**PROIECT**

la disciplina

Introducere in Baze de Date

Managementul unui cabinet de medicina de familie

Ioan Cristian Belbe

Marius Cristian Dulău

An academic :2016 – 2017

PROIECT de SEMESTRU

Catedra de Calculatoare

Disciplina : Introducere in Baze de Date

Coordonator: ing. Denisa DAN

Data : 17.01.2017

Cuprins

## 1. Introducere

Introducere, argumente, scop si obiective specifice

## 2. Analiza cerintelor utilizatorilor ( Specificatiile de proiect)

* Ipoteze specifice domeniului ales pentru proiect ( cerinte, constrangeri )
* Organizare structurata( tabelar) a cerintelor utilizator
* Determinarea si caracterizarea de profiluri de utilizatori ( admin, user intern, user extern…diversi alti “actori”)

## 3. Modelul de date si descrierea acestuia

* Entitati si atributele lor ( descriere detaliata – implementarea fizica)
* Diagrama EER/UML pentru modelul de date complet
* Normalizarea datelor
* Vederi, triggere, proceduri
* Interogări MySQL

## 4. Detalii de implementare

* Descrierea funcțională a modulelor
* Manual de utilizare/instalare (diferentiat pe tipuri de actori)
* Elemente de securizare a aplicatiei
* Diagrama de clase a aplicației

5. Concluzii, limitări și dezvoltări ulterioare

6. Bibliografie

1. Introducere
2. Formularea problemei

Într-un cabinet medical lucrează 2 medici. Pacienții vor să fie consultați de un medic la care se înscriu și li se generează o fișă. Medicul dorește să își gestioneze programul prin programări telefonice. Medicul trece într-un registru de consultații pacienții pe care i-a consultat și în caz de tratament (boală) eliberează o rețetă. Deoarece programul acestuia este foarte încărcat are nevoie de personal auxiliar care să îl ajute în anumite situații. În funcție de nevoile pacienților se pot elibera adeverințe medicale, avize epidemiologice și bilete de trimitere/internare. Fiecare medic are mai mulți regiștri: bolnavi cronici, copii sub un an, femei însărcinate și chemări la domiciliu. Pentru pacienții bolnavi cronici, medicul trebuie să își noteze boala pacientului, daata examinării și observațiile constatate în legătură cu evoluția stării pacientului. În registrul de chemări la domiciliu se notează data consultului și scopul chemării. Pentru copii sub un an se rețin în registrul corespunzător vaccinul și data vaccinării. Pentru femeile însărcinate se notează în registru data consultului și observațiile legate de sarcină. Deoarece există multe boli definite în lumea medicinei se poate considera un registru care să conțină tratamente pentru bolile cunoscute.

Cabinetul dispune de un inventar pentru medicamente, aparatură medicală și obiecte sanitare.

1. Scop

Aplicația are ca scop gestionarea managementului cabinetului și păstrarea istoricului medical al pacienților. De asemenea, aplicația reduce gestionarea dosarelor fizice prin înlocuirea cu unele virtuale micșorând astfel timpul de căutare în arhivă.

1. Obietive specifice

Obiectivul este o asociere cât mai riguroasă a datelor pacientului. Acest lucru trebuie să aibă loc atât din punctul de vedere al gestiunii bazei de date, cât și al interfeței. Se urmărește crearea unui mediu interactiv de muncă prin intermediul calculatorului.

1. Analiza cerintelor utilizatorilor (Specificațiile de proiect)
2. Ipoteze specifice domeniului ales pentru proiect

Dorim realizarea unui sistem de gestiune a pacienților și istoricului lor medical, precum și al medicilor și recuzitei din cabinet. Am luat in considerare urmatoarele entități, pentru ca aplicația să acopere cât mai bine informațiile necesare unui cabinet:

* Medici: au un cont creat de către un administrator de sistem. Autentificarea medicului în aplicație se face cu ajutorul unui nume de utilizator și a parolei aferente contului de utilizator.
* Pacienți: medicul doresc să-și aibă în evidență pacienții. Pentru fiecare pacient se stochează informații legate de nume, prenume, cnp, adresă, telefon, sex, data nașterii, înălțime, masă corporală, tensiune cardiacă, glicemie și grupa sanguină.
* Registre: un medic poate avea mai multe registre în care să țină evidența anumitor tipuri de date:
* copii sub un an: vaccinul și data vaccinării
* femei însărcinate: date referitoare la consult lunar
* bolnavi cronici: boala de care suferă pacientul în cauză, data consultului și observațiile legate de starea
* chemări la domiciliu: un medic poate efectua muncă pe teren prin consultul pacienților la domicili. Se rețin în registru data și scopul chemării
* consultații ale pacienților: data consultației, diagnostic,
* programări: medicul își poate gestiona mai bine orarul prin programări ale pacienților la o anumită dată
* tratamentele bolilor: medicina este un domeniu foarte complex și orice inexactitate face diferența între însănătoșire sau agravarea bolii. În acest scop este nevoie de un “dicționar” de tratamente pentru boli
* Documente: un medic poate elibera rețete, adeverințe medicale, avize epidemiologice și bilete de trimitere la unități specializate.
* Recuzită medicală: într-un cabinet medical există aparatură medicală, obiecte igienico-sanitare, medicamente și vaccinuri.
* Personal auxiliar: poate fi o asistentă sau un medic înlocuitor

1. Organizare structurata a cerintelor

Pentru ca obiectivele impuse să fie atinse, am creat anumite tabele pentru a reține informații specifice fiecărei entități de mai sus. Prin urmare avem urmatoarele tabele fiecare cu structura prezentată în dreptul titlului:

Conturi: username: username, parola, cnp, drept\_admin

Medic: id\_medic, nume, prenume, cnp, data\_angajarii

Pacient: id\_pacient, id\_medic, nume, prenume, cnp, adresa, telefon, sex, data\_nasterii

Masuratori: id\_pacient, inaltime, masa\_corporala, grupa\_sanguina, tensiune\_cardiaca, glicemie

Personal\_auxiliar: id\_medic, nume, prenume, cnp, functia

Registru\_consultatii: id\_consultație, id\_pacient, data\_consultație, prezentare, diagnostic, tratament

Programari: id\_medic, id\_pacient, data\_programarii

Vaccin: id\_vaccin, nume\_vaccin, inventar

Registru\_tratamente: boala, tratament

Registru\_bolnavi\_cronici: id\_pacient, boala, data\_examinare, observatii

Registru\_copii\_sub\_1\_an: id\_pacient, id\_vaccin, data\_vaccinare

Registru\_femei\_insarcinate: id\_pacient, data\_examinare, observatii

Registru\_chemari\_domiciliu: id\_pacient, data\_chemare, scop

Reteta: id\_reteta, id\_medic, id\_pacient, diagnostic, lista\_medicamente

Bilet\_trimitere: id\_pacient, diagnostic, data\_trimitere, unitate\_trimitere

Adeverinte\_medicale: id\_pacient, data\_eliberare, scop

Aviz\_epidemiologic: id\_pacient, data\_eliberare, scop

Aparatura\_medicala: id\_aparat, nume\_aparat, inventar

Obiecte\_igienico\_sanitare: id\_obiect, nume\_obiect, inventar

Medicamente: id\_medicament, nume\_medicament, inventar

1. Determinarea si caracterizarea de profiluri de utilizatori

Aplicația este structurată pentru două tipuri de utilizatori: administrator și medic/personal auxiliar. Administratorul poate crea sau modifica conturi pentru medici și personalul auxiliar al acestora. Medicul și pesonalul auxiliar pot adăuga pacienți și au acces la datele persoanelor nou introduse. De asemenea, personalul medical(medici și personal auxiliar) au acces la inventarul cabinetului și are dreptul de a-l modifica. Medicii și personalul auxiliar au dreptul de a urmări istoricul medical al unui pacient prin intermediul registrelor menționate mai sus.

1. Modelul de date si descrierea acestuia
2. Entități și atributele lor

Cum am specificat mai sus, avem de-a face cu următoarele tabele: Conturi, Medici, Pacient, Masuratori, Personal\_auxiliar, Registru\_consultatii, Programari, Vaccin, Registru\_tratamente, Registru\_bolnavi\_cronici, Registru\_copii\_sub\_1\_an, Registru\_femei\_insarcinate, Registru\_chemari\_domiciliu, Reteta, Bilet\_trimitere, Adeverinte\_medicale, Aviz\_epidemiologic, Aparatura\_medicala, Obiecte\_igienico\_sanitare și Medicamente . Codul SQL de creare a acestor tabele este următorul:

CREATE TABLE IF NOT EXISTS Conturi

(

username varchar(30) not null,

parola varchar(30) not null,

cnp varchar(30) not null,

drept\_admin char not null

);

CREATE TABLE IF NOT EXISTS Medic

(

id\_medic int not null,

nume varchar(32) not null,

prenume varchar(32) not null,

cnp varchar(13) not null,

data\_angajarii date not null,

PRIMARY KEY(id\_medic)

);

CREATE TABLE IF NOT EXISTS Pacient

(

id\_pacient int not null,

id\_medic int not null,

nume varchar(32) not null,

prenume varchar(32) not null,

cnp varchar(13) not null,

adresa varchar(128) not null,

telefon varchar(10),

sex char,

data\_nasterii date,

PRIMARY KEY(id\_pacient),

CONSTRAINT pacient\_medic FOREIGN KEY(id\_medic) REFERENCES Medic(id\_medic) ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE IF NOT EXISTS Masuratori

(

id\_pacient int not null,

inaltime double,

masa\_corporala double,

grupa\_sanguina varchar(3),

tensiune\_cardiaca varchar(10),

glicemie int,

CONSTRAINT pacient\_masuratori FOREIGN KEY(id\_pacient) REFERENCES Pacient(id\_pacient) ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE IF NOT EXISTS Personal\_auxiliar

(

id\_medic int not null,

nume varchar(32) not null,

prenume varchar(32) not null,

cnp varchar(13) not null,

functia varchar(32) not null,

CONSTRAINT personal\_medic FOREIGN KEY(id\_medic) REFERENCES Medic(id\_medic) ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE IF NOT EXISTS Programari

(

id\_medic int not null,

id\_pacient int not null,

data\_programarii datetime not null,

CONSTRAINT orar\_medic FOREIGN KEY(id\_medic) REFERENCES Medic(id\_medic) ON DELETE CASCADE ON UPDATE CASCADE,

CONSTRAINT programare\_pacient FOREIGN KEY(id\_pacient) REFERENCES Pacient(id\_pacient) ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE IF NOT EXISTS Registru\_consultatii

(

id\_consultatie int not null,

id\_pacient int not null,

data\_consultatie date not null,

prezentare varchar(2) not null,

diagnostic varchar(128),

tratament varchar(128),

PRIMARY KEY(id\_consultatie),

CONSTRAINT consult\_pacient FOREIGN KEY(id\_pacient) REFERENCES Pacient(id\_pacient) ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE IF NOT EXISTS Registru\_femei\_insarcinate

(

id\_pacient int not null,

data\_examinare date not null,

observatii varchar(512),

CONSTRAINT femei\_insarcinate\_pacient FOREIGN KEY(id\_pacient) REFERENCES Pacient(id\_pacient) ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE IF NOT EXISTS Registru\_copii\_sub\_1\_an

(

id\_pacient int not null,

id\_vaccin int not null,

data\_vaccinare date not null,

CONSTRAINT copil\_pacient FOREIGN KEY(id\_pacient) REFERENCES Pacient(id\_pacient) ON DELETE CASCADE ON UPDATE CASCADE,

CONSTRAINT copil\_vaccin FOREIGN KEY(id\_vaccin) REFERENCES Vaccin(id\_vaccin) ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE IF NOT EXISTS Registru\_bolnavi\_cronici

(

id\_pacient int not null,

boala varchar(32) not null,

data\_examinare date not null,

observatii varchar(512),

CONSTRAINT boala\_pacient FOREIGN KEY(id\_pacient) REFERENCES Pacient(id\_pacient) ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE IF NOT EXISTS Registru\_tratamente

(

boala varchar(32) not null,

tratament varchar(128) not null,

PRIMARY KEY(boala)

);

CREATE TABLE IF NOT EXISTS Vaccin

(

id\_vaccin int not null,

nume\_vaccin varchar(128) not null,

inventar int not null,

PRIMARY KEY(id\_vaccin)

);

CREATE TABLE IF NOT EXISTS Adeverinte\_medicale

(

id\_pacient int not null,

data\_eliberare date not null,

scop varchar(128) not null,

CONSTRAINT adeverinta\_pacient FOREIGN KEY(id\_pacient) REFERENCES Pacient(id\_pacient) ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE IF NOT EXISTS Bilet\_trimitere

(

id\_pacient int not null,

diagnostic varchar(128) not null,

data\_trimitere date not null,

unitate\_trimitere varchar(128),

CONSTRAINT trimitere\_pacient FOREIGN KEY(id\_pacient) REFERENCES Pacient(id\_pacient) ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE IF NOT EXISTS Reteta

(

id\_reteta int not null,

id\_medic int not null,

id\_pacient int not null,

diagnostic varchar(128) not null,

lista\_medicamente varchar(256) not null,

PRIMARY KEY(id\_reteta),

CONSTRAINT reteta\_medic FOREIGN KEY(id\_medic) REFERENCES Medic(id\_medic) ON DELETE CASCADE ON UPDATE CASCADE,

CONSTRAINT reteta\_pacient FOREIGN KEY(id\_pacient) REFERENCES Pacient(id\_pacient) ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE IF NOT EXISTS Registru\_chemari\_domiciliu

(

id\_pacient int not null,

data\_chemare date not null,

scop varchar(128) not null,

CONSTRAINT chemare\_pacient FOREIGN KEY(id\_pacient) REFERENCES Pacient(id\_pacient) ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE IF NOT EXISTS Medicamente

(

id\_medicament int not null,

nume\_medicament varchar(32) not null,

inventar int not null,

PRIMARY KEY(id\_medicament)

);

CREATE TABLE IF NOT EXISTS Obiecte\_igienico\_sanitare

(

id\_obiect int not null,

nume\_obiect varchar(32) not null,

inventar int not null,

PRIMARY KEY(id\_obiect)

);

CREATE TABLE IF NOT EXISTS Aparatura\_medicala

(

id\_aparat int not null,

nume\_aparat varchar(32) not null,

inventar int not null,

PRIMARY KEY(id\_aparat)

);

CREATE TABLE IF NOT EXISTS Aviz\_epidemiologic

(

id\_pacient int not null,

data\_eliberare date not null,

scop varchar(128) not null,

CONSTRAINT aviz\_pacient FOREIGN KEY(id\_pacient) REFERENCES Pacient(id\_pacient) ON DELETE CASCADE ON UPDATE CASCADE

);

1. Diagrama EER/UML pentru modelul de date complet

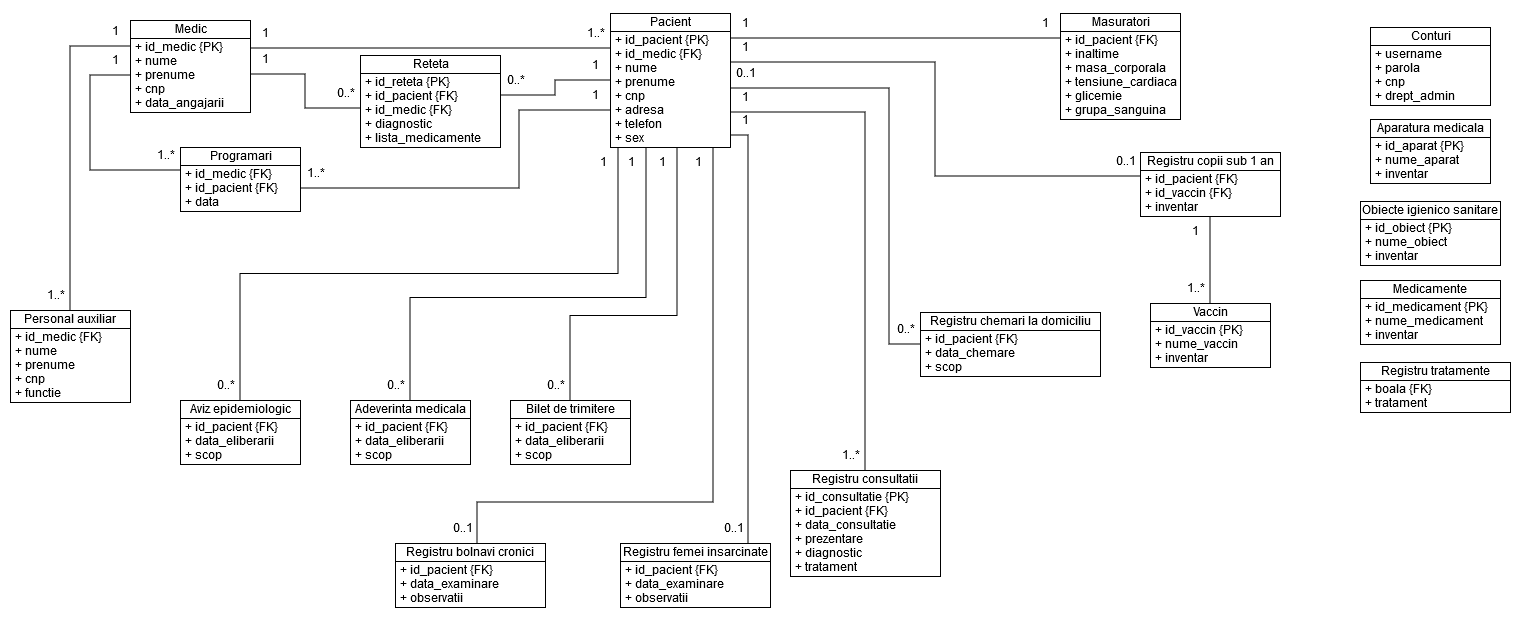


Diagrama UML pentru modelul de date complet

Fig. 1. Diagrama UML

1. Normalizarea datelor

Prima formă normală este o formă normală utilizată în normalizarea bazelor de date. Prima formă normală exclude posibilitatea existentei grupurilor repetitive cerând ca fiecare câmp într-o bază de date să cuprindă numai o valoare atomică. De asemenea, prima formă normală cere și ca fiecare înregistrare să fie definită astfel încăt să fie identificată în mod unic prin intermediul unei chei primare.

Baza noastră de date folosesște chei primare pentru a identifica celelalte cămpuri în mod unic și nu există grupuri repetitive.

A doua formă normală cere ca toate elementele unei tabele să fie dependente funcțional de totalitatea cheii primare.Dacă unul sau mai multe elemente sunt dependente funcțional numai de o parte a cheii primare, atunci ele trebuie sa fie separate in tabele diferite. Dacă tabela are o cheie primara formată din numai un atribut, atunci ea este automat în 2NF (a 2-a formă normală). Unele tabele ale bazei noastre de date au chei străine care sunt dependente de cheile primare ale altor tabele.

În a treia formă normală toate atributele non-chei ale unei relații depind numai de chei candidate ale acelei relații. Toate atributele non-cheie sunt (trebuie sa fie) mutual independente. Un atribut non-cheie care este mutual independent este cnp-ul pacientului.

1. Vedetri, triggere, proceduri

Vederi

*Vedere pentru numele, prenumele, cnp-ul copiilor și vaccinurilor care li s-au administrat la o anumită dată*

CREATE VIEW `VaccinCopil` AS SELECT Pacient.nume AS Nume, Pacient.prenume AS Prenume, Pacient.cnp AS CNP, Vaccin.nume\_vaccin AS Vaccin, registru\_copii\_sub\_1\_an.data\_vaccinare AS `Data vaccinare`

FROM Pacient, Vaccin, Registru\_copii\_sub\_1\_an

WHERE Pacient.id\_pacient = registru\_copii\_sub\_1\_an.id\_pacient AND vaccin.id\_vaccin = registru\_copii\_sub\_1\_an.id\_vaccin;

*Vedere pentru numele, prenumele, cnp-ul femeilor însărcinate și data examinării*

CREATE VIEW RegistruMame AS SELECT Pacient.nume AS Nume, Pacient.prenume AS Prenume, Pacient.cnp AS CNP, registru\_femei\_insarcinate.data\_examinare AS `Data examinare`, registru\_femei\_insarcinate.observatii AS Observatii

FROM Pacient, Registru\_femei\_insarcinate

WHERE Pacient.id\_pacient = registru\_femei\_insarcinate.id\_pacient;

*Vedere pentru numele, prenumele, cnp-ul pacientului și indicele de masă corporală*

CREATE VIEW `Tabel bmi` AS SELECT Pacient.nume AS Nume, Pacient.prenume AS Prenume, Pacient.cnp AS CNP, Masuratori.inaltime AS Inaltime, Masuratori.masa\_corporala,

CASE

WHEN masuratori.masa\_corporala/pow(masuratori.inaltime, 2) < 18.5 THEN 'Subponderal'

WHEN masuratori.masa\_corporala/pow(masuratori.inaltime, 2) BETWEEN 18.5 AND 24.99 THEN 'Greutate normala'

WHEN masuratori.masa\_corporala/pow(masuratori.inaltime, 2) BETWEEN 25 AND 29.99 THEN 'Supraponderal'

WHEN masuratori.masa\_corporala/pow(masuratori.inaltime, 2) BETWEEN 30 AND 34.99 THEN 'Obezitate gradul I'

WHEN masuratori.masa\_corporala/pow(masuratori.inaltime, 2) BETWEEN 35 AND 39.99 THEN 'Obezitate gradul II'

ELSE 'Obezitate morbida'

END AS BMI FROM Pacient, Masuratori WHERE Pacient.id\_pacient = Masuratori.id\_pacient;

Triggere

*Trigger pentru validarea codului numeric personal*(se folosește procedură pentru validarea codului numeric personal în caz că are 13 cifre)

DELIMITER $

CREATE TRIGGER cnp\_corect BEFORE INSERT ON Pacient

FOR EACH ROW

BEGIN

SET @raspuns = null;

IF NOT(char\_length(NEW.cnp) = 13) THEN

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'CNP-ul introdus nu este valid, nu are 13 cifre.';

ELSE

CALL cnp\_valid(NEW.cnp, @raspuns);

IF(@raspuns = 'N') THEN

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'Acest cnp este invalid.';

ELSE

IF NEW.cnp IN(SELECT cnp FROM Pacient) THEN

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'CNP-ul introdus exista deja in baza de date.';

END IF;

END IF;

END IF;

END$

DELIMITER ;

*Trigger pentru lungimea numărului de telefon*

DELIMITER $

CREATE TRIGGER telefon\_corect BEFORE INSERT ON Pacient

FOR EACH ROW

BEGIN

IF NOT(char\_length(NEW.telefon) = 10 OR char\_length(NEW.telefon) = 0) THEN SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'Numarul de telefon introdus nu este valid, nu are 10 cifre.';

ELSE

IF NEW.telefon IN(SELECT telefon FROM Pacient) THEN

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'Numarul de telefon introdus exista deja in baza de date.';

END IF;

END IF;

END$

DELIMITER ;

*Trigger pentru grupă sanguină corectă*

DELIMITER $

CREATE TRIGGER grupa\_sange\_corecta BEFORE INSERT ON Masuratori

FOR EACH ROW

BEGIN

IF NEW.grupa\_sanguina NOT IN('01', 'A2', 'B3', 'AB4') THEN

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'Grupa de sange incorecta. Grupele de sange sunt: 01, A2, B3, AB4';

END IF;

END$

DELIMITER ;

*Trigger pentru ca un medic să nu elibereze 2 adeverințe pentru același pacient*

DELIMITER $

CREATE TRIGGER adeverinte\_data BEFORE INSERT ON Adeverinte\_medicale

FOR EACH ROW

BEGIN

IF NEW.data\_eliberare IN(SELECT data\_eliberare FROM adeverinte\_medicale) AND NEW.id\_pacient IN(SELECT id\_pacient FROM adeverinte\_medicale) THEN

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'Adeverinta medicala a fost deja eliberata pe aceasta data.';

END IF;

END$

DELIMITER ;

*Trigger pentru a detecta dacă la cabinet lucrează deja 2 medici*

DELIMITER $

CREATE TRIGGER medici\_2 BEFORE INSERT ON medic

FOR EACH ROW

BEGIN

IF (SELECT COUNT(id\_medic) FROM Medic) = 2 THEN

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'In acest cabinet pot lucra doar 2 medici.';

END IF;

END$

DELIMITER ;

Proceduri

*Procedură pentru determinarea sexului din cnp*

DELIMITER $

CREATE PROCEDURE sex\_din\_cnp(IN cnp varchar(13), OUT sex\_pacient char)

BEGIN

IF SUBSTRING(cnp, 1, 1) IN('1', '5') THEN

SET sex\_pacient = 'M';

ELSE

IF SUBSTRING(cnp, 1, 1) IN('2', '6') THEN

SET sex\_pacient = 'F';

END IF;

END IF;

END$

DELIMITER ;

*Procedură pentru determinarea zilei de naștere din cnp*

DELIMITER $

CREATE PROCEDURE data\_nastere\_din\_cnp(IN cnp varchar(13), OUT data\_nastere date)

BEGIN

DECLARE an varchar(4);

DECLARE data\_n varchar(16);

IF SUBSTRING(cnp, 1, 1) IN('1', '2') THEN

SET an = concat('19', SUBSTRING(cnp, 2,2));

ELSE

IF SUBSTRING(cnp, 1, 1) IN('5', '6') THEN

SET an = concat('20', SUBSTRING(cnp, 2, 2));

END IF;

END IF;

SET data\_n = concat(an, '-', SUBSTRING(cnp, 4, 2), '-', SUBSTRING(cnp, 6, 2));

SET data\_nastere = STR\_TO\_DATE(data\_n, "%Y-%m-%d");

END$

DELIMITER ;

*Procedură pentru inserare pacieți noi*

DELIMITER $

CREATE PROCEDURE Pacient\_nou(IN id\_medic int, IN nume varchar(32), IN prenume varchar(32), IN cnp varchar(13), IN adresa varchar(128), IN telefon varchar(10))

BEGIN

SET @idPacient = NULL;

SELECT @idPacient := MAX(id\_pacient) FROM Pacient;

SET @idPacient := @idPacient + 1;

SET @sex\_pacient = NULL;

CALL sex\_din\_cnp(cnp, @sex\_pacient);

SET @data\_n\_cnp = NULL;

CALL data\_nastere\_din\_cnp(cnp, @data\_n\_cnp);

INSERT INTO pacient VALUES(@idPacient, id\_medic, nume, prenume, cnp, adresa, telefon, @sex\_pacient, @data\_n\_cnp);

END$

DELIMITER ;

*Procedură pentru validarea unui cod numeric personal*

DELIMITER $

CREATE PROCEDURE cnp\_valid(IN cnp\_dat varchar(13), OUT decizie char)

BEGIN

DECLARE x INT DEFAULT 1;

DECLARE suma INT DEFAULT 0;

DECLARE cifra\_control INT DEFAULT 0;

DECLARE numar\_control varchar(12) DEFAULT "279146358279";

SET decizie = 'N';

WHILE x <= 12 DO

SET suma = suma + CONVERT(SUBSTRING(cnp\_dat, x, 1), UNSIGNED) \* CONVERT(SUBSTRING(numar\_control, x, 1), UNSIGNED);

SET x = x + 1;

END WHILE;

SET cifra\_control = suma % 11;

IF(cifra\_control = 10) THEN

SET cifra\_control = 1;

END IF;

IF(cifra\_control = CONVERT(SUBSTRING(cnp\_dat, 13, 1), UNSIGNED)) THEN

SET decizie = 'Y';

END IF;

END$

DELIMITER ;

*Procedură pentru autentificare*

DELIMITER $

CREATE PROCEDURE Autentificare\_persoane(IN user\_name varchar(30), IN pass varchar(30), OUT dreptAdmin char, OUT mesaj varchar(256), OUT idMedic int)

BEGIN

DECLARE done int DEFAULT 0;

DECLARE utilizator1 varchar(30) DEFAULT "";

DECLARE parola1 varchar(30) DEFAULT "";

DECLARE cnp1 varchar(13) DEFAULT "";

DECLARE cursor\_autentificare CURSOR FOR SELECT \* FROM Conturi WHERE username = user\_name AND parola = pass;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;

SET dreptAdmin = 'X';

SET mesaj = "Utilizatorul nu exista sau parola incorecta !";

SET idMedic = 0;

OPEN cursor\_autentificare;

get\_autentificare : LOOP

FETCH cursor\_autentificare INTO utilizator1, parola1, cnp1, dreptAdmin;

IF done = 1 THEN

LEAVE get\_autentificare;

END IF;

IF dreptAdmin = 'N' THEN

BLOCK2: BEGIN

DECLARE done2 int DEFAULT 0;

DECLARE nume1 varchar(30) DEFAULT "";

DECLARE prenume1 varchar(30) DEFAULT "";

DECLARE cursor\_date CURSOR FOR SELECT id\_medic, nume, prenume FROM medic WHERE cnp = cnp1;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done2 = 1;

OPEN cursor\_date;

get\_date : LOOP

FETCH cursor\_date INTO idMedic, nume1, prenume1;

IF done2 = 1 THEN

LEAVE get\_date;

END IF;

END LOOP;

CLOSE cursor\_date;

IF idMedic = 0 THEN

BLOCK3: BEGIN

DECLARE done3 int DEFAULT 0;

DECLARE cursor\_date2 CURSOR FOR SELECT id\_medic, nume, prenume FROM personal\_auxiliar WHERE cnp = cnp1;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done3 = 1;

OPEN cursor\_date2;

get\_date2 : LOOP

FETCH cursor\_date2 INTO idMedic, nume1, prenume1;

IF done3 = 1 THEN

LEAVE get\_date2;

END IF;

END LOOP;

CLOSE cursor\_date2;

END BLOCK3;

END IF;

SET mesaj = concat("Bine ai venit, ", prenume1, " ", nume1, "!");

END BLOCK2;

ELSE

SET mesaj = "Bine ai venit, administrator de sistem !";

END IF;

END LOOP;

CLOSE cursor\_autentificare;

END$

DELIMITER ;

*Procedură pentru actualizarea datelor legate de sex și data nașterii*(folosit în Workbench pentru actualizarea datelor din tabela Pacient, acestea fiind inițial nule)

DROP PROCEDURE IF EXISTS Pacient\_Update;

DELIMITER $

CREATE PROCEDURE Pacient\_Update()

BEGIN

DECLARE done INT DEFAULT 0;

DECLARE c\_pacient VARCHAR(13);

DECLARE cursor\_pacient CURSOR FOR SELECT cnp FROM Pacient;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;

OPEN cursor\_pacient;

get\_cnp: LOOP

FETCH cursor\_pacient INTO c\_pacient;

IF done = 1 THEN

LEAVE get\_cnp;

END IF;

SET @sex\_pacient = NULL;

SET @data\_n = NULL;

CALL sex\_din\_cnp(c\_pacient, @sex\_pacient);

CALL data\_nastere\_din\_cnp(c\_pacient, @data\_n);

UPDATE Pacient SET sex = @sex\_pacient, data\_nasterii = @data\_n WHERE cnp = c\_pacient;

END LOOP;

END$

DELIMITER ;

*Procedură pentru calculul caloriilor necesare unui pacient*

DELIMITER $

CREATE PROCEDURE metabolism\_bazal(IN cnp\_param varchar(13), IN activitate varchar(20), OUT MB double)

BEGIN

DECLARE sex char DEFAULT NULL;

DECLARE data\_n date DEFAULT NULL;

DECLARE masa double DEFAULT (SELECT masa\_corporala FROM Masuratori WHERE id\_pacient = (SELECT id\_pacient FROM Pacient WHERE cnp = cnp\_param));

DECLARE h double DEFAULT (SELECT inaltime FROM Masuratori WHERE id\_pacient = (SELECT id\_pacient FROM Pacient WHERE cnp = cnp\_param));

CALL sex\_din\_cnp(cnp\_param, sex);

CALL data\_nastere\_din\_cnp(cnp\_param, data\_n);

IF(sex = 'M') THEN

SET MB = 13.397 \* masa + 4.799 \* h \* 100 - 5.667 \* datediff(curdate(), data\_n) / 365 + 88.362;

ELSE

SET MB = 9.247 \* masa + 3.098 \* h \* 100 - 4.330 \* datediff(curdate(), data\_n) / 365 + 447.593;

END IF;

IF(strcmp(activitate, 'sedentar') = 0) THEN

SET MB = 1.2 \* MB;

ELSEIF(strcmp(activitate, 'activitate usoara') = 0) THEN

SET MB = 1.375 \* MB;

ELSEIF(strcmp(activitate, 'activitate moderata') = 0) THEN

SET MB = 1.55 \* MB;

ELSEIF(strcmp(activitate, 'foarte activ') = 0) THEN

SET MB = 1.725 \* MB;

ELSEIF(strcmp(activitate, 'extrem de activ') = 0) THEN

SET MB = 1.9 \* MB;

END IF;

END$

DELIMITER ;

1. Interogări MYSQL
2. Să se afișeze numele, prenumele, cnp-ul, vaccinul și data vaccinării (sub forma dd-mm-yyyy) pentru fiecare copil sub un an al primulului medic (id\_medic = 1) care are numele de familie începând cu litera ‘P’ și vaccinurile pentru tuse și tuberculoză.

SELECT Nume, Prenume, CNP, Vaccin, DATE\_FORMAT(`Data vaccinare`, '%d-%m-%Y') AS `Data vaccinarii` FROM `VaccinCopil`

WHERE CNP IN(SELECT cnp FROM Pacient WHERE datediff(curdate(), data\_nasterii) / 365 < 1 AND id\_medic = 1) AND Nume LIKE 'P%' AND Vaccin LIKE '%tu%';

1. Afișați pacienții unui medic ordonați după nume.

SELECT Nume, Prenume, CNP, Adresa, Telefon, Sex, DATE\_FORMAT(`data\_nasterii`,'%d/%m/%Y') AS `Data nasterii` FROM Pacient

WHERE id\_medic = 1 ORDER BY Nume;

1. Afișați familiile înscrise la un medic.

SELECT nume AS `Nume familie`, COUNT(nume) AS `Numar membrii`, adresa AS `Adresa` FROM Pacient GROUP BY nume HAVING `Numar membrii` > 1;

1. Afișați persoanele dintr-o anumită categorie de vârstă.

SELECT nume, prenume, cnp FROM Pacient

WHERE datediff(curdate(), data\_nasterii) / 365 BETWEEN 18 AND 35;

1. Afișați orarul săptămânii curente pentru chemările la domiciliu.

SELECT Pacient.nume AS Nume, Pacient.prenume AS Prenume, Pacient.cnp AS CNP, DATE\_FORMAT(Registru\_chemari\_domiciliu.data\_chemare, '%d.%m.%Y') AS `Data consult`, registru\_chemari\_domiciliu.scop AS Scop

FROM Pacient JOIN registru\_chemari\_domiciliu

WHERE Pacient.id\_pacient = registru\_chemari\_domiciliu.id\_pacient

HAVING CNP IN(SELECT cnp FROM Pacient WHERE id\_medic = 1) AND yearweek(STR\_TO\_DATE(`Data consult`, '%d.%m.%Y')) = yearweek(curdate());

1. Afișați copii sub un an și vaccinurile care li s-au administrat, precum și data vaccinării.

SELECT Nume, Prenume, CNP, Vaccin, DATE\_FORMAT(`Data vaccinare`,'%m-%d-%Y') AS `Data vaccinarii`

FROM `VaccinCopil`

WHERE CNP IN (SELECT cnp FROM Pacient WHERE datediff(curdate(), data\_nasterii)/365 < 1 AND id\_medic = 1);

1. Afișați registrul de evidență al bolnavilor cronici.

SELECT Pacient.nume AS Nume, pacient.prenume AS Prenume, pacient.cnp AS CNP, registru\_bolnavi\_cronici.boala AS Boala, DATE\_FORMAT(registru\_bolnavi\_cronici.data\_examinare, '%d.%m.%Y') AS `Data examinare`, registru\_bolnavi\_cronici.observatii AS Observatii

FROM Pacient JOIN registru\_bolnavi\_cronici

WHERE Pacient.id\_pacient = registru\_bolnavi\_cronici.id\_pacient AND Pacient.id\_medic = 1;

1. Din orarul programărilor pe săptămâna curentă afișați persoanele al căror nume începe cu litera ‘P’ și cnp-ul are primele cifre ‘17’.

SELECT Pacient.nume AS Nume, pacient.prenume AS Prenume, pacient.cnp AS CNP, programari.data\_programarii AS `Data programare`

FROM Pacient JOIN Programari

WHERE Pacient.id\_pacient = Programari.id\_pacient AND Programari.id\_medic = 1 AND NUME LIKE 'p%' AND CNP LIKE '17%'

HAVING yearweek(curdate())=yearweek(programari.data\_programarii)

ORDER BY `Data programare`;

1. Afișați aparatele medicale care încep cu litera ‘c’

SELECT id\_aparat AS `Id`, nume\_aparat AS `Nume aparat`, inventar AS `Inventar`

FROM Aparatura\_medicala WHERE nume\_aparat LIKE 'c"%';

1. Afișați codul numeric personal al femeilor cu vârsta mai mare sau egală decât 18 ani

SELECT cnp FROM Pacient

WHERE sex = 'F' AND datediff(curdate(), data\_nasterii)/365 >= 18 AND id\_medic = 1;

1. Afișați codul numeric personal al copiilor sub un an

SELECT cnp FROM Pacient WHERE datediff(curdate(), data\_nasterii)/365 < 1 AND id\_medic = 1;

1. Afișați numărul pacienților unui medic

SELECT COUNT(id\_pacient) AS `Numar pacienti` FROM Pacient WHERE id\_medic = 1;

1. Pentru fiecare pacient cu vârstă mai mare sau egală cu 18 ani să se determine dacă poate dona sânge

SELECT nume, prenume, cnp,

CASE

WHEN datediff(curdate(), data\_nasterii) / 365 >= 18 AND masuratori.masa\_corporala >= 50 THEN 'DA'

ELSE 'NU'

END AS Posibiliate\_donare FROM Pacient, Masuratori WHERE Pacient.id\_pacient = Masuratori.id\_pacient;

1. Inserați în registru\_copii\_sub\_1\_an o tuplă selectând id-ul copilului, numele vaccinului și data curentă

INSERT INTO registru\_copii\_sub\_1\_an(id\_pacient, id\_vaccin, data\_vaccinare) VALUES((SELECT id\_pacient from pacient where cnp = '6160216409967'),(SELECT id\_vaccin from vaccin where nume\_vaccin = 'Vaccinul contra hepatitei de tip B'),curdate());

1. Pentru fiecare pacient sa se determine indicele de masa corporala

SELECT nume, prenume, cnp,

CASE

WHEN masuratori.masa\_corporala/pow(masuratori.inaltime, 2) < 18.5 THEN 'Subponderal'

WHEN masuratori.masa\_corporala/pow(masuratori.inaltime, 2) BETWEEN 18.5 AND 24.99 THEN 'Greutate normala'

WHEN masuratori.masa\_corporala/pow(masuratori.inaltime, 2) BETWEEN 25 AND 29.99 THEN 'Supraponderal'

WHEN masuratori.masa\_corporala/pow(masuratori.inaltime, 2) BETWEEN 30 AND 34.99 THEN 'Obezitate gradul I'

WHEN masuratori.masa\_corporala/pow(masuratori.inaltime, 2) BETWEEN 35 AND 39.99 THEN 'Obezitate gradul II'

ELSE 'Obezitate morbida'

END AS bmi FROM Pacient, Masuratori WHERE Pacient.id\_pacient =Masuratori.id\_pacient;

1. Detalii de implementare
2. Descrierea funcțională a modulelor

Aplicația utilizează o clasă de tip „Controller” pentru a avea o arhitectură mai apropiată arhitecturii MVC (Model View Controller). În acestă clasă avem variabile instanță de tipuri referitoare la o bază de date: conexiune (MySqlConnection), executarea comenzilor (MySqlCommand), citirea datelor (MySqlDataReader), adaptarea datelor (MySqlDataAdapter), introducerea datelor în tabel – crearea de seturi (DataSet). În acestă clasă există și variabile instanță pentru id-ul medicului, mesajul de afișare la logare și dreptul de administrator. Clasa Controller a aplicației este instanțiată la logare. După autentificare este transmisă ca parametru tuturor fereastrelor grafice. De asemenea, aceastp clasă conține implementate metode pentru lucrul cu informația din baza de date.

1. Manual de utilizare/instalare

Manual de utilizare

Aplicația pornește cu o fereastră de logare. Utilizatorul introduce utilizatorul și parola, apoi apasă pe butonul “Autentificare” (Fig. 2).

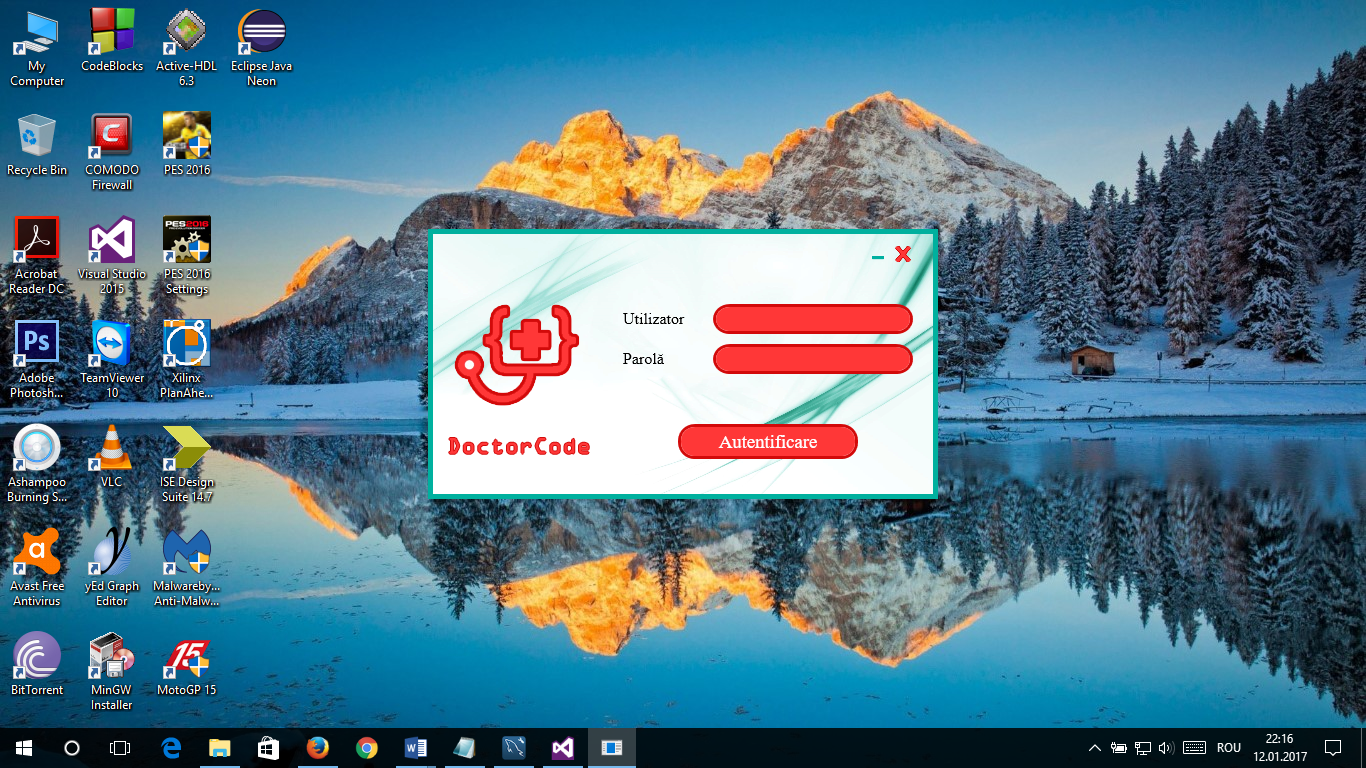


Fig. 2. Fereastra de autentificare

În funcție de numele de utilizator și parolă pragramul detectează dacă utilizatorul este medic sau administrator. Apoi se afișează un mesaj personalizat (Fig. 3). Pentru medic afișează “Bine ai venit, nume prenume!”, unde nume, prenume sunt numele și prenumele medicului. Pentru administrator se va afișa “Bine ai venit, administrator de sistem!”. În caz că numele de utilizator sau parola sunt incorecte se afișează mesajul “Utilizatorul nu exista sau parola incorecta !”.

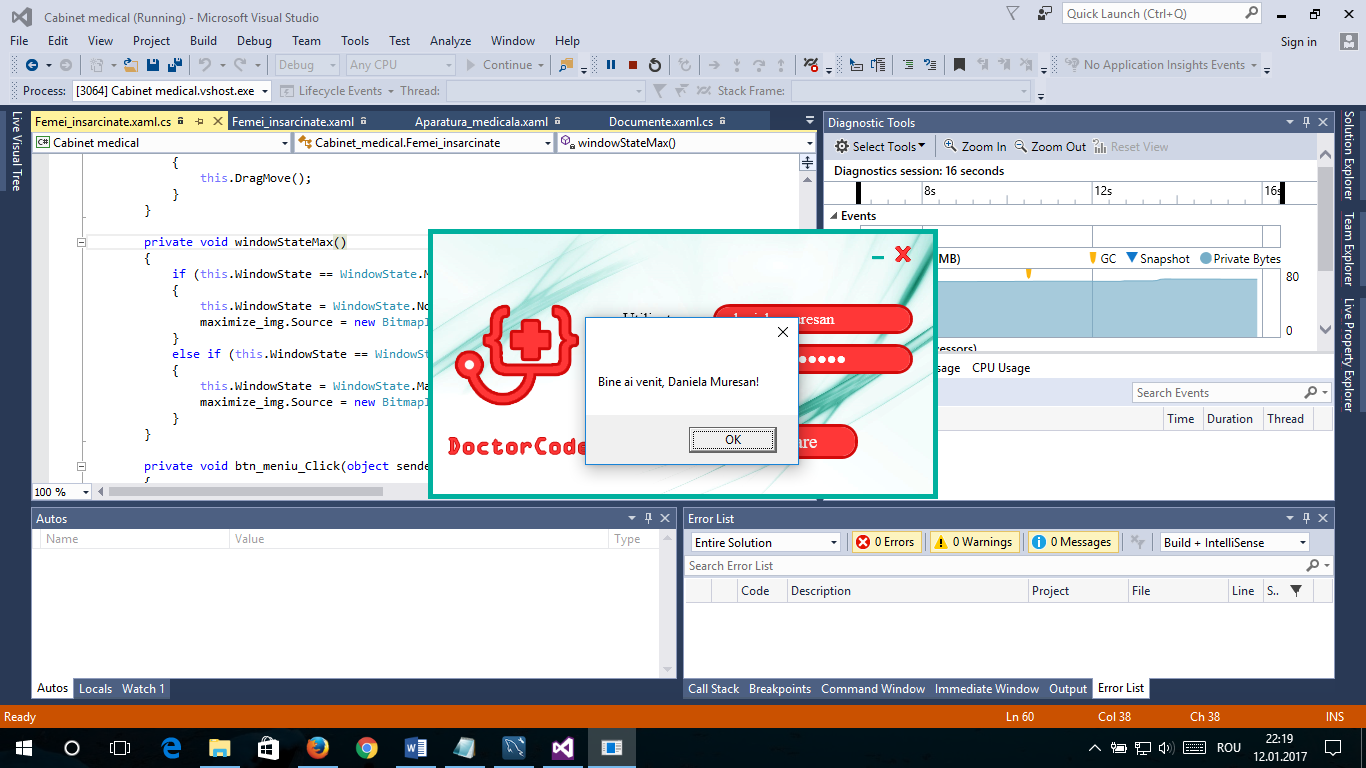


Fig. 3. Mesajul afișat după apăsarea butonului de autentificare

După autentificarea reușită se intră în fereastra de meniu (Fig. 4). Butoanele sunt sub formă de hexagon. Când mouse-ul se află pe un buton va apărea un tooltip care va indica unde poate trimite butonul prin apăsarea sa. Există 14 butoane pentru pacienți, programări, consultații, registru copii sub un an, registru femei însărcinate, registru bolnavi cronici, chemări la domiciliu, registru tratamente boli, vaccinuri, documente (adeverințe, avize, bilete de trimitere, rețete), medicamente din cabinet, obiecte sanitare din cabinet, aparatura medicală din cabinet și logout.

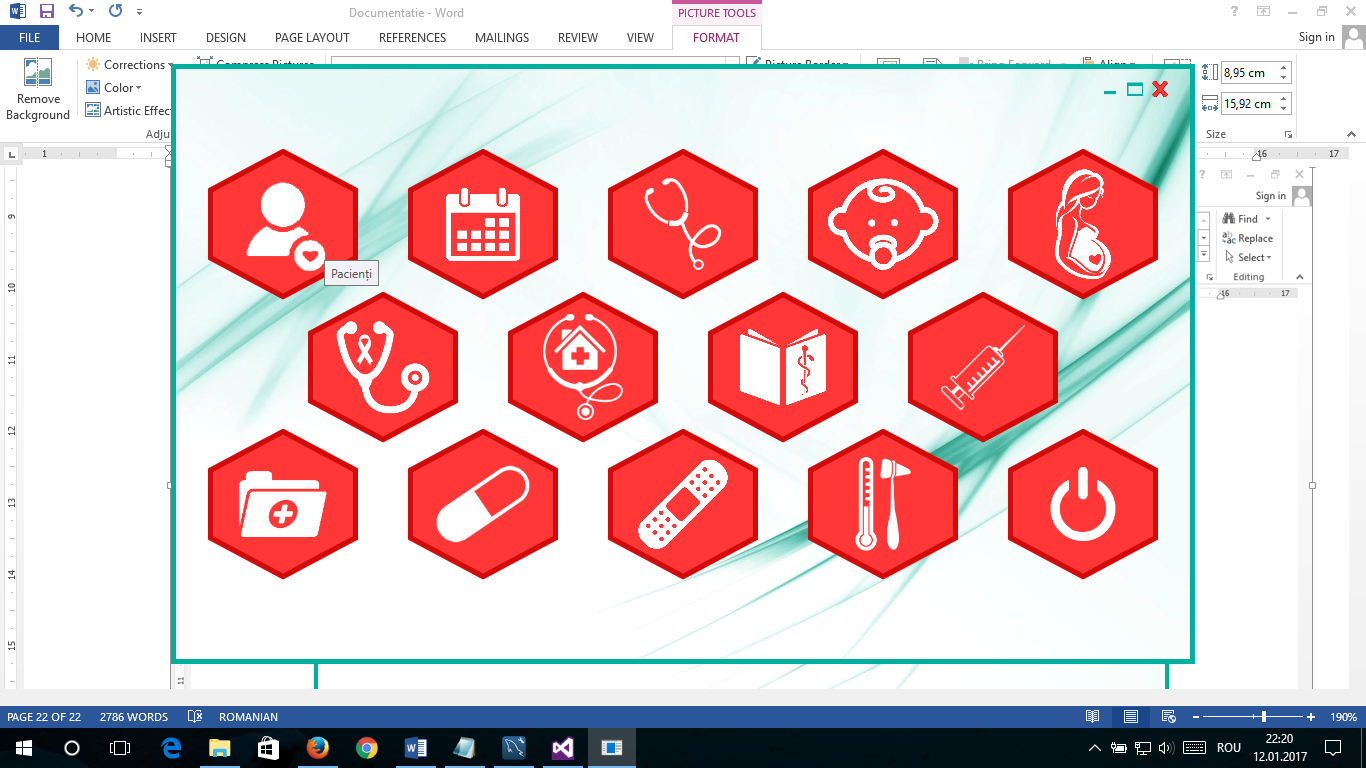


Fig. 4. Meniul aplicației

Apăsând pe butonul “Pacienți” vor apărea trei tab-uri: “Vizualizare pacienți”, “Inserare pacient” și “Date pacient”. În “Vizualizare pacienți” medicul își poate vizualiza toți pacienții, afla lucruri legate de măsurători ale pacienților și familiile înscrise. În “Inserare pacient” există un formular de înscriere a unui pacient (Fig. 5). În “Date pacient” se pot vizualiza datele unui pacient după cnp , acestea putând fi modificate. Butonul în formă de hexagon din partea de jos a ferestrei trimite către meniu (Fig. 4).

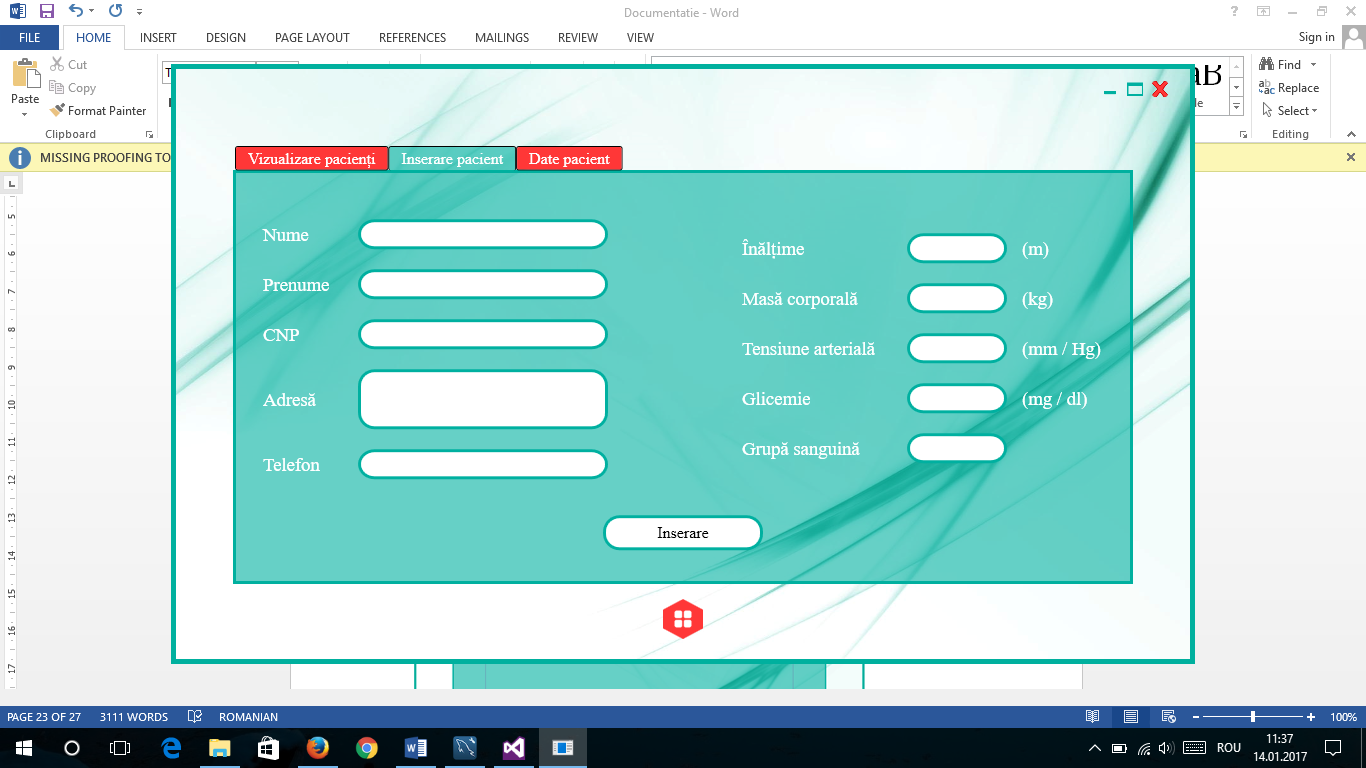


Fig. 5. Formularul de inserare al unui pacient

Apăsând pe butonul “Programări” medical poate vizualiza orarul programărilor și de a insera noi program în orar. Fereastră “Consultații” prezintă două tab-uri: unul pentru istoricul consultațiilor și altul pentru a insera o nouă consultație în tabela de istoric al consultațților.

Fereastra “Registru copii sub un an” conține două tab-uri: “Registru de evidență a copiilor sub un an” și “Administrare vaccin”. Tab-ul “Registru de evidență a copiilor sub un an” (Fig. 6) conține un tabel toți copii sub un an. Se pot căuta copii după nume sau cnp. În caz că există mai mulți copii cu același nume de familie, cnp-ul este utilizat pentru a identifica un singur bebeluș. De asemenea, se pot căuta toți copii cărora li s-a administrat un anumit vaccin și data administrării. Căutarea bebelușilor după vaccin și dată se poate combina cu numele și cnp-ul (Fig. 7).

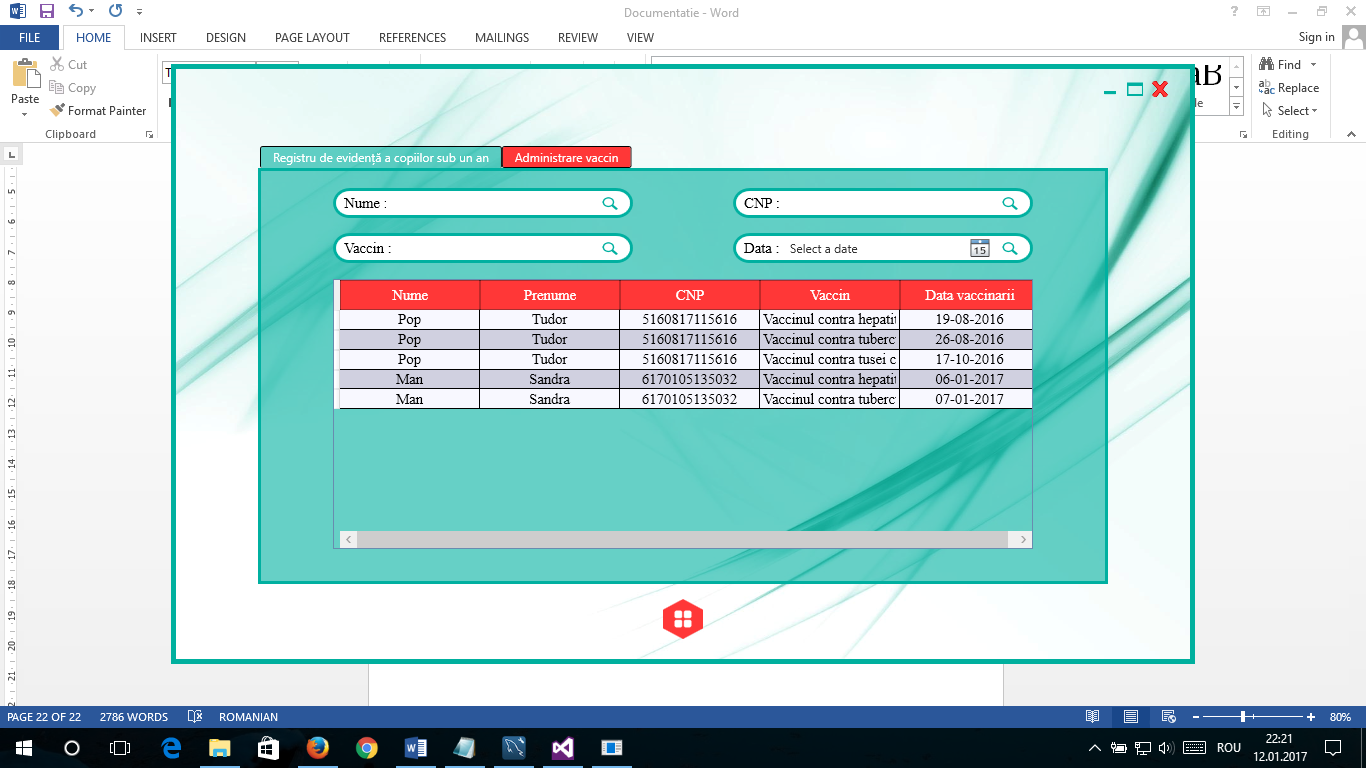


Fig. 6. Tab-ul “Registru de evidență a copiilor sub un an”

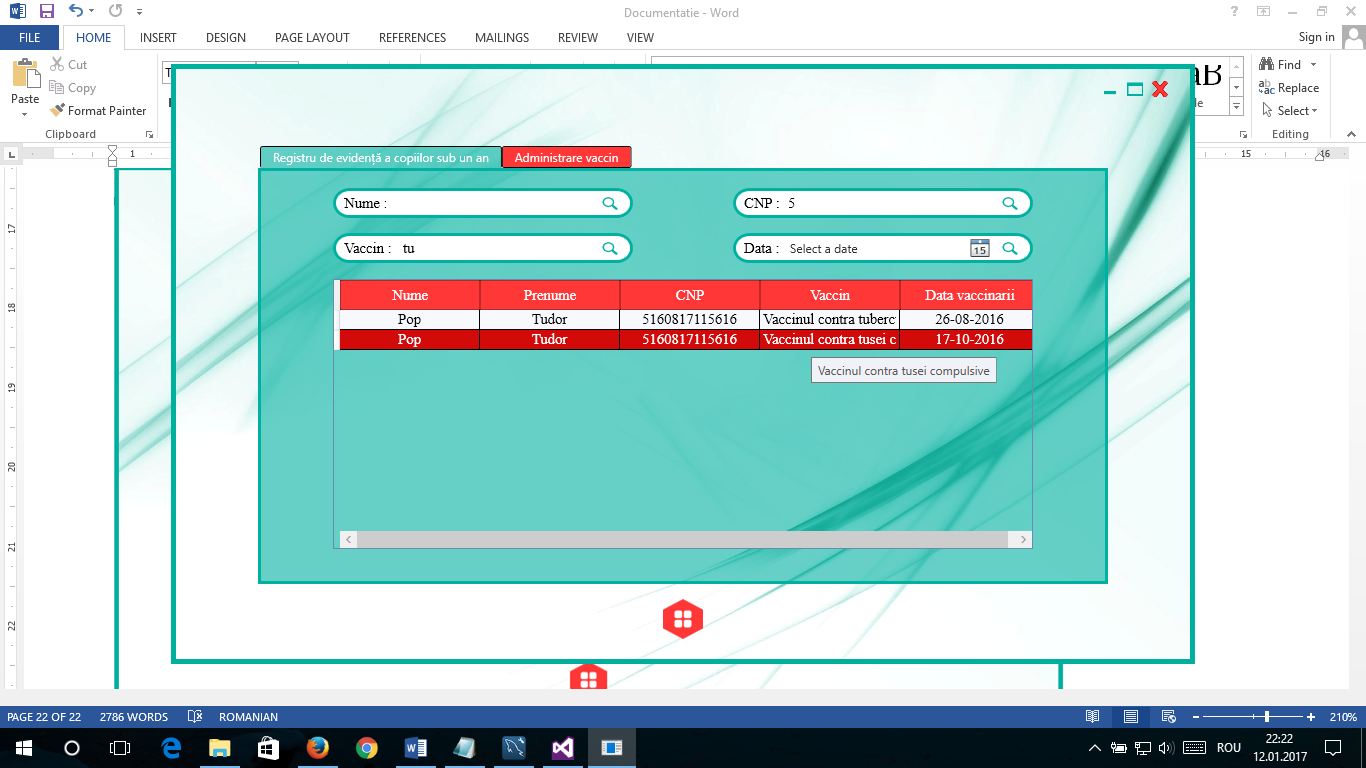


Fig. 7. Căutare după nume, cnp, vaccin, data vaccinării

În Fig. 7 se poate observa că dacă mouse-ul se află pe un rând, acel rând se va colora în roșu închis și textul va fi alb față de un rând normal. Totodată, dacă mouse-ul se află pe o anumită celulă din tabel se va afisș un tooltip cu informația din acea celulă.

În tab-ul “Administrare vaccin”(Fig. 8) se află un mic formular de inserare a datelor în tabel. Pacientul este inserat cu ajutorul codului numeric personal, iar vaccinul se alege dintr-o listă furnizată de baza de date prin intermediul interfeței cu utilizatorul. Data care este inserată este data curentă.

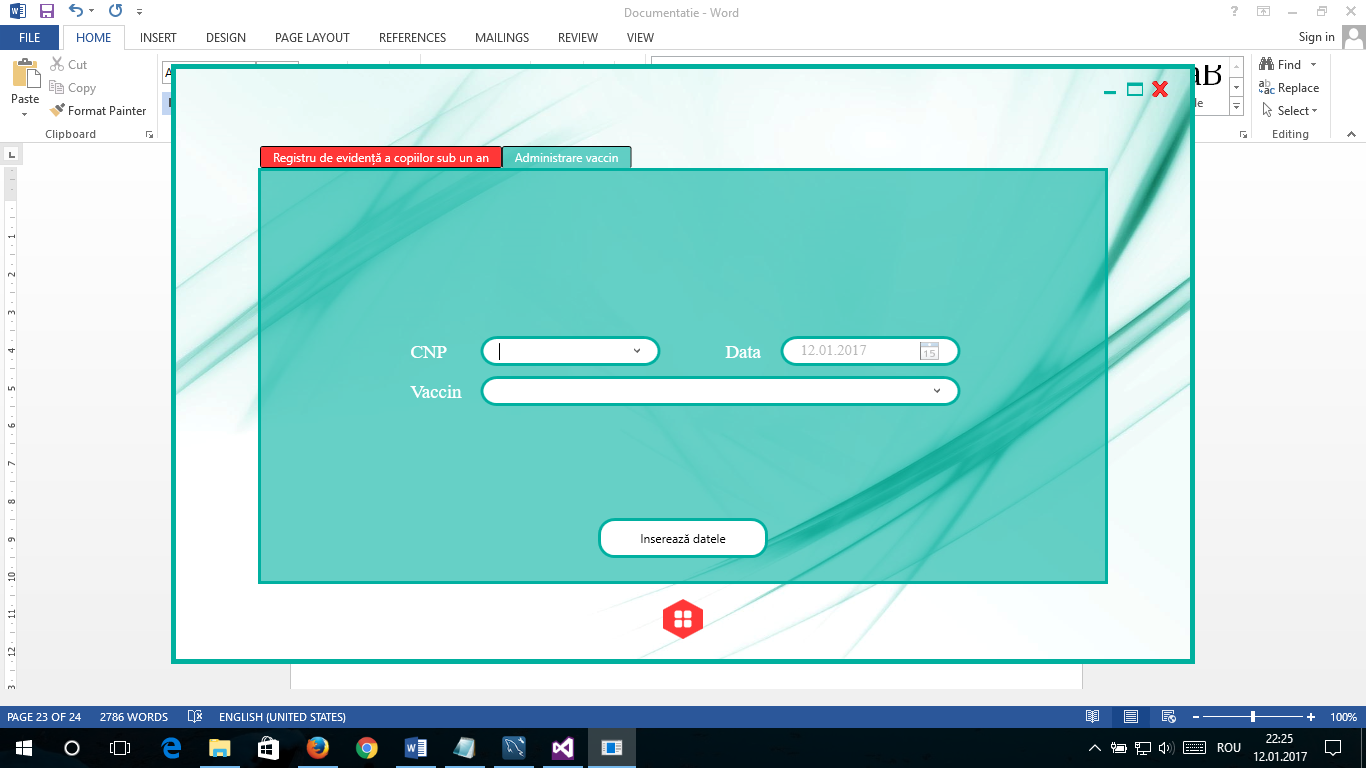


Fig. 8. Tab-ul “Administrare vaccin”

Ferestrele “Registrul femei însărcinate”, “Registrul bolnavi cronici” și “Registrul chemări la domiciliu” au arhitectura precum fereastra “Registru copii sub un an”. Singura diferență este la “Registrul chemări la domiciliu” unde sunt afișate doar consultațiile pe săptămâna curentă, nu ca un istoric în cazul celorlalte.

Fereastra “Obiecte sanitare din cabinet” are un tab care conține un tabel și o bară de căutare pentru obiect(Fig. 9). Atunci când se introduce o informație în bara de căutare tabelul va afișa toate rândurile tabelului inițial care corespund căutării (Fig. 10). Se pot actualiza datele dând click pe celula tabelei și scriind noua valoare. După ce se apasă pe butonul “Salvează modificările” se salvează modificările efectuate și se va afișa pe ecran mesajul “Modificările au fost efectuate și salvate cu succes” (Fig. 11).

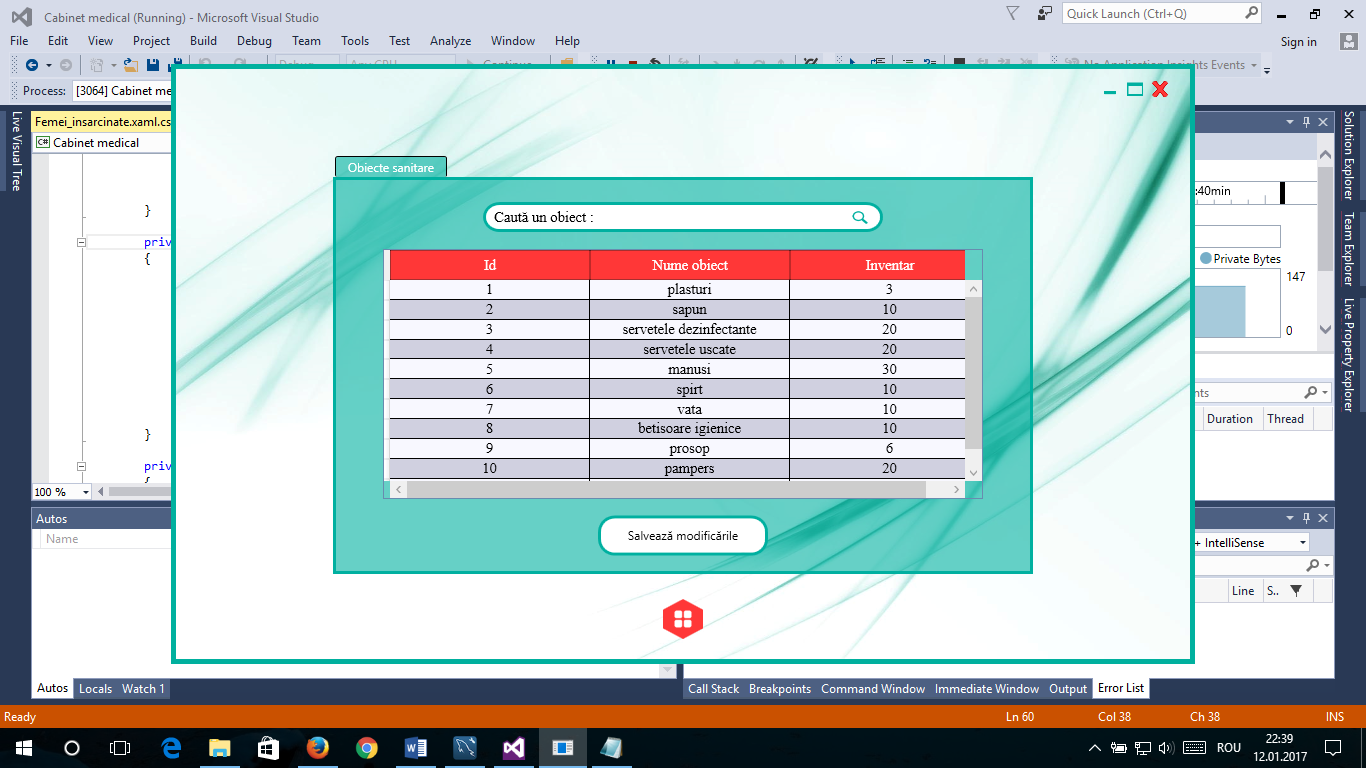


Fig. 9. Fereastra “Obiecte sanitare din cabinet” și componentele sale

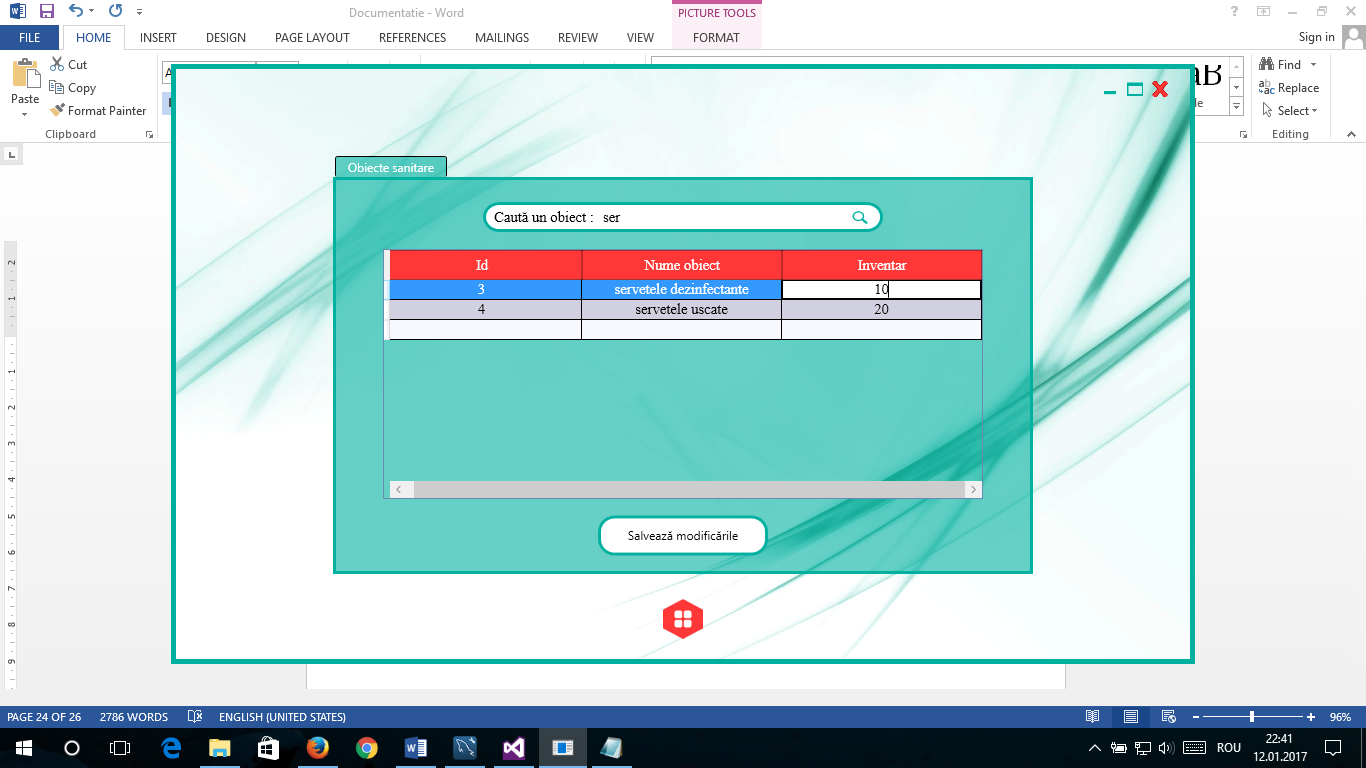


Fig. 10. Căutarea de informații în tabel

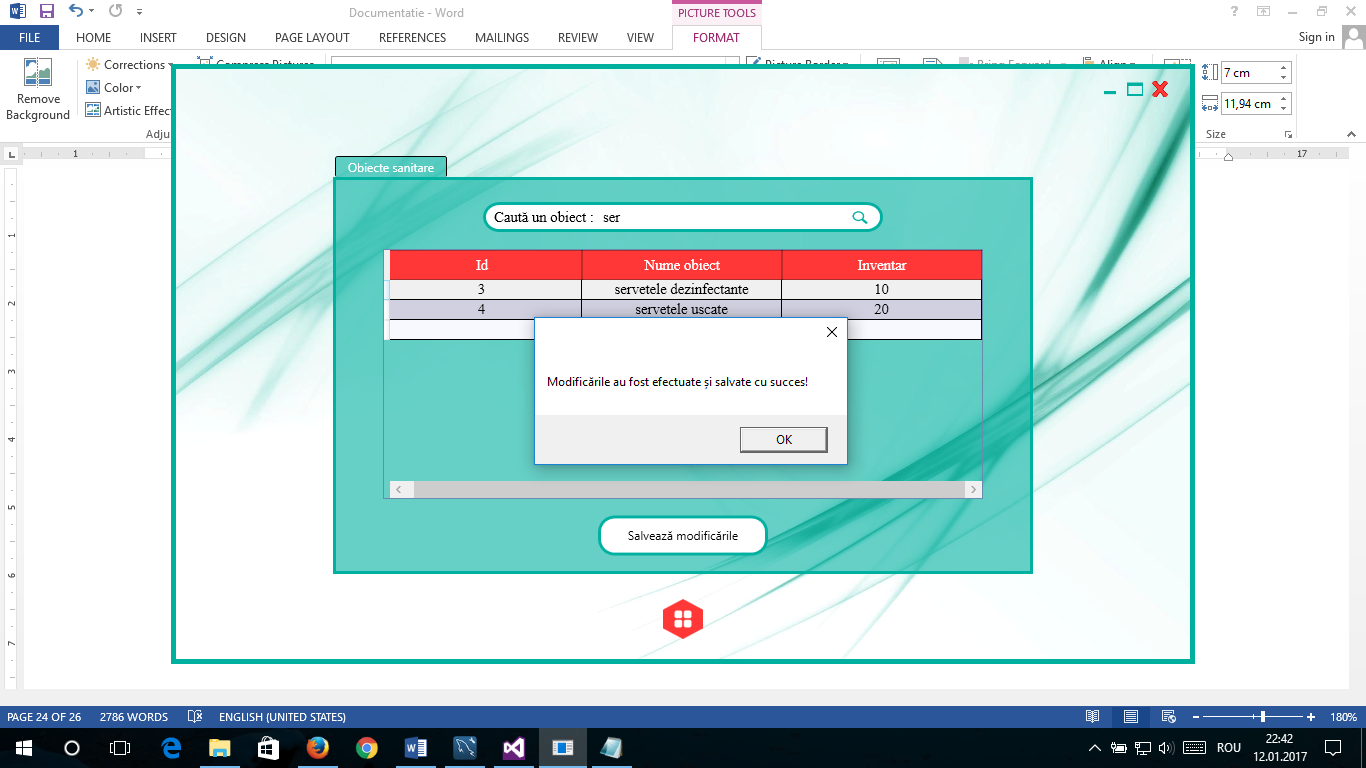


Fig. 11. Salvarea modificărilor

Ferestrele “Medicamente din cabinet”, “Aparatura medicala din cabinet”, “Vaccinuri” și “Registrul de tratamente pentru boli” au arhitectura precum fereastra “Obiecte sanitare din cabinet”.

Manual de instalare

Se dă dublu-click pe “Setup” (Fig. 12).

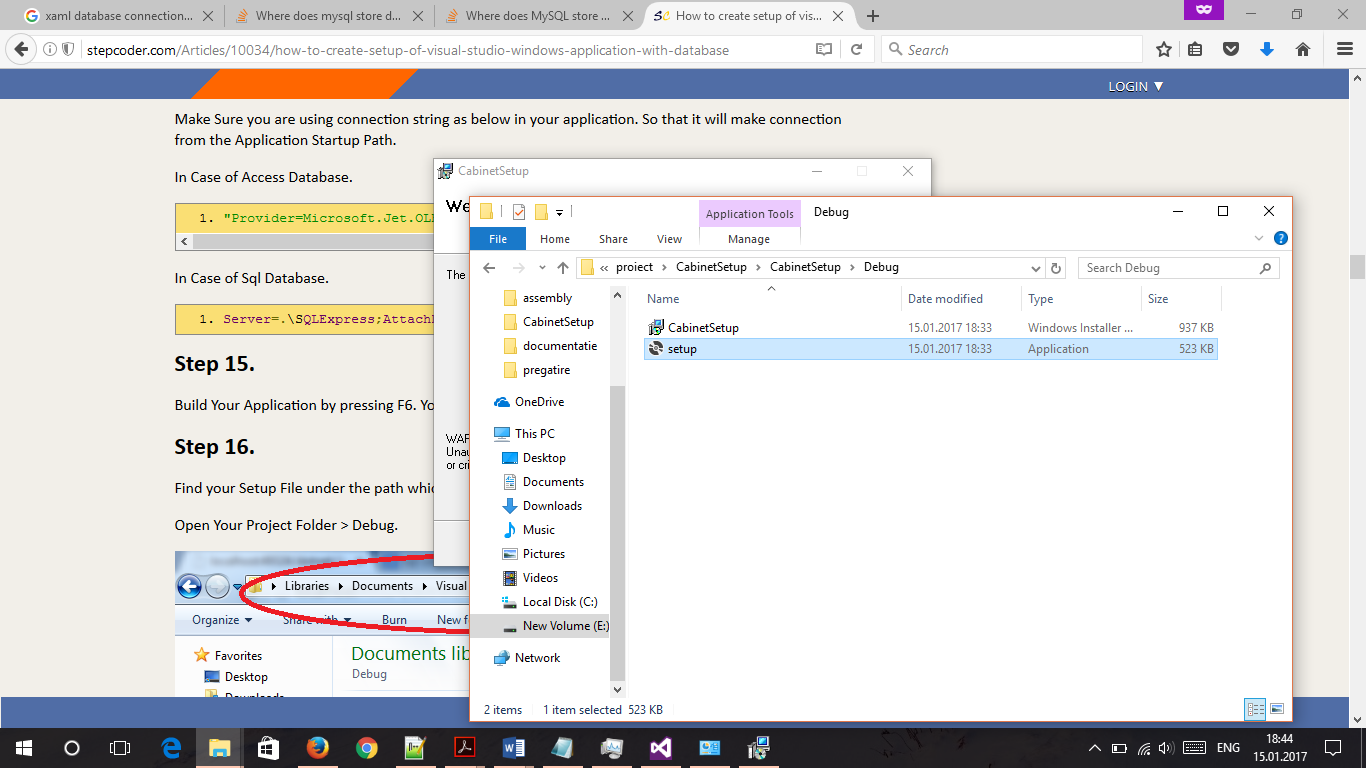


Fig. 12 Setup

Se va deschide o nouă fereastră în care apăsăm “Next” (Fig. 13).

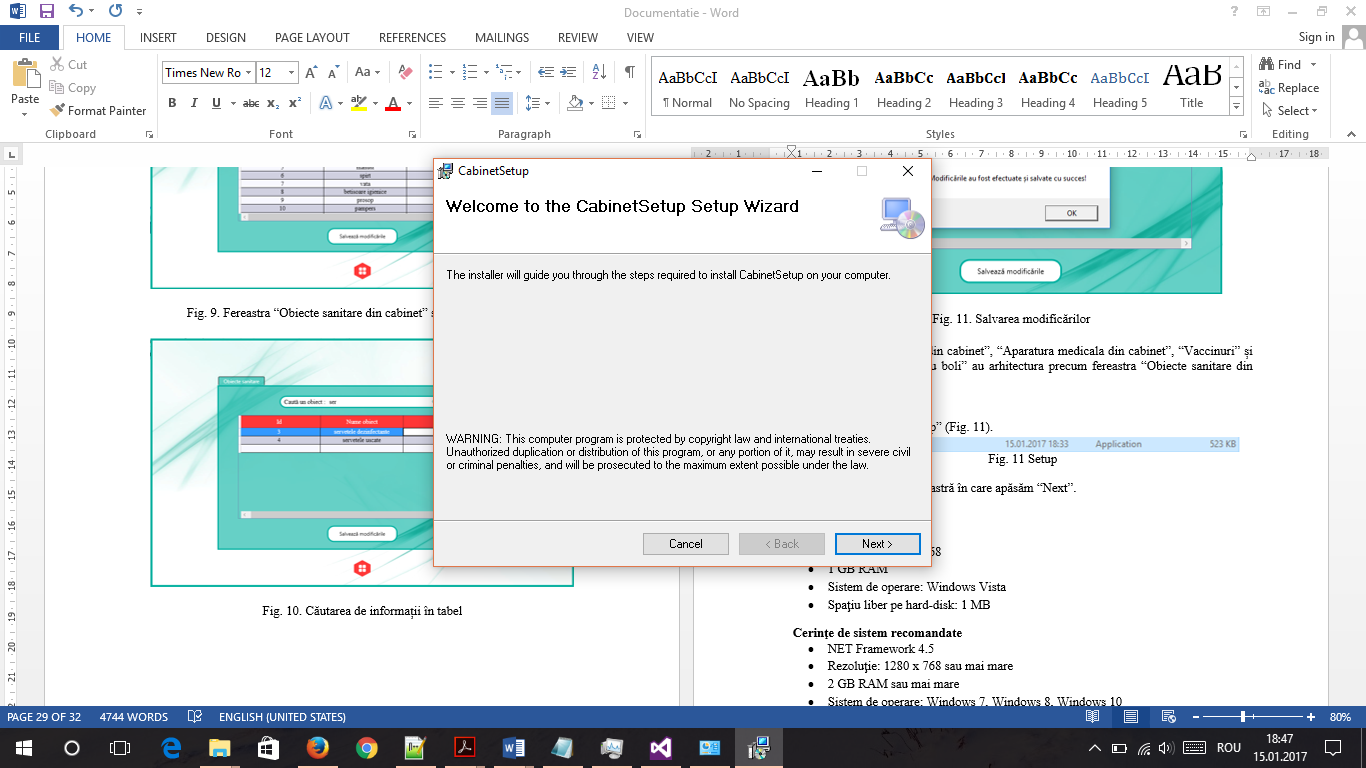


Fig. 13. Apăsarea butonului “Next”

Se allege locația unde se dorețte instalarea softului(Fig. 14).

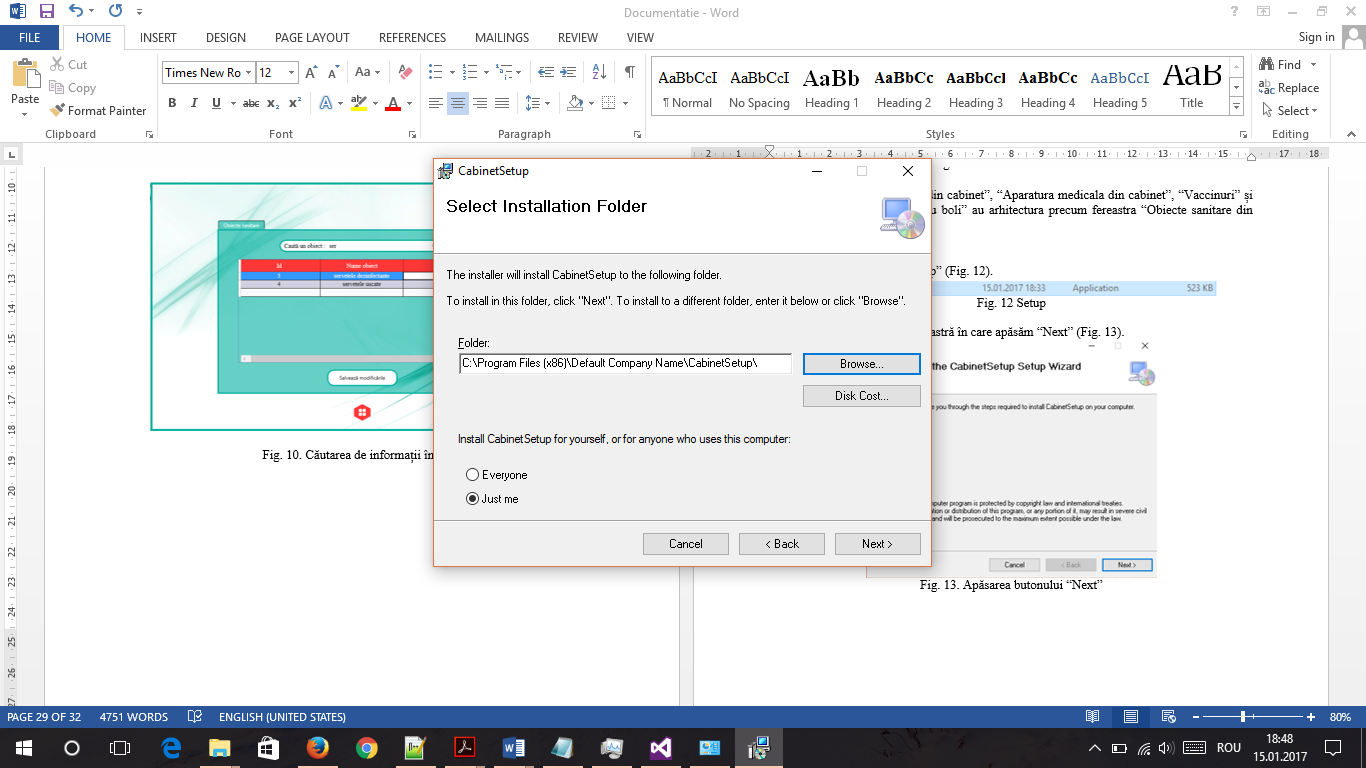


Fig. 14. Selectarea locației de instalare

Apăsăm “Next” și asteptăm să se finalizeze instalarea(Fig. 15).

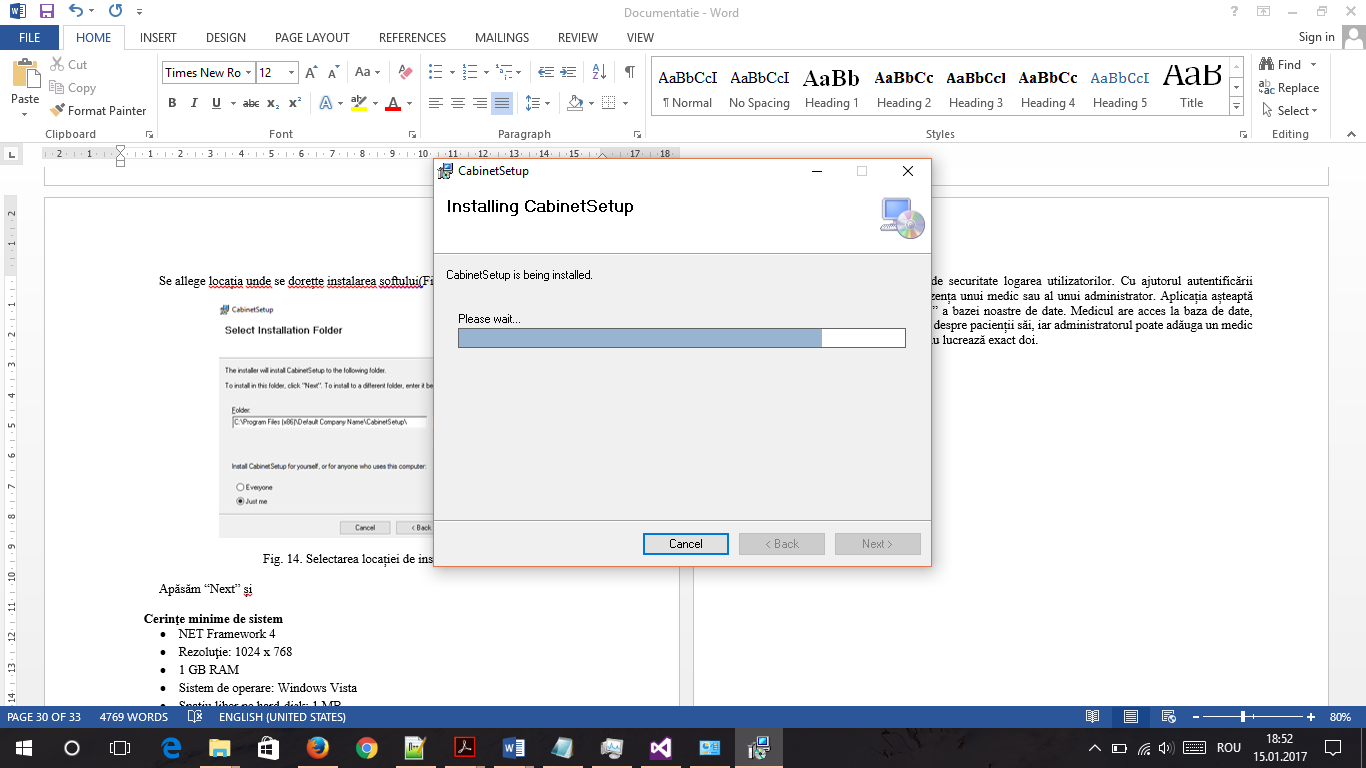


Fig. 15. Instalare

După finalizarea instalării avem pe Desktop iconița aplicației(Fig. 16).



Fig. 16. Iconița aplicației pe Desktop în partea dreaptă jos

**Cerinţe minime de sistem**

* NET Framework 4
* Rezoluţie: 1024 x 768
* 1 GB RAM
* Sistem de operare: Windows Vista
* Spaţiu liber pe hard-disk: 2 MB

**Cerinţe de sistem recomandate**

* NET Framework 4.5
* Rezoluţie: 1280 x 768 sau mai mare
* 2 GB RAM sau mai mare
* Sistem de operare: Windows 7, Windows 8, Windows 10
* Spaţiu liber pe hard-disk: 2 MB

1. Elemente de securizare a aplicației

Aplicația are ca element de securitate logarea utilizatorilor. Cu ajutorul autentificării utilizatorului se detectează prezența unui medic sau al unui administrator. Aplicația așteaptă date valide din tabela “Conturi” a bazei noastre de date. Medicul are acces la baza de date, aceasta conținând multiple date despre pacienții săi, iar administratorul poate adăuga un medic nou în cazul în care la cabinet nu lucrează exact doi.

1. Diagrama de clase a aplicației (Model de ierarhie)

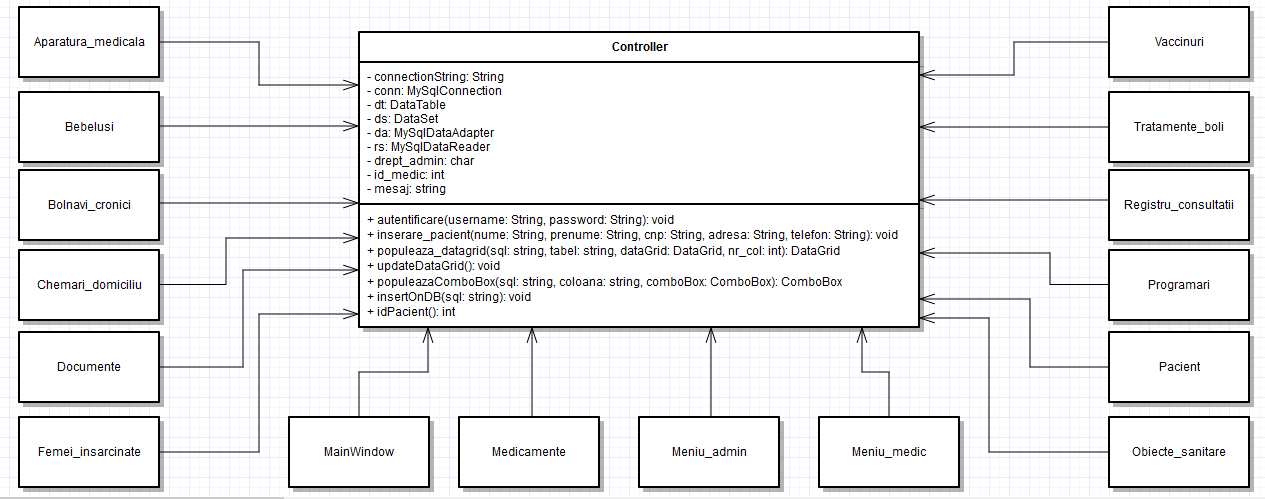


Fig. 2. Diagrama de clase a aplicației (Model de ierarhie

1. Concluzii, limitări și dezvoltări ulterioare

În concluzi, acest soft este util pentru un cabinet medical din punct de vedere al informației stocate digital, aceasta având proprietatea de a fi găsită într-un timp foarte scurt. Aplicația este într-o stare funcțională atât din punct de vedere al interfeței cu utilizatorul cât și al bazei de date

Momentan softul are unele limitări dacă ne raportăm la posibilități de dezvoltare ulterioară.

Printre posibilitățile de dezvoltare ulterioară se numără:

* Extinderea bazei de date la un întreg dispensar
* Crearea de grafice pentru evoluția masei corporale, tensiunii cardiace, glicemiei, înălțimii în funcție de vârstă
* Implementarea printării documentelor medicale din aplicație
* Implementarea grafică orarului programărilor sub o formă caracteristică
* Transmiterea datelor la mai multe calculatoare în interiorul cabinetului pentru eficientizare
* Implementarea unei interfețe grafice mai prietenoase
* Creare istoric medical pentru fiecare pacient
* Conectarea telefonului medicului la aplicație pentru a afișa la ecranul monitorului cine apelează în caz că telefonul este pe silențios
* Accesarea istoricului medical și al graficelor și de către pacient de acasă
* Alertare medic pentru vaccinuri nefăcute copiilor
* Modificarea bazei de date în cazul administrării vaccinurilor; extinderea pentru toți pacienții, nu doar pentru copiii sub un an
* În cazul ferestrelor cu obiecte sanitare, medicamente, aparatură medicală să adăugăm imagini care simbolizează lucrurile amintite înainte
* Trimitere de comenzi în cazul obiectelor sanitare, medicamentelor și vaccinurilor atunci când se atinge un prag minim al inventarului
* Trimitere mail pacientului pentru a-și aminti de programare
* Inserarea atributului fumator la tabela Pacient/Masuratori și calcularea speranței de viață
* Administratorul poate stabili salariul unui medic și poate acorda un bonus aceluia cu cei mai mulți pacienți

1. Bibliografie

[1] <http://www.scritub.com/medicina/MANAGEMENTUL-IN-MEDICINA-FAMIL15419192317.php>

[2] <http://www.slabestecuserban.ro/cum-sa-afli-de-cate-calorii-ai-nevoie-ca-sa-slabesti/>

[3] <http://www.csid.ro/family/crestere-copii/lista-de-vaccinuri-obligatorii-si-optionale-sunt-sau-nu-motive-de-ingrijorare-13951592/>

[4] <http://markcordell.blogspot.ro/2008/12/calling-mysql-stored-procedures-from-c.html>

[5] <http://stackoverflow.com/questions/6255084/fill-datagrid-from-mysql-database-in-c-sharp-wpf>

[6] <https://dev.mysql.com/doc/connector-net/en/connector-net-programming-stored-using.html>

[7] <http://stepcoder.com/Articles/10034/how-to-create-setup-of-visual-studio-windows-application-with-database>

[8] <http://www.elth.pub.ro/~preda/teaching/BDE/BDE_5.pdf>