Marin Kaluža[1]
Krešimir Troskot[2]
Bernard Vukelić[3]

Professional paper
UDC 004.03

COMPARISON OF FRONT-END FRAMEWORKS FOR WEB APPLICATIONS DEVELOPMENT[4]

ABSTRACT

Modern web applications, due to the functionalities they provide in their user interfaces, have a complex program structure. Manually writing a program code, due to the complexity of the entire application, can result in uneven quality and content of individual application parts. Maintaining such developed applications is more difficult. Because of this, web applications are often developed by using different frameworks. A framework allows structuring, simpler and more uniform program script writing, and thus easier web application maintenance. There are various frameworks that can be used in the development of web applications, for different parts of the application. Those analyzed in this paper are used in the development of front end parts of web applications. According to their design, a web application can be developed as the Multi Page (MPA) or the Single Page (SPA). This paper explains the difference between MPA and SPA web applications. The advantages and disadvantages of MPA are demonstrated in relation to SPA web applications. Required characteristics that the framework should have in order to be optimized for creating MPA and SPA web applications are set. The hypothesis has been tested: There is a framework that is optimized for the development of both MPA and SPA applications. Possibilities, architecture and development techniques of a web application using front end frameworks, as well as the suitability of such frameworks for the development of MPA and SPA web applications have been analysed. Choosing a framework for the hypothesis testing has been performed based on the popularity of available frameworks. The required characteristics have been analyzed on the three most popular frameworks: Angular, Vue.js and React-js. It has been shown that the Vue.js framework is the most optimized framework for the development of both MPA and SPA applications.

Key words: *SPA, MPA, framework*

1. INTRODUCTION

In recent years, there has been a great demand for highly sophisticated and complex web applications, in order to replace the old desktop application in all areas. There is also the need to create desktop applications for mobile devices. The complex web applications can be the so called Multi Page applications (MPA) and Single Page applications (SPA).

[1] PhD, Senior Lecturer, Polytechnic of Rijeka, Vukovarska 58, 51 000 Rijeka, Croatia. E-mail: mkaluza@veleri.hr
[2] BSc in Informatics, Student, Polytechnic of Rijeka, Vukovarska 58, 51 000 Rijeka, Croatia. E-mail: ktroskot@veleri.hr
[3] MSc, Senior Lecturer, Polytechnic of Rijeka, Vukovarska 58, 51 000 Rijeka, Croatia. E-mail: bvukelic@veleri.hr
[4] Received: 15 January 2018; Accepted: 1 March 2018

Web application development is often based on the use of a certain framework (FW). Are frequently used FWs for creating web applications in line with the development of both MPA and SPA applications? Which FW is easier to use in developing web applications? Are FWs suitable for managing front end parts of a web application?

The purpose of the paper is to demonstrate the suitability of the available FWs for the creation of MPA and SPA web applications. The aim is to define key features that enable a more customized development of MPA and SPA web applications.

Hypothetically, it can be said that there is a front end FW which is optimized for creating MPA and SPA applications.

This paper will explain the meanings of MPA and SPA, and the need to develop such web applications. The issues affecting the development of MPA and SPA applications will be defined. Frequently used FWs will be analysed by qualitative assessments of each issue affecting the development of MPA and SPA applications.

2. RELATED WORK

To create a good user experience, it is important to choose a suitable web application architecture. A more detailed analysis of architectural features and user needs is often required when selecting between a Single Page Application (SPA) and a Multi Page Application (MPA). MPA applications are usually intended for larger systems with a large number of different types of services, which prefer more types of interactions with their visitors. SPA is a newer approach to web application development and is often used in the development of simpler applications with less content (Dimi, 2017).

Multi-page applications work in a “traditional” way. Any change in the browser (e.g.. display or transmission of data) involves retrieving new pages from the server (Neoteric, 2016). Routes are registered on the server, and each HTTP request from the client to server involves retrieving a new HTML page. This means that the request sent to the server will always retrieve the page showing the results of the request (so called view), or an error. Most of the application logic is on the server, and the client is only the recipient of the retrieved page.

This web application development approach is suitable for developing smaller applications and when, for example, Javascript[5] is used for simpler activities performed over the loaded page (user interface) or for animation development. However, if there is a need to create a more demanding user interface, the page may become very complex and it can be loaded with lots of data and program elements (Javascript is often used) that load the browser and the client computer. Since the creation of complex pages on the server and their transfer and presentation to the client requires a lot of time and degrades user experience, at the beginning of 2000s the MPA was improved by introducing AJAX[6], which enabled refreshing only parts of

[5] JavaScript is a lightweight, interpreted, programming language with first-class functions. - URL https://developer.mozilla.org/bm/docs/Web/JavaScript

[6] AJAX stands for Asynchronous JavaScript And XML. - URL https://developer.mozilla.org/en-US/docs/Web/Guide/AJAX/Getting_Started

a page rather than entire pages. This technique has helped to improve user experience, but it has also enabled the development of more complex web pages, making the management of the source code more complex. This is one of the reasons why FWs appear (Shimanovsky, 2016).

SPA is a complete web application with only one page that serves as a "shell" for all parts of the user interface. Most resources (Javascript, HTML5[7], CSS[8]) are loaded only once throughout the application work cycle, and data is transmitted from previous to new state. The initial HTML document uploaded at the first opening of an application represents the starting point for the rest of the application. Each subsequent part is loaded dynamically and independently of the "shell", without re-loading the whole page, giving the user the perception that the page has changed. The shell is mostly minimal in structure and often contains a single, empty tag (<div>) that will "host" the rest of the application content (Emmit, Scott, 2016).

After the initial request to the server, the browser loads the complete HTML page. Each subsequent client and server interaction is performed using AJAX technique, which means that the browser only updates a part of the page that changes (data) without refreshing the entire page (Saxena, 2014).

The main difference between MPA and SPA work cycles is in the nature of the request and the response, i.e. one MPA work cycle ends by receiving the response, and the SPA work cycle lasts and depends on the work of the user through the user interface. SPAs use AJAX when submitting a request to a server, and as a response they receive data (usually in JSON[9]) and receive small parts of HTML to view received data. Once the data arrives from the server, the client side will form the received content and display it at a specific location on the shell.

Some of the advantages of the SPA compared to the MPA are the following:

- Quick Response Time - Since SPA takes on the structure of the site in advance, there is no need for constant requests to obtain a new site from the server. When a user clicks on an area, changes are made "instantly", which gives a sense of interaction with mobile or desktop applications. The user does not have to wait or, at least, does not have that feeling of waiting. This is a great advantage and the reason why SPAs are so popular today.
- Separating presentation from business logic – The code that manages the user interface behaviour is contained on the client side instead on the server. This allows the developer to focus on what is important to the user experience and to the critical business logic elements on the server. Thus, mixing presentation and business logic is more difficult, allowing you to maintain and update each side separately.

[7] HTML5 is the latest evolution of the standard that defines HTML. URL https://developer.mozilla.org/en-US/docs/Web/Guide/HTML/HTML5

[8] Cascading Style Sheets (CSS) is a stylesheet language used to describe the presentation of a document written in HTML or XML. URL https://developer.mozilla.org/en-US/docs/Web/CSS

[9] JSON (JavaScript Object Notation) is a lightweight data-interchange format. URL - https://www.json.org/

- Faster and easier data transfer (bandwidth) - Transactions with the server are easier and faster because, after initial delivery, only data is sent to or received from the server (Emmit, Scott, 2016, 13).
- Possibility of offline support - Since SPA uses Javascript running on the client side, the connection to the server is not necessary all the time. It is possible to ensure that the application still works when the user is not connected or temporarily does not use the Internet connection. In that case, local data on the client will be synchronized with those on the server (Apps Team, 2013).

There are certain disadvantages of the SPA in relation to the MPA:

- Search engine optimization (SEO) is difficult to implement. It refers to optimizing a website (application) to achieve the highest rank in ranking content on search engines (Fink, Flatow, 2014). Each search engine has three functions:

    1. crawling – it refers to the disclosure of web page information. This includes location scanning and collecting details on each page: headlines, images, keywords, other related pages, etc.
    2. indexing – processing and storing the data collected through crawling in the database.
    3. serving – the purpose of this function is to retrieve the relevant content at the time of the user's query in the search engine (Bruce, 2016).

  When a user performs a specific action on the user interface in the SPA, and new data needs to be downloaded, Javascript will adjust only part of the page without refreshing and opening a new one. The problem arises because crawling involves scanning content on the basis of URLs and page changes, and the SPA has only one initial page. The crawler will not know that the page has changed (Mikowski, 2014).

  SEO for SPA has been improving every day. Today, there are tools and ways that enable the visualization of parts of an application on the server so that the crawler can see what the user sees. However, this is still in the development and more difficult to manage than in the other (MPA) approach (Zanon, 2015).

- Javascript must be enabled - If the user disables Javascript in the browser, the application will not be shown the way it should. Since such users account for 1.3%, this can be a problem (Hein, 2010).
- Security requires more work - this principle is not unsafe, but since this way of creating applications is relatively new, some security issues are partially resolved and more effort is needed to study ways and best practices to address specific security issues.

3. RESEARCH METHODOLOGY

3. 1 Analyzed platforms and frameworks

The following will explain the issues that will be analyzed for FW adaptation checking for MPAs and SPAs development. Given that there are differences between MPA and SPA applications, there are also different needs in the developmnet of such applications. For this reason, separate issues will be defined for the frameworks adaptation analysis for the development of MPAs and SPAs.

Table 1. Questions for analysis regarding the MPA

Question	Explanation
MPA	
Is only js import sufficient to get started?	For MPAs, which often have a lot of pages on which they control smaller parts of DOM[1], a convenient and easy startup can be just by adding scripts. This also refers to the creation of simpler components, and the FW that enables this is considered more manageable and easier to use. (Burgess, 2016)
Does the framework include too much overhead?	Since choosing a particular framework often entails a large number of already built-in plugins, this issue concerns the ability of the FW to provide the developer with a higher degree of control over the size of the application, i.e. by selecting only the necessary parts of the FW (Neuhaus, 2017). i.e. whether writing a lot of code is also necessary for creating smaller functionalities?
Is the workflow necessary?	Does the framework require some workflow to be used when creating each page.
Server – side rendering	If the goal is to build a Multi-Page application, one of the reasons might be a better optimization the application in the search engines. Server-side rendering (SSR) refers to the execution of the application on the server, allowing the crawlers to see the content. Therefore, this will provide the answer on the question whether FW has the SSR option. (React Community, 2016)
SPA	
The ability to write and maintain a large quantity of complex code	A cleaner, more organized JavaScript code and a well-modulated architecture, in which each part of the application forms a separate part, is a step forward in building a scalable and sustainable SPA (Takada, 2013) (Bruce, 2016). Using modules also helps to achieve data integrity, program code organization, and avoiding recurring names. Without such an approach to building SPA, the work with global variables and functions, above all, would soon become unmanageable (Emmit, Scott, 2016, 129). In order for the above to be possible, the following is necessary: a) workflow – which will enable the creation of applications by modules and which, by providing the user with the right tools and following the best practices, will properly bundle all the modules into one js document; b) the existence of the best practice - available in the documentation and describes the naming of files and folders since it is the only way for the application to be sustainable.

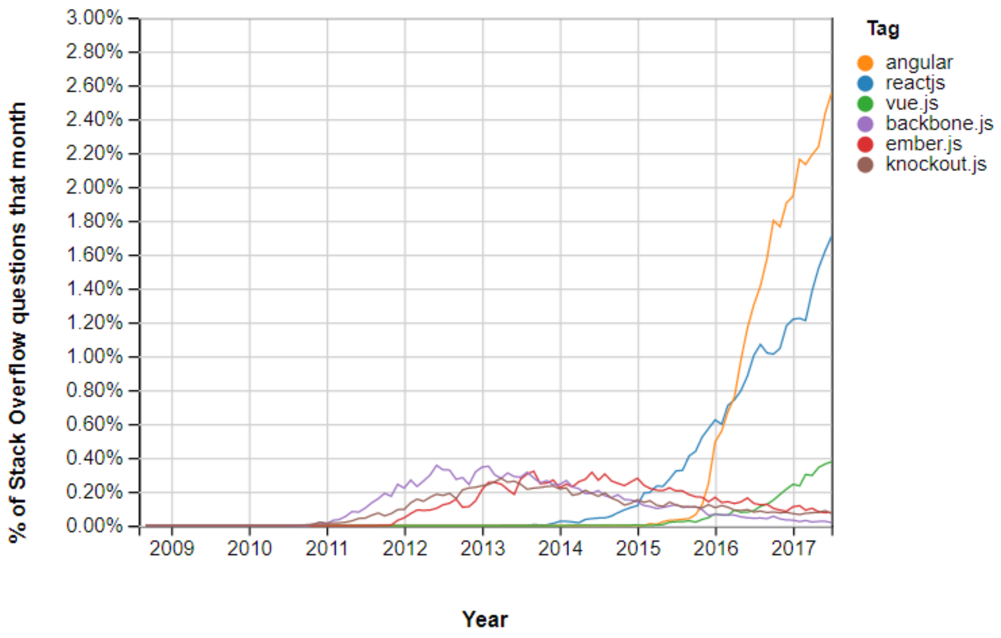
Question	Explanation
MPA	
Should the developer rely on a large quantity of third party packages?	This issue refers to the question whether the FW is complete, i.e. whether all the packages needed to build a SPA are already embedded in it or is there an officially issued package, or is it necessary to receive community packages, which could be a problem. Can we rely on the maintenance (promptness of updates) of the package? This issue is very important when it comes to building a SPA where, since it is a larger and more complex Javascript application, all these packages will really be needed and there should be less worrying about finding other packages for missing features. This just means more dependency to worry about and more potential points of failure (Köhr, Schlaudraff, 2017).
What is the possibility of compressing the application, minimizing the bundle file?	One of the challenges of building SPA is the speed of initial opening of the site. In order for speed to be as fast as possible, the bundle file, or script that must be downloaded in order to launch the application, should be reduced to the smallest possible size. It refers to the process and simplicity of the transformation of the development program code into a concise production one (Dias, 2016).
Data state management	Development of the SPA is often complex because the application work cycle does not stop, and user activity on the application results in changes in the application state on the client. Some changes in the application state come through AJAX, but most of the changes do no come from the server, but from the local variables in the application. Various application state managers can be used for coherent DOM update. This question will answer whether there is a clear way to utilize the FW to ensure DOM state management. Is there an officially issued package or it is on the developer who, in such a case, must rely on the community (Botto, 2016)?

Source: analysed by authors

The three most popular Javascript front end FWs will be compared, the popularity of which continues to grow. The figure below shows the ranking of all available Javascript FWs. Since this work refers to FWs for the front end side, the following are isolated and will be analysed: Angular, React and Vue.js.

React, with over 86,000 Github stars (https://github.com/facebook/react), is considered probably the most popular Javascript front end FW. Vue.js, with about 81,000 Github stars (https://github.com/vuejs/vue), is one of the fastest growing when it comes to popularity (developed only in 2016). On the other hand, Angular with its huge backdrop, Google (from whom it has been developed) and Microsoft, whose language it uses (Typescript), is one of the most stable. The analysis (Figure 1) was carried out among the six most well-known Javascript front end FWs, and it was established that only the three mentioned ones were currently gaining popularity while the others were stagnating.

Figure 1. The popularity scale of the 6 most well-known Javascript frameworks



Source: https://insights.stackoverflow.com/trends?tags=reactjs%2Cangular%2Cvue.js%2Cember.js%2Cbackbone.js%2Cknockout.js

Vue.js is today one of the fastest growing Javascript FWs when it comes to popularity. On its official pages it is described as accessible, comprehensive and high performing FW for development of interactive interfaces (Vue, 2018-1). Since FW is component-oriented and the component, after a certain internal behaviour / computation, returns the UI template in the form of output, all of these components are reusable within the page, or within other components. Vue.js uses virtual DOM which does not exist in the browser but in the memory, and the result is a much faster access than in the actual DOM. Such updated virtual DOM is in the end presented as actual DOM (Andersen, 2017). This FW is most popular in the Far East, and some of the companies that use it are Alibaba, Xiaomi, Baidu, Tencent etc. (Clockwise Software, 2017).

React is presented as a "Javascript library for building user interfaces". It was released in March 2013 by Facebook, which uses the React components on several of its pages, instead of one SPA (Neuhaus, 2017). The same as Like Vue.js, React uses a component-based architecture, which makes it possible to write code in a simpler and more sustainable way (React, 2018-1). Although not mandatory, the components are usually written in JSX (Javascript XML), a React-specific extension of Javascript, which allows the use of HTML within Javascript (React, 2018-2). Since it uses virtual DOM, React creates the data structure in the memory, calculates differences between virtual and real DOM, and then updates the actual DOM in the browser in a very fast and efficient way (Kurian, 2017).

Angular is FW for creating client applications in HTML and JavaScript or in a language such as TypeScript compiled into JavaScript. It was developed by Google in 2010 as AngularJS, and in 2014 it was completely reworked and rewritten and since then it has been operating under the name Angular. FW consists of several libraries, some of which are basic and some optional. Angular applications are developed by writing HTML templates using "angularized" tagging, writing component classes to manage these templates, adding application logic to services, and registering components and service in modules. The application is launched by loading the root module. Angular takes over the presentation of the application content in the browser and responds to the user's interaction according to the instructions you have entered (Angular, 2018-1). Angular 4 will be used in the analyses in this paper.

3. 2 FRAMEWORK ANALYSIS

Below, an analysis of the three separate FWs will be performed according to all the questions listed in Tables 1 and 2. The degree to which the conditions are met will be explained and defined in the questions from tables for the following FWs: Vue.js, React and Angular.

3.2.1 Is only js import sufficient to get started?

If we want to control just one small part of the DOM, the best way of starting with a FW is by adding just a script. Possible answers to this question will be either yes or no.

Working with Vue.js FW is entirely possible by simply adding a script via CDN[10] or locally by downloading from the official site. A quick start is also possible through NPM[11]. There is no workflow required and there is no need to use Vue.js files that come with it (Vue, 2018-2).

In Angular, you cannot get started by adding script. Workflow is needed due to the very concept in which this FW works (a modular model based on components created with TypeScript) and therefore it requires project-building tools. Typescript is an open-source language developed by Microsoft and it is a superset of Javascript that provides additional features such as optional static types, classes, modules, and the like (TypeScript, 2017). Since

[10] Content delivery networks (CDN) is the transparent backbone of the Internet in charge of content delivery. URL - https://www.incapsula.com/cdn-guide/what-is-cdn-how-it-works.html

[11] Node package manager (NPM) is the package manager for JavaScript and the world's largest software registry. URL - https://www.npmjs.com/

browsers do not understand this language, it is necessary to first translate it into Javascript, after which, like any other Javascript code, it can be easily executed in the browser. Translation in Angular is performed using a Typescript compiler that is already embedded within the Webpack tool that comes with this FW (Angular, 2018-2).

Although the React.js official site strongly supports getting started by creating workflow, it can be executed by simply adding a script, just as it was done with Vue.js (React, 2018-3). In this case, it is necessary to write in ECMAScript[12] 5 (ES5) since the browsers are still incompatible with ECMAScript 6 (ES6) (Zaytsev, 2018). ECMAScript is a scripting language standard that specifies the basic features that a scripting language should provide and how to implement them.

Both Typescript and ES6 can be compiled into a browser with specific packages, allowing you to send uncompiled code to the browser. This leads to a lot of unnecessary code, more precisely, the entire compiler being sent. In addition, compilation would take place at initial loading of the application, which would result in poor user experience. For these reasons, this option will not be considered (Elliott, 2016).

3.2.2 Does the framework include too much overhead?

Although some FWs come with a bunch of additional features and packages some projects may not require them. Particularly when it comes to MPA, they will not be needed on each page, and if left out, this would result in unnecessary overhead. Possible answers to this question will be either yes or no.

Angular comes with a lot of add-ons. Although optimization can cut off a lot of unnecessary code, the structure of the background on which FW "lies" is "burdened with code". It requires lots of code and it is complicated for controlling a small part of the page. It uses its own module loading system. It uses its own "angularized" language, which uses structural directives, pipes, declarations, modules, injectors, services, decorators etc. (Yiang, 2016)(Cordle, 2017).

Vue.js, on the other hand, does not include o lot of overhead. This FW includes a lot of elements that might not be needed on one page, however it enables the activation of those required elements. Also, there is no concept that is exaggerated.

React is also quite "simple". It was built for the purpose of managing the front end part of the application and its original orientation (purpose) is focused on controlling and manipulating the view. It allows selecting exactly what the developer needs, but also it provides the ability to use a large number of third-party packages. Accordingly, React does not include a lot of overhead.

3.2.3 Is the workflow necessary?

To create simpler components, it would be desirable to use simple js import. Over time, when more functions are added, for example, routing, the state controller, and the http module, a

[12] EcmaScript (ES) is the standardized scripting language that JavaScript implement.
URL – https://www.wintellect.com/5-great-features-in-es6-harmony/

workflow that will connect all components will be required. This question refers to the need for workflow at the beginning of work with FW, which could be unacceptable for the purpose of MPA development and control of a small part of DOM.

Angular requires a workflow. Typescript is a language that is not understood by browsers, therefore the compiling is needed. Also, it is necessary to gather all the files in the bundle, and to do a lot of optimization (compression), all of which the build workflow enables.

Vue.js does not put emphasis on the workflow and it does not need it. Getting started is simple by adding js script. In this case, it is necessary to work in ES5. Getting started by setting up a workflow could be useful and and could result in certain benefits but is not entirely needed (Vue, 2018-3).

React partially requires a workflow. It can be used without it, but when its ES6 language is used, a workflow is required. ES6, unlike ES5, allows you to write a cleaner and "simpler" code or code with fewer lines (React, 2018-4). Examples of program code in React's official documentation are written in ES6, which may be a complicating factor for a user who wants to work in ES5.

3.2.4 Server – side rendering

This is an essential requirement if high search engine optimization is the goal for the web application. As the MPA can be made up of several smaller SPAs, it is relevant for the FW to have this option that will enable the rendering of each of these SPAs on the server.

Angular has released an official guide for server-side rendering which is Angular Universal. This is the central part that connects Node.js[13] with Angular by running the same application on both the server and in the browser. It generates static pages in the applications on the server through the Server-Side Rendering (SSR) process. It can generate and serve these pages in response to browser requests. Moreover, it can also pre-generate pages as HTML files that will be later served (Angular, 2018-3).

There is also an officially released Vue.js guide for building Vue application rendered on the server. The guide is placed on a special domain, separate from Vue documentation. It is very detailed and it assumes that the user is already familiar with Vue.js FW and that he has a decent knowledge of Node.js and Webpack. Also, the documentation refers to Nuxt.js, a FW issued by the community and which is presented as a higher degree solution (Vue, 2018-4). It offers certain additional features, however, it limits the developer's direct control over the application structure (Vue, 2018-5).

React lacks an official documentation on SSR. The React API includes an object called ReactDOMServer, and its purpose is to display the components in the form of a HTML code. Also, it is necessary to learn how to use other packages, such as React Router and Redux, in order for the server-side and client-side rendering to be performed without any problems. The React team announced that official support will soon be released. There is also a FW that can be used to create a React SSR application, called Next.js. Accordingly, React enables SSR, but without official support, and with the use of additional third-party packages (Bejar, 2017).

[13] Node.js is a JavaScript runtime built on Chrome's V8 JavaScript engine. URL - https://nodejs.org/en/

3.2.5 The ability to write and maintain a large quantity of complex code

SPA applications have a complex structure and a large amount of program parts, which are often placed in a lot of files. In order to make such a complex and "rich" structure sustainable, it is important to have a workflow and the best practice of naming files and folders. Possible answers to this question will be yes, no or partially.

Angular allows for a large quantity of complex code to be written and maintained. There are many best practices for naming and structuring files and they are all published on the Angular official page in a special section of the Styleguide. This section contains a guide designed for Angular application syntax, conventions and structure, with a detailed explanation of their purpose. As mentioned above, Angular includes a workflow through the CLI[14], which allows you to set a workflow, a building process, in a very simple way (Angular, 2018-4).

There is no clear best practice in Vue.js. If the approach to registering components is followed, as shown in the official documentation, it is apparent that kebab-case[15], camelCase[16] or PascalCase[17] names are allowed, which may make it difficult for major applications to be maintained (Vue, 2018-6). This does not mean that it could not be done and that the developer could not use a specified access, but rather that it is not explicitly specified and the developer is able to select. This requires consideration at the beginning of the development. Vue.js, as well as Angular, allows to create a workflow through the CLI, and therefore this FW is marked as "partially".

Best practices for React are found in the community. Creating a workflow is enabled through the official create-react-app package (Abranov, 2016). Due to the above, React partly provides the possibility to write and maintain a lot of complex code.

3.2.6 Should the developer rely on third party packages?

As mentioned above, this issue deals with the packages needed for building the SPA. Are they already contained in it or is there an officially released package, or do we need to rely on the community to download appropriate packages? Possible answers to this question will be yes, no or partially.

Angular does not require a lot of third-party packages. There are also community packages, however, most of these packages are already included in Angular. The Angular team provides all packages needed for creating a complete SPA. Here are some of the packages that are not included

[14] Command Line Interface (CLI) is a user interface to a computer's operating system or an application in which the user responds to a visual prompt by typing in a command on a specified line, receives a response back from the system, and then enters another command, and so forth. URL – http://searchwindowsserver.techtarget.com/definition/command-line-interface-CLI

[15] The naming style used in LispLanguage and related languages. All lowercase with - separating words. URL – http://wiki.c2.com/?KebabCase

[16] CamelCase describes a compound word which uses capital letters to delimit the word parts. URL – https://www.computerhope.com/jargon/c/camelcas.htm

[17] The first letter in the identifier and the first letter of each subsequent concatenated word are capitalized. URL – https://msdn.microsoft.com/en-us/library/x2dbyw72(v=vs.71).aspx

in other FWs, such as: form validation packages and an official router that allows navigating the application page when there is a change in URL. In addition to the aforementioned packages, there is also an Angular Material package that provides material style and certain pre-built components that can be included in the application. There is a lot of official packages with Angular, and this practically reassures that the packages will remain reviewed and updated on time (Angular, 2018-5).

Vue.js has an official router package and a Vuex state management package that are maintained and updated on time. The form validation package is not included in this FW, so the developer is forced to use a third party package that may not be completed or maintained. This could be a problem and it depends on the functionality of a developing application. This is why Vue.js has to rely partly on third-party packages (Vue, 2018).

React requires a lot of third-party packages. State management packages, router and form validation package are not official packages and it is necessary to rely on a community where multiple options often exist, and this presents a difficult choice for new developers in this area. Also, a specific experience is required when it comes to choosing between all these packages, often with very similar purposes (Npmjs, 2013-2017).

3.2.7 What is the possibility of compressing the application, minimizing the bundle file?

The most common cause of poorly optimized SPA is the bundle file size. This is the file that must be downloaded at the initial page load. It results in slow initial page load and, therefore, its size needs to be reduced as much as possible.

When developing an application using Angular FW, a user writes their development code that is run in the browser on the development server. In that case, templates will be delivered to the browser, which means they will not be compiled. The HTML code inside the templates where the Angular Syntax is used (e.g. ngFor[18], ngIf[19], etc.) will be delivered to a browser that does not understand it. For this reason, Angular includes a compiler which is responsible for "understanding" such HTML code written in components, and for compiling during running (Just-in-Time). This will increase the size of the bundle file and reduce the performance of the application running. In order for the performance to increase, Ahead-of-Time (AOT) should be started as a part of the workflow. This type of compilation makes it possible for the entire code to be compiled in Javascript during the workflow, which means that only Javascript code will be delivered to the browser without the Angular Compiler and without the need to compile and parse the templates during the execution. This is not the default Angular option and needs to be activated through the CLI, and in most cases it is activated by switching to the production mode of the application (Angular, 2018-6) (Yiang, 2016) (Zia, 2017).

Vue.js allows a developer to choose between two approaches: runtime-only and runtime+compiler. Runtime-only approach implies non-delivery of a compiler. This means that instead of the template within the instance, a developer need to use ".vue" files that will

[18] Built-in Angular directive for iterating through array. URL - https://angular.io/guide/displaying-data
[19] Built-in Angular directive for testing if something is or not. URL - https://angular.io/api/common/NgIf

be compiled into Javascript. These files along with a normal HTML code also contain a vue-specific code compiled as part of a workflow, so what is delivered to a browser at the end is an optimized code that is only Javascript and does not include any compiler nor any HTML code. Runtime + compiler approach allows to control small parts of a DOM by using templates (components) within an instance. In this case, ".vue" files do not need to be used, but in doing so, the bundle file is not highly optimized, which means it is bigger and the performance of application is decreased (Vue, 2018-7).

The optimization process with React is minimal. The entire program code is already contained in Javascript and there is no HTML code that needs to be parsed.

All three FWs can be further optimized by using Lazy loading, Treeshaking, or by manual enhancement of code. Lazy loading enables splitting applications into modules. This means that only the component that is needed at a given time will be loaded (Basal, 2017). Treeshaking also enables activation only of required program code parts. This means removing all that is unnecessary in our code. This option is included as basic in Angular (Zia, 2017). Since Angular's bundle file size can be as twice as large, it is important to conduct an optimization process (Osbourne, 2017).

All three FWs enable a simple optimization process through CLI, but Angular requires more optimization.

3.2.8 Data state management

During the use of the application, changes in the state of the application are performed, which means that there are frequent changes in the states of the individual modules. Maintaining the changes in the state of individual modules may be a problem for the developer. Managing the state is one of the most difficult parts of the SPA development in a safe, and time-wise sustainable environment. This question will answer whether there is a clear method and an officially issued package for managing the state.

In Angular, packages provided by an official team include state management services and they are sufficient for small and medium-sized applications development. For larger Angular applications, with lots of asynchronous activities and where there are many states that will be shared and manipulated through multiple components and modules, managing the state by using the service can be demanding. The package ngrx/store can be used for state management (Rangle, 2017). It is a third-party package that enables managing the state of the application via observables (Ngrx, 2017). Observables help in managing asynchronous data, such as data coming from a backend. This package is well integrated with Angular because both use observables (Looper, 2017). The Ngrx package allows to manage a state, but the problem is that this third-party package is developed by a community developer group. It can become deprecated, be rarely maintained or may change often. This package does not support Angular 5, and the project that uses this package cannot be updated to a newer version (Bersling, 2017). Angular allows to manage the state of a large SPA by using a non-official package.

For Vue.js, there is Vuex. This is an official package that is included in Vue.js. It facilitates the state management, even for larger applications. It serves as a central storage for all components in the application with rules ensuring mutation of the state performed only in a predictable manner. Vuex does not limit the way the code is structured, so it is important to learn the proper way of management by following best practices. Since Vuex is the official package of Vue.js, Vue.js provides the best way to manage the state (Vue, 2018-8).

In React, state management is enabled by isolating the state of the components at the highest level and then switching it to lower levels as properties, and allowing top-down state flow. In general, the state management is carried out quite elegantly. The problem may arise when component trees are large, and when there is a need to change state with structurally distant components, or in case when one component is not descendant of the other, and both components depend on the same state. In order to avoid "multiple sources of truth", it would be right to store the common state in the nearest common ancestor, or any other common ancestor and send it down as properties. This is a good approach for small and medium-sized applications, but it is "clumsy" for large applications, and performance problems are also possible (Crawford, 2016). Therefore, a package for state management should be used. For this, the following packages are usually used: Redux and Flux. The packages are regularly maintained. With React, it is possible to efficiently manage the state of the application in large and very large applications, but to a somewhat lesser extent than with Vue.js, for which an official package exists.

4. DISCUSSION

Table 3 shows the results of FW analysis. The first column contains questions that were analysed in FWs. Other columns show the results of the analysis expressed as text (with possible values: No, Partially, Yes) and as figures (with possible values: 0, 1, 1.5, 2).

The rows of the table describing the results of the analysis according to individually analysed FWs are divided into groups MPA and SPA, so the questions below the group name belong to that group.

Table 3. Framework analysis

Question	**Angular**		**Vue.js**		**React.js**	
MPA						
Is only js import sufficient to get started?	No	0	Yes	2	Yes	1.5
Does the framework include too much overhead?	Yes	0	No	2	No	2
Is the workflow necessary?	Yes	0	No	2	Partly	1
Server – side rendering	Yes	2	Yes	2	Yes	1
SPA						
The ability to write and maintain a large quantity of complex code	Yes	2	Partly	1	Partly	1
Should the developer rely on third party packages?	No	2	Partly	1	Yes	0
What is the possibility of compressing the application, minimizing the bundle file?	Yes	2	Yes	2	Yes	2
Data state management	Yes	1	Yes	2	Yes	1.5

Source: analysed by authors

In the first part of the table, which represents the analysis in the context of MPA development, there is a difference between Angular and Vue.js. The sum of the first three questions is 0 for Angular, and 6, which is the maximum, for Vue.js. This shows that Vue.js, compared to Angular, is "simple" and can be used as a library. In certain questions, React.js meets or, at least, partially meets the criteria. Both Angular and Vue.js enable server-side rendering, through official package. React.js enables server-side rendering through third-party package, as the official has only been announced.

The second part of the table shows the analysis in the context of the SPA application building. Angular is the FW which, in comparison to other analysed FWs, is the best for writing and maintaining a lot of complex code. Other FWs lack a clearly defined file and folder naming practices, and therefore they only partly enable writing and maintaining a lot of complex code. Angular, in complete contrast to React.js, does not require reliance on a lot of third-party packages, while Vue.js requires it only partially. All three FWs allow for easy optimization, while management of the state is, for now, officially available only with Vue.js.

Table 4. Results of numerical values of framework analysis

Framework	**MPA**	**SPA**	**TOTAL**
Vue.js	8	6	14
React.js	5.5	4.5	10
Angular	2	7	9

Source: analysed by authors

Table 4 shows the results of numerical values of FW analysis by groups MPA and SPA. The last column contains the sum of MPA and SPA. The table shows that Vue.js is the most suitable FW analysed for developing MPA. The second one is React.js which is somewhat less suitable for developing MPA. Angular, according to the results, is not suitable for the development of MPAs. Due to its size, complexity, and high-optimization needs, it is not tailored to control small parts of the DOM.

According to the results for SPA, Angular is the most suitable (from all the FWs analysed) for developing such applications. Vue.js comes second, with a slightly lower, but still high result when compared to Angular. The worst result is recorded for React.js which is not fully adapted to the development of SPAs, although the developed software community makes its use possible, so improvements can be expected.

5. CONCLUSION

This paper has analysed the suitability of the front end frameworks for the development of MPA and SPA web applications. The characteristics of MPA and SPA have been explained. The difference between MPAs in comparison to SPAs has been shown. The reasons for the selection have been explained and the Angular, Vue.js and React-js front end frameworks have been described. The elements that have been analysed in selected frameworks, have been set and described in the form of questions.

According to the results of the analysis of the selected frameworks, due to similar and high values in the MPA and SPA categories, the Vue.js framework has proved to be suitable for the development of MPA and SPA applications. React also has similar results in both MPA and SPA categories, but of lower values, which shows that it can also be used in the development of MPA and SPA web applications. Angular, according to the results of the analysis, is not suitable for the development of MPAs. Considering that Angular has the best result in SPA category, Angular is currently the best framework for creating SPA applications.

Given that the results of the analysis showed similar results in two cases, it can be concluded that the most popular frameworks are suitable for the development of both types of MPA and SPA web applications, with the exception of Angular which is, due to its structure and abilities, more suitable for the development of SPA web applications.

REFERENCES

Abranov, D. (2016) *Create Apps with No Configuration https://reactjs.org/blog/2016/07/22/create-apps-with-no-configuration.html* (17.12.2018.)

Andersen, B. (2017) *Understanding the Virtual DOM https://codingexplained.com/coding/front-end/vue-js/understanding-virtual-dom* (13.12.2017.)

Angular (2018-1) *Architecture Overview https://angular.io/guide/architecture* (23.12.2017.)

Angular (2018-2) *Webpack https://angular.io/guide/webpack*(23.12.2017.)

Angular (2018-3) *Angular Universal: server-side rendering https://angular.io/guide/universal* (19.12.2017.)

Angular (2018-4) *Style Guide https://angular.io/guide/styleguide* (17.12.2018.)

Angular (2018-5) *Npm Packages https://angular.io/guide/npm-packages* (22. 12.2017.)

Angular (2018-6) *The Ahead-of-Time (AOT) Compiler https://angular.io/guide/aot-compiler* (11.1.2018.)

Apps Team (2013) *An Intro Into Single Page Applications (SPA) https://blog.4psa.com/an-intro-into-single-page-applications-spa/* (10.12.2017.)

Basal, N. (2017) *Optimizing the Performance of Your Angular Application https://netbasal.com/optimizing-the-performance-of-your-angular-application-f222f1c16354* (5.1.2018.)

Bejar, J. (2017) *A Review Of Server Side Rendering In Javascript Frameworks https://wyeworks.com/blog/2017/9/6/a-review-of-ssr-in-javascript-frameworks* (17.12.2018.)

Bersling (2017) *State Management: ngrx/store vs Angular services https://www.bersling.com/2017/06/05/state-management-ngrxstore-vs-angular-services/* (11.1.2018.)

Bruce, J. (2017) *How Do Search Engines Work? http://www.makeuseof.com/tag/how-do-search-engines-work-makeuseof-explains/* (13.12.2017.)

Burgess, M. (2016) *JavaScript Frameworks Are Great https://medium.com/@mattburgess/javascript-frameworks-are-great-2df4a3f0b24d* (10.12.2017.)

Clockwise Software (2017) Frontend frameworks showdown: Angular vs. React vs. Vue https://clockwise.software/blog/angular-vs-react-vs-vue/ (13.12.2017.)

Cordle, C. (2017) Why Angular 2/4 Is Too Little, Too Late https://medium.com/@chriscordle/why-angular-2-4-is-too-little-too-late-ea86d7fa0bae (13.1.2018.)

Crawford, C. (2016) What the Flux? An Overview of the React State Management Ecosystem https://thenewstack.io/flux-overview-react-state-management-ecosystem/ (13.1.2018.)

Dias, A. (2016) Challenges in a (really) large single-page application - how to optimize script downloads http://alvarodias.org/articles/challenges-in-a-really-large-single-page-application-how-to-optimize-script-downloads (10.12.2017.)

Dimi (2017) Discover Single-Page Vs. Multipage Design: Pros & Cons (2017) https://themefuse.com/single-page-vs-multipage-design/ (20.12.2017.)

Elliott, E. (2016) Angular 2 vs React: The Ultimate Dance Off https://medium.com/javascript-scene/angular-2-vs-react-the-ultimate-dance-off-60e7dfbc379c (23.12.2017.)

Emmit, A., Scott, Jr., (2016) SPA Design and Architecture: Understanding single-page web applications https://github.com/transidai1705/javascript-ebooks/blob/master/%5BSPA%20Design%20and%20Architecture%20Understanding%20Single%20Page%20Web%20Applications%201st%20Edition%20by%20Emmit%20Scott%20-%202016%5D.pdf (20.11.2017.)

Fink, G., Flatow, I., (2014) Pro Single Page Application Development: Using Backbone.js and ASP.NET http://pepa.holla.cz/wp-content/uploads/2015/10/Pro-Single-Page-Application-Development.pdf (15.11.2017.)

Hein, R. (2010) How Many Users Have JavaScript Disabled https://www.searchenginepeople.com/blog/stats-no-javascript.html (13.12.2017.)

Köhr, J., Schlaudraff, J. (2017) Angular 2 in a multi-page application https://blog.novatec-gmbh.de/angular-2-in-a-multi-page-application/ (20.12.2017.)

Kurian, G., G. (2017) How Virtual-DOM and diffing works in React https://medium.com/@gethylgeorge/how-virtual-dom-and-diffing-works-in-react-6fc805f9f84e (13.12.2017.)

Looper, J. (2017) An Introduction to Observables for Angular Developers https://developer.telerik.com/topics/web-development/introduction-observables-angular-developers/ (11.1.2018.)

Mikowski, M. (2014) How to optimize single-Page sites for search engines https://www.webdesignerdepot.com/2013/10/how-to-optimize-single-page-sites-for-search-engines/ (13.12.2017.)

Neoteric (2016) Single-page application vs. multiple-page application https://medium.com/@NeotericEU/single-page-application-vs-multiple-page-application-2591588efe58 (20.12.2017.)

Neuhaus, J. (2017) Angular vs. React vs. Vue: A 2017 comparison https://medium.com/unicorn-supplies/angular-vs-react-vs-vue-a-2017-comparison-c5c52d620176 (10.12.2017.)

Ngrx (2017) Ngrx Store https://github.com/ngrx/store (5.1.2018.)

Npmjs (2013-2017) React Packages https://www.npmjs.com/search?q=react&page=1&ranking=popularity (22.12.2017.)

Osbourne, S. (2017) JavaScript Framework Battle: 'Hello World' in each CLI https://medium.com/dailyjs/javascript-framework-battle-hello-world-in-each-cli-cfdba8bf5e4b (5.1.2018.)

Rangle (2017) Angular 2 Training Book https://angular-2-training-book.rangle.io/handout/state-management/ (5.1.2018.)

React (2018-1) React.js https://reactjs.org/ (13.12.2017.)

React (2018-2) Introducing JSX https://reactjs.org/docs/introducing-jsx.html (13.12.2017.)

React (2018-3) Try React https://reactjs.org/docs/try-react.html (23.12.2017.)

React (2018-4) *React Without ES6 https://reactjs.org/docs/react-without-es6.html (*19.12.2017.)

React Community (2017) Server Rendering https://github.com/reactjs/redux/blob/master/docs/recipes/ServerRendering.md (19.12.2017.)

Saxena, R. (2014) Single Page Application (SPA) Using AngularJS, Web API and MVC 5 http://www.c-sharpcorner.com/uploadfile/rahul4_saxena/single-page-application-spa-using-angularjs-web-api-and-m/ (10.12.2017.)

Shimanovsky, S. (2016) How is PowerApps giving control back to the business? http://www.eikospartners.com/blog/multi-page-web-applications-vs.-single-page-web-applications (10.12.2017.)

Takada, M. (2013) Single page apps in depth http://singlepageappbook.com/index.html (15.11.2017.)

TypeScript (2017) Typescriptlang https://www.typescriptlang.org/index.html (23.12.2017.)

Vue (2018) Plugins https://vuejs.org/v2/guide/plugins.html (22. 12.2017.)

Vue (2018-1) https://vuejs.org/ (13.12.2017.)

Vue (2018-2) Explanation of Different Builds https://vuejs.org/v2/guide/installation.html#Explanation-of-Different-Builds (23.12.2017.)

Vue (2018-3) Introduction https://vuejs.org/v2/guide/ (19.12.2017.)

Vue (2018-4) Server-Side Rendering https://vuejs.org/v2/guide/ssr.html (19.12.2017.)

Vue (2018-5) Vue.js Server-Side Rendering Guide https://ssr.vuejs.org/en/ (17.12.2018.)

Vue (2018-6) Component Naming Conventions https://vuejs.org/v2/guide/components.html#Component-Naming-Conventions (17.12.2018.)

Vue (2018-7) Runtime + Compiler vs. Runtime-only https://github.com/vuejs/vue/tree/dev/dist#runtime--compiler-vs-runtime-only (27.12.2018.)

Vue (2018-8) State Management https://vuejs.org/v2/guide/state-management.html (13.1.2018.)

Yiang, Y., Z. (2016) Angular 2 is terrible https://meebleforp.com/blog/36/angular-2-is-terrible (19.12.2017.)

Zanon, D. (2015) AngularJS: How to create a SPA crawlable and SEO friendly? https://zanon.io/posts/angularjs-how-to-create-a-spa-crawlable-and-seo-friendly (13.12.2017.)

Zaytsev, J. (2018) Combat Table https://kangax.github.io/compat-table/es6/ (23.12.2017.)

Zia, D. (2017) Optimizing Angular 2 Applications http://www.discoversdk.com/blog/optimizing-angular-2-applications (27.12.2018.)

Marin Kaluža[1]
Krešimir Troskot[2]
Bernard Vukelić[3]

Stručni rad
UDK 004.03

USPOREDBA FRONT END FRAMEWORKA ZA IZRADU *WEB*-APLIKACIJA [4]

SAŽETAK

Moderne web-*aplikacije, zbog funkcionalnosti koje omogućuju u korisničkom sučelju, imaju složenu programsku strukturu. Ručno pisanje programskog koda zbog složenosti cijele aplikacije može rezultirati neujednačenom kvalitetom i sadržajem pojedinih aplikacijskih djelova. Održavanje tako razvijanih aplikacija otežano je. Zbog toga se* web-*aplikacije često razvijaju korištenjem različitih* frameworka. Framework *omogućuje strukturiranje, jednostavnije i ujednačenije pisanje programskog koda, te time olakšava održavanje* web-*aplikacije. Postoji puno* frameworka *koji se mogu koristiti u razvoju* web-*aplikacija, i to za različite dijelove aplikacije, a oni analizirani u ovom radu koriste se u razvoju* front end *dijela* web-*aplikacije. Prema načinu izvođenja* web-*aplikacije mogu biti* Multi Page *(MPA) ili* Single Page *(SPA). U radu je objašnjena različitost između MPA i SPA* web-*aplikacija. Pokazane su prednosti i nedostatci MPA u odnosu na SPA* web-*aplikacije. Postavljene su zahtijevane karakteristike* frameworka *koji je optimiziran za izradu MPA i SPA* web-*aplikacija. Testirana je hipoteza: Postoji* framework *koji je prilagođen za izradu i MPA i SPA aplikacija. Analizirane su mogućnosti, arhitektura i načini razvoja* web-*aplikacija pomoću* front end frameworka *te prilagođenost takvih* frameworka *za razvoj MPA i SPA* web-*aplikacija. Izvršen je odabir* frameworka *za testiranje hipoteze prema popularnosti na tržištu. Zahtijevane karakteristike analizirane su na 3 najpopularnija* frameworka*: Angular, Vue.js i React-js. Pokazano je da je Vue.js* framework *najoptimiziraniji* framework *za izradu i MPA i SPA aplikacija.*

Ključne riječi: *SPA, MPA, framework*

[1] Dr. sc., viši predavač, Veleučilište u Rijeci, Vukovarska 58, 51 000 Rijeka, Hrvatska. *E-mail*: mkaluza@veleri.hr
[2] Bacc. inf., student, Veleučilište u Rijeci, Vukovarska 58, 51 000 Rijeka, Hrvatska. *E-mail*: ktroskot@veleri.hr
[3] Mr. sc., viši predavač, Veleučilište u Rijeci, Vukovarska 58, 51 000 Rijeka, Hrvatska. *E-mail*: bvukelic@veleri.hr
[4] Datum primitka rada: 15. 1. 2018.; datum prihvaćanja rada: 1. 3. 2018.