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

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

**SPECIALIZAREA: TEHNOLOGII INFORMATICE AVANSATE**

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

**Profesor îndrumător,**

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

**Masterand,**

**Liviu George CHIRVASE**

**GALAȚI**

**2016**

CUPRINS

INTRODUCERE 4

CAPITOLUL 1 6

CERINȚE ŞI SPECIFICAŢII 6

1.1 Cerinţele aplicaţiei 6

1.2 Specificaţii 7

CAPITOLUL 2 8

ANALIZA PROBLEMEI 8

CAPITOLUL 3 12

PROIECTAREA SISTEMULUI 12

3.1. PROIECTAREA BAZEI DE DATE 12

3.1.1. Formularea scopului 12

3.1.2. Modelul conceptual al bazei de date 14

3.1.3. Modelul logic al bazei de date 15

3.2. PROIECTAREA APLICAŢIEI 17

3.2.1. Arhitectura aplicaţiei 17

3.2.2. Şirul evenimentelor 19

CAPITOLUL 4 20

IMPLEMENTAREA APLICAŢIEI 20

4.1. NOŢIUNI TEORETICE 20

4.1.1. World Wide Web 20

4.1.2. Protocol HTTP 20

4.1.3. Modelul client - server 21

4.1.4. Site-uri web 23

4.1.5. Hypertext 23

4.2. TEHNOLOGII FOLOSITE 24

4.2.1. Firebug 24

4.2.2. Server-ul Apache 24

4.2.3 Sistemul de baze de date MySQL 25

4.3. PROGRAMARE WEB 28

4.3.1. HTML 28

4.3.2. CSS 30

4.3.2.1. Diferență între CSS și HTML 33

4.3.2.2. Avantajele CSS-ului 34

4.3.3. SERVLET 34

4.3.4. JAVA SERVER PAGES (JSP) 37

4.3.5. JAVASCRIPT 38

4.3.5.1. Ajax 40

4.4. IMPLEMENTAREA FIZICĂ A BAZEI DE DATE 43

4.5. IMPLEMENTAREA FIZICĂ A APLICAŢIEI 44

CAPITOLUL 5 48

TESTARE ŞI EVALUARE 48

5.1. TESTAREA 49

5.1.1. Strategii de testare 49

5.1.2. Testul de securitate 50

5.1.3. Testarea arhitecturii Client-Server 51

5.2. ASPECTE POZITIVE 51

CAPITOLUL 6 53

CONCLUZII 53

MANUAL DE UTILIZARE 54

BIBLIOGRAFIE 63

# INTRODUCERE

Evaluarea este activitatea complexă prin care se determină nivelul și calitatea pregătirii studenților pe parcursul programelor de studii, precum și competențele de care dispun absolvenții la finalizarea studiilor. Ansamblul metodelor, formelor, tipurilor și criteriilor de evaluare și notare formează sistemul de evaluare a performanțelor profesional-științifice ale studenților.

Evaluarea performanțelor profesional-științifice ale studenților se înscrie în succesiunea coerentă și interdependentă a acțiunilor principale ce alcătuiesc procesul de învățământ, respectiv proiectare-predare-învățare-evaluare.

Evaluarea performanțelor profesional-științifice ale studenților este parte integrantă a sistemului de evaluare pedagogică a proceselor și structurilor educaționale.

Rezultatele evaluării studenților reprezintă o sursă relevantă de informații pentru evaluarea curriculumu-lui, a prestației pedagogice a cadrelor didactice, a eficienței procesului de învățământ, a funcționalității structurilor academice.

Rezultatele evaluării performanțelor studenților sunt integrate procedurilor de analiză colegială, de monitorizare și de evaluare periodică a programelor de studii; în același timp rezultatele evaluării reprezintă un indicator sintetic al rezultatelor învățării.

Monitorizarea situaţiei şcolare în mediul universitar reprezintă o problemă, ca urmare a faptului că nu există un sistem de centralizare al notării studenţilor, care să permită cadrelor didactice să gestioneze mai uşor atât partea de notare, cât şi partea de raportare şi statistică a situaţiei acestora.

Acest impediment poate fi eliminat prin intermediul aplicaţiei web pe care mi-am propus să o realizez, care pe lângă platformele deja existente in sistemul universitar actual (ex: platforma note.ugal.ro - pentru vizualizarea situaţiei individuale a fiecărui student; platforma edu.csed.ugal.ro - dedicata interacţiunii profesor - student, pentru vizualizarea de cursuri, anunţuri privitoare la examene sau rezultatele obţinute de studenţi etc.), are scopul de a realiza rapoarte şi statistici referitoare la activitatea întreprinsă de studenţi.

Lucrarea este compusă dintr-o secţiune publică, destinată în special studeţilor pentru urmărirea rezultatelor la examene, şi o secţiune privată destinată cadrelor didactice din departamentul "Calculatoare şi Tehnologia Informaţiei". În secţiunea publică, cadrul didactic are posibilitatea afişării rezultatelor obţinute de studenţi, fapt ce vine în ajutorul studentului pentru a afla situaţia examinării.

Secţiunea privată, accesată de cadrele didactice prin autentificare, nu se rezumă doar la partea de raportare, ci aduce o îmbunătăţire semnificativă şi pe partea de notare a studenţilor. Acest fapt este evidenţiat de modulul de generare în format electronic a rezultatelor obţinute de studenţi la examene, pe baza notelor introduse de cadrul didactic. În acelaşi timp, cadrele didactice beneficiază de avantajul realizării statisticilor pe baza datelor introduse în cadrul aplicaţiei, care reflectă activitatea comună cadru didactic-student. Astfel cadrele didactice, pe baza acestor rezultate vor avea evidenţa studenţilor restanţieri, vor putea stabili noi metode de predare şi evaluare, care să îmbunătăţească situaţia studenţilor şi să crească promovabilitatea si performanţele profesionale.

Lucrarea este structurată pe şase capitole:

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

# CAPITOLUL 1

# CERINȚE ŞI SPECIFICAŢII

## **1.1 Cerinţele aplicaţiei**

Identificarea şi înţelegerea necesităţilor grupului ţintă, în acest caz departamentul de "Calculatoare şi Tehnologia Informaţiei", reprezintă un prim pas în elaborarea temei propuse. În urma consultării personalului didactic am obţinut un cumul de informaţii necesare în stabilirea cerinţelor şi obiectivelor propuse în scopul realizării acestei lucrări.

Pentru realizarea aplicaţiei propuse, pe baza informaţiilor prezentate, au fost stabilite urmatoarele cerinţe:

* Crearea unei aplicaţii web, în scopul monitorizării situaţiei şcolare a studenţilor de la specializarea "Calculatoare şi Tehnologia Informaţiei";
* Proiectarea bazei de date care va reprezenta suport pentru aplicaţia web;
* Monitorizarea în cadrul aplicaţiei doar a disciplinelor specifice specializării menţionate mai sus;
* Evidenţierea a doua secţiuni distincte, o secţiune publică şi o secţiune privată;
* Accesul în secţiunea privată se va realiza prin intermediul unei interfeţe de autentificare;
* Realizarea unei subsecţiuni în cadrul secţiunii private a interfeţei pentru importarea de fişiere Excel în baza de date cu informaţiile actualizate în fiecare an universitar;
* Crearea unui sistem de introducere a notelor în baza de date şi generarea de documente în format PDF, conform datelor inserate;
* Realizarea unui sistem de rapoarte şi statistici.

## **1.2 Specificaţii**

Proiectul îşi propune să realizeze o fuziune a platformelor actuale privitoare la situaţia şcolară a studenţilor şi să creeze un sistem de centralizare şi monitorizare, care să îmbunătaţească modul de verificare asupra activitaţii studenţilor de la specializarea "Calculatoare şi Tehnologia Informaţiei".

Aplicaţia se adresează în primul rând cadrelor didactice, care vor beneficia de noul sistem de raportare şi statistici şi generarea acestora în format printabil, precum şi studenţilor pentru a vizualiza rezultatele examinării lor.

Ca mediu de programare se va folosi Eclipse EE IDE utilizat pentru dezvoltarea proiectelor web, care folosesc tehnologii precum JSP, SERVLET, HTML, CSS şi JavaScript.

Pentru realizarea aplicaţiei am folosit:

* serverul Web Tomcat Apache
* serverul de DB MySQL

Aplicaţia Web instalată în serverul Apache răspunde la adresa localhost. Pentru sistemul de baze de date se foloseşte MySQL Workbench, care ajută la proiectarea şi administrarea sistemului de baze de date.

# CAPITOLUL 2

# ANALIZA PROBLEMEI

Facultatea este o unitate didactică și administrativă în cadrul unei instituții de învățământ superior, condusă de un decan și în care se predă un ciclu de discipline științifice înrudite între ele, pentru pregătirea studenților și a doctoranzilor într-un anumit domeniu de specialitate.

Structura tipică a unui an universitar cuprinde două semestre care se finalizează cu câte o sesiune de examene. În toamnă se poate organiza o a treia sesiune de refacere. Stagiul practic se poate desfăşura după sesiunea II sau, pentru anumite grupe de studenţi, sub forma unor activităţi de cercetare-proiectare pe toată durata celor două semestre.

CONTRACTUL DE STUDII

Învăţământul bazat pe credite permite studentului să-şi definească propriul program formativ pe baza ofertei şi a regulilor care stabilesc specificul diplomei acordate de facultate, la care aspiră studentul.

Efectuarea tuturor lucrărilor practice (seminar, laborator, proiect) constituie condiţia necesară pentru promovarea disciplinelor la care sunt prevăzute astfel de activităţi didactice.

Recuperarea absenţelor la lucrările practice se poate face gratuit în timpul semestrului pentru maxim 2 lucrări, în cazul disciplinelor la care Planul de învăţământ prevede 14 lucrări şi maxim o lucrare pentru cele la care sunt prevăzute 7 lucrări. Recuperările vor fi efectuate conform programării indicate de cadrul didactic, de regulă cu altă formaţie de studiu, în cursul aceleiaşi săptămâni şi numai în cazuri excepţionale în ultima săptămână a semestrului.

Studenţii care după încheierea semestrului rămân cu absenţe la lucrările practice, vor putea să le recupereze numai în regim cu taxă. În aceste situaţii, recuperarea activităţilor practice poate fi făcută în sesiunea curentă sau în cea de restanţe, la datele fixate de cadrul didactic care a condus activitatea didactică la formaţia lor de studiu.

CREDITE

*Creditul* sau *numărul de credite* reprezintă un punctaj asociat fiecărei discipline de studiu şi este proporţional cu cantitatea de muncă necesară studentului pentru a promova disciplina, cuprinzând atât activităţile desfăşurate în mod organizat (cursuri, seminarii, lucrări practice de laborator, proiect, stagii practice, participarea la procesul de evaluare a cunoştinţelor dobândite) cât şi munca individuală depusă pentru asimilarea cunoştinţelor predate, efectuarea temelor, elaborarea referatelor şi proiectelor.

PREZENŢA LA ACTIVITĂŢILE DIDACTICE

Efectuarea tuturor lucrărilor practice (seminar, laborator, proiect) constituie condiţia necesară pentru promovarea disciplinelor la care sunt prevăzute astfel de activităţi didactice.

Recuperarea absenţelor la lucrările practice se poate face gratuit în timpul semestrului pentru maxim 2 lucrări, în cazul disciplinelor la care Planul de învăţământ prevede 14 lucrări şi maxim o lucrare pentru cele la care sunt prevăzute 7 lucrări. Recuperările vor fi efectuate conform programării indicate de cadrul didactic, de regulă cu altă formaţie de studiu, în cursul aceleiaşi săptămâni şi numai în cazuri excepţionale în ultima săptămână a semestrului.

Studenţii care după încheierea semestrului rămân cu absenţe la lucrările practice, vor putea să le recupereze numai în regim cu taxă. În aceste situaţii, recuperarea activităţilor practice poate fi făcută în sesiunea curentă sau în cea de restanţe, la datele fixate de cadrul didactic care a condus activitatea didactică la formaţia lor de studiu.

EVALUAREA CUNOŞTINŢELOR

Modul de evaluare a cunoştinţelor, pentru fiecare disciplină în parte, va fi adus la cunoştinţa studenţilor de titularul de curs la începutul semestrului respectiv. În sistemul de evaluare bazat pe E.C.T.S. nota finală se calculează ca o medie ponderată între nota obţinută de student la activităţile didactice de pe parcursul semestrului şi nota de la examen. Pentru promovare este necesar ca ambele note să fie cel puţin egale cu 5.

Nota care reflectă activitatea studentului la activităţile didactice de pe parcursul semestrului se calculează ca o medie fără rotunjiri, a cel puţin trei note, dintre care cel puţin două trebuie să fie obţinute în urma unor teste date cu formaţia de studiu, în condiţii cât mai apropiate de cele de la examen (subiecte, supraveghere, notare).

Studenţii care nu obţin pe parcursul semestrului minim nota 5 pot cere o examinare suplimentară (gratuită) în ultima şedinţă de activităţi practice. Nota obţinută la această verificare suplimentară se adaugă la celelalte note obţinute în timpul semestrului. Dacă şi după această notare nu realizează cel puţin 5.00, atunci studenţii în cauză vor putea cere refacerea în regim cu taxă a testelor susţinute pe parcursul semestrului. Refacerea va avea loc la datele fixate de cadrul didactic care a condus activitatea didactică cu formaţia lor de studiu

Dacă nici în aceste condiţii nu a reuşit să obţină cel puţin nota 5.00, studentul poate fi primit totuşi la examen, însă activitatea de pe parcursul semestrului va trebui refăcută în totalitate, în regim cu taxă, similar cu obţinerea creditelor restante. Chiar dacă a promovat examenul, dar nu a promovat activitatea practică, studentul are statutul de restanţier în regim cu credite restante. Taxa pe care va trebui să o achite în anul universitar următor pentru refacerea aplicaţiilor practice va fi stabilită ca un procent din taxa aferentă disciplinei, egal cu ponderea evaluării pe parcurs în nota finală. În mod similar, studenţii care nu au promovat examenul, dar au promovat activităţile practice, vor reface în anul universitar următor examenul. Şi în această situaţie studentul are statutul de restanţier în regim cu credite restante, iar taxa achitată va fi calculată proporţional cu ponderea examenului în evaluarea finală.

Într-un an universitar, orice examen sau colocviu poate fi susţinut fără taxă, o dată în sesiunea în care a fost programat şi încă o dată în sesiunea a III-a (de refacere).

Un examen poate fi susţinut în sesiunea III şi pentru MĂRIREA NOTEI cu condiţia de integralitate după sesiunea din vară. Notele acordate la proiectei nu mai pot fi mărite.

CONDIŢII DE PROMOVARE, DURATA MAXIMA A STUDIILOR

La începutul fiecărui an universitar studentul va depune o cerere prin care va solicita înscrierea în anul universitar şi încheierea contractului de studii pentru anul precedent.

Dacă studentul nu a promovat toate disciplinele obligatorii şi opţionale, dar a obţinut numărul de credite minim necesar continuării studiilor (30 de credite din 60), disciplinele nepromovate pot fi parcurse din nou de student, cu taxa, în anul universitar următor, cu toate activităţile didactice aferente nepromovate (curs, seminar, laborator, proiect etc. dacă nu s-a obţinut media 5 la activitatea pe parcurs sau numai examenul). Media generală anuală se calculează ca o medie a notelor obţinute la disciplinele contractate ponderate cu creditele aferente disciplinei respective.

ÎNTRERUPEREA STUDIILOR

Întreruperea studiilor se poate face numai la începutul anului universitar în perioada în care se încheie contractele de studii. Numărul de întreruperi şi durata acestora sunt în concordanţă cu reglementările legale în vigoare. Întreruperea studiilor se poate face şi în timpul anului universitar dacă există motive independente de voinţa studentului (certificat medical prelungit, caz de forţă majoră).

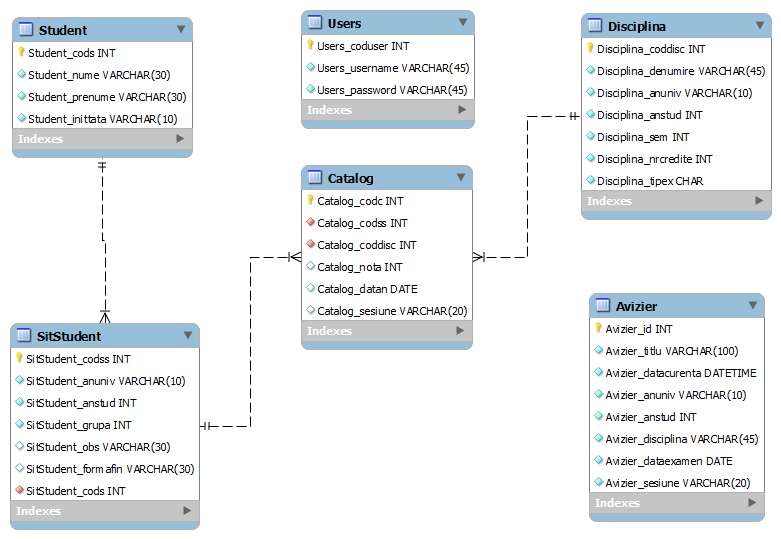
# CAPITOLUL 3

# PROIECTAREA SISTEMULUI

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

### **3.1.1. Formularea scopului**

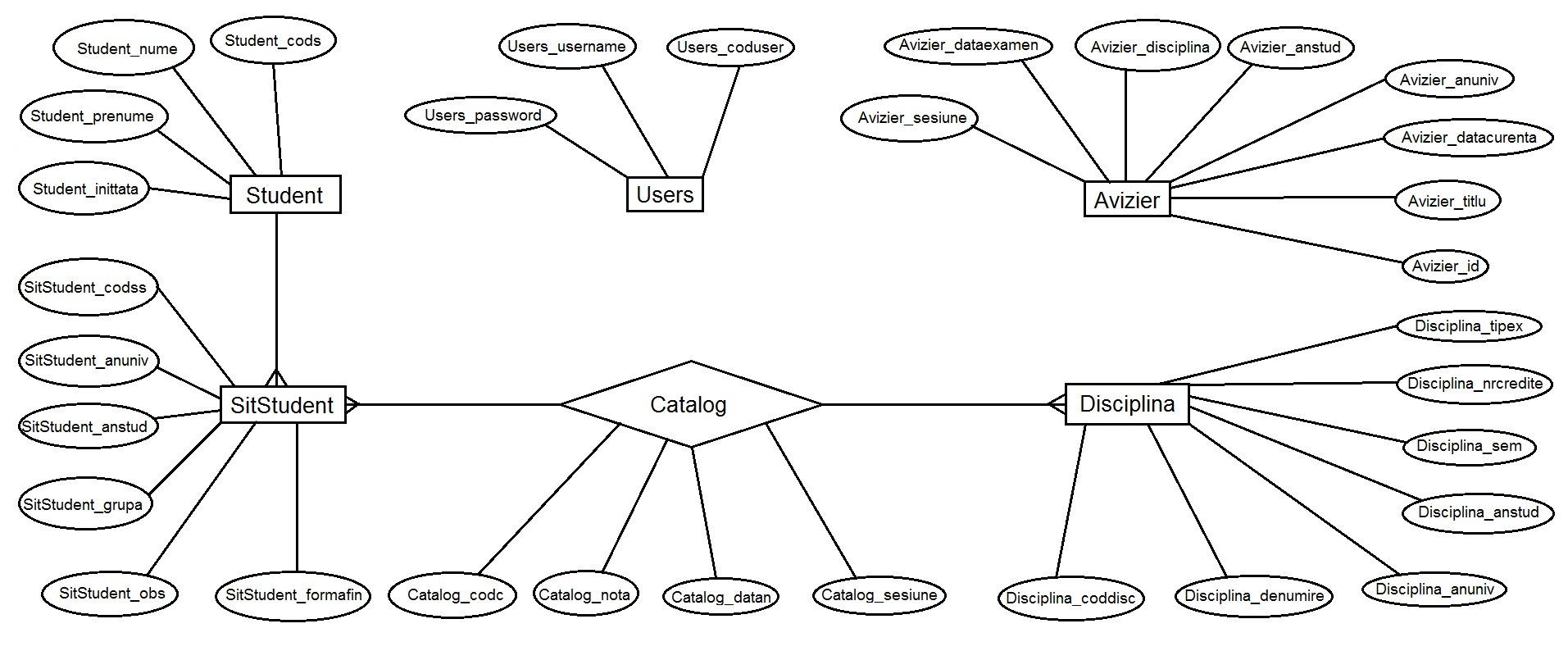
Baza de date conţine şase tabele, majoritatea accesibile doar din partea web a aplicaţiei. Scopul bazei de date este de a stoca datele astfel încât să se poată obţine informaţii despre studenţi, activitatea acestora, precum şi rezultatele obţinute în urma examinării.



Fiecare tabelă are un scop bine stabilit:

* **Student** - tabela care conţine datele de identificare a studenţilor din fiecare an universitar. Această tabelă se populează o dată cu înscrierea unui student în primul an de studiu;
* **SitStudent -** tabela ce conţine informaţii despre fiecare student pentru fiecare an universitar. La începerea unui an universitar unei entităţi de tip student îi corespunde o înregistrare în această tabelă. Legătura dintre student şi situaţie student este de tip 1:M şi se realizează prin cheia străină SitStudent\_cods, care reprezintă codul studentului, în această tabelă;
* **Disciplina -** tabela care stochează disciplinele din fiecare an universitar;
* **Catalog -** tabela ce conţine detalii legate de notele obţinute de studenţi la disciplinele studiate. Aceasta tabelă creează legătura M:M între entităţile SitStudent şi Disciplină, prin cheile străine Catalog\_codss şi Catalog\_coddisc, care reprezintă codul aferent situaţiei unui student şi codul disciplinei;
* **Avizier -** tabela care stochează informaţii referitoare la postările care se efectuează în cadrul avizierului pentru studenţi;
* **Users** - tabela care înregistrează utilizatorii ce au acces în secţiunea privată a aplicaţiei.

### **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 **Student**

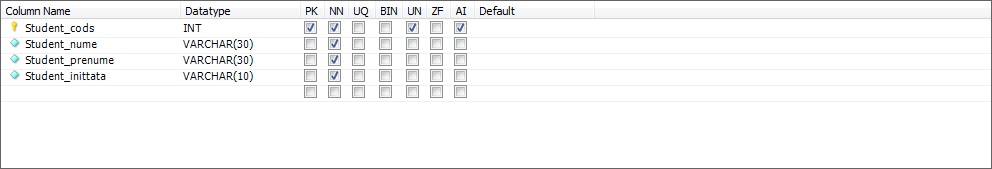


Tabela **SitStudent**

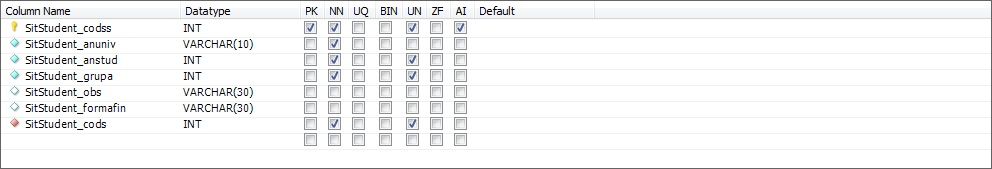


Tabela **Disciplina**

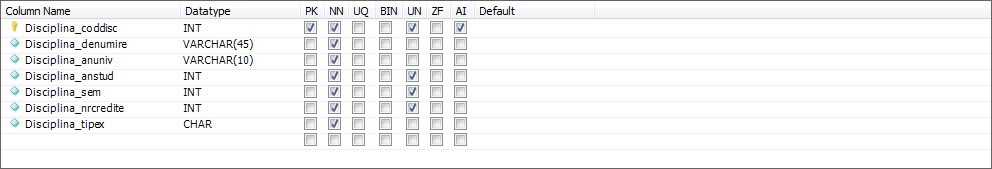


Tabela **Catalog**

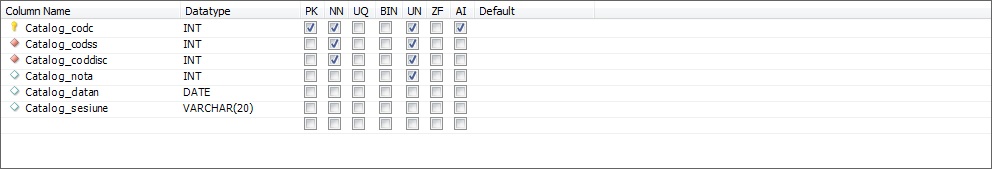


Tabela **Avizier**

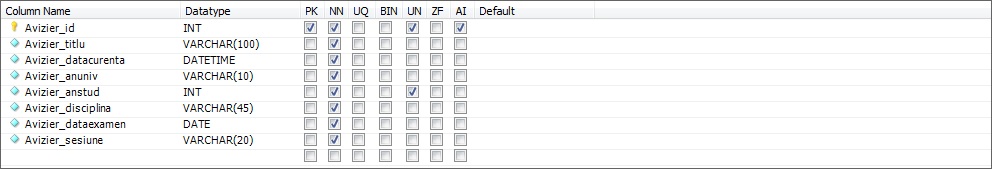
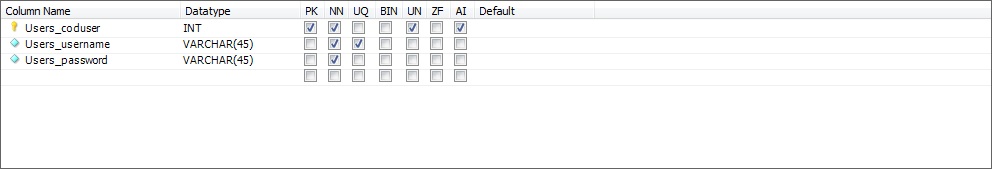


Tabela **Users**



## **3.2. PROIECTAREA APLICAŢIEI**

### **3.2.1. Arhitectura aplicaţiei**

În principiu arhitectura aplicațiilor este structurată pe următoarele trei nivele:

Nivelul de prezentare

HOMEE

Browser

Nivelul de logică a aplicației

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

Nivelul de date

1.Baza de date

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

2.Serverul de aplicație

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

3.Interfața cu utilizatorul

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

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

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

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

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

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

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

### **3.2.2. Şirul evenimentelor**

Această etapă constă în crearea interfeţei cu utilizatorul precum şi a funcţionalităţilor care folosesc baza de date.

Aplicaţia urmăreşte să conţină următorul şir de evenimente:

1. Accesarea secţiunii publice pentru vizualizarea rezultatelor examinării de catre utilizatori;

2. Autentificarea utilizatorilor: interogarea bazei de date şi salvarea în sesiune a utilizatorului. Afişarea pe pagină a utilizatorului autentificat;

3. Importarea fişierelor Excel: preluarea datelor din fişierele încărcate şi stocarea informaţiei în baza de date;

4. Inserarea notelor:

* selectarea criteriilor pentru listarea studenţilor care urmează să fie notaţi;
* selectarea notei din combo box pentru fiecare student şi salvarea acesteia în baza de date;
* vizualizarea notelor inserate în format printabil şi postarea rezultatelor la avizierul studenţilor;

5. Vizualizarea unor situaţii şcolare în urma selectării criteriilor specificate;

6. Vizualizarea unor grafice care reflectă situaţia studenţilor pe baza informaţiilor preluate din baza de date

# CAPITOLUL 4

# IMPLEMENTAREA APLICAŢIEI

## **4.1. NOŢIUNI TEORETICE**

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

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

La baza funcţionării web-ului stau trei standarde:

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

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

### **4.1.2. Protocol HTTP**

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

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

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

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

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

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

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

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

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

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

Pentru paginile dinamice procedura este următoarea:

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

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

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

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

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

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

7. Interpretorul Java execută codul primit

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

9. MySQL returnează rezultatele înapoi inbterpretorului Java

10. Interpretorul Java returnează rezultatele server-ului web

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

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

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

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

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

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

### **4.1.5. Hypertext**

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

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

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

## **4.2. TEHNOLOGII FOLOSITE**

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

### **4.2.1. Firebug**

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

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

### **4.2.2. Server-ul Apache**

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

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

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

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

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

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

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

### **4.2.3 Sistemul de baze de date MySQL**

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

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

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

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

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

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

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

Cele mai uzuale operaţii cu bazele de date sunt:

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

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

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

Alţi modificatori sunt:

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

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

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

## **4.3. PROGRAMARE WEB**

### **4.3.1. HTML**

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

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

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

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

Un document HTML are de obicei umătoarea componenţă:

1. Versiunea HTML a documentului

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

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

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

Tipurile de marcare în limbajul HTML sunt:

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

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

<html>

<head>

<title>TITLUL</title>

</head>

<body>

CONŢINUT

</body>

</html>

Exemple de tag-uri:

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

OBSERVAŢIE

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

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

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

### **4.3.2. CSS**

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

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

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

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

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

**Cod:**

<html>

<head>

<style>

b {color:red;}

i{color:blue;}

</style>

</head>

<body>

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

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

</body>

</html>

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

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

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

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

selector { proprietate:valoare}

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

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

Exemplu:

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

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

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

Exemplu de selectori CSS:

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

#### 4.3.2.1. Diferență între CSS și HTML

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

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

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

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

#### 4.3.2.2. Avantajele CSS-ului

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

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

### **4.3.3. SERVLET**

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

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

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

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

Caracteristicile principale ale servlet-urilor se referă la următoarele:

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

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

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

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

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

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

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

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

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

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

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

O pagină JSP cuprinde în structura sa:

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

### **4.3.5. JAVASCRIPT**

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

#### 4.3.5.1. Ajax

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

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

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

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

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

Exemplu de script:

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

$(document).ready(function(){ $('.stergere))

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

listareNote = **function**(an\_univ, an\_studiu, discipline, sesiune, datanotarii) {

$.get("listareNote.jsp", {

anUniv : an\_univ,

anStudiu : an\_studiu,

disciplina : discipline,

sesiune : sesiune,

dataNotarii : datanotarii

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

$('#listare\_note').html(data);

});

}

$(**function**() {

$('#bt\_Afisare').click(

**function**() {

listareNote($('#an\_univ').val(), $('#an\_studiu').val(), $('#discipline').val(), $('#sesiune').val(), $('#datanotarii').val());

});

});

</script>

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

Exemplu de comenzi pentru crearea tabelelor:

CREATE TABLE IF NOT EXISTS `Licenta`.`Catalog` (

`Catalog\_codc` INT UNSIGNED NOT NULL AUTO\_INCREMENT ,

`Catalog\_codss` INT UNSIGNED NOT NULL ,

`Catalog\_coddisc` INT UNSIGNED NOT NULL ,

`Catalog\_nota` INT UNSIGNED NULL ,

`Catalog\_datan` DATE NULL ,

`Catalog\_sesiune` VARCHAR(20) NULL ,

PRIMARY KEY (`Catalog\_codc`) ,

INDEX `fk\_SitStudent\_has\_Plan\_Plan1` (`Catalog\_coddisc` ASC) ,

INDEX `fk\_SitStudent\_has\_Plan\_SitStudent1` (`Catalog\_codss` ASC) ,

CONSTRAINT `fk\_SitStudent\_has\_Plan\_SitStudent1`

FOREIGN KEY (`Catalog\_codss` )

REFERENCES `Licenta`.`SitStudent` (`SitStudent\_codss` )

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_SitStudent\_has\_Plan\_Plan1`

FOREIGN KEY (`Catalog\_coddisc` )

REFERENCES `Licenta`.`Disciplina` (`Disciplina\_coddisc` )

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

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

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

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

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

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

**static** String *url*="jdbc:mysql://localhost:3306/licenta";

**static** String *userName* = "root";

**static** String *password* = "root";

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

*con* = DriverManager.*getConnection*(*url*,*userName*,*password*);

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

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

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

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

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

Prima pagină a aplicaţiei este *index.jsp*. În acestă pagină utilizatorul are ca opţiuni accesarea secţiunii "Avizier studenţi", la click pe butonul "Avizier studenţi", sau secţiunea de logare, care permite autentificarea, în urma interogării bazei de date cu datele specificate de utilizator.

După logare este salvat în sesiune numele utilizatorului, după care se face interogarea. Pentru realizarea acestui lucru am creat un obiect de tip *UserBean*, după care am apelat metoda login() de verificare propriu-zisă a autenticităţii utilizatorului, care se regăseşte în clasa *UserDAO*. Numele utilizatorului apare în bara de meniuri, de unde acesta are posibilitatea de a schimba parola contului, fapt realizat prin intermediul clasei *EditProfileServlet*, sau de a ieşi din aplicaţie, prin operaţia de delogare realizată în clasa *LogoutServlet*.

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

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

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

La rândul lor aceste pagini apelează alte fişiere jsp sau clase java de tip servlet, pentru acţiunile de pe server. *importFile.jsp* conţine două formulare pentru încărcarea de fişiere excel.

<form id=*"ImportDiscipline"* action=*"UploadDiscipline"* method=*"post"* enctype=*"multipart/form-data"*>

<form id=*"ImportStudenti"* action=*"UploadStudenti"* method=*"post"* enctype=*"multipart/form-data"*>

Cele doua formulare au ca acţiune două servlet-uri UploadDiscipline şi UploadStudenti, care preiau datele din fişierele excel încărcate şi le salvează în baza de date. Încărcarea fişierelor excel a fost posibilă prin specificarea proprietăţii enctype=*"multipart/form-data"*> în cele două formulare.

În secţiunea "Inserare note" realizată prin intermediul fişierului *noteStudenti.jsp* am creat un formular care permite inserarea criteriilor în vederea listării studenţilor pentru notare. În această pagină am folosit funcţii jquery pentru listarea disciplinelor din anul de studiu selectat, pentru listarea studenţilor dupa criteriile specificate şi pentru selectarea datei calendaristice dintr-un datepicker. Fiecărui student listat i se poate acorda o notă prin selectarea acesteia dintr-un combo box şi la click pe butonul "Salvare" se inserează nota în baza de date. Se oferă posibilitatea ca la terminarea inserării notelor să se genereze rezultatele în format pdf prin click pe butonul "Vizualizare" şi postarea acestora la avizierul studenţilor la click pe butonul "Postare". Pentru generarea rezulatelor în format pdf am utilizat libraria iText, în cadrul servlet-ului *VizualizareNote,* apelat la rândul său la evenimentul submit pe butonul de "Vizualizare". Postarea la avizierul studenţilor se realizează prin apelul fişierului *postareAvizier.jsp*, care la click pe butonul "Postare" inserează în baza de date o nouă înregistrare cu detaliile care vor fi postate la avizier.

În secţiunea "Situaţii" sunt prezentate, prin intermediul a trei pagini jsp: *situaţieGenerală.jsp*, *situaţieStudent.jsp* şi *situaţieRestanţeDisciplină.jsp*, care la rândul lor apelează servlet-urile aferente fiecărei pagini: *VizualizareSituaţieGenerală*, *VizualizareSituaţieStudent* şi *VizualizareSituaţieRestanţeDisciplină,*  trei rapoarte pe baza datele stocate cu referire la situaţia generală a creditelor, mediei aritmetice şi mediei ponderate a studenţilor dintr-un an universitar şi an de studiu specificate îin formularul inclus în *situaţieGenerală.jsp,* la situaţia notelor unui student pe parcursul anilor de studiu specificat în formularul prezent în *situaţieStudent.jsp* şi la situaţia restanţierilor pe fiecare an universitar la o disciplina selectată în formularul paginii *situaţieRestanţeDisciplină.jsp.*

În ultima secţiune "Statistici" am realizat prin intermediul paginilor jsp *restanţieriPeAn.jsp* şi *restanţieriPeDiscipline.jsp* două statistici grafice care reflectă numărul studenţilor restanţieri la disciplinele dintr-un an universitar şi an de studiu, criterii selectate din formularul paginii *restanţieriPeAn.jsp,* şi numărul studenţilor restanţieri la o disciplină selectată din pagina *restanţieriPeDiscipline.jsp* pe toţi anii universitari în care s-a studiat aceasta. Pentru generarea graficelor specificate, am utilizat librăria jfreechart în cadrul celor două servlet-uri apelate de paginile jsp de mai sus.

# CAPITOLUL 5

# TESTARE ŞI EVALUARE

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

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

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

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

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

## **5.1. TESTAREA**

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

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

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

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

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

Strategiile de testare se concentrează pe funcţionalitatea şi utilizabilitatea produsului.

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

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

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

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

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

Testarea acestei aplicaţii web a implicat şi:

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

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

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

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

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

Testarea acestei arhitecturi presupune trei niveluri diferite:

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

## **5.2. ASPECTE POZITIVE**

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

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

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

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

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

# CAPITOLUL 6

# CONCLUZII

**Lucrarea de faţă întruneşte cerinţele impuse de un sistem de urmărire a situaţiei şcolare a studfenţilor în mediul universitar, punând accent pe rezolvarea comunicării în timp real. Aceasta implementează o platformă disponibilă cadrelor didactice şi studenţilor cu scopul furnizării informaţiilor referitoare la rezultatele examinării şi generarea unor statistici pe baza datelor furnizate.**

**Este un sistem complet de monitorizare a situaţiei şcolare care utilizează unelte web, iar prin simplitatea pe care o oferă poate fi folosit cu încredere în mediul pentru care a fost proiectat şi îşi îndeplineşte cu stricteţe sarcinile.**

**În concluzie, prin realizarea acestei aplicaţii se rezolvă problematica cumulării sistemelor de notare şi monitorizare a studenţilor din mediul universitar, fiind o unealtă importantă pentru dezvoltarea şi îmbunătăţirea promovabilităţii şi calităţii profesionale.**

# MANUAL DE UTILIZARE

Acest capitol vine în ajutorul viitorilor utilizatori ai aplicaţiei prin prezentarea interfeţei şi a modului de utilizare al acesteia. Acţiunile fiecărui utilizator al aplicaţiei sunt prezentate în ordinea în care vor apărea în cursul navigării.

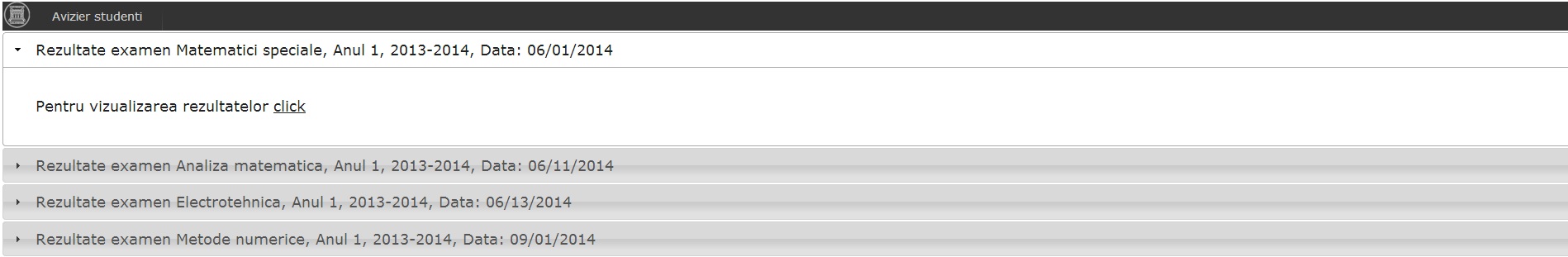
Aplicaţia este constituită din două secţiuni majore: o secţiune publică, destinată în principal studenţilor pentru informarea acestora privitor la rezultatele obţinute la examene şi o secţiune privată pentru cadrele ddidactice din departamentul de "Calculatoare şi Tehnologia Informaţiei".

Elementul de legatură între cele doua secţiuni este constituit de pagina Web principală, care pe lângă funcţia de autentificare pentru personalul didactic, pentru accesul în secţiunea privată, are şi rolul de a oferi posibilitatea accesării secţiunii publice.



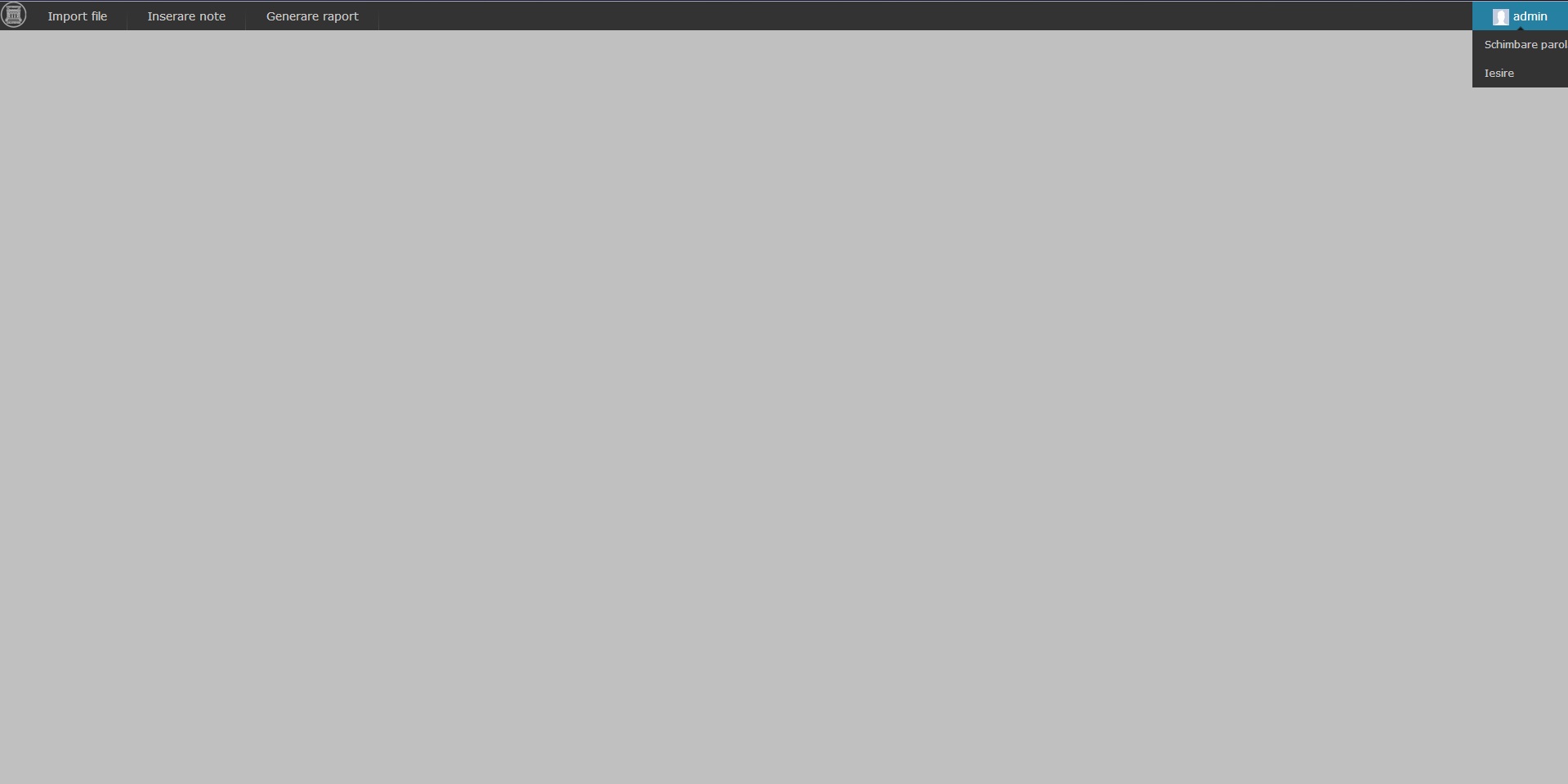
**Figura 7.1.** Pagina de start

Secţiunea publică este alcătuită din header, partea superioară a paginii, care conţine sigla universităţii împreună cu titlul paginii "Avizier studenţi" şi body-ul paginii, care conţine o funcţie JQuery pentru crearea avizierului. Pentru vizualizarea unei secţiuni din pagină, utilizatorul trebuie sa dea click pe una dintre acestea, care se va deschide şi va oferi posibilitatea accesării unui link către fişierul pdf cu rezultate.



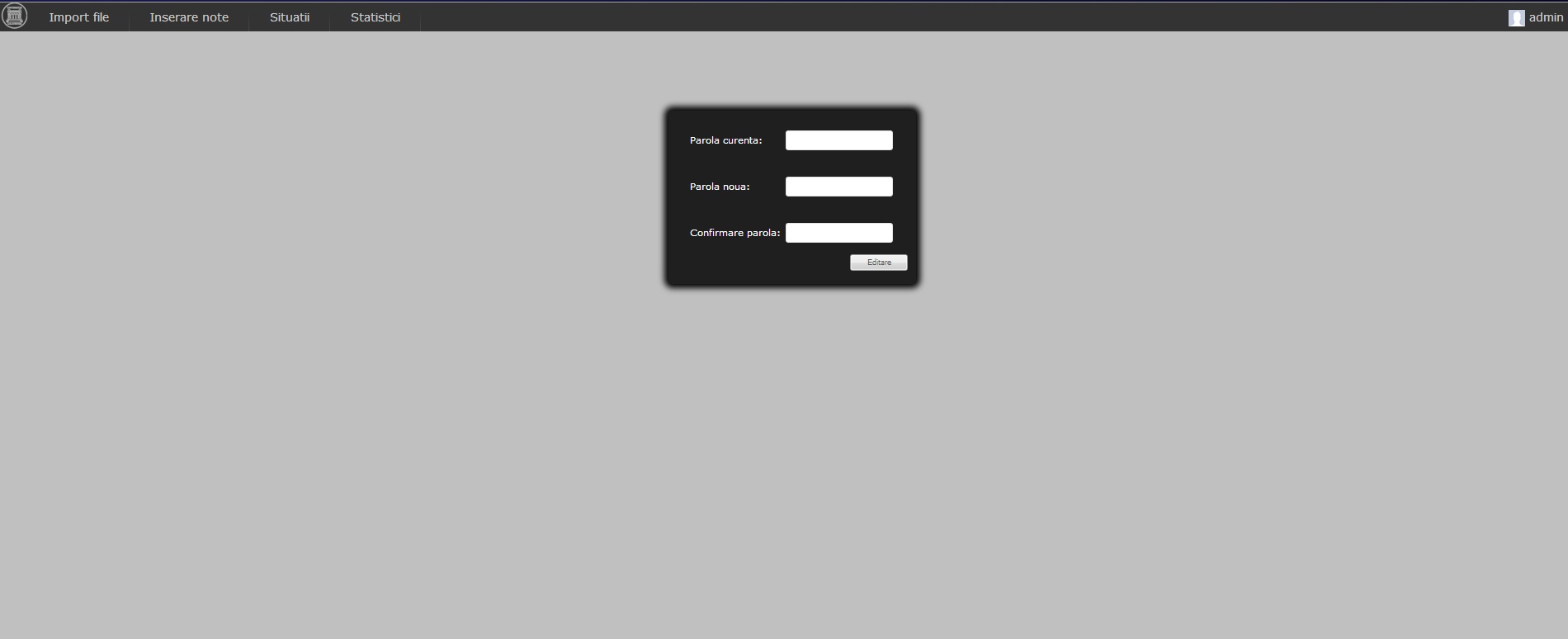
**Figura 7.2.** Avizier studenţi

Interfaţa utilizatorului autentificat prezintă în partea superioară header-ul alcătuit din sigla universitaţii şi subsecţiunile Import file, Inserare note, Situaţii şi Statistici.



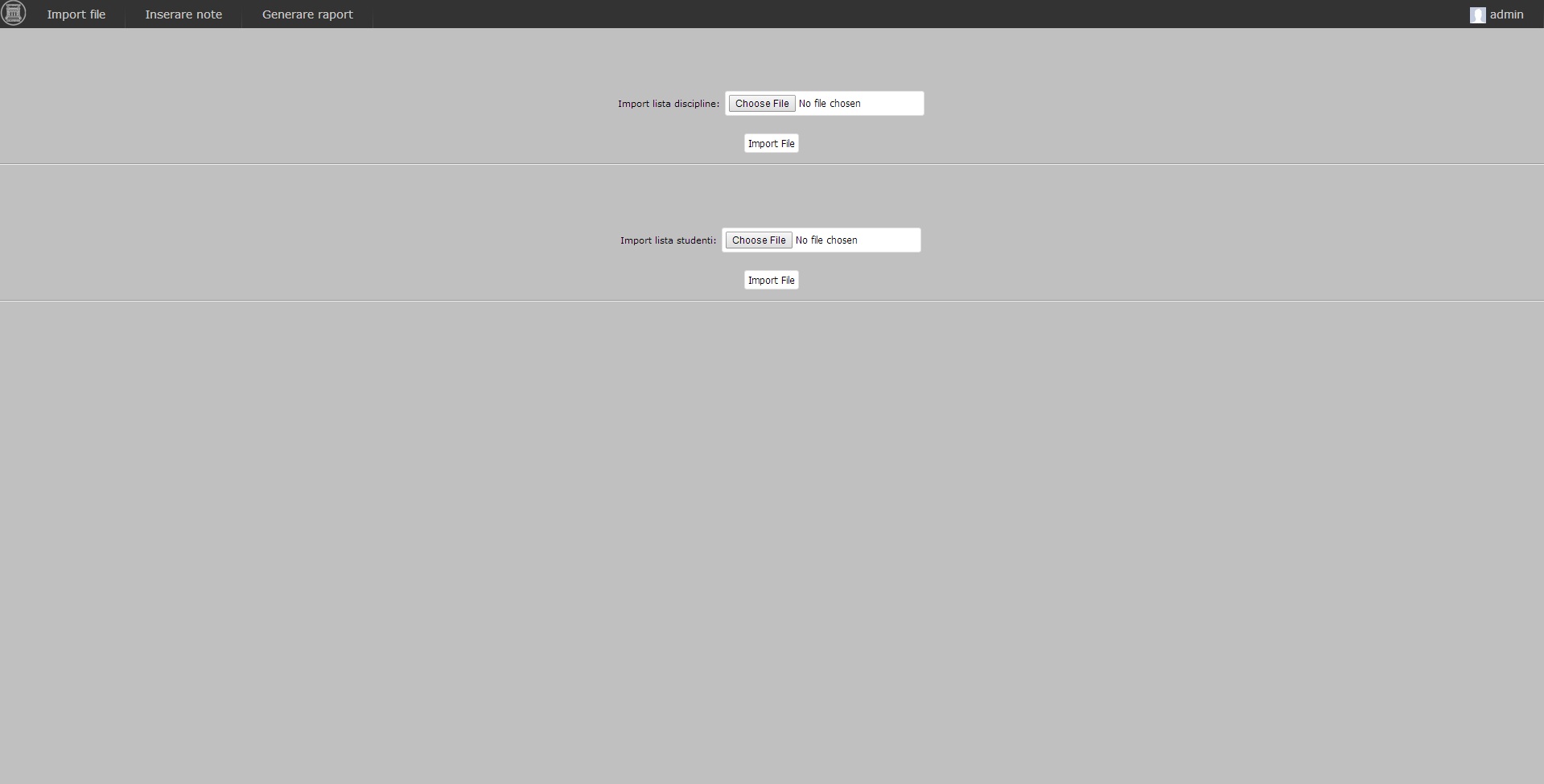
**Figura 7.3.** Utilizator autentificat

În partea dreaptă a header-ului se regăseşte secţiunea destinată utilizatorului autentificat, caruia i se oferă posibilitatea de a-şi schimba parola şi de a ieşi din aplicaţie.



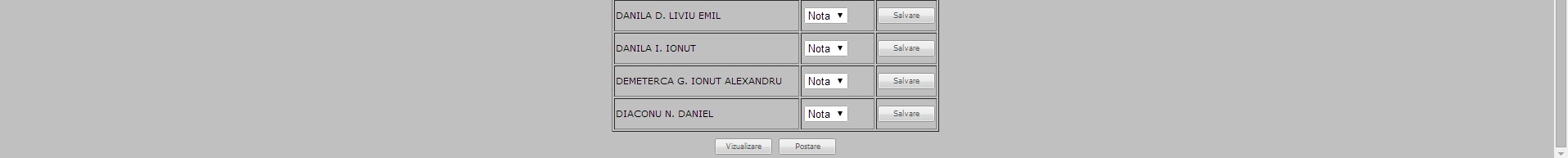
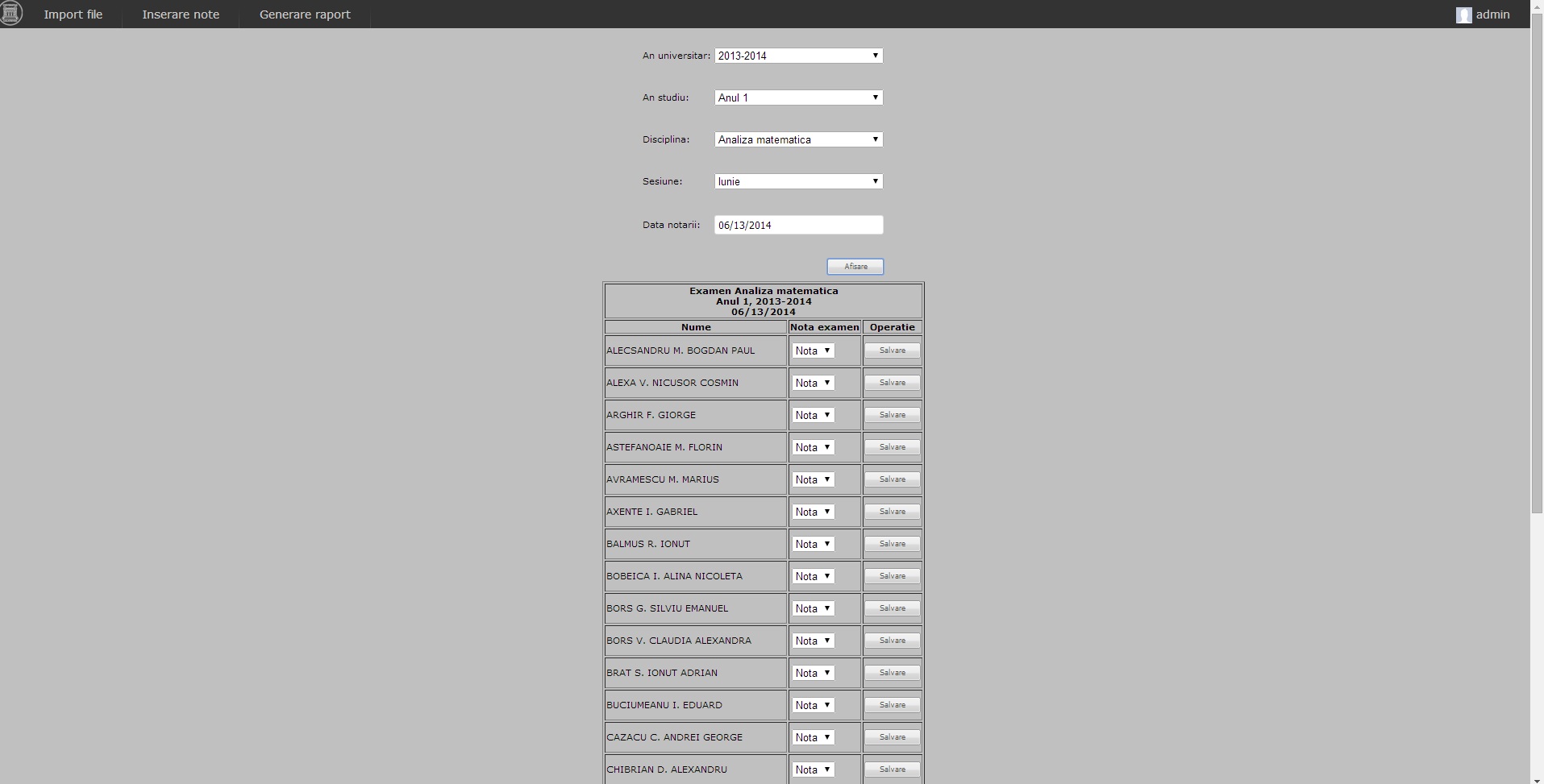
**Figura 7.4.** Schimbare parolă

Accesarea la click a secţiunii "Import file" deschide în body-ul paginii două file choosere, prin care utilizatorul poate încărca şi stoca date referitoare la discipline şi studenţi din fişiere excel, în acestă ordine, în baza de date a aplicaţiei.

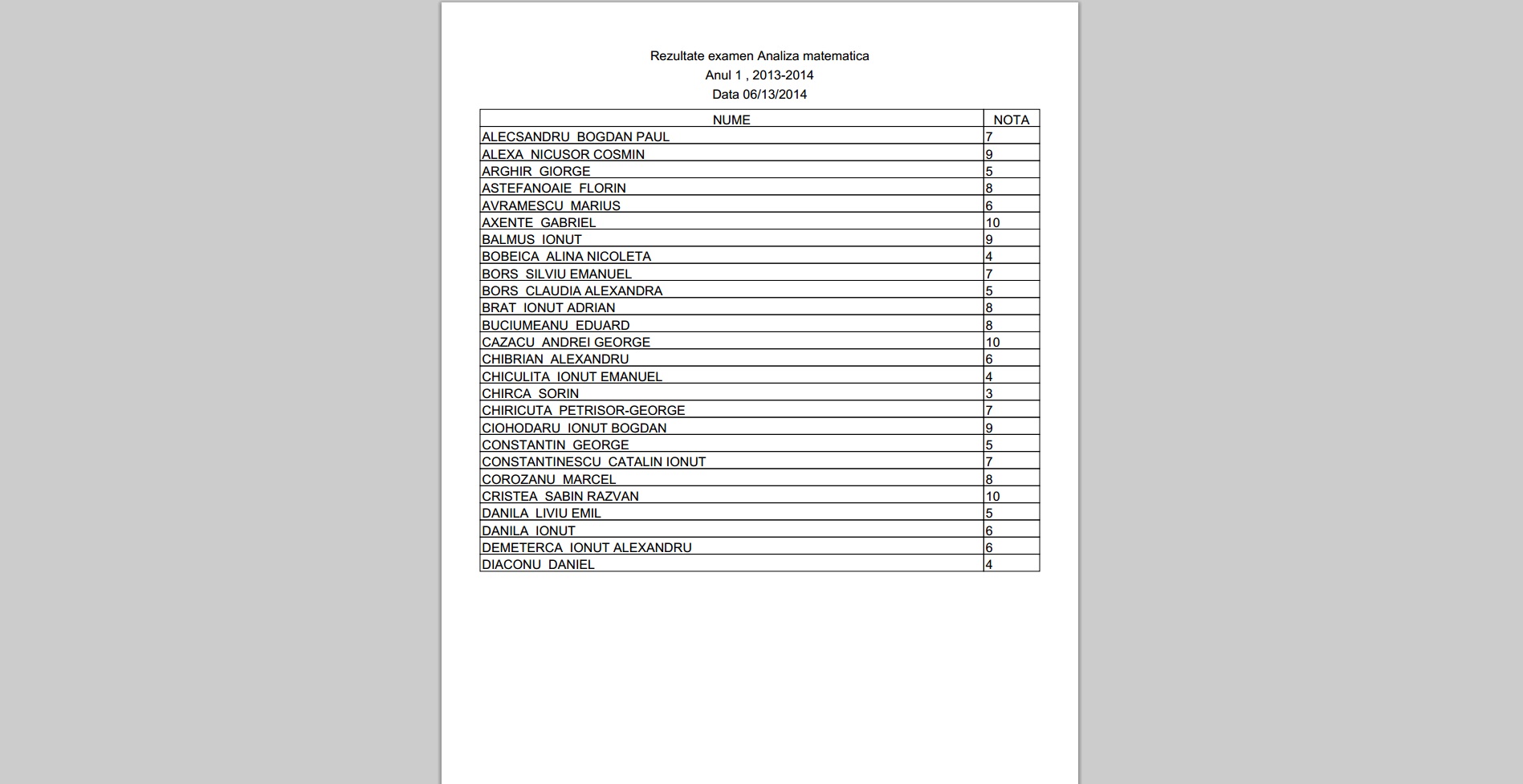


**Figura 7.5.** Import file

În secţiunea "Inserare Note" se oferă posibilitatea cadrului de didactic să filtreze studenţii după criteriile selectate prin intermediul combo box-urilor şi a date picker-ului, în vederea listării şi notării acestora, în urma examenului susţinut. După procesul de notare, cadrul didactic are ca opţiuni în partea de jos a paginii, vizualizarea în format PDF a rezultatelor la examenul pentru care au fost inserate notele anterior şi postarea acestora în secţiunea publică a aplicaţiei, la avizierul studenţilor.

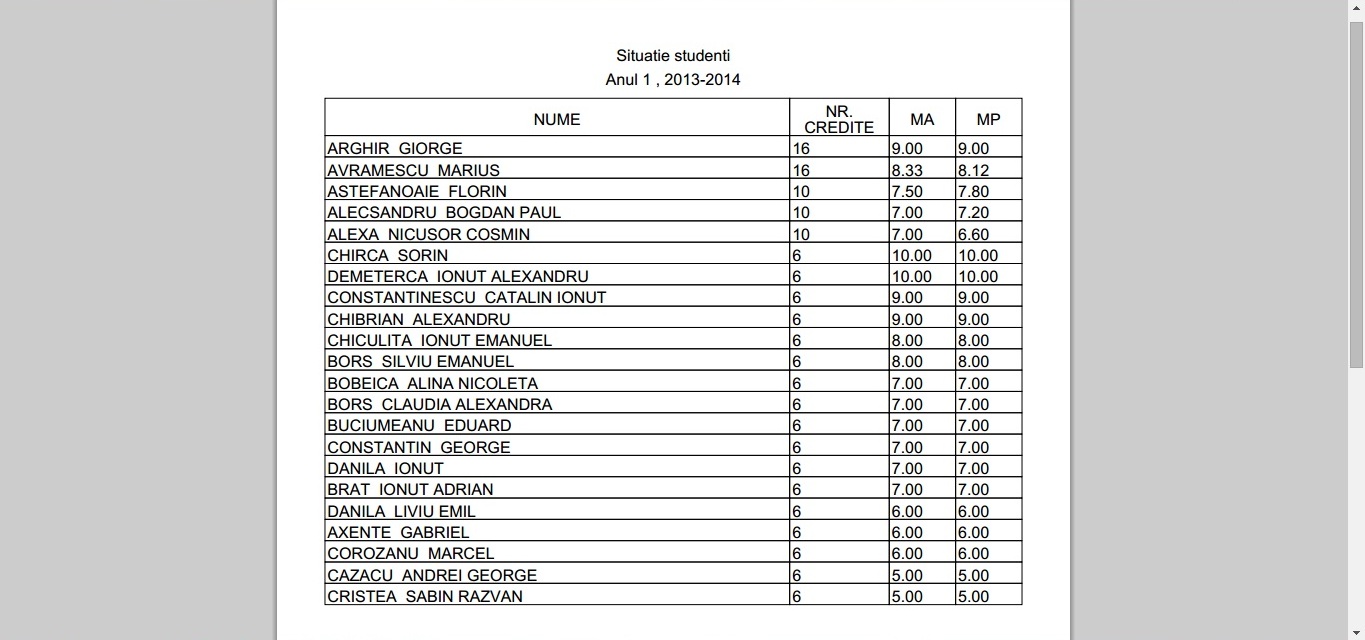
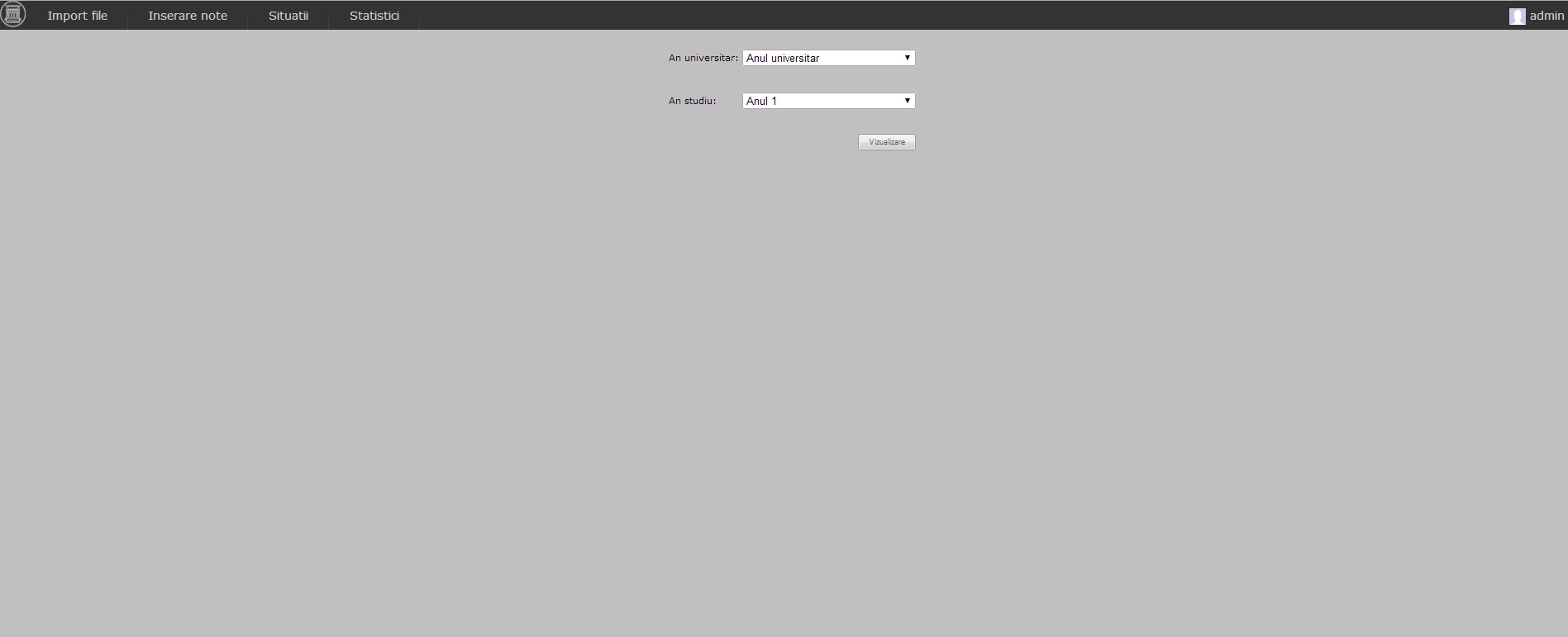


**Figura 7.6.** Inserare note

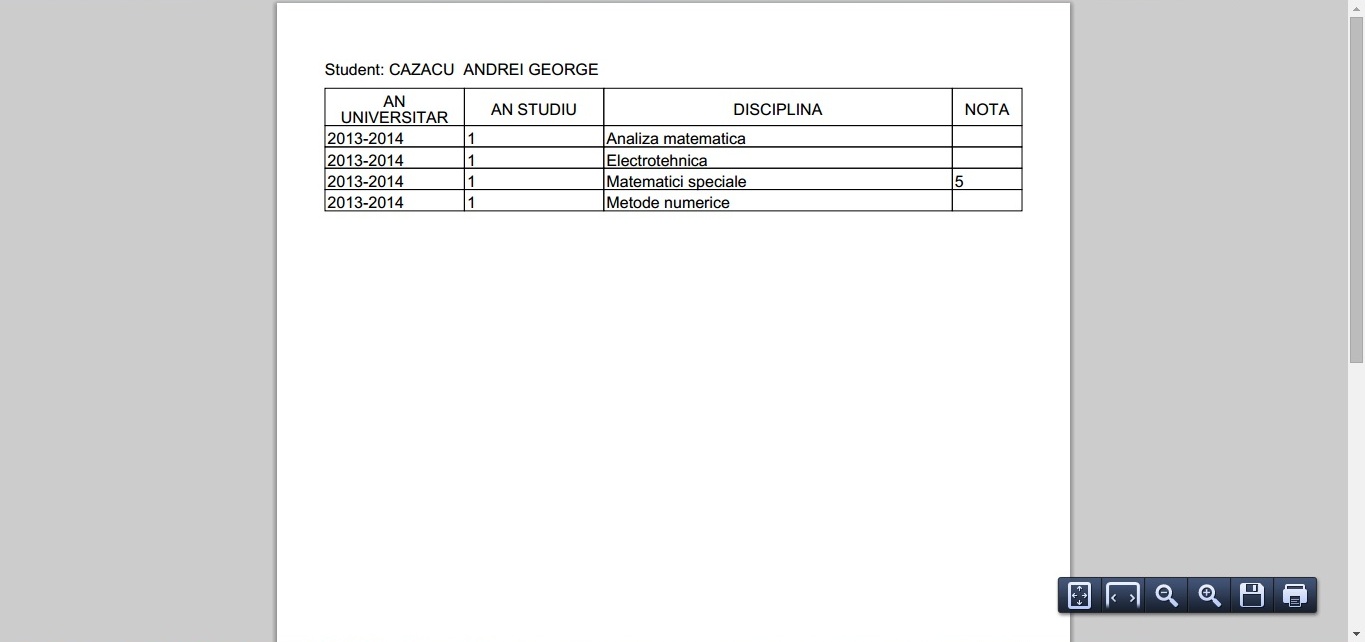
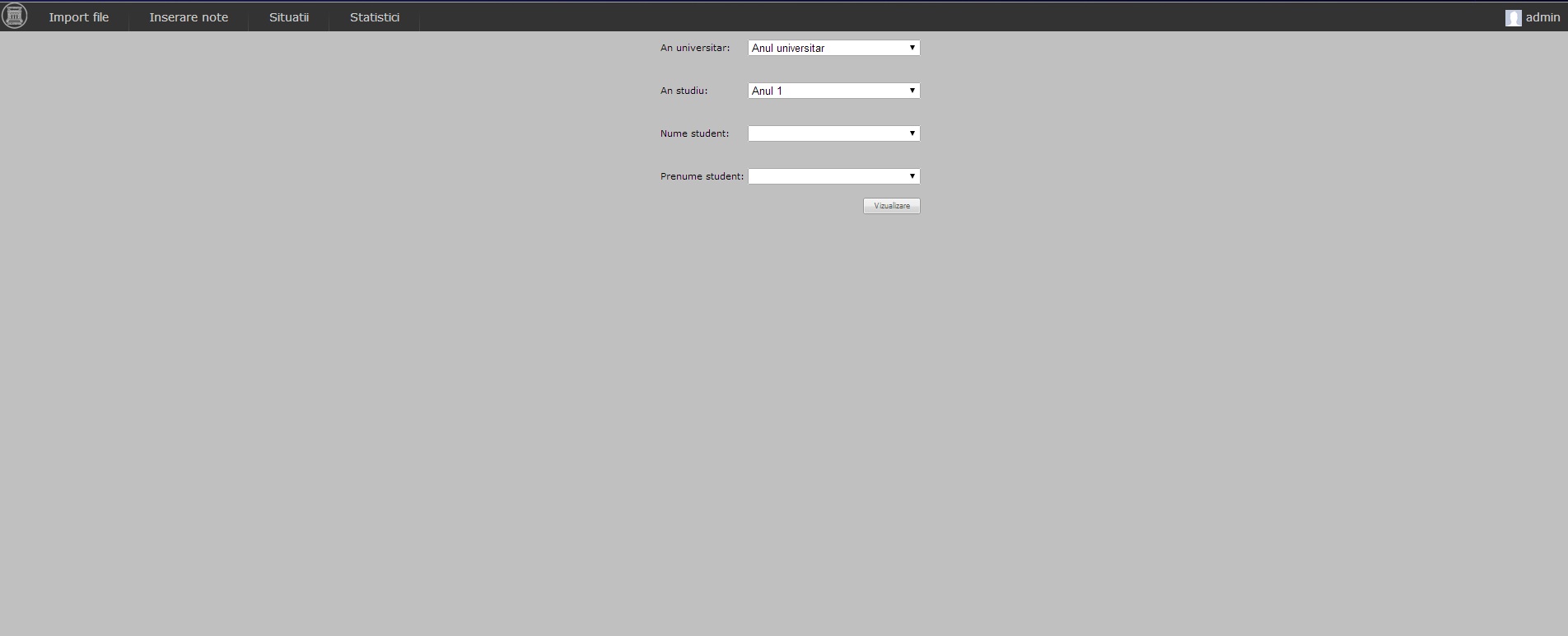


**Figura 7.7.** Vizualizare PDF

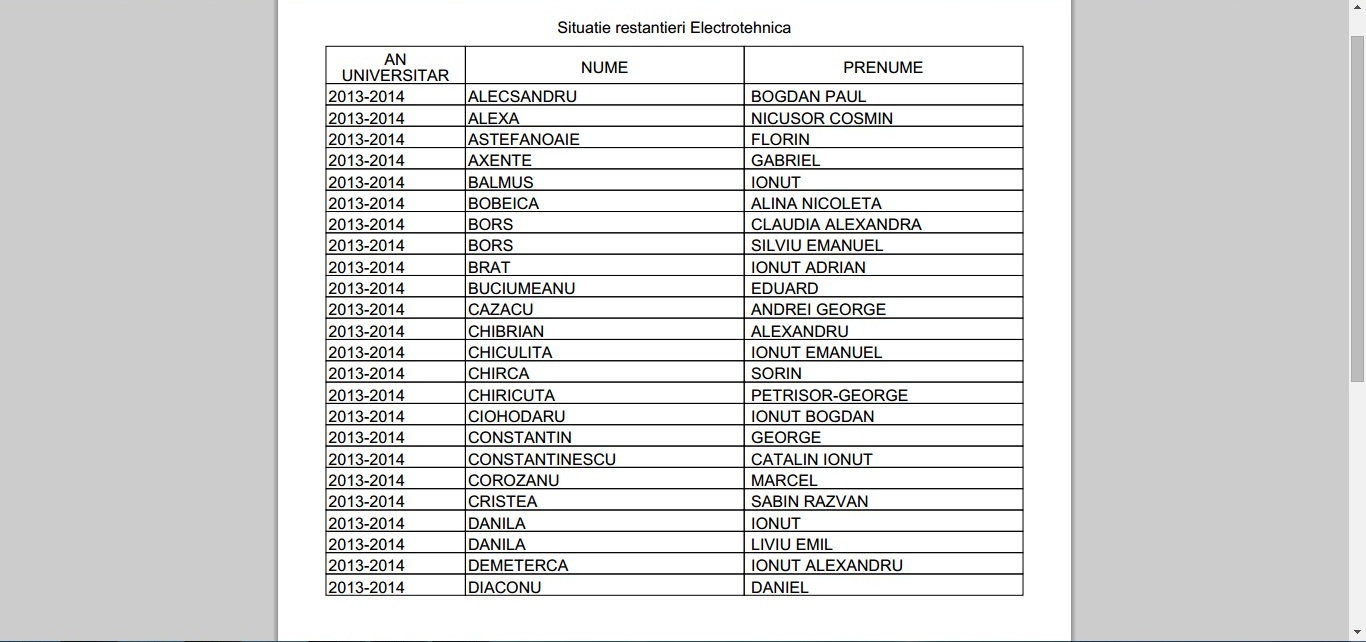
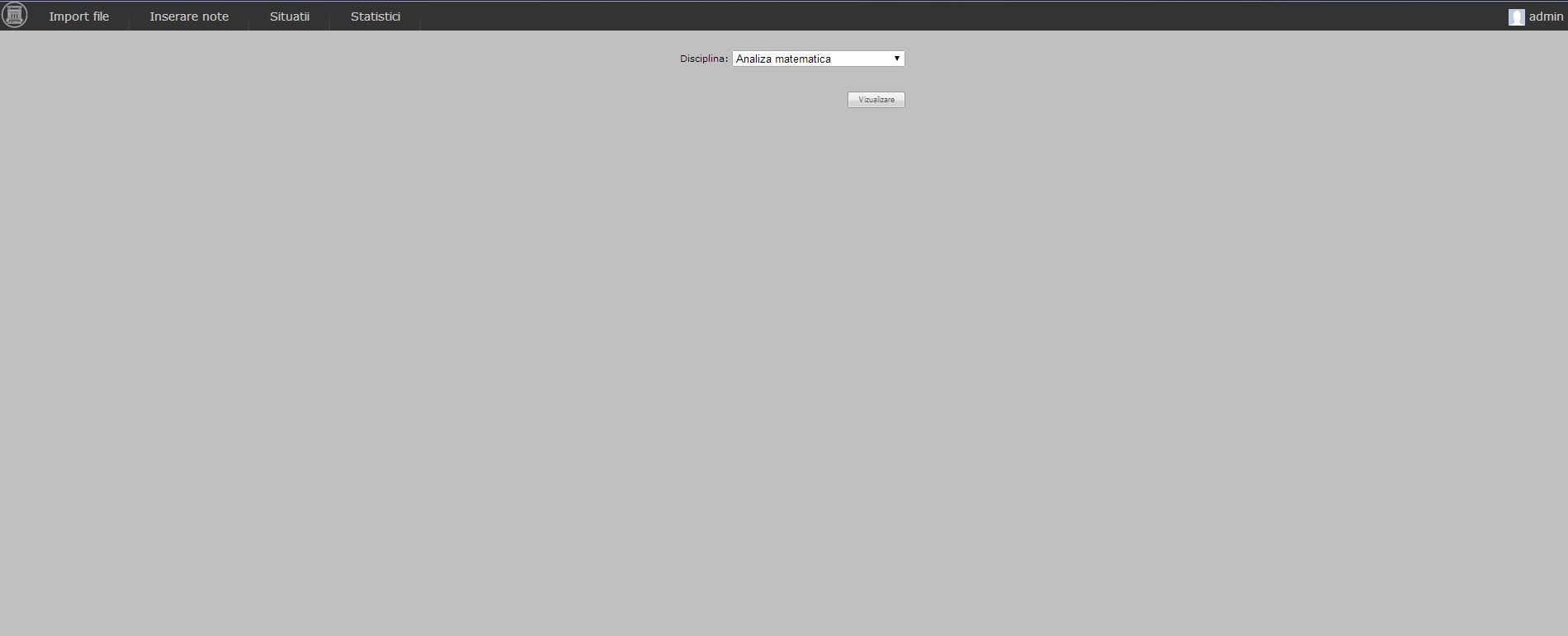
Secţiunea de situaţii ale studenţilor oferă trei variante de generare a acestora: Situaţie generală, Situaţie student şi Situaţie restanţieri pe disciplină. Pentru prima situaţie sunt selectate ca şi criterii anul universitar şi anul de studiu pentru generarea situaţiei numărului de credite, al mediei aritmetice şi ponderate pentru criteriile selectate. La accesarea situaţiei unui student trebuie indicat anul universitar, anul de studiu, numele şi prenumele studentului pentru care se doreşte vizualizarea situaţii pe an universitar şi an de studiu la disciplinele studiate. Ultima situaţie pusă la dispoziţie de aplicaţie propune ca element de intrare disiplina, selectată dintr-un combo box, iar ca element de ieşire utilizatorul va putea vizualiza lista tuturor restanţierilor la disciplina selectată pe fiecare an universitar.



**Figura 7.8.** Situaţie generală

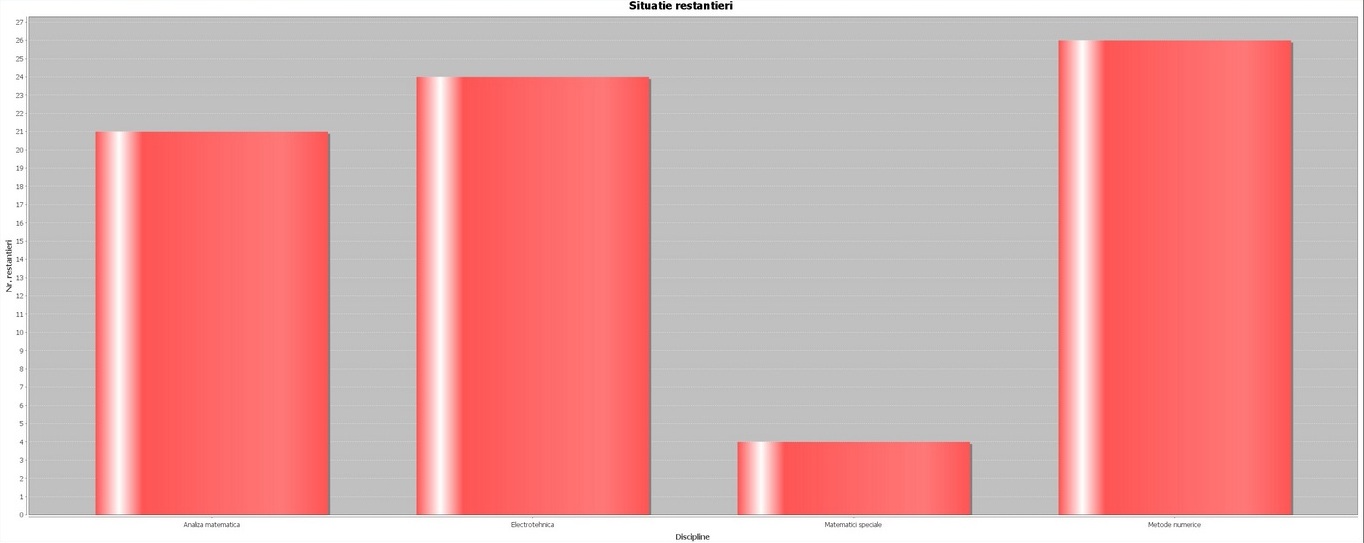
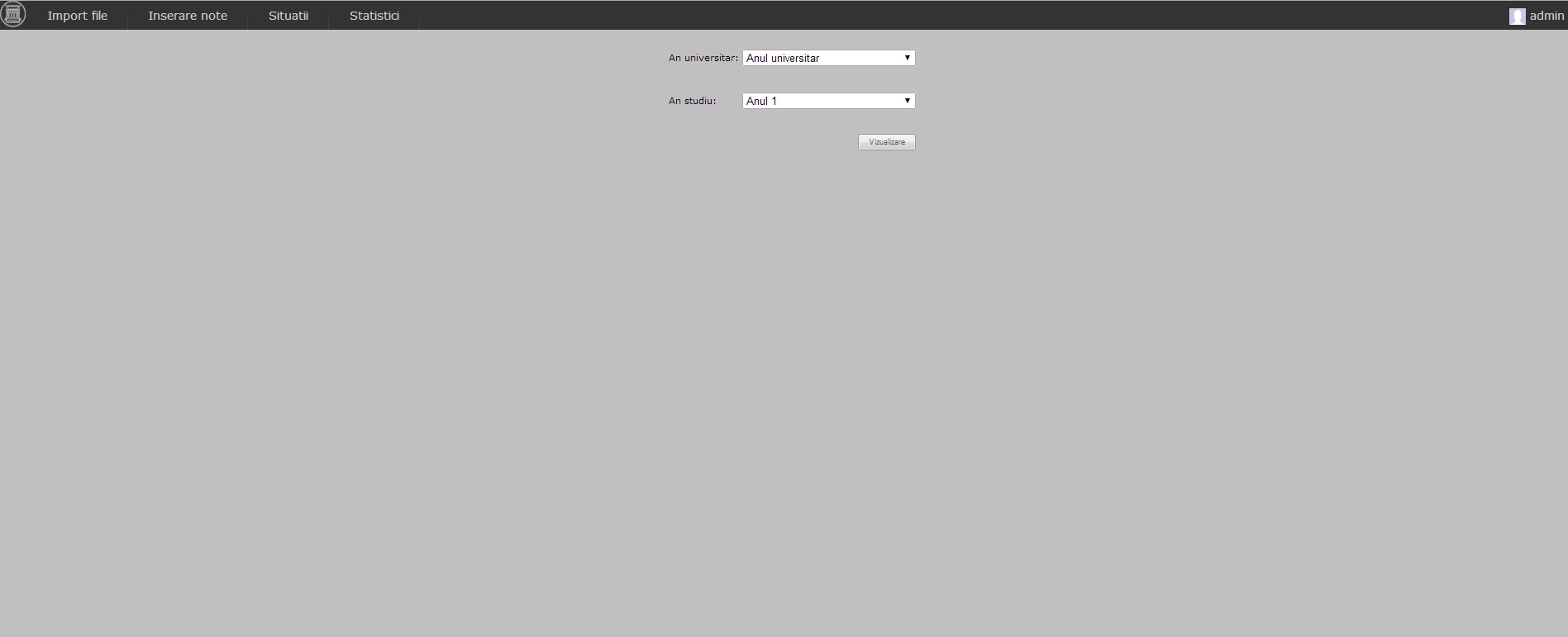


**Figura 7.9.** Situaţie student

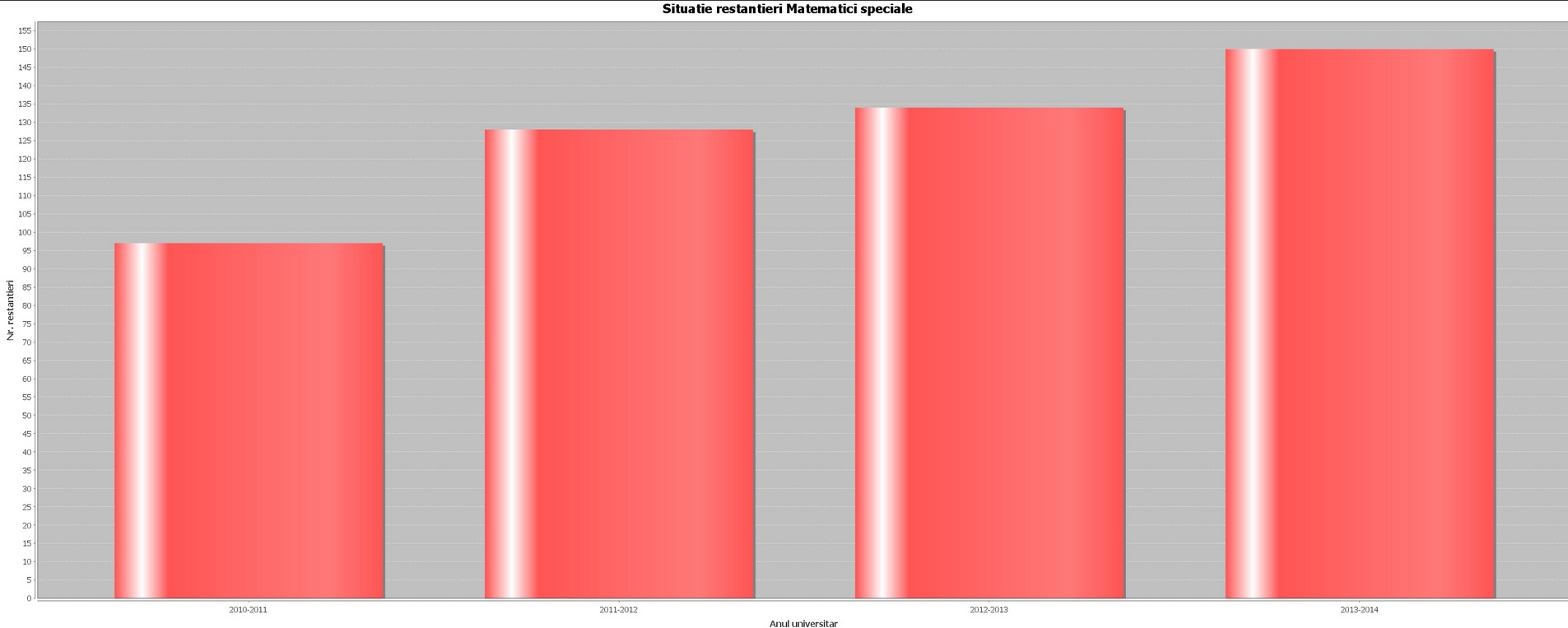
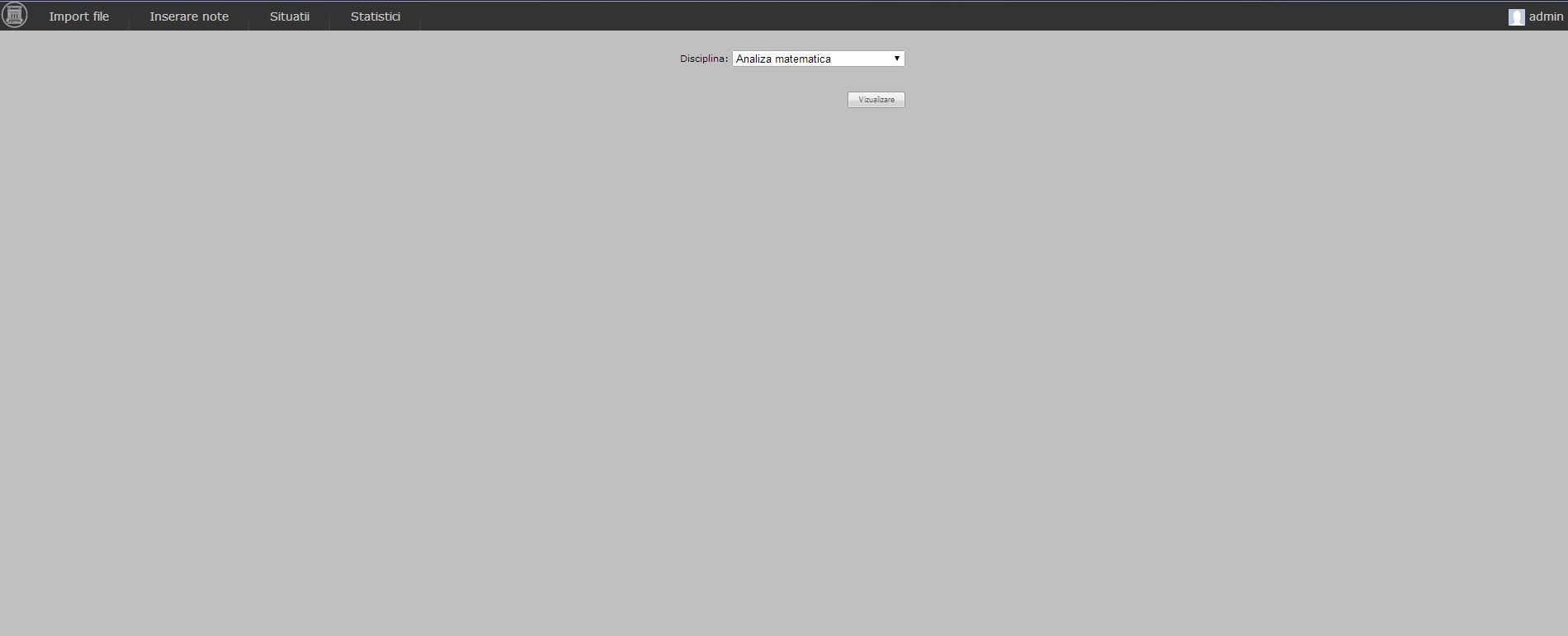


**Figura 7.10.** Situaţie restanţieri

Secţiunea de raportare a activităţii studenţilor este formată din două tipuri de rapoarte, unul pentru raportarea în format grafic a restanţierilor pe baza selecţiei de către utilizator a anului universitar şi anului de studiu, şi cel de-al doilea pentru raportarea în format grafic a restanţierilor la o disciplină selectată pe toţi anii universitari.



**Figura 7.11.** Statistică 1



**Figura 7.12.** Statistică 2

# 

# BIBLIOGRAFIE

**Cărți:**

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

**Resurse Web:**

* http://www.w3resource.com/
* http://www.w3schools.com/