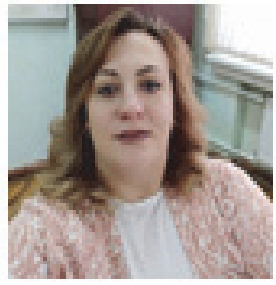

Implementarea unui coș de cumpărături în JavaScript

Constanța LOJECICO, constanta.lojecico1@gmail.com
Adriana CATRUC, catruc@ase.md
Academia de Studii Economice din Moldova, or. Chișinău, R. Moldova.

Abstract: Pentru crearea unui magazin online complex este necesară folosirea mai multor tehnologii și adăugarea mai multor funcționalități decât în cazul în care dezvoltăm un website informațional. Un element cheie desigur că este coșul de cumpărături. Prezenta lucrare cuprinde destul de detaliat aspecte teoretice și practice ale modalităților de concepere și implementare a coșului de cumpărături folosind JavaScript.

To create a complex online store, it is necessary to use several technologies and add more features than if we were developing an informational website. A key element of course is the shopping cart. This paper covers in great detail the theoretical and practical aspects of how to design and implement the shopping cart using JavaScript.

Keywords: Computer Science, JavaScript, Shopping Cart.

Introducere

Pentru crearea unui magazin online complex este necesară folosirea mai multor tehnologii și adăugarea mai multor funcționalități decât în cazul în care dezvoltăm un website informațional. Un element cheie desigur că este coșul de cumpărături. În acest proiect am analizat cum cu ajutorul limbajului de programare JavaScript putem adăuga produsele în coș, cum adăugăm funcțiile pentru modificarea numărului de produse în coș, ștergerea produselor din coș și pentru ajustarea prețului total. Pentru ca toate acestea să funcționeze, este necesară stocarea informației pe server și manipularea acestora cu tehnologii precum Node.js, Express.js, etc. Atunci când lucrăm cu proprietăți precum prețul produsului, este important ca acestea să nu fie pe partea de front-end, căci devine ușor pentru alți utilizatori să modifice valoarea acestora. În capitolul 1, am descris tehnologiile utilizate pentru crearea acestui proiect, caracteristicile principale ale acestora, avantajele și cum pot fi folosite. În capitolul 2, subcapitolul 1 am arătat cum dezvoltăm partea de front-end, iar în subcapitolul 2, am demonstrat cum putem trece lista produselor de pe partea client, pe partea de server.

Tehnologii utilizate

Reprezintă o scurtă introducere a principalelor noțiuni legate de Internet. Sunt explicate tehnologiile utilizate pentru dezvoltarea front-end și cele de back-end.

1.Descrierea HTML și CSS – care au fost utilizate pentru dezvoltarea front-end.

2.Protocolul HTTP – unde este explicat cum lucrează acest protocol și cum se realizează comunicarea dintre client și server.
3.Limbajul de programare Java Script – o introducere în limbajul JS și cum putem folosi localStorage pentru stocarea datelor.
4.Node.js – acest subcapitol descrie caracteristicile cele mai importante ale mediului Node.js și de ce el este o alegere perfectă pentru programatori.
5.Express.js – subcapitol ce descrie Express.js și cum funcționează acesta.
6.Gateway de plată pentru website – include explicarea noțiunii de gateway de plată (eng. Payment gateway).

Tehnologii utilizate în proiectarea paginilor web pe baza de client

HTML (Hyper Text Markup Language) este un limbaj folosit pentru crearea paginilor web ce utilizează un set de tag-uri de marcare pentru a descrie pagini web, nefiind considerat un limbaj de programare. Tag-urile de marcare HTML sunt de obicei numite tag-uri HTML. [1]
CSS (Cascading Style Sheets) este codul care stilizează conținutul web. La fel ca HTML, CSS nu este un limbaj de programare. Nici nu este un limbaj de markup. CSS este un limbaj de foaie de stil. CSS este ceea ce utilizați pentru a stiliza selectiv elemente HTML. [1]

Protocolul HTTP

HTTP (Hypertext Transfer Protocol) este protocol folosit pentru accesarea datelor pe World Wide Web (www), și anume este responsabil pentru comunicarea dintre web servere și clienți. Clienții și serverele comunică prin schimbul de mesaje. Mesajele trimise de client, care de obicei este un browser Web, se numesc cereri (eng. Requests), iar mesajele trimise de server ca răspuns, se numesc răspunsuri (eng. Responses).not number the pages. [2]

Limbajul de programare JavaScript

JavaScript este un limbaj de scriptare foarte puternic. JavaScript este utilizat în principal pentru îmbunătățirea interacțiunii unui utilizator cu pagina web. Cu alte cuvinte, o pagină web devine mai interactivă, cu ajutorul JavaScript. JavaScript este, de asemenea, utilizat pe scară largă în dezvoltarea jocurilor și dezvoltarea aplicațiilor mobile.
JavaScript a fost dezvoltat de Brendan Eich în 1995, care a apărut în Netscape, un popular browser de atunci. Limbajul a fost numit inițial Live Script și ulterior a fost redenumit JavaScript. Există mulți programatori care cred că JavaScript și Java sunt la fel. De fapt, JavaScript și Java nu au nicio legătură. Java este un limbaj de programare foarte complex, în timp ce JavaScript este doar un limbaj de scriptare. Sintaxa JavaScript este în mare parte influențată de limbajul de programare C. [3] JavaScript localStorage – HTML DOM Windows localStorage este furnizat de Browser și ne permite să stocăm date ca perechi cheie-valoare în browserul nostru web folosind un obiect. Datele sunt stocate ca pereche cheie-valoare, iar cheile sunt unice. Cheile și valorile sunt întotdeauna în formatul UTF-16 DOM String care este stocat în localStorage. LocalStorage deține datele fără dată de expirare, care sunt disponibile pentru utilizator chiar și după închiderea ferestrei browserului. Este util în diverse situații, cum ar fi păstrarea datelor coșului de cumpărături sau păstrarea datelor utilizatorului logat.

Mediul de execuție NODE.JS

Node.js este o bibliotecă și un mediu de rulare JavaScript open-source, multiplatformă, pentru rularea aplicațiilor web în afara browserului clientului. A fost dezvoltat în 2009 de către Ryan Dahl. Programatorii folosesc Node.js pentru a crea aplicații web pe partea de server și este perfect pentru aplicațiile care folosesc intens date, deoarece utilizează un model asincron, bazat pe evenimente. [4]

- Node.js folosește programare asincronă: toate API-urile bibliotecii Node.js sunt asincrone (adică,

neblocante), așa că un server bazat pe Node.js nu așteaptă ca API-ul să returneze date. Serverul apelează API-ul și, în cazul în care nu sunt returnate date, serverul trece la următorul API, modulul Events din Node.js ajută serverul să obțină un răspuns de la apelul API anterior. Acest lucru ajută și la viteza Node.js.

- Node.js este foarte rapid: fiind construit pe motorul JavaScript V8 al Google Chrome, biblioteca sa este extrem de rapidă pentru execuția codului.
- Cu un singur thread: Node.js folosește un model cu un singur thread cu buclă de evenimente. Ca rezultat, poate oferi servicii unui număr mult mai mare de solicitări decât serverele tradiționale precum Apache HTTP Server.
- Foarte scalabil: serverul Node.js răspunde într-un mod neblocant, făcându-l foarte scalabil, în contrast cu serverele tradiționale, care creează fire limitate pentru a gestiona cererile.
- Node Package Manager (NPM): Node Package Manager are peste 50.000 de pachete, astfel încât orice funcționalitate este necesară pentru o aplicație poate fi importată cu ușurință din NPM.
- No buffering: Aplicațiile Node.js nu tamponează niciodată date. Aceste aplicații pur și simplu scot datele în bucăți.
- Licență: Node.js este lansat sub licența MIT.

EXPRESS.JS

Express.js este un Java script Framework gratis si open-source, bazat pe Node.js, care suporta dezvoltarea pe partea de server si pe partea de client. Express ne ajută să setăm rutarea și să gestionăm ciclul cerere/răspuns. Express este flexibil, deoarece exista numeroase module disponibile pe managerul de pachete (npm), care pot fi conectate direct la Express.

Cum funcționează Express?

Aplicațiile Express funcționează prin trimiterea unei secvențe de apeluri la nivelul mediu. Middleware-ul are acces în două cursuri pentru a solicita obiecte și obiecte de răspuns. Utilizarea cadrului Express oferă control asupra tuturor obiectelor de solicitare și răspuns. [5]

Gateway de plată pentru website

Un gateway de plată procesează cardurile de credit în magazinele online. Transferă informațiile cheie între site-uri web / dispozitive mobile și procesatorii de plăți / bănci și invers. Practic, este o reprezentare online a unui terminal real de punct de vânzare pe care îl vedem în magazinele offline. Pentru a asigura securitatea tranzacțiilor și pentru a proteja site-ul web de fraudă, un gateway de plată criptează toate informațiile sensibile: numărul cardului de credit, data de expirare și codul CVV. Există două tipuri de gateway-uri de– negăzduit și găzduit.

Gateway-uri de plată găzduite

Aceste gateway-uri îndepărtează clientul de pe site pentru a finaliza plata și apoi îl redirecționează înapoi odată ce procesul este finalizat. Ei au grijă de securitatea tranzacțiilor și sunt destul de ușor de integrat în site-ul web. Cele mai cunoscute gateway-uri găzduite sunt PayPal, Wise (fostă TransferWise), Amazon Payments, Stripe, SagePay.

Gateway-uri de plată negăzduite (integrate).

Gateway-uri de plată negăzduite (integrate) sunt acele gateway-uri care se integrează fără probleme în site și cu ajutorul cărora utilizatorul finalizează plata direct pe website. Securitatea vine ca un dezavantaj cu soluțiile negăzduite, deoarece funcționează pe serverul website-ului și stochează de obicei informațiile despre cardurile de credit ale clienților. Astfel, trebuie să fie asigurată protecția împotriva fraudei, stocarea securizată a informațiilor deținătorului de card și conformitatea cu PCI (de preferință). Marea majoritate a gateway-urilor negăzduite vin cu API-uri extinse și module ușor disponibile pentru a fi integrate în coșul de cumpărături. Cu toate acestea, e nevoie programare personalizată pentru integrare, ceea ce înseamnă costuri adiționale pentru afacere. Unele dintre gateway-urile de plată negăzduite: Authorize.net, SagePay Direct, MangoPay, etc.

Implementarea coșului de cumpărături

Partea unui site web cu care utilizatorul interacționează direct se numește front end. Este, de asemenea,

denumită „partea client” a aplicației. Include tot ceea ce utilizatorii experimentează direct: culori și stiluri de text, imagini, grafice și tabele, butoane, culori și meniul de navigare. HTML, CSS și JavaScript sunt limbile folosite pentru dezvoltarea front-end. Structura, designul, comportamentul și conținutul a tot ceea ce se vede pe ecranele browser-ului atunci când site-urile web, aplicațiile web sau aplicațiile mobile sunt deschise sunt implementate de dezvoltatorii front-end. Receptivitatea și performanța sunt două obiective principale ale front-end-ului. Dezvoltatorul trebuie să se asigure că site-ul este receptiv, adică apare corect pe dispozitivele de toate dimensiunile, nicio parte a site-ului nu ar trebui să se comporte anormal, indiferent de dimensiunea ecranului.

Backend-ul este partea de server a site-ului. Stochează și aranjează datele și, de asemenea, se asigură că totul pe partea client a site-ului web funcționează bine. Este partea site-ului web pe care nu o puteți vedea și cu care nu puteți interacționa. Este porțiunea de software care nu vine în contact direct cu utilizatorii. Părțile și caracteristicile dezvoltate de designerii backend sunt accesate indirect de către utilizatori printr-o aplicație front-end. Activități, cum ar fi scrierea de API-uri, crearea de biblioteci și lucrul cu componente ale sistemului fără interfețe cu utilizatorul sau chiar sisteme de programare științifică, sunt de asemenea incluse în backend.

Implementarea front-end

Pagina principală conține logo-ul magazinului, lista produselor în vânzare și un menu în partea dreaptă. La îndreptarea cursorului deasupra fiecărui produs, apare opțiunea de adăugare a produsului în coșul de cumpărături. La adăugarea produsului în coș, se modifică numărul de produse în coș.

Pagina afișează produsele din coșul de cumpărături. Partea header e la fel ca și pe pagina principală. În coșul de cumpărături, avem indicat fiecare produs adăugat – imaginea, numele, prețul în lei, cantitatea, prețul total al produsului și prețul total pentru toate produsele.

Figura 1. Pagina principală.

Datorită funcționalităților adăugate cu ajutorul JavaScript în fișierul main.js, putem modifica cantitatea produsului, la fel și șterge produsul din coș. La fiecare modificare, se ajustează nu doar cantitatea dar și prețul total.

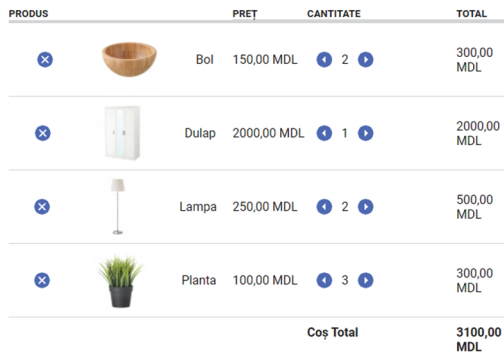

Figura 2. Pagina coșului de cumpărături

Implementarea back-end

Primul pas pentru implementarea back-end a fost instalarea node.js de pe website-ul oficial. Pentru verificarea versiunii node.js sau dacă acesta a fost instalat, putem rula în terminal node –v. După acest pas, am creat fișierul server.js. În terminalul VS Code, am rulat comanda: npm init -y. Această comandă creează fișierul package.json, care mai apoi ne permite să instalăm alte pachete/dependențe.

Următorul pas a fost instalarea severului/express: npm i express. [6]

Organizarea ajută cu adevărat la menținerea coerenței, mai ales într-o echipă mai mare. Consecvența în structura proiectului echivalează cu predictibilitatea

locului în care se poate găsi codul, ceea ce, la rândul său, ajută la productivitatea întregii echipe. Se recomandă de implementat întotdeauna lucrurile ușor previzibile, cu o structură consistentă pentru a minimiza sau elimina presupunerile. În rezultat vom avea un cod ușor de citit și înțeles pentru ceilalți colegi din domeniu. Structura fișierelor după implementarea back-end:

```
>controllers
    pagesCtrlFile.js
    productsCtrlFile.js
>node_modules
>public
    >fonts
    >images
    main.js
    styles.css
>routes
    pages.js
    products.js
>views
    cart.hbs
    index.hbs
package-lock.json
package.json
products.jsserver.js
```

Primul folder este controllers, care va găzdui toate controlerele necesare pentru aplicație. Aceste metode de controler primesc cererea de la rute și le convertesc în răspunsuri HTTP folosind orice middleware, după cum este necesar.

Routes - rutarea determină modul în care o aplicație răspunde la o solicitare a clientului către un anumit punct final. De exemplu, un client poate face o solicitare http GET, POST, PUT sau DELETE pentru diferite adrese URL. În continuare, avem folderul ROUTES care va avea un singur fișier pentru fiecare set logic de rute. De exemplu, pot exista rute pentru fiecare tip de resursă.

Views - acest folder include fișierele html, pentru pagina principală și pentru pagina coșului de cumpărături, doar că a fost schimbată extensia din .html în .hbs (comanda în terminal pentru instalarea hbs npm i hbs).

Public – Images: folderul cu imaginile folosite, în acest caz, imaginile produselor. Main.js: fișierul JS principal, care conține funcțiile care definesc comportamentul paginilor și a tuturor elementelor la interacțiunea cu utilizatorul. Styles.css: fișierul css pentru stilizarea paginilor și elementelor.

Products.js - Acest fișier include lista tuturor produselor.

Server.js

```
const express = require('express');
const path = require('path');
const app = express();
const publicDirectory = path.join (__dirname, './public');
app.use(express.static(publicDirectory));
app.set('view engine', 'hbs');
app.use('/', require('./routes/pages'));
app.use('/products', require('./routes/products') );
app.listen(5000, () => {
console.log("Server is running on port 5000");})
```

În prima linie de cod, am folosit *funcția require* pentru a include „modulul expres”. Înainte de a începe să folosim modulul expres, trebuie să facem un obiect din acesta.

Pentru vizualizarea proiectului, în terminalul VS Code rulăm *comanda npm* start. Vom vedea mesajul din cod: *Server is running on port 5000*. Numărul portului se poate schimba în cod. Respectiv, deschidem *http://localhost:5000/* și se va afișa respectiv pagina principală.

Concluzii

În această lucrare am analizat cum cu ajutorul limbajului de programare JavaScript putem adăuga produsele în coș, cum adăugăm funcțiile pentru modificarea numărului de produse în coș, ștergerea produselor din coș și pentru ajustarea prețului total.

Pentru ca toate acestea să funcționeze, este necesară stocarea informației pe server și manipularea acestora cu tehnologii precum Node.js, Express.js, etc.

ENG: In this paper we have analyzed how using the JavaScript programming language we can add products to the cart, how we add functions to change the number of products in the cart, delete products from the cart and adjust the total price. For all this to work, it is necessary to store the information on the server and manipulate it with technologies such as Node.js, Express.js, etc.

Baza metodologică

Drept bază metodologică și teoretico-ştiinţifică al lucrării, au servit lucrările de profil ale cercetătorilor și specialiștilor din cadrul comunității Web din S.U.A și Rusia, precum și cărțile HTML & CSS și Java script & JQUERY de Jon Duckett.

ENG: The methodological and theoretical-scientific basis of the papers served as the profile papers of researchers and specialists in the web community in the U.S. and Russia, as well as HTML & CSS and Java script & JQUERY books by Jon Duckett.

Referințe

[7] Jon Duckett, HTML & CSS design and build websites.

[8] https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview

[9] Jon Duckett, JavaScript & JQUERY interactive front-end web development

[10] https://www.simplilearn.com/tutorials/nodejs-tutorial/what-is-nodejs

[11] https://blog.logrocket.com/organizing-express-js-project-structure-better-productivity/

[12] https://docs.npmjs.com/cli/v8/configuring-npm/package-json