**Projektowanie aplikacji biznesowych (PAB) – laboratorium nr 3**

# *YAML, Projektowanie API – Swager Editor*

## Zadania (*YAML*):

• specyfikacja: https://yaml.org/

• https://docs.ansible.com/ansible/latest/reference_appendices/YAMLSyntax.html

• *YAML syntax example*, *YAML* vs *JSON*, zastosowania *YAML*:

https://www.redhat.com/en/topics/automation/what-is-yaml

• typy danych *YAML: https://learn.getgrav.org/16/advanced/yaml*

• *JSON* to *YAML converter:* https://www.json2yaml.com/

• *YAML Schema*: https://asdf-standard.readthedocs.io/en/1.6.0/schemas/yaml_schema.html

| Zadanie 3.1:<br>Zapoznać się z następującymi zagadnieniami dotyczącymi formatu YAML:<br>• składnia *YAML*, komentarze,<br>• listy, elementy list, słowniki,<br>• typy danych zawartości *YAML'a*, możliwe wartości *Boolean*,<br>• wieloliniowe wartości, podział \| i >.<br>Napisany na poprzednich zajęciach *JSON* z rozkładem jazdy pociągów przekonwertować na *YAML* za pomocą poniższego narzędzia online.<br>Wskazać wady i zalety obu formatów. |
|---|
| https://www.json2yaml.com/ |

## Początek laboratorium:

• przejść pod adres https://editor.swagger.io/,

• zapoznać się z zawartością przykładowej *dokumentacji API – Swagger Petstore*,

• usunąć całą zawartość, wkleić zawartość pliku *Countries API Lab003 start.yaml*,

• przećwiczyć skróty klawiaturowe na zaznaczonym obszarze: *Tab* oraz *Shift* + *Tab*.

## Zadania (*Swagger Editor*):

• https://dane.gov.pl/media/ckeditor/2020/05/29/standard-api.pdf

(rekomendacje nazewnictwa zasobów, znak / na końcu *URL*, kody *HTTP*, zasada bezstanowości)

• specyfikacja *OpenAPI*: https://swagger.io/specification/

• przewodnik po *OpenAPI*: https://swagger.io/docs/specification/about/

• blog o projektowaniu REST API: https://www.mscharhag.com/p/rest-api-design

Zadanie 3.2:

Napisać (w *components* → *schemas*) definicje modelu *Country* reprezentującego kraje. Nazewnictwo po angielsku.

*Kraje* mają następujące właściwości (wszystkie oprócz *id* są wymagane):

– *id*, liczba całkowita, przykładowa wartość: `1`,

– *nazwa*, ciąg znaków, max. długość 50 znaków, przykładowa wartość: `Name`,

– *kod* (ISO 3166), ciąg znaków, max. długość 3 znaki, przykładowa wartość: `COD`,

– powierzchnia całkowita (w km²), liczba całkowita większa od 0, przykładowa: `100000`,

– *język*, ciąg znaków, max. długość 50 znaków, przykładowa wartość: `language`.

https://swagger.io/docs/specification/data-models/
https://swagger.io/docs/specification/data-models/keywords/
https://swagger.io/specification/#schema-object

Simple Model
Model with Example
Data Types


Zadanie 3.3:

W kolejnych zadaniach zaprojektować i napisać ścieżki (*endpoint'y*) standardowych operacji *CRUD'owych* dla krajów. Projektować *API* według wymagań *2 poziomu modelu dojrzałości Richardsona*. Stosować dobre praktyki odnośnie projektowania *REST API* i *API internetowych*.

Uwzględnić:

– *opis endpoint'u*,

– *tagi*,

– parametry wejściowe żądania (nazwa, opis, typ, format, zakresy, itp.),

– zawartość żądania (referencja do danego schematu),

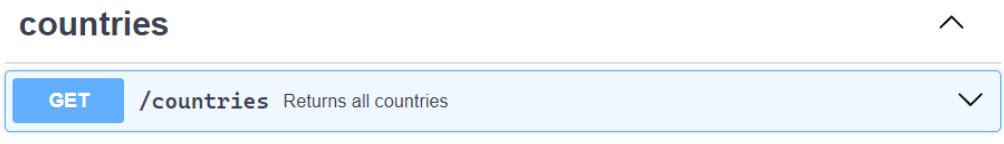
– odpowiedź żądania (możliwe przypadki [poznane na AI1] – kody *HTTP*, opis, zawartość).

https://restfulapi.net/resource-naming/
https://www.mscharhag.com/p/rest-api-design
Basic CRUD operations
https://www.mscharhag.com/api-design/http-status-codes

https://swagger.io/docs/specification/paths-and-operations/
https://swagger.io/docs/specification/grouping-operations-with-tags/
https://swagger.io/specification/#paths-object
Paths Object
Path Item Object
Operation Object
Reference Object Example

Zadanie 3.4:

Zaprojektować i napisać ścieżkę (*endpoint*) dla operacji pobrania wszystkich krajów.



Zadanie 3.5:
Zaprojektować i napisać ścieżkę (*endpoint*) dla operacji pobrania danego kraju.
Napisać (w *components* → *schemas*) definicje parametru ścieżki *countryId* do wykorzystywania w tym i niektórych następnych zadaniach.

https://swagger.io/docs/specification/components/
Components Example

```
components:
  parameters:
    countryId:
      name: countryId
      in: path
      description: ID of country to use
      required: true
      schema:
        type: integer
```

https://swagger.io/docs/specification/using-ref/
Using $ref

```
    parameters:
      - $ref: '#/components/parameters/countryId'
```

Zadanie 3.6:
Zaprojektować i napisać ścieżkę (*endpoint*) dla operacji dodania nowego kraju.

–

Zadanie 3.7:
Zaprojektować i napisać ścieżkę (*endpoint*) dla operacji edycji danego kraju.

–

Zadanie 3.8:
Zaprojektować i napisać ścieżkę (*endpoint*) dla operacji usunięcia danego kraju.

–

## Kontynuacja laboratorium:

• włączyć panel zarządzania *XAMPP* oraz uruchomić moduły *Apache* oraz *MySQL*. Następnie kliknąć przycisk *Admin* przy *MySQL*, w celu przejścia do panelu *phpMyAdmin*.

• pobrać na pulpit archiwum `Lab003_PAB_start.zip`, w którym umieszczony jest projekt startowy do wykonania zadań oraz rozpakować to archiwum,

• przejść do rozpakowanego folderu oraz uruchomić skrypt `start.bat` 2x kliknięciem.

## Zadania (*Swagger* i *Laravel*):

• https://laravel.com/docs/10.x/telescope

• https://laravel.com/docs/10.x/eloquent-resources

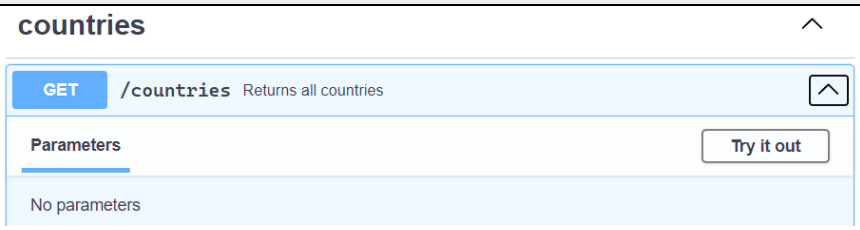
• https://laravel.com/docs/10.x/validation#form-request-validation

Zadanie 3.9:
Otworzyć terminal *cmd (Command Prompt)* w *VSCode*. Następnie uruchomić aplikację.
W przeglądarce przejść do *Telescope* oraz pozostawić go otwartym.

```
php artisan serve
```

http://localhost:8000/telescope/requests

Zadanie 3.10:
Za pomocą ***Try it out*** każdego z *endpoint'ów* wykonać żądania wszystkich możliwych rodzajów (z przewidywanym sukcesem, z przewidywanym niepowodzeniem: brak odnalezienia zasobu, błędy walidacji).




Zadanie 3.11:
Po wykonaniu żądań z poprzedniego zadania, przeglądnąć ich szczegóły w *Telescope*.

http://localhost:8000/telescope/requests

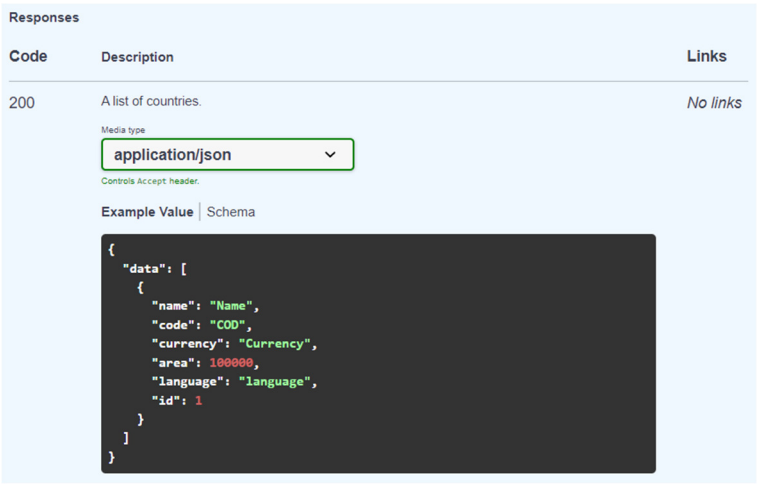
Zadanie 3.12:
Wskazać właściwość kraju, która niepotrzebnie nadmiarowo pojawiała się w niektórych operacjach, oraz wskazać te operacje.

–

Zadanie 3.13:
Dostosować *dokumentację API* do tego zaprogramowanego w *Laravel'u*.
Uwzględnić zmiany zaproponowane w poprzednim zadaniu.
Wykorzystać schemat *Country* do utworzenia 4 nowych schematów:
• *CountryResource* do używania w miejscach określania struktury zwracanego pojedynczego kraju,
• *CountryCollection* do używania w miejscach określania struktury zwracanej tablicy krajów,
• *StoreCountryRequest* do używania w miejscach określania struktury obiektu nowego dodawanego kraju,
• *UpdateCountryRequest* do używania w miejscach określania struktury obiektu edytowanego kraju.
W przypadku podobieństwa zawartości schematów można wykorzystać *allOf*.

https://swagger.io/docs/specification/data-models/oneof-anyof-allof-not/
allOf

Zadanie 3.14:
Po wykonaniu wcześniejszego zadania, znów spróbować ***Try it out*** każdego z *endpoint'ów i sprawdzić czy* w *Responses* → *Example Value* oraz *Request Body* → *Example Value* postać przykładowych *JSONów* odpowiada tym zwracanym przez aplikację.



* – zadania/podpunkty do samodzielnego dokończenia/wykonania,

* – zadania/podpunkty dla zainteresowanych.

Po zakończonym laboratorium należy skasować wszystkie pobrane oraz utworzone przez siebie pliki z komputera w sali laboratoryjnej.

Wersja pliku: v1.0

## Inne:*

https://docs.docker.com/compose/compose-file/03-compose-file/

https://docs.docker.com/compose/compose-file/07-volumes/

https://docs.spring.io/spring-boot/docs/current/reference/html/application-properties.html

https://dev.to/fullstackhacker/spring-boot-yaml-example-1kb2