1. Asztali alkalmazás fejlesztés konzolos része

Autók – konzolos feladat

Az autok.csv nevű állomány tartalmazza azokat az autókat, amelyeket egy adott évben eladtak. A vizsga során ezt a fájlt feldolgozva kell feladatokat megoldania. Megoldása során vegye figyelembe a következőket:

- A képernyőre írást igénylő részfeladatok eredményének megjelenítése előtt írja a képernyőre a feladat sorszámát (például: 5. feladat)!
- Az egyes feladatokban a kiírásokat a minta szerint készítse el!
- Az ékezetmentes azonosítók és kiírások is elfogadottak.
- A feladatok megoldása során törekedjen a tiszta kód alapelveinek a betartására.
- 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 az azonos szerkezetű, de tetszőleges bemeneti adatok mellett is helyes eredményt adjon!

A fájl UTF-8 kódolású és minden sorában az eladott autók adatai szerepelnek. Az állomány első sora a mezőneveket tartalmazza, az adatokat pontosvessző választja el egymástól a mintának megfelelően:

autok.csv részlet

```
Sorszám;Márka;Modell;Gyártási év;Szín;Eladott darabszám;Átlagos eladási ár;
1;Toyota;Corolla;2019;Fehér;500;20000;
2;Ford;Fiesta;2018;Kék;600;18000;
3;Volkswagen;Golf;2020;Piros;800;22000;
4;Honda;Civic;2017;Fekete;450;19000;
5;Hyundai;i30;2021;Zöld;700;21000;
...
```

- Sorszám: Az autó sorszáma az eladott darabszám alapján
- Márka: Az autó márkája
- Modell: Az autó modellje
- Gyártási év: Az autó gyártási éve
- Szín: Az autó színe
- Eladott darabszám: Hány darabot adtak el az adott modellből
- Átlagos eladási ár: Az autó átlagos eladási ára euróban

Megoldását úgy készítse el, hogy az azonos szerkezetű, de tetszőleges bemeneti adatok esetén is helyes eredményt adjon! A képernyőre írja ki a feladat sorszámát a mintának megfelelően az eredmény megjelenítése előtt! A feladatokban a kiírásokat a minta szerint készítse el!

Feladatok

1. Készítsen konzolos alkalmazást a következő feladatok megoldására, melynek projektjét autoapp néven mentse el! (1 pont)
2. Hozzon létre saját osztályt Auto azonosítóval, mely adattagjaiban egy-egy autó adatait tudja majd tárolni! Az adattagok létrehozása előtt olvassa el a konzolos rész feladatait! (1 pont)
3. Az Auto osztály konstruktora kapja meg paraméterként a forrásállomány egy sorát, és inicializálja az osztály adattagjait! (1 pont)
4. Készítsen statikus metódust az Auto osztályhoz, ami beolvassa a paraméterében megadott nevű állomány sorait, és tárolja az adatokat egy Auto osztályon alapuló összetett adatszerkezetben! A metódus segítségével olvassa be a autok.csv nevű állományt! Ügyeljen arra, hogy az állomány első sora az adatok fejlécét tartalmazza! (2 pont)
5. Határozza meg és írja ki a képernyőre, hány autó adatait tároltuk a forrásfájlban! (1 pont)
6. Számolja meg, hogy a forrásállományban szereplő autók átlagosan hány darabot adtak el, majd jelenítse meg a képernyőn 1 tizedesjegy pontossággal! (2 pont)
7. Írja ki a minta szerint azoknak az autóknak a márkáit és modelljeit, valamint a gyártási évet, amelyeket az elmúlt 5 évben gyártottak. (3 pont)
8. Írassa ki a mintának megfelelően, hogy az egyes márkákhoz hány eladott autó tartozik! (4 pont)

Minta

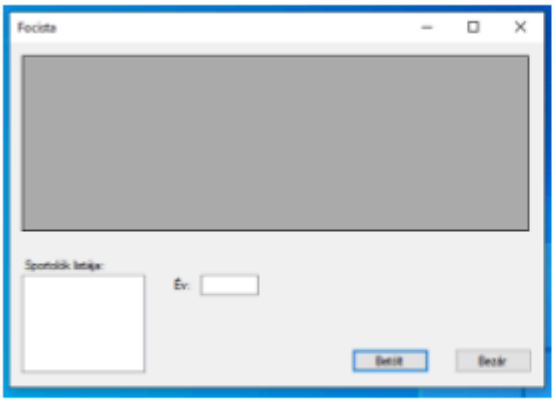
```
5. feladat: 15 autó található a listában
6. feladat: Az autók esetében az átlagosan eladott darabszám 625.3
7. feladat: Az elmúlt 5 évben gyártott autók:
    - Toyota Corolla: 2019
    - Volkswagen Golf: 2020
    - Hyundai i30: 2021
    - BMW 3 Series: 2019
    - Nissan Leaf: 2020
    - Tesla Model 3: 2021
    - Renault Clio: 2019
    - Mercedes-Benz A-Class: 2020
    - Audi A3: 2019
    - Chevrolet Bolt: 2021
8. feladat: Legsikeresebb márkák listája az eladott darabszám alapján:
    - Tesla: 1200 darab
    - Mercedes-Benz: 800 darab
    - Volkswagen: 800 darab
    - Hyundai: 700 darab
    - Renault: 670 darab
```

```
    - BMW: 650 darab
    - Audi: 620 darab
    - Ford: 600 darab
    - Peugeot: 590 darab
    - Nissan: 550 darab
    - Skoda: 550 darab
    - Toyota: 500 darab
    - Honda: 450 darab
    - Fiat: 400 darab
    - Chevrolet: 300 darab
```

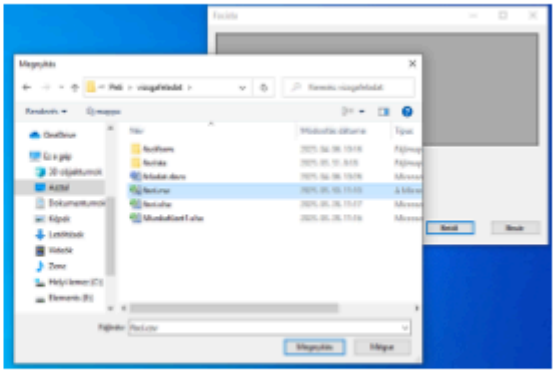
2. Asztali alkalmazás fejlesztés grafikus része

Készítsen grafikus alkalmazást a mintának megfelelően az előző feladatban használt autok.csv fájl adatainak megjelenítésére és kezelésére. A feladat megoldásához Java vagy C# programozási nyelvhet használhat.

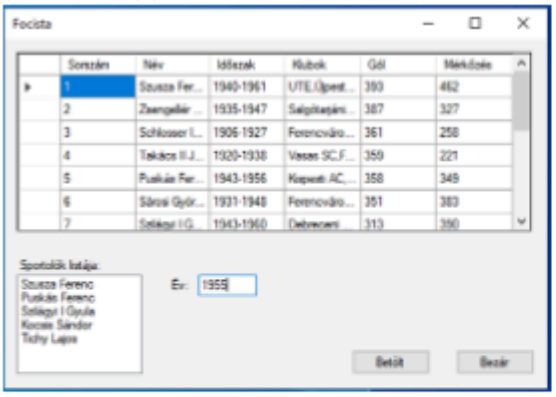
1. Készítsen Windows Forms alkalmazást, melynek projektjét `autoform` néven mentse el! Az alkalmazásablak címsorában megjelenő felirat `Autók` legyen! (1 pont)
2. Az előző feladatban elkészített saját `Auto` osztályt használja fel az új Windows Forms alkalmazásban. (1 pont)
3. Hozzon létre nyomógombot "Betölt" felirattal, melynek megnyomásakor megjelenik egy megnyitás párbeszéd ablak, majd ennek segítségével olvassa be az autok.csv állományt! Az adatok beolvasásához használja az előzőekben létrehozott Auto osztály megfelelő metódusát! (2 pont)
4. A betöltött adatokat jelenítse meg a mintához hasonló módon DataGridView vezérlő segítségével! (3 pont)
5. Használjon ListBox vezérlőt, majd kérjen be egy gyártási évet TextBox vezérlő segítségével! A beírt gyártási év alapján a ListBox vezérlőben jelenítse meg azon autók márkáit és modelleit, amelyek ebben az évben kerültek gyártásra! (2 pont).
6. Készítsen „Bezár” felirattal nyomógombot, amivel ki tudunk lépni a programból. Kilépés előtt kérdezzen rá, hogy valóban ki szeretnénk-e lépni. (2 pont)



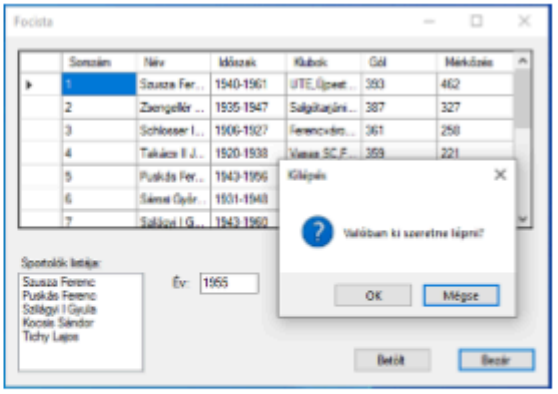
(a) Adatbetöltés előtti UI



(b) Fájlkiválasztás a betöltés gombra kattintva



(c) Betöltött adatok megjelenítése, évszám keresés



(d) Kilépést megerősítő üzenet

1. ábra. A felhasználói felület és műveletek különböző állapotai

3. Reszponzív viselkedésű weboldal

Forrásállományok letöltése.

A következő feladatban a Nevenincs Bt. 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.png, minta02.png és a minta03.png állományokban talál, melyeket tilos a megoldásában felhasználni!

Nyissa meg az *index.html, aruk.html, megrendeles.html* és a *style.css* állományokat és szerkessze azok tartalmát az alábbiak szerint:

1. A header elem jobb felső sarkában található napraforgó képet mindegyik oldalon alakítsa át hivatkozássá, kattintásra az index.html nyíljon meg

azonos lapon! A logó formázását módosítsa úgy, hogy kerüljön át a bal oldalra!

2. A footer elem stílusát módosítsa az oldalakon oly módon, hogy ne legyen eltartás, a teljes szélességet töltse ki! A feladatot a megfelelő Bootstrap osztály módosításával oldja meg!
3. A footer elemen belül nagy(large) méretnél az oszlopok arányát `2:4:2:4` arányban ossza el mindhárom oldalnál!
4. A „nyito” azonosítóval ellátott elem háttérképe legyen a fokep.jpg, ne ismétlődjön és töltse ki a teljes területet!
5. A vetőmagokat tartalmazó oldalon szúrjon be egy újabb árut az eddigiek után, az adatokat megtalálja az aru.txt állományban. Ügyeljen arra, hogy az új elem stílusa megegyezzen az oldalon lévő többi áru stílusával!
6. Állítson be árnyékot az árukat tartalmazó oldalon található növények képeire, amikor fölé húzzuk az egeret! Jobbra és lefele `2px`-es méretű, `20px`-es elmosódottságú és fekete színű legyen!
7. A dália képére kattintva a megrendeles.html nyíljon meg azonos oldalon!
8. Média query segítségével oldja meg, hogy max. 768px méretig a footer elemben lévő szöveg balra igazodjon!

4. Backend programozás

Az alábbi feladatban a virágbolt weboldalának backend szerverét kell elkészítenie.

Hozzon létre backend szerver projektet az Ön által választott programnyelven, illetve fejlesz- tési környezetben! A projektmappát `Vezeteknev_Keresztnev_backend` formában nevezze el! A feladat megoldása során ékezetmentes azonosítókat és állományneveket is használhat.

Készítsen adatbázist `viragbolt` néven, karakterkódolása legyen utf-8, illesztése a magyar szabályok szerinti! Mentse el az adatbázist létrehozó SQL scriptet!

4.1. Adatbázis létrehozása

Hozz létre egy adatbázisfájlt `viragbolt.db` néven.

tartozik

KATEGORIAK

int
id
PK
Egyedi azonosító
string
nev
Kategória neve
ARUK
int
id
PK
Egyedi azonosító
string
nev
Termék neve
string
leiras
Termék leírása
int
keszlet
Készlet mennyisége
int
ar
Termék ára
string
kepUrl
Termék képének URL címe
int
kategoriaId
FK
Kategória azonosítója

Az adatbázis tartalmát az alábbi script segítségével hozhatod létre:

```
CREATE TABLE IF NOT EXISTS kategoriak (
    id INTEGER PRIMARY KEY,
    nev TEXT NOT NULL
);

CREATE TABLE IF NOT EXISTS aruk (
    id INTEGER PRIMARY KEY,
    nev TEXT NOT NULL,
    kategoriaId INTEGER,
    leiras TEXT,
    keszlet INTEGER,
    ar INTEGER,
    kepUrl TEXT,
    FOREIGN KEY(kategoriaId) REFERENCES kategoriak(id)
```

```
);
```

4.2. Virágok lekérdezése

- Lekérdezi az összes virágot az adatbázisból.
- A válasz tartalmazza a virág kategóriáját és annak információt is.

Metódus	URL	Body	Válasz
GET	`/api/flowers`	üres	JSON

Lehetséges válasz üzenet (200 OK státuszkóddal)

```
[
    {
        "id": 1,
        "nev": "Rózsa",
        "kategoria": {
            "id": 1,
            "nev": "Virágok"
        },
        "leiras": "A rózsa a rózsafélék (Rosaceae) családjába tartozó növények közös neve. A
rózsafélék családjába mintegy 100-150 faj tartozik, melyek a mérsékelt égövön őshonosak.",
        "keszlet": 10,
        "ar": 1000,
        "kepUrl": "https://example.com/roza.jpg"
    },
    {
        "id": 2,
        "nev": "Tulipán",
        "kategoria": {
            "id": 1,
            "nev": "Virágok"
        },
        "leiras": "A tulipán (Tulipa) a liliomfélék (Liliaceae) családjába tartozó növények
nemzetsége. A tulipánok a törökországi Szökevény-hegységben őshonosak.",
        "keszlet": 20,
        "ar": 800,
        "kepUrl": "https://example.com/tulipan.jpg"
    }
]
```

4.3. Virág lekérdezése

- Lekérdezi egy virág nevét és leírását az adatbázisból az azonosító alapján.
- A válasz tartalmazza a virág kategóriáját és annak információt is.
- Ha nem található a virág, akkor egy 404-es hibakódot küld vissza.

Metódus	URL	Body	Válasz
GET	`/api/flowers/<id>`	üres	JSON

Lehetséges válasz üzenet (200 OK státuszkóddal)


```
{
    "id": 1,
    "nev": "Rózsa",
    "kategoria": {
        "id": 1,
        "nev": "Virágok"
    },
    "leiras": "A rózsa a rózsafélék (Rosaceae) családjába tartozó növények közös neve. A rózsafélék családjába mintegy 100-150 faj tartozik, melyek a mérsékelt égövön őshonosak.",
    "keszlet": 10,
    "ar": 1000,
    "kepUrl": "https://example.com/roza.jpg"
}
```


Lehetséges válasz üzenet (404 NOT FOUND státuszkóddal)


```
{
    "error": "A virág nem található!"
}
```


4.4. Új virág hozzáadása

- Új virág hozzáadása az adatbázishoz.
- Az adatokat a kérés törzsén keresztül kell elküldeni JSON formátumban.
- Sikeres hozzáadás esetén a válaszkód *201 CREATED* legyen.

Metódus	URL	Body	Válasz
POST	`/api/flowers`	JSON	JSON

4.5. Virág frissítése

- Egy meglévő virág adatainak frissítése az adatbázisban.
- Támogassa a *név, leírás, készlet, ár* frissítését.
- Az adatokat a kérés törzsén keresztül kell elküldeni JSON formátumban.
- Ha nem található a virág, akkor egy 404-es hibakódot küld vissza.

Metódus	URL	Body	Válasz
PUT	`/api/flowers/<id>`	JSON	JSON

Példa a kérés törzsében elküldött JSON tartalmára


```
{
```

```
    "nev": "Vörös rózsa",
    "leiras": "A vörös rózsa a rózsafélék (Rosaceae) családjába tartozó növények közös neve.",
    "ar": 1200,
    "keszlet": 5
}
```


Lehetséges válasz üzenet (404 NOT FOUND státuszkóddal)


```
{
    "error": "A virág nem található!"
}
```


4.6. Virág törlése

- Egy meglévő virág törlése az adatbázisból.
- Ha nem található a virág, akkor egy 404-es hibakódot küld vissza.

Metódus	URL	Body	Válasz
DELETE	`/api/flowers/<id>`	nincs	JSON

Lehetséges válasz üzenet (404 NOT FOUND státuszkóddal)


```
{
    "error": "A virág nem található!"
}
```


4.7. Kategóriák lekérdezése

- Lekérdezi az összes kategóriát az adatbázisból.

Metódus	URL	Body	Válasz
GET	`/api/categories`	üres	JSON

Lehetséges válasz üzenet (200 OK státuszkóddal)


```
[
    {
        "id": 1,
        "nev": "Szobanövények"
    },
    {
        "id": 2,
        "nev": "Szabadtéri növények"
    }
]
```

4.8. Tesztelés

- Készíts Postman kollekciót, amely teszteli az összes elérhető végpontot, hogy könnyen tesztelhető legyen az alkalmazás.
- Az elkészített kollekciótexportált ki, a fájlnév legyen `Viragbolt.postman_collection.json`.

5. Frontend programozás

A következő feladatban egy frontend alkalmazás készítését kell elvégeznie a kiadott leírás szerint! Az alkalmazás felhasználói felületének kialakítását a reszponzív weboldal feladatrésznél szereplő minták segítik.

Az adatokat az előző feladatban elkészített backend szerver szolgáltatja, az ott létrehozott végpontokat kell használnia, az ott ismertetett formában kell érkeznie a válaszoknak.

Ha nem, vagy csak részben sikerült megoldania az előző feladatokat, akkor az alábbi API végpontok használatával oldja meg ezeket a feladatokat:

- `https://faiskola.richardkorom.hu/api/docs`

Feladatok

Az Ön által választott JavaScript keretrendszer (Angular, VueJs vagy React) valamelyikével készítse el az alábbi feladatokat:

1. Adja hozzá az alkalmazáshoz a választott UI keretrendszer (pl.: Bootstrap) legfrissebb verzióját!
2. Hozzon létre egy komponenst, amely a webalkalmazás nyitó oldala lesz:
    - A komponenst a "3. Reszponzív viselkedésű weboldal" feladatánál található minta alapján hozza létre!
    - A komponens létrehozása során használhatja a *3. Reszponzív weboldal* feladatrész állományait. (Pl.: index.html, style.css)
3. Hozzon létre egy újabb komponenst a virágbolt termékeinek a megjelenítéséhez!
    - A komponenst a *3. Reszponzív weboldal* feladatrész állományait használva hozza létre. (Pl.: *aruk.html*)
    - Az oldalon szereplő adatokat a kiszolgáló szerver `/api/flowers` URL-jén szereplő REST API függvénnyel kérdezze le!

- Állítsa be a komponensnek az eléréséhez a `/flowers` útvonalat!
- A kezdőoldalon a "Virágbolt termékei" feliratú részhez állítsa be, hogy arra kattintva navigáljon át a `/flowers` útvonalra!

4. Hozza létre az adott termék megrendelését megvalósító komponenst!
   - A komponens létrehozása során használhatja a *3. Reszponzív weboldal* feladatrész állományait. (Pl.: *aruk.html*)
   - Az oldalon szereplő adatokat a kiszolgáló szerver `/api/flowers/{id}` URL-jén szereplő REST API függvénnyel kérdezze le!
   - A mennyiség, és a "Megrendelem" gomb csak akkor jelenjen meg, ha van készleten az áruból, különben a "Jelenleg nincs a termékből készleten, keresse fel oldalunkat később!" üzenet jelenjen meg!
   - Ha van belőle készleten, akkor a mennyiség beviteli mező értéke alapértelmezetten 1 legyen, s ne vehessen fel nagyobb értéket, mint amennyi a készlet az adott termékből!
   - A "Megrendelés" gomb megnyomásakor küldje el a szerver `api/flowers` URL-re a készlet értékét módosításra! (készlet értéke – a megrendelt mennyiség)
   - Sikeres küldés esetén jelenítsen meg üzenetet a sikeres módosításról, majd navigáljon át a `/flowers` útvonalra! Hiba esetén a sikertelenségről is jelenítsen meg üzenetet, majd szintén navigáljon át a `/flowers` útvonalra!
   - Állítsa be a komponensnek az eléréséhez a `/rendeles` útvonalat!
   - A termékek oldalon állítsa be, hogy az adott fa képére kattintva navigáljon át a `/rendeles` útvonalra!
5. Forráskódját a választott technológiának megfelelően adja le:
   - Tömörítés előtt törölje a felesleges állományokat! Ügyeljen arra, hogy feladatmegoldást tartalmazó mappát/állományt ne töröljön!
   - A forráskódját tömörítse be *Vezetéknév_Keresztnév_frontend.zip* néven!