# Capitolul 3. PROIECTAREA aplicației

Proiectarea aplicației pentru planificare inteligentă are ca obiectiv dezvoltarea unui sistem informatic capabil să genereze automat orare universitare, prin integrarea tehnologiilor moderne de inteligență artificială. Această etapă presupune definirea unei arhitecturi robuste, care să gestioneze eficient componentele esențiale ale aplicației – interfața utilizator (frontend), logica aplicației și serviciile (backend), precum și sistemul de gestiune a datelor (baza de date).

Un element distinctiv al proiectului îl reprezintă componenta de inteligență artificială, integrată pentru a automatiza procesul de alocare a resurselor educaționale conform unui set de reguli și constrângeri predefinite. Astfel, sistemul oferă o soluție modernă și adaptabilă, care combină o interfață intuitivă cu o logică de planificare avansată, permițând utilizatorului să introducă și să gestioneze date complexe într-un mod simplificat, obținând în final un orar valid, coerent și complet, generat în mod automat.

## 3.1 Arhitectura și funcționalitatea proiectului

Arhitectura aplicației este de tip client–server și evidențiază separarea clară între componentele principale: interfața frontend realizată în React.js, logica de procesare din backend (Flask API), baza de date MySQL și modulele de generare și validare a orarului.

După autentificare, utilizatorul interacționează cu interfața aplicației pentru a introduce date despre grupe, săli, profesori și reguli, care sunt apoi procesate prin rutele corespunzătoare din backend. Datele sunt stocate într-o bază relațională, iar generarea orarului este realizată fie cu ajutorul unui model AI (GPT-4), fie cu un algoritm clasic implementat local. Diagrama arhitecturală (Figura 4) reflectă această structură modulară și fluxul logic al aplicației.

A diagram of a flowchart

AI-generated content may be incorrect.

#### Fig. 4. Diagrama arhitecturală a aplicației

## 3.2 Diagrame UML pentru modelarea structurii

Diagramele UML (Unified Modeling Language) sunt reprezentări grafice standardizate utilizate pentru a vizualiza, specifica, construi și documenta componentele unui sistem software. Acestea ajută la înțelegerea arhitecturii aplicației și facilitează comunicarea între membrii echipei de dezvoltare.[6]

### 3.2.1. Diagrama pachetelor

Diagramele de pachete sunt utilizate pentru a reprezenta organizarea logică a componentelor unui sistem software, grupate în module sau pachete. Ele evidențiază dependențele dintre module și oferă o imagine de ansamblu asupra structurii proiectului.[7]

În cadrul aplicației dezvoltate, au fost create mai multe diagrame de pachete pentru:

1. Backend-ul aplicației este structurat modular, respectând principiile unei arhitecturi bine organizate și scalabile. Componentele principale sunt grupate în pachete distincte, fiecare având un rol clar în funcționarea sistemului. Diagrama din Figura 5 evidențiază această organizare logică, în care sunt delimitate modulele pentru conexiunea la baza de date, logica de generare și validare a orarului, rutarea cererilor, testarea funcționalităților și inițializarea aplicației. Această abordare facilitează dezvoltarea clară și întreținerea eficientă a codului.

A diagram of a diagram

AI-generated content may be incorrect.

#### Fig. 5. Diagrama pachetelor- Backend

1. Structura frontend-ului este organizată clar și logic, având ca punct central directorul src/, care reunește toate componentele necesare funcționării interfeței aplicației. Așa cum este ilustrat în Figura 6, codul este împărțit în module pentru pagini, funcționalități logice, testare, stilizare și inițializare. Această organizare facilitează atât dezvoltarea, cât și mentenanța aplicației, oferind o bază solidă pentru extinderea ulterioară a funcționalităților.

A diagram of a company

AI-generated content may be incorrect.

#### Fig.6 Diagrama pachetelor- Frontend

1. Rutele din backend sunt organizate în mod modular, fiecare fișier având o responsabilitate clară în gestionarea unei funcționalități specifice. Așa cum se observă în Figura 7, toate aceste rute sunt centralizate în modulul routes, care le grupează și le integrează în aplicația principală. Această structurare permite o mai bună organizare a codului, o gestionare eficientă a cererilor API și o extensibilitate ușoară în cazul adăugării de noi funcționalități.

A diagram of a diagram

AI-generated content may be incorrect.

#### Fig. 7 Diagrama pachetelor- Routes

1. Directorul Pages reunește toate componentele vizuale principale ale aplicației, fiecare dintre ele corespunde unei funcționalități esențiale. După cum este ilustrat în Figura 8, aceste pagini sunt responsabile pentru interacțiunea directă cu utilizatorul – de la autentificare și administrare, până la generarea și vizualizarea orarului. Structura este clară și intuitivă, permițând o navigare fluentă între secțiunile aplicației și o experiență de utilizare coerentă.

A diagram of a computer program

AI-generated content may be incorrect.

#### Fig. 8 Diagrama pachetelor- Pages

### 3.2.2 Diagrama claselor

Diagrama de clase oferă o vedere de ansamblu asupra componentelor logice implicate în generarea orarului și modul în care acestea interacționează. Așa cum este ilustrat în Figura 9, sunt reprezentate clasele principale ale backend-ului, împreună cu atributele și metodele relevante.

Structura evidențiază clasele responsabile de generarea orarului (fie prin algoritm AI, fie prin algoritm clasic), validarea acestuia și accesul la date. Relațiile dintre clase sunt marcate prin asocieri clare, ilustrând dependențele funcționale din cadrul sistemului. Acest model contribuie la o înțelegere clară a arhitecturii interne și oferă o bază solidă pentru extinderea ulterioară a funcționalităților.

A diagram of a computer

AI-generated content may be incorrect.

#### Fig. 9 Diagrama clasei -Backend

Figura 10 prezintă modelarea logică a componentelor frontend implicate în procesul de configurare a regulilor și generarea efectivă a orarului. Diagrama evidențiază relațiile dintre componentele vizuale React (precum SetareReguli și GeneratedTimetable) și hook-urile logice asociate (useSetariReguli, useGeneratedTimetable, useOrarGenerator etc.).

Fiecare element gestionează atribute și metode proprii, specifice rolului său în aplicație. Relațiile dintre componente sunt exprimate clar, prin interacțiuni precum utilizare, transmitere de reguli sau validare. Această structură modulară asigură o separare eficientă între interfață și logică, contribuind la claritatea și întreținerea codului în aplicația React.

A diagram of a company

AI-generated content may be incorrect.

#### Fig. 10 Diagrama claselor- Frontend

### 3.2.3 Diagrama de componente

Diagrama prezentată în Figura 11 oferă o vedere de ansamblu asupra arhitecturii aplicației, concentrându-se pe conexiunile dintre componentele frontend, backend și baza de date. Se evidențiază fluxul de date de la interfața utilizatorului până la stocarea informațiilor, subliniind comunicarea dintre module.

Componentele React din frontend interacționează cu backend-ul Flask prin rute specifice, fiecare dintre acestea corespunde unei entități din baza de date MySQL. Fiecare cerere trimisă din interfață este gestionată de un modul backend care efectuează operații asupra unei tabele dedicate. Diagrama reflectă astfel organizarea clară a aplicației și separarea logică a responsabilităților.

A diagram of a diagram

AI-generated content may be incorrect.

#### Fig. 11 Diagrama de componente ale aplicației

## 3.3 Diagrame UML pentru modelarea comportamentului

Pentru a surprinde comportamentul sistemului în raport cu utilizatorul și cu procesele interne, au fost utilizate mai multe diagrame UML comportamentale. Acestea evidențiază fluxurile logice, interacțiunile dinamice dintre componente și cazurile de utilizare ale aplicației.

### 3.3.1 Diagrama cazurilor de utilizare

Figura 12 ilustrează cazurile de utilizare asociate interacțiunii utilizatorului cu aplicația. Diagrama evidențiază principalele funcționalități accesibile acestuia, oferind o vedere de ansamblu asupra fluxului general de lucru.

Utilizatorul are posibilitatea să se autentifice, să configureze regulile necesare generării orarului, să declanșeze procesul de generare propriu-zis, să vizualizeze rezultatul și, la final, să exporte orarul în format PDF sau Excel. Aceste acțiuni acoperă întregul ciclu de utilizare al aplicației și reflectă funcționalitățile esențiale puse la dispoziție într-un mod intuitiv.

A diagram of a network

AI-generated content may be incorrect.

#### Fig. 12 Diagrama cazurilor de utilizare

### 3.3.2 Diagrama de secvență

Figura 13 ilustrează diagrama de secvență aferentă procesului de generare a orarului cu ajutorul modelului AI. Aceasta evidențiază fluxul cronologic de mesaje între principalele componente ale aplicației: utilizator, interfața React, backend-ul Flask, modulul de generare (OrarGenerator), API-ul OpenAI și baza de date MySQL.

Procesul începe cu acțiunea utilizatorului, care declanșează cererea de generare. Datele și regulile sunt procesate secvențial, transmise către modelul AI, iar răspunsul obținut este salvat și apoi returnat către frontend pentru afișare. Diagrama oferă o imagine clară asupra colaborării dintre module în cadrul acestui flux automatizat.

A diagram of a company

AI-generated content may be incorrect.

#### Fig. 13 Diagrama de secvență

### 3.3.3 Diagrama de activitate

Figura 14 prezintă diagrama de activitate care descrie, pas cu pas, logica funcțională a procesului de generare a orarului. Diagrama urmărește traseul parcurs de la inițierea cererii până la afișarea sau exportul rezultatului.

Procesul începe cu introducerea regulilor de către utilizator, continuă cu transmiterea acestora către backend, colectarea datelor din baza de date, generarea propriu-zisă a orarului (prin AI sau algoritm clasic) și validarea rezultatului. În funcție de validare, sistemul decide dacă orarul este afișat și poate fi exportat sau dacă se generează un mesaj de eroare. Acest model vizual evidențiază deciziile critice din fluxul aplicației și modul în care sunt gestionate rezultatele.

A diagram of a company

AI-generated content may be incorrect.

#### Fig. 14 Diagrama de activitate