# Capitolul 4. Implementarea și descrierea aplicației

Acest capitol detaliază procesul de realizare a aplicației destinate generării automate a orarului universitar, utilizând tehnici moderne de inteligență artificială. Sunt prezentate limbajele de programare, framework-urile și bibliotecile utilizate, precum și modul în care acestea au fost integrate pentru a răspunde cerințelor funcționale și tehnice ale proiectului.

Se evidențiază etapele esențiale ale implementării, alături de funcționalitățile principale dezvoltate și provocările întâmpinate în procesul de dezvoltare. De asemenea, este realizată o comparație între soluția propusă și metoda tradițională de creare a orarului, pentru a sublinia avantajele aduse de digitalizarea acestui proces.

În încheiere, este inclus un ghid de utilizare al aplicației, care ilustrează modul concret în care utilizatorul poate interacționa cu platforma pentru a introduce date, a configura reguli și a genera orare complete, valide și personalizate.

## 4.1 Limbajele de programare implicate în dezvoltarea aplicației

În dezvoltarea aplicației pentru generarea automată a orarului universitar 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.

Limbajele selectate au permis realizarea unei aplicații moderne, interactive și ușor de extins, oferind suport atât pentru gestionarea datelor, cât și pentru integrarea cu servicii de inteligență artificială. Fiecare limbaj a avut un rol bine definit în cadrul sistemului, contribuind la funcționarea coerentă și eficientă a întregii aplicații.

### 4.1.1 Java Script

JavaScript este un limbaj de programare interpretat, dinamic și orientat pe evenimente, utilizat pe scară largă pentru dezvoltarea aplicațiilor web. În cadrul acestei aplicații, JavaScript a fost ales pentru realizarea părții de frontend, datorită capacității sale de a crea interfețe interactive, rapide și intuitive.

Aplicația utilizează biblioteca React.js, bazată pe JavaScript, pentru a construi componente reutilizabile care răspund în timp real la modificările de stare. Prin intermediul acestui limbaj, utilizatorul poate introduce date în formulare, vizualiza orarul generat în mod dinamic și interacționa cu sistemul într-un mod fluent. Tot JavaScript facilitează comunicarea asincronă cu backend-ul scris în Python, prin trimiterea cererilor HTTP și preluarea rezultatelor generate de inteligența artificială.

De asemenea, JavaScript a permis integrarea unor mecanisme moderne de feedback vizual, precum notificările de tip toast sau alertele personalizate, care îmbunătățesc experiența utilizatorului. Funcționalitățile de validare automată a orarului, export în PDF sau Excel, precum și navigarea între pagini sunt implementate tot cu ajutorul acestui limbaj. Fragmentul de cod prezentat în Figura 15 exemplifică utilizarea hook-ului useEffect din React pentru monitorizarea stării aplicației și actualizarea raportului de validare în funcție de orarul generat.

#### 

#### Fig. 15 Exemplu de cod JavaScript folosit în aplicație

### 4.1.2 Python

Python este un limbaj de programare de nivel înalt, interpretat și cunoscut pentru sintaxa sa clară și expresivă, ceea ce îl face ideal pentru dezvoltarea rapidă a aplicațiilor. Datorită flexibilității și ecosistemului său bogat, Python este utilizat extensiv în domenii precum dezvoltare web, inteligență artificială și analiză de date [10].

În cadrul acestei aplicații, Python a fost folosit pentru realizarea părții de backend, alături de framework-ul Flask. Acesta gestionează logica principală a sistemului, comunicarea cu baza de date și integrarea cu serviciul AI oferit de OpenAI GPT-4. Backend-ul preia datele transmise din interfața React, le prelucrează, generează orarul în format JSON și îl validează înainte de a-l trimite înapoi către utilizator.

Python a fost ales pentru acest proiect datorită ușurinței de utilizare, compatibilității excelente cu serviciile AI și suportului vast pentru biblioteci dedicate web-ului și bazelor de date. În Figura 16 este ilustrat un fragment de cod din modulul orar\_generator.py, în care sunt organizate grupele și subgrupele în funcție de nivelul și anul de studiu. Se utilizează structuri de tip defaultdict pentru stocarea eficientă a datelor, iar logica aplicată pregătește informațiile necesare algoritmului de generare, respectând regulile prestabilite.

#### 

#### Fig. 16 Exemplu de cod Python folosit în aplicație

### 4.1.3 HTML5

HTML5 reprezintă versiunea modernă a limbajului de marcare HTML și este esențial în structurarea conținutului vizual al aplicațiilor web. Deși interfața aplicației este construită în React.js, toate componentele JSX definite în cod sunt transpuse automat în elemente HTML5 valide, care sunt interpretate de browser.

Astfel, aplicația redă o structură semantică coerentă, bazată pe elemente HTML5 precum butoane, formulare, tabele și alerte, toate generate din componentele React. Acest lucru asigură o compatibilitate ridicată cu toate browserele moderne, precum și o accesibilitate crescută prin utilizarea atributelor standard, precum role, aria-hidden sau disabled.

Fragmentul de cod prezentat în Figura 17 ilustrează un exemplu concret în care este definit un buton HTML5, folosit pentru inițierea procesului de generare a orarului cu ajutorul AI. Atunci când procesul este în desfășurare, butonul afișează o animație de tip loader; în caz contrar, este prezentat un text sugestiv pentru utilizator. Chiar dacă este scris în JSX, codul este interpretat în HTML5, ceea ce permite o integrare nativă cu CSS și JavaScript pentru un comportament interactiv și modern.

#### 

#### Fig. 17 Exemplu de cod HTML5 folosit în aplicație

### 4.1.4 CSS3

CSS3 (Cascading Style Sheets, versiunea 3) este limbajul dedicat stilizării aplicațiilor web, oferind suport pentru personalizarea aspectului vizual, a layout-ului și a responsivității interfeței. În cadrul acestei aplicații, CSS3 este utilizat pentru a crea o experiență vizuală coerentă, intuitivă și adaptabilă diferitelor dimensiuni de ecran[12].

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 18).

#### A screenshot of a computer AI-generated content may be incorrect.

#### Fig. 18 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 19).

#### A close-up of a number AI-generated content may be incorrect.

#### Fig. 19 Exemplu de cod CSS3 – prin stiluri inline - folosit în aplicație

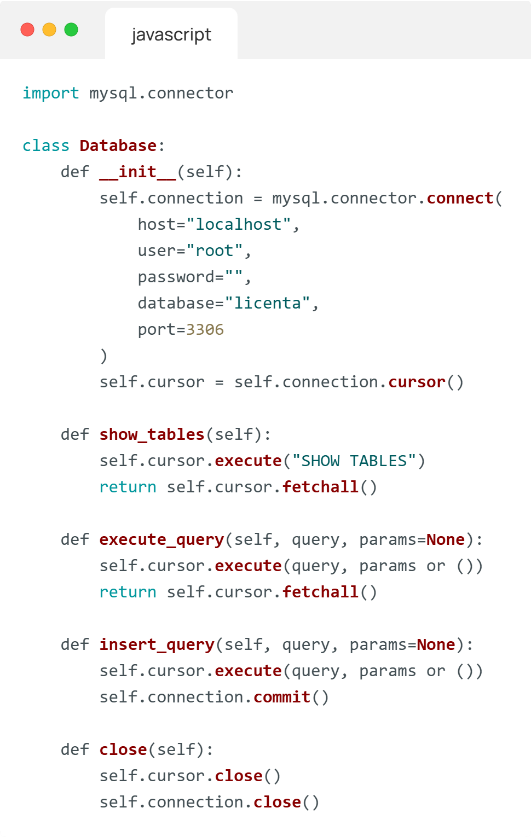
Fișierul App.css completează această abordare prin stiluri personalizate aplicate global – precum animații, efecte de tranziție sau personalizarea fonturilor aplicației. Combinarea acestor metode a permis o stilizare flexibilă și modernă, adaptată nevoilor vizuale ale utilizatorului.

### 4.1.5 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.

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.

Figura 21 prezintă clasa Database, care centralizează metodele de conectare și interogare, fiind utilizată în întregul sistem pentru lucrul cu datele.



#### Fig. 21 Exemplu de cod SQL folosit în aplicație

## 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 construirea interfețelor de utilizator moderne și interactive. În cadrul acestei aplicații, React este responsabil pentru partea de frontend, oferind o structură componentizată, ușor de întreținut și extins [14].

Cu ajutorul unor mecanisme precum useState și useEffect, aplicația gestionează eficient starea internă și actualizează automat interfața pe măsură ce datele sunt modificate. Aceste funcționalități permit încărcarea dinamică a profesorilor, sălilor și grupelor, precum și afișarea în timp real a orarului generat.

Figura 21 prezintă un exemplu de custom hook (useOrarGenerator) utilizat pentru preluarea datelor din backend și stocarea acestora în variabile de stare. Acest fragment este reutilizat în mai multe componente ale aplicației pentru a menține datele sincronizate și a facilita interacțiunea dintre utilizator și interfață.

A screen shot of a computer program

AI-generated content may be incorrect.

#### 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 [15]. 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 prelucrează 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.



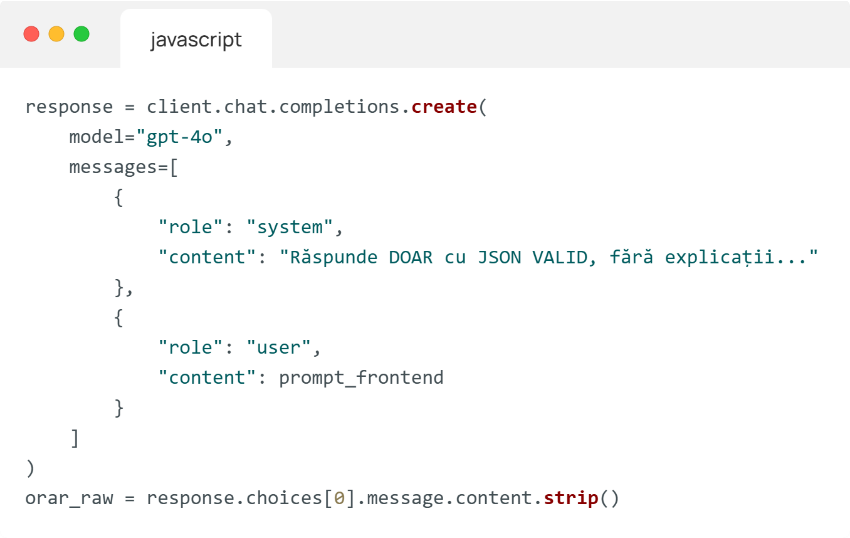
#### Fig. 22 Exemplu de cod Flask folosit în aplicație

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

GPT-4 este un model avansat de inteligență artificială generativă, dezvoltat de OpenAI, bazat pe arhitectura transformer. În cadrul acestei aplicații, este utilizată versiunea GPT-4o, care, deși este multimodală, este folosită exclusiv pentru generarea de text – mai precis, a orarelor universitare în format JSON valid [16].

Integrarea GPT-4 se realizează direct în backend-ul Flask, prin intermediul pachetului oficial openai și a unei chei API securizate. Modelul este apelat prin metoda chat.completions.create(), unde se transmite un prompt generat pe baza datelor introduse de utilizator. Răspunsul este configurat să fie exclusiv în format JSON, fără explicații suplimentare, pentru a putea fi procesat direct în interfață.

Figura 23 prezintă un exemplu de cod în care se face această integrare. Răspunsul generat este preluat și convertit într-un orar afișabil, complet și coerent, eliminând nevoia unui algoritm clasic de planificare. Astfel, soluția propusă valorifică procesarea limbajului natural (NLP) pentru a genera automat rezultate personalizate, pe baza regulilor definite de utilizator.



#### Fig. 23 Exemplu de cod de integrare AI folosit în aplicație

### 4.2.4 Bootstrap 5

Bootstrap 5 este un framework CSS open-source dedicat dezvoltării rapide a interfețelor web moderne și responsive. Acesta oferă un set extins de componente predefinite, precum butoane, formulare, meniuri și sisteme de layout flexibile, toate bazate pe o grilă de tip rânduri și coloane. Spre deosebire de versiunile anterioare, Bootstrap 5 a eliminat dependența de jQuery, devenind astfel mai eficient și mai compatibil cu framework-uri moderne precum React [17].

În cadrul aplicației, Bootstrap 5 a fost utilizat pentru construirea unei interfețe responsive, compatibile cu diferite rezoluții și dispozitive. Stilizarea componentelor precum butoane, carduri, formulare sau bare de navigație s-a realizat prin clase Bootstrap standard (btn, card, container, alert, navbar etc.), ceea ce a permis o dezvoltare rapidă și un design coerent. Structura paginilor s-a bazat pe sistemul de grid integrat, ușor de adaptat în JSX.

Figura 24 ilustrează un exemplu concret de utilizare a Bootstrap în cadrul componentei Home.jsx, unde stilizarea este realizată exclusiv prin clase oferite de acest framework, fără a fi necesară scrierea de CSS suplimentar.

A screenshot of a computer code

AI-generated content may be incorrect.

#### Fig. 24 Exemplu de cod Bootstrap folosit în aplicație

### 4.2.5 MySQL

MySQL este un sistem de gestiune a bazelor de date relaționale, open-source, recunoscut pentru stabilitate, performanță și ușurința integrării în aplicații web moderne [18]. În cadrul aplicației dezvoltate, MySQL joacă un rol central, asigurând stocarea persistentă a tuturor datelor esențiale – de la informațiile despre profesori, săli și grupe, până la regulile introduse de utilizator și orarele generate.

Baza de date este structurată astfel încât să permită manipularea eficientă a datelor și relaționarea clară între entități. Prin intermediul pachetului mysql.connector, backend-ul Flask comunică direct cu baza MySQL, permițând operații de inserare, actualizare, interogare și ștergere a datelor.

Figura 25 prezintă un exemplu de cod în care este definită o rută pentru adăugarea grupelor în baza de date. Informațiile sunt primite sub formă de obiect JSON din frontend, apoi inserate în tabela grupe printr-o interogare SQL executată din backend. Acest mod de operare asigură un flux de date coerent și securizat între componentele aplicației.



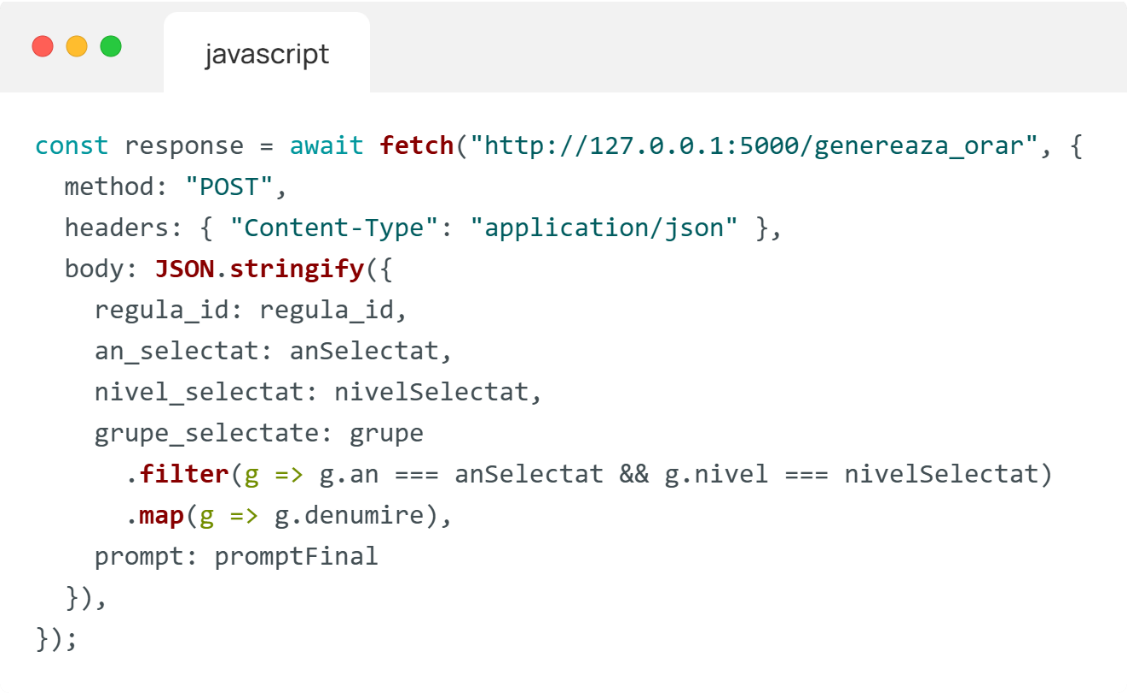
#### Fig. 25 Exemplu de cod Flak și MySQL folosit în aplicație

### 4.2.6 Fetch API

Fetch API este o interfață JavaScript modernă care permite efectuarea de cereri HTTP către server, într-un mod asincron și elegant, fără a reîncărca pagina. Aceasta înlocuiește vechiul mecanism XMLHttpRequest, oferind un mod mai intuitiv de a comunica cu backend-ul în aplicațiile web [18].

Un avantaj major al metodei fetch() este faptul că returnează un obiect Promise, permițând astfel gestionarea facilă a răspunsurilor, erorilor și a logicii asincrone într-un mod clar și eficient.

În cadrul acestei aplicații, Fetch API este utilizat pentru a trimite cereri POST către backend-ul Flask. Datele introduse de utilizator – precum nivelul, anul de studiu, grupele selectate și promptul AI – sunt trimise în format JSON către endpoint-ul /genereaza\_orar. Figura 26 prezintă un exemplu concret de implementare, în care datele sunt filtrate și mapate înainte de a fi transmise serverului, pentru a asigura generarea corectă a orarului pe baza criteriilor selectate.



#### Fig. 26 Exemplu de cod Fetch folosit în aplicație

### 4.2.7 SweetAlert2

SweetAlert2 este o bibliotecă JavaScript modernă, utilizată pentru afișarea mesajelor modale interactive, într-un format elegant și personalizabil [19]. Aceasta înlocuiește alertele implicite ale browserului (alert(), confirm(), prompt()) cu ferestre responsive și ușor de integrat, contribuind la o experiență vizuală mai plăcută pentru utilizator.

În cadrul aplicației, SweetAlert2 este folosit pentru confirmarea acțiunilor importante, afișarea mesajelor de succes sau eroare, precum și pentru interacțiuni personalizate, cum ar fi editarea denumirii unui orar generat. Biblioteca permite integrarea directă cu funcții asincrone și gestionarea simplă a fluxului aplicației prin Promises.

Figura 27 prezintă un exemplu concret de utilizare, în care este afișată o alertă interactivă pentru redenumirea unui orar, oferind utilizatorului o interfață prietenoasă și clară pentru a introduce o nouă valoare.



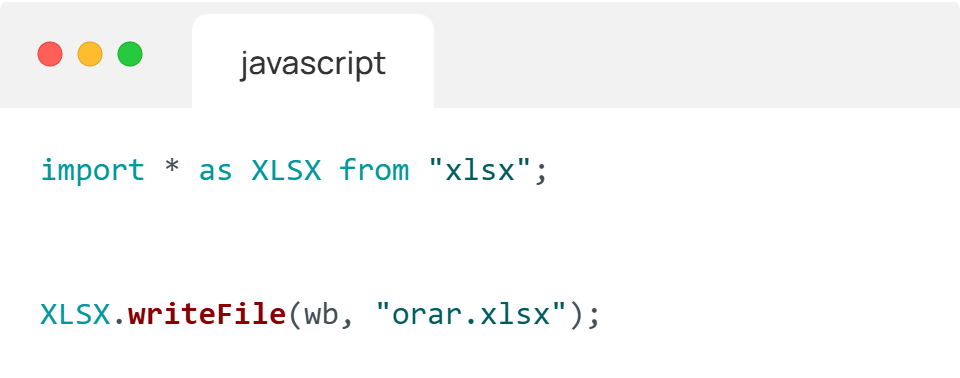
#### Fig. 27 Exemplu de cod SweetAlert folosit în aplicație

### 4.2.8. SheetJS (xlsx)

SheetJS, cunoscută și sub denumirea de xlsx, este o bibliotecă JavaScript open-source care permite citirea, scrierea și manipularea fișierelor Excel (format .xlsx) direct în browser sau în aplicații Node.js [20]. Este extrem de utilă pentru exportul sau importul de date structurate în format tabelar, fără a depinde de Microsoft Excel instalat local.

Exemplu concret:

* Creează câte un sheet Excel pentru fiecare grupă din orarul generat.
* Datele sunt organizate pe zile, intervale orare, activitate, profesor și sală.
* Exportul se face în fișierul orar.xlsx, folosind metoda XLSX.writeFile() (Figura28):



#### Fig. 28 Exemplu de cod SheetJS folosit în aplicație

### 4.2.9 html2pdf.js

html2pdf.js este o bibliotecă JavaScript care permite convertirea unui element HTML într-un fișier PDF, direct din browser, fără a necesita software suplimentar [21]. Este utilă pentru aplicațiile web care trebuie să exporte conținut vizual în format tipărit sau arhivabil.

Utilizare concretă în proiectul curent:

* Convertește conținutul tabelului orar (elementul #orar-afisat) într-un fișier PDF.
* Păstrează stilurile CSS aplicate (culori, fonturi, dimensiuni).
* PDF-ul este generat în format A4, orientare landscape, și salvat local ca orar.pdf.

Exemplu de cod (Figura 29):

A screenshot of a computer

AI-generated content may be incorrect.

#### Fig. 29 Exemplu de cod html2pdf.js folosit în aplicație

## 4.3. Implementarea și funcționalitatea aplicației

Aplicația propusă are ca scop generarea automată a orarelor universitare 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 universitar are o arhitectură modulară de tip client-server, având următoarele componente:

* + Frontend – React.js: interfața web interactivă;
  + Backend – Flask (Python): API REST + integrare AI;
  + Bază de date – MySQL: stocarea datelor academice și a orarelor generate;
  + AI Engine – OpenAI GPT-4 (sau alternativ model local): generarea orarului pe baza unui prompt detaliat.

### 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](https://reactjs.org/) – o bibliotecă JavaScript pentru crearea interfețelor web moderne, bazată pe componente reutilizabile și un model reactiv.
* **Bundler/Toolchain**: [Vite](https://vitejs.dev/) – ales în locul clasicului Webpack pentru viteze superioare de dezvoltare și build, inclusiv:
  + HMR (Hot Module Replacement) extrem de rapid;
  + Builduri optimizate cu Tree Shaking;
  + Suport nativ pentru ESModules.
* **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 etc.
* **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](https://flask-cors.readthedocs.io/) 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**:
  + Poate fi folosit **offline**, fără conexiune la internet.
  + 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 Implemetarea funcționalităților principale

Aplicația a fost dezvoltată modular, fiecare funcționalitate fiind structurată pe trei niveluri:

* interfața utilizatorului (Frontend),
* logica de procesare și expunerea API-urilor (Backend – Flask),
* persistența datelor (Bază de date – MySQL).

Fiecare componentă permite introducerea, vizualizarea, modificarea și ștergerea datelor relevante pentru generarea unui orar academic complet.

Pentru o imagine de ansamblu asupra implementării, în tabelul de mai jos sunt sintetizate funcționalitățile principale ale aplicației, împreună cu corespondențele lor la nivel de interfață, backend și structură de bază de date:

##### Tabelul 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)" |

Deși la nivel conceptual se poate propune o idee pentru a dezvolta o aplicație mobilă, nu este atât de ușor precum pare deoarece este nevoie mai întâi de o analiză complexă, care să cuprindă un studiu al aplicațiilor deja existente pe piată, un studiu al tehnologiilor care urmează să fie incluse, sau să se cunoască cerințele minime de sistem pentru ca aplicația să poată rula optim. Toate aceste aspecte au fost luate în considerare și în cazul aplicației Personal Doctor, unde a fost mai întâi nevoie de un studiu riguros a mai multor aplicații similare, în care s-au evidențiat aspectele pozitive și cele negative care aveau să fie luate în considerare pentru proiectarea viitoarei aplicații.

## 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 tabelelor 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.

1. 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.
2. 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.
3. Timpul de răspuns al generatorului AI: În unele cazuri, modelul GPT-4 avea un timp de răspuns prea mare sau returna răspunsuri incomplete. Pentru a preveni blocarea aplicației, s-a implementat un sistem de afișare cu spinner și notificare de eroare la timeout.
4. Exportul PDF/Excel și păstrarea formatării: 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.
5. Modularizarea codului și separarea logicii de interfață: Codul componentelor React a devenit greu de întreținut pe măsură ce funcționalitățile au crescut. A fost necesară o refactorizare generală, extrăgând logica în fișiere separate (ex: useProfesoriLogic.js, useSaliLogic.js) pentru a îmbunătăți claritatea și reutilizarea.

## 4.5 Comparația cu metoda clasică

Pentru a evidenția eficiența și avantajele aduse de utilizarea inteligenței artificiale în generarea orarului universitar, am realizat o comparație între două abordări:

|  |  |  |
| --- | --- | --- |
| Criteriu de comparare | Algoritm Clasic (Hardcoded) | Algoritm AI (OpenAI GPT / model local) |
| Flexibilitate | Limitată – reguli hardcodate, modificările necesită rescrierea codului | Foarte ridicată – se adaptează pe baza unui prompt cu reguli în format JSON |
| Personalizare | Doar prin modificare în cod (ex: ore master, pauze, max cursuri) | Personalizabil din interfață (formular HTML sau reguli JSON) |
| Timp de dezvoltare | Ridicat – necesită tratarea manuală a fiecărei reguli | Redus – promptul AI gestionează logică complexă cu mai puține linii de cod |
| Acoperire logică | Hardcodate pe structură fixă: cursuri → an, seminare → grupă, lab/proiect → subgrupă | Se respectă automat structura: an, grupă, subgrupă + se pot adăuga reguli noi |
| Validare orar | Manuală, prin metode Python (valideaza\_cursuri\_sincronizate) | Automatizată cu funcții de validare + feedback textual și vizual în browser |
| Control asupra profesorilor/sălilor | Selectați aleator + validați în cod | Selectați în mod inteligent din baza de date, pe baza disponibilității |
| Reutilizare | Limitată – algoritmul e specific unei structuri (licență/master) | Generalizabil – același motor poate genera orare pentru orice structură |
| Rezultate | Coerente, dar pot apărea suprapuneri dacă logica nu acoperă toate cazurile | În general coerente și complete, cu auto-corectare și ajustare din prompt |
| Interactivitate | Lipsită – utilizatorul final nu poate modifica reguli fără programare | Ridicată – utilizatorul poate modifica regulile în interfață |

Metoda clasică (AlgoritmClasic)

Această variantă implică extragerea grupelor din baza de date și filtrarea lor pe baza denumirii (ex: LI1a, LI2b). Profesorii și disciplinele sunt atribuite aleator, iar activitățile sunt plasate în orar în funcție de disponibilitatea sălilor și regulile codificate manual. Se aplică constrângeri fixe, precum „maximum 4 ore pe zi” sau „pauză Miercuri 14–16”, prin funcții Python dedicate.

Metoda AI (OrarGenerator + GPT)

Prin această abordare, toate datele – grupe, subgrupe, profesori, săli – sunt preluate automat din baza de date, iar regulile sunt descrise textual într-un prompt GPT. Modelul AI generează un orar complet în format JSON, care este apoi validat automat cu funcții proprii (ex: valideaza\_orar). Utilizatorul poate modifica regulile direct din interfața aplicației, fără a fi necesară intervenția în cod, ceea ce sporește semnificativ gradul de accesibilitate și control.

## 4.6 Ghidul utilizării aplicației

Această secțiune prezintă modul concret de utilizare a aplicației web pentru generarea automată a orarelor universitare, 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 universitar cu ajutorul Inteligenței Artificiale.

#### A screenshot of a computer AI-generated content may be incorrect.

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.

#### A screenshot of a computer AI-generated content may be incorrect.

#### 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.

#### A screenshot of a computer AI-generated content may be incorrect.

#### 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.

#### A screenshot of a computer AI-generated content may be incorrect.

#### Funcționalitățile aplicației

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

#### A screenshot of a computer AI-generated content may be incorrect.

#### 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.

A screenshot of a computer

AI-generated content may be incorrect.

*Dashboard-ul aplicației*

Utilizatorul poate selecta una dintre cele 4 secțiuni principale:

* **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.

A screenshot of a computer

AI-generated content may be incorrect.

*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.

A screenshot of a computer

AI-generated content may be incorrect.

*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.

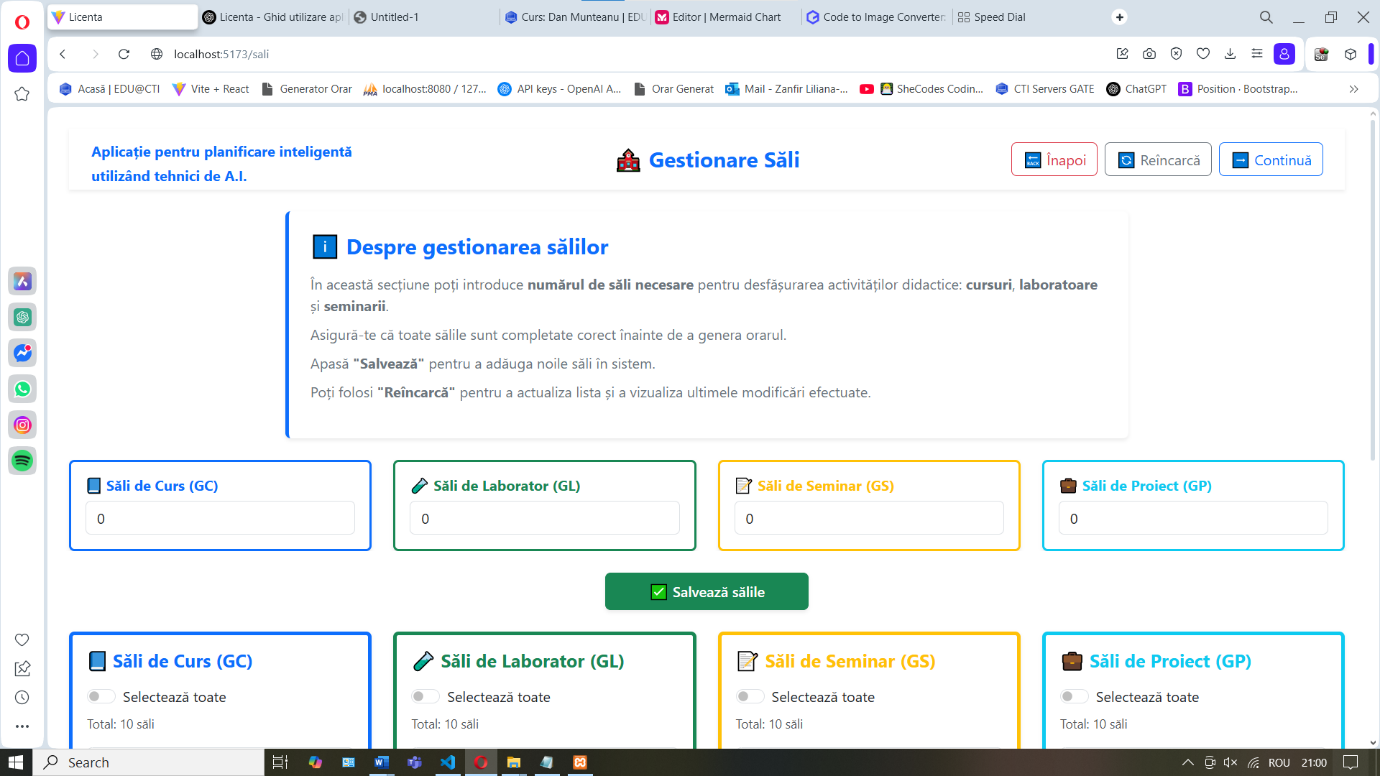
Toate grupele generate vor fi folosite ulterior la asocierea activităților în orar.

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. 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



*Interfața pentru gestionarea sălilor*

🔹 Funcționalități:

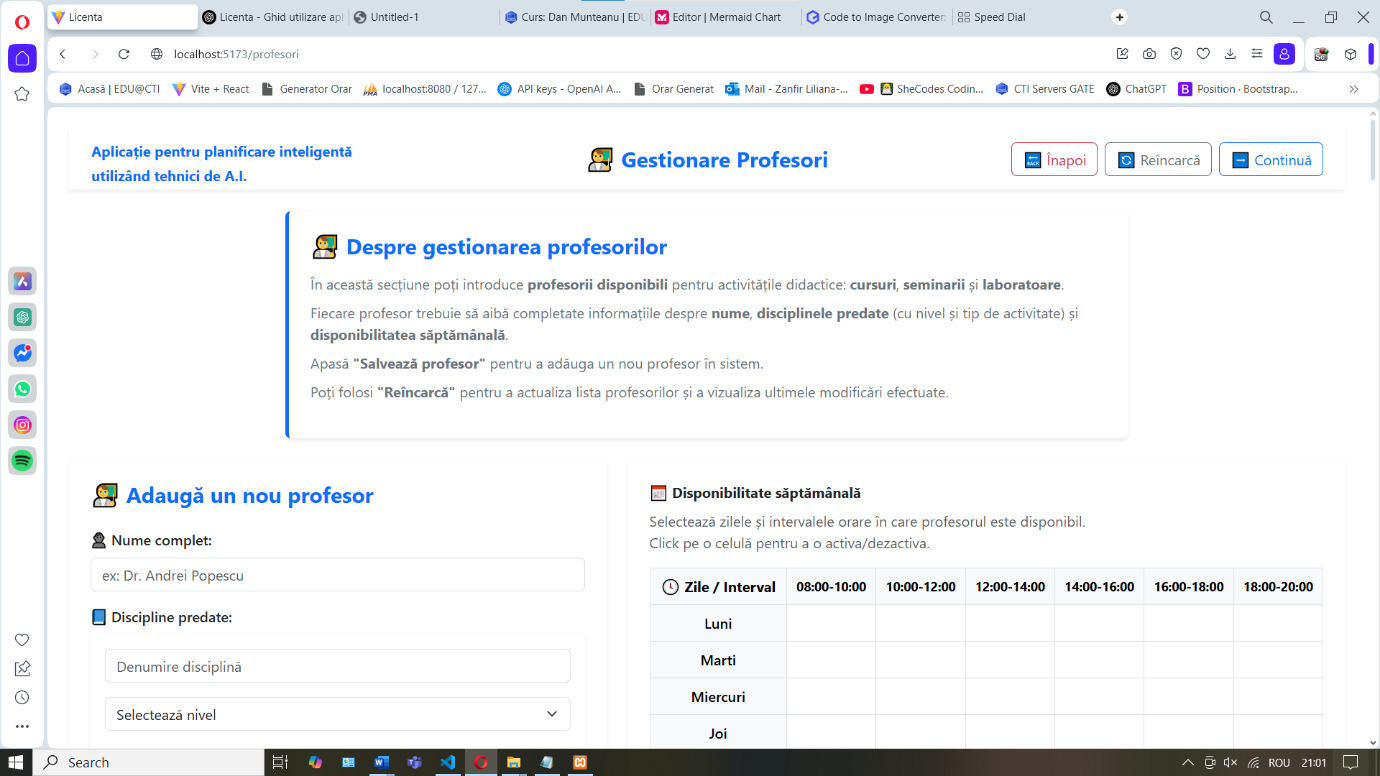
* Introducerea numărului de săli pentru fiecare tip
* Generarea automată a codurilor (ex: GC01, GL02)
* Vizualizarea sălilor generate, grupate pe categorii
* Selecție multiplă sau individuală
* Ștergerea sălilor selectate
* Reîncărcarea listei din baza de date

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

Pasul 5

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.



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

Pasul 6: Setarea regulilor de generare

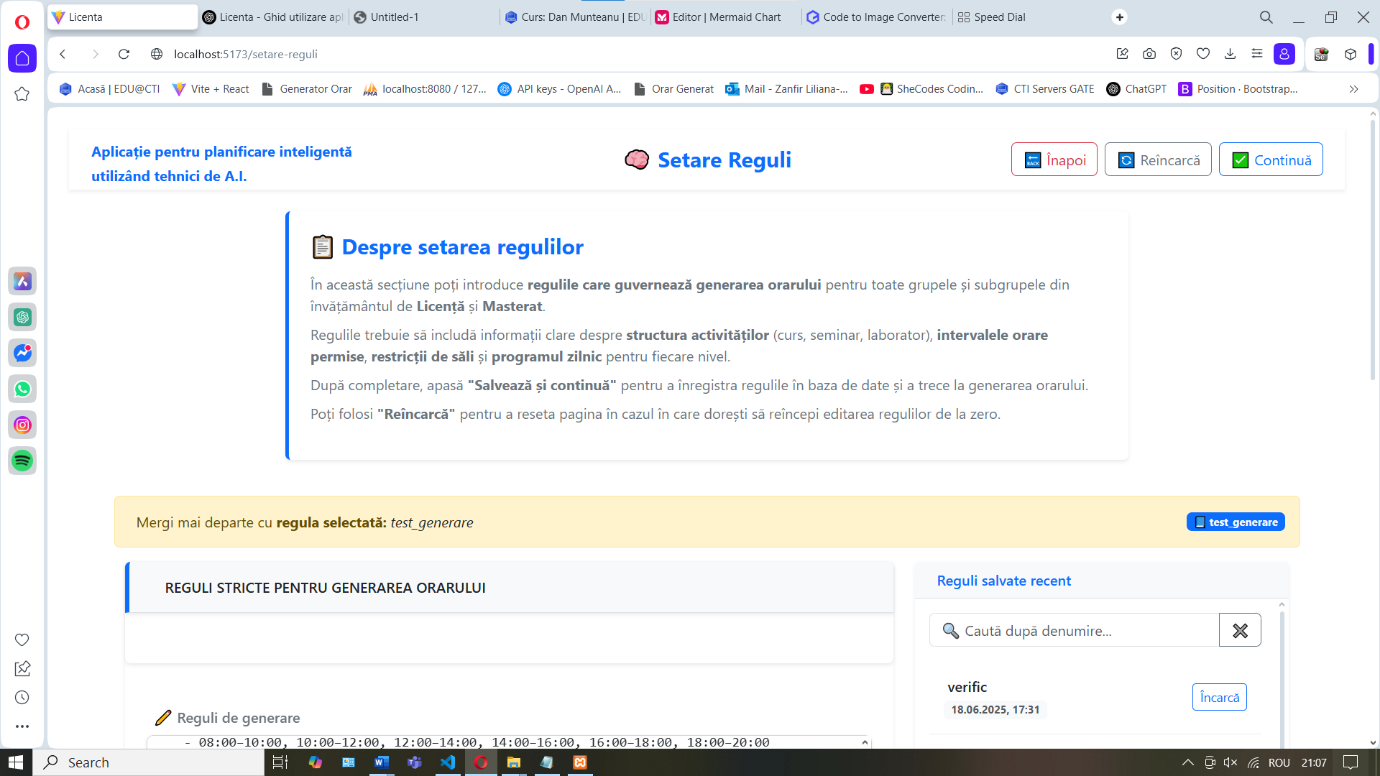
În această secțiune, utilizatorul definește regulile care guvernează generarea orarului pentru toate grupele și subgrupele din învățământul de Licență și Master.

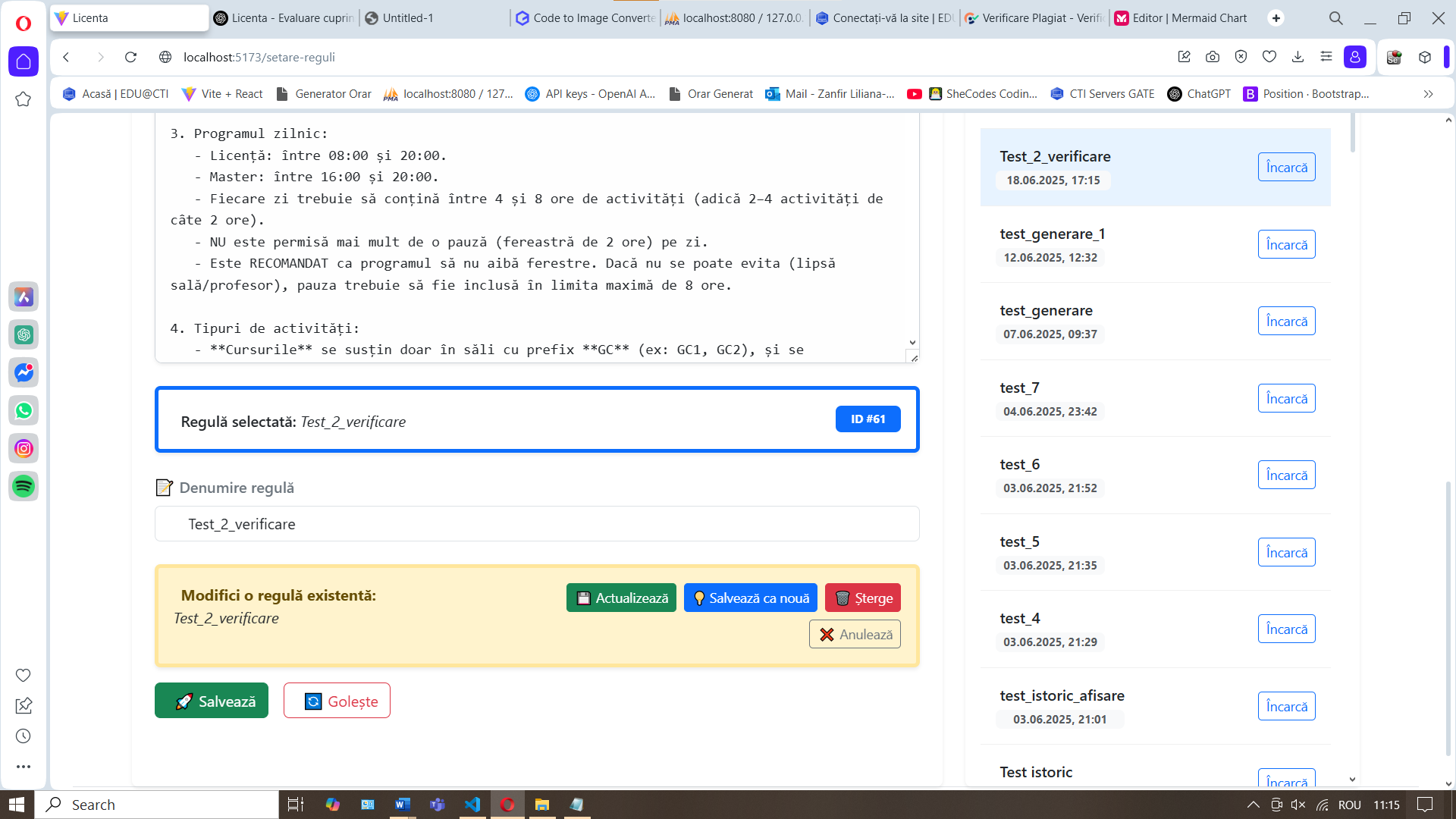
Ce trebuie completat:

* Structura activităților: se stabilește cum sunt distribuite cursurile (pe an), seminarele (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:

* Denumește 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.





Pasul 7

Pasul 7: Generarea orarului

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

Ce ai de făcut:

1. Selectează regula de generare dorită.
2. Alege nivelul de studiu (Licență sau Master).
3. Alege anul de studiu pentru care vrei să generezi orarul.
4. Apasă butonul verde „Generează orar cu AI” pentru o planificare automată folosind modelul inteligent.
   * 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.

A screenshot of a computer

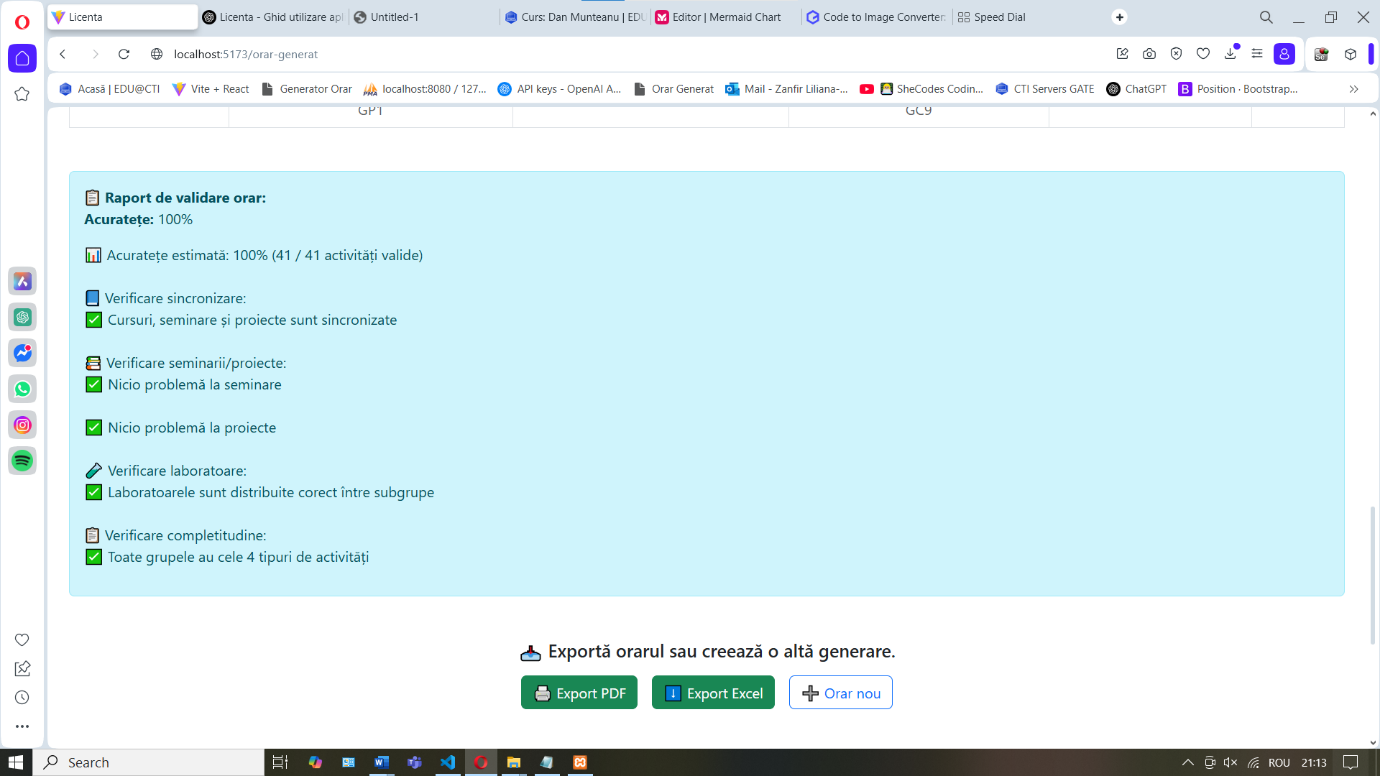
AI-generated content may be incorrect.

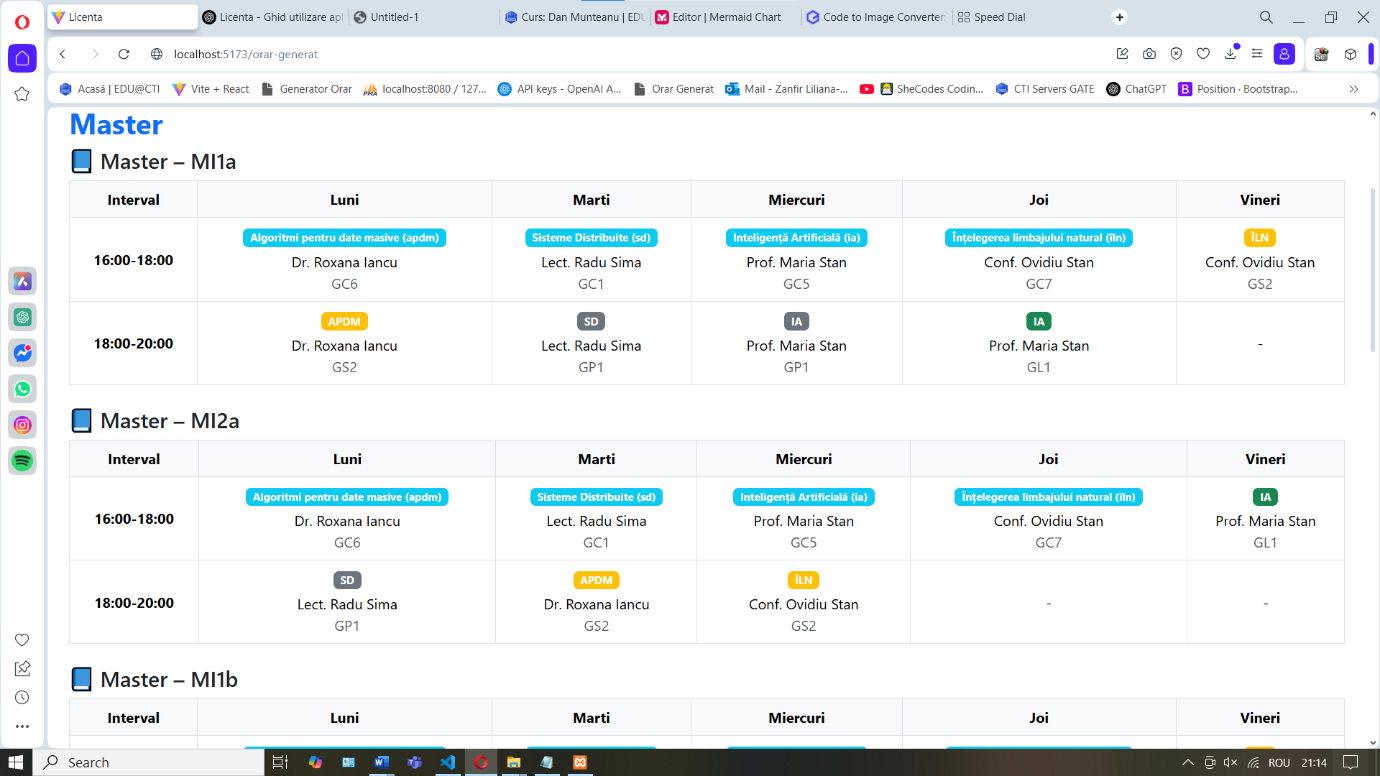
Afișarea orarului generat și validarea sa

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):
   * ✅ Acuratețea generării: procent estimat (ex: 100%)
   * ✅ Verificări sincronizare:
     + Cursuri, seminare, proiecte – verificate să nu se suprapună.
     + Laboratoare – distribuite corect pe subgrupe.
     + Completitudine – toate grupele au activitățile aferente.
2. Orarul afișat vizual:
   * 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)
   * Include:
     + Denumirea completă a disciplinei + acronim (ex: *Algoritmi pentru date masive (apdm)*)
     + Numele cadrului didactic (ex: *Dr. Roxana Iancu*)
     + Sala în care are loc activitatea (ex: *GC6*)
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.
* Export Excel – generează un fișier tabelar cu toate activitățile detaliate: ziua, intervalul, disciplina, tipul, profesorul și sala.

📌 *Aceste funcții se află în partea de jos a paginii de orar generat.*

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

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 universitare.

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).