

**VÁCI SZAKKÉPZÉSI CENTRUM
BORONKAY GYÖRGY
MŰSZAKI TECHNIKUM ÉS GIMNÁZIUM**

VIZSGAREMEK

Barta Balázs

Genzelmann Erik

Kollár András

2026.

**VÁCI SZAKKÉPZÉSI CENTRUM
BORONKAY GYÖRGY
MŰSZAKI TECHNIKUM ÉS GIMNÁZIUM**



VIZSGAREMEK

ShopScout

Konzulens: Kemenes Tamás

Készítette: Barta Balázs

Wiezl Csaba

Genzelmann Erik

Kollár András

Hallgatói nyilatkozat

Alulírottak, ezúton kijelentjük, hogy a vizsgaremek saját, önálló munkánk, és korábban még sehol nem került publikálásra.

Barta Balázs

Genzelmann Erik

Kollár András

Konzultációs lap

Vizsgázók neve: Barta Balázs, Genzelmann Erik, Kollár András

Vizsgaremek címe: ShopScout

Program nyújtotta szolgáltatások:

- Aktuális termékárak megtekintése.
- Boltok központi adatainak és elérhetőségeinek kezelése.
- Üzlethelyiségek digitális elrendezésének kialakítása.
- Termékinformációk teljes körű megtekintése és módosítása.
- Vonalkódolvasás alapú keresés.
- Termékek megjelenítése a digitális üzletben.

Sorszám	A konzultáció időpontja	A konzulens aláírása
1.	2025.10.16.	
2.	2025.11.20.	
3.	2025.12.11.	
4.	2026.01.15	
5.	2026.02.19.	
6.	2026.02.26.	

A vizsgaremek beadható:

Vác, 2026.....

A vizsgaremeket átvettetem:

Vác, 2026.....

.....

.....

Konzulens

A szakképzést folytató

intézmény felelőse

Tartalomjegyzék

Hallgatói nyilatkozat	3
Konzultációs lap	4
1. Bevezetés	8
1.1. Projekt leírása.....	8
1.2. Követelmények	9
1.2.1. A vizsgarész megnevezése	9
1.2.2. A vizsgarész ismertetése.....	10
1.2.3. A szoftveralkalmazás elvárásai	10
1.2.4. A megosztott anyagnak tartalmaznia kell	10
1.2.5. A vizsgarész bemutatása	11
1.2.6. A vizsgaremek elkészítésére rendelkezésre álló idő	11
1.3. Célok	11
1.4. Tervezés	12
1.4.1. Alapkoncepció.....	12
1.4.2. A tervezés fázisai	13
1.5. Munkamegosztás	14
1.5.1. Barta Balázs.....	14
1.5.2. Genzelmann Erik.....	15
1.5.3. Kollár András.....	16
2. Fejlesztési dokumentáció	17
2.1. Technológiák.....	17
2.1.1. Front-end	17
2.1.2. Back-end.....	17
2.1.3. Adatbázis-kezelés	18
2.1.4. Kollaboráció	18
2.1.5. Verziókezelés	18
2.1.6. Külső API-k	19
2.1.7. AI.....	19
2.2. Adatbázis	19
2.2.1. Bevezetés	19
2.2.2. Az adatbázis struktúrája és normalizáltsága	20
2.2.3. Technológiai implementáció: Entity Framework Core.....	20
2.2.4. Teljes adatbázis diagram.....	22
2.2.5. Táblák	23
2.2.5.1. Stores	23
2.2.5.2. StoreBrands.....	24
2.2.5.3. Products	25
2.2.5.4. ProductDetails	26
2.2.5.5. ProductCategory.....	28
2.2.5.6. ProductImages	29

2.2.5.7. StoreAttribute.....	30
2.2.5.8. ProductPackagings.....	31
2.2.5.9. ProductPerStore	32
2.2.5.10. LayoutObjects.....	33
2.2.5.11. Walls	34
2.2.5.12. Shelves.....	35
2.2.5.13. Id-Name táblák	36
2.2.5.14. Kapcsolótáblák	37
2.2.5.15. Beépített Identity Framework táblák	39
2.2.5.16. Egyéb	42
2.3. Jogosultságok, jogosultsági szintek.....	44
2.3.1. Nem regisztrált felhasználó	44
2.3.2. Regisztrált felhasználó	44
2.3.3. Admin	45
2.4. Weboldal fejlesztése	45
2.4.1. Technológiai háttér	45
2.4.2. Rendering és kommunikáció a szerverrel	46
2.4.2.1. Az Interactive Auto működése és előnyei	46
2.4.2.2. Szolgáltatások regisztrációja és a Program.cs	47
2.4.2.3. Implementáció a két oldalon	48
2.4.3. Api dokumentáció	50
2.4.3.1. StoreLayoutController.....	50
2.4.3.2. ProductController.....	53
2.4.4. Függőségek, alkalmazás beállítások	56
2.4.4.1. Naplázás (Serilog)	56
2.4.4.2. Üzleti logika és Szolgáltatások (Scoped Services)	57
2.4.4.3. Rendszerszintű biztonsági és kommunikációs beállítások	59
2.5. A projekt tesztelése	61
2.5.1. Bevezetés és tesztelési stratégia	61
2.5.1.1. A készre jelentés feltételei (Definition of Done)	61
2.5.1.2. Alkalmazott tesztelési módszertanok	61
2.5.1.3. Egységtesztek (Unit Tests)	62
3. Felhasználói dokumentáció	67
3.1. Telepítési útmutató	67
3.1.1. Webszerver	67
3.1.2. Rendszerkövetelmények	67
3.1.2.1. Szoftveres feltételek	67
3.2. Használati eset diagram	68
3.3. Weboldal felhasználói dokumentáció	69
3.3.1. Rendszerkövetelmények	69
3.3.1.1. Szoftveres feltételek	69
3.3.1.2. Hardveres feltételek	69

3.3.1.3. Hálózati feltételek	69
3.4. Felhasználói felület (UI) bemutatása	70
3.4.1. Üdvözlő oldal (Landing page)	70
3.4.2. Főoldal	71
3.4.2.1. Keresés	72
3.4.2.2. Szűrők	73
3.4.3. Alsó menü elemei.....	74
3.4.3.1. Vonalkód olvasó	74
3.4.3.2. Beállítások (bejelentkezett felhasználók).....	74
3.4.4. Bejelentkezés.....	78
3.4.5. Regisztráció	79
3.4.6. Kategóriák.....	79
3.4.7. Termék részletes adatlapja	81
3.4.8. Bejelentkezett felhasználók – kedvencek.....	83
3.4.8.1. Kereső főoldal.....	83
3.4.8.2. Termékoldal	84
3.4.9. Termékek szerkesztője – bejelentkezett felhasználóknak	84
3.4.9.1. Rögzített akcióság („Gomb sziget”)	85
3.4.9.2. Adatbeviteli és módosítási funkciók	85
3.4.10. Bolt adatlapja	87
3.4.10.1. Fejléc és valós idejű állapotjelzők.....	87
3.4.10.2. Geográfiai adatok és navigáció (Helyszín modul).....	87
3.4.10.3. Üzemeltetési információk (Nyitvatartás modul).....	88
3.4.10.4. Helyi termékkínálat (Termékek ebben a boltban).....	88
3.4.10.5. Árazás és akciók vizualizációja	88
3.4.10.6. Beltéri navigáció és termékhelymeghatározás	89
3.4.11. Üzlet alaprajzának szerkesztője – bejelentkezett felhasználóknak	90
3.4.11.1. Kitöltött nézet, megszerkesztett bolt	90
3.4.11.2. Polc adatainak panelje és termék felrakás	91
3.4.12. Admin kezelőfelület	92
4. Továbbfejlesztési lehetőségek.....	93
5. Mellékletek	95
5.1. Publikus Github repository.....	95
5.2. Online elérhetőség a ShopScout-hoz (weblap URL)	95
6. Felhasznált szakirodalmak, források jegyzéke	96
6.1. Könyvek, tankönyvek.....	96
6.2. Tutoriálok, online források.....	98

1. Bevezetés

1.1. Projekt leírása

A **ShopScout** egy komplex szoftvermegoldás, amely a fizikai vásárlási folyamat digitalizálását és hatékonyságának növelését célozza meg. A projekt elsődleges célja, hogy releváns információkkal lássa el a felhasználót az áruházi környezetben, elősegítve az időhatékony tájékozódást és a tudatos fogyasztói döntéshozatalt. A rendszer egy integrált navigációs és ár-összehasonlító platformként funkcionál, amely a közösségi adatgyűjtés (crowdsourcing) révén teremt közvetlen kapcsolatot a tárolt termékkatalogusok és a fizikai eladótér között. Mivel a szoftver – technológiai –, teljesen független a kereskedelmi egységek saját informatikai infrastruktúrájától, az adatbázis tartalmát – beleértve a pontos polckiosztást, a térképadatokat és a termékárakat –, kizárolag a felhasználók rögzítik és tartják naprakészen. Ez a megközelítés lehetővé teszi a precíz, polcszintű helymeghatározást és a különböző üzletek valós kínálatának objektív összehasonlítását, miközben a vásárlói közösség tapasztalataira építve optimalizálja a teljes beszerzési folyamatot.

Munkánk során nagyban támaszkadtunk a legfrissebb marketing (elsősorban vásárlás-módszertani) kutatásokra. A saját bőrünkön tapasztaljuk, hogy a modern eladás már nem reklám, hanem segítségnyújtás. Számunkra fontos, hogy miként válnak a vásárlók „jelzőfényekké” más vásárlók számára¹, a C2C és ennek közösségi hatása. A hagyományos márkanevük ereje csökken, mert a vásárlók már pillanatok alatt ellenőrizni tudják a „valós értéket” (vélemények, specifikációk alapján). A kereskedelemben – egy online webáruház létrehozásánál – még akkor is ha erősen általánosított, mint a minden-, kulcsfontosságú annak megértése, hogy a vevők miként navigálnak az információtengerben². Az előzőek miatt foglalkoztunk az online kereskedelelem történetével³, ezen belül elsősorban az Amazonnal, mely a „vásárlótól vásárlónak” típusú információáramlást vezette, megvédte a konzervatív kereskedők támadásaitól, és elterjesztette. Vizsgálódtunk a technikai és

¹ Seth Godin: *This is Marketing* magyar megjelenésben: *Na, ez már marketing!*

² Itamar Simonson & Emanuel Rosen: *Absolute Value*

³ Jeff Bezos & Brad Stone: *The Everything Store*

pszichológiai oldal területén is, azt kerestük, miként lehet optimalizálni a „vásárlói utat”, onnantól, hogy a vevő keres valamit, egészen addig, míg a futár át nem adja neki a csomagot. Igyekeztünk a generációs különbségeket is figyelembe venni: a Baby Boomers és az X generáció a szöveges, részletes véleményekben bízik (pl. Árukereső, Amazon vélemények), a Millennials (Y generáció) az influenszerek és a blogok véleményében, a Z és Alfa generáció tagjai a vizuális és azonnali jelek (TikTok, Instagram Reels, QR-kód a polcon) alapján navigálnak⁴. Ezen generációs attitűdök metszetében a **ShopScout** nem csupán egy technológiai válasz, hanem egy digitális híd a fizikai valóság és az információs igény között. Felismertük, hogy miközben az idősebb generációk a hitelesített adatok stabilitását és a szöveges pontosságot keresik, a fiatalabbak számára a vizualitás és a gyorsaság az alapértelmezett elvárás. Szoftverünk ezen feszültséget oldja fel azáltal, hogy a közösségi tudást strukturált, mégis azonnal hozzáférhető formába önti.

A „vásárlói út” nálunk tehát nem csupán egy absztrakt marketingmodell, hanem egy jól definiált algoritmus, amely a digitális jelekből valós, fizikai iránytűt formál. Mivel a rendszer önszerveződő és független az üzletláncok zárt rendszereitől, az információ hitelességét maga a közösség garantálja. Ebben a megközelítésben a kód sorai a fizikai polcok sorai között nyernek értelmet: célunk, hogy a felhasználó ne csupán passzív szemlélője, hanem aktív alakítója legyen annak az adatvagyonnak, amely végül a legrövidebb és leglogikusabb utat mutatja meg a modern kereskedelem zsúfolt labirintusában.

1.2. Követelmények

1.2.1. A vizsgarész megnevezése

Szoftverfejlesztés és -tesztelés vizsgaremek vizsgarész

⁴ Steigerval Krisztián: *Generációk harca a figyelemért*

1.2.2. A vizsgarész ismertetése

A vizsgázóknak minimum 2, maximum 3 fős fejlesztői csapatot alkotva kell a vizsgát megelőzően egy komplex szoftveralkalmazást lefejleszteniük.

1.2.3. A szoftveralkalmazás elvárásai

- Életszerű, valódi problémára nyújt megoldást.
- Adattárolási és -kezelési funkciókat is megvalósít.
- RESTful architektúrának megfelelő szerver és kliens oldali komponenseket egyaránt tartalmaz.
- A kliens oldali komponens vagy komponensek egyaránt alkalmasak asztali és mobil eszközökön történő használatra. Mobil eszközre kifejlesztett kliens esetén natív mobil alkalmazás, vagy azzal hozzávetőlegesen megegyező felhasználói élményt nyújtó webes kliens egyaránt alkalmazható. Asztali eszközökre fejlesztett kliens oldali komponensnél mindenkorban szükséges webes megvalósítás is, de emellett opcionálisan natív, asztali alkalmazás is a csomag része lehet. (pl. A felhasználóknak szánt interfész webes megjelenítést használ, míg az adminisztrációs felület natív asztali alkalmazásként készül el).
- A forráskódnak a tiszta kód elveinek megfelelően kell készülnie.
- A szoftver célját, komponenseinek technikai leírását, működésének műszaki feltételeit és használatának rövid bemutatását tartalmazó dokumentáció is része a csomagnak.

1.2.4. A megosztott anyagnak tartalmaznia kell

- A szoftver forráskódja.
- Natív asztali alkalmazások esetén a program telepítőkészlete.
- Az adatbázis adatbázismodell-diagramja.
- Az adatbázis export fájlja (dump).
- A szoftveralkalmazás dokumentációja.
- A tesztekhez végzett kód, valamint a teszteredmények dokumentációja.

1.2.5. A vizsgarész bemutatása

A vizsgarész során a vizsgázó gyakorlati bemutatóval összekapcsolt szóbeli előadás formájában mutatja be a:

- szoftver célját,
- műszaki megvalósítását,
- működését,
- forráskódját,
- a csapaton belüli munkamegosztást, a fejlesztési csapatban betöltött szerepét, a fej- lesztés során használt projektszervezési eszközöket.

A fentieken túl maximum 3-5 perces angol nyelven tartott szóbeli előadás formájában összefoglalót ad a szoftver céljáról és működéséről, valamint angolul válaszol a vizsgáztató végfelhasználói szerepben feltett maximum 2-3 kérdésére.

Amennyiben a munkacsapat más tagjai is azonos csoportban vizsgáznak, akkor a bemutatót közösen is megtarthatják, de ebben az esetben is biztosítani kell, hogy minden vizsgázó egyenlő arányban vegyen részt a bemutatóban, illetve minden vizsgázónak önállóan kell bemutatnia a saját feladatrészét magyarul és angolul egyaránt.

1.2.6. A vizsgaremek elkészítésére rendelkezésre álló idő

A vizsgaremet a záróvizsga tanévében kell a vizsgázónak elkészítenie.

1.3. Célok

A projekt célja a ShopScout szoftverrendszer prototípusának kifejlesztése. A fejlesztési folyamat során egy funkcionálisan megalapozott, az alapvető működési mechanizmusokat bemutató megoldás készül, amely alkalmas a rendszer főbb komponenseinek és logikai felépítésének demonstrálására.

A megvalósítás nem törekszik a teljes körű, minden lehetséges használati esetre kiterjedő implementációra, hanem a legfontosabb funkciók – különös tekintettel a beltéri navigációra és a közösségi alapú adatkezelésre – példaszerű kidolgozására helyezi a hangsúlyt. A cél a rendszer komponensei közötti

együttműködés, az adatfolyamok és a felhasználói interakciók szemléltetése valósághű, de korlátozott funkcionális környezetben. A prototípus elsősorban technológiai és architekturális demonstráció, nem végleges, kereskedelmi felhasználásra szánt termék.

A projekt másodlagos, nem funkcionális céljai között szerepel a modern szoftverfejlesztési eszközök és iparági sztenderdek elmélyült megismerése, valamint a rendszerszintű mérnöki gondolkodásmód fejlesztése. A munka során kiemelt szempont volt az alábbi technológiák és szakterületek gyakorlati alkalmazása:

- Relációs adatbázis-kezelés a termékkatárok, árinformációk és térképi koordináták strukturált tárolására és hatékony lekérdezésére.
- REST-alapú API tervezése és implementálása a kliensoldali alkalmazások és a központi szerver közötti konzisztens kommunikáció biztosítására.
- Mobilfejlesztés gyakorlati megvalósítása, fókuszálva az intuitív felhasználói felületre és a helyalapú információk megjelenítésére.
- Webes biztonsági szempontok érvényesítése, különös tekintettel a felhasználói hitelesítésre, a jogosultságkezelésre és az adatok védelmére.
- Moduláris szoftverarchitektúra kialakítása, törekedve a kód újrafelhasználhatóságára és a tiszta, jól dokumentált struktúrára.
- Professzionális fejlesztői eszközök (például Visual Studio, Postman, Git verziókezelő) készségszintű és hatékony használata a fejlesztési ciklus során.

1.4. Tervezés

1.4.1. Alapkoncepció

- **A Probléma:** A nagy alapterületű áruházakban a vásárlók sok időt veszítenek a termékek keresésével. Az üzletláncok saját applikációi gyakran hiányosak, nem adnak pontos polchelyet, és nem teszik lehetővé az árak valós idejű összehasonlítását a konkurenciával.

- **A Megoldás (A ShopScout lényege):** Egy üzletfüggetlen, crowdsourcing (közösségi adatgyűjtés) alapú platform. A rendszer a felhasználók által beküldött adatokra építve digitalizálja az üzletek belső terét.
- **Technológiai Függetlenség:** Ez a koncepció legfontosabb eleme. Nem várunk az üzletláncok API-jaira vagy jóindulatára; a közösség erejét használjuk az adatbázis frissítésére (árak, akciók, polckiosztás).
- **Értékajánlat (Value Proposition):**
 - **A vásárlónak:** Stresszmentes navigáció és garantált megtakarítás.
 - **A közösségnek:** Egy platform, ahol az információ megosztása mindenki számára előnyös (adatvezérelt tudatosság).

1.4.2. A tervezés fázisai

Kutatási fázis és igényfelmérés (Research):

- **Piackutatás:** Létező megoldások elemzése (pl. Google Maps beltéri nézete, üzletláncok saját appjai). Hol buknak el ezek? (Válasz: adatok frissessége, polcszintű pontosság hiánya).
- **Felhasználói interjúk:** Annak igazolása, hogy van valós igény a beltéri navigációra és a közösségi árfigyelésre.

Funkcionális tervezés:

- **User Stories meghatározása:** "Vásárlóként szeretném látni a leggyorsabb útvonalat a tej és a kenyér között, hogy **elkerüljem a felesleges köröket az üzletben.**"
- **MVP (Minimum Viable Product) kijelölése:** Mi az a minimum szint, amivel az app már működik? (Pl. alaprajz megjelenítése + árlista).
- **Adatvalidációs mechanizmus tervezése:** Mivel közösségi adatokról van szó, meg kell tervezni, hogyan szűrjük ki a téves árakat vagy helyszíneket (pl. pontrendszer a hiteles feltöltőknek).

Rendszerarchitektúra és Adatmodell tervezése:

- **Adatbázis-séma:** Hogyan tároljuk a téradatokat? (Üzletlánc → Üzlet → Részleg → Polcsor → Polchely → Termék).

- **Backend logika:** Az árak összehasonlításának algoritmusa és a térképi koordináták kezelése.
- **API tervezés:** A frontend és a backend közötti kommunikációs végpontok (REST vagy GraphQL).

Tesztelési terv összeállítása:

- **Unit tesztek:** Az alapvető számítások (pl. ár-összehasonlítás) ellenőrzése.
- **Manuális "terepi" teszt:** Egy valós áruházban ellenőrizni, hogy a tervezett térkép és a valóság összhangban van-e.

1.5. Munkamegosztás

1.5.1. Barta Balázs

- **Felhasználókezelés (ASP.NET Core Identity):** Biztonságos hitelesítési rendszer és **Role-based Access Control (RBAC)** implementálása; a vendég szerepkörhöz tartozó személyre szabott funkciók (kedvenc boltok és termékek mentése/kezelése) kidolgozása, valamint az **Adminisztrátori** jogosultság elkülönítése az admin panel és a háttéradatok eléréséhez.
- **Vonalkód-beolvasás (ZXing):** Kliensoldali kamera-hozzáférés és valós idejű képfeldolgozás a **ZXing** könyvtár integrálásával; a különböző kódformátumok (pl. EAN-13, QR) dekódolása és automatikus termékpárosítás a ShopScout adatbázisával a manuális keresés kiváltása érdekében.
- **Adatmodellezés (EF Core ORM):** Relációs adatbázis-struktúra kialakítása **Code-First** megközelítéssel; a **Shop**, **Product** és a kapcsolótáblák entitásainak és tulajdonságainak definiálása, az objektum-relációs leképezés (Mapping), valamint az automatizált sémakezelés és verziókövetés biztosítása **Migrations** használatával.
- **Termékadatlap (Product Page):** Dinamikus adatmegjelenítés a termékspecifikációk és árak teljes körű prezentálására; hibrid

adatbetöltési stratégia alkalmazása, ahol a kritikus információk **Eager Loading** (Include) használatával azonnal, míg a másodlagos adatok aszinkron módon, a háttérben töltődnek be a válaszidő minimalizálása és a gördülékeny felhasználói élmény (UX) érdekében.

- **PWA Támogatás (Progressive Web App):** Alkalmazásszerű felhasználói élmény (UX) biztosítása a Web App Manifest integrálásával; platformfüggetlen telepíthetőség ("Add to Home Screen") és teljes képernyős (Standalone) megjelenítés a böngésző UI-elemeinek elrejtésével, valamint az egyedi ikonok és indítóképernyők kezelése a natív app-érzet elérése érdekében.
- **Bolt adatlap (Shop Page):** Az üzletspecifikus törzsadatok (elérhetőség, nyitvatartás, pontos lokáció) és a teljes kapcsolódó termékkínálat dinamikus vizualizációja; az adatok hatékony betöltése az üzlet azonosítója alapján, valamint a térképes koordináták megjelenítése a felhasználói tájékozódás segítésére.

1.5.2. Genzelmann Erik

- **Kereső algoritmus:** Hatékony adatszűrési és lekérdezési logika implementálása a termékek és boltok gyors eléréséhez; kulcsszó alapú keresés, prediktív találati lista és rangsorolás kialakítása.
- **Főoldal design és felépítés:** Kártyaalapú, reszponzív elrendezés a legfontosabb termékkadatok (név, kép, kategória) strukturált prezentálásához
- **Termékszerkesztés (User-driven Update):** Interaktív CRUD-funkcionalitás kialakítása a termékkadatok naprakészen tartásához; a felhasználói beviterek szerveroldali **validálása** és jogosultság-ellenőrzése a módosítási folyamat során, valamint az adatok aszinkron frissítése az adatbázisban.
- **Bolt szerkesztése:** Elkülönített adminisztrációs felület az üzleti információk (pl. név, cím, nyitvatartási idők) módosítására

1.5.3. Kollár András

- **Bolt alaprajz tervező:**
 - **Interaktív alaprajz-tervező (Layout Editor):** Dinamikus 2D-s szerkesztőfelület a bolt belső struktúrájának kialakításához; falak és polcok rajzolása, törlése, valamint koordináta-alapú áthelyezése a fizikai elrendezés pontos leképezése érdekében.
 - **Polckiosztás és termékpozicionálás:** Termékek logikai hozzárendelése a polcelemekhez, és a polcon belüli pontos helyzetük meghatározása; térbeli adatok kezelése az áruházi navigáció támogatásához, lehetővé téve a termékek fizikai elhelyezkedésének vizuális követését.
 - **WASM alapú kliensoldali logika és API:** Nagy teljesítményű **WebAssembly (WASM)** architektúra a szerkesztőfelület logikájának futtatásához; dedikált API végpontok kidolgozása a szerkesztett alaprajzok és polkonfigurációk perzisztens mentéséhez.
 - **Mobil-optimalizált interakciók:** Precíz, érintőképernyőre tervezett szerkesztési funkciók és finomhangolt interakciók a gördülékeny mobilhasználat érdekében.
- **Görgetés alapú dinamikus adatbetöltés (Lazy Load):** A főoldali termékkínálat folyamatos, görgetéshez kötött betöltése a kezdeti válaszidő minimalizálása érdekében; az adatok aszinkron, szakaszos lekérése a hálózati terhelés optimalizálása és a zökkenőmentes, végtelenített böngészési élmény biztosítása céljából.

2. Fejlesztési dokumentáció

2.1. Technológiák

2.1.1. Front-end

HTML: A HTML a weboldalak szerkezeti felépítéséért felelős leírónyelv.

Razor: Egy olyan szintaxis, amely lehetővé teszi C# kód beágyazását a HTML-be, így dinamikus tartalomgenerálást tesz lehetővé.

CSS: A stíluslapok (Cascading Style Sheets) a weboldalak vizuális megjelenéséért, az elrendezésért, a színekért és a tipográfiáért felelősek.

JavaScript: Egy magas szintű, interpretált programozási nyelv, amely a böngészőben futva teszi interaktívá a weboldalakat, lehetővé téve a dinamikus tartalomfrissítést és eseménykezelést.

Bootstrap: A legnépszerűbb nyílt forráskódú CSS-keretrendszer, amely előre definiált osztályokkal és komponensekkel segíti a reszponzív és modern felhasználói felületek gyors kialakítását.

Zxing (Zebra Crossing): Egy nyílt forráskódú, többplatformos képfeldolgozó könyvtár, amely különféle **1D és 2D vonalkódformátumok** (például EAN-termékkódok és QR-kódok) felismerésére és dekódolására szolgál. Alkalmazása lehetővé teszi, hogy a mobileszközök kamerája digitális bemeneti eszközként funkcionáljon, így biztosítva a termékek azonnali, automatizált azonosítását a fizikai vonalkódok beolvasásával.

2.1.2. Back-end

ASP.NET: A Microsoft által fejlesztett nyílt forráskódú keretrendszer modern, felhőalapú webes alkalmazások és szolgáltatások készítésére.

Blazor client: Lehetővé teszi interaktív webes felhasználói felületek készítését C# nyelven JavaScript helyett. A böngészőben fut WebAssembly segítségével, így kliensoldali logikát valósíthatunk meg .NET alapokon.

Identity framework: Egy komplex tagsági (membership) rendszer, amely kezeli a felhasználók regisztrációját, bejelentkezését, a jelszavak titkosítását, valamint a szerepkör alapú jogosultságkezelést.

2.1.3. Adatbázis-kezelés

MSSQL: A Microsoft relációs adatbázis-kezelő rendszere (RDBMS), amely magas szintű adatbiztonságot, skálázhatóságot és hatékony adatlekérdezési lehetőségeket biztosít.

Entity Framework Core: Egy modern objektum-relációs leképző (ORM), amely lehetővé teszi a fejlesztők számára, hogy az adatbázissal .NET objektumokon keresztül dolgozzanak, minimalizálva a manuális SQL-írást.

2.1.4. Kollaboráció

Discord: Kommunikációs platform, amely szöveges, hang- és videóalapú kapcsolattartást biztosít a fejlesztői csapatok számára, segítve a gyors információáramlást.

LiveShare: A Visual Studio kiterjesztése, amely lehetővé teszi a fejlesztők számára a valós idejű közös kódolást (pair programming) és hibakeresést ugyanabban a projektben.

GitHub Projects (Issue, Kanban): Projektmenedzsment eszköz, amely a Kanban módszertan segítségével vizualizálja a feladatok állapotát, az Issue-k pedig a hibajegyek és fejlesztési igények követésére szolgálnak.

2.1.5. Verziókezelés

Git: Elosztott verziókezelő rendszer, amely nyomon követi a forráskódban történt változtatásokat, lehetővé teszi a korábbi verziókhöz való visszatérést és a párhuzamos munkavégzést (branching).

Github: Felhőalapú platform a Git tárolók (repository-k) hosztolására, amely közösségi fejlesztési és automatizációs funkciókkal egészíti ki a verziókezelést.

2.1.6. Külső API-k

Arfigyelo api: A platform nyilvános végpontjaira épülő megoldás, amely a hálózati forgalom (Network traffic) elemzése során azonosított publikus adatforrásokat használja. A modul feladata a **napi szintű, automatizált árszinkronizáció**, amely biztosítja az alkalmazásban szereplő termékkatádatok folyamatos frissességét és hitelességét.

SmarterMail api: Levelezőszerver-interfész, amelyen keresztül az alkalmazás programozott módon képes e-maileket küldeni (pl. visszaigazolások, értesítések) és kezelní a fiókokat.

Open Food Facts api: Egy globális, közösségi szerkesztésű, nyílt forráskódú élelmiszer-adatbázis elérését biztosító interfész. Lehetővé teszi a termékek vonalkód (EAN) alapú azonosítását, valamint a hozzájuk tartozó részletes metaadatok – például összetevők, tápértékadatok, allergének és környezeti lábnyom (Eco-score) – lekérdezését.

2.1.7. AI

Gemini 3: A Google legújabb multimodális nagy nyelvi modellje, amely komplex szöveges és logikai feladatok megoldására, kódgenerálásra és adatelemzésre képes.

Claude: Az Anthropic által fejlesztett AI modell, amely kiemelkedő képességekkel rendelkezik a pontos szövegértelmezésben, a biztonságos kódolásban és a természetes nyelvű kommunikációban.

2.2. Adatbázis

2.2.1. Bevezetés

A ShopScout rendszer központi elemét a strukturált adatbázis-kezelő réteg alkotja, amely a perzisztens adattárolásért, az adatintegritás fenntartásáért és az információk hatékony lekérdezhetőségéért felelős. Az adatbázis elsődleges feladata a rendszerfolyamatok során keletkező dinamikus adatok biztonságos tárolása és logikai egységbe rendezése.

2.2.2. Az adatbázis struktúrája és normalizáltsága

A tervezési fázisban kiemelt szempont volt a redundancia minimalizálása és az adatok közötti logikai ellentmondások elkerülése. Ennek érdekében az adatbázis sémája a negyedik normálformának (4NF) megfelelően lett kialakítva.

A 4NF alkalmazása biztosítja, hogy:

- A rendszer mentes legyen a többszörös értékű függőségektől (*multi-valued dependency*), így az egymástól független, de egyazon entitáshoz tartozó adathalmazok (például egy termék különböző üzletekben elérhető árai és a termékhez tartozó kategóriák) külön táblákba kerültek.
- Minimalizáltuk az adatduplikációt, ami jelentősen növeli a tárolás hatékonyságát és csökkenti a módosítási anomáliák kockázatát.
- Az adatok közötti kapcsolatok tiszták és átláthatóak maradnak a rendszer skálázódása esetén is.

2.2.3. Technológiai implementáció: Entity Framework Core

A fejlesztés során az Entity Framework Core (EF Core) objektum-relációs leképező (ORM) keretrendszer alkalmaztuk. Ez a technológia magas szintű absztrakciót biztosít, amely közvetítő közegként szolgál a C# nyelv objektumorientált paradigmája és a relációs adatbázis-motor között.

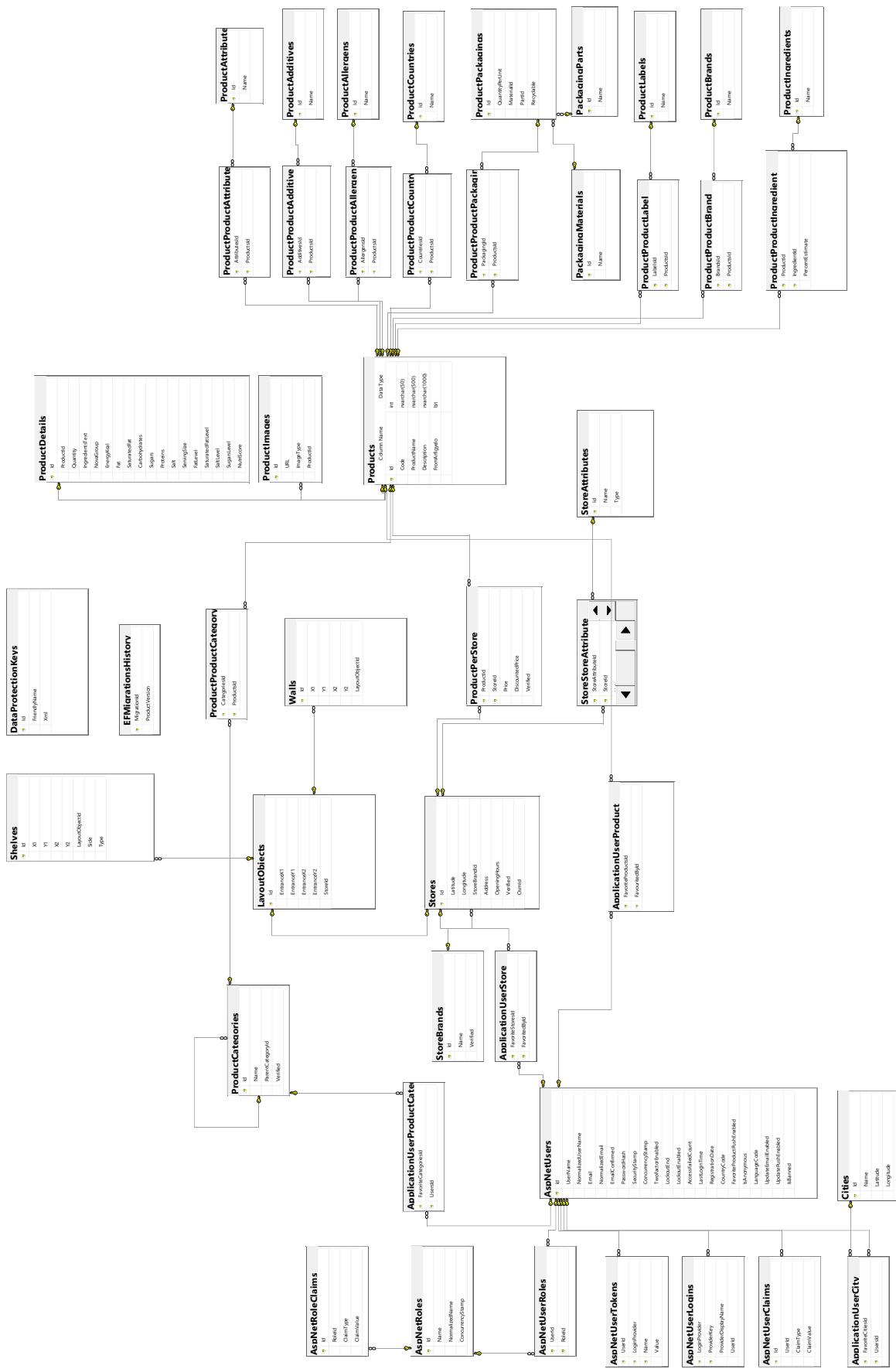
Az implementáció során a **Code-First** megközelítést követtük, amely az alábbi módszertani előnyöket biztosította:

- **Modell-vezérelt fejlesztés:** Az adatbázis fizikai sémáját C# osztályok (adatmodellek) formájában definiáltuk. Ez garantálja, hogy a forráskód és az adatstruktúra minden fázisban szinkronban maradjon.
- **Automatizált séma-migráció:** Az EF Core migrációs mechanizmusait használva az adatbázis szerkezeti változtatásai verziókövetetté váltak. A modellek módosítása után a keretrendszer automatikusan generálta le a szükséges SQL DDL utasításokat.

- **Típusbiztos lekérdezés-kezelés:** A nyers SQL-kódok helyett a LINQ (Language Integrated Query) technológiát alkalmaztuk, amely már fordítási időben képes kiszűrni a hibákat, növelte ezzel a rendszer stabilitását és karbantarthatóságát.

Az EF Core és a szigorú normalizálási elvek együttes alkalmazása lehetővé tette egy olyan adatbázis-réteg létrehozását, amely stabil alapot biztosít a ShopScout komplex keresési és dinamikus adatkezelési műveleteihez.

2.2.4. Teljes adatbázis diagram



2.2.5. Táblák

2.2.5.1. Stores

```
1 public class Store : IVerifiable
2 {
3     public int Id { get; set; }
4     public long? OsmId { get; set; }
5     public string Latitude { get; set; }
6     public string Longitude { get; set; }
7     public string? Address { get; set; }
8     public bool Verified { get; set; } = false;
9     public string? OpeningHours { get; set; }
10    public virtual LayoutObject? Layout { get; set; }
11
12    // Navigation Property
13    public virtual StoreBrand? StoreBrand { get; set; }
14    public virtual ICollection<ProductPerStore> ProductPerStore { get; set; } = new
15        List<ProductPerStore>();
16    public virtual ICollection< ApplicationUser> FavoritedBy { get; set; } = new
17        List< ApplicationUser>();
18    public virtual ICollection<StoreStoreAttribute> StoreAttributes { get; set; } = new
19        List<StoreStoreAttribute>();
20    public virtual ICollection<StoreChange> Changes { get; set; } = new List<StoreChange>();
21 }
```

	Név	Adattípus	Leírás/Attribútum
🔍	Id	int	Egyedi azonosító (PK)
	Latitude	nvarchar(MAX)	Szélességi koordináta
	Longitude	nvarchar(MAX)	Hosszúsági koordináta
🔍	StoreBrandId	int	Márka azonosítója (FK)
	Address	nvarchar(MAX)	Az üzlet pontos címe
	OpeningHours	nvarchar(MAX)	Nyitvatartási rend
	Verified	bit	Ellenőrzöttségi állapot
	OsmId	bigint	OpenStreetMap azonosító

A fenti **Stores** tábla az üzlethelyiségek központi adatbázisa, amely a fizikai egységek helyszíni és üzleti adatait rendszerezzi. Az üzletek digitális azonosításának és térképes leképezésének szerepét tölti be, összekapcsolva a földrajzi koordinátákat, a pontos címet és a márkat a naprakész, hiteles adatszolgáltatás érdekében.

2.2.5.2. StoreBrands

```
1 public class StoreBrand : INTToNTable, IVerifiable
2 {
3     public int Id { get; set; }
4     public string Name { get; set; }
5     public bool Verified { get; set; } = false;
6
7     public virtual ICollection<Store> Stores { get; set; } = new List<Store>();
8 }
```

	Név	Adattípus	Leírás/Attribútum
🔍	Id	int	Egyedi azonosító (PK)
	Name	nvarchar(50)	Az üzletlánc megnevezése
	Verified	bit	Ellenőrzöttségi állapot

A **StoreBrand** entitás elsődleges feladata az üzlethálózatok vagy márkák – például a megnevezés és a hitelességi állapot – típusbiztos tárolása, valamint a kapcsolódó üzletekkel való reláció fenntartása.

2.2.5.3. Products

```

1 public class Product
2 {
3     [Key]
4     public int Id { get; set; }
5
6     [Required]
7     [MaxLength(50)]
8     public string Code { get; set; } = "";
9
10    public bool FromArfigyelo { get; set; } = false;
11
12    [MaxLength(500)]
13    public string ProductName { get; set; } = "";
14
15    [MaxLength(1000)]
16    public string? Description { get; set; }
17
18    // Navigation Properties
19    public virtual ProductDetails? Details { get; set; }
20    public virtual ICollection<ProductProductIngredient> ProductIngredients { get; set; } = new
        List<ProductProductIngredient>();
21    public virtual ICollection<ProductImage> ProductImages { get; set; } = new List<ProductImage>();
22    public virtual ICollection<ProductAllergen> Allergens { get; set; } = new List<ProductAllergen>();
23    public virtual ICollection<ProductAdditive> Additives { get; set; } = new List<ProductAdditive>();
24    public virtual ICollection<ProductLabel> Labels { get; set; } = new List<ProductLabel>();
25    public virtual ICollection<ProductPackaging> Packaging { get; set; } = new List<ProductPackaging>();
26    public virtual ICollection<ProductAttribute> Attributes { get; set; } = new List<ProductAttribute>();
27    public virtual ICollection<ProductPerStore> ProductPerStore { get; set; } = new List<ProductPerStore>();
28    public virtual ICollection<ProductBrand> Brands { get; set; } = new List<ProductBrand>();
29    public virtual ICollection<ProductCountry> Countries { get; set; } = new List<ProductCountry>();
30    public virtual ICollection<ProductCategory> Categories { get; set; } = new List<ProductCategory>();
31    public virtual ICollection<ProductChange> Changes { get; set; } = new List<ProductChange>();
32    public virtual ICollection<ApplicationUser> FavouredBy { get; set; } = new List<ApplicationUser>();
33 }

```

	Név	Adattípus	Leírás/Attribútum
🔍	Id	int	Egyedi azonosító (PK)
	Code	nvarchar(50)	Termék vonalkódja
	ProductName	nvarchar(500)	Termék megnevezése
	Description	nvarchar(1000)	Rövid termékleírás
	FromArfigyelo	bit	Árfigyelőből származik-e

A **Products** tábla a rendszer központi terméknyilvántartója, amely az árucikkek legfontosabb törzsadatait. A modell struktúrája szándékosan különválasztja ezeket az alapadatokat a kiterjedt kiegészítő információktól – mint a fotók, kategóriák vagy allergének –, amelyek navigációs tulajdonságokon keresztül, külön kapcsolódó táblákban érhetők el. Ez az architektúra hatékony adatkezelést tesz lehetővé, hiszen keresésnél vagy listázásnál a rendszernek csak a legszükségesebb részleteket kell betöltenie, ami jelentősen csökkenti az erőforrás-használatot és gyorsítja a felhasználói felület megjelenítését.

2.2.5.4. ProductDetails

```
1 public class ProductDetails
2 {
3     [Key]
4     public int Id { get; set; }
5     [MaxLength(100)]
6     [ForeignKey("Product")]
7     public int ProductId { get; set; }
8     public string? Quantity { get; set; }
9     [MaxLength(4000)]
10    public string? IngredientsText { get; set; }
11    public NutriScore NutriScore { get; set; }
12    public byte NovaGroup { get; set; }
13    public double? EnergyKcal { get; set; }
14    public double? Fat { get; set; }
15    public double? SaturatedFat { get; set; }
16    public double? Carbohydrates { get; set; }
17    public double? Sugars { get; set; }
18    public double? Proteins { get; set; }
19    public double? Salt { get; set; }
20    [MaxLength(100)]
21    public string? ServingSize { get; set; }
22    public NutrientLevel? FatLevel { get; set; }
23    public NutrientLevel? SaturatedFatLevel { get; set; }
24    public NutrientLevel? SaltLevel { get; set; }
25    public NutrientLevel? SugarsLevel { get; set; }
26    // Navigation Property
27    public virtual Product Product { get; set; } = null!;
28 }
29
30 public enum NutriScore : byte
31 {
32     Unknown, A, B, C, D, E
33 }
```

A **ProductDetails** entitás a termékek részletes kiegészítő adatait tárolja, mint például a tápértékeket, a kiszerelést és a feldolgozottsági szinteket. A modell a termék tápanyag-besorolását közvetlenül a **NutriScore enumhoz** rendeli hozzá, amely rögzített kategóriák (A–E) alapján határozza meg a besorolást.

	Név	Adattípus	Leírás/Attribútum
🔍	Id	int	Egyedi azonosító (PK)
🔍	ProductId	Int	A kapcsolódó termék egyedi azonosítója (FK)
	Quantity	nvarchar(MAX)	A termék kiszerelése vagy mennyiségi adatai
	IngredientsText	nvarchar(4000)	Az összetevők részletes szöveges felsorolása
	NovaGroup	tinyint	Élelmiszer-feldolgozottsági szintet jelölő mutató
	EnergyKcal	float	A termék energiatartalma kilokalóriában
	Fat	float	Zsírtartalom mennyisége
	SaturatedFat	float	Telített zsírsavak mennyisége
	Carbohydrates	float	Szénhidráttartalom
	Sugars	float	Cukortartalom
	Proteins	float	Fehérjetartalom
	Salt	float	Sótartalom
	ServingSize	nvarchar(100)	Ajánlott adagolási egység mérete
	FatLevel	tinyint	Zsírtartalom relatív szintjének minősítése.
	SaturatedFatLevel	tinyint	Telített zsírsavak relatív szintjének minősítése.
	SaltLevel	tinyint	Sótartalom relatív szintjének minősítése.
	SugarsLevel	tinyint	Cukortartalom relatív szintjének minősítése.
	NutriScore	tinyint	Tápérték-besorolás szerinti minősítés.

2.2.5.5. ProductCategory

```
1 public class ProductCategory : INToNTable, IVerifiable
2 {
3     [Key]
4     public int Id { get; set; }
5     public string Name { get; set; }
6     public bool Verified { get; set; } = false;
7     public ProductCategory? ParentCategory { get; set; }
8
9     // Navigation Property
10    public virtual ICollection<Product> Products { get; set; } = new List<Product>();
11    public virtual ICollection<ApplicationUser> Users { get; set; } = new
12        List<ApplicationUser>();
13    public virtual ICollection<ProductCategory> SubCategories { get; set; } = new
14        List<ProductCategory>();
15 }
```

	Név	Adattípus	Leírás/Attribútum
🔍	Id	int	Egyedi azonosító (PK)
	Name	nvarchar(450)	Kategória megnevezése
🔍	ParentCategoryId	int	Szülő kategória azonosító
	Verified	bit	Ellenőrzöttségi állapot

A **ProductCategory** entitás a termékek rendszerezését teszi lehetővé, ahol egy termék egyszerre több kategóriához is tartozhat.

Az egyik legfontosabb technikai jellemzője, hogy **önmagára hivatkozik (self-referencing)**: a ParentCategory (szülő) és a SubCategories (alkategóriák) tulajdonságok révén az adatbázis egy fa struktúrát alkot. Ez a felépítés lehetővé teszi a kategóriák tetszőleges mélységű egymásba ágyazását (például: Élelmiszer > Tejtermék > Sajtok).

2.2.5.6. ProductImages

```
1 public class ProductImage
2 {
3     public int Id { get; set; }
4     public string? URL { get; set; }
5     public ProductImageType ImageType { get; set; }
6     public virtual Product Product { get; set; }
7 }
8
9 public enum ProductImageType
10 {
11     Primary,
12     Ingredients,
13     Nutrition,
14     Packaging
15 }
```

	Név	Adattípus	Leírás/Attribútum
🔍	Id	int	Egyedi azonosító (PK)
	URL	nvarchar(MAX)	A kép webcíme
	ImageType	int	A kép típusa (főkép, összetevők stb.)
🔍	ProductId	int	A kapcsolódó termék azonosítója

A **ProductImage** entitás a termékekhez kapcsolódó vizuális információk és fényképes dokumentációk kezeléséért felel. Fontos technikai részlet, hogy az adatbázis nem magukat a képfájlokat, hanem kizártlag azok **URL-címeit tárolja**, amelyek külső **CDN (Content Delivery Network)** szerverekre mutatnak.

A rendszer a képek rendeltetését a **ProductImageType enumhoz** rendeli hozzá, amely lehetővé teszi a főkép (Primary), az összetevők (Ingredients), a tápterhéktáblázat (Nutrition) és a csomagolás (Packaging) fotóinak elkülönített azonosítását.

2.2.5.7. StoreAttribute

```
1 public class StoreAttribute
2 {
3     public int Id { get; set; }
4     public string Name { get; set; }
5     public StoreAttributeType Type { get; set; }
6
7     // Navigation Properties
8     public virtual ICollection<StoreStoreAttribute> Stores { get; set; } = new
9     List<StoreStoreAttribute>();
10
11 public enum StoreAttributeType
12 {
13     Payment,
14     Contact,
15     Other
16 }
```

	Név	Adattípus	Leírás/Attribútum
🔍	Id	int	Egyedi azonosító (PK)
	Name	nvarchar(MAX)	Attribútum megnevezése
	Type	tinyint	Attribútum típusa

A **StoreAttribute** entitás elsődleges feladata az üzleti egységekhez rendelhető dinamikus jellemzők – például fizetési módok és elérhetőségek – típusbiztos tárolása.

2.2.5.8. ProductPackagings

```
1 public class ProductPackaging
2 {
3     [Key]
4     public int Id { get; set; }
5     public bool? Recyclable { get; set; }
6
7     [MaxLength(100)]
8     public string? QuantityPerUnit { get; set; }
9
10    // Navigation Property
11    public virtual PackagingMaterial? Material { get; set; }
12    public virtual PackagingPart? Part { get; set; }
13    public virtual ICollection<Product> Products { get; set; } = new List<Product>();
14 }
```

	Név	Adattípus	Leírás/Attribútum
🔍	Id	int	Egyedi azonosító (PK)
	QuantityPerUnit	nvarchar(100)	Az egységnyi kiszereles szöveges meghatározása
🔍	MaterialId	int	A felhasznált csomagolóanyag azonosítója (FK)
🔍	PartId	int	A csomagolási rész azonosítója (FK)
	Recyclable	bit	Az újrahasznosíthatóságot jelző logikai (bit) mutató

A **ProductPackaging** entitás elsődleges feladata a termékek csomagolási jellemzőinek – például az újrahasznosíthatóságnak és a kiszerelesi egységeknek – a strukturált tárolása. A modell biztosítja a terméktörzs, valamint a specifikus csomagolóanyagok és típusok közötti típusbiztos kapcsolatot.

2.2.5.9. ProductPerStore

```

1 public class ProductPerStore : IVerifiable
2 {
3     // Composite Key: ProductId + StoreId
4     public int ProductId { get; set; }
5     public int StoreId { get; set; }
6
7     public virtual Store Store { get; set; }
8     public virtual Product Product { get; set; }
9     public int? Price { get; set; }
10    public int? DiscountedPrice { get; set; }
11    public bool Verified { get; set; } = false;
12 }
```

	Név	Adattípus	Leírás/Attribútum
🔍	ProductId	int	A kapcsolt termék azonosítója (FK)
🔍	StoreId	int	A kapcsolt bolt azonosítója (FK)
	Price	int	A termék adott üzletben érvényes ára
	DiscountedPrice	int	A termékre vonatkozó aktuális kedvezményes értékesítési ár
	Verified	bit	Ellenőrzöttségi állapot

A **ProductPerStore** entitás elsődleges feladata a termékek üzletágankénti elérhetőségének és egyedi árazásának – például az alapár és az akciós ár – a típusbiztos tárolása. A modell megvalósítja a termékek és üzletek közötti kapcsolati hálót, kiegészítve azt az adatok hitelességét igazoló ellenőrzési logikával.

2.2.5.10. LayoutObjects

```

1 public class LayoutObject
2 {
3     public int Id { get; set; }
4
5     public int EntranceX1 { get; set; }
6     public int EntranceY1 { get; set; }
7     public int EntranceX2 { get; set; }
8     public int EntranceY2 { get; set; }
9
10    [ForeignKey(nameof(Store))]
11    public int StoreId { get; set; }
12    public virtual Store Store { get; set; } = null!;
13
14    public virtual ICollection<Wall> Walls { get; set; } = new List<Wall>();
15    public virtual ICollection<Shelf> Shelves { get; set; } = new List<Shelf>();
16 }
```

	Név	Adattípus	Leírás/Attribútum
🔍	Id	int	Egyedi azonosító (PK)
	X1	int	Az üzlet bejáratának kezdőpontját meghatározó koordináták
	Y1	int	Az üzlet bejáratának kezdőpontját meghatározó koordináták
	X2	int	Az üzlet bejáratának végpontját meghatározó koordináták
	Y2	int	Az üzlet bejáratának végpontját meghatározó koordináták
	StoreId	int	A kapcsolódó üzletre mutató idegen kulcs

A **LayoutObject** entitás elsődleges feladata az üzleti egységek térbeli szerkezetének – például a bejáratok koordinátáinak és a belső elrendezésnek – a típusbiztos tárolása. A modell összefogja az adott üzlethez tartozó falak és polcok rendszerét, biztosítva a fizikai alaprajz digitális leképezését.

2.2.5.11. Walls

```
1 public class Wall
2 {
3     public int Id { get; set; }
4     public int X1 { get; set; }
5     public int Y1 { get; set; }
6     public int X2 { get; set; }
7     public int Y2 { get; set; }
8
9     public virtual LayoutObject LayoutObject { get; set; } = null!;
10 }
```

	Név	Adattípus	Leírás/Attribútum
🔍	Id	int	Egyedi azonosító (PK)
	EntranceX1	int	A fal szakaszának kezdőpontját meghatározó koordináták
	EntranceY1	int	A fal szakaszának kezdőpontját meghatározó koordináták
	EntranceX2	int	A fal szakaszának végpontját meghatározó koordináták
	EntranceY2	int	A fal szakaszának végpontját meghatározó koordináták
🔍	LayoutObjectId	int	A falat tartalmazó alaprajzi egység azonosítója (FK)

A **Wall** entitás elsődleges feladata az üzlettér fizikai határainak – például a falak szakaszait definiáló koordinátáknak – a típusbiztos tárolása. A modell a kezdő- és végpontok rögzítésével teszi lehetővé a belső tér geometriai leképezését, szoros egységen a befoglaló alaprajzi objektummal.

2.2.5.12. Shelves

```

1 public class Shelf
2 {
3     public int Id { get; set; }
4     public int X1 { get; set; }
5     public int Y1 { get; set; }
6     public int X2 { get; set; }
7     public int Y2 { get; set; }
8     public ShelfType Type { get; set; } = ShelfType.Shelf;
9     public ShelfSide Side { get; set; }
10
11    public virtual LayoutObject LayoutObject { get; set; } = null!;
12 }
13
14 public enum ShelfType
15 {
16     [Display(Name = "Polc")] Shelf,
17     [Display(Name = "Hűtő, fagyasztó")] Fridge,
18     [Display(Name = "Zöldség, gyümölcs")] Produce,
19     [Display(Name = "Pékáru")] Bakery,
20     [Display(Name = "Húspult")] Butcher,
21 }
22
23 public enum ShelfSide
24 {
25     [Display(Name = "Bal")] Left,
26     [Display(Name = "Jobb")] Right,
27     [Display(Name = "Mindkettő")] Both,
28 }

```

	Név	Adattípus	Leírás/Attribútum
🔍	Id	int	Egyedi azonosító (PK)
	X1	int	Kezdőpont koordinátái
	Y1	int	Kezdőpont koordinátái
	X2	int	Végpont koordinátái
	Y2	int	Végpont koordinátái
	LayoutObjectId	int	A befoglaló alaprajzi objektumhoz tartozó idegen kulcs (FK)
	Side	int	A polc hozzáférhetőségi oldala
	Type	int	A polc funkcionális besorolása (pl. hűtő, pékáru)

A **Shelf** entitás elsődleges feladata az üzlettérben található berendezési tárgyak – például polcok, hűtők és pultok – pontos helyzetének, típusának és elhelyezkedésének típusbiztos tárolása.

2.2.5.13. Id-Name táblák

```
1 public class ProductAdditive : INTToNTable
2 {
3     [Key]
4     public int Id { get; set; }
5
6     [MaxLength(100)]
7     public string Name { get; set; }
8
9     // Navigation Property
10    public virtual ICollection<Product> Products { get; set; } = new List<Product>();
11 }
```

	Név	Adattípus	Leírás/Attribútum
🔍	Id	int	Egyedi azonosító (PK)
	Name	nvarchar(100)	Kiegészítő adat

Az Id-Name táblák szerepe

Ezeknek a tábláknak a feladata a **normalizálás**: ahelyett, hogy minden terméknél szövegesen (stringként) tárolnánk az ismétlődő adatokat (például országneveket vagy márkkákat), egy központi táblában rögzítjük őket. Ez biztosítja az adatkonzisztenciát és csökkenti az adatbázis méretét.

A projektben található Id-Name táblák

- ProductAttributes: A termékek különféle tulajdonságai.
- ProductAdditives: Az élelmiszeripari adalékanyagok (E-számok) gyűjteménye.
- ProductAllergens: A lehetséges allergének listája.
- ProductCountries: A származási országok megnevezései.
- ProductLabels: Különféle minősítések és címek.
- ProductBrands: A rendszerben szereplő márkanevök.
- ProductIngredients: Az összetevők törzsadat-listája.
- PackagingMaterials: A csomagoláshoz használt anyagtípusok.
- PackagingParts: A csomagolás részegységeinek megnevezései.

2.2.5.14. Kapcsolótáblák

Az Entity Framework Core alkalmazásával a **több a többhöz (N:N)** kapcsolatok menedzselése rendkívül hatékonnyá válik. A keretrendszer a C# modellekben deklarált gyűjtemények (jellemzően ICollection<T>) alapján képes automatikusan generálni a szükséges kapcsolótáblákat az adatbázis-sémában a migrációk során.

Ez a megközelítés magasabb szintű absztrakciót biztosít, így a fejlesztés során nincs szükség a kapcsolótáblák közvetlen, manuális kezelésére. Elegendő az érintett entitások gyűjteményein keresztül hivatkozni a kapcsolódó adatokra. Amikor például egy felhasználó profiljához egy új kedvenc üzletet rendelünk, az EF Core a háttérben – a mentési folyamat részeként – automatikusan rögzíti a megfelelő azonosító-párosításokat a köztes táblában. Ez a megoldás nemcsak a kódot teszi átláthatóbbá, hanem csökkenti a manuális adatbázis-műveletekből eredő hibalehetőségeket is.

	Név	Adattípus	Leírás/Attribútum
	Pl.: AdditivesId	int	Kiegészítő tábla kulcs (PK)
	Pl.: ProductsId	int	Product tábla kulcs (PK)

Termékhez kapcsolódó táblák:

- **ProductProductCategory:** Összeköti a termékeket a hozzájuk tartozó kategóriákkal.
- **ProductProductAttribute:** A termékek és az egyedi tulajdonságok közötti kapcsolat.
- **ProductProductAdditive:** Meghatározza, melyik termékben milyen adalékanyagok találhatók.
- **ProductProductAllergen:** A termékek és a bennük lévő allergének párosítása.
- **ProductProductCountry:** A termékek származási országait rögzíti.

- **ProductProductPackaging:** A termékeket rendeli hozzá a csomagolási adatokhoz.
- **ProductProductLabel:** A termékekhez tartozó minősítéseket és címkéket kezeli.
- **ProductProductBrand:** A termékek és márkkák közötti kapcsolatot tárolja.
- **ProductProductIngredient:** Az összetevők és termékek kapcsolata (itt extra adatként a százalékos arány is rögzítésre kerül).

Felhasználói és üzleti adatok kapcsolatai:

- **ApplicationUserProductCategory:** A felhasználók által kedvelt termékkategóriákat tárolja.
- **ApplicationUserStore:** A felhasználók kedvenc üzleteinek listáját kezeli.
- **ApplicationUserProduct:** A felhasználók által kedvencnek jelölt termékek gyűjteménye.
- **ApplicationUserCity:** A felhasználók által kedvencnek jelölt városok gyűjteménye.
- **StoreStoreAttribute:** Az üzletek és a hozzájuk tartozó extra tulajdonságok (pl. parkoló, akadálymentesített) kapcsolata.
- **AspNetUserRoles:** Feladata a jogosultságkezelés biztosítása: lehetővé teszi, hogy egy felhasználó egyszerre több szerepkörrel (például Adminisztrátor és Vásárló) rendelkezzen.

2.2.5.15. Beépített Identity Framework táblák

Az **ASP.NET Core Identity** keretrendszer beépített táblái a felhasználók hitelesítéséért (authentication) és jogosultságkezeléséért (authorization) felelnek. Biztosítják a biztonságos jelszótárolást, a kétképpes azonosítást, a szerepkörök (admin, user stb.) kezelését és a külső szolgáltatókkal (pl. Google) való bejelentkezést.

AspNetRoles

A rendszerben használható felhasználói szerepkörök (pl. Admin, User) meghatározása.

	Név	Adattípus	Leírás
	Id	nvarchar(450)	Szerepkör egyedi azonosítója (PK).
	Name	nvarchar(256)	Szerepkör megnevezése.
	NormalizedNome	nvarchar(256)	Keresést segítő normalizált név.
	ConcurrencyStamp	nvarchar(MAX)	Adatütközés-kezelési jelző.

AspNetUserTokens

Biztonsági tokenek tárolása (pl. jelszó-visszaállítás).

	Név	Adattípus	Leírás
	UserId	nvarchar(450)	Felhasználó azonosító (FK).
	LoginProvider	nvarchar(450)	Kibocsátó neve.
	Name	nvarchar(450)	Token neve.
	Value	nvarchar(MAX)	Token értéke.

AspNetUsers

A felhasználók központi tárolására és alapvető hitelesítési adataik kezelésére szolgáló tábla.

	Név	Adattípus	Leírás/Attribútum
	Id	nvarchar(450)	Egyedi azonosító (PK).
	UserName	nvarchar(256)	A felhasználó egyedi bejelentkezési neve.
	NormalizedUserName	nvarchar(256)	A felhasználónév normalizált formája kereséshez.
	Email	nvarchar(256)	Kapcsolattartási e-mail cím.
	NormalizedEmail	nvarchar(256)	Az e-mail cím normalizált formája kereséshez.
	EmailConfirmed	bit	Az e-mail cím hitelesítettségének jelzője
	PasswordHash	nvarchar(MAX)	A jelszó titkosított lenyomata
	SecurityStamp	nvarchar(MAX)	Biztonsági ellenőrző kód a munkamenetekhez
	ConcurrencyStamp	nvarchar(MAX)	Az egyidejű adatmódosítások kezelésére szolgáló jelző
	TwoFactorEnabled	bit	A kétrépcsős azonosítás állapotát jelző mutató
	LockoutEnd	datetimeoffset(7)	A felhasználói kizárási lejáratának pontos ideje
	LockoutEnabled	bit	A fiók ideiglenes zárolhatóságának állapota
	AccessFailedCount	int	A sikertelen bejelentkezési kísérletek számlálója
	LastLoginTime	datetime2(7)	Az utolsó sikeres bejelentkezés időpontja
	RegistrationDate	datetime2(7)	A felhasználói regisztráció dátuma

	CountryCode	nvarchar(MAX)	A felhasználó országkódja
	IsAnonymous	bit	Az anonim felhasználói státuszt jelző mutató
	LanguageCode	nvarchar(MAX)	A preferált nyelvi beállítás
	IsBanned	bit	A felhasználó kitiltott állapotát jelző logikai mutató.

AspNetRoleClaims

Egyes szerepkörökhez rendelt specifikus jogosultságok tárolása.

	Név	Adattípus	Leírás
🔍	Id	int	Bejegyzés azonosítója (PK).
	RoleId	nvarchar(450)	Kapcsolódó szerepkör azonosítója (FK).
	ClaimType	nvarchar(MAX)	Jogosultság típusa.
	ClaimValue	nvarchar(MAX)	Jogosultság értéke.

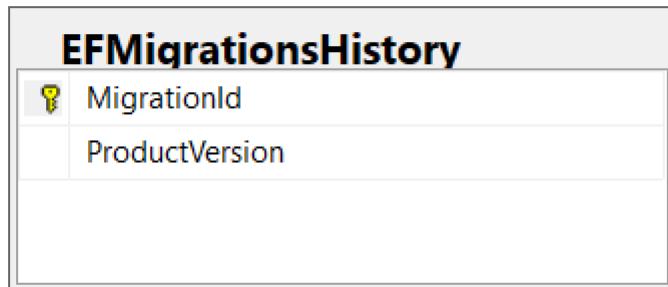
AspNetUserLogins

Külső hitelesítő szolgáltatók (pl. Google) adatai.

	Név	Adattípus	Leírás
🔍	LoginProvider	nvarchar(450)	Szolgáltató neve.
	ProviderKey	nvarchar(450)	Szolgáltatói kulcs.
	ProviderDisplayName	nvarchar(MAX)	Megjelenítendő név.
	UserId	nvarchar(450)	Felhasználó azonosító (FK).

2.2.5.16. Egyéb

__EFMigrationsHistory



Az **__EFMigrationsHistory** tábla az Entity Framework Core keretrendszer belső, rendszerszintű táblája, amely a Code-First fejlesztési folyamat során a séma változások nyomon követéséért felel. Feladata az alkalmazásban létrehozott migrációk naplázása, biztosítva ezzel, hogy a keretrendszer felismerje, mely adatbázis-módosítások kerültek már korábban végrehajtásra. Ennek köszönhetően elkerülhető a redundáns műveletek futtatása, és szinkronban tartható a C# kódban definiált adatmodell az aktuális adatbázis-szerkezettel.

	Név	Adattípus	Leírás/Attribútum
🔍	MigrationId	nvarchar(150)	A migrációs fájl egyedi azonosítója, amely alapján az EF Core azonosítja az adott szerkezeti módosítást.
	ProductVersion	nvarchar(32)	Az Entity Framework Core azon verziószáma, amellyel a migrációt legenerálták.

DataProtectionKeys

DataProtectionKeys	
 Id	
	FriendlyName
	Xml

A **DataProtectionKeys** tábla az ASP.NET Core keretrendszer Data Protection API-jának egyik központi eleme, amely a titkosítási kulcsok perzisztens tárolásáért felel. Feladata, hogy biztonságos helyet biztosítson azoknak a kriptográfiai kulcsoknak, amelyeket a rendszer az érzékeny adatok – például hitelesítési sütik (cookies) vagy állapotadatok – védelmére használ. Ez a megoldás garantálja, hogy az alkalmazás újraindítása után is képes legyen a korábban titkosított adatok visszafejtésére.

	Név	Adattípus	Leírás/Attribútum
 Id		int	Egyedi azonosító (PK)
	FriendlyName	nvarchar(MAX)	A titkosítási kulcs megkülönböztetésére és azonosítására szolgáló megnevezés.
	Xml	nvarchar(MAX)	A titkosítási kulcs adatait tartalmazó, szerializált XML formátumú tartalom.

2.3. Jogosultságok, jogosultsági szintek

2.3.1. Nem regisztrált felhasználó

A **nem regisztrált felhasználók** számára a rendszer korlátozott, kizártlag olvasási hozzáférést biztosít a nyilvános adatokhoz. A vendéghasználók böngészhetik a teljes termékkínálatot, megtekinthetik a boltok adatait és az alaprajzokat, azonban az alábbi műveletek végrehajtására nincs jogosultságuk:

- új termékek feltöltése a rendszerbe;
- meglévő termékek adatainak szerkesztése;
- boltok alaprajzának vagy elrendezésének módosítása;
- termékek vagy üzletek hozzáadása a kedvencek listájához.

A felsorolt funkciók igénybevételéhez a felhasználónak regiszálnia kell és be kell jelentkeznie a saját profiljába.

2.3.2. Regisztrált felhasználó

A **regisztrált felhasználók** a rendszer minden alapfunkciójához hozzáférnek, beleértve a személyes fiókhoz kapcsolódó adminisztratív műveleteket is. Jogosultak a profiladatok, így a felhasználónév, az e-mail-cím és a jelszó módosítására, valamint a felület egyedi igények szerinti személyre szabására.

A bejelentkezett felhasználók az alábbi többletjogosultságokkal rendelkeznek a vendégekkel szemben:

- új termékek feltöltése és a meglévő termékadatok szerkesztése;
- az üzletek alaprajzának és belső elrendezésének módosítása;
- termékek és boltok hozzáadása a kedvencek listájához, valamint azok kezelése;
- a felhasználói fiókhoz tartozó biztonsági beállítások és személyes adatok karbantartása.

2.3.3. Admin

Az adminisztrátori szerepkör a rendszerirányítási felülethez (**Admin Panel**) való teljes körű hozzáférést biztosítja, amelyen keresztül az alkalmazás minden eleme ellenőrizhető és módosítható. Az adminisztrátorok a következő kiemelt funkciókat kezelhetik:

- a felhasználói fiókok adatainak és jogosultsági szintjeinek módosítása;
- a termékek, üzletek és alaprajzok adatainak teljes körű szerkesztése és felügyelete;
- a szervernaplók (**logs**) elérése a rendszeresemények és hibák diagnosztizálásához;
- a forgalmi adatok (**traffic data**) és látogatottsági statisztikák nyomon követése.

Ezek a jogosultságok lehetővé teszik a rendszer zavartalan működésének biztosítását, valamint a tartalom és a felhasználói bázis folyamatos karbantartását.

2.4. Weboldal fejlesztése

2.4.1. Technológiai háttér

A weboldal fejlesztése a modern **ASP.NET Core** keretrendszerre épülő **Blazor United** (Blazor Web App) architektúra mentén valósul meg. Ez a megközelítés lehetővé teszi a szerveroldali és a kliensoldali renderelés előnyeinek ötvözését egyetlen megoldáson belül. A fejlesztés során a forráskód három fő projektre tagolódik, amelyek meghatározott feladatkörökért felelnek a rendszer működése során.

A projektstruktúra felépítése és az egyes egységek feladatai a következők:

ShopScout: Ez a solution központi, szerveroldali projektje. Feladata az alkalmazás hosztolása, a beérkező HTTP-kérések kiszolgálása és a szerveroldali üzleti logika futtatása. Itt kapnak helyet az adatbázis-elérést

biztosító adatkörnyezetek (Entity Framework Core), az API végpontok, valamint az autentikációs és autorizációs folyamatok kezelése. Ez a projekt felel a kezdeti HTML tartalom legenerálásáért (SSR) és a kliensoldali kód letöltésének vezérléséért is.

ShopScout.Client: Ez a projekt tartalmazza a felhasználói felület (UI) azon részeit, amelyek a böngészőben, kliensoldalon futnak **WebAssembly (WASM)** technológia segítségével. Itt találhatók az interaktív komponensek, az ügyféloldali útvonalválasztás (routing) és a böngészőben futó megjelenítési logika. A kliensoldali projekt közvetlenül kommunikál a szerverrel az adatok lekérése és módosítása érdekében.

ShopScout.SharedLib: Ez egy közös osztálykönyvtár (Class Library), amely a kód újrafelhasználhatóságát biztosítja a szerver és a kliens között. Itt definiáltuk azokat a modellek, adatátviteli objektumokat (DTO), interfészket és validációs szabályokat, amelyeket minden fő projekt használ. Ezzel elkerülhető a kódduplikáció, és garantálható, hogy a két oldal azonos adatstruktúrákkal és szabályrendszerrel dolgozzon.

Ez a felosztás tiszta elválasztást biztosít a felelősségi körök között, miközben a megosztott könyvtár révén megkönnyíti a típusbiztos adatkezelést és a rendszer karbantarthatóságát.

2.4.2. Rendering és kommunikáció a szerverrel

Az **Interactive Auto** renderelési mód a Blazor keretrendszer legrugalmásabb megoldása, amely egyesíti a szerveroldali és a kliensoldali futtatás előnyeit.

2.4.2.1. Az Interactive Auto működése és előnyei

Ez a mód hibrid megoldást kínál: az első betöltéskor a komponensek Blazor Server módban indulnak el, így a felhasználó azonnal látja a tartalmat, miközben a **háttérben letöltődnek a WebAssembly futtatásához szükséges fájlok**. A következő látogatáskor a rendszer már automatikusan a kliens erőforrásait használja (WebAssembly), ezzel tehermentesítve a szervert.

Előnyei a hagyományos módokkal szemben:

Gyorsabb kezdés: Elkerüli a tiszta WebAssembly hosszú kezdeti töltési idejét.

Kisebb szerverterhelés: Amint a kliensoldali kód aktívvá válik, a szervernek nem kell fenntartania a folyamatos kapcsolatot (Circuit) minden egyes felhasználó számára.

Folytonos élmény: A váltás a két mód között észrevétlen a felhasználó számára.

2.4.2.2. Szolgáltatások regisztrációja és a Program.cs

A hibrid **Interactive Auto** megjelenítés alapköve az absztrakció. Mivel a komponensek hol a szerveren, hol a kliensoldalon futnak, a függőségek befecskendezésekor (Dependency Injection) egy közös interfész használunk. Ez biztosítja, hogy a felhasználói felület kódja változatlan maradjon, függetlenül attól, hogy az adatok közvetlenül az adatbázisból vagy egy API-n keresztül érkeznek.

A közös interfész (IProductService)

Az interfész a ShopScout.SharedLib projektben definiáljuk, így mind a szerver, mind a kliens oldal látja azt. Ez a kapcsolat garantálja, hogy minden oldal ugyanazokat a metodusokat valósítja meg.

```
1 public interface IProductService
2 {
3     Task<Product?> GetProductAsync(string barcode);
4     Task<List<Product>> GetAllProductsAsync(int page);
5     Task<List<Product>> GetProductsSearchAsync(string search_term, int page);
6     Task<Product?> GetProductByIdAsync(string id);
7     Task AttachRestAsync(Product product);
8 }
```

Szerveroldali regisztráció: Itt az interfészhez a valódi, adatbázis-elérést biztosító osztályt rendeljük hozzá.

```
1 builder.Services.AddScoped<IProductService, ProductService>();
```

Kliensoldali regisztráció: Itt az interfészhez az API hívásokat kezelő kliens osztályt rendeljük hozzá.

```
1 builder.Services.AddScoped<IProductService, ClientProductService>();
```

2.4.2.3. Implementáció a két oldalon

A két projektben eltérő módon valósítjuk meg ugyanazt az interfészt:

Szerveroldal (ShopScout): A **ProductService** szerveroldali implementációja közvetlen kapcsolatban áll az adatbázis-réteggel az **Entity Framework Core** (EF Core) segítségével. Mivel ez a szolgáltatás a **ShopScout** projektben (szerveroldalon) fut, teljes hozzáférése van az adatbázis-kontextushoz, így nincs szüksége köztes hálózati hívásokra az adatok eléréséhez.

```
1 namespace ShopScout.Services;
2
3 public class ProductService : IProductService
4 {
5     public async Task<Product?> GetProductAsync(string barcode)
6     {
7         // get product logic
8     }
9 }
```

Kliensoldal (ShopScout.Client): A **ClientProductService** a ShopScout.Client projektben helyezkedik el, és feladata a szerveroldali üzleti logika elérése a böngészőből. Mivel a WebAssembly (WASM) környezet biztonsági okokból nem létesíthet közvetlen kapcsolatot az adatbázissal, a szolgáltatás a szabványos **HTTP protokollon** keresztül, JSON formátumú adatcserével kommunikál a szerver API végpontjaival.

```
1 namespace ShopScout.SharedLib.Services;
2
3 public class ClientProductService : IProductService
4 {
5     private readonly HttpClient _httpClient;
6     public ClientProductService(HttpClient httpClient)
7     {
8         _httpClient = httpClient;
9     }
10
11    public async Task<List<Product>> GetAllProductsAsync(int page)
12    {
13        var result = await _httpClient.GetFromJsonAsync<List<Product>>($""/api/product/{page}");
14        return result;
15    }
16 }
```

A ClientProductService felépítése

A szolgáltatás a konstruktőrben megkapja a **HttpClient** egy példányát, amelyet a kérések összeállításához és elküldéséhez használ. minden metódus, amely az **IProductService** interfészben szerepel, itt egy-egy API hívássá alakul.

Főbb megvalósítási jellemzők

JSON szerializáció: A GetFromJsonAsync és PostAsJsonAsync metódusok automatikusan kezelik a C# objektumok és a JSON formátum közötti átalakítást.

Relatív útvonalak: A HttpClient a regisztrációkor kapott alapértelmezett címet (Base Address) használja, így a kérésekben elegendő a relatív API útvonalakat (pl. api/product/...) megadni.

Végpontok megfeleltetése: minden itt indított hívás mögött állnia kell egy megfelelő kontrollernek a szerveroldali ShopScout projektben, amely fogadja a kérést, és meghívja a szerveroldali ProductService-t az adatok lekéréséhez.

Ez a rétegzett felépítés biztosítja, hogy a felhasználói felület (UI) számára transzparens maradjon az adatlekérés módja: a komponens csak meghívja az IProductService metódusát, és nem kell tudnia arról, hogy az adatbázisból vagy egy távoli API-ból érkezik-e a válasz.

2.4.3. Api dokumentáció

2.4.3.1. StoreLayoutController

A StoreLayoutController végpontjai az üzletek fizikai struktúrájának és a polcrendszereknek a kezeléséért felelnek. Az alábbiakban részletezzük az egyes végpontok feladatait, a várt adatokat és a visszatérési értékeket.

Üzlet alaprajzának lekérése (GET)

```
1 [HttpGet("{id}")]
2 public async Task<ActionResult> GetStoreLayout(int id)
3 {
4     var layout = await _storeLayoutService.GetShopLayout(id);
5     if (layout == null)
6     {
7         return NotFound();
8     }
9     return Ok(layout);
10 }
```

Az **api/StoreLayout/{id}** végponton keresztül az adott azonosítóval rendelkező üzlet teljes alaprajzi adatait kérhetjük le. A hívás során a rendszer ellenőrzi az adatbázisban a megadott azonosítót. Amennyiben az üzlet létezik, a válasz egy 200-as OK kóddal és a layout objektummal tér vissza. Ha az azonosító nem található, a végpont 404-es (Not Found) hibaüzenetet küld.

Üzlet alaprajzának mentése (POST)

```
1 [HttpPost("{id}")]
2 public async Task<Store> StoreLayout([FromBody] LayoutDto layout, int id)
3 {
4     return await _storeLayoutService.StoreWalls(layout, id);
5 }
```

Az **api/StoreLayout/{id}** végpont szolgál az üzlet falainak és alapvető geometriai struktúrájának rögzítésére vagy felülírására. A kérés törzsében (Body) egy LayoutDto típusú objektumot kell küldeni, amely tartalmazza a falak koordinátáit és típusait. A sikeres mentést követően a rendszer a módosított Store entitást adja vissza válaszként.

Polc adatainak frissítése (PUT)

```
1 [HttpPut("{id}/shelf")]
2 public async Task<ActionResult<Shelf>> UpdateShelf([FromBody] ShelfDto shelf, int id)
3 {
4     try
5     {
6         return Ok(await _storeLayoutService.UpdateShelf(shelf, id));
7     }
8     catch (Exception ex)
9     {
10        return BadRequest(ex.Message);
11    }
12 }
```

Az **api/StoreLayout/{id}/shelf** végponton keresztül egy adott üzlethez tartozó polc adatait lehet módosítani. A bemeneti adat egy ShelfDto objektum, amely a polc fizikai paramétereit (például pozíció, méret) írja le. Sikeres frissítés esetén a válasz a módosított Shelf objektum, hiba vagy érvénytelen adatok esetén pedig 400-as (Bad Request) válasz érkezik a hiba pontos leírásával.

Termék polchoz rendelése (POST)

```
1 public record ProductToShelfResponse(Store Store, ShelfDto Shelf);
2 public record AddProductData(ProductPerStoreDto Pps, float D);
3
4 [HttpPost("shelf/{shelfId}/product")]
5 public async Task<ActionResult<ProductToShelfResponse>> AddProductToShelf(
6     [FromBody] AddProductData productData,
7     int shelfId)
8 {
9     try
10    {
11        var ppsEntity = productData.Pps.ToEntity();
12        var updatedShelf = await _storeLayoutService.AddProductToShelf(ppsEntity,
13            shelfId, productData.D);
14        return Ok(new ProductToShelfResponse(updatedShelf.store,
15            updatedShelf.shelf));
16    }
17    catch (Exception ex)
18    {
19        return BadRequest(ex.Message);
20    }
21 }
```

Az **api/StoreLayout/shelf/{shelfId}/product** végpont felel azért, hogy egy adott terméket fizikailag is elhelyezzen a polcrendszerben. A kérés törzsében az AddProductData rekordot várja a rendszer, amely tartalmazza a termék üzletspecifikus adatait és a polcon belüli relatív pozíóját. A művelet

eredményeként egy ProductToShelfResponse objektum érkezik vissza, amely tartalmazza a frissített üzletadatokat és a módosult polc adatait is.

Termék eltávolítása a polcról (POST)

```
1 [HttpPost("shelf/{shelfId}/product/remove")]
2 public async Task<ActionResult<ProductToShelfResponse>>
    RemoveProductFromShelf([FromBody] ProductPerStoreDto productPerStore, int shelfId)
3 {
4     try
5     {
6         var ppsEntity = productPerStore.ToEntity();
7         var updatedShelf = await
        _storeLayoutService.RemoveProductFromShelf(ppsEntity, shelfId);
8         return Ok(new ProductToShelfResponse(updatedShelf.store,
    updatedShelf.shelf));
9     }
10    catch (Exception ex)
11    {
12        return BadRequest(ex.Message);
13    }
14 }
```

Az **api/StoreLayout/shelf/{shelfId}/product/remove** végpont segítségével törölhető egy termék és a polc közötti fizikai kapcsolat. A kéréshez egy ProductPerStoreDto objektum szükséges. A rendszer a sikeres eltávolítás után szintén egy ProductToShelfResponse rekordot küld vissza, amely tükrözi a polc tartalmának aktuális, frissített állapotát.

Polc adatainak lekérése (GET)

```
1 [HttpGet("shelf/{shelfId}")]
2 public async Task<ActionResult<Shelf>> GetShelf(int shelfId)
3 {
4     try
5     {
6         var shelf = await _storeLayoutService.GetShelfAsync(shelfId);
7         if (shelf == null)
8             return NotFound();
9         return Ok(shelf);
10    }
11    catch (Exception ex)
12    {
13        return BadRequest(ex.Message);
14    }
15 }
```

Az **api/StoreLayout/shelf/{shelfId}** végpont egy konkrét polc részletes információinak lekérésére szolgál. Ez a lekérdezés visszaadja a polc paramétereit és a rajta elhelyezett termékek listáját is.

2.4.3.2. ProductController

A **ProductController** felelős a termékkatalogus eléréséért, a keresési funkciókért és az összetett szűrési feltételek kiszolgálásáért. A kontroller közvetlenül az **IProductService** interfészét használja az adatok lekéréséhez, biztosítva a konzisztens adatelérést a teljes rendszerben.

Termék lekérése vonalkód alapján (GET)

```
1 [HttpGet("{barcode}")]
2 public async Task<ActionResult<Product>> GetProductByBarcode(string barcode)
3 {
4     var product = await _productService.GetProductAsync(barcode);
5     return Ok(product);
6 }
```

Az **api/Product/{barcode}** végponton keresztül egy konkrét termék adatait kérhetjük le a hozzá tartozó egyedi vonalkód segítségével. A rendszer a megadott karakterláncot alapján keres az adatbázisban, és a megtalált Product objektummal tér vissza. Ez a funkció különösen hasznos a mobilalkalmazásban történő vonalkód-leolvasás során.

Összes termék listázása lapozással (GET)

```
1 [HttpGet("{page:int}")]
2 public async Task<ActionResult<List<Product>>> GetAllProducts(int page = 1)
3 {
4     var product = await _productService.GetAllProductsAsync(page);
5     return Ok(product);
6 }
```

Az **api/Product/{page}** végpont lehetővé teszi a rendszerben tárolt összes termék lekérdezését. A nagy adatmennyiség miatt a válasz lapozott formában érkezik: a **page** paraméter határozza meg, hogy melyik adatszeletet kívánjuk megjeleníteni. Amennyiben a paraméter nincs megadva, a rendszer alapértelmezés szerint az első oldalt adja vissza.

Egyszerű keresés indítása (GET)

```
1 [HttpGet("search/{search_term}/page/{page:int}")]
2 public async Task<ActionResult<List<Product>>> Search(string search_term, int page = 1)
3 {
4     var product = await _productService.GetProductsSearchAsync(search_term, page);
5     return Ok(product);
6 }
```

Az **api/Product/search/{search_term}/page/{page}** végpont egy szabadszavas keresési funkciót valósít meg. A felhasználó által megadott search_term (keresési kifejezés) alapján a rendszer a termékek neveiben és leírásaiban keres. A találati lista szintén lapozható, így biztosítva a gyors betöltést és a stabil hálózati forgalmat.

Szűrt keresés kifejezés nélkül (POST)

```
1 [HttpPost("filter/page/{page:int}")]
2 public async Task<ActionResult<List<Product>>> FilteredSearch(int page, [FromBody]
3     ProductFilterParams filters)
4 {
5     var result = await _productService.GetProductsFilteredAsync(null, filters, page);
6     return Ok(result);
}
```

Az **api/Product/filter/page/{page}** végpont akkor használatos, amikor a felhasználó nem ír be keresőszót, de specifikus szűrési feltételeket (például árkategória, márka vagy kategória) állít be. A szűrési paramétereket a kérés törzsében (Body) egy ProductFilterParams objektumban kell elküldeni. A rendszer ezen paraméterek alapján szűkíti le a termékkínálatot.

Szűrt keresés kifejezéssel kombinálva (POST)

```
1 [HttpPost("filter/{search_term}/page/{page:int}")]
2 public async Task<ActionResult<List<Product>>> FilteredSearchWithTerm(string search_term,
3     int page, [FromBody] ProductFilterParams filters)
4 {
5     var result = await _productService.GetProductsFilteredAsync(search_term, filters, page);
6     return Ok(result);
}
```

Az **api/Product/filter/{search_term}/page/{page}** végpont a legösszetettebb keresési mód. Lehetővé teszi, hogy a megadott keresőszót és a részletes szűrési feltételeket egyszerre alkalmazzuk. A rendszer először a kifejezésre keres rá, majd a kapott találatokat tovább szűri a ProductFilterParams objektumban megadott értékek alapján.

Alkalmazott adatstruktúrák és DTO-k

A keresési és szűrési folyamatok során az alábbi kulcsfontosságú elemet használjuk:

ProductFilterParams: Ez a DTO tartalmazza az összes olyan választható szempontot, amely alapján a felhasználó szűkítheti a találatokat. Ide tartozhatnak a kategória-azonosítók, minimum és maximum árak, vagy egyéb termékjellemzők. Azért alkalmazunk POST metódust a szűrésnél, mert az összetett szűrési objektum túl hosszú és bonyolult lenne ahhoz, hogy URL paraméterként (Query string) adjuk át.

Ez a felépítés támogatja a nagy teljesítményű keresést, miközben rugalmas marad a felhasználói igényekkel szemben.

2.4.4. Függőségek, alkalmazás beállítások

A ShopScout alkalmazás stabilitásának és skálázhatóságának alapköve a **Program.cs** fájlban definiált szolgáltatásregisztráció. A modern .NET konvenciókat követve a rendszer a **Dependency Injection (DI)**, azaz a függőségi injektálás elvére épül. Ez a tervezési minta lehetővé teszi, hogy az alkalmazás egyes komponensei (mint például az adatbázis-kezelő vagy a hitelesítési szolgáltatás) lazán csatoltak, könnyen tesztelhetőek és modulárisan cserélhetőek legyenek.

2.4.4.1. Naplózás (Serilog)

```
1 var logTemplate = "{Timestamp:yyyy-MM-dd HH:mm:ss.fff zzz} [{Level:u3}] [{SourceContext}]
  {Message:lj}{NewLine}{Exception}";
2
3 Log.Logger = new LoggerConfiguration()
4     .MinimumLevel.Information()
5     .WriteTo.Logger(lc => lc
6         .Filter.ByIncludingOnly(Matching.FromSource("Microsoft.EntityFrameworkCore"))
7         .WriteTo.File("logs/ef-core-.txt",
8             rollingInterval: RollingInterval.Day,
9             shared: true,
10            fileSizeLimitBytes: 10 * 1024 * 1024, // 10 MB limit per file
11            outputTemplate: logTemplate,
12            rollOnFileSizeLimit: true,
13            retainedFileCountLimit: 7))
14
15     .WriteTo.Logger(lc => lc
16         .Filter.ByIncludingOnly(e => e.Level == LogEventLevel.Error)
17         .WriteTo.File("logs/errors-.txt",
18             rollingInterval: RollingInterval.Day,
19             shared: true,
20             outputTemplate: logTemplate))
21
22     .WriteTo.Logger(lc => lc
23         .Filter.ByIncludingOnly(e => e.Level == LogEventLevel.Fatal)
24         .WriteTo.File("logs/fatals-.txt",
25             rollingInterval: RollingInterval.Day,
26             shared: true,
27             outputTemplate: logTemplate))
28
29     .WriteTo.Logger(lc => lc
30         .Filter.ByExcluding(Matching.FromSource("Microsoft.EntityFrameworkCore"))
31         .Filter.ByIncludingOnly(e => e.Level == LogEventLevel.Information || e.Level ==
32             LogEventLevel.Warning)
33         .WriteTo.File("logs/app-.txt",
34             rollingInterval: RollingInterval.Day,
35             shared: true,
36             outputTemplate: logTemplate))
37
38     .WriteTo.Console(
39         restrictedToMinimumLevel: LogEventLevel.Error,
40         outputTemplate: logTemplate)
41
42 builder.Host.UseSerilog();
```

A rendszer egy robusztus, **Serilog** alapú naplózást használ, amely nem csak a konzolra ír, hanem különböző fájlokba szűri az eseményeket a fontosságuk és forrásuk alapján. Az összes naplót az admin felületből lehet elérni.

- **EF Core logok:** Külön fájlba (ef-core-.txt) kerülnek az adatbázis-műveletek.
- **Hiba- és kritikus naplók:** Az Error és Fatal szintű események külön dedikált fájlokat kaptak a gyorsabb hibakeresés érdekében.
- **Alkalmazás logok:** A minden nap működés információi az app-.txt-be kerülnek.

2.4.4.2. Üzleti logika és Szolgáltatások (Scoped Services)

```
1 builder.Services.AddScoped<IdentityUserAccessor>();
2 builder.Services.AddScoped<IdentityRedirectManager>();
3 builder.Services.AddScoped<AuthenticationStateProvider,
    IdentityRevalidatingAuthenticationStateProvider>();
4 builder.Services.AddScoped<LogService>();
5 builder.Services.AddScoped<IProductService, ProductService>();
6 builder.Services.AddScoped<IStoreService, StoreService>();
7 builder.Services.AddScoped<IStoreLayoutService, StoreLayoutService>();
8 builder.Services.AddScoped<IAdminService, AdminService>();
9 builder.Services.AddScoped<IIImageStorageService, GoogleCloudImageStorage>();
10 builder.Services.AddScoped<IUserAccessor, UserAccessor>();
11 builder.Services.AddScoped<ICategoryService, CategoryService>();
12 builder.Services.AddScoped<ServerCookieService>();
13 builder.Services.AddScoped<StorageService>();
14 builder.Services.AddScoped<LocationService>();
15 builder.Services.AddScoped<UserService>();
16 builder.Services.AddScoped<AccountNavbarService>();
17 builder.Services.AddScoped<ArfigyeloFetchService>();
```

A fenti kódrészlet az alkalmazás üzleti logikáját és alapvető funkcióit kezelő szolgáltatások regisztrációját tartalmazza. Ezek a szolgáltatások **Scoped** élettartammal rendelkeznek, ami azt jelenti, hogy minden egyes felhasználói munkamenet (vagy HTTP kérés) saját, elkülönített példányt kap belőlük.

A szolgáltatásokat feladatkörük szerint az alábbi csoportokra oszthatjuk:

1. Felhasználókezelés és biztonság

- **IdentityUserAccessor** és **IdentityRedirectManager**: A bejelentkezett felhasználók adatainak eléréséért, valamint a hitelesítés során történő átirányításokért felelnek.

- **AuthenticationStateProvider**: Biztosítja, hogy az alkalmazás minden pontos információval rendelkezzen a felhasználó aktuális bejelentkezési állapotáról.
- **UserAccessor és UserService**: Általánosabb szinten kezelik a felhasználói profilokat és a jogosultságokat.

2. Alapvető üzleti funkciók

- **IProductService és ICategoryService**: A termékek és kategóriák kezelését (lekérdezés, mentés) végzik. Az interfészek használata lehetővé teszi, hogy a háttérben lévő adatbázis-logika könnyen módosítható legyen.
- **IStoreService**: Az üzletek adataival kapcsolatos műveletekért felelős.

3. Speciális és kiegészítő szolgáltatások

- **IStoreLayoutService**: Az üzleteken belüli elrendezések (térképek, polcok helyzete) logikáját kezeli.
- **ImageStorageService (GoogleCloudImageStorage)**: A termékekről vagy üzletekről készült képek feltöltését és tárolását végzi a Google Cloud felhőalapú rendszerében.
- **ArfigyeloFetchService**: A ShopScout egyik kiemelt funkciója, amely az árak automatikus figyeléséért és frissítéséért felelős.
- **IAdminService**: Kifejezetten az adminisztrátori felülethez kapcsolódó, emelt szintű műveleteket hajtja végre.

4. Támogató modulok

- **LocationService**: A földrajzi helymeghatározáshoz és távolságméréshez szükséges számításokat végzi.
- **ServerCookieService és StorageService**: Az adatok ideiglenes tárolását és a szerveroldali sütik kezelését oldják meg.
- **AccountNavbarService**: A felhasználói felület fejlécének (navigációs sáv) dinamikus megjelenítéséért felel.

2.4.4.3. Rendszerszintű biztonsági és kommunikációs beállítások

A ShopScout alkalmazás biztonsági alapjait és alapvető kommunikációs csatornáit az alábbi három konfigurációs egység határozza meg. Ezek a beállítások biztosítják a rendszer skálázhatóságát, a felhasználói munkamenetek védelmét és a külső szolgáltatásokkal való kapcsolatot.

Adatvédelem (Data Protection)

```
1 builder.Services.AddDataProtection()
2     .PersistKeysToDbContext<ApplicationContext>()
3     .SetApplicationName("ShopScout");
```

A .NET keretrendszer számos adatot (például hitelesítési tokeneket) automatikusan titkosít. Alapértelmezés szerint a titkosításhoz használt kulcsokat a rendszer a memóriában tárolja, ami felhőalapú környezetben vagy több kiszolgáló használata esetén a munkamenetek elvesztéséhez vezethetne.

- **PersistKeysToDbContext:** A titkosítási kulcsok nem a lokális fájlok közé, hanem az SQL adatbázisba kerülnek. Ez garantálja, hogy a szerver újraindulása vagy bővítése után is érvényesek maradnak a korábban kiadott titkosított adatok.
- **SetApplicationName:** Egyedi azonosítót adunk az alkalmazásnak a titkosítási motoron belül. Ez biztosítja az adatok elkülönítését, így más alkalmazások nem férhetnek hozzá a ShopScout által titkosított információkhoz.

Hitelesítési folyamatok és süti-kezelés

```
1 builder.Services.ConfigureApplicationCookie(options =>
2 {
3     options.Cookie.Name = "AuthCookie";
4     options.ExpireTimeSpan = TimeSpan.FromDays(30);
5     options SlidingExpiration = true;
6     options.Cookie.HttpOnly = true;
7     options.Cookie.IsEssential = true;
8});
```

A felhasználók bejelentkezési állapotát szabályozó paraméterek a biztonság és a kényelem közötti egyensúlyt szolgálják:

- **Cookie.Name:** Az alapértelmezett technikai név helyett a projekt egyedi azonosítót használ a hitelesítési sütihez, ami segít az alkalmazás azonosításában.
- **ExpireTimeSpan:** A beállítás 30 napos érvényességi időt határoz meg, így a felhasználóknak ritkábban kell újra bejelentkezniük.
- **SlidingExpiration:** Aktív használat esetén a lejáratú idő automatikusan meghosszabbodik. Ez biztosítja, hogy a folyamatosan tevékenykedő felhasználók munkamenete ne szakadjon meg váratlanul.
- **HttpOnly:** Fontos biztonsági korlátozás, amely megakadályozza, hogy a kliensoldali szkriptek hozzáférjenek a süti tartalmához. Ez hatékony védelmet nyújt az XSS (Cross-Site Scripting) típusú támadások ellen.
- **IsEssential:** Ez a jelzés közli a böngészővel, hogy a süti a rendszer alapvető működéséhez elengedhetetlen. Emiatt a szigorúbb adatvédelmi szűrések vagy korlátozások mellett is létrejön a kapcsolat a belépéshez.

E-mail szolgáltatás regisztrációja

```
1 builder.Services.AddSingleton<ShopScout.Services.IEmailSender, EmailSender>();
```

A felhasználókkal való kapcsolattartást (pl. jelszóemlékeztetők, visszaigazolások) egy absztrakciós rétegen keresztül kezeljük.

- **AddSingleton:** Mivel az e-mail küldő szolgáltatás nem tárol felhasználóspecifikus adatokat, az alkalmazás teljes élettartama alatt egyetlen példány szolgálja ki az összes kérést, ami erőforrás-hatékony megoldás.
- **IEmailSender interfész:** A konkrét megvalósítás (EmailSender) egy interfész mögé van rejtve. Ez lehetővé teszi, hogy a későbbi fejlesztés során a kód többi részének módosítása nélkül cserélhető legyen.

2.5. A projekt tesztelése

A szoftverfejlesztési életciklus egyik legkritikusabb szakasza a minőségbiztosítás, amely garantálja, hogy a leszállított vizsgaremek megfelel a funkcionális követelményeknek és a technológiai elvárásoknak. A tesztelési folyamat célja a szoftverhibák azonosítása, a rendszerműködés stabilitásának verifikálása, valamint a felhasználói élmény optimalizálása.

2.5.1. Bevezetés és tesztelési stratégia

A projekt fejlesztése során a tesztelés nem egy lezáró szakaszként, hanem a kódolással párhuzamosan végzett tevékenysékként jelent meg. A stratégia alapját a többszintű ellenőrzés adta, amely a forráskód automatizált vizsgálatától a manuális felhasználói próbáig terjedt.

2.5.1.1. A készre jelentés feltételei (Definition of Done)

A szoftvert akkor tekintettem teszteltnek és a vizsgabizottság számára bemutathatónak, amikor teljesültek az alábbi kritériumok:

- **Funkcionális megfelelőség:** minden, a specifikációban rögzített user story (felhasználói igény) megvalósult és hiba nélkül működik.
- **Kritikus hibák mentessége:** Nincs olyan ismert hiba, amely az alkalmazás összeomlásához vagy adatvesztéshez vezetne.
- **Válaszidő:** A Blazor WebAssembly kliens és az ASP.NET Core API közötti kommunikáció késleltetése elfogadható szinten marad.
- **Stabilitás:** A szoftver különböző böngészőkben és felbontások mellett is konzisztens működést mutat.

2.5.1.2. Alkalmazott tesztelési módszertanok

A minőségbiztosítás során az alábbi szakmai módszereket és eszközöket alkalmaztuk:

1. **Statikus kódelemzés (Static Code Analysis):** A fejlesztés során a Visual Studio beépített analizátorait és a .NET keretrendszer kódstílus-ellenőrzőit

használtuk. Ez lehetővé tette a potenciális logikai hibák, a nem használt változók és a szintaktikai pontatlanságok kiszűrését még a futtatás előtt.

2. **Egységesztelés (Unit Testing):** A szoftver üzleti logikáját (különösen a Shared és Server projektekben található algoritmusokat) egységesztekkel ellenőriztem. Ezek a tesztek izoláltan vizsgálják az egyes metódusok működését, garantálva, hogy a későbbi módosítások nem rontják el a már meglévő funkciókat (regressziós védelem).
3. **Hibakeresés és diagnosztika:** A fejlesztési fázisban a keretrendszer **Debug** módját használtam, amely lehetővé tette a töréspontok (breakpoints) elhelyezését és a változók aktuális értékének vizsgálatát futás közben. A szerveroldali folyamatokat strukturált naplózással követtem nyomon, így az API kérések és az adatbázis-műveletek státusza (pl. HTTP 200 OK vagy 500 Internal Server Error) visszakereshetővé vált.
4. **Felderítő tesztelés (Exploratory Testing):** A manuális tesztelés során nem csak a tervezett útvonalakat, hanem a váratlan felhasználói interakciókat is vizsgáltam. Ez a módszer segített feltárni azokat a peremeseteket, mint például az üres keresési feltételek vagy a speciális karakterek bevitel a szerkesztőfelületen.

2.5.1.3. Egységesztek (Unit Tests)

Az alkalmazás üzleti logikájának ellenőrzése automatizált egységesztek segítségével történt. A tesztkönyezet az **xUnit** keretrendszerre és a **Moq** könyvtárra épül, amely lehetővé teszi a függőségek (például az adatbázis-elérés vagy az Identity-kezelő) izolálását. A teszesetek minden esetben az **Arrange-Act-Assert** (Előkészítés-Végrehajtás-Ellenőrzés) mintát követik.

Felhasználói szolgáltatások ellenőrzése (UserService)

A UserService tesztelése során az automatikus felhasználónév-generáló algoritmus működése állt a középpontban. A vizsgálat célja annak igazolása, hogy a rendszer képes az e-mail címek alapján konzisztens és egyedi azonosítókat létrehozni.

A vizsgált logikai pontok:

- **Névképzés:** Az e-mail cím előtagjának kinyerése és tisztítása.
- **Karakterfeltöltés (Padding):** Rövid e-mail címek esetén a minimális hosszt biztosító automatikus kiegészítés ellenőrzése.
- **Ütközéskezelés:** Annak igazolása, hogy már létező felhasználónév esetén a rendszer numerikus utótággal garantálja az egyediséget.

```

1 [Fact]
2 public async Task SetUniqueUserName_WhenUsernameExists_AppendsNumericSuffix()
3 {
4     // Arrange
5     var user = new ApplicationUser { Email = "testuser@example.com" };
6     var existingUser = new ApplicationUser { UserName = "testuser" };
7
8     _userManagerMock.SetupSequence(x => x.FindByNameAsync(It.IsAny<string>()))
9         .ReturnsAsync(existingUser) // First call returns existing user
10        .ReturnsAsync(( ApplicationUser? )null); // Second call returns null
11
12    // Act
13    await _userService.SetUniqueUserName(user);
14
15    // Assert
16    Assert.Equal("testuser1", user.UserName);
17 }

```

```

1 [Fact]
2 public async Task
3     SetUniqueUserName_WhenMultipleUsernamesExist_IncrementsUntilUnique()
4 {
5     // Arrange
6     var user = new ApplicationUser { Email = "testuser@example.com" };
7     var existingUser1 = new ApplicationUser { UserName = "testuser" };
8     var existingUser2 = new ApplicationUser { UserName = "testuser1" };
9     var existingUser3 = new ApplicationUser { UserName = "testuser2" };
10
11    _userManagerMock.SetupSequence(x => x.FindByNameAsync(It.IsAny<string>()))
12        .ReturnsAsync(existingUser1)
13        .ReturnsAsync(existingUser2)
14        .ReturnsAsync(existingUser3)
15        .ReturnsAsync(( ApplicationUser? )null);
16
17    // Act
18    await _userService.SetUniqueUserName(user);
19
20    // Assert
21    Assert.Equal("testuser3", user.UserName);
22 }

```

Adatkezelési segédfunkciók ellenőrzése (ProductService)

A ProductService tesztjei az adatformázási és objektum-hozzárendelési folyamatok stabilitását vizsgálják. Ezek a metódusok felelnek a nyers bemeneti

adatok (például UI-ról érkező karakterláncok) feldolgozásáért és az adatbázis-kapcsolatok (N:N táblák) előkészítéséért.

A vizsgált logikai pontok:

- **Szövegformázás:** A speciális karakterek (kötőjelek, alulvonások) szóközre cserélése és a prefixek eltávolítása.
- **Gyűjteménykezelés:** Adatok hozzáadása meglévő listákhoz és az üres (null) bemenetek hibamentes kezelése.
- **Stabilitás:** A feldolgozás megszakítása és hibatűrése váratlan bemeneti formátumok esetén.

```
1 [Fact]
2 public void AttachToCollection_WithValidStrings_AddsFormattedItems()
3 {
4     // Arrange
5     var inputCollection = new List<string>
6     {
7         "prefix:item-name",
8         "another_item_name",
9         "simple"
10    };
11    var attachingCollection = new List<TestNToNTTable>();
12
13    // Act
14    ProductService.AttachToCollection(inputCollection, attachingCollection);
15
16    // Assert
17    Assert.Equal(3, attachingCollection.Count);
18    Assert.Equal("item name", attachingCollection[0].Name);
19    Assert.Equal("another item name", attachingCollection[1].Name);
20    Assert.Equal("simple", attachingCollection[2].Name);
21 }
```

```
1 [Fact]
2 public void AttachToCollection_AppendsToExistingCollection()
3 {
4     // Arrange
5     var inputCollection = new List<string> { "new-item" };
6     var attachingCollection = new List<TestNToNTTable>
7     {
8         new TestNToNTTable { Name = "existing" }
9     };
10
11    // Act
12    ProductService.AttachToCollection(inputCollection, attachingCollection);
13
14    // Assert
15    Assert.Equal(2, attachingCollection.Count);
16    Assert.Equal("existing", attachingCollection[0].Name);
17    Assert.Equal("new item", attachingCollection[1].Name);
18 }
```

Külső adatforrások és API integráció tesztelése (ArfigyeloFetchService)

Az ArfigyeloFetchService felelős a külső árfigyelő rendszerből származó adatok lekéréséért, értelmezéséért és az adatbázis frissítéséért. A tesztek célja a komplex JSON struktúrák értelmezésének validálása és az üzleti logika (például az árváltozások detektálása) helyességének ellenőrzése.

A vizsgált funkciók és tesztesetek:

- Kategória-leképezés (MapCategoriesTolds):** A tesztek ellenőrzik a többszintű, egymásba ágyazott JSON kategóriafa rekurzív feldolgozását. A vizsgálat kiterjed a kis- és nagybetűk közötti különbségtétel kezelésére is, biztosítva a robusztus keresést.
- Azonosító kinyerése (GetIdsFromJsonString):** A szoftver egy alacsony szintű JSON-olvasót használ a termékazonosítók kinyerésére. A teszteset igazolja, hogy az algoritmus a megfelelő mélységen (depth) találja meg az adatokat a JSON-struktúrán belül.
- Adatszinkronizáció (UpdateEntitiesFromApiData):** Ez a tesztcsoport azt vizsgálja, hogy a rendszer helyesen azonosítja-e az árváltoztásokat. Csak akkor történik módosítás-számlálás és adatfrissítés, ha az API-ból érkező ár ténylegesen eltér a már tárolt értéktől.

```
1 [Fact]
2 public void GetIdsFromJsonString_ShouldExtractIdsAtCorrectDepth( )
3 {
4     var json = @"
5     [
6         {
7             ""wrapper"": {
8                 ""id"": ""product123"",
9                 ""name"": ""Test Product""
10            }
11        },
12        {
13            ""wrapper"": {
14                ""id"": ""product456""
15            }
16        }
17    ]"; // A kód 3-as mélységet vár (CurrentDepth == 3) a JSON readerben.
18     var result = ArfigyeloFetchService.GetIdsFromJsonString(json);
19     Assert.Equal(2, result.Count);
20     Assert.Contains("product123", result);
21     Assert.Contains("product456", result);
22 }
```

```

1 [Fact]
2 public void UpdateEntitiesFromApiData_ShouldUpdatePrices_WhenPricesChange()
3 {
4     // Arrange
5     var dbEntities = new List<ProductPerStore>
6     {
7         new ProductPerStore
8         {
9             Price = 1000,
10            DiscountedPrice = 900,
11            Product = new Product { Code = "PROD-1" },
12            Store = new Store { StoreBrand = new StoreBrand { Name = "Tesco" } }
13        }
14    };
15
16    var apiDataDict = new Dictionary<string, ArfigyeloFetchService.ApiProductResponse>
17    {
18        {
19            "PROD-1", new ArfigyeloFetchService.ApiProductResponse
20            {
21                Id = "PROD-1",
22                ChainStores = new List<ArfigyeloFetchService.ApiChainStore>
23                {
24                    new ArfigyeloFetchService.ApiChainStore
25                    {
26                        Name = "Tesco",
27                        Prices = new List<ArfigyeloFetchService.ApiPrice>
28                        {
29                            new ArfigyeloFetchService.ApiPrice { Type = "NORMAL", Amount = 1200 }, // Változott
30                            new ArfigyeloFetchService.ApiPrice { Type = "DISCOUNTED", Amount = 850 } // Változott
31                        }
32                    }
33                }
34            }
35        };
36    };
37
38    // Act
39    int changesCount = ArfigyeloFetchService.UpdateEntitiesFromApiData(dbEntities, apiDataDict);
40
41    // Assert
42    Assert.Equal(1, changesCount);
43    Assert.Equal(1200, dbEntities[0].Price);
44    Assert.Equal(850, dbEntities[0].DiscountedPrice);
45 }

```

3. Felhasználói dokumentáció

3.1. Telepítési útmutató

3.1.1. Webszerver

A szoftver architektúrája a Blazor WebAssembly ASP.NET modellen alapul. Ez a felépítés több fő részegységből áll: a kliensoldali (Client), a szerveroldali (Server) és a további projektből.

1. Először az alábbi linken található GitHub repository-t kell leklónozni:

<https://github.com/KGBSS/ShopScout-public>

2. Majd visual studio-n keresztül futtatni a "ShopScout" nevű projektet.

3.1.2. Rendszerkövetelmények

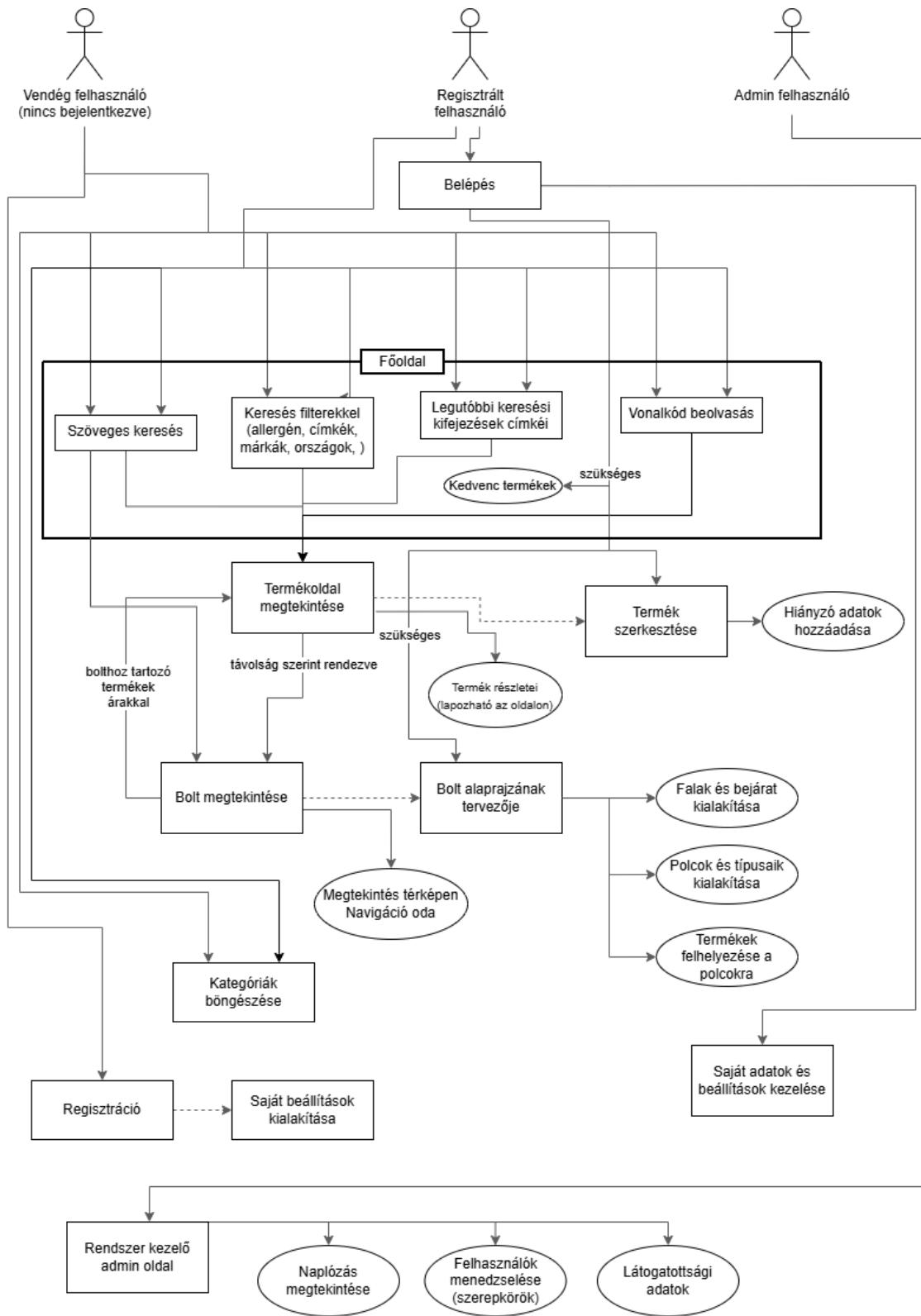
3.1.2.1. Szoftveres feltételek

Runtime (Futtatókörnyezet): .NET 9 SDK, ASP.NET kiegészítővel

Adatbázis: Local mssql database (A visual studio autómatikusan létrehozza)

Böngésző: Bármely modern böngésző megfelelő, ajánlott: Chrome, vagy Brave

3.2. Használati eset diagram



3.3. Weboldal felhasználói dokumentáció

3.3.1. Rendszerkövetelmények

3.3.1.1. Szoftveres feltételek

- **Támogatott böngészők:** A weboldal a modern böngészők legfrissebb verzióira van optimalizálva (pl. Google Chrome 90+, Mozilla Firefox 88+, Microsoft Edge 90+, Safari 14+).
- **Operációs rendszer:** Windows 10/11, macOS, Linux, Android vagy iOS (mivel a rendszer reszponzív).
- **Szükséges beállítások:** A böngészőben engedélyezni kell a **JavaScript** futtatását és a **Sütik (Cookies)** használatát a teljes funkcionalitás érdekében.

3.3.1.2. Hardveres feltételek

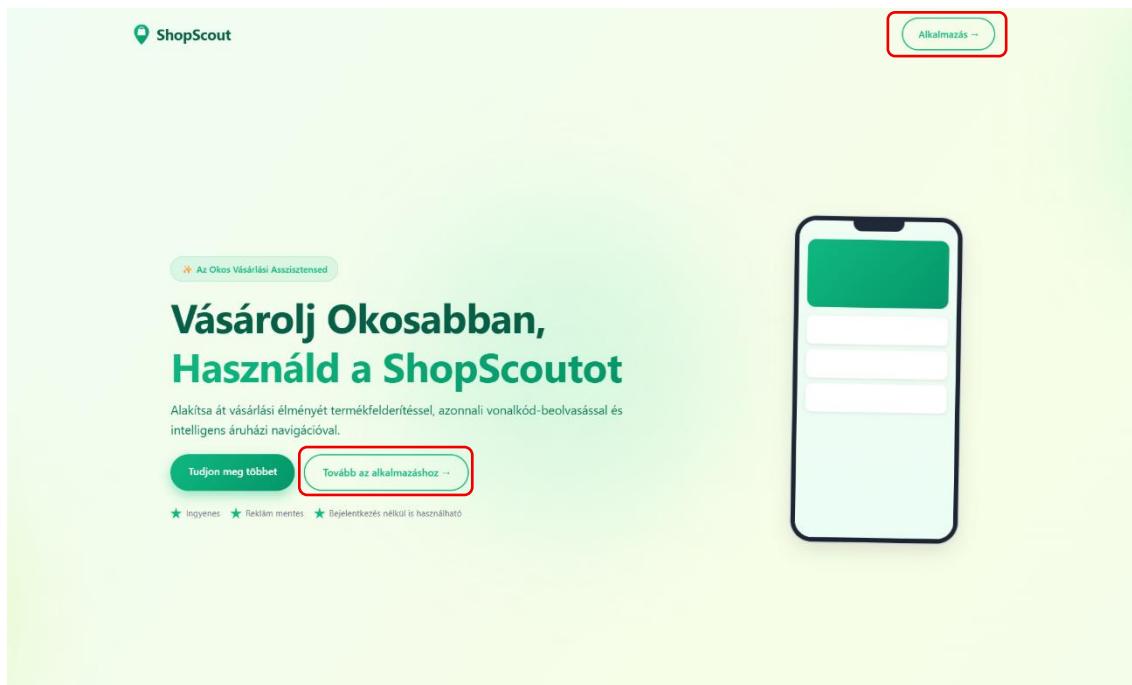
- **Processzor:** Bármilyen modern kétfogas processzor (1.6 GHz+).
- **Memória (RAM):** Minimum 2 GB RAM (ajánlott 4 GB a böngésző sima futtatásához).
- **Képernyőfelbontás:** Minimum 360x640 (mobil), ajánlott 1920x1080 (asztali monitor) a kényelmes használathoz.
- **Beviteli eszközök:** Billentyűzet és egér, vagy érintőképernyő.

3.3.1.3. Hálózati feltételek

- **Internetkapcsolat:** Folyamatos szélessávú internetkapcsolat szükséges.

3.4. Felhasználói felület (UI) bemutatása

3.4.1. Üdvözlő oldal (Landing page)

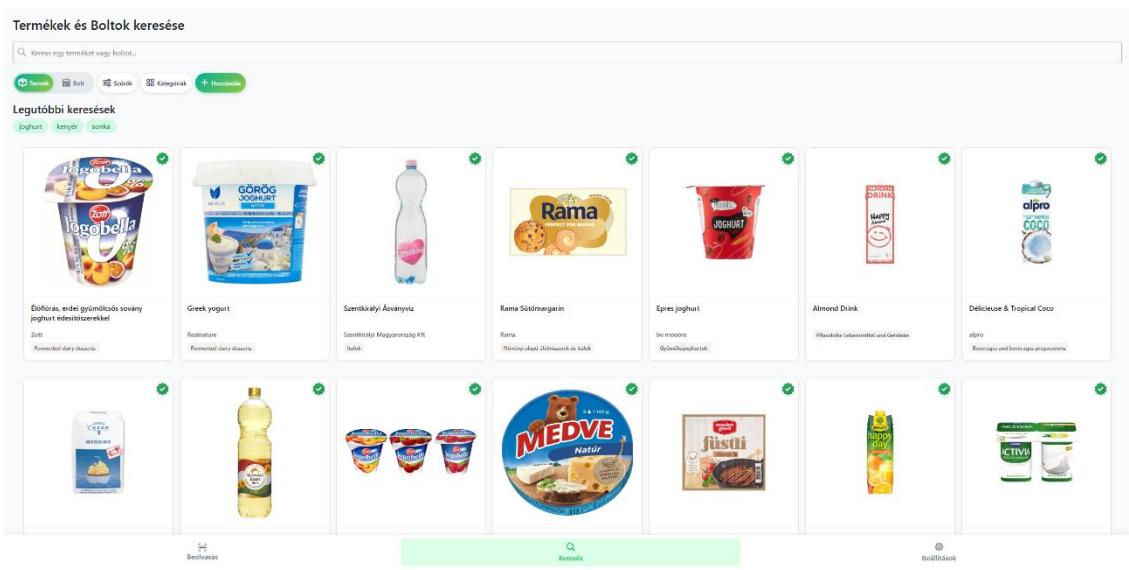


Az alkalmazás indítását követően a felhasználó az üdvözlő oldalra érkezik. A felület kialakítása a modern UI/UX irányelveket követi: letisztult, reszponzív és vizuálisan támogatja a gyors navigációt.

Az „Alkalmazás →” vagy a „Tovább az alkalmazáshoz →” gombra kattintva lehet a főoldalt elérni.



3.4.2. Főoldal



Ez a felület az alkalmazás funkcionális központja, ahol a felhasználók az adatok közötti navigációt és a konkrét termékkeresést végzik.

Megaláthatóak az oldalon:

- Globális keresősáv:** A felület legfelső részén elhelyezkedő szöveges beviteli mező, amely az adatbázisban történő kulcsszavas keresések alapja.
- Keresési módválasztó (Kétállású kapcsoló):** Közvetlenül a keresősáv alatt található választógombok, amelyekkel definiálható a keresés hatóköré (**Termék** vagy **Bolt**).
- Keresésfinomító eszközök:**
 - Szűrők :** Az eredményhalmaz szűkítésére szolgáló paraméterezési lehetőségek.
 - Kategóriák:** A termékcsoporthoz szerinti böngészést támogató aloldal elérésére szolgáló vezérlő.
- Keresési előzmények:** Dinamikusan frissülő kulcsszólista, amelyek a korábbi sikeres keresések gyors megismétlését teszik lehetővé.
- Dinamikus eredménylista:** Az aktuális keresési feltételeknek megfelelő termékek vagy boltok kártya-alapú megjelenítése, amely tartalmazza a képi és szöveges adatokat.

- **Alsó navigációs sáv:** Állandóan jelen lévő vezérlőmodul az alkalmazás fő funkciói (Beolvasás, Keresés, Beállítások) közötti váltáshoz.

3.4.2.1. Keresés



A keresni kívánt kifejezés beírása után egy töltő ikon fog megjelenni, ezzel jelezve, hogy éppen dolgozik az algoritmus.

A mező tartalmának gyors kitörlésére a jobb oldalon található „X” gomb szolgál.

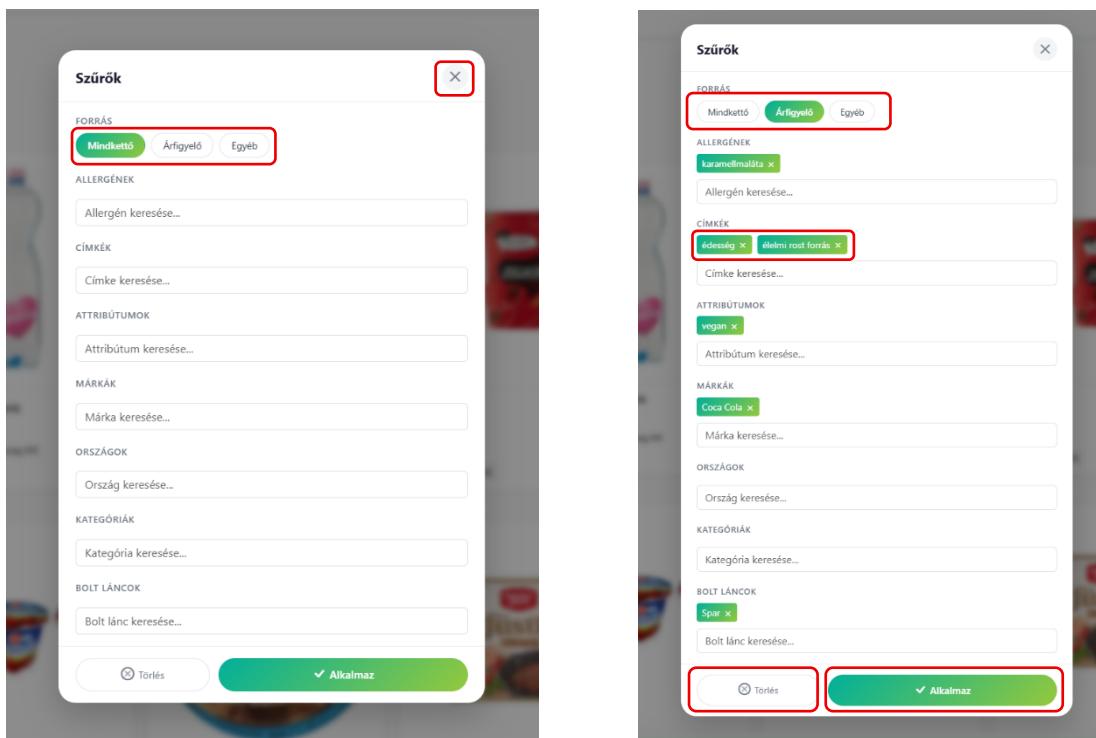
The screenshot shows a section titled "Legutóbbi keresések" (Recent searches). Below it are three circular buttons containing the words "sonka", "joghurt", and "kenyér".

Majd ha egy termékre is rákattintott a felhasználó, akkor előre kerül a legutóbbi kifejezés, ebben az esetben a „sonka”.



Ez a terméket jelző kártya, a termék nevével, márkJával és az egyik kategóriájával. A pipa a sarokban azt jelzi, hogy hivatalos forrásból származik, azaz az ezt bemutató oldalon fog a felhasználó találni olyan boltokat amik árulják ezt a terméket. Erre kattintva lehet eljutni a terméket részletesen bemutató aloldalra.

3.4.2.2. Szűrők



A pontos találati lista előállítása érdekében az alkalmazás egy komplex, modális ablakban megjelenő szűrési felületet biztosít. Ez a felület lehetővé teszi a keresési eredmények finomhangolását több attribútum mentén.

A legfelső lehetőséggel a hivatalos forrásokból származó termékeket tudjuk szűrni, amikhez biztosan bolt is fog tartozni. A másik lehetőséggel az egyéb feltöltött és az OpenFoodFacts közösségi adatbázisból származó termékeket lehet szűrni. Ezt a kettőt vegyíteni is lehet a „Mindkettő” opcióval.

A további keresési panelek a föléjük írt kategóriában keresik a rendszerben lévő opciókat és azok közül egy legördülő listából lehet kiválasztani a megfelelőt, ami utána a mező fölött meg is jelenik. Ezt az „x” jelzéssel törölni is lehet a szűrési feltételek közül.

Az ablakot a jobb felső sarokban található „x” gombbal lehet becsukni, ezzel megszüntetve a szűrők beállítását.

A kiválasztott szűrőket a „Törlés” gombba van lehetőség alaphelyzetbe állítani.

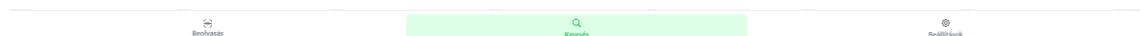
Az „Alkalmaz” gombbal a szűrők aktiválódnak és elindul a keresés ezek alapján.



A beállított szűrőket az ablak becsukása után is lehet látni, törölni vagy egyszerre az összeset is törölni.

A szűrők beállítása után is ugyan úgy lehet a keresőmezőt használni, és az eredmények minden egyik szűrőnek és a keresési kulcsszónak is meg fognak felelni.

3.4.3. Alsó menü elemei



Ez a menü a legtöbb oldalon elérhető az egyszerű hozzáférhetőség érdekében, ezzel erősítve a webapplikáció PWA támogatásán keresztül megvalósuló mobilapplikáció jellegét.

A középső menü a főoldalra vezet, ami az előbbiekben került bemutatásra.

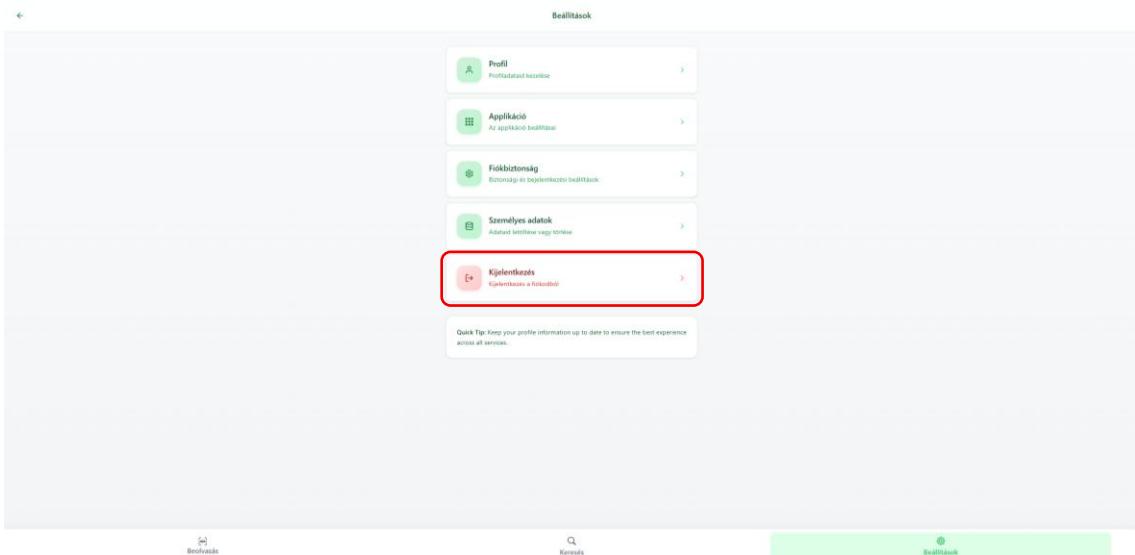
3.4.3.1. Vonalkód olvasó



Az eszköz kameráját megnyitva a rendszer felismeri az adott termék vonalkódját, amire az eszköz irányítva lett, majd megnyitja a hozzá tartozó adatlapot.

3.4.3.2. Beállítások (bejelentkezett felhasználók)

A még nem regisztrált felhasználók a bejelentkezési oldalra kerülnek átirányításra, de a már bejelentkezett felhasználók a beállításaiat láthatják.



Itt kerülnek részletezésre a beállítások fül menüpontjai amik az ábrán is láthatóak. A „Kijelentkezés” gombbal lehet az adott felhasználót kijelentkeztetni a rendszerből.

Profil

The screenshot shows the 'Profile' section of a user's account. It includes:

- A placeholder profile picture with a letter 'J'.
- An 'Edit Profile' button.
- Email input field: admin@admin.hu
- Username input field: admin
- A green gradient button labeled 'Élmény személyre szabása' (Customize experience).
- An 'Account Information' section:
 - Member Since: 2026. február 26., csütörtök
 - Email-cím: Nincs megerősítve (with a 'Visszaigazoló email küldése' button)
 - Last Login: 2026. február 27., péntek 23:38
- A note at the bottom: 'Privacy Notice: Your profile information is private and only visible to you.'

A felhasználó itt tudja a profiljának adatait megnézni és szerkeszteni.

Szerkesztés:

Email
admin@admin.hu

Username
admin

Testreszabás

Üdvözünk a ShopScout-ban!

Állítsa be a fiókod néhány egyszerű lépében, hogy a legtöbbet hozz ki az elérőnyöld.

→ Kezdjük el a beállítást

Nyelv és ország

Nyelv: Magyar

Ország: Magyarország

Tovább

Profil információk

Felhasználónév: admin

Másik ne lassák a nevem

Ira bejelentkezés minden tevékenységeidél az „anonymous” név jelent meg

Vissza Tovább

Kedvenc helyek

Ezek nem tesznek láthatóuk mások számára

Kedvenc városok

Keresz várost...

Kedvenc kategóriák

Keresz kategóriát...

Kedvenc boltok

Vissza Tovább

Értesítési beállítások

Válasszon ki, hogy milyen értesítéseket szeretné kapni.

Kedvenc termékek

Értesítés kedvenc termékek akciójáról

Push értesítés

Alkalmazás frissítések

Új funkciók és frissítések

Push értesítés

E-mail

Vissza Tovább

Engedélyek

Engedélyezd az alkalmazásnak a következő hozzáféréseket a jobb elérőnyél.

Helymeghatározás (GPS)

Közeli boltok és boltok megjelenítéséhez szükséges

✓ Engedélyezés

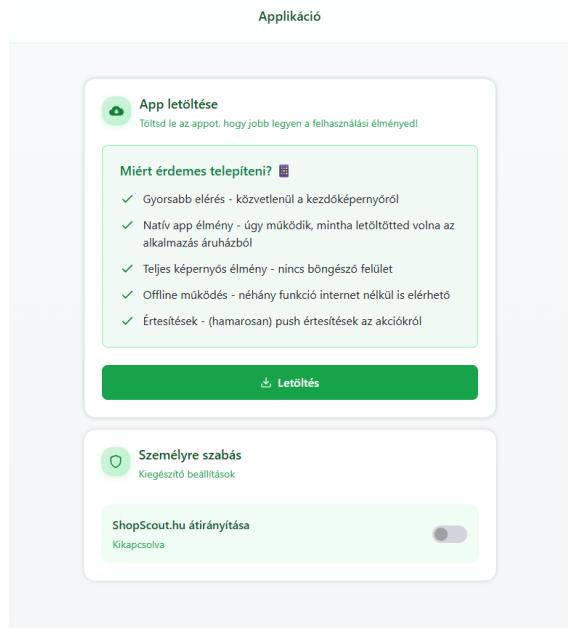
Ezeket az engedélyeket bármikor módosíthatod a beállításokban.

Vissza Befejezés

Ezeken a paneleken, a regisztráció után vagy a megfelelő gombra való kattintás után, testre lehet szabni a profil tulajdonságait, hogy a lehető legmegfelelőbben tudjon működni és kiszolgálja a regisztrált felhasználót.

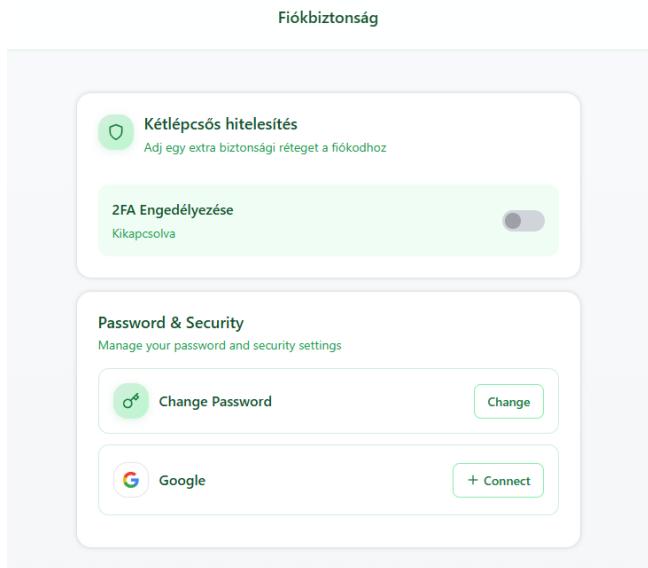
A helyszolgáltatások engedélyezésére különösen nagy szükség van, mert csak ezen beállítás bekapcsolása után fogja tudni a webalkalmazás a felhasználóhoz közeli boltokat felajánlani.

Applikáció beállításai



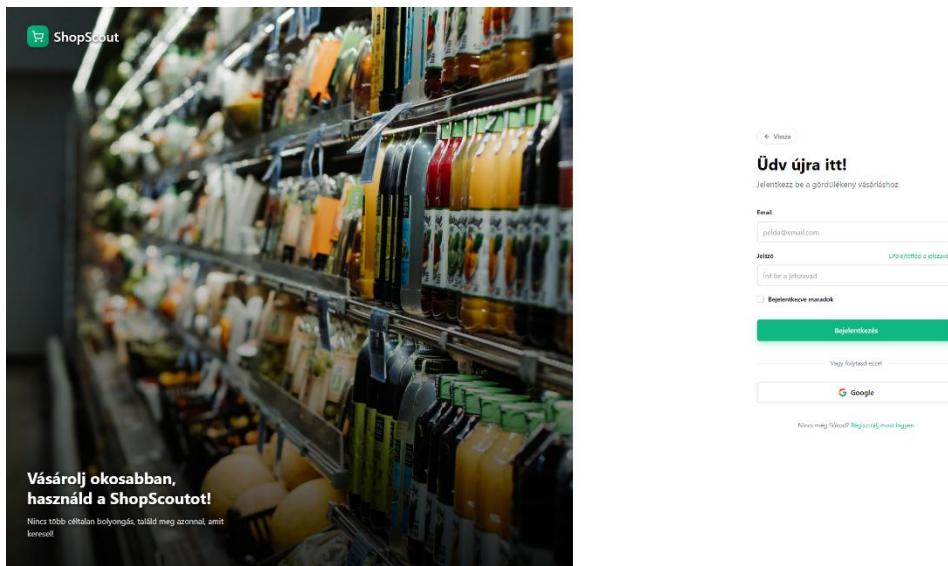
Itt lehetséges a weblap letöltése és a főképrenyőhöz hozzáadása. Ezek után minden telefonon, minden asztali számítógépen vagy tabletén natív alkalmazáshoz hasonló megjelenése lesz, eltüntetve az általános böngésző vezérlőit.

Fiók biztonsági beállításai



Itt van lehetőség két lépcsős azonosítás beállítására és a jelszó megváltoztatására. Emellett össze lehet kapcsolni a fiókot Google fiókkal is a könnyebb bejelentkezés érdekében.

3.4.4. Bejelentkezés



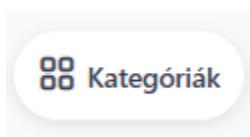
Az autentikációs felület modern, vertikálisan osztott képernyős elrendezést alkalmaz, amely a vizuális márkaépítést ötvözi a funkcionális bejelentkezési folyamatot. A bal oldali panel az alkalmazás fő üzenetét és arculatát hangsúlyozza, míg a jobb oldali űrlap a hitelesítés technikai lebonyolításáért felel. A felhasználók választhatnak a hagyományos e-mail és jelszó alapú azonosítás, illetve a korszerű, Google OAuth protokollon alapuló gyorsított bejelentkezés között. A felület tartalmazza a munkamenet-kezeléshez szükséges „Bejelentkezve maradok” funkciót, továbbá közvetlen logikai kapcsolatot biztosít a jelszóemlékeztetőhöz és az új fiók létrehozásához szükséges regisztrációs modulhoz.

3.4.5. Regisztráció



A regisztrációs felület a bejelentkezési oldalhoz hasonlóan kettős felépítést alkalmaz, biztosítva ezzel az alkalmazás vizuális konzisztenciáját és a professzionális megjelenést. Az oldal bal oldali része az üdvözlő és bejelentkező modulokkal megegyező arculati elemeket és háttérképet használ, míg a jobb oldalon elhelyezett interaktív űrlap az új felhasználói fiók létrehozásához szükséges adatbeviteli mezőket tartalmazza. A regisztrációs folyamat során a szoftver bekéri a felhasználó e-mail címét, valamint a biztonsági előírásoknak megfelelő jelszót, amelynek pontosságát egy megerősítő mező is ellenőrzi. Az űrlap tartalmazza az Általános Szerződési Feltételek és az Adatvédelmi Szabályzat elfogadására vonatkozó tájékoztatást, továbbá lehetőséget biztosít a gyorsított, Google OAuth alapú regisztrációra is. A felület alsó részén elhelyezett hivatkozás segítségével a már regisztrált felhasználók zökkenőmentesen válthatnak vissza a bejelentkezési nézetre.

3.4.6. Kategóriák



A főoldalon ezen gombra kattintáskor lehet elérni a kategóriák aloldalt.

Kategóriák

Keres a kategóriákat között...

Tejtermék, sajt, tojás	Hús, hal, felvágott	Zöldség, gyümölcs	Pékáru	Tartós élelmiszerök...	Speciális étrend	Baba	Háztartás	Szépség és higiénia	Joghurt, túró, kefir	Vajak, vajkrem, margarin
Tejdesszert	Marha	Egyéb húskészítmények	Szalonna	Halak	Gyümölcs	Zöldség	Kenyérfelek	Tésztá, háztartási keksz	Liszt, cukor, olaj	Konzervék, savanyúságok
Víz, gyümölcslé	Rizs	Hüvelyesek, magvak	Tea és kávé	Kakaó	Növényi termékek	Laktózmentes termékek	Laktózmentes tejdesszert	Bébiétel, tápszer	Babaápolás	Eldobható pelenka
Háztartási papíráru és...	Ruhatisztítás	Tisztítószerek	Tisztálkodás	Szépségápolás	Intim higiénia	Natúr joghurt	Kefir és kaukázszi kefir	Tehéntúró	Gyümölcsgoghurt	Margarin
Vaj	Sütőmargarin	Túródesszert	Marhalábszár	Darált marhahús	Fehérpecsente	Sertésszír	Vírsli	Császárszalonna	Kolozsvári szalonna	Pontyszelet
Harcsafile	Tengeri halfilé	Alma	Körte	Földieper	Banán	Citrom	Narancs	Paradicsom	Zöldpaprika	Sárgárapépa
Petrezzelyemgyök	Vöröshagyma	Étkezési burgonya	Étkezési uborgonya	Fejes káposzta	Kelkáposzta	Karfiol	Fehér kenyér, cipő és vénki	Félbarna kenyér, cipő és vénki	Félbarna kenyér, cipő és vénki	Teljes kiőrlésű kenyér, cipő és vénki
										Rozsos kenyér, cipő és vénki

Háztartási keksz **Étolaj, napraforgó** **Finomsízt**

Beolvásás

Keresés

Beállítások

A szoftver hatékonyabb adatkezelése és a felhasználói navigáció megkönnyítése érdekében az alkalmazás egy dedikált Kategóriák modullal rendelkezik. Ez a felület az adatbázisban tárolt termékek adott tulajdonság alapján csoportosításához rendelt összefoglaló nevét használja fel a strukturált megjelenítéshez.

Egy adott kategóriára történő kattintással a rendszer automatikusan szűri az eredménylistát, így a felhasználó csak az adott termékcsoporthoz tartozó elemeket látja, ezzel redukálva a keresési zajt. A kategóriák egy többszintes rendszert alkotnak, ezért egy kategóriára kattintáskor először alkategóriákat lehet találni, majd a végső szinthez érve kilistázza a megfelelő termékeket.

[← Vissza](#)

Tengeri halfilé

Keres a kategóriákat között...

Rozmár lazacfilé bőr nélküli, tisztított, nyers gyorsfagyasztott 540 g Tengeri halfilé	Rozmár Lazac porcida, Salmo salar, bőrös gyorsfagyasztott 1 kg Tengeri halfilé	Selfish lazacfilé natur 250g Tengeri halfilé	Norvég lazacfilé bőr nélküli 160g Tengeri halfilé	FROSTA lazacfilé 180 g Tengeri halfilé	Auchan Kedvenc Atlanti lazacfilé színes (Salmo Salar) gyorsfagyasztott szálak... Tengeri halfilé	Fűszeres lazacfilé bőrös 250g Tengeri halfilé
Fűszeres lazacfilé toscana 250g Tengeri halfilé	Fűszeres lazacfilé corsica 250g Tengeri halfilé	Norvég lazacfilé+fetasajz Tengeri halfilé	SPAR PREMIUM norwegian salmon fillet Tengeri halfilé	SeaFood Hungary Paprikás norwegian salmon fillet Tengeri halfilé	Gygran mare lazacfilé Tengeri halfilé	Lazacfilé, bőrös, 4x125g, 500g Tengeri halfilé

Beolvásás

Keresés

Beállítások

A bizonyos „szintekről”, alkategóriákból vagy termékkistából a „Vissza” gombbal bármikor vissza lehet lépni az előző szintre.

A keresőmezőbe pedig a rendszerben lévő kategóriák között lehet keresni és arra szűkíteni a találatokat.

The screenshot shows a search bar at the top with the placeholder "Keresés". Below it is a horizontal navigation bar with categories: Édes krém, Édes kenhető élelmiszer, Édesség, Édesburgonya, Édesítőszerek, Mesterségesen édesített italok, Tejes csokoládé édesítőszerekkel, and Cremedesserts. Under the "Édesség" category, there is a sub-section titled "Összes kategória" with a search bar. Below this, there are six product cards: Pilóta vaníliás karika, Mini Chocolate Bananas, dodó karika, Sombori galuska, Fruit mix in dark chocolate, and Mozart Pralines. Each card includes a small image of the product and its name in Hungarian.

Az „Összes kategória” gomb pedig a kategória panel főoldalára vezeti a felhasználót, az úgynevezett „gyökérkategóriákhoz”.

3.4.7. Termék részletes adatlapja

The screenshot shows a detailed product page for Nutella. At the top, there is a large image of a 400g jar of Nutella. Below the image, the product name "Nutella" is displayed with a yellow star icon. The page includes nutritional information: Nutri-Score (B), Nova besorolás (4), and Energia (539 kcal). There are tabs for "Termékkategóriák" (Petit déjeuner, Produits à tartiner, Produits à tartiner sucrés, Pâtes à tartiner, Pâtes à tartiner aux noisettes) and "Táptértek" (Tápanyag, Energia, Zár, Amelyből telített zsírsavak). The page also features sections for "Összetevők", "Feldolgoztság", "Csomagolás", and "Képek".

A termékoldal célja az adott árucikk teljes körű specifikációjának megjelenítése. A felület modern, kártya alapú elrendezést használ, amely vizuálisan különíti el az általános információkat a mélyebb szakmai adatoktól.

A felület felső része a gyors tájékozódást és a navigációt támogatja:

- **Navigáció:** A bal felső sarokban található „Vissza a kereséshez” gomb biztosítja a folyamatos felhasználói utat.
- **Azonosítók:** Megjelenítésre kerül a termék pontos megnevezése és vonalkódja.
- **Interakciók:** A felhasználó a szerkesztés ikonnal (jogosultságfüggvényében) kezdeményezheti az adatok frissítését.

A termék fotója mellett kapnak helyet a legfontosabb leíró adatok:

- **Márka és kiszerelés:** Egyértelműen látható a gyártó és a nettó tömeg.
- A „Termékkategóriák” szekciónban dinamikus címkék formájában láthatók a besorolási kategóriák, amelyek segítik a termék környezetének értelmezését.

A szoftver három kulcsfontosságú indikátort emel ki az egészségügyi és táplálkozási mutatók tekintetében, amelyek segítik a tudatos vásárlói döntést:

- **Nutri-Score:** Egy tudományosan megalapozott skála (A-E), amely a termék általános tápanyagtartalmát értékeli.
- **NOVA besorolás:** A feldolgozottsági szintet jelző mutató (1-4-ig), amely figyelmezteti a felhasználót az ultra-feldolgozott élelmiszerekre.
- **Energiaérték:** A 100 grammra vetített kalóriatartalom kiemelt megjelenítése.

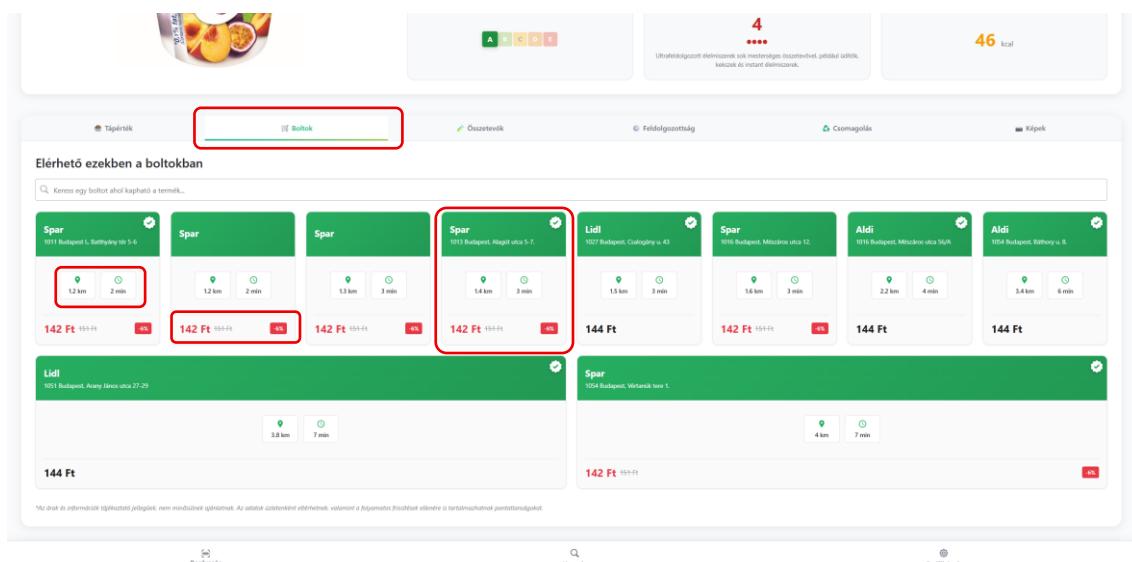
Az oldal többi részén egy füles elrendezésű rendszer található, amely lehetővé teszi a nagy mennyiségű adat szelektált megjelenítését:

1. **Tápérték:** Egy részletes táblázat, amely tartalmazza a legfontosabb tápanyagok (energia, zsír, szénhidrát, fehérje stb.) mennyiségét 100 g-ra vonatkoztatva, valamint a napi beviteli referenciaérték (NRV%) arányát.
2. **Összetevők:** A termék alkotóelemeinek listája, kiemelve az esetleges allergéneket.
3. **Feldolgozottság:** A NOVA besorolás részletes indoklása.

4. Csomagolás: Információk a csomagolóanyagokról és az újrahasznosításról.

5. Képek: További nagyfelbontású fotók a termékről és a címkéről.

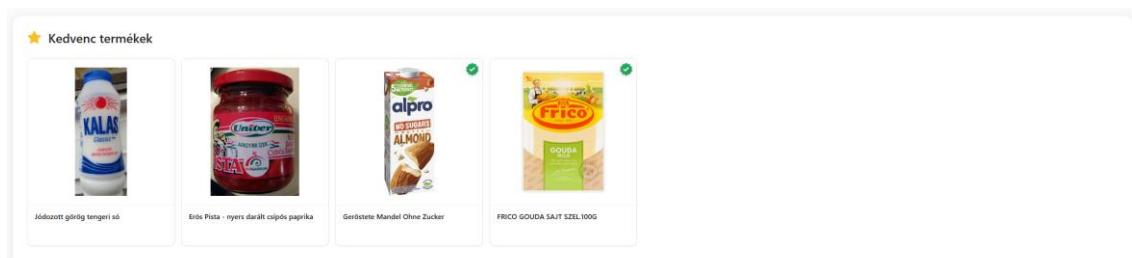
Az olyan termékeknél, ami üzletlánchoz vagy bolthoz van rendelve, egy plusz fül is megjelenik, ahol a boltok láthatóak, amikben kapható a termék. Az termék ára az adott boltban szintén megjelenítésre kerül. Ha akciós a termék, akkor pedig ez külön van jelezve.



Továbbá ha a helymeghatározási szolgáltatások engedélyezve vannak, akkor a bolt távolsága és az odajutási idő is fel van tüntetve.

3.4.8. Bejelentkezett felhasználók – kedvencek

3.4.8.1. Kereső főoldal



A profillal rendelkező felhasználók a főoldalon láthatják az általuk kedvenceknek jelölt termékeket, ezzel is könnyítve a gyakran megtekintett termékek elérését.

Egy terméket a dedikált oldalán lehet a kedvencek közé felvenni a csillag ikonnal. Az üres csillag azt jelzi, hogy még nincs felvéve a kedvencek közé, a teli csillag pedig azt, hogy a felhasználó kedvencnek jelölte már.

3.4.8.2. Termékoldal

The screenshot shows a product page for 'Erős Pista - nyers darált csípős paprika'. At the top, there's a search bar and a 'Vissza a kereséshez' button. The main product image is a jar of red paprika. Below it, the product name is displayed with a yellow star icon. The brand is 'Univer'. The weight is listed as 200g. A 'Termékkategóriák' section shows categories like Condiments, Sauces, and Chili sauces. Nutritional information includes a Nutri-Score of A+, a Nova score of 4 (red), and 41 kcal energy content. Below the main content, there are tabs for Tápteríték, Összetevők, Feldolgoztatás, Csomagolás, and Képek. The Tápteríték tab is active, showing nutritional values per 100g: Energy 41 kcal (2%), Carbohydrates 2 g (3%), Protein 0 g (0%), and Fat 0 g (0%).

Rama Sütőmargarin

FRICO GOUDA SAJT SZEL.100G

3.4.9. Termékek szerkesztője – bejelentkezett felhasználóknak

The screenshot shows a product page for 'Élőflórás, erdei gyümölcös sovány joghurt édesítőszerekkel'. The product image is a container of Zott Jogobella yogurt. The main product name is displayed with a yellow star icon. The brand is 'Zott'. The weight is listed as 150g. A 'Termékkategóriák' section shows categories like Fermented dairy desserts, Tejtermékek, Édesített ételek, Desszertek, Fermentált tejtermékek, Tejszínes desszertek, Joghurtok, Fermented dairy desserts with fruits, and Gyümölcsgyoghurtok. Nutritional information includes a Nutri-Score of A+, a Nova score of 4 (red), and 46 kcal energy content. Below the main content, there are tabs for Tápteríték, Boltok, Összetevők, Feldolgoztatás, Csomagolás, and Képek. The Tápteríték tab is active, showing nutritional values per 100g: Energy 46 kcal (2%), Carbohydrates 0.1 g (0%), Protein 0.1 g (0%), and Fat 0.1 g (0%).

A szerkesztési funkciók kizárolag hitelesített, megfelelő jogosultsággal rendelkező felhasználók számára érhetők el. A rendszer **RBAC (Role-Based Access Control)** alapú védelmet alkalmaz: a szerkesztő felületre mutató

vezérlők csak sikeres bejelentkezés után jelennek meg, megakadályozva az illetéktelen adatmódosítást.

3.4.9.1. Rögzített akcióság („Gomb sziget”)

A felület felső részén egy állandóan látható, rögzített pozíójú vezérlőpanel található. Ez a komponens biztosítja, hogy a szerkesztési folyamat bármely szakaszában elérhetők legyenek a kritikus műveletek:

- **Vissza:** Navigáció a módosítások mentése nélkül a termék megtekintő üzemmódjába.
- **Visszaállítás:** A munkamenet során végzett összes nem mentett változtatás elvetése és az eredeti állapot betöltése.
- **Mentés:** Az adatok validálása és aszinkron továbbítása a backend szerver felé, majd azok véglegesítése az adatbázisban.

3.4.9.2. Adatbeviteli és módosítási funkciók

A szerkesztő felület támogatja a hiányzó attribútumok pótlását és a meglévő adatok korrekcióját modern UI-elemek segítségével:

- **Dinamikus keresőmezők és címkek:** A többértékű tulajdonságok (pl. márka, kategóriák, országok) kereshető beviteli mezőkkel módosíthatóak. A kiválasztott elemek eltávolítható **chip-komponensekként** jelennek meg.
- **Entitásbővítés:** Bizonyos vezérlők (pl. országok, márkkák) mellett található „+ Új” gombbal a felhasználó új törzsadatokat vihet fel a rendszerbe, ha az adott elem még nem szerepel a választólistában.
- **Mezőszintű visszaállítás:** minden beviteli mező mellett egy dedikált visszaállító ikon található. Erre kattintva az adott mező értéke egyedileg, a többi módosítás érintése nélkül állítható vissza az adatbázisban tárolt eredeti állapotra.

Különböző típusú beviteli mezők.

Zott x

Szentkirályi Magyarország Kft

Szentkirályi Magyarország Kft.

Szerencs Csoki

Tokimeki

Vitaking

Wotkin's

+ Create: kj

Tápanyagszintek

	Zsír	Alacsony	Közepes	Magas	(edit)
	Telített zsírsavak	Alacsony	Közepes	Magas	(edit)
	Cukor	Alacsony	Közepes	Magas	(edit)
	Só	Alacsony	Közepes	Magas	(edit)

Összetevő elemzés

	Joghurt	+ New	53.67 %	(edit)
	gyümölcs	+ New	23.22 %	(edit)
	oligofruktóz szirup	+ New	11.61 %	(edit)
	szeder	+ New	11.61 %	(edit)
	áfonya	+ New	5.8 %	(edit)
	Új összetevő neve...	+ New	% %	(edit)

Adalékanyagok

e950 Élelmiszer-adalékanyag	(edit)
e951 Élelmiszer-adalékanyag	(edit)
Adalékanyag keresése... + New (edit)	
E141 Klorofill(in)-rézkomplex <i>Klorofill(in) (E140) színtartalmával, magas réztartalommal vitatott zöld színzékkel. Bőrirritáció, asztmás, légzési nehézségeket okozhat.</i> MEDIUM	
E142 Zöld S <i>Középöntrontronyból színtetizált zöld színzék, több országban betiltották. Többek között szószok, desszertek, édességek, jégkrémek és konzerv borsók színezéséhez használják. Okozhat hiperaktivitást, dímotárolásdugót, allergiát, illetve rökkentést lehet.</i> HIGH	
E150 Karamell (a-d)	

Termék képek

Termék eleje

<https://cdnarfigyeloprodweu.azureedge.net>

Tápértékdatok

<https://images.openfoodfacts.org/images/>

Csomagolás

	Rész: <input type="text" value="Rész neve..."/>	+ New
	Anyag: <input type="text" value="Pp 5 Polypropylene"/>	+ New
	Mennyiség pl. 1x	
	Újrahasznosítás: Igen Nem	

[+ Új csomagolás hozzáadása](#)

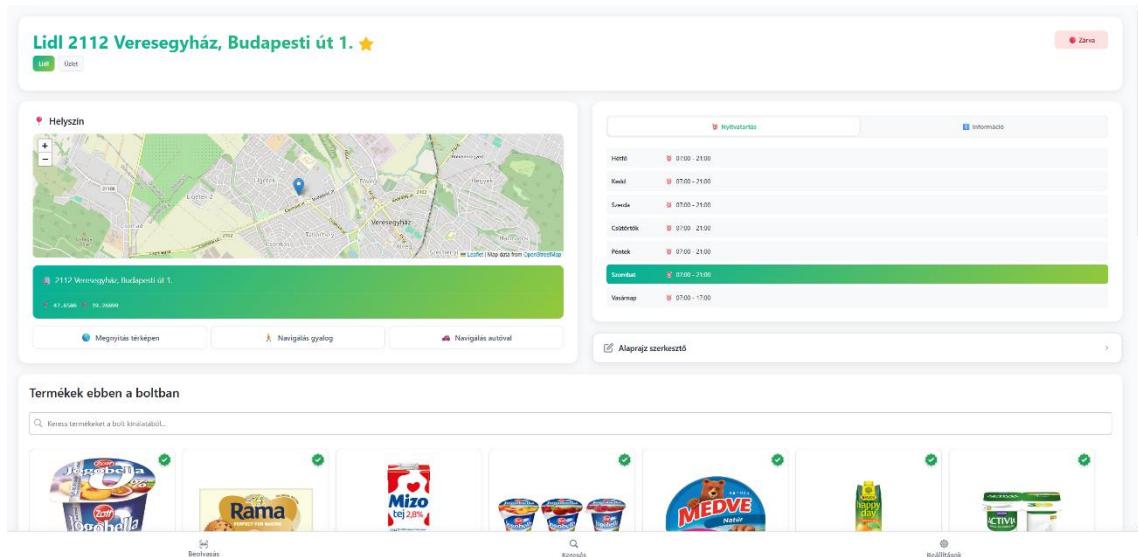
Nincs kép kiválasztva
Adj meg egy URL-t!

Csomagolás

Osszetevők

<https://images.openfoodfacts.org/images/>

3.4.10. Bolt adatlapja



A boltoldal célja, hogy egy adott üzletről minden releváns kereskedelmi és földrajzi információt egyetlen, átlátható felületen összesítsen. Az oldal felépítése a logikai tagolást követi: felül az azonosítók, középen a helyszíni és üzemeltetési adatok, alul pedig a helyi kínálat található.

3.4.10.1. Fejléc és valós idejű állapotjelzők

Az oldal tetején a bolt pontos megnevezése és címe látható

A név mellett elhelyezett csillag ikonnal az üzlet menthető a kedvencek közé a későbbi gyors eléréshez.

A jobb felső sarokban egy dinamikus állapotjelző kártya (pl. „Zárva” vagy „Nyitva”) azonnali vizuális visszajelzést ad az üzlet aktuális elérhetőségéről a rendszeridő alapján.

3.4.10.2. Geográfiai adatok és navigáció (Helyszín modul)

A felület bal oldali egysége az üzlet fizikai elhelyezkedését mutatja be:

- Interaktív térkép:** Egy beágyazott térképi komponens jelöli az üzlet pontos pozícióját.
- Navigációs vezérlők:** A térkép alatt dedikált gombok segítik a felhasználót a helyszín megközelítésében: „Megnyitás térképen”, „Navigálás gyalog” és „Navigálás autóval”. Ezek a gombok külső

terképszoftverekhez (pl. Google Maps) biztosítanak API-szintű kapcsolatot.

3.4.10.3. Üzemeltetési információk (Nyitvatartás modul)

A jobb oldali panel a bolt időbeli elérhetőségét részletezi:

- **Heti menetrend:** Egy strukturált táblázat tartalmazza a hét minden napjára vonatkozó nyitvatartási időt.
- **Aktuális nap kiemelése:** A rendszer automatikusan zöld háttérrel emeli ki az aktuális napot, megkönnyítve az azonnali tájékozódást.
- **Alaprajz szerkesztő:** Az „Alaprajz szerkesztő” menüpont lehetőséget biztosít az üzlet belső elrendezésének kezeléséhez (jogosultság függvényében).

3.4.10.4. Helyi termékkínálat (Termékek ebben a boltban)

- Az oldal alsó szekciójába az adott üzletben elérhető árukészletet listázza:
- **Belső kereső:** Egy dedikált keresőmező segítségével a felhasználó kifejezetten az adott bolt választékában böngészhet.
- **Termékkártya-rács:** Az üzlethez rendelt termékek a korábban bemutatott kártya alapú elrendezésben jelennek meg, tartalmazva a fotót és a legfontosabb alapadatokat.

3.4.10.5. Árazás és akciók vizualizációja

Termék	Ár	Megjegyzések
Joghobella Elsőször	144 Ft	Fermented dairy desserts Működésben
Rama Südmargarin	599 Ft	Növényi alapú előkezavaró és zsírok -10%
Mizo Tej 2,8%	399 Ft	Túrókrémek
Joghobella brokkoli, malina	144 Ft	Dairy
Medve Natur	519 Ft	Túrókrémek
Salt happy 100% Orange	1399 Ft	Pfannenfleisch Létesítmények és Gyümölcsök
Activia	368 Ft	Fermented dairy desserts
Vegusto Vegetál		
Lactose free milk		
Schar Pan Carre		
Südmargarin		
Natúr Orlas Pöttölys Túró Rudi		
Kristálycukor		
Rozs kenyér		

A bolt specifikus terméklistájában az árak és kedvezmények megjelenítése a felhasználói döntéstámogatás érdekében kiemelt figyelmet kapott. A rendszer az adatbázisból érkező árinformációkat dinamikusan kezeli:

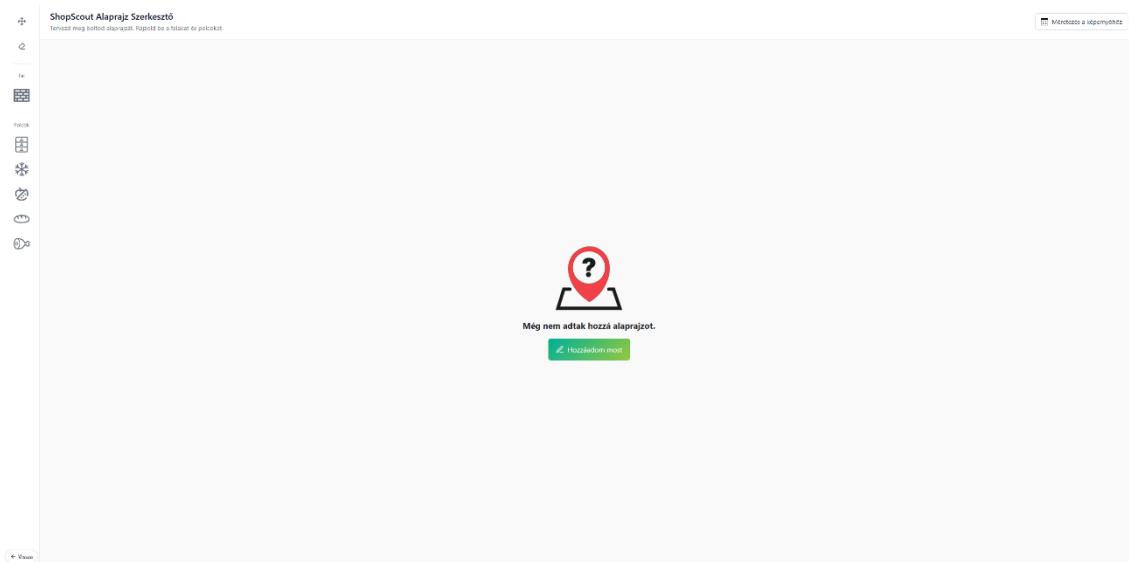
- **Normál árazás:** Az érvényes fogyasztói ár jól olvasható, félkövér betűtíppussal jelenik meg a termékkártyák bal alsó sarkában.
- **Akciós ajánlatok:** Amennyiben egy termékhez kedvezmény kapcsolódik, a felület vizuálisan is elkülöníti azt a normál ártól. Az eredeti ár áthúzva, szürke színnel jelenik meg, míg az aktuális akciós ár feltűnő piros színnel kerül kiemelésre.
- **Elérhetőség jelzése:** A termékkártyákon elhelyezett zöld pipa ikon jelzi a termék adatainak ellenőrzött státuszát, bizonyos esetekben pedig a „Megtekintés boltban” gomb közvetlen interakciót tesz lehetővé.

3.4.10.6. Beltéri navigáció és termékhelymeghatározás

Az alkalmazás egyedi funkciója a vizuális alaprajz, amely segít a felhasználónak lokalizálni a kiválasztott terméket az áruház eladóterén belül.

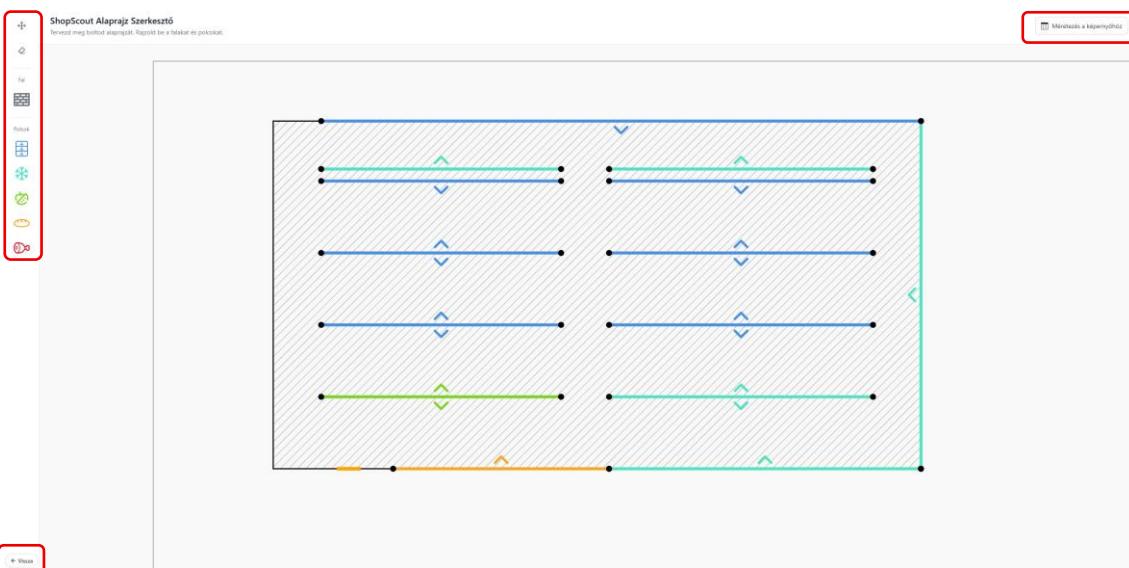
- **Interaktív alaprajz:** A bolt adatlapjáról elérhető modul egy 2D-s felülnézetű térképet jelenít meg, amely egy koordináta-rendszer alapú rácshálón alapul.
- **Polcrendszerek reprezentációja:** Az áruház polcsorait színes vonalak és nyilak jelölik. A különböző színek (kék, zöld, narancs) a különböző termékkategóriák elkülönítésére szolgálnak, a nyilak pedig a polc irányát jelzik.
- **Termékpozíció jelölése:** A keresett termék egy kerekített ikon formájában jelenik meg pontosan azon a ponton, ahol az a valóságban a polcon található. Ez a vizuális visszajelzés minimalizálja a kereséssel töltött időt és növeli a vásárlási hatékonyságot.
- **Reszponzív megjelenítés:** A térképmodul támogatja a nagyítást és az eltolást, biztosítva a részletek pontos megtekintését kisebb kijelzővel rendelkező mobileszközökön is.

3.4.11. Üzlet alaprajzának szerkesztője – bejelentkezett felhasználóknak



Ez a fejezet a szoftver egyik speciális adminisztrációs felületét, az **Alaprajz szerkesztő** modul kezdeti, úgynevezett „üres állapotát” mutatja be. Ez a nézet fogadja a felhasználót, amennyiben az adott üzlethez még nem került feltöltésre vagy kialakításra beltéri térkép.

3.4.11.1. Kitöltött nézet, megszerkesztett bolt



Az eszköztár felépítése:

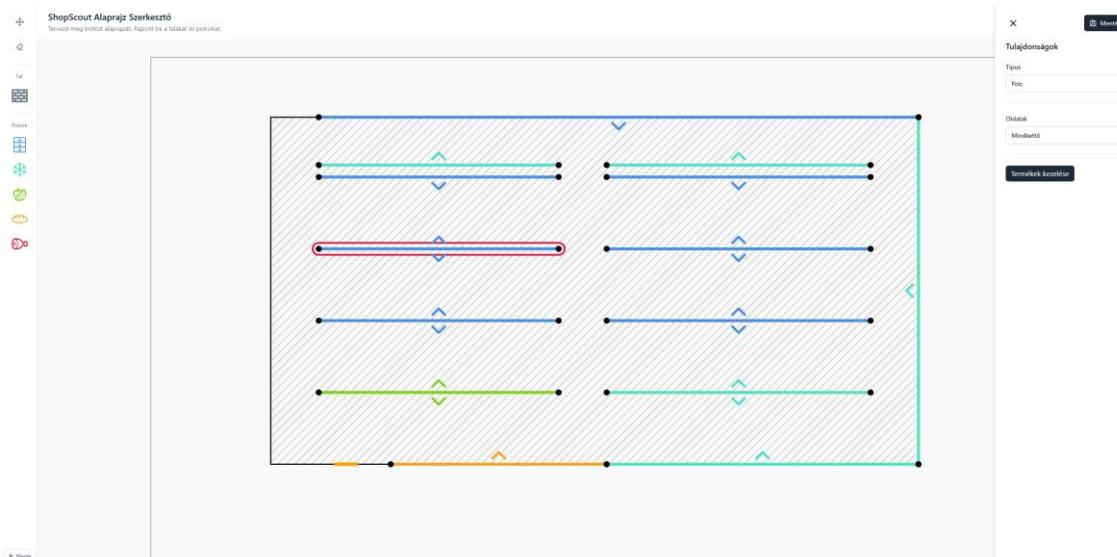
A bal oldali vertikális menüsáv már tartalmazza a tervezéshez szükséges eszközöket, előrevetítve a modul képességeit:

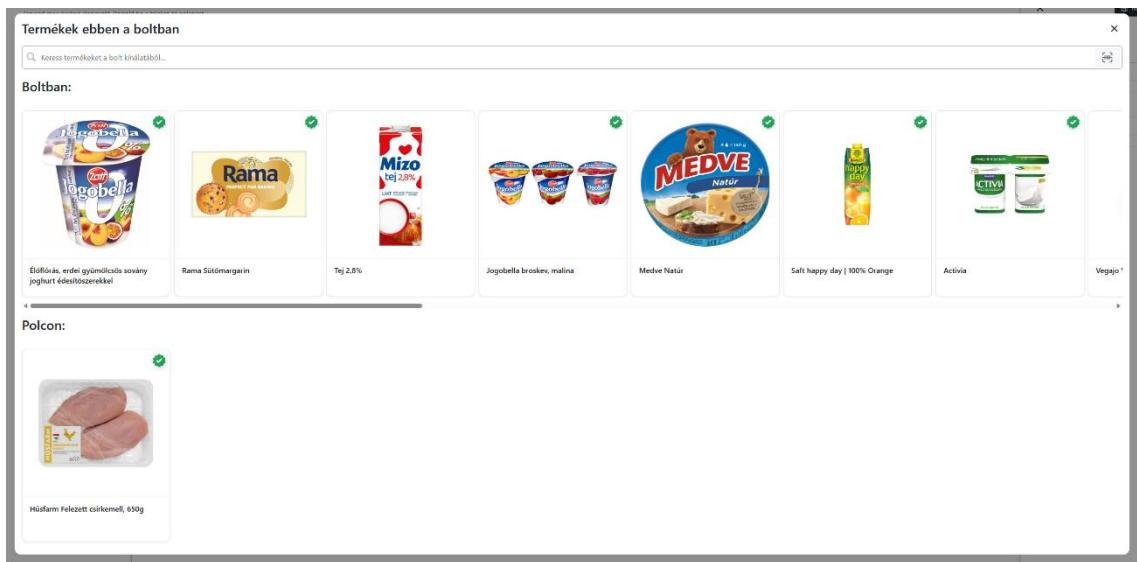
- **Strukturális elemek:** Külön ikonok szolgálnak a falak és a polcrendszerek elhelyezésére.
 - **Kategóriaspecifikus jelölők:** Az eszköztár alsóbb részein található piktogramok (pl. fagyasztott áru, gyümölcs, pékáru, húsáru ikonok) lehetővé teszik a termékcsoporthoz köthetően a zónákra bontását és vizuális azonosítását az alaprajzon.

Navigációs keretrendszer:

A felület felső sávja tartalmazza a modul megnevezését és a méretezési opciókat, míg a bal alsó sarokban elhelyezett „Vissza” gomb biztosítja a zökkenőmentes navigációt az üzlet adatlapjára, megakadályozva a felhasználó „elakadását” a folyamatban.

3.4.11.2. Polc adatainak panelje és termék felrakás





Egy adott polcelem kijelölésekor a jobb oldali tulajdonságok panelen módosíthatóak annak technikai paraméterei, mint például a polc típusának meghatározása vagy a kétoldalas kiszolgálás beállítása. A felület szerves részét képező termékkezelő modális ablak segítségével a bolt teljes árukészletéből válogatva rendelhetők hozzá konkrét termékek az aktuálisan kiválasztott polcszakaszhoz. Ez az integrált adatkezelési folyamat garantálja a vizuális alaprajz és a háttérben futó termékkatalogusz közötti konzisztenciát, ami elengedhetetlen a pontos beltéri navigáció megvalósításához.

3.4.12. Admin kezelőfelület

The screenshot shows the ShopScout management panel. On the left, a sidebar lists navigation options: Dashboard, System Logs, Users (which is currently selected), Products, and Stores. The main content area is titled "MANAGEMENT PANEL" and "User Management". It includes a search bar for "Search username or email..." and a dropdown for "All Roles". There is also a checkbox for "Show Anonymous Only" and a "Search" button.

User Management:

USER	ROLES	REGISTERED	LAST LOGIN	ACTIONS
user@futon.hu	User	2006.02.25.	Just now	[Edit] [Ban]
admin admin@futon.hu	Admin - User	2006.02.25.	Just now	[Edit] [Ban]
kollarandza2006 kollarandza2006@gmail.com	Admin - User	2006.02.20.	2h ago	[Edit] [Ban]
erndt225 erndt7@gmail.com	Admin - User	2006.01.17.	1d ago	[Edit] [Ban]
be barbadoz@gmail.com	Admin - User	2005.10.29.	2h ago	[Edit] [Ban]

System Logs:

Below the user management table, there is a link to "View logs" and a note: "→ Back to Shop".

4. Továbbfejlesztési lehetőségek

Mint minden programnak, a mienknek is van számtalan továbbfejlesztési lehetősége.

- Gyorsbillentyűk hozzáadása: A tapasztalatból felhasználók sokszor előre definiált, vagy akár beállítható billentyűzet kombinációkat is használnak a program kezelése közben, ezzel is gyorsítva a munka sebességét.
- A felhasználók pontozása: a felhasználók értékelhetik egymás tanácsait, útmutatásait. Ezek alapján létre hozunk majd felhasználói kategóriákat, mellyel azt jelöljük, hogy mely felhasználók mennyire releváns és megbízható tanácsokat adnak.
- Marketing: az alkalmazást szeretnénk hosszú távon üzemeltetni. A piaci megjelenéshez befektetőkre van szükségünk, és kellenek olyan támogatók is, akik produceri munkát végeznek.
- Platform-specifikus funkciók beépítése:
 - Mobil eszköz (mint egy iránytű a terepen) használatával:
 - A mobil az elsődleges eszköz, hiszen a vásárló a boltban ezzel navigál. Itt a legfontosabb a gyorsaság és a kezek szabadsága.
 - A PWA segítségével lehetővé tesszük az Offline módot (gyenge bolti térerőnél is eltérhető térképet) és a kezdőképernyőre való telepítést app-bolt nélkül.
 - Kiterjesztett valóság (AR): a felhasználó kamerán keresztül nézi a polcot, a kijelzőn egy virtuális nyíl mutatja, hogy merre található a keresett árucikk.
 - Haptikus visszacsatolás: ha a felhasználó a rögzített polchely közelébe ér, a telefon egy rövid rezgéssel jelezhet – így nem kell folyamatosan a képernyőt figyelni.
 - A „stratégiai tervező”, vagyis a web/asztali nézet:
 - Ekkor a felhasználó nem a boltban van, hanem otthon, a vásárlást tervez. A hangsúly az adatvizualizáció és az optimalizáláson van.

- Útvonal-optimalizálás: A Blazor alapú webes felületen a rendszer kiszámolhatja a leggyorsabb bevásárlókört, több üzletet is érintve, figyelembe véve a közösség által rögzített legfrissebb árakat.
 - Hőterképes elemzés: a felhasználók láthatják, melyik napszakban a legaktívabb a közösség az adott boltban, vagyis hogy mikor érkeznek a friss adatok.
 - Tömeges adatkezelés: itt van lehetőség a közösségi adatgyűjtés moderálására és a megbízhatósági pontok kezelésére (gamification).
- A „hands-free élmény”, vagyis az okosóra:
 - A bevásárlókocsit tologatva a telefon nyomkodása kényelmetlen. Az okosóra a jövő bevásárlólistája!
 - Csuklón hordott lista: az MSSQL adatbázisból szinkronizált lista megjelenik az órán. Ha a vevő bepakol valamit a kosarába, eg érintéssel kihúzza, és a rendszer azonnal frissíti a szerveren, hogy a családtag otthon, vagy egy másik boltban lássa: „ez már megvan”.
 - Közelségi riasztás: abban az esetben, ha egy termék közelébe ért a vásárló, jelzést kap erről.
 - A „Húha-faktor”, vagyis a smart glass (okosszemüveg):
 - Ez még valóban csak a jövő zenéje, de hát a „jövő itt van, és sosem lesz vége!⁵”
 - HUD-navigáció: a szemüvegre vetített head-up-display segítségével a vásárló úgy látja a polcokat, mintha egy videójátékban lenne. Az árak és a C2C üzenetek (pl.: „Figyelj, ez a téTEL lejárat közel!”) közvetlenül a termék mellett lebegnek a látóterében.

⁵ Menyhárt Jenő: *Mocskos idők* című verséből

5. Mellékletek

5.1. Publikus Github repository

Github Repository link: <https://github.com/KGBSS/ShopScout-public>

Adatbázis export: https://boronkay-my.sharepoint.com/:u/g/personal/barta_balazs2005_tanulo_boronkay_hu/IQATtPenipvIQ402NzWTaxMzAaP5AbCpsJXbF8-mXmScKtA?e=yAHKPv

A teszteléshez a felhasználók:

- username: user@user.hu, password: TestPassword1!
- username: admin@admin.hu, password: AdminUser1!

Weboldal: <https://shopscout.hu/>

5.2. Online elérhetőség a ShopScout-hoz (weblap URL)

[ShopScout Weboldal](#)

6. Felhasznált szakirodalmak, források jegyzéke

6.1. Könyvek, tankönyvek

1. Szerző: Aditya Bhargava; cím: *Algoritmusok okosan – Illusztrált útmutató programozóknak*; kiadó: HVG Könyvek; kiadás éve: 2021.; ISBN: 9789633049969; fordító: Laki Beáta
2. Szerző: Andrew Lock; cím: *ASP.NET Core in Action, Third Edition*; kiadó: Manning Publications; kiadás éve: 2023.; ISBN: 9781617298318
3. Szerző: Chloe Thomas; cím: *eCommerce Delivery Strategy: How to use delivery to drive sales and customer satisfaction*; kiadó: Kernu Publishing; kiadás éve: 2016.; ISBN: 978-1909321151
4. Szerző: Chris Sainty; cím: *Blazor in Action*; kiadó: Manning Publications; kiadás éve: 2022.; ISBN: 9781617298646
5. Szerző: David Flanagan; cím: *JavaScript: The Definitive Guide* 7th Edition; kiadó: O'Reilly Media; kiadás éve: 2020.; ISBN: 9781491952023
6. Szerző: Itamar Simonson & Emanuel Rosen; cím: *Abszolút érték: What Really Influences Customers in the Age of (Nearly) Perfect Information*; kiadó: Harper Business; kiadás éve: 2014.; ISBN: 978-0062215611
7. Szerző: Itamar Simonson & Emanuel Rosen; cím: *Abszolút érték – Mi befolyásolja a vásárlót a (majdnem) tökéletes információ korában?*; kiadó: HVG Könyvek; kiadás éve: 2014.; ISBN: 9789633041857; fordító: Rohonyi András
8. Szerző: Itzik Ben-Gan; cím: *T-SQL Fundamentals* (4th Edition); kiadó: Microsoft Press; kiadás éve: 2023.; ISBN: 9780138102104
9. Szerző: Jeff Bezos & Brad Stone; cím: *Minden eladó – Jeff Bezos és az Amazon kora*; kiadó: HVG Könyvek; kiadás éve: 2014.; ISBN: 9789633041550; fordító: Tábori Zoltán
10. Szerző: Jeff Bezos & Brad Stone; cím: *The Everything Store: Jeff Bezos and the Age of Amazon*; kiadó: Little, Brown and Company; kiadás éve: 2013.; ISBN: 978-0316219266

11. Szerző: Joe Casabona; cím: *HTML and CSS: Visual QuickStart Guide* (9th Edition); kiadó: Peachpit Press; kiadás éve: 2020.; ISBN: 9780136702542
12. Szerző: Jon P Smith; cím: *Entity Framework Core in Action*, Second Edition; kiadó: Manning Publications; kiadás éve: 2021.; ISBN: 9781617298363
13. Szerző: Kulcsár István Róbert; cím: *Webáruház marketing*; Szerzői kiadás (támogató: ShopRenter); kiadás éve: 2018.; ISBN: 9786150017105
14. Szerző: Matt Lambert; cím: *Bootstrap 5 Quick Start* (2nd Edition); kiadó: Kindle Direct Publishing / Matt Lambert; kiadás éve: 2021.; ISBN: 9798501250109
15. Szerző: Seth Godin; cím: *Na, ez már marketing! – Nem látnak, amíg nem tanulsz meg látni*; kiadó: HVG Könyvek; kiadás éve: 2020.; ISBN: 9789633048412; fordító: Andó Éva
16. Szerző: Seth Godin; cím: *This Is Marketing: You Can't Be Seen Until You Learn to See*; kiadó: Penguin Random House; kiadás éve: 2018.; ISBN: 9780241370148
17. Szerző: Steigervald Krisztián; cím: *Generációk harca a figyelemért – Hogyan értsük meg egymást a digitális korban?*; kiadó: Partvonal Kiadó; kiadás éve: 2023.; ISBN: 9789632522067
18. Szerző: Tóth Árpád; cím: *Webfejlesztés – HTML5, CSS3, JavaScript* (3. bővített kiadás); kiadó: BBS-Info; kiadás éve: 2022.; ISBN: 9786156347060;

6.2. Tutoriálok, online források

1. **freeCodeCamp.org** - [Blazor Fundamentals Tutorial – Learn Blazor Step-by-Step](#)
2. **IamTimCorey** - [Intro to Blazor in .NET 8 - SSR, Stream Rendering, Auto, and more...](#)
3. **Microsoft** - [Build your first web app with ASP.NET Core using Blazor](#)
4. **IamTimCorey** - [Entity Framework Best Practices - Should EFCore Be Your Data Access of Choice?](#)
5. **Simon Sez IT** - [Microsoft Word Advanced Tutorial - Microsoft Word Tips and Tricks](#)
6. **MDN Web Docs** - [JavaScript](#)
7. **W3Schools** - [JavaScript Tutorial](#)
8. **JAVASCRIPT.INFO** - [The Modern JavaScript Tutorial](#)