

**Proiectarea și implementarea unei platforme Web pentru produse Handmade**

**Inginerie software si aplicatii in comunicatii de date**

**Profesor coordonator:**

**Șef lucr.dr.ing. Cătălin Bogdan Ciobanu**

**Participanți:**

**Petre Mihaela – Alexandra**

**Olteanu Raluca**

**Hermenean Vlad**

**BRAȘOV, 2024**

**Departamentul Electronică si Calculatoare**

**Programul de studii: Tehnologii și Sisteme de Telecomunicații**

Proiectarea și implementarea unei platforme Web pentru produse Handmade

**Profesor coordonator:**

Șef lucr.dr.ing. Cătălin Bogdan Ciobanu

**Cuprins**

Cuprins

[1. Introducere 4](#_Toc166807974)

[1.1 Concepte 4](#_Toc166807975)

[1.1.1 Pagina Web 4](#_Toc166807976)

[1.1.2 Handmade 4](#_Toc166807977)

[1.2 Istoric 5](#_Toc166807978)

[2. Obiectivele si scopul proiectului 7](#_Toc166807979)

[2.1 OBIECTIVE: 7](#_Toc166807980)

[2.2 Scopul: 7](#_Toc166807981)

[3. Descrierea Paginii WEB 8](#_Toc166807982)

[4. Mini Tutorial de utilizare 10](#_Toc166807983)

[5 Tehnologii utilizate 14](#_Toc166807984)

[5.1 Frontend: Angular 14](#_Toc166807985)

[5.2 Visual Studio Code 2022: 15](#_Toc166807986)

[5.3 MySQL: 16](#_Toc166807987)

[6 GitHub 17](#_Toc166807988)

[6.1 Avantaje 18](#_Toc166807989)

[7 Testare 20](#_Toc166807990)

[8 Management 20](#_Toc166807991)

[9 Lecții învățate 22](#_Toc166807992)

[10 Concluzii 22](#_Toc166807993)

[11 Codul sursă 23](#_Toc166807994)

[11.1 Autentificare: 23](#_Toc166807995)

[11.2 Coș de cumpărături: 29](#_Toc166807996)

[11.3 Produse: 32](#_Toc166807997)

[11.4 Dashboard: 36](#_Toc166807998)

[12 Bibliografie 40](#_Toc166807999)

# Introducere

## 1.1 Concepte

În cadrul acestui proiect, este esențial să stabilim fundamentele teoretice pentru a înțelege și a dezvolta în mod corespunzător două aspecte cheie: paginile web și produsele handmade.

### Pagina Web

Pagina web reprezintă un document electronic accesibil prin intermediul unui browser web, destinat să furnizeze informații și servicii care să le permită utilizatorilor interacțiunea cu resursele online.

Crearea unor astfel de pagini necesită cunoștințe în domenii asociate dezvoltării web cu caracteristici cheie precum:

* HTML (Hyper Text Markup Language) – acest markup definește structura și conținutul paginii Web, asfel încât, prin intermediul elementelor HTML, se organizează sau formatează textul, se introduc imagini sau video și se creează legături către alte resurse online.
* CSS (Cascading Style Sheets) – este utilizat pentru a putea controla aspectul visual al paginilor web, deoarece prin intermediul acestuia se definesc proprietăți precum culoarea, dimensiunea și fontul textului, dar și aspectul imaginilor sau alte caracteristici de design.
* JavaScript – acest limbaj de programare este utilizat pentru a agăuga interactivitate și funcționalități dinamice paginilor web de exemplu aspecte vizuale și validarea datelor introduse de utilizatori.

### Handmade

Produsele handmade sunt obiecte realizate manual, cu atenție și expertiză de către un artizan. Aceasta denumire provine de la modul lor specific de fabricare ilustrat de folosirea instrumentelor artizanale și a mâinilor, lucru care le deosebește de cele realizate în masă, într-o fabrică, sau într-un mediu industrial.



Figura 1. Reprezentarea activității Handmade [1]

Acest concept este strâns legat de tradiție, calitate și individualitate, fiind apreciat pentru autenticitatea și unicatul fiecărui obiect.

## 1.2 Istoric

În ultimele decenii, industria produselor handmade a cunoscut o creștere semnificativă, ilustrată în mare măsură de creșterea interesului consumatorilor pentru produsele autentice, unice și fabricate cu atenție la detalii.

Odată cu avansul tehnologic, producția în masă și automatizarea au devenit cele mai întâlnite moduri de fabricare. Astfel, dominând multe aspecte ale economiei moderne, există o reînviere a aprecierii pentru meșteșugurile tradiționale și pentru produsele care îmbină calitatea și creativitatea.

În acest context, internetul și platformele de e-commerce au jucat un rol important în transformarea și creșterea industriei handmade. Evoluția tehnologică a permis micilor întreprinzători și artizanilor să-și expună produsele în întreaga lume, să-și construiască branduri și să-și extindă afacerile fără a fi limitați de spațiul fizic al unui magazin tradițional.

Cu toate acestea, creșterea rapidă și accesibilitatea piețelor online au adus și noi provocări, precum concurența intensificată și nevoia de a oferi o experiență de cumpărături online aflată la înălțimea așteptărilor consumatorilor moderni.

Comparativ cu anii anteriori, se observă o creștere continuă a numărului de platforme web dedicate produselor handmade și a diversității acestora. De la magazine online generale care includ și secțiuni dedicate produselor handmade, până la platforme specializate care pun accentul pe artizanat și creativitate, oferta devenind tot mai variată și mai adaptată nevoilor și preferințelor creatorilor, cât și ale consumatorilor.

În România, industria produselor handmade a avut o evoluție notabilă în ultimii ani, reflectată atât în creșterea numărului de artizani și mici întreprinzători, cât și în diversificarea produselor și a canalelor de vânzare.

Fiind bogată în tradiții meșteșugărești și în talente creative, internetul este o oportunitate pentru promovarea și comercializarea acestor produse către publicul larg, atât în țară, cât și în afara granițelor.

Astfel, în această eră digitală, paginile web au devenit un instrument esențial pentru micii întreprinzători în promovarea și vânzarea produselor handmade. O pagină web bine concepută nu numai că poate oferi o platformă eficientă pentru prezentarea și comercializarea produselor, dar poate crea o experiență de cumpărături online plăcută și convenabilă pentru clienți, pornind de la design-ul estetic și intuitiv, la funcționalități avansate precum căutarea și filtrarea produselor.

# Obiectivele si scopul proiectului

## OBIECTIVE:

1.Crearea unei interfețe intuitive: unul dintre obiectivele principale ale proiectului este de a dezvolta o interfață de utilizator intuitivă și ușor de folosit, care să ofere o experiență plăcută și eficientă utilizatorilor, indiferent de nivelul lor de experiență în utilizarea platformelor online.

2.Asigurarea securității datelor: un alt obiectiv important al proiectului este de a asigura securitatea și confidențialitatea datelor utilizatorilor. Prin implementarea unor măsuri robuste de securitate și criptare a datelor sensibile, se urmărește protejarea informațiilor personale și financiare ale utilizatorilor.

3.Facilitarea gestionării produselor și comenzilor: platforma trebuie să ofere un set complet de funcționalități pentru gestionarea produselor și comenzilor. Aceasta include permiterea artizanilor să își încarce și să își gestioneze produsele pe platformă, să permită gestionarea comenzilor primite și comunicarea eficientă cu clienții.

4.Optimizarea experienței de cumpărare: un alt obiectiv important al proiectului este de a optimiza experiența de cumpărare pentru clienți. Acest lucru implică oferirea unei interfețe de utilizator plăcute, furnizarea de informații detaliate despre produse și facilizarea procesului de căutare, selectare și achiziționare a produselor.

5.Dezvoltarea unei comunități active: proiectul își propune să dezvolte o comunitate activă de artizani și clienți. Prin facilitarea interacțiunilor și schimbului de experiențe între membrii comunității, se urmărește crearea unei atmosfere prietenoase și suportive pentru toți utilizatorii platformei.

Acestea sunt obiectivele principale ale proiectului, care ghidează dezvoltarea și implementarea platformei de vânzare a produselor handmade. Prin atingerea acestor obiective, se urmărește oferirea unei experiențe de utilizare de calitate și creșterea valorii adăugate pentru toți utilizatorii implicați.

## Scopul:

Scopul acestui proiect este de a crea o platformă online dedicată vânzării de produse handmade, care să ofere un mediu sigur și eficient pentru artizani să își prezinte și să își vândă creațiile, iar clienților să găsească și să achiziționeze produse unice și de calitate. Platforma își propune să faciliteze interacțiunea între artizani și clienți, susținând comunitatea de produse handmade și promovând talentul și creativitatea acestora.

# Descrierea Paginii WEB

Proiectul intitulat “Proiectarea și implementarea unei platforme Web pentru produse Handmade” reprezintă o pagina WEB ce ilustrează un Shop Online pus la dispoziția iubitorilor de produse handmade, ușurând achiziționarea acestora.

Pagina Web are o secțiune de înregistrare pentru noii utilizatori, cu diferite câmpuri precum “First Name”, “Last Name”, “Password”, destinate completării cu datele dorite de către aceștia. După înregistrare se poate observa mesajul “Registration Succesful” care ne arată că înregistarea a fost făcută cu success.

Următoarea pagină este cea de autentificare, unde utilizatori își vor introduce datele create anterior pentru accesul pe pagina principală a shop-ului. După efectuarea acestora, se va afișa mesajul “Login Succesful”, care confirmă autentificarea.

Acest process de autentificare se realizeaza cu ajutorul bazei de date unde sunt stocate datele stabilite de utilizatori.

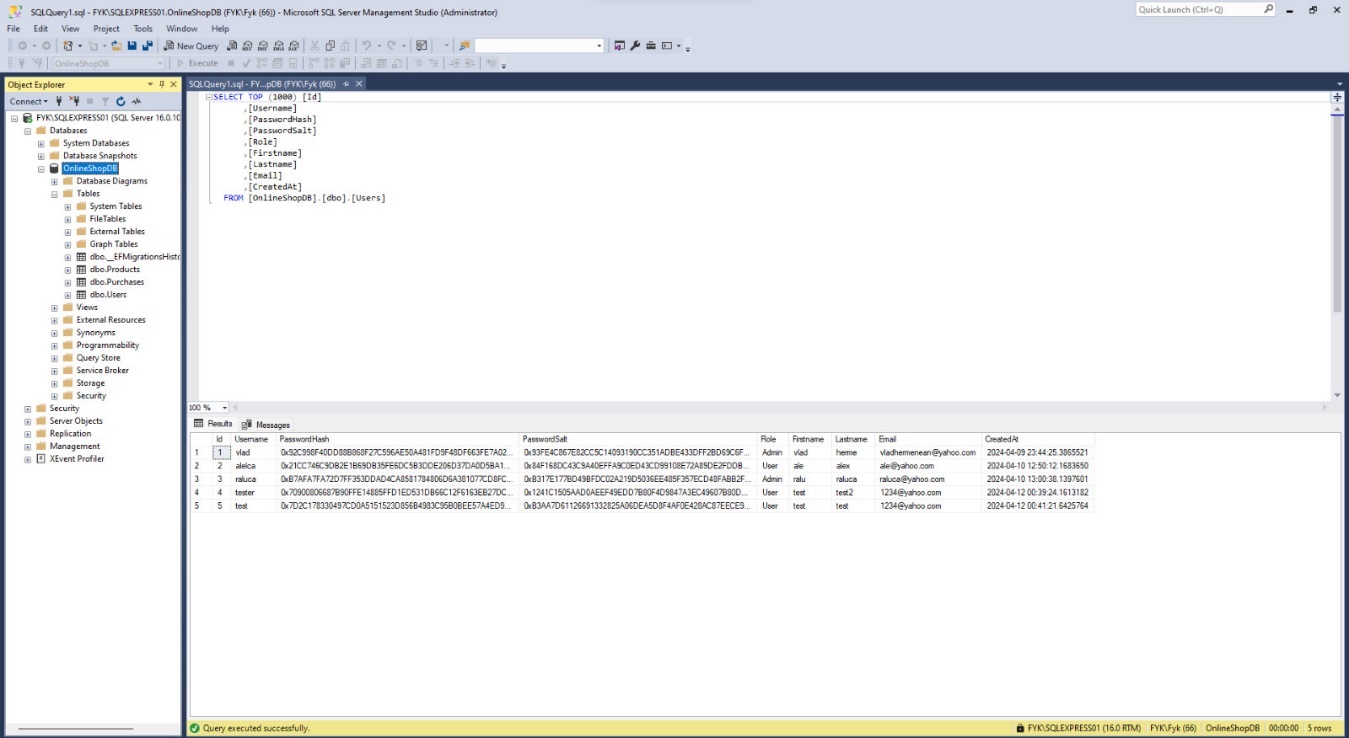


Figura 2. Baza de date a utilizatorilor

Odată cu accesul în pagina principală, utilizatorii vor putea vedea produsele publicate, o descriere succintă a acestora, cât și prețul lor.

După vizualizarea produselor, dorința de a le achiziționa se poate realiza prin adăugarea produselor în coșul de cumpărături.

De asemenea, pagina oferă posibilitatea de a adăuga produse care pot fi expuse publicului larg. Acest lucru se realizeaza prin baza de date.

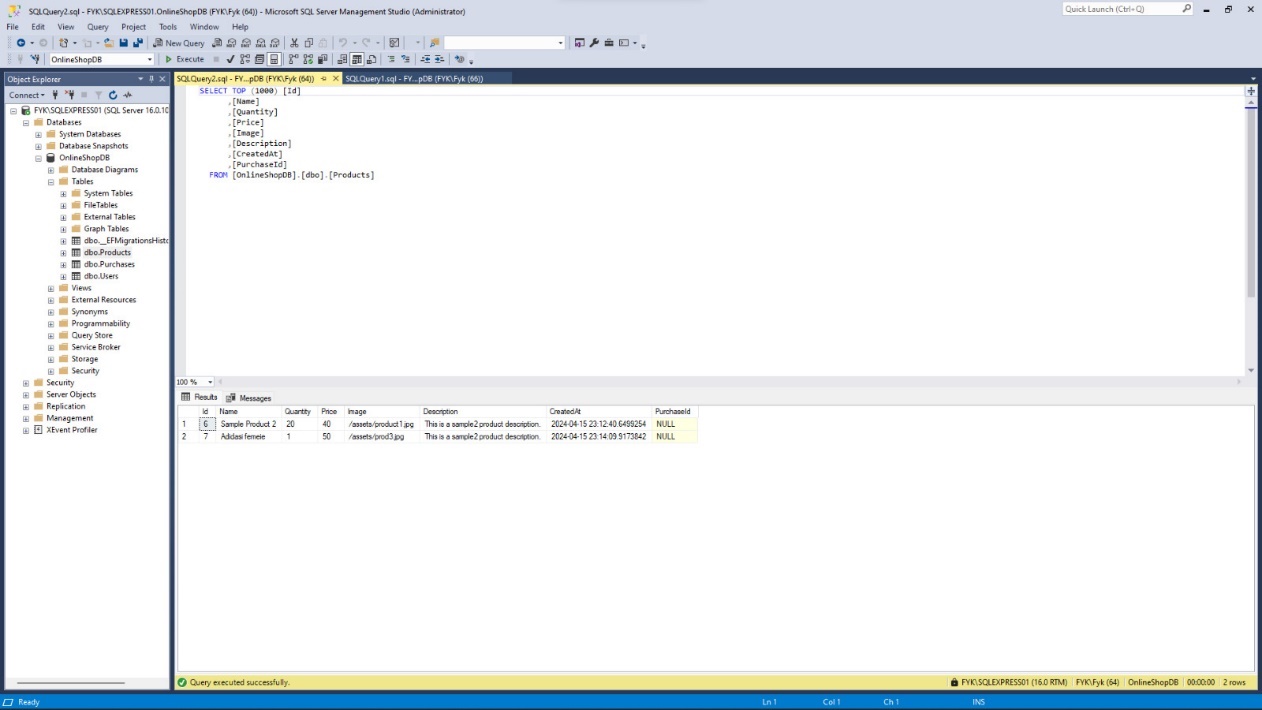


Figura 3. Baza de date produs

# Mini Tutorial de utilizare

Pentru utilizarea paginii WEB, vom accesa link-ul “http://localhost:4200/” generat în terminal.

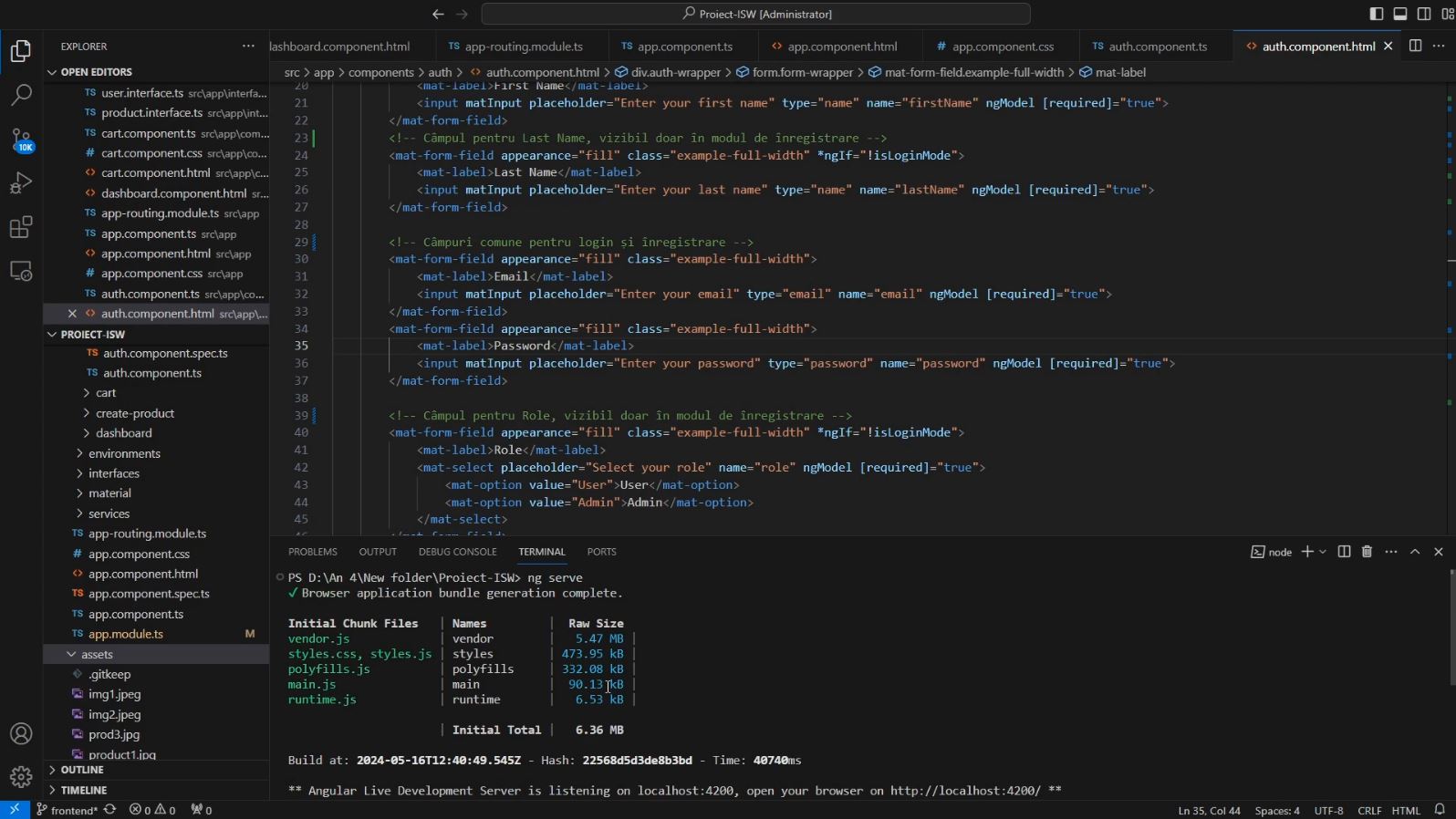


Figura 4. Terminalul codului

Acesta ne va redirectiona către browser unde se va afișa pagina de înregistrare, destinată noilor utilizatori, urmată de completarea datelor în rubricile speciale și apăsarea butonului de “SIGN UP” Pentru utilizatorii deja existenți se poate apăsa pe textul albastru “Login” din partea de jos a ferestrei “Create Account”.

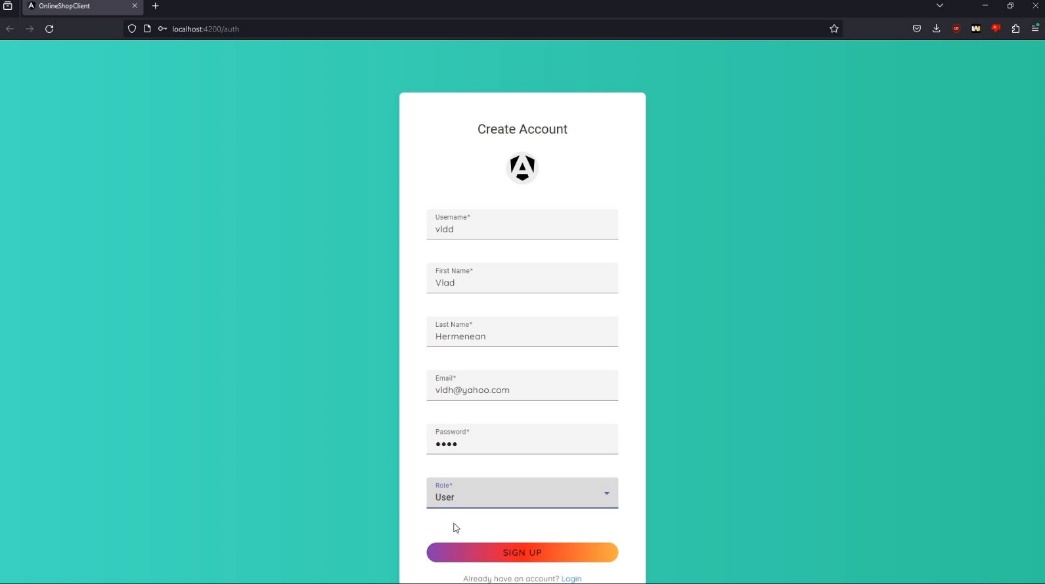


Figura 5. Pagina de înregistrare

După ce contul a fost creat, se va deschide pagina de autentificare, unde va fi necesar ca utilizatorii să își introducă datele create în secțiunea de înregistrare.

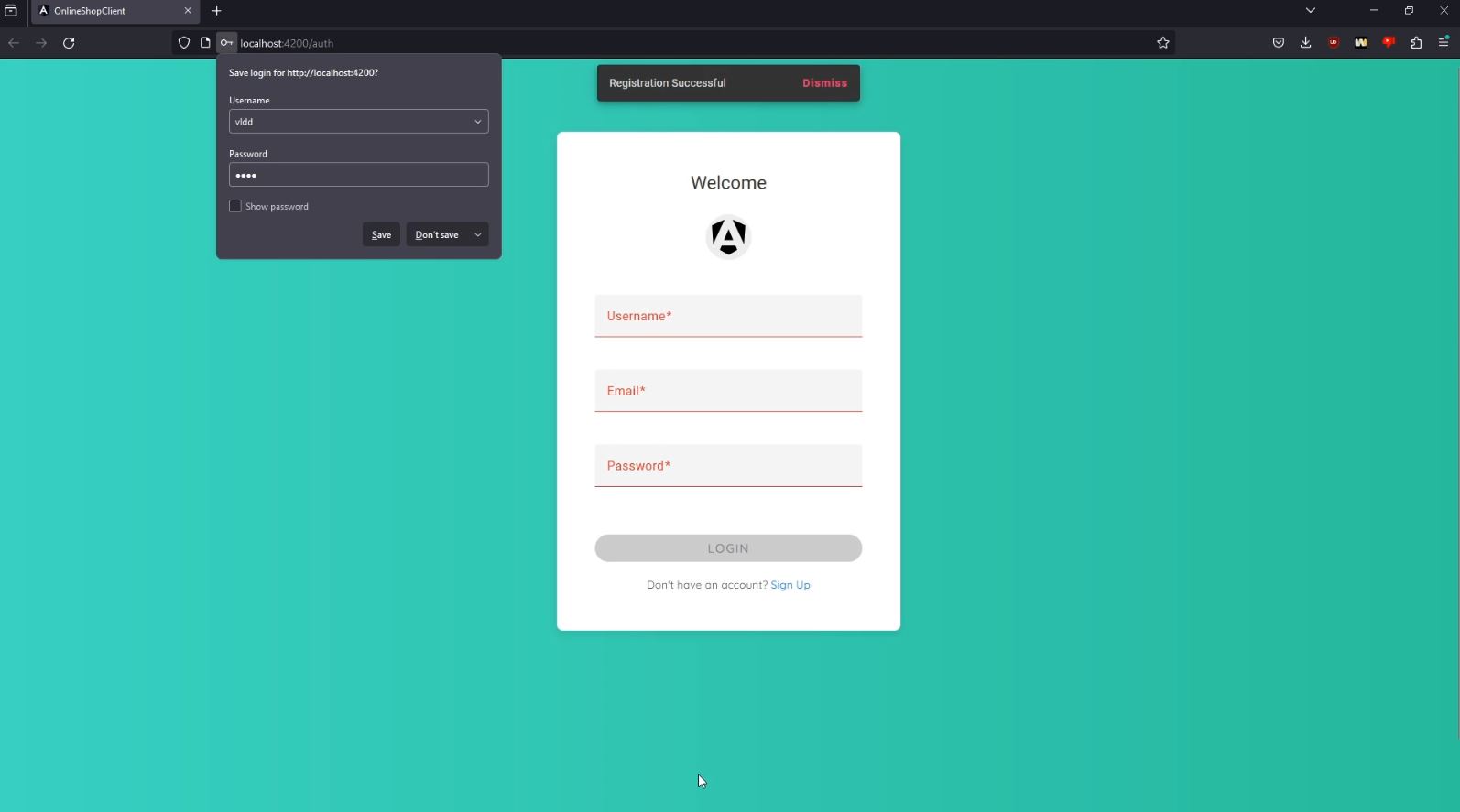


Figura 6. Pagina de autentificare

Pentru a duce la capăt acest pas, este necesară apăsarea butonului “LOGIN”. Astfel, vom fi redirecționați către pagina principală a acestui magazin online.

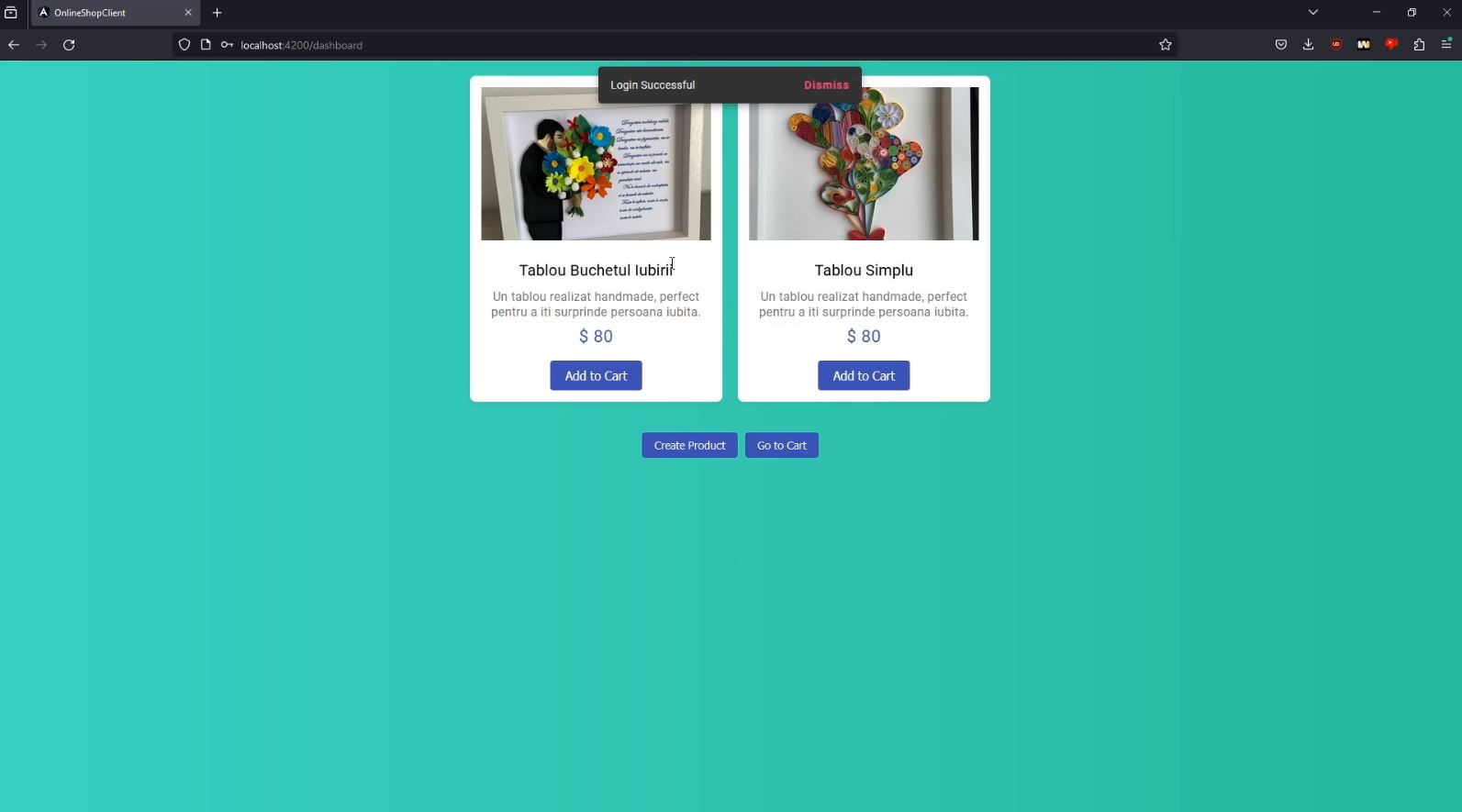


Figura 7. Pagina principală a shop-ului

După vizualizarea produselor, le putem adăuga pe cele dorite în coșul de cumpărături prin apăsarea butonului “Add to Cart”, însă este necesar ca numărul de apăsări al acestui buton să fie egal cu numărul de cantități dorite ale produsului respectiv.

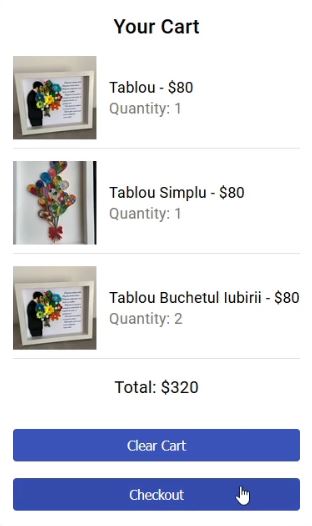


Figura 8. Coșul de cumpărături

Pentru a naviga in coșul de cumpărături trebuie apăsat butonul “Go to Cart”, care ne va afișa coșul cu produsele selectate de noi, cu optiunea de a goli coșul de cumpărături prin folosirea butonului “Clear Cart” și opțiunea de achiziționare a produselor din coșul de cumpărături prin apăsarea butonului “Checkout”, după care coșul se va goli automat.

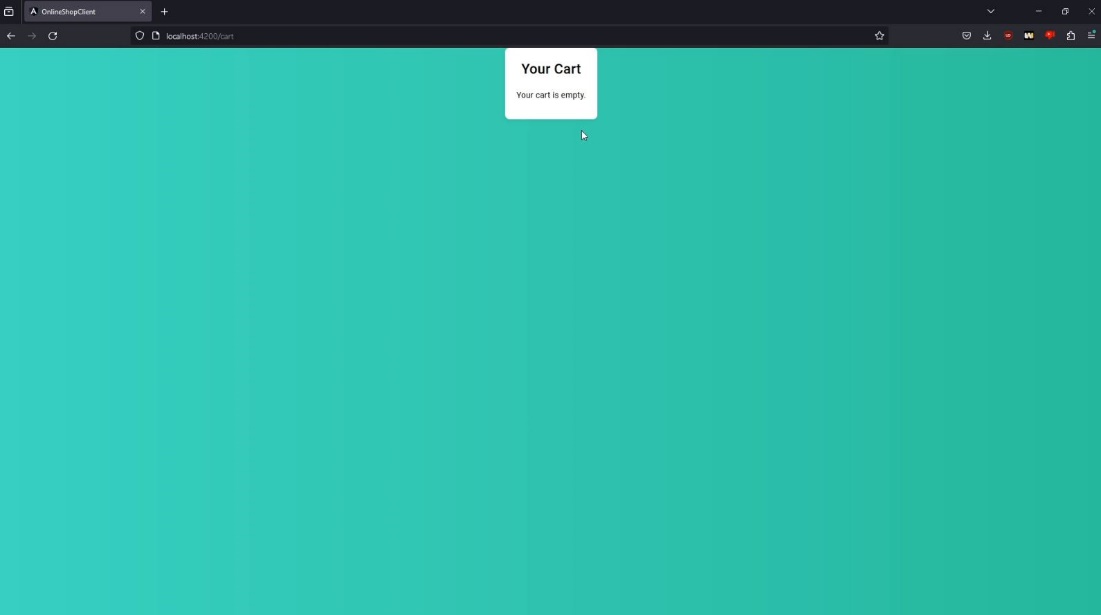


Figura 9. Finalizare comandă

Pentru adăugarea unui nou produs, în pagina principală se va apăsa butonul “Create Product” care ne va deschide fereastra “Create Product” unde se stabilește prețul, cantitatea, numele produsului, descrierea și poza care se va afișa, urmată de folosirea butonului “Create”.

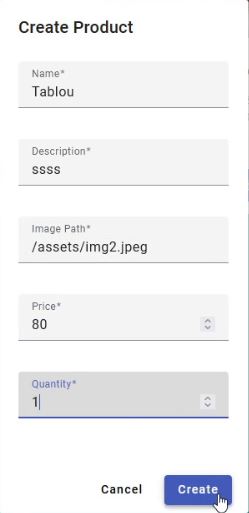


Figura 10. Adăugare produs

Pentru a putea vedea noul produs adăugat, este nevoie de un refresh al paginii web.

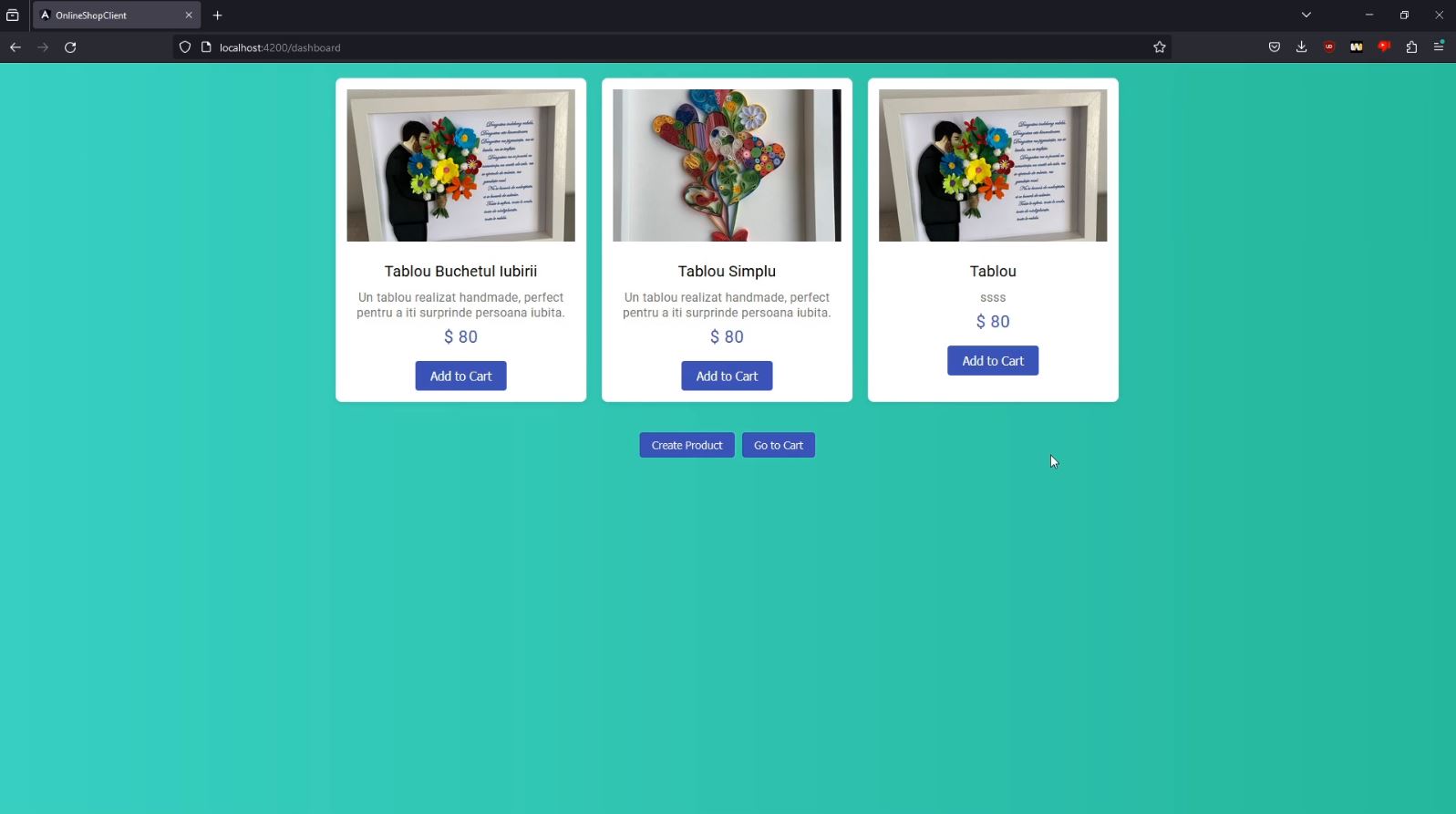


Figura 11. Adăugare produs cu succes

# Tehnologii utilizate

## Frontend: Angular

Angular este un framework de dezvoltare web open-source, dezvoltat și întreținut de către Google. A fost creat pentru a facilita dezvoltarea aplicațiilor web single-page (SPA) și a aplicațiilor web de înaltă performanță.

Angular oferă un set complet de instrumente și funcționalități pentru dezvoltarea interfeței de utilizator, gestionarea stării aplicației și comunicarea eficientă cu backend-ul.



Figura 12. Angular Logo [2]

***Caracteristici Principale:***

Component-Based Architecture: Angular folosește o arhitectură bazată pe componente, în care interfața de utilizator este împărțită în componente reutilizabile și independent gestionate.

Two-Way Data Binding: Angular oferă suport pentru two-way data binding, ceea ce înseamnă că modificările făcute în interfața de utilizator sunt reflectate automat în modelul de date și invers.

Dependency Injection: Angular integrează un sistem puternic de injectare a dependențelor, care facilitează gestionarea dependențelor și testarea unitară a componentelor.

## Visual Studio Code 2022:

Visual Studio Code (VS Code) este un editor de cod open-source, dezvoltat de către Microsoft. Este un instrument puternic și flexibil, utilizat pentru dezvoltarea și testarea aplicațiilor web, mobil și cloud. VS Code oferă suport pentru o gamă largă de limbaje de programare și extensii, permițând dezvoltatorilor să personalizeze și să îmbunătățească experiența de dezvoltare.

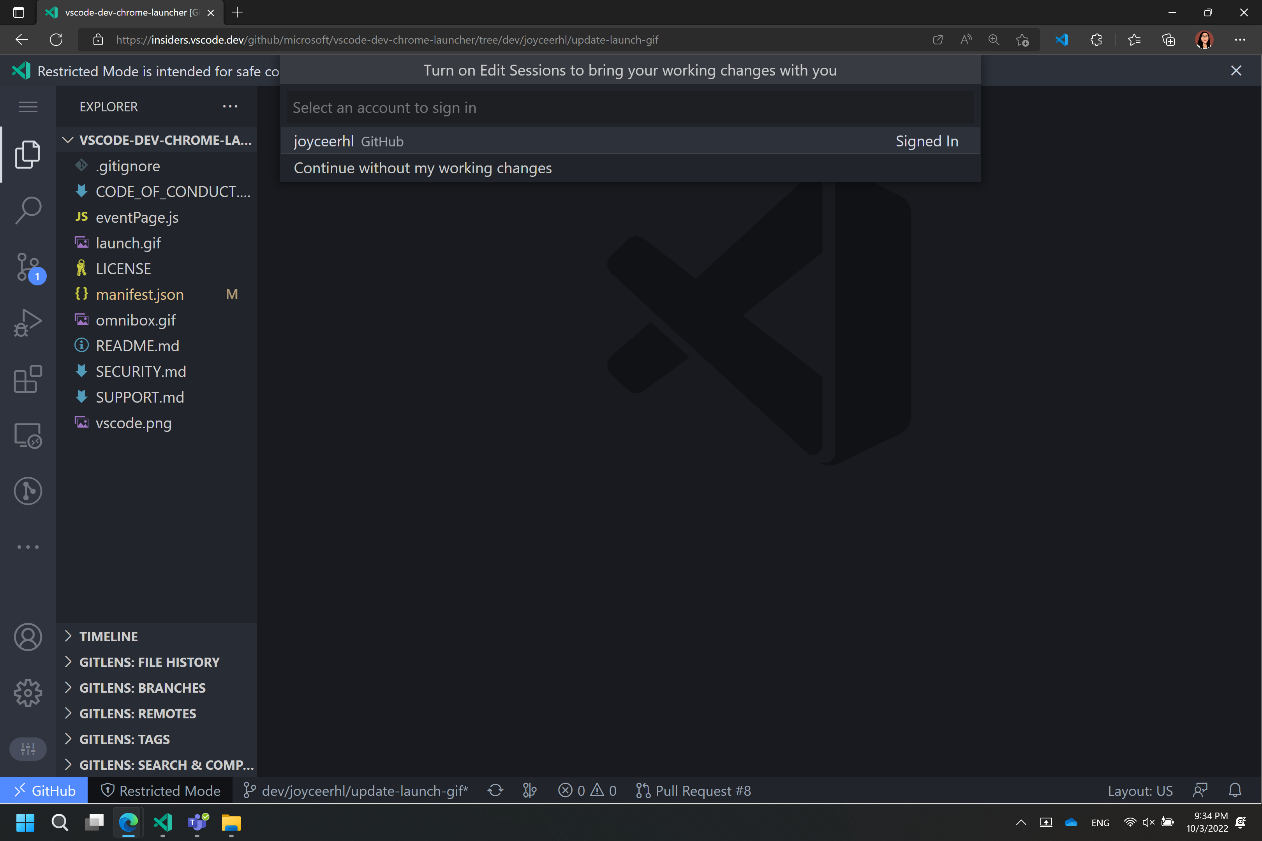


Figura 13. Interfața programului Visual Studio Code 2022 [3]

***Caracteristici Principale:***

Lightweight: VS Code este un editor de cod ușor și rapid, care se lansează rapid și rulează eficient chiar și pe computerele cu resurse limitate.

Highly Customizable: VS Code vine cu un sistem extensibil de extensii și teme, care permite dezvoltatorilor să își personalizeze și să își configureze mediul de lucru în funcție de preferințele lor.

Integrated Development Environment (IDE) Features: deși este un editor de cod minimalist, VS Code oferă o gamă largă de funcționalități integrate de IDE, inclusiv completarea automată a codului, depanare, control al versiunilor și multe altele.

## MySQL:

MySQL este un sistem de gestionare a bazelor de date relaționale open-source, dezvoltat inițial de către MySQL AB, iar ulterior de către Oracle Corporation. Este unul dintre cele mai populare și mai utilizate sisteme de bază de date din lume, cunoscut pentru performanța, fiabilitatea și scalabilitatea sa.



Figura 14. MySQL Logo [4]

***Caracteristici Principale:***

Relational Database Management System (RDBMS): MySQL este un RDBMS care utilizează modelul relațional pentru stocarea și gestionarea datelor în tabele interconectate.

Scalability: MySQL oferă suport pentru scalabilitate orizontală și verticală, permițând extinderea și gestionarea eficientă a bazelor de date în funcție de cerințele aplicației.

Security: MySQL vine cu un set complet de funcționalități de securitate, inclusiv autentificare, autorizare, criptare și auditare, pentru a asigura protecția datelor sensibile.

# GitHub

GitHub este o platformă de dezvoltare colaborativă a software-ului, care permite programatorilor să lucreze împreună la proiecte de cod într-un mod eficient și transparent.

A fost lansată în 2008 și s-a transformat rapid într-o comunitate vastă și activă pentru dezvoltatorii din întreaga lume.

Principala funcție a GitHub este de a servi ca un sistem de gestionare a versiunilor pentru proiecte de cod sursă deschisă și privată. Git folosește un sistem de control al versiunilor distribuite pentru a urmări modificările făcute de diferiți colaboratori și pentru a facilita colaborarea și integrarea schimbărilor într-un mod organizat și eficient.

Pe lângă funcționalitățile de bază de gestionare a codului sursă, GitHub oferă și o serie de alte caracteristici utile, cum ar fi problemele și cererile de tragere (pull requests), care facilitează discuția și revizuirea codului, sisteme de urmărire a problemelor (issue tracking), wiki-uri pentru documentație și multe altele.

Una dintre cele mai puternice aspecte ale GitHub este comunitatea sa activă și deschisă, unde dezvoltatorii participă pentru a descoperi proiecte noi, pentru a contribui la proiectele existente și pentru a împărtăși cunoștințe și experiențe într-un mediu colaborativ.

Un repository în GitHub este un spațiu digital dedicat pentru stocarea și gestionarea unui proiect de software sau a altui tip de proiect folosind sistemul de control al versiunilor Git.

Aici sunt păstrate toate fișierele și istoricul complet al modificărilor, iar dezvoltatorii pot lucra în mod colaborativ, să urmărească și să gestioneze schimbările în codul sursă. Este structurat într-un mod care reflectă arhitectura proiectului și poate include facilități suplimentare precum probleme, cereri de tragere și wiki-uri pentru documentație.

## Avantaje

Folosirea GitHubului vine cu numeroase avantaje pentru dezvoltatorii de software și pentru echipele de proiect.

Câteva dintre acestea sunt:

- Controlul versiunilor: GitHub oferă un sistem puternic de control al versiunilor prin intermediul Git, permițând dezvoltatorilor să urmărească și să gestioneze istoricul complet al modificărilor făcute în codul sursă. Acest lucru permite revenirea la versiuni anterioare, compararea diferitelor versiuni și gestionarea eficientă a dezvoltării de software.

- Colaborare eficientă: GitHub facilitează colaborarea între dezvoltatori și echipe de proiect prin intermediul funcționalităților precum branch-uri, cereri de tragere și comentarii pe cod. Dezvoltatorii pot lucra în mod concurent la diferite funcționalități ale proiectului și pot integra schimbările în mod organizat și eficient.

- Transparență și vizibilitate: platforma GitHub oferă un mediu transparent pentru dezvoltarea proiectului, permițând tuturor membrilor echipei să urmărească progresul, să aducă modificări și să ofere feedback. Acest lucru promovează transparența și colaborarea deschisă în cadrul echipei.

- Gestionarea task-urilor și problemelor: GitHub include un sistem de urmărire a problemelor (issues) care permite echipei să gestioneze și să urmărească sarcinile, bug-urile și cerințele de funcționalitate într-un mod organizat. Acest lucru facilitează planificarea și prioritizarea activităților în cadrul proiectului.

- Documentare și gestionare a cunoștințelor: prin intermediul wiki-urilor și al funcționalităților de comentarii, GitHub permite echipei să se documenteze în cadrul proiectului și să împărtășească cunoștințele și experiențele.

- Comunitate activă: GitHub este o comunitate vastă și activă de dezvoltatori din întreaga lume, oferind oportunități pentru colaborare, învățare și împărtășirea cunoștințelor.

Dezvoltatorii pot descoperi proiecte noi, pot contribui la proiectele existente și pot interacționa cu alți profesioniști din domeniul IT.

Link-ul către proiect pe GitHub este: https://github.com/Raluca-Olteanu/Proiect-ISW

# Testare

Testarea unui proiect este o etapă crucială în dezvoltarea oricărui produs software sau aplicație. Acest proces implică evaluarea și validarea funcționalităților, performanței și securității produsului înainte de lansarea acestuia către utilizatori.

Pentru testare s-au introdus date neînregistrate pentru a observa dacă autentificarea va mai avea loc.

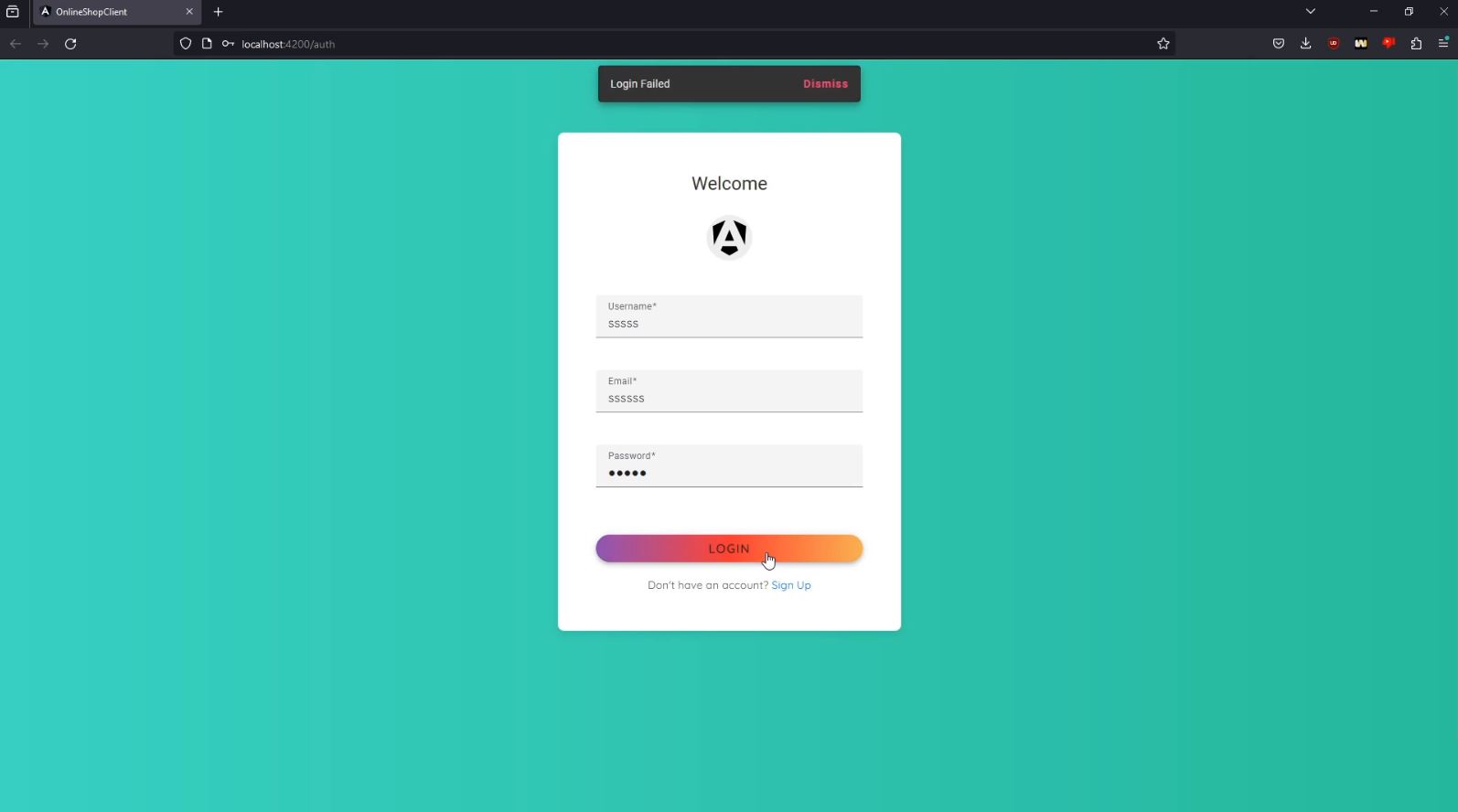


Figura 15. Testare autentificare

Astfel, datele nefiind stocate în baza de date, s-a afișat mesajul “Login Failed”, iar autentificarea nu a avut succes.

# Management

Managementul este important în realizarea unui proiect de succes, indiferent de domeniul în care acesta operează. Iată câteva aspecte cheie care subliniază importanța managementului în acest proces:

- Planificare eficientă: managementul implică elaborarea unui plan detaliat pentru proiect, stabilirea obiectivelor, identificarea resurselor necesare și crearea unui calendar de implementare. O planificare solidă ajută la evitarea întârzierilor, la gestionarea eficientă a resurselor și la stabilirea unor așteptări realiste pentru echipele implicate.

- Alocarea resurselor: Asigurarea că resursele (financiare, umane, materiale) sunt alocate în mod eficient și sunt utilizate în conformitate cu obiectivele proiectului este un pas important în realizarea proiectului. Un management eficient poate identifica resursele necesare și poate gestiona bugetul și calendarul proiectului pentru a maximiza eficiența.

- Coerență și coeziune: managementul facilitează comunicarea și coordonarea între diferitele echipe și departamente implicate într-un proiect. Prin stabilirea unor linii clare de responsabilitate și prin promovarea colaborării, managementul contribuie la menținerea coerenței și coeziunii în cadrul echipei.

- Identificarea și gestionarea riscurilor: managementul este responsabil pentru identificarea potențialelor riscuri și amenințări care ar putea afecta proiectul și pentru dezvoltarea unor strategii de gestionare a acestora. Prin anticiparea și abordarea proactivă a riscurilor, managementul poate minimiza impactul acestora asupra proiectului.

- Monitorizare și control: un aspect esențial al managementului este monitorizarea progresului proiectului și identificarea oricăror devieri față de planul inițial. Prin implementarea unor mecanisme de control și raportare, managementul poate interveni în mod prompt pentru a corecta problemele și a asigura respectarea termenelor și bugetului.

- Evaluare și îmbunătățire continuă: după finalizarea proiectului, managementul este important în evaluarea rezultatelor și în identificarea lecțiilor învățate. Aceste informații pot fi utilizate pentru îmbunătățirea proceselor și practicilor în viitoarele proiecte, contribuind astfel la creșterea performanței organizaționale.

Astfel, managementul este esențial în realizarea unui proiect de succes, asigurând planificarea și coordonarea eficientă a resurselor, gestionarea riscurilor și monitorizarea progresului. Un management eficient poate contribui la îndeplinirea obiectivelor proiectului în termeni de timp, costuri și calitate.

# Lecții învățate

1. Învățarea continuă și explorarea resurselor: în domeniul tehnologiei, există întotdeauna noi concepte, tehnologii și instrumente de explorat și de învățat.
2. Debugging și Troubleshooting: am învățat să identificăm și să remediem erorile și problemele în timpul dezvoltării folosind instrumentele de debugging și de troubleshooting disponibile în Angular, Visual Studio Code și MySQL.
3. Planificare și gestionare a aimpului: trebuie pus accentual și pe importanța unei planificări detaliate și a gestionării eficiente a timpului în timpul dezvoltării proiectului.
4. Comunicare și colaborare eficientă: o comunicare clară și o colaborare eficientă între membrii echipei sunt esențiale pentru succesul proiectului. Am constatat că menținerea unei comunicări deschise și transparente a contribuit la evitarea confuziei și a divergențelor în privința obiectivelor și a cerințelor proiectului.
5. Flexibilitate și adaptabilitate: flexibilitatea și capacitatea de a te adapta la schimbările cerințelor sau la problemele neprevăzute sunt cheia succesului în dezvoltarea unui proiect.
6. Înțelegerea nevoilor utilizatorilor: prioritizarea nevoilor și experienței utilizatorilor este un pas important în dezvoltarea unei aplicații de success.

# Concluzii

Considerăm ca proiectul intitulat „Proiectarea și implementarea unei platforme Web pentru produse Handmade”, are o aplicabilitate în viața reală, fiind un avantaj în viața pasionaților de produse originale.

Deoarece este mult mai ușor de vizionat, achiziționat, cât și de promovat, este un beneficiu atât pentru producători, cât și pentru iubitori.

# Codul sursă

## Autentificare:

.auth-wrapper {

  background: white;

  border-radius: 8px;

  box-shadow: 0px 4px 12px rgba(0, 0, 0, 0.1);

  padding: 50px;

  width: 350px;

  margin: 5% auto;

  text-align: center;

}

.login-header {

  margin-bottom: 30px;

}

.login-header h1 {

  font-size: 24px;

  color: #333;

}

.login-icon {

  background: #eee;

  font-size: 40px;

  width: 60px;

  height: 60px;

  line-height: 60px;

  border-radius: 50%;

  margin: 10px auto;

}

.example-full-width {

  width: 100%;

  margin-bottom: 20px;

}

.mat-form-field {

  font-size: 14px;

}

.submit-button {

  width: 100%;

  background: linear-gradient(90deg, rgba(131,58,180,1) 0%, rgba(253,29,29,1) 50%, rgba(252,176,69,1) 100%);

  color: white;

  padding: 15px 20px;

  margin-top: 20px;

  border: none;

  border-radius: 25px;

  box-shadow: none;

  font-size: 16px;

  text-transform: uppercase;

}

.submit-button:disabled {

  background: #ccc;

}

.submit-button:hover {

  opacity: 0.9;

}

.switch-mode {

  margin-top: 20px;

  font-size: 14px;

  color: #555;

}

.switch-mode span {

  color: #007bff;

  cursor: pointer;

}

body {

  background: linear-gradient(to right, #46e1c2, #34c99f); /\* Turquoise to green gradient \*/

  height: 100%;

  width: 100%;

}

:host {

  display: block;

  height: 100vh;

  width: 100vw;

  background: linear-gradient(to right, #46e1c2, #34c99f); /\* Turquoise to green gradient \*/

}

.auth-wrapper,

.login-header,

.form-wrapper,

button,

input,

mat-form-field {

font-family: 'Quicksand', sans-serif;

}

<div class="auth-wrapper">

    <div class="login-header">

        <!-- Afișează un mesaj diferit în funcție de modul curent -->

        <h1>{{ isLoginMode ? 'Welcome' : 'Create Account' }}</h1>

        <!-- Logo-ul Angular folosit pentru formularele de login și înregistrare -->

        <img src="https://upload.wikimedia.org/wikipedia/commons/2/2c/Angular\_icon.svg" class="login-icon" alt="Angular Logo" />

    </div>

    <form #authForm="ngForm" (ngSubmit)="onSubmit(authForm)" class="form-wrapper">

        <!-- Câmpurile pentru înregistrare -->

        <!-- Câmpul pentru Username -->

        <mat-form-field appearance="fill" class="example-full-width">

            <mat-label>Username</mat-label>

            <input matInput placeholder="Enter your username" type="userName" name="userName" ngModel [required]="true">

        </mat-form-field>

        <!-- Câmpul pentru First Name, vizibil doar în modul de înregistrare -->

        <mat-form-field appearance="fill" class="example-full-width" \*ngIf="!isLoginMode">

            <mat-label>First Name</mat-label>

            <input matInput placeholder="Enter your first name" type="name" name="firstName" ngModel [required]="true">

        </mat-form-field>

        <!-- Câmpul pentru Last Name, vizibil doar în modul de înregistrare -->

        <mat-form-field appearance="fill" class="example-full-width" \*ngIf="!isLoginMode">

            <mat-label>Last Name</mat-label>

            <input matInput placeholder="Enter your last name" type="name" name="lastName" ngModel [required]="true">

        </mat-form-field>

        <!-- Câmpuri comune pentru login și înregistrare -->

        <mat-form-field appearance="fill" class="example-full-width">

            <mat-label>Email</mat-label>

            <input matInput placeholder="Enter your email" type="email" name="email" ngModel [required]="true">

        </mat-form-field>

        <mat-form-field appearance="fill" class="example-full-width">

            <mat-label>Password</mat-label>

            <input matInput placeholder="Enter your password" type="password" name="password" ngModel [required]="true">

        </mat-form-field>

        <!-- Câmpul pentru Role, vizibil doar în modul de înregistrare -->

        <mat-form-field appearance="fill" class="example-full-width" \*ngIf="!isLoginMode">

            <mat-label>Role</mat-label>

            <mat-select placeholder="Select your role" name="role" ngModel [required]="true">

                <mat-option value="User">User</mat-option>

                <mat-option value="Admin">Admin</mat-option>

            </mat-select>

        </mat-form-field>

        <!-- Butonul de submit -->

        <button mat-raised-button class="submit-button" type="submit" [disabled]="!authForm.valid">

            {{ isLoginMode ? 'Login' : 'Sign Up' }}

        </button>

        <!-- Link pentru schimbarea modului -->

        <div class="switch-mode">

            {{

                isLoginMode ? 'Don\'t have an account?' : 'Already have an account?'

            }}

            <span (click)="onSwitchMode()">{{ isLoginMode ? 'Sign Up' : 'Login' }}</span>

        </div>

    </form>

</div>

import { ComponentFixture, TestBed } from '@angular/core/testing';

import { AuthComponent } from './auth.component';

describe('AuthComponent', () => {

  let component: AuthComponent;

  let fixture: ComponentFixture<AuthComponent>;

  beforeEach(() => {

    TestBed.configureTestingModule({

      declarations: [AuthComponent]

    });

    fixture = TestBed.createComponent(AuthComponent);

    component = fixture.componentInstance;

    fixture.detectChanges();

  });

  it('should create', () => {

    expect(component).toBeTruthy();

  });

});

import { Component } from '@angular/core';

import { NgForm } from '@angular/forms';

import { Router } from '@angular/router';

import { MatSnackBar } from '@angular/material/snack-bar';

import { AuthService } from 'src/app/services/auth.service';

import { UserService } from 'src/app/services/user.service';

import { User } from 'src/app/interfaces/user.interface';

@Component({

  selector: 'app-auth',

  templateUrl: './auth.component.html',

  styleUrls: ['./auth.component.css']

})

export class AuthComponent {

  // Constructorul clasei AuthComponent injectează serviciile necesare

  constructor(

    private authService: AuthService,

    private userService: UserService,

    private router: Router,

    private \_snackBar: MatSnackBar) { }

  // Variabilă care indică dacă este mod de login sau de înregistrare

  isLoginMode: boolean = true;

  // Funcția care se execută la submit-ul formularului

  onSubmit(form: NgForm) {

    // Verifică dacă formularul este valid

    if (form.valid) {

      // Dacă este mod de login

      if (this.isLoginMode) {

        // Apelează metoda de login din AuthService și se abonează la rezultatul acesteia

        this.authService.login(form.value).subscribe({

          next: async (res: any) => {

            // Resetează formularul

            form.reset();

            // Stochează token-ul primit

            this.authService.storeToken(res.token);

            // Stochează informațiile utilizatorului primit

            this.authService.storeUser(res);

            console.log(res);

            // Afișează un mesaj de succes folosind MatSnackBar

            this.\_snackBar.open('Login Successful', 'Dismiss', {

              duration: 3000,

              horizontalPosition: 'center',

              verticalPosition: 'top',

            });

            // Redirecționează utilizatorul către dashboard

            this.router.navigate(['dashboard']);

          },

          // În caz de eroare, afișează un mesaj de eroare folosind MatSnackBar

          error: (err: any) => {

            this.\_snackBar.open('Login Failed', 'Dismiss', {

              duration: 3000,

              horizontalPosition: 'center',

              verticalPosition: 'top',

            });

          }

        });

      } else {

        // Dacă este mod de înregistrare

        this.authService.register(form.value).subscribe({

          next: async (res: any) => {

            // Resetează formularul

            form.reset();

            // Stochează token-ul primit

            this.authService.storeToken(res.token);

            // Stochează informațiile utilizatorului primit

            this.authService.storeUser(res);

            // Afișează un mesaj de succes folosind MatSnackBar

            this.\_snackBar.open('Registration Successful', 'Dismiss', {

              duration: 3000,

              horizontalPosition: 'center',

              verticalPosition: 'top',

            });

            // Schimbă modul în mod de login

            this.isLoginMode = true;

          },

          // În caz de eroare, afișează un mesaj de eroare folosind MatSnackBar

          error: (err: any) => {

            this.\_snackBar.open('Registration Failed', 'Dismiss', {

              duration: 3000,

              horizontalPosition: 'center',

              verticalPosition: 'top',

            });

          }

        });

      }

    } else {

      // Dacă formularul nu este valid, afișează un mesaj folosind MatSnackBar

      this.\_snackBar.open('Please fill in all required fields', 'Dismiss', {

        duration: 3000,

        horizontalPosition: 'center',

        verticalPosition: 'top',

      });

    }

  }

  // Funcția care schimbă modul între login și înregistrare

  onSwitchMode() {

    this.isLoginMode = !this.isLoginMode;

  }

}

## Coș de cumpărături:

.cart {

    padding: 20px;

    max-width: 800px;

    margin: 0 auto;

    background-color: #fff;

    border-radius: 8px;

    box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);

  }

  .cart h2 {

    font-size: 24px;

    margin-bottom: 20px;

    text-align: center;

  }

  .cart-item {

    display: flex;

    align-items: center;

    margin-bottom: 15px;

    border-bottom: 1px solid #e0e0e0;

    padding-bottom: 10px;

  }

  .cart-item-image {

    width: 100px;

    height: 100px;

    object-fit: cover;

    margin-right: 15px;

  }

  .cart-item-info {

    flex-grow: 1;

  }

  .cart-item-info p {

    margin: 0;

    font-size: 18px;

  }

  .cart-item-info p + p {

    margin-top: 5px;

    color: #757575;

  }

  .cart h3 {

    font-size: 20px;

    margin-top: 20px;

    text-align: center;

  }

  .cart button {

    margin-top: 20px;

    width: 100%;

    padding: 10px;

    border: none;

    background-color: #3f51b5;

    color: white;

    border-radius: 4px;

    font-size: 16px;

    transition: background-color 0.3s;

  }

  .cart button:hover {

    background-color: #303f9f;

  }

<div class="cart">

    <h2>Your Cart</h2>

    <!-- Verifică dacă există produse în coș -->

    <div \*ngIf="(cartItems$ | async) as cartItems">

      <!-- Afișează produsele din coș dacă există, altfel afișează un mesaj că coșul este gol -->

      <div \*ngIf="cartItems.size > 0; else emptyCart">

        <!-- Iterează prin produsele din coș și le afișează -->

        <div \*ngFor="let entry of cartItems.entries()" class="cart-item">

          <!-- Imaginea produsului -->

          <img [src]="entry[0].image" alt="{{entry[0].name}}" class="cart-item-image">

          <div class="cart-item-info">

            <!-- Numele și prețul produsului -->

            <p>{{entry[0].name}} - ${{entry[0].price}}</p>

            <!-- Cantitatea produsului -->

            <p>Quantity: {{entry[1]}}</p>

          </div>

        </div>

        <!-- Afișează suma totală a produselor din coș -->

        <h3>Total: ${{ totalAmount$ | async }}</h3>

        <!-- Butoane pentru golirea coșului și finalizarea comenzii -->

        <button (click)="clearCart()">Clear Cart</button>

        <button (click)="checkout()">Checkout</button>

      </div>

      <!-- Mesaj afișat când coșul este gol -->

      <ng-template #emptyCart>

        <p>Your cart is empty.</p>

      </ng-template>

    </div>

</div>

import { ComponentFixture, TestBed } from '@angular/core/testing';

import { CartComponent } from './cart.component';

describe('CartComponent', () => {

  let component: CartComponent;

  let fixture: ComponentFixture<CartComponent>;

  beforeEach(() => {

    TestBed.configureTestingModule({

      declarations: [CartComponent]

    });

    fixture = TestBed.createComponent(CartComponent);

    component = fixture.componentInstance;

    fixture.detectChanges();

  });

  it('should create', () => {

    expect(component).toBeTruthy();

  });

});

import { Component, OnInit } from '@angular/core';

import { CartService } from 'src/app/services/cart.service';

import { Product } from 'src/app/interfaces/product.interface';

import { Observable } from 'rxjs';

@Component({

  selector: 'app-cart',

  templateUrl: './cart.component.html',

  styleUrls: ['./cart.component.css']

})

export class CartComponent implements OnInit {

  // Variabile care stochează produsele din coș și suma totală, ca fluxuri observabile

  cartItems$!: Observable<Map<Product, number>>;

  totalAmount$!: Observable<number>;

  // Constructorul clasei CartComponent injectează serviciul CartService

  constructor(private cartService: CartService) {}

  // Metoda ngOnInit se execută la inițializarea componentei

  ngOnInit() {

    // Obține produsele din coș și suma totală folosind serviciul CartService

    this.cartItems$ = this.cartService.getItems();

    this.totalAmount$ = this.cartService.getTotalAmount();

  }

  // Metoda care golește coșul de cumpărături

  clearCart() {

    this.cartService.clearCart();

  }

  // Metoda pentru finalizarea comenzii

  checkout() {

    // Apelează metoda checkout din CartService și se abonează la rezultat

    this.cartService.checkout().subscribe(() => {

      // Afișează un mesaj de succes în cazul în care comanda este finalizată cu succes

      alert('Checkout successful!');

    }, (error) => {

      // Afișează un mesaj de eroare în cazul în care comanda nu reușește

      alert('Checkout failed. Please try again.');

    });

  }

}

## Produse:

mat-form-field {

    width: 100%;

    margin-bottom: 15px;

  }

  mat-dialog-content {

    padding: 20px;

  }

  mat-dialog-actions {

    padding: 10px 20px;

  }

  button[mat-dialog-close] {

    margin-right: 10px;

  }

<h2 mat-dialog-title>Create Product</h2>

<mat-dialog-content>

    <form [formGroup]="createProductForm">

        <!-- Câmp pentru numele produsului -->

        <mat-form-field appearance="fill">

            <mat-label>Name</mat-label>

            <input matInput formControlName="productName" type="text" required>

            <mat-error \*ngIf="createProductForm.controls['productName'].hasError('required')">Product name is required.</mat-error>

        </mat-form-field>

        <!-- Câmp pentru descrierea produsului -->

        <mat-form-field appearance="fill">

            <mat-label>Description</mat-label>

            <input matInput formControlName="productDescription" type="text" required>

            <mat-error \*ngIf="createProductForm.controls['productDescription'].hasError('required')">Product description is required.</mat-error>

        </mat-form-field>

        <!-- Câmp pentru calea imaginii produsului -->

        <mat-form-field appearance="fill">

            <mat-label>Image Path</mat-label>

            <input matInput formControlName="imagePath" type="text" required>

            <mat-error \*ngIf="createProductForm.controls['imagePath'].hasError('required')">Image Path is required.</mat-error>

        </mat-form-field>

        <!-- Câmp pentru prețul produsului -->

        <mat-form-field appearance="fill">

            <mat-label>Price</mat-label>

            <input matInput formControlName="productPrice" type="number" required>

            <mat-error \*ngIf="createProductForm.controls['productPrice'].hasError('required')">Product price is required.</mat-error>

        </mat-form-field>

        <!-- Câmp pentru cantitatea produsului -->

        <mat-form-field appearance="fill">

            <mat-label>Quantity</mat-label>

            <input matInput formControlName="productQuantity" type="number" required>

            <mat-error \*ngIf="createProductForm.controls['productQuantity'].hasError('required')">Product quantity is required.</mat-error>

        </mat-form-field>

    </form>

</mat-dialog-content>

<mat-dialog-actions align="end">

    <!-- Buton pentru anularea acțiunii -->

    <button mat-button mat-dialog-close>Cancel</button>

    <!-- Buton pentru crearea produsului, dezactivat dacă formularul nu este valid -->

    <button mat-raised-button color="primary" (click)="onSubmit()" [disabled]="createProductForm.invalid">Create</button>

</mat-dialog-actions>

import { ComponentFixture, TestBed } from '@angular/core/testing';

import { CreateProductComponent } from './create-product.component';

describe('CreateProductComponent', () => {

  let component: CreateProductComponent;

  let fixture: ComponentFixture<CreateProductComponent>;

  beforeEach(() => {

    TestBed.configureTestingModule({

      declarations: [CreateProductComponent]

    });

    fixture = TestBed.createComponent(CreateProductComponent);

    component = fixture.componentInstance;

    fixture.detectChanges();

  });

  it('should create', () => {

    expect(component).toBeTruthy();

  });

});

import { Component, OnInit } from '@angular/core';

import { FormBuilder, FormGroup, Validators } from '@angular/forms'; // Importă Validators pentru validarea formularelor

import { MatDialogRef } from '@angular/material/dialog';

import { MatSnackBar } from '@angular/material/snack-bar';

import { Product } from 'src/app/interfaces/product.interface';

import { UserService } from 'src/app/services/user.service';

@Component({

  selector: 'app-create-product',

  templateUrl: './create-product.component.html',

  styleUrls: ['./create-product.component.css']

})

export class CreateProductComponent implements OnInit {

  // Definire variabilă pentru formularul de creare produs

  createProductForm!: FormGroup;

  // Constructorul clasei CreateProductComponent injectează serviciile necesare

  constructor(private dialogRef: MatDialogRef<CreateProductComponent>,

    private formBuilder: FormBuilder,

    private userService: UserService,

    private snackBar: MatSnackBar) { }

  // Metoda ngOnInit se execută la inițializarea componentei

  ngOnInit() {

    // Inițializează formularul cu câmpuri și validatori

    this.createProductForm = this.formBuilder.group({

      productName: ['', Validators.required], // Câmp pentru numele produsului, obligatoriu

      productDescription: ['', Validators.required], // Câmp pentru descrierea produsului, obligatoriu

      imagePath: ['', Validators.required], // Câmp pentru calea imaginii produsului, obligatoriu

      productPrice: ['', Validators.required], // Câmp pentru prețul produsului, obligatoriu

      productQuantity: ['', Validators.required] // Câmp pentru cantitatea produsului, obligatoriu

    });

  }

  // Metoda care se execută la submit-ul formularului

  onSubmit() {

    // Creează un obiect de tip Product folosind valorile din formular

    const product: Product = {

      name: this.createProductForm.value.productName,

      description: this.createProductForm.value.productDescription,

      image: this.createProductForm.value.imagePath,

      price: this.createProductForm.value.productPrice,

      quantity: this.createProductForm.value.productQuantity

    };

    // Apelează metoda createProduct din UserService și se abonează la rezultat

    this.userService.createProduct(product).subscribe(() => {

      // Afișează un mesaj de succes folosind MatSnackBar

      this.snackBar.open('Product created', 'Close', {

        duration: 2000,

        horizontalPosition: 'center',

        verticalPosition: 'top',

      });

      // Închide dialogul

      this.dialogRef.close();

    });

  }

}

## Dashboard:

/\* Stiluri pentru componenta dashboard \*/

.dashboard {

  display: flex;

  flex-wrap: wrap;

  gap: 20px;

  justify-content: center;

  padding: 20px;

}

.product {

  background-color: #fff;

  border-radius: 8px;

  box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);

  overflow: hidden;

  width: 300px;

  text-align: center;

  padding: 15px;

  display: flex;

  flex-direction: column;

  justify-content: space-between;

}

.product img {

  max-width: 100%;

  height: 200px; /\* Setează o înălțime fixă pentru imagini \*/

  object-fit: cover; /\* Menține aspectul imaginii și acoperă spațiul \*/

  margin-bottom: 15px;

}

.product-info {

  font-size: 18px;

  flex-grow: 1;

}

.product-info h3 {

  font-size: 20px;

  margin: 10px 0;

}

.product-info p {

  font-size: 16px;

  color: #757575;

  margin: 0;

  margin-bottom: 10px;

}

.product-info h4 {

  font-size: 22px;

  margin: 10px 0;

  color: #3f51b5;

}

.product-info button {

  background-color: #3f51b5;

  color: white;

  border: none;

  padding: 10px 20px;

  border-radius: 4px;

  font-size: 16px;

  transition: background-color 0.3s;

  margin-top: 10px;

}

.product-info button:hover {

  background-color: #303f9f;

}

.dashboard-buttons {

  display: flex;

  justify-content: center;

  gap: 10px;

  margin-top: 20px;

}

.dashboard-buttons button {

  padding: 8px 16px;

  background-color: #3f51b5;

  color: white;

  border: none;

  border-radius: 4px;

  font-size: 14px;

  cursor: pointer;

  transition: background-color 0.3s;

  flex: none; /\* Schimbă de la flex: 1 la flex: none pentru a evita întinderea \*/

}

.dashboard-buttons button:hover {

  background-color: #303f9f;

}

<div class="dashboard">

  <!-- Iterează prin produsele și le afișează -->

  <div class="product" \*ngFor="let product of products">

    <img [src]="product.image.toString()" alt="{{product.name}}">

    <div class="product-info">

      <h3>{{product.name}}</h3>

      <p>{{product.description}}</p>

      <h4>$ {{product.price}}</h4>

      <button (click)="addToCart(product)">Add to Cart</button>

    </div>

  </div>

</div>

<div class="dashboard-buttons">

  <!-- Buton pentru deschiderea dialogului de creare produs -->

  <button (click)="openCreateProductDialog()">Create Product</button>

  <!-- Buton pentru navigarea către coșul de cumpărături -->

  <button routerLink="/cart">Go to Cart</button>

</div>

import { ComponentFixture, TestBed } from '@angular/core/testing';

import { DashboardComponent } from './dashboard.component';

describe('DashboardComponent', () => {

  let component: DashboardComponent;

  let fixture: ComponentFixture<DashboardComponent>;

  beforeEach(() => {

    TestBed.configureTestingModule({

      declarations: [DashboardComponent]

    });

    fixture = TestBed.createComponent(DashboardComponent);

    component = fixture.componentInstance;

    fixture.detectChanges();

  });

  it('should create', () => {

    expect(component).toBeTruthy();

  });

});

import { Component, OnInit } from '@angular/core';

import { Product } from 'src/app/interfaces/product.interface';

import { UserService } from 'src/app/services/user.service';

import { CreateProductComponent } from '../create-product/create-product.component';

import { MatDialog } from '@angular/material/dialog';

import { CartService } from 'src/app/services/cart.service';

@Component({

  selector: 'app-dashboard',

  templateUrl: './dashboard.component.html',

  styleUrls: ['./dashboard.component.css']

})

export class DashboardComponent implements OnInit {

  // Variabilă pentru stocarea produselor

  products: any;

  // Constructorul clasei DashboardComponent injectează serviciile necesare

  constructor(private userService: UserService, private dialog: MatDialog, private cartService: CartService) { }

  // Metoda ngOnInit se execută la inițializarea componentei

  ngOnInit() {

    // Obține toate produsele folosind UserService și le stochează în variabila products

    this.userService.getAllProducts().subscribe(

      (products) => {

        this.products = products;

        console.log(this.products);

      },

      (error) => {

        console.error('Error fetching products', error);

      }

    );

  }

  // Metoda pentru adăugarea unui produs în coșul de cumpărături

  addToCart(product: Product): void {

    this.cartService.addToCart(product);

    console.log('Added to cart:', product);

  }

  // Metoda pentru deschiderea dialogului de creare produs

  openCreateProductDialog() {

    const dialogRef = this.dialog.open(CreateProductComponent, {

      width: '300px'

    });

    // Se execută după ce dialogul este închis

    dialogRef.afterClosed().subscribe(() => {

      // Tratează orice acțiuni după închiderea dialogului

    });

  }

}

# Bibliografie

[1] <https://ideideafaceri.manager.ro/articole/dosare-de-afaceri-6/pretul-unui-produs-handmade-stabilirea-valorii-corecte-21625.html>

[2] https://logowik.com/angular-new-logo-vector-62174.html

[3] https://code.visualstudio.com/updates/v1\_72

[4] <https://wikitech.wikimedia.org/wiki/File:Mysql_logo.png>

Kerzner, H. (2017). Project Management: A Systems Approach to Planning, Scheduling, and Controlling. John Wiley & Sons.

<https://github.com/about>

Chandler, Alfred D. (1977). "The Visible Hand: The Managerial Revolution in American Business". Harvard University Press.