Projektowanie aplikacji biznesowych (PAB) – laboratorium nr 6

Swagger Editor, Laravel REST API, JWT

Początek laboratorium:

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

• usunąć całą zawartość, wkleić zawartość pliku *Countries API & JWT Lab006 start.yaml*,

• pobrać na pulpit archiwum `Lab006_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 (*Swagger Editor, Laravel*):

• https://swagger.io/blog/code-first-vs-design-first-api

Zadanie 6.1: Uruchomić serwer deweloperski *php*. Sprawdzić ustawienie ścieżek związanych z uwierzytelnianiami i autoryzacją użytkowników w pliku *routes\api.php*.

`POST api/auth/forgot-password ..... AuthController@forgotPassword`
`POST api/auth/login ..... AuthController@login`
`POST api/auth/logout ..... AuthController@logout`
`POST api/auth/me ..... AuthController@me`
`POST api/auth/refresh ..... AuthController@refresh`
`POST api/auth/reset-password ..... AuthController@resetPassword`

Zadanie 6.2: Przejść do *Swagger Editor*. Za pomocą ***Try it out*** sprawdzić działanie *endpoint'u* „*login*". Uzyskać *token'y* dla *admin'a Jana* i *user'a Marty* (oraz skopiować je do pliku tekstowego). • email: `jan@email.com`, hasło: `1234`, • email: `marta@email.com`, hasło: `1234`. Dla *endpoint'ów* operacji *CRUD* krajów w aplikacji obowiązuje obecnie: • tylko „*zalogowani*" (w znaczeniu „przedstawiający się ważnym *token'em*") użytkownicy mogą odczytywać dane krajów/dane danego kraju, • tylko użytkownicy z rolą „*admin*" mogą wykonywać operacje dodawania/modyfikacji/usuwania danego kraju.
–

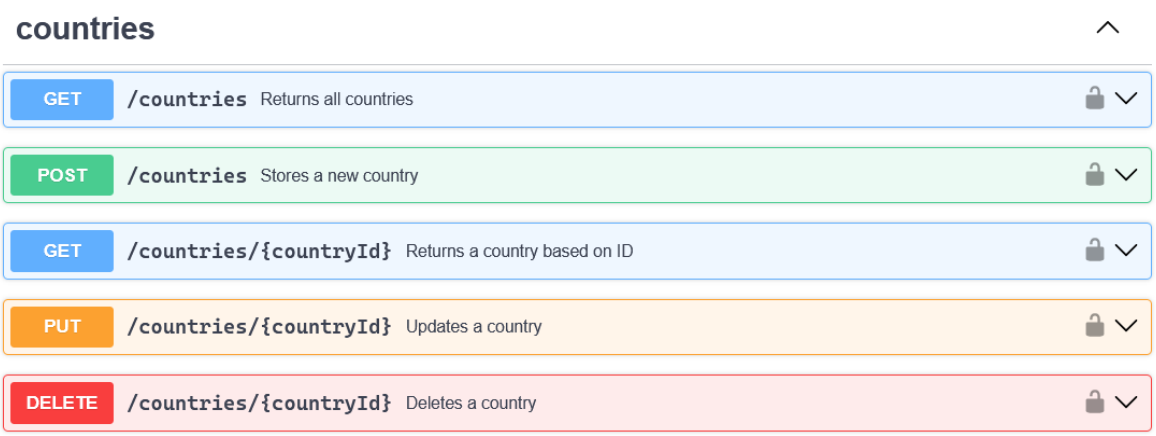
Zadanie 6.3:

Dodać (w *components* → *securitySchemes*) definicję dla schematu bezpieczeństwa „*bearerAuth*”.

Uzupełnić dokumentację *endpoint'ów* wszystkich operacji *CRUD* dotyczących krajów o stosowanie tego schematu.

https://swagger.io/docs/specification/v3_0/authentication/bearer-authentication#describing-bearer-authentication

Describing Bearer Authentication



Zadanie 6.4:

Dodać (w *components* → *responses*) definicję dla odpowiedzi o błędnym uwierzytelnieniu żądania.

Uzupełnić dokumentację *endpoint'ów* dla wszystkich operacji *CRUD* dotyczących krajów o możliwy przypadek odpowiedzi wykorzystującej tą definicję.

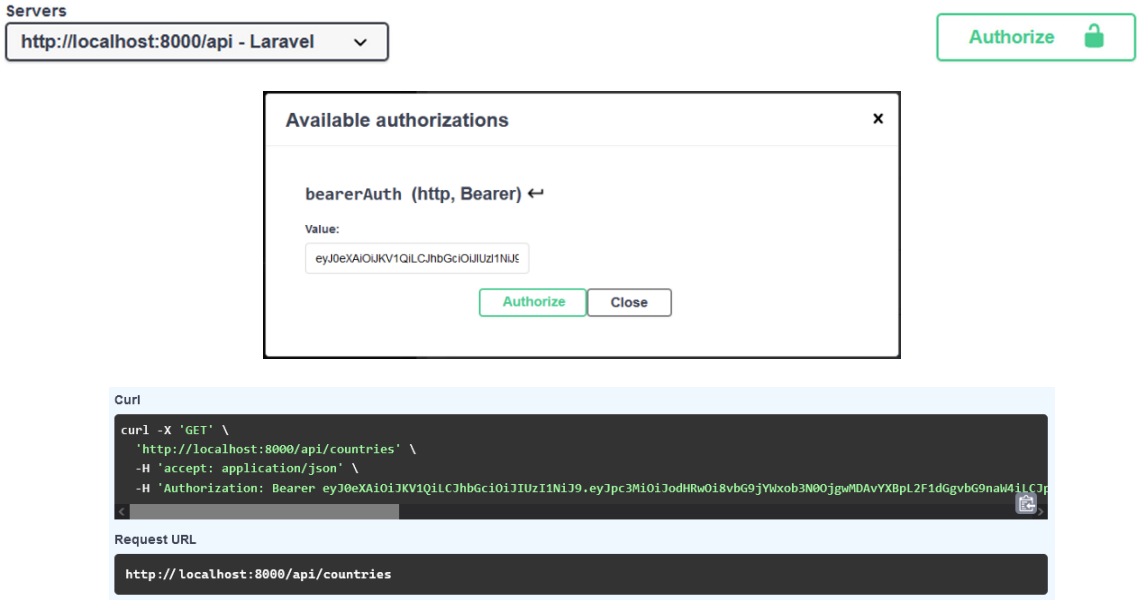
https://swagger.io/docs/specification/v3_0/authentication/bearer-authentication#401-response

401 Response

Zadanie 6.5:

Pobrać wszystkie kraje bez podawania *token'a*.

„*Zalogować*” Jana (zapamiętać jego *token*). Następnie pobrać wszystkie kraje.



Zadanie 6.6:
„*Wylogować*” Jana (zapomnieć jego *token*).
„*Zalogować*” Martę (wcześniej utworzonym *token’em*).
Następnie spróbować dodać nowy kraj.

Curl

```
curl -X 'POST' \
  'http://localhost:8000/api/countries' \
  -H 'accept: application/json' \
  -H 'Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpc3MiOiJodHRwOi8vbG9jYWxob3N0OjgwMDAvYXBpL2F1dGgvbG9naW4iLCJp
  -H 'Content-Type: application/json' \
  -d '{
  "name": "Name",
  "code": "COD",
  "currency": "Currency",
  "area": 100000,
  "language": "language"
}'
```

Request URL

```
http://localhost:8000/api/countries
```

Server response

Code	Details
403 *Undocumented*	Error: Forbidden Response body

```
{
  "message": "This action is unauthorized.",
```

Zadanie 6.7:
Dodać (w *components* → *responses*) analogicznie do zadania 6.4 definicję dla odpowiedzi o błędnej *autoryzacji* żądania.
Uzupełnić dokumentację *endpoint’ów* dla wszystkich operacji *CRUD* dotyczących modyfikacji krajów o możliwy przypadek odpowiedzi wykorzystującej tą definicję.

403	This action is unauthorized	*No links*

Zadanie 6.8: *
Napisać *endpoint* dla operacji pobrania danych „*zalogowanego*” użytkownika, tak, aby pokrywał się z działaniem funkcji kontrolera `me()`.
Dodać (w *components* → *schemas*) definicję dla zwracanego obiektu użytkownika.

POST `/auth/me` Get the authenticated user

Zadanie 6.9: *
Napisać *endpoint* dla operacji „*wylogowania*” użytkownika (unieważnienie token’a), tak, aby pokrywał się z działaniem funkcji kontrolera `logout()`.

POST `/auth/logout` Log the user out (Invalidate the token)

Zadanie 6.10: *
Napisać *endpoint* dla operacji „*odświeżania*” tokena (przypomnieć co robi ta operacja), tak, aby pokrywał się z działaniem funkcji kontrolera `refresh()`.

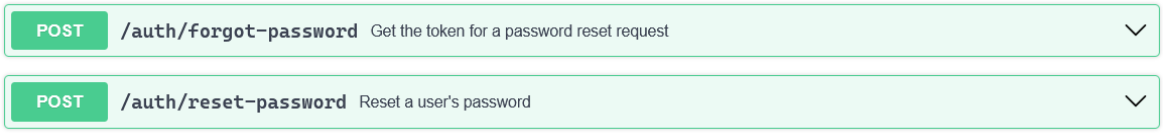
Zadania (*Swagger* i *Laravel*, resetowanie hasła):

Zadanie 6.11:
Napisać *endpoint'y* dla procesu *resetowania* (*przypominania*) *hasła*.
Z powodów technicznych akcja przygotowywania zawartości maila z linkiem oraz jego wysyłania zastąpiona jest wypisaniem samego *token'a* resetowania hasła w postaci logu na konsoli.

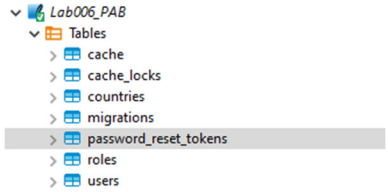
https://laravel.com/docs/11.x/passwords

Zadanie 6.12:
Napisać dokumentację *endpoint'u* obsługi żądania mającego na celu uzyskanie *token'a*, tak, aby pokrywał się z działaniem funkcji kontrolera `forgotPassword()`:
• metoda *POST*,
• ścieżka: …`/auth/forgot-password`,
• należy do grupy *auth* (w *tags*),
• żądanie nie potrzebuje uwierzytelniania,
• w wymaganym ciele żądania ma być *JSON* z *emailem użytkownika*, który chce dokonać zresetowania hasła,
• w przypadku:
– pozytywnym oczekiwany jest status 202 (*Accepted*), z pustym ciałem odpowiedzi,
– niepozytywnym wynikającym z nieodnalezienia użytkownika oczekiwany jest status 404,
– niepozytywnym wynikającym z braku/podania pustego lub w nieprawidłowej postaci adresu email, oczekiwany status to 422.

https://developer.mozilla.org/en-US/docs/Web/HTTP/Status/202



Zadanie 6.13:
Przetestować za pomocą (***Try it out***) oczekiwane sytuacje ustalone w poprzednim zadaniu. Po pomyślnej, sprawdzić zawartość tabeli *password_reset_tokens*. Odczytać *token* z logu w konsoli.



Zadanie 6.14: Napisać dokumentację *endpoint'u* obsługi żądania mającego na celu zrealizowanie zmiany hasła, tak, aby pokrywał się z działaniem funkcji kontrolera `resetPassword()`: • metoda *POST*, • ścieżka: …`/auth/reset-password`, • należy do grupy *auth* (w *tags*), • żądanie nie potrzebuje uwierzytelniania, • w wymaganym ciele żądania ma być *JSON* z: – otrzymanym wcześniej *token'em*, – *email'em użytkownika*, który chce dokonać zresetowania hasła, – nowym hasłem użytkownika (min. 3 znaki), • w przypadku: – pozytywnym oczekiwany jest status 200 (z wiadomością dla użytkownika), – niepozytywnym wynikającym z m.in podania błędnego *token'a*, oczekiwany jest status 400, – niepozytywnym wynikającym z braku/podania pustego lub w nieprawidłowej postaci wymaganych danych, oczekiwany status to 422.
–
Zadanie 6.15: Przetestować za pomocą (***Try it out***) oczekiwane sytuacje ustalone w poprzednim zadaniu. Po pomyślnej, sprawdzić zawartość tabeli *password_resets*. Następnie uzyskać nowy *token JWT* posługując się nowym hasłem.
–
Zadanie 6.16: * Zapoznać się ze statusem 409. Zaproponować sytuacje, gdzie można ustalić zwracanie tego statusu zamiast 422.
https://developer.mozilla.org/en-US/docs/Web/HTTP/Status/409

* – 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://auth0.com/blog/forbidden-unauthorized-http-status-codes

Don't let the client know...

An origin server that wishes to "hide" the current existence of a forbidden target resource MAY instead respond with a status code of 404 (Not Found).