	Backend	
1	Töltse ki megfelelően az adatbázishoz való csatlakozás adatait a backend.js fájlban.	SQL importálása Web backend betöltése Csomagok frissítés VS Code-ban npm i npm audit fix npm i nodemon package.json módosítása "scripts" résznél "start": "nodemon backend.js" indítás: npm start package.json -> "start": "nodemon backend.js" host: 'localhost', user: 'root', password: '', database: 'kelettravel2024'
2	Készítsen végpontot az adatbázis kapcsolatfelvetel táblájának bővítésére,	app.use(express.urlencoded({ extended: true })); PHPmyAdmin, tábla beszúrás: SQL előnézete, copy
3	a végpont kezelje a beérkező adatokat,	

4	az összes bemenő adatot jól kezelje,	"app.post('/kapcsolatok', (req, res) => {
		const { nev, email, telefon, megjegyzes} = req.body;
		//console.log(nev, email, telefon, megjegyzes);
		
		kapcsolat();
		connection.query('INSERT INTO `kapcsolatfelvetel` (id, nev, email, telefon, megjegyzes) VALUES (NULL, ?, ?, ?, ?)',
		[nev, email, telefon, megjegyzes], (err, result) => {
		if (err) {
		return res.status(500).json('Hiba');
		}
		
		res.status(200).json('Sikeres felvitel');
		connection.end()
		});
		});"

5	és kimenete a Hiba vagy Sikeres felvitel szöveg legyen.	

6	Új végpont létrehozása a "celok" táblán belüli kereséshez, (megadott kulcsszó alapján keressen az utazási célok nevében, részkifejezésre, frontenden, Weblapon van keresés úticélra)	app.post('/celok', (req, res) => {
		const searching = req.body.searching;
		//console.log(searching);
		
		kapcsolat();
		connection.query('SELECT * FROM celok WHERE celok_nev LIKE ?', [`%${searching}%`], (err, result) => {
		if (err) {
		return res.status(500).json('Hiba');
		}

7	a keresés végpont kezelje a beérkező adatokat,	
8	jól keressen csak egy szórészletre,	

9	és kimenete a talált adatsor legyen.
10	Készítsen végpontot a kapcsolatfelvetel tábla egy rekordjának törlésére,
11	bemenő adat a törlendő rekord kapcsolat id-ja legyen,
12	a törlés parancsát jól alkalmazza,
13	és kimenete a Hiba vagy Sikeres törlés szöveg legyen.
14	Tesztelje Postmanben a kapcsolatfelvitel funkciót, készítsen az eredményességről képernyőképet,
15	majd tesztelje a törlés funkciót, eredményességről készüljön képernyőkép!
+	**VÁRHATÓ:** WEB Backend lekérdezés 2 táblából. GET pl.: SELECT * FROM celok JOIN orszag ON celok.celok_orszag = orszag.orszag_id;
	Konzolos C#

```
  }

  res.status(200).json(result);
  connection.end();
 });
});
```

```
app.delete('/kapcsolatok/:id', (req, res) => {
 const id = req.params.id;
 //console.log(id);

 kapcsolat();

 connection.query('DELETE FROM kapcsolatfelvetel WHERE id = ?', [id], (err, result) => {
  if (err) {
   return res.status(500).json('Hiba');
  }

  res.status(200).json('Sikeres törlés');
  connection.end();
 })
});
```

```
http://localhost:3001/kapcsolatok
Headers: Content-Type, application/json
Body: x-www-form-urlencoded
nev: Teszt Elek
email: teszt@teszt.hu
telefon: 123456789
megjegyzes: minden mehet
```

16	Osztály létrehozása a "célok" tábla összes mezőjére (karakterhelyesen)	New projekt Console App (.Net Framework) Place solution and project in the same directory Project jobb egér gomb Add item/Class/Celok.cs public class Celok prop tab tab public int celok_id { get; set; } public string celok_nev { get; set; } public string celok_kep { get; set; } public string celok_kultura_honap { get; set; } public int celok_orszag { get; set; }

17	Osztály létrehozása a "kapcsolatfelvetel" tábla összes mezőjére (karakterhelyesen)	public class Kapcsolatfelvetel
		{
		public int id { get; set; }
		public string nev { get; set; }
		public string email { get; set; }
		public string telefon { get; set; }
		public string megjegyzes { get; set; }
		}

18	"Celok" osztály bővítése új metódussal (KetSzo néven), melynek visszatérési értéke igaz logikai érték, ha a cél neve két szóból áll, egyébként hamis	public bool KetSzo()
		{
		string[] tmp = celok_nev.Split();
		if (tmp.Length > 1)
		{
		return true;
		}
		else
		{
		return false;
		}
		VAGY
		
		public bool KetSzo()
		{
		return celok_nev.Contains(" ");
		}

19	"Kapcsolat" osztály bővítése új tulajdonsággal(property) (Hianyos néven), mely értéke igaz logikai érték, ha a név, email vagy telefon bármelyike üres, egyébként hamis	public bool Hianyos
		{
		get
		{
		if (nev == "" || email == "" || telefon == "")
		{
		return true;
		}
		else
		{
		return false;
		}
		}
		}

20	Célok letárolása az adatbázis táblájából, listába, felhasználva a "Celok" osztályt (NewtonSoft.Json és NetworkHelper dll-ek beimportálása)	Jobb oldalon, References/Add reference
		két dll-t adjuk hozzá
		
		Program.cs
		using NetworkHelper; beszúrása
		
		namespace KeletTravelKonzolos
		{
		class Program
		{
		static string host = "http://localhost:3000";
		static List<Celok> celokLista = Backend.GET($"{host}/celok").Send().ToList<Celok>();
		static void Main(string[] args)
		{
		
		Console.ReadKey();
		}
		}
		}

21	Úticélok darabszámának kiírása (1.feladat)	// 1. feladat Console.WriteLine($"1. feladat: Az elérhető célok száma: {celokLista.Count()}");
22	Azon úticélok nevének kiíratása, amelyek nem kétszavasok, KetSzo függvény használata. (2.feladat)	// 2. feladat Console.WriteLine("2. feladat: Egyszavas célok:"); celokLista.Where(x => x.KetSzo() == false).ToList().ForEach(x => Console.WriteLine(x.celok_nev));
23	Egy keresendő szó bekérése (3.feladat),	// 3. feladat Console.Write("3. feladat: Adj meg egy keresendő szót: "); string keresendo = Console.ReadLine();

24	majd a keresett szó megkeresése az úti cél nevében, cél nevének és a hónap kiírása, amikor a legtöbb kultúrális esemény zajlik ,	
25	találatok számának is kiírása képernyőre,	
26	a keresés eredménye fájlba is legyen kiírva, fájl neve a beért szó pl. sziget.txt fájlba,	
27	ugyanaz legyen fájlban, mint a képernyőn: a talált úticélok neve (hónap) és alatta a találatok száma.	
28	A kultúra hónapja szerinti csoportosításban hány (darab) úti cél van (4.feladat),	
29	rendezés darabszám szerint csökkenően,	
30	majd hónap és darabszám kiíratása képernyőre.	
	Grafikus C#	
31	A projecten belül új WPF megnyitása, és Consolos program elérése, minden class nyilvánossá tétele (NewtonSoft.Json és NetworkHelper dll-ek beimportálása)	Solution jobb egér gomb: Add new project: WPF App (.Net framework) Project/References/ Add references: 2 dll using NetworkHelper; Project/Add Existing item.../ megkeressük a Konzolosból az osztályokat (kapcsolatfelvétel és célok.cs) és Add As Link
32	Tabcontrol 1. fülén belül Listbox-ban, jelenjenek meg az úticélok nevei	
33	hívja meg hozzá a szükséges Backend végpontot,	

```
        // 3. feladat
        Console.Write("3. feladat: Adj meg egy keresendő szót: ");
        string keresendo = Console.ReadLine();

        var talalat = celokLista.Where(x => x.celok_nev.Contains(keresendo)).Select(x => $"{x.celok_nev} {x.
celok_kultura_honap}").ToList();
        Console.WriteLine("Találatok:");
        talalat.ForEach(x => Console.WriteLine(x));
        Console.WriteLine($"Találatok száma: {talalat.Count()}");
        // fájlba írás
        var tartalom = talalat.Prepend("Talalatok: ").Append($"Találatok száma: {talalat.Count()}");
        File.WriteAllLines($"{keresendo}.txt", tartalom);



        // 4. feladat
        Console.WriteLine("4. feladat");
        celokLista.GroupBy(x => x.celok_kultura_honap).Select(x => new
        {
           honap = x.Key,
           db = x.Count()
        }).OrderByDescending(x => x.db).ToList().ForEach(x => Console.WriteLine($"{x.honap}: {x.db}"));

        Console.ReadKey();
```

```
using KeletTravelKonzolos;
  public partial class MainWindow : Window
  {
     string host = "http://localhost:3000";
     public MainWindow()
     {
        InitializeComponent();
        lbCelok.ItemsSource = Backend.GET($"{host}/celok").Send().ToList<Celok>();
        lbCelok.DisplayMemberPath = "celok_nev";
     }
  }
```

34	ha a listbox elemeire kattintunk,	private void lbCelok_SelectionChanged(object sender, SelectionChangedEventArgs e)
		{
		Celok cel = lbCelok.SelectedItem as Celok;
		lblCelNev.Content = cel.celok_nev;
		lblCelHonap.Content = cel.celok_kultura_honap;
		}

35	mellette jelenjen meg az úti cél neve és látogatáshoz ideális hónap.	
36	A Tabcontrol 2. fülén készítsen kapcsolatfelvételhez űrlapot,	private void btnFelvetel_Click(object sender, RoutedEventArgs e) {

37	gombnyomásra vigye fel az adatokat az adatbázis kapcsolatfelvetel táblájába	Kapcsolatfelvetel adatok = new Kapcsolatfelvetel()
		{
		nev = tbNev.Text,
		email = tbEmail.Text,
		telefon = tbTelefon.Text,
		megjegyzes = tbMegjegyzes.Text
		};

38	a felvitelt, a szükséges Backend végpont hívásával oldja meg,	
39	minden szükséges adatot küldjön át a végpontnak,	

40	ha a név mező üres, ne vigyen fel adatot, hibaüzenetet adjon.	if (adatok.Hianyos)
		{
		MessageBox.Show("Minden mező kitöltése kötlező!");
		}
		else
		{
		string uzenet = Backend.POST($"{host}/kapcsolatok").Body(adatok).Send().Message;
		MessageBox.Show(uzenet);
		}

	HTML+CSS+Bootstrap	
41	A bg-secondary osztályok cseréje bg-primary-re	CTRL+H replace: bg-secondary -> bg-primary
42	Szolgáltalásokon belül 2 oszlopos helyett 3 oszolopos elrendezés, a 3. elem a szolg3.jpg képpel,	"col-md-6" -> "col-md-4"
43	melyen kattintva modális ablak jelenjen meg, a képpel és seged.txt-ben található szöveggel, Utas biztosítás címmel.	
44	Új menüpont, Keresés menüpont, menüben való kattintásra gördítsen le a Weblapon,	<li class="nav-item">Keresés
45	Keresés szekció létrehozása (Úti célok szekció mintájára) ,	placeholder csere
46	benne beviteli mező ('keresendo' azonosítóval) és gomb (hasonlóan, mint lent Üzenet küldése gomb), kattintásra 'kereses' függvény,	id="kereses" id="keresendo" id="searchingButton" type="button" onclick="kereses()"
47	alatta új keret, teljes szélességgel, "talalat" azonosítóval (találatoknak) .	
48	Üzenet küldése gombra "kapcsolat" függvény meghívása, alatta keret, "siker" azonosítóval,	type="button" onclick="kapcsolat()" </form> alá külön egy div

49	mely 20px szöveg méretű, félkövér, felső margója 10px-es (stíluslapon).	stilus.css fájlban:
		#siker {
		font-size: 20px;
		font-weight: bold;
		margin-top: 10px;
		}

50	Alul, Kapcsolat részben 4. elem az "Úti cél, vagy egyéb megjegyzés" 2 soros beviteli szöveg legyen, "message" azonosítóval.	<textarea name="mesage" id="message" cols="63" rows="2" placeholder="Úti cél, vagy egyéb megjegyzés" ></textarea>
	Frontend	
51	Oldja meg, hogy a kapcsolatfelvétel szekcióban "Úti cél vagy egyéb megjegyzés" részbe írt szöveg bekerüljön az adatbázisba, ehhez a "kapcsolat" függvényt módosítsa az alap.js-ben,	megjegyzes:document.getElementById("message").value

52	továbbá, a fetch parancsban a method és body mezőt megfelelően állítsa be,	fetch("http://localhost:3000/kapcsolatok", {
		method: "POST",
		body: JSON.stringify(bemenet),
		headers: {"Content-type": "application/json; charset=UTF-8"}
		}

53	és a "siker" azonítójú keretbe írja a Backendről vissszatérési értékként megkapott "A felvitel sikerült." szöveget.	document.getElementById("siker").innerHTML = y;
54	Oldja meg, hogy a 'célok' megjelenjenek a Weblap 'Úti célok' szekciójában, a Backend végpont megfelelő hívásával,	index.html -> <body id="page-top" onload="uticelok()">
55	a visszatérő adatokat-úticélokat, json-ban kezelje, rakja keretekbe,	alap.js: async function uticelok() { const res = await fetch("http://localhost:3000/celok"); const celok = await res.json(); console.log(celok); let celokHTML = ""; for (const cel of celok) { celokHTML += ` <div class="col-lg-4 ml-auto"> <img src="kepek/${cel.celok_kep}" alt="${cel.celok_neve}" class="img-fluid"> <p class="alairas">${cel.celok_nev}</p> </div> `; } document.getElementById("celok").innerHTML = celokHTML; }
56	a keretek osztálya "col-lg-4" és a "ml-auto" osztályok legyenek,	
57	a keretbe a képet helyezze el (jó eléréssel, reszponzív képméretezéssel),	
58	a keretbe illesszen bekezdést, mely osztálya "alairas", legyen benne a cél neve	
59	és a célok képei, illetve a célok nevei, a megfelelő helyen jelenjenek meg a Weblapon, "celok" azonosítójú keretben, a minta szerint.	

60	Oldja meg, hogy a Keresés szekcióban lehessen keresni az úti cél nevében, ehhez hozzon létre "kereses" függvényt, mely a keresésre írt Backend végpontot hívja meg,	async function kereses() {
		const searching = document.getElementById('keresendo').value;
		
		const res = await fetch('http://localhost:3000/celok', {
		method: "POST",

61	a "keresendo" azonosítójú beviteli mezőbe írt szöveget átküldi a Backend végpontnak,
62	és a keresés eredményét, számozatlan felsorolásba helyezi,
63	minden listaelemre beállítja a "lista" osztályt, berakja a 100px szélességű képet és
64	a "kephez" osztályú szöveges mezőbe rakja az úti cél nevét,
65	és a keresés eredménye megjelenik a Weblapon a "talalat" azonosítójú keretben, a minta szerint.

```
    headers: { "Content-type": "application/json; charset=UTF-8" },
    body: JSON.stringify({ searching })
  });
  const adatok = await res.json();

  let keresesHTML = "ul";
  for (const adat of adatok) {
    keresesHTML += `
          <li class="lista">
            <img src="kepek/${adat.celok_kep}" alt="${adat.celok_nev}" style="width: 100px;">
            <span class="kephez">${adat.celok_nev}</span>
          </li>
        `;
  }
  keresesHTML += "</ul>";
  document.getElementById("talalat").innerHTML = keresesHTML;
}
```