

## **Пример разработки модуля дисциплины для системы тестирования на основе Laravel**

*Якимов Антон Сергеевич*

*Приамурский государственный университет имени Шолом-Алейхема*

*Студент*

*Баженов Руслан Иванович*

*Приамурский государственный университет имени Шолом-Алейхема*

*Кандидат педагогических наук, доцент, зав. кафедрой информационных систем, математики и правовой информатики*

### **Аннотация**

В данной статье приведен пример разработки модуля дисциплины на основе Laravel, который будет являться вспомогательным элементом в виде упорядочивания тестов для тестирования участников по оценке знаний.

**Ключевые слова:** laravel, фреймворк, система, модуль, дисциплина, компьютерное тестирование

## **Example of development of the discipline module for the testing system based on Laravel**

*Yakimov Anton Sergeevich*

*Sholom-Aleichem Priamursky State University*

*Student*

*Bazhenov Ruslan Ivanovich*

*Sholom-Aleichem Priamursky State University*

*Candidate of pedagogical sciences, associate professor, Head of the Department of Information systems, Mathematics and Law Informatics*

### **Abstract**

This article is an example of the development of a discipline module based on Laravel, which will be an auxiliary element in the ordering of tests for testing participants in knowledge assessment.

**Keywords:** laravel, framework, system, module, discipline, computer testing

В наше время на этапе развития образования уделяется все больше времени для внедрения эффективных форм контроля знания. На данный момент в информационной сфере наиболее распространенной формой контроля знания является компьютерное тестирование.

Сама разработка компьютерного тестирования пользуется популярностью среди многих людей. А.В. Попов [1] рассматривал

значимость тестирования как инструмента дистанционного контроля знаний, достоинства и недостатки тестирования в высших учебных заведениях. В.С. Лужных и Т.Н. Хацевич [2] разработали системы для тестирования и самотестирования знаний учащихся по различным дисциплинам. В.В. Гладких и С.А. Поздняков [3] описали информационную среду проведения научных исследований при помощи построения и публикации в сети Интернет тестов и опросов. Н.А. Батурин и Н.Н. Мельникова [4] описали различные технологии разработки тестов. С.А. Донских, В.Н. Сёмин и Т.М. Абрамович [5] ставили своей целью проанализировать тестовые задания категории тестов обучения. Ю.М. Артемкина и др. [6] разработали и разместили в сети Интернет банк компьютерных контролирующих тестов по основным разделам общей химии. И.Н. Андреева [7] реализовала интернет-тестирование по дисциплине «Русский язык и культура речи». Л.В. Ларина [8] дала обзор компьютерных систем тестирования знаний студентов, разработанных и используемых в российских вузах.

В данной статье будет рассмотрен принцип разработки модуля дисциплины на основе серверного фреймворка Laravel с помощью языка программирования PHP, где в дальнейшем данный модуль послужит вспомогательной опорой для системы тестирования.

Для того чтобы создать модуль дисциплины на Laravel, нужно подготовить таблицу, модель, контроллер и шаблон.

С помощью файла миграции подготавливаем необходимые поля для базы данных по дисциплине (рис. 1), и делаем миграцию.

```
class CreateDisciplinesTable extends Migration
{
    public function up()
    {
        Schema::create('disciplines', function (Blueprint $table)
        {
            $table->increments('id');
            $table->string('name');
            $table->string('link')->unique();
            $table->text('description')->nullable();
            $table->timestamps();
        });
    }

    public function down()
    {
        Schema::drop('disciplines');
    }
}
```

Рисунок 1 – Миграция дисциплины

Таблица дисциплины содержит в себе такие поля, как *id* (*идентификатор*), *name* (*название*), *link* (*ссылка*), *description* (*описание*) и *timestamp* (*дата создания и обновления записи*).

Далее приступаем к созданию модели. Для этого создаем файл модели по дисциплине, и производим соответствующие настройки (рис. 2).

```
class Discipline extends Model
{
    protected $table = 'disciplines';
    protected $fillable = ['name', 'link', 'description'];

    public function tests() {
        return $this->hasMany('App\Models\Test', 'discipline_id', 'id');
    }
}
```

Рисунок 2 – Модель дисциплины

В данной модели указываем, с помощью переменной *\$table*, к какой таблице будет привязана текущая модель. Под переменной *\$fillable* указываем перечень полей таблицы, которые будут доступны для массового заполнения во время записи данных.

Метод *tests()* – это отношение «один ко многим» между таблицей дисциплины и таблицей тестов.

По завершению создания модели, приступаем к разработке самого контроллера дисциплины. Сам контроллер создаем в виде CRUD-системы для управления и контроля данных над дисциплинами.

Контроллер будет содержать в себе несколько методов. Рассмотрим каждую из них. Первый метод это *index()* (рис. 3).

```
public function index ()
{
    return view('admin.disciplines.index', [
        'disciplines' => Discipline::all()
    ]);
}
```

Рисунок 3 – Контроллер дисциплины, метод *index()*

В нем мы получаем список всех дисциплин и отображаем на экране пользователя с помощью шаблона */admin/disciplines/index.blade.php*. Сам путь шаблона указан в первом параметре метода *view()*.

Далее рассмотрим метод *create()* (рис. 4).

```
public function create ()
{
    return view('admin.disciplines.create');
}
```

Рисунок 4 – Контроллер дисциплины, метод *create()*

Самый простой метод в данном контроллере. В нем выводим шаблон для создания дисциплины.

Следом у нас идет метод *store()* (рис. 5).

```
public function store (Request $request)
{
    $this->validate($request, [
        'name' => 'required|max:255',
        'link' => 'required|alpha_dash|unique:disciplines|max:255',
    ],[],[
        'name' => '"Название дисциплины"',
        'link' => '"URL"'
    ]);

    $create = [
        'name' => $request->name,
        'link' => $request->link
    ];

    if ($request->description)
    {
        $create['description'] = $request->description;
    }

    Discipline::create($create);

    return
    redirect(route('admin.disciplines'))
    ->with('success', 'Дисциплина «'. $request->name .'» успешно добавлена!');
}
```

Рисунок 5 – Контроллер дисциплины, метод *store()*

Данный метод вызывается при POST-запросе во время добавления дисциплины с шаблона *create.blade.php*. Внутри этого метода сначала идет валидация/проверка переданных данных на соответствие по заданным правилам.

Например, поле *name* (*название*) обязателен для заполнения и должен иметь не более 255 символов в длину. Поле *link* (ссылка) имеет идентичные правила, как и у *name*, за исключением еще одной условия – это проверка данного поля на уникальность в таблице *disciplines*, т.к. по нашей задумке, ссылка должна быть всегда уникальной и не должно быть дубликатов.

Далее идет добавление записи в базу данных с помощью метода класса *Discipline::create()*, и вывод результата об успешном добавлении дисциплины.

Также есть метод *edit()* (рис. 6).

```
public function edit($id)
{
    $data['discipline'] = Discipline::findOrFail($id);

    return view('admin.disciplines.edit', $data);
}
```

Рисунок 6 – Контроллер дисциплины, метод edit()

Этот метод получает переданный ему идентификатор дисциплины и проверяет на его наличие в базе данных. Если дисциплина найдена, то отображаем страницу редактирования дисциплины, в противном же случае, отобразится страница ошибки 404.

Есть еще метод *update()* (рис. 7).

```
public function update(Request $request, $id)
{
    $discipline = Discipline::findOrFail($id);

    $this->validate($request, [
        'name' => 'required|max:255',
        'link' => 'required|alpha_dash|unique:disciplines,link,.' . $id .' |max:255',
    ], [], [
        'name' => '"Название дисциплины"',
        'link' => '"URL"'
    ]);

    $update = [
        'name' => $request->name,
        'link' => $request->link
    ];

    if ($request->description)
    {
        $update['description'] = $request->description;
    }

    $discipline->update($update);

    return
    redirect(route('admin.disciplines'))
    ->with('success', 'Дисциплина «'. $request->name .'» успешно обновлена!');
}
```

Рисунок 7 – Контроллер дисциплины, метод update()

Метод *update()* вызывается при POST-запросе со страницы редактирования дисциплины, для того чтобы обновить какие-либо измененные данные. Структура *update()* метода идентична с предыдущим *store()* методом. Только в данном случае, идет не создание записи дисциплины в базу данных, а обновление существующей записи.

Также у нас остался еще последний метод, это *delete()*.

```
public function delete($id)
{
    $discipline = Discipline::findOrFail($id);
    $discipline->delete();

    return
    redirect(route('admin.disciplines'))
    ->with('warning', 'Дисциплина «'. $discipline->name .'» удалена!');
}
```

Рисунок 8 – Контроллер дисциплины, метод delete()

Данный метод позволяет удалять указанную дисциплину из базы данных, и выводить уведомление о совершенном действии.

Теперь создание контроллера закончено. Далее необходимо добавить маршрутизацию для данного контроллера. Для этого в файле роута перечисляем следующие маршруты (рис. 9), где DisciplinesController – это контроллер дисциплины.

```
// Дисциплины
Route::prefix('disciplines')->name('disciplines')->group(function()
{
    Route::get('/')
        ->uses('DisciplinesController@index');

    Route::get('/create')
        ->name('.create')
        ->uses('DisciplinesController@create');

    Route::post('/store')
        ->name('.store')
        ->uses('DisciplinesController@store');

    Route::get('/edit/{id}')
        ->name('.edit')
        ->uses('DisciplinesController@edit');

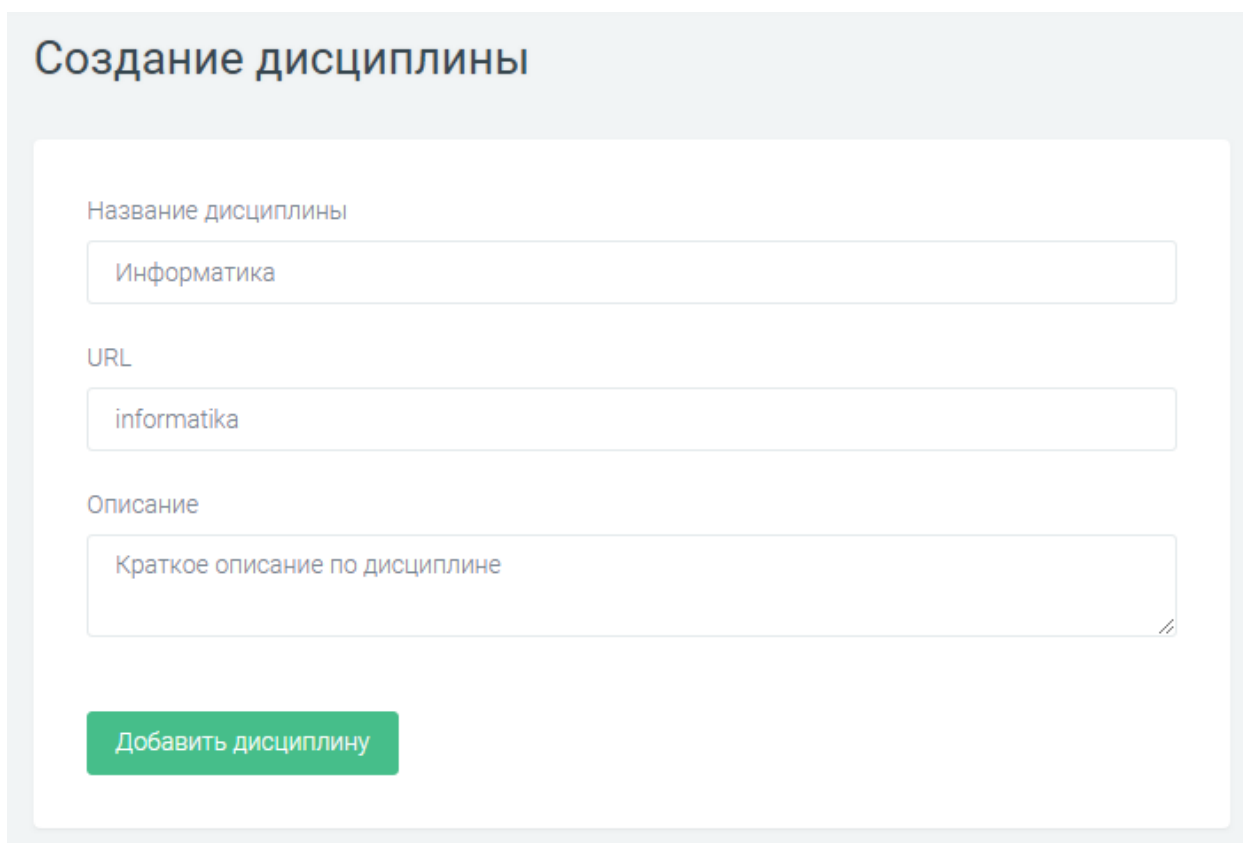
    Route::post('/update/{id}')
        ->name('.update')
        ->uses('DisciplinesController@update');

    Route::get('/delete/{id}')
        ->name('.delete')
        ->uses('DisciplinesController@delete');
});
```

Рисунок 9 – Маршрутизация дисциплины

На этом разработка внутреннего модуля дисциплины закончена. Давайте ознакомимся с получившимся результатом разработанного модуля.

Так выглядит шаблон добавления дисциплины после вызова метода *create()* с помощью маршрутизации по ссылке */disciplines/create* (рис. 10).



### Создание дисциплины

Название дисциплины

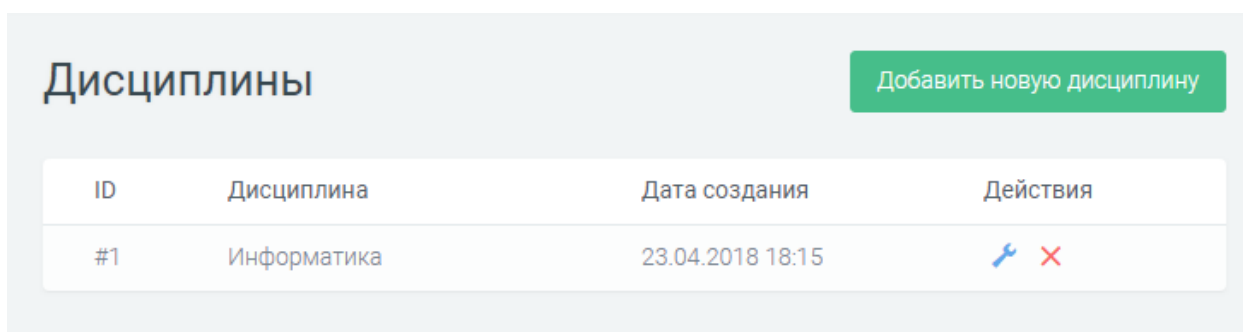
URL

Описание

Добавить дисциплину

Рисунок 10 – Создание дисциплины

После добавления дисциплины, у нас происходит редирект на страницу списка дисциплин (рис. 11).



### Дисциплины

Добавить новую дисциплину



ID	Дисциплина	Дата создания	Действия
#1	Информатика	23.04.2018 18:15	 

Рисунок 11 – Список дисциплин

При клике на синюю иконку, мы переходим на страницу редактирования дисциплины (рис. 12).

## Редактирование дисциплины «Информатика»

Название дисциплины

URL

Описание

Обновить дисциплину

Рисунок 12 – Редактирование дисциплины

Если изменить данные и нажать на кнопку обновления дисциплины, то при успешном обновлении данных, мы увидим следующее уведомление (рис. 13).

## Дисциплины

Добавить новую дисциплину

Дисциплина «Информатика» успешно обновлена!



ID	Дисциплина	Дата создания	Действия
#1	Информатика	23.04.2018 18:15	 

Рисунок 13 – Успешное обновление дисциплины

Если нажать на красную кнопку у выбранной дисциплины, то произойдет удаление данной записи и выдаст соответствующее уведомление (рис. 14).



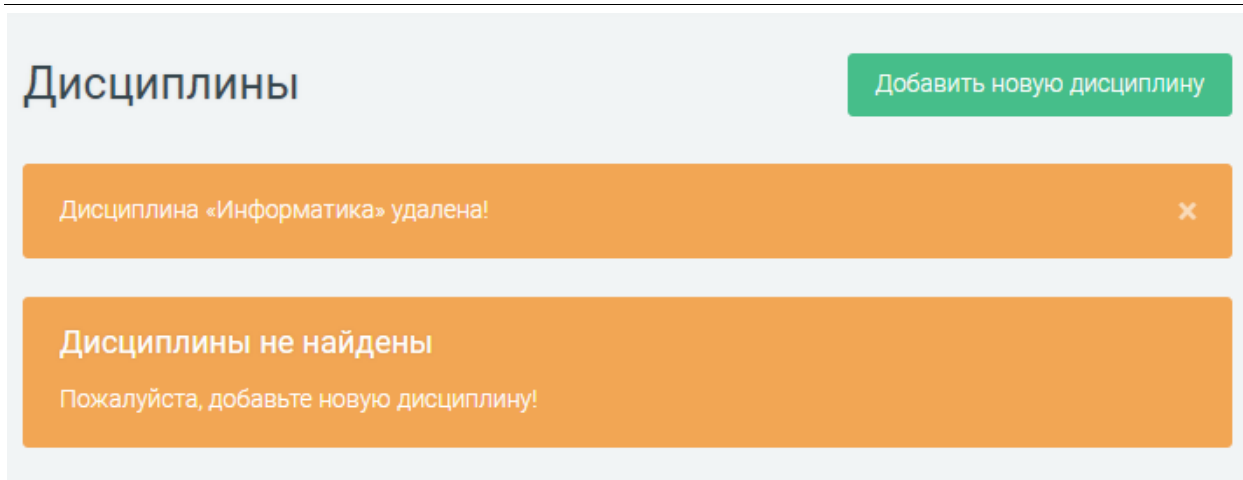


Рисунок 14 – Удаление дисциплины

В данной статье был приведен пример разработки дисциплины на основе Laravel, где не только познакомились с самим процессом разработки, но и продемонстрировали возможность быстрой и доступной разработки модулей разного уровня. В дальнейшем планируется реализация модуля тестов с вложенными вопросами и ответами для создания полноценной системы тестирования по оценке знаний учащихся.

### Библиографический список

1. Попов А.В. Тестирование как метод контроля качества знаний студентов // ТРУДЫ СПБГИК. 2013. №. С.283-286.
2. Лужных В.С., Хацевич Т.Н. Компьютерные системы тестирования // Интерэкспо Гео-Сибирь. 2006. №. С.183-185.
3. Гладких В.В., Поздняков С.А. Онлайн-конструктор тестов и опросов как инфраструктура проведения научных исследований // Концепт. 2016. №S3. С.31-35.
4. Батулин Н.А., Мельникова Н.Н. Технология разработки тестов: часть VI // Вестник ЮУрГУ. Серия: Психология. 2011. №18 (235). С.48-59.
5. Донских С.А., Семин В.Н., Абрамович Т.М. Разработка компьютерного теста по материаловедению // Вестник Таганрогского института имени А.П. Чехова. 2010. №1. С.183-190.
6. Артемкина Ю.М., Загоскин Ю.Д., Кузнецов Н.М., Паркина М.П., Щербаков В.В. Банк компьютерных контролирующих тестов по общей химии // Успехи в химии и химической технологии. 2014. №9 (158). С.92-94.
7. Андреева И.Н. Интернет-тестирование по дисциплине «Русский язык и культура речи» // Вестник ЧГУ. 2006. №7. С.101-111.
8. Ларина Л.В. Компьютерные системы тестирования знаний студентов на различных этапах оценки успеваемости // ОНВ. 2013. №1 (117). С.43-46.