# Feladat

Feladatunk egy alkalmazás készítése, amelyet több felhasználó fog használni. Online alkalmazásunkban a felhasználók termékeket vásárolhatnak, ehhez regisztrálhatnak is az oldalon, majd vásárolhatnak. Regisztráció nélkül is böngészhetnek.

Az alkalmazásnak több fő feladat ellátásával kell megküzdenie: a vásárlók rendelésének feldolgozása, illetve futárok kezelése.

# 1. Szereplők és igényeik

**Bejelentkezett felhasználó**

* Legyen képes vásárolni a rögzített termékekből
* A kosár tartalmát megtekinteni, módosítani, megvásárolni.
* Legyen képes saját adatait módosítani

**Látogató**

* Tudjon böngészni
* Legyen képes regisztrálni

**Adminisztrátor**

* Bejelentkezett felhasználók adatainak elérése
* Bejelentkezett felhasználók adatainak módosítása
* Bejelentkezett felhasználók adatainak törlése
* Termék adatok módosítása
* Termék adatok törlése
* Új termék felvétele

# 2. Use-Case Funkció lista

Az alkalmazásban a vásárlókat két csoportra tudjuk osztani. Ezek a csoportok a látogató és a regisztrált vásárló vagy egyszerűen vásárló. A két csoport között a regisztráció tesz különbséget, így hiába a látogató és a vásárló is el tudja végezni ugyanazt a webáruházban, mégis van némi különbség a lehetőségeik között.

## 2.1. Bejelentkezett felhasználó

* Böngészés
* A rögzített termékek között keresési lehetőség
* Kosár használata
* Kijelentkezés

## 2.2. Látogató

* Böngészés
* A rögzített termékek között keresési lehetőség
* Legyen képes regisztrálni (név, email, jelszó)
* Legyen képes bejelentkezni

A látogató és a bejelentkezett vásárló is tud az oldalon a termékek között böngészni, váltani. Mindketten képesek keresni is az oldalon terméket, és tudják a termékeket kategóriánként kilistázni.

A látogatónak ha vásárlási szándéka van, előbb regisztrálnia kell, hogy a kosárba rakott termékeket megvásárolhassa. A vásárló már nem regisztrálhat, mert regisztrált fiókkal rendelkezik.

Regisztráció után a látogató bejelentkezhet. Bejelentkezés után vásárlóként kezeljük a látogatót.

A regisztrált felhasználó termékeket tud helyezni kosarába.

Vásárolni csak a bejelentkezett felhasználó tud.

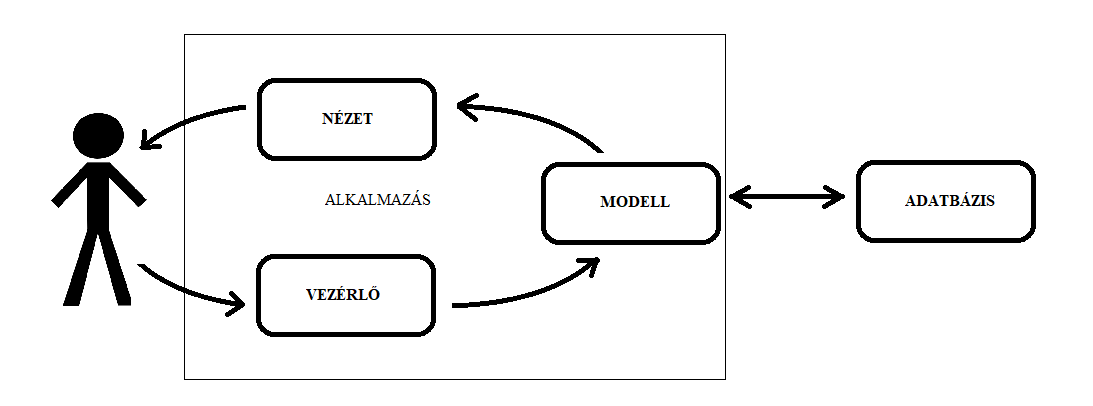
Kijelentkezés lehetőség csak a bejelentkezés után érhető el, ezért kijelentkezni csak a vásárló tud bejelentkezés után bármikor.

## 2.3. Adminisztrátor

* Az adminisztrációs felületet csak az adminisztrátor érheti el
* Listázni tudja a termékeket.
* Az összes típusú felhasználót tudja törölni, módosítani, kilistázni

# 3. Rendszerterv

A fejlesztés során az MVC szemléletmód figyelembevételével igyekeztünk elkülöníteni az adatok elérését és az üzleti logikát az adatok megjelenítésétől és a felhasználói interakciótól. Ezzel biztosítjuk azt, hogy a felhasználói felület ne befolyásolja a kezelést, illetve az adatok a későbbi fejlesztések során, ha kell, könnyen átszervezhetők legyenek a felhasználói felület változtatása nélkül.



MVC modell

# 4. Modellek

Az adatok tárolására több modellre is szükség van, úgy mint admin, regisztrált felhasználók, termék kategóriák, termékek, futárok, szolgáltatók és vezetőik.

**User**

* firstName (String)
* lastName (int)
* email (String)
* password (String)
* address (String)
* phone (String)
* userType (ENUM)

**Product**

* companyId (long)
* name (String)
* imageURL (String)
* description (String)
* category (ENUM)
* price (double)

**Company**

* userID (long) //CEO
* name (String)

**Login**

* email (String)
* password (String)

**userOrder**

* productIDs (List<Long>)
* totalPrice (double)
* orderStatus (ENUM)
* orderDate (LocalDate)
* date (date)

# 5. Tesztelés

A felhasználó szempontjából a rendszer futtatásához szükséges bármilyen operációs rendszer, amelyen futtatható HTML, CSS és JavaScript technológiával készült alkalmazás megjelenítésére képes böngésző.

A fejlesztés alatt a rendszer tesztelése desktop eszközön több böngészőben (Google Chrome, ZEN) Linux operációs rendszerek alatt történt.

A szerveroldalról nézve a rendszernek szüksége van egy olyan platformra, ami MySQL szervereket képes futtatni, a fejlesztés során ezt localhost-al oldottuk meg.

# 6. Bevezetés, éles üzemmód

* Az elkészült munka megtekintéséhez szükséges egy MySQL szervert futtatni localhost-on, a 3306-os porton. Telepítéskor ügyeljünk arra, hogy mint a felhasználónév, mint a jelszó „root” legyen.
* Szükség lesz továbbá a JetBrains IntelliJ Java IDE-re (lehet community vagy ultimate) a backend futtatásához
* Frontend esetén a WebStorm-t célszerű használni
* IntelliJ-ben nyissuk meg a productapi mappát és futassuk a StreakBackendApplication.java fájlt.
* A backend automatikusan hoz létre adatbázist a futó MySQL szerveren és tölti fel kezdeti adatokkal
* Frontend indítása előtt célszerű a fejlesztőkörnyezet termináljában futtatni az npm –i parancsot, aztán npm run dev-vel indítsuk is el
* A böngészőben a <http://localhost:5173-as> porton érhető majd el a webalkalmazás

# Összegzés

Az általunk fejlesztett REST Api alkalmazás minden CRUD funkciót kihasználva mutatja be a kliens és szerver oldal közötti http kommunikációt.

A fejlesztés során fontos szempont volt, hogy a jogkörök legyenek kezelve, admin és regisztrált felhasználó valamint látogató jogosultságainak elkülönítése az alkalmazás különböző rétegeihez.

Az elkészített alkalmazás nem tökéletes, további javításokra és fejlesztésekre szorul.

Sajnálatos módon számos funkció a fejlesztési idő rövidsége miatt nem kerülhetett bele a szoftverbe.