BGSzC Pestszentlőrinci Technikum

1184 Budapest Hengersor 34.

1. Parfümvilág

**Konzulens tanár:** **Készítette:**

Dobrocsi Róbertné Laczy Miklós, Kertész Márton

Tartalom

[1 Bevezetés 3](#_Toc195519081)

[1.1 Feladat leírás 3](#_Toc195519082)

[1.2 A felhasznált ismeretek 3](#_Toc195519083)

[1.3 A felhasznált szoftverek 4](#_Toc195519084)

[2 Felhasználói dokumentáció 6](#_Toc195519085)

[2.1 A program általános specifikációja 6](#_Toc195519086)

[2.2 Rendszerkövetelmények 6](#_Toc195519087)

[2.2.1 Hardver követelmények 6](#_Toc195519088)

[2.2.2 Szoftver követelmények 7](#_Toc195519089)

[2.3 3. A program telepítése 7](#_Toc195519090)

[2.4 A program használatának a részletes leírása 8](#_Toc195519091)

[3 Fejlesztői dokumentáció 21](#_Toc195519092)

[3.1 Az alkalmazott fejlesztői eszközök 21](#_Toc195519093)

[3.2 Adatmodell leírása 23](#_Toc195519094)

[3.3 Részletes feladatspecifikáció, algoritmusok 27](#_Toc195519095)

[3.4 Tesztelési dokumentáció 42](#_Toc195519096)

[4 Összefoglalás 43](#_Toc195519097)

[4.1 Önértékelés 43](#_Toc195519098)

[4.2 Továbbfejlesztési lehetőségek 45](#_Toc195519099)

[5 Felhasznált irodalom 46](#_Toc195519100)

# Bevezetés

## Feladat leírás

A Parfümvilág weboldal lehetővé teszi a felhasználók számára, hogy:

* Böngésszenek parfümök között, megismerve azok alapvető adatait (márka, név, nem, illatjegyek, leírás).
* Keressenek parfümökre név vagy márka alapján.
* Szűrjék a találatokat márka, illatjegyek és nem szerint.
* Rendezzék a találatokat név vagy ár alapján.
* Megtekintsék a parfümök részletes adatlapját, beleértve a képet, leírást, illatjegyeket és az elérhetőségi információkat.
* Regisztráljanak és bejelentkezzenek a személyre szabott funkciók eléréséhez.
* Kedvencként jelöljenek meg parfümöket és megtekintsék a kedvenceik listáját.
* Értékeléseket írjanak és olvassanak parfümökről (illatfelhő, tartósság, ár-érték arány, összbenyomás, szöveges vélemény).
* Szerkesszék saját profiljukat (név, email, jelszó, telefonszám).
* Olvasgassanak parfümökkel kapcsolatos friss híreket egy külső API segítségével.
* Tájékozódjanak az oldal működéséről (Rólunk) és felhasználási feltételeiről (ÁSZF).

Különlegesség, hogy az alkalmazásban megjelenő termékadatok és árak egy partnerprogram keretében, jelenleg a Notino webáruháztól származnak. Bár a rendszer felépítése lehetővé tenné több áruház integrálását és valódi árösszehasonlítást, az iskolai projekt keretei között a Notino adatainak feldolgozására és megjelenítésére koncentráltunk. Az adatokat egy megkapott adatbázisból töltöttük be.

## A felhasznált ismeretek

A projekt megvalósítása során az alábbi kulcsfontosságú technológiákat és koncepciókat alkalmaztuk:

* **Frontend:**
  1. **React:** Komponens alapú felhasználói felület fejlesztéséhez.
  2. **JavaScript (ES6+):** A frontend logika megvalósításához.
  3. **HTML5 & CSS3:** A weboldal struktúrájának és megjelenésének kialakításához, beleértve a reszponzív dizájnt (Flexbox, Grid, Media Queries).
  4. **React Router**
  5. **React Context**
  6. **Axios/Fetch API:** Aszinkron HTTP kérések kezelésére a backend API felé.
  7. **Bootstrap:** Néhány komponens és a grid rendszer használata alapján
* **Backend:**
  1. **Node.js:** Szerveroldali JavaScript futtatókörnyezet.
  2. **Express.js:** Webes keretrendszer a Node.js-hez, API végpontok és middleware-ek kezelésére.
  3. **MySQL:** Relációs adatbázis-kezelő rendszer az adatok tárolására.
  4. **Middleware:** Funkciók (pl. hitelesítés ellenőrzése) beékelése a kérés-válasz ciklusba.
* **Általános Koncepciók:**
  1. **Full-Stack Fejlesztés:** Mind a frontend, mind a backend réteg fejlesztése.
  2. **MVC (Model-View-Controller) / MVVM-szerű Architektúra:** A kód strukturálása logikai rétegekre (Routes, Controllers, Models frontend oldalon Services, Components, Pages).
  3. **Adatbázis Tervezés:** Relációs séma kialakítása.
  4. **API Kommunikáció:** Frontend és backend közötti adatcsere mechanizmusai.
  5. **Verziókezelés (Git):** A kód verzióinak követése és együttműködés

## A felhasznált szoftverek

* **Kódszerkesztő:** Visual Studio Code
* **Webböngésző:** Google Chrome, Microsoft Edge
* **Node.js:** Szerveroldali JavaScript futtatókörnyezet (ajánlott LTS verzió, pl. v18.x vagy v20.x)
* **npm (Node Package Manager) vagy yarn:** JavaScript csomagkezelő (Node.js-sel együtt települ)
* **MySQL Adatbázis Szerver:** Adatok tárolására
* **MySQL Adatbázis Kezelő Eszköz:** phpMyAdmin
* **Git:** Verziókezelő rendszer (a kód követéséhez)
* **Postman:** API végpontok tesztelésére szolgáló eszközök (opcionális, de ajánlott a backend fejlesztéshez)
* **Operációs Rendszer:** Windows

# Felhasználói dokumentáció

## A program általános specifikációja

A Parfümvilág egy webes alkalmazás, amelynek célja, hogy segítse a felhasználókat a parfümök világában való eligazodásban. Az oldal központi eleme egy kereshető és szűrhető parfümkatalógus, amely részletes információkat nyújt az egyes illatokról. A felhasználók megtekinthetik a parfümök márkáját, nevét, leírását, illatjegyeit, árát (a Notino kínálata alapján) és a hozzá tartozó képet.

A regisztrált felhasználók számára további funkciók érhetők el:

* **Kedvencek:** Lehetőség van parfümök mentésére egy személyes listába.
* **Értékelések:** A felhasználók értékelhetik a parfümöket több szempont szerint (illatcsík, tartósság, ár-érték, összbenyomás) és szöveges véleményt is írhatnak, valamint olvashatják mások értékeléseit.
* **Profil:** Saját felhasználói fiók adatainak (név, email, jelszó, stb.) megtekintése és módosítása.

Az oldal emellett egy "Hírek" szekciót is tartalmaz, ahol parfümökkel kapcsolatos aktualitások olvashatók, valamint statikus oldalakat biztosít a működésről (Rólunk) és a jogi feltételekről (ÁSZF).

Az alkalmazás fő célja, hogy egy helyen gyűjtse össze a releváns információkat, segítve a felhasználókat a számukra tökéletes parfüm megtalálásában és a tájékozott döntéshozatalban. Az adatokat egy központi adatbázisból nyeri, amely a Notino affiliate programjának köszönhetően tartalmazza annak termékeit.

## Rendszerkövetelmények

### Hardver követelmények

Az alkalmazás egy weboldal, így a használatához nincs szükség speciális hardverre a felhasználó részéről. Az alábbi minimális követelmények javasoltak a zökkenőmentes felhasználói élményhez:

* **Eszköz:** Bármilyen számítógép (asztali vagy laptop), táblagép vagy okostelefon.
* **Processzor:** Modern processzor (az elmúlt 5-7 évből).
* **Memória (RAM):** Legalább 2 GB RAM (böngésző futtatásához).
* **Képernyőfelbontás:** Legalább 360x640 pixel (mobil), de az élmény 1024x768 pixeles vagy nagyobb felbontáson optimális.
* **Internetkapcsolat:** Stabil internetkapcsolat szükséges az oldal betöltéséhez és az adatok lekérdezéséhez.

### Szoftver követelmények

* **Operációs Rendszer:** Bármilyen modern operációs rendszer (Windows 10+, macOS 10.15+, Linux disztribúciók, iOS 13+, Android 7+).
* **Webböngésző:** Az alkalmazás használatához egy naprakész, modern webböngésző szükséges. Ajánlott böngészők:
  + Google Chrome (legfrissebb verzió)
  + Mozilla Firefox (legfrissebb verzió)
  + Safari (legfrissebb verzió)
  + Microsoft Edge (Chromium alapú, legfrissebb verzió)

## 3. A program telepítése

A Parfümvilág egy webalkalmazás, ami azt jelenti, hogy a felhasználóknak nem kell semmit telepíteniük a saját számítógépükre vagy mobileszközükre a használatához.

Az alkalmazás eléréséhez mindössze a következőkre van szükség:

1. Nyisson meg egy modern webböngészőt (lásd 2.2.2 Szoftver követelmények).
2. Írja be az alkalmazás webcímét a böngésző címsorába: http://localhost:3000 (vagy a végleges publikus URL).
3. Nyomja meg az Enter billentyűt.

Az oldal betöltődik, és azonnal használható. Nincs szükség további lépésekre a felhasználó részéről a program futtatásához.

## A program használatának a részletes leírása

**Ez a fejezet részletesen bemutatja a Parfümvilág webalkalmazás fő funkcióinak használatát lépésről lépésre.**

**Első látogatás és Főoldal**

Amikor először látogatja meg a Parfümvilág weboldalát, a főoldal fogadja.

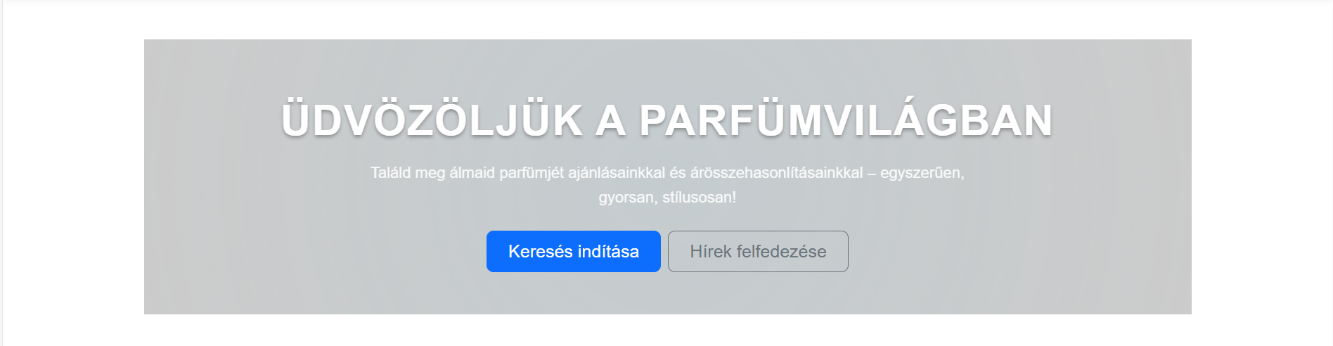
1. **Navigációs Sáv (Navbar):** A lap tetején található a fő navigációs menü. Itt érheti el a különböző szekciókat: "Hírek", "Keresés", "Rólunk". Ha nincs bejelentkezve, itt találja a "Bejelentkezés" és "Regisztráció" gombokat is. Bejelentkezés után ezek helyett a "Kedvencek" és a "Profil" linkek, valamint a "Kilépés" gomb jelenik meg.

Navigációs sáv (kijelentkezett állapot):

Navigációs sáv (bejelentkezett állapot):



1. **Hero Szekció:** A főoldal központi eleme egy figyelemfelkeltő kép vagy üzenet, amely bemutatja az oldal célját és gombokat kínál a keresés megkezdéséhez vagy a hírek felfedezéséhez.



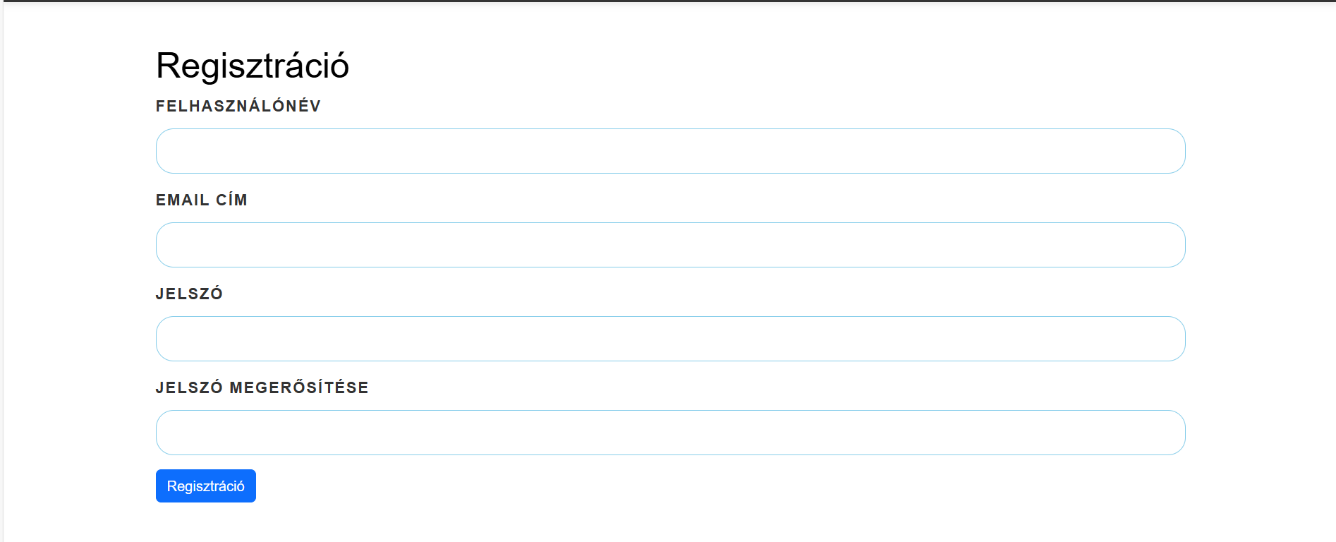
1. **Kiemelt Ajánlatok:** A hero szekció alatt található egy válogatás az ajánlott vagy népszerű parfümökből. Ezekre a parfüm kártyákra kattintva a részletes adatlapjukra navigálhat.



1. **Lábléc (Footer):** A lap alján található a lábléc, amely kapcsolati információkat, közösségi média linkeket, ügyfélszolgálati időket és az Általános Szerződési Feltételekre (ÁSZF) mutató linket tartalmazza.

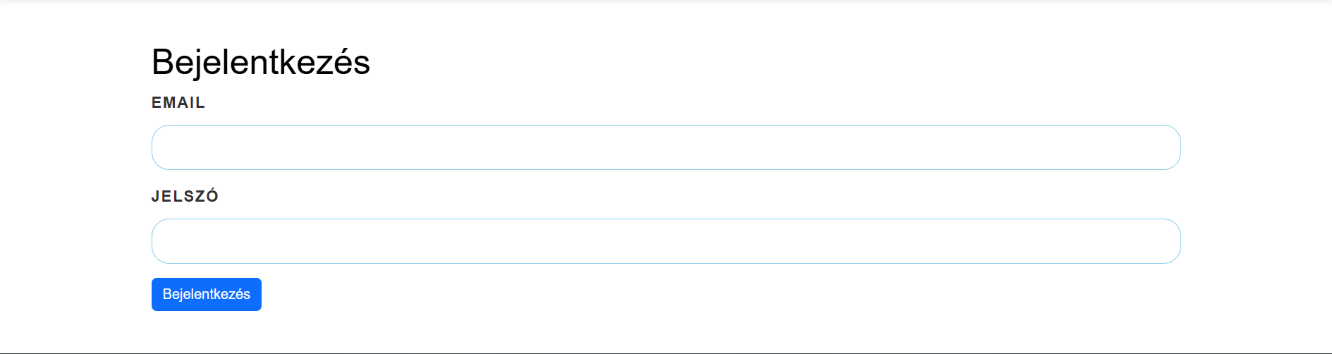
**Regisztráció**

Új felhasználói fiók létrehozásához kövesse az alábbi lépéseket:

1. Kattintson a navigációs sáv jobb felső sarkában található "Regisztráció" linkre.
2. A megjelenő regisztrációs űrlapon töltse ki a kért adatokat:
   * **Felhasználónév:** Adjon meg egy tetszőleges felhasználónevet.
   * **Email cím:** Adja meg érvényes email címét. Ezt fogja használni a bejelentkezéshez.
   * **Jelszó:** Válasszon egy biztonságos jelszót (legalább 6 karakter hosszút).
   * **Jelszó megerősítése:** Írja be újra a választott jelszót.  
     Regisztrációs űrlap mezőkkel
3. Ellenőrizze a megadott adatokat.
4. Kattintson a "Regisztráció" gombra.
5. **Sikeres regisztráció:** Ha minden adat helyes és az email cím még nem foglalt, a rendszer létrehozza a fiókot, automatikusan bejelentkezteti, és átirányítja a Profil oldalra. A navigációs sávon megjelennek a bejelentkezett felhasználóknak szóló menüpontok.
6. **Sikertelen regisztráció:** Hiba esetén (pl. a jelszavak nem egyeznek, a jelszó túl rövid, az email cím már foglalt) az űrlap felett egy piros hibaüzenet jelenik meg, amely tájékoztatja a probléma okáról. Javítsa az adatokat és próbálja újra.

**Bejelentkezés**

Ha már rendelkezik regisztrált fiókkal, az alábbi lépésekkel jelentkezhet be:

1. Kattintson a navigációs sáv jobb felső sarkában található "Bejelentkezés" linkre.
2. A megjelenő bejelentkezési űrlapon adja meg a regisztrációkor használt adatait:
   * **Email:** Írja be az email címét.
   * **Jelszó:** Írja be a jelszavát.
3. Kattintson a "Bejelentkezés" gombra.
4. **Sikeres bejelentkezés:** Ha az email cím és a jelszó helyes, a rendszer bejelentkezteti, és átirányítja a Profil oldalra (vagy arra az oldalra, ahonnan a bejelentkezést kezdeményezte, ha úgy van beállítva). A navigációs sávon megjelennek a bejelentkezett felhasználóknak szóló menüpontok ("Kedvencek", "Profil", "Kilépés").
5. **Sikertelen bejelentkezés:** Ha az email cím vagy a jelszó helytelen, az űrlap felett egy piros hibaüzenet jelenik meg ("Hibás email cím vagy jelszó!"). Ellenőrizze a beírt adatokat és próbálja újra.

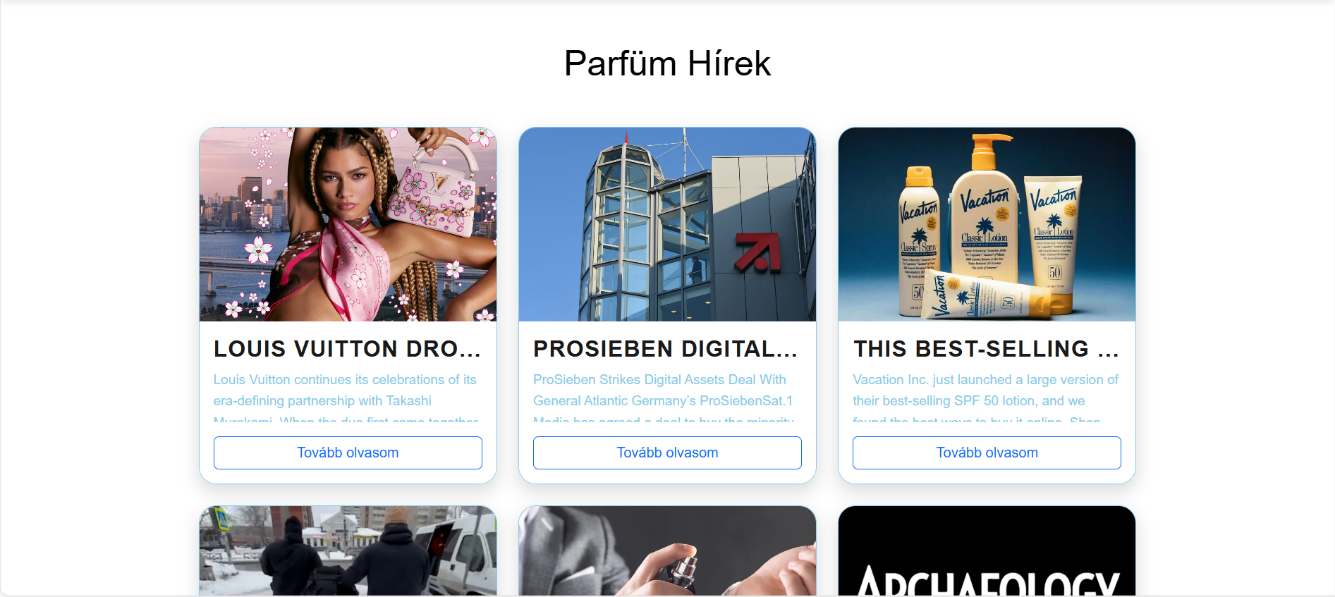
**Navigáció az oldalon**

Az oldalon való közlekedéshez használja a fő navigációs elemeket:

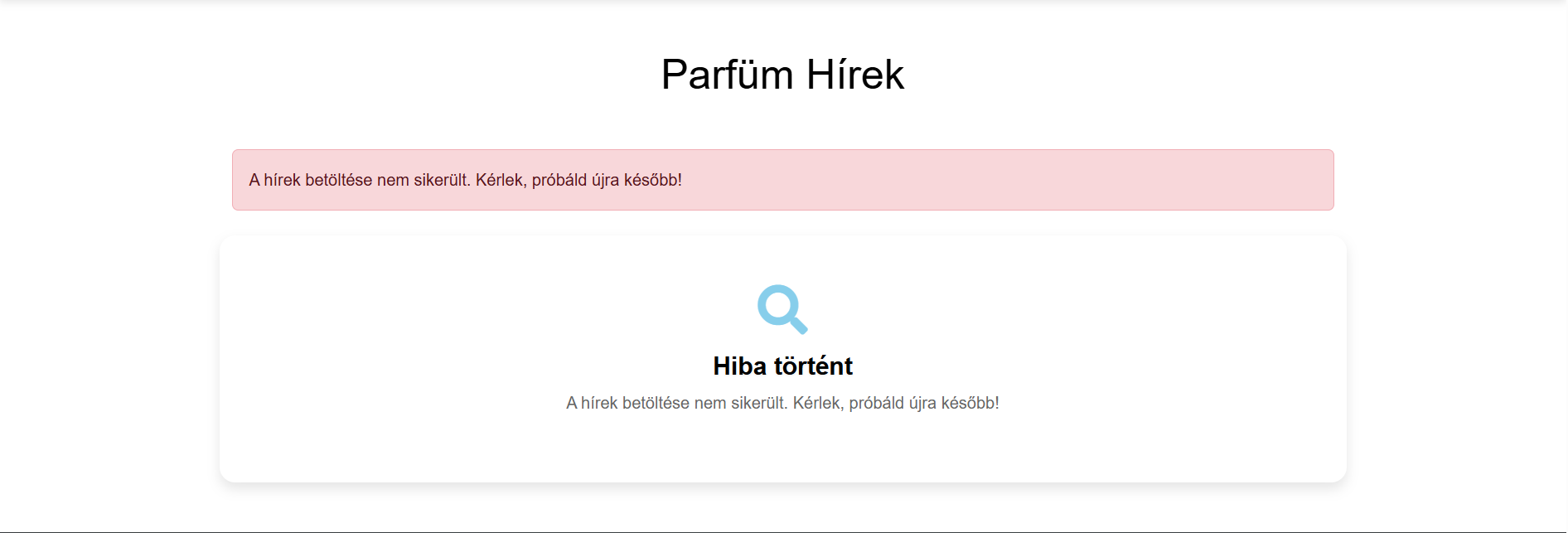
1. **Navigációs Sáv (Navbar):**
   * **Parfümvilág Logó/Név:** Mindig visszavisz a főoldalra.
   * **Hírek:** A parfümökkel kapcsolatos legfrissebb híreket tartalmazó oldalra navigál.
   * **Keresés:** A parfümkatalógus kereső és szűrő oldalára visz.
   * **Rólunk:** Az oldal bemutatkozó oldalát nyitja meg.
   * **Kedvencek (csak bejelentkezve):** A kedvencként megjelölt parfümök listáját mutatja.
   * **Profil (csak bejelentkezve):** A felhasználói profil oldalára navigál, ahol megtekintheti és szerkesztheti adatait. Tartalmazhatja a felhasználó nevét és profilképét is.
   * **Bejelentkezés / Regisztráció (csak kijelentkezve):** A megfelelő űrlapokra vezetnek.
   * **Kilépés (csak bejelentkezve):** Kijelentkezteti a felhasználót az oldalról.
   * **Hamburger Menü (mobilon):** Kisebb képernyőkön a navigációs linkek egy lenyitható menübe kerülnek, amelyet a hamburger ikonra (három vízszintes vonal) kattintva lehet megnyitni/bezárni.  
     [ÁBRA HELYE: Hamburger menü mobilon (nyitott állapot)]
2. **Lábléc (Footer):**
   * **Kapcsolati linkek:** Közvetlen linkek térképhez, telefonszámhoz, email címhez.
   * **Közösségi média ikonok:** Linkek az oldal közösségi profiljaihoz.
   * **ÁSZF link:** Az Általános Szerződési Feltételek oldalára mutat.
3. **"Vissza" gomb:** Egyes aloldalakon (pl. Parfüm Adatlap) található egy "Vissza" gomb, amely az előzőleg megtekintett oldalra navigál.
4. **Belső linkek:** Az oldalon található parfüm kártyákra, gombokra (pl. "Keresés indítása") kattintva az alkalmazáson belüli megfelelő oldalakra juthat.

**Hírek böngészése**

A parfümök világával kapcsolatos hírek és cikkek megtekintése:

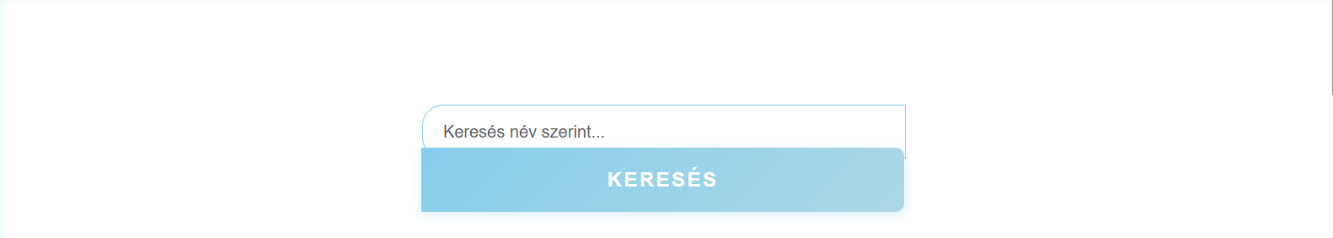
1. Kattintson a navigációs sávban a "Hírek" menüpontra.
2. Az oldalon kártya formátumban jelennek meg a legfrissebb hírek, amelyeket egy külső hírforrásból (NewsAPI) tölt be az alkalmazás.
3. Minden hírkártya tartalmazza:
   * Egy képet (ha elérhető).
   * A cikk címét.
   * A cikk rövid leírását (excerpt).
   * Egy "Tovább olvasom" gombot.

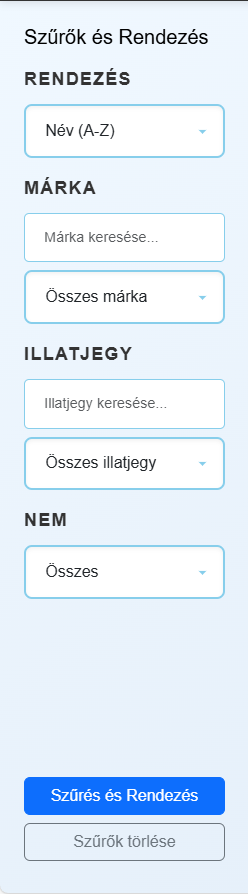
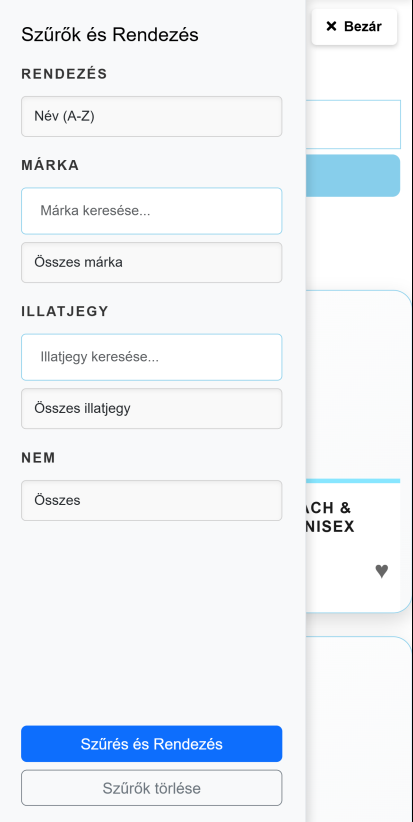
1. Egy adott hír elolvasásához kattintson a hozzá tartozó "Tovább olvasom" gombra. Ezzel megnyílik a hír eredeti forrása egy új böngészőablakban vagy fülön.
2. Ha nem sikerül betölteni a híreket (pl. hálózati hiba miatt), egy hibaüzenet jelenik meg az oldalon. Ha nincsenek releváns hírek, erről is tájékoztatást kap.



**Parfümök keresése és szűrése**

A parfümkatalógus böngészése, keresés és a találatok szűkítése:

1. Kattintson a navigációs sávban a "Keresés" menüpontra. Ezzel a fő katalógusoldalra jut.
2. **Keresőmező:** Az oldal tetején található egy keresőmező. Írja be a keresett parfüm nevét vagy a márka egy részét, majd nyomja meg az Entert vagy kattintson a "Keresés" gombra. A lista frissül a keresési feltételnek megfelelő parfümökkel.
3. **Szűrőpanel (Oldalsáv):**
   * **Elérés:** Nagyobb képernyőkön (asztali nézet) a szűrőpanel általában a bal oldalon látható. Kisebb képernyőkön (mobil/tablet) a szűrőpanel alapértelmezetten rejtett lehet, és egy "Szűrők" vagy hasonló feliratú gombra kattintva lehet előhívni.

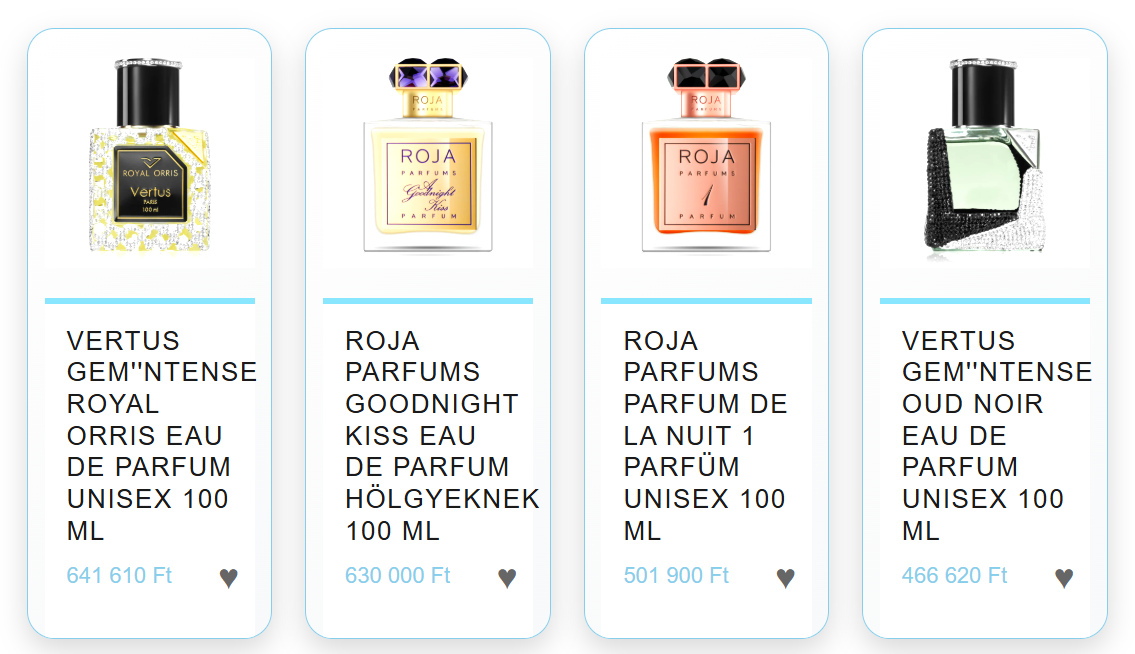
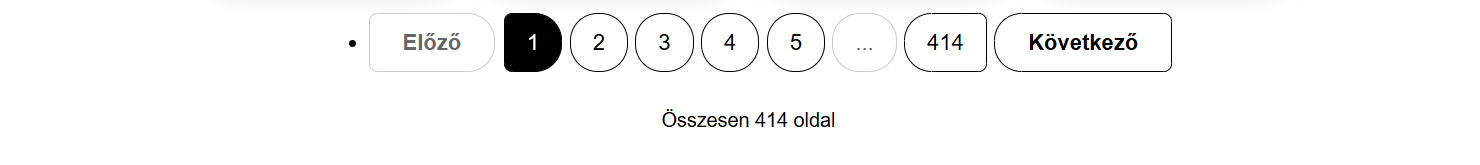
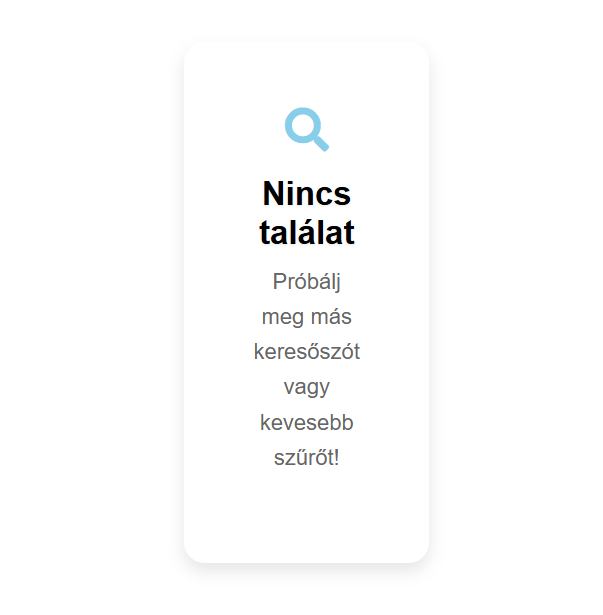


Megnyitott szűrőpanel mobilon

Szűrőpanel (Sidebar) asztali nézetben

Szűrők gomb mobilon

* + **Használat:** A szűrőpanelen az alábbi lehetőségekkel szűkítheti a találati listát:
    - **Rendezés:** Válassza ki, hogy a találatokat Név (A-Z vagy Z-A) vagy Ár (növekvő vagy csökkenő) szerint szeretné-e rendezni.
    - **Márka:** Válasszon ki egy márkát a legördülő listából. A listában kereshet is a márka nevére gépelve. Csak a kiválasztott márkához tartozó parfümök jelennek meg.
    - **Illatjegy:** Válasszon ki egy illatjegyet a legördülő listából. A listában kereshet is az illatjegy nevére gépelve. Csak azokat a parfümöket fogja látni, amelyek tartalmazzák a kiválasztott jegyet.
    - **Nem:** Válassza ki a kívánt kategóriát (Összes, Férfi, Női, Unisex).  
      Szűrőpanel elemei (Rendezés, Márka, Jegy, Nem)]
  + **Szűrés Alkalmazása:** A kívánt szűrők és rendezési mód beállítása után kattintson a szűrőpanel alján található "Szűrés és Rendezés" (vagy hasonló) gombra. A parfümök listája ennek megfelelően frissül.
  + **Szűrők Törlése:** A "Szűrők törlése" gombra kattintva visszaállíthatja az összes szűrőt és a rendezést az alapértelmezett állapotba.

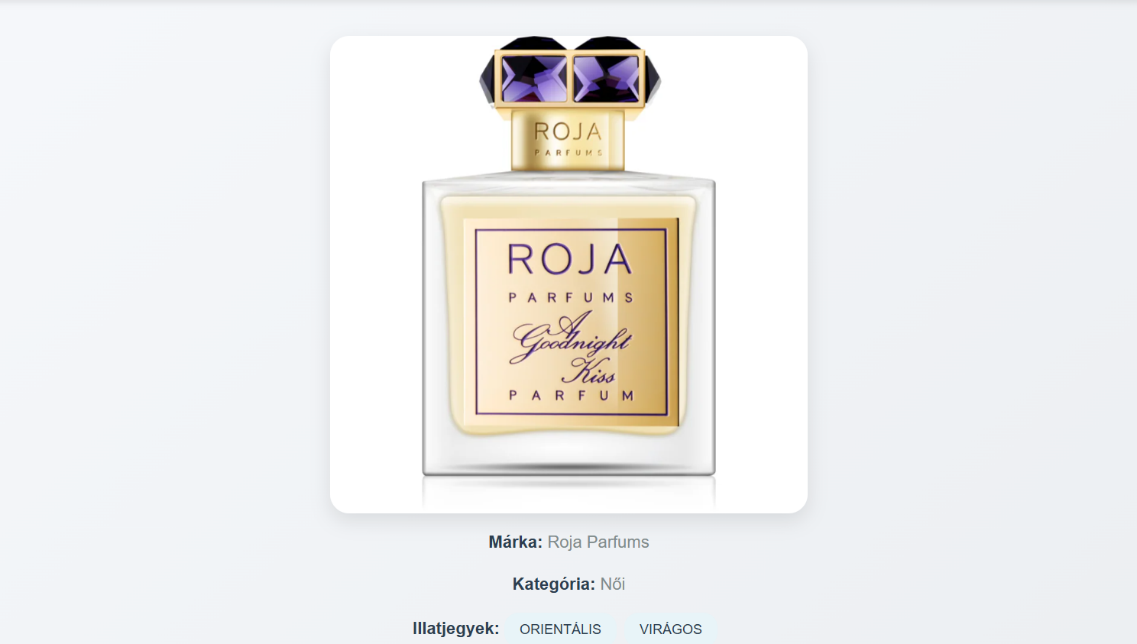
1. **Találati Lista:** A keresési és szűrési feltételeknek megfelelő parfümök kártya nézetben jelennek meg. Minden kártya mutatja a parfüm képét, nevét és árát (a Notino alapján a legolcsóbbat, ha van). Bejelentkezett felhasználók itt is láthatják és használhatják a kedvenc (szív) ikont.
2. **Lapozás (Pagination):** Ha a találatok száma meghaladja az egy oldalon megjeleníthető mennyiséget (pl. 24), a lista alatt megjelenik a lapozó. A számokra vagy az "Előző"/"Következő" gombokra kattintva navigálhat a találati oldalak között. Az aktuális oldalszám ki van emelve.
3. **Nincs Találat:** Ha a keresési/szűrési feltételekre nincs egyetlen találat sem, egy informatív üzenet jelenik meg a listában.
4. **Parfüm Adatlapra Navigálás:** Bármelyik parfüm kártyájára kattintva megtekintheti annak részletes adatlapját

**Parfüm adatlap megtekintése**

Egy konkrét parfüm részletes adatainak és a hozzá kapcsolódó információknak a megtekintése:

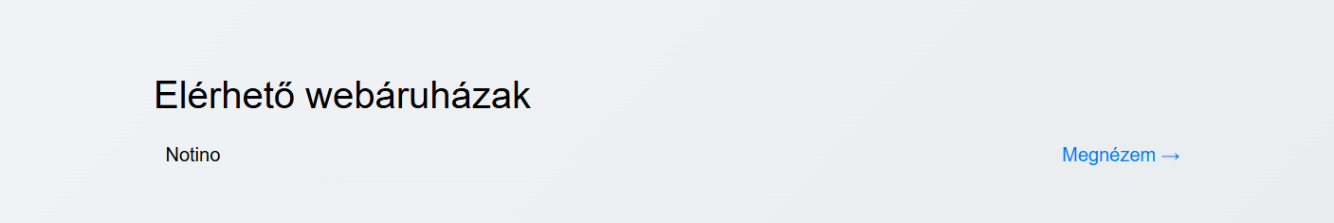
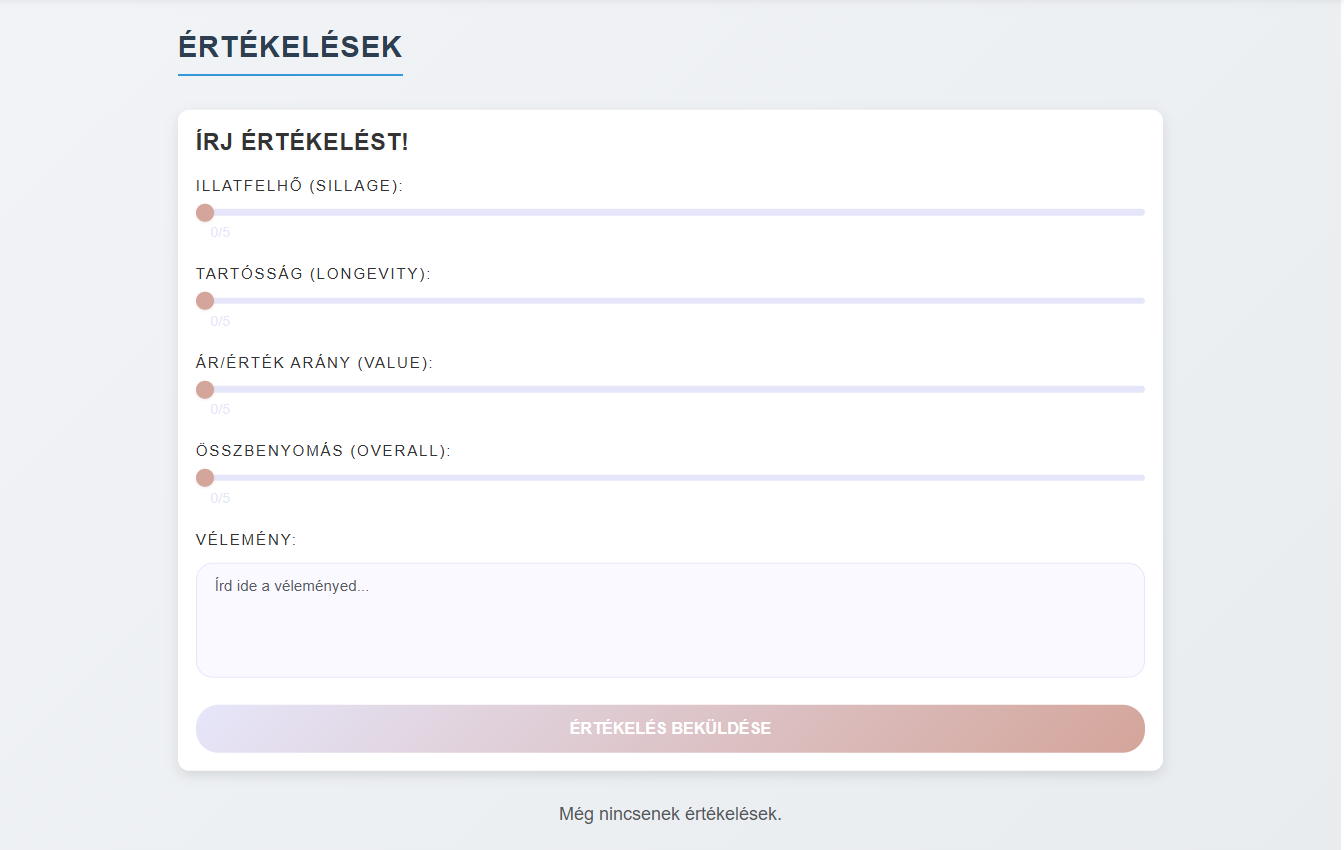
1. Navigáljon egy parfüm adatlapjára a keresési találati listából, a főoldali ajánlatokból vagy a kedvencek listájából a parfüm kártyájára kattintva.
2. **Vissza gomb:** Az oldal tetején található "← Vissza" gombbal visszatérhet az előző oldalra (pl. a keresési találatokhoz).
3. **Fő Információk:**
   * **Név és Márka:** Jól láthatóan megjelenik a parfüm teljes neve és a márkája.
   * **Kedvenc Gomb:** A név mellett található a szív ikon. Ha be van jelentkezve, erre kattintva adhatja hozzá vagy távolíthatja el a parfümöt a kedvencei közül. A gomb színe jelzi az aktuális állapotot (pl. piros, ha kedvenc).



* + **Kép:** Megjelenik a parfüm képe. A képre kattintva az kinagyítható teljes képernyős nézetbe. A nagyított nézetből a kép melletti területre vagy a bezárás (X) ikonra kattintva lehet visszatérni.  
     Parfüm adatlap – Kép:

Parfüm adatlap - Nagyított kép nézet

* + **Kategória (Nem):** Megjelenik, hogy férfi, női vagy unisex parfümről van-e szó.
  + **Illatjegyek:** Címke (tag) formátumban láthatóak a parfüm főbb illatjegyei.  
     Parfüm adatlap - Illatjegy címkék, leírás
  + **Ár:** Megjelenik a parfüm aktuális (legolcsóbb elérhető) ára a Notino kínálatából, formázva (pl. "38 000 Ft"). Ha az ár nem elérhető, erről tájékoztatást kap.
  + **Leírás:** Rövid vagy hosszabb leírás a parfümről, annak karakteréről, összetevőiről.

1. **Elérhető Webáruházak:**
   * Ez a szekció listázza azokat a (jelen projektben csak a Notino) webáruházakat, ahol a parfüm elérhető.
   * Minden bolthoz tartozik egy "Megnézem →" vagy hasonló link, amelyre kattintva közvetlenül a parfüm oldalára juthat a Notino webáruházban (új böngészőfülön nyílik meg).  
      Parfüm adatlap - Elérhető webáruházak szekció
2. **Értékelések Szekció:**
   * **Értékelés Írása:** Ha be van jelentkezve, itt talál egy űrlapot, ahol értékelheti a parfümöt. Ha nincs bejelentkezve, egy üzenet jelenik meg a bejelentkezés szükségességéről, egy linkkel a bejelentkezési oldalra.
   * **Értékelések Listája:** Az űrlap alatt (vagy ha nincs űrlap, akkor önállóan) láthatóak a parfümhöz már beküldött felhasználói értékelések. Minden értékelés mutatja a felhasználó nevét, az értékelés dátumát, a megadott pontszámokat (illatcsík, tartósság, stb.) csillagokkal vagy csúszkákkal megjelenítve, és a szöveges véleményt.  
      Parfüm adatlap - Értékelések szekció

# Fejlesztői dokumentáció

A fejlesztői dokumentáció célja, hogy a segítse program logikájának, illetve a program kódjának a megértését, illetve a program továbbfejlesztését. Szakemberek számára készül, elvárás tehát a pontosság és a szakmai jellegű stílus.

## Az alkalmazott fejlesztői eszközök

* **React (v18.x):** JavaScript könyvtár a felhasználói felület építéséhez. Komponens alapú architektúrát, virtuális DOM-ot és JSX szintaxist használ.
* **React Router DOM (v6.x):** Kliensoldali útválasztást (routing) és navigációt biztosító könyvtár. Lehetővé teszi a Single Page Application (SPA) működést.
* **React Context API:** Beépített React mechanizmus az állapot globális megosztására komponensfák között anélkül, hogy props-okat kellene manuálisan továbbadni minden szinten.
* **Axios:** Promise-alapú HTTP kliens böngészőhöz és Node.js-hez. A backend API-val való kommunikációra használjuk a services rétegben. Könnyen kezelhetők vele a kérések, válaszok és hibák.
* **Fetch API:** Beépített böngészős API aszinkron hálózati kérésekhez. A Sidebar.js-ben és Catalog.js-ben (NewsAPI) került felhasználásra.
* **CSS3:** A megjelenés és elrendezés stílusozására. Főbb jellemzők:
  + **Szelektorok:** Osztály (.class), ID (#id), attribútum, pszeudo-osztályok.
  + **Layout:** Flexbox és CSS Grid a rugalmas és reszponzív elrendezésekhez.
  + **Reszponzivitás:** @media query-k használata a különböző képernyőméretekhez való alkalmazkodáshoz.
  + **Animációk/Tranzíciók:** @keyframes és transition tulajdonságok a dinamikusabb felhasználói élményért.
  + **Szervezés:** A style.css egy nagyméretű fájl, amely szinte az összes stílust tartalmazza. Ez potenciális karbantartási kihívást jelenthet (lásd 4.2 Továbbfejlesztési lehetőségek). Az App.css minimális, általános stílusokat tartalmaz.
* **Bootstrap (v5.x):** CSS/JS keretrendszer. Valószínűleg a grid rendszere (container, row, col-\*), néhány alapvető stílus (btn, alert, card - bár ezek felül lehetnek írva), és esetleg modális ablakok (LoginPromptModal) vagy a navbar lenyíló menüjének működéséhez van használatban (bár a JS része nem látszik a kódban).
* **Font Awesome (feltételezett):** Ikonok (pl. fas fa-heart, fab fa-facebook-f) használata alapján.

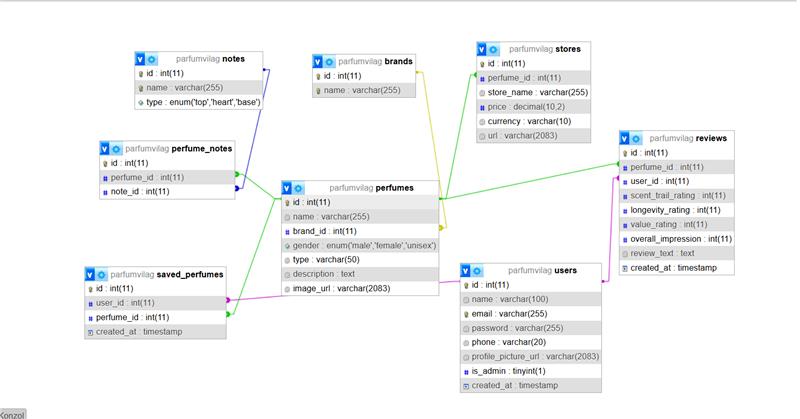
**Backend technológiák**

* **Node.js (v18.x / v20.x LTS):** Aszinkron, eseményvezérelt JavaScript futtatókörnyezet a szerveroldali logika futtatásához.
* **Express.js (v4.x):** Minimalista és rugalmas Node.js webalkalmazás keretrendszer. Kezeli az útválasztást, middleware-eket és a kérés-válasz ciklust.
* **mysql:** Node.js driver a MySQL adatbázissal való kommunikációhoz. Lehetővé teszi SQL lekérdezések futtatását és az eredmények feldolgozását. Támogatja a paraméteres lekérdezéseket az SQL injection megelőzésére.
* **jsonwebtoken (jwt):** JWT tokenek létrehozására (aláírására) és ellenőrzésére szolgáló könyvtár a felhasználói hitelesítéshez.
* **bcryptjs:** Jelszavak biztonságos hashelésére és ellenőrzésére szolgáló könyvtár. A hash és compare funkciókat használja.
* **cors:** Express middleware a CORS beállítások kezelésére, lehetővé téve a frontend számára az API elérését más domainről/portról.
* **body-parser:** Express middleware a bejövő kérések törzsének (pl. JSON, URL-encoded) feldolgozására és a req.body objektumhoz való csatolására.
* **dotenv:** Modul a környezeti változók .env fájlból való betöltésére. Biztonságosabbá teszi a konfigurációs adatok (pl. adatbázis jelszó, JWT titok) kezelését.

**Adatbázis**

* **MySQL:** Relációs adatbázis-kezelő rendszer (RDBMS). Táblákban, sorokban és oszlopokban tárolja az adatokat. SQL (Structured Query Language) nyelvet használ az adatok lekérdezésére és manipulálására.

## Adatmodell leírása

**Adatbázis séma (ER diagram)**

Az alábbi ER (Entity-Relationship) diagram szemlélteti a főbb táblák közötti kapcsolatokat.

Adatbázis ER diagramja - Táblák: users, perfumes, brands, notes, perfume\_notes, stores, reviews, saved\_perfumes. Kapcsolatok jelölése: egy-a-többhöz, több-a-többhöz]

**Diagram magyarázata:**

* **users ↔ reviews:** Egy felhasználónak több értékelése lehet, egy értékelés egy felhasználóhoz tartozik (egy-a-többhöz).
* **users ↔ saved\_perfumes:** Egy felhasználó több parfümöt menthet el kedvencként, egy mentett parfüm egy felhasználóhoz tartozik (egy-a-többhöz).
* **perfumes ↔ reviews:** Egy parfümhöz több értékelés tartozhat, egy értékelés egy parfümhöz tartozik (egy-a-többhöz).
* **perfumes ↔ saved\_perfumes:** Egy parfümöt több felhasználó is elmenthet kedvencként, egy mentett parfüm egy parfümhöz tartozik (egy-a-többhöz).
* **perfumes ↔ stores:** Egy parfüm több boltban is elérhető lehet, egy bolti bejegyzés egy parfümhöz tartozik (egy-a-többhöz).
* **brands → perfumes:** Egy márka több parfümöt gyárthat, egy parfüm egy márkához tartozik (egy-a-többhöz).
* **perfumes ↔ notes (perfume\_notes kapcsolótáblán keresztül):** Egy parfümnek több illatjegye lehet, és egy illatjegy több parfümben is előfordulhat (több-a-többhöz). A perfume\_notes tábla valósítja meg ezt a kapcsolatot.

**Táblák részletes leírása**

Az alábbiakban részletezzük a főbb adatbázis-táblák várható oszlopait és azok típusait.

1. **users**
   * Feladata: A regisztrált felhasználók adatainak tárolása.
   * Oszlopok:
     + id (INT, PRIMARY KEY, AUTO\_INCREMENT): Egyedi felhasználói azonosító.
     + name (VARCHAR): Felhasználó neve.
     + email (VARCHAR, UNIQUE): Felhasználó email címe (egyedi).
     + password (VARCHAR): Hash-elt felhasználói jelszó.
     + phone (VARCHAR, NULLABLE): Felhasználó telefonszáma (opcionális).
     + profile\_picture\_url (VARCHAR, NULLABLE): Felhasználó profilképének URL-je (opcionális).
     + created\_at (TIMESTAMP, DEFAULT CURRENT\_TIMESTAMP): Regisztráció időbélyege.
     + updated\_at (TIMESTAMP, DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP): Utolsó módosítás időbélyege.
     + is\_admin (BOOLEAN/TINYINT, DEFAULT 0, feltételezett): Jelzi, hogy a felhasználó adminisztrátor-e (opcionális).
2. **perfumes**
   * Feladata: A parfümök alapvető adatainak tárolása.
   * Oszlopok:
     + id (INT, PRIMARY KEY, AUTO\_INCREMENT): Egyedi parfüm azonosító.
     + name (VARCHAR): Parfüm neve.
     + brand\_id (INT, FOREIGN KEY → brands.id): A parfüm márkájának azonosítója.
     + gender (ENUM('male', 'female', 'unisex')): Parfüm neme/kategóriája.
     + description (TEXT, NULLABLE): Parfüm leírása.
     + image\_url (VARCHAR, NULLABLE): Parfüm képének URL-je.
     + created\_at (TIMESTAMP, DEFAULT CURRENT\_TIMESTAMP): Rekord létrehozásának ideje.
     + updated\_at (TIMESTAMP, DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP): Rekord utolsó módosításának ideje.
3. **brands**
   * Feladata: Parfümmárkák tárolása.
   * Oszlopok:
     + id (INT, PRIMARY KEY, AUTO\_INCREMENT): Egyedi márka azonosító.
     + name (VARCHAR, UNIQUE): Márka neve (egyedi).
4. **notes**
   * Feladata: Illatjegyek tárolása.
   * Oszlopok:
     + id (INT, PRIMARY KEY, AUTO\_INCREMENT): Egyedi illatjegy azonosító.
     + name (VARCHAR, UNIQUE): Illatjegy neve (egyedi, pl. "Citrus", "Vanília", "Fás").
5. **perfume\_notes** (Kapcsolótábla)
   * Feladata: A parfümök és illatjegyek közötti több-a-többhöz kapcsolat megvalósítása.
   * Oszlopok:
     + perfume\_id (INT, FOREIGN KEY → perfumes.id, PRIMARY KEY részeként): Parfüm azonosítója.
     + note\_id (INT, FOREIGN KEY → notes.id, PRIMARY KEY részeként): Illatjegy azonosítója.
     + (Összetett elsődleges kulcs a perfume\_id és note\_id pároson).
6. **stores**
   * Feladata: Parfümök elérhetőségének tárolása webáruházakban (jelenleg csak Notino).
   * Oszlopok:
     + id (INT, PRIMARY KEY, AUTO\_INCREMENT): Egyedi bolti bejegyzés azonosító.
     + perfume\_id (INT, FOREIGN KEY → perfumes.id): A parfüm azonosítója, amelyhez ez az elérhetőség tartozik.
     + store\_name (VARCHAR): A webáruház neve (pl. "Notino").
     + url (VARCHAR): A parfüm konkrét URL-je az adott webáruházban.
     + price (DECIMAL/FLOAT): A parfüm ára az adott boltban.
     + currency (VARCHAR): Az ár pénzneme (pl. "Ft" vagy "HUF").
     + last\_checked\_at (TIMESTAMP, NULLABLE): Az ár és elérhetőség utolsó ellenőrzésének ideje (opcionális).
7. **reviews**
   * Feladata: Felhasználói értékelések tárolása parfümökről.
   * Oszlopok:
     + id (INT, PRIMARY KEY, AUTO\_INCREMENT): Egyedi értékelés azonosító.
     + perfume\_id (INT, FOREIGN KEY → perfumes.id): Az értékelt parfüm azonosítója.
     + user\_id (INT, FOREIGN KEY → users.id): Az értékelést író felhasználó azonosítója.
     + scent\_trail\_rating (TINYINT): Illatcsík értékelése (1-5).
     + longevity\_rating (TINYINT): Tartósság értékelése (1-5).
     + value\_rating (TINYINT): Ár-érték arány értékelése (1-5).
     + overall\_impression (TINYINT): Összbenyomás értékelése (1-5).
     + review\_text (TEXT, NULLABLE): Szöveges vélemény.
     + created\_at (TIMESTAMP, DEFAULT CURRENT\_TIMESTAMP): Értékelés létrehozásának ideje.
     + updated\_at (TIMESTAMP, DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP): Értékelés utolsó módosításának ideje.
     + (Lehet UNIQUE index a perfume\_id és user\_id párosra, hogy egy user csak egyszer értékelhessen egy parfümöt).
8. **saved\_perfumes**
   * Feladata: Felhasználók által kedvencként mentett parfümök kapcsolatának tárolása.
   * Oszlopok:
     + id (INT, PRIMARY KEY, AUTO\_INCREMENT): Egyedi mentési azonosító.
     + user\_id (INT, FOREIGN KEY → users.id): A felhasználó azonosítója.
     + perfume\_id (INT, FOREIGN KEY → perfumes.id): A mentett parfüm azonosítója.
     + created\_at (TIMESTAMP, DEFAULT CURRENT\_TIMESTAMP): Mentés időbélyege.
     + (UNIQUE index a user\_id és perfume\_id párosra).

## Részletes feladatspecifikáció, algoritmusok

**Architektúra áttekintése**

A Parfümvilág alkalmazás egy klasszikus kliens-szerver architektúrát követ:

1. **Frontend (Kliensoldal):**
   * React alapú Single Page Application (SPA).
   * A böngészőben fut.
   * Felelős a felhasználói felület megjelenítéséért, a felhasználói interakciók kezeléséért és a backend API-val való kommunikációért.
   * Az állapotkezelést (pl. bejelentkezési státusz, keresési eredmények) komponens szintű state és a React Context API segítségével valósítja meg.
   * A navigációt a React Router DOM kezeli.
2. **Backend (Szerveroldal):**
   * Node.js és Express.js alapú RESTful API.
   * Felelős az üzleti logika végrehajtásáért, az adatbázis-műveletek kezeléséért és az adatok JSON formátumban történő szolgáltatásáért a frontend felé.
   * Kezeli a felhasználói hitelesítést (JWT) és autorizációt (middleware).
   * MySQL adatbázissal kommunikál az adatok perzisztens tárolásához.
3. **Adatbázis:**
   * MySQL szerver, amely tárolja az összes alkalmazásadatot (felhasználók, parfümök, márkák, értékelések stb.).
4. **Külső Szolgáltatások:**
   * **NewsAPI:** Hírek lekérésére szolgál a "Hírek" oldalon.

**Adatfolyam (Példa: Parfüm keresés):**

1. A felhasználó beír egy keresőszót és/vagy kiválaszt szűrőket a frontend felületén (Sidebar).
2. A frontend (pl. Search.js komponens) elküld egy GET kérést a backend /api/perfumes/all végpontjára, query paraméterekként átadva a keresőszót, szűrőket, rendezési opciót és oldalszámot.
3. A backend Express útválasztója a kérést a perfumeController.getAllPerfumes funkcióhoz irányítja.
4. A kontroller validálja a beérkező paramétereket.
5. A kontroller dinamikusan felépíti a megfelelő SQL lekérdezést (JOIN-okkal, WHERE feltételekkel, ORDER BY, LIMIT, OFFSET).
6. A kontroller (vagy a Perfume modell) lefuttatja az SQL lekérdezést az adatbázison a mysql driver segítségével. Lefuttat egy COUNT lekérdezést is a teljes találati számhoz.
7. Az adatbázis visszaadja a találati listát és a darabszámot.
8. A kontroller feldolgozza az eredményeket, kiszámolja a totalPages-t.
9. A kontroller visszaküldi a választ JSON formátumban a frontendnek (tartalmazza a parfümök listáját és a lapozási információkat).
10. A frontend (Search.js) fogadja a JSON választ.
11. A React frissíti a komponens állapotát (perfumes, totalPages, currentPage).
12. A React újrarendereli a felhasználói felületet, megjelenítve a keresési eredményeket (parfüm kártyákat) és a lapozót.

**Frontend Architektúra**

A frontend React komponensekből épül fel, amelyek pages és components könyvtárakba vannak szervezve.

**Struktúra:**

* **src/**
  + **App.js**: A fő alkalmazáskomponens, itt található a BrowserRouter, a Routes definíciók és az AuthContext.Provider.
  + **App.css**: Alapvető, globális stílusok.
  + **style.css**: Részletes, komponens-specifikus és reszponzív stílusok fő fájlja.
  + **pages/**: Az egyes oldalaknak megfelelő fő komponensek (Home.js, Search.js, PerfumeDetail.js, Login.js, Register.js, Profile.js, Favorites.js, Catalog.js, About.js, Aszf.js). Ezek általában több kisebb komponenst használnak.
  + **components/**: Újrafelhasználható UI komponensek (Navbar.js, Footer.js, PerfumeCard.js, Sidebar.js, ReviewForm.js, ReviewList.js, LoginPromptModal).
  + **services/**: Az API hívások logikáját tartalmazó modulok (authService.js, perfumeService.js, savedPerfumeService.js, reviewService.js, userService.js, stb.). Elkülönítik az adatlekérési logikát a komponensektől.

[ÁBRA HELYE: Frontend könyvtárstruktúra]

**Kulcs Komponensek Felelősségei:**

* **App.js:**
  + Beállítja a React Routert (BrowserRouter).
  + Definiálja az útvonalakat (Routes, Route), hozzárendelve az oldal komponenseket az URL-ekhez.
  + Körbeveszi az alkalmazást az AuthContext.Provider-rel, hogy a hitelesítési állapot elérhető legyen mindenhol.

const authContextValue = {

    isLoggedIn,

    user,

    token,

    login,

    logout,

  };

  return (

    <AuthContext.Provider value={authContextValue}>

      {" "}

      <Router future={{ v7\_startTransition: true, v7\_relativeSplatPath: true }}>

        <Navbar />

        <div className="container my-5">

          <Routes>

            <Route path="/" element={<Home />} />

            <Route path="/katalogus" element={<Catalog />} />

            <Route path="/kereses" element={<Search />} />

            <Route path="/rolunk" element={<About />} />

            {/\* Pass login function to Login component \*/}

            <Route path="/bejelentkezes" element={<Login />} />

            {/\* Pass login function to Register component \*/}

            <Route path="/regisztracio" element={<Register />} />

            {/\* Protected Route Example (adjust as needed) \*/}

            <Route

              path="/profil"

              element={isLoggedIn ? <Profile /> : <Login />}

            />

            <Route path="/parfume/:id" element={<PerfumeDetail />} />

            <Route path="/aszf" element={<Aszf />} />

            {/\* Protected Route for Favorites \*/}

            <Route

              path="/kedvencek"

              element={isLoggedIn ? <Favorites /> : <Login />}

            />

          </Routes>

        </div>

        <Footer />

      </Router>

    </AuthContext.Provider>

  );

}

* **Navbar.js:**
  + Megjeleníti a fő navigációs linkeket.
  + Dinamikusan változik a tartalom attól függően, hogy a felhasználó be van-e jelentkezve (isLoggedIn context alapján).
  + Megjeleníti a felhasználó nevét/profilképét vagy a Bejelentkezés/Regisztráció linkeket.
  + Tartalmazza a Kijelentkezés gombot.
  + Kezeli a mobil nézetben a hamburger menü nyitását/zárását (isOpen state).
* **Sidebar.js:**
  + A "Keresés" oldalon jelenik meg.
  + Tartalmazza a szűrési (Márka, Illatjegy, Nem) és rendezési (Név, Ár) opciók űrlap elemeit.
  + Lekéri a márkák és illatjegyek listáját a backendről (fetch).
  + Lehetővé teszi a legördülő listákban való keresést (filteredBrands, filteredNotes).
  + A "Szűrés és Rendezés" gombra kattintva frissíti az URL query paramétereit (useSearchParams, navigate), ami újratölti a Search.js adatlekérését.
  + Mobilon egy gombbal (filter-toggle) nyitható/csukható (isOpen state).
* **PerfumeCard.js:**
  + Egyetlen parfüm kártyáját jeleníti meg (kép, név, ár).
  + Linkként működik a parfüm részletes adatlapjára.
  + Tartalmazza a Kedvenc (szív) gombot, amelynek állapotát (isFavorite) a savedPerfumeService-en keresztül ellenőrzi és frissíti.

const handleToggleFavorite = async (e) => {

    e.preventDefault();

    e.stopPropagation();

    if (!isLoggedIn) {

      setShowLoginPrompt(true);

      return;

    }

    if (loadingFavorite || !id) return;

    setLoadingFavorite(true);

    try {

      let nowFavorite;

      if (isFavorite) {

        await removeFavorite(id);

        nowFavorite = false;

      } else {

        await addFavorite(id);

        nowFavorite = true;

      }

      setIsFavorite(nowFavorite);

      if (onFavoriteChange) {

        onFavoriteChange(id, nowFavorite);

      }

    } catch (error) {

      console.error(

        `PerfumeCard (${id}): Hiba a kedvencek kezelésekor:`,

        error

      );

      if (error && error.error === "Token érvénytelen!") {

        console.warn(

          `PerfumeCard (${id}): Invalid token detected on toggle. Logging out.`

        );

        logout();

      } else {

        alert(

          `Hiba: ${error.message || "Nem sikerült módosítani a kedvencet."}`

        );

      }

    } finally {

      setLoadingFavorite(false);

    }

  };

* **PerfumeDetail.js:**
  + Megjeleníti egy parfüm összes részletét (kép, adatok, boltok, értékelések).
  + Lekéri a parfüm adatait ID alapján (getPerfumeById service).
  + Kezeli a képnagyítást (isImageZoomed state).
  + Tartalmazza a Kedvenc gombot, hasonló logikával, mint a PerfumeCard.
  + Megjeleníti a ReviewForm (ha be van jelentkezve) és a ReviewList komponenseket.
  + Kezeli a bejelentkezési figyelmeztetést (showLoginPrompt state).
* **Search.js:**
  + A fő keresőoldal.
  + Rendereli a Sidebar-t.
  + Kezeli a keresőmező állapotát (searchTerm).
  + Lekéri a parfümöket a backendről a getAllPerfumes service segítségével, átadva az URL-ből (useSearchParams) és a state-ből származó keresési/szűrési/rendezési/lapozási paramétereket.
  + Megjeleníti a találatokat PerfumeCard komponensek segítségével.
  + Kezeli a lapozást (currentPage, totalPages, handlePageChange).
  + Megjeleníti a "Nincs találat" üzenetet.
* **Profile.js:**
  + Megjeleníti és szerkeszthetővé teszi a bejelentkezett felhasználó adatait.
  + Lekéri a felhasználói adatokat (getUser service) vagy a contextből veszi.
  + Kezeli a szerkesztési módot (editing state).
  + Validálja a bemeneti adatokat (validateForm).
  + Elküldi a frissített adatokat a backendnek (updateUser service).
  + Kezeli a frissítési és betöltési állapotokat (loading, updateLoading).

Kezeli a hiba- és sikerüzeneteket.  
  const handleEditSubmit = async (e) => {

    e.preventDefault();

    if (!validateForm()) return;

    setUpdateLoading(true);

    setUpdateError("");

    setUpdateSuccess("");

    setValidationErrors({});

    const dataToUpdate = {

name: newName.trim(),

      email: newEmail.trim(),

      phone: phoneNumber.trim() || null,

      profile\_picture\_url: imageUrl.trim() || null,

    };

    if (newPassword.trim()) dataToUpdate.password = newPassword.trim();

    try {

      console.log("[handleEditSubmit] Calling updateUser with:", dataToUpdate);

      const response = await updateUser(dataToUpdate);

      console.log(

        "[handleEditSubmit] Received response from updateUser:",

        response

      );

      if (response && response.success && response.user) {

        console.log("[handleEditSubmit] Update successful.");

        setUser(response.user);

        contextLogin(response.user, localStorage.getItem("token")); //

        initializeForm(response.user);

        setEditing(false);

        setUpdateSuccess("Adatok sikeresen frissítve!");

        setTimeout(() => setUpdateSuccess(""), 4000);

      } else {

        throw new Error(

          response?.error || "Sikertelen frissítés a szerver oldalon."

        );

      }

    } catch (err) {

      console.error("[handleEditSubmit] Error during update:", err);

      setUpdateError(err.message || "Hiba történt a mentés során.");

    } finally {

      console.log("[handleEditSubmit] Setting updateLoading to false.");

      setUpdateLoading(false);

    }

  };

* **Favorites.js:**
  + Lekéri a bejelentkezett felhasználó kedvenc parfümjeinek *részletes* adatait (getMyFavoritePerfumesDetails service). Ez a service először lekéri az ID-kat, majd egy batch kéréssel a parfüm adatokat.
  + Megjeleníti a kedvenceket PerfumeCard komponensekkel.
  + Kezeli a betöltési és hibaállapotokat.
  + Átirányít a bejelentkezési oldalra, ha a felhasználó nincs bejelentkezve.

**Állapotkezelés (State Management):**

* **Lokális State (useState):** A legtöbb komponens saját állapotot kezel (pl. űrlap adatok, betöltési állapotok, hibaüzenetek, UI állapotok mint isOpen, isImageZoomed).
* **Globális State (useContext - AuthContext):**
  + isLoggedIn: Boolean jelzi, hogy van-e bejelentkezett felhasználó.
  + user: Objektum a bejelentkezett felhasználó adataival (id, name, email, stb.).
  + token: A felhasználó JWT tokenje.
  + login(user, token): Függvény a bejelentkezési állapot és adatok beállítására (localStorage-ba is menti a tokent).
  + logout(): Függvény a kijelentkezéshez (törli az állapotot és a tokent a localStorage-ból).

rtékeket olvassák ki a contextből.  
export const AuthContext = createContext();

function App() {

  const [isLoggedIn, setIsLoggedIn] = useState(false);

  const [user, setUser] = useState(null);

  const [token, setToken] = useState(localStorage.getItem("token"));

  useEffect(() => {

    const storedToken = localStorage.getItem("token");

    const storedUser = localStorage.getItem("user");

    if (storedToken) {

      setToken(storedToken);

      setIsLoggedIn(true);

      if (storedUser) {

        try {

          setUser(JSON.parse(storedUser));

        } catch (e) {

          console.error("Failed to parse stored user data", e);

          logout();

        }

      }

    }

  }, []);

  const login = useCallback((userData, userToken) => {

    try {

      console.log("[Context] Updating context login:", userData);

      localStorage.setItem("token", userToken);

      localStorage.setItem("isLoggedIn", "true");

      if (userData) {

        localStorage.setItem("user", JSON.stringify(userData));

      } else {

        localStorage.removeItem("user");

      }

      setToken(userToken);

      setUser(userData);

      setIsLoggedIn(true);

    } catch (error) {

      console.error("[Context] Error during login update:", error);

    }

  }, []);

  const logout = useCallback(() => {

    localStorage.removeItem("token");

    localStorage.removeItem("isLoggedIn");

    localStorage.removeItem("user");

    localStorage.removeItem("username");

    localStorage.removeItem("email");

    setToken(null);

    setUser(null);

    setIsLoggedIn(false);

  }, []);

**API Hívások:**

* A services/ könyvtárban lévő fájlok felelősek az Axios segítségével történő API hívásokért.
* Minden service funkció általában egy specifikus API végpontot hív meg.
* Kezelik a kérések összeállítását (paraméterek, body, headerek - pl. x-auth-token hozzáadása a getAuthConfig segítségével).
* Feldolgozzák a választ vagy a hibát, és visszaadják a komponensnek.

**Backend Architektúra**

A backend egy Node.js/Express alkalmazás, amely RESTful API-t biztosít a frontend számára.

* **Struktúra:**
* server.js / app.js: A fő belépési pont. Inicializálja az Express alkalmazást, beállítja a middleware-eket (CORS, body-parser), betölti a környezeti változókat (dotenv), csatlakozik az adatbázishoz (db.js), regisztrálja az útvonalakat (routes), és elindítja a szervert. Tartalmaz egy általános hibakezelő middleware-t is.
* db.js: (Hiányzik) Beállítja és exportálja a MySQL adatbázis-kapcsolatot (connection pool ajánlott). Lehet, hogy tartalmaz egy queryAsync segédfüggvényt is a Promise-alapú lekérdezésekhez.
* routes/: Minden fő erőforráshoz (auth, perfumes, users, reviews, stb.) külön útvonal fájl tartozik. Ezek definiálják az API végpontokat (URL-ek és HTTP metódusok) és hozzárendelik őket a megfelelő kontroller funkciókhoz. Alkalmazzák az authMiddleware-t a védett útvonalakra.
* router.get("/batch", PerfumeController.getPerfumesByIds);
* console.log("--- perfumeRoutes.js loaded ---");
* console.log("Middleware added (if any)");
* router.get("/price-range", (req, res, next) => {
* console.log(">>> GET /price-range route matched <<<");
* PerfumeController.getPriceRange(req, res, next);
* });
* router.get("/all", PerfumeController.getAllPerfumes);
* router.get("/random", PerfumeController.getRandomPerfumes);
* router.get("/featured", PerfumeController.getFeaturedPerfumes);
* router.get("/:id", PerfumeController.getPerfumeById);
* router.get("/price-range", PerfumeController.getPriceRange);
* controllers/: Tartalmazzák az üzleti logikát. Minden kontroller funkció egy adott útvonalhoz kapcsolódik. Feldolgozzák a req (kérés) objektumot (paraméterek, query string, body), meghívják a megfelelő modell funkciókat az adatbázis-műveletekhez, és a res (válasz) objektum segítségével visszaküldik a választ (JSON adat vagy hiba).
* exports.getPerfumeById = (req, res, next) => {
* const perfumeIdParam = req.params.id;
* const perfumeId = parseInt(perfumeIdParam, 10);
* if (isNaN(perfumeId) || perfumeId <= 0) {
* console.warn(`Invalid perfume ID requested: ${perfumeIdParam}`);
* return res.status(400).json({ error: "Érvénytelen parfüm azonosító." });
* }
* const perfumeSql = `
* SELECT
* p.\*,
* b.name AS brand\_name,
* GROUP\_CONCAT(DISTINCT nt.name SEPARATOR ', ') AS notes /\* nt alias a notes táblának \*/
* FROM perfumes p
* LEFT JOIN brands b ON p.brand\_id = b.id
* LEFT JOIN perfume\_notes pn ON p.id = pn.perfume\_id
* LEFT JOIN notes nt ON pn.note\_id = nt.id /\* nt alias használata \*/
* WHERE p.id = ?
* GROUP BY p.id
* `;
* const storesSql = `
* SELECT store\_name, url, price, currency
* FROM stores
* WHERE perfume\_id = ?
* ORDER BY price ASC
* `;
* db.query(perfumeSql, [perfumeId], (err, perfumeResult) => {
* if (err) {
* console.error("SQL Hiba (parfüm):", err);
* return next(err);
* }
* if (perfumeResult.length === 0) {
* return res
* .status(404)
* .json({ error: "Parfüm nem található ezzel az azonosítóval!" });
* }
* db.query(storesSql, [perfumeId], (storeErr, storeResult) => {
* if (storeErr) {
* console.error("SQL Hiba (üzletek):", storeErr);
* return next(storeErr);
* }
* const perfumeData = {
* ...perfumeResult[0],
* notes: perfumeResult[0].notes ? perfumeResult[0].notes.split(", ") : [],
* stores: storeResult || [],
* };
* res.status(200).json(perfumeData);
* });
* });
* };
* logikája]
* models/: (Hiányzik) Az adatbázis-interakciók logikáját tartalmazzák. Minden modell egy adatbázis-táblának felel meg. Funkciókat exportálnak az adatok lekérdezésére (SELECT), beszúrására (INSERT), frissítésére (UPDATE) és törlésére (DELETE). Ideális esetben ezek a funkciók Promise-okat adnak vissza. Itt történik a jelszó hashelés (User.createUser, User.updateUser) és az SQL lekérdezések összeállítása.
* const getUserByEmail = async (email) => {
* const results = await queryAsync("SELECT \* FROM users WHERE email = ?", [
* email,
* ]);
* return results.length > 0 ? results[0] : null;
* };
* const createUser = async (user) => {
* try {
* const hash = await bcrypt.hash(user.password, 10);
* const newUser = { ...user, password: hash };
* const { name, email, password, phone, profile\_picture\_url, is\_admin } =
* newUser;
* const result = await queryAsync(
* "INSERT INTO users (name, email, password, phone, profile\_picture\_url, is\_admin) VALUES (?, ?, ?, ?, ?, ?)",
* [
* name,
* email,
* password,
* phone || null,
* profile\_picture\_url || null,
* is\_admin || 0,
* ]
* );
* return { insertId: result.insertId };
* } catch (error) {
* if (error.code === "ER\_DUP\_ENTRY") {
* throw new Error("Ez az email cím már foglalt!");
* }
* throw error;
* }
* };
* middleware/:
* authMiddleware.js: (Hiányzik) Ellenőrzi a bejövő kérések x-auth-token headerét. Verifikálja a JWT tokent a jsonwebtoken és a JWT\_SECRET segítségével. Ha a token érvényes, dekódolja a payload-ot (ami tartalmazza a userId-t) és hozzáadja a req.user objektumhoz (pl. req.user = { id: decoded.id }). Ha a token érvénytelen vagy hiányzik, 401-es (Unauthorized) hibával válaszol.
* const authMiddleware = (req, res, next) => {
* const token = req.header('x-auth-token');
* if (!token) return res.status(401).json({ error: 'Nincs hitelesítési token!' });
* try {
* const decoded = jwt.verify(token, process.env.JWT\_SECRET);
* req.user = { id: decoded.id };
* next();
* } catch (error) {
* res.status(401).json({ error: 'Token érvénytelen!' });
* }
* };
* **Hibakezelő Middleware**: Az app.js végén található. Elkapja az útvonal kezelőkben vagy middleware-ekben keletkezett hibákat (next(err) hívással), naplózza őket, és egységes JSON hibaüzenetet küld vissza a kliensnek.
* app.use((err, req, res, next) => {
* console.error(
* "!!! Unhandled Error Caught by Central Handler:",
* err.stack || err
* );
* const statusCode = err.statusCode || 500;
* const errorMessage = err.message || "Ismeretlen szerverhiba.";
* res.status(statusCode).json({
* error: errorMessage,
* });
* });

**Adatforrások**

Az alkalmazás több forrásból szerzi be az adatait:

1. MySQL Adatbázis:

* Ez a fő adatforrás.
* Itt tárolódnak a felhasználói fiókok, parfümök alapadatai (név, márka, nem, leírás, kép URL), márkák, illatjegyek, az összerendelésük (perfume\_notes), felhasználói értékelések (reviews) és a kedvencek (saved\_perfumes).
* Notino Affiliate Adatok: Az adatbázisban található parfümök és a hozzájuk tartozó bolti (stores) adatok (URL, ár) a Notino partnerprogramjából származnak. Ezeket az adatokat feltételezhetően egy külön folyamat (pl. egy script, vagy manuális import) töltötte be az adatbázisba a projekt indulásakor vagy rendszeres időközönként (bár a frissítés logikája nem része a jelenlegi kódnak). Az alkalmazás nem élőben hívja a Notino API-ját, hanem a saját adatbázisából dolgozik. Ez fontos korlátja az ár-összehasonlító funkciónak, mivel csak egy forrás adatai érhetők el.

1. Frontend (Felhasználói Bevitel):

* Regisztrációs és profiladatok.
* Keresési és szűrési feltételek.
* Értékelések adatai (pontszámok, szöveg).
* Kedvencek jelölése.

1. NewsAPI (Külső API):

* A "Hírek" oldal (Catalog.js) közvetlenül a NewsAPI szolgáltatást hívja meg egy API kulcs segítségével, hogy parfümökkel kapcsolatos híreket kérjen le.
* Ez az egyetlen olyan adatforrás, amely valós időben, külső szolgáltatótól érkezik a frontend oldalon.
* Biztonsági megjegyzés: Az API kulcs közvetlen beágyazása a frontend kódba nem biztonságos gyakorlat.

## Tesztelési dokumentáció

**Frontend tesztelés**

* **Böngésző Kompatibilitás:** Manuális tesztelés a leggyakrabban használt böngészők legfrissebb verzióiban (Chrome, Firefox, Edge). Fókusz a megjelenítésbeli különbségekre és a JavaScript funkcionalitás helyességére.
* **Felhasználói Felület (UI) Tesztelés:** Az elemek (gombok, űrlapok, képek, linkek) vizuális ellenőrzése, olvashatóság, konzisztens megjelenés biztosítása. Hover effektek, aktív állapotok ellenőrzése.
* **Funkcionális Tesztelés:** A 3.4.2 pontban leírt tesztesetek végrehajtása a felhasználói felületen keresztül. Interakciók (kattintások, gépelés, választás) tesztelése.
* **Reszponzivitás Tesztelés:** Az oldal megjelenésének és használhatóságának ellenőrzése különböző képernyőméreteken és orientációkon (álló/fekvő) a böngésző fejlesztői eszközeivel (Device Mode) és lehetőség szerint valós eszközökön. Különös figyelem a mobil menüre, a grid rendszerek töréspontjaira és az elemek átfedésére/szétcsúszására.
* **Hibatűrés:** Mi történik, ha az API hívás sikertelen? Megjelenik-e a hibaüzenet a felhasználónak? Az alkalmazás stabil marad-e? (Pl. API szerver leállításával tesztelhető).

# Összefoglalás

## Önértékelés

**Név:** Laczy Miklós

Ebben a projektben én feleltem a Parfümvilág weboldal kinézetéért és működéséért, vagyis a frontend részért. A célom az volt, hogy egy könnyen használható, jól kinéző oldalt készítsek, ami a háttérben futó rendszerrel (backend) is jól együttműködik.

**1. Célok Elérése**

Szerintem a kitűzött célokat sikerült elérni. A weboldal fő részei elkészültek:

Tervünk egy parfümös oldal létrehozása volt, ahol a vélemény írásokkal és azzal, hogy profilt tudunk létrehozni egy kisebb közösséget hozzunk létre, illetve minden értékelés személyesebb legyen.

Ezt szerintünk sikerült is megvalósítani, az oldal többi funkciói is hibátlanul működnek.

**2. Nehézségek és Megoldások**

Voltak azért kihívások is a munka során:

Reszponzívivás megoldása, dizájnok létrehozása, minden a megfelelő helyen legyen. Verziók nyomon követése.

**3. Amit Tanultam**

Csapatmunka, idő megfelelő beosztása, frontend formázások, kapcsolatok létrehozása a backenddel.

Összességében elégedett vagyok az eredménnyel, mert a weboldal működik és lefedi a fő funkciókat. Sokat tanultam a modern webfejlesztésről, és bár vannak még területek (mint a CSS jobb szervezése vagy az automatikus tesztelés), ahol fejlődhetek, hasznos tapasztalatokat szereztem.

Összességében jónak értékelem a munkám és a munkánkat, ez alatt az idő alatt fejlődtem a legtöbbet és tapasztaltam a legtöbbet.

**Név:** Kertész Márton

A Parfümvilág projekt során kitűzött célunk egy olyan webes alkalmazás létrehozása volt, amely lehetővé teszi a felhasználók számára parfümök keresését, szűrését és értékelését.

**1. Célok Elérése**

Hatékony keresési rendszer : A felhasználóknak lehetőségük van parfümök szűrésére márkák, illatok, ár és név alapján. Ezt a célt sikeresen elértem azáltal, hogy a backend oldalon optimalizált SQL lekérdezéseket implementáltam, amelyek biztosítják a gyors válaszidőt még nagy adathalmaz esetén is.

**2. Nehézségek és Megoldások**  
A fejlesztés során több kihívással is szembesültem:

Duplikált adatok a keresési eredmények között : A parfümök több illatjegyet tartalmazhattak, ami miatt ugyanaz a parfüm többször is megjelent a listában. Ezt a problémát a DISTINCT kulcsszó és az adatbázis normalizálásával oldottam meg.

Lapozás hibái : A frontend és a backend közötti kommunikáció során előfordult, hogy a lapozás nem működött megfelelően. A problémát úgy oldottam meg, hogy pontosabb paraméterezést vezettem be a backend API-ban, és a frontend oldalon is ellenőriztem a kapott adatok helyességét.

**3. Amit Tanultam**  
Ezen a projekt során számos új készséget szereztem és alkalmaztam:

Backend fejlesztés : Az Express.js és MySQL használatával mélyebben megismertem a RESTful API-k kialakítását és az adatbázis-kezelés finomságait. Például megtanultam, hogyan kell hatékonyan írni összetett SQL lekérdezéseket, és hogyan kell kezelni a különböző HTTP kéréseket.

 Csapatmunka és kommunikáció : A projekt során fontos szerepet játszott a frontend és backend fejlesztők közötti együttműködés. A rendszeres egyeztetések és a GitHub használata segített abban, hogy a munka zökkenőmentesen haladjon.

## Továbbfejlesztési lehetőségek

1. **Valódi Árösszehasonlítás:** További webáruházak (pl. Douglas, Marionnaud, kisebb hazai webshopok) API-jának az integrálása
2. **Adminisztrációs Felület:** Egy külön felület létrehozása adminisztrátorok számára, ahol kezelhetik a parfümöket, márkákat, illatjegyeket, felhasználókat, értékeléseket és a kiemelt ajánlatokat.
3. **Ajánló Rendszer:** Felhasználói preferenciák (kedvencek, értékelések, keresési előzmények) alapján személyre szabott parfüm ajánlások implementálása.
4. **Részletesebb Felhasználói Profil:** További profiladatok (pl. kedvenc márkák, illatjegyek, profil leírás) hozzáadása, felhasználók által feltöltött profilkép kezelése (nem csak URL).
5. **Közösségi Funkciók:** Felhasználók közötti interakciók (pl. kommentelés értékelésekhez, felhasználói listák megosztása).
6. **Frontend Refaktorálás:**

* **CSS:** CSS Modules, Styled Components vagy Tailwind CSS bevezetése a jobb stíluskezelés és karbantarthatóság érdekében. A style.css felbontása kisebb, komponens-specifikus fájlokra.

1. **Backend Optimalizálás:**

* **SQL Optimalizálás:** A komplex lekérdezések (főleg getAllPerfumes) teljesítményének elemzése és optimalizálása (indexelés, lekérdezés átírása). ORDER BY RAND() helyettesítése hatékonyabb megoldással.
* **Modell Réteg Egységesítése:** Minden modell funkció átírása Promise-alapú (async/await) működésre.
* **Validáció:** Robusztusabb validációs réteg bevezetése (express-validator).

1. **Automatizált Tesztelés:** Unit tesztek (Jest, React Testing Library a frontendhez; Jest a backendhez) és integrációs tesztek (Supertest a backend API-hoz) írása a kód minőségének és megbízhatóságának növelése érdekében. End-to-end tesztek (Cypress, Playwright) a fő felhasználói folyamatokhoz.
2. **Biztonság:**

* NewsAPI kulcs áthelyezése a backendre (proxy végpont létrehozása).
* Rate limiting bevezetése.
* helmet middleware használata.
* Alaposabb input szanitizálás.

1. **API Dokumentáció:** Swagger/OpenAPI dokumentáció generálása a backend API-hoz.
2. **Deployment:** Az alkalmazás "élesítése" egy felhőszolgáltatóra (pl. Heroku, Vercel, Netlify a frontendhez; Heroku, Render, AWS/GCP/Azure a backendhez és adatbázishoz).

# Felhasznált irodalom

* React Hivatalos Dokumentáció: [https://reactjs.org/docs/getting-started.html](https://www.google.com/url?sa=E&q=https%3A%2F%2Freactjs.org%2Fdocs%2Fgetting-started.html)
* Node.js Hivatalos Dokumentáció: [https://nodejs.org/en/docs/](https://www.google.com/url?sa=E&q=https%3A%2F%2Fnodejs.org%2Fen%2Fdocs%2F)
* Express.js Hivatalos Weboldal: [https://expressjs.com/](https://www.google.com/url?sa=E&q=https%3A%2F%2Fexpressjs.com%2F)
* MySQL Hivatalos Dokumentáció: [https://dev.mysql.com/doc/](https://www.google.com/url?sa=E&q=https%3A%2F%2Fdev.mysql.com%2Fdoc%2F)
* MDN Web Docs (HTML, CSS, JavaScript): [https://developer.mozilla.org/](https://www.google.com/url?sa=E&q=https%3A%2F%2Fdeveloper.mozilla.org%2F)
* Axios Dokumentáció: [https://axios-http.com/docs/intro](https://www.google.com/url?sa=E&q=https%3A%2F%2Faxios-http.com%2Fdocs%2Fintro)
* React Router DOM Dokumentáció: [https://reactrouter.com/en/main](https://www.google.com/url?sa=E&q=https%3A%2F%2Freactrouter.com%2Fen%2Fmain)
* JSON Web Tokens (JWT) Bevezetés: [https://jwt.io/introduction](https://www.google.com/url?sa=E&q=https%3A%2F%2Fjwt.io%2Fintroduction)
* bcryptjs npm oldal: [https://www.npmjs.com/package/bcryptjs](https://www.google.com/url?sa=E&q=https%3A%2F%2Fwww.npmjs.com%2Fpackage%2Fbcryptjs)
* NewsAPI Dokumentáció: [https://newsapi.org/docs](https://www.google.com/url?sa=E&q=https%3A%2F%2Fnewsapi.org%2Fdocs)

.