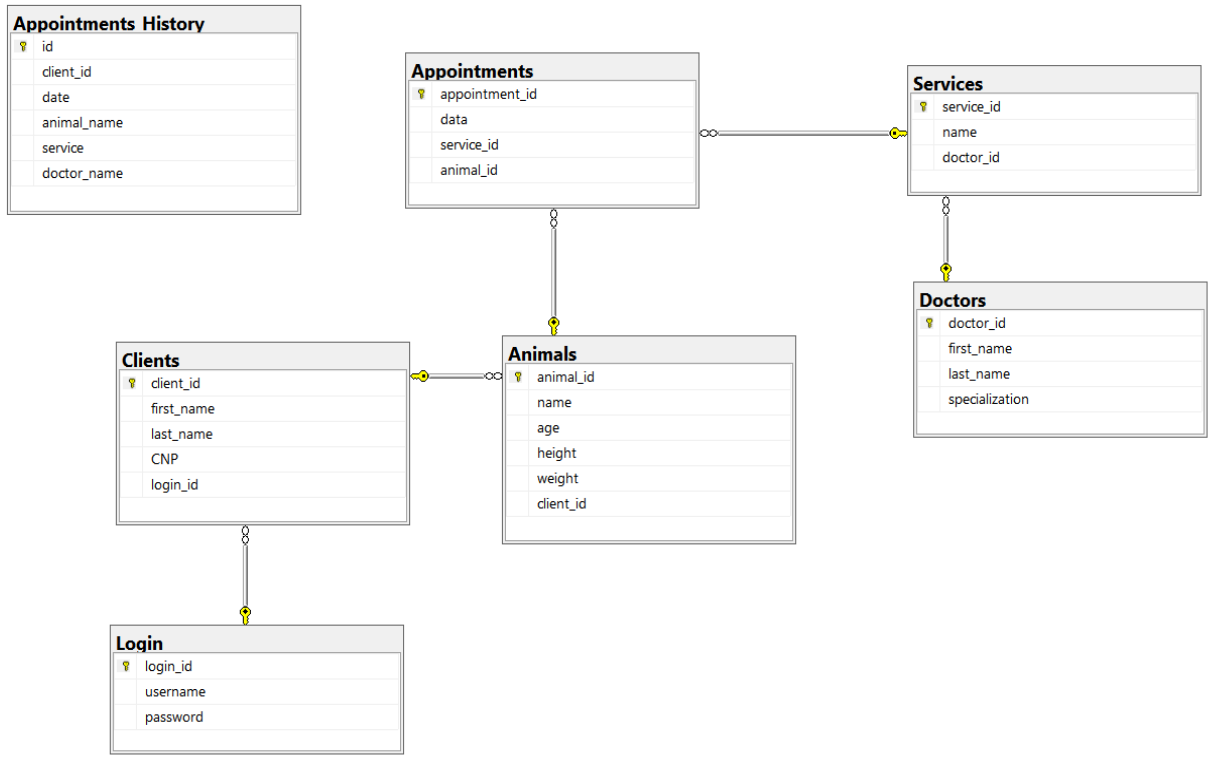
Savu Ioana Rusalda -343C3

Clinica veterinara

Diagrama bazei de date:



Structura tabelelor:

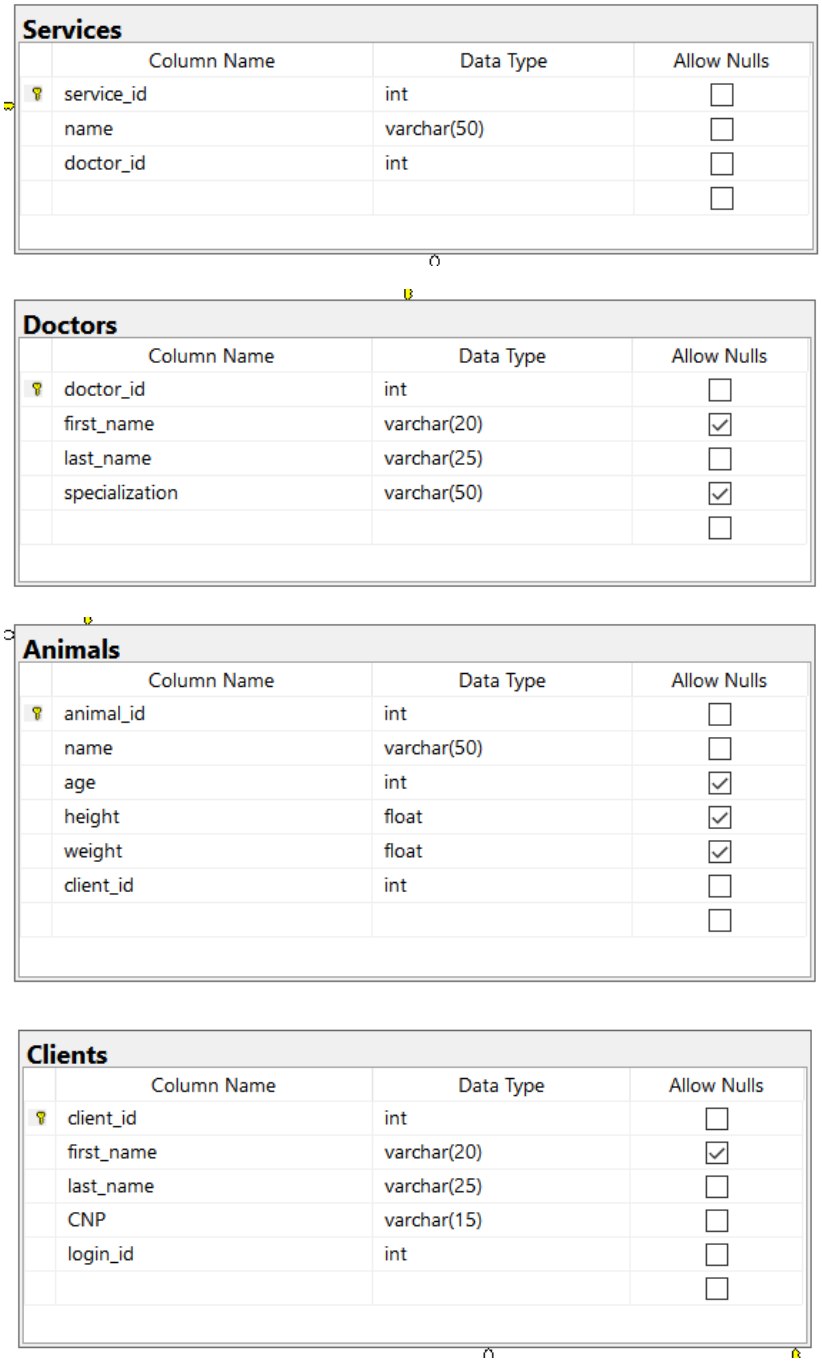
Appointments History

Column Name	Data Type	Allow Nulls
id	int	☐
client_id	int	☐
date	date	☐
animal_name	varchar(50)	☐
service	varchar(50)	☐
doctor_name	varchar(50)	☐
		☐

Appointments

Column Name	Data Type	Allow Nulls
appointment_id	int	☐
data	datetime	☐
service_id	int	☐
animal_id	int	☐
		☐

Savu Ioana Rusalda -343C3



Services

Column Name	Data Type	Allow Nulls
service_id	int	☐
name	varchar(50)	☐
doctor_id	int	☐
		☐

Doctors

Column Name	Data Type	Allow Nulls
doctor_id	int	☐
first_name	varchar(20)	☑
last_name	varchar(25)	☐
specialization	varchar(50)	☑
		☐

Animals

Column Name	Data Type	Allow Nulls
animal_id	int	☐
name	varchar(50)	☐
age	int	☑
height	float	☑
weight	float	☑
client_id	int	☐
		☐

Clients

Column Name	Data Type	Allow Nulls
client_id	int	☐
first_name	varchar(20)	☑
last_name	varchar(25)	☐
CNP	varchar(15)	☐
login_id	int	☐
		☐

CNP – cheie unica (nu pot exista 2 clienti cu acelasi CNP)

Login

Column Name	Data Type	Allow Nulls
login_id	int	☐
username	varchar(50)	☐
password	varchar(50)	☑
		☐

Savu Ioana Rusalda -343C3

Descrierea procedurilor și funcțiilor:

```
ALTER PROCEDURE [dbo].[SignUP]
    -- Add the parameters for the stored procedure here
    @username varchar(50),
    @password varchar(50),
    @error varchar(50) OUT
AS
BEGIN TRY
    IF EXISTS(SELECT * FROM dbo.Login WHERE Username= @username)
        THROW 51002, 'User ID Already Exists. Cannot Insert', 1
    ELSE
        BEGIN TRY
            INSERT INTO dbo.Login( Username, Password)
            VALUES( @username, @password)
        END TRY
        BEGIN CATCH
            SELECT ERROR_MESSAGE() AS Error
        END CATCH
END TRY
BEGIN CATCH
    SELECT ERROR_MESSAGE() AS Error
    SET @error = ERROR_MESSAGE()
END CATCH
```

Procedura verifica daca la crearea unui user nou, username-ul ales de client exista deja. In aceasta procedura am introdus si o exceptie. Daca username-ul exista deja in baza de date se va arunca o exceptie ce urmeaza a fi prinsa. Blocul Try de tartare a exceptiei incearca introducerea datelor unui nou cont in baza de date. Blocul catch intoarce un mesaj de eroare in cazul in care username-ul a fost deja folosit.

```
ALTER PROCEDURE [dbo].[Appointments_Doctor]
    -- Add the parameters for the stored procedure here
    @doctor_id int,
    @date date,
    @count int OUT
AS
BEGIN
    SET NOCOUNT ON;
    DECLARE @AppoimentsCursor CURSOR;
    DECLARE @doctor int;

    SET @AppoimentsCursor = CURSOR
    FOR SELECT
    S.doctor_id
    FROM dbo.[Services] S JOIN dbo.[Appointments] A
    ON S.service_id = A.service_id
    WHERE CAST(A.data AS DATE) = @date;

    OPEN @AppoimentsCursor;
    FETCH NEXT FROM @AppoimentsCursor INTO
    @doctor;
    SET @count = 0;

    WHILE @@FETCH_STATUS = 0
    BEGIN
        IF @doctor = @doctor_id
            SET @count = @count + 1;
        FETCH NEXT FROM @AppoimentsCursor INTO @doctor;
    END
END
```

Cu ajutorul unui cursor se retin id-urile doctorilor existenti in clinica, care au programari la data primita ca parametru. Cu ajutorul unei bucle while se parcurg medicii si daca id-ul este cel al medicului cautat

se creste valoarea variabilei @count. Variabila @count numara programarile unui doctor la o anumita data, aceasta fiind si valoarea de return a procedurii.

```
ALTER   PROCEDURE [dbo].[AppointmentsForClient]
    -- Add the parameters for the stored procedure here
    @client int
AS
BEGIN
    SET NOCOUNT ON;
    DECLARE @animal_name varchar(50),@animal_id int;
    DECLARE @AppData TABLE (appointment_id int,
                            animal_name varchar(50),
                            data date,
                            service varchar(50),
                            doctor_name varchar(50))
    DECLARE @AnimalCursor CURSOR;
    SET @AnimalCursor = CURSOR
    FOR SELECT
    name, animal_id
    FROM Animals
    WHERE client_id = @client;

    OPEN @AnimalCursor;

    FETCH NEXT FROM @AnimalCursor INTO
    @animal_name, @animal_id;

    WHILE @@FETCH_STATUS = 0
    BEGIN
        INSERT INTO @AppData
        SELECT A.appointment_id , @animal_name animal_name, A.data, S.name,
                D.first_name + ' ' + D.last_name AS doctor_name
        FROM Appointments A JOIN Services S ON A.service_id = S.service_id,
        Doctors D
        WHERE S.doctor_id = D.doctor_id
        AND
        A.animal_id = @animal_id;

        FETCH NEXT FROM @AnimalCursor INTO
        @animal_name, @animal_id;
    END
    CLOSE @AnimalCursor;
    DEALLOCATE @AnimalCursor;

    SELECT * FROM @AppData;
END
```

Procedura foloseste un tabel temporar(@AppData) in care salveaza informatiile ce trebuie intoarse ca rezultat. Cu ajutorul unui cursor se retin numele si id-urile animalelor inregistrate in clinica pentru clientul primit ca parametru. Cu ajutorul unei bucle while se insereaza date in tabel pentru fiecare animal realizandu-se un join intre tabela de programari si cea de proceduri.

```
ALTER PROCEDURE [dbo].[Sp_Login]
    -- Add the parameters for the stored procedure here
    @Admin_id NVARCHAR(100),
    @Password NVARCHAR(100),
    @Isvalid BIT OUT
AS
BEGIN
    SET NOCOUNT ON;
SET @Isvalid = (SELECT COUNT(1) FROM dbo.Login WHERE username = @Admin_id AND password = @Password)

SELECT  @Isvalid as N'@Isvalid'
END
```

Savu Ioana Rusalda -343C3

Procedura verifica daca credentialele cu care clientul incearca sa se autentifice sunt corecte. Procedura realizeaza un select pe tabela Login, filtrand dupa username si parola. Daca credentialele exista procedura va intoarce 1, in caz contrar, va intoarce 0.

```
ALTER PROCEDURE [dbo].[validCNP]
    -- Add the parameters for the stored procedure here
    @first_name varchar(25),
    @last_name varchar(25),
    @cnp varchar(50),
    @error varchar(50) OUT
AS
BEGIN TRY
    DECLARE @login_id int
    IF EXISTS(SELECT * FROM dbo.Clients WHERE CNP= @cnp)
        THROW 51002, 'INNCORECT CNP', 1
    ELSE
        BEGIN TRY
            SELECT @login_id = [dbo].[getClientGeneratedId]()
            INSERT INTO dbo.Clients(first_name, last_name, cnp, login_id)
            VALUES (@first_name, @last_name, @cnp, @login_id)
        END TRY
        BEGIN CATCH
            SELECT ERROR_MESSAGE() AS Error
        END CATCH
END TRY
BEGIN CATCH
    SELECT ERROR_MESSAGE() AS Error
    SET @error = ERROR_MESSAGE()
END CATCH
```

Procedura verifica unicitatea CNP-ului in cadrul tabelei Clients. In cadrul procedurii am folosit si o exceptie, aceasta fiind aruncata in momentul in care se gaseste un client cu acelasi cnp ca cel primit ca parametru. In blocul Try al exceptiei se incearca inserarea datelor despre client, iar in blocul Catch este intors un mesaj de eroare.

```
ALTER   FUNCTION [dbo].[DoctorsFrequence]
(
    -- Add the parameters for the function here
    @client int
)
RETURNS TABLE
AS
RETURN
(
    SELECT doctor_name AS Doctor_name, COUNT(*) As Freq
    FROM Appointments_History
    WHERE client_id = @client
    GROUP BY doctor_name
)
```

Functia numara de cate ori un client a fost la un doctor. Clientului i se va afisa numele doctorului si in dreptul sa numarul de programari pe care le-a realizat la acesta.

```
ALTER   FUNCTION [dbo].[getClientGeneratedId]
(
)
RETURNS int
AS
BEGIN
    DECLARE @id int;

    SELECT @id = Max(login_id)
    FROM Login;

    RETURN @id;

END
```

Functia intoarce id-ul generat pentru un client nou. Deoarece id-urile sunt generate crescator, clientul nou introdus va avea id-ul cel mai mare din tabela de Login.

```
ALTER FUNCTION [dbo].[getClientId]
(
    -- Add the parameters for the function here
    @username varchar(50)
)
RETURNS int
AS
BEGIN
    DECLARE @id int;

    SELECT @id = C.client_id FROM Clients C, Login L WHERE  L.login_id = C.login_id
    AND L.username = @username;

    RETURN @id

END
```

Functia intoarce id-ul unui client in functie de username-ul folosit. Acest lucru se face printr-un join intre tabela de Login si cea de client.

Triggere :

- Delete_animal - folosit pentru stergerea unui animal. Se vor sterge mai intai programarile asociate si apoi animalul
- AddHistory – folosit in adaugarea programarilor in tabela de istoric, dupa stergere
- Delete_client – folosit pentru stergerea unui client. Este asemanator Delete_animal. Va sterge prima data programarile animalelor associate, apoi animalele clientului, iar mai apoi clientul

Descrierea aplicatiei:

In realizarea aplicatiei am folosit .net si urmatoarele framework-uri:

- Microsoft.EntityFrameworkCore.Design
- Microsoft.EntityFrameworkCore.SqlServer
- Microsoft.AspNetCore.Session

Aplicatia este scrisa dupa modelul MVC(Model-View-Controller).

Controllere folosite:

- AnimalsController – pentru interactiunea cu tabela Animals

Savu Ioana Rusalda -343C3

- AppController – pentru interactiunea cu tabela intoarsa de procedura “AppointmentsForClient”
- AppointmentsController – pentru interactiunea cu tabela Appointments
- AppointmentsHistoriesController – pentru interactiunea cu tabela AppointmetsHistory
- ClientsController – pentru interactiunea cu datele referitoare la clienti
- DoctorFreqController - pentru interactiunea cu tabela intoarsa de procedura “DoctorFrequence”
- HomeController – pentru afisarea paginii de start
- LoginController – pentru interactiunea cu tabela Login
- ServiceViewController – pentru interactiunea cu tabela de procedure din care clientul trebuie sa aleaga
- SignUpController – pentru pagina de SignUp
- UserController – pentru afisarea actiunilor ce pot fi realizate de User

Modele folosite:

Pe langa modelele corespunzatoare fiecarei tabele din baza de date am mai generat urmatoarele modele:

- App.cs care continue attribute specifice coloanelor din tabelul intors de procedura “AppointmentsForClient”
- IntReturn.cs care are un atribut de tip int pentru valoarea de return a procedurilor de verificare
- DoctorFreq.cs care continue attribute specifice coloanelor din tabelul intors de procedura “DoctorFrequence”
- ViewServiceModel.cs care continue attribute specifice coloanelor din tabelul de alegere al procedurilor(nume procedura- nume doctor care realizeaza procedura)

View-uri folosite:

- Pagina de Home – realizata in Index.cshtml (folder Home)
- Pagina de Login – realizata in Index.cshtml (folder Login)
- Pagina de SingUP – realizata in Index pentru intoducerea de username si parola
  Pentru introducere date client se face redirectare catre pagina Create.cshtml din folderul Clients
- Pagina User – realizata in Index.cshtml (folder User) -afiseaza actiunile pe care un user le poate face dupa logare
- Pagina Appointments – realizata in Index.cshtml(folder Appointments) afiseaza programarile user-ului current. De asemenea se pot vedea detalii despre o programare cu Details.cshtml, se poate sterge o programare cu Delete.cshtml.
- Pagina Creara Programare -redirectionare catre Create.cshtml din folderul Appointments
- Pagina Animals – realizata cu Index.cshtml(folder Animals) pentru afisarea animalelor inregistrate de user-ul current. De asemenea se pot vedea detalii despre un animal cu Details.cshtml, se poate sterge un animal cu Delete.cshtml sau creearea unui animal nou cu Create.cshtml .
- Pagina History - realizata in Index.cshtml (folder AppointmentsHistories) pentru a vedea istoricul programarilor
- Pagina DoctorFreq - realizata in Index.cshtml (folder DoctorFreq) pentru a vedea frecventa cu care un client a fost la anumiti medici
- Pagina App – realizat in Index.cshtml pentru o afisare a programarilor diferita de cea default din (Appointments)

- Pagina ServiceView - realizata in Index.cshtml (folder ServiceView) pentru vizualizarea listei de proceduri din care clientul va alege pentru a face o programare(asociere nume procedura-nume medic)

Modul de conexiune la baza de date:

Conexiunea se realizeaza cu ajutorul framework-ului: Microsoft.EntityFrameworkCore.SqlServer.

Am folosit comenzile:

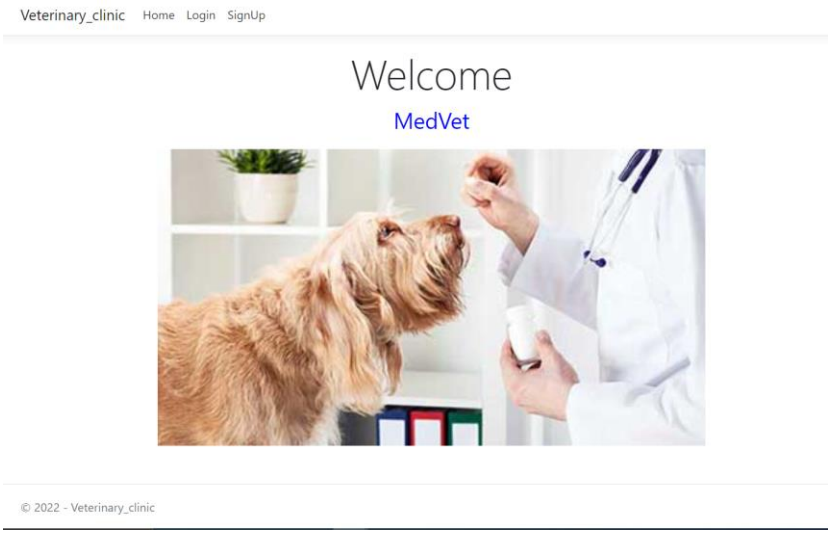
- dotnet add package Microsoft.EntityFrameworkCore.SqlServer -v 5.0.12
- dotnet ef dbcontext scaffold "Data Source=localhost,1433;Initial Catalog=Clinic;Persist SecurityInfo=True;UserID=SA;Password=parolaAiaPuternic4!" Microsoft.EntityFrameworkCore.SqlServer -o DB –force

Stringul corespunzator conexiunii se gaseste in fisierul ClinicContext.cs, acolo unde se realizeaza conexiunea si creearea modelelor folosite in aplicatie.

Rutare:

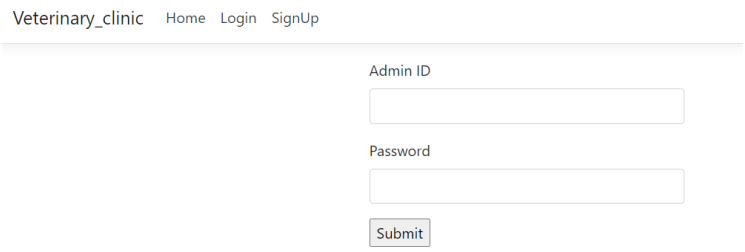
- Home Page - https://localhost:5001/
- SignUp Page1 - https://localhost:5001/signUP
- SignUp Page2 - https://localhost:5001/clients/create
- Login Page - https://localhost:5001/login
- User Page - https://localhost:5001/User?username=<name>
- Appointments Page - https://localhost:5001/App
- Details Appointments Page - https://localhost:5001/App/Details/<id>
- Delete Appointments Page - https://localhost:5001/Appointments/delete/<id>
- Animals Page - https://localhost:5001/Animals
- Edit Animal Page - https://localhost:5001/Animals/Edit/<id>
- Details Animal Page - https://localhost:5001/Animals/Details/<id>
- Delete Animal Page - https://localhost:5001/Animals/Delete/<id>
- Make Appointments - https://localhost:5001/ServiceView
- Create Appointments - https://localhost:5001/Appointments/create/<id>
- History Page - https://localhost:5001/AppointmentsHistories
- History Doctor Page - https://localhost:5001/DoctorFreq

Home Page:

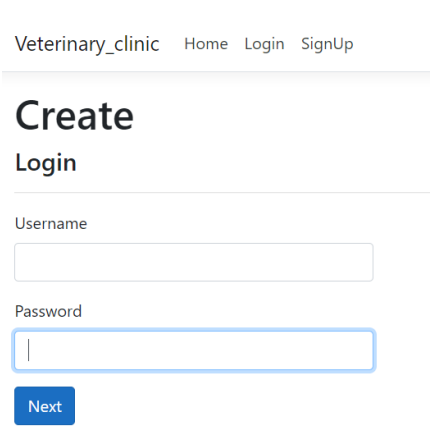


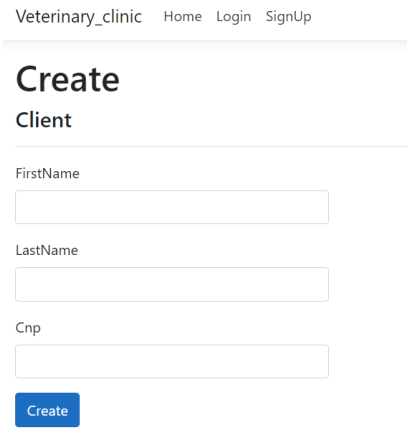
Savu Ioana Rusalda -343C3

Login Page:

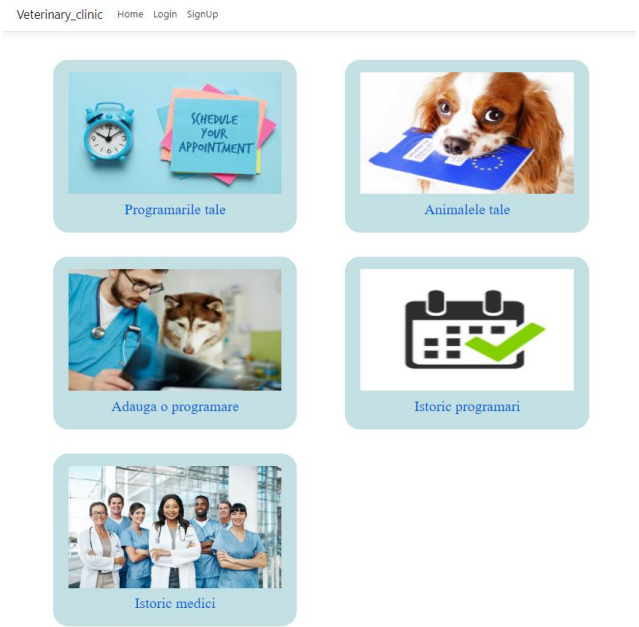


SignUp Page1&2



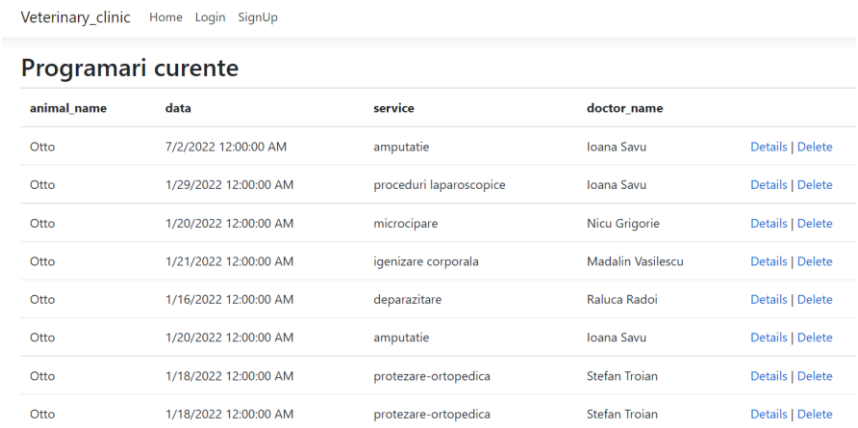


User Page

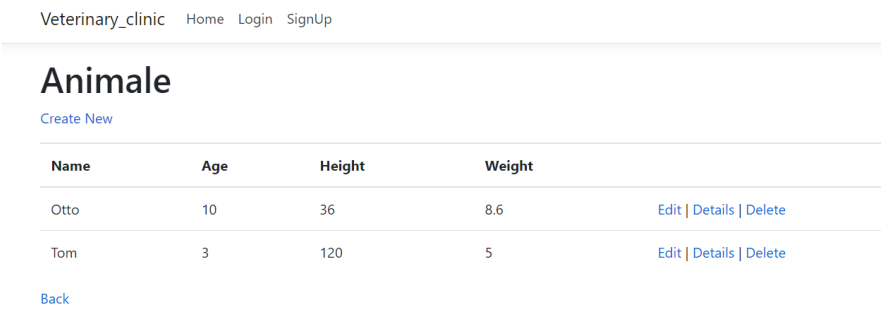


Savu Ioana Rusalda -343C3

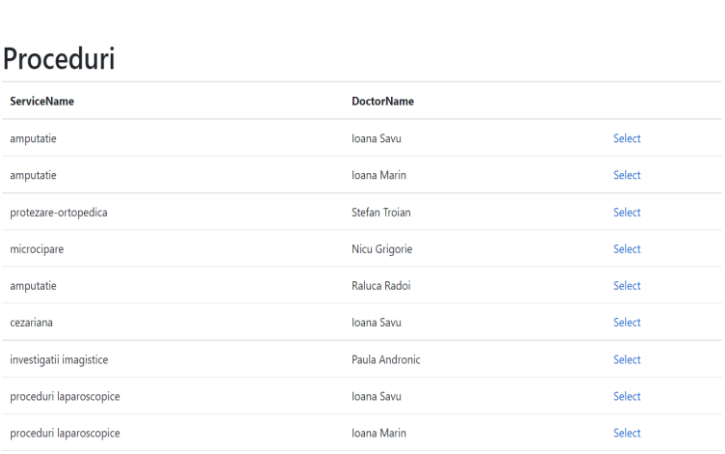
Appointments Page

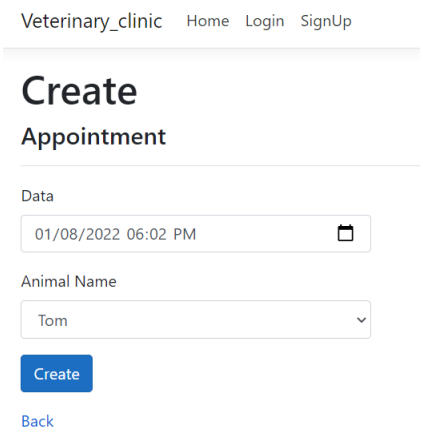


Animals Page



Make Appointment Page





Savu Ioana Rusalda -343C3

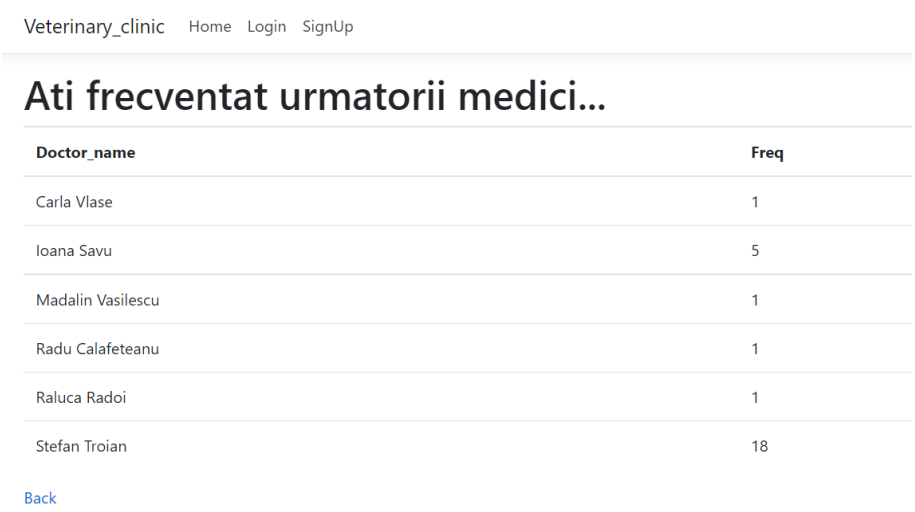
History Page

Veterinary_clinic Home Login SignUp

History

Date	AnimalName	Service	DoctorName
7/2/2022 12:00:00 AM	Bubu	3	Stefan Troian
1/29/2022 12:00:00 AM	Tom	deparazitare	Radu Calafeteanu
1/22/2022 12:00:00 AM	Tom	igenizare corporala	Madalin Vasilescu
3/16/2022 12:00:00 AM	Kim	deparazitare	Raluca Radoi
1/29/2022 12:00:00 AM	Bubu	vaccinare	Carla Vlase
1/13/2022 12:00:00 AM	Otto	protezare-ortopedica	Stefan Troian
1/18/2022 12:00:00 AM	Otto	protezare-ortopedica	Stefan Troian

Doctor History Page:



Veterinary_clinic Home Login SignUp

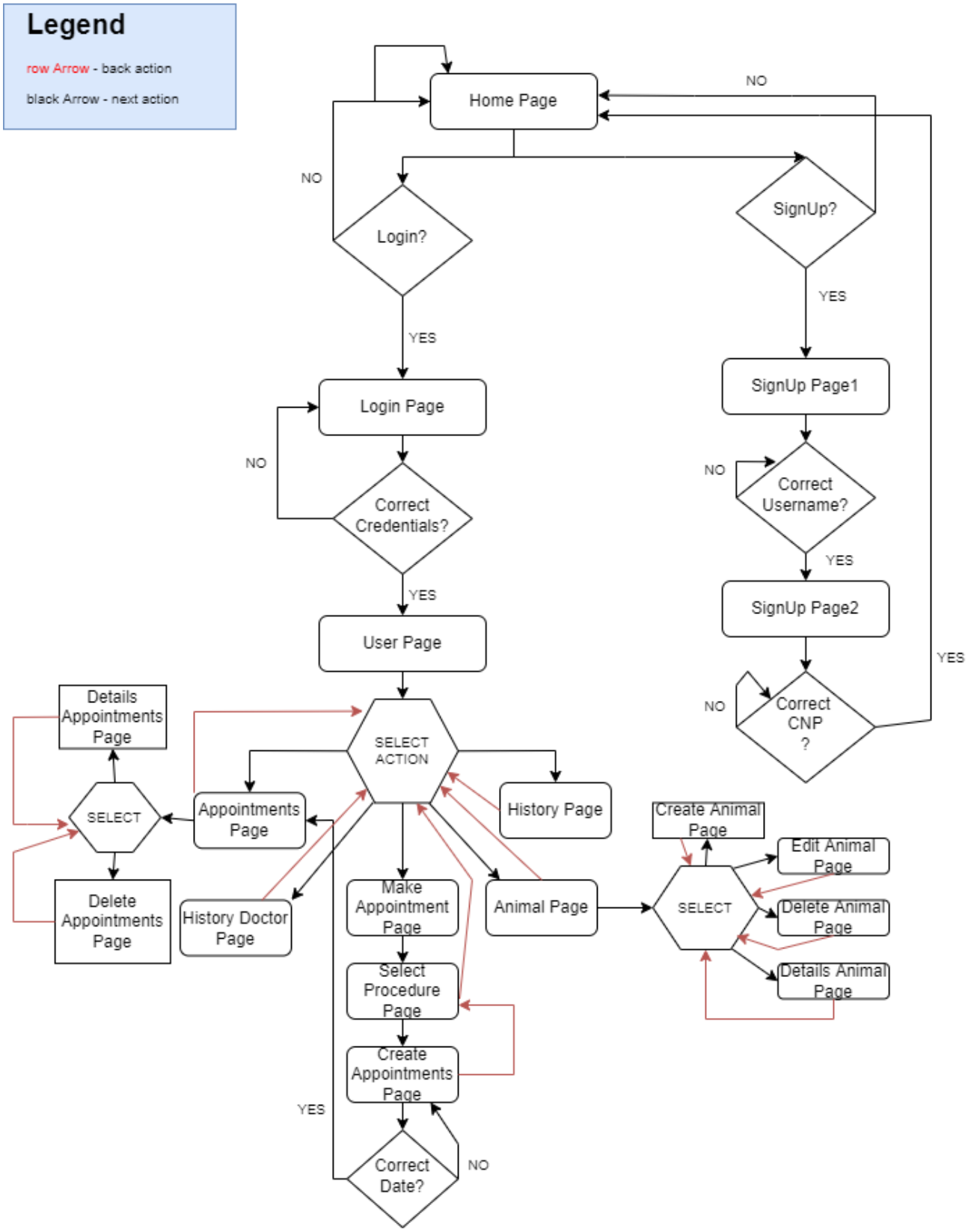
Ati frecventat urmatorii medici...

Doctor_name	Freq
Carla Vlase	1
Ioana Savu	5
Madalin Vasilescu	1
Radu Calafeteanu	1
Raluca Radoi	1
Stefan Troian	18

Back

Savu Ioana Rusalda -343C3

Workflow:



Concluzii:

Aplicatia implementeaza toate cerintele: minim 2 functii, minim 2 proceduri, minim 2 triggere, cursoare, exceptii, minim 2 rapoarte, minim 6 interfete.

Cu toate acestea, mi-as dori sa modific ulterior flow-ul aplicatiei adaugand mai multe tipuri de utilizatori, fiecare cu propriile actiuni si mai multe functionalitati. Acest lucru va duce si la modificarea functiilor din baza de date.

Spre exemplu: In momentul de fata o programare este trecuta in istoric in momentul cand este stearsa. La o viitoare actualizare as aplicatiei mi-as dori ca programarea sa treaca in istoric in momentul in care este marcata ca finalizat de catre un user cu rol de admin.

Bibliografie:

https://docs.microsoft.com/en-us/aspnet/core/?view=aspnetcore-6.0

https://stackoverflow.com/questions/43357926/calling-sql-procedure-from-c-sharp-asp-net

https://github.com/AskingAlexander/about-me-and-microsoft?fbclid=IwAR0vYn1Tx9yNhL7xXUioqVm2Ml-lAmNO_5KsiTztTTWiXB8KadO-hqwCbiI