Tartalomjegyzék

[1. Feldolgozandó adatbázisok 4](#_Toc121301242)

[2. A cég bemutatása a főoldalon 5](#_Toc121301243)

[2.1. A feladat leírása 5](#_Toc121301244)

[2.2. Látványtervek a projekthez 5](#_Toc121301245)

[3. Legyen Regisztráció, Bejelentkezési lehetőség 6](#_Toc121301246)

[3.1. A feladat leírása 6](#_Toc121301247)

[3.2. A kód dokumentálva 6](#_Toc121301248)

[4. Legalább 3 felhasználói szerepet különböztessen meg 7](#_Toc121301249)

[4.1. A feladat leírása 7](#_Toc121301250)

[5. Legyen egy oldal, ahol a választott adatbázisból jelenít meg adatokat 8](#_Toc121301251)

[5.1. A feladat leírása 8](#_Toc121301252)

[6. Az egyik oldalon legyen egy kapcsolat űrlap 9](#_Toc121301253)

[6.1. A feladat leírása 9](#_Toc121301254)

[7. Űrlap megtekintése 10](#_Toc121301255)

[7.1. A feladat leírása 10](#_Toc121301256)

[8. Valósítson meg az alkalmazásban egy RESTful API-t 11](#_Toc121301257)

[8.1. A feladat leírása 11](#_Toc121301258)

[9. Használják a GitHub (github.com) verziókövető rendszert 12](#_Toc121301259)

[9.1. A feladat leírása 12](#_Toc121301260)

[10. Alkalmazását töltse fel és valósítsa meg Interneten a Microsoft Azure felhőben 13](#_Toc121301261)

[10.1. A feladat leírása 13](#_Toc121301262)

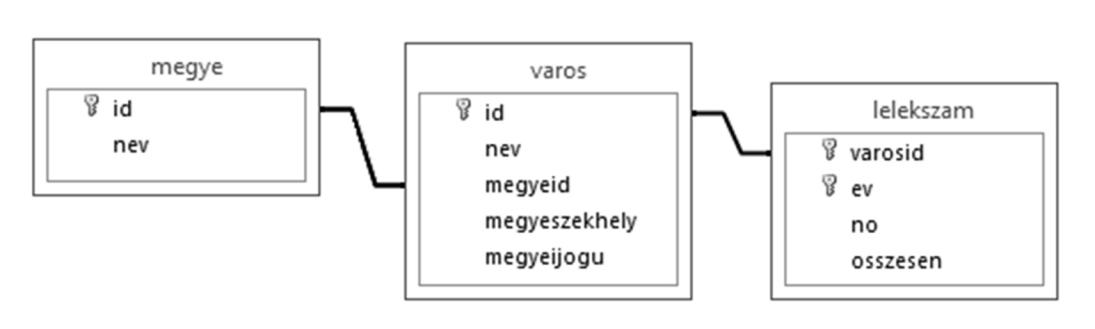
[11. GitHub link és elérhetőség 14](#_Toc121301263)

[12. Kódrészletek 15](#_Toc121301264)

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. A cég bemutatása a főoldalon
   1. A feladat leírása

Az első oldalon mutassa be a céget egy látványos weboldalon. A főoldalon egy leírást találhatunk. Ahonnan a további oldalakra tudunk továbblépni. A feladat megoldása során egy fiktív cég oldalát készítjük el, amely tartalmazza a települések adatait.

* 1. Látványtervek a projekthez

A képen szöveg látható

Automatikusan generált leírás

1. Legyen Regisztráció, Bejelentkezési lehetőség
   1. A feladat leírása

- A „Belépés” menüpont akkor látható, ha nincs bejelentkezve a felhasználó.

- A „Kilépés” menüpont akkor látható, ha be van jelentkezve a felhasználó.

- A rendszer fejlécen jelenítse meg a bejelentkezett felhasználót, ha be van lépve

A program három különböző felhasználó típust különböztet meg. Admin, Látogató és Felhasználó. A bejelentkezési felületen egy legördülő listából lehet megkülönböztetni a szerepköröket.

A képen szöveg látható

Automatikusan generált leírás

* 1. A bejelentkezési felület kódja dokumentálva

<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <meta http-equiv="X-UA-Compatible" content="IE=edge">  
 <meta name="viewport" content="width=device-width, initial-scale=1.0">  
 <title>Bejelentkezés</title>  
</head>  
<body>  
   
</body>  
</html>

1. Legalább 3 felhasználói szerepet különböztessen meg
   1. A feladat leírása

Legalább 3 felhasználói szerepet különböztessen meg:

Admin, User, Látogató

A menüpontok megjelenése és az oldalak elérhetősége változik attól függően,

hogy melyik felhasználó használja az oldalt. (pl. admin oldal)

1. Legyen egy oldal, ahol a választott adatbázisból jelenít meg adatokat
   1. A feladat leírása

Legyen egy oldal, ahol a választott adatbázisból jelenít meg adatokat

Ehhez 3 tábla adatait használja fel az adatbázisból.

A képen szöveg látható

Automatikusan generált leírás

A főoldalon az Adatbázis menüpont alatt elérhető a lokális adatbázisból beolvasott adatok, amely ott megtekinthető.

1. Az egyik oldalon legyen egy kapcsolat űrlap
   1. A feladat leírása

Az egyik oldalon legyen egy kapcsolat űrlap, amelynek segítségével üzenetet

lehet küldeni az oldal tulajdonosa számára. Ellenőrizze megfelelően az űrlap

helyes kitöltését szerver oldali validációval is. Az elküldött Űrlap adatokat

mentse le az adatbázisba.

1. Űrlap megtekintése
   1. A feladat leírása

Tegye lehetővé megtekinteni egy hatodik oldalon táblázatban az előző pontban (2 pont)

elküldött üzeneteket az adatbázisból fordított időrend szerint (a legfrissebb

legyen elől). Írja minden üzenethez a küldés idejét és az üzenetküldő nevét.

Ha nem bejelentkezett felhasználó írta, akkor: "Vendég".

* 1. A feladat leírásaA feladathoz készített HTML fájl

1. Valósítson meg az alkalmazásban egy RESTful API-t
   1. A feladat leírása

Valósítson meg az alkalmazásban egy RESTful API-t.

Tesztelje az API funkcióit cURL-el és Postman-al is. Mindkét tesztről tegyen

képernyőképeket a dokumentációba

1. Használják a GitHub (github.com) verziókövető rendszert
   1. A feladat leírása

Használják a GitHub (github.com) verziókövető rendszert.

(Kötelező elem! A forrás ez alapján lesz javítva)

A GitHub-on a projektmunka módszert alkalmazzák: látszódjék, hogy a

csoport tagjai melyik részt készítették el és kb. fele-fele arányban járuljanak

hozzá a projekthez. Ne csak a kész alkalmazást töltsék fel egy lépésben,

hanem a részállapotokat is még legalább 5 lépésben személyenként.

A képen szöveg látható

Automatikusan generált leírás

1. Alkalmazását töltse fel és valósítsa meg Interneten a Microsoft Azure felhőben
   1. A feladat leírása

Alkalmazását töltse fel és valósítsa meg Interneten a Microsoft Azure felhőben.

(Kötelező elem! A működés ez alapján lesz javítva)

1. GitHub link és elérhetőség

<https://github.com/Gergely37/Java_beadando_feladat_GY>

1. Kódrészletek

//  
// Source code recreated from a .class file by IntelliJ IDEA  
// (powered by FernFlower decompiler)  
//  
  
package org.springframework.jdbc.datasource;  
  
import java.sql.Connection;  
import java.sql.SQLException;  
import org.springframework.core.NamedThreadLocal;  
import org.springframework.lang.Nullable;  
import org.springframework.util.Assert;  
import org.springframework.util.StringUtils;  
  
public class UserCredentialsDataSourceAdapter extends DelegatingDataSource {  
 @Nullable  
 private String username;  
 @Nullable  
 private String password;  
 @Nullable  
 private String catalog;  
 @Nullable  
 private String schema;  
 private final ThreadLocal<JdbcUserCredentials> threadBoundCredentials = new NamedThreadLocal("Current JDBC user credentials");  
  
 public UserCredentialsDataSourceAdapter() {  
 }  
  
 public void setUsername(String username) {  
 this.username = username;  
 }  
  
 public void setPassword(String password) {  
 this.password = password;  
 }  
  
 public void setCatalog(String catalog) {  
 this.catalog = catalog;  
 }  
  
 public void setSchema(String schema) {  
 this.schema = schema;  
 }  
  
 public void setCredentialsForCurrentThread(String username, String password) {  
 this.threadBoundCredentials.set(new JdbcUserCredentials(username, password));  
 }  
  
 public void removeCredentialsFromCurrentThread() {  
 this.threadBoundCredentials.remove();  
 }  
  
 public Connection getConnection() throws SQLException {  
 JdbcUserCredentials threadCredentials = (JdbcUserCredentials)this.threadBoundCredentials.get();  
 Connection con = threadCredentials != null ? this.doGetConnection(threadCredentials.username, threadCredentials.password) : this.doGetConnection(this.username, this.password);  
 if (this.catalog != null) {  
 con.setCatalog(this.catalog);  
 }  
  
 if (this.schema != null) {  
 con.setSchema(this.schema);  
 }  
  
 return con;  
 }  
  
 public Connection getConnection(String username, String password) throws SQLException {  
 return this.doGetConnection(username, password);  
 }  
  
 protected Connection doGetConnection(@Nullable String username, @Nullable String password) throws SQLException {  
 Assert.*state*(this.getTargetDataSource() != null, "'targetDataSource' is required");  
 return StringUtils.*hasLength*(username) ? this.getTargetDataSource().getConnection(username, password) : this.getTargetDataSource().getConnection();  
 }  
  
 private static final class JdbcUserCredentials {  
 public final String username;  
 public final String password;  
  
 public JdbcUserCredentials(String username, String password) {  
 this.username = username;  
 this.password = password;  
 }  
  
 public String toString() {  
 return "JdbcUserCredentials[username='" + this.username + "',password='" + this.password + "']";  
 }  
 }  
}