

**UNIVERSITATEA “DUNĂREA DE JOS” DIN GALAȚI
FACULTATEA DE AUTOMATICĂ, CALCULATOARE,
INGINERIE ELECTRICĂ ȘI 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	
CERINȚE ȘI SPECIFICAȚII	7
1.1 Cerințele aplicației	7
1.2 Specificații	8
CAPITOLUL 2	
ANALIZA PROBLEMEI	10
CAPITOLUL 3	
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 APLICAȚIEI	18
3.2.1. Arhitectura aplicației.....	18
3.2.2. Șirul evenimentelor	20
CAPITOLUL 4	
IMPLEMENTAREA APLICAȚIEI	22
4.1. NOȚIUNI 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 APLICAȚIEI.....	49
4.5.1.	APLICAȚIA ANDROID	49
4.5.2.	APLICAȚIA WEB.....	51
CAPITOLUL 5		
TESTARE ȘI EVALUARE.....		54
5.1.	TESTAREA	55
5.1.1.	Strategii de testare.....	55
5.1.2.	Testul de securitate	56
5.1.3.	Testarea arhitecturii Client-Server	57
5.2.	ASPECTE POZITIVE	57
CAPITOLUL 6		
CONCLUZII.....		59
MANUAL DE UTILIZARE.....		60
BIBLIOGRAFIE.....		71

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 în ultimul timp, în special datorită răspîndirii 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 aceleași 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 și 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 două titluri: 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 și 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 unui 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 fiecărui 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 facilitare 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 în 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 în sistem, cu ajutorul username-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 nepreluat 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 concordați între statusul înregistrat de curier și statusul real al comenzii. Ultima opțiune și probabil și cea mai importanta este vizualizarea comenzilor preluate care urmează să 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 șase capitole:

- Primul capitol prezintă cerințele și specificațiile aplicației la nivel funcțional și tehnic;
- Capitolul doi este alcătuit din analiza problemei, evidențiind termenii specifici folosiți în cadrul unei companii de curierat;
- Capitolul trei tratează proiectarea sistemului la nivelul bazei de date și a aplicației, comunicarea și șirul evenimentelor la nivel de principiu;
- Capitolul patru urmărește modul cum a fost implementată aplicația la nivel tehnic
- Capitolul cinci prezintă rezultatele evaluării punctând aspectele pozitive ale sistemului și modul de realizare a testării acestuia;
- Ultimul capitol cuprinde concluzii referitoare la aplicația prezentată.

CAPITOLUL 1

CERINȚE ȘI SPECIFICAȚII

1.1 Cerințele aplicației

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 dobândite 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 asigurate (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 autentificare

- Vizualizarea de către operatori în urma autentificării a comenzilor în diferite stadii precum și asignarea acestora către curieri;
- Generarea codurilor unice de identificare (AWB) a coletelor, precum și a codurilor de tip QR, în scopul scanării;
- Generarea în format PDF a taloanelor aplicate coletelor.

1.2 Specificații

Proiectul își 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 dezvoltare, în special aplicația Android se adresează personalului companiei de curierat pentru o bună gestionare a comenzilor, dar și clienților acestora 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 IDE recomandat pentru dezvoltarea proiectelor web, care folosesc tehnologii precum JSP, SERVLET, HTML, CSS și JavaScript

Pentru realizarea aplicației 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 se 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.

Aplicația Web va fi dezvoltată în Eclipse 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 punct de legătură 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 către 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 inițială. 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”, avînd 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, avînd 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 să 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 asigurate, 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 aplică pe fiecare dintre colete. După această procedură operatorul sortează coletele în funcție de destinație, urmând să le scaneze 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 parcursul întregului tranzit al coletului / coletelor, clientul trebuie să 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 obține informații despre personalul firmei de curierat, punctele de lucru ale acestuia, despre comenzile aflate în diferite stadii sau informații despre clienții care au apelat la serviciile firmei de curierat.

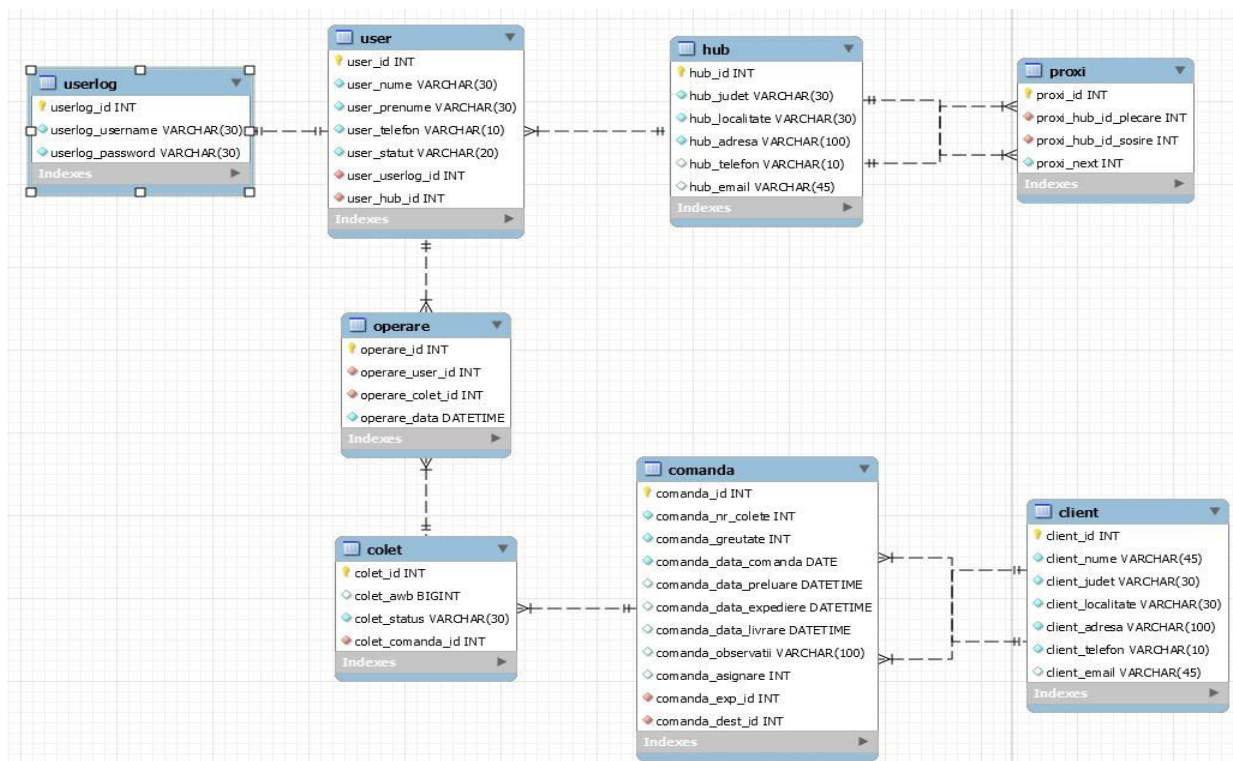


Figura 3.1.1. Schema bazei de date

Fiecare tabelă are un scop bine stabilit:

- **userlog** – tabela conține datele de autentificare ale personalului firmei de curierat, cu ajutorul cărora pot accesa sistemul. Tabela este populată la crearea conturilor de autentificare 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 când este efectuată o comandă cu informațiile 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

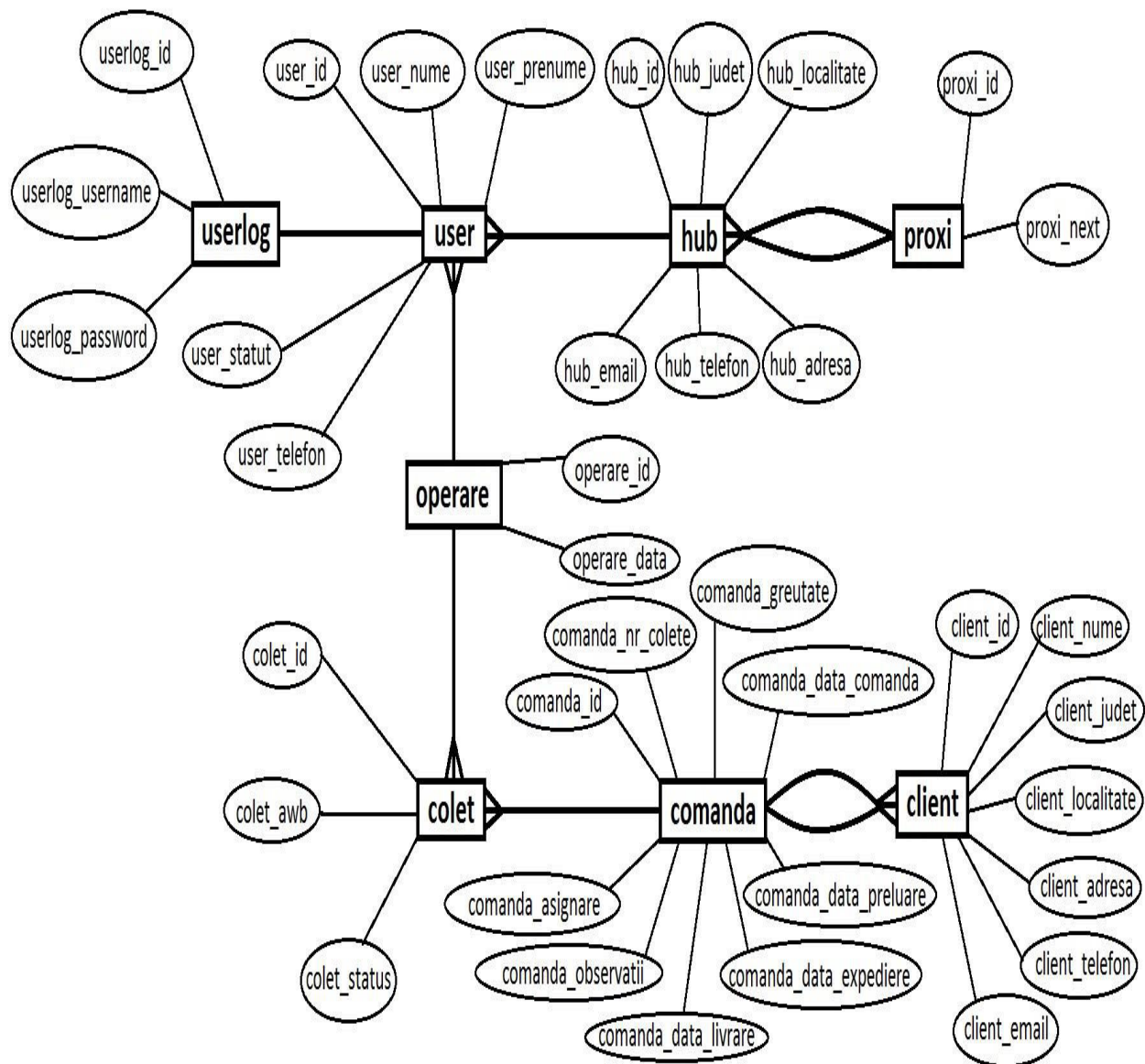


Figura 3.1.2. Modelul conceptual al bazei de date

3.1.3. Modelul logic al bazei de date

Tabela **userlog**

Column Name	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	Default
userlog_id	INT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	
userlog_username	VARCHAR(30)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
userlog_password	VARCHAR(30)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

Tabela **user**

Column Name	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	Default
user_id	INT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
user_nume	VARCHAR(30)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
user_prenume	VARCHAR(30)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
user_telefon	VARCHAR(10)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
user_statut	VARCHAR(20)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
user_userlog_id	INT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
user_hub_id	INT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

Tabela **hub**

Column Name	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	Default
hub_id	INT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	
hub_judet	VARCHAR(30)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
hub_localitate	VARCHAR(30)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
hub_adresa	VARCHAR(100)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
hub_telefon	VARCHAR(10)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
hub_email	VARCHAR(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

Tabela **proxi**

Column Name	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	Default
proxi_id	INT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
proxi_hub_id_plecure	INT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
proxi_hub_id_sosire	INT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
proxi_next	INT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

Tabela **operare**

Column Name	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	Default
operare_id	INT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
operare_user_id	INT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
operare_colet_id	INT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
operare_data	DATETIME	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

Tabela **colet**

Column Name	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	Default
colet_id	INT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	
colet_awb	BIGINT	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
colet_status	VARCHAR(30)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
colet_comanda_id	INT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

Tabela **comanda**

Column Name	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	Default
comanda_id	INT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	
comanda_nr_colete	INT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
comanda_greutate	INT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
comanda_data_comanda	DATE	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
comanda_data_preluare	DATETIME	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
comanda_data_expediere	DATETIME	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
comanda_data_livrare	DATETIME	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
comanda_observatii	VARCHAR(100)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
comanda_asignare	INT	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
comanda_exp_id	INT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
comanda_dest_id	INT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

Tabela **client**

Column Name	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	Default
client_id	INT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	
client_nume	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
client_judet	VARCHAR(30)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
client_localitate	VARCHAR(30)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
client_adresa	VARCHAR(100)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
client_telefon	VARCHAR(10)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
client_email	VARCHAR(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

3.2. PROIECTAREA APLICAȚIEI

3.2.1. Arhitectura aplicației Web

În principiu arhitectura aplicațiilor este structurată pe următoarele trei nivele:

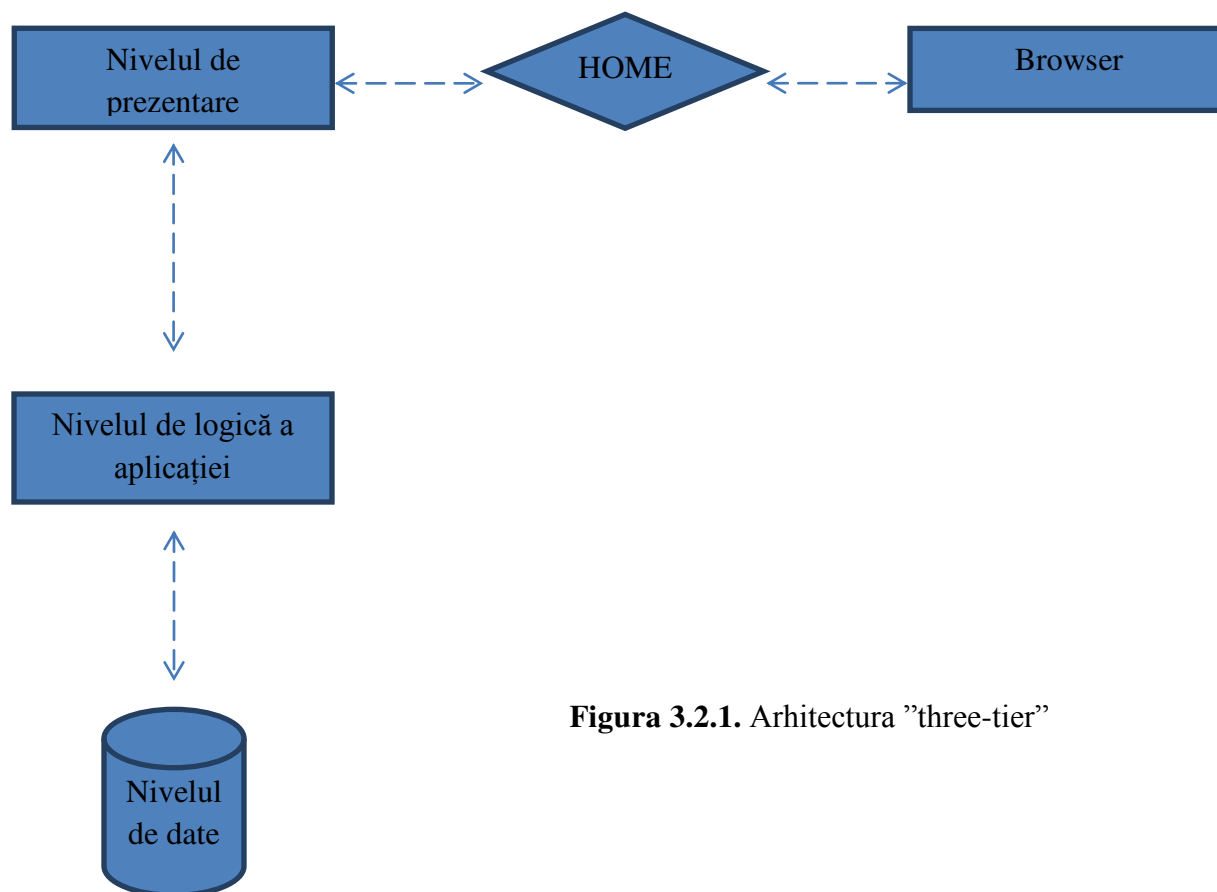


Figura 3.2.1. Arhitectura ”three-tier”

1. Baza de date

Serverul SQL. Acest nivel include logica aplicației (răspunde 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 aceste cerințe, arhitecturile sunt influențate de constrângeri tehnice cum ar fi sistemul utilizat (de exemplu 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 interactivă, 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. Șirul 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 prindează 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 punctul de lucru verifică comenzile care trebuie livrate, preiau un anumit număr de comenzi, scanează coletele, iar prin această scanare sunt asigurați automat la comanda respectivă. În urma scanării la accesarea aplicației de pe telefon vor putea vizualiza în secțiunea “Livrări” comenzile de livrat asigurate utilizatorului lor. De asemenea 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, modificându-se 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 APLICAȚIEI

4.1. NOȚIUNI TEORETICE

4.1.1. World Wide Web

Termenul de World Wide Web este un sistem de documente și informații care pot fi accesate prin rețeaua mondială Internet. Documentele care se află pe diferite calculatoare server în diferite locații pot fi accesate cu ajutorul unui identificator unic numit URL. Informația, care poate conține și imagini este afișată cu ajutorul unui program de navigare în web numit browser, care descarcă paginile web de pe un server și le afișează pe un terminal “client” la utilizator.

La baza funcționării web-ului stau trei standarde:

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

Unele dintre cele mai cunoscute aplicații browser sunt Microsoft Internet Explorer, Mozilla Firefox, Apple Safari, Google Chrome, Mozilla Camino, Opera Software - Opera, Nintendo DS browser și Flock. Aplicația browser pe care am folosit-o în cadrul aplicației este Mozilla Firefox.

4.1.2. Protocol HTTP

HTTP este metoda utilizată pentru accesarea informațiilor în Internet care sunt păstrate pe servere www. Fișierul trimis la destinație poate fi un document HTML, un fișier grafic, de sunet, animație sau video și 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, fișiere cu stil (CSS), script-uri (JavaScript), dar pot fi și pagini generate dinamic (JSP).

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

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

4.1.3. Modelul client - server

Termenul de client - server provine de la metoda tradițională de accesare a unui computer numit server de către computere aflate la distanță. Serverele utilizează baze de date relaționale în stocarea și întreținerea datelor între care există referințe. Acest model este o combinație a trei tehnologii: sistemul relațional de management al bazelor de date(DBMS), rețeaua și interfața 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 acțiuni. Server-ul execută cererea și trimite răspunsul înapoi clientului. Pentru a putea îndeplini cererea, server-ul poate referi o sursă de informații(baze de date) să efectueze procesări asupra datelor, să controleze periferice sau să efectueze cereri adiționale altor servere.

Procesarea este sincronă, clientul trimite cererea și așteaptă până când server-ul îi trimite răspunsul.

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

- Corelare: clientul face cererea și server- ul răspunde imediat trimițând 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 și un url la care el ascultă rezultatele. Acesta știe 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 și sosește 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 fișierul încorporează cod Java și îl trimite interpretorului Java
7. Interpretorul Java execută codul primit
8. Dacă codul conține și formulări MySQL, trimite datele sistemului MySQL
9. MySQL returnează rezultatele înapoi în interpretorului Java
10. Interpretorul Java returnează rezultatele server-ului web

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

4.1.4. Site-uri web

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

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

- paginile sale se creează în mod dinamic și automat în funcție de acțiunea utilizatorului;
- site-ul web se creează și este administrat de către utilizatorii lui.

Inițial paginile erau statice, trimiteau mereu aceeași informație(text plus imagini). Ulterior acestea au devenit dinamice în sensul că în funcție de anumite informații trimise de clientul browser, server-ul trimite diferite informații pentru aceeași pagină.

4.1.5. Hypertext

Înainte de apariția și dezvoltarea calculatoarelor, stocarea informațiilor se realiza fie în creierul uman, fie sub formă scrisă, ulterior, tipărită pe hârtie, în cărți și pe alte materiale.

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

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

4.2. TEHNOLOGII FOLOSITE

În continuare sunt prezentate tehnologiile și limbajele folosite în realizarea aplicației.

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 aplicației Android :

Activity (Activitate)

- reprezintă o interfață cu utilizatorul, fereastră sau formular;

În cadrul aplicației am folosit 8 activități :

- **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 realizează modificările aferente în baza de date.

AndroidManifest.xml:

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

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

AndroidManifest.xml:

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

Intent (Intenție)

- reprezintă o entitate folosită pentru a descrie o operațiune care urmează să fie executată;
- am folosit această componentă pentru a putea face posibilă deschiderea unei activități.

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 și care conține și multe alte instrumente, printre care o consolă javascript, posibilitatea de a monitoriza timpii de execuție sau de încărcare pentru o pagină, imagini etc.

Pentru accesul la Firebug trebuie instalat browser-ul Firefox și 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 alternativă viabilă și necomercială, în domeniul serverelor de web. Reușita acestui proiect este în mare măsură legată de fenomenele inițiate prin oameni ca Linus sau Stalman, ce au avut ca rezultat produse extraordinare, puternice și eficiente cum ar fi Linux, Emacs precum și toate pachetele software apărute sub licența GNU.

Crearea unui process de sine stătător – The Apache Project este rezultatul reunirii 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 reunească într-un proiect de dezvoltare și îmbunătățire a serverului inițial. 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 rulează 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 în 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 configurații de rețea. Acest server permite aplicațiilor server și client să ruleze pe același calculator. Acesta este un host pe care utilizatorii îl folosesc pentru a accesa rețeaua proprie a computer-ului. Adresa pe care o folosește 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 relațional. O bază de date este o modalitate de stocare a unor informații și date pe un suport extern (un dispozitiv de stocare), cu posibilitatea extinderii ușoare și a regăsirii rapide a acestora.

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

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

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

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

MySQL este un produs bazat pe relația client-server. MySQL este scris în C și C++. Parser-ul său pentru SQL este scris în Yacc și un analizor lexical mai puțin cunoscut numit `sql_lex.cc`.

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

Cele mai uzuale operații cu bazele de date sunt:

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

În MySQL spațiul alocat pe discul server-ului este în funcție 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 opțiunea UNSIGNED, care este un modificador de coloană, nu vor mai fi valori negative ci vor începe de la 0.

Alți modifcatori sunt:

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

MySQL Workbench este un instrument de bază de date vizuală pentru arhitecți, dezvoltatori, și DBA. MySQL Workbench furnizează modelare de date, dezvoltare SQL,

instrumente de administrare și configurare pentru, și multe altele. MySQL Workbench Permite unui Db administrator sau unui dezvoltator să proiecteze vizual, modelul, să genereze, și să gestioneze baze de date.

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

4.3. PROGRAMARE WEB

4.3.1. HTML

Limbajul în care sunt scrise paginile WEB se numește 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 funcție de posibilitățile hardware ale sistemului pe care se vizualizează pagina, și de posibilitățile browserului, pagina va fi afișată cu mai multe sau mai puține caracteristici de formatare (un browser în mod text nu va putea folosi fonturi de mărimi diferite).

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

Etichetele HTML sunt încadrate între paranteze <>, <tag> - eticheta de început și </tag> - eticheta de sfârșit. Efectul corespunzător etichetei este aplicat textului dintre eticheta de început și cea de sfârșit.

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

Un document HTML are de obicei următoarea componență:

1. Versiunea HTML a documentului

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

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

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

Tipurile de marcare în limbajul HTML sunt:

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

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

```
<html>

  <head>

    <title>TITLUL</title>

  </head>

  <body>

    CONȚINUT

  </body>

</html>
```

Exemple de tag-uri:

- <I>...</I> - stabilește stilul italic (înclinat);
- ... - stabilește stilul bold (îngroșat);
- <U>...</U> - stabilește stilul underline (subliniat);
- <P>...</P> - pentru a începe un nou paragraf;
- <TABLE>...</TABLE> - pentru a insera un nou tabel
- <TITLE>...</TITLE> - pentru a intitula o pagină

OBSERVAȚIE

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

Exemplu: Textul poate fi **boldat**, *italic* sau subliniat.

Codul pentru această propoziție este: <p>Textul poate fi boldat, <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 SVG.

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 și a background-ului până și la poziționarea 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 fișiere externe, fișiere CSS.

Acestea pot fi definite în partea de head a documentului html, sau pot fi definite în fișierul css extern care este apelat tot în secțiunea de head a paginii. În documentul HTML codul CSS poate 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>
```

Consecință: Orice text cuprins între va fi afișat cu roșu oriunde s-ar afla în document. Orice text afișat cu italice va fi de culoare albastră.

- Folosind metoda fișierelor CSS externe, toate paginile (X)HTML vor folosi același fișier de stil. Pentru a obține un aspect în toate paginile trebuie modificat un singur fișier, și anume cel de stil (.css), și efectul se va observa pe toate paginile (X)HTML, ce folosesc acel fișier.

CSS este creat pentru a face separarea între conținutul unui document și 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) proprietății respective, iar valoarea reprezintă stilul pe care îl aplici proprietății.

selector { proprietate:valoare }

Fiecare selector poate avea multiple proprietăți și fiecare proprietate din acel selector are valori independente. Proprietatea este separată de valoare cu semnul ":". Toate proprietățile împreună cu valorile lor, aparținând aceleiași selector sunt cuprinse între "{}".

Multiplele valori din aceeași proprietate sunt separate prin virgulă "," și dacă o valoare conține 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 proprietăți tuturor coloanelor dintr-un tabel, proprietăți cum ar fi: fontul, marimea scrisului, spațierea, 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 și 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 popularității web-ului designerii au început să caute diferite posibilități de a adăuga layout documentelor online. Pentru a răspunde acestor cerințe, producătorii de browsere (în acea

vreame Microsoft si Netscape) au inventat noi taguri HTML precum 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 layout pages). Multe taguri noi de layout precum <blink> erau recunoscute numai de unele browsere. O formulă comună ce apare pe site-uri era ”Aveți nevoie de browserul X pentru a vedea această pagină”. CSS a fost inventat pentru a remedia aceasta situație, furnizându-le designerilor facilități sofisticate pentru editarea layoutului, suportate de toate browserele.

În același timp, separarea site-urilor de prezentarea pentru documente de conținutul documentelor ușurează foarte mult întreținerea 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 clienților și construiește dinamic pagina HTML de răspuns. Servlet-urile sunt componente ale aplicațiilor server, independente de platformă, care extind dinamic serverele care au suport Java integrat. Ele sunt independente și 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 distanță și terminând cu protocolul HTTP pentru cereri către servere Web. Cu ajutorul servlet-urilor, așadar, se extinde funcționalitatea unei aplicații de tip server informațional (nu neapărat server HTTP), un prim domeniu de aplicare fiind, bineînțeles, 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, interfața grafică utilizator, lipsește din servlet-uri, ele neavând nevoie de așa ceva din moment ce rulează în interiorul serverelor. În rest, ele se aseamănă mult, un servlet fiind și el o componentă de aplicație, 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 și să prelucreze informația în mod dinamic, adică să ruleze aplicații Web și nu doar să transfere documente statice. Soluția Java, menține executabilul persistent pe server, între cererile clienților, 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 creșterea timpului).

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

Caracteristicile principale ale servlet-urilor se referă la următoarele:

- clasele și metodele necesare pentru a defini și utiliza un servlet sunt încapsulate în pachete Java;
- tehnologia se utilizează pentru a dezvolta soluții bazate pe Web: accesul securizat la pagini Web, asigurarea interacțiunii cu bazele de date, generarea dinamică a paginilor HTML, manipularea informațiilor care identifică unic un client pe parcursul uneia sau a mai multor sesiuni;

- se poate crea câte un servlet pentru fiecare funcție din paginile Web (conectare, înregistrare, actualizare etc.) sau unul singur care să gestioneze toate tranzacțiile din paginile Web respective, în mod dinamic;
- tehnologia este o soluție optimă pentru aplicațiile care utilizează intensiv bazele de date (serverul se ocupă de accesul la date iar clienții formulează cererile de regăsire). Partea de cod se scrie o singură dată și se stochează rezident, o singură dată, pe server. La actualizarea codului se va face o singură înlocuire, pe server, și nu la fiecare utilizator în parte;
- la inițiere se pot deschide conexiuni la baze de date care devin astfel rezidente între apelurile clienților;
- comunicarea client-server se realizează în mai mulți pași, astfel: clientul formulează și trimite către server o cerere Web; serverul o direcționează către servlet pentru a fi procesată (ceea ce implică de multe ori și accesul la o bază de date); răspunsul (sub formă de pagini HTML, imagini etc.) este returnat serverului și apoi clientului care a formulat cererea;
- un servlet poate fi apelat dintr-o pagină HTML sau dintr-un applet;
- principalele situații de utilizare a tehnologiei se referă la generarea paginilor Web dinamice și la realizarea aplicațiilor multi-nivel (multi-tier) cu JDBC. În acest ultim caz, servlet-ul poate accesa o varietate de baze de date prin intermediul JDBC și poate realiza, parțial sau total, interfața cu utilizatorul prin pagini Web dinamice.

Paginile JSP și componentele servlet sunt funcțional 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 părți de cod HTML în pagina de răspuns, atunci paginile JSP sunt mai simplu de folosit. În schimb, dacă respectiva cerere necesită multiple operații 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 aplicații ce cuprind pagini Web cu conținut dinamic precum HTML, DHTML, XHTML și XML. Sun a încercat să depășească limitările soluțiilor actuale pentru generarea de pagini cu conținut dinamic prin dezvoltarea unei tehnologii care:

- să funcționeze pe orice server Web sau de aplicații;
- să separe logica ce stă în spatele aplicației de aspectul paginii;
- să permită dezvoltare și testare rapidă;
- să simplifice procesul de dezvoltare de aplicații interactive Web.

Tehnologia JSP a fost creată să satisfacă aceste cerințe, fiind rezultatul unei cooperări la nivelul industriei software dintre producătorii de servere Web, servere de aplicații, sisteme tranzacționale și unelte de dezvoltare.

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

Pe partea de server, un engine (motor) JSP interpretează scriptlet-urile și tag-urile JSP, generează conținutul cerut (accesând componente JavaBean, baze de date folosind JDBC sau prin includerea de fișiere) și 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 și în cadrul unor unelte interactive de dezvoltare a componentelor și paginilor de Web. Creatorii de pagini Web nu sunt întotdeauna programatori familiarizați cu limbaje de scripting. JSP încapsulează funcționalitățile necesare pentru crearea

de conținut dinamic în tag-uri de tip XML specifice JSP. Tag-urile JSP standard pot accesa și instanța componente JavaBean, pot seta sau obține atribute ale bean-urilor, pot face download la applet-uri și pot executa funcții 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 funcțiile folosite cel mai frecvent.

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

O pagină JSP (*.jsp) este o pagină HTML sau XML ce cuprinde elemente adiționale (tag-uri, declarații, scriplet-uri) pe care motorul JSP le procesează și 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 informații 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: declarații, scriplet-uri și 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 încastate (embedded objects) în alte aplicații.

JavaScript este util pentru a verifica validitatea informațiilor introduse într-un formular înainte ca datele să fie trimise către server. O mențiune importantă: programele care rulează pe calculatorul utilizatorului sunt numite aplicații client-side (aflate pe partea de client) și programele care rulează pe server (inclusiv CGI-urile) sunt numite aplicații 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 și interpretat de către browser. Tag-ul <script> este prezent cel mai frecvent în secțiunea <head> a paginii HTML, deși se poate pune și în secțiunea <body>. De obicei script-urile care urmează să afișeze mesaje pe ecran sunt scrise în secțiunea <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 niște funcții 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 ierarhie, fiecare element fiind văzut ca un obiect, iar obiectul având proprietăți și metode.

La baza limbajului stau trei blocuri funcționale și anume: valor (tipuri de date suportate de JavaScript), obiecte (colecții de proprietăți care pot fi apelate cu un singur nume), funcții (care pot fi executate de o aplicație; funcțiile 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 script-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 să îl pună la dispoziție un browser, ceea ce oferă premiza scrierii de script-uri portabile, care să funcționeze 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 întâlnită 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 încărcare 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 comunicarea cu server-ul fără a fi nevoie să încarce o nouă pagină.

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

- reprezentări standard folosind HTML și CSS;
- afișare dinamică folosind DOM (Document Object Model);
- manipulări de date folosind XML;
- preluare de date folosind protocolul asincron XMLHttpRequest;
- JavaScript pentru a lega totul împreună.

Modelul clasic al unei aplicații web funcționează în felul următor: majoritatea acțiunilor utilizatorului declanșează 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 informație serializată JSON, care cu ajutorul JavaScript este procesată și afișată 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 și downloadează doar script-ul JavaScript și datele, paginile web Ajax pot părea că se încarcă relativ repede. De asemenea, mulțumită funcționalității "load on demand" a conținutului, unele pagini web încarcă stub-uri ale event handler-elor, iar apoi rulează funcțiile "on the fly". Această tehnică reduce considerabil lățimea de bandă folosită pentru aplicații 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 puțin programatic este că Ajax tinde să încurajeze programatorii să separe clar metodele, funcțiile și format-urile folosite în diferite aspecte ale transferului de informații 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 tabelor:

```
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 și a standardului ACID. Acest standard reprezintă un set de proprietăți care garantează că tranzacțiile bazei de date sunt procesate.

4.5. IMPLEMENTAREA FIZICĂ A APLICAȚIEI

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 consultarea 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 interoghează 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 curent. În plus din același ecran se lansează 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 câte 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 căruia s-a aplicat un ArrayAdapter pentru a seta stilul dorit listei. În urma particularizării listei vom putea observa că fiecare item conține caște un obiect grafic de tip ImageView și un TextView. La

evenimentul de click pe un item al listei se lansează activitatea *DetaliiPreluareActivity* prin intermediul unui intent care va transmite ca parametru identificatorul comenzii selectate.

Activitatea *DetaliiPreluareActivity* are rolul de a afișa informațiile referitoare la comanda selectată prin utilizarea mai multor 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 *checkbox*-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 rezultatul. 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 curentă și se redirecționează către activitatea de autentificare.

4.5.2. APLICAȚIA WEB

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

Fiind o aplicație 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 și apelată în toate fișierele de tip JSP sau Servlet, care interacționează cu baza de date, prin crearea unei noi instanțe 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 = "Ichirvase";

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 fișierele "disertatie_syle.css" și "styles.css", cu ajutorul limbajului CSS. Acestea au fost incluse în folder-ul "css" și 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 aplicației este *index.jsp*. În această pagină utilizatorul are ca opțiuni accesarea secțiunilor "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 autenticității utilizatorului, care se regăsește 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 ieși din aplicație, prin operația 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 secțiunea "Comenzi" se disting trei subcategorii: “Comenzi nepreluate”, “Comenzi în curs de preluare” și “Comenzi preluate”. Secțiunea “Comenzi nepreluate” este accesată prin intermediul fișierului *comenziNepreluate.jsp*, care va ajuta operatorul să vizualizeze toate comenzile nepreluate și să aibă posibilitatea de a asigna un curier la fiecare dintre acestea. După 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 preluare</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 ȘI EVALUARE

Testarea aplicației poate fi numită faza finală a procesului de dezvoltare web pentru orice aplicație. Aceasta este faza în care site-ul sau orice alt software implicat în crearea acestuia sunt testate pentru prezența de erori și bug-uri. Testarea nu numai că detectează bug-urile, dar, ajută utilizatorul să ia cunoștință 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 către auto-operare echipei de testare pentru evaluare (în cazul de față o singură persoană). Site-ul este testat pe baza unor tehnici diversificate și apoi evaluat. Procesul nu numai că asigură inexistența de bug-uri dar face site-ul mai sofisticat pentru clienții care îl vizitează. Procesul, îmbunătățește în mod evident standardul proiectului dumneavoastră.

Deși 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 proiectării, link-urile întrerupte din website, analiza de conținut pentru greșeli gramaticale, de aliniere a textului, așezarea de ansamblu în pagină, verificarea fiabilității și funcționalității script-urilor de programare backend și așa mai departe. Deși aceste teste par foarte simple, de fapt, necesită acordarea unei atenții sporite pentru că aceste detalii minore, atunci când sunt perseverente, ar putea zădărnici activitatea clienților pe website.

Testarea este de fapt un proces în lanț având în vedere că responsabilitatea sa, nu se termină doar prin specificarea de erori și bug-uri. Greșelile care sunt notate trebuie 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 că este gata de lansare.

Este adevărat că 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 în viitor. Website-ul 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 importanța și frecvența de utilizare, astfel:

Frecvența/importanța	Strategică	Importanța	Slabă
Mare	1	2	3
Medie	1	3	4
Mică	2	3	4

În continuare voi împărți 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 funcționări 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 funcționalitatea și utilizabilitatea produsului.

Ca și metode de testare am folosit metode statice, analiza programului înainte de a fi lansat în execuție, independent de datele de intrare, metode dinamice, care constau în execuția programului. Ca și metode statice am realizat testarea specificațiilor și examinarea codului.

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

Interfața aplicației afișează același aspect al paginii pe diferite rezoluții, diferite sisteme de operare.

Testarea de integrare subliniază ideea de funcționare corectă a aplicației în ceea ce privește compatibilitatea dintre componente eliminând problema depistării erorilor de interfață între module, integritatea semantică a structurilor de date fiind păstrată.

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

Testarea acestei aplicații web a implicat și:

- Verificarea aplicației în conformitate cu cerințele clientului;
- Testarea tuturor modulelor;
- Broken links - verificarea tuturor link-urilor astfel încât să nu existe legături defecte.

5.1.2. Testul de securitate

Presupune verificarea mecanismelor de protecție implementate în sistem, de fapt protecția 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 obține prin spargerea sistemului.

5.1.3. Testarea arhitecturii Client-Server

Aplicațiile bazate pe arhitectura client-server sunt considerate aplicații complexe datorită numărului și diversității modulelor de prelucrare, Aceasta presupune o aplicație distribuită. În general o aplicație distribuită cuprinde un sistem central, mai multe subsisteme conectate la sistemul central și mai mulți clienți conectați la un subsistem. Complexitatea arhitecturii este reflectată și în testare.

Testarea acestei arhitecturi presupune trei niveluri diferite:

- client individual, caz în care aplicația client este testată individual, în mod deconectat și are ca rezultat acceptarea sau respingerea modulelor; În ceea ce privește testarea clientului, testarea are ca efect eșecul de conectare la partea interactivă a aplicației, clientul fiind deconectat;
- client și server, caz în care acestea sunt testate împreună dar nu se ia în considerare rețeaua și are ca rezultat acceptarea sau respingerea interacțiunii client-server.;
- client, server și rețeaua, caz în care se testează tot ansamblul împreună și se verifică dacă sistemul este corect, complet și funcționează în mediul real.

5.2. ASPECTE POZITIVE

În urma evaluării proiectului reies caracteristici care fac aplicația intuitivă și mărește sfera de aplicabilitate al acestui produs, putând fi folosit cu încredere și ușurință.

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

- Performanță
- Fiabilitate crescută prin rezolvarea specificațiilor
- Menținerea ușoară a codului prin folosirea documentației de cod și utilizarea structurii de alternare a codului Java cu HTML

Documentația aplicației duce la o bună creștere a utilizabilității produsului și implicit la creșterea operaționalității în vederea utilizării codului de alte aplicații decât cea de care a fost creat.

- Consistență - prin folosirea consecventă a standardului ales, terminologie cunoscută, consistență în mesaje de erori, plasarea butoanelor fără a crea confuzii
- Flexibilitatea de introducere a datelor este în conformitate cu obiceiurile utilizatorilor
- Concret - aplicația este bine precizată și bine definită
- Accesibilitatea

CAPITOLUL 6

CONCLUZII

Lucrarea de față întrunește cerințele 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 asigurate către aceștia prin opțiunea de vizualizare a comenzile în curs de preluare sau livrare asigurate 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.

MANUAL DE UTILIZARE

Acest capitol vine în ajutorul personalului firmei de curierat care va utiliza pe viitor acest sistem, prin prezentarea interfeței și a modului de utilizare al acestuia. Acțiunile fiecărui utilizator al sistemului vor fi prezentate în ordinea desfășurării evenimentelor din cadrul unei firme de curierat.

Sistemul este constituit din două aplicații: aplicația Android dedicată întregului personal, atât curieri cât și operatori, cu mici restricții pentru operatori, care nu vor avea accesibile opțiunile de vizualizare a listelor de preluări și livrări, dar vor putea efectua operația scanare din cadrul aplicației; cea de-a doua aplicație este platforma web, accesibilă doar clienților și personalului de tip operator. La rândul său aplicația web va fi alcătuită din două secțiuni: o parte publică, în care clientul va avea posibilitatea trimiterii unei comenzi, precum și verificarea statusului comenzii sale pe parcursul tranzitului acesteia către destinație, și o parte privată accesibilă doar personalului de tip operator care în urma autentificării în sistem va putea consulta listele de comenzi în diferite stadii și opera asupra acestora.

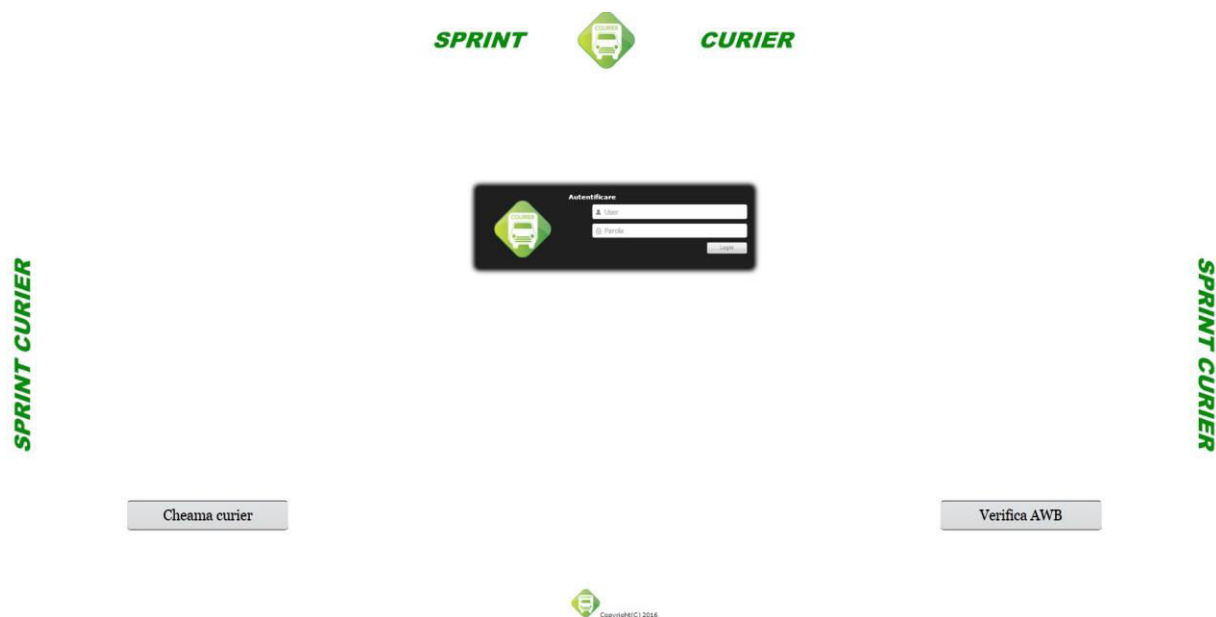


Figura 7.1. Pagina de start

Primul pas important este realizat de către client care va trimite o comanda spre firma de curierat prin accesarea aplicației web. Acest lucru se realizează la apăsarea butonului “Cheama curier”, acțiune ce va deschide o fereastră conținând un formular dedicate inserării informațiilor despre expeditor, destinatar, respective detaliile comenzii. În secțiunile destinate expeditorului și destinatarului vor trebui completate informații precum numele, județu, localitatea, adresa. date de contact ale celor doi, telefonul obligatoriu și eventual o adresa de email. În secțiunea de “Detalii comanda” vor fie inserate detalii referitoare la numărul coletelor, greutatea estimativă exprimată în kilograme și observații privitoare la comanda, un exemplu ar putea fi “conținut fragil”.

SPRINT CURIER

Expeditor

Nume:

Judet:

Localitate:

Adresa:

Telefon:

Email:

Destinatar

Nume:

Judet:

Localitate:

Adresa:

Telefon:

Email:

Detalii comanda

Numar colete:

Greutate totala (kg):

Observatii:

Trimite comanda

SPRINT CURIER

Figura 7.2. Trimitere comandă

După efectuarea comenzii de către client, un operator urmează să se autentifice în cadrul aplicației web în secțiunea de autentificare prin introducerea credențialelor, a username-ului și a parolei în cele două câmpuri “User” și “Parola” din formularul de autentificare de pe pagina principală. După apăsarea butonului „Login” utilizatorul curent va accesa pagina principală, unde din meniul din stânga sus, poziționat după logo-ul companiei va putea accesa opțiunea “Comenzi”. La trecerea mouse-ului peste meniul “Comenzi” se vor deschide trei submeniuri în această ordine: “Comenzi nepreluate”, ”Comenzi în curs de preluare” și ”Comenzi preluate”. La click pe prima opțiune “Comenzi nepreluate” operatorul va putea vizualiza comenzile efectuate

de clienți sub forma unei liste, cu informații precum “Id comanda”, “Numar colete”, “Data comanda” și “Asignare”. În cadrul rubricii “Asignare” se vor regăsi două elemente grafice, o listă derulantă care va fi populată cu personalul de tip curier arondat punctului de lucru asociat operatorului autentificat și un buton pentru confirmarea asignării curierului selectat din lista derulantă. Cea de-a doua opțiune a meniului “Comenzi” este “Comenzi în curs de preluare”, care la rândul său va fi accersată la evenimentul de click al mouse-ului, urmând a se deschide o nouă listă de această dată fără opțiunea de asignare doar cu informațiile specificate și mai sus despre comenzile aflate în stadiul “în curs de preluare”. În cadrul acestei liste se vor putea vizualiza comenzile care au fost asignate anterior unui curier din submeniul “Comenzi nepreluate”. Ultima opțiune a meniului principal “Comenzi preluate” va fi accesată asemeni celorlalte opțiuni prin apăsarea pe submeniul dorit, urmând ca în această secțiune operatorul să vizualizeze în cadrul unei liste comenzile asociate punctului său de lucru aflate în stadiul “preluat”. Lista current[conține informațiile comune celor două opțiuni de mai sus, “Id comanda”, “Numar colete”, “Data comanda” , în plus se va remarca rubrica “Generare AWB”, care pentru fiecare înregistrare va avea un buton dedicat generării codului unic de identificare și al talonului care va fi aplicat pe fiecare dintre coletele comenzii.



Figura 7.3. User logged - meniu Comenzi

Lista comenzi nepreluate			
Id comanda	Numar colete	Data comanda	Asignare
1	1	2016-05-20	Asignare [dropdown] <input type="button" value="Asigna"/>
3	1	2016-05-20	Asignare [dropdown] <input type="button" value="Asigna"/>
4	1	2016-05-20	Asignare [dropdown] <input type="button" value="Asigna"/>
7	3	2016-05-29	Asignare [dropdown] <input type="button" value="Asigna"/>

Figura 7.4. Comenzi nepreluate

Lista comenzi în curs de preluare		
Id comanda	Numar colete	Data comanda
1	1	2016-05-20
3	1	2016-05-20
7	3	2016-05-29

Figura 7.5. Comenzi în curs de preluare

Lista comenzi preluate			
Id comanda	Numar colete	Data comanda	Generare AWB
1	1	2016-05-20	Generare AWB
3	1	2016-05-20	Generare AWB
7	3	2016-05-29	Generare AWB

Figura 7.6. Comenzi preluate

AWB: 1464536701980			
EXPEDITOR:	DESTINATAR:	DETALII COMANDA:	QR CODE
Nume:	Nume:	Colete:	
Popescu Vasile	Ionescu Claudiu	1 / 1	
Judet:	Judet:		
Galati	Ilfov		
Localitate:	Localitate:	Greutate totala:	
Galati	Bucuresti	2	
Adresa:	Adresa:		
aaa	bbb		
Telefon:	Telefon:	Observatii:	
111	222	-	
Email:	Email:		
-	-		

Figura 7.7. Talon generat



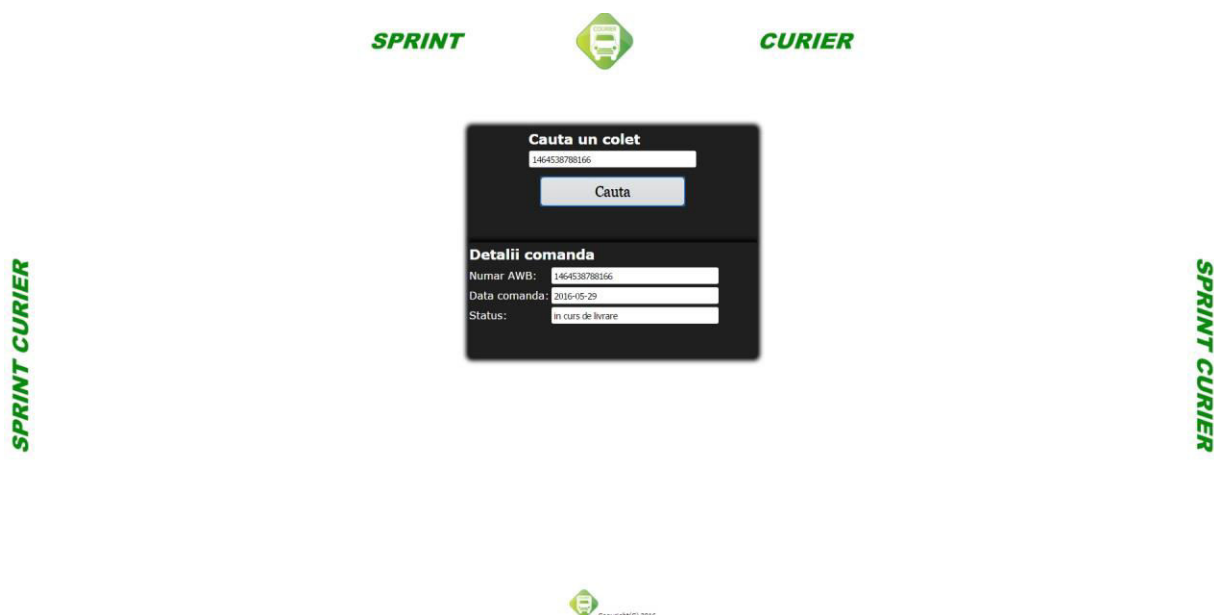
Figura 7.8. User logged - operații profil

În aceeași pagină de utilizator autentificat, de această dată în partea din dreapta sus se va regăsi o iconiță urmată de username-ul, numele și prenumele operatorului autentificat. La trecerea cu mouse-ul peste aria în care se află informațiile specificate anterior se vor derula în jos două submeniuri, având opțiunile de “Schimbare parola” și „Iesire”. La click de mouse pe prima opțiune se va deschide un formular care va oferi posibilitatea utilizatorului de a schimba parola curentă. Formularul deschis va solicita operatorului să insereze pe rând în cele trei câmpuri disponibile “Parola curentă”, “Parola nouă” și “Confirmare parola”. În primul camp se va introduce parola actuală, iar în celelalte două campuri va fi inserată noua parolă, urmând a se apăsa butonul “Editare” pentru salvarea informațiilor introduse.



Figura 7.8. Schimbare parolă

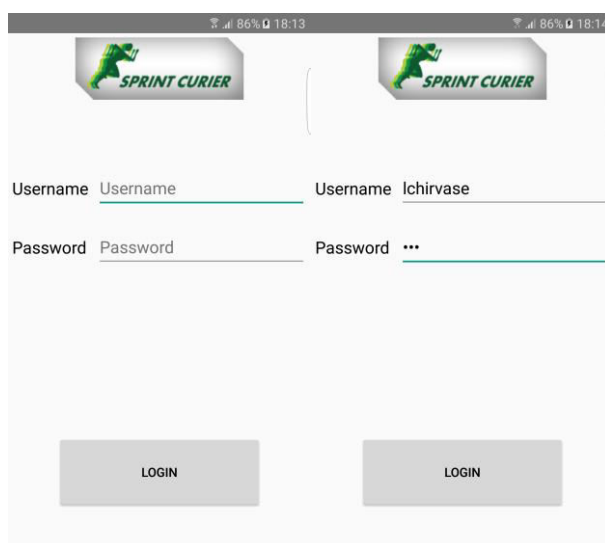
După efectuarea comenzii de către client și procesarea acesteia de un operator al firmei de curierat utilizatorul de tip client va putea verifica stadiul curent al comenzii sale. Acest lucru este posibil din aplicația web în partea publică a acesteia, la apăsarea butonului “Verifica AWB” din partea dreaptă jos a paginii principale. La accesarea acestei secțiuni se va deschide o nouă pagină conținând un formular cu un câmp pentru inserarea AWB-ului pentru care se dorește obținerea statusului curent. După inserarea AWB-ului utilizatorul va confirma prin apăsarea butonului “Cauta” pentru afișarea informațiilor. În urma acestei operații se va deschide un nou formular cu informații privind ‘Numar AWB’, ‘Data comanda’ și ‘Status’.



The screenshot displays the SPRINT CURIER web application interface. At the top, the SPRINT CURIER logo is centered, flanked by the words 'SPRINT' and 'CURIER'. Below the logo is a dark grey box containing the search functionality. The box has a title 'Cauta un colet' and a text input field with the value '1464538788166'. A 'Cauta' button is positioned below the input field. Underneath the search box, there is a section titled 'Detalii comanda' (Order details) which lists the following information: 'Numar AWB: 1464538788166', 'Data comanda: 2016-05-29', and 'Status: in curs de livrare'. The entire interface is framed by a light grey border with the text 'SPRINT CURIER' repeated vertically on both sides. At the bottom center, there is a small logo and the text 'Copyright(C) 2016'.

Figura 7.9. Verificare AWB

Pe lângă aplicația web prezentată mai sus personalul va avea acces și la aplicația Android care va trebui instalată pe telefonul de serviciu al fiecărui angajat al firmei de curierat, cu precizarea că pe telefon este necesar să ruleze sistemul de operare Android și să fie disponibilă o sursă de internet fie că este de tip WI-FI sau de la un furnizor de internet, în scopul conectării la server-ul Apache și la cel de baze de date.



The screenshot shows the SPRINT CURIER Android application's login screen. The screen is divided into two identical login sections. Each section features the SPRINT CURIER logo at the top, followed by two input fields: 'Username' and 'Password'. Below the password field is a 'LOGIN' button. The status bar at the top of the screen shows a battery level of 86% and the time 18:13. The background is a light grey color.

Figura 7.10. Autentificare

După îndeplinirea condițiilor menționate mai sus, utilizatorul poate accesa aplicația “DisertatieAndroidApp”, care va deschide o primă pagină dedicată autentificării în sistem. Aceasta va conține două câmpuri pentru inserarea username-ului și a parolei, precum și un buton de confirmare a datelor, ce permite intrarea în sistem în urma validării informațiilor introduse de utilizator.

În ipoteza unei autentificări nereușite în aplicație se va face vizibil în partea de jos a ecranului un mesaj sugestiv “Login failed”. În cazul finalizării cu succes a autentificării utilizatorul va fi directat către ecranul principal al aplicației denumit “User Panel”, care cuprinde lista de opțiuni a aplicației, printre care se numără lista de comezni de preluat, lista de comenzi de livrat, modulul de scanare și operațiile de profil pentru schimbarea parolei și ieșirea din aplicație. Oricare dintre opțiunile enumerate anterior vor putea fi accesate prin simpla apăsare a butonului asociat opțiunii dorite. Trebuie menționat că primele două opțiuni ale meniului sunt dedicate doar curierilor, iar pentru operatori vor apărea ca fiind blocate.

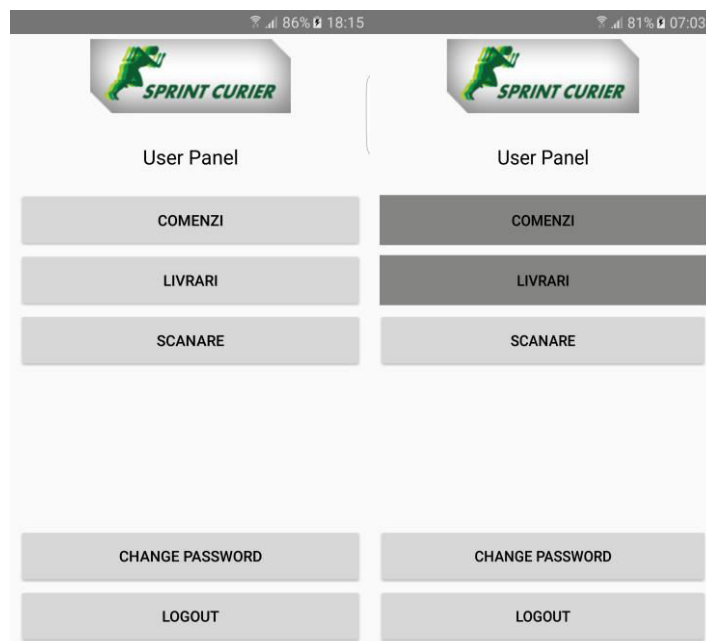


Figura 7.11. Meniu principal

Prin accesarea primei opțiuni din meniul principal utilizatorul va putea consulta lista de comenzi de preluat asociate utilizatorului său. La deschidere se vor putea distinge titlu “Lista preluari” și lista efectivă de comenzi de preluat, fiecare item al acesteia fiind reprezentat printr-

un icon și id-ul comenzii. În cazul lipsei comenzilor pentru utilizatorul curent lista va fi goală iar la accesare va apărea un mesaj de atenționare “Lipsa comenzi”. În plus pentru a obține informații suplimentare despre oricare dintre comenzile acestei liste se va putea face click pe comanda dorită, operație ce va deschide un alt ecran cu informațiile necesare preluării comenzii respective. În cadrul acestui ecran se vor putea vizualiza informații precum numele expeditorului, adresa locației de unde se va prelua comanda, numărul de telefon pentru contactarea expeditorului și numărul coletelor de preluat. Tot în acest ecran după informațiile de mai sus se va observa și posibilitatea de bifare/debifare a unui checkbox pentru schimbarea statusului. Această ultimă opțiune se va bifa în momentul în care curierul a preluat comanda, iar pentru confirmare se va apăsa butonul “SALVARE”.

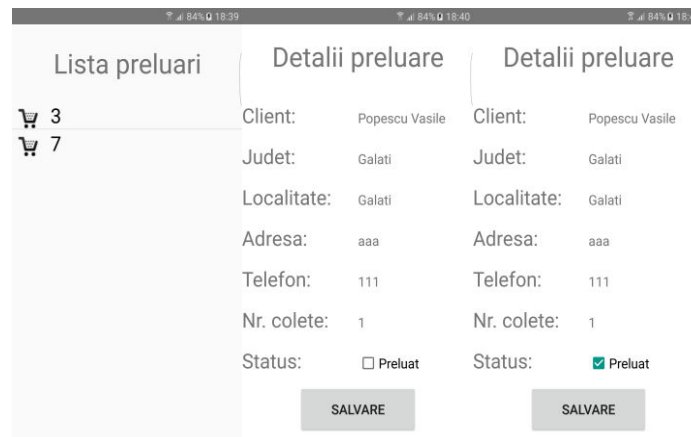


Figura 7.12. Informații preluare

Cea de-a doua opțiune a meniului principal este cumva similară cu ce am despris mai sus, singura diferență apare la datele care sunt afișate spre vizualizare, de această dată se va accesa butonul “LIVRARI”, iar ecranul care se va deschide va conține titlul “Lista livrari” și o listă alcătuită din comenzile de livrat asociate utilizatorului curent. Și în acest caz există aceeași opțiune de click pe fiecare item al listei în scopul vizualizării informațiilor de această dată datele referitoare la expeditor. În josul ecranului se poate observa checkbox-ul de bifat în cazul efectuării livrării și butonul de salvare a informației. În acest caz opțiunea de actualizare a statusului din acest punct al aplicației este mai mult de siguranță, având în vedere că în mod normal curierul la livrare are obligația scanării și implicit actualizării statusului automat, dar în ipoteza omiterii acestei operații, va avea posibilitatea actualizării statusului din această secțiune.

The screenshot displays a mobile application interface for delivery management. It is divided into three main sections:

- Lista livrari:** A list of deliveries. The first item is labeled '1' and the second '7', each with a small truck icon.
- Detalii livrare (Left):** A form showing details for the first delivery (ID 1). The fields are: Client: Ionescu Claudiu, Judet: Ilfov, Localitate: Bucuresti, Adresa: bbb, Telefon: 222, Nr. colete: 1, and Status: ☐ Livrat. A 'SALVARE' button is at the bottom.
- Detalii livrare (Right):** A form showing details for the second delivery (ID 7). The fields are: Client: Ionescu Claudiu, Judet: Ilfov, Localitate: Bucuresti, Adresa: bbb, Telefon: 222, Nr. colete: 1, and Status: ☒ Livrat. A 'SALVARE' button is at the bottom.

Figura 7.13. Informații livrare

Modulul de scanare se accesează la click pe opțiunea “Scanare” care va lansa scanner-ul pentru efectuarea operației de scanare a coletelor și actualizarea informațiilor. În urma scanării QR Code-ului existent pe talonul coletului se va deschide un ecran cu informații referitoare la coletul scanat, printre care id-ul coletului, awb-ul și statusul curent, și un buton de actualizare a informațiilor, pentru actualizarea statusului coletului scanat.

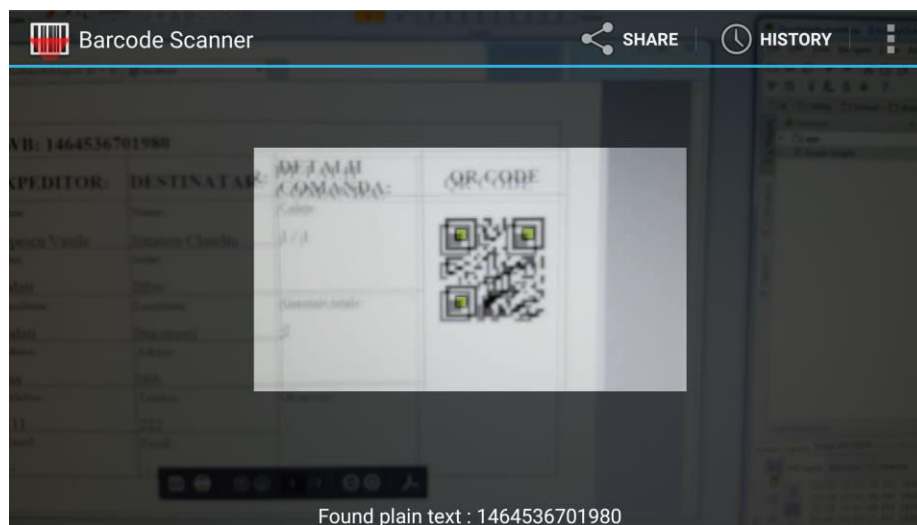
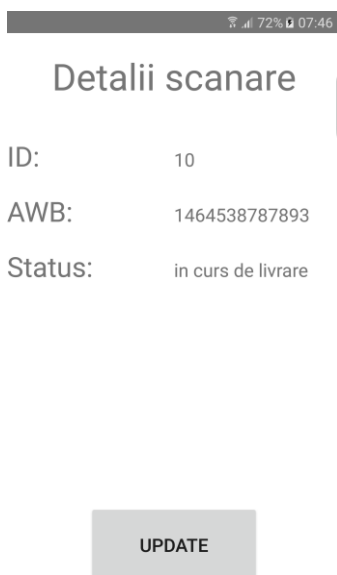


Figura 7.14. Captură scanner

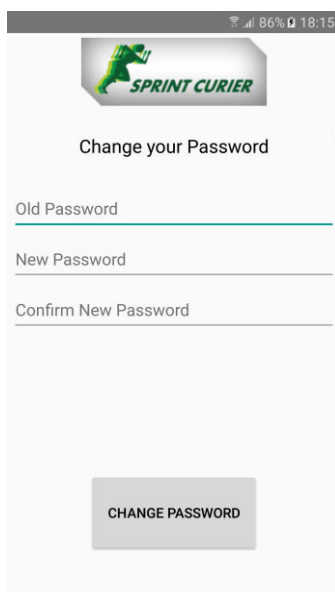


The screenshot shows a mobile application interface with a status bar at the top displaying signal strength, 72% battery, and the time 07:45. The main heading is "Detalii scanare". Below it, there are three rows of data: "ID:" with value "10", "AWB:" with value "1464538787893", and "Status:" with value "in curs de livrare". At the bottom, there is a grey button labeled "UPDATE".

Detalii scanare	
ID:	10
AWB:	1464538787893
Status:	in curs de livrare
<input type="button" value="UPDATE"/>	

Figura 7.15. Informații scanare

Opțiunea de schimbare a parolei este disponibilă la click pe butonul “CHANGE PASSWORD ” din meniul principal, operație ce va deschide un ecran cuprinzând trei câmpuri de completat pe rând cu informații privind parola curentă, noua parolă și reintroducerea spre confirmare a noii parole. După completarea acestor câmpuri se va confirma operația de schimbare a parolei prin intermediul butonului din partea de jos a ecranului “CHANGE PASSWORD”. În cazul neinserării unor câmpuri sau a inserării parolei actuale necorespunzătoare contului current se vor trimite mesaje de avertizare/eroare în partea de jos a ecranului.



The screenshot shows a mobile application interface for 'SPRINT CURIER'. At the top, there is a status bar with signal strength, 86% battery, and the time 18:15. Below the status bar is the application logo, which features a green and yellow running figure and the text 'SPRINT CURIER'. The main heading is 'Change your Password'. There are three input fields: 'Old Password', 'New Password', and 'Confirm New Password'. At the bottom, there is a grey button labeled 'CHANGE PASSWORD'.

Figura 7.16. Schimbare parola

În cazul în care utilizatorul și-a încheiat activitatea sau dorește să părăsească aplicația are posibilitatea prin accesarea ultimei opțiuni din meniul principal, printr-un click pe butonul “LOGOUT”.

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 „Tehnici de evaluare și testare software”, anul 2016
- Cornelia Tudorie, curs „Sisteme avansate cu baze de date”, anul 2015

Resurse Web:

- <http://www.w3resource.com/>
- <http://www.w3schools.com/>
- <https://developer.android.com/index.html>