# Tolna Vármegyei SZC Apáczai Csere János Technikum és Kollégium

## Vizsgaremek

**Készítették:**

**Kenéz Bence**

**Kreszl Tamás**

**Tóth Krisztián**

**Dombóvár**

**2025**

# Tolna Vármegyei SZC Apáczai Csere János Technikum és Kollégium

Szakma megnevezése: Szoftverfejlesztő és tesztelő

A szakma azonosító száma: 506131203

## Vizsgaremek

SneakR

**Készítették:**

**Kenéz Bence**

**Kreszl Tamás**

**Tóth Krisztián**

**Dombóvár**

**2025**

# Tartalomjegyzék

**Feladat rövid ismertetése** ..... 4
**Bevezetés** ..... 4
**Cél** ..... 4
**Jövőbeli tervek** ..... 4
**Használt technológiák** ..... 5
**Programozási Nyelvek** ..... 5
**Adatbázis típusa** ..... 5
**Egyéb technológiák** ..... 5
**Adatbázis** ..... 7
**Adatbázis rövid magyarázata** ..... 7
**Fejlesztés módszertan** ..... 9
**Scrum** ..... 9
**Elvégzett teszt** ..... 10
**Manuális tesztelés:** ..... 10
**Program működésének részletes leírása** ..... 11
**Fejlesztői dokumentáció** ..... 12
**Fejlesztési technológiák** ..... 12
**HTTP protokollok** ..... 13
**A rétegek szerepköre a projektben** ..... 13
**A projektben való rétegek felépítése** ..... 15
**Frontend részletes leírása** ..... 16
**Lokális szerver futtatása** ..... 17
**Csapatmunka** ..... 18
**Szerepek:** ..... 18
**Jövőbeli tervek:** ..... 20

# Feladat rövid ismertetése

## Bevezetés

A projekt fő szempontja, hogy egy modern, digitális megoldást kínáljon a cipővásárlás és -értékesítés világában, amely egyszerre szolgálja ki a hagyományos webshop és a közösségi piactér igényeit. A tervezett webes alkalmazás nemcsak a saját termékek hatékony online értékesítését teszi lehetővé, hanem egy resell szekcióval is kiegészül, ahol felhasználók hirdethetik meg saját cipőiket más érdeklődők számára. A vásárlók és eladók közötti átlátható, biztonságos és könnyen kezelhető platform révén a projekt célja, hogy új szintre emelje az online cipőkereskedelem élményét. Az alkalmazás olyan funkciókat biztosít, mint a termékfeltöltés, keresés, rendeléskezelés, amelyek egyaránt támogatják az egyszerű vásárlást és a hatékony eladást.

GitHub Repository Link

## Cél

A SneakR egy olyan közösség, amely a divat, a fenntarthatóság és a technológia találkozását támogatja. Célunk, hogy minden felhasználó számára optimális élményt nyújtsunk – legyen szó ritka limited edition cipők kereséséről vagy a saját tároló kiürítéséről.

## Jövőbeli tervek

A projekt jövőbeli fejlesztési irányai között szerepel a platform funkcióinak bővítése és továbbfejlesztése annak érdekében, hogy még teljesebb vásárlói és eladói élményt nyújtson. A jelenlegi webshop- és resell funkciók alapot adnak egy olyan rendszer számára, amely a jövőben számos új lehetőséggel bővülhet. Terveink között szerepel például értékelési és visszajelzési rendszer bevezetése, amely segíti a bizalomépítést a felhasználók között, illetve különböző szűrési és ajánlási funkciók fejlesztése, amelyek megkönnyítik a vásárlók számára a termékek közötti eligazodást.

# Használt technológiák

## Programozási Nyelvek

*A projectünk során a következő programozási nyelveket használtuk:*

1. **Frontend**
    - Angular(webalkalmazás-keretrendszer)
    - HTML
    - TypeScript
    - CSS
2. **Backend**
    - Java

## Adatbázis típusa

Adatbázisunk egy MySQL-re épülő, nagy teljesítményű relációs adatbázis-rendszer. A relációs adatbázisok előnye, hogy strukturáltan tárolják az adatokat, miközben könnyen kezelhetők és lekérdezhetők. Az MySQL, mint vezető nyílt forráskódú platform, robust keretrendszert biztosít az adatok biztonságos tárolásához, gyors kezeléséhez és skálázható működtetéséhez, így kiválóan támogatja üzleti folyamataitokat.

## Egyéb technológiák

1. **Jira**
    - A Jira egy komplex projektmenedzsment platform, amely lehetővé teszi csapatok számára a projektfeladatok dinamikus nyomon követését, a feladatok reszponzív kezelését, valamint a fejlesztési folyamatok strukturált szervezését. Az eszköz kiemelkedik a feladatok prioritás-alapú rendszerezésében, az Agile módszertanok (pl. Scrum) támogatásában, valamint a csapattagok közötti valós idejű kommunikáció és együttműködés elősegítésében. Rendszeresen generál áttekintő jelentéseket és metrikákat, így átláthatóbbá teszi a haladást, és segít a hatékony döntéshozatalban.
2. **Git**
    - A Git egy elosztott verziókezelő rendszer, amely kifejezetten a fejlesztők számára kínál hatékony megoldásokat a kódbázis változtatásainak nyomon követésére, a módosítások strukturált kezelésére és a csapaton belüli zökkenőmentes együttműködésre.

Az elosztott architektúrájának köszönhetően lehetővé válik a párhuzamos fejlesztés, a részletes változáselőlmények (commitok) automatikus rögzítése, valamint a korábbi állapotok gyors és biztonságos visszaállítása. Így akár offline munkavégzés mellett is garantálja az adatok integritását és a folyamatok átláthatóságát.

3. **Figma**
    - A Figma egy felhőalapú, többplatformos design- és prototípuskészítő platform, amely lehetővé teszi tervezők és fejlesztők számára, hogy zökkenőmentesen együttműködjenek a felhasználói felületek (UI/UX) tervezése, prototípusok fejlesztése és a designrendszerek optimalizálása során. A valós idejű szerkesztés és megosztás funkcióinak köszönhetően a csapatok azonnal reagálhatnak változtatásokra, akár különböző helyszíneken vagy időzónákban dolgozva. A Figma nemcsak a designfolyamatok automatizálását segíti elő (pl. komponenskönyvtárak, interaktív prototípusok), hanem integrálódik más fejlesztői eszközökkel is, így garantálva a tervezés és implementáció közötti folytonosságot. Emellett a verziókövetés és a kommentelési lehetőségek átláthatóbbá teszik a visszajelzési ciklusokat, és maximalizálják a kreatív együttműködés hatékonyságát.

# Adatbázis

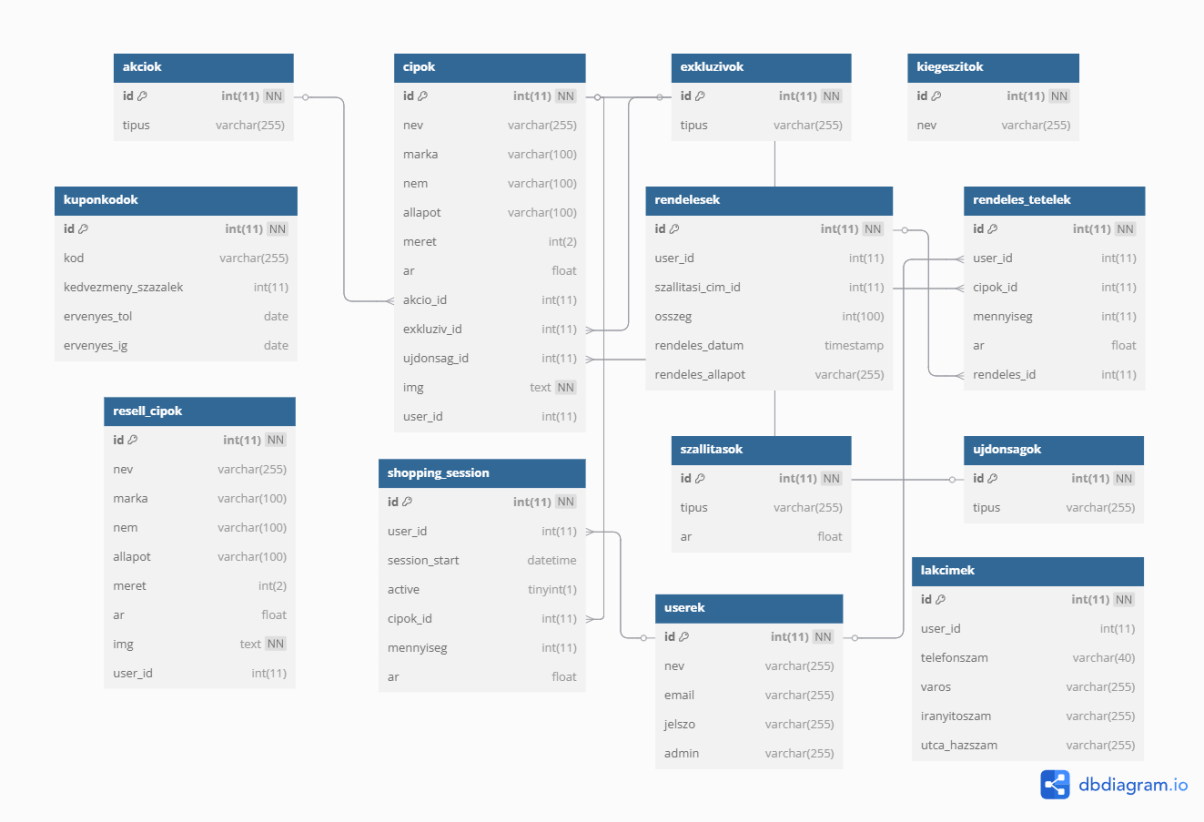
## Adatbázis rövid magyarázata

Az alkalmazás optimális működéséhez kulcsfontosságú, hogy az összes releváns adatot strukturáltan, hatékonyan és biztonságosan tároljuk. Ennek alapvető feltétele egy jól megtervezett adatbázis, amely nemcsak a webalkalmazás stabilitását és teljesítményét biztosítja, hanem alapvető építőeleme a felhasználói élmény javításának és az adatok gyors, pontos kezelésének. A rendezett adattárolás lehetővé teszi, hogy az alkalmazás zökkenőmentesen hozzáférjen, feldolgozzon és frissítsen információkat, ezzel is növelve a rendszer hatékonyságát.

Az objektumorientált programozás (OOP) integrációja a backend rendszerbe döntő szerepet játszik ezen célok elérésében. Az OOP lehetővé teszi, hogy minden entitást (pl. felhasználók, termékek, tranzakciók) önálló osztályokkal modellezzünk, amelyek egyértelműen definiálják a tulajdonságokat, viselkedéseket és kapcsolatokat más entitásokkal. Ez az elv:

- Egyszerűsíti a karbantartást (módosítások lokálisak maradnak),
- Megkönnyíti a bővítést (új funkciók modulárisan építhetők be),
- Támogatja a skálázhatóságot (az osztályok függetlenül működnek és terheléselosztásra alkalmasak).

Ezen túlmenően, az OOP-alapú struktúra átláthatóbbá teszi a kódot, elősegíti a csapaton belüli együttműködést, és csökkenti a hibák kockázatát azáltal, hogy az adatok és metódusok logikailag elkülönülnek. Így a rendszer nemcsak hatékonyabb, de jövőállóbbá válik a változó igényekhez való adaptálás terén.




A képen a rendszerünk adatbázis sémájának ER diagramja (Entity-Relationship diagram) látható, ami az adatbázisban tárolt objektumok (entitások) és azok kapcsolatainak vizuális megjelenítése. A diagram elemei között táblák (entitások), attribútumok és kapcésolatok (relációk) találhatók. Az egyes táblák elsődleges kulcsait (PK, Primary Key) és idegen kulcsait (FK, Foreign Key) is jelölik.

A diagram több fő részre osztható, mindegyik különböző funkcionális területeket képvisel:

1. **Felhasználói adatok:** A userek tábla tárolja a felhasználók alapadatait, mint a Id, Név, Email, Jelszo, Admin. Kapcsolódik a rendeles_tetelek, shopping_session táblához.
2. **Termék információk:** A cipok, resell_cipok kezelik általunk feltöltött cipőket, és a felhasználók által feltöltött cipők.
3. **Rendelések:** A rendelesek tábla tárolja a leadott rendelések adatait. Mint a rendelés ID-ját, User-ID, szállitasi_cim_id, osszeg, rendeles_datum, rendeles_allapot-ot.

Az idegen kulcsok felelősek a diagram bonyolult kapcsolati rendszerének fenntartásáért és az adatok sértetlenségéért (integritásáért). Ezek biztosítják, hogy az összekapcsolt adatok mindig következetesek és aktuálisak maradjanak. Az ER diagram pedig teljes képet ad arról, hogyan kapcsolódnak egymáshoz az adatok, és miként lehet őket hatékonyan kezelni az iskolai rendszeren belül.

# Fejlesztés módszertan

## Scrum

1. **Rendszeres fejlesztési ciklusok**
    - A Scrum keretrendszer alkalmazásával a fejlesztést rendszeres, általában három- vagy négy hetes ciklusokra (sprintekre) bontottuk. Ennek köszönhetően a kifejlesztett funkciókhoz gyorsan és ütemezetten hozzáférhettünk, ami azonnali visszajelzést tett lehetővé mind a csapat, mind az érintettek számára.
2. **Csapatmunka és átláthatóság**
    - A Scrum bevezetése javította a projekt átláthatóságát és a csapaton belüli együttműködést. Mivel a Scrum értekezletek rendszeresek voltak, és a projekt backlogot folyamatosan frissítettük, mindenki naprakész volt a projekt állásával és a prioritásokkal kapcsolatban.
3. **Rugalmasság a változásokra**
    - Az iteratív fejlesztés és a rendszeres retrospektív értekezletek révén a Scrum keretrendszer képessé tett minket a visszajelzések könnyű integrálására és a változásokra való gyors reagálásra. Ez a rugalmasság segített optimalizálni a projekt céljait és a fejlesztett funkciókat a munka során.

# Elvégzett teszt

## Manuális tesztelés:

- A manuális tesztelés során a tesztelő emberi közreműködéssel, kézzel végzi a vizsgálatot. Ez a megközelítés nélkülözhetetlen annak felmérésére, hogy milyen a felhasználói élmény, és mennyire magától értetődő az alkalmazás funkcióinak használata. Ennek érdekében a tesztelő közvetlenül használja az alkalmazást, ellenőrzi a kezelőfelületet, és valós felhasználói helyzeteket szimulál.
- A tesztelés során két kulcsfontosságú területre összpontosítunk. Az egyik a felhasználói élmény felmérése, amely során azt vizsgáljuk, mennyire könnyen kezelhető és érthető az alkalmazás a felhasználók számára, és hogyan viselkedik általános használat mellett.
- A másik lényeges terület a funkcionális validáció: itt arról győződünk meg, hogy az alkalmazás minden funkciója pontosan a tervek és specifikációk szerint működik-e, amit különböző funkciók, forgatókönyvek és bemenetek tesztelésével ellenőrzünk.

## Automatizált (Unit) tesztelés:

- A unit tesztek kulcsfontosságúak a kód stabilitásának és megbízhatóságának biztosításában. Segítségükkel ellenőrizhetjük, hogy az egyes modulok a tervezett módon működnek, és a változtatások nem okoznak hibákat a meglévő funkciókban.
- A unit tesztelés célja a korai hibafeltárás; automatizáltságának és ismételhetőségének köszönhetően a fejlesztők gyorsabban javíthatják a talált hibákat.

## Használt technológiák

- Manuális Tesztelés: Chrome
- NetBeans: Unit

# Program működésének részletes leírása

Az oldal két fő felhasználói profilra fókuszál:

- Felhasználó
- Adminisztrátor

Az alkalmazás a bejelentkezés során ellenőrzi hogy a bejelentkező felhasználó rendelkezik-e admin jogosultságokkal, vagy sem. Ezt figyelembe véve irányítja át a felhasználót az alkalmazás különböző részeire.

## *Felhasználóként bejelentkezve*

Főbb funkciói:

1. **Vásárlás a Webshopból:**
    - **Kosárba helyezés:** A webshop kínálatából cipőt a kosarába tenni.
    - **Kosár kezelése:** Megtekinteni, módosítani a kosár tartalmát.
    - **Megrendelés:** A kosárban lévő termékeket megrendelni a fizetési és szállítási adatok megadásával.
    - **Rendelési előzmények megtekintése:** Korábbi vásárlásait nyomon követni.
2. **Tevékenységek a Resell Oldalon:**
    - **Cipő feltöltése eladásra:**
    - Megadni a cipő adatait (márka, modell, méret, állapot, leírás).
    - Képeket feltölteni a cipőről.
    - Árat meghatározni.
    - **Saját hirdetések kezelése:** Megtekinteni, szerkeszteni vagy törölni a feltöltött cipőhirdetéseit.
    - **Vásárlás a resell oldalról:** Más felhasználók által feltöltött cipőket megvásárolni (hasonlóan a webshopos vásárláshoz: kosárba tétel, fizetés).
    - **Rendelési előzmények megtekintése:** Korábbi vásárlásait nyomon követni.

### Böngészés és Keresés:

- **Termékek megtekintése:** Böngészni a cipőket mind a webshop fő oldalán, mind a resell szekcióban.
- **Keresés és Szűrés:** Cipőkre keresni név, márka, méret, ár, állapot (resellnél) és egyéb szempontok alapján.
- **Termékadatlapok megtekintése:** Részletes információkat és képeket nézni a cipőkről.

### *Adminként bejelentkezve*

Főbb funkciói:

1. **Termékek feltöltése a webshopra**
2. **Felhasználók kezelése**
   - Az adminisztrátor törölheti és szerkesztheti a felhasználói profilt
3. **Hirdetések kezelése**
   - Az adminisztrátor törölheti és szerkesztheti a felhasználók vagy a saját termékei hirdetését.

# Fejlesztői dokumentáció

## Fejlesztési technológiák

A fejlesztés során frontend oldalon az Angular keretrendszert alkalmaztuk, amely modern, dinamikus és felhasználóbarát felhasználói felületet biztosít. A backend implementációhoz a JAX-RS REST API-t választottuk, amely a Java EE szabványos specifikációjára épül. Előnyei közé tartozik az egyszerű annotáció-alapú konfiguráció, a keretrendszerfüggetlenség, valamint a könnyű integrálhatóság más Java EE technológiákkal.

Az alkalmazás adatkezelési alapját egy relációs adatbázis, a MySQL képezi. Ez a megoldás lehetővé teszi az adatok hatékony tárolását és kezelését, ezáltal biztosítva az alkalmazás megbízhatóságát, stabil működését és adatbiztonságát.

## HTTP protokollok

*A szerver és a kliens közötti kommunikáció során két alapvető HTTP metódust használtunk a kérések kezelésére.*

1. GET
   - Olvasási műveletekhez használt, egyszerűen tesztelhető, és a paraméterek könnyedén megoszthatók az URL-ben.
   - Elsősorban adatok lekérésére alkalmazzák, a válaszok pedig gyorsítótárazással hatékonyabbá tehetők.
2. POST
   - Az írási műveletekhez ideális HTTP metódus, amely biztonságos módon a kérés törzsében továbbítja az adatokat, így jól alkalmazható állapotváltozások kezelésére.
   - Mivel az adatok nem az URL-ben, hanem a kérés testében szerepelnek, a POST metódus különösen alkalmas érzékeny információk továbbítására.
   - Gyakran használják adatküldésre a szerver felé, például űrlapok beküldésekor vagy fájlok feltöltése során.

# A rétegek szerepköre a projektben

## Config Package

**Szerep:** A Config package felelős az alkalmazás konfigurációs beállításainak kezeléséért.

**Funkciók:**

- **Konfigurációkezelés**: A Config csomag olyan osztályokat tartalmaz, amelyek az alkalmazás konfigurációs beállításainak kezeléséért felelnek, mint például a CORS szabályok, adatbázis-kapcsolati adatok, vagy a tokenek beállításai.
- **Beállítások betöltése és inicializálása**: Ez a réteg felelős az alkalmazás működéséhez szükséges konfigurációs adatok betöltéséért és az ezekhez kapcsolódó osztályok inicializálásáért.

## Controller Package

**Szerep:** A Controller package felelős a bejövő HTTP kérések kezeléséért és azok feldolgozásáért.

**Funkciók:**

- **HTTP kérések kezelése:** A Controller réteg feladata a kliensektől érkező HTTP kérések fogadása.
- **Kérések feldolgozása:** A Controller osztályok képesek a beérkező kérésekből adatokat kinyerni, majd ezek alapján meghívják a megfelelő *Service* rétegeket az üzleti logika végrehajtására.
- **Válasz generálása**: A Controller osztályok által visszaadott értékek jellemzően HTTP válaszok, amelyeket a rendszer a kliens felé továbbít.

## Modell Package

**Szerep:** A Modell package a háttérbeli adatok reprezentálásáért felelős

**Funkciók:**

- **Adatmodellek létrehozása:** A Model réteg olyan osztályokat tartalmaz, amelyek az alkalmazás különböző adategységeit reprezentálják, például a felhasználókat, osztályokat, jegyeket és egyéb entitásokat.
- **Adatok validálása:** Ez a réteg gyakran magában foglalja az adatellenőrzési logikát is, amely biztosítja, hogy a modellekhez tartozó adatok megfeleljenek az elvárt formátumnak és szabályoknak.

## Service Package

**Szerep:** A Service package felelős a logika kezelésért és végrehajtásáért

**Funkciók:**

- **Üzleti logika megvalósítása:** A Service réteg tartalmazza azokat az osztályokat, amelyek az alkalmazás üzleti logikáját valósítják meg, például adatlekérdezést, adatmanipulációt, illetve üzleti szabályok alkalmazását.
- **Adatkezelés és műveletvégzés:** A Service osztályok gyakran kapcsolatban állnak az adatbázissal vagy más külső adatforrásokkal annak érdekében, hogy adatokat lekérdezzenek, módosítsanak vagy töröljenek.

# A projektben való rétegek felépítése

## Config Package

**CorsFilter osztály:** Ez az osztály felelős a Cross-Origin Resource Sharing (CORS) beállítások kezeléséért a backend-en. Segít megakadályozni vagy engedélyezni a kereséseket különböző eredetű forrásokból.

**Token osztály:** Ez az osztály lehetőséget biztosít a tokenek kezelésére, például a JWT (JSON Web Token) generálására és ellenőrzésére.

## Controller Package

- **ApplicationConfig:** A globális alkalmazásbeállításokat kezeli
- **LakcimController:** A lakcímekkel kapcsolatos kéréseket kezeli.
- **RendelesController:** A rendelések lekérését létrehozását kezeli.
- **ResellShoeController:** A Resell cipők hozzáadása, módosítása és lekérdezésével foglalkozik.
- **ShoeController:** A cipők hozzáadása, módosítása és lekérdezésével foglalkozik.
- **UserController:** A felhasználókkal kapcsolatos műveletek irányítása.

## Service Package

- **LakcimService:** A lakcímekkel kapcsolatos függvények validációját tartalmazza
- **RendelesService:** Az rendelésekkel kapcsolatos függvények validációját tartalmazza.
- **ResellShoeService:** A Resell cipőkkel kapcsolatos függvények validációját tartalmazza
- **ShoeService:** Az cipőkkel kapcsolatos függvények validációját tartalmazza.
- **UserService:** A felhasználókkal kapcsolatos függvények validációját tartalmazza.

# Frontend részletes leírása

## Könyvtárstruktúra

A projektünk átlátható és jól szervezett könyvtárstruktúrával rendelkezik, amely megkönnyíti a fejlesztést és a navigációt. Az alapvető struktúra a következő fő elemekből áll:

- **src:** Az alkalmazás teljes forráskódját tartalmazó gyökérkönyvtár.
- **app:** Az alkalmazás központi mappája, ahol megtalálhatóak a komponensek, szolgáltatások és modellek.
- **assets:** Statikus erőforrásokat, például képeket és egyéb fájlokat tartalmazó könyvtár.

## Komponensek

Az alkalmazás minden oldalát és funkcióját külön-külön komponensekre bontjuk:

- **about**: Ez a komponens az oldalról szóló információkat jeleníti meg
- **admin**: Az adminisztrációs felületet biztosító komponens, amely lehetővé teszi az adminok számára az alkalmazás adatainak kezelését.
- **contact**: Ez a komponens biztosítja a kapcsolatfelvételt, ahol a felhasználók üzenetet küldhetnek az oldal üzemeltetőinek.
- **gyik**: A GYIK (Gyakran Ismételt Kérdések) oldalért felelős komponens, ahol a leggyakoribb kérdésekre találhatók válaszok.
- **login**: Ez a komponens felelős a felhasználók bejelentkezési felületének megjelenítéséért, ahol megadhatják felhasználónevüket és jelszavukat.
- **password**: Ez a komponens a jelszókezeléssel kapcsolatos funkciókat biztosítja, például jelszó emlékeztetés.
- **products**: A termékek listázásáért és megjelenítéséért felelős komponens, ahol a felhasználók böngészhetnek az elérhető cipők között.
- **register**: A regisztrációs felületet biztosító komponens, ahol az új felhasználók létrehozhatják saját fiókjukat.
- **resell**: A resell funkció központi komponense, amely összefogja a viszonteladással kapcsolatos aloldalakat és funkciókat.
- **resell-cart**: Ez a komponens a viszonteladásra szánt termékek kosarának megjelenítését és kezelését végzi.
- **resell-products**: A felhasználók által viszonteladásra kínált cipőket listázó komponens.
- **resell-sell**: A viszonteladásra feltöltött termékek létrehozásáért és szerkesztéséért felelős komponens.

- **resell-user**: A viszonteladó felhasználók adatait és tevékenységét megjelenítő komponens.
- **select**: Egy kiválasztást segítő komponens, amely lehetővé teszi például termékkategóriák vagy opciók kiválasztását.
- **shoe-controller**: Ez a komponens a cipőkkel kapcsolatos adatok kezeléséért felelős, például: törlés.
- **webshop**: A webshop főoldalát megjelenítő komponens, amely a termékek böngészésére és vásárlására szolgál.
- **webshop-cart**: A vásárlói kosarat megjelenítő komponens, amelyben a felhasználók megtekinthetik és kezelhetik a kiválasztott termékeiket.

## Stíluslapok

Projektünk stílusait kizárólag natív CSS-ben írjuk, preprocesszorok használata nélkül. Minden komponenshez külön stílusfájl tartozik, ami lehetővé teszi a stílusok elszigetelését és egyszerűbb karbantartását. A CSS szabályokat úgy alakítjuk ki, hogy biztosítsák az alkalmazás egységes megjelenését, miközben megőrzik az egyes komponensek egyedi vizuális jellemzőit. Emellett kiemelt figyelmet fordítunk arra, hogy elkerüljük a túlzott vagy felesleges stílusokat, és kizárólag a valóban szükséges szabályokat alkalmazzuk a letisztult és hatékony megjelenés érdekében.

## Responzív dizájn

Az alkalmazásunk kialakítása során kiemelt figyelmet fordítunk a reszponzivitásra, így minden oldal és komponens képes rugalmasan alkalmazkodni a különböző kijelzőméretekhez és eszközökhöz. Ennek köszönhetően a felhasználói élmény minden platformon egységes és felhasználóbarát marad.

## Lokális szerver futtatása

Az alkalmazás működését alapos tesztelésnek vetettük alá egy komplex, több szerverből álló lokális környezetben. A fejlesztés során részletesen ellenőriztük az összes komponens – beleértve a frontend és backend funkciókat, az adatbázis-kezelést, valamint a szerverek közötti kommunikációt – működését és együttműködését. A tesztelési folyamat célja az volt, hogy garantáljuk az alkalmazás egységes, hatékony és összehangolt működését. A folyamatos ellenőrzések révén biztosítottuk, hogy a rendszer minden része megbízhatóan és magas teljesítménnyel működjön a különböző környezetekben is.

# Csapatmunka

## Szerepek:

**A projektben Kenéz Bence a következő szerepeket töltötte be, és a következő területekért felelt:**

1. Project manager (Projektmenedzser)
    - Feladata a projekt egészének irányítása és felügyelete.
    - Koordinálta a csapatmunkát és ütemezte a fejlesztési folyamatokat.
    - Hatékony kommunikáció biztosítása a csapattagok között és a projekt érintettjeivel.
2. Frontend fejlesztő
    - A frontend részleg fejlesztése és kialakítása volt az ő feladata
    - Megvalósította a felhasználói felület tervezését és implementálását.
    - Felelős volt az alkalmazás előtéri funkcionalitásának kialakításáért és megvalósításáért.

**A projektben Kreszl Tamás a következő szerepeket töltötte be, és a következő területekért felelt:**

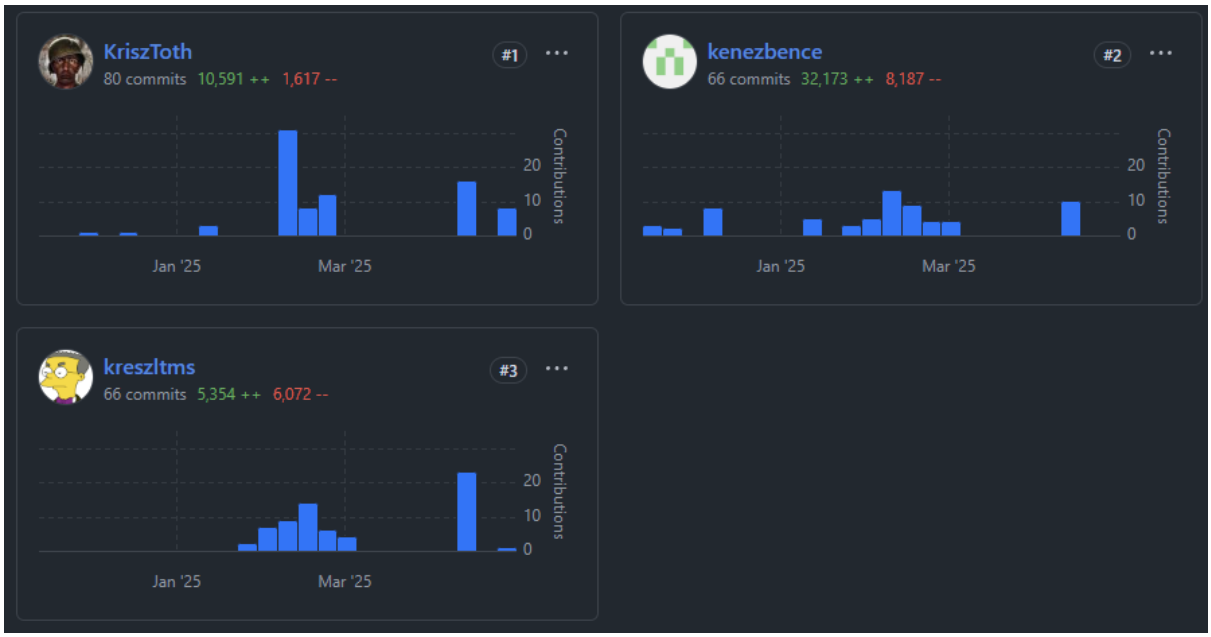
1. Adatbázisfejlesztő
    - Felelős volt az adatbázis kialakításáért és felépítéséért.
    - Megvalósította az adatbázis struktúráját és kapcsolódási logikáját az rendszer számára.
    - Biztosította az adatbázis hatékony működését és az adatok integritását a projekt specifikációinak megfelelően.
2. Manuális tesztelő
    - Visszajelzéseket adott a fejlesztőcsapatnak a teszteredmények alapján.
    - Biztosította, hogy az alkalmazás megfeleljen a minőségi követelményeknek

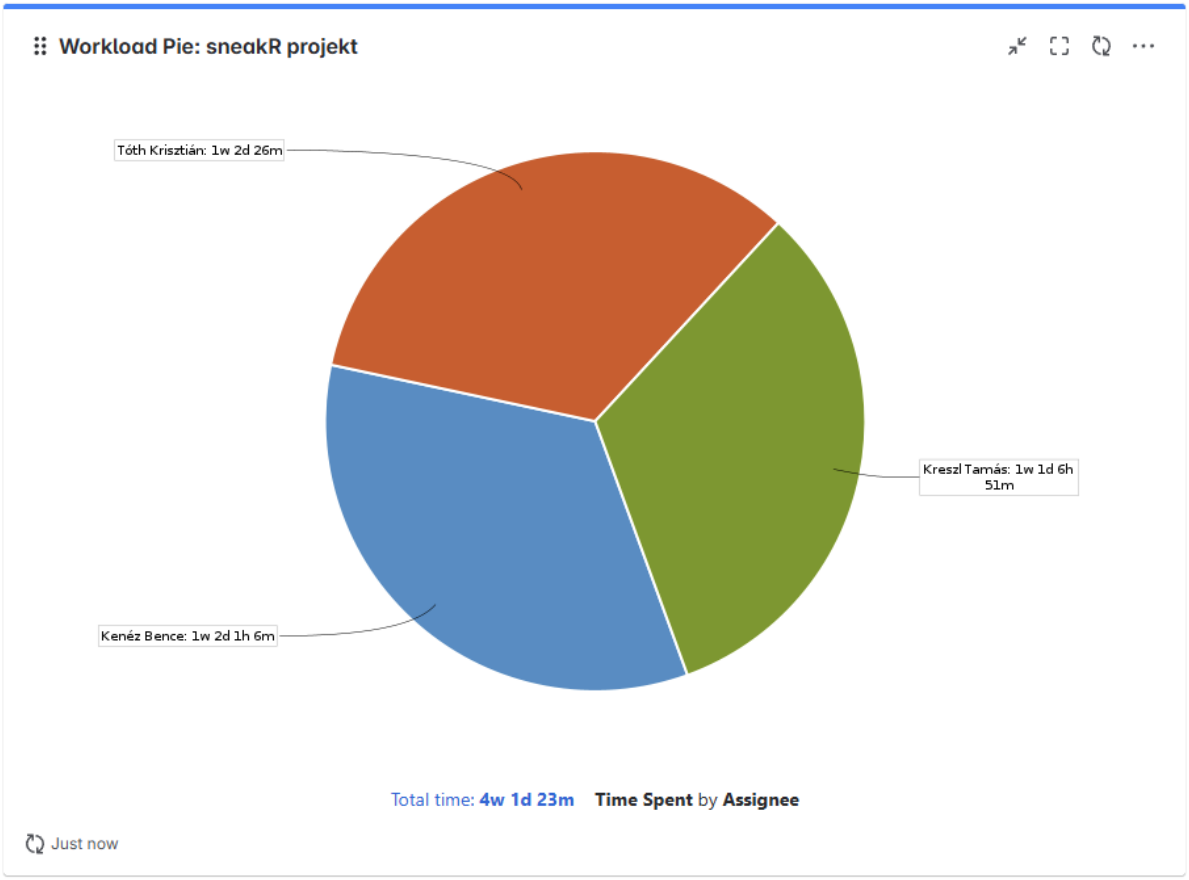
**A projektben Tóth Krisztián a következő szerepeket töltötte be, és a következő területekért felelt:**

1. Backend fejlesztő
    - Felelős volt a háttérrendszer kialakításáért és implementációjáért.
    - Megvalósította a szerveroldali logikát és funkciókat.
    - Biztosította, hogy a rendszer hatékonyan működjön.

2. Unit tesztelő
    - Tesztelte a backend részegységeit.
    - A kód minőségének és funkcionalitásának biztosításáért implementálta és futtatta a unit teszteket.
    - Meggyőződött róla, hogy a backend kódja megfelel a tervezett specifikációknak és szabványoknak.

A csapatmunkánk során szerzett tapasztalatok alapján megállapítottuk, hogy az adaptív projektmenedzsment megközelítés kiemelkedően hatékony volt a rugalmas és hatékony fejlesztési folyamatok kialakításában. Az adaptív módszer különösen hasznosnak bizonyult a változó követelmények közös kezelésében. A csapatmunkánk során szerzett tapasztalatok alapján azonosítottunk olyan területeket, ahol a jövőben tovább fejlődhetünk, különös tekintettel a projektmenedzsment csapaton belüli finomhangolására.






# Jövőbeli tervek:

## SimplePay implementáció:

Az egyik kiemelt jövőbeli tervünk, hogy a weboldalunkra implementáljuk a SimplePay fizetési rendszert. Ezzel a lépéssel szeretnénk biztosítani felhasználóink számára egy biztonságos és kényelmes fizetési lehetőséget. A SimplePay integrálása várhatóan javítani fogja a felhasználói élményt és egyszerűsíti a vásárlási folyamatot. Ez a fejlesztés fontos része a weboldalunk további bővítésének és a szolgáltatásaink színvonalának emelésének.

## Az oldalon lévő cipőkhöz több méret hozzárendelése:

A jövőbeni terv szerint minden cipőmodellhez több méretet lehet majd hozzárendelni, amelyek közül a vásárlók választhatnak. A méretekhez külön készletnyilvántartás kapcsolódik, így valós idejű láthatóságot biztosítva az egyes méretek elérhetőségéről (pl. "elfogyott" jelzés). A népszerű vagy limitált méretek kiemelésével növeljük a vásárlói érdeklődést. Ez a fejlesztés optimalizálja a felhasználói élményt és hatékonyabbá teszi a készletkezelést.