

*Anexa 6 la Procedura de înscriere la examenele de finalizare a studiilor*

# PROIECT DE DIPLOMĂ

Îndrumător proiect/Coordonator științific,  
Ș.l.dr.ing. Mihaela ȚIPLEA

Absolvent,  
Liliana- Petruța ZANFIR

Galați  
2025

PROGRAMUL DE STUDII: Calculatoare și Tehnologia Informației

# **APLICAȚIE PENTRU PLANIFICARE INTELIGENTĂ UTILIZÂND TEHNICI DE AI**

Coordonator științific,  
Ș.l.dr.ing. Mihaela ȚIPLEA

Absolvent,  
Liliana- Petruța ZANFIR

Galați  
2025

## Rezumat

Proiectul „Aplicație pentru planificare inteligentă utilizând tehnici de AI” are ca scop realizarea unei soluții software moderne care automatizează procesul de generare a orarelor , adaptată la nevoile reale ale instituțiilor de învățământ din România. Aceasta integrează tehnici de inteligență artificială și metode deterministe pentru a obține orare coerente, corecte și complete pentru toate grupele și subgrupele existente.

Aplicația permite definirea detaliată a regulilor educaționale (cursuri comune pe an, seminare și proiectele pe grupă, laboratoare pe subgrupă, pauze, săli disponibile, discipline, disponibilitatea profesorilor etc.), iar generarea se poate face atât cu ajutorul modelului GPT-4 (OpenAI), cât și printr-un algoritm clasic propriu.

În plus, sistemul oferă funcționalități de validare a structurii orarului generat, verificând absența suprapunerilor între profesori, săli sau activități, precum și respectarea tuturor constrângerilor educaționale impuse.

În urma analizei aplicațiilor existente, s-a constatat lipsa unei soluții complete, adaptate specificului educațional din România, care să combine inteligența artificială cu metode deterministe clasice pentru generarea unui orar coerent și corect. Prin urmare, dezvoltarea unei astfel de aplicații a fost considerată necesară.

Generarea orarului este susținută de modelul GPT-4 oferit de OpenAI, care propune orare complete conform regulilor introduse, și de un algoritm clasic propriu, care aplică o abordare logică deterministă asupra datelor disponibile. Cele două metode au fost evaluate comparativ, în funcție de flexibilitate, viteză de execuție și calitatea rezultatelor.

Lucrarea își propune furnizarea unei soluții moderne, prietenoase cu utilizatorul, clare și intuitive, care să sprijine procesul de planificare a resurselor educaționale și să răspundă cerințelor actuale ale mediului academic.

Rezultatele obținute confirmă eficiența unei aplicații care îmbină AI-ul cu reguli personalizabile, oferind un instrument fiabil pentru planificarea academică, ușor de utilizat și adaptabil la nevoile reale ale utilizatorilor.

Pe viitor, direcțiile de dezvoltare pot include extinderea aplicației pentru a deservi mai multe instituții de învățământ, integrarea cu platforme educaționale deja existente (precum cataloage online) și adăugarea unor mecanisme de învățare automată, capabile să adapteze regulile de generare a orarului în funcție de preferințele și istoricul utilizatorilor.

## CUPRINS

Introducere.....	1
Capitolul 1. Studiul problemei și analiza soluțiilor existente .....	2
1.1 Motivația utilizării unei aplicații automate pentru generarea orarului.....	2
1.2 Analiza aplicațiilor existente pentru generarea orarului.....	3
1.2.1 UniTime.....	4
1.2.2 aSc TimeTables .....	5
1.2.3 Generator-Orare (Horarium.ai) .....	5
1.3 Profilul utilizatorului și scopul utilizării aplicației .....	6
1.4 Identificarea operațiilor efectuate de utilizator în cadrul aplicației.....	7
Capitolul 2. Specificarea cerințelor .....	9
2.1 Cerințe privind gestionarea și prelucrarea datelor .....	9
2.2 Cerințe privind generarea și validarea orarului .....	10
Capitolul 3. Arhitectura și proiectarea sistemului .....	11
3.1 Arhitectura și funcționalitatea proiectului.....	11
3.2 Diagrame UML pentru modelarea structurii .....	12
3.2.1. Diagrama pachetelor .....	12
3.2.2 Diagrama claselor .....	13
3.2.3 Diagrama de componente.....	15
3.3 Diagrame UML pentru modelarea comportamentului.....	15
3.3.1 Diagrama cazurilor de utilizare .....	15
3.3.2 Diagrama de secvență .....	16
3.3.3 Diagrama de activitate .....	16
Capitolul 4. Implementarea și descrierea aplicației .....	18
4.1. Limbajele de programare implicate în dezvoltarea aplicației.....	18
4.1.1 JavaScript.....	18
4.1.2 Python.....	19
4.1.3 SQL .....	19
4.1.4 HTML5.....	20
4.1.5 CSS3 .....	20
4.2. Framework-urile și bibliotecile implicate în dezvoltarea aplicației.....	21
4.2.1 React.js.....	21
4.2.2 Flask.....	21

4.2.3 Integrarea OpenAI GPT-4 în backend-ul Flask .....	22
4.2.4 Tehnologii auxiliare utilizate în aplicație.....	23
4.3. Implementarea și funcționalitatea aplicației.....	23
4.3.1 Configurarea mediului de dezvoltare .....	24
4.3.2 Implementarea funcționalităților principale.....	25
4.3.3 Ghidul utilizării aplicației .....	27
4.4. Probleme întâmpinate în timpul implementării .....	36
4.5. Comparația cu metoda clasică de generare a orarelor .....	38
Capitolul 5. Testarea și validarea aplicației .....	40
5.1 Testarea manuală.....	40
5.2 Testarea automată a backendului.....	42
5.2.1 Testarea funcționalității de generare orar folosind AI .....	42
5.2.2 Testarea generatorului propriu de orar .....	43
5.2.3 Testarea validării orarului generat .....	44
5.2.4 Testarea componentelor auxiliare.....	44
5.3 Testarea automată a frontendului .....	45
5.3.1 Testarea componentei Home.jsx .....	45
5.3.2 Testarea componentei GeneratedTimetable.jsx .....	45
5.3.3 Testarea componentei SetareReguli.jsx .....	46
Concluzii .....	47
Bibliografie .....	49

## LISTA FIGURILOR

Fig. 1 Interfața aplicației UniTime – module de alocare, orar și sugestii automate .....	4
Fig. 2 Interfața aplicației aSc TimeTables – vizualizarea orarului pe clase și zile .....	5
Fig. 3 Interfața aplicației Horarium.ai – vizualizare orar generat.....	6
Fig. 4 Diagrama arhitecturală a aplicației.....	11
Fig. 5 Diagrama pachetelor- Backend .....	12
Fig. 6 Diagrama pachetelor- Frontend.....	12
Fig. 7 Diagrama pachetelor- Routes.....	13
Fig. 8 Diagrama pachetelor- Pages .....	13
Fig. 9 Diagrama clasei -Backend.....	14
Fig. 10 Diagrama claselor- Frontend .....	14
Fig. 11 Diagrama de componente ale aplicației.....	15
Fig. 12 Diagrama cazurilor de utilizare .....	16
Fig. 13 Diagrama de secvență.....	16
Fig. 14 Diagrama de activitate.....	17
Fig. 15 Exemplu de cod JavaScript folosit în aplicație .....	18
Fig. 16 Exemplu de cod Python folosit în aplicație .....	19
Fig. 17 Exemplu de cod SQL folosit în aplicație .....	19
Fig. 18 Exemplu de cod HTML5 folosit în aplicație.....	20
Fig. 19 Exemplu de cod CSS3 – prin clase - folosit în aplicație.....	20
Fig. 20 Exemplu de cod CSS3 – prin stiluri inline - folosit în aplicație .....	21
Fig. 21 Exemplu de cod React folosit în aplicație.....	21
Fig. 22 Exemplu de cod Flask folosit în aplicație .....	22
Fig. 23 Integrarea OpenAI GPT-4 în backend-ul Flask .....	22
Fig. 24 Exemplu de cod Bootstrap folosit în aplicație.....	23
Fig. 25 Exemplu de cod Fetch folosit în aplicație.....	23
Fig. 26 Accesarea aplicației și autentificarea utilizatorului.....	27
Fig. 27 Autentificare.....	27
Fig. 28 Înregistrare .....	28
Fig. 29 Funcționalitățile aplicației .....	28
Fig. 30 Caracteristicile aplicației .....	29
Fig. 31 Dashboard-ul aplicației .....	29
Fig. 32 Validare configurare completă.....	29
Fig. 33 Interfața pentru gestionarea grupelor .....	30
Fig. 34 Interfața pentru gestionarea sălilor .....	31
Fig. 35 Gestionarea profesorilor .....	31
Fig. 36 Setarea regulilor de generare .....	32
Fig. 37 Editarea regulilor de generare .....	33
Fig. 38 Generarea orarului.....	33
Fig. 39 Raport de validare a orarului.....	34
Fig. 40 Orarul afișat vizual.....	34
Fig. 41 Exportul orarului generat în format PDF .....	35

Fig. 42 Exportul orarului generat în format Excel.....	35
Fig. 43 Testarea funcționalității de generare orar folosind Ai .....	43
Fig. 44 Testarea generatorului propriu de orar .....	43
Fig. 45 Testarea validării orarului generat .....	44
Fig. 46 Testarea componentelor auxiliare.....	44
Fig. 47 Testarea componentei Home.jsx.....	45
Fig. 48 Testarea componentei GeneratedTimetable.jsx.....	45
Fig. 49 Testarea componentei SetareReguli.jsx .....	46

## LISTA TABELELOR

Tabel 1 Implementarea funcționalităților principale .....	26
Tabel 2 Compararea cu metoda clasică .....	38

## INTRODUCERE

Organizarea eficientă a activităților educaționale reprezintă, în prezent, o provocare majoră pentru instituțiile de învățământ superior. Numărul ridicat de discipline, grupe, subgrupe, săli și cadre didactice implică un proces de planificare complex, consumator de timp și predispus la erori. Generarea manuală a orarului presupune un efort semnificativ și, adesea, nu reușește să răspundă în mod optim cerințelor reale ale studenților și profesorilor.

Odată cu evoluția tehnologică, a apărut nevoia firească de soluții inteligente și automatizate pentru acest proces. Avansul inteligenței artificiale a deschis oportunități concrete pentru digitalizarea planificării academice, reducând efortul uman și creșterea preciziei. Astfel, dezvoltarea unei platforme capabile să genereze automat orare complete, coerente și adaptate nevoilor reale devine nu doar oportună, ci esențială în contextul actual.

Aplicația propusă oferă o soluție modernă, adaptată structurii academice din România. Platforma ține cont de tipul activităților, de distribuția acestora pe ani, grupe și subgrupe, precum și de disponibilitatea profesorilor. În plus, este gândită să fie intuitivă, cu o interfață prietenoasă și un timp de învățare redus pentru utilizatori.

Pentru a răspunde unor scenarii variate, aplicația integrează două metode de generare a orarului:

- una bazată pe inteligență artificială (GPT-4 – OpenAI), care analizează reguli exprimate în limbaj natural;
- una algoritmică clasică, ce aplică reguli deterministe bine definite.

Funcționalitățile cheie ale aplicației includ:

- generarea automată a orarului pentru toate grupele și subgrupurile, conform datelor introduse și regulilor definite;
- integrarea a două metode complementare de generare: AI (GPT-4) și clasică;
- configurarea de reguli personalizate privind intervale orare, pauze, distribuția activităților;
- gestionarea completă a datelor academice (profesori, discipline, săli, grupe și subgrupe);
- validarea automată a orarului generat (detectarea suprapunerilor de săli, profesori sau activități);
- exportul orarului în formate PDF și Excel pentru distribuție sau arhivare;
- interfață grafică modernă și receptivă, ușor de utilizat;
- posibilitatea extinderii în viitor cu module precum planificarea examenelor sau a activităților extracurriculare.

Prin toate aceste componente, aplicația aduce o contribuție semnificativă la modernizarea procesului educațional, oferind un instrument eficient pentru alocarea optimă a resurselor și generarea unor orare fiabile și ușor de accesat. Soluția propusă susține digitalizarea activităților academice și vine în sprijinul personalului didactic și administrativ, reducând efortul repetitiv și îmbunătățind semnificativ calitatea procesului de planificare.



## CAPITOLUL 1. STUDIUL PROBLEMEI ȘI ANALIZA SOLUȚIILOR EXISTENTE

Pentru dezvoltarea unei aplicații sustenabile și eficace, a fost necesar un studiu preliminar care să includă o analiză critică a soluțiilor deja existente pe piață. Acest demers a urmărit în mod particular funcționalitățile oferite de aceste aplicații, modul de interacțiune al utilizatorului cu interfața și identificarea elementelor care pot fi îmbunătățite pentru a asigura o experiență de utilizare cât mai intuitivă și eficientă.

Ordonarea activităților academice într-un mod coerent și funcțional reprezintă o provocare esențială pentru orice instituție de învățământ superior. Generarea manuală a orarului este un proces anevoios, complex și consumator de timp, care presupune corelarea multiplilor factori: disponibilitatea cadrelor didactice, distribuția disciplinelor, capacitatea și tipologia sălilor, precum și compatibilitatea dintre grupe și subgrupe. În lipsa unui instrument automatizat, acest proces riscă să conducă la erori, suprapuneri sau o utilizare ineficientă a resurselor.

În acest context, adoptarea unei aplicații automatizate pentru generarea orarului nu este doar oportună, ci și necesară. O astfel de soluție reduce substanțial efortul administrativ, oferind rezultate rapide, coerente și adaptabile cerințelor specifice ale fiecărei facultăți sau specializări.

Scopul acestui capitol este de a evidenția relevanța unui sistem de tip TTGS (Timetable Generation System), de a realiza o analiză comparativă a aplicațiilor existente cu funcționalități similare, de a contura profilul utilizatorului vizat și de a prezenta, succint, funcționalitățile de bază implementate în cadrul acestei lucrări

### 1.1 Motivația utilizării unei aplicații automate pentru generarea orarului

Generarea unui orar reprezintă o activitate esențială, dar deosebit de complexă din punct de vedere administrativ. Aceasta necesită luarea în considerare a numeroși factori, precum: disponibilitatea profesorilor, alocarea disciplinelor, capacitatea și tipul sălilor în care se desfășoară activitățile, gruparea studenților în funcție de nivel (an, grupă sau subgrupă), precum și evitarea suprapunerilor sau a pauzelor nejustificate din program.

În prezent, în multe instituții de învățământ, acest proces este realizat manual sau cu ajutorul unor aplicații semi-automatizate. Drept urmare, se consumă mult timp, pot apărea erori frecvente, conflicte de program, distribuții dezechilibrate ale activităților sau chiar omiterea unor discipline. În plus, aceste metode sunt adesea rigide, greu de modificat și dificil de validat într-un mod transparent.

În ultimii ani, s-a observat o tendință clară spre automatizarea acestui proces prin utilizarea sistemelor informatice dedicate. Platforme precum UniTime [1], aSc TimeTables [2] sau GeneratorOrare [3] oferă facilități avansate pentru planificarea automată a orarelor. Acestea permit definirea de constrângeri rigide și flexibile (hard și soft constraints), gestionarea preferințelor cadrelor didactice și generarea orarelor optimizate, folosind algoritmi de satisfacere a constrângerilor sau algoritmi genetici.

Totodată, inteligența artificială devine o soluție tot mai răspândită pentru generarea automată a orarelor. Modele avansate, precum GPT-4 [4], permit formularea regulilor în limbaj natural și generarea de orare structurate (în format JSON, HTML etc.). Acest lucru deschide noi perspective pentru personalizarea și flexibilizarea planificării academice, chiar dacă aplicarea acestor tehnologii în mediul este încă la început.

Utilizarea unei aplicații automatizate pentru generarea orarului este justificată printr-o serie de avantaje concrete:

- Reducerea timpului și a efortului administrativ: Generarea manuală a unui orar complet poate dura zile sau săptămâni, în special în instituțiile cu mai multe programe, grupe și cadre didactice. Un sistem automat poate realiza această sarcină în câteva secunde sau minute, în funcție de complexitate, eliminând munca repetitivă și costisitoare.
- Gestionarea eficientă a regulilor și constrângerilor: O aplicație automatizată poate procesa simultan un număr mare de reguli, atât rigide (ex. imposibilitatea ca un profesor sau o sală să fie alocate simultan la mai multe activități), cât și flexibile (ex. preferințe orare ale cadrelor didactice, evitarea pauzelor lungi, programări într-un anumit interval orar pentru o activitate specifică la nivel de facultate).
- Reducerea erorilor și validarea structurată a orarului: Automatizarea contribuie la eliminarea erorilor frecvente, cum ar fi suprapunerile de activități, lipsa sălilor disponibile sau omisiunea unor discipline. Sistemul oferă validări automate care asigură coerența și corectitudinea orarului generat.
- Flexibilitate și adaptabilitate la modificări: În cazul modificării unor parametri (ex. indisponibilitatea unui profesor sau rezervarea unei săli), aplicația permite regenerarea orarului.
- Transparență și validare accesibilă: Orarul generat este prezentat într-un format clar și structurat, putând fi exportat în formate uzuale (PDF, Excel, HTML). Acest lucru facilitează revizuirea, validarea și distribuirea către studenți, cadre didactice sau personal administrativ.

În concluzie, generarea manuală a orarului este un proces laborios și predispus la erori. Soluțiile moderne, bazate pe algoritmi sau inteligență artificială, oferă un cadru mai eficient, mai flexibil și mai sigur pentru planificarea academică. Acestea pot transforma semnificativ modul în care instituțiile de învățământ organizează activitățile didactice, contribuind la un sistem educațional mai bine structurat și adaptat nevoilor actuale.

## 1.2 Analiza aplicațiilor existente pentru generarea orarului

Pentru a fundamenta procesul de proiectare al aplicației propuse, a fost realizată o analiză comparativă a principalelor soluții existente pe piață destinate generării automate a orarelor. Această analiză s-a concentrat pe modul de funcționare al aplicațiilor, gradul de interactivitate oferit utilizatorului, particularitățile interfeței grafice și aspectele care pot fi optimizate pentru a facilita o utilizare mai intuitivă și eficientă.

În urma analizării mai multor platforme, s-a constatat că multe aplicații actuale dispun de interfețe greoaie, dificil de utilizat și configurat, necesitând adesea instalare locală și un consum ridicat de resurse. În plus, unele soluții sunt costisitoare, au timpi mari de procesare și presupun o curbă de învățare ridicată pentru utilizatorii fără experiență tehnică. Suplimentar, afișarea simultană a unor volume mari de informații într-o singură vizualizare poate afecta negativ experiența de utilizare, ducând la confuzie sau omisiuni.

Pentru această analiză, au fost selectate trei aplicații reprezentative: UniTime, aSc TimeTables și Generator-Orare (Horarium.ai). Criteriile de selecție au inclus notorietatea în domeniu, diversitatea funcționalităților oferite și diferențele de abordare tehnică și practică. Scopul acestei analize a fost identificarea punctelor forte și a limitărilor fiecărei aplicații, în vederea extragerii unor concluzii utile pentru dezvoltarea unei soluții personalizate, adaptate nevoilor specifice ale instituțiilor de învățământ superior.

### 1.2.1 UniTime

UniTime este un sistem open-source pentru programarea activităților academice, dezvoltat de Universitatea Indiana și utilizat astăzi de numeroase universități din întreaga lume [1]. Popularitatea sa se datorează flexibilității ridicate și capacității de configurare detaliată, fiind o soluție potrivită în special pentru instituțiile de mari dimensiuni (Figura 1).

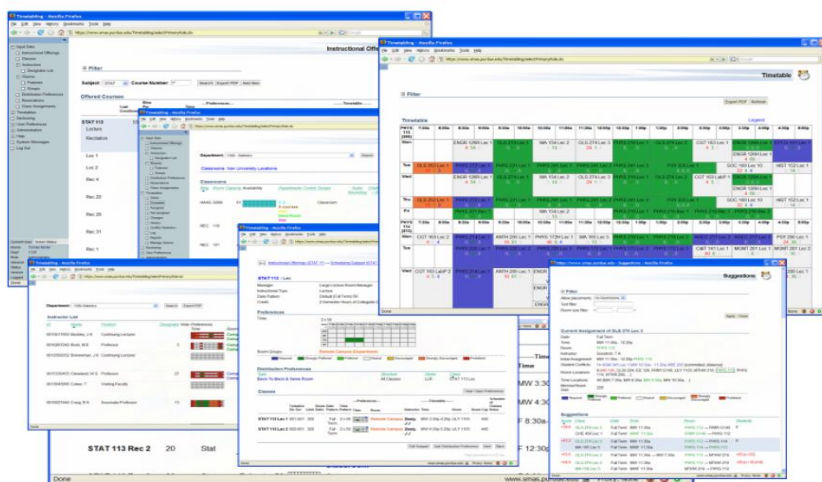


Fig. 1 Interfața aplicației UniTime – module de alocare, orar și sugestii automate

Platforma permite definirea preferințelor profesorilor, alocarea automată a resurselor, aplicarea de constrângeri rigide și flexibile, integrarea cu sisteme externe și generarea programelor personalizate pentru fiecare utilizator. Un avantaj important este posibilitatea configurării detaliate a regulilor, ceea ce oferă un control extins asupra modului de generare a orarului.

Totuși, utilizarea UniTime implică și o serie de dificultăți. Instalarea este complicată și necesită cunoștințe tehnice avansate, iar interfața grafică, deși completă, nu este ușor de utilizat pentru persoanele fără experiență tehnologică. Platforma nu este optimizată



INFO Ecatrina Boama Laborator	MAT Cristian Stanciu	BIO Stefania Gheorghescu
MAT Cristian Stanciu	ENG Larisa Cristea FRA Sabina Popescu	EDFIZ Cosmin Popescu Sala de sport
ROM Monica Stoian	ROM Monica Stoian	GEOGRAFIE Marian Cretu
INFO Ecatrina Boama Laborator	INFO Ecatrina Boama Laborator	MAT Cristian Stanciu
ROM Monica Stoian	MAT Cristian Stanciu	BIO Stefania Gheorghescu

Fig. 3 Interfața aplicației Horarium.ai – vizualizare orar generat

Procesul este complet automatizat: după introducerea datelor esențiale (profesori, clase, săli, materii și reguli), aplicația generează rapid orare conforme cu cerințele uzuale. Interfața este clară, lizibilă și poate fi accesată și de pe dispozitive mobile, deși utilizarea optimă se face de pe desktop.

Cu toate acestea, Horarium are și limitări: nu permite configurarea de reguli avansate sau integrarea cu alte sisteme prin API, iar versiunea completă este disponibilă doar contra cost. În lipsa unei versiuni gratuite funcționale, accesibilitatea poate fi restrânsă.

În concluzie, Horarium.ai este o opțiune eficientă pentru școli care doresc o soluție rapidă și ușor de folosit, dar mai puțin potrivită pentru instituții cu nevoi complexe sau .

### 1.3 Profilul utilizatorului și scopul utilizării aplicației

Aplicația descrisă în acest document a fost creată folosind un model centralizat de utilizare, astfel încât același utilizator, având rolul de administrator complet, este responsabil de toate etapele de configurare și de generare a orarului. Adaptarea acestui model pentru un singur utilizator se datorează simplității și eficienței sale funcționale și, de asemenea, pentru că reproduce lumea reală de operare care se întâmplă în multe instituții mici și mijlocii, unde o singură persoană (responsabilă cu orarul) este însărcinată cu generarea orarelor (în general, această persoană face parte din personalul secretariatului sau este responsabilă de activitățile academice).

Utilizatorul final nu este împărțit în roluri (de exemplu, studenți, profesori, secretariat sau biroul decanului), având acces complet la toate părțile sistemului. Acest utilizator elaborează informațiile necesare despre personalul didactic, disciplinele aferente, grupurile și subgrupurile de studenți, sălile disponibile și, de asemenea, despre regulile și constrângerile impuse în procesul de planificare. În acest sens, aplicația nu este un mediu colaborativ, ci funcționează ca un instrument de lucru pentru utilizatorul care deține toate drepturile asupra întregii proceduri.

Avantajul acestui model este că oferă consistență operațională, simplifică administrarea - nu este nevoie de autentificare multiplă, nu este nevoie de drepturi de acces diferite și nu este nevoie de sincronizarea diferitelor roluri de utilizator. Totul este într-un singur loc, controlat global, cu un flux direct, rapid și previzibil.

Aplicația se adresează în special personalului administrativ implicat în organizarea programului academic, oferindu-i un instrument modern, rapid și ușor de utilizat pentru generarea orarelor. Automatizarea procesului contribuie la reducerea efortului manual și a erorilor umane, păstrând în același timp un nivel ridicat de control asupra datelor.

Obiectivele aplicației sunt:

- Centralizarea și simplificarea procesului de planificare academică: Platforma oferă o interfață prietenoasă, modernă și centralizată, care facilitează accesul utilizatorilor la toate funcționalitățile necesare generării și ajustării unui orar complet.
- Accelerarea procesului de generare a orarului: Comparativ cu metodele tradiționale de planificare (precum utilizarea fișierelor Excel sau completarea manuală), aplicația permite construirea unui orar coerent într-un interval de timp mult mai scurt, crescând semnificativ eficiența personalului administrativ.
- Aplicare statică bazată pe reguli și constrângeri instituționale: Motorul de generare ține cont de regulile rigide impuse de instituție, cum ar fi evitarea suprapunerii activităților, alocarea corectă a sălilor, respectarea disponibilității profesorilor și a structurii grupelor și subgrupelor, minimizând astfel erorile și conflictele.
- Asigurarea flexibilității în reorganizarea orarului: Modificările punctuale sau de ansamblu pot fi realizate fără a compromite structura generală a orarului.
- Export facil în formate lizibile și compatibile: Orarul generat poate fi exportat în formate comune și ușor de distribuit (PDF, Excel), fără a necesita intervenții ulterioare de rearanjare sau conversie. Acest aspect facilitează publicarea orarului pe platforme interne sau afișarea acestuia în format tipărit.

Utilizarea unei astfel de aplicații nu doar că optimizează munca personalului academic, ci contribuie direct la creșterea transparenței și previzibilității programului pentru studenți și cadre didactice. Prin reducerea erorilor umane și a timpului de lucru necesar planificării, instituția beneficiază de un proces mai clar, mai rapid și mai puțin susceptibil la modificări neplanificate.

Astfel, prin adoptarea unui model unitar și centralizat de utilizare, aplicația propusă reușește să simplifice întregul proces de generare a orarului, oferind un control deplin asupra datelor și regulilor implicate. Lipsa unui sistem multi-utilizator este o alegere intenționată, orientată spre eficiență și operativitate în contextul instituțiilor cu resurse limitate, unde un singur responsabil poate gestiona întreaga activitate.

#### 1.4 Identificarea operațiilor efectuate de utilizator în cadrul aplicației

Aplicația de generare automată a orarului este concepută pentru a oferi utilizatorului control complet asupra întregului flux de creare, personalizare și export al unui orar academic valid. Printr-o interfață intuitivă, fiecare operație realizată de utilizator contribuie direct la formarea unei structuri educaționale coerente, respectând regulile și constrângerile impuse de instituție. Toate funcționalitățile sunt centralizate

într-un panou unic, adaptat unui sistem cu un singur utilizator, fără roluri distincte sau drepturi diferențiate.

În prima etapă, utilizatorul introduce toate datele relevante necesare pentru generarea orarului. Aceste date includ configurarea anilor de studiu, grupelor și subgrupelor, introducerea sălilor de activitate (de tip curs, seminar, laborator și proiect), precum și definirea completă a cadrelor didactice – cu numele și prenumele, disciplinele predate, tipurile de activități, nivelurile de studii la care vor preda disciplina respectivă și disponibilitate detaliată. De asemenea, utilizatorul are posibilitatea de a crea, modifica sau șterge seturi de reguli care ghidează procesul de generare automată a orarului. Aceste reguli sunt redactate în format structurat și salvat în baza de date pentru a fi reutilizate.

Ulterior, utilizatorul declanșează procesul de generare a orarului, alegând între două metode disponibile: algoritmul clasic implementat local sau generatorul bazat pe inteligență artificială (GPT-4). Indiferent de metodă, utilizatorul nu intervine direct în procesul de alocare a activităților, însă are control asupra datelor de intrare și poate relansa procesul cu noi condiții, dacă este necesar. După generare, utilizatorul analizează orarul rezultat și poate opta pentru salvarea acestuia, redenumirea sa, încărcarea unui orar anterior sau ștergerea versiunilor vechi care nu mai sunt relevante.

În final, aplicația oferă funcționalități extinse pentru vizualizarea orarului într-un format tabelar, filtrarea acestuia după nivelul de studii și anul academic, precum și exportul în formate uzuale – PDF sau Excel. Aceste operații sunt realizate direct de utilizator prin intermediul interfeței grafice, fără a necesita cunoștințe tehnice avansate.

Prin urmare, operațiile realizate de utilizator acoperă întregul spectru de activități: introducerea, modificarea și ștergerea datelor, setarea regulilor de generare, alegerea metodei de generare, gestionarea versiunilor de orar și exportul acestora, reflectând o interacțiune completă, coerentă și facilă cu aplicația.

## CAPITOLUL 2. SPECIFICAREA CERINȚELOR

Aplicația propusă are ca obiectiv principal furnizarea unei soluții digitale moderne pentru generarea automată a orarului, utilizând reguli și constrângeri prestabilite, precum și date reale extrase din baza de date (ani de studiu, grupe și subgrupe, săli, cadre didactice, disponibilitatea cadrelor, discipline). Procesul tradițional de creare a orarelor academice este adesea anevoios, consumator de timp, predispus la erori și dificil de adaptat la modificări ulterioare.

Prin această aplicație, se urmărește automatizarea întregului flux de planificare, oferind utilizatorului o interfață intuitivă și un set complet de funcționalități pentru introducerea datelor, generarea și validarea unui orar coerent, complet și adaptat cerințelor instituționale.

### 2.1 Cerințe privind gestionarea și prelucrarea datelor

Pentru a asigura un proces complet de generare a orarului, aplicația trebuie să gestioneze în mod eficient toate datele esențiale implicate în organizarea academică. Această componentă a sistemului vizează operațiile necesare pentru introducerea, modificarea, căutarea și stocarea datelor referitoare la grupe, săli, profesori și reguli de generare. Gestionarea corectă a acestor informații constituie baza funcțională pentru o planificare coerentă și automatizată a activităților didactice.

Sistemul va permite următoarele operații:

#### 1. Introducerea datelor:

- Definirea grupelor și subgrupelor, incluzând informații precum anul de studiu, denumirea și nivelul (Licență/Master);
- Înregistrarea sălilor disponibile, cu specificarea tipului (curs, seminar, laborator, proiect);
- Adăugarea cadrelor didactice, împreună cu disciplinele predate, nivelurile de predare, tipurile de activități asociate și intervalele de disponibilitate;
- Stabilirea regulilor de generare, care includ intervale orare permise, durate de pauză, programul general și formatul afișării activităților în orar.

#### 2. Modificarea datelor:

- Actualizarea disciplinelor asociate unui profesor;
- Modificarea disponibilității acestuia;
- Editarea sălilor de curs;
- Restructurarea grupelor și subgrupelor existente;
- Încărcarea și editarea regulilor din baza de date.

#### 3. Căutare și filtrare:

- Căutarea rapidă după numele profesorului, grupei sau disciplinei;
- Filtrarea după nivel de studiu, an, grupă, subgrupă.

#### 4. Gestionarea orarelor salvate:

- Salvarea orarelor generate în baza de date;
- Reîncărcarea și consultarea ulterioară a acestora;



- Vizualizarea unui istoric al orarelor create anterior.

Prin centralizarea acestor operații într-un sistem unitar și accesibil, aplicația oferă utilizatorului control complet asupra tuturor datelor necesare generării orarului. Flexibilitatea în actualizarea și filtrarea informațiilor, alături de posibilitatea de salvare și reutilizare a datelor introduse, contribuie la optimizarea procesului de planificare academică și la reducerea semnificativă a timpului alocat gestionării manuale a acestora.

## 2.2 Cerințe privind generarea și validarea orarului

După colectarea și structurarea datelor relevante, etapa centrală a aplicației constă în generarea efectivă a orarului. Această funcționalitate presupune alocarea inteligentă și automată a activităților didactice în funcție de reguli prestabilite, disponibilitatea resurselor și tipologia activităților. În paralel, sistemul trebuie să includă mecanisme de validare a orarului rezultat, pentru a detecta posibile conflicte și incoerențe. Astfel, sunt asigurate coerența, fezabilitatea și calitatea planificării orarelor generate.

Aplicația va include funcționalități avansate pentru generarea și validarea orarului:

### 1. Generarea orarului:

- Selectarea metodei de generare: Utilizarea unui model AI (ex. GPT-4)/ utilizarea unui algoritm determinist clasic;
- Alocarea automată a activităților, ținând cont de disponibilități, tipuri de activități și regulile configurate de utilizator.

### 2. Validarea orarului:

- Suprapuneri de profesori sau săli;
- Lipsa activităților într-o zi sau existența unor intervale orare neacoperite;
- Desincronizări ale cursurilor comune la nivel de an.

### 3. Exportul orarului:

- Exportul local în format PDF sau Excel;
- Posibilitatea de partajare sau arhivare a orarului generat.

Prin integrarea acestor funcționalități, aplicația asigură nu doar automatizarea procesului de generare a orarului, ci și validarea riguroasă a rezultatului, eliminând conflictele și optimizând distribuirea resurselor. Posibilitatea de a exporta orarul în formate accesibile și reutilizabile consolidează caracterul practic al aplicației și facilitează utilizarea directă a rezultatelor în mediul academic.

Prin definirea clară a cerințelor privind atât gestionarea datelor, cât și procesul de generare și validare a orarului, aplicația propusă se conturează ca un instrument complet și eficient pentru planificarea academică. Îmbinând funcționalități avansate de introducere, actualizare și filtrare a datelor cu algoritmi inteligenți de generare și mecanisme automate de verificare a conflictelor, sistemul oferă un mediu unificat, fiabil și ușor de utilizat.

## CAPITOLUL 3. ARHITECTURA ȘI PROIECTAREA SISTEMULUI

Proiectarea aplicației pentru planificare inteligentă are ca obiectiv dezvoltarea unui sistem informatic capabil să genereze automat orare, prin integrarea tehnologiilor moderne de inteligență artificială. Această etapă presupune definirea unei arhitecturi robuste, care să gestioneze eficient componentele esențiale ale aplicației – interfața utilizatorului (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.

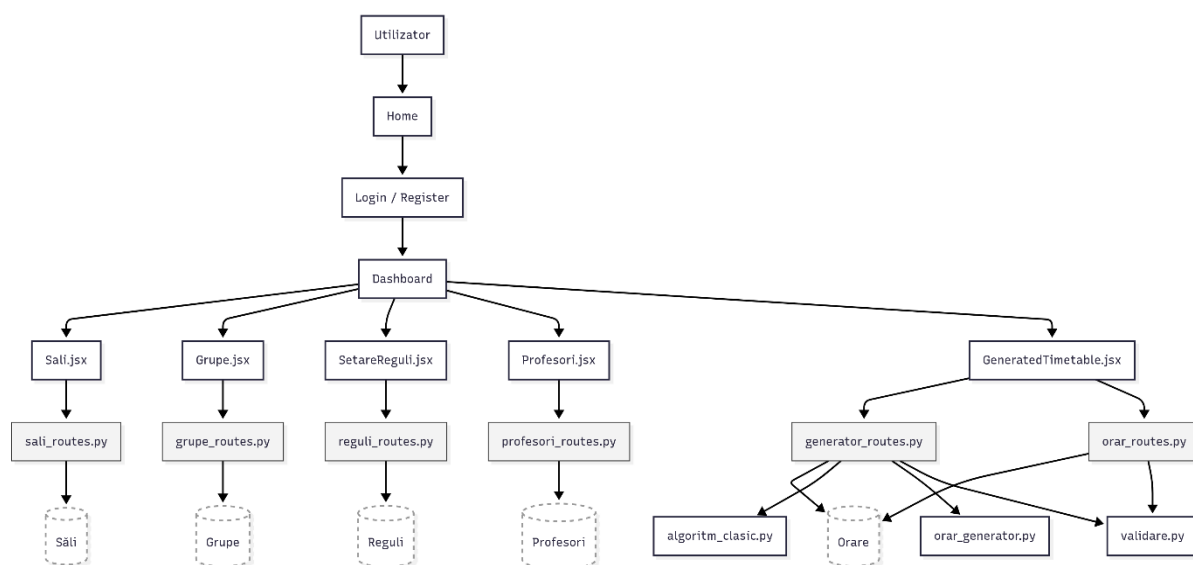


Fig. 4 Diagrama arhitecturală a aplicației

### 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.

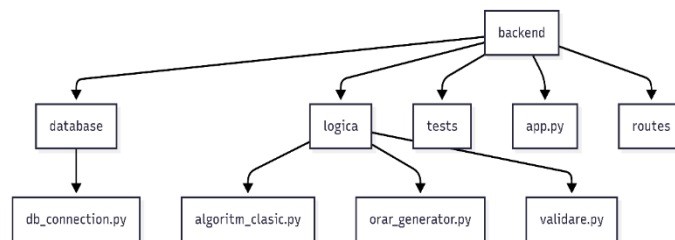


Fig. 5 Diagrama pachetelor- Backend

2. 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.

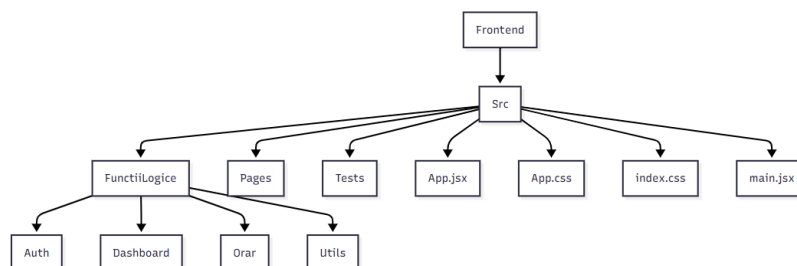


Fig. 6 Diagrama pachetelor- Frontend

3. 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.

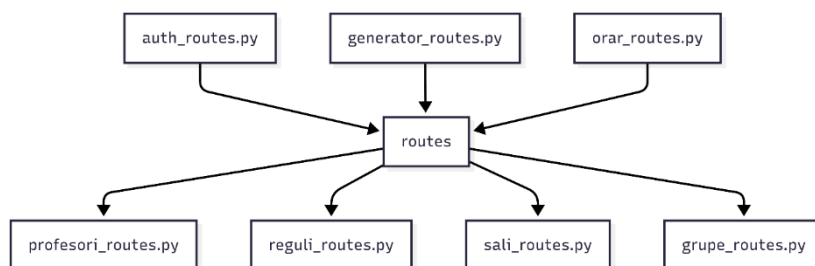


Fig. 7 Diagrama pachetelor- Routes

4. 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.

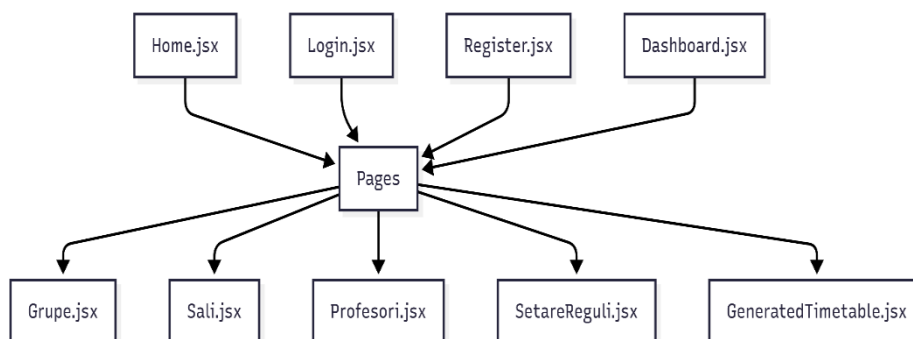


Fig. 8 Diagrama pachetelor- Pages

### 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 [8]. 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 explicite, 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.

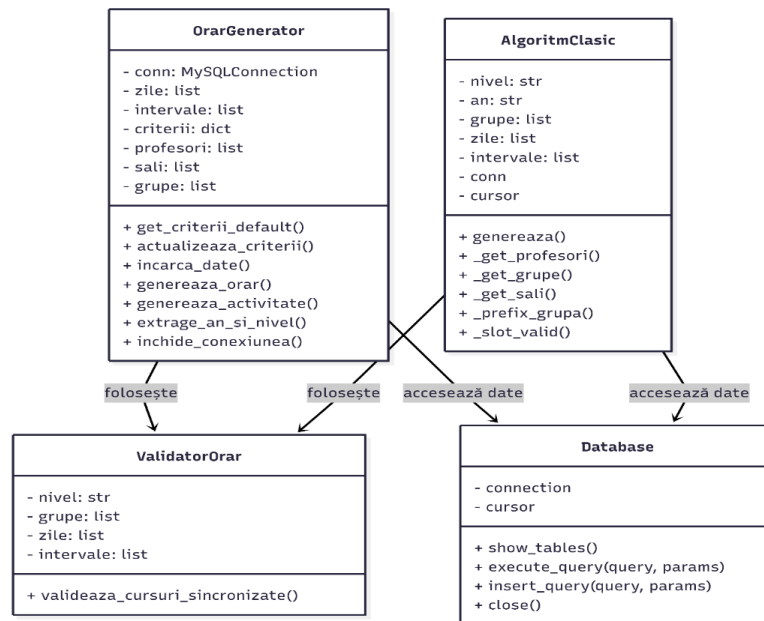


Fig. 9 Diagrama clasei -Backend

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.

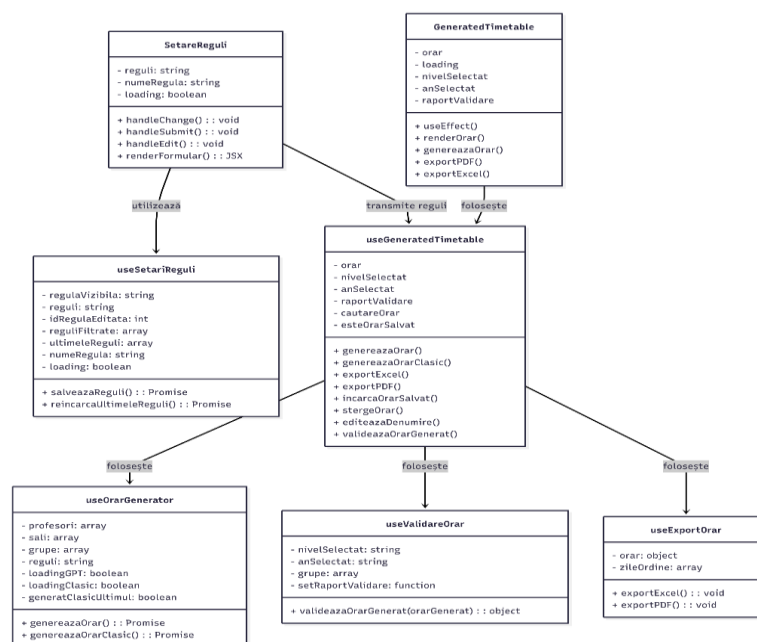


Fig. 10 Diagrama claselor- Frontend

### 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 [9].

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.

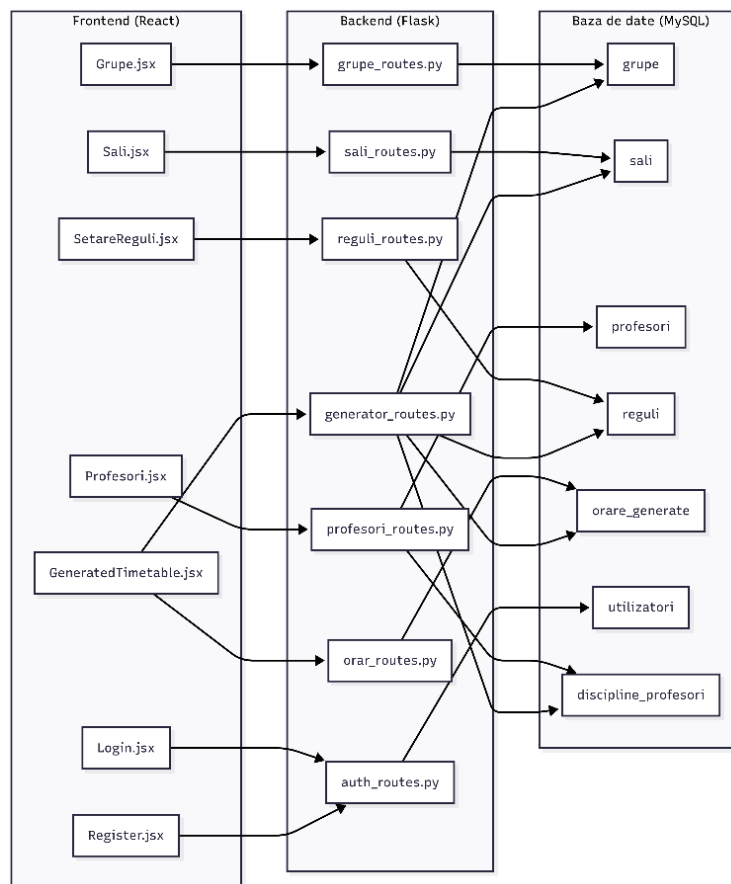


Fig. 11 Diagrama de componente ale aplicației

### 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 [10].

#### 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.

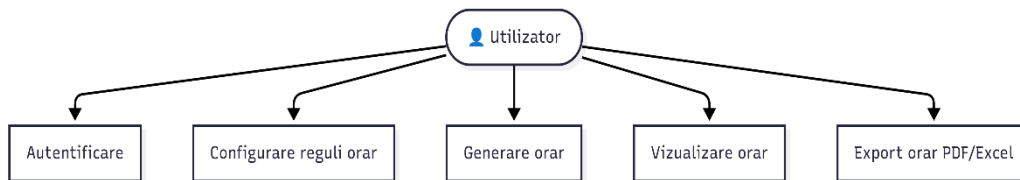


Fig. 12 Diagrama cazurilor de utilizare

### 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.

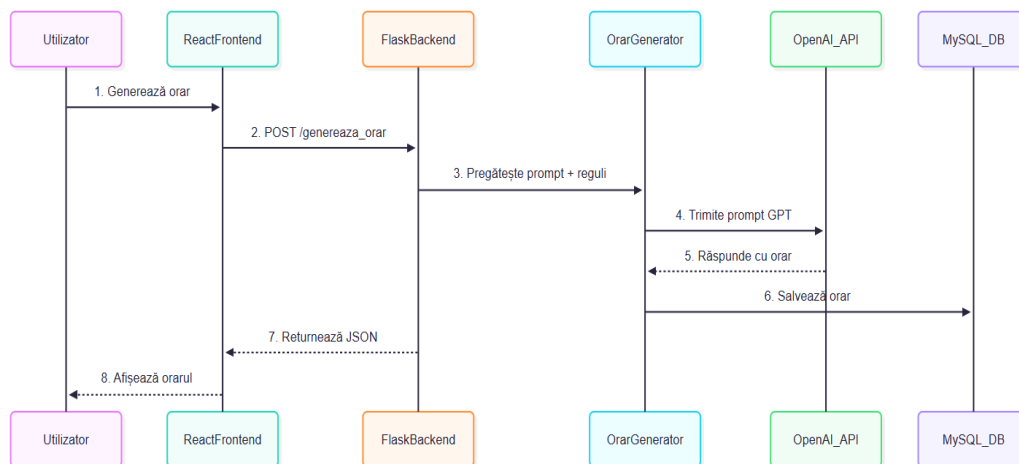


Fig. 13 Diagrama de secvență

### 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 generat.

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.

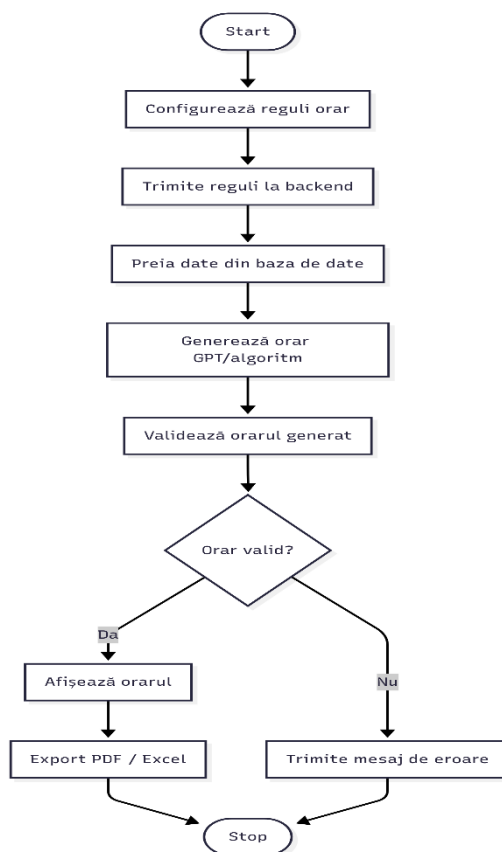


Fig. 14 Diagrama de activitate

În ansamblu, proiectarea sistemului a urmărit dezvoltarea unei arhitecturi clare, modulare și scalabile, capabilă să susțină un proces complex de generare automată a orarului universitar. Separarea componentelor în interfață (frontend), logică de procesare (backend) și gestiunea datelor (bază de date) a permis o organizare eficientă a aplicației, facilitând extinderea și întreținerea acesteia.

Integrarea inteligenței artificiale ca mecanism de generare a orarului aduce un plus semnificativ de valoare, automatizând procese care anterior necesitau intervenție umană repetitivă. Utilizarea diagramelor UML a contribuit la o înțelegere aprofundată a structurii aplicației, a interacțiunilor dintre componente și a comportamentului sistemului în raport cu acțiunile utilizatorului.

Această etapă de proiectare a reprezentat fundamentul necesar pentru o implementare coerentă și eficientă a aplicației, asigurând premisele unui sistem robust, flexibil și adaptat cerințelor actuale din mediul academic.



## CAPITOLUL 4. IMPLEMENTAREA ȘI DESCRIEREA APLICAȚIEI

Acest capitol oferă o imagine de ansamblu asupra modului în care a fost dezvoltată aplicația pentru generarea automată a orarului universitar, utilizând tehnologii moderne și inteligența artificială. Procesul de implementare s-a desfășurat în cadrul editorului Visual Studio Code, un mediu de dezvoltare versatil și extensibil, ideal pentru proiecte web de tip full-stack. Acesta a facilitat organizarea modulară a codului, integrarea extensiilor pentru depanare și conectarea rapidă cu serviciile externe necesare (baze de date, API-uri, modele AI).

Sunt prezentate în continuare limbajele și tehnologiile utilizate, pașii de implementare, funcționalitățile-cheie dezvoltate și dificultățile întâmpinate. Capitolul se încheie cu o comparație între abordarea modernă, automatizată, și metoda clasică de creare a orarului, pentru a evidenția avantajele aduse de digitalizarea acestui proces.

### 4.1. Limbajele de programare implicate în dezvoltarea aplicației

În dezvoltarea aplicației pentru generarea automată a orarului au fost utilizate mai multe limbaje de programare, alese în funcție de specificul fiecărei componente și de cerințele proiectului. Aplicația este structurată pe o arhitectură de tip client – server, ceea ce a impus utilizarea unor tehnologii distincte pentru partea de interfață și pentru logica de procesare din spate.

#### 4.1.1 JavaScript

JavaScript este un limbaj de programare interpretat, dinamic și orientat pe evenimente, utilizat pe scară largă în dezvoltarea aplicațiilor web. În această aplicație, a fost folosit pentru realizarea interfeței frontend, datorită capacității sale de a crea componente interactive și rapide [11].

Biblioteca React.js, bazată pe JavaScript, permite construirea de componente reutilizabile care răspund dinamic la modificările de stare. JavaScript gestionează introducerea datelor, afișarea orarului generat și comunicarea asincronă cu backend-ul în Python, prin cereri HTTP.

De asemenea, facilitează integrarea notificărilor vizuale (toast, alerte), validarea automată a orarului, exportul în PDF/Excel și navigarea între pagini. Figura 15 prezintă un exemplu de utilizare a hook-ului `useEffect` pentru actualizarea raportului de validare în funcție de starea orarului.

```
useEffect(() => {  
  if (orar && nivelSelectat && anSelectat && grupe.length > 0) {  
    const raport = valideazaOrarGenerat(orar);  
    setRaportValidare(raport);  
  }  
}, [orar, nivelSelectat, anSelectat, grupe]);
```

Fig. 15 Exemplu de cod JavaScript folosit în aplicație

### 4.1.2 Python

Python este un limbaj de programare de nivel înalt, recunoscut pentru sintaxa sa clară și expresivă, ideal pentru dezvoltarea rapidă a aplicațiilor. Datorită flexibilității și ecosistemului extins, este larg utilizat în dezvoltare web, inteligență artificială și prelucrarea datelor [12].

În această aplicație, Python a fost folosit pentru dezvoltarea backend-ului, împreună cu framework-ul Flask. Acesta gestionează logica aplicației, comunicarea cu baza de date și integrarea cu serviciul AI - OpenAI GPT-4. Datele sunt preluate din interfața React, procesate și validate, apoi orarul generat este returnat în format JSON.

Python a fost ales pentru simplitatea utilizării, compatibilitatea cu tehnologiile AI și suportul puternic pentru biblioteci web și de lucru cu baze de date. Figura 16 exemplifică organizarea grupelor și subgrupelor în `orar_generator.py`, folosind structuri eficiente precum `defaultdict` pentru pregătirea datelor necesare generării orarului.

```
def genereaza_orar(self):
    from collections import defaultdict
    import copy

    self.incarca_date()
    orar = {}
    grupe_pe_an = defaultdict(set)
    subgrupe_map = defaultdict(set)
    cursuri_generate_pe_an = defaultdict(set)

    for g in self.grupe:
        den = g["denumire"]
        nivel, an = self.mapare_grupe[den]
        grupa_baza = self.grupa_si_subgrupa.get(den, {}).get("grupa_baza", "")
        grupe_pe_an[f"{nivel}-{an}"].add(den)
        if "subgrupa":
            subgrupe_map[grupa_baza].add(den)
```

Fig. 16 Exemplu de cod Python folosit în aplicație

### 4.1.3 SQL

SQL (Structured Query Language) este limbajul utilizat pentru interogarea și manipularea bazelor de date relaționale. În această aplicație, SQL este folosit pentru gestionarea datelor din baza de date MySQL, care include informații despre profesori, grupe, săli și reguli [13]. Figura 17 prezintă clasa `Database`, care centralizează metodele de conectare și interogare, fiind utilizată în întregul sistem pentru lucrul cu datele.

```
import mysql.connector

class Database:
    def __init__(self):
        self.connection = mysql.connector.connect(
            host="localhost",
            user="root",
            password="",
            database="licenta",
            port=3306
        )
        self.cursor = self.connection.cursor()

    def show_tables(self):
        self.cursor.execute("SHOW TABLES")
        return self.cursor.fetchall()

    def execute_query(self, query, params=None):
        self.cursor.execute(query, params or ())
        return self.cursor.fetchall()

    def insert_query(self, query, params=None):
        self.cursor.execute(query, params or ())
        self.connection.commit()

    def close(self):
        self.cursor.close()
        self.connection.close()
```

Fig. 17 Exemplu de cod SQL folosit în aplicație

Prin comenzi SQL, aplicația poate introduce, actualiza și extrage datele necesare generării orarului, asigurând coerența și relaționarea corectă a acestora. În backend-ul scris în Python, interacțiunea cu baza de date este realizată prin modulul `mysql.connector`, care permite executarea comenzilor SQL într-un mod sigur și eficient.

#### 4.1.4 HTML5

HTML5 este versiunea modernă a limbajului de marcare utilizat pentru structurarea conținutului web și stă la baza afișării vizuale a aplicației. Deși interfața este dezvoltată în React.js, toate componentele JSX sunt convertite automat în elemente HTML5 standard, interpretate de browser [14].

Aplicația beneficiază astfel de o structură semantică clară, folosind elemente precum butoane, formulare, tabele și alerte. Acestea asigură compatibilitate cu toate browserele moderne și respectă standardele de accesibilitate prin atribute precum `role`, `aria-hidden` sau `disabled`.

Figura 18 prezintă un exemplu de buton HTML5 definit în JSX, utilizat pentru declanșarea generării orarului. În timpul procesului, butonul afișează un loader animat și oferă un mesaj informativ. Deși scris în JSX, codul este transpus în HTML5, permițând integrarea fluentă cu CSS și JavaScript pentru o experiență interactivă și modernă.

```
<button
  className="btn btn-success"
  onClick={genereazaOrar}
  disabled={loadingGPT || loadingClasic}
>
  {loadingGPT ? (
    <>
      <span className="spinner-border spinner-border-sm me-2" role="status" aria-hidden="true"></span>
      Se generează...
    </>
  ) : (
    "🚀 Generează orar cu AI"
  )}
</button>
```

Fig. 18 Exemplu de cod HTML5 folosit în aplicație

#### 4.1.5 CSS3

CSS3 (Cascading Style Sheets, versiunea 3) este limbajul utilizat pentru definirea stilurilor vizuale în aplicațiile web, permițând personalizarea aspectului, a layout-ului și a comportamentului responsive al interfeței. În cadrul acestei aplicații, CSS3 este utilizat pentru a crea o experiență vizuală coerentă, intuitivă și adaptabilă diferitelor dimensiuni de ecran [15].

Stilizarea este realizată în principal prin intermediul framework-ului Bootstrap, care oferă o suită extinsă de clase predefinite pentru butoane, tabele, carduri și formulare. Acestea aplică rapid stiluri moderne privind culorile, marginile, umbrele sau spațierea (Figura 19).

```
<button className="btn btn-success">Generează orar cu AI</button>
<table className="table table-bordered text-center align-middle"></table>
<div className="alert alert-info">Orarul a fost generat cu succes.</div>
```

Fig. 19 Exemplu de cod CSS3 – prin clase - folosit în aplicație

Complementar, pentru ajustări specifice fiecărei componente, sunt utilizate stiluri inline scrise direct în JSX, care permit un control rapid asupra proprietăților CSS esențiale (Figura 20).

```
<div style={{ fontSize: "0.75rem", maxWidth: "360px" }}></div>
```

Fig. 20 Exemplu de cod CSS3 – prin stiluri inline - folosit în aplicație

Fișierul App.css completează această abordare prin aplicarea de stiluri CSS globale – precum animații, tranziții sau personalizarea fonturilor aplicației – contribuind la o coerență vizuală extinsă.

## 4.2. Framework-urile și bibliotecile implicate în dezvoltarea aplicației

Pe lângă limbajele de programare utilizate (JavaScript, Python, SQL), dezvoltarea aplicației a implicat integrarea mai multor tehnologii moderne care susțin atât interfața utilizatorului, cât și funcționalitatea serverului și comunicarea între componente.

Aceste tehnologii au fost alese pentru eficiență, flexibilitate, compatibilitate cu inteligența artificială și pentru a permite o dezvoltare rapidă și modulară.

### 4.2.1 React.js

React.js este o bibliotecă JavaScript dezvoltată de Meta, utilizată pentru crearea interfețelor web moderne, interactive și reactive. În această aplicație, React gestionează partea de frontend, oferind o arhitectură modulară, ușor de întreținut și extins [16].

Prin hook-urile `useState` și `useEffect`, aplicația actualizează automat interfața în funcție de modificările de stare, permițând încărcarea dinamică a profesorilor, sălilor, grupelor și afișarea în timp real a orarului generat.

Figura 21 ilustrează un custom hook (`useOrarGenerator`), folosit pentru preluarea datelor din backend și menținerea sincronizării acestora în mai multe componente, facilitând interacțiunea fluentă cu utilizatorul.

```
import { useState, useEffect } from "react";

const useOrarGenerator = () => {
  const [profesori, setProfesori] = useState([]);
  const [sali, setSali] = useState([]);

  useEffect(() => {
    const incarcaDate = async () => {
      const res = await fetch("http://localhost:5000/date_orar");
      const data = await res.json();
      setProfesori(data.profesori || []);
      setSali(data.sali || []);
    };

    incarcaDate();
  }, []);

  return { profesori, sali };
};
```

Fig. 21 Exemplu de cod React folosit în aplicație

### 4.2.2 Flask

Flask este un microframework web scris în Python, recunoscut pentru simplitatea și flexibilitatea sa în dezvoltarea rapidă a aplicațiilor web [17]. În cadrul acestei aplicații, Flask este utilizat pentru gestionarea logicii de backend, comunicarea cu baza de date și procesarea cererilor primite de la interfața React.

Aplicația definește rute specifice care primesc date din frontend, le prelucreză cu ajutorul algoritmilor de generare și returnează răspunsuri în format JSON sau HTML. De asemenea, Flask permite integrarea cu baza de date MySQL, prin intermediul modulului `mysql.connector`, facilitând astfel accesul eficient la datele despre profesori, săli, grupe și orare salvate.

În Figura 22 este ilustrat un exemplu de rută Flask care primește datele din formular, generează un orar complet, îl filtrează după an și nivel, validează structura acestuia și returnează rezultatul sub formă de pagină HTML. Alte rute importante din aplicație includ funcționalități precum generarea orarului prin algoritmul propriu sau salvarea și încărcarea datelor academice în format JSON.

```
@generator_bp.route("/genereaza_orar_propriu", methods=["GET", "POST"])
def genereaza_orar_propriu():
    generator = OrarGenerator()
    nivel_selectat = "Licenta"
    an_selectat = "I"

    if request.method == "POST":
        nivel_selectat = request.form.get("nivel", "Licenta")
        an_selectat = request.form.get("an", "I")
        generator.actualizeaza_criterii(request.form)

    orar_complet = generator.genereaza_orar()

    orar_filtrat = {
        grupa: continut
        for grupa, continut in orar_complet.items()
        if generator.extrage_an_si_nivel(grupa) == (nivel_selectat, an_selectat)
    }

    raport_validare = valideaza_orar(orar_filtrat)
    html = genereaza_html(orar_filtrat, generator.criterii, "") + raport_validare
    generator.inchide_conexiunea()

    return render_template_string(html)
```

Fig. 22 Exemplu de cod Flask folosit în aplicație

#### 4.2.3 Integrarea OpenAI GPT-4 în backend-ul Flask

GPT-4 este un model avansat de inteligență artificială generativă, dezvoltat de OpenAI, utilizat în această aplicație pentru generarea automată a orarelor în format JSON valid[18]. Este implementată versiunea GPT-4o, integrată în backend-ul Flask prin intermediul pachetului oficial `openai` și al unei chei API securizate.

Modelul este apelat prin metoda `chat.completions.create()`, folosind un prompt construit din datele introduse de utilizator. Răspunsul generat este exclusiv în format JSON, fără explicații suplimentare, pentru a putea fi procesat direct în interfața aplicației.

Figura 23 prezintă un exemplu de cod care ilustrează această integrare. Răspunsul primit este transformat într-un orar complet și coerent, afișabil pentru utilizator. Astfel, aplicația valorifică capacitatea modelului de a înțelege și interpreta limbajul uman pentru a genera automat soluții personalizate, eliminând necesitatea unui algoritm clasic de planificare.

```
response = client.chat.completions.create(
    model="gpt-4o",
    messages=[
        {
            "role": "system",
            "content": "Răspunde DOAR cu JSON VALID, fără explicații..."
        },
        {
            "role": "user",
            "content": prompt_frontend
        }
    ]
)
orar_raw = response.choices[0].message.content.strip()
```

Fig. 23 Integrarea OpenAI GPT-4 în backend-ul Flask

#### 4.2.4 Tehnologii auxiliare utilizate în aplicație

Pentru dezvoltarea interfeței și funcționalităților aplicației, au fost integrate mai multe biblioteci și instrumente moderne. Bootstrap 5 a fost folosit pentru realizarea unei interfețe responsive, bazată pe grid și componente UI predefinite (Figura 24).

```
<nav className="navbar navbar-expand-lg bg-white shadow-sm px-4 py-3 w-100">
  <div className="container-fluid d-flex justify-content-between align-items-center">
    <Link to="/" className="navbar-brand fw-bold fs-5 text-primary">
      Aplicație pentru planificare inteligentă utilizând tehnici de A.I.
    </Link>
    <div className="ms-auto">
      <Link to="/login" className="btn btn-outline-primary">
        Autentificare
      </Link>
    </div>
  </div>
</nav>
```

Fig. 24 Exemplu de cod Bootstrap folosit în aplicație

Pentru trimiterea cererilor HTTP din frontend, s-a utilizat Fetch API, care permite interacțiuni asincrone fără reîncărcarea paginii (Figura 25).

```
const response = await fetch("http://127.0.0.1:5000/genereaza_orar", {
  method: "POST",
  headers: { "Content-Type": "application/json" },
  body: JSON.stringify({
    regula_id: regula_id,
    an_selectat: anSelectat,
    nivel_selectat: nivelSelectat,
    grupe_selectate: grupe
      .filter(g => g.an === anSelectat && g.nivel === nivelSelectat)
      .map(g => g.denumire),
    prompt: promptFinal
  }),
});
```

Fig. 25 Exemplu de cod Fetch folosit în aplicație

MySQL a asigurat stocarea structurată a datelor (profesori, săli, grupe, reguli, orare), comunicarea cu backend-ul Flask fiind realizată prin mysql.connector. Mesaje de confirmare și alertele personalizate sunt gestionate cu SweetAlert2, care înlocuiește alertele browserului cu ferestre moderne. Exportul orarului în format Excel este realizat cu SheetJS (xlsx), fiecare grupă fiind salvată într-un sheet distinct, iar exportul în PDF se face prin html2pdf.js, care transformă conținutul HTML într-un document A4 stilizat.

#### 4.3. Implementarea și funcționalitatea aplicației

Aplicația propusă are ca scop generarea automată a orarelor prin utilizarea tehnicilor de inteligență artificială. Este o aplicație web interactivă care permite introducerea și gestionarea datelor educaționale (profesori, săli, grupe, reguli), iar apoi generează automat orare săptămânale complete, coerente și conforme cu cerințele academice.

Aplicația dezvoltată pentru generarea automată a orarului are o arhitectură modulară de tip client-server, având următoarele componente: Frontend, Backend, Bază de date – MySQL, AI Engine – OpenAI GPT-4 (sau alternativ model local).

### 4.3.1 Configurarea mediului de dezvoltare

Realizarea aplicației a început cu pregătirea și configurarea unui mediu de dezvoltare modern, scalabil și ușor de întreținut. Mediul este împărțit în două componente principale – frontend (interfața utilizator) și backend (logica aplicației și comunicația cu baza de date și motorul AI).

#### Frontend – React.js cu Vite

- Tehnologie principală: React.js – o bibliotecă JavaScript pentru crearea interfețelor web moderne, bazată pe componente reutilizabile și un model reactiv.
- Bundler/Toolchain: Vite – ales în locul clasicului Webpack pentru viteze superioare de dezvoltare și build.
- Structură componentizată: Fiecare entitate are propriul fișier: Profesori.jsx, Sali.jsx, Grupe.jsx, SetareReguli.jsx, GeneratedTimetable.jsx. Logica separată în hooks personalizați: useProfesoriLogic.js, useSaliLogic.js, useSetariReguliLogic și altele.
- Stilizare: utilizarea combinată a Bootstrap și Tailwind CSS pentru un aspect modern și responsive.
- Alte biblioteci frontend: react-toastify, sweetalert2 – notificări/toasturi; html2pdf.js – export în PDF; xlsx (SheetJS) – export în Excel.

#### Backend – Flask (Python)

- Framework: Flask – microframework Python minimalist și flexibil.
- Structurare modulară: Backendul este împărțit în module (Blueprints), fiecare gestionând o categorie de date: routes/profesori\_routes.py, routes/sali\_routes.py, routes/grupe\_routes.py, routes/reguli\_routes.py, routes/orar\_routes.py, routes/generator\_routes.py – integrează AI.
- Middleware: Flask-CORS pentru a permite comunicarea între frontend (localhost:5173) și backend (localhost:5000).
- Gestionare variabile de mediu: Se folosește python-dotenv pentru încărcarea cheilor API și a configurației (ex: OPENAI\_API\_KEY) din fișierul .env.

#### Bază de date – MySQL

- Tip SGBD: MySQL 8 – sistem de gestiune a bazelor de date relaționale, cu suport pentru tranzacții și integritate referențială.
- Interfață de administrare: utilizarea aplicațiilor precum *phpMyAdmin* sau *MySQL Workbench* pentru gestionarea tabelelor.
- Tabele principale: profesori, discipline, activitati, Sali, grupe, subgrupe, reguli – stochează reguli în format JSON, orare – salvează orarele generate
- Conexiune: realizată prin mysql-connector-python, cu funcții generice în fișierul db\_connection.py.

### Integrare AI – GPT-4 / Algoritm clasic în Python

Aplicația suportă două metode de generare automată a orarului:

#### a) Generare cu AI – OpenAI GPT-4

- Integrare API: Se utilizează SDK-ul oficial openai în Python. Cheia API este stocată în fișierul .env și încărcată cu dotenv.
- Generare prompt: La apelul endpointului /genereaza\_orar, backendul: preia datele din baza de date (profesori, sali, grupe, reguli); generează automat un prompt detaliat cu toate cerințele academice (ex: cursuri comune pe an, laboratoare pe subgrupă, miercuri 14–16 pauză); trimite promptul către GPT-4.
- Răspunsul modelului: Este un obiect JSON cu structura completă a orarului, pe zile, ani și grupe. Înainte de afișare, răspunsul este validat pentru: structură corectă; respectarea regulilor (ex: fără suprapuneri, pauze, distribuție uniformă etc.).

#### b) Generare clasică – Algoritm propriu Python

- Implementare locală: Aplicația include o variantă proprie de algoritm euristic, scrisă în Python, fără utilizarea rețelelor neuronale. Acesta este definit într-un modul dedicat (ex: algoritm\_propriu.py) și apelat prin endpointul /genereaza\_algoritm\_propriu.
- Funcționare: Algoritmul preia aceleași date ca în varianta AI (profesori, săli, grupe, reguli). Aplică reguli fixe și constrângeri logice (ex: ocuparea sălilor disponibile, orar 08–20, pauze max. 2h). Prioritizează plasarea cursurilor, apoi a seminarelor și laboratoarelor, respectând ordinea logică și disponibilitatea.
- Avantaje: Permite debugging complet și modificări rapide în codul sursă. Este ideal pentru testarea și compararea rezultatelor cu cele generate de GPT-4.

Mod de selecție între cele două metode: Utilizatorul poate alege din interfața aplicației metoda dorită:

- „Generează cu AI (GPT-4)” → trimite datele către API-ul OpenAI.
- „Generează cu Algoritm Propriu” → folosește algoritmul local scris în Python.

### 4.3.2 Implementarea funcționalităților principale

Aplicația a fost dezvoltată pe baza unei arhitecturi modulare, organizată pe trei niveluri principale: interfața utilizatorului (frontend), logica de procesare și interacțiune (backend – Flask) și persistența datelor (baza de date – MySQL). Fiecare nivel gestionează operații esențiale precum introducerea, vizualizarea, modificarea și ștergerea datelor necesare generării automate a orarului universitar.

Tabelul de mai jos sintetizează funcționalitățile principale ale aplicației, evidențiind corespondențele dintre cele trei straturi arhitecturale.



Tabel 1 Implementarea funcționalităților principale

Funcționalitate	Frontend	Backend (Flask)	Bază de date (MySQL)
Gestionare profesori	Formular cu validare, listă profesori, editare/ștergere	/adauga_profesor, /toti_profesorii, /sterge_profesor, /actualizeaza_profesor	profesori, discipline, activitati, disponibilitate,
Gestionare săli	Adăugare automată coduri, categorii, ștergere/editare	/adauga_sali, /toate_sali, /sterge_sali, /actualizeaza_sala	sali (cod, tip),
Gestionare grupe	Formular denumire grupă, validare, filtrare, căutare	/adauga_grupa, /toate_grupe, /sterge_grupa	grupe, subgrupe (nivel, an),
Gestionare reguli	Editor reguli JSON, salvare/încărcare, selectare	/salveaza_reguli, /incarca_reguli, /sterge_reguli	reguli (id, denumire, continut_json),
Generare orar (GPT-4)	Buton generare AI, spinner, notificări	/genereaza_orar (prompt GPT-4, validare răspuns)	orare (continut_json, metoda, data),
Generare orar (algoritm propriu)	Buton generare local, afișare orar	/genereaza_algoritm_propriu (algoritm clasic Python)	orare (structură compatibilă algoritmului),
Afișare orar	Tabel orar cu selecție nivel/an/grupă, activități colorate	/incarca_orar, /valideaza_orar	orare (structură completă, validată),
Export orar (PDF/Excel)	Buton export PDF și Excel, toast succes	/export_orar_pdf, /export_orar_excel	n/a (export se face din frontend sau temporar),
Salvare/ștergere orare	Listă orare salvate, opțiune ștergere și încărcare	/sterge_orar, /salveaza_orar	orare (id, continut_json, data)"

Implementarea aplicației a fost realizată în mod modular, acoperind toate funcționalitățile necesare pentru generarea completă a orarelor academice, de la gestionarea datelor de bază până la integrarea cu un model AI performant. Structura clară pe cele trei niveluri – interfață, logică și stocare – asigură scalabilitate și mentenanță ușoară.

Totodată, deși aplicația este în prezent orientată spre platforme web, arhitectura adoptată permite, în perspectivă, extinderea către dispozitive mobile. O astfel de dezvoltare ar necesita o analiză tehnică prealabilă, incluzând studii comparative, selecția unui framework mobil adecvat (precum React Native sau Flutter) și definirea cerințelor specifice pentru a păstra experiența fluidă a utilizatorului.

### 4.3.3 Ghidul utilizării aplicației

Această secțiune prezintă modul concret de utilizare a aplicației web pentru generarea automată a orarelor, bazată pe tehnici de inteligență artificială. Scopul aplicației este de a oferi o soluție completă, modernă și ușor de folosit pentru personalul academic și administrativ implicat în planificarea activităților didactice.

Platforma a fost concepută pentru a ghida utilizatorul pas cu pas, de la autentificare și introducerea datelor despre cadrele didactice, săli și grupe, până la definirea regulilor de generare și obținerea unui orar valid, coerent și exportabil în format PDF sau Excel.

Ghidul este însoțit de capturi de ecran din interfața aplicației, pentru a facilita înțelegerea și utilizarea corectă a fiecărei funcționalități.

Prin acest ghid, utilizatorii aplicației vor putea înțelege și exploata întregul potențial al platformei, fără a necesita cunoștințe tehnice avansate.

#### Pasul 1 – Accesarea aplicației și autentificarea utilizatorului

La deschiderea aplicației web, utilizatorul este întâmpinat de o interfață modernă, unde este prezentat scopul platformei: generarea automată a orarului cu ajutorul Inteligenței Artificiale (Figura 26).

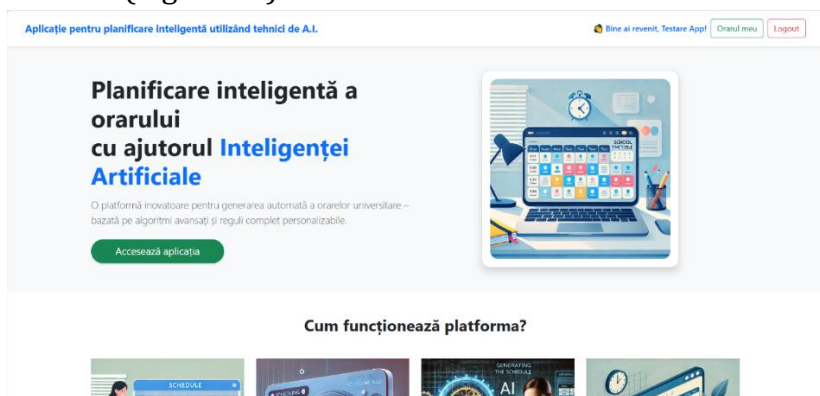


Fig. 26 Accesarea aplicației și autentificarea utilizatorului

În colțul din dreapta sus al interfeței se află opțiunile de autentificare și înregistrare. Accesul la funcționalitățile aplicației este permis doar utilizatorilor autentificați.

#### Autentificare

Dacă utilizatorul are deja un cont, va apăsa pe butonul „Autentificare”, fiind redirecționat către o pagină dedicată, unde va introduce adresa de email și parola (Figura 27).

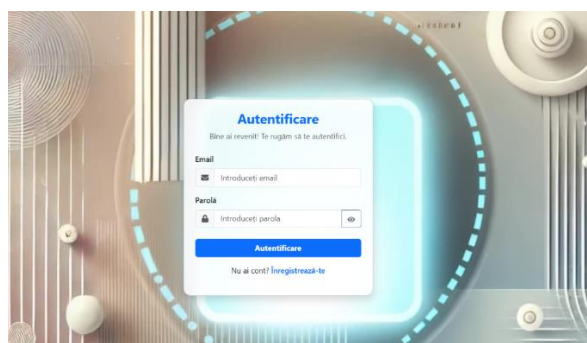


Fig. 27 Autentificare

După autentificare cu succes, utilizatorul este direcționat către home, de unde poate începe pașii către generarea orarului.

### Înregistrare

Dacă utilizatorul nu are încă un cont, poate accesa opțiunea „Înregistrează-te”, care îl redirecționează către formularul de creare cont. Aici trebuie să completeze: Numele complet, adresa de email, parola și confirmarea acesteia (Figura 28).

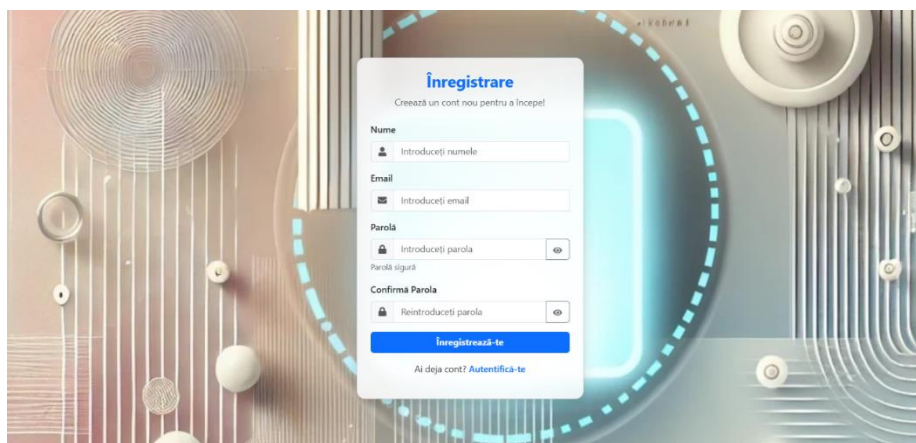



Fig. 28 Înregistrare

După înregistrare, utilizatorul este autentificat automat și poate accesa interfața completă a aplicației.

Pagina principală oferă și o prezentare succintă a pașilor necesari și a funcționalităților aplicației (Figura 29).



Fig. 29 Funcționalitățile aplicației

După ce utilizatorul s-a autentificat, poate începe generarea orarului apăsând pe butonul:  „Începe generarea orarului”. Acesta redirecționează către platforma completă, dashboard, unde se introduc datele și se configurează regulile (Figura 30).

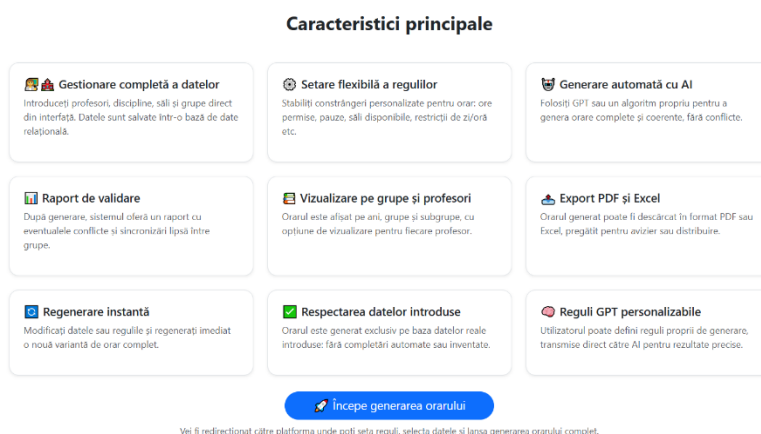


Fig. 30 Caracteristicile aplicației

## Pasul 2 – Dashboard-ul: alegerea secțiunii de configurare

După apăsarea butonului „Începe generarea orarului”, utilizatorul este redirecționat către pagina principală (Dashboard). Aceasta oferă o prezentare clară și structurată a pașilor necesari pentru generarea orarului (Figura 31).

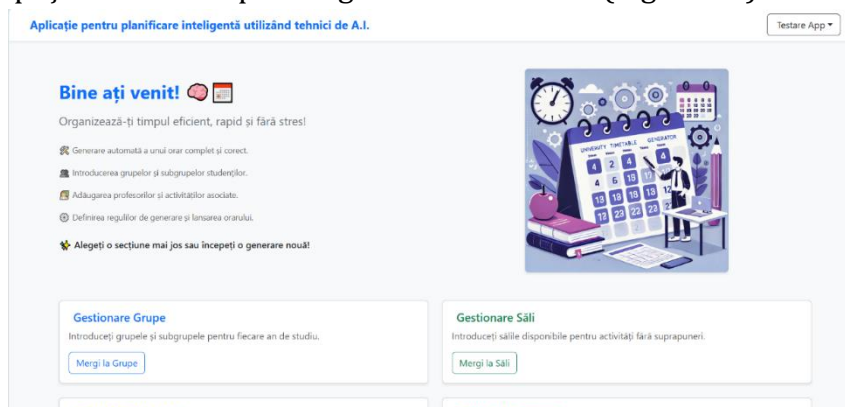


Fig. 31 Dashboard-ul aplicației

Utilizatorul poate selecta una dintre cele 4 secțiuni principale (Figura 32):

- Grupe – adăugare grupe și subgrupe;
- Săli – introducerea sălilor disponibile;
- Profesori – adăugare profesori, discipline și activități;
- Reguli – definirea regulilor personalizate.

Sub cardurile informative, este afișat un verificator vizual care indică dacă toate secțiunile necesare au fost completate.

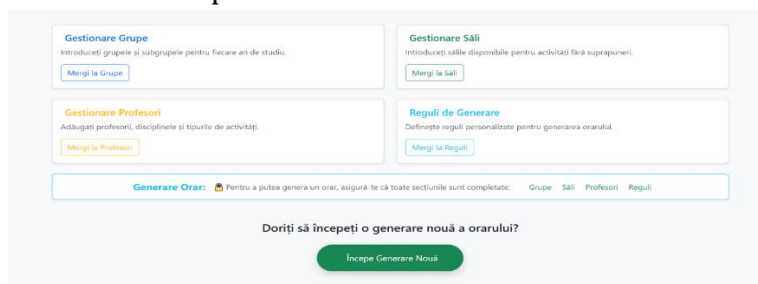


Fig. 32 Validare configurare completă

Când toate secțiunile sunt completate, se activează butonul: „Începe Generare Nouă”, care lansează procesul propriu-zis de generare a orarului.

### Pasul 3 – Gestionarea grupelor și subgrupelor

În această etapă, utilizatorul introduce grupele și subgrupele necesare pentru generarea orarului. Acestea sunt structurate pe nivel de studii (Licență/Master) și ani de studiu (Figura 33).

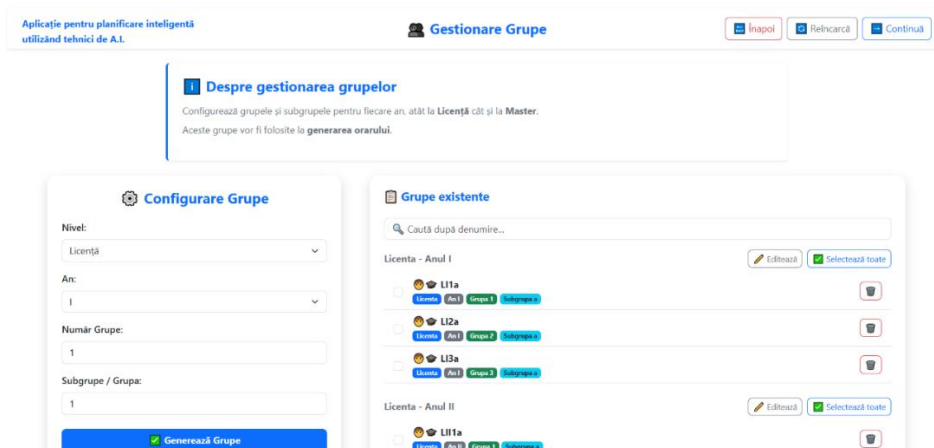


Fig. 33 Interfața pentru gestionarea grupelor

Funcționalități disponibile:

- Selectarea nivelului: Licență sau Master;
- Alegerea anului (I, II, III, IV);
- Introducerea numărului de grupe și subgrupe;
- Generarea automată a denumirilor (ex: LI1a, LI2b);
- Căutare rapidă după denumirea grupei;
- Editare manuală (adăugare individuală, ex: „2b”);
- Ștergere selectivă sau în bloc.

Toate grupele generate vor fi folosite ulterior la asocierea activităților în orar.

După completarea acestei secțiuni, utilizatorul poate apăsa pe „→ Continuă” pentru a trece la gestionarea sălilor.

### Pasul 4 – Gestionarea sălilor

În această etapă se introduc sălile disponibile pentru activitățile didactice (Figura 34). Aplicația acceptă 4 tipuri de săli, fiecare identificată printr-un prefix distinct:

- GC – Săli de Curs;
- GL – Săli de Laborator;
- GS – Săli de Seminar;
- GP – Săli de Proiect.

Funcționalități:

- Introducerea numărului de săli pentru fiecare tip;
- Generarea automată a codurilor (ex: GC1, GL2);
- Vizualizarea sălilor generate, grupate pe categorii;

- Selecție multiplă sau individuală;
- Ștergerea sălilor selectate;
- Reîncărcarea listei din baza de date.

După ce sunt introduse toate sălile necesare, utilizatorul trebuie să apese „Salvează sălile”. Ulterior, poate continua către etapa de configurare a profesorilor.

Fig. 34 Interfața pentru gestionarea sălilor

### Pasul 5 – Gestionarea profesorilor

Această secțiune permite introducerea profesorilor disponibili pentru activitățile didactice: cursuri, seminarii și laboratoare. Este esențial ca pentru fiecare profesor să fie completate toate informațiile necesare pentru o planificare corectă a orarului (Figura 35).

Zile / Interval	08:00-10:00	10:00-12:00	12:00-14:00	14:00-16:00	16:00-18:00	18:00-20:00
Luni						
Marti						
Miercuri						
Joi						

Fig. 35 Gestionarea profesorilor





Ce completezi pentru fiecare profesor:

1. Nume complet – de exemplu: Dr. Andrei Popescu.
2. Discipline predate – adaugi o disciplină (ex: „Programare Java”) și selectezi:
  - Nivelul: Licență sau Master;
  - Tipul: Curs, Seminar sau Laborator;
  - Poți adăuga mai multe discipline pentru același profesor.
3. Disponibilitatea săptămânală:
  - Se afișează o grilă orară: zile (Luni–Vineri) și intervale (08:00–20:00);



- Dai click pe celulele din grilă pentru a marca orele în care profesorul este disponibil;
- Celulele activate se colorează, cele inactive rămân albe.

Funcționalități disponibile:

-  Reîncarcă – actualizează lista cu profesorii existenți din baza de date;
-  Salvează profesor – adaugă profesorul cu toate datele înregistrate;
-  Înapoi – revine la pasul anterior (Gestionare Săli);
-  Continuă – trece la pasul următor (ex. configurare reguli sau generare orar).

#### Pasul 6: Setarea regulilor de generare

În această secțiune, utilizatorul definește regulile care guvernează generarea orarului pentru toate grupele și subgrupele din învățământul de Licență și Master (Figura 36).

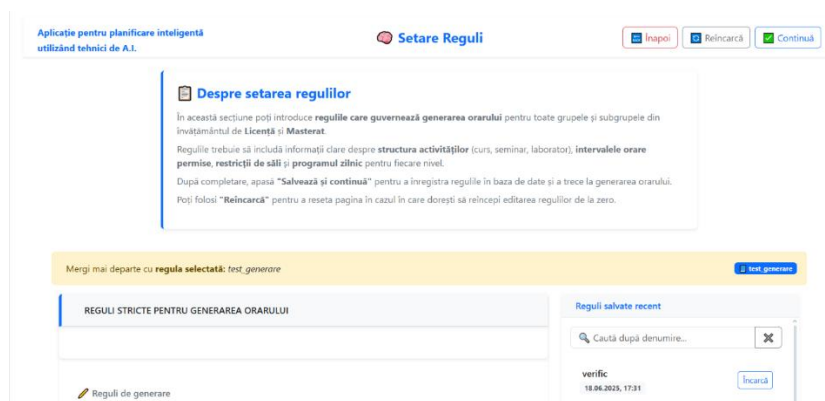


Fig. 36 Setarea regulilor de generare

Ce trebuie completat (Figura 37):

- Structura activităților: se stabilește cum sunt distribuite cursurile (pe an), seminarele și proiectele (pe grupă), laboratoarele (pe subgrupă).
- Intervale orare: se definește programul permis pentru fiecare zi.
  - Licență: 08:00 – 20:00;
  - Master: 16:00 – 20:00;
- Restricții speciale:
  - Miercuri 14:00–16:00 trebuie să fie liber pentru toți;
  - Sunt permise maximum două pauze de 2h/zi;
- Validitate orar: trebuie să existe activități pentru fiecare grupă/subgrupă în fiecare zi (cu excepția pauzei de miercuri).

Acțiuni disponibile:

- Denumeste regula în câmpul „Denumire regulă”.
- Poți salva ca regulă nouă, actualiza o regulă existentă sau șterge.
- Regula este salvată în baza de date și poate fi reutilizată.

Fig. 37 Editarea regulilor de generare

### Pasul 7: Generarea orarului

După introducerea tuturor datelor (profesori, săli, grupe și reguli), se poate trece la generarea efectivă a orarului (Figura 38).

Fig. 38 Generarea orarului

### Pași de execuție:

1. Se alege anul de studiu pentru care vrei să generezi orarul.
2. Se alege nivelul de studiu (Licență sau Master).
3. Apasă butonul verde „Generează orar cu AI” pentru o planificare automată folosind modelul inteligent.
4. Alternativ, poți folosi „Generează clasic” dacă vrei varianta algoritmică tradițională.

### Opțiuni suplimentare:

- Poți edita, încărca sau șterge orarele generate anterior.
- Se poate vizualiza istoricul complet al orarelor salvate pentru fiecare an.





## Pasul 8: Afișarea orarului generat și validarea sa

După apăsarea pe „Generează orar cu AI”, aplicația afișează automat orarul rezultat, împreună cu un raport complet de validare.

Detalii afișate:

### 1. Raport de validare a orarului (secțiune albastră) (Figura 39):

-  Acuratețea generării: procent estimat (ex: 100%);
-  Verificări sincronizare:
- Cursuri – verificate să nu se suprapună;
- Seminariile și proiectele să se desfășoare cu grupa;
- Laboratoare – distribuite corect pe subgrupe;
- Completitudine – toate grupele au activitățile aferente.

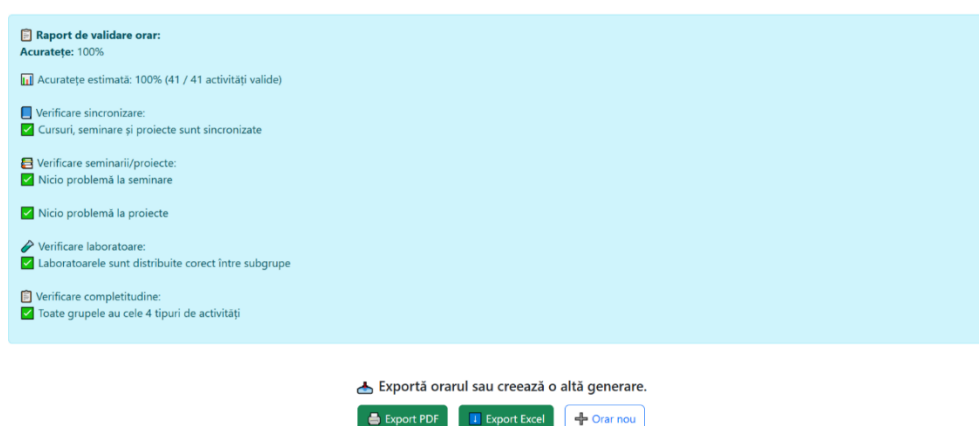


Fig. 39 Raport de validare a orarului




### 2. Orarul afișat vizual include (Figura 40):

- Structurat pe ani, grupe și subgrupe (ex: Master – MI1a, MI2a etc.);
- Pentru fiecare zi (Luni–Vineri) și fiecare interval orar (ex: 16:00–18:00);
- Denumirea completă a disciplinei + acronim (ex: *Baze de date (BD)*);
- Numele cadrului didactic (ex: *Dr. Roxana Iancu*);
- Sala în care are loc activitatea (ex: *GC6*).

<b>Master</b>					
<b>Master – MI1a</b>					
Interval	Luni	Marti	Miercuri	Joi	Vineri
16:00-18:00	<b>Algoritmi pentru date masive (apdm)</b> Dr. Roxana Iancu GC6	<b>Sisteme Distribuite (sd)</b> Lect. Radu Sima GC1	<b>Inteligentă Artificială (ia)</b> Prof. Maria Stan GC5	<b>Inteligentă limbajului natural (iln)</b> Conf. Ovidiu Stan GC7	<b>ILN</b> Conf. Ovidiu Stan GS2
18:00-20:00	<b>APDM</b> Dr. Roxana Iancu GS2	<b>SD</b> Lect. Radu Sima GP1	<b>IA</b> Prof. Maria Stan GP1	<b>IA</b> Prof. Maria Stan GL1	-
<b>Master – MI2a</b>					
Interval	Luni	Marti	Miercuri	Joi	Vineri
16:00-18:00	<b>Algoritmi pentru date masive (apdm)</b> Dr. Roxana Iancu GC6	<b>Sisteme Distribuite (sd)</b> Lect. Radu Sima GC1	<b>Inteligentă Artificială (ia)</b> Prof. Maria Stan GC5	<b>Inteligentă limbajului natural (iln)</b> Conf. Ovidiu Stan GC7	<b>IA</b> Prof. Maria Stan GL1
18:00-20:00	<b>SD</b> Lect. Radu Sima GP1	<b>APDM</b> Dr. Roxana Iancu GS2	<b>ILN</b> Conf. Ovidiu Stan GS2	-	-
<b>Master – MI1b</b>					
Interval	Luni	Marti	Miercuri	Joi	Vineri

Fig. 40 Orarul afișat vizual

## 3. Opțiuni export orar:

-  Export PDF;
-  Export Excel;
-  Buton pentru generare nouă.

## Exportul orarului generat (Excel și PDF)

După generarea cu succes a orarului și validarea acestuia, aplicația permite exportul orarului în două formate:

- Export PDF – generează un fișier cu orarul afișat într-un format printabil, tabelar și colorat, pe grupe (Figura 41).

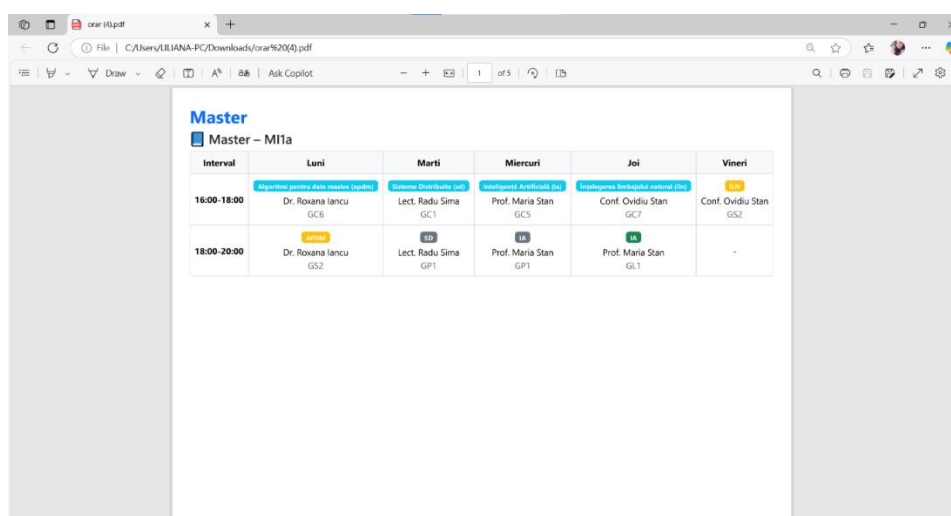


Fig. 41 Exportul orarului generat în format PDF

- Export Excel – generează un fișier tabelar cu toate activitățile detaliate: ziua, intervalul, disciplina, tipul, profesorul și sala (Figura 42).

	Nivel	Grupa	Zi	Interval	Disciplina	Tip	Profesor	Sala
1	Master	MI1a	Luni	16:00-18:00	Algoritmi pentru date masive (apdm)	Curs	Dr. Roxana Iancu	GC6
2	Master	MI1a	Luni	18:00-20:00	APDM	Seminar	Dr. Roxana Iancu	GS2
3	Master	MI1a	Marti	16:00-18:00	Sisteme Distribuite (sd)	Curs	Lect. Radu Sima	GC1
4	Master	MI1a	Marti	18:00-20:00	SD	Proiect	Lect. Radu Sima	GP1
5	Master	MI1a	Miercuri	16:00-18:00	Inteligență Artificială (ia)	Curs	Prof. Maria Stan	GC5
6	Master	MI1a	Miercuri	18:00-20:00	IA	Proiect	Prof. Maria Stan	GP1
7	Master	MI1a	Joi	16:00-18:00	Înțelegerea limbajului natural (lin)	Curs	Conf. Ovidiu Stan	GC7
8	Master	MI1a	Joi	18:00-20:00	IA	Laborator	Prof. Maria Stan	GL1
9	Master	MI1a	Vineri	16:00-18:00	ILN	Seminar	Conf. Ovidiu Stan	GS2

Fig. 42 Exportul orarului generat în format Excel

După generarea și validarea cu succes a orarului, utilizatorul are posibilitatea de a:

- reveni în oricare dintre secțiunile aplicației (Grupe, Săli, Profesori, Reguli) pentru a modifica sau actualiza datele introduse;
- salva orarele generate pentru consultare ulterioară;
- genera orare noi folosind aceleași reguli sau un set diferit de reguli;
- exporta orarul în formate profesionale, adaptate pentru printare sau distribuire digitală.

Această flexibilitate permite utilizatorului să testeze mai multe scenarii, să îmbunătățească planificarea academică și să obțină rapid o versiune optimă a orarului, adaptată cerințelor instituției.

Aplicația este concepută pentru a fi intuitivă, modulară și complet reutilizabilă, oferind un proces eficient și inteligent de creare a orarelor .

Recomandări finale pentru utilizatori:

- Asigură-te că toate datele introduse (profesori, săli, grupe, reguli) sunt corecte și complete înainte de generarea orarului;
- Pentru rezultate optime, stabilește reguli clare și realiste, ținând cont de disponibilitatea resurselor;
- Folosește opțiunea de salvare a regulilor pentru a putea relansa rapid o generare similară în viitor;
- După generare, verifică raportul de validare și efectuează eventualele ajustări necesare înainte de exportul final;
- Testează mai multe variante de orar dacă ai scenarii alternative (ex. distribuții diferite ale disciplinelor sau profesori noi).

#### 4.4. Probleme întâmpinate în timpul implementării

Pe parcursul dezvoltării aplicației, au fost identificate și soluționate mai multe dificultăți tehnice și funcționale, specifice unui sistem complex care integrează baze de date, interfață web, inteligență artificială și exporturi dinamice. Dintre cele mai relevante probleme întâmpinate, menționăm:

1. Structurarea inițială a datelor: Definirea unei scheme de bază de date care să reflecte corect relațiile dintre profesori, discipline, săli, grupe și activități s-a dovedit complexă. A fost necesară normalizarea tabelor și introducerea de tabele intermediare pentru a gestiona cazurile în care un profesor predă mai multe discipline, la diferite niveluri și tipuri de activități.
2. Sincronizarea activităților pe grupe și subgrupe: În procesul de generare a orarului, a fost dificilă implementarea regulilor conform cărora:
  - cursurile trebuie să fie comune întregului an;
  - seminarele/proiectele să fie comune la nivel de grupă;
  - laboratoarele să fie individuale pe subgrupă.

Inițial, aceste activități apăreau dublate sau erau plasate în același interval pentru mai multe subgrupe, necesitând reguli suplimentare de validare.

3. Limitările modelului AI în generarea JSON valid: Utilizarea GPT-4 pentru generarea orarului a dus uneori la rezultate invalide JSON, cu erori de sintaxă sau formate incorecte. A fost necesară adăugarea unei funcții de validare automată și corectare a structurii JSON înainte de afișare. În plus, AI-ul nu a reușit inițial să genereze simultan orare complete și corecte pentru toate grupele și anii de studiu, omițând adesea ani sau zile întregi, ceea ce a impus completări manuale sau reformulări ale promptului. Am testat iterativ numeroase versiuni ale promptului, în colaborare cu AI-ul, pentru a respecta toate constrângerile academice.
4. Inconsistența între datele introduse și cele folosite la generare: Inițial, în orarul generat apăreau discipline, săli sau profesori care nu fuseseră introduși de utilizator. Această problemă a fost cauzată de faptul că AI-ul „inventase” date în lipsa unor instrucțiuni clare. S-a rezolvat prin actualizarea promptului GPT pentru a specifica explicit: „folosește doar datele existente în baza de date”.
5. Încărcarea prea multor reguli în promptul AI: Atunci când s-au introdus foarte multe reguli detaliate, promptul pentru GPT devenea prea lung, ceea ce ducea la erori de API (ex. `context_length_exceeded`) sau răspunsuri tăiate. Soluția a fost comprimarea promptului prin reformulare, gruparea regulilor în categorii și eliminarea redundanțelor.
6. Atribuirea sălilor fără suprapuneri: Gestionarea automată a sălilor (pe categorii: curs, seminar, laborator) a fost provocatoare, deoarece inițial se puteau asigna aceeași sală în același interval orar pentru mai multe activități. A fost implementat un mecanism de verificare a disponibilității sălilor în timpul generării.
7. Timpul de răspuns al generatorului AI: În unele cazuri, modelul GPT-4 avea un timp de răspuns prea mare sau returna răspunsuri incomplete.
8. Exportul PDF/Excel și păstrarea formătărilor: Exportul orarului în PDF și Excel a necesitat ajustări multiple pentru a păstra formatul tabelar, culorile activităților și lizibilitatea pe toate dimensiunile de ecran. Bibliotecile `html2pdf.js` și `SheetJS` au fost configurate cu stiluri CSS dedicate.
9. Modularizarea și întreținerea codului React: Pe măsură ce aplicația a crescut în complexitate, codul componentelor React devenise greu de întreținut. S-a realizat o refactorizare generală, prin separarea logicii în hook-uri dedicate (`useProfesoriLogic.js`, `useSaliLogic.js` etc.), pentru a îmbunătăți claritatea, reutilizarea și testabilitatea aplicației.

Depășirea acestor provocări a contribuit semnificativ la consolidarea aplicației, care a evoluat într-o soluție robustă, scalabilă și adaptabilă nevoilor reale din mediul academic. Totodată, integrarea inteligenței artificiale a oferit lecții valoroase despre limitele și potențialul acestora în contexte educaționale complexe.

#### 4.5. Comparația cu metoda clasică de generare a orarelor

Pentru a evidenția eficiența și avantajele aduse de utilizarea inteligenței artificiale în generarea orarului, a fost realizată o comparație detaliată între două abordări fundamentale: algoritmul clasic (bazat pe reguli codificate manual) și algoritmul AI (bazat pe generare prin model GPT, local sau OpenAI). Această comparație se bazează atât pe aspecte tehnice, cât și pe perspective de utilizare și mentenanță (Tabelul 2).

Tabel 2 Comparația cu metoda clasică

<i>Criteriu de comparare</i>	<i>Algoritm Clasic (Hardcoded)</i>	<i>Algoritm AI (OpenAI GPT / model local)</i>
<i>Flexibilitate</i>	<i>Limitată – reguli hardcoded, modificările necesită rescrierea codului</i>	<i>Foarte ridicată – se adaptează pe baza unui prompt cu reguli în format JSON</i>
<i>Personalizare</i>	<i>Doar prin modificare în cod (ex: ore master, pauze, max cursuri)</i>	<i>Personalizabil din interfață (formular HTML sau reguli JSON)</i>
<i>Timp de dezvoltare</i>	<i>Ridicat – necesită tratarea manuală a fiecărei reguli</i>	<i>Redus – promptul AI gestionează logică complexă cu mai puține linii de cod</i>
<i>Acoperire logică</i>	<i>Hardcoded pe structură fixă: cursuri → an, seminare → grupă, lab/proiect → subgrupă</i>	<i>Se respectă automat structura: an, grupă, subgrupă + se pot adăuga reguli noi</i>
<i>Validare orar</i>	<i>Manuală, prin metode Python (valideaza_cursuri_sincronizate)</i>	<i>Automatizată cu funcții de validare + feedback textual și vizual în browser</i>
<i>Control asupra profesorilor/sălilor</i>	<i>Selectați aleator + validați în cod</i>	<i>Selectați în mod inteligent din baza de date, pe baza disponibilității</i>
<i>Reutilizare</i>	<i>Limitată – algoritmul e specific unei structuri (licență/master)</i>	<i>Generalizabil – același motor poate genera orare pentru orice structură</i>
<i>Rezultate</i>	<i>Coerente, dar pot apărea suprapuneri dacă logica nu acoperă toate cazurile</i>	<i>În general coerente și complete, cu auto-corectare și ajustare din prompt</i>
<i>Interactivitate</i>	<i>Lipsită – utilizatorul final nu poate modifica reguli fără programare</i>	<i>Ridicăta – utilizatorul poate modifica regulile în interfață</i>
<i>Timp de generare</i>	<i>Sub 1–2 secunde (execuție cod local)</i>	<i>5–15 secunde (în funcție de complexitatea promptului și modelul folosit)</i>
<i>Calitatea orarului generat</i>	<i>Variabilă – depinde strict de acoperirea logicii în cod; riscul de suprapuneri este mai mare</i>	<i>În general ridicată – reguli respectate dacă promptul este bine formulat</i>

<i>Complexitate implementare</i>	<i>Mare – necesită tratamente individuale pentru fiecare excepție și regulă</i>	<i>Redusă – implementarea AI e centrată pe redactarea promptului și validarea rezultatului</i>
<i>Costuri</i>	<i>Doar timp de dezvoltare (fără costuri financiare directe)</i>	<i>Pot apărea costuri pentru API OpenAI, dar pot fi evitate folosind modele locale</i>

Metoda clasică, deși rapidă în execuție și complet locală, se dovedește rigidă și dificil de adaptat la noi cerințe. Orice modificare a regulilor impune rescrierea logicii în cod, ceea ce implică timp suplimentar și un risc crescut de erori. În plus, calitatea orarului depinde exclusiv de capacitatea programatorului de a prevedea toate cazurile posibile.

În schimb, metoda bazată pe AI permite o abordare mult mai flexibilă și accesibilă, permițând oricărui utilizator – chiar fără cunoștințe tehnice – să modifice regulile după necesități. Deși timpul de generare este ușor mai mare și pot exista costuri asociate utilizării modelelor externe, avantajele legate de calitatea orarului, respectarea constrângerilor complexe și interactivitate sunt semnificative.

Această abordare permite scalarea aplicației pentru multiple instituții, programe și structuri curriculare, fără a fi necesară rescrierea codului pentru fiecare caz nou, ci doar ajustarea regulilor în interfață. În plus, integrarea funcțiilor de validare automată și feedback vizual aduce un plus de încredere și control asupra procesului.

## CAPITOLUL 5. TESTAREA ȘI VALIDAREA APLICAȚIEI

Testarea reprezintă o componentă fundamentală în procesul de dezvoltare software, având rolul de a asigura calitatea, stabilitatea și funcționarea corectă a aplicației. Indiferent de cât de bine este proiectat sau dezvoltat un sistem, testarea este esențială pentru a detecta erorile, pentru a valida comportamentul aplicației în diverse scenarii și pentru a garanta o experiență coerentă și sigură pentru utilizatorii finali.

În general, testarea se împarte în două mari categorii:

- Testarea manuală, în care un tester uman verifică funcționalitățile aplicației, urmând scenarii prestabilite sau explorând liber comportamentul sistemului;
- Testarea automată, în care teste predefinite sunt executate automat prin intermediul unor framework-uri specializate (ex: JUnit, PyTest, Jasmine, PHPUnit), permițând validarea repetitivă, rapidă și fiabilă a componentelor aplicației.

În cadrul acestei lucrări, testarea a vizat atât partea de frontend (interfața utilizatorului), cât și backend-ul (API-ul și logica de generare a orarului), pentru a asigura buna funcționare a aplicației de la introducerea datelor până la exportul orarului final.

### 5.1 Testarea manuală

Testarea manuală a fost utilizată pentru a verifica funcționarea corectă a aplicației din perspectiva utilizatorului final. Aceasta a constat în parcurgerea interfeței grafice, introducerea datelor în diverse componente și observarea comportamentelor aplicației în diferite scenarii de utilizare. Au fost vizate componente esențiale precum autentificarea, gestionarea entităților (profesori, săli, grupe), generarea orarului și funcțiile de export și salvare.

Testele au fost realizate în mod sistematic, urmărind pași concreți și evaluând răspunsurile aplicației pentru fiecare caz.

#### Testul 1 – Autentificare

##### Scenariul 1 – Date corecte

- Pasul 1: Se deschide pagina de autentificare.
- Pasul 2: Se introduce admin / parola123.
- Pasul 3: Se apasă pe „Autentificare”.
- Rezultat: Se accesează pagina principală (home).

##### Scenariul 2 – Parolă greșită

- Pasul 1: Se introduce admin / gresit.
- Rezultat: Se afișează mesajul: „Date de autentificare incorecte”.

##### Scenariul 3 – Câmpuri goale

- Pasul 1: Se lasă necompletat user/parolă.
- Rezultat: Se afișează mesajul: „Toate câmpurile sunt obligatorii!”.

##### Scenariul 4 – Acces direct la pagină securizată

- Pasul 1: Se încearcă accesarea /dashboard fără autentificare.
- Rezultat: Redirecționare către login.

## Testul 2 – Gestionarea profesorilor

### Scenariul 1 – Adăugare validă

- Pasul 1: Se completează toate câmpurile: nume, discipline, tipuri, nivel.
- Rezultat: Profesorul apare în listă, cu mesaj de confirmare.

### Scenariul 2 – Lipsă nume

- Pasul 1: Se lasă câmpul „Nume” gol.
- Rezultat: Mesaj de eroare vizibil: „Atenție! Te rugăm să completezi toate câmpurile obligatorii.”

### Scenariul 3 – Mai multe discipline

- Pasul 1: Se adaugă „Programare, Baze de date”.
- Rezultat: Disciplinele apar separate și corect afișate.

## Testul 3 – Generarea orarului

### Scenariul 1 – Generare completă

- Pasul 1: Se introduc profesori, săli, grupe, reguli.
- Pasul 2: Se accesează pagina de generare.
- Pasul 3: Se apasă „Generează orar cu AI”.
- Rezultat: Orarul este generat pentru toate grupele și zilele.

### Scenariul 2 – Lipsă săli

- Pasul 1: Nu se introduc săli.
- Rezultat: Butonul de generare nu este disponibil.

### Scenariul 3 – Lipsă regulă selectată

- Pasul 1: Se accesează pagina fără selectarea unei reguli.
- Rezultat: Apare Swal cu mesajul: „Regulă neselectată”.

## Testul 4 – Exportul orarului

### Scenariul 1 – Export PDF

- Pasul 1: Se generează un orar complet.
- Pasul 2: Se apasă pe „Exportă PDF”.
- Rezultat: Se descarcă fișierul orar.pdf.

### Scenariul 2 – Export Excel

- Pasul 1: Se apasă „Exportă Excel”.
- Rezultat: Se descarcă orar.xlsx.

### Scenariul 3 – Fără orar generat

- Pasul 1: Se accesează pagina fără a genera orar.
- Rezultat: Butonul de export nu apare deloc.

## Testul 5 – Salvarea și încărcarea orarelor anterioare

### Scenariul 1 – Salvare implicită

- Pasul 1: Se generează un orar.
- Rezultat: Orarul apare în lista „Orare salvate anterior”.



**Scenariul 2 – Încărcare orar existent**

- Pasul 1: Se apasă „Încarcă” pe un orar din listă.
- Rezultat: Orarul se afișează în aplicație.

**Scenariul 3 – Editare orar**

- Pasul 1: Se apasă „Editează” pe un orar salvat.
- Rezultat: Datele pot fi modificate și salvate din nou.

**Scenariul 4 – Ștergere orar**

- Pasul 1: Se apasă „Șterge” și se confirmă.
- Rezultat: Orarul este eliminat din listă.

**Scenariul 5 – Căutare orar**

- Pasul 1: Se caută după nume.
- Rezultat: Lista este filtrată instant.

**Testarea automată**

Testarea automată reprezintă un proces esențial în dezvoltarea aplicațiilor software moderne, asigurând validarea funcționalităților sistemului fără intervenție umană. În cadrul acestui proiect, testarea automată a fost utilizată pentru a verifica corectitudinea componentelor backend și frontend ale aplicației web pentru generarea orarelor, contribuind astfel la creșterea fiabilității și mentenanței codului.

Scopul principal al testării automate este de a detecta rapid erorile și regresiile, facilitând dezvoltarea iterativă și introducerea de noi funcționalități fără riscul deteriorării celor existente. În plus, testele automate oferă un grad ridicat de încredere în comportamentul aplicației în scenarii diverse.

Pentru acest proiect, testele automate au fost structurate în două mari categorii: testarea backendului și testarea frontendului.

Testarea automată a permis dezvoltarea controlată a aplicației și a contribuit la menținerea calității codului pe parcursul întregului ciclu de dezvoltare.

**5.2 Testarea automată a backendului**

Testarea backendului a fost realizată cu ajutorul frameworkului pytest, o bibliotecă Python eficientă și extensibilă, folosită pentru a valida corectitudinea logicii aplicației. Testele au fost organizate pe fișiere separate, în funcție de funcționalitățile testate, fiecare acoperind o zonă esențială a aplicației: rutare și API (endpoints), interacțiuni cu baza de date, reguli de generare și validare a orarului.

**5.2.1 Testarea funcționalității de generare orar folosind AI**

Testul `test_generare_orar_ai` validează funcționarea endpointului `/genereaza_orar`, responsabil de generarea orarului cu ajutorul modelului de inteligență artificială GPT-4o (OpenAI) (Figura 43). Testul transmite un payload JSON cu un prompt specific și parametri de configurare (nivel, an, grupe), apoi verifică dacă răspunsul este un JSON

valid, structurat corect pe nivel, grupă, zi și interval orar. De asemenea, testul confirmă că fiecare activitate conține toate câmpurile esențiale: activitate, tip, profesor și sală.

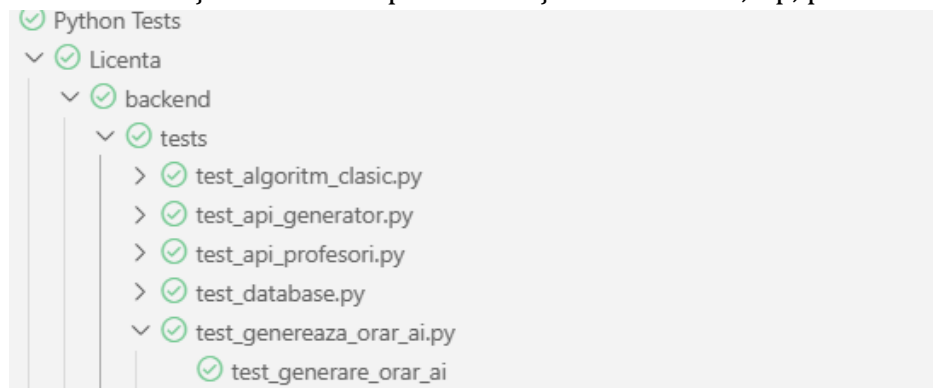


Fig. 43 Testarea funcționalității de generare orar folosind Ai

### 5.2.2 Testarea generatorului propriu de orar

Fișierul `test_orar_generator.py` conține teste pentru clasa `OrarGenerator`, care implementează logica de generare personalizată a orarului (Figura 44). Au fost utilizate tehnici de mocking (cu `unittest.mock`) pentru a izola logica de baza de date.

Testele implementate acoperă următoarele aspecte esențiale:

- Inițializarea corectă a generatorului cu reguli implicite (pauză miercuri, max. 8h/zi).
- Extragerea nivelului și anului din denumirea grupei (ex: LM2a → Master, anul II).
- Actualizarea criteriilor (ore pe zi, pauze, reguli).
- Generarea efectivă a orarului conform regulilor (inclusiv pauza de miercuri).
- Limitarea numărului de cursuri diferite (ex: max. 9).

Toate testele au trecut, confirmând că logica de generare funcționează conform cerințelor și poate fi configurată flexibil.

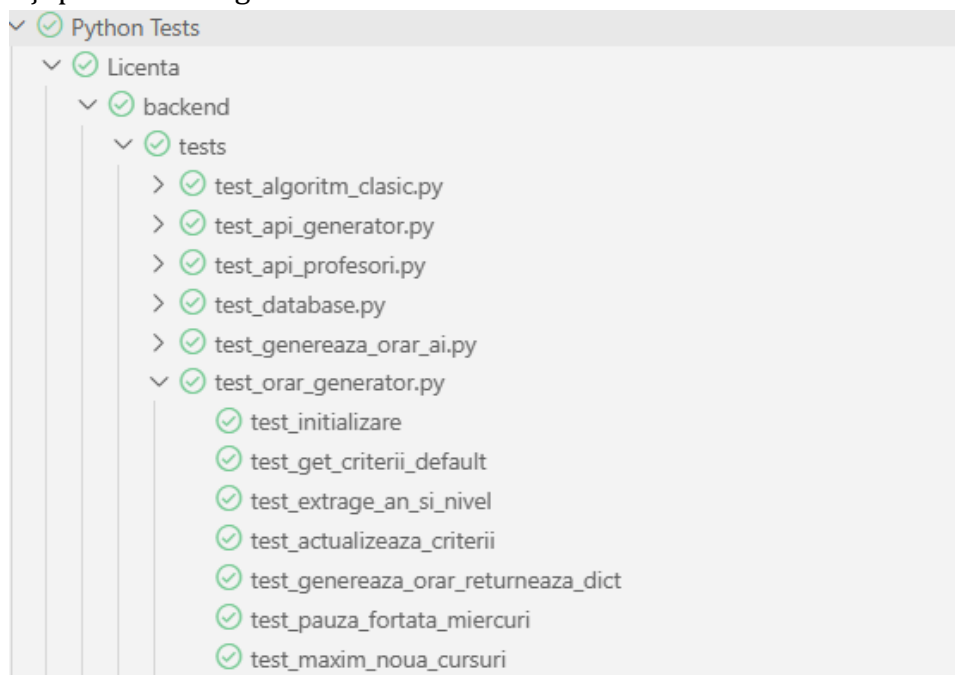


Fig. 44 Testarea generatorului propriu de orar

### 5.2.3 Testarea validării orarului generat

Fișierul `test_validator_orar.py` conține teste pentru clasa `ValidatorOrar`, care verifică dacă cursurile comune sunt sincronizate corect între grupele unui an (Figura 45).

Testele acoperă:

- Sincronizare validă – același curs apare în același interval, zi și sală pentru toate grupele (ex: LI1a și LI1b).
- Sincronizare invalidă – același curs apare în zile diferite la grupe diferite, iar validarea semnalează eroarea.

Ambele teste au fost rulate cu succes, demonstrând că sistemul detectează atât cazurile corecte, cât și erorile de aliniere.

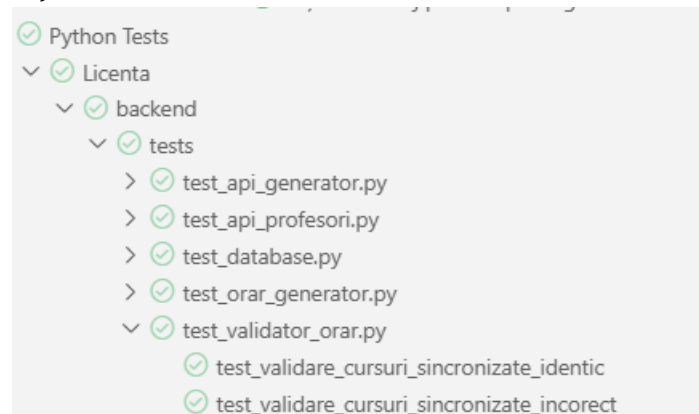


Fig. 45 Testarea validării orarului generat

### 5.2.4 Testarea componentelor auxiliare

Testele ilustrate în imaginile de mai sus acoperă componentele esențiale ale aplicației backend. Au fost verificate operațiile CRUD pentru profesori și reguli, interacțiunile cu baza de date, corecta funcționare a endpointurilor API și inițializarea componentelor auxiliare (Figura 46).



Fig. 46 Testarea componentelor auxiliare

De asemenea, testele validează structura datelor generate, setările implicite și respectarea constrângerilor funcționale privind generarea orarului – fără a include logica bazată pe AI sau pe algoritmul propriu, care sunt testate separat.

Toate testele au fost scrise în `pytest` și executate cu succes, confirmând stabilitatea funcțională a aplicației.

### 5.3 Testarea automată a frontendului

Testarea automată a interfeței aplicației a fost realizată cu ajutorul bibliotecilor Jest și React Testing Library, urmărind validarea componentelor principale din aplicația React. Fiecare componentă esențială (precum pagina Home, formularul pentru profesori, setarea regulilor și generarea orarului) a fost testată separat pentru a verifica afișarea corectă a elementelor, comportamentul la interacțiuni și reacția la datele introduse. Aceste teste contribuie la menținerea stabilității aplicației și asigură o experiență consecventă pentru utilizator, chiar și după actualizări ale codului.

#### 5.3.1 Testarea componentei Home.jsx

Componenta Home.jsx este pagina principală și afișează conținut diferit în funcție de starea de autentificare a utilizatorului. Testele au fost realizate cu Jest și React Testing Library, folosind mock pentru logica de autentificare (useHomeLogic) (Figura 47).

Testele acoperă:

- Utilizator neautentificat: titlul aplicației, butonul „Autentifică-te”, mesaj informativ, buton de generare dezactivat.
- Utilizator autentificat: mesaj de bun venit, butoanele „Logout” și „Orarul meu”, butonul de generare activ și mesajul de redirecționare.

Toate testele au fost executate cu succes, confirmând afișarea corectă a interfeței în ambele scenarii.

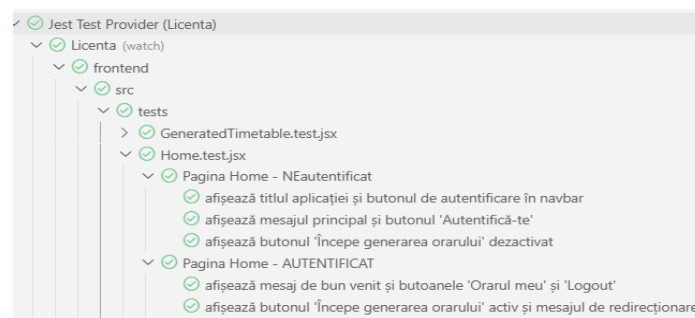


Fig. 47 Testarea componentei Home.jsx

#### 5.3.2 Testarea componentei GeneratedTimetable.jsx

Componenta GeneratedTimetable gestionează generarea, afișarea și validarea orarelor, fiind una dintre cele mai complexe din aplicație (Figura 48). Testarea a fost realizată cu Jest și React Testing Library, utilizând mock-uri pentru hook-urile implicate. S-a verificat funcționarea corectă a butoanelor principale și auxiliare, afișarea și filtrarea orarelor salvate, precum și apelul corect al funcțiilor de generare. Toate testele au fost executate cu succes, confirmând stabilitatea și funcționalitatea completă a componentei.

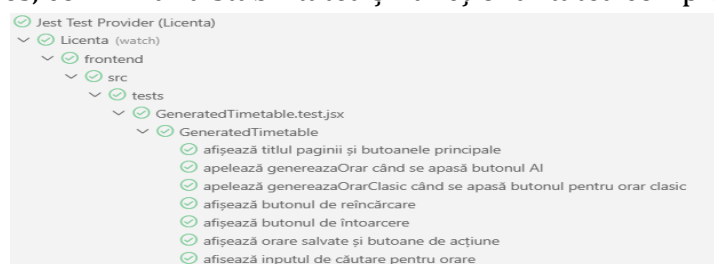


Fig. 48 Testarea componentei GeneratedTimetable.jsx

### 5.3.3 Testarea componentei SetareReguli.jsx

Componenta SetareReguli permite utilizatorului să definească, salveze și reutilizeze reguli personalizate în format JSON pentru generarea orarului (Figura 49). Testarea s-a realizat cu Jest și React Testing Library, folosind mock pentru useSetariReguli.

Testele acoperă:

- Afișarea titlului și descrierii componentei.
- Prezența câmpului pentru denumirea regulii (cu placeholder sugestiv).
- Funcționarea butonului „Salvează” (apelează funcția salveazaReguli).
- Afișarea mesajului „Nu există reguli salvate” când lista este goală.

Toate testele au fost executate cu succes, validând afișarea și interactivitatea componentei în toate scenariile relevante.

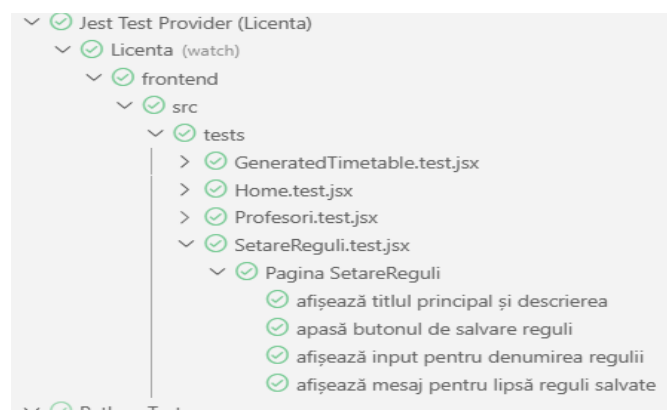


Fig. 49 Testarea componentei SetareReguli.jsx

În urma procesului riguros de testare, atât manuală, cât și automată, s-a confirmat faptul că aplicația funcționează în mod stabil, coerent și conform cerințelor funcționale definite. Testarea manuală a evidențiat comportamentul aplicației din perspectiva utilizatorului final, validând corecta desfășurare a operațiunilor esențiale precum autentificarea, gestionarea datelor, generarea orarului și exportul acestuia. În paralel, testarea automată a asigurat verificarea logicii interne, a interacțiunilor cu baza de date și a componentelor critice de backend și frontend, contribuind la menținerea calității codului în contextul unor dezvoltări iterative.

Prin urmare, testarea a demonstrat că aplicația este nu doar funcțională, ci și fiabilă, oferind o experiență de utilizare intuitivă și predictibilă. Această etapă a avut un rol esențial în consolidarea încrederii în produsul final și în susținerea livrării unui sistem capabil să gestioneze cu acuratețe și eficiență procesul complex de generare a orarelor.

## CONCLUZII

Lucrarea de față a avut ca obiectiv dezvoltarea unei aplicații web moderne pentru generarea automată a orarelor, utilizând tehnici de inteligență artificială și o arhitectură modulară bazată pe tehnologii actuale (React, Flask, MySQL, OpenAI API). Rezultatele obținute confirmă fezabilitatea și eficiența unei astfel de soluții în contextul digitalizării proceselor academice.

Contribuția originală a lucrării constă în:

- integrarea unui model de inteligență artificială (GPT-4) pentru generarea orarului pe baza unor reguli prestabilite și a datelor reale din baza de date;
- posibilitatea comparării rezultatelor generate de AI cu cele obținute printr-un algoritm clasic propriu;
- implementarea unei interfețe moderne și intuitive, care include validări, export în PDF/Excel și funcționalități de filtrare, salvare și editare a orarelor;
- respectarea unor constrângeri reale și complexe, precum distribuția activităților pe ani, grupe și subgrupe, precum și disponibilitatea profesorilor și a sălilor.

Elementul de noutate al proiectului îl constituie utilizarea unui model AI generativ capabil să transforme automat reguli exprimate în limbaj natural într-o structură coerentă de orar, completă pentru toate grupele și subgrupele, cu verificări de consistență și sincronizare integrate.

Aplicația poate fi utilizată cu succes în mediul academic pentru a reduce semnificativ timpul și efortul necesare elaborării unui orar coerent. De asemenea, poate fi extinsă în alte contexte în care este necesară planificarea automată a resurselor, în funcție de constrângeri multiple.

Aspecte pozitive identificate:

- Generare automată completă a orarului.
- Interfață prietenoasă și funcționalități extinse.
- Posibilitatea de validare și editare ulterioară a orarelor generate.

Limitări constatate:

- Răspunsurile modelului AI pot varia în funcție de formularea promptului, necesitând ajustări suplimentare.
- Integrarea unui model local sau open-source performant este în curs de explorare și optimizare.
- Procesul de testare și validare a presupus un efort suplimentar, din cauza complexității regulilor aplicate.

Direcții viitoare de dezvoltare:

- Implementarea unui modul de gestionare a profilului utilizatorului (administrator, profesor, student), cu funcționalități personalizate;
- Crearea unor interfețe dedicate pentru profesori și studenți, care să permită vizualizarea individuală a orarelor;
- Integrarea și testarea altor modele AI gratuite sau locale, pentru a crește autonomia față de serviciile comerciale;

- Dezvoltarea unui mod de generare offline, bazat pe un model propriu sau pe un algoritm determinist;
- Adăugarea unui sistem de notificări automate (e-mail sau push) la actualizarea orarului;
- Extinderea aplicației pentru programarea examenelor sau a altor activități extracurriculare;
- Realizarea unei versiuni mobile, compatibile cu Android și iOS, pentru acces facil și portabil.

În concluzie, proiectul realizat aduce o contribuție aplicabilă și valoroasă în domeniul planificării educaționale, demonstrând că inteligența artificială poate reprezenta un instrument eficient, sigur și scalabil pentru automatizarea proceselor administrative esențiale. Lucrarea deschide perspective reale pentru cercetări și dezvoltări viitoare, în direcția unei educații complet digitalizate și inteligent susținute tehnologic.

## BIBLIOGRAFIE

- [1] <https://www.unitime.org>, 18-06-2025.
- [2] <https://www.asctimetables.com>, 18-06-2025.
- [3] <https://generator-orare.ro/presentation> , 18-06-2025.
- [4] <https://platform.openai.com/docs/models/gpt-4.1>, 18-06-2025.
- [5] <https://generator-orare.ro/login>, 18-06-2025.
- [6] <https://support.microsoft.com/ro-ro/topic/crearea-unei-diagrame-de-clasă-uml-de6be927-8a7b-4a79-ae63-90da8f1a8a6b> , 18-06-2025.
- [7] <http://easy-learning.neuro.pub.ro:8888/Laboratoare/ Mase/L08 uml/UML.htm>, 18-06-2025.
- [8] <https://support.microsoft.com/ro-ro/topic/crearea-unei-diagrame-de-clasă-uml-de6be927-8a7b-4a79-ae63-90da8f1a8a6b>, 20-06-2025.
- [9] <https://support.microsoft.com/ro-ro/topic/crearea-unei-diagrame-de-componente-uml-aa924ecb-e4d2-4172-976e-a78fa157b074> , 18-06-2025.
- [10] [https://www.researchgate.net/publication/350801633\\_Diagrame\\_UML\\_pentru\\_modelarea\\_comportamentului\\_claselor\\_a\\_interactiunii\\_dintre\\_clase\\_si\\_a\\_implementarii\\_acestora\\_in\\_cazul\\_unui\\_sistem\\_de\\_gestiune\\_prin\\_metoda\\_ABC\\_a\\_costurilor\\_unei\\_organizatii](https://www.researchgate.net/publication/350801633_Diagrame_UML_pentru_modelarea_comportamentului_claselor_a_interactiunii_dintre_clase_si_a_implementarii_acestora_in_cazul_unui_sistem_de_gestiune_prin_metoda_ABC_a_costurilor_unei_organizatii), 18-06-2025.
- [11] <https://devdocs.io/javascript/>, 20-06-2025.
- [12] <https://docs.python.org/3/>, 20-06-2025.
- [13] <https://www.phpmyadmin.net/docs/>, 20-06-2025.
- [14] <https://devdocs.io/html/>, 20-06-2025.
- [15] <https://devdocs.io/css/>, 20-06-2025.
- [16] <https://legacy.reactjs.org>, 20-06-2025.
- [17] <https://flask.palletsprojects.com/en/stable/>, 20-06-2025.
- [18] <https://platform.openai.com/docs>, 20-06-2025.