**PROJEKTMUNKA**

Info-Tech-Service kft weboldala

**Témavezető:** **Készítette:**

Balogh Zoltán Balogh Bálint

Szabó Ákos Juhászné Kovács Ildikó

Rékási László

NYÍREGYHÁZA

2022

Tartalomjegyzék

[Feladat megadása 2](#_Toc102338424)

[1. Szereplők és igényeik 2](#_Toc102338425)

[2. Use-Case Funkció lista 3](#_Toc102338426)

[2.1. Bejelentkezett felhasználó 3](#_Toc102338427)

[2.2. Látogató 3](#_Toc102338428)

[2.3. Adminisztrátor 4](#_Toc102338429)

[3. Magas szintű rendszerterv 5](#_Toc102338430)

[4. Screenek 6](#_Toc102338431)

[5. Modellek 8](#_Toc102338432)

[6. Alkalmazások kiválasztása 11](#_Toc102338433)

[6.1. Front-end 11](#_Toc102338434)

[6.2. Backend 11](#_Toc102338435)

[6.3. Adatbázis 12](#_Toc102338436)

[6.4. Tesztelés 12](#_Toc102338437)

[7. Routing 13](#_Toc102338438)

[8. Middlewarek (algoritmusok) 17](#_Toc102338439)

[9. Implementáció 19](#_Toc102338440)

[10. Tesztelés 20](#_Toc102338441)

[11. Bevezetés, éles üzemmód 21](#_Toc102338442)

[12. További fejlesztési lehetőségek 21](#_Toc102338443)

# Feladat megadása

Feladatunk egy alkalmazás készítése, amelyet több felhasználó fog használni. Online alkalmazásunkban a felhasználók termékeket vásárolhatnak, valamint szolgáltatásokat is igénybe vehetnek, ehhez regisztrálhat az oldalon a felhasználó, majd ezután vásárolhat, véleményezhet. Regisztráció nélkül is kereshet, olvashat az oldalon.

Az alkalmazásnak több fő feladat ellátásával kell megküzdenie: a vásárlók rendelésének feldolgozása, szolgálatás megrendelése, az üzlet árukészletének kezelése, és persze a számlázás. Mindezek webes felületen történnek meg.

# 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 adatainek módosítása
* Bejelentkezett felhasználók adatainek törlése
* Termék adatok módosítása
* Termék adatok törlése
* Új termék felvétele

# 2. Use-Case Funkció lista

Az alkalmazásban a felhasználókat vagy 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
* A rögzített termékek kilistázhatóak kategória alapján
* Vásárlói ’kosár’ használata
* Kijelentkezés

## 2.2. Látogató

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

A látogató és a bejelentkezett vásárló is tud az oldalon a termékek és szolgáltatások között böngészni, termékek között 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 regisztrációkor meg kell adnia bizonyos adatokat szállítással és számlázással kapcsolatban. A vásárló már nem regisztrálhat, mert regisztrált fiókkal rendelkezik.

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

A regisztált felhasználó tud terméket a kosárba helyezni, termék mennyiségét megadni.

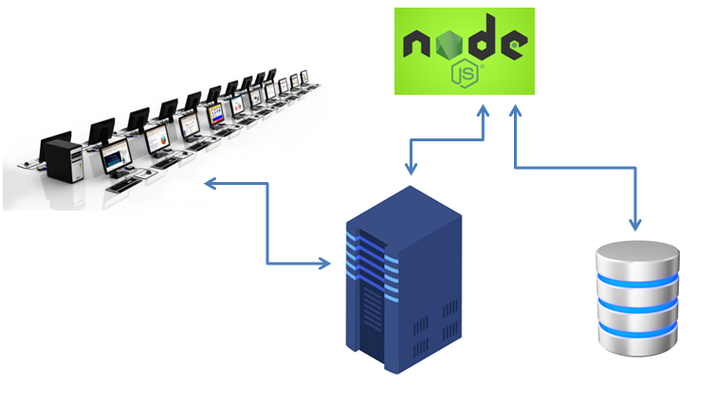
Vásárolni és szolgáltatást igényelni csak a bejelentkezett felhasználó tud, mivel szükségesek a regisztáláskor megadott adatok.

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

## 2.3. Adminisztrátor

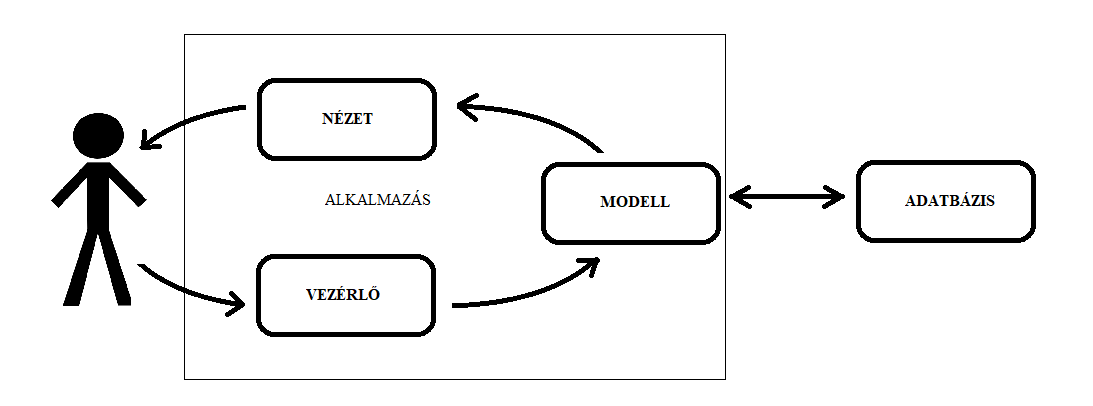
* Terméket adminisztációs felületen lehet feltölteni, módosítani és törölni
* Az adminisztrációs felületet csak az adminisztrátor érheti el
* Listázni tudja a termékeket.
* Felhasználókat tud törölni, módosítani, kilistázni

# 3. Magas szintű rendszerterv



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 átszervezhetőek legyenek a felhasználói felület változtatása nélkül.



MVC modell

# 4. Screenek

Egy internetes alkalmazás készítésekor fontos több szempontot is figyelembe venni. A végtermék olyan felhasználóknak készül, akikkel közvetlen kapcsolat nem áll fennt, így ha a felhasználó úgy érzi nem kapott kellő információt az oldalon, akkor lehet nem fog vásárolni, és vissza sem tér az oldalra. Ezért kiemelkedően fontos az oldal kinézetével és tartalmával kapcsolatban, hogy a látogatók bizalmát elnyerje, könnyen kezelhető és logikus felépítésű legyen. Minden információ egyértelmű legyen, például az elérhetőségek feltüntetése, a termékekről elég információ legyen feltűntetve, vagy például az árak ÁFÁ-val terheltek, vagy sem. Az ilyen alapvető adatok hiánya miatt a vásárló más boltot választhat.



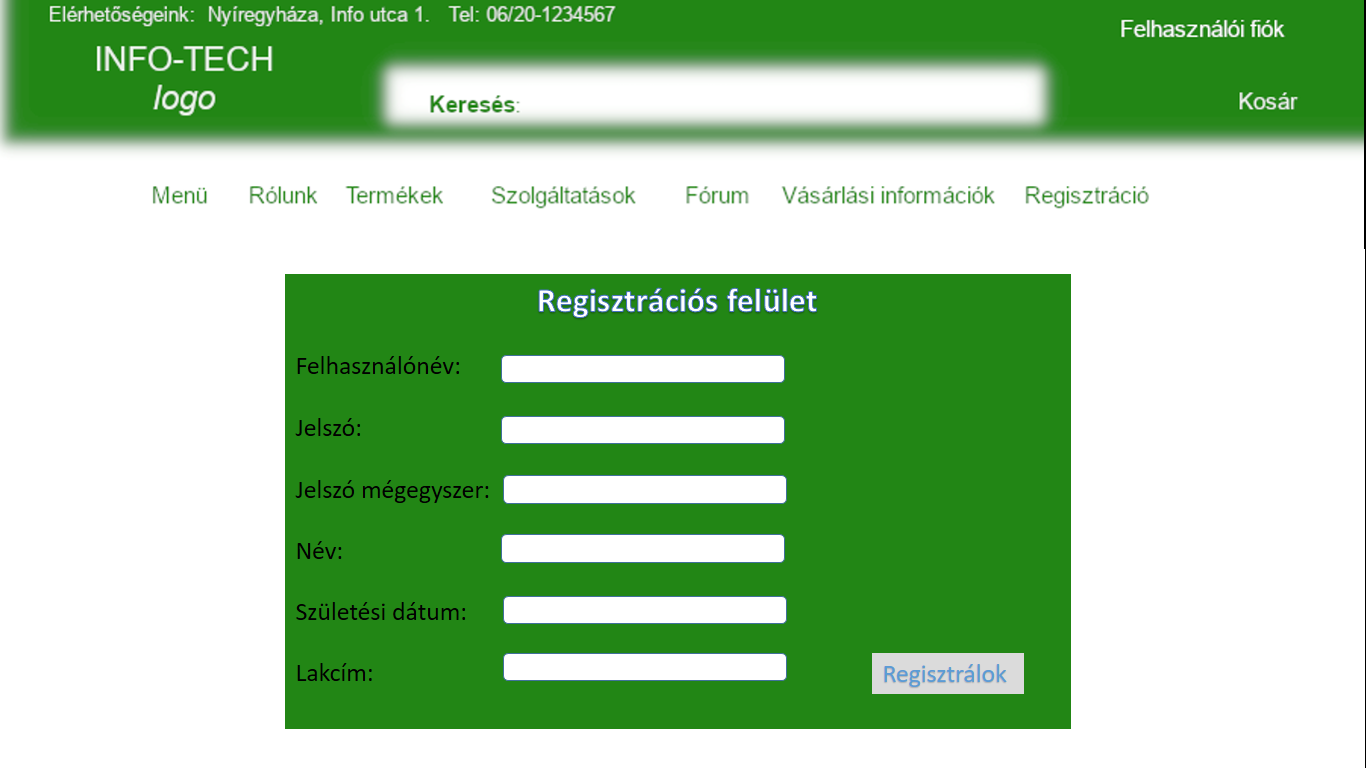
Kezdő képernyő terv

A termékek megjelenítésénél a kezdeti szakaszában még bootstrap slider-ben gondolkodott a csapat, de ezt elvetettük hamar, mert látványos volt, de rontotta az átláthatóságát a termékeknek. Végül az oldalon megjelenő termékeket kis dobozokban gondoltuk megjeleníteni, mert így egymás mellett és alatt több termék megjeleníthető és összehasonlítható, valamint az oldal kellően áttekinthető.



Termékek megjelenítésének oldal terve

A tervezés szakaszában is egyértelmű volt, hogy a regisztrációs felülethez egy letisztult, egyszerű megjelenésű oldalt képzelünk el.



# 5. Modellek

Az adatok tárolására több modellre is szükség van, úgy mint admin, regisztrált felhasználók, termék kategóriák, termékek, szolgáltatások.

**admin**{

* login (String)
* password (int)
* name (String)

}

**cart {**

* ID (int)
* userID (int)
* productID (int)
* serviceID (int)
* prod\_amount (int)
* date (date)

}

**categories** {

* ID (int)
* description (varchar)

}

**distribution** (

* catID (int)
* productID (int)
* name (varchar)
* amount (varchar)
* picture (varchar)
* description (varchar)
* vat\_id (int)
* net\_value (int)

}

**purchases** {

* ID (int)
* userID (int)
* productID (int)
* prod\_amount (int)
* date (date)
* shipping\_date (date)
* shipping\_adress (varchar)
* payment\_method (varchar)
* serviceID (int)

}

**services** {

* serviceID (int)
* description ( String)
* net\_value (int)
* vat\_id (int)

}

**suppliers** {

* ID (int)
* name ( String)
* reg\_number (String)
* tax\_reg (String)
* address (String)
* bank\_account (String)
* email (String)

}

**users** {

* userID (int)
* login (varchar)
* password (varchar)
* name (varchar)
* phone (varchar)
* birth (date)
* email (varchar)
* billing\_address (varchar)
* shipping\_address (varchar)
* tax\_reg (varchar)

}

**bills** (számlák) {

* bill\_id (int)
* productID (int)
* prod\_amount (int)
* serviceID (int)
* net\_value (int)
* gross\_value (int)
* vat\_id (int)

}

**bill\_head** {

* ID (int)
* userID( int)
* supplierID (int)
* date (date)
* total (int)

}

**bill\_switch**{

* ID (int)
* bill\_id (int)

}

# 6. Alkalmazások kiválasztása

## 6.1. Front-end

* **React**

A React az egyik nyílt forráskódú JavaScript könyvtár. Interaktív felhasználói felületek felépítésére szolgál. Ez egy hatékony, deklaratív és rugalmas könyvtár. A Model-View- Controller (MVC) V, azaz View elemével foglalkozik. Ez nem egy teljes keret, hanem csak egy előlapi könyvtár. Lehetővé teszi összetett felhasználói felületek létrehozását vagy elkészítését, összetevőkként ismert, elkülönített és apró kóddarabok felhasználásával.

(<https://hu.education-wiki.com/9050114-what-is-react>)

* **Javascript**

A JavaScript (röviden JS) egy objektumalapú scriptnyelv, amelyet elterjedten használnak weboldalakon, a web programozási nyelve. A Javascript futásideje egyszálú, ami azt jelenti, hogy egyszerre csak egy darab kódot képes végrehajtani. Ennek a megoldására használunk async függvényeket.

* **Html**

A HTML (Hypertext Markup Language) egy leíró nyelv, melyet weboldalak elkészítésére használhatunk.

* **Bootstrap**

A Bootstrap egy nyílt forráskódú keretrendszer (framework), mely **HTML, CSS, JavaScript technológiákat használ.** Nagyon könnyen, és minimális energia befektetéssel tudunk jól kinéző, bármilyen képernyőméreten szépen megjelenő weboldalakat készíteni a Bootstrap segítségével.

* **axios**

Az Axios egy könyvtár, amelyet HTTP -kérések küldésére használnak a böngészőből Node és Express.js platformon keresztül.

## 6.2. Backend

* **Node js**

A Node.js egy nyíltforrású, többplatformos, szerveroldali JavaScript futtatókörnyezet a Google Chrome V8 JavaScript*-*motoron alapulva. Leginkább webes applikációk, dinamikus weboldalak fejlesztésére használják, de akár szerveroldali feldolgozószkripek készítésére is kiválóan alkalmas.

(http://maas.hu/learning/it/prog/javascript/nodejs/#gsc.tab=0)

* **express**

Az **Express** (amit gyakran **Express.js**-nek, vagy **ExpressJS**-nek is hívnak) egy szerver oldali (back end) applikáció fejlesztői keretrendszer Node.js-hez. Használata és terjesztése ingyenes, szabad szoftver az MIT licensze alatt. Elsősorban webes applikációkat és API-kat fejleszthetünk vele, és rendkívül sikeres életútja során viszonylag hamar az általános webes applikációs keretrendszerré vált Node.js alatti fejlesztések esetében. (http://maas.hu/learning/it/prog/javascript/express/)

* **bcrypt** – hashelés (jelszó)

A Bcrypt egy népszerű és megbízható módszer a jelszavak kivonására.

* **nodemon – no restart**

A nodemon egy olyan eszköz, amely segít a node.js alapú alkalmazások fejlesztésében azáltal, hogy automatikusan újraindítja a csomópont-alkalmazást, ha a rendszer fájlváltozásokat észlel a könyvtárban.

* **MySql – node module**

A MySQL egy rendkívül népszerű relációs adatbázis-kezelő rendszer. A MySQL-t gyakran használják adatbázisként a Node-dal épített projektekhez és alkalmazásokhoz.

## 6.3. Adatbázis

* **Xampp**

Egy platformfüggetlen webszerver-szoftvercsomag, azok számára, akik adatbázissal támogatott webszervert szeretnének üzemeltetni, a webes alkalmazások készítését, tesztelését és futtatását oldja meg.

* **MariaDB**

A MariaDB Server az egyik legnépszerűbb nyílt forráskódú relációs adatbázis.

## 6.4. Tesztelés

* **Postman**

A Postman egy komplett eszköztár API fejlesztők számára. Olyan eszköz, amelyet az API-k megbízható teszteléséhez használhatunk. A Postman leegyszerűsíti az API életciklusának minden lépését, és egyszerűsíti az együttműködést.

# 7. Routing

GET /admin/

|  |  |
| --- | --- |
| tervezett feladat | Összes admin listázása |
| bemenet |  |
| kimenet | Az adminok listája |

GET /admin/:id

|  |  |
| --- | --- |
| tervezett feladat | Adott admin adatainak kislistázása |
| bemenet | paraméterként kapott azonosító |
| kimenet | Adott admin listája |

GET /users

|  |  |
| --- | --- |
| tervezett feladat | Összes felhasználó kislistázása |
| bemenet |  |
| kimenet | A felhasználók listája, adataik |

GET /users/:id

|  |  |
| --- | --- |
| tervezett feladat | Adott felhasználó adatainak kislistázása |
| bemenet | paraméterként kapott azonosító |
| kimenet | Adott felhasználó listája |

PUT /users

|  |  |
| --- | --- |
| tervezett feladat | Felhasználó rögzítése |
| bemenet | Felhasználó adatai |
| kimenet | visszajelzés a művelet sikerességéről |

PATCH /users

|  |  |
| --- | --- |
| tervezett feladat | Felhasználó adatainak módosítása |
| bemenet | módosítandó adatok |
| kimenet | visszajelzés a művelet sikerességéről |

GET /vat

|  |  |
| --- | --- |
| tervezett feladat | Áfa lekérdezése |
| bemenet |  |
| kimenet | Lebegőpontos érték (az aktuális áfa százalék értéke) |

POST /cart

|  |  |
| --- | --- |
| tervezett feladat | Adott felhasználó kosarának tartalmát listázza |
| bemenet | Felhasználó azonosító |
| kimenet | Felhasználó kosara |

PUT /cart

|  |  |
| --- | --- |
| tervezett feladat | Termék rögzítése a kosárban |
| bemenet | Egy adott termék azonosítója, darabszáma, felhasználói azonosító, hozzáadás dátuma |
| kimenet | Visszajelzés a művelet sikerességéről |

DELETE /cart

|  |  |
| --- | --- |
| tervezett feladat | Termék törlése a kosárból |
| bemenet | Felhasználó azonosító, termék azonosító |
| kimenet | Visszajelzés a művelet sikerességéről |

GET /search

|  |  |
| --- | --- |
| tervezett feladat | Keresés összes termékre |
| bemenet |  |
| kimenet | lista az összes termékről (átirányítás a /products-ra) |

GET /search/:key

|  |  |
| --- | --- |
| tervezett feladat | Keresés kifejezés alapján |
| bemenet | kulcs kifejezés |
| kimenet | Minden termék, melynek a nevében vagy leírásában megtalálható a kifejezés |
|  |  |

GET /products

|  |  |
| --- | --- |
| tervezett feladat | Összes termék kislistázása |
| bemenet |  |
| kimenet | Az összes termék listája |

GET /products/:key

|  |  |
| --- | --- |
| tervezett feladat | Adott kategóriájú termék listázása |
| bemenet | paraméterként kapott útvonal |
| kimenet | Adott kategórájú termék listája |

GET /products/:id

|  |  |
| --- | --- |
| tervezett feladat | Adott termék kislistázása |
| bemenet | Termék azonosítója |
| kimenet | Adott termék listája |

DELETE /product

|  |  |
| --- | --- |
| tervezett feladat | Adott termék törlése |
| bemenet | Terméknév |
| kimenet | Visszajelzés a törlés sikerességéről |

PUT /product

|  |  |
| --- | --- |
| tervezett feladat | Új termék rögzítése |
| bemenet | Termék adatai |
| kimenet | Visszajelzés a termék felvételének sikerességéről |

PATCH /product

|  |  |
| --- | --- |
| tervezett feladat | Adott termék adatának módosítása |
| bemenet | Módosítandó adat |
| kimenet | Visszajelzés a módosítás sikerességéről |

GET /services

|  |  |
| --- | --- |
| tervezett feladat | Összes szolgáltatás listázása |
| bemenet |  |
| kimenet | A szolgáltatások listája |

GET /services/:id

|  |  |
| --- | --- |
| tervezett feladat | Adott szolgáltatás kislistázása |
| bemenet | Szolgáltatás azonosítója |
| kimenet | Adott szolgáltatás listája |

GET /categories

|  |  |
| --- | --- |
| tervezett feladat | Összes kategória listázása |
| bemenet |  |
| kimenet | a kategóriák listája |

GET /categories/:id

|  |  |
| --- | --- |
| tervezett feladat | Adott kategória kilistázása azonosító alapján |
| bemenet | Kategória azonosítója |
| kimenet | Adott kategória azonosító alapján |

POST /login

|  |  |
| --- | --- |
| tervezett feladat | Authentikáció |
| bemenet | felhasználónév, jelszó |
| kimenet | Felhasználónév és egy jogosultságot meghatározó érték (isAdmin) |

DELETE /user

|  |  |
| --- | --- |
| tervezett feladat | Töröl egy felhasználót a bejelentkezési név megadásával |
| bemenet | Felhasználó bejelentkezési neve |
| kimenet | Visszajelzés a művelet sikerességéről |

DELETE /user/:id

|  |  |
| --- | --- |
| tervezett feladat | Töröl egy felhasználót azonosító alapján |
| bemenet | Felhasználó azonosítója |
| kimenet | A törölt felhasználó renderelt listája |

# 8. Middlewarek (algoritmusok)

A Middleware függvények definíció szerint olyan függvények, amik hozzáférnek a request (kérés - req) és response (válasz - res) objektumokhoz és kezelik azokat. (http://www.inf.u-szeged.hu/~tarib/javascript/webszerverek.html#middleware-fuggvenyek-hibakezeles)

Az alkalmazás sok kérést szolgál ki, ezért több Middleware-re is szükség volt.

const {userList,user,products,product,search,admins,services,categories,vat,cart} = require('../middlewares/userMWs/userMW');

const {addProduct,admin} = require('../middlewares/adminMWs/adminMW');

const {login} = require('../middlewares/userMWs/userAuth');

**admins()** - adminok listáját adja vissza

**admin()** - adott admin adatait listázza ki

**userList()** - felhasználók lekérdezése

**user()** - Ha a request method:

* PUT, hozzáad egy usert a DB-hez
* DELETE, töröl egy usert
* PATCH, módosít egy usert
* Különben header param alapján visszadob 1 adott usert

**login()** - bejelentkezés

**vat()** - Áfa %-ot kérdez le, és alakít át (a bruttó számítás logikája a front-end-en)

**cart()** - POST - egy felhasználóhoz tartozó termékeket adja vissza

* PUT - hozzáad egy terméket a kosárhoz
* DELETE - töröl egy terméket a kosárból

**search(**) - Ha van header param, akkor azt használja kulcsszónak, különben átirányít az összes termék listázására

**products()** - Ha van header param:

* 'param' esetén a csak a 'param' kategóriájú termékeket kérdezi le
* Különben az összes termék lekérdezése

**product()** - Ha a request method:

* GET - header param alapján egy termék adatait kérdezi le
* PATCH - req.body elemei alapján frissít
* DELETE - név alapján egy terméket töröl

**addProduct()** - termék hozzáadása // csak admin jogosultság

**services()** - Ha van header param, akkor ID szerint szolgáltatás, különben az összes

**categories()** - Ha van header param, akkor adott ID-vel rendelkező kategóra adatai, különben az összes kategória a válasz

# 9. Implementáció

Az implementáció a tervezést követő folyamat. A tervezés során megtörténik a feladatokra bontás, valamint az egyes feladatokban érintett funkcionalitás és szerkezet modellezése. Az implementáció során a modellekből készítünk működő, futtatható kódot.   
(https://okt.sed.hu/rf2/gyakorlat/tananyag/implementacio/implementacio/)

# 10. Tesztelés

A felhasználó szempontjából a renszer futtatásához elegendő bármely olyan operációs rendszer, amelyen futtatható HTML, CSS és Javascript technológiával készült alkalmazás megjelenítésére képes böngésző.

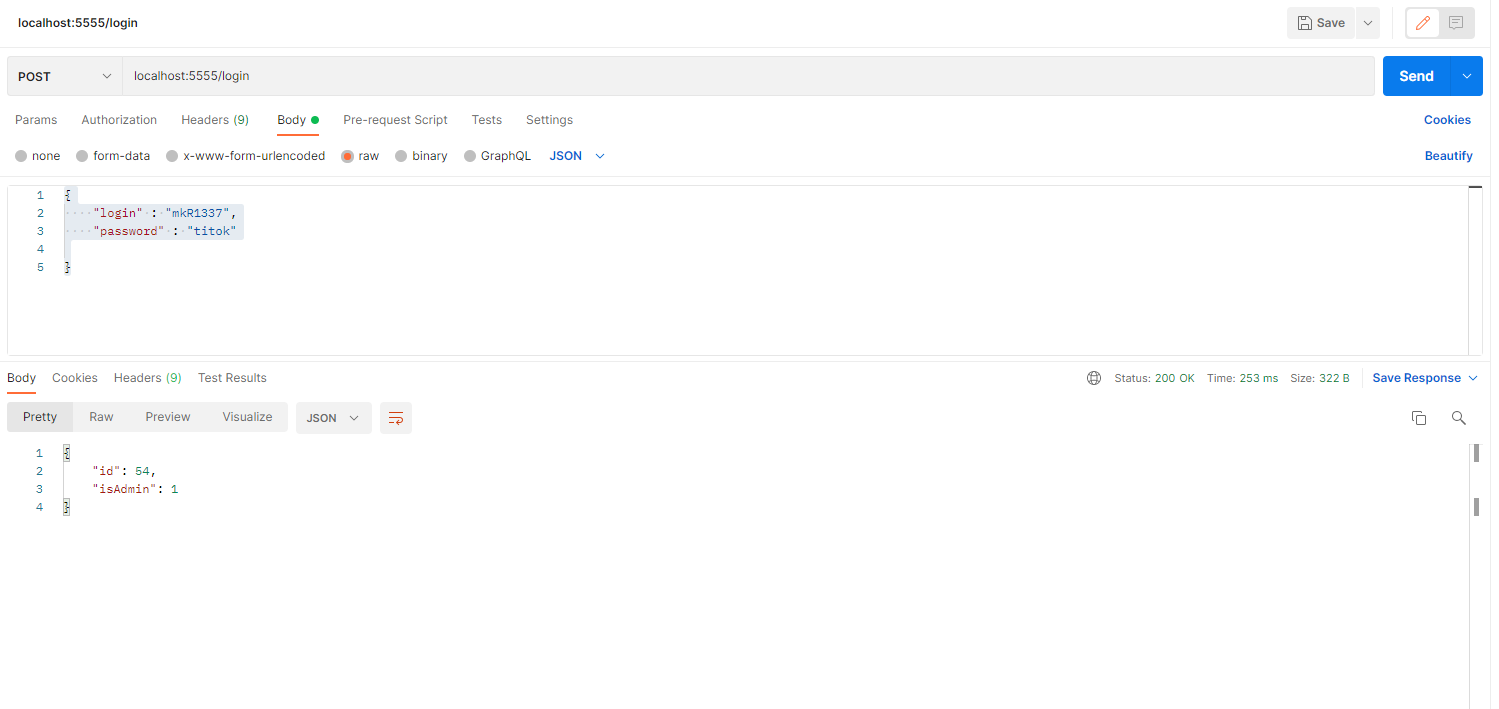
A fejlesztés alatt a rendszer tesztelése desktop eszközön több böngészőben (Google Chrome 100.0.4896.127, Mozilla Firefox 99.0.1) Windows 10 Enterprise operációs rendszer alatt történt. Továbbá mobil eszközön Android operációs rendszer alatt is teszteltük.

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

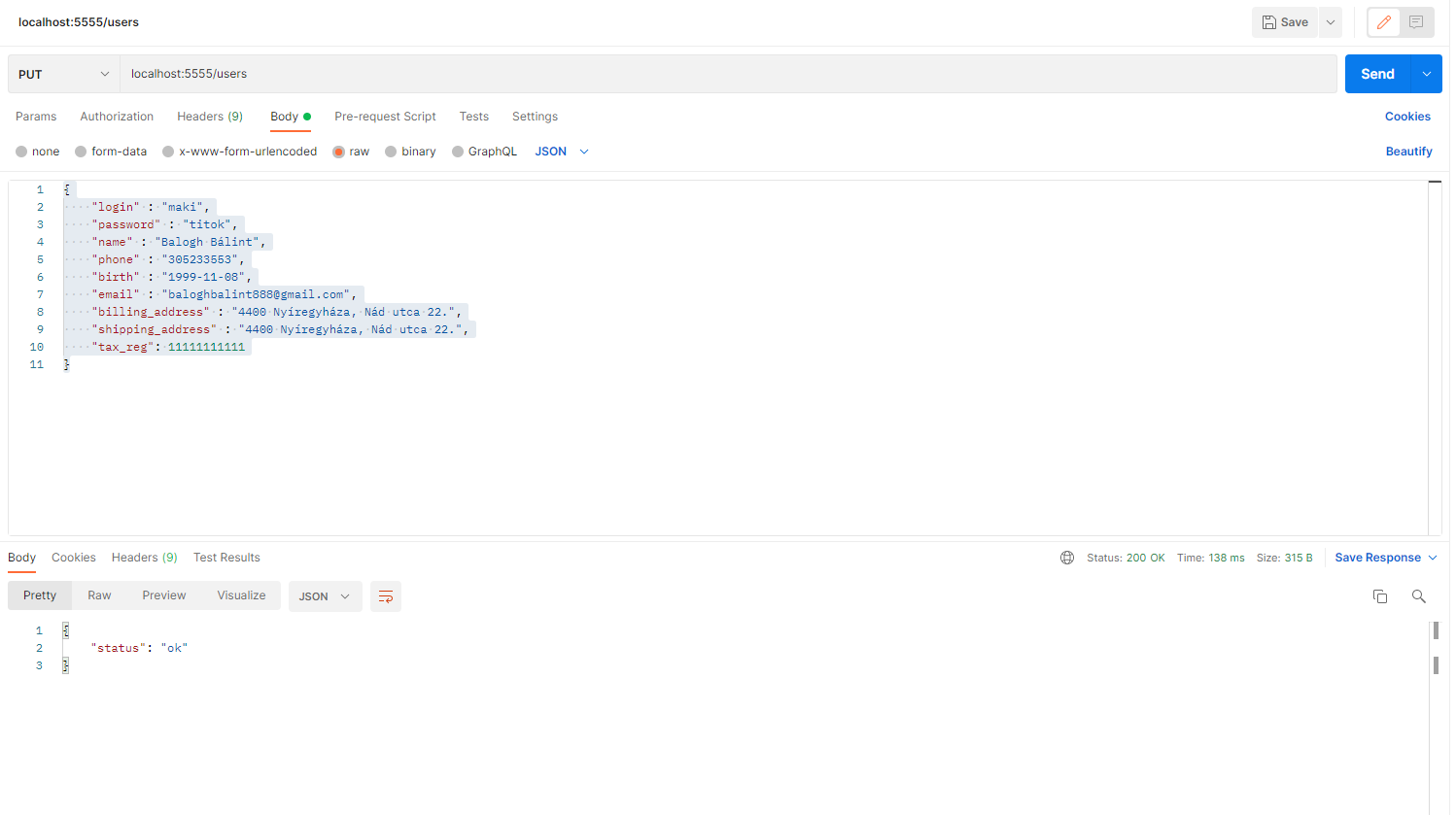
A feladat végrehajtása során folyamatosan jelen volt a tesztelés. Már a statikus oldal megtervezésekor és készülésekor tesztelve volt a megjelenés reszponzivitása, az olvashatóság, a láthatóság.

Az adatbázis tartalommal feltöltése után az oldal tartalmi része is tesztelve, ellenőrizve volt. Ezt Postman-ban valósítottuk meg, amely azonnali visszajelzést adott a fejlesztés sikerességéről, esetleges hibáiról. Ilyen volt például a felhasználók jelszavának tesztelése:

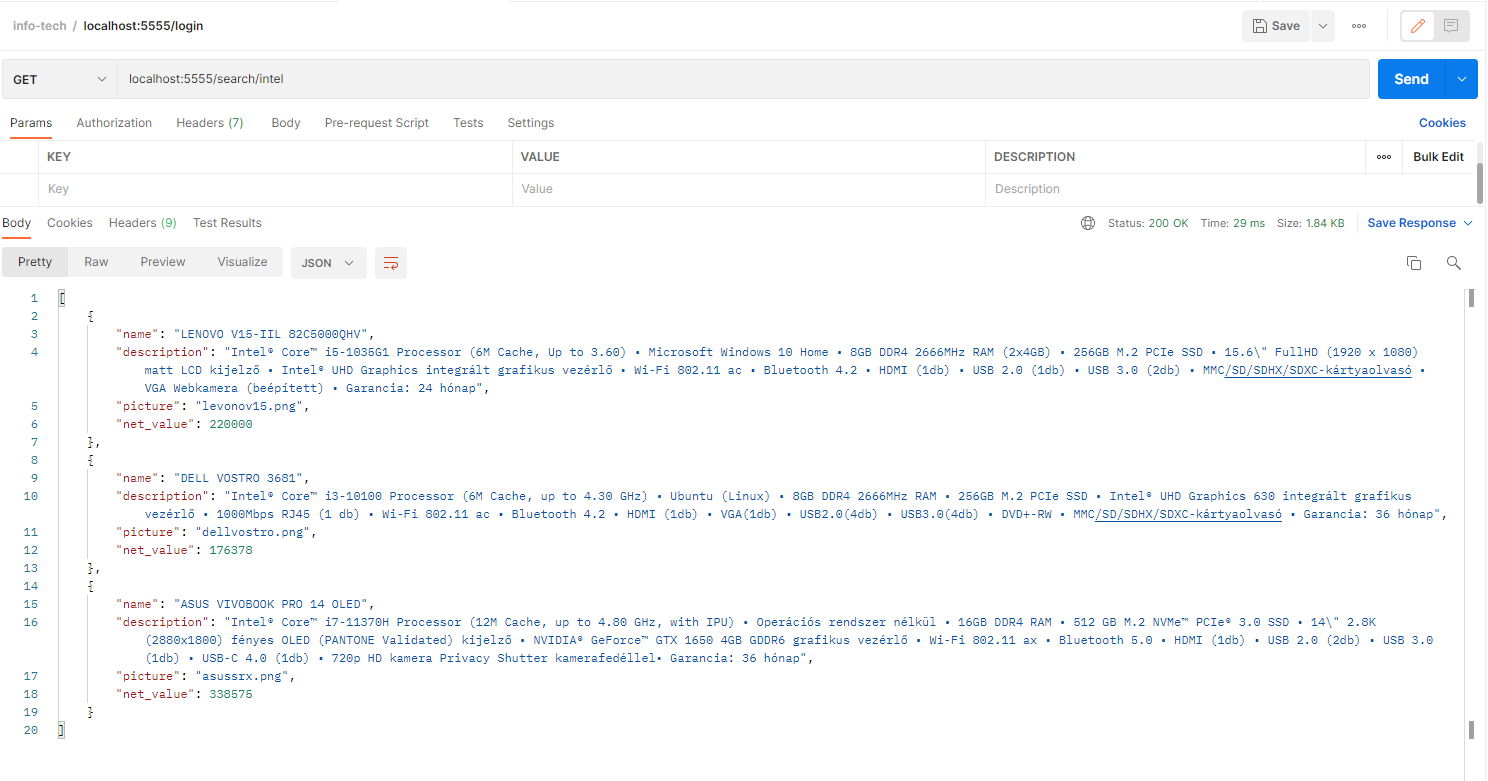
Új felhasználó felvétele esetén (PUT kérés) a jelszót lehasseljük. A fejlesztés során előre felvett felhasználónak például titok jelszót adtunk meg, melyet átalakítottunk. Login page esetén a frontend beküldi az adatot, a backend lehasseli, és a lehasselt jelszó adott pillanatban egyezik az adatbázisból kivett userhez között jelszóval, akkor a felhasználót átengedi.



Login authentikáció tesztelése Postman-ben



Regisztráció tesztelése Postman-ben



Kulcsszóra való keresés tesztelése Postman-ben

A fejlesztés során egy JEST teszt is készült, ahol az áfa értékének megadása lett tesztelve, hogy hibás érték ne jelenhessen meg.

const {vat} = require('./vatFormatter')

describe("Áfa százalék formázó tesztjei",() =>{

    test('0 értékre', ()=>{

        expect(vat(0)).toEqual(0)

        expect(vat(0)).not.toEqual(!0)

    })

    test('negatív értékre', ()=>{

        expect(vat(-12)).toEqual(0)

        expect(vat(-12)).not.toEqual(-1.12)

    })

    test('helyes értékre', ()=>{

        expect(vat(27)).toEqual(1.27)

        expect(vat(27)).not.toEqual(1)

    })

    test('100 vagy nagyobb értékre', ()=>{

        expect(vat(100)).toEqual(0)

        expect(vat(101)).toEqual(0)

        expect(vat(101)).not.toEqual(!0)

    })

})

# 11. Bevezetés, éles üzemmód

* Az elkészült munka megtekintéséhez szükséges a XAMPP letöltése. Látogassunk el a következő oldalra: <http://www.apachefriends.org/en/xampp.html>, és onnan töltsük le az operációs rendszerünknek megfelelő XAMPP-verziót.
* Telepítsük fel. A letöltött futtatható állományt indítsuk el, és kövessük a képernyőn megjelenő utasításokat. Windows alatt a javasolt telepítési könyvtár c:\xampp.
* töltse le a Windows telepítőt a Node.js webhelyről. Miután kiválasztotta az igényeinek megfelelő verziót, futtassa a telepítőt.
* Indítunk kettő Visual Studio Code alkalmazást, egyikben a Front mappát nyitjuk meg, a másikban a Back-end mappát kell megnyitni az Open Folder menüponttal.
* mindkét Visual Studio Code-ban indítunk egy Terminált a Terminal menu/New Terminal-lal. (Vagy a Command Line is megfelel)
* Sikeres telepítések után a xampp-ot indítsuk el és az Apache-t és MysQL-t is Startoljuk, és a MysQL Admin gombját válasszuk.
* A phpMyAdmin-ba importáljuk be a kapott adatbázist.
* Mindkét Visual Studio Code-ban a Terminal-ban npm –i paranccsal telepítsük az összes modult.
* Mindkét Visual Studio Code-ban npm start paranccsal indítsuk az alkalmazást.

# 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 néhány funkció a fejlesztési idő rövidsége miatt nem kerülhetett bele a szoftverbe. Ilyen hiányosságok például a chat szoba, emailes visszajelzés, az elfelejtett jelszó vagy a https biztonságos kapcsolat lehetőségei.

A biztonság fejlesztése mellett a SEO (Search Engine Optimalization) is nagy figyelmet élvez, ugyanis a fejlesztés során fő szempont volt a több platformos megjelenés, viszont szintén az idő rövidsége miatt nincs megfelelően kereső optimalizálva az oldal.

A teljes alkalmazás Git repository-ja:

<https://github.com/baloghbalint888/info_tech>

A kliens oldal Git repository-ja (history-val)

<https://github.com/rekalaca/info>

A szerver oldal Git repository-ja (history-val)

<https://github.com/baloghbalint888/backend>