Szoftverfejlesztő és -tesztelő próba vizsgafeladatsor

2025.

1. Grafikus és konzolos részt egyaránt tartalmazó asztali alkalmazás fejlesztése

A következő feladatban egy szerepjáték karaktereivel kapcsolatos szöveges állomány áll a rendelkezésére, melyekkel programozási feladatokat kell megoldania. A feladat megoldása során vegye figyelembe a következőket:

- A képernyőre írást igénylő részfeladatok eredményének megjelenítése előtt írja a képernyőre a feladat sorszámát (*például: 4. feladat:*)!
- Az egyes feladatokban a kiírásokat a minta szerint készítse el!
- Az ékezetmentes azonosítók és kiírások is elfogadottak. Az azonosítókat kis- és nagybetűkkel is kezdheti. Az elnevezéskor tartsa szem előtt a tiszta kód elveit!
- A program megírásakor az állományban lévő adatok helyes szerkezetét nem kell ellenőriznie, feltételezheti, hogy a rendelkezésre álló adatok a leírtaknak megfelelnek.
- A megoldását úgy készítse el, hogy azonos szerkezetű, de tetszőleges bemeneti adatok mellett is helyes eredményt adjon!

Az adatforrás a `karakterek.csv` elnevezésű UTF-8 kódolású szöveges állomány, melynek első sora a mezőneveket tartalmazza. Az adatforrásban a következő adatokat találja meg:

- **karakter neve:** *szöveges adat*
- **faj megnevezése:** *szöveges adat*
- **rövid leírás:** *szöveges adat (maximum 40 karakter)*
- **szint:** *egész szám (35 – 65 között)*

A fájl soraiban található adatokat pontosvesszők választják el egymástól.

1. Készítsen konzolos alkalmazást a következő feladatok megoldására, melynek projektjét *SzerepjatekKarakterek* néven mentse el!
2. Készítsen saját osztályt `Karakter` azonosítóval, amely képes az adatforrás sorainak adatait tárolni (*karakter neve, faj megnevezése, leírás, szint*)!
3. Olvassa be az adatforrás adatait és tárolja az adatokat a `Karakter` osztály segítségével egy olyan adatszerkezetben, amely használatával a további feladatok megoldhatók!
4. Készítsen valós értékkel visszatérő statikus metódust, amely megadja az *elfek* szintjeinek átlagát! A metódus a keresett faj nevét és a szerepjáték karaktereket tartalmazó adatszerkezetet paraméterként fogadja! A kapott eredményt három tizedesjegy pontossággal jelenítse meg a kimeneten!
5. Határozza meg és írja ki a minta szerint, hogy melyik fajból található a legtöbb a fájlban!
6. Jelenítse meg azon karakterek neveit, akiknek a leírásában szerepel az „*okos*” tulajdonság!

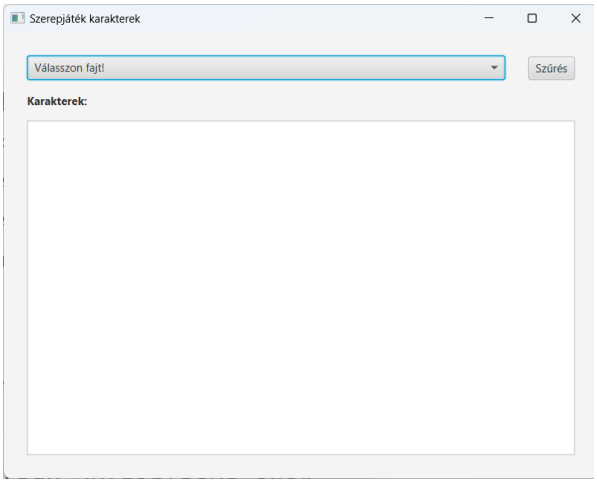
A program kimenete

```
4. feladat:
Az elfek szintátlaga: 57,400
5. feladat:
A leggyakoribb faj: ember
6. feladat:
Okos karakterek: Zara Thrain Urzog
```

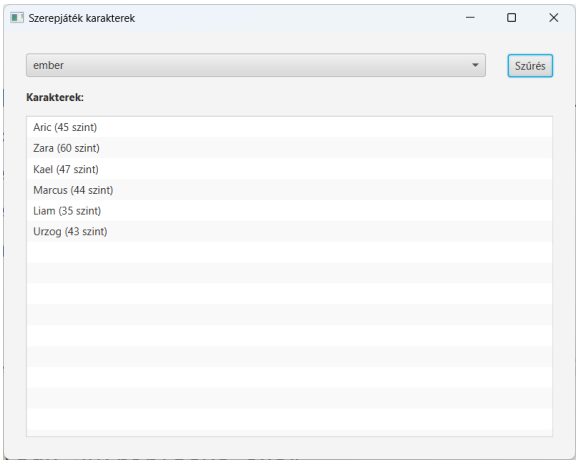
A grafikus alkalmazás elkészítése előtt hozzon létre a lokális adatbáziskezelő rendszerében (XAMPP) *szerepjatekkarakterek* néven egy UTF-8 kódolású, magyar egybevetésű adatbázist! Ezt követően futtassa le vagy importálja be a `szerepjatekarakterek.sql` szkriptet az adatbázisba!

7. Készítsen grafikus alkalmazást a következő feladatok megoldására, melynek projektjét *SzerepjatekKarakterekGUI* néven mentse el! A megjelenő ablak reszponzív viselkedésű legyen! Az ablakban levő vezérlők típusai feleljenek meg a mintának!
8. Csatlakozzon a *szerepjatekkarakterek* adatbázishoz, majd töltse be a lenyíló listába a különböző fajokat! A lista első eleme a „*Válasszon fajt!*” opció legyen!
9. A „*Szűrés*” gomb megnyomásakor szűrje le az adatbázisból azokat a szerepjáték karaktereket nevükkel és szintjükkel együtt, akik megfelelnek a kiválasztott fajnak! Az eredményt töltse be az ablak további részét kitöltő vezérlőbe a minta szerint!
10. Ügyeljen arra, hogy ha a „*Válasszon fajt!*” opció van kijelölve, akkor a program jelenítsen meg hibaüzenetet a mintának megfelelően!

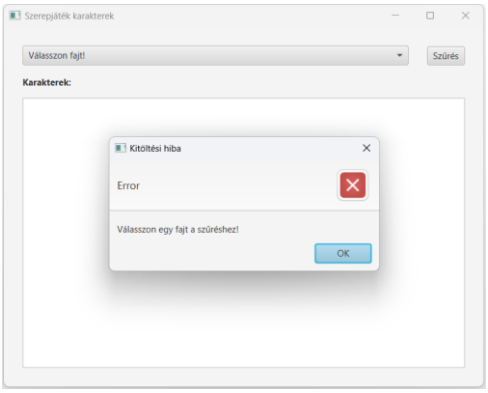
A program grafikus kimenetei



Indulóképernyő



Szűrés



Hibajelzés

Másolja be az `Ork.java` fájlt a projektjébe és készítse el az alábbi tesztet **JUnit5 keretrendszerben**!

11. Tesztelje le az `okos()` metódus helyes működését!

2. Frontend programozás

A következő feladatban egy frontend alkalmazást készítését kell elvégeznie a kiadott leírás szerint.

Az alábbiakban a *https://reeldev.hu/api/probavizsga/karakterek* URL-en elérhető webszerver REST API függvényéhez kap segítséget:

- **URL:** *https://reeldev.hu/api/probavizsga/karakterek*
  Metódus: *GET*
  Válasz: *Az adatbázisban lévő karakterek adatai (név, faj, leírás, szint).*
- **Válasz minta:**

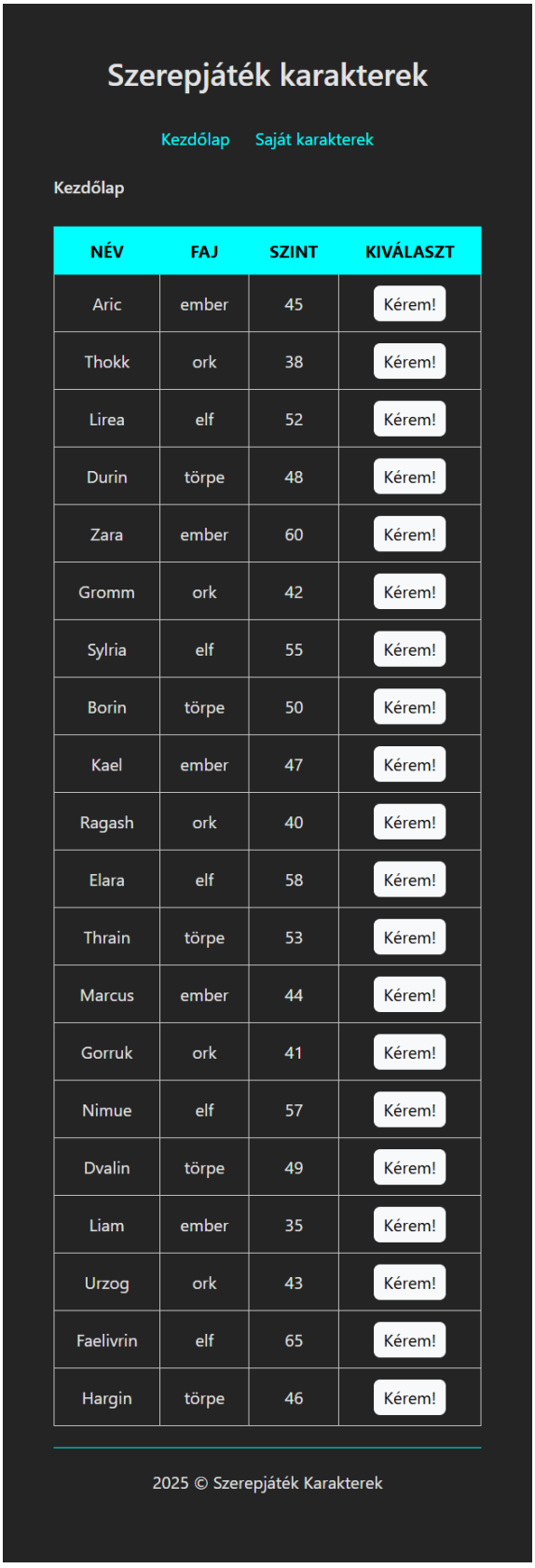
```
[
  {
    "név": "Aric",
    "faj": "ember",
    "leírás": "Kardmester, tűzvarázs, becsületes",
    "szint": 45
  },
  ...
]
```

Csomagolja ki és nyissa meg a *SzerepjatekKarakterek.zip* állományban található mappát, amely egy frontend projekt vázát és a megoldáshoz szükséges csomagok telepítéseit tartalmazza! A projektben lévő fájlokat használja fel, szabadon szerkesztheti és bővítheti őket!

Az alábbi feladatokat **Vue.js** JavaScript keretrendszer felhasználásával oldja meg! Az adattároláshoz ***Pinia tárolót*** alkalmazzon!

1. Állítsa be az alkalmazásban a navigációt! Hozzon létre két menüpontot a `menusav` komponensben „*Kezdőlap*” és „*Saját karakterek*” néven! Ha az első menüre kattintunk, akkor oldalfrissítés nélkül töltődjön be a `kezdolap` komponens, ha a másodikra, akkor pedig a `karaktereim` komponens a navigációs sáv alá!
2. Hozzon létre a `kezdolap` komponensben egy `karkterekLekerese` függvényt, amely betöltésekor kérdezze le a fenti API-ban tárolt adatokat! Az adatokból képezzen `Karakter` típusú objektumokat, amihez használja a *classes* mappában található *Karakter* osztályt! Az objektumokat tárolja el egy erre alkalmas adatszerkezetben!
3. Hozzon létre egy `karakterLista` komponenst, amely a webalkalmazás nyitó oldalán jelenjen meg! A komponens a paraméterként átadott objektumtömb adatait jelenítse meg táblázatos formában a minta szerint!
4. A táblázat utolsó oszlopában helyezzen el egy „*Kérem!*” feliratú gombot, ami az adott szerepjáték karaktert a felhasználó saját karakterei közé helyezi és eltávolítja a táblázatból (ha ismételten a „*Kezdőlap*” menüpontra kattintunk, akkor a saját karakterek ne jelenjenek meg a táblázatban)!
5. Ha a „*Saját karakterek*” menüpontra kattintunk, akkor jelenítse meg az eddig hozzáadott karaktereket a minta szerint (a táblázat utolsó oszlopa itt ne jelenjen meg)! A megoldáshoz itt is alkalmazza a már korábban elkészített `karakterLista` komponenst!
6. A projektben talál egy `szerepjatek` komponenst.
   a. Tesztelje le, hogy ha nincs kiválasztott faj, akkor nem jelenik meg semmi a bekezdésben!
   b. Tesztelje le, hogy ha lenyíló listából kiválasztunk egy fajt, akkor az űrlap alatti bekezdésben valóban a kiválasztott faj megnevezése szerepel!

Az alkalmazás kinézete



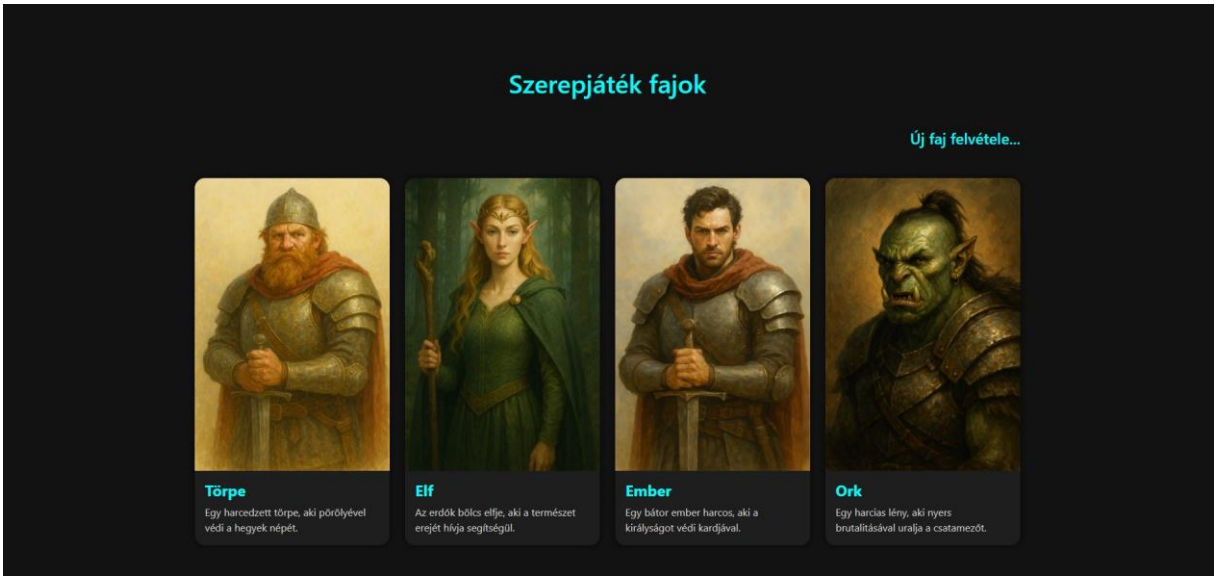
3. Reszponzív weboldal készítése

Ebben a feladatban egy szerepjáték weboldalát kell módosítania a feladatleírás és a minta szerint. Ahol a feladat másként nem kéri, a formázási beállításokat a style.css stílusállományban végezze el úgy, hogy az új szelektorokat az állomány végén helyezze el!

Nagyobb felbontású, színes mintákat a kész weboldalról a minta01.jpg, minta02.jpg és minta03.jpg állományokban talál, melyet a megoldásban nem használhat fel!

Nyissa meg a species.html, new.html és a style.css állományokat és szerkessze azok tartalmát az alábbiak szerint:

1. A style.css állományban megfelelő szelektor alkalmazásával állítsa be, hogy a Szerepjáték fajok oldalon a kártyák címének (pl. Ork) szövegszíne ugyanolyan legyen, mint az oldal címének színe!
2. Módosítsa a card-img-top Bootstrap kijelölőt úgy, hogy a kép két felső sarkát 1rem mértékben lekerekítse.
3. Állítsa be a transform tulajdonságot scale(1.05) értékre, ami akkor aktiválódik, ha a kártyák fölé visszük az egeret.
4. Biztosítsa a 4 kártya mintának megfelelő reszponzív megjelenését!
5. Hozza létre a minta szerinti hivatkozást, ami az új fajok felvételére szolgáló űrlap oldalára vezet. A hivatkozás ne legyen aláhúzva, ugyanabban az ablakban nyíljon meg és minden állapotában aqua színű legyen.
6. Az új faj felvételére szolgáló űrlapot egészítse ki 2 radio gombbal a minta szerint, és ügyeljen arra, hogy stílusa és szerkezete megegyezzen a többi elemével! Alapértelmezetten a Humanoid gomb legyen kijelölve.
7. A new.html oldal végén a javascript kód feladata, hogy egy számlálóban megjelenítse, hogy hány képesség lett kiválasztva a fajhoz. A script futásakor a konzolon az alábbi hiba jelenik meg: Uncaught TypeError: Cannot set properties of null (setting 'textContent') at updateAbilityCount. Javítsa a hibát!
8. Validálja hibátlanra a css és html oldalakat!



4. Backend feladat

Készítsen egy REST API-t, amely egy fantasy szerepjáték világának karaktereit és azok fajait kezeli! Az alábbi feladatsorban részletezve találja a megvalósítandó pontokat.

1. Hozzon létre backend szerver projektet az Ön által választott programnyelven, illetve fejlesztési környezetben! A projektmappát „Vezetéknév_Keresztnév_backend” formában nevezze el!
2. Hozza létre a szerepjatek adatbázist, abban a `fajok` nevű táblát a következő mezőkkel:
   - `id` INT, AUTO_INCREMENT, PRIMARY KEY
   - `nev` VARCHAR(255)
   - `leiras` TEXT
3. Töltse fel a `fajok` táblát a következő adatokkal:

id	nev	leiras
1	Ember	Sokoldalú faj
2	Elf	Gyors és bölcs
3	Ork	Erős, de lassú
4	Törp	Kitartó harcos nép

4. Importálja az előre elkészített `karakterek` táblát a karakterek.sql fájlból. A `karakterek` tábla az alábbi mezőket tartalmazza: id, nev, szint, ero, ugyesseg, faj_id. A `faj_id` mező az imént létrehozott `fajok` tábla id mezőjére hivatkozik.
5. Valósítson meg egy végpontot, amely lekérdezi az adott fajhoz tartozó karaktereket név alapján.
   Az URL tartalmazza a `faj` paramétert (pl. /api/karakterek?faj=elf).
   A válasz tartalmazza a karakterek adatait, beleértve a faj nevét és azonosítóját is.
   Ha nincs találat, vagy a faj nem létezik, adjon vissza hibaüzenetet.

Metódus	URL	Body	Válasz
GET	/api/karakterek?faj=elf	üres	JSON

Hibás fajnév – 400 BAD REQUEST

```
{ "hiba": "nincs találat" }
```

Sikeres válasz – 200 OK

```
[
  {
    "id": 2,
    "nev": "Lindir",
    "szint": 5,
    "ero": 7,
    "ugyesseg": 10,
    "faj": {
      "id": 2,
      "nev": "Elf"
    }
  }
]
```

6. Készítsen végpontot új karakter rögzítésére JSON formátumban.
   A kérés JSON formátumban tartalmazza a karakter nevét, szintjét, erejét, ügyességét és fajának azonosítóját.
   Sikeres felvétel esetén adja vissza az új karakter azonosítóját.
   Hibás kérés esetén (pl. hiányzó mező, nem létező faj_id) adjon hibaüzenetet.

Metódus	URL	Body	Válasz
POST	/api/karakter	JSON	JSON/szöveges

Példa kérés (JSON):

```
{
  "nev": "Thorg",
  "szint": 3,
  "ero": 10,
  "ugyesseg": 5,
  "faj_id": 3
}
```

Sikeres válasz – 201 CREATED

```
{ "id": 6 }
```

Hibás kérés – 400 BAD REQUEST

```
{ "hiba": "Hiányzó adat vagy érvénytelen faj" }
```

7. Hozzon létre végpontot az adatbázisban szerepelő karakter törlésére az alábbi beállításokkal! Az id-ben a törölni kívánt karakter azonosítója szerepeljen! Sikeres törlés esetén 204 NO CONTENT státuszkóddal térjen vissza! Amennyiben a megadott azonosító nem létezik 404 NOT FOUND státuszkóddal és az „Karakter nem létezik” üzenettel térjen vissza!

Metódus	URL	Body	Válasz
DELETE	/api/karakter	JSON	üres/szöveges

Példa kérés (JSON):

```
{ "id": 5 }
```

8. Hozzon létre Postman kollekciót backend_teszt néven, melyben az Ön által létrehozott összes útvonalat ellenőrzi! Exportálja a kollekciót a projekt gyökér könyvtárába Backend_teszt néven!
9. A forráskódját tömörítse be Vezetéknév_Keresztnév_backend.zip néven!