Rok akademicki	Rodzaj studiów	Kierunek	Prowadzący	Grupa	Sekcja
2022/2023	Dzienne	IPpp	mgr inż. Dawid Jurczyński	2	4

Sprawozdanie z ćwiczenia REST API

Wykonawca ćwiczenia:

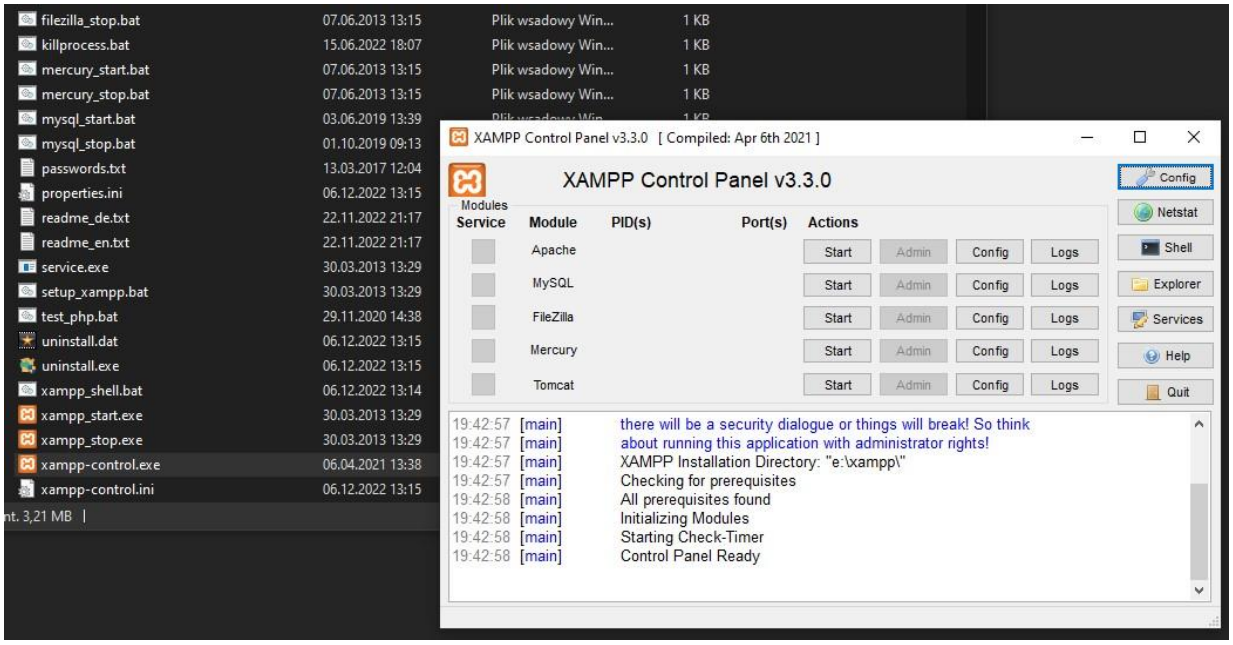
Kamil Klecz

Numer Studenta: 305373

Link do repozytorium GitHub:
https://github.com/Kamil-Klecz/Projekt-REST-API/tree/main

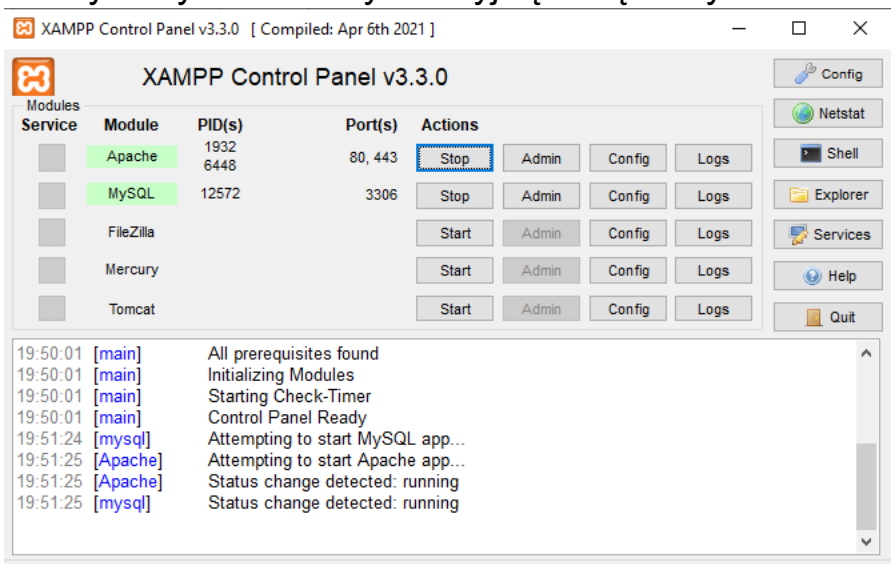
Etapy wykonywania zadania:

1. Zainstalować pakiet XAMPP



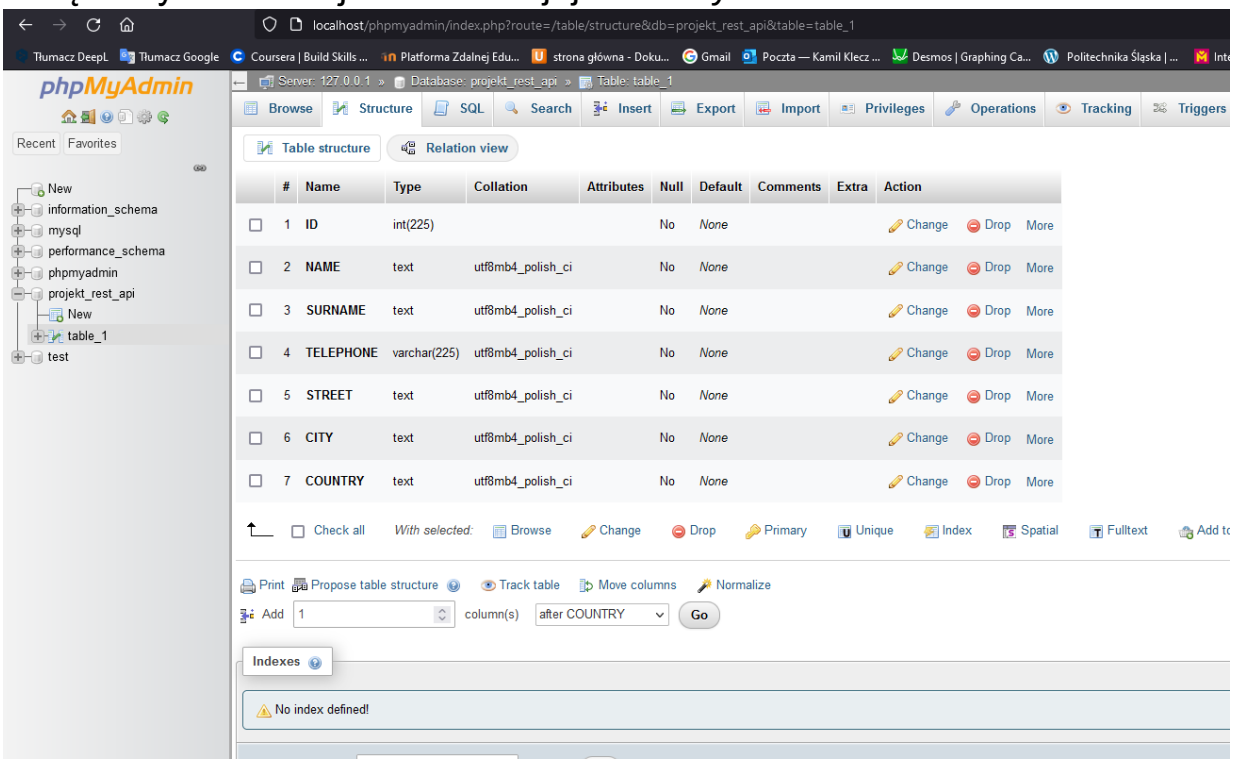
Pomyślnie zainstalowałem pakiet XAMPP bez żadnych komplikacji

2. Wykorzystać należy relacyjną bazę danych.



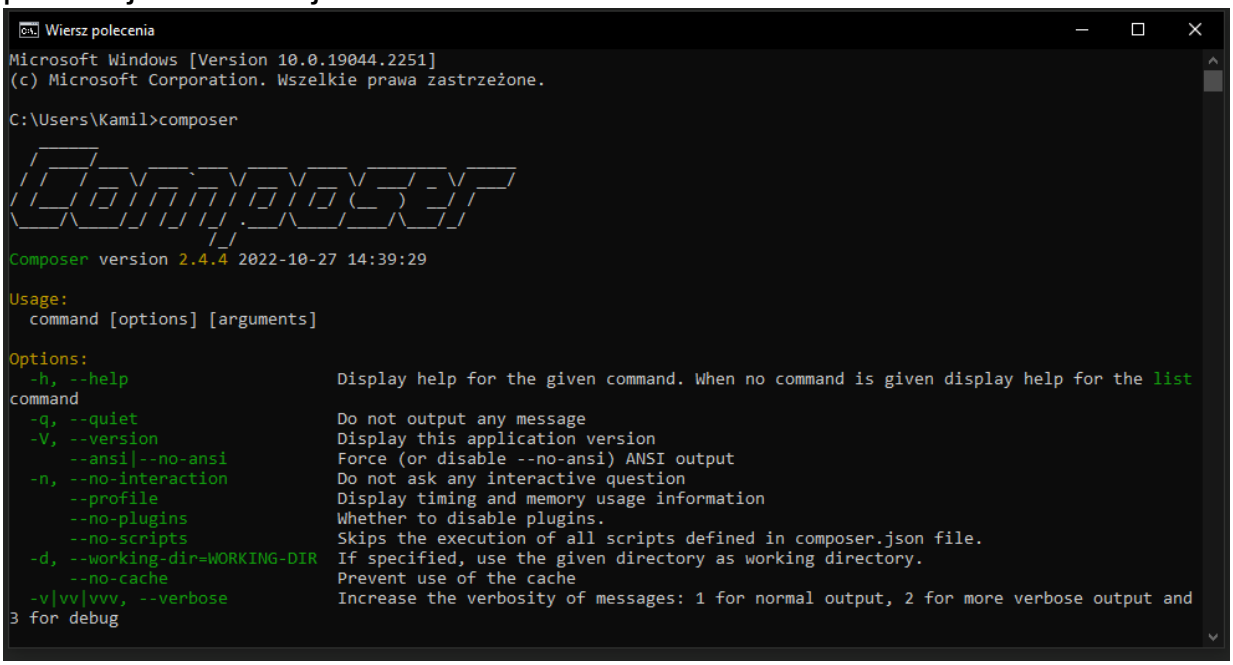
Włączyłem funkcję serwera PHP (Apache) oraz serwera SQL (MySQL) w programie XAMPP.

Po wejściu na http://localhost/phpmyadmin, utworzyłem nową relacyjną bazę danych i zainicjalizowałem jej kolumny



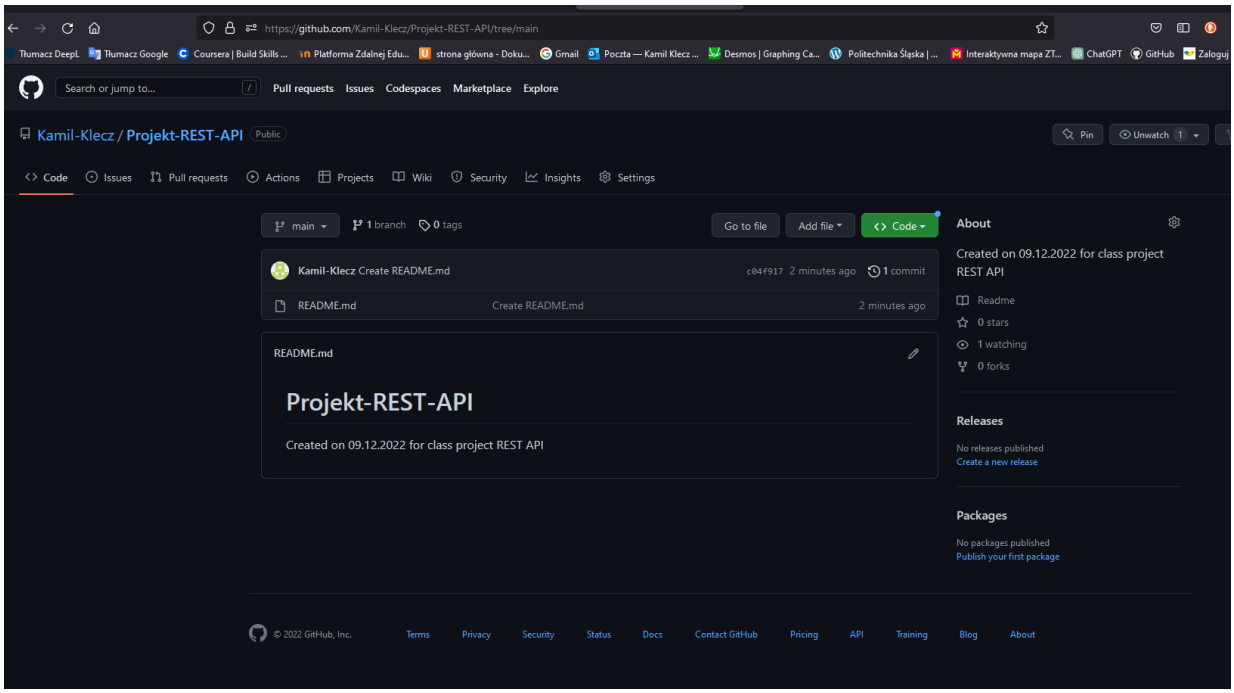
3. Zainstalować program Composer:

Pomyślnie zainstalowałem program Composer według instrukcji na stronie podanej w instrukcjach do zadania.



Nie napotkałem żadnych problemów podczas instalacji

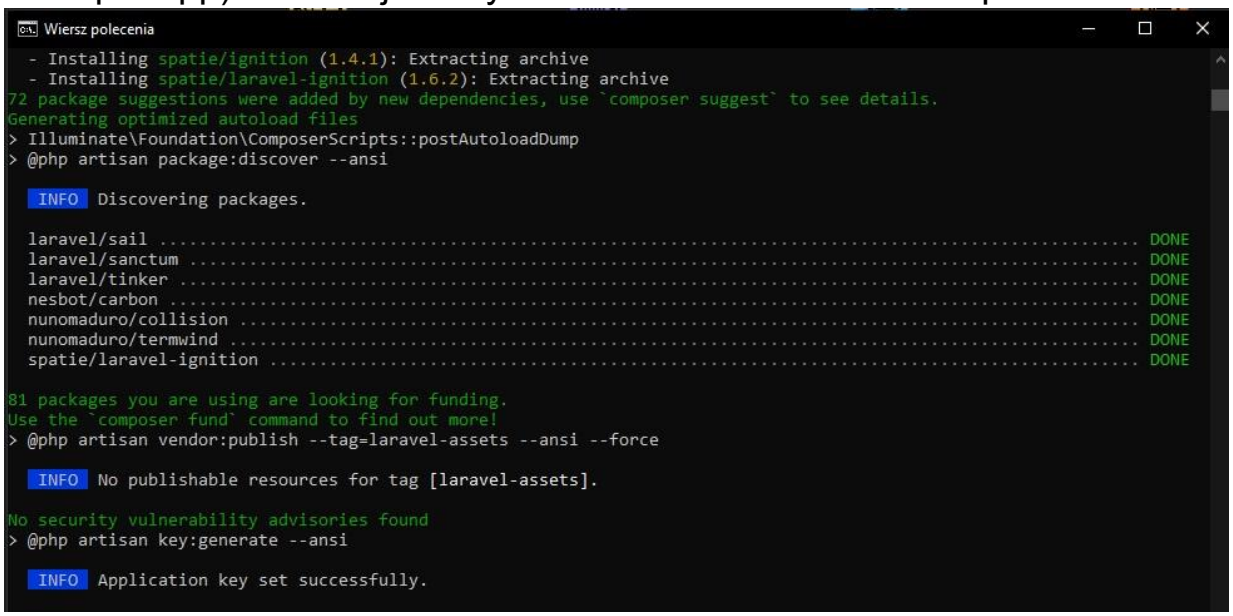
4. Utworzyć nowe repozytorium GitHub:



Utworzyłem nowe publiczne repozytorium GitHub dla mojego projektu

5. Utworzyć nowy projekt oparty na frameworku Laravel:

Utworzyłem nowy projekt używając komend dostępnych na stronie Laravela podanej w instrukcji (composer create-project laravel/laravel example-app). Instalacja wszystkich zależności zaszła bez przeszkód.



Oprócz tego zainstalowałem także Laravela globalnie (przy pomocy komendy ‘composer global require laravel/installer’)

Następnie włączyłem develpoment server:

```
C:\Users\Kamil\example-app>php artisan serve

  INFO  Server running on [http://127.0.0.1:8000].

  Press Ctrl+C to stop the server
```

6. Poprawnie skonfigurować projekt korzystając z dokumentacji:

Sprawdziłem obecne ustawienia aplikacji za pomocą komendy ‘php artisan about’:

```
C:\Users\Kamil\example-app>php artisan about

  Environment ..........
  Application Name ..... Laravel
  Laravel Version ..... 9.43.0
  PHP Version ..... 8.1.12
  Composer Version ..... 2.4.4
  Environment ..... local
  Debug Mode ..... ENABLED
  URL ..... localhost
  Maintenance Mode ..... OFF

  Cache ..........
  Config ..... NOT CACHED
  Events ..... NOT CACHED
  Routes ..... NOT CACHED
  Views ..... NOT CACHED

  Drivers ..........
  Broadcasting ..... log
  Cache ..... file
  Database ..... mysql
  Logs ..... stack / single
  Mail ..... smtp
  Queue ..... sync
  Session ..... file
```

Sprawdziłem czy wartości w pliku .env zgadzały się z instrukcjami na podanej stronie. (https://blog.quickadminpanel.com/how-to-use-laravel-env-example-files/).

Wartości dotyczące baz danych w większości zgadzały się z ustawieniami wcześniej utworzonej przeze mnie bazy danych:

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=laravel
DB_USERNAME=root
DB_PASSWORD=
```

	User name	Host name	Password	Global privileges	User group	Grant	Action
☐	Any	%	No	USAGE		No	Edit privileges
☐	pma	localhost	No	USAGE		No	Edit privileges
☐	root	127.0.0.1	No	ALL PRIVILEGES		Yes	Edit privileges
☐	root	::1	No	ALL PRIVILEGES		Yes	Edit privileges
☐	root	localhost	No	ALL PRIVILEGES		Yes	Edit privileges

Problemy:
Na początku nie wiedziałem co oznaczają niektóre parametry ale odpowiedzi na moje pytania znalazłem w Internecie. Okazało się, że parametr ‘DB_DATABASE’ oznaczający nazwę mojej bazy danych jest

niepoprawny. Błąd naprawiłem podmieniając nazwę ‘laravel’ na nazwę mojej bazy danych ‘projekt_rest_api’.

Następnie musiałem sprawdzić, czy połączenie działa, nie wiedziałem jednak jak to zrobić.

Odpowiedzi na pytania:

1. Co to jest migracja?

   Migracja to pojedyńczy plik definiujący tabelę w bazie danych. Wiele plików migracyjnych tworzy system wersjonowania bazy danych, pozwalający na łatwe dzielenie się jej strukturą z innymi lub odtwarzanie jej struktury z przeszłości. Każdy plik znajduje w sobie kod z instrukcją, jak należy utworzyć daną tabelę.

   Aby utworzyć plik migracyjny w Laravelu, należy użyć komendy:
   php artisan make:migration [nazwa tabeli]

   W Laravelu migracje zlokalizowane są domyślnie w folderze zlokalizowanym w database/migrations/

2. Co to są Kody HTTP?

   Kody HTTP to ustandaryzowane odpowiedzi serwera sieciowego na zapytania ze strony aplikacji klienta. Kody te informują o statusie zapytania (np. czy zostało ono odrzucone) oraz jakie czynności klient powinien wykonać (np. przejść autoryzację). Znaczna część kodów jest przekazywanych bez wiedzy użytkownika i służą one jedynie komunikacji między aplikacją a serwerem. Niektóre kody (zwłaszcza kody błędów) są wyświetlane użytkownikowi po ich otrzymaniu od serwera (np. kod 404 mówiący o nieodnalezieniu przez serwer zasobu

podanego w URL oraz żadnej informacji o istnieniu tego zasobu w przeszłości).

Kody HTTP dzielimy na kody informacyjne, przekierowania, powodzenia, błędu aplikacji klienta oraz błędu serwera HTTP.

3. Co to jest programowanie obiektowe?

   Programowanie obiektowe to sposób organizacji struktury danego programu, w którym program ten definiuje się za pomocą tzw. obiektów – czyli programistycznych reprezentacji struktur danych zawierających cechy danego obiektu oraz metody (funkcje definiujące zachowanie tego obiektu oraz jego relacje z innymi obiektami).

   W programowaniu obiektowym, program jest zbudowany z wielu obiektów oddziałowujących na siebie w taki sposób aby wykonać dane zadanie. Każdy obiekt ma swoje własne dane i metody, dzięki czemu może być używany oddzielnie w wielu programach. Nadaje to programom modularność, oraz łatwiejszą do zrozumienia sieć relacji między ich wewnętrznymi elementami. Ułatwia to znacznie konserwację, tworzenie i ponowne wykorzystywanie kodu w programie.

4. Dlaczego wykorzystuje się pliki .env?

   Pliki .env służą do przechowywania wartości konfiguracyjnych środowiska, w którym działa program. Przetrzymuje się w nich także wiele wrażliwych informacji np. hasła, tokeny zabezpieczające czy klucze API.

   Gdy aplikacja jest uruchamiana, zazwyczaj wczytuje ona dane z plików .env i bazując na nich ustawia parametry środowiska. To pozwala aplikacji na dostęp do tych parametrów później i korzystania z nich w kodzie.

   Plików .env używa się aby odseparować wrażliwe informacje od kodu aplikacji, zwiększając jej bezpieczeństwo. Ułatwia to także zarządzanie parametrami środowiska, gdyż zlokalizowane są one w jednym miejscu.

5. Czym charakteryzuje się relacyjna baza danych?

Relacyjna baza danych charakteryzuje się przetrzymywaniem danych w tabelach podzielonych na kolumny i wiersze, pomiędzy którymi występują relacje – każda tabela posiada unikatowy klucz umożliwiający jej

identyfikację. Dzięki temu tabele mogą odwoływać się do innych tabel i zawartych w nich informacji.
Umożliwia to „łączenie” tabel relacjami np. tabela z klientami może odwoływać się do tabeli z produktami tak aby wyszukać wszystkie produkty zakupione przez danego klienta.

Relacyjne bazy danych bardzo ułatwiają nawigację po danych i umożliwiają budowanie skomplikowanych baz danych bez tworzenia ogromnych tabel. Relacyjną bazę danych łatwo również rozbudowywać poprzez tworzenie nowych tabel i relacji.

6. Wykorzystywanie plików w formacie json:

Pliki .json używane są do przetrzymywania danych, które muszą być dzielone z różnymi programami lub usługami. Używanie tego samego rodzaju pliku pozwala na uniwersalność i standaryzację.

Ponadto pliki .json mogą być łatwo odczytywane i zmieniane za pomocą edytorów tekstów. Używają standaryzowanej składni dzięki czemu można ich używać nawet w programach pisanych w różnych językach.

W plikach json dane zapisywane są w parach klucz-wartość oddzielonych dwukropkiem (np. „age”:28, „name”:”John”), gdzie pierwszy wyraz to klucz identyfikujący danę a drugi to wartość do niej przypisana.

Pliki .json używane są często w tworzeniu stron internetowych oraz są wspierane przez wiele języków programowania oraz frameworków.