Jegyzőkönyv

Web technológia alapjai

Burgonya Webshop

Készítette: Koubridis Michael

Neptunkód: LWUIJ2

Dátum: 2024.05.13.

Tartalomjegyzék

1. Bevezetés

A Krumpliárusító Webshop (BK Borsod Krumpli) célja egy online platform létrehozása, amely kizárólag krumpli értékesítésre specializálódik. A webshop lehetőséget nyújt a felhasználóknak, hogy kényelmesen böngésszenek és vásároljanak különböző krumplifajtákból, közvetlenül a termelőktől, készítőktől.

1.1. A webáruház célja

A projekt alapvető célja, hogy létrehozzon egy dedikált online értékesítési pontot, amely kifejezetten a különböző típusú és fajtájú krumplikról szól. Ezáltal a webshop egy egyedülálló piacteret biztosít a krumpli termelőknek és vásárlóknak. A cél, hogy a webshop a legjobb minőségű krumplit kínálja megfizethető áron, miközben fenntartja a felhasználóbarát böngészési és vásárlási élményt.

1.2. Áttekintés

A BK webshopnál érdemes kiemelni az oldal főbb funkcióit, amelyek a felhasználói interakciókat és a vásárlási folyamatot támogatják. Ezek közé tartozik a terméklistázás, a kosár és pénztár, valamint a felhasználói fiókok kezelése. Ezen funkciók mindegyike kulcsfontosságú a zökkenőmentes vásárlási élmény biztosítása szempontjából, amely hozzájárul a webshop sikeréhez és a vásárlói elégedettség növeléséhez.

2. Mappa struktúra

A webáruház forráskódjának mappa struktúrája jól szervezett, ami segíti a fejlesztési folyamatot és a karbantartást.

Főbb mappák és azok tartalmai

- CSS
- Images
- Javascript
    - Index.js
    - JQuery.js
- Cart-page.html
- Checkout.html
- Index.html
- Product-list.html

- Products.json
- Register-page.html

2.1. CSS

A `CSS` mappa tartalmazza az összes stíluslapot, amelyek a webáruház vizuális megjelenését definiálják. A `style.css` fájl az oldalak általános kinézetét állítja be, beleértve a betűtípusokat, színeket és elrendezést.

2.2. Images

Az `Images` mappa a webáruházban használt összes képet tartalmazza. Ide tartoznak a termékfotók, ikonok és egyéb grafikák, mint például:

- `banner-model.png`: A főoldalon megjelenő banner kép.
- `heart-icon.png`: Kedvencek ikonja.
- `cart.png`: Bevásárlókosár ikonja.

2.3. JavaScript

A `JavaScript` mappa a dinamikus funkciókat biztosító JavaScript fájlokat tartalmazza.

2.3.1. Index.js

Az `index.js` fájl a kezdőoldal interaktivitásáért felelős. Ez tartalmazza a termékadatok betöltéséhez szükséges kódokat, az AJAX kérések kezelését és a felhasználói interakciókat, mint például a termékek hozzáadása a kosárhoz.

2.3.2. JQuery.js

A `jQuery.js` fájl a jQuery könyvtár, amely egyszerűsíti a HTML dokumentumok kezelését, eseménykezelést és animációkat. Ebben a projektben a jQuery-t használják az AJAX alapú form adatok küldésére, hogy javítsák a felhasználói élményt az oldal újratöltése nélkül.

3. jQuery

3.1. AJAX kérések kezelése

A jQuery AJAX metódusai lehetővé teszik a fejlesztők számára, hogy aszinkron kéréseket hajtsanak végre a szerverre anélkül, hogy az oldal teljes újratöltésére lenne szükség. Ez a technika javítja a weboldalak sebességét és felhasználói élményét. Az AJAX kérések segítségével dinamikusan frissíthetjük a weboldal tartalmát válaszként a felhasználói interakciókra.

3.2. Form adatkezelés

A jQuery szintén nagyon hasznos eszköz a form adatok kezelésére. Lehetővé teszi, hogy egyszerűen gyűjtsük össze a formon megadott adatokat, és elküldjük azokat a szervernek AJAX kérések segítségével. Az alábbi példa bemutatja, hogyan kezelhetjük a form adatokat és küldhetjük el aszinkron módon.

4. Index.js

Az `index.js` fájl kulcsfontosságú szerepet tölt be a webáruház működésében, mivel ez felelős a termékadatok dinamikus lekérdezéséért és megjelenítéséért.

4.1. Fetch API használata

A Fetch API modern, promise alapú eszköz a webes API-kkal való aszinkron kommunikációra. Ez a technológia lehetővé teszi, hogy a `index.js` fájlban HTTP kéréseket indítsunk a szerver felé anélkül, hogy újratöltenénk az oldalt. A Fetch API használata javítja a weboldal teljesítményét, mivel csökkenti a szükségtelen oldalfrissítések számát és gyorsabbá teszi az adatbetöltést.

4.2. JSON adatok kezelése, szerver segítségével

Ennek a megvalósításához egy JSON node szerverhez szükséges repository-t vettem alapul amely segítségével fel tudtam tölteni a Weboldalamhoz szükséges DB tartalmakat. Ez azért hasznos mert, ha ebben a repositoryban új termék kerül hozzáadásra, akkor a pipeline-on keresztül automatikusan kihostolásra kerül. Ez a weboldalon az oldal forráskódjának való módosítása nélkül is láthatóvá válik.

5. BK Borsod Krumpli

5.1. Webáruház célja

A webáruház fő célja egy felhasználóbarát, online platform létrehozása, amely kizárólag krumplifélék árusítására specializálódik. Az áruház lehetővé teszi a felhasználók számára, hogy kényelmesen böngésszenek és vásároljanak különböző krumplifajtákból, közvetlenül a termelőktől.

5.2. Vásárlói kör

Az áruház elsősorban az otthoni fogyasztókra és a gasztronómiai élményeket keresőkre fókuszál. Kiemelt figyelmet fordít az egészségtudatos vásárlókra is, akik különféle krumplifajták között szeretnének választani.

5.3. Bejelentkezési folyamat

A webáruház bejelentkezési folyamatát egy felhasználóbarát modal (modális ablak) segítségével oldottuk meg. Ez a modal egy felugró ablak formájában jelenik meg, amely az oldal fölött helyezkedik el. Ezáltal a felhasználók a bejelentkezési folyamat során nem hagyják el az aktuális oldalt, ami nagymértékben javítja a felhasználói élményt és lehetővé teszi a gyors és zökkenőmentes bejelentkezést.

6. Példakódok

6.1 Termékek lekérése

Ez a rész bemutatja, hogyan történik a termékek adatok lekérése a webáruház backend rendszeréből. A folyamat magában foglalja az adatok aszinkron lekérdezését, amely lehetővé teszi a felhasználó számára, hogy folyamatosan frissülő tartalmat kapjon anélkül, hogy az oldal újratöltődne. Ez a módszer különösen hasznos nagy adatmennyiség kezelésekor, és javítja a felhasználói élményt a gyors és hatékony adatbetöltés révén.

```
async function showProducts(Url) {
  fetch("https://json-server-p686.onrender.com/products")
    .then((response) => response.json())
    .then((json) => {
      productsList = json;
      productsList.forEach((product) => {
        htmlToReturn = `<div class="col col-xl-4 col-lg-4 col-md-6 col-sm-12">
            <div class="prod-card mb-4" id='${product.id}'>
              <div class="icons d-flex justify-content-center" id="card_icons">
                <a class="heart"><img class='img1' src="Images/heart-icon-trans.png" alt="inner heart
icon"></a>
                <a href="Product-view.html"><img src="Images/eye-icon-trans.png" alt="inner eye icon"></a>
                <a class="shopping"><img src="Images/shopping-icon-trans.png" alt="inner shopping
icon"></a>
              </div>
              <img class="card-img-top" src="Images/${product.imageName}.png" alt="Card image cap">
              <div class="card-body d-flex flex-column align-items-center">
                <h5 class="card-title">${product.name}</h5>
                <p class="card-text mb-0"><strong>${product.priceAfterDiscount}</strong>
<del>${product.price}</del><span class="offer">(60%Off)</span></p>
                <div class="stars-group d-flex align-items-center mt-2" id="starsgroup">`;
        lowStar = 5 - Math.floor(product.ratings);
        for (i = 1; i <= product.ratings; i++) {
          reviews += `
```

```
                <img class="stars" src="Images/star.png" alt="star-rating" role="icon" aria-label='star
rating'>`;
        }


                                …
                        …
       document.getElementById("productListArea2").innerHTML += htmlToReturn2;
      });
      document.querySelectorAll(".prod-card").forEach((card) => {
        card.children[0].style.visibility = "hidden";
      });


                        …
                        …


      document.querySelectorAll(".prod-card").forEach((img) => {
        let img_3 = img.childNodes[1].childNodes[5].childNodes[0];
        img_3.addEventListener("mouseout", (func4) => {
          img_3.src = "Images/shopping-icon-trans.png";
        });
      });
    });
}
showProducts(productsListUrl);
```

6.2 JQuery használata

A JQuery egy rendkívül népszerű JavaScript könyvtár, amelyet az adatok dinamikus manipulálására és az AJAX kérések kezelésére használtam. A könyvtár egyszerűsíti a JavaScript kódokat, lehetővé téve a fejlesztők számára, hogy kevesebb kóddal, de nagyobb hatékonysággal implementálhassanak bonyolult funkciókat. A JQuery használatával a webáruház képes aszinkron módon kommunikálni a szerverrel, a felhasználói interakciókat finomhangolni, és a felület elemeit dinamikusan frissíteni.

```
$(document).ready(function () {
    $(".collapse.show").each(function () {
        $(this).prev(".card-header").find(".fa").addClass("fa-
minus").removeClass("fa-plus");
    });

    $(".collapse").on('show.bs.collapse', function () {
        $(this).prev(".card-header").find(".fa").removeClass("fa-
plus").addClass("fa-minus");
```

```
    }).on('hide.bs.collapse', function () {
        $(this).prev(".card-header").find(".fa").removeClass("fa-
minus").addClass("fa-plus");
    });
});


$('.carousel .carousel-item').each(function(){
    var minPerSlide = 3;
    var next = $(this).next();
    if (!next.length) {
        next = $(this).siblings(':first');
    }
    next.children(':first-item').clone().appendTo($(this));

    for (var i=0;i<minPerSlide;i++) {
        next=next.next();
        if (!next.length) {
            next = $(this).siblings(':first');
        }

        next.children(':first-item).clone().appendTo($(this));
    }
});
```