**CAPITOLUL 1. INTRODUCERE**

**1.1 Situatia actuala și motivatia temei**

Fenomenul cel mai reprezentativ pentru societatea actuala este dezvoltarea accelerata a tehnologiei care a inceput acuma cateva zeci de ani, deoarece aceasta a devenit un element foarte important in viata fiecarui om. Astfel in ziua de astazi, dispozitivele electronice ne fac viata mai usoara și mai comoda avand posibilitatea de a accesa și trimite informatii cu ajutorul a catorva clickuri sau atingeri de ecran.

Cu timpul, companiile s-au adaptat treptat la acest stil de viata al oamenilor, devenind in cele din urma un standard ca acestea sa aiba o prezenta online intr-un fel sau altul, fie prin existenta unor website-uri pentru clienti prin care acestia pot lua contact cu firma sau se pot informa, fie prin exista unor platforme destinate angajatilor care le permite acestora sa gestioneze și sa automatizeze anumite elemente cheie din cardul business-ului. In general companiile sunt dispuse sa investeasca tot mai mult in dezvoltarea unor astfel de platforme pe masura, iar cele care aleg sa nu investeasca in domeniul online pierd o multime de beneficii fiind depasite de catre ceilalti competitori.

Un exemplu extrem de intalnit este prezenta unei aplicatii de e-commerce pentru magazine.

Cu ajutorul unei astfel de aplicatii, se pot gestiona automat stocuri și vanzari, eliminandu-se erorile umane daca aplicatia este proiectata corect și respecta cerintele impuse. De altfel este important ca navigarea aplicatiei sa fie intuitivă și usoara. De asemenea un site cu un aspect neplacut sau neingrijit le poate inspira utilizatorilor nesiguranta și neincredere pierzandu-se astfel posibili clienti in ciuda serviciilor oferite mai ales in cadrul unor firme mai putin cunoscute.

In concluzie o platformă online este o nevoie pentru aproape orice business, deoarece se economiseste timp și bani, se elimina erorile umane din managementul contabilitatii și logisticii și le ofera posibililor clienti oportunitatea de a apela la serviciile firmei rapid și usor.

**1.2 Scopul lucrarii**

Se urmareste dezvoltarea unei platforme online special concepute pentru management-ului unui magazin care comericalizeaza articole sportive pentru ski și snowboarding.

Clientii, la randul lor, au la dispozitie o interfață intuitivă și prietenoasă prin care pot vizualiza și filtra produsele dupa mai multe criterii, pe care le pot salva ulterior intr-o lista de produse favorite sau intr-un cos de cumparaturi. Aplicatia le pe permite utilizatorilor sa plaseze comenzi și sa cumpere produsele dorite, deci implicit este prezent și un sistem de autentificare.

Anumiti utilizatori autentificati și autorizati cu rol de administrator au posibilitatea de a monitoriza și modifica stocurile produselor și de a gestiona comenzi prin intermediul unei interfețe diferite.

Aceasta aplicatie le ofera o solutie eficientă și convenabilă atat administratorilor cat și clienților. Prin intermediul acestei aplicatii software, se pot aduce imbunatatiri semnificative in gestionarea elementelor de logistica și in relatia cu clientii, crescanda satisfacțiaacestora și vanzarile afacerii.

**1.3 Structura lucrării**

Lucrarea este împărțită in X capitole principale .

Capitolul 1 – se prezinta motivatia temei, scopullucrării și cateva concepte generale

Capitolul 2 –se vor analiza cerintele sistemului informatic folosind diagrame

Capitolul 3 – se va proiecta aplicatia și vor fi prezentate cateva concepte importante din diferite perspective folosind diagrame

Capitolul 4 – se vor prezenta detalii tehnice legate de implementarea propriuzisa a aplicatiei

**1.4 Baze de date – concepte generale**

**1.4.1 – Definitia bazei de date**

O baza de date este o colectie organizata de informatie, de obicei localizata pe un dispozitiv de stocare. In general o baza de date este controlata de un sistem de gestiune a bazelor de date (DBMS – database management system). Impreuna, baza de date, DBMS-ul și aplicatiile asociate cu acestea formeaza un sistem informational cu baze de date.

In functie de cerintele care trebuie satisfacute, pot exista o multime de tipuri de baze de date, iar cateva dintre acestea sunt:

1. B.D. relationale – cel mai intalnit tip de baze de date și cel folosit in aplicatie
2. B.D. non-relationale – alt tip comun de baza de date
3. B.D. JSON
4. B.D. orientate pe obiecte

**1.4.2 – Baze de date centralizate și distribuite**

In etapa de proiectare a unui sistem informatic cu baze de date, trebuie analizate cerintele sistemului și caracteristicile fiecarui tip de baza de date.

Principala diferenta intre cele doua categorii de baze de date este ca in baza de date centralizata, informatiile sunt manipulate și stocate intr-o singura locatie, pe cand intr-o baza de date distribuita(decentralizata) informatiile se gasesc in mai multe locatii.

**1.4.2.a – Avantajele unei baze de date centralizate**

Datele se afla pe un singur dispozitiv. Orice informatie poate fi obtinuta, adaugata sau modificata rapid deoarece totul este intr-un singur loc.

Securitatea datelor este usor de intretinut deoarece toate datele sunt stocate intr-o singura locatie iar masurile de securitate pot fi concentrate pe singurul punct de acces. In cazul pierderii de date acestea pot fi recuperate daca exista o copie de rezerva actualizata periodic și stocata pe un alt dispozitiv.

Modelul datelor este simplu de proiectat in comparatie cu o baza de date distribuita. Datele sunt consistente datorita relatiilor stabilite intre tabele. In cazul unei baze de date dristribuite, datele sunt implicit mult mai complexe și mai greu de manipulat.

Redundanta minima- acest avantaj este o urmare a modelului simplu al datelor. Informatia redundanta este cea care este intalinita in doua sau mai multe locuri. Sansele de aparitie a acestora sunt proportionale cu complexitatea modelului datelor.

**1.4.2.b Dezavantajele unei baze de date centralizate**

Limitari din cauza vitezei- daca sunt multi utilizatori care acceseaza acelasi set de informatii intr-un timp scurt, pot aparea probleme de performanta semnificative, fenomen care nu apare atat de des intr-o baza de date decentralizata.

Daca exista probleme legate de sistemul care detine baza de date, aceasta poate fi inaccesibila pana la remedierea problemelor, lucru rezolvat in cadrul bazei de date distribuite: daca o locatie “cade”, celelalte locatii ii vor indeplinii functia, iar serviciul corespunzator functioneaza in continuare.

Riscul distrugerii informatiei- in cazul in care se intampla ceva cu sistemul fizic, exista posibilitatea ca toate datele stocate sa fie pierdute pentru totdeauna daca nu a fost facuta o copie de rezerva a datelor inaintea incidentului.

**CAPITOLUL 2. CERINTELE SISTEMULUI**

Aplicatia conceputa va trebui sa indeplineasca cateva cerinte minime pentru a-si atinge scopul. Programatorul trebuie sa urmeze aceste cerinte pentru ajunge la produsul final dorit.

La final aplicatia trebuie sa indeplineasca functiile de baza specifice oricarei platfome de eCommerce și anume:

* Autentificare și autorizare pentru fiecare utilizator in parte ( atat pentru clienti cat și pentru angajati).
* Efectuarea de operatii CRUD (Create Retrieve Update Delete) asupra datelor prin prezenta unor interfețe specializate.
* Stocarea articolelor sportive in baza de date trebuie sa fie cat mai detaliata. Echipamentele destinate sporturilor de iarna sunt foarte diversificate iar stocarea acestora trebuie sa reflecte realitatea cat mai bine.
* Avand in vedere modelul de date, trebuie implementata o functie de filtrare pe masura, cat mai amanuntita.
* Interfata clienților trebuie sa fie simpla și intuitiva. Trebuie sa se poata ajunge in orice punct al aplicatiei in cel mult 3 sau 4 clickuri.
* Clientii trebuie sa aiba posibilitatea de a salva produse intr-o lista de favorite.
* Clientii sa poata sa initializeze comenzi pe baza unui cos de cumparaturi.
* Clientii pot vizualiza starea curenta a comenzilor plasate și istoricul comenzilor

Cateva dintre aceste cerinte pot fi descrise cu ajutorul unor diagrame.

In continuare se va prezenta diagrama cazurilor de utilizare pentru un site de ecommece.

**Diagrama cazurilor de utilizare (Use case)**

Orice program software trebuie sa interactioneze cu exteriorul fie cu alti oameni, fie cu alte sisteme.

Diagrama cazurilor de utilizare (use case diagram) este o diagrama UML, utilizata in faza de proiectare și analiza a unui sistem software pentru a defini functionalitatile sistemului și pentru a clarifica cerintele și nevoile impuse atunci cand acestea trebuiesc transmise altor personae, intr-un mod usor de inteles, independent de tehnologia utilizata in implementarea aplicatiei.

Programatorii, pot urmarii aceste diagrame pe parcursul intregului proces de dezvoltare pentru a se asigura ca sistemul este construit conform specificatiilor și cerintelor initiale.

O diagrama use case are cateva componente specifice: actori, use case-uri și relatiile dintre ele.

**Actorii**

Acestia reprezinta niste entitati externe care pot avea roluri diferite și care interactioneaza cu sistemul descris. Actorii pot fii niste simplii utilizatori, sisteme software sau dispozitive hardware și se reprezinta grafic printr-o persoana.

Shape

Description automatically generated with low confidence

Figura 2.1. Reprezentarea grafica a unui actor

**Use case-uri**

Acestea reprezinta actiunile pe care le poate executa un actor in cadrul sistemului. De obicei sunt evidentiate cele mai relevante actiuni, și nu se prezinta detalii cu privire la modul de executie sau la implementarea acestora

Din punct de vedere grafic, un use case se reprezinta printr-un oval in interiorul caruia se afla numele cazului de utilizare. Acesta trebuie sa fie simplu și usor de inteles pentru oameni și sa reprezinte o actiune.

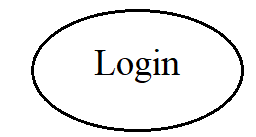


Figura 2.2. Reprezentarea grafică a unui use-case care descrie actiunea de logare

**Tipuri de relatii**

Intre use case-uri și actori se stabilesc relatii care sunt importante pentru descrierea modului in functioneaza sistemul și cerintele de implementare ale acestuia.

Exista cateva tipuri de relatii: asociere, dependenta și generalizare.

**Relatia de asociere**

**Relatia de asociere** se poate defini intre use case-uri sau intre actori și use case-uri. Aceasta sugereaza comunicarea intre componentele pe care le uneste și se poate reprezenta grafic printr-o linie.

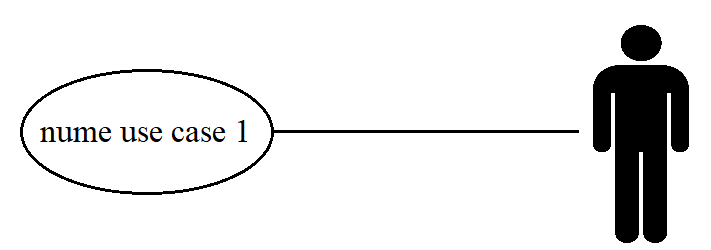


Figura 2.4. Reprezentarea grafica a unei relatii de asociere intre un use case și un actor

**Relatia de dependenta**

Aceasta se poate forma intre doua cazuri de utilizare și cuprinde doua tipuri: relatie de includere sau de extindere.

**Relatia de includere**

Relatia de tip **include** este folosita atunci cand executarea unui use case este conditionata de executia altui use case.

In exemplul urmator, Use Case-ul A include use Case-ul B și se reprezinta grafic printr-o sageata cu linie punctata pe care scrie “include”, iar sageata este indreptata catre use Case-ul inclus. Cu

alte cuvinte, atunci cand se executa cazul A și cazul B se executa obligatoriu, iar cazul B este incomplet fara A.

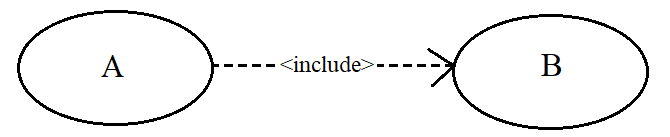


Figura 2.5. Depeneta de tip **include**

**Relaxia de extindere**

Relatia de extindere este folosita pentru a-i adauga sistemului o noua functionalitate, pentru ca un Use Case il extinde pe celalalt. In exemplul urmator se prezinta faptul ca use case-ul B extinde use case-ul de baza A. A-ul trebuie sa aiba sens de unul singur, iar B-ul practic ii adauga ceva in plus lui A și este dependent de A.

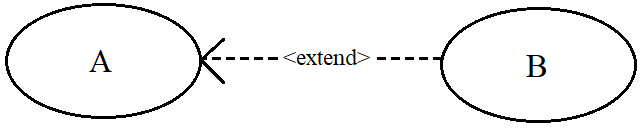


Figura 2.6. Dependenta de tip **extend**

**Relatia de generalizare**

Aceasta este similara relatiei de mostenire din programarea orientata pe obiect. Se arata ca un actor mosteneste rolul și use case-urile parintelui și se foloseste uneori cand se doreste evidentierea unor specializari sau generalizari. De exemplu actor “Customer” este mostenit de catre alti doi actori “Existing user” și “Guest user”. Customer-ul poate sa se uite la produse. Intre timp, mostenitorii sai au atat use case-urile proprii, cat și use case-ul Customer-ului.

A picture containing screenshot, circle, black and white

Description automatically generated

Figura 2.7 – Diagrama Use Case pentru un magazin online

In cele ce urmeaza, se vor explica componentele diagramei și semnificatia cazurilor de utilizare:

-Avem 4 actori: Customer, Existing User ,Guest User și Admin

-Customer este orice utilizator al site-ului. Acesta are posibilitatea de a vizualiza și de a filtra produsele

-Existing User este orice utilizator care este prezent in baza de date . Acesta poate vizualiza produse și aplica filtre de cautare. Are posibilitatea de a se autentifica cu contul propriu pe baza caruia poate executa alte cateva actiuni: adaugare de produse intr-o lista de favorite și un cos de cumparaturi și vizualizarea acestora. Pe baza continutului din cosul de cumparaturi poate plasa comenzi, ale caror stari poate fi urmarit.

- Guest User este orice utilizator care nu este inregistrat in baza de date. Acesta se poate inregistra și poate vizualiza catalogul de produse. Se evidentiaza faptul ca un simplu user vizitator are privilegii limitate in comparatie cu un user prezent in baza de date

-Admin – un admin este un actor care are alte privilegii fata de orice Customer. Acesta poate edita catalogul de produse și poate vizualiza datele de contact ale fiecarui user si comenzile acestuia. De asemenea are datoria de a edita starea comenzilor.

**Diagrama entitiati si relatii (ERD)**

Un pas foarte important in dezvoltarea unei aplicatii este creearea unei diagrame ERD.

In acest stagiu al proiectarii, se prezinta la nivel de concept entitatile aplicatiei

**CAPITOLUL 4 – Implementarea aplicatiei**

Aplicatia este constituita din 3 parti componente principale:

1. Frontend-ul : reprezinta interfața prin care utilizatorii (atat administratorii cat și clientii) iau contact cu aplicatia. S-a folosit React, o biblioteca JavaScript folosita pentru dezvoltarea de interfețe de utilizator și Typescript , care exinde sintaxa limbajului JavaScript, permitandu-se definirea tipurilor de date in mod static și nu dinamic precum in JavaScript-ul simplu. Framework-ul React a fost folosit pentru creearea unor aplicatii precum Facebook, Netflix, Instagram, Discord și multe altele.

2. Backend-ul: reprezinta “creierul” aplicatiei. In functie de datele primite de la partea de frontend, acesta introduce informatii in baza de date sau le extrage și le trimite sub forma de raspuns inapoi catre frontend. Am folosit Java Spring Boot. Acestea este un framework al limbajului Java care ofera suport pentru dezvoltarea de aplicatii web. Aici am definit modelele entitatiilor bazei de date și API-ul prin care se face comunicarea cu frontend-ul. De asemenea am implementat și o parte de autentificare și autorizare a utilizatorilor prin Spring Security care este tot un framework al limbajului Java.

3. Baza de date: S-a folosit o baza de date relationala MySQL, unde am salvat toate informatiile aplicatiei, pornind de la tipurile de produse la datele utilizatorilor.