Universitatea Babeș-Bolyai Cluj-Napoca
Facultatea de Matematica-Informatica

PROIECT

INGINERIA SISTEMELOR SOFT

Profesor îndrumător: Zsigmond Imre

Echipa "The Superficial Deers":

Moldovan Andrei – team leader

Moldovan Daniel

Munteanu Ciprian

Nan Tudor

Oancea Alin

Olaru-Copciuc Iustin-Cristian

Pașcalău Loredana

Semigrupa 225/2

Prezentarea cerinței aplicației

Problema principala pe care trebuie sa o rezolve aplicatia este cea a administrarii intregului sistem de donare de sange: de la cetateanul care vrea sa doneze pana la pacientul care are nevoie de sange.

Tot acest sistem are la baza traseul pungii de sange.

In prima faza, aplicatia trebuie sa permita oricarui individ sa isi creeze cont pentru a putea dona sange. Astfel, prin creearea contului si prin completarea unor informatii personale, acesta poate trece pe la un centru pentru a dona. Aplicatia trebuie sa fie conceputa in asa fel incat sa instiintieze donatorii de conditiile de donare, sa sprijine utilizatorii ei sa doneze si alta data si sa puna la dispozitie donatorilor analizele de sange ale acestora.

Dupa momentul in care un utilizator isi exprima dorinta de a dona si se prezinta la un centru de donare/transfuzie intervine al doilea tip de utilizator al aplicatiei: personalul centrului de transfuzie. Acesta are un rol important de a administra pungile de sange. Prin atributiile personalului centrului de transfuzie se numara: administrarea donatorilor, administrarea pungilor de sange (analize de sange care ajung si la donator), administrarea cererilor de sange primite de la medici.

Un actor important al aplicatiei este medicul. Acesta poate sa trimita cereri de sange pentru pacienti si sa vizualizeze stadiul acestor cereri. Tot el face administrarea pacientilor din spital.

Aplicatia trebuie sa mai permita: comunicarea intre medici si personalul centrului de transfuzie si notificarea donatorilor cand e nevoie de sange.

Funcționalitați

Functional Requirements:

- Aplicatia contine 3 tipuri de utilizatori : donator , medic si personalul de la centrul de transfuzie. In functie de tipul de utilizator care este logat functionalitatile acestuia sunt specifice tipului contului.
- In functie de tipul de utilizator, la inregistrare, ii sunt cerute diferite date specifice pentru a se valida tipul contului. Spre exemplu, conturile medicilor si a persoanelor de la centrul de transfuzie au un camp in plus, care reprezinta o licenta primita de un administrator de retea. Contul de donator nu are nevoie de aceasta licenta.
- Toti utilizatorii se pot loga cu contul sau cu adresa de email si parola daca in prealabil s-au inregistrat
- Exista o comunicare intre medici si personalul centru transfuzie prin diferite metode printre care se numara emailul si chatul
- DONATORI - din momentul in care utilizatorul s-a logat cu un cont de donator acesta poate realiza urmatoarele actiuni:
  - Vizualizare informatii despre cerintele pentru donare
    - Inainte de completarea chestionarului pentru donare ii sunt aduse la cunostinta donatorului cateva informatii precum : conditiile de adminisibilitate pentru donare(condtiile pe care donatorul trebuie sa le indeplineasca pentru a se realiza donarea), bolile de care a suferit sau sufera si ar putea anula donarea
  - Completare chestionar pentru donare
    - Reprezinta un completarea unui chestionar in care donatorul trebuie sa completeze cateva date personale, dar si informatii utile(sunt optionale) si isi poate alege o zi in care ar dori sa doneze
  - Vizualizare rezultate analize
    - In urma analizelor donatorul isi poate accesa contul pentru a vedea rezultatele analizelor, daca acestea au fost introduse in sistem de catre personalul avizat
  - Cand poate dona urmatoarea data

        - De asemenea donatorul poate vedea data in care acesta mai poate dona
    - Primeste notificari cand e nevoie de sange
        - Donatorul este notificat cand este nevoie de sange, aceasta notificare realizandu-se de catre personalul specializat
- MEDICII
    - Administrarea pacientilor din spitalul de care apartine
        - Un medic poate adauga pacienti
    - Trimitere cereri de sange pentru pacienti
        - Un medic poate sa creeze o cerere de sange, sa trimita o cerere de sange sau sa o anuleze
    - Vizualizarea statusului cererilor
        - In orice moment medicul poate sa vada in ce stare se afla cererile deja inregistrate
- PERSONALUL CENTRULUI DE TRANSFUZIE
    - Administrare donator si cereri donatori
        - Personalul de la centrul de transfuzie poate adauga donatori, pacienti, accepta o cerere sau sa o anuleze,
    - Recolteaza sange si completeaza formular
        - Completeaza un formular cu toate informatiile necesare pentru a se putea realiza o donare
    - Administrare pungi de sange
        - Stafful poate adauga pungi de sange, trimite pungi de sange sau sa le anuleze valabilitatea din diferite considerente
    - Vizualizare stoc current
        - In orice moment personalul de la centrul de transfuzii poate vedea stocul curent de sange

Non-Functional Requirements

- Fara delay la comunicarea cu baza de date
- Minimum requiremnets pentru install aplicatie:
  - 2GB ram, 500mb HDD
- Informatii generale despre aplicatie
- Documentare
- Backup baza de date
- Portabilitate: windows, linux, mac
- Stabilitate: backend server 99.5%
- Securitate
- Tehnologii
- Orm

Etapele proiectării

- Analiza cerintei

  - Initial ne-am intalnit, toata echipa, de cateva ori pentru a discuta si clarifica eventualelele nelamuriri in legatura cu cerinta primita. Dupa ce am citit de enuntul problemei si am analizat fiecare detaliu am facut o lista cu intrebari / clarificari. Ulterior am intrebat profesorul de curs, dar si pe cel de la seminar si astfel ne-am asigurat ca am inteles cerinta bine si am putea face o strategie de rezolvare a problemei.

- Strategia de rezolvare

  - Dupa cunoasterea in detaliu a cerintei, am reusit sa venim cu cateva idei de inceput. Au fost mici conflicte deoarece, echipa fiind formata din 7 oameni, parerile au fost impartite dar intr-un final am ajuns la un rezultat comun, ajutati in special si de tehnical leader-ul nostru(Moldovan Daniel) care a reusit sa asculte fiecare parere si sa aleaga daca propunerile facute de noi sunt bune sau nu. De asemenea, in etapa de alegere a strategiei de rezolvare am ales si tehnologiile ce ne vor ajuta in rezolvarea problemei.

- Probleme intampinate pe parcurs
    - Putem considera ca o prima problema intampinata a reprezentat-o intelegerea diferita a cerintei si a trebuit sa clarificam , ca echipa, acest lucru
    - O alta problema ar putea fi lucrul in echipa, deoarece poate nu toti am mai facut asta si uneori faceam aceleasi task-uri din lipsa de comunicare in echipa, aceasta problema a fost rezolvata prin mai multe intalnire si mai multe discutii si intrebari despre parcursul proiectului
    - Respectarea diagramelor, o alta problema intampinata de echipa a fost aceea ca in cazul cerintelor in care se cerea ca codul sa fie generat din diagrama noi am facut invers, problema fiind rezolvata ulterior prin generarea codului din diagrame
    - O alta problema intampinata a fost folosirea unui sistem de versionare, si anume git-ul, deoarece initial am vrut sa folosim in singur branch, ulterior alegand ca fiecare task / persoana sa aiba un singur branch si master-ul sa fie mereu functional
    - Deoarece nu toti membrii echipei au fost familia cu sistemul de aisgnare a task-urilor, initial nu am asignat task-urile sau daca le rezolvam uitam sa le logam pe trello, o data cu evolutia proiectului acest lucru a disparut
- Cerinte functionale si non functionale, Use cases, Design diagrama de clase um, Design baza de date
- Implementare GUI
- Implementare functionalitati

Metodologia de testare

1. Unit test pentru comunicarea cu serverul de pe backend:
    a. Junit in java
    b. Trimitere request dummy spre serverul http flask
    c. Serverul raspunde cu un raspuns identic cu request-ul Comunicarea functioneaza
2. Manual Testing:
    a. Gui Testing
    b. Rapid software testing
    c. Validari + Corner cases
    d. Completare scenario de testare

TEHNOLOGII UTILIZATE ÎN CADRUL PROIECTULUI

Tehnologii utilizate la nivel de server

1.ORM - *SQLAlchemy*

Object / Relational Mapping (ORM) este o tehnica de programare ce face posibila accesarea si manipularea obiectelor fara ca programatorii sa fie interesati de sursa de date de unde provin aceste obiecte. Aceasta tehnica a aparut din nevoia de a depasi diferentele de paradigma dintre modelul orientat pe obiecte (sustinut de limbajele de programare de nivel inalt actuale) si modelul relational (utilizat de cele mai populare sisteme de gestiune a bazelor de date).

SQLAlchemy este un toolkit SQL si un ORM (object-relational mapper) pentru limbajul de programare Python. Am ales acest ORM datorită sintaxei simple și eficienței crescute.

Am folosit ORM efectiv in programul nostru prin maparea initiala a tabelelor prin definirea numele tabelei din baza de date, a fiecarei coloane si atributelor specifice acesteia din baza de date si a unei relatii(optional) prin care se definesc foreign key-urile in obiectele noastre.

De asemenea am definit operatiile CRUD pe obiectele din baza de date(insert, update, select, delete).

Exemplu creare entitate analize:

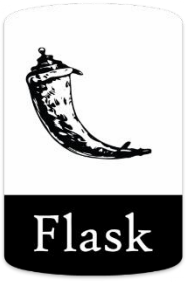
```
class Analize(DB):
    __tablename__ = 'Analize'

    id = Column(Integer, autoincrement=True, primary_key=True)
    id_sange_brut = Column(Integer, ForeignKey('SangeBrut.id'))
    alt = Column(Boolean, nullable=False)
    sif = Column(Boolean, nullable=False)
    antihtlv = Column(Boolean, nullable=False)
    antihtcv = Column(Boolean, nullable=False)
    antihiv = Column(Boolean, nullable=False)
    hb = Column(Boolean, nullable=False)

    sange_brut = relationship('SangeBrut', back_populates='analize')

```

2.Server dezvoltare – Flask



Este un framework pentru limbajul de programare Python care include:

- un server de development și debugger
- procesare de requesturi de tip RESTful

suport integrat pentru unit testing

Am ales acest server pentru dezvoltare deoarece aplicatia este construita in stilul architectural REST, iar acest framework oferă un mod simplu si elegant de tartare a cererilor de tip HTTP.

Exmplu de procesare request HTTP pentru login pe backend:

```
    def login_request(self):
        self.request_data = request.get_json()
        self.logger.debug("Got Login Request JSON: {}".format(self.request_data))
        user = self.request_data["username"]
        password = self.request_data["password"]

        status, user_type = self.controller.login(user, password)

        return_dict = {"status": status, "user_type": user_type}
        if status == 0:
            return_dict["message"] = "Login cu success!"
        else:
            return_dict["message"] = "Username sau parola invalide"

        self.logger.debug("Returning response for Login Request: {}".format(return_dict))
```

Exmplu de trimitere request HTTP de pe frontend:

```
private JSONObject sendRequest(HttpURLConnection http, String jsonString){
    byte[] outJson = jsonString.getBytes(StandardCharsets.UTF_8);
    int length = outJson.length;

    http.setFixedLengthStreamingMode(length);
    http.setRequestProperty("Content-Type", "application/json; charset=UTF-8");
    try {
        http.connect();
        try(OutputStream os = http.getOutputStream()) {
            os.write(outJson);
        }
        InputStream inputStream = http.getInputStream();
        BufferedReader streamReader = new BufferedReader(new InputStreamReader(inputStream,
        StringBuilder responseStrBuilder = new StringBuilder();

        String inputStr;
        while ((inputStr = streamReader.readLine()) != null)
            responseStrBuilder.append(inputStr);
        return new JSONObject(responseStrBuilder.toString());

    } catch (Exception e) {
        e.printStackTrace();
        return null;
    }
}
```

3.MySQL

Este un RDBMS(relational database management system) open source. Am ales această tehnologie deoarece toți membrii echipei sunt familiarizați cu sintaxa si utilizarea acesteia.

Partea de back-end nu comunică direct cu MySql, ci prin intermediul ORM-ului, astfel, baza de date MySql este folosita pentru stocarea tuturor datelor folosite in cadrul aplicatiei.

Pentru depanarea problemelor bazei de date si accesarea manuală am folosit MySql Workbench.

Tehnologii utilizate la nivel de client

1. JavaFX

Este o platformă pentru dezvoltarea aplicațiilor desktop, fiind inclusă în librăria „nativă” din Java. Am ales această tehnologie deoarece toți membrii echipei cunosc acest framework și considerăm că este o platformă rapidă și eficientă de dezvoltare a aplicațiilor de tip desktop.

În plus, folosim librării externe pentru facilitarea dezvoltării interfeței grafice precum: Jfoenix,ControlsFX,FontAwesomeFx.

2. Gluon

Este un tool folosit pentru design-ul aplicației, fiind un înlocuitor pentru Scene Builder, editor-ul prestabilit pentru javaFx. Am ales acest tool deoarece are o performanță mai ridicată decât tool-ul prestabilit, având totodată si suport pentru diferitele librării care conțin componente java pentru interfață(Jfoenix,ControlsFX,FontAwesomeFX).

3. JFoenix, ControlsFX, FontAwesomeFX

Sunt librării open-source care adaugă componente java pentru un design de tip minimalist pe front-end. Aceste librării sunt utilizate impreună cu Gluon pentru o dezvoltare eficientă a interfeței utilizatorului.

Am folosit JFoenix pentru componente precum JFXButton și JFXComboBox, ControlsFX a fost folosit pentru elemente precum SegmentedButton, iar

FontAwesomeFx pentru diferitele “glyph”-uri din aplicație.

Tehnologii utilizate pentru dezvoltarea aplicației

1. StarUML

Este un tool gratuit ce permite crearea diagramelor UML. A fost deosebit de utilă în faza de proiectare a bazei de date cât și de-a lungul dezvoltării aplicației.

2. Trello

Aplicație de gestiune a task-urilor în echipă. A fost aleasă datorită disponibilității pe mai multe platforme (Windows, Android, IOS) și notificărilor constante a membrilor echipei în legatură cu dezvoltarea proiectului.

3. GitHub

Este un serviciu de hostare si de control al versiunii care folosește Git.

În primul rând, acesta e folosit în cadrul proiectului pentru a păstra în mod securizat fișierele proiectului, având acces la proiect doar membrii echipei. În al doilea rând acesta permite lucrul pe aceleași fișiere concurent, de către mai mulți membrii ai echipei.

Abordarea noastră pentru dezvoltare a fost urmatoarea: dupa crearea task-urilor pe trello, fiecarui task i s-a asociat un branch pe GitHub. Astfel, membrii echipei au ales task-uri si au dezvoltat branch-ul asociat task-ului, urmând ca la final sa facă o cerere de unire cu branch-ul principal(branch-ul care conține ultima iterație funcțională a proiectului).

4. SourceTree

Este un client pentru Git care oferă o interfață grafică intuitivă pentru gestionarea branch-urilor de pe GitHub

Am mai folosit: Github Desktop sau direct din consola, fiecare cum a considerat.

Scenarii de testare

- **Scenariu de testare:** adaugare punga sange in stocul curent al centrului de transfuzii

Test case indentifier	TestAddPungaSange
Test location	In fereastra staff transfuzii
Feature to be tested	Punga cu sange a fost creata, despartita in cele 3 pungi si adaugate la stocul curent.
Feature fail/pass criteria	Testul este trecut daca pagina cu stocul curent se incarca cu succes contine cu exact 3 pungi mai mult decat inainte.
Means of control	1.Metoda TestAddPungaSange este apelata.
Data	2. Toate datele necesare pentru adaugarea unei noi donari sunt luate din fisierul testaddpungasange.txt
Test procedure	Se va face logarea automata cu un cont de staff transfuzii. Se apasa butonul stoc curent si se salveaza in memorie valorile pentru stoc inainte de adaugare.Se apasa butonul “Adauga cerere donare” si se va completa formularul cu datele din fisierul de intrare, urmat de un click pe “trimite formular” .Apoi, se va naviga printr-un click pe butonul “Vizualizare cereri donari” la meniul cu lista cererilor si se va selecta cererea cu numele si prenumele identice cu cele din fisierul de intrare. Se va apasa pe completare chestionar si se vor introduce date CORECTE, tot din fisierul de intrare si va apasa butonul de validare formular. Se va selecta din nou intrarea si se va apasa de data aceasta pe butonul cu prelucrarea sangelui.In cele din urma, se va selecta pentru ultima oara intrarea din tabel si se va apasa butonul de

	completare a analizelor, fiind alese din nou valori conforme si apasat butonul de trimitere rezultat analize. Acum va naviga la meniul de stoc curent printr-un click pe butonul de stoc curent si se compara valorile curente cu cele existente in memorie de la inceputul testului.In caz ca exista cu cate o punga mai mult, testul trece, altfel esueaza.
Special requirements	Trebuie sa ruleze pe un server de test, fara ca alti clienti sa fie conectati sau sa influenteze baza de date.

- **Scenariu de testare:** istoric donari

Test case indentifier	IstoricDonari
Test location	In fereastra donatorului
Feature to be tested	Donatorul trebuie sa poata sa isi vizualizeze donarile si rezultatele analizelor corespunzatoare
Feature fail/pass criteria	Testul este trecut daca tabelul cu donari se incarca cu succes si rezultatele analizelor pot fi vizualizate(daca sunt gata)
Means of control	1. IstoricDonariControlle.updateThis() este apelat dupa incarcarea fisierelor FXML si incarca in TableView toate donarile.
Data	2. Toate datele necesare sunt citite din baza de date. 3. Donarile sunt afisate in tabel(daca exista) si rezultatele analizelor sunt afisate intr-o fereastra separata dupa selectarea donarii. Daca rezultatele nu sunt gata inca, fereastra respectiva nu se va deschide.
Test procedure	Testerul se va loga cu un cont de donator. Se apasa butonul “Vizualizare istoric” si trebuie sa apara tabelul cu donari. La apasarea pe o donare trebuie sa se deschida fereastra cu rezultate analiza.
Special requirements	Serverul trebuie sa fie pornit si donatorul trebuie sa aiba donari in baza de date.

- **Scenariu de testare:**Persitsenta datelor la cererile de sange

Test case ID	PersistentaDatelorCerereSange
Test executed by	Iustin
Feature to be tested	Persistenta datelor la adaugarea unui cereri
Description/Summary of test	Se va testa daca la adaugarea unei cereri de sange, informatiile de acestea vor fi persistate in baza de date
Pre-condition	Trebuie sa exista conexiune la internet, utilizatorul sa fie logat cu un cont de medic si toate field-urile speicfice cererii sa fie completate
Test steps	Utilizatorul se logheaza cu un cont de medic valid Selecteaza tab-ul “Cerere sange” Se completeaza formularul specific cu datele de test Se apasa pe butonul “Trimitere” pentru a trimite cererea de sange
Test data	In formularul de detalii persoana se introduc urmatoarele date: In field-ul “Nume pacient” : Andrei Vasile In field-ul “CNP Client” : 1980911220090 In field-ul “Grupa de sange” : A2 In field-ul “RH” : POZITIV In field-ul “Importanta” : Medie In field-ul “Punga trombocite” : 1 In field-ul “Punga globule rosii” : 1 In field-uri “Punga plasma” : 1
Expected results	Informatiile sunt trimise sunt validate si apare o fereastra cu statusul cererii : in cazul in care datele au fost introduse corect utilizatorul este instiintat printr-o fereastra cu mesajul succes(cu datele de test utilizate la acest test), in cazul in care datele nu sunt valide va aparea un mesaj cu eroare respectiva
Post-condition	Cererea de sange este persistata in baza de date
Requirments	Serverul trebuie sa fie pornit si utilizatorul sa exista in baza de date

Scenariu de utilizare

- **Scenariu de utilizare istoric donari**

Scenariu	Donatorul doreste sa isi vada rezultatele analizelor
Actor	Donator
Conditii de intrare	Donatorul trebuie sa fie logat si rezultatele analizelor trebuie sa fie in baza de date
Conditiii de iesire	Donatorul a vazut rezultatele analizelor
Pasi	1. Se apasa butonul “Vizualizare istoric” 2. Se selecteaza donarea corespunzatoare 3. Daca rezultatele sunt gata, ele vor fi afisate intr-o noua fereastra

- **Scenarii de utilizare: register donator**

Nume use case	donatorRegister
Actor	donatorul
Conditie intrare	none
Conditie iesire	s-a inregistrat cu succes
Pasi:	1. deschide aplicatia 2. completeaza campurile cerute 3. da clic pe "Register" 4. corecteaza eventualele erori 5. asteapta confirmarea inregistrarii cu succes

- **Scenariu de utilizare: Adăugare rezultate analize pentru donator**

Actor: Staff transfuzii
Condiție intrare: Utilizatorul să fie logat pe un cont de staff transfuzii. Sa existe o punga de sange de la donator si sa fie gata de analizat.
Condiții de ieșire: Punga de sange a fost analizata, donatorul pungii poate vedea rezultatele analizei.

Pasi:

1. Click pe butonul de vizualizare cereri de donare
2. Alegerea din tabelul cu cereri de donare punga de sange care trebuie analizata urmata de un click pe aceasta.
3. Click pe butonul de completare analize donatie
4. Alegerea pe rand a: grupei de sange, rh si rezultatelor pentru boli din lista.
5. Click pe butonul din partea de jos a ferestrei “Trimitere rezultat analiza”
6. Click pe ok in fereastra de confirmare

Scenarii de utilizare generale medic

Register

- completeaza campurile necesare
- completeaza licenta primita
- daca datele personale si licenta sunt valide contul se va salva in baza de date, iar aceasta va avea posibilitatea sa se logheze in sistem
- daca datele introduse nu sunt valide acesta primeste un mesaj de eroare in urma caruia va fii posibilia remedierea greselilor

Login

- daca introduce username-ul/email-ul (doar una dintre acestea) si parola cu care acesta s-a inregistrat, i se va deschide fereastra de administrare pentru medic
- in caz contrar va primii un mesaj de eroare si va fii rugat sa incerce din nou

Logout

- din fereastra de administrare aceasta va avea posibilitatea de logout, actiune care il va intoarce in fereastra de login

Meniul principal

- in meniul principal medicul are posibilitatea de vizualizare a pungilor de sange prelucrat, disponibil in cel mai apropiat centru
- poate sa vada cele mai importante 5 cereri de sange

Cerere sange

- in meniul pentru cerere de sange, poate sa faca o cerere de sange pentru un anumit pacient specificand datele personale ale pacientului, importanta cererii de sange si numarul de pungi din fiecare tip de care are nevoie
- daca la apasarea butonului Trimitere datele introduse sunt valide cererea este inregistrata in baza de date aceasta fiind vizibila in meniul Vizualizare cereri sange
- in caz contrar, acesta este avertizat si are posibilitatea remedierii greselilor

Vizualizare cereri sange

- in Vizualizare cereri sange persoana logata poate vizualiza cererile in astepare din spitalul de care apartine, de asemenea poate anula o cerere de sange
- in tabelul de cereri completate poate vizualiza toate cererile care au fost onorate pana in momentul de fata din spital
- pentru fiecare dintre cele doua tabele are posibilitatea de a filtra cererile dupa numele pacientului

Management pacienti din spital

- in fereastra de Management pacienti, poate vizualiza toti pacientii din spitalul de care apartine
- medicul are dreptul de a adauga noi pacienti, pentru fiecare completeaza datele personale ale acestuia
- daca pacientul nu exista aceasta este salvat in baza de date si este disponibil pentru crearea unei noi cereri de sange pentru acesta
- pentru vizualizarea pacientilor medicul are acces la filtrarea pacientilor din spital dupa nume

Scenarii de utilizare generale staff transfuzii

Register

- completeaza campurile necesare si licenta primita, la fel ca la medic

Login

- daca introduce username-ul/email-ul (doar una dintre acestea) si parola cu care acesta s-a inregistrat, i se va deschide fereastra de administrare a centrului de transfuzii
- in caz contrar va primii un mesaj de eroare si va fii rugat sa incerce din nou

Logout

- din fereastra de administrare aceasta va avea posibilitatea de logout, actiune care il va intoarce in fereastra de login

Vizualizare status curent:

- donari de sange facute in ziua curenta
- stoc curent de pungi de: trombocite, globule rosii, plasma, sange
- notificari

Vizualizare cereri de sange in asteptare

- va fi afisat un tabel cu datele despre pacient(nume, grupa de sange, RH, pungi trombocite, pungi globule rosii, pungi plasma, data, importanta)
- dupa selectarea unui pacient, va fi posibila trimiterea unui mesaj catre doctorul responsabil de acesta

Vizualizare cereri de donare in asteptare

- va fi afisat un tabel cu datele despre donator(nume, prenume, grupa, RH, etapa curenta)
- dupa selectarea unui donator, se va putea trece la etapa curenta din procesul de donare(prelevare, pregatire, calificare)
- se va putea filtra tabela dupa anumite criterii(nume donator, prenume donator, etapa)

Adaugare cerere donare

- un donator poate completa formularul la centrul de transfuzie. In acest caz, personalul va inregistra formularul respectiv in baza de date. In aplicatie va aparea ca si cerere de donare

Scenarii de utilizare generale donator

Register:
-completeaza campurile necesare
-daca campurile au fost completate corect, contul donatorului va fi inregistrat in BD
-daca nu, utilizatorul va primi un mesaj de eroare cu campurile gresite

Log in:
-introduce username si parola
-daca sunt cele corecte, se va lansa aplicatia

Log out:
-se revine la fereastra de log in

Vizualizare istoric:
-donatorul va vedea o tabela cu toate donatiile sale, avand coloanele: numar donare, data donare, nume centru de donare, status
-acestea pot fi filtrate dupa data, centru de donare si status

Vizualizare analize:
- in istoric, la apasarea pe status-ul unei donari de sange, daca status-ul este ‘ready’, donatorul va putea vizualiza rezultatele analizelor de sange corespunzatoare acelei donari

Doresc sa donez:
- donatorului i se vor aminti conditiile pe care trebuie sa le indeplineasca pentru a putea dona sange: conditii fizice, sa nu fi avut unele boli sau tratamente.
- dupa ce donatorul a luat la cunostinta aceste conditii, el va trebui sa completeze un formular in care apar campuri obligatorii(nume, prenume, sex, telefon, domiciliu, resedinta) si campuri optionale(grupa de sange, RH, zilele in care este disponibil si daca doneaza pentru un anumit pacient)
- dupa completarea formularului, donatorului i se va reaminti cum sa se pregateasca inainte de a dona sange, precum si ce ii este interzis

Diagrama de use case

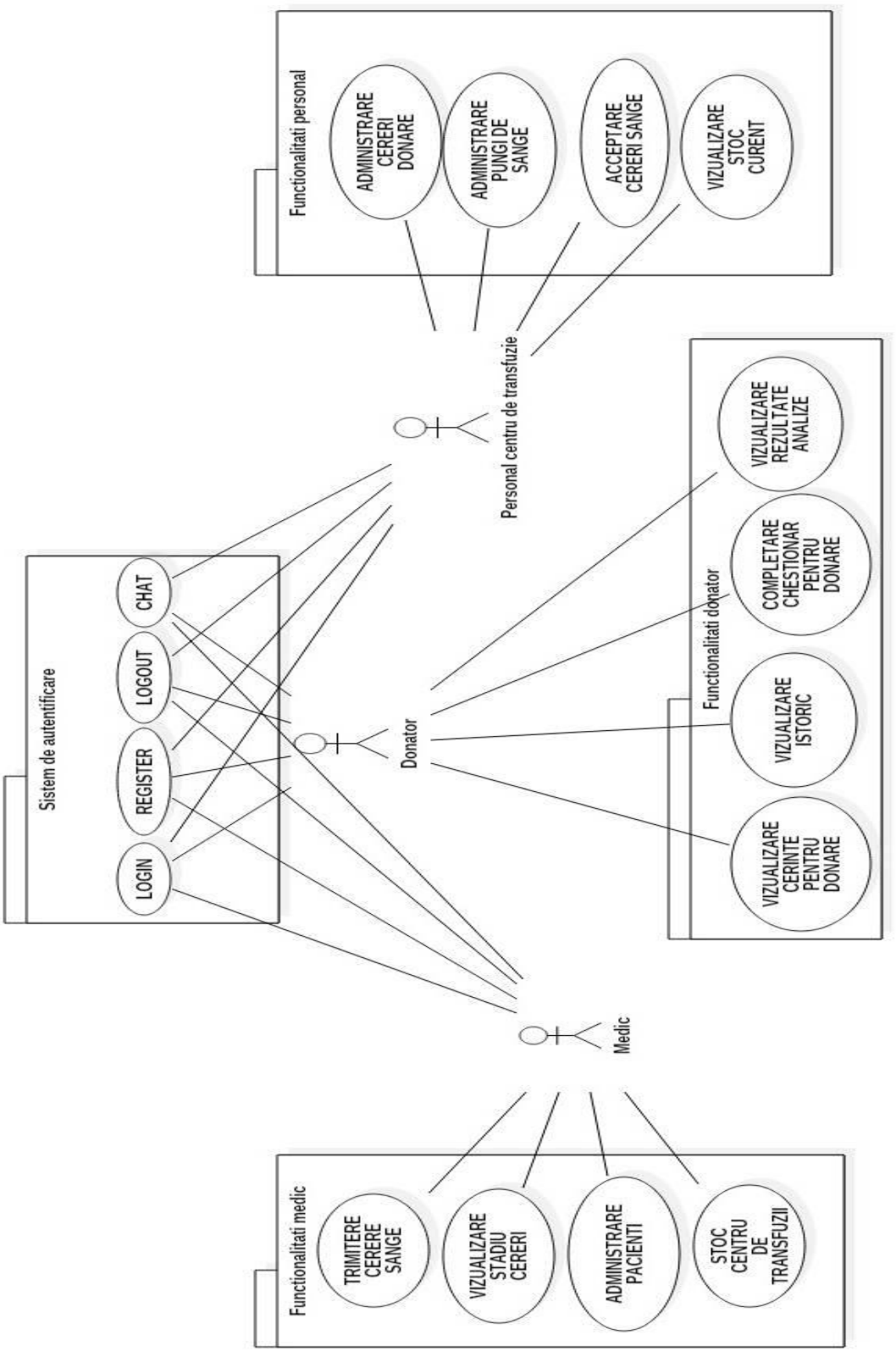
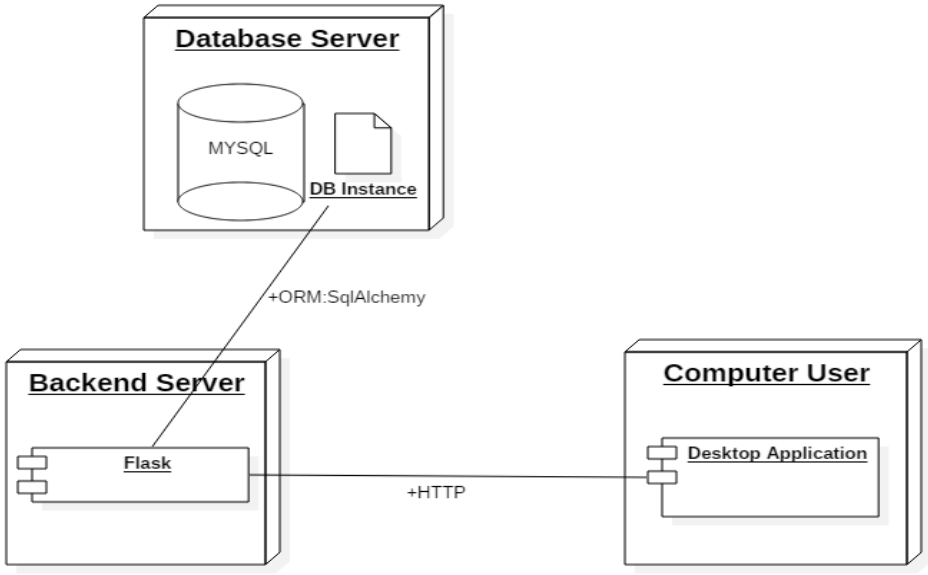
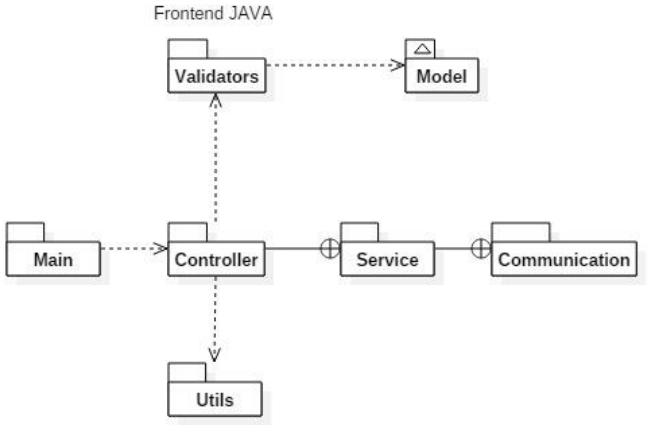


Diagrama de arhitectura





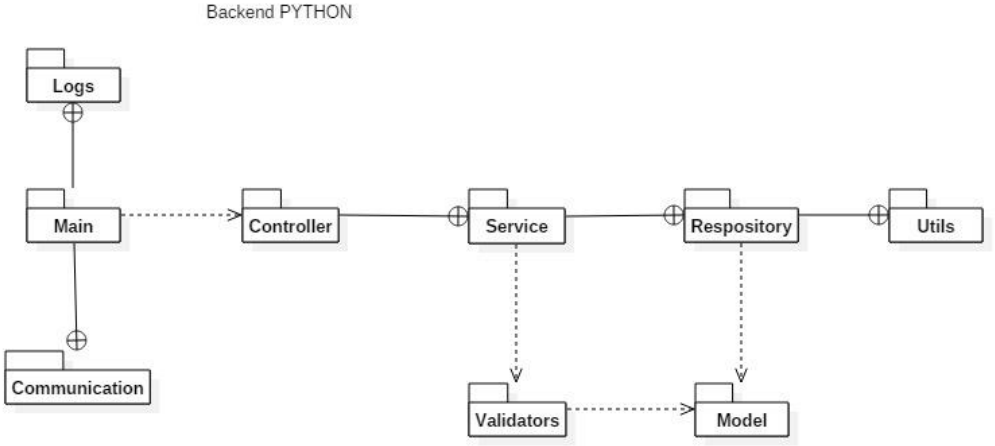


Diagrama baza de date

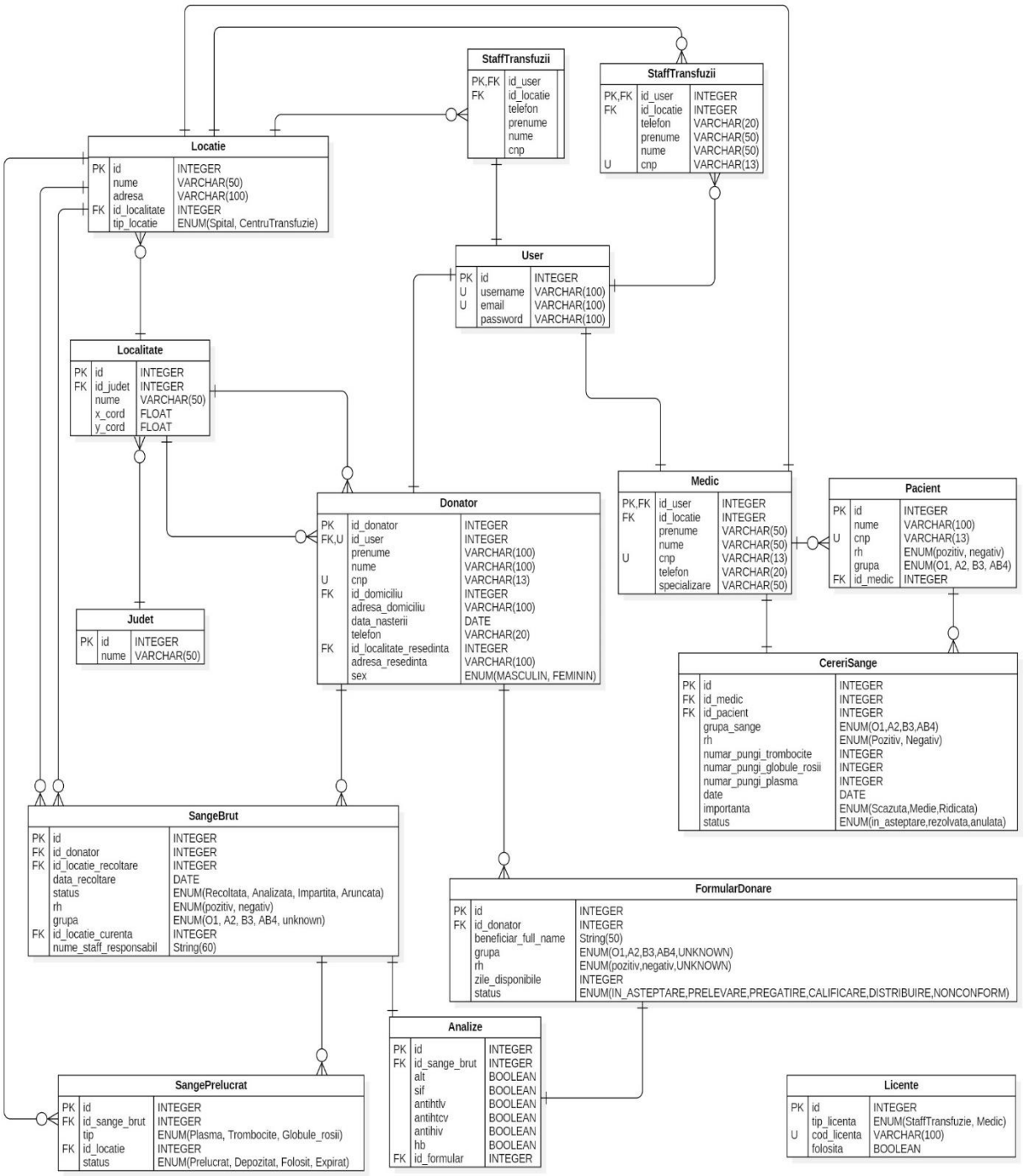
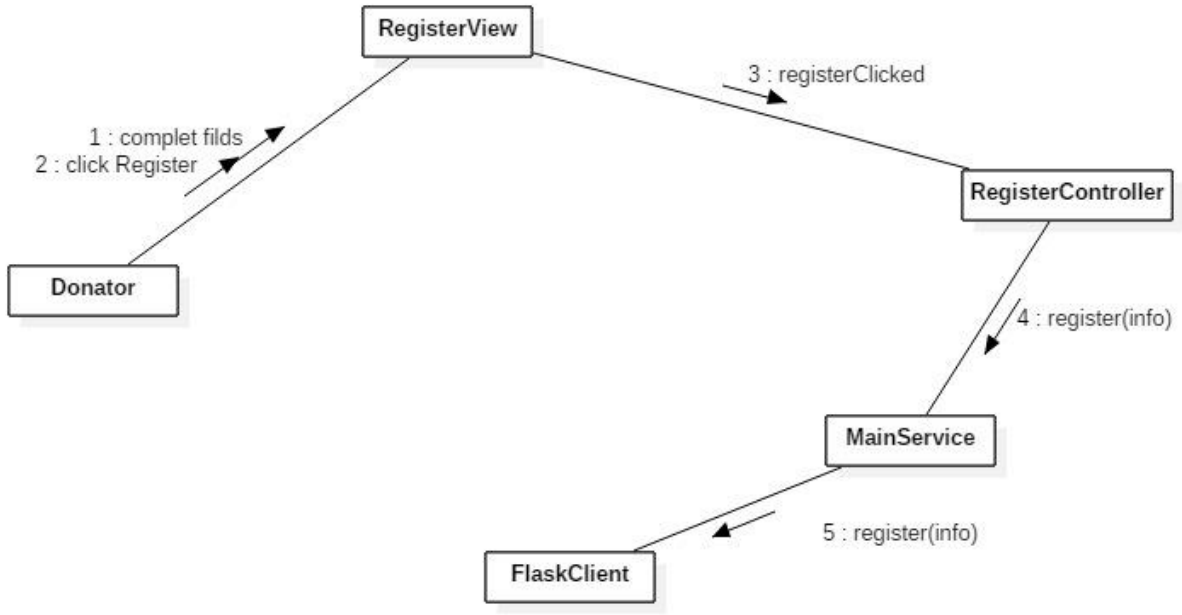
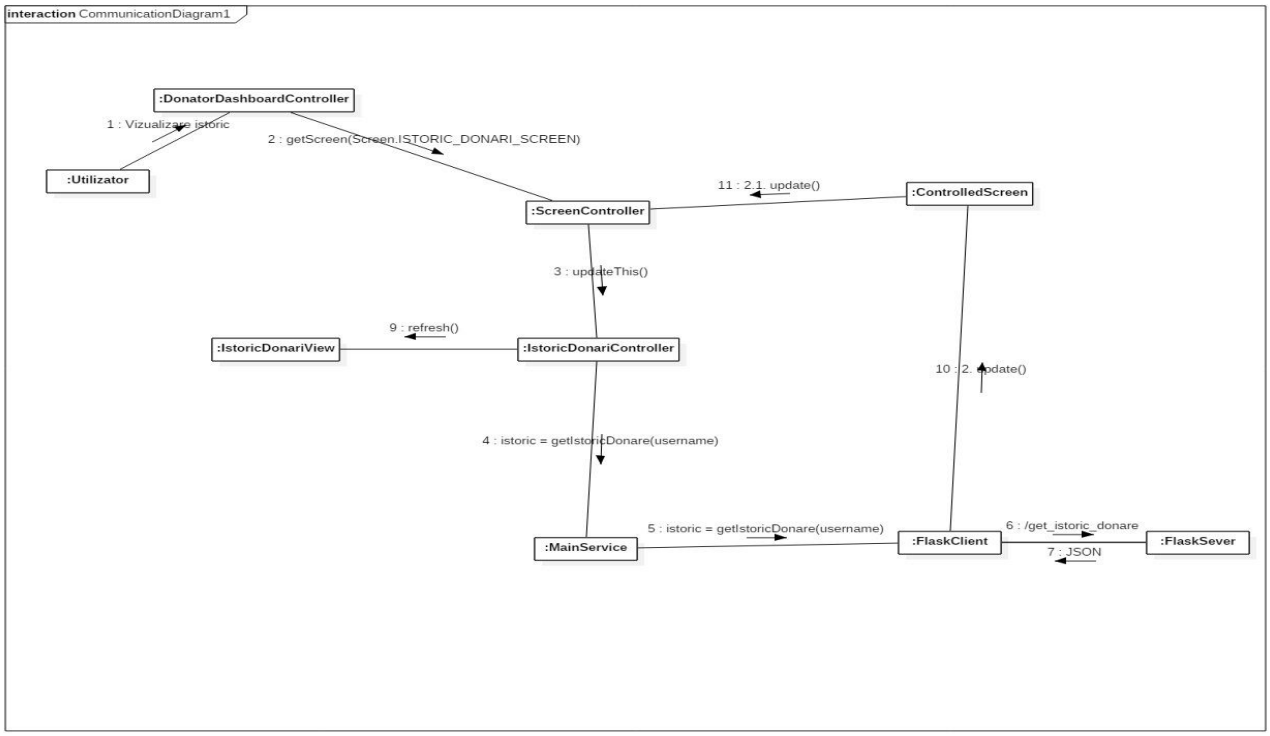


Diagrama de comunicare

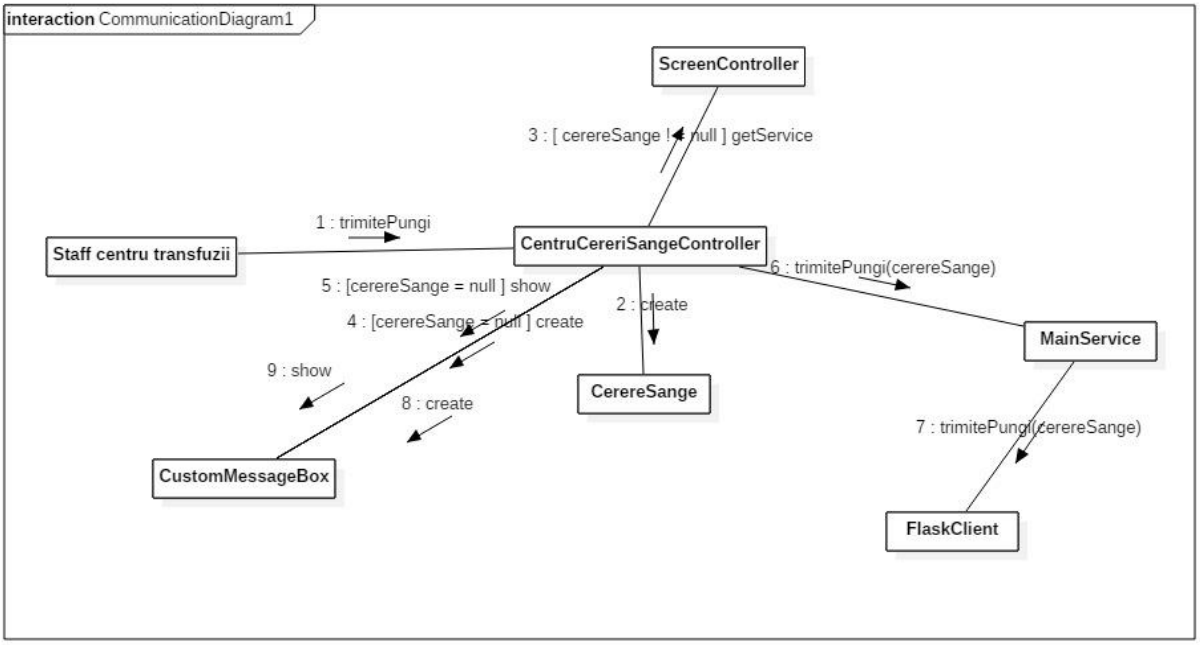
- Register



- Istoric donari



- Staff: trimiterea pungilor de sange



- Medic: Management Pacienti

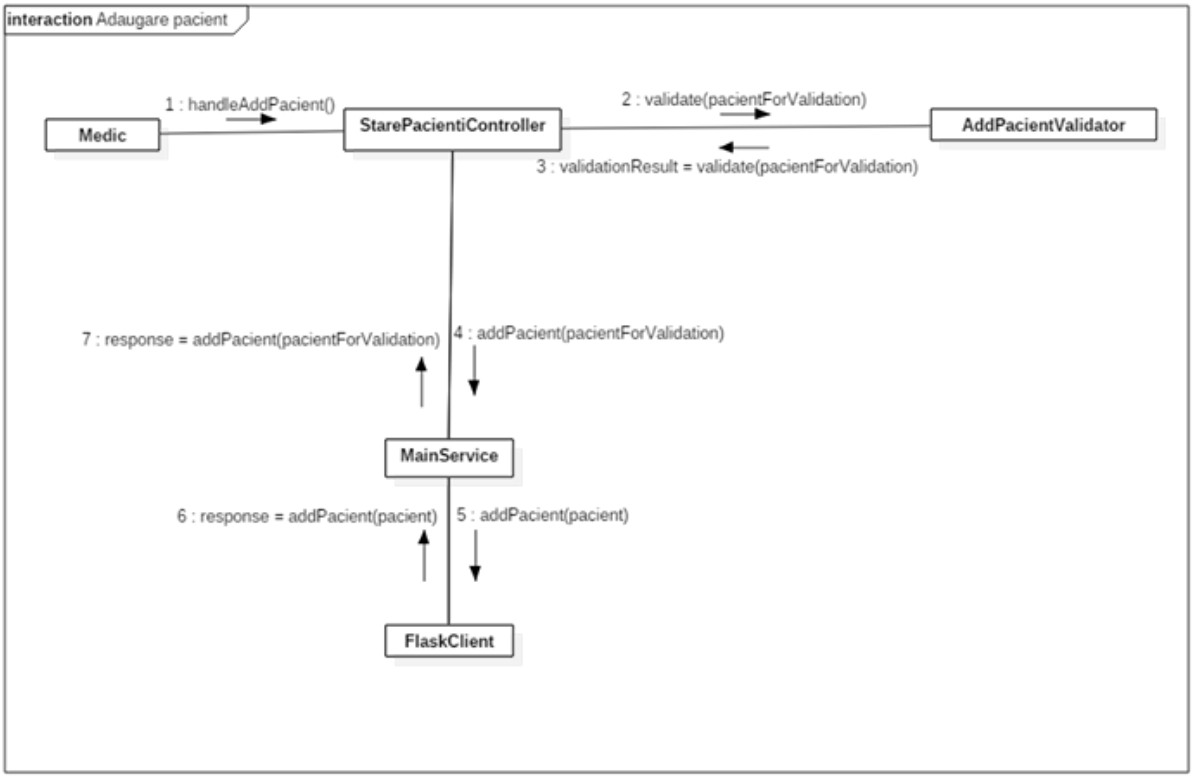
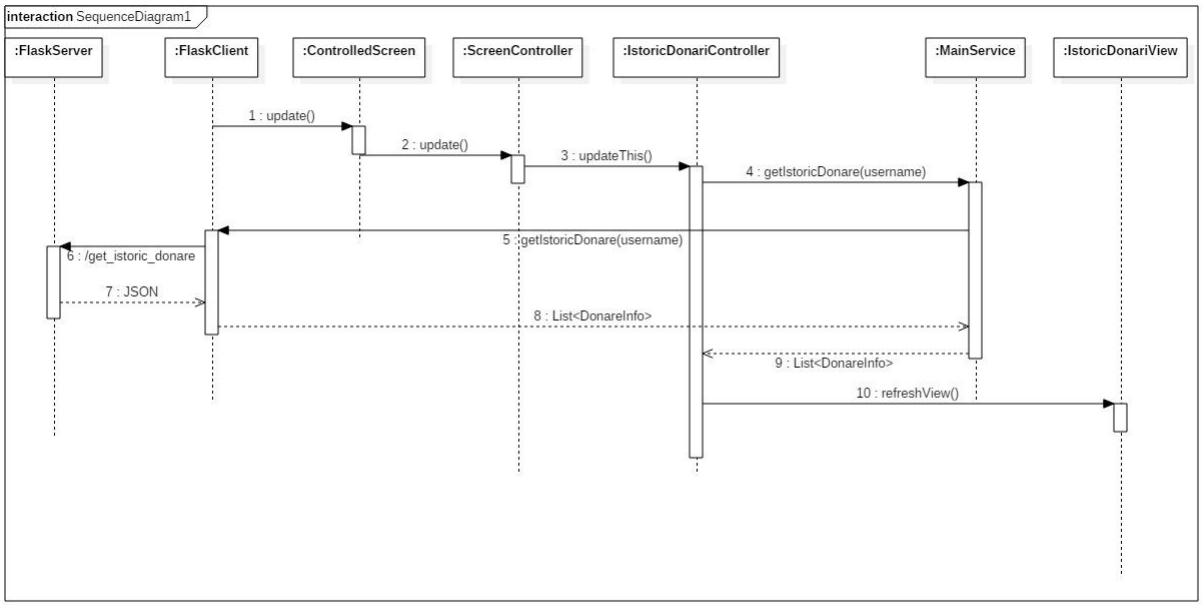
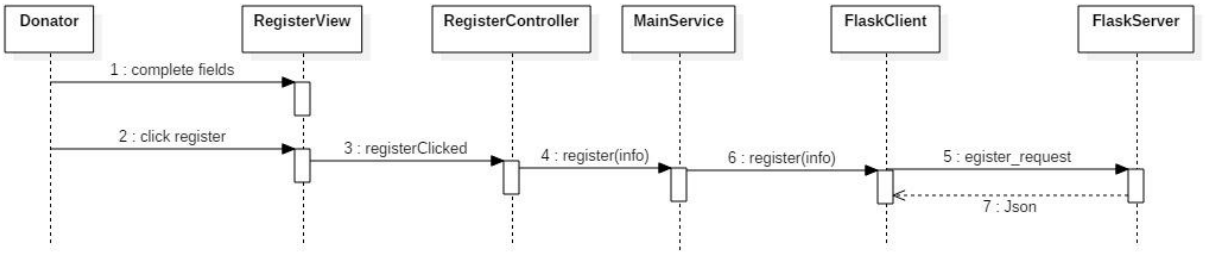


Diagram secventa

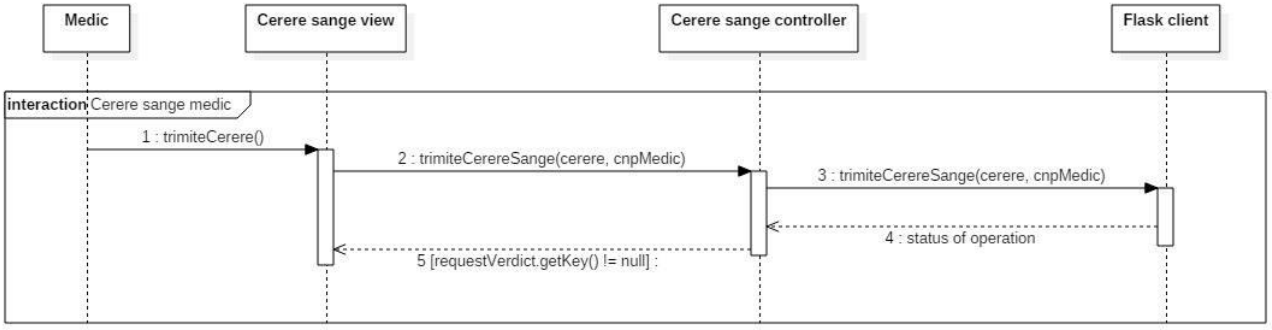
- Donator: Istoric de donari

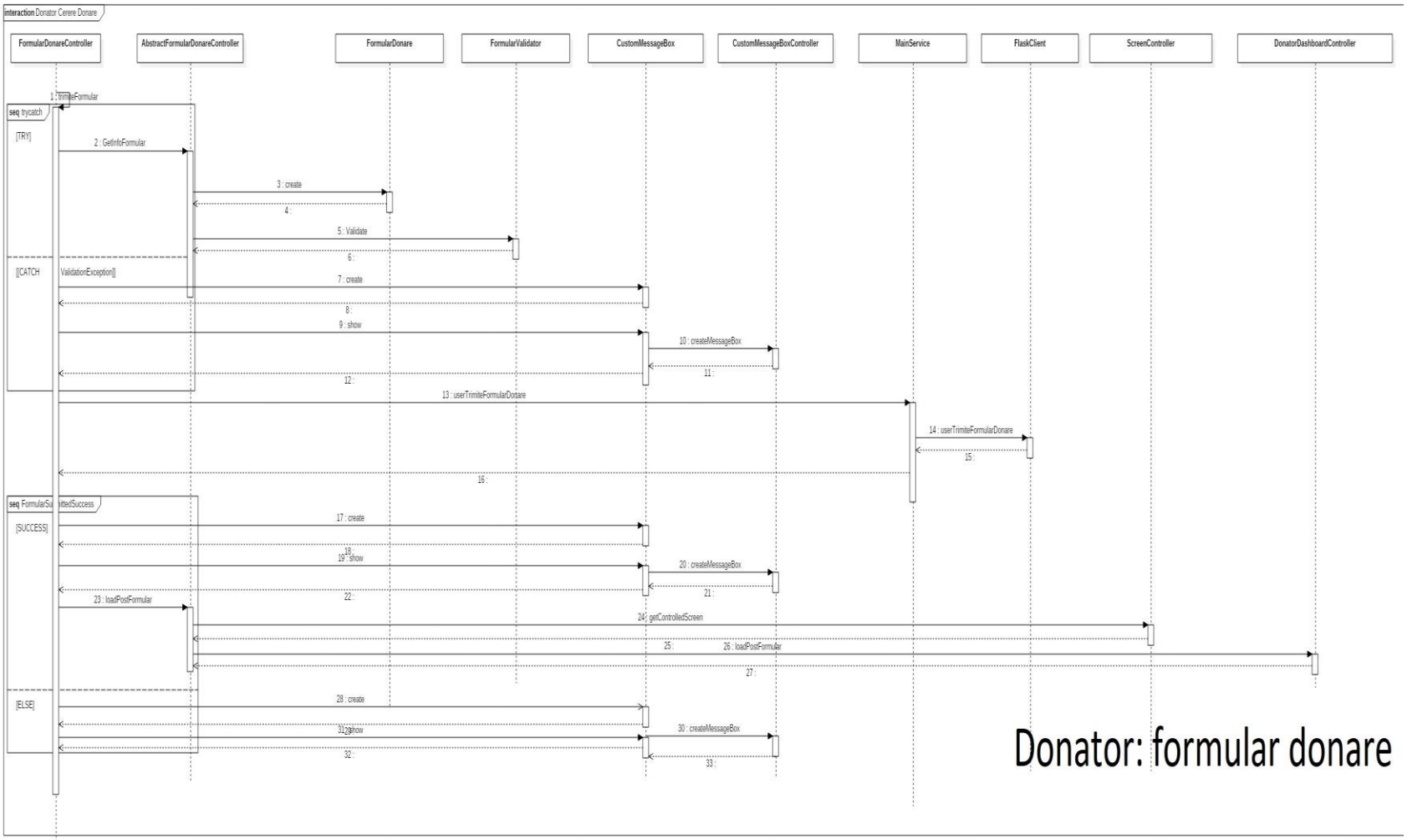


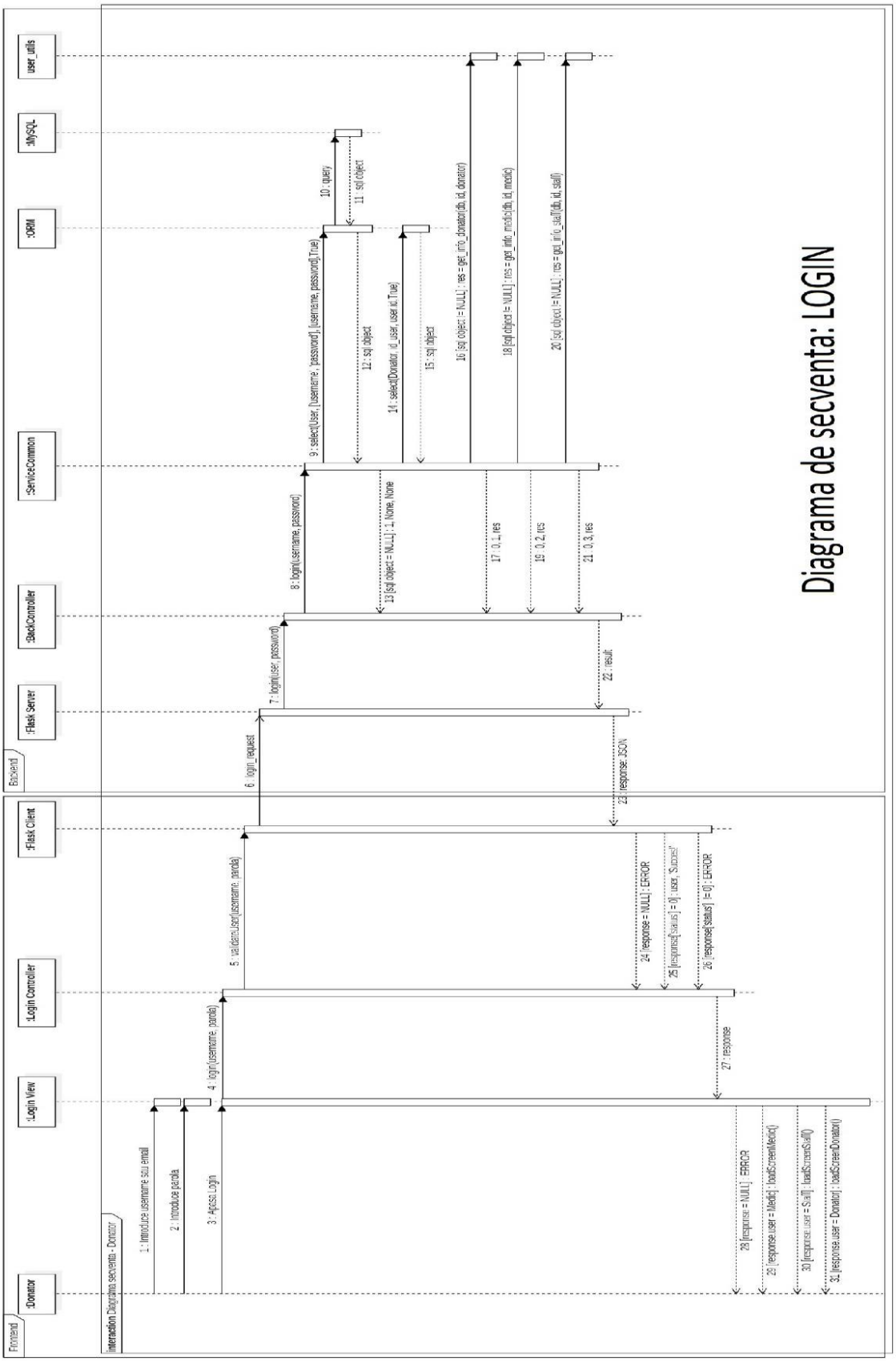
- Register

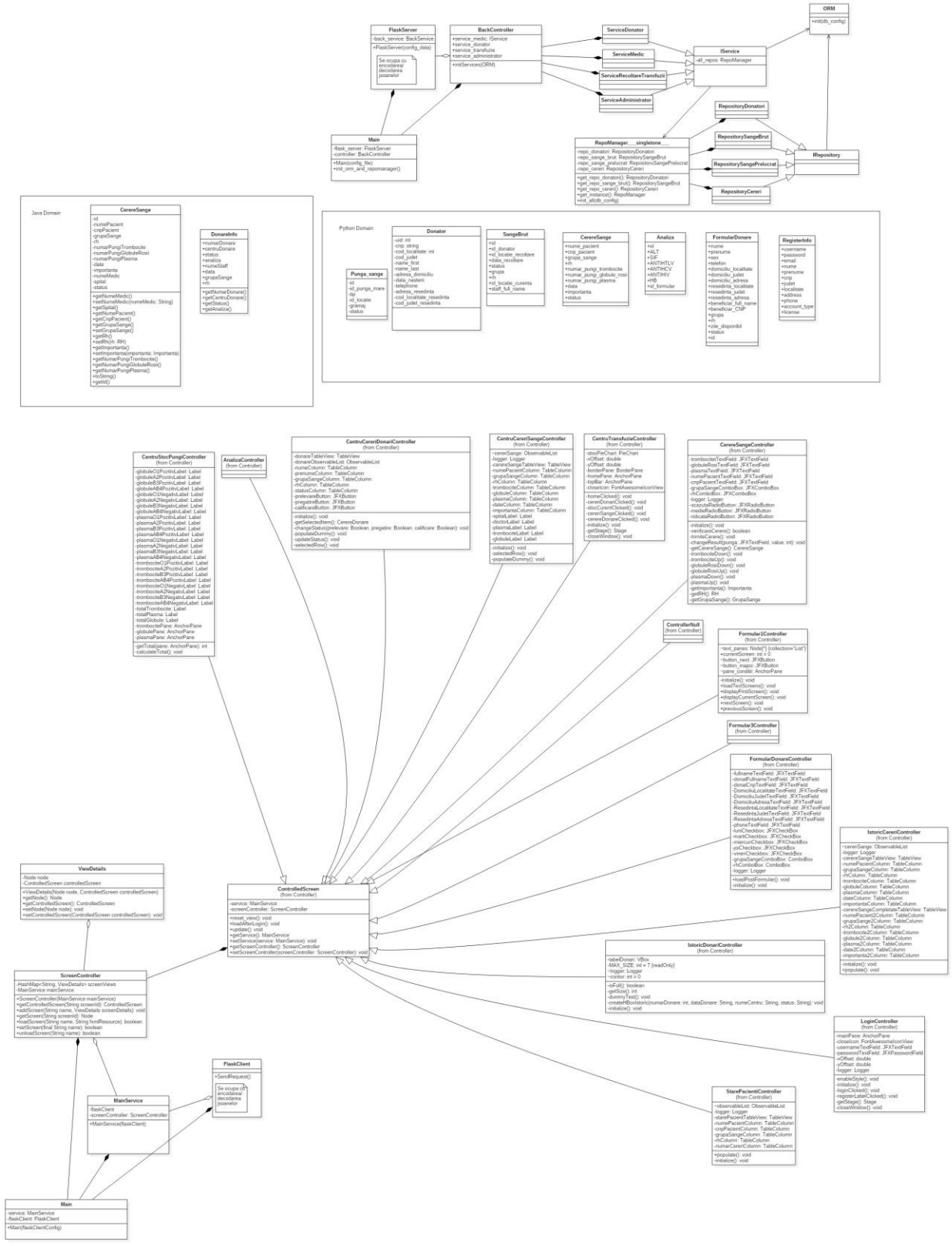


- Medic: cerere sange




interaction Donator Cerere Donare
FormularDonareController
AbstractFormularDonareController
FormularDonare
FormularValidator
CustomMessageBox
CustomMessageBoxController
MainService
FlaskClient
ScreenController
DonatorDashboardController
1 : trimiteFormular
seq trycatch
[TRY]
2 : GetInfoFormular
3 : create
4 :
5 : Validate
6 :
[[CATCH
ValidationException]]
7 : create
8 :
9 : show
10 : createMessageBox
11 :
12 :
13 : userTrimiteFormularDonare
14 : userTrimiteFormularDonare
15 :
16 :
seq FormularSu
ittedSuccess
[SUCCESS]
17 : create
18 :
19 : show
20 : createMessageBox
21 :
22 :
23 : loadPostFormular
24 : getControlledScreen
25 :
26 : loadPostFormular
27 :
[ELSE]
28 : create
29 :
30 : createMessageBox
31 :show
32 :
33 :
Donator: formular donare


Frontend
Backend
interaction Diagrama secventa - Donator
:Donator
:Login View
:Login Controller
:Flask Client
:Flask Server
:BackController
:ServiceCommon
:ORM
:MySQL
user_utils
1 : introduce username sau email
2 : introduce parola
3 : Apasa Login
4 : login (username, parola)
5 : validam user(username, parola)
6 : login_request
7 : login(user, password)
8 : login(username, password)
9 : select(User, [username = 'password'], [username, password],True)
10 : query
11 : sql object
12 : sql object
13 [sql object = NULL] : 1, None, None
14 : select(Donator, id_user, user.id,True)
15 : sql object
16 [sql object = NULL] : res = get_info_donator(db, id_donator)
17 : 0, 1, res
18 [sql object != NULL] : res = get_info_medic(db, id_medic)
19 : 0, 2, res
20 [sql object != NULL] : res = get_info_staff(db, id_staff)
21 : 0, 3, res
22 : result
23 : response: JSON
24 [response = NULL] : ERROR
25 [response['status'] = 0] : user, 'Success!'
26 [response['status'] != 0] : ERROR
27 : response
28 [response = NULL] : ERROR
29 [response.user = Medic] : loadScreenMedic()
30 [response.user = Staff] : loadScreenStaff()
31 [response.user = Donator] : loadScreenDonator()
Diagrama de secventa: LOGIN


FlaskServer
BackController
ServiceDonator
ServiceMedic
ServiceRecoltareTransfuzii
ServiceAdministrator
IService
ORM
Main
RepoManager___singletone___
RepositoryDonatori
RepositorySangeBrut
RepositorySangePrelucrat
RepositoryCereri
IRepository
Java Domain
CerereSange
DonareInfo
Python Domain
Donator
SangeBrut
CereriSange
Analiza
FormularDonare
RegisterInfo
Punga_sange
CentruCereriDonariController
CentruCereriSangeController
CentruTransfuzieController
CereriSangeController
AnalizaController
CentruStocPungiController
ControllerNull
Formular1Controller
Formular3Controller
FormularDonareController
IstoricCereriController
ViewDetails
ControlledScreen
ScreenController
IstoricDonariController
LoginController
StareePacientiController
FlaskClient
MainService
Main