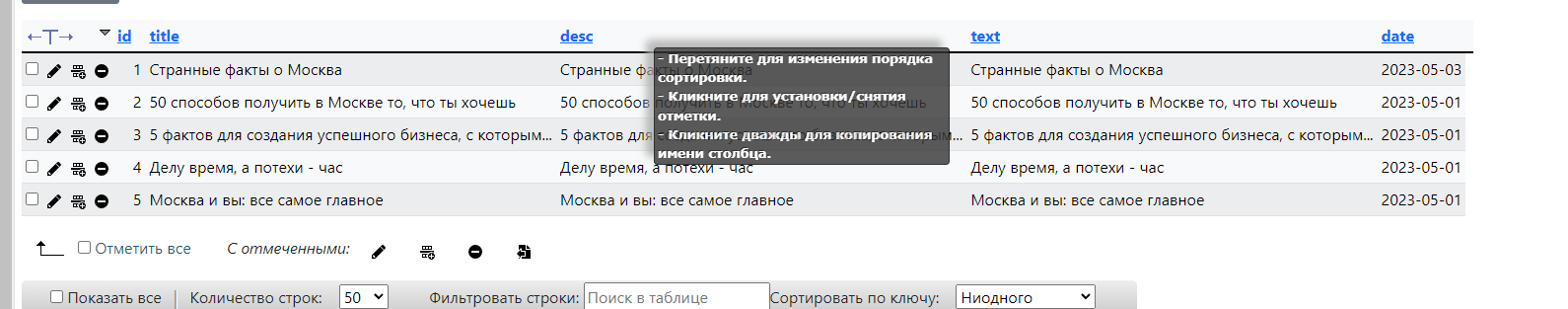
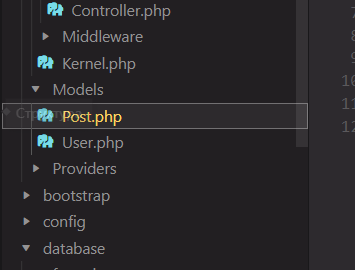
**Отчет по практической работе №9 по LARAVEL   
Плотников Никита, ИСиП-20-302**

**Задание №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



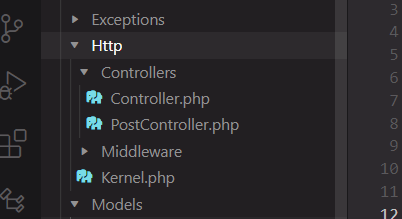
**Задание №2**

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



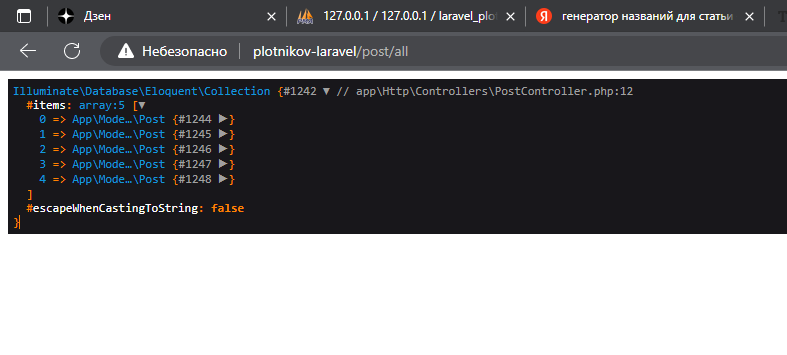
**Задание №3**

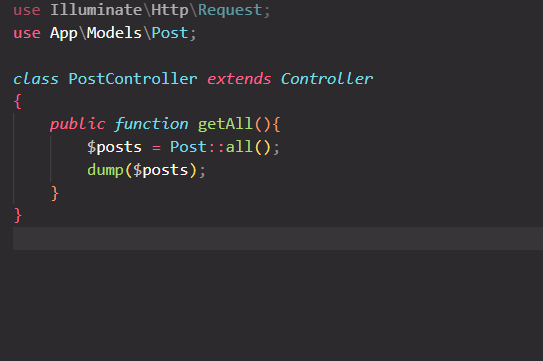
Сделайте контроллер **PostController** для работы со статьями.



**Задание №4**

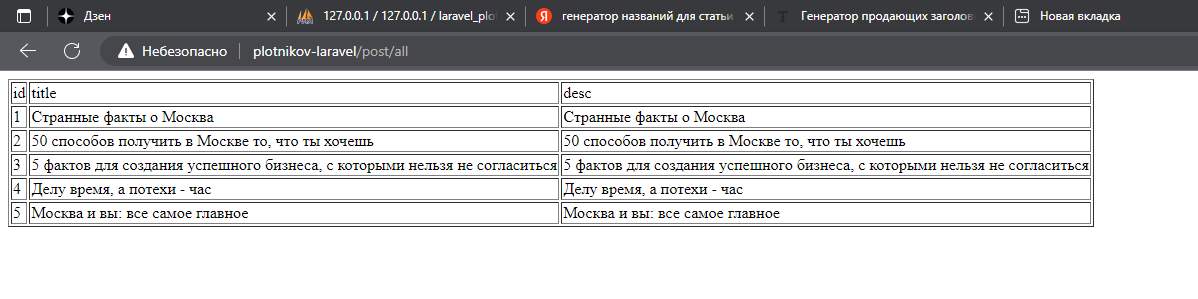
В контроллере **PostController** сделайте действие **getAll** для получения списка всех статей. Пусть это действие будет доступно по адресу **/post/all/**.



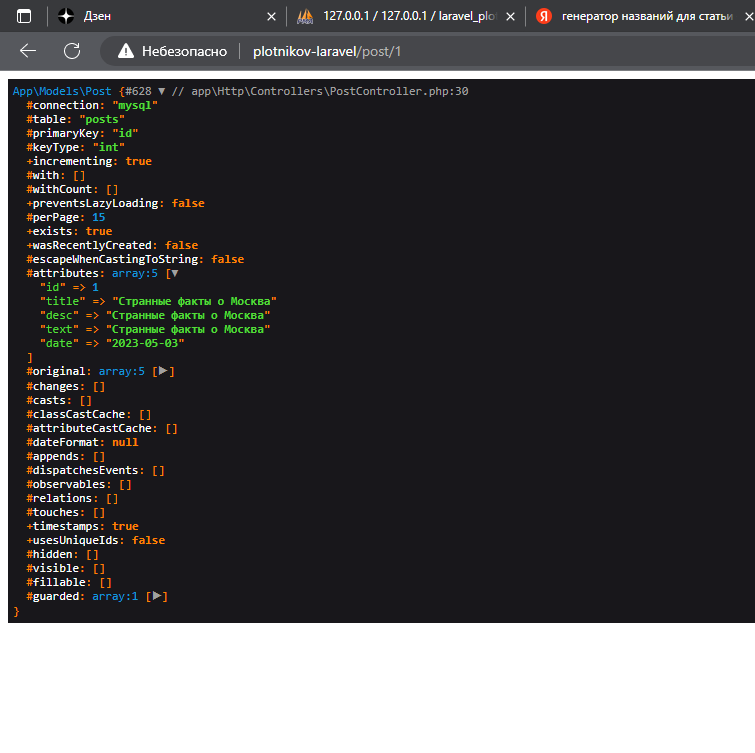
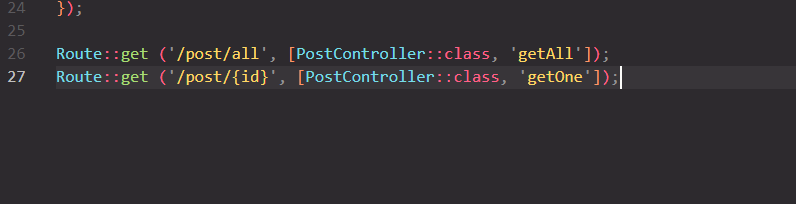
**Задание №5**

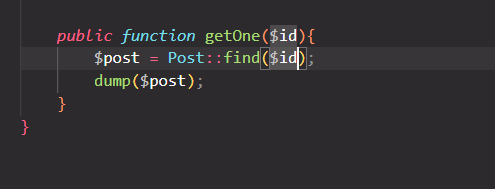
Используя модель **Post**, созданную в предыдущем уроке для таблицы **posts**, получите в действии **getAll** массив всех статей, передайте его в представление и выведите циклом на экран в виде HTML таблицы с колонками **id**, **title** (заголовок) и **desc** (то есть без текста статьи).

**Задание №6**

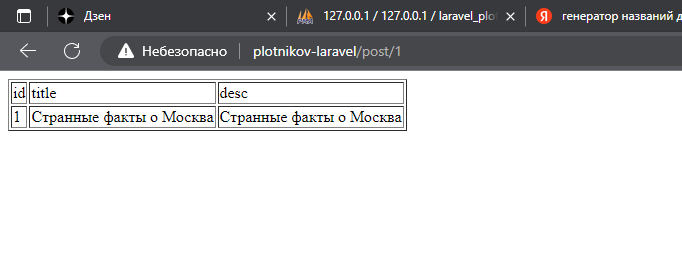
В контроллере **PostController** сделайте действие **getOne** для получения одной статьи по ее id. Пусть это действие будет доступно по адресу **/post/:id/**, где **:id** представляет собой **id** желаемой записи.Не забудьте наложить ограничение регулярным выражением на наш параметр.



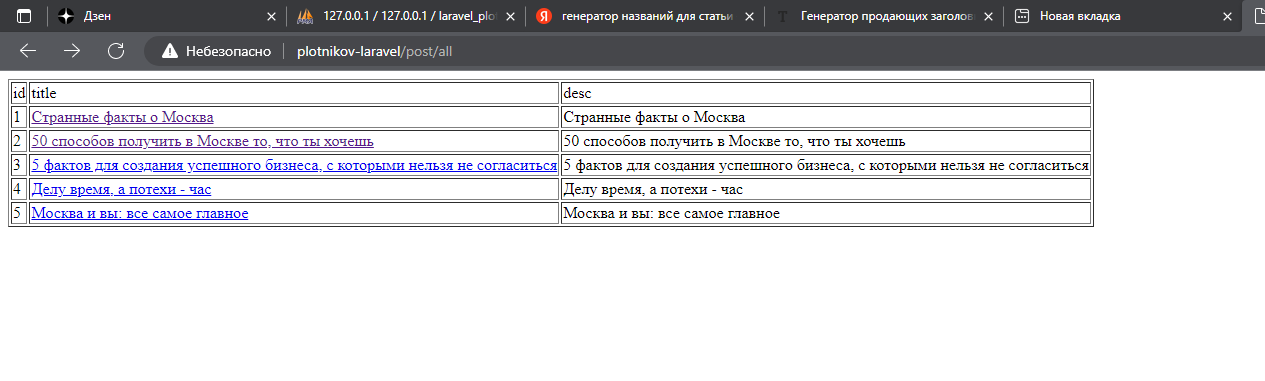
**Задание №7**

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.

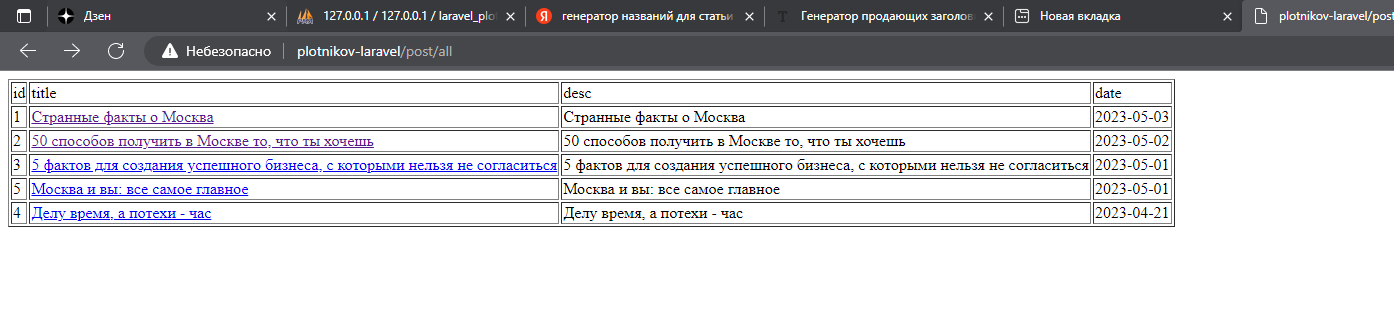
**Задание №8**

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

**Задание №9**

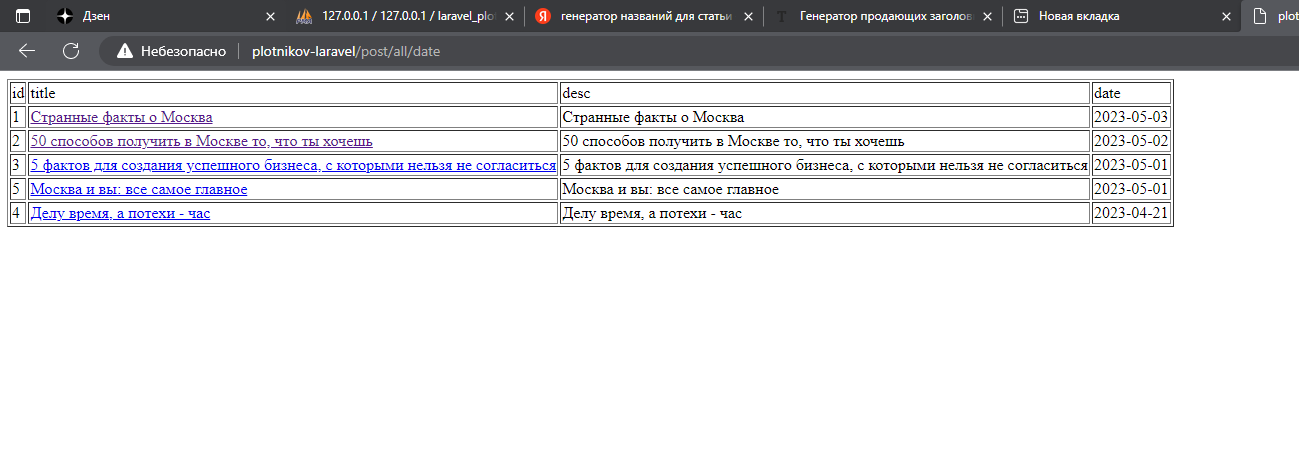
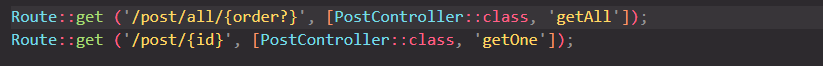
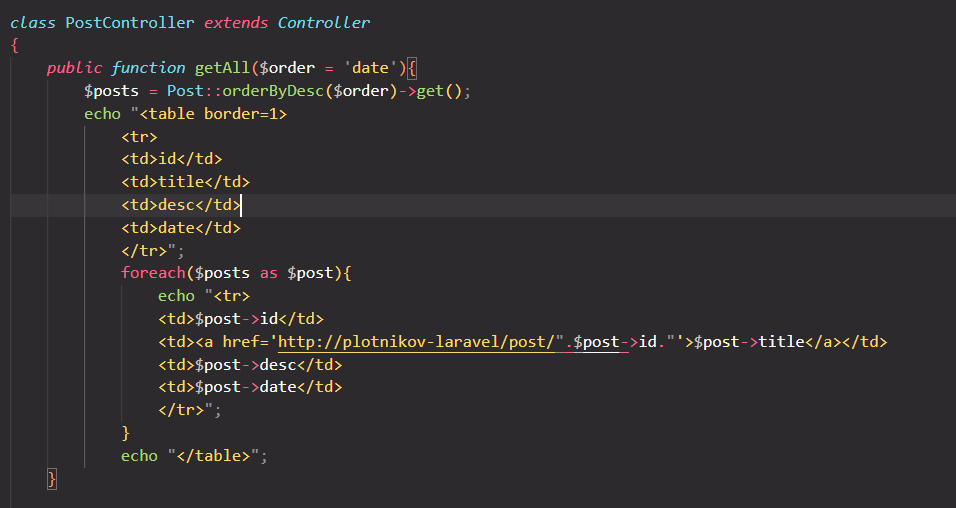
Отредактируйте действие **getAll** так, чтобы получаемые статьи были отсортированы по убыванию даты публикации.





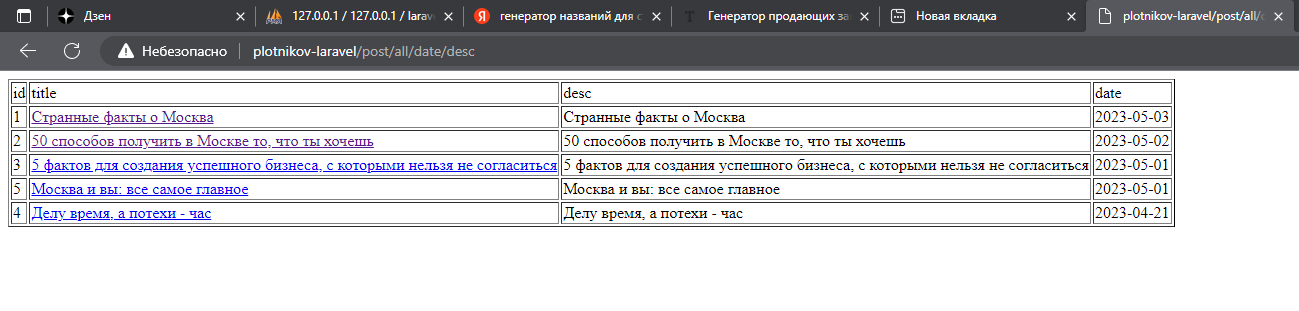
**Задание №10**

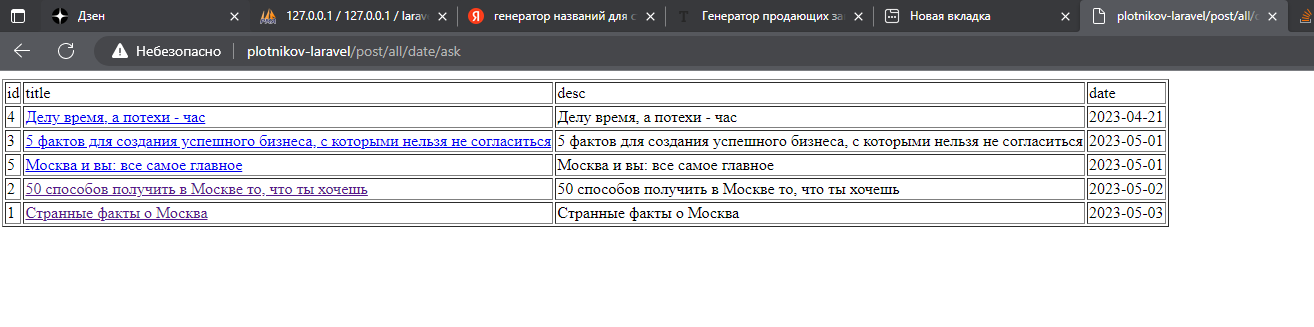
Отредактируйте маршрут действия **getAll** так, чтобы вместо адреса **/post/all/** нашей действие стало доступно по адресу **/post/all/:order/**, где **:order** представляет собой имя поля, по которому выполнять сортировку.Сделайте так, чтобы сортировать можно было по полям **id**, **title** и **date**.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

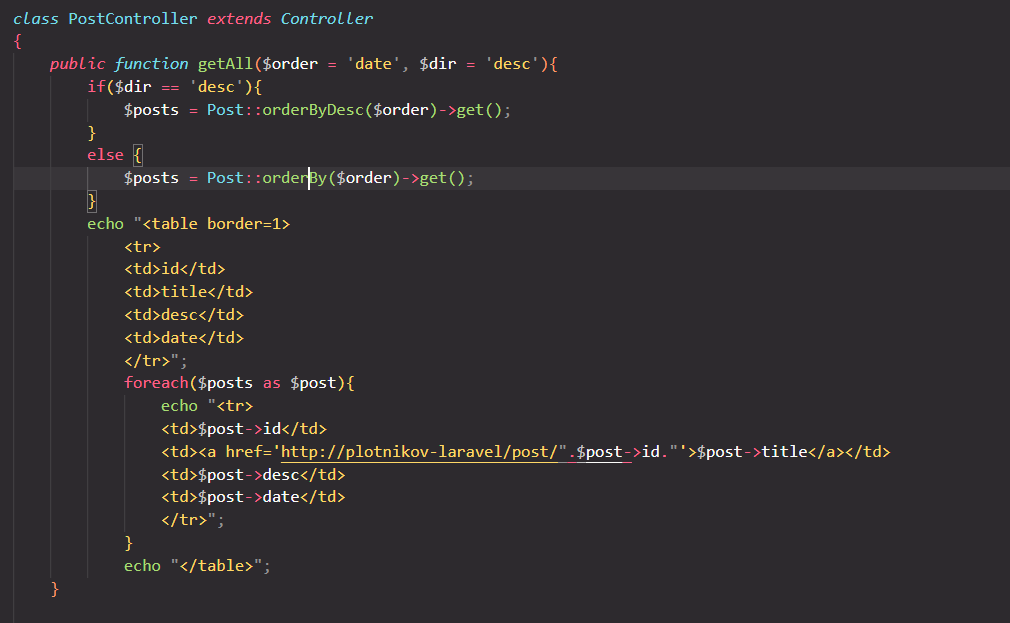


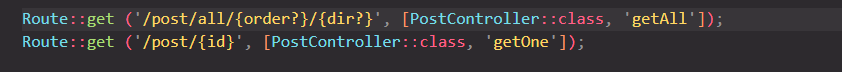
**Задание №11**

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**.



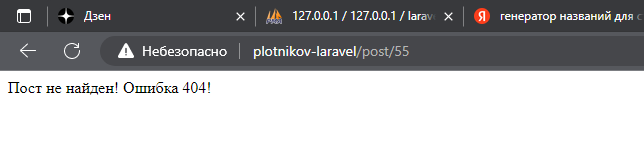


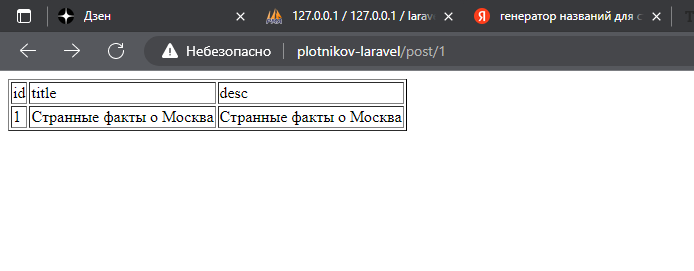




**Задание №12**

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

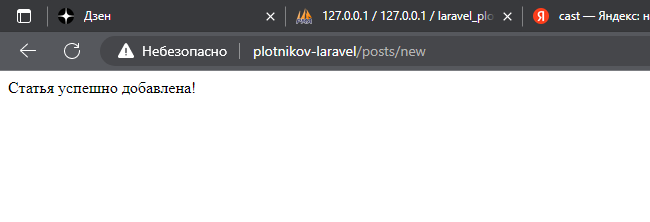




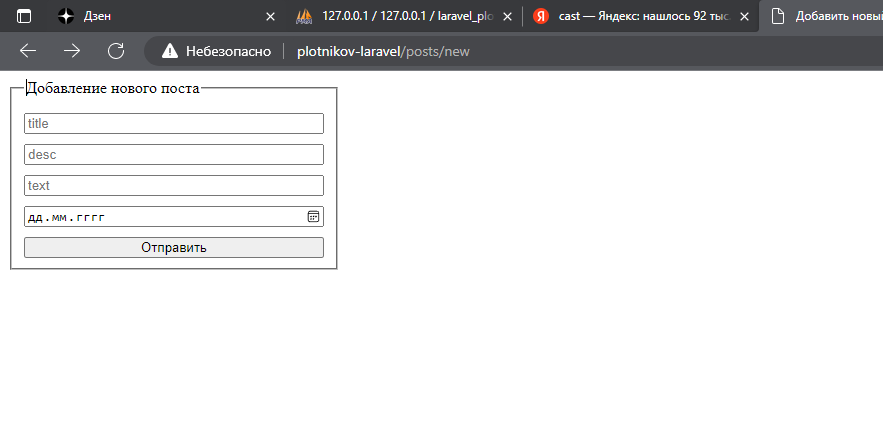


**Задание №13**

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.

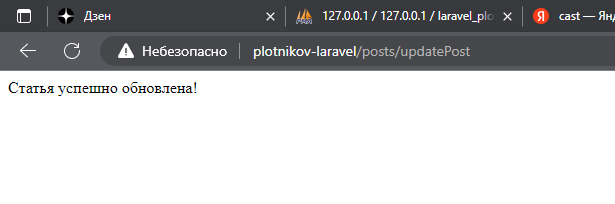
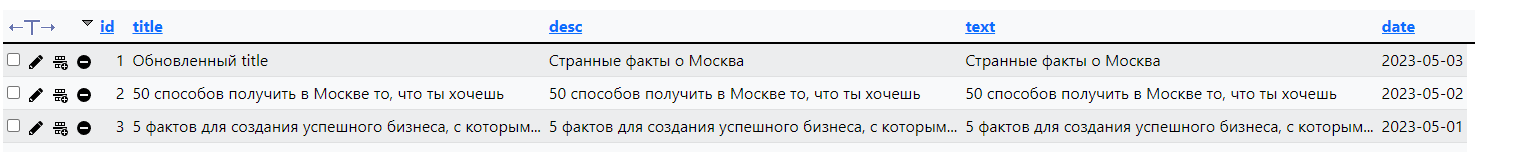


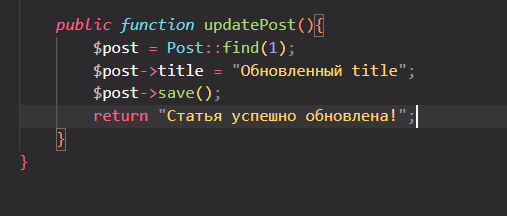




**Задание №14**

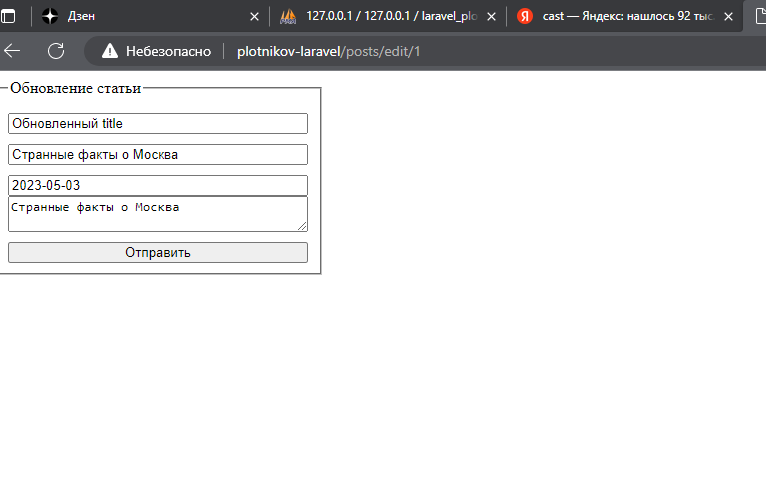
Сделайте какое-нибудь действие, по заходу на которое для статьи с id 1 поменяйте **title** и короткое описание **desc**.

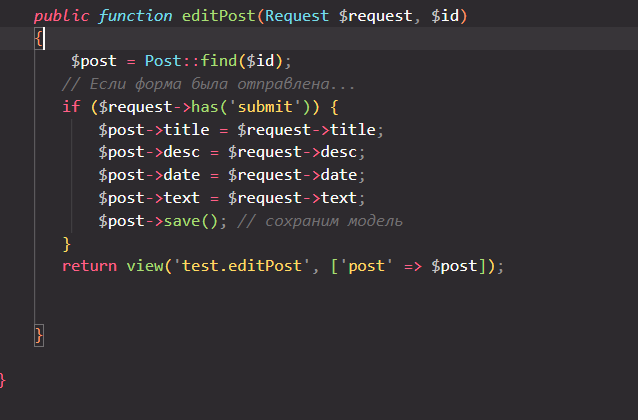
 

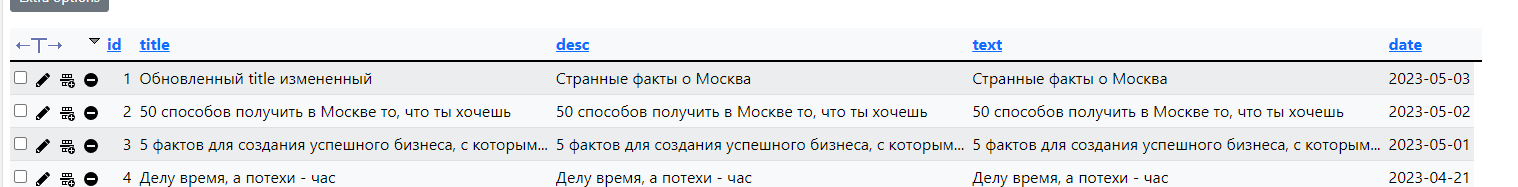


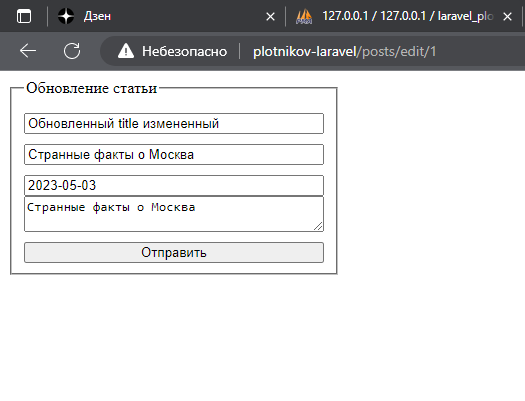
**Задание №15**

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.





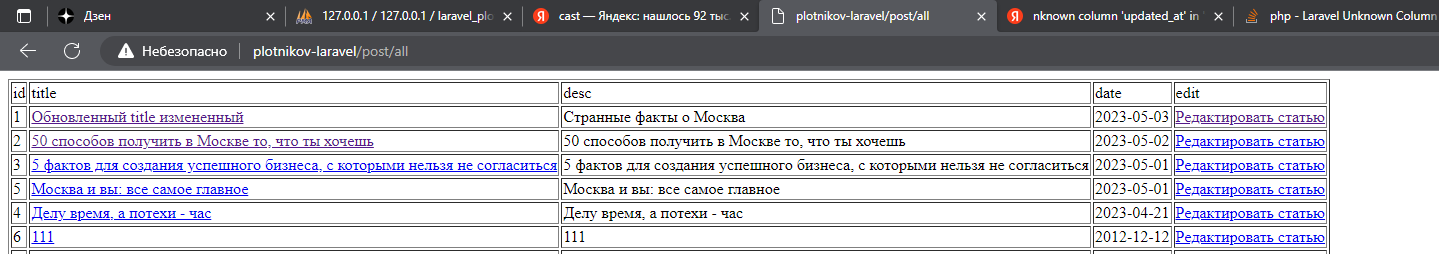


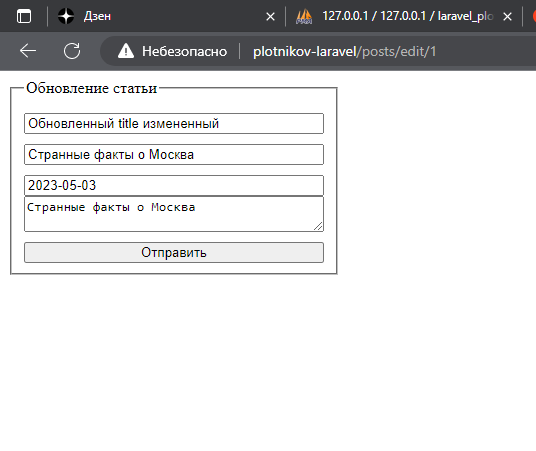
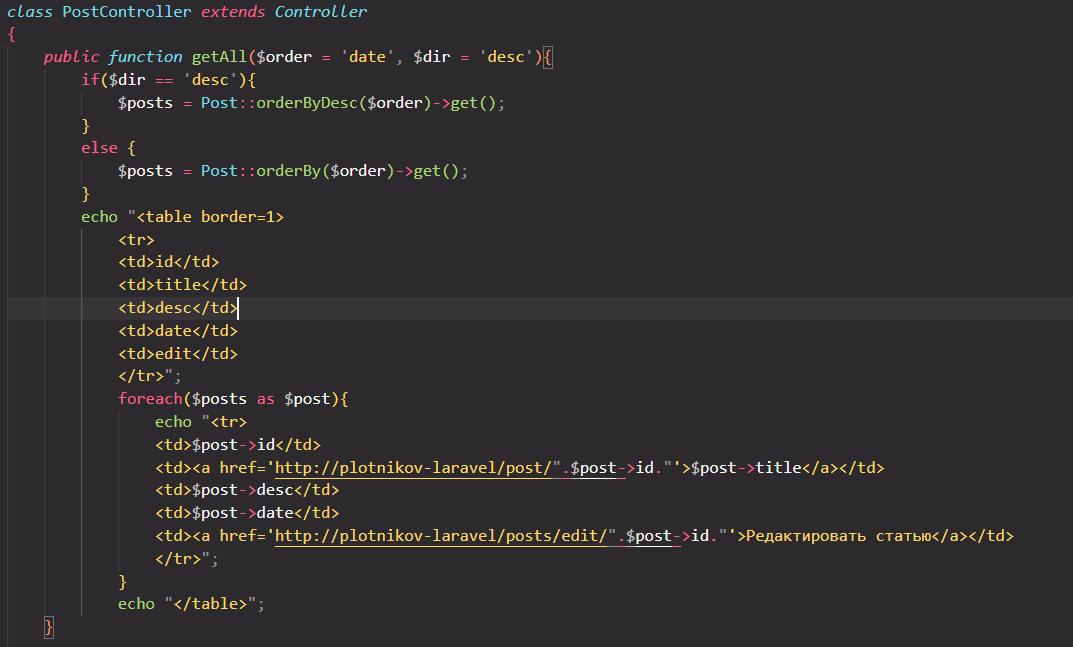




**Задание №16**

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

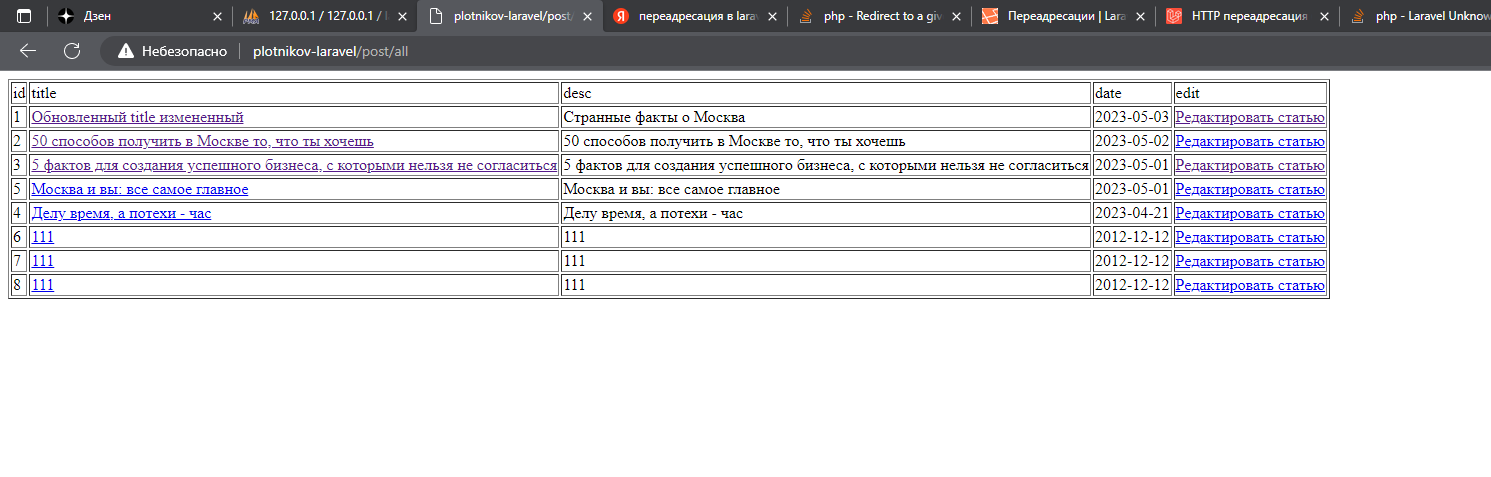


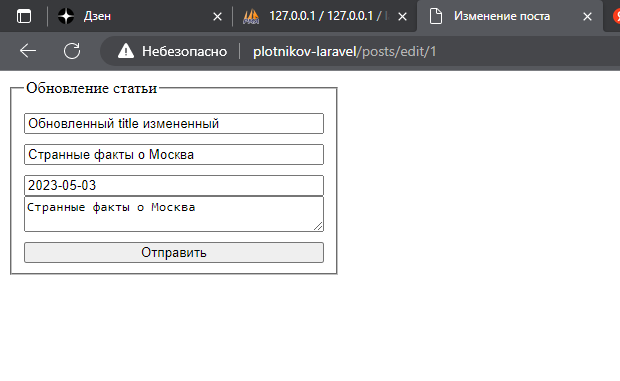
 

**Задание №17**

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

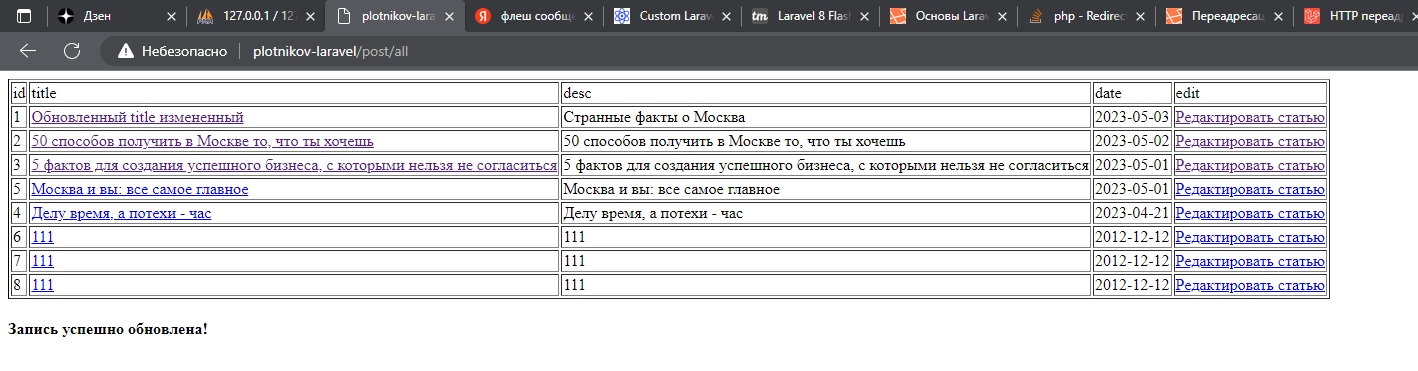


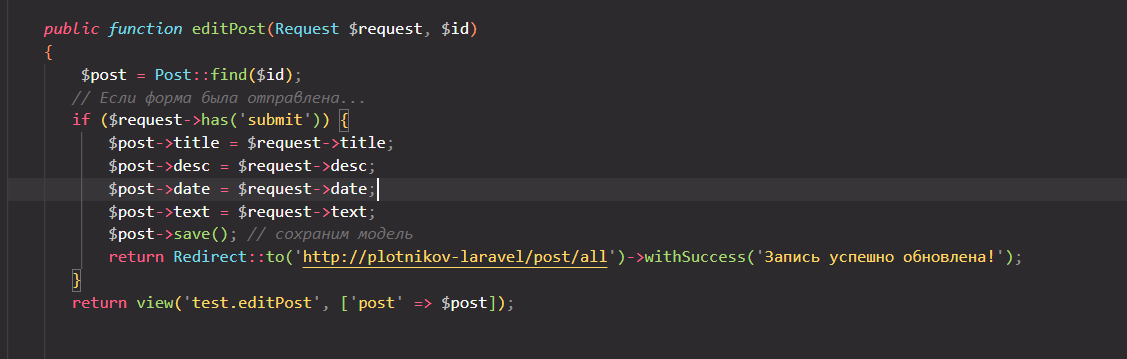




**Задание №18**

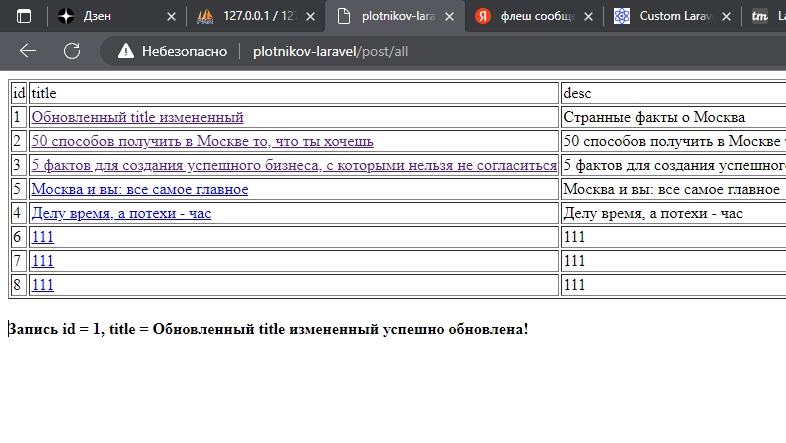
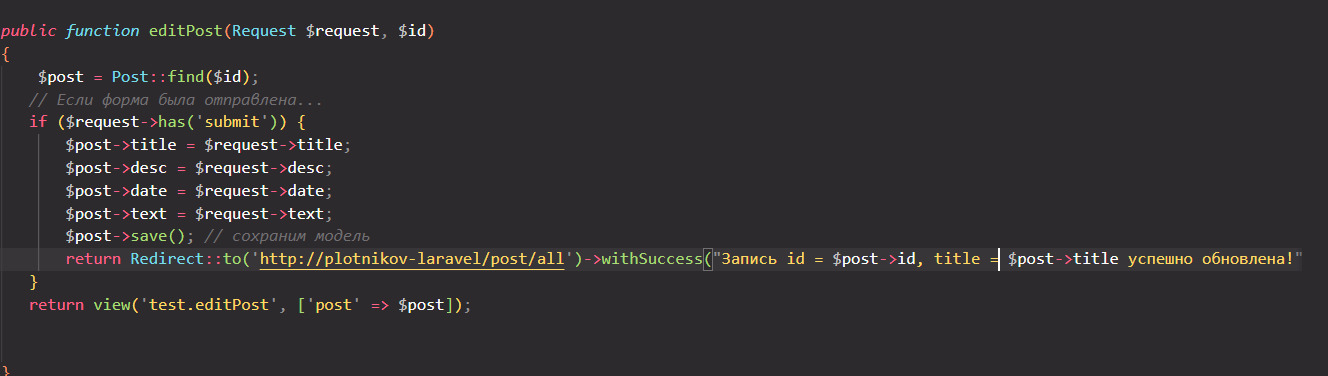
Модифицируйте предыдущую задачу так, чтобы при редиректе отправлялось флеш сообщение об успешном обновлении записи. Выводите это сообщение в представлении действия **getAll**





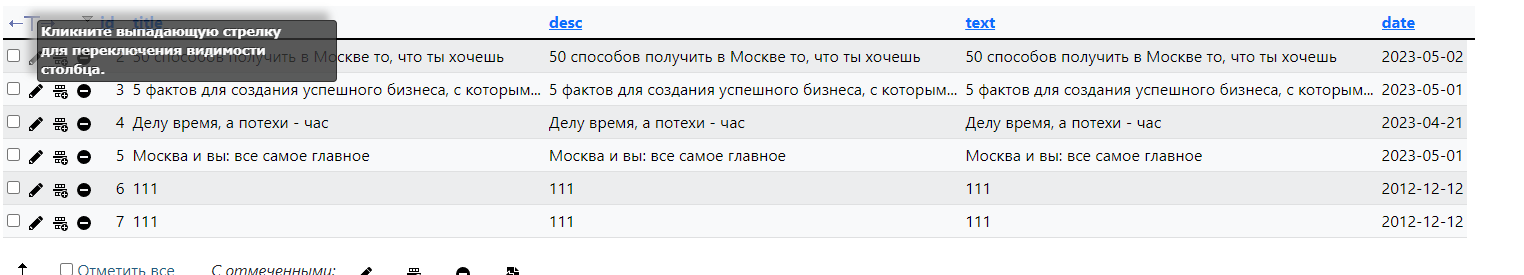
**Задание №19**

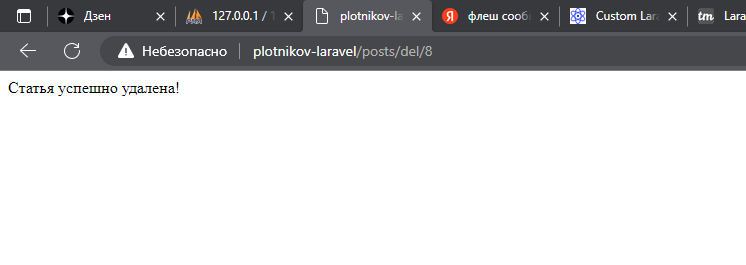
Модифицируйте предыдущую задачу так, чтобы во флеш сообщении был указан id и title статьи, подвергнувшейся изменению.

**Задание №20**

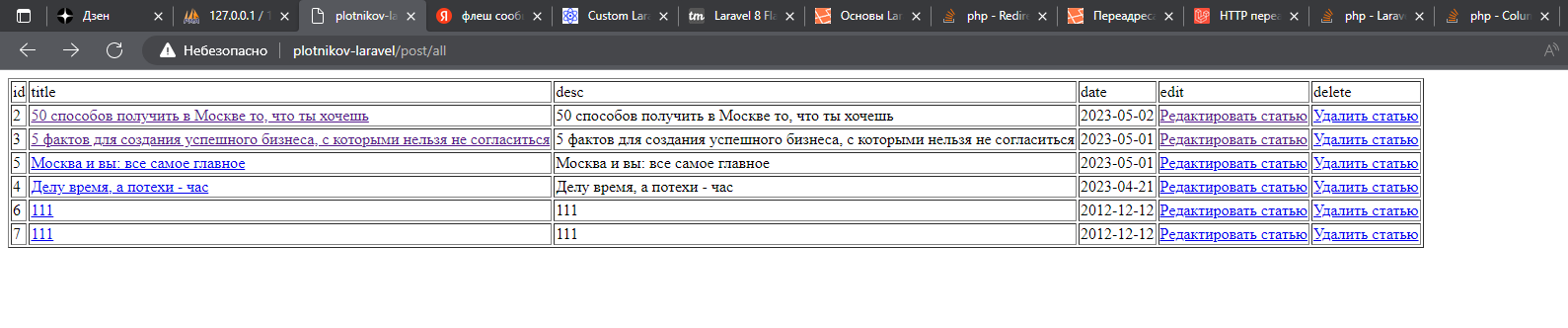
В контроллере **PostController**, созданном в предыдущих уроках, сделайте действие **delPost** для удаления статьи. Пусть это действие будет доступно по адресу **/post/del/:id**, где **:id** будет содержать **id** статьи для удаления.

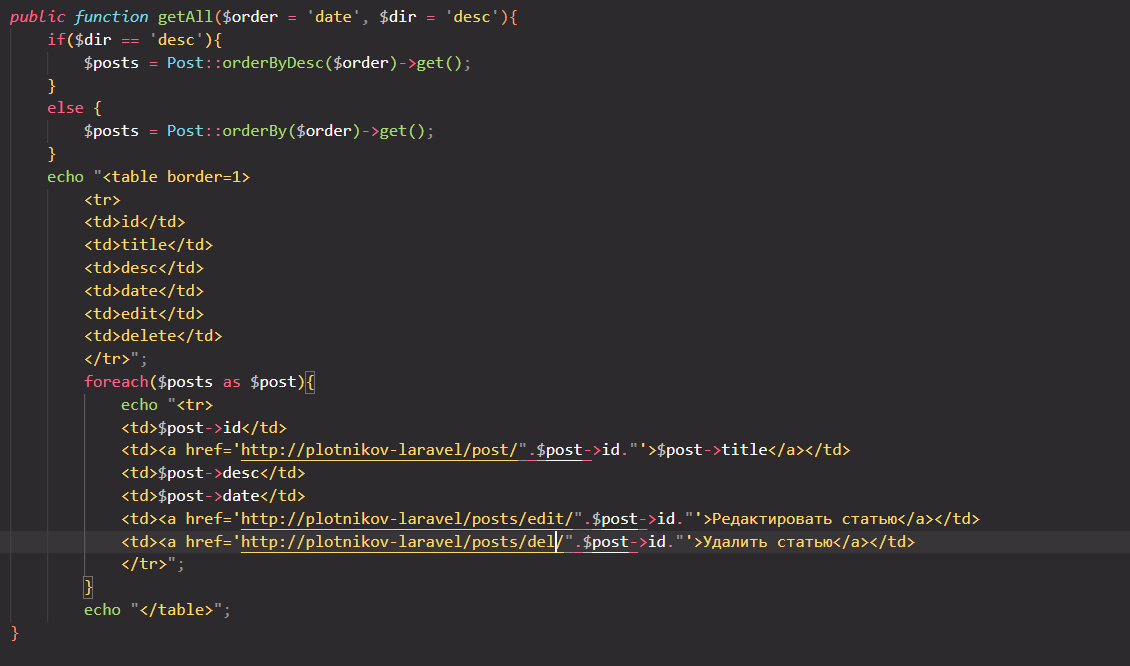




**Задание №21**

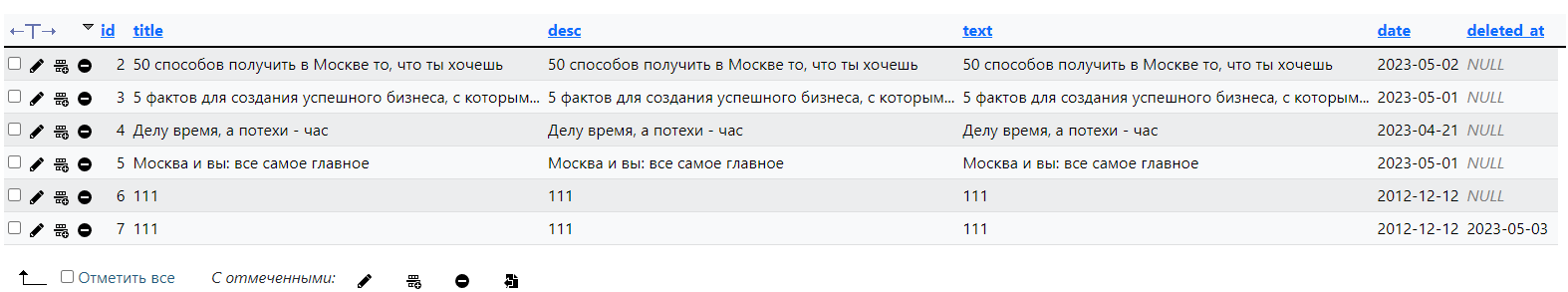
Отредактируйте представление действия **getAll** так, чтобы появилась еще одна колонка со ссылкой на удаления соответствующей статьи.

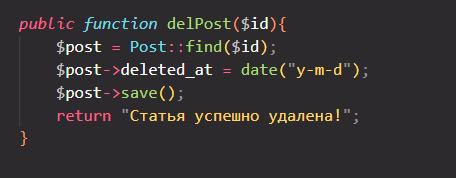




**Задание №22**

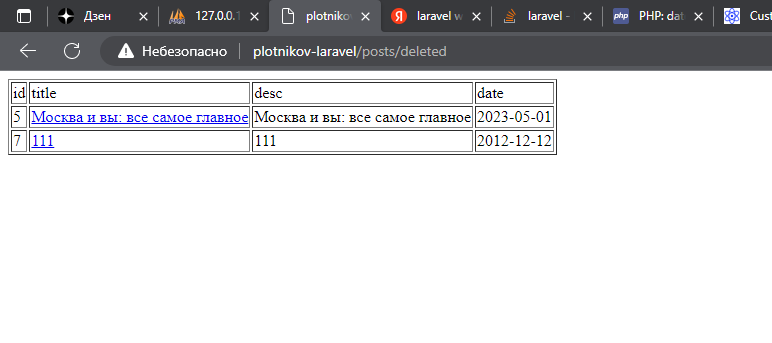
Реализуйте мягкое удаление статей.  
Мягкое удаление означает, что запись на самом деле остаётся в базе данных, но в таблице для записи устанавливается поле deleted\_at.





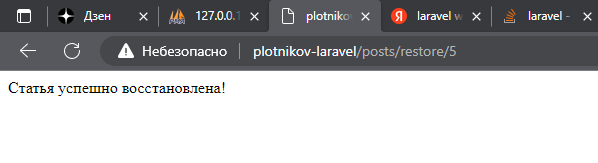
**Задание №23**

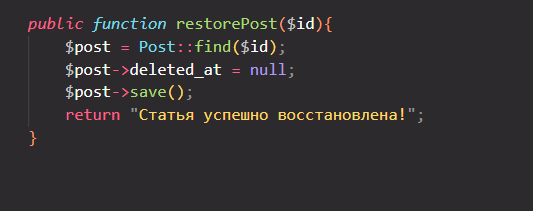
Сделайте действие **getDeletedPost**, выводящее список удаленных статей.

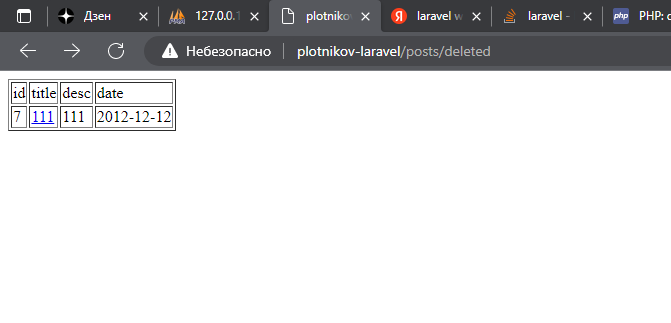
 

**Задание №24**

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







**Задание №25**

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

