УДК 004

Компонентный подход в разработке веб приложений

Голубь Илья Сергеевич
Приамурский государственный университет имени Шолом-Алейхема
Магистрант

Научный руководитель:
Глаголев Владимир Александрович
Приамурский государственный университет имени Шолом-Алейхема
к.г.н., доцент кафедры информационных систем, математики и правовой информатики

Аннотация
В данной работе приведен пример разработки веб-приложения с использованием компонентного подхода на основе VueJs и yii2 php. Сравнивается стандартная структура приложения и структура при компонентном подходе. Сделан вывод, о том, что данная структура поможет увеличить читабельность кода и упростить доработку приложения.
Ключевые слова: VueJs, php, yii2, разработка, веб-приложение, Компонентный подход.

Component Approach to Web Application Development

Golub Ilya Sergeevich
Sholom Aleichem Priamursky State University
Undergraduate

Scientific adviser:
Glagolev Vladimir Alexandrovich
Sholom Aleichem Priamursky State University
Candidate of Geographical Sciences, Associate Professor of the Department of Information Systems, Mathematics and Legal Informatics

Abstract
This paper gives an example of developing a web application using a component approach based on VueJs and yii2 php. The standard structure of the application and the structure of the component approach are compared. It is concluded that this structure will help increase code readability and simplify application development.
Keywords: VueJs, php, yii2, development, web application, component approach.

На данный момент активно продвигается front и backend разработка. Так компонентный подход во frontend’е заключается в разбиение приложения на составляющие, которые можно использовать не однократно и

не для одного приложения, т.е. компоненты должны быть пере используемыми. В backend’е этот поход затрагивает микросервисную архитектуру, чем лучше и понятные будет разбиение на Микросервисы, тем больше шансов использовать их в других приложениях. Так же, компонентный подход во frontend’е позволяет использовать отдельные функции серверной части приложения, обращаясь с ним непосредственно из разных компонентов.

В статье Е. О. Казначеевой, И. Б. государева рассмотрены два подхода, используемые при разработке современных веб-приложений: модульный и компонентный. Целью обоих подходов является компоновка написанного кода для повторного использования. Рассмотрены каждый из упомянутых подходов, а также выделены их общие черты и различия. Сделан вывод о дальнейшем использовании модулей и компонентов при разработке веб-приложений[4].

В работе А. А. Акулова, Р. Р. Бариев приведены три фреймворка, каждый из них был описан со стороны их отличия друг от друга. Проанализированы их статистики использования среди разработчиков[2].

В статье А. С. Гарбарчука рассмотрена архитектура SPA как технология создания современных сайтов, показаны её преимущества и влияние на конечную производительность сайта[3].

В книге P. So рассказывается о том, что Vue.js является одним из самых эффективных JavaScript-проектов за последние годы благодаря своей гибкости и внедрению в такие сообщества, как Laravel. Его концепция в первую очередь ориентирована на идеал постепенной адаптации , при котором Vue.js может использоваться в качестве библиотеки и исключительно в качестве декоратора пользовательских интерфейсов или в качестве полноценной среды для высокопрофессиональных архитектур. Эта миссия отличает Vue.js от некоторых других проектов, которые мы рассмотрим в этих главах[1].

Целью данной работы является рассмотреть компонентный подход при использовании Фреймворков VueJs и yii2 php.

Методом исследования является сравнение обычного создания приложения и создание приложения при помощи компонентного подхода. Сейчас большую популярность приобретает микросервисная архитектура приложений, которая подразумевает то, что приложение будет разбито на отдельные сервисы, которые будут практически не зависимы друг от друга. Так же, как и серверная часть приложения, которая разбивается на компоненты, плюс компонентного подхода в том, что каждый отдельный компонент может быть использован не ограниченное количество раз, так же, компонент может быть использован во всех других компонентах, что является большим плюсом при возникновении потребности в доработке. Так же, структура проекта на yii2 строится по паттерну Model-View-Controller(MVC).

При создании стандартного приложения, используя стандартные подходы, мы получаем стандартный набор папок, например infrastructure,

services и так далее, пере используемыми файлами в данной структуре будут являться сторонние библиотеки, расширяющие стандартные возможности языка. Выбрав компонентный подход можно использовать, например, написанный компонент отображающий календарь и обрабатывающий выбор даты пользователем. Данные компоненты могут быть использованы в разных частях приложения, что означает, что авторский компонент с календарем и обработкой данных пользователем может быть использован на главной странице, на под страницах, а так же в модальных окнах. При этом, мы не будем дублировать или переписывать код, а просто пере используем компонент.

Компонентный подход в создании приложений упрощает не только работу разработчика в части ухода от дублирования кода, но и добавляет ему возможность создавать свои компоненты. Например можно легко создать свой собственный компонент таким образом:

```
namespace app\components;

use yii\base\Component;
use yii\helpers\Html;

class MessageComponent extends Component{ // объявляем класс
    public $content;

    public function init(){ // функция инициализации. Если данные не будут переданы в компонент, то он выведет текст "Текст по умолчанию"
        parent::init();
        $this->content= 'Текст по умолчанию';
    }

    public function display($content=null){ // функция отображения данных
        if($content!=null){ //проверка строки на пустоту
            $this->content= $content;
        }
        echo Html::encode($this->content); // вывод данных
    }

}
```

Рис. 1. Компонент MessageComponent

После написания кода компонента его необходимо зарегрестрировать в файле config/web.php.

```
'components' => [
    'message' => [
      'class' => 'app\components\MessageComponent',
    ],
],
```

Рис. 2. Регистрация компонента

В свою очередь лицевая часть проекта, которая рендерится на стороне пользователя, а именно frontend, при помощи современных фреймворков таких, как например VueJs, тоже следует делить на компоненты. Отдельные компоненты в frontend’е так же могут быть переиспользованы как внутри одного проекта, так и внутри разных проектов, делаются они тоже очень легко.

```
// Определяем новый компонент, названный button-counter
Vue.component('button-counter', {
  data: function () {
    return {
      count: 0
    }
  },
  template: '<button v-on:click="count++">Счётчик кликов – {{ count }}</button>'
})
```

Рис. 3. Компонент button-counter

С помощью привязки к событиям во VueJs v-on мы привязываемся к клику по кнопке и увеличиваем счетчик. Теперь нам надо наш компонент. Компонент объявляется в теге скрипт в специальном виде:

import button-counter from ‘./button-counter’

а так же требуется объявить его в специальном теге.

export default {components: button-counter }.

```
<div id="components-demo">
  <button-counter></button-counter>
</div>
```

В отдельных компонентах можно использовать обращение к разным микросервисам или разным частям монолитного приложения. Так же, создавая экземпляры компонентов и при обращении этих компонентов к одному и тому же endpoint’у данные будут не зависимы, т.к. данные компонентов хранятся внутри самих компонентов. Так же, существует библиотека Vuex, для хранения состояния приложения, т.е. для сохранения каких то общих данных к котором должен осуществляться доступ из любой точки приложения.

Как видно из рассмотренного материала, использование данных подходов и паттернов не является сложным и не вызывает особых затруднений.

Рассмотрев данные подходы можно сказать, что данные подходы и паттерны положительно повлияют на приложение, на читабельность кода и снизят сложности при доработки такого приложения.

Библиографический список

1. So P. Vue. js //Decoupled Drupal in Practice. – Apress, Berkeley, CA, 2018. – С. 381-397.
2. Акулов А. А., Бариев Р. Р. Исследование популярных javascript фреймворков для разработки клиентской части веб-приложения

//Цифровизация экономики: направления, методы, инструменты. – 2019. – С. 358-362.
3. Гарбарчук А. С., Гриценко Е. М. Применение SPA-архитектуры для улучшения производительности сайта на примере проекта" Вулкан-М" // Научное и образовательное пространство: перспективы развития. 2018. С. 158-160.
4. Казначеева Е. О., Государев И. Б. Модульный и компонентный подходы в программировании // Альманах научных работ молодых ученых университета итмо. 2018. С. 150-152.