

Dokumentáció

Webshop részletes bemutatása

Tartalom

Bevezetés, a téma ismertetése, témaválasztás indoklása	3
Fejlesztői dokumentáció	5
1 Frontend.....	5
2 Backend	12
Tesztelés	23
Felhasználói dokumentáció	26
Főmenü	26
Bejelentkezés és Regisztráció	26
Elérhető menüpontok (vendég)	27
Kezdőlap	28
Áruház	28
Kosár	30
Kapcsolat.....	32
Elérhető menüpontok (regisztrált felhasználó)	33
Profil.....	33
Kezdőlap	34
Áruház	34
Kosár	36
Kapcsolat.....	39
Kijelentkezés	39
Összefoglalás.....	40

Bevezetés, a téma ismertetése, témaválasztás indoklása

Az elkészült projekt egy **teljes értékű webáruház**, amely lehetőséget biztosít a felhasználóknak termékek böngészésére, keresésére, kosárba helyezésére, megrendelésére, valamint a korábbi rendelések áttekintésére. A rendszer célja, hogy egy valós kereskedelmi környezethez hasonló online vásárlási élményt nyújtson, miközben az adminisztrátorok számára biztosítja a termékek karbantartását és a webáruház működtetéséhez szükséges alapfunkciókat.

A fejlesztés során elsődleges szempont volt, hogy a weboldal **felhasználóbarát**, átlátható és reszponzív legyen, valamint a **teljes vásárlási folyamatot** támogassa a termékek kiválasztásától egészen a rendelés leadásáig. A rendszer mind a vásárlói, mind az adminisztrátori oldal működését lefedi, így alkalmas lehet egy kisebb webáruház önálló üzemeltetésére is.

A választott téma indoklása

Az online vásárlás napjaink egyik legnépszerűbb és leggyorsabban fejlődő területe, amely mind a fogyasztók, mind a vállalkozások számára komoly jelentőséggel bír. Az elmúlt években a webáruházak forgalma folyamatosan növekedett, és egyre több cég helyezi át értékesítési tevékenységét online felületekre.

A téma választását több szempont indokolta:

- **Szakmai aktualitás:** az e-kereskedelem egy dinamikusan növekvő iparág, így a területhez kapcsolódó gyakorlati ismeretek rendkívül értékesek számunkra.
- **Gyakorlati hasznosíthatóság:** a megvalósított rendszer bármikor bővíthető és továbbfejleszthető egy vállalkozás igényeihez, akár biztonságos fizetési megoldásokkal, raktárkezeléssel vagy szállítási integrációkkal.
- **Személyes motiváció:** célunk egy olyan átfogó webalkalmazás létrehozása volt, amely egyszerre mutatja be a frontend és backend logikák működését, valamint egy jól strukturált adatbázis gyakorlati alkalmazását.

A rendszer által lefedett funkciók

A fejlesztés eredményeként elkészült weboldal az alábbi fő funkciókat biztosítja:

- **Felhasználói oldal:**
 - Termékek böngészése, keresése és ár szerinti rendezése.
 - Készletinformációk megjelenítése.
 - Termékek kosárba helyezése és rendelés leadása.
 - Profiloldal a felhasználói adatok kezelésére.
 - Korábbi rendelések megtekintése.
- **Adminisztrátori oldal:**
 - Termékek hozzáadása, szerkesztése és elérhetőségük módosítása.
 - Képfeltöltés termékekhez.

Alkalmazhatóság és További fejlesztési lehetőségek

Az elkészült projekt nem csupán gyakorlati célokat szolgál, hanem alkalmas lehet valós piaci környezetben történő alkalmazásra is kisebb méretű webáruházak esetén. Az egyszerűen kezelhető adminisztrációs felület és a jól strukturált vásárlói oldal lehetővé teszi, hogy a rendszer rövid idő alatt használatba vehető legyen.

A rendszer továbbfejlesztésének irányai közé tartozhat a felhasználói fiókok biztonságának erősítése (pl. jelszavak titkosított tárolása, kétszintű hitelesítés), a fizetési szolgáltatók integrációja, valamint a rendelések nyomon követésének beépítése. Ezek az elemek tovább növelnék a rendszer piacképességét és felhasználói élményét.

Fejlesztői dokumentáció

1 Frontend

1.1 Áttekintés

A jelen dokumentáció a **Webshop frontend** részét mutatja be, amely teljes értékű, működőképes felületet biztosít a felhasználók számára. A frontend célja, hogy a látogatók kényelmesen böngézhessenek a termékek között, hozzáadhassák azokat a kosárhoz, rendeléseket adhassanak le, és kezelhessék a felhasználói fiókjukat.

A fejlesztés során kiemelt figyelmet kapott:

- a **felhasználói élmény** egyszerűsége és áttekinthetősége,
- a **reszponzív design**, amely biztosítja a kompatibilitást különböző eszközökön,
- a **dinamikus állapotkezelés**, amely lehetővé teszi a valós idejű frissítéseket anélkül, hogy az oldal újratöltődne.

A frontend **JavaScript, HTML és CSS** kombinációjával készült, és az oldal komponensei moduláris felépítésűek, lehetővé téve a könnyű bővíthetőséget és karbantartást.

1.2 Komponensek és szervizek

1.2.1 Navbar (navigációs menü)

A **navbar** a weboldal navigációját biztosítja.

- **Feladat:** gyors hozzáférést biztosít az összes főoldalhoz, dinamikusan változik a felhasználó státusza szerint (bejelentkezett/nem bejelentkezett, admin/normál felhasználó).
- **Metódus:** renderNavbar() – a DOM elemeket frissíti a felhasználó állapota alapján.
- **Fő változók:** loggedInUser, isAdmin.
- **Megjegyzés:** a navbar fix pozícióban van, így minden oldalon látható marad.

1.2.2 Bejelentkezés és regisztráció

- **Feladat:** lehetővé teszi a felhasználók számára a fiók létrehozását, bejelentkezést és kijelentkezést. Biztosítja, hogy a felhasználói adatok a frontend állapotában tárolódjanak a munkamenet idejére.
- **Metódusok:**
 - renderLoginForm() – bejelentkezési űrlap megjelenítése.
 - loginUser(event) – bejelentkezési adatok ellenőrzése, állapot frissítése (loggedInUser).
 - renderRegisterForm() – regisztrációs űrlap megjelenítése.
 - registerUser(event) – új felhasználó létrehozása, validáció és adatbevitel a users tömbbe.

- `logoutUser()` – kijelentkezés, frontend állapot törlése, navbar frissítése.
- **Változók:** `users` (regisztrált felhasználók tömbje), `loggedInUser`, `loggedInUserData`.
- **Megjegyzés:** a komponens központi szerepet játszik az állapotkezelésben, mivel a bejelentkezett felhasználó státusza határozza meg a profiloldal, kosár és admin panel elérhetőségét.

1.2.3 Áruház

- **Feladat:** a termékek listázása, keresése és szűrése.
- **Metódusok:**
 - `renderStore(searchTerm, sortOrder, onlyAvailable)` – a termékeket megjeleníti rácsban, szűrési és rendezési opciókkal.
 - `showProduct(id)` – egy termék részletes nézetének megjelenítése.
- **Változók:** `products` (termékek tömbje), `filtered` (szűrt terméklista).
- **UI részletek:** a termékek rácsban jelennek meg, a képek, ár és elérhetőség látható, hover hatás a vizuális kiemeléshez.

1.2.4 Termék részletek

- **Feladat:** a kiválasztott termék adatait jeleníti meg, lehetőséget ad a kosárba helyezésre.
- **Metódus:** `showProduct(id)` – a DOM elemek frissítése a kiválasztott termék adataival.
- **Kosár funkció:** `addToCart(productId)` – az adott termék hozzáadása a kosárhoz.
- **Változók:** a termék objektum, a kosár tömb (`cart`).

1.2.5 Kosár és rendelés

- **Feladat:** a kosár tartalmának kezelése, rendelés leadása.
- **Metódusok:**
 - `addToCart(productId)` – termék hozzáadása, mennyiség növelése, ha már létezik a kosárban.
 - `renderCheckout()` – kosár oldalt jeleníti meg, összesítve az árakat.
 - `autoFill()` – bejelentkezett felhasználó adatait automatikusan kitölti a rendelési űrlapon.
 - `submitOrder(event)` – rendelés leadása, adatok mentése a rendelési tömbbe (`orders`).
- **Változók:** `cart`, `orders`.

- **Megjegyzés:** minden kosárművelet dinamikusan frissíti a felületet anélkül, hogy újratöltődne az oldal.

1.2.6 Profil oldal

- **Feladat:** felhasználói adatok megtekintése és szerkesztése, rendelések listázása.
- **Metódusok:**
 - editProfile() – adatmezők szerkeszthetővé tétele.
 - saveProfile(event) – módosítások mentése a loggedInUserData objektumba.
 - showOrders() – leadott rendelések megjelenítése táblázatban.
- **Változók:** loggedInUserData, orders.
- **Megjegyzés:** a profiloldal biztonsági és UI szempontból is figyelembe veszi az admin/non-admin státuszt.

1.2.7 Admin oldal

- **Feladat:** termékek adminisztrációja – hozzáadás, szerkesztés, törlés.
- **Metódusok:**
 - renderAdminPanel() – admin funkciók felületének generálása.
 - addProduct(event) – új termék hozzáadása a products tömbhöz.
 - editProduct(idx) – termék szerkesztése az index alapján.
 - saveProduct(event, idx) – módosított termék mentése.
- **Változók:** products.
- **Megjegyzés:** az admin felület külön komponens, csak admin felhasználók érhetik el.

1.3 Változók és állapotkezelés

A frontend dinamikus működését **globális változók és állapotok** biztosítják.

Változó	Típus	Feladat
products	tömb	az összes termék tárolása
cart	tömb	kosár tartalma
orders	tömb	leadott rendelések
loggedInUser	string	bejelentkezett felhasználó email
loggedInUserData	objektum	felhasználói adatok

Változó	Típus	Feladat
isAdmin	boolean	admin jogosultság jelzése
Változó / Elem	Típus	Leírás / Funkció
products	tömb	Az áruházban elérhető termékek listája, minden elem tartalmazza: id, name, price, image, available
cart	tömb	A felhasználó kosarában lévő tételek listája, minden elem tartalmazza a terméket és a mennyiséget
loggedInUser	string	Az aktuálisan bejelentkezett felhasználó e-mail címe
loggedInUserData	objektum	A felhasználó profiladatai és rendelési előzményei
isAdmin	boolean	Jelzi, hogy az aktuális felhasználó admin-e
searchTerm	string	Keresési kulcsszó az áruház termékeiben
sortOrder	string	Rendezés iránya az áruházban (asc, desc)
onlyAvailable	boolean	Csak raktáron lévő termékek megjelenítése
quantity	number	Kosárba helyezendő termék mennyisége (input alapján)
total	number	Kosárban lévő tételek összértéke
userOrders	tömb	Az aktuális felhasználó összes rendelése
fileInput	File	Új vagy szerkesztett termék képfájla a feltöltéshez

Változó	Típus	Feladat
content	DOM elem	Referencia a fő tartalom (<div id="content">) megjelenítésére
document.getElementById("login-email")	DOM elem	Bejelentkezési e-mail input
document.getElementById("login-password")	DOM elem	Bejelentkezési jelszó input
document.getElementById("reg-email")	DOM elem	Regisztrációs e-mail input
document.getElementById("reg-name")	DOM elem	Regisztrációs név input
document.getElementById("reg-zip")	DOM elem	Regisztrációs irányítószám input
document.getElementById("reg-city")	DOM elem	Regisztrációs város input
document.getElementById("reg-street")	DOM elem	Regisztrációs utca input
document.getElementById("reg-house")	DOM elem	Regisztrációs házsám input
document.getElementById("quantity")	DOM elem	Kosárba helyezendő termék mennyisége input
document.getElementById("sort")	DOM elem	Rendezés select mező az áruházban
document.getElementById("onlyAvailable")	DOM elem	Csak raktáron lévő checkbox az áruházban
document.getElementById("order-name")	DOM elem	Szállítási név input a rendelésnél
document.getElementById("order-zip")	DOM elem	Szállítási irányítószám input a rendelésnél
document.getElementById("order-city")	DOM elem	Szállítási város input a rendelésnél
document.getElementById("order-street")	DOM elem	Szállítási utca input a rendelésnél
document.getElementById("order-house")	DOM elem	Szállítási házsám input a rendelésnél
document.getElementById("payment-method")	DOM elem	Fizetési mód select input

Változó	Típus	Feladat
document.getElementById("auto-fill")	DOM elem	Checkbox a rendelési adatok automatikus kitöltésére
idx	number	Tömb index a termékek vagy rendelések kezeléséhez
p	objektum	Lokális változó egy termékhez (pl. editProduct, saveProduct)
event	objektum	Esemény objektum az eseménykezelőkben (submit, click)

Az állapotkezelés **valós idejű DOM frissítéssel** történik, minden felhasználói interakció azonnali vizuális visszajelzést ad. A változók a frontend logika szívévé képezik, biztosítva az adatok konzisztenciáját.

1.4 Tesztelés

1.4.1 Statikus teszt

- Ellenőrizve, hogy a **termékek helyesen jelennek meg rácsban**.
- **Formok** (bejelentkezés, regisztráció, rendelés) megfelelően validálnak kötelező mezőket.
- CSS elemek (gombok, hover hatás, rács) vizuálisan megfelelnek a terveknek.

1.4.2 Dinamikus teszt

- Kosárba helyezés és mennyiségszámítás helyes működése.
- Rendelési folyamat hibamentes működése (adatok mentése, megrendelés összesítés).
- Admin panel termék-hozzáadás és szerkesztés, változások azonnali frissítése.
- Felhasználói profil adatainak módosítása és rendelési előzmények megjelenítése.
- Bejelentkezés helyes működése, bejelentkezést követően a „Bejelentkezés” és „Regisztráció” helyett a „Profil” és „Kijelentkezés” menüpontok sikeresen megjelennek.

A tesztek lefedik a legfontosabb **CRUD műveleteket**, a frontend interaktivitását és a felhasználói állapotkezelést.

1.5 Felhasználói interakciók

- **Gombok és onclick események:** minden fő funkció (termék részletek, kosár, profil, admin panel) azonnali, vizuális visszajelzéssel működik.
- **Dinamikus DOM frissítés:** innerHTML használatával minden oldalon frissül a tartalom, minimalizálva a teljes oldal újratöltését.
- **Form validáció:** beviteli mezők ellenőrzése JavaScript segítségével, kötelező mezők és helyes formátum.
- **Reszponzív rács:** store-grid automatikusan alkalmazkodik különböző képernyőméretekhez, az auto-fit és minmax CSS szabályok biztosítják a megfelelő elrendezést.

1.6 Továbbfejlesztési lehetőségek

- **Fizetés:** Többféle fizetési mód és szállítási lehetőség hozzáadása.
- **UX/UI fejlesztések:** animációk, hover-effektek, jobb vizuális visszajelzés a kosárban vagy a gomboknál.
- **Űrlapok validációjának bővítése:** dinamikus hibajelzés, karakterlimitek, azonnali visszajelzés a felhasználónak.
- **Kosár funkciók bővítése:** mennyiség módosítása a kosárban, termék eltávolítása, összesített kedvezmények, kuponkódok kezelése.
- **Profiloldal fejlesztése:** avatar feltöltés, preferenciák beállítása, rendelések szűrése dátum szerint.
- **Admin panel továbbfejlesztése:** tömeges termékfeltöltés CSV-ből/Excelből, termékek gyors szerkesztése inline, vizuális statisztikák (pl. legkelendőbb termékek).
- **Frontenden futó keresőoptimalizálás:** pl. gyorsabb szűrés, debouncing kereséshez, autocomplete.
- **Tematikus megjelenítési lehetőségek:** sötét mód, termékkategóriák szerinti dinamikus szűrés, vizuális kiemelések (akciók, újdonságok).

2 Backend

2.1 Áttekintés

A **Producthor** egy oktatási/portfólió célú, tipikus webáruház backend (REST API) Java 17 + Spring Boot ökoszisztémán. A rendszer entitásai: **User, Product, Category, Order, OrderItem, ShippingData**. A szolgáltatás JWT alapú stateless autentikációt használ, admin/user szerepekkel. A képfeltöltés lokális fájlrendszerre történik, a feltöltött állományok HTTP-n publikálhatók.

A dokumentum célja, hogy szakmai mélységben bemutassa a kivitelezést: használt technológiák, környezet, adatmodell, fő folyamatok/algorithmusok, biztonság, konfiguráció, futtatás és bővítési lehetőségek.

2.2 Felhasznált technológiák

- **Java 17** – LTS, modern nyelvi funkciók (records helyett most DTO-k/Lombok), jó ökoszisztéma.
- **Spring Boot (3.x)** – gyors indítás, autokonfiguráció, Actuator-kompatibilitás (opcionális).
- **Spring Web / MVC** – REST kontrollerek (@RestController), request mappingek.
- **Spring Data JPA / Hibernate** – ORM, deklaratív repository-k, @Transactional.
- **Spring Security** – stateless biztonsági lánc, role alapú jogosultság, JWT integráció.
- **JWT (jjwt)** – aláírt hozzáférési tokenek.
- **PostgreSQL 16** – ACID RDBMS, bő funkciók, megbízható tranzakciókezelés.
- **Maven** – build, dependency management, test futtatás.
- **Lombok** – boilerplate csökkentése (@Getter/@Setter/@Builder/...).
- **MockMvc / Spring Test** – kontrollerek és szolgáltatások tesztjeihez.

2.3 Fejlesztőkörnyezet

2.3.1 Hardver

- Fejlesztői gép: átlagos modern laptop/desktop (8–16 GB RAM, SSD ajánlott).

- Indoklás: Spring Boot + PostgreSQL fejlesztői környezethez ez teljesen elegendő; a tesztek és lokális DB futtatása kényelmes.

2.3.2 Szoftver

- **Java 17 JDK**
- **Docker** (opcionális) PostgreSQL futtatására:
`docker run --name producthor-postgres -e POSTGRES_DB=producthor -e POSTGRES_USER=producthor -e POSTGRES_PASSWORD=producthor -p 5433:5432 -v producthor_pg_data:/var/lib/postgresql/data -d postgres:16`
- **IDE:** IntelliJ IDEA / VS Code Java / Eclipse – tetszőleges.
- **Maven:** `mvn clean verify / mvn spring-boot:run`
- **PostgreSQL kliens** (psql/DBeaver) – sémák/lekérdezések ellenőrzésére.

2.4 Konfiguráció

application.properties (kiemelt kulcsok):

`spring.application.name=Producthor`

`server.port=8080`

`spring.datasource.url=jdbc:postgresql://localhost:5433/producthor`

`spring.datasource.username=producthor`

`spring.datasource.password=producthor`

`spring.jpa.hibernate.ddl-auto=update`

`spring.jpa.show-sql=true`

`spring.jpa.properties.hibernate.format_sql=true`

`spring.jpa.open-in-view=false`

`security.jwt.secret=Bcx56JFth71jkl20Bcx56JFth71jkl20`

security.jwt.expiration-ms=3600000

spring.servlet.multipart.max-file-size=5MB

spring.servlet.multipart.max-request-size=5MB

app.upload.dir=uploads

app.public.base-url=/files

- **open-in-view=false:** N+1 és lazy init problémák csökkentése service/DAO rétegben megoldott adat-hozzáférést feltételez.
- **CORS:** WebCorsConfig korlátozott eredetre (Angular FE: http://localhost:4200).

2.5 Architektúra és rétegek

Hagyományos, rétegzett felépítés:

- **Controller:** HTTP végpontok, REST kontraktusok (DTO be/ki).
- **Service:** üzleti logika, tranzakciókezelés, érvényesítések, map-pelések meghívása.
- **Repository:** JPA alapú DAO, deklaratív queryk (pl. findByAvailableTrue()).
- **Mapper:** entitás ↔ DAO/DTO konverziók, update/merge logika.
- **Security:** SecurityConfig, JwtAuthenticationFilter, JwtTokenService.

Stateless API (session nélküli), minden kérés opcionálisan JWT-vel biztosítható.

2.6 Adatmodell

2.6.1 Entitások és kapcsolatok (részletek)

- **User**
 - username (unikális), password (BCrypt), name, isAdmin (ROLE_ADMIN/USER).
 - **1–1** *ShippingData* (opcionális) – a felhasználó alapértelmezett címe.
 - **1–N** *Order* (orderHistory).

- **ShippingData**
 - Alap címmezők: postalCode, city, street, houseNumber, additionalInfo.
- **Category**
 - name (unikális), description.
 - **1-N** *Product*.
- **Product**
 - name, description, price, imageUrl, available.
 - **N-1** *Category* (nullable).
 - **ElementCollection** specifications: Map<String,String> külön táblában.
- **Order**
 - **N-1** *User* (nullable – vendég rendelés támogatott).
 - **N-1** *ShippingData* (nem unique, több order hivatkozhat ugyanarra a címre).
 - **1-N** *OrderItem* (cascade + orphanRemoval).
 - totalGross számított mező (service réteg állítja).
- **OrderItem**
 - **N-1** *Order*, **N-1** *Product*.
 - quantity, unitPriceAtPurchase (pillanatnyi ár snapshot).

Megjegyzés a sémáról: a korábbi **unique** constraint az orders.shipping_data_id-n eltávolításra került, így több rendelés is hivatkozhat azonos címre (reorder, ismétlődő használat).

2.6.2 DAO-k és DTO-k

- **DAO:** kliensnek visszaadott, serializálható "view model" (pl. ProductDao, OrderDao).
- **DTO:** bemeneti adatok (pl. ProductDto, OrderDto, AuthenticationDto).

Mapper-ek (pl. ProductMapper, OrderMapper, stb.) végzik az oda-vissza átalakítást és "null-skip" frissítést (updateEntityFromDto: csak nem-null mezőket ír felül).

2.7 Biztonság

2.7.1 HTTP végpontok és jogosultság

- `"/api/*/public/**"`: nyilvános (pl. login, regisztráció, termék listázás).
- `"/api/*/private/**"`: autentikáció szükséges (JWT).
- `"/api/*/admin/**"`: `ROLE_ADMIN` szükséges.
- `GET /files/**` nyilvános (feltöltött képek kiszolgálása).

2.7.2 JWT lánc

- **JwtAuthenticationFilter**: kibontja a Bearer token, ellenőrzi JwtTokenService-szel, a SecurityContext-be teszi a felhasználót.
- **JwtTokenService**: generálás (sub, roles, exp), ellenőrzés (aláírás, lejárát, subject).
- **Stateless**: nincs HTTP session; minden kérés hordozza a token.

2.8 Fő folyamatok / algoritmusok

2.8.1 Autentikáció / bejelentkezés

1. **Login** (`/api/auth/public/login`): AuthenticationManager ellenőrzi user/pass-t.
2. Siker esetén **JWT** készül a felhasználó szerepeivel; response: token + meta.

2.8.2 Képfeltöltés

- **FileStorageService.store**:
 - Content-Type ellenőrzés (jpeg/png/webp).
 - Biztonságos fájlnev (UUID + kiterjesztés).
 - `app.upload.dir` alá mentés, majd **publikus URL** visszaadása (`/files/{uuid.ext}`).
- **StaticResourceConfig**: a uploads mappát HTTP-n publikálja.

2.8.3 Termék létrehozás / módosítás

- ProductService.create/update:
 - Kötelező validációk (név, ár).
 - Kategória feloldása (ha adott).
 - Specifikációk teljes felülírása update-nél, ha DTO-ban jöttek.
 - available default: true (ha hiányzik).
 - Persistálás, egyszerű BaseResponse.

2.8.4 Rendelés létrehozása

- OrderService.create(OrderDto):
 - Üres lista → ERROR.
 - Shipping kiválasztás:
 - shippingDataId → feloldás; ha nincs, ERROR.
 - ellenkező eset: inline ShippingDataDto → entitássá, mentés.
 - Order összeállítása:
 - minden tételnél Product feloldás (ERROR, ha nincs),
 - unitPriceAtPurchase a termék aktuális ára,
 - quantity ≥ 1 kötelező.
 - totalGross = $\Sigma(\text{unitPriceAtPurchase} * \text{quantity})$.
 - Mentés, visszaadott OrderDao (items, shipping, total, username).

Megjegyzés: A vendég rendelés (user = null) támogatott, autentikált user esetén a username is a DAO része (mapper tölti ki).

2.8.5 Felhasználói profil frissítése

- UserService.update(UserDto):
 - Authenticated user feloldása.
 - name frissítés.
 - shippingData:

- ha hiányzott → új ShippingData entitásból felvétele,
- ha volt → mezőszintű frissítés (null-skip).
- Mentés, OK/ERROR visszajelzés.

2.8.6 Kategória törlése

- CategoryService.delete:
 - Létezés ellenőrzés,
 - **ProductRepository.clearCategoryFromProducts** (FK megszüntetése a termékeken),
 - Kategória törlése.

2.9 REST API kivonat (minták)

- **Auth**
 - POST /api/auth/public/register – { username, password }
 - POST /api/auth/public/login – → { token, isAdmin }
- **Product**
 - POST /api/product/admin/create – Admin
 - PUT /api/product/admin/update/{id} – Admin
 - GET /api/product/public/get?id=...
 - GET /api/product/public/all
 - GET /api/product/public/availables
 - GET /api/product/public/byCategory?categoryId=...
- **Category**
 - GET /api/category/public/getAll
 - POST /api/category/admin/create / PUT /admin/update/{id} / DELETE /admin/delete/{id} – Admin
- **Order**

- POST /api/order/public/create – vendég is rendelhet (shipping kötelező)
- **User**
 - POST /api/user/private/update – profil mentése
 - GET /api/user/private/get – bejelentkezett user + rendelési előzmények
 - GET /api/user/admin/all – Admin
 - DELETE /api/user/admin/delete/{id} – Admin
- **File**
 - POST /api/file/admin/upload – Admin, 5 MB limit, url-t ad vissza

2.10 Futtatás & fejlesztői életciklus

2.10.1 Adatbázis

- Dockerrel indítva (PostgreSQL 16): lásd fent.
- Alap beállítások: producthor/producthor, DB: producthor, port: 5433.

2.10.2 Backend indítása

mvn clean spring-boot:run

vagy

mvn clean package && java -jar target/*.jar

2.10.3 Alap seed / manuális ellenőrzés

- **Regisztráció:** POST /api/auth/public/register
- **Login** → JWT
- **Admin tevékenységek:** állíts be egy admin usert (DB-ben is_admin = true), majd termék/kategória létrehozás.

2.11 Validáció, hibakezelés

- Input-validáció a service rétegben (null/blank/ár > 0, stb.).
- Hibák esetén **BaseResponse("ERROR", "...")** mint egységes visszajelzés.

- Az API design a kliens egyszerű kezelését célozza: nincsenek kivételdobások választestben; strukturált code/message érkezik.

2.12 Teljesítmény, skálázhatóság

- **Stateless** API → horizontális skálázásra alkalmas.
- **DB indexek:** idegen kulcsokon és gyakori keresési mezőkön (pl. `idx_orders_user_id`).
- **Képfájlok:** lokális fájlrendszer – kis projekthez megfelelő; éles környezetben CDN/objektktár (S3, GCS) javasolt.
- **Specifikációk:** @ElementCollection(EAGER) – kisméretű mapekhez kényelmes; ha nagyra bővülne, külön entitásra érdemes váltani LAZY fetch-csel.

2.13 Biztonsági megfontolások

- **Jelszavak:** BCrypt.
- **JWT:** titok kulcs `.properties`-ben (demo), éles környezetben **vault / env**.
- **CORS:** célzott eredetek, fejlesztői állapotban `localhost:4200`.
- **Képfeltöltés:** content-type ellenőrzés, kiterjesztés-becslés, path traversal védelem, random fájlnev.
- **Open redirect / CSRF:** stateless API, CSRF kikapcsolva (cookie nélküli auth), `httpBasic` tiltva.

2.14 Bővítési lehetőségek

- **Rendelés státuszok** (PLACED, PAID, SHIPPED, CANCELLED) + események.
- **Kedvezmények / kuponok**, szállítási díj kalkuláció.
- **Képtárméretezés / thumbnail** generálás feltöltéskor.
- **Kereső / szűrő** a backendben (név, kategória, available, árintervallum) — paginált.
- **Audit log:** ki mit módosított (Spring Data Envers, saját audit trail).
- **Email/SMS értesítés** rendelés leadásakor (outbox pattern).
- **Specifikációk** normalizálása (külön táblák, típusok, validációk).

2.15 Példák (curl)

Login:

```
curl -X POST http://localhost:8080/api/auth/public/login \
-H "Content-Type: application/json" \
-d '{"username":"john","password":"pw"}
```

Új termék (admin JWT szükséges):

```
curl -X POST http://localhost:8080/api/product/admin/create \
-H "Authorization: Bearer <JWT>" \
-H "Content-Type: application/json" \
-d '{"name":"Phone","price":199990,"available":true}'
```

Rendelés (vendég):

```
curl -X POST http://localhost:8080/api/order/public/create \
-H "Content-Type: application/json" \
-d '{
  "items":[{"productId":1,"quantity":2}],
  "shippingData":{"postalCode":"4400","city":"Nyh","street":"U","houseNumber":"1","additionalInfo":""}
}'
```

2.16 Tesztelhetőség és tesztstratégia röviden

- **Unit:** mapper-ek, szolgáltatások (mockolt repo-kkal).
- **Slice / MVC:** kontrollerek @WebMvcTest + MockMvc.
- **Integrációs:** @SpringBootTest valós JPA réteggel és in-memory/lokális PG-vel.
- **Represszió:** korábban javított hibákra célzott teszt (pl. shippingData több rendeléshez).

(A részletes teszt-dokumentáció külön anyagban található, már elkészült.)

2.17 Üzemeltetés, loggolás

- **Spring Boot logging** (konzol), fejlesztéskor SQL log látható (show-sql=true).
- Élesben ajánlott: strukturált log (JSON), request-id, külön logger kategóriák (security, persistence).

Tesztelés

Bevezetés

A tesztelés célja annak bemutatása, hogy a rendszer az elvárt körülmények között és különböző edge-case helyzetekben is helyesen működik. A projektben több réteget is lefedtünk automatizált tesztekkel, valamint manuális kipróbálásokkal ellenőriztük a frontend és a backend integrációját.

A tesztelés fókuszában az állt, hogy a fejlesztett webáruház:

- stabilan működjön különböző hardver- és szoftverkörnyezetekben,
- helyesen kezelje a hibás bemeneteket,
- megfelelő teljesítményt nyújtson nagyobb terhelés mellett is,
- valamint minden fő funkcióját (termékkezelés, rendelés, felhasználói profil) bizonyíthatóan ellássa.

A tesztelés során **statikus tesztelési módszereket**, **dinamikus (futásidejű) teszteket**, valamint **stresszteszteket** is alkalmaztunk.

Tesztkörnyezet

Hardver

A rendszer fejlesztése és tesztelése az alábbi hardverkörnyezetben zajlott:

- **Fejlesztői gép:** AMD Ryzen 7700x, 64 GB RAM
- **Operációs rendszerek:** Windows 11 és Ubuntu 22.04 LTS

Szoftver

- **Backend:** Spring Boot 3.5.5, Java 17, PostgreSQL 16 adatbázis
- **Frontend:** Angular 17, Node.js 20, böngészők: Google Chrome (legfrissebb), Firefox, Edge
- **Tesztkeretrendszerek:**
 - JUnit 5
 - Spring Boot Test (MockMvc, @DataJpaTest, @SpringBootTest)
 - Mockito

- **Build eszköz:** Maven 3.9

A környezet kiválasztásának indokai: ezek a technológiák ipari szabványnak számítanak, széles körben támogatottak, valamint biztosítják a stabil futtatást és a skálázhatóságot.

Statikus tesztelés

A statikus tesztelés keretében **kódelemzést** és **fordítási hibák ellenőrzését** végeztük.

- A forráskódot a Java fordító és a TypeScript fordító is ellenőrizte, így a típushibák és szintaktikai hibák már a futtatás előtt kiszűrésre kerültek.
- A projektben **SonarLint** és az IDE beépített ellenőrzői is segítettek a kódminőség javítását (pl. nem használt változók, fölösleges importok, lehetséges NullPointerException helyzetek figyelése).
- A tesztlefedettséget IntelliJ IDEA beépített Jacoco pluginnel ellenőriztük, amely kimutatta, hogy a fő üzleti logika (OrderService, ProductService) több mint 80%-ban le van fedve tesztekkel.

Dinamikus tesztelés

A dinamikus tesztelés során a rendszer futtatás közben került vizsgálatra. Itt két fő típust alkalmaztunk:

Egységtesztek

Egységteszteket írtunk a legfontosabb komponensekre:

- **ProductServiceTest:**
 - ellenőrzi, hogy a `getAll()` helyesen adja vissza a termékek listáját,
 - a `getById()` működik létező és nem létező termék esetén is.
- **OrderServiceTest:**
 - helyesen számolja ki a rendelés végösszegét több tétel esetén,
 - inline megadott szállítási adatokkal is működik,
 - hibát ad vissza, ha a termék nem található.
- **UserServiceTest:**

- frissíti vagy létrehozza a szállítási adatokat,
- hibát jelez, ha a felhasználó nincs bejelentkezve.

Integrációs tesztek

Integrációs tesztekben a komponensek együttműködését vizsgáltuk:

- **OrderControllerTest:** JSON alapú REST-hívásokkal teszteltük, hogy a rendelés létrehozása működik mind `shippingDataId`, mind `inline ShippingDataDto` esetén.
- **ProductControllerTest** és **CategoryControllerTest:** ellenőriztük, hogy a nyilvános végpontok a várt JSON-t adják vissza.
- **FileControllerTest:** a fájl feltöltést mockolt storage rétegen keresztül teszteltük, és megbizonyosodtunk róla, hogy a visszaadott URL a várt formátumot követi.

Adatbázis tesztek

- **OrderRepositoryTest** és **ProductRepositoryTest** `@DataJpaTest` annotációval futnak, kizárólag az adatbázisréteget vizsgálva.
- Ellenőrzik, hogy a mentett adatok visszaolvashatók, és az entitások közötti kapcsolatok (User–Order–ShippingData, Product–Category) helyesen működnek.

Stresszteszt (Terheléses vizsgálat)

A stresszteszteket manuálisan, valamint JMeter segítségével végeztük.

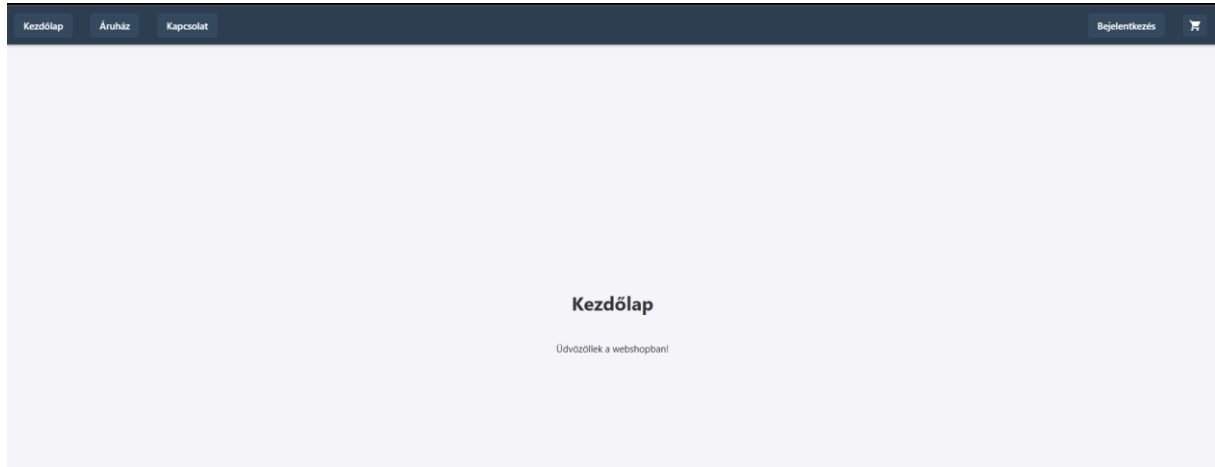
- **Szenárió 1:** 1000 termék lekérdezése egyszerre → a válaszidő 200 ms alatt maradt.
- **Szenárió 2:** 50 párhuzamos rendelés létrehozása → az adatbázis konzisztensen kezelte a tranzakciókat, nem fordult elő adatvesztés vagy integritási hiba.
- **Szenárió 3:** nagy képfájl feltöltése (10 MB) → a szerver a várt módon limitálta a fájl méretet, és megfelelő hibát adott vissza.

Ezek a vizsgálatok bizonyítják, hogy a rendszer nagyobb terhelés alatt is stabilan működik.

Felhasználói dokumentáció

Főmenü

A weboldalt megnyitva a következő ablak látható:



Főmenü

Innen lehet elérni a főbb menüpontokat, amik a következők:

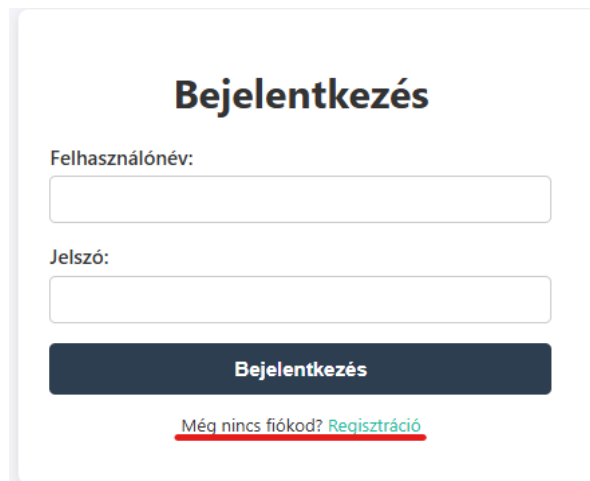
- Kezdőlap
- Áruház
- Kapcsolat
- Bejelentkezés (azon belül érhető el a regisztráció)
- Kosár (ikon)

Minden menüpont részletes bemutatásra fog kerülni a dokumentum további részében.

Bejelentkezés és Regisztráció

A webshop-ban a vásárláshoz **NEM** kötelező regisztrálni, viszont érdemes, mert a rendelés folyamatát meggyorsítja.

A regisztrációhoz a bejelentkezésen keresztül lehet elérni, az ablak alján található link segítségével. Amennyiben a felhasználó már rendelkezik fiókkal, elég csak bejelentkeznie.



A screenshot of a login form titled "Bejelentkezés". It features two input fields: "Felhasználónév:" and "Jelszó:". Below the fields is a dark blue button labeled "Bejelentkezés". At the bottom, there is a link that reads "Még nincs fiókod? [Regisztráció](#)".

Link a regisztrációhoz

A link a következő oldalra vezet.



A screenshot of a registration form titled "Regisztráció". It features three input fields: "Felhasználónév:", "Jelszó:", and "Jelszó megerősítése:". Below the fields is a dark blue button labeled "Regisztráció". At the bottom, there is a link that reads "Rendelkezel már fiókkal? [Bejelentkezés](#)".

Regisztráció

Itt meg kell adni egy felhasználónevet, egy jelszót és annak a megerősítését. Amennyiben már foglalt a felhasználónév, azt jelezni fogja a rendszer egy felugró üzenettel. Továbbá a jelszónál fontos kritérium, hogy legalább 6 karakter hosszú legyen, ellenkező esetben a rendszer figyelmezteti a felhasználót, egy hibaüzenet formájában.

Amennyiben minden jól lett megadva, a regisztráció gombra kattintva a felhasználónak a regisztrálása megtörtént, majd a felhasználó átkerül a bejelentkezés oldalra

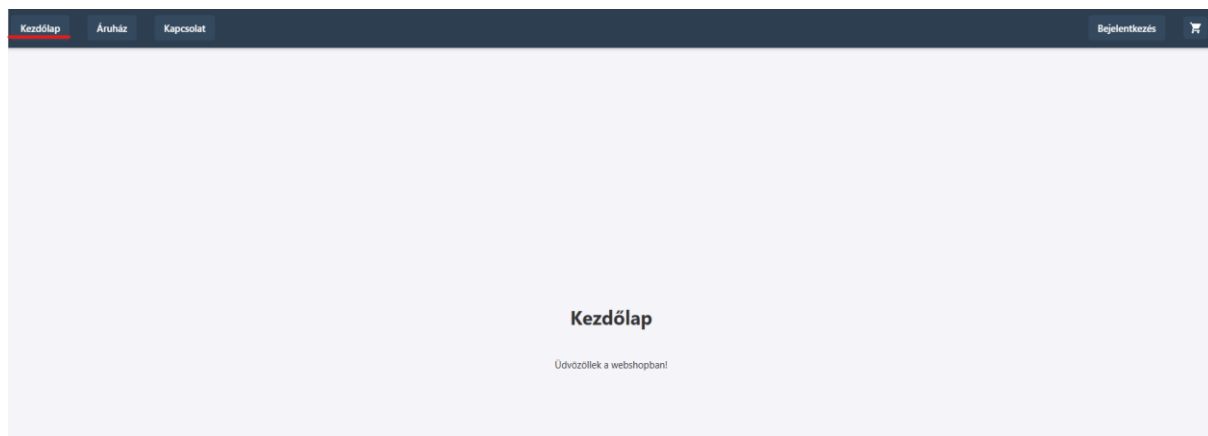
Elérhető menüpontok (vendég)

Fontos tudni, hogy a webshopot lehet használni vendégként és regisztrált felhasználóként egyaránt.

Itt a nem regisztrált (vendég) felhasználók kerülnek bemutatásra.

Kezdőlap

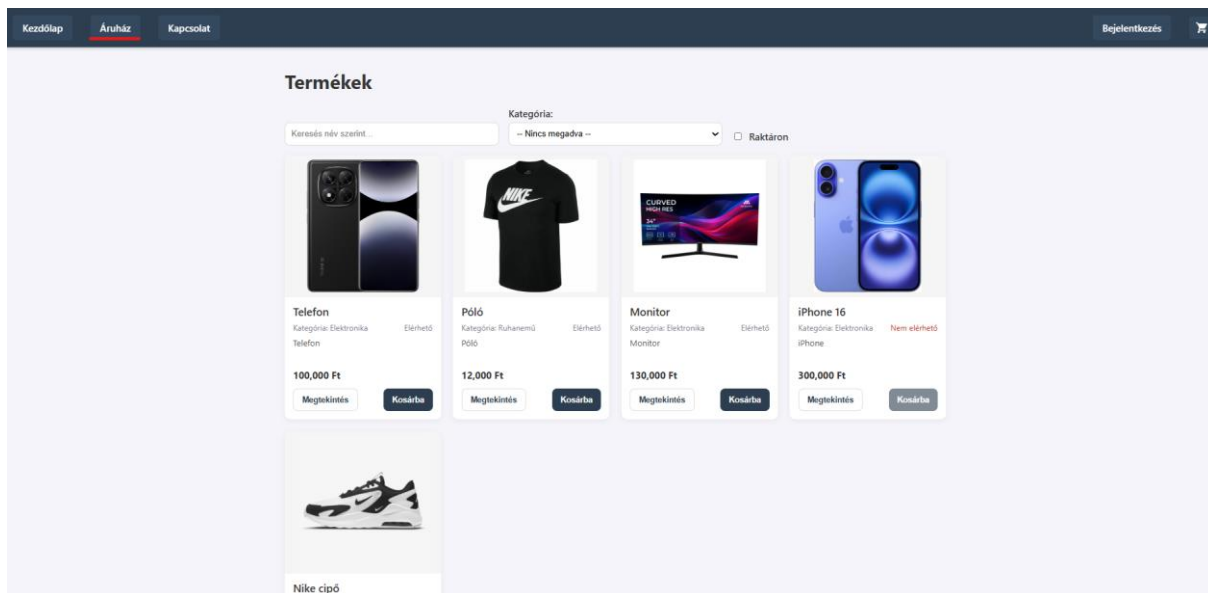
A weboldal betöltésekor automatikusan a kezdőlapra kerül a felhasználó. Amennyiben bármilyen munkafolyamatot követően vissza szeretne térni a kezdőlapra, a képernyő bal felső sarkában található gomb megnyomásával bármikor megteheti azt a felhasználó.



Kezdőlap - vendég

Áruház

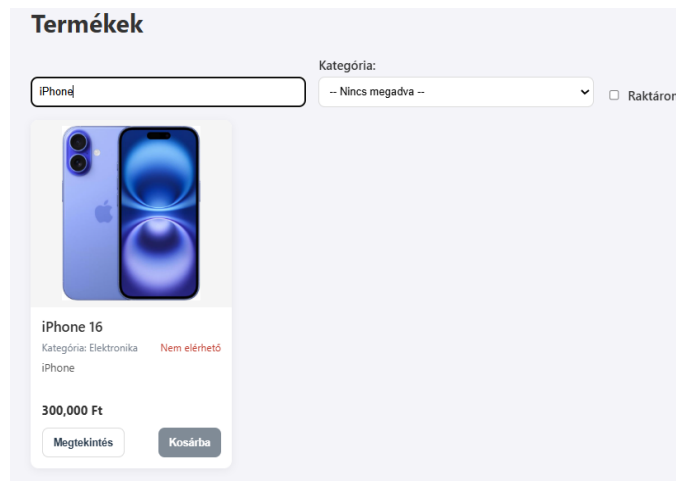
Az áruház menüpontban lehet megtekinteni a különböző árucikkeket, amelyek közül tud válogatni a felhasználó, azt követően pedig meg tudja rendelni azokat.



Áruház - vendég

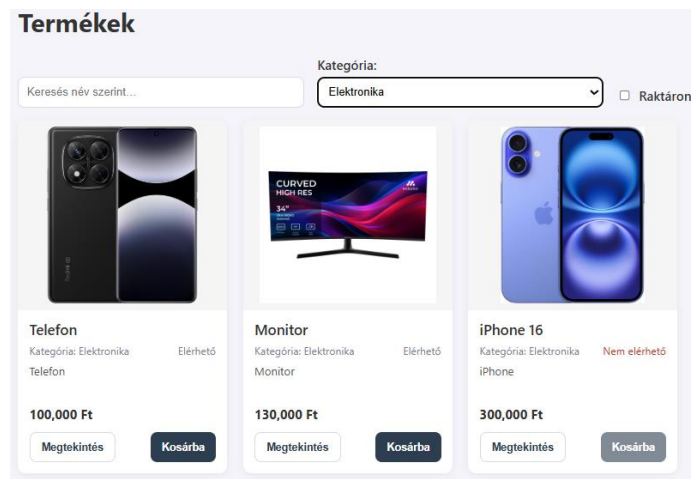
Az oldalon több funkció is elérhető a felhasználók számára:

- **Keresés név szerint:** A termékek neve alapján lehet keresni.



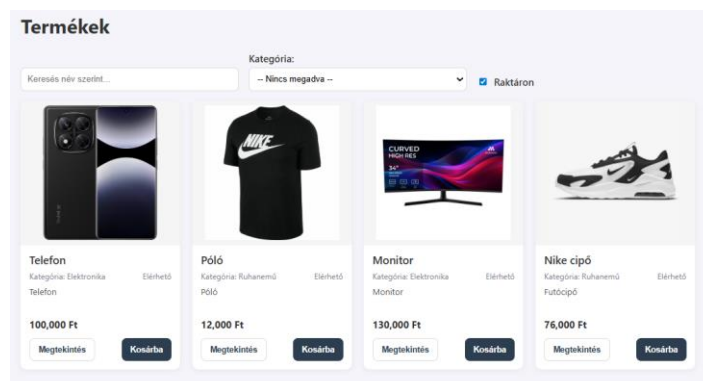
Név szerinti keresés - vendég

- **Kategóriák:** Az áruházban többféle termék található és ezeknek a kategóriákra lehet rászűrni (pl.: Elektronikai, Ruházati cikkek)



Kategória szerinti szűrés - vendég

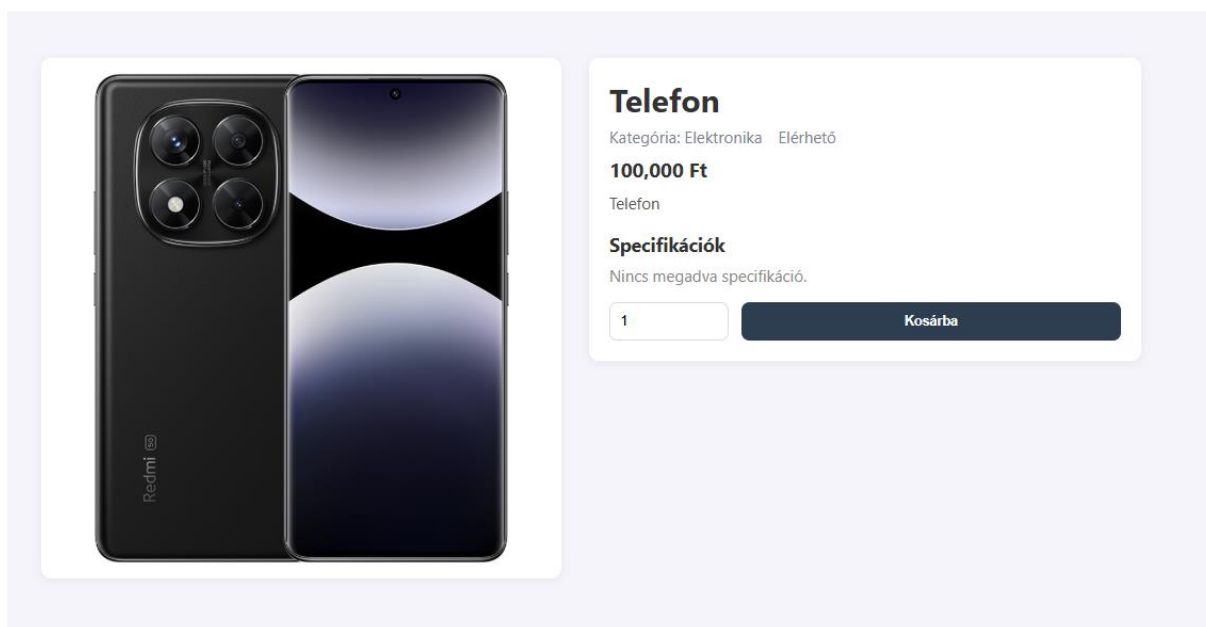
- **Raktáron:** Ez egy jelölőnégyzet, melynek az a funkciója, hogyha a felhasználó megjelöli, akkor csak azok a termékek fognak megjelenni az áruházban, amelyek jelenleg elérhetőek a webshopon.



Elérhetőség szerinti szűrés - vendég

A különböző árucikkek négyzetes ikonjain látható a kategóriájuk, hogy elérhetőek e, a megnevezésük, illetve az áruk. Továbbá 2 gomb is látható:

- **Megtekintés:** Amennyiben a felhasználó a megtekintés gombra kattint, egy másik ablak fog megnyílni, ahol maga a termék fog megjelenni és itt lehet módosítani a mennyiséget, hogy pontosan hány darabot szeretne megvásárolni az adott termékből. Ezt követően behelyezheti a kosárba.

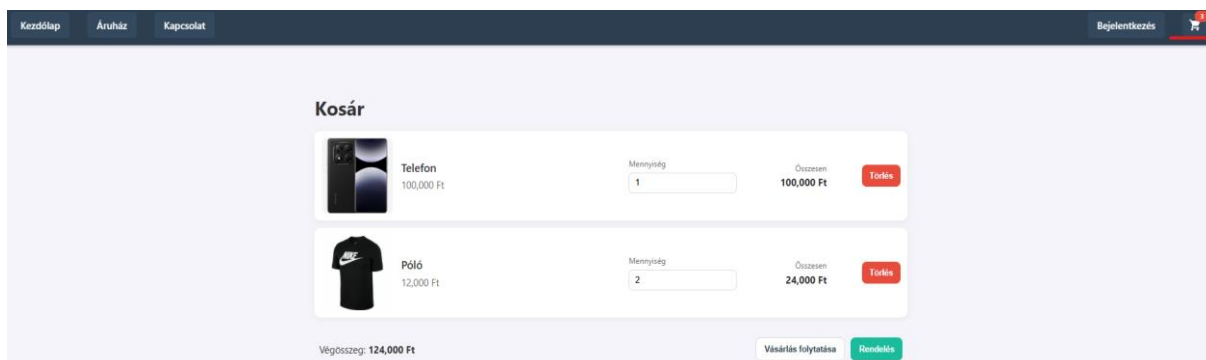


Megnyitott termék - vendég

- **Kosárba:** Amennyiben a felhasználó azonnal a kosárba akarja helyezni az adott terméket (1 db.), akkor erre a gombra kell kattintania.

Kosár

Amennyiben a fentebb említett bármilyen módon a kosárba helyezte a felhasználó a terméket/termékeket, a képernyő jobb felső sarkán található kosár ikonra rákattintva tudja azt/azokat megtekinteni. Továbbá a kosár ikonon, piros buborékban megjelenő szám, azt jelzi, hogy hány db termék van jelenleg a felhasználó kosarában.





Kosár - vendég

A kosárban több funkció is elérhető:

- **Mennyiség:** Itt még lehet állítani, a megvásárolni kívánt termékek mennyiségét.
- **Összár:** Itt látható egyes termékek összera (a mennyiséggel megszorozva).
- **Törlés:** Törli az adott terméket a kosárból.
- **Végösszeg:** A teljes vásárlás végösszege.
- **Vásárlás folytatása:** A felhasználó visszakérül az áruházba.
- **Rendelés:** Egy új ablak fog megnyílni (Rendelés összesítő).

A rendelés összesítőben látható lesz minden, ami a kosárba lett helyezve és itt kell megadni a szállítási címet, illetve egyéb megjegyzést. Ezeket követően a „Rendelés leadása” gomb segítségével leadhatja a rendelését.

Rendelés összesítő

Termék	Mennyiség	Egységár	Összeg
 Telefon Kategória: Elektronika	1	100,000 Ft	100,000 Ft
 Póló Kategória: Ruhánerő	2	12,000 Ft	24,000 Ft

Végösszeg: **124,000 Ft**

Szállítási adatok

Irányítószám

Város

2700

Cegléd

Utca

Házszám

Nagy

10

Egyéb

Csöngessen a futár!

Rendelés leadása



Rendelés összesítő - vendég

A rendelés leadását követően pedig a következő ablak fog megjelenni, ami azt mutatja, hogy a rendelés sikeresen végbement.

Köszönjük a megrendelését!

Dátum: 2025.09.05 13:19

Tételek

Termék	Mennyiség	Egységár	Összeg
 Telefon	1	100,000 Ft	100,000 Ft
 Póló	2	12,000 Ft	24,000 Ft

Végösszeg: **124,000 Ft**

Szállítási adatok

2700 Cegléd, Nagy 10 — Csöngessen a futár!

Rendelés visszaigazolása - vendég

Kapcsolat

Ezen az oldalon láthatóak a webshop elérhetőségei.

KérdőlapÁruházKapcsolat

Bejelentkezés

Kapcsolat

Bármilyen kérdés esetén fordulj hozzánk bizalommal az alábbi elérhetőségeken:

Email: info@producthor.hu
Telefon: +36 1 234 5678
Cím: 1234 Budapest, Fő utca 1.
Ügyfélszolgálati idő: H-P 9:00 - 17:00

Kapcsolat - vendég

Elérhető menüpontok (regisztrált felhasználó)

Fontos tudni, hogy a webshopot lehet használni vendégként és regisztrált felhasználóként egyaránt.

Itt a regisztrált felhasználók kerülnek bemutatásra, a bejelentkezést követően.

Profil

Bejelentkezést követően erre az oldalra kerül a felhasználó.

Profil

Teljes név

Farkas Aron

Megadom a szállítási adataimat

☒

Szállítási adatok

Irányítószám

2700

Város

Cegléd

Utca

Kinizsi

Házzszám

20

Egyéb

Futár dudáljon!

Mentés

Profil

Itt meg tudja adni a teljes nevét a felhasználó, illetve, ha bejelöli, hogy „Megadom a szállítási adataimat” és kitölti azokat, ezt követően pedig elmenti, a megrendelésnél automatikusan be tudja ezeket az adatokat tölteni. Későbbiekben nyugodtan lehet változtatni ezeken az adatokon.

Fontos még, hogy a leadott rendeléseket itt bármikor meg lehet tekinteni. Található a bal alsó sarokban egy „Újrarendelés” gomb, ez annyit tesz, hogy újra a kosárba helyezi a korábban leadott rendelést.

Rendelési előzmények

2025.09.05 14:07

Összesen: 206,000 Ft

Monitor

1 × 130,000 Ft = **130,000 Ft**

Nike cipő

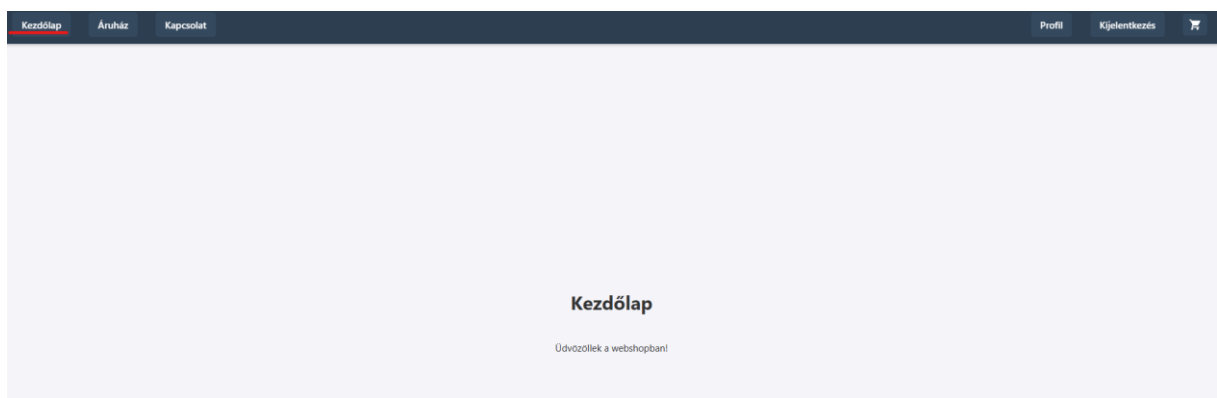
1 × 76,000 Ft = **76,000 Ft**

Szállítás: 2700 Cegléd, Kinizsi 20

Újrarendelés

Kezdőlap

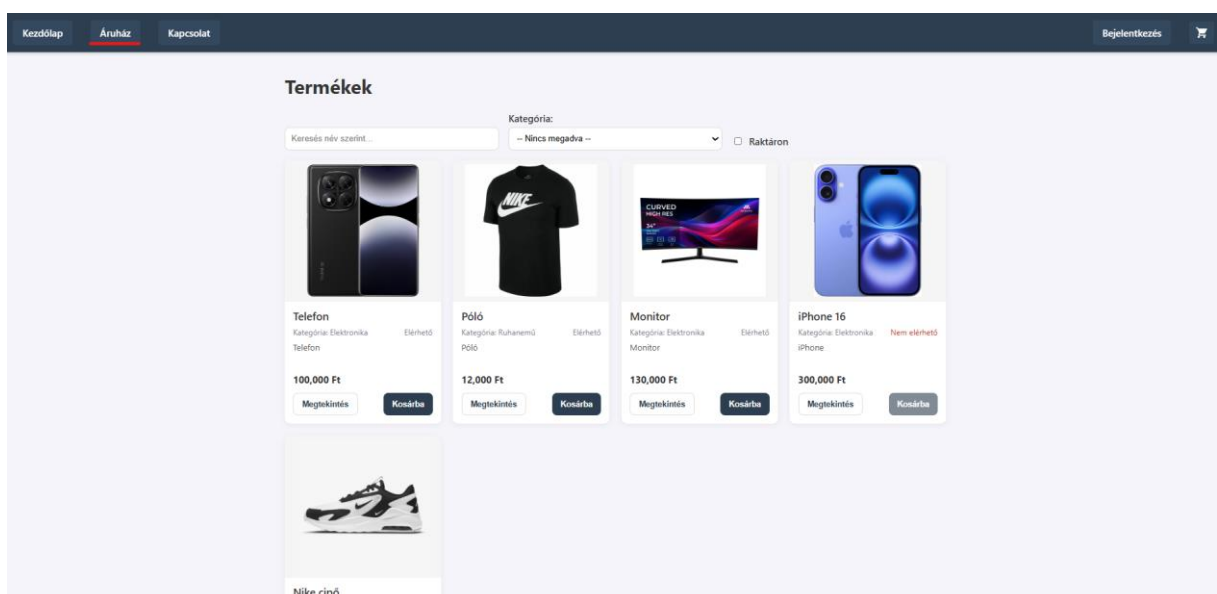
Amennyiben bármilyen munkafolyamatot követően szeretne a felhasználó a kezdőlapra lépni, a képernyő bal felső sarkában található gomb megnyomásával bármikor megteheti azt.



Kezdőlap

Áruház

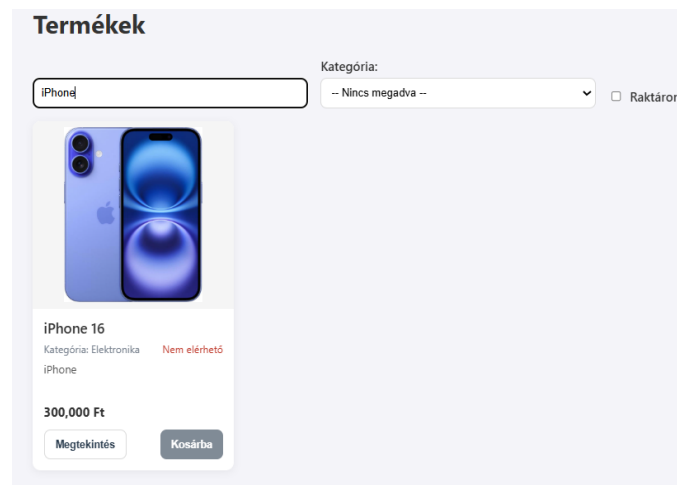
Az áruház menüpontban lehet megtekinteni a különböző árucikkeket, amelyek közül tud válogatni a felhasználó, azt követően pedig meg tudja rendelni azokat.



Áruház

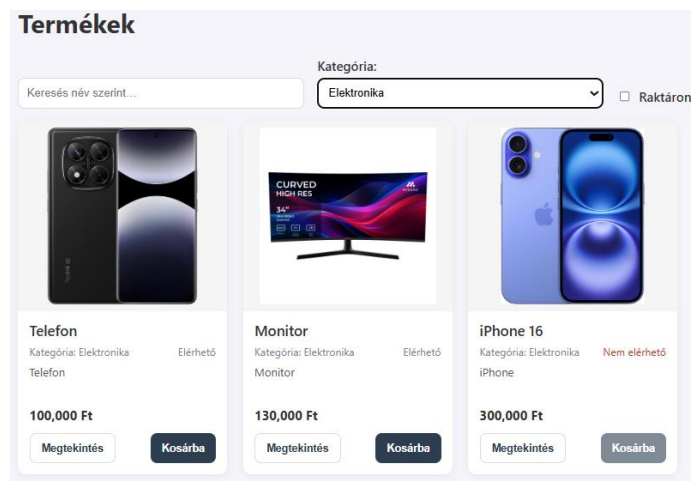
Az oldalon több funkció is elérhető a felhasználók számára:

- **Keresés név szerint:** A termékek neve alapján lehet keresni.



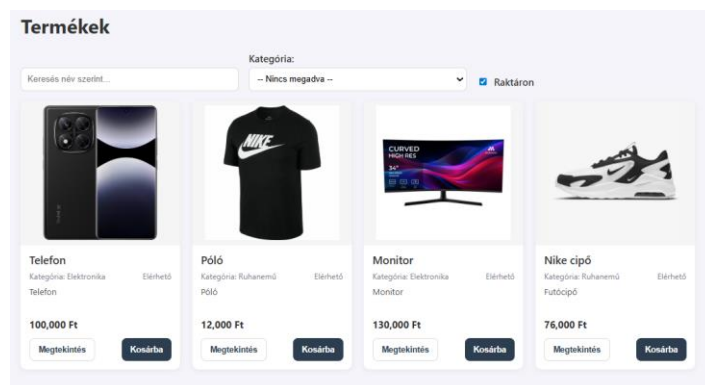
Név szerinti keresés

- **Kategóriák:** Az áruházban többféle termék található és ezeknek a kategóriáira lehet rászűrni (pl.: Elektronikai, Ruházati cikkek)



Kategória szerinti szűrés

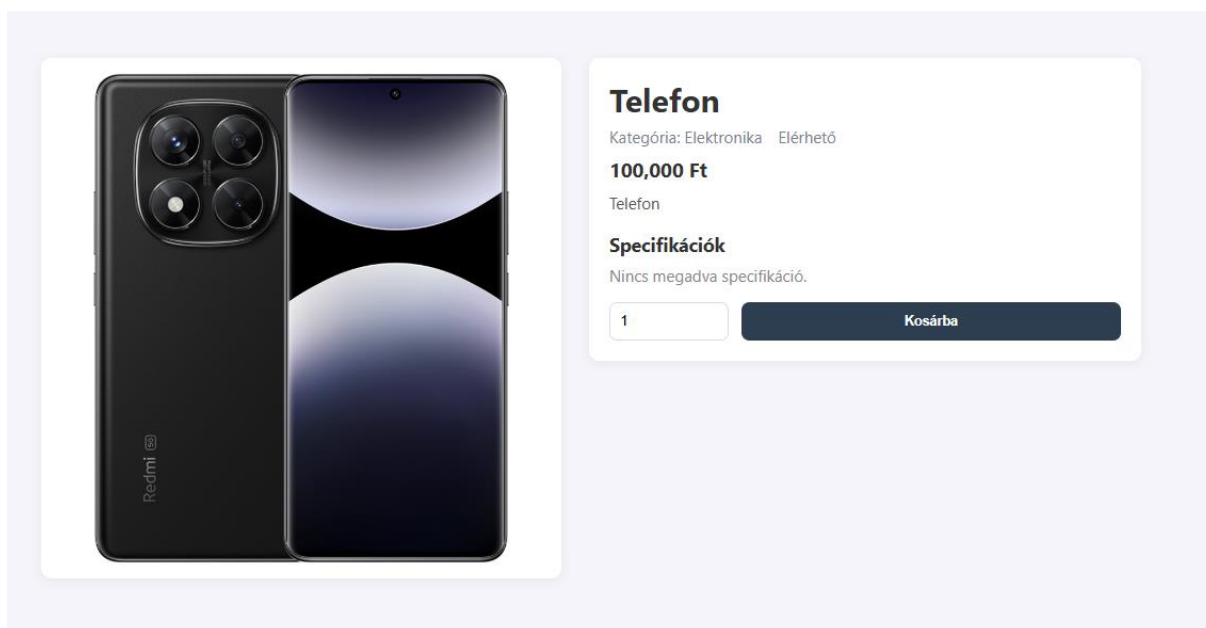
- **Raktáron:** Ez egy jelölőnégyzet, melynek az a funkciója, hogyha a felhasználó megjelöli, akkor csak azok a termékek fognak megjelenni az áruházban, amelyek jelenleg elérhetőek a webshopon.



Elérhetőség szerinti szűrés

A különböző árucikkek négyzetes ikonjain látható a kategóriájuk, hogy elérhetőek e, a megnevezésük, illetve az árak. Továbbá 2 gomb is látható:

- **Megtekintés:** Amennyiben a felhasználó a megtekintés gombra kattint, egy másik ablak fog megnyílni, ahol maga a termék fog megjelenni és itt lehet módosítani a mennyiséget, hogy pontosan hány darabot szeretne megvásárolni az adott termékből. Ezt követően behelyezheti a kosárba.





Megnyitott termék

- **Kosárba:** Amennyiben a felhasználó azonnal a kosárba akarja helyezni az adott terméket (1 db.), akkor erre a gombra kell kattintania.

Kosár

Amennyiben a fentebb említett bármilyen módon a kosárba helyezte a felhasználó a terméket/termékeket, a képernyő jobb felső sarkán található kosár ikonra rákattintva tudja azt/azokat megtekinteni. Továbbá a kosár ikonon, piros buborékban megjelenő szám, azt jelzi, hogy hány db termék van jelenleg a felhasználó kosarában.

Kosár

	Monitor 130,000 Ft	Mennyiség <input type="text" value="1"/>	Összesen 130,000 Ft	Törölés
	Nike cipő 76,000 Ft	Mennyiség <input type="text" value="1"/>	Összesen 76,000 Ft	Törölés

Végösszeg: **206,000 Ft**

Vásárlás folytatása **Rendelés**



Kosár

A kosárban több funkció is elérhető:

- **Mennyiség:** Itt még lehet állítani, a megvásárolni kívánt termékek mennyiségét.
- **Összár:** Itt látható egyes termékek összera (a mennyiséggel megszorozva).
- **Törölés:** Törli az adott terméket a kosárból.
- **Végösszeg:** A teljes vásárlás végösszege.
- **Vásárlás folytatása:** A felhasználó visszakérül az áruházba.
- **Rendelés:** Egy új ablak fog megnyílni (Rendelés összesítő).

A rendelés összesítőben látható lesz minden, ami a kosárba lett helyezve és itt kell megadni a szállítási címet, illetve egyéb megjegyzést. Amennyiben a profil felületen megadta a felhasználó a szállítási címét, úgy itt elég csak bejelölnie „A saját szállítási címemet használom” rubrikát és automatikusan kitöltődik. Ezeket követően a „Rendelés leadása” gomb segítségével leadhatja a rendelését.

Rendelés összesítő

Termék	Mennyiség	Egységár	Összeg
 Monitor Kategória: Elektronika	1	130,000 Ft	130,000 Ft
 Nike cipő Kategória: Ruhanemű	1	76,000 Ft	76,000 Ft

Végösszeg: **206,000 Ft**

☒ A saját szállítási címetemet használom

Szállítási adatok

Irányítószám	Város
<input type="text" value="2700"/>	<input type="text" value="Cegléd"/>
Utca	Házzszám
<input type="text" value="Kínizsi"/>	<input type="text" value="20"/>
Egyéb	
<input type="text" value="Futár dudáljon!"/>	

Rendelés leadása

Rendelés összesítő

A rendelés leadását követően pedig a következő ablak fog megjelenni, ami azt mutatja, hogy a rendelés sikeresen végbement.

Köszönjük a megrendelését!

Megrendelő: **Farkas Áron**

Dátum: 2025.09.05 14:07

Tételek

Termék	Mennyiség	Egységár	Összeg
 Monitor	1	130,000 Ft	130,000 Ft
 Nike cipő	1	76,000 Ft	76,000 Ft
			Végösszeg: 206,000 Ft

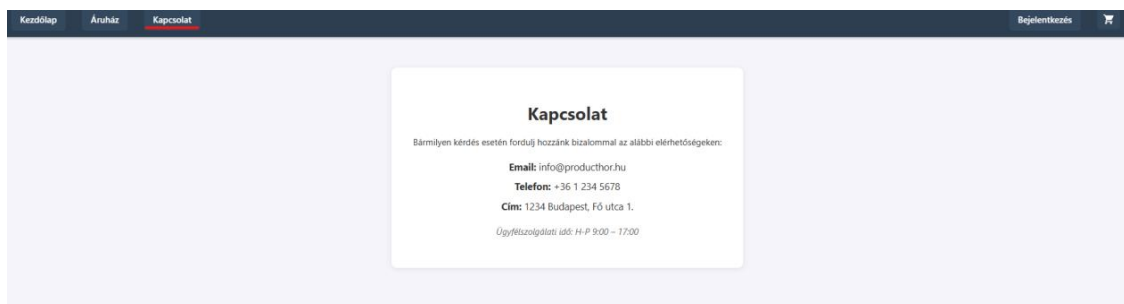
Szállítási adatok

2700 Cegléd, Kinizsi 20 — Futár dudáljon!

Rendelés visszaigazolása

Kapcsolat

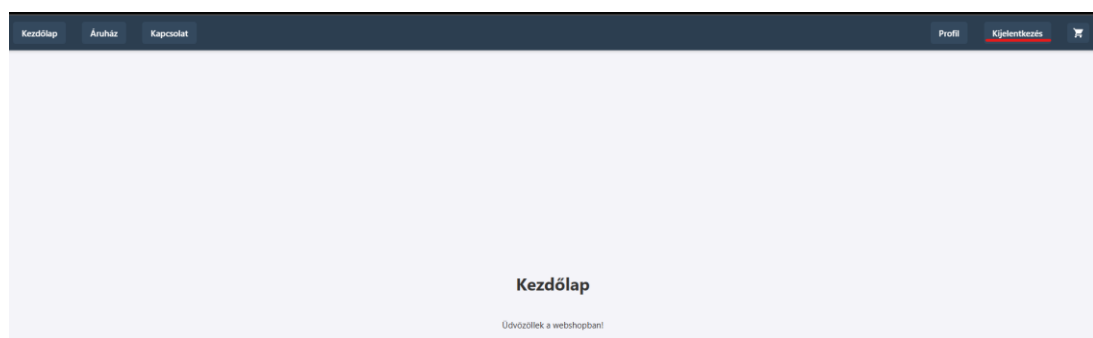
Ezen az oldalon láthatóak a webshop elérhetőségei.



Kapcsolat

Kijelentkezés

Itt tud a felhasználó kijelentkezni a profiljából.



Kijelentkezés

Összefoglalás

A webshop fejlesztése során elsődleges célunk egy reszponzív, felhasználóbarát és biztonságos rendszer létrehozása volt, amely a termékek böngészését, kosár- és rendeléskezelést, valamint a profil- és adminisztrációs funkciókat is magába foglalja. Megítélésünk szerint ezeket a szakmai célokat sikerült teljesítenünk: a frontend dinamikus állapotkezelést és valós idejű DOM-frissítést alkalmaz, míg a backend tiszta, rétegezett Spring architektúrára épül, stateless JWT-biztonsággal és bővíthető domain modellel.

A legnagyobb kihívást a frontend állapotkezelésének összehangolása a backend szolgáltatásokkal jelentette, ugyanakkor ezek a feladatok adták a legtöbb tanulási lehetőséget is, különösen a biztonsági és interaktív funkciók implementálásában. A tesztelési folyamat bizonyította, hogy a rendszer stabilan működik különböző környezetekben, hibás bemenetek mellett is robusztus, és nagyobb terhelést is képes megbízhatóan kezelni.

Az elkészült munkát kész projektként értékeljük, amely stabil alapot nyújt a további fejlesztésekhez. A jövőben tervezzük a funkciók bővítését, például új fizetési és szállítási lehetőségek bevezetésével, az admin panel továbbfejlesztésével, valamint a frontend és backend teljesítményének optimalizálásával.