DOI: 10.12731/2227-930X-2018-4-137-152
УДК 004

СОВРЕМЕННЫЙ WEB НА ПРИМЕРЕ СОЗДАНИЯ ПРОСТОГО МЕНЕДЖЕРА ЗАДАЧ

Хамидуллин М.Р., Бадыков И.В.

В статье рассматривается процесс создания веб приложения «Менеджер задач». В разработке используются современные веб инструменты. Непосредственно, в самой статье ведется полный разбор написания простейшего менеджера задач, с использованием полного перечня современных технологий. Которые, включают в себя технологии клиентской и серверной стороны. В серверной части будет использоваться полный стек серверного языка, в нашем случае php, и работы с базой данных mysql, используя популярный фреймворк – Laravel. В свою очередь клиентская сторона, будет включать в себя асинхронную работу (без перезагрузки страницы) с серверной стороной, используя джаваскрипт фреймворк – VueJs. Полный стек затрагиваемых технологий будет следующим: Клиентская часть – HTML, CSS, JavaScript, VueJs, также серверная часть – php, Mysql, Laravel. Также, будет краткое знакомство с менеджером зависимостей php-comporser. Который имеет возможность скачки пустого первичных файлов для запуска проекта.

***Цель** – создать современное веб приложение.*

***Метод или методология проведения работы:** в статье использовались информационно-технические методы, в момент создания приложения.*

***Результаты:** получены наиболее информативные способы, создания веб приложения, основанные на полном асинхронном подходе.*

***Область применения результатов:** полученные результаты целесообразно применять в созданиях любых веб проектов.*

***Ключевые слова:** менеджер задач; программное обеспечение; веб; сайты; современный веб.*

MODERN WEB ON THE EXAMPLE OF CREATING A SIMPLE TASK MANAGER

Khamidullin M.R., Badykov I.V.

The article describes the process of creating a web application "Task Manager". The development uses modern web tools. Directly, in the article itself, a complete analysis of the writing of the simplest task manager is carried out using the full list of modern technologies. Which include client and server side technologies. The server side will use the full server language stack, in our case php, and work with the mysql database using the popular Laravel framework. In turn, the client side will include asynchronous work (without reloading the page) with the server side, using the javascript framework – VueJs. The full stack of technologies involved will be as follows: The client side – HTML, CSS, JavaScript, VueJs, and the server side – php, Mysql, Laravel. Also, there will be a brief acquaintance with the dependency manager php-composer. Which has the ability to download empty primary files to run a project.

Purpose. *Create a modern web application.*

Methodology. *The article used information technology methods at the time of creating the application.*

Results. *The most informative methods for creating web applications based on a complete asynchronous approach are obtained.*

Practical implications. *The results obtained should be used in the creation of any web projects.*

Keywords: *task manager; software; web; sites; modern web.*

Каждый ребенок, не говоря уже о взрослом, знает, что такое веб-сайты, и что на нем он может получить. Дети интересуются сайтами в первую для поисков мультиков и игр, а взрослые в поисках услуг, информации и любимых сериалов.

Веб-сайты – это наипростейшая возможность, связать множество людей с интересующих их информацией, не выходя из дома. Почти каждая компания, организация или обычные индивидуаль-

ные предприниматели имеют свою сайт-визитку, не говоря о различных интернет проектах тех или иных услуг, которые рвутся в вершину самых посещаемых и популярных сайтов, в поисках награды [1, с. 16].

Мало кто знает, но каждым годом веб становится все тяжелее и тяжелее, в меру развития технологий, знать только языки программирования уже мало, нужно знать различные библиотеки, используемые при создании сайтов, которые в свою очередь часто обновляются, становясь все лучше, быстрее и без сомнений, для новичков понимания – тяжелее.

Инструментарий для создания современного веб сайта

Рядовые пользователи, заходя на тот или иной сайт, даже могут не задумываться как он на самом деле работает, какие механизмы он использует, на какие составляющие он делится и какое количество человек разрабатывало данный сайт. Веб приложения делятся на две составляющие, это Front-End (передний план или по-другому – пользовательская часть) и Back-End (задний план, серверная часть). На рис. 1 изображены области веб сайта с включающими в себя языков программирования [12, с. 16].

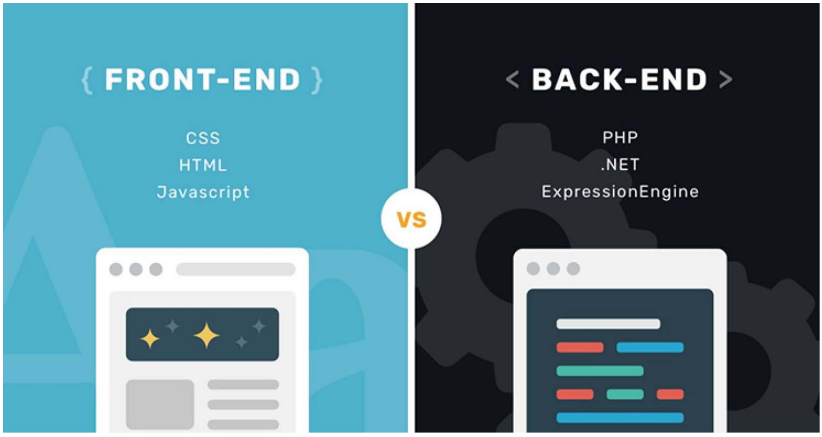

Рис. 1. Краткий список областей веб приложения с включающими в себя языками

Пользовательский интерфейс, или как известно мастерам по созданиям сайтов – Front-End, это передний план сайта, то, что видят посетители сайта. То есть, это HTML, CSS и JavaScript.

HTML (Hyper Text Markup Language – «Язык гипертекстовой разметки») является фундаментальной частью почти каждого веб-сайта, разработанный в 1986 году. Изобретателем которого является британский ученый Тимоти Джон Бернерс-Ли. Это скелет сайта, куда нанизываются, css, js, php и другие технологии и языки [2, с. 16].

CSS (Cascading Style Sheets) – каскадные таблицы стилей. Преимущественно используется как средство описания, оформления внешнего вида веб-страниц. Именно благодаря CSS мы имеем изящно красиво оформленные веб-сайты.

Фреймворки CSS, или также известен как библиотеки CSS, это заранее заготовленный набор стилей, которые упрощают разработку веб приложения. Существует различные библиотеки стилей, которые по функционалу очень похожи друг на друга, в основном их отличает только сам дизайн (дизайн форм, кнопочек, шрифтов и прочее).

JavaScript – прототипно-ориентированный сценарный язык программирования, используется как встраиваемый язык для программного доступа к объектам приложений. Наиболее широкое применение находит в браузерах как язык сценариев для придания интерактивности веб-страницам.

VueJs – это фреймворк JavaScript библиотеки, который является полностью самостоятельным объектом. Предназначенный для создания пользовательских интерфейсов веб приложения. Его отличительная черта в сравнении с другими фреймворков в том, что он пригоден для постепенного внедрения в веб-приложение [14, с. 16].

Back-End это серверная составляющая веб приложения. Где происходит полная работа серверного языка с базами данных. Серверная часть является мозгом во всей этой связке, если клиентская часть отвечает только за вывод, то сервер это планировщик и управленец всей работы.

PHP (PHP: Hypertext Preprocessor – «PHP: процессор гипертекста») – является наипопулярнейшим языком программирования общего назначения, с открытым исходным кодом. Изначально создавался языком программирования для работы с Web-сайтами, его код может легко внедряться в HTML код.

Язык программирования PHP – активно используется для построения динамических web-сайтов, используется для создания внутренней части сайта На PHP написана львиная доля всех сайтов мира, сравнивая с другими языками программирования, а именно около 80%. Также, популярнейшая CMS (Content Management System) WordPress, написанная на PHP, занимает около 25% всех сайтов мира.

MySql – свободная реляционная система управления базами данных. Она является решением для малых и средних веб-приложений. Обычно MySQL используется в качестве хранения различного рода информации в базе данных, к которому обращаются пользователь или клиент для получения запрашиваемой информации.

Фреймворк PHP Laravel – фреймворк языка php, который использует паттерн MVC в несколько раз увеличивает как продуктивность, так и упрощенный способ написания, задней части веб приложения на объектно-ориентированном языка PHP.

Связка Laravel и VueJs это огромный потенциал в создании динамического веб приложения. Это дает не только удобную синхронную разработку, но и превращает веб-сайт в мощное приложение, с полной асинхронной обработкой как пользовательских данных, так и серверных.

Большинство сайтов в интернете – это простые сайты, использующие скудный набор механизмов. Мы же в этой статье опишем создание простого веб приложение с использованием полного механизма современных инструментов.

Алгоритм создания веб приложения

Пошаговая инструкция создания проекта:

1. Первым делом нужно развернуть сам ларавел 5.7 проект на компьютере;

2. Следующее – это работа с базами данных, а именно, создать файл модели и миграций;
3. После, настроить файл маршрутов со стороны back-end, для перехвата запросов от браузера и работы по api (интерфейс прикладного программирования, англ. application programming interface);
4. Затем создаем контроллер, который отвечает за вывод главной страницы, отдает данные по существующим задачам, добавляет задачу и завершает задачу в базах данных.
5. Настраиваем вывод представления главной страницы, где присоединяем скрипты VueJs (библиотека javascript). С последующей полной настройкой асинхронного функционала веб-приложения (вывод, добавить, просмотр и завершение задачи).

Создание менеджера задач

Мы будем создавать простой функционал менеджера задач, с использованием базы данных MySql. Сам продукт, будет иметь возможность добавление новой задачи в базу данных, чтения всего списка задач, единичного завершения задачи и все это будет работать асинхронно. Заключительное изображение рабочей программы на рис. 2.

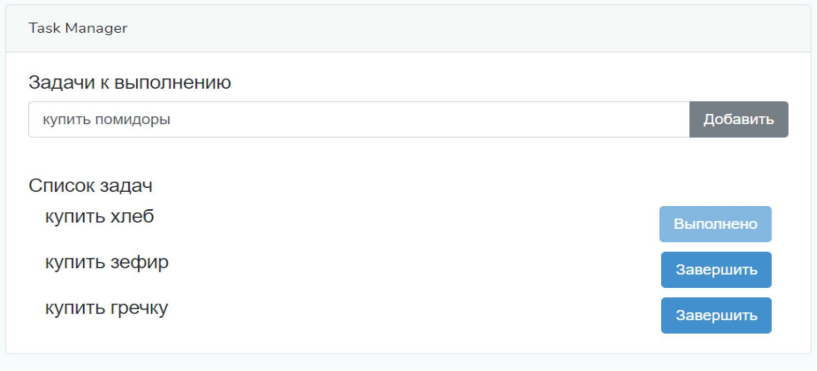

Рис. 2. Конечная программа менеджера задач

Также, не мало знать изначально структуру файлов и директорий проекта (рис. 3).

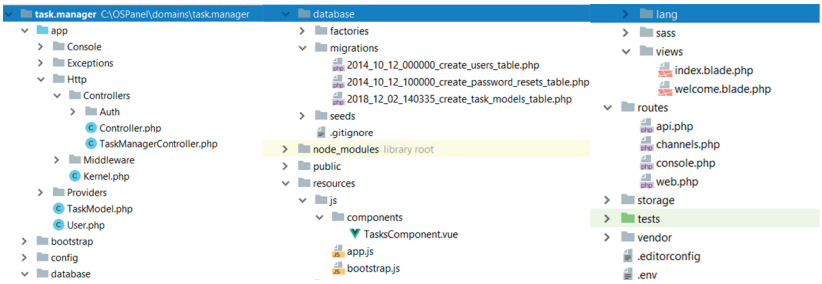

Рис. 3. Структура каталогов и файлов проекта

Первый этап. Развертывание проекта. На компьютере уже должны быть установлены веб окружение (стандартный веб сервер, например php 7.1+MySql+phpMyAdmin+apache) и инструмент управления зависимостями в PHP – Composer.

Скачиваем через композер чистый проект ларавел последней версии командой:

```
composer global require laravel/installer
```

Следующее, устанавливаем его в нужной нам директории командой:

```
laravel new blog
```

где, “blog” это название проекта.

Второй этап. Настройка миграций и работы проекта с базами данных.

Миграции – это что-то вроде системы контроля версий для вашей базы данных. Они позволяют команде программистов изменять структуру БД, в то же время оставаясь в курсе изменений других участников [8, с. 16].

Модели базы данных – содержит в себе бизнес логику приложения, или как приложение взаимодействует с базой данных.

Перед тем, как начать создавать файлы миграций и модели, нам нужно создать саму базу данных и настроить подключение к базе данных в файле .env (файл настроек, который содержит

логины и пароли разных служб, сейчас актуально – работа с базой данных).

В phpMyAdmin, вкладке баз данных – создаем базу данных «task.manager», указав кодировку “utf8_general_ci”. В файле .env вводим данные для подключения с созданной базой данных:

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=task.manager
DB_USERNAME=root
DB_PASSWORD=
```

Для создании файлов миграций и модели, мы будем пользоваться командой в командной строке – php artisan make:model TaskModel – migration. Файл миграций будет содержать метод для создания таблицы, с его полями и свойствами (рис. 4).

```
class CreateTaskModelsTable extends Migration
{
        /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create( table: 'task_models', function (Blueprint $table) {
            $table->increments( column: 'id');
            $table->text( column: 'name');
            $table->text( column: 'state');
            $table->timestamps();
        });
    }
```

Рис. 4. Файл миграций по созданию таблицы task_models

где, name это имя задания, а state это состояние (завершено или актуально).

В файле модели, будем указывать имя таблицы, для связывания данного файла. Внутри класса нужно объявить свойство $table с присвоением значения имени таблицы:

```
public $table = 'task_models';
```

После выполнения этих операций, нам нужно запустить миграций, для установки таблицы в базу данных, командой:

```
php artisan migrate
```

Третий этап. Настройка маршрутов в серверной стороне.

Маршруты – это файл, который отвечает за контроль путей, которые были введены в браузере, с перенаправлением их по своим контроллерам или выполнив скрипты в самом файле роутинга. В ларавел проекте, файл маршрутов находиться в директории «routes», и название этого файла – "web.php". Для работы с главной страницей приложения мы пропишем следующий код:

```
Route::get('/', "TaskManagerController@index");
```

Данная строка отлавливает главную страницу сайта и перенаправляет его обработку в метод «index» класса TaskManagerController.

Также, для асинхронной работы клиентской части с серверной, нам нужны маршруты по работе с api, которые будут отвечать запросам от front-end части:

```
Route::get('get-tasks', 'TaskManagerController@getTasks');
Route::post('add-task', 'TaskManagerController@addTask');
Route::post('finish-task', 'TaskManagerController@finishTask');
```

Где, первая строка, маршрут «get-tasks» это получение всех задач из метода «getTasks», класса «TaskManagerController». Следующий маршрут – «add-task», это маршрут, который принимает post данные в методе «addTask», в классе «TaskManagerController». Конечный маршрут «finish-task», принимающий post данные для завершения конкретной задачи, обрабатывается методом «finishTask», в том же классе.

Четвертый этап. Настройка работы мозга операций и приложения в целом, контроллера, – TaskManagerController.php. Первым делом, нам нужно обработать обращение первого маршрута, запроса главной страницы:

```
public function index(){
 return view('index');
}
```

Метод index() возвращает файл представления index.blade.php.

Также, нам необходимо прописать методы по обработке запросов по api:

```
public function getTasks(){
 return TaskModel::all();
}
```

Данная функция – getTasks() отвечает за возврат всех существующих задач.

Следующий метод отвечает за добавление новой задачи в базу данных, с использованием ранее созданной модели «TaskModel.php»:

```
public function addTask(Request $request){
 $task = $request->input('task');
 $record = new TaskModel();
 $record->name = $task;
 $record->state = 'actual';
 $record->save();
}
```

Где, мы создаем новый объект $task, с последующим его заполнением данными, после его сохранив в базе данных

Конечный метод отвечает за изменение состояния задачи, от состояния «актуально», к состоянию «завершено».

```
public function finishTask(Request $request){
 $id = $request->input('id');
 $record = TaskModel::find($id);
 $record->state = 'finish';
 $record->save();
}
```

Здесь мы получаем post данные, извлекаем значение отправленного идентификатора задачи, ищем его в таблице методом «find()» через модель «TaskModel», найдя нашу строку, мы меняем его поле «state» на «finish». В заключений, это все сохраняем под тем же идентификатором задачи.

Пятый этап. Настройка всего функционала представления, с асинхронной работой клиентской части с серверной т.е. работа без полной перезагрузки страницы.

Настройка файла javascript – TaskComponent.vue, файл, который является главным в клиентской части, в нем содержится сам html код, и методы обработчики и вывода данных из серверной части. Html код будет храниться внутри тегов <template></template>, на рис. 5 проиллюстрирован html код файла. Перед началом написания vuejs файла, рекомендуется запустить в командной строке обработчик, который будет собирать vue компоненты в один js файл, сохраняя его по пути public/js/app.js, командой – npm run watch.

Рис. 5. Часть html VueJs компонента

В VueJs компоненте также содержится и javascript код, который помещается между тегами <script></script>. В нем написали три метода, которые обращаются серверной части, для получения всех задач, для добавления новой задачи и изменения выбранной задачи по идентификатору. На рис. 6 отображена скриптовая часть данного компонента:

Рис. 6. Часть сценарий VueJs компонента – TaskComponent.vue

В методе mount() в vuejs, отрабатывает при загрузке страницы, в нашем случае он запускает метод getTasks().

Окромя, написание кода в vuejs компонете, нам также необходимо его инициализировать в index.blade.php файле представлений, заранее установив его в vuejs обработчике. На рис. 7 можете видеть файл index.blade.html и app.js.

```
<!Doctype html>
<html>
    <head>
        <title>Task Manager</title>

        <link href="css/app.css" rel="stylesheet" type="text/css">
    </head>
    <body>
    <div id="app">
        <br>
        <tasks-component></tasks-component>
    </div>

    <script src="js/app.js"></script>
    </body>
</html>
```

```
require('./bootstrap');

window.Vue = require('vue');

Vue.component('tasks-component',
    require('./components/TasksComponent.vue'));

const app = new Vue({
    el: '#app',
});
```

Рис. 7. На левой стороне изображения index.blade.php, а с правой app.js

В файле index.blade.php, обязательно добавление app.js в конце, перед закрытием тега <body>. А также, добавление к основному тегу идентификатора «app», который установлен в TaskComponent. vue для связи с текущем компонентом. Внутри него, мы уже можем обращаться к данному компоненту по его имени, указав его как тэг, нашем случае это – «tasks-component». А в файле app.js, необходимо установить данный компонент, дав ему имя для связи в html файле представления.

Заключение

Таким образом было написано простейшее веб-приложение с использованием современных технологий. Это все дает грандиозные возможности, для создания мощного веб-приложения, при взаимодействии асинхронного подхода между клиентской частью и серверной.

Список литературы

1. Сайт [электронный ресурс] // Материал из Википедии – свободной энциклопедии. Свободный режим доступа: https://ru.wikipedia.org/wiki/%D0%A1%D0%B0%D0%B9%D1%82 (дата обращения 11.12.2018).

2. HTML [электронный ресурс] // Материал из Википедии – свободной энциклопедии. Свободный режим доступа: https://ru.wikipedia.org/wiki/HTML (дата обращения 13.12.2018).
3. CSS [электронный ресурс] // Материал из Википедии – свободной энциклопедии. Свободный режим доступа: https://ru.wikipedia.org/wiki/CSS (дата обращения 13.12.2018).
4. Javascript [электронный ресурс] // Материал из Википедии – свободной энциклопеди. Свободный режим доступа: https://ru.wikipedia.org/wiki/JavaScript (дата обращения 13.12.2018).
5. Введение в Vuejs [электронный ресурс] // Головной русскоязычный сайт VueJS. Свободный режим доступа: https://ru.vuejs.org/v2/guide/index.html (дата обращения 13.12.2018).
6. Laravel – экосистема, а не просто PHP-фреймворк [электронный ресурс] // «Хабр» – крупнейший в Европе ресурс для IT-специалистов. Свободный режим доступа: https://habr.com/post/334776/ (дата обращения 15.12.2018).
7. Настраиваем вашу первую модель Laravel 4 [электронный ресурс] // Русскоязычный портал Ларавел. Свободный режим доступа: https://laravel.ru/posts/38 (дата обращения 15.12.2018).
8. Миграции в ларавел [электронный ресурс] // Русскоязычный портал Ларавел. Свободный режим доступа: http://laravel.su/docs/5.0/migrations (дата обращения: 15.12.2018).
9. PHP-роутинг (Routing) для новичков [электронный ресурс] // Блог вебмастера. Свободный режим доступа: http://maxsite.org/page/routing (дата обращения: 16.12.2018).
10. Быстрый роутинг на PHP [электронный ресурс] // «Хабр» – крупнейший в Европе ресурс для IT-специалистов. Свободный режим доступа: https://habr.com/post/134406/ (дата обращения 16.12.2018).
11. Асинхронный UI: будущее веб-интерфейсов [электронный ресурс] // «Хабр» – крупнейший в Европе ресурс для IT-специалистов. Свободный режим доступа: https://habr.com/post/132834/ (дата обращения 16.12.2018).
12. Front-end and back-end. Interaction in plain words [электронный ресурс] // Lvivity сайт интересных решений. Свободный режим до-

ступа: https://lvivity.com/front-end-back-end-interaction (дата обращения 12.12.2018).
13. PHP page information [электронный ресурс] // Официальный портал языка – Php. Свободный режим доступа: http://php.net/ (дата обращения 16.12.2018).
14. About Vuejs [электронный ресурс] // Официальный портал языка – VueJs. Свободный режим доступа: https://vuejs.org/ (дата обращения 16.12.2018).
15. Event Handling. Listening to events in VueJs [электронный ресурс] // Официальный портал языка – VueJs. Свободный режим доступа: https://vuejs.org/v2/guide/events.html (дата обращения 16.12.2018).

References

1. Sayt [Site]. *Wikipedia*. https://ru.wikipedia.org/wiki/%D0%A1%D0%B0%D0%B9%D1%82
2. HTML. *Wikipedia*. https://ru.wikipedia.org/wiki/HTML
3. CSS. *Wikipedia*. https://ru.wikipedia.org/wiki/CSS
4. Javascript. *Wikipedia*. https://ru.wikipedia.org/wiki/JavaScript
5. Vvedenie v Vuejs [Introduction to Vuejs]. *Golovnoy russkoyazychnyy sayt VueJS* [Leading Russian site VueJS]. https://ru.vuejs.org/v2/guide/index.html
6. Laravel – ekosistema, a ne prosto PHP-freymvork [Laravel is an ecosystem, not just a PHP framework]. «Khabr» – krupneyshiy v Evrope resurs dlya IT-spetsialistov ["Habr" is the largest resource for IT specialists in Europe]. https://habr.com/post/334776/
7. Nastraivaem vashu pervuyu model' Laravel 4 [Set up your first Laravel 4 model]. *Russkoyazychnyy portal Laravel* [Russian portal Laravel]. https://laravel.ru/posts/38
8. Migratsii v laravel [Migrations to Laravel]. *Russkoyazychnyy portal Laravel* [Russian portal Laravel]. http://laravel.su/docs/5.0/migrations
9. PHP-routing (Routing) dlya novichkov [PHP routing (Routing) for beginners]. *Blog vebmastera* [Blog webmasters]. http://maxsite.org/page/routing

10. Bystryy routing na PHP [Fast PHP routing]. «Khabr» – krupneyshiy v Evrope resurs dlya IT-spetsialistov ["Habr" is the largest resource for IT specialists in Europe]. https://habr.com/post/134406/
11. Asinkhronnyy UI: budushchee veb-interfeysov [Asynchronous UI: the future of web interfaces]. «Khabr» – krupneyshiy v Evrope resurs dlya IT-spetsialistov ["Habr" is the largest resource for IT specialists in Europe]. https://habr.com/post/132834/
12. Front-end and back-end. Interaction in plain words [Front-end and back-end. Interaction in plain words]. *Lvivity sayt interesnykh resheniy* [Lvivity site of interesting decisions]. https://lvivity.com/front-end-back-end-interaction
13. PHP page information [PHP page information]. *Ofitsial'nyy portal yazyka – Php* [The official portal of the language – Php]. http://php.net/
14. About Vuejs [About Vuejs]. *Ofitsial'nyy portal yazyka – VueJs* [Official portal of the language – VueJs]. https://vuejs.org/
15. Event Handling. Listening to events in VueJs [electronic resource]. *Ofitsial'nyy portal yazyka – VueJs* [Official portal of the language – VueJs]. https://vuejs.org/v2/guide/events.html

ДАННЫЕ ОБ АВТОРАХ

Бадыков Ильнар Василович, студент группы 23403, кафедра «Информационные технологии»
Набережночелнинский филиал Казанского национального исследовательского технического университета им. А.Н. Туполева
ул. Академика Королева, 1, г. Набережные Челны, республика Татарстан, Российская Федерация
ilnarb98@mail.ru

Хамидуллин Марат Раисович, доцент, кандидат экономических наук
Набережночелнинский филиал Казанского национального исследовательского технического университета им. А.Н. Туполева
ул. Академика Королева, 1, г. Набережные Челны, республика Татарстан, Российская Федерация
nayka_prom@mail.ru

DATA ABOUT THE AUTHORS

Badykov Ilnar Vasilovich, Student Group 23403, Department of "Information Technology"
Kazan National Research Technical University
1, Akademika Koroleva str., Naberezhnye Chelny, Republic of Tatarstan, Russian Federation
ilnarb98@mail.ru

Khamidullin Marat Raisovich, PhD in Economics
Kazan National Research Technical University
1, Akademika Koroleva str., Naberezhnye Chelny, Republic of Tatarstan, Russian Federation
nayka_prom@mail.ru
ORCID: 0000-0002-3326-0955