**Sistem informatic de recomandare pentru o platformă de cursuri bazat pe inteligența artificială**

# 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

**Nume proiect:** **LearnJy** – Platformă Online de Cursuri cu Inteligență Artificială  
**Acronim:** **LearnJy** **-LMS** (Learning Management System)

**LearnJy** este o platformă modernă de învățare online, integrată cu un sistem avansat de inteligență artificială (AI) pentru personalizarea experienței utilizatorilor. Aceasta permite adaptarea dinamică a conținutului educațional în funcție de progresul cursanților, oferind recomandări inteligente și suport interactiv.

**Sistem existent și sistem propus:**

* **Sistem existent:** Diverse platforme de e-learning fără funcționalități AI avansate.
* **Sistem propus:** Un sistem LMS inteligent, care integrează algoritmi AI pentru personalizarea traseului educațional, analiza performanței și feedback automatizat.

**Evoluția așteptată a documentului:**  
Acest document va fi actualizat pe măsură ce proiectul avansează, reflectând schimbările în arhitectură, integrarea de noi tehnologii și ajustările în funcție de cerințele utilizatorilor.

### Considerații de Securitate și Confidențialitate

**LearnJy** gestionează date sensibile, inclusiv:

* Informații personale ale cursanților și instructorilor
* Progresul academic și date de performanță
* Interacțiuni și preferințe de învățare

Pentru protejarea acestor date, sistemul va implementa măsuri precum:

* **Criptarea datelor** (atât în tranzit, cât și în repaus)
* **Politici stricte de acces și permisiuni** pentru protejarea informațiilor
* **Conformitate cu reglementările GDPR și alte standarde de securitate**

Aceste măsuri vor asigura confidențialitatea și integritatea datelor utilizatorilor, prevenind accesul neautorizat sau utilizarea abuzivă.

### Scopul documentului

Acest document are rolul de a defini și structura arhitectura și designul sistemului **LearnJy**, o platformă online de cursuri bazată pe inteligență artificială. Documentul furnizează echipei de dezvoltare îndrumări clare privind proiectarea componentelor software și hardware, precum și modul în care cerințele funcționale și non-funcționale sunt transpuse în soluții tehnice concrete.

Documentul de proiectare este elaborat incremental și iterativ pe parcursul ciclului de viață al dezvoltării, adaptându-se la schimbările de cerințe, tehnologie și metodologie de dezvoltare utilizată. Acesta servește ca referință principală pentru toate aspectele arhitecturale și tehnice ale sistemului.

### ****Publicul Țintă****

Documentul este destinat următoarelor părți interesate:

* **Managerul de proiect** – pentru supravegherea implementării și alinierii cu obiectivele strategice;
* **Echipa de proiect și dezvoltare** – pentru implementarea și testarea soluției conform specificațiilor tehnice;
* **Designerii UI/UX** – pentru definirea interfeței utilizatorului și experienței de învățare;
* **Clientul/Utilizatorii finali** – anumite secțiuni, precum designul UI, pot fi partajate pentru feedback și aprobare.

Prin această documentație, se asigură coerența în dezvoltarea platformei **LearnJy**, facilitând colaborarea eficientă între echipe și livrarea unui produs final de calitate.

## Prezentare generală și abordări de proiectare

Această secțiune descrie principiile și strategiile utilizate pentru proiectarea și implementarea sistemului **Learnjy**, având ca scop crearea unei platforme moderne, scalabile și eficiente, care utilizează inteligența artificială pentru a îmbunătăți experiența utilizatorilor.

### ****2.1 Prezentare Generală****

**Learnjy** este o platformă de e-learning bazată pe inteligență artificială, proiectată pentru a oferi o experiență de învățare personalizată și interactivă. Sistemul va include funcționalități avansate precum:

* **Recomandări AI** pentru cursuri, bazate pe progresul utilizatorilor și preferințele lor.
* **Evaluări automate** și feedback inteligent.
* **Analiză avansată a performanței cursanților.**
* **Gamificare și interacțiune socială** pentru o experiență de învățare captivantă.

### 2.2 Presupuneri/ Constrângeri/ Riscuri

**2.2.1 Presupuneri**

* Utilizatorii finali vor avea acces constant și suficient la internet pentru utilizarea optimă a platformei.
* Majoritatea utilizatorilor vor accesa sistemul de pe dispozitive personale (PC, laptop, smartphone).
* Platforma se va baza exclusiv pe tehnologii cloud, presupunându-se că serviciile cloud alese (AWS) vor fi disponibile constant.
* Utilizatorii vor avea niveluri variate de cunoștințe digitale, deci presupunem că interfața și materialele educaționale sunt suficient de intuitive.
* Algoritmii AI vor beneficia de suficiente date istorice de utilizare pentru a genera recomandări personalizate eficiente.

### ****2.2.2 Constrângeri****

În proiectarea și implementarea sistemului **Learnjy**, există o serie de constrângeri tehnice și operaționale care influențează designul hardware-ului, software-ului și comunicațiilor sistemului. Aceste limitări trebuie luate în considerare pentru a asigura funcționarea optimă a platformei.

#### ****Mediu hardware și software****

* Platforma trebuie să fie compatibilă cu **majoritatea browserelor moderne** (Chrome, Firefox, Edge, Safari).
* Sistemul trebuie să fie **optimizat pentru dispozitive mobile** (iOS și Android).
* Resursele de calcul pentru AI trebuie să fie **scalabile**, ceea ce impune utilizarea unor infrastructuri cloud precum AWS, Azure sau Google Cloud.
* Performanța platformei trebuie să rămână stabilă chiar și în condiții de **încărcare mare** (ex. sesiuni simultane ale utilizatorilor).

#### ****Mediu utilizator final****

* Platforma trebuie să fie **ușor accesibilă** pentru utilizatori cu diferite niveluri de competență digitală.
* Necesitatea unui **design UI/UX intuitiv**, cu suport pentru multiple limbi și un mod accesibil pentru utilizatorii cu dizabilități.
* Accesul la internet poate varia, ceea ce necesită optimizarea **încărcării resurselor** pentru conexiuni slabe.

#### ****2.2.3**** Riscuri

* **Scalabilitate insuficientă** – Creșterea rapidă a numărului de utilizatori poate afecta performanța platformei.
* **Securitatea datelor** – Posibile atacuri cibernetice sau breșe de securitate care compromit informațiile utilizatorilor.
* **Conformitate GDPR** – Riscul de neconformitate cu reglementările privind protecția datelor personale.
* **Disponibilitatea AI** – Probleme în funcționarea algoritmilor de inteligență artificială din cauza lipsei de date relevante.
* **Performanța rețelei** – Probleme de latență sau întreruperi ale conexiunii care afectează experiența utilizatorilor. la presupuneri ce pot adauga

### ****3. Considerații de Proiectare****

Această secțiune descrie problemele critice care trebuie abordate înainte de definirea completă a soluției pentru platforma **Learnjy**, incluzând obiectivele, metodele de dezvoltare și strategiile arhitecturale.

### ****3.1 Obiective și Linii Directoare****

Pentru dezvoltarea **Learnjy**, sunt stabilite următoarele obiective și principii de design:

#### ****Obiective principale****

* **Scalabilitate ridicată** – Sistemul trebuie să suporte un număr mare de utilizatori simultan.
* **Personalizare prin AI** – Utilizarea algoritmilor de inteligență artificială pentru recomandări și feedback personalizat.
* **Accesibilitate și compatibilitate** – Platforma trebuie să fie ușor accesibilă pe web și dispozitive mobile.
* **Securitate și conformitate** – Implementarea măsurilor de securitate pentru protecția datelor utilizatorilor
* **Experiență utilizator (UX) optimizată** – Interfață modernă, intuitivă, ușor de utilizat.

### ****3.2 Metode de Dezvoltare****

Pentru designul și dezvoltarea sistemului software LearnJy au fost adoptate următoarele metode și abordări specifice:

* **Arhitectură bazată pe microservicii (FastAPI)** – Această abordare a fost aleasă datorită capacității excelente de modularizare, scalabilitate și întreținere ușoară. Fiecare serviciu rulează independent și poate fi dezvoltat separat, ceea ce facilitează dezvoltarea în echipă și gestionarea ușoară a complexității sistemului.
* **Proiectare bazată pe componente (React)** – Pentru frontend a fost selectat framework-ul React datorită ușurinței în construirea de interfețe modulare, interactive și performante. Această abordare permite reutilizarea eficientă a componentelor, scurtând timpul de dezvoltare și îmbunătățind coerența vizuală și funcțională.

### Contingențe și soluții alternative:

* În cazul în care apar probleme de performanță sau disponibilitate cu furnizorii cloud (AWS), există opțiunea migrării parțiale sau integrale a serviciilor către furnizori alternativi precum Google Cloud sau Microsoft Azure.
* Dacă algoritmii AI inițiali nu livrează rezultate optime, există posibilitatea implementării unor modele alternative mai simple, bazate pe reguli sau recomandări colaborative.

Aceste metode au fost alese după analizarea atentă a cerințelor proiectului, oferind flexibilitatea și robustețea necesară pentru succesul implementării platformei LearnJy.

### ****3.3 Strategii de Arhitectură****

#### ****Arhitectura generală****

Learnjy va adopta o **arhitectură bazată pe microservicii**, unde fiecare componentă rulează independent Iată o versiune îmbunătățită și adaptată clar la contextul tău actual:

* **Utilizarea FastAPI pentru backend (Python)**- FastAPI a fost ales datorită performanței ridicate, documentării excelente (OpenAPI), și compatibilității native cu Python, limbaj popular și matur pentru aplicații AI și backend.
* **React (Next.js) pentru Frontend**- Framework-ul Next.js (bazat pe React) a fost selectat datorită capabilității de server-side rendering (SEO optimizat), componentizării ușoare și performanței ridicate. Alternativele analizate (Angular, Vue.js) au fost respinse din cauza complexității inițiale (Angular) și popularității mai reduse în contextul proiectului actual (Vue.js).
* **Neo4j ca bază de date**- Neo4j permite modelarea eficientă și flexibilă a relațiilor complexe între utilizatori, cursuri și recomandări. Alte alternative precum PostgreSQL sau MongoDB au fost luate în calcul, însă au fost respinse datorită dificultăților în gestionarea relațiilor complexe și recomandărilor bazate pe graf.
* **Utilizarea serviciului AWS S3 pentru stocare**- Stocarea datelor multimedia (imagini, documente cursuri) se face pe AWS S3 datorită fiabilității, scalabilității și ușurinței integrării cu infrastructura cloud existentă. Alternativa unei soluții on-premise a fost exclusă din cauza complexității de gestionare și a costurilor de întreținere.
* **Engine AI bazat pe TensorFlow/PyTorch**- Algoritmii de inteligență artificială folosesc framework-uri populare și robuste (TensorFlow/PyTorch), ceea ce permite adaptabilitate și performanță ridicată în recomandări. Se preferă TensorFlow datorită documentației ample și suportului comunitar, iar PyTorch ca soluție alternativă în caz de necesitate specifică.
* **Strategii de comunicare și sincronizare**- Comunicarea între microservicii și frontend-backend se face prin API-uri REST standardizate, completate de WebSockets pentru notificări și actualizări în timp real. Această alegere simplifică managementul comunicării și oferă flexibilitate pentru extindere ulterioară.

Aceste decizii și strategii au fost selectate pentru a atinge obiectivele de performanță, scalabilitate, securitate și flexibilitate stabilite inițial, menținând în același timp complexitatea la un nivel gestionabil și adaptabil.

### 4 Arhitectura Sistemului și Proiectarea Arhitecturii

LearnJy utilizează o arhitectură bazată pe microservicii, în care fiecare componentă a fost separată logic pentru a oferi modularitate și scalabilitate maximă. Sistemul este proiectat să comunice prin intermediul API-urilor REST standardizate și al WebSocket-urilor pentru comunicarea în timp real.

### 4.1 Vedere Logică

Arhitectura logică a LearnJy este împărțită astfel încât fiecare componentă îndeplinește responsabilități clare:

* **Layer-ul de prezentare (Frontend)** comunică prin intermediul unor apeluri REST către **layer-ul aplicație (Backend)**.
* Backend-ul procesează aceste solicitări, interacționează cu **AI Engine** pentru recomandări și comunică cu **baza de date Neo4j** pentru a gestiona datele necesare.
* Datele multimedia sunt stocate și gestionate prin AWS S3, fiind accesate de frontend prin link-uri securizate generate de backend.

### 4.2 Arhitectură Hardware

Sistemul LearnJy este implementat integral în cloud, folosind serviciile Amazon Web Services (AWS):

* Servere de aplicație și API gestionate de Amazon EC2.
* Baza de date (Neo4j) găzduită pe instanțe dedicate.
* Stocare multimedia asigurată prin AWS S3.

Deoarece sistemul este 100% cloud-based, infrastructura hardware propriu-zisă este abstractizată și nu implică dispozitive fizice dedicate.

### 4.3 Arhitectură Software

**Elemente logice și tehnologii folosite:**

* **Frontend:**
  + React (Next.js) – interfața grafică și server-side rendering (SSR).
  + Axios – pentru apeluri HTTP REST.
  + WebSockets – notificări live și actualizări.
* **Backend:**
  + FastAPI (Python) – API REST performant.
  + JWT – autentificare și autorizare.
  + Pytest – testare automată.
* **Baza de date:**
  + Neo4j – gestionare relațională și recomandări bazate pe graf.
* **AI Engine:**
  + TensorFlow/PyTorch – machine learning pentru recomandări.
* **Stocare:**
  + AWS S3 – gestionare fișiere multimedia.

**Justificări ale alegerilor tehnologice:**

* React a fost ales pentru flexibilitatea și viteza sa în dezvoltarea interfețelor interactive.
* FastAPI a fost preferat datorită performanței excelente, suportului pentru tipuri, documentării automate OpenAPI și ușurinței integrării cu bibliotecile de AI din Python.
* Neo4j oferă capacități excelente pentru gestionarea datelor complexe, relaționale, ceea ce îl face ideal pentru un sistem de recomandare.

### 4.4 Arhitectura Informațiilor

**Date stocate și gestionate în sistem:**

* **Utilizatori:** date personale (nume, email, parolă criptată, progres academic, preferințe de învățare). Date sensibile gestionate conform standardului GDPR.
* **Cursuri:** date despre conținut (texte, video-uri, imagini), evaluări, rezultate teste, feedback utilizatori.
* **Date financiare:** gestionate extern prin Stripe (carduri, tranzacții – date criptate extern).

### 4.5 Arhitectura Comunicațiilor Interne

Sistemul LearnJy folosește două tipuri de comunicație între componente:

* **REST API (HTTP/HTTPS):**  
  Folosit pentru majoritatea interacțiunilor între frontend și backend, backend și baza de date.
* **WebSockets:**  
  Utilizat pentru notificări și actualizări în timp real (exemplu: mesaje interne, progresul în timp real al utilizatorilor).

**Echipamente și infrastructură:**

* Abstractizată și gestionată prin servicii AWS (fără echipamente hardware dedicate fizic).

**Diagrama simplificată a comunicațiilor:**

A diagram of a web site

Description automatically generated

Figura 1. Diagrama de colaborare

**5. Proiectarea sistemului**

### 5.1 Proiectarea bazei de date

Sistemul LearnJy utilizează Neo4j ca bază de date de tip graf, ideală pentru gestionarea relațiilor complexe dintre entitățile aplicației.

#### 5.1.1 Obiecte de date și structuri de date rezultante:

A screenshot of a computer

Description automatically generated

Figura 2. Baza de date din Neo4j

Principalele obiecte de date și relațiile dintre acestea sunt:

* **Utilizator**: Date personale (nume, email, parola criptată, rol utilizator).
  + Relații: INROLAT în cursuri, ARE\_PROGRESUL pentru lecții, TRIMITE și PRIMESTE mesaje, A\_SCRIS recenzii.
* **Curs**: Date despre cursuri (titlu, descriere, data început, data final, status curs).
  + Relații: ARE\_CERTIFICAT, FACE\_PARTE\_DIN o categorie, CONTINE lecții.
* **Lectie**: Conținut educațional detaliat (titlu, conținut lecție).
  + Relații: CONTINE teste, PENTRU\_LECTIA există progresul utilizatorului.
* **Test**: Evaluarea cunoștințelor (denumire, dificultate, întrebări).
  + Relații: ARE\_INTREBARILE și SUSTINE\_TESTUL.
* **Intrebare**: Întrebări asociate unui test (text întrebare, tip întrebare).
  + Relații: ARE\_RASPUNSUL.
* **Raspuns**: Variantele de răspuns disponibile (text răspuns, corect/greșit).
* **Certificat**: Datele certificatului acordat utilizatorilor la finalizarea cursului (data emiterii, identificator unic).
* **Progres**: Informații despre progresul utilizatorului (status progres, procentaj completat).
* **Categorie**: Categorii de cursuri (denumire, descriere).
* **Mesaje**: Mesajele trimise și primite între utilizatori (conținut, timestamp).
* **Recenzie**: Evaluări și opinii despre cursuri (text recenzie, rating numeric).

#### 5.1.2 Fișiere și baze de date:

**5.1.2.1 Baze de date**

Structura bazei de date Neo4j constă în:

| **Noduri (Labels)** | **Proprietăți asociate** | **Relații asociate** |
| --- | --- | --- |
| Utilizator | id, nume, email, parola, rol | INROLAT, ARE\_PROGRESUL, TRIMITE, PRIMESTE, A\_SCRIS |
| Curs | id, titlu, descriere, data\_inceput, data\_final, status | ARE\_CERTIFICAT, FACE\_PARTE\_DIN, CONTINE |
| Lectie | id, titlu, continut | CONTINE, PENTRU\_LECTIA |
| Test | id, titlu, dificultate | ARE\_INTREBARILE, SUSTINE\_TESTUL |
| Intrebare | id, text, tip | ARE\_RASPUNSUL |
| Raspuns | id, text, corect | - |
| Certificat | id, data\_emitere, identificator | - |
| Progres | id, status, procentaj | - |
| Categorie | id, denumire, descriere | - |
| Mesaje | id, continut, timestamp | TRIMITE, PRIMESTE |
| Recenzie | id, text, rating | A\_SCRIS |

**Relații și semnificații:**

* INROLAT: Un utilizator înscris la un curs.
* ARE\_CERTIFICAT: Cursul poate emite certificate.
* ARE\_PROGRESUL: Utilizatorul are un progres în cadrul unei lecții.
* CONTINE: Cursul include lecții și lecțiile includ teste.
* SUSTINE\_TESTUL: Utilizatorul parcurge teste.
* ARE\_INTREBARILE: Testul are întrebări specifice.
* ARE\_RASPUNSUL: Întrebările au variante multiple de răspuns.
* TRIMITE și PRIMESTE: Comunicare între utilizatori prin mesaje directe.
* FACE\_PARTE\_DIN: Cursurile aparțin unei categorii specifice.
* A\_SCRIS: Utilizatorul scrie recenzii despre cursuri.

**5.1.2.2 Fișiere non-DBMS**

* Fișiere multimedia (imagini, video, PDF) ale lecțiilor: Stocate pe AWS S3, accesate direct prin link-uri securizate furnizate de backend.

### 5.2 Conversii de date

Inițial, datele sunt încărcate în Neo4j prin conversia fișierelor CSV în grafuri, folosind instrumente de import date standard Neo4j (Neo4j Import Tool). Datele ulterioare sunt gestionate direct prin interfața REST/API backend (FastAPI), fără conversii adiționale.

### 5.3 Interfețe utilizator

Clase principale de utilizatori în LearnJy:

| **Clasă utilizatori** | **Roluri și responsabilități principale** | **Nr. utilizatori estimați** |
| --- | --- | --- |
| Studenți | Acces cursuri, testare, mesagerie | Aprox. 10.000 simultan |
| Instructori | Creare și administrare conținut cursuri, evaluări | Aprox. 500 simultan |
| Administratori | Management utilizatori, raportare și administrare | Aprox. 50 simultan |

#### 5.3.1 Intrări

Interfața grafică React (Next.js):

* Ecran autentificare și înregistrare utilizatori (email, parolă, date personale).
* Ecrane de management conținut (formular creare cursuri, lecții și teste).
* Ecran pentru comunicare directă (mesagerie internă).
* Formulare pentru recenzii și feedback.

**Validări:**

* Validarea email-ului și a parolelor.
* Validări pentru date numerice (evaluări, progres).
* Validarea dimensiunilor fișierelor multimedia (max. 100 MB/fișier).

#### 5.3.2 Ieșiri

* Dashboard personalizat (recomandări cursuri, progres lecții).
* Certificate PDF generate automat.
* Rapoarte de activitate (administratori și instructori).
* Ecran rezultate teste și feedback imediat.
* Ecran mesagerie și notificări în timp real.

### 5.4 Proiectarea interfețelor cu utilizatorul

* Interfețele grafice sunt proiectate să fie intuitive și responsive (desktop și mobile), având elemente clare și feedback rapid către utilizatori.
* Designul respectă standardele WCAG 2.1 pentru accesibilitate.

## 6. Scenarii de utilizare

Scenariu tipic:

* Utilizatorul se autentifică sau înregistrează în platformă.
* Primește recomandări personalizate generate de AI.
* Utilizatorul alege cursul dorit și se înscrie.
* Parcurge lecțiile și susține teste intermediare.
* Primește notificări și mesaje directe din partea instructorilor sau altor studenți.
* Finalizează cursul și primește automat certificatul.
* Poate scrie recenzii și acorda feedback.

Aceste scenarii ilustrează clar modul în care utilizatorii vor interacționa cu sistemul LearnJy și definesc comportamentul așteptat al platformei.

A black background with white circles and text

Description automatically generated

Figura 3. Diagrama cazurilor de utilizare

## 7. Proiectare de detaliu

### 7.1 Proiectare hardware de detaliu

Platforma LearnJy este integral cloud-based și nu necesită achiziția sau integrarea de componente hardware dedicate. Toate resursele sunt provisionate prin servicii cloud.

### 7.2 Proiectare software de detaliu

**Serviciu:** user\_authentication\_service

* **Clasificare:** Serviciu aplicație (backend)
* **Definiție:** Gestiunea autentificării și autorizării utilizatorilor
* **Cerințe:** Login, înregistrare, verificare rol, token JWT
* **Structuri de date interne:** Obiecte User, Token, Role
* **Constraingeri:** Timp de răspuns < 200ms; tokenurile expiră la 1h
* **Compoziție:** Folosește subservicii pentru criptare (bcrypt) și generare JWT
* **Interacțiuni:** Utilizat de frontend și gateway; verifică accesul la toate endpoint-urile protejate
* **Procesare:** Primește credentiale → validează → generează JWT → returnează token
* **Interfețe:** /login, /register, /me, folosind OpenAPI 3.0
* **Design & Integrare:** Integrează Redis pentru blacklist tokenuri și rate-limiting

**Serviciu:** course\_recommendation\_service

* **Clasificare:** Serviciu AI
* **Definiție:** Recomandă cursuri relevante pentru fiecare utilizator
* **Cerințe:** Rulare zilnică + feedback în timp real
* **Structuri de date:** Grafuri Neo4j, vectori de similaritate
* **Constraingeri:** Modelul AI trebuie să răspundă în < 2s
* **Procesare:** Extrage progresul și preferințele → generează scoruri → returnează top N cursuri
* **Interfețe:** /recommendations/{user\_id}

### 7.3 Proiectare detaliată de securitate

* **Autentificare:** pe bază de email/parolă, cu tokenuri JWT semnate; opțional 2FA
* **Autorizare:** pe bază de roluri (Student, Instructor, Admin)
* **Jurnalizare și audit:** toate acțiunile sensibile sunt logate și pot fi urmărite (ex. login, acces date personale, emitere certificat)
* **Criptare:** date în tranzit (TLS 1.2+), date sensibile criptate în repaus (AES-256)
* **Porturi de rețea:** doar HTTPS (443) și WebSocket (wss) sunt expuse
* **Prevenire intruziuni:** firewall aplicat în cloud, rate limiting, validare input anti-injection

### 7.4 Proiectare de detaliu pentru performanța sistemului

* **Capacitate estimată:** 10.000 utilizatori simultan, > 1 mil. conturi
* **Timp răspuns API:** < 500 ms pentru majoritatea endpointurilor
* **Disponibilitate:** SLA minim 99.5%, obiectiv 99.9%
* **Performanță:** scalare automată prin AWS EC2 autoscaling / containerizare (Kubernetes - opțional)
* **Fiabilitate:** replicare baze date, fallback la cache (ex. Redis)
* **Backup:** zilnic, cu retenție 30 zile
* **Puncte critice:** server de autentificare, serviciu AI, interfață Neo4j – toate monitorizate cu alerte

### 7.5 Proiectare detaliată a comunicațiilor interne

* **Componente:** frontend (client), backend (API), baza de date (Neo4j), motor AI, serviciu email
* **Format date:** JSON (pentru REST), text/json pentru WebSocket
* **Interfețe:** RESTful APIs + WebSockets pentru notificări
* **Număr clienți:** până la 10.000 conexiuni active pe WebSocket
* **Topologie logică:**
  + Frontend → Gateway → Backend → Neo4j / AI Engine
  + AI Engine → Backend → Frontend
* **Comunicații:** asincrone acolo unde este posibil (ex: joburi de generare certificat)

## 8. Controale pentru verificarea integrității sistemului

* **Securitate internă:** acces pe bază de roluri; separarea clară a privilegiilor
* **Audit:** loguri persistente pentru toate acțiunile critice, păstrate minim 1 an
* **Piste audit:** fiecare acțiune logată include ID utilizator, IP, timestamp, endpoint, payload
* **Validare date:** toate câmpurile sunt validate server-side (ex. email, progres, scoruri)
* **Verificare modificări:** orice modificare asupra datelor critice (ex. notă test, emitere certificat) este jurnalizată
* **Identificare completă:** fiecare log include: user\_id, IP, sesiune, timestamp, acțiune

Anexa A: Gestiunea modificărilor documentului

Instrucțiuni: Furnizați informații despre modul în care dezvoltarea și distribuția documentului va fi controlată și urmărită. Utilizați tabelul de mai jos pentru a furniza numărul de versiune, data versiunii, autorul/deținătorul versiunii și o scurtă descriere a motivului pentru crearea versiunii revizuite.

Tabel 1 – Înregistrarea modificărilor asupreaa documentului curent

| versiune | Data | Autorul/Deținătorul | Descriere |
| --- | --- | --- | --- |
| <X.X> | <ZZ/LL/AAAA> | <nume autor> | <Descrierea modificării> |
| <X.X> | <ZZ/LL/AAAA> | <nume autor> | <Descrierea modificării> |
| <X.X> | <ZZ/LL/AAAA> | <nume autor> | <Descrierea modificării> |

Anexa B: Acronime

*Instrucțiuni: Furnizați o listă de acronime și traduceri literale asociate utilizate în cadrul documentului. Enumerați acronimele în ordine alfabetică folosind un format tabular, așa cum este ilustrat mai jos.*

Tabel 2 - Acronime

| Acronim | Forma completă |
| --- | --- |
| <Acronim> | <Forma completă> |
| <Acronim> | <Forma completă> |
| <Acronim> | <Forma completă> |

Anexa C Documente la care se face referire

*Instrucțiuni: Sintetizați relația acestui document cu alte documente relevante. Furnizați informații de identificare pentru toate documentele folosite pentru a ajunge la și/sau referite în acest document (de exemplu, documente conexe și/sau asociate, documente prealabile, documentație tehnică relevantă, etc.).*

Tabel 3 – Documente la care se facce referire

| Nume document | Locație sau URL | Dată emitere document |
| --- | --- | --- |
| < Nume document > | <Locație sau URL> | <ZZ/LL/AAAA> |
| < Nume document > | <Locație sau URL> | <ZZ/LL/AAAA> |
| < Nume document > | <Locație sau URL> | <ZZ/LL/AAAA> |