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

PROIECT DE DIPLOMĂ

Îndrumător proiect/Coordonator ştiinţific,
Ş.l.dr.ing. Mihaela ŢIPLEA

Absolvent,
Liliana- Petruța ZANFIR

Galaţi
2025

PROGRAMUL DE STUDII: Calculatoare şi Tehnlogia Informaţiei

APLICAȚIE PENTRU PLANIFICARE INTELIGENTĂ UTILIZÂND TEHNICI DE AI

Coordonator ştiinţific,
Ş.l.dr.ing. Mihaela ŢIPLEA

Absolvent,
Liliana- Petruţa ZANFIR

Galaţi
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 universitare, 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

LISTA FIGURILOR

LISTA TABELELOR

INTRODUCERE

În prezent, organizarea eficientă a activităţilor educaţionale reprezintă o provocare majoră pentru instituţiile de învăţământ superior. Numărul mare de discipline, grupe, subgrupe, săli şi profesori implică un proces de planificare complex, consumator de timp şi predispus la erori umane. Generarea manuală a orarului universitar nu doar că presupune eforturi semnificative, dar adesea nu reuşeşte să răspundă cerinţelor reale ale studenţilor şi cadrelor didactice.

Pe măsură ce tehnologia s-a dezvoltat de-a lungul timpului, nevoia de o soluţie inteligentă şi automatizată pentru a îndeplini această funcţie a crescut odată cu avansarea inteligenţei artificiale.

Astfel, proiectarea şi dezvoltarea acestui tip de instrument, capabil să producă automat programe complete, validate şi personalizate, devine obligatorie pentru instituţiile contemporane, dar şi o oportunitate de eficientizare şi digitalizare a procesului academic.

Platforma dezvoltată oferă o soluţie completă de generare automată a orarului, adaptată structurii academice din România. Aceasta ţine cont de tipul activităţii (curs, seminar, laborator, proiect), de distribuţia pe ani, grupe şi subgrupe, de disponibilitatea profesorilor, generând orare corecte şi coerente pentru întreaga structură universitară. Platforma este gândită să fie uşor de utilizat, cu un timp de acomodare redus, oferind utilizatorului o experienţă clară şi fluentă.

Pentru a răspunde nevoilor diverse ale utilizatorilor, aplicaţia integrează două metode de generare: una bazată pe inteligenţă artificială (GPT-4 – OpenAI), care construieşte orarul pe baza unor reguli scrise în limbaj natural, şi o metodă clasică, algoritmică, ce respectă reguli stricte impuse prin logica implementată.

Funcţionalităţile propuse pentru realizarea acestei aplicaţii includ următoarele:

- Generarea automată a orarului pentru toate grupele şi subgrupurile unei facultăţi, în funcţie de datele introduse şi regulile definite;
- Integrarea a două metode de generare: una bazată pe inteligenţă artificială (GPT-4), care prelucrează reguli în limbaj natural, şi una clasică, deterministă, care aplică algoritmi proprii pentru plasarea activităţilor;
- Configurarea regulilor personalizate: intervale orare pentru licenţă/master, restricţii privind pauzele, tipuri de activităţi per structură (cursuri pentru ani, seminare şi proiecte pe grupe, laboratoare pe subgrupe);
- Gestionarea datelor academice: introducerea şi modificarea profesorilor, disciplinelor, sălilor (curs, seminar, laborator, proiect) şi a structurii de grupe/subgrupe;
- Validarea automată a orarului generat, prin detectarea suprapunerilor de profesori, săli sau activităţi şi verificarea respectării tuturor regulilor impuse;
- Exportul orarului în format PDF şi Excel, pentru utilizare oficială şi distribuire rapidă;
- Interfaţă grafică modernă şi receptivă, care permite navigarea uşoară şi clară între componentele aplicaţiei;

- Posibilitatea de extindere cu module suplimentare în viitor, cum ar fi planificarea examenelor sau a activităţilor extracurriculare.

Prin urmare, această aplicaţie aduce o contribuţie semnificativă la modernizarea procesului educaţional, oferind un instrument eficient pentru alocarea resurselor şi generarea de orare fiabile, rapide şi uşor accesibile. Soluţia propusă susţine digitalizarea completă a activităţilor academice şi vine în sprijinul personalului didactic şi administrativ, reducând sarcinile repetitive şi optimizând procesul de planificare.

CAPITOLUL 1. STUDIUL PROBLEMEI ŞI ANALIZA SOLUȚIILOR EXISTENTE

Pentru dezvoltarea unei aplicaţii sustenabile şi eficace, a fost necesar un studiu preliminar care să includă o analiză generală a aplicațiilor deja existente pe piaţă. Această analiză pune accentul pe funcţionalităţile oferite, modul în care utilizatorul interacţionează cu interfaţa, precum şi pe identificarea elementelor ce pot fi îmbunătăţite pentru a asigura o navigare cât mai intuitivă şi eficientă.

Ordonarea activităţilor academice într-un mod coerent şi eficient reprezintă o sarcină esenţială pentru orice instituţie de învăţământ superior. Generarea manuală a orarului universitar este un proces complex şi consumator de timp, care trebuie să ţină cont de numeroşi factori: disponibilitatea cadrelor didactice, distribuţia disciplinelor, capacitatea şi tipul sălilor de curs, dar şi compatibilitatea dintre grupe și subgrupe. În absenţa unei soluţii automatizate, acest demers poate conduce la erori, suprapuneri sau programări ineficiente.

În acest context, implementarea unei aplicaţii automatizate pentru generarea orarului devine nu doar oportună, ci necesară. O astfel de soluţie reduce considerabil efortul administrativ şi oferă rezultate rapide, coerente și adaptabile specificului fiecărei facultăţi sau specializări.

Acest capitol are ca scop evidenţierea importanţei unui sistem de tip TTGS (Timetable Generation System), analiza comparativă a aplicaţiilor existente cu funcţionalităţi similare, definirea profilului utilizatorului căruia i se adresează aplicaţia, precum şi prezentarea funcţionalităţilor de bază dezvoltate în cadrul acestei lucrări.

1.1 Motivaţia utilizării unei aplicaţii automate pentru generarea orarului universitar

Generarea orarului universitar reprezintă o activitate esenţială, dar deosebit de complexă din punct de vedere administrativ. Aceasta necesită luarea în considerare a numeroşi factori, precum: disponibilitatea profesorilor, alocarea disciplinelor, capacitatea şi tipul sălilor în care se desfăşoară activităţile, gruparea studenţilor în funcţie de nivel (an, grupă sau subgrupă), precum şi evitarea suprapunerilor sau a pauzelor nejustificate din program.

În prezent, în multe instituţii de învăţământ, acest proces este realizat manual sau cu ajutorul unor aplicaţii semi-automatizate. Drept urmare, se consumă mult timp, pot apărea erori frecvente, conflicte de program, distribuţii dezechilibrate ale activităţilor 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 tendinţă clară spre automatizarea acestui proces prin utilizarea sistemelor informatice dedicate. Platforme precum UniTime [1], aSc TimeTables [2] sau GeneratorOrare [3] oferă facilităţi avansate pentru planificarea automată a orarelor. Acestea permit definirea de constrângeri rigide şi flexibile (hard şi soft constraints), gestionarea preferinţelor cadrelor didactice şi generarea orarelor

optimizate, folosind algoritmi de satisfacere a constrângerilor (CSP) 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 universitar 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 păstrând, pe cât posibil, structura inițială. Astfel, adaptarea la schimbări este rapidă și fără impact major asupra planificării generale.
- 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, contribuind la un proces educațional transparent și eficient.

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 aplicaţiilor existente pentru generarea orarului

Pentru a fundamenta procesul de proiectare al aplicaţiei propuse, a fost realizată o analiză comparativă a principalelor soluţii existente pe piaţă destinate generării automate a orarelor universitare. Această analiză s-a concentrat pe modul de funcţionare al aplicaţiilor, gradul de interactivitate oferit utilizatorului, particularităţile interfeţei grafice şi aspectele care pot fi optimizate pentru a facilita o utilizare mai intuitivă şi eficientă.

În urma analizării mai multor platforme, s-a constatat că multe aplicaţii actuale dispun de interfeţe greoaie, dificil de utilizat şi configurat, necesitând adesea instalare locală şi un consum ridicat de resurse. În plus, unele soluţii sunt costisitoare, au timpi mari de procesare şi presupun o curbă de învăţare ridicată pentru utilizatorii fără experienţă tehnică. Suplimentar, afişarea simultană a unor volume mari de informaţii într-o singură vizualizare poate afecta negativ experienţa de utilizare, ducând la confuzie sau omisiuni.

Pentru această analiză, au fost selectate trei aplicaţii reprezentative: UniTime, aSc TimeTables şi Generator-Orare (Horarium.ai). Criteriile de selecţie au inclus notorietatea în domeniu, diversitatea funcţionalităţilor oferite şi diferenţele de abordare tehnică şi practică. Scopul acestei analize a fost identificarea punctelor forte şi a limitărilor fiecărei aplicaţii, în vederea extragerii unor concluzii utile pentru dezvoltarea unei soluţii personalizate, adaptate nevoilor specifice ale instituţiilor de învăţământ superior.

1.2.1 UniTime

UniTime este un sistem open-source de programare academică, dezvoltat iniţial de Universitatea Indiana şi ulterior adoptat de numeroase colegii şi universităţi din întreaga lume [1]. Popularitatea sa se datorează nivelului ridicat de configurabilitate şi capacităţii de extindere, fiind potrivit mai ales pentru instituţii de mari dimensiuni (Figura 1).

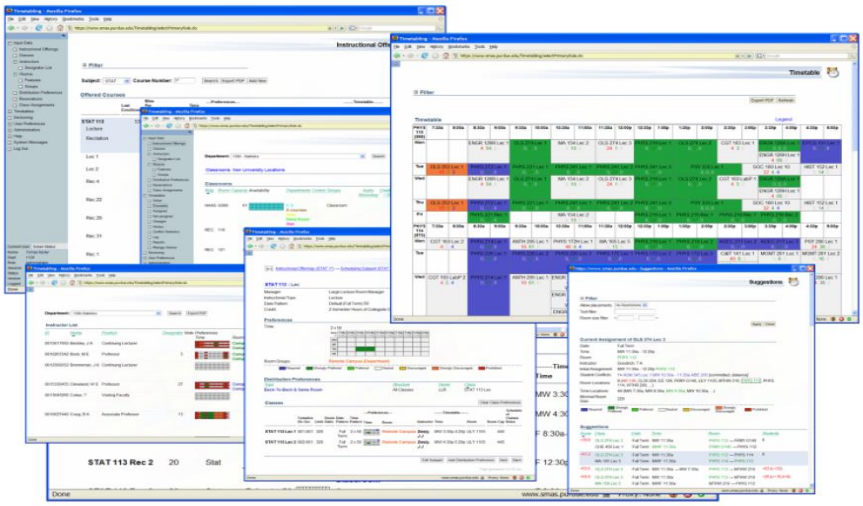

Fig. 1 Interfaţa aplicaţiei UniTime – module de alocare, orar şi sugestii automate

Aplicaţia oferă funcţionalităţi avansate, printre care:

- definirea preferinţelor cadrelor didactice;
- alocarea automată a resurselor (săli, clădiri, echipamente);
- aplicarea de constrângeri rigide şi flexibile (hard/soft constraints);
- selecţia algoritmului de programare (solver);
- exportul orarului în format iCalendar;
- integrarea cu sisteme externe existente;
- vizualizarea programelor personalizate pentru studenţi şi profesori.

Un avantaj semnificativ este posibilitatea de a configura în detaliu regulile şi constrângerile, adaptând planificarea la particularităţile instituţiei. Interfaţa de administrare permite gestionarea sesiunilor, resurselor, activităţilor şi aspectelor vizuale ale programului.

Totuşi, în ciuda versatilităţii sale, UniTime prezintă şi o serie de dezavantaje majore:

- Instalarea este complexă, necesitând cunoştinţe tehnice avansate şi configurarea unor componente multiple precum Java, Apache Tomcat, MySQL şi fişiere de sistem (ex: custom.properties). Instituţiile fără o echipă IT dedicată pot întâmpina dificultăţi în utilizarea platformei.
- Interfaţa grafică, deşi funcţională, nu este prietenoasă pentru utilizatorii non-tehnici. Navigarea poate fi dificilă la început, în special din cauza lipsei unei documentaţii clare şi concentrate pe utilizatorul final. De asemenea, aplicaţia nu este optimizată pentru dispozitive mobile.
- Comunitatea de utilizatori este limitată, iar suportul tehnic oficial lipseşte, ceea ce face rezolvarea problemelor dependentă de documentaţia tehnică, adesea fragmentată şi dificil de înţeles pentru personalul administrativ.

În concluzie, UniTime este o soluţie foarte puternică şi flexibilă pentru programarea academică, recomandată în special instituţiilor mari, care pot valorifica avantajele sale prin personal tehnic specializat. Cu toate acestea, complexitatea sa de configurare şi utilizare o face mai puţin potrivită pentru instituţiile de dimensiuni mici sau pentru utilizatorii fără competenţe tehnice solide.

1.2.2 aSc TimeTables

aSc TimeTables este una dintre cele mai cunoscute aplicaţii comerciale pentru generarea automată a orarelor, fiind utilizată pe scară largă în instituţii de învăţământ din ţări precum Germania, Statele Unite, India, Brazilia, dar şi în numeroase state din Africa şi Australia [2]. Aplicaţia este disponibilă atât sub formă de program desktop pentru Windows, cât şi ca serviciu online, oferind planuri tarifare variate – de la 99euro (pentru şcoli primare) până la 1995 euro (versiunea profesională).

Aplicaţia este apreciată pentru gradul ridicat de automatizare, interfaţa prietenoasă şi simplitatea cu care un orar complet poate fi generat în doar câteva clicuri. Introducerea datelor iniţiale – precum clase, profesori, discipline şi reguli de planificare – se face intuitiv, fără a necesita pregătire tehnică prealabilă. Algoritmul încorporat

evaluează milioane de combinaţii posibile în câteva secunde, generând un orar optimizat care respectă toate constrângerile: absenţa suprapunerilor, menţinerea perioadelor libere pentru profesori, utilizarea eficientă a sălilor şi evitarea blocajelor orare (Figura 2).

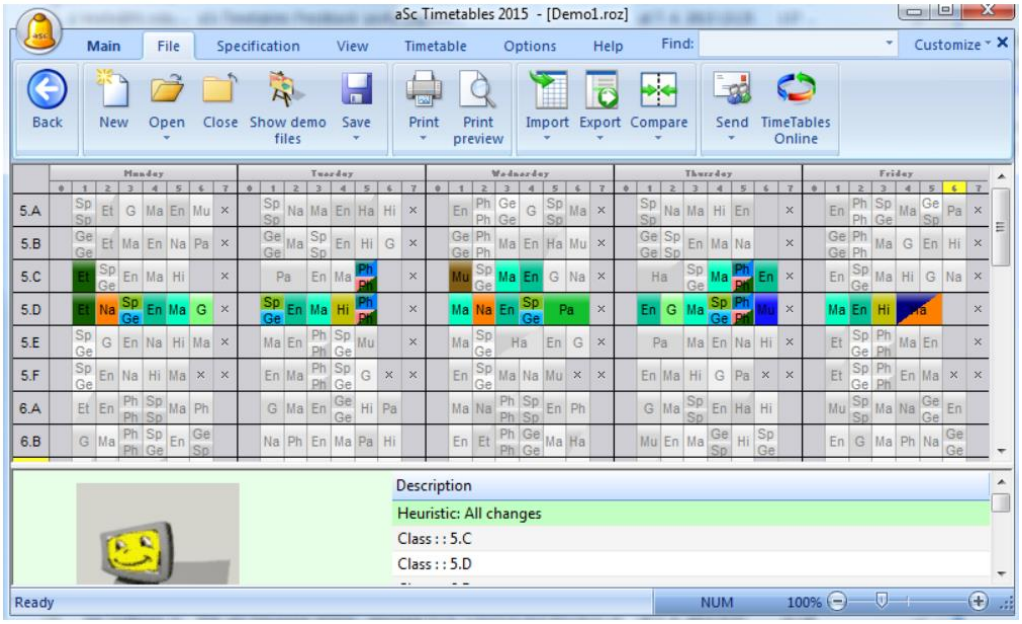

Fig. 2 Interfaţa aplicaţiei aSc TimeTables – vizualizarea orarului pe clase şi zile

Un alt punct forte este posibilitatea de export al orarului în multiple formate – PDF, Excel, HTML sau imagine – precum şi integrarea cu platforma EduPage, care permite gestionarea elevilor, cataloage electronice, prezenţă şi comunicarea cu părinţii. Disponibilitatea aplicaţiei în limba română şi existenţa unei versiuni adaptate sistemului educaţional românesc constituie un avantaj semnificativ pentru utilizatorii locali. Suportul tehnic este prompt, iar documentaţia detaliată şi tutorialele uşor de urmărit facilitează accesul chiar şi pentru utilizatorii fără cunoştinţe IT avansate.

Totuşi, aplicaţia prezintă şi limitări importante:

- Versiunea gratuită este extrem de restrictivă, lipsind funcţionalităţi esenţiale precum exportul şi salvarea orarului, ceea ce o face aproape inutilizabilă fără achiziţionarea unei licenţe.
- Gradul de personalizare este limitat comparativ cu alte platforme mai flexibile. Nu permite definirea de reguli avansate prin cod sau configurarea unor algoritmi proprii.
- Versiunea desktop necesită instalare locală şi funcţionarea ei completă depinde de conexiunea la internet, mai ales pentru funcţiile avansate precum integrarea cu EduPage sau partajarea online.

În concluzie, aSc TimeTables este o soluţie performantă pentru generarea automată a orarelor, recomandată în special instituţiilor de învăţământ preuniversitar sau universitar cu cerinţe de complexitate medie. Deşi este mai puţin flexibilă şi extensibilă decât alte aplicaţii avansate, rămâne o alegere excelentă datorită interfeţei intuitive, rapidităţii de generare şi simplităţii în utilizare.

1.2.3 Generator-Orare (Horarium.ai)

Generator-Orare, redenumit recent Horarium.ai, este o aplicaţie online dedicată programării automate a orarelor şcolare [3]. Spre deosebire de alte soluţii mai tehnice sau greu de configurat, Horarium se remarcă prin simplitatea utilizării şi prin interfaţa intuitivă, fiind accesibil direct din browser, fără a necesita instalare locală sau configuraţii suplimentare (Figura 3).

Fig. 3 Interfaţa aplicaţiei Horarium.ai – vizualizare orar generat

Sistemul este complet automatizat şi ghidează utilizatorul prin fiecare etapă a procesului. După introducerea elementelor esenţiale (profesori, materii, clase, săli şi reguli generale), aplicaţia generează orarul respectând constrângerile tipice impuse în sistemul educaţional. Interfaţa este adaptată utilizatorilor non-tehnici, ceea ce o face ideală pentru personalul secretarial sau administrativ fără pregătire IT avansată. Navigarea în aplicaţie este fluidă, iar rezultatele sunt afişate într-un mod clar, lizibil şi structurat.

Un avantaj important este faptul că platforma nu solicită efort tehnic din partea utilizatorului – majoritatea acţiunilor fiind automatizate şi uşor de urmărit. Designul este responsive şi poate fi accesat şi de pe dispozitive mobile, însă experienţa optimă rămâne pe ecranele mari, pentru o vizualizare completă a orarului.

Totuşi, Horarium.ai prezintă şi unele limitări semnificative:

- Nu dispune de o versiune mobilă dedicată şi, deşi interfaţa este responsivă, funcţionalitatea completă nu este optimizată pentru utilizarea pe telefoane sau tablete.
- Nu permite configurarea de constrângeri avansate sau definirea unor reguli complexe, precum cele legate de preferinţele profesorilor, intervale orare rezervate sau interdependenţe între activităţi.
- Nu oferă posibilitatea de import/export extins sau integrare cu sisteme externe prin API.

- Platforma este accesibilă contra cost, iar modelul de tarifare poate deveni o barieră pentru unele instituţii, în lipsa unei versiuni gratuite funcţionale.

În concluzie, Horarium.ai este o soluţie eficientă pentru instituţiile care îşi doresc un instrument simplu, rapid şi accesibil pentru generarea orarelor. Este ideală pentru utilizatorii care nu dispun de resurse tehnice avansate şi doresc un rezultat funcţional cu un efort minim. Cu toate acestea, pentru aplicaţii de nivel universitar sau pentru integrarea într-un sistem educaţional complex, limitările sale privind flexibilitatea şi personalizarea pot deveni constrângătoare.

1.3 Profilul utilizatorului şi scopul utilizării aplicaţiei

Aplicaţia descrisă în acest document a fost creată folosind un model centralizat de utilizare, astfel încât acelaşi utilizator, având rolul de administrator complet, este responsabil de toate etapele de configurare şi de generare a orarului universitar. Adaptarea acestui model pentru un singur utilizator se datorează simplităţii şi eficienţei sale funcţionale şi, de asemenea, pentru că reproduce lumea reală de operare care se întâmplă în multe instituţii mici şi 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 activităţile academice).

Utilizatorul final nu este împărţit în roluri (de exemplu, studenţi, profesori, secretariat sau biroul decanului), având acces complet la toate părţile sistemului. Acest utilizator elaborează informaţiile necesare despre personalul didactic, disciplinele aferente, grupurile şi subgrupurile de studenţi, sălile disponibile şi, de asemenea, despre regulile şi constrângerile impuse în procesul de planificare. În acest sens, aplicaţia nu este un mediu colaborativ, ci funcţionează ca un instrument de lucru pentru utilizatorul care deţine toate drepturile asupra întregii proceduri.

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

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

Obiectivele aplicaţiei sunt:

- Centralizarea şi intuitivizarea procesului de planificare academică: Platforma oferă o interfaţă prietenoasă, modernă şi centralizată, care facilitează accesul utilizatorilor la toate funcţionalităţile necesare generării şi ajustării unui orar universitar complet.
- Accelerarea procesului de generare a orarului: Comparativ cu metodele tradiţionale de planificare (precum utilizarea fişierelor Excel sau completarea manuală), aplicaţia 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. Aplicația este suficient de robustă pentru a susține scenarii de reconfigurare rapidă, păstrând totodată consistența planificării.
- Export facil în formate lizibile și compatibile: Orarul generat poate fi exportat în formate comune și ușor de distribuit (PDF, Excel, HTML), 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 universitar, 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 universitar 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 nume, discipline predate, tipuri de activități, niveluri de studii la care vor preda 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

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

Prin această aplicaţie, se urmăreşte automatizarea întregului flux de planificare, oferind utilizatorului o interfaţă intuitivă şi un set complet de funcţionalităţi pentru introducerea datelor, generarea şi validarea unui orar coerent, complet şi adaptat cerinţelor instituţionale.

2.1 Cerinţe privind gestionarea şi prelucrarea datelor

Pentru a asigura un proces complet de generare a orarului, aplicaţia trebuie să gestioneze în mod eficient toate datele esenţiale implicate în organizarea academică. Această componentă a sistemului vizează operaţiile necesare pentru introducerea, modificarea, căutarea şi stocarea datelor referitoare la grupe, săli, profesori şi reguli de generare. Gestionarea corectă a acestor informaţii constituie baza funcţională pentru o planificare coerentă şi automatizată a activităţilor didactice.

Sistemul va permite următoarele operaţii:

1. Introducerea datelor:

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

2. Modificarea datelor:

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

3. Căutare şi filtrare:

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

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 universitar. 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. DENUMIREA CAPITOLULUI

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

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

3.1 Arhitectura și funcționalitatea proiectului

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

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

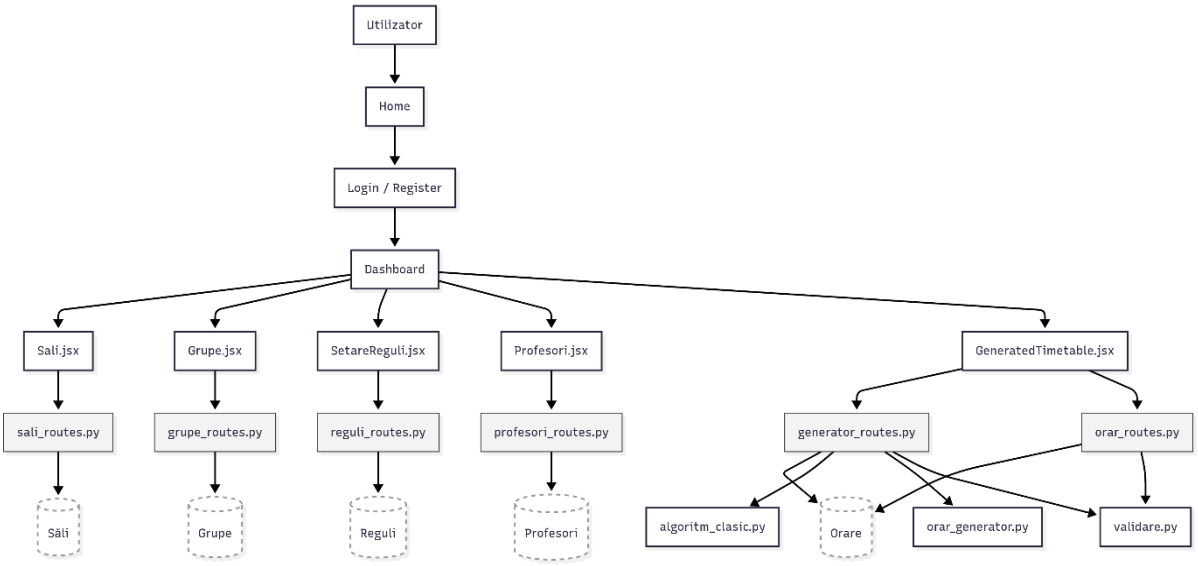

Fig. 4 Diagrama arhitecturală a aplicației

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.

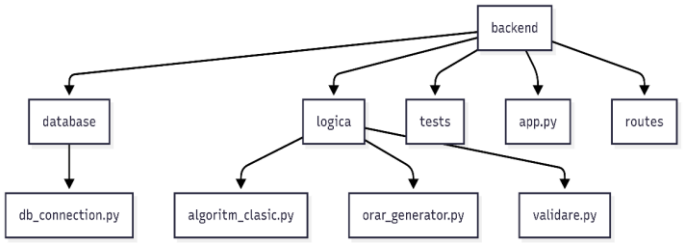

Fig. 5 Diagrama pachetelor- Backend

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.

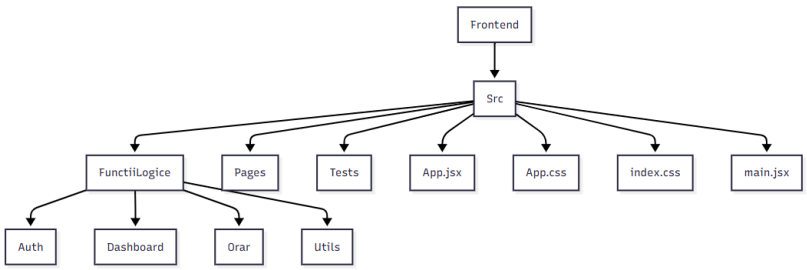

Fig. 6 Diagrama pachetelor- Frontend

3. Rutele din backend sunt organizate în mod modular, fiecare fişier având o responsabilitate clară în gestionarea unei funcţionalităţi specifice. Aşa cum se observă în Figura 7, toate aceste rute sunt centralizate în modulul routes, care le grupează şi le integrează în aplicaţia principală. Această structurare permite o mai bună organizare a codului, o gestionare eficientă a cererilor API şi o extensibilitate uşoară în cazul adăugării de noi funcţionalităţi.

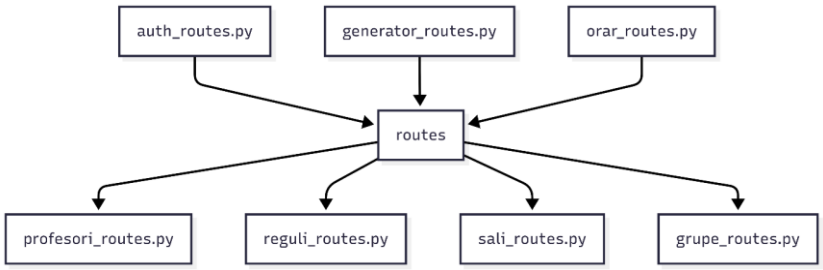

Fig. 7 Diagrama pachetelor- Routes

4. Directorul Pages reuneşte toate componentele vizuale principale ale aplicaţiei, fiecare dintre ele corespunde unei funcţionalităţi esenţiale. După cum este ilustrat în Figura 8, aceste pagini sunt responsabile pentru interacţiunea directă cu utilizatorul – de la autentificare şi administrare, până la generarea şi vizualizarea orarului. Structura este clară şi intuitivă, permiţând o navigare fluentă între secţiunile aplicaţiei şi o experienţă de utilizare coerentă.

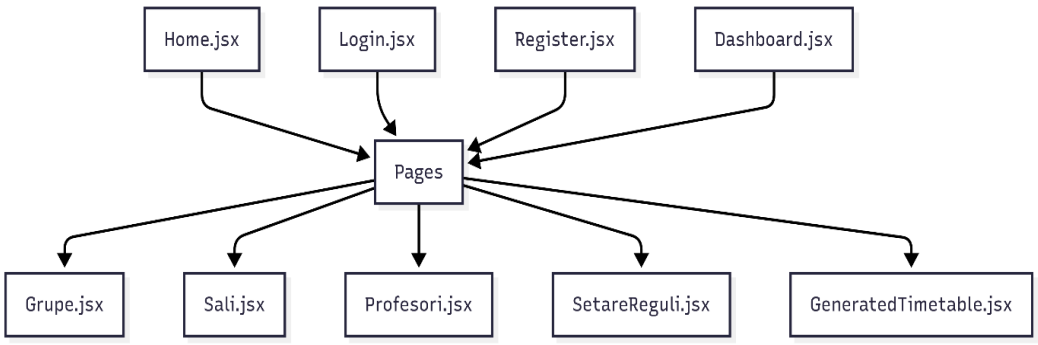

Fig. 8 Diagrama pachetelor- Pages

3.2.2 Diagrama claselor

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

Structura evidenţiază clasele responsabile de generarea orarului (fie prin algoritm AI, fie prin algoritm clasic), validarea acestuia şi accesul la date. Relaţiile dintre clase sunt marcate prin asocieri clare, ilustrând dependenţele funcţionale din cadrul sistemului. Acest model contribuie la o înţelegere clară a arhitecturii interne şi oferă o bază solidă pentru extinderea ulterioară a funcţionalităţilor.

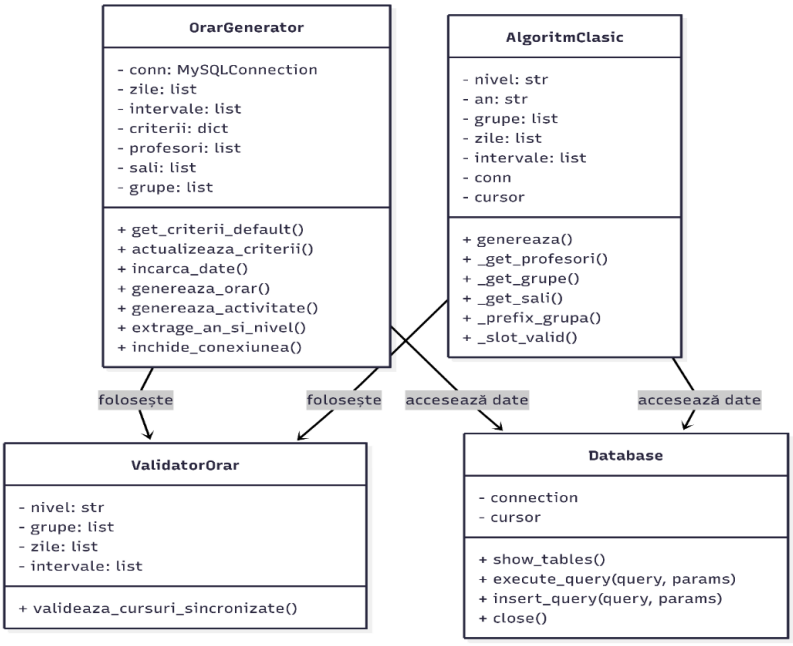

Fig. 9 Diagrama clasei -Backend

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.

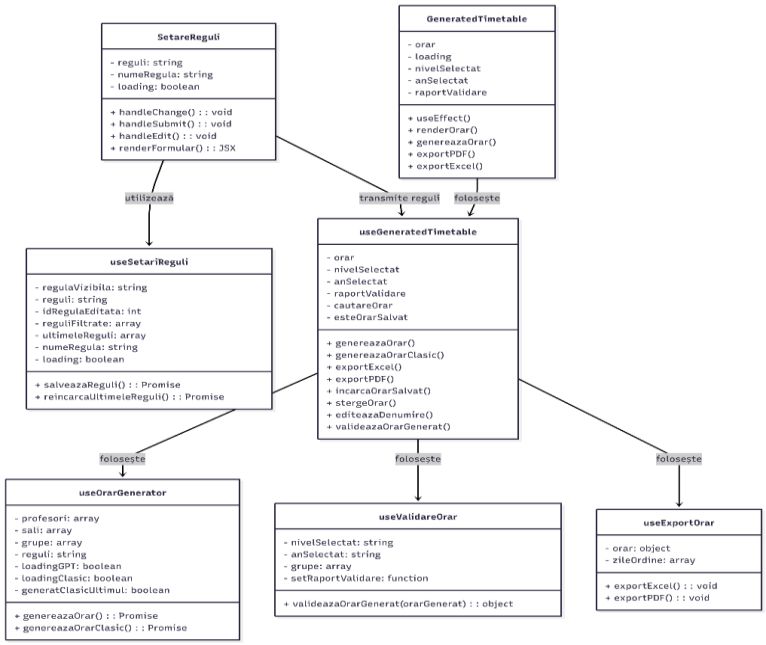

Fig. 10 Diagrama claselor- Frontend

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.

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.

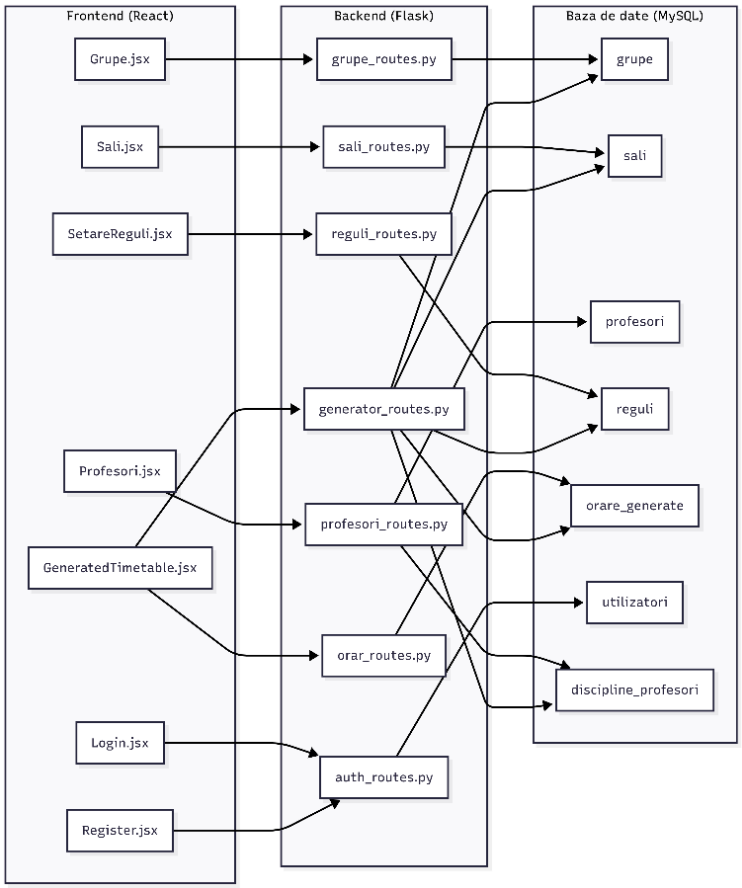

Fig. 11 Diagrama de componente ale aplicației

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.

3.3.1 Diagrama cazurilor de utilizare

Figura 12 ilustrează cazurile de utilizare asociate interacţiunii utilizatorului cu aplicaţia. Diagrama evidenţiază principalele funcţionalităţi 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 declanşeze procesul de generare propriu-zis, să vizualizeze rezultatul şi, la final, să exporte orarul în format PDF sau Excel. Aceste acţiuni acoperă întregul ciclu de utilizare al aplicaţiei şi reflectă funcţionalităţile esenţiale puse la dispoziţie într-un mod intuitiv.

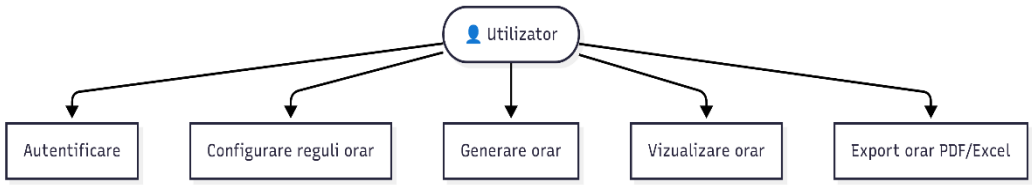

Fig. 12 Diagrama cazurilor de utilizare

3.3.2 Diagrama de secvenţă

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

Procesul începe cu acţiunea utilizatorului, care declanşează cererea de generare. Datele şi regulile sunt procesate secvenţial, transmise către modelul AI, iar răspunsul obţinut este salvat şi apoi returnat către frontend pentru afişare. Diagrama oferă o imagine clară asupra colaborării dintre module în cadrul acestui flux automatizat.

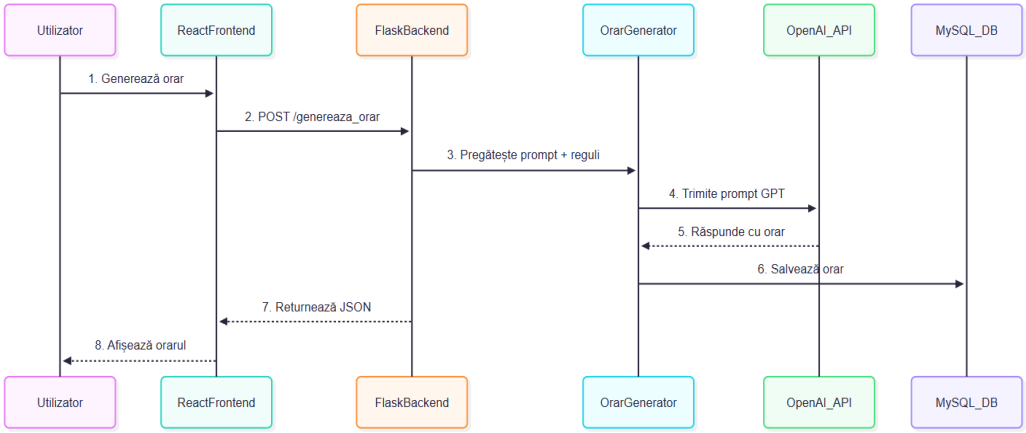

Fig. 13 Diagrama de secvenţă

3.3.3 Diagrama de activitate

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

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

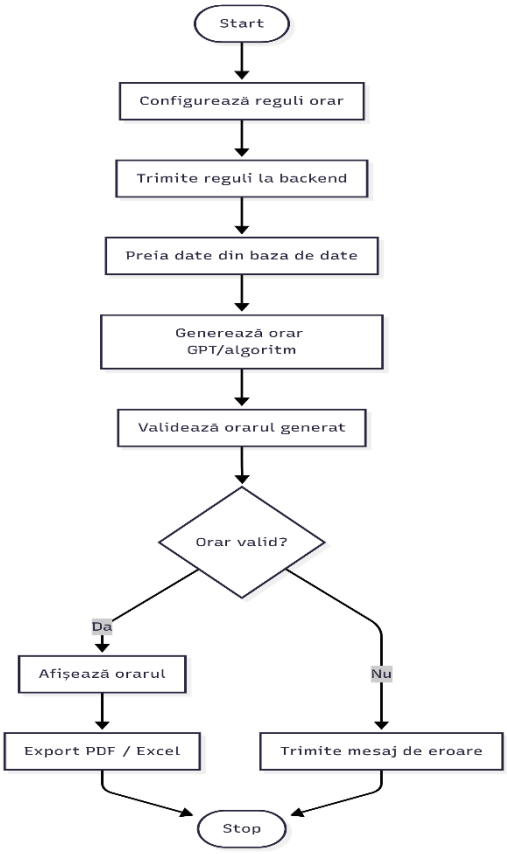

Fig. 14 Diagrama de activitate

CAPITOLUL 4. IMPLEMENTAREA ŞI DESCRIEREA APLICAȚIEI

4.1 Limbajele de programare implicate în dezvoltarea aplicației

4.2 Framework-urile şi bibliotecile implicate în dezvoltarea aplicației

4.3 Implementarea şi funcționalitatea aplicației

4.4 Probleme întâmpinate în timpul implementării

CAPITOLUL 5. TESTAREA 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 reacțiilor 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 (dashboard).

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: „Completați toate câmpurile”.

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: „Numele este obligatoriu".

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.

Scenariul 4 – Orar incomplet/invalid
- Pasul 1: Se generează orar cu date lipsă.
- Rezultat: Exportul nu este disponibil.

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.

5.2. 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 universitare, 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: realizată cu ajutorul bibliotecii pytest în Python, a vizat testarea rutelor API, interacțiunii cu baza de date, integritatea datelor și răspunsurile corecte la solicitările HTTP.
- Testarea frontendului: realizată cu ajutorul bibliotecii Jest și a utilitarului React Testing Library, a urmărit verificarea componentelor vizuale, interacțiunilor utilizatorului, validarea formularelor și afișarea corectă a datelor în interfață.

Testarea automată a permis dezvoltarea controlată a aplicației și a contribuit la menținerea calității codului pe parcursul întregului ciclu de dezvoltare. În continuare, vor fi prezentate testele relevante implementate pentru componentele backend și frontend, împreună cu explicațiile aferente.

5.2.1 Testarea backend

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

Fișierul test_api_generator.py include testele unitare pentru rutele API responsabile de generarea orarului, din blueprint-ul generator_bp. Testele au fost realizate cu pytest și Flask test_client, permițând simularea cererilor HTTP fără rularea serverului.

Au fost verificate următoarele scenarii:

- GET /genereaza_orar_propriu – pagina de generare se afișează corect.
- POST valid – trimite date complete și primește un orar valid.
- POST invalid – trimite date lipsă și primește eroare 400.

Toate testele au fost executate cu succes, validând funcționarea corectă a API-ului în scenarii diverse.

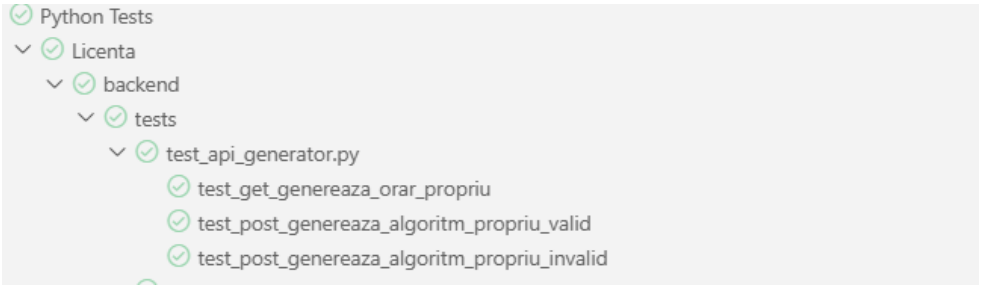

Fig. 15 Testarea funcționalității de generare orar

5.2.2 Testarea gestionării profesorilor

Fișierul test_api_profesori.py conține teste automate pentru operațiile CRUD legate de profesori (creare, listare, actualizare, ștergere). Testele au fost realizate cu pytest și Flask test_client, simulând cereri HTTP fără rularea serverului.

Testele acoperă:

- Adăugare profesor cu date complete (inclusiv discipline și disponibilitate).
- Verificare în listă după adăugare (/toti_profesorii).
- Actualizare profesor existent (nume, activități, disponibilitate).
- Ștergere profesor pentru curățare (cleanup).

Toate testele au trecut cu succes, validând funcționarea corectă a backendului pentru gestionarea profesorilor.

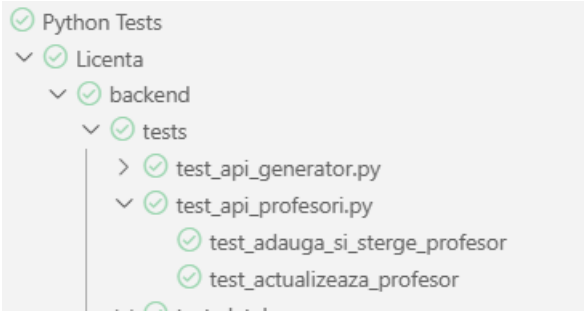

Fig. 16 Testarea gestionării profesorilor

5.2.3 Testarea interacțiunii directe cu baza de date

Fișierul test_database.py verifică operațiile SQL fundamentale folosind clasa personalizată Database, fără a implica API-ul Flask.

Testele acoperă:

- test_show_tables – verifică existența tabelei profesori.
- test_select_profesori – validează interogările SELECT și structura rezultatelor.
- test_insert_and_delete_profesor – simulează adăugarea și ștergerea unui profesor de test, confirmând integritatea operațiilor.

Toate testele au fost rulate cu succes, confirmând funcționarea corectă a conexiunii și manipulării datelor în MySQL.

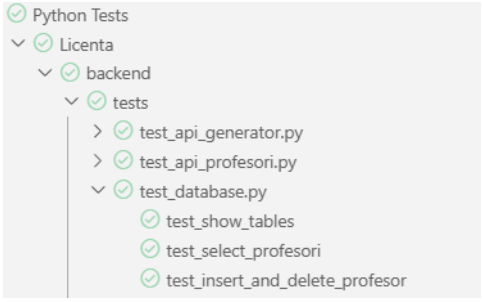

Fig. 17 Testarea interacțiunii directe cu baza de date

5.2.4 Testarea generatorului propriu de orar

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

Testele acoperă:

- 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.

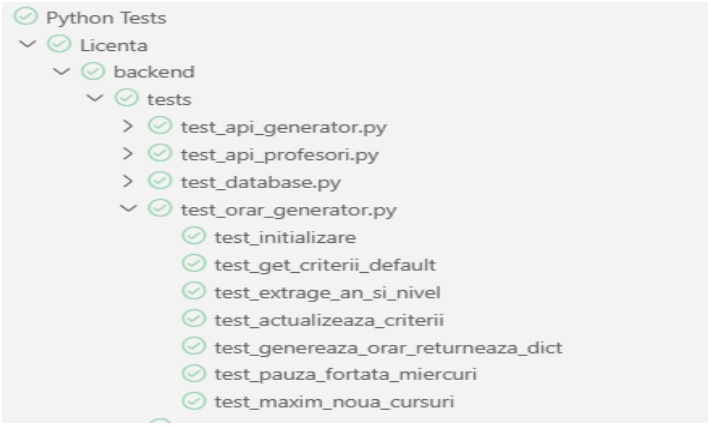

Fig. 18 Testarea generatorului propriu de *orar*

5.2.5 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.

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.

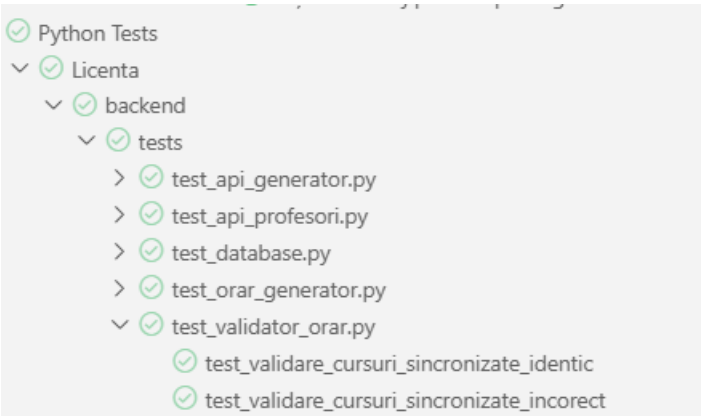

Fig. 19 Testarea validării orarului generat

5.3 Testarea frontend

Testarea automată a componentelor frontend are un rol esențial în asigurarea funcționării corecte a interfeței utilizator. Într-o aplicație modernă construită cu React, testele frontend validează dacă elementele vizuale sunt afișate corect, dacă interacțiunile utilizatorului produc rezultatele așteptate și dacă starea aplicației se actualizează corespunzător în urma acestor interacțiuni.

Pentru acest proiect, testarea frontend a fost realizată folosind biblioteca Jest, împreună cu React Testing Library, două dintre cele mai populare unelte pentru testarea componentelor React. Acestea permit simularea comportamentului real al utilizatorului (de exemplu: apăsarea butoanelor, completarea câmpurilor, selectarea opțiunilor din formulare), precum și validarea conținutului vizual afișat în pagină.

Testele au fost organizate pe fișiere separate, câte unul pentru fiecare componentă importantă din aplicație, cum ar fi:

- pagina principală (Home.test.jsx),
- formularul de introducere a profesorilor (Profesori.test.jsx),
- formularul de setare reguli (SetareReguli.test.jsx),
- pagina de generare și afișare a orarului (GeneratedTimetable.test.jsx).

În cadrul fiecărui test, s-au verificat:

- prezența în DOM a elementelor cheie (titluri, butoane, formulare),
- comportamentul interactiv al utilizatorului,

- reacția componentelor la date de intrare și acțiuni,
- eventuale mesaje de eroare sau confirmare.

Prin această suită de teste, se asigură că aplicația răspunde corect la acțiunile utilizatorului, iar modificările aduse interfeței nu introduc erori de afișare sau funcționare.

5.3.1 Testarea componentei GeneratedTimetable.jsx

Componenta GeneratedTimetable este una dintre cele mai complexe, responsabilă pentru generarea, afișarea, validarea și gestionarea orarelor universitare. Testarea s-a realizat cu Jest și React Testing Library, utilizând mock-uri pentru toate hook-urile logice (useOrarGenerator, useValidareOrar, useOrarSalvat etc.).

Testele acoperă:

- Afișarea titlului și butoanelor principale: „Generează cu AI” și „Generează clasic”.
- Apelul corect al funcțiilor generezaOrar și generezaOrarClasic la click.
- Funcționarea butoanelor auxiliare: „Reîncarcă”, „Înapoi”.
- Afișarea listei de orare salvate (cu opțiuni de editare, ștergere, încărcare).
- Filtrarea orarelor după denumire.

Toate testele au trecut cu succes, validând comportamentul complet și robust al acestei componente critice.

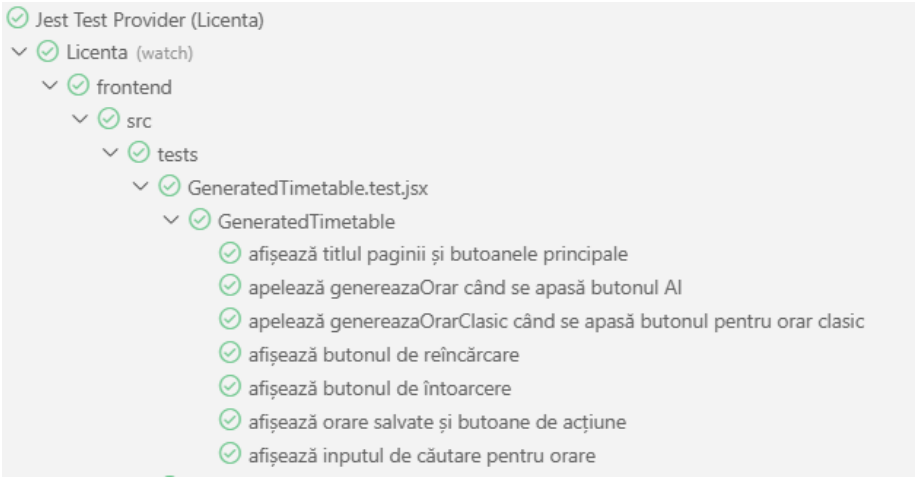

Fig. 20 Testarea componentei GeneratedTimetable.jsx

5.3.2 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).

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.

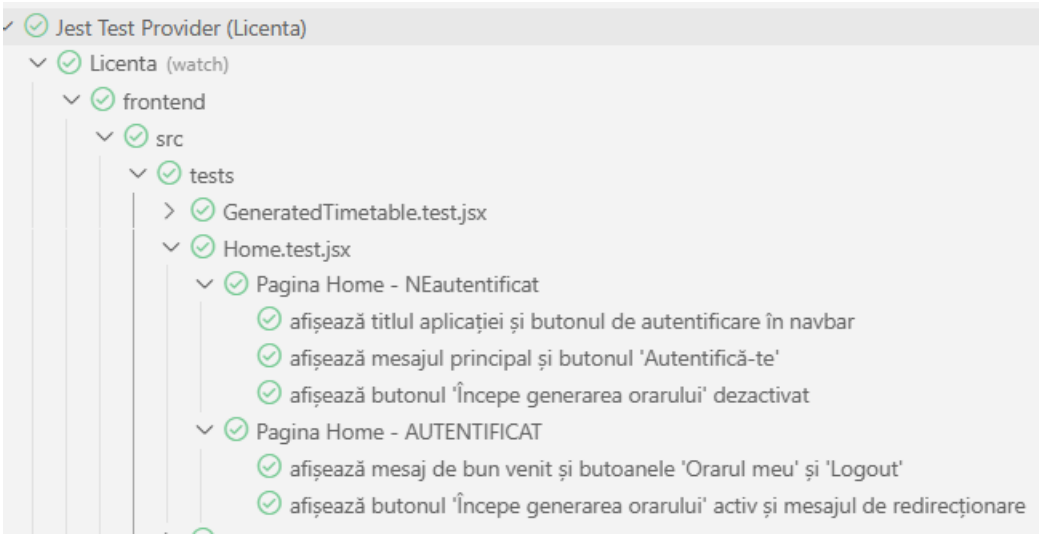

Fig. 21 Testarea componentei Home.jsx

5.3.3 Testarea componentei Profesori.jsx

Testarea componentei Profesori – Profesori.test.jsx

Componenta Profesori.jsx permite adăugarea şi configurarea profesorilor (nume, discipline, nivel, tip activitate, disponibilitate). Este una dintre cele mai complexe componente, motiv pentru care a fost testată automat cu Jest şi React Testing Library, folosind mock pentru useProfesoriLogic.

Testele acoperă:

- Afişarea formularului şi titlului principal.
- Comportamentul corect când lista este goală (*„Niciun profesor găsit”*).
- Funcţionalitatea butonului „Salvează profesor” (apelează adaugaProfesor).
- Posibilitatea de a adăuga mai multe discipline.
- Afişarea tabelului de selecţie a disponibilităţii săptămânale.

Toate testele au trecut cu succes, confirmând funcţionalitatea completă a interfeţei şi reacţia corectă la diverse stări ale aplicaţiei.

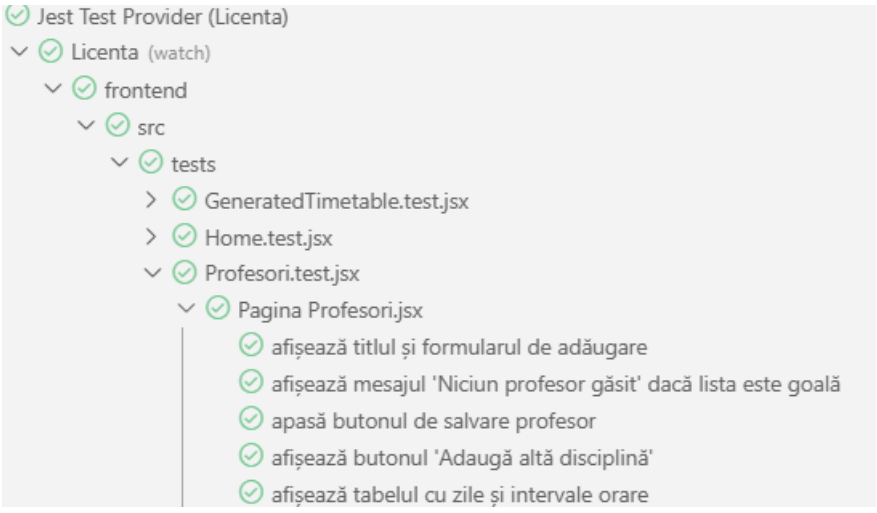

Fig. 22 Testarea componentei Profesori.jsx

5.3.4 Testarea componentei SetareReguli.jsx

Componenta SetareReguli permite utilizatorului să definească, salveze și reutilizeze reguli personalizate în format JSON pentru generarea orarului. 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.

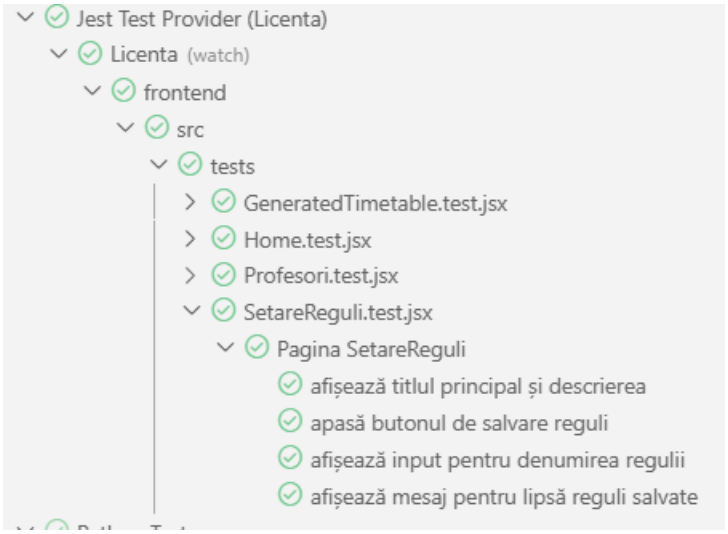

Fig. 23 Testarea componentei SetareReguli.jsx

CONCLUZII

Vor cuprinde într-o formă cât mai concisă principale rezultatele obţinute în tema tratată, subliniindu-se contribuţia adusă prin propriile cercetări. Se vor scoate în evidenţă elementele de noutate ale proiectului/lucrării. Dacă rezultatele obţinute pot fi aplicate în activitatea de cercetare, producţie sau în alte domenii de activitate, economică sau socială, se vor face recomandările corespunzătoare.

Un bilanţ al aspectelor pozitive şi negative din activitatea de dezvoltare a proiectului de diplomă sau a lucrării de disertaţie va încheia partea scrisă a lucrării.

BIBLIOGRAFIE

[1] https://www.unitime.org
[2] https://www.asctimetables.com
[3] https://generator-orare.ro/presentation
[4] https://platform.openai.com/docs/models/gpt-4.1
[5] https://generator-orare.ro/profile
[6] https://support.microsoft.com/ro-ro/topic/crearea-unei-diagrame-de-clasă-uml-de6be927-8a7b-4a79-ae63-90da8f1a8a6b
[7] http://easy-learning.neuro.pub.ro:8888/Laboratoare/_Mase/L08_uml/UML.htm
[8] https://code.visualstudio.com/docs
[9] https://devdocs.io/javascript/
[10] https://docs.python.org/3/
[11] https://devdocs.io/html/
[12] https://devdocs.io/css/
[13] https://www.phpmyadmin.net/docs/
[14] https://legacy.reactjs.org
[15] https://flask.palletsprojects.com/en/stable/
[16] https://platform.openai.com/docs
[17] https://getbootstrap.com/docs/5.0/getting-started/introduction/
[18] https://www.w3schools.com/jsref/api_fetch.asp
[19] https://sweetalert.js.org/docs/
[20] https://docs.sheetjs.com/docs/
[21] https://www.html2pdf.co.uk/api-documentation

Bibliografia cuprinde cărţi, capitole din cărţi, articole tipărite, articole şi lucrări prezentate la conferinţe şi disponibile on line, site-uri.

Redactarea se va face aplicând criteriul alfabetic numelor autorilor. Dacă acelaşi autor a scris mai multe lucrări care apar în bibliografie, acestea se vor include în ordine cronologică, de la cea mai veche spre cea editată recent. Vor fi menţionate: numele autorului/autorilor, titlul integral al lucrării, editura, locul publicării, anul apariţiei lucrării.

Pentru articolele apărute în reviste, se menţionează în ordine: numele autorului/autorilor, titlul articolului, titlul revistei, ISBN/ISSN, anul apariţiei, numărul.

Paginile web se vor ordona alfabetic şi vor avea menţionată data accesării lor. Această precizare va fi făcută deoarece există posibilitatea ca pagina citată de pe internet să fie ulterior ştearsă.

[22] I. Voncilă, D. Călueanu, N. Badea, R. Buhosu, Cr. Munteanu, „Maşini Electrice”, Editura Fundaţiei Universitare „Dunărea de Jos” din Galaţi, 2003;
[23] I. Şuşnea, M. Vlase, „A software instrument for the assessment of creativity in the educational environment”, The Annals of the University "Dunărea de Jos" of Galaţi

Romania Fascicle III Electrotechnics Electronics Automatic Control Informatics, ISSN 1221-454X, 2016, Volum 39, Numărul 1;

[24] https://ro.wikipedia.org/wiki/Tranzistor, 23-06-2016.

ANEXA 1

Anexele (dacă există) – nu se numerotează ca şi capitol, se numerotează crescător (Anexa 1, Anexa 2, etc).

Anexele vor conține elemente precum:

- porţiuni de cod;
- tabele de date;
- tabele de rezultate de ieşire;
- alte elemente specifice la care s-a făcut referire în cadrul proiectului/lucrării.