A blue and white logo with wings

Description automatically generated

PROIECT

AFACERI ELECTRONICE

EMILIAN CÎRCIUMARU

BDSA

AN 2 – DECEMBRIE 2024

**GitHub Link:** [**https://github.com/emilcirciumaru/proiect\_ae2**](https://github.com/emilcirciumaru/proiect_ae2)

**Prezentare Video Link :** [**https://www.youtube.com/watch?v=VZ98gYcyw2I**](https://www.youtube.com/watch?v=VZ98gYcyw2I)

**Prezentare Generală**

Această aplicație web are frontend bazat pe Angular, conectat la un backend Node.js și o bază de date MySQL. Aplicația permite gestionarea produselor, filtrarea, sortarea și integrarea plăților. Este structurată modular și utilizează API-uri REST pentru comunicarea între backend și frontend.

**Funcționalități Principale**

**1. Afișarea Produselor și Filtrare:**

* Afișează o listă de produse preluate din baza de date MySQL.
* Permite filtrarea produselor pe categorii.
* Oferă funcționalitate de sortare (ascendent/descendent).

A screenshot of a jewelry

Description automatically generated

**2. Gestionarea Coșului:**

* Adăugare produse în coș.
* Vizualizarea produselor din coș împreună cu costul total.

A screenshot of a jewelry store

Description automatically generated

**3. API REST:**

* Gestionarea categoriilor și produselor printr-un API REST.
* API-urile permit preluarea datelor pe baza categoriilor și a altor criterii.

**4. Integrare cu Stripe:**

* Integrare Stripe pentru procesarea plăților.

A screenshot of a computer

Description automatically generated

**Folosire Fakestore API pentru a popular baza de date.**

Această aplicație utilizează API-ul FakeStore pentru a popula baza de date MySQL cu produse și categorii. FakeStore API este un serviciu extern care oferă date de test, precum produse, prețuri și categorii, utilizate pentru a simula o aplicație de comerț electronic.

În cadrul proiectului, este implementat un script Node.js (populate.js) care:

* Accesează FakeStore API folosind axios.
* Preia listele de produse și categorii.
* Salvează aceste date în baza de date MySQL folosind modelele Sequelize.

**Integrarea în aplicație**

1. **Accesarea FakeStore API**:
   * Datele sunt obținute folosind biblioteca axios pentru a trimite cereri HTTP GET către endpoint-urile FakeStore API.
2. **Salvarea în baza de date**:
   * Datele obținute sunt prelucrate și salvate utilizând modelele Sequelize definite pentru tabelele products și categories.

**Frontend**

**Tehnologii Utilizate**

* **Framework:** Angular
* **UI Framework:** Angular Material
* **Bibliotecă HTTP:** HttpClient

**Funcționalități Frontend**

1. **Listă Dinamică de Produse:**
   * Preia produsele din API-ul backend.
   * Permite filtrarea pe categorii și sortarea pe preț (ascendent/descendent).
2. **Interfață Interactivă:**
   * Design complet responsiv utilizând componentele Angular Material.
   * Sistem de grilă dinamic pentru afișarea produselor.

A screenshot of a computer

Description automatically generated

1. **Gestionarea Coșului:**
   * Adăugare/ștergere produse.
   * Calcularea dinamică a costului total.

A screenshot of a computer

Description automatically generated

**Utilizare**

1. **Vizualizare Produse:**
   * Navigați pe pagina principală pentru a vizualiza și filtra produsele.
2. **Adăugare în Coș:**
   * Faceți clic pe un produs pentru a-l adăuga în coș.
3. **Plată:**
   * Vizualizați coșul și inițiați procesul de plată utilizând Stripe.

**Structura Frontend**

graphql

Copy code

src/

├── app/

│ ├── components/

│ │ ├── header/ # Componenta Header pentru navigare

│ │ └── footer/ # Componenta Footer (dacă este implementată)

│ ├── models/ # Modele TypeScript pentru structuri de date

│ ├── pages/

│ │ ├── home/ # Pagină principală cu produse

│ │ └── cart/ # Pagină pentru gestionarea coșului

│ ├── services/ # Servicii Angular pentru comunicarea cu API-ul

│ └── app.module.ts # Definirea modulului Angular

└── environments/

├── environment.ts # Configurare pentru mediu de dezvoltare

└── environment.prod.ts # Configurare pentru mediu de producție

**Backend**

**Tehnologii Utilizate**

* **Framework:** Node.js
* **Bază de date:** MySQL
* **ORM:** Sequelize

**Funcționalități Backend**

1. **API REST:**
   * Operații CRUD pentru produse și categorii.
   * Filtrare dinamică a produselor pe categorii.
2. **Gestionarea Bazei de Date:**
   * Stochează informații despre produse și categorii în MySQL.
   * Suportă sincronizare și migrare utilizând Sequelize.
3. **Procesare Plăți:**
   * Integrare API Stripe pentru plăți.

**Structura Backend**

graphql

Copy code

server/

├── models/ # Modele Sequelize pentru entitățile bazei de date

│ ├── Category.js # Model pentru categorii

│ └── Product.js # Model pentru produse

├── db.js # Configurarea bazei de date Sequelize

├── routes/ # Rute pentru API-uri

│ ├── products.js # Rute API pentru operațiuni cu produse

│ └── categories.js # Rute API pentru operațiuni cu categorii

├── server.js # Fişier principal server

└── config/ # Fişiere de configurare (ex: chei Stripe)

**Instalare și Configurare**

**Cerințe Preliminare**

* Node.js și npm
* Server MySQL
* Angular CLI
* Cont Stripe pentru integrarea plăților

**Configurare Backend**

1. Clonați repository-ul:

bash

git clone https://github.com/username/repository.git

cd repository/server

1. Instalați dependențele:

bash

npm install

1. Configurați baza de date:
   * Creați un fișier .env în directorul serverului cu următorul conținut:

makefile

Copy code

DB\_HOST=localhost

DB\_USER=root

DB\_PASSWORD=parola

DB\_NAME=nume\_baza\_date

STRIPE\_SECRET\_KEY=cheie-secreta-stripe

1. Rulați serverul:

bash

node server.js

1. Verificați funcționarea serverului: Accesați http://localhost:4242/categories în browser sau Postman.

**Configurare Frontend**

1. Navigați în directorul frontend:

bash

cd ../frontend

1. Instalați dependențele:

bash

npm install

1. Actualizați URL-ul backend în src/environments/environment.ts:

typescript

export const environment = {

production: false,

apiUrl: 'http://localhost:4242'

};

1. Rulați aplicația Angular:

bash

ng serve

1. Deschideți aplicația în browser: Accesați http://localhost:4200.

**Endpoint-uri**

1. **Categorii**
   * **GET** /categories: Preia toate categoriile.
   * **Răspuns:**

json

[

{ "id": 1, "name": "Electronice", "description": "Gadget-uri electronice" },

{ "id": 2, "name": "Modă", "description": "Haine și accesorii" }

]

1. **Produse**
   * **GET** /products/:categoryId: Preia produsele pe baza ID-ului categoriei.
   * **GET** /product/:id: Preia un produs specific după ID.
2. **Checkout**
   * **POST** /checkout: Procesează o plată utilizând Stripe.
   * **Exemplu corp cerere:**

json

{

"items": [

{ "name": "Produs 1", "price": 100, "quantity": 1, "product": "imagine\_url" }

]

}