Dezvoltarea unei aplicații de tip mobile pentru saloanele de înfrumusețare

# Documentul de proiectare

Cuprins

[1. Introducere 1](#_Toc160527836)

[1.1 Scopul documentului 1](#_Toc160527837)

[2. Prezentare generală și abordări de proiectare 2](#_Toc160527838)

[2.1 Prezentare generală 2](#_Toc160527839)

[2.2 Presupuneri/ Constrângeri/ Riscuri 2](#_Toc160527840)

[2.2.1 Presupuneri 2](#_Toc160527841)

[2.2.2 Constrângeri 2](#_Toc160527842)

[2.2.3 Riscuri 3](#_Toc160527843)

[3. Considerațiii de proiectare 4](#_Toc160527844)

[3.1 Obiective și linii directoare (ghiduri) 4](#_Toc160527845)

[3.2 Metode de dezvoltare 4](#_Toc160527846)

[3.3 Strategii de arhitectură 4](#_Toc160527847)

[4. Arhitectura Sistemului și Proiectarea Arhitecturii 6](#_Toc160527848)

[4.1 Vedere logică 6](#_Toc160527849)

[4.2 Arhitectură hardware 6](#_Toc160527850)

[4.3 Arhitectură software 6](#_Toc160527851)

[4.4 Arhitectura informațiilor 7](#_Toc160527852)

[4.5 Arhitectura de comunicații interne 7](#_Toc160527853)

[4.6 Diagrama de arhitectură a sistemului 8](#_Toc160527854)

[5. Proiectarea sistemului 9](#_Toc160527855)

[5.1 Proiectarea bazei de date 9](#_Toc160527856)

[5.1.1 Obiecte de date și structuri de date rezultante 9](#_Toc160527857)

[5.1.2 Fișiere și baze de date 9](#_Toc160527858)

[5.2 Conversii de date 9](#_Toc160527859)

[5.3 Interfețe utilizator 10](#_Toc160527860)

[5.3.1 Intrări 10](#_Toc160527861)

[5.3.2 Ieșiri 10](#_Toc160527862)

[5.4 Proiectarea interfețelor cu utilizatorul 10](#_Toc160527863)

[6. Scenarii de utilizare 11](#_Toc160527864)

[7. Proiectare de detaliu 12](#_Toc160527865)

[7.1 Proiectare hardware de detaliu 12](#_Toc160527866)

[7.2 Proiectare software de deatliu 12](#_Toc160527867)

[7.3 Proiectare detaliată de securitate 13](#_Toc160527868)

[7.4 Proiectare de detaliu pentru performanța sistemului 14](#_Toc160527869)

[7.5 Proiectare detaliată a comunicațiilor interne (între componente) 14](#_Toc160527870)

[8. Controale pentru verificarea integrității sistemului 15](#_Toc160527871)

[Anexa A: Gestiunea modificărilor documentului 16](#_Toc160527872)

[Anexa B: Acronime 17](#_Toc160527873)

[Anexa C Documente la care se face referire 18](#_Toc160527874)

## Introducere

Documentul de Proiectare a Sistemului descrie modul în care cerințele funcționale și non-funcționale înregistrate în Documentul de Cerințe se transformă în specificații de proiectare a sistemului, mai tehnice, pe baza cărora se construiește sistemul. Se documentează proiectarea sistemului la un nivel înalt și specificațiile detaliate de proiectare.

Se descriu obiectivele și considerațiile de proiectare, se furnizează o prezentare generală la nivel înalt a arhitecturii sistemului și se descrie proiectarea datelor asociate cu sistemul, precum și interfața om-mașină și scenariile operaționale.

Proiectarea sistemului la nivel înalt este descompusă în continuare în specificații detaliate de proiectare pentru fiecare componentă a sistemului, inclusiv hardware, comunicațiile interne, software, controalele de integritate a sistemului și interfețele externe.

### Scopul documentului

Documentul are rolul de a ghida procesul de dezvoltare printr-o viziune clară și structurată asupra modului în care sistemul este conceput, implementat și integrat. El servește ca punct de referință atât pentru dezvoltatori, cât și pentru oricare altă parte interesată (ex. coordonatorul proiectului sau evaluatori), oferind o imagine completă asupra aspectelor tehnice și funcționale ale aplicației.

Prin acest livrabil se documentează și urmăresc informațiile necesare pentru a defini eficient arhitectura și designul sistemului, în scopul de a oferi echipei de dezvoltare îndrumare asupra arhitecturii sistemului ce urmează să fie dezvoltat.

Documentele de proiectare sunt produse incremental și iterativ pe parcursul ciclului de viață al dezvoltării sistemului, în funcție de circumstanțele particulare ale proiectului de tehnologie informațională (IT) și de metodologia de dezvoltare a sistemului utilizată.

Publicul țintă este managerul de proiect, echipa de proiect și echipa de dezvoltare. Anumite părți ale acestui document, cum ar fi interfața cu utilizatorul (UI), pot fi împărtășite cu clientul/utilizatorul și cu alte părți interesate ale căror contribuții/aprobări sunt necesare în UI.

## Prezentare generală și abordări de proiectare

Această secțiune descrie principiile și strategiile care vor fi utilizate ca ghiduri în momentul proiectării și implementării sistemului.

### Prezentare generală

GlowApp a fost concepută ca o aplicație web destinată saloanelor de înfrumusețare care doresc să-și digitalizeze procesele de programare, promovare și gestionare a serviciilor. Într-un context în care digitalizarea devine esențială pentru eficiență și vizibilitate, aplicația oferă o soluție centralizată, ușor de utilizat, atât pentru clienți cât și pentru parteneri (saloane).

Proiectarea sistemului a fost organizată în jurul unei arhitecturi **client-server**, cu separarea clară a responsabilităților între front-end și back-end, și o bază de date relațională pentru persistența datelor. Aplicația este modulară, astfel încât să poată fi extinsă ușor cu funcționalități noi în viitor.

### Presupuneri/ Constrângeri/ Riscuri

#### Presupuneri

În elaborarea și implementarea sistemului GlowApp s-au făcut următoarele presupuneri și s-au identificat anumite dependențe care influențează funcționarea aplicației:

* **Infrastructură hardware și software**:
  + Serverul de aplicație rulează într-un mediu compatibil cu **Node.js** și **MySQL** (ex: Linux sau Windows cu WAMP/XAMPP, sau server dedicat cu acces SSH).
  + Utilizatorii finali accesează aplicația printr-un **browser web modern** (ex: Chrome, Firefox, Edge), compatibil cu HTML5 și JavaScript.
* **Sistem de operare**:
  + Nu există o restricție specifică privind sistemul de operare utilizat de client, atâta timp cât acesta poate rula un browser web standard.
* **Stocare fișiere**:
  + Fișierele (imagini încărcate de parteneri) sunt stocate local pe serverul back-end, iar calea către fișier este salvată în baza de date. Se presupune existența unui spațiu de stocare suficient pe server pentru acest scop.
* **Caracteristicile utilizatorilor finali**:
  + Utilizatorii aplicației (atât clienți, cât și parteneri) au un nivel minim de competențe digitale necesare pentru a naviga într-o aplicație web și a folosi funcții precum autentificare, completarea de formulare, navigare prin meniuri, încărcare de fișiere.
  + Partenerii (saloanele) pot gestiona singuri contul și conținutul publicat (servicii, imagini etc.) fără a necesita intervenția unui administrator tehnic.
* **Modificări și extindere**:
  + Se presupune că în versiuni viitoare, sistemul ar putea necesita integrarea cu servicii externe (ex: notificări SMS/email, procesatori de plăți online), astfel arhitectura a fost gândită modular, cu separarea clară a logicii de business de interfața cu utilizatorul.

#### Constrângeri

Proiectarea și dezvoltarea aplicației GlowApp a fost influențată de o serie de constrângeri tehnice și operaționale, care au avut un impact direct asupra arhitecturii sistemului, alegerii tehnologiilor și modului de implementare:

* **Mediu hardware și software**:
  + Aplicația este dezvoltată pentru a funcționa pe servere accesibile și cu resurse limitate, motiv pentru care s-a optat pentru o arhitectură ușoară, bazată pe Node.js și MySQL, cu stocare locală pentru fișiere.
  + Nu se utilizează tehnologii cloud sau microservicii, pentru a reduce complexitatea și costurile de implementare.
* **Mediu utilizator final**:
  + Interfața a trebuit să fie cât mai simplă și intuitivă, deoarece utilizatorii finali (clienți și parteneri din saloane) pot avea un nivel tehnic variabil.
  + Platforma a fost optimizată pentru utilizare pe desktop și tabletă, cu funcționalitate limitată pe mobil, ceea ce poate restrânge accesibilitatea pentru un segment de utilizatori.
* **Disponibilitatea resurselor**:
  + Proiectul este dezvoltat cu resurse limitate (timp, echipă, echipamente), ceea ce a determinat excluderea anumitor funcționalități avansate (ex: sistem de notificări push, integrări externe).
* **Cerințe de securitate și confidențialitate**:
  + Pentru protecția datelor personale, s-au impus restricții legate de autentificare (cu hashing parole), validarea fișierelor încărcate, limitarea accesului la resurse în funcție de rolul utilizatorului și prevenirea accesului neautorizat la fișierele de pe server.
  + De asemenea, aplicația trebuie să respecte bunele practici legate de GDPR (ex: informarea utilizatorilor, protecția datelor sensibile, ștergerea contului).
* **Depozitarea și distribuția datelor**:
  + Imaginile încărcate de parteneri sunt stocate local pe server, ceea ce impune o bună organizare a fișierelor și limitarea dimensiunii acestora pentru a nu afecta performanța generală a aplicației.
  + Nu este prevăzută în această etapă o funcție de backup automat sau distribuție redundantă a datelor.
* **Cerințe de performanță**:
  + Aplicația trebuie să răspundă rapid la interacțiuni (autentificare, căutare servicii, încărcare imagini), chiar și pe conexiuni mai lente.
  + Limitările de performanță pot apărea odată cu creșterea numărului de utilizatori sau de fișiere media stocate, necesitând eventuale optimizări viitoare.
* **Cerințe de verificare și validare**:
  + Sistemul trebuie testat atât funcțional (login, programări, gestionare imagini), cât și la nivel de securitate (acces neautorizat, upload fișiere nepermise).
  + Lipsa unei echipe dedicate de testare impune o verificare atentă din partea dezvoltatorului în toate etapele de implementare.

#### Riscuri

**Risc: Acces neautorizat la datele utilizatorilor**

* *Descriere:* Datele personale ale utilizatorilor (emailuri, imagini, parole) pot fi vulnerabile în lipsa unor măsuri de securitate adecvate.
* *Măsuri de reducere:* Implementarea autentificării cu parole hash-uite (bcrypt), validarea atentă a inputului utilizatorului, restricționarea accesului la rutele de tip admin/partener și aplicarea de politici CORS stricte.

**Risc: Suprasolicitarea serverului sau epuizarea spațiului de stocare**

* *Descriere:* Încărcarea unui număr mare de imagini sau creșterea numărului de utilizatori poate afecta performanța aplicației.
* *Măsuri de reducere:* Limitarea dimensiunii fișierelor încărcate, curățarea periodică a fișierelor nefolosite, arhitectură modulară pentru o eventuală migrare în cloud în versiuni ulterioare.

**Risc: Nealinierea la cerințele legale (ex: GDPR)**

* *Descriere:* Colectarea și stocarea datelor personale trebuie să respecte reglementările privind confidențialitatea.
* *Măsuri de reducere:* Informarea clară a utilizatorilor privind colectarea datelor, opțiuni de ștergere a contului, restricționarea accesului la informații sensibile doar pentru utilizatorii autorizați.

## Considerații de proiectare

Înainte de a construi o arhitectură completă și de a implementa funcționalitățile aplicației, au fost identificate o serie de aspecte esențiale care influențează modul în care sistemul va fi dezvoltat, structurat și extins în viitor. Aceste considerații stau la baza deciziilor de proiectare și asigură coerenta, scalabilitate și securitate în cadrul aplicației GlowApp.

### Obiective și linii directoare (ghiduri)

**Obiective principale ale designului:**

* **Securitate și confidențialitate**: protejarea datelor utilizatorilor (inclusiv a fișierelor media), utilizarea parolelor hash-uite și a validării riguroase a datelor.
* **Ușurință în utilizare**: interfață simplă, intuitivă, cu fluxuri logice pentru toate rolurile (client, partener, admin).
* **Performanță decentă pe infrastructură minimă**: GlowApp trebuie să funcționeze bine chiar și pe servere cu resurse limitate.
* **Extensibilitate**: sistemul trebuie să permită adăugarea de noi funcționalități (plăți, notificări, sistem de review etc.).

**Linii directoare de codare și implementare:**

* **Structurare modulară**: separarea logică a componentelor în backend (auth, users, services, uploads etc.).
* **RESTful API**: rutele sunt organizate conform convențiilor REST pentru claritate și scalabilitate.
* **Cod curat și comentat**: fiecare modul include comentarii pentru ușurința întreținerii.
* **Folosirea de convenții standard de numire**: variabile și funcții denumite clar și consecvent (în engleză).
* **Validare input**: toate datele trimise de utilizator sunt validate atât pe front-end cât și pe back-end.
* **Upload imagini securizat**: verificare extensii și dimensiuni, și denumire unică pentru fișiere salvate local.

### Metode de dezvoltare

S-a adoptat o metodă **structurată și incrementală**, cu influențe de prototipare rapidă. Aplicația a fost dezvoltată în etape, fiecare funcționalitate fiind testată și rafinată după implementare.

**Metodologii și tehnologii folosite:**

* **Orientare pe obiecte** pentru logica serverului și manipularea datelor.
* **UML** a fost utilizat în stadiile inițiale pentru modelarea relațiilor între entități (diagrama ER, diagrama cazuri de utilizare).
* **Prototipare rapidă** pentru interfața utilizator – testarea directă cu utilizatori reali pentru ajustarea UI.
* **Stack tehnologic:**
  + Backend: Node.js cu Express
  + Bază de date: MySQL
  + Stocare locală: fișiere media organizate în directoare pe server
  + Frontend: Aplicație mobilă dezvoltată în React Native

**Contingente posibile și soluții alternative:**

* În cazul în care numărul de imagini sau utilizatori devine prea mare, este prevăzută posibilitatea migrării în cloud (de ex. AWS S3 pentru stocare și Heroku/Vercel pentru deploy).
* Dacă apar nevoi legate de scalabilitate, se poate trece la un sistem modular cu microservicii.
* În lipsa unor integrări externe (ex: plăți, SMS), sistemul este construit inițial fără dependențe externe.

### Strategii de arhitectură

**Arhitectură generală:**

* Aplicația urmează modelul **client-server**, cu backend RESTful și stocare de fișiere locală.
* Backend-ul este împărțit în module (auth, users, appointments, uploads) pentru o mai bună organizare și mentenanță.
* Datele sunt persistate într-o bază relațională MySQL
* Fișierele media sunt stocate pe server, structurate în foldere denumite strategic pentru a fi ccesate prin endpointuri generalizate

**Decizii și justificări:**

* **Node.js cu Express** a fost ales pentru simplitate și rapiditate în dezvoltare, dar și pentru compatibilitatea cu o gamă largă de pachete utile.
* **MySQL** a fost preferat în locul unei baze NoSQL pentru că datele au o structură clară și relațională (ex: utilizatori, servicii, programări).
* **Mecanism de upload cu validare extensii și mărime**: pentru a preveni abuzuri sau încărcări de fișiere neconforme.

**Alternative analizate:**

* **Firebase sau MongoDB**: respinse pentru că modelul de date relațional se potrivea mai bine și nu erau necesare funcționalități în timp real sau nesting complex.
* **Stocare în cloud (Amazon S3, Firebase Storage)**: amânată pentru o versiune viitoare, din considerente de simplitate și costuri.
* **Sisteme de management al fișierelor externe**: respinse pentru a menține controlul local asupra fișierelor și a reduce dependențele externe.

## Arhitectura Sistemului și Proiectarea Arhitecturii

Această secțiune conturează designul arhitecturii sistemului și hardware-ului.

### Vedere logică

Aplicația **GlowApp** este organizată într-o arhitectură client-server, unde funcționalitatea este distribuită între două mari componente: **front-end-ul (React Native)** și **backend-ul (Node.js cu Express)**. Aplicația mobilă interacționează cu backend-ul prin API-uri RESTful, iar datele sunt gestionate și stocate într-o bază de date **MySQL**.

### Arhitectură hardware

Aplicația **GlowApp** utilizează o arhitectură hardware centralizată, în care serverele de aplicație și de baze de date sunt găzduite pe un server dedicat. În plus, stocarea imaginilor este realizată local pe serverul respectiv.

**Componente hardware**:

* **Serverul de aplicație**: Rulează **Node.js** cu Express și gestionează cererile API.
* **Serverul de baze de date**: Gazduiește baza de date **MySQL**, responsabilă cu stocarea și gestionarea datelor.
* **Stocare locală**: Fișierele media (imaginile încărcate de utilizatori și parteneri) sunt stocate într-un sistem de fișiere localizat pe server.
* **Firewall și securitate**: Serverele sunt protejate prin firewalluri care controlează accesul la aplicație și la baza de date.

**Estimarea resurselor hardware**:

* **Procesor**: Serverul de aplicație va necesita un procesor multi-core pentru a gestiona cererile simultane.
* **Memorie**: Serverele vor avea cel puțin 8 GB de RAM pentru a susține funcționarea aplicației și gestionarea mai multor utilizatori simultani.
* **Stocare**: Depozitarea fișierelor media va necesita aproximativ 500 GB de stocare pe disc pentru a gestiona fișierele de mari dimensiuni.

### Arhitectură software

Aplicația **GlowApp** utilizează mai multe componente software pentru a îndeplini funcționalitățile dorite:

* **Frontend**:
  + **React Native** – Utilizat pentru dezvoltarea aplicației mobile cross-platform.
  + **Biblioteci și API-uri externe**: Cum ar fi **React Navigation** pentru navigație și **Axios** pentru efectuarea cererilor HTTP către backend.
* **Backend**:
  + **Node.js + Express** – Servește drept platformă principală pentru gestionarea cererilor API și a logicii aplicației.
  + **MySQL** – Baza de date relațională utilizată pentru stocarea datelor utilizatorilor, partenerilor și altor entități relevante.
  + **Middleware-uri**: Cum ar fi **JWT (JSON Web Tokens)** pentru autentificare și **multer** pentru gestionarea încărcării fișierelor.
* **Alte tehnologii și servicii**:
  + **Cloud storage** (dacă este cazul) – Dacă aplicația se va extinde și va gestiona un volum mare de imagini sau fișiere media, s-ar putea folosi o soluție cloud pentru stocare (ex: AWS S3).

**Interfețe și API-uri**:

* **API RESTful**: Backend-ul expune o serie de endpointuri care permit aplicației mobile să interacționeze cu baza de date și să preia/stocheze date.
* **Autentificare și autorizare**: Se folosesc tokenuri JWT pentru securizarea endpointurilor API.

### Arhitectura informațiilor

Informațiile gestionate de **GlowApp** includ:

* **Informații despre utilizatori**: Datele de autentificare, detaliile contului și activitățile utilizatorului în aplicație.
* **Informații despre parteneri**: Detalii despre saloane și servicii oferite.
* **Imagini și fișiere media**: Fotografii încărcate de utilizatori și parteneri, stocate pe server sau pe platforme de stocare externă.
* **Informații de plată** (dacă este cazul): Date despre tranzacțiile realizate prin aplicație.

**Informații sensibile**:

* Datele personale ale utilizatorilor, inclusiv informațiile de autentificare, vor fi stocate criptate.
* Informațiile financiare și de plată vor fi gestionate conform normelor de securitate (ex: PCI-DSS pentru plăți).

### Arhitectura de comunicații interne

Comunicarea între componentele aplicației se realizează printr-o arhitectură de rețea internă, utilizând protocoale HTTP pentru cererile API și FTP/SFTP pentru transferul fișierelor media.

**Echipamente necesare**:

* **Routere și comutatoare**: Pentru a asigura conectivitatea între servere și utilizatori.
* **Firewall-uri**: Pentru a proteja infrastructura de accesul neautorizat.
* **Sisteme de backup**: Pentru a preveni pierderea datelor.

**Estimarea resurselor rețelei**:

* **Lățimea de bandă**: Este necesar un minim de 1 Gbps pentru a asigura performanța optimă în gestionarea cererilor de la utilizatori și a fișierelor media.
* **Lățimea de bandă pentru stocare**: Sistemul de stocare ar trebui să aibă o lățime de bandă de cel puțin 500 Mbps pentru a asigura transferul rapid al fișierelor media.

### Diagrama de arhitectură a sistemului

A diagram of a server system

AI-generated content may be incorrect.

## Proiectarea sistemului

### Proiectarea bazei de date

Baza de date a fost proiectată în MySQL, utilizând o arhitectură relațională clasică. Datele sunt normalizate pentru a evita redundanța și pentru a asigura integritatea referențială între entități. Sistemul gestionează date esențiale legate de utilizatori, saloane, programări, servicii, portofolii și fișiere media.

#### Obiecte de date și structuri de date rezultante

Obiectele funcționale principale includ:

Obiect de date

Utilizator Reprezintă utilizatori finali (clienți) și parteneri (saloane)

Programare Stochează programările efectuate de utilizatori

Serviciu Conține toate serviciile oferite de parteneri

Portofoliu Include poze și descrieri din portofoliul unui partener

Imagine Reține calea fișierelor media stocate local

Review Păstrează recenziile utilizatorilor despre parteneri

Toate tabelele au chei primare (id) și relații de tip foreign key pentru a lega entitățile. De exemplu, appointments.user\_id este FK spre users.id, iar portfolio.partner\_id spre users.id (cu rol de partener).

#### Fișiere și baze de date

Sistemul utilizează o bază de date relațională MySQL pentru gestionarea datelor structurate, iar pentru fișierele media (ex: imagini din portofolii) se utilizează un sistem de fișiere local organizat ierarhic. Această combinație permite o separare clară între datele tranzacționale și conținutul static.

##### Baze de date

Baza de date MySQL conține tabele relaționale care acoperă entitățile principale ale aplicației: utilizatori, parteneri, programări, servicii, recenzii, portofolii și imagini. Fiecare entitate este gestionată printr-un tabel dedicat, cu relații între ele realizate prin chei externe. De exemplu, programările fac legătura între utilizatori, parteneri și servicii, iar recenziile sunt asociate cu utilizatorii și partenerii. Fiecare tabel conține un identificator unic (id), atribute descriptive relevante (ex: nume, descriere, email, dată, status), precum și timestampuri pentru creare și actualizare.

Datele sunt normalizate pentru a reduce redundanța și pentru a permite interogări eficiente. Integritatea referențială este asigurată prin utilizarea constrângerilor de tip foreign key. Pentru performanță, sunt definite indexuri pe coloanele cele mai frecvent utilizate în interogări, precum emailul utilizatorului, data programării sau categoria serviciului. Backupul bazei de date este realizat zilnic, cu salvarea fișierelor de tip dump SQL și păstrarea acestora într-un spațiu securizat.

##### Fișiere non-DBMS

Pe lângă datele relaționale, sistemul stochează imagini încărcate de parteneri în cadrul portofoliilor. Aceste fișiere sunt stocate local, pe server, într-un director denumit uploads/, cu o structură ierarhică organizată în funcție de ID-ul partenerului. Fiecare fișier are un nume unic generat automat, pentru a evita coliziunile și a permite referințierea facilă.

Fișierele media sunt utilizate exclusiv pentru ieșire (afișare în aplicație) și sunt accesate prin URL-uri generate automat, de forma /uploads/<nume\_fisier>.jpg. În momentul încărcării, metadatele fișierului (cale, dată încărcare) sunt înregistrate într-o bază de date, în tabelul images, iar fișierul este asociat ulterior cu portofoliul unui partener. Accesul la aceste fișiere este controlat prin logica aplicației, iar încărcarea se face exclusiv prin endpointuri autentificate.

Dimensiunea fișierelor este în general cuprinsă între 500 KB și 2 MB, iar volumul estimat de fișiere este de aproximativ 10.000 în primul an de funcționare, ceea ce implică un spațiu de stocare necesar de aproximativ 20 GB. Sistemul include o rutină de backup automat pentru aceste fișiere, sincronizată cu backupul bazei de date, și oferă posibilitatea de restaurare rapidă în caz de pierdere sau deteriorare a datelor.

Actualizarea acestor fișiere este rară (în general doar la adăugare sau ștergere), iar citirea lor este frecventă, în special în paginile publice ale aplicației. Fișierele sunt statice și nu necesită procesare adițională la runtime, ceea ce le face ideale pentru servire rapidă din cache sau CDN în versiuni ulterioare ale aplicației.

### Conversii de date

În cadrul proiectului, nu sunt prevăzute conversii majore de date provenite din sisteme existente, întrucât aplicația este dezvoltată de la zero. Cu toate acestea, pot apărea situații punctuale în care partenerii existenți (saloane sau profesioniști care au deja date în alt format) doresc importul manual al unor informații precum liste de servicii, imagini sau programări anterioare.

Pentru aceste cazuri, sistemul oferă posibilitatea efectuării de conversii personalizate, utilizând fișiere CSV sau Excel furnizate de parteneri. Aceste fișiere pot include informații precum: denumiri de servicii, prețuri, descrieri, program de lucru și date despre clienți. Conversia acestor date se face manual sau semi-automat, printr-un script de import validat de echipa tehnică.

Datele importate sunt supuse validării înainte de a fi persistate în baza de date: sunt verificate tipurile de câmpuri, valorile permise, structura documentului sursă și potențialele duplicate. Fișierele-sursă folosite în procesul de import sunt stocate temporar și șterse automat după procesare, pentru a proteja confidențialitatea și a evita stocarea inutilă.

Pentru viitor, se are în vedere dezvoltarea unui modul de import configurabil direct din interfața partenerului, care să permită încărcarea fișierelor cu date preformatate și maparea câmpurilor direct în aplicație.

### Interfețe utilizator

Aplicația este proiectată pentru a deservi mai multe clase de utilizatori, fiecare având un set specific de responsabilități și interacțiuni cu sistemul. Clasele principale de utilizatori sunt:

* **Clienți (utilizatori finali):** persoane care accesează aplicația pentru a căuta saloane sau profesioniști, a vizualiza servicii, a face programări și a lăsa recenzii.
* **Parteneri (saloane sau profesioniști individuali):** utilizatori care oferă servicii și își administrează profilul, serviciile, programările și comunicarea cu clienții.

Pentru fiecare dintre aceste clase de utilizatori, aplicația oferă interfețe personalizate și niveluri de acces corespunzătoare.

#### Intrări

Clienții interacționează cu aplicația prin ecrane mobile care permit:

* căutarea rapidă de servicii și saloane (prin cuvinte cheie, locație sau filtre),
* completarea formularelor de programare (data, ora, serviciul dorit, opțional observații),
* autentificare (cu email și parolă sau login social),
* completarea profilului personal (nume, email, număr de telefon etc.).

Partenerii introduc date prin:

* formulare de adăugare/editare servicii,
* calendar interactiv pentru gestionarea programărilor,
* formulare pentru adăugarea fotografiilor de portofoliu,
* setări generale ale contului (program, politici de anulare etc.).

Administrarea aplicației implică interfețe web pentru:

* gestionarea utilizatorilor,
* revizuirea de conținut (recenzii, fotografii),
* rularea de rapoarte și vizualizarea indicatorilor.

Toate elementele de date sunt validate la nivel de client și server. Se folosesc reguli precum:

* câmpuri obligatorii (ex: email, nume),
* validări de tip (ex: adrese de email, numere de telefon),
* limite de lungime și formate predefinite (ex: pentru parole sau coduri poștale),
* controale de tip dropdown, calendare și autocomplete pentru evitarea erorilor.

#### Ieșiri

Ieșirile generate de sistem includ:

* **pentru clienți:** confirmări de programare, notificări (push și email), afișarea programărilor viitoare și istoricul serviciilor;
* **pentru parteneri:** liste detaliate cu programările primite, statistici privind activitatea (număr clienți, recenzii, venituri estimate), alerte în caz de anulări sau modificări;
* **pentru administratori:** rapoarte de activitate, log-uri de sistem, notificări de utilizatori raportați sau activitate suspectă.

Toate ieșirile sunt prezentate în interfețe prietenoase, optimizate pentru mobil (pentru clienți și parteneri) și desktop (pentru administratori), cu respectarea regulilor de acces și autentificare. Conținutul este actualizat în timp real, acolo unde este necesar (de exemplu, programările disponibile).

### Proiectarea interfețelor cu utilizatorul

Interfețele aplicației sunt proiectate cu accent pe simplitate, accesibilitate și ușurință în utilizare, adaptate pentru fiecare clasă de utilizatori. Aplicația este dezvoltată folosind React Native pentru partea mobilă și React pentru interfața de administrare, cu un design responsive și intuitiv.

Interfața client (mobil)

* Ecran principal: afișează o listă cu saloane/parteneri recomandați, filtre de căutare și bannere promoționale.
* Căutare: bară de căutare rapidă, cu autocomplete și filtre (categorie serviciu, locație, preț, rating).
* Profil salon/profesionist: descriere, portofoliu foto, recenzii, buton de programare.
* Programare: formular cu selecție de serviciu, dată, oră, date de contact.
* Cont utilizator: vizualizare programări viitoare, istoric servicii, date personale, opțiuni notificări.

Interfața partener (mobil)

* Dashboard: sumar al programărilor zilei, notificări importante, statistici rapide.
* Gestionare servicii: adăugare, modificare, ștergere servicii oferite.
* Calendar: interfață interactivă pentru vizualizarea și modificarea programărilor.
* Galerie portofoliu: adăugare și organizare imagini.
* Setări cont: date business, program de lucru, politici.

## Scenarii de utilizare

## Proiectare de detaliu

Această secțiune oferă toate informațiile necesare echipei de dezvoltare pentru a construi, integra și implementa atât componentele hardware, cât și cele software, într-un sistem funcțional coerent.

### Proiectare hardware de detaliu

Sistemul necesită un mediu server care să ruleze aplicația backend dezvoltată în Node.js, cu suport pentru MySQL și stocare locală a imaginilor. Specificațiile hardware minime sunt următoarele:

* Procesor: minim 2.6 GHz, quad-core
* RAM: cel puțin 8 GB
* SSD: minim 100 GB spațiu disponibil pentru aplicație și baze de date
* Placă de rețea Gigabit Ethernet
* Sursă de alimentare compatibilă cu cerințele serverului

Pentru desfășurarea aplicației mobile (frontend React Native), dispozitivele client sunt telefoane Android sau iOS cu specificații moderne (minim Android 8 / iOS 13).

Imaginile sunt stocate local pe serverul backend, într-un director protejat, și sunt accesate prin linkuri relative generate din aplicație. Nu sunt necesare componente hardware personalizate; toate echipamentele pot fi achiziționate de la furnizori consacrați (ex: Dell, HP, Lenovo) sau din infrastructura de hosting deja existentă.

### Proiectare software de detaliu

**Serviciul de autentificare**

* Clasificare: serviciu API (backend)
* Definiție: permite autentificarea utilizatorilor și partenerilor în aplicație, folosind email și parolă
* Cerințe: validare date, generare token JWT
* Structuri de date: utilizatori (ID, email, parolă hash-uită, rol, token-uri active)
* Constrângeri: parolă minim 8 caractere, token valid 24h, protecție CSRF
* Compoziție: modul de criptare parole, generator JWT
* Interacțiuni: interacționează cu serviciul de utilizatori și cel de notificări
* Procesare: verifică credențialele, compară hash, generează token JWT
* Interfețe: endpoint POST /api/auth/login, exportă funcția authenticateUser

**Serviciul de programări**

* Clasificare: serviciu API (backend)
* Definiție: gestionează rezervările pentru servicii
* Cerințe: creare, modificare, anulare și listare programări
* Structuri de date: programări (ID, userId, salonId, serviciu, dată, oră, status)
* Constrângeri: verificare conflict programări, reguli orare
* Compoziție: modul calendar, validare orar partener
* Interacțiuni: folosește serviciul de utilizatori și servicii, trimite notificări
* Procesare: verifică disponibilitatea, salvează rezervarea

Toate serviciile sunt implementate în Node.js. Codul este documentat și modularizat, fiecare modul având comentarii explicative în antet.

### Proiectare detaliată de securitate

Aplicația implementează următoarele mecanisme de securitate:

* Autentificare bazată pe token JWT, protejat cu semnătură HMAC
* Autorizare pe bază de roluri (utilizator, partener, admin)
* Loguri de acces și acțiuni importante stocate în baza de date
* Criptare parole cu bcrypt, salt de 10 runde
* Porturi: 443 (HTTPS) și 80 (pentru redirect HTTPS)
* Prevenirea intruziunilor prin validare input și rate-limiting la login

### Proiectare de detaliu pentru performanța sistemului

Sistemul este optimizat pentru un volum de aproximativ 10.000 utilizatori activi și câteva sute de programări pe zi. Așteptările de performanță vizează:

* Timp de răspuns API: <500ms
* Disponibilitate: 99.5% uptime lunar
* Backup zilnic al bazei de date și arhivare săptămânală
* Replicare opțională în mediu de producție pentru disponibilitate ridicată

Singurul punct de eșec este serverul backend, care poate fi replicat sau mutat în cloud (AWS, Azure) pentru scalabilitate și redundanță.

### Proiectare detaliată a comunicațiilor interne (între componente)

Arhitectura presupune comunicarea între frontend (React Native) și backend (Node.js) prin HTTPS. Backendul comunică local cu serverul MySQL.

Specificațiile de comunicare includ:

* Formatul datelor: JSON
* Număr de servere: 1 server backend, mai mulți clienți (telefoane mobile)
* Topologia rețelei: client-server prin internet
* Sincronizarea nu este necesară, se folosesc cereri asincrone HTTP

Conectivitatea internă între module este realizată prin apeluri de funcții interne în Node.js, iar între aplicație și baza de date, prin MySQL driver nativ.

## Controale pentru verificarea integrității sistemului

Această secțiune descrie pașii necesari pentru instalarea și implementarea sistemului, atât din punct de vedere software, cât și hardware. Sunt incluse cerințele pentru configurare inițială, mediul de instalare, procesul de implementare, și procedurile post-implementare necesare pentru punerea în funcțiune a sistemului.

1. **Cerințe de mediu pentru instalare**

Pentru backend:

* Sistem de operare: Ubuntu 20.04 LTS sau o altă distribuție Linux stabilă
* Node.js versiunea 18.x
* MySQL versiunea 8.x
* Acces la internet pentru actualizări și instalarea pachetelor
* Spațiu de stocare disponibil: minim 100 GB SSD
* RAM recomandată: 8 GB
* Porturi deschise: 443 (HTTPS), 80 (HTTP), 3306 (MySQL – local only)

Pentru frontend (aplicație mobilă):

* Instalare pe dispozitive mobile (Android 8+ sau iOS 13+)
* Acces la App Store / Google Play pentru distribuirea aplicației
* Configurare pentru notificări push (Firebase Cloud Messaging)

1. **Configurare inițială**

**Pentru server:**

1. Instalarea și configurarea Node.js și MySQL
2. Clonarea codului sursă din repository-ul aplicației
3. Configurarea fișierului .env cu datele de conexiune la baza de date și cheile de securitate
4. Rularea scripturilor de inițializare a bazei de date
5. Crearea folderelor necesare pentru stocarea imaginilor
6. Configurarea unui reverse proxy (ex: Nginx) și certificat SSL

**Pentru aplicația mobilă:**

1. Configurarea variabilelor de mediu în fișierul .env pentru legătura cu API-ul
2. Generarea aplicației cu react-native build pentru Android/iOS
3. Testarea locală pe emulator/dispozitiv
4. Publicarea aplicației în Google Play / App Store
5. **Procesul de instalare și implementare**
6. Instalarea backendului pe serverul de producție
7. Importul datelor inițiale (categorii servicii, parteneri de test, utilizatori admin)
8. Implementarea aplicației mobile pe platformele respective
9. Testare end-to-end (login, programări, upload imagini, notificări)
10. Lansare oficială

**Considerații post-implementare**

* Monitorizarea performanței serverului și a bazei de date
* Activarea sistemului de backup automat (zilnic)
* Configurarea unui sistem de alertare pentru erori server
* Oferirea de suport tehnic pentru utilizatori în perioada de lansare
* Posibilitatea scalării backendului în cloud, în funcție de creșterea numărului de utilizatori

Anexa A: Gestiunea modificărilor documentului

Tabel 1 – Înregistrarea modificărilor asupreaa documentului curent

| versiune | Data | Autorul/Deținătorul | Descriere |
| --- | --- | --- | --- |
| v0.1 | 08.04.2025 | Boșnegeanu Adrian-Gabriel | Completarea inițială a documentului cu descrierea versiunii mobile pre-alpha |

Anexa B: Acronime

Tabel 2 - Acronime

| Acronim | Forma completă |
| --- | --- |
| UI | User Interface (Interfață cu utilizatorul) |
| UX | User Experience (Experiența utilizatorului) |
| DB | Bază de date |
| API | Aplicațion Programing Interface |
| BE | Back-End (Serverul Node.js) |
| FE | Front-End (Aplicația GlowApp din dispozitivul utilizatorului) |
| ID | Identificator Unic |
| PK | Primary Key (Cheie Primară) |
| FK | Foreign Key (Cheie Externă) |
| .js | Limbajul de programare JavaScript |
| SQL | Structured Querry Language |
| CSS | Cascading Style Sheets |

Anexa C Documente la care se face referire

Tabel 3 – Documente la care se facce referire

| Nume document | Locație sau URL | Dată emitere document |
| --- | --- | --- |
| Documentul de specificare a cerințelor  Software Requirements Specification  (SRS) Document | GitHub - GlowApp | 08.04.2025 |