A Csoport

Információk:

1. Stringeknél pontos egyezést kérek

2. Ha a pontodat megszeretnéd nézni dolgozat írása közben, akkor fontos, hogy a C:\ meghajtóra dolgozz, mert csak ott engedi a tesztelést várható időn belül.

3. Leadáshoz töltsd fel a kiosztott vizsgafelhasználód meghajtójára a teljes projektet a node_modules mappa ***NÉLKÜL.***
A meghajtón hozz létre egy mappát a következő néven:
backend_controllerdoga_<TELJES_NÉV>.
Példa:
backend_controllerdoga_Kássa_Gergő

1. Feladat

- Készítsd el az ***app.js*** és ***server.js*** fileokat a gyökérkönyvtárban.
- Az alkalmazást futtasd tetszőleges porton
- Hozd létre a megfelelő mappaszerkezetet a tanult ***architektúra*** alapján
- A root folderben található ***errorHandler.js*** file-t használd fel a tanult módon.
  FONTOS! Figyelj az útvonalak sorrendjére!

2. Feladat

- Készíts egy ***routert***, ami a ***/watches*** útvonalra fog hallgatni.
  FONTOS, hogy a fálj nevét a tanult módon nevezd el.
- Ehhez a routerhez készíts egy ***controllert*** is (tanult elnevezéssel)

3. Feladat (GET /watches)

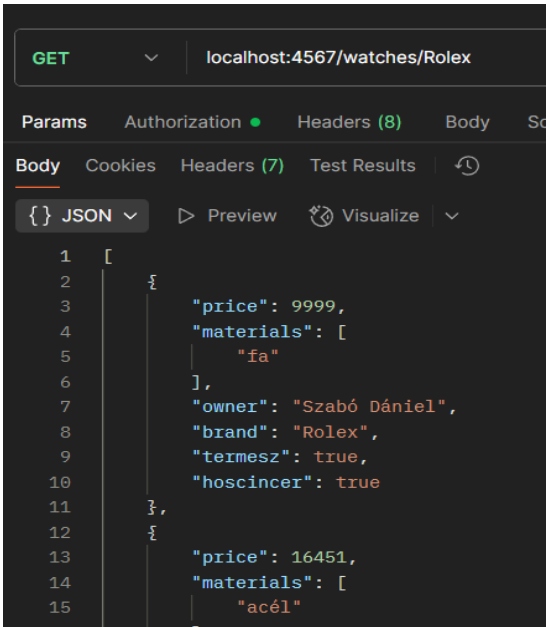
- Készíts egy ***GET*** metódust megvalósító függvényt a controlleredben.
  A metódus neve legyen: ***getWatches***
  Ebben a metódusban térj vissza a statikus tömb adataival.
  Az adatokat a ***data.txt-ben*** találod (másolható formában)
- Válaszolj a statikus tömböd teljes adathalmazával a ***/watches*** endpointon
- A válaszod státuszkódja legyen ***200.***
- Fontos, hogy ***JSON*** formátumban válaszolj az adatokkal.

- Példa a működésre (nem teljes):


```
[
    {
        "price": 9999,
        "materials": [
            "fa"
        ],
        "owner": "Szabó Dániel",
        "brand": "Rolex",
        "termesz": true,
        "hoscincer": true
    },
    {
        "price": 16451,
        "materials": [
            "acél"
        ],
        "owner": "Kássa Gergő",
        "brand": "Rolex"
    },
```


4. Feladat (GET /watches/<parameter>)

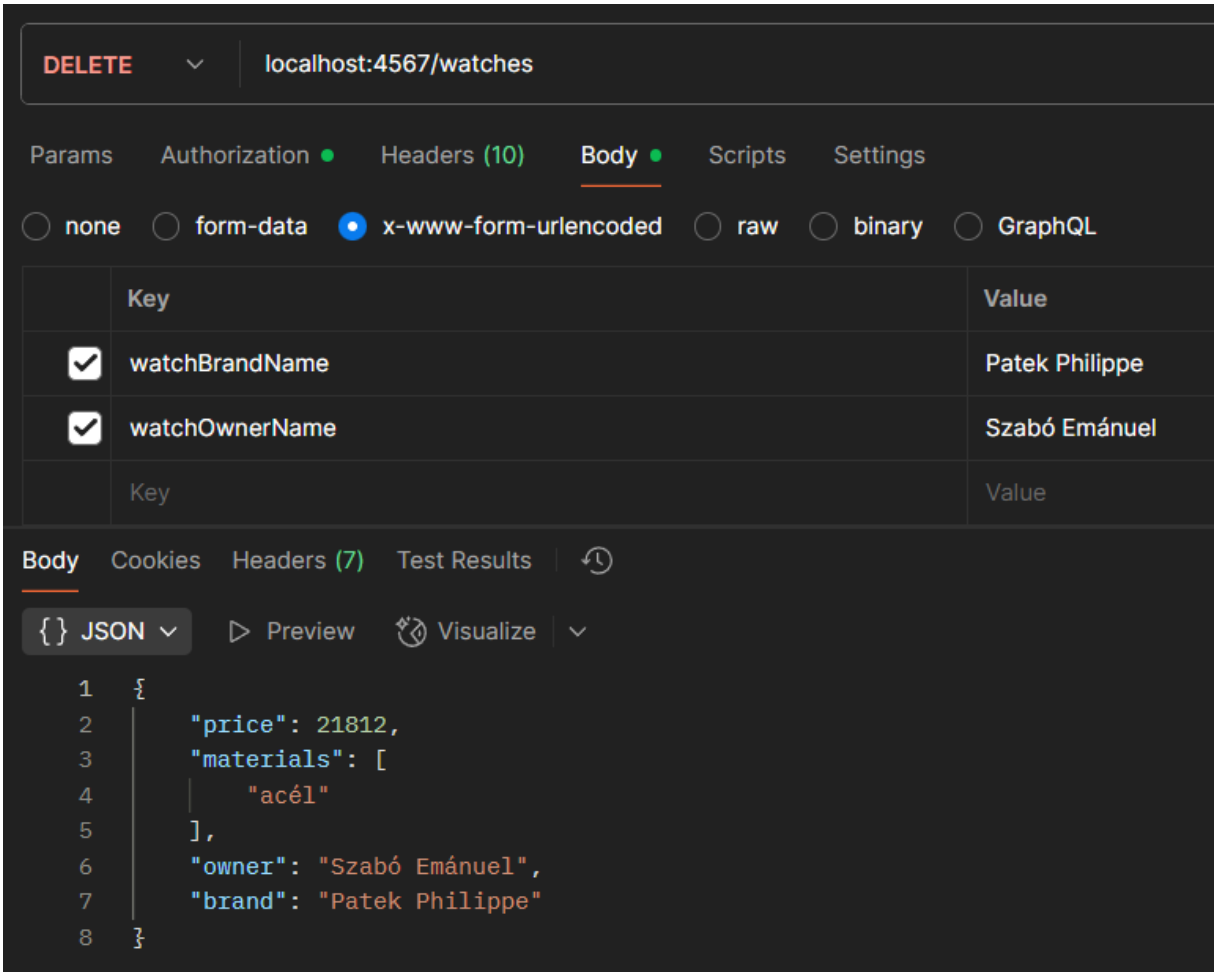
- Készíts egy paraméteres ***GET*** metódust, amiben válaszolni fogunk a ***kérés paramétereiben*** található márkájú órákkal.
  A metódus neve legyen: ***getWatchesByBrand***
- Válaszolj a statikus tömböd részleges adataival a ***/watches/<MEGADOTT_ÉRTÉK>***
- Használj paraméteres middleware-t a feladat megvalósításához
- A válasz státuszkódja legyen ***200.***
- Példa a működésre:



5. Feladat (DELETE /watches)

- Készíts egy ***DELETE*** metódust megvalósító metódust a controlleredben, amiben töröljük a kérés ***törzsében*** megadott adatok alapján az órákat.
  A metódus neve legyen: ***deleteWatch.***
- Válaszolj a törölt objektum adataival ***200-as*** státuszkóddal a ***/watches*** endpointon.
- A kérés törzsében használd a következő változóneveket:
  - ***watchOwnerName:*** óra tulajdonosának neve
  - ***watchBrandName:*** óra márkája
- Ha bármelyik kulcs hiányozna a kérésből, dobjunk hibát ***400-as*** hibakóddal a következő szöveggel:
  A kereséshez mindkét adat szükséges!
- Ha több órát találunk a megadott adatokkal (watchOwnerName és watchBrandName), akkor töröljük az utolsó előfordulását az objektumnak.
- Ha nem találnánk órát a megadott paraméterekkel, akkor dobjunk ***404-es*** hibakóddal (státuszkód) hibát a következő üzenettel:
  Nem található óra ilyen paraméterekkel!
- [PRO TIP]: Használd a következő függvényeket:

- Példa a működésre, ahol a tulajnak több ugyan olyan márkájú órája van:



Pontod / Jegyed megjelenítéséhez

1. Futtasd a ***start.bat*** scriptet
2. Megnyílik console-on a pontjaid feladatonkénti bontása (ezt az ablakot ne zárd be)
3. Létrejön egy pont.txt file a root folderedben, ami tartalmazza a pontodat, a jegyedet, összesen megszerezhető pontokat és a matekot, amivel számolva van a jegyed a jelenlegi alkalmazásod állapota alapján.

Tippek

Nem kell újrafuttatnod semmit, *.js kiterjesztésű fileok mentése esetén újraindul a test script.

Célszerű feladatonként checkolni a pont.txt file-t.

Lock in