Facultatea de Matematică și Informatică

Universitatea București

**Document de Analiză și Proiectare pentru Aplicația Software**

**TITLUL PROIECTULUI**

**Analiza de prelucrare a datelor in educatie**

**Nume Student**

**Rădulescu Emanuel-Andrei**

**Adresa de email**

**emanuel-andrei.radulescu@s.unibuc.ro**

**Data**

**05.13.2025**

Contents

[**1. Introducere** 3](#_Toc198037783)

[**1.1 Scopul documentului** 3](#_Toc198037784)

[**1.2 Domeniul de aplicare** 4](#_Toc198037785)

[**1.3 Definiții, acronime și abrevieri** 4](#_Toc198037786)

[**1.4 Referințe** 4](#_Toc198037787)

[**1.5 Prezentare generală** 5](#_Toc198037788)

[**2. Analiza cerințelor** 6](#_Toc198037789)

[**2.1 Cerințe funcționale** 6](#_Toc198037790)

[**2.2 Cerințe non-funcționale** 7](#_Toc198037791)

[**2.3 Cazuri de utilizare (Use Cases)** 8](#_Toc198037792)

[**2.4 Constrângeri** 9](#_Toc198037793)

[**2.5 Riscuri** 10](#_Toc198037794)

[**3. Arhitectura și proiectarea sistemului** 11](#_Toc198037795)

[**3.1 Arhitectura generală** 11](#_Toc198037796)

[**3.2 Modelul de date** 12](#_Toc198037797)

[**3.3 Diagrame UML (Unified Modelling Language)** 13](#_Toc198037798)

[**3.4 Tehnologii și stack de dezvoltare** 14](#_Toc198037799)

[**3.5 Interfața utilizatorului (UI/UX)** 15](#_Toc198037800)

[**4.** **Implementare** 16](#_Toc198037801)

[**4.1** **Structura codului** 16](#_Toc198037802)

[Descriere pe componente: 16](#_Toc198037803)

[**4.2 Module și componente** 17](#_Toc198037804)

[**4.3 Fluxul de date** 18](#_Toc198037805)

[**5. Testare** 20](#_Toc198037806)

[**6.** **Deployment și mentenanță** 23](#_Toc198037807)

[**7. Concluzii** 26](#_Toc198037808)

# **1. Introducere**

## **1.1 Scopul documentului**

Scopul acestei lucrări este de a proiecta și implementa o platformă de analiză a datelor de cercetare în domeniul educației, cu accent pe evaluarea performanței academice a studenților.

## **1.2 Domeniul de aplicare**

*Descrierea domeniului aplicației și utilizatorilor vizați.*

Lucrarea se încadrează în domeniul informaticii aplicate în educație, cu aplicabilitate directă în instituții de învățământ , centre de cercetare educațională și departamente de analiză a datelor . Platforma poate fi utilizată pentru monitorizarea performanței studenților, generarea de rapoarte statistice, dar și determinarea unor situatii scolare mai complexe.

## **1.3 Definiții, acronime și abrevieri**

Această secțiune definește termenii, acronimele și abrevierile utilizate în document pentru a asigura o înțelegere clară și consecventă a conceptelor discutate.

* Shiny-Un pachet R care permite crearea de aplicații web interactive folosind doar R.
* UI-Interfață Utilizator (User Interface); partea aplicației care este vizibilă și interactivă pentru utilizator.
* RMarkdown (.Rmd)-Format de fișier utilizat pentru a crea documente dinamice (PDF, HTML etc.) în R, combinând cod și text.
* DT-Pachet pentru tabele interactive, bazat pe JavaScript DataTables.
* Server-Componenta aplicației care gestionează logica, prelucrează datele și generează rezultatele

## **1.4 Referințe**

Această secțiune conține lista documentelor, standardelor și surselor utilizate în procesul de analiză și proiectare a aplicației. Referințele asigură validitatea și conformitatea soluției propuse cu cele mai bune practici și cerințe industriale.

**1.Surse Web și Bibliografie**

* **Chang, W., Cheng, J., Allaire, J. J., Xie, Y., & McPherson, J. (2021).** shiny: Web Application Framework for R. R package version 1.7.1. <https://CRAN.R-project.org/package=shiny>
* **Attali, D. (2021).** shinyjs: Easily Improve the User Experience of Your Shiny Apps in Seconds. <https://CRAN.R-project.org/package=shinyjs>
* **Wickham, H., François, R., Henry, L., & Müller, K. (2022).** dplyr: A Grammar of Data Manipulation. <https://CRAN.R-project.org/package=dplyr>
* **Xie, Y. (2022).** rmarkdown: Dynamic Documents for R. <https://rmarkdown.rstudio.com>

**2. Alte surse de inspirație sau documentație**

* RStudio (2023). Shiny - Web Applications in R. <https://shiny.rstudio.com>
* RStudio (2023). R Markdown: The Definitive Guide. <https://bookdown.org/yihui/rmarkdown/>
* Wickham, H., & Grolemund, G. (2017). *R for Data Science*. O’Reilly Media. https://r4ds.had.co.nz

## **1.5 Prezentare generală**

Această aplicație interactivă a fost realizată utilizând limbajul de programare **R** și framework-ul **Shiny**, având ca scop principal dezvoltarea unei interfețe web moderne care permite vizualizarea și analiza interactivă a datelor. Prin intermediul unei interfețe grafice intuitive, utilizatorii pot interacționa direct cu seturile de date, pot selecta parametrii de interes și pot genera automat grafice, tabele și rapoarte dinamice.

Aplicația integrează componente vizuale precum **grafice create cu ggplot2**, **tabele dinamice cu DT**, **filtre și controale reactive** și posibilitatea de a **exporta rezultatele în format PDF sau HTML**, folosind rmarkdown și knitr. De asemenea, a fost utilizată funcționalitatea reactive() și observeEvent() pentru a asigura o experiență fluidă și adaptabilă la schimbările din interfață.

Structura aplicației este modulară, fiind organizată în mai multe tab-uri sau panouri (dashboard), fiecare corespunzând unei funcționalități specifice: încărcare de date, vizualizare, analiză statistică, export de rezultate etc.

Această aplicație este utilă atât pentru scopuri educaționale (în învățarea programării în R și a vizualizării de date), cât și pentru aplicații practice în domenii precum economie, medicină, ecologie sau științe sociale, unde analiza interactivă a datelor este esențială.

# **2. Analiza cerințelor**

## **2.1 Cerințe funcționale**

Aplicația trebuie să îndeplinească următoarele cerințe funcționale:

1.Autentificare și autorizare

* + Utilizatorii trebuie să se autentifice folosind un nume de utilizator și o parolă.
  + Administratorii pot gestiona drepturile de acces ale utilizatorilor.

1. Vizualizarea datelor
   * Datele încărcate pot fi vizualizate într-un tabel interactiv (folosind DT::datatable).
   * Utilizatorul poate sorta, filtra și căuta în cadrul tabelului.
2. Selectarea variabilelor pentru o analiza complexa
   * Utilizatorul poate selecta coloane din setul de date pentru a fi analizate.
   * Selectarea se face prin controale de tip selectInput() sau checkboxGroupInput().
3. Generarea de grafice/rapoarte
   * Aplicația permite generarea de grafice (ex: histogramă, diagramă de bare, scatter plot) în funcție de datele selectate.
   * Exportul rapoartelor în formate standard (PDF, CSV, Excel).
4. Exportul rezultatelor
   * Utilizatorul poate exporta datele analizate sau graficele sub formă de fișier PDF sau HTML.
   * Exportul se face prin integrarea cu rmarkdown si downloadHandler
5. Interfata dinamica si reactiva
   * Toate elementele interfeței se actualizează automat în funcție de acțiunile utilizatorului (prin reactive(), observe() și render\*())..

## **2.2 Cerințe non-funcționale**

Aplicația trebuie să îndeplinească următoarele cerințe non-funcționale pentru a asigura performanța, securitatea și scalabilitatea necesare utilizatorilor:

1.**Ușurință în utilizare (usability)**

* Interfața aplicației este intuitivă și ușor de folosit, fără a necesita cunoștințe avansate de R.
* Comenzile sunt clare, etichetate corespunzător, iar navigarea se face logic între secțiuni.

2.**Compatibilitate**

* Aplicația este compatibilă cu principalele browsere web moderne (Google Chrome, Mozilla Firefox, Microsoft Edge).
* Funcționează corect pe sisteme de operare Windows, macOS și Linux.

3.**Performanță**

* Timpul de încărcare și de reacție al aplicației este redus, chiar și în cazul fișierelor .csv de dimensiuni medii (~10.000 de rânduri).
* Răspunsul interactiv al aplicației este aproape instantaneu pentru majoritatea operațiunilor.

4.**Fiabilitate și stabilitate**

* Aplicația gestionează corect erorile de încărcare sau formatare incorectă a fișierelor.
* Mecanisme de verificare a validității datelor sunt implementate pentru a evita prăbușirea aplicației.

5. **Securitate**

* Datele încărcate nu sunt salvate pe server, ci procesate doar local (în sesiunea utilizatorului).
* Aplicația nu permite rularea de cod nesigur sau accesul neautorizat la fișierele sistemului.

## **2.3 Cazuri de utilizare (Use Cases)**

Această secțiune descrie interacțiunile dintre utilizator și sistem prin intermediul unor scenarii specifice. Mai jos sunt prezentate principalele cazuri de utilizare ale aplicației, însoțite de o diagramă UML generală.

UC1 – Încărcarea fișierului .csv

* **Descriere**: Utilizatorul selectează și încarcă un fișier .csv din sistemul propriu.
* **Precondiții**: Aplicația este pornită.
* **Pași**:
  1. Utilizatorul apasă butonul "Încarcă fișier".
  2. Selectează fișierul .csv.
  3. Fișierul este citit și prelucrat.
* **Rezultat**: Datele sunt afișate într-un tabel interactiv.

UC2 – Vizualizarea datelor

* **Descriere**: După încărcare, utilizatorul poate vedea datele în format tabelar.
* **Pași**:
  1. Utilizatorul accesează tab-ul "Date brute".
  2. Se afișează un tabel cu datele încărcate.
* **Rezultat**: Tabelul poate fi sortat, filtrat și exportat.

UC3 – Generarea de statistici descriptive

* **Descriere**: Utilizatorul selectează coloane pentru care dorește statistici (medie, mediană, deviație standard etc.).
* **Pași**:
  1. Utilizatorul accesează tab-ul "Statistici".
  2. Selectează coloanele dorite.
  3. Apasă pe butonul "Generează statistici".
* **Rezultat**: Se afișează rezultatele calculelor statistice.

## **2.4 Constrângeri**

Această secțiune descrie limitările tehnice, legale și operaționale care pot influența dezvoltarea și utilizarea aplicației..

1. **Constrângeri tehnice**

 Aplicația rulează în mediu **R/Shiny**, ceea ce presupune instalarea prealabilă a pachetelor necesare (shiny, tidyverse, DT, ggplot2 etc.).

 Dimensiunea fișierelor .csv încărcate este limitată la **5 MB** (sau o altă limită impusă de server/local).

 Fișierele acceptate trebuie să fie **corect formate**: delimitate prin virgulă sau punct și virgulă, fără caractere nevalide.

1. **Constrângeri legale**
   * Utilizatorul este responsabil pentru conținutul fișierelor încărcate – aplicația nu validează dacă datele sunt conforme cu legislația privind **protecția datelor (ex: GDPR)**.
   * Aplicația nu stochează datele încărcate pe termen lung – toate datele sunt procesate **temporar**, în sesiunea curentă.
2. **Constrângeri operaționale**

* Aplicația trebuie rulată într-un mediu care suportă R și Shiny (ex: RStudio sau un server cu shiny-server).

## **2.5 Riscuri**

Această secțiune evidențiază principalele riscuri asociate aplicației, precum și metodele propuse pentru prevenirea sau gestionarea acestora.

**1.** **Probleme de confidențialitate a datelor**

**Descriere:** Utilizatorul poate încărca date personale sau sensibile fără a ști că acestea nu sunt protejate.  
**Impact:** Încălcarea reglementărilor GDPR sau scurgeri de informații.  
**Gestionare:** Se adaugă un mesaj informativ care avertizează că datele sunt procesate temporar și nu sunt stocate.

**2. Fișiere incorect formate**

**Descriere:** Utilizatorul poate încărca fișiere .csv care nu respectă formatul așteptat (delimitatori greșiți, lipsă antet, caractere speciale etc.).  
**Impact:** Aplicația poate returna erori sau afișa date greșite.  
**Gestionare:** Se implementează validări preliminare (ex: verificarea antetului, numărului de coloane, detectarea delimiterului), cu mesaje de eroare clare.

**3. Utilizare greșită a aplicației**

**Descriere:** Utilizatorii fără experiență pot interpreta greșit interfața sau rezultatele.  
**Impact:** Analize incorecte sau confuzii.  
**Gestionare:** Interfața este intuitivă, iar fiecare pas este însoțit de explicații și exemple.

# **3. Arhitectura și proiectarea sistemului**

## **3.1 Arhitectura generală**

Această aplicație este construită pe o arhitectură de tip **Model-View-Controller (MVC)**, organizată într-un model **client-server**, care separă clar responsabilitățile și asigură o scalabilitate mai bună a aplicației. Arhitectura este împărțită în următoarele niveluri:

1. Clientul (Frontend)

Clientul este interfața grafică cu utilizatorul (UI), fiind responsabilă de:

* Interacțiunea utilizatorului cu aplicația (navigare, introducere date, declanșare acțiuni)
* Transmiterea cererilor către server
* Preluarea răspunsurilor de la server și afișarea acestora într-un format accesibil și atractiv

Interfața este implementată folosind [ex. HTML, CSS, JavaScript / R + Shiny UI], cu elemente dinamice și componente responsive.

2. Serverul (Backend)

Serverul gestionează logica aplicației, accesul la date și procesarea cererilor primite de la client. Acesta este organizat în trei componente principale:

* **Modelul (Model):** gestionează datele aplicației și interacționează cu baza de date. Conține structurile de date și logica pentru manipularea acestora.
* **Controlerul (Controller):** procesează cererile venite de la client, apelează modelul pentru date și pregătește răspunsurile.
* **Vederea (View):** generează conținutul care este transmis clientului (ex. pagini HTML, grafice, tabele dinamice).

Serverul este implementat folosind [ex. R + Shiny Server / Node.js / Flask / Django], oferind un set de funcționalități accesibile printr-o interfață web.

3. Baza de date

Baza de date este responsabilă pentru stocarea persistentă a datelor (ex. informații despre utilizatori, date academice, rezultate statistice). Poate fi implementată folosind un sistem de gestiune a bazelor de date precum [ex. PostgreSQL, SQLite, MongoDB].

Accesul la bază se face exclusiv prin intermediul serverului, care controlează operațiile de interogare și modificare.

## **3.2 Modelul de date**

Modelul de date al aplicației este proiectat pentru a susține analiza detaliată a performanței studenților, administrarea conturilor de utilizator și generarea de rapoarte personalizate. Acesta este structurat în mai multe entități relaționate logic între ele, după cum urmează:

**Diagrama Entitate-Relatie (ERD)**

Utilizatori

Această tabelă stochează informațiile despre conturile care pot accesa aplicația.

* id\_utilizator (PK) – identificator unic
* username – numele de utilizator
* parola – parola criptată
* rol – rolul în aplicație (ex: "admin", "profesor")

Studenti

Conține informații despre studenții analizați.

* id\_student (PK) – identificator unic
* id\_utilizator\_fk (FK) – face legătura cu utilizatorul asociat (dacă e cazul)
* nume, prenume – datele de identificare
* clasa – clasa sau anul de studiu
* mediu – mediul de proveniență (urban/rural)

Rezultate

Această tabelă păstrează notele sau rezultatele evaluărilor.

* id\_rezultat (PK) – identificator unic
* id\_student\_fk (FK) – referință către Studenti
* disciplina – materia
* nota – valoarea notei
* semestru – semestrul în care a fost acordată nota

Log\_Autentificare

Urmărește activitatea de autentificare în aplicație pentru analiză și securitate.

* id\_log (PK) – identificator unic
* id\_utilizator\_fk (FK) – referință către Utilizatori
* data\_autentificare – data și ora accesului
* succes – indicator binar (TRUE/FALSE) privind reușita autentificării

**Descrierea entităților și relațiilor**

* Relația **1 la 1** între Utilizatori și Studenti (dacă fiecare student are cont propriu)
* Relația **1 la N** între Studenti și Rezultate – un student poate avea mai multe rezultate
* Relația **1 la N** între Utilizatori și Log\_Autentificare – un utilizator poate avea mai multe logări înregistrate

## **3.3 Diagrame UML (Unified Modelling Language)**

Această secțiune prezintă principalele diagrame UML utilizate pentru modelarea aplicației.

* 1. Diagrama de cazuri de utilizare (Use Case Diagram)

Această diagramă descrie interacțiunile dintre utilizatori (actori) și funcționalitățile principale ale aplicației.

**Actori:**

* **Administrator** – gestionează utilizatori
* **Profesor** – analizează datele și generează rapoarte
* **Student** – vizualizează propriile performanțe (dacă e implementat)

**Cazuri de utilizare:**

* Autentificare
* Vizualizare date studenți
* Filtrare date (pe clasă, mediu, semestru etc.)
* Analiză statistică
* Generare raport PDF
* Vizualizare loguri de autentificare (doar admin)

**2.** Diagrama de clase

Această diagramă descrie structura logică a aplicației, clasele implicate și relațiile dintre ele. Fiind o aplicație Shiny, clasele pot fi interpretate ca module funcționale sau structuri de date.Utilizatorul introduce acreditările.

1. Utilizator
2. Autentificare
3. Student

## **3.4 Tehnologii și stack de dezvoltare**

Această aplicație a fost dezvoltată utilizând următorul set de tehnologii:

Limbaje de programare:

* **R** – limbajul principal utilizat pentru dezvoltarea aplicației, prelucrarea datelor, analize statistice și generarea rapoartelor.
* **HTML / CSS** – utilizate pentru stilizarea interfeței în cadrul aplicației Shiny.
* **JavaScript (opțional)** – utilizat pentru interactivitate suplimentară în interfață, acolo unde este necesar (de ex. validări, animații etc.).

Framework-uri și biblioteci:

* **Shiny** – framework-ul principal pentru dezvoltarea aplicației web interactive în R.
* **shinydashboard** – pentru crearea unei interfețe cu panouri și taburi.
* **shinyjs** – pentru integrarea de funcționalități JavaScript în aplicație.
* **ggplot2** – pentru realizarea de grafice și vizualizări statistice.
* **dplyr / tidyr** – pentru manipularea datelor.
* **DT** – pentru afișarea tabelelor interactive.
* **rmarkdown + knitr + pagedown** – pentru generarea rapoartelor PDF.
* **bcrypt / digest** – pentru criptarea parolelor și securizarea autentificării.

Bază de date:

* **Fișier CSV sau RDS** – pentru stocarea datelor despre studenți (într-o versiune simplificată).
* **(Opțional)** **SQLite** – pentru gestionarea utilizatorilor și autentificării, dacă se preferă o soluție persistentă și scalabilă.

Alte unelte:

* **RStudio** – mediu de dezvoltare integrat folosit pentru scrierea codului și testarea aplicației.
* **Git** – pentru controlul versiunilor și colaborare (dacă aplicația este dezvoltată în echipă).

## **3.5 Interfața utilizatorului (UI/UX)**

Aplicația a fost proiectată având în vedere simplitatea, claritatea și ușurința în utilizare, pentru a oferi utilizatorului final (profesori, administratori educaționali) o experiență fluidă și intuitive.

Structura generală a interfeței:

Interfața este construită folosind pachetul shinydashboard, care oferă o organizare clară în panouri și taburi, după cum urmează:

* **Pagini disponibile:**
  1. **Autentificare** – Pagina inițială cu formular de logare (utilizator și parolă), cu mesaje de feedback (autentificare reușită/eșuată).
  2. **Tabel studenți** – Vizualizarea datelor brute despre studenți sub formă de tabel interactiv (DT::datatable) cu opțiuni de filtrare și sortare.
  3. **Analize statistice** – Pagina de vizualizare a graficelor (medii pe clasă, distribuții, comparații între medii, corelații etc.), generate cu ggplot2.
  4. **Raport PDF** – Pagina de generare a unui raport personalizat în format PDF pe baza datelor filtrate.
  5. **Deconectare** – Buton de logout vizibil după autentificare.

Elemente de design:

* **Paletă de culori**: Nuanțe de albastru și gri, cu accent pe claritate vizuală și contrast ridicat pentru lizibilitate.
* **Navigare laterală (sidebar)**: Permite acces rapid la secțiunile aplicației.
* **Componente interactive**:
  + Selectoare (selectInput, checkboxGroupInput) pentru filtrarea datelor.
  + Butoane (actionButton, downloadButton) cu funcționalități clare și feedback vizual.
* **Feedback pentru utilizator**: Mesaje de succes sau eroare, animații ușoare (prin shinyjs) pentru validări.

Responsivitate și accesibilitate:

* Interfața este adaptabilă pentru diferite dimensiuni de ecran.
* Elemente interactive au fost etichetate clar pentru a fi ușor de înțeles chiar și pentru utilizatori cu experiență minimă tehnică.

# **Implementare**

# **Structura codului**

Codul aplicației este organizat modular pentru a separa clar logica de interfață, funcționalitatea și gestionarea datelor, asigurând astfel scalabilitate, întreținere ușoară și claritate.

### Descriere pe componente:

* **global.R**  
  Conține:
  + Pachetele încărcate (library(shiny), ggplot2, shinyjs, DT, etc.)
  + Încărcarea datelor inițiale (studenti.csv)
  + Funcții globale disponibile în server.R și ui.R
* **ui.R**  
  Definirea interfeței utilizator:
  + Structura dashboard-ului (shinydashboard)
  + Pagina de autentificare (dacă nu este logat)
  + Taburi: Tabel, Analize, Raport PDF
* **server.R**  
  Conține:
  + Logica reactivă a aplicației
  + Validarea autentificării (prin funcții din auth.R)
  + Procesarea datelor filtrate
  + Generarea graficelor
  + Crearea și descărcarea raportului PDF
* **auth.R**
  + Funcții pentru:
    - Verificarea credențialelor (user/parolă)
    - Gestionarea sesiunii autentificate
    - Deconectare
* **statistici.R**
  + Funcții dedicate pentru calcule statistice:
    - Medii pe clase
    - Distribuții de note
    - Corelații și comparații
* **pdf\_export.R**
  + Funcție ce folosește rmarkdown pentru generarea dinamică a unui raport PDF pe baza datelor filtrate
* **helpers.R**
  + Funcții reutilizabile:
    - Filtrarea datelor în funcție de medii, clase, etc.
    - Formatare tabele
    - Validări de input
* **www/**
  + CSS personalizat pentru design interfață
  + Logo sau alte elemente vizuale
  + Eventual scripturi JavaScript pentru îmbunătățirea experienței UI
* **date/**
  + Conține fișierul de date de intrare (inițial studenti.csv)
  + Poate include și fișiere salvate de sesiuni anterioare

## **4.2 Module și componente**

Aplicația este compusă din mai multe module software, fiecare cu o responsabilitate clară în arhitectura generală. Această structurare modulară permite dezvoltarea și întreținerea ușoară a aplicației, precum și reutilizarea codului.

1. Modulul de **Autentificare** (auth.R)

* **Funcționalitate:**
  + Validarea credențialelor utilizatorilor pe baza unei liste predefinite sau a unui fișier extern.
  + Asigurarea accesului doar utilizatorilor autentificați.
  + Gestionarea sesiunii și deconectarea utilizatorului.

2. Modulul **Interfață Utilizator (UI)** (ui.R)

* **Funcționalitate:**
  + Definirea aspectului vizual al aplicației folosind shinydashboard.
  + Structurarea conținutului în taburi:
    - Tabelul studenților
    - Analize statistice
    - Raport PDF
  + Integrarea formularului de login dacă utilizatorul nu este autentificat.

3. Modulul **Server și logică reactivă** (server.R)

* **Funcționalitate:**
  + Procesarea cererilor utilizatorului și actualizarea interfeței în mod reactiv.
  + Apelarea funcțiilor din celelalte module (autentificare, filtre, statistici, export).
  + Gestionarea fluxului de date și a interacțiunii între UI și date.

4. Modulul **Filtrare și manipulare date** (helpers.R)

* **Funcționalitate:**
  + Aplicarea de filtre asupra datasetului pe baza selecțiilor făcute în UI (ex: mediu, clasă, gen).
  + Transformări și validări asupra datelor brute pentru a fi compatibile cu afișarea și analiza.

5. Modulul **Analize Statistice** (statistici.R)

* **Funcționalitate:**
  + Calcule statistice precum: medii generale, comparații între clase, distribuții.
  + Generarea de grafice (bar charts, histograme, boxplots) pe baza datelor filtrate.
  + Afișarea vizuală a rezultatelor în tabul „Analize”.

## **4.3 Fluxul de date**

Această secțiune descrie modul în care datele sunt procesate și utilizate în cadrul aplicației, de la încărcarea inițială până la generarea analizelor și exportul raportului.

1. Încărcarea datelor

* La inițializarea aplicației, fișierul studenti.csv este încărcat în memoria aplicației prin read.csv în fișierul global.R.
* Aceste date conțin informații despre studenți: nume, prenume, clasă, mediu, gen, note etc.

2. Autentificarea utilizatorului

* Înainte de accesul la date, utilizatorul trebuie să se autentifice.
* Dacă autentificarea reușește, aplicația continuă cu afișarea conținutului din ui.R, altfel rămâne pe ecranul de login.

3. Filtrarea datelor (date\_filtrate)

* După autentificare, datele brute sunt filtrate în funcție de selecțiile utilizatorului din UI (ex: clasă, mediu, gen).
* Acest proces este gestionat de o funcție reactivă (date\_filtrate()), care returnează un subset din studenti.csv, actualizat în timp real la modificarea selecțiilor.

4. Afișarea tabelului de studenți

* Datele filtrate sunt afișate într-un tabel interactiv (DT::datatable()), în tabul „Tabel studenți”.
* Utilizatorul poate sorta, căuta sau exporta aceste date local.

# **5. Testare**

**5.1Strategia de testare**

Strategia de testare pentru aplicația Shiny se va axa pe o combinație între testarea manuală și automată, având ca scop asigurarea funcționării corecte a aplicației, securitatea și performanța acesteia. Testarea va include următoarele etape și abordări:

1. **Testare unitară**
   * Verificarea funcționării corecte a fiecărui modul individual al aplicației (de exemplu, autentificarea utilizatorului, gestionarea datelor, generarea rapoartelor).
   * Testarea va utiliza librării precum testthat pentru testarea funcționalităților din R, inclusiv a funcțiilor reactive și a interacțiunilor UI.
2. **Testare de integrare**
   * Verificarea corectitudinii interacțiunii între modulele aplicației Shiny (de exemplu, între formularele de input și generarea rapoartelor).
   * Testarea integrării funcționalităților prin simularea interacțiunilor între utilizator și aplicație folosind shinytest.
3. **Testare de performanță**
   * Evaluarea răspunsului aplicației la un număr mare de cereri simultane și evaluarea timpului de răspuns pentru funcțiile reactive.
   * Utilizarea unor instrumente precum profvis pentru a analiza performanța aplicației la diferite încărcături de date.
4. **Testare de securitate**
   * Identificarea vulnerabilităților de securitate, inclusiv protecția împotriva atacurilor de tip SQL Injection sau Cross-Site Scripting (XSS).
   * Folosirea unor scanări de securitate automate (de exemplu, OWASP ZAP) și analiza cerințelor de securitate ale aplicației (de exemplu, criptarea datelor sensibile).
5. **Testare de acceptanță**
   * Evaluarea aplicației din perspectiva utilizatorilor finali, asigurându-se că cerințele funcționale sunt îndeplinite.
   * Crearea de scenarii de testare bazate pe cerințele inițiale ale utilizatorilor și obținerea de feedback pentru optimizarea interfeței și experienței utilizatorului.
6. **Automatizarea testelor**
   * Implementarea testelor automate pentru regresie și validare continuă.
   * Integrarea testelor în pipeline-ul CI/CD folosind GitHub Actions pentru a asigura testarea continuă a aplicației în momentul modificării codului.

5.2 Tipuri de teste

În cadrul acestui proiect Shiny, vor fi folosite mai multe tipuri de teste pentru a asigura calitatea și funcționalitatea aplicației:

1. **Teste unitare**
   * Testează funcționarea corectă a fiecărei funcții sau componente a aplicației, de exemplu, validarea datelor de intrare sau funcțiile de calcul.
   * Se utilizează testthat pentru testarea funcționalităților backend (R).
2. **Teste de integrare**
   * Asigură că modulele interconectate ale aplicației funcționează corect împreună, de exemplu, validarea că datele introduse prin formular sunt corect procesate și reflectate în rapoarte.
   * Se utilizează shinytest pentru a simula interacțiuni utilizator-aplicație și pentru a verifica integrarea corectă a componentelor.
3. **Teste de performanță**
   * Se analizează timpul de răspuns al aplicației și se monitorizează performanța aplicației sub sarcină crescută.
   * Se utilizează profvis pentru a analiza performanța aplicației și a observa zonele care pot fi optimizate.
4. **Teste de securitate**
   * Testează protecția aplicației împotriva vulnerabilităților, cum ar fi injectarea SQL, atacurile XSS sau CSRF.
   * Se utilizează instrumente de securitate ca OWASP ZAP pentru scanarea automată a vulnerabilităților aplicației.
5. **Teste de acceptanță**
   * Verifică dacă aplicația respectă cerințele utilizatorilor și cerințele funcționale.
   * Scenariile de testare vor include validarea fluxurilor de utilizator, cum ar fi autentificarea, procesul de generare a rapoartelor, și testarea interfeței utilizator.

5.3 Plan de testare

Planul de testare va include scenarii detaliate pentru a valida funcționalitățile aplicației în conformitate cu cerințele funcționale și non-funcționale.

1. **Obiectivele testării**
   * Asigurarea că aplicația funcționează conform specificațiilor (autentificare, generare rapoarte).
   * Verificarea performanței aplicației și a comportamentului sub sarcină ridicată.
   * Testarea securității pentru a preveni vulnerabilitățile comune.
2. **Scenarii de testare**
   * **Testarea autentificării**  
     Scenariul: Un utilizator încearcă să se autentifice cu un nume de utilizator și parolă corecte.
     + Pași:
       1. Accesarea paginii de autentificare.
       2. Introducerea unui nume de utilizator și a unei parole corecte.
       3. Apăsarea butonului „Autentificare”.
       4. Sistemul redirecționează utilizatorul către dashboard.
     + Rezultatul așteptat: Utilizatorul este autentificat cu succes și accesul este permis.
   * **Testarea generării de rapoarte**  
     Scenariul: Un utilizator generează un raport de activitate pe baza datelor introduse.
     + Pași:
       1. Accesarea secțiunii „Rapoarte”.
       2. Selectarea tipului de raport dorit.
       3. Aplicarea filtrelor relevante.
       4. Generarea și descărcarea raportului.
     + Rezultatul așteptat: Raportul este generat și descărcat corect.
   * **Testarea performanței**  
     Scenariul: Evaluarea timpului de răspuns al aplicației la cereri simultane.
     + Pași:
       1. Simularea a 1000 de cereri simultane către server.
       2. Monitorizarea timpului de răspuns.
     + Rezultatul așteptat: Timpul mediu de răspuns nu depășește 2 secunde.
   * **Testarea securității**  
     Scenariul: Testarea aplicației pentru vulnerabilități la atacuri de tip SQL Injection.
     + Pași:
       1. Introducerea unui input malițios în câmpurile de login.
       2. Observarea reacției sistemului.
     + Rezultatul așteptat: Aplicația respinge inputul malițios și nu permite atacuri SQL.
3. **Instrumente utilizate pentru testare**
   * testthat pentru testele unitare.
   * shinytest pentru testele de integrare.
   * profvis pentru analiza performanței.
   * OWASP ZAP pentru scanarea securității aplicației.

# **Deployment și mentenanță**

**6.1 Strategie de Deployment**

Procesul de implementare al aplicației Shiny urmează o abordare bazată pe standardele DevOps, asigurându-se astfel livrarea rapidă și sigură a noilor versiuni ale aplicației pentru analiza performanței studenților.

1. **Mediile de deploy**
   * **Mediu de dezvoltare (Development)** – Testare și dezvoltare continuă de funcționalități noi și îmbunătățiri ale aplicației.
   * **Mediu de testare (Staging)** – Validare a noilor funcționalități înainte de lansarea în mediul de producție.
   * **Mediu de producție (Production)** – Aplicația finală, accesibilă utilizatorilor finali (profesori, administratori, studenți).
2. **Procesul CI/CD (Continuous Integration / Continuous Deployment)**
   * **Commit & Push** – Codul sursă al aplicației este trimis la un repository GitHub/GitLab.
   * **Testare automată** – Se rulează teste unitare și de integrare utilizând framework-uri precum testthat pentru R.
   * **Build automatizat** – Aplicația este construită și testată folosind GitHub Actions pentru R.
   * **Containerizare** – Aplicația Shiny este împachetată într-un container Docker pentru portabilitate și scalabilitate.
   * **Orchestrare** – Aplicația este implementată în Kubernetes pentru a asigura scalabilitatea automată și gestionarea resurselor.
   * **Deploy automatizat** – Folosirea GitHub Actions pentru livrarea automată a noilor versiuni în mediul de producție.
   * **Monitorizare și logging** – Utilizarea Prometheus și Grafana pentru monitorizarea performanței aplicației, iar logs-urile sunt centralizate pentru analiză cu ELK Stack.
3. **Tehnologii utilizate**
   * **Sisteme de versionare:** GitHub
   * **Pipeline CI/CD:** GitHub Actions
   * **Containerizare:** Docker
   * **Orchestrare:** Kubernetes
   * **Monitorizare:** Prometheus, Grafana, ELK Stack
   * **Hosting:** DigitalOcean, AWS

**6.2 Administrarea și Monitorizarea**

În această secțiune se prezintă strategiile și instrumentele folosite pentru administrarea aplicației Shiny, asigurându-se că performanța, securitatea și disponibilitatea aplicației sunt la cele mai înalte standarde.

1. **Monitorizarea performanței**
   * **Prometheus & Grafana** – Colectează și analizează metrice privind utilizarea resurselor (CPU, RAM, utilizarea rețelei).
   * **Shiny Server Logs** – Monitorizează performanța aplicației Shiny și detectează erori sau întârzieri.
   * **RStudio Connect** – Oferă raportări detaliate despre utilizarea aplicației și performanța interfeței.
2. **Monitorizarea securității**
   * **OWASP ZAP** – Utilizat pentru scanări de securitate periodice ale aplicației Shiny, identificând vulnerabilități potențiale.
   * **Audit Trail & Logging** – Înregistrarea acțiunilor importante pentru conformitate și pentru a detecta orice activitate neautorizată.
3. **Administrarea aplicației**
   * **Docker & Kubernetes Dashboard** – Monitorizarea și administrarea containerelor aplicației Shiny.
   * **Logs Shiny Server** – Centralizarea jurnalelor de aplicație pentru a analiza erorile și a optimiza performanța.
4. **Notificări și alerte**
   * **Slack & Email Alerts** – Alerta echipei de dezvoltare și administratorilor în caz de erori critice sau probleme de performanță.
   * **PagerDuty** – Escaladarea alertelor în caz de erori majore sau downtime

**6.3 Plan de Mentenanță**

Planul de mentenanță al aplicației Shiny asigură actualizarea constantă și corectarea problemelor, cu scopul de a menține aplicația performantă și sigură pe termen lung.

1. **Tipuri de mentenanță**
   * **Mentenanță corectivă** – Remedierea bug-urilor identificate de utilizatori sau detectate în urma testelor automate.
   * **Mentenanță preventivă** – Actualizări periodice pentru optimizarea performanței aplicației și prevenirea posibilelor defecțiuni.
   * **Mentenanță evolutivă** – Implementarea de noi funcționalități, precum îmbunătățiri ale interfeței sau adăugarea unor noi tipuri de analize pentru studenți.
   * **Mentenanță adaptivă** – Ajustări ale aplicației pentru a se asigura că funcționează corect cu versiunile noi ale R și ale altor biblioteci.
2. **Program de mentenanță**
   * **Zilnic** – Monitorizarea performanței și înregistrarea alertelor de securitate.
   * **Săptămânal** – Revizuirea și corectarea bug-urilor critice.
   * **Lunar** – Aplicarea actualizărilor de securitate și testarea noilor funcționalități.
   * **Trimestrial** – Optimizarea performanței și evaluarea arhitecturii aplicației.
   * **Anual** – Revizuirea arhitecturii aplicației și migrarea la noi tehnologii sau platforme de hosting.
3. **Instrumente și procese de mentenanță**
   * **Gestionarea Ticketelor** – Utilizarea Jira pentru urmărirea problemelor și cerințelor.
   * **Automatizarea Patch-urilor** – Aplicarea actualizărilor și corecțiilor folosind scripturi automatizate.
   * **Testare după Actualizări** – Teste automate după fiecare actualizare pentru validarea stabilității aplicației.
   * **Backup și Recuperare** – Realizarea de backup-uri automate și testarea periodică a proceselor de restaurare a aplicației.
4. **Suport Tehnic**
   * **Asistență 24/7** – Suport continuu prin email și chat pentru utilizatori și echipele tehnice.
   * **Documentație** – Ghiduri interne pentru echipele tehnice și utilizatori despre utilizarea aplicației.
   * **Canale de comunicare** – Slack și Microsoft Teams pentru escaladarea problemelor tehnice.

# **7. Concluzii**

Această secțiune oferă o sinteză a principalelor rezultate ale dezvoltării aplicației și evidențiază direcții posibile pentru evoluția sa viitoare.

**1. Rezumat al principalelor constatări**

* Aplicația a fost proiectată ca un instrument intuitiv și interactiv pentru analiza performanței studenților, fiind construită cu ajutorul platformei **R Shiny**, ceea ce permite generarea dinamică a conținutului pe baza inputurilor utilizatorului.
* Funcționalitățile implementate — autentificare, filtrare pe clase și medii, vizualizări grafice, tabele dinamice și export în PDF — oferă un suport valoros pentru luarea deciziilor în mediul educațional.
* Structura modulară a aplicației permite ușor adăugarea de noi funcționalități și scalarea ulterioară.
* S-au utilizat practici moderne de deployment (CI/CD, Docker, GitHub Actions), ceea ce permite actualizarea rapidă și sigură a aplicației.
* Sistemul de mentenanță propus asigură performanță ridicată și disponibilitate continuă prin monitorizare activă și backup automatizat.

**2. Recomandări pentru dezvoltări viitoare**

* **Extinderea funcționalităților**: Adăugarea unor noi tipuri de analize (ex. predicții privind riscul de abandon școlar), integrarea cu baze de date externe (ex. catalog electronic, sisteme ERP educaționale).
* **Creșterea interoperabilității**: Export în formate suplimentare (Excel, Word), conectare prin API la alte platforme educaționale.
* **Automatizarea proceselor**: Automatizarea notificărilor și a generării periodice de rapoarte pentru profesori sau directori.
* **Optimizarea performanței**: Îmbunătățirea timpilor de răspuns prin caching local și optimizarea interogărilor pe volume mari de date.
* **Îmbunătățirea experienței utilizatorilor**: Ajustarea interfeței și fluxurilor de lucru în funcție de feedback-ul utilizatorilor (profesori, elevi, administratori).
* **Consolidarea securității**: Adăugarea de autentificare cu doi factori (2FA) și audit trail pentru acțiunile utilizatorilor cu roluri administrative.