Text

Description automatically generated

LUCRARE DE LICENȚĂ

Life Sponsors

Aplicație pentru Centrele de Donare de Sânge

**Absolvent: Blendea Alin Daniel**

**Coordonator științific: Lect. univ. dr. Vlad Monescu**

Brașov, 2022

Cuprins

[**1.** **Introducere** 5](#_Toc106571957)

[**1.1.** **Problema donării de sânge** 5](#_Toc106571958)

[**1.2.** **Scopul aplicației** 6](#_Toc106571959)

[**1.3.** **Scurtă descriere a aplicației** 6](#_Toc106571960)

[**1.4.** **Diferențe și îmbunătățiri față de aplicațiile deja existente** 7](#_Toc106571961)

[**2.** **Etapele proiectării aplicației** 8](#_Toc106571962)

[**2.1.** **Proiectarea și crearea bazei de date** 8](#_Toc106571963)

[SQL Server 2019 8](#_Toc106571964)

[SQL Server Management Studio 2018 9](#_Toc106571965)

[Protocolul Shared Memory 9](#_Toc106571966)

[Protocolul TCP/IP 10](#_Toc106571967)

[Descrierea sistemului informatic de evidență a datelor din centrele de donare 11](#_Toc106571968)

[Crearea modelului EER 11](#_Toc106571969)

[Chei primare și străine 15](#_Toc106571970)

[Normalizarea bazei de date 15](#_Toc106571971)

[Forma normală 1 (1NF) 16](#_Toc106571972)

[Forma normală 2 (2NF) 16](#_Toc106571973)

[Forma normală 3 (3NF) 16](#_Toc106571974)

[Generalizarea bazei de date 16](#_Toc106571975)

[Crearea bazei de date în SQL Server 18](#_Toc106571976)

[**2.2.** **Proiectarea diagramelor aplicației** 21](#_Toc106571977)

[Diagrama EER (Enhanced Entity-Relationship) a bazei de date 21](#_Toc106571978)

[Diagrama cazurilor de utilizare 23](#_Toc106571979)

[Diagrama de desfăsurare a aplicației desktop 25](#_Toc106571980)

[**2.3.** **Găsirea unui design generic al aplicației și al site-ului** 26](#_Toc106571981)

[Logo-ul aplicației 26](#_Toc106571982)

[Design pattern: Model-View-ViewModel (MVVM) 27](#_Toc106571983)

[Commands 28](#_Toc106571984)

[Converters 30](#_Toc106571985)

[Helpers 31](#_Toc106571986)

[Design-ul ferestrelor aplicației 32](#_Toc106571987)

[**2.4.** **Crearea interfeței grafice a aplicației desktop** 34](#_Toc106571988)

[**2.5.** **Crearea unui sistem de register și login** 35](#_Toc106571989)

[Înregistrarea în aplicație 35](#_Toc106571990)

[Simple Mail Transfer Protocol 37](#_Toc106571991)

[POP/IMAP 37](#_Toc106571992)

[Cum funcționează SMTP? 37](#_Toc106571993)

[Trimiterea de e-mail-uri prin intermediul aplicațiilor terțe 39](#_Toc106571994)

[Criptarea și decriptarea prin Advanced Encryption Standard (AES) 39](#_Toc106571995)

[Autentificarea în aplicație 40](#_Toc106571996)

[Funcția de uitare a parolei 41](#_Toc106571997)

[**2.6.** **Popularea bazei de date** 42](#_Toc106571998)

[Adăugare folosind SSMS 42](#_Toc106571999)

[Proceduri Stocate 43](#_Toc106572000)

[Legarea aplicației de baza de date prin ADO.NET 44](#_Toc106572001)

[Manipularea bazei de date direct din codul sursă 44](#_Toc106572002)

[**2.7.** **Implementarea funcționalităților** 45](#_Toc106572003)

[Data Binding 45](#_Toc106572004)

[Funcționalități specifice medicilor 45](#_Toc106572005)

[Funcționalități specifice donatorilor 46](#_Toc106572006)

[Funcționalități specifice personalului de la centrul de recoltare 47](#_Toc106572007)

[Opțiunea de urmărire a statisticilor 49](#_Toc106572008)

[**2.8.** **Crearea site-ului web cu informații pentru utilizator** 50](#_Toc106572009)

[Design-ul de bază a site-ului web 50](#_Toc106572010)

[Implementarea paginii de start 51](#_Toc106572011)

[Implementarea paginii de contact 52](#_Toc106572012)

[Implementarea paginii de anunțuri 52](#_Toc106572013)

[Implementarea paginii ghidului de utilizare 53](#_Toc106572014)

[Implementarea paginii de informații legate de aplicație 54](#_Toc106572015)

[**2.9.** **Crearea asistentului virtual** 55](#_Toc106572016)

[Crearea contului de IBM Cloud și accesarea resurselor 56](#_Toc106572017)

[Crearea și configurarea asistentului virtual 57](#_Toc106572018)

[Intents 58](#_Toc106572019)

[Nodurile de dialog 59](#_Toc106572020)

[Entități 60](#_Toc106572021)

[2.10. Integrarea asistentului virtual în site-ul web. 61](#_Toc106572022)

[**3.** **Descrierea tehnologiilor folosite** 62](#_Toc106572023)

[**3.1.** **Microsoft Visual Studio 2019** 62](#_Toc106572024)

[**3.2.** **Microsoft Visual Studio Code** 62](#_Toc106572025)

[**3.3.** **Draw.io și StarUML** 63](#_Toc106572026)

[**3.4.** **GitHub** 63](#_Toc106572027)

[**3.5.** **Trello** 64](#_Toc106572028)

[**3.6.** **Adobe Photoshop 2021** 64](#_Toc106572029)

[**3.7.** **Limbaje de programare folosite:** 64](#_Toc106572030)

[C# 65](#_Toc106572031)

[HTML 65](#_Toc106572032)

[CSS 65](#_Toc106572033)

[JavaScript 66](#_Toc106572034)

[BOOTSTRAP 66](#_Toc106572035)

[**3.8.** **IBM Cloud** 66](#_Toc106572036)

[**4.** **Dezvoltare ulterioară** 68](#_Toc106572037)

[**5.** **Concluzii** 69](#_Toc106572038)

[**6.** **Bibliografie și webografie** 70](#_Toc106572039)

# **Introducere**

## **Problema donării de sânge**

Zilnic, peste 1000 de unități de sânge sunt folosite pentru transfuzii sanguine în spitalele din România. Acest lucru, împreună cu numărul mic de donatori din țara noastră, duce la imposibilitatea acoperirii tuturor cererilor și, astfel, la pierderea unui număr considerabil de vieți omenești. La nivel mondial, doar câteva țări au un număr majoritar de persoane adulte care sunt interesate de donarea de sânge.

A group of people walking on a street

Description automatically generated with low confidence

Figura 1.1.1: Cozi formate în fața centrelor de donare din Ucraina, în urma invaziei din 2022

România se află în continuare pe ultimele locuri în Europa sau chiar în lume la capitolul donării de sânge, cu un procent de doar 2% donatori din totalul populației, comparativ cu alte state în care mai mult de 50% din populația adultă donează sânge (de exemplu, Arabia Saudită și India). Chiar dacă în situații limită atât românii cât și persoane din alte țări au demonstrat solidaritate şi simț civic, stând ore în şir la coadă pentru a dona sânge (Figura 1.1.1.), acest lucru reprezintă, de obicei, o soluție temporară a unei probleme constante. Un stoc prea mare de sânge este inutil, întrucât componentele sanguine au un termen de valabilitate relativ scurt.

Mai mult decât atât, bolnavii au în general nevoie de unități de sânge sigure, de la donatori care să îndeplinească anumite condiții înainte de donare, iar ajutarea unor anumiți bolnavi necesită prezentarea la recoltare de câte două, sau chiar de câte trei ori pe an.

## **Scopul aplicației**

Lucrarea de față își propune să ușureze procesul prin care spitalele din România obțin sânge pentru transfuziile sanguine, de a scăpa de cozile de la centrele de donare prin implementarea unui sistem de programări și de a încuraja cât mai multe persoane să doneze sânge.

## **Scurtă descriere a aplicației**

Aplicația prezentată este formată din trei componente:

* + Aplicație desktop
  + Bază de date
  + Site web

Aplicația va permite medicilor din spitale să trimită cereri de anumite componente sanguine, donatorii vor completa un chestionar medical, iar personalul de la centrul de recoltare va putea verifica dacă aceștia sunt apți sau nu pentru donare, urmând să îi programeze într-o anumită dată. Odată programați, donatorii sunt notificați prin mail, pot vedea data în care trebuie să se prezinte la centrul de recoltare și pot vedea un istoric al donărilor efectuate pe un anumit cont de donator.

De asemenea, orice tip de utilizator va putea accesa secțiunea de statistici, va putea deschide site-ul aplicației pentru a se informa, sau poate utiliza funcția de chatbot a aceluiași site pentru a pune întrebări.

Graphical user interface

Description automatically generatedGraphical user interface, website

Description automatically generated

Figura 1.3.1: Capturi de ecran ale ferestrei de start a aplicației și a unei pagini a site-ului web

## **Diferențe și îmbunătățiri față de aplicațiile deja existente**

În comparație cu celelalte deja existente, aplicația pentru centrele de donare de sânge realizată în cadrul acestei lucrări oferă posibilitatea utilizatorilor de a vedea statistici cu privire la:

* volumul total de sânge donat de la începerea folosirii aplicației,
* volumul fiecărei grupe de sânge donate,
* bugetul fiecărei companii care oferă anumite recompense donatorilor de sânge, cât și ce fel de recompense s-au oferit până la momentul vizualizării statisticilor
* contul fiecărui donator. Pentru vizionarea datelor reținute în fiecare cont, este necesară o autentificare prin 2 factori, folosind parola contului și CNP-ul persoanei care dorește să își vadă informațiile. Astfel, nimeni nu poate vedea datele altei persoane.

Un alt punct forte al aplicației este reprezentat chiar de către posibilitatea înregistrării și autentificării tuturor utilizatorilor folosind adresa de e-mail personală. Mail-urile sunt trimise automat iar confirmarea conturilor se face introducând un cod unic trimis către adresa utilizată. Parolele reținute în baza de date sunt criptate, fapt care conferă o mai bună siguranță a sistemului de autentificare.

De altfel, aplicația desktop este creată într-un mod care permite utilizarea acesteia la nivel național, dat fiind faptul că orice spital din România poate fi adăugat în baza de date.

# **Etapele proiectării aplicației**

În ceea ce privește etapele proiectării aplicației, s-a ținut cont de următoarea planificare, fiecare dintre functionalități fiind implementate pe rand.

## **Proiectarea și crearea bazei de date**

În acest subcapitol, voi prezenta sistemul de evidență al programărilor, al conturilor și al celorlalte date necesare funcționării aplicației. Pentru implementarea acestui sistem, s-a folosit o bază de date, care este o colecție de date creată și menținută computerizat, care permite operațiile CRUD. Datele reținute sunt corelate din punct de vedere logic, reflectând aspecte ale lumii reale și fiind destinate unui anumit grup de utilizatori.

Printre avantajele utilizării unei baze de date in aplicația creată, se numără volumul mare de date care poate fi stocat, posibilitatea mai multor utilizatori de a modifica și adăuga date, dar și faptul că datele de care avem nevoie pot fi găsite intr-un mod ușor si rapid, datorită faptului că bazele de date pot fi sortate.

### SQL Server 2019

Pentru stocarea și gestionarea bazei de date, am folosit SQL Server 2019, care este un Sistem de Gestiune a Bazelor de Date Relaționale (SGBDR) dezvoltat de Microsoft. În comparație cu un SGBD, SQL Server 2019 este mai eficient, permițând și stocarea unui volum mai mare de date. De asemenea, într-un SGBDR, datele sunt reținute în tabele relaționale, pe când un SGBD poate reține datele doar în fișiere. O reprezentare grafică a acestui lucru poate fi observată în imaginea următoare.

Diagram

Description automatically generated

Figura 2.1.1: stocarea datelor într-un SGBD în comparație cu un SGBDR

### SQL Server Management Studio 2018

SQL Server Management Studio (SSMS) este un mediu integrat de dezvoltare și gestionare a oricărei infrastructuri SQL. Folosind SSMS, am putut configura, gestiona, administra si dezvolta fiecare componentă necesară stocării bazei de date în SQL Server 2019. De asemenea, SSMS suportă și instanțe ale altor Sisteme de Gestiune a Bazelor de Date Relaționale, cum ar fi Azure SQL Database, Azure SQL Managed Instance, SQL Server on Azure VM, și Azure Synapse Analytics.

În continuare voi prezenta cele două protocoale folosite în conectarea la baza de date a aplicației pentru centrele de donare de sânge.

### Protocolul Shared Memory

SQL Server Shared Memory Protocol este folosit de către SSMS pentru a se conecta la o instanță de SQL Server existentă pe același computer. Acest protocol este folosit, în general, pentru a testa și a vedea dacă alte protocoale de rețea au anumite probleme.

Protocolul Shared Memory este cel mai simplu de utilizat, întrucât nu necesită configurări de rețea în plus.

### Protocolul TCP/IP

Pentru a se conecta la SQL Server, SSMS folosește protocolul TCP/IP. Acesta permite mai multor calculatoare să se conecteze și să comunice între ele, permițând primirea si executarea mai multor comenzi în același timp.

Printre configurările necesare utilizările acestui protocol se numără și selectarea unui port pe care să îl asculte. Implicit, acest port este TCP 1433.

Schimbarea protocolului TCP folosit pentru conectarea la instanța de SQL Server implică restartarea servciului de SQL Server, pentru ca schimbările să își facă efectul. Acest lucru poate fi făcut din nodul SQL Server Services din fereastra consolei SQL Server Configuration Manager, apăsând pe butonul de Restart Service din partea de sus (Figura 2.1.2.).

Graphical user interface, text, application, chat or text message

Description automatically generated

Figura 2.1.2: Restartarea instanței de SQL Server folosind SQL Server Configuration Manager

Pentru crearea și gestionarea bazei de date am folosit interfața grafică a SQL Server Management Studio 2018, conectându-mă inițial la instanța de SQL Server prin Shared Memory și trecând, mai apoi, la protocolul TCP/IP.

### Descrierea sistemului informatic de evidență a datelor din centrele de donare

Fiecare donare de sânge, efectuată sau doar programată, este identificată printr-un id unic, data programării donării, proprietatea de a fi sau nu efectuată, cantitatea de sânge donat și CNP-urile donatorului și, eventual, al pacientului căruia i se donează sângele.

Conturile reținute în baza de date sunt identificare printr-un id unic, un mail și o parolă criptată, iar despre utilizatori se vor reține Codurile Numerice Personale pentru identificarea fiecăruia în parte, împreună cu alte date fără de care aplicația nu poate funcționa la potențial maxim.

De asemenea, se rețin date de identificare pentru fiecare chestionar medical completat de către donatori, pentru fiecare cerere de donare trimisă de către medici și pentru fiecare pungă de sânge formată.

Recompensele primite de către donatori sunt și ele stocate în baza de date, pentru o mai bună gestionare a acestora.

### Crearea modelului EER

În urma studierii descrierii de mai sus, se disting următoarele entități:

* Cont
* Donator
* Medic
* Personal de recoltare
* Pacient
* Cerere de donare
* Chestionar medical
* Donare
* Spital
* Recompensă
* Pungă de sânge
* Componentă de sânge

Astfel, se poate forma o serie de tipuri de relații între entitățile sistemului, serie pe are o voi trece în tabelul de mai jos:

|  |  |  |  |
| --- | --- | --- | --- |
| **Tip de entitate** | **Tip de relație** | **Tip de entitate** | **Cardinalitate** |
| Cont | aparține de | Donator | 1:N |
| Cont | aparține de | Medic | 1:N |
| Cont | aparține de | Personal de recoltare | 1:N |
| Medic | completează | Cerere de donare | 1:N |
| Medic | aparține de | Spital | 1:N |
| Donator | completează | Chestionar medical | 1:N |
| Donator | efectuează | Donare | 1:N |
| Donator | donează | Pungă de sânge | 1:N |
| Personal de recoltare | adaugă | Punga de sânge | 1:N |
| Pacient | aparține de | Spital | 1:N |
| Pacient | primește | Donare | 1:N |
| Pungă de sânge | conține | Componente sanguine | 1:N |
| Cerere de donare | aparține de | Spital | 1:N |
| Donare | oferă | Recompensă | 1:N |

Tabel 2.1.1: Relații între entități

Mai departe voi defini entitățile, precizând atributele și domeniile fiecăreia. Termenii folosiți pentru domenii au următoarele semnificații:

* integer – tip de dată reprezentând un număr întreg
* varchar(x) – tip de dată reprezentând un șir de caractere alfanumerice, de dimensiune x
* date – tip de dată reprezentând o dată calendaristică
* bit – tip de dată binară, care poate lua valorile true sau false
* nvarchar(x) - tip de dată reprezentând un șir de caractere alfanumerice, de dimensiune x, dar de dimensiune mai mare decât varchar

Definirile fiecărei entități pot fi găsite în tabelul următor. Alături de numele atributelor, am trecut (PK) dacă acesta este o cheie primară în entitatea respectivă sau (FK) dacă reprezintă o cheie străină.

|  |  |  |
| --- | --- | --- |
| **Tip entitate** | **Atribute** | **Domeniu** |
| Cont | id\_cont **(PK)** | integer |
| email | varchar(30) |
| parola | nvarchar(200) |
| type | varchar(20) |
| Donator | cnp\_donator **(PK)** | varchar(13) |
| nume | varchar(50) |
| domiciliu | varchar(200) |
| resedinta | varchar(200) |
| email | varchar(30) |
| telefon | varchar(13) |
| grupa\_sanguina | varchar(3) |
| id\_cont **(FK)** | integer |
| Medic | id\_medic **(PK)** | integer |
| nume | varchar(50) |
| email | varchar(30) |
| id\_spital **(FK)** | integer |
| id\_cont **(FK)** | integer |
| Personal\_Recoltare | id\_personal **(PK)** | integer |
| nume | varchar(50) |
| email | varchar(30) |
| id\_cont **(FK)** | integer |
| Pacient | cnp\_pacient **(PK)** | varchar(13) |
| nume | varchar(50) |
| grupa\_sanguina | varchar(3) |
| high\_priority | bit |
| id\_spital **(FK)** | integer |
| Cerere\_Donare | id\_cerere **(PK)** | integer |
| status | varchar(20) |
| grupa\_sanguina | varchar(3) |
| trombocite | bit |
| globule\_rosii | bit |
| plasma | bit |
| id\_medic **(FK)** | int |
| cnp\_pacient **(FK)** | varchar(13) |
| Chestionar medical | id\_chestionar **(PK)** | integer |
| greutate | varchar(50) |
| puls | varchar(50) |
| tensiune | varchar(50) |
| interventii\_chirurgicale\_recente | bit |
| sarcina | bit |
| consum\_grasimi | bit |
| tratament | bit |
| alte\_boli | varchar(50) |
| grupa\_sanguina | varchar(3) |
| aprobat | bit |
| cnp\_donator **(FK)** | varchar(13) |
| cnp\_pacient **(FK)** | varchar(13) |
| Donare | id\_donare **(PK)** | integer |
| data | date |
| isDone | bit |
| email | varchar(13) |
| grupa\_sanguina | varchar(30) |
| cantitate\_ml | varchar(3) |
| cnp\_donator **(FK)** | integer |
| cnp\_pacient **(FK)** | varchar(13) |
| id\_beneficiu **(FK)** | integer |
| Spital | id\_spital **(PK)** | integer |
| denumire | varchar(50) |
| judet | varchar(30) |
| Benefits | id\_beneficiu **(PK)** | integer |
| denumire | varchar(100) |
| nr\_total | integer |
| nr\_ramase | integer |
| cost\_per\_buc | integer |
| Componente\_Sange | id\_punga **(PK)** | integer |
| trombocite | bit |
| globule\_rosii | bit |
| plasma | bit |
| Punga\_Sange | id\_punga **(PK)** | integer |
| data | date |
| grupa\_sanguina | varchar(3) |
| cnp\_donator **(FK)** | varchar(13) |
| id\_personal **(FK)** | integer |
| cnp\_pacient **(FK)** | integer |

Tabel 2.1.2: Entitățile bazei de date

### Chei primare și străine

Se numește cheie o coloană sau un set de coloane care poate fi utilizat pentru a identifica sau a accesa un rând sau un set de rânduri într-o bază de date.

Cheia primară este coloana care definește în mod unic un rând dintr-un tabel al bazei de date relaționale. Aceasta impune constrângerea implicita NOT NULL, ceea ce înseamnă că orice coloană definită ca fiind cheie principală nu poate avea valori NULL în ea.

Cheia străină, pe de altă parte, reprezintă o constrângere referențială între două tabele. Aceasta definește ocoloană sau un set de coloane într-un tabel numit tabelul de referință, care se referă la un set de coloane dintr-un alt tabel, numit tabelul referit. Diferența dintre cele două chei este că cea principală identifică unic un rând din tabel, pe când cea străină este o referință către o cheie primară dintr-un tabel extern.

### Normalizarea bazei de date

La momentul creării unei baze de date, există posibilitatea ca, datorită felului în care au fost proiectate unele tabele, să apară anumite anomalii de introducere, ștergere sau actualizare a datelor. Prin normalizarea unei baze de date urmărim minimizarea acestor anomalii.

În momentul proiectării bazei de date pentru lucrarea de față, am ținut cont de regulile celor 3 forme normale încă de la început, minimizând din start redundanța datelor reținute.

### Forma normală 1 (1NF)

Pentru ca baza de date să fie adusă la forma normală 1, toate atributele oricărei relații trebuie să fie domenii de valori atomice, fără să mai poată fi descompuse. Astfel, un exemplu de aducere la forma normală 1 poate fi reprezentat de situația în care există mai mulți donatori cu același nume. În acest caz, este necesară introducerea unui identificator unic pentru fiecare donator, acesta fiind chiar CNP-ul persoanei respective.

### Forma normală 2 (2NF)

Aducerea bazei de date la forma normală 2 presupune, în primul rând, aducerea la 1NF și căutarea dependențelor parțiale de cheia principală, adică atributele care depind funcțional de un subset de atribute a cheii primare. Dacă cheia primară este compusă dintr-un singur atribut, atunci relația este în 2NF. Dacă există dependențe parțiale, se vor șterge atributele care depind parțial de cheia principală și se creează o relație nouă care să se compună din atributele șterse împreună cu determinantul lor.

### Forma normală 3 (3NF)

Pentru a aduce baza de date în forma normală 3, aceasta trebuie să se afle în 2NF iar în cazul existenței dependenței tranzitive, se șterg coloanele care sunt tranzitiv dependente de cheia primară și se creează o relație nouă cu aceste coloane, împreună cu determinantul lor (cheia primară).

Dependența tranzitivă poate fi definită în felul următor: dacă atributele A, B și C sunt în relațiile A -> B și B -> C, atunci spunem că atributul C este dependent tranzitiv de atributul A, via B.

### Generalizarea bazei de date

Generalizarea reprezintă procesul de minimizare a diferențelor dintre entități, pentru identificarea caracteristicilor comune.

Procesul de generalizare este o aproximare bottom-up a superclaselor, din subclasele originale. Astfel, generalizarea este inversa specializării. De exemplu, pentru tipurile de entități Donator, Medic și Personal de recoltare se observă că unele atribute ale lor caracterizează toate trei tipurile: nume, email și id\_cont. De aici rezultă necesitatea creării unei superclase care să conțină toate atributele comune celor trei tipuri.

În cazul sistemului de față, voi introduce superclasa Utilizator, care să aibă cele trei entități cu atribute comune.

Graphical user interface, diagram, application, Teams

Description automatically generated

Figura 2.1.3: Reprezentarea grafică a superclasei Utilizator, în relație cu subclasele sale

Subclasele sunt non disjuncte, întrucât un membru al superclasei aparține mai multor subclase. Astfel, se aplică regula disjuncției. De asemenea, se aplică și regula participării, care este, în acest caz, totală, deoarece toți membrii superclasei sunt și membri ai subclaselor.

Voi prezenta diagrama EER a bazei de date normalizate în capitolul 2.2. Proiectarea diagramelor aplicației.

### Crearea bazei de date în SQL Server

În continuare, voi prezenta câteva comenzi SQL folosite pentru crearea tabelelor și a cheilor străine. Baza de date a fost creată folosind atât interfața grafică a SQL Server Management Studio, cât și comenzi SQL executate direct din terminalul de comandă.

Pentru crearea bazei de date, a fost nevoie să urmez pașii descriși mai jos:

* Deschiderea SSMS-ului,
* Conectarea la instanța de SQL Server,
* Click Dreapta pe folderul Databases din instanța de SQL Server, folder găsit în Object Explorer,
* Click pe New Database (Figura 2.1.4.),
* Introducerea numelui urmată de Click pe butonul Ok.

Graphical user interface, application

Description automatically generated

Figura 2.1.4: Crearea unei baze de date folosind interfața grafică a SSMS

Odata ce am creat baza de date, am putut crea tabelele prin următorii pași:

* Click dreapta pe folderul Tables din baza de date a aplicației,
* Click pe New -> Table (Figura 2.1.5.),
* Introducerea coloanelor și a tipului de date reținut în acestea (Figura 2.1.6.),
* Salvarea tabelului prin Ctrl + S, urmat de introducerea numelui acestuia.

Graphical user interface, application

Description automatically generated

Figura 2.1.5: Crearea unui tabel folosind interfața grafică a SSMS

Graphical user interface, application, table

Description automatically generated

Figura 2.1.6: Crearea unui tabel folosind interfața grafică a SSMS

Voi prezenta mai departe o altă metodă de creare a tabelelor în baza de date și legarea de relații între ele, folosind comenzi SQL. Sintaxele acestora sunt următoarele:

* Selectarea bazei de date curente:
  + USE DATABASE nume\_baza\_de\_date;
* Crearea unui tabel:
  + CREATE TABLE nume\_tabel (coloana1 tip\_data, coloana2 tip\_data, …);
* Crearea relației între două tabele prin inserarea unei chei străine în tabelul t1 din t2:
  + ALTER TABLE t1 ADD FOREIGN KEY (t1\_key) REFERENCES t2 (t2\_key);

În continuare voi scrie comenzile pe care le-am folosit pentru a crea doua dintre tabelele bazei de date și crearea relației dintre ele:

CREATE TABLE Cont(

id\_cont INT PRIMARY KEY,

email VARCHAR(30),

parola NVARCHAR(200),

type VARCHAR(20)

);

CREATE TABLE Personal\_Recoltare(

id\_personal INT PRIMARY KEY,

nume VARCHAR(50),

id\_cont INT NOT NULL

);

ALTER TABLE Personal\_Recoltare ADD FOREIGN KEY (id\_cont) REFERENCES Cont(id\_cont);

## **Proiectarea diagramelor aplicației**

În crearea aplicației pentru centrele de donare de sânge, am considerat foarte importantă cunoașterea încă din start a numărului de ferestre și pagini necesare atât aplicației desktop, cât și site-ului web, a felului în care utilizatorii interacționează cu aplicația și a modului în care diferite componente ale aplicației comunică între ele.

Astfel, un al doilea pas important pe care îl voi prezenta în această lucrare va fi proiectarea câtorva diagrame ale aplicației, care m-au făcut să înțeleg mai bine ceea ce am de făcut mai departe. Pentru crearea acestor diagrame am folosit două tool-uri gratuite pe care le voi prezenta în capitolul 3: Draw.io și StarUML.

### Diagrama EER (Enhanced Entity-Relationship) a bazei de date

Acest tip de diagramă ajută la crearea și menținerea unei baze de date, având un rol important în modelarea datelor. Folosirea ei aduce câteva beneficii, printre care se numără și interpretarea și înțelegerea mai ușoară a bazei de date.

Odată ce am normalizat baza de date a aplicației, am putut proiecta diagrama EER a acesteia folosind informațiile descrise anterior. Baza de date conține un număr de 12 entități care comunică între ele prin intermediul a 15 relații.

Diagrama poate fi găsită pe pagina următoare.

Graphical user interface, application, Teams

Description automatically generated

Figura 2.2.1: Diagrama EER a bazei de date

### Diagrama cazurilor de utilizare

Interpretând această diagramă, putem observa tipurile de utilizatori ai aplicației și felul în care aceștia interacționează cu diferite funcționalități. În acest caz, este necesară proiectarea a două diagrame de utilizare, atât pentru aplicația desktop cât și pentru site-ul web.

A picture containing bubble chart

Description automatically generated

Figura 2.2.2: Diagrama cazurilor de utilizare ale aplicației desktop

Diagram

Description automatically generated

Figura 2.2.3: Diagrama cazurilor de utilizare ale site-ului web

Cazurile de utilizare din diagrame sunt reprezentate de un Use Case Shape ( ), iar utilizatorii aplicației sunt reprezentați de câte un Actor Shape (stickman).

Rolurile jucate de obiectele externe subsistemelor sunt reprezentate de actori, iar cazurile de utilizare sunt seturile de evenimente care au loc atunci cand actorii interacționează cu acestea. Asocierea, reprezentată prin săgeți, ilustrează participarea actorilor în cazurile de utilizare.

Diagrama cazurilor de utilizare nu afișează ordinea în care se efectuează operațiile, ci rezumă doar modul în care utilizatorii interacționează cu sistemul.

În lucrarea de față, sistemul este format din două subsisteme legate între ele: Blood Donor App și Blood Donor Site, reprezentate printr-un chenar dreptunghic. În al doilea subsistem, tipurile de utilizator au fost generalizate și reprezentate printr-un actor intitulat “User”.

### Diagrama de desfăsurare a aplicației desktop

Cu ajutorul diagramei de desfășurare, am reprezentat arhitectura fizică a lucrării de față. Aceasta este formată dintr-un server de baze de date (instanța de SQL Server) și din clienții reprezentați de calculatoarele pe care rulează aplicația desktop. Conexiunea este făcută prin TCP/IP iar clienții pot să acceseze site-ul web prin intermediul internetului.

De asemenea, chatbot-ul integrat în site se află în IBM Cloud, pe care îl voi descrie mai amplu în capitolul 3.Descrierea tehnologiilor folosite.

Diagram

Description automatically generated

Figura 2.2.4: Diagrama de desfășurare a Aplicației pentru Centrele de Donare de Sânge

## **Găsirea unui design generic al aplicației și al site-ului**

Odată ce am creat arhitectura celor două subsisteme, am putut veni cu o idee de design care să trimită, prin culori, contraste și background-uri, la sistemul medical și la importanța pe care această aplicație ar putea să o aibă în viețile oamenilor.

### Logo-ul aplicației

Iconița este simbolică, reprezentând grafic două mâini, cu una dintre ele oferindu-i celeilalte o picătură de sânge. Astfel, atât donatorii cât și pacienții se pot regăsi în logo-ul aplicației.

A picture containing text

Description automatically generated

Figura 2.3.1: Logo-ul Aplicației pentru Centrele de Donare de Sânge

Ideea logo-ului a fost preluată de pe un site de stock images (DepositPhotos) și modificat în Adobe Photoshop în funcție de nevoia unui contrast mai inchis sau mai deschis al culorilor.

Imaginea prezentată în Figura 2.3.1. este folosită în mai multe locuri din aplicație, fiind inclusiv iconița aplicației desktop, cea a site-ului și în diverse alte locuri în partea de sus a ferestrelor si a paginilor.

Titlurile ferestrelor sunt alese în așa fel încât orice utilizator să își poată da seama în ce parte a aplicației sau a site-ului doar uitându-se la titlu.

### Design pattern: Model-View-ViewModel (MVVM)

Un design pattern reprezintă o soluție generală a unei probleme des apărute în design-ul sistemelor software. Design pattern-ul este, de obicei, reprezentat de o descriere sau de un șablon care se poate urma pentru a rezolva problema respectivă.

Există trei tipuri de design patterns folosite în software design:

* creaționale,
* comportamentale,
* de structură.

Cele creaționale se referă la instanțierea claselor, pe când cele comportamentale fac referire la comunicarea dintre obiectele claselor.

În lucrarea de față, însă, mă voi axa pe descrierea unui design pattern de structură pe care l-am folosit în dezvoltarea aplicației desktop, și anume Model-View-ViewModel (MVVM).

MVVM, la fel ca celelalte design patterns de structură, pune accentul pe compoziția claselor și ale obiectelor. Astfel, prin MVVM putem descompune aplicația în mai multe componente. În cazul acestei aplicații, MVVM este o soluție excelentă pentru separarea interfeței grafice de partea de back-end.

Printre avantajele folosirii acestui design pattern, se numără:

* modularitatea aplicației,
* dezvoltarea mai ușoară, în paralel a interfeței grafice,
* minimizarea codului de business logic din spatele interfeței grafice,
* testarea mai ușoară a aplicației.

Prin Model ne referim la partea de back-end a unei componente, iar prin View facem referire la UI. Prin implementarea MVVM, cele două nu sunt dependente una față de cealaltă.

Clasă de tip ViewModel este cea care face legătura între cele două. Aceasta este creată specific pentru a întâlni cerințele View-ului, conținând anumite proprietăți ale acestuia.

Poziționarea claselor aplicației în fișiere se poate observa în imaginea de mai jos (Figura 2.3.2.):

A screenshot of a computer

Description automatically generated with low confidence

Figura 2.3.2: Structura aplicației desktop, folosind MVVM.

Dupa cum se poate observa, am folosit și comenzi, helpere și convertoare, a căror folosință o voi explica în continuare.

### Commands

În C#, comenzile sunt folosite ca o alternativă a apelării anumitor funcții. De multe ori, în momentul în care se face click pe un buton din interfața grafică este nevoie de executarea unei funcții cu parametri. Aceștia pot fi ușor transmiși funcției respective prin utilizarea atributului “CommandParameter” în fișierul .xaml.

Pentru a folosi aceste comenzi, însă, este necesară în primul rând implementarea lor. Voi explica, în continuare clasa de comandă creată și utilizată de mine, RelayCommand.

Text

Description automatically generated

Figura 2.3.3: O parte din clasa RelayCommand.

Așa cum se poate observa în Figura 2.3.3, clasa moștenește proprietățile interfeței ICommand și are ca membri o acțiune și un predicat. În constructor, se inițializează membrul commandTask cu acțiunea care trebuie executată în momentul apelării comenzii. Prin cadrul comenzilor se poate verifica, de asemenea, dacă o acțiune poate fi executată sau nu.

Mai jos voi insera imagini cu modul în care aceste comenzi sunt reținute ca proprietăți în ViewModels și felul în care pot fi legate de anumite elemente de UI.

Text

Description automatically generated

Figura 2.3.4: Comandă membră al ViewModel-ului de cont de donator, prezentând apelarea metodei de Login din cadrul Model-ului respectiv.

Text

Description automatically generated

Figura 2.3.5: Legarea comenzii de Login de un buton din UI. Se observă trimiterea unui parametru multiplu, format din e-mail și parolă, folosind un converter.

### Converters

Prin converters, valorile introduse in UI pot fi trimise către back-end, formând instanțe ale claselor de tip ViewModel. Astfel, convertoarele fac comunicarea între interfața grafică (Views) și ViewModels.

Mai jos voi insera lista convertoarelor create (Figura 2.3.6.). Acestea moștenesc interfața IMultiValueConverter și tot ce fac este să returneze o instanță a ViewModel-ului necesar, folosind valorile introduse luate din UI.

Text

Description automatically generated

Figura 2.3.6: Convertoare create pentru a face legătura între Views și ViewModels.

### Helpers

Clasele de tip Helper ajută, în general, la rezolvarea anumitor probleme. Un exemplu potrivit pentru această situație ar fi cazul în care avem nevoie să aflăm rădăcina pătrată a unui număr. Pentru a nu implementa acest algoritm de fiecare dată când avem nevoie de el, putem crea o metodă Helper pe care să o apelăm atunci când calculăm acest lucru.

În aplicația desktop din cadrul acestei lucrări, am creat două clase de tip Helper, așa cum apar în structura din Figura 2.3.7.

Text

Description automatically generated with medium confidence

Figura 2.3.7: Clase Helper create în cadrul aplicației

Am implementat clasa VisualStudioProvider pentru a putea obține calea către soluția proiectului direct din cod, oricare ar fi aceasta. Acest lucru m-a ajutat în momentul în care am făcut legătura între aplicația desktop și site-ul web.

BaseVM, pe de altă parte, este o clasă moștenită de celelalte ViewModels, care ajută la implementarea interfeței INotifyPropertyChanged în acestea.

### Design-ul ferestrelor aplicației

Așa cum am menționat anterior, am încercat să aleg un design care să trimită, prin culori, contraste și background-uri, la sistemul medical. Astfel am ales pentru ferestrele principale o imagine de fundal de culoare albastră iar pentru ferestrele secundare de schimbare a parolei și a introducerii de coduri de confirmare, a unei imagini de fundal de o culoare mult mai deschisă, așa cum se poate observa în imaginile de mai jos (Figurile 2.3.8 și 2.3.9).



Figura 2.3.8: Imaginea de fundal a ferestrelor principale

Background pattern

Description automatically generated

Figura 2.3.9: Imaginea de fundal a ferestrelor secundare

În ceea ce privește butoanele din cadrul aplicației desktop, am folosit o aplicație web numită MakeText.io pentru a genera imagini cu extensia .png care să conțină doar textul introdus, într-un format înfrumusețat, cu fundal transparent. Aceste imagini au fost mai apoi setate drept content al butoanelor, înlocuind, astfel, textul lor. Aplicația este gratuită, ușor de utilizat și se poate alege dintre mai multe stiluri de text. Pentru aplicația de față, am folosit un stil de text de culoare roșie, din motive evidente.

Am schimbat, de asemenea, câteva proprietăți ale butoanelor pentru a le conferi o aparență diferită față de cea implicită. Proprietățile cărora le-am schimbat valorile au fost următoarele:

* Proprietatea “CornerRadius” la valoarea “12”, pentru rotunjirea colțurilor,
* Proprietatea “BorderBrush” la valoarea “Black”, pentru setarea culorii chenarului,
* Proprietatea “BorderThickness” la valoarea “2”, pentru setarea grosimii chenarului.

De asemenea., pentru butonul care trimite utilizatorul către site-ul aplicației am folosit drept content o imagine ce trimite la ideea de “help”: un buton rotund cu un semn al întrebării în interior.

Câteva butoane din cadrul aplicației pot fi văzute în imaginile de mai jos.

Icon

Description automatically generatedA picture containing text, clipart

Description automatically generatedText

Description automatically generatedGraphical user interface

Description automatically generated with low confidence

Figura 2.3.10: Butoane din cadrul aplicației desktop

Aplicația web de stilizare a textului a mai fost folosită, de altfel, și pentru crearea imaginilor puse ca titlu în partea vizuală a ferestrelor. Un exemplu de astfel de text poate fi văzut în Figura 2.3.11.

A blue sign with white text

Description automatically generated with low confidence

Figura 2.3.11: Titlu stilizat în aplicația web MakeText.io

## **Crearea interfeței grafice a aplicației desktop**

Aplicația desktop este creată folosind framework-ul UI Windows Presentation Foundation (WPF). Acesta este un subsistem grafic dezvoltat de Microsoft, similar cu WinForms, folosit pentru a crea interfețe grafice ale aplicațiilor bazate pe ferestre și a fost lansat în 2006 ca parte a .NET Framework 3.0.

Pentru dezvoltarea ferestrelor cu ajutorul WPF, am creat niște ferestre cu extensia .xaml, care au în spate câte un fișier de business logic cu extensia .cs. Ferestrele pot fi găsite în folderul de Views, pe care îl voi afișa mai jos (Figura 2.4.1.).

Text

Description automatically generated with medium confidence Text

Description automatically generated with low confidence

Figura 2.4.1: Ferestrele aplicației desktop.

După cum se poate observa, s-a creat un număr relativ mare de ferestre, datorat atât numărului de 3 tipuri de utilizatori, cât și a numărului mare de funcționalități ale acestora și a părții de statistici a aplicației.

## **Crearea unui sistem de register și login**

În acest capitol, voi prezenta funcționalitatea de autentificare a aplicației desktop din cadrul acestei lucrări și voi defini câteva dintre noțiunile de care am avut nevoie pentru implementarea acesteia. Aplicația dispune, de asemenea, și de o funcționalitate pentru cazul în care utilizatorul a uitat parola contului său.

### Înregistrarea în aplicație

Imediat dupa alegerea tipului de utilizator din fereastra de start a aplicației, utilizatorul ale de ales dintre a se autentifica în aplicație sau de a crea un cont nou, dacă acesta nu a mai utilizat aplicația până la momentul respectiv.

Pentru a crea un cont de orice tip, utilizatorul are nevoie de o adresă de e-mail validă și de o parolă care să fie de minim 6 caractere. De asemenea, este necesară confirmarea parolei prin repetarea ei, pentru a ne asigura că nu există greșeli în introducerea acesteia.

Atât în partea de autentificare cât și în cea de înregistrare, parola este ascunsă în momentul scrierii, evitând astfel cazurile în care parola este văzută de o persoană din apropierea calculatorului (Figura 2.5.1).

Graphical user interface

Description automatically generated

Figura 2.5.1: Ascunderea parolei la introducere

De altfel, în cazul introducerii unei parole invalide sau în cazul în care prima parolă introdusă nu coincide cu cea confirmată, utilizatorul va fi notificat printr-un mesaj pop-up, așa cum poate fi observat în imaginea de mai jos. (Figura 2.5.2.) Aceeași metodă de informare este aplicată și în cazul în care se găsește o problemă la validarea adresei de e-mail.

Graphical user interface, text, application, chat or text message

Description automatically generatedGraphical user interface, text, application, chat or text message

Description automatically generated

Figura 2.5.2: Validarea parolei introduse

Validarea adresei de e-mail se face în trei pași:

* verificarea printr-un regex dacă inputul prezintă forma unei adrese valide([utilizator]@[domeniu]),
* verificarea existenței unui cont asociat adresei de email introduse,
* confirmarea adresei de email prin introducerea unui cod unic trimis pe aceasta la momentul în care se dorește crearea contului. (Figura 2.5.3)

Codul unic este generat automat, în mod aleatoriu, prin inițializarea unei instanțe a clasei Random, și este format din 6 caractere alfanumerice.

Graphical user interface, diagram, application

Description automatically generated

Figura 2.5.3: Solicitarea codului unic trimis prin e-mail

### Simple Mail Transfer Protocol

Pentru trimiterea oricărui tip de e-mail automat de către aplicație, am folosit un protocol de transfer care poate fi utilizat atât pentru a trimite, cât și pentru a primi mesaje prin poșta electronică. Acest protocol poartă numele de Simple Mail Transfer Protocol (SMTP). În cadrul aplicației mele, serverul de SMTP este folosit doar pentru a trimite e-mailuri de la o adresă la alta.

### POP/IMAP

POP și IMAP reprezintă metode de accesare e e-mailurilor. Prin metoda POP, un e-mail poate fi descărcat și citit doar pe un singur calculator, întrucât acesta este șters din serviciul de e-mail dupa descărcarea sa. Astfel, dacă se încearcă accesarea contului de pe un alt dispozitiv, mesajele care au fost descărcate deja nu vor mai fi disponibile.

Pe de altă parte, IMAP permite vizionarea e-mailurilor de pe mai multe dispozitive, prin accesarea acestora direct din serviciul de e-mail.

### Cum funcționează SMTP?

Protocolul SMTP trimite din codul aplicației un mesaj către Serverul SMTP. Mesajul este trimis mai apoi în cloud, de unde este preluat de către Serverul POP/IMAP. În final, e-mail-ul ajunge în inboxul utilizatorului, de unde poate fi vizionat prin metoda de accesare aleasă de aplicația de mesagerie folosită. Modul de funcționare al SMTP poate fi observat și în figura din pagina următoare. (Figura 2.5.4.).

Comunicarea între serverele de e-mail se face, de obicei, prin portul standard TCP 25. Acesta nu este, însă, cel utilizat de aplicația din lucrarea de față, întrucât e-mailurile trimise folosind acest port sunt văzute ca spam. În consecință, portul folosit în aplicație pentru acest lucru este 587.

Diagram

Description automatically generated

Figura 2.5.4: Trimiterea de e-mail-uri folosind SMTP

Codul metodei care trimite e-mailul către adresa utilizatorului poate fi observat în imaginea de mai jos (Figura 2.5.5.):

Text

Description automatically generated

Figura 2.5.5: Trimiterea de e-mail-uri folosind SMTP, transpusă în codul sursă

### Trimiterea de e-mail-uri prin intermediul aplicațiilor terțe

După cum se poate observa în codul anterior, ca adresă care să trimită e-mailurile am folosit adresa mea personală, utilizând serviciul de e-mail Yahoo. Acest lucru se datorează faptului că, deși creasem inițial o nouă adresă de e-mail doar pentru trimiterea acestor mesaje, Gmail a încetat să mai permită accesul aplicațiilor terțe în adresele de mail în data de 30 mai 2022.

Pentru a trimite mesaje prin e-mail din exterior, adresa expeditorului trebuie configurată din setările serviciului de mail folosit. Serviciul generează o parolă unică în mod aleatoriu prin care aplicația se poate conecta din codul sursă.

### Criptarea și decriptarea prin Advanced Encryption Standard (AES)

Nu în ultimul rând, în cadrul acestui subcapitol voi prezenta metoda de criptare și decriptare folosită pentru a asigura o oarecare protejare a parolelor introduse în baza de date.

Criptarea AES reprezintă o specificație a criptării datelor stabilită de U.S. National Institute of Standards and Technology (NIST). Deși este o metodă destul de greu de implementat, aceasta este una dintre cele mai sigure, fiind cu mult peste criptarea DES și Triple DES.

AES este valabil în diferite pachete de criptare și este primul și singurul cifru accesibil care este aprobat de către U.S. National Security Agency (NSA) pentru informații top secrete.

Atât criptarea cât și decriptarea datelor se face din cod, chiar înainte ca datele să fie introduse în baza de date sau folosite în aplicație. Astfel, în momentele în care nu este nevoie de ele, datele sunt protejate și stocate la loc sigur.

Legarea bazei de date de aplicație și stocarea datelor, criptate sau nu, o voi explica în următorul capitol. (2.6. Popularea bazei de date).

### Autentificarea în aplicație

După crearea unui cont de orice tip, utilizatorul se va putea autentifica în aplicație folosind opțiunea de Login. Pentru a se conecta, însă, contul trebuie să fie de același tip cu cel pentru care a fost creat. Această restricție se va elimina în momentul în care aplicația va ajunge să fie folosită în centrele de donare, deoarece pe fiecare calculator va fi instalată o singură parte a aplicației, folosind un flag. Astfel, se elimină cazurile în care un simplu donator s-ar putea înregistra cu un cont de personal de recoltare sau de medic pentru a face anumite operații la care nu are acces.

Fereastra de autentificare a aplicației desktop arată în felul următor (Figura 2.5.6):

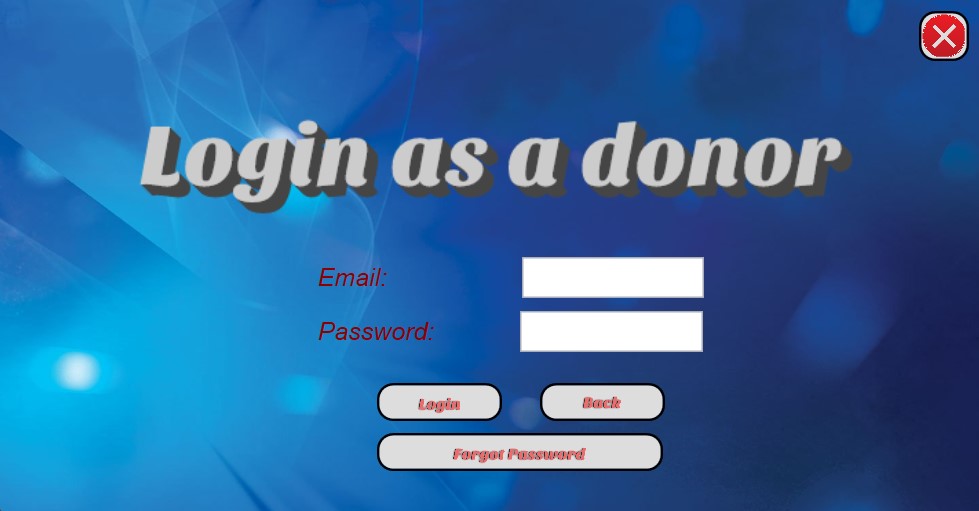


Figura 2.5.6: Fereastra de login a aplicației desktop, pentru donatori

Validările și formatările textului introdus prezente în sistemul de autentificare sunt aceleași cu cele din sistemul de înregistrare.

Dacă în subcapitolul trecut era vorba despre o criptare a parolelor înainte de trimiterea lor în baza de date, în acest pas parolele respective sunt decriptate și comparate cu inputul din câmpul Password. Dacă cele două parole coincid, atunci utilizatorul este autentificat și trimis în meniul specific tipului său de utilizator.

### Funcția de uitare a parolei

Un caz aparte îl reprezintă situația în care utilizatorul și-a uitat parola folosită la crearea contului. Pentru rezolvarea acestei probleme, am implementat o funcționalitate a sistemului de autentificare care ajută utilizatorii să își modifice parola, dând click pe butonul “Forgot Password” prezent în Figura 2.5.6. de mai sus.

În momentul în care utilizatorul dă click pe acest buton, i se cere adresa de e-mail a contului, se verifică dacă această adresă aparține cu adevărat unui cont deja existent și se trimite un e-mail cu structura și tehnologiile deja prezentate anterior în subcapitolul despre înregistrarea unui cont.

Un exemplu de e-mail trimis cu acest scop poate fi observat în imaginea de mai jos (Figura 2.5.7.):

Graphical user interface, text, application, email

Description automatically generated

Figura 2.5.7: E-mail trimis din aplicație pentru schimbarea parolei unui cont.

Dacă utilizatorul introduce corect codul primit prin e-mail, este trimis către o fereastră în care i se cere setarea unei noi parole a contului și confirmarea acesteia pentru a asigura corectitudinea ei. Are loc din nou o verificare a numărului de caractere ale parolei și, dacă aceasta îndeplinește toate criteriile, este setată ca parolă nouă.

## **Popularea bazei de date**

Odată ce am implementat sistemul de login al aplicației, am decis că un următor pas important ar fi popularea bazei de date, pentru a o pregăti de testarea celorlalte funcționalități care aveau a fi implementate.

Adăugarea datelor în sistem s-a făcut în două modalități:

* Din interfața grafică a programului SQL Server Management Studio,
* Direct din aplicație, mai ales a conturilor, folosind proceduri stocate și iterări ale entității Cont.

### Adăugare folosind SSMS

SSMS permite adăugarea de date în tabelele create direct din interfața grafică. Pentru a face acest lucru, se poate da click dreapta pe numele tabelului din Object Explorer și click pe Edit Top 200 Rows. Se va deshchide o fereastră similară celei din Figura 2.6.1. în care se vor putea adăuga, șterge sau modifica datele deja existente în tabelul respectiv.

Graphical user interface, application

Description automatically generated

Figura 2.6.1: Modificarea câmpurilor unui tabel folosind SSMS

### Proceduri Stocate

Procedurile stocate pot fi scrise în cod SQL și reținute în baza de date folosind SSMS. Acestea reprezintă metode care pot fi salvate, modificate și reutilizate de fiecare dată când avem nevoie. Este posibilă, de asemenea, trimiterea de parametri, astfel încât să se execute procedura în funcție de valorile trimise.

Pentru a crea o procedură stocată, am dat click dreapta pe folderul Stored Procedures, hover pe New și click pe opțiunea Stored Procedure. La urmarea acestor pași, se deschide o fereastră similară cu cea din Figura 2.6.2. în care se poate scrie codul SQL pentru crearea procedurii.

Graphical user interface, text, application, email

Description automatically generated

Figura 2.6.2: Crearea unei proceduri stocate în SSMS

Fereastra include, inițial, anumite informații care nu sunt necesare procedurii, dar se poate observa comanda SQL de creare a acesteia și faptul ca pot fi utilizați mai mulți parametri atunci când o definim.

### Legarea aplicației de baza de date prin ADO.NET

Pentru a putea utiliza și modifica baza de date din codul aplicației, este necesară implementarea acesteia drept Model item în codul sursă. Din fericire, există o tehnologie în .NET Framework care se ocupă cu acest lucru., tehnologie pe care o voi prezenta mai departe.

ADO.NET este o tehnologie de accesare a datelor dezvoltată de Microsoft și lansată în cadrul .NET Framework. Datorită ADO.NET, am putut utiliza o serie de componente software deja existente pentru a accesa tabelele și procedurile stocate din baza de date a aplicației.

Pentru a face acest lucru, am creat în proiect un nou item de tip ADO.NET Entity Data Model, cu extensia .edmx, am făcut câteva configurări și am bifat componentele de care aveam nevoie. La final, s-a generat un Connection String în fișierul App.config, care face de fapt legătura între aplicație și modelul respectiv.

Astfel, după acest pas, baza de date a fost pregătită de utilizare.

### Manipularea bazei de date direct din codul sursă

Odată ce legătura cu baza de date a fost făcută, am putut utiliza procedurile stocate ca simple metode în codul sursă, iar tabelele au putut fi iterate pentru a afla, adăuga, modifica sau chiar șterge anumite date prezente.

În figura 2.6.3. de mai jos, se pot observa cele două modalități de manipulare a bazei de date prin cod C#.

Text

Description automatically generated

Figura 2.6.3: Aprobarea unui chestionar medical printr-o procedură stocată și adăugarea unui rând nou în tabelul Donare direct din codul sursă al aplicației, urmate de o salvare a modificărilor.

## **Implementarea funcționalităților**

Așa cum am descris și în incipitul acestei lucrări, scopul de bază este de a ușura munca celor implicați în donarea de sânge. Acest lucru duce, implicit, la necesitatea unui design intuitiv, pe care l-am descris in subcapitolele trecute, dar și a unor funcționalități utile, care să ușureze cu adevărat procesul de donare.

Voi prezenta în acest capitol, pe scurt, fiecare funcționalitate și felul în care a fost implementată.

### Data Binding

Având în vedere că, printre multe altele prezentate anterior, noțiunea de Data Binding a jucat un rol important în implementariea tuturor funcționalităților, o voi menționa aici. Prin această tehnică, am putut lega două surse de date și le-am putut sincroniza. Astfel, în cazurile în care a fost nevoie de afișarea în interfața grafică a unor lucruri stocate în baza de date, Data Binding a reprezentat soluția generală.

### Funcționalități specifice medicilor

Funcționalitatea de bază la care au acces medicii în cadrul acestei aplicații este de a putea trimite cereri de componente de sânge oricând sunt necesare în spitalul din care face parte. Astfel, dacă dintr-un incident nefericit un pacient are nevoie de o transfuzie, medicul respectiv poate completa cererea și o poate adăuga în baza de date.

Pentru a face legătura între pacient si cererea respectivă, este nevoie ca pacientul să se afle în baza de date. De aceea, medicul poate, de asemenea, să înroleze pacienții în acest program. Dacă din anumite motive pacientul nu dorește acest lucru, medicul poate trimite cererea de componente sanguine fără să fie făcută pentru un anumit pacient.

O altă funcționalitate accesibilă medicilor este aceea de a vedea o fereastră cu numărul de componente de sânge necesare, după tip: plasmă, globule roșii sau trombocite.

### Funcționalități specifice donatorilor

Donatorii au acces, în cadrul acestei aplicații, la completarea unor chestionare medicale în vederea informării personalului de recoltare că aceștia sunt dispuși să doneze sânge. Printre informațiile oferite în acest chestionar se numără Codul Numeric Personal al donatorului, numele complet, domiciliul și reședința, numărul de telefon, greutatea și câteva informații opționale care vor putea fi aflate și modificate în momentul în care donatorul se prezintă la centrul de recoltare. De altfel, donatorul poate opta să doneze pentru o anumită persoană, dar în acest caz se va face o verificare dacă cele două grupe sanguine coincid.

O a doua funcționalitate a contului de donator este aceea de a vedea toate donările care au fost efectuate folosind acea adresă de e-mail. În lista donărilor este prezent și un câmp care afișează dacă donarea a fost efectuată sau nu, câmp inițializat cu valoarea “False”, dar care poate fi schimbat în “True” în momentul donării. Lista mai conține, de altfel, data la care a fost programat donatorul și CNP-ul său.

O ultimă funcționalitate specifică donatorilor este aceea de a vedea informații legate de fiecare donator în parte, care a folosit adresa de e-mail conectată. Pentru a vedea acest lucru, însă, este nevoie de Codul Numeric Personal al persoanei ale căror statistici se doresc afișate. Acest lucru asigură faptul că fiecare donator poate vedea doar informațiile sale, neavând acces la celelalte.

Cu alte cuvinte, această secțiune a aplicației poate fi văzută ca un profil personal al donatorilor, care poate fi accesat prin 2 Factor Authentication. Primul factor este reprezentat de parola contului iar al doilea este chiar CNP-ul donatorului.

Printre informațiile oferite în această secțiune se numără:

* Numele donatorului,
* Numărul donărilor completate,
* Recompensele luate după donarea de sânge,
* Contravaloarea totală a recompenselor, în RON.

### Funcționalități specifice personalului de la centrul de recoltare

Personalul centrelor de recoltare are, ca tip de utilizator, cele mai multe funcționalități. Cea mai importantă, însă, este cea de aprobare sau respingere a chestionarelor medicale trimise de către potențialii donatori. Pentru a putea face acest lucru, am pus în fereastra respectivă o listă cu toate cererile aflate în așteptare. Pentru a accepta una dintre cereri, personalul de recoltare trebuie să dea click pe rândul unuia dintre donatori, să introducă data la care se poate prezenta la recoltare și să dea click pe butonul “Approve”. În cazul în care unul dintre donatori nu atinge criteriile necesare pentru donare, acesta poate fi respins prin click pe rândul asociat lui și click pe butonul “Decline”. Odată ce se apasă pe butonul de aprobare, se trimite un e-mail către adresa donatorului și se creează un nou câmp în cadrul tabelului “Donare” din baza de date.

De altfel, în cadrul acestei ferestre, personalul de recoltare poate modifica anumite câmpuri ale chestionarului în urma analizelor făcute. Fereastra specifică acestei funcționalități se poate vedea mai jos, în Figura 2.7.1.

Table

Description automatically generated with low confidence

Figura 2.7.1: Funcționalitatea de programare a donatorilor

Se observă în imaginea de mai sus faptul că, neștiind ce grupă de sânge are în momentul completării formularului, unul dintre donatori a lăsat acest câmp liber, câmp care poate fi schimbat inaintea programării acestuia.

O a doua funcționalitate importantă pentru acest tip de utilizator este cea de a seta o donare ca fiind completată. Acest proces are loc după ce donatorului i-a fost recoltată o anumită cantitate de sânge, care va fi introdusă într-un TextBox. Tot în acest pas se alege și recompensa donatorului, prin selectarea unuia dintr-o listă (Figura 2.7.2.).

Graphical user interface, table

Description automatically generated

Figura 2.7.2: Completarea donării și alegerea recompensei dorite de donator

Ultimele două funcționalități ale personalului de recoltare sunt reprezentate de adăugarea de spitale în baza de date și adăugarea și gestionarea recompenselor stocate.

Pentru adăugarea unui spital, personalul de recoltare trebuie să introducă numele și județul acestuia. De asemenea, pentru a putea urmări daca acest lucru a fost făcut deja de altcineva, din cadrul aceluiași centru de recoltare sau din cadrul altuia, utilizatorului i se pune la dispoziție o listă cu toate spitalele deja aflate în acest program. După completarea câmpurilor, se face click pe butonul de adăugare.

Gestionarea recompenselor se face în 2 pași, primul reprezentând adăugarea efectivă a unui tip de recompense prin introducerea denumirii, numelui companiei care oferă recompensa, contravaloarea unei singure instanțe și numărul de instanțe ale acestei recompense. Al doilea pas este reprezentat de cazul în care se dorește o creștere a numărului de recompense ale unui singur beneficiu, caz în care se alege o anumită recompensă dintr-o listă și se introduce numărul de instanțe adăugate.

### Opțiunea de urmărire a statisticilor

Indiferent de persoana care folosește aplicația, utilizatorul va avea acces la urmărirea câtorva tipuri de statistici prin apăsarea butonului “Statistics” din meniul de start al aplicației desktop. În urma acestui click, se va deschide o fereastră care va prezenta trei opțiuni de statistici:

* Benefits budget
* Remaining benefits
* Volume donated

Prin implementarea acestor părți ale aplicației, am oferit șansa donatorilor și tuturor celorlalți utilizatori de a vedea ce recompense pot primi încă dinaintea efectuării recoltării efective de sânge, de a vedea în ce măsură se implică unele companii în acest proces și de a vedea cât de mult sânge s-a donat din momentul lansării acestei aplicații.

Un exemplu de fereastră a statisticilor poate fi văzută mai jos, în Figura 2.7.3.

A picture containing graphical user interface

Description automatically generated

Figura 2.7.3: Fereastră care prezintă numărul de recompense rămase și tipul lor.

Se observă în imaginea de mai sus și faptul că, prin mouse hover pe o parte a grafului, este prezentat procentul pe care îl are fiecare tip recompensă în baza de date.

## **Crearea site-ului web cu informații pentru utilizator**

În locul unei simple ferestre de ajutor în cadrul aplicației desktop, am decis să creez un site web care să ofere utilizatorului diverse informații legate de procesul de donare de sânge, cerințele care trebuie să fie îndeplinite înaintea donării, anunțuri și informații de contact, un ghid de utilizare al aplicației și un chatbot care să răspundă la diverse întrebări.

Toate aceste funcționalități au fost implementate cu ajutorul HTML, CSS, JavaScript și a platformei IBM Cloud. Voi descrie aceste tehnologii în capitolul 3. Descrierea tehnologiilor folosite.

### Design-ul de bază a site-ului web

La fel ca și până în acest punct, am încercat să păstrez un design curat, care să trimită, prin culori, contraste și background-uri, la sistemul medical și la importanța pe care o poate avea această aplicație în viețile oamenilor. Astfel, logo-ul aplicației a fost inserat în partea de sus a paginilor, pe un fundal de culoare închisă. De asemenea, în toate paginile se găsește un meniu de navigare care rămâne în partea de sus a paginii în cazul în care se face scroll iar imaginile și obiectele care conțin text au aceleași valori ale unor proprietăți în tot site-ul.

O altă funcționalitate legată de design, prezentă în toate paginile, este cea de mod nocturn. Activarea acesteia face culorile obiectelor să fie mai inchise și setează culoarea scrisului ca fiind albă. Toate aceste componente ale site-ului pot fi văzute în Figura 2.8.1:

Graphical user interface, website

Description automatically generatedGraphical user interface, website

Description automatically generated

Figura 2.8.1: Site-ul aplicației Life Sponsors, în Light Mode și Dark Mode

Iconița site-ului este aceeași cu cea a aplicației desktop iar titlurile paginilor sunt sugestive pentru funcționalitățile oferite de către acestea.

Odată ce am creat “scheletul” fiecărei pagini în parte, am putut începe să lucrez, separat, la fiecare componentă a site-ului. Multe dintre stilurile acestora sunt implementate de mine în același fișier CSS, în timp ce altele sunt luate din Bootstrap.

### Implementarea paginii de start

Chiar și în ziua de azi, există multe persoane cărora le e frică să doneze sânge din frica de a nu li se face rău sau de a se întâmpla ceva în timpul donării. Tocmai de aceea, am ales ca pe pagina de start a site-ului să pun câteva imagini cu atmosfera din centrele de donare, pentru a arăta donatorilor că nu au de ce să se teme. Personalul de recoltare din fiecare centru este mereu calificat și ar trebui să dea dovadă de profesionalism, atât prin acțiunile lor, cât și prin metodele folosite.

Astfel, dacă donatorul îndeplinește toate condițiile necesare donării, ceea ce este de verificat înainte de fiecare donare, nu ar trebui să se întâmple nimic care să pună în pericol viața donatorului. Imaginile sunt inserate imediat sub meniu.

O a doua componentă cheie a acestei pagini este reprezentată de un Google Calendar, inserat ca obiect în partea dreaptă a paginii. Acesta oferă tuturor utilizatorilor să își urmărească mai ușor programul și să vadă sărbătorile ortodoxe importante în care poate nu s-ar putea programa sau efectua donări de sânge. Pe această pagină am adăugat, de asemenea, o secțiune cu informații generale despre procesul de donare de sânge și despre cerințele necesare donării. Pagina de start a site-ului poate fi văzută în Figura 2.8.2.

Graphical user interface, website

Description automatically generated

Figura 2.8.2: Pagina de start a site-ului web.

### Implementarea paginii de contact

Pe a doua pagină a site-ului web se găsesc informații de contact al centrului de donare, împreună cu un link către o pagină de Facebook. Pentru demonstrarea funcționalității pentru această lucrare, am lăsat un link care duce spre profilul personal de Facebook, dar acest link poate fi schimbat în orice moment, în funcție de preferințele centrelor de donare.

Informațiile sunt puse pe un obiect care se rotește la mouse hover, sub forma unui card. Astfel, pe partea de spate a cardului am putut pune link-ul către pagina de Facebook iar în față informații ca adresa centrului de donare, adresă de mail și un număr de telefon de contact. Cele două părți ale cardului se pot vedea în Figura 2.8.3.

Graphical user interface, text, application

Description automatically generatedGraphical user interface, application, Teams

Description automatically generated

Figura 2.8.3: Cele două părți ale cardului de pe pagina de contact.

### Implementarea paginii de anunțuri

Printre funcționalitățile importante ale oricărei platforme se numără și posibilitatea de a posta anumite anunțuri pentru utilizatorii ei. De aceea am decis să implementez și o pagină pentru acest lucru, unde se pot afișa anunțuri oficiale, care pot fi legate strict de sistemul medical și de procesul de donare, sau nu.

Aici se pot pune link-uri și către anunțul care poate fi afișat, un titlu corespunzător, o imagine reprezentativă anunțului și o scurtă descriere a ceea ce se afla în spatele link-ului. Anunțurile sunt afișate în număr de câte 3, unul sub altul, așa cum se poate vedea în Figura 2.8.4.

Graphical user interface, application, website, Teams

Description automatically generated

Figura 2.8.4: Pagina de anunțuri a site-ului web

### Implementarea paginii ghidului de utilizare

Deși interfața grafică a aplicației este una destul de intuitivă, am luat în calcul și cazul în care unele persoane ar putea întâmpita dificultăți în utilizarea acesteia. De aceea, există o secțiune a site-ului dedicată unui ghid de utilizare, care explică pas cu pas ceea ce poate face fiecare utilizator si pașii care trebuie urmați pentru anumite acțiuni. Pagina respectivă se poate vedea în Figura 2.8.5. de mai jos.

Graphical user interface, website

Description automatically generated

Figura 2.8.5: Pagina ghidului de utilizare al aplicației

De asemenea, tot pe această pagină utilizatorul va fi întâmpinat de chatbot-ul aplicației, un asistent virtual antrenat pentru a răspunde la eventualele întrebări venite din partea utilizatorului. Implementarea acestuia și integrarea lui în site vor fi explicate în capitolele 2.9 și 2.10.

### Implementarea paginii de informații legate de aplicație

Pe ultima pagină implementată apar, pe scurt, informații legate de aplicația desktop și tehnologiile folosite pentru crearea acesteia. Pentru a ține un design cât mai curat pe celelalte pagini și ferestre din cadrul acestei lucrări, aici este locul în care se vor fi puse drepturile de autor și alte lucruri legate de asta.

Pagina acestor informații poate fi văzută mai jos, în Figura 2.8.6.

Graphical user interface, website

Description automatically generated

Figura 2.8.6: Pagina informațiilor despre aplicație.

## **Crearea asistentului virtual**

În multe dintre platformele online existente în orice domeniu, se află, de obicei, o secțiune de Q&A sau de FAQ. De cele mai multe ori, acestea ocupă un spațiu imens în pagină iar pentru a găsi răspunsul la o întrebare, utilizatorul trebuie să caute prin multe alte întrebări.

Cu toții am auzit cel puțin o dată de asistenții virtuali dezvoltați în prezent: Siri, Alexa sau Google Assistant. Aceștia au început să fie integrați în diferite dispozitive încă din anul 2011, în frunte cu Siri, și au reprezentat un pas imens pentru modul în care inteligența artificală poate ajuta utilizatorii unui smartphone.

Pentru site-ul aplicației din această lucrare, am decis să implementez un asistent virtual folosind resursa Watson Assistant al platformei IBM Cloud, care să înlocuiască formula clasică de FAQ, îmbunătățind timpul în care utilizatorul găsește răspunsul la o întrebare. De altfel, acesta ocupă un spațiu mai mic în pagină și are un design modern, similar cu al unui chat de pe platformele de socializare.

Astfel, chatbot-ul aplicației Life Sponsors este ușor de utilizat de către oricine a mai folosit vreodată o aplicație de online chat. Câteva imagini ale acestuia pot fi văzute în Figura 2.9.1.

Graphical user interface, text, application, chat or text message

Description automatically generatedGraphical user interface, text

Description automatically generatedGraphical user interface, text, application

Description automatically generated

Figura 2.9.1: Imagini ale asistentului virtual din cadrul site-ului web

Voi prezenta, în continuare, pașii urmați pentru a crea acest chatbot și integrarea sa în site-ul web.

### Crearea contului de IBM Cloud și accesarea resurselor

Pentru a folosi orice resursă a platformei IBM Cloud, a trebuit să îmi creez si să îmi activez un cont, alegând și unul din planurile oferite. Acest lucru s-a făcut rapid, prin adresa de e-mail personală.

Odată ce a fost creat contul, am putut accesa resursele oferite de IBM prin planul Lite, de tip pay-as-you-go, care oferă acces la mai mult de 50 de resurse. Pentru a face asta, am mers în Dashboard, la secțiunea “Resources” -> “Services and Software” (Figura 2.9.2.).

Graphical user interface, application

Description automatically generated

Figura 2.9.2: Accesarea resurselor din IBM Cloud direct din Dashboard

După acest pas, am creat o nouă resursă de tip Watson Assistant-8e și am început configurarea asistentului virtual personal.

### Crearea și configurarea asistentului virtual

Pentru început, resursa de tip Watson Assistant trebuie pornită din meniul “Manage”. După acest pas inițial, se poate crea un nou asistent prin click pe butonul Create “assistant”. Se introduc câteva date și se dă click pe “Create assistant” (Figura 2.9.3.).

Graphical user interface, text, application, email

Description automatically generated

Figura 2.9.3: Crearea unui asistent virtual

După crearea asistentului, se poate da click pe acesta pentru a vedea câteva informații despre el.

Graphical user interface, text, application, email, Teams

Description automatically generated

Figura 2.9.4: Pagina de prezentare din IBM Cloud, a asistentului folosit în site.

În Figura 2.9.4. de mai sus, observăm că pentru antrenarea chatbot-ului pentru această lucrare, s-au folosit 23 intents, 5 entități și 24 noduri de dialog. Voi explica fiecare din aceste noțiuni mai jos și voi oferi exemple din asistentul creat de mine.

### Intents

Intent-urile reprezintă scopul unui anumit text trimis de către utilizator. Acestea pot fi văzute drept câte un set de întrebări sau de texte, sub aceeași categorie, cărora li se poate oferi un răspuns general. Intrnt-urile au în fața numelui simbolul ‘#’.

Pentru aplicația Life Sponsors, am încercat să antrenez asistentul cu mai multe versiuni ale aceleiași întrebări, puse sub același intent. Astfel, un intent din cadrul asistentului arată în felul următor (Figura 2.9.5.):

Graphical user interface, text, application, email

Description automatically generated

Figura 2.9.5: Exemple de mesaje reținute în intent-ul #cerinte

În momentul în care asistentul recunoaște, prin inteligența artificială, într-un mesaj faptul că un utilizator s-a referit la un anumit intent, acesta va fi prins și tratat printr-un nod de dialog, noțiune pe care o voi explica mai departe.

### Nodurile de dialog

Prin nodurile de dialog am setat răspunsurile efective ale chatbot-ului, în funcție de intent-urile detectate. Pentru a face acest lucru, am setat fiecăruia dintre cele 24 de noduri o condiție prin care, dacă se recunoaște un anumit intent în mesajul trimis de către utilizator, se va alege în mod aleatoriu unul dintre mesajele prestabilite în acel nod. Fereastra de setare a acestor condiții și mesaje poate fi văzută mai jos, în Figura2.9.6.

Graphical user interface, text, application, email

Description automatically generated

Figura 2.9.6: Fereastra de gestionare a unui nod de dialog creat prin Watson Assistant

Mai mult decât atât, IBM Cloud permite prin intermediul acestei resurse vizualizarea unui ‘tree’ al nodurilor create, pentru gestionarea mai ușoară a decurgerii conversației dintre utilizator și asistent. Astfel, se pot crea noduri părinte sau copil altor noduri și pot fi categorizate în foldere. Un astfel de ‘tree’ al nodurilor poate fi văzut în Figura 2.9.7.

Graphical user interface, application

Description automatically generated

Figura 2.9.7: Fereastra de prezentare a nodurilor de dialog a conversației

### Entități

Entitățiile dintr-un chatbot reprezintă modificatorii folosiți de către utilizatori pentru a descrie ceea ce au ei nevoie. Dacă prin intent-uri ne refeream la un set de întrebări sau de texte asemănătoare, prin entități ne referim la anumite obiecte sau instanțe din cadrul aceleiași categorii. Aceste entități sunt precedate de simbolul ‘@’.

De exemplu, orice text în format de tipul dd/mm/yyyy va fi recunoscut ca o entitate de tipul @sys-date.

Pe lângă aceasta, am mai adăugat încă 4 entități în cadrul acestui bot, și anume @sys-time, @sys-percentage, @sys-currency și @sys-number. Toate cele 5 entități folosite au fost implementate de către IBM, iar folosirea lor implică doar bifarea unei opțiuni în cadrul secțiunii de ‚’System Entities’ din meniu.

## Integrarea asistentului virtual în site-ul web.

Odată ce a fost creat, asistentului I s-a mai putut face o ultimă configurare prin crearea unei integrări de tip WebChat. Aici s-au putut schimba culorile, mesajele de întâmpinare a utilizatorului și opțiunile care să apară pe ecranul principal al chatbot-ului. Culorile setate în momentul finalizării asistentului virtual se pot vedea în Figura 2.10.1.

A screenshot of a computer

Description automatically generated with medium confidence

Figura 2.10.1: Culori utilizate în interfața asistentului virtual

După ce ultimele configurări au fost făcute, asistentul virtual a fost pregătit pentru testare și integrare. Pentru testare, am trimis un link prietenilor, care au jucat rolul de simpli donatori, care nu știau nimic despre aplicație. În acest fel, am putut îmbunătății calitatea experienței utilizatorilor înainte de integrarea în site a chatbot-ului.

Odată ce testarea si modificările au fost făcute cu succes, resursa Watson Assistant din IBM Cloud a generat un script care a putut fi inserat în documentul .html în care aveam nevoie de acest bot, mai exact în howto.html, în care apare ghidul de utilizare al aplicației.

Text

Description automatically generated

Figura 2.10.2: Scriptul generat, folosit la integrarea asistentului virtual în site-ul web

# **Descrierea tehnologiilor folosite**

Pe lângă tehnologiile deja prezentate anterior, au mai fost câteva care ar merita menționate și descrise. Printre acestea se numără IDE-uri, software-uri de design, de gestionare a planificării proiectului și limbajele de programare.

## **Microsoft Visual Studio 2019**

Pentru implementarea funcționalităților oferite de aplicația desktop Life Sponsors, am folosit Microsoft Visual Studio 2019, un IDE dezvoltat de Microsoft pentru crearea programelor software , a site-urilor, aplicațiilor și serviciilor web și a aplicațiilor mobile.

Ediția folosită a fost Community, gratuită pentru studenți, iar sistemul de operare folosit a fost Windows 11. Versiunea Community prezintă aceleași caracteristici cu cea Professional și a fost introdusă în 2014 ca ediție nouă a IDE-ului, orientată către dezvoltatorii individuali de software și echipe mai mici.

Printre caracteristicile folosite de mine se numără Editorul de Cod, Debugger-ul, Designere pentru WPF și pentru clase, Editorul de Proprietăți și câteva Explorere.

## **Microsoft Visual Studio Code**

Pe lângă Visual Studio, am folosit și Visual Studio Code, care este un editor de coduri sursă dezvoltat tot de Microsoft. Caracteristicile precum evidențierea sintaxei, refactorizarea codului și instalarea extensiilor fac această tehnologie să fie potrivită pentru crearea paginilor web, motiv pentru care l-am folosit la implementarea site-ului aplicației.

Simplitatea acestuia poate fi și motivul pentru care 70% dintre utilizatorii care au completat chestionarul dat de Stack Overflow în 2021 au spus că îl utilizează.

### **Draw.io și StarUML**

Pentru proiectarea diagramelor, am folosit două tool-uri gratuite: Draw.io și StarUML. Ambele sunt programe software ce permit crearea de diagrame de tip flowchart, wireframe, relaționale și UML.

Există, totuși, câteva diferențe între acestea, una dintre ele fiind faptul că Draw.io este cross-platform și open source, în timp ce StarUML este creat de către MKLab și licențiat sub o versiune modificată de GNU GPL. De asemenea, Draw.io poate fi utilizat direct într-un browser web, pe când StarUML trebuie descărcat și instalat în memoria internă a calculatorului.

## **GitHub**

Sistemele de versionare sunt folosite pentru gestionarea codului sursă a diferitelor programe sau documente. Folosind această tehnologie, atât codul sursă cât și schimbările acestuia sunt stocate în cloud iar în cazul unei defecțiuni a hardware-ului pe care se face dezvoltarea proiectului, codul nu are de suferit.

Ca sistem de versionare a codului am folosit GitHub, care a devenit cea mai mare gazdă a codurilor sursă în luna Noiembrie, 2021.

Pe lângă caracteristicile specifice oricărui sistem de versionare care folosește Git, GitHub are câteva funcționalități în plus, printre care se numără și posibilitatea de a genera diverse grafice. Unul dintre aceste grafice poate fi văzut în Figura 3.4.1. în care se observă numărul de schimbări aduse proiectului pe decursul a câteva luni.

Chart, bar chart

Description automatically generated

Figura 3.4.1: Grafic reprezentând numărul de commit-uri trimise pe parcursul dezvoltării aplicației

## **Trello**

Platformele de project management reprezintă o abordare profesională de gestionare a sarcinilor necesare dezvoltării oricărei aplicații. Deși Jira este cea mai folosită alegere pentru acest lucru, am decis să folosesc una mai simplă și mai ușor de utilizat, întrucât nu a fost nevoie de implicarea altor membri de echipă.

Trello este o aplicație web dezvoltată de Trello Enterprise, o sucursală a companiei Atlassian încă din 2017. În această aplicație se pot crea board-uri de în stilul celor Kanban cu diferite sarcini ce pot fi atribuite membrilor unei echipe.

Peentru această lucrare, am creat un board cu 6 liste: Backlog, In queue, Working on it, Paused, Done și Trash.

## **Adobe Photoshop 2021**

Adobe Photoshop e un editor grafic dezvoltat și publicat de Adobe Inc. Pe parcursul a 22 de ani de dezvoltare, acest program software a devenit un standard atât în industria de raster graphics editing, dar și în crearea de artă digitală.

Pentru a aduce atât aplicației desktop cât și site-ului web o interfață cu un design cât mai estetic, am editat unele imagini în software-ul menționat anterior.

## **Limbaje de programare folosite:**

Prin limbajele de programare ne referim la seturile de reguli prin care se convertește codul scris în cod mașină. Voi explica, pe scurt, procesul prin care se ajunge de la cod scris în limbaj de programare la un fișier executabil al aplicației.

După ce codul sursă este scris, verificat și pre-procesat, compilatorul îl transformă în cod de asamblare. Prin procesul de asamblare, se va obține codul obiect care, împreună cu bibliotecile de legături dinamice (Dynamic link libraries - DLL) va contribui la crearea fișierușui executabil.

### C#

Pentru aplicația desktop, am folosit limbajul de programare C#, lansat pentru prima dată în anul 2000. Acesta este, în general, folosit pentru a crea aplicații de tip desktop, mobile, jocuri, servicii cloud-based și site-uri web.

Am ales să folosesc acest limbaj de programare pentru crearea aplicației desktop deoarece ușurează munca depusă pentru realizarea interfeței grafice prin folosirea WPF, tehnologie descrisă anterior.

### HTML

HTML (Hyper Text Markup Language) este un limbaj bazat pe SGML (Standard Generalized Markup Language), un standard internațional aprobat în anul 1986, care permite crearea de documente pentru paginile web.

Prin HTML, am reușit să creez paginile site-ului web și am putut crea obiecte individuale pe care să le stilizez separat față de restul paginii.

### CSS

CSS (Cascading Style Sheets) este utilizat la îmbunătățirea felului în care se prezintă o pagină web. CSS premite stabilirea proprietăților pentru elemente HTML și pune la dispoziție mai mult de 100 de proprietăți, fiind un instrument avansat de creare a site-urilor profesionale.

Proprietățile CSS pot fi definite în 3 feluri: inline, intern și extern. Pentru această lucrare, am folosit toate 3 tipurile, stilizările externe fiind puse într-un fișier cu extensia .css.

### JavaScript

Pe lângă HTML și CSS, JavaScript reprezintă una dintre tehnologiile de bază ale Word Wide Web-ului. Începând cu 2022, JavaScript este prezent în 98% din site-urile web existente, pe partea de client. Astfel, JS rulează în general pe calculatorul vizitatorului, secvențele de cod fiind traduse și executate de către browsere și trimise acestuia odată cu fișierul HTML din care face parte.

Pentru această lucrare, am folosit atât scripturi externe prin crearea unui fișier cu extensia .js, cât și interne pentru integrarea asistentului virtual în site-ul web.

### BOOTSTRAP

Deși CSS face simplă implementarea și setarea anumitor proprietăți obiectelor HTML, nu toată lumea iubește să facă acest lucru. Pentru dezvoltatorii de genul, există framework-ul Bootstrap, care aduce câteva stiluri deja implementate și gata de utilizare.

Astfel, am decis ca pe lângă stilurile externe și in-line create, să utilizez și câteva elemente grafice din elementele oferite de Bootstrap. Acestea au fost folosite în special la pagina de anunțuri, pentru crearea unor butoane mai estetice și pentru afișarea acestora într-un chenar, împreună cu imaginea, titlul și descrierea anunțurilor.

## **IBM Cloud**

Pentru a înțelege cum funcționează IBM Cloud, este necesar să știm câteva noțiuni de bază legate de Cloud Computing. Prin folosirea acestor tehnologii, putem folosi servicii ca servere, spațiu de stocare, baze de date, software-uri și analitice direct prin intermediul internetului.

Printre beneficiile utilizării unor servicii din cloud se numără costul redus al mentenanței infrastructurilor IT prin plătirea doar a serviciilor folosite. Pe cealaltă parte a sistemului, companiile care oferă astfel de servicii pot beneficia de economii majore prin livrarea aceluiași serviciu mai mai multor clienți.

Companiile care doresc să treacă la tehnologiile oferite de Cloud pot alege una din cele 4 tipuri de Cloud Computing existente: Private Cloud, Public Cloud, Hybrid Cloud sau Multicloud.

Ca tipuri de servicii, însă, există cele de tip Infrastructure-as-a-Service (IaaS), Platforms-as-a-Service (PaaS) și Software-as-a-Service (SaaS). Pentru a înțelege mai bine cum funcționează aceste tipuri, voi insera mai jos cine se ocupă de mentenanța fiecărui serviciu în funcție de caz (Figura 3.8.1.).

Graphical user interface

Description automatically generated

Figura 3.8.1: Cele 3 tipuri de servicii cloud, comparativ cu tehnologiile tradiționale On-Site

IBM oferă toate aceste posibilități de trecere în cloud clienților săi prin intermediul setului de servicii IBM Cloud. Acesta a fost lansat în anul 2011 cross-platform, fiind valabil pentru sisteme de operare precum Linux, Microsoft Windows, iOS și Android. Platforma oferă posibilitatea de abonare la un plan Pay-As-You-Go pentru a asigura faptul că nu există taxare pentru servicii care nu sunt utilizate.

Pentru această lucrare, am folosit IBM Watson Assistant, un serviciu de tip SaaS, Pay-As-You-Go, prin care utilizatorii pot crea și încorpora în software-ul lor un asistent virtual ce utilizează inteligența artificială.

# **Dezvoltare ulterioară**

Deși atât pentru aplicația desktop cât și pentru site-ul web am reușit să implementez funcționalitățile necesare folosirii acestora în centrele de donare, există anumite lucruri care ar fi ajutat și mai mult procesul de donare și ar fi simplificat unele funcționalități.

Printre acestea, se numără:

* + Crearea unei versiuni mobile a acestei aplicații,
  + Migrarea bazei de date în cloud,
  + Crearea unui server SMTP propriu, folosind un SMTP Server Open Source,
  + Publicarea site-ului web pe internet folosind un provider existent.

Toate aceste idei de dezvoltare ar adăuga o notă de securitate și de accesibilitate aplicației, dar ar aduce, în același timp, niște costuri suplimentare.

# **Concluzii**

În încheiere, aș vrea să menționez faptul că, deși aplicația creată aduce câteva îmbunătățiri modului în care are loc donarea de sânge, aceasta poate doar să încurajeze cetățenii să se prezinte în număr mai mare la centrele de recoltare pe tot parcursul anului, nu doar atunci când se oferă anumite recompense în plus.

Problema prezentată la începutul acestei lucrări este una veche și, deși s-a încercat rezolvarea ei prin diverse metode, acestea au putut doar să o diminueze.

Singura opțiune rămasă, din punctul meu de vedere, este ca fiecare dintre noi să realizeze că prin această cale ni se oferă posibilitatea să salvăm până la 3 vieți de fiecare dată când donăm sânge și să purtăm mai des în minte spusele uneia dintre cele mai cunoscute victime ale Holocaustului:

“*No one has ever become poor by giving.*”

-Anne Frank

# **Bibliografie și webografie**

* “ [Windows Presentation Foundation Unleashed](https://books.google.com/books?id=TSDxCqUBt18C&q=Windows+Presentation+Foundation)”, Adam Nathan
* <https://en.wikipedia.org/wiki/>
* <https://support.microsoft.com/ro-ro>
* <https://www.oracle.com/ro/database/>
* <https://sourcemaking.com/design_patterns>
* <https://www.sqlshack.com/sql-server-network-configuration/>
* <https://www.edrawsoft.com/article/what-is-eer-diagram.html>
* <https://www.duocircle.com/content/smtp-email/smtp-server-example>
* <https://www.zdnet.com/article/what-is-cloud-computing-everything-you-need-to-know-about-the-cloud/>