**Vue.js CRUD Application cu API Express.js**

În urma parcurgerii acestei documentații, vom crea un API, și o aplicație client care vă simula o aplicație de cashflow management, având o baza de date MySQL. Aplicația vă implementa validare de autentificare și criptare de parole.

1. **Descarcați Visual Studio Code:** [**https://code.visualstudio.com/**](https://code.visualstudio.com/)

**A screenshot of a computer

Description automatically generated**

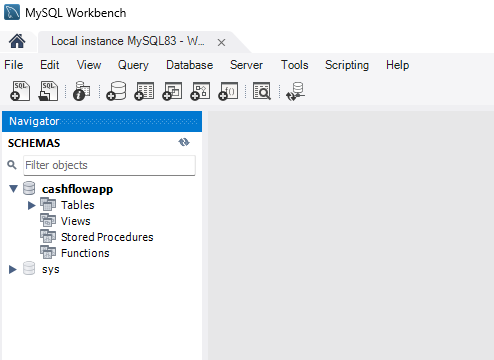
1. **Descarcați Node.js:** [**https://nodejs.org/en**](https://nodejs.org/en)

**A screenshot of a computer screen

Description automatically generated**

1. **Descarcați și configurați serverul local:** [**https://dev.mysql.com/downloads/mysql/**](https://dev.mysql.com/downloads/mysql/)

Configurați user-ul **root** și atribuiți o parolă acestuia (va fi folosită ulterior la conectare). Dupa conectare, folositi **MySQL Workbench** pentru a crea o bază de date numită **cashflowapp**.



1. **Creați un director în care se vă stoca aplicația**

**A screenshot of a computer

Description automatically generated**

În interiorul directorului selectat, creați un subdirector numit server. Acesta se vă deschide cu VS Code.

1. **Se va deschide un terminal în VS Code și se va selecta directorul server**

**A screenshot of a computer

Description automatically generated**

1. **Crea****ți template-ul de server folosind Express.js**

În terminal se vor rula următoarele comenzi:

* **npx express-generator**
* **npm install**
* **npm audit fix --force** (daca exista, sau pana nu mai exista vulnerabilitati)

Aplicația de server va putea fi pornită cu **npm start** șiva rula la URL-ul: <http://localhost:3000/>.

A screenshot of a computer

Description automatically generatedDupă finalizarea pașilor, directorul server va arata precum în poza următoare:

1. **Instalați Swagger UI pentru documentația/testarea API-ului**

In terminalul directorului server, rulați următoarele comenzi:

* **npm install swagger-ui-express**
* **npm install swagger-jsdoc**

1. **Configurați CORS (cross-origin resource sharing) pentru a putea realiza HTTP request-urile externe**

In terminalul directorului **server** se va rula următoarea comandă:

* **npm install cors**

În fișierul **app.js** din directorul **server**, se vor face următoarele modificări pentru a permite aplicației de client, care va fi realizată în pașii următori, să acceseze endpoint-urile API-ului.

**A screen shot of a computer program

Description automatically generated**

1. **Configurați Swagger-UI**

În fișierul **app.js** din directorul **server**, se vor realiza următoarele modificari pentru a permite documentarea/testarea de endpoint-uri pentru API.

A screen shot of a computer program

Description automatically generated

API-ul va putea fi accesat la URL-ul: [**http://localhost:3000/api/**](http://localhost:3000/api/)

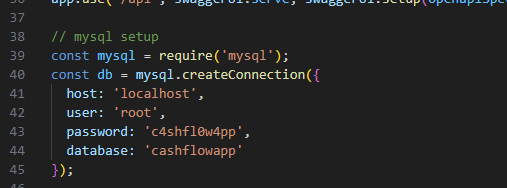
**A screen shot of a computer

Description automatically generated**

1. **Realiza****ți legatura cu serverul MySQL**

In terminalul fișierului server rulați următoarea comandă:

* **npm install mysql**

 În fișierul **app.js** din directorul **server**, se vor declara variabilele responsabile pentru interacțiunea cu serverul **MySQL**.

1. **Conecta****ți-vă la baza de date, creați tabelele, si faceți conexiunea accesibil****ă din toate rutele**

În fișierul **app.js** din drectorul **server**, realizați următoarele modificări. Următorul snippet de cod se vă ocupa și de crearea tabelelor necesare, în caz că acestea nu exista deja în baza de date **cashflowapp** din **MySQL.** Codul se va situa in continuarea setup-ului inițial din pasul 10.

Cod sursă: <https://pastebin.com/6keP3Lsn>

A screen shot of a computer program

Description automatically generated

In continuarea codului de conectare, vom face variabila db accesibilă din orice ruta:

A computer screen shot of a computer code

Description automatically generated

1. **Creați un template de aplicație client folosind Vue.js**

În terminal, selectați directorul de baza al proiectului, iar în terminal se vor realiza următoarele operații (referință: <https://github.com/vuejs/create-vue>)

A screenshot of a computer program

Description automatically generated

1. **Configurare Index.html din client și importare CSS Font Awesome**

În fișierul **index.html** din proiectul **Cashflow** create cu Vue, anterior, se vor realiza următoarele modificări:

**A screenshot of a computer program

Description automatically generated**

1. **Initializare App.vue ca un empty template**

Se va deschide fișierul **App.vue** din subdirectorul **src** corespunzator proiectului de client și se vor face următoarele modificări:

A screenshot of a computer

Description automatically generated

1. **Instalați Boostrap în aplicația client**

In terminalul fisierului client se va rula urmatoarea comanda:

* **npm install bootstrap@5**

1. **Importați Boostrap la nivelul aplicației**

În fișierul **main.js** din aplicația client, se vor face următoarele modificări pentru importarea bootstrap, și folosirea routerelor.

A screen shot of a computer program

Description automatically generated

1. **Incepeți configurarea endpoint-urilor pentru API: metoda get pentru users**

În fișierul **users.js**, din subdirectorul **routes** al serverului, se va realiza codul pentru endpoint-ul de get pentru users.

A screen shot of a computer

Description automatically generated

1. **(OPȚIONAL) Cod pentru documentația Swagger a metodei**

Codul va fi scris respectând indentarea, deasupra metodei.

Cod sursă: <https://pastebin.com/mVFFiHWY>

**A screenshot of a computer screen

Description automatically generated**

1. **Configurare endpoint get user dup****ă nume**

În același fișier **users.js**, se va configura endpoint-ul de get user după username.

A computer screen with text on it

Description automatically generated

1. **(OPȚIONAL) Cod pentru documentația Swagger a metodei**

Codul va fi scris respectând indentarea, deasupra metodei.

Cod sursă: <https://pastebin.com/3h8Rxs3m>

**A screenshot of a computer program

Description automatically generated**

1. **Instalați JWT (JSON Web Token)**

În terminalul aplicației de server, se va instala JWT, folosind următoarea comanda:

* **npm install jsonwebtoken**

1. **Configurare endpoint de add user**

În fișierul **users.js** din subdirectorul routes al aplicației server, se vor face următoarele modificări:

A screen shot of a computer code

Description automatically generated

Variabila **JWT** va fi folosită pentru verificarea logării, iar crypto va fi folosită pentru criptarea parolei userilor.

În continuare, se va începe prima parte din codul pentru endpoint-ul de addUser.

A screen shot of a computer code

Description automatically generated

Aici se vor realiza verificările de baza , precum și niste regex check-uri pentru caractere speciale, și un caracter uppercase.

Continuarea codului de endpoint, cu password hashing:

A screen shot of a computer program

Description automatically generated

1. **(OPȚIONAL) Cod pentru documentația Swagger a metodei**

Codul va fi scris respectând indentarea, deasupra metodei.

Deoarece metodele au o descriere din ce în ce mai complexă, iar acest proces nu este unul de o importanta majora, se recomanda copy-paste-ul deasupra metodei addUser de pe link-ul următor: <https://pastebin.com/1FUbRz5j>

1. **Configurare endpoint de login, și exportare router.**

În același fișier **users.js**, se vă adaugă codul pentru endpoint-ul care se vă ocupa cu logarea user-ului.

A screen shot of a computer code

Description automatically generated

Aici se va folosi și package-ul JWT, care va crea în local storage un token, folosind o cheie privată (în cazul nostru: “cashflow-key”) , cu o dată de expirare (în cazul nostru 24h). Signatura acestui token va fi verificată la fiecare accesare care necesită autentificare. Dacă această verificare eșuează, user-ul va fi trimis pe pagina de login, care o vom scrie ulterior.

1. **(OPȚIONAL) Cod pentru documentația Swagger a metodei**

Codul va fi scris respectând indentarea, deasupra metodei.

Deoarece metodele au o descriere din ce în ce mai complexă, iar acest proces nu este unul de o importanta majora, se recomanda copy-paste-ul deasupra metodei login de pe link-ul următor: <https://pastebin.com/MNbu8tG4>

1. **Instalare axios pentru apelare de endpoint-uri mai ușoară**

În terminalul aplicației de server, se vă apela următoarea comandă:

* **npm install axios**

1. **Adăugare componenta pentru register a utilizatorului**

În fișierul aplicației client, in subdirectorul **components**, se va crea un fișier **UserRegistration.vue.**

A screenshot of a computer program

Description automatically generated

1. **Template pentru UserRegistration.vue**În interiorul fișierului creat, se va configura partea de template, obligatorie pentru o componentă vue.

Cod sursă: <https://pastebin.com/P4EfMjgK>

**A screen shot of a computer program

Description automatically generated**

1. **Configurare route pentru pagina de register**

În subdirectorul **router**, din aplicația client, se vă accesa fișierul **index.js**. Aici se va crea o rută default, și o rută pentru pagina de register. De asemenea, se va scrie codul care verifică daca o pagina necesită autentificare, și va verifica existența token-ului.

A screen shot of a computer program

Description automatically generated

1. **Configurare router link in App.vue**

În fișierul **App.vue** din aplicația client, configurați router link-ul pentru a putea încarca pagina scrisa anterior.

A screen shot of a computer program

Description automatically generated

La o accesare a URL-ului, pagina va arata în felul următor:

A screenshot of a computer

Description automatically generated

1. **Configurare metode și apelare API cu axios**

Se va declara un element **script** după cel de template din **UserRegistration.vue**. Această parte de cod va fi responsabilă pentru apelarea endpoint-urilor din API.

A screen shot of a computer program

Description automatically generated

După realizarea acestui pas, pagina de register va funcționa corect, și user-ul va fi trimis către endpoint-ul de login, dacă înregistrarea de utilizator a reușit. Se poate observa cum metoda register, care este asignata butonului apelează endpoint-ul asociat din API, validarea register-ului realizandu-se în backend. Dacă validarea eșuează, se va afișa un toast message pe ecran cu motivul.

1. **(OPȚIONAL) Stilizare pentru intreg proiectul.**

În fișierul **main.css** , din subdirectorul **src/assets** al aplicației client, se vor introduce următoarele stiluri **CSS**. Stilurile conțin și componente ce le vom crea ulterior în aplicație. În urma aplicării stilurilor, pagina va arăta în felul următor:

A screen shot of a login form

Description automatically generated

Conținutul fișierului **main.css** se poate găsi la link-ul următor: <https://pastebin.com/3W5PNMHp>

1. **Creare pagină de login**

Precum pagina de register, vom crea un nou fișier UserLogin.vue în subdirectorul components din aplicația client.

A screenshot of a computer

Description automatically generated

1. **Template-ul din UserLogin.vue**

În pagina nou creata, se va adaugă componentă de template, necesară oricărei pagini Vue.

Cod sursă: <https://pastebin.com/YENRJ2sp>

A screen shot of a computer program

Description automatically generated

Această pagină este aproape identică cu cea de register, singurele diferențe fiind lipsa celor doua câmpuri necesare pentru validarea parolei, acum fiind nevoie doar de unul, și introducerea unui element cu router link către pagina de **register**. O altă diferență este numele metodei atașate la buton, în loc de register, aceasta se va numi **login**.

1. **Script-ul din UserLogin.vue**

Vom parcurge pașii invers fata de anterior, iar acum vom scrie codul care va utiliza API-ul pentru realizarea operațiunii de logare. Vom declara un nou bloc script în continuare la cel template.

A screen shot of a computer program

Description automatically generated

Se poate observa cum metoda login, atașată butonului apelează endpoint-ul asociat din API, iar toată logica de logare se realizează în backend. Daca logarea eșuează, un toast message va apărea pe ecranul utilizatorului.

1. **Configurare route pentru pagina de login**

În fișierul **index.js** al aplicației client, se vă crea rouțe-ul pentru pagina creata anterior:

**A screen shot of a computer program

Description automatically generated**

1. **Configurare router link in App.vue**

Precum pentru endpoint-ul de register, se va crea un router link în fișierul App.vue al aplicației client.

A screenshot of a computer program

Description automatically generated

După configurările de mai sus, la accesarea URL-ului corespunzător, pagina de login va fi funcțională.

A screenshot of a computer screen

Description automatically generated

1. **Creare endpoints pentru dashboard-ul aplicației**

Vom crea în subdirectorul **routes**, din aplicația server, un fișier **cashflowlog.js**, care va fi folosit pentru definirea endpoint-urilor utilizate în gestionarea cashflow-ului.

A screenshot of a computer program

Description automatically generated

1. **Endpoint pentru get la log-uri**

În fișierul nou creat, vom configura endpoint-ul de get:

A black screen with text on it

Description automatically generated

Vom selecta intrările în cashflow care au fost create de user-ul logat (transfer de la user-ul logat către alt user), împreună cu cele în care user-ul logat este setat că și destinatar (transfer de la alt user către user-ul logat).

Nota: Pentru fiecare endpoint din cashflow vom verifica JWT-ul pe care îl vom trimite într-un authorization header la apelarea acestuia.

1. **(OPȚIONAL) Cod pentru documentația Swagger a metodei**

Codul va fi scris respectând indentarea, deasupra metodei.

Se recomandă copy-paste-ul deasupra metodei get de pe link-ul următor: <https://pastebin.com/t7aLhqg9>

1. **Endpoint pentru insert la log-uri**

În același fișier, vom configura endpoint-ul de insert:

A screen shot of a computer program

Description automatically generated

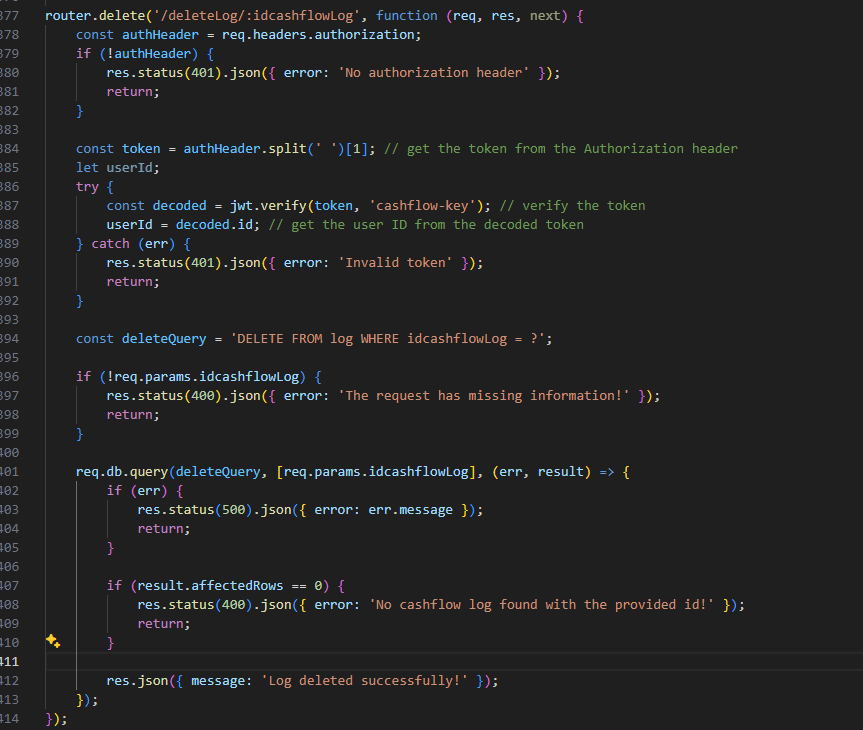
1. **(OPȚIONAL) Cod pentru documentația Swagger a metodei**

Codul va fi scris respectând indentarea, deasupra metodei.

Se recomandă copy-paste-ul deasupra metodei insert de pe link-ul următor: <https://pastebin.com/mXs2iN4d>

1. **Endpoint pentru delete la log-uri**

În același fișier, vom configura endpoint-ul de delete:



1. **(OPȚIONAL) Cod pentru documentația Swagger a metodei**

Codul va fi scris respectând indentarea, deasupra metodei.

Se recomandă copy-paste-ul deasupra metodei delete de pe link-ul următor: <https://pastebin.com/GnjnfD3Q>

1. **Configurare endpoint de update pentru log-uri**

În același fișier, vom configura endpoint-ul de update:

A screenshot of a computer program

Description automatically generated

1. **(OPȚIONAL) Cod pentru documentația Swagger a metodei**

Codul va fi scris respectând indentarea, deasupra metodei.

Se recomandă copy-paste-ul deasupra metodei update de pe link-ul următor: <https://pastebin.com/sS61Z4Ea>

1. **Export la fișierul configurat**

A black screen with white text

Description automatically generated La finalul fișierului **cashflowlog.js**, vom adaugă următoarea linie de cod pentru a realiza exportul fișierului.

1. **Configurare router pentru log-urile cashflow in aplicația server**

În fișierul **app.js** din aplicația server, vom adaugă următoarele linii pentru a putea avea acces la router-ul exportat.

A screen shot of a computer code

Description automatically generated

A screen shot of a computer program

Description automatically generated

Configurarea este identica cu cea de la router-ul pentru users.

După acest lucru, la un restart al aplicației server, interfață vă arata în felul următor:

A screenshot of a computer

Description automatically generated

Pentru testarea noilor endpoint-uri create, va fi nevoie de un JWT valid. Acest lucru este implementat pentru limitarea accesului la endpoint-urile critice, și a permite apelarea acestora doar de useri care au acces la acestea.

Pentru a obține un token valid, acesta se poate lua din local storage după o sesiune de login efectuată cu succes. Acesta se poate găsi la secțiunea **application**, **local storage**, după accesarea hoț key-ului **CTRL+Shift+I** în browser.

A screenshot of a computer

Description automatically generated