

# Beauty Salon

Nume: Drăguș Andreea

Grupa: 30236

Cuprins

[Beauty Salon 1](#_Toc131068144)

[Cuprins 2](#_Toc131068145)

[Deliverable 1 3](#_Toc131068146)

[1. Specificatia proiectului 3](#_Toc131068147)

[2. Cerinte functionale 3](#_Toc131068148)

[3. Cazuri de utilizare (usecase) 4](#_Toc131068149)

[4. Cerinte non-functionale 7](#_Toc131068150)

[5. Design Constraints 8](#_Toc131068151)

[6. Glossary 8](#_Toc131068152)

# Deliverable 1

## 1. Specificatia proiectului

Acest proiect reprezinta o poarta intermediara intre clientii si antreprenorii care au in vizor domeniul infrumusetarii. Astfel, la baza acestei lucrari sta o platforma interactiva prin intermediul careia informatiile sa fie transmise cu usurinta intre clienti si un salon de beauty.

Scopul proiectului este eficientizarea timpului si a organizarii unui salon. Astfel, toate informatiile relevante pentru un client sunt la un click distanta si se pot realiza de asemenea si programari.

## 2. Cerinte functionale

Aplicatia este conceputa astfel incat sa perceapa 3 tipuri de utilizatori: admin, client, angajat. Cele trei tipuri de actori au atat caracteristici comune precum: nume, username, parola, dar si elemente specifice: clientul are puncte de loialitate, iar angajatul are un domeniu specific in care activeaza: hair-dresser, nail-tech sau make-up artist.

Fiecare tip de utilizator dispune de functionalitati diferite:

* ***Admin:*** poate sa gestioneze baza de date si sa faca diverse interogari precum: sa adauge utilizatori de orice tip in baza de date, sa stearga utilizatori, sa adauge programari, sa stearga programari, sau sa actualizeze informatiile deja existente.
* ***Client:*** poate sa isi creeze un cont, sa isi actualizeze diferite date, sa caute angajati dupa diverse criterii precum domeniul de activitate, sa creeze o programare sau sa stearga din programarile sale. Cu fiecare programare adaugata, punctele de loialitate ale clientului se incrementeaza cu o valoare proportionala cu valoarea serviciilor din programare.
* ***Angajat:*** poate sa isi creeze cont, sa isi actualizeze detaliile contului, sa creeze sau sa stearga din propriile programari.

Fiecare actiune mentionata mai sus este insotita de validari si verificari astfel incat sa nu poata fi realizata daca nu indeplineste anumite criterii.

La adaugarea unui user, se verifica coerenta datelor: field-urile de nume, username si parola trebuie sa fie nenule si tipul de utilizator sa fie unul dintre cele 3 valide. Daca din aceasta perspectiva este totul in regula, atunci se face o verificare in baza de date pentru a ne asigura ca nu exista un alt user care sa aiba acelasi username. La finalizarea cu succes a verificarilor, userul e adaugat atat in tabelul de user cat si in tabelele aferente tipului de user.

In cazul in care o actiune nu este validata cu succes, se vor arunca exceptii specifice: InvalidUserException, InvalidClientException, InvalidEmployeeException, InvalidAppointmentException.

La update-ul unui user, se aplica din nou aceleasi cerinte mentionate mai sus, astfel incat sa ne asiguram ca nu exista update-uri inconsistente.

La stergerea unui user, daca acesta este de tipul client sau angajat, se vor sterge si toate programarile aferente lui.

Interogarile care au ca scop gasirea unui user sau a mai multora conform diferitor criterii, vor arunca exceptii specifice in cazul in care nu este niciun utilizator care sa indeplineasca cerintele: UserNotFoundException, ClientNotFoundException, EmployeeNotFoundException, AppointmentNotFoundException.

O entitate importanta a proiectului este Appointment. Aceasta este alcatuita din un client, un angajat, o data respectiv ora si o lista de servicii. Pretul total al appointment-ului este calculat ca suma tutror preturilor serviciilor incluse in lista si va fi transpus sub forma unor puncte de loialitate alocate clientului. Pentru a salva in baza de date o programare, asupra acesteia se fac multiple verificari: clientul/angajatul sa fie valizi si sa fie regasiti in tabelele aferente celor doua tipuri de entitati si in tabelul care contine toti userii, data sa nu fie in trecut sau intr-o zi nelucratoare (sambata sau duminica), timpul sa respecte urmatoarele criterii: sa fie in intervalul de lucru (08:00- 18:00) si minutele sa fie 0 sau 30. Dupa ce se realizeaza validarile mentionate, se verifica si programele clinetului si al angajatului, astfel incat sa nu existe pentru nici unul dintre ei o alta programare care sa se desfasoare la data si timpul respectiv. Daca aceste cerinte sunt satisfacute, programarea este adaugata in baza de date si in listele de programari ale celor doua entitati.

## 3. Cazuri de utilizare (usecase)

Use case 1: Register

Level: User-goal

Primary Actor: User

Main success scenario: inregistrare cu success, salvare in baza de date a noului user

Use case 2: Login

Level: Subfunction

Primary Actor: User

Main success scenario: logare cu success, utilizatorul a fost identificat in baza de date

Use case 3: Update account

Level: Subfunction

Primary Actor: User

Main success scenario: update cu success, salvare in baza de date a modificarilor

Use case 4: Delete account

Level: Subfunction

Primary Actor: Admin

Main success scenario: delete cu success, stergerea contului din baza de date si a programarilor aferente acestuia

Use case 5: Create appointment

Level: User-goal

Primary Actor: Client, Angajat

Main success scenario: programarea este valida si salvata in baza de date

Use case 6: Update Appointment

Level: Subfunction

Primary Actor: Client, Angajat

Main success scenario: update cu success, salvarea in baza de date a modificarilor

Use case 7: Delete appointment

Level: Subfunction

Primary Actor: Admin

Main success scenario: delete cu success, stergerea programarii din baza de date si din listele de programari a clientului si a angajatului

Use case 7: Find appointments by criteria

Level: User-goal

Primary Actor: User

Main success scenario: gasirea cu success a unei sau a mai multor programari care sa respecte criteriile de cautare

Use case 8: Find users by criteria

Level: Subgoal

Primary Actor: Admin

Main success scenario: gasirea cu success a unui sau a mai multor utilizatori care sa respecte criteriile de cautare

Use case 9: Find clients by criteria

Level: Subgoal

Primary Actor: Admin

Main success scenario: gasirea cu success a unui sau a mai multor clienti care sa respecte criteriile de cautare

Use case 10: Find employees by criteria

Level: Subgoal

Primary Actor: Admin

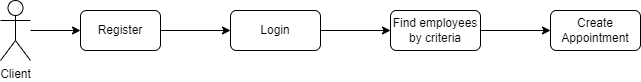
Main success scenario: gasirea cu success a unui sau a mai multor angajati care sa respecte criteriile de cautare

Use case 11: Beauty Salon

Level: High-Level

Primary Actor: Client

Main success scenario: parcurgerea cu succes a tuturor pasilor necesari realizarii cu success a unei programari in cadrul aplicatiei



## 

## 4. Cerinte non-functionale

1. Fiabilitate

Sistemul construit este fiabil din mai multe puncte de vedere. Aparitia unei erori critice in cadrul aplicatiei are o probabilitate scazuta intrucat aceasta are la baza sa un schelet complex de validari care arunca exceptii controlate astfel incat sa blocheze orice interactiune nepermisa a utilizatorului cu sistemul. Au fost testate si cuprinse in validari un procent considerabil de situatii exceptionale care pot aparea in cadrul utilizarii aplicatiei, fapt care reduce semnificativ probabilitatea de aparitie a unei erori critice. In cazul in care se intampla acest lucru, timpul de redresare nu ar trebui sa depaseasca 10 minute, intrucat aplicatia are o structura clara si robusta, iar timpul aferent identificarii sursei problemei este redus considerabil de acest aspect.

De asemenea, a fost realizata o testare riguroasa a intregului sistem si a fiecarui caz de utilizare, atat in cazul pozitiv cat si in cazul negativ, fiind implementate un total de 65 de teste JUnit, care sa asigure functionalitatea corecta a sistemului.

2. Utilizabilitate

Din perspectiva utilizatorului, aplicatia va avea un aspect simplu, astfel incat utilizarea acesteia sa aiba un grad minimalist de dificultate. Optiunile pe care fiecare tip de utilizator le are sunt clar stabilite, iar modul de interfatare ale acestora va fi unul user-friendly.

3. Portabilitate si compatibilitate

Aplicatia este portabila, putand fii rulata pe diverse dispozitive cu diverse sisteme de operare. Cu toate acestea, exista restrictii cu privire la versiunile de java si spring-boot folosite: 19 si respectiv 3.0.4. Un alt aspect important il reprezinta dependintele care trebuie instalate inainte de rularea proiectului, fiind vitala integrarea lor.

4. Scalabilitate

Din punctul de vedere al scalabilitatii, aplicatia poate fi extinsa cu necesitate minima de a modifica restul componentelor, intrucat exista un grad redus de cuplare intre clase, pachete si respectiv layere. Scheletul proiectului este proiectat intr-o maniera puternic coeziva, astfel incat toate clasele au functionalitati targetate si foarte clar definite.

## 5. Design Constraints

Din punct de vedere arhitectural, proiectul este structurat dupa modelul MVC, astfel incat sa existe o departajare cat mai clara intre layerele aplicatiei. Astfel, exista un fir logic al traversarii datelor, pornind de la legatura cu baza de date reprezentata de layer-ul de repository, continuand cu layerul de service si ajungand la cel de controller. De asemenea, exista definite pachete aditionale aferente validarilor si a exceptiilor definite.

Din punct de vedere al limbajului de programare, proiectul este scris in Java si Spring Boot, versiunile 19 si 3.0.4.

Au fost folosite diverse librarii aditionale pentru a reduce cat mai mult din complexitatea si masa codului. Astfel, pentru generarea metodelor accesoare/mutatoare si a unor constructori au fost folosite adnotari specifice Lombok. De asemenea, pentru a genera in mod automat tabelele aferente aplicatiei s-au folosit adnotari specifice Hibernate, cu ajutorul carora s-au definit si relatiile dintre tabele. In cadrul testelor, au fost folosite diverse elemente care tin de libraria Mockito.

# 6. Glossary

***MVC*** este un model în proiectarea software-ului utilizat în mod obișnuit pentru implementarea interfețelor utilizator, a datelor și a logicii de control. Subliniază o separare între logica de business a software-ului și afișaj. Această „separare a preocupărilor” asigură o mai bună diviziune a proiectului și o întreținere îmbunătățită.

***Repository pattern*** separă logica de acces la date și o mapează la entitățile din logica de business. Realizează logica de acces la date mapata la entitatile din domeniu. În modelul Repository, entitățile de domeniu, logica de acces la date și logica de afaceri comunica folosind interfețe.

***Service pattern*** este un model arhitectural, aplicat în cadrul paradigmei de proiectare orientată către servicii, care urmărește organizarea serviciilor într-un set de layere logice. Serviciile care sunt clasificate într-un anumit nivel partajează funcționalitatea.

***Nivelul Controller*** controlează domeniul de aplicare a tranzacției și gestionează informațiile legate de sesiune pentru cerere. Controlerul trimite mai întâi la o comandă și apoi apelează logica corespunzătoare de procesare a vizualizării pentru a da răspunsul.