

Mariusz Wieczorek

mariusz.wieczorek@kabat.pl

Streszczenie

Co to jest DTO  
Co to jest DAO

WEB API

Testowanie

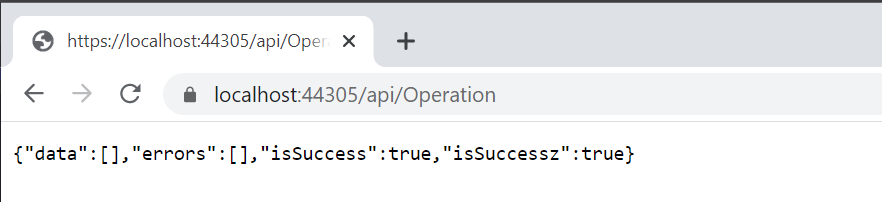
**Testowanie Aplikacji**

W pliku **LaunchSettings.Json** konfigurujemy jaka strona ma wystartować, gdy wpiszemy adres serwera w przeglądarce.

|  |
| --- |
| {  "$schema": "http://json.schemastore.org/launchsettings.json",  "iisSettings": {  "windowsAuthentication": false,  "anonymousAuthentication": true,  "iisExpress": {  "applicationUrl": "http://localhost:61651",  "sslPort": 44305  }  },  "profiles": {  "IIS Express": {  "commandName": "IISExpress",  "launchBrowser": true,  "launchUrl": "swagger",  "environmentVariables": {  "ASPNETCORE\_ENVIRONMENT": "Development"  }  },  "MyFinances.WebApi": {  "commandName": "Project",  "dotnetRunMessages": "true",  "launchBrowser": true,  "launchUrl": "swagger",  "applicationUrl": "https://localhost:5001;http://localhost:5000",  "environmentVariables": {  "ASPNETCORE\_ENVIRONMENT": "Development"  }  }  }  } |

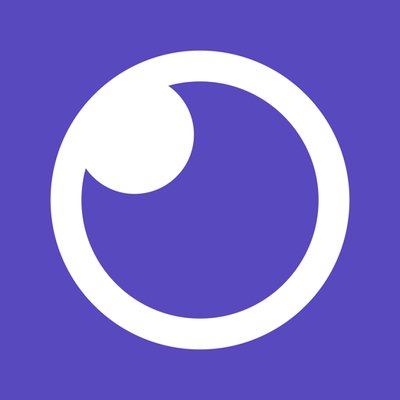
**Testowanie Aplikacji**

Przeglądarka nie nadaje się do testowania aplikacji Web API typu REST.   
Testować w niej możemy jedynie akcje typu HttpGet.  
Po wpisaniu w przeglądarce adresu jak poniżej, otrzymamy odpowiedź:

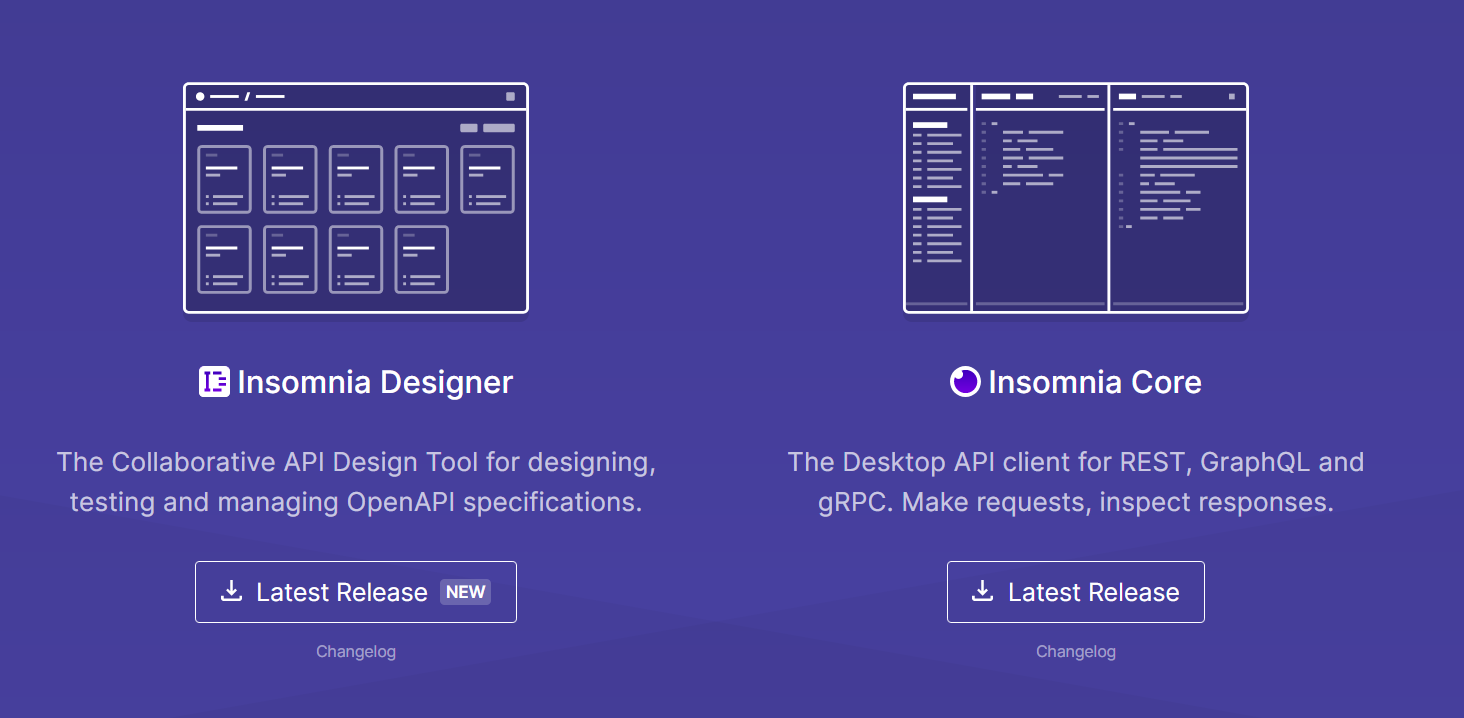


Aby przetestować inne metody niż HttpGet musimy użyć zewnętrznego narzędzia.  
Przykładem takiego narzędzia jest Insomnia.

**Insomnia**

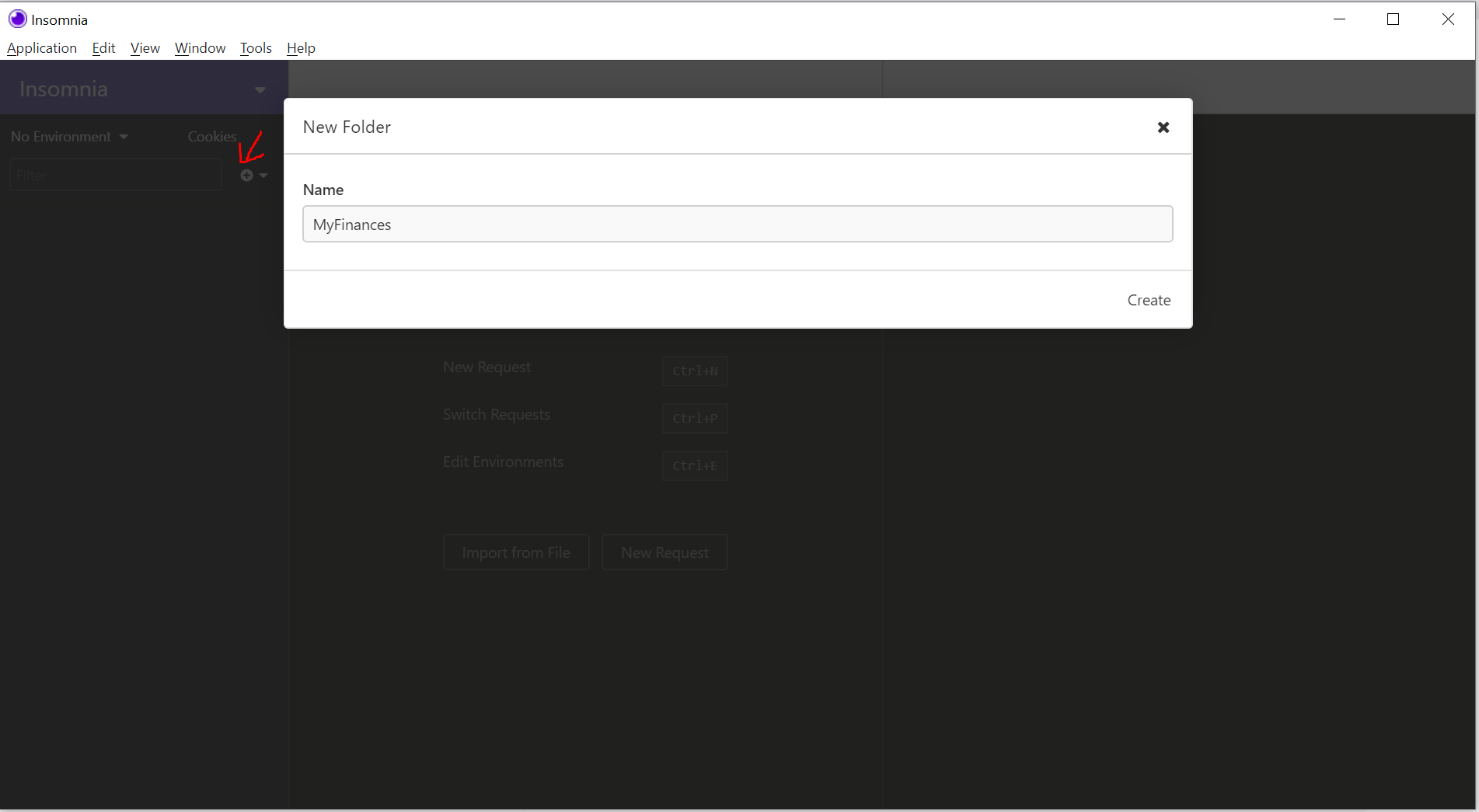


<https://insomnia.rest/>

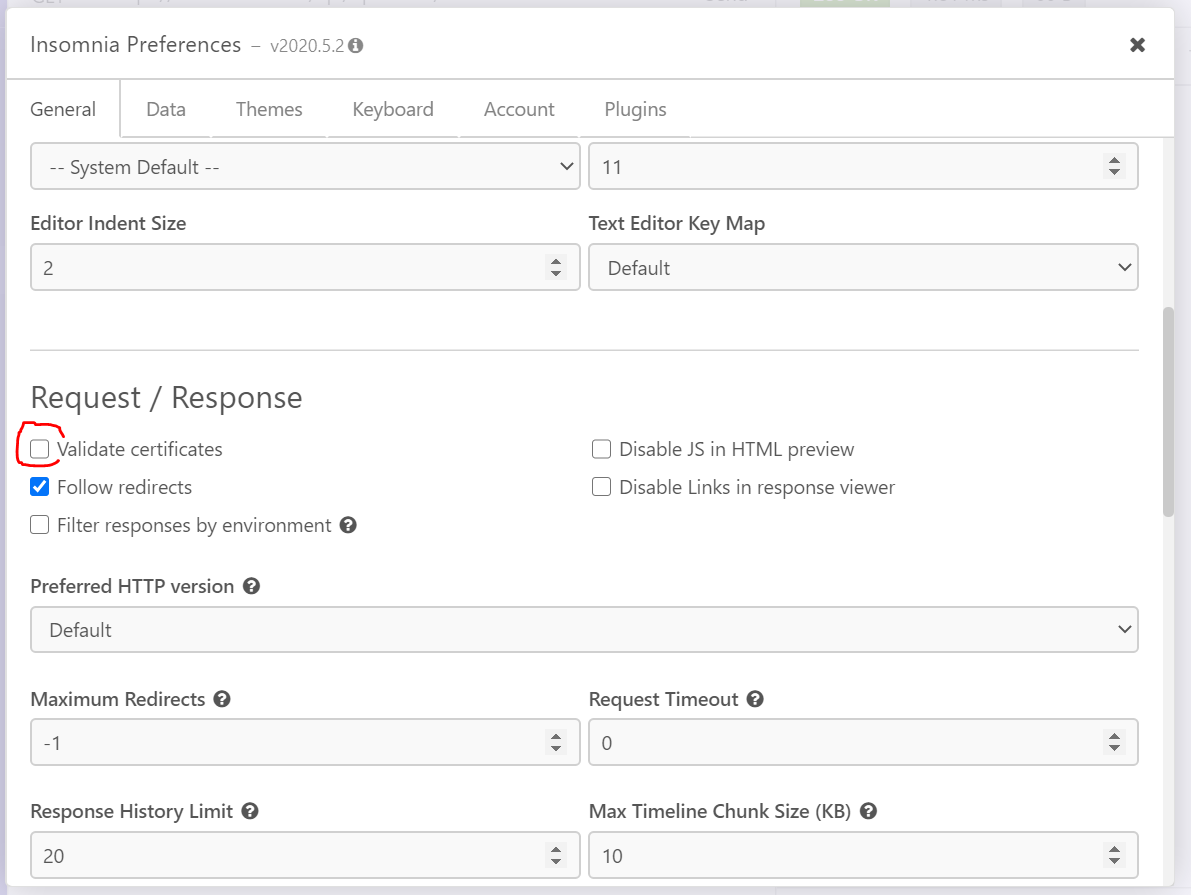


**Konfiguracja aplikacji**

Na początek dodajemy nowy folder, w którym będziemy przechowywać wszystkie request’y dla danej aplikacji.



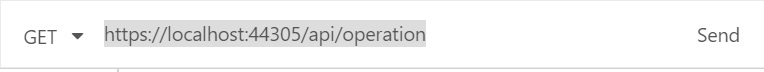
Co ważne w opcjach programu wyłączamy walidację SSL

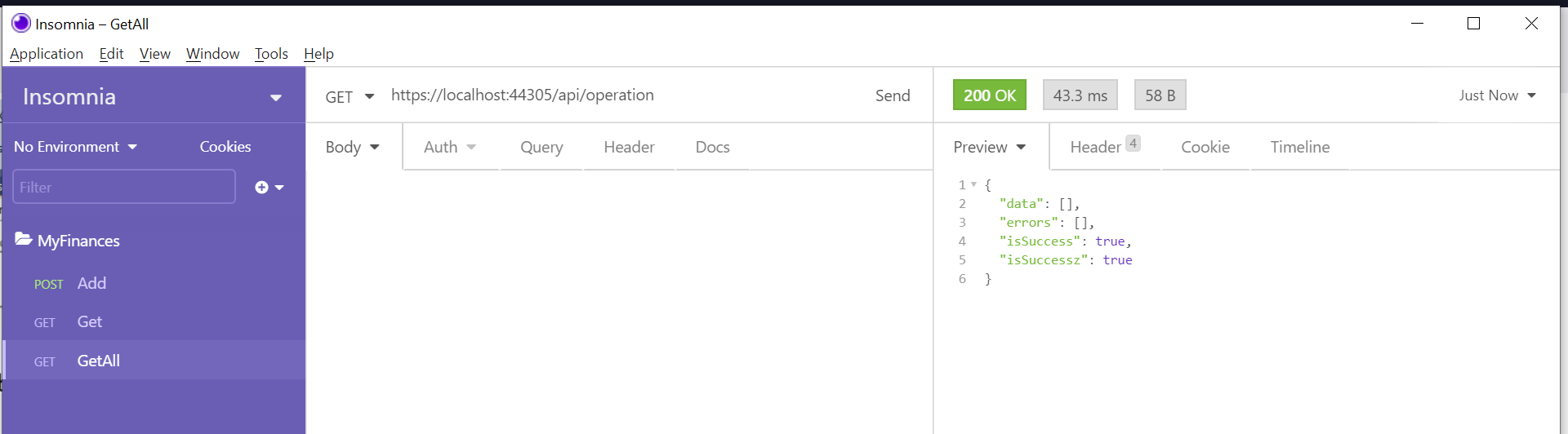


**Testujemy naszą aplikację**

Zakładamy katalog MyFinancies

Testujemy GetAll, który pobierze wszystkie wpisy.



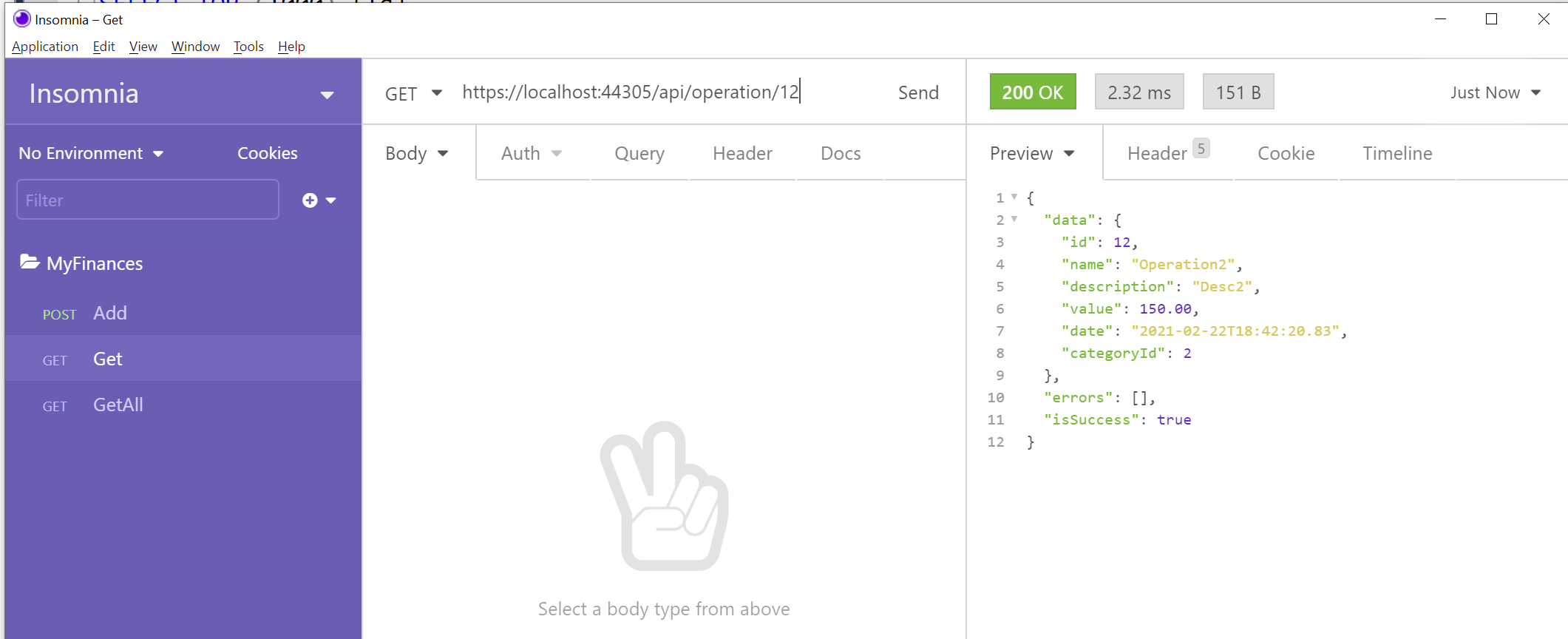


|  |
| --- |
| {  "data": null,  "errors": [],  "isSuccess": true,  } |

**Testujemy akcję Get(int id)**

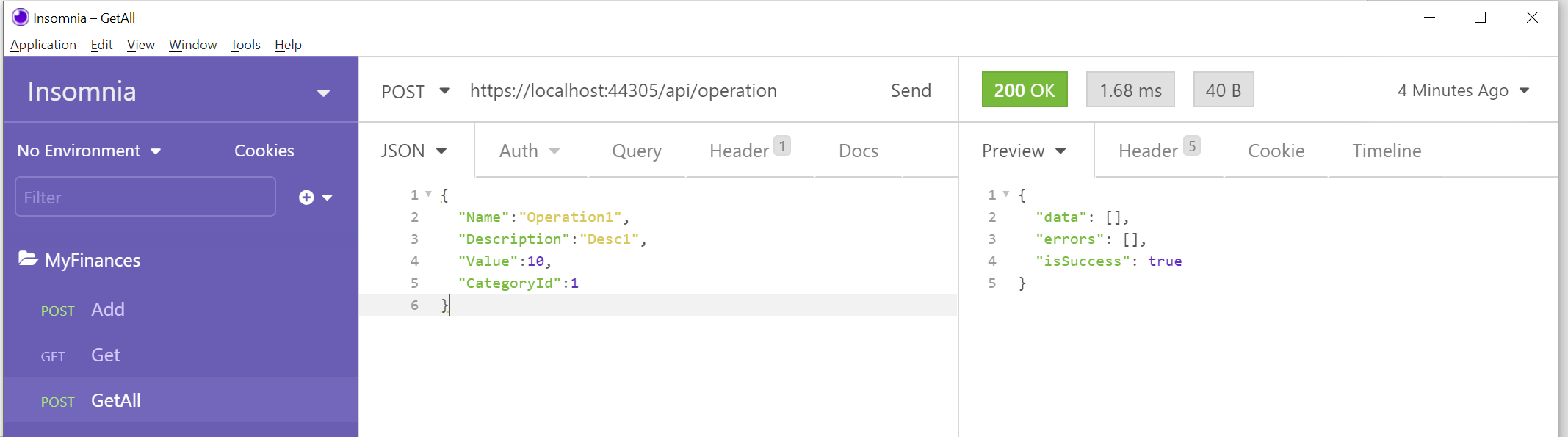
W kontrolerze drobna zmiana. Wprowadzamy zabezpieczenie powodujące, że metoda ToDao()  
się nie wykona w przypadku nulla, i nie spowoduje powstanie wyjątku.

|  |
| --- |
| **[HttpGet("{id}")]**  public DataResponse<OperationDto> Get(int id)  {  var responce = new DataResponse<OperationDto>();  try  {  responce.Data = \_unitOfWork.Operation.Get(id)?.ToDto();  } |



|  |
| --- |
| {  "data": {  "id": 12,  "name": "Operation2",  "description": "Desc2",  "value": 150.00,  "date": "2021-02-22T18:42:20.83",  "categoryId": 2  },  "errors": [],  "isSuccess": true  } |

**Testujemy akcję Add()**



Treść Json’a

|  |
| --- |
| {  "Name":"Operation1",  "Description":"Desc1",  "Value":10,  "CategoryId":1  } |

Odpowiedź

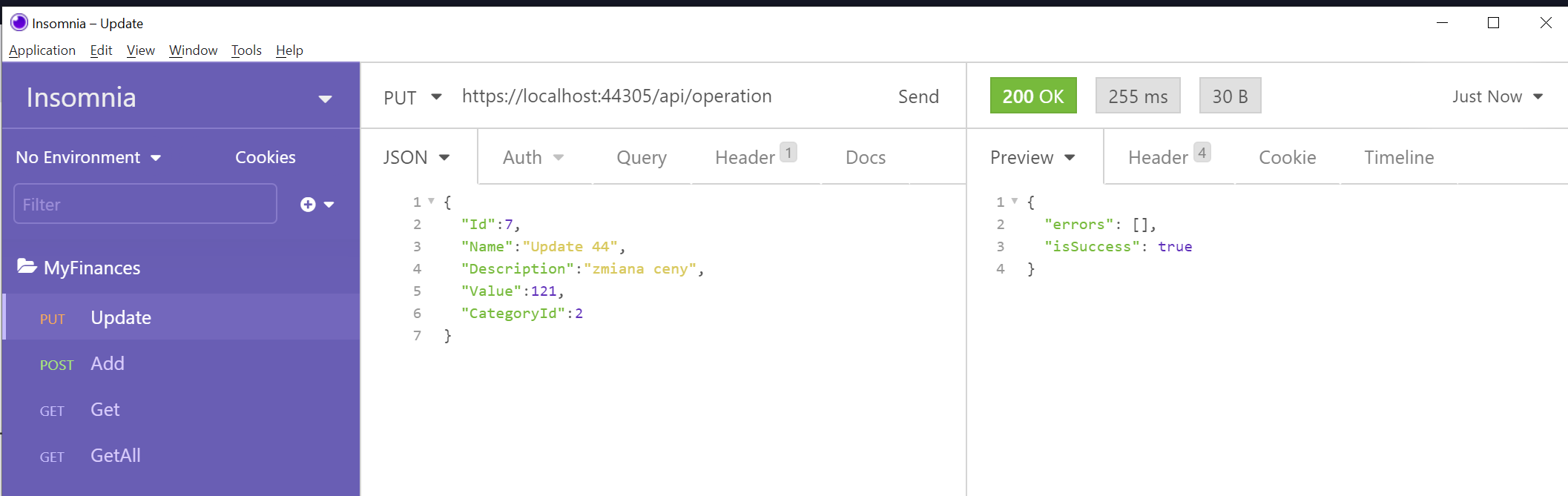
|  |
| --- |
| {  "data": 13,  "errors": [],  "isSuccess": true  } |

Drobna modyfikacja kontrolera, w poprzedniej wersji w odpowiedzi nie było numeru id

|  |
| --- |
| **[HttpPost]**  public DataResponse<int> Add(OperationDto operationDto )  {  var responce = new DataResponse<int>();  try  {  var operation = operationDto.ToDao();  \_unitOfWork.Operation.Add(operation);  \_unitOfWork.Complete();  // tutaj jest już znane id dodanego rekordu  responce.Data = operation.Id;  }  catch (Exception ex)  {  // logowanie do pliku ...  responce.Errors.Add(new Error(ex.Source, ex.Message));  }  return responce;  } |

**Testujemy akcję Update()**

a



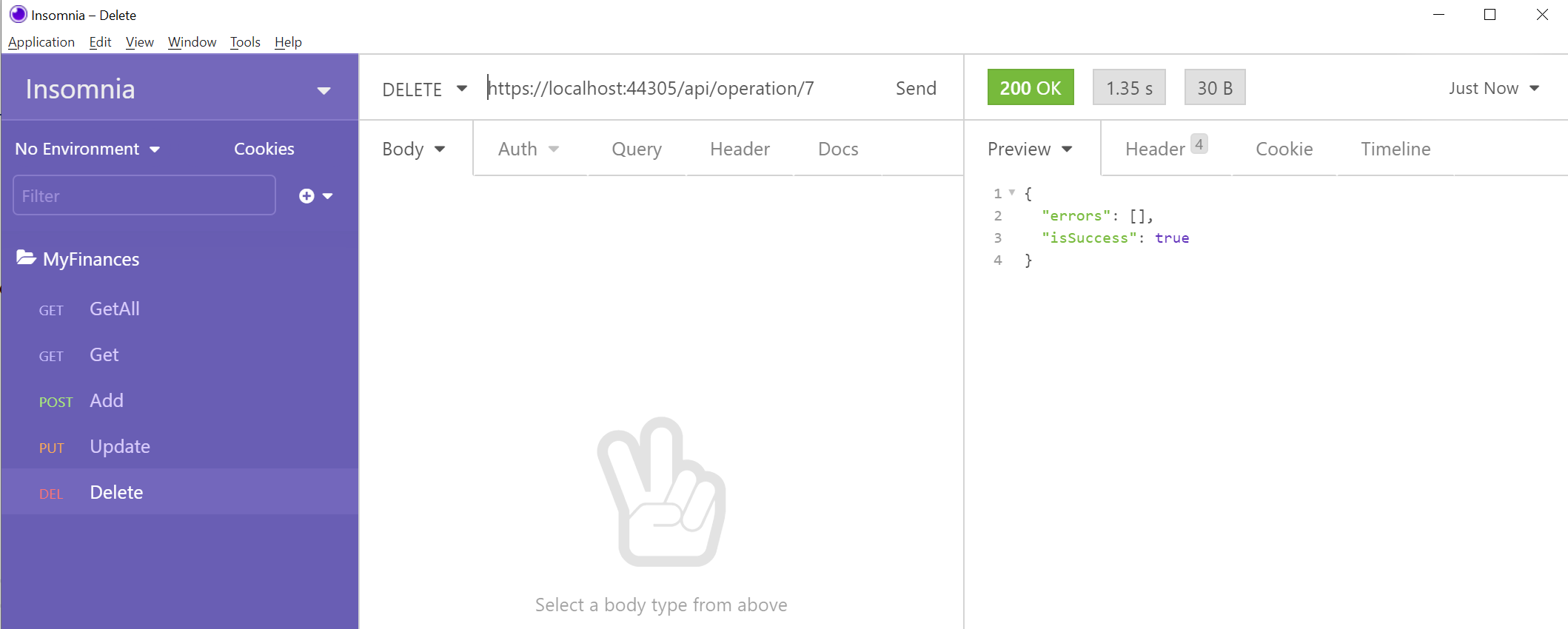
Treść Json’a

|  |
| --- |
| {  "Id":7,  "Name":"Update 44",  "Description":"zmiana ceny",  "Value":121,  "CategoryId":2  } |

Odpowiedź serwera

|  |
| --- |
| {  "errors": [],  "isSuccess": true  } |

**Testujemy akcję Delete()**



Odpowiedź z serwera

|  |
| --- |
| {  "errors": [],  "isSuccess": true  } |