



Universitatea POLITEHNICA din Bucureşti
Facultatea Automatică și Calculatoare
Departamentul Automatică și Informatică Industrială

LUCRARE DE LICENȚĂ

**Modelarea și generarea automată a orarului în
cadrul unei facultăți**

Coordonator

Prof. Dr. Ing. Anca IONIȚĂ

Absolvent

Diana DIMITRIU

2018

Cuprins

PARTEA I - CADRU GENERAL.....	4
1 INTRODUCERE.....	5
2 PREZENTAREA DOMENIULUI DIN CARE FACE PARTE LUCRAREA	6
2.1 FUNDAMENTE TEORETICE	6
2.2 STADIUL ACTUAL AL DOMENIULUI	7
2.3 APLICATII SIMILARE.....	8
2.3.1 <i>Edutimer</i>	8
2.3.2 <i>School Management Software</i>	9
3 DOCUMENTATIE TEHNICA.....	9
3.1 ECHIPAMENTE UTILIZATE	9
3.2 TEHNOLOGII SOFTWARE.....	10
3.2.1 <i>Descrierea modelelor si specificatiilor UML</i>	10
3.2.2 <i>Baze de date MS SQL Server</i>	14
3.2.2.1 <i>Structura unei baze de date relationale</i>	15
3.2.2.2 <i>Proiectarea unei baze de date</i>	16
3.2.2.3 <i>Crearea unei baze de date si a unui tabel</i>	17
3.2.2.4 <i>Stergerea si modificaea unui tabel</i>	18
3.2.2.5 <i>Inserarea, modificarea si stergerea datelor intr-o baza de date</i>	19
3.2.2.6 <i>Interrogarea unei baze de date</i>	20
3.2.2.7 <i>Interrogarea mai multor tabele</i>	22
3.2.2.8 <i>Vederi</i>	23
3.2.2.9 <i>Conexiunea cu baza de date</i>	24
3.2.3 <i>Programare orientata obiect Java</i>	25
3.2.3.1 <i>Notiuni de introductive</i>	25
3.2.3.2 <i>Structura limbajului Java</i>	26
3.2.3.3 <i>Literali, separatori si comentarii in Java</i>	26
3.2.3.4 <i>Operatori</i>	28
3.2.3.5 <i>Variabile, declarare si initializare</i>	29
3.2.3.6 <i>Vectori</i>	30
3.2.3.7 <i>Obiecte</i>	30
3.2.3.8 <i>Clase</i>	31
3.2.3.9 <i>Clasa speciala Object</i>	32
3.2.3.10 <i>Interpretarea exceptiilor</i>	32
3.2.3.11 <i>Pachete in Java</i>	33
3.2.3.12 <i>Interfete grafice cu utilizatorul</i>	34
3.2.3.13 <i>Comunicarea cu baza de date (JDBC)</i>	37
PARTEA A II-A - CONTRIBUTII ALE PROIECTULUI.....	41
4 MOTIVATIE	42

4.1	PROBLEMA ABORDATA	42
4.2	SOLUTIA PROPUZA.....	43
5	METODA DE DEZVOLTARE.....	51
5.1	ANALIZA SI SPECIFICAREA CERINTELOR	51
5.1.1	<i>Surse de informare</i>	55
5.1.1.1	<i>Structurarea bazei de date</i>	56
5.1.1.2	<i>Prezentarea tabelelor</i>	57
5.1.1.3	<i>Dezvoltarea aplicatiei Java</i>	64
5.1.2	<i>Cazuri de utilizare</i>	69
5.1.3	<i>Functionalitati</i>	71
5.1.4	<i>Restrictii si alte cerinte nefunctionale</i>	74
5.2	ELEMENTE DE PROIECTARE	76
5.2.1	<i>Arhitectura.....</i>	76
6	DESCRIEREA APlicatiei PRACTICE	77
6.1	GHID DE UTILIZARE	77
6.1.1	<i>Roluri si permisiuni</i>	78
6.1.2	<i>Rolul Student</i>	78
6.1.3	<i>Rolul Cadru didactic.....</i>	80
6.1.4	<i>Rolul Administrator</i>	85
6.2	VERIFICAREA SI VALIDAREA APlicatiei	100
6.2.1	<i>Scenarii si date de test.....</i>	101
6.2.2	<i>Metrici si indicatori de calitate</i>	104
6.2.2.1	<i>Fiabilitatea.....</i>	105
6.2.2.2	<i>Mantenanta.....</i>	105
6.2.2.3	<i>Testabilitatea.....</i>	106
6.3	REZULTATE OBTINUTE	106
6.3.1	<i>Rezultate curente vs asteptari.....</i>	106
7	CONCLUZII	107
	BIBLIOGRAFIE.....	109
	ANEXA 1. SECVENTE DE COD	110
	ANEXA 2. BAZA DE DATE	117
	ANEXA 3. FISIERE DE DATE	118

Partea I - Cadru general

1 Introducere

In lucrarea de fata am ales sa tratez subiectul modelarii si generarii automate a orarului in cadrul unei facultati deoarece consider ca aceasta reprezinta una dintre cele mai importante problematici in ceea ce priveste organizarea eficienta a unei structuri de invatamant superior si nu numai. Studenta fiind, m-am confruntat de-a lungul anilor cu situatii in care un orar afisat in prima faza a fost modificat ulterior intr-o maniera semnificativa, acest fapt datorandu-se fie imposibilitatii de a ajunge la cursuri a cadrelor didactice, fie insuficientei spatiului fizic de studiu, a salilor de curs sau de laborator si seminar. Aceste aspecte m-au determinat sa abordez prezentia tema, cunoscand totodata si faptul ca nu exista la ora actuala in cadrul Universitatii Politehnice din Bucuresti o aplicatie care sa fie folosita pentru generarea orarelor, acestea fiind elaborate de catre o parte din cadrele didactice, ceea ce reprezinta o munca anevoieasa si un consum mare de timp intr-o perioada aglomerata, aceea a inceputului de an universitar.

De asemenea, am avut in vedere si problematica flexibilitatii in structurarea unui orar, cunoscand faptul ca mici modificari ale acestuia pot avea ca efect un efort suplimentar depus pentru a le putea integra. Factorul uman, ce reprezinta "motorul" acestui proces dificil de organizare a tuturor materiilor, de-a lungul unui semestru poate introduce erori inerente ce cauzeaza verificari si reverificari, deci un adaos la consumul de timp.

Traind intr-o asa numita era a tehnologiei, in care se incearca automatizarea a tot ceea ce ne inconjoara am ales sa studiez principiul de "Campus Inteligent", prezentat in [1]. Din lucrarea citata am extras cele trei mari componente ce trebuie asociate in vederea folosirii eficiente a resurselor, acestea sunt: echipamentele, activitatile si nu in ultimul rand, oamenii. Tot din acest raport am constatat si faptul ca in prezent la Institutul Indian de Tehnologie Delhi exista deja ideea de "Campus Inteligent" ce se bazeaza pe formulare digitale, pe protejarea mediului prin utilizarea cat mai redusa a hartiei pentru munca de birou si folosirea cat mai mare a aplicatiilor software, pe organizarea eficienta a timpului, pe alocarea facila a spatiului de studiu si a personalului, pe informatii personalizate si servicii de invatare interactive si digitalizate. Exista deci o colectivitate formata din studenti, cadre didactice si personal administrative care sustin acest campus si vegheaza la crearea unui mediu academic digitalizat, un mediu sanatos ale carui concepte merita, in opinia mea, sa fie urmante.

Univeritatea Politehnica din Bucuresti, unul dintre cele mai mari campusuri universitare ale Romaniei mi-a demonstrat in cei patru ani de facultate faptul ca a adoptat forme de comunicare si de invatare inteligente, flexibile si in pas cu tendintele de digitalizare ale vremii si tinde catre transformarea dintr-o forma traditionala de invatamant superior intr-un "Campus intelligent".

Am ales sa aprofundez paradigma programare orientata pe obiect pe care am studiat-o de-a lungul anilor si sa o aplic in rezolvarea problemelor de generare a orarelor, in scopul aducerii unui plus in procesul de modernizare a procedurilor si organizarii din cadrul unei facultati.

In realizarea aplicatiei propriu-zise am folosit limbajul Java, un limbaj folosit pe scara larga pe toate dispozitivele actuale, desktop-uri, laptop-uri, tablete si smartphone-uri. Am ales acest limbaj datorita usurintei de utilizare a sa pe diferite masini si accesibilitatii de conectare la o gama larga de sisteme de baze de date. Pentru sistemul de gestiune a bazelor de date relationale am optat pentru Microsoft SQL Server deoarece este cel mai popular in ceea ce priveste interogarile bazelor de date si poate fi usor de instalat si de folosit.

2 Prezentarea domeniului din care face parte lucrarea

2.1 Fundamente teoretice

Domeniul prezentei lucrari este reprezentat de unul dintre cele mai importante sectoare ale societatii din toate timpurile si anume domeniul educational. Inca din antichitate este prezenta in natura umana setea de cunoastere si de descoperire a lumii inconjuratoare ce poate fi linistita numai prin invatare. Cand vorbim despre educatie, vorbim despre un domeniu in continua schimbare, de-a lungul timpului metodele de invatare, continutul sistemului de invatamant si al materialelor, dar si telurile invatamantului suferind numeroase modificari si prezentand o evolutie spectaculoasa.

Daca ar fi sa vorbim despre educatie referindu-ne strict la acea latura scolară a termenului, o definitie imediata ar putea fi activitatea prin care individul, in spate scolarul, studentul, se dezvolta din punct de vedere intelectual si profesional in vederea pregatirii pentru o viata sanatoasa si echilibrata de adult. Sistemul de educatie are in centrul sau omul, pe care prin toate mijloacele si formele de invatamant trebuie si il sustina si sa il pregateasca din punct de vedere profesional pentru a putea construi astfel o societate inteligenta si creativa, capabila sa se perfectioneze odata cu trecerea timpului. In acelasi timp, domeniul educational reprezinta piatra de temelie in viata oricarui om, doar prin aceasta individul uman inteleagand lumea si fenomenele ce il inconjoara, putand astfel sa isi largeasca orizonturile de cunoastere.

In prezent in Romania domeniul educational este structurat secential, continand doua niveluri, fiecare dintre acestea fiind compus din mai multe cicluri cu obligativitate diferita. Astfel putem distinge: nivelul pre-universitar si nivelul universitar.

In continuare ne vom referi strict la nivelul universitar pentru care este conceputa aplicatia de generare a orarelor, cu precizarea ca aceasta poate fi adaptata si pentru nivelul pre-universitar. Acest nivel la care facem referire contine urmatoarele componente: studii de licenta, studii de master, studii de doctorat si formare continua si nu reprezinta un nivel de studii obligatoriu.

Lucrarea prezinta, de asemenea, apartenenta si la un al doilea domeniu, acela fiind domeniul aplicatiilor software, ce inglobeaza concepte de programare si este format din sistem de operare, driver si aplicatii. Programele software fac parte din categoria mare a domeniului tehnologiei informatiei (IT), tehnologie ce prelucreaza informatiile prin diverse procese precum: achizitionare, procesare, stocare, transformare si trimitere.

Acest al doilea domeniu este unul de referinta in timpurile noastre, fiind aplicat unei multitudini de activitati umane precum: constructiile, aici referindu-ne la aplicatii software de realizare a arhitecturilor de cladiri si nu numai, medicina, in care intalnim aplicatii de programare a consultatiilor si evidenta pacientilor, industrie, aceasta fiind dominat de aplicatii ce controleaza procesele de fabricatie si roboti programati si nu in ultimul rand in educatie ca in cazul aplicatiei de fata.

In ceea ce priveste imbinarea dintre aplicatiile software si domeniul educatiei putem constata o sustinere reciproca a acestora, in sensul ca prin aplicatii software se sustine buna functionare si organizare a formelor de invatamant, a evidentei cadrelor didactice si a studentilor, dar si facilitarea accesului rapid la materiale de curs si laborator si a altor documente de interes. Pe de cealalta parte, educatia sustine dezvoltarea de aplicatii software prin invatarea studentilor si orientarea acestora catre cele mai bune solutii ale momentului in vederea digitalizarii cator mai multe concepte.

2.2 Stadiul actual al domeniului

In prezent, domeniul educational cunoaste o dezvoltare impresionanta datorata in special aplicatiilor software, fiind prezente pe piata internationala aplicatii software ce imbină metodele traditionale de invatare cu cele moderne, astazi fiind disponibile diverse platforme prin care cadrele didactice pot crea lectii interactive si sustine prezentari atractive, captand atentia si interesul studentilor.

O alta categorie de software utilizata in mediul universitar este reprezentata de aplicatiile prin care studentilor le sunt comunicate rezultatele evaluarilor si situatia lor scolară prin intermediul platformelor online, chiar Universitatea Politehnica "Bucuresti" folosind o astfel de platforma(studenti.pub.ro). Putem privi in continuare spre interesul acordat procesului de invatare prin punerea la dispozitie a platformelor pe care cadrele didactice pot incarca materiale de curs si laborator pentru studenti si pot face anunturi importante pentru acestia, iar studentii pot incarca teme de casa si pot adresa cadrelor didactice intrebari legate de subiectele care ii intereseaza.

Lista aplicatiilor folosite in domeniul educational poate fi continuata cu aplicatii orientate catre conducerea si administratia unei facultati, care pun la dispozitie facilitati precum: crearea de rapoarte, informatii cu privire la cursele autobuzelor scolare valabile pentru tarile care au aceasta metoda de transport a studentilor implementata, inregistrarea si editarea usoara a detaliilor cadrelor didactice si ale studentilor, asignarea atributiilor personalului administrativ, gruparea statica si dinamica a studentilor si gestiunea bibliotecii universitatii.

In figura de mai jos (vezi Figura 1.) sunt ilustrate datele extrase din [2] cu privire la structura persoanelor cu varste cuprinse intre 16 – 74 de ani ce utilizeaza calculatorul in Romania anului 2017. O imediata analiza a figurii arata faptul ca indiferent de varsta, in secolul al XXI-lea oamenii folosesc echipamentele IT, deci nevoia de aplicatii software indiferent de domeniul de utilizare ale acestora este in continua crestere, acestea usurand activitatile cotidiene ale populatiei si reducand riscul aparitiilor diferitelor erori, fie ele de natura umana sau de alta natura.

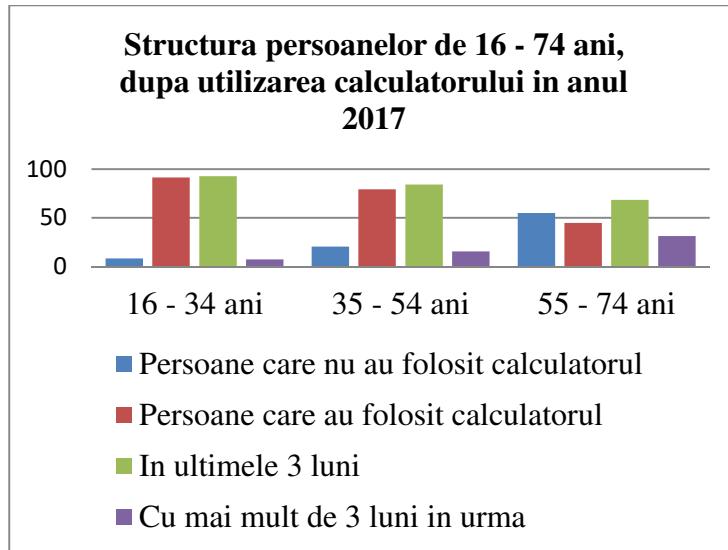


Figura 1. Structura persoanelor de 16 - 74 ani, dupa utilizarea calculatorului in anul 2017

2.3 Aplicatii similare

In realizarea prezentei lucrari documentarea in legatura cu ceea ce presupune o aplicatie de gestionare si modelare automata a orarului a fost esentiala in analiza si specificarea cerintelor si in determinarea constrangerilor ce fac parte din acest domeniu. Astfel prin studiu aprofundat al temei am regasit si cateva aplicatii similare cu cea dezvoltata in cadrul proiectului de fata, pe care le voi prezenta succint in paragrafele urmatoare.

2.3.1 Edutimer

Edutimer este un software de generare automata a orarului pentru o institutie de invatamant dezvoltata de Cybrosys Technologies dupa numeroase studii bazate pe observarea utilizatorului in diverse situatii.

Aplicatia are in vedere o gama larga de probleme ce pot aparea de-a lungul modelarii unui orar si acorda atentie sporita utilizarii resurselor intr-un mod cat mai eficient. Bazata pe sistemul in care elevii selecteaza preferintele de participare la diverse cursuri, aceasta unealta organizeaza in mod automat optiunile fara a mai fi necesara interventia operatorului uman.

Pe langa usurinta cu care aplicatia se lauda in introducerea datelor, dezvoltatorii spun ca produsul lor minimizeaza pauzele ce apar in timpul saptamanii in orarul cadrelor didactice prin definirea unui numar maxim de intervale orare libere si specificarea zilelor in care cadrul didactic poate preda. Ca functionalitate suplimentara, Edutimer are o interfata grafica prietenoasa, utilizatorul putand face modificari asupra orarului printr-un simplu click al mouse-ului.

2.3.2 School Management Software

School Management Software este o aplicatie vanduta de cei de la RenWeb si reprezinta o solutie integrata de automatizare a gestionarii prezentelor elevilor, a planificarii orelor de curs dar si de centralizare a notelor obtinute de elevi.

Pe langa partea de gestionare a orarului, aplicatia este dezvoltata pe inca doua ramuri importante, cum sunt administrarea unei institutii de invatamant si partea de contabilitate a unei institutii de invatamant. Astfel ea prezinta ca functionalitati in plus fata de aplicatia precedenta urmatoarele: posibilitatea de gestiune a facturilor catre clienti si a celor venite de la furnizori, generarea de rapoarte de admitere in cadrul facultatii, gestiunea salilor de studiu, a bibliotecii, dar si a cabinetelor medicale, gestiunea finantelor, dar si un portal dedicat parintilor.

3 Documentatie tehnica

Pentru dezvoltarea unei aplicatii ca cea prezentata in aceasta lucrare, echipamentele tehnice si tehnologiile utilizate joaca un rol determinant, astfel ca fara echipamentele fizice nu ar exista mediul fizic pe care sa poata fi scris codul, iar fara mediile de dezvoltare sursele nu ar putea fi interpretate, iar rezultatele nu ar putea fi atinse si vizualizate. In continuare se vor prezenta din punct de vedere teoretic, echipamentele si tehnologiile utilizate in vederea obtinerii aplicatiei de modelare si generare a orarului unei facultati.

3.1 Echipamente utilizate

De-a lungul procesului de dezvoltare a prezentului software s-a folosit ca echipament tehnic un laptop HP, avand ca sistem de operare instalat Windows 10 Home de tip 64-bit Operating System. Cateva dintre caracteristicile tehnice important de amintit ar fi procesorul Intel(R) Core(TM) i3-7100 CPU x64-based processor cu frecventa de 2.40 GHz, avand memorie RAM de 4 GB. Acest echipament a fost folosit, de asemenea, ca server local pentru mediile de dezvoltare precum MS SQL Server.

In prima faza a dezvoltarii s-a creat un model al aplicatiei prin intermediul uneltei Astah Community, unaalta simpla si gratuita de modelare UML (Unified Modeling Language). Astah Community este folosit preponderant in scop academic sau commercial, fiind util realizarii diagramelor de clase, a cazurilor de utilizare, a diagramelor de sechete si activitati, dar si a diagramelor de comunicare, de componente si deployment, toate acestea fiind esentiale in etapa de proiectare a unei aplicatii, ajutand la buna inteleger a cerintelor si functionarii procesului, dar si la determinarea posibilelor probleme.

In cea de-a doua etapa a realizarii aplicatiei s-a creat un model relational pe baza diagramei de clase obtinuta in faza precedenta. Pentru aceasta s-a ales SQL Server Management Studio (SSMS) ce reprezinta o aplicatie desktop pentru gestionarea unei infrastructuri SQL pronind de la serverul SQL si pana la baza de date SQL. Prin intermediul acestei unelte se poate configura, urmari si administra instante SQL, la fel cum se pot realiza scripturi si interrogari ale bazei de date in vederea manipularii datelor inregistrate.

Cea de a treia faza a dezvoltarii proiectului a fost constituita din scrierea efectiva a codului sursa cu ajutorul mediului Eclipse Luna. Acest mediu este dezvoltat in Java de cei de la Eclipse Foundation si este folosit pentru realizarea aplicatiilor in limbaj Java si cu ajutorul unor extensii si in alte limbaje precum PHP, Perl si Python. Ceea ce face ca Eclipse-ul sa fie un echipament de dezvoltare raspandit la scara larga este faptul ca toate sistemele de operare pot lucra cu aceasta unealta. Acest program nu a fost folosit numai pentru dezvoltare, ci a fost si baza procesului de testare a aplicatiei, scriindu-se astfel teste unitare, rezultatele fiind usor de interpretat prin intermediul sau.

3.2 Tehnologii software

In ordinea etapelor dezvoltarii se vor prezenta dupa cum urmeaza toate tehnologiile software care au dus la obtinerea rezultatului final al lucrarii.

3.2.1 Descrierea modelelor si specificatiilor UML

In cadrul dezvoltarilor aplicatiilor software se disting patru mari etape de lucru:

Etapa de analiza – reprezinta acea faza a proiectului in care sunt specificate si analizate cerintele problemei ce trebuie rezolvata, obtinandu-se ca rezultat un model orientat obiect. Acest model poate fi un model obiectual, ce releva componentele statice ale proiectului, definind clasele, obiectele si relatiile dintre acestea care corespund conceptelor cu care proiectul lucreaza. Un alt tip de model este reprezentat de modelul dinamic, ce contureaza partea dinamica a claselor sau obiectelor, vorbind aici despre stari in care un obiect de poate afla sau evenimentele care determina schimbarile de stari. Modelul functional reprezinta un al treilea posibil tip de model, ce descrie cum anume functioneaza aplicatia prin definirea modului in care obiectele se comporta, fara insa a aborda problema din punct de vedere al algoritmului care se va implementa. Iau nastere in aceasta etapa diagrame precum diagramele de clase si diagramele de obiecte.

Etapa de proiectare – reprezinta cea de-a doua faza a proiectului in care elementele conceptuale definite anterior sunt transpuse in elemente de implementare. In aceasta etapa este definita aplicatia din punct de vedere architectural, stabilindu-se resursele si platformele ce se vor folosi si modul in care aplicatia va fi dezvoltata. Se descriu, de asemenea, si clasele intr-o maniera mai detaliata si se stabileste si structura diagramelor de interactiuni aferente fiecarui scenariu.

Etapa de implementare – reprezinta etapa in care proiectul este “tradus” intr-un limbaj de programare.

Etapa de testare – reprezinta etapa in care sunt verificate atat modulele ce compun aplicatia din punct de vedere individual, cat si sistemul in intregimea sa.

Limbajul UML (Unified Modeling Language) este una dintre cele mai folositoare conventii utilizata in cadrul dezvoltarii aplicatiilor software, realizand o modelare vizuala a sistemului in vederea descrierii conceptelor intr-o maniera standardizata usor de inteles.

UML a luat nastere in anii 1990 in trei forme foarte asemanantoare realizate de James Rumbaugh, Grady Booch si Ivar Jacobson. In anul 1995 cei trei creatori se hotarasc sa dezvolte acest limbaj impreuna, unindu-si fortele in crearea unei unelte extreme de folositoare programatorilor pentru relatarea viziunilor.

Importanta folosirii limbajului UML in dezvoltarea proiectelor este sustinuta si de economia de timp pe care acesta o acorda prin crearea unei metode de notare comună atat pentru analisti, cat si pentru clienti, reducand neintelegerile intre acestia si dezacordurile ce pot provoca intarzieri in finalizarea aplicatiilor.

Distingem in cadrul UML din punct de vedere al componentelor diverse tipuri de diagrame folosite atunci cand se doreste obtinerea unui model al sistemului, ce insumeaza in fapt o multitudine de vizuni. Vorbim astfel despre:

Diagrama de clase - in care sunt reprezentate clasele si relatiile dintre acestea, clasele fiind de fapt un grup de obiecte caracterizate prin aceleasi componenti si aceleasi atribute.

Diagrama de obiecte – in care sunt reprezentate obiectele, instantieri ale claselor cu valori distincte ale atributelor.

Diagrama cazurilor de utilizare – aceste tipuri de diagrame descriu sistemul din perspectiva comportamentului acestuia atunci cand interacioneaza cu utilizatorul. Distingem aici doua componente distincte utilizatorul sau actorul ce este parte a mediului exterior sistemului si cazul de utilizare ce reprezinta parte interna a sistemului.

Diagrama de stare – este diagrama in care sunt marcate stările obiectelor la diferite momente de timp. Astfel de diagrame sunt cuprinse intre starea de inceput a sistemului si cea de final.

Diagrama de sevante – este alcătuita din obiecte si mesajele pe care acestea le trimit de la unul la altul, organizarea fiind data de aspectul temporal intr-o maniera progresiva.

Diagrama de colaborari – este asemanatoare diagramei de sevante, in sensul ca are aceleasi componente, obiectele si mesajele dintre acestea, insa in acest caz accentul nu cade pe dimensiunea temporală, ci pe cea contextuală, axandu-se pe spatialitate.

Diagrama de activitati – sunt diagrame introduse in vederea explicarii evenimentelor si transformarilor ce au loc de-a lungul functionarii sistemului.

Orientarea pe obiect in cadrul limbajului UML este definita printr-o serie de concepte ce respecta principiile fundamentale de POO (Programare Orientata Obiect). Astfel, se pune accent pe obiecte, ce reprezinta instantieri ale claselor si care le este descris comportamentul prin intermediul operatiilor, care impreuna cu atributele formeaza ceea ce se numesc caracteristicile obiectelor.

Atunci cand vorbim despre orientare pe obiect, fie in contextul unui limbaj de programare, fie in contextul UML, trebuie avute in vedere cateva notiuni de baza ce faciliteaza intelegerea mai buna a conceptului prin definirea unor proprietati specifice obiectelor si claselor. Au fost definita in acest sens:

Abstractizarea – presupune excluderea trasaturilor unui obiect in favoarea accentuarii trasaturilor mai relevante

Agregarea – reprezinta un tip de relatie intre clase, in cadrul careia se contureaza o legatura de tip compositie intre obiecte sau intre clase. Avem astfel in cadrul agregarii definite conceptele de composit si component, in care compositul reprezinta obiectul agregat, ce este format dintr-o multime de componente sau obiecte componente. Este insa important de precizat faptul ca duratele de viata ale compositului si componentelor sunt extreme de diferite, in sensul ca, distrugerea compositului duce la eliminarea componentelor, insa disparitia unei componente nu produce si disparitia compositului.

Asocierea – ca si agregarea, asocierea este o definitie a unui tip de legatura intre clase sau intre obiecte bazata pe colaborarea dintre acestea.

Incapsularea – acest principiu se bazeaza pe masarea transformarilor ce au loc in cadrul operatiilor definite pentru un obiect.

Mostenirea – exprima un tip de relatie ce poate avea loc intre clase, distingandu-se doua componente diferite si anume clasele si subclasele, acestea din urma dobândind operatiile si atributele claselor pe care le mostenesc si putand avea la randul lor operatii si atributie propriei.

Polimorfismul – reprezinta cazurile in care operatiile cu acelasi nume din cadrul diferitelor obiecte efectueaza transformari distincte.

Trimiterea de mesaje – defineste modul prin care obiectele comunica in cadrul unui sistem, astfel ca un obiect trimite un mesaj sau o cerere de efectuare a unei operatii catre un alt obiect, iar acesta raspunde primului prin analiza si executarea operatiei cerute.

In figura urmatoare (vezi Figura 2.) este relevant modul in care o clasa este reprezentata folosind programul Astah Community. Se observa ca in prima parte se specifica numele clasei, acesta trebuind sa fie scris cu litera mare. In cea de-a doua parte sunt precizate atributele clasei si tipul acestora, ele reprezentand proprietatile clasei si avand valori distincte pentru fiecare obiect ce instantiaza respectiva clasa. Ultima parte este rezervata operatiilor, ce sunt de fapt actiunile pe care utilizatorul le poate cere unei clase sa le intreprinda. Asa cum se poate observa, operatiile pot avea sau nu parametrii de intrare, ce semnifica factorii cu care se lucreaza in cadrul actiunilor si in mod obligatoriu trebuie precizat tipul rezultatului intors de respectiva operatie.

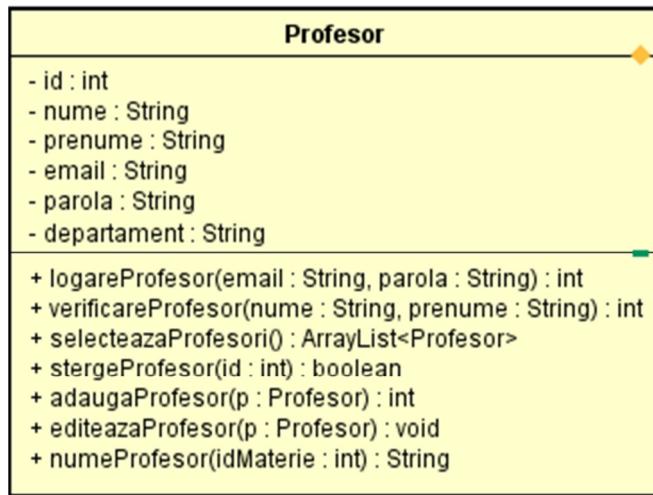


Figura 2. Reprezentare clasa Profesor

In ceea ce priveste tipurile de relatii ce se pot defini intre clase distingem trei concepte diferite precum:

Asocierea – asa cum a fost precizat si anterior aceasta relatie se bazeaza pe o imperechere intre clase, fiecare avand un rol bine stabilit din punct de vedere al conexiunii. O clasa poate fi asociata cu mai multe clase in acelasi timp, asocierea putand fi de multe ori restrictionata de constrangeri. Apar, de asemenea, situatii in care asocierea poate sa aiba la randul sau atribute si operatii, luand nastere clasa de asociere care sustine conexiunea dintre clasele asociate.

Compozitia - reprezinta relatia de agregare de tip parte – intreg intre clase, in care ciclul de viata al partii este gestionat de ciclul de viata al intregului, fara insa a fi considerata si reciproca.

Generalizarea - reprezinta relatia prin care se trece de la clasa derivata, la clasa de baza, cu alte cuvinte se face tranzitia de la cazul particular la cazul general.

In conceperea relatiilor dintre clase, o alta notiune importanta de amintit este reprezentata de gradul de multiplicitate al relatiei sau asa cum mai este numita cardinalitatea relatiei. In fapt, aceasta notiune ofera informatii cu privire la numarul de obiecte ce apartin unei clase si pot fi legate de obiectele altor clase cu care s-a stabilit o relatie de asociere.

In reprezentarea urmatoarea (vezi Figura 3.) se arata modul de redare al unei relatii de tip asociere in cadrul diagramei de clase realizata pentru dezvoltarea prezentei aplicatii. Astfel, se observa faptul ca un cadru didactic poate preda niciuna, una sau mai multe materii, acest lucru fiind concretizat in cadrul diagramei prin specificarea multiplicitatii asociierii, dar si prin exprimarea rolului clasei Profesor in raport cu clasa Materie.

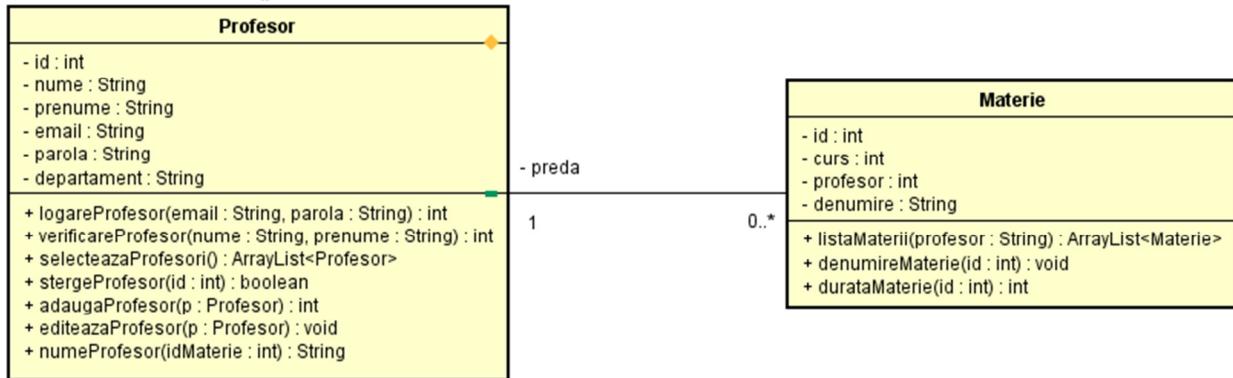


Figura 3. Reprezentare asociere Profesor - Materie

3.2.2 Baze de date MS SQL Server

Asa cum am amintit si in introducere, MS SQL Server reprezinta unul dintre cele mai puternice si populare sisteme de gestiune a bazelor de date relationale. A fost dezvoltat de compania Microsoft Corp originara din Statele Unite ale Americii. Ca si extensie procedurala sistemul foloseste T-SQL, iar ca limbaj de interogare - SQL.

Popularitatea sistemului este data de faptul ca odata cu aparitia noilor versiuni, SQL Server poate fi folosit pentru baze de date de dimensiune variabila, de la cele mai mici pana la cele de dimensiuni foarte mari. Din punct de vedere istoric, acest prim program de management al bazelor de date relationale al Microsoft a concurat cu celelalte programe existente pe piata vremii ale companiilor mari precum Oracle, IBM si Sysbase. Ca si exclusivitate pe piata bazelor de date versiunea SQL Server 7.0 este primul server ce prezinta GUI, versiunile care il succed pana in prezent avand performante imbunatatite, unele IDE si ETL si suporta date XML. De asemenea, MS SQL Server suporta Open Database Connectivity (ODBC), ceea ce le permite limbajelor de programare sa interactioneze cu baza de date.

Datele sunt stocate in acest sistem respectand modelul relational, alcatuind tabele formate din linii si coloane. Coloanele pot contine tipuri SQL Server de tip primar cum ar fi: intreg, zecimal, caractere, siruri de caractere, dar si tipuri complexe precum: XML, date spatiale si binare. Administrarea bazelor de date MS SQL Server se poate face prin folosirea interfetei grafice Microsoft SQL Server Management Studio, iar rularea sa poate fi realizata pe multe dintre platformele software existente: Windows, Linux si Docker containers.

Marele avantaj prezentat de bazele relationale este folosirea limbajului SQL (Structured Query Language), un limbaj simplu, dar foarte puternic in ceea ce priveste accesul datelor stocate in mai multe tabele, filtrarea lor, dar si afisarea, rezumarea si sortarea rezultatelor.

In ceea ce priveste stocarea datelor, aceasta se face pentru siguranta si usurarea administrarii bazei de date sub forma unui singur fisier, facand posibila recuperarea fisierului prin realizarea copiei de siguranta.

Vorbim in final si despre efectul sistemelor de gestiune a bazelor de date asupra procesorului, acestea necesitand mai multe cicluri de procesare pentru cererile de date decat orice alt tip de fisier normal, insa se asigura siguranta datelor, iar in privinta accesului la distanta singurele date ce se transmit prin intermediul retelei fiind rezultatele cerute, programul SQL realizand toate operatiile.

3.2.2.1 Structura unei baze de date relationale

Cand se face referire la termenul de baze de date relationale se vorbeste de fapt despre baze de date in care vizualizarea acestora de catre utilizator se face sub forma de tabele. Tabelele reprezinta in sine relatiile in cadrul unei baze de date, avand ca nume ale coloanelor atributelor datelor ce se afla pe fiecare coloana.

Mai jos este ilustrat un tabel parte a bazei de date a aplicatiei (vezi Figura 4.), in care numele coloanelor reprezinta atributele unei sali de cursuri si anume: id-ul salii, numarul de locuri din sala, etajul la care se afla sala si numele acesteia, toate atributele realizand o scurta descriere a salilor.

Id	Nr_locuri	Etaj	Nume
1	100	0	EC001
2	150	0	PR002
3	200	1	AN030

Figura 4. Tabelul Sala extras din baza de date a aplicatiei

Pe primul rand sunt asezate descrierile pentru fiecare coloana, urmand ca pe celelalte randuri sa existe descrierea unei singure sali. De exemplu, pe al doilea rand este descrisa sala EC001 ce are 100 de locuri si se afla la parterul cladirii.

Prima coloana, coloana denumita Id, joaca in acest tabel rolul de „cheie primara”, rol important in referirea unui anumit rand din tabel, fiind unica pentru fiecare inregistrare. Exista, insa, cazuri in care tabelele nu contin nicio coloana care sa contine o valoare unica pentru fiecare inregistrare, caz in care se combina valorile mai multor coloane luand nastere o „cheie primara compusa”.

Pentru ca vorbim de baze de date relationale, vorbim si de capacitatea de a defini relatii intre datele ce fac parte din mai multe tabele.

In cele doua tabele ce urmeaza (vezi Figura 5/Figura 6.) sunt descrise materiile din cadrul facultatii, durata cursului fiecarei materii, dar si cadrul didactic care predă fiecare materie.

In primul tabel se poate observa coloana ID, cu rol de „cheie primara” si atributele unei intrari de tip materie: denumirea, durata cursului si cadrul didactic ce predă materia. In acelasi timp, in cel de-al doilea tabel este evidențiată coloana Id, ce reprezinta „cheia primara”, urmata de atributele unui cadrul didactic: nume, prenume, email, parola contului din aplicatie si departamentul din care acesta face parte.

Se observă ca atributul Id (din tabelul Profesor) are rol dublu, de „cheie primara” a tabelului original, dar pentru ca reprezinta si coloana distinctă numita Profesor in tabelul Materie se mai numeste si „cheie străină” a acestuia din urma. Astfel ca, tabelul Materie stocheaza si atributul Profesor, ce reprezinta id-ul cadrului didactic ce predă o anumita materie. De exemplu, materia cu denumirea „Baze de Date (BD)” este predată de cadrul didactic „Popescu”.

Id	Denumire	Curs	Profesor
9	Baze de Date (BD)	2	1
10	Mecatronica (MCT)	2	1
11	Ingineria Reglarii Automate (IRA)	3	2
12	Aplicatii Multimedia (AM)	2	1
13	Informatii Structurate (XIS)	2	1
14	Managementul Proiectelor (MP)	2	2
15	Gestiunea si Managementul Documentelor (GMD)	2	2
18	Retele de Calculatoare (RC)	2	1
19	Sisteme de Conducerea Fabricatiei (SCF)	2	2
20	Securitatea sistemelor de calcul (SSC)	2	2
21	Matematica 1	2	1

Figura 5. Tabel Materie extras din baza de date a aplicatiei

Id	Nume	Prenume	Email	Parola	Departament
1	Popescu	Ion	popescuion@upb.ro	popescu	ACS
2	Ionescu	Maria	ionescumaria@upb.ro	ionescu	ACS
3	Georgescu	Andrei	georgescuandrei@upb.ro	georgescu	ACS
4	Constantinescu	Alexandra	constantinescualexandra@upb.ro	constantinescu	ACS

Figura 6. Tabel Profesor extras din baza de date a aplicatiei

3.2.2.2 Proiectarea unei baze de date

Un aspect important ale bazelor de date relationale este reprezentat de crearea bazei de date intr-o forma clara, usor de inteles si de interactionat cu ea. Astfel, sunt definite cateva reguli prin care se respecta principiul de tabel ideal:

- Este exclusa folosire campurilor calculate
- Este exclusa folosirea campurilor duplicate
- Datele continute in tabel trebuie sa nu fie redundante
- Fiecare camp este necesar sa contine o singura valoare
- Tabelul trebuie sa prezinte in mod obligatoriu o „cheie primara”

In continuare definim cateva aspecte ce insumate dau nastere conceptului de camp perfect:

- Campul nu contine valori obtinute prin calcule sau concatenarea altor campuri
- Campul trebuie sa contine o singura valoare
- Campul nu poate fi o colectie de componente mai mici
- Campul defineste o caracteristica diferita a unei inregistrari din tabel
- La aparitia in alte tabele campul nu isi modifica caracteristicile

O alta problematica ce trebuie avuta in vedere in cazul proiectarii unei baze de date este reprezentata de alegera tipului de date in mod corespunzator pentru fiecare coloana. Majoritatea bazelor de date accepta urmatoarele tipuri generale de baze de date:

- Caracter: CHAR(n) – reprezinta un singur sir de caractere cu lungimea maxima n caractere
- Caracter: VARCHAR(n) – reprezinta un sir de caractere cu lungime variabila si lungime maxima n caractere
- Intreg: INT – reprezinta un intreg cuprins intre -2147483648 si 2147483647
- Intreg: SMALLINT – reprezinta un intreg cuprins intre -32768 si 32767
- Float: FLOAT(n) – reprezinta un numar cu virgula mobila, avand n numarul de biti folositi pentru mantisa
- Date: DATE – reprezinta o data calendaristica in format yyyy-mm-dd
- Time: TIME – reprezinta ora in format hh:mm:ss

3.2.2.3 Crearea unei baze de date si a unui tabel

Pentru crearea unei baze de date sau a unui tabel se foloseste limbajul T-SQL (Transact - SQL), acest limbaj este unu special, fiind o subcomponenta a limbajului SQL. Comenzile T-SQL sau cele SQL se pot scrie atat in interpretoare SQL, cat si in Java, in general acestea nefiind receptive la diferente intre literele majuscule si minuscule.

In cadrul Microsoft SQL Server Management Studio atat bazele de date, cat si tabelele pot fi create fie cu ajutorul interfetei grafice disponibile, fie prin scrierea de comenzi.

Mai jos este prezentat un exemplu de comanda pentru crearea bazei de date a prezentei aplicatii:

```
CREATE DATABASE [SmartCampus]
CONTAINMENT = NONE
ON PRIMARY
( NAME = N'SmartCampus', FILENAME = N'C:\Program Files\Microsoft SQL
Server\MSSQL14.MSSQLSERVER\MSSQL\DATA\SmartCampus.mdf' , SIZE = 8192KB ,
MAXSIZE = UNLIMITED, FILEGROWTH = 65536KB )
LOG ON
( NAME = N'SmartCampus_log', FILENAME = N'C:\Program Files\Microsoft SQL
Server\MSSQL14.MSSQLSERVER\MSSQL\DATA\SmartCampus_log.ldf' , SIZE = 8192KB ,
MAXSIZE = 2048GB , FILEGROWTH = 65536KB )
GO
```

In cazul de mai sus „Smartcampus” reprezinta numele bazei de date, atributul CONTAINMENT este setat cu NONE, specificand daca baza de date este izolata sau nu, atributul ON specifica fisierele de pe disc folosite pentru a inmagazina sectiunile de date, prin PRIMARY se specifica principalul pachet de fisiere, iar LOG ON specifica fisierele folosite pentru logurile bazei de date.

In cele ce urmeaza este prezentata crearea tabelului Profesor din cadrul bazei de date a aplicatiei SmartCampus:

```
CREATE TABLE [dbo].[Profesor] (
    [id] [int] IDENTITY(1,1) NOT NULL,
    [nume] [varchar](50) NOT NULL,
    [prenume] [varchar](50) NOT NULL,
    [email] [varchar](200) NOT NULL,
    [parola] [varchar](200) NOT NULL,
    [departament] [varchar](50) NOT NULL,
    CONSTRAINT [PK_Profesor] PRIMARY KEY CLUSTERED
    (
        [id] ASC
    )
GO
```

In acest caz, Profesor reprezinta numele tabelului ce se doreste a fi creat, specificandu-se mai departe atributele id, de tip intreg, specificandu-se prin proprietatea IDENTITY (1,1) generarea automata a acestui camp incepand de la valoarea 1, incrementarea facandu-se cu valoarea 1, iar NOT NULL arata faptul ca nu sunt acceptate valori de tip null pentru atributul id. Mai departe se observa atributele nume, prenume, email, parola si departament de tipul sir de caractere de dimensiunea maxima 50, respective 200 si faptul ca nici aceste attribute nu accepta valori de tip null. In finalul secentei se evidențiaza prin CONSTRAINT definirea atributul id ca fiind cheie primara a tabelului Profesor. Este bine pentru evitarea erorilor sa se defineasca proprietatea IDENTITY in special la "cheile primare" ale tabelelor asa cum arata exemplul de mai sus.

3.2.2.4 Stergerea si modificaea unui tabel

Stergerea unei coloane dintr-un tabel sau a unui intreg tabel reprezinta o operatie ireversibila, astfel ca datele eliminate sunt pierdute complet.

In cazul in care se doreste stergerea unui tabel aceasta se poate realiza prin urmatoarea comanda:

```
DROP TABLE [dbo].[An]
GO
```

In acest caz An reprezinta numele tabelului ce se doreste a fi sters.

Daca vorbim despre modificarea unui tabel, limbajul SQL pune la dispozitie o comanda de tip ALTER TABLE, nefiind necessara stergerea intregului tabel si recrearea sa pentru operarea modificarilor dorite. Acest tip de comanda poate fi executata operatiuni de stergere sau adaugare de coloane, de modificare a tipului de date continut in anumite coloane sau de adaugare, stergere sau modificare de proprietati pentru anumite attribute.

Astfel, pe parcursul dezvoltarii aplicatiei dupa crearea tabelului An a trebuit sa ii adaug acestuia o „cheie strina” catre tabelul Facultate, iar aceasta operatie am realizat-o prin secenta urmatoare, in care atributul facultate din cadrul tabelului An reprezinta „cheia strina” ce refera atributul id din tabelul Facultate:

```
ALTER TABLE [dbo].[An] WITH CHECK ADD CONSTRAINT [FK_An_Facultate] FOREIGN
KEY([facultate])
REFERENCES [dbo].[Facultate] ([id])
GO
```

3.2.2.5 Inserarea, modificarea si stergerea datelor intr-o baza de date

Cele trei comenzi distincte pentru operatiunile de inserare, modificare si stergere de date sunt „INSERT”, „UPDATE” si „DELETE”, fiind folosite sub diferite forme.

Comanda „INSERT” este folosita pentru inserarea datelor intr-un tabel prin adaugarea unui rand nou si are urmatoarea sintaxa:

```
INSERT INTO [dbo].[Profesor]
    ([nume]
     ,[prenume]
     ,[email]
     ,[parola]
     ,[departament])
VALUES
    ('Popescu'
     ,'Ion'
     ,'popescuion@upb.ro'
     ,'popescu'
     ,'ACS')
GO
```

Astfel au fost specificate numele tabelului in care se doreste a fi inserata noua inregistrare si anume tabelul Profesor, attributele pentru care sunt introduse valorile in ordinea in care acestea apar in tabel: nume, prenume, email, parola, departament si valorile ce se doresc a fi introduse pentru fiecare atribut in parte. A se observa faptul ca desi exista si coloana Id, care reprezinta “cheia primara” a tabelului nu se seteaza nicio valoare pentru aceasta deoarece ea este generata automat, acest lucru fiind specificat odata cu crearea tabelului prin folosirea proprietatii IDENTITY pentru atributul id.

Alte forme de existenta ale acestei comenzi ar putea fi cea in care nu apar specificate attributele tabelului sau cea in care nu apar valori pentru toate attributele, introducandu-se in cazul in care nu exista restrictii valoarea null. Daca in cea de-a doua situatie sunt specificate restrictii si seincearca introducerea valoarii null, comanda “INSERT” esueaza.

Există cazuri in care utilizatorul isi doreste sa modifice una dintre inregistrarile din tabel, iar acest lucru este posibil cu ajutorul comenzi “UPDATE”, ce are urmatoarea structura:

```
UPDATE [dbo].[Profesor]
    SET [nume] = 'Ionescu'
        ,[prenume] = 'Maria'
        ,[email] = 'ionescumaria@upb.ro'
        ,[parola] = 'ionescu'
        ,[departament] = 'ACS'
    WHERE <id = 2>
GO
```

Asadar, in tabela Profesor se modifica la inregistrarea ce are id-ul egal cu 2, attributele: nume, prenume, email, parola si departament setandu-se valorile de mai sus. Daca nu s-ar fi specificat conditia WHERE ar fi fost modificate toate coloanele tabelului, astfel dovedindu-se faptul ca in functie de aceasta conditie pot fi actualizate una sau mai multe inregistrari.

Pentru stergerea inregistrarilor dintr-un tabel se foloseste comanda “DELETE”, prin aceasta putand fi eliminate o singura inregistrare, un numar limitat de inregistrari sau toate inregistrarile continue intr-un tabel. In cele ce urmeaza este prezentata varianta in care se doreste stergerea unei singure inregistrari si anume iesirea cu id-ul egal cu 3 din cadrul tabelului Profesor:

```
DELETE FROM [dbo].[Profesor]
WHERE <id = 3>
GO
```

3.2.2.6 Interogarea unei baze de date

Data Manipulating Language (DML) reprezinta o subcomponenta a limbajului SQL folosita pentru a putea lucra cu datele stocate intr-o baza de date relationala. Acest limbaj ofera posibilitatea de interogare a bazei de date si de afisare a rezultatelor pe care userul doreste sa le analizeze.

Cea mai utilizata comanda a acestui limbaj este reprezentata de comanda “SELECT” ce poate fi prezenta in forme de la cele mai simple pana la cele mai complicate in functie de datele care se doresc a fi obtinute.

Forma simpla a comenzii “SELECT” este:

```
SELECT * FROM Profesor
```

Aceasta comanda va avea ca rezultat totalitatea inregistrarilor continue in tabelul Profesor. Afisarea se va face sub forma de tabel, ce contine toate coloanele din tabelul la care se doreste sa se obtina accesul si toate inregistrarile care se regasesc in acesta in ordinea in care ele apar stocate.

In exemplul urmator se defineste modul de selectare doar a unor coloane dintr-un tabel, respectiv se doreste afisarea tuturor denumirilor materiilor din tabelul Materie:

```
SELECT denumire FROM Materie
```

O observatie importanta in situatia de mai sus ar fi aceea ca se pot selecta oricate coloane se doreste, insa denumirea acestora trebuie separata prin virgula.

Prin folosirea comenzii “WHERE” se pot selecta dintr-un tabel doar acele inregistrari care respecta o anumita conditie, astfel toate celelalte inregistrari sunt excluse de la afisare. Mai jos este ilustrat un exemplu de afisare a denumirii materiei ce are id-ul 13 si se regaseste in tabelul Materie, comanda avand ca rezultat un tabel format dintr-un singur rand si o singura coloana, coloana denumire si valoarea acesteia a inregistrarii cu id-ul 13:

```
SELECT denumire FROM Materie WHERE id = 13
```

Pentru a impune conditiile in limbajul DML se folosesc operatorii relationali urmatori:

- = - Verifica daca valorile a doi operanzi sunt egale, iar daca acestea sunt, conditia ia valoarea “true”
- != - Verifica daca valorile a doi operanzi sunt diferite, iar daca acestea sunt, conditia ia valoarea “true”
- <> - Verifica daca valorile a doi operanzi sunt diferite, iar daca acestea sunt, conditia ia valoarea “true”
- > - Verifica daca valoarea primului operand este mai mare decat a celui de-al doilea, iar daca aceasta este, conditia ia valoarea “true”
- < - Verifica daca valoarea primului operand este mai mica decat a celui de-al doilea, iar daca aceasta este, conditia ia valoarea “true”
- >= - Verifica daca valoarea primului operand este mai mare sau egala decat a celui de-al doilea, iar daca aceasta este, conditia ia valoarea “true”
- <= - Verifica daca valoarea primului operand este mai mica sau egala decat a celui de-al doilea, iar daca aceasta este, conditia ia valoarea “true”

In plus fata de operatorii amintiti mai sus exista si operatori non – algebrici ce pot fi folositi in SQL:

- BETWEEN – Verifica daca o valoare a unui atribut este inclusa intre valorile altor doua atribute (exemplu: i BETWEEN j AND k)
- LIKE – Verifica daca valoarea unui atribut este echivalenta cu o valoare specificata de utilizator
- NOT LIKE – verifica daca valoarea unui atribut nu este echivalenta cu o valoare specificata de utilizator
- IN – Verifica daca valoarea unui atribut se regaseste intr-o lista specificata de utilizator ce poate contine unul, doi sau mai multi membri. (exemplu: i IN (k, j))
- NOT IN - Verifica daca valoarea unui atribut nu se regaseste intr-o lista specificata de utilizator ce poate contine unul, doi sau mai multi membri. (exemplu: i NOT IN (k, j))
- IS NULL – Verifica daca un atribut are valoarea null.
- IS NOT NULL – verifica daca un atribut nu are valoarea null.
-

In cazul operatorului LIKE ne confruntam cu doua posibile scenario, cel in care referim zero, unul sau mai multe caractere, acest lucru realizandu-l cu semnul “%” si cel in care referim un singur caracter, facand posibil acest lucru cu ajutorul “_”. Cele doua semne pot aparea si in diverse combinatii intre ele sau impreuna cu caractere, ceea ce este important de precizat fiind faptul ca sirurile de caractere se marcheaza prin apostrof la inceput si sfarsit.

De asemenea, in cadrul interogarilor sunt posibile si aparitii ale expresiilor logice care produc rezultate in combinatie cu operatorii logici pecum:

- AND – Permite folosirea cu mai multe conditii. Intoarce valoarea “true” in cazul in care toate conditiile sunt indeplinite.
- OR - Permite folosirea cu mai multe conditii. Intoarce valoarea “true” in cazul in care macar o conditie este indeplinita.
- NOT – Permite folosirea cu o singura conditie. Intoarce valoarea “true” in cazul in care conditia este falsa.

In cadrul interogarilor pot aparea si necesitatea de grupare si sortare a rezultatelor obtinute, acestea realizandu-se cu anumite clauza precum “ORDER BY”, “GROUP BY” si “HAVING”.

Cu ajutorul “ORDER BY” se pot ordona rezultatele in functie de preferinte ascendent sau descendant, acest fapt specificandu-se prin introducerea ASC pentru ascendent sau DESC pentru descendant dupa numele atributului. In mod implicit daca apare aceasta clauza datele sunt afisate in ordine ascendentă. Se pot ordona, de asemenea, rezultatele in functie de mai multe atribute, insa acestea trebuie separate prin virgula.

Clauza “GROUP BY” este strans legata de cea prezentata mai sus in sensul ca valoarea conform careia se face gruparea trebuie sa fie continua si intr-o clauza de tip ORDER BY care succede clauza de fata. Acest cadru se completeaza cu clauza “HAVING” prin care se include in rezultat doar grupurile selectate.

Astfel se obtine o forma complexa a interogarilor, avand urmatoarea structura:

```
SELECT col1, col2, col3...coln
FROM Tabel
WHERE ConditieSelectare
GROUP BY CampulDeGrupare
ORDER BY CampulDeOrdonare
HAVING ConditieFiltrare
```

In acest caz col 1...col n reprezinta atributele ce se doresc a fi afisate, Tabel reprezinta tabelul care se doreste a fi accesat, ConditieSelectare reprezinta constrangerile definite de user pentru afisarea rezultatului, CampulDeGrupare reprezinta campul dupa ale carui valori se vor grupa rezultatele, CampulDeOrdonare reprezinta campul dupa care se doreste a fi facuta ordonarea, ConditieFiltrare reprezinta conditia dupa care sunt soritate datele.

3.2.2.7 Interogarea mai multor tabele

Cerinta de minimizare a informatiilor redundante la nivelul unei baze de date relationale se refera la construirea tabelelor ce alcataiesc baza intr-o maniera organizata, acestea continand informatii clare si intr-un mod cat mai simplificat. Aceasta atrage dupa sine problema gradului scazut de infomaticie ce se obtine prin interogari ce vizeaza un singur tabel. Astfel, s-a ales ca solutie stabilirea legaturilor intre tabele si realizarea prin intermediul acestora a interogarilor asupra tabelelor multiple. Se creaza deci acolo unde se doresc “chei straine” care refera tabelul sau tabelele cu care sa stableasca o relatie, acestea trebuie sa fie construite astfel incat sa nu incalce principiul integritatii referirii.

In algebra relationala vorbim datorita acestor “chei straine” de diferite tipuri de jonctiuni precum: echijonctiunea, autojonctiunea si jonctiunea externa. Jonctiunea interna apare numai in clauza FROM, avand urmatoarea forma:

```
SELECT * FROM Materie INNER JOIN Profesor ON Materie.profesor =
Profesor.id where Profesor.email = 'popescuion@upb.ro'
```

Se poate observa deci ca in acest exemplu tabelul Materie reprezinta tabelul din care se doresc a fi selectate toate atributele intrarilor, acesta fiind legat de tabelul Profesor la care se refera prin intermediul „cheii straine” Materie.profesor ce este identica cu „cheia primara” Profesor.id din tabelul Profesor. Interrogarea prezentata are scopul de a selecta toate atributele materiei predate de cadrul didactic ce detine emailul „popescuion@upb.ro”.

Aceasta sintaxa este considerata ca fiind una usor de manipulat si interpretat, insa odata cu realizarea jonctiunilor intr-un numar ridicat de tabele gradul de complexitate creste. Asa cum se poate vedea mai jos, in cazul jonctiunilor mai complicate se tine cont de faptul ca operatorul de jonctiune prezinta proprietatea de asociativitate.

```
SELECT DISTINCT Interval_Orar.* FROM Plan_Invatamant p, Materie m, An a INNER JOIN Facultate f ON a.facultate = f.id, interval_orar INNER JOIN zi ON zi.id = interval_orar.ziua INNER JOIN serie ON serie.id = interval_orar.seria WHERE serie.denumire = 'AA' AND p.materie = Interval_Orar.materia AND p.semestru = '2' AND interval_orar.seria = p.serie AND p.an = '1' AND f.denumire = 'Facultatea de Automatica si Calculatoare' AND interval_orar.preferinta = '1' ORDER BY Interval_Orar.ziua, interval_orar.ora_start
```

Aceasta interrogare are ca rezultat orarul pentru anul 1, semestrul al doilea, al seriei AA din cadrul Facultatii de Automatica si Calculatoare. Se observa faptul ca tabelul An contine o “cheie strina” catre tabelul facultate, pentru ca in acest tabel este nevoie sa stim in orice moment in cadrul carei facultati exista o anumita inregistrare. De asemenea, tabelul Interval_Orar ce contine date despre intervalele orare ocupate trebuie sa aiba legatura cu tabelul Zi, utilizat in aceasta aplicatie ca nomenclator in care sunt trecute toate zilele saptamanii lucratore, dar si cu tabelul Serie si el definit ca nomenclator al seriilor din cadrul unei facultati. Aceste legaturi au fost realizate deoarece atunci cand se afiseaza rezultatele unei interrogari ale tabelului Interval_orar trebuie sa stim pentru ce grup de studenti este acesta valabil si in ce zi din saptamana a fost ocupat.

3.2.2.8 Vederi

Pentru implementarea aplicatiei de modelare si gestionare a orarului in cadrul unei facultati a fost necesara in plus fata de folosirea tablelor crearea si utilizarea de vederi. Acestea au fost construite in concordanta cu datele introduse in baza de date, avand forma identica cu cea a tablelor, insa fiind create ca definitii SQL. In aceste forme datele sunt actualizate pe baza a ceea ce este inregistrat in tablele si nu pot contine definitii de strangeri.

Poate fi remarcata in secenta ce urmeaza sintaxa specifica realizarii vederii Alocare din cadrul bazei de date a aplicatiei ce face subiectul lucrarii:

```
CREATE VIEW [guest].[Alocari]
AS
SELECT dbo.Materie.profesor AS id_profesor, dbo.Materie.id AS id_materie,
dbo.Profesor.nume, dbo.Profesor.prenume, dbo.Profesor.email,
dbo.Profesor.departament, dbo.Materie.denumire AS materie, dbo.Materie.curs
AS ore_curs FROM dbo.Materie INNER JOIN dbo.Profesor ON dbo.Materie.profesor
= dbo.Profesor.id
GO
```

A fost creata vederea cu numele Alocari in care au fost introduse ca atribute id-ul profesorului, id-ul materiei, numele, prenumele, email-ul si departamentul cadrului didactic, dar si denumirea si durata unui curs pentru fiecare materie. Se remarcă prezenta operatorului AS prin care se specifică modul în care se dorește a fi denumit un atribut în cadrul vederii ca de exemplu atributul curs din cadrul tabelului Materie se dorește a fi denumit ore_curs.

3.2.2.9 Conexiunea cu baza de date

Pentru a putea interacționa cu baza de date MS SQL Server trebuie să se stabilească conexiunea cu serverul de baze de date. Aceasta conexiune face posibila realizarea de operații și interogări în baza de date însă doar după ce parametrii de conectare au fost verificati și validati ca fiind corecti. Datele de conectare ce trebuie introduse de utilizator sunt compuse din tipul serverului și numele acestuia, tipul de autentificare, numele și parola de utilizator.

In imaginea ce urmează (vezi Figura 7.) este realizată conexiunea către serverul de baze de date având denumirea “LAPTOP-H4AJ3TNV”, cu utilizatorul root într-un mod de autentificare specific SQL Server.

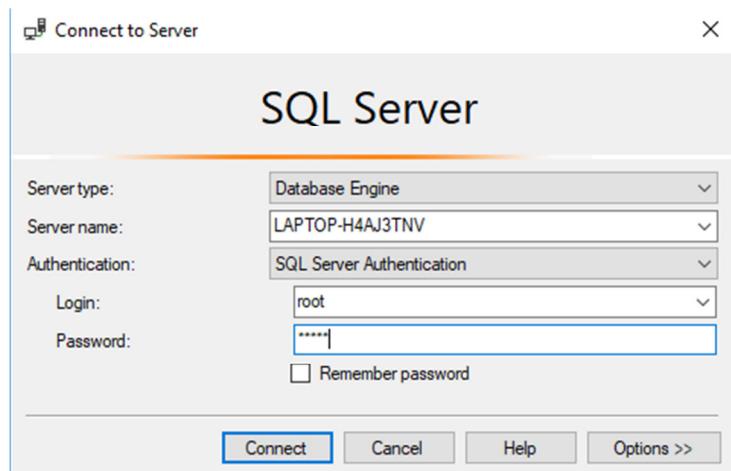


Figura 7. Conexiunea cu baza de date

Aplicația de fata a fost dezvoltată cu ajutorul limbajului Java, conexiunea cu baza de date facându-se în acest caz prin intermediul JDBC în afara serverului aplicatiei, clasa DriverManager gestionând stabilirea de conexiuni.

3.2.3 Programare orientata obiect Java

In procesul de dezvoltare a aplicatiilor orientate obiect se disting trei mari etape de lucru si anume: analiza orientata obiect, proiectarea orientata obiect si programarea orientata obiect. Analiza reprezinta etapa in care se studiaza domeniul din care aplicatia face parte si sunt definite conceptele ce apartin acestui domeniu, rezultatul obtinut la finalul acestei faze fiind structura sistemului, cerintele funktionale ale acestuia, dar si un model simplist a ceea ce se doreste a fi dezvoltat. Cel de-al doilea pas in realizarea proiectelor software este reprezentat de proiectarea solutiei software ce se bazeaza pe analiza facuta anterior si are rolul de a specifica operatiile si atributele componentelor ce alcatuiesc sistemul, dar are si rol in definirea cerintelor non – funktionale ale acestuia si realizarea unui model mai complex, care sa descrie in detaliu ceea ce urmeaza a fi programat. Etapa de programare sau implementare asa cum mai este ea denumita reprezinta momentul in care tot ceea ce a fost analizat si proiectat anterior este transformat si exprimat cu ajutorul unui limbaj de programare.

La ora actuala sunt prezente o multitudine de limbaje de programare orientate obiect, cele mai cunoscute fiind cele bazate pe clase, in care obiectele reprezinta instante ale acestora si sunt alcatuite din atribute reprezentate cu ajutorul datelor si metode evidente prin cod. Pentru aplicatia dezvoltata in cadrul acestei lucrari a fost utilizat limbajul Java datorita portabilitatii sale ridicate.

3.2.3.1 Notiuni de introductive

Limbajul de programare orientata obiect Java a luat nastere la inceputul anului 1990, cel care l-a conceput fiind cunoscutul James Gosling, acesta facand parte la acea vreme din cadrul companiei Sun Microsystems, transformata mai tarziu in Oracle. Cinci ani mai tarziu, Java este lansat pe piata, iar programatorii incep sa foloseasca acest limbaj in special pentru aplicatiile distribuite. Odata cu evolutia s-a spectaculoasa, acest limbaj cunoaste o raspandire pe scara larga datorita posibilitatii de a putea fi utilizat pe diferite dispozitive precum desktop-uri, telefoane mobile, tablete si alte echipamente tehnologice „inteligente”.

Originile acestui limbaj se gasesc in sintaxe precum C si C++ insa se deosebeste de acestea prin faptul ca prezinta performante ridicate, este un limbaj sigur, care odata scris corect si complet poate fi rulat oriunde. In prezent a inceput sa fie utilizat si pentru aplicatii destinate retelelor de calculatoare.

Portabilitatea limbajului este data de compilarea codului Java cu ajutorul masinii virtuale, ce nu depinde de sistemul de operare al platformei pe care ruleaza aplicatia, Windows, Ubuntu sau MacOS. Se interpreteaza deci ceea ce este scris de catre dezvoltator intr-un format de tip cod de octeti, ce se afla in ierarhia codurilor intre codul masina si codul sursa.

O alta caracteristica importanta a acestui limbaj ce ii confera superioritate in raport cu limbajul din care a luat nastere este aceea a orientarii pe obiect, Java fiind o forma non-procedurala de programare, nivelul de abstractizare atingand cote maxime.

Faptul ca Java nu depinde de configuratia hardware sau cea a procesorului si nici de sistemul de operare al mediului pe care ruleaza este controlat prin intermediul diferitelor platforme dedicate diferitelor domenii de aplicabilitate si dispozitive. Amintim in cele ce urmeaza patru dintre astfel ce platforme:

- Java Card: permite rularea aplicatiilor mici Java sa ruleze pe diverse dispozitive de stocare intr-o maniera sigura
- Java ME (Micro Edition): permite rulare aplicatiilor bazate pe Java pe dispozitivele mobile sau dispozitive ce prezinta limitare de spatiu de stocare, afisare si prelucrare
- Java SE (Standard Edition): permite rularea aplicatiilor Java pe dispozitive obisnuite precum laptop-uri, desktopuri, servere si alte dispozitive similare
- Java EE (Enterprise Edition): permite aplicatiilor Java inalte, de tip client-server sa ruleze folosind atat Java SE, cat si API-uri

3.2.3.2 Structura limbajului Java

Din punct de vedere al structurii lexicale, Java este un limbaj sensibil in ceea ce priveste literele mari sau mici, fiind bazat pe setul de caractere Unicode, un succesor al codului ASCII. Imposibilitatea ASCII de a putea reprezenta mai mult de 256 de caractere a determinat aparitia Unicode extins capabil sa defineasca peste un milion de caractere.

Limbajul Java prezinta restrictionarea folosirii unor anumite cuvinte, denumite „cuvinte cheie” in cadrul scrierii de cod deoarece acestea reprezinta deja obiecte, clase, metode sau interfete pe care compilatorul stie sa le interpreteze intr-un anume mod. Aceste cuvinte reprezinta fie denumirile unor tipuri de date precum: boolean, byte, char, float, int, long, short; fie denumirile unor instructiuni: break, case, continue, else, extends, for, goto, if, implements, import, new, return, switch, try, while; fie modalitatile de control asupra metodelor, claselor si variabilelor din java cum ar fi: final, void, protected, public, private, fie referiri la diverse componente precum: class, instance, this. Este, de asemenea, restrictionata folosirea cuvintelor true, false sau null, ce nu reprezinta cuvinte cheie, insa prezinta o semnificatie aparte pentru compilatorul java cu referire la valorile posibile ale anumitor variabile.

3.2.3.3 Literali, separatori si comentarii in Java

Prin literali in limbajul Java se intlege specificarea tipurilor de constante ce pot fi atribuite unor variabile. Distingem astfel, urmatoarele tipuri de literali:

- Integer (Intregi): pot fi normali reprezentati pe 32 de biti sau lungi reprezentati pe 64 de biti, accepta 3 tipuri de baze de numeratie: baza 8, baza 10 si baza 16 si se exprima prin folosirea „cuvantului cheie” int.
- Floating point (Flotanti): reprezinta numerele cu cel putin o zecimala dupa virgula, notatiile exponentiale, valorile normale, prin folosirea sufixului F sau cele duble prin folosirea sufixului D si se reprezinta prin „cuvantul cheie” float sau double, cu specificatia ca in mod predefinit double reprezinta tipul literalilor flotanti.

- Boolean(Logici): reprezinta valorile posibile pentru o variabila logica si anume true (adevarat) sau false(fals) si se identifica prin folosirea „cuvantului cheie” boolean.
- Character (Caracter): reprezinta caracterele codului Unicode, ei pot fi o singura litera sau o secventa de litere marcata intre apostrofuri si se identifica prin folosirea „cuvantului cheie” char
- String (Siruri de caractere): reprezinta un sir de caractere format din niciunul sau mai multe caractere marcate intre ghilimele si se identifica prin folosirea „cuvantului cheie” string

Important de mentionat la acest capitol este faptul ca exista sechete speciale in Java ce sunt intepretate intr-un mod specific cum ar fi:

- „\n” – linie noua
- „\b” – backspace
- „\r” – inceput de rand
- „\f” – pagina noua
- „\t” – tab nou orizontal
- „\”” – ghilimele
- „\\” – backslash
- „\’” – apostrof
-

In ceea ce priveste separatorii in limbajul Java intalnim urmatoarele tipuri de separatori: paranteze rotunde, paranteze patrate, acolade, punct si virgula, virgula si punct. Fiecare dintre acestia are un rol bine definit in cadrul codului Java dupa cum urmeaza:

- ; - rol in finalizarea unei linii de cod
- , - rol in separarea variabilelor in cadrul unei declaratii
- . - rol in separarea numelui unei clase si membrui sai sau intre un membru al unei clase si metodele sale
- () - rol in specificarea parametriilor de intrare ai unei metode, chiar si atunci cand aceasta nu necesita date de intrare, caz in care intre paranteze nu se precizeaza nimic
- [] - rol in specificarea unei variabile ca fiind de tip multime de elemente, dar si in accesarea unui anumit element din cadrul unei multimii.
- {} - rol in specificarea corpului unei metode sau a unei clase, dar si in initializarea unei multimii de elemente.

Comentariile pot fi introduse in limbajul Java prin 3 modalitati diferite, fiecare dintre ele avand o anume semnificatie. Rolul comentariului este acela de a face codul mai usor de inteles persoanelor ce au contact pentru prima data cu acesta, fiind folositor si dezvoltatorului in sensul ca in cadrul unei aplicatii complexe acesta poate reveni mai usor asupra unor linii si isi poate reaminti rapid scopul cu care acestea au fost scrise.

In secenta de cod urmatoare s-a introdus in aplicatia ce face subiectul lucrarii un comentariu de o singura linie prin care se specifica ca ceea ce urmeaza declarat reprezinta meniul de generare orare.

```
//meniu generare orar
meniuGenerare = new JMenu("Generare orare");
baraMeniu.add(meniuGenerare);
```

Mai jos este ilustrata o a doua varianta de introducere a unui comentariu prin care sunt comentate o serie de mai multe linii de cod ce au fost scrise insa s-au dovedit a fi nefolositoare in cadrul prezentei aplicatii, deci s-a dorit omiterea lor in timpul rularii.

```
/*fisiere =
AfisareContinutFolder.numeleFisierelor(numeFacultate.getText().toString().toLowerCase());

System.out.println(fisiere.size());
if(fisiere.size() != 0){
    for(int i = 0; i<fisiere.size(); i++){
        System.out.println("Am intrat aici");
        nf.setText(fisiere.get(i).toString().substring(26,
fisiere.get(i).length()-1));
        nf.setBounds(300, 100 + i, 200, 200);
        nf.setVisible(true);
    }
}*/
```

O a treia forma de comentariu este cea prin care se marcheaza comentariile cu privire la documentarea, iar aceasta poate fi analizata in liniile ce urmeaza:

```
/*
 * Create the panel.
 */
```

3.2.3.4 Operatori

In cadrul limbajului Java distingem urmatoarele tipuri de operatori cu semnificatiile aferente:

- Operatori de asignare: =
- Operatori matematici: + - folosit pentru adunare
 - - folosit pentru scadere
 - * - folosit pentru inmultire
 - / - folosit pentru a obtine catul unei impartiri
 - % - folosit pentru a obtine restul unei impartiri
- Operatori unari: + - folosit pentru a specifica faptul ca o valoare este pozitiva
 - - folosit pentru a nega o expresie
 - ++ - folosit pentru incrementarea cu 1
 - - - folosit pentru decrementarea cu 1

- ! – folosit pentru a nega o expresie din punct de vedere logic
- Operatori relationali: == - folosit pentru a verifica egalitatea a doi operanzi
 != - folosit pentru a verifica inegalitatea a doi operanzi
 >- folosit pentru a verifica daca un operand este mai mare ca celalalt
 <- folosit pentru a verifica daca un operand este mai mic ca celalalt
 >= - folosit pentru a verifica daca un operator este mai mare sau egal cu celalalt
 <= - folosit pentru a verifica daca un operator este mai mic sau egal cu celalalt
- Operatori conditionali: && - folosit pentru a specifica conditia AND
 || - folosit pentru a specifica conditia OR
 ?: - folosit ca prescurtare a expresiei if-then-else

Ca si functie suplimentara a operatorului + este cunoscuta concatenarea sirurilor de caractere ce este evidentata in sevenita de mai jos, unde se atribuie un sir de caractere unei etichete, sirul fiind personalizat cu ajutorul html.

```
link = new JLabel("<html><center>Orare generate cu aplicatia SmartCampus"
+ "</center></html>");
```

3.2.3.5 Variabile, declarare si initializare

Variabilele reprezinta in Java modalitatea de a retine date, dandu-le un nume si avand posibilitatea de a putea fi accesate si manipulate ulterior.

Declararea variabilelor se face intotdeauna inainte de a fi folosite, aceasta operatie constand in specificarea unui tip si a unui nume pentru fiecare variabila. Pe de cealalta parte, initializarea reprezinta operatia prin care unei variabile i se atribuie o valoare cu ajutorul operatorului de asignare =.

Există uneori necesitatea declarării unei variabile care să nu își modifice valoarea pe tot parcursul rularii, astfel ca iau naștere variabilele constante, care se disting de restul variabilelor prin specificarea în declarare a cuvantului final. Alte tipuri de variabile utilizate în programarea orientată obiect se disting în funcție de locul în care este făcută declararea lor în interiorul codului. Amintim aici variabilele definite la nivelul instantelor, valoarea acestora fiind diferită pentru fiecare obiect al unei clase, variabilele definite la nivelul claselor, identificate prin cuvantul static, valoarea acestora fiind disponibilă tuturor metodelor dintr-o anumită clasa, variabile definite local la nivelul metodelor, cu vizibilitate redusă doar în cadrul metodei în care sunt declarate.

Este arătat mai jos modul în care în cadrul aplicației a fost declarată o variabilă la nivelul unei metode. Se observă faptul în cadrul metodei getTableModelAlocari() prin intermediul căreia se conturează continutul tabelului ce urmează să fie afisat în interfața grafică, se declară variabila nrAlocari.

```
private DefaultTableModel getTableModelAlocari() {
```

```

    alocari = AlocareDB.selecteazaalocari();
    int nrAlocari = alocari.size();

    alocariTabel = new Object[nrAlocari][8];
    for (int i = 0; i < nrAlocari; i++) {
        Alocare aloc = alocari.get(i);
        alocariTabel[i][0] = aloc.getId_profesor();
        alocariTabel[i][1] = aloc.getId_materie();
        alocariTabel[i][2] = aloc.getNume();
        alocariTabel[i][3] = aloc.getPrenume();
        alocariTabel[i][4] = aloc.getEmail();
        alocariTabel[i][5] = aloc.getDepartament();
        alocariTabel[i][6] = aloc.getMaterie();
        alocariTabel[i][7] = aloc.getOre_curs();
    }

}
return new DefaultTableModel(alocariTabel, columns);
}

```

3.2.3.6 Vectori

Este uneori necesara folosirea unei variabile ce ocupa un spatiu mai mare in memorie si contine mai multe elemente, adica vorbim de folosirea unui vector. Identic cu modul in care sunt declarate variabilele obisnuite, la declararea vectorilor este necesar sa se specifica tipul elementelor ce compun vectorul si un nume pentru acesta. In plus, apar in sintaxa parantezele patrate ce fac diferenta intre declararea unui vector si cea a altui tip de variabila. Exista, de asemenea, si diferentieri la nivel de initializare, in cazul vectorilor fiind necesara aparitia cuvantului cheie „new”.

Liniile ce urmeaza ilustreaza declararea vectorului columns ce contine elemente de tip sir de caractere.

```

private final static String[] columns = new String[] {"Id profesor", "Id
materie", "Nume", "Prenume", "Email", "Departament", "Materie", "Ore_Curs"};

```

3.2.3.7 Obiecte

Obiectele reprezinta in limbajul Java un mod de instantiere a claselor, fiind adesea folosite pentru a putea transpunere in cod obiecte din lumea reala. Un obiect inglobeaza componente conexe si stari prin care se defineste modul de interactiune cu alte elemente si de actiune a acestora in cadrul unei aplicatii.

Pentru a crea un obiect trebuie realizate operatiile de declarare, prin care se specifica clasa din care obiectul face parte, instantiere, prin care se realizeaza alocarea de memorie, adica se construieste obiectul folosind cuvantul new si initializare, ce este posibila cu ajutorul constructorilor clasei.

Dupa crearea obiectului, acesta poate fi folosit pentru a lucra impreuna cu celalalte elemente la construirea programului software prin apelarea de metode sau accesarea valorilor unor variabile specifice lui.

```

Alocare a = new Alocare();
a.setId_profesor(results.getInt(1));
a.setId_materie(results.getInt(2));
a.setNume(results.getString(3));
a.setPrenume(results.getString(4));
a.setEmail(results.getString(5));
a.setDepartament(results.getString(6));
a.setMaterie(results.getString(7));
a.setOre_curs(results.getInt(8));

```

In secventa de mai sus se remarcă crearea obiectului “a”, instantierea sa și definirea sa ca fiind de tipul Alocare. Mai departe se apelează metode de tip set definite în clasa pe care obiectul o instantiază pentru a stabili valorile atributelor obiectului.

3.2.3.8 Clase

In limbajul Java clasele joacă un rol important, ele reprezentând un model pentru obiectele ce le instantiază. Prin intermediul claselor sunt specificate atribute comune ale obiectelor, dar și metode ale acestora. Pentru că vorbim despre programare orientată pe obiect, vorbim și despre o ierarhie a claselor, astfel că orice clasa dintr-un program Java poate avea un parinte unic sau și clasa superioară și o multitudine de clase inferioare. Există însă și o excepție în acest sens, aceea fiind clasa Object care nu poseda o clasa parinte.

La declararea unei clase se specifică și tipul acesteia prin intermediul modificatorilor, astfel că se diferențiază clasele de tip public, ce pot fi utilizate fără a tine cont de pachetul în care acestea se gasesc sau final, ceea ce înseamnă că acea clasa nu poseda subclase, acest tip fiind preferat în programarea orientată pe obiect deoarece parametrii metodelor trebuie să aparțină de cele mai multe ori unei clase și nu unei subclase.

```

public class An {
    private int id, facultate;
    private String denumire;
    public An(int id, String denumire, int facultate) {
        this.id = id;
        this.denumire = denumire;
        this.facultate = facultate;
    }
    public An() {
    }
    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
    ...
}

```

Se arată mai sus modul de declarare al clasei An din cadrul pezentei aplicații în care sunt declarate în prima fază variabilele ce reprezintă atributul pe care fiecare obiect îl va avea în cazul în care unei instantieri a clasei și anume: id și facultate de tipul întreg și denumire de tipul string.

In continuare se specifica doi constructori, in fapt metode ale clasei, prin intermediul carora se vor crea obiecte ale clasei, cei doi fiind diferentiatii de faptul ca primul are parametrii, reprezentati de atributele declarate anterior, iar cel de al doilea este un constructor gol. Urmatoarele linii releva definirea unor metode ce returneaza id-ul si seteaza id-ul unui obiect de tip An.

3.2.3.9 Clasa speciala Object

Clasa speciala Object reprezinta singura clasa din cadrul limbajului Java ce nu are definita nici o clasa parinte, reprezentand radacina in organizarea ierarhica a claselor Java. Fiecare clasa este o subclasa a acestei clase, preluand de la aceasta metode cu utilizare diferita precum .equals(), metoda folosita pentru a verifica egalitatea si .toString(), metoda folosita pentru conversia in sir de caractere.

Toate obiectele din cadrul unei aplicatii Java sunt descendenti ai clasei Object si pot accesa si eficientiza metodele definite in aceasta deoarece ele nu sunt definite ca fiind final. Se poate spune deci, ca aceasta clasa speciala reprezinta un sablon pentru toate clasele si obiectele ce o succed.

Este prezentat in continuare un exemplu de utilizare a metodei .equals() in cadrul aplicatiei aferente acestei lucrari in care se verifica daca valorile selectate a doua liste de tip drop-down contin caractere.

```
if(choiceMaterie.getItemAt(choiceMaterie.getSelectedIndex()).equals(" ") ==  
false && choiceZi.getItem(choiceZi.getSelectedIndex()).equals(" ") == false  
&& listaSerie.getSelectedIndex() != -1) {...}
```

3.2.3.10 Interpretarea exceptiilor

Clasa java.lang.Thread are proprietatea de a controla exceptiile ce apar de-a lungul rularii unei aplicatii scrisa in limbajul Java. Aceste exceptii reprezinta mai bine spus o modalitate prin care se comunica utilizatorului faptul ca programul nu a rulat pana la final din cauza erorilor.

Exceptiile sunt o unealta foarte importanta in dezvoltarea unui proiect Java deoarece fiind implicat operatorul uman in tot acest proces pot fi introduse inevitabil erori. Atunci cand apare o eroare, in mesajul afisat catre programator exceptiile specifica si randul, dar si clasa unde s-a produs intreruperea programului, dar si motivul pentru care rularea nu s-a finalizat.

```
Exception in thread "AWT-EventQueue-0" java.lang.Error: Unresolved  
compilation problem:  
    sql cannot be resolved to a variable  
    at Main.ProfesorDB.logareprofesor(ProfesorDB.java:43)  
        at GUI.Login$2.actionPerformed(Login.java:161)
```

Este specificat in acest exemplu faptul ca in clasa ProfesorDB la linia 43 s-a produs o eroare, variabila sql fiind necunoscuta compilatorului. Pe ultima linie se specifica si faptul ca aceasta eroare are impact si asupra liniei 161 din clasa Login.

Existenta blocurilor try – catch face posibila evitarea erorilor ca cele prezentate mai sus.

In cadrul aplicatiei din aceasta lucrare s-a folosit un bloc de tip try-catch in care in cadrul try a fost introdus codul ce se doreste a fi rulat in vederea deschiderii unei pagini web cu ajutorul Java, iar in partea de catch se face referire la clasa IOException, o clasa generica specifica operatiilor I/O ce nu ajung sa ruleze complet.

```

if(Desktop.isDesktopSupported()) {
    Desktop desktop = Desktop.getDesktop();
    try {
        desktop.browse(uri);
        link.setForeground(Color.RED);
    } catch (IOException ex) {
        // TODO Auto-generated catch block
        ex.printStackTrace();
    }
}

```

3.2.3.11 Pachete in Java

Adesea in realizarea de aplicatii Java sunt folosite API-uri sau Application Programming Interface, ce reprezinta un set de pachete puse la dispozitie de platforma Java in vederea controlarii accesului si folosirii unor clase si interfete, precum si pentru a evita dezacordurile in ceea ce priveste numirile. Pachetele sunt formate din clase, interfete, adnotari si enumerari asemanatoare prin intermediul carora se gestioneaza denumirile si se protejeaza accesul.

Sunt enumerate in continuare cateva dintre cele mai frecvent utilizate pachete java:

- java.swing – pachet folosit pentru crearea de interfete grafice.
- java.awt – pachet folosit atat pentru crearea interfetelor, cat si pentru grafica si imagini.
- java.awt.event – pachet folosit pentru a putea lucra cu evenimentele componentelor AWT.
- java.sql – pachet folosit pentru ca o aplicatie java sa se poata conecta la o baza de date si pentru a efectua operatii in cadrul acestia.
- java.io – pachet folosit pentru a lucra cu fluxurile de date de intrare si iesire, dar si pentru manipularea sistemelor de fisiere.
- java.lang – pachet utilizat pentru construirea efectiva a limbajului Java.

Pentru a putea utiliza un pachet si toate facilitatile pe care acesta le pune la dispozitie programatorul trebuie sa specifica inainte de a declara clasa ce pachet doreste sa importe. Sintaxa de import a pachetelor este ilustrata mai jos printre-o secventa de cod din cadrul aplicatiei in care se importa cateva dintre cele mai importante pachete, specificandu-se numele acestora.

```

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.awt.image.BufferedImage;
import java.io.File;
import java.io.IOException;
import java.net.URI;
import java.net.URISyntaxException;
import java.net.URL;

```

3.2.3.12 Interfete grafice cu utilizatorul

Crearea unei interfete grafice cu utilizatorul este una dintre cele mai importante etape in cadrul dezvoltarii aplicatiilor Java deoarece prin acestea se poate stabili o legatura stransa intre operatorul uman si software-ul creat. In cadrul prezentei aplicatii s-a utilizat pachetul java.swing pentru crearea unei astfel de interfete din motive strans legate de independenta acestui pachet de platforma pe care aplicatia lucreaza. Spre deosebire de java.awt, pachetul swing prezinta aceasta proprietate, punand la dispozitie numeroase clase si obiecte ce au ca scop obtinerea de ecrane bine construite si eficiente.

Tipul componentelor ce sunt intalnite in acest pachet variază in functie de scopul utilizarii lor, astfel ca putem vorbi despre componente atomice precum JLabel, JButton, JscrollBar, despre componente complexe JList si JTable, despre containere intermedii precum JPanel, JScrollPane si JtabbedPane, despre containere complexe JFrame, JDialog si Jwindow, despre meniuri cum ar fi JMenu si JMenuItem si despe componente specifice editarii de text ca JTextField si JPasswordField.

Desi este un pachet puternic in comparatie cu predecesorul sau java.awt, pachetul javax.tinde sa ii ia locul pachetului swing in cadrul aplicatiilor ce necesita interfete grafice datorita dezvoltarii cu pasi repezi a limbajului Java.

In cele ce urmeaza vor fi descrise cateva dintre componentele acestui pachet ce au fost folosite in cadrul aplicatiei.

JFrame

Folosirea JFrame face posibil lucru cu unul dintre containerele de nivel superior al limbajului Java, in interiorul caruia se permite pozitionarea oricarei componente a pachetului swing. JFrame prezinta, de asemenea, facilitati precum setarea unui titlu specific fiecarei ferestre, dimensionarea acestora si folosirea de butoane cu scopul inchiderii ferestrelor.

Vom ilustra mai jos un exemplu in care s-a creat o fereastra denumita f, careia i s-au setat modalitatea de inchidere predefinita, modalitatea de extensie, dimensiunea dar si pozitionarea in centrul ecranului.

```
JFrame f = new JFrame();
f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
f.setExtendedState(JFrame.MAXIMIZED_BOTH);
f.setSize(2147, 2147);
f.setLocationRelativeTo(null);
```

JPanel

JPanel reprezinta un container ce se afla pe scara containerelor swing intre JFrame si componente atomice, astfel reprezinta un container intermediu in care se pot introduce diverse componente.

In exemplul ce urmeaza a fost creat un JPanel cu denumirea pa, caruia i s-au setat ca limite culoarea de fundal si vizibilitatea, iar apoi a fost adaugat la fereastra conceputa anterior prin scrierea ultimelor doua linii.

```
PanelOrare pa = new PanelOrare();
pa.setBounds(800, 800, 800, 800);
pa.setBackground(Color.white);
pa.setVisible(true);
f.getContentPane().add(pa);
f.setContentPane(pa);
```

JLabel

JLabel-ul reprezinta o componenta prin intermediul careia sunt create etichetele. Etichetele reprezinta in cadrul interfetelor grafice Java modalitatea de comunicare cu utilizatorul, oferind acestuia informatii prin intermediul imaginilor sau text-ului, acestea neputand fi modificate pe parcursul rularii aplicatiei.

Se evidentaiza mai jos un exemplu simplu de creare a unei etichete, careia i se stabeleste un text cu un format personalizat, dar si fontul textului si culoarea acestuia si dimensiunile etichetei.

```
link = new JLabel("<html><center>Orare generate cu aplicatia SmartCampus"
                  + "</center></html>");
link.setForeground(Color.BLACK);
link.setFont(new Font("Arial", Font.ITALIC, 15));
link.setBounds(550, 150, 693, 107);
```

JButton

Acest tip de componente apartin clasei AbstractButton si reprezinta elementul de baza in cadrul unei aplicatii ce prezinta interfata grafica cu utilizatorul deoarece apasarea unui buton face posibila aparitia evenimentelor. Pe baza evenimentelor se definesc mai departe activitatile ce vor avea loc. Este o practica buna ca butonul sa aiba atasat un text sugestiv sau o imagine care sa descrie ceea ce se va intampla daca el v-a fi apasat.

```
incarca = new JButton("Publica orarele");
incarca.setBounds(50, 400, 200, 50);
incarca.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent arg0) {
        if(textLink.getText().length() != 0)
        {
            retineLink();
            JOptionPane.showMessageDialog(null,
                "Link-ul orarelor a fost publicat",
                "Information",
                JOptionPane.INFORMATION_MESSAGE);

        }
        else JOptionPane.showMessageDialog(null,
            "Datele introduse nu sunt valide.\n"
            + "Completați toate campurile.",
            "Error",
            JOptionPane.ERROR_MESSAGE);
    }
});
```

Se poate observa in exemplul precedent crearea butonului incarca cu eticheta “Publica orarele” si dimensiunile specificate. Butonul are rolul ca atunci cand este apasat sa verifice daca un camp de tip text este gol, in cazul in care acesta nu este se retine ceea ce este scris in camp si se afiseaza un mesaj de informare catre utilizator, iar in cazul in care campul este gol si butonul a fost apasat se afiseaza un mesaj de eroare catre utilizator, prin care acesta este avertizat ca pentru a trece mai departe trebuie sa completeze campul text.

JComboBox

JComboBox reprezinta componenta prin care in interfetele grafice Java sunt introduse liste de tip drop – down pentru a limita utilizatorul sa introduca date ce pot afecta aplicatia in mod definitoriu. Aceasta componenta imbina functionalitatile unui camp text cu unui buton, in sensul ca utilizatorul poate selecta ceea ce doreste dintr-o lista predefinita care ii este afisata, iar la apasarea unui element al listei se poate specifica ce evenimente urmeaza a fi executate.

Este prezentat un exemplu in care se creaza un JComboBox format dintr-o lista de siruri de caractere, caruia ii sunt setate limitele, culoarea de fundal dar si primul element, urmand ca toate celelalte elemente sa fie reprezentate de denumirile materiilor pe care un cadru didactic le preda.

```
final JComboBox<String> choiceMaterie = new JComboBox <String>();
choiceMaterie.setBounds(50, 200, 200, 25);
choiceMaterie.setBackground(Color.white);
choiceMaterie.addItem(" ");
String profesor = Login.retineProfesor();
materii = MaterieDB.listamaterii(profesor);
for (int i = 0; i < materii.size(); i++) {
    choiceMaterie.addItem(materii.get(i).getDenumire());
}
```

JTable

JTable face posibila afisarea mai usoara a continutului unei baze de date in interfetele grafice, fiind componenta responsabila de afisarea si modificarea de tabele cu doua dimensiuni. Pentru ca de cele mai multe ori nu se cunoaste dinainte dimensiunea tabelul ce urmeaza a fi afisat sau aceasta este variabila, tabelele sunt afisate de cele mai multe ori in interiorul unui JScrollPane, util atunci cand trebuie sa navigam in cadrul unui tabel de dimensiuni considerabile.

```
tabel = new JTable();
tabel.setBounds(300, 200, 900, 300);
tabel.setAutoCreateColumnsFromModel(true);
tabel.setGridColor(Color.DARK_GRAY);
tabel.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
scrollPane = new JScrollPane(tabel);
scrollPane.setBounds(200, 200, 910, 300);
scrollPane.setVisible(true);
```

Se creaza astfel tabelul “tabel”, ii sunt setate limitele, culoarea, tipul de selectie si faptul ca are coloanele create in functie de modelul pe care il afiseaza. Mai departe tabelul este introdus intr-un JScrollPane.

Pentru a putea popula tabelul cu inregistrari din baza de date se foloseste urmatoarea metoda prin care sunt extrase datele din data si salvate sub forma de lista, care este mai apoi parcursa si salvata in forma unei matrici bidimensionale.

```
private DefaultTableModel getTableModelProfesori() {
    profesori = ProfesorDB.selectezaprofesori();
    int nrProfesori = profesori.size();
    profesoriTabel = new Object[nrProfesori][6];
    for (int i = 0; i < nrProfesori; i++) {
        Profesor prof = profesori.get(i);
```

```

        profesoriTabel[i][0] = prof.getId();
        profesoriTabel[i][1] = prof.getNume();
        profesoriTabel[i][2] = prof.getPrenume();
        profesoriTabel[i][3] = prof.getEmail();
        profesoriTabel[i][4] = prof.getParola();
        profesoriTabel[i][5] = prof.getDepartament();

    }

    return new DefaultTableModel(profesoriTabel, columns);
}

```

3.2.3.13 Comunicarea cu baza de date (JDBC)

Pentru a putea dezvolta aplicatii software care sa acceseze sistemele de gestiune a bazelor de date este necesara folosirea interfetei Java DataBase Connectivity (JDBC) pusa la dispozitie de platforma Java. Aceasta interfata este in sine un API prin intermediul caruia programatorii pot comunica cu baza de date, pot scrie interogari in limbajul structurat SQL si pot obtine rezultate provenite din baza de date in vederea prelucrarii acestora.

JDBC este o interfata standardizata, astfel ca indiferent de programul Java ce o utilizeaza acesta poate stabili o conexiune cu o gama variata de sisteme de gestiune a bazelor de date, sigura constrangere fiind folosirea unui driver specific bazei cu care se doreste conectarea.

In aplicatia dezvoltata pentru aceasta lucratie s-a indicat DriverManager-ului ce driver JDBC sa utilizeze pentru a putea obtine o conexiune cu baza de date MS SQL Server. In cazul de fata clasa com.microsoft.sqlserver.jdbc.SQLServerDriver pune la dispozite implementarea interfetei java.sql.Driver. Pentru incheierea procesului de comunicare cu baza de date s-au precizat in clasa denumita Constante parametrii utilizati pentru accesul la baza, acestia fiind prezentati mai jos.

```

public class Constante {

    // conexiune baze de date
    public static final String user = "root";
    public static final String password = "root0";
    public static final String dbConnectionUrl =
    "jdbc:sqlserver://LAPTOP-4AJ3TNV\\MSSQLSERVER:1433;databaseName=SmartCampus";
    public static final String driverClassName =
    "com.microsoft.sqlserver.jdbc.SQLServerDriver";
}

```

Se observa ca a fost specificat cu ce nume de utilizator sa se realizeze conectarea, parola asociata acestuia pentru serverul MS SQL Server, dar si driverClassName-ul, ce reprezinta clasa care face implementarea java.sql.Driver si dbConnecionUrl-ul, ce specifica printr-un url serverul serviciului MS SQL Server.

Pentru ca a fost folosit ca server laptop-ul personal in url-ul de conectare se observa prezenta numelui calculatorului, urmand ca in cazul in care serverul se afla pe un calculator diferit sa se specifice in interiorul dbConnectionUrl numele nouului mediu de rulare.

Pentru a putea verifica conexiunea cu baza de date si modul in care aceasta functioneaza sau nu, in fiecare clasa in care exista interogari ale bazei s-au introdus urmatoarele linii de cod:

```

private static Connection databaseConnection = null;
static {
    try {
        Class.forName(DRIVER_CLASS_NAME).newInstance();
    } catch (Exception ex) {
        ex.printStackTrace();
    }

    try {
        databaseConnection = DriverManager.getConnection(
            DB_CONNECTION_URL, USER, PASSWORD);
    } catch (SQLException e) {
        e.printStackTrace();
    }
}

```

In aceasta situatie se creeaza mai intai o conexiune nula cu baza de date, care dupa verificarea parametrilor de intrare va fi sau nu stabilita. In continuare primul try are rolul de a construi un canal de comunicatie cu sistemul de baze de date prin crearea unei instante a clasei com.microsoft.sqlserver.jdbc.SQLServerDriver. Daca aceasta prima incercare de conectare esueaza, functia printStackTrace va afisa exceptiile care au cauzat erori. Prin cel de-al doilea try se incearca obtinerea conexiunii efective prin specificarea url-ului serverului, a utilizatorului si parolei cu care se doreste conectarea. In cazul in care unul dintre parametrii se dovedeste a fi gresit, incercarea de conectare esueaza si se afiseaza mesajul de eroare.

De cele mai multe ori erorile de conectare cu baza de date provin din construirea incorecta a url-ului serverului bazei. Astfel, in cele ce urmeaza se va specifica forma corecta a acestuia pentru cazul MS SQL Server:

```
jdbc:sqlserver://LAPTOP-H4AJ3TNV\MSSQLSERVER:1433;databaseName=SmartCampus
```

Am luat ca exemplu url-ul folosit in aplicatia de fata, in care:

- LAPTOP-H4AJ3TNV - reprezinta numele statiei pe care serverul ruleaza
- MSSQLSERVER:1433 - reprezinta numele serverului si portul acestuia
- databaseName=SmartCampus - reprezinta modul de a specifica baza de date la care se doreste stabilirea conexiunii, SmartCampus in acest caz

O alta situatie care genereaza de cele mai multe ori erori este aceea in care portul pentru baza de date nu este deschis sau este specificat incorect, in acest caz utilizatorul trebuie sa modifice setarile TCP/IP in mod corespunzator sau sa determine portul cu care serverul lucreaza.

Statement

Folosind interfata statement dezvoltatorul unei aplicatii poate scrie sevante SQL pe care apoi sa le execute in baza de date. Aceasta interfata sustine alte doua interefete folositoare lucrului cu bazele de date prin intermediul Java precum CallableStatement si PreparedStatement si are implementate metode precum executeQuery, folosita pentru interogarile de baze de date exprimate prin sevanta SQL SELECT, oferind ca rezultat un tabel cu inregistrarile care satisfac cerintele si se regasesc in baza de date.

Ca exemplu se ilustreaza folosirea executeQuery in vederea obtinerii tuturor inregistrarilor din tabelul Profesor al bazei de date SmartCampus.

```

public static List<Profesor> selectezaprofesori(){
    List<Profesor> listaprofesori = null;
    String sql = "select * from profesor";
    try{
PreparedStatement selectStatement = databaseConnection.prepareStatement(sql);
ResultSet results = selectStatement.executeQuery();
.
.
.
}

```

O alta metoda a interfetei statement este executeUpdate, folosita pentru operatii de tip INSERT, UPDATE si DELETE in vederea introducerii, modificarii sau stergerii inregistrarilor in si din baza de date. Se poate observa in secenta de cod urmatoarea folosirea acestei metode pentru a se realiza inserarea unei noi inregistrari in tabelul Profesor al bazei de date a aplicatiei.

```

if (databaseConnection != null) {
    String sql = "INSERT INTO profesor (nume, prenume, email, parola,
departament) VALUES(?, ?, ?, ?, ?)";
    try {
        PreparedStatement insertStatement = databaseConnection
            .prepareStatement(sql);

        insertStatement.setString(1, p.getNum());
        insertStatement.setString(2, p.getPrenume());
        insertStatement.setString(3, p.getEmail());
        insertStatement.setString(4, p.getParola());
        insertStatement.setString(5, p.getDepartament());
        rowsAffected = insertStatement.executeUpdate();
    }
}

```

PreparedStatement

PreparedStatement este interfata ce o completeaza pe cea prezentata anterior prin introducerea unor noi optiuni de interactiune cu baza de date. Ca avantaj evident al acestei interfete distingem posibilitatea de a folosi sintaxa SQL in multiple randuri. Parametrii pe care PreparedStatement ii suporta pot fi de doua tipuri in functie de momentul in care acestia se folosesc si anume parametrii de intrare si parametrii de rulare.

Folosirea parametrilor trebuie facuta intr-un mod adevarat in sensul ca in interiorul interogarii SQL acestia sunt inlocuiti cu semnul intrebarii, ca mai apoi sa l-i se specifica valoarea pentru fiecare in parte. Se arata mai jos modul de a construi un obiect de tip PreparedStatement, cu precizarea ca executia se realizeaza asemanator cu cea a obiectelor de tip Statement.

```

public static String numeprofesor(int idmaterie){
    String nume = null;
    String sql = "select Profesor.nume, profesor.prenume from Profesor
inner join Materie on Materie.profesor = profesor.id where materie.id = ?";
    try {
PreparedStatement selectStatement = databaseConnection.prepareStatement(sql);
selectStatement.setInt(1, idmaterie);
.
.
.
}

```

In sevenita prezentata se doreste selectarea numelui si prenumelui cadrului didactic care predă o anumita materie, precizant ca parametrul de intrare id-ul materiei. Se observă, de asemenea, pregatirea în prealabil a conexiunii cu baza de date.

ResultSet

Interrogarea bazei de date vizează de cele mai multe ori obținerea de rezultate, ce au forma unui obiect de tip ResultSet în cadrul aplicațiilor Java. Aceste obiecte sunt alcătuite dintr-un număr variabil de linii și coloane în funcție de interogarea SQL care se executa în baza, înregistrările gasite în baza și rezultatul obținut.

```
public static String denumire(int idzi) {
    String denumire = null;
    String sql = "select zi.denumire from zi where zi.id = ?";
    try{
        PreparedStatement selectStatement = databaseConnection.prepareStatement(sql);
        selectStatement.setInt(1, idzi);
        ResultSet results = selectStatement.executeQuery();
        while (results.next()) {
            denumire = results.getString(1);
        }
    }
```

In exemplul prezentat se dorește selectarea denumirii unei zile din tabelul Zi aflat în baza de date a aplicatiei în funcție de id-ul acesteia. Se observă construirea obiectului de tip ResultSet, în cadrul căruia sunt parcuse toate înregistrările linie cu linie prin metoda `getDataType()`, unde `DataType` reprezintă tipul de date din baza de date, iar coloanele sunt parcuse prin specificarea numărului acestora între parantezele metodei `getDataType()`.

Partea a II-a - Contributii ale proiectului

4 Motivatie

Acest capitol este dedicat prezentarii modului in care tema lucrarii de fata a fost aleasa si abordata, dar si a modului in care au fost gasite si implementate solutii in vederea obtinerii unei aplicatii robuste, sigure si cu o interfata usor de utilizat pentru modelarea si gestionarea automata a orarului in cadrul unei facultati.

4.1 Problema abordata

Asa cum a fost specificat atat in introducere cat si in celalalte capitole anterioare prezenta lucrare se concentreaza in jurul analizei, proiectarii si implementarii unei aplicatii de modelare si gestionare a orarului unei facultati.

Desi in prezent exista numeroase aplicatii ce au ca scop automatizarea diferitelor procese din domeniul gestiunii unei institutii de invatamant, pornind de la software-urile de contabilitate, continuand cu cele de administrare a orarului si salilor de studiu si incheind cu cele de organizare a datelor personale si scolare ale studentilor, niciuna dintre acestea nu abordeaza modelarea orarului din perspectiva preferintelor cadrelor didactice.

Am ales un astfel de punct de vedere considerand ca este important ca pe tot parcursul unui semestru sa nu aiba loc schimbari majore in ceea ce priveste structura orelor de curs. De cele mai multe ori apar situatii neplacute, in care desi a fost alocat un interval orar pentru o anume materie, cadrul didactic responsabil de aceasta nu poate fi prezent in fata studentilor sau din diferite motive. Am dorit sa elimin constrangerile impuse de orar prin exprimarea libera a preferintelor in cadrul unei aplicatii, care sa centralizeze toate optiunile si sa genereze in mod automat orarul pe baza acestora.

In vederea realizarii unui astfel de software s-a pornit in prima faza de la studierea modului in care sunt elaborate in prezent orarele si posibilitatile pe care cadrele didactice le au in legatura cu modificarea acestora. Am realizat in aceasta etapa faptul ca acest proces este realizat intr-o maniera neautomatizata, in care se tine cont doar de capacitatea si disponibilitatea salilor de studiu, punctul de vedere al cadrului didactic jucand un rol secundar in tot acest context.

Pentru a putea implementa conceptul acestei lucrari s-au realizat mai departe modele care sa prelucreze lumea reala cat mai aproape de adevar si care sa ajute la o transpunere a unor concepte concrete intr-un limbaj de programare care se va ocupa de latura de automatizare a acestui proiect.

Desigur ca dezvoltarea unei astfel de aplicatii se bazeaza in mod indisutabil si pe alegerea unui algoritm de lucru extrem de precis, care sa aiba in vedere toate componentele sistemului, dar si toate starile in care acesta poate sa se gaseasca la un anumit moment de timp. Aici ne referim strict la tot ceea ce inseamna un orar din punct de vedere al elementelor din care este alcătuit si anume: zilele saptamanii, intervale orare, sali ca spatiu de studiu si capacitatea acestora, materii, cadre didactice, ani de studiu si grupuri de studenti, dar ne raportam si la ceea ce se intampla atunci cand exista suprapunerile de preferinte sau sali indisponibile. Algoritmul ales nu trebuie sa fie unul complex, dificultatea sa redusa aducand un plus in economia timpului de procesare a informatiilor si afisare a acestora.

Problema de punere in aplicare a algoritmului si de transpunere a lumii reale in obiecte de programare a fost completata de crearea unei baze de date solida care sa sustina aplicatia din punct de vedere a stocarii informatiilor si de comunicarea sa cu o platforma de programare, in care sa poata fi scris codul sursa, in vederea compilarii lui si rularii aplicatiei.

4.2 Solutia propusa

Odata cu modelarea sistemului folosind limbajul UML si aprofundarea principiilor fundamentale ale algoritmilor, ale bazelor de date si ale limbajelor de programare orientata pe obiect a luat nastere rezolvarea problemei de prelucrare si gestionare a orarelor, ce s-a concretizat mai apoi intr-o implementare unei aplicatii care sa raspunda cerintelor lucrarii de fata. Dezvoltarea a inceput intr-o maniera simplista, care s-a complicat pe parcurs, pe masura ce au aparut situatii neprevazute din punct de vedere al functionarii algoritmului ales initial, dar si din perspectiva transformarii lumii reale in limbaj Java si limitarile care deriva din aceasta.

Astfel s-a ajuns la forma finala a aplicatiei de fata in care in centrul atentiei a fost gasirea unui algoritm de optimizare pe restrictii care sa satisfaca atat nevoia de performanta ridicata, cat si nevoia de solutionare a problemelor si conflictelor ce pot aparea intr-o maniera care sa nu afecteze functionarea sistemului si sa asigure o precizie cat se poate de ridicata.

Definitia algoritmilor a fost respectata intocmai, astfel ca au fost definite o serie de etape logice, in urma rezolvării carora s-a ajuns la indeplinirea sarcinii de modelare a orarului in functie de preferintele introduse de cadrele didactice. S-au definit in prima faza ceea ce reprezinta specificatiile de intrare in cadrul algoritmului de fata si s-a ajuns la urmatoarea abordare: cadrul didactic trebuie sa isi poata alege ziua in care doreste sa inregistreze o preferinta, materia pentru care doreste sa execute acest lucru, dar si seria la care va preda aceasta materie. Tot ca date de intrare au fost definite si ora la care cadrul didactic doreste sa inceapa predarea, dar si preferinta pe care acesta o are. Datele de iesire sunt reprezentate in tot acest context de datele de intrare inregistrate in baza de date a aplicatiei, insa sunt restrictionate de algoritm pentru a nu aparea situatii neplacute precum suprapuneri atat din punct de vedere temporal, cat si din punct de vedere al spatiului de studiu.

Se va descrie in continuare algoritmul utilizat in procesul de dezvoltare, modul sau de implementare putand fi studiat in Anexa 1, paragraful dedicat algoritmului de modelare a orarului in functie de preferinte.

Asa cum a fost amintit mai sus, vorbim despre un set de date de intrare de o dimensiune considerabila, pornind de la alegerea zilei in care se doreste a fi facuta o inregistrare si sfarsind cu preferinta setata pentru inregistrarea in cauza. Apare in aceasta situatie o problema legata de posibilitatea de alocare a intervalului orar preferat, ceea ce conduce la alegerea unui algoritm de optimizare cu restrictii. Restrictiile reprezinta de fapt relatiile dintre variabile, avand totodata si rolul principal in definirea variabillor decizionale.

Faptul ca domeniul ales este unul complex, ce depinde de o gama variata de factori intareste, de asemenea, ideea de rezolvare a unei probleme cu restrictii ce necesita „spargerea” in probleme mai mici si rezolvarea pe rand a acestora pentru a se putea obtine rezultatul dorit cu o performanta si o acuratete demne de luat in seama.

Algoritmul de functionare al aplicatiei poate fi exprimat in limbaj natural astfel:

Pasul 1: Cadrul didactic se autentifica in aplicatie cu email-ul si parola aferente contului sau si selecteaza din lista de roluri disponibile rolul de „Profesor”.

Pasul 2: Daca autentificarea reuseste se va trece la executia pasului 3, iar daca autentificarea esueaza se va opri executia programului pana in momentul in care se vor introduce date de autentificare corecte. Se observa astfel restrictionarea din punct de vedere al rolurilor in cadrul unei facultati.

Pasul 3: Cadrului didactic ii este afisata interfata unde poate inregistra o preferinta.

Pasul 4: Cadrul didactic alege una din zilele lucratoare ale saptamanii. Acest pas a fost restrictionat in ceea ce priveste zilele de lucru ale saptamanii, astfel ca nu se pot aloca intervale orare in zilele de weekend.

Pasul 5: Cadrul didactic alege una dintre materiile din lista disponibila. Aceasta lista este afisata pentru fiecare cadru didactic in parte in functie de materiile pe care acesta le predă, observandu-se aici o restrictionare de alocare a intervalelor bazata pe propriile subiecte abordate, fara a putea face inregistrari pentru subiectele dedicate altor cadre didactice.

Pasul 6: Cadrul didactic alege seria pentru care doreste sa aloce un interval orar. Lista seriilor disponibile este afisata in functie de planul de invatamant al acestora. Vorbim aici despre restrangerea posibilitatilor de a inregistra o preferinta pentru o anumita materie in orarul unei serii ce nu are prevazuta in planul de invatamant materia respectiva.

Pasul 7: Cadrul didactic apasa butonul de vizualizare si poate vedea inregistrările ce se gasesc in baza de date a aplicatiei pentru ziua si seria de studenti aleasa, in functie de semestrul in care materia respectiva apare in planul de invatamant al seriei.

Pasul 8: Cadrul didactic apasa butonul de adaugare pentru a putea aloca un interval orar.

Pasul 9: Cadrul didactic alege ora la care doreste sa inceapa predarea materiei si seteaza preferinta pentru aceasta. Alegerea orei este, de asemenea, limitata intre ora 8 si ora 18, avand in vedere faptul ca orele de curs nu pot incepe mai devreme de ora 8 dimineata si nu se pot incheia mai tarziu de ora 20, iar in ceea ce priveste preferintele ce pot fi setate acestea trebuie sa aiba una dintre valorile „Favorit”, „Indiferent” sau „Nedorit”.

Pasul 10: In functie de ora aleasa si preferinta setata se stabileste daca alocarea poate avea loc sau nu. In acest sens au fost definite cateva reguli de alocare:

- daca nu este disponibila nicio sala de cursuri in intervalul orar selectat ca „Favorit”, alocarea esueaza, disponibilitatea salilor calculandu-se in prealabil cu ajutorul bazei de date unde sunt inscrise informatii despre gradul de ocupare al salilor;

- daca intervalul orar ales, ce este cuprins intre ora de inceput setata de cadrul didactic si ora de final calculata ca suma dintre ora aleasa si durata materiei inscrisa in planul de invatamant al seriei, se suprapune cu un alt interval orar setat ca „Favorit” sau cu o fractiune dintr-un interval orar setat ca „Favorit”, alocarea esueaza;

- daca materia pe care cadrul didactic doreste sa o inregistreze pentru un anumit interval orar cu preferinta „Favorit” apare deja in orarul seriei ca fiind repartizata cu preferinta „Favorit”, alocarea esueaza;

- daca cadrul didactic a selectat un interval orar ca fiind „Favorit” si anterior a alocat ca „Favorit” acelasi interval orar, in aceeasi zi, in acelasi semestru, indiferent pentru ce materie sau serie de studenti, alocarea esueaza;

- daca un cadrul didactic doreste sa seteze ca „Favorit” un interval orar ce se suprapune cu o fractiune sau un intreg interval orar setat deja ca „Indiferent” sau „Nedorit”, alocarea se efectueaza, gradul de preferinta „Favorit” avand prioritate.

Se observa, astfel, in figura urmatoare (vezi Figura 8.), modul grafic in care au fost tratate situatiile in care sistemul se poate afla in incapacitatea de a lua o decizie. Aceste circumstante nu pot fi eliminate prin prisma faptului ca ele nu pot avea loc, sistemul trebuiind sa fie pregatit sa gestioneze posibile greseli ce pot aparea in mod intentionat sau nu. Nu putem exclude variantele in care un cadrul didactic seteaza acelasi interval ca ”Favorit”, ceea ce ar insemana ca un om sa se poata prezenta fizic in doua locuri diferite, dar nici situatii precum cea in care o anumita serie sa aiba alocate doua intervale orare pentru aceeasi materie, ceea ce semnifica o supraincarcare de

materie din punct de vedere al planului de invatamant. De asemenea, nu putem permite ca alocarea de timp sa fie posibila fara a fi admisa si o alocare de spatiu fizic in care sa aiba loc cursul dedicat unei materii si nu pot fi realizabile suprapuneri temporale in ceea ce priveste seria, indiferent de materiile predate, deoarece acest lucru ar inseamna ca studentii sa trebuiasca sa fie prezenti la doua cursuri diferite in acelasi timp.

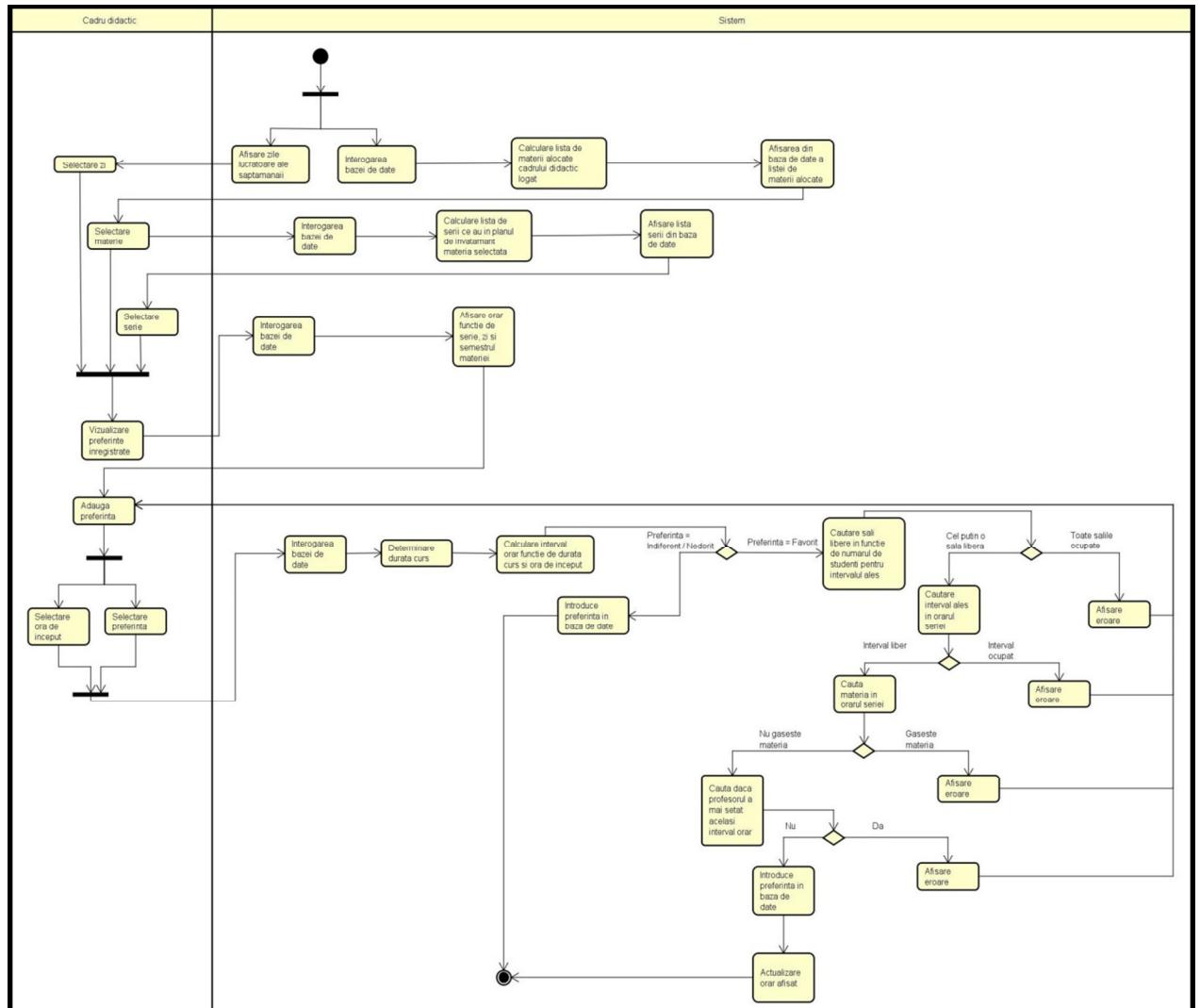


Figura 8. Algoritmul pentru inregistrarea preferintelor

Pentru a putea prezenta mai in detaliu solutia aleasa s-a folosit o reprezentare sub forma de diagrame de activitati a intregului mod de functionare al aplicatiei. Se vor expune in continuare in ordine cronologica toate subprocesele ce intregesc sistemul.

Procesul de logare

Pentru a putea dezvolta un software sigur din punct de vedere al sarcinilor indeplinite de catre utilizator s-a ales definirea unei modalitati de autentificare in aplicatie, putand astfel fi identificate trei roluri distincte cu permisiuni diferite in cadrul aplicatiei. Exista, astfel trei niveluri de autentificare, acestea fiind: administratorul aplicatiei, cadrul didactic si studentul. Fiecare dintre aceste roluri au definite functionalitati diferite, reducand astfel posibilitatea de a introduce in sistem date ce pot provoca situatii conflictuale.

In diagrama ce urmeaza (vezi Figura 9.) sunt ilustrati pasi din cadrul procesului de logare. Se poate observa ca aplicatia este lansata de catre utilizator, indiferent de rolul acestuia, urmand ca sistemul sa afiseze fereastra in care se pot introduce credentialele de logare. Utilizatorul trebuie sa introduca email-ul si parola asociate contului sau dar trebuie sa specifiche si rolul pe care il are in cadrul rularii aplicatiei.

Daca utilizatorul nu are un cont pentru a putea folosi software-ul atunci acesta trebuie sa solicite inregistrarea. Pentru a finaliza procesul de cerere de inregistrare utilizatorul trebuie sa completeze in fereastra pusa la dispozitie de catre sistem datele cu care doreste sa creeze un cont. Aplicatia v-a trimite mai departe catre administratorul ei un email cu detaliiile solicitarii.

Pe de alta parte, daca utilizatorul detine un cont de autentificare, dupa introducerea datelor, software-ul va verifica folosind baza de date informatiile introduse. Daca credentialele au fost verificate ca fiind valide, se va deschide fereastra dedicata rolului utilizatorului in cauza urmand sa se desfasoare subprocesul aferent acestuia cu permisiunile corespunzatoare. Daca, in schimb, credentialele nu sunt valide, utilizatorul va primi un mesaj de eroare, sistemul reafisand fereastra in care se pot introduce datele de autentificare.

Astfel, niciun cadru didactic sau student nu se poate autentifica in aplicatie cu rolul de administrator, acesta fiind unul extrem de important din punct de vedere al gestiunii datelor introduse in aplicatie, avand, totodata si o responsabilitate ridicata in ceea ce priveste generarea si publicarea orarelor in cadrul unei facultati. In acelasi timp, nici studentul sau administratorul nu poate accesa un cont dedicat cadrelor didactice, deoarece doar acestea pot inregistra preferinte legate de sustinerea orelor de curs.

Ceea ce s-a definit ca fiind functionalitate comună pentru toate niveluri de permisiuni este doar partea de vizualizare a orarelor, ce constituie, de asemenea si unica actiune pe care studentul o poate realiza in cadrul aplicatiei pana la momentul de fata.

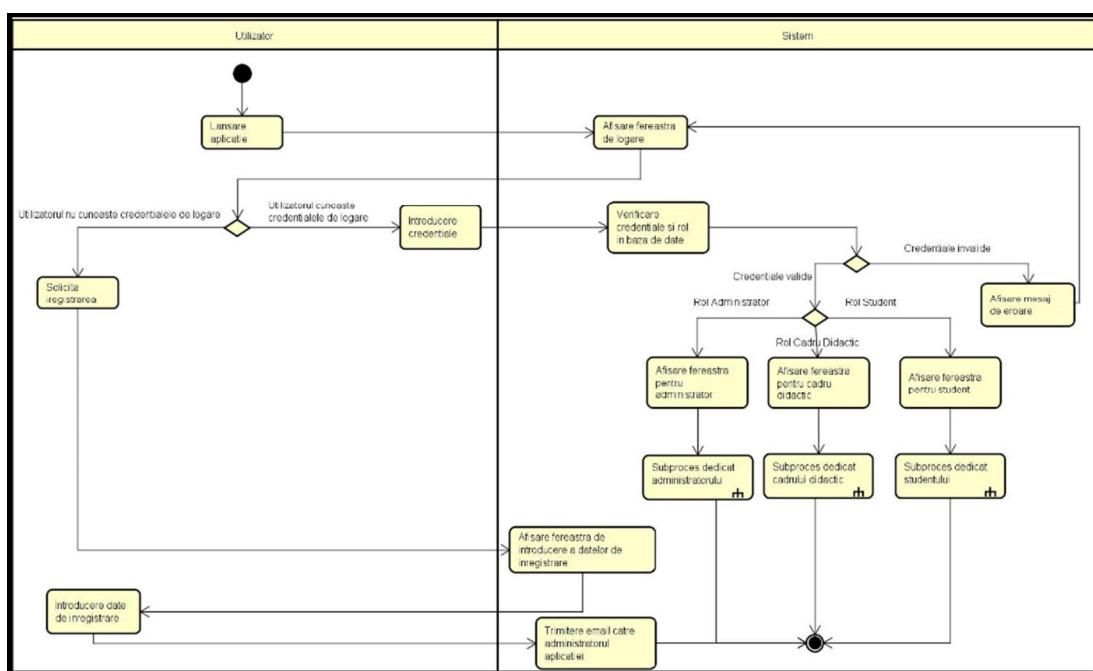


Figura 9. Procesul de logare

Subproces dedicat administratorului

Subprocesul dedicat administratorului reprezinta totalitatea functionalitatilor sistemului puse la dispozitie pentru rolul de administrator si permisiunile acestuia.

Dupa cum se poate observa in reprezentarea de mai jos (vezi Figura 10.) administratorul poate selecta meniuul dorit, aceasta actiune fiind urmata de afisarea de catre sistem a fereastrii corespunzatoare meniului selectat. Administratorul poate alege sa gestioneze inregistrarile din baza de date prin introducerea de noi informatii, prin actualizarea informatiilor existente sau prin eliminarea informatiilor ce nu mai sunt necesare sistemului. Toate aceste operatii sunt realizate cu ajutorul sistemului care comunica cu baza de date si isi concentreaza atentia asupra evitarii erorilor de continut ce pot aparea.

O alta functionalitate specifica rolului de administrator este reprezentata de gestionarea orarelor. In meniul pentru orare personalul avizat poate genera orare in functie de facultatea si de semestrul pentru care doreste sa se realizeze acest lucru. Sistemul ajuta si in acest caz la incheierea activitatii cu succes interogand baza de date si extragand informatiile in functie de parametrii introdusi de catre administrator, generand orarele pentru fiecare an si serie in format tabelar si salvandu-le la o adresa de pe statia la care se executa operatia. Tot in meniul pentru orare, administratorul poate alege sa publice orarele salvate local pentru a le pune la dispozitie si celorlalți membri ai facultatii. Astfel, se va salva in baza de date adresa publica la care pot fi vizualizate ultimele versiuni de orare. Pentru partea de vizualizare a orarelor, sistemul extrage din baza de date calea cea mai noua varianta de orare si redirectioneaza administratorul catre acea adresa prin intermediul unui link, deschizand o pagina web catre un director partajat in care acestea se afla.

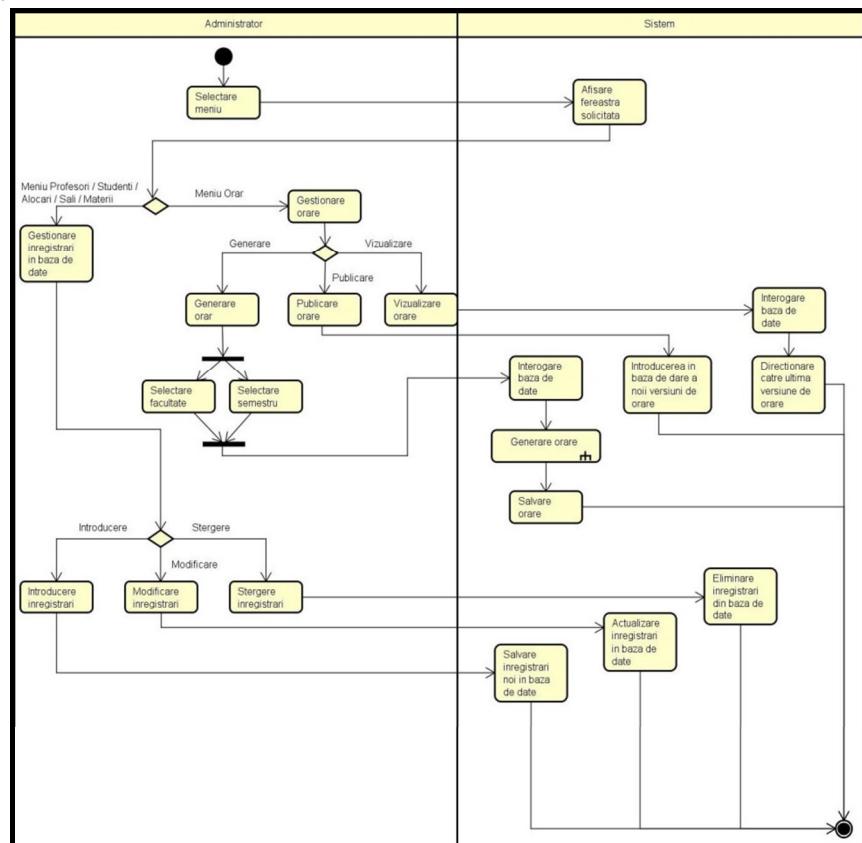


Figura 10. Subproces dedicat administratorului

Algoritmul pentru generarea orarelor

In figura anterioara (vezi Figura 10.) se poate remarcă faptul că generarea orarelor reprezintă de fapt un subproces al sistemului, a cărui mod de funcționare este ilustrat în imaginea urmatoare (vezi Figura 11.), în care se descrie într-o manieră grafică algoritmul conceput pentru generarea orarelor. Se observă, de asemenea, faptul că acest algoritm este urmat în totalitate de către sistem, factorul uman nefiind implicat în desfasurarea lui, acest fapt sustinând ideea de generare automată a orarului ce face subiectul lucrării de fata.

In limbaj uzuial se poate exprima acest algoritm astfel:

Pasul 1: Sistemul selectează din lista anilor de studiu ai facultății selectate de către administrator fiecare înregistrare. Aceasta listă se obține prin interogarea bazei de date.

Pasul 2: Pentru fiecare an selectat, sistemul identifică lista de serii și extrage fiecare înregistrare. Aceasta listă se obține, de asemenea, cu ajutorul bazei de date a aplicației.

Pasul 3: Pentru fiecare serie, sistemul extrage din baza de date listă de materii din planul de învățământ al seriei.

Pasul 4: Sistemul interoghează baza de date și extrage listă de materii ce se află deja înregistrate în orarul seriei.

Pasul 5: Sistemul verifică dacă fiecare materie din planul de învățământ este deja înregistrată în orarul seriei.

Pasul 5.1: Dacă toate materiile sunt înregistrate în orar, se generează orarul în format XLS și se încheie algoritmul.

Pasul 5.2: Dacă cel puțin o materie nu este înregistrată în orar, se calculează primul interval disponibil în orarul seriei.

Pasul 5.2.1: Se verifică disponibilitatea salilor pentru intervalul gasit și numarul de studenți ai seriei.

Pasul 5.2.2: Dacă nu s-a gasit nicio sală disponibilă, se calculează urmatorul interval disponibil în orarul seriei și se reia algoritmul de la pasul 5.2.1.

Pasul 5.2.3: Dacă s-a gasit cel puțin o sală disponibilă, se verifica disponibilitatea cadrului didactic ce are alocată materia.

Pasul 5.2.3.1: Dacă cadrul didactic este indisponibil, se calculează urmatorul interval disponibil în orarul seriei și se reia algoritmul de la pasul 5.2.1.

Pasul 5.2.3.2: Dacă cadrul didactic este disponibil, se adaugă materia în orarul seriei și se reia algoritmul de pasul 4.

Modul de implementare al acestui algoritm poate fi studiat în Anexa 1, paragraful dedicat algoritmului de generare automată a orarului.

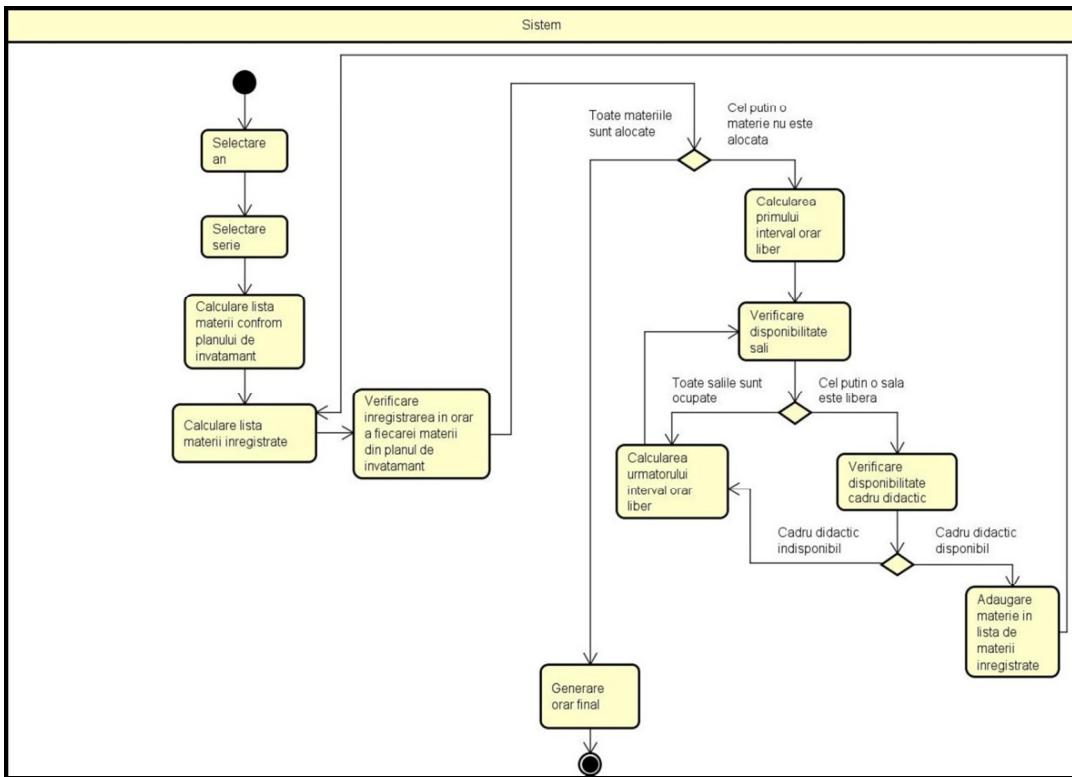


Figura 11. Algoritmul pentru generarea orarelor

Subprocesul dedicat cadrului didactic

In diagrama de activitati urmatoare (vezi Figura 12.) a fost realizata o reprezentare grafica in care sunt evidenitate functionalitatatile dedicate cadrelor didactice. Se poate observa cum cadrul didactic poate alege meniul pe care il doreste, iar sistemul raspunde prin afisarea ferestrei solicitate.

In cazul in care cadrul didactic alege sa isi exprime preferintele in legatura cu setarea intervalelor orare pentru propriile materii se parurge algoritmul de inregistrare a preferintelor (vezi Figura 8.) acestea fiind introduse in baza de date. Daca algoritmul se incheie prin afisarea de erori, sistemul reafiseaza fereastra in care pot fi exprimate preferintele, fortand astfel cadrul didactic sa reia procesul fara a le introduce pe cele initiale in baza de date.

De asemenea, cadrul didactic poate vizualiza orarele disponibile, aceasta operatie realizandu-se identic cu cea in care administratorul doreste sa vizualizeze orarele. Sistemul interogheaza baza de date pentru a vedea care este ultima versiune de orare publicata si pentru a putea directiona cadrul didactic catre adresa web la care se gaseste directorul ce le contine pe acestea.

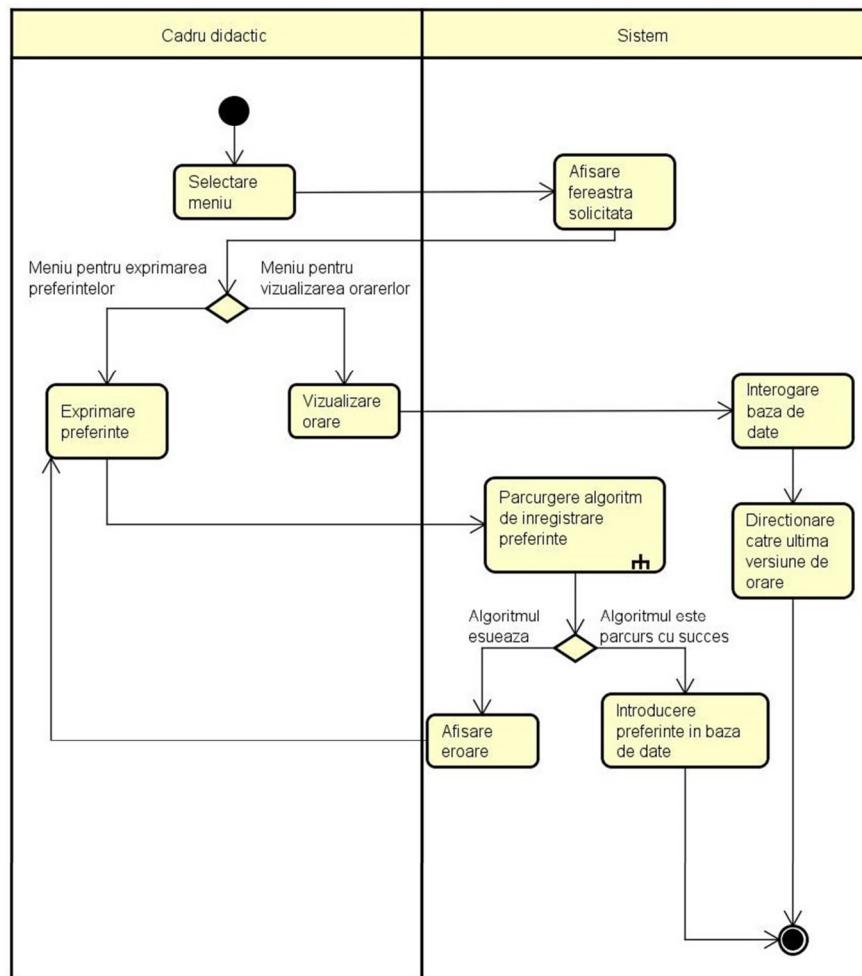


Figura 12. Subprocesul dedicat cadrului didactic

Subprocesul dedicat studentului

In cazul in care un utilizator se autentifica in aplicatie cu rolul de student, acesta are la dispozitie o singura functionalitate a sistemului si anume vizualizarea de orare. Ca si in cazul administratorului si al cadrelor didactice, aceasta functionalitate este realizata cu ajutorul sistemului care calculeaza ultima versiune de orare cu ajutorul interogarii bazei de date si conduce studentul, in acesta situatie, catre adresa paginii web la care se gaseste directorul ce contine orarele. La aceasta adresa studentul poate vizualiza orarul dorit sau poate chiar sa il descarce pe propria statie.

Tot acest subprocess este reprezentat in figura urmatoare cu ajutorul unei diagrame de activitati (vezi Figura 13.).

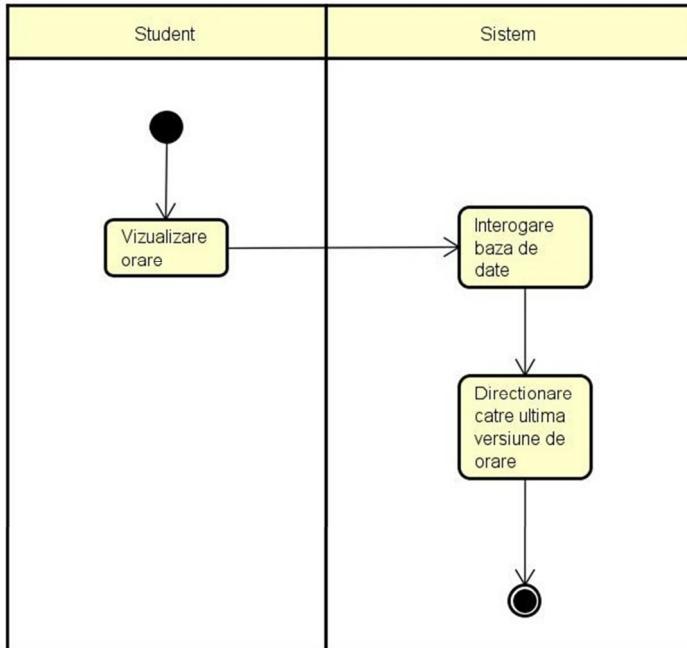


Figura 13. Subprocesul dedicat studentului

5 Metoda de dezvoltare

Acest capitol este dedicat descrierii metodelor de implementare a aplicatiei ce face subiectul acestei lucrari. Descrierea software-ului implementat va fi facuta atat din perspectiva constructiei modelului si analizarii acestuia, cat si din perspectiva mediilor de programare folosite pentru a transpunere prototipul creat in limbaj de programare si a obtine functionalitatile dorite.

5.1 Analiza si specificarea cerintelor

Prima etapa in realizarea aplicatiei de fata, ce are ca scop facilitarea procesului de modelare si generare automata a orarelor in cadrul unei facultati, a fost, asa cum este firesc, etapa de analiza si specificarea cerintelor. Acest prim pas in dezvoltarea software-ului de fata a avut ca scop crearea unui model al sistemului, un model solid si lipsit de ambiguitati, care sa ajute la intelegherea clara a cerintelor si la formalizarea acestora. De asemenea, in aceasta etapa au fost identificate si o serie de inconsistente ce au fost rezolvate prin adaptarea cerintelor in vederea obtinerii unei aplicatii robuste, care sa satisfaca specificatiile.

Inceperea dezvoltării sistemului ce face subiectul acestei lucrari, a fost marcata de definirea tuturor componentelor ce intra in alcătuirea unui orar, dar si de definirea rolurilor si a permisiunilor factorilor umani ce pot folosi aplicatia. Astfel, cerinta de securitate a aplicatiei a fost respectata prin crearea a trei roluri distincte pentru factorul uman in cadrul aplicatiei, fiecare dintre acestea avand functionalitati specifice, evitand aparitia posibilitatii introducerii,

modificarii sau stergerii in mod eronat a datelor folosite de catre sistem, aceste actiuni avand o importanta deosebita in functionarea corecta a aplicatiei.

Cele trei nivele de permisiuni posibile vor fi definite in cele ce urmeaza, pentru fiecare specificandu-se si modul in care utilizator poate interactiona cu sistemul.

Nivelul Administrator

In cazul utilizatorului de tip administrator vorbim, in contextul aplicatiei de fata, de rolul cel mai important in gestionarea datelor ce compun un orar in cadrul unei facultati, dar si a datelor despre cadrele didactice si studentii facultatii.

Pentru acest utilizator au fost definite permisiuni de introducere, actualizare si stergere de date precum materii ce alcatuiesc planul de invatamant al unei serii din cadrul facultatii, alocarii ale cadrelor didactice pentru diverse materii, sali de studii ca spatiu efectiv unde se pot tine ore de curs. De asemenea, administratorul poate gestiona si informatiile personale ale cadrelor didactice si ale studentilor.

Cea mai importanta functionalitate specifica acestui nivel este aceea a generarii si publicarii de orare. Ne referim aici la faptul ca prin selectarea unei facultati si a unui semestru administratorul poate obtine cu ajutorul bazei de date a sistemului toate orarele pentru fiecare serie de studenti in parte, acest proces fiind complet automatizat, nemaifiind necesare actiuni suplimentare ale factorului uman. Publicare orarelor, inseamna, de fapt, incarcarea acestora la o anumita adresa web si facerea lor vizibila catre toti ceilalți utilizatori ai aplicatiei.

Nivelul Cadru Didactic

Cadrul didactic reprezinta pentru aplicatia lucrarii de fata factorul uman cel mai important in procesul de modelare a orarului unei facultati. In acest sens, cadrele didactice pot selecta ziua, materia, seria si intervalul orar pe care doreste sa il fixeze in programul semestrial al seriei.

O a doua interacciune pe care un utilizator de tip cadru didactic o poate avea cu aplicatia este reprezentata de posibilitatea de a vizualiza orarele din cadrul facultatii prin simpla accesare a unei adrese web.

Nivelul Student

Acest nivel de permisiune este cel mai limitat dintre toate nivelele aplicatiei deoarece un utilizator cu rol de student nu poate decat sa vizualizeze orarele la adresa web unde se gasesc acestea.

Pentru a pute face aceasta separatie de permisiuni a fost identificata necesitatea crearii unei forme de autentificare in aplicatie, forma bazata pe introducerea de credentiale unice pentru fiecare utilizator, indiferent de rol si specificarea atributiilor pe care acesta le are in cadrul facultatii. S-a avut in vedere si situatia in care utilizator poate sa nu detina date de autentificare, specificandu-se astfel cerinta de creare a unui mod de solicitare de inregistrare, ce se va concretiza mai departe printr-o interfata in care utilizatorul, fie el cadru didactic sau student, sa poate introduce datele cu care doreste sa acceseze aplicaia, acestea fiind trimise mai departe catre analiza administratorului aplicatiei, care are, de asemenea, si rol in solutionarea acestor cereri.

Din punct de vedere al interacțiunii utilizatorului cu sistemul s-a precizat cerinta de creare de interfeete grafice diferențiate pe nivelele de permisiuni, prin intermediul carora fiecare utilizator poate indeplini doar sarcinile specifice rolului sau.

Mai departe au fost identificate cerintele pe care algoritmul de modelare al orarului, dar si cel de generare al acestuia trebuie sa le indeplineasca pentru a putea atinge scopul aplicatiei si anume acela de obtinere a orarelor din cadrul unei facultati. Astfel, pentru metoda de modelare s-a definit ca exigenta urmarirea preferintelor introduse de cadrul didactic pentru intervalul orar pe care acesta doreste sa il fixeze. Cu alte cuvinte, fiecare cadru didactic atunci cand doreste sa introduca un interval orar trebuie sa selecteze si preferinta pe care o are in raport cu acesta din urma. Spre exemplu, pentru a putea stabili un slot orar pentru o anumita materie, cadrul didactic are la dispozitie trei optiuni: „Favorit”, „Indiferent” sau „Nedorit” prin care isi poate exprima opinia cu privire la posibilitatea sustinerii orelor de curs. Fiecare dintre aceste optiuni are un grad diferit de importanta, astfel ca „Favorit” reprezinta intervalul orar preferat de cadrul didactic, urmat de „Indiferent”, ce constituie intervalul orar in care cadrul didactic este disponibil sa sustina cursul, dar reprezinta in acelasi timp si lipsa acestuia de a avea o preferinta specifica pentru acest interval. „Nedorit” reprezinta optiunea pe care cadrul didactic o are la dispozitie pentru a putea exprima imposibilitarea de a sustine ore de curs intr-un anume interval orar.

Spre deosebire de algoritmul de modelare, cel de generare trebuie sa respecte cerinta de integralitate a datelor, in sensul ca daca un cadru didactic nu inregistreaza nicio preferinta legata de o anumita materie ce se regaseste in planul de invatamant al unei serii, aceasta nu poate lipsi cu desavarsire din orarul seriei, ea trebuind sa fie alocata in mod automat, fara a trebui sa se mai tina cont de preferintele cadrului didactic ce sustine cursuri pentru acea materie.

In figura urmatoare (vezi Figura 14.) a fost modelat sistemul din punct de vedere al componentelor ce alcataiesc un orar in cadrul unei facultati. Modelarea s-a realizat cu ajutorul unei diagrame de clase in care au fost reprezentate toate elementele ce au rol important in procesul de modelare a unui orar, dar si relatiile dintre acestea. Componentele au fost reprezentate sub forma de clase, pentru fiecare specificandu-se atributele definitorii, dar si operatiile pe care le pot efectua. In ceea ce priveste relatiile dintre elemente, acestea sunt caracterizate prin gradul de cardinalitate, dar precizeaza si rolul pe care clasele il au in cadrul fiecarei relatii.

Distingem astfel, faptul ca un interval orar este asociat cu o sala, o materie si o preferinta printr-o relatie de tipul unu-la-mai-multi, aceasta aratand faptul ca o anumita sala poate fi ocupata in niciunul, unul sau mai multe intervale orare, in timp ce o materie, poate fi setata in niciunul, unul sau mai multe intervale orare, iar fiecare preferinta se poate regasi la niciunul, unul sau mai multe intervale orare. Exista o relatie de asociere si intre intervalele orare si fiecare serie, aceasta din urma putand avea setat la un anumit moment al existentei aplicatiei niciunul, unul sau mai multe intervale orare. S-a ilustrat printr-o relatie de compositie cu multiplicitate unu-la-mai-multi faptul ca fiecare zi lucratoare din saptamana poate avea setat niciunul, unul sau mai multe intervale orare.

In ceea ce priveste fiecare materie, se poate observa imediat faptul ca aceasta este asociata unui cadrul didactic printr-o relatie de asociere de tipul unu-la-mai-multi, fiecare cadrul didactic putand sa sustina cursuri pentru niciuna, una sau mai multe materii. Materia reprezinta in acelasi timp si o parte componenta a planului de invatamant, acesta din urma fiind asociat printr-o relatie de asociere de tip unu-la-unu de clasele serie si an, ceea ce ilustreaza faptul ca pentru fiecare serie si fiecare an exista un plan de invatamant special definit.

La randul sau, fiecare serie are definita o unica specializare si reprezinta parte componenta a unui an, fiind la randul sau alcatauita din mai multe grupe. Aceasta caracteristica este precisata cu ajutorul relatiile de tip asociere unu-la-unu si de tip compositie de multiplicitate unu-la-mai-multi.

Se precizeaza, de asemenea, si pozitia studentului in cadrul mare al sistemului, acesta din urma avand asociata o singura grupa si un singur an, fapt ce sustine imposibilitatea studentului de a invata in mai multi ani si mai multe grupe in acelasi timp.

In final, este evidentiat grafic prin intermediul unei relatii de compozitie, faptul ca o facultate este compusa din mai multi ani de studiu.

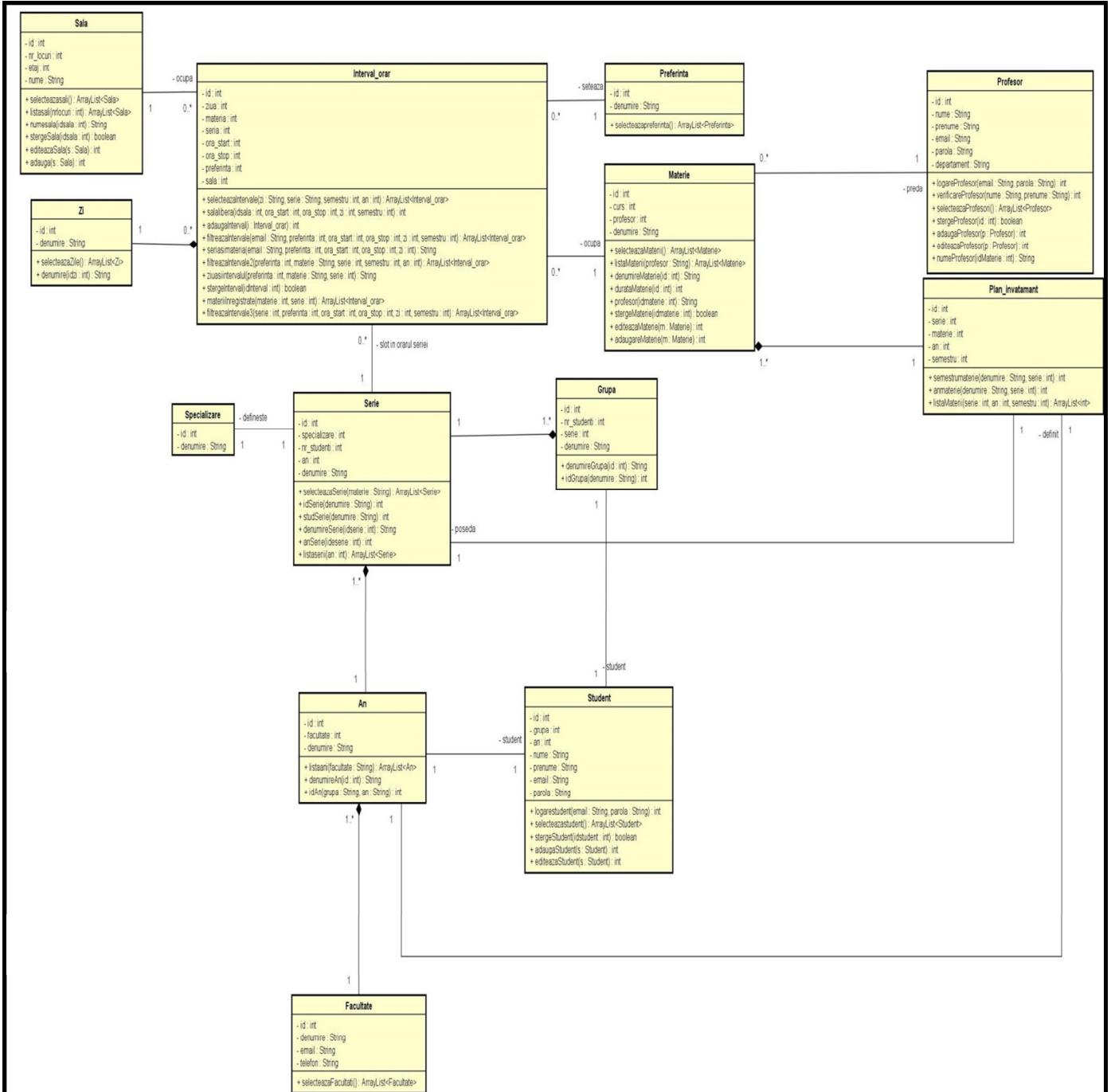


Figura 14. Diagrama de clasă a sistemului

5.1.1 Surse de informare

Aplicatia ce face subiectul lucrarii de fata este menita, asa cum a fost specificat si in capitolele anterioare sa sa imbunatateasca procesul de creare al orarelor prin modelarea acestora in concordanta cu preferintele cadrelor didactice, dar si prin generarea acestora in mod automat. Aplicatia ofera, de asemenea, posibilitatea gestionarii informatiilor importante folosite in acest proces, precum date despre cadrele didactice si alocarea acestora pentru fiecare materie, date despre studentii facultatii, dar si date despre materiile ce alcataiesc un anume plan de invatamant si salile de studiu disponibile ale facultatii. Aceasta facilitate este dedicata doar persoanelor cu rol de administrator, evitand astfel posibilitatea ca persoanele neautorizate sa altereze date insemnante din punct de vedere al functionarii corecte a sistemului.

Dezvoltarea aplicatiei de modelare si generare automata a orarelor in cadrul unei facultati a fost realizata cu ajutorul mediului de programare Eclipse, versiunea Luna, una dintre cele mai stabile versiuni gratuite ce pot fi gasite pe internet. Pentru ca este un software ce necesita stocare si manipulare de date intr-o masura semnificativa, s-a folosit in realizarea aplicatiei serverul MS SQL Server si limbajul SQL specific acestuia in vederea gestionarii bazei de date a aplicatiei. Cele doua mari componente ale aplicatiei, partea de programare in Eclipse si partea de gestionare a bazei de date in MS SQL Server, comunica intre ele prin intermediul unui JDBC, numit jdbc:sqlserver://LAPTOP-H4AJ3TNV\\MSSQLSERVER:1433;databaseName=SmartCampus. In construirea bazei de date s-a avut in permanenta in vedere respectarea modelului obtinut in etapa de analiza si specificare a cerintelor (vezi Figura 14.), dar si faptul ca utilizatorii carora aceasta se adreseaza sunt atat studentii, cat si cadrele didactice, acestia din urma avand un rol insemnat in introducerea de informatii pe baza carora mai tarziu se vor genera orarele. Astfel, a aparut si necesitatea aparitiei unui al treilea rol, acela fiind cel de administrator care sa poata gestiona datele din baza si sa poata crea orarele pe baza acestora.

Un mod simplu de a descrie cum poate fi folosita aplicatia intr-o maniera simpla si utila este dat de respectarea urmatorilor pasi:

- Administratorul aplicatiei introduce la fiecare inceput de an universitar date despre cadrele didactice active, despre materiile din planul de invatamant al fiecarei serii, dar si date cu privire la alocarea fiecarui cadru didactic pentru una sau mai multe materii.
- Administratorul actualizeaza la fiecare inceput de an universitar datele despre salile in care se pot tine ore de curs in cadrul facultatii.
- Administratorul introduce la fiecare inceput de an universitar date referitoare la studentii facultati ce pot vizualiza orarele in anul universitar curent.
- Se seteaza o perioada in care cadrele didactice sa poata introduce preferintele legate de anumite intervale orare pe care doresc sa le fixeze pentru diferite materii in cadrul orarului unei anume serii.
- Administratorul aplicatiei genereaza orarele pentru fiecare serie si le face publice pentru a putea fi accesate de toti ceilalti utilizatori ai aplicatiei.

Un alt aspect important in realizarea aplicatiei a fost reprezentat de crearea de interfete grafice care sa fie atat placute, cat si intuitive din punct de vedere al utilizatorilor finali ai software-ului. Modul in care toate aceste elemente ale sistemului au fost implementate va fi prezentat in continuare, urmarind ordinea cronologica a etapelor.

5.1.1.1 Structurarea bazei de date

In figura urmatoare (vezi Figura 15.) a fost creata o diagrama a bazei de date cu ajutorul Microsoft SQL Server Management Studio in care au fost evidențiate toate tabelele ce compun baza de date, dar și relațiile dintre acestea. Baza de date denumita SmartCampus este alcătuită asa cum se poate observa din 14 tabele diferite și o vedere, acestea fiind construite respectand principiul integrității datelor. In Anexa 2, a fost introdus un paragraf dedicat creării bazei de date in care poate fi studiat un exemplu de creare a unui tabel.

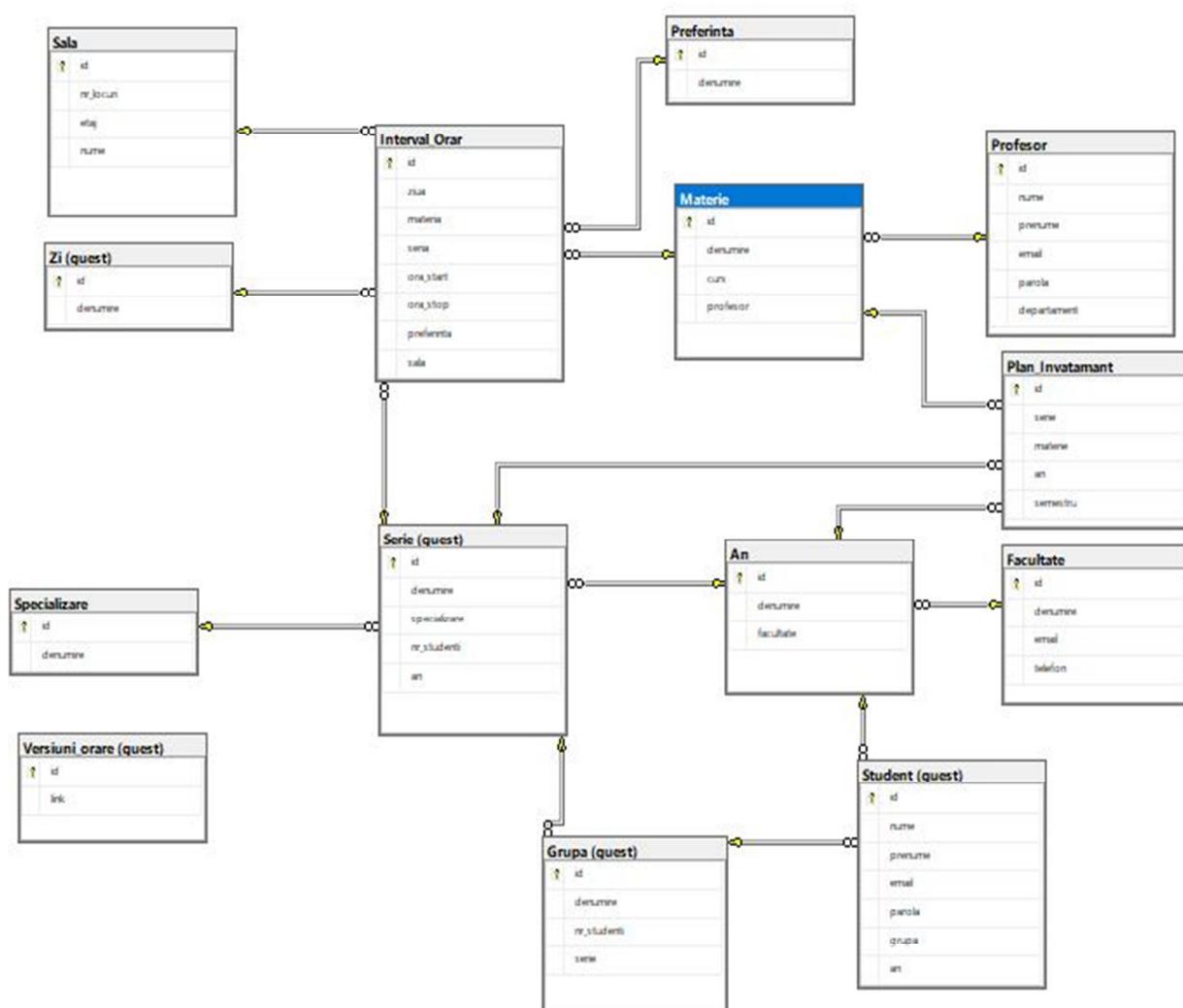


Figura 15. Structura bazei de date

Fiecare tabel ilustrat in figura anterioara (vezi Figura 15.) indeplineste un rol bine definit din punct de vedere al stocarii informatiilor. Distingem astfel urmatoarele:

- Sala: tabel cu rol in gestionarea salilor de studiu ca spatiu efectiv al orelor de curs
- Zi: tabel cu rol nomenclator in gestionarea zilelor saptamanii in cadrul orarului fiecarei serii
- Interval_orar: tabel cu rol in gestionarea intervalelor orare ocupate in cadrul orarului fiecarei serii
- Serie: tabel care se ocupa cu gestionarea seriilor, ca grupuri de studenti, din cadrul unei facultati
- Grupa: tabel care se ocupa, de asemenea cu gestionarea grupurilor de dimensiuni reduse de studenti
- Student: tabel cu rol in gestionarea studentilor ce sunt inscrisi in cadrul facultatii
- An: tabel cu rol in gestionarea anilor de studiu din cadrul unei facultati
- Facultate: tabel ce se ocupa cu gestionarea facultatilor din cadrul unei universitatii
- Plan_Invatamant: tabel ce se ocupa cu gestionarea planurilor de invatamant pentru fiecare serie din cadrul unei facultati
- Profesor: tabel cu rol in gestionarea cadrelor didactice din cadrul unei facultati
- Materie: tabel cu rol in gestionarea materiilor ce alcatuiesc planurile de invatamant
- Preferinta: tabel cu rol de nomenclator in definirea preferintelor in legatura cu un interval orar
- Specializare: tabel cu rol de gestionare a specializarilor din cadrul unei facultati
- Versiuni_orare: tabel ce se ocupa cu gestionarea versiunilor de orare publicate

5.1.1.2 Prezentarea tabelelor

In continuare vor fi prezentate toate tabelele bazei de date SmartCampus din punct de vedere structural, dar si al tipurilor de date pe care acestea le stocheaza. In procesul de proiectare a bazei de date s-a definit pentru fiecare tabel o coloana denumita „id”, ce are rol de identificator unic pentru fiecare inregistrare a tabelului, reprezentand de fapt „cheia primara a tabelului”. Aceasta coloana a fost definita ca fiind autoincrementabila, putand lua valori incepand de la 1 si continuand cu un adaos de 1. Aceasta impreuna cu declararea sirurilor de caractere ca VARCHAR de dimensiune variabila reprezinta caracteristicile comune ale tuturor tabelelor ce alcatuiesc baza de date a aplicatiei asociate prezentei lucrari.

Tabelul An

In figura urmatoare (vezi Figura 16.) se poate observa structura tabelului denumit An, ce este alcătuit din 3 coloane:

- Id: cu rol de „cheie primara” a tabelului;
- Denumire: cu rol de specificare a numelui fiecarui an;
- Facultate: cu rol de „cheie strina” către tabelul Facultate, reprezentand coloana prin care se realizează referirea tabelului Facultate în cadrul tabelului An.

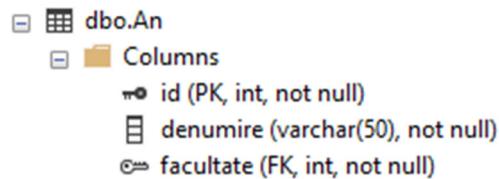


Figura 16. Tabelul An

Tabelul Facultate

Imaginea urmatoare (vezi Figura 17.) ilustrează structura tabelului Facultate, alcătuit din 4 coloane:

- Id: cu rol de „cheie primara” a tabelului;
- Denumire: cu rol de specificare a numelui fiecarei facultăți din cadrul universității;
- Email: cu rol de specificare a email-ului facultății;
- Telefon: cu rol de specificare a numărului de telefon al facultății.

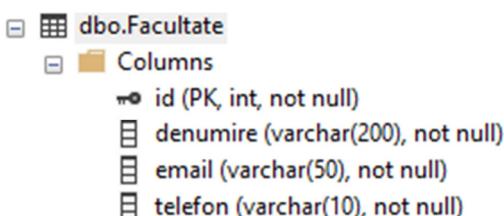


Figura 17. Tabelul Facultate

Tabelul Interval_orar

In imaginea urmatoare (vezi Figura 18.) se poate observa structura tabelului Interval_orar, ce reprezintă tabelul care stocăzează cele mai importante date ale aplicației și care este alcătuit din 8 coloane:

- Id: cu rol de „cheie primara” a tabelului;
- Ziua: cu rol de „cheie strina” către tabelul Zi, reprezentand coloana prin care se realizează referirea tabelului Zi în cadrul tabelului Interval_orar;
- Materia: cu rol de „cheie strina” către tabelul Materie, reprezentand coloana prin care se realizează referirea tabelului Materie în cadrul tabelului Interval_orar;
- Seria: cu rol de „cheie strina” către tabelul Serie, reprezentand coloana prin care se realizează referirea tabelului Serie în cadrul tabelului Interval_orar;
- Ora_start: cu rol de specificare a orei de început pentru un interval orar;
- Ora_stop: cu rol de specificare a orei de sfârșit a unui interval orar;

- Preferinta: cu rol de „cheie straina” catre tabelul Preferinta, reprezentand coloana prin care se realizeaza referirea tabelului Preferinta in cadrul tabelului Interval_orar;
- Sala: cu rol de „cheie straina” catre tabelul Sala, reprezentand coloana prin care se realizeaza referirea tabelului Sala in cadrul tabelului Interval_orar.

dbo.Interval_Orar	
Columns	
id	(PK, int, not null)
ziua	(FK, int, not null)
materia	(FK, int, not null)
seria	(FK, int, not null)
ora_start	(int, not null)
ora_stop	(int, not null)
preferinta	(FK, int, not null)
sala	(FK, int, not null)

Figura 18. Tabelul Interval_orar

Tabelul Materie

In figura ce urmeaza (vezi Figura 19.) poate fi observata structura tabelului Materie, ce este alcătuit din 4 coloane:

- Id: cu rol de „cheie primara” a tabelului;
- Denumire: cu rol de specificare a denumirii fiecarei materii;
- Curs: cu rol de specificare a numarului orelor de curs ale materiei;
- Profesor: cu rol de „cheie straina” catre tabelul Profesor, reprezentand coloana prin care se realizeaza referirea tabelului Profesor in cadrul tabelului Materie.

dbo.Materie	
Columns	
id	(PK, int, not null)
denumire	(varchar(200), not null)
curs	(int, not null)
profesor	(FK, int, not null)

Figura 19. Tabelul Materie

Tabelul Plan_invatamant

In imaginea de mai jos (vezi Figura 20.) este ilustrata structura tabelului Plan_Invatamant, ce contine 5 coloane:

- Id: cu rol de „cheie primara” a tabelului;
- Serie: cu rol de „cheie straina” catre tabelul Serie, reprezentand coloana prin care se realizeaza referirea tabelului Serie in cadrul tabelului Plan_Invatamant
- Materie: cu rol de „cheie straina” catre tabelul Materie, reprezentand coloana prin care se realizeaza referirea tabelului Materie in cadrul tabelului Plan_Invatamant
- An: cu rol de „cheie straina” catre tabelul An, reprezentand coloana prin care se realizeaza referirea tabelului An in cadrul tabelului Plan_Invatamant
- Semestru: cu rol in specificarea semestrului in care seria are ore de curs pentru o anume materie.

dbo.Plan_Invatamant	
Columns	
id	(PK, int, not null)
serie	(FK, int, not null)
materie	(FK, int, not null)
an	(FK, int, not null)
semestru	(int, not null)

Figura 20. Tabelul Plan_Invatamant

Tabelul Preferinta

Imaginea urmatoare (vezi Figura 21.) defineste structura tabelului Preferinta, ce a fost construit in cadrul aplicatiei cu rol de nomenclator, fiind alcătuit din numai 2 coloane:

- Id: cu rol de „cheie primara” a tabelului;
 - Denumire: cu rol in exprimarea gradului de referinta
- Astfel, tabelul Preferinta contine numai 3 inregistrari posibile: „Favorit”, „Indiferent”, „Nedorit”.

dbo.Preferinta	
Columns	
id	(PK, int, not null)
denumire	(varchar(50), not null)

Figura 21. Tabelul Preferinta

Tabelul Profesor

In figura urmatoare (vezi Figura 22.) se poate observa structura tabelului Profesor, alcătuit din 6 coloane:

- Id: cu rol de „cheie primara” a tabelului;
- Nume: cu rol in specificarea numelui cadrului didactic;
- Prenume: cu rol in specificarea prenumelui cadrului didactic;
- Email: cu rol in specificarea email-ului cadrului didactic;
- Parola: cu rol in specificarea parolei asociata cadrului didactic pentru accesul in aplicatie;
- Departament: cu rol in specificarea departamentului din care face parte cadrul didactic in cadrul facultatii.

dbo.Profesor	
Columns	
id	(PK, int, not null)
nume	(varchar(50), not null)
prenume	(varchar(50), not null)
email	(varchar(200), not null)
parola	(varchar(200), not null)
departament	(varchar(50), not null)

Figura 22. Tabelul Profesor

Tabelul Sala

Structura tabelului Sala este ilustrata in figura ce urmeaza (vezi Figura 23.) si se poate observa faptul ca este alcătuită din 4 coloane:

- Id: cu rol de „cheie primară” a tabelului;
- Nr_locuri: cu rol în specificarea capacitatii salii;
- Etaj: cu rol în specificarea etajului la care se gaseste sala;
- Nume: cu rol în specificarea denumirii salii.

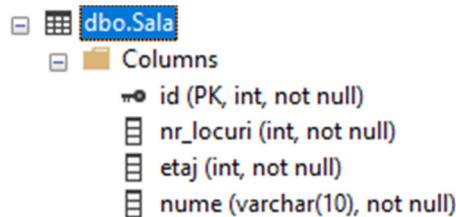


Figura 23. Tabelul Sala

Tabelul Specializare

In figura urmatoare (vezi Figura 24.) este ilustrata structura tabelului Specializare, definit, de asemenea, ca nomenclator si fiind alcătuit din 2 coloane:

- Id: cu rol de „cheie primară” a tabelului;
- Denumire: cu rol în specificarea denumirii unei specializari.

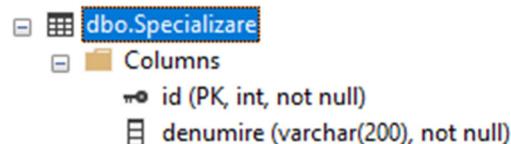


Figura 24. Tabelul Specializare

Tabelul Grupa

Figura urmatoare (vezi Figura 25.) ilustreaza structura tabelului Grupa, alcătuit din 4 coloane:

- Id: cu rol de „cheie primară” a tabelului;
- Denumire: cu rol în specificarea numelui fiecarei grupe;
- Nr_studenti: cu rol în specificarea numărului de studenți din cadrul unei grupe;
- Serie: cu rol de „cheie străină” către tabelul Serie, reprezentând coloana prin care se realizează referirea tabelului Serie în cadrul tabelului Grupa.

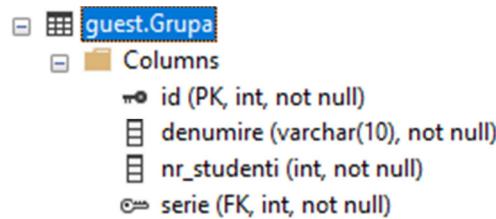


Figura 25. Tabelul Grupa

Tabelul Serie

Tabelul Serie este descris din punct de vedere structural in figura urmatoare (vezi Figura 26.), in care se poate observa faptul ca acesta este alcătuit din 5 coloane:

- Id: cu rol de „cheie primara” a tabelului;
- Denumire: cu rol in specificarea numelui seriei;
- Specializare: cu rol de „cheie strina” catre tabelul Specializare, reprezentand coloana prin care se realizeaza referirea tabelului Specializare in cadrul tabelului Serie;
- Nr_studenti: cu rol in specificarea numarului de studenti din cadrul unei serii;
- An: cu rol de „cheie strina” catre tabelul An, reprezentand coloana prin care se realizeaza referirea tabelului An in cadrul tabelului Serie;

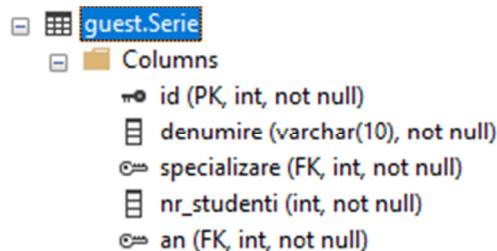


Figura 26. Tabelul Serie

Tabelul Student

In figura urmatoare (vezi Figura 27.) este ilustrata structura tabelului Student, ce este alcătuit din 7 coloane:

- Id: cu rol de „cheie primara” a tabelului;
- Nume: cu rol in specificarea numelui studentului;
- Prenume: cu rol in specificarea prenumelui studentului;
- Email: cu rol in specificarea email-ului studentului;
- Parola: cu rol in specificarea parolei asociata studentului pentru accesul in aplicatie;
- Grupa: cu rol de „cheie strina” catre tabelul Grupa, reprezentand coloana prin care se realizeaza referirea tabelului Grupa in cadrul tabelului Student;
- An: cu rol de „cheie strina” catre tabelul An, reprezentand coloana prin care se realizeaza referirea tabelului An in cadrul tabelului Student.

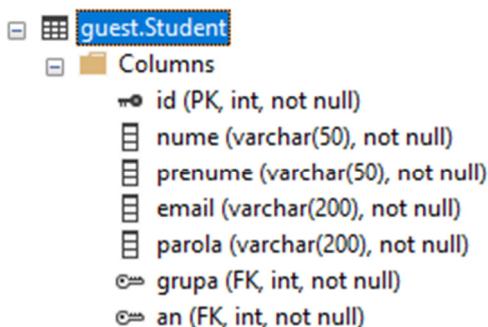


Figura 27. Tabelul Student

Tabelul Versiuni_orare

Figura urmatoare (vezi Figura 28.) ilustreaza structura tabelului Versiuni_orare, alcătuit din 2 coloane:

- Id: cu rol de „cheie primara” a tabelului;
- Link: cu rol în specificarea adresei web la care se pot vizualiza orarele generate.

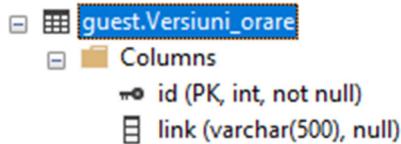


Figura 28. Tabelul Versiune_orare

Tabelul Zi

In imaginea urmatoare (vezi Figura 29.) se poate observa structura tabelului Zi, ce este definit ca nomenclator in cadrul aplicatiei si este alcătuit din 2 coloane:

- Id: cu rol de „cheie primara” a tabelului;
- Denumire: cu rol in specificarea denumirii fiecarei zile.

Tabelul zi poate contine doar 5 inregistrari si anume „Luni”, „Marti”, „Miercuri”, „Joi” si „Vineri”.

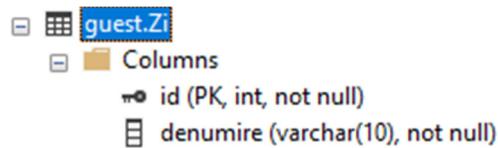


Figura 29. Tabelul Zi

Pentru a putea gestiona alocările cadrelor didactice pentru fiecare materie a fost construită în baza de date o vedere a carei structură este ilustrată în imaginea de mai jos (vezi Figura 30.). Așa cum se poate observa în aceasta vedere sunt stocate atât „cheile primare” ale tabelelor Profesor și Materie, cât și numele, prenumele, email-ul și departamentul cadrului didactic, dar și denumirea și durata orelor de curs pentru o materie. Scriptul prin care s-a creat aceasta vedere poate fi studiat în Anexa 2, paragraful dedicat creării de vederi.

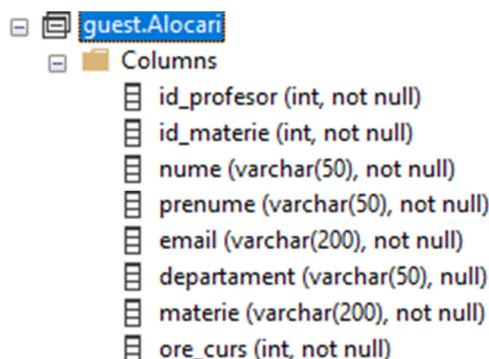


Figura 30. View Alocari

5.1.1.3 Dezvoltarea aplicatiei Java

In ceea ce priveste dezvoltarea aplicatiei din punct de vedere al utilizarii limbajului Java s-au construit trei pachete distincte cu sarcini si roluri specifice in cadrul rularii si functionarii corecte a aplicatiei.

Pachetul Main

Acest pachet a fost creat scopul de a construi toate clasele aplicatiei, impreuna cu metodele lor specifice, dar si cu scopul de a realiza comunicarea cu baza de date.

Se poate observa in sevenita de cod ilustrata mai jos modul in care a fost construita, spre exemplu clasa Interval_orar. Pentru a putea crea un constructor pentru o clasa anume trebuie avuta in vedere definirea clasei ca fiind publica, iar denumirea constructorului trebuie sa fie identica cu cea a clasei la care acesta se refera.

S-a folosit exemplul de constructor cu parametrii, acestia reprezentand de fapt coloanele tabelului Interval_orar din baza de date. Referirea catre variabilele definite la nivelul clasei se face intotdeauna prin intermediul sintagmei this.

```
package Main;

public class Interval_orar {

    private int id, ziua, materia, seria, ora_start, ora_stop, preferinta,
    sala;

    public Interval_orar(int id, int ziua, int materia, int seria,
                         int ora_start, int ora_stop, int preferinta, int sala) {
        this.id = id;
        this.ziua = ziua;
        this.materia = materia;
        this.seria = seria;
        this.ora_start = ora_start;
        this.ora_stop = ora_stop;
        this.preferinta = preferinta;
        this.sala = sala;
    }
}
```

O alta modalitate de crearea a unui constructor este aceea a constructorului fara parametrii descris in sevenita urmatoare de cod, acesta fiind extrem de util in cazul in care se doreste la un moment dat in aplicatie apelarea construirii unei clase fara a mai specifica si parametrii acestora. Fara acest tip de constructor aplicatia ar genera erori atunci cand acesta se apeleaza, dar nu este definit.

```
public Interval_orar() {
}
```

S-au precizat la nivelul fiecarei clase si metode de definire si preluare a valorilor variabilelor specifice clasei, ce ajuta la introducerea datelor in baza de date, dar si la interpretarea rezultatelor interogarilor facute catre baza de date. Aceste metode sunt metode de tip get() si set(), un exemplu de realizare al acestora fiind cel ilustrat de urmatoare sevenita de cod in care variabila de interes este id-ul unui interval orar.

```

public int getId() {
    return id;
}

public void setId(int id) {
    this.id = id;
}

```

Conform acestei structuri au fost definite toate clasele aplicatiei, ce reprezinta de fapt tabelele ce se regasesc in baza de date, ale caror informatii vor fi manipulate mai departe cu ajutorul interfefelor grafice.

Pachetul Main contine asa cum am amintit si anterior pe langa clasele prin care se defineste fiecare tabel din baza de date si clase care realizeaza conexiunile cu baza de date, in care sunt scrise interogari catre aceasta.

Fiecare dintre clasele dedicate comunicarii cu baza de date includ un driver special prin intermediul caruia se realizeaza conexiunea si anume driver-ul com.microsoft.sqlserver.jdbc. Pentru a realiza comunicarea cu baza de date intr-un mod corect toate clasele care doresc sa acceseze baza de date trebuie sa contina urmatoarea sevenita de cod.

```

public class Interval_orarDB {

    private final static String DB_CONNECTION_URL =
Metode.Constante.dbConnectionString;
    private final static String DRIVER_CLASS_NAME =
Metode.Constante.driverClassName;
    private final static String USER = Metode.Constante.user;
    private static final String PASSWORD = Metode.Constante.password;

    private static Connection databaseConnection = null;

    static {
        try {
            Class.forName(DRIVER_CLASS_NAME).newInstance();
        } catch (Exception ex) {
            ex.printStackTrace();
        }

        try {
            databaseConnection = DriverManager.getConnection(
                DB_CONNECTION_URL, USER, PASSWORD);
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}

```

In sevenita anterioara se observa faptul ca prima linie specifica calea catre baza de date, a doua reprezinta driverul folosit in vederea realizarii conexiunii, in timp ce prin cea de-a treia si cea de-a patra linie sunt specificate credentialele de autentificare in baza de date, mai exact numele de utilizator si parola asociata acestuia.

Urmatoarele linii ale sevenitei sunt reprezentate de doua blocuri de tip try-catch prin intermediul carora se incearca deschiderea unei sesiuni si testarea conexiunii cu baza de date.

Blocurile de tip catch sunt extrem de utile in cazul in care nu se poate stabili conexiunea cu baza de date si apar erori, acestea fiind redate catre utilizator prin intermediul consolei.

Pentru ca numarul de clase ce realizeaza astfel de conexiuni este egal cu numarul de tabele ce se gasesc in baza de date, modul in care a fost gandita aplicatia fiind acela de a crea cate o clasa pentru manipularea fiecarui tabel, posibilitatea introducerii unor date de comunicare gresite a fost luata in considerare si eliminata prin crearea pachetului Metode si a clasei Constante in interiorul acestuia, aceasta abordare conducand la declararea si initializarea datelor de conectare cu baza de date o singura data.

In interiorul claselor ce realizeaza conexiunea cu baza de date se introduc si obiecte care contin sechete de tip SQL, acestea fiind folositoare in vederea introducerii, stergerii sau actualizarii informatiilor din baza de date si usurand in acelasi timp si procesul de dezvoltare deoarece se interogheaza baza de date intr-un mod rapid prin apelarea metodelor in cadrul carora obiectele au fost definite. Metodele despre care vorbim trebuie denumite intr-un mod cat mai intuitiv, care sa ofere informatii despre rezultatele ce se obtin in urma interogarii.

In continuare este prezentata o secheta de cod in care se doreste sa se obtina orarul unei anumite serii in functie de anul, ziua si semestrul selectat. Metoda denumita selecteazaIntervale intoarce un rezultat de tipul lista de intervale orare, avand ca parametrii de intrare ziua, seria, semestrul si anul pentru care se doreste sa fie aduse informatiile din baza de date. In interiorul metodei se observa transpunerea rezultatului intors de instructiunea sql din tabel de valori in lista de elemente de tip interval orar, aceasta fiind extinsa folositoare in construirea interfetei grafice in care utilizatorul doreste sa manipuleze datele si sa vizualizeze rezultatele actiunilor sale in timp real.

```
//Afiseaza orarul seriei in functie de semestru materiei selectate si anul in care seria face materia
    public static List<Interval_orar> selecteazaIntervale(String zi, String serie, int semestru, int an){
        List<Interval_orar> lista = null;
        String sql = "select Interval_Orar.* from Plan_Invatamant p,
interval_orar inner join zi on zi.id = interval_orar.ziua "
                    + "inner join serie on serie.id = interval_orar.seria
where zi.denumire = ? and serie.denumire = ?"
                    + " and p.materie = Interval_Orar.materia and
p.semestru = ? and interval_orar.seria = p.serie and p.an = ?";
        try{
            PreparedStatement selectStatement =
databaseConnection.prepareStatement(sql);
            selectStatement.setString(1, zi);
            selectStatement.setString(2, serie);
            selectStatement.setInt(3, semestru);
            selectStatement.setInt(4, an);
            ResultSet results = selectStatement.executeQuery();
            lista = new ArrayList<Interval_orar>();

            while (results.next()) {
                Interval_orar i = new Interval_orar();
                i.setId(results.getInt(1));
                i.setZiua(results.getInt(2));
                i.setMaterie(results.getInt(3));
                i.setSeria(results.getInt(4));
                i.setOra_start(results.getInt(5));
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

```

        i.setOra_stop(results.getInt(6));
        i.setPreferinta(results.getInt(7));
        i.setSala(results.getInt(8));
        lista.add(i);
    }

} catch (SQLException e) {
    e.printStackTrace();
}

return lista;
}

```

Un alt tip de metode extrem de importante din punct de vedere al manipularii datelor cu ajutorul interfetei grafice sunt metodele de adaugare, actualizare sau stergere a informatiilor din baza de date. O metoda prin intermediul careia utilizatorul poate introduce date in baza de date este prezentata in secventa urmatoare. Metoda adaugaInterval intoarce un rezultat de tip int si are ca parametru de intrare o variabila de tip interval orar. Se observa prezenta secventei SQL de tip INSERT prin intermediul careia utilizatorul poate introduce in baza de date o preferinta cu privire la un anumit interval orar. In blocul try se pregateste conexiunea cu baza de date, urmand ca mai apoi datele ce sunt introduse de catre utilizator in interfata grafica sa fie preluate si inserate in baza de date cu ajutorul metodei insertStatement. Rezultatul de tip integer al metodei este extrem de util dezvoltatorului aplicatiei atunci cand acesta doreste sa verifice marja de succes a metodei.

```

//Adauga interval
public static int adaugaInterval(Interval_orar i){
    int rowsAffected = 0;

    if (databaseConnection != null) {
        String sql = "INSERT INTO interval_orar
(ziua,materia,seria,ora_start,ora_stop,preferinta,sala)
VALUES (?,?,?,?,?,?,?,?)";
        try {
            PreparedStatement insertStatement =
databaseConnection.prepareStatement(sql);

            insertStatement.setInt(1, i.getZiua());
            insertStatement.setInt(2, i.getMateria());
            insertStatement.setInt(3, i.getSeria());
            insertStatement.setInt(4, i.getOra_start());
            insertStatement.setInt(5, i.getOra_stop());
            insertStatement.setInt(6, i.getPreferinta());
            insertStatement.setInt(7, i.getSala());

            rowsAffected = insertStatement.executeUpdate();

        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
    return rowsAffected;
}

```

O alta clasa extrem de importanta in realizarea aplicatiei, ce este prezenta in acest pachet, este reprezentata de clasa in care se construiesc orarele in format tabelar XLS, urmand ca acestea sa fie salvate pe statia locala a administratorului. Un exemplu de orar in forma XLS se poate regasi in Anexa 3.

Pachetul Metode

Acest pachet a fost construit pentru a gazdui cea mai importanta clasa a aplicatiei din punct de vedere al realizarii conexiunii cu baza de date, dar si din punct de vedere al securitatii aplicatiei. Astfel, in acest pachet a fost definita clasa Constante, in care au fost definite numele de utilizator si parola cu ajutorul carora se realizeaza autentificarea in baza de date, dar si calea catre aceasta si numele clasei driver-ului folosit. Aceste patru variabile reprezinta constantele de comunicare cu baza de date, fiecare clasa definita in pachetul Main accesand date din tabele diferite.

Tot in clasa constanta, au fost definite si credentialele de autentificare pentru utilizatorul cu rol de administrator in aplicatie. Acest nivel de permisiuni are un rol vital in manipularea informatiilor ce se gasesc in baza de date, persoanele neautorizate fiind necesar sa nu poata accesa aplicatia din postura de administrator.

Aceasta separare a variabilelor cu valori constante pe parcursul rularii aplicatiei a avut ca scop usurarea procesului de intretinere a aplicatiei, in sensul ca, in cazul aparitiei unor schimbari ale sistemului de gestiune a bazelor de date sau ale credentialelor de autentificare, modificarea acestora sa se faca intr-un singur loc, fara a fi necesara verificarea tuturor claselor ce folosesc aceste variabile.

Pachetul GUI (Graphic User Interface)

Acest pachet a fost creat pentru a fi definite in interiorul sau diversele clase cu ajutorul carora s-au construit interfetele grafice pentru utilizator. In cadrul aplicatiei de fata au fost realizate o fereastra de autentificare, in care utilizatorul poate introduce credentialele de intrare in aplicatie, o fereastra dedicata solicitarii inregistrarii in aplicatie, in care utilizatorul specifica datele cu care doreste sa creeze un cont, urmand ca acestea sa fie trimise catre administratorul aplicatiei, dar si trei ferestre principale diferite pentru fiecare tip de utilizator definit si anume: administrator, cadru didactic si student.

Ca si caracteristica comună doate ferestrele principale contin tab-uri pentru a facilita vizualizare informatiilor din baza de date intr-o maniera organizata. Pe langa acestea, fiecare fereastra mai poate contine meniuri pentru navigarea intre taburile disponibile, panouri in care se pot organiza si fixa in spatiu elementele din care este formata fereastra, campuri in care pot fi introduse datele ce se doresc a fi inregistrate in baza de date, butoane denumite intuitiv pentru ca utilizatorul sa poata executa operatiile dorite, dar si tabele in care sunt afisate si informatiile ce se gasesc la momentul curent in baza de date si etichete ce indica utilizatorului lamuriri cu privire la datele vizualizate.

Aceste tipuri de clase au fost construite cu ajutorul seturilor de instrumente Swing si AWT puse la dispozitie de limbajul Java, cateva secvente de cod prin care au fost declarate si initializate componentele interfetelor grafice putand fi studiate intr-unul din capitolele anterioare si anume in capitolul 3.2.3.12, denumit Interfete grafice cu utilizatorul.

Rezultatele obtinute prin rularea aplicatiei, in spuma modul in care ferestrele construite sunt afisate utilizatorului pot fi vizualizate in capitolul 6.1, denumit Ghid de utilizare.

5.1.2 Cazuri de utilizare

In figura ce urmeaza (vezi Figura 31. Diagrama cazurilor de utilizare) este ilustrata diagrama cazurilor de utilizare ale sistemului de modelare si generare automata de orare in cadrul unei facultati. Acest tip de diagrama a fost construit pentru a intelege mai bine cum poate fi folosita aplicatia de diferite tipuri de utilizatori si rolul acestora in cadrul sistemului. In continuare se va prezenta in detaliu fiecare caz de utilizare ce poate fi observat grafic si in diagrama.

Este important de precizat faptul ca in diagrama au fost ilustrate cazurile de utilizare in cadrul caror administratorul aplicatiei doreste sa adauge, sa actualizeze sau sa eliminate inregistrari, insa termenul de inregistrare a fost ales ca termen generic, in realitate administratorul putand initia actiunile de introducere, modificare si stergere de informatii pentru mai multe tipuri de date precum date cu referire la cadrele didactice si studentii facultatii, dar si date legate de alocarile cadrelor didactice pe materii, de materii in sine si de salile in care se pot sustine ore de curs.

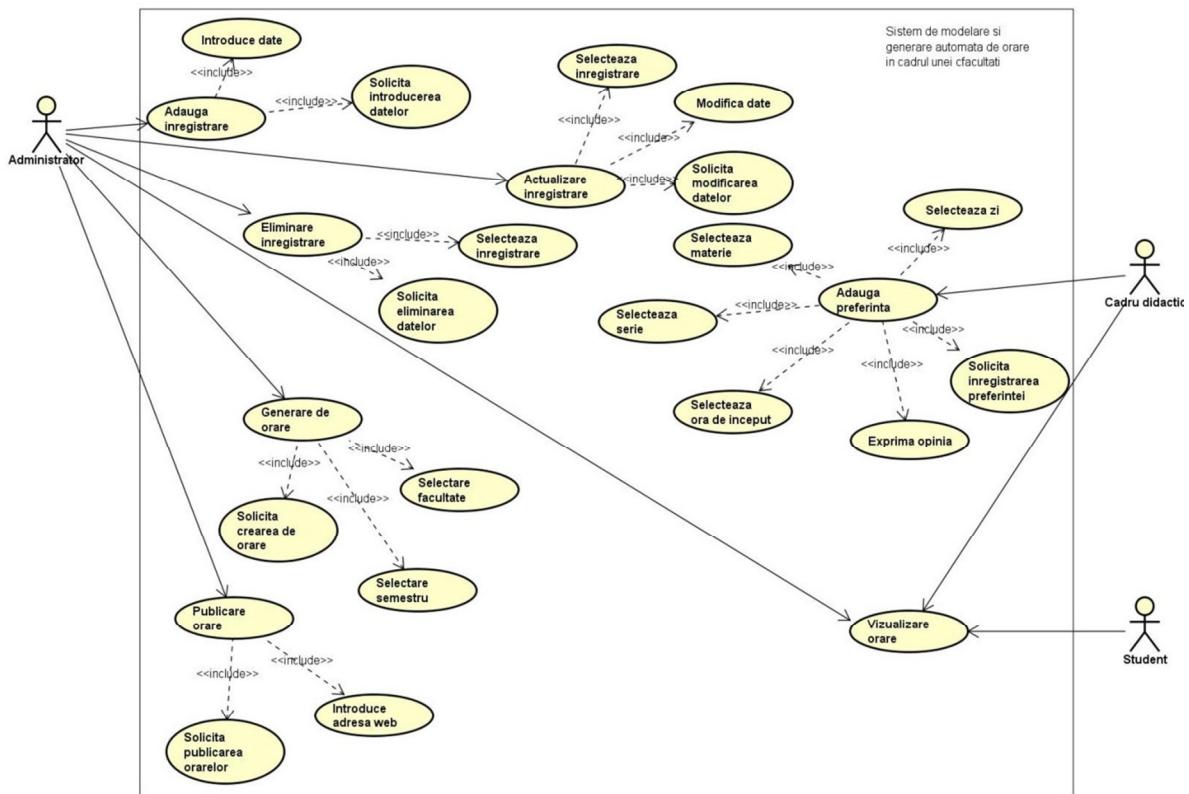


Figura 31. Diagrama cazurilor de utilizare

Adaugarea inregistrarilor

- Administratorul introduce in interfata grafica datele pe care doreste sa le inregistreze
- Administratorul solicita introducerea datelor in sistem.
- Sistemul preia datele din interfata grafica si incearca introducerea acestora in baza de date.
- Daca se intampina erori, sistemul afiseaza un mesaj de eroare catre utilizator. In cazul in care nu apar erori, datele sunt introduse in baza de date.

Actualizarea inregistrarilor

- Administratorul selecteaza inregistrarea pe care doreste sa o modifice.
- Administratorul actualizeaza in interfata grafica datele pe care doreste sa le modifice.
- Administratorul solicita actualizarea datelor.
- Sistemul preia datele din interfata grafica si incearca actualizarea acestora in baza de date.
- Daca se intampina erori, sistemul afiseaza un mesaj de eroare catre utilizator. In cazul in care nu apar erori, datele sunt modificate in baza de date.

Eliminarea inregistrarilor

- Administratorul selecteaza inregistrarea pe care doreste sa o elimine.
- Administratorul solicita stergerea datelor.
- Sistemul preia datele din interfata grafica si incearca eliminarea acestora in baza de date.
- Daca se intampina erori, sistemul afiseaza un mesaj de eroare catre utilizator. In cazul in care nu apar erori, datele sunt eliminate din baza de date.

Generarea orarelor

- Administratorul selecteaza facultatea pentru care doreste sa genereze orarele.
- Administratorul selecteaza semestrul pentru care doreste sa genereze orarele.
- Administratorul solicita generarea de orare.
- Sistemul preia datele introduse de catre administratorul aplicatiei si genereaza pe baza acestora orarele in format XLS, urmand ca mai apoi sa le salveze pe statia locala de pe care a fost rulata aplicatia.
- In cazul in care sunt intapinate erori se afiseaza un mesaj specific catre utilizator, iar in cazul in care generarea s-a incheiat cu succes se afiseaza un mesaj de confirmare catre utilizator.

Publicarea orarelor

- Administratorul introduce adresa web la care pot fi gasite orarele.
- Administratorul solicita publicarea orarelor.
- Sistemul inregistreaza in baza de date adresa web la care se poate gasi ultima versiune de orare.

Adaugarea de preferinte

- Cadrul didactic selecteaza ziua dorita.
- Cadrul didactic selecteaza materia ce doreste sa o introduca in orar.
- Cadrul didactic selecteaza seria pentru care doreste sa realizeze adaugarea preferintei.
- Cadrul didactic selecteaza ora la care doreste sa inceapa sustinerea orelor de curs.
- Cadrul didactic selecteaza opinia pe care o are in legatura cu intervalul ales.
- Sistemul preia informatiile introduse de cadrul didactic din interfata grafica si urmeaza algoritmul de modelare al orarului folosind aceste date.

- In cazul producerii de erori sistemul le comunica pe acestea catre utilizator, si asteapta reintroducerea datelor. Daca nu au aparut erori pe durata parcurgerii algoritmului de modelare, sistemul va afisa un mesaj de confirmare catre utilizator.

Vizualizarea orarelor

- Oricare utilizator al aplicatiei (administratorul / cadrul didactic / studentul) poate vizualiza orarele.
- Sistemul redirectioneaza automat in acest caz utilizatorul catre adresa web a ultimei versiuni de orare publicata.

5.1.3 Functionalitati

In acest subcapitol va fi prezentata aplicatia din punct de vedere al functionalitatilor pe care aceasta le pune la dispozitie publicului sau tinta. Fiecare proces ce completeaza software-ul creat in lucrarea de fata va fi descris in detaliu pentru o mai buna inteleghere a tuturor operatiilor pe care aplicatia poate sa le execute in vederea modelarii si generarii de orare in mod automat, dar si pentru a clarifica modul in care actiunile utilizatorului in cadrul aplicatiei impacteaza rezultatul final al acesteia.

Autentificarea in aplicatie

Pentru a putea face diferențierea in ceea ce priveste permisiunile pe care un utilizator le are in cadrul aplicatiei, sistemul pune la dispozitie o interfata grafica in care utilizatorul poate introduce credentialele specifice lui.

Odata cu lansarea in executie a aplicatiei, sistemul afiseaza fereastra de autentificare si asteapta introducerea datelor de intrare in aplicatie. Datele introduse de catre utilizator sunt verificate cu ajutorul bazei de date, iar daca acestea sunt validate ca fiind corecte se afiseaza fereastra specifica nivelului de permisiuni al utilizatorului autentificat. In cazul in care datele introduse de catre utilizator nu sunt valide sistemul afiseaza un mesaj de eroare explicit catre utilizator si asteapta corectarea informatiilor furnizate anterior.

Solicitarea inregistrarii in aplicatie

Daca un utilizator nu beneficiaza de date de autentificare in aplicatie, cu alte cuvinte nu are un cont creat pentru aplicatia de fata, acesta poate solicita inregistrarea in aplicatie prin apasarea butonului “Solicita inregistrare” din fereastra de autentificare.

Sistemul afiseaza mai departe o fereastra in care utilizatorul trebuie sa introduca datele cu care doreste sa creeze contul, iar dupa apasarea butonul “Trimite solicitare” aplicatia va trimite un email cu format predefinit in care va ingloba datele furnizate de catre utilizator. Emailul va fi trimis catre administratorul aplicatiei in mod implicit.

Gestionarea alocarilor / materiilor / cadrelor didactice / studentilor /salilor

Aceasta functionalitate este dedicata rolului de administrator, acesta beneficiind de o gama larga de permisiuni ce vizeaza introducerea, actualizarea si stergerea de informatii in si din baza de date a aplicatiei.

Cand utilizatorul se autentifica in aplicatie cu o serie de credentiale specifice rolului de administrator sistemul verifica veridicitatea acestora si afiseaza interfata grafica special constructa pentru acest rol, in care acesta are posibilitatea de gestionare a alocarilor cadrelor didactice pe materii, a materiilor incluse in planul de invatamant al unei serii, dar si a cadrelor didactice, a studentilor si a salilor de studiu din cadrul facultatii.

Gestionarea categoriilor de informatii precizate mai sus presupune posibilitatea introducerii unor date noi in sistem, modificarea datelor ce se gasesc deja in baza de date a aplicatiei, dar si eliminarea datelor redundante.

Astfel, in momentul in care fereastra dedicata administratorului este deschisa, aceasta este populata cu 5 taburi fiecare dintre acestea fiind util vizualizarii in timp real a informatiilor ce se gasesc in baza de date a aplicatiei. Navigarea intre taburi se poate face la nivel de meniuri prin selectarea optiunii “Vizualizeaza” din cadrul meniului specific tipului de date sau se poate face prin simpla schimbare a tabului ce se doreste a fi afisat.

Daca administratorul alege sa introduca informatii noi in sistem poate apasa butonul “Adauga” disponibil in fiecare tab al aplicatiei sau in meniurile dedicate fiecarui tip de date. Dupa introducerea datelor cerute de catre sistem si apasarea din nou a butonului “Adauga”, sistemul verifica ca datele introduse sa fie relevante si le introduce in sistem, afisandu-le imediat si in tabul de vizualizare. Daca datele introduse sunt identificate de catre sistem ca fiind irelevante, acesta va afisa un mesaj de eroare catre utilizator si asteapta introducerea unui set de date corect.

In ceea ce priveste, modificarea datelor din baza de date, administratorul poate alege o inregistrare din tabelul in care se pot vizualiza informatiile ce se afla in sistem la momentul current, iar odata cu apasarea butonului “Editeaza” se va afisa o fereastra cu datele selectate in care se asteapta efectuarea actualizarilor dorite. Dupa aceasta se apasa butonul “Editeaza” din nou, iar datele vor fi actualizate in sistem si pot fi vizualizate in timp real in tab-ul specific.

In cazul in care administratorul doreste sa stearga date din sistem, acesta trebuie sa selecteze o inregistrare din tabelul in care se pot vizualiza informatiile stocate in sistem si sa apese apoi butonul “Sterge”. Sistemul va deschide in continuare o fereastra in care se poate verifica inregistrarea ce se doreste a fi stearsa, administratorul confirmand executarea acestei actiuni prin apasarea din nou a butonului “Sterge”.

Generarea de orare

Aceasta functionalitate reprezinta principala operatie pentru care a fost dezvoltata aplicatia de fata. Este o functionalitate specifica, de asemenea, doar rolului de administrator, fiind in realitate reprezentata de procesul automatizat de generare al orarelor.

Pentru a putea genera orare administratorul trebuie sa acceseze meniul dedicat orarelor si sa aleaga optiunea “Genereaza orare”. Sistemul va pune la dispozitie o fereastra in care administratorul poate selecta facultatea si semestrul pentru care doreste sa realizeze generarea. Selectarea facultatii se poate face dintr-o lista de tip drop-down in care sunt afisate doar facultatile inregistrate in baza de date, iar selectarea semestrului se realizeaza prin selectarea uneia din cele doua optiuni “Semestrul 1” sau “Semestrul 2”, prezente de asemenea intr-o lista de tip drop-down. Dupa apasarea butonului “Genereaza orare” aplicatia va comunica cu baza de date in vederea extragerii informatiilor folositoare.

Tot prin intermediul bazei de date sistemul va verifica in continuare fiecare materie din planul de invatamant din punct de vedere al prezentei sale in orar. Daca se gaseste o materie din planul de invatamant al seriei lipsa in orarul acesteia, se va introduce in mod automat in primul

interval orar liber al seriei, alocandu-se implicit o sala potrivita din punct de vedere al numarului de locuri.

Mai departe sistemul creaza orarele intr-o forma tabelara XLS cu ajutorul API-ului Java responsabil cu citirea si scrierea de documente in format Microsoft Office, si anume API-ul numit Apache POI. Dupa crearea fiecarui orar, se creeaza un director special pentru fiecare facultate in parte, in care va fi salvat fiecare orar generat. Calea la care se creeaza directoarele este: C:\\Users\\Public\\Documents\\SmartCampus\\.

Publicarea orarelor

Aceasta functionalitate este, de asemenea, specifica rolului de administrator si a fost gandita pentru ca orarele generate sa poata fi vizualizate de toti utilizatorii aplicatiei. Astfel pentru o buna functionare a aplicatiei, dar si pentru a asigura o acuratete mare a datelor furnizate operatia de publicare a orarelor trebuie sa succeada operatia de generare a acestora.

Astfel, dupa generare orarelor, administratorul poate sa le publice prin alegerea din meniul dedicate orarelor si sa aleaga optiunea “Publica orare”. Sistemul va pune la dispozitie o fereastra in care se poate introduce o adresa web, spre exemplu catre un director partajat in care se gasesc orarele . Dupa ce butonul “Publica orare” este apasat, sistemul introduce in baza de date calea catre ultima versiune de orare publicate.

Modelarea orarelor

Aceasta functionalitate a fost dezvoltata special pentru cadrele didactice in vederea imbunatatirii modului in care intervalele orare sunt setate in cadrul procesului de creare a unui orar, fiind de fapt modul prin care cadrele didactice isi pot exprima opinia cu privire la fixarea unui anumit slot orar pentru sustinerea orelor de curs ale unei materii specifice.

In cazul de fata vorbim despre autentificarea unui utilizator cu rol de cadru didactic, urmata de deschiderea de catre a sistem a ferestrei dedicate acestui nivel de permisiuni. In aceasta fereastra sunt disponibile doua tab-uri, primul dedicat procesului de modelare al orarelor, iar cel de-al doilea dedicat vizualizarii orarelor.

Daca vorbim despre alegerea primul tab, vorbim despre situatia in care cadrul didactic alege sa fixeze intervale orare pentru materiile la care acesta sustine ore de curs. Aceasta modelare incepe cu selectarea zilei in care se doreste sa se fixeze intervalul. Aceasta selectare este posibila cu ajutorul unei liste de tip drop-down in care sunt prezente doar zilele lucratoare ale saptamanii.

O a doua selectie este reprezentata de selectarea materiei pe care cadrul didactic doreste sa o inregistreze in orarul seriei, aceasta realizandu-se, de asemenea, cu ajutorul unei liste de tip drop-down, ce este populata numai cu denumirile materiilor pentru care cadrul didactic este alocat.

Al treilea pas al procesului de modelare este reprezentat de selectarea seriei pentru care se doreste facuta inregistrarea, aceasta fiind facilitata prin intermediul unei liste in care se pot regasi doar seriile de student ce au in planul de invatamant materia ce se doreste a fi fixata.

Dupa completarea tuturor pasilor decriși mai sus se afiseaza in partea din dreapta a ferestrei un tabel in care pot fi vizualizate materiile fixate la momentul curent in orarul seriei alese, aceasta afisare fiind facuta in functie de ziua care a fost aleasa la prima selectie. Odata cu aparitia datelor extrase din baza de date a aplicatiei este pus la dispozitia cadrului didactic si un buton denumit “Adauga preferinta”. La apasarea butonului se va deschide o fereastra in care utilizatorului ii sunt afisate atat materia pentru care se face alocarea slotului de timp, cat si durata

acesteia. Mai departe cadrul didactic trebuie sa selecte ora la care doreste sa inceapa sustinerea cursului si opinia pe care acesta o are in raport cu intervalul ales. Selectarea orei de inceput se poate face dintr-o lista de tip drop-down in care sunt prezente toate orele zile cuprinse intre ora 8 si ora 18, in timp ce selectarea preferintei se poate face dintr-o lista de tip drop-down in care se gasesc trei optiuni “Favorit”, insemnand intervalul preferat, “Indiferent”, reprezentand intervalul fata de care cadrul didactic nu doreste sa isi exprime o opinie clara si “Nedorit”, intervalul in care cadrul didactic nu poate sustine ore de curs.

Procesul de modelare se incheie cu apasarea butonului “Adauga preferinta” si verificarea datelor ce se doresc a fi introduce cu cele deja prezente in baza de date. Daca sistemul va identifica conflicte in baza de date precum suprapuneri temporale sau spatiale, atunci va afisa un mesaj de eroare explicit catre utilizator. In caz contrar, in care adaugarea se incheie cu succes, sistemul v-a afisa un mesaj de confirmare, urmat de afisarea in orarul seriei a materiei selectate, intervalul orar ales, dar doar in cazul in care a fost setata preferinta “Favorit”.

Vizualizare orarelor

Vizualizarea orarelor reprezinta functionalitatea comună tuturor nivelor de permisiuni ale aplicatiei, fiind totodata si cea mai modesta dintre toate functionalitatatile aplicatiei ce face subiectul lucrarii de fata.

Atat administratorul aplicatiei, cat si cadrele didactice si studentii facultatii pot vizualiza orarele prin simpla apasare a link-ului disponibil in tabul “Vizualizare orare”, singurul tab disponibil daca vorbim despre utilizatorul de tip student. Dupa apasarea link-ului utilizatorul va fi directionat automat de catre sistem spre adresa web la care se gaseste directorul cu orarele pentru fiecare facultate si fiecare serie.

5.1.4 Restrictii si alte cerinte nefunctionale

Pentru a putea obtine o aplicatie robusta, a carei functionare sa satisfaca cerintele specificate a fost necesara formularea unor restrictii ce au ca scop evitarea intrarii sistemului in stare de eroare si rezolvarea conflictelor ce pot aparea de-a lungul rularii software-ului. Aplicatia ce face subiectul lucrarii de fata interactioneaza asa cum este firesc cu utilizatorii sai, avand astfel de a face cu factorul uman ce poate introduce intentionat sau nu date inconsistente care sa produca situatii conflictuale. Vor fi prezentate in acest subcapitol toate restrictiile intr-o ordine cronologica a impunerii lor.

Prima restrictie impusa a fost cea in care se specifica separarea functionalitatilor pe nivele de permisiuni. Au fost definite astfel 3 tipuri de utilizatori fiecare dintre acestea avand anumite operatii pe care le poate executa, cele 3 roluri sunt: rolul administrator, rolul cadrul didactic si rolul student. Pentru a putea implementa aplicatia astfel incat sa respecte aceasta restrictie a fost necesara crearea unei forme de autentificare ce impune la randul ei restrictii. Astfel, daca un utilizator nu introduce corect credentialele de autentificare acesta nu poate accesa aplicatia si functionalitatile dedicate rolului sau.

Administratorul reprezinta utilizatorul cu cele mai multe permisiuni in cadrul aplicatiei, in sensul ca acesta poate introduce, actualize si sterge informatii din baza de date, dar poate in acelasi timp si sa genereze si sa publice orare. Actiunea de manipulare a datelor cu privire la alocarile cadrelor didactice pe materii, dar si a informatiilor referitoare la cadrele didactice si a studentilor din cadrul facultatii a fost definita pentru acest rol deoarece aceste informatii sunt extrem de importante din punct de vedere al functionarii corecte a aplicatiei, persoanele neautorizate neavand permisiunea de a le altera. Generarea si publicarea de orare a fost dedicata

rolului de administrator din motive de evitare a suprapunerilor de informatii, cu alte cuvinte daca fiecare utilizator ar avea dreptul sa genereze si sa publice orare ar putea exista situatia in care in fiecare zi sa ia nastere multiple versiune de orare. Prin restrictionarea acestei actiuni se poate controla astfel versiunile de orare si aparitia lor catre ceilalti utilizatori ai aplicatiei.

Procesul de generare automata a orarelor include in sine o serie de restrictii definite in vederea respectarii principiului datelor consistente si complete pentru ca orarele fiecarei serii sa contin numai materiile pe care aceasta le are specificate in planul de invatamant. Astfel, in cadrul acestui proces trebuie verificata fiecare materie din planul de invatamant in orarul seriei, iar in cazul in care aceasta nu apare fixata trebuie introdusa in orar in mod automat. Introducerea se face in primul slot orar liber din programul seriei, insa si de data aceasta trebuie respectate restrictiile temporale si spatiale, in sensul ca trebuie verificata disponibilitatea salilor in acest interval orar, dar si disponibilitatea cadrelor didactice. In cazul in care cel putin una dintre aceste constrangeri nu este respectata se va calcula urmatorul interval orar liber si procesul va fi reluat.

In ceea ce priveste restrictiile impuse cadrelor didactice vorbim de fapt despre restrictiile algoritmului de modelare a orarelor, importante din punct de vedere al evitarii suprapunerilor de intervale orare fixate, dar si din punct de vedere al introducerii datelor intr-un mod corect si relevant. Astfel, cadrul didactic nu poate selecta decat una dintre zilele lucratoare ale saptamanii, in zilele de sambata si duminica neputand fi fixate ore de curs. In cea de-a doua etapa, cadrul didactic nu poate selecta decat fixarea uneia dintre materiile alocate lui, introducerea de date in numele unui alt cadru didactic fiind interzisa din punct de vedere logic, un om neputand exprima preferintele altuia. Dupa selectarea materiei, cadrul didactic trebuie sa selecteze seria pentru care doreste sa faca fixarea intervalului orar, fiind restrictionat si de aceasta de data prin afisarea doar a seriilor ce au in planul de invatamant materia care se doreste a fi introdusa in orar, evitandu-se, astfel posibilitatea introducerii materiilor ce nu sunt de interes pentru o serie in orarul acesteia.

Mai departe vorbim in acest proces de restrictii impuse la nivelul orei de inceput a slotului orar, in sensul ca niciun curs nu poate incepe mai devreme de ora 8 si nu se poate termina mai tarziu de ora 20. Preferintele pe care cadrul didactic le poate seta apartin listei: "Preferat", "Indiferent" sau "Nedorit", respectandu-se astfel o unitate din punct de vedere al interpretarii datelor introduse.

Sunt definite in continuare cateva restrictii ce au rol in evitarea suprapunerilor de intervale orare din punct de vedere al cadrelor didactice, dar si al seriei ca grup de studenti. Astfel, cadrul didactic nu poate seta acelasi interval orar ca fiind "Preferat", deoarece acest lucru al insemana ca doreste sa sustina cursuri pentru 2 materii diferite in acelasi timp. De asemenea, cadrul didactic nu poate seta un interval orar ca fiind "Preferat", daca acesta se suprapune cu un interval orar deja fixat in orarul seriei pentru ca acest fapt ar insemana ca o serie de studenti ca fie prezenta la doua cursuri diferite in acelasi timp.

S-a impus insa si o restrictie referitoare la faptul ca o serie de studenti nu poate avea setata in orar aceeasi materie de mai multe ori deoarece s-ar produce o supraincarcare a programei universitare, dar si o restrictie referitoare la salile de curs in care se pot sustine ore, in sensul ca acestea trebuie sa fie suficient de mari din punct de vedere al capacitatii astfel incat sa gazduiasca toti studentii ce alcatuiesc o anume serie.

Referitor la restrictiile impuse rolului de student vorbim aici doar de restrictionarea acestuia in aplicatie a tuturor operatiilor disponibile celorlalți utilizatori, singura actiune permisa a studentilor fiind aceea de a vizualiza orare si de a le descarca de la adresa web la care acestea se gasesc.

5.2 Elemente de proiectare

5.2.1 Arhitectura

Acesta capitol este destinat descrierii aplicatiei ce face subiectul lucrarii de fata din punct de vedere arhitectural. Pentru o intelegerere mai usoara a componentelor ce formeaza arhitectura aplicatiei a fost realizata schema ce este prezentata in figura urmatoare (vezi Figura 32.).

Asa cum se poate observa aplicatia de modelare si generare automata a orarelor ruleaza pe orice statie de lucru pe care este instalata si contine trei mari componente ce ajuta la functionarea ei corecta, extrem de importante si in ceea ce priveste interactiunea cu utilizatorii. Distingem aici pachetul „Interfata grafica SWING”, prin intermediul careia au fost create si personalizate toate interfetele grafice ale aplicatiei, dar si pachetele „Clase pentru returnarea datelor”, cu rol in extragerea datelor de interes pentru utilizator si „Algoritmi si acces la date”, cu rol in realizarea operatiilor ce tin de sistem si accesarea datelor importante, aceasta fiind de altfel si „creierul” software-ului dezvoltat in aceasta lucrare.

Pentru a putea sa indeplineasca functiile pentru care a fost creata, aplicatia comunica cu o serie de sisteme externe cum ar fi, baza de date, sistemul de fisiere, serverul pentru email si cloud-ul prin diverse interfete. Astfel pentru a putea comunica cu baza de date se folosesc clasele speciale ale interfetei JDBC, pusa la dispozitie de limbajul de programare Java, in timp ce pentru a putea citi si scrie fisiere din sistemul de fisiere se foloseste interfata IO a aceluiasi limbaj. Baza de date si sistemul de fisiere se poate vedea ca sunt gazduite de acelasi server, pe de alta parte, email-ul si cloud-ul fiind gazduite de servere diferite, iar comunicarea cu acestea facandu-se in primul caz prin protocoul SMTP (Simple Mail Transfer Protocol), iar in cel de-al doilea prin protocolul HTTPS (HyperText Transfer Protocol Secure).

Toate cele patru sisteme au fost necesare in dezvoltarea aplicatiei, fiecare fiind folosit in mod corespunzator, baza de date pentru a putea stoca si manipula informatiile importante, iar sistemul de fisiere pentru a putea scrie fisiere de tip XLS, care sa reprezinte in fapt orarele generate pentru fiecare serie. In ceea ce priveste email-ul, acesta este extrem de util atunci cand un utilizator doreste sa obtina un cont de inregistrare in aplicatie, solicitarea acestuia fiind transmisa pe emailul asociat contului de administrator. Nu in ultimul rand, cloud-ul acesta a fost definit pentru a putea publica la o adresa web orarele generate de catre aplicatie pentru ca acestea sa poata fi vizualizate de ceilalți utilizatori ai aplicatiei.

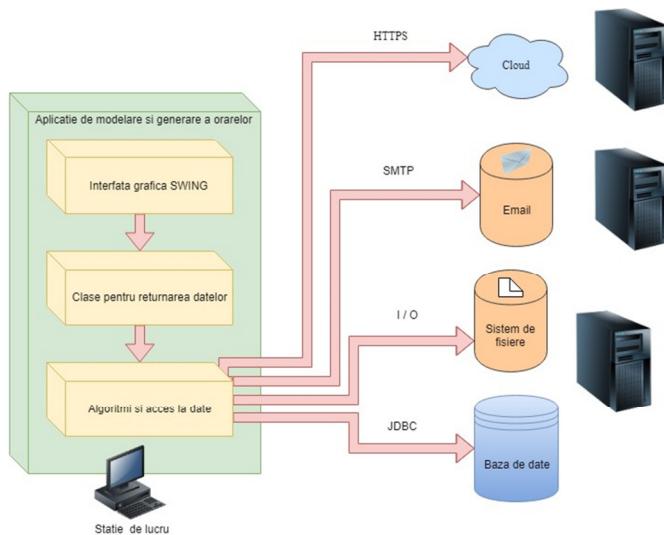


Figura 32. Structura arhitecturala

6 Descrierea aplicatiei practice

Aplicatia prezentata in aceasta lucrare este destinata realizarii orarului scolar din invatamantul univesitar. Bazata pe experienta mai multor ani de intocmire a orarului, folosind tehnologii noi de programare si rapiditatea limbajelor moderne, aplicatia de modelare si generare automata a orarelor este extrem de rapida si puternica, orarul a aproximativ 1200 ore de curs fiind generat in numai cateva secunde.

Prin intermediul acestui instrument este realizat orarul pe facultati, cadre didactice, semestre si serii. Totodata sunt oferite facilitati pentru a imparti un an universitar ca grup de studenti in serii si apoi in grupe de studiu.

Programul prezinta urmatoarele caracteristici importante:

- Generarea automata a orarului in functie de conditiile impuse pentru profesori, serii si sali;
- Posibilitatea de introducere manuala a anumitor ore/intervale orare si materii;
- Rapoartele orarului pot fi atat salvate in format XLS cat si exportate in vederea incarcarii acestora pe un disc de retea oferindu-se astfel acces online tuturor utilizatorilor aplicatiei.

6.1 Ghid de utilizare

Acet capitol este dedicat prezentarii modului in care aplicatia de modelare si generare automata a orarului functioneaza, prezentand totalitatea caracteristicilor acesteia inca de la partea de autentificare si pana la generarea efectiva a orarelor, detaliindu-se fiecare dinre rolurile utilizatorilor ce o pot folosi.

6.1.1 Roluri si permisiuni

In functie de profilul utilizatorului care efectueaza autentificarea, aplicatia poate pune la dispozitie un set de caracteristici ce uneori pot fi distincte, iar alteori identice. Astfel, putem spune pe scurt ca un utilizator cu rol de cadru didactic poate seta preferinte orare in functie de materia predata pentru orarele efective, in timp ce operatiunea de generare ii va reveni unui utilizator cu rol de administrator. In figura urmatoare (vezi Figura 33.) sunt detaliate toate functionalitatatile aplicatiei pe tipuri de utilizatori.

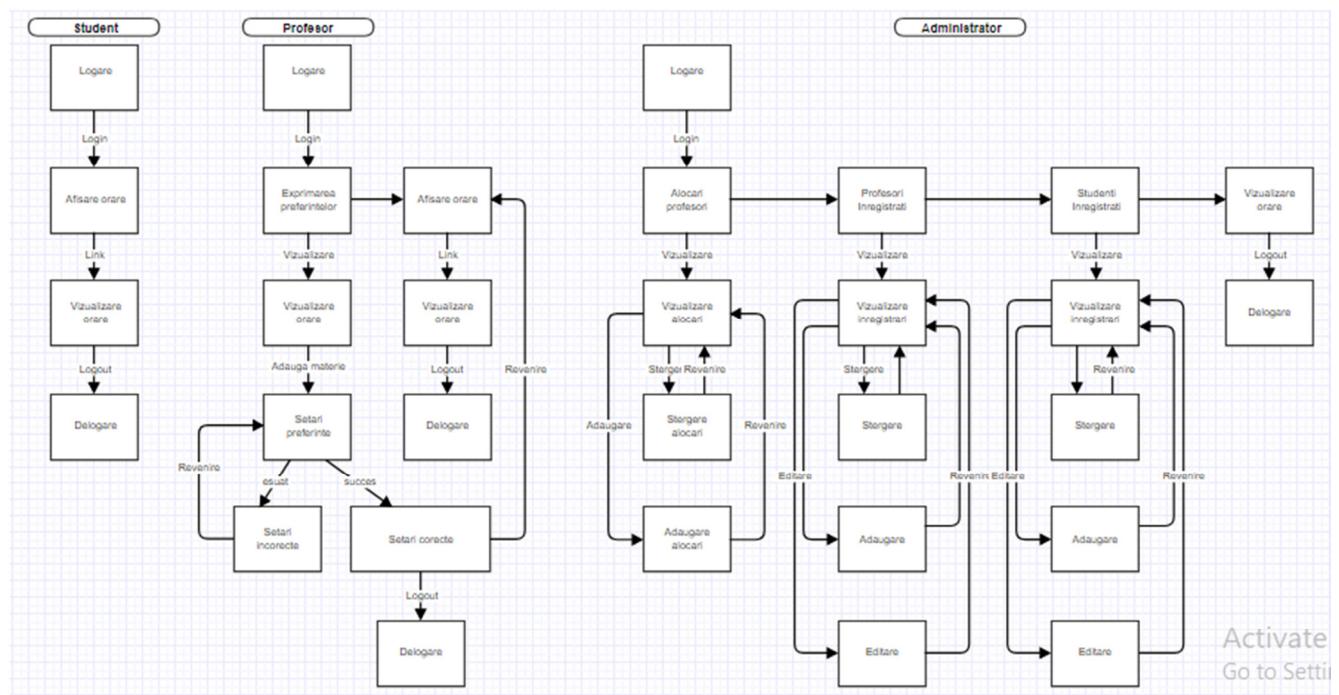


Figura 33. Schema functionalitatilor pe tipuri de utilizatori

6.1.2 Rolul Student

Rolul student este cel mai simplist profil al aplicatiei prezентate. Principala functionalitate a rolului student este aceea de vizualizare a orarelor.

Autentificarea in aplicatie se face conform imaginii de mai jos (vezi Figura 34.) prin introducerea unui nume de utilizator/email si a unei parole de acces, acestea fiind urmate de selectarea rolului „Student” dintr-o lista de tip drop-down si de apasarea butonului „Login”.

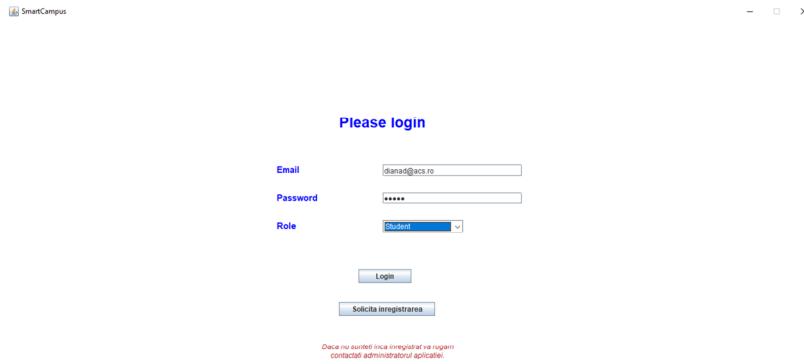


Figura 34. Fereastra de autentificare

In cazul in care utilizatorul nu dispune de credentiale de acces valide, acesta va trebui sa trimita administratorului aplicatiei o cerere de inregistrare prin apasarea butonului „Solicita inregistrare”. Apasare butonului „Solicita inregistrare” va deschide o fereastra in care utilizatorului i se va cere sa introduca date personale, acestea fiind necesare administratorului pentru acordarea accesului. O astfel de fereastra este ilustrata in imaginea de mai jos (vezi Figura 35.).

Figura 35. Fereastra de inregistrare

Daca autentificarea utilizatorului s-a efectuat cu succes urmatorul ecran afisat va fi cel de vizualizare orare, de altfel si singurul ecran disponibil pentru rolul student (vezi Figura 36.).



Figura 36. Fereastra de vizualizare orare

Ecranul „Afisare orare” contine ca unica functionalitate adresa web la care rolul student poate accesa orarele disponibile. Odata accesata aceasta adresa, sistemul va trimite utilizatorul catre o locatie externa unde administratorul a incarcat orarele. Utilizatorul student va avea optiunea de a descarca orarul din aceasta locatie externa (vezi Figura 37.).

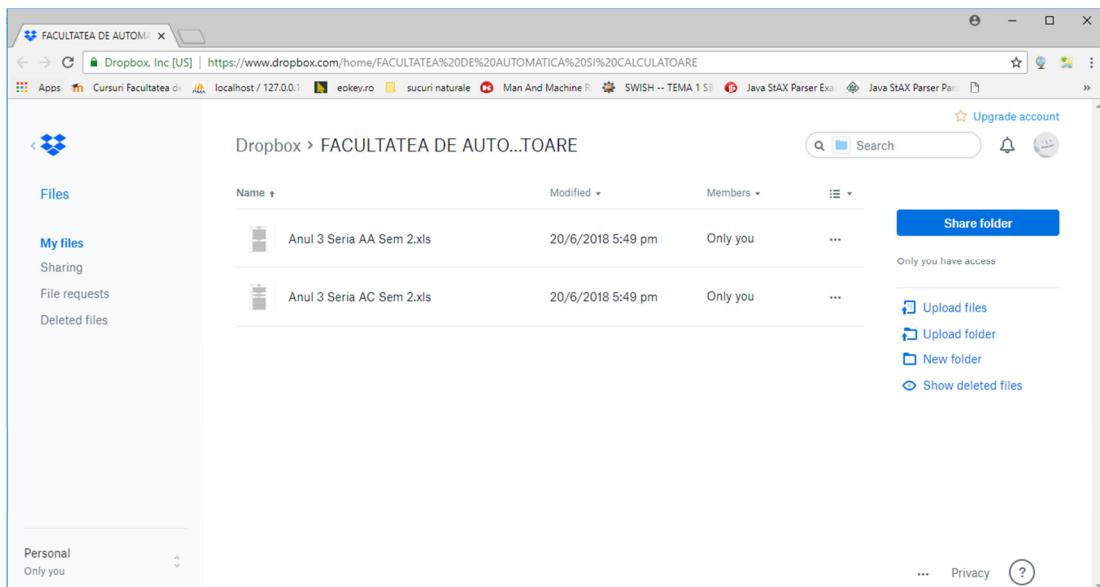


Figura 37. Adresa web pentru orare incarcate

6.1.3 Rolul Cadru didactic

Rolul cadru didactic este cel de-al doilea profil al aplicatiei prezentate, fiind oarecum mai complex decat rolul student. Principala functionalitate a rolului cadru didactic este aceea de a seta preferinte orare de predare in functie de intervale orare, serii si materii.

Autentificarea in aplicatie se face conform pasilor din capitolul anterior, singura exceptie fiind selectarea rolului „Profesor” dintr-o lista de tip drop-down si de apasarea butonului „Login” (vezi Figura 38.).

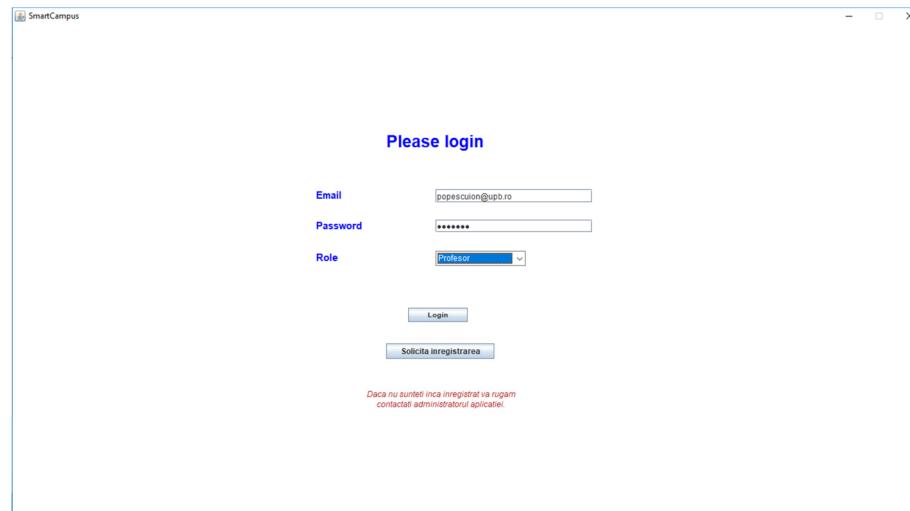


Figura 38. Fereastra de autentificare

Ulterior autentificarii cu succes in aplicatie, cele doua meniuri accesibile rolului cadre didactice vor fi „Exprimarea preferintelor” si „Afisarea orarului”.

Meniul „Exprimarea preferintelor” va contine initial o lista de tip drop-down din care cadrul didactic poate selecta ziua pentru care doreste sa seteze o preferinta (vezi Figura 39.).



Figura 39. Fereastra de exprimare a preferintelor

Dupa selectarea unei zile de Luni pana Vineri, o noua lista de tip drop-down va fi afisata, lista ce va permite selectarea materiei pentru care se doreste a se adauga preferinta. Este de mentionat faptul ca materiile prezente in lista de selectie vor fi strict materiile pentru care cadrul didactic autentificat in aplicatie va avea drept de setare a preferintelor (vezi Figura 40.). Aceasta

functionalitate va fi posibila in functie de alocarile efectuate pentru cadrul didactic in cauza de catre rolul administrator in pagina de administrare.



Figura 40. Fereastra de exprimare a preferintelor

Odata ce atat ziua, cat si materia au fost selectate, aplicatia va afisa o lista cu seriile ce se pot selecta pentru conditiile selectate anterior (vezi Figura 41.).



Figura 41. Fereastra de exprimare a preferintelor

Dupa alegerea unei serii se poate apasa butonul „Vizualizeaza” iar in partea dreapta a ecranului va aparea un tabel ce va prezenta intervalele orare libere si cele deja ocupate din ziua respectiva de catre alte cadre didactice.



Figura 42. Fereastra de exprimare a preferintelor

In functie de acest tabel cadrul didactic poate vedea intervalele orare in care isi va putea exprima preferintele. Pentru aceasta se va proceda la apasarea butonului „Adauga materia”, ce va afisa o fereastra in care se va putea seta ora de incepere a cursului, automat calculandu-se intervalul orar in functie de durata cursului din planul de invatamant al seriei (vezi Figura 43.).

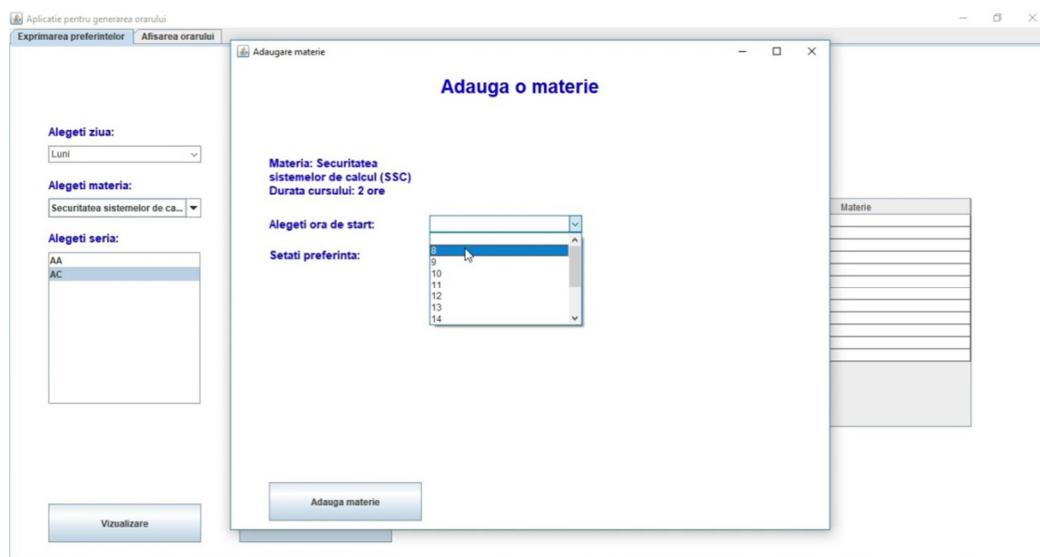


Figura 43. Fereastra de adaugare a preferintei

Dupa alegerea orei de start urmatoarea lista de tip drop-down va fi aceea de exprimare concisa a preferintei cadrului didactic, in care vor fi puse la dispozitia acestuia trei calificative pentru ora selectata anterior, calificative ce vor determina gradul de prioritate al preferintelor in generarea orarului:

- Favorit – semnifica disponibilitatea ce a mai mare a cadrului didactic pentru sustinerea cursului. Prioritate 1.
- Indiferent– semnifica o eventuala disponibilitate a cadrului didactic pentru sustinerea cursului. Prioritate 2.
- Nedorit– semnifica indisponibilitatea cadrului didactic pentru sustinerea cursului. Prioritate 3.

Popularea tabelului de vizualizare a intervalelor orare se va face doar in situatia alegerii ca favorit a unui interval, acest calificativ avand gradul de prioritate cel mai mare. In acest caz intervalul setat ca favorit va fi vizibil ca ocupat tuturor celorlalte cadre didactice. In cazul in care din diverse motive se doreste setarea ca favorit a unui interval orar ocupat, aplicatia va genera un mesaj de avertizare, ce va instiinta utilizatorul de problema aparuta cerandu-i acestuia selectarea unui alt interval (vezi Figura 44.).

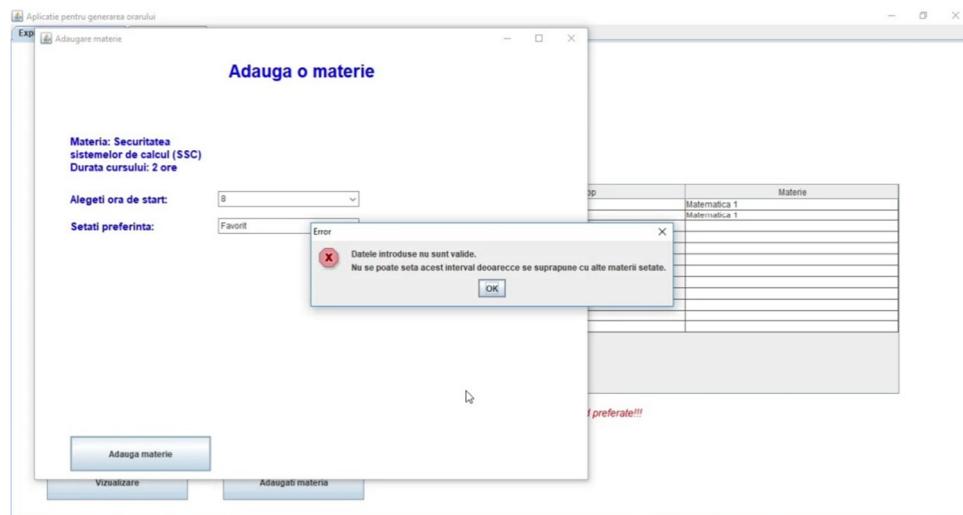


Figura 44. Eroare la adaugarea preferintei

De asemenea, alegerea unui interval disponibil si setarea acestuia ca favorit va returna cadrului didactic autentificat un mesaj de succes si va popula tabelul de vizualizare a intervalelor orare cu selectia proaspata efectuata (vezi Figura 45.).

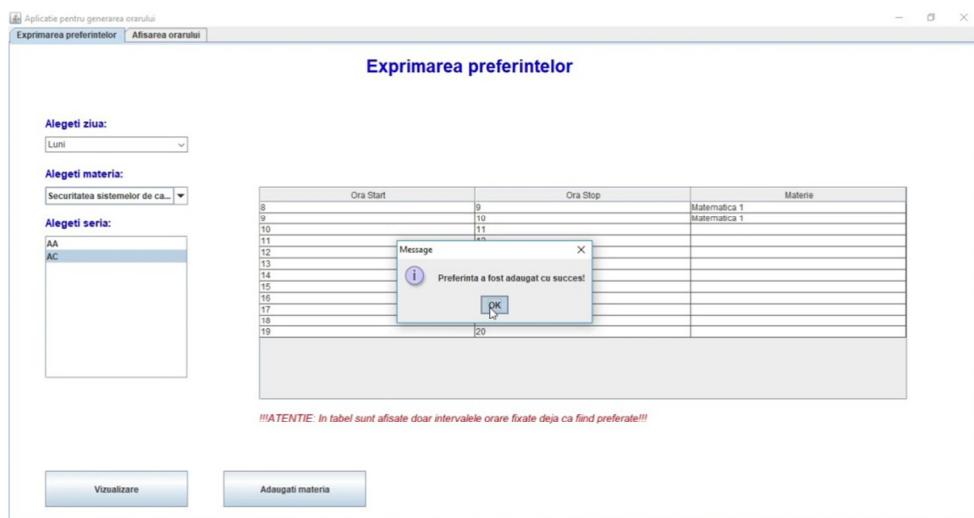


Figura 45. Succes la adaugarea preferintei

Meniul „Afisarea orarelor” contine ca unica functionalitate adresa web la care cadrul didactic poate accesa orarele disponibile. Odata accesata aceasta adresa, sistemul va trimite utilizatorul catre o locatie externa unde administratorul a incarcat orarele. Cadrul didactic va avea optiunea de a descarca orarul dorit din locatia externa (vezi Figura 46.).

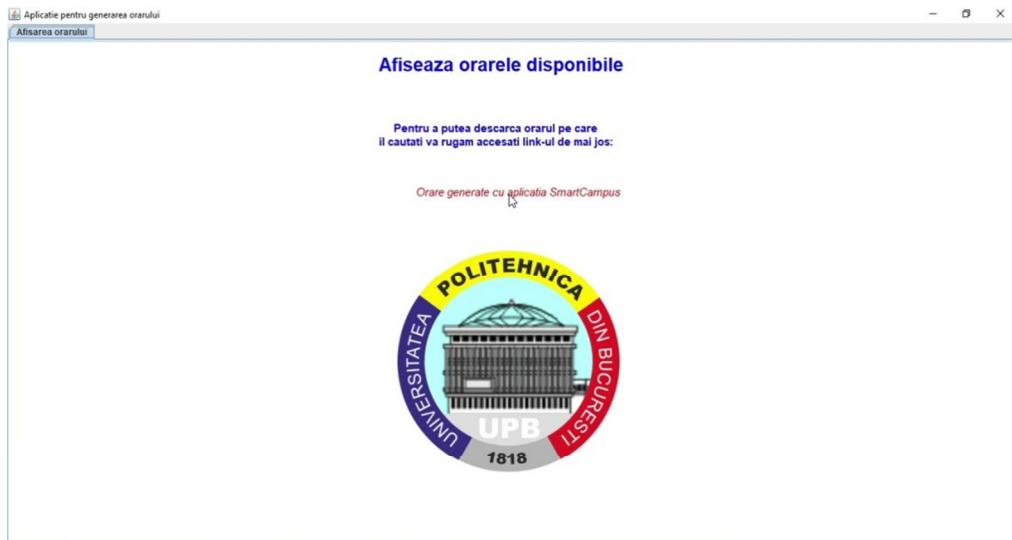


Figura 46. Fereastra de vizualizare orare

6.1.4 Rolul Administrator

Rolul Administrator este cel de-al treilea profil al aplicatiei prezentate, fiind si cel mai complex dintre roluri. Principala functionalitate a rolului administrator este aceea de a genera orarele efective, in functie de preferințele setate anterior de catre fiecare cadrul didactic, de disponibilitatea salilor de curs si de planul de invatamant al fiecarei serii.

In figura urmatoare sunt ilustrate intr-o maniera grafica toate functionalitatile utilizatorului cu rol de administrator (vezi Figura 47.)

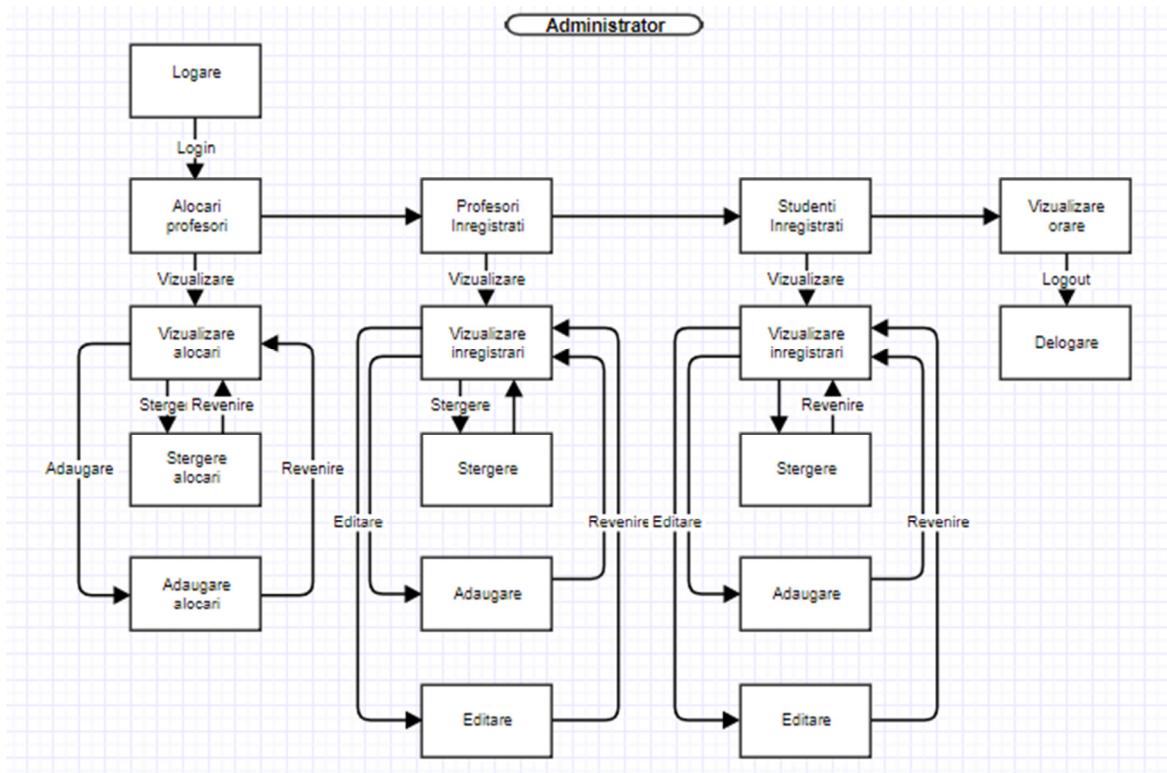


Figura 47. Functionalitati ale rolului de administrator

Autentificarea in aplicatie se face conform pasilor din capitolele anterioare singura exceptie fiind selectarea rolului „Administrator” din lista de tip drop-down urmata de apasarea butonului „Login” (vezi Figura 48.).

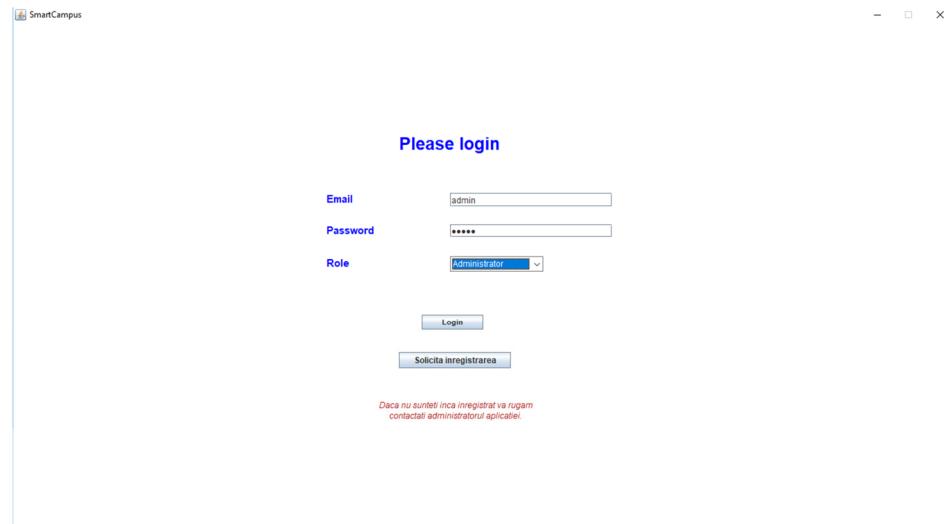


Figura 48. Fereastra de autentificare

Ulterior autentificarii cu succes in aplicatie cele sase meniuri accesibile rolului Administrator vor fi:

- Alocari profesori
- Profesori inregistrati
- Studenti inregistrati
- Materii inregistrate
- Sali inregistrate
- Generare orare

Meniu Alocari Profesori

In acest ecran administratorul are la dispozitie o functie de vizualizare a unui tabel ce cuprinde totalitatea datelor referitoare la cadrele didactice implicate in procesul de invatamant, tabel structurat pe coloane cu numele, prenumele, email-ul, departamentul, materia si numarul orelor de curs ale fiecarui cadru didactic (vezi Figura 49.).

Alocari

Vizualizare

Id profesor	Id materie	Nume	Prenume	Email	Departament	Materie	Ore_Curs
1	9	Popescu	Ion	popescuion@upb.acs	Baze de Date (BD)	2	
1	10	Popescu	Ion	popescuion@upb.acs	Mecatronica (MCT)	2	
2	11	Ionescu	Maria	ionescumaria@upb.acs	Ingineria Reglarii	3	
1	12	Popescu	Ion	popescuion@upb.acs	Aplicatii Web	2	
2	13	Popescu	Ion	popescuion@upb.acs	Informatica Structurata	2	
2	14	Ionescu	Maria	ionescumaria@upb.acs	Managementul Pr.	2	
2	15	Ionescu	Maria	ionescumaria@upb.acs	Gestuarea si Mana	2	
1	18	Popescu	Ion	popescuion@upb.acs	Retele de Calculat.	2	
2	19	Ionescu	Maria	ionescumaria@upb.acs	Sisteme de Cond...	2	
2	20	Ionescu	Maria	ionescumaria@upb.acs	Securitatea sist...	2	
1	21	Popescu	Ion	popescuion@upb.acs	Matematica 1	2	

Stergere Adaugare

Figura 49. Meniu Alocari Profesori

Scopul acestui ecran este acela de a avea acces in permanenta la situatia cadrelor didactice active pentru o mai buna organizare a orarelor, fiind totodata folosit si ca un panou de comanda, administratorul putand sa aloce fiecarui cadru didactic una sau mai multe materii, materii care ulterior vor fi afisate in lista de materii ale cadrului didactic respectiv la autentificarea acestuia in meniul de setare preferinte.

Optiunile disponibile in acest meniu sunt cele de Vizualizare, Stergere si Adaugare.

Butonul Vizualizare – afiseaza inregistrarile in tabel. In cazul in care tabelul este deja populat cu date butonul „Vizualizeaza” are functie de reincarcare, aducand in pagina ultimele modificari.

Butonul Stergere – va fi functional in urma selectarii unei intrari din tabel, selectie ce se doreste a fi stearsa. La apasare, acesta va deschide o noua fereastra ce va cuprinde intrarea selectata, fereastra ce are involuntar rolul de ecran de confirmare si care cuprinde randul ce se doreste a fi sters si optiunea de stergere si cea de renuntare (vezi Figura 50.).

In cazul in care initial nici o intrare din tabel nu a fost selectata un mesaj de atentionare va fi afisat, mesaj ce va indemna utilizatorul la selectia intrarii ce se doreste a fi stearsa.

Id profesor	Id materie	Nume	Prenume	Email	Departament	Materie	Ore_Curs
2	15	Ionescu	Maria	Ionescuma	ACS	Gestunea	2

Figura 50. Stergere alocare

Butonul Adaugare – foloseste la adaugarea unui nou cadru didactic impreuna cu caracteristicile ce il insotesc, acestea fiind – Denumirea materiei, Numele Profesorului, Prenumele Profesorului si Numarul orelor de curs. La finalul formularului de adaugare alocare se va regasi butonul „Adauga alocare” care va salva noua intrare, o va afisa in tabelul din pagina „Vizualizare alocari” si va inchide fereastra de adaugare intorcand administratorul in ecranul de „Vizualizare” (vezi Figura 51.).

Figura 51. Adauga alocare

Posibilitatea de editare a intrarilor deja prezente in ecranul de „Vizualizare alocari” nu este disponibila, fiind vorba de intrari simple ale tabelului afisat.

Astfel, daca se doreste modificarea unei alocari aceasta se va face prin stergerea alocarii existente si adaugarea uneia noi. La fel se va proceda spre exemplu si pentru alocarea a doua sau mai multe materii aceluiasi cadru didactic, introducandu-se cate o intrare noua pentru fiecare materie.

Meniul Profesori inregistrati

In acest ecran administratorul are la dispozitie o functie de vizualizare a unui tabel ce cuprinde totalitatea datelor referitoare la cadrele didactice implicate in procesul de invatamant, tabel structurat pe coloane cu numele, prenumele, emailul, parola si departamentul fiecarui cadru didactic (vezi Figura 52.).

The screenshot shows a Windows application window titled 'Profesori'. At the top, there is a menu bar with 'Administrare Aplicatie', 'Alocare Profesori', 'Profesori inregistrati' (which is selected), 'Studenti inregistrati', 'Generare orare', and 'Ajutor'. Below the menu is a sub-menu with 'Alocari Profesori', 'Profesori inregistrati' (selected), 'Studenti inregistrati', and 'Vizualizare ore'. The main area is titled 'Profesori' and contains a button labeled 'Vizualizare'. Below this is a table with the following data:

ID	Nume	Prenume	Email	Parola	Departament
1	Popescu	Ion	popescuion@upb.ro	popescu	ACS
2	Ionescu	Maria	ionescumaria@upb.ro	ionescu	ACS
3	Georgescu	Andrei	georgescuandrei@upb.ro	georgescu	ACS
4	Constantinescu	Alexandra	constantinescualexandra	constantinescu	ACS

At the bottom of the window are three buttons: 'Stergere' (Delete), 'Adaugare' (Add), and 'Editare' (Edit).

Figura 52. Meniu Profesori Inregistrati

Scopul acestui ecran este acela de a organiza, permite sau restrictiona accesul utilizatorilor cu rol de cadru didactic in orice moment la aplicatia de organizare a orarelor.

Optiunile disponibile in acest meniu sunt cele de Vizualizare, Stergere, Adaugare si Editare.

Butonul Vizualizare – afiseaza intrarile in tabel. In cazul in care tabelul este deja populat cu date butonul vizualizeaza are functie de reincarcare, aducand in pagina ultimele modificari.

Butonul Stergere – va fi functional in urma selectarii unei intrari din tabel, selectie ce se doreste a fi stearsa. La apasare, acesta va deschide o noua fereastra ce va cuprinde intrarea selectata, fereastra ce are involuntar rolul de ecran de confirmare si care cuprinde randul ce se doreste a fi sters si optiunea de stergere si cea de renuntare (vezi Figura 53.).

In cazul in care initial nici o intrare din tabel nu a fost selectata un mesaj de atentionare va fi afisat, mesaj ce va indemna utilizatorul la selectia intrarii ce se doreste a fi stearsa.

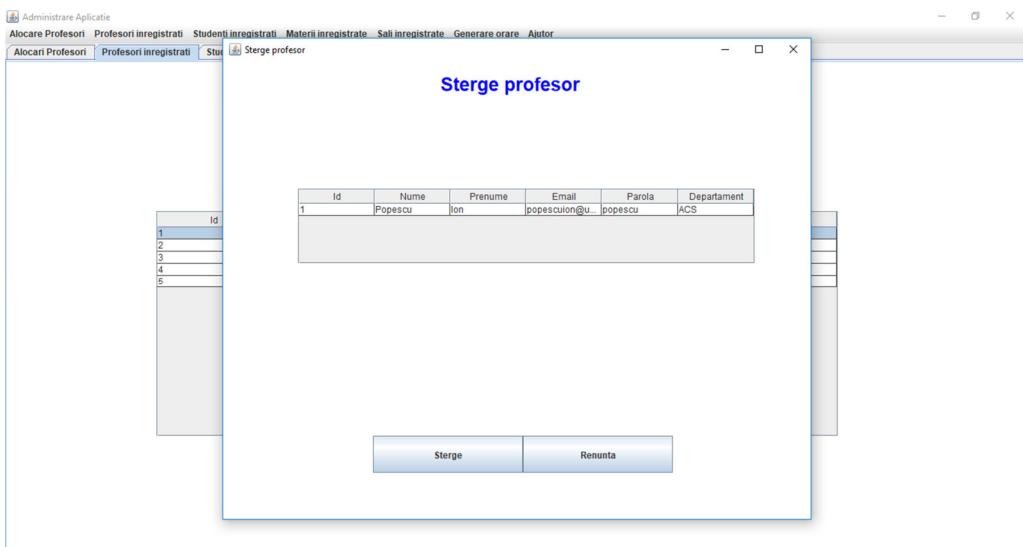


Figura 53. Sterge profesor

Butonul Adaugare – foloseste la adaugarea unui nou cadru didactic impreuna cu caracteristicile ce il insotesc, acestea fiind – Numele profesorului, Prenumele profesorului, Emailul profesorului, Parola si Departamentul (vezi Figura 54.). La finalul formularului de adaugare se va regasi butonul Adauga inregistrare care va salva noua intrare, o va afisa in tabelul din pagina Profesori Inregistrati si va inchide fereastra de adaugare intorcand administratorul in ecranul de Vizualizare.

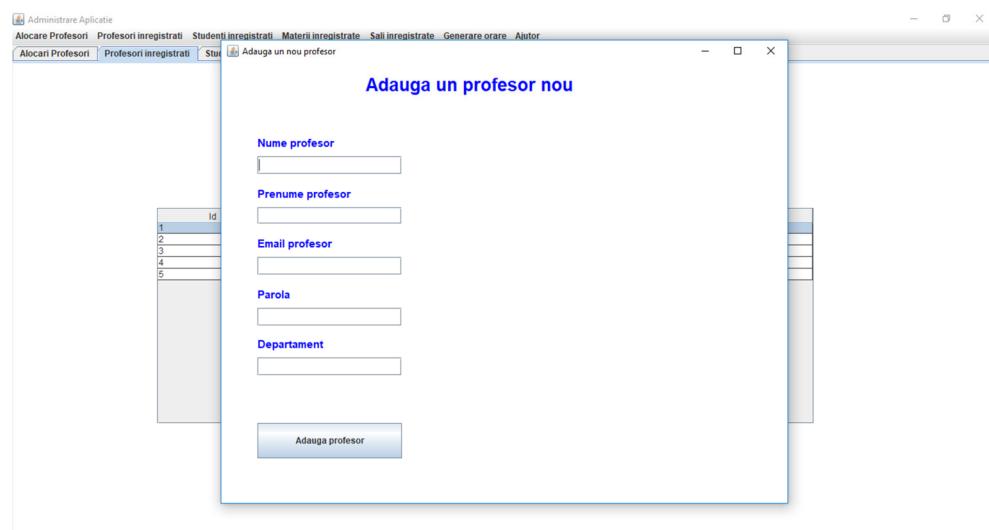


Figura 54. Adaugare profesor

Butonul Editare - daca se doreste modificarea unei inregistrari aceasta se va face prin editarea celei existente spre deosebire de cazul alocarii cadrelor didactice, unde aceasta optiune nu era disponibila.La apasare, acesta va deschide o noua fereastra ce va cuprinde intrarea selectata, cu mentiunea ca la dublu-click pe oricare din campurile din tabel, textul sau devine editabil, administratorul putand in orice moment dupa efectuarea modificarilor, fie sa salveze prin apasarea butonului „Salveaza modificarile”, fie sa renunte prin apasarea butonului „Renunta” (vezi Figura 55.).

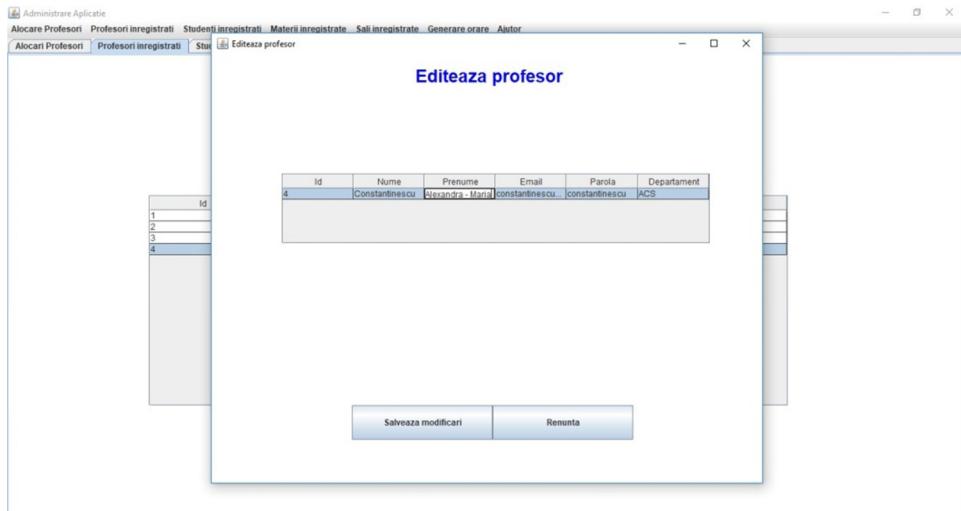


Figura 55. Editeaza profesor

Meniul Studenti inregistrati

In acest ecran administratorul are la dispozitie o functie de vizualizare a unui tabel ce cuprinde totalitatea datelor referitoare la studentii implicați în procesul de învățământ, tabel structurat pe coloane cu numele, prenumele, emailul, parola, grupa și anul fiecarui student (vezi Figura 56.).

Studenti							
Vizualizare							
Stergere Adaugare Editare							
Id	Nume	Prenume	Email	Parola	Grupa	An	
3	Dimitriu	Diana	diana@acs.ro	diana	341A3	4	
4	Antonescu	Marcel	marcel@acs.ro	marcel	341AB	3	

Figura 56. Meniu Studenti Inregistrati

Scopul acestui ecran este acela de a organiza, permite sau restrictiona accesul utilizatorilor cu rol de studenti in orice moment la aplicatia de organizare a orarelor.

Optiunile disponibile in acest meniu sunt cele de Vizualizare, Stergere, Adaugare si Editare.

Butonul Vizualizare – afiseaza intrarile in tabel. In cazul in care tabelul este deja populat cu date butonul vizualizeaza are functie de reincarcare, aducand in pagina ultimele modificari.

Butonul Stergere – va fi functional in urma selectarii unei intrari din tabel, selectie ce se doreste a fi stearsa. La apasare, acesta va deschide o noua fereastra ce va cuprinde intrarea selectata, fereastra ce are involuntar rolul de ecran de confirmare si care cuprinde randul ce se doreste a fi sters si optiunea de stergere si cea de renuntare (vezi Figura 57.).

In cazul in care initial nici o intrare din tabel nu a fost selectata un mesaj de atentionare va fi afisat, mesaj ce va indemna utilizatorul la selectia intrarii ce se doreste a fi stearsa.

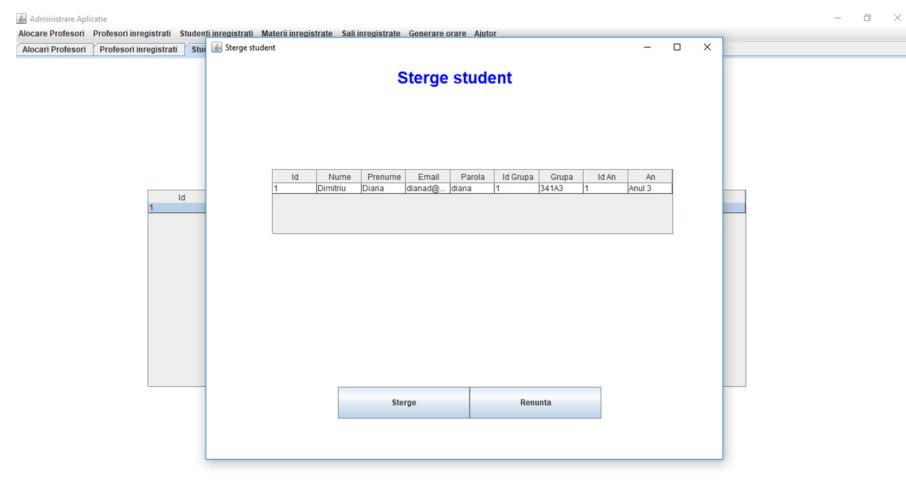


Figura 57. Stergere Student

Butonul Adaugare – foloseste la adaugarea unui nou student impreuna cu caracteristicile ce il insotesc acestea fiind – Numele studentului, Prenumele studentului, Emailul studentului, Parola si Grupa si Anul (vezi Figura 58.).

La finalul formularului de adaugare student se va regasi butonul Adauga inregistrare care va salva noua intrare, o va afisa in tabelul din pagina Studenti Inregistrati si va inchide fereastra de adaugare intorcand administratorul in ecranul de Vizualizare.

Figura 58. Adaugare Student

Butonul Editare - daca se doreste modificarea unei inregistrari aceasta se va face prin editarea celei existente spre deosebire de cazul alocarii cadrelor didactice, unde acesta optiune nu era disponibila (vezi Figura 59.).

Id	Nume	Prenume	Email	Parola	Id Grupa	Grupa	Id An	An
1	Dimitru	Danu	danu@q.com	Danu	1	341A3	1	Anul 3

Figura 59. Editeaza Student

La apasare, acesta va deschide o noua fereastra ce va cuprinde intrarea selectata, cu mentiunea ca la dublu-click pe oricare din campurile din tabel, textul sau devine editabil, administratorul putand in orice moment dupa efectuarea modificarilor fie sa salveze prin apasarea butonului „Salveaza modificarile”, fie sa renunte prin apasarea butonului „Renunta”.

Meniul Materii Inregistrate

In acest ecran administratorul are la dispozitie o functie de vizualizare a unui tabel ce cuprinde totalitatea datelor referitoare la materiile implicati in procesul de invatamant, tabel structurat pe coloane cu denumirea, durata, cadrul didactic si id-ul acestuia pentru fiecare materie (vezi Figura 60.).

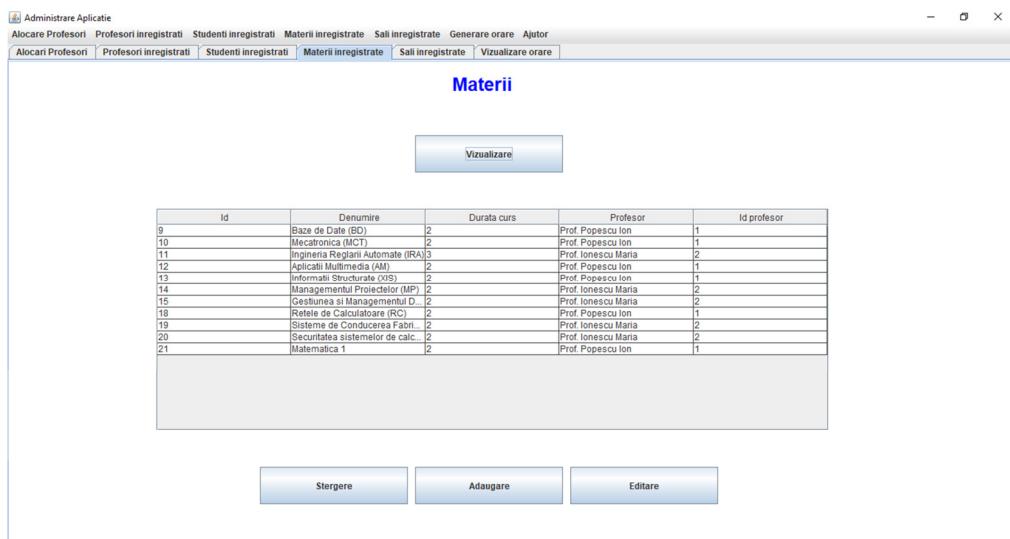


Figura 60. Meniu Materii Inregisterate

Scopul acestui ecran este acela de a organiza lista de materii in orice moment al aplicatiei de organizare a orarelor.

Optiunile disponibile in acest meniu sunt cele de Vizualizare, Stergere, Adaugare si Editare.

Butonul Vizualizare – afiseaza intrarile in tabel. In cazul in care tabelul este deja populat cu date butonul vizualizeaza are functie de reincarcare, aducand in pagina ultimele modificari.

Butonul Stergere – va fi functional in urma selectarii unei intrari din tabel, selectie ce se doreste a fi stearsa. La apasare, acesta va deschide o noua fereastra ce va cuprinde intrarea selectata, fereastra ce are involuntar rolul de ecran de confirmare si care cuprinde randul ce se doreste a fi sters si optiunea de stergere si cea de renuntare (vezi Figura 61.).

In cazul in care initial nici o intrare din tabel nu a fost selectata un mesaj de atentionare va fi afisat, mesaj ce va indemna utilizatorul la selectia intrarii ce se doreste a fi stearsa.

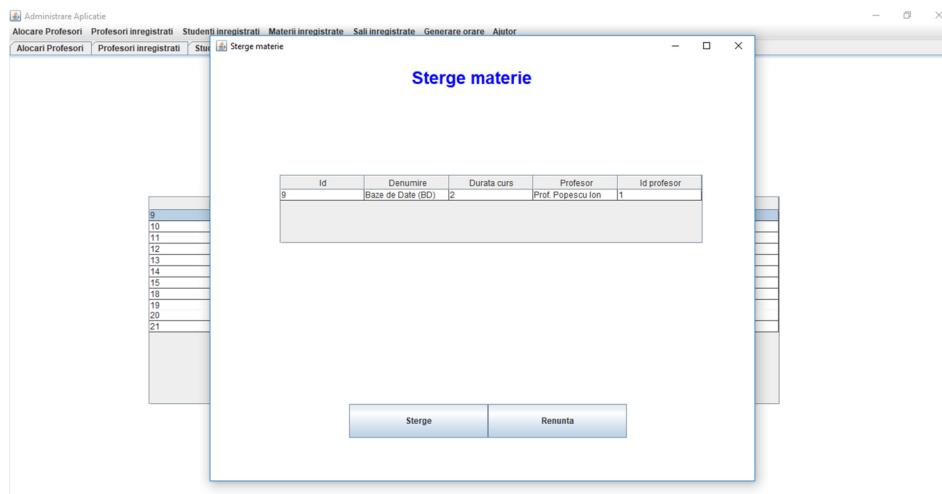


Figura 61. Stergere Materie

Butonul Adaugare – foloseste la adaugarea unei noi materii impreuna cu caracteristicile ce o insotesc acestea fiind – Denumirea materiei, Durata materiei, Numele profesorului si Prenumele profesorului (vezi Figura 62.).

La finalul formularului de adaugare materie se va regasi butonul Adauga inregistrare care va salva noua intrare, o va afisa in tabelul din pagina Materii Inregistrate si va inchide fereastra de adaugare intorcand administratorul in ecranul de Vizualizare.

Figura 62. Adaugare Materie

Butonul Editare - daca se doreste modificarea unei inregistrari aceasta se va face prin editarea celei existente spre deosebire de cazul alocarii cadrelor didactice, unde acesta optiune nu era disponibila (vezi Figura 63.).

Figura 63. Editeaza Materie

La apasare, acesta va deschide o noua fereastra ce va cuprinde intrarea selectata, cu mentiunea ca la dublu-click pe oricare din campurile din tabel, textul sau devine editabil,

administratorul putand in orice moment dupa efectuarea modificarilor fie sa salveze prin apasarea butonului „Salveaza modificarile”, fie sa renunte prin apasarea butonului „Renunta”.

Meniul Sali Inregistrate

In acest ecran administratorul are la dispozitie o functie de vizualizare a unui tabel ce cuprinde totalitatea datelor referitoare la salile implicati in procesul de invatamant, tabel structurat pe coloane cu numarul de locuri, etajul si denumirea pentru fiecare sala (vezi Figura 64.).

The screenshot shows a Windows application window titled 'Sali'. The menu bar includes 'Administrare Aplicatie', 'Alocare Profesori', 'Profesori inregistrati', 'Studenti inregistrati', 'Materii inregistrate', 'Sali inregistrate', 'Generare orare', and 'Ajutor'. Below the menu, tabs are visible: 'Alocari Profesori', 'Profesori inregistrati', 'Studenti inregistrati', 'Materii inregistrate', 'Sali inregistrate' (which is selected and highlighted in blue), and 'Vizualizare orare'. The main content area is titled 'Sali' and contains a button labeled 'Vizualizare'. Below this is a table with the following data:

	Id	Numar locuri	Etaj	Nume
1	100	0	EC001	
2	150	0	PR002	
3	200	1	AN030	
1004	200	1	EC101	

At the bottom of the page are three buttons: 'Stergere', 'Adaugare', and 'Editare'.

Figura 64. Meniu Sali Inregistrate

Scopul acestui ecran este acela de a organiza lista de sali in orice moment al aplicatiei de organizare a orarelor.

Optiunile disponibile in acest meniu sunt cele de Vizualizare, Stergere, Adaugare si Editare.

Butonul Vizualizare – afiseaza intrarile in tabel. In cazul in care tabelul este deja populat cu date butonul vizualizeaza are functie de reincarcare, aducand in pagina ultimele modificari.

Butonul Stergere – va fi functional in urma selectarii unei intrari din tabel, selectie ce se doreste a fi stearsa. La apasare, acesta va deschide o noua fereastra ce va cuprinde intrarea selectata, fereastra ce are involuntar rolul de ecran de confirmare si care cuprinde randul ce se doreste a fi sters si optiunea de stergere si cea de renuntare (vezi Figura 65.).

In cazul in care initial nici o intrare din tabel nu a fost selectata un mesaj de atentionare va fi afisat, mesaj ce va indemna utilizatorul la selectia intrarii ce se doreste a fi stearsa.

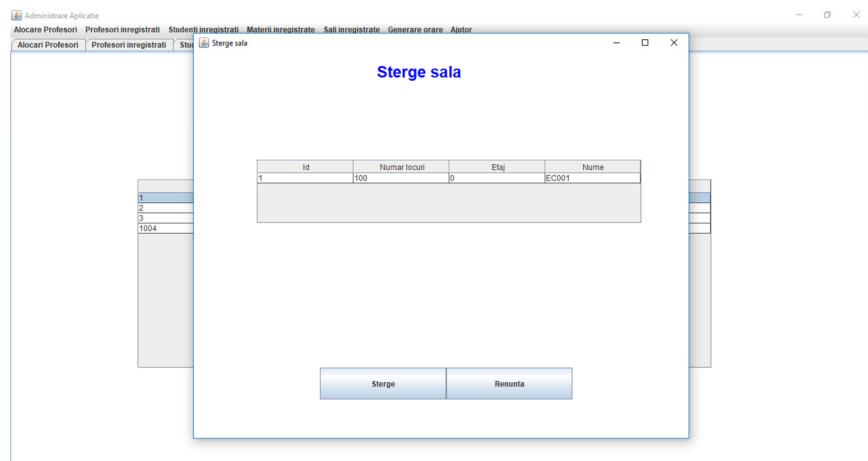


Figura 65. Stergere Sala

Butonul Adaugare – foloseste la adaugarea unei noi materii impreuna cu caracteristicile ce o insotesc acestea fiind – Numarul de locuri din sala, Etajul salii si Numele salii(vezi Figura 66.).

La finalul formularului de adaugare sala se va regasi butonul Adauga inregistrare care va salva noua intrare, o va afisa in tabelul din pagina Sali Inregistrate si va inchide fereastra de adaugare intorcand administratorul in ecranul de Vizualizare.

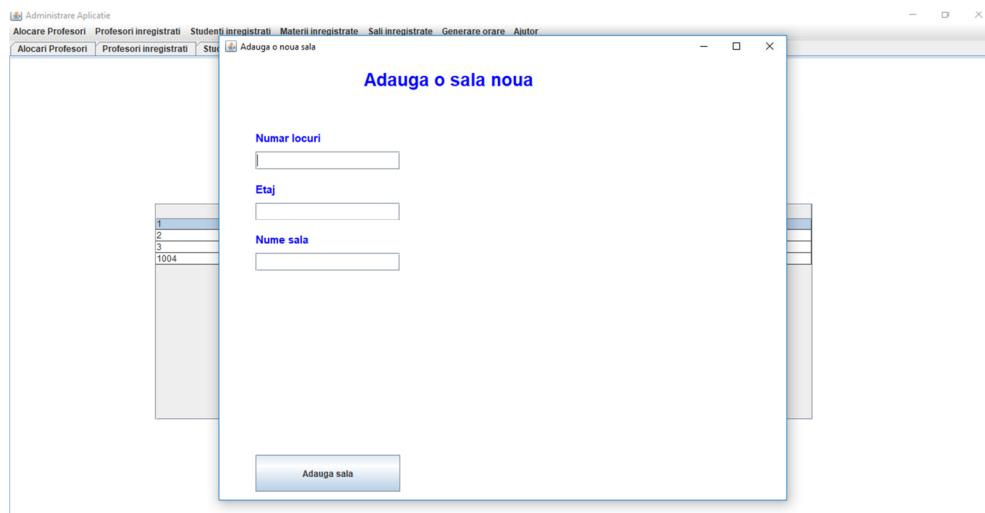


Figura 66. Adaugare Sala

Butonul Editare - daca se doreste modificarea unei inregistrari aceasta se va face prin editarea celei existente spre deosebire de cazul alocarii cadrelor didactice, unde acesta optiune nu era disponibila (vezi Figura 67.).

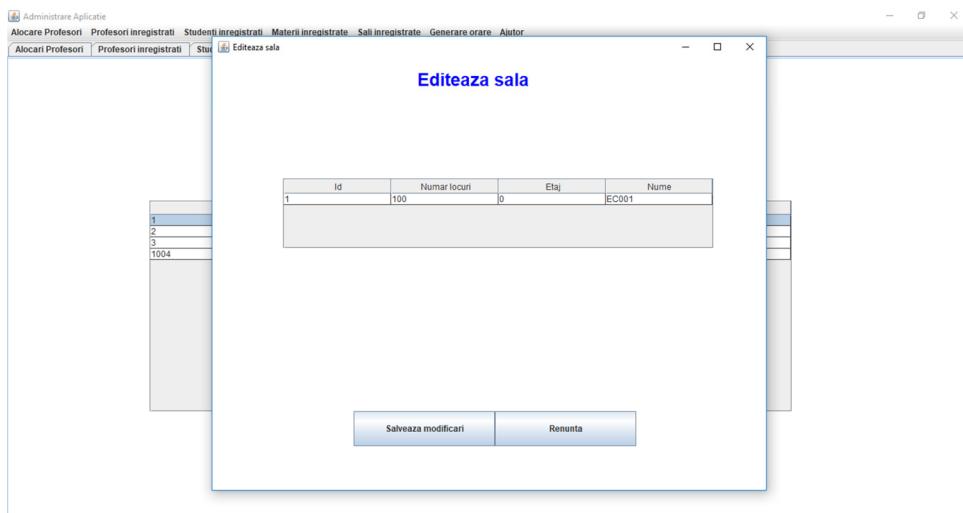


Figura 67. Editeaza Sala

La apasare, acesta va deschide o noua fereastra ce va cuprinde intrarea selectata, cu mentiunea ca la dublu-click pe oricare din campurile din tabel, textul sau devine editabil, administratorul putand in orice moment dupa efectuarea modificarilor fie sa salveze prin apasarea butonului „Salveaza modificarile”, fie sa renunte prin apasarea butonului „Renunta”.

Meniul Generarea Orarelor

Prin intermediul acestui meniu este realizat orarul pe facultati, semestre si serii de studenti.

Datorita complexitatii aplicatiei prezentate in randurile de mai sus si a faptului ca procesul de setare a preferintelor cadrelor didactitice se poate realiza foarte usor si intuitiv, generarea automata a orarului se va putea realiza prin simpla selectie a facultati pentru care se doreste orarul si a unuia dintre semestre (vezi Figura 68.).

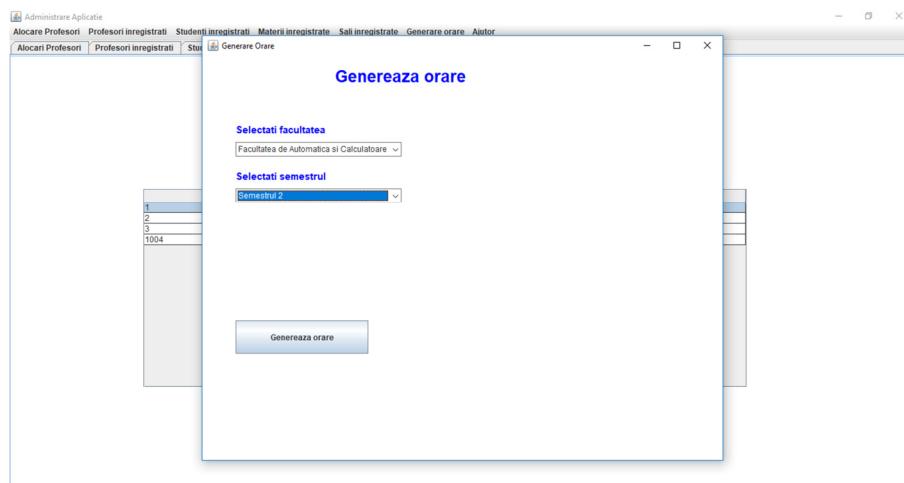


Figura 68. Meniul Generarea Orarelor

Ulterior apasarii butonului Genereaza Orare un mesaj de confirmare ce va informa administratorul ca orarul a fost generat cu succes va fi afisat. Orarul va fi salvat local pe disk la destinatia setata implicit de catre administrator (vezi Figura 69.).

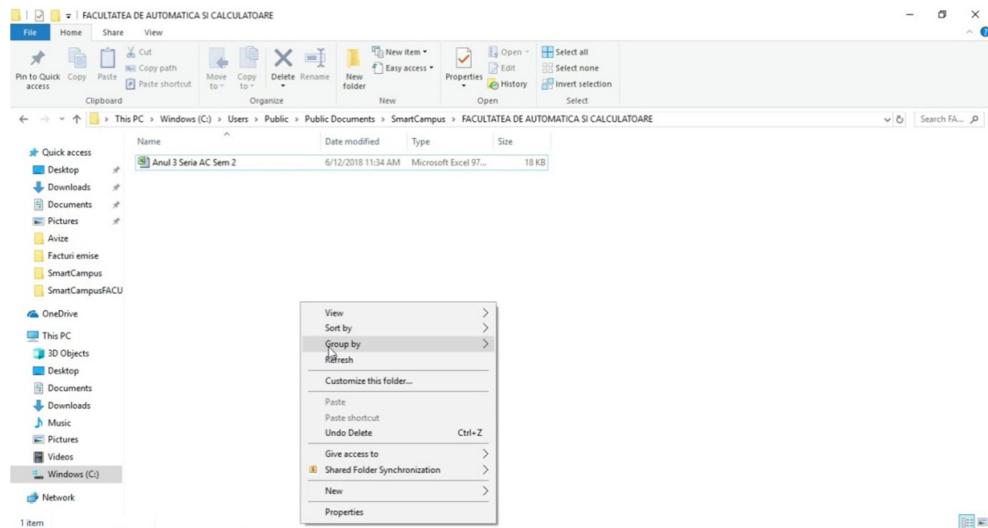


Figura 69. Directorul unde se gasesc orarele

Orarul prospat generat va fi salvat ca un fisier XLS cu denumirea de forma an/serie/semestru. Structurarea partii grafice si completarea fisierului este formatata automat de catre aplicatie in functie de datele si continutul fiecarui orar (vezi Figura 70.).

		A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
Ziua	Ora	Serie: AC																				
		Disciplina de sala																				
Ziua	8 - 10	Matematica 1 Prof. Popescu Ion Sala: PR002																				
Luni	10 - 16																					
	16 - 18	Securitatea sistemelor de calcul (SSC) Prof. Ionescu Maria Sala: PR002																				
	18 - 20																					
Marti	8 - 20																					
Miercuri	8 - 20																					

Figura 70. Exemplu de orar

Ulterior acestui pas administratorul poate incarca orarul pe o locatie externa unde atat utilizatorii cu rol de studenti cat si cadrele didactice vor avea optiunea de a-l descarca. Aceasta actiune se realizeaza prin simpla completare a adresei web a locatiei respective si apasarea butonului „Publica Orarele” (vezi Figura 71.). Aceasta locatie poate fi accesibila atat in retea facultatii cat si intr-un mediu extern putand fi accesata de oriunde si de oricine are la dispozitie un nume de utilizator si o parola .

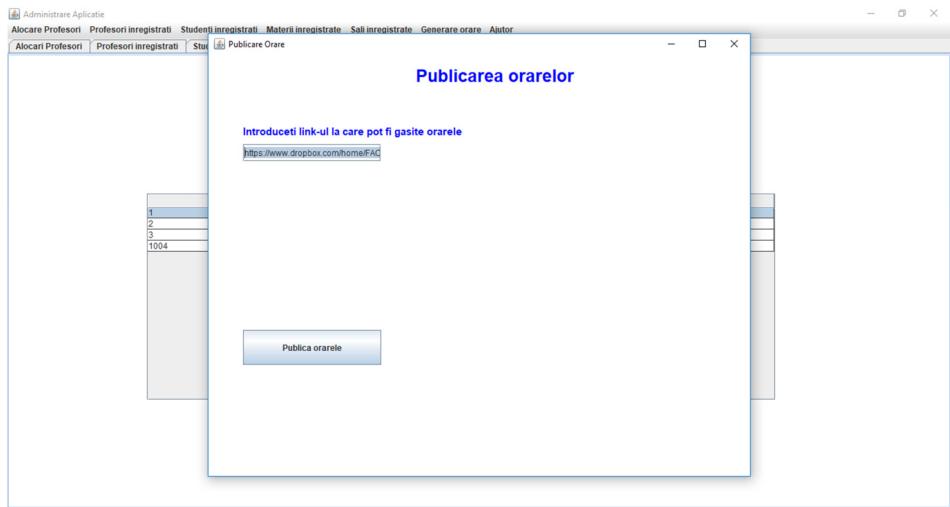


Figura 71. Publicarea orarelor

6.2 Verificarea si validarea aplicatiei

Prin aceste activitati s-a verificat daca aplicatia satisface specificatiile sale, in timpul fiecarei faze a ciclului sau de dezvoltare. Verificarea si validarea aplicatiei de modelare si generare automata a orarelor unei facultati a fost realizata:

- Verificand ca fiecare functionalitate indeplineste cerintele specificate;
- Verificand fiecare functionalitate astfel incat sa poata fi utilizata ca intrare pentru o alta activitate;
- Asigurand ca fiecare functionalitate este verificata, pe cat posibil, de o persoana diferita de aceea care a dezvoltat-o;
- Asigurand ca efortul de verificare si validare este adevarat si suficient pentru ca fiecare functionalitate sa fie operationala.

Tehnicile folosite pentru verificarea si validarea aplicatiei de modelare si generare automata a orarelor unei facultati au fost urmatoarele:

- **Analiza statica**

Examinarea unor documente (specificatii, modele conceptuale, diagrame de clase, cod sursa, documentatii de utilizare).

Activitati de inspectare a codului, analiza algoritmilor, demonstrarea corectitudinii.

- **Analiza dinamica**

Examinarea comportamentului programului cu scopul de a evidenția defectiuni posibile.

Activitatea de executie propriu-zisa a programului (testare).

Obiectivul activitatilor de verificare si validare a fost acela de a reduce erorile software la un nivel acceptabil.

6.2.1 Scenarii si date de test

Pentru testarea efectiva a aplicatiei de modelare si generare automata a orarelor unei facultati se vor descrie mai jos cateva dintre posibilele scenarii de test ce au fost implementate.

Scenariu	Date de intrare	ID test	Pasi	Rezultate Asteptate
Verifica logarea unui utilizator cu rol de student cu cont existent	Email = “dianad@acs.ro” Parola = “diana” Rol = “Student”	1	1.1 lansare aplicatie 1.2 introducere email 1.3 introducere parola 1.4 selectare rol student 1.5 apasare “Login”	Logarea se efectueaza cu succes. Utilizatorul poate ajunge in meniu “Vizualizare orare”.
Verifica posibilitatea de logare a unui utilizator care detine un cont de student ce selecteaza un rol de profesor	Email = “dianad@acs.ro” Parola = “diana” Rol = “Profesor”	2	2.1 lansare aplicatie 2.2 introducere email 2.3 introducere parola 2.4 selectare rol profesor 2.5 apasare “Login”	Logarea nu se va putea efectua. Utilizatorul va primi un mesaj de avertizare, fiind informat ca datele introduse nu sunt cele corecte.
Verifica posibilitatea de logare a unui utilizator care detine un cont de student ce selecteaza un rol de administrator	Email = “dianad@acs.ro” Parola = “diana” Rol = “Administrator”	3	3.1 lansare aplicatie 3.2 introducere email 3.3 introducere parola 3.4 selectare rol administrator 3.5 apasare “Login”	Logarea nu se va putea efectua. Utilizatorul va primi un mesaj de avertizare, fiind informat ca datele introduse nu sunt cele corecte.
Verificare posibilitate de trimitere solicitare de inregistrare	Email = “ion@acs.ro” Parola = “ion” Rol = “Student” Grupa si Anul = “Grupa: 341A3, An: 4” To = “smartcampusapp@yahoo.com” Parola = “parola1234” Host = “smtp.mail.yahoo.com”	4	4.1 lansare aplicatie 4.2 apasare “Solicita inregistrare” 4.3 completare date de intrare 4.4 apasare “Trimite solicitare”	Solicitarea se va trimite catre administrator. Utilizatorul va primi un mesaj de confirmare a trimiterii solicitarii.

Verifica setarea preferintelor unui utilizator cadru didactic pentru un interval in care acesta este disponibil	Email = "popescuion@upb.ro" Preferinta = "Favorit" Ora_start = 8 Ora_stop = 10 Zi = "Vineri" Semestru = 2	5	5.1 lansare aplicatie 5.2 introducere date de autentificare 5.3 navigare la setare preferinte 5.4 completare date si alegere interval liber 5.5 apasare buton de setare a preferintelor	Mesajul de confirmare conform caruia preferințele au fost setate cu success va fi afisat.
Verifica setarea preferintelor unui utilizator cadru didactic in orarul unei serii intr-un interval orar liber	Serie = "AA" Ziua = "Miercuri" Materia = "Ingineria Reglarii Automate (IRA)" Sala = "AN030" Email = "popescuion@upb.ro"	6	6.1 lansare aplicatie 6.2 introducere date de autentificare 6.3 navigare la setare preferinte 6.4 completare date si alegere un interval liber 6.5 apasare buton de setare a preferintelor	Mesajul de confirmare conform caruia preferințele au putut fi setate va fi afisat.
Verifica disponibilitate interval orar intr-o anumita zi, pentru o anumita serie, intr-un anumit semestru	Serie = "AC" Preferinta = "Favorit" Ora_start = 8 Ora_stop = 11 Zi = "Marti" Semestru = 2	7	7.1 lansare aplicatie 7.2 introducere date de autentificare 7.3 navigare la setare preferinte 7.4 completare date si alegere un interval liber 7.5 apasare buton de setare a preferintelor	Mesajul de confirmare conform caruia preferințele au putut fi setate va fi afisat.
Verifica existenta materie in orarul seriei pentru un anumit an si un anumit semestru	Preferinta = "Favorit" Materie = "Ingineria Reglarii Automate (IRA)" Seria = "AA" An = "Anul 3" Semestru = 2	8	8.1 lansare aplicatie 8.2 introducere date de autentificare 8.3 navigare la setare preferinte 8.4 completare date si alegere un interval liber 8.5 apasare buton de setare a preferintelor	Mesajul de atentionare conform caruia preferințele nu au putut fi setate va fi afisat. Utilizatorul va fi informat ca materia aleasa exista deja in orarul seriei selectate.
Verifica disponibilitate sala intr-un anumit interval orar	Ora_start = 8 Ora_stop = 10 Zi = "Luni" Semestru = 1 Sala =	9	9.1 lansare aplicatie 9.2 introducere date de autentificare 9.3 navigare la setare preferinte	Mesajul de confirmare conform caruia preferințele au putut fi setate va fi afisat.

	“AN030”		9.4 completare date si alegere un interval liber 9.5 apasare buton de setare a preferintelor	
Verifica adaugarea in aplicatie a unui cadru didactic de catre un administrator la libera alegere a acestuia	Email = “admin” Parola = “admin” Nume cadru didactic = “Popa” Prenume cadru didactic = “Ionel” Email cadru didactic = “popaionel@upb.ro” Parola cadru didactic = “popa” Departament cadru didactic = “ACS”	10	10.1 autentificare in aplicatie cu rol administrator 10.2 navigare la meniul adaugare profesori 10.3 adaugare cadru didactic nou conform datelor de test 10.4 delegare din rolul administrator 10.5 autentificare cu rol de profesor conform numelui de utilizator si parolei nou create	Autentificarea se efectueaza cu succes. Utilizatorul proaspăt creat se poate loga în aplicatie cu rol de profesor.
Verifica editare date in aplicatie ale unui student de catre un administrator	Email = “admin” Parola = “admin” Email actualizat student = “dianad@acs.ro”	11	11.1 autentificare in aplicatie cu rol administrator 11.2 navigare la meniul editare student 11.3 actualizare student conform datelor de intrare 11.4 delegare din rolul administrator 11.5 logare cu rol de student conform numelui de utilizator actualizat	Logarea se efectueaza cu succes. Utilizatorul proaspăt actualizat se poate autentifica în aplicatie cu rol de student.
Verifica eliminare date in aplicatie ale unei materii de catre un administrator	Email = “admin” Parola = “admin” Id Materie = 1	12	12.1 autentificare in aplicatie cu rol administrator 12.2 navigare la meniul stergere materie 12.3 stergere materie conform datelor de intrare 12.4 delegare din rolul administrator	Logarea se efectueaza cu succes. Utilizatorul nu mai poate gasi materia în lista de materii.

			12.5 logare cu rol de profesor 12.6 navigare la meniul de adaugare preferinte	
Verifica generarea orarului de catre un administrator pentru un semestru si o anumita facultate	Email = “admin” Parola = “admin” Facultate = “Facultatea de automatica si calculatoare” Semestru = “2”	13	13.1. autentificare in aplicatie cu rol administrator 13.2 navigare la meniul generare orare 13.3 selectare facultatea si semestrul conform datelor de intrare 13.4 genereaza orarele	Orarele vor fi generate. Acestea vor cuprinde preferintele cadrelor didactice pe intervalele orare selectate, dar si materiile nesetate de catre cadrele didactice, dar care se afla in orarul seriei.
Verifica publicarea orarelor	Email = “admin” Parola = “admin” Adresa web = “https://www.dropbox.com/home/FACULTATEA%20DE%20AUTOMATICA%20SI%20CALCULATOARE”	14	14.1. autentificare in aplicatie cu rol administrator 14.2 navigare la meniul publicare orare 14.3 introducere adresa web conform datelor de intrare 14.4 publica orarele	Orarele vor fi publicate la adresa web specificata. Acestea vor putea fi vizualizate de toti utilizatorii aplicatiei.
Verifica vizualizare orarelor	Email = “dianad@acs.ro” Parola = “diana”	15	15.1. autentificare in aplicatie cu rol student 15.2 navigare la meniul afisare orare 15.3 accesare adresa web	Orarele vor vor putea fi vizualizate de catre utilizatorul cu rol de student intr-o pagina web deschisa automat de sistem.

6.2.2 Metrici si indicatori de calitate

Metricii in testarea unei aplicatii cum este cea de modelare si generare automata a orarelor unei facultati inglobeaza modele, indicatori si proprietatile acestora, precum si modalitati de evaluare si validare.

Pentru masurarea calitatii software a aplicatiei prezentate s-au folosit trei tipuri de metrici ca fiind cele mei simple si uzuale in cazul de fata:

- Fiabilitatea
- Mantenanta
- Testabilitatea

6.2.2.1 Fiabilitatea

Reprezinta raportul dintre numarul rularilor cu succes al aplicatiei si numarul total de rulari al acesteia fiind exprimat in procente si definit prin urmatoarea formula de calcul:

$$Fiabilitatea = \frac{F1}{F2} \times 100$$

Unde: F1= numarul rularilor cu succes

F2= numarul total al rularilor

In cazul aplicatiei noastre putem calcula in functie de aceasta formula care este procentul de fiabilitate sau mai bine zis cat de fiabila este aplicatia de modelare si generare automata a orarelor. Luand in calcul un numar total de aproximativ 3000 de rulari de la finalizarea dezvoltarii aplicatiei pana in prezent din care doar 200 de rulari au esuat, va rezulta un numar de 2800 rulari cu succes ale aplicatiei. Aplicand formula de mai sus va rezulta un coeficient de fiabilitate de 93%.

$$Fiabilitatea = \frac{2800}{3000} \times 100 = 93\%$$

Procentul rezultat in urma aplicarii formulei va semnifica probabilitatea de functionare a aplicatiei fara a avea intreruperi majore dupa ce aceasta va fi lansata, fiind utilizata de utilizatori normali.

6.2.2.2 Mentenanta

Reprezinta raportul dintre numarul defectelor ce au trebuit corectate si numarul total de defecte, fiind exprimat in procente si definit prin urmatoarea formula de calcul:

$$Mentenanta = \frac{M1}{M2} \times 100$$

Unde: M1= numarul defectelor corectate

M2= numarul total al defectelor

In cazul aplicatiei noastre putem calcula in functie de aceasta formula care este procentul de mentenanta sau mai bine zis, cat de usor sau greu este de intretinut aplicatia de modelare si generare automata a orarelor. Luand in calcul un numar total de 283 de defecte de la inceperea dezvoltarii aplicatiei pana in prezent din care doar 28 de defecte au ramas necorectate fiind considerate minore si fara impact in buna functionare a aplicatiei, va rezulta un numar de 255 defecte corectate. Aplicand formula de mai sus, va rezulta un coeficient de mentenanta de 90%.

$$Mentenanta = \frac{255}{283} \times 100 = 90\%$$

Procentul rezultat in urma aplicarii formulei va semnifica gradul de usurinta in intretinerea aplicatiei si oferirea de suport dupa ce aceasta va fi lansata, fiind utilizata de

utilizatori normali si totodata cat de rapid se vor putea rezolva potențialele defecte ce vor putea aparea in faza de utilizare.

6.2.2.3 Testabilitatea

Reprezinta raportul dintre numarul scenariilor de test executate si numarul total de scenarii de test create fiind exprimat in procente si definit prin urmatoarea formula de calcul:

$$Testabilitatea = \frac{T1}{T2} \times 100$$

Unde: T1= numarul scenariilor de test executate

T2= numarul total de scenarii de test create

In cazul aplicatiei noastre putem calcula in functie de aceasta formula care este procentul de testabilitate sau mai bine zis cat de bine sau mai putin bine a fost testata aplicatia de modelare si generare automata a orarelor putandu-ne astfel sa ne dam seama daca exista riscul aparitiei unor defecte majore atunci cand aplicatia se va afla in utilizare. Luand in calcul un numar total de aproximativ 84 scenarii de test create de la finalizarea dezvoltarii aplicatiei pana in prezent din care toate au fost executate 84, va rezulta acoperire de tastare a aplicatiei de 100%.

$$Testabilitatea = \frac{84}{84} \times 100 = 100 \%$$

Procentul rezultat in urma aplicarii formulei va semnifica cat de bine a fost testata aplicatia de modelare si generare automata a orarelor putandu-ne astfel sa ne dam seama daca exista riscul aparitiei unor defecte majore atunci cand aplicatia va fi lansata, fiind utilizata de utilizatori normali.

6.3 Rezultate obtinute

6.3.1 Rezultate curente vs asteptari

Testele demonstreaza destul de clar faptul ca intrari specifice conduc la rezultate specifice. Metodele formale demonstreaza logic faptul ca toate intrarile care indeplinesc preconditii definite produc iesiri care satisfac postconditii definite.

Fiabilitatea aplicatiei de modelare si generare a orarelor a scos la iveala faptul ca un numar de 2800 rulari ale aplicatiei au fost efectuate cu succes. Aplicand formula din capitolul anterior a rezultat un coeficient de fiabilitate de 93%, in contextul in care s-a dorit procentul maxim. Cu toate ca, in ceea ce priveste fiabilitatea oricarei aplicatii, este normal sa tindem spre absolut, diferența de doar 7 procente intre rezultat si asteptari este una imbucuratoare.

Mantenanta aplicatiei de modelare si generare a orarelor a dovedit faptul ca un numar de 255 de defecte existente au fost corectate cu succes. Aplicand formula din capitolul anterior a rezultat un coeficient de mantenanta de 90%, in contextul in care s-a dorit un procent minim de 95%.

Din studii de caz generale pe alte categorii de aplicatii software, in ceea ce priveste mantenanta se astepta ca acesta sa fie cat mai apropiata de valoare maxima avand ca scop reducerea costurilor post implementare. Cu toate acestea diferenta minima rezultata in cazul aplicatiei noastre este una acceptabila, ce poate fi imbunatatita cu usurinta in viitor.

Testabilitatea aplicatiei de modelare si generare a orarelor a depasit asteptarile initiale dovedind ca metodele formale aplicate in testarea aplicatiei au avut succes in specificarea si verificarea functionalitatilor. Astfel, un numar de 84 de scenarii existente au fost testate cu succes.

Aplicand formula din capitolul anterior a rezultat o acoperire in ceea ce priveste testarea aplicatiei de 100% in contextul in care s-a dorit un procent minim de 90%, probabil si datorita timpului scurt dedicat procesului de testare.

Cu toate acestea, asteptarile au fost depasite si s-a reusit performanta acoperiri scenariilor de testare in totalitate reusind sa reducem riscul ulterior al aparitiei defectelor.

7 Concluzii

Realizarea aplicatiei ce reprezinta latura practica a lucrarii "Modelarea si generarea automata de orare in cadrul unei facultati" a atras dupa sine o serie de beneficii atat personale, cat mai ales profesionale din punctul meu de vedere.

Pe parcursul dezvoltarii am invatat si aprofundat principiile programarii orientate obiect, un concept puternic pe care se bazeaza toate aplicatiile dezvoltate in prezent, dar am inteles intr-un mod mult mai clar principiile de proiectare si gestionare ale bazelor de date, fiind astfel acum mult mai pregatita din punct de vedere profesional in ceea ce priveste analiza, proiectarea si implementarea unei aplicatii care sa includa comunicarea intre mai multe componente.

Spre finalul dezvoltarii proiectelor am avut ocazia de a invata si concepte legate de testarea aplicatiilor, ce sunt extreme de importante in sensul realizarii unei aplicatii robuste si stabile, lipsita de erori si conflicte ce pot afecta functionarea ei, producand astfel neplaceri utilizatorilor sai.

Ca si elemente principale ale dezvoltarii aplicatiei la care s-a facut referire pe tot parcursul lucrarii de fata se pot identifica algoritmul de modelare al orarelor, dar si cel de generare a acestora, algoritmi fara de care functionarea corecta a aplicatiei ar fi pusa in pericol. Aceste doi algoritmi au ajutat la organizarea construirii unor orare lipsite de date redundante, date corecte si consistente din punct de vedere al utilizatorilor, dar si din punct de vedere al programelor universitare. Proiectarea acestor algoritmi a fost benefica pentru mine, ajutandu-ma sa construiesc sistemul pe care mi l-am dorit, insa este folositoare si in revolutionarea modului in care sunt construite in prezent orarele universitare, in sensul ca pun in prim plan atat cadrele didactice, care isi pot creea orarul dupa propriile preferinte, insa au in vedere si studentii, avand grija sa dea nastere unui orar lipsit de intereruperi mari si materii ce nu sunt de interes.

O alta componenta extrem de importanta in ceea ce priveste dezvoltarea aplicatiei este reprezentata de crearea interfetelor grafice ce faciliteaza interacciunea utilizatorului cu sistemul, aplicatia de fata fiind dedicata cadrelor didactice, studentilor, dar si personalului administrativ din cadrul unei facultati. Fara utilizatori aplicatia ar fi lipsita de date, generarea rapoartelor fiind inutila in lipsa factorului uman.

Exista cateva imbunatatiri ce ar putea fi dezvoltate in viitor pentru a face aplicatia mai utila in cadrul unei facultati, dar si mai rapida si prietenoasa pentru utilizatorii sai. In ceea ce priveste utilitatea, putem sa vorbim despre imbunatatiri legate de modelarea orarelor de catre studenti, in sensul ca acestia ar putea sa alega pe viitor laboratoarele la care doresc sa participe, insa nu din punct de vedere al programei, ci din punct de vedere al intervalelor orare.

Cu alte cuvinte, studentii vor putea avea la dispozitie o lista cu sloturi orare in care se tin ore de laborator pentru materiile din programa lor, iar ei pot alege in limita locurilor disponibile ora la care doresc sa se prezinte la laborator. Astfel, se eficientizeaza si mai mult orarul din punct de vedere al factorului temporal, atat pentru studenti, cat si pentru cadrele didactice, existand o continuitate in ceea ce priveste prezenta acestora la facultate.

Daca ne referim la imbunatitiri ale performantelor, cu siguranta aplicatia poate beneficia de versiuni viitoare care sa prezinte algoritmi mult mai eficienti atat din punct de vedere al timpului de executie, cat si din punct de vedere al spatiului de memorie folosit in timpul rularii. Partea grafica a software-ului, poate, de asemenea, sa beneficieze de perfectionari in sensul ca ferestrele afisate utilizatorului pot fi personalizate mult mai prietenos si intuitiv, astfel incat utilizatorii sa poata interactiona mai usor cu aplicatia.

Cunoscand aceste posibile retusuri ale aplicatiei, as fi bucuroasa sa pot sa dezvolt in continuare acest proiect in vederea utilizarii lui in realitate in cadrul unei facultati.

Posibilitatea de a ajuta in mod practic la implementarea cu succes a prezentei lucrari in viata reala, ajutand astfel atat cadrele didactice ce au contribuit in mod direct pe durata anilor de studiu la dezvoltarea mea profesionala, cat si comunitatea de studenti din care am facut parte si care a marcat in mod pozitiv formarea mea ca individ, ar putea constitui continuarea prezentei lucrari si de ce nu, pentru ca in permanenta este nevoie de imbunatatire, imbunatatirea modului de organizare al Universitatii Politehnice din Bucuresti in ceea ce priveste modelarea si generarea orarelor unei facultati.

Bibliografie

- [1] Arpan Kumar Kar and MP Gupta, „How to make a Smart Campus – Smart Campus Programme in IIT Delhi”, Technical Report, October 2015.
- [2] Institutul National de Statistica, Accesul Populatiei la Tehnologia Informatiilor si Comunicatiilor, INS, 2017.
- [3] Alberto Rodrigues da Silva, „Model-driven engineering: A survey support by the unified conceptual model”, 6 June 2015.
- [4] Anca Ionita, Curs Ingineria Sistemelor de Programe, Facultatea de Automatica si Calculatoare, UPB, 2016.
- [5] OMG, OMG Unified Modeling Language TM (OMG UML), Version 2.5.1, December 2017, disponibil la www.omg.org.
- [6] Dorin Carstoiu, Curs Baze de Date, Facultatea de Automatica si Calculatoare, UPB, 2016.
- [7] Hector Garcia-Molina, Jeffrey D. Ullman and Jennifer Widom, „Database Systems: The Complete Book”, Upper Saddle River, New Jersey: Prentice Hall, 2001.
- [8] Mihai Caramihai, Curs Programare Orientata pe Obiecte, Facultatea de Automatica si Calculatoare, UPB, 2015.
- [9] Alexandra Cernian si Andrei Hossu, Curs Aplicatii Web si Java, Facultatea de Automatica si Calculatoare, 2016.
- [10] Mihai Caramihai, Curs Structuri de Date si Algoritmi, Facultatea de Automatica si Calculatoare, UPB, 2014.
- [11] Romica Trandafir, Modele si Algoritmi de Optimizare, Editura Agir, Bucuresti, 2004.

Surse online:

- [12] Wikipedia, enciclopedia libera, (2007). Educatie. [online] Disponibil la: <https://ro.wikipedia.org/wiki/Educa%C8%9Bie> [Accesat la: 13 Iun 2018]
- [13] Wikipedia, enciclopedia libera, (2009). Educatia in Romania. [online] Disponibil la:https://ro.wikipedia.org/wiki/Educa%C8%9Bia_%C3%AEn_Rom%C3%A2nia [Accesat la: 13 Iun 2018]
- [14] Wikipedia, enciclopedia libera, (2009). Tehnologia informatiei. [online] Disponibil la: https://ro.wikipedia.org/wiki/Tehnologia_informa%C8%9Biei [Accesat la: 13 Iun 2018]
- [15] Edutimer, (2009). Cybrosys, Edutimer. [online] Disponibil la: <http://www.edutimer.com/Default.aspx> [Accesat la: 13 Iun 2018]
- [16] Capterra, (2007). School Management Software. [online] Disponibil la: <https://www.capterra.com/p/45768/School-Management-Software/> [Accesat la: 13 Iun 2018]
- [17] Ivan, I. si Popescu M., Metrici software [online] Disponibil la: <http://www.software-metrics.ase.ro/articole/METRICI%20%20SOFTWARE.htm> [Accesat la: 18 Iun 2018]

Anexa 1. Secvențe de cod

Algoritmul de modelare al orarelor

```
protected void adaugaIntervalNou() {
    // Declarare si initializare interval
    Interval_orar i = new Interval_orar();
    i.setMaterie(idmaterie(PanelPreferinte.retineMaterie(),
    Login.retineProfesor()));
    i.setOra_start(Integer.valueOf(oraaleasa));
    i.setOra_stop(Integer.valueOf(oraaleasa)+Integer.valueOf(durata(PanelPr
    eferinte.retineMaterie(), Login.retineProfesor())));
    if(preferintaaleasa.equals("Favorit")) i.setPreferinta(1);
    else if(preferintaaleasa.equals("Indiferent")) i.setPreferinta(2);
    else i.setPreferinta(3);
    i.setZiua(PanelPreferinte.retineZi());
    i.setSerie(SerieDB.idSerie(PanelPreferinte.retineSerie()));

    // Calculare sali de capacitate potrivita in raport cu numarul de studenti ai
    // seriei
    sali=
    SalaDB.listasali(SerieDB.studSerie(PanelPreferinte.retineSerie()));
    System.out.println("numarul de sali gasite este:" + sali.size() );
    jloop: for(int j = 0; j < sali.size(); j++){
        setat = false;

        // Verificare disponibilitate sala in intervalul orar alesint
        ok=Interval_orarDB.salalibera(sali.get(j).getId(),
        Integer.valueOf(oraaleasa), Integer.valueOf(oraaleasa) +
        Integer.valueOf(durata(PanelPreferinte.retineMaterie(),
        Login.retineProfesor())), PanelPreferinte.retineZi(),
        Plan_invatamantDB.semestrumaterie(MaterieDB.denumireMaterie(i.getMateria()),i
        .getSerie()));
        System.out.println(ok);
        if(ok == 0) {i.setSala(sali.get(j).getId());
            setat = true;}
        else {setat = false;
        }
        if(setat == true) {break jloop;
        }
    }

    // Verificare disponibilitate cadre didactice, interval orar liber si
    // verificare existenta materie in orar
    intervale = Interval_orarDB.filtreazaIntervale(Login.retineProfesor(),
    1, i.getOra_start(), i.getOra_stop(), i.getZiua(),
    Plan_invatamantDB.semestrumaterie(MaterieDB.denumireMaterie(i.getMateria()),i
    .getSerie());
    intervale2 = Interval_orarDB.filtreazaIntervale2(1,
    MaterieDB.denumireMaterie(i.getMateria()), i.getSerie(),
    Plan_invatamantDB.semestrumaterie(MaterieDB.denumireMaterie(i.getMateria()),
    i.getSerie()),Plan_invatamantDB.anmaterie(MaterieDB.denumireMaterie(i.getMate
    ria()),i.getSerie()));
    intervale3 =
    Interval_orarDB.selecteazaIntervale(ZiDB.denumire(i.getZiua()),
```

```

SerieDB.denumire(i.getSeria()),
Plan_invatamantDB.semestruMaterie(MaterieDB.denumireMaterie(i.getMateria()),
i.getSeria()),
Plan_invatamantDB.anmaterie(MaterieDB.denumireMaterie(i.getMateria()),
i.getSeria()));
kloop: for(int k = 0; k<intervale3.size(); k++) {
    Interval_orar inter = intervale3.get(k);
    imposibil = false;
    if(inter.getPreferinta() == 1 && i.getOra_start() ==
inter.getOra_start() )
        {imposibil = true;
        }
    else if((inter.getPreferinta() == 1 && i.getOra_start() != inter.getOra_start()) &&
            (i.getOra_start() > inter.getOra_start() &&
             i.getOra_start() < inter.getOra_stop()))
        {imposibil = true;
        System.out.println("diana");
        }
    else if((inter.getPreferinta() == 1 && i.getOra_start() != inter.getOra_start()) &&
            i.getOra_stop() > inter.getOra_start() &&
            i.getOra_stop() < inter.getOra_stop())
        imposibil = true;
    else imposibil = false;
    if (imposibil){break kloop;
    }
}
}

// Configurare mesaje de eroare in functie de conflictul intampinat si
configurare mesaj de confirmare
if(imposibil == false && i.getPreferinta() ==1){
    if((intervale.size() == 0 && intervale2.size() == 0)){
        rezultat = Interval_orarDB.adaugaInterval(i);
    }
    else if(intervale.size() != 0 && intervale2.size() == 0)
        JOptionPane.showMessageDialog(null,
            "Datele introduse nu sunt valide.\n"
            + "Ati selectat deja acest interval
pentru: " + Interval_orarDB.seriasimaterie(Login.retineProfesor(),
i.getPreferinta(), i.getOra_start(), i.getOra_stop(), i.getZiua()),
            "Error",
            JOptionPane.ERROR_MESSAGE);
    else if(intervale.size() == 0 && intervale2.size() != 0)
        {JOptionPane.showMessageDialog(null,
            "Datele introduse nu sunt valide.\n"
+ "Seria " + PanelPreferinte.retineSerie() + " are materia " +
PanelPreferinte.retineMaterie() + " alocata pentru " +
interval_orarDB.ziuasiintervalul(1, MaterieDB.denumireMaterie(i.getMateria()),
i.getSeria()),
            "Error",
            JOptionPane.ERROR_MESSAGE); }
    else {JOptionPane.showMessageDialog(null,
            "Datele introduse nu sunt valide.\n"
+ "Seria " + PanelPreferinte.retineSerie() + " are materia " +
PanelPreferinte.retineMaterie() + " alocata pentru " +

```

```

Interval_orarDB.ziuasiintervalul(i.getPreferinta(),
MaterieDB.denumireMaterie(i.getMateria()), i.getSeria())
+"\\n Ati selectat deja acest interval pentru: " +
Interval_orarDB.seriasimateria(Login.retineProfesor(), i.getPreferinta(),
i.getOra_start(), i.getOra_stop(), i.getZiua()),
"Error",
JOptionPane.ERROR_MESSAGE);
}
else if(imposibil == false && (i.getPreferinta() == 2 ||
i.getPreferinta() == 3)) rezultat = Interval_orarDB.adaugaInterval(i);
else JOptionPane.showMessageDialog(null,
"Datele introduse nu sunt valide.\n" +
"Nu se poate seta acest interval deoarece se suprapune cu alte materii
setate.",
"Error",
JOptionPane.ERROR_MESSAGE);
String mesaj = new String();
if (rezultat != 0) {
mesaj = "Preferinta a fost adaugat cu succes!";
SwingUtilities.getRoot(this).dispatchEvent(new
WindowEvent((Window) SwingUtilities.getRoot(this),
WindowEvent.WINDOW_CLOSING));
JOptionPane.showMessageDialog(this, mesaj);
}
}

```

Algoritmul de generare orare

```

protected void generareOrare(){
// Preluare facultate si semestru din interfata grafica
// Construire nume cale la care se vor salva orarele
String facultate = denumireFacultate.getText().toString().toUpperCase();
System.out.println(facultate);
String cale = SalvareEXCEL.creareDirectorFacultate(facultate);
System.out.println(cale);
int semestru = Integer.valueOf(numarulSemestrului.getText().toString());
System.out.println(semestru);

// Selectare ani de studiu pentru facultatea introdusa
ani = AnDB.listaani(facultate);
if(ani.size()!=0){
for(int i = 0; i< ani.size(); i++){
// Selectare lista de serii pentru fiecare an de studiu al facultatii
serii = SerieDB.listaserii(ani.get(i).getId());
for(int j = 0; j< serii.size(); j++){
// Obtine materiile din planul de invatamant si orarul actual al seriei
materii = Plan_invatamantDB.listaMaterii(serii.get(j).getId(),
ani.get(i).getId(), semestru);
orar = GenerareOrarDB.orar(serii.get(j).getDenumire(), semestru,
ani.get(i).getId(), facultate);
for(int k = 0; k < materii.size(); k++){
int m = materii.get(k);
// Verifica prezenta fiecarei materii din planul de invatamant in
orar
boolean gasit = false;
for(int l = 0; l<orar.size(); l++)
}
}
}
}
}

```

```

        if(m == orar.get(1).getMateria()) gasit = true;
        // Pentru fiecare materie ce nu este gasita in interval se
calculeaza primul interval orar
        // liber atat din punct de vedere al cadrului didactic, cat si al
studentilor si al salilor de cursuri
        // Cand se gaseste un interval liber corespunzator se introduce
materia in orarul seriei automat daca aceasta
        // nu a fost setata anterior
        if (gasit == false) {
            if(orar.get(0).getZiua() == 1 && orar.get(0).getOra_start()==8 &&
orar.get(0).getOra_stop() + MaterieDB.durataMaterie(m) <=
orar.get(1).getOra_start() && orar.get(1).getZiua() ==1)
                {System.out.println("Am intrat pe aici");
                sali = SalaDB.listasali(serii.get(j).getNr_studenti());
                tloop: for(int t = 0; t < sali.size(); t++){
                    boolean setat = false;
                    int ok = Interval_orarDB.salalibera(sali.get(t).getId(),
orar.get(0).getOra_stop(), orar.get(0).getOra_stop() +
MaterieDB.durataMaterie(m), 1, semestru);
                    System.out.println(ok);
                    if(ok == 0) {
                        if(Interval_orarDB.filtreazaIntervale(MaterieDB.profesor(m),
1, orar.get(0).getOra_stop() , orar.get(0).getOra_stop() +
MaterieDB.durataMaterie(m), 1, semestru).size() ==0)
                            {Interval_orar inter =new Interval_orar();
                            inter.setMateria(m);
                            inter.setOra_start(orar.get(0).getOra_stop());
                            inter.setOra_stop(orar.get(0).getOra_stop() +
MaterieDB.durataMaterie(m));
                            inter.setPreferinta(1);
                            inter.setSala(sali.get(t).getId());
                            inter.setSeria(serii.get(j).getId());
                            inter.setZiua(1);
                            Interval_orarDB.adaugaInterval(inter);
                            setat = true;}
                        else setat = false;}
                    else {setat = false;}
                    if(setat == true) {break tloop;}}
                }
            else if(orar.get(0).getZiua() == 1 && orar.get(0).getOra_start()!=8 &&
8 + MaterieDB.durataMaterie(m) <= orar.get(0).getOra_start() &&
orar.get(1).getZiua() ==1)
                {sali = SalaDB.listasali(serii.get(j).getNr_studenti());
                tloop: for(int t = 0; t < sali.size(); t++){
                    boolean setat = false;
                    int ok = Interval_orarDB.salalibera(sali.get(t).getId(), 8,
8 + MaterieDB.durataMaterie(m), 1, semestru);
                    System.out.println(ok);
                    if(ok == 0) {
                        if(Interval_orarDB.filtreazaIntervale(MaterieDB.profesor(m), 1, 8
, 8 + MaterieDB.durataMaterie(m), 1, semestru).size() ==0)
                            {Interval_orar inter =new Interval_orar();
                            inter.setMateria(m);
                            inter.setOra_start(8);
                            inter.setOra_stop(8 + MaterieDB.durataMaterie(m));
                            inter.setPreferinta(1);
                            inter.setSala(sali.get(t).getId());}
                    }
                }
            }
        }
    }
}

```

```

        inter.setSeria(serii.get(j).getId());
        inter.setZiua(1);
        Interval_orarDB.adaugaInterval(inter);
        setat = true;
    else setat = false;
    else {setat = false;}
    if(setat == true) {break tloop;
}
else if(orar.get(0).getZiua() == 1 && orar.get(0).getOra_start() !=8 &&
8 + MaterieDB.durataMaterie(m) > orar.get(0).getOra_start())
{sali = SalaDB.listasali(serii.get(j).getNr_studenti());
tloop: for(int t = 0; t < sali.size(); t++){
boolean setat = false;
for(int n=orar.get(0).getOra_stop(); n<18;
n+=MaterieDB.durataMaterie(m)
    {int ok = Interval_orarDB.salalibera(sali.get(t).getId(), n, n +
MaterieDB.durataMaterie(m), 1, semestru);
    System.out.println(ok);
    if(ok == 0) {
        if(Interval_orarDB.filtreazaIntervale(MaterieDB.profesor(m), 1, n
, n + MaterieDB.durataMaterie(m), 1, semestru).size() ==0 &&
Interval_orarDB.filtreazaIntervale3(serii.get(j).getId(), 1, n , n +
MaterieDB.durataMaterie(m), 1, semestru).size() ==0
            {Interval_orar inter =new Interval_orar();
            inter.setMateria(m);
            inter.setOra_start(n);
            inter.setOra_stop(n + MaterieDB.durataMaterie(m));
            inter.setPreferinta(1);
            inter.setSala(sali.get(t).getId());
            inter.setSeria(serii.get(j).getId());
            inter.setZiua(1);
            Interval_orarDB.adaugaInterval(inter);
            setat = true;
        else setat = false;
    }
else {setat = false;}}

if(setat == true) {break tloop;}}

else if(orar.get(0).getZiua() == 1 && orar.get(0).getOra_start() ==8 &&
orar.get(0).getOra_stop() + MaterieDB.durataMaterie(m) >=
orar.get(1).getOra_start()&& orar.get(1).getZiua() ==1)
{sali = SalaDB.listasali(serii.get(j).getNr_studenti());
tloop: for(int t = 0; t < sali.size(); t++){
boolean setat = false;
int ok = Interval_orarDB.salalibera(sali.get(t).getId(),
orar.get(1).getOra_stop(), orar.get(1).getOra_stop() +
MaterieDB.durataMaterie(m), 1, semestru);
    System.out.println(ok);
    if(ok == 0) {
        if(Interval_orarDB.filtreazaIntervale(MaterieDB.profesor(m), 1,
orar.get(1).getOra_stop() , orar.get(1).getOra_stop() +
MaterieDB.durataMaterie(m), 1, semestru).size() ==0
            {Interval_orar inter =new Interval_orar();
            inter.setMateria(m);
            inter.setOra_start(orar.get(1).getOra_stop());
            inter.setOra_stop(orar.get(1).getOra_stop() +
MaterieDB.durataMaterie(m));
```

```

        inter.setPreferinta(1);
        inter.setSala(sali.get(t).getId());
        inter.setSeria(serii.get(j).getId());
        inter.setZiua(1);
        Interval_orarDB.adaugaInterval(inter);
        setat = true;
    else setat = false;
else {setat = false;}
if(setat == true) {break tloop;}}
```



```

else if(orar.get(0).getZiua() == 1 && orar.get(0).getOra_start() == 8
&& orar.get(1).getZiua() == 2)
{ sali = SalaDB.listasali(serii.get(j).getNr_studenti());
  tloop: for(int t = 0; t < sali.size(); t++){
    boolean setat = false;
    for(int n = orar.get(0).getOra_stop(); n < 18;
n += MaterieDB.durataMaterie(m))
      int ok = Interval_orarDB.salalibera(sali.get(t).getId(), n, n +
MaterieDB.durataMaterie(m), 1, semestru);
      if(ok == 0) {
        if(Interval_orarDB.filtreazaIntervale(MaterieDB.profesor(m), 1,
n, n + MaterieDB.durataMaterie(m), 1, semestru).size() == 0)
          {System.out.println(MaterieDB.profesor(m));
          Interval_orar inter =new Interval_orar();
          inter.setMateria(m);
          inter.setOra_start(n);
          inter.setOra_stop(n + MaterieDB.durataMaterie(m));
          inter.setPreferinta(1);
          inter.setSala(sali.get(t).getId());
          inter.setSeria(serii.get(j).getId());
          inter.setZiua(1);
          Interval_orarDB.adaugaInterval(inter);
          setat = true;
        else setat = false;
      else {setat = false;}
      if(setat == true) break tloop;

    else{for(int n = orar.get(1).getOra_stop(); n < 18;
n += MaterieDB.durataMaterie(m))
      int ok = Interval_orarDB.salalibera(sali.get(t).getId(), n,
n + MaterieDB.durataMaterie(m), 2, semestru);
      if(ok == 0) {
        System.out.println("ora libera" + n);
        if(Interval_orarDB.filtreazaIntervale(MaterieDB.profesor(m),
1, n, n + MaterieDB.durataMaterie(m), 2, semestru).size() == 0)
          {System.out.println(MaterieDB.profesor(m));
          Interval_orar inter =new Interval_orar();
          inter.setMateria(m);
          inter.setOra_start(n);
          inter.setOra_stop(n + MaterieDB.durataMaterie(m));
          inter.setPreferinta(1);
          inter.setSala(sali.get(t).getId());
          inter.setSeria(serii.get(j).getId());
          inter.setZiua(2);
          Interval_orarDB.adaugaInterval(inter);
          setat = true;
        break tloop;}
```

```

        else setat = false;
    else {setat = false;}}}}
else if(orar.get(0).getZiua() != 1){
    sali = SalaDB.listasali(serii.get(j).getNr_studenti());
    tloop: for(int t = 0; t < sali.size(); t++) {
        boolean setat = false;
        for(int n = 8; n < 18; n+=MaterieDB.durataMaterie(m))
            {int ok = Interval_orarDB.salalibera(sali.get(t).getId(),
n,n + MaterieDB.durataMaterie(m), 1, semestru);
System.out.println(ok);
if(ok == 0) {
    if(Interval_orarDB.filtreazaIntervale(MaterieDB.profesor(m), 1, n,
n + MaterieDB.durataMaterie(m), 1, semestru).size() ==0)
        {Interval_orar inter =new Interval_orar();
inter.setMateria(m);
inter.setOra_start(n);
inter.setOra_stop(n + MaterieDB.durataMaterie(m));
inter.setPreferinta(1);
inter.setSala(sali.get(t).getId());
inter.setSeria(serii.get(j).getId());
inter.setZiua(1);
Interval_orarDB.adaugaInterval(inter);
setat = true;}
else setat = false;}
else {setat = false;}}
if(setat == true) {break tloop;}}
else {sali = SalaDB.listasali(serii.get(j).getNr_studenti());
tloop: for(int t = 0; t < sali.size(); t++)
    {boolean setat = false;
    for(int z = 1; z<6; z++){
        for(int n = 8; n<=18; n+=MaterieDB.durataMaterie(m) ){
            int ok =
Interval_orarDB.salalibera(sali.get(t).getId(), n,n +
MaterieDB.durataMaterie(m), z, semestru);
if(ok == 0) {
    if(Interval_orarDB.filtreazaIntervale(MaterieDB.profesor(m), 1, n,
n + MaterieDB.durataMaterie(m), z, semestru).size() ==0)
        {Interval_orar inter =new Interval_orar();
inter.setMateria(m);
inter.setOra_start(n);
inter.setOra_stop(n + MaterieDB.durataMaterie(m));
inter.setPreferinta(1);
inter.setSala(sali.get(t).getId());
inter.setSeria(serii.get(j).getId());
inter.setZiua(z);
Interval_orarDB.adaugaInterval(inter);
setat = true;}
else setat = false;}
else {setat = false;}}
if(setat == true) {break tloop;}}}}}
// Salvare orar complet in directorul corespunzator facultatii
orar2 = GenerareOrarDB.orar(serii.get(j).getDenumire(), semestru,
ani.get(i).getId(), facultate);
denumireFisier = retineSerieAnSemetru(serii.get(j).getDenumire(),
ani.get(i).getDenumire());

```

```

if(SalvareEXCEL.creareDirectorFacultate(facultate).equals("Nu am putut crea
directorul") == false) SalvareEXCEL.populeazaOrar(orar2);
System.out.println(SalvareEXCEL.populeazaOrar(orar2));
}

// Configurare mesaje de eroare sau confirmare
JOptionPane.showMessageDialog(null,
                            "Orarele au fost generate!",
                            "Information",
                            JOptionPane.INFORMATION_MESSAGE);
else JOptionPane.showMessageDialog(null,
                            "Datele introduse nu sunt valide.\n" + "Facultatea nu se
regaseste in aplicatie!",
                            "Error",
                            JOptionPane.ERROR_MESSAGE);
}

```

Anexa 2. Baza de date

Exemplu creare tabel Interval_Orar in baza de date SmartCampus

```

CREATE TABLE [dbo].[Interval_Orar](
    [id] [int] IDENTITY(1,1) NOT NULL,
    [ziua] [int] NOT NULL,
    [materia] [int] NOT NULL,
    [seria] [int] NOT NULL,
    [ora_start] [int] NOT NULL,
    [ora_stop] [int] NOT NULL,
    [preferinta] [int] NOT NULL,
    [sala] [int] NOT NULL,
    CONSTRAINT [PK_Interval_Orar_1] PRIMARY KEY CLUSTERED
(
    [id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO

ALTER TABLE [dbo].[Interval_Orar] WITH CHECK ADD CONSTRAINT
[FK_Interval_Orar_Materie] FOREIGN KEY([materia])
REFERENCES [dbo].[Materie] ([id])
GO

ALTER TABLE [dbo].[Interval_Orar] CHECK CONSTRAINT [FK_Interval_Orar_Materie]
GO

ALTER TABLE [dbo].[Interval_Orar] WITH CHECK ADD CONSTRAINT
[FK_Interval_Orar_Preferinta] FOREIGN KEY([preferinta])
REFERENCES [dbo].[Preferinta] ([id])
GO

ALTER TABLE [dbo].[Interval_Orar] CHECK CONSTRAINT
[FK_Interval_Orar_Preferinta]
GO

```

```

ALTER TABLE [dbo].[Interval_Orar] WITH CHECK ADD CONSTRAINT
[FK_Interval_Orar_Sala] FOREIGN KEY([sala])
REFERENCES [dbo].[Sala] ([id])
GO

ALTER TABLE [dbo].[Interval_Orar] CHECK CONSTRAINT [FK_Interval_Orar_Sala]
GO

ALTER TABLE [dbo].[Interval_Orar] WITH CHECK ADD CONSTRAINT
[FK_Interval_Orar_Serie] FOREIGN KEY([seria])
REFERENCES [guest].[Serie] ([id])
GO

ALTER TABLE [dbo].[Interval_Orar] CHECK CONSTRAINT [FK_Interval_Orar_Serie]
GO

ALTER TABLE [dbo].[Interval_Orar] WITH CHECK ADD CONSTRAINT
[FK_Interval_Orar_Zi] FOREIGN KEY([ziua])
REFERENCES [guest].[Zi] ([id])
GO

ALTER TABLE [dbo].[Interval_Orar] CHECK CONSTRAINT [FK_Interval_Orar_Zi]
GO

```

Exemplu creare vedere Alocari in baza de date SmartCampus

```

CREATE VIEW [guest].[Alocari]
AS
SELECT dbo.Materie.profesor AS id_profesor, dbo.Materie.id AS id_materie,
dbo.Profesor.nume, dbo.Profesor.prenume, dbo.Profesor.email,
dbo.Profesor.departament, dbo.Materie.denumire AS materie,
dbo.Materie.curs AS ore_curs
FROM dbo.Materie INNER JOIN dbo.Profesor ON dbo.Materie.profesor =
dbo.Profesor.id
GO

```

Anexa 3. Fisiere de date

Exemplu fisier XLS ce include orarul seriei AC pentru semestrul 2 Anul 3

