УДК 004

Пример реализация API на фреймворке Laravel

Стрельцова Марина Николаевна
Приамурский государственный университет им. Шолом-Алейхема
Студент

Аннотация
На сегодняшний день существует множество различных сервисов и сайтов, предоставляющих различную информацию клиентам: данные погоды, состояние биржи валюты, цены на товары в каком-либо онлайн-магазине и многое другое. Разработчики используют такое понятие, как API, которое представляет собой описание способов взаимодействия между программами. В данной статье описан пример реализации собственного API на фреймворке Laravel. Описаны методы GET и POST, а также проведено тестирование сайта на передачу запрашиваемых данных с помощью Postman в формате JSON.
Ключевые слова: API, Laravel, JSON, программный интерфейс, Postman

Features of the Adeptly service for creating interactive simulators

Streltsova Marina Nikolaevna
Sholom-Aleichem Priamursky State University
Student

Abstract
To date, there are many different services and sites that provide various information to customers: weather data, the state of the currency exchange, prices for goods in an online store, and much more. Developers use such a concept as API, which is a description of the ways of interaction between programs. This article describes an example of implementing your own API on the Laravel framework. The GET and POST methods are described, and the site is tested for transmitting the requested data using Postman in JSON format.
Keywords: API, Laravel, JSON, programming interface, Postman

1. Введение

1.1 Актуальность исследования

На сегодняшний день существует множество различных сервисов и сайтов, предоставляющих различную информацию клиентам: данные погоды, состояние биржи валюты, цены на товары в каком-либо онлайн-магазине и многое другое. Разработчики используют такое понятие, как API, которое представляет собой описание способов взаимодействия между программами. API играет роль посредника между внутренними и внешними программными функциями, обеспечивая настолько эффективный обмен

информацией, что конечные пользователи обычно его просто не замечают. Именно поэтому API является очень популярным инструментов для обмена информацией и интеграцией различных продуктов.

В данной статье описан пример реализации собственного API на фреймворке Laravel. Описаны методы GET и POST, а также проведено тестирование сайта на передачу запрашиваемых данных с помощью Postman в формате JSON.

1.2 Обзор исследований

Разработка информационных систем для интернет-магазина с веб-интерфейсом с использованием фреймворка Laravel и серверного API RajaOngkir описывается в научном исследовании P.S. Prawito и R Rahadi [1]. В статье Т.Л. Тен и соавторов представлена модель разработки Restful API, основанная на языке PHP и структуре Laravel. Рассматриваются основные технические проблемы, которые необходимо решить при построении Restful API, и приведены детали реализации на основе Laravel [2]. А. Ю. Манвелян и Т.А. Галаган в своей научной работе рассматривают варианты разделения данных на Vuex модули при разработке пользовательских интерфейсов с помощью клиентского фреймворка Vue.js с применением в качестве основы для по построения API серверного фреймворка Laravel [3]. Анализ проблем и перспектив развития программных интерфейсов (API) и открытых банковских платформ с целью обеспечения эффективности финансовых результатов, а также финансовой безопасности рассмотрели в своей статье Н. Н. Новосёлова, В. В. Хубулова и З.М. Яковенко [4]. Целью статьи А.Ю. Хотько и А. С. Романеня является создание открытого API, с помощью которого можно автоматизировать процесс персонализации карты, а также интегрировать различные сервисы, тем самым ускоряя и упрощая процесс создания сервисов с использованием карт [5].

1.3 Цель исследования

Целью данного исследования является написание и тестирование собственного API на фреймворке Laravel.

2. Методы исследования

В рамках данной статьи будет использоваться готовый проект на Laravel с тестовыми данными. Будут написаны методы GET и POST для передачи информации по API и тестирования с помощью Postman в формате JSON. Рассмотрены действия index, show и store контроллера для отображения всех записей таблицы базы данных, отображения конкретной записи по id и добавления записи соответственно. Поддерживаемые действия и пути для доступа к ним описаны в официальной документации к Laravel (кроме действий create и edit) (Рис. 1).

Метод	URI	Действие	Имя маршрута
GET	/photos	index	photos.index
GET	/photos/create	create	photos.create
POST	/photos	store	photos.store
GET	/photos/{photo}	show	photos.show
GET	/photos/{photo}/edit	edit	photos.edit
PUT/PATCH	/photos/{photo}	update	photos.update
DELETE	/photos/{photo}	destroy	photos.destroy

Рисунок 1 – Поддерживаемые действия

3. Результаты исследования

Для начала перейдем в папку проекта на Laravel и создадим API контроллер следующей командной (Рис. 2-3).

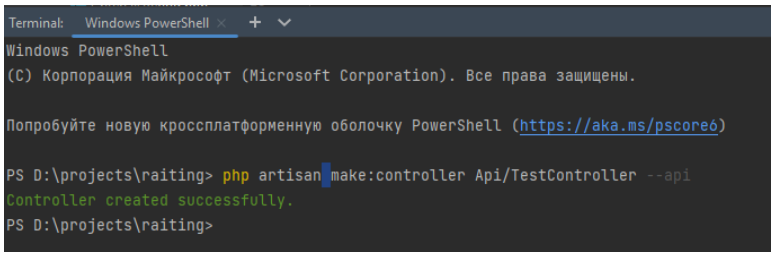

Рисунок 2 – Создание API контроллера

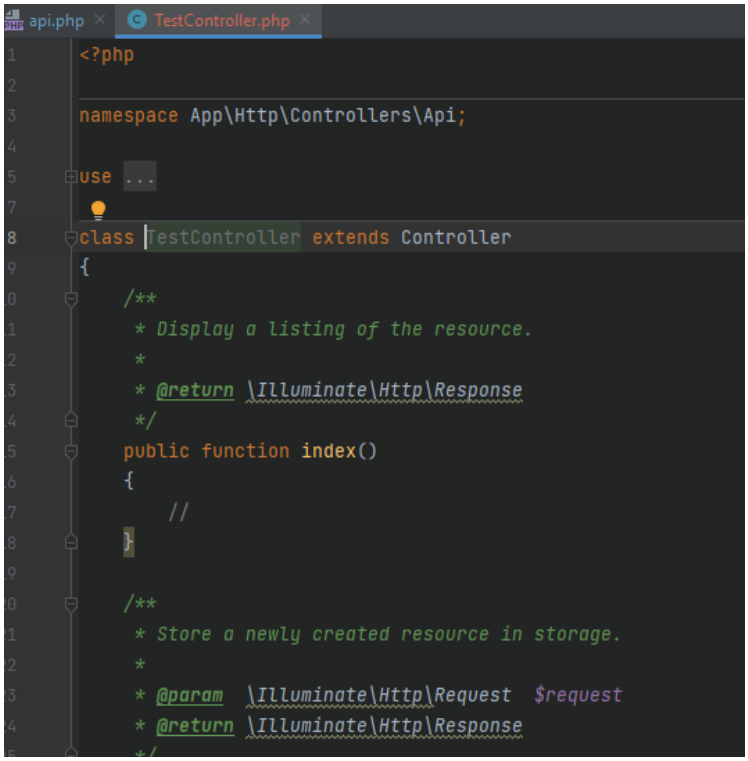

Рисунок 3 – Код внутри API контроллера

Тестовая база данных выглядит следующим образом (Рис. 4).

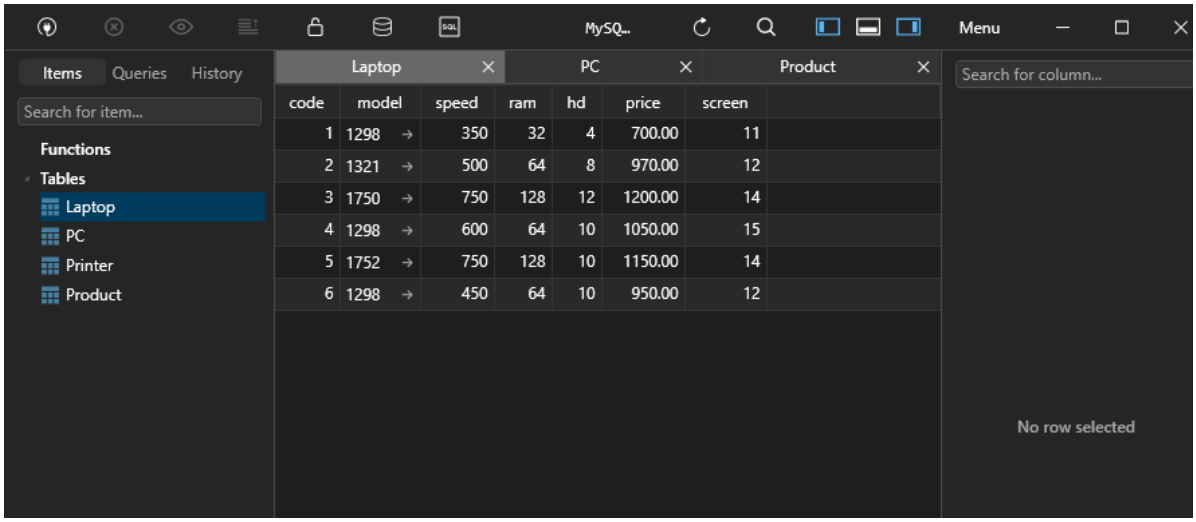

Рисунок 4 – Тестовая база данных

Далее создадим 4 модели для взаимодействия фреймворка с тестовой базой данных: Laptop, PC, Printer и Product в папке Models. Все модели имеют идентичную структуру (Рис. 5).

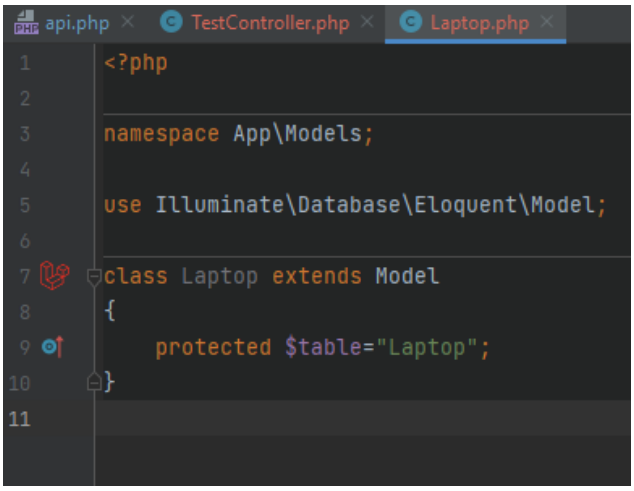

Рисунок 5 – Пример созданной модели для таблицы Laptop

Теперь напишем API маршрут для действия index к таблице Laptop (Рис.6).

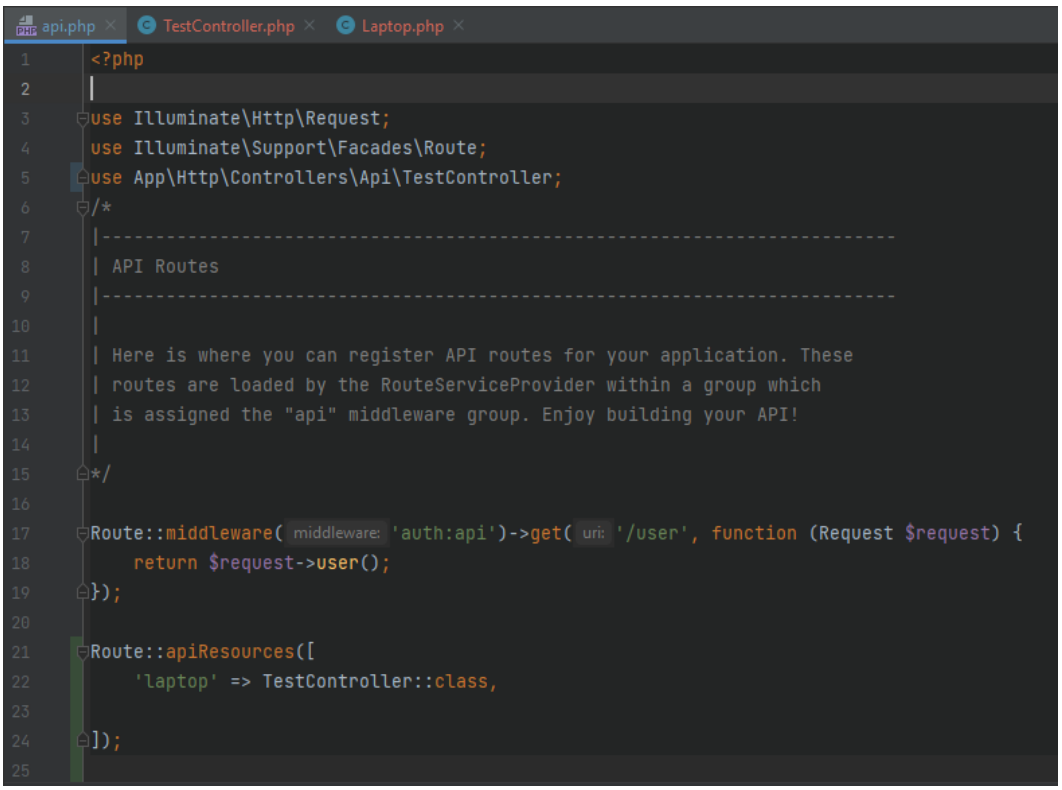

Рисунок 6 – Маршрут для доступа к таблице Laptop

Следующим шагом перейдём в контроллер и в действии index напишем код для возращения всех значений таблицы Laptop (Рис. 7).

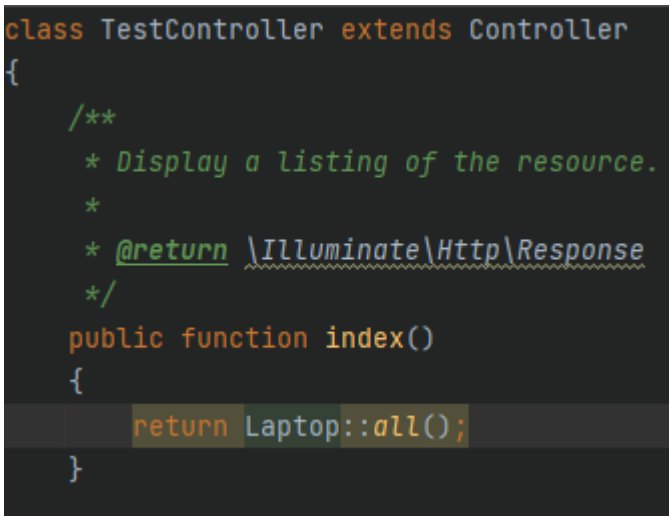

Рисунок 7 – Действие index контроллера TestController

Теперь можно протестировать работу API, перейдя в Postman и написав запрос к сайту (Рис. 8).

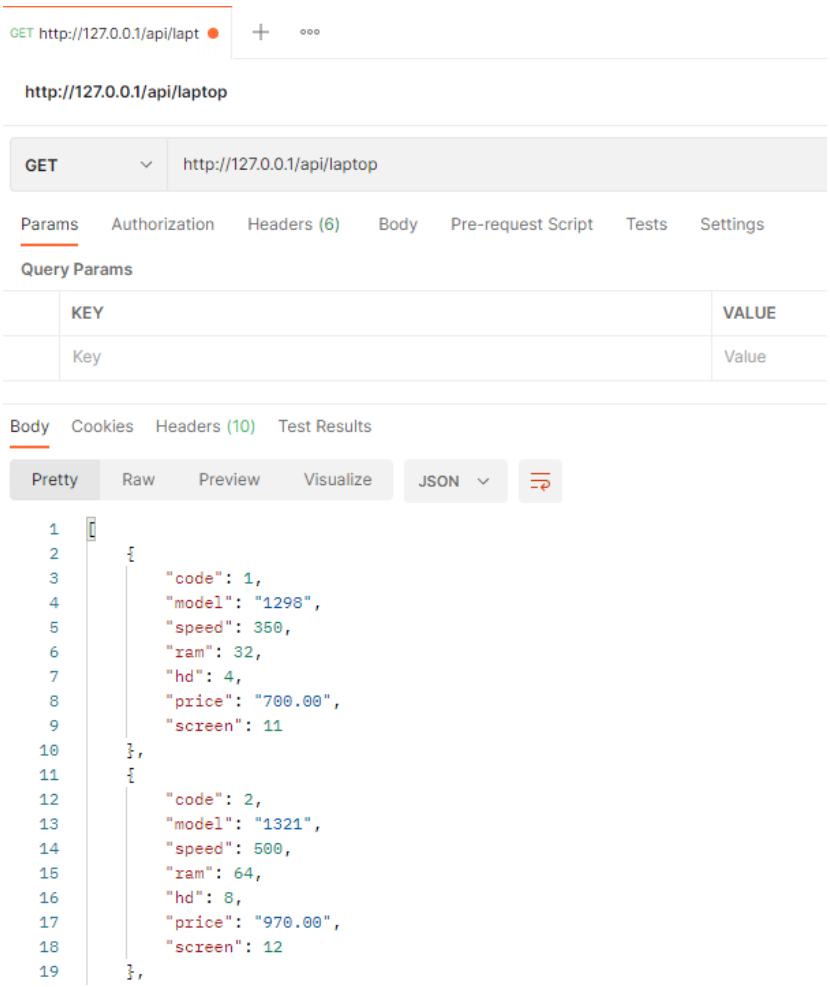

Рисунок 8 – Результат запроса к сайту

На скриншоте 8 видно, что сайт отдает все записи из таблицы Laptop в формате JSON, который Postman преобразует в «читаемый» для пользователя вид. Данные в таком формате можно парсить, вытаскивая нужную информацию и передавая их для дальнейшей обработки.

Аналогично опишем другие действия: show и store. Создадим модель Printer и добавим в действие show необходимый код (Рис.9-10).

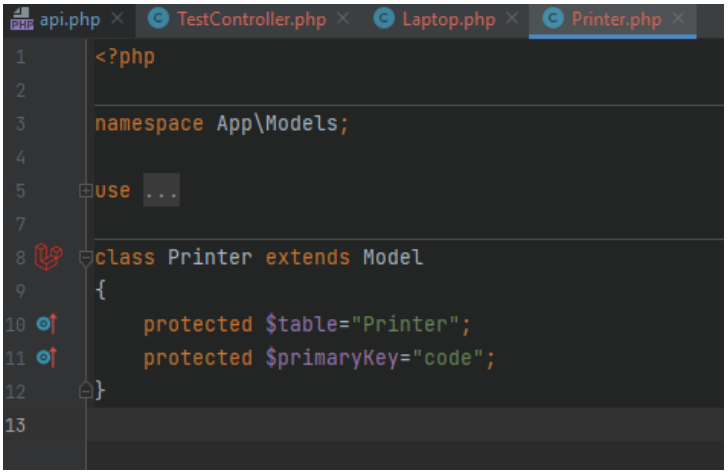

Рисунок 9 – Модель Printer

```php
public function show($id)
{
    return Printer::find($id);
}
```

Рисунок 10 – Действие show контроллера TestController

Результат действий можно проверить перейдя в Postman (Рис. 11).

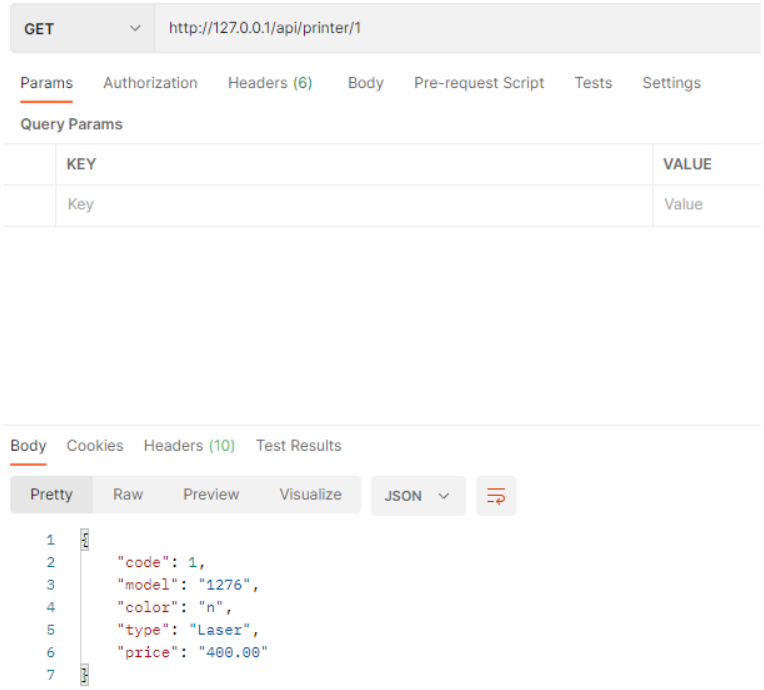

Рисунок 11 – Результат запроса к модели Printer

Далее опишем действие store для добавления записи в таблицу Laptop. В модель таблицы добавим переменную $fillable для защиты передаваемых полей (Рис.12).

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;

class Laptop extends Model
{
    protected $table="Laptop";
    protected $fillable=['code','model'];
    public $timestamps = false;
}
```

Рисунок 12 – Обновление модели Laptop

Теперь необходимо создать Request для описания правил валидации передаваемых полей (Рис.13-14).

```
PS D:\projects\raiting> php artisan make:request LaptopStoreRequest
Request created successfully.
```

Рисунок 13 – Создание Request

```php
<?php

namespace App\Http\Requests;

use Illuminate\Foundation\Http\FormRequest;

class LaptopStoreRequest extends FormRequest
{
    /**
     * Determine if the user is authorized to make this request.
     *
     * @return bool
     */
    public function authorize()
    {
        return true;
    }

    /**
     * Get the validation rules that apply to the request.
     *
     * @return array
     */
    public function rules()
    {
        return [
            'code'=>'required',
            'model'=>'required'
        ];
    }
}
```

Рисунок 14 – Код Request

Далее опишем действие store в контроллере для создания записи и ее отображения (Рис.15).

```php
    public function store(LaptopStoreRequest $request)
    {
        return Laptop::create($request->validated());

    }
```

Рисунок 15 – Код Request

Протестируем написанный запрос в Postman (Рис.16-17).

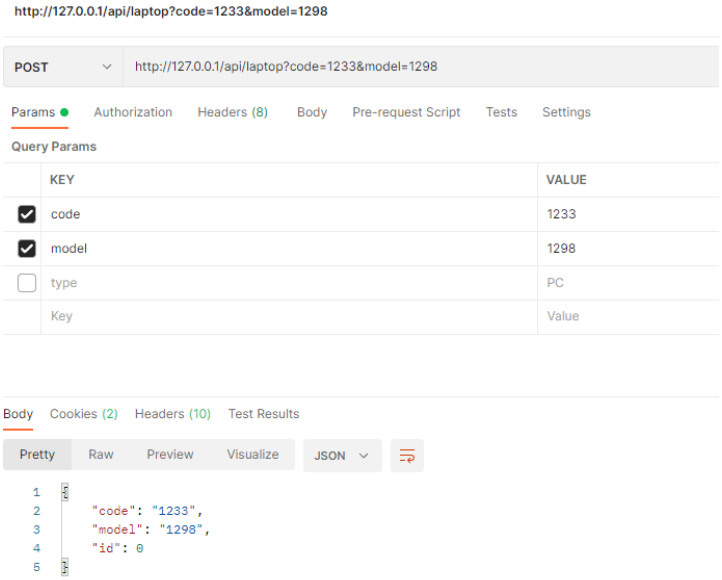

Рисунок 16 – Тестирование запроса действия show

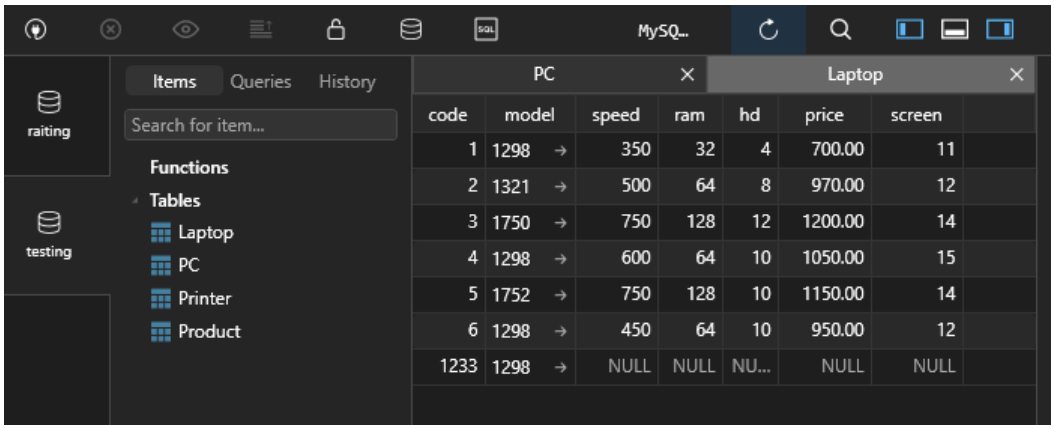

Рисунок 17 – Добавленная запись в базе данных

4. Выводы

В рамках данной статьи был описан пример реализации собственного API на фреймворке Laravel. Написаны методы GET и POST для передачи информации по API и тестирования с помощью Postman в формате JSON. Рассмотрены действия index, show и store контроллера для отображения всех записей таблицы базы данных, отображения конкретной записи по id и добавления записи соответственно.

API является универсальным «языком общения» программ между собой, который позволяет разработчикам эффективно интегрировать сервисы между собой и делать разработку более безопасной, выводя ряд функций в отдельное приложение, где они будут скрыты.

Библиографический список

1. Prawito P. S., Rahadi R. Perancangan Sistem Informasi Toko Online Berbasis Web dengan Menggunakan Laravel dan Api Rajaongkir // Syntax Literate; Jurnal Ilmiah Indonesia. 2020. Т. 5. №. 12. С. 1657-1668.

2. Тен Т. Л., Шинекенев Н. А., Когай Г. Д. Построение REST API на основе веб-фреймворка Laravel // Перспективы развития строительного комплекса. 2018. №. 12. С. 369-372.
3. Манвелян А. Ю., Галаган Т. А. Применение хранилища состояния Vuex при разработке пользовательских интерфейсов с помощью Vue. JS на основе фреймворка Laravel // Современные инновации в науке и технике. 2021. С. 166-169.
4. Новосёлова Н. Н., Хубулова В. В., Яковенко З. М. Разработка открытых платформ на основе программных интерфейсов (API) с целью обеспечения финансовой эффективности и безопасности банков и перехода к открытым бизнес-моделям // Вестник Института дружбы народов Кавказа (Теория экономики и управления народным хозяйством). Экономические науки. 2020. №. 4.
5. Хотько А. Ю., Романеня А. С. MapDial–программный продукт на основе карт с открытым программным интерфейсом // Первый шаг в науку : тезисы докладов 72-й научно-технической конференции учащихся, студентов и магистрантов, Минск, 21-29 апреля 2021 г. Минск: БГТУ, 2021. С. 853-854.
6. Laravel URL: https://laravel.com/ (дата обращения: 15.06.2022).