**Szegedi SZC Vasvári Pál Gazdasági és Informatikai Technikum**

**Az 54 481 02 számú Gazdasági informatikus szakképesítés záródolgozata  
  
  
  
  
  
  
Shoply  
Intelligens bevásárlólista kezelő és termék katalógus rendszer**

**Készítette:**

**Lajkó Levente, Takács András, Kopincu Máté**

**Szeged  
2/14.be szoft (2024-25)**

Tartalomjegyzék

[Felhasználói dokumentáció 3](#__RefHeading___Toc800_1675044539)

[A program általános specifikációja 3](#__RefHeading___Toc802_1675044539)

[Többplatformos támogatás 3](#__RefHeading___Toc804_1675044539)

[Főbb funkciók 3](#__RefHeading___Toc806_1675044539)

[Technikai követelmények 4](#__RefHeading___Toc808_1675044539)

[Biztonság és adatvédelem 4](#__RefHeading___Toc810_1675044539)

[Rendszerkövetelmények 5](#__RefHeading___Toc812_1675044539)

[Hardver követelmények 5](#__RefHeading___Toc814_1675044539)

[Minimális konfiguráció: 5](#__RefHeading___Toc816_1675044539)

[Ajánlott konfiguráció: 5](#__RefHeading___Toc818_1675044539)

[Szoftver követelmények 5](#__RefHeading___Toc820_1675044539)

[Operációs rendszerek: 5](#__RefHeading___Toc822_1675044539)

[Szükséges szoftverkomponensek: 5](#__RefHeading___Toc824_1675044539)

[A program telepítése 6](#__RefHeading___Toc826_1675044539)

[Webes felület telepítése 6](#__RefHeading___Toc828_1675044539)

[Asztali alkalmazás telepítése 6](#__RefHeading___Toc830_1675044539)

[Hibaelhárítás 7](#__RefHeading___Toc834_1675044539)

[Fejlesztői Dokumentáció 8](#__RefHeading___Toc836_1675044539)

[1. A téma választásának oka 8](#__RefHeading___Toc3109_1675044539)

[2. Az alkalmazott fejlesztői eszközök 9](#__RefHeading___Toc840_1675044539)

[Fejlesztői környezetek és eszközök 9](#__RefHeading___Toc842_1675044539)

[Külső modulok és könyvtárak 11](#__RefHeading___Toc2698_1675044539)

[3. Az adatmodell leírása 12](#__RefHeading___Toc846_1675044539)

[Kapcsolatok és relációk 12](#__RefHeading___Toc848_1675044539)

[Adatbázis indexek 12](#__RefHeading___Toc850_1675044539)

[4. A program architektúrája 14](#__RefHeading___Toc854_1675044539)

[4.1. Rendszerarchitektúra 14](#__RefHeading___Toc856_1675044539)

[4.1.1. Frontend architektúra 14](#__RefHeading___Toc858_1675044539)

[4.1.2. Backend architektúra 14](#__RefHeading___Toc860_1675044539)

[4.1.3. Asztali alkalmazás architektúra 14](#__RefHeading___Toc862_1675044539)

[4.2. Kommunikációs architektúra 14](#__RefHeading___Toc864_1675044539)

[4.2.1. API Kommunikáció 14](#__RefHeading___Toc866_1675044539)

[4.3. Használati forgatókönyvek és folyamatábrák (UML) 16](#__RefHeading___Toc2678_3334749236)

[4.3.1. Asztali alkalmazás forgatókönyvek 16](#__RefHeading___Toc2680_3334749236)

[4.3.2. Mobil alkalmazás forgatókönyvek 19](#__RefHeading___Toc2682_3334749236)

[4.3.3. Webes alkalmazás forgatókönyvek 22](#__RefHeading___Toc2684_3334749236)

[5. A program működésének leírása 25](#__RefHeading___Toc876_1675044539)

[5.1. Adatfolyamok 25](#__RefHeading___Toc878_1675044539)

[5.2. Hibakezelés 26](#__RefHeading___Toc884_1675044539)

[6. A program tesztelése 27](#__RefHeading___Toc890_1675044539)

[7. A program forráskódjának dokumentálása 29](#__RefHeading___Toc892_1675044539)

[7.1. Kommentek használata 29](#__RefHeading___Toc894_1675044539)

[7.2. JSDoc dokumentáció 29](#__RefHeading___Toc896_1675044539)

[7.3. API dokumentáció 30](#__RefHeading___Toc898_1675044539)

[7.4. Teszt dokumentáció 30](#__RefHeading___Toc900_1675044539)

[7.5. Kód dokumentációs eszközök 31](#__RefHeading___Toc902_1675044539)

[8. Összefoglalás és jövőbeli fejlesztési lehetőségek 32](#__RefHeading___Toc906_1675044539)

[8.1. A projekt jelenlegi állapota 32](#__RefHeading___Toc908_1675044539)

[8.2. Jövőbeli fejlesztési lehetőségek 32](#__RefHeading___Toc910_1675044539)

[8.3. Záró gondolatok 33](#__RefHeading___Toc916_1675044539)

# Felhasználói dokumentáció

## A program általános specifikációja

A Shoply alkalmazás egy többplatformos megoldás, amely segítségével a felhasználók könnyedén kezelhetik és szinkronizálhatják vásárlólistáikat. Az alkalmazás a következő főbb jellemzőkkel és funkciókkal rendelkezik:

### Többplatformos támogatás

* Webes felület: Modern, reszponzív webalkalmazás, amely bármely böngészőből elérhető
* Asztali alkalmazás: Natív Windows alkalmazás a kényelmes használathoz
* Mobil alkalmazás: Később tervezett mobil verzió iOS és Android platformokra

### Főbb funkciók

* Vásárlólisták kezelése:
  + Új listák létrehozása
  + Meglévő listák szerkesztése
  + Listák törlése
  + Termékek hozzáadása és eltávolítása
  + Mennyiségek és egységek beállítása
* Felhasználói fiókok:
  + Regisztráció és bejelentkezés
  + Személyes beállítások kezelése
  + Adatok biztonságos tárolása
* Szinkronizáció:
  + Valós idejű szinkronizáció az összes eszköz között
  + Offline mód támogatása
  + Automatikus adatbiztonsági mentések
* Egyéb funkciók:
  + Kategóriák szerinti rendezés
  + Keresési lehetőségek
  + Megosztási opciók
  + Értesítések és emlékeztetők

### Technikai követelmények

* Webes felület:
  + Modern webböngésző (Chrome, Firefox, Edge, Safari)
  + Internet kapcsolat (offline mód esetén nem szükséges)
* Asztali alkalmazás:
  + Windows 10 vagy újabb operációs rendszer
  + Minimum 2GB RAM
  + 500MB szabad tárhely

### Biztonság és adatvédelem

* Titkosított adattárolás
* Biztonságos hitelesítés
* Adatvédelem és GDPR megfelelőség
* Rendszeres biztonsági frissítések

Az alkalmazás célja, hogy egyszerű és hatékony megoldást nyújtson a vásárlólisták kezelésére, miközben biztosítja az adatok biztonságát és a felhasználói élményt. A többplatformos megközelítés lehetővé teszi, hogy a felhasználók bármikor, bárhonnan hozzáférhessenek listáikhoz, és szükség esetén megoszthassák azokat másokkal.

# Rendszerkövetelmények

## Hardver követelmények

### Minimális konfiguráció:

* Processzor: Intel Core i3 vagy AMD Ryzen 3 (1.8 GHz vagy gyorsabb)
* RAM: 2 GB
* Tárhely: 500 MB szabad hely

### Ajánlott konfiguráció:

* Processzor: Intel Core i5 vagy AMD Ryzen 5 (2.4 GHz vagy gyorsabb)
* RAM: 4 GB vagy több
* Tárhely: 1 GB szabad hely
* Kijelző: 1920x1080 felbontás
* Internet kapcsolat: 10 Mbps letöltési sebesség

## Szoftver követelmények

### Operációs rendszerek:

* Windows: 10 vagy újabb verzió (64 bites)
* Webes felület: Bármely modern böngésző (Chrome 90+, Firefox 90+, Edge 90+, Safari 14+)

### Szükséges szoftverkomponensek:

* Node.js: 16.x vagy újabb verzió (a backend működéséhez)
* MongoDB: 4.4 vagy újabb verzió (adatbázis szerver)
* .NET Framework: 6.0 vagy újabb (az asztali alkalmazáshoz)
* Docker: 20.10 vagy újabb verzió (opcionális, konténeres futtatáshoz)

# A program telepítése

## Webes felület telepítése

1. Előfeltételek ellenőrzése
   * Ellenőrizze, hogy a böngészője naprakész-e
   * Győződjön meg róla, hogy van internet kapcsolata
2. Regisztráció és bejelentkezés
   * Látogasson el a https://vasarlolista.hu oldalra
   * Kattintson a "Regisztráció" gombra
   * Töltse ki a regisztrációs űrlapot
   * Erősítse meg az email címét
   * Jelentkezzen be az új fiókjával

## Asztali alkalmazás telepítése

1. Telepítő letöltése
   * Látogasson el a https://vasarlolista.hu/letoltes oldalra
   * Válassza ki az operációs rendszerének megfelelő verziót
   * Kattintson a "Letöltés" gombra
2. Telepítés indítása
   * Nyissa meg a letöltött telepítőfájlt (Vasarlolista\_Setup.exe)
   * Ha megjelenik a felhasználói fiók vezérlés párbeszédpanel, kattintson az "Igen" gombra
3. Telepítési beállítások
   * Válassza ki a telepítési nyelvet (magyar)
   * Olvassa el és fogadja el a licencszerződést
   * Válassza ki a telepítési könyvtárat (alapértelmezetten: C:\Program Files\Vasarlolista)
   * Válassza ki a telepítendő komponenseket:
   * Asztali alkalmazás (kötelező)
   * Start menü parancsikon (ajánlott)
   * Asztali parancsikon (ajánlott)
   * Automatikus frissítések (ajánlott)
4. Telepítés befejezése
   * Kattintson a "Telepítés" gombra
   * Várja meg a telepítés befejezését
   * Kattintson a "Befejezés" gombra
   * Az alkalmazás automatikusan elindul

## Hibaelhárítás

Ha problémák merülnek fel a telepítés során:

1. Webes felület esetén:
   * Törölje a böngésző gyorsítótárát
   * Próbálja meg másik böngészőt használni
   * Ellenőrizze az internet kapcsolatot
2. Asztali alkalmazás esetén:
   * Ellenőrizze, hogy megfelel-e a rendszer a minimális követelményeknek
   * Indítsa újra a számítógépet
   * Ha szükséges, telepítse újra a .NET Framework-ot
3. Mobil alkalmazás esetén:
   * Ellenőrizze, hogy van-e elegendő tárhely
   * Frissítse az Android rendszert
   * Törölje és telepítse újra az alkalmazást

# Fejlesztői Dokumentáció

## 1. A téma választásának oka

A Shoply alkalmazás készítését több ok is alátámasztja. A mai gyors élet és a digitális világ miatt egyre többen keresnek olyan megoldásokat, amik könnyebbé és gyorsabbá teszik a mindennapi dolgokat. A bevásárló lista intézése is ilyen terület, ahol a papír helyett egyre többen akarják digitálisan csinálni.

A projekt témájának választását főként a következő okok indokolták:

1. **Piac igénye:** A bevásárló lista egy olyan dolog, amire mindenkinek szüksége van, akár egyedül él, akár családja van, akár egy kisebb baráti kör. A mostani appok vagy túl egyszerűek, vagy túl bonyolultak, és sokszor nem lehet őket sokféle telefonon vagy gépen használni.
2. **Műszaki kihívás:** Ez a projekt ad egy lehetőséget, hogy modern fejlesztési módokat és új technikákat próbáljunk ki. Például, hogy az app működjön a neten, asztali gépen és mobilon is. Azt is meg akarjuk oldani, hogy a lista azonnal frissüljön, ha valaki változtat rajta, és hogy akkor is lehessen nézni a listát, ha nincs internet. Ezek nehéz feladatok, de ha sikerül megcsinálni, sokat tanulunk belőle.
3. **Lehet továbbfejleszteni:** Az app úgy van kitalálva, hogy később könnyen hozzá lehessen adni új dolgokat. Például recept kezelését, okos javaslatot, hogy mit vegyen, vagy hogy a családtag könnyen meg tudja osztani a listát.
4. **Új dolgot próbálhatunk ki:** Az app tervezésekor kipróbálhatunk új technikákat és megoldásokat. Például mesterséges értelemmel elemezhetjük, hogy mit szokott venni, vagy blockchain technikával biztonságosan kezelhetjük a megosztott listákat.
5. **Jó a környezetnek és a társadalomnak:** Ha digitális bevásárló listát használ valaki, kevesebb papírt pazarol, ami jó a környezetnek. Ráadásul, ha meg tudja osztani a listát, az segíthet a családtagoknak és barátoknak, hogy jobban tudjanak együttműködni.

A projekt célja nem csak az, hogy egy működő appot csináljunk, hanem hogy egy olyan új megoldást hozzunk létre, ami kihasználja a mai technikai lehetőségeket és tényleg hasznos az ember számára. Emellett mi, fejlesztőként is sokat tanulhatunk és kipróbálhatunk új dolgokat.

## 2. Az alkalmazott fejlesztői eszközök

### Fejlesztői környezetek és eszközök

Programozási nyelvek és keretrendszerek:

* Frontend (Web):
  + TypeScript
  + React.js
  + Next.js
  + Tailwind CSS
  + Material-UI komponenskönyvtár
* Backend:
  + Node.js
  + Express.js
  + TypeScript
  + MongoDB (adatbázis)
  + Mongoose (ODM)
* Asztali alkalmazás:
  + .NET 6.0
  + C#
  + WPF (Windows Presentation Foundation)

Fejlesztői környezetek (IDE-k):

* Visual Studio Code:
  + Webes és backend fejlesztéshez
  + Használt bővítmények:
    - ESLint
    - Prettier
    - TypeScript és JavaScript IntelliSense
    - GitLens
    - Docker
* Visual Studio 2022:
  + Asztali alkalmazás fejlesztéséhez
  + .NET fejlesztői eszközök

Verziókezelés és együttműködés:

* Git verziókezelő rendszer
* GitHub tároló és együttműködési platform
* GitHub Actions CI/CD folyamatokhoz

Adatbázis és szerver:

* MongoDB NoSQL adatbázis
* Docker konténerizáció
* Docker Compose több konténer koordinálásához

Tesztelési eszközök:

* Jest unit és integrációs tesztekhez
* React Testing Library frontend komponens tesztekhez
* Postman API teszteléshez

Dokumentáció és kommunikáció:

* Markdown dokumentáció írásához
* Draw.io diagramok és folyamatábrák készítéséhez
* Microsoft Word végső dokumentáció formázásához

### Külső modulok és könyvtárak

Frontend:

* react-router-dom: Útvonalak kezeléséhez
* axios: HTTP kérések kezeléséhez
* react-query: Állapotkezelés és adatlekérdezés
* socket.io-client: Valós idejű kommunikáció

Backend:

* jsonwebtoken: JWT token kezelés
* bcryptjs: Jelszó titkosítás
* cors: Cross-Origin Resource Sharing
* dotenv: Környezeti változók kezelése

Asztali alkalmazás:

* MaterialDesignInXAML: Modern UI komponensek
* Newtonsoft.Json: JSON kezelés
* MongoDB.Driver: Adatbázis kapcsolat

Minden külső modul és könyvtár megfelel a nyílt forráskódú licenc feltételeinek, és a projekt package.json és .csproj fájljainak függőségei között dokumentálva van. A használt komponensek verziószámai és licenc feltételei a projekt gyökérkönyvtárában található LICENSE és README.md fájlokban szerepelnek.

## 3. Az adatmodell leírása

### Kapcsolatok és relációk

A rendszer MongoDB NoSQL adatbázist használ, amely lehetővé teszi a rugalmas adatszerkezet kezelését és a gyors fejlesztést.Kapcsolatok és relációk

1. User - ShoppingList:
   * Egy felhasználó több listát birtokolhat (1:N)
   * Egy lista egy tulajdonossal rendelkezik (1:1)
   * Egy lista több felhasználóval megosztható (N:M)
2. User - Category:
   * Egy felhasználó több kategóriát hozhat létre (1:N)
   * Egy kategória egy felhasználóhoz tartozik (1:1)
3. ShoppingList - Category:
   * Egy lista több kategóriát tartalmazhat (1:N)
   * Egy kategória több listában is szerepelhet (N:M)
4. User - Notification:
   * Egy felhasználó több értesítést kaphat (1:N)
   * Egy értesítés egy felhasználóhoz tartozik (1:1)

### Adatbázis indexek

1. User tábla:
   * username (egyedi)
   * email (egyedi)
   * createdAt
2. ShoppingList tábla:
   * owner
   * sharedWith.user
   * createdAt
   * updatedAt
3. Category tábla:
   * createdBy
   * name (egyedi per user)
4. Notification tábla:
   * userId
   * createdAt
   * read

## 4. A program architektúrája

### 4.1. Rendszerarchitektúra

A rendszer három fő komponensből áll:

1. Frontend (Web alkalmazás)
2. Backend (API szerver)
3. Asztali alkalmazás

### 4.1.1. Frontend architektúra

A webes felület React.js és Next.js keretrendszereken alapul, amelyek lehetővé teszik:

* Server Side Rendering (SSR)
* Statikus oldal generálás (SSG)
* Inkrementális statikus regenerálás (ISR)

### 4.1.2. Backend architektúra

A backend Node.js és Express.js keretrendszereken alapul, RESTful API-t biztosítva

### 4.1.3. Asztali alkalmazás architektúra

A WPF alkalmazás MVVM (Model-View-ViewModel) mintát követ

### 4.2. Kommunikációs architektúra

### 4.2.1. API Kommunikáció

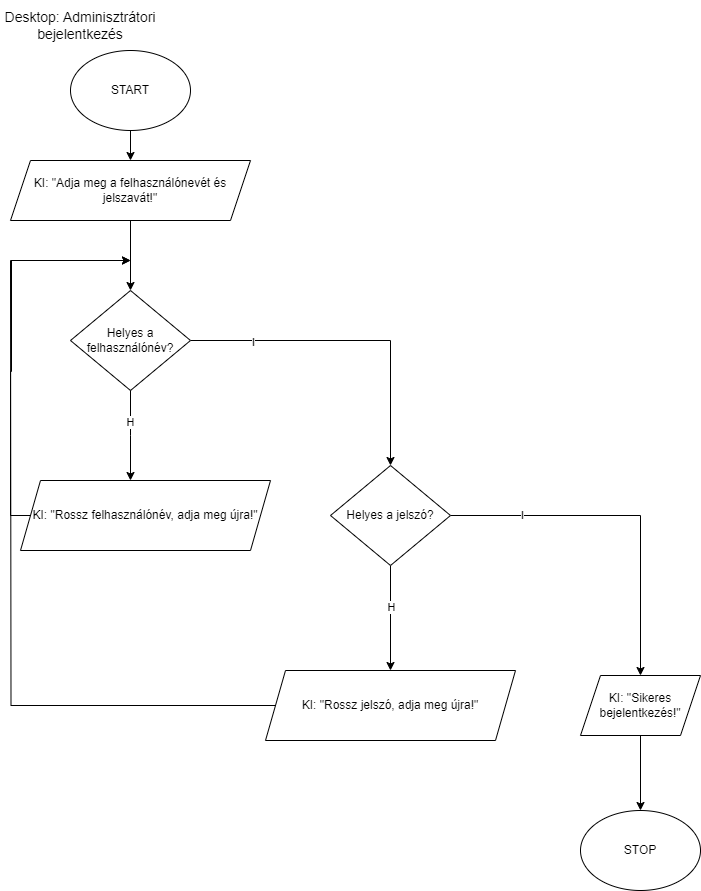
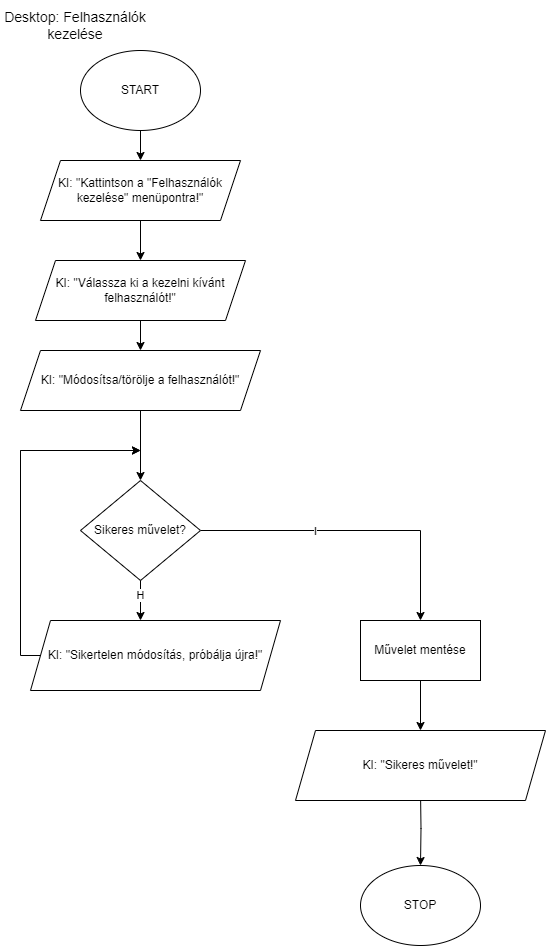
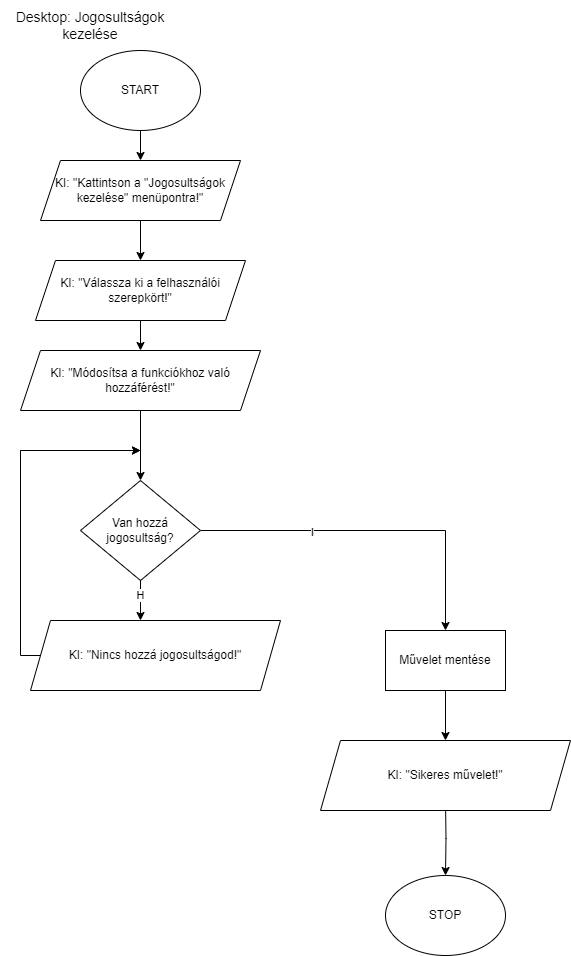
A rendszer RESTful API-t használ a különböző komponensek közötti kommunikációhoz. Az API végpontok a következők:

* Autentikáció és felhasználókezelés:
  + /api/auth/register - Új felhasználó regisztrálása
  + /api/auth/login - Bejelentkezés
  + /api/users/me - Saját felhasználói profil lekérése
  + /api/users/profile - Profil frissítése
  + /api/users/password - Jelszó módosítása
  + /api/users/search - Felhasználók keresése
* Bevásárlólisták kezelése:
  + /api/lists - Listák listázása és új lista létrehozása
  + /api/lists/{listaId} - Egy lista részleteinek lekérése, frissítése és törlése
  + /api/lists/{listaId}/share - Lista megosztása
  + /api/lists/{listaId}/unshare - Megosztás visszavonása
  + /api/lists/{listaId}/products - Termékek kezelése a listában
* Termékkatalógus és kategóriák:
  + /api/productCatalogs - Katalóguselemek kezelése
  + /api/categories - Kategóriák kezelése
  + /api/products - Termékek kezelése
* Adminisztrációs funkciók:
  + /api/admin/users - Felhasználók adminisztrálása
  + /api/admin/promote - Felhasználó előléptetése adminná
  + /api/admin/demote - Admin jogosultság visszavonása
* Statisztikák:
  + /api/statistics - Rendszer szintű statisztikák
  + /api/statistics/personal - Felhasználói szintű statisztikák
  + /api/statistics/users - Felhasználói növekedés statisztikák
  + /api/statistics/lists - Lista aktivitás statisztikák
  + /api/statistics/products - Termék statisztikák

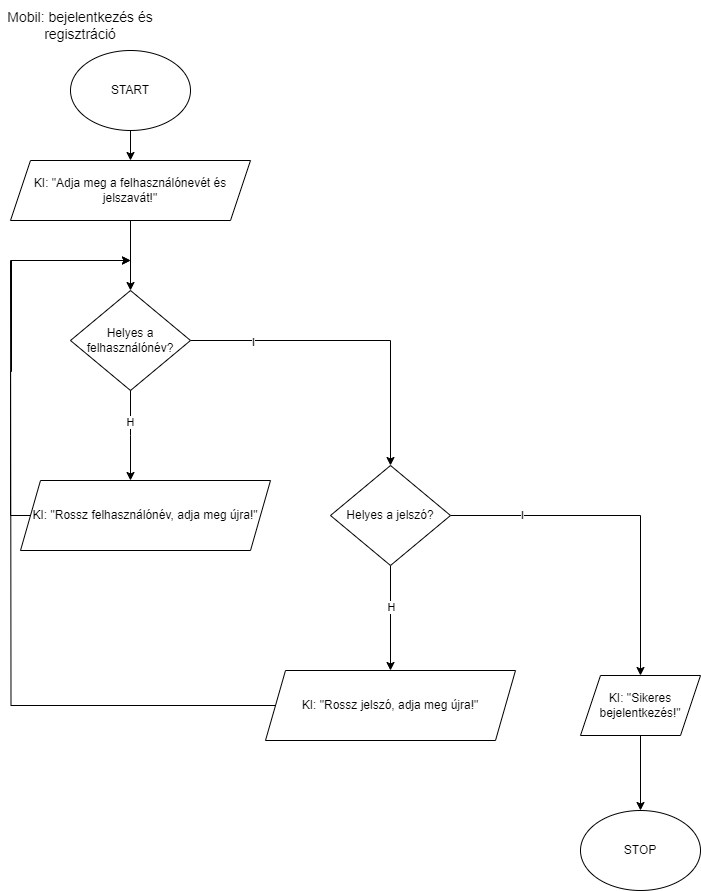
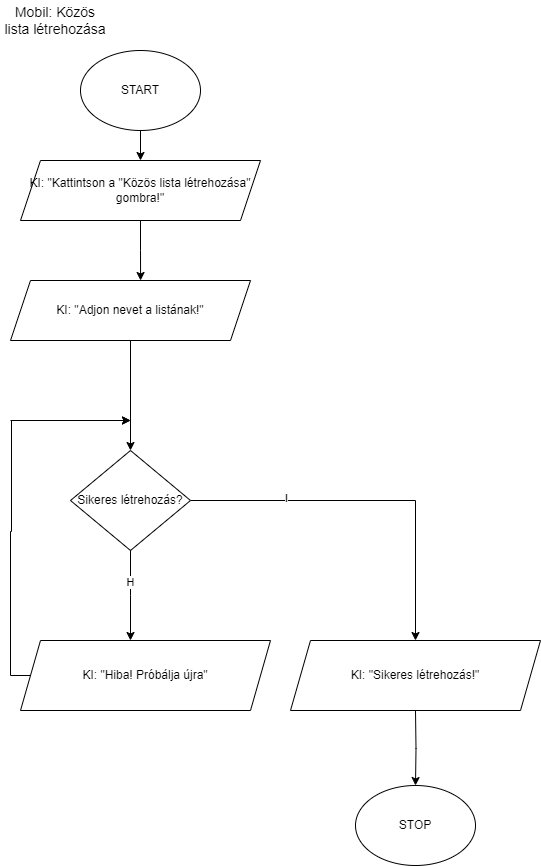
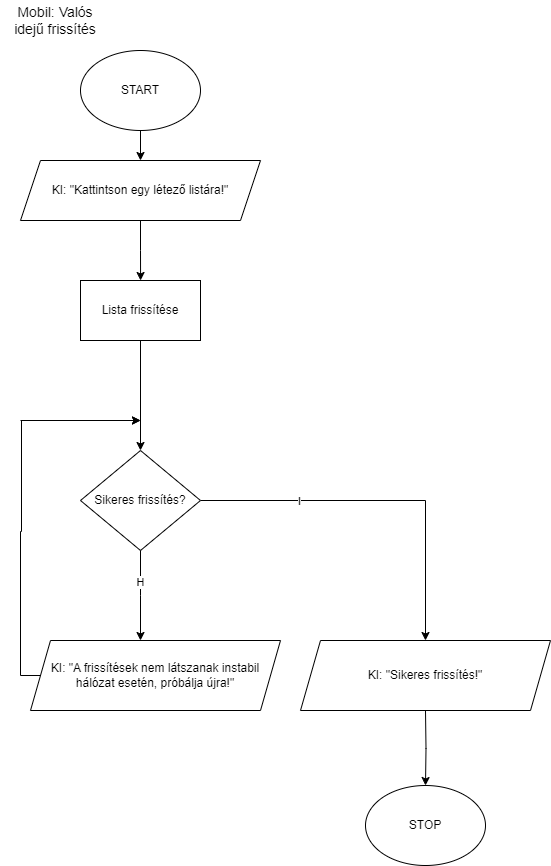
### ****4.3. Használati forgatókönyvek és folyamatábrák (UML)****

A Shoply alkalmazás működésének pontosabb megértéséhez a leggyakoribb felhasználói forgatókönyvekhez kapcsolódóan tevékenységi és szekvenciadiagramokat készítettünk. Ezek az ábrák bemutatják, hogyan zajlik egy-egy folyamat a különböző platformokon (web, asztali, mobil), és hogyan történik az adatok kezelése, a jogosultságok ellenőrzése vagy a valós idejű frissítés.

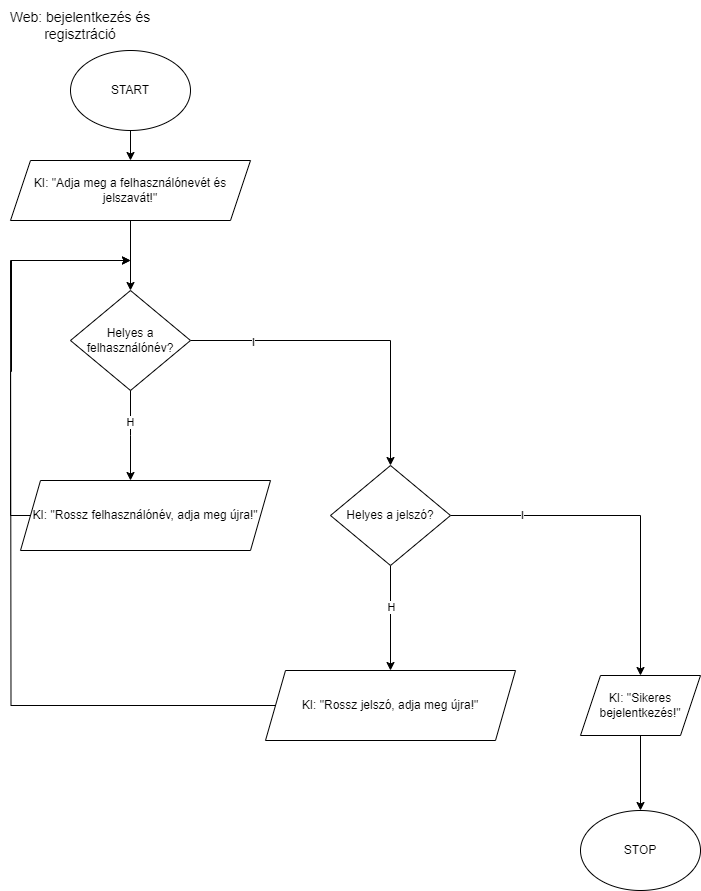
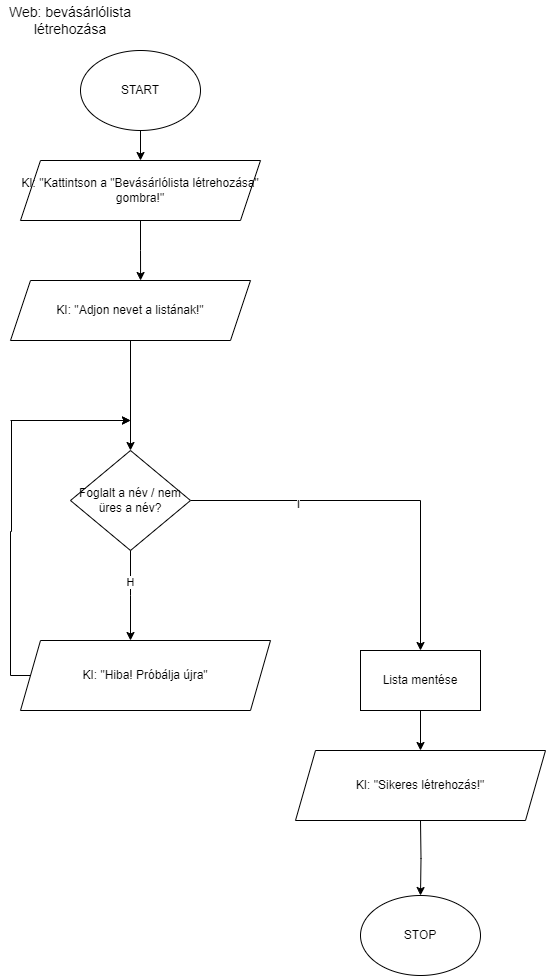
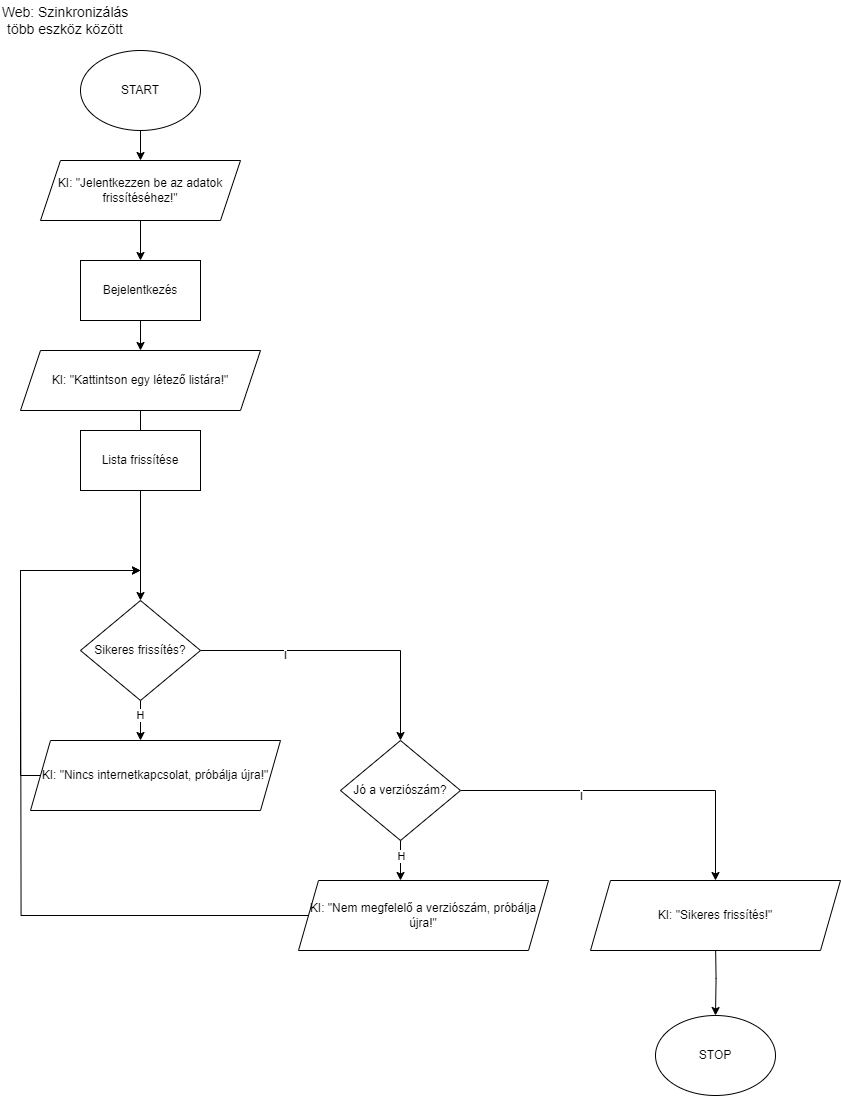
### ****4.3.1. Asztali alkalmazás forgatókönyvek****

* **Papirforgatokonyvek-Desktop\_bejelentkezes.drawio**  
  Ez a diagram szemlélteti az asztali alkalmazásban történő bejelentkezési folyamat lépéseit, beleértve az adatok validálását, szerveroldali hitelesítést, és a sikeres vagy sikertelen visszajelzéseket.
* **Papirforgatokonyvek-Desktop\_felhasznalok\_kezelese.drawio**  
  A diagram bemutatja, hogyan történik a felhasználók kezelése admin jogosultságú felhasználók számára: új felhasználó létrehozása, meglévő adatok módosítása vagy törlés.
* **Papirforgatokonyvek-Desktop\_jogosultsag\_kezelese.drawio**  
  Ez az ábra azt illusztrálja, hogyan történik a különböző jogosultsági szintek kezelése, például egy felhasználó adminná tétele vagy admin jogainak visszavonása.

### ****4.3.2. Mobil alkalmazás forgatókönyvek****

* **Papirforgatokonyvek-Mobil\_bejelentkezes.drawio**  
  A mobil bejelentkezési folyamatot mutatja be – a felhasználói adatbevitel után történő ellenőrzést, az esetleges hibák visszajelzését, valamint a sikeres hitelesítést.
* **Papirforgatokonyvek-Mobil\_kozos\_lista\_letrehozasa.drawio**  
  Ez a diagram részletesen bemutatja, hogyan hozható létre egy közös, másokkal megosztható bevásárlólista mobilalkalmazásban.
* **Papirforgatokonyvek-Mobil\_valos\_ideju\_frissites.drawio**  
  Itt látható, hogyan működik a valós idejű szinkronizáció: egy másik felhasználó által végrehajtott módosítás hogyan jelenik meg automatikusan a mobilalkalmazásban.

### ****4.3.3. Webes alkalmazás forgatókönyvek****

* **Papirforgatokonyvek-Web\_bejelentkezes.drawio**  
  Ez az ábra a webes bejelentkezés folyamatát mutatja: adatbekérés, szerveroldali hitelesítés, visszajelzés a felhasználónak.
* **Papirforgatokonyvek-Web\_bevasarlolista\_letrehozasa.drawio**  
  A vásárlólista webes felületen történő létrehozásának lépéseit tartalmazza, a kategória- és tételválasztástól a mentésig.
* **Papirforgatokonyvek-Web\_szinkronizacio.drawio**  
  Bemutatja, hogyan valósul meg a valós idejű adatfrissítés és szinkronizáció a webes alkalmazásban több eszköz vagy felhasználó között.

## 5. A program működésének leírása

### 5.1. Adatfolyamok

*5.1.1. Valós idejű szinkronizáció*

1. WebSocket kapcsolat:

* Kapcsolat létesítése a szerverrel
* Szoba (room) csatlakozás
* Események figyelése:
  + LIST\_CREATED
  + LIST\_UPDATED
  + LIST\_DELETED
  + ITEM\_ADDED
  + ITEM\_UPDATED
  + ITEM\_DELETED

1. Offline mód:
   * LocalStorage használata
   * IndexedDB adattárolás
   * Változtatások várólistázása
   * Online állapot visszaállásakor szinkronizálás

*5.1.2. Adatbiztonsági mentés*

1. Automatikus mentés:
   * Napi teljes mentés
   * Óránkénti különbségi mentés
   * Felhasználói adatok mentése
   * Listák mentése
2. Visszaállítási folyamat:
   * Mentés kiválasztása
   * Adatok ellenőrzése
   * Visszaállítás végrehajtása
   * Konfliktusok kezelése

### 5.2. Hibakezelés

*5.2.1. Frontend hibakezelés*

1. Válaszidő túllépés:
   * Timeout kezelés
   * Újrapróbálkozás
   * Felhasználói értesítés
2. Hálózati hibák:
   * Offline mód aktiválása
   * Változtatások várólistázása
   * Kapcsolat visszaállításakor szinkronizálás
3. Validációs hibák:
   * Űrlap validáció
   * Kliens oldali ellenőrzés
   * Szerver oldali validáció

*5.2.2. Backend hibakezelés*

1. Adatbázis hibák:
   * Kapcsolat újrapróbálkozás
   * Tranzakció kezelés
   * Konzisztencia ellenőrzés
2. Autentikációs hibák:
   * Token érvényesség
   * Jogosultság ellenőrzés
   * Munkamenet kezelés
3. Rendszerhibák:
   * Naplózás
   * Riasztások
   * Automatikus helyreállítás

## **6. A program tesztelése**

A programok le tesztelése nagyon fontos dolog a szoftver készítésénél. Ez segít abban, hogy a program kódja jó legyen és ne legyenek benne rossz dolgok. A tesztelés nem egyféleképpen történik, hanem több lépésben. Először megnézzük a kicsi részeket külön-külön, ezt hívják unit tesztnek. Aztán megnézzük, hogy az egész program együtt hogy működik, az elejétől a végéig, ez az end to end tesztelés.

Amikor a program kinézetét (frontend) teszteljük, akkor a React dolgokat nézzük meg alaposan, mint a komponenseket, a hook-okat, a Redux store-t és a kis segéd programokat. Ehhez a Jest és a React Testing Library nevű eszközöket használjuk. Ezekkel külön tudjuk nézni a részeket, és azt is meg tudjuk csinálni, mintha a felhasználó nyomkodná a gombokat. A program hátterének (backend) a tesztelésekor a controllereket, a service réteget, a model ellenőrzéseket és a köztes dolgokat nézzük át, itt is a Jest a segítségünk.

Az integrációs tesztek azt nézik, hogy a program különböző darabjai hogy dolgoznak együtt. Ez azt jelenti, hogy megnézzük az API végpontokat, a WebSocket kapcsolatokat, az adatbázis dolgait és hogy tudunk-e más külső dolgokhoz kapcsolódni. Az különösen fontos, hogy a program eleje és háttere hogy beszélget egymással, mert itt szoktak a legtöbb hiba és lassúság előjönni.

Az end to end tesztek azt nézik meg, hogy a felhasználó hogy tudja használni az egész programot. Például, hogy tud-e regisztrálni és bejelentkezni, listát csinálni és átírni, megosztani dolgokat és hogy működik a program akkor is, ha nincs internet. Ehhez a Cypress nevű eszközt használjuk, ami olyan, mintha igazi böngészőben néznénk a programot.

A tesztelés egy automatikus folyamat része (CI/CD). Amikor valaki megváltoztat egy kis kódot, a rendszer magától lefuttatja az unit teszteket, és ha azok jók voltak, akkor elindítja az integrációs teszteket. Egy külön helyen (staging) még alaposabban megnézik a programot, például azt, hogy milyen gyorsan működik, és emberek is átnézik kézzel.

A tesztelés közben nézünk néhány számot. Azt akarjuk, hogy a program elejének a kódjából legalább 80%-ot lefedjenek a tesztek, a hátterénél meg legalább 85%-ot. A nagyon fontos részeknél meg szeretnénk, ha 100%-os lenne a lefedettség. Azt is nézzük, hogy egy oldal mennyi idő alatt tölt be (ne legyen több mint 2 másodperc), az API válasza mennyi idő (kevesebb mint 200 ezredmásodperc), a WebSocket mennyit késik (kevesebb mint 100 ezredmásodperc) és az adatbázis lekérdezések mennyi időt vesznek igénybe (kevesebb mint 50 ezredmásodperc).

A tesztelés eredményeit mindig leírjuk és figyeljük. Minden egyes tesztről részletes leírást készítünk, hogy mi volt a jó, mi volt a rossz, hol vannak a határok és mi történik, ha valami nem sikerül. A teszt jelentések maguktól készülnek el, és a hibákat aszerint rangsoroljuk, hogy mennyire sürgős a javításuk. A regressziós tesztek pedig azt nézik meg, hogy ha valami újat fejlesztünk, akkor a régi dolgok továbbra is jól működjenek.

A tesztelés folyamatát mindig próbáljuk jobbá tenni, és a tapasztalatok alapján finomítjuk. A cél az, hogy mindig jó minőségű szoftvert adjunk ki, ami megbízhatóan működik és a felhasználók is szeretik.

## 7. A program forráskódjának dokumentálása

A forráskód dokumentálása a szoftverfejlesztés egyik alapvető eleme, amely segíti a kód megértését és karbantarthatóságát. A projektben többféle dokumentációs módszert alkalmazunk.

### 7.1. Kommentek használata

A kódban kétféle kommentet használunk:

1. Sor kommentek: Egy soros magyarázatok, amelyek a kód azonnali környezetéhez kapcsolódnak
2. Blokk kommentek: Többsoros leírások, amelyek a funkciók vagy osztályok működését magyarázzák

Példa a projektből (list.service.test.js):

### 7.2. JSDoc dokumentáció

A JavaScript/TypeScript kódban a JSDoc szabványt követjük, amely lehetővé teszi a kód automatikus dokumentációjának generálását. A JSDoc kommentek tartalmazzák:

* A függvény vagy osztály leírását
* A paraméterek típusát és leírását
* A visszatérési érték típusát és leírását
* Kivételek dokumentációját

Példa a projektből (auth.service.test.js):

### 7.3. API dokumentáció

A REST API végpontok dokumentációját a Swagger/OpenAPI specifikációval készítjük. Ez lehetővé teszi:

* Az API végpontok automatikus dokumentációját
* A kérés-válasz formátumok leírását
* A hitelesítési módszerek dokumentációját
* Példa kérések és válaszok megjelenítését

Példa a projektből (statisticsAPI.http):

### 7.4. Teszt dokumentáció

A projektben részletes teszt dokumentációt készítünk, amely leírja:

* A tesztelési stratégiát
* A teszteseteket
* A tesztelési környezetet
* A tesztelési eredményeke

Példa a projektből (frontend-tesztesetek.md):

### 7.5. Kód dokumentációs eszközök

A projektben a következő dokumentációs eszközöket használjuk:

* TypeDoc: TypeScript kód dokumentáció generálásához
* Swagger UI: API dokumentáció megjelenítéséhez
* ESLint: Kód stílus és dokumentációs szabályok ellenőrzéséhez
* Prettier: Kód formázás egységesítéséhez

## 8. Összefoglalás és jövőbeli fejlesztési lehetőségek

### 8.1. A projekt jelenlegi állapota

A Shoply alkalmazás egy komplex, többplatformos rendszer, amely sikeresen implementálja a következő fő funkcionalitásokat:

* Felhasználói regisztráció és hitelesítés
* Vásárlólisták kezelése
* Valós idejű szinkronizáció
* Megosztási lehetőségek
* Kategória kezelés
* Statisztikák és jelentések

A rendszer architektúrája modern technológiákat használ, és a kód megfelelően dokumentált, tesztelt, valamint karbantartható formában került kialakításra.

### 8.2. Jövőbeli fejlesztési lehetőségek

*8.2.1. Funkcionalitás bővítése*

* Intelligens ajánlórendszer: Gépi tanulás alapú termékajánlások implementálása
* Árfigyelő rendszer: Különböző üzletek árainak összehasonlítása és követése
* Receptkezelés: Receptek hozzáadása és automatikus bevásárlólista generálás
* Költségvetés tervezés: Személyes költségvetés követése és elemzése
* QR kód integráció: Termékek gyors hozzáadása QR kód beolvasással

*8.2.2. Technológiai fejlesztések*

* Offline működés: Teljes funkcionalitás offline módban
* PWA fejlesztés: Progressive Web App funkcionalitás kiterjesztése
* Teljesítmény optimalizálás: Gyorsítótárazás és betöltési idők javítása
* Biztonsági fejlesztések: Kétfaktoros hitelesítés bevezetése
* API verziókezelés: API verziókezelési rendszer implementálása

*8.2.3. Platformok kiterjesztése*

* iOS és Android alkalmazások: Natív mobil alkalmazások fejlesztése
* Widget támogatás: Operációs rendszer szintű widgetek implementálása
* Okoseszköz integráció: Intelligens otthoni eszközökkel való integráció
* Voice assistant: Hangalapú vezérlés implementálása

### 8.3. Záró gondolatok

A Shoply alkalmazás jelenlegi verziója szilárd alapot biztosít a további fejlesztésekhez. A modern technológiai stack, a jól strukturált kódbázis és a részletes dokumentáció lehetővé teszi a rendszer hatékony továbbfejlesztését és karbantartását.

A projekt során szerzett tapasztalatok és a felhasználói visszajelzések alapján a rendszer jól szolgálja eredeti célját, miközben számos lehetőséget kínál a további bővítésre és fejlesztésre. A jövőbeli fejlesztések során is kiemelt figyelmet kell fordítani a kód minőségére, a felhasználói élményre és a rendszer skálázhatóságára.