

Evidența pacienților la un medic de familie

PROIECT BAZE DE DATE

NUME : Cumpănășoiu Laurențiu

GRUPA : 333AA

Cuprins

1. Definirea cerintelor.....	3
1.1. Lista functionalitatilor si identificarea componentelor.....	3
1.2. Organizarea dispensarului.....	3
2. Proiectarea bazei de date.....	4
2.1. Identificarea tabelelor.....	4
2.2. Identificarea campurilor si a tipurilor de date.....	4
2.3. Identificarea relatiilor dintre tabele.....	5
2.4. Impunerea constrangerilor de integritate.....	6
2.5. Diagrama bazei de date.....	8
3. Corelarea interfetei vizuale cu baza de date.....	8
3.1. Insert.....	8
3.2. Update.....	9
3.3. Delete.....	9
3.4. Interogari simple.....	9
3.5. Interogari complexe.....	10
4. Bibliografie.....	10

1. Definirea cerintelor

Aplicatia "Paciento" din cadrul proiectului are rolul de a asigura administrarea si gestionarea pacientilor de toate varstele care sunt inscrisi la un medic de familie in cadrul unui dispensar, insa va putea fi utilizata si in alte unitati medicale (spitale, sanatorii etc). Aplicatia este destinata pentru utilizarea de catre medicii de familie ai dispensarului respectiv , nu si pentru asistentii medicali ai acestuia.

Aceasta aplicatie este implementata in limbajul de programare Python si, pentru baza de data necesara ei, utilizeaza SQL Server. Vom avea si o interfata vizuala si grafica ce va facilita destinatarilor ei accesul mai rapid la persoanele din baza de data respectiva.

1.1. Lista functionalitatilor si identificarea componentelor

- a) Sa se poata crea un cont de catre medicul de familie cu care sa se poate loga in aplicatie (deci sa se poata adauga un medic nou)
- b) Sa se poata adauga o noua persoana in baza de date (un nou-nascut sau o persoana care s-a mutat de la un medic de familie la altul)
- c) Sa se poata sterge o persoana care nu mai este in evidenta medicului de familie pentru ca s-a mutat in alta tara, alt oras, la alt medic sau care a decedat
- d) Sa se poata salva informatii utile (date personale) despre pacienti
- e) Sa poata fi incluse informatii despre diagnosticile pe care le au pacientii, in urma caruia au fost eliberate retete medicale (sa avem un istoric al diagnosticilor)
- f) Sa se poata face modificari si prelucrari la nivelul informatiilor si fiselor medicale ale pacientilor
- g) Sa se poata face programari pentru controalele uzuale

1.2. Organizarea dispensarului

In cadrul dispensarului lucreaza un numar finit de medici de familie, fiecare avand mai multi asistenti medicali, si alti angajati ai cladirii. Fiecare medic are un numar variabil de pacienti. Fiecare pacient are o fisa medicala ce contine atat date personale si alte informatii utile.

Se stie ca in cazul categoriilor vulnerabile (nou-nascuti, gravide) e nevoie de un control regulat. Astfel, medicul are nevoie de o lista cu programari pentru pacientii din aceste categorii.

De asemenea, in cadrul controlului medical, care este efectuat regulat in cazul categoriilor vulnerabile, sau fortat, atunci cand pacientii se imbolnavesc, medicul poate

depista diferite boli/raceli pe care pacientii sai le au la acel moment. Astfel, cadrul medical poate prescrie o reteta cu medicamente pentru tratare potrivit cu diagnosticul respectiv.

2. Proiectarea bazei de date

2.1. Identificarea tabelor

Pentru identificarea tabelor, este esentiala identificarea corecta a entitatiilor care compun sistemul in lumea reala. Apoi, pentru fiecare entitate se va crea in baza de date cate un tabel. Asadar, observam urmatoarele entitati care apar in cadrul organizarii dispensarului si au legatura cu subiectul gestionarii pacientilor :

- a) Medic
- b) Pacienti
- c) Control
- d) Diagnostic
- e) PacientDiagnostic
- f) Medicamente

2.2. Identificarea campurilor si a tipurilor de date

Fiecare entitate de mai sus e caracterizata de o serie de atribute.

MEDIC	IDmedic	Nume	Prenume	CNP	Telefon	Email	DataNasterii	NrCabinet
TIP DATA	integer	varchar	varchar	char	char	var char	date	integer

PACIENTI	TIP DATA
IDpacient	integer
Nume	varchar
Prenume	varchar
CNP	char
Telefon	char
Email	varchar
DataNasterii	date
Strada	varchar
Numar	char
Localitate	varchar
Judet	varchar
Sex	char

Universitatea Politehnică din București
Facultatea de Automatică și Calculatoare

CONTROL	IDcontrol	Tip	Descriere
TIP DATA	integer	varchar	text

DIAGNOSTIC	IDdiagnostic	Nume	Descriere
TIP DATA	integer	varchar	text

PACIENT_DIAGNOSTIC	IDpacient_diagnostic	Data_diagnosticare	Detalii_diagnostic
TIP DATA	integer	date	text

MEDICAMENTE	IDmedicament	Nume	Data_expirare	Descriere
TIP DATA	integer	varchar	date	text

2.3. Identificarea relatiilor dintre tabele

Pentru a identifica relatiile care se stabilesc intre tabele, vom folosi urmatoarea matrice:

	Medic	Pacienti	Control	Diagnostic	Pacient_ Diagnostic	Medicamente
Medic		1:N	1:N			
Pacienti	1:1		1:N		1:N	
Control	1:1	1:N				
Diagnostic					1:N	
PatientDiagnostic		1:1		1:1		1:N
Medicamente					1:N	

Medic – Pacienti : 1:N + 1:1 = 1:N

Medic – Control : 1:N + 1:1 = 1:N

Pacienti – Control : 1:N + 1:N = N:N => inca un tabel de legatura numit **Programari**

Pacienti – PatientDiagnostic : 1:N + 1:1 = 1:N


Diagnostic – PatientDiagnostic : 1:N + 1:1 = 1:N

PatientDiagnostic – Medicamente : 1:N + 1:N = N:N => inca un tabel de legatura numit **Retete**

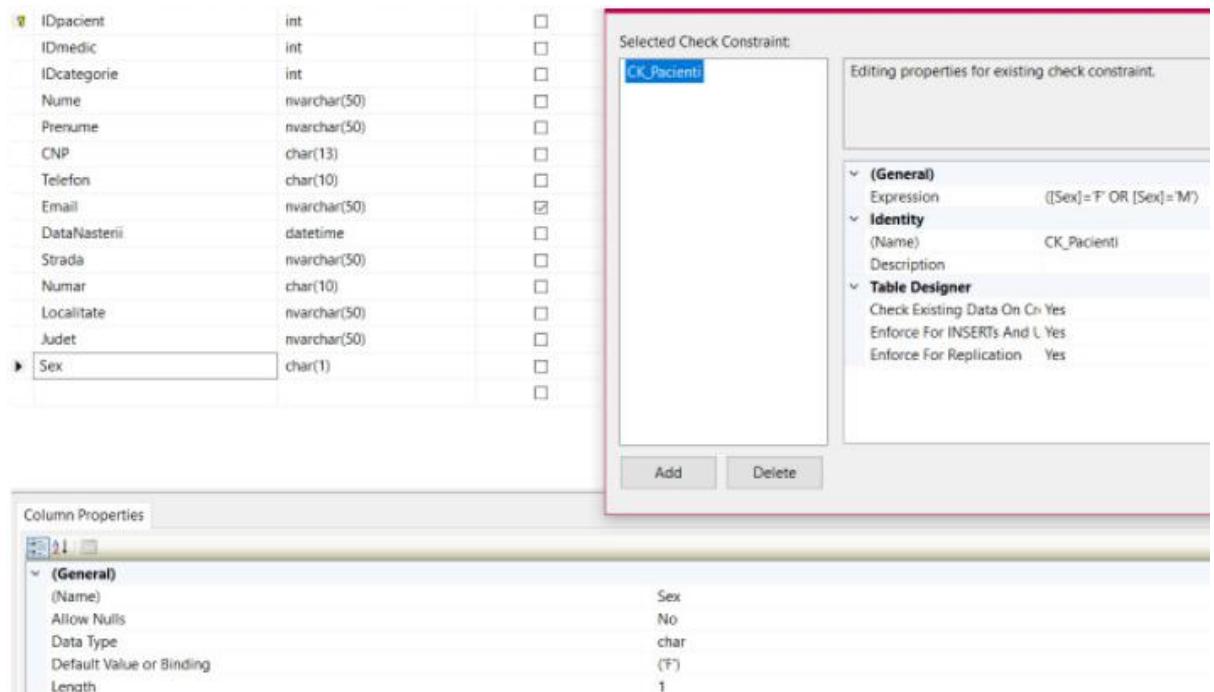
2.4. Impunerea constrangerilor de integritate

Constrangerile de tip primary key, unique, foreign key, not null, check si default pentru valorile din campurile entitatilor noastre sunt exemplificate in imaginile surprinse mai jos:

- Medic:

	Column Name	Data Type	Allow Nulls
	IDmedic	int	<input type="checkbox"/>
	Nume	nvarchar(50)	<input type="checkbox"/>
	Prenume	nvarchar(50)	<input type="checkbox"/>
	CNP	char(13)	<input type="checkbox"/>
	Telefon	char(10)	<input type="checkbox"/>
	Email	nvarchar(50)	<input type="checkbox"/>
	DataNasterii	datetime	<input type="checkbox"/>
	NrCabinet	int	<input type="checkbox"/>
	Parola	nvarchar(30)	<input type="checkbox"/>

- Pacienti:



The screenshot displays the 'Pacienti' table in SQL Server Enterprise Manager. The table has the following columns and data types:

Column Name	Data Type	Allow Nulls
IDpacient	int	<input type="checkbox"/>
IDmedic	int	<input type="checkbox"/>
IDcategorie	int	<input type="checkbox"/>
Nume	nvarchar(50)	<input type="checkbox"/>
Prenume	nvarchar(50)	<input type="checkbox"/>
CNP	char(13)	<input type="checkbox"/>
Telefon	char(10)	<input type="checkbox"/>
Email	nvarchar(50)	<input checked="" type="checkbox"/>
DataNasterii	datetime	<input type="checkbox"/>
Strada	nvarchar(50)	<input type="checkbox"/>
Numar	char(10)	<input type="checkbox"/>
Localitate	nvarchar(50)	<input type="checkbox"/>
Judet	nvarchar(50)	<input type="checkbox"/>
Sex	char(1)	<input type="checkbox"/>

The 'Selected Check Constraint' dialog box is open, showing the constraint 'CK_Pacienti'. The expression is: `[(Sex) = 'F' OR (Sex) = 'M']`. The dialog also shows the 'General' tab with the constraint name 'CK_Pacienti' and the 'Table Designer' tab with the following settings:

- Check Existing Data On Cn: Yes
- Enforce For INSERTs And L: Yes
- Enforce For Replication: Yes

The 'Column Properties' window for the 'Sex' column is also visible, showing the following properties:

- (Name): Sex
- Allow Nulls: No
- Data Type: char
- Default Value or Binding: ('F')
- Length: 1

- Control:

Universitatea Politehnică din București
Facultatea de Automatică și Calculatoare

	Column Name	Data Type	Allow Nulls
▶	IDcontrol	int	<input type="checkbox"/>
	IDmedic	int	<input type="checkbox"/>
	Tip	nvarchar(30)	<input type="checkbox"/>
	Descriere	ntext	<input checked="" type="checkbox"/>

- Diagnostic:

	Column Name	Data Type	Allow Nulls
▶	IDdiagnostic	int	<input type="checkbox"/>
	Nume	nvarchar(50)	<input type="checkbox"/>
▶	Descriere	ntext	<input type="checkbox"/>

- Medicamente:

	Column Name	Data Type	Allow Nulls
▶	IDmedicament	int	<input type="checkbox"/>
	Nume	varchar(50)	<input type="checkbox"/>
	Data_expirare	date	<input type="checkbox"/>
	Descriere	ntext	<input type="checkbox"/>

- Programari:

	Column Name	Data Type	Allow Nulls
▶	IDpacient	int	<input type="checkbox"/>
▶	IDcontrol	int	<input type="checkbox"/>
	Data_programare	datetime	<input type="checkbox"/>
	Detalii_programare	ntext	<input checked="" type="checkbox"/>

- PatientDiagnostic:

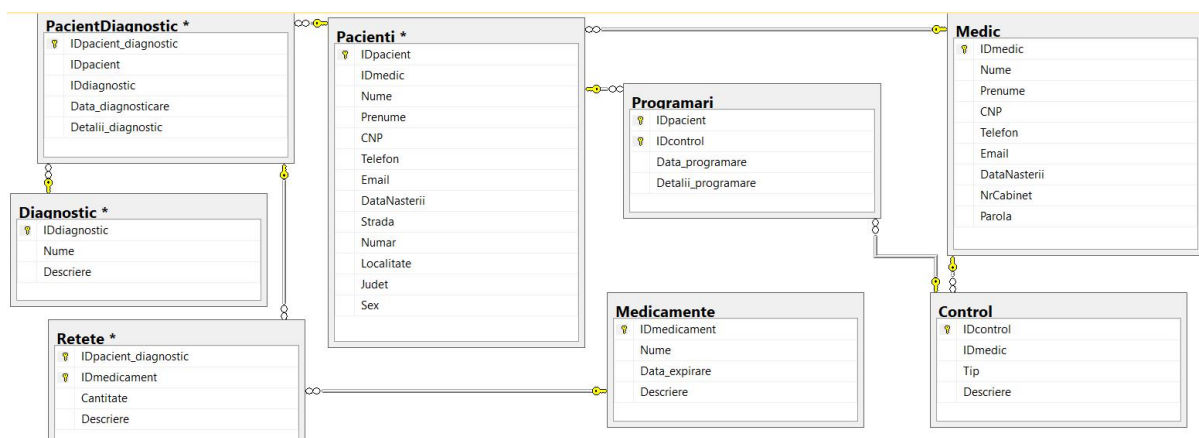
	Column Name	Data Type	Allow Nulls
▶	IDpacient_diagnostic	int	<input type="checkbox"/>
	IDpacient	int	<input type="checkbox"/>
	IDdiagnostic	int	<input type="checkbox"/>
	Data_diagnosticare	smalldatetime	<input type="checkbox"/>
	Detalii_diagnostic	ntext	<input type="checkbox"/>

- Retete:

Universitatea Politehnică din București
Facultatea de Automatică și Calculatoare

	Column Name	Data Type	Allow Nulls
🔑	IDpacient_diagnostic	int	<input type="checkbox"/>
🔑	IDmedicament	int	<input type="checkbox"/>
	Cantitate	float	<input type="checkbox"/>
	Descriere	ntext	<input type="checkbox"/>

2.5. Diagrama bazei de date



3. Corelarea interfetei vizuale cu baza de date

Precizari si observatii:

- Interogările specificate la **insert**, **update** si **delete** sunt doar cateva exemple (nu sunt toate interogările folosite in cadrul aplicatiei).
- Intre { } care apar in cadrul interogărilor pot fi specificate inregistrari pe care le dam noi (in aplicatie sunt parametri specificati in cadrul unei functii si folositi ca atare).
- Toate interogările se gasesc in aplicatie in fisierul **model.py** pentru a fi mai usor de identificat si deoarece aceasta este fisierul care face legatura cu baza de date.

3.1. Insert

a) `INSERT INTO Medic (Nume, Prenume, CNP, telefon, email, dataNasterii, nrCabinet, parola) VALUES ({nume}, {prenume}, {CNP}, {telefon}, {email}, {dataNasterii}, {nrCabinet}, {parola})`

Universitatea Politehnică din București
Facultatea de Automatică și Calculatoare

```
b) INSERT INTO Pacienti (IDmedic, Nume, Prenume, CNP, Telefon, Email, DataNasterii, Strada, Numar, Localitate, Judet, Sex) VALUES ((SELECT IDmedic FROM Medic WHERE email = '{emailMedic}' and parola = '{parolaMedic}'), '{nume}', '{prenume}', '{CNP}', '{telefon}', '{email}', '{dataNasterii}', '{strada}', '{numar}', '{localitate}', '{judet}', '{sex[0]}')
```

3.2. Update

```
a) UPDATE Retete SET Cantitate = '{cantitate}', Descriere = '{descriere}' WHERE IDpacient_diagnostic = '{IDpacientDiagnostic}' and IDmedicament in (SELECT IDmedicament FROM Medicamente WHERE Nume = '{numeMedicament}')
```

```
b) UPDATE Programari SET Data_programare = '{data_programare}', Detalii_programare = '{detalii}' WHERE IDpacient in (SELECT IDpacient FROM Pacienti WHERE CNP = '{cnp_pacient}') and IDcontrol in (SELECT IDcontrol FROM Control WHERE Tip = '{tip_control}')
```

3.3. Delete

```
a) DELETE FROM PacientDiagnostic WHERE IDpacient in (SELECT IDpacient FROM Pacienti WHERE CNP = '{CNP}') and Data_diagnosticare = '{data_diagnostic}'
```

```
b) DELETE FROM Programari WHERE Data_programare < '{current_date}'
```

3.4. Interogari simple

```
1) SELECT P.Nume + ' ' + P.Prenume + ', ' + P.CNP FROM Pacienti P INNER JOIN Medic M on P.IDmedic = M.IDmedic WHERE M.email = '{emailMedic}' and M.parola = '{parolaMedic}' ORDER BY P.Nume, P.Prenume ASC
```

```
2) SELECT P.Nume + ' ' + P.Prenume + ', ' + P.CNP FROM Pacienti P INNER JOIN Medic M on P.IDmedic = M.IDmedic WHERE (P.Nume like '{text}%' or P.Prenume like '{text}%') and M.email = '{emailMedic}' and M.parola = '{parolaMedic}' ORDER BY P.Nume, P.Prenume ASC
```

```
3) SELECT count(P.IDpacient) FROM Pacienti P INNER JOIN Medic M on P.IDmedic = M.IDmedic WHERE M.email = '{emailMedic}' and M.parola = '{parolaMedic}' GROUP BY P.IDmedic HAVING count(P.IDpacient) > 0
```

```
4) SELECT PD.IDpacient_diagnostic FROM PacientDiagnostic PD INNER JOIN Pacienti P on PD.IDpacient = P.IDpacient WHERE P.CNP = '{CNP}' and Data_diagnosticare = '{data_diagnostic}'
```

```
5) SELECT M.Nume, R.Cantitate, R.Descriere FROM Retete R INNER JOIN Medicamente M on R.IDmedicament = M.IDmedicament WHERE R.IDpacient_diagnostic = '{IDpacientDiagnostic}' and M.Nume = '{numeMedicament}'
```

```
6) SELECT C.Tip FROM Control C INNER JOIN Medic M on C.IDmedic = M.IDmedic WHERE M.Email = '{emailMedic}' and M.Parola = '{parolaMedic}'
```

```
7) SELECT C.Tip, Pr.Data_programare, Pr.Detalii_programare FROM Control C INNER JOIN Programari Pr on C.IDcontrol = Pr.IDcontrol INNER JOIN Pacienti P on Pr.IDpacient = P.IDpacient WHERE Pr.Data_programare = '{data_programare}' and P.CNP = '{cnp_pacient}'
```

3.5. Interogari complexe

```
1) SELECT D.Nume + ', ' + convert(varchar, PD.Data_diagnosticare, 120) FROM Diagnostic D INNER JOIN PacientDiagnostic PD on D.IDdiagnostic = PD.IDdiagnostic WHERE PD.IDpacient = (SELECT IDpacient FROM Pacienti WHERE CNP = '{CNP}') ORDER BY D.Nume ASC
```

```
2) SELECT M.Nume + ' , ' + str(datediff(day, getdate(), M.Data_expirare)) FROM Medicamente M INNER JOIN Retete R on M.IDmedicament = R.IDmedicament WHERE R.IDpacient_diagnostic exists (SELECT Re.IDpacient_diagnostic FROM Retete Re INNER JOIN PacientDiagnostic PD on Re.IDpacient_diagnostic = PD.IDpacient_diagnostic INNER JOIN Pacienti P on PD.IDpacient = P.IDpacient WHERE PD.Data_diagnosticare = '{data_diagnostic}' and P.CNP = '{cnp_pacient}')) and datediff(day, getdate(), M.Data_expirare) > 0 ORDER BY datediff(day, M.Data_expirare, getdate()) ASC
```

```
3) SELECT M.Nume + ' , ' + str(datediff(day, getdate(), M.Data_expirare)) FROM Medicamente M INNER JOIN Retete R on M.IDmedicament = R.IDmedicament WHERE R.IDpacient_diagnostic in (SELECT Re.IDpacient_diagnostic FROM Retete Re INNER JOIN PacientDiagnostic PD on Re.IDpacient_diagnostic = PD.IDpacient_diagnostic INNER JOIN Pacienti P on PD.IDpacient = P.IDpacient WHERE PD.IDpacient_diagnostic = '{IDpacientDiagnostic}' and P.CNP = '{cnp_pacient}')) ORDER BY datediff(day, M.Data_expirare, getdate()) ASC
```

```
4) SELECT P.Nume + ' ' + P.Prenume + ' , ' + P.CNP + ' , ' + convert(varchar, Pr.Data_programare, 120) FROM Pacienti P INNER JOIN Programari Pr on P.IDpacient = Pr.IDpacient WHERE P.IDmedic = (SELECT IDmedic FROM Medic WHERE Email = '{emailMedic}' and Parola = '{parolaMedic}')) and Pr.Data_programare > getdate() ORDER BY Pr.Data_programare ASC
```

```
5) SELECT P.Nume + ' ' + P.Prenume + ' , ' + P.CNP FROM Pacienti P WHERE P.DataNasterii <= ALL(SELECT DataNasterii FROM Pacienti WHERE IDmedic = (SELECT M.IDmedic FROM Medic M WHERE M.email = '{emailMedic}' and M.parola = '{parolaMedic}')) ORDER BY P.Nume, P.Prenume ASC
```

4. Bibliografie

<https://www.analyticsvidhya.com/blog/2021/06/how-to-access-use-sql-database-with-pyodbc-in-python/>

<https://doc.qt.io/qtforpython/>

<https://doc.qt.io/qt-5/qtdesigner-manual.html>

<https://doc.bccnsoft.com/docs/PyQt5/designer.html>

