Szia!

Ebben a dokumentumban a 2022-es Nyári zeroToHero programozás verseny első fordulójának a feladatsorát olvashatod.

**Ne feledd, a feladatok megoldására egy hét áll a rendelkezésre.**

**Beküldési határidő: 2022.08.18. 23:59**

A feladatokat a következő lépések segítségével adhatod be:

### **Beadás módja**

1. Töltsd le ezt a Google Docs doksit a gépedre
2. Töltsd ki a Sárgán aláhúzott helyeket!
3. Miután kitöltötted a dokumentumot, mentsd le .pdf vagy .doc formátumban
4. Küldd el az így elkészült feladatlapot és a megoldott, zip fájlba csomagolt feladatot mint csatolmányt emailben a következő paraméterekkel:
   1. Tárgy: **[1.forduló] <Teljes neved>** (pl. “[1.forduló] Kovács András”)
   2. Cím: [**programozas@pcf.hu**](mailto:programozas@pcf.hu)
5. Ellenőrizd, hogy tényleg megfelelően elküldted-e a levelet és csatoltad-e a megoldott feladatot is :)

Sok sikert kívánunk a versenyhez!

Az első kitöltendő kérdésként kérlek add meg az neved és az email címed, melyről a megoldásokat küldöd:

**Név:**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**E-mail cím amiről a megoldást küldöd:**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# 

## 

## **Üdvözöllek a 2022-es zeroToHero programozás versenyének első fordulójában**

**Ez a forduló két megoldandó feladatot tartalmaz.**

Az ***első feladatrész*** négy kérdésből áll, melyekre pár mondatos, egybefüggő válaszokat várunk.

A ***második feladatrész*** egy több rész feladatból álló bemelegítő feladvány, melyben egy egyszerű alkalmazást kell fejleszteni és javítani.

A feladat a következő URL-ről tölthető le:  
[Keretrendszerek](https://drive.google.com/drive/folders/1BLmGLWenLSRXRjlh1evSs9NkfVTlTIGq?usp=sharing)  
[Étterem menü](https://drive.google.com/file/d/1802wIE2IhV5B75VUcyI_8o3P8gnCZGWE/view?usp=sharing)  
[Items](https://drive.google.com/file/d/1MxpZRcFLCgOYNsPPPlCdLZO0RrIGRHIr/view?usp=sharing)

**A beküldött projekt fordítható és futtatható kell hogy legyen!**

## 

## 

## **Feladat 1**

A kérdésekre adj bővebb, legalább 1-2 mondatos válaszokat:

1. **Mi az a CSV:** (1 pont)  
     
   \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
     
   **Mondj 2 lehetséges példát a használatára:** (1 pont)  
     
   \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
2. **Mit értünk REST végpont alatt:** (1 pont)  
     
   \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
     
   **Milyen HTTP metódust használunk általában, ha az adott végponton új, tárolandó adatot kérünk be:** (1 pont)  
     
   \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
3. **Mi az az in-memory cache, milyen megoldások létezhetnek arra, hogy idővel a cache tartalma eltérhet a háttértár tartalmától**:  
    (3 pont)  
     
   \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## 

## 

## **Feladat 2 - Alkalmazás továbbfejlesztése**

A következő feladatban a Los Pollos Hermanos csak csirkével és semmi mással nem foglalkozó gyorsétteremlánc kasszázó rendszerét kell továbbfejlesztened. Egy valós idejű belső jegyző rendszert kell csinálnod, ami képes a bemeneti adatokat a feladatleírásoknak megfelelően kezelni. A lementendő adatokat bármilyen formában mentheted, amíg azok a program újraindítása után sem vesznek el. A mellékelt keretrendszerek által használt dependenciákon kívül ne vegyél fel mást. A menu.csv felépítése a következő: egy sor egy menü adatait tartalmazza, pontosvesszőkkel elválasztva az elemeket. Először a menü ára, utána a menü neve, majd a sor végéig a menü elemei. Nem minden menü áll ugyanannyi elemből, de mindegyikben van legalább kettő! **Az items.csv felépítése egyszerűbb, egy sorban két rekord szerepel. Először az adott étel neve, majd az ára.**

1. Hozz létre egy POST végpontot /cashier/ url végződéssel, melyen keresztül egy json kérést tud fogadni az alkalmazás, melynek szerkezetét a gyakorlati feladatok után láthatod. A kérésben egy kasszás regisztrációs adatai fognak érkezni. Ezeket egy egyedi, generált azonosítóval le kell tárolni. Későbbi feladatok megoldásához a kasszáshoz tartozó adatokat bővítheted.   
    (2 pont)
2. Hozz létre egy POST végpontot /sale/ url végződéssel, melyen keresztül egy json kérést tud fogadni az alkalmazás, melynek szerkezetét a gyakorlati feladatok után láthatod. A kérésben a megrendelés elemei érkeznek be listásan. Külön tagek alatt a rendelés ára és a kasszás azonosítója. A feladat a rendelést egy egyedi azonosítóval, az árral, a kasszás azonosítójával és a rendelés elemeivel lementeni. A kasszák limitációja hogy a menüket is csak elemenként tudják felküldeni, viszont statisztikákhoz fontos hogy mikor volt menüs eladás és mikor nem. A letölthető menu.csv fájlban van pár példa lehetséges menü elemekre, amikor olyan ételek érkeznek be a kérésben amelyekből össze lehet kombinálni egy ebben a csv fájlban megtalálható menüt akkor ezt a rendelést egy olyan menüként mentsd le amivel megtaláltad az elemeket a csv-ben. Amennyiben több menübe is beleillik valamilyen kombináció akkor első körben a nagyobb elemszámú menüt próbáld kirakni a rendelésből, ha egyenlő elemszámúak akkor az olcsóbbat. Egy elem csak egy menübe tartozhat. (4 pont)
3. Előfordul hogy a kasszás figyelmetlen és annak ellenére hogy valaki menüt rendel nem a menü árát üti be a kasszába. Amennyiben a rendelésben érkező összeg nagyobb mint a menü ára (vagy a kirakható menü-k árai + a maradék) akkor az adott eladást meg kell jelölni egy ‘REFUND’ státusszal, egyébként egy ‘PROCESSED’ státusszal. (3 pont)
4. Hozz létre egy GET végpontot /sale/{id} végződéssel, ezen a végponton legyen lehetőség az eladások kiolvasására id alapján.  
    (2 pont)
5. Hozz létre egy GET végpontot /cashier/{id} végződéssel, ezen a végponton legyen lehetőség az adott kasszáshoz tartozó összes rendelés kiolvasására, listásan.  
    (2 pont)
6. Hozz létre egy PUT végpontot /sale/{id} végződéssel. A végponton legyen lehetőség utólag visszajárót adni egy hibás rendelésért. Ilyenkor a megrendelés ára álljon át a menük által meghatározott árakra. A státusza állítódjon ‘REFUND’-ról ‘REFUNDED’-re. Viszont a histórikus adatok megőrzése érdekében egy ‘difference’ adattagba vegyük fel hogy mekkora különbözetet kellett visszaadni az ügyfélnek.  
    (3 pont)
7. Hozz létre egy GET végpontot /cashier/ végződéssel. Ezen a végponton nem érkezik bemeneti paraméter. Minimum 1, maximum 2 kasszás azonosítóval válaszol. Először számolja ki, hogy ki az a kasszás akinek jelenleg a legmagasabb túlfizetései vannak (‘REFUND’ státuszú rendeléseket kell csak figyelembe venni). Illetve ki az a kasszás aki a már visszaadott adatokkal együtt a legmagasabb túlfizetést okozta (‘REFUND’ + ‘REFUNDED’). (4 pont)

PostCashierRequest példa:

{

"firstName":"Trevor",

“lastName”:”Sterling”

}

PostSaleRequest példa:

{

“soldItems”: [“double chickenburger”, “large coke”, “medium fries”],

“cashierId”: “b5bb47f3-103e-4107-b5aa-35e9f7f6f394”

“price”: 25.99

}

## Pontozás

| Feladat 1: | 7 pont |
| --- | --- |
| Feladat 2: | 20 pont |
| **Összesen:** | **27 pont** |