**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**

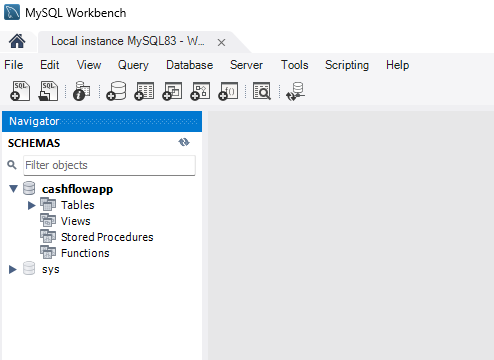
1. **Descarcați Node.js:** [**https://nodejs.org/en**](https://nodejs.org/en)

**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/)

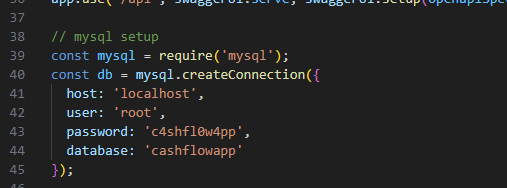
**A screen shot of a computer

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**

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

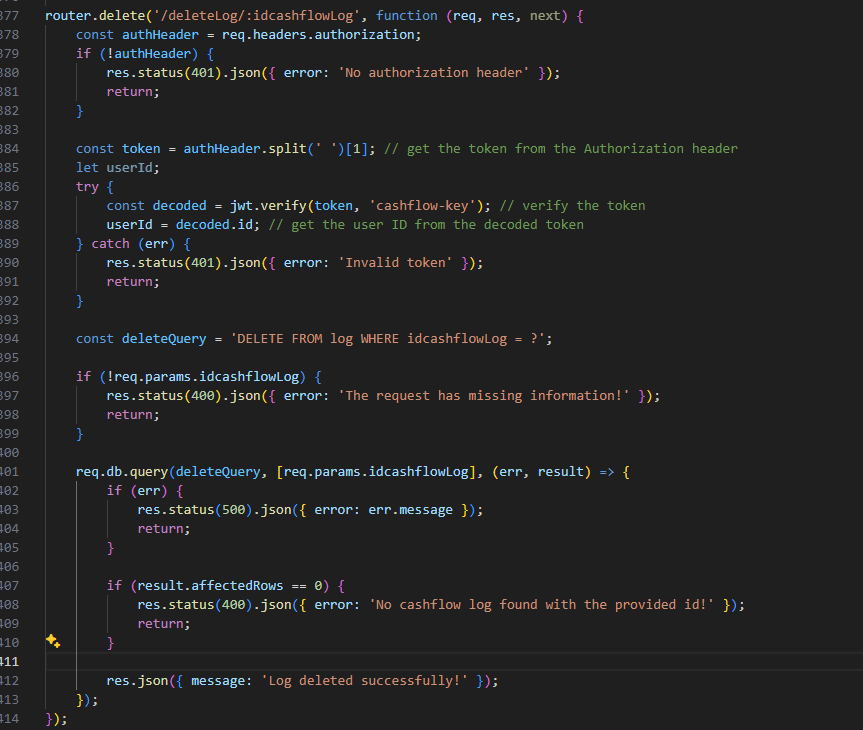
1. **(OPTIONAL) Cod pentru documentatia Swagger a metodei**

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