Text

Description automatically generated

LUCRARE DE LICENȚĂ

Aplicație pentru Centrele de Donare de Sânge

**Absolvent: Blendea Alin Daniel**

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

Brașov, 2022

Cuprins:

* Introducere
  + problema donarii de sange
  + scopul aplicatiei
  + Scurta descriere a aplicatiei
  + comparativ cu cele existente
* Etapele proiectarii aplicatiei
* Descrierea celorlalte tehnologii folosite
  + Visual Studio
  + Android Studio
  + SSMS
  + Firebase
  + ADO.NET
  + Github
  + Trello
  + Chart design softwares
  + Adobe Photoshop
* prezentarea aplicatiei
  + diagrame
  + design pattern
  + files
  + instructiuni de utilizare
  + ….
* concluzii
* directii de dezvoltare ulterioara
* bibliografie

pot fi puse anexe care sa contina grafice, poze, cod sursa, exemplificari

# **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: 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), 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 generated

Figura 1.2: Capturi de ecran ale ferestrei de start a aplicației și a paginii de pornire 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: 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 2.2).

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

Description automatically generated

Figura 2.2: 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: 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 2.2: 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 2.3: 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 2.4.),
* Introducerea numelui urmată de Click pe butonul Ok.

Graphical user interface, application

Description automatically generated

Figura 2.4: 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 2.5.1),
* Introducerea coloanelor și a tipului de date reținut în acestea (Figura 2.5.2),
* Salvarea tabelului prin Ctrl + S, urmat de introducerea numelui acestuia.

Graphical user interface, application

Description automatically generated

Figura 2.5.1: Crearea unui tabel folosind interfața grafică a SSMS

Graphical user interface, application, table

Description automatically generated

Figura 2.5.2: 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 2.7: 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 2.8: Diagrama cazurilor de utilizare ale aplicației desktop

Diagram

Description automatically generated

Figura 2.9: 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 2.10: 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 2.11: 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 2.11. 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 2.12.):

A screenshot of a computer

Description automatically generated with low confidence

Figura 2.12: 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 2.13: O parte din clasa RelayCommand.

Așa cum se poate observa în Figura 2.13, 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 2.14: Comandă membră al ViewModel-ului de cont de donator, prezentând apelarea metodei de Login din cadrul Model-ului respectiv.

Text

Description automatically generated

Figura 2.15: 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 2.16.). 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 2.16: 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 2.17.

Text

Description automatically generated with medium confidence

Figura 2.17: 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 2.18 și 2.19).



Figura 2.18: Imaginea de fundal a ferestrelor principale

Background pattern

Description automatically generated

Figura 2.19: 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(Figura 2.20.).

Icon

Description automatically generatedA picture containing text, clipart

Description automatically generatedText

Description automatically generatedGraphical user interface

Description automatically generated with low confidence

Figura 2.20: 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 2.21.

Logo

Description automatically generated

Figura 2.21: Titlu stilizat în aplicația web MakeText.io

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

## **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 capitol 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).

## **Popularea bazei de date**

## **Implementarea funcționalităților**

[intro]

### Funcționalități specifice medicilor

### Funcționalități specifice donatorilor

### Funcționalități specifice personalului de la centrul de recoltare

* Crearea unui site de ajutor cu informații legate de modul de utilizare al aplicației.
* Crearea unui chatbot care să răspundă la diverse întrebări legate de procesul de donare de sânge, folosind resursele oferite de IBM Cloud,
* Integrarea chatbot-ului în site-ul web.

# Descrierea tehnologiilor folosite

Tehnologii folosite:

* IBM Cloud
* Microsoft Visual Studio 2019
* Microsoft SQL Server Management Studio 2018
* ADO.NET Entity Data Models
* Github
* Trello
* Lucidchart
* Draw.io
* StarUML
* Adobe Photoshop 2021
* Limbaj de programare folosit: C#

### 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.

In momentul in care, pentru un pacient, este nevoie de sange, medicul va completa o cerere prin care va specifica pentru cine este nevoie de sange, ce grupa, gradul de urgenta (ridicat, mediu, scazut) si in ce locatie. Cererea va fi preluata de personalul centrului care in cazul in care nu are suficient sange va anunta cel mai apropiat donator (geografic) si care are grupa necesara si de asemenea este eligibil pentru donare (a trecut timpul limita de la ultima donare). Cererile vor fi onorate pe baza gradului de urgenta, iar in cazul in care persoana pentru care se cere sange este un donator activ, atunci acesta va avea prioritate.

In momentul in care un donator se prezinta voluntar pentru donare, acesta poate dona pentru o anumita persoana, dar este important sa specifice numele persoanei pentru care doneaza. Medicul pacientului pentru care se doneaza poate sa vada in orice moment daca a donat un numar suficient de persoane pentru pacientul sau.

DEMO

Testarea si demonstrarea unora dintre functionalitatile aplicatiei se pot observa in printscreen-urile de mai jos.

Graphical user interface, website

Description automatically generated

A picture containing text, monitor, electronics

Description automatically generated

*DONOR FUNCTIONALITY:*

Graphical user interface

Description automatically generated

Graphical user interface, application

Description automatically generated

*MEDIC FUNCTIONALITY:*

*A picture containing text, electronics

Description automatically generated*

A picture containing text, electronics, blue

Description automatically generated

*STAFF MEMBER FUNCTIONALITY:*

*Graphical user interface

Description automatically generated*

*A screenshot of a computer

Description automatically generated with medium confidence*

Dezvoltare ulterioara

-make app’s own smtp server

-android app

-database migration to postgres or firebase

Links:

*Trello:* [*https://trello.com/b/x2uj1Nca/blood-donor-app-project-management*](https://trello.com/b/x2uj1Nca/blood-donor-app-project-management)

*GitHub Repo:* [*https://github.com/alinblendea/RNABloodDonorApp*](https://github.com/alinblendea/RNABloodDonorApp)

*Fig 1.1.* [*https://news.sky.com/video/ukraine-huge-queues-at-blood-bank-after-russian-invasion-12550539*](https://news.sky.com/video/ukraine-huge-queues-at-blood-bank-after-russian-invasion-12550539)