Programare WEB
Documentaţie
Grading Script App

Botescu Mihai: mihai.botescu00@e-uvt.ro
Beze Robert: robert.beze00@e-uvt.ro

25 mai 2022

Rezumat

Aplicatia presupune un script pentru notele studentilor obtinute la diferite materii pe parcursul semestrului cat si la examen. Aplicatia va fi utilizata atat de profesori cat si de studenti. Accesul la aplicatie se va realiza pe baza autentificarii cu credentialele conturilor de e-uvt, prin email-ul instituţional.

Cuprins

Capitolul 1

Introducere

1.1 Definirea problemei

Din perspectiva studentului, acesta acceseaza platforma, se autentifica folosind credentialele. Este prezent un meniu cu mai multe optiuni (materiile la care acesta este inscris). Studentul selecteaza o materie, si este redirectionat la o pagina care expune 3 meniuri: notele la componentele de la curs, notele de la componentele de la laborator, nota finala. Din perspectiva profesorului, acesta detine un meniu pt fiecare materie predata; pentru fiecare meniu, un dropdown list cu studentii inscrisi la acel curs.

1.2 Resursele aplicaţiei

1.2.1 Stabilirea resurselor

Resursele [1] stabilite la nivel de aplicaţie sunt următoarele:

- Materia (materiile; colectie) resursa principala
  - Nota (notele; colectie)
  - Prezenta (prezentele; colectie -¿ pentru fiecare Curs, Seminar/Laborator in parte)
- Utilizatorul (student/profesor) resursa principala
  - Profesor (singleton)
  - Studentul (sau studentii – colectie)

1.2.2 Operaţii pe resursele vizate in aplicaţie

Figura 1.1 sugerează operaţiile [1] realizate pe fiecare dintre resurse în parte:

Figura 1.1: Operaţiile pe resursele vizate în aplicaţie

Capitolul 2

Ingineria sistemului

2.1 Arhitectura propusă

Se va merge pe ideea de a utiliza modelul client-server, care să respecte constrângerile ReSTful [1]. Mai precis, va avea loc separarea serviciilor de UI, prin intermediul API-ului. API-ul expune servicii care sunt consumate în UI. Interacţiunea client - server va avea loc prin cereri tipice (de tip GET, POST, PUT, DELETE). Server-ul va emite client-ului, în urma unei cereri, un răspuns tipic (de regulă, în format JSON). Modele de cereri şi răspunsuri din partea client-ului şi a server-ului au fost ilustrate în Sprint-ul anterior, în momentul definirii şi dezvoltării API-ului [2]. Separat de comunicarea client-server, vor exista serviciile definite şi implementate.

2.2 Cazuri de utilizare ale aplicaţiei

Aplicaţia va avea, la cel mai înalt nivel, din perspectiva utilizatorului şi a modului de lucru cu aceasta, 2 principali actori (consideraţi ca fiind de asemenea şi resurse unitare): profesorul şi studentul. Astfel, identificăm 2 scenarii pe baza acestor actori: utilizare din perspectiva studentului, utilizare din perspectiva profesorului.

2.2.1 Cazuri de utilizare din perspectiva studentului

Diagrama de mai jos 2.1 reprezintă (sumar) etapele pe care le va parcurge studentul în utilizarea aplicaţiei.

2.2.2 Cazuri de utilizare din perspectiva profesorului

Diagrama de mai jos 2.2 reprezintă (sumar) etapele pe care le va parcurge profesorul în utilizarea aplicaţiei.

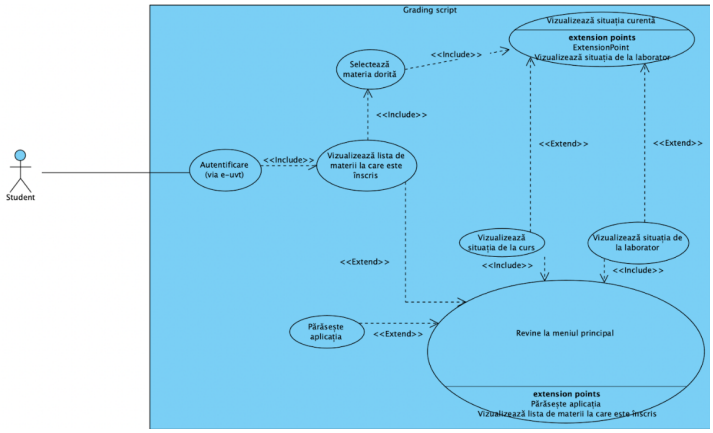

Figura 2.1: Diagrama cazurilor de utilizare din perspectiva studentului

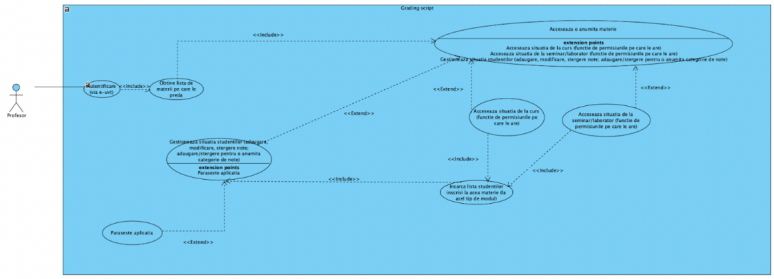

Figura 2.2: Diagrama cazurilor de utilizare din perspectiva profesorului

2.3 Schimburile de mesaje între utilizatorii aplicaţiei

Interacţiunile şi schimburile de mesaje dintre actori au fost evidenţiate prin prisma definirii API-ului.

2.4 Stocarea datelor

Figura 2.3 desemnează diagrama entitate-relaţie a bazei de date prin prisma căreia urmează să putem gestiona fluxul de date din aplicaţia noastră. De menţionat este că se vor utiliza 2 tipuri de baze de date: Maria DB (integrat în phpMyAdmin) şi sqlite3 (integrat via fişier local, parte din proiect).

La o materie există mai multe note, dar un anumit tip de notă poate aparţine unei singure materii (relaţie one-to-many). La o materie, există mai multe prezenţe, dar un tip de prezenţă aparţine unei singure materii (relaţie one-to-many). La fel, un student are mai multe tipuri de prezenţe, în timp ce, o prezenţă aparţine unui student (relaţie one-to-many). În cazul studentului,

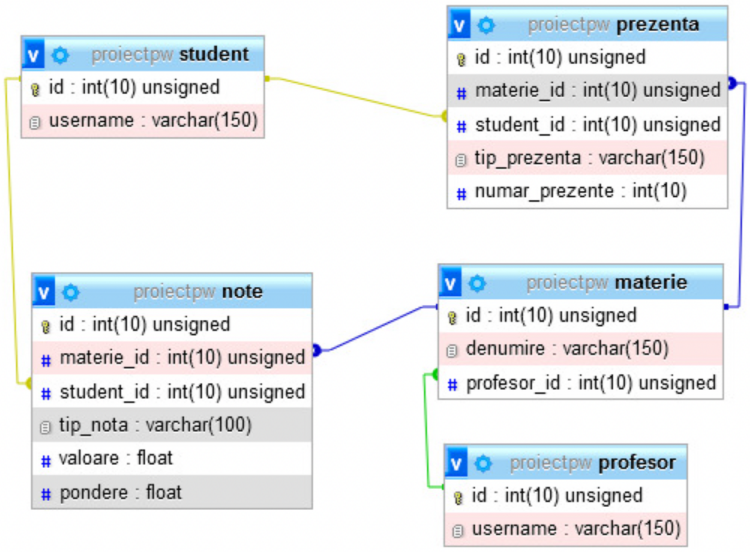

Figura 2.3: Diagrama entitate-relaţie reprezentând schema bazei de date utilizate

acesta poate obţine mai multe tipuri de note, în timp ce, o notă aparţine unui student (relaţie one-to-many). Studentul şi materia se află în relaţie many-to-many, fiindcă un student poate fi înscris la mai multe materii, şi reciproc, o materie conţine o listă de studenţi înscrişi la aceasta. Au fost introduse şi o varietate de constrângeri (constrângeri de cheie - primară, străină; constrângeri pentru valorile pe care le pot lua anumite câmpuri – de exemplu, nota este un număr real cu valori de la 1 la 10).

2.5 Ingineria cerinţelor

2.5.1 Cerinţe pentru Sprint 2

Prin intermediul MS Teams, s-au decis a fi abordate următoarele cerinţe:

1. Modelarea frontend-ului - (cel puţin 70-80%) finalizat până la finalul sprint-ului 2
2. Implementarea unor elemente de interfaţă şi a unor interacţiuni minimale
3. Proiectarea sistemului, ingineria sistemului, arhitectura acestuia
4. Dezvoltarea modelului bazei de date (integral: schema bazei de date, tabele, câmpuri, restricţii: chei primare, chei străine; restricţii asupra câmpurilor tabelelor)
5. Implementarea principalelor operaţii care să ofere o dezvoltare a cerinţelor minimale: majoritatea operaţiilor de tip GET vor fi implementate până la finalul SPRINT-ului 2
6. Implementarea răspunsurilor pentru principalele operaţii care să ofere o dezvoltare a cerinţelor minimale (operatii de get) – identic cu cerinţa anterioară, legat de deadline.

2.5.2 Cerinţe pentru Sprint 3

Prin intermediul MS Teams, s-au decis a fi abordate următoarele cerinţe:

1. Implementarea operaţiilor de modificare (rutele PUT, POST, DELETE)
2. Expunerea serviciilor pentru rutele PUT, POST, DELETE
3. Consumul serviciilor pentru rutele PUT, POST, DELETE
4. Finalizare front-end
5. Instalarea aplicaţiei (configurarea selectării tipului bazei de date folosite: Maria DB sau sqlite3; popularea bazei de date cu câteva date minimale pentru ca aplicaţia să fie utilizabilă şi funcţională)

Capitolul 3

Tehnologii utilizate

3.1 Descrierea stivei de tehnologii

3.1.1 Front-end

* HTML, CSS [3], Bootstrap - pentru definirea şi modelarea elementelor tipografice

* JQUERY [4] [5] (AJAX [6]) - Apeluri asincrone, pentru a consuma serviciile expuse în API

3.1.2 Back-end

* PHP [2] (v7), Slim 3, Twig: pentru rutarea paginilor, expunerea serviciilor

* PDO [2]: Nivelul de abstractizare utilizat pentru a putea comunica din aplicaţie cu baza de date via query-uri

3.1.3 Software utilizat

MAMP (rulare server local), PHPMYADMIN (interfaţă pentru DBMS), Sqlite Studio Browser (pentru a testa dacă datele sunt salvate corect în baza de date sqlite), ReSTer (extensie Google pentru testare ReST), Postman (program pentru testare ReST), Composer (pentru a instala dependinţele necesare pentru o funcţionalitate minimă pentru PHP).

Capitolul 4

Testarea aplicaţiei

Ca şi medii de testare ReST, s-au utilizat: cURL, ReSTer (extensie pentru web browser-ul Google Chrome) şi Postman (aplicaţie desktop).

4.1 Cereri tip GET

Testăm o cerere tip GET (în cazul de faţă, studentul autentificat este `mihai.botescu00@e-uvt.ro`).

`GET /student/materii` returnează lista de materii frecventate de studentul autentificat, ca în exemplul din figura 4.1.

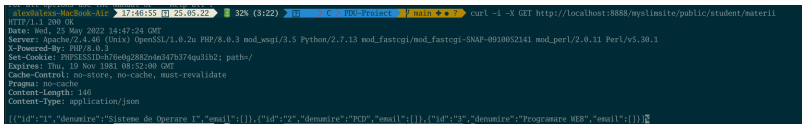

Figura 4.1: Testare cerere GET via curl (success)

`GET /profesor/materii` returnează lista de materii predate de profesorul autentificat, ca în exemplul din figura 4.2. În cazul de faţă, cum profesorul nu este autentificat, vom primi un mesaj de eroare.

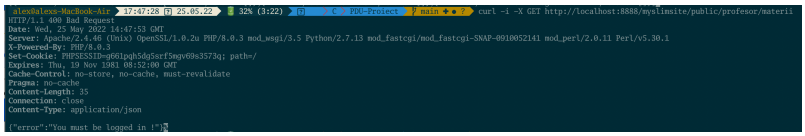

Figura 4.2: Testare cerere GET via curl (error)

`GET /student/materii/1/note` returnează notele studentului autentificat, de la materia cu ID-ul 1, ca în exemplul din figura 4.3.

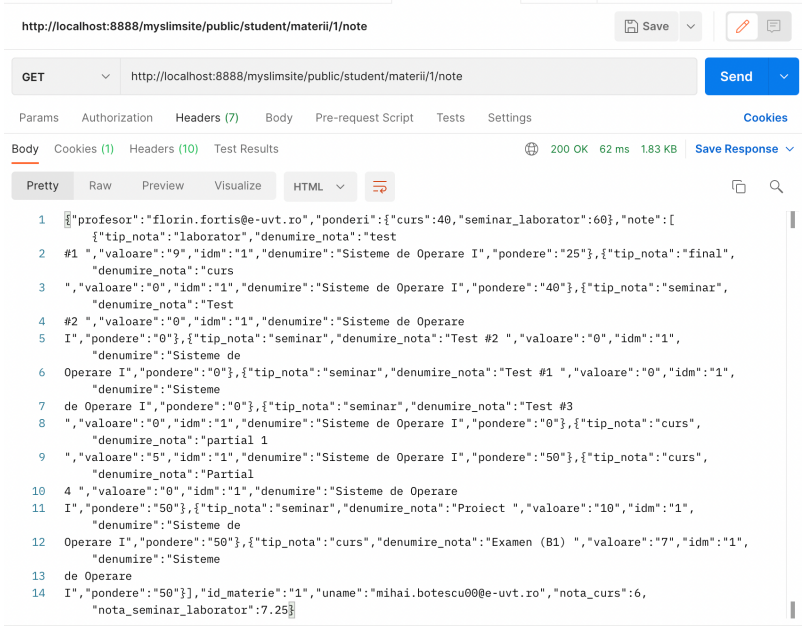

Figura 4.3: Testare cerere GET via Postman (success)

`GET /profesor/materii/1/studenti/1/note` returnează notele studentului cu ID-ul 1, de la materia cu ID-ul 1, ca în exemplul din figura 4.4. Ne autentificăm cu profesorul, pentru ca extragerea datelor să fie posibilă.

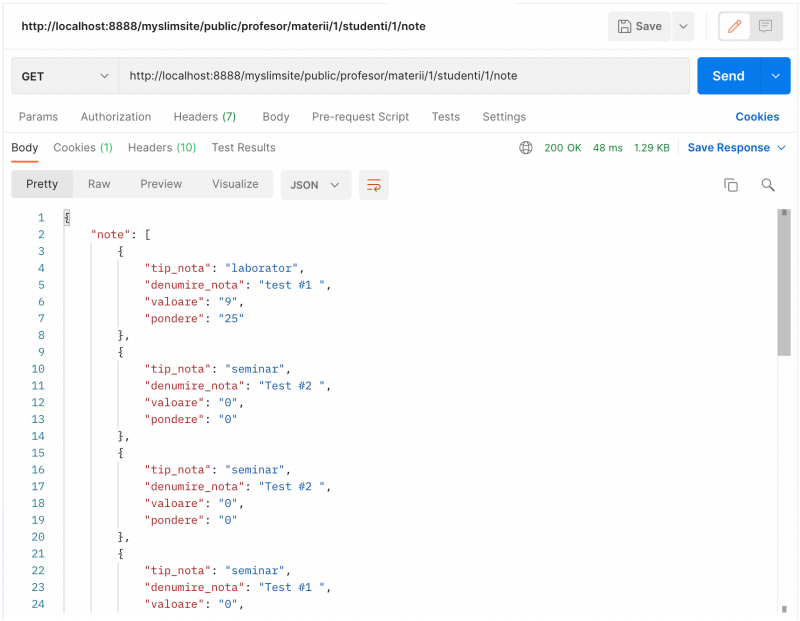

Figura 4.4: Testare cerere GET via Postman (success)

4.2 Cereri tip POST

Adăugăm o notă nouă la o anumită materie, pentru profesorul autentificat curent.

De exemplu, pentru SO1, cum se vede în figura 4.5.

Iar, dacă încercăm să adăugăm din nou aceeaşi notă, vom primi eroare, aşa cum sugerează figura 4.6.

4.3 Cereri tip PUT

Modificăm o anumită notă, la o anumită materie, pentru un anume student, aşa cum sugerează figura 4.7.

4.4 Cereri tip DELETE

ştergem un anume tip de notă, de la o anumită materie, pentru toţi studenţii înscrişi la aceasta, aşa cum sugerează figura 4.8.

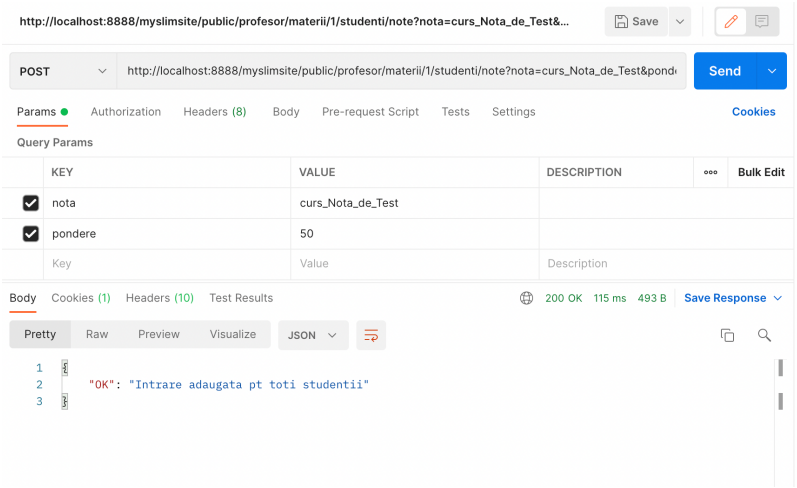

Figura 4.5: Testare cerere POST via Postman (success)

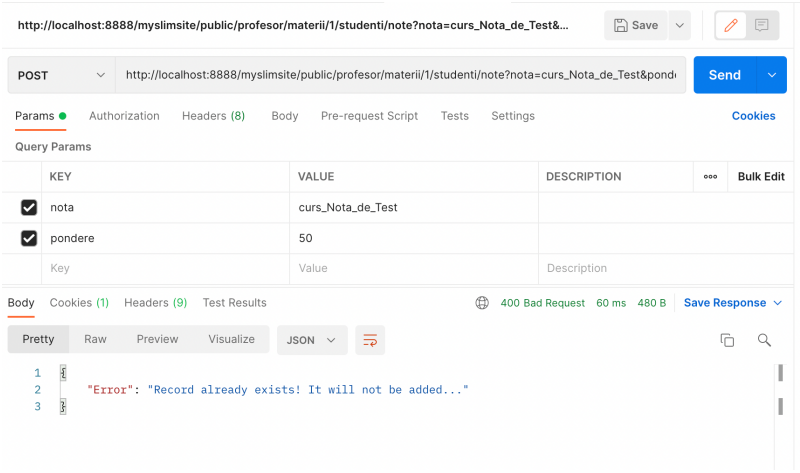

Figura 4.6: Testare cerere POST via Postman (error)

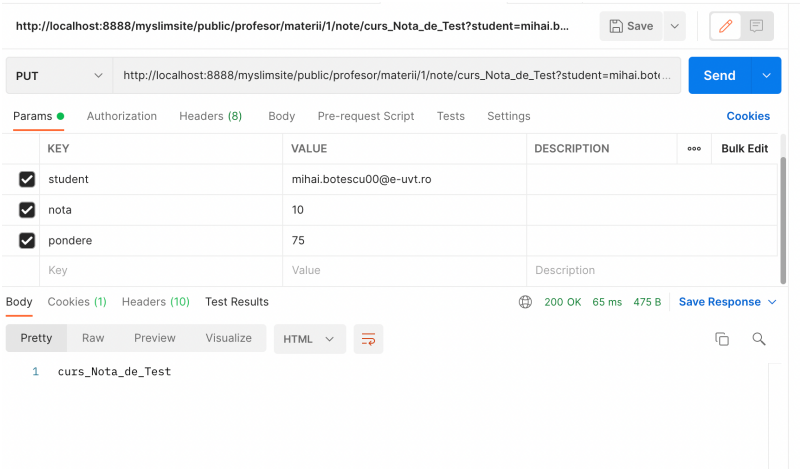

Figura 4.7: Testare cerere PUT via Postman (success)

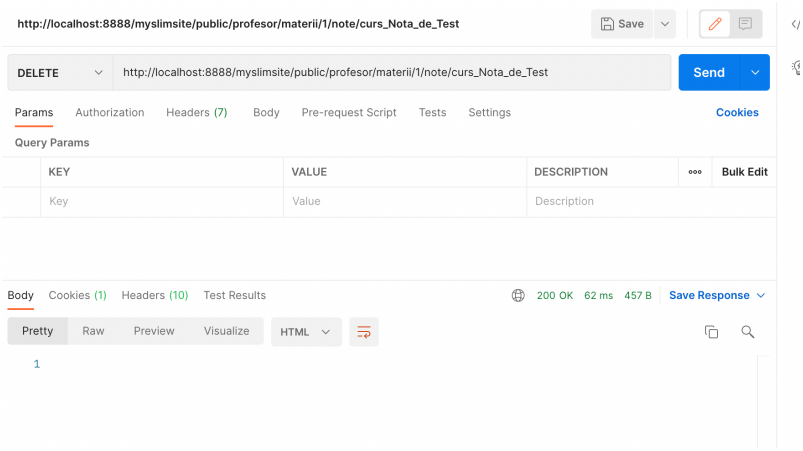

Figura 4.8: Testare cerere DELETE via Postman (success)

Capitolul 5

Concluzii şi direcţii viitoare

Ca şi direcţii viitoare de realizare, se doreşte în primul rând extinderea aplicaţiei spre utilizarea acesteia pe telefon mobil (deci creearea unui client iOS/Android care să poată consuma serviciile ReST expuse aici şi să le uniformizeze în propria interfaţă). Mai apoi, pentru profesor, export-ul de date dintr-un fişier CSV pentru înregistrarea studenţilor în aplicaţie.

Bibliografie

[1] M. Masse, *REST API Design Rulebook*. O'Reilly Media, Inc, USA, Nov. 2011.

[2] V. Vaswani, *PHP: A BEGINNER'S GUIDE*. Beginner's Guide, New York, NY: Osborne/McGraw-Hill, Oct. 2008.

[3] J. Duckett, *HTML and CSS*. Wiley John + Sons, Nov. 2011.

[4] R. York, *Web Development with jQuery*. Nashville, TN: John Wiley & Sons, Mar. 2015.

[5] A. Rauschmayer, *JavaScript for impatient programmers*. Independently Published, Aug. 2019.

[6] J. L. Ford, *Ajax programming for the absolute beginner*. Florence, AL: Delmar Cengage Learning, Oct. 2008.