**UNIVERSITATEA “DUNĂREA DE JOS” DIN GALAȚI**

**FACULTATEA DE AUTOMATICĂ, CALCULATOARE, INGINERIE ELECTRICĂ ŞI ELECTRONICĂ**

**SPECIALIZAREA: TEHNOLOGII INFORMATICE AVANSATE**

**SISTEM INFORMATIC DE GESTIUNE ȘI MONITORIZARE PENTRU O FIRMĂ DE CURIERAT**

**Profesor îndrumător,**

**Conf.dr.ing. Cornelia TUDORIE**

**Masterand,**

**Liviu George CHIRVASE**

**GALAȚI**

**2016**

CUPRINS

INTRODUCERE 4

CAPITOLUL 1 7

CERINȚE ŞI SPECIFICAŢII 7

1.1 Cerinţele aplicaţiei 7

1.2 Specificaţii 8

CAPITOLUL 2 10

ANALIZA PROBLEMEI 10

CAPITOLUL 3 12

PROIECTAREA SISTEMULUI 12

3.1. PROIECTAREA BAZEI DE DATE 12

3.1.1. Formularea scopului 12

3.1.2. Modelul conceptual al bazei de date 14

3.1.3. Modelul logic al bazei de date 15

3.2. PROIECTAREA APLICAŢIEI 18

3.2.1. Arhitectura aplicaţiei 18

3.2.2. Şirul evenimentelor 20

CAPITOLUL 4 22

IMPLEMENTAREA APLICAŢIEI 22

4.1. NOŢIUNI TEORETICE 22

4.1.1. World Wide Web 22

4.1.2. Protocol HTTP 22

4.1.3. Modelul client - server 23

4.1.4. Site-uri web 25

4.1.5. Hypertext 25

4.2. TEHNOLOGII FOLOSITE 26

4.2.1. Android 26

4.2.2. Firebug 29

4.2.3. Server-ul Apache 29

4.2.4 Sistemul de baze de date MySQL 30

4.3. PROGRAMARE WEB 33

4.3.1. HTML 33

4.3.2. CSS 35

4.3.2.1. Diferență între CSS și HTML 38

4.3.2.2. Avantajele CSS-ului 39

4.3.3. SERVLET 39

4.3.4. JAVA SERVER PAGES (JSP) 42

4.3.5. JAVASCRIPT 44

4.3.5.1. Ajax 46

4.4. IMPLEMENTAREA FIZICĂ A BAZEI DE DATE 48

4.5. IMPLEMENTAREA FIZICĂ A APLICAŢIEI 49

4.5.1.APLICAȚIA ANDROID 49

4.5.2. APLICAȚIA WEB 51

CAPITOLUL 5 55

TESTARE ŞI EVALUARE 55

5.1. TESTAREA 56

5.1.1. Strategii de testare 56

5.1.2. Testul de securitate 57

5.1.3. Testarea arhitecturii Client-Server 58

5.2. ASPECTE POZITIVE 58

CAPITOLUL 6 60

CONCLUZII 60

BIBLIOGRAFIE 61

# INTRODUCERE

În momentul de față curieratul este un termen des întâlnit și utilizat de majoritatea persoanelor, în special pentru modalitatea rapidă și eficientă de tranzitare a produselor, în acest context al coletelor, între diverse locații. Popularitatea companiilor de curierat a cunoscut o ascensiune in ultimul timp, în special datorită raspîndirii intense a magazinelor online și a comerțului electronic, care se află într-o continuă dezvoltare. Odată cu dezvoltarea comerțului electronic a crescut și cererea pentru distribuirea produselor achiziționate de la magazinele online, fapt ce a dus la o creștere exponențială a firmelor de curierat.

Firma de curierat are menirea de a realiza transferul pachetelor de la expeditor către destinatar în condiții optime de siguranță și integritate. În terminologia din domeniul curieratului pachetele sunt numite colete, iar un cumul de mai multe colete care au aceleasi informații de expediere și livrare, precum și aceleași informații calendaristice se consideră a forma o comandă.

În general firmele de curierat sunt alcătuite din personal împărțit în două categorii, pe de o parte operatorii, care gestionează coletele din punctul de lucru la care sunt arondați și de cealaltă parte se află curierii care își desfășoară activitatea “pe teren”, aceștia urmând a prelua, tranzita si livra coletele între expeditor, punctele de lucru ale firmei de curierat și destinatar. Mai sus a fost menționat termenul de punct de lucru, care va mai apărea în cadrul lucrării sub alte doua titlulaturi acelea de sediu sau hub. O altă clasificare a companiilor de curierat se poate face după aria de acoperire, unde de disting alte două categorii, curierat la nivel național si la nivel internațional. În prezenta lucrare se face referire la o firmă de curierat care își desfășoară activitatea doar la nivel național, dar ideea lucrării poate fi extinsă oricât de mult.

Procedura de expediere – primire a unuia sau a mai multor colete se aplica la majoritatea firmelor de curierat și constă în inițierea unei comenzi pe platforma web a companiei, după care operatorul din hub-ul la care este arondat expeditorul recepționează comanda, tot acesta va atribui comanda unui curier. Curierul este informat de asignarea la comanda respectivă și în cel mai scurt timp acesta se va deplasa la adresa expeditorului pentru ridicarea comenzii. Ulterior comanda este adusă la punctul de lucru, urmând ca un operator să genereze AWB-ul pentru fiecare dintre coletele din cadrul comenzii ridicate de curier, AWB ul fiind codul unic de identificare a unui colet. După stabilirea identității fiecarui colet al comenzii, se generează talonul care se va aplica pe fiecare dintre colete pentru identificarea mai ușoară a informațiilor expeditorului, destinatarului și a detaliilor comenzii. Pe lângă aceste informații pe talon va fi printat AWB-ul în clar , precum și codul pentru realizarea scanării pe parcursul expediției.

Operațiunile de preluare, livrare, verificare a statusului curent al unei comenzi pot fi mai simple și mai la îndemână atât pentru personalul firmei de curierat cât și pentru client. Astfel procedura expediției se poate îmbunătăți prin simpla utilizare a telefonului de serviciu primit de personal din partea firmei de curierat pentru o buna comunicare. Ținând cont că în momentul actual piața de telefonie mobilă se bazează în mare măsură pe Android ca sistem de operare pentru telefoanele utilizate, acestea ar putea avea o dublă întrebuințare și ar reprezenta o facilitate în munca desfășurată în firma de curierat. Prin intermediul telefoanelor “inteligente” firma de curierat nu se va mai limita doar la convorbiri telefonice, ci va putea utiliza telefonul și ca scanner pentru tranzitarea coletelor și informarea corectă și in timp optim a clientului despre statusul comenzii.

Lucrarea este alcatuită din două aplicații care conlucrează în scopul consituirii unui sistem informatic pentru gestionarea și monitorizarea pentru o firmă de curierat. Pe de o parte se regăsește aplicația Android, reprezentând partea care facilitează munca personalului companiei și de cealaltă parte se află aplicația web care își aduce aportul la crearea unui sistem complet.

Aplicația Android va fi distribuită și instalată pe telefonul primit din partea firmei de curierat, urmând ca fiecare curier / operator să se poată autentifica in sistem, cu ajutorul usernaname-ului și a parolei asociate acestuia. În urma autentificării aplicația oferă printre opțiuni: vizualizarea comenzilor de preluat și a comenzilor de livrat asociate utilizatorului curent, aceste opțiuni fiind disponibile doar curierilor. În plus aplicația are o secțiune de scanare, special concepută pentru scanarea coletelor în tranzitul acestora către destinatar, opțiune disponibilă pentru ambele categorii de angajați. Ultimele două opțiuni se referă strict la profilul de utilizator, acestea permit schimbarea parolei și ieșirea din contul utilizator.

Aplicația Web este compusă la rândul său din două secțiuni, o secțiune publică destinată în special clienților, unde aceștia pot efectua comenzi și pot vizualiza statusul curent al unui colet. Cealaltă secțiune, de această dată una privată este dedicată personalului de tip operator, care are opțiunea de a vizualiza comenzile nepreluate și a le procesa prin asignarea unui curier arondat punctului de lucru în scopul preluării acestora. O altă opțiune este vizualizarea comenzilor în curs de preluare pentru a asigura bunul mers al expedierii și a verifica în cazul unor incidente statusul corect și concordața între statusul înregistrat de curier și statusul real al comenzii. Ultima opțiune și probabil și cea mai importanta este vizualizarea comenzilor preluate care urmeazaă sa fie distribuite mai departe spre livrare. În această secțiune operatorul are opțiunea de generare AWB și de generare a talonului care se va aplica pe colet.

Lucrarea este structurată pe şase capitole:

* Primul capitol prezintă cerinţele şi specificaţiile aplicaţiei la nivel funcţional şi tehnic;
* Capitolul doi este alcătuit din analiza problemei, evidenţiind termenii specifici folosiţi în cadrul unei companii de curierat;
* Capitolul trei tratează proiectarea sistemului la nivelul bazei de date şi a aplicaţiei, comunicarea şi şirul evenimentelor la nivel de principiu;
* Capitolul patru urmăreşte modul cum a fost implementată aplicaţia la nivel tehnic
* Capitolul cinci prezintă rezultatele evaluării punctând aspectele pozitive ale sistemului şi modul de realizare a testării acestuia;
* Ultimul capitol cuprinde concluzii referitoare la aplicaţia prezentată.

# 

# CAPITOLUL 1

# CERINȚE ŞI SPECIFICAŢII

## **1.1 Cerinţele aplicaţiei**

Identificarea deficiențelor și înțelegerea necesităților în cadrul structurii și funcționării unei companii de curierat, reprezintă un prim pas în elaborarea temei propuse. În urma consultării diferitelor platforme web ale firmelor de curierat existente și din experiența unor comenzi efectuate online și livrate de companii de curierat, am obținut un cumul de informații necesare în stabilirea cerințelor și obiectivelor propuse în scopul realizării acestei lucrări.

Pentru realizarea sistemului informatic propus, pe baza informațiilor dobîndite din sursele expuse mai sus, au fost stabilite următoarele cerințe:

* Crearea unui sistem informatic dedicat unei firme de curierat, în scopul monitorizării și gestionării în timp optim a activității din cadrul companiei, precum și îmbunătățirea tranzitului expeditor – destinatar;
* Proiectarea unei baze de date comune ca suport în dezvoltarea sistemului alcătuit din aplicația Android și aplicația Web;
* Crearea unei aplicații Android, în scopul monitorizării mai eficiente a comenzilor;
* Vizualizarea de către curieri a comenzilor asignate (preluări și livrări) ;
* Actualizarea promptă a statusului comenzilor prin intermediul sistemului de scanare;
* Evidențierea în cadrul aplicației Web a două secțiuni distincte, o secțiune publică și o secțiune privată;
* Limitarea accesului în secțiunea privată prin intermediul unei interfețe de autentificareș
* Vizualizarea de către operatori în urma autentificării a comenzilor în sdiferite stadii precum și asignarea acestora către curieri;
* Generarea codurilor unice de identificarea (AWB) a coletelor, precum și a codurilor de tip QR, în scopul scanării;
* Generarea în format PDF a taloanelor aplicate coletelor.

## 

## **1.2 Specificaţii**

Proiectul îşi propune să reducă costurile investiției în dispozitivele de scanare și să utilizeze telefonul, o resursă deja existentă în munca actuală efectuată de curieri, ca pe o resursă multifuncțională atât în convorbiri cât și pentru scanarea coletelor. Sistemul își propune pe această cale să aducă și o îmbunătățire a transmiterii datelor și a informării corecte asupra statusului comenzilor.

Sistemul propus spre dezvotare, în special aplicația Android se adresează personalului companiei de curierat pentru o bună gestionare a comenzilor, dar și clienților acesteia care vor beneficia de informații prompte și corecte.

Mediul de programare va fi alcătuit din două componente, astfel se va folosi Android Studio, tool-ul recomandat de cei de la Android pentru dezvoltarea de aplicații în acest limbaj, iar pentru dezvoltarea aplicației Web se va utiliza Eclipse EE IDE recomandat pentru dezvoltarea proiectelor web, care folosesc tehnologii precum JSP, SERVLET, HTML, CSS şi JavaScript

Pentru realizarea aplicaţiei se va folosi:

* server-ul Web Tomcat Apache
* server-ul de DB MySQL

Aplicația Android se va dezvolta în tool-ul menționat mai sus, iar pentru testare pe poate utiliza un device cu sistem de operare Android sau se testează cu ajutorul unui emulator configurat de către dezvoltator și lansat prin intermediul Android Studio.

Aplicaţia Web va fi dezvoltată în Eclipse EE IDE, iar pentru testarea se va utiliza server-ul Apache Tomcat independent sau prin intermediul aplicației XAMPP care încorporează o serie de tool uri. O altă variantă de vizualizare și testare a aplicației este direct din Eclipse EE IDE prin crearea unei instanțe a server-ului Apache Tomcat și lansarea aplicației.

Pentru sistemul de baze de date se va utiliza server-ul de DB MySQL, iar ca și utilitare se va utiliza XAMPP pentru pornirea și oprirea instanței server-ului de DB și MySQL Workbench pentru proiectarea, administrarea și generarea unei scheme de ansamblu a bazei de date utilizată.

În plus pentru realizarea conexiunii între aplicația Android și server-ul de baze de date se vor folosi fișiere de tip php ca punt de legatură între cele două entități.

# CAPITOLUL 2

# ANALIZA PROBLEMEI

Firma de curierat este o societate economică, avînd ca obiect de activitate colectarea și livrare comenzilor de la un expeditor câtre un destinatar prin perceperea unei taxe încasate la preluare sau livrare.

Clientul firmei de curierat este entitatea care interacționează cu personalul firmei de curierat în primă fază prin intermediul interfeței web, ca urmare a solicitării unui curier în scopul transmiterii unei comenzi și ulterior la ridicarea comenzii solicitate, precum și la locația de livrare unde clientul are rolul de a recepționa comanda intițială. Astfel clientul are un dublu rol prin faptul că acesta poate juca rolul expeditorului sau al destinatarului. Expeditorul se consideră a fi clientul care trimite o comandă, solicitând un curier, iar destinatarul este cel care recepționează comanda.

Personalul firmei de curierat este alcătuită din două categorii de angajați, pe de o parte se regăsesc operatorii care își desfășoară activitatea la un punct de lucru, care poartă denumirea de hub, și au ca atribuții preluarea, procesarea, sortarea și transmiterea către urmatorul punct de lucru a coletelor până la locația de livrare. De cealaltă parte se regăsesc curierii care își desfășoară activitatea “pe teren”, având ca atribuții preluarea, tranzitarea și livrare comenzilor atribuite acestora.

Punctul de lucru sau hub-ul este locația în care își desfășoară activitatea personalul cu predilecție operatorii. Punctul de lucru se identifică prin adresa locației în care se regăsește, eventual un număr de telefon sau o adresă de email pentru relații oferite clienților. În acest caz se consideră că firma de curierat își desfășoară activitatea la nivel național, având hub-uri în fiecare dintre județele României.

Coletul este denumirea atribuită unui pachet de bunuri care se tranzitează între expeditor și destinatar. Comanda către firma de curierat se definește ca fiind entitatea ce înglobează un cumul de colete în scopul preluării și livrării acestora.

Astfel clientul în rol de expeditor dorește să transmită unul sau mai multe colete către clientul în rol de destinatar. Pentru acest fapt transmite o comandă către o firmă de curierat precizând detaliile necesare. Detaliile necesare trimiterii unei comenzi către firma de curierat sunt informațiile referitoare la expeditor și destinatar, printre care se numără numele celui care trimite comanda și a celui care va primi comanda, adresa de preluare și de livrare, se consideră adresa de preluare ca fiind aceeași cu a expeditorului, numărul de telefon al celor două entități expeditor și destinatar, precum și o adresă de email, care nu este obligatorie.

Următorul pas este preluarea comenzii de către un operator de la cel mai apropiat hub raportat la locația expeditorului, urmând ca acesta sa asigneze un curier arondat la hub-ul respectiv în scopul preluării comenzii de la expeditor. Curierul se va deplasa la locațiile indicate de informațiile primite la consultarea comenzilor asignate, va prelua comenzile și le va preda operatorului la hub.

Operatorul procesează fiecare comandă, fapt care presupune generarea AWB-ului pentru fiecare colet, AWB-ul fiind codul unic de identificare al unui colet, și generarea și printarea talonului care se aplica pe fiecare dintre colete. După această procedură operatorul sortează coletele în funcție de destinație, urmănd a le scana pe cele care se distribuie la alte puncte de lucru. Comenzile care au ajuns la hub-ul de destinație vor fi distribuite către curierii arondați acelui hub în scopul scanării și distribuirii coletelor către destinatari.

Pe parcusul întregului tranzit al coletului / coletelor, clientul trebuie sa aibă posibilitatea vizualizării stadiului în care se află comanda acestuia.

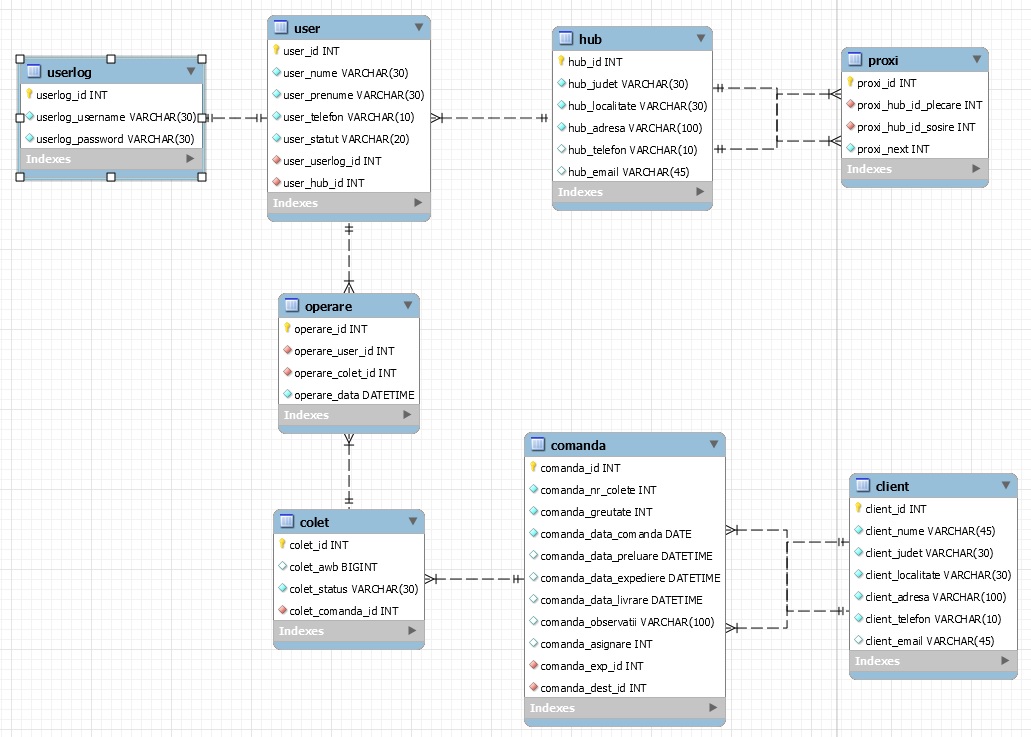
# CAPITOLUL 3

# PROIECTAREA SISTEMULUI

## **3.1. PROIECTAREA BAZEI DE DATE**

### **3.1.1. Formularea scopului**

Baza de date este alcătuită din opt tabele, partajate între aplicația Android și aplicația Web. Scopul bazei de date este de a stoca datele astfel încât să se poată obţine informaţii despre personalul firmei de curierat, punctele de lucru ale acesteia, despre comenzile aflate în diferite stadii sau informații despre clienții care au apelat la serviciile firmei de curierat.

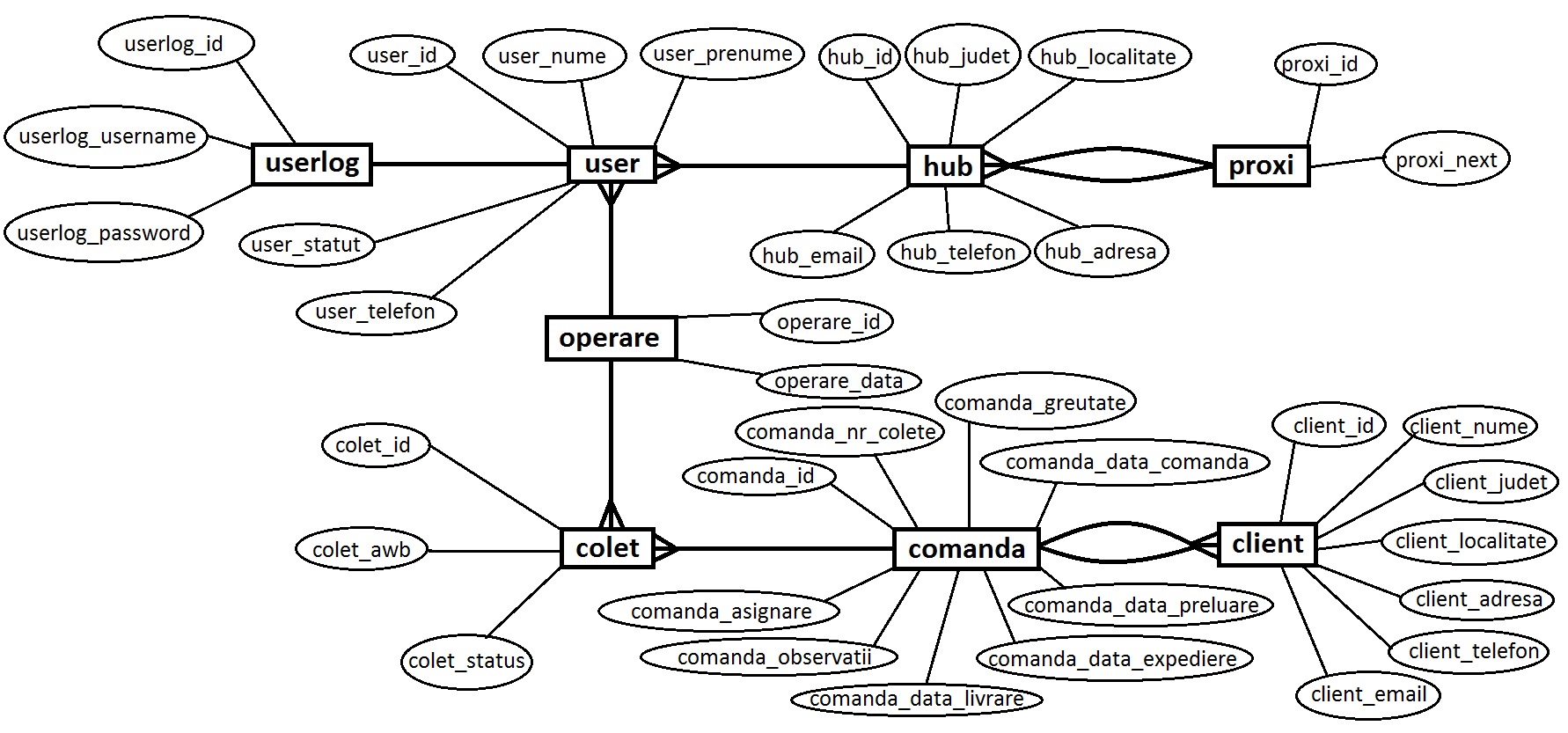


Fiecare tabelă are un scop bine stabilit:

* **userlog** – tabela conține datele de autentificare ale personalului firmei de curierat, cu ajutor cărora pot accesa sistemul. Tabela este populată la crearea conturilor de autentifficare de către un administrator;
* **user** – tabela conține informații despre personal, statutul fiecărei persoane în cadrul firmei (curier / operator), precum și punctul de lucru la care este arondat;
* **hub –** tabela deservește la stocarea informațiilor legate de fiecare punct de lucru
* **proxi –** tabela ce mapează graficul stabilit de firma de curierat în privința traseelor urmate în cursul tranzitului între diferite puncte de lucru până la destinație; - indică următoare locație două puncte de lucru;
* **operare –** este tabela care înregistrează toate interacțiunile între utilizator și colet precum și data la care s-a interacționat;
* **colet –** tabela care stochează informațiile referitoare la un colet, codul unic, statusul și comanda din care face parte.Se populează în momentul în care este efectuată o comandă, cu statusul inițial “nepreluat” și cu AWB-ul null;
* **comanda –** tabela ce conține informații referitoare la comenzi. Tabela se populează cand este efectuată o comandă cu informațile introduse de expeditor, cu excepția datelor referitoare la expediție;
* **client –** tabela are un dublu rol în cadrul sistemului, se stochează atât datele expeditorului cât și informațiile despre destinatar, cu mențiunea că expeditorul și destinatarul sunt create prin două înregistrări diferite.

### 

### **3.1.2. Modelul conceptual al bazei de date**



**Figura 3.1.2. Modelul conceptual al bazei de date**

### **3.1.3. Modelul logic al bazei de date**

Tabela **userlog**

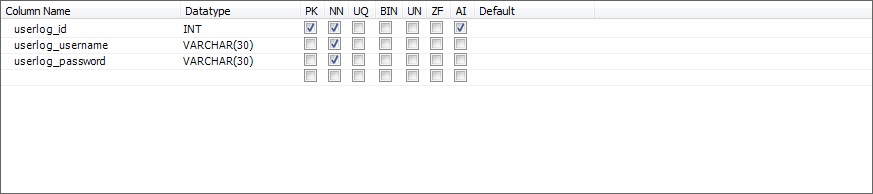


Tabela **user**

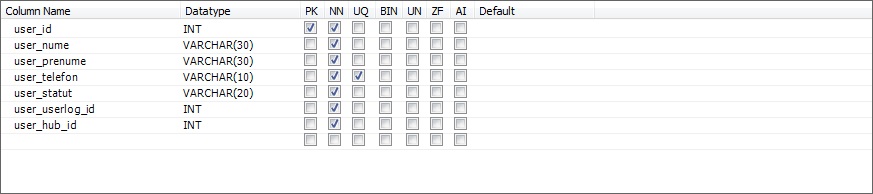


Tabela **hub**

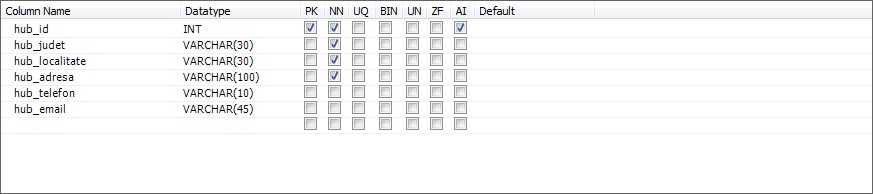


Tabela **proxi**

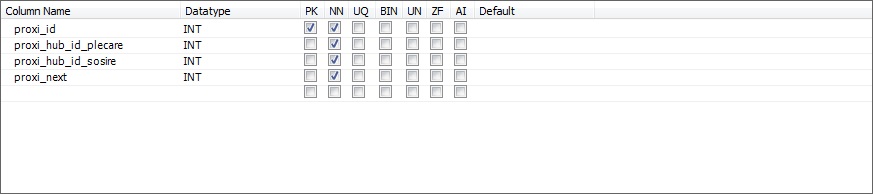


Tabela **operare**

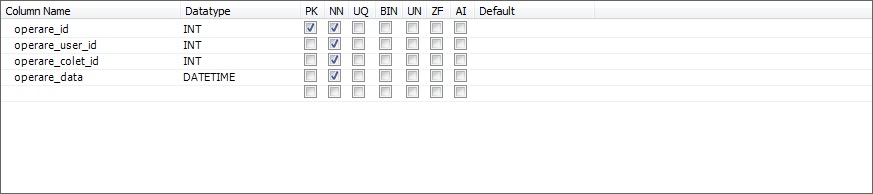


Tabela **colet**

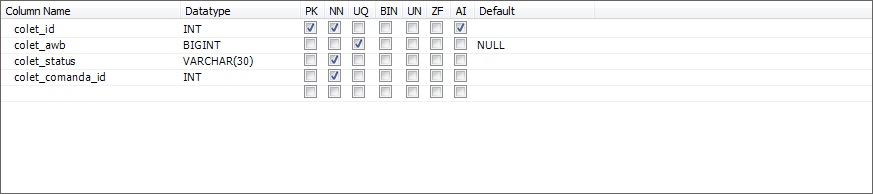


Tabela **comanda**

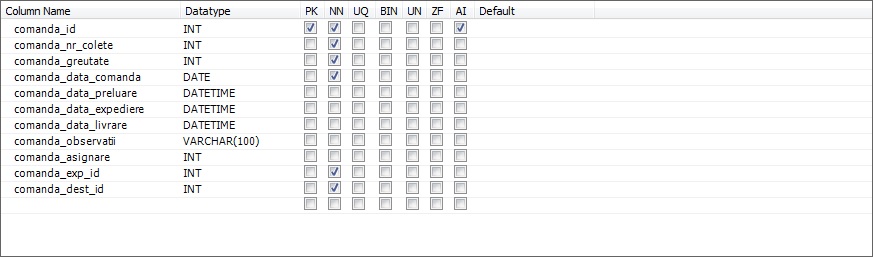
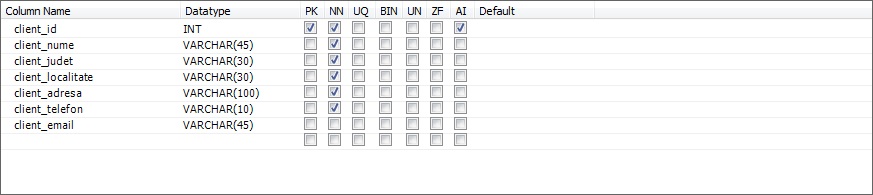


Tabela **client**



## 

## **3.2. PROIECTAREA APLICAŢIEI**

### **3.2.1. Arhitectura aplicaţiei Web**

În principiu arhitectura aplicațiilor este structurată pe următoarele trei nivele:

Nivelul de prezentare

HOME

Browser

Nivelul de logică a aplicației

**Figura 3.2.1. Arhitectura ”three-tier”**

Nivelul de date

1.Baza de date

Serverul SQL. Acest nivel include logica aplicației (răaspunde la interogările aplicației, face prelucrări masive, etc)

2.Serverul de aplicație

Asigură legătura între nivelul 3 (clientul) și nivelul 1 (serverul SQL) și realizează diverse prelucrări de date.

3.Interfața cu utilizatorul

Graphical User Interface – este clientul aplicației, numit ”thin client”, deoarece înglobează foarte puțină logică a aplicației și se limitează doar la validări și niște prelucrări minimale.

Fiecare dintre aceste nivele ale aplicației pot rula pe sisteme hard separate. Cea mai comună configurație este ca baza de date și serverul de aplicație să fie pe același server.

**1. Baza de date folosită** – Microsoft SQL Server – este un produs robust de la Microsoft, care asigură o capacitate de stocare extensibilă, securitatea datelor, instrumente de analiză și management a datelor, multiple posibilități de accesare / import , export a datelor, în diferite formate, respectiv integrate cu alte surse de date.

**2. Serverul de aplicație** – este nivelul la care se realizează majoritatea prelucrărilor de date.

**3. Programul client** – este interfața prin care utilizatorul interacționează cu aplicația. La acest nivel se fac foarte puține prelucrări de date, în majoritate doar validări de date. Interfața este perfect integrată cu mediul Windows, bogată în opțiuni, permițând flexibilitate în utilizare.

Arhitectura unei aplicații este influențată în principal de cerințe funcționale – serviciile oferite de sistem – și considerațiile privind calitatea (scalabilitatea sau performanța). Dincolo de acește cerințe, arhitecturile sunt influențate de constrângeri tehnice cum ar fi sistemul utilizat ( de exmplu sistemul de operare folosit), middleware, sistemele de moștenire care vor fi integrate, standardele utilizate, regulile de dezvoltare (de exemplu ghiduri de scriere a codului) sau aspecte de distribuire (de exemplu distribuirea în diverse locații a unei companii).

Deoarece sistemele software sunt în permanență schimbare arhitecturile sunt de obicei dezvoltate într-o manieră interativa, ceea ce nu garantează o arhitectura solidă. O abordare iterativă nu este suficientă pentru rezolvarea problemelor specifice de proiectare precum integrarea sistemelor de moștenire în dezvoltarea unei arhitecturi (șabloane de proiectare sunt foarte eficiente în sprijinul deciziilor de proiectare).

### **3.2.2. Şirul evenimentelor**

Această etapă constă într-o intercalare de evenimente desfășurate în cadrul celor două aplicații care formează sistemul. Sistemul urmărește să urmeze următoare secvență de evenimente:

1. Clientul accesează secțiunea “Cheamă curier” din cadrul aplicației Web și completează câmpurile solicitate în interfață, urmând ca la trimiterea informațiilor să fie înregistrată o comandă;

2. Operatorul se autentifică în sistem, accesează secțiunea ”Comenzi nepreluate” și asignează un curier pentru fiecare dintre înregistrări;

3. Curierul accesează aplicația Android de pe telefonul mobil, se autentifică în aplicație și accesează secțiunea “Comenzi”, unde se regăsesc toate comenzile asignate utilizatorului curent;

4. Curierul vizualizează detaliile fiecărei comenzi și se deplasează la fie care locație pentru a ridica comenzile;

5. După preluarea unei comenzi utilizatorul are opțiunea de a bifa comanda respectivă și a-i schimba statusul în comanda preluată;

6. Operatorul între timp poate verifica dacă comanda și-a schimbat statusul prin accesarea secțiunii “Comenzi preluate” din interfața Web;

7. Comenzile preluate de curier ajung la punctual de lucru unde operatorul generează awb-ul pentru fiecare colet al comenzilor și implicit generează și printează talon care se va aplica pe fiecare colet;

8. După alocarea codului unic coletelor, comanda trece în statusul procesat, urmând a fi sortate în funcție de destinație. Comenzile care ajung la punctul de lucru din aria de destinație sunt preluate de curierii arondați la acel punct de lucru, celelalte sunt scanate de operator și distribuite la urmatorul punct de lucru;

9. Curierii care ajung la punctual de lucru verifică comenzile care trebuie livrate, preiau un anumit număr de comenzi, scanează coletele, iar prin această scanare sunt asignați automat la comanda respectivă. În urma scanării la accesarea aplicațtie de pe telefon vor putea vizualiza în secțiunea “Livrări” comenzile de livrat asignate utilizatorului lor. De asemeni operatorul poate consulta lista de comenzi în curs de livrare pentru toți curierii de la acel punct de lucru, dar din interfața Web;

10. Curierul se va deplasa la fiecare din locațiile specificate urmând a livra la destinatar. În momentul livrării se scanează coletul / coletele, modificânduși statusul în livrat. În cazul în care curierul a omis să scaneze la livrare are opțiunea de a bifa ca fiind livrată comanda din aceeași secțiune de “Livrări”.

# CAPITOLUL 4

# IMPLEMENTAREA APLICAŢIEI

## **4.1. NOŢIUNI TEORETICE**

### **4.1.1. World Wide Web**

Termenul de World Wide Web este un sistem de documente şi informaţii care pot fi accesate prin reţeaua mondială Internet. Documentele care se află pe diferite calculatoare server în diferite locaţii pot fi accesate cu ajutorul unui identificator unic numit URL. Informaţia, care poate conţine şi imagini este afişată cu ajutorul unui program de navigare în web numit browser, care descarcă paginile web de pe un server şi le afişează pe un terminal “client” la utilizator.

La baza funcţionării web-ului stau trei standarde:

* HTTP - Stiva de protocoale OSI prin care serverul web şi browserul clientului comunică între ele;
* HTML - Standard de definire şi prezentare a paginilor web;
* URI - Sistem universal ce identifică resursele web de regăsire a paginilor web.

Unele dintre cele ami cunoscute aplicaţii browser sunt Microsoft Internet Explorer, Mozilla Firefox, Apple Safari, Google Chrome, Mozilla Camino, Opera Software - Opera, Nintendo DS browser şi Flock. Aplicaţia browser pe care am folosit-o în cadrul aplicaţiei este Mozilla Firefox.

### **4.1.2. Protocol HTTP**

HTTP este metoda utilizată pentru accesarea informaţiilor în Internet care sunt păstrate pe servere www. Fişierul trimis la destinaţie poate fi un document HTML, un fişier grafi, de sunet, animaţie sau video şi de asemenea un program executabil pe server sau un editor de text.

Când se accesează o adresă se cere convertirea de către protocolul DNS într-o adresă IP, se realizează transferul prin protocolul TCP pe portul 80 standard al server-ului HTTP, ca răspuns la cererea HTTP-GET. Răspunsul poate fi pagini HTML, fişiere cu stil (CSS), script-uri (JavaScript), dar pot fi şi pagini generate dinamic (JSP).

Transferul datelor se poate realiza în combinaţie cu o cerere pentru o resursă (HTTP-GET), în combinaţie cu o cerere specială (HTTP-POST).

HTTP se bazează pe modelul cerere-răspuns care are loc între browser şi server-ul web.

### **4.1.3. Modelul client - server**

Termenul de client - server provine de la metoda tradiţională de accesare a unui computer numit server de către computere aflate la distanţă. Serverele utilizează baze de date relaţionale în stocarea şi întreţnerea datelor între care există referinţe. Acest model este o combinaţie a trei tehnologii: sistemul relaţional de management al bazelor de date(DBMS), reţeaua şi interfaţa client(bazată pe GUI/PC).

În modelul client - server comunicarea ia în general forma unui mesaj de cerere prin care clientul solicită server-ului executarea unei anumite acţiuni. Server-ul execută cererea şi trimite răspunsul înapoi clientului. Pentru a putea îndeplini cererea, server-ul poate referi o sursă de informaţii(baze de date) să efectueze procesări asupra datelor, să controleze periferice sau să efectueze cereri adiţionale altor servere.

Procesarea este sincronă, clientul trimite cererea şi aşteaptă până când server-ul îi trimite răspunsul.

Server-ul face procesări care implică o latenţă mare şi se poate ca clientul să primească o excepţie de tip timeout, caz în care se folosesc două metode de procesare asincronă:

* Corelare: clientul face cererea şi server- ul răspunde imediat trimiţând un id de corelare, după care clientul mai face o cerere cu ace id ca să primească rezultatul final;
* Callback: clientul face cererea normal plasând tag-ul de comandă şi un url la care el ascultă rezultatele. Acesta ştie cum să asocieze rezultatele, bazat pe acel tag pe care server-ul îl trimite odată cu răspunsul. Acest procedeu este server-server deoarece browser-ul nu are un server web la care să asculte rezultatele procesării pe server.

Pentru paginile dinamice procedura este următoarea:

1. Se introduce adresa <http://localhost:8080/DisertatieWebApp/index.jsp> în browser

2. Browser-ul caută ip-ul pentru această adresă

3. Browser-ul face o cerere la această adresă pentru server-ul web al primei pagini

4. Cererea trece prin Internet şi soseşte la server-ul web

5. Server-ul web primind cererea, aduce pagina de pe hard disk

6. Cu pagina în memorie, server-ul web observă că fişierul încorporeză cod Java şi îl trimite interpretorului Java

7. Interpretorul Java execută codul primit

8. Dacă codul conţine şi formulări MySQL, trimite datele sistemului MySQL

9. MySQL returnează rezultatele înapoi inbterpretorului Java

10. Interpretorul Java returnează rezultatele server-ului web

11. Server-ul returnează pagina clientului web care o afişează

### **4.1.4. Site-uri web**

Noţiunea de site web reprezintă o grupă de pagini web multimedia accesibile în internet şi care sunt conectate între ele prin hyperlink-uri. În mod normal un site web este administrat de un web master, dar există şi alte posibilităţi.

Site- ul web se actualizează automat şi permanent pe baza unei baze de date:

* paginile sale se creează în mod dinamic şi automat în funcţie de acţiunea utilizatorului;
* site-ul web se creează şi este administrat de către utilizatorii lui.

Iniţial paginile erau statice, trimiteau mereu aceeaşi informaţie(text plus imagini). Ulterior acestea au devenit dianmice în sensul că în funcţie de anumite informaţii trimise de clientul browser, server-ul trimite diferite informaţii pentru aceeaşi pagină.

### **4.1.5. Hypertext**

Înainte de apariţia şi dezvoltarea calculatoarelor, stocarea informaţiilor se realiza fie în creierul uman, fie sub formă scrisă, ulterior, tipărită pe hârtie, în cărţi şi pe alte materiale.

Hypertext-ul este textul afişat pe display-ul unui computer sau alte electronice cu referinţe la alte texte la care cititorul poate avea acces imediat printr-un click de mouse. Hypertext-ul este folosit şi pentru a descrie tabele, imagini sau alte prezentări de conţinut cu hyperlink-uri. Hypertext-ul este conceptul care defineşte structura World Wide Web.

Un site web este alcătuit din mai multe pagini web create cu ajutorul limbajului de marcare HTML şi limbaje de programare cum ar fi Java fiind accesibile vizitatorului prin intermediul protocolului HTTP care transferă informaţia de la server la browser.

## **4.2. TEHNOLOGII FOLOSITE**

În continuare sunt prezentate tehnologiile şi limbajele folosite în realizarea aplicaţiei.

### **4.2.1. Android**

### Android este o platformă software și un sistem de operare pentru dispozitive mobile bazată pe nucleul Linux, dezvoltată inițial de compania Google, iar mai târziu de consorțiul comercial Open Handset Alliance. Android permite dezvoltatorilor să scrie cod gestionat în limbajul Java, controlând dispozitivul prin intermediul bibliotecilor Java dezvoltate de Google. Aplicațiile scrise în C și în alte limbaje pot fi compilate în cod mașină ARM și executate, dar acest model de dezvoltare nu este sprijinit oficial de către Google.

### **Componentele folosite în cadrul aplicaţiei Android :**

### **Activity (Activitate)**

### - reprezintă o interfaţă cu utilizatorul, fereastră sau formular;

### În cadrul aplicaţiei am folosit 8 activităţi :

### **- LoginActivity** – este activitatea principală creată pentru procesul de autentificare a utilizatorilor în cadrul aplicației.

### AndroidManifest.xml:

### *<application android:allowBackup="true" android:icon="@drawable/app\_icon" android:label="@string/app\_name" android:supportsRtl="true" android:theme="@style/Theme.AppCompat.DayNight.NoActionBar"> <activity android:name=".ui.LoginActivity"> <intent-filter> <action android:name="android.intent.action.MAIN"/> <category android:name="android.intent.category.LAUNCHER"/> </intent-filter> </activity>*

### **- UserLoggedActivity** – este activitatea care se deschide în urma autentificării, conține ecranul principal cu opțiunile oferite de aplicație.

### AndroidManifest.xml:

### *<activity android:name=".ui.UserLoggedActivity"/>*

### **PreluariListActivity** – conține o lista tuturor comenzilor de ridicat de la expeditor asociate utilizatorului curent. Această activitate este lansată din lista de opțiuni a ecranului deschis în urma autentificării, la selectarea opțiunii “Comenzi”.

### AndroidManifest.xml:

### *<activity android:name=".ui.PreluariListActivity"/>*

### - **DetaliiPreluareActivity** – conține detalii despre comanda selectată din lista de comenzi de preluat. Activitatea se deschide la click pe un item al listei de preluări asociată utilizatorului curent.

### AndroidManifest.xml:

### *<activity android:name=".ui.DetaliiPreluareActivity"/>*

### - **LivrariListActivity** – conține o lista tuturor comenzilor de ridicat de la expeditor asociate utilizatorului curent. Această activitate este lansată din lista de opțiuni a ecranului deschis în urma autentificării, la selectarea opțiunii “Livrari”.

### AndroidManifest.xml:

### *<activity android:name=".ui.LivrariListActivity"/>*

### - **DetaliiLivrareActivity** – conține detalii despre comanda selectată din lista de comenzi de livrat. Activitatea se deschide la click pe un item al listei de livrări asociată utilizatorului curent.

### AndroidManifest.xml:

### *<activity android:name=".ui.DetaliiLivrareActivity"/>*

### - **ScanActivity** – este activitatea care preia informațiile obținute în urma scanării QR code-urilor și se realizeză modificările aferente în baza de date.

### AndroidManifest.xml:

### *<activity android:name=".ui.ScanActivity"/>*

### - **ChangePasswordActivity** – prin intermediul acestei activitaţi utilizatorul curent își poate modifica parola curentă.

### AndroidManifest.xml:

### *<activity android:name=".ui.ChangePasswordActivity"/>*

### Intent (Intenție)

### - reprezintă o entitate folosită pentru a descrie o operaţiune care urmează să fie executată;

### - am folosit această componentă pentru a putea face posibilă deschiderea unei activitaţi.

### Exemplu :

### *Intent i =* ***new*** *Intent(getApplicationContext(), ScanActivity.****class****); i.putExtra(DisertatieAppConstants.****USER\_INTENT****,* ***mUserBean****); i.putExtra(DisertatieAppConstants.****COLET\_INTENT****,* ***mColetBean****); startActivity(i);*

### **4.2.2. Firebug**

Firebug este o extensie a browser-ului Firefox cu ajutorul căreia am făcut debug, am editat, monitorizat codul html, css, javascript şi care conşine şi multe alte instrumente, printre care o consolă javascript, posibilitatea de a monotoriza timpii de execuţie sau de încărcare pentru o pagină, imagini etc.

Pentru accesul la Firebug trebuie instalat browser-ul Firefox şi extensia Firebug.

### **4.2.3. Server-ul Apache**

Unul dintre cele mai utilizate servere de web, Apache Web Server este un efort de a oferi o alternative viabila si necomercială, în domeniul serverelor de web. Reușita acestui proiect este în mare măsură legată de fenomenele initiate prin oameni ca Linus sau Stalman, ce au avut ca rezultat produse extraordinare, puternice și eficiente cum ar fi Lunux, Emacs precum și toate pachetele software apărute sub licența GNU.

Crearea unui process de sine stătător – The Apache Project este rezultatul reunurii unui mare număr de voluntari, comunicând prin intermediul Internetului. Aceștia sunt cunoscuți sub numele de Apache Group. În plus, sute de utilizatori din întreaga lume au contribuit la proiect prin cele mai diverse mijloace, de la cod sursă până la documentație HTML.

Apache își are originea în ideile și codul aflat în cel mai popular server HTTP al timpului NCSA. În 1995 cel mai performant server de web era practice cel dezvoltat de NCSA, University of Illinois, de către Rob McCool. În momentul în care acesta a plecat și nu a mai continuat dezvoltarea serverului său, au început să apară o mulțime de programatori ce produceau versiuni modificate și îmbunătățite ale acestui server, Un grup de asemenea programatori au decis în februarie 1995 să se reuneasca într-un proiect de dezvoltare și îmbunătățire a serverykui initial. Pornind de la NCSA httpd 1.3 ei au construit, prin adăugarea celor mai bune patch-uri, versiunea Apache 0.6.2 care a fost lansată în aprilie 1995.

Configurarea serverului în cazul cel mai simplu impune doar modificarea, în httpd.conf, a numelui serverului precum și, în toate fișierele, a căilor (path) ce definesc locul de instalare a serverului precum și a documentelor html.

După configurarea serverului se impune pornirea acestuia. Există două moduri de rulare a serverului. Modul implicit și cel mai utilizat de rulare este modul daemon. În mod daemon acesta este pornit și ruleaza în background, activându-se de la sine de mai multe ori, asigurând astfel un bun timp de răspuns la cereri HTTP. Al doilea mod ar fi rularea serverului prin intermediul daemonului inetd. În acest caz, inetd are grijă de interceptarea unei cereri HTTP și pornirea unei copii a unui server de web pentru tratarea acelei cereri. Nu se recomandă acest mod de rulare în cazuri speciale.

Pornirea serverului (modul daemon) se rezumă la executarea binarului cu parametrii adecvați (în principal stabilirea căii fișierului primar de configurare httpd.conf) după care acesta va raporta orice erori sau probleme precum și accesele de web in cazul rulării, prin intermediul unor fișiere de log localizate în directorul logs/. În cazul unor erori sau porniri defectuoase se recomandă analizarea fișierului error-log din logs/.

Am folosit acest server local pentru a testa soft-ul fără a avea nevoie de configurări de reţea. Acest server permite aplicaţiilor server şi client să ruleze pe acelaşi calculator. Acesta este un host pe care utilizatorii îl folosesc pentru a accesa reţeaua proprie a computer-ului. Adresa pe care o foloseşte este 127.0.0.1.

### 

### **4.2.4 Sistemul de baze de date MySQL**

MySQL este un sistem de gestiune a bazelor de date. Acesta se bazează pe modelul relaţional. O bază de date este o modalitate de stocare a unor informaţii şi date pe un suport extern (un dispozitiv de stocare), cu posibilitatea extinderii uşoare şi a regăsirii rapide a acestora.

Un sistem de gestiune al bazelor de date reprezintă totalitatea programelor utilizate pentru crearea, interogarea şi întreţinerea unei baze de date. Include două categorii de module: module care sunt comune cu cele ale sistemelor de operare ale calculatoarelor şi module cu funcţii specifice bazei de date.

MySQL este un sistem foarte rapid şi robust de management al bazelor de date. Acesta foloseşte limbajul SQL, limbajul standard de interogare al bazelor de date. Deşi este folosit foarte des împreună cu limbajul de programare Java, cu MySQL se pot construi aplicaţii în orice limbaj major.

SQL este un limbaj neprocedural. SQL este ceea ce face posibil un sistem de gestiune al bazelor de date relaţionale SGBDR(Relational Database Management System + RDBMS). Ceea ce diferenţiază un SGBD de un SGBDR este faptul că ultimul asigură un limbaj de baze de date orientat pe mulţimi. Pentru majoritatea sistemelor de administrare a bazelor de date relaţionale, acest limbaj este SQL.

Structured Query Language (SQL - limbajul structura de interogare) este limbajul standard folosit pentru manipularea şi regăsirea datelor din aceste baze de date relaţionale. SQL a fost dezvoltat pentru a servi bazele de date relaţionale.

MySQL este un produs bazat pe relaţia client-server. MySQL este scris în C şi C++. Parser-ul său pentru SQL este scris în Yacc şi un analizor lexical mai puţin cunoscut numit sql\_lex.cc.

Administrarea MySQL se poate face din linie comandă sau folosind browser-ul şi accesând aplicaţia MySQL Workbench.

Cele mai uzuale operaţii cu bazele de date sunt:

* CREATE - crează o bază de date sau o tabelă
* DROP - şterge o bază de date sau o tabelă
* ALTER - modifică structura unei tabele
* INSERT - adaugă înregistrări într-o tabelă
* DELETE - şterge înregistrări dintr-o tabelă
* UPDATE - actualizează înregistrările dintr-o tabelă
* SELECT - interoghează o tabelă

În MySQL spaţiul alocat pe discul server-ului este în funcţtie de tipul de date. Câteva dintre tipurile de date folosite în bazele de date MySQL sunt: int, bigint, varchar, date, datetime etc.

Tipul de date întregi încep de la valori negative la valori pozitive. Dacă se adaugă opţiunea UNSIGNED, care este un modificator de coloană, nu vor mai fi valori negative ci vor începe de la 0.

Alţi modificatori sunt:

* AUTO\_INCREMENT funcţionează cu orice tip întreg.La fiecare rând nou adăugat în baza de date numărul asociat va fi incrementat;
* NULL înseamnă fără valoare (diferit de spaţiu sau zero);
* NOT NULL înseamnă că orice înregistrare va fi considerată ceva;
* PRIMARY KEY este identificatorul unic al unei înregistrări dintr-o tabelă şi care nu poate fi NULL.

MySQL Workbench este un instrument de bază de date vizuală pentru arhitecţi, dezvoltatori, şi DBA. MySQL Workbench furnizează modelare de date, dezvoltare SQL, instrumente de administrare şi configurare pentru, şi multe altele. MySQL Workbench Permite unui Dba sau unui dezvoltator să proiecteze vizual, modelul, să genereze, şi să gestioneze baze de date.

Acesta cuprinde toate uneltele necesare pentru crearea unui model de date complex precum şi unelte de inginerie inversă, şi caracteristici cheie pentru gestionarea schimbărilor şi crearea documentaţiei care în mod normal necesită atât de mult timp şi efort. MySQL Workbench este disponibil pe Windows, Linux şi Mac OS.

## **4.3. PROGRAMARE WEB**

### **4.3.1. HTML**

Limbajul în care sunt scrise paginile WEB se numște HTML. El este derivat din SGML, și este format din seturi de tag-uri inserate în text, care dau directive asupra modului în care să se formateze textul. În functie de posibilitățile hard ale sistemului pe care se vizualizeaza pagina, și de posibilitățile browserului, pagina va fi afișată cu mai mutle sau mai puține caracteristici de formatare (un browser în mod text nu va putea folosi fonturi de mărimi diferite).

HTML (HyperTextMakupLanguage) este utilizat în World Wide Web pentru descrierea hypertextelor. HTML este un limbaj de descriere, conţinând elemente ce permit construirea paginilor web.

Etichetele HTML sunt încradrate între paranteze <>, <tag> - eticheta de început şi </tag> - eticheta de sfarşit. Efectul corespunzător etichetei este aplicat textului dintre eticheta de început şi cea de sfarşit.

Există versiuni diferite de browsere şi versiuni diferite ale limbajului HTML, un tag recunoscut de un browser nu poate fi recunoscut de un browser diferit sau mai puţin recent.

Un document HTML are de obicei umătoarea componenţă:

1. Versiunea HTML a documentului

2. Secţiunea HEAD cu etichetele <head></head>

3. Secţiunea BODY cu etichetele <body></body> sau <frameset></frameset>

Toate paginile HTML încep şi se termină cu etichetele <html></html>. Head conţine titlul paginii între etichetele <title></title>, descrieri de tip <meta>, stiluri pentru formarea textului şi link-uri către fişiere externe.

Tipurile de marcare în limbajul HTML sunt:

* Marcare structurală. Exemplu <h1>text</h1>
* Marcare pentru prezentare. Descrie cum apare un text. Exemplu: <strong>Exemplu</strong>
* Marcare pentru hyperlink. Leagă părţi ale unui document cu alte documente. Exemplu: <ahref="link">Exemplu</a>

Tag-urile HTML indică efecte aplicate diferit părţilor de pagină în programele browser. Orice fişier html are următoarea structură:

<html>

<head>

<title>TITLUL</title>

</head>

<body>

CONŢINUT

</body>

</html>

Exemple de tag-uri:

* <I>...</I> - stabileşte stilul italic (înclinat);
* <B>...</B> - stabileşte stilul bold (îngroşat)ş
* <U>...</U> - stabileşte stilul underline (subliniat)ş
* <P>...</P> - pentru a începe un nou paragrafş
* <TABLE>...</TABLE> - pentru a insera un nou tabel
* <TITLE>...</TITLE> - pentru a intitula o pagină

OBSERVAŢIE

Putem introduce în corpul unui document HTML şi comentarii folosind tag-ul <!...->. Acest tag se poate insera oriunde în document, iar comentariul nu va fi afişat.

Exemplu: Textul poate fi **bolda**t, *italic* sau subliniat.

Codul pentru această propoziţie este: <p>Textul poate fi <b>boldat</b>, <i>italic</i> sau <u>subliniat</u></p>.

### **4.3.2. CSS**

CSS (Cascading Style Sheets) este un standard pentru formatarea elementelor unui document HTML. Stilurile se pot atașa elementelor HTML prin intermediul unor fișiere externe sau în cadrul documentului, prin elementul <style> și/sau atributul style. CSS se poate utiliza și pentru formatarea elementelor XHTML, XML și SSVGL.

CSS este limbaj (style language) care definește ”layout-ul” pentru documentele HTML. CSS acopera culori, font-uri, margini (borders), linii, înălțime, lățime, imagini de fundal, poziții avansate și multe alte opțiuni. CSS oferă mai multe opțiuni, este mai sofisticat. În plus, este suportat de toate browserele actuale.

În documentele W3C, CSS nu este definit ca un nou limbaj, ci ca un mecanism care permite formatarea documentului HTML. De la culoarea literelor şi a background-ului până şi la poziţionarea elementelor de pe o pagină web, totul este stilizat în CSS. Stilurile pe o pagină pot fi încorporate în pagina respectivă sau pot fi chemate din fişiere externe, fişiere CSS.

Acestea pot fi definite în partea de head a documentului html, sau pot fi definite în fişierul css extern care este apelat tot în secţiunea de head a paginii. În documentul HTML codul CSS poatre fi introdus în mai multe moduri:

* Codul CSS este prezent în pagina web, iar efectul său se aplică asupra întregului document;

**Cod:**

<html>

<head>

<style>

b {color:red;}

i{color:blue;}

</style>

</head>

<body>

<p align=center><b>Execmplu CSS boldat</b></p>

<p><i>Exemplu CSS italic</i></p>

</body>

</html>

Consecinţă: Orice text cuprins între <b></b> va fi afişat cu roşu oriunde s-ar afla în document. Orice text afişat cu italice va fi de culoare albastră.

* Folosind metoda fişierelor CSS externe, toate paginile (X)HTML vor folosi acelaşi fişier de stil. Pentru a obţine un aspect în toate paginile trebuie modificat un singur fişier, şi anume ce de stil (.css), şi efectul se va observa pe toate paginile (X)HTML, ce folosesc acel fişier.

CSS este creat pentru a face separarea între conţinutul unui document şi forma de prezentare a documentului, incluzând elemente ca fonturi, culori etc.

Selectorul este elementul (X)HTML pe care am dorit să-l stilizez. Proprietatea este chiar titlul (numele) proprietăţii respective, iar valoarea reprezintă stilul pe care îl aplici proprietăţii.

selector { proprietate:valoare}

Fiecare selector poate avea multiple proprietăţi şi fiecare proprietate din acel selector are valori independente. Proprietatea este separată de valoare cu semnul ":". Toate proprietăţile împreună cu valorile lor, aparţinând aceluiaşi selector sunt cuprinse între "{}".

Multiplele valori din aceeaşi proprietate sunt separate prin virgulă "," şi dacă o valoare conţine mai mult de un cuvânt acestea se cuprind între ghilimele '"'.

Exemplu:

table td{ font-size: 15px; font-weight: normal;text-align:center;

color: #556E6B;font-size:12px;line-height: 16px;}

În acest exemplu prin CSS se oferă proprietăţi tuturor coloanelor dintr-un tabel, proprietăţi cum ar fi: fontul, marimea scrisului, spaţierea, culoarea textului, grosimea textului, aranjarea în pagină.

Exemplu de selectori CSS:

* .class - Exemplu: .intro - selectează toate elementele cu clasa ".intro";
* #id - Exemplu: #nume - selectează toate elementele cu id-ul "#nume";
* \* - selectează toate elementele;
* element - Exemplu: p - selectează toate elementele de tip paragraf cu etichetă <p>
* element, element - Exemplu: div, p - selectează toate elementele p şi toate elementele div;
* element element - Exemplu: div p - selectează toate elementele p din interiorul unui element div;
* element > element - Exemplu: div > p - selectează toate elementele p care au ca părinte un element div;
* element + element - Exemplu div + p - selectează toate elementele p plasate imediat după un element div;
* [atribut] - Exemplu: [target] - selectează toate elementele cu atributul "target";
* ;link - Exemplu: a:link - selectează toate link-urile nevizitate;
* ;visited - Exemplu: a:visited - selectează toate link-urile vizitate;
* ;hover - Exemplu: a:hover - selectează toate link-urile asupra cărora este mouse-ul.

#### 4.3.2.1. Diferență între CSS și HTML

HTML este folosit pentru a structura conținutul în timp ce CSS este folosit pentru a formata conținutul.

În perioada de început a web-ului, HTML era folosit numai pentru structura textului. Textul se putea marca cu taguri precum <h1> și <p> pentru a marca titlul sau un paragraf. Odată cu creșterea popularitpții web-ului designerii au început să caute diferite posibilități de a adăuga layout documentelor online. Pentru a răspunde acestor cerințe, producatorii de browsere (în acea vreme Microsoft si Netscape) au inventat noi taguri HTML precum <font> care diferă față de tagurile originale HTML prin faptul că definesc layout-ul si nu structura.

Acest lucru a dus și la o situație unde tagurile originale de structură ca <table> să fie folosite necorespunzător pe pagini de layout (to layour pages). Multe taguri noi de layout precum <blink> erau recunoscute numai de unele browsere. O formulă comună ce aparea pe site-uri era ”Aveți nevoie de browserul X pentru a vedea această pagină”. CSS a fost inventat pentru a remedia aceasta sitație, furnizându-le designerilor facilități sofisticate pentru editarea layoutului, suportate de toate browserele.

În acelși timp, separarea site-urilor de prezentarea pentru documente de conținutul documentelor ușureaza foarte mult intreținerea lor.

#### 4.3.2.2. Avantajele CSS-ului

CSS a reprezentat un element revoluționar în lumea web-designului iar beneficiile concrete includ:

* Controlarea layoutului documentelor dintr-o singură pagina de stiluri
* Control mai exact al layoutului
* Aplicare de layouturi diferite pentru tipuri media diferite (ecran, printare, etc)

### **4.3.3. SERVLET**

Un servlet este o clasă Java care prelucrează cererile clienţilor şi construieşte dinamic pagina HTML de răspuns. Servlet-urile sunt componente ale aplicaţiilor server, independente de platformă, care extind dinamic serverele care au suport Java integrat. Ele sunt independente şi de protocol, asigurând un cadru general pentru servicii pe baza modelului cerere-răspuns. Acest binecunoscut model este des folosit la programarea sistemelor distribuite, începând cu apeluri de proceduri la distanţă şi terminând cu protocolul HTTP pentru cereri către servere Web. Cu ajutorul servlet-urilor, aşadar, se extinde funcţionalitatea unei aplicaţii de tip server informaţional (nu neapărat server HTTP), un prim domeniu de aplicare fiind, bineînţeles, extinderea serverelor Web.

Servlet-urile pot fi considerate echivalentul applet-urilor pe partea de server, ele fiind asemănătoare din multe puncte de vedere. Însă, principala caracteristică a applet-urilor, interfaţa grafică utilizator, lipseşte din servlet-uri, ele neavând nevoie de aşa ceva din moment ce rulează în interiorul serverelor. În rest, ele se aseamănă mult, un servlet fiind şi el o componentă de aplicaţie, scrisă în Java, care poate fi transferată la un sistem care are nevoie de ea.

Servlet-urile reprezintă o alternativă oferită de Java pentru rezolvarea problemelor legate de timp apărute odată cu programarea Common Gateway Interface - CGI. Acestea facilitează dezvoltarea unor module care permit servere-lor Web să se conecteze şi să prelucreze informaţia în mod dinamic, adică să ruleze aplicaţii Web şi nu doar să transfere documente statice. Soluţia Java, menţine executabilul persistent pe server, între cererile clienţilor, spre deosebire de CGI unde fiecare cerere client lansează un nou proces pe server (ceea ce duce rapid la epuizarea resurselor serverului Web, adică la creşterea timpului).

Ciclul de viaţă al servlet-urilor este o proprietate specifică a acestora. Servlet-urile se încarcă în mod dinamic, serverele oferind facilităţi de administrare a încărcării şi a iniţializării acestora. Există şi posibilitatea, deloc de neglijat, de a specifica încărcarea anumitor servlet-uri la lansarea în execuţie a serverului. Odată încărcate, acestea devin parte integrantă a serverului. Procesul de încărcare este transparent pentru utilizator, acesta trebuind să specifice doar locaţia servlet-ului (local sau la distanţă), numele clasei ce conţine servlet-ul şi numele sub care acesta va fi cunoscut de către server (alias).

Caracteristicile principale ale servlet-urilor se referă la următoarele:

* clasele şi metodele necesare pentru a defini şi utiliza un servlet sunt încapsulate în pachete Java;
* tehnologia se utilizează pentru a dezvolta soluţii bazate pe Web: accesul securizat la pagini Web, asigurarea interacţiunii cu bazele de date, generarea dinamică a paginilor HTML, manipularea informaţiilor care identifică unic un client pe parcursul uneia sau a mai multor sesiuni;
* se poate crea câte un servlet pentru fiecare funcţie din paginile Web (conectare, înregistrare, actualizare etc.) sau unul singur care să gestioneze toate tranzacţiile din paginile Web respective, în mod dinamic;
* tehnologia este o soluţie optimă pentru aplicaţiile care utilizează intensiv bazele de date (serverul se ocupă de accesul la date iar clienţii formulează cererile de regăsire). Partea de cod se scrie o singură dată şi se stochează rezident, o singură dată, pe server. La actualizarea codului se va face o singură înlocuire, pe server, şi nu la fiecare utilizator în parte;
* la iniţiere se pot deschide conexiuni la baze de date care devin astfel rezidente între apelurile clienţilor;
* comunicarea client-server se realizează în mai mulţi paşi, astfel: clientul formulează şi trimite către server o cerere Web; serverul o direcţionează către servlet pentru a fi procesată (ceea ce implică de multe ori şi accesul la o bază de date); răspunsul (sub formă de pagini HTML, imagini etc.) este returnat serverului şi apoi clientului care a formulat cererea;
* un servlet poate fi apelat dintr-o pagină HTML sau dintr-un applet;
* principalele situaţii de utilizare a tehnologiei se referă la generarea paginilor Web dinamice şi la realizarea aplicaţiilor multi-nivel (multi-tier) cu JDBC. În acest ultim caz, servlet-ul poate accesa o varietate de baze de date prin intermediul JDBC şi poate realiza, parţial sau total, interfaţa cu utilizatorul prin pagini Web dinamice.

Paginile JSP şi componentele servlet sunt funcţional interschimbabile, dar unele aspecte de programare se rezolvă mai simplu într-una sau alta din tehnologii. În cazul în care cererea clientului necesită includerea unei mari părţi de cod HTML în pagina de răspuns, atunci paginile JSP sunt mai simplu de folosit. În schimb, dacă respectiva cerere necesită multiple operaţii de prelucrare a datelor, este indicată folosirea componentelor servlet.

### **4.3.4. JAVA SERVER PAGES (JSP)**

JavaServer Pages este tehnologia platformei Java pentru construirea de aplicaţii ce cuprind pagini Web cu conţinut dinamic precum HTML, DHTML, XHTML şi XML. Sun a încercat să depăşească limitările soluţiilor actuale pentru generarea de pagini cu conţinut dinamic prin dezvoltarea unei tehnologii care:

* să funcţioneze pe orice server Web sau de aplicaţii;
* să separe logica ce stă în spatele aplicaţiei de aspectul paginii;
* să permită dezvoltare şi testare rapidă;
* să simplifice procesul de dezvoltare de aplicaţii interactive Web.

Tehnologia JSP a fost creată să satisfacă aceste cerinţe, fiind rezultatul unei cooperări la nivelul industriei software dintre producătorii de servere Web, servere de aplicaţii, sisteme tranzacţionale şi unelte de dezvoltare.

Prin tehnologia JavaServer Pages, proiectanţii de pagini folosesc tag-uri obişnuite HTML sau XML pentru formatarea rezultatului şi tag-uri JSP sau scriplet-uri pentru generarea conţinutului dinamic al paginii. Logica ce stă în spatele generării conţinutului este cuprinsă în tag-uri şi componente JavaBean, legătura dintre acestea făcându-se în scriplet-uri şi totul fiind executat pe server. Astfel, proiectanţii de pagini sau Web master-ii pot edita şi lucra cu pagini JSP fără a afecta generarea conţinutului dinamic.

Pe partea de server, un engine (motor) JSP interpretează scriplet-urile şi tag-urile JSP, generează conţinutul cerut (accesând componente JavaBean, baze de date folosind JDBC sau prin includerea de fişiere) şi trimite rezultatele înapoi sub forma unei pagini HTML (sau XML) către browser.

Tehnologia JSP permite reutilizarea componentelor precum JavaBeans, Enterprise JavaBeans sau a tag-urilor atât independent, cât şi în cadrul unor unelte interactive de dezvoltare a componentelor şi paginilor de Web. Creatorii de pagini Web nu sunt întotdeauna programatori familiarizaţi cu limbaje de scripting. JSP încapsulează funcţionalităţile necesare pentru crearea de conţinut dinamic în tag-uri de tip XML specifice JSP. Tag-urile JSP standard pot accesa şi instanţia componente JavaBean, pot seta sau obţine atribute ale bean-urilor, pot face download la applet-uri şi pot executa funcţii ce ar fi dificil de implementat. Tehnologia JSP este extensibilă prin dezvoltarea de biblioteci de tag-uri definite de utilizator. Cu timpul vor fi create biblioteci proprii de tag-uri pentru funcţiile folosite cel mai frecvent.

Tehnologia JSP este complet independentă de platformă atât în ceea ce priveşte paginile Web dinamice, cât şi serverele de Web şi componentele acestora. Aceasta este explicabil deoarece limbajul de scripting pentru paginile JSP se bazează pe Java şi în special pe modul de manipulare a obiectelor în acest limbaj.

O pagină JSP (\*.jsp) este o pagină HTML sau XML ce cuprinde elemente adiţionale (tag-uri, declaraţii, scriplet-uri) pe care motorul JSP le procesează şi le elimină returnând o pagină standard HTML/XML. Ea corespunde unui document ce descrie procesarea unei cereri pentru a crea un răspuns.

O pagină JSP cuprinde în structura sa:

* cod HTML/XML standard - cod ce rămâne neinterpretat de motorul JSP;
* directive JSP - directive ce furnizează informaţii globale independente conceptual de o anumită cerere adresată paginii JSP;
* tag-uri JSP - spre deosebire de directive, tag-urile depind de fiecare cerere în parte adresată paginii JSP;
* elemente de scripting - acestea putând fi: declaraţii, scriplet-uri şi expresii.

### 

### **4.3.5. JAVASCRIPT**

JavaScript este un limbaj de programare orientat obiect bazat pe conceptul prototipurilor. Este folosit mai ales pentru introducerea unor funcționalități în paginile web, codul Javascript din aceste pagini fiind rulat de către browser.

Limbajul este binecunoscut pentru folosirea sa în construirea site-urilor web, dar este folosit și pentru accesul la obiecte încastrate (embedded objects) în alte aplicații.

JavaScript este util pentru a verifica validitatea informaţiilor introduse într-un formular înainte ca datele să fie trimise către server. O menţiune importantă: programele care rulează pe calculatorul utilizatorului sunt numite aplicaţii client-side (aflate pe partea de client) şi programele care rulează pe server (inclusiv CGI-urile) sunt numite aplicaţii server-side (aflate pe partea de server).

A fost dezvoltat inițial de către Brendan Eich de la Netscape Communications Corporation sub numele de Mocha, apoi LiveScript, și denumit în final JavaScript.

Cea mai des întâlnită utilizare a JavaScript este în scriptarea paginilor web. Programatorii web pot îngloba în paginile HTML script-uri pentru diverse activități cum ar fi verificarea datelor introduse de utilizatori sau crearea de meniuri și alte efecte animate.

Un script JavaScript este un program inclus într-o pagină HTML. Deoarece este încadrat în tag-ul <script>, textul script-ului nu apare pe ecran, dar este rulat şi interpretat de către browser. Tag-ul <script> este prezent cel mai frecvent în secţiunea <head> a paginii HTML, deşi se poate pune şi în secţiunea <body>. De obicei script-urile care urmează să afişeze mesaje pe ecran sunt scrise în secţiunea <body>.

Cu ajutorul limbajului JavaScript pot fi gestionate mai multe evenimente diferite. Cele mai importante sunt: click, change, focus, load, mouseover, select, submit, unload. Pentru a sesiza aceste evenimente, browser-ul trebuie să ofere nişte funcţii speciale care să indice momentul în care a re loc respectivul eveniment. Handler-ele corespunzătoare evenimentelor prezentate anterior sunt: onMouseOver, onBlur, onSelect, onLoad, onFocus, onSubmit, onClick, onChange, onUnload.

JavaScript organizează toate elementele unei pagini web într-o iererhie, fiecare element fiind văzut ca un obiect, iar obiectul având proprietăţi şi metode.

La baza limbajului stau trei blocuri funcţionale şi anume: valor (tipuri de date suportate de JavaScript), obiecte (colecţii de proprietaţi care pot fi apelate cu un singur nume), funcţii (care pot fi executate de o aplicaţie; funcţiile unui obiect se numesc metode).

Browserele rețin în memorie o reprezentare a unei pagini web sub forma unui arbore de obiecte și pun la dispoziție aceste obiecte scipt-urilor JavaScript, care le pot citi și manipula.

Arborele de obiecte poartă numele de Document Object Model sau DOM. Există un standard W3C pentru DOM-ul pe care trebuie sa il pună la dispoziție un browser, ceea ce oferă premiza scrierii de scipt-uri portabile, care să funționeze pe toate browserele.

În practică, însă, standardul W3C precum DOM este incomplet implementat. Deși tendința browserelor este de a se alinia standardului W3C, unele din acestea încă prezintă incompatibilități majore, cum este cazul Internet Explorer.

O tehnică de construire a paginilor web tot mai îmtâlnită în ultimul timp este AJAX, abreviere de la ”Asynchronous JavaScript and XML”.

Această tehnică constă în executarea de cereri HTTP în fundal, fără a reîncărca toată pagina web și actualizarea numai anumitor porțiuni ale paginii prin manipularea DOM-ului paginii.

Tehnica AJAX permite construirea unor interfețe web cu timp de răspuns mic. Întrucât operația (costisitoare ca timp) de incărcare a unei pagini HTML complete este în mare parte eliminată.

#### 4.3.5.1. Ajax

Numele este prescurtarea de la Asynchronous JavaScript And XML. În principal Ajax nu este un limbaj de programare ci mai degrabă un atu în realizarea unui website. Ajax face posibilă comunicarae cu server-ul fără a fi nevoie să încarce o nouă pagină.

Ajax nu este o tehnologie în sine, ci cuprinde mai multe tehnologii, ficare fiind puternică în felul ei dar adunate într-o metodă şi mai puternică. Ajax încorporează:

* reprezentări standard folosind HTML şi CSS;
* afişare dinamică folosind DOM (Document Object Model);
* manipulări de date folosinf XML;
* preluare de date folosind protocolul asincrom XMLHttpRequest;
* JavaScript pentru a lega totul împreună.

Modelul clasic al unei aplicaţii web funcţioneză în felul următor: majoritatea acţiunilor utilizatorului declanşează un HTTP Request înapoi la server-ul web. Server-ul face procesări, colectează date, formatează numere, comunică cu diferite subsisteme, apoi returnează text, fragment html, sau chiar informaţie serializată JSON, care cu ajutorul JavaScript este procesată şi afişată pe pagină clientului. Acesta este un model adaptat după modelul original al web-ului ca un mediu hypertext.

Datorită faptului că generează local pagina HTML şi downloadează doar script-ul JavaScript şi datele, paginile web Ajax pot părea că se încarcă relativ repede. De asemenea, mulţumită funcţionalităţii "load on demand" a conţinutului, unele pagini web încarcă stub-uri ale event handler-elor, iar apoi rulează funcţiile "on the fly". Această tehnică reduce considerabil lăţinmea de bandă folosită pentru aplicaţii web. În plus clientul de Ajax împarte workload-ul cu server-ul astfel încât încărcarea acestuia din urmă este redusă. Un alt beneficiu de ordin mai puţin programatic este că Ajax tinde să încurajeze programatorii să separe clar metodele, funcţiile şi format-urile folosite în diferite aspecte ale transferului de informaţii pe web.

Exemplu de script:

<script type=*"text/javascript"*>

showColetInfos = **function**(awb) {

$.get("infoAWB.jsp", {

awb : awb,

}).done(**function**(data) {

$("#afisare\_rezultat").html(data);

});

$("#afisare\_rezultat").show();

}

$(**function**() {

$("#bt\_Cautare").click(**function**() {

showColetInfos($("#search\_awb").val());

});

});

</script>

## **4.4. IMPLEMENTAREA FIZICĂ A BAZEI DE DATE**

Exemplu de comenzi pentru crearea tabelelor:

CREATE TABLE IF NOT EXISTS `disertatie\_db`.`comanda` (

`comanda\_id` INT NOT NULL AUTO\_INCREMENT,

`comanda\_nr\_colete` INT NOT NULL,

`comanda\_greutate` INT NOT NULL,

`comanda\_plata` VARCHAR(30) NOT NULL,

`comanda\_data\_comanda` DATE NOT NULL,

`comanda\_observatii` VARCHAR(100) NULL,

`comanda\_data\_expediere` DATETIME NULL,

`comanda\_data\_preluare` DATETIME NULL,

`comanda\_exp\_id` INT NOT NULL,

`comanda\_dest\_id` INT NOT NULL,

PRIMARY KEY (`comanda\_id`),

INDEX `fk\_comanda\_exp\_idx` (`comanda\_exp\_id` ASC),

INDEX `fk\_comanda\_dest\_idx` (`comanda\_dest\_id` ASC),

CONSTRAINT `fk\_comanda\_exp`

FOREIGN KEY (`comanda\_exp\_id`)

REFERENCES `disertatie\_db`.`client` (`client\_id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_comanda\_dest`

FOREIGN KEY (`comanda\_dest\_id`)

REFERENCES `disertatie\_db`.`client` (`client\_id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

La crearea tabelelor s-a folosit engine-ul InnoDB care oferă suportul pentru utilizarea cheilor străine şi a standardului ACID. Acest standard reprezintă un set de proprietăţi care garantează că tranzacţiile bazei de date sunt procesate.

## **4.5. IMPLEMENTAREA FIZICĂ A APLICAŢIEI**

### **4.5.1. APLICAȚIA ANDROID**

În aplicația Android am creat o serie de ecrane, denumite în rândul dezvoltatorilor Android activități, care se apelează între ele prin intermediul unor Intent-uri.

Astfel aplicația se deschide cu activitatea de autentificare *LoginActivity.* Activitatea de autentificare cuprinde două câmpuri de tip EditText pentru inserarea username-ului și a parolei și un buton pentru confirmarea datelor și verificarea corectitudinii prin consultatae bazei de date. În cadrul acestei activități se crează conexiunea la server-ul de DB MySQL prin apelul unui fișier php via HTTP request care interogheză baza de date și returnează rezultatul în aplicație sub formă de obiect de tip JSON. În urma interogării bazei de date se verifică corectitudinea parametrilor primiți din interfața aplicației și se permite sau nu accesul după validarea datelor.

După realizarea autentificării în aplicație va fi lansată activitatea *UserLogged ,* de unde se pot vizualiza comenzile de preluat și comenzile de livrat asignate utilizatorului utilizatorului curent. În plus din același ecran se lansarea modulul de scanare, iar la o scanare validă este returnat rezultatul care va fi trimis prin intermediul unui intent către activitatea de prelucrare a informației scanate. Ultimele două opțiuni din activitatea *UserLogged* sunt opțiuni de schimbare a parolei și de ieșire din cont prin delogare. Fiecare dintre opțiunile prezentate mai sus sunt reprezentate prin cate un buton pentru a putea fi lansate.

Prin accesarea activității *PreluariListActivity* se poate vizualiza lista de comenzi asignată utilizatorului curent. Pentru afișarea listei s-a folosit un obiect de tip ListView, asupra caruia s-a aplicat un ArrayAdapter pentru a seta stilul dorit listei. În urma particularizării listei vom putea observa cî fiecare item conține caâte un obiect grafic de tip ImageView și un TextView. La evenimentul de clik pe un item al listei se la lansa activitatea *DetaliiPreluareActivity* prin intermediul unui intent care va transmite ca parametru identificatorul comenzii selectate.

Activitatea DetaliiPreluareActivityare rolul de a afișa informațiile referitoare la comanda selectată prin utilizarea mai multe obiecte grafice de tip TextView. În plus în aceeași activitate se poate remarca un CheckBox și un Button care oferă posibilitatea actualizării comenzii la bifarea check box-ului și salvarea acesteia în baza de date la click pe buton.

Asemeni activităților de vizualizare și actualizare a comenzilor de preluat este structurată și secțiunea comenzilor de livrat prin intermediul activităților *LivrariListActivity și DetaliiLivrareActivity.*

Modulul de scanare este lansat din activitatea *UserLoggedActivity,* prin apelarea librăriei ZXing care realizează scanarea propriu-zisă returnând rezurtatul. În cazul unei scanări reușite se lansează activitatea *ScanActivity* care primește prin intent la deschidere conținutul QR code–ului scanat. Activitatea afișează detaliile aferente AWB-ului conținut de QR code, obținute din baza de date și oferă posibilitatea actualizării statusului coletului scanat printr-un click pe butonul de update.

La accesarea activității *ChangePasswordActivity* se va deschide un ecran ce cuprinde trei elemente grafice de tip EditText pentru inserarea vechii parole, a noii parole precum și confirmarea noii parole în scopul schimbării parolei aferente contului utilizatorului curent. Pentru confirmarea datelor inserate utilizatorul are la dispoziție butonul “Change Password” care va realiza modificările în baza de date atât timp cât informațiile furnizate sunt corecte, altfel va fie semnalat prin intermediul unui Toast, un mesaj de avertizare specific android-ului.

Ultima opțiune din meniul principal nu este reprezentată printr-o activitate precum celelalte opțiuni, doar se distruge activitatea current și se redirectează către activitatea de autentificare.

### **4.5.2. APLICAȚIA WEB**

În realizarea paginilor web am structurat pagina în funcţie de tipul acesteia: JSP, HTML, CSS, JS etc. Conţinutul diferă de la pagină la pagină, în schimb ce header-ul se regăseşte la fel pe toate paginile.

Fiind o aplicaţie care lucrează cu o bază de date am creat o clasă Java prin care am realizat conexiunea la baza de date. Această clasă este inclusă şi apelată în toate fişierele de tip JSP sau Servlet, care interacţionează cu baza de date, prin crearea unei noi instanţe a acestei clase.

În această clasă am realizat legătura la baza de date astfel:

static String *URL* = "jdbc:mysql://localhost:3306/disertatie\_db";

static String *USER* = "lchirvase";

static String *PSW* = "test";

Class.*forName*("com.mysql.jdbc.Driver");

*conn* = DriverManager.*getConnection*(*URL*, *USER*, *PSW*);

Stilizarea structurilor html existente pe fiecare pagină s-a realizat în fişierele "disertatie\_syle.css" și "styles.css", cu ajutorul limbajului CSS. Acestea au fost incluse în folder-ul "css" şi incluse pe paginile create prin specificarea în head-ul paginii:

<link href=*"css/* disertatie\_syle.css*"* rel=*"stylesheet"* type=*"text/css"*/>

<link href=*"css/styles.css"* rel=*"stylesheet"* type=*"text/css"*/>

Prima pagină a aplicaţiei este *index.jsp*. În acestă pagină utilizatorul are ca opţiuni accesarea secţiunilor "Cheamă curier" și “Verifică AWB”. La click pe butonul "Cheamă curier" se accesează *cheamaCurier.jsp,* secțiune în care utilizatorul poate completa un formular pentru efectuarea unei comenzi către firma de curierat. Cea de-a doua secțiune specificată mai sus destinată oricărui utilizator fără autentificare este cea care va accesa “verificaAWB.jsp”, de unde se poate verifica statusul coletelor din cadrul unei comenzi efectuate anterior.

*<form name="CheamaCurierForm" action="CheamaCurierServlet" method="post">*

Pe lângă cele două secțiuni publice destinate oricăror tipuri de utilizatori, pe prima pagină se distinge și o secțiune de autentificare, care vizează personalul firmei de curierat. După logare este salvat în sesiune numele utilizatorului, după care se face interogarea. Pentru realizarea acestui lucru am creat un obiect de tip *UserBean*, după care am apelat metoda login() de verificare propriu-zisă a autenticităţii utilizatorului, care se regăseşte în clasa *UserDAO*. Numele utilizatorului apare în bara de meniuri, de unde acesta are posibilitatea de a schimba parola contului, fapt realizat prin intermediul clasei *EditProfileServlet*, sau de a ieşi din aplicaţie, prin operaţia de delogare realizată în clasa *LogoutServlet*.

Pentru accesarea celorlalte pagini din bara de meniuri au fost incluse link-uri către pagina dorită:

<a href=*'#SchimbareParola'* onclick="includeFile('#SchimbareParola')">

<div id=*"SchimbareParola"* class=*"P"*><%@ include file = *"schimbareParola.jsp"* %> </div>

În secţiunea "Comenzi" se disting trei subcategorii: “Comenzi nepreluate”, “Comenzi ]n curs de preluare” și “Comenzi preluate”. Secțiunea “Comenzi nepreluate” este accesată prin intermediul fişierului *comenziNepreluate.jsp,* care va ajuta operatorul să vizualizeze toate comenzile nepreluate și să aibă posibilitatea de a asigna un curier la fiecare dintre acestea. Dupa asignarea curierului comanda va dispărea din această secțiune urmând a fi regăsită în secțiunea comenzi în curs de preluare. Vizualizarea categoriei “Comenzi în curs de preluare” se realizează prin accesarea fișierului *comenziInCursDePreluare.jsp*. Cea de-a treia categorie este “Comenzi preluate” lansată cu ajutorul fișierului *comenziPreluate.jsp,* de unde operatorul are posibilitatea generării codului unic al coletelor (AWB) și a talonului care se va aplica asupra fiecărui colet în parte.

*<li class='has-sub'><a href='#Comenzi'><span>Comenzi</span></a>*

*<ul>*

*<li><a href="#ComenziNepreluate" onclick="includeFile('#ComenziNepreluate');">*

*<span>Comenzi nepreluate</span></a></li>*

*<li><a href="#ComenziInCursDePreluare" onclick="includeFile('#ComenziInCursDePreluare');">*

*<span>Comenzi in curs de preluare</span></a></li>*

*<li class='last'>*

*<li><a href="#ComenziPreluate" onclick="includeFile('#ComenziPreluate');">*

*<span>Comenzi preluate</span></a></li>*

*</ul></li>*

<div id=*"ComenziNepreluate"* class=*"P"*><%@include file=*"comenziNepreluate.jsp"*%></div>

<div id=*"ComenziInCursDePreluare"* class=*"P"*><%@ include file=*"comenziInCursDePreluare.jsp"*%></div>

<div id=*"ComenziPreluate"* class=*"P"*><%@ include file=*"comenziPreluate.jsp"*%></div>

# CAPITOLUL 5

# TESTARE ŞI EVALUARE

**Testarea aplicaţiei poate fi numită faza finală a procesului de** [dezvoltare web](http://blog.web-design-site.ro/tag/dezvoltare-web/) **pentru orice aplicaţie.** Aceasta este faza în care site-ul sau orice alt software implicat în crearea acestuia sunt testate pentru prezenţa de erori si bug-uri. Testarea nu numai că detecteaza bug-urile, dar, ajută utilizatorul să ia cunostință de  acestea, astfel încât să nu se repete în dezvoltarea altor site-uri în viitor.

Testarea este o fază în care **website-ul finalizat** este trimis catre auto-operare echipei de testare pentru evaluare (în cazul de față o singură persoană). Site-ul este testat pe baza unor tehnici diversificate si apoi evaluat. Procesul nu numai că asigură inexistența de bug-uri dar face site-ul mai sofisticat pentru clientii care il viziteaza. Procesul, îmbunătățește în mod evident standardul proiectului dumneavoastră.

Desi procesul de testare diferă pentru fiecare produs care este testat, procesul general de testare al unui website ar implica testarea site-ului din punct de vedere al proiectarii, link-urile intrerupte din website, analiza de continut pentru greseli gramaticale, de aliniere a textului, asezarea de ansamblu in pagina, verificarea fiabilității și functionalității script-urilor de programare backend și așa mai departe. Deși aceste teste par foarte simple, de fapt, necesita acordarea unei atentii sporite pentru că aceste detalii minore, atunci când sunt perseverente, ar putea zădărnici activitatea clienților pe website.

Testarea este de fapt un proces in lanţ având în vedere ca responsabilitatea sa, nu se termină doar prin specificarea de erori și bug-uri,  Greșelile care sunt notate  trebuiesc imediat rectificate.  Site-ul corectat, va fi din nou testat pentru aceleași erori sau pentru prezența unor erori noi. **Odată dovedită lipsa de erori, website-ul se spune ca este gata de lansare.**

Este adevarat ca mulți dintre cei care încep noi proiecte privesc testarea ca fiind o fază care poate fi omisă pentru moment, din diverse motive cum ar fi economia de timp sau lipsa de fonduri. Dar ei nu realizează că aceste erori minore și greșeli în site ar putea provoca probleme fatale in viitor. [Website-ul](http://blog.web-design-site.ro/tag/website/) **ar trebui, fără doar și poate să fie testat.**

## **5.1. TESTAREA**

### **5.1.1. Strategii de testare**

Înainte de scrierea testelor este preferabilă stabilirea unor detalii. Testele vor fi clasificate după importanţa şi frecvenţa de utilizare, astfel:

|  |  |  |  |
| --- | --- | --- | --- |
| Frecvența/importanta | Strategică | Importanța | Slabă |
| Mare | 1 | 2 | 3 |
| Medie | 1 | 3 | 4 |
| Mică | 2 | 3 | 4 |

**În continuare voi împărţi diferitele tipuri de teste în funcție de coeficientul de utilizare.**

* Testele cu coeficient 1 sunt critice: Un număr mare de teste este necesar pentru asigurarea unei bune funcționari în toate circumstanțele.
* Testele cu coeficient 2 sunt importante: Nefuncționarea este neplăcută, dar nu produce situații blocante.
* Testele cu coeficient 3: Nefuncționarea acestora nu deranjează utilizatorul
* Testele cu coeficient 4 nu sunt deloc importante.

Strategiile de testare se concentrează pe funcţionalitatea şi utilizabilitatea produsului.

Ca şi metode de testare am folosit metode statice, analiza programului înainte de a fi lansat în execuţie, independent de datele de intrare, metode dinamice, care constau în execuţia programului. Ca şi metode statice am realizat testarea specificaţiilor şi examinarea codului.

La examinarea codului, la compilare programul nu a fost refuzat din cauza nerespectării criteriilor de corectitudine, nu au existat variabile neiniţializate astfel nefiind probleme la execuţie.

Interfaţa aplicaţiei afişează acelaşi aspect al paginii pe diferite rezoluţii, diferitre sisteme de operare.

Testarea de integrare subliniază ideea de funcţionare corectă a aplicaţiei în ceea ce priveşte compatibilitatea dintre componente eliminând problema depistării erorilor de interfaţă între module, integritatea semantică a structurilor de date fiind păstrată.

Ca şi testare de integrare am folosit testarea de sus în jos. Am pornit cu modulul rădăcină, testarea conexiunii dintre client şi server, la acre am adăugat treptat restul nivelelor inferioare.

Testarea acestei aplicaţii web a implicat şi:

* Verificarea aplicaţiei în conformitate cu cerinţele clientului;
* Testarea tuturor modulelor;
* Broken links - verificarea tuturor link-urilor astefel încât să nu existe legături defecte.

### **5.1.2. Testul de securitate**

Presupune verificarae mecanismelor de protecţie implementate în sistem, de fapt protecţia la intrările neautorizate în sistem. Rolul unui proiect de securitate al unui sistem este să facă astfel încât costul de spargere al sistemului să fie mai mare decât beneficiile pe care le obţine prin spargerea sistemului.

### **5.1.3. Testarea arhitecturii Client-Server**

Aplicaţiile bazate pe arhiteectura client-server sunt considerate aplicaţii complexe datorită numărului şi diversităţii modulelor de prelucrare, Aceasta presupune o aplicaţie distribuită. În general o aplicaţie distribuită cuprinde un sistem central, mai multe subsisteme conectate la sistemul central şi mai mulţi clienţi conectaţi la un subsistem. Complexitatea arhitecturii este reflectată şi în testare.

Testarea acestei arhitecturi presupune trei niveluri diferite:

* client individual, caz în care aplicaţia client este testată individual, în mod deconectat şi are ca rezultat acceptarea sau respingerea modulelor; În ceea ce priveşte testarea clientului, testarea are ca efect eşecul de conectare la partea interactivă a aplicaţiei, clientul fiind deconectat;
* client şi server, caz în care acestea sunt testate împreună dar nu se ia în considerare reţeaua şi are ca rezultat acceptarea sau respingerea interacţiunii client-server.;
* client, server şi reţeaua, caz în care se testează tot ansamblul împreună şi se verifică dacă sistemul este corect, complet şi funcţionează în mediul real.

## **5.2. ASPECTE POZITIVE**

În urma evaluării proiectului reies caracteristici care fac aplicaţia intuitivă şi măreşte sfera de aplicabilitate al acestui produs, putând fi folosit cu încredere şi uşurinţă.

Pentru realizarea unei bune interfeţe cu utilizatorul s-au desprins următoarele caracteristici importante:

* Performanţă
* Fiabilitate crescută prin rezolvarea specificaţiilor
* Mentenanţa uşoară a codului prin folosirea documentaţiei de cod şi utilizarea structurii de alternare a codului Java cu HTML

Documentaţia aplicaţiei duce la o bună creştere a utilizabilităţii produsului şi implicit la creşterea operaţionalităţii în vederea utilizării codului de alte aplicaţii decât cea de care a fost creat.

* Consistenţă - prin folosirea consecventă a standardului ales, terminologie cunoscută, consistenţă în mesaje de erori, plasarea butoanelor fără a crea confuzii
* Flexibilitatea de introducere a datelor este in conformitate cu obiceiurile utilizatorilor
* Concret - aplicaţia este bine precizată şi bine definită
* Accesibilitatea

# CAPITOLUL 6

# CONCLUZII

**Lucrarea de faţă întruneşte cerinţele impuse de un sistem de gestiune și monitorizare pentru o firmă de curierat punând accent pe eliminarea unor costuri suplimentare, precum și o mai bună informare a clientului asupra stadiului în care se află comenzile.**

**Sistemul informatic prezentat îmbină eficiența adusă de aplicația Android care contribuie la simplificarea muncii depuse de un curier, cu partea de aplicație web care oferă posibilitatea clientului într-o manieră simplă de a-și verifica statusul curent la comenzii, precum și operatorului de a avea o evidență asupra comenzilor în diferite stadii.**

**În concluzie, prin realizarea acestui sistem se rezolvă problematica informării corecte și prompte a clientului despre stadiul comenzilor, precum și modalitatea de informare curierilor asupra comenzilor asignate către aceștia prin opțiunea de vizualizare a comenzile în curs de preluare sau livrare asignate lor. O altă facilitate adusă personalului este eliminarea unui device suplimentar de scanare și alocarea unei duble funcționalități telefonului mobil intens utilizat și înaintea implementării acestui sistem.**

# BIBLIOGRAFIE

**Cărți:**

* David Flanagan - "JavaScript: The DefinitiveGuide", 6th Edition, Editura O'Reilly Media, 2011
* Shelley Power - "JavaScript Cookbook", Editura O'Reilly Media, 2007
* Hans Bergsten - "JavaServer Pages", 2nd Edition, Editura O'Reilly Media, 2002
* Bruce W. Perry - "Java Servlet & JSP Cookbook", Editura O'Reilly Media, 2004
* Bryan Basham, Kathy Sierra, Bert Bates - "Head First Servlets and JSP", Editura O'Reilly Media, 2004
* Năstase Florea - ”Tehnologia aplicațiilor Web”, Editura Economică , 1998
* S. Buraga - ”Proiectarea site-urilor Web”, Editura Polirom, 2002
* Sabina Munteanu, curs „Fiabilitate”, anul 2013
* Cornelia Tudorie, curs „Baze de date”, anul 2011

**Resurse Web:**

* http://www.w3resource.com/
* http://www.w3schools.com/
* https://developer.android.com/index.html