Tartalomjegyzék

[1. Feldolgozandó adatbázisok 4](#_Toc121083466)

[2. CRUD Alkalmazás 5](#_Toc121083467)

[2.1. A feladat leírása 5](#_Toc121083468)

[2.2. Látványtervek a projekthez 5](#_Toc121083469)

[3. RestFul kliens Alkalmazás 7](#_Toc121083470)

[3.1. A feladat leírása 7](#_Toc121083471)

[3.2. A kód dokumentálva 7](#_Toc121083472)

[4. RestFul szerver Alkalmazás 11](#_Toc121083473)

[4.1. A feladat leírása 11](#_Toc121083474)

[5. SOAP kliens Alkalmazás 12](#_Toc121083475)

[5.1. A feladat leírása 12](#_Toc121083476)

[6. Adatbányász menü 13](#_Toc121083477)

[6.1. A feladat leírása 13](#_Toc121083478)

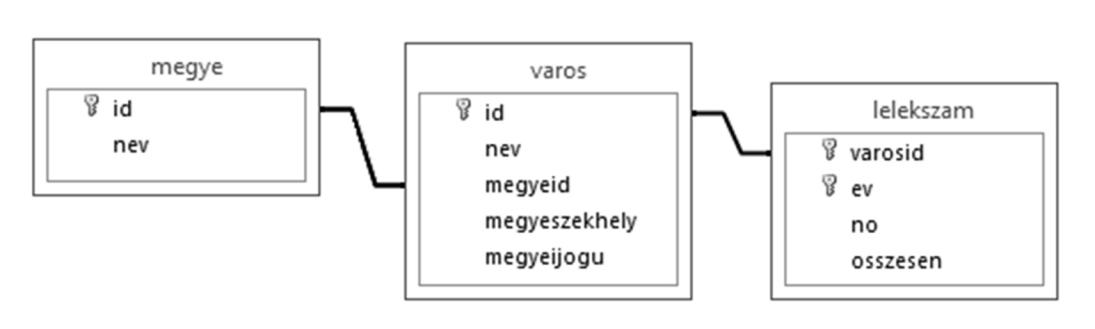
[7. Egyéb menü 14](#_Toc121083479)

[7.1. A feladat leírása 14](#_Toc121083480)

1. Feldolgozandó adatbázisok

Hazánkban a városok száma 1990-től dinamikusan emelkedett, 2019-re megközelítette a 350-et. A várossá nyilvánítás szabályai 2015-ben jelentősen szigorodtak, így a közeljövőben kevés település kaphat városi rangot. Az adatbázis a jelenlegi városok és Budapest kerületeinek néhány adatát tárolja. A városok lélekszámát a várossá nyilvánítás évét követő évtől évente tartalmazza.

|  |  |  |
| --- | --- | --- |
| **Táblák:** |  |  |
| ***megye*** (*id*, *nev*) |  |  |
| *id* | A megye azonosítója (szám), ez a kulcs |  |
| *nev* | A megye neve (szöveg), minden érték egyedi; Budapestet is tartalmazza | |
| ***varos*** (*id*, *nev*, *megyeid*, *megyeszekhely*, *megyeijogu*) | |  |
| *id* | A város azonosítója (szám), ez a kulcs; ide soroljuk a fővárosi kerületeket | |
|  | is. |  |
| *nev* | A város neve (szöveg); minden érték egyedi; Budapest esetén a kerület | |
|  | neve |  |
| *megyeid* | A megye azonosítója, ahol a város található (szám); budapesti | |
|  | kerületeknél ez Budapest azonosítója |  |
| *megyeszekhely* | A város megyeszékhely besorolását adja (logikai); ha megyeszékhely, | |
|  | akkor értéke igaz |  |
| *megyeijogu* | A város megyei jogú besorolását adja meg (logikai); ha megyei jogú, | |
|  | akkor értéke igaz |  |
| ***lelekszam*** (*varosid*, *ev*, *no*, *osszesen*) | |  |
| *varosid* | A város azonosítója (szám), ez a kulcs |  |
| *ev* | Az adatrögzítés éve (szám), ez a kulcs |  |
| *no* | Adott városban adott évben a településen élő nők száma (szám) | |
| *osszesen* | Adott városban adott évben a településen élők száma (szám) | |



1. ábra. A táblák adatkapcsolatai és megjelenítése

1. CRUD Alkalmazás
   1. A feladat leírása

**Adatbázis menü**: egy CRUD alkalmazás a választott adatbázissal (5 pont)

**Olvas almenü:** Megjeleníi az adatbázis adatait egy táblázatban. Az adatbázis legalább 3 táblájának adatait használja fel. Ha túl sok adatot tartalmaz az adatbázis, akkor elég csak egy részét megjeleníteni.

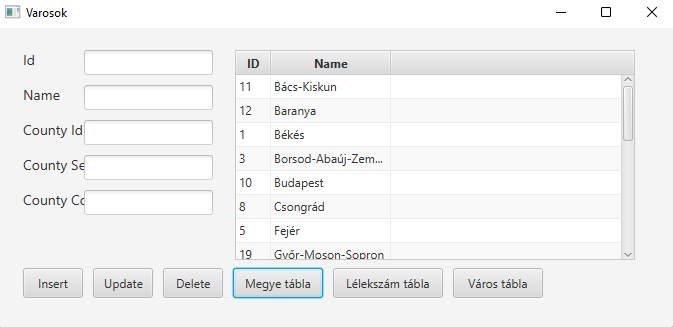
**Olvas2 almenü:** Hasonló, mint az előző, de az oldalon először jelenjen meg egy Űrlap, aminek segítségével tud szűrni az adatbázis adataira. Az űrlapon használja a következő elemeket: szöveges beviteli mező, lenyíló lista, radio gomb, jelölőnégyzet.

**Ír almenü:** Az oldalon jelenjen meg egy űrlap, aminek segítségével új rekordot tud hozzáfűzni az adatbázis egyik táblájához.

**Módosít almenü:** Az oldalon jelenjen meg egy űrlap, aminek segítségével egy már meglévő rekordot tud módosítani az adatbázis egyik táblájánál. A rekord azonosítója lenyíló listából legyen választható.

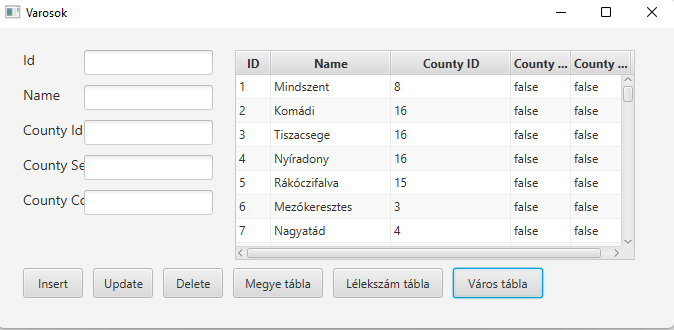
**Töröl almenü:** Az egyik tábla egy rekordjának törléséhez. A rekord azonosítója lenyíló listából legyen választható.

* 1. Látványtervek a projekthez



A képen asztal látható

Automatikusan generált leírás



1. RestFul kliens Alkalmazás
   1. A feladat leírása

**Rest1 menü:** Készítsen egy Restful klienst az eladáson bemutatott (5 pont)

https://gorest.co.in Rest szerverhez. A Create, Read, Update, Delete almenükben valósítsa meg a hozzájuk tartozó műveletet.

* 1. A kód dokumentálva

import javax.net.ssl.HttpsURLConnection;  
import java.io.\*;  
import java.net.URL;  
  
public class RestKliens {  
 static String *token* = "526b08bfb1b072683fdf288a758e007e6c5155e694d4aae17f4da50a54f33e15";  
 static HttpsURLConnection *connection*;  
 static void segéd1(){  
 // Setting Header Parameters  
 *connection*.setRequestProperty("Content-Type", "application/json");  
 *connection*.setRequestProperty("Authorization", "Bearer " + *token*);  
 *connection*.setUseCaches(false);  
 *connection*.setDoOutput(true);  
 }  
 static void segéd2(String params) throws IOException {  
 BufferedWriter wr = new BufferedWriter(new OutputStreamWriter(*connection*.getOutputStream(), "UTF-8"));  
 wr.write(params);  
 wr.close();  
 *connection*.connect();  
 }  
 static void segéd3(int code) throws IOException {  
 int statusCode = *connection*.getResponseCode(); // Getting response code  
 System.*out*.println("statusCode: "+statusCode);  
 if (statusCode == code) { // If responseCode is code, data fetch successful  
 BufferedReader in = new BufferedReader(new InputStreamReader(*connection*.getInputStream()));  
 StringBuffer jsonResponseData = new StringBuffer();  
 String readLine = null;  
 while ((readLine = in.readLine()) != null) // Append response line by line  
 jsonResponseData.append(readLine);  
 in.close();  
 System.*out*.println("List of users: " + jsonResponseData.toString()); // Print result in string format  
 } else {  
 System.*out*.println("Hiba!!!");  
 }  
 *connection*.disconnect();  
 }  
 static void GET(String ID) throws IOException { // Get a list of users  
 System.out.println("\nGET...");  
 String url = "https://gorest.co.in/public/v1/users";  
 if(ID!=null)  
 url=url+"/"+ID;  
 URL usersUrl = new URL(url); // Url for making GET request  
 connection = (HttpsURLConnection) usersUrl.openConnection();  
 connection.setRequestMethod("GET"); // Set request method  
 if(ID!=null)  
 connection.setRequestProperty("Authorization", "Bearer " + token);  
 segéd3(HttpsURLConnection.HTTP\_OK);  
 }  
  
 static void POST(String name, String gender, String email, String status) throws IOException {  
 System.out.println("\nPOST...");  
 URL postUrl = new URL("https://gorest.co.in/public/v1/users"); // Url for making POST request  
 connection = (HttpsURLConnection) postUrl.openConnection();  
 connection.setRequestMethod("POST"); // Set POST as request method  
 segéd1();  
 // Adding Body payload for POST request  
 String params = "{\"name\":\""+name+"\", \"gender\":\""+gender+"\", \"email\":\""+email+"\", \"status\":\""+status+"\"}";  
 segéd2(params);  
 segéd3(HttpsURLConnection.HTTP\_CREATED);  
 }  
  
 /\*  
 Gyakorló feladat:  
 Itt a PUT metódusnál mind a 4 adatot módosítjuk.  
 Átalakítani a PUT metódust úgy, hogy 1,2,3,4 módosítandó adat esetén is jól működjön.  
 \*/  
 static void PUT(String cityId, String cityName, Integer countyId, String countySeat, String countyCouncil) throws IOException {  
 System.out.println("\nPUT...");  
 String url = "https://gorest.co.in/public/v1/users"+"/"+cityId;  
 URL postUrl = new URL(url); // Url for making PUT request  
 connection = (HttpsURLConnection) postUrl.openConnection();  
 connection.setRequestMethod("PUT"); // Set PUT as request method  
 segéd1();  
 String params = "{\"cityName\":\""+cityName+"\", \"countyId\":\""+countyId+"\", \"countySeat\":\""+countySeat+"\", \"countyCouncil\":\""+countyCouncil+"\"}"; // Adding Body payload for POST request  
 segéd2(params);  
 segéd3(HttpsURLConnection.HTTP\_OK);  
 }  
  
  
 static void DELETE(String ID) throws IOException {  
 System.out.println("\nDELETE...");  
 String url = "https://gorest.co.in/public/v1/users"+"/"+ID;  
 URL postUrl = new URL(url); // Url for making PUT request  
 connection = (HttpsURLConnection) postUrl.openConnection();  
 connection.setRequestMethod("DELETE"); // Set DELETE as request method  
 segéd1();  
 segéd3(HttpsURLConnection.HTTP\_NO\_CONTENT);  
 }  
  
 public static void main(String[] args) throws IOException {  
 GET(null);  
 POST("Izsák", "1", "0","0");  
 String ID="3399";  
 GET(ID);  
 PUT(ID,"Dabas", 2, "0","0");  
 GET(ID);  
 DELETE(ID);  
 GET(ID);  
 GET(null);  
 }  
}

1. RestFul szerver Alkalmazás
   1. A feladat leírása

Készítsen a Microsoft Azure felhőben egy **ReatFul szerver** szolgáltatást a választott **(5 pont)**  
adatbázis egyik táblája alapján.

**Rest2 menü:** Készítsen egy Restful klienst a Microsoft Azure felhőben létrehozott Rest szerverhez. A Create, Read, Update, Delete almenükben valósítsa meg a hozzájuk tartozó műveletet.

1. SOAP kliens Alkalmazás
   1. A feladat leírása

**SoapKliens menü:** Készítsen egy SOAP klienst a Magyar Nemzeti Bank **(5 pont)**  
webszolgáltatásához.   
**Letöltés almenü**: Letölti az összes adatot helyi adatbázisba.  
**Letöltés2 almenü**: Az oldalon megjelenik egy űrlap (beviteli mezők, lenyíló lista, radio gomb, jelölő négyzet). Letölti a kiválasztott adatokat helyi adatbázisba és a Microsoft Azure felhő adatbázisába is.  
**Grafikon almenü**: Letöltés2 almenühöz hasonlóan választ egy adatsort és megjeleníti az adatokat grafikonon is.  
Minta: <https://www.mnb.hu/arfolyam-lekerdezes>

1. Adatbányász menü
   1. A feladat leírása

**Adatbányászat menü:** **(5 pont)**  
Válasszanak ki egy fájlt az **Adatbányászat-források** mappából (az órán vizsgát **credit-g.arff** fájlt nem lehet választani). Válasszanak ki ebben egy olyan **attribútumot**, ami nem sok fajta adatot tartalmaz. Általában az utolsó Class nevű attribútum ilyen.   
**Döntési fa almenü**: Készítsenek döntési fával gépi tanuló algoritmus a kiválasztott attribútum előrejelzésére. Eredményül írassák ki a **Döntési fa.txt** fájlba a következő adatokat:   
Tanító és kiértékelő halmaz mérete,   
TP, TN, FP, FN, Correctly Classified Instances, Incorrectly Classified Instances  
A döntési fa kiíratása  
A kiértékelő halmaz minden egyedére a prediktált és a valós érték  
**Több algoritmus almenü**: A választott adathalmazra futtassa le a tanult algoritmusokat. A **Gépi tanulás.txt** fájlba írassa ki mindegyikre:   
TP, TN, FP, FN, Correctly Classified Instances, Incorrectly Classified Instances  
Válassza ki az algoritmusok közül azt, amelyiknek a legjobb lett a Correctly Classified Instances eredménye és írassa ki az algoritmus nevét az alkalmazás ablakába.  
**Több algoritmus almenü-2**: Lenyíló listából lehessen választani a tanult algoritmusok közül. Az algoritmust lefuttatva írassa ki az alkalmazás ablakába a következő adatokat:  
TP, TN, FP, FN, Correctly Classified Instances, Incorrectly Classified Instances

1. Egyéb menü
   1. A feladat leírása

**Páthuzamos almenü:** Mutassa be a párhuzamos programvégrehajtást egy oldalon. pl. egy gombra való kattintás után egy Label-ben 1 másodpercenként, a másik Labelbe 2 másodpercenként jelenjen meg egy változó szöveg.  
**Stream almenü**: Olvassa ki a válsztott adatbázis minden adatát egy Stream-be. Késztsen egy űrlapot (beviteli mezők, lenyíló lista, radio gomb, jelölő négyzet), ahol több szűrőfeltételt is beállíthat az adatokra. Jelenítse meg egy táblázatban a szűrt adatokat.

1. GitHub link és elérhetőség

https://github.com/Gergely37/Java\_beadando\_feladat\_EA