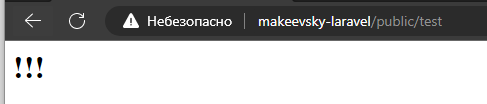
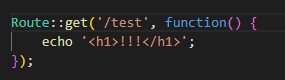
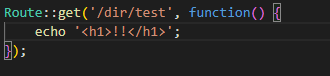
Выполнил: Макеевский Никита, ИСИп-302

**Практическая работа №1**

1. Сделайте так, чтобы при обращении на адрес /test/ в браузер выводилось сообщение '!'.



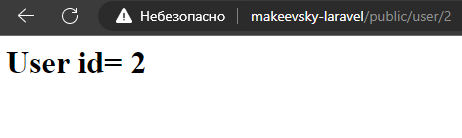
1. Сделайте так, чтобы при обращении на адрес /dir/test/ в браузер выводилось сообщение '!!'.



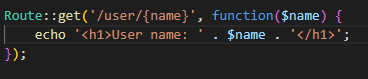


1. Пусть дан адрес вида /user/:id/, где вместо :id может быть любое число. Сделайте так, чтобы при обращении на адрес такого вида в браузер выводилось содержимое переданного параметра.





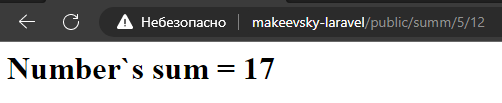
1. Пусть дан адрес вида /user/:name/, где вместо :name может быть любая строка. Сделайте так, чтобы при обращении на адрес такого вида в браузер выводилась строка 'имя юзера :name', где вместо :name будет переданная строка.



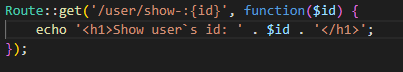


1. Пусть дан адрес вида /sum/:num1/:num2/, где вместо :num1 и :num2 могут быть любые числа. Сделайте так, чтобы при обращении на адрес такого вида в браузер выводилось сумма переданных чисел.



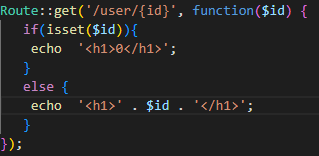


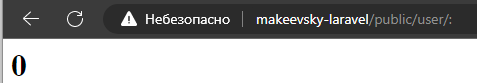
1. Пусть дан адрес вида /user/show-:id/, где вместо :id может быть любое число. Сделайте так, чтобы при обращении на адрес такого вида в браузер выводилось содержимое переданного параметра.



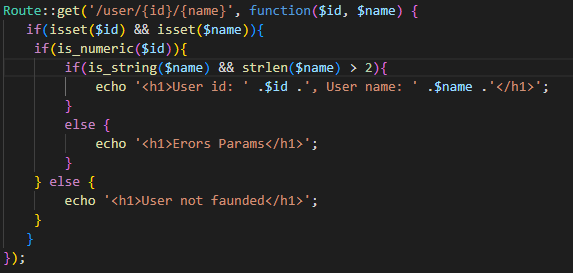


1. Пусть дан адрес вида /user/:id/, где вместо :id может быть любое число. Сделайте так, чтобы параметр id был не обязательный и, если он не передан, вместо него будет передаваться 0.

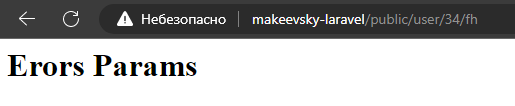


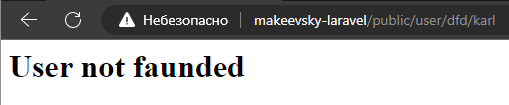


1. Сделайте маршрут вида /user/:id/:name, где вместо :id должно быть число, а вместо :name - строка, состоящая из маленьких латинских букв количеством более 2-х. Добавьте проверку параметров регулярными выражениями.

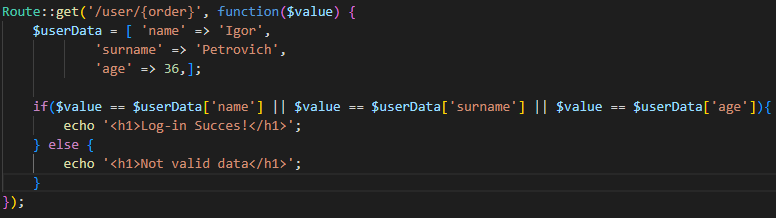


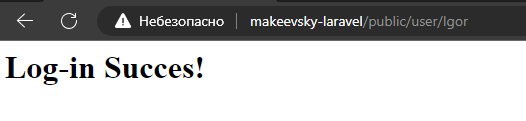


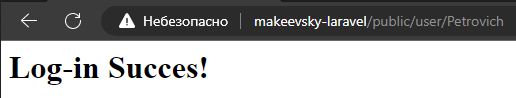


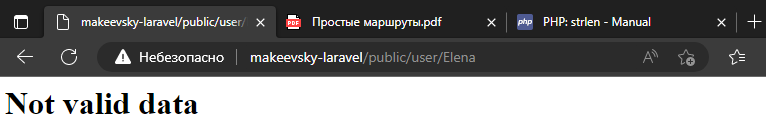


1. Сделайте маршрут вида /users/:order/, где вместо :order должно быть одно из значений: 'name', 'surname' или 'age'.

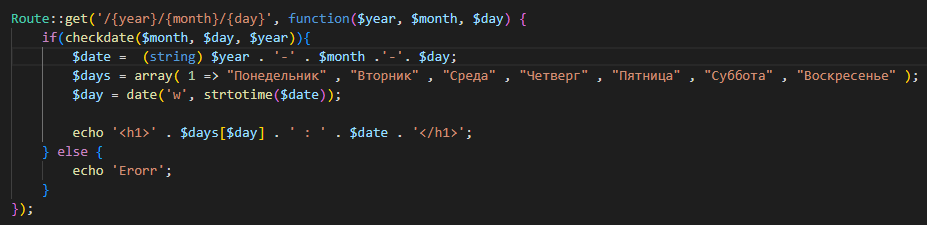


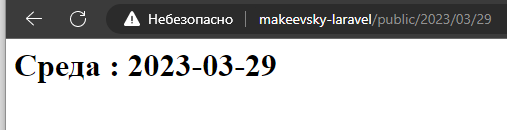






1. Сделайте маршрут вида /:year/:month/:day/, где вместо :year должен быть год, вместо :month - месяц, вместо :day - день. Выведите на экран день недели, соответствующий этой дате.

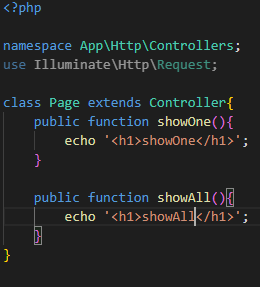




**Практическая работа 2 по Laravel. Контроллеры**

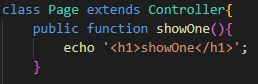
**Задание**

1. Сделайте контроллер **Page.**

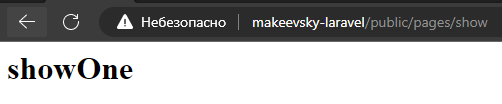


1. В контроллере **Page** сделайте действие **showOne**. Сделайте так, чтобы данное действие было доступно по адресу **/pages/show/**. По обращению к данному действию выведите на экран какую-нибудь строку.

В Controllers/Page.php

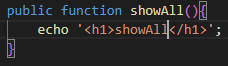


В routes/Web.php

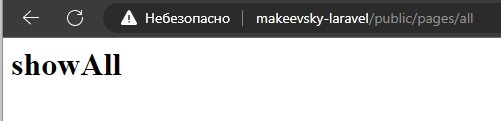
1. В контроллере Page сделайте действие showAll. Сделайте так, чтобы данное действие было доступно по адресу /pages/all/. По обращению к данному действию выведите на экран какую-нибудь строку.

В Controllers/Page.php



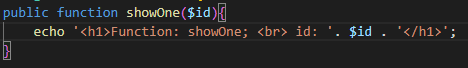
В routes/Web.php





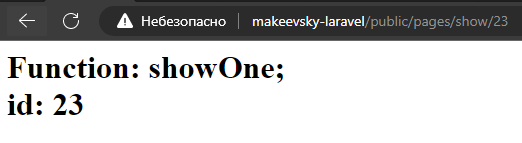
1. Модифицируйте действие showOne контроллера Page. Сделайте так, чтобы данное действие было доступно по адресу /pages/show/:id/, где вместо :id будет какое-нибудь число. По обращению к данному действию выведите на экран переданное число.

В Controllers/Page.php



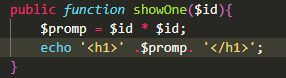
В routes/Web.php





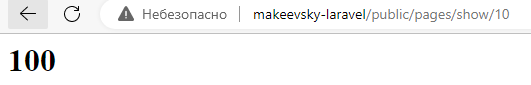
1. Модифицируйте предыдущую задачу так, чтобы на экран выводился квадрат переданного числа.

В Controllers/MainController.php



В routes/Web.php





1. Сделайте в действии **showOne** следующий массив:

$pages = [

   1 => **'страница 1'**,

   2 => **'страница 2'**,

   3 => **'страница 3'**,

   4 => **'страница 4'**,

   5 => **'страница 5'**,

];

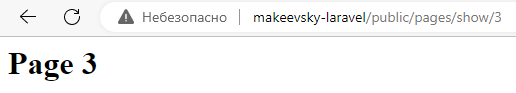
Выведите на экран элемент массива, номер которого соответствует переданному параметру. Если страницы с таким номером нет - выведите сообщение об этом.

В Controllers/MainController.php



В routes/Web.php

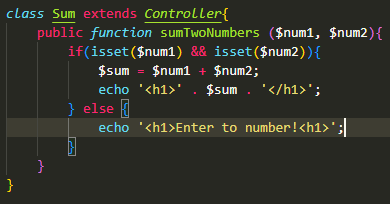






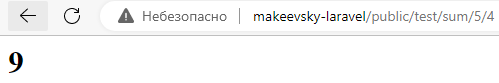
1. Создайте контроллер **Test**, а в нем действие **sum**. Сделайте так, чтобы данное действие было доступно по адресу **/test/sum/:num1/:num2/**, где где вместо **:num1** и **:num2** будут какие-нибудь числа. Пусть по обращению к данному действию на экран выводится сумма переданных чисел.

В Controllers/Sum.php



В routes/Web.php





1. Создайте контроллер **Employee** (работник). В конструкторе класса запишите в свойство **$this->employees** массив с работниками:

**namespace** App\Http\Controllers;

**use** App\Http\Controllers\Controller;

**class** TestController **extends** Controller

{

**private $employees**;

**public function** \_\_construct()

   {

       $this->**employees** = [

           1 => [

**'name'** => **'user1'**,

**'surname'** => **'surname1'**,

**'salary'** => 1000,

           ],

           2 => [

**'name'** => **'user2'**,

**'surname'** => **'surname2'**,

**'salary'** => 2000,

           ],

           3 => [

**'name'** => **'user3'**,

**'surname'** => **'surname3'**,

**'salary'** => 3000,

           ],

           4 => [

**'name'** => **'user4'**,

**'surname'** => **'surname4'**,

**'salary'** => 4000,

           ],

           5 => [

**'name'** => **'user5'**,

**'surname'** => **'surname5'**,

**'salary'** => 5000,

           ],

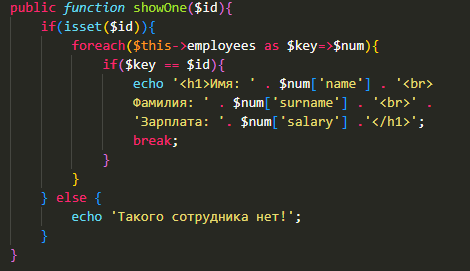
       ];

   }

}

1. В контроллере **Employee** сделайте действие showOne. Параметром оно должно принимать номер работника в массиве $this->employees и выводить на экран строку с именем, фамилией и зарплатой работника.

В Controllers/Employee.php



В routes/Web.php





1. В контроллере **Employee** сделайте действие **showField**. Параметрами оно должно принимать номер работника в массиве **$this->employees** и название одного из полей ('name', 'surname' или 'salary'). Действие выводить на экран значение соответствующего поля соответствующего работника.

В Controllers/Employee.php



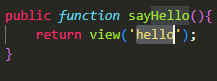
В routes/Web.php



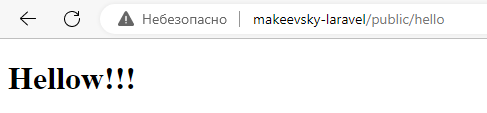


**Практическая работа 3 по Laravel. Представления**

1. Сделайте представление для какого-нибудь действия одного из ваших контроллеров.

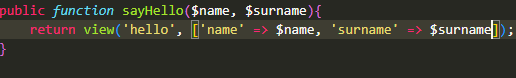






1. Пусть в действии контроллера даны переменные $name (имя), $surname (фамилия). Передайте значения этих переменных в представление и выведите содержимое каждой из этих переменных на экран.





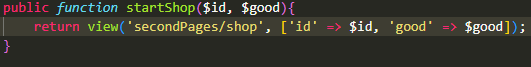




1. Создайте в папке с представлениями несколько подпадок. Перенесите ваше представление в соответствующую папку. Сделайте так, чтоб данное представление отображалось с учетом иерархии подпапок.



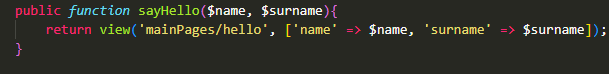




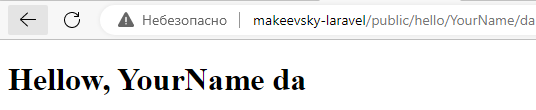


\_---------------------------------------------------------------------------------------------







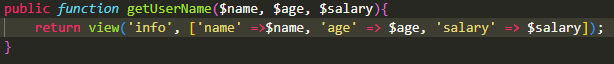


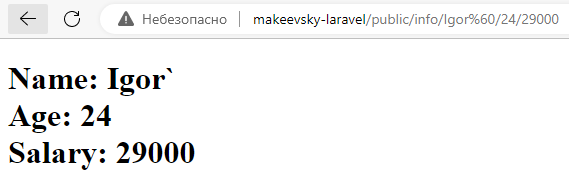
**Задание 4.** Шаблонизатор Blade

1. Пусть в действии контроллера даны переменные $name (имя), $age (возраст) и $salary (зарплата). Передайте значения этих переменных в представление и выведите содержимое каждой из этих переменных в отдельном абзаце.

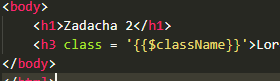


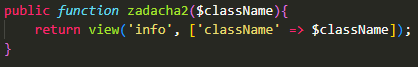




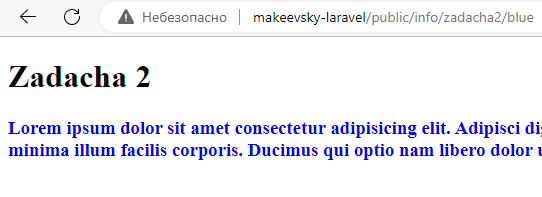
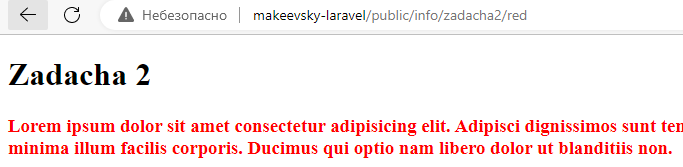


1. Пусть в действии дана переменная, содержащая CSS класс. Передайте эту переменную в представление и для какого-нибудь тега значением атрибута **class** укажите нашу переменную.





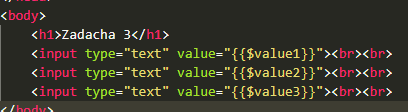


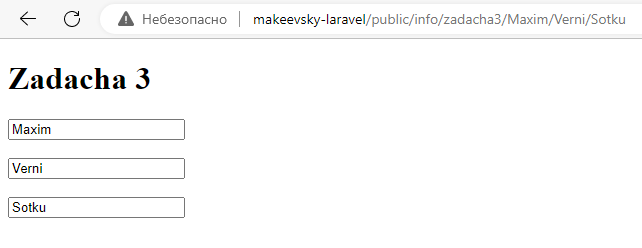
 

1. Пусть в представлении даны 3 инпута. Передайте из действия в представление 3 переменные, значения которых запишите в атрибуты **value** наших инпутов.



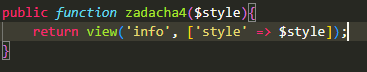




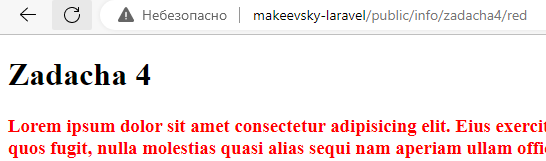


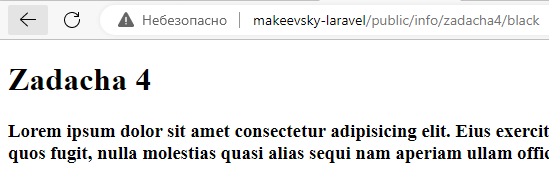
1. ,5. Пусть в представлении дан абзац. Передайте из действия в представление переменную, содержащую CSS код, задающий красный цвет текста. С помощью атрибута **style** покрасьте наш абзац в красный цвет.





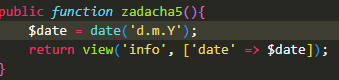


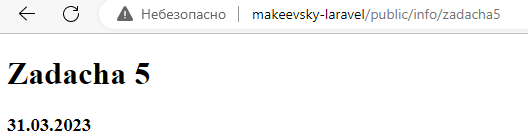




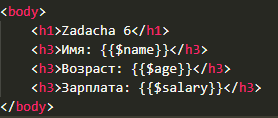
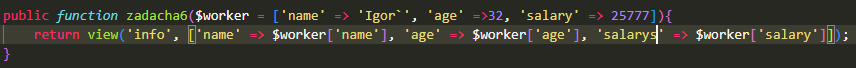
1. Выведите в представлении текущую дату в формате *день.месяц.год* используя блоки php-кода.

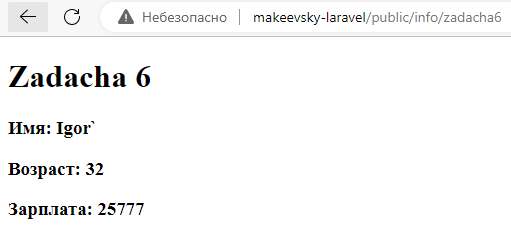


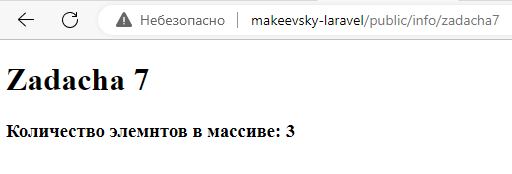


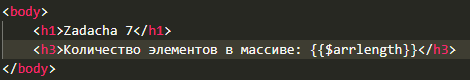
1. Пусть из действия в представление передаются данные работника в виде массива. Пусть в массиве будет ключ **name** (имя), ключ **age** (возраст) и ключ **salary** (зарплата). Выведите каждый элемент массива в отдельном абзаце.

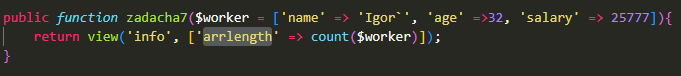




1. Передайте в представление какой-нибудь массив. Выведите на экран количество элементов в этом массиве.

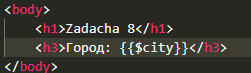
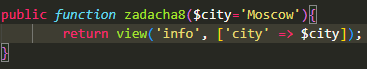


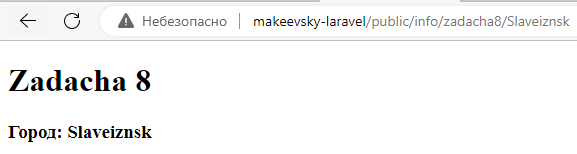




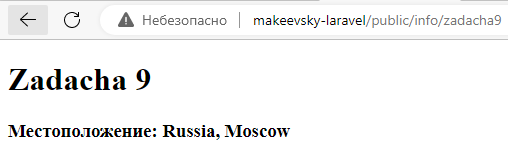
1. Пусть из действия в представление передается переменная **$city** (город). Выведите в представлении названия города из этой переменной. Если же город не передан - пусть по умолчанию выведется город **Москва**.

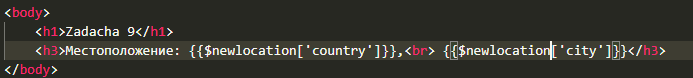


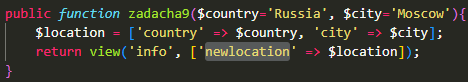
 



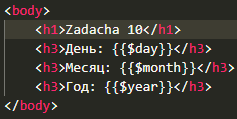
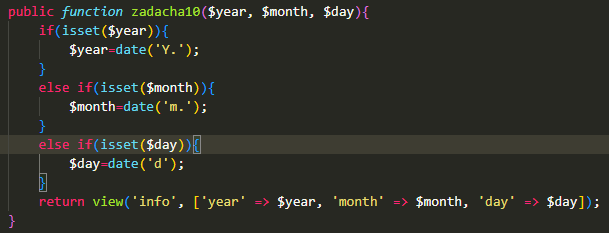
1. Пусть из действия в представление передается массив **$location** (местоположение) с ключами **country** (страна), **city** (город). Выведите каждый элемент это массива в отдельном абзаце. Сделайте так, чтобы, если не задана страна, то по умолчанию вывелась **Россия**, а если не задан город, то по умолчанию вывелась **Москва**.

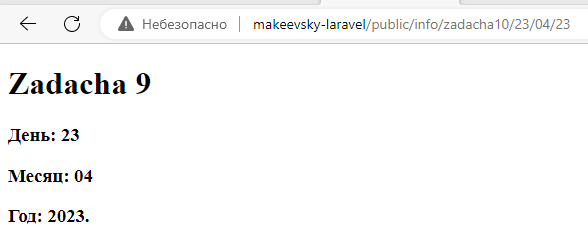




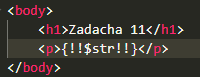
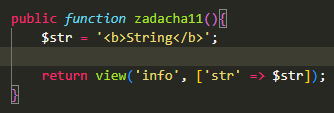
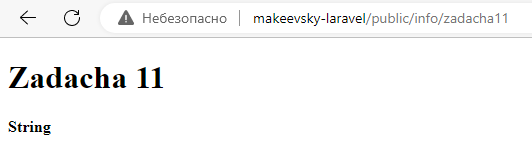


1. Пусть из действия в представление передаются переменные **$year** (год), **$month** (месяц) и **$day** (день). Сделайте так, чтобы, если какая-либо из этих переменных не задана, то вместо нее выведется текущее значение (текущий год, месяц или день).



1. Пусть в переменной **$str** хранится строка с тегами, например, **'<b>строка</b>'**. Выведите эту строку на экран так, чтобы теги выполнили свое действие (то есть чтобы в данном случае текст стал жирным).

1. С помощью комментариев Blade закомментируйте какую-нибудь часть кода представления.



**Практическая работа 5.** **Условные директивы и циклы в шаблонизаторе Blade**

**Инструкция:**

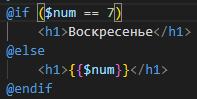
1. Продолжить выполнение в проекте \_фамилия-laravel
2. Выполнить приведенные ниже задания. Результаты выполнения поместить в отчет по практической работе в следующем виде:

* текст задания
* скрин вашего кода
* скрин результата в браузере (на скрине должна быть видна адресная строка)

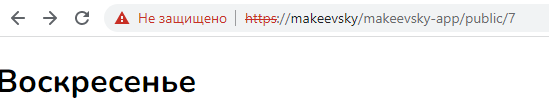
**Задания для выполнения:**

Условия:

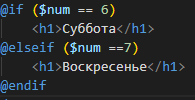
1. Пусть из действия в представление передается номер дня недели. Сделайте так, чтобы, если передан номер воскресенья, то на экран вывелся текст 'Воскресенье'. Пусть текст выводится в абзаце.

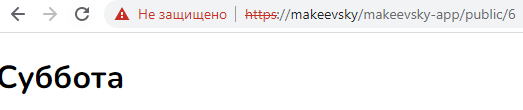
 





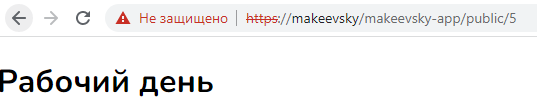
1. Модифицируйте предыдущую задачу так, чтобы на экран выводилось названия дня недели в том случае, если сегодня выходной день (то есть или суббота или воскресенье). Для решения задачи напишите две директивы **@if**.

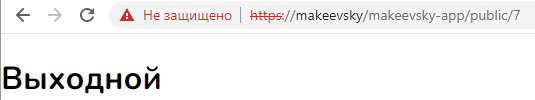




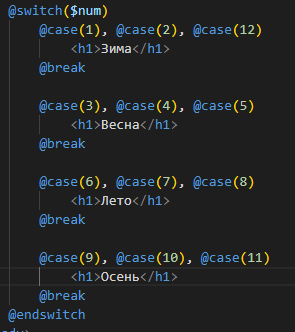
1. Пусть из действия в представление передается номер дня недели. Сделайте так, чтобы, если номер дня недели соответствует выходному дню, то вывелось сообщение об этом, в противном случае выведите сообщение о том, что передан номер рабочего дня недели.

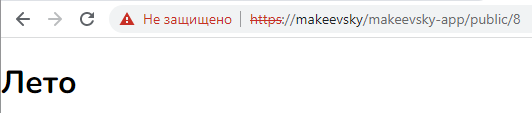




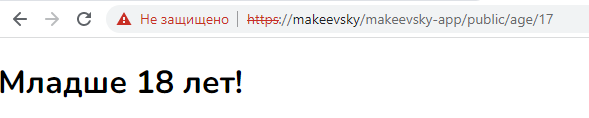


1. Пусть из действия в представление передается номер месяца. С помощью директив **@if** и **@elseif** определите пору года, в которую попадает переданный месяц. Если переданный номер не соответствует ни одному месяцу - выведите сообщение об этом.





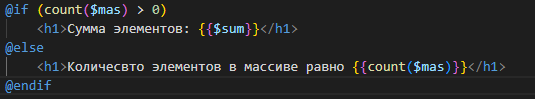
1. Пусть из действия в представление передается возраст пользователя. Если возраст пользователь несовершеннолетний, то выведите сообщение об этом. Для решения задачи воспользуйтесь директивой **@unless**.

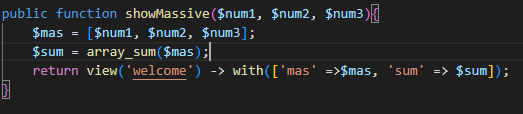




1. Пусть из действия в представление передается массив с числами. Если количество элементов в этом массиве больше нуля, то выведите на экран сумму этих элементов, а если количество элементов равно нулю, то выведите сообщение об этом.



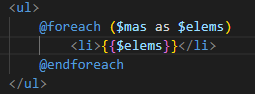


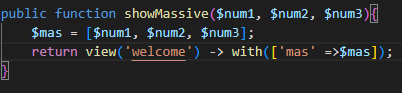


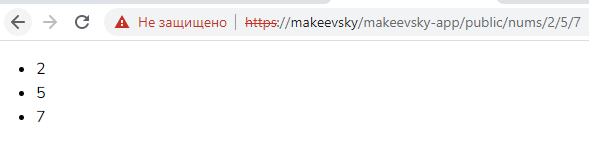


Циклы:

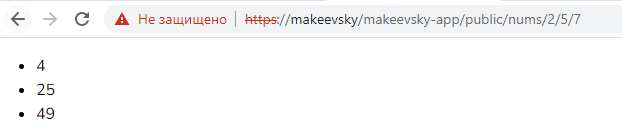
1. Передайте из действия в представление массив с числами. Выведите этот массив в виде списка ul.

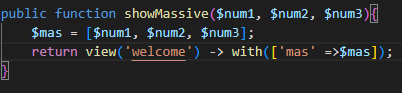


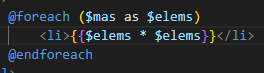




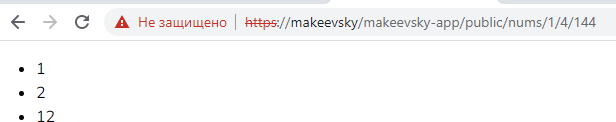
1. Модифицируйте предыдущую задачу так, чтобы в пункты списка выводились квадраты элементов массива.

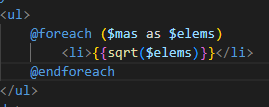


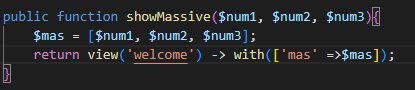




1. Модифицируйте предыдущую задачу так, чтобы в пункты списка выводились квадратные корни элементов массива.

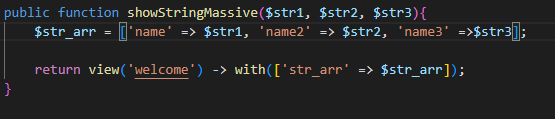


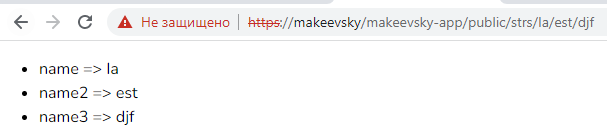




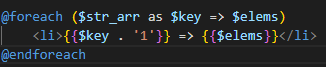
1. Передайте из действия в представление массив со строками. Выведите этот массив в виде списка **ul** так, чтобы выводились и ключи, и значения элементов массива.

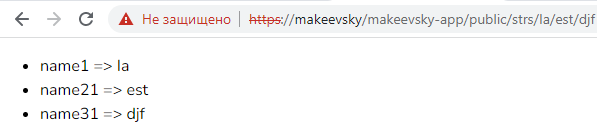




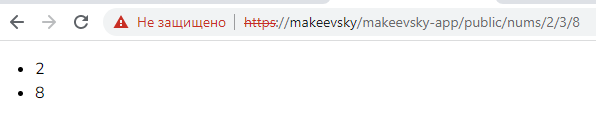


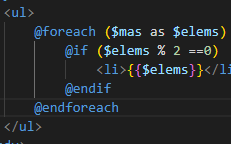
1. Модифицируйте предыдущую задачу так, чтобы на выводе к ключу прибавлялась единица.



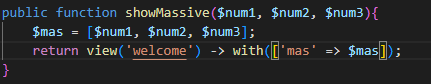


1. Передайте из действия в представление массив с числами. Выведите этот массив в виде списка **ul**. Сделайте так, чтобы в список попадали только элементы, значениями которых служат четные числа.



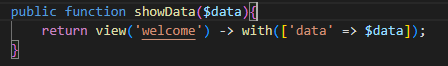




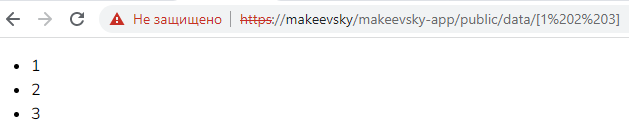


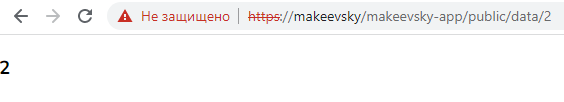
1. Из действия в представление передается переменная **$data**. Эта переменная может содержать массив с числами либо быть обычным числом. Если переменная **$data** - массив, то выведите элементы этого массива в виде списка **ul**, а если переменная **$data** - число, то выведите это число в абзаце.



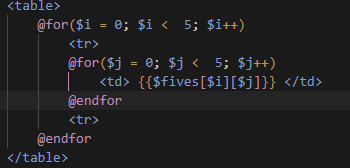
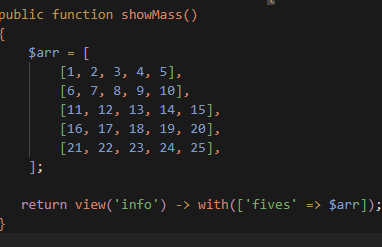








1. Передайте из действия в представление массив с числами от 1 до 25. Выведите его в виде таблицы 5 на 5.





1. Передайте из действия в представление массив с работниками:

$employees = [

   [

**'name'** => **'user1'**,

**'surname'** => **'surname1'**,

**'salary'** => 1000,

   ],

   [

**'name'** => **'user2'**,

**'surname'** => **'surname2'**,

**'salary'** => 2000,

   ],

   [

**'name'** => **'user3'**,

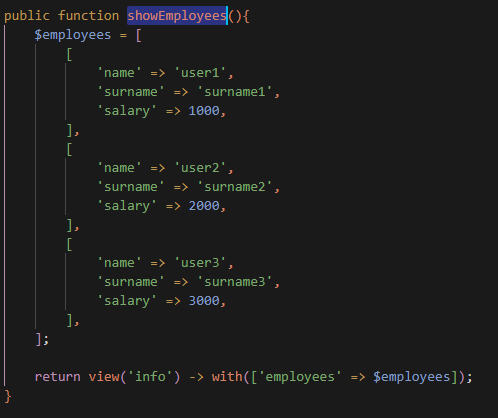
**'surname'** => **'surname3'**,

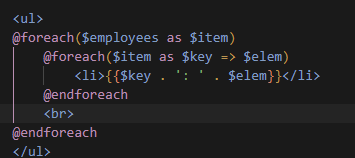
**'salary'** => 3000,

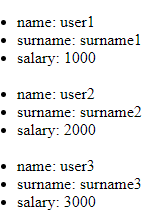
   ],

];

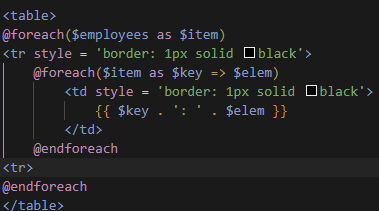
Выведите этот массив в виде списка **ul**.





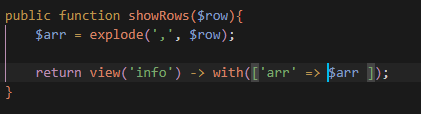


10. Выведите массив из предыдущей задачи в виде таблицы **table**.



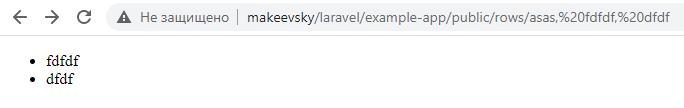
11.Из действия в представление передается массив со строками. Выведите элементы этого массива в виде списка **ul**. Сделайте так, чтобы перед значением элемента, выводился еще и порядковый номер итерации, начиная с 0.

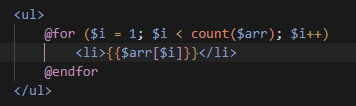






12. Модифицируйте предыдущую задачу так, чтобы порядковый номер итерации начинался с 1.

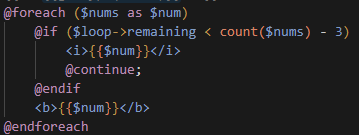


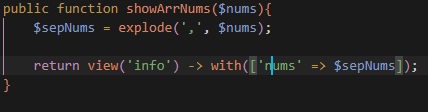


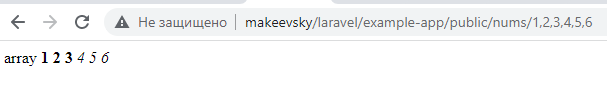
13. Модифицируйте предыдущую задачу так, чтобы для первой **li** добавлялся CSS класс **first**, а для последней - класс **last**.



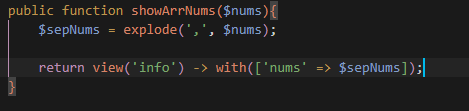
14. Из действия в представление передается массив с числами. Выведите каждый элемент этого массива в своем теге **b**. С помощью **$loop->remaining** три последних элемента массива выведите не в теге **b**, а в теге **i**.

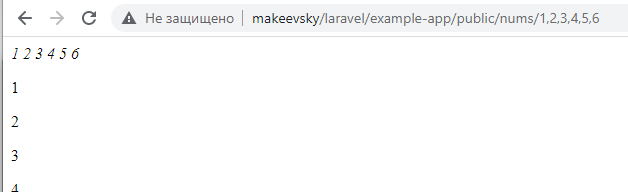




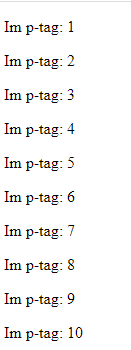


15. Воспользуйтесь примером работы с циклами выше и проверьте работу директив @break, @continue, а также директивы **@forelse**



16. С помощью цикла **$@for** выведите 10 абзацев, заполненных числами от 1 до 10.

**Практическая работа 6. Наследование шаблонов Blade**

**Задания для выполнения:**

1. Пусть дана вот такая верстка:

<!**DOCTYPE html**>

<**html**>

  <**head**>

     <**title**>**Тайтл страницы**</**title**>

  </**head**>

  <**body**>

     <**header**>

**хедер**

</**header**>

     <**aside**>

**сайдбар**

</**aside**>

     <**main**>

**контент**

</**main**>

     <**footer**>

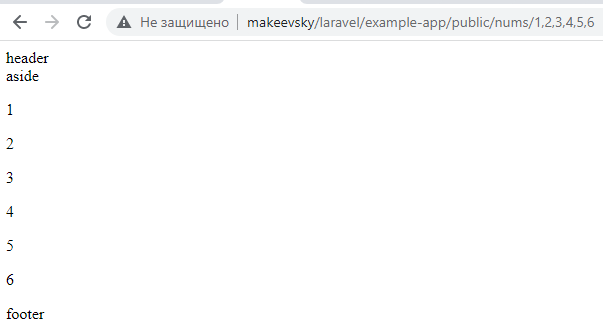
**футер**

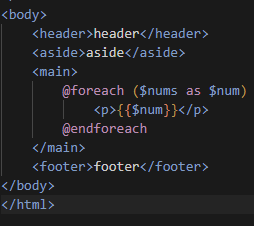
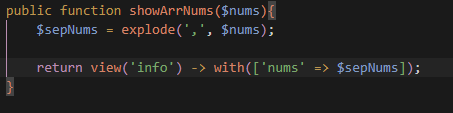
</**footer**>

  </**body**>

</**html**>

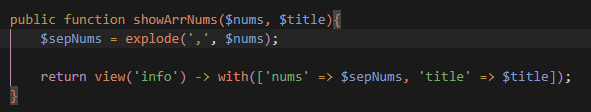
Сделайте данную верстку макетом вашего сайта так, чтобы в тег **main** загружались представления действий контроллеров.

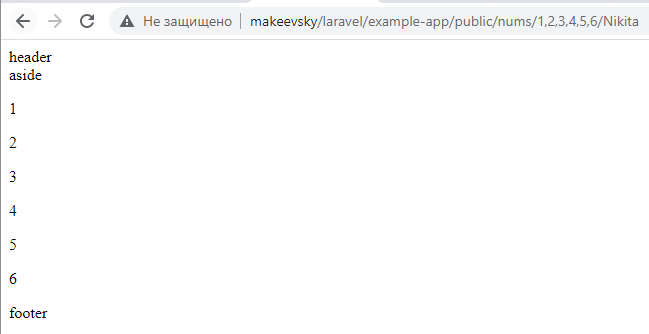


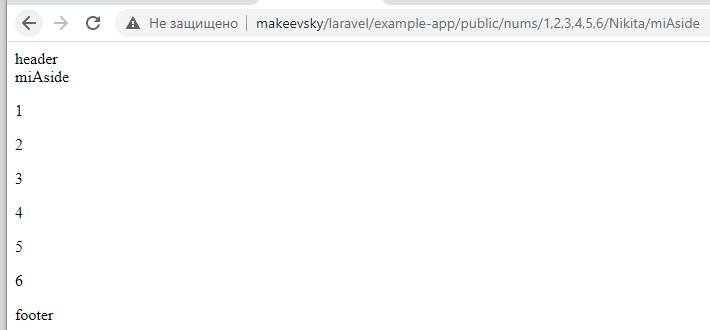
2. Сделайте так, чтобы каждое действие отправляло в представление переменную **$title**, содержащую тайтл страницы. Модифицируйте код предыдущей задачи так, чтобы переданный тайтл попадал в тег **title** макета сайта.

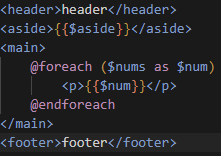


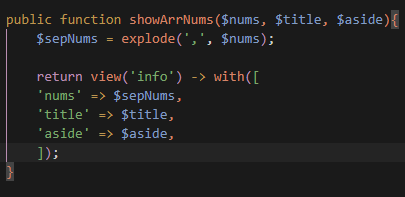


3. Сделайте так, чтобы каждое действие отправляло в представление переменную **$aside**, содержащую дополнительную часть сайдбара. Модифицируйте код предыдущей задачи так, чтобы переданные данные попадали в сайдбар сайта, не удаляя при этом его содержимое, а дописывая переданные данные в конец сайдбара.

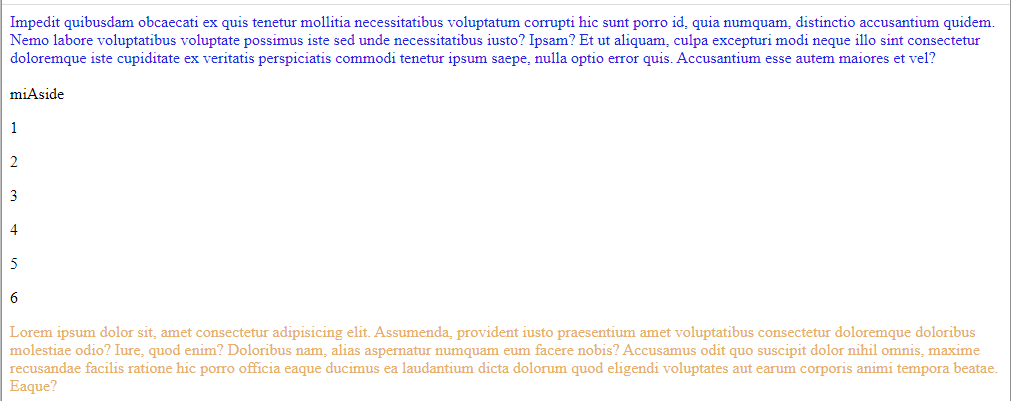


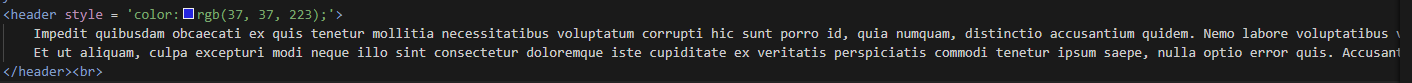




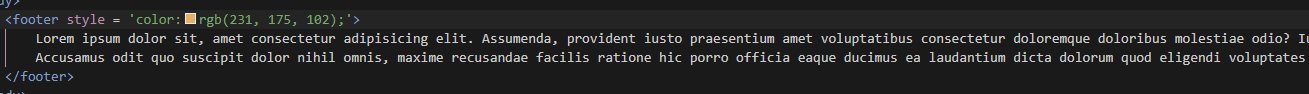


4. Вынесите хедер и футер сайта в отдельные подшаблоны. Разместите их в папке **elems**. Подключите их к основному макету.

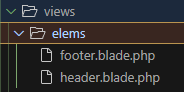
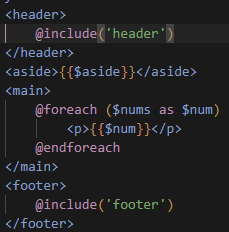


Heder.blade.php: 

Footer.blade.php



Main.blade.php



5. Передайте из действия в представление массив ссылок:

$links = [

   [

**'text'** => **'text1'**,

**'href'** => **'href1'**,

   ],

   [

**'text'** => **'text2'**,

**'href'** => **'href2'**,

   ],

   [

**'text'** => **'text3'**,

**'href'** => **'href3'**,

   ],

];

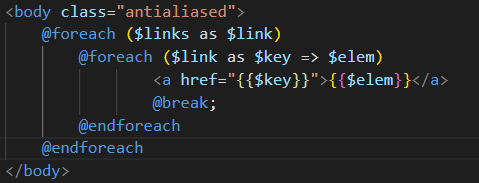
Сформируйте из нашего массива следующий HTML код:

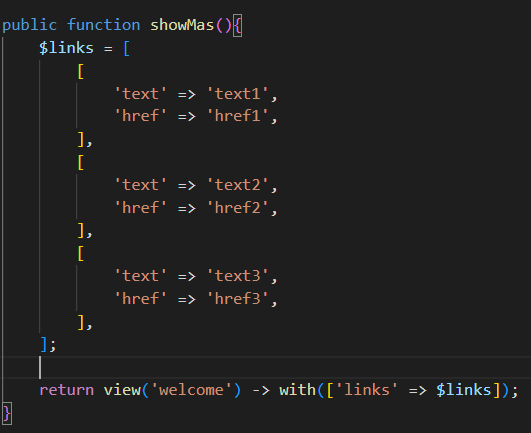
<**a** href=**"http://href1"**>**text1**</**a**>

<**a** href=**"http://href2"**>**text2**</**a**>

<**a** href=**"http://href3"**>**text3**</**a**>

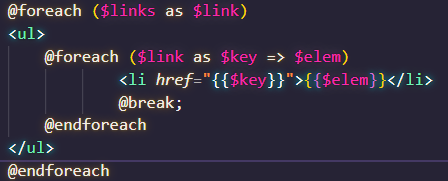


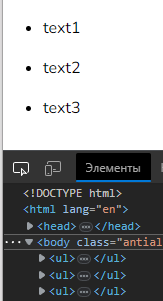






1. Модифицируйте предыдущую задачу так, чтобы каждая ссылка выводилась в отдельном теге **li** тега **ul**.





7. Передайте из действия в представление массив с работниками:

$employees = [

   [

**'name'** => **'user1'**,

**'surname'** => **'surname1'**,

**'salary'** => 1000,

   ],

   [

**'name'** => **'user2'**,

**'surname'** => **'surname2'**,

**'salary'** => 2000,

   ],

   [

**'name'** => **'user3'**,

**'surname'** => **'surname3'**,

**'salary'** => 3000,

   ],

   [

**'name'** => **'user4'**,

**'surname'** => **'surname4'**,

**'salary'** => 4000,

   ],

   [

**'name'** => **'user5'**,

**'surname'** => **'surname5'**,

**'salary'** => 5000,

   ],

];

Выведите этот массив в виде следующей таблицы:

<**table**>

  <**tr**>

     <**td**>**user1**</**td**>

     <**td**>**surname1**</**td**>

     <**td**>1000</**td**>

  </**tr**>

  <**tr**>

     <**td**>**user2**</**td**>

     <**td**>**surname2**</**td**>

     <**td**>2000</**td**>

  </**tr**>

  <**tr**>

     <**td**>**user3**</**td**>

     <**td**>**surname3**</**td**>

     <**td**>3000</**td**>

  </**tr**>

  <**tr**>

     <**td**>**user4**</**td**>

     <**td**>**surname4**</**td**>

     <**td**>4000</**td**>

  </**tr**>

  <**tr**>

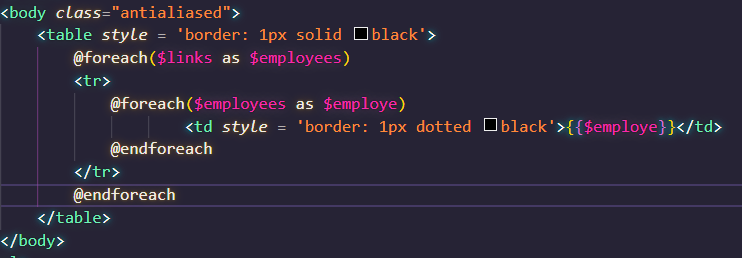
     <**td**>**user5**</**td**>

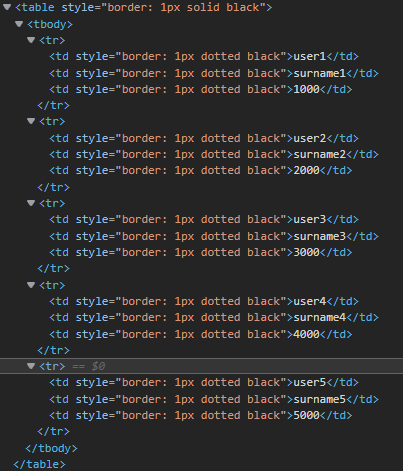
     <**td**>**surname5**</**td**>

     <**td**>5000</**td**>

  </**tr**>

</**table**





8. Модифицируйте предыдущую задачу так, чтобы в первом ряду появились заголовки таблицы:

<**table**>

  <**tr**>

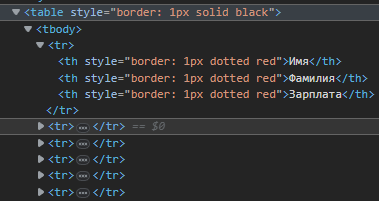
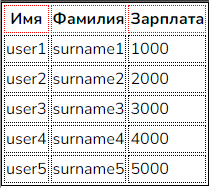
     <**th**>**Имя**</**th**>

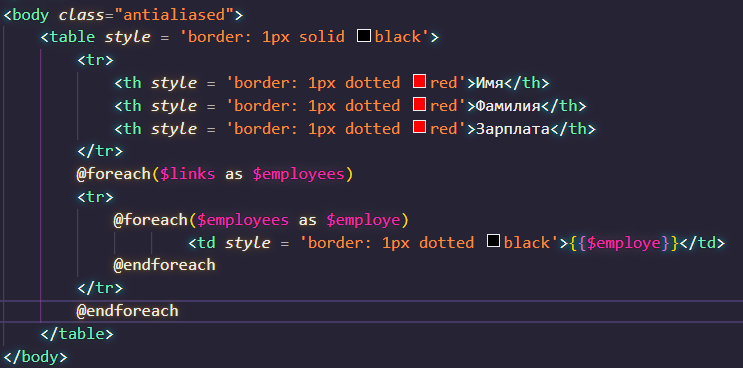
     <**th**>**Фамилия**</**th**>

     <**th**>**Зарплата**</**th**>

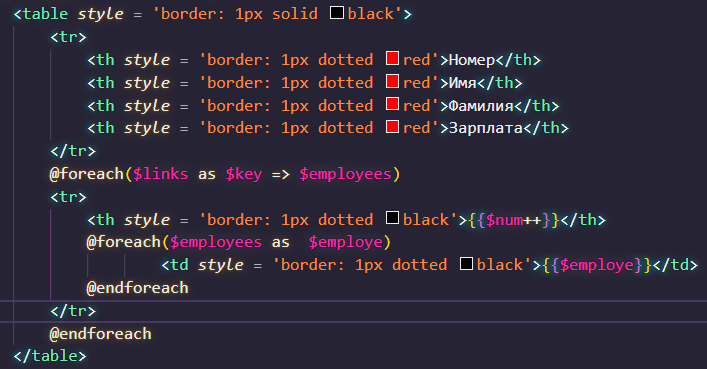
  </**tr**>

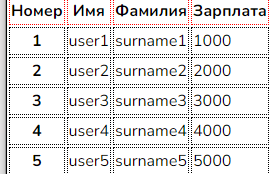
  ...



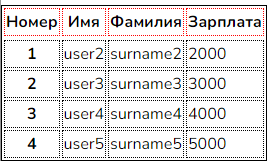


9. Модифицируйте предыдущую задачу так, чтобы появилась еще одна колонка в начале таблицы, в которой будут выводится порядковые номера работников (то есть ключи элементов массива). Нумерация должна начинаться с единицы, а не с нуля.





10.Модифицируйте предыдущую задачу так, чтобы на экран выводились только работники с зарплатой более **2000**.





11. Передайте из действия в представление массив с пользователей:

$users = [

   [

**'name'** => **'user1'**,

**'surname'** => **'surname1'**,

**'banned'** => **true**,

   ],

   [

**'name'** => **'user2'**,

**'surname'** => **'surname2'**,

**'banned'** => **false**,

   ],

   [

**'name'** => **'user3'**,

**'surname'** => **'surname3'**,

**'banned'** => **true**,

   ],

   [

**'name'** => **'user4'**,

**'surname'** => **'surname4'**,

**'banned'** => **false**,

   ],

   [

**'name'** => **'user5'**,

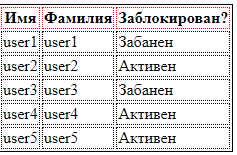
**'surname'** => **'surname5'**,

**'banned'** => **false**,

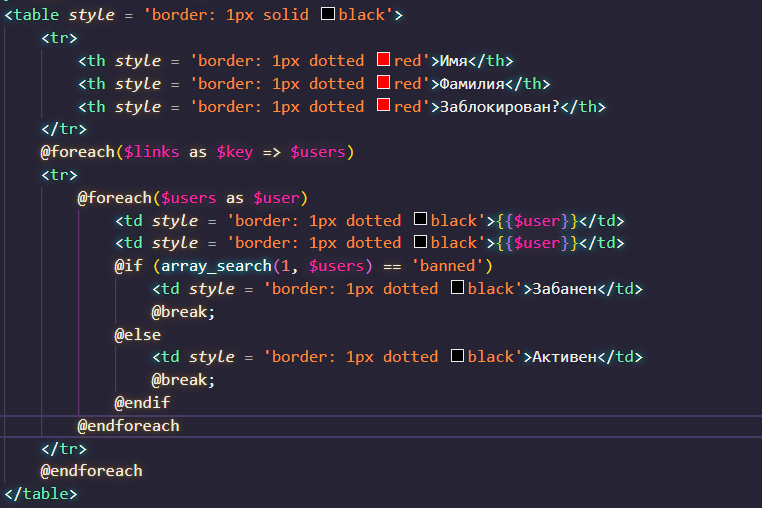
   ],

];

Выведите этот массив в виде HTML таблицы с тремя колонками: имя, фамилия и статус. В последней колонке должен выводится статус пользователя: забанен он или нет. Если ключ **banned** для данного пользователя имеет значение **true**, то выводите слово 'забанен', а если **false**, то слово 'активен'.





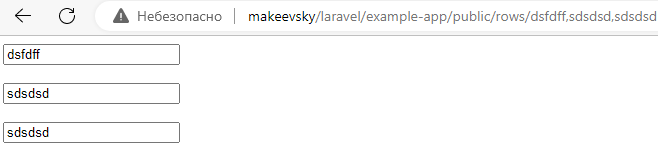


12. Модифицируйте предыдущую задачу так, чтобы цвет текста **tr** с забаненным пользователем был красного цвета, а с активным пользователем - зеленого.



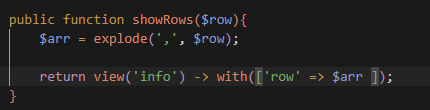


13. Из действия в представление передается массив со строками. Переберите этот массив циклом и в каждой итерации цикла создайте инпут, сделав значением инпута элемент массива.





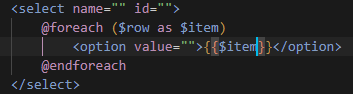




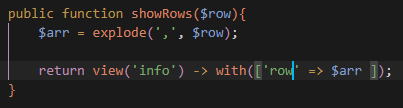
14. Из действия в представление передается массив со строками. Сделайте из этого массива выпадающий список **select**.



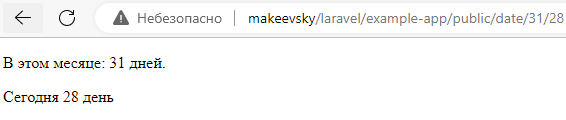






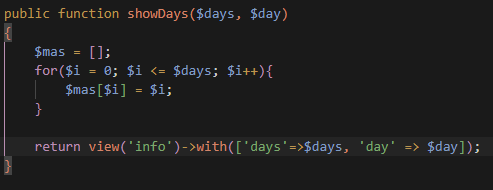


15. Сделайте в действии контроллера массив с числами от 1 до последнего дня текущего месяца. Передайте этот массив в представление. Сделайте также переменную, в которой будет хранится номер текущего дня. Также передайте эту переменную в представление.



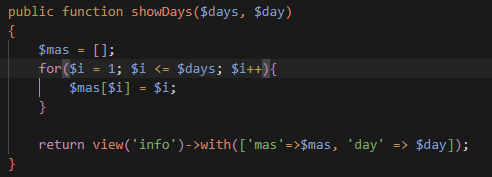
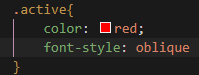
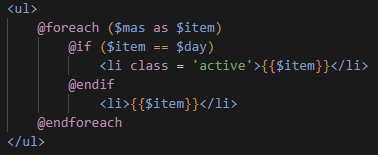






16. Переберите циклом переданный массив и выведите его в виде списка **ul**. При этом тегу **li**, в котором хранится номер текущего дня месяца добавьте класс **active**.



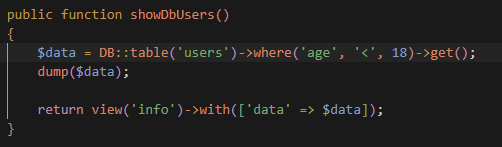


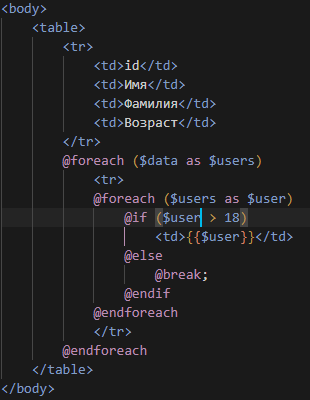
**Практическая работа 7. Работа с базой данных**

1. Сделайте через PhpMyAdmin таблицу **users** с полями **id**, **name** (имя), **surname** (фамилия), **age** (возраст). Добавьте в эту таблицу 5 юзеров.

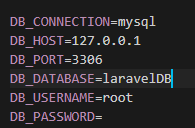


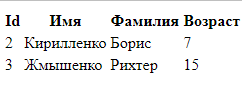
1. С помощью фасада **DB** получите всех юзеров, возраст которых более 18 лет. Передайте полученных юзеров в представление и выведите их в виде таблицы **table**.





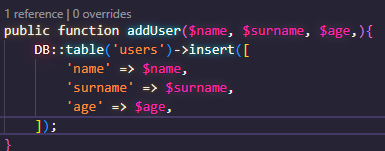






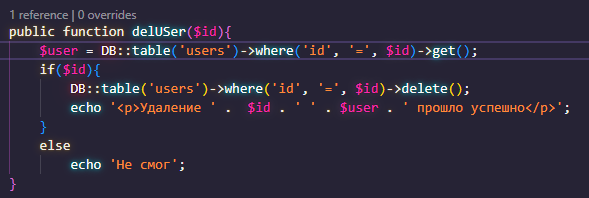
1. С помощью фасада **DB** добавьте нового юзера в таблицу **users**.





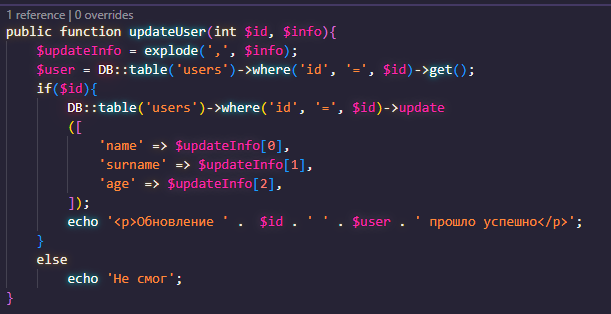
1. С помощью фасада **DB** удалите какого-нибудь нового юзера из таблицы **users**.



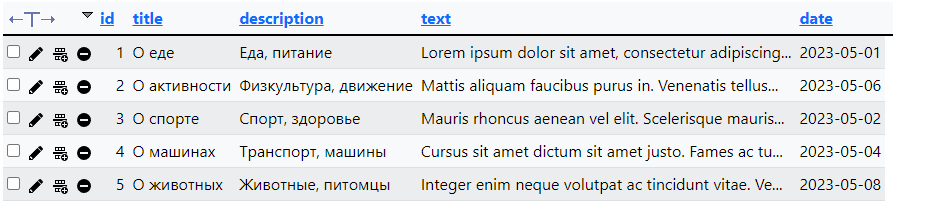
1. С помощью фасада **DB** поменяйте имя и фамилию какому-нибудь юзеру из таблицы **users**.

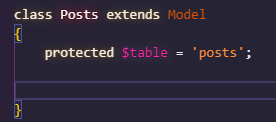


**Практическая работа Модели в Laravel**

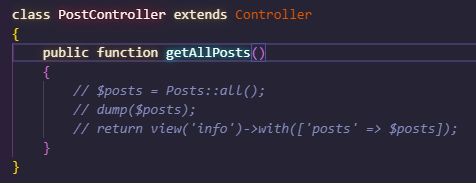
1. Сделайте через PhpMyAdmin таблицу **posts** (статьи) с полями **id**, **title** (заголовок), **desc** (короткое описание статьи), **text** (текст статьи), **date** (дата публикации). Добавьте в эту таблицу 5 статей.



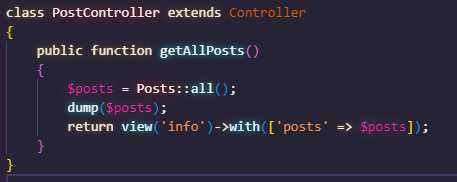
1. Сделайте модель Eloquent для таблицы **posts** из предыдущей задачи.



1. Сделайте контроллер **PostController** для работы со статьями.



1. В контроллере **PostController** сделайте действие **getAll** для получения списка всех статей. Пусть это действие будет доступно по адресу **/post/all/**.

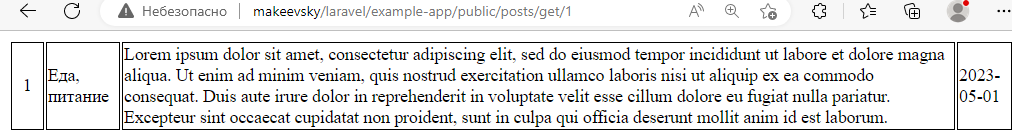


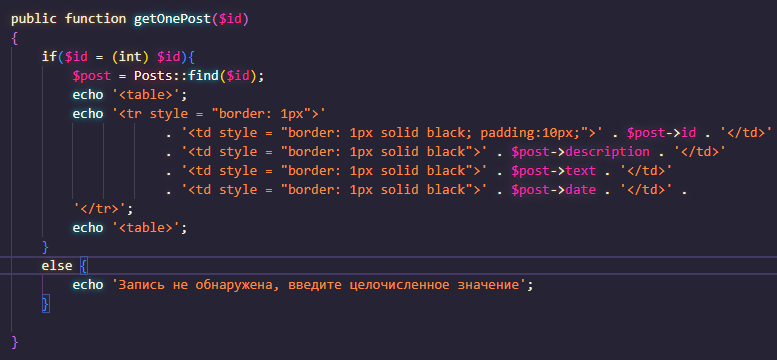
1. Используя модель **Post**, созданную в предыдущем уроке для таблицы **posts**, получите в действии **getAll** массив всех статей, передайте его в представление и выведите циклом на экран в виде HTML таблицы с колонками **id**, **title** (заголовок) и **desc** (то есть без текста статьи).





1. В контроллере **PostController** сделайте действие **getOne** для получения одной статьи по ее id. Пусть это действие будет доступно по адресу **/post/:id/**, где **:id** представляет собой **id** желаемой записи.Не забудьте наложить ограничение регулярным выражением на наш параметр.





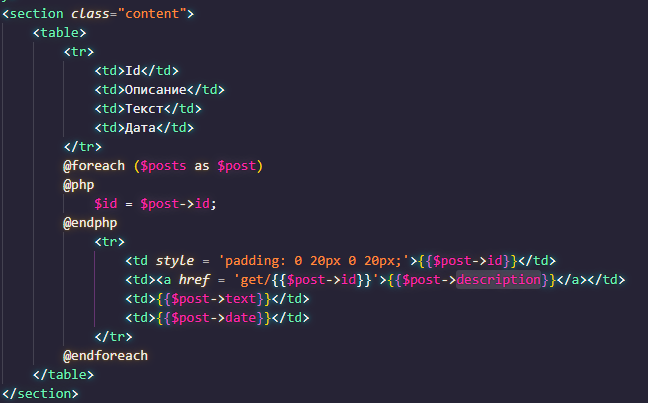


1. В действии **getOne** из таблицы **posts** получите статью соответствующую переданному параметру. Передайте полученную статью в представление и выведите ее на экран, оформив соответствующим HTML кодом.

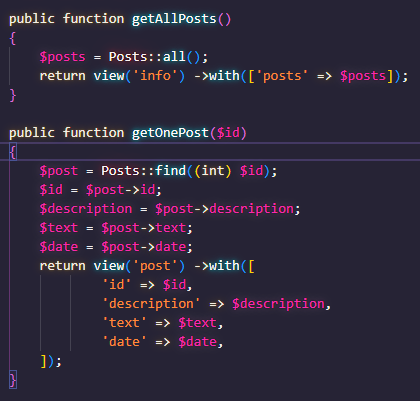
См 6, ???.

1. Отредактируйте представление действия **getAll** так, чтобы заголовок каждой из выводимых статей стал ссылкой на соответствующую страницу одной статьи.

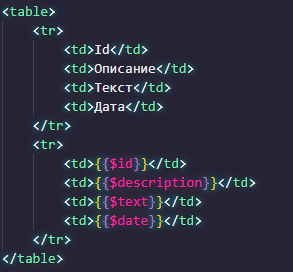
Все посты:

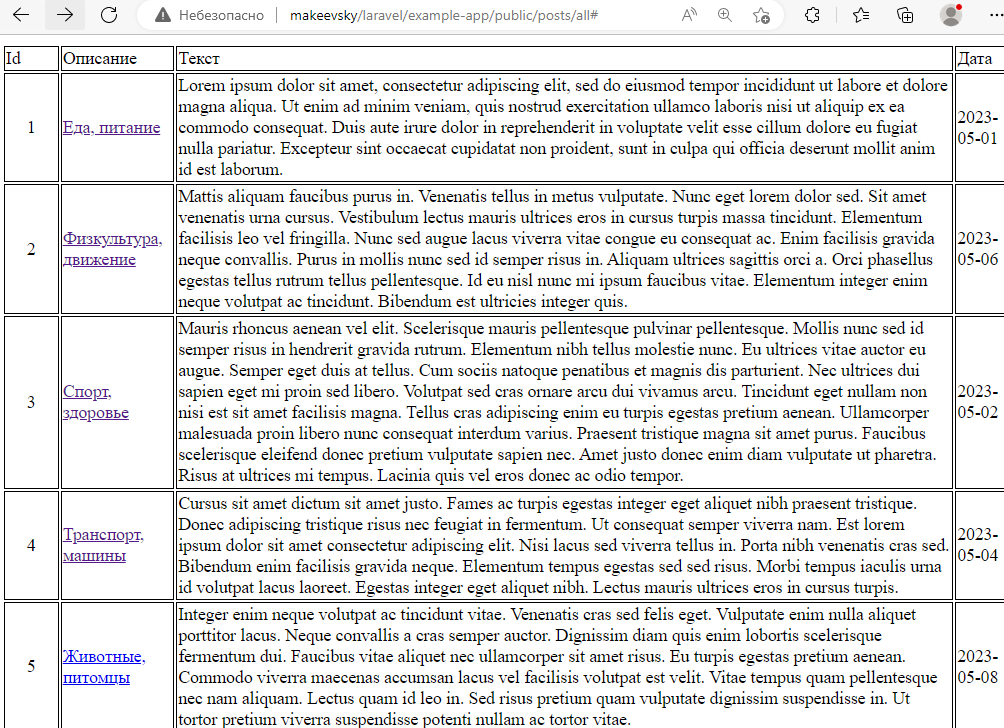


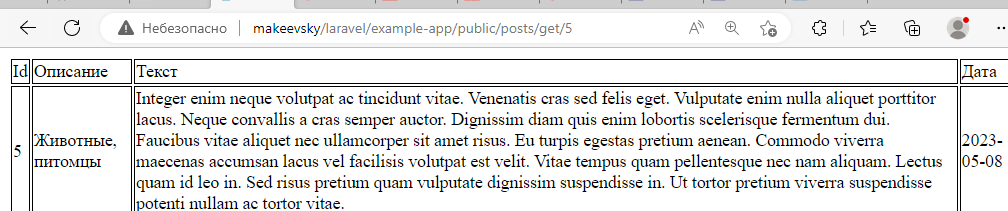
Контроллер:



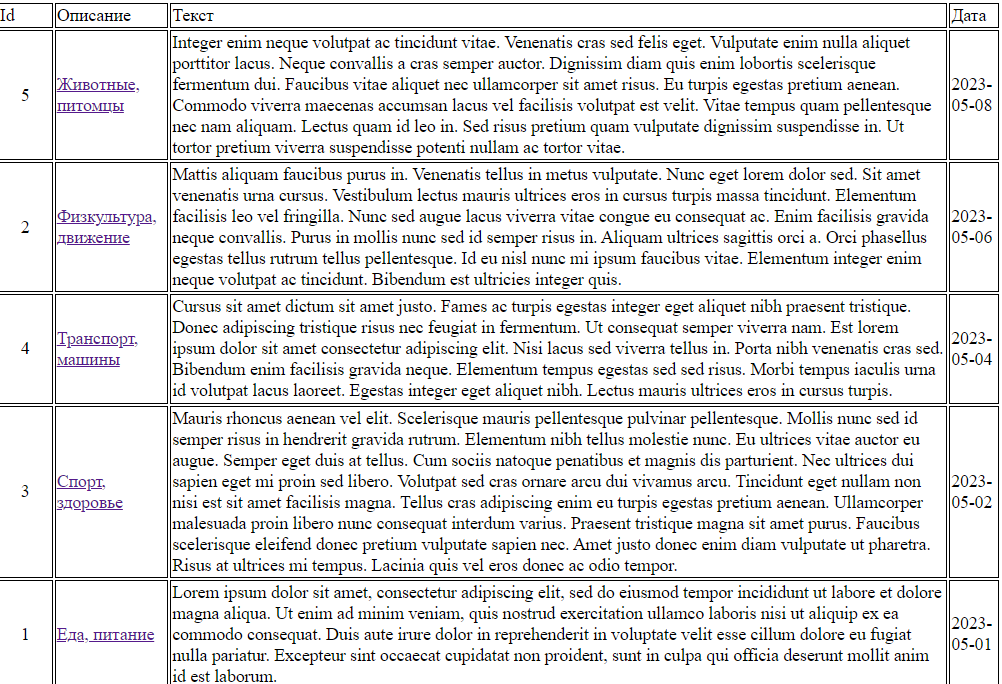
Вывод одного поста:

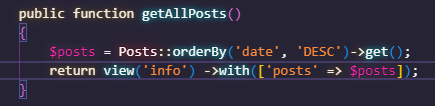






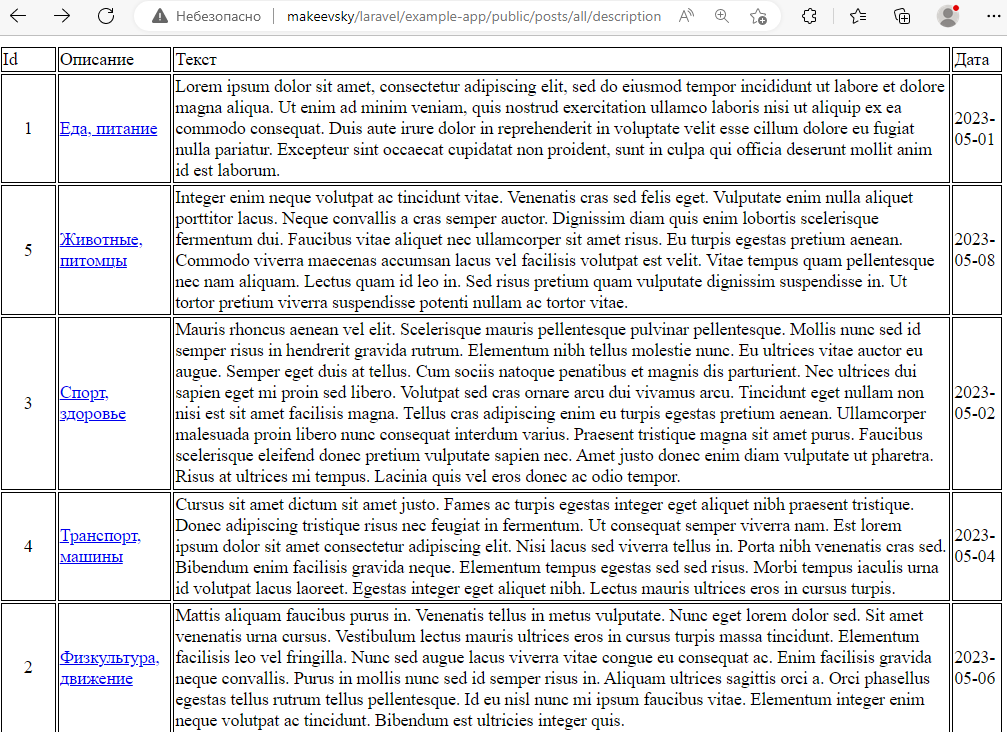
1. Отредактируйте действие **getAll** так, чтобы получаемые статьи были отсортированы по убыванию даты публикации.



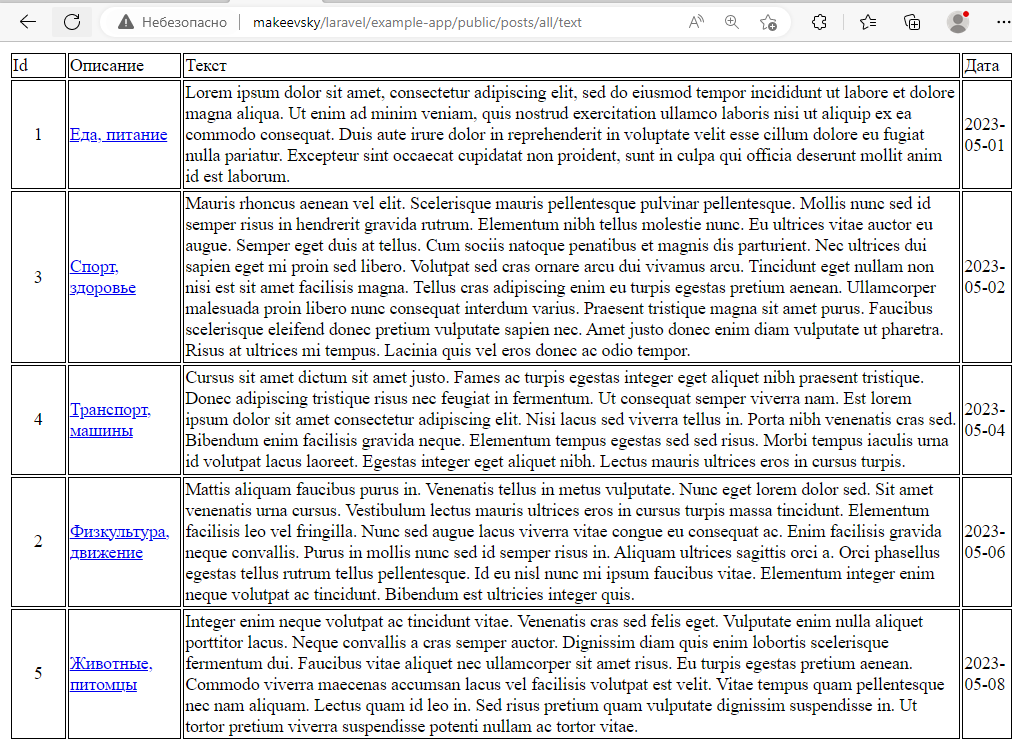


1. Отредактируйте маршрут действия **getAll** так, чтобы вместо адреса **/post/all/** нашей действие стало доступно по адресу **/post/all/:order/**, где **:order** представляет собой имя поля, по которому выполнять сортировку.Сделайте так, чтобы сортировать можно было по полям **id**, **title** и **date**. Причем сортировка должна быть по убыванию значения поля.Сделайте так, чтобы параметр **:order** был не обязательным и по умолчанию имел значение **date**.





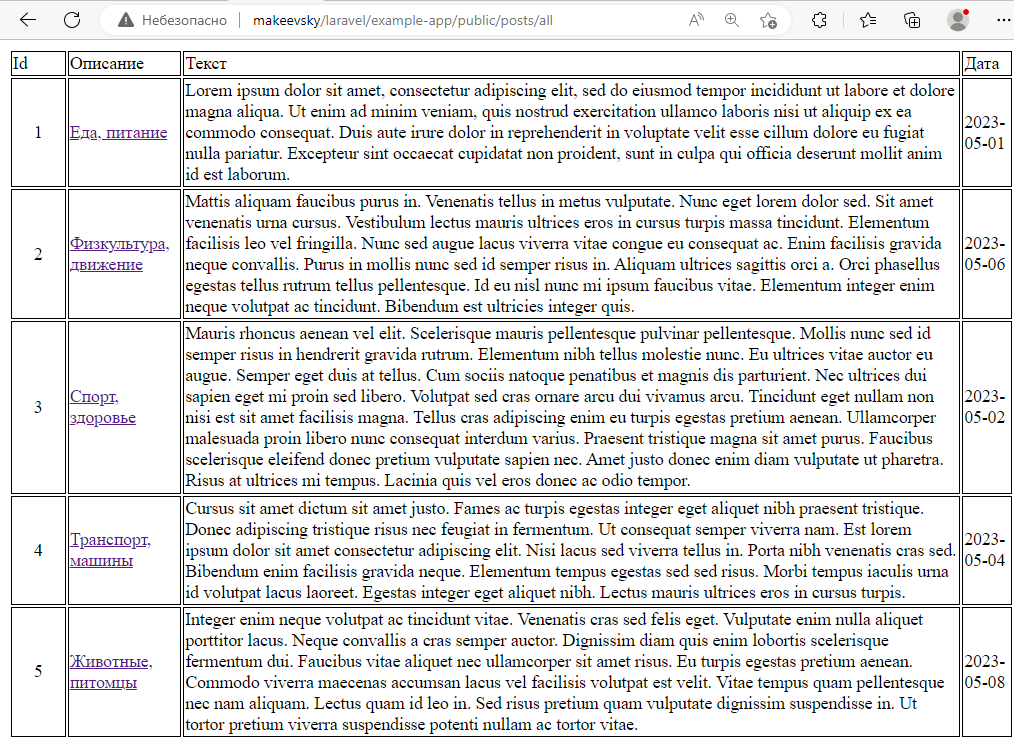
Если поле сортировки равно «text», то сортирует по Дате



1. Отредактируйте маршрут действия **getAll** так, чтобы появился еще один параметр **:dir**, представляющий собой направление сортировки (по убыванию или по возрастанию). То есть наш маршрут станет выглядеть так: **/post/all/:order/:dir**.Пусть параметр **:dir** может иметь только два значения: **asc** или **desc**. При этом пусть наш параметр также является не обязательным и по умолчанию имеет значение **desc**.



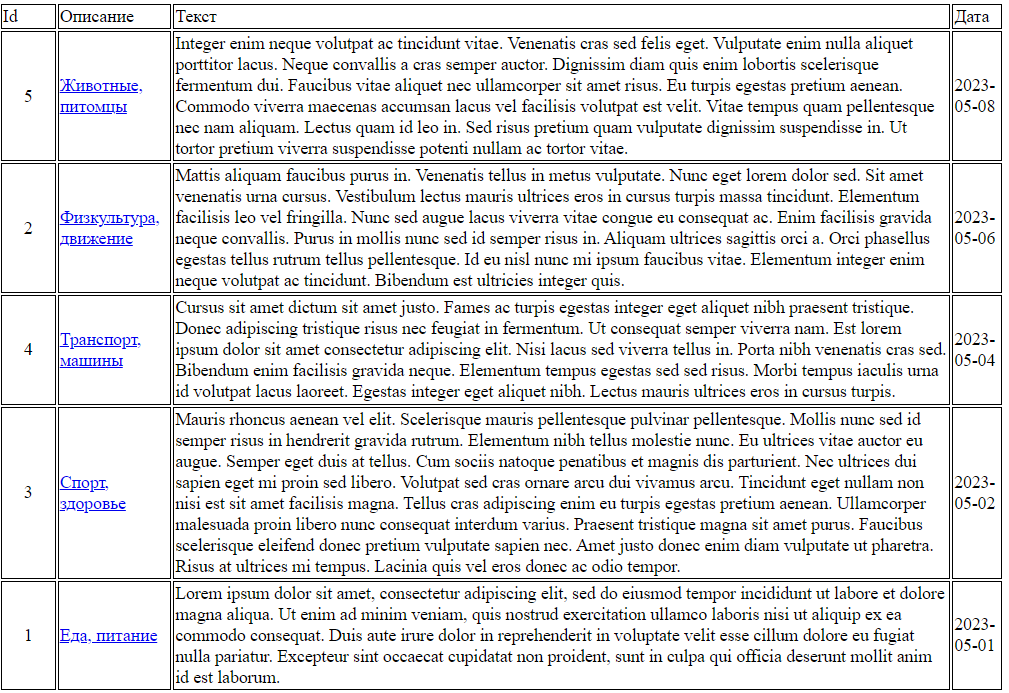
Без сортировки:



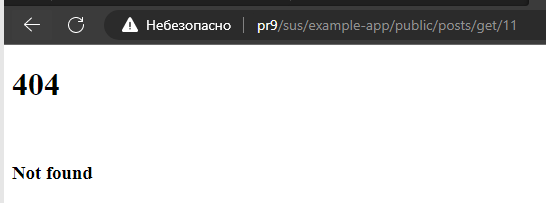
С сортировкой ASC:

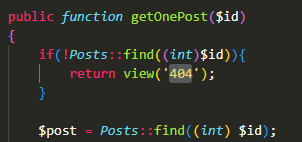


С сортировкой DESC:



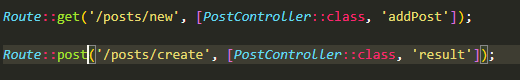
1. Отредактируйте действия **getOne** так, чтобы, если для переданного параметром id не существует записи в базе данных, то выводилась ошибка 404.





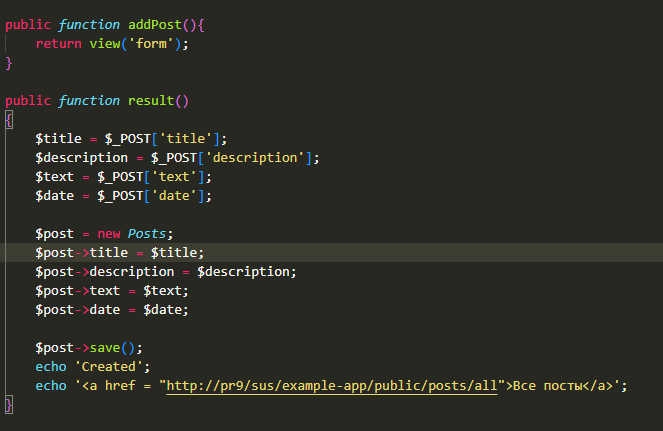
1. В контроллере **PostController**, созданном в предыдущем уроке, сделайте действие **newPost** для создания новой статьи. Пусть это действие будет доступно по адресу **/post/new/**. В представлении действия покажите форму для добавления новой записи. После отправки формы сохраните новую запись.



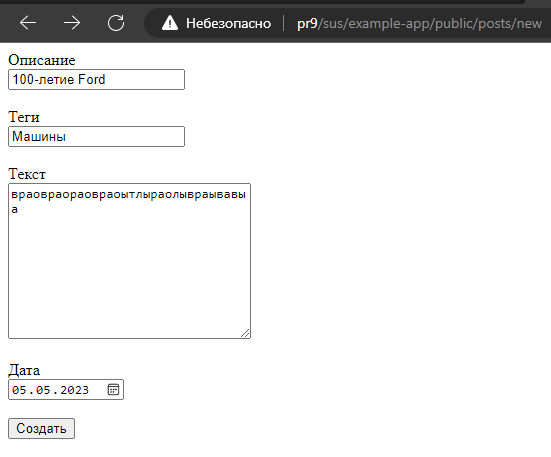


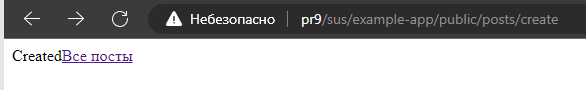
Info.blade.php

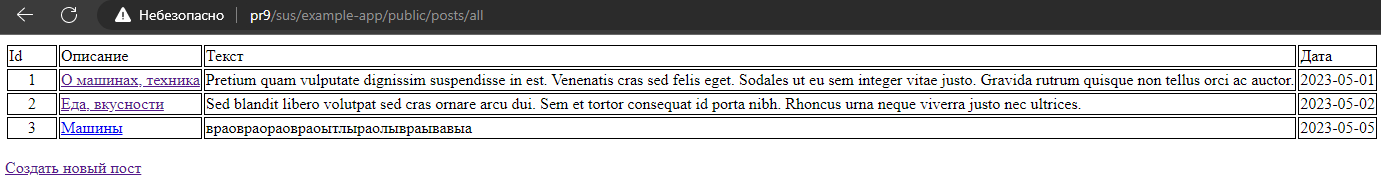




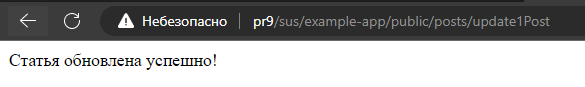
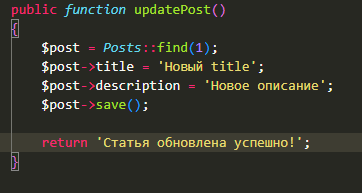








1. Сделайте какое-нибудь действие, по заходу на которое для статьи с id 1 поменяйте **title** и короткое описание **desc**.



1. Давайте теперь сделаем отдельное действие для редактирования статьи. Сделаем так, чтобы при заходе на это действие показывалась форма, уже заполненная данными из базы данных.При желании эти данные можно будет поменять и обновить их в базе данных.Итак, сделаем в контроллере **PostController** действие **editPost**:

**namespace** App\Http\Controllers;

**use** App\Http\Controllers\Controller;

**class** PostController **extends** Controller

{

**public function** editPost()

   {

**return** view(**'test.editPost'**);

   }

}

В представлении действия покажем форму для редактирования статьи:

<**form** action=**""** method=**"POST"**>

  <**input** name=**"title"**>

  <**input** name=**"desc"**>

  <**input** name=**"date"**>

  <**textarea** name=**"text"**></**textarea**>

  <**input** name=**"submit"** type=**"submit"**>

</**form**>

Так как форма отправляется методом **POST**, то для маршрута, соответствующего нашему действию, разрешим как метод **get**, так и **post**:

Route::match([**'get'**, **'post'**], **'post/edit/{id}'**, **'PostController@editPost'**);

Обратите внимание на то, что в адресе маршрута записан параметр **id**, представляющий собой **id** записи, редактирование которой мы осуществляем.

Давайте доработаем наш контроллер, подключив нужные классы и выполнив инъекцию класса Request, а также передав параметр **id** в метод действия **editPost**:

**namespace** App\Http\Controllers;

**use** App\Http\Controllers\Controller;

**use** Illuminate\Http\Request;

// подключаем класс Request

**use** App\Post;

// подключаем модель Post

**class** PostController **extends** Controller

{

**public function** editPost(Request $request, $id)

   {

**return** view(**'test.editPost'**);

   }

}

Давайте теперь сделаем так, чтобы по заходу на страницу данные записи с переданным **id** уже были записаны в форме.

Для этого получим запись по ее **id** и передадим полученные данные в представление:

**namespace** App\Http\Controllers;

**use** App\Http\Controllers\Controller;

**use** Illuminate\Http\Request;

// подключаем класс Request

**use** App\Post;

// подключаем модель Post

**class** PostController **extends** Controller

{

**public function** editPost(Request $request, $id)

   {

       $post = Post::find($id); // получаем запись по id

**return** view(**'test.editPost'**, [**'post'** => $post]);

   }

}

Выведем переданные данные в представлении:

<**form** action=**""** method=**"POST"**>

  <**input** name=**"title"** value=**"{{** $post->**title }}"**>

  <**input** name=**"desc"** value=**"{{** $post->**desc }}"**>

  <**input** name=**"date"** value=**"{{** $post->**date }}"**>

  <**textarea** name=**"text"**>{{ $post->**text** }}</**textarea**>

  <**input** name=**"submit"** type=**"submit"**>

</**form**>

Давайте теперь поймаем отправку формы. Здесь я использую хитрость: не буду проверять наличие всех полей формы, а просто проверю с помощью метода **has** наличие в **$request** поля **submit**. Данное имя соответствует кнопке отправки формы (я дал этой кнопке атрибут name в значении **submit**).

Итак, вот проверка:

**namespace** App\Http\Controllers;

**use** App\Http\Controllers\Controller;

**use** Illuminate\Http\Request;

**use** App\Post;

**class** PostController **extends** Controller

{

**public function** editPost(Request $request, $id)

   {

       $post = Post::find($id);

       // Если форма была отправлена...

**if** ($request->has(**'submit'**)) {

       }

**return** view(**'test.editPost'**, [**'post'** => $post]);

   }

}

Давайте теперь по отправке формы запишем данные формы в нашу модель и сохраним ее:

**namespace** App\Http\Controllers;

**use** App\Http\Controllers\Controller;

**use** Illuminate\Http\Request;

**use** App\Post;

**class** PostController **extends** Controller

{

**public function** editPost(Request $request, $id)

   {

       $post = Post::find($id);

       // Если форма была отправлена...

**if** ($request->has(**'submit'**)) {

           $post->**title** = $request->**title**;

           $post->**desc** = $request->**desc**;

           $post->**date** = $request->**date**;

           $post->**text** = $request->**text**;

           $post->save(); // сохраним модель

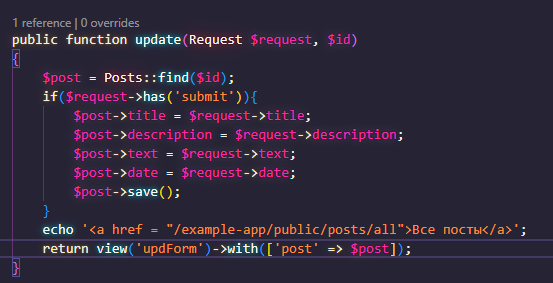
       }

**return** view(**'test.editPost'**, [**'post'** => $post]);

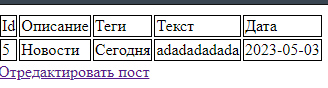
   }

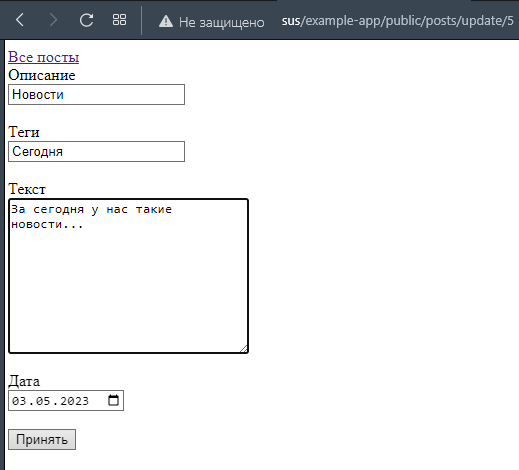
}

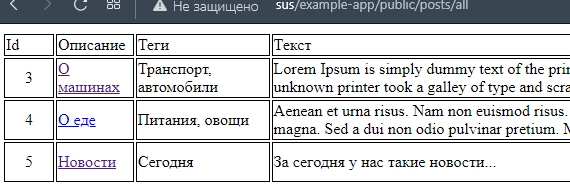
Обратите внимание на то, что после попадания в иф, поля нашей модели **$post** изменятся и уже в измененном виде попадут в представление - то есть в форме после ее отправки уже будут стоять не изначальные значения из БД, а те, которые вбил пользователь. Вот такой хитрый прием.





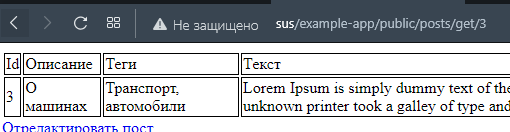




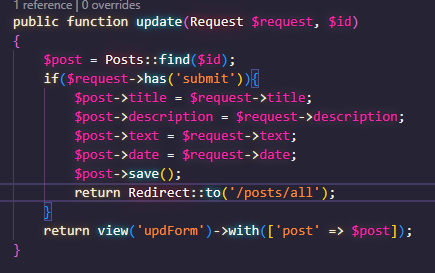


16. Отредактируйте представление действия **getAll** так, чтобы появилась еще одна колонка со ссылкой на редактирование соответствующей статьи.

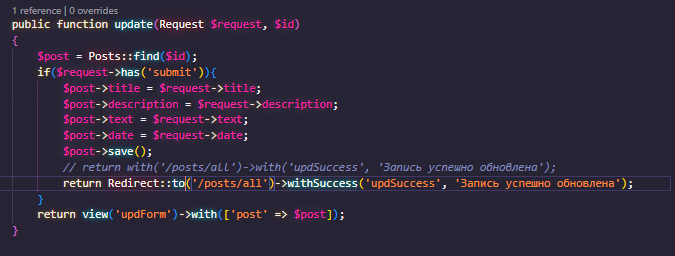


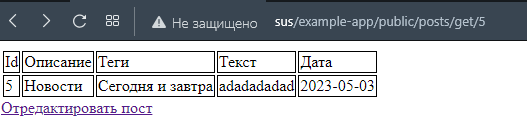


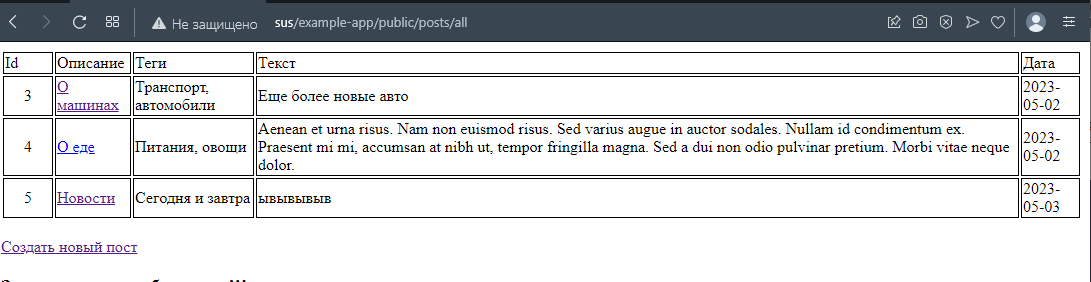
17. Модифицируйте код действия **editPost** так, чтобы после сохранения формы выполнялся редирект на список всех записей (то есть на действие **getAll**).



18. Модифицируйте предыдущую задачу так, чтобы при редиректе отправлялось флеш сообщение об успешном обновлении записи. Выводите это сообщение в представлении действия **getAll**.

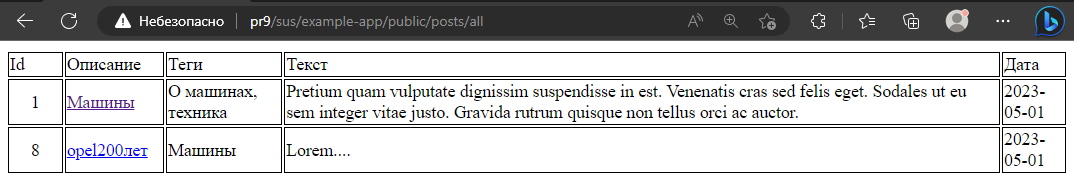


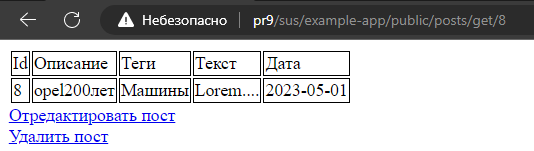


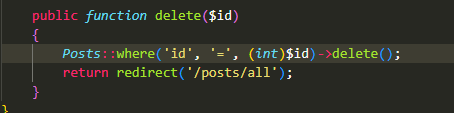
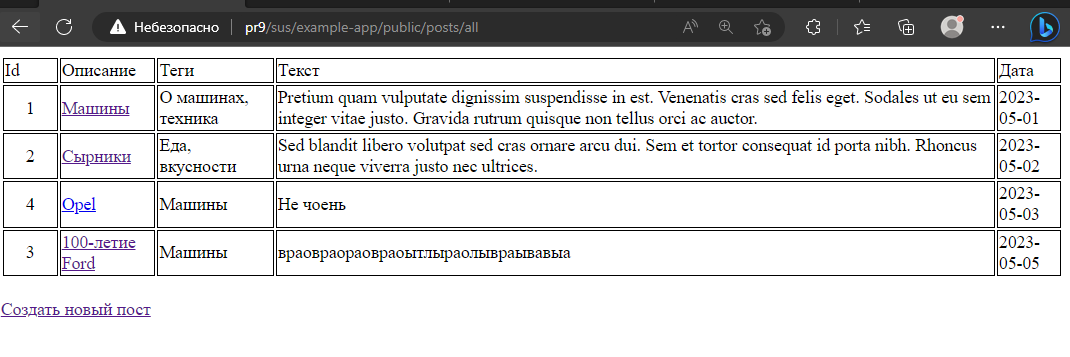


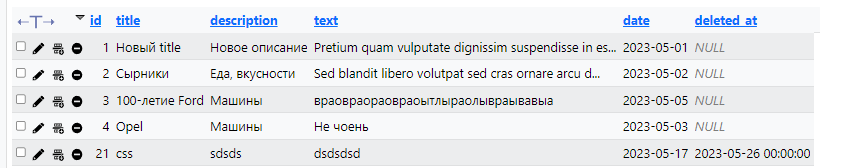
19. Модифицируйте предыдущую задачу так, чтобы во флеш сообщении был указан id и title статьи, подвергнувшейся изменению.

20. В контроллере **PostController**, созданном в предыдущих уроках, сделайте действие **delPost** для удаления статьи. Пусть это действие будет доступно по адресу **/post/del/:id**, где **:id** будет содержать **id** статьи для удаления.



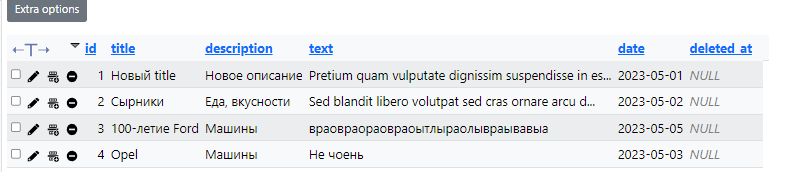






21. Отредактируйте представление действия **getAll** так, чтобы появилась еще одна колонка со ссылкой на удаления соответствующей статьи. После удаления выполняйте редирект обратно на список статей с флеш сообщением об успешном удалении статьи. Сообщение должно содержать **title** удаленной статьи.

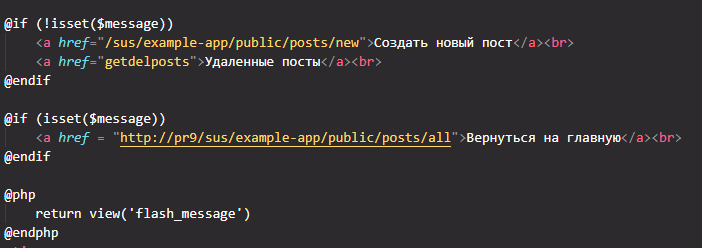
22. Реализуйте мягкое удаление статей. Мягкое удаление означает, что запись на самом деле остаётся в базе данных, но в таблице для записи устанавливается поле deleted\_at.

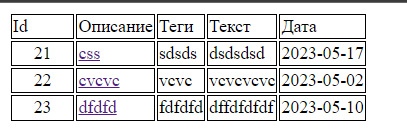


23. Сделайте действие **getDeletedPost**, выводящее список удаленных статей.



Главный блейд



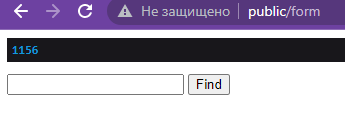
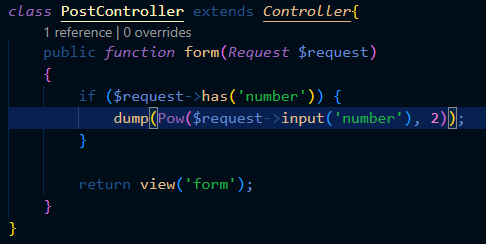


24. Сделайте действие **restorePost**, восстанавливающее удаленную статью обратно.

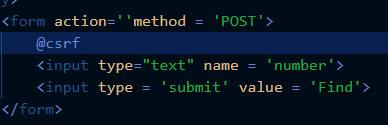
25. В представлении действия **getDeletedPost** для каждой статьи сделайте ссылку на ее восстановление.

**9. Работа с формами Laravel**

1. Сделайте форму с инпутом, принимающим своим значением число. После отправки формы выведите через **var\_dump** квадрат этого числа

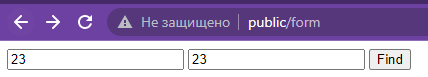


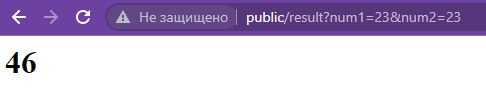
 

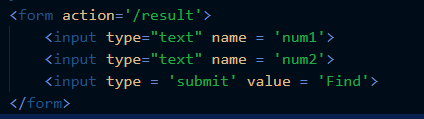
1. Модифицируйте предыдущую задачу так, чтобы квадрат переданного числа выводился в представлении над формой.

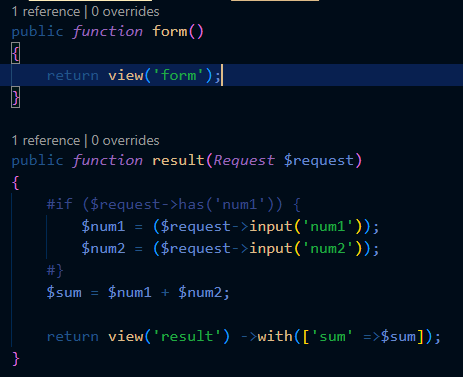


1. Сделайте форму с двумя инпутами, в которые будут вводится числа. После отправки формы выведите на экран сумму этих чисел, а форму уберите.



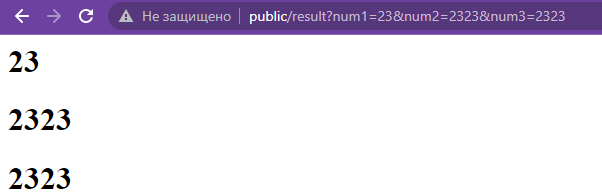


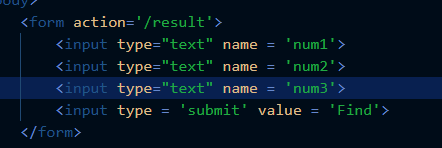
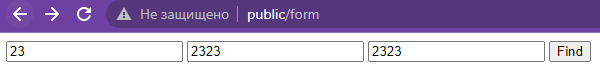




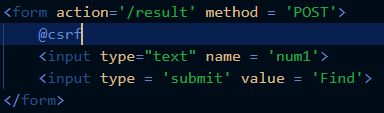


1. В каком-нибудь действии сделайте форму с тремя инпутами, в которые будут вводится числа. Отправьте эту форму на другое действие и выведите переданные числа в представлезачнии этого действия.





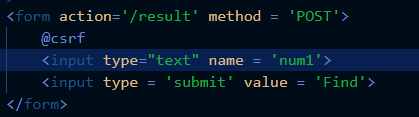
1. Сделайте два действия. В представлении первого действия покажите форму, которая будет отправляться на второе действие методом **POST**. Выведите данные, отправленные через форму, в представлении второго действия.

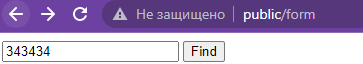


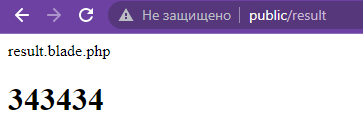




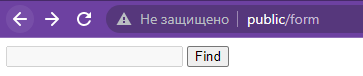


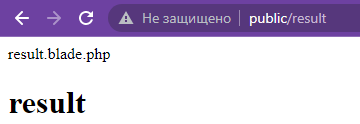


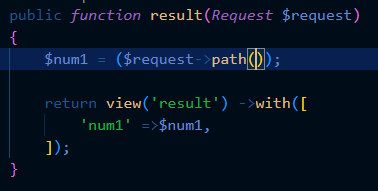




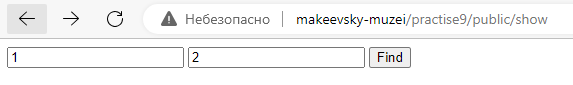
1. Выведите на экран метод HTTP запроса.



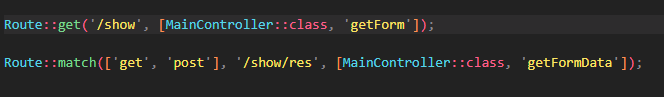




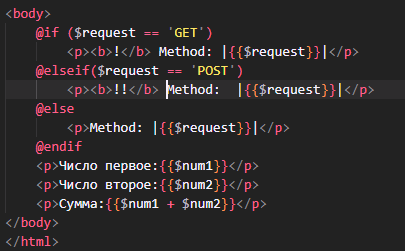
1. Если метод HTTP запроса - **GET**, то выведите на экран сообщение '!', а если **POST**, то сообщение '!!'.

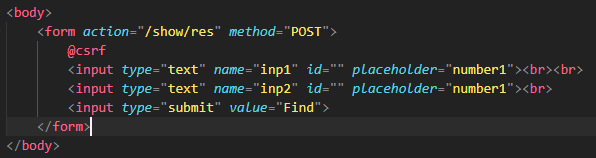


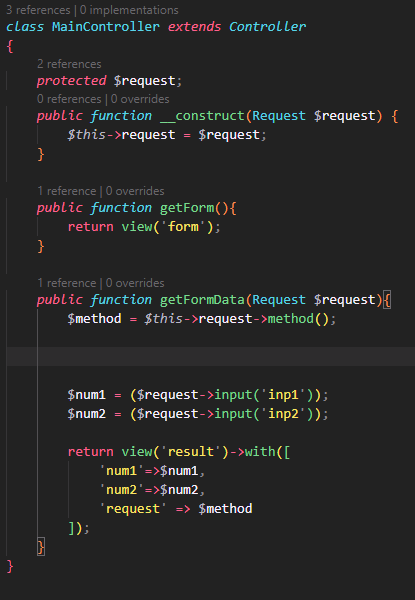




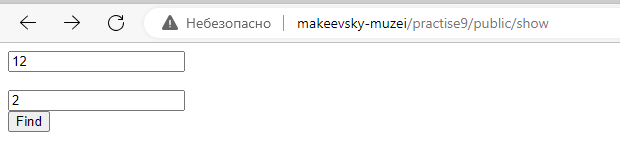
Result.blade.php

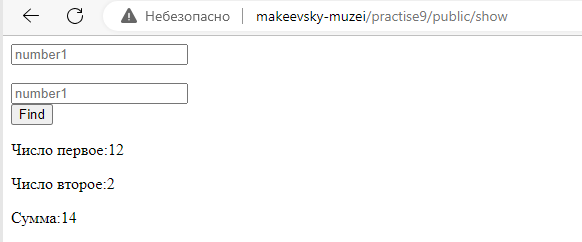




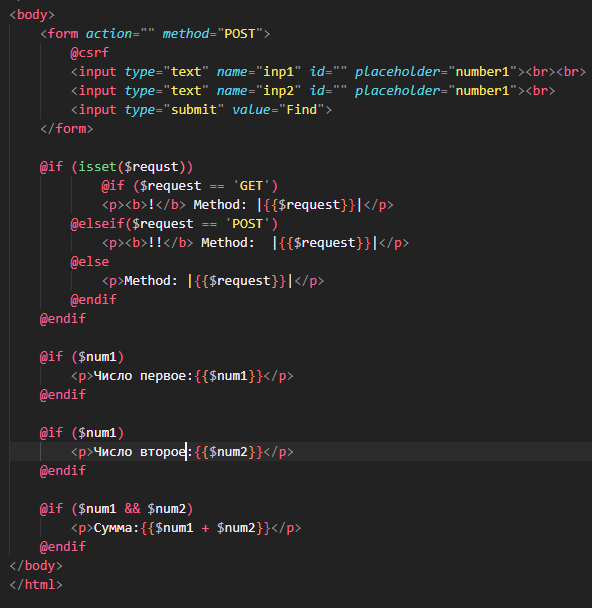


1. Сделайте форму, которая будет спрашивать у пользователя два числа с помощью двух инпутов. После отправки формы методом **POST** выведите на экран сумму введенных чисел. Пусть и форма, и результат показываются одним действием.

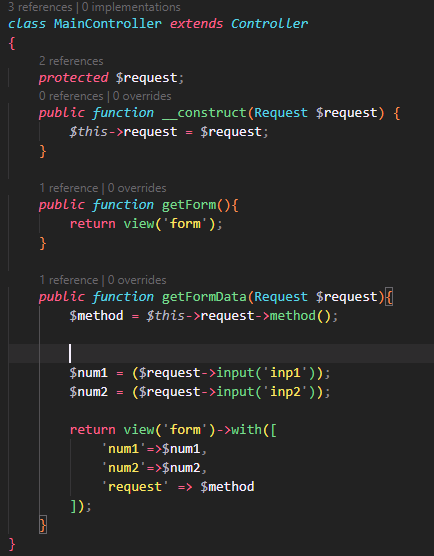


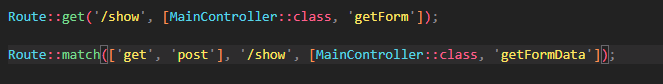


Form.blade.php

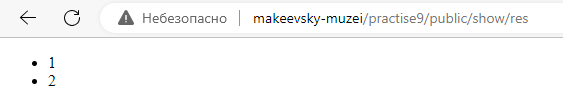


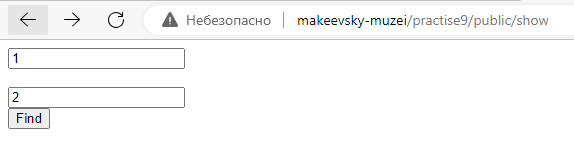
Controller:

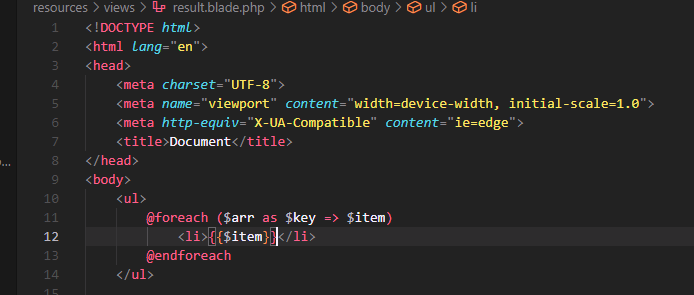


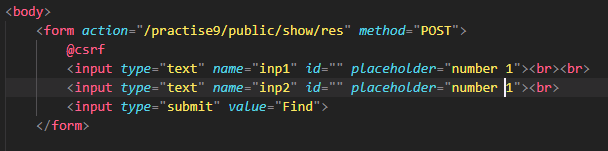


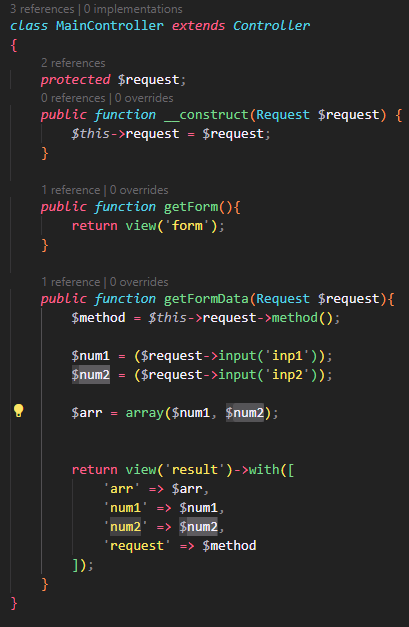
1. Пусть в вашей форме есть произвольное количество инпутов. После отправки формы получите массив отправленных значений, отправьте его в представление и выведите эти данные в виде списка **ul**.

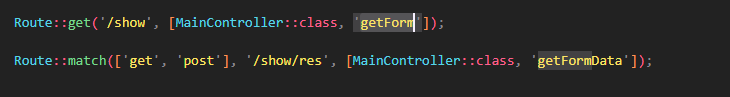












1. Пусть у нас будет форма, имитирующая регистрацию пользователя на сайте. Спросите с помощью этой формы у пользователя его имя, фамилию, email, желаемый логин, желаемый пароль (инпут с типом password). После отправки формы выведите на экран в виде списка **ul** все отправленные поля, кроме поля с паролем и email. Решите задачу сначала через **except**, а затем через **only**.