Projektowanie aplikacji biznesowych (PAB) – laboratorium nr 5

Swagger Editor, Laravel REST API – relacja m-t-m

Początek laboratorium:

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

• usunąć całą zawartość, wkleić zawartość pliku *Movies API Lab005 start.yaml*,

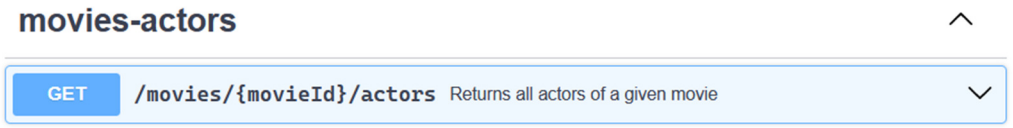
• zapoznać się z zawartością pliku oraz wygenerowaną zawartością (strona prawa).

Zadania (*Swagger Editor*):

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

Zadanie 5.1: Zapoznać się z artykułem dotyczącym relacji *many-to-many*. Jako przykład przedstawiono powiązanie pomiędzy *użytkownikami* i *grupami*. Poruszone są dwa aspekty: • kwestia przechowywania dodatkowych informacji o powiązaniu, • kwestia możliwej liczby istniejących powiązań pomiędzy dwoma konkretnymi rekordami.
https://www.mscharhag.com/api-design/rest-many-to-many-relations Introduction Modeling sub-resources and GET operations Adding and Removing users
Zadanie 5.2: W kolejnych zadaniach zaprojektować i napisać ścieżki (*endpoint'y*) operacji dotyczących *aktorów* występujących *w filmach* (*relacja m-t-m*): • zarządzanie *obsadą aktorów z poziomu filmów*, • dany *aktor* może występować w wielu filmach, • w danym filmie może występować wielu aktorów, • aktor w danym filmie ma tylko jedną rolę, • o roli aktora nie są przechowywane dodatkowe informacje. Oczywiście: • projektować *API* według wymagań *2 poziomu modelu dojrzałości Richardsona*, • stosować dobre praktyki odnośnie projektowania *REST API* i *API internetowych*, • uwzględnić wszystkie elementy jak w zadaniu 3.3 z laboratorium nr 3, • dostosować *API* pod używanie definicji modeli przesyłanych z/do *Laravel'a*.
–

Zadanie 5.3:
Zaprojektować i napisać ścieżkę (*endpoint*) dla operacji pobrania wszystkich aktorów występujących w danym filmie (obsada filmu).



Zadanie 5.4:
Zaprojektować i napisać ścieżkę (*endpoint*) dla dodania aktora do obsady filmu (tzn. dany aktor będzie występować w filmie, w danym filmie ma tylko jedną rolę).
Zadbać, żeby operacja była *idempotentna*.

https://www.mscharhag.com/api-design/http-idempotent-safe
Idempotent HTTP methods

https://restfulapi.net/idempotent-rest-apis

Zadanie 5.5:
Zaprojektować i napisać ścieżkę (*endpoint*) dla usunięcia aktora z obsady filmu (tzn. dany aktor traci rolę w filmie).
Zadbać, żeby operacja była *idempotentna*.

–

Zadanie 5.6:
Wykonać eksport dokumentacji:
• *File → Save as YAML*, który zapisać jako *Movies API Lab005.yaml*.

–

Kontynuacja laboratorium:

• pobrać na pulpit archiwum `Lab005_PAB_start.zip`, w którym umieszczony jest projekt startowy do wykonania zadań oraz rozpakować to archiwum,

~~• przejść do rozpakowanego folderu oraz w przypadku posiadania innych ustawień niż domyślne (np. połączenia z bazą), wykonać ich zmianę w `.env.example` oraz `start.…`,~~

• uruchomić skrypt `start.bat` (*Windows*, 2x kliknięciem) lub `start.sh` (inne systemy, przez polecenie `bash start.sh`),

• wyświetlić zawartość bazy danych *SQLite* z pliku *database.sqlite* za pomocą np. *DBeaver'a*.

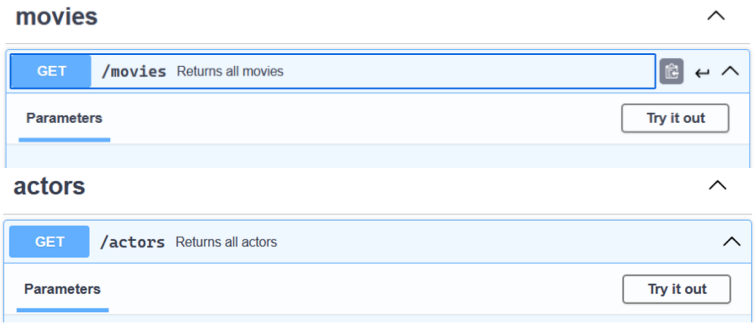
Zadania (*Laravel, m-t-m*):

• https://laravel.com/docs/10.x/eloquent-relationships#many-to-many

Zadanie 5.7:
Otworzyć terminal *cmd (Command Prompt)* w *VSCode*.
Następnie uruchomić aplikację.

```
php artisan serve
```

Zadanie 5.8:
Za pomocą ***Try it out*** *endpoint'ów* dla pobierania wszystkich hoteli i aktorów zapoznać się z obecną zawartością bazy danych.



Zadanie 5.9:
Wykonać następujące etapy programowania *kontrolera filmów*, tak, aby jego działanie pokrywało się 1:1 z napisaną wcześniej dokumentacją w *.yaml*:
• dodać nową funkcję **`getMovieCast()`** w *MovieController.php* oraz zaprogramować ją żeby zwracała obsadę danego filmu,
• dodać nową funkcję **`addActorToMovieCast()`** w *MovieController.php* oraz zaprogramować ją żeby dodawała aktora do obsady filmu, natomiast jeśli aktor jest już w niej, to nie jest dodawany drugi raz,
• dodać nową funkcję **`removeActorFromMovieCast()`** w *MovieController.php* oraz zaprogramować ją żeby usuwała aktora z obsady filmu,
• ustawić ścieżki dla nowych funkcji w pliku *routes/api.php*,
• w dokumentacji *.yaml* dopisać nazwy nowych funkcji jako **`operationId`** danej ścieżki.
Używać operacje na metodzie **`actors()`** *modelu Movie*.

```
Route::get('movies/{movie}/actors', [MovieController::class, 'getMovieCast']);
...
...
```

https://swagger.io/docs/specification/paths-and-operations operationId

https://laravel.com/docs/10.x/eloquent-relationships#attaching-detaching

https://stackoverflow.com/questions/62104188/laravel-eloquent-attach-vs-syncwithoutdetaching

Zadanie 5.10:
Za pomocą ***Try it out*** *endpoint'ów* dotyczących *obsady filmów* wykonać następujące *żądania*, aby sprawdzić działanie zaprogramowanych wcześniej funkcji:
• pobrać całą obsadę filmu *Skyfall*,
• pobrać całą obsadę filmu *Spectre*,
• usunąć *Judi Dench* z obsady *Spectre*,
• usunąć ponownie *Judi Dench* z obsady *Spectre*,
• pobrać całą obsadę filmu *Spectre*,
• dodać *Christoph'a Waltz'a* do obsady Spectre,
• dodać *Léa Seydoux* do obsady Spectre,
• dodać *Naomie Harris* do obsady Spectre,
• dodać 5 razy *Ben'a Whishaw'a* do obsady Spectre,
• pobrać całą obsadę filmu *Spectre*.

–

Zadanie 5.11: *

Powrócić do *Swagger Editor'a*, zmodyfikować schemat *MovieResource*, tak aby znajdowała się w nim tablica aktorów (aktorzy z obsady filmu).

–

Zadanie 5.12: *

W definicji ścieżki pobrania wszystkich filmów ustalić opcjonalny parametr w ciągu zapytania (*Query parameter*) o nazwie `includeActors`, w którym użytkownik może przekazać wartość *true* lub *false*.

https://swagger.io/docs/specification/describing-parameters/#query-parameters
https://swagger.io/docs/specification/data-models/data-types/#boolean

https://api.gov.au/sections/naming-conventions.html
Query Parameter Names

Zadanie 5.13: *

Wykonać następujące etapy:
• uzupełnić funkcję `toArray()` w *MovieResource.php* o tablicę aktorów (jeśli jest załadowana),
• uzupełnić funkcję `index()` w *MovieController.php* o zwracanie filmów z załączonymi aktorami, jeśli użytkownik wykonał żądanie zawierające parametr (z wartością) `includeActors=true`.

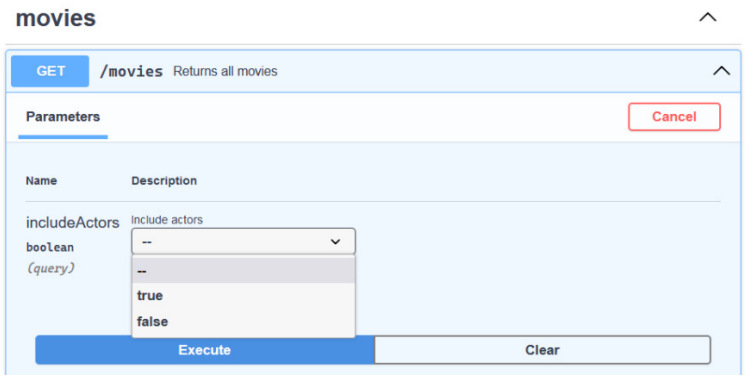
```
'actors' => ActorResource::collection($this -> whenLoaded('actors')),
```

https://laravel.com/docs/11.x/requests#retrieving-boolean-input-values

Zadanie 5.14: *

Sprawdzić działanie *endpoint'u* pobrania wszystkich filmów w ***Try it out***.

http://localhost:8000/api/movies
http://localhost:8000/api/movies?includeActors=true
http://localhost:8000/api/movies?includeActors=false



Zadanie 5.15: *

Stwierdzić czy w poprzednim zadaniu wystąpił *problem N+1*.
Jeśli nie wystąpił, to zmienić „*pogorszyć*" kod tak żeby występował (w celu zasymulowania sytuacji).
Jeśli wystąpił, to „*poprawić*" kod tak żeby już nie występował.

https://restfulapi.net/rest-api-n-1-problem
https://laravel-news.com/laravel-n1-query-problems

Zadanie 5.16: * Przygotować (ręcznie) plik *lab6.rest*, w którym napisać wszystkie w którym napisać wszystkie możliwe żądania (oraz ich wszystkie możliwe rezultaty) do *API aktorów i filmów* (inne filmy z serii *007* i aktorzy w nich występujący). Rozszerzenie *REST Client dla VSCode* lub wbudowana obsługa *.rest* w *PhpStorm*.
–
Zadanie 5.17: * Zaprojektować i zaprogramować funkcjonalność ustalenia całej obsady filmu „*na raz*” (w tym całkowite nadpisanie obecnej obsady).
–
Zadanie 5.18: * Zapoznać się z podziałem na „*bezpieczne*” metody *HTTP*.
https://www.mscharhag.com/api-design/http-idempotent-safe Safe HTTP methods

* – 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://swagger.io/docs/specification/links