoare în timp real.

### *1.2.5 Procese și activități*

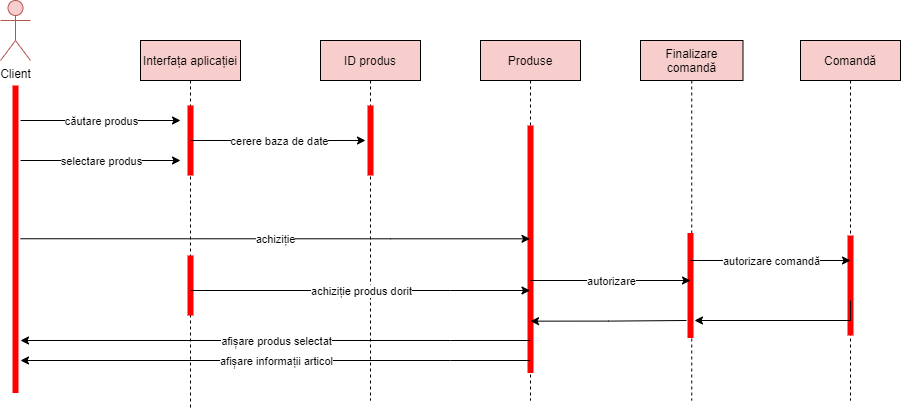
Figura 7 exemplifică activitatea firmei care este reprezentată de vânzarea hainelor la mâna a doua. Aplicația cuprinde toate cerințele unui magazin online pentru a satisfice utilizatorul. Procesul de achiziții online trebuie să cuprindă procese tehnologice care vor îmbunătăți vânzarea. Clientul accesează magaziul și își îndreaptă atenția către un anumit produs, pe care ulterior îl poate adăuga în coșul său, dar îi va putea lăsa și o recenzie.

Clientul introduce datele de autentificare și accesează magazinul. Dacă produsul căutat este găsit, atunci utilizatorul are posibilitatea de a pune articolul în coș. Coșul se poate vizualiza sau edita în funcție de opțiunile cumpărătorului. Așa cum reiese din diagrama de activități, comanda poate fi plasată pentru a merge mai departe în procesul de cumpărare sau are posibilitatea de a se întoarce pe pagina cu produse. După ce este bifata modalitatea de plată, se efectuează trimiterea comenzii. Așadar, pașii principali în procesul de cumpărare este realizat prin diagrama de activități.

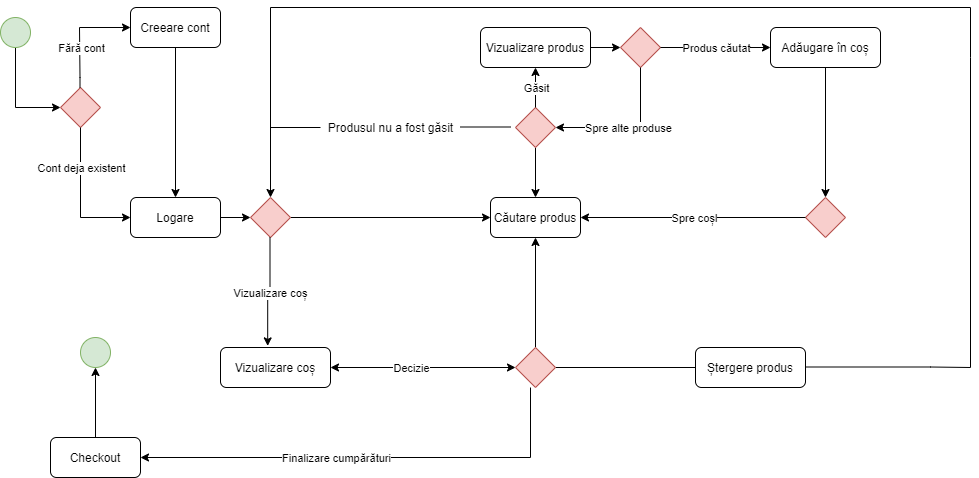


Figură 7 Diagrama de activități (Sursa: Proiecția autoarei)

Figura 8 scoate în evidență interacțiunile clienților cu partea de back-end. În momentul în care un client caută un produs, în baza de date este căutat după ID și este listat. Dacă utilizatorul dorește să achiziționeze un produs, acestuia i se vor afișa imaginea cu produsul și detaliile necesare, așa cum reiese din figura 8. Când se ajunge la pasul de finalizare comandă, acesta este nevoit să introducă datele pentru a putea fi notificat cu privire la comanda sa.



Figură 8 Diagrama de secvență (Sursa: Proiecția autoarei)

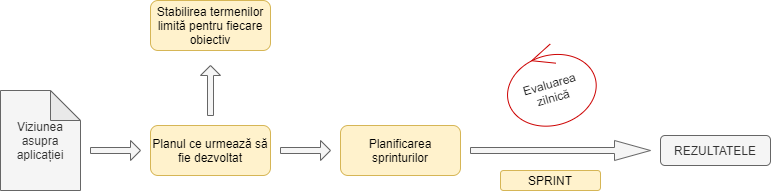
Figura 9 reprezintă acțiunile pe care sistemul le realizează în funcție de opțiunile clientului. La fiecare pas, partea de back-end primește o nouă formă. Orice utilizator are dreptul de a își crea cont, iar mai departe acesta va avea acces la pagina cu produse.

Figură 9 Diagrama de stare (Sursa: Proiecția autoarei)

## 1.3 Modelul de dezvoltare

Modelul de dezvolatare ales pentru aplicație de tip e-commerce “CCStore” este modelul Agile. Acest model reprezintă un cadru conceptual în dezvoltarea proiectelor de tip software. În dezvoltarea unui proiect pot apărea factori de risc care pot împiedica finalizarea proiectului. Factorii cel mai des întâlniți sunt: factorii de planificare, tehnologici, externi și cei de experiență.

Modelele de tipul Agile minimează riscurile prin dezvoltarea softului în intervale scurte care se numesc “timeboxes”. Principiul care stă la baza acestui model se bazează pe divizarea proiectului (planificare, analiza cerințelor, codificare, testare, documentare). Dintre toate metodele puse la dispoziție de Agile am ales să folosesc Scrum, deoarece pune accent pe organizare.



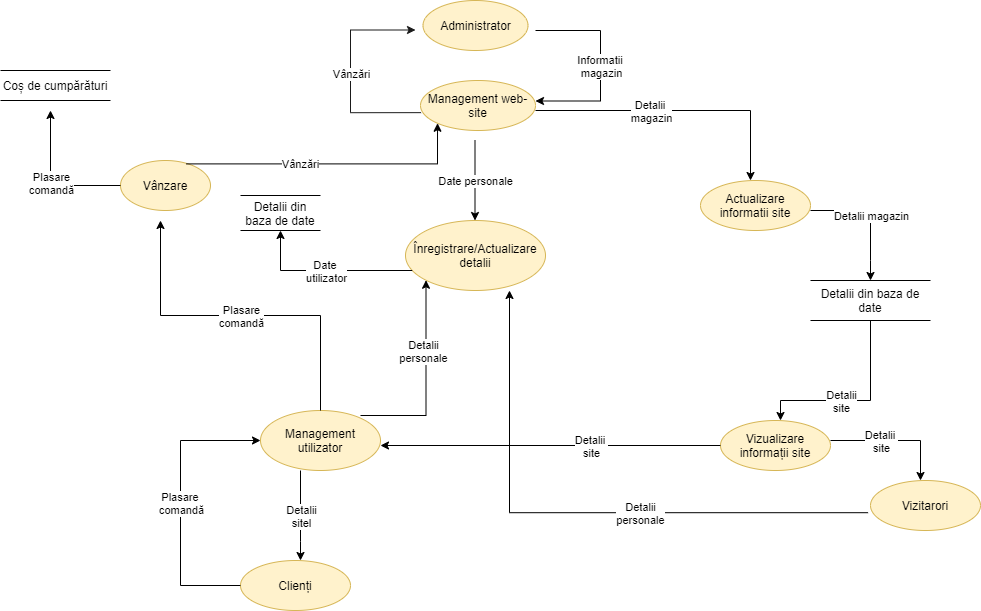
Figură 10 Diagrama Agile-Scrum (Sursa: Proiecția autoarei)

Scrum este un cadru care utilizează mentalitatea Agile pentru dezvoltarea, livrarea si susținerea produselor complexe. Scrum este iterativ și incremental pentru gestionarea muncii. Este conceput pentru o împărțire clară a obiectivelor ce trebuie îndeplinite, numite sprinturi care trebuie duse la capăt în timp scurt. Prin metoda scrum proiectul se evalueză zilnic, iar la finalul fiecărui sprint se reevaluează munca depusă pentru feedback (Cohen, (2004)).

# **Proiectarea sistemului informatic**

## 2.1 Proiectarea logică

Ținând cont de faptul ca scopul aplicației este acela de a realiza vânzare de produse prin intermediul mediului online, realizarea acestui magazin va constitui o rezolvarea a problemelor privind deplasarea și timpul.

Figura 11 este utilizată pentru a pune în evidență cum sunt procesate și transferate datele. Cu ajutorul acestei diagrame putem identifica sursa datelor și modul în care acestea interacționează. Diagrama fluxului de date prezentată mai jos scoate în evidentă interacțiunile dintre utilizatorii magazinului, vizitatori și administrator.

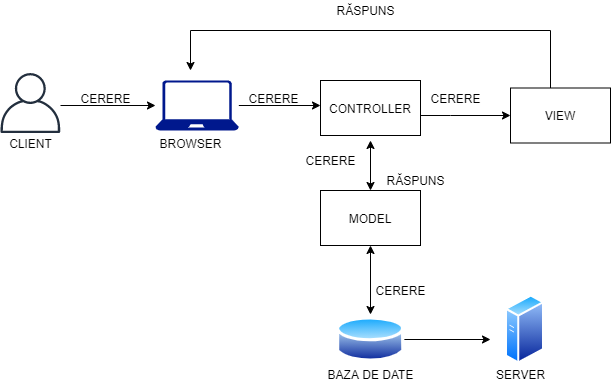
Figură 11 Diagrama Fluxului de Date (Sursa: Proiecția autoarei)

### *2.1.1 Arhitectura sistemului*

Arhitectura client-server este o arhitectură ce realizeaza atât funcțiile de care depinde clientul, însă și cele de care depinde serverul, aceasta oferind posibilitatea pentru schimbul de informații. Această arhitectură reprezintă o comunicare între server și client, fiind formată din server de aplicații, server de baza de date și calculator. În aplicația mea toate datele pe care utilizatorul le folosește sunt stocate în *phpMyAdmin,* iar de acolo sunt folosite de către server pentru a oferi răspunsuri. În același timp, această arhitectură separă structura de detaliile interne ale subsistemelor individuale.

Conceput de client-server împarte executarea unei unități în activitățile clientului și răspunsurile serverului. În aplicația mea clienții sunt utilizatorii ce dețin un cont de autentifcare. Cerințele lor sunt redate imediat de server, iar acest lucru se realizează printr-o stație de lucru ce execută părți ale aplicației. Figura 12 reprezintă legătura dintre browser și baza de date, mai exact de client și server.

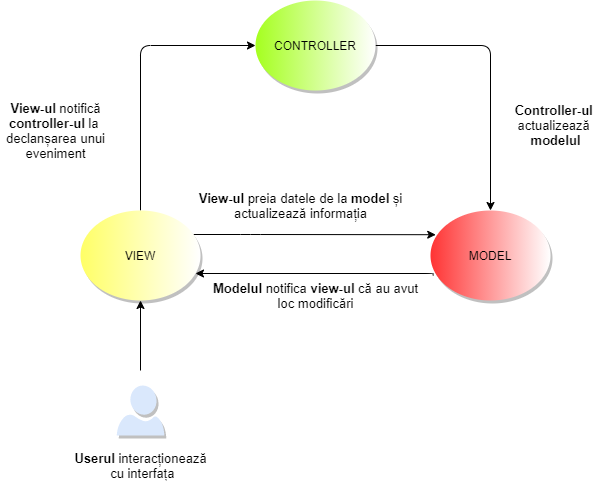
În momentul în care un client dorește să viziteze o anumită pagina, prin apăsarea butonului specific primește răspuns de la controller si este redirecționat acolo unde își dorește. Dacă de exemplu un utilizator adaugă un produs în coș, controller-ul acționează și atenționează modelul pentru a face posibil acest lucru, prelunând din baza de date informația necesară. Mai departe modelul apelează view-ul pentru a oferi toate datele.

 Fluxul de date din figura 12 este următorul: clientul trimite o cerere către browser, care mai departe apelează controller-ul, urmând apelarea view-ului, acesta trimițând mesaj către browser. Controllerul atenționează modelul că trebuie să încarce un element nou în baza de date, iar aceasta mai departe atenționeaza serverul (Hanson, (2000)) .

Figură 12 Arhitectura Client-Server (Sursa: Proiecția autoarei)

Pentru realizarea unui magazin online este nevoie de o arhitectura bine stabilită și o bună organizare. Principalele aspecte care vor fi luate în considerare sunt: organizarea bazei de date, interfața aplicației, limbajele utilizare, complexitatea magazinului. Trebuie să avem în vedere și funcționalitățile expuse ca de exemplu: login, plasare comandă, istoric comenzi. În ceea ce privește proiectarea tehnică, un magazin online trebuie să îndeplinească unele calități standard: aplicația să fie prietenoasă, login și securitatea aplicației, baza de date să fie bine definită, plasare de comandă online.

Pentru realizara aplicației am folosit modelul MVC. Modelul MVC permite separarea clară a componentelor pentru a putea fi schimbate mai ușor fără a fi modificate restul modulelor ce ajută la formarea site-ului. Așa cum este indicat și din nume, aplicația este împărțită în trei componente de baza: *model* care stochează datele, *view* se ocupă cu interfața și c*ontroller* care prelucrează datele. Figura 12 scoate în evidență faptul că arhitectura MVC permite ca o aplicație să fie realizată cu ușurință, codul poate fi utilizat de mai multe ori și separă logica aplicației față de interfată. În cele ce urmează am realizat diagrama MVC și descrierea fiercărui component:



Figură 13 Paradigma Model-View-Controller (Sursa: Proiecția autoarei)

**Modelul** reprezintă partea care păstrează informațiile, iar acesta trebuie creat la începutul implementării. Acest component răspunde comenzilor pe care le primește de la controller și păstrează legătura cu baza de date. Modelul este utilizat mereu împreună cu bază de date unde să păstreze toate datele. Pentru ca informațiile din baza de date să fie accesate, a fost realizat câte un repository pentru fiecare model. Modelul este independent ceea ce înseamnă că poate fi pus în funcțiune fără să existe o aplicație grafică. Datele sunt stocate sub formă de proprietăți și pot fi modificate oricând.

Tot în partea de model se pot valida câmpurile aflate în baza de date și se pot crea metode de schimbare și accesare a informațiilor. În momentul în care este apelată o funcție , view-ul trimite o cerere către model, iar acesta trebuie să poată descifra pentru a trimite validare.

**Controller-ul** reprezintă componenta principală a arhitecturii MVC. Acesta se comportă precum o asociere între Model și View și realizeaza logica din spate. Folosind clasa Model acesta poate manipula informațiile, iar interacționarea este realizată cu clasa View.

În momentul în care clienții intră în contact cu interfața, controller-ul este notificat de acțiunea pe care o va declanșa. Atunci când utilizatorul dorește să facă modificări, model-ul este anunțat automat, ceea ce este posibil doar datorită legăturii cu baza de date. Controller-ul poate procesa cererea HTTP și de a genera o pagina web.

**View-ul** este folosit pentru ca modelul în forma pentru interacțiunea cu clientul. Acesta este utilizat împreună cu Codeigniter . O vizualizare de cele mai multe ori este o pagină web, însă cu ajutorul acestui framework, o vizualizare poate fi de asemenea, un parte din pagină, cum ar fi un antet sau un subsol. Poate fi și o pagina RSS. (Deacon, (2009))

### *2.1.2 Baza informațională*

Baza informațională a aplicației conține următoarele entități grupate astfel și cu urmatoarele date: administrarea aplicației va cuprinde date prin interemediul cărora administratorul se va putea autentifica și astfel va avea acces la datele necesare, partea ce va conține categoriile de produse încadrate spre vânzare, partea de produse care va conține produsele puse spre vânzare împreuna cu partea unde se vor introduce reviewurile specifice și imagini cu fiecare produs, partea de useri care se vor putea loga pe baza unui username și a unei parole și partea de comezi care va prezenta detalii despre comenzile existente pe site, coșul de cumpărături și modalitatea de plată. De asemenea, utilizatorii vor avea posibilitatea de a selecta între limba română și engleză.

## 2.2 Proiectarea tehnică

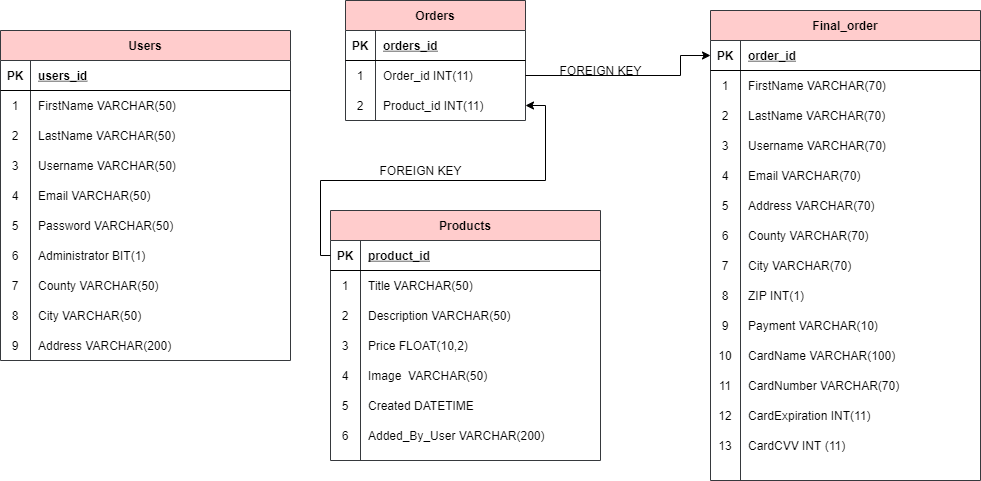
Interfața grafică GUI reprezintă calea de comunica dintre sistemul informatic și utilizator. Acesta reprezintă și modul de funcționare, nu doar partea de design, fiind o balanță între cerințele utilizatorului și ale aplicației. Această interfață grafică pune la dispoziție legătura a utilizatorului cu sistemul informatic prin introducerea de cereri de date și răspunsuri de la sistem. Principalul rol al interfeței este de oferi funcționalitățile prin care se pot introduce date, se afișează informațiilre din baza de date cu ajutorul paginilor web.

Aplicația pune la dispoziție pentru interfața destinată clienților informațiile prezente pe site: Home, Produse, Autentificare pentru cei neautentificați, iar pentru cei care au cont vor apărea în plus butoanele pentru Coș, Istoric comenzi.

Partea de administrator conține următoarele funcționalități: ștergerea și editarea utilizatorilor, produselor și comenzilor. Order pentru confirmarea comenzilor și details pentru vizualizarea specificațiilor introduse de utilizator pentru fiecare produs. În cazul anumitor pagini, se va afișa conținutul în mod spontan cu informații ce se află în baza de date. De exemplu prin adăugarea unui nou produs acesta va fi vizibil și în interfața grafică.

Sistemul informatic va cuprinde doua module, unul pentru administrator și unul pentru utilizator. Elementele interfeței cu clientul sunt butoanele submit, casetele de tip text, input-uri, label-uri. Proiectarea interfeței pune accent pe ceea ce poate face clientul, mecanism de naviagre, manipularea eficientă a evenimentelor de adăugare în coșul de utilizator, de adăugare a unui produs ce dorește să îl vândă, iar pentru partea de administrator prezența meniului pentru gestionarea datelor și a formularelor pentru manipularea datelor (Tiwari, (2016)).

### *Structura fizica a datelor*

Aplicația mea conține 4 tabele relaționale reprezentate de figura 13 după cum urmează: user, products, orders și final\_order. Aici vor fi introduse toate datele utilizatorilor, dar și comenzile.

Figură 14 Diagrama bazei de date (Sursa:Proiecția autoarei)

Într-un sistem informatic, baza de date reprezintă partea cea mai importantă în procesul de realizare a aplicației. Reprezintă un mod eficient de organizare a informațiilor stocate în urma procesării aplicației, informații ce au o logică și pot fi interogate ușor. Baza de date reprezintă totalitatea informațiilor ce se găsesc pe conținutul paginilor web.

În cadrul bazei de date există informații precum: datele utilizatorilor, produsele existente, coșul de cumpărături și verificări pentru logare. O bază de date ilustrează dateleaplicației web, aceastea fiind stocate prin MySql ce rulează pe serverul Apache în cadru XamppServer.

### *2.2.2 Tehnologii specifice*

Pentru a dezvolta aplicația, am utilizat limbaje de programare precum: PHP, MySql, JavaScript, HTML și CSS. De asemenea, am folosit serverul Xampp ca și componentă locală server. Iar ca și framework am utilizat CodeIgniter.

Am decis să dezvolt aplicația cu ajutorul limbajului PHP datorită flexibilității și numărului mare de tutoriale și a librăriilor de cod scrise și puse la dispoziție în mod gratuit. PHP este un limbaj de script pe partea de server conceput pentru realizarea aplicațiilor web. Codul PHP este interpretat de serverul web și generează HTML sau al produs ce trebuie finalizat.

CodeIgniter este un cadru de aplicații web open source pentru limbajul PHP. Spre deosebire de alte alte cadre PHP, în ceea ce privește CodeIgniter documentația este completă, acoperind fiecare aspect. CodeIgniter va rula, de asemenea, în medii de găzduire partajată, deoarece are performanțe excepționale. Acesta este compatibil atât cu PHP 4 cât și cu PHP 5, deci va rula pe majoritatea serverelor web.

CodeIgniter folosește ca model de proiectare MVC, care este o modalitate de a organiza aplicația în trei părți separate:

* clasa model - este stratul de abstractizare a bazei de date
* clasa views - conține fișierele șablon front-end
* clasa controller - oferă logica aplicației.

CodeIgniter folosește și modelul de design Singleton. Acesta fiind un mod de a încărca clasele, astfel încât dacă acestea sunt apelate de mai multe ori vor fi returnate. Acest lucru este util și pentru conexiunile la baza de date, deoarece este nevoie de o singura conexiune indiferent de cât de des este apelată clasa (Griffiths, 2010).

Am ales acest framework pentru realizarea aplicației mele, deoarece acesta mi-a oferit avantaje, cum ar fi:

* mărimea framework-ului pentru spațiu de stocare este mică
* asigură compatibilitate cu mai multe versiuni de PHP
* are o documentație clară și completă
* este un framework care nu necesită multă configurare
* rulează pe majoritatea host-urilor web.

În aplicația mea, controller-ul conține clasele: *add\_item* ce este utilizată în crearea logică pentru adăugarea unui articol vestimar, dar doar pentru cei care au cont de utilizator, *administrator* unde sunt stocate datele pentru a putea modifica, șterge, adăuga articole sau utilizator, *cos* această clasă a fost creată pentru a putea adăuga produsele în coșul de cumpărături, dar am utilizat o funcție ce permite doar utilizatorilor să îndeplinească sarcina, *delogare* unde este creată funcția prin care utilizatorul poate părăsi aplicația fără să rămână logat, *finalizare* clasa a fost creată pentru a putea realiza formularul de finalizare, *home* clasa pentru încărcarea view-ului specific și de a crea funcțiile pentru internaționalizare, *istoric* unde care cere încărcarea view-ul specific,*register* aici sunt stocate funcțiile pentru ca utilizatorul să își poată crea cont cu care mai apoi va naviga în aplicație, *login* este concepută pentru ca utilizatorul să își poată introduce datele inserate în pagina de register, *products* aici sunt create funțiile de adăugare și de ștergere produs.

Model-ul conține clasele: *products\_model* ce face posibila legătura cu baza de date pentru stocarea informațiilor legate de produsele adaugate și clasa *register\_model* ce păstrează în baza de date informațiile despre utilizatori, *email\_model* care stochează informațiile în baza de date pentru a se face posibilă trimiterea emailului de confirmare, *finalizare\_model* care salvează în baza de date informațiile introduse de utilizator, *grocery\_crud\_model* clasă ce face posibilă crearea paginii de administrator și care salvează informațiile în baza de date, *istoric\_model* care face posibilă salvarea comenzilor anterioare și ­*users\_model* care caută username-ul introdus pentru a putea transmite mesajul.

Views conține clasele: *header* unde am creat nav-bar-ul pentru aplicația mea, iar acesta conține butoanele Home, Coș, Produse, Autentificare, RO(pentru limba română) și EN(pentru limba engleză), add\_item, administrator, cos, finalizare, home, products, loginform, registerform si istoric unde am aranjat in pagina pentru interacțiunea cu utilizatorul.

Pentru crearea funcției de administrator am utilizat librăria GroceryCRUD care este un generator de rețea CodeIgniter (Rodriguez-Echeverria, (2016)). Această librărie poate genera automat operațiile CRUD, deoarece are mult caracteristici, cum ar fi :

* creare automata a intrărilor după tipul câmpurilor
* relația de baza de date doar cu o singură linie de cod
* formular de validare pentru client și validare din partea serverului
* editare, ștergere, anulare
* schimbarea cu ușurință a unui tip de date pentru un câmp
* funcționalitate multilingivistică
* funcționează cu toate browserele și cu toate sistemele de operare.

Aspectul și interactivitatea aplicației au fost conturate cu ajutorul librăriei Bootstrap, HTML, CSS și JavaScript. Bootstrap ușor de înțeles și are capacitatea de a crea o interfață aerisită și interactivă. De asemenea, acesta poate fi utilizat pe orice tip de browser. Bootsrap este unul dintre cele mai cunoscute cadre HTML, CSS și JavaScript pentru a implementa o aplicație web. El are un impact fundamental asupra modului în care se realizeză interfața.

Boostrap nu funcționează izolat, ci mai degrabă aplicat la o mare varietate de proceduri statistice. Algoritmul bootstrap funcționează prin extragerea a mai multor eșantioane independente, replicații ale bootstrap-ului corespunzătoare și estimarea erorii standard prin derivația empirică a replicațiilor (Boos, May 2003).

Pentru a păstra totul în siguranță am folosit GitHub, fiind cea mai mare gazdă de cod. De asemena, GitHub este un flagship pentru dezvoltarea curentă a surselor deschise, un loc pentru dezvoltatori și o platformă pentru cod. Acesta oferă controlul versiunii distribuite și gestionarea codului sursă, adăugând propriile caracteristici.

Am utilizat GitHub, deoarece nu limitează numărul proiectelor ce pot fi încărcate pe platformă, iar faptul că poate fi utilizat exclusive online, doar cu ajutorul unui cont, face navigarea simplă, iar datele vor fi în maximă siguranță. Pentru a putea trimite mai departe proiectul trebuie doar copiat link-ul oferit de GitHub și trimis. Site-ul păstreză proiectele în funcție de ceea ce dorește utilizatorul, private sau publice. Cele private pot fi vizibile doar de către persoana ce a încărcat proiectul, iar cele publice pot fi vizualizate de oricine. Lucrările încărcate pe GitHub conțin date voluminoase, informații socio-lingvistice, cât și informații despre stare și activitate.

Cu ajutorul aplicației SourceTree orice modificare ulterioară adusă proiectului a putut fi adăugată pe GitHub foarte simplu, doar folosind butonul de update ce este pus la dispoziție de platformă. Această aplicație simplifică modul de interacțiune cu depozitele aflate pe GitHub, având posibilitatea de a vedea exact ce modificări s-au adus și daca suntem de acord cu ele. Modificări ce nu se doresc a fi încărcate, pot fi sterșe chiar din aplicație, astfel pe GitHub să fie încărcat exact ceea ce dorim (De Jonge, 2002)gui.

De asemenea, face sistemele software inflexibile deoarece intregrarea componentelor suplimentare ale codului sursă în arbori și procesele de construcție este dificilă. Obiectul lucrării este de a spori reutilizarea codului.

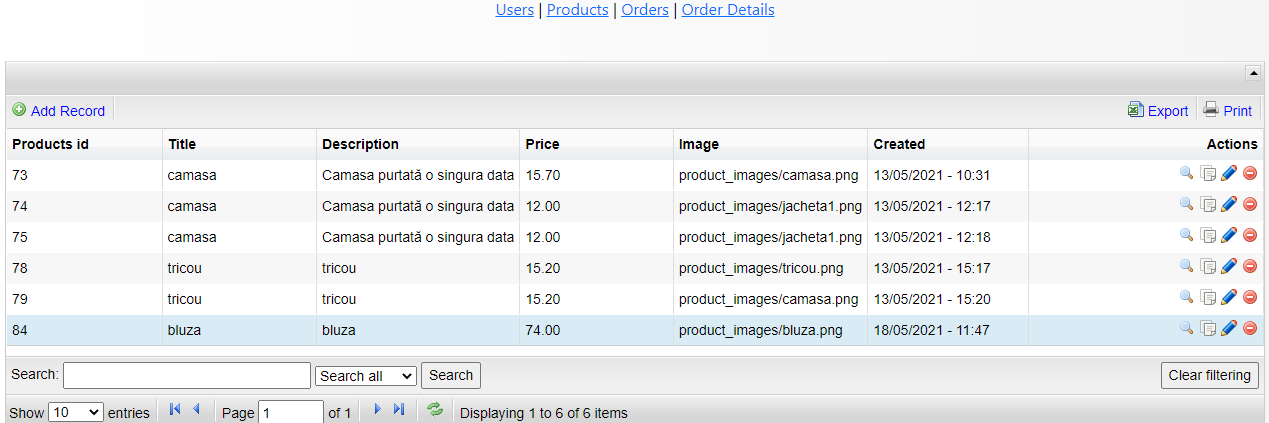
Proiectul meu de licență este disponibil pe link-ul ce va fi trimis de mine doar pesoanelor dorite: <https://github.com/IscuPaula/Licenta.git>. Tot aici am încărcat și toate modificările aduse documentației (Kalliamvakou, (2016) ).

# **Implementarea sistemului informatic**

Implementarea sistemului informatic începe cu mult înaintea dezvoltării software-ului. Această etapă este un proces în care se află beneficiile sistemului. Pentru crearea aplicației mele am ținut cont de ceea ce aș dori eu să văd atunci când accesez un site. M-am axat pe un front-end prietenos, ușor de utilizat și pe înțelesul tuturor. De asemenea, am ales culori cât mai simple, alegând nuanțe de alb și gri. Pentru background am ales să folosesc un gradient care pornește de la alb și merge către un alb afumat (white smoke).

Am creat și un logo pentru aplicația mea pe care l-am adăugat în majoritatea paginilor. Am ales inițialele C C care reies din cuvintele Cheap(ieftin) și Clothes(haine), mai pe scurt CheapClothes(HaineIeftine) care aș dori să scoată în evidență motivul principal pentru care a fost creată aplicația și anume vinderea hainelor pe care oamenii nu le mai folosesc.

Am creat navbar-uri pentru pagini penru a putea adăuga butoanele de navigare a aplicației.Am adăugat modificări ulterioare și toate au fost încărcate pe GitHub.

Folosind librăria GroceryCRUD am creat cu ușurință pagina de administrator.Aici sunt disponibile funcționalități precum ștergerea, editare, adăugarea, dar și vizionarea atât a produselor cât și a utilizatorilor. Aceasta este open source, este ușor de utilizat și foarte rapidă.

Figură 15 Administrator (Sursa: Proiecția autoarei)

Cei care accesează platforma trebuie să își creeze un cont pentru a putea avea acces la cât mai multe pagini. Există posibilitatea de a naviga prin site însă doar pe pagina unde sunt produsele. Pentru fiecare câmp am creat validări. Toate câmpurile sunt obligatorii, numele de utilizator, emailul și parola au câteva în plus. Cum ar fi: număr minim și maxim de caractere, utilizarea unui simbol, a unei cifre.

Câmpul pentru introducerea adresei de email sunt obligați să folosească unul valid și de asemenea utilizarea simbolului @ este obligatarie. Numele de utilizator și parola disponibile pe pagina de login trebuie să coincidă cu cele introduse la înregistrare. Toate informațiile vor fi stocate în baza de date prin ajutorul modelului creat pentru acest pas. Cu ajutorul metodei GET informațiile vor fi preluate datele introduse în câmpurile specile, iar funcția trebuie apelată în controller.

function insert\_users() {

$fname = $this->input->post('fname');

$lname = $this->input->post('lname');

$uname = $this->input->post('uname');

$email = $this->input->post('email');

$password = $this->input->post('password'); $data=array

( 'FirstName'=> $fname,

'LastName'=> $lname,

'Username'=> $uname,

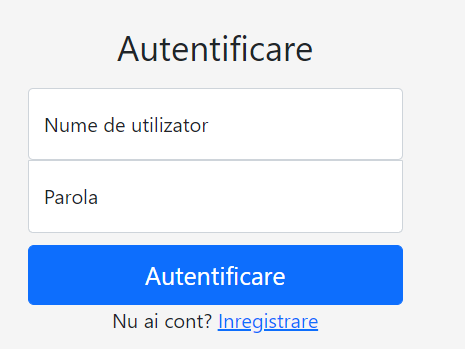
'Email' => $email,

'Password' => $password);

$this->db->insert('users',$data);}

Listare 1 Funția de inserarea utilizatorilor

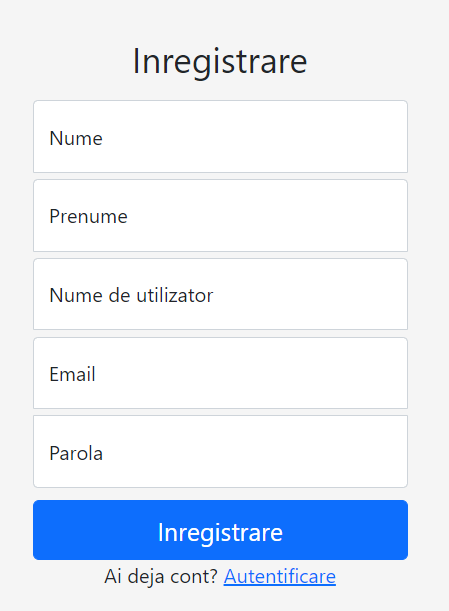
Acest cod reprezint de figura 15 exemplifică cum a fost creată legătura dintre aplicație și baza de date pentru a putea fi înregistrate toate datele. Din baza de date, informațiile vor fi preluate prin apelarea funcției *inser\_users* în controller.



Figură 17 Autentificarea clientului (Sursa: Proiecția autoarei)

Figura 16 reprezintă câmpurile pe care le vede un utilizator în momentul autentificării. Numele de utilizator trebuie să conțină minimum 5 caractere, iar parola trebuie aiba în creare sa tot minimum 5 caractere, însă utilizarea unei litere mari, unui număr și a unui simbol este obligatorie. Prin apăsarea butonului *Autentificare* se va deschide prima pagina, mai exact *Home* de unde utilizator va putea naviga mai departe.

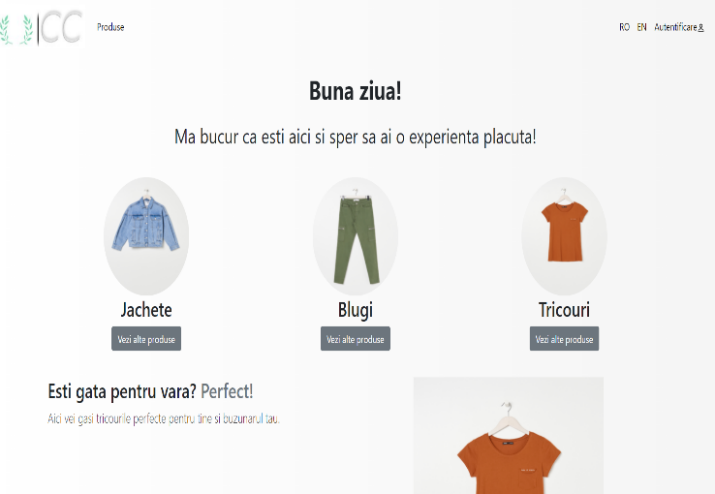
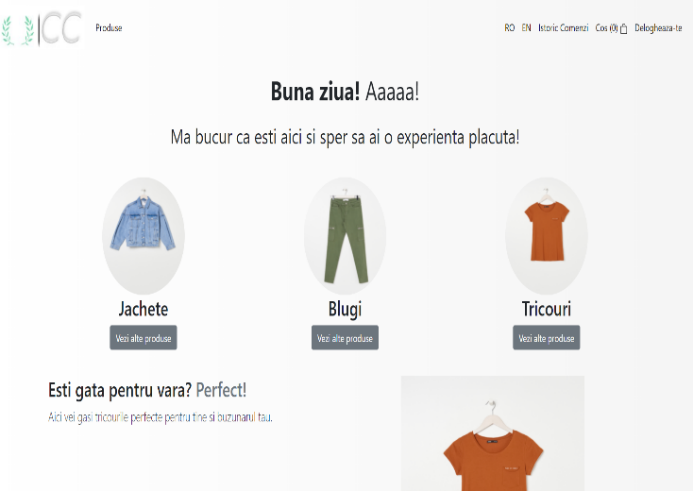
Dacă utilizator însă nu are cont, a fost creat un link pentru redirecționarea către pagina de Înregistrare. Atât username-ul cât și parola trebuie să coincidă cu cele din baza de date, mai exact cu cele care au fost introduse în momentul înregistrării, iar acest lucru este scos în evidență de figura 17.

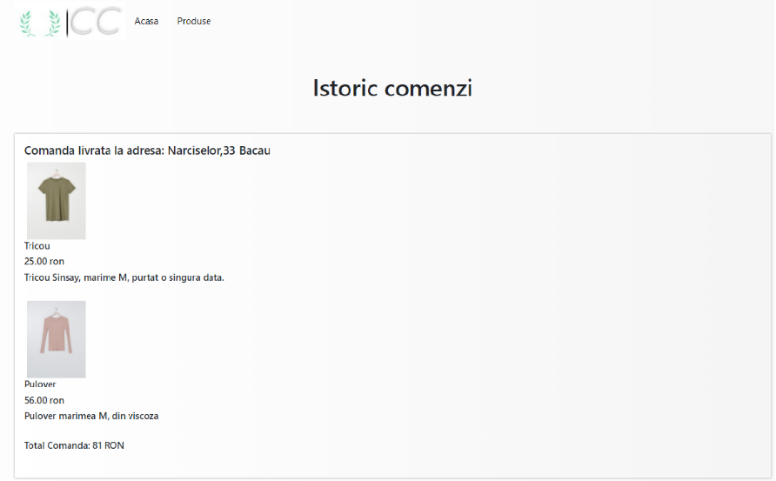


Figură 18 Înregistrarea vizitatorului (Sursa: Proiecția autoarei)

Pentru început voi prezenta pagina de home, atât pentru utilizatori cât și pentru cei ce nu au cont. În funcție de ora la care se accesează aplicația mesajul de întâmpinare se schimbă. Până în ora 12 va apărea *Bună dimineața*, de la ora 12 până la ora 17 va fi afișat *Bună ziua*, iar de la ora 17 va fi disponibil *Bună seara,* așa cum arată figura 18, iar cest lucru a fost posibil folosind cod de JavaScript.

Lângă mesajul de întâmpinare va fi afișat numele utilizatorul dacă acesta are un cont disponibil, iar această funcționalitate a fost creată cu ajutorul sesiunilor. Linia de cod specifică este <?=$\_SSESION[”username”]?>.

Figură 19 Pagina principală (Sursa:Proiecția autoarei)

Legat de funcționalitatea de delogare, în momentul apăsării pe buton utilizatorul va fi redirecționat către pagina *Home* însă nu va mai fi afișat username-ul său. Tot pe această pagina utilizatorul are la dispoziție și istoricul comenzilor sale. În momentul finalizării comenzii, clientul va fi automat redirecționat aceasta.

Figură 20 Istoric (Sursa: Proiecția autoarei)

Pe pagina principală numită *Home* pentru cei ce au cont sunt prezente butoanele de coș și delogare, dar și butoanele pentru internaționalizare (RO, EN). În funcție de preferințele utilizatorului, aplicația este disponibilă în română și engleză, iar pentru acest lucru este necesară scrierea următoarei linii de cod în fiecare clasă din view: <?php $this->load->helper('traducere')?>. Internaționalizarea a fost creată manual, adică fiecare rând pe care am dorit să îl traduc, cu ajutorul funcției <?php echo traducere()?>,am scris textul atât în română cât și în engleză.

În fișierul *helpers* pus la dispoziție de CodeIgniter am creat funcția pentru internaționalizarea aplicației. Variabila isset este diferită de null și toti parametrii vor fi setați. Citirea se va face de la stanga la dreapta, indiferent de ce variabilă este întâlnită.

<?php

function traducere($romana, $engleza){

if(isset($\_SESSION["limba"])){

if ($\_SESSION["limba"] == "romana"){

return $romana;

}elseif ($\_SESSION["limba"] == "engleza") {

return $engleza;

}else{

return $romana;}

}else{

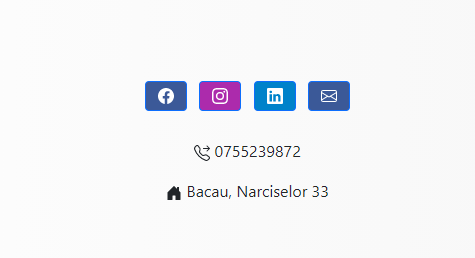
$\_SESSION["limba"] = "romana";}}?>

Listare 2 Funcția de traducere

Utilizatorii au acces atât la pagina pe care sunt prezentate produsele, însă au și dreptul de a le adăuga în coș. Pentru cei ce nu au cont de utilizare le sunt disponibile funcționalitățile de schimbarea a limbii și de autentificare. Aceștia au posibilitatea de a vedea produsele, însă în momentul apăsării butonului *Adaugă în coș* vor fi redirecționați către pagina de autentificare.

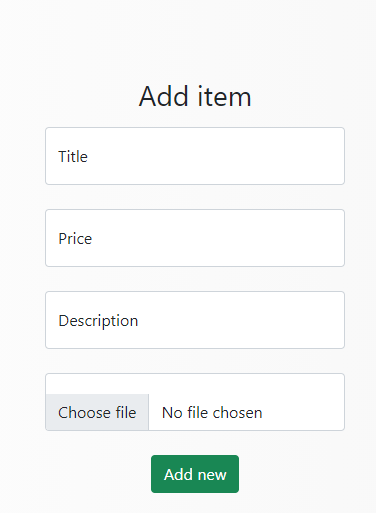
Butoanele prezentate au fost create cu ajutorul limbajului folosind atributul *class* ce este utilizat pentru a indica numele de clasă, iar mai departe acesta va fi manipulat utilizând JavaScript.

Pe pagina principală *Home* am creat și un footer pentru a putea adăuga câteva detalii despre mine. Butoanele prezentate mai jos sunt funcționale, iar în funcție de care va fi accesat, se va deschide o pagină specifică după cum urmează: facebook, instragram, LinkedIn și email. Pentru a putea crea această funcționalitate m-am folosit de atributul href ce specifică adresa URL relativă unei pagini. Am atașat figura 21 penstru a evidenția acest lucru.



Figură 22 Subsolul paginii Home a aplicației (Sursa: Proiecția autoare)

Figura 22 arată că în momentul apăsării pe butonul *Produse* atât utilizatorul, cât și vizitatorul vor fi redirecționati către pagina unde se pot vedea produsele disponibile pe site. Butonul intitulat *Adaugă produs* după cum se poate observa în figura 22 șipoate fi accesat doar de utilizatori, iar aceștia au posibilitatea de a își vinde propriul lor articol.



Figură 23 Adăugare produs nou de către utilizator (Sursa: Proiecția autoarei)

Câmpurile au validări care obligă utilizatorul să completeze fiecare label. Folosind modelul *Products\_model.php­* toate informațiile vor fi stocate în baza de date, iar cu ajutorul funcției *get* vor putea fi văzute în pagina specială pentru produse. După ce produsul va fi adăugat de către utilizator, acesta va fi redirecționat către pagina unde sunt prezentate produsele, doar dacă toate validările sunt îndeplinite. Dacă utilizatorul nu este conectat, folosind funcția *redirect,* acesta va fi trimis către pagina unde este nevoit să își introducă credențialele.

După ce articolul a fost adăugat, pagina va fi reîncărcată și produsul va fi afișat. Sub imagine vor fi afișate: prețul, descrierea, numele persoanei care și-a încărcat produsul și nelipsitul buton de a adăuga în coș așa cum reiese din figura 24. Pentru a crea această funcționalitate am utilizat sintaxa foreach. Aceasta repetă toate elementele dintr-o matrice. În momentul în care PHP găseste sintaxa foreach, acesta atribuie primul element al tabloului variabilei care urmează. Când s-a ajuns la ultimul element, bucla se termină.

<?php foreach($products as $product){?>

<div class="card" style="width:14rem">

<img class="card-img-top" src="<?php echo base\_url($product->image);?> "/>

<div class="card-body">

<h5 class="card-title"><?=$product->title?></h5>

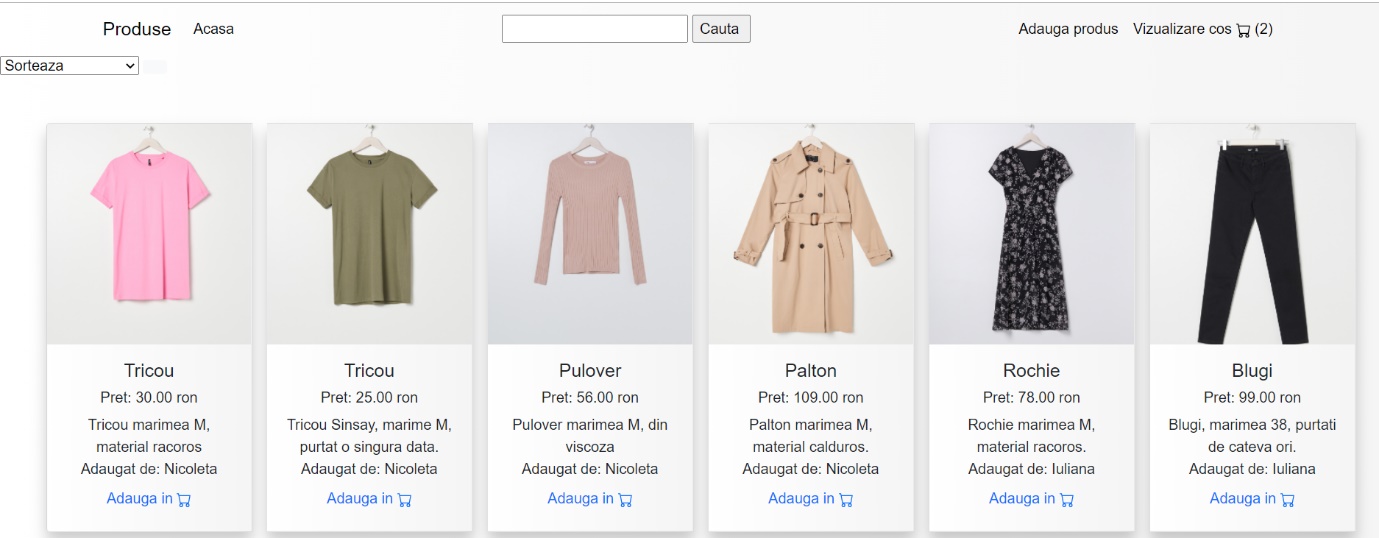
<h6 class="card-text"> <?php echo traducere('Pret:', "Price:")?> <?=$product->price?> ron</h6>

<h7 class="card-text"><?=$product->description?></h7> <br>

<h7 class="card-text"><?php echo traducere('Adaugat de: ', "Added by: ")?> <?=$product->added\_by\_user?></h7>

<a class="nav-link" href="<?php echo base\_url().'products/add\_product/'.$product->products\_id?>"> <?php echo traducere('Adauga in ', "Add to ")?></div></div><?php }?>

Listare 3 Funcția foreach



Figură 25 Produse (Sursa: Proiecția autoarei)

Produsele pot fi căutate după nume pentru a putea fi găsite mai ușor și de asemenea acestea pot fi sortate în funție de preț, folosind funția *get\_items\_filtered* ­ aflată în *products\_model.* Am creat un query care preia din baza de date, din tabelul *products*, interogările. De asemenea, este pusă la dispoziție și filtrarea după nume. La fel ca și pentru sortarea după preț și aici am create un query ce a luat informațiile necesare din baza de date pentru a face posibilă funcționalitatea.

function get\_items\_filtered($filter){

if ($filter == "pretDescr"){

$result = $this->db->query('SELECT \* FROM products ORDER BY price DESC')->result();

}else if($filter == "pretCresc"){

$result = $this->db->query('SELECT \* FROM products ORDER BY price ASC')->result();}

return $result;}

Listare 4 Filtrare în funție de preț

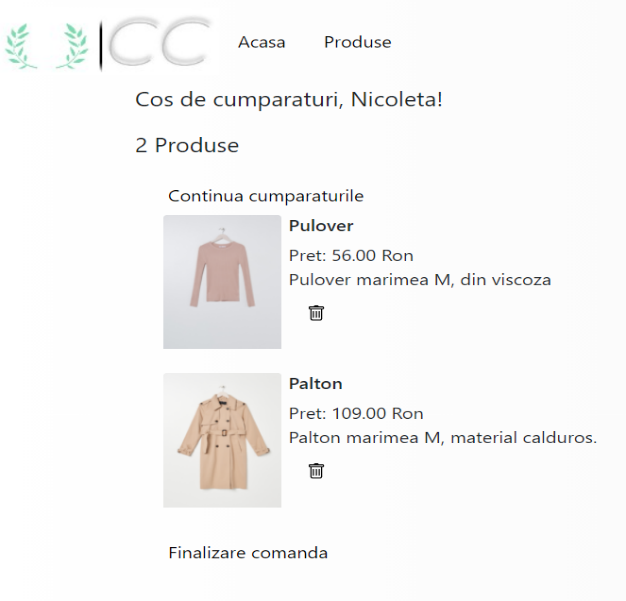
function get\_items\_filtered\_name($filterName){

$result = $this->db->query('SELECT \* FROM products WHERE title RLIKE "'.$filterName.'"')->result();

return $result;}

Listare 5 Filtrare în funcție de nume

Pentru a adăuga în coș, utilizatorul trebuie să fie obligatoriu autentificat. În momentul apăsării pe butonul de *adaugă în coș* informațiile cu referile la produsul dorit vor fi afișate în coș, însă utilizatorul va rămâne tot pe pagina de produse, acest lucru fiind posibil *get* aflată în model. Folosind sesiunile, mi-am stocat informațiile pentru a putea fi utilizate pe mai multe pagini. Numele utilizatorului va fi afișat împreună cu mesajul *Coșul de cumpărături*.



Figură 27 Coș de cumpărături (Sursa: Proiecția autoarei)

De asemena sunt prezente și butoanele pentru a continua cumpărăturile, iar clientul va fi redirecționat către pagina cu produse și cel pentru a finaliza comanda ce îl va duce pe o pagina pentru a completa datele pentru livrare și plată, după cum se poate observa în figura 26. Produsele pot fi ștese din coș folosind funția remove\_product aflată în controller-ul products. Am utilizat sesiunile pentru șterge produsul doar pentru cel ce este conectat în momentul respectiv, iar după ce articolul va fi șters, se va face reîncărcarea paginii.

Pentru a putea fi plasată comanda s-au creat funcțiile foreach cu ajutorul cărora se preiau informațiile din baza de date. Codul din listarea cu numărul 6 reprezintă faptul că a fost creat un query pentru a putea fi listate interogările din baza de date.

foreach ($data as $order\_info){

$query = $query . '"'. $order\_info . '",';}

$query = substr\_replace($query ,"",-1);

$query = $query . ')';

$result = $this->db->query($query);

$insert\_id = $this->db->insert\_id();

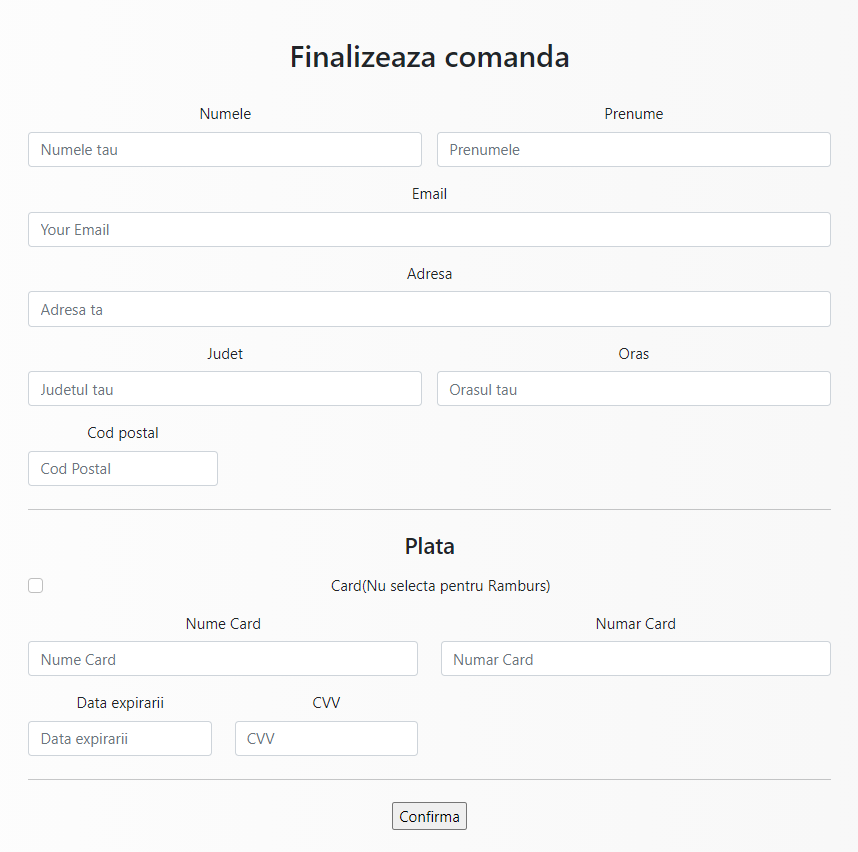
foreach ($\_SESSION["shopping\_cart"] as $product){

$query = 'INSERT INTO orders (order\_id,product\_id) VALUES ('.$insert\_id.','.$product.');';

$this->db->query($query);}

Listare 6 Funcția de creare a comenzii

După apăsarea butonului finalizare comandă se va deschide automat formularul pentru finalizare, așa cum este exemplificat în figura 27, prezentată mai jos.



Figură 28 Finalizare comandă (Sursa: Proiecția autoarei)

Datele ce vor fi introduce în aceast formular vor fi salvate în baza de date folosind funcția *crează\_comandă* din *finalizare­\_model.* Odată cu apăsare butonului *confirmă* atât clientul, cât și cel care a încărcat produsul vor fi anunțați printr-un email cu informații privind aceste acțiuni. Clientul va fi notificat cu sumarul comenzii și produsele cumpărate, iar vânzătorul va primi notificare legată de faptul că produsul său a fost vândut.

Pentru a crea opțiunea confirmarea prin email am utilizat platforma SendinBlue care pune la dispoziție trimiterea a 300 de astfel de mesaje pe zi. Mi-am creat un email numit *cc.store.no.replygmail.com* de pe care vor fi trimite automat aceste emailuri. În *email­­\_model* am introdus mesajul ce va fi trimi și de aceea adresa de email introdusă în formularul pentru finaliarea comenzii trebuie să fie unul valid. Pentru început am setat setările pentru SMTP ce au fost configurate după cum urmează:

$config = Array(

'protocol' => 'smtp',

'smtp\_host' => 'smtp-relay.sendinblue.com',

'smtp\_port' => 587,

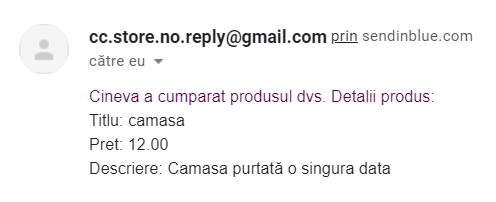
'smtp\_user' => 'cc.store.no.reply@gmail.com',

'smtp\_pass' => 'P7XMdr6TnOmNEfbJ');

Listare 7 Confirmare prin email

Protocolul pentru trimiterea mesajelor a fost setat ca și smtp, deoarece prin acesta se poate indica serverului un mechanism de autentificare. *Smtp\_host* este adresa furnizată de către Sendin Blue, iar portul 587 este folosit pentru serverul de poșta electronica. *Smtp\_user* reprezintă emailul de pe care vor fi trimise mesajele.





Figură 30 Răspunsul automat trimis prin email (Sursa: Proiecția autoarei)

După ce a fost trimisă notificarea prin email, comanda plasată se va salva automat în pagina istoric comenzi. Pentru ca informațiile să fie preluate din baza de date am folosit operatorul RLKE ce efectuează un argument.

Folosind, de asemenea și aici sesiunile, se va prelua din username-ul care a plasat comanda pentru a se putea crea istoricul pe baza informațiilor necesare. Istoriul va fi salvat chiar si după delogarea utilizatorului. Am creat un array pentru a salva în baza de date adresa, produsele și sumarul comenzii.

$order = array(

"adresa" => $result->address. " " . $result->county,

"produse" => $produse,

"total\_comanda" => $total\_comanda,);

array\_push($history,$order);

Listare 8 Șir array pentru istoric

# **Testarea sistemului informatic**

Testarea software-ului este o tehnică des întâlnită pentru verificarea validării și calității. Testarea reprezintă o procedură pentru un program sau pentru un sistem pentru a fi găsite erorile. Testarea poate fi de două feluri: Black Box Testing și White Box Tesing.

Black Box Testing numită și testarea funcțională, proiectează cazuri de test pe baza informațiilor specifice. Aceasta nu are acces la codul sursă intern ci pune accent pe specificațiile cerințelor. Această testare este făcută de către clienți și este posibilă doar pe produsul finit.

White Box Testing numită și testare structurală, proiectează cazuri de test pe baza logicii codului. Aceasta are acces la cod și este folosită pentru a depista erorile logice din cod. De asemenea, aceasta poate descoperi erorile tipografice si programarea incorectă. Aceasta este posibilă la toate nivelurile de dezvoltare (Nidhra, (2012)).

Prin tabelului cu numărul 3, am creat o testare Black Box pentru rularea aplicației. Aceasta a fost testată pe toate cele trei browsere, iar rezultatele sunt conform cerințelor. Rularea este rapidă, iar navigarea a rămas la fel de simplă .

Tabel 3 Rularea aplicației

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nr Crt | Denumire | Pași | Rezultat | Status |
| 1 | Rularea pe browser - ul Google Chrome | 1.Se deschide Chrome  2.Se introduce adresa | Va aparărea pagina specială pentru vizitatori | În regulă |
| 2 | Rularea pe browser - ul Internet Explorer | 1.Se deschide Explorer  2.Se introduce adresa | Va aparărea pagina specială pentru vizitatori | În regulă |
| 3 | Rularea pe browser - ul Mozilla Firefox | 1.Se deschide Mozilla  2.Se introduce adresa | Va aparărea pagina specială pentru vizitatori | În regulă |

Pentru accesul în aplicație am creat o testare tot de tipul Black Box pentru a vedea dacă rezultatele corespund cu cerințele, asa cum se poate observa în tabelul 4. Am verificat fiecare buton ce pagina deschide, iar în urma acestei testări am concluzionat că totul funcționează conform cerințelor.

Tabel 4 Accesul în aplicație

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nr Crt | Denumire | Pași | Rezultat | Status |
| 1 | Înregistrare | Se introduc datele necesare în formularul pentru validare, conform regulilor. | Apariția paginii de autentificare | În regulă |
| 2 | Autentificare | Introducerea username-ului și parolei | Deschiderea paginii principale | În regulă |
| 3 | Articol nou | Se accesează pagina de adăugare produs nou și se completează formularul | Deschiderea paginii cu produse | În regulă |
| 4 | Adăugare în coș | Se apasă pe buton de adăugare în coș prezent sub articol | Se actualizează coșul | În regulă |
| 5 | Finalizare comandă | După accesarea paginii specifică pentru coș, se va face finalizarea | Se deschide pagina unde trebuie să se introducă datele de livrare | În regulă |
| 6 | Istoric comenzi | După finalizarea comenzii se va face confirmarea prin email | Se va deschide istoricul comenzilor | În regulă |

O testare Black Box am creat și pentru administrator. Mai exact, am verificat dacă cerințele de la începutul aplicației corespund cu ceea ce s-a finalizat. Am descris pașii și rezultate în tabelul cu numărul 5.

Tabel 5 Administrator

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nr Crt | Denumire | Pași | Rezultat | Status |
| 1 | Înregistrarea | Se vor introduce datele necesare | Se va deschide pagina de autentificare | În regulă |
| 2 | Logarea | Se vor introduce username-ul și parola adăugate anterior | Se va deschide pagina pentru efectuarea operațiilor | În regulă |
| 3 | Ștergerea | În funcție de operațiile care doresc a fi făcute, se deschide o pagină specifică | Se vor putea șterge atât utilizatori cât și produse sau comenzi | În regulă |
| 4 | Editarea | În funcție de operațiile care doresc a fi făcute, se deschide o pagină specifică | Se vor putea edita atât utilizatori cât și produse sau comenzi | În regulă |
| 5 | Vizualizarea | În funcție de operațiile care doresc a fi făcute, se deschide o pagină specifică | Utilizatorii, produsele și comenzile pot doar să fie vizualizate | În regulă |
| 6 | Crearea | În funcție de operațiile care doresc a fi făcute, se deschide o pagină specifică | Administratorul poate crea un produs, o comandă sau chiar un utilizator | În regulă |

# **Concluzii**

Scopul acestei lucrări de licență a fost acela de a crea un magazin online de haine second-hand. Am ales această tema, deoarece m-am gândit la persoanele care aruncă, ori le înghesuie prin dulap. Folosind această aplicație, oamenii pot achiziționa haine al preț redus, ceea ce este foarte convenabil mai ales pentru studenți. Am ales să folosesc o interfață simplă, utilizând culori cât mai deschise, iar navigarea să fie simplă și pe înțelesul tuturor.

De asemenea, am ales un limbaj de programare ușor de folosit și de înțeles, iar PHP a fost exact ceea ce am avut nevoie pentru a ajunge la scopul final. Am încercat să mă organizez și să îmi pun termene pentru realizarea task-urilor, ceea ce a fost spre binele meu. Pe viitor cu siguranță o să aduc îmbunătățiri proiectului și poate o să ajung să-l folosesc ca și sursă proprie de venit.

Folosind și un framework foarte bine organizat, codul a fost ușor de scris iar legăturile veneau, de cele mai multe ori, de la sine. Bazându-se pe MVC, acesta mi-a venit în ajutor pentru a scrie un cod corect și coorent. Această aplicație a avut ca și grup țintă tinerii, pentru că trăim în lumea tehnologiei, iar în ultima perioadă cele mai multe cumpărături au fost în sistem online.

Pentru a realiza o aplicație cât mai complexă am folosit platforma Sendin Blue pentru a putea crea opțiunea de confirmare a comenzii prin email. Tot confirmare prin email va primi și vânzătorul în momentul în care produsul său a fost cumpărat.

Așadar, cu pasi mici am reușit să creez aplicația pe care mi-am dorit-o, numită CC-Store. Consider că site-ul este unul prietenos, complet și poate fi utilizat fără complicații de oricine.

# **Bibliografie**

Boos, D. D. (May 2003). *"Introduction to the Bootstrap World.".*

Cohen, D. L. ((2004)). *An introduction to agile methods.* Adv. Comput., 62(03), 1-66.

De Jonge, M. (2002). *"Source tree composition.".* International Conference on Software Reuse. Springer, Berlin, Heidelberg,.

Deacon, J. ((2009)). *"Model-view-controller (mvc) architecture.".* Online][Citado em: 10 de março de 2006.].

Griffiths, A. (2010). *CodeIgniter 1.7 Professional Development.* Packt Publishing Ltd.

Hanson, M. D. ((2000)). *"The client/server architecture.".*

Kalliamvakou, E. e. ((2016) ). *"An in-depth study of the promises and perils of mining GitHub.".*

Nidhra, S. a. ((2012)). *"Black box and white box testing techniques-a literature review." .* International Journal of Embedded Systems and Applications (IJESA) 2.2 .

Rodriguez-Echeverria, R. e. ((2016)). *"Automating IFML Specification of CRUD Operations.".*

Tiwari, M. K. ((2016)). *"An interactive graphical user interface (GUI) for the CATGIXRF program–for microstructural evaluation of thin film and impurity doped surfaces.".* X‐Ray Spectrometry 45.4.