# 6 – AJAX - Asynchronous JavaScript And XML

AJAX: Asynchronous JavaScript And XML.

https://www.w3schools.com/js/js_ajax_intro.asp
https://en.wikipedia.org/wiki/Ajax_(programming)

AJAX is a developer's dream, because you can:

- Read data from a web server - after the page has loaded
- Update a web page without reloading the page
- Send data to a web server - in the background

A HTML fájlban nem hivatkozunk a PHP fájlra: majd csak a Javascript fájlban fogunk!

- Nincs közvetlen kapcsolat a HTML és a PHP között végig
  A JS kommunikál a HTML-el és a PHP-val is külön-külön
- A **JS irányítóként működik** (az oldalbetöltődése után)
  - A JavaScript-nek manapság egyre nagyobb szerepe van.


## JSON bevezető - Csak olvasmány

**Majd látjuk, hogy az AJAX híváskor az adatokat JSON formában továbbítjuk a felek között {…}.**
A JSON formát nem kell ismernünk, mert a protokoll a háttérben megoldja, így a következő csak olvasmány:

**JSON:**
A JSON (JavaScript Object Notation) egy kis méretű, **szöveg alapú szabvány**, ember által olvasható **adatcserére**. A JavaScript szkriptnyelvből alakult ki egyszerű adatstruktúrák és asszociatív tömbök reprezentálására (a JSON-ban objektum a nevük). A JavaScripttel való kapcsolata ellenére **nyelvfüggetlen**, több nyelvhez is van értelmezője.
A JSON-t **legtöbbször egy szerver és egy kliens számítógép közti adatátvitelre** használják (legtöbbször AJAX technológiával), **az XML egyik alternatívájaként**. Általánosságban **strukturált adatok tárolására, továbbítására szolgál**.

**egy JSON Példa:**
A következő példa egy személyt leíró objektum JSON-os reprezentációja. Az objektum a vezetéknév és a keresztnév számára string típusú mezőket hagy, az életkornak számot, tartalmaz egy címet reprezentáló asszociatív tömböt (objektum) és egy listát (tömböt), amely a telefonszám-objektumokat tartalmazza.

```
{
    "vezetekNev": "Kovács",
    "keresztNev": "János",
    "kor": 25,
    "cim":
    {
       "utcaHazszam": "2. utca 21.",
       "varos": "New York",
```

```
        "allam": "NY",
        "iranyitoSzam": "10021"
      },
      "telefonSzam":
      [
        {
          "tipus": "otthoni",
          "szam": "212 555-1234"
        },
        {
          "tipus": "fax",
          "szam": "646 555-4567"
        }
      ]
}
```


## Az AJAX Feladat - CRUD

A következő oldalon található egy web szolgáltatás (API) a következő tulajdonságokkal:

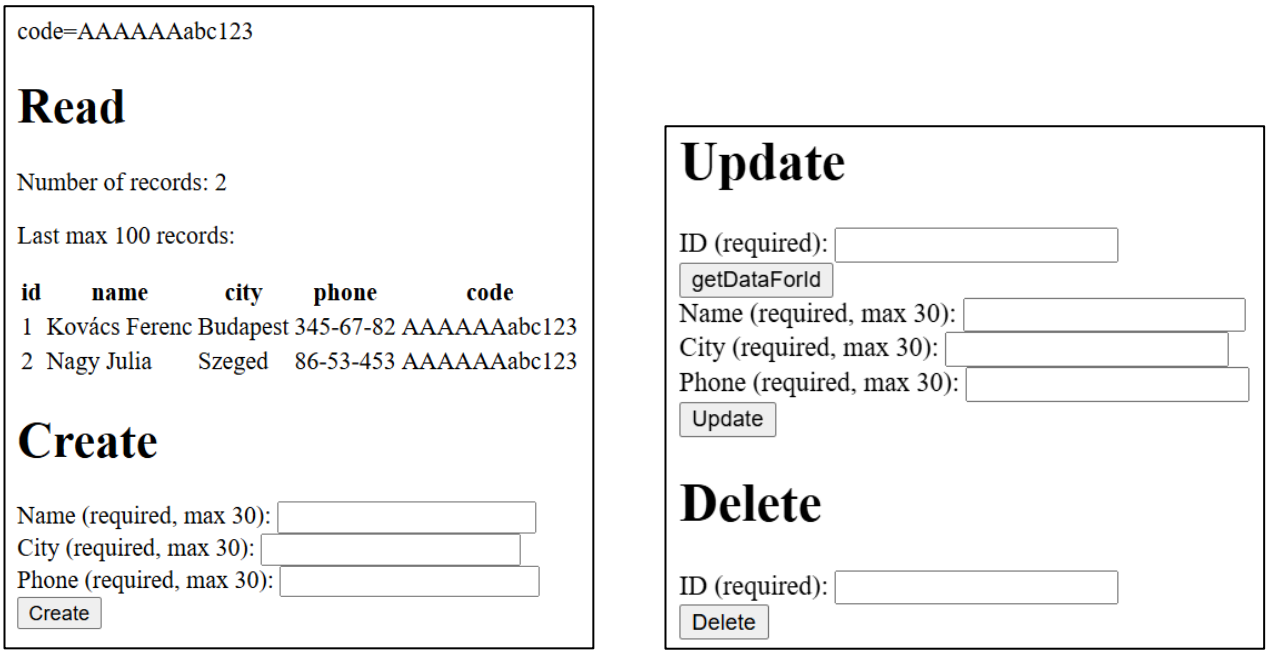
http://gamf.nhely.hu/ajax1/

- Az API a háttérben lévő adatbázisból olvas/ír.
- CRUD műveleteket valósít meg: Create, Read, Update, Delete
- POST kéréseket fogad paraméterekkel
- A tábla minden adata String típusú (VARCHAR).
- Minden kéréshez el kell küldeni a code paramétert, ami azonosítja a felhasználót
  A code paraméter formája legyen: **AAAAAAefg456**
  ahol AAAAAA a hallgató Neptun kód, efg456: pár karakteres kód, amit a hallgató talál ki és ír a Neptun kód után, így a kódot csak ő ismeri. A következő kéréseknél, csak a hallgató kódjához tartozó sorokhoz enged hozzáférést
- op=read és code paraméterre:
  Lekérdezi az adatbázistábla adatait.
  Visszadja a következő adatokat egy tömbben JSON formában:
  "rowCount" a rekordok száma
  "maxNum" maximálisan átadott rekordok száma, ami 100
  "list" => array() a rekordok egy listában kulcs-érték párokban:
  kulcsok: id, name, city, phone, code
- op= create és name, city, phone, code paraméterekre:
  A name, city, phone, code paraméterek első 100 karakteréből képzett rekordot beírja a táblába.
  Visszaadja, hogy az utasítás hány rekordra volt hatással (0 vagy 1)
- op= update és id, name, city, phone, code paraméterekre:
  Módosítja az adott id-hez és code-hoz tartozó rekordot a magadott új értékekkel
  Visszaadja, hogy az utasítás hány rekordra volt hatással (0 vagy 1)
- op= delete és id, code paraméterekre:
  Törli az adott id-hez és code-hoz tartozó rekordot
  Visszaadja, hogy az utasítás hány rekordra volt hatással (0 vagy 1)

Készítsen AJAX alkalmazást az API-t felhasználva és jelenítse meg az adatokat weboldalon.
A következő fájlokat kell elkészíteni: **ajax.html, ajax.js**

A create és update funkcióknál a beviteli mező nem lehetnek üresek és max 30 karakter értéket tartalmazhatnak. Ezeket ellenőrizze a JS fájlban (validation).
Az Update résznél legyen egy beviteli mező, amibe beírjuk az ID-t, és egy getDataForId gomb, amivel először kiolvastatjuk az adatokat beviteli mezőkbe, ahol módosíthatjuk azokat.
A Create, Update, Delete műveletek sikerességéről adjunk visszajelzést a weboldalon.

**Az oldal kinézete:**




## Az API web-szolgáltatás tesztelése cURL-el – 5 perc – Kidolgozás előtt ellenőrizzük, hogy működik-e a web-szolgáltatás

Parancssorban:
Előtte egy sorba kell rendezni!

**A, READ**
curl -X POST "http://gamf.nhely.hu/ajax1/" -H "Content-Type: application/x-www-form-urlencoded" -d "code=AAAAAAabc123&op=read"

```
curl -X POST "http://gamf.nhely.hu/ajax1/" -H "Content-Type: application/x-www-form-urlencoded" -d "code=AAAAAAabc123&op=read"
```

**Válasz:**
{"list":[{"id":1,"name":"Kov\u00e1cs Ferenc","city":"Budapest","phone":"345-67-82","code":"AAAAAAabc123"},{"id":2,"name":"Nagy Julia","city":"Szeged","phone":"86-53-453","code":"AAAAAAabc123"}],"rowCount":2,"maxNum":100}

**B, CREATE**
curl -X POST "http://gamf.nhely.hu/ajax1/" -H "Content-Type: application/x-www-form-urlencoded" -d "code=AAAAAAabc123&op=create&name=Nagy Ilona&city=Debrecen&phone=70/345-12-46"

**Válasz:**
1

Ha újra lefuttatjuk a Read-et, akkor látjuk, hogy hozzáadta az új rekordot. Ez látszik az AJAX alkalmazásunkon is.

| id | name | city | phone | code |
|---|---|---|---|---|
| 1 | Kovács Ferenc | Budapest | 345-67-82 | AAAAAAabc123 |
| 2 | Nagy Julia | Szeged | 86-53-453 | AAAAAAabc123 |
| 33 | Nagy Ilona | Debrecen | 70/345-12-46 | AAAAAAabc123 |

Most id=33-as volt az új rekord.

**C, UPDATE**
Írjuk át az ID=33-as rekord City mezőjét Miskolc-ra:

curl -X POST "http://gamf.nhely.hu/ajax1/" -H "Content-Type: application/x-www-form-urlencoded" -d "code=AAAAAAabc123&op=update&id=33&name=Nagy Ilona&city=Miskolc&phone=70/345-12-46"

**Válasz:**
1

Ha újra lefuttatjuk a Read-et, akkor látjuk a változást. Ez látszik az AJAX alkalmazásunkon is.

**D, DELETE**

Töröljük az ID=33-as rekordot:

curl -X POST "http://gamf.nhely.hu/ajax1/" -H "Content-Type: application/x-www-form-urlencoded" -d "code=AAAAAAabc123&**op=delete**&id=33"

**Válasz:**
1

Ha újra lefuttatjuk a Read-et, akkor látjuk a változást. Ez látszik az AJAX alkalmazásunkon is.

## Megoldás

A következő 2 megoldásnál a HTML fájl megegyezik:

**ajax.html (nincs benne újdonság)**

```
<!DOCTYPE HTML>
<html>
 <head>
 <meta charset="utf-8">
 <script type="text/javascript" src = "ajax.js"></script>
 <title>Ajax</title>
 </head>
 <body>
  <div id = 'code'></div>
  <div id = 'readDiv'></div>
  <div id = 'createDiv'>
   <h1>Create</h1>
   Name (required, max 30): <input id="name1" type="text"></input><br>
   City (required, max 30): <input id="city1" type="text"></input><br>
```

```
    Phone (required, max 30): <input id="phone1" type="text"></input><br>
    <button type="submit" onclick="create();">Create</button><br>
    <div id = 'createResult'></div>
   </div>
   <div id = 'updateDiv'>
    <h1>Update</h1>
    ID (required): <input id="idUpd" type="text"></input><br>
    <button type="submit" onclick="getDataForId();">getDataForId</button><br>
    Name (required, max 30): <input id="name2" type="text"></input><br>
    City (required, max 30): <input id="city2" type="text"></input><br>
    Phone (required, max 30): <input id="phone2" type="text"></input><br>
    <button type="submit" onclick="update();">Update</button><br>
    <div id = 'updateResult'></div>
   </div>
   <div id = 'deleteDiv'>
    <h1>Delete</h1>
    ID (required): <input id="idDel" type="text"></input><br>
    <button type="submit" onclick="deleteF();">Delete</button><br>
    <div id = 'deleteResult'></div>
   </div>
  </body>
</html>
```

**Két megoldást nézünk meg:**

- újabb Fetch API-val (2015-től)
- régebbi XMLHttpRequest (XHR) JavaScript osztállyal (1999-től) – csak olvasmány

## Megoldás – 1 – az újabb Fetch API-val

**async, await, fetch**

**async:** async makes a function return a **Promise**

The keyword **async before a function** makes the function return a **promise**

https://www.w3schools.com/js/js_async.asp

**Promise**: https://www.w3schools.com/js/js_promise.asp

A **Promise** is an Object that links **Producing code** and **Consuming code**

"I **Promise** a Result!"

"**Producing code**" is code that can take some time

"**Consuming code**" is code that must wait for the result

You must use a Promise method to handle promises.

**fetch**: elhoz

https://www.w3schools.com/jsref/api_fetch.asp

The fetch() method starts the process of fetching a resource from a server.

The fetch() method returns a **Promise** that resolves to a Response object.

**await**: makes a function wait for a Promise

asszinkon működéssel együtt használható (A JavaScript asszinkron működésű)

The **await** keyword can only be used inside an **async** function.

https://www.w3schools.com/js/js_async.asp

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/await

The **await** keyword makes the function pause the execution and wait for a resolved promise before it continues.

GET módszeres továbbításnál az URL-ben adjuk tovább a paramétereket
POST módszeres továbbításnál a body-ban továbbítjuk azokat.

**ajax.js**

```
code="AAAAAAabc123";
url="http://gamf.nhely.hu/ajax1/";
async function read() {
  document.getElementById("code").innerHTML="code="+code;
  let response = await fetch(url, {
      method: 'post',
      cache: 'no-cache',
      headers: {
        'Content-Type': 'application/x-www-form-urlencoded',
      },
      body: "code="+code+"&op=read"
  });
  let data = await response.text();
  data = JSON.parse(data);
  let list = data.list;
  // !!! Először String-ben elkészítjük és csak a végén adjuk hozzá a DOM-hoz: divRead.innerHTML=str;
  str="<H1>Read</H1>";
  str+="<p>Number of records: "+data.rowCount+"</p>";
  str+="<p>Last max "+data.maxNum+" records:</p>";
  str+="<table><tr><th>id</th><th>name</th><th>city</th><th>phone</th><th>code</th></tr>";
  for(let i=0; i<list.length; i++)
    str +=
"<tr><td>"+list[i].id+"</td><td>"+list[i].name+"</td><td>"+list[i].city+"</td><td>"+list[i].phone+"</td
><td>"+list[i].code+"</td></tr>";
  str +="</table>";
  document.getElementById("readDiv").innerHTML=str;
}

async function create(){
  // name: reserved word
  nameStr = document.getElementById("name1").value;
  city = document.getElementById("city1").value;
  phone = document.getElementById("phone1").value;
  if(nameStr.length>0 && nameStr.length<=30 && city.length>0 && city.length<=30 &&
phone.length>0 && phone.length<=30 && code.length<=30){
    let response = await fetch(url, {
      method: 'post',
      cache: 'no-cache',
      headers: {
        'Content-Type': 'application/x-www-form-urlencoded',
      },
      body: "code="+code+"&op=create&name="+nameStr+"&city="+city+"&phone="+phone
    });
    let data = await response.text();
    if(data>0)
      str="Create successful!";
    else
    str="Create NOT successful!";
    document.getElementById("createResult").innerHTML=str;
    document.getElementById("name1").value="";
    document.getElementById("city1").value="";
```

```
    document.getElementById("phone1").value="";
    read();
  }
  else
    document.getElementById("createResult").innerHTML="Validation error!!";
}

async function getDataForId() {
  let response = await fetch(url, {
      method: 'post',
      cache: 'no-cache',
      headers: {
        'Content-Type': 'application/x-www-form-urlencoded',
      },
// A már használt op=read kérés segítségével ki tudjuk nyerni az adott ID-hez tartozó adatokat:
      body: "code="+code+"&op=read"
  });
  let data = await response.text();
  data = JSON.parse(data);
  let list = data.list;
  for(let i=0; i<list.length; i++)
// kiválasztjuk azt a rekordot, amelyiknek az ID-je megegyezik a megadott ID-vel,
//         és az datokat beírjuk a beviteli mezőkbe:
    if(list[i].id==document.getElementById("idUpd").value){
      document.getElementById("name2").value=list[i].name;
      document.getElementById("city2").value=list[i].city;
      document.getElementById("phone2").value=list[i].phone;
    }
}

async function update(){
  // name: reserved word
  id = document.getElementById("idUpd").value;
  nameStr = document.getElementById("name2").value;
  city = document.getElementById("city2").value;
  phone = document.getElementById("phone2").value;
  if(id.length>0 && id.length<=30 && nameStr.length>0 && nameStr.length<=30 && city.length>0 &&
city.length<=30 && phone.length>0 && phone.length<=30 && code.length<=30){
    let response = await fetch(url, {
      method: 'post',
      cache: 'no-cache',
      headers: {
        'Content-Type': 'application/x-www-form-urlencoded',
      },
      body:
"code="+code+"&op=update&id="+id+"&name="+nameStr+"&city="+city+"&phone="+phone
    });
    let data = await response.text();
    if(data>0)
      str="Update successful!";
    else
    str="Update NOT successful!";
    document.getElementById("updateResult").innerHTML=str;
```

```
// a végén kiürítjük a beviteli mezőket:
    document.getElementById("idUpd").value="";
    document.getElementById("name2").value="";
    document.getElementById("city2").value="";
    document.getElementById("phone2").value="";
    read();
  }
  else
    document.getElementById("updateResult").innerHTML="Validation error!!";
}

//delete: resetved word
async function deleteF(){
  id = document.getElementById("idDel").value;
  if(id.length>0 && id.length<=30){
    let response = await fetch(url, {
      method: 'post',
      cache: 'no-cache',
      headers: {
        'Content-Type': 'application/x-www-form-urlencoded',
      },
      body: "code="+code+"&op=delete&id="+id
    });
    let data = await response.text();
    if(data>0)
      str="Delete successful!";
    else
    str="Delete NOT successful!";
    document.getElementById("deleteResult").innerHTML=str;
    document.getElementById("idDel").value="";
    read();
  }
  else
    document.getElementById("deleteResult").innerHTML="Validation error!!";
}

window.onload = function() {
    read();
};
```

## Megoldás – 2 - régebbi XMLHttpRequest (XHR) JavaScript osztállyal – Csak olvasmány

**ajax.js**

```
code="AAAAAAabc123";
url="http://gamf.nhely.hu/ajax1/";
var xmlHttp=new XMLHttpRequest();
function read() {
  document.getElementById("code").innerHTML="code="+code;
  xmlHttp.open("POST",url,true);
  xmlHttp.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
  var params = "code="+code+"&op=read";
  xmlHttp.onreadystatechange = () => {
```

```
    if(xmlHttp.readyState == 4 && xmlHttp.status == 200) {
      let data = xmlHttp.responseText;
      data = JSON.parse(data);
      let list = data.list;
      str="<H1>Read</H1>";
      str+="<p>Number of records: "+data.rowCount+"</p>";
      str+="<p>Last max "+data.maxNum+" records:</p>";
      str+="<table><tr><th>id</th><th>name</th><th>city</th><th>phone</th><th>code</th></tr>";
      for(let i=0; i<list.length; i++)
        str +=
"<tr><td>"+list[i].id+"</td><td>"+list[i].name+"</td><td>"+list[i].city+"</td><td>"+list[i].phone+"</td
><td>"+list[i].code+"</td></tr>";
      str +="</table>";
      document.getElementById("readDiv").innerHTML=str;
    }
  };
  xmlHttp.send(params);
}

function create(){
  // name: reserved word
  nameStr = document.getElementById("name1").value;
  city = document.getElementById("city1").value;
  phone = document.getElementById("phone1").value;
  if(nameStr.length>0 && nameStr.length<=30 && city.length>0 && city.length<=30 &&
phone.length>0 && phone.length<=30 && code.length<=30){
    xmlHttp.open("POST",url,true);
    xmlHttp.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
    var params = "code="+code+"&op=create&name="+nameStr+"&city="+city+"&phone="+phone;
    xmlHttp.onreadystatechange = () => {
      if(xmlHttp.readyState == 4 && xmlHttp.status == 200) {
        let data = xmlHttp.responseText;
        if(data>0)
          str="Create successful!";
        else
        str="Create NOT successful!";
        document.getElementById("createResult").innerHTML=str;
        document.getElementById("name1").value="";
        document.getElementById("city1").value="";
        document.getElementById("phone1").value="";
        read();
      }
    };
    xmlHttp.send(params);
  }
  else
    document.getElementById("createResult").innerHTML="Validation error!!";
}

function getDataForId() {
  xmlHttp.open("POST",url,true);
  xmlHttp.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
  var params = "code="+code+"&op=read";
```

```
  xmlHttp.onreadystatechange = () => {
    if(xmlHttp.readyState == 4 && xmlHttp.status == 200) {
      let data = xmlHttp.responseText;
      data = JSON.parse(data);
      let list = data.list;
      for(let i=0; i<list.length; i++)
        if(list[i].id==document.getElementById("idUpd").value){
          document.getElementById("name2").value=list[i].name;
          document.getElementById("city2").value=list[i].city;
          document.getElementById("phone2").value=list[i].phone;
        }
    }
  };
  xmlHttp.send(params);
}

function update(){
  // name: reserved word
  id = document.getElementById("idUpd").value;
  nameStr = document.getElementById("name2").value;
  city = document.getElementById("city2").value;
  phone = document.getElementById("phone2").value;
  if(id.length>0 && id.length<=30 && nameStr.length>0 && nameStr.length<=30 && city.length>0 &&
city.length<=30 && phone.length>0 && phone.length<=30 && code.length<=30){
    xmlHttp.open("POST",url,true);
    xmlHttp.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
    var params =
"code="+code+"&op=update&id="+id+"&name="+nameStr+"&city="+city+"&phone="+phone;
    xmlHttp.onreadystatechange = () => {
      if(xmlHttp.readyState == 4 && xmlHttp.status == 200) {
        let data = xmlHttp.responseText;
        if(data>0)
          str="Update successful!";
        else
        str="Update NOT successful!";
        document.getElementById("updateResult").innerHTML=str;
        document.getElementById("idUpd").value="";
        document.getElementById("name2").value="";
        document.getElementById("city2").value="";
        document.getElementById("phone2").value="";
        read();
      }
    };
    xmlHttp.send(params);
  }
  else
    document.getElementById("updateResult").innerHTML="Validation error!!";
}

//delete: resetved word
function deleteF(){
  id = document.getElementById("idDel").value;
  if(id.length>0 && id.length<=30){
```

```
   xmlHttp.open("POST",url,true);
   xmlHttp.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
   var params = "code="+code+"&op=delete&id="+id;
   xmlHttp.onreadystatechange = () => {
    if(xmlHttp.readyState == 4 && xmlHttp.status == 200) {
      let data = xmlHttp.responseText;
      if(data>0)
        str="Delete successful!";
      else
      str="Delete NOT successful!";
      document.getElementById("deleteResult").innerHTML=str;
      document.getElementById("idDel").value="";
      read();
    }
   };
   xmlHttp.send(params);
  }
  else
   document.getElementById("deleteResult").innerHTML="Validation error!!";
}
window.onload = function() {
   read();
};
```

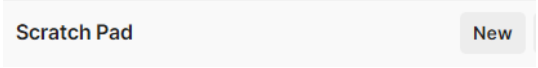
## Az API web-szolgáltatás tesztelése Postman-el – 10 perc – Hasonló a cURL-hez, de grafikus alkalmazással

https://www.postman.com/
https://www.postman.com/downloads/
Portable (nem kell telepíteni):
https://portapps.io/app/postman-portable/
Online: regisztrálni kell, de ingyenes:
https://www.postman.com/downloads/

**Töltsük le a Portable verziót.**
Indítsuk el az exe fájlt => Kicsomagolja a kért mappába.
Indítsuk el az alkalmazást.
Nem kell account-ot regisztrálni.

https://learning.postman.com/docs/getting-started/introduction/
https://learning.postman.com/docs/getting-started/sending-the-first-request/
https://learning.postman.com/docs/sending-requests/requests/

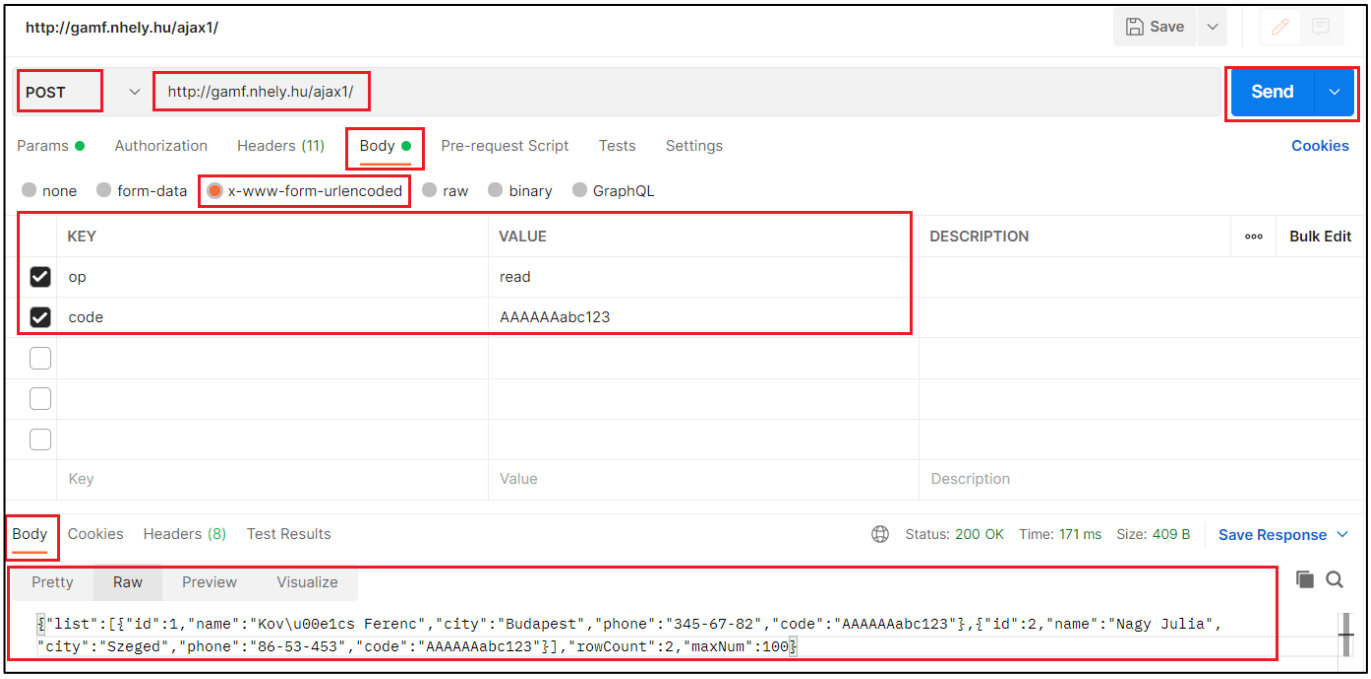
**Create a request**



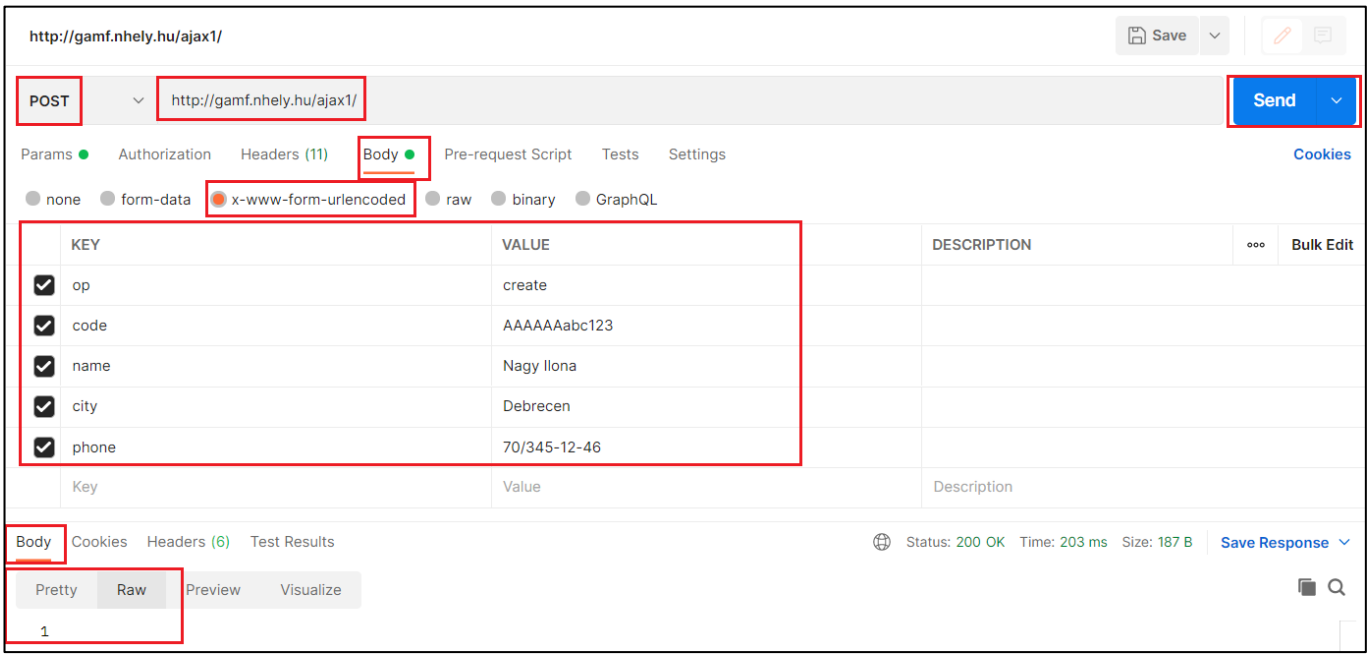

Scratch Pad / New
=> HTTP Request

http://gamf.nhely.hu/ajax1/

**A, Read**




## B, Create



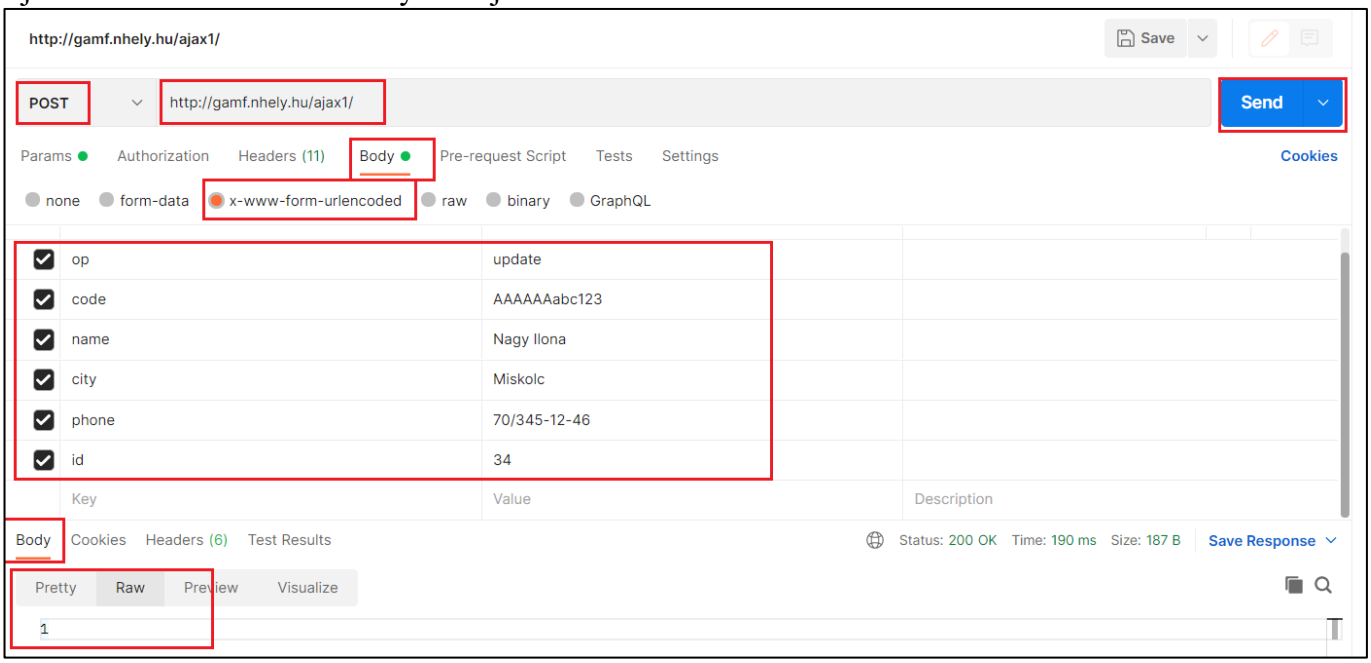

Ha újra lefuttatjuk a Read-et, akkor látjuk, hogy hozzáadta az új rekordot. Ez látszik az AJAX alkalmazásunkon is.

| id | name | city | phone | code |
|---|---|---|---|---|
| 1 | Kovács Ferenc | Budapest | 345-67-82 | AAAAAAabc123 |
| 2 | Nagy Julia | Szeged | 86-53-453 | AAAAAAabc123 |
| 34 | Nagy Ilona | Debrecen | 70/345-12-46 | AAAAAAabc123 |

Most id=34-es volt az új rekord.

**C, Update**

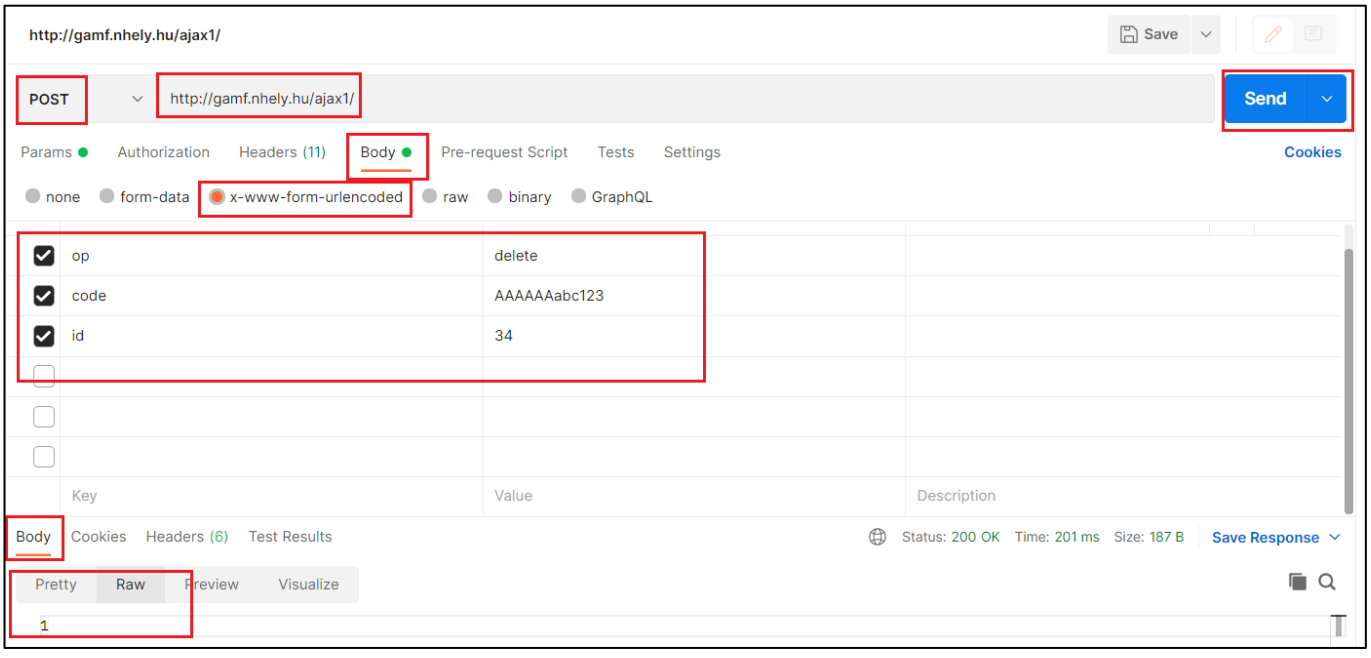
Írjuk át az ID=34-es rekord City mezőjét Miskolc-ra:




Ha újra lefuttatjuk a Read-et, akkor látjuk a változást. Ez látszik az AJAX alkalmazásunkon is.

**D, Delete**

Töröljük az ID=34-es rekordot:



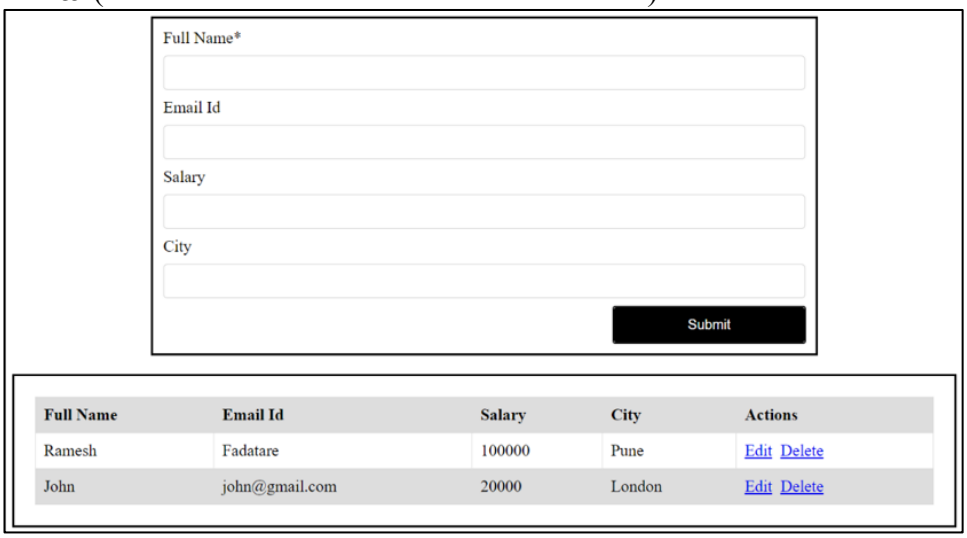

Ha újra lefuttatjuk a Read-et, akkor látjuk a változást. Ez látszik az AJAX alkalmazásunkon is.

## Gyakorló házi feladat – Rendezzék át a kinézetet

A feladatban a 4 CRUD funkciót külön részeken valósítottuk meg. Rendezze át a kinézetet a következő formába:

- Legyen egy űrlap a **Create** és az **Update** műveletekhez
- A rekordok után tegyen **Edit** és **Delete** hivatkozásokat

**Minta** (a feladatban más mezőneveket használunk)




## 7-React alapok - JavaScript library - framework

Van, ahol a React-ot library-nak, van ahol framework-nek hívják.

### Bevezetés

**Néhány React leírás:**
https://reactjs.org/tutorial/tutorial.html
https://www.w3schools.com/REACT/default.asp
https://react-tutorial.app/
https://www.javatpoint.com/reactjs-tutorial
https://www.tutorialspoint.com/reactjs/index.htm
https://www.newline.co/fullstack-react/
https://ibaslogic.com/react-tutorial-for-beginners/
https://www.freecodecamp.org/news/react-tutorial-build-a-project/

- React is a JavaScript library for building user interfaces.
- React is used to build single-page applications.
- React allows us to create reusable UI components.

#### Miért épp a React? - React vs Angular vs Vue

https://survey.stackoverflow.co/2023/#most-popular-technologies-webframe-prof
A megkérdezettek közül hányan használták: