**Лабораторна робота №3**

**Тема. Клієнт-серверна архітектура ПЗ. Створення RESTful API**

**Мета.** Ознайомитися з принципами клієнт-серверної архітектури та зрозуміти її роль у сучасних програмних системах. Навчитися створювати RESTful API для взаємодії між клієнтом і сервером. Закріпити практичні навички роботи з HTTP-запитами та відповідями. Розвинути вміння проєктувати та реалізовувати ендпойнти для типових CRUD-операцій.

**ЗАВДАННЯ**

**Завдання 1 — Створити свій Fake Online REST Server і виконати базові HTTP-запити**

**Завдання 2 — Робота з API сервісу CATAAS (Cat as a Service) та SwaggerAPI**

**Завдання 3 — Console-клієнт до публічного REST API**

**Завдання 4 — Інтеграція з Dog.CEO API у WinForms**

**Завдання 5 — WinForms-клієнт до публічного REST API**

**Хід роботи:**

Завдання 1

Підготовано репозиторій із файлом db.json, що містить користувачів та інформацію про їхні підписки: <https://github.com/BashukOleksii/Fake-API>.

Виконано запити у Postman:

Запит GET /users:

Status code: 200 OK

Body:

[

    {

        "id": 1,

        "name": "Alice",

        "email": "AliceEmail@gmail.com",

"coundDays": 100},

    {

        "id": 2,

        "name": "Tom",

        "email": "TomEmail@gmail.com",

        "countDays": 50

    },

    {

        "id": 3,

        "name": "John",

        "email": "JohnEmail@gmail.com",

        "countDays": 151

    },

    {

        "id": 4,

        "name": "Kate",

        "email": "KateEmail@gmail.com",

        "countDays": 1

    },

    {

        "id": 5,

        "name": "Sam",

        "email": "SamEmail@gmail.com",

        "countDays": 30

    }

]

Headers:

|  |  |
| --- | --- |
| Date | Sat, 27 Sep 2025 15:49:29 GMT |
| Content-Type | application/json; charset=utf-8 |
| Transfer-Encoding | chunked |
| Connection | keep-alive |
| access-control-allow-credentials | true |
| Cache-Control | no-cache |
| etag | W/"1eb-bTNTDOcOPVQfMJXFZ4BBsyxhSRs" |
| expires | -1 |
| nel | {"report\_to":"heroku-nel","response\_headers":["Via"],"max\_age":3600,"success\_fraction":0.01,"failure\_fraction":0.1} |
| pragma | no-cache |
| report-to | {"group":"heroku-nel","endpoints":[{"url":"https://nel.heroku.com/reports?s=rArF59kJ%2BJwO7%2BUqQHG9tE3ESx4XPI%2B9aHe22qTB5ow%3D\u0026sid=929419e7-33ea-4e2f-85f0-7d8b7cd5cbd6\u0026ts=1758988169"}],"max\_age":3600} |
| reporting-endpoints | heroku-nel="https://nel.heroku.com/reports?s=rArF59kJ%2BJwO7%2BUqQHG9tE3ESx4XPI%2B9aHe22qTB5ow%3D&sid=929419e7-33ea-4e2f-85f0-7d8b7cd5cbd6&ts=1758988169" |
| Server | cloudflare |
| vary | Origin, Accept-Encoding |
| via | 2.0 heroku-router |
| x-content-type-options | nosniff |
| x-powered-by | Express |
| cf-cache-status | DYNAMIC |
| Content-Encoding | br |
| CF-RAY | 985c26b93c9f3dd0-WAW |
| alt-svc | h3=":443"; ma=86400 |

Запит GET /users/1

Status code – 200 OK

Body:

{

"id": 1,

"name": "Alice",

"email": "AliceEmail@gmail.com",

"coundDays": 100

}

Headers:

|  |  |
| --- | --- |
| Date | Sat, 27 Sep 2025 15:59:05 GMT |
| Content-Type | application/json; charset=utf-8 |
| Transfer-Encoding | chunked |
| Connection | keep-alive |
| access-control-allow-credentials | true |
| Cache-Control | no-cache |
| etag | W/"57-Ua24HB085RMeA4UwCbi4y9ddcIQ" |
| expires | -1 |
| nel | {"report\_to":"heroku-nel","response\_headers":["Via"],"max\_age":3600,"success\_fraction":0.01,"failure\_fraction":0.1} |
| pragma | no-cache |
| report-to | {"group":"heroku-nel","endpoints":[{"url":"https://nel.heroku.com/reports?s=jhCweKOHTzXm86m1s73EJr3w9q4IHqE%2BQ3XzJ6fP6B0%3D\u0026sid=929419e7-33ea-4e2f-85f0-7d8b7cd5cbd6\u0026ts=1758988745"}],"max\_age":3600} |
| reporting-endpoints | heroku-nel="https://nel.heroku.com/reports?s=jhCweKOHTzXm86m1s73EJr3w9q4IHqE%2BQ3XzJ6fP6B0%3D&sid=929419e7-33ea-4e2f-85f0-7d8b7cd5cbd6&ts=1758988745" |
| Server | cloudflare |
| vary | Origin, Accept-Encoding |
| via | 2.0 heroku-router |
| x-content-type-options | nosniff |
| x-powered-by | Express |
| cf-cache-status | DYNAMIC |
| Content-Encoding | br |
| CF-RAY | 985c34c9890dbf45-WAW |
| alt-svc | h3=":443"; ma=86400 |

Запит GET /subscriptions?\_sort=ownerId&\_order=desc&\_limit=5

Status code – 200 OK

Body:

[

{

"id": 10,

"service": "AplleTv",

"ownerId": 5,

"type": "watch"

},

{

"id": 8,

"service": "Amazon",

"ownerId": 4,

"type": "watch"

},

{

"id": 9,

"service": "Spotify",

"ownerId": 4,

"type": "listen"

},

{

"id": 7,

"service": "Megogo",

"ownerId": 3,

"type": "watch"

},

{

"id": 4,

"service": "Netflix",

"ownerId": 2,

"type": "watch"

}

]

Headers:

|  |  |
| --- | --- |
| Date | Sat, 27 Sep 2025 16:04:46 GMT |
| Content-Type | application/json; charset=utf-8 |
| Transfer-Encoding | chunked |
| Connection | keep-alive |
| access-control-allow-credentials | true |
| access-control-expose-headers | X-Total-Count |
| Cache-Control | no-cache |
| etag | W/"1b0-rVFZJyZ2yQth0LEmia1TD59X0WA" |
| expires | -1 |
| nel | {"report\_to":"heroku-nel","response\_headers":["Via"],"max\_age":3600,"success\_fraction":0.01,"failure\_fraction":0.1} |
| pragma | no-cache |
| report-to | {"group":"heroku-nel","endpoints":[{"url":"https://nel.heroku.com/reports?s=ImqFRz0WhIHIH%2FYvNUW8rN11K1VlTnrwoFCWSAybsu4%3D\u0026sid=929419e7-33ea-4e2f-85f0-7d8b7cd5cbd6\u0026ts=1758989086"}],"max\_age":3600} |
| reporting-endpoints | heroku-nel="https://nel.heroku.com/reports?s=ImqFRz0WhIHIH%2FYvNUW8rN11K1VlTnrwoFCWSAybsu4%3D&sid=929419e7-33ea-4e2f-85f0-7d8b7cd5cbd6&ts=1758989086" |
| Server | cloudflare |
| vary | Origin, Accept-Encoding |
| via | 2.0 heroku-router |
| x-content-type-options | nosniff |
| x-powered-by | Express |
| x-total-count | 10 |
| cf-cache-status | DYNAMIC |
| Content-Encoding | br |
| CF-RAY | 985c3d1e39e2eeb6-WAW |
| alt-svc | h3=":443"; ma=86400 |

Запит GET /subscriptions?type=game:

Status code = 200 OK

Body:

[

{

"id": 3,

"service": "XBox",

"ownerId": 1,

"type": "game"

},

{

"id": 5,

"service": "PlayStation",

"ownerId": 2,

"type": "game"

},

{

"id": 6,

"service": "XBox",

"ownerId": 2,

"type": "game"

}

]

Headers:

|  |  |
| --- | --- |
| Date | Sat, 27 Sep 2025 16:08:42 GMT |
| Content-Type | application/json; charset=utf-8 |
| Transfer-Encoding | chunked |
| Connection | keep-alive |
| access-control-allow-credentials | true |
| Cache-Control | no-cache |
| etag | W/"ff-3G/UeDaYsoq1XoiCscEm4qqqGkQ" |
| expires | -1 |
| nel | {"report\_to":"heroku-nel","response\_headers":["Via"],"max\_age":3600,"success\_fraction":0.01,"failure\_fraction":0.1} |
| pragma | no-cache |
| report-to | {"group":"heroku-nel","endpoints":[{"url":"https://nel.heroku.com/reports?s=yfxn6cRv4YgHhwexlRHgMn2GuOHqiK3JY2mEbwhhLyU%3D\u0026sid=929419e7-33ea-4e2f-85f0-7d8b7cd5cbd6\u0026ts=1758989322"}],"max\_age":3600} |
| reporting-endpoints | heroku-nel="https://nel.heroku.com/reports?s=yfxn6cRv4YgHhwexlRHgMn2GuOHqiK3JY2mEbwhhLyU%3D&sid=929419e7-33ea-4e2f-85f0-7d8b7cd5cbd6&ts=1758989322" |
| Server | cloudflare |
| vary | Origin, Accept-Encoding |
| via | 2.0 heroku-router |
| x-content-type-options | nosniff |
| x-powered-by | Express |
| cf-cache-status | DYNAMIC |
| Content-Encoding | br |
| CF-RAY | 985c42de3ba7c06d-WAW |
| alt-svc | h3=":443"; ma=86400 |

Виконано запит GET /subscriptions?type=watch&ownerId=1

Status code – 200 OK

Body:

[

{

"id": 1,

"service": "Netflix",

"ownerId": 1,

"type": "watch"

},

{

"id": 2,

"service": "YouTube",

"ownerId": 1,

"type": "watch"

}

]

Headers

|  |  |
| --- | --- |
| Date | Sat, 27 Sep 2025 16:25:37 GMT |
| Content-Type | application/json; charset=utf-8 |
| Transfer-Encoding | chunked |
| Connection | keep-alive |
| access-control-allow-credentials | true |
| Cache-Control | no-cache |
| etag | W/"ae-UCQ85Nuedc1MyTtLDWK8f8KRNzQ" |
| expires | -1 |
| nel | {"report\_to":"heroku-nel","response\_headers":["Via"],"max\_age":3600,"success\_fraction":0.01,"failure\_fraction":0.1} |
| pragma | no-cache |
| report-to | {"group":"heroku-nel","endpoints":[{"url":"https://nel.heroku.com/reports?s=1IR5XurCa8a%2FEtcv53udKCkmvSJiOu9yvJtrpw%2BKTP4%3D\u0026sid=929419e7-33ea-4e2f-85f0-7d8b7cd5cbd6\u0026ts=1758990337"}],"max\_age":3600} |
| reporting-endpoints | heroku-nel="https://nel.heroku.com/reports?s=1IR5XurCa8a%2FEtcv53udKCkmvSJiOu9yvJtrpw%2BKTP4%3D&sid=929419e7-33ea-4e2f-85f0-7d8b7cd5cbd6&ts=1758990337" |
| Server | cloudflare |
| vary | Origin, Accept-Encoding |
| via | 2.0 heroku-router |
| x-content-type-options | nosniff |
| x-powered-by | Express |
| cf-cache-status | DYNAMIC |
| Content-Encoding | br |
| CF-RAY | 985c5baa6b08c3ae-WAW |
| alt-svc | h3=":443"; ma=86400 |

Завдання 2

**Робота з Postman.**

Запит GET /cat?json=true

Status -200 OK

Body:



Рисунок 1 – Результат запиту i

Headers:

|  |  |
| --- | --- |
| Access-Control-Allow-Headers | X-Requested-With, Content-Type, Accept, Origin, Authorization |
| Access-Control-Allow-Methods | GET, POST, PUT, DELETE, OPTIONS |
| Access-Control-Allow-Origin | \* |
| Content-Length | 41506 |
| Content-Type | image/jpeg |
| Date | Sat, 27 Sep 2025 17:34:10 GMT |

Запит GET /cat/says/Hello world

Status -200 OK

Body:



Рисунок 2 – результат запиту ii

Headers:

|  |  |
| --- | --- |
| Access-Control-Allow-Headers | X-Requested-With, Content-Type, Accept, Origin, Authorization |
| Access-Control-Allow-Methods | GET, POST, PUT, DELETE, OPTIONS |
| Access-Control-Allow-Origin | \* |
| Content-Length | 25250 |
| Content-Type | image/jpeg |
| Date | Sat, 27 Sep 2025 17:37:05 GMT |

Запит GET cat/says/API?height=300&width=300&filter=negate

Status -200 OK

Body:



Рисунок 3 – результат запиту iii

Headers:

|  |  |
| --- | --- |
| Access-Control-Allow-Headers | X-Requested-With, Content-Type, Accept, Origin, Authorization |
| Access-Control-Allow-Methods | GET, POST, PUT, DELETE, OPTIONS |
| Access-Control-Allow-Origin | \* |
| Content-Length | 19647 |
| Content-Type | image/jpeg |
| Date | Sat, 27 Sep 2025 17:43:32 GMT |

Запит GET cat/gif

Status -200 OK

Body:



Рисунок 4 – результат запиту iv

Headers:

|  |  |
| --- | --- |
| Access-Control-Allow-Headers | X-Requested-With, Content-Type, Accept, Origin, Authorization |
| Access-Control-Allow-Methods | GET, POST, PUT, DELETE, OPTIONS |
| Access-Control-Allow-Origin | \* |
| Content-Length | 832216 |
| Content-Type | image/gif |
| Date | Sat, 27 Sep 2025 17:45:57 GMT |

Запит GET api/cats?\_limit=5

Status -200 OK

Body:

[

{

"id": "04eEQhDfAL8l5nt3",

"tags": [

"two",

"double",

"black"

],

"mimetype": "image/jpeg",

"createdAt": "2022-07-18T11:28:29.596Z"

},

{

"id": "05Xd4JtN14983pns",

"tags": [

"Cute"

],

"mimetype": "image/jpeg",

"createdAt": "2024-05-27T17:55:08.552Z"

},

{

"id": "09wFxpacQzvf9jfM",

"tags": [

"Maskcat"

],

"mimetype": "image/jpeg",

"createdAt": "2021-08-17T06:26:37.959Z"

},

{

"id": "0B2g7aTANObiqPJJ",

"tags": [

"creation"

],

"mimetype": "image/jpeg",

"createdAt": "2016-11-25T03:46:12.562Z"

},

{

"id": "0BTTVEVWXNyOgXYd",

"tags": [],

"mimetype": "image/jpeg",

"createdAt": "2020-10-19T18:52:55.627Z"

},

{

"id": "0C2bQ39x8kuhx31p",

"tags": [

"sara",

"looking"

],

"mimetype": "image/jpeg",

"createdAt": "2021-11-11T10:16:22.061Z"

},

{

"id": "0DVs2d6bIVIt3ehk",

"tags": [

"birthday",

"cake",

"happy"

],

"mimetype": "image/gif",

"createdAt": "2024-02-06T20:07:13.052Z"

},

{

"id": "0EsIYDG0at0TPpPD",

"tags": [

"fat"

],

"mimetype": "image/jpeg",

"createdAt": "2022-03-26T23:13:25.966Z"

},

{

"id": "0F0IKAPOdWiE755P",

"tags": [

"meet",

"cute"

],

"mimetype": "image/jpeg",

"createdAt": "2024-06-18T09:46:45.702Z"

},

{

"id": "0GC9MRUAqxhBzPyA",

"tags": [

"cute"

],

"mimetype": "image/png",

"createdAt": "2024-09-15T15:45:25.375Z"

}

]

Headers:

|  |  |
| --- | --- |
| Access-Control-Allow-Headers | X-Requested-With, Content-Type, Accept, Origin, Authorization |
| Access-Control-Allow-Methods | GET, POST, PUT, DELETE, OPTIONS |
| Access-Control-Allow-Origin | \* |
| Content-Size | 1101 |
| Content-Type | application/json |
| Date | Sat, 27 Sep 2025 17:49:24 GMT |
| Transfer-Encoding | chunked |

Запит GET cat/gif?json=true

Status -200 OK

Body:

{

"id": "3Z6CcYkHotdUXQC9",

"tags": [

"kitten",

"please",

"cute",

"gif"

],

"created\_at": "2017-10-19T08:59:41.920Z",

"url": "https://cataas.com/cat/3Z6CcYkHotdUXQC9?position=center",

"mimetype": "image/gif"

}

Headers:

|  |  |
| --- | --- |
| Access-Control-Allow-Headers | X-Requested-With, Content-Type, Accept, Origin, Authorization |
| Access-Control-Allow-Methods | GET, POST, PUT, DELETE, OPTIONS |
| Access-Control-Allow-Origin | \* |
| Content-Length | 192 |
| Content-Type | application/json; charset=utf-8 |
| Date | Sat, 27 Sep 2025 17:50:33 GMT |

**Робота з SwaggerAPI.**

**Запит 1.**

Request URL: https://cataas.com/cat?type=medium&position=center

Code: 200

Response body:



Рисунок 5 – Запит 1 body

Response headers:

access-control-allow-headers: X-Requested-With,Content-Type,Accept,Origin,Authorization

access-control-allow-methods: GET,POST,PUT,DELETE,OPTIONS

access-control-allow-origin: \*

content-length: 19092

content-type: image/jpeg

date: Sat,27 Sep 2025 18:54:58 GMT

**Запит 2.**

Request URL: https://cataas.com/cat/cute?position=center&width=300&height=300

Code: 200

Response body:



Рисунок 6 – Запит 2 body

Response headers:

access-control-allow-headers: X-Requested-With,Content Type,Accept,Origin,Authorization

access-control-allow-methods: GET,POST,PUT,DELETE,OPTIONS

access-control-allow-origin: \*

content-length: 16793

content-type: image/jpeg

date: Sat,27 Sep 2025 19:01:52 GMT

**Запит 3.**

Request URL:https://cataas.com/api/cats?limit=5&skip=0

Code: 200

Response body:

[

{

"id": "04eEQhDfAL8l5nt3",

"tags": [

"two",

"double",

"black"

],

"mimetype": "image/jpeg",

"createdAt": "2022-07-18T11:28:29.596Z"

},

{

"id": "05Xd4JtN14983pns",

"tags": [

"Cute"

],

"mimetype": "image/jpeg",

"createdAt": "2024-05-27T17:55:08.552Z"

},

{

"id": "09wFxpacQzvf9jfM",

"tags": [

"Maskcat"

],

"mimetype": "image/jpeg",

"createdAt": "2021-08-17T06:26:37.959Z"

},

{

"id": "0B2g7aTANObiqPJJ",

"tags": [

"creation"

],

"mimetype": "image/jpeg",

"createdAt": "2016-11-25T03:46:12.562Z"

},

{

"id": "0BTTVEVWXNyOgXYd",

"tags": [],

"mimetype": "image/jpeg",

"createdAt": "2020-10-19T18:52:55.627Z"

}

]

Response headers:

access-control-allow-headers: X-Requested-With,Content-Type,Accept,Origin,Authorization

access-control-allow-methods: GET,POST,PUT,DELETE,OPTIONS

access-control-allow-origin: \*

content-length: 543

content-type: application/json

date: Sat,27 Sep 2025 19:04:08 GMT

Завдання 3

Було спроєктовано програму для отримання інформації з REST Countries та виведення отриманої інформації у консоль.

Програма:

using System;

using System.Net.Http;

using System.Text.Json;

using System.Threading.Tasks;

public class Country

{

    public Name name { get; set; }

    public List<string> capital { get; set; }

    public string region { get; set; }

    public double area { get; set; }

    public long population { get; set; }

}

public class Name

{

    public string common { get; set; }

    public string official { get; set; }

}

class Progrm

{

    private static readonly HttpClient \_http = new HttpClient {

        BaseAddress = new Uri("https://restcountries.com"),

        Timeout = TimeSpan.FromSeconds(15)

    };

    static async Task Main()

    {

        try

        {

            Console.WriteLine("Запит до REST Countries...");

            var response = await \_http.GetAsync("/v3.1/name/ukraine");

            response.EnsureSuccessStatusCode();

            var json = await response.Content.ReadAsStringAsync();

            Console.WriteLine("\nRaw JSON:");

            Console.WriteLine(json);

            var country = JsonSerializer.Deserialize<List<Country>>(json);

            if(country != null)

            {

                Console.WriteLine("\n ===== Результат парсингу =====");

                Console.WriteLine($"Country common name: {country[0].name.common}");

                Console.WriteLine($"Country official name: {country[0].name.official}");

                Console.WriteLine($"Country capital: {country[0].capital[0]}");

                Console.WriteLine($"Country region: {country[0].region}");

                Console.WriteLine($"Country area: {country[0].area}");

                Console.WriteLine($"Country population: {country[0].population}");

            }

        }

        catch (Exception ex){

            Console.WriteLine("Помилка " + ex.Message);

        }

    }

}

Результат:

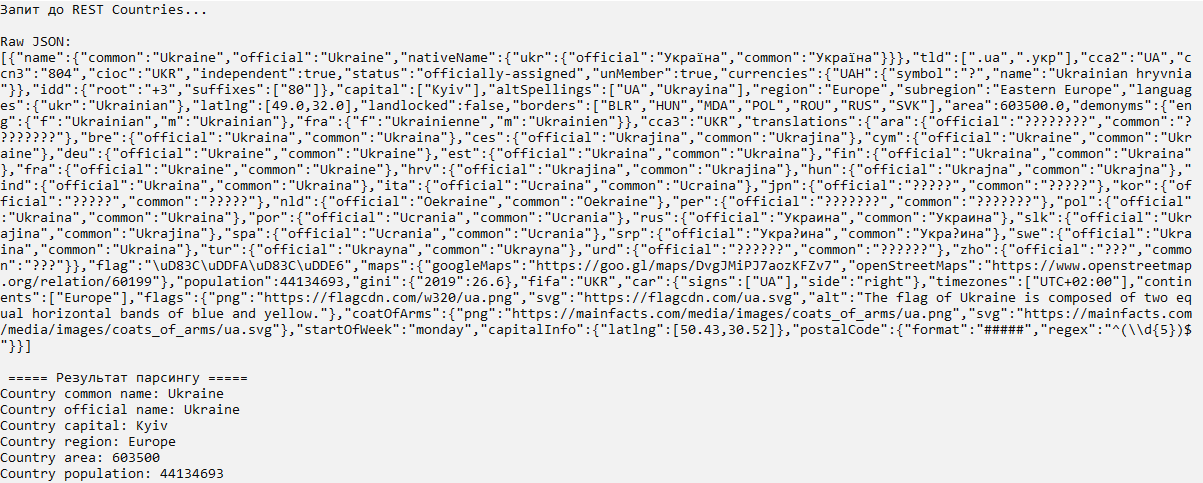


Рисунок 7 – Результат консольної програми

Завдання 4.

**Завдання 4.1:**

Було розроблено WinForm проєкт для відображення зображення собаки за допомогою endpoint - <https://dog.ceo/api/breeds/image/random>.

Програма:

using System;

using System.Net.Http;

using System.Text.Json;

using System.Windows.Forms;

namespace WinFormsAppRESTAPI4.\_1

{

    public partial class Form1 : Form

    {

        private static readonly HttpClient \_http = new HttpClient

        {

            BaseAddress = new Uri("https://dog.ceo"),

            Timeout = TimeSpan.FromSeconds(30)

        };

        public Form1()

        {

            InitializeComponent();

        }

        private void textBox2\_KeyPress(object sender, KeyPressEventArgs e)

        {

            e.Handled = true;

        }

        private async void button1\_Click(object sender, EventArgs e)

        {

            try

            {

                textBox2.Clear();

                button1.Enabled = false;

                var response = await \_http.GetAsync("/api/breeds/image/random";

                response.EnsureSuccessStatusCode();

                var json = await response.Content.ReadAsStringAsync();

                var Dog = JsonSerializer.Deserialize<Dog>(json, new JsonSerializerOptions { PropertyNameCaseInsensitive = true });

                if (Dog != null)

                {

                    pictureBox1.LoadAsync(Dog.message);

                    textBox2.Text += "{" + Environment.NewLine + $"message: {Dog.message}," + Environment.NewLine + $"status: {Dog.status}" + Environment.NewLine + "}";

                }

            }

            catch (Exception ex)

            {

                MessageBox.Show($"Помилка: {ex.Message}", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Stop); ;

            }

            finally

            {

                button1.Enabled = true;

            }

        }

    }

    public class Dog

    {

        public string message { get; set; }

        public string status { get; set; }

    }

}

Результат:

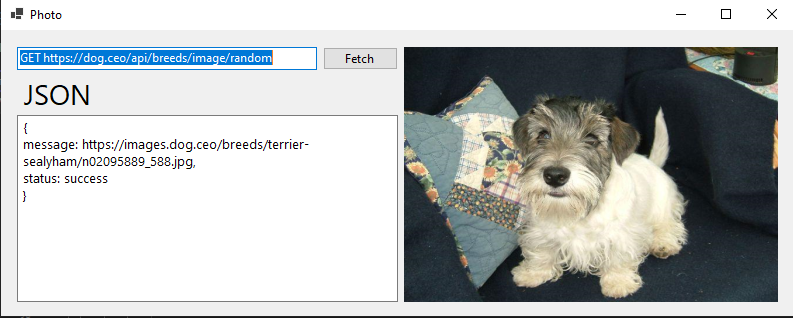


Рисунок 8 – Рузультат завдання 4.1

**Завдання 4.2**

Було розроблено WinForm проєкт для відображення зображення собак за допомогою endpoint - [https://dog.ceo/api/breeds/image/random/{n}](https://dog.ceo/api/breeds/image/random/%7bn%7d).

Програма:

using System;

using System.Net.Http;

using System.Text.Json;

using System.Windows.Forms;

namespace WinFormsAppRESTAPI4.\_1

{

    public partial class Form1 : Form

    {

        int index = 0;

        int max = 0;

        List<string> photos;

        private static readonly HttpClient \_http = new HttpClient

        {

            BaseAddress = new Uri("https://dog.ceo"),

            Timeout = TimeSpan.FromSeconds(30)

        };

        public Form1()

        {

            InitializeComponent();

            photos = new List<string>();

        }

        private void textBox2\_KeyPress(object sender, KeyPressEventArgs e)

        {

            e.Handled = true;

        }

        private async void button1\_Click(object sender, EventArgs e)

        {

            if (string.IsNullOrEmpty(textBox3.Text))

                return;

            int N = int.Parse(textBox3.Text);

            if (N > 50)

            {

                MessageBox.Show("Максимальна кількість 50 зображень", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Stop);

                return;

            }

            string request = $"/api/breeds/image/random/{N}";

            try

            {

                index = 0;

                max = N;

                textBox2.Clear();

                button1.Enabled = false;

                var response = await \_http.GetAsync(request);

                response.EnsureSuccessStatusCode();

                var json = await response.Content.ReadAsStringAsync();

                var Dogs = JsonSerializer.Deserialize<Dog>(json);

                if (Dogs != null)

                {

                    photos = Dogs.message;

                    pictureBox1.LoadAsync(Dogs.message[0]);

                    textBox2.Text += "{" + Environment.NewLine + $"message: ";

                    foreach (string item in Dogs.message)

                        textBox2.Text += item + "," + Environment.NewLine;

                    textBox2.Text += $"status: {Dogs.status}" + Environment.NewLine + "}";

                }

                else

                    photos.Clear();

            }

            catch (Exception ex)

            {

                MessageBox.Show($"Помилка: {ex.Message}", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Stop);

                max = 0;

                photos.Clear();

            }

            finally

            {

                button1.Enabled = true;

            }

        }

        private void textBox3\_KeyPress(object sender, KeyPressEventArgs e)

        {

            if (char.IsDigit(e.KeyChar) || char.IsControl(e.KeyChar))

                return;

            e.Handled = true;

        }

        private void button2\_Click(object sender, EventArgs e)

        {

            Button button = (Button)sender;

            string text = button.Text;

            if(text == ">" && photos.Count > 0 && index + 1 < max)

                index++;

            else if(text == "<" && index > 0 && photos.Count > 0)

                index--;

            if (photos.Count > 0)

                pictureBox1.LoadAsync(photos[index]);

        }

    }

public class Dog

    {

        public List<string> message { get; set; }

        public string status { get; set; }

    }

}

Результат:

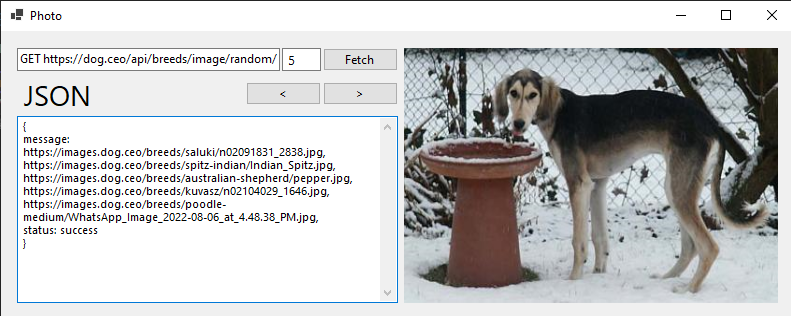


Рисунок 9 – Результат завдання 4.2 перше фото

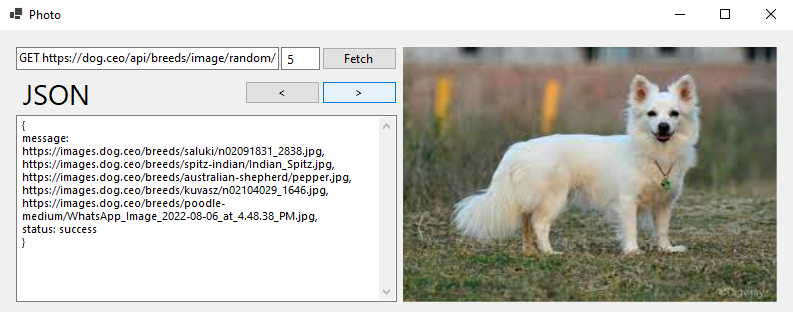


Рисунок 10 – Результат завдання 4.2 друге фото

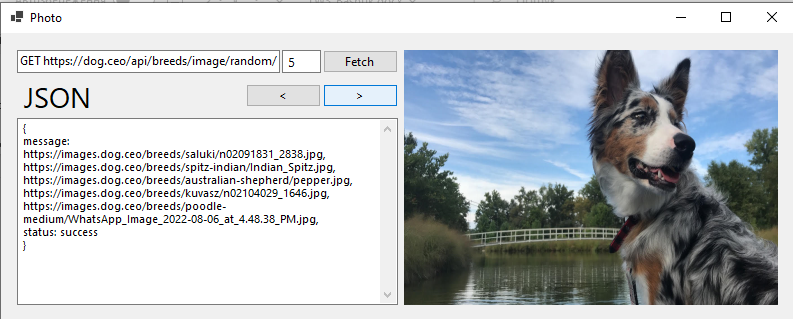
****

Рисунок 11 - Результат завдання 4.2 третє фото

Завдання 5.

Було розроблено багатомодульний WinForm проєкт для реалізації програми для відображення країн та додаткової інормації про них.

DTO.cs:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace RESTAPI\_5

{

    public class Country

    {

        public Name name { get; set; }

        public List<string> capital { get; set; }

        public string region { get; set; } = string.Empty;

        public string subregion { get; set; } = string.Empty;//

        public double area { get; set; }

        public long population { get; set; }

        public Flags flags { get; set; }

    }

    public class Name

    {

        public string common { get; set; } = string.Empty;

        public string official { get; set; } = string.Empty;

    }

    public class Flags

    {

        public string png { get; set; } = string.Empty ;

        public string alt { get; set; } = string.Empty;

    }

}

APIClient.cs:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Net.Http;

using System.Text.Json;

using System.Net;

using System.Security.Cryptography;

namespace RESTAPI\_5

{

    public class APIClient

    {

        const string baseAdres = "https://restcountries.com/v3.1/";

        static readonly HttpClient \_http = new HttpClient();

        public string ErorMessage { get; private set; }

        public APIClient(int seconds)

        {

            \_http.Timeout = TimeSpan.FromSeconds(seconds);

            \_http.BaseAddress = new Uri(baseAdres);

        }

        public async Task<List<Country>> Request(string url)

        {

            try

            {

                var response = await \_http.GetAsync(url);

                response.EnsureSuccessStatusCode();

                string json = await response.Content.ReadAsStringAsync();

                var counties = JsonSerializer.Deserialize<List<Country>>(json);

                return counties;

            }

            catch (HttpRequestException hre)

            {

                ErorMessage = $"Помилка мережi або сервера {hre.Message}";

                if (hre.StatusCode.HasValue)

                {

                    switch (hre.StatusCode)

                    {

                        case HttpStatusCode.BadRequest: ErorMessage = "Неправильний запит, не можна обробити.";break;

                        case HttpStatusCode.Unauthorized: ErorMessage = "Сервер не може вас розпізнати"; break;

                        case HttpStatusCode.Forbidden: ErorMessage = "Сервер заборонив доступ"; break;

                        case HttpStatusCode.NotFound: ErorMessage = "Інформації за запитом не знайдено"; break;

                        case HttpStatusCode.InternalServerError: ErorMessage = "Внутрішні проблеми сервера";break;

                        case HttpStatusCode.NotImplemented: ErorMessage = "Функцію не реалізовано"; break;

                        case HttpStatusCode.BadGateway: ErorMessage = "Поганий шлюз"; break;

                        case HttpStatusCode.ServiceUnavailable: ErorMessage = "Сервіс зараз недоступний"; break;

                        case HttpStatusCode.GatewayTimeout: ErorMessage = "Час очікування шлюзу закінчвся"; break;

                    }

                }

                return null;

            }

            catch (TaskCanceledException)

            {

                ErorMessage = "Вичерпався час очікування";

                return null;

            }

            catch (JsonException)

            {

                ErorMessage = "Невірний формат.";

                return null;

            }

            catch(Exception e)

            {

                ErorMessage = $"Помилка {e.Message}";

                return null;

            }

        }

    }

}

Form1.cs:

using System.Data;

namespace RESTAPI\_5

{

    public partial class Form1 : Form

    {

        DataTable dt;

        List<Country> countries;

        APIClient aPIClient;

        public Form1()

        {

            InitializeComponent();

            aPIClient = new APIClient(15);

            LoadTable();

            Load("all?fields=name,capital,region,subregion,area,population,flags");

        }

        private void LoadTable()

        {

            dt = new DataTable();

            dt.Columns.Add("Поширена назва", typeof(string));

            dt.Columns.Add("Офіційна назва", typeof(string));

            dt.Columns.Add("Столиця", typeof(string));

            dt.Columns.Add("Регіон", typeof(string));

            dt.Columns.Add("Площа", typeof(double));

            dt.Columns.Add("Населення", typeof(long));

        }

        private async Task Load(string url)

        {

            Loading(false, "Заванаження");

            countries = await aPIClient.Request(url);

            if (countries != null)

            {

                ListToTable();

                UploadDataGrid();

            }

            else

                MessageBox.Show(aPIClient.ErorMessage, "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);

            Loading(true, "Робота");

        }

        private void ListToTable()

        {

            dt.Rows.Clear();

            foreach (Country country in countries)

                dt.Rows.Add(country.name.common, country.name.official, returnCapital(country.capital), country.region, country.area, country.population);

        }

        private string returnCapital(List<string> c)

        {

            if (c == null)

                return "Немає";

            string capitals = "";

            foreach (string captital in c)

                capitals += captital + " ";

            return capitals;

        }

        private void UploadDataGrid()

        {

            dataGridView1.DataSource = null;

            dataGridView1.DataSource = dt;

        }

        private void Loading(bool status, string text)

        {

            StatusLabel.Text = text;

            button1.Enabled = status;

            button2.Enabled = status;

            findBox.Enabled = status;

            dataGridView1.Enabled = status;

            if (!status)

                this.Cursor = Cursors.WaitCursor;

            else

                this.Cursor = Cursors.Default;

        }

        private async void button2\_Click(object sender, EventArgs e)

        {

            await Load("all?fields=name,capital,region,subregion,area,population,flags");

            findBox.Clear();

        }

        private async void button1\_Click(object sender, EventArgs e)

        {

            if (string.IsNullOrEmpty(findBox.Text))

                return;

            await Load($"name/{findBox.Text}");

            findBox.Clear();

        }

        private async Task Information(string name)

        {

            Loading(false, "Відображення");

            List<Country> c = await aPIClient.Request($"name/{name}");

            if (c == null)

            {

                MessageBox.Show($"Помилка: {aPIClient.ErorMessage}", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);

                return;

            }

           Common.Text = c[0].name.common;

            Official.Text = c[0].name.official;

            Capital.Text = returnCapital(c[0].capital);

            Sub.Text = c[0].subregion;

            Region.Text = c[0].region;

            Area.Text = c[0].area.ToString();

            Population.Text = c[0].population.ToString();

            pictureBox1.LoadAsync(c[0].flags.png);

            Flag.Text = c[0].flags.alt;

            Loading(true, "Показ");

        }

        private async void dataGridView1\_CellClick(object sender, DataGridViewCellEventArgs e)

        {

            string cName = string.Empty;

            if (e.RowIndex < 0)

                return;

            cName = dataGridView1.Rows[e.RowIndex].Cells[0].Value.ToString();

            await Information(cName);

        }

    }

}

Результат роот програми:

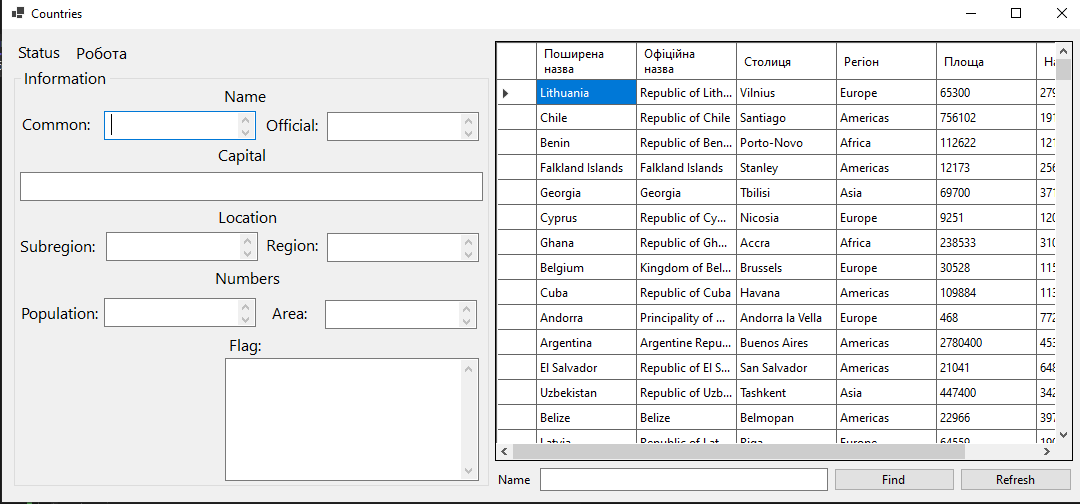


Рисунок 12 – Початковий екран 5 завдання

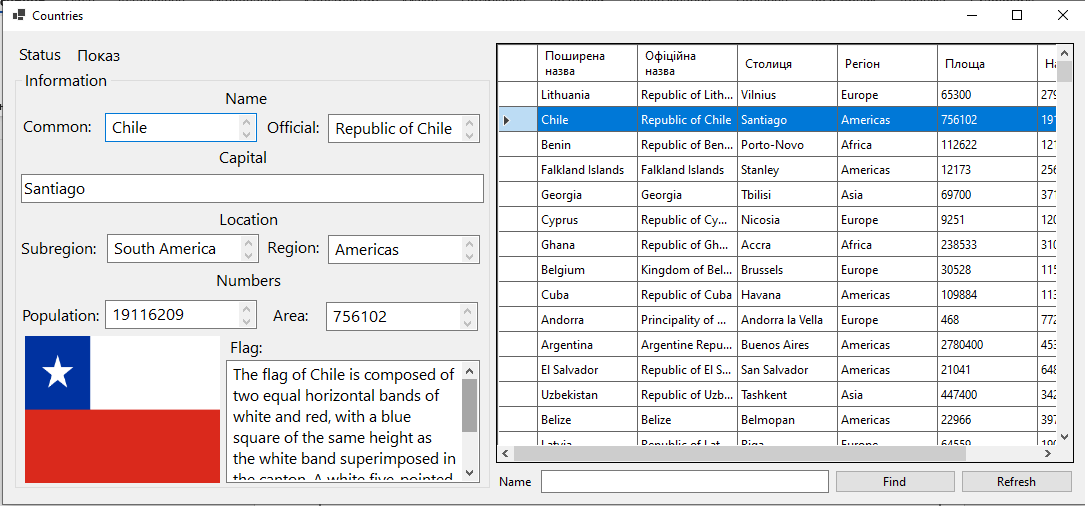


Рисунок 13 - Обрано країну

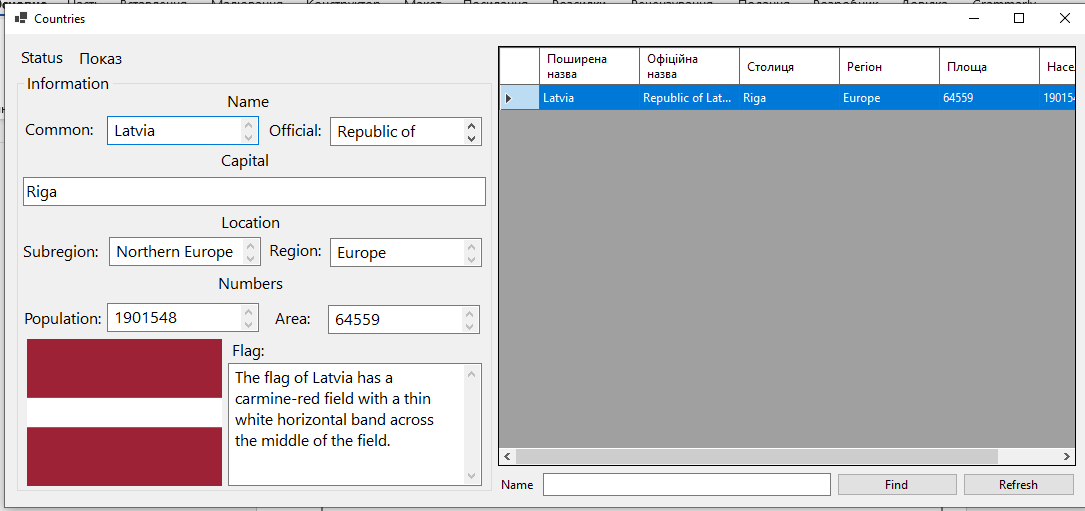


Рисунок 14 – Пошук за допомогою Name

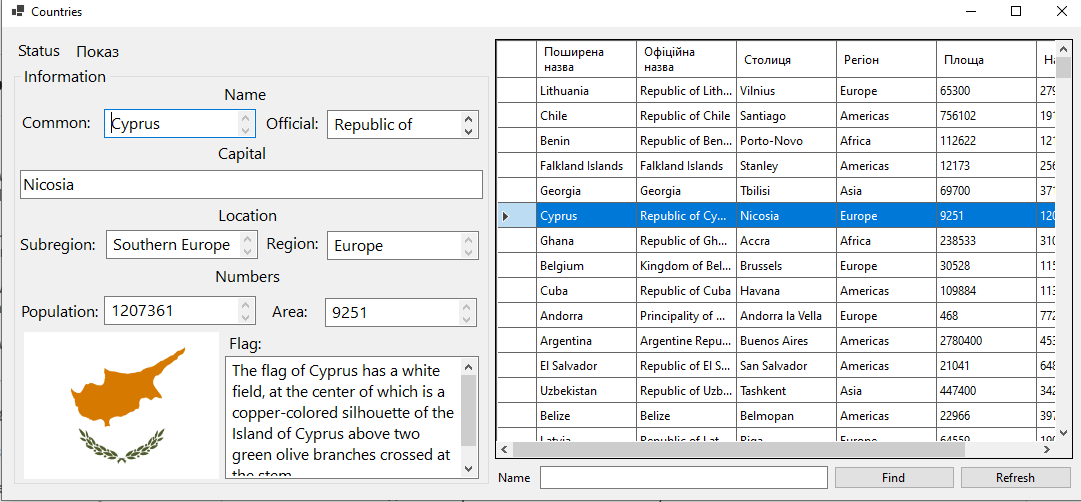


Рисунок 15 – Оновлено завантажувальний екран

**Висновок.**

Я ознайомиавя з принципами клієнт-серверної архітектури та зрозуміти її роль у сучасних програмних системах. Навчився створювати RESTful API для взаємодії між клієнтом і сервером. Закріпив практичні навички роботи з HTTP-запитами та відповідями. Розвив вміння проєктувати та реалізовувати ендпойнти для типових CRUD-операцій.