

# Követelményspecifikáció

## Ecommerce Shopping Assistant készítése MI támogatásával

Mesterséges intelligenciát alkalmazó megoldás fejlesztése 4

### Feladatkiírás

Az elkészítendő alkalmazás egy ecommerce shopping assistant, amely segíti a felhasználókat online vásárlási élményük optimalizálásában. Az alkalmazás két fő részből áll: egy Angular alapú webshop frontendből, egy Spring Boot alapú backendből, és egy MySQL adatbázisból. Az MI (mesterséges intelligencia) modul pedig vásárlási szokások alapján személyre szabott termékajánlásokat generál a felhasználók számára.

### A fejlesztői csapat

A csapat tagjai:

csapattag neve	neve Neptun-kód	E-mail cím
Drahos Zsolt	UCZFU3	zsdrahos@gmail.com
Dittrich Ákos	ZEBSPO	dittrich.kos@gmail.com
Ködmön Dániel Ákos	C33KMB	dani.kodmon@gmail.com
Richter-Cserey Máttyás	NVIN7M	matyas.richtercserey@gmail.com

### Feladat kiosztás:

csapattag neve	Fő feladat
Drahos Zsolt	Frontend fejlesztő, Adatbázis
Dittrich Ákos	Frontend fejlesztő
Ködmön Dániel Ákos	Backend
Richter-Cserey Máttyás	MI fejlesztő, backend

Feladatkörökben a fejlesztés során lehetnek átfedések.

### Részletes feladatleírás

A projekt célja egy olyan ecommerce shopping assistant alkalmazás létrehozása, amely lehetővé teszi a felhasználók számára, hogy kényelmesen és hatékonyan böngésszenek és vásároljanak termékeket az online áruházban. Az alkalmazás számos funkciót kínál, beleértve a termékek böngészését, keresését, kosárba helyezését, rendelés leadását, valamint személyre szabott termékajánlásokat.

Az MI elemzi a felhasználók vásárlási szokásait és korábbi vásárlásaikat, majd olyan termékajánlásokat generál, amelyek illeszkednek az egyes felhasználók ízléséhez és preferenciáihoz. Ezáltal a felhasználók könnyedén felfedezhetnek új termékeket, amelyek érdekesek lehetnek számukra.

A felhasználói funkciók a következők:

- Regisztráció és bejelentkezés: A felhasználók regisztrálhatnak és bejelentkezhetnek az alkalmazásba.
- Termékek böngészése: A felhasználók böngészhetik az áruházban elérhető termékeket kategóriák, árak alapján.
- Kosárkezelés: A felhasználók kosárba helyezhetik a kiválasztott termékeket, módosíthatják a kosár tartalmát, és leadhatnak rendelést.
- Személyre szabott ajánlások: Az MI modul elemzi a felhasználók vásárlási szokásait és előző vásárlásaikat, majd személyre szabott termékajánlásokat kínál nekik.
- Profilkezelés: A felhasználók módosíthatják profilinformációikat.

### **Technikai paraméterek**

Az alkalmazás technikai paraméterei a következők:

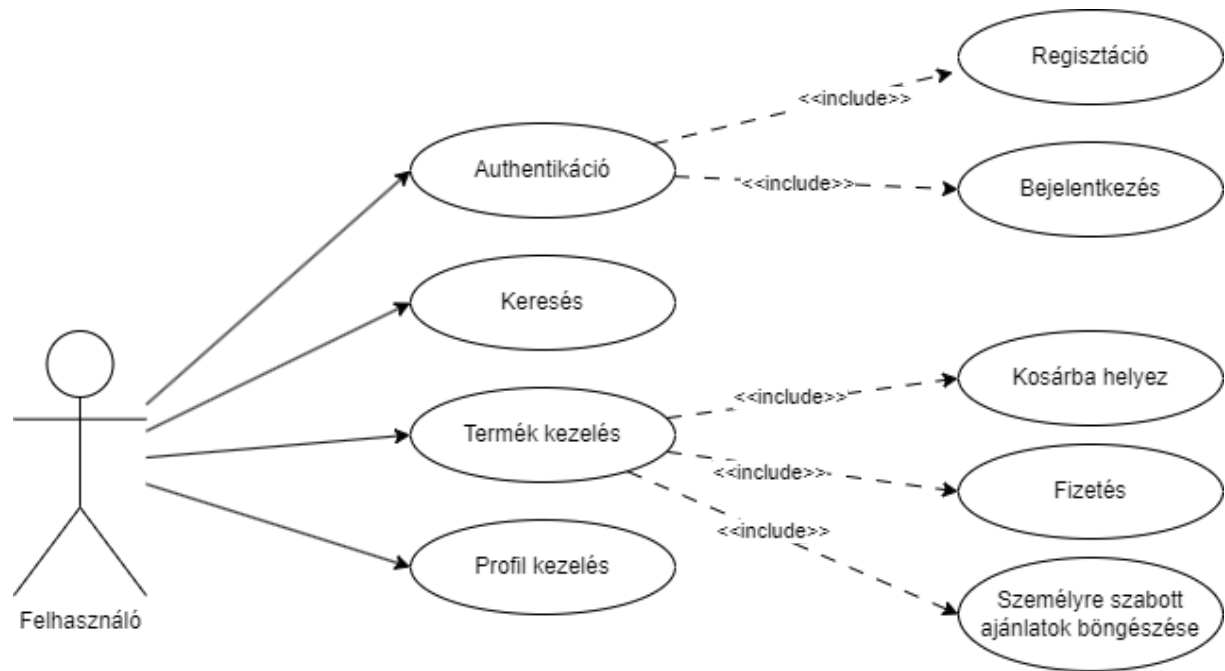
- Frontend: Angular keretrendszer használata a felhasználói felület kialakításához.
- Backend: Spring Boot keretrendszer használata a szerveroldali logika megvalósításához.
- Adatbázis: MySQL adatbázis használata a termék- és felhasználói adatok tárolásához.
- MI modul: Egy mesterséges intelligencia modul implementálása a személyre szabott termékajánlások generálásához, mely backenden fog futni.

### **Szótár**

- Termékkategória: Az áruházban elérhető termékek csoportosítása kategóriákba, például "ruházat," "elektronika," "sportfelszerelés," stb.
- Kosár: A felhasználó által megvásárlásra kiválasztott termékek gyűjteménye.
- Profilinformáció: A felhasználó által megadott személyes adatok, például nevük, e-mail címük, telefonszámuk stb.
- MI (mesterséges intelligencia) modul: Deep-learning alapú model, amely analizálja a felhasználók vásárlási szokásait és személyre szabott termékajánlásokat készít számukra.

## Essential Use-case-ek

Use-case diagram:



# **Ecommerce Shopping Assistant készítése MI támogatásával**

## **Rendszerterv**

Mesterséges intelligenciát alkalmazó megoldás fejlesztése 4  
Szoftverarchitektúrák tárgy házi feladat

Dittrich Ákos

Drahos Zsolt

Ködmön Dániel Ákos

Richter-Cserey Mátyás

# Tartalom

<b>Tartalom</b>	<b>5</b>
<b>Bevezetés</b>	<b>7</b>
<b>MI modul</b>	<b>8</b>
Felhasznált technológiák:	8
Adat:	8
Adat feldolgozás:	9
Tanító adatok:	9
A neurális háló felépítése és tanítása:	9
Kimenet és loss függvény:	10
Bemeneti előfeldolgozás:	11
Futtatás és eredmény interpretáció:	11
<b>Adatbázis</b>	<b>12</b>
ER diagram:	12
Adatok:	12
<b>Backend</b>	<b>13</b>
Rest API - Spring Boot	13
Model	14
Controller	15
Service	17
Repository	17
OpenAPI végpontleírás	17
<b>Frontend</b>	<b>18</b>
Angular structure	18
Dependency injection (DI)	19
Services:	19
Component:	19
Lazy loading	19
Angular Material	19
Grafikus felhasználói felület és funkciók	20
Terméklista:	20
Kosár nézet:	21
Profil nézet:	22
Bejelentkezés és regisztráció:	23
<b>Továbbfejlesztési lehetőségek</b>	<b>24</b>

<b>Telepítési útmutató</b>	<b>25</b>
Adatbázis: MySQL	25
Backend: Java Spring Boot	26
Frontend: Angular	26

# Bevezetés

Az online vásárlás az életünk szerves részévé vált, és az igények változásával egyre inkább az egyedi és személyre szabott élményekre törekszünk. Az e-kereskedelmi platformok fejlődése és elterjedése olyan új kihívásokat teremtett, melyek megoldására az E-commerce Shopping Assistant rendszer készült.

Az E-commerce Shopping Assistant egy átfogó alkalmazás, melynek fő célja a felhasználók online vásárlási élményének optimalizálása. A rendszer három fő komponensből áll: egy modern Angular alapú webshop frontendből, melyet a felhasználók azonosítás nélkül is böngészhetnek és vásárolhatnak; egy hatékony Spring Boot alapú backendből, mely gondoskodik az üzleti logika kezeléséről, és egy MySQL adatbázisból, ami az adatok tárolásáért és kezeléséért felelős.

Az E-commerce Shopping Assistant különlegessége az MI (mesterséges intelligencia) modul, mely a vásárlási szokások elemzésére épülve személyre szabott termék ajánlásokat generál a felhasználók számára. Ez a funkció lehetővé teszi, hogy a felhasználók még hatékonyabban és gyorsabban találjanak rá azokra a termékekre, melyek valóban érdeklik és megfelelnek az egyedi igényeiknek.

Az alkalmazás összetett rendszere és az MI modul egyesítése lehetővé teszi a felhasználók számára, hogy széles termékkínálatból könnyedén és gyorsan válogathassanak, miközben személyre szabott ajánlásokat kapnak a saját vásárlási szokásaik alapján. Az E-commerce Shopping Assistant nem csupán egy újabb online vásárlási platform, hanem egy olyan eszköz, amely átalakítja és optimalizálja a vásárlási élményt, elősegítve a felhasználók számára az egyszerűbb, gyorsabb és személyre szabottabb vásárlást az online térben.

# MI modul

## Felhasznált technológiák:

A háló tanító script Python 3.7-ben készül, a következő csomagok felhasználásával:

Tensorflow (2.10.1), Keras, Pandas, Numpy, Scipy

## Adat:

Az adatok az amazon 2018-as termék adatbázisából származnak. A nyers adatok kb. 15GB terjedelemmel rendelkeznek. Ebben benne vannak a következő kategóriák:

- Háztartási gépek
- Mobil telefonok és kiegészítők
- Elektronika
- Filmek és sorozatok
- Szoftver
- Videó játékok

Az adatok ezeket a mezőket tartalmazzák:

- asin
- title
- feature
- description
- price
- imageURL
- imageURL
- related
- salesRank
- brand
- categories
- also\_buy
- also\_viewed

Ezek közül az MI szempontjából az asin, title, also\_buy és also\_view mezők a fontosak. Az utóbbi kettő az adott termékkel együtt vásárolt és megtekintett termékek azonosítói találhatóak, az asin a termék azonosító (10 karakter, számok és nagy betűk), a title pedig a termék neve.



## Adat feldolgozás:

Az adatok eléggé irregulárisak, hiányosak, így előfeldolgozást igényelnek. Az előfeldolgozáshoz kiszedjük az üres és rosszul formált adatokat. Ezen kívül kidobjuk azokat a termékeket amelyeknek az `also_buy` és `also_view` mezője is üres. Ezekből a listákból kivesszük azokat az azonosítókat amik nem szerepelnek a felhasznált kategóriákban (vagyis nem mutatnak ki az adathalmazból).

Az `also_buy` és `also_view` mezőket összevontjuk "`related_products`" néven és ezt felhasználva generálunk a neurális hálónak tanító adatot.

## Tanító adatok:

A hálónak adathármasokat kell generálnunk, melyeknek a struktúrája a következő: egy viszonyítási alap ("`anchor`"), egy pozitív és egy negatív minta. Minden termékhez annyi ilyen hármaszt készítünk, ahány hozzá kapcsolódó termék van (`related_products`). Az adott termék lesz az `anchor`, a kapcsolódó termékek a pozitív minták és negatív mintának pedig választunk egyet a többi termék közül. Egy-egy minta a termék kódjából és nevéből áll.

## A neurális háló felépítése és tanítása:

A hálónak két bemenete van. Egyik a termék azonosítót várja, a másik pedig a nevet. Ez a része a hálónak teljesen külön kezeli a két bemenetet.

Az azonosító kezelése jóval egyszerűbb, itt csupán egy `embedding` és egy `dense` réteget használunk a feldolgozásra.

A terméknev kezelése kissé összetettebb. Itt a szöveg feldolgozásra, manapság igen népszerű, `transformer` architektúrát alkalmazzuk. A bemenetet előbb vektorizáljuk egy `TextVectorization` réteggel. Ezután `embedding` réteg következik, majd 2 `transformer encoder` réteg, 4 db 128-as méretű `attention head`-el, végül pedig egy `dense` réteg.

Ezekután a modell ét részét egyesítjük egy összefűzéssel, a végére pedig két további `dense` réteg kerül, az utolsó réteg normalizált változata pedig a kimenet. Természetesen a hálóban a megfelelő helyeken `dropout` és `batch normalization` rétegeket használunk.

## Kimenet és loss függvény:

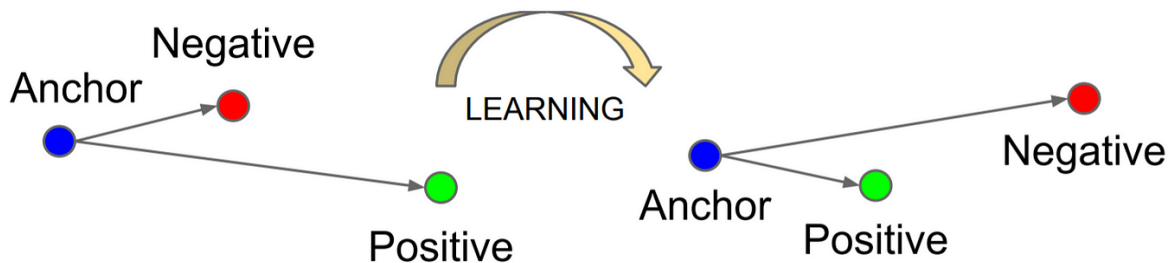
A háló a termékeket egy magas dimenziószámú vektortérben helyezi el, ennek megfelelően a kimenete egy 256 dimenziós vektor. Ezt úgy próbálja megtenni, hogy az egymáshoz kapcsolódó termékek minél közelebb kerüljenek egymáshoz, klasztereket formálva a térben. Ennek érdekében a Triplet Loss nevű veszteségfüggvényt alkalmazzuk, ami a következőképp működik.

A triplet loss három fő összetevőből áll: az "anchor" (horgony) példány, a "positive" (pozitív) példány és a "negative" (negatív) példány. Az anchor példány egy bemeneti minta, aminél azt szeretnénk, hogy a háló képes legyen megtanulni a hasonlóságot más példányokkal. A positive példány egy hasonló példány az anchorhoz, míg a negative példány egy eltérő példány, amely különbözik az anchor-tól.

A triplet loss célja, hogy minimalizálja az anchor és a positive példány közötti távolságot, és maximalizálja az anchor és a negative példány közötti távolságot. A veszteség függvénye matematikailag így néz ki:

$$L(A, P, N) = \max(d(A, P) - d(A, N) + \alpha, 0)$$

ahol  $d(A, P)$  az anchor és a positive példány közötti távolság,  $d(A, N)$  az anchor és a negative példány közötti távolság, és  $\alpha$  egy előre meghatározott küszöbérték, amely megakadályozza, hogy a hálózat egyszerűen mindent eltávolítson a negatív példányoktól.



A triplet loss segít a mély neurális hálózatnak abban, hogy olyan reprezentációkat tanuljon, amelyek hasznosak lehetnek a hasonlóságok és különbségek kinyeréséhez az input adathalmazban.

Ennek megfelelően a hálót meg kell háromszoroznunk és az új kimenet ezeknek a konkatenációja lesz, annak érdekében, hogy a hármasokat párhuzamosan tudjuk futtatni. Ugyanakkor ehhez a modell ugyanazon példányát használjuk, hogy a súlyok azonosak legyenek a három közt.

## Bemeneti előfeldolgozás:

A termék azonosító egy 10 számból és karakterből álló string, pl. B00002N7HY. Ezt valahogy egyértelműen le kell fordítanunk a modell által értelmezhető számra, úgy, hogy az lehetőleg könnyen bővíthető legyen, és ne függjön a termékek sorrendjétől (ami nem igaz pl. egy egyszerű növekvő id alapú look-up-table-re). Kihashálhatjuk viszont, hogy az azonosító csak számokat és betűket tartalmaz, tehát tudjuk a lehetséges értékkészletét ( $10 + 26 = 36$ ) és hosszát. Így át tudunk rá tekinteni, mint 36-os számrendszerbeli számra, ez a megfeleltetés pedig egyértelmű és könnyedén bővíthető. Az átalakítás után a kapott számot normalizáljuk 0 és 1 közé, majd visszaalakítjuk stringé, mert a modell így várja a bemenetet (az egységesség érdekében).

A termék név feldolgozása csupán a központozási karakterek eltávolításából áll.

## Futtatás és eredmény interpretáció:

A modell célja, hogy bizonyos termékek megadásával visszakapjuk az azokhoz leginkább hasonló termékeket. Mivel a modell a vektortérben a termékeket hasonlóság szerint helyezi el, már csak a kimeneti vektorok interpretációjával kell foglalkozni. Ez úgy oldjuk meg, hogy az összes létező termékhez legeneráljuk a hozzá tartozó vektort, ezeket pedig elmentjük. Futásidőben az aktuálisan feldolgozandó termékekre külön-külön lefuttatjuk a modellt (a valóságban persze ez batch-elve történik, így a modellnek csak egyszer kell futnia, mindegyikre párhuzamosan, amennyiben megfelelően kis számú bemenet van). Ezután a kapott vektorokat összeadjuk és normalizáljuk (vagyis lényegében kiszámoljuk az átlagukat), végül pedig a kapott vektor távolságát kiszámítjuk az összes eltárolt vektorhoz viszonyítva. A vektortérben ehhez legközelebb eső vektorokhoz tartozó termékek lesznek a leghasonlóbbak. Ebből kiválasztva az N legközelebbit, meg is van az ajánlásunk.

# Adatbázis

A rendszer alapját képező adatbázis a következő táblákból áll:

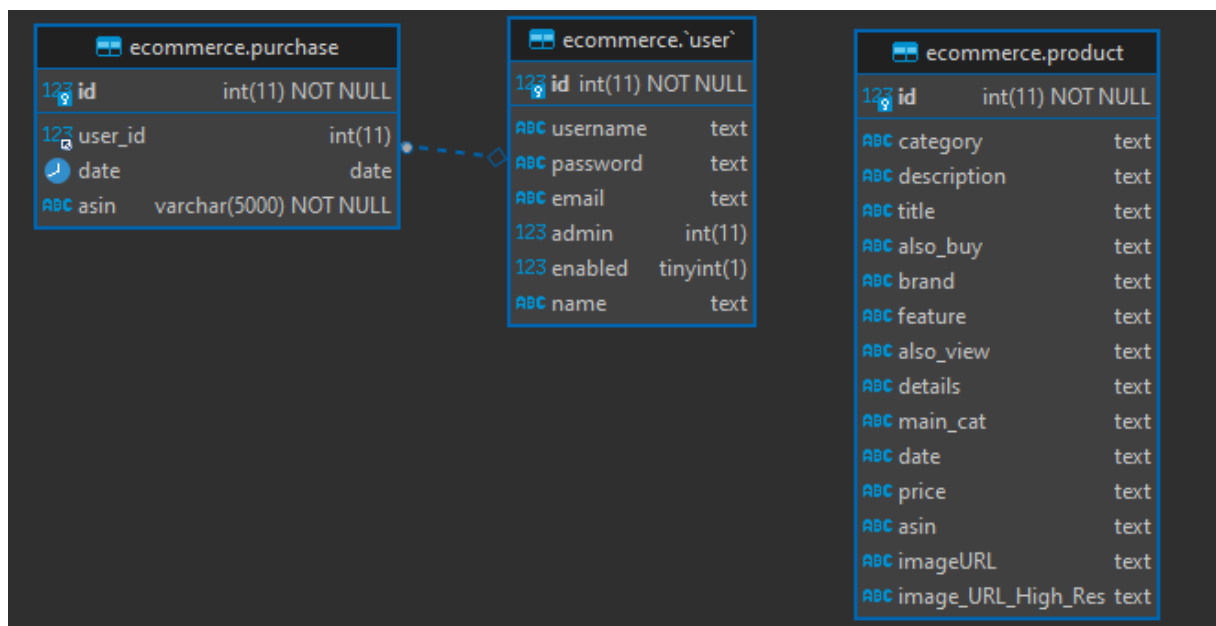
**product tábla:** Az e-kereskedelmi rendszer termékeinek adatait tárolja.

**user tábla:** A felhasználók adatait tárolja.

**purchase tábla:** A vásárlások adatait tárolja. A purchase tábla user\_id oszlopa idegen kulcsként hivatkozik a user tábla id oszlopára, így kapcsolódik a vásárlások a felhasználói adatokhoz.

Ez az adatbázis szerkezet lehetővé teszi a termékek, felhasználók és vásárlások hatékony kezelését és nyilvántartását az e-kereskedelmi rendszer számára.

## ER diagram:



## Adatok:

Adatbázisban illetve az MI tanításához megközelítőleg 170.000 rekord adatot használtunk fel, melyek az [Amazon Review Data \(2018\)](https://cseweb.ucsd.edu/~jmcauley/datasets/amazon_v2/) nevű adatkészletből származnak. Ezek az adatok különböző termékek információt, további megvásárolt termékek azonosítóit és egyéb releváns információkat tartalmaznak az Amazon platformról. Az adatkészlet lehetőséget biztosított számunkra arra, hogy változatos termékkategóriákból és vásárlói viselkedésből származó adatokat felhasználjunk az MI modell tanításához és validálásához.

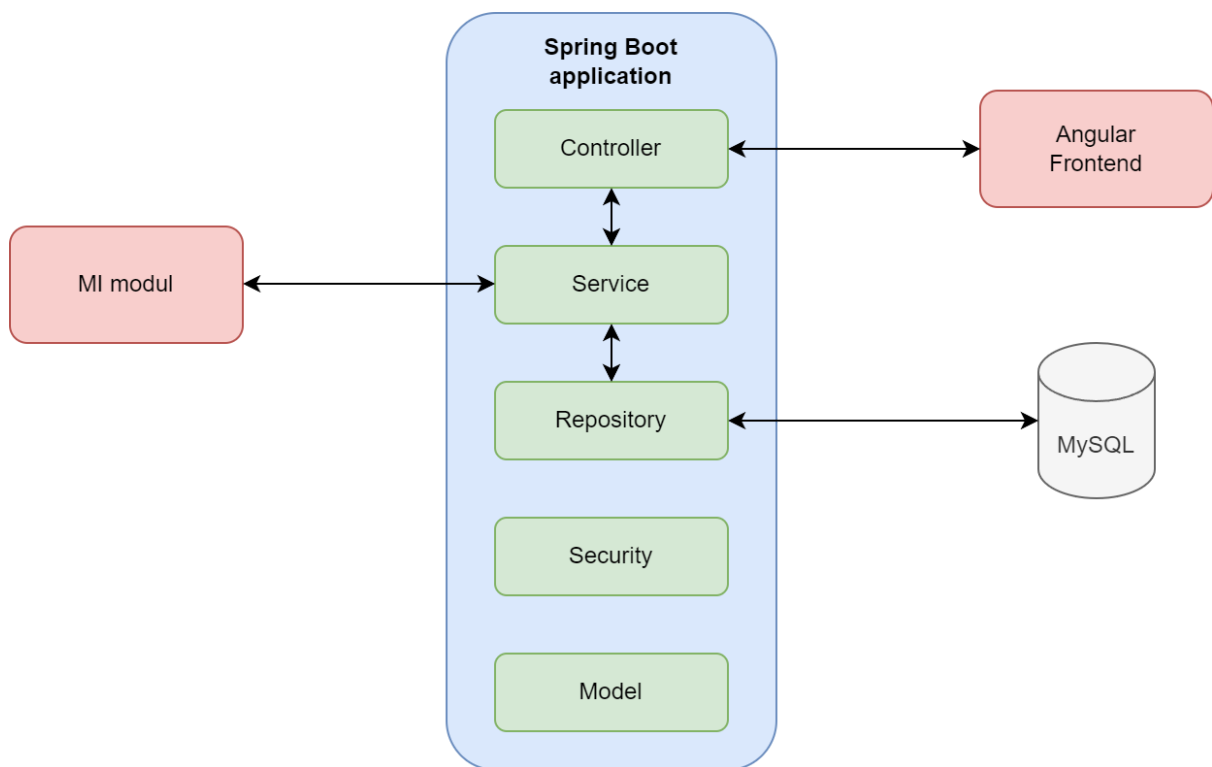
Adatok: [https://cseweb.ucsd.edu/~jmcauley/datasets/amazon\\_v2/](https://cseweb.ucsd.edu/~jmcauley/datasets/amazon_v2/)

# Backend

## Rest API - Spring Boot

Java Spring Boot egy olyan, nyílt forráskódú eszköz, amely megkönnyíti a mikroszolgáltatások és webalkalmazások Java-alapú keretrendszerek használatával való létrehozását. A beépített HTTP-szerverrel és fejlesztőeszközökkel a telepítés és a fejlesztés is lehetővé válik. A Spring Boot továbbá támogatja a RESTful elveket, és könnyen kezelhetővé teszi az útvonalakat, a HTTP-módszereket, valamint a formátumokat, például a JSON-t. <sup>1</sup>

Az Spring Boot alkalmazás felépítése a következő:



---

<sup>1</sup> <https://azure.microsoft.com/hu-hu/resources/cloud-computing-dictionary/what-is-java-spring-boot>

Az autentikációt a Spring Security valósítja meg. Egy Basic autentikáció lett felkonfigurálva és a felhasználók adatai saját adatbázis táblába kerülnek elmentésre.

Ahhoz hogy egy végpontot elérjünk a következő autentikációs headert kell tartalmaznia a kérésnek, ahol a base64() a tartalom base64 típusú kódolását jelenti.

*"Authentication" : "Basic base64("username:password")"*

## Model

Az alkalmazás Model könyvtárában találhatóak az alkalmazás működéséhez szükséges model osztályok. Ezek a modelosztályok a következők:

**User** model. Egy felhasználót reprezentál.

Long <b>id</b>	Adatbázis egyedi azonosítója
String <b>username</b>	A felhasználó felhasználóneve
String <b>password</b>	A felhasználó (hash-elt) jelszava
boolean <b>enabled</b>	A felhasználó engedélyezve van-e (bejelentkezés tiltása)
String <b>name</b>	Felhasználó teljes neve

**Product** model. Egy terméket reprezentál.

Integer <b>id</b>	Adatbázis egyedi azonosítója
String <b>category</b>	A termék kategóriáinak listája
String <b>description</b>	A termék leírása
String <b>title</b>	A termék neve
String <b>brand</b>	A termék márkája
String <b>feature</b>	A termék tulajdonságainak listája
String <b>main_cat</b>	A termék fő kategóriája
String <b>price</b>	A termék ára
String <b>asin</b>	A termék egyedi azonosítója
String <b>imageURLHighRes</b>	A termékhez tartozó kép URL címe

**ProductResponse** model. Szűrhető lapozható terméklista visszaadására szolgáló modelosztály.

Integer <b>allItems</b>	A keresési feltételeknek megfelelő összes létező elemek száma
Integer <b>page</b>	Az aktuálisan kért oldal száma
Integer <b>pageCount</b>	A keresési és lapozási feltételeknek megfelelő összes oldalak száma
List<Product> <b>products</b>	A keresési és lapozási feltételeknek megfelelő elemek az aktuális oldalon

**Purchase** model. Egy termék vásárlást reprezentál.

Integer <b>id</b>	Adatbázis egyedi azonosítója
Integer <b>user_id</b>	A felhasználó azonosítója aki megvette a terméket (idegen kulcs User id-ra)
String <b>asin</b>	A megvásárolt termék egyedi id-ja
Date <b>date</b>	A vásárlás dátuma

## Controller

A Controller könyvtárában találhatóak a REST végpontok megvalósításai.

Az alkalmazáshoz kapcsolt frontend, jelen esetben Angular weboldal a kapcsolatot a szerverrel az API REST végpontjain keresztül éri elé. Backenden a végpontok két csoportra vannak osztva. Ezek a:

- ApplicationController
- ProductController

Az **App controller**-hez tartoznak az általánosabb jellegű végpontok:

### GET api/login

Megfelelő felhasználónév és jelszó páros megadásával visszaadja a felhasználó adatait. Sikertelen autentikáció esetén (mint minden végpont eséréen) HTTP 401 üzenet érkezik.

## **POST** api/register

Az egyetlen autentikáció nélkül, publikusan elérhető végpont. A kérés body-ba megadható a regisztrálni kívánt felhasználó adatai.

A **Product controller**-hez tartoznak a termékekhez, vásárláshoz kapcsolódó végpontok.

## **GET** api/products/{id}

Egy megadott adatbázis id alapján lekéri az adott id-hoz tartozó termék adatait.

## **DELETE** api/products/{id}

Egy megadott adatbázis id alapján törli az adott id-hoz tartozó terméket.

## **GET** api/products/asin/{asinId}

Egy megadott egyedi termékazonosító alapján lekéri az adott azonosítóhoz tartozó termék adatait.

## **GET** api/products

Végpont a szűrhető, lapozható terméklista lekéréséhez. A következő URL query-k adhatóak meg:

- **pageSize** (kötelező): Hány elemet adjon vissza, mekkora az oldalméret frontenden.
- **page** (kötelező): Hányadik oldalt szeretnénk lekérni. (lapozással határozzuk meg)
- **qTitle**: Szűrés a termékek nevére (title mező)
- **qCat**: Szűrés a termékek kategóriájára (categories mező)
- **qMinPrice**: Szűrés termékekre minimum ár beállításával.
- **qMaxPrice**: Szűrés termékekre maximum ár beállításával.

## **POST** api/products/order/{asinId}

Egy termék megvásárlására szolgáló végpont. A megadott azonosítóval létrehoz egy rekordot az adatbázis *purchase* táblájában a megadott termék azonosítóval, a vásárló felhasználó azonosítójával és az aktuális dátummal.

## **GET** api/products/previous-orders

Egy felhasználó legutolsó 5 megvásárolt termékének a részleteit kérdezi le a végpont.

## **GET** api/products/user-recommendations

Visszaad 4 terméket amelyet az MI modul a felhasználó korábbi vásárlásai alapján ajánl.

## **POST** api/products/cart-recommendations

Visszaad 4 terméket amelyet az MI modul a kosárban lévő elemei alapján ajánl.



## Service

Az Service létesít kapcsolatot az alkalmazás Controller és Repository rétege között. Itt kapcsolódik az alkalmazáshoz az MI modul is (python szkript futtatása). A Service réteg valósítja meg az egyéb backend specifikus feladatokat, számításokat. Ilyen például az összes elem lapozhatóságának és kereshetőségének megoldása. A frontendről kapott információk alapján kiszámításra kerül hogy egy adott oldalméret és oldalszám alapján az SQL lekérdezést mekkora limittel és offsettel kell kiértékelni. Ilyen feladat továbbá a korábbi vásárlások azonosítói alapján összeállítani az azonosítókhoz tartozó termék részletek listájának összeállítása is.

## Repository

A repository valósítja meg az adathozzáférést az alkalmazás számára. Egy repository interface egy adatbázis táblához tartozik (User, Product, Purchase repositoryk). A repository interface-ek a JPA repository-t egészítik ki. Ennek segítségével elérhetőek az alap CRUD (create, read, update, delete) metódusok, illetve kiegészíthetjük egyedi natív SQL lekérdezésekkel is. Ilyen natív, egyedi lekérdezések például a lapozható, kereshető lista lekérdezés, vagy az egyedi *asin* azonosító alapján történő keresés.

## OpenAPI végpontleírás

A végpontokhoz OpenApi leírás is készült, amely a backend főkönyvtárában (src, pom.xml, stb. szintjén) *openapi\_ecommerce.json* néven elérhető. A Json tartalmat Swagger editorba beillesztve (<https://editor.swagger.io/>) megvizsgálhatóak a végpontok, illetve azok bemenetei, kimenetei.

# Frontend

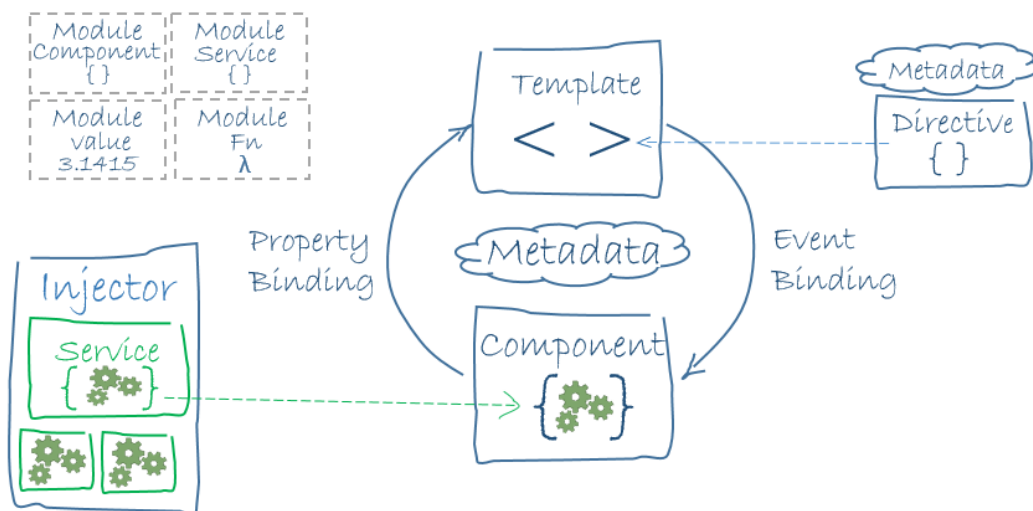
Az frontend alkalmazásunkban számos modern és hatékony architektúrális elemet alkalmazunk annak érdekében, hogy biztosítsuk a tiszta kódstruktúrát, az újrafelhasználhatóságot és a felhasználóbarát felhasználói felületet.

A felület alapvetően desktop nézet-re készült. Az Angular struktúráját követve, ahol a komponensek az alkalmazás építőkövei, különböző oldalak és funkciók különálló komponensekbe vannak csoportosítva.

## Angular structure

Az Angular egy TypeScript alapú framework amely rendkívül elterjedt dinamikus webes single page alkalmazások fejlesztésénél. A komponensek az Angular-alkalmazás építőkövei. Ezek valósítják meg az alkalmazás egy adott megjelenített részének logikáját és felhasználói felületét. Minden komponensnek saját sablonja, stílusa és metaadatai vannak. A template határozza meg a komponens nézetének szerkezetét és elrendezését. HTML-jelölést tartalmaz, amely Angular markup-al van kiegészítve. A komponensek és a template-ek binding segítségével interaktálnak.

A szolgáltatások olyan funkciókat biztosítanak, amelyek több komponensben is megoszthatók, és elősegítik a kód újrafelhasználhatóságát és a problémák elkülönítését. A service-ek lényegében olyan typescript osztályok amelyek nem kapcsolódnak szorosan egyetlen adott megjelenítési egység logikájához.



## Dependency injection (DI)

A dependencia injekció (DI) révén könnyen kezelhetjük a komponensek közötti függőségeket, például szolgáltatások használatát. Az Angular keretrendszer DI rendszere lehetővé teszi, hogy a szolgáltatások globálisan elérhetők legyenek, biztosítva ezzel az alkalmazás egyszerűbb tesztelhetőségét és kódolhatóságát. Különböző funkionalitásokat mint például a kosár kezelést, a termékeknel felmerülő szerver kommunikációt, vagy autentikációval kapcsolatos logikát mind igyekeztünk külön service-ekbe kiszervezni.

## Services:

Az API kérést végrehajtó szolgáltatások a komponensekben előforduló Observer mintákon keresztül valósulnak meg. Az Observer minta lehetővé teszi az eseményvezérelt kommunikációt a különböző alkotóelemek között. Amikor egy HTTP kérést kell kezdeményezni, például egy termék lekérdezése vagy a kosár frissítése céljából, a megfelelő szolgáltatás ezt végzi el. Az eredményt az Observer mintákon keresztül közvetíti a komponensek felé. A komponensek előfizetnek ezekre az Observer-ekre, és reagálnak az eseményekre, például frissítik a felhasználói felületüket az új adatokkal, biztosítva a dinamikus és aszinkron működést. Ez a tervezési minta hozzájárul az alkalmazás hatékony működéséhez és a kód tisztaságához.

## Component:

Az újrafelhasználható komponensek révén a fejlesztés hatékonyabbá válik, mivel a különböző részekből álló alkalmazásban könnyedén használhatunk már meglévő komponenseket. Ez növeli a karbantarthatóságot és csökkenti az ismétlődő kódrészeket. Külön komponensbe szerveztük a például a terméket megjelenítő kártyákat, valamint a dialógusokat amikor több helyen is újrahasznosítunk.

## Lazy loading

A Lazy Loading segítségével optimalizálhatjuk az alkalmazás betöltési sebességét, mivel csak akkor tölt be egy adott modult vagy oldalt, amikor azt valóban szükséges. Ez javítja a felhasználói élményt, különösen nagyobb méretű alkalmazások esetén. A fejlesztés során törekedtem bizonyos logikailag összefüggő komponensek külön modulba szervezését és ennek megfelelően lazy loading-ot implementálni.

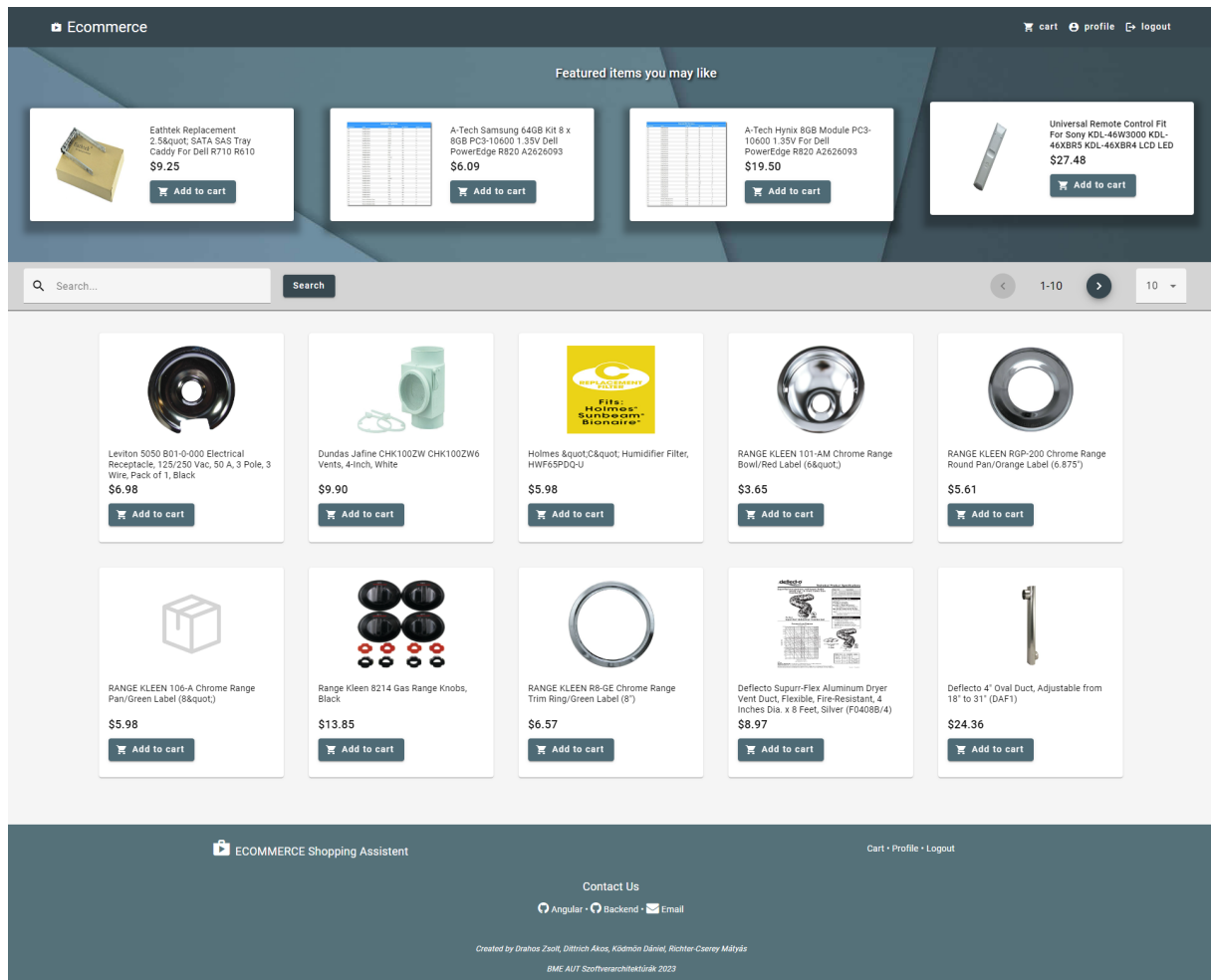
## Angular Material

A design tekintében főként angular material ui komponens framework-öt valamint natív css-t használtunk. Ez nagyban felgyorsította a fejlesztési időt valamint egységesebb végeredmény született design szempontból.

# Grafikus felhasználói felület és funkciók

A felület célja a felhasználónak könnyen használható felületet biztosítani.

## Terméklista:



A funkciók között szerepel a terméklista, ahol keresési lehetőségekkel és lapozási funkcióval navigálhatsz. Az ajánlott termékek rész olyan választékot mutat be, amelyek a vásárlási szokásaid alapján személyre szabottan megjelenített termékeket tartalmaz.

Ecommerce

🛒 cart

👤 profile

🚪 logout

My Shopping Cart:

1


View cart items

2

Fill out shipping information

3


Payment




Certified Appliance Accessories 3-Wire Open-Eyelet 30-Amp Dryer Cord, 4ft

Certified Appliance Accessories

\$14.75






Deflecto Dryer Lint Trap Kit, Supurr-Flex Flexible Metallic Duct

Deflecto


\$13.90



Subtotal: \$28.65

Next

Featured items you may like




3.5" USB External Floppy Disk Drive Portable 1.44 MB FDD for PC Windows 2000/XP/Vista/Windows 7/8/10

Dustproof Scratch-Resistant External Bag Case.No External

\$11.88


Add to cart



KuWIFI 4G LTE Pocket WiFi Router Unlocked LTE 4G Mobile WiFi Hotspot Portable 4G Router with sim Card Slot Goods for Travel and Business Trip Support LTE FDD B1/B3/B5

\$4.01


Add to cart



Unlocked 4G Huawei Ascend Mate7 MT7-TL10 6.0 Inch EMUI 3.0 Smart Phone Hisilicon Kirin 925 8 Core RAM 3GB ROM 32GB Dual SIM FDD-LTE WCDMA GSM - International

\$500.00

Add to cart



Esync USB Floppy Drive 3.5" USB External Floppy Disk Drive Portable 1.44 MB FDD USB Drive Plug and Play for PC Windows 10 7 8 Windows XP Vista Mac Black

\$13.99

Add to cart

ECOMMERCE Shopping Assistant

Cart

Profile

Logout

Contact Us

Angular

Backend

Email

Created by Drahos Zsolt, Dittich Akos, Kódmón Dániel, Richter-Cserey Mátys

BME AUT Szoftverarchitektúra 2023


A kosár nézetben a felhasználó megnézheti a kosár tartalmát. itt található egy ajánlás rész ami bemutatja azokat a termékeket, amelyek érdekesek lehetnek a kosár tartalma alapján.

## Profil nézet:

Ecommerce


cartprofilelogout

Recently purchased items:




Qooltek Mini Digital Hygrometer Thermometer Indoor Humidity Monitor with Temperature Humidity Gauge Meter for Cars Incubators and Brooders Clim Pet (Fahrenheit)

\$9.98




Eiroal Humidity Monitor Digital Indoor Hygrometer - Thermometer - Alarm Clock with LCD Display - Temperature Gauge Humidity Meter for Home or Greenhouse, Basement or Office

\$10.99




Eiroal Humidity Monitor Digital Indoor Hygrometer - Thermometer - Alarm Clock with LCD Display - Temperature Gauge Humidity Meter for Home or Greenhouse, Basement or Office

\$10.99



Freezer Meter Food Safety Thermometer, Traceable Temperature Monitor Without Wires or Batteries

\$7.99




Deflecto Dryer Lint Trap Kit, Supurr-Flex Flexible Metallic Duct

\$13.90

Test User

test@test.com



ECOMMERCE Shopping Assistant

Cart • Profile • Logout

Contact Us

Angular • Backend • Email

Created by Drághos Zsolt, Dittrich Akos, Kádár Dániel, Richter Cserey Máttyás

BME AUT Szoftverarchitektúrák 2023

A profilnézet lehetővé teszi a vásárlási előzmények megtekintését, így könnyen nyomon nyomon követhető, korábban vásárolt termékeket az alkalmazásban.

## Bejelentkezés és regisztráció:

Ecommerce

login

Welcome to Ecommerce Shopping Assistant!

login

sign up

username\*

theUser

password\*

\*\*\*\*\*

login

ECOMMERCE Shopping Assistant

Logout

Contact Us

[Angular](#) • [Backend](#) • [Email](#)

Created by Drághos Zsolt, Dittrich Akos, Kódmón Dániel, Richter-Cserey Mátys

BME-AUT-Softwarearchitektúrák 2023

Az ablakokon keresztül beléphez a felhasználó vagy új fiókot hozhatsz létre. Felhasználói fiók szükséges a teljes vásárlói élmény eléréséhez mint például a vásárlás, vásárlási előzmény vagy ajánlások megtekintéséhez.

# Továbbfejlesztési lehetőségek

- OAuth autentikáció backenden
- Részletesebb keresés / szűrő frontenden
- Új termék felvételének lehetősége frontenden
- Neurális háló további tanítása / fejlesztése, konkrét vásárlási adatok felhasználása



# Telepítési útmutató

## Adatbázis: MySQL

Hozz létre egy új adatbázist a projekt számára a következő paranccsal:

```
-- ecommerce.product definition

CREATE TABLE `product` (
  `category` text DEFAULT NULL,
  `description` text DEFAULT NULL,
  `title` text DEFAULT NULL,
  `also_buy` text DEFAULT NULL,
  `brand` text DEFAULT NULL,
  `feature` text DEFAULT NULL,
  `also_view` text DEFAULT NULL,
  `details` text DEFAULT NULL,
  `main_cat` text DEFAULT NULL,
  `date` text DEFAULT NULL,
  `price` text DEFAULT NULL,
  `asin` text DEFAULT NULL,
  `imageURL` text DEFAULT NULL,
  `image_URL_High_Res` text DEFAULT NULL,
  `id` int(11) NOT NULL AUTO_INCREMENT,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=169796 DEFAULT CHARSET=latin1 COLLATE=latin1_swedish_ci;

-- ecommerce.`user` definition

CREATE TABLE `user` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `username` text DEFAULT NULL,
  `password` text DEFAULT NULL,
  `email` text DEFAULT NULL,
  `admin` int(11) DEFAULT NULL,
  `enabled` tinyint(1) DEFAULT 1,
  `name` text DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=24 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;

-- ecommerce.purchase definition

CREATE TABLE `purchase` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `user_id` int(11) DEFAULT NULL,
  `date` date DEFAULT NULL,
  `asin` varchar(5000) NOT NULL,
  PRIMARY KEY (`id`),
  KEY `purchase_FK` (`user_id`),
  CONSTRAINT `purchase_FK` FOREIGN KEY (`user_id`) REFERENCES `user` (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=14 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
```

Adatok feltöltéséhez importálja be az output.json fájlt.

## Backend: Java Spring Boot

Rendszerkövetelmények:

- Java 17
- Maven: 3.3+

És a megfelelő URIk módosítása a saját elérési útvonalnak megfelelően.  
(*src/main/java/com/ecommerce/service/ProductService.java*)

```
ProcessBuilder builder = new  
ProcessBuilder("C:\\Users\\<user>\\AppData\\Local\\Programs\\P  
ython\\Python37\\python.exe", pythonScriptPath);  
  
String pythonScriptPath =  
"src\\main\\resources\\model_inference.py";
```

Spring Boot app futtatás: pl. Visual Studio / IntelliJ segítségével:

*src/main/java/com/ecommerce/EcommerceApplication* futtatása.

MI-hez szükséges követelmények:

- Python 3.7.0
- Tensorflow (pip install tensorflow)
- Numpy (pip install numpy)

**FONTOS!** **vectors.npy** fájl szükséges az MI futtatásához, azonban az AUT portál feltöltési méret limitje miatt ezt nem volt lehetséges feltölteni (hasonlóan az adatbázishoz). E-mail megkeresésre továbbítani tudjuk a fájlt.

## Frontend: Angular

Rendszerkövetelmények:

- Node js telepítése
- Npm ellenőrzése, hogy települt-e nodejs-el :
  - npm -v

A futtatáshoz a következő parancsokat kell kiadni az angular projekt src mappájában:  
(*ecommerce-angular/src*)

- npm install
- ng serve