Universitatea Politehnica din Timişoara
Facultatea de Automatică şi Calculatoare
Calculatoare şi Tehnologia Informaţiei

Bee Charity: Caritate și umanitate

Proiect de diplomă

Candidat:

Boiţiu Ionuţ – Cătălin

Conducător ştiinţific:

Conf. dr. Ing. Marinescu Cristina

Timişoara

2019

Cuprins

1 Introducere

1.1 Contextul aplicaţiei

Lumea în care trăim a fost din vechi timpuri plină de lipsuri şi de oameni ce au nevoie de ajutor.

În prezent, există o mulţime de cazuri televizate cu familii în situaţii dificile, care nu au nici ce le pune pe masă copiilor, nu au mereu cu ce să îi îmbrace pentru a-i trimite la şcoală sau bani pentru a le oferi minimul condiţiilor pentru o educaţie decentă. O mulţime de cazuri în care oameni în vârstă ajung în centre pentru bătrâni, centre care adesea ar avea nevoie de un mic ajutor pentru a le putea oferi celor în vârstă un moment de bucurie.

Revenind la primul exemplu, se poate observa că în urma acestor lipsuri, din cauza condiţiilor materiale sau financiare precare, numeroase familii ajung să îşi abandoneze copiii la diferite centre pentru creştere a copilului(centre de plasament) sau chiar în spitale(imediat după naştere), aceştia din urmă ajungând tot în custodia caselor de copii.

Mai exact, un studiu făcut în 2017[1], prezintă unele date statistice care ne spun că la fiecare nouă ore, un bebeluș e abandonat în maternitate sau pediatrie. Astfel, un număr de 1200 de copii sunt abandonați în fiecare an la vârste extrem de fragede. Anual, în programul de protecție specială intră 4000 de copii. Se aproximează că mai există încă 7500 de copii prezenți în cele 159 de centre de plasament, existente la nivelul țării.

Aparte celor două cazuri majore, mai poate fi adus în discuţie un exemplu precum categoria oamenilor aflaţi într-o situaţie defavorizată, grea. Chiar şi aceştia ar trebui să poată beneficia de ajutor, dacă e posibil.

Astfel, aplicaţia doreşte sprijinirea atât a centrelor care se angajează în asumarea responsabilităţii creşterii copiilor, a acelor familii care întâmpină dificultăţi în creşterea copiilor, a oamenilor în situaţii speciale sau chiar centrele de bătrâni.

Pe scurt, un ajutor înspre ONG-urile deja existente sau a acelor pedepsiți de viață.

1.2 Motivaţia

Motivaţia a venit din dorinţa de a acorda un ajutor celor care au nevoie, dorinţă ce s-a dezvoltat în decursul anilor în urma a mai multor experienţe trăite.

Sunt numeroase cunoştinţele aflate în situaţii dificile, unde un mic ajutor, le-ar face viaţa mai frumoasă. Consider că atât zona rurală, cât şi cea urbană a ţării ar putea contribui la procesul caritabil prin intermediul acestei aplicaţii.

Ideea a fost inspirată în urma unei priviri de ansamblu în jur, unde am putut observa o mulţime din semeni în situaţii triste.

1.3 Tema proiectului

Tema proiectului este reprezentată de actul de bunătate în sine, de caritate, de oameni ce vor să facă un bine şi vor să ajute printr-o donaţie.

Cei ce vor beneficia de pe urma acestor acte, se doresc a fi în principal copiii. În această categorie se încadrează totuşi şi bătrânii sau cei aflaţi în situaţii speciale.

2 Analiza domeniului problemei

A fost făcută o analiză a domeniului problemei, astfel că la nivel local(Timişoara), din totalul de 30 de ONG-uri, există 0 organizaţii nonguvernamentale care să se ocupe cu ceea ce vrea să se atingă prin acest proiect.

Ce este un ONG?
ONG[2] vine de la organizaţie nonguvernamentală şi reprezintă acea organizaţie care în mod normal este orientată înspre partea caritabilă, cu scopul satisfacerii nevoilor grupurilor de persoane dezavantajate. Adesea, oamenii care iau parte la astfel de acte sunt voluntari și se implică în ajutorul celorlalți.

Câteva ONG-uri la nivel de oraș(Timișoara) care sunt active și care se implică în protecția și educația copiilor sau a persoanelor cu nevoi speciale: Centrul Casa Mama Copil, Centrul Fratii lui Onisim, Fundația Timișoara ‘89, Fundația Petru Voi, Corabia Prieteniei.

Un exemplu bun ar fi, Centrul Casa Mama Copil are în grijă doisprezece copii. Acestora le este oferită o educație decentă, siguranță, ocrotire, un mediu care să le insufle apartenența în cadrul unei familii. Un alt exemplu bun ar fi Centrul Fratii lui Onisim, care oferă adăpost și sprijin pentru copiii străzii, locuitori ai canalelor și gărilor. Aceste din urmă a fost înființat în 1992, la cerințele pastorului Petru Dugulescu și a început cu un număr de șapte copii luați din canale. În scurt timp acesta a crescut ajungând la 43, iar în ziua de azi fiind chiar și mai mare.

Aceste ONG-uri au fost înființate de oameni care doresc să ofere un sprijin și care doresc să facă un bine celorlalți. Există diferite moduri prin care cei cu situații mai bune pot oferi ajutor, precum: donarea unor bunuri nefolosite, mai exact obiecte decorative, haine, electronice, obiecte de papetărie, etc.

Un ajutor cu un caracter caritabil ar putea veni din partea cetăţenilor sub formă de bani, deoarece în Romania există posibilitatea donării acelui 2% din impozitul pe venit. Donația s-ar putea îndrepta către ONG-urile existente. Sau chiar diferite sume de bani, din dorința de a face un bine , sume ce nu intră în categoria acelui 2%.

În Timişoara există numeroase centre, pe lângă cele date exemplu, care s-ar bucura mult de sprijinul celor din jur.

Prin urmare, actul carităţii se îndreaptă începând de la case de copii, case de bătrâni, oameni în situaţii dificile, până la familii care necesită un sprijin pentru a-şi creşte copiii.

Astfel, actul de curaj şi bunătate ar putea fi făcut chiar de acasă. Detalii mai multe pot fi aflate în capitolele ce urmează.
E datoria morală a fiecăruia să ofere ajutor, atâta timp cât e cu putinţă.

Un gest mic azi, pentru un mai bun mâine!

3 Bazele teoretice

3.1 Tehnologii utilizate

Proiectul în sine a fost implementat folosind tehnologia *Windows Forms* pusă la dispoziţie de cei de la *Microsoft*[3], aplicaţia fiind dezvoltată pentru a rula în mediu *Desktop*.

Limbajele de programare folosite pentru implementarea proiectului/aplicaţiei sunt următoarele:

- *C#(C Sharp)* folosit pentru dezvoltarea părţii de backend.
- *PHP* folosit ca interfaţă(*API*) prin care se comunică bazei de date instrucţiuni. Practic, pentru procesul de manipulare a bazei de date.
- *SQL* exclusiv pentru gestiunea bazei de date(*select, insert, update, delete*), folosit împreună cu *PHP*

MySQL a fost utilizat ca sistem de baze de date, rulat pe un server creat local şi care conţine datele ce sunt gestionate de aplicaţie.

Pe post de *server*, a fost folosit propriul laptop, iar pentru parte de creare şi configurare a serverului s-a făcut uz de serviciile aplicaţiei *XAMPP*.

Un serviciu binecunoscut, folosit în aplicaţie e cel de *Gmail*, care oferă o siguranţă şi o încredere mai mare ca altele existente pe piaţă. *Gmail* este cel mai securizat serviciu gratuit de email existent la ora acuală.

Un alt serviciu cunoscut este a celor de la *LibraPay* prin care se pot face transferuri bancare.

3.1.1 C#(C Sharp)

C# este un limbaj de programare apărut în anul 2000, care a fost creat pentru a concura cu un limbaj aflat deja pe piaţă la acea vreme, anume limbajul *Java*.

Acesta este derivat din *C++,* dar prezintă influenţe din alte limbaje. Se află în strânsă legătură cu Arhitectura .Net pe care funcţionează şi este destinat creării de *software* pentru *platforma .Net Framework.*

În *C++,* acel ++ vine de la incrementarea cu 1, dar şi de la faptul că este mai evoluat decât limbajul *C*.

C# se aseamănă cu *C++,* acel # venind de la diez-ul din muzică, unde o notă muzicală urmată de # reprezintă faptul că este mai înaltă cu un semiton.

Caracteristicile limbajului sunt numeroase, aducându-i un mare succes pe piaţă. Dovadă a succesului, o reprezintă faptul că face parte din top 10 cele mai utilizate limbaje la nivel mondial[4].

Principalele calităţi pe care le prezintă ar fi:

- Simplitatea, utilitatea generală
- e modern şi are o productivitate mare
- în cazul aplicaţiilor mari şi complexe, prezintă stabilitate şi durabilitate
- este total orientat pe obiecte(astfel că orice entitate, în C# reprezintă un obiect)
- în medii distribuite, oferă suport pentru dezvoltarea componentelor, astfel, poate fi numit şi limbaj orientat către componente

O altă trăsătură importantă o prezintă suportul pentru internaţionalizare, adică oferă posibilitatea de a scrie aplicaţii care se pot adapta într-un timp scurt în diferite regiuni ale lumii, unde sunt vorbite limbi diferite, fără a fi nevoie de schimbări în arhitectura software.

Ca urmare, *C#* pune la dispoziţie posibilitatea de scriere a unor aplicaţii pentru sisteme complexe care funcţionează pentru o gamă largă de sisteme de operare, cât şi pentru sisteme dedicate(*embedded systems*), cum ar fi: mp3 playere, semafoare, telefoane mobile, ceasuri digitale etc.

Arhitectura .Net este o componentă *software* care oferă un mediu de programare şi de execuţie a aplicaţiilor pentru sistemele de operare *Microsoft*.

.Net Framework este un mediu care pune la dispoziţie posibilitatea dezvoltării şi rulării aplicaţiilor şi a serviciilor Web, independente de platformă.

.Net e format din două părţi importante.
O explicație foarte bună legată de aceste părți, am găsit în cuprinsul cărții scrisă de Constantin și Susana Gălățean: C# pentru liceu, Programare în Visual C# 2008 Express Edition:

- ***"Common Language Runtime****(CLR*) care este mediul de execuţie al programelor. Totodată aceasta este partea ce se ocupă cu managementul şi execuţia codului scris în limbaje specifice .Net. CLR prezintă şi servicii importante precum portabilitatea aplicaţiilor, securitatea acestora, managementul memoriei şi tratarea excepţiilor."
- ***„Base Class Library****"* reprezintă biblioteca de clase .Net şi oferă servicii precum interfaţa cu utilizatorul, posibilitatea conectării cu bazele de date şi accesarea datelor, dezvoltarea aplicaţiilor web. Codul bibliotecii este unul precompilat, şi

este organizat de regulă în funcţii, numite metode, pe care programatorul le poate apela din propriul program. Metodele, la rândul lor, aparţin claselor, care sunt organizate şi separate între ele cu ajutorul spaţiilor de nume (*namespaces*). Ceea ce este de reţinut, este că programatorii combină propriul cod cu codul Bibliotecii de Clase .NET pentru producerea de aplicaţii. ”

Prezentarea platformei .Net:

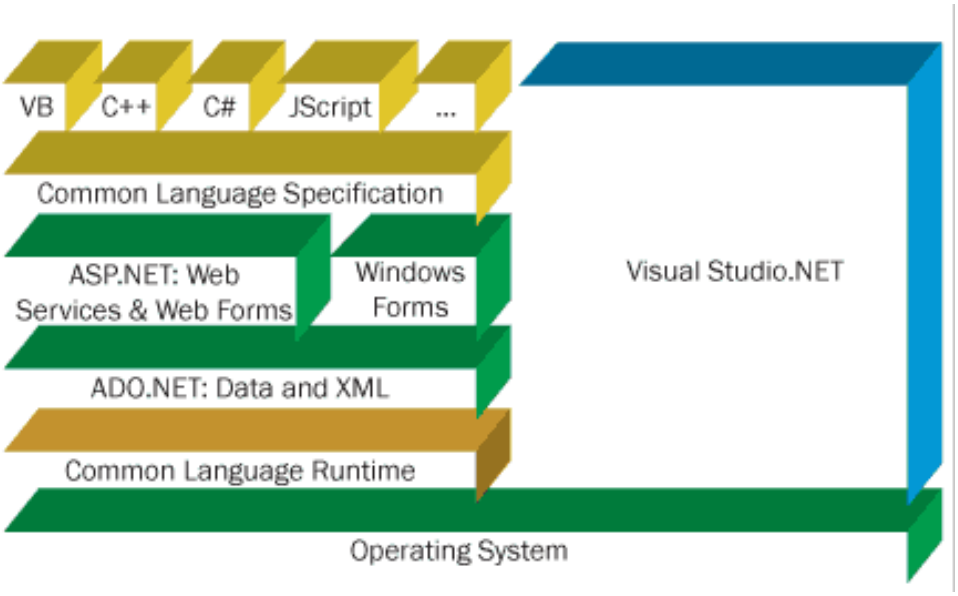


Figura 1.[5]

În partea stângă a imaginii[5](figura 1), sunt prezentate componentele platformei, prezentate generic. Pe nivelul de sus se află limbajele acceptate de platformă, după cum se vede, şi C#. Următorul nivel îi corespune CLS-ului, un set de trăsături ale platformei, care asigură interoperabilitatea limbajelor .Net.

Dintre elementele ce au fundal verde, noi ne ocupăm doar de Windows Forms adică de crearea de aplicaţii ce deţin interfaţă grafică.

Cea mai mare importanţă dintre componentele platformei o prezintă Common Language Runtime, fiind mediul de execuţie care conţine toate tool-urile ce permit executarea aplicaţiei dezvoltate.

Dacă în C sau C++ programatorul trebuie să decidă când trebuie alocată o anumită zonă de memorie sau când trebuie eliberată, aici lucrurile stau altfel. În C# este integrat un mecanism de gestiune a memoriei, mecanism prin care memoria este eliberată automat când anumite obiecte nu mai sunt utilizate. C# mai conţine şi un mecanism de tratare automată a excepţiilor, mai exact a erorilor apărute în momentul rulării aplicaţiei. Chiar şi în acest caz, programatorul

trebuie să îşi creeze un sistem propriu de tratare a erorilor, în funcţie de modul de funcţionare al aplicaţiei.

Nu o să fie omisă aducerea în discuţie a categoriei limbajelor din care face parte C#.

Există două tipuri de limbaje folosite, precum limbaje interpretate şi limbaje compilate. C# face parte din cea de-a doua categorie.

Cum funcţionează un limbaj compilat? O explicaţie despre modul de funcţionare e în cele ce urmează:

- Codul scris într-un astfel de limbaj se numeşte cod sursă.
- Codul sursă este transformat de compilator într-un cod apropiat de nivelul maşinii, adică un limbaj cunoscut de calculator, numit cod executabil(de exemplu, fişiere cu extensia .exe).
- În urma compilării pot apărea erori, ceea ce duce la nefuncţionarea programului. Pentru a merge totul normal, erorile trebuie înlăturate, iar codul va trebui recompilat. Un cod scris corect, ce trece de procesul de compilare, va produce un cod executabil, iar astfel, aplicaţia va fi rulată.

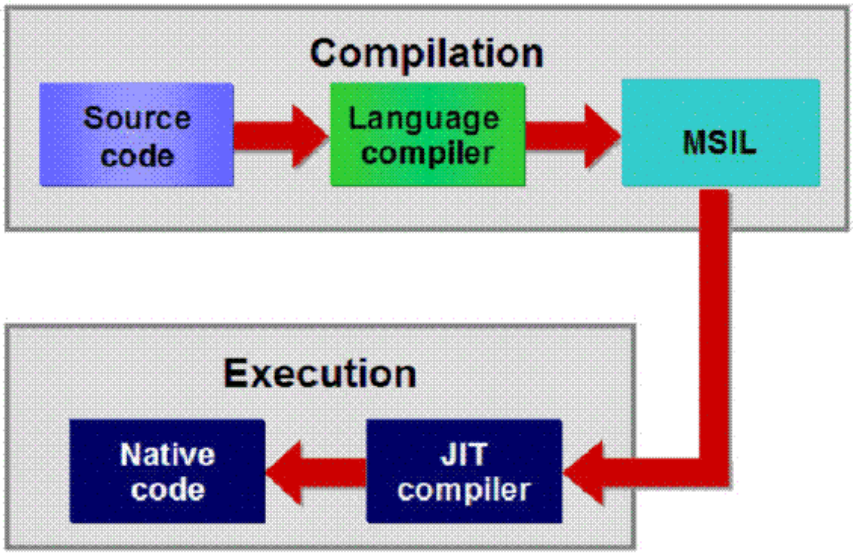


Figura 2.[5]

Totuşi, chiar dacă C# este un limbaj compilat, lucrurile stau puţin diferit în cazul nostru.

Astfel că după compilarea unui program scris în C#, nu se creează cod executabil, ci este creat un fişier assembly care are extensia .exe sau .dll. Un fişier de acest fel nu poate să fie rulat pe un sistem pe care nu există infrastructura .Net.

Ceea ce conţine un astfel de fişier este un cod special, denumit *CIL(Common Intermediate Language)* ce defineşte un set de instrucţiuni portabile, care sunt independente de orice tip de procesor şi platformă.

Mai sus, pe diagramă[5](figura 2), în loc de CIL apare MSIL, care vine de la Microsoft Intermediate Language.
Un program compilat în formatul CIL poate fi rulat pe orice sistem pe care a fost instalat Common Language Runtime.
CLR activează un compilator special în moementul execuţiei unui program, compilator numit JIT(Just in Time). Codul CIL este preluat de JIT şi transformat în cod executabil(binar).

Acesta este modul în care se asigură portabilitatea aplicaţiilor .Net.

Pentru mai multe detalii, se poate consulta pagina celor de la Microsoft:
https://docs.microsoft.com/en-us/dotnet/csharp/

3.1.2 PHP

Numele limbajului provine din limba engleză, de la *PHP*[6]: *Hypertext Preprocessor*, este un limbaj *open souce* şi *server-side*, ce e folosit pe scară largă pentru a crea pagini Web dinamice şi aplicaţii independente în mod CLI, linie de comandă.

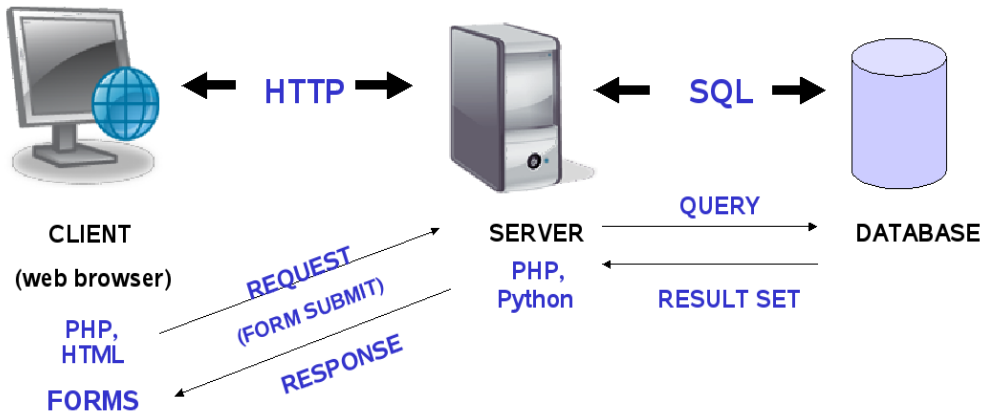


Figura 3. [7]

PHP face parte din categoria llimbajelor interpretate, ceea ce înseamnă că fişierele ce au cod *PHP*, sunt interpretate ca atare în momentul execuţiei, adică la execuţia unei părţi din codul

sursă, nu este necesară modificarea codului într-o formă intermediară(binar), ci este folosit exact aşa cum a fost scris.

Principalul scop al *PHP*-ului este de a permite celor ce scriu cod să creeze rapid pagini web generate dinamic, însă cu acesta se pot realiza mult mai multe.

Modul folosit în această aplicaţie este de API(Aplication Programming Interface) cu baza de date. Scopul exact va fi redat mai jos, în capitolul 5.

3.1.3 MYSQL

MySQL reprezintă un sistem de baze de date ce e utilizat pe web și care rulează pe un server.

A fost dezvoltat de firma Oracle și poartă numele fiicei co-fondatorului Montz Wildenius, anume My.

Este ideal atât pentru aplicații de dimensiuni mici cât și pentru cele de dimensiuni mari, prezentând atât ușurință în utilizare, cât și viteză.

MySQL poate fi descărcat și folosit gratuit și rulează pe diferite platforme.

Informațiile într-o bază de date *MySQL* sunt stocate în tabele, care sunt structurate în linii și coloane. Informația din tabele este gestionată cu ajutorul limbajului *SQL*, care adesea e folosit împreună cu *PHP* ce împreună sunt *cross-platform*.

3.1.4 SQL

SQL(Structured Query Language)[8] este un limbaj de programare utilizat pentru manipularea și gestiunea bazelor de date relaționale care la origine folosește algebra relațională.

Scopul folosirii SQL este inserarea datelor, interogații a bazelor de date, actualizarea și ștergerea unor informații, modificarea și crearea schemelor, precum și accesul la date.

Caracteristici ale SQL:

- Este prezentat în engleză și folosește termeni precum *select, insert, update, delete* etc. ca parte din setul de comenzi
- Este un limbaj neprocedural. În SQL trebuie să specifici ce dorești să obții, nu modul cum vrei să le obții.
- SQL operează asupra unei singure înregistrări la un moment dat. Metoda comună unui set de înregistrări este reprezentată de un tabel.
- Limbajul poate fi folosit de programatori de aplicații, personal de management și multe alte tipuri de utilizatori

- În SQL, în funcție de task-ul avut, există numeroase tipuri de comenzi, precum:
    - Date interogate
    - Inserare, extragere și ștergere a rândurilor într-un tabel
    - Crearea, modificarea și ștergerea obiectelor de tip bază de date
    - Controlul accesului la baza de date și la obiectele de tip bază de date

SQL reprezintă un standard în domeniu fiind standardizat de ANSI-ISO și e cel mai popular limbaj utilizat pentru crearea, modificarea, regăsirea și manipularea datelor de către SGBD-urile(Sistemele de Gestiune a Bazelor de Date) relaționale.

3.1.5 Windows Forms

Multe dintre sistemele de operare utilizate, creează evenimente și utilizează ferestre pentru a interacționa cu utilizatorii. În ceea ce privește *Windows Forms[3]*, pentru a crea o fereastră, sunt utilizate rutine ale bibliotecii Win32.

Windows Forms conține o bibliotecă de clase, structuri, interfețe, metod șamd. Care fac parte din *framework-ul Microsoft .Net*. Acesta pune la dispoziție o platformă pentru a putea crea aplicații complexe pentru *desktop* sau *laptop*.

Ceea ce în programarea *Windows* se numește fereastră, în .Net se numește *form*. De aici și denumirea de *Windows Forms*.

Ce sunt formurile?
Formurile reprezintă obiecte care redau unele proprietăți ce definesc modul de afișare și metode care răspund unor evenimente ce reprezintă interacțiunea cu utilizatorul.

În urma setării proprietăților unui *form* și scriind codul ce reprezintă răspunsul la evenimente, se creează obiecte care se comportă conform cerințelor cerute de aplicație.
Un form este o instanță a clasei *Form*(sau ale unor clase derivate din aceasta).

Crearea unei aplicații *Windows* necesită utilizarea de clase din doua assembly-uri standard[9]:

- *System.Windows.Forms.dll*, care conține clase pentru reprezentarea elementelor grafice(formulare, butoane etc.) și mecanismul de execuție pentru acestea
- *System.Drawing.dll*, care conține structurile folosite pentru partea de geometrie(dreptunghiuri, dimensiuni etc.) și funcții pentru desenarea pe ecran

3.1.5.1 Elementele de bază: clasele Control și Form

Intern, un formular *Winform(Figura 4)* este reprezentat printr-un arbore de obiecte(Figura 5).

Un exemplu:

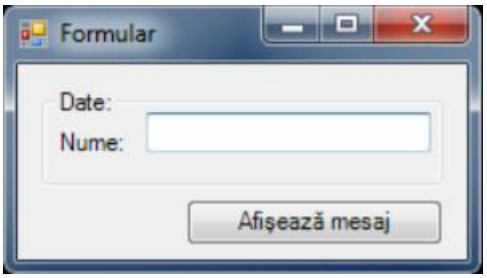


Figura 4.[9]

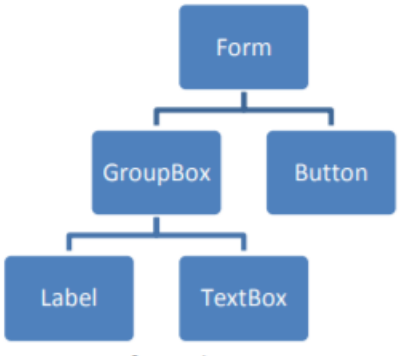


Figura 5.[9]

Fiecărui obiect vizual(control), care există în cadrul unui formular(buton, text etc.), îi corespunde un obiect din arbore.

Pentru a construi un formular funcțional, trebuie parcurse mai multe etape:

- Crearea unui obiect de tip formular și a obiectelor corespunzătoare controalelor conținute
- Setarea proprietăților pentru fiecare obiect în parte(poziție,dimensiune, text, culori etc.)
- Adăugarea obiectelor în arbore
- Setarea acțiunilor care trebuie îndeplinite de controale
- Lansarea formularului prin metoda statică Application.Run sau una din metodele ShowDialog sau Show, conținute în clasa Form

Elementele necesare pentru construirea arborelui se găsesc în cadrul clasei Control.

3.1.5.2 Poziționarea controalelor

În cadrul fiecărei ferestre, fiecare control ocupă o zonă rectangulară(Figura 6). Sistemul de coordonate are punctul de origine în colțul din stânga sus al ferestrei, iar ca unitate de masură, e folosit pixelul.

Pentru exprimarea elementelor geometrice se utilizează următoarele structuri de bază definite în System.Drawing(Figura 6):

- Point: coordonatele unui punct prin proprietățile X și Y
- Size: dimensiunile unui dreptunghi/control prin proprietățile Width și Height
- Location: proprietate de tip Point ce determină poziția față de controlul părinte

Exemplu de poziționare a unui obiect control:

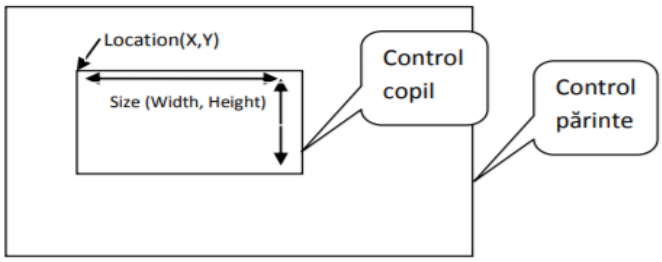


Figura 6.[9]

Alte proprietăți ale clasei Control care determină poziționarea controlului: Top, Left, Bottom, Right, Margin, Padding, Anchor, Dock etc.

Mai multe detalii la adresa: https://docs.microsoft.com/en-us/dotnet/api/system.windows.forms.control.right?view=netframework-4.8

3.1.5.3 Alte proprietăți

Toate controalele utilizate în cadrul bibliotecii *Windows Forms*, au parte de proprietăți puse la dispoziție de clasa *Control*. Modul folosirii proprietăților poate varia de la un control la altul.

Cele mai importante proprietăți(pe lângă cele de mai sus) sunt:

- *Text*: permite modificarea textului afișat de control
- *Name*: permite denumirea controlului după bunul plac, pentru o asociere mai usoară
- *Tag*: permite asocierea unui obiect propriu controlului
- *ForeColor, BackColor*: permit schimbarea culorii pentru textul folosit și pentru fundalul controlului
- *Font*: permite schimbarea stilului utilizat pentru afișarea textului în cadrul controlului
- *Enabled*: oferă posibilitatea activării și dezactivării controlului

3.1.5.4 Atașarea de acțiuni

Prin afișarea formularului, se produce următorul proces:

- Aplicația pornește o buclă în care se preiau mesajele venite de la sistemul de operare(apăsări de taste, modificări ale poziției mouseului etc.)
- Conform fiecărui mesaj preluat, este identificat controlul responsabil de procesarea mesajului și se apelează funcția ce corespunde controlului respectiv
- Funcția apelată realizează modificările necesare asupra controlului(schimbă textul afișat, schimbă caracteristicile unei ferestre, comută starea butonului etc.), după care face apel la codul specificat de utilizator

Pentru mai multe detalii, poate fi consultată următoarea pagină: https://docs.microsoft.com/en-us/dotnet/framework/winforms/creating-event-handlers-in-windows-forms

3.1.5.5 Structura standard pentru un formular

La adăugarea unui formular nou se generează automat clasa derivată din *Form*. Pe urmă, codul care aparține de clasa respectivă, e generat în două fișiere distincte: *Formular.cs* unde se află codul ce poate fi modificat de utilizator și *Formular.Design.cs* care conține codul generat automat și care nu poate fi modificat manual(câmpuri corespunzătoare controalelor, implementarea funcției *InitializeComponent*).

Scrierea aplicațiilor de acest gen(*Winform*) este posibilă prin utilizarea *componentei Windows Forms* din *Visual Studio*.

Acesta oferă funcționalități de bază pentru aplicații *Winforms* precum proiecte de tip *Windows Forms Applications* care au adăugate referințe pentru *System.Windows.Forms* și *System.Drawing* și generează automat codul pentru programul principal și un formular gol.

De menționat că *Windows Forms* a fost lansat sub licența celor de la *MIT(Massachusetts Institute of Technology)* și este un *proiect open source*. Acesta poate fi găsit pe *GitHub* la adresa: https://github.com/dotnet/winforms.

Mai multe informații despre aceste ferestre pot fi găsite la: https://docs.microsoft.com/en-us/dotnet/framework/winforms/

3.1.6 Gmail

Gmail[10] reprezintă o aplicație web, implementată și pusă la dispoziția utilizatorului de cei de la Google.

Oferă servicii de securitate ridicată și o bună calitate în comunicare.

Modul de utilizare a serviciului în cazul nostru va fi relatat la partea de implementare, mai exact, capitolul 5.

3.1.7 LibraPay

LibraPay[11] este un serviciu de transfer bancar românesc, prin care utilizatorii pot face donații.

3.2 Tool-uri folosite

3.2.1 Microsoft Visual Studio

Microsoft Visual Studio[12] este un mediu de dezvoltare integrat(*integrated development environment – IDE*). Acesta poate fi folosit pentru a dezvolta aplicații consolă și aplicații cu interfață grafică pentru toate platformele suportate de *Microsoft Windows(ex: .Net Framework, Windows Mobile etc.).*
Microsoft Visual Studio oferă editor, compilator/debugger și mediu de proiectare(designer) pentru mai multe limbaje de programare precum: *Microsoft Visual C#, Microsoft Visual C++, Microsoft Visual Basic, Micosoft Visual Web Developer*.
Acesta vine și cu o facilitate importantă a editorului de cod, anume *IntelliSense*, care atunci când se scrie un nume sau un cuvânt cheie în editor, se poate utiliza pentru a genera automat tipurile, variabilele și metodele vizibile într-un anumit context, urmând ca programatorul să o aleagă pe cea dorită pentru a o utiliza. Pe de altă parte, oferă siguranța că ceea ce a fost scris, e corect.

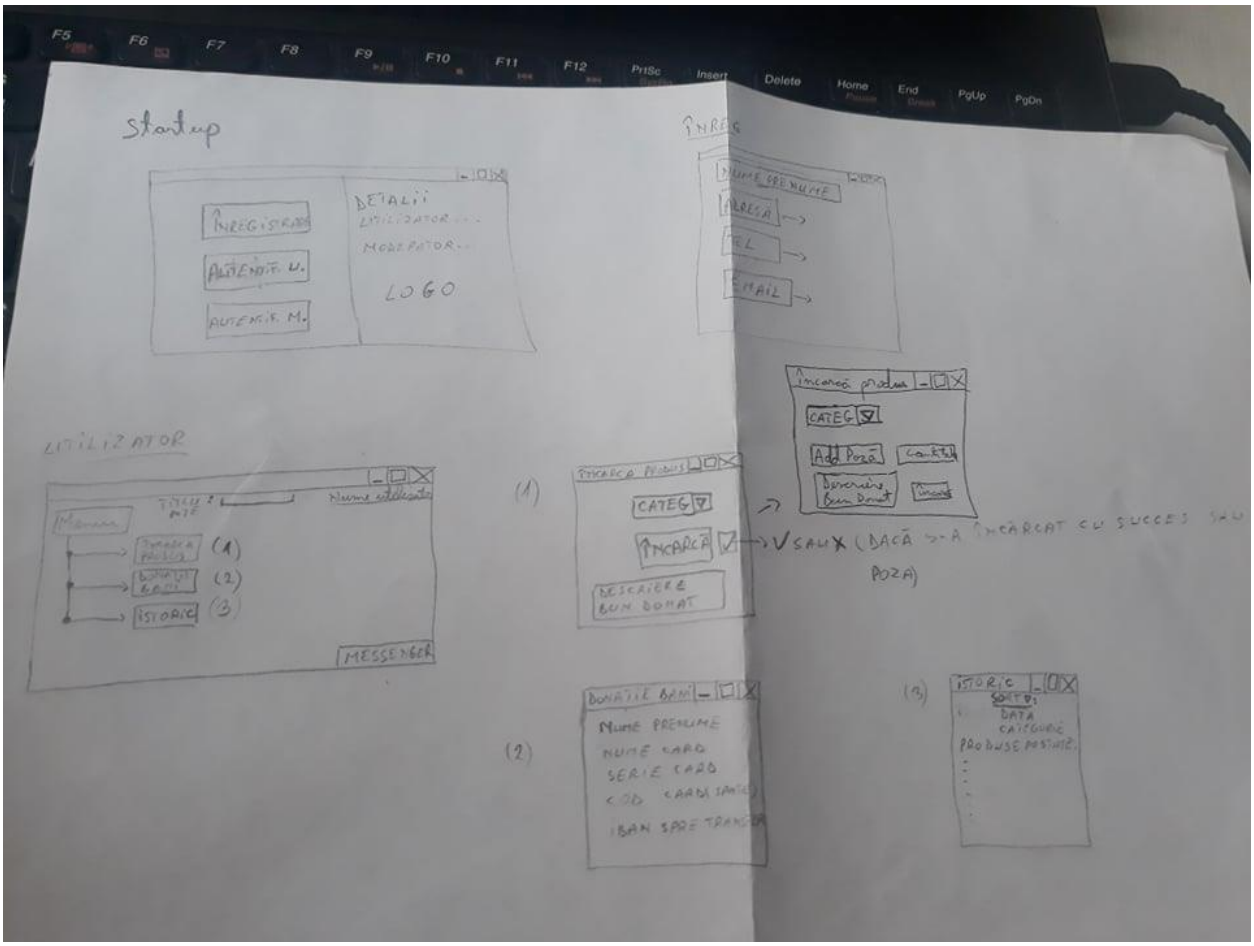
3.2.2 XAMPP

Xampp[13] reprezintă un pachet de programe *free software, open source* și *cross – platform*. Constă în *Apache HTTP Server, MySQL Database* și interpretoare pentru scripturile scrise în limbajele *PHP* și *Perl*.

Este utilizat în principal pentru dezvoltarea proiectelor web și asigură suport pentru manipularea bazelor de date.
După instalare, adresa de localhost a serverului *Xampp* poate fi tratată ca un *server* la distanță. Utilizatorul *MySQL* implicit este "*root*", iar parola este inexistentă implicit. Cu ajutorului programului *resetroot.bat* din subdirectorul *mysql* al directorului unde e instalat *Xampp*, se poate reseta utilizatorul și parola.

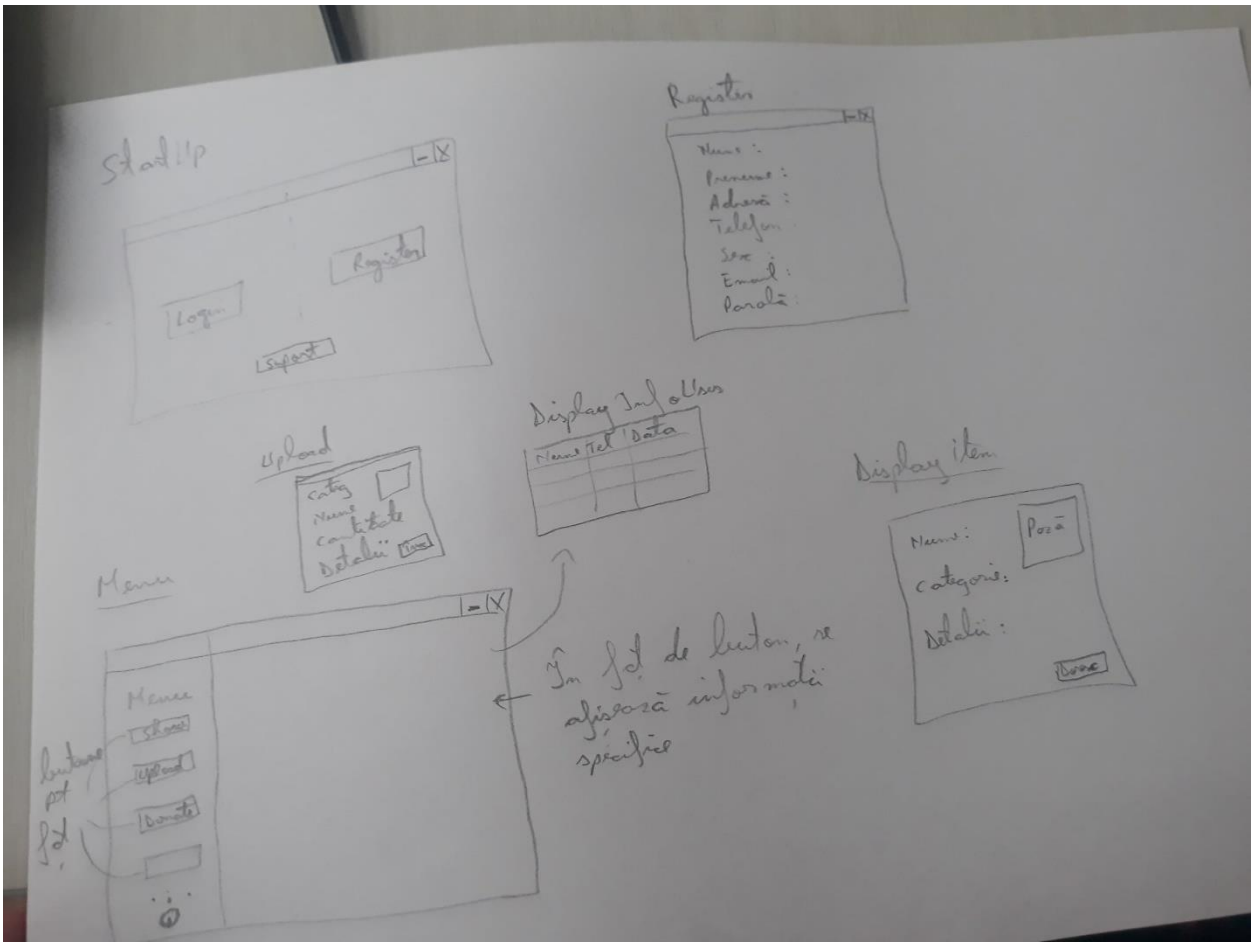
4 Soluția propusă și metodologia de proiectare/dezvoltare

Inițial proiectării aplicației, partea de interfață a fost făcută pe foaie după cum urmează:

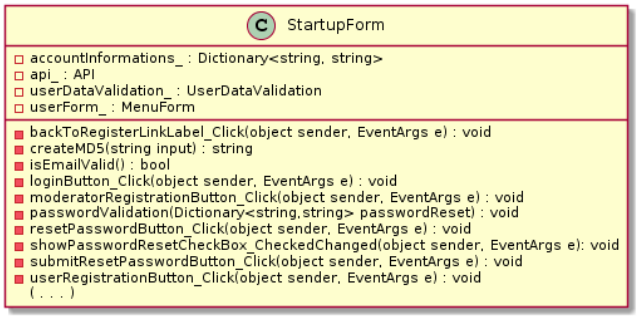


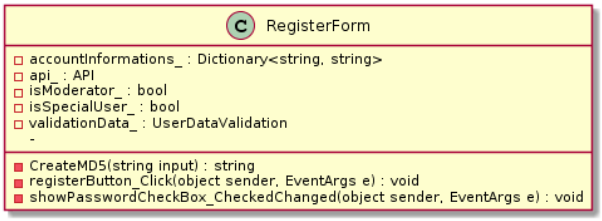
Spun inițial, deoarece pe parcurs, după implementarea de prima dată, am făcut mici schimbări la unele interfețe. Astfel că am ajuns după o perioadă de căutare și informare legată de un design mai bun, să schimb în totalitate partea de interfață, plus adăugarea unor noi funcționalități odată cu cea nouă.

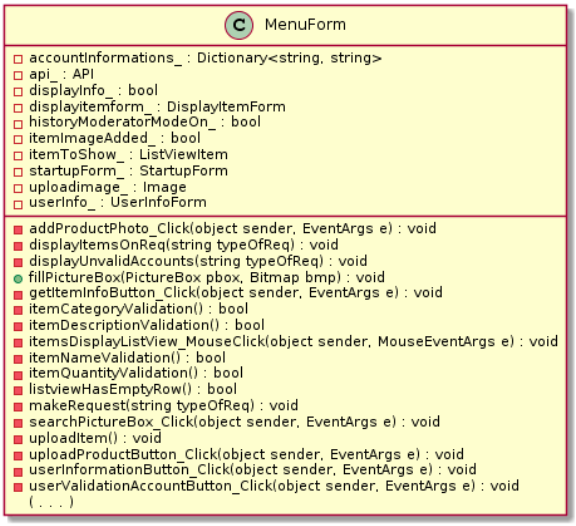
Noul design, prezentând o estetică mai plăcută(forma existentă în momentul de față):

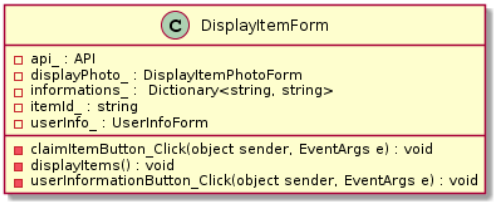


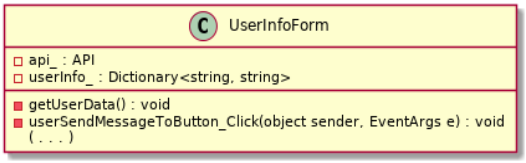
Pentru descrierea structurii claselor, s-a folosit PlantUml, iar clasele ce fac parte din aplicație sunt după cum urmează:

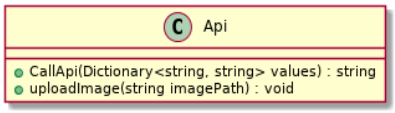


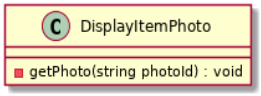


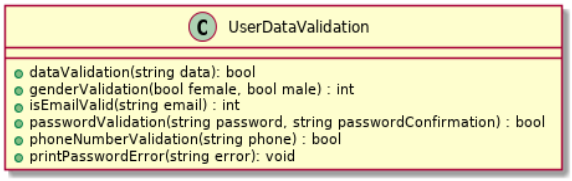


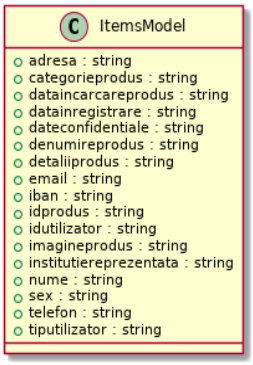










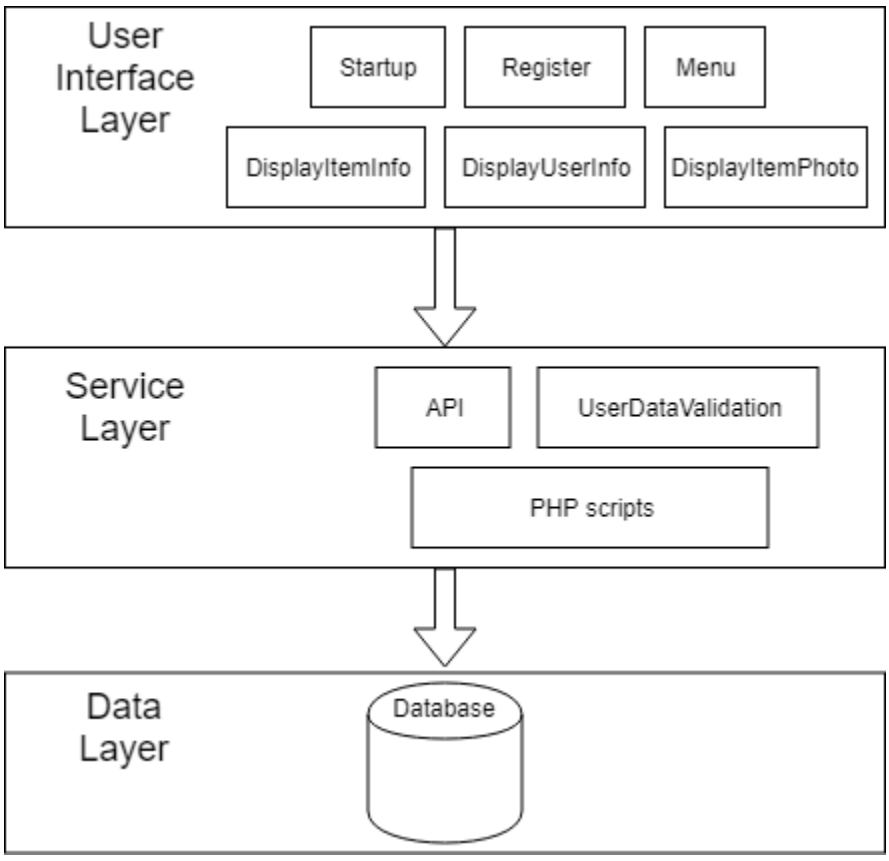


Mai sus se poate vedea structura fiecărei clase prezente în proiect. Clasele ce conțin „*Form*” la final sunt derivate ale clasei *Form*, care pune la dispoziție setul de controale și partea grafică pentru contruirea unei ferestre. Celelalte, au fost folosite în diferite moduri precum *API*(trimiterea cererilor înspre server alături de datele importante), *ItemsModel* unde s-au stocat informații luate din baza de date, *UserDataValidation* unde s-au făcut validările informațiilor pentru parte de înregistrare.

De precizat că *ItemsModel* e format în totalitate din accesori, adică un set de metode *get()* și *set()* prin care pot fi accesate câmpuri private ale clasei pentru setare sau citire a informației.

În unele diagrame se poate observa terminația (. . .) de la final. Asta reprezintă faptul că în diagramă nu au fost incluse toate metodele și câmpurile clasei, ci doar cele cu care s-a lucrat mai mult și care întrunesc partea de funcționalitate, adică cele mai importante.

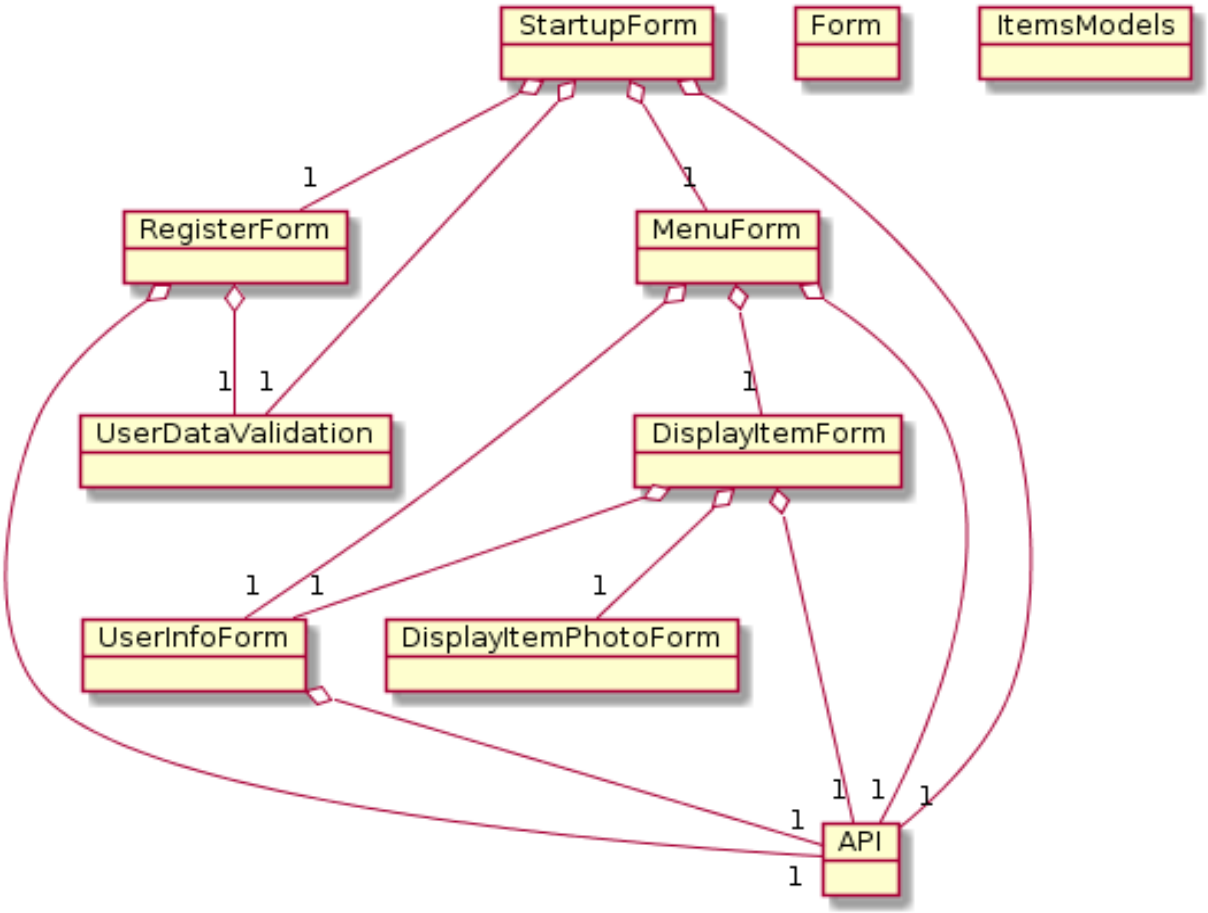
Mai jos se poate vedea structura pe layere a aplicației. În prima parte e partea de interfață cu utlizatorul. Acesta prin interacțiunea cu interfața, trimite o cerință care merge înspre layerul cu servicii care la rândul lui apelează unele scripturi de pe server. Acesta în ultimă instanță trimite unele comenzi bazei de date, care întoarce diferite rezultate.



Reprezentarea accesului la câmpurile claselor sau metode s-a făcut prin următoarele figuri:

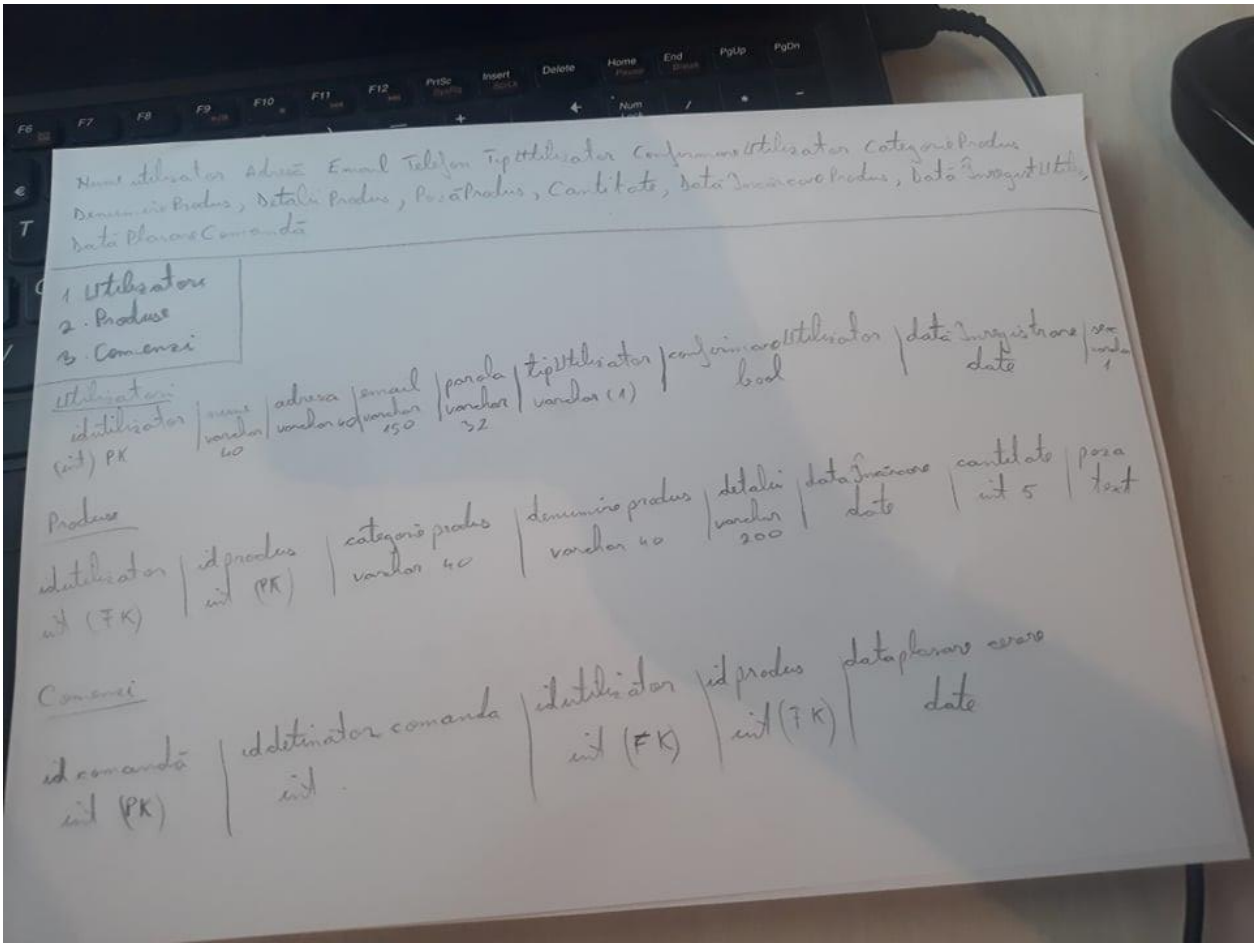
-	□	■	private
#	◇	◆	protected
~	△	▲	package private
+	○	●	public

Schema arhitecturii poate fi regăsită mai jos:



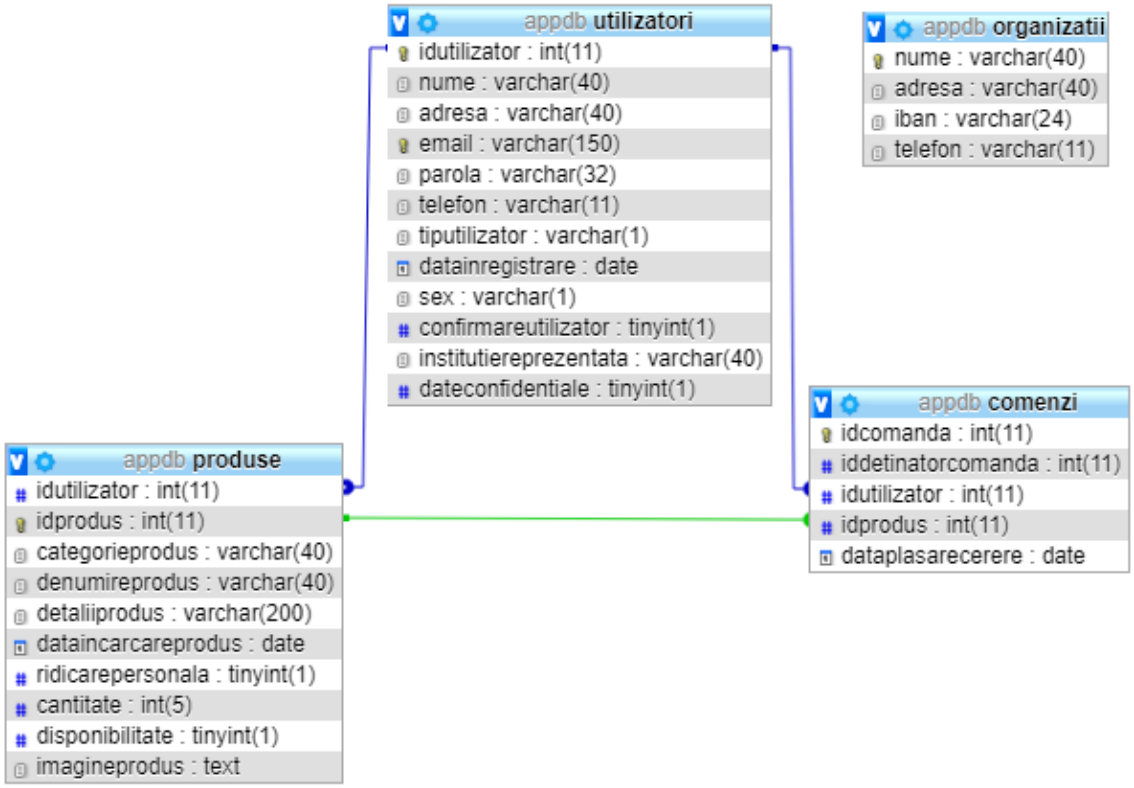
Pentru o vedere mai lizibilă a arhitecturii, s-a omis legătura ce reprezintă moștenirea clasei *Form*. Astfel că toate clasele ce conțin „*Form*” în numele propriu, moștenesc direct această clasă.

Pentru configurarea bazei de date, s-au făcut inițial normalizările pe foaie, iar conform acestora, a fost creată și configurată baza de date cu tabelele respective.



De menționat este că s-au adăugat câteva coloane noi(coloana *disponibilitate* in tabela produse, *ridicarepersonala* in aceeasi tabela etc.), față de cele prezente pe foaie, plus tabela *organizatii*, care contine informatii despre anumite organizatii existente și care vor să dispună de ajutor din partea *Bee Charity*.

Structura bazei de date e următoarea:



Baza de date e formată dintr-un total de patru tabele.

Unul care stochează informațiile legate de utilizatori, unul informațiile despre produse, într-unul sunt stocate comenzile și unul care conține informații despre organizații.

Prima tabelă(*utilizatori*) e cea care stochează utilizatorii și conține o cheie primară, anume idutilizator. Aceasta e unică și pe baza ei, se vor adăuga în tabela produse, înregistrări care să figureze fiecărui utilizator. Tabela *produse* conține o cheie primară denumită idprodus și o cheie străină(idutilizator), ce figurează sub formă de relație *one-to-many* cu cheia primară din prima tabelă. În mod asemănator e structurată și tabela de comenzi, care prezintă cheile străine idprodus și idutilizator, pentru a putea fi aflate informații despre utilizatorul și produsul postat de acesta, dar și cheia primară idcomandă pentru a putea oferi comenzi unice și pentru a putea afla date stricte legate de o comandă în parte.

Mai există și tabela *organizatii* care stochează date despre anumite ONG-uri care doresc să fie văzute de către utilizatori, care pot ajuta prin donații directe, aceste organizații.

5 Implementare

Aplicația a fost implementată folosind un număr de șase clase de tipul *Winform* și trei clase ajutătoare, plus clasa *Program* în care se află metoda *main*, de unde se rulează aplicația. Informația a fost stocată în baza de date cu numele "*appdb*" care vine de la *application database* și care rulează pe serverul local de la adresa: http://127.0.0.1/phpmyadmin/.

În timpul configurării bazei de date, s-au luat mici decizii de schimbare a unor tabele, mai exact de adăugare a unor coloare noi pentru a putea face unele verificări pentru unele cerințe ale aplicației și a unor constrângeri.

Exemple de cod din momentul lucrării la configurarea bazei de date:

- Pentru optimizarea memoriei bazei de date, s-a ales ca tipul utilizatorului să fie format din o singură literă(M = moderator, U = utilizator, S = utilizator special) și sexul acestuia la fel

```
ALTER TABLE `utilizatori` CHANGE `tiputilizator` `tiputilizator` VARCHAR(1) CHARACTER SET utf32 COLLATE utf32_general_ci NOT NULL;

ALTER TABLE `utilizatori` CHANGE `sex` `sex` VARCHAR(1) CHARACTER SET utf32 COLLATE utf32_general_ci NOT NULL;
```

- Adăugarea cheilor și constrângerilor pe tabele

```
ALTER TABLE `produse` ADD FOREIGN KEY (`idutilizator`) REFERENCES `utilizatori`(`idutilizator`) ON DELETE CASCADE ON UPDATE CASCADE;

ALTER TABLE `comenzi` ADD FOREIGN KEY (`idutilizator`) REFERENCES `utilizatori`(`idutilizator`) ON DELETE RESTRICT ON UPDATE RESTRICT;
```

Prezentarea funcționării:

Modul de lucru al aplicației constă în crearea unui obiect de tipul clasei *StartupForm*, în interiorul clasei *Program*, din namespace-ul *BeeCharity*, în care se află metoda *main()*.

```
namespace BeeCharity
{
    static class Program
    {
        [STAThread]
        static void Main()
        {
            bool instanceCountOne = false;

            using (Mutex mtex = new Mutex(true, "MyRunningApp", out
instanceCountOne))
            {
                if(instanceCountOne)
                {
                    Application.EnableVisualStyles();
                    Application.SetCompatibleTextRenderingDefault(false);
                    Application.Run(new StartupForm());
                    mtex.ReleaseMutex();
                }
                else
                {
                    MessageBox.Show("Bee Charity deja ruleaza!");
                }
            }

        }
    }
}
```

În *main* am folosit o variabilă de tip boolean ce ajută la verificarea stării aplicației(dacă aceasta rulează deja sau nu) și un obiect *Mutex pentru a oferi threadului care rulează ownership* pe bucata de cod cuprinsă între acoladele imediat următoare și care returnează un *boolean* dacă mutex-ul există și aplicația rulează.

Prima metodă folosită, *EnableVisualStyles(),* face posibilă folosirea efectelor vizuale, anume: culori, fonturi și alte elemente care ajută la crearea formului din punct de vedere vizual.

A doua metodă, *SetCompatible RenderingDefault(false)* este folosită pentru a configura modul inițial de afișare a textului în controalele ce urmează a fi adăugate în aplicație. Parametru primit de metodă e setat pe valoarea ***false***, astfel ca noile controale să facă uz de *GDI(Graphic Design Interface)* bazată pe clasa *TextRendering*.

Metoda folosită în ultima instanță, e cea de *Run()* care primește ca parametru un obiect de tipul clasei *StartupForm*. Prin intermediul acestei metode, e practic rulată aplicația.

Ca implementare, apelurile înspre baza de date se fac prin prisma unor fișiere *.php* care sunt executate în urma unui anumit tip de cerință venită din aplicație, prin intermediul clasei *API*.

Partea de *PHP* a fost utilizată din principiul securității datelor. Pentru a le proteja, în cazul în care se încearcă un atac la adresa aplicației, dacă cererile pentru procesarea diferitelor informații se făceau direct din clasele „*form*”, structura bazei de date putea fi aflată cu ușurință și datele puteau fi compromise relativ ușor. În cazul folosirii limbajului *php*, din aplicație se trimite doar tipul cererii înspre *server*, care prezintă o securitate avansată și proprie. Cerința e preluată de un fișier php specific acesteia, care nu poate fi accesat din afară. Astfel, persoana ce încearcă să acceseze informațiile din interiorul bazei de date, nu are succes în operațiunea de furt de date.

Cerințele preluate de fișierul php în cauză, sunt scrise în *SQL* și sunt trimise bazei de date pentru a putea fi interpretate și pentru a produce un rezultat. În caz de succes, datele vor fi afișate în mod contextual în unele controale din interiorul aplicației, iar în mod contrar, se vor afișa mesaje specifice cu problemele apărute.

În continuare, va fi prezentat cuprinsul claselor care întrunesc acest proiect. De menționat este că se vor lua doar părțile(metodele) cu importanță mai ridicată.

5.1 StartupForm

Partea de *login* a fost implementată după cum urmează:

```
private void loginButton_Click(object sender, EventArgs e)
        {
            api_ = new BeeCharity.API();
            accountInformations_["type"] = "login";
            accountInformations_["username"] = usernameTextBox.Text;
            accountInformations_["password"] =
CreateMD5(passwordTextBox.Text);
            string response = api_.CallApi(accountInformations_);

            accountInformations_["userLoggedInId"] = response.Remove(0,
1).ToString();
            accountInformations_["userType"] = response[0].ToString();
```

Metoda face apel la server prin intermediul obiectului api_, metoda CallApi(). Aceasta primește ca parametru un obiect de tip dicționar(accountInformations_) care e populat cu datele necesare tipului de cerință pentru care se face apelul(inregistrare in acest caz).

Continuarea:

```
            if (accountInformations_["userType"] == "M" ||
accountInformations_["userType"] == "S" || accountInformations_["userType"]
== "U")
            {
                this.Hide();
                accountInformations_.Remove("password");
                userForm_ = new MenuForm(this, accountInformations_);
                userForm_.Show();
                usernameTextBox.Text = "";
                passwordTextBox.Text = "";
            }
```

```
            else if (response == "invalidAccount")

            ...
            // afișarea unor mesaje în cazul unor probleme
            ...

        }
```

În caz de succes a cererii(se returnează din baza de date, tipul utilizatorului), se va crea un obiect de tip *MenuForm* ce prezintă meniul aplicației. În caz de răspuns negativ din partea serverului, se vor afișa mesaje specifice de gestiune a problemei în cauză.

Implementarea părții de resetare a parolei se poate vedea mai jos:

```
private void submitResetPasswordButton_Click(object sender, EventArgs e)
        {
            Dictionary<string, string> passwordReset = new Dictionary<string,
string>();
            passwordReset["type"] = "passwordReset";
            passwordReset["emailPasswordReset"] =
emailPasswordResetTextBox.Text;
            passwordReset["newPassword"] =
CreateMD5(passwordResetTextBox.Text);
            string response = "";

            userDataValidation_ = new UserDataValidation();
            int validEmail =
userDataValidation_.isEmailValid(emailPasswordResetTextBox.Text);
```

Pentru partea de validare a emailului și a parolei noi, am folosit clasa *UserDataValidation* pentru a verifica formatul emailului și a parolei.

```
            if (validEmail == 1)
            {
                bool validPassword =
userDataValidation_.passwordValidation(passwordResetTextBox.Text,
passwordResetConfirmationTextBox.Text);
                if (validPassword == true)
                {
                    response = api_.CallApi(passwordReset);
                    if (response == "resetPasswordSuccess")
                    {
                        MessageBox.Show("Pentru resetarea parolei, vi-a fost
trimis un email de confirmare");

                        setSomeVisibility();//metodă folosită pentru setarea
//vizibilității unor controale
                    }
                    else if (response == "resetPasswordFailed")

                    ...
                  //Afișarea unor mesaje în cazul unor probleme apărute
                  ...

        }
```

În urma metodelor apelate pentru a valida datele, au fost întoarse niște valori și pe baza acestora, în caz de succes, se va face operația de resetare a parolei: se apelează baza de date, se interpretează codul *php* specific resetării parolei de pe *server.*

Codul PHP conform acestei operațiuni:

```
<?php
	...
	//conectarea la baza de date și decode-ul fișierului json trimis din aplicație

	$currentTime = round(microtime(true) * 1000);//se creează un fișier ce //va avea ca nume un șir de cifre(timpul din momentul resetării)

	file_put_contents($currentTime, $data["newPassword"]."\n".$data["emailPasswordReset"]);

	$to = $data["emailPasswordReset"];

	$subject = "Test email";

	$txt = "Salut! \n Am observat ca vrei sa-ti schimbi parola. Daca doresti sa confirmi acest lucru, foloseste acest link: http://127.0.0.1/confirmare_parola.php?token=".$currentTime;
```

Mai sus sunt setate informațiile ce vor fi trimise prin email utilizatorului.

Se continuă(mai jos) cu un *SELECT* pentru a verifica dacă utilizatorul pentru care se vrea resetarea parolei există în baza de date, iar în caz de succes, se trimite email acestuia pentru resetare.

```
	$query = "SELECT idutilizator FROM utilizatori where email = '".$data["emailPasswordReset"]."'";

	$result=mysqli_query($db,$query);

	$row=mysqli_fetch_row($result);

	if($row[0] != ''){

		if(mail($to,$subject,$txt))

		//Mesaje în caz de succes sau eșec în trimiterea emailului
	}
?>
```

Pe scurt, scriptul php funcționează astfel: codul preia emailul introdus în secțiunea de email a procesului de resetare a parolei din aplicație și noua parolă dorită și verifică existența emailului în baza de date. În caz de succes, se trimite un *email* utilizatorului în care e introdus un *link* cu noua parolă și *emailul* specific contului. Prin accesarea linkului, noua parolă va fi introdusă în baza de date în locul celei vechi acolo unde se află o potrivire a emailului în urma clauzei *UPDATE.* În caz contrar, se trimite un mesaj specific situației(*email* invalid sau eroare la trimiterea *emailului* în cazul unor probleme tehnice).

În continuare, după accesarea emailului, utilizatorul își va putea reseta parola prin accesarea link-ului trimis(resetarea parolei o poate face doar utilizatorul în cauză, prin intermediul propriului email).

Acesta face uz de următoarea secvență de cod, care va actualiza parola în baza de date. Acel *UPDATE* menționat mai sus, se întâmplă în porțiunea aceasta.

//citirea fisierului *token* și preluarea datelor ce se vor actualizate

...

$query = "UPDATE utilizatori SET parola = '".$newPassword."' WHERE email = '".$email."'";

...

//apelul bazei de date cu respectiva cerință și afișarea unui mesaj în caz de succes sau eșec

Pentru partea de înregistrare, se crează o nouă fereastră, care preia responsabilitatea validării datelor și a creării unui cont, prin următorul cod:

```
        private void userRegistrationStartUpFormButton_Click(object sender,
EventArgs e)
        {
            accountInformations_["userType"] = "U";
            if (specialUserCheckBox.Checked == true)
            {
                accountInformations_["userType"] = "S";
            }
            userRegistration_ = new
RegisterForm(accountInformations_["userType"]);
            specialUserCheckBox.Checked = false;
            userRegistration_.Show();
        }
```

Înregistrarea se face sub forma unui cont de utilizator, utilizator caz special sau moderator.

5.2 RegisterForm

În urma alegerii tipului de cont, va fi creat o fereastră unde vor fi introduse unele date necesare persoanei care vrea să se înregistreze.

Pentru înregistrarea propriu-zisă, a fot scrisă următoarea parte:

```
private void registerButton_Click(object sender, EventArgs e)
        {
            int validEmail;
            validationData_ = new UserDataValidation();
            api_ = new BeeCharity.API();
```

```
            ...
//Se creează obiectele pentru validarea datelor și conexiunea cu serverul
//după care urmează o serie de verificări, pentru validarea datelor.
            ...

if (validEmail == 1)

            {
                accountInformations_["type"] = "register";
                accountInformations_["name"] = nameTextBox.Text + " " +
surnameTextBox.Text;
                accountInformations_["address"] = addressTextBox.Text;
                accountInformations_["phone"] = phoneTextBox.Text;
                accountInformations_["email"] = emailTextBox.Text;
                accountInformations_["password"] =
CreateMD5(passwordTextBox.Text);
                accountInformations_["passwordConfirmation"] =
CreateMD5(passwordConfirmationTextBox.Text);
                accountInformations_["data"] =
DateTime.Now.ToString("yyyyMMdd",
System.Globalization.CultureInfo.GetCultureInfo("en-US"));
                accountInformations_["organization"] =
organizationTextBox.Text;
                if (personalDataCheckBox.Checked == true)
                {
                    accountInformations_["personalData"] = "confidential";
                }
                else
                {
                    accountInformations_["personalData"] = "neconfidential";
                }
                if (accountInformations_["userType"] == "S")
                {
                    isSpecialUser_ = true;
                }
                response = api_.CallApi(accountInformations_);
```

După ce se verifică și formatul emailului să fie valid, se populează dicționarul *accountInformations*_cu datele necesare înregistrării și se apelează prin intermediul *api*_ serverul, pentru a insera noile date în baza de date.

```
if (response == "registerSuccess")
                {
                    if (isSpecialUser_ == true)
                    {
                        MessageBox.Show("S-a efectuat inregistrarea.\nPentru
a fi validat, contul dumneavoastra are nevoie de aprobare din partea unui
moderator.\nVerificati-va emailul periodic!");

                    }
                    else if (isModerator_ == true)
                    {
                        MessageBox.Show("S-a efectuat inregistrarea.\nPentru
a fi validat, contul dumneavoastra are nevoie de aprobare din partea unui
administrator.\nVerificati-va emailul periodic!");
                    }
```

```
                    else
                    {
                        MessageBox.Show("Inregistrare cu succes!");
                    }
                            ...
```

În cazul în care înregistrarea s-a efectuat cu succes, se va afișa un mesaj anume, în caz contrar se vor afișa alte tipuri de mesaje.

5.3 MenuForm

Partea de meniu e putin mai complicată, deoarece prezintă o funcționalitate mai mare.

Prima parte, și anume cea de afișare a produselor a fost implementată după cum urmează:

```
private void displayItemsOnReq(string typeOfReq)
        {
           ...
      //Declararea și inițializarea unor variabile/obiecte/câmpuri
          response = api_.CallApi(accountInformations_);
          //apelul bazei de date pentru afișarea informațiilor
            try
            {
                var elements =
JsonConvert.DeserializeObject<List<ItemsModel>>(response);

                if (typeOfReq == "itemClaimedHistory")
            //afișarea istoricului cererilor
                {
                    string[] colstring  = { "Utilizator", "Categorie produs",
"Denumire produs", "Data cerere" };//setarea numelui coloanelor
                    Array.Copy(colstring, 0, listviewColumnsHeader, 0,
colstring.Length);

                    foreach (var el in elements)
                    {
                        itemToShow_ = new ListViewItem(el.nume);
                        itemToShow_.SubItems.Add(el.categorieprodus);
                        itemToShow_.SubItems.Add(el.denumireprodus);
                        itemToShow_.SubItems.Add(el.dataplasarecerere);
                        itemToShow_.SubItems.Add(el.idutilizator);
                        itemToShow_.SubItems.Add(el.idprodus);

itemToShow_.SubItems.Add(accountInformations_["userLoggedInId"]);
                        accountInformations_["userDisplayInfo"] =
el.idutilizator;
                        historyModeratorModeOn_ = true;
                        itemsDisplayListView.Items.Add(itemToShow_);
                    }//Populatrea coloanelor controlului ListView cu date.
```

În mod asemănător se setează coloanele și valorile acestora pentru afișarea produselor postate de alți utilizatori, propriul istoric al postărilor sau/afișarea conturilor invalide(utilizator caz special).

```
                ...
                //Afișarea titlului coloanelor controlului ListView
                foreach (ColumnHeader lstViewCol in
itemsDisplayListView.Columns)
            {
                lstViewCol.Text = listviewColumnsHeader[colIndex];
                colIndex++;
            }

        //Setarea vizibilității panoului cu produse.
            itemsDisplayPanel.Visible = true;
            loadItemsPanel.Visible = false;
        }
              //Tratare de erori și mesaje specifice
              ...
```

Partea de upload a unui produs e implementată după cum urmează:

```
private void uploadProductButton_Click(object sender, EventArgs e)
        {
            accountInformations_["type"] = "productUpload";

            ...
            //Verificări ale informațiilor introduse de utilizator.
            ...

            string response = api_.CallApi(accountInformations_);

            if (response == "uploadSuccess")
            {
                setControlsEmpty();

                api_.uploadImage(uploadPhotoPictureBox.Text);

                  ...
                  //Afișarea unor mesaje
                  ...
              }
```

Pentru validarea datelor ce urmează a fi încărcate, s-au folosit metode pentru validare.

Exemple:

```
private bool itemQuantityValidation()
        {
            if (quantityNumericUpDown.Value == 0)
            {
                return false;
            }
            accountInformations_["itemQuantity"] =
quantityNumericUpDown.Value.ToString();
```

```
            return true;

        }
private bool itemDescriptionValidation()
        {
            if (productDescriptionTextBox.Text == "")
            {
                return false;
            }
            accountInformations_["itemDescription"] =
productDescriptionTextBox.Text;
            return true;

        }
```

Prima metodă validează cantitatea ce se vrea încărcată, iar cea de-a doua descrierea.

Pentru încărcarea imaginii local, s-a folosit următoarea metodă:

```
private void addProductPhoto_Click(object sender, EventArgs e)
        {
            try
            {
                OpenFileDialog dialog = new OpenFileDialog();
                dialog.Filter = "Image Files(*.BMP; *.JPG; *.PNG)|*.BMP;
*.JPG; *.PNG" + "|All files(*.*)|*.*";
                dialog.CheckFileExists = true;
                dialog.Multiselect = false;

                if (dialog.ShowDialog() == DialogResult.OK)
                {
                    uploadimage_ = Image.FromFile(dialog.FileName);
                    uploadPhotoPictureBox.Image = uploadimage_;
                    uploadPhotoPictureBox.Text = dialog.FileName;
                    itemImageAdded_ = true;

                    Bitmap bm = new Bitmap(uploadPhotoPictureBox.Text);
                    fillPictureBox(uploadPhotoPictureBox, bm);
                }
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.ToString());
                MessageBox.Show("A aparut o eroare", "Eroare",
MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }
```

În timp ce pentru încărcarea acesteia în baza de date s-a folosit metoda uploadImage din clasa API.

```
public void uploadImage(string imagePath)
        {
            System.Net.WebClient Client = new System.Net.WebClient();
            Client.Headers.Add("Content-Type", "binary/octet-stream");
```

```
            byte[] result =
Client.UploadFile("http://127.0.0.1/upload_photo.php", "POST", imagePath);
        }
```

Procesul de încărcare a pozei se face prin încărcarea acesteia la nivel local, într-un control denumit *PictureBox*, pe urmă se apelează metoda descrisă mai sus cu numele complet(*path*-ul) pozei pentru a fi încarcată pe *server*, iar mai apoi, se face încărcarea în baza de date,a numelui pozei. Aceasta se regăsește pe server cu numele aferent id-ului utilizatorului din baza de date, care e unic, astfel că fiecare poză are un nume unic după care este căutată pe server. Adresa unde sunt salvate pozele este: *http://localhost/images/*.

Butonul de căutare a unui produs are codul *PHP* aferent:

```
<?php

//conexiunea cu baza de data

$myArray = array();

      $searchWord = $data["searchDetail"];

      $queryUser = "SELECT u.idutilizator, u.nume,p.idprodus,
p.denumireprodus, p.categorieprodus, p.dataincarcareprodus FROM produse p
INNER JOIN  utilizatori u ON u.idutilizator = p.idutilizator AND
confirmareutilizator = 1 AND u.email <> '".$data["username"]."' AND
p.disponibilitate = 1 AND u.nume LIKE '$searchWord%'";

      $queryItem = "SELECT u.idutilizator, u.nume,p.idprodus,
p.denumireprodus, p.categorieprodus, p.dataincarcareprodus FROM produse p
INNER JOIN  utilizatori u ON u.idutilizator = p.idutilizator AND
confirmareutilizator = 1  AND u.email <> '".$data["username"]."' AND
p.disponibilitate = 1 AND p.denumireprodus LIKE '$searchWord%'";
```

Sunt prezene două *query-uri în SQL ce vor fi trimise bazei de date, pentru căutarea produsului după nume sau dacă acesta conține un subșir de caractere precum celor introduse în câmpul pentru căutare. În caz de nepotrivire, se caută în același mod după un utilizator.*

```
      $results = mysqli_query($db, $queryItem);
      if(mysqli_num_rows($results) > 0)
      {

            while($row = $results->fetch_array(MYSQLI_ASSOC))
            {
                  $myArray[] = $row;
            }
      }
      else
      {
            $results = mysqli_query($db, $queryUser);
            if(mysqli_num_rows($results) > 0)
            {
                  while($row = $results->fetch_array(MYSQLI_ASSOC))
            {
                  $myArray[] = $row;
```

```
            }
            }
        }
        echo json_encode($myArray);
```

Se procesează căutarea. Dacă nu s-a găsit nimic conform primei căutări(după un produs) sau celei de a doua(după utilizator), aplicației i se returnează practic o listă goală de caractere, interpretată ca *"[ ]"*. Conform acesteia, se va afișa un mesaj pentru a înștiința utilizatorul că nu s-a găsit nimic în baza de date, în urma celor introduse pentru căutare.

Tot în partea de meniu mai există opțiunea de afișare a organizațiilor prezente/adăugate în baza de date și a informațiilor acestora. Astfel, în caz de donație a unei sume de bani, datele pot fi luate din lista meniului(*IBAN* și *Numele organizatiei*). Partea de *display* nu o să mai fie adăugată căci s-a arătat în prima parte un exemplu.

În cazul unui moderator, acesta are acces și la o listă a unor conturi ce nu au fost validate. Acesta poate să valideze un cont la cererea cuiva. Pentru a nu mă lungi, o să fac abstracție și de partea aceea de cod.

5.4 DisplayItemForm

Pentru a afișa un produs selectat din lista afișată în meniu, se creează un obiect de tipul clasei DisplayItemForm. Codul aferent utilizat pentru afișarea datelor produsului selectat, e următorul:

```
private void displayItems()
        {
            string userTypeDisplayed = "";
            string response = "";
            informations_["type"] = "itemAllDataAbout";
            api_ = new BeeCharity.API();
            response = api_.CallApi(informations_);
```

Partea de mai sus realizează conexiunea cu baza de date și primirea unui răspuns. Acesta constă deținerea unor date despre produsul dorit.

```
            try
            {
                if (response != null)
                {
                    var elements =
JsonConvert.DeserializeObject<List<ItemsModel>>(response);

                    foreach (var el in elements)
                    {
                        organizationLabel.Text = el.institutiereprezentata;
                        usernameTextBox.Text = el.nume;
                        itemCategoryTextBox.Text = el.categorieprodus;
                        itemNameTextBox.Text = el.denumireprodus;
```

```
                        itemUploadDateTextBox.Text = el.dataincarcareprodus;
                        itemDetailsTextBox.Text = el.detaliiprodus;
                        userTypeDisplayed = el.tiputilizator;

                        itemImagePictureBox.Load("http://127.0.0.1/images/" +
informations_["itemId"] + ".jpg");

                                    ...
```

Ceea ce se realizează în bucata de cod de mai sus e popularea controalelor cu informații despre bunul care se vrea a fi văzut în detaliu. Metoda continuă cu gestiunea unor mesaje, pentru a le afișa utilizatorului în situații problematice.

Pentru butonul de cerere a produsului, adică atunci când se “comandă” acesta, s-a implementat următorul cod:

```
private void claimItemButton_Click(object sender, EventArgs e)
        {
            string response = "";
            api_ = new BeeCharity.API();
            informations_["type"] = "claimItem";
            informations_["claimData"] = DateTime.Now.ToString("yyyyMMdd",
System.Globalization.CultureInfo.GetCultureInfo("en-US"));
            response = api_.CallApi(informations_);
            if (response == "successRequest")
            {
                MessageBox.Show("Cererea a fost procesata.");
                this.Close();
            }
            else if (response == "failedRequest")
            {
                MessageBox.Show("Nu s-a putut procesa cererea. Au aparut ceva
probleme!");
            }
            else if(response == "unavailableProduct")
            {
                MessageBox.Show("Produsul selectat a fost deja luat.");
                this.Close();
            }
            else
            {
                MessageBox.Show("Probleme tehnice. Contactati
administratorul!");
            }
        }
```

Se poate vedea un exemplu complet, cum s-a gestionat situația, cu mesaje care tratează situațiile speciale.

Partea de *PHP* care gestionează datele din tabele, mai exact cel al comenzilor se poate observa mai jos:

```
$queryVerify = "SELECT idprodus FROM produse WHERE idprodus = $itemId AND
disponibilitate <> 0";
```

Primul *query* verifică dacă produsul selectat e disponibil. În mod normal, produsele care au fost deja comandate nu vor mai fi afișate în secțiunea produselor disponibile, ba chiar nu vor putea

fi văzute deloc în afară de cei care le-au comandat sau postat. Totuși, pentru a evita orice problemă, am ales să adaug condiția de disponibilitate în respectivul *request*.

```
      $results = mysqli_query($db, $queryVerify);


      if(mysqli_num_rows($results) > 0)
      {
            $query = "INSERT INTO comenzi (iddetinatorcomanda, idprodus,
idutilizator, dataplasarecerere) VALUES ($useronwcommandid, $itemId,
$userdisplayinfoaboutid, '".$data["claimData"]."'); UPDATE produse SET
disponibilitate = 0 WHERE idprodus = $itemId";

                  if($db->multi_query($query))
```

Partea de mai sus face ca în urma verificării produsului în baza de date, să se creeze un alt *request* care să populeze tabela comenzi cu datele necesare unei comenzi. Dacă totul funcționează normal, comanda va fi plasată cu succes, altfel, se vor afișa mesaje de gestiune a problemei apărute.

5.5 UserInfoForm

Procedura prin care sunt afișate informațiile despre un anumit utilizator, este asemănătoare cu cea a afișării datelor unui produs. Ceea ce diferă sunt informațiile luate din tabele.

De aceea, aici o să evit adăugarea de informații în plus, deoarece nu prezintă ceva prea diferit.

5.6 DisplayItemPhotoForm

Clasa aceasta face doar să afișeze poza produsului selectat pentru afișare în clasa *DisplayItemForm,* la dimensiuni mai mari.

```
private void getPhoto(string photoId)
        {
            this.itemPhotoPictureBox.SizeMode = PictureBoxSizeMode.Zoom;
            itemPhotoPictureBox.Load("http://127.0.0.1/images/" + photoId +
".jpg");
        }
```

Metoda preia de pe server imaginea prezentă în *DisplayItemForm*(conform unui anume produs) și o afișează în modul *Zoom,* pentru o vedere mai clară a acesteia.

5.7 API

Clasa *API* e cea care gestionează conectarea cu *serverul*, trimiterea de cereri și răspunsuri înspre și de la acesta.

Metoda prin care se face acest lucru este următoarea:

```
private string databaseReq(string phpFile, Dictionary<string, string> values)
{
            WebRequest request = WebRequest.Create("http://127.0.0.1/" +
phpFile);
//Adresa serverului + fisierul apelat în urma cererii.

            request.Method = "POST";

            byte[] json =
Encoding.UTF8.GetBytes(JsonConvert.SerializeObject(values));
            request.ContentType = "application/x-www-form-urlencoded";
            request.ContentLength = json.Length;
```

De partea cu *POST* se poate face abstracție deoarece nu se folosește baza de date direct din aplicație, ci prin server. Pe urmă se realizează conversia datelor cererii într-un fișier json, deoarece urmează folosită partea de *PHP* și pentru ca datele să poată fi citire și interpretate corect. Practic, json s-a folosit pentru a se face cu ușurință translatarea informațiilor de la un limbaj la altul.

```
            using (Stream dataStream = request.GetRequestStream())
            {
                dataStream.Write(json, 0, json.Length);
                using (WebResponse response = request.GetResponse())
                {
                    using (Stream responseStream =
response.GetResponseStream())
                    {
                        using (StreamReader reader = new
StreamReader(responseStream))
                        {
                            string responseFromServer = reader.ReadToEnd();
                            if (responseFromServer != "Failed to connect to
database!")
                                return responseFromServer;
                            else return "";
                        }
                    }
                }
            }
        }
```

Realizarea conexiunii cu serverul, trimiterea și gestionarea răspunsului acestuia s-a făcut după cum se poate vedea mai sus prin diferite funcții și obiecte gata definite în librăria de care dispune C#.

Metoda primește un obiect de tip dicționar ce conține anumite date, necesare unei anume cereri, ce se va trimite serverului și pe baza căruia va fi întors un răspuns.

5.8 UserDataValidation

Clasa *UserDataValidation*(folosită in special la partea de înregistrare) conține o serie de metode, unele ce folosesc *regex*-uri pentru validarea datelor unui utilizator.

6 Utilizare

În urma apelării funcției *main()* din clasa *Program*, aplicația se va afla în modul *run* unde va fi creat un obiect de tipul *StartupForm*.

În exemplul de mai jos, va fi afișat meniul pentru un cont tip *moderator* deoarece prezintă funcționalități mai multe.

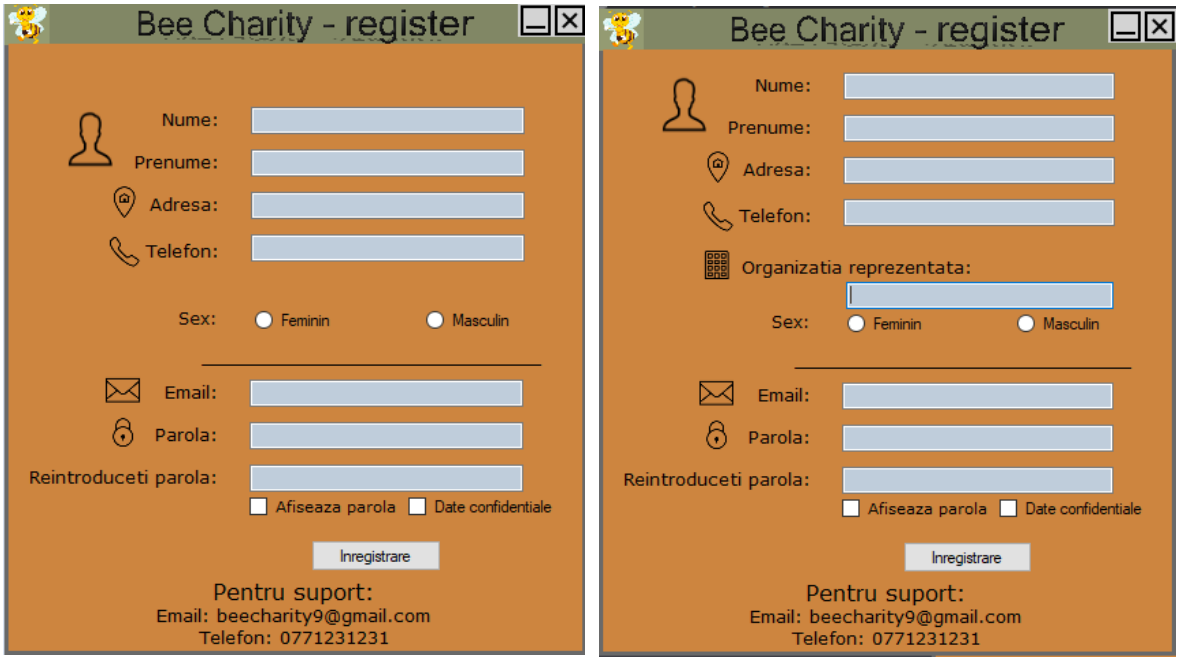
Rezultatul e următorul:

Prima parte a utilizării aplicației constă în folosirea funcționalităților precum: crearea unui cont(obligatorie pentru a putea utiliza aplicația) în diferite roluri, autentificare sau resetarea parolei.

Tipul contului ce se vrea creat, poate fi ales din următoarele variante: mod utilizator, mod moderator sau mod utilizator caz special(în urma bifării căsuței din dreptul textului „Utilizator caz special” și apăsarea butonului pe care apare textul „Utilizator”).

Detaliile pentru tipurile conturilor pot fi regăsite în fișierul Readme.txt anexat aplicației.

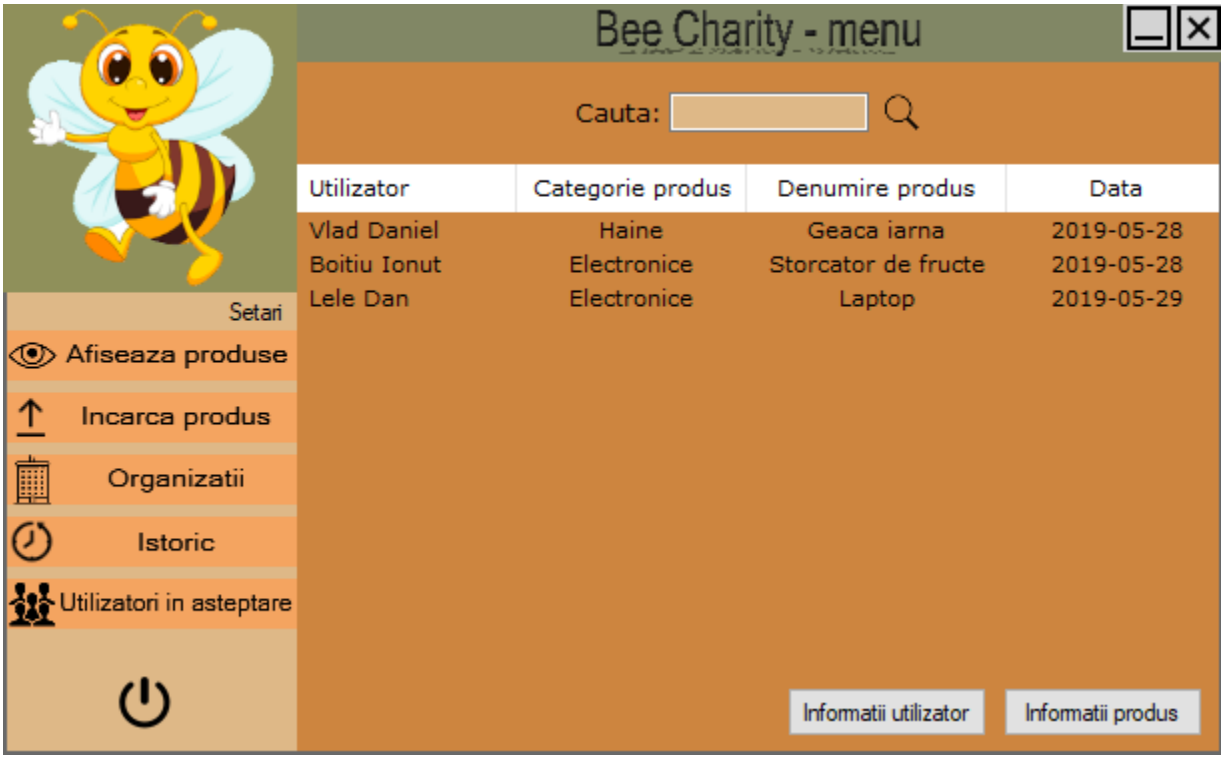
Rezultatul pentru pasul de înregistrare:



În stânga este fereastra pentru înregistrarea unui utilizator normal sau într-o situație specială(defaforizată), pe când în dreapta este fereastra pentru înregistrarea unui moderator. Acesta din urmă va trebui să specifice în plus organizația pe care o reprezintă.

Dacă se dorește autentificarea unui utilizator/moderator, ce s-a obținut în urma autentificării pentru un utilizator/moderator existent este următorul meniu:

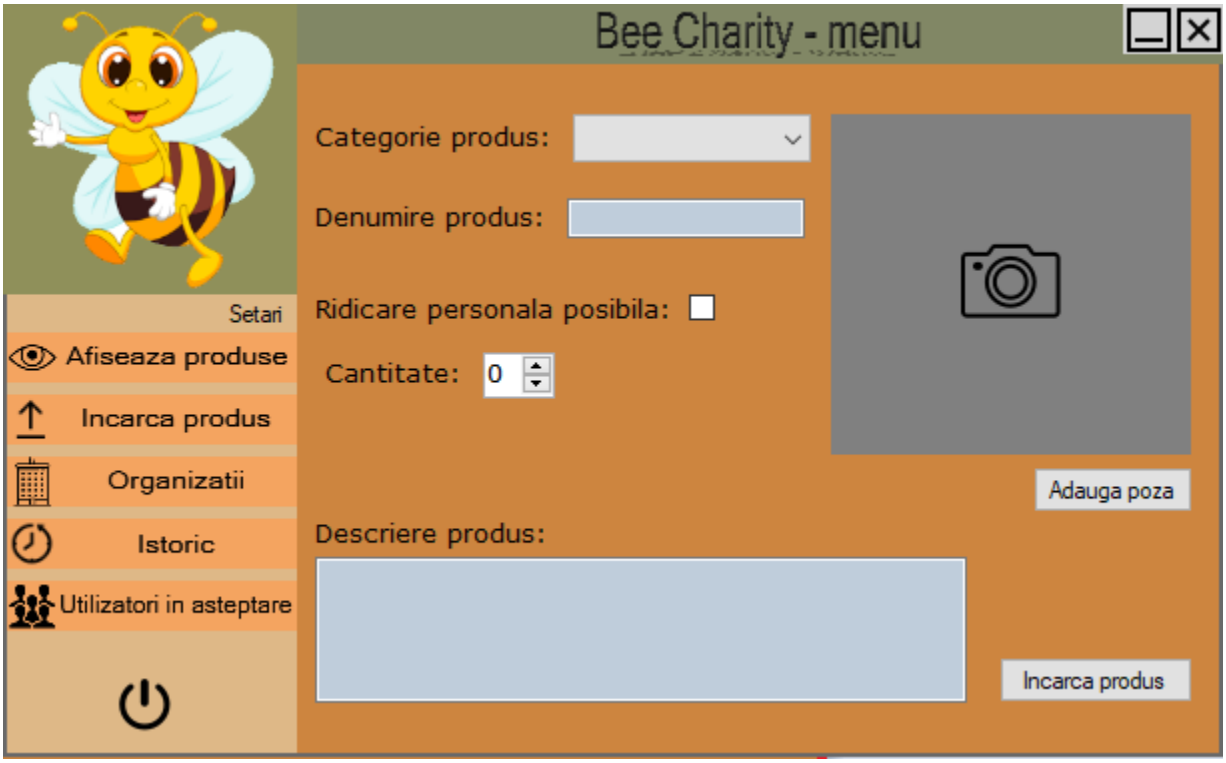
- Pentru afișarea listei produselor



Aici sunt prezente două butoane care pot oferi informații mai în detaliu despre un anume produs sau despre un utilizator înregistrat. Mai există și partea de căutare, ce este poziționată în partea de sus a meniului. Astfel, un moderator poate vizualiza mai îndeaproape un anume produs și îl poate însuși în urma apăsării butonului *Informatii produs* unde va fi afișat produsul detaliat. În urma afișării detaliilor bunului respectiv, se poate da *click* pe butonul *Doresc produs.*

Pentru testare, a fost populată baza de date cu informații fictive.

- Pentru încărcarea unui produs:



Sunt prezente aici o serie de diferite controale care preiau date despre ceea ce se vrea încărcat.

Primul conține diferite categorii după care un bun poate fi clasificat, urmat de un control ce preia numele bunului respectiv.

Mai e prezent și un *checkBox* pentru a specifica dacă utilizatorul e de acord ca produsul să fie preluat personal de acasă de la el. În cazul în care acesta nu e bifat, utilizatorul și moderatorul vor trebui să discute pentru a găsi o metodă de ridicare a bunului.

Controlul pentru cantitate redă numărul de produse ce sunt puse pe lista de posibile donații.

E prezent și un control pentru poza produsului, încărcarea neputând fi validată dacă detaliile legate de produsul în cauză nu sunt însoțite de o poză a acestuia.

- Pentru afișarea listei organizațiilor:

În această secțiune, sunt afișate datele unor organizații care au fost înregistrate în sistem.

De menționat este că adăugarea acestora, se face de către administrator, partea de adăugare nefiind validă pentru niciun tip de utilizator.

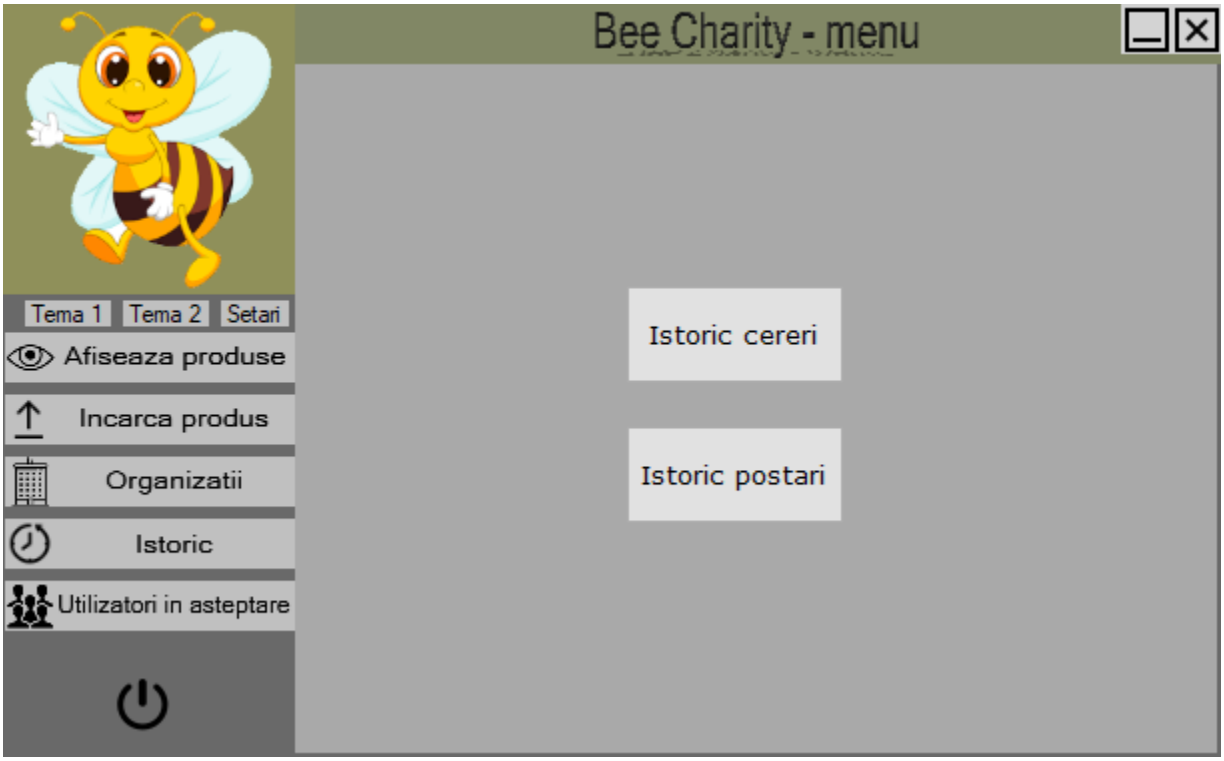
Explicația acestei alegerii e că numărul de ONG-uri nu este unul foarte mare, astfel că dacă o organizație dorește să beneficieze de sprijinul nostru, resprezentantul acesteia va trebui să i-a legătura cu administratorul(poate trimite email pe adresa specificată la suport, pe primul *Form*). Acesta le va da dreptul(adăuga în baza de date) de a face parte din lista organizațiilor prezente care să dispună de ajutor din partea *Bee Charity*.

În urma adăugării organizației în lista, aceasta va putea fi vizualizată de orice utilizator, iar în cazul unei donații, utilizatorul poate alege o organizație din lista respectivă.

În acest mod funcționează partea de ajutor al organizațiilor care sunt prezente în baza de date.

Celălalt tip de ajutor(donarea bunurilor), fiind acordat prin intermediul reprezentanților organizațiilor.

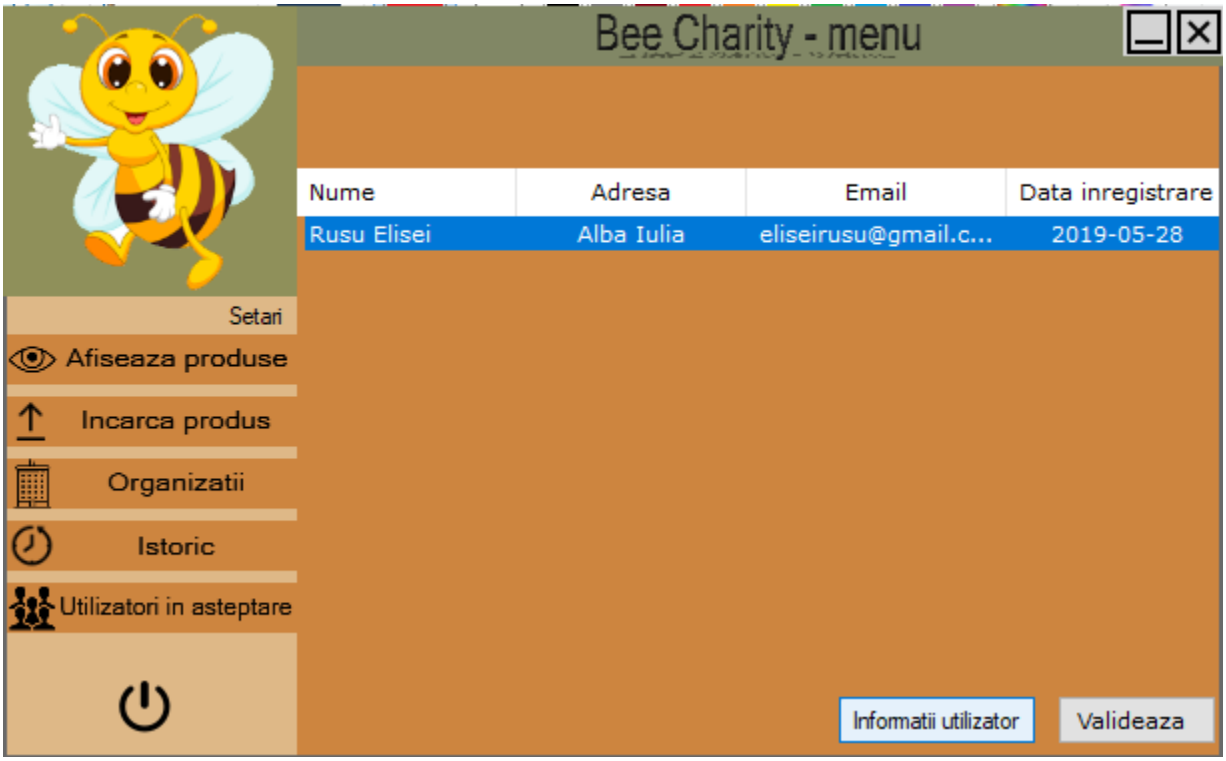
- Pentru afișarea istoricului:



Această parte e afișată pentru un moderator care poate să aleagă să vadă lista cu produse postate sau cerute. Pentru un utilizator, istoricul e afișat direct, ca lista afișată în urma apăsării butonului de *Afiseaza produse*.

A fost introdus o funcționalitate de schimbare a temei, astfel că utilizatorul poate comuta pe un alt mod vizual, în ceea ce priveste culoarea fundalului.

- Pentru afișarea listei de utilizatori speciali:



Aici vor fi afișate conturile utilizatorilor care sunt în așteptarea validării. O persoană care se înregistrează cu un astfel de cont, ar trebui să discute înainte cu un moderator sau cu administratorul aplicației, pentru ca ulterior, contul să îi fie validat.

Mai sunt prezente și două butoane, unul ce afișează informații despre utilizatorul selectat și altul care validează contul acestuia.

Butonul de setări prezent oferă posibilitatea schimbării culorilor aplicației, pentru personalizare.

În partea de afișare a produselor, după ce s-a selectat un produs din lista respectivă, se pot afișa informații despre acesta sau chiar despre utilizator.

În urma selectării de afișare a informațiilor despre un produs, este creeată fereastra următoare:

Dacă se dorește vizualizarea pozei produsului în dimensiuni mai mari, se va apăsa *click* pe imagine.

Rezultatul va fi:

Pentru afișarea datelor unui utilizator, se va apăsa butonul "*Informatii utilizator*" fie din fereastra unde sunt afișate informații despre produs, fie direct din meniu.

Rezultatul obținut va fi următorul:

Această fereastră oferă informații detaliate despre un anumit utilizator.

În urma apăsării butonului de contact, persoana logată va fi direcționată înspre email urmând ca aceasta să compună un mesaj și să îl expedieze destinatarului, iar persoanei care se vrea a fi contactată, îi va fi trimis un email de înștiințare cu emailul utilizatorului care o caută și contextual în care o caută.

De menționat e faptul că dacă în momentul înregistrării s-a ales ca datele să fie confidențiale, un utilizator normal nu va putea vedea datele celui care s-a înregistrat cu această opțiune.

Un moderator totuși, va avea acces la informațiile acelei persoane fără restricție.

Datele(informațiile) ce s-au putut vedea în acest capitol sunt fictive.

7 Concluzii și direcții de continuare a dezvoltării

7.1 Licență aplicație

În cazul primirii licenței, aplicația ar putea fi utilizată cu succes în urma adăugării unei metode de distribuție a bunurilor donate. De exemplu un moderator are comandate mai multe bunuri de la proprietari diferiți, astfel că ar putea fi pusă la dispoziție din partea primăriei, o locație unde lumea poate să depună bunurile respective și moderatorul să le ridice într-un anumit interval de timp, după ce bunurile au fost plasate.

O altă modalitate ar fi crearea unui serviciu de curierat. Astfel, o dată la două saptămâni, o persoană angajată de primărie să se ocupe cu gestiunea comenzilor.

7.2 Migrarea aplicației pe web

Cu siguranță credibilitatea unei aplicații web este mai mare. Prin urmare, aplicația ar putea fi dezvoltată pentru partea de web și ar fi mai ușor de folosit. În prezent, una dintre cele mai folosite tipuri de aplicații sunt cele web. Astfel că, rata succesului ar fi mai ridicată.

7.3 Dezvoltarea aplicației pe mobil

Conform unui studiu făcut în 2016, jumătate din populație, mai exact 53% folosește *smartphone-ul.* Încă începând cu anul 2013, datele au fost în continuă creștere și cu siguranță numărul a crescut considerabil până în 2019. Statistica realizată confirmă faptul că numărul de aplicații pentru calculator a luat-o descendent, în timp ce numărul aplicațiilor pentru smartphone e în continuă creștere.

Conform acestor date statistice, aplicația ar putea fi îndreptată înspre mediul mobil.

Un alt argument ar fi că numărul persoanelor cu acces la un laptop/calculator este mult mai mic decât numărul celor care au acces la un telefon mobil.

Astfel, o viitoare îmbunătățire ar fi trecerea aplicației pe mobil, adăugându-se și un punct de feed-back, din partea utilizatorilor, pentru a putea implementa idei noi și pentru a dezvolta aplicația.

Un serviciu oferit în plus, aici în toate cele trei variante, este cel de mentenanță, pentru a avea o funcționare cât mai bună și pentru a satisface clienții.

Bibliografie și referințe

1. https://ro.wikipedia.org/wiki/Organiza%C8%9Bie_neguvernamental%C4%83
2. https://bettercarenetwork.org/sites/default/files/attachments/Child%20Abandonment%20and%20Its%20Prevention%20Summary%20Romanian%20Language%20-%20Romania.pdf

   http://srmagazine.ro/romania-2017-la-fiecare-9-ore-un-bebelus-e-parasit-sectiile-de-maternitate/

3. https://docs.microsoft.com/en-us/dotnet/framework/winforms/
4. https://www.tiobe.com/tiobe-index/
5. https://www.code-it.ro/tutoriale-c-prezentarea-limbajului/
6. https://www.php.net/manual/ro/intro-whatcando.php
7. https://csus-dspace.calstate.edu/bitstream/handle/10211.3/181760/2016DeshpandePurva.pdf?sequence=3
8. https://www.w3schools.com/sql/sql_intro.asp
9. http://ase.softmentor.ro/POO/IntroducereWindowsForms.pdf
10. https://ro.wikipedia.org/wiki/Gmail
11. https://www.librapay.ro/serviciu_procesare_plati_online/
12. https://visualstudio.microsoft.com/vs/
13. http://www.runceanu.ro/adrian/wp-content/cursuri/tw2014/Laborator1(2014).pdf
14. http://www.timisoreni.ro/info/ong_i_organizaii/
15. C# pentru liceu - Programare în Visual C# 2008 Express Edition , autori: Constantin Gălăţan, Susana Gălăţan
16. Introducere în Programarea .Net Framework, autori: Mihai Tătăran, Petru Jucovschi, Tudor – Ioan Salomie

Referințele prezente au fost accesate în intervalul: 10.02.2019 – 07.06.2019