Python 1. beadandó feladat

# ELTE IK PNYFT - 2025

# REST API készítése FastAPI alapokon

Az API egy webbolt felhasználóinak a kezelését teszi lehetővé. Tartalmazza a felhasználók bolthoz rendelését, a kosár hozzáadását egy adott vásárlóhoz, valamint a vásárló kosarába tudunk helyezni termékeket a végpontok segítségével. A behelyezett termékeket lehet módosítani, vagy törölni. A megvásárolt termékek összeértékét, vagyis a fizetendő összeget is le lehet kérdezni egy adott vásárló kosarában található termékek árai alapján. Az API lehetőséget biztosít egy adott vásárló adatainak a lekérdezésére, az összes vásárló megjelenítésére, vagy egy adott vásárlói kosár tartalmának a megjelenítésére.

Az API ezekkel a funkciókkal segítségünkre lehet egy webbolt elkészítéséhez, de a felhasználói felületet nem tartalmazza. Az adatokat egy data.json nevű fájlban tárolja.

A feladat, amit meg kell oldanunk, egy FastAPI implementációjának a kiegészítése. A kód egy része már meg van írva a mellékelt könyvtárakban. Az előre elkészített fájlokban instrukciók vannak az adott modul funkcióinak a megírásához és véglegesítéséhez.

## A mellékelt fájlok tartalma

A képen szöveg, képernyőkép, szoftver, Multimédiás szoftver látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

* A data.json a JSON az adatok tárolására szolgál. Ebben elhelyeztünk néhány teszt adatot, amelyeket fel lehet használni az alkalmazás kipróbálásához.
* A filehandler.py és a filereader.py a JSON adatok fájlból történő beolvasására és fájlba írására használható. A függvényeket ki kell egészíteni ahhoz, hogy képesek legyenek JSON adatokat írni és olvasni.
* A routes.py tartalmazza az alkalmazás végpontjait. Ezeket a függvényeket kell kiegészíteni a megfelelő funkciókkal.
* A schemas.py a JSON adatok kliens és szerver közötti mozgatásához, valamint a JSON adatok kezelésére alkalmas osztályokat tartalmazza. Ezeket az osztályokat ki kell egészíteni a megfelelő mezőkkel a JOSN adatok alapján. Ügyelni kell az adatok helyességére is a fájlban leírtak alapján.
* A main.py tartalmazza a FastAPI main modulját, amelyet futtatni kell az alkalmazás elindításához a uvicorn python csomag segítségével.

# Feladatok

Indítsuk el a REST API-t és nézzük meg az endpointok listáját a FastAPI általautomatikusan generált UI felületén. Az API elindításakor a rendszer generálni fog egy URL-t és egy portot rendel a futó programhoz. Ennek az URL-nek a kiegészítésével elérjük az alkalmazás /doc endpointjával a webes felületet.

***uvicorn mainfileneve:appneve --reload***

**(5 pont)**

Az alábbi képernyőt kell látnunk a webböngészőben:

A képen szöveg, képernyőkép, szám, Betűtípus látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

* Az itt látható végpontokat kell elkészítenünk.
* Az adduser segítségével egy felhasználót tudunk hozzáadni a webshophoz.   
  **(5 pont)**
* Az addshoppingbag egy kosarat rendel egy felhasználóhoz. **(5 pont)**
* Az additem egy terméket ad az adott felhasználó kosarához. **(5 pont)**
* Az updateitem módosítja az adott termék attribútumait egy felhasználó kosarában. **(10 pont)**
* A deleteitem töröl egy terméket a kosárból. **(5 pont)**
* A user egy adott felhasználó adatait adja vissza. **(5 pont)**
* A users visszaadja az összes felhasználót. **(5 pont)**
* A shoppingbag egy kosárban található összes termék adatait adja vissza.

**(5 pont)**

* A getusertotal egy vásárló kosarában található termékek értékét adja vissza.   
  **(10 pont)**
* A Route az üzlet nevét adja vissza. Ez a funkció már implementálva van a main.py fájlban

Az alkalmazás webes felületén (/docs) a fenti feladatokat el lehet végezni. Minden egyes funkció a router fájlban leírt módon viselkedik, és az ott meghatározott adatokat adja vissza Szabványos JSON formátumban, a JSONResponse függvénnyel, státus kóddal együtt.

A végpontok megírása során az alábbi szabályokat kell betartani **(20 pont)**:

* Minden végpontnál adjuk meg a response\_modell értékét (típus).
* Ügyeljünk a típusok megadására a függvényekben is.
* A függvények visszatérési értéke JSONResponse() legyen
* Minden függvény tartalmazzon hibakezelést, hiba esetén dobjon egy HTTPException-t és a megfelelő status code-ot.
* A függvények a JSON adatok mentéséhez és visszaolvasásához a filehandler.py és a filereader.py fájlt használják. Tegyük elérhetővé ezeket az alkalmazásban! Az adatokat a data.json fájlba kell menteni.
* A HTTP válaszok minden esetben tartalmazzák a megfelelő status code-ot, pl 404 - Not found, vagy 200 – OK.

# Az alkalmazás használata

A main.py fájl a REST API main modulja indítja el. A futtatásához telepíteni kell a pip csomagkezelővel a uvicorn és a fastapi csomagokat:

**pip install uvicorn, fastapi**

vagy

**pip3 install uvicorn, fastapi**

Ezután az alkalmazás a következő paranccsal futtatható a terminálban:

**uvicorn main:app --reload --port 9000**

A port nem kötelező opció.

A futó alkalmazás a következő URL-en érhető el:

**127.0.0.1:9000**

A webes UI pedig az alábbi URL használatával:

**127.0.0.1:9000/docs**

A dokumentáció pedig itt:

**127.0.0.1:9000/redoc**

Az adatok kezelésére használható osztályokat a megadott schema alapján ki kell dolgozni. A schema.py tartalmazza azok küldésére és fogadására készített osztályokat. Az osztályokban az adatok legyenek validálva! **(10 pont)**

* Az int adatok nem lehetnek negatívak.
* Az email mező csak e-mail formátumot fogadhat el.
* Hiba esetén ValueErrort kell dobni és lehetőség szerint a kliens oldalon is jelezni kell a hibát.

# A fájlkezelők implementálása

A data.json fájlban található adatok kezelésére két Python modult készítettünk. Az egyik, a filehandler.py, ami a fájlok írására használható függvényeket, a másik a filereader.py, ami a JSON adatok olvasására készült függvényeket tartalmaz. A függvények törzse nincs implementálva, ezért a beadandó elkészítéséhez ezeket a függvényeket is implementálni kell az alábbi leírás alapján **(10 pont)**:

## filehandler.py

Új felhasználó hozzáadása:

new\_user = {

"id": 4, # Egyedi felhasználó azonosító

"name": "Szilvás Szabolcs",

"email": "szabolcs@plumworld.com"

}

Felhasználó hozzáadása a JSON fájlhoz:

add\_user(new\_user)

Hozzáadunk egy új kosarat egy meglévő felhasználóhoz:

new\_basket = {

"id": 104, # Egyedi kosár azonosító

"user\_id": 2, # Az a felhasználó, akihez a kosár tartozik

"items": [] # Kezdetben üres kosár

}

add\_basket(new\_basket)

Új termék hozzáadása egy felhasználó kosarához:

user\_id = 2

new\_item = {

"item\_id": 205,

"name": "Szilva",

"brand": "Stanley",

"price": 7.99,

"quantity": 3

}

Termék hozzáadása a kosárhoz:

add\_item\_to\_basket(user\_id, new\_item)

Hogyan használjuk a fájlt? Importájuk a függvényeket a filehandler.py modulból:

from filehandler import (

add\_user,

add\_basket,

add\_item\_to\_basket,

)

# A JSON fájl elérési útja

JSON\_FILE\_PATH = ""

def load\_json() -> Dict[str, Any]:

with open(JSON\_FILE\_PATH, "r", encoding="utf-8") as file:

pass

def save\_json(data: Dict[str, Any]) -> None:

pass

def add\_user(user: Dict[str, Any]) -> None:

pass

def add\_basket(basket: Dict[str, Any]) -> None:

pass

def add\_item\_to\_basket(user\_id: int, item: Dict[str, Any]) -> None:

pass

## filereadr.py

Felhasználó adatainak lekérdezése:

user\_id = 1

user = get\_user\_by\_id(user\_id)

Felhasználó kosarának tartalmának lekérdezése:

user\_id = 1

basket = get\_basket\_by\_user\_id(user\_id)

Összes felhasználó lekérdezése:

users = get\_all\_users()

Felhasználó kosarában lévő termékek összárának lekérdezése:

user\_id = 1

total\_price = get\_total\_price\_of\_basket(user\_id)

Hogyan futtassuk a fájlt? Importájuk a függvényeket a filehandler.py modulból:

from filereader import (

get\_user\_by\_id,

get\_basket\_by\_user\_id,

get\_all\_users,

get\_total\_price\_of\_basket

)

# A JSON fájl elérési útja

JSON\_FILE\_PATH = ""

def load\_json() -> Dict[str, Any]:

pass

def get\_user\_by\_id(user\_id: int) -> Dict[str, Any]:

pass

def get\_basket\_by\_user\_id(user\_id: int) -> List[Dict[str, Any]]:

pass

def get\_all\_users() -> List[Dict[str, Any]]:

pass

def get\_total\_price\_of\_basket(user\_id: int) -> float:

pass

# A végpontok implementációja

Az alábbi végpontokat kell kidolgozni:

Felhasználó hozzáadása a bolthoz. A bemenet egy User típus, a visszatérési érték a felvitt felhasználó rekordja.

**@routers.post('/adduser', response\_model=User)**

**def adduser(user: User) -> User:**

**pass**

Kosár hozzáadása egy User-hez. A bemenő paraméter a felhasználó azonosítója, a visszatérési érték a következő szöveg: „Sikeres kosár hozzárendelés.”

**@routers.post('/addshoppingbag')**

**def addshoppingbag(userid: int) -> str:**

**pass**

Termék berakása a felhasználó kosarába. A bemenő paraméterek a felhasználó azonosítója és a termék. A visszatérési érték a kosár tartalma.

**@routers.post('/additem', response\_model=Basket)**

**def additem(userid: int, item: Item) -> Basket:**

**pass**

Egy adott termék attribútumainak módosítása. A bemenő paraméter a felhasználó azonosítója, valamint a termék azonosítója és új attribútumai. Termék cserével is megoldható a feladat. A visszatérési érték a kosár tartalma.

**@routers.put('/updateitem')**

**def updateitem(userid: id, itemid: int, updateItem: Item) -> Basket:**

**pass**

Egy termék törlése az adott felhasználó kosarából. A bemenő paraméterek a felhasználó azonosítója, valamint a termék azonosítója. A visszatérési érték a kosár tartalma.

**@routers.delete('/deleteitem')**

**def deleteitem(userid: int, itemid: int) -> Basket:**

**pass**

Egy adott felhasználó adatainak a megjelenítése. A bemenő paraméter a felhasználó azonosítója, a visszatérési érték a felhasználó rekordja.

**@routers.get('/user')**

**def user(userid: int) -> User:**

**pass**

Az összes felhasználó lekérdezése az adatbázisból. Nincs bemenő paraméter és a kosarak nem kerülnek megjelenítésre. A visszatérési érték a felhasználók listája.

**@routers.get('/users')**

**def users() -> list[User]:**

**pass**

Egy adott felhasználó kosarának megjelenítése. A paramétere a felhasználó azonosítója. A visszatérési érték a termékek.

**@routers.get('/shoppingbag')**

**def shoppinbag(userid: int) -> list[Item]:**

**pass**

Egy adott felhasználó kosarában lévő termékek értékét adja vissza. A bemenő paramétere a felhasználó azonosítója, a visszatérési érték az összeg.

**@routers.get('/getusertotal')**

**def getusertotal(userid: int) -> float:**

**pass**

**Kérdés esetén a gyakorlatvezetőhöz, vagy az előadás vezetőjéhez lehet fordulni.**

A feladat értékeléséhez minden végpontnak működnie kell. A pontszám a megvalósítás minőségétől is függ, vagyis, ha egy végpont működik, de a megadott feltételeknek nem felel meg, akkor kevesebb pontot ér.

Az elérhető maximális pontszám: **100 pont**

**Jó munkát!**