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

In urma parcurgerii acestei documentatii, vom crea un API, si o aplicatie client care va simula o aplicatie de cashflow management, avand o baza de date MySQL.

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

**A screenshot of a computer

Description automatically generated**

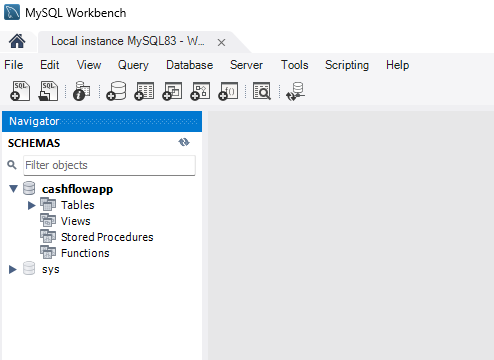
1. **Descarcati Node.js:** [**https://nodejs.org/en**](https://nodejs.org/en)

**A screenshot of a computer screen

Description automatically generated**

1. **Descarcati si configurati serverul local:** [**https://dev.mysql.com/downloads/mysql/**](https://dev.mysql.com/downloads/mysql/)

Configurati user-ul **root** si atribuiti o parola acestuia (va fi folosita ulterior la conectare). Dupa conectare, folositi **MySQL Workbench** pentru a crea o baza de date numita **cashflowapp**.



1. **Creati un director in care se va stoca aplicatia**

**A screenshot of a computer

Description automatically generated**

In interiorul directorului selectat, creati un subdirector numit server. Acesta se va deschide cu VS Code.

1. **Se va deschide un terminal in VS Code si se va selecta directorul server**

**A screenshot of a computer

Description automatically generated**

1. **Creati template-ul de server folosind Express.js**

In terminal se vor rula urmatoarele comenzi:

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

Aplicatia de server va putea fi pornita cu **npm start** si va rula la URL-ul: <http://localhost:3000/>.

A screenshot of a computer

Description automatically generatedDupa finalizarea pasilor, directorul server va arata precum in poza urmatoare:

1. **Instalati Swagger UI pentru documentatia/testarea API-ului**

In terminalul directorului server, rulati urmatoarele comenzi:

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

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

In terminalul directorului **server** se va rula urmatoarea comanda:

* **npm install cors**

In fisierul **app.js** din directorul **server**, se vor face urmatoarele modificari pentru a permite aplicatiei de client, care va fi realizata in pasii urmatori, sa acceseze endpoint-urile API-ului.

**A screen shot of a computer program

Description automatically generated**

1. **Configurati Swagger-UI**

In fisierul **app.js** din directorul **server**, se vor realiza urmatoarele 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/)

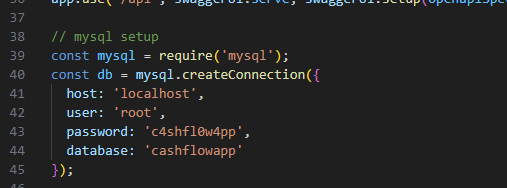
**A screen shot of a computer

Description automatically generated**

1. **Realizati legatura cu serverul MySQL**

In terminalul fisierului server rulati urmatoarea comanda:

* **npm install mysql**

In fisierul **app.js** din directorul **server**, se vor declara variabilele responsabile pentru interactiunea cu serverul **MySQL**.

1. **Conectati-va la baza de date, creati tabelele, si faceti conexiunea accesibila din toate rutele**

In fisierul **app.js** din drectorul **server**, realizati urmatoarele modificari. Urmatorul snippet de cod se va ocupa si de crearea tabelelor necesare, in caz ca acestea nu exista deja in baza de date **cashflowapp** din **MySQL.** Codul se va situa in continuarea setup-ului initial din pasul 10.

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

A screen shot of a computer program

Description automatically generated

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

A computer screen shot of a computer code

Description automatically generated

1. **Creati un template de aplicatie client folosind Vue.js**

In terminal, selectati directorul de baza al proiectului, iar in terminal se vor realiza urmatoarele operatii (referinta: <https://github.com/vuejs/create-vue>)

A screenshot of a computer program

Description automatically generated

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

In fisierul **index.html** din proiectul **Cashflow** create cu Vue, anterior, se vor realiza urmatoarele modificari:

**A screenshot of a computer program

Description automatically generated**

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

Se va deschide fisierul **App.vue** din subdirectorul **src** corespunzator proiectului de client si se vor face urmatoarele modificari:

A screenshot of a computer

Description automatically generated

1. **Instalati Boostrap in aplicatia client**

In terminalul fisierului client se va rula urmatoarea comanda:

* **npm install bootstrap@5**

1. **Importati Boostrap la nivelul aplicatiei**

In fisierul **main.js** din aplicatia client, se vor face urmatoarele modificari pentru importarea bootstrap, si folosirea routerelor.

A screen shot of a computer program

Description automatically generated

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

In fisierul **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. **(OPTIONAL) Cod pentru documentatia Swagger a metodei**

Codul va fi scris respectand indentarea, deasupra metodei.

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

**A screenshot of a computer screen

Description automatically generated**

1. **Configurare endpoint get user dupa nume**

In acelasi fisier **users.js**, se va configura endpoint-ul de get user dupa username.

A computer screen with text on it

Description automatically generated

1. **(OPTIONAL) Cod pentru documentatia Swagger a metodei**

Codul va fi scris respectand indentarea, deasupra metodei.

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

**A screenshot of a computer program

Description automatically generated**

1. **Instalati JWT (JSON Web Token)**

In terminalul aplicatiei de server, se va instala JWT, folosind urmatoarea comanda:

* **npm install jsonwebtoken**

1. **Configurare endpoint de add user**

In fisierul **users.js** din subdirectorul routes al aplicatiei server, se vor face urmatoarele modificari:

A screen shot of a computer code

Description automatically generated

Variabila **JWT** va fi folosita pentru verificarea logarii, iar **crypto** va fi folosita pentru criptarea parolei userilor.

In continuare, se va incepe prima parte din codul pentru endpoint-ul de addUser.

A screen shot of a computer code

Description automatically generated

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

Continuarea codului de endpoint, cu password hashing:

A screen shot of a computer program

Description automatically generated

1. **(OPTIONAL) Cod pentru documentatia Swagger a metodei**

Codul va fi scris respectand indentarea, deasupra metodei.

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

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

In acelasi fisier users.js, se va adauga codul pentru endpoint-ul care se va ocupa cu logarea user-ului.

A screen shot of a computer code

Description automatically generated

Aici se va folosi si package-ul JWT, care va crea in local storage un token, folosind o cheie privata (in cazul nostru: “cashflow-key”) , cu o data de expirare (in cazul nostrum 24h). Signatura acestui token va fi verificata la fiecare accesare care necesita autentificare. Daca aceasta verificare esueaza, user-ul va fi trimis pe pagina de login, care o vom scrie ulterior.

1. **(OPTIONAL) Cod pentru documentatia Swagger a metodei**

Codul va fi scris respectand indentarea, deasupra metodei.

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

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

In terminalul aplicatiei de server, se va apela urmatoarea comanda:

* **npm install axios**

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

In fisierul aplicatiei client, in subdirectorul **components**, se va crea un fisier **UserRegistration.vue.**

A screenshot of a computer program

Description automatically generated

1. **Template pentru UserRegistration.vue**In interiorul fisierului creat, se va configura partea de template, obligatorie pentru o componenta vue.

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

**A screen shot of a computer program

Description automatically generated**

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

In subdirectorul **router**, din aplicatia client, se va accesa fisierul **index.js**. Aici se va crea o ruta default, si o ruta pentru pagina de register. De asemenea, se va scrie codul care verifica daca o pagina necesita autentificare, si va verifica existenta token-ului.

A screen shot of a computer program

Description automatically generated

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

In fisierul **App.vue** din aplicatia client, configurati router link-ul pentru a putea incarca pagina scrisa anterior.

A screen shot of a computer program

Description automatically generated

La o accesare a URL-ului, pagina va arata in felul urmator:

A screenshot of a computer

Description automatically generated

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

Se va declara un element **script** dupa cel de template din **UserRegistration.vue**. Aceasta parte de cod va fi responsabila pentru apelarea endpoint-urilor din API.

A screen shot of a computer program

Description automatically generated

Dupa realizarea acestui pas, pagina de register va functiona corect, si user-ul va fi trimis catre endpoint-ul de login, daca inregistrarea de utilizator a reusit. Se poate observa cum metoda register, care este asignata butonului apeleaza endpoint-ul asociat din API, validarea register-ului realizandu-se in backend. Daca validarea esueaza, se va afisa un toast message pe ecran cu motivul.

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

In fisierul **main.css** , din subdirectorul **src/assets** al aplicatiei client, se vor introduce urmatoarele stiluri CSS. Stilurile contin si componente ce le vom crea ulterior in aplicatie. In urma aplicarii stilurilor, pagina va arata in felul urmator:

A screen shot of a login form

Description automatically generated

Continutul fisierului **main.css** se poate gasi la link-ul urmator: <https://pastebin.com/3W5PNMHp>

1. **Creare pagina de login**

Precum pagina de register, vom crea un nou fisier UserLogin.vue in subdirectorul components din aplicatia client.

A screenshot of a computer

Description automatically generated

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

In pagina nou creata, se va adauga componenta de template, necesara oricarei pagini Vue.

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

A screen shot of a computer program

Description automatically generated

Aceasta pagina este aproape identica cu cea de register, singurele diferente fiind lipsa celor doua campuri necesare pentru validarea parolei, acum fiind nevoie doar de unul, si introducerea unui element cu router link catre pagina de **register**. O alta diferenta este numele metodei atasate la buton, in loc de register, aceasta se va numi **login**.

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

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

A screen shot of a computer program

Description automatically generated

Se poate observa cum metoda login, atasata butonului apeleaza endpoint-ul asociat din API, iar toata logica de logare se realizeaza in backend. Daca logarea esueaza, un toast message va aparea pe ecranul utilizatorului.

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

In fisierul **index.js** al aplicatiei client, se va crea route-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 in fisierul **App.vue** al aplicatiei client.

A screenshot of a computer program

Description automatically generated

Dupa configurarile de mai sus, la accesarea URL-ului corespunzator, pagina de login va fi functionala.

A screenshot of a computer screen

Description automatically generated

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

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

A screenshot of a computer program

Description automatically generated

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

In fisierul nou creat, vom configura endpoint-ul de get:

A black screen with text on it

Description automatically generated

Vom selecta intrarile in cashflow care au fost create de user-ul logat (transfer de la user-ul logat catre alt user), impreuna cu cele in care user-ul logat este setat ca si destinatar (transfer de la alt user catre user-ul logat).

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

1. **(OPTIONAL) Cod pentru documentatia Swagger a metodei**

Codul va fi scris respectand indentarea, deasupra metodei.

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

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

In acelasi fisier, vom configura endpoint-ul de insert:

A screen shot of a computer program

Description automatically generated

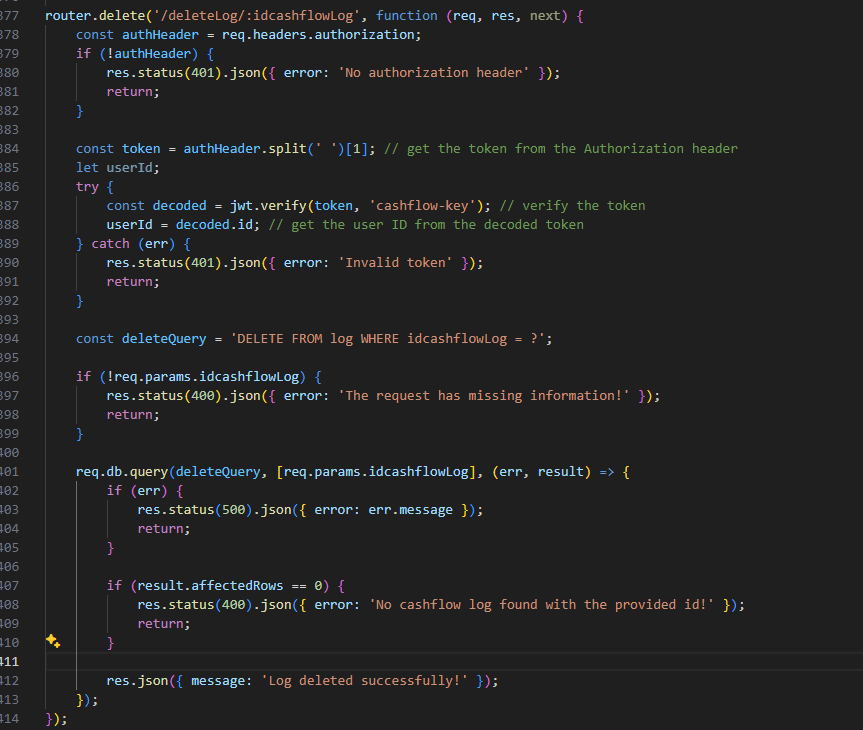
1. **(OPTIONAL) Cod pentru documentatia Swagger a metodei**

Codul va fi scris respectand indentarea, deasupra metodei.

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

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

In acelasi fisier, vom configura endpoint-ul de delete:



1. **(OPTIONAL) Cod pentru documentatia Swagger a metodei**

Codul va fi scris respectand indentarea, deasupra metodei.

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

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

In acelasi fisier, vom configura endpoint-ul de update:

A screenshot of a computer program

Description automatically generated

1. **(OPTIONAL) Cod pentru documentatia Swagger a metodei**

Codul va fi scris respectand indentarea, deasupra metodei.

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

1. **Export la fisierul configurat**

A black screen with white text

Description automatically generatedLa finalul fisierului **cashflowlog.js**, vom adauga urmatoarea linie de cod pentru a realiza exportul fisierului.

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

In fisierul app.js din aplicatia server, vom adauga urmatoarele 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.

Dupa acest lucru, la un restart al aplicatiei server, interfata va arata in felul urmator:

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, si a permite apelarea acestora doar de useri care au acces la acestea.

Pentru a obtine un token valid, acesta se poate lua din local storage dupa o sesiune efectuata de login. Acesta se poate gasi la sectiunea application, local storage, dupa accesarea hot key-ului CTRL+Shift+I in browser.

A screenshot of a computer

Description automatically generated