PROIECT INDIVIDUAL

**Curs:** Managementul Dezvoltării Softului

**Nume student:** Pînzaru Laurențiu

**Tema proiectului:** Sistem de gestionare a studenților și profesorilor – Notificări, orare, rezultate academice.

**Tehnologii utilizate:** Front-end (React), Back-end (Node.js), Bază de Date (MySQL), Git, CI/CD Tools (GitHub Actions).

**Instrumente de management:** YouTrack, GitHub.

# Scopul proiectului

## Descrierea obiectivelor proiectului:

Proiectul „Sistem de gestionare a studenților și profesorilor” are ca scop principal eficientizarea proceselor administrative și academice în cadrul instituțiilor de învățământ. Obiectivele cheie sunt:

* + **Centralizarea Informațiilor:** Crearea unei platforme unificate pentru gestionarea datelor studenților și profesorilor, incluzând orare, rezultate academice și notificări.
  + **Îmbunătățirea Comunicării:** Facilitarea unei comunicări rapide și eficiente între studenți, profesori și personalul administrativ prin intermediul unui sistem de notificări robust.
  + **Optimizarea Proceselor:** Automatizarea și simplificarea sarcinilor administrative legate de gestionarea orarelor și a rezultatelor academice, reducând efortul manual și erorile.

# Definirea cerințelor și specificațiilor proiectului (SRS)

## Descriere generală a aplicației

Sistemul de gestionare a studenților și profesorilor va fi o aplicație web accesibilă printr-un browser, concepută pentru a integra și centraliza diverse funcționalități academice și administrative. Acesta va funcționa ca o platformă independentă, dar va fi proiectat pentru a permite integrări viitoare cu alte sisteme existente în instituțiile de învățământ. Sistemul va oferi o interfață intuitivă și ușor de utilizat pentru studenți, profesori și personal administrativ, asigurând o experiență fluidă și eficientă.

## Funcționalități principale

Funcționalitățile principale ale sistemului, detaliate în SRS, pot fi formulate ca User Stories, incluzând, dar fără a se limita la:

### Gestionarea Utilizatorilor:

* + - **US1:** Ca student, vreau să mă pot autentifica în sistem folosind email și parolă pentru a accesa informațiile mele academice.
    - **US2:** Ca personal administrativ, vreau să pot înregistra noi utilizatori (studenți, profesori) în sistem pentru a le gestiona accesul.
    - **US3:** Ca utilizator, vreau să-mi pot gestiona profilul personal (ex: schimbare parolă, actualizare date de contact) pentru a menține informațiile mele.

### Gestionarea Orarelelor:

* + - **US4:** Ca student, vreau să pot vizualiza orarul meu personalizat pentru a ști când și unde am cursuri, seminarii și examene.
    - **US5:** Ca profesor, vreau să pot vizualiza orarul cursurilor pe care le predau pentru a-mi organiza activitatea didactică.
    - **US6:** Ca personal administrativ, vreau să pot crea și modifica orare pentru cursuri, seminarii și laboratoare pentru a asigura o planificare eficientă.

### Gestionarea Notificărilor:

* + - **US7:** Ca profesor, vreau să pot trimite notificări către studenții unui anumit curs pentru a comunica anunțuri importante.
    - **US8:** Ca student, vreau să pot vizualiza notificările primite în cadrul sistemului pentru a fi la curent cu informațiile relevante.
    - **US9:** Ca sistem, vreau să generez notificări automate pentru modificări de orar sau publicarea rezultatelor academice pentru a informa rapid utilizatorii.

### Gestionarea Rezultatelor Academice:

* + - **US10:** Ca profesor, vreau să pot introduce notele studenților pentru cursurile mele pentru a actualiza situația lor academică.
    - **US11:** Ca student, vreau să pot vizualiza istoricul complet al notelor mele pentru a-mi monitoriza progresul academic.
    - **US12:** Ca profesor, vreau să pot înregistra prezențele studenților la cursuri și seminarii pentru a ține evidența participării.

### Gestionarea Cursurilor și Grupelor:

* + - **US13:** Ca personal administrativ, vreau să pot crea noi cursuri și să aloc profesori la acestea pentru a organiza oferta educațională.
    - **US14:** Ca personal administrativ, vreau să pot înscrie studenți la cursuri și să-i organizez în grupe de studiu pentru o gestionare eficientă.

# Organizarea proiectului

## Structura echipei

Deși este un proiect individual, în contextul unei echipe de dezvoltare, rolurile ar fi următoarele:

* + **Product Owner:** Responsabil pentru definirea și prioritizarea Product Backlog-ului, maximizarea valorii produsului și reprezentarea nevoilor părților interesate (ex: conducerea instituției, studenți, profesori).
  + **Scrum Master:** Facilitează procesul Scrum, elimină obstacolele, asigură respectarea principiilor Agile și ajută echipa să devină auto-organizată și eficientă.
  + **Echipa de Dezvoltare:** Responsabilă pentru implementarea funcționalităților, testarea, integrarea și livrarea incrementelor de produs. Aceasta ar include:
    - **Dezvoltatori Frontend:** Responsabili pentru interfața utilizator și experiența utilizatorului (UI/UX).
    - **Dezvoltatori Backend:** Responsabili pentru logica de afaceri, API- uri și interacțiunea cu baza de date.
    - **Dezvoltatori Fullstack:** Acoperă atât partea de interfață, cât și cea de backend, asigurând integrarea completă a fluxurilor de date între UI și server, oferind flexibilitate și viteză de livrare.
    - **Ingineri QA (Quality Assurance):** Responsabili pentru planificarea și execuția testelor (unitare, de integrare, funcționale, de performanță, de securitate) și asigurarea calității software-ului.
    - **DevOps Engineer:** Responsabil pentru implementarea și mentenanța pipeline-ului CI/CD, automatizarea proceselor de construire, testare și implementare.

## Metodologia aleasă

Metodologia aleasă pentru acest proiect este **Agile Scrum**.

## Justificarea alegerii metodologiei

Metodologia Agile Scrum a fost aleasă datorită următoarelor avantaje, care se aliniază perfect cu cerințele și natura acestui proiect:

* + **Flexibilitate și Adaptabilitate:** Permite adaptarea rapidă la cerințele în schimbare și la feedback-ul primit, esențial într-un proiect educațional unde nevoile pot evolua.
  + **Livrare Iterativă și Incrementală:** Proiectul a fost dezvoltat în mod iterativ, prin Sprint-uri săptămânale, fiecare livrând un set clar de funcționalități complete și testabile (ex: autentificare, notificări, orare).
  + **Transparență și Vizibilitate:** Toate aspectele proiectului (Product Backlog, Sprint Backlog, progres) sunt vizibile pentru toți membrii echipei și părțile interesate, facilitând colaborarea și luarea deciziilor.
  + **Calitate Îmbunătățită:** Prin integrarea continuă și testarea frecventă, se asigură o calitate superioară a produsului final.
  + **Gestionarea Riscurilor:** Riscurile sunt identificate și abordate în mod proactiv în fiecare Sprint, reducând impactul potențial al problemelor.

# Planificare și gestionare în YouTrack

Planificarea și gestionarea proiectului se vor realiza utilizând un instrument dedicat, precum **YouTrack**, pentru a asigura o urmărire eficientă a sarcinilor și a progresului.

## Crearea unui Product Backlog

Product Backlog-ul este o listă ordonată de cerințe, funcționalități, îmbunătățiri și corecții de erori. Acesta este gestionat de Product Owner și este populat inițial pe baza cerințelor din SRS. Fiecare element din backlog este o User Story cu criterii de acceptare clare și o estimare în Story Points.

**Exemple de elemente din Product Backlog:**

|  |  |  |
| --- | --- | --- |
| **User Story ID** | **Descriere** | **Estimare Story Points** |
| US1 | Ca student, vreau să mă pot autentifica în sistem. | 3 |
| US2 | Ca profesor, vreau să pot introduce notele studenților. | 8 |
| US3 | Ca administrator, vreau să pot crea orare pentru cursuri. | 5 |
| US4 | Ca student, vreau să pot vizualiza orarul meu personalizat. | 3 |
| US5 | Ca profesor, vreau să pot trimite notificări studenților. | 5 |

## Organizarea taskurilor pe Sprinturi

Proiectul va fi împărțit în Sprint-uri cu o durată fixă (de exemplu, 2 săptămâni). La începutul fiecărui Sprint, echipa selectează un set de User Stories din Product Backlog pentru a le finaliza. Acestea formează Sprint Backlog-ul.

### Exemplu de planificare a Sprint-urilor (pentru un proiect de 30 de ore, Sprint-uri de 1 săptămână):

**• Sprint 1 (7–14 iunie 2025)**    • Obiectiv: Punerea bazelor aplicației – sistem de autentificare, înregistrare și gestionare profil, plus structură inițială pentru modulul de orar.  
    • Sprint Backlog:  
        – SCSP‑1: Autentificare student (formular UI + API JWT)  
        – SCSP‑2: Înregistrare utilizatori (formular UI + API)  
        – SCSP‑3: Gestionare profil utilizator (UI + API actualizare)  
        – SCSP‑4: Vizualizare orar basic student (UI + DB)

**• Sprint 2 (15–22 iunie 2025)**    • Obiectiv: Extinderea modulului de orare și introducerea sistemului de notificări manuale.  
    • Sprint Backlog:  
        – SCSP‑5: Vizualizare orar profesor (UI + API)  
        – SCSP‑6: Creare & editare orar administrator (UI + validare backend)  
        – SCSP‑7: Trimitere notificări (UI + API)  
        – SCSP‑8: Vizualizare notificări student (UI + API)

**• Sprint 3 (23–20 iulie 2025)**• Obiectiv: Automatizarea notificărilor și implementarea modulului de note.  
    • Sprint Backlog:  
        – SCSP‑9: Notificări automate la evenimente (backend trigger)  
        – SCSP‑10: Introducere note profesor (UI tabel editabil + API tranzacțional)

**• Sprint 4 (1–7 iulie 2025)**    • Obiectiv: Prezentarea și filtrarea rezultatelor – dashboard note și evidență prezență – și crearea cursurilor.  
    • Sprint Backlog:  
        – SCSP‑11: Vizualizare note student (dashboard React + API)  
        – SCSP‑12: Evidență prezență (UI tabel + API)  
        – SCSP‑13: Creare cursuri și alocare profesori (UI + API)

**• Sprint 5 (8–14 iulie 2025)**• Obiectiv: Gestionarea înscrierii studenților la cursuri/grupe și stabilirea procesului de livrare continuă (CI/CD + teste unitare).  
    • Sprint Backlog:  
        – SCSP‑14: Înscriere studenți la cursuri/grupe (UI drag‑and‑drop + API)  
        – Configurare GitHub Actions (CI/CD pipeline)  
        – Scriere teste unitare backend (Mocha/Chai)

**• Sprint 6 (15–21 iulie 2025)**    • Obiectiv: Import masiv de studenți și testare finală a întregului flux.  
    • Sprint Backlog:  
        – SCSP‑15: Import masiv studenți din CSV/Excel (backend batch + raportare)  
        – Testare manuală end‑to‑end (login, orar, notificări, note, profil)

## Captură de ecran cu Boardul Scrum

## Taskuri estimative

Estimarea efortului pentru fiecare task se realizează folosind Story Points, o unitate de măsură relativă care reflectă complexitatea, riscul și volumul de muncă. Echipa utilizează Planning Poker pentru a ajunge la un consens.

**Exemple de estimări Story Points:**

|  |  |  |
| --- | --- | --- |
| **User Story ID** | **Descriere** | **Estimare Story Points** |
| US1 | Ca student, vreau să mă pot autentifica în sistem. | 3 |
| US2 | Ca profesor, vreau să pot introduce notele studenților. | 8 |
| US3 | Ca administrator, vreau să pot crea orare pentru cursuri. | 5 |
| US8 | Ca personal administrativ, vreau să pot importa în masă studenți. | 13 |

## Stabilirea duratei sprinturilor

Durata fiecărui Sprint este fixă și scurtă, de obicei între 1 și 4 săptămâni. Pentru acest proiect, având în vedere durata totală de 30 de ore, s-a optat pentru Sprint-uri de **1 săptămână** pentru a permite un feedback mai rapid și o adaptare continuă.

# Estimarea timpului

## Estimarea totală în Story Points

Pe baza estimărilor pentru fiecare **User Story**, totalul Product Backlog-ului ajunge la **≈93 Story Points** (SP). Monitorizând *velocity*-ul echipei (numărul mediu de SP finalizați într‑un Sprint), putem prognoza durata proiectului.

**Exemplu de calcul al duratei**

* **Velocity medie estimată:** ~16 Story Points/Sprint
* **Număr estimat de Sprint‑uri:** 93 SP ÷ 16 SP/Sprint ≈ 6 Sprint‑uri
* **Durată totală estimată:** 6 Sprint‑uri × 1 săptămână/Sprint = 6 săptămâni  
  *(aproximativ 45 de ore de lucru efectiv, incluzând și activități de management, testare și documentare)*

# Arhitectura proiectului

## Diagrama arhitecturii sistemului (Microservicii, MVC, Monolit etc.)

Sistemul va urma o arhitectură pe trei niveluri (3-tier architecture) pentru a asigura scalabilitate, performanță și separarea preocupărilor. Această arhitectură este compusă din:

* + **Nivelul de Prezentare (Frontend):** Interfața utilizator cu care interacționează direct utilizatorii.
  + **Nivelul de Logică (Backend):** Conține logica principală a aplicației și procesează cererile.
  + **Nivelul de Date (Bază de Date):** Stochează toate datele persistente ale sistemului.

## Tehnologii și frameworkuri utilizate

* + **Frontend:** React pentru o interfață responsivă și dinamică.
  + **Backend:** Node.js pentru logica de afaceri și API-uri RESTful.
  + **Bază de Date:** MySQL pentru stocarea datelor structurate (utilizatori, orare, note).
  + **Controlul Versiunilor:** Git (cu GitHub) pentru gestionarea codului sursă și colaborare.
  + **CI/CD:** GitHub Actions pentru automatizarea proceselor de construire, testare și implementare.

## Justificarea alegerilor tehnice

Alegerile tehnice sunt justificate de necesitatea de a construi un sistem scalabil, performant, sigur și ușor de mentenut. Arhitectura pe trei niveluri permite o separare clară a preocupărilor, facilitând dezvoltarea paralelă și scalarea independentă a componentelor. Utilizarea React și Node.js asigură o dezvoltare rapidă și o experiență utilizator optimă. MySQL este o bază de date robustă și populară pentru datele structurate ale sistemului, iar Git și GitHub Actions sunt esențiale pentru un flux de lucru Agile și o livrare continuă de calitate.

# Provocări și soluții

Pe parcursul proiectului, au fost identificate mai multe provocări legate atât de aspecte tehnice, cât și de organizare. Iată cele mai relevante dificultăți și modul în care au fost abordate:

### Provocare: Estimarea precisă a efortului pentru funcționalități complexe

### Soluție: S-a utilizat tehnica *Planning Poker* în sesiunile de rafinare a Product Backlog-ului, implicând toate rolurile în procesul de estimare. User Stories mari au fost descompuse în sub-taskuri mai mici și mai clare, reducând incertitudinea și crescând acuratețea estimărilor.

### Provocare: Menținerea ritmului de lucru într-un proiect individual

* + - **Soluție:** Pentru a simula disciplina echipei Agile, am introdus sesiuni de lucru zilnice cu obiective mici, ca un pseudo „Daily Scrum”.  
      Board-ul YouTrack a fost utilizat pentru a urmări constant progresul și a marca sarcinile în funcție de stadiu: To Do, In Progress, Preview, Done.

### Provocare: Integrarea unui pipeline CI/CD funcțional.

* + - **Soluție:** Inițial, configurarea GitHub Actions a fost o provocare din cauza lipsei de exemple exacte pentru stack-ul proiectului.  
      S-a pornit cu un workflow simplu pentru build și test, apoi s-au adăugat treptat pași pentru deploy și rularea automată a testelor.  
      Template-uri existente și documentația GitHub Actions au fost reutilizate și adaptate.

### Provocare: Detectarea și gestionarea erorilor (bug-uri).

* + - **Soluție:** Pentru fiecare Sprint, au fost presupuse și documentate bug-uri posibile, apoi adăugate în YouTrack ca Bug issues și distribuite pe coloanele relevante (ex: Bugs, In Progress, Preview).  
      Această abordare a permis simularea realistă a procesului de testare și fixare, chiar dacă nu a existat o echipă QA dedicată.

Aceste provocări au fost transformate în oportunități de învățare, consolidând înțelegerea principiilor Agile și a importanței automatizării și a colaborării în dezvoltarea software.