Czołem !

Dziękujemy Ci za wybranie naszego ogłoszenia. W związku z Twoją chęcią rozpoczęcia pracy w stacku Laravel i VueJS - przygotowaliśmy dla Ciebie kilka zadań, które pozwolą nam zweryfikować twoją znajomość poszczególnych języków oraz podejście jakie masz do programowania.

Zadanie składa się z części dotyczącej PHP, Laravel, VueJS - poszczególne sekcje możesz podzielić na foldery w swoim repozytorium, by łatwiej można było to przeglądać i analizować. Zadanie możesz udostępnić jako publiczne repozytorium. Zwracamy uwagę na to jakie commity przesyłasz - nie ma znaczenia, czy użyjesz języka polskiego czy angielskiego. Zadania związane z Laravel i Vue - mogą być wykonane w jednym projekcie. Tutaj pozostawiam dowolność :)

Jeżeli zakończysz zadanie, prześlij mi informację na maila. Po przesłaniu potwierdzenia, że się skończyło - zaczynam sprawdzać.
Gotowe zadanie udostępnij na email: damian@onx.com.pl

Zadanie związane z PHP:

_____ 1 zadanie

W ramach przetwarzania danych należy ukończyć implementację metody “make” klasy Pipeline.

Metoda “make” powinna przyjmować zmienną liczbę funkcji i zwracać nową funkcję, która przyjmuje jeden parametr $arg.
Zwrócona funkcja powinna wywołać pierwszą funkcję w “make” z parametrem $arg i wywołać drugą funkcję z wynikiem pierwszej funkcji.
Zwrócona funkcja powinna kontynuować wywoływanie każdej funkcji w “make” w kolejności, postępując według tego samego wzorca i zwracać wartość z ostatniej funkcji.

Na przykład wywołanie make(function($var) { return $var * 3; }, function($var) { return $var + 1; }, function($var) { return $var / 2; }), a następnie wywołanie zwróconej funkcji z argumentem 3 powinno zwrócić 5.

_____ 2 zadanie

Interfejs użytkownika zawiera dwa rodzaje kontrolek wprowadzania danych: TextInput, który akceptuje wszystkie teksty oraz NumericInput, który akceptuje tylko cyfry.

Zaimplementuj klasę TextInput, która zawiera:

Metodę publiczną add($text) - dodającą podany tekst do bieżącej wartości.
Metodę publiczną getValue() - zwracającą bieżącą wartość (string).

Zaimplementuj klasę NumericInput, która: Dziedziczy po TextInput. Przedefiniowuje metodę add tak, aby każdy tekst nienumeryczny był ignorowany.

_____ 3 zadanie

Klasa RankingTable śledzi wyniki każdego gracza w lidze. Po każdej grze, gracz zapisuje swój wynik za pomocą funkcji recordResult().

Ranking gracza w lidze jest obliczany zgodnie z następującą logiką:

Gracz z najwyższym wynikiem jest sklasyfikowany jako pierwszy (rank 1). Gracz z najniższym wynikiem jest sklasyfikowany jako ostatni.
Jeśli dwóch graczy ma ten sam wynik, to gracz, który rozegrał mniej gier, jest sklasyfikowany wyżej.
Jeśli dwóch graczy ma ten sam wynik i rozegrali tę samą liczbę gier, to gracz, który był pierwszy na liście graczy, jest sklasyfikowany wyżej.
Zaimplementuj funkcję playerRank, która zwraca gracza o podanym rankingu.

```
$table = new RankingTable(array('Jan', 'Maks', 'Monika'));
$table->recordResult('Jan', 2);
$table->recordResult('Maks', 3);
$table->recordResult('Monika', 5);
echo $table->playerRank(1);
```

Wszyscy gracze mają taki sam wynik. Jednak Maks i Monika rozegrali mniej gier niż Jan, a ponieważ Monika znajduje się przed Maks na liście graczy, jest on sklasyfikowany jako pierwszy. Dlatego powyższy kod powinien zwrócić "Monika".

_____ 4 zadanie

Jeden z rodzajów słownika, czyli tezaurus, zawiera słowa i ich synonimy. Poniżej znajduje się przykład struktury danych, która definiuje tezaurus:
array("market" => array("trade"), "small" => array("little", "compact"))
Należy zaimplementować funkcję getSynonyms, która przyjmuje słowo jako ciąg znaków i zwraca wszystkie synonimy dla tego słowa w formacie JSON, jak w poniższym przykładzie.
Na przykład wywołanie $thesaurus->getSynonyms("small") powinno zwrócić:

```
'{"word":"small","synonyms":["little", "compact"]}'
```

podczas gdy wywołanie z słowem, dla którego nie ma synonimów, np. $thesaurus->getSynonyms("asleast") powinno zwrócić:
'{"word":"asleast","synonyms":[]}'

Zadania dotyczące Laravela:

1. Pobierz informacje na temat klienta, pracownika, który jest do niego przypisany oraz zamówień - jakie rzeczy ostatnio kupił. Model klienta jest powiązany kluczami z pozostałymi, więc informacje są do uzyskania poprzez relację. Do wykonania tego zadania pracuj na Eloquent ORM.

2. Wielu użytkowników może mieć wiele samochód w firmie. Stwórz relację, która pozwala przypisywać użytkownika do auta i weryfikować czy korzysta on z tego auta w danym momencie.
3. Stwórz Test HTTP, który będzie weryfikował poprawność tworzenia, wyświetlania, edycji i usuwania pojazdów, klientów oraz pracowników. Oczekujemy poprawnych odpowiedzi, więc trzeba stworzyć pełną logikę działania.
4. Stwórz system notyfikacji - gdy użytkownik zostaje przypisany do pojazdu, wyślij notyfikację do niego oraz do administratora systemu, że samochód został przypięty. W momencie, gdy użytkownik zostanie dezaktywowany, również powinien dostać stosowną informację.

Zadania dotyczące Vue

1. Z poziomu frontendu pobierz informację na temat klienta, przygotuj listę klientów oraz podgląd konkretnego klienta ze wszystkimi informacjami na jego temat - pracownik który do niego przypisany, ostatnio kupione rzeczy oraz ile łącznie wydał na zakupy. Też wyświetl samochód który posiada klient. Musisz odpowiednio przygotować routing po stronie frontendu.
2. Przygotuj zarządzanie klientem - dodawanie, usuwanie, edycję. Podczas dodawania/update musisz odpowiednio dokonać walidacje wprowadzonych danych
3. Przy liście klientów przygotuj paginację, sortowanie filtrację oraz wyszukiwanie
4. Utwórz prostą autentykacje oraz zapisz globalnie dane o użytkowniku w aplikacji. Użyj do tego Pinii
5. Przepisz poniższy fragment kodu na bardziej czytelny zapis

```
arrA.filter(x => !arrB.includes(x)).concat(arrB.filter(x => !arrA.includes(x)))
```