Vizsgázó neve:………………………..

Szakmai vizsga

**Projektfeladat**

Az ágazat megnevezése: **Informatika és távközlés**

Szakma megnevezése: **Szoftverfejlesztő és -tesztelő technikus szakma**

A vizsgatevékenység megnevezése:

**Asztali- és webes szoftverfejlesztés, adatbázis-kezelés feladatsor**

Időtartam: **240 perc**

A vizsgatevékenység aránya a teljes szakmai vizsgán belül:

**90%**

Jóváhagyta:

Dátum: 2023. ……………………………

……………………………..

Módos Gábor

igazgató

**2023**

Vizsgaszervező neve:

**Győri SZC Jedlik Ányos Gépipari és Informatikai Technikum és Kollégium**

Vizsga dátuma:

**--------------**

Vizsga helyszíne:

**Győri SZC Jedlik Ányos Gépipari és Informatikai Technikum és Kollégium**

**9021 Győr, Szent István út 7.**

Használható segédeszköz:

A gyakorlati vizsgatevékenység során a jelölt, a feladat kidolgozása közben az alábbi eszközöket használhatja:

* **internethasználattal rendelkező asztali számítógép**

# Grafikus és konzolos részt egyaránt tartalmazó asztali alkalmazás fejlesztése

A következő feladatban az Északi-középhegység néhány kilátójának fontosabb adatait állnak 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:* 3. 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.*
* *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 következő feladatok megoldásához két .csv kiterjesztésű fájl áll a rendelkezésére adatforrásként a forrás könyvtárban. Mindkét állomány UTF-8 kódolású és az első sor tartalmazza a mezőneveket.

**locations.csv – hegységeket tartalmazza**

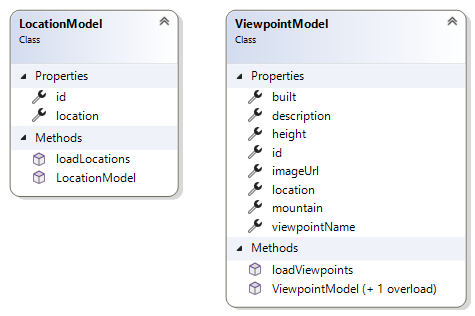
* id: kulcs mező, egész szám
* location: hegység neve, szöveg

**viewpoints.csv – kilátók adatait tartalmazza**

* id: kulcs mező, egész szám
* viewpointName: a kilátó neve, szöveg
* mountain: a hegy neve, ahol a kilátó található, szöveg
* height: a kilátó magassága méterben, valós szám
* description: leírás, szöveg
* built: építés éve, dátum
* imageUrl: képre mutató webes hivatkozás
* location: a location.csv-ben lévő kapcsolódó rekordra mutató idegen kulcs, szám

**Feladatok**

1. Készítsen konzolos alkalmazást a következő feladatok megoldására, melynek projektjét Eszaki\_kozephegyseg\_kilatoi néven mentse el!
2. Készítsen saját osztályokat LocationModel és ViewpointModel azonosítóval, melynek az adattagjainak azonosítóit és láthatósági szintjét az alábbi osztálydiagramok szemléltetik!



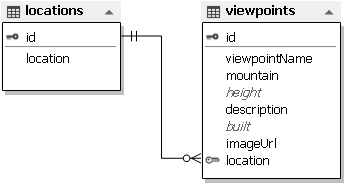
1. ábra Osztálydiagramok

1. Készítsen statikus metódust a LocationModel osztályhoz az adatforráshoz történő kapcsolódásra, és az adatforrásban levő adatok betöltésére! A metódus neve loadLocations legyen! A metódus visszatérési értéke a LocationModel osztályból képzett lista legyen! A metódus paramétere a betöltendő fájl neve legyen! Az adatfájl soraiban lévő adatok feldolgozásához használjon konstruktort!
2. Készítsen statikus metódust ViewpointModel osztályhoz az adatforráshoz történő kapcsolódásra, és az adatforrásban levő adatok betöltésére! A metódus neve loadViewpoints legyen! A metódus visszatérési értéke a ViewpointModel osztályból képzett lista legyen! A metódus paramétere a betöltendő fájl neve legyen! Az adatfájl soraiban lévő adatok feldolgozásához használjon konstruktort!
3. Olvassa be, tárolja el az adatforrások adatait a LocationModel és ViewpointModel osztályok segítségével egy olyan összetett adatszerkezetben, amely használatával a további feladatok megoldhatók!
4. Határozza meg és írja ki a minta szerint a legmagasabb kilátó adatait!

A képen szöveg látható

Automatikusan generált leírás

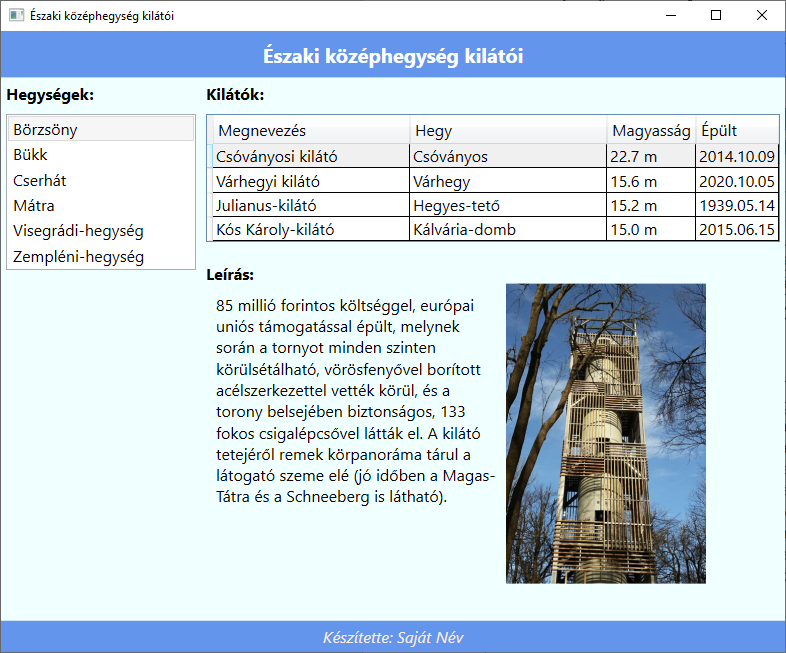
1. A grafikus alkalmazás elkészítése előtt a lokális adatbáziskezelő rendszerébe importálja be a forrás könyvtárban levő database.sql fájlt! *Ha a 3. feladatrész (Backend programozás) 2. feladata alapján hozott már létre adatbázist, akkor ezt a lépést kihagyhatja (A két feladatrész ugyanazt az adatbázist használja).* Az adatbázis struktúrája:



1. Készítsen grafikus alkalmazást a következő feladatok megoldására, melynek projektjét Eszaki\_kozephegyseg\_kilatoi\_GUI néven mentse el!
2. A megjelenő ablak a minták alapján reszponzív viselkedésű legyen! Az ablakban levő vezérlők típusai és a feliratok feleljenek meg a mintának!
3. Hozzon létre Models néven alkönyvtárat a projekten belül! Generálja le/Hozza létre az adatbázis kapcsolódáshoz, illetve a modellekhez szükséges osztályokat a Models mappába! A feladat megoldásánál használhat Entity Framework-öt.
4. A képernyő bal oldalán levő listában egy elemet kiválasztva az adott hegységben található kilátók adatai jelenjenek meg az ablak jobb felső részén levő, a mintának megfelelő komponensben! Az adatok megjelenítését adatkötésekkel oldja meg!
5. Amennyiben az ablak jobb felső részén levő rácsban kiválasztásra kerül egy kilátó, alatta jelenjen meg a leírása és a hozzá tartozó fotó!
6. A megjelenítéshez használt ablakmérete legalább 800x660 képpont, az ablakban használt betűméret 16 pontos legyen!
7. A fejléc és a lábrész háttérszíne CornflowerBlue, a középső rész háttérszíne Azure legyen!
8. Az ablak jobb felső részén levő rácsban kerüljön kikapcsolásra az új sor beszúrás, a sortörlés, az oszlop és sor átméretezés, a rendezés, és ne lehessen a cellákat szerkeszteni! Csak a minta szerinti oszlopok létezzenek és az adatok magasság szerint csökkenő sorrendben jelenjenek meg!
9. A kép férjen el arányosan a szöveg mellett!

*Minden további paraméter értéke szabadon megválasztható.*

**Minták**





# Reszponzív viselkedésű weboldal – Katar VB

A következő feladatban a katari foci világbajnokságról szóló weboldalt 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, minta03.png és a minta04.png állományokban talál, melyet tilos a megoldásában felhasználni!

Nyissa meg az index.html, stadionok.html, csapatok.html, jegyvasarlas.html, a jegykalk.js és a style.css állományokat és szerkessze azok tartalmát az alábbiak szerint:

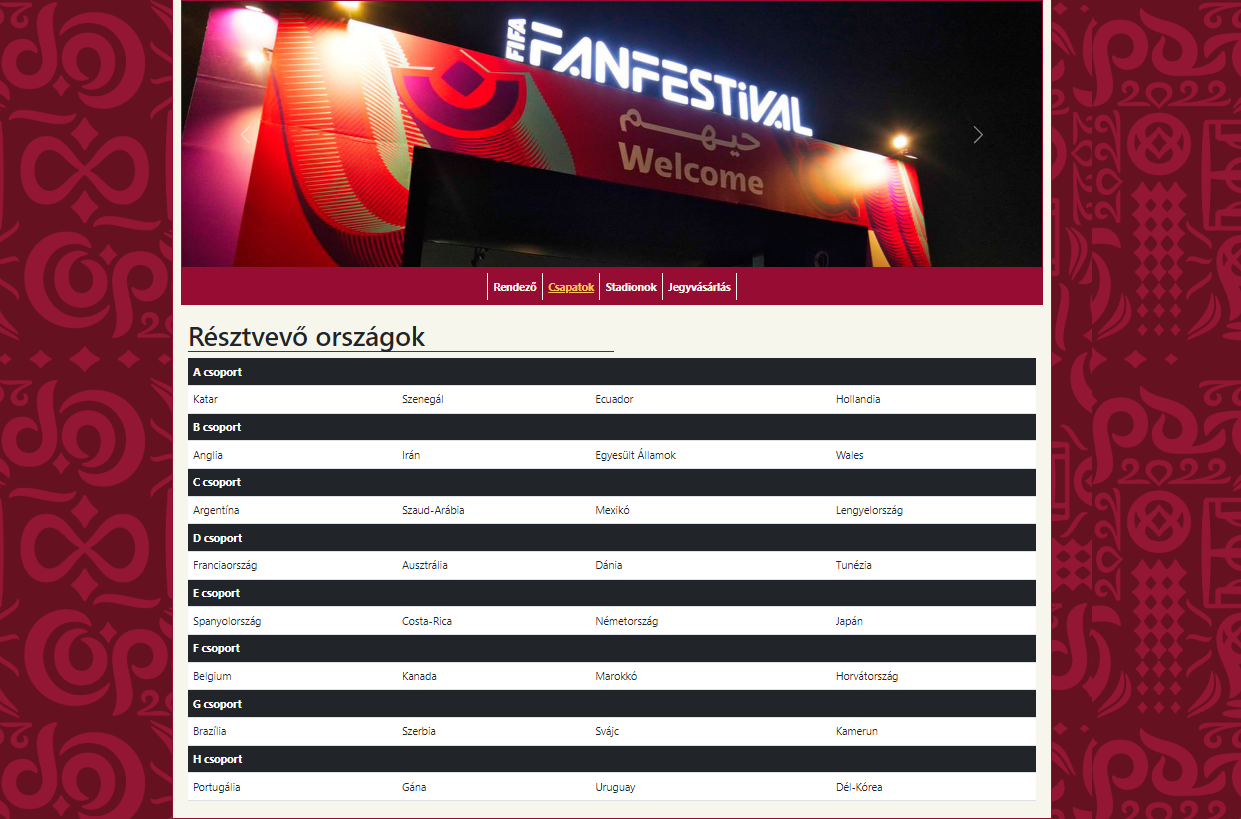
1. A nyitó oldal (index.html) oldal forráskódja hibás, mivel hiányzik belőle a head és a body html elem. Szúrja be a hiányzó elemeket a megfelelő helyre!
2. Állítsa be a stíluslapon, hogy az oldal háttere a hatter.png kép legyen!
3. A stíluslapon a megfelelő Bootstrap osztályra történő hivatkozással állítsa be, hogy az aktív laphoz tartozó menüpont betűszíne#fec459legyen és aláhúzottan jelenjen meg!
4. A stadionok.html oldalra szúrjon be egy új stadiont a már ott található stadionok után! A stadion adatait megtalálja a stadion.txt állományban. A létrehozott elemeken alkalmazza ugyanazokat a stílusosztályokat, melyeket a többi stadionnál is megtalál!
5. A stíluslapon új szelektor segítségével oldja meg, ha a stadionok vagy a jegyvásárlás oldalon található gomb(ok) fölé visszük az egeret, akkor azok átlátszósága 50%-kal csökkenjen! (Halványabb legyen.)
6. A megfelelő Bootstrap osztályok felhasználásával érje el, hogy a stadionok oldalon található kártyák nagyméretű kijelzőkön 4, közepes méretű kijelzőkön 2 oszlopban jelenjenek meg!
7. A csapatok.html oldalon a táblázat fejléc típusú elemeinél alkalmazzon oszlopösszevonást úgy, hogy a csoportok neveit tartalmazó mezők töltsék ki vízszintesen a táblázat teljes szélességét!
8. A jegyvasarlas.html oldalon egy űrlap található. Rendelje a gombra kattintás eseményhez a kalkulal JavaScript függvényt!
9. Módosítsa az űrlapon a személyek száma megadásához tartozó mezőt úgy, hogy a felhasználó csak számértéket vihessen be, az érték ne lehessen kisebb, mint 1 és nagyobb, mint 10!
10. Az űrlapon a „Név” és „E-mail” szöveghez tartozó mezők alapértelmezetten tartalmazzák a „pl: Kovács Béla” és „valami@valami.hu” helyfoglaló példaszövegeket, melyek automatikusan tűnjenek el, ha a felhasználó gépelni kezd!
11. A jegykalk.js állományban kizárólag a JavaScript függvény vázát találja. Írja meg a függvényt úgy, hogy az a leírtaknak megfelelően működjön:
    * A „Név” és „E-mail” szövegekhez tartozó mezők kitöltése kötelező.
    * Az adatvédelmi szabályzatot kötelező elfogadni.
    * Amennyiben a felhasználó valamely mezőt nem töltötte ki, vagy nem fogadta el az adatvédelmi tájékoztatót, akkor a JavaScript alert() függvényének segítségével jelenítse meg „A név és e-mail mező kitöltése, valamint az adatvédelmi tájékoztató elfogadása kötelező!” szöveget!
    * Ha minden mező kitöltése megfelelő és a felhasználó a gombra kattint, akkor számolja ki a jegy(ek) árát a létszám és az alábbi táblázat alapján! Az árat jelenítse meg az oldalon az erre szolgáló HTML elem segítségével!

|  |  |
| --- | --- |
| **Szektor** | **Jegy ár (1 fő)** |
| A szektor | 250 000 Ft |
| B szektor | 225 000 Ft |
| C szektor | 200 000 Ft |
| D szektor | 175 000 Ft |
| E szektor | 150 000 Ft |
| F szektor | 125 000 Ft |
| G szektor | 100 000 Ft |

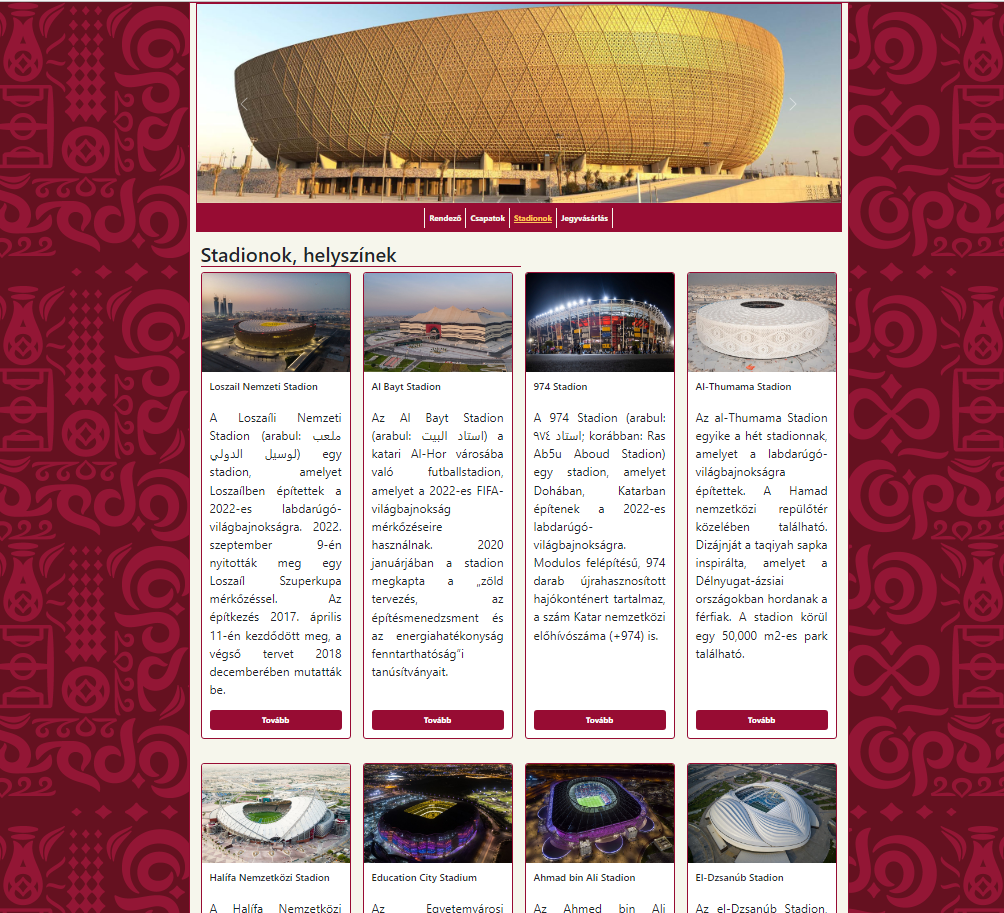
**index.html minta:**



**csapatok.html minta:**



**stadionok.html minta:**



**jegyvasarlas.html minta:**

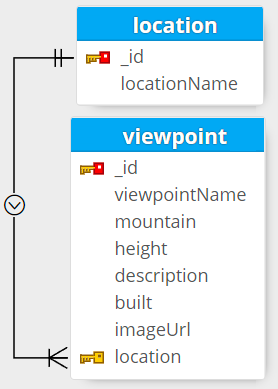


# Backend programozás – Északi-középhegység kilátói

Ebben a feladatban az Északi-középhegység néhány kilátójának fontosabb adatait kiszolgáló backend-szerver alkalmazást kell elkészítenie a leírás és a minták alapján.

Megoldása csak akkor lesz teljes értékű, ha a projektjében a modell(ek) és kontroller(ek) pontosan a feladatspecifikáció szerintiek, felesleges kódot nem tartalmaznak.

1. Hozzon létre backend-szerver projektet az Ön által választott programnyelven! A projektmappát „Vezetéknév\_Keresztnév\_backend” formában nevezze el!
2. Indítsa el a választott lokális adatbázis szervert, majd futtassa a forrás mappában lévő megfelelő parancsállományt (database.sql, vagy mongo.bat) az adatbázis és a táblák (kollekciók) létrehozásához, valamint a mintaadatok feltöltéséhez! *Ha MySql-t választ adatforrásként és az 1. feladatrész (Grafikus és konzolos részt egyaránt tartalmazó asztali alkalmazás fejlesztése) 7. feladata alapján hozott már létre adatbázist, akkor ezt a lépést kihagyhatja (A két feladatrész ugyanazt az adatbázist használja).*
3. Hozzon létre modelleket az alábbi ábra alapján! A location modell a hegységek, a viewpoint modell pedig a kilátókhoz tartozó tulajdonságokat határozza meg! A modellekben állítsa be a mezőket, adattípusokat és a jelzett megszorításokat!



|  |  |
| --- | --- |
| **típus és megszorítás** | **példa** |
| ***Egész szám*, kötelező, elsődleges kulcs** | 1 |
| ***Szöveg*, egyedi, kötelező, max: 30 karakter** | "Börzsöny" |

|  |  |
| --- | --- |
| **típus és megszorítás** | **példa** |
| ***Egész szám*, kötelező, elsődleges kulcs** | 2 |
| ***Szöveg*, egyedi, kötelező, max: 50 karakter** | "Julianus-kilátó" |
| ***Szöveg*, egyedi, kötelező, max: 30 karakter** | "Hegyes-tető" |
| ***Tizedes törtet tartalmazó szám (double)*** | 15,2 |
| ***Szöveg*, kötelező** | "A csaknem tizen…" |
| ***Dátum*** | 1939-05-14 |
| ***Szöveg*, kötelező, max: 50 karakter** | "…/julianus.jpg" |
| ***Egész szám*, kötelező, idegen kulcs** | 1 |

1. Továbbá állítsa be a következő megszorításokat is a megfelelő modellben:
   1. Ha az imageUrl mező értéke nem kerül megadásra, akkor az alapértelmezett érték   
      "http://elit.jedlik.eu/viewpoints/no-img.jpg" legyen!
   2. A built mezőbe nem kerülhet az aktuális dátumnál nagyobb dátum! Ha a feltétel nem teljesül, akkor "Az aktuális dátumnál nem adhat meg későbbi dátumot a built mezőben!" szöveg legyen megtalálható a hibaüzenetben!
   3. A height mezőbe legalább 1 m magasságot kell megadni a kilátó magasságának! Ha a feltétel nem teljesül, akkor "Egy kilátónak legalább 1 méter magasnak kell lennie!" szöveg legyen megtalálható a hibaüzenetben!

**A feladatok megoldása során ügyeljen arra, hogy a megoldása az elvárt végpontokon kívül egyéb végpontokat ne tartalmazzon! A végpontok funkcionalitását megvalósító kontrollerekben a szükséges metódusokon kívül felesleges kódokat ne hozzon létre!**

1. Hozzon létre egy végpontot, amely egy megadott nevű hegység kilátóit listázza ki! Amennyiben van legalább egy találat, akkor adja vissza a talált kilátókat! Ha egyetlen kilátó sincs az adott hegységben, akkor az alábbi minta szerinti válaszüzenetet küldje vissza!

|  |  |  |  |
| --- | --- | --- | --- |
| Metódus | URL | Body | Válasz |
| GET | */api/locations/:locationName/viewpoints* | üres | JSON array/ JSON object |

Például (**200 OK** státuszkóddal):

[

{

"viewpointName": "Kalapat-hegyi kilátó",

"mountain": "Kalapat-tető",

"location": 2,

"height": 33.3,

"description": "A kilátótoronyból körpanoráma…",

"built": "2000-08-20T00:00:00.000Z",

"imageUrl": "http://elit.jedlik.eu/viewpoints/kalapat.jpg"

}

...

]

Válasz üzenet (**404 NOT FOUND** státuszkóddal):

{"message": "Ebben a hegységben nem találtam kilátót."}

1. Készítsen végpontot a /api/viewpoints/:id útvonalra, melynek használatával **PATCH** típusú kéréssel a megadott azonosítójú kilátó adatait tudja módosítani az adatbázisban!   
   Ha a megadott kilátó nem létezik, akkor a "${id} azonosítóval nem létezik kilátó!" üzenetet és a 404‑es státuszkódot küldje vissza válaszként! A végponthoz tartozó kódsorokat védje try‑catch szerkezettel és hiba esetén a beépített hibaüzenetet küldje vissza a válaszüzenetben 400‑as státuszkóddal! Sikeres módosítás esetén a 200‑as státuszkódot és a módosított dokumentumot (rekordot) küldje válaszként!

|  |  |  |  |
| --- | --- | --- | --- |
| Metódus | URL | Body | Válasz |
| PATCH/PUT | /api/viewpoints/:id | JSON object | JSON object |

|  |  |
| --- | --- |
| Minta body (**PATCH**): | Minta body (**PUT**): |
| {  "height": 30.0,  "built": "2015-05-04"  } | {  "viewpointName": "Galya-kilátó",  "mountain": "Galya-tető",  "location": 4,  "height": 30,  "description": "Az ország legmagasabban álló…",  "built": "2015-05-04T00:00:00.000Z",  "imageUrl": <http://elit.jedlik.eu/viewpoints/galya.jpg>  } |

Minta válasz sikeres módosítás után:

{

"\_id": 10,

"viewpointName": "Galya-kilátó",

"mountain": "Galya-tető",

"location": 4,

"height": 30,

"description": "Az ország legmagasabban álló…",

"built": "2015-05-04T00:00:00.000Z",

"imageUrl": "http://elit.jedlik.eu/viewpoints/galya.jpg"

}

Minta hibaüzenetek:

{"message": "19 azonosítóval nem létezik kilátó!"}

{"message": "Validation failed: built: Az aktuális dátumnál nem adhat meg   
későbbi dátumot a built mezőben!"}

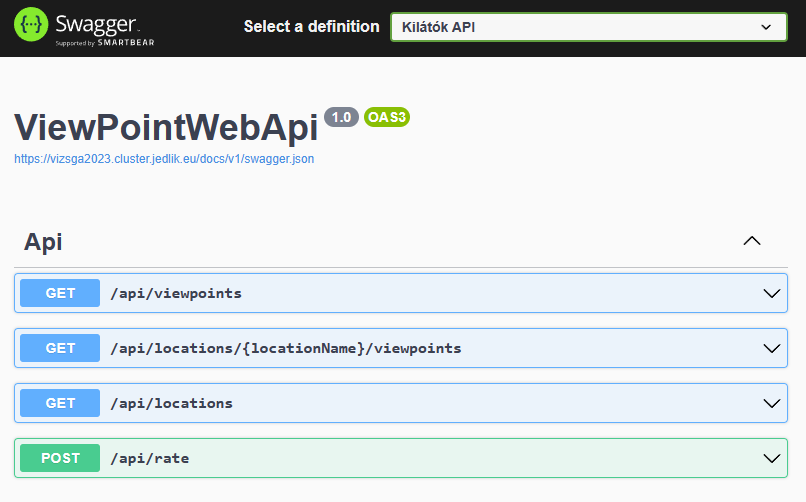
1. Hozzon létre Thunder Client/Postman kollekciót, melyben az Ön által létrehozott végpontokat ellenőrzi! Exportálja a kollekciót a projekt gyökér könyvtárába Teszt.json néven!
2. Adja le a teljes projektmappát tömörítve (node\_modules könyvtár nélkül)!

# Frontend programozás

A következő feladatban a magyarországi kilátókat bemutató front-end webalkalmazást kell készítenie, ahol a látogatóknak lehetőségük van értékelni az egyes kilátókat. A webalkalmazást a kiadott leírás szerint kell elkészítenie, a felhasználói felületek kialakítását segédanyagok, minták (pl.: Minta.mp4) segítik, melyeket **nem kötelező** követnie a megoldásában.

A feladat elkészítése során a https://vizsga2023.cluster.jedlik.eu címen található REST API függvényekkel kell dolgoznia.

A https://vizsga2023.cluster.jedlik.eu/docs URL címen segítséget kap a webszerveren elérhető REST API függvényekkel kapcsolatban:



Az elérhető REST API függvények:

* /api/locations

Metódus: GET

Válasz: (200-as státuszkóddal) Az adatbázisban található hegységek adatai

Válasz minta:

[

{

"id": 1,

"locationName": "Börzsöny"

},

{

"id": 2,

"locationName": "Bükk"

},

{

"id": 3,

"locationName": "Cserhát"

},

...

]

* /api/locations/{locationName}/viewpoints

Metódus: GET

Válasz: (200-as státuszkóddal) Az adatbázisban a {locationName} nevű hegységben található kilátók részletes adatai.

Válasz minta:

[

{

"viewpointName": "Avasi kilátó",

"mountain": "Avas",

"location": 2,

"height": 72.2,

"description": "Tévétoronyként és kilátóként egyaránt ...",

"built": "1963.05.04",

"imageUrl": "https://vizsga2023.cluster.jedlik.eu/images/avas.jpg"

},

...

{

"viewpointName": "Petőfi-kilátó",

"mountain": "Bálvány",

"location": 2,

"height": 18.5,

"description": "A Bükk-vidék legmagasabban fekvő ...",

"built": "2016.06.22",

"imageUrl": "https://vizsga2023.cluster.jedlik.eu/images/petofi.jpg"

}

]

* /api/viewpoints

Metódus: GET

Válasz: (200-as státuszkóddal) Az adatbázisban található összes kilátó neve

Válasz minta:

[

{

"id": 8,

"viewpointName": "Avasi kilátó",

"mountain": "Avas"

},

...

{

"id": 4,

"viewpointName": "Várhegyi kilátó",

"mountain": "Várhegy"

},

{

"id": 14,

"viewpointName": "Vörös-kő kilátó",

"mountain": "Vörös-kő"

}

]

* /api/rate

Metódus: POST

A kérés törzsében kell a szerver felé elküldeni az értékelés adatait.

Minta a kérésre:

{

"viewpointId": 5,

"email": "minta.geza@email.hu",

"rating": 8,

"comment": "Szép kilátás, kellemes séta a kilátóig"

}

Sikeres rögzítés esetén a válasz: (201-es státuszkóddal):

{

"count": 8,

"average": 8.25

}

Sikertelen rögzítés esetén (400-as státuszkóddal):

Hibaüzenet szövege

**Feladatok**

Az Ön által választott JavaScript keretrendszer (Angular, VueJs vagy React) valamelyikével készítse el az alábbi feladatokat! A feladatok elkészítését forrásállományok segítik, melyek a választott technológiának megfelelő alkönyvtárban találhatók a források mappában.

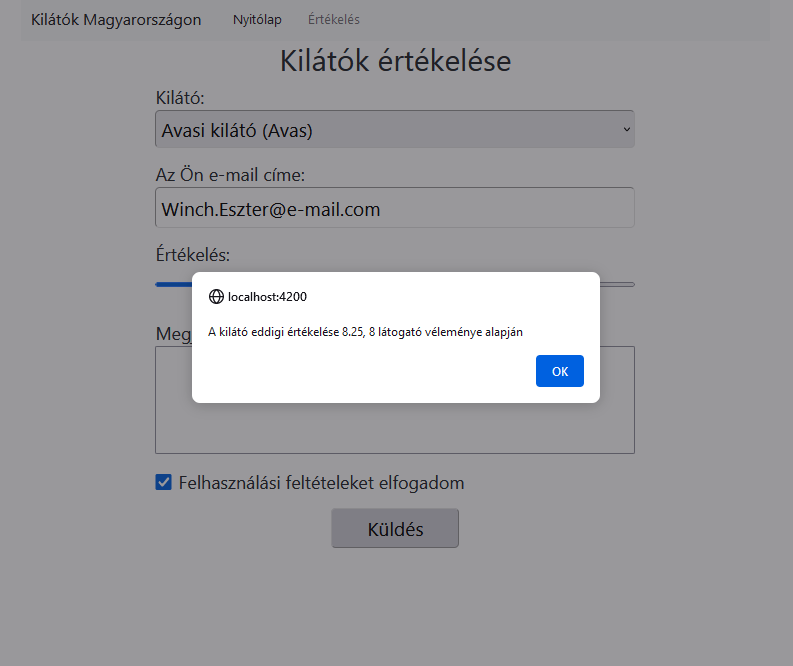
1. Készítse el az alkalmazás menüjeként szolgáló navigációs sávot az alábbi minta alapján:

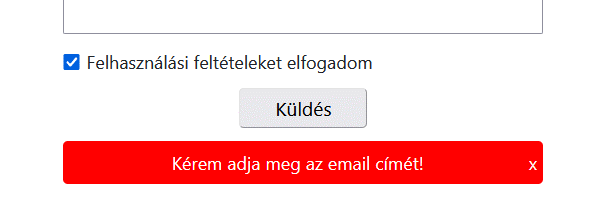


1. Hozzon létre egy komponenst, ami a webalkalmazás nyitó oldala lesz!
   1. Az oldal tetején lévő – a hegységeket tartalmazó – lenyíló lista lehetőségeinek feltöltéséhez használja a /api/locations URL-en található REST API függvényt!
   2. A kiválasztott hegységben lévő kilátók adatait a kiszolgáló szerver /api/locations/{locationName}/viewpoints URL-jén található REST API függvénnyel kérdezze le! (*Az URL-ben szereplő {locationName} helyére a kiválasztott hegység nevét illessze be!)*
   3. Az adatokat a kiadott mintához hasonlóan jelenítse meg! A megjelenítés során használhatja a forrás könyvtárban található állományokat.
   4. A kezdőoldalon a „Nyitólap” felirathoz állítsa be, hogy arra kattintva navigáljon a program erre a komponensre!



1. Hozzon létre egy újabb komponenst amely az értékelést fogja elvégezni:
   1. A komponens létrehozása során használhatja a források könyvtárban szereplő állományokat.
   2. Az összes kilátót tartalmazó lenyíló listához az elemeket a szerver /api/viewpoints URL-jéről töltse le!
   3. A „Küldés” gombra kattintva az űrlapon szereplő adatokat küldje el a szerver /api/rate URL-re az API leírás alapján!
   4. Sikeres küldés esetén jelenítse meg a szerver által visszaadott adatokat: *eddig hányan értékelték a kilátót, és jelenleg mennyi az átlagos értékelése*, majd navigáljon át a kilátók leírását tartalmazó oldalra.
   5. Sikertelen küldés esetén jelenítse meg a szervertől kapott hibaüzenetet!
   6. Állítsa be ennek a komponensnek az eléréséhez a /rating útvonalat!
   7. A kezdőoldalon az „Értékelés” felirathoz állítsa be, hogy arra kattintva navigáljon át a /rating útvonalra!





1. Forráskódját a választott technológiának megfelelően adja le:
   1. Tömörítés előtt törölje a felesleges állományokat! **Ügyeljen arra, hogy feladatmegoldást tartalmazó mappát/állományt ne töröljön!**
   2. A forráskódját tömörítse be Vezetéknév\_Keresztnév\_frontend.zip néven!