ИКОНОМИЧЕСКИ УНИВЕРСИТЕТ – ВАРНА
КАТЕДРА „ИНФОРМАТИКА“

Р Е Ф Е Р А Т

НА ТЕМА:

Предимства на React. Разработка на уеб приложение.

ПО ДИСЦИПЛИНА:

„Езици за програмиране“

Изготвил: докт. Йордан Николаев Вълков

Варна 2020 г.

Съдържание

Увод

Създаването на клиентската част на уеб приложенията изисква задълбочени познания на – HTML, CSS, JavaScript, известни като „троицата“ на фронтенд разработката. Всяка една от тези технологии търпи своята еволюция, което от своя страна поражда необходимостта от изучаване на съвременни уеб технологии, като: SCSS/SASS, LESS, Bootstrap, ZURB Foundation, jQuery, ECMAScript, React, Angular, VueJS и мн. др.

През последното десетилетие SPA[1] станаха популярни. Работни рамки като Angular, Ember и Backbone помогнаха на JavaScript разработчиците да изграждат съвременни уеб приложения. Списъкът не е изчерпателен. Съществува широка гама от SPA работни рамки. Повечето от изброените са от първото поколение. Angular и Backbone са създадени през 2010 г., а Ember през 2011 г. Две години по-късно Facebook създава библиотеката React. Цялата екосистема около React прави възможно изграждането на SPA. Тя е иновативна и допринася за намаляване на времето и разходите за разработка. Възприета е от множество водещи, технологични компании, като Airbnb, Netflix и разбира се Facebook. Всички те заедно с огромна общност от разработчици, инвестират в бъдещето на React.

Освен чистата React библиотека, разработчиците могат да използват и други React версии. Те включват React Native за мобилни приложения, ReactJS.NET за крос-платформени приложения.

Целта на този реферат е да се разкрият предимствата и недостатъците на React, да се идентифицират стъпките по инсталация, както и да бъде разработено примерно уеб приложение.

[1] Single Page Application, Wikipedia, 2020 <https://en.wikipedia.org/wiki/Single-page_application> (15.02.2020)

I. Предимства и недостатъци на React

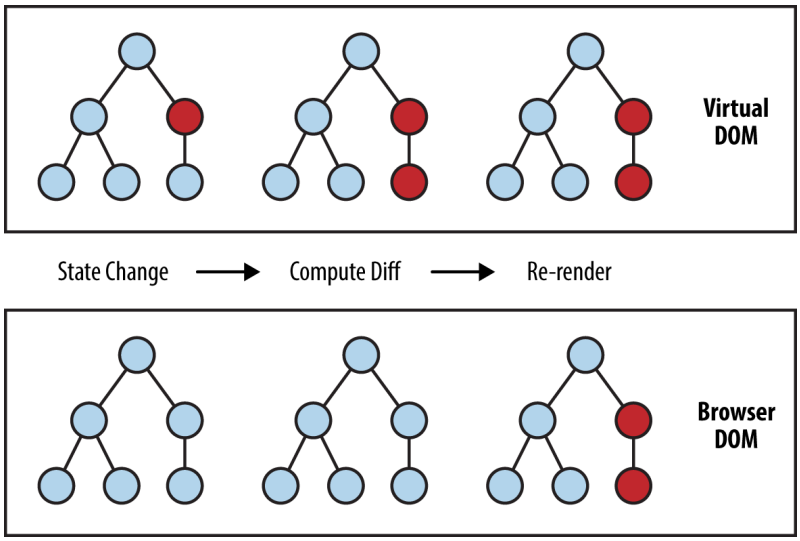
В тази част ще бъдат извършени сравнения между React и един от основните конкуренти - Angular, както и ще бъдат разкрити предимствата и недостатъците на React.

Библиотека срещу работна рамка.

React не е MVC[2] работна рамка като Angular, а библиотека, която представлява само "V" (View) слоя и е най-подходяща за изграждане на интерактивни потребителски интерфейси. React позволява единствено да бъдат рендирани компонентите, като видими елементи в браузъра. Angular се счита за работна рамка, защото разполага с много вградени функционалности. Например, не е необходимо да бъде инсталиран рутер за url адресите на страниците, защото Angular разполага с тази функционалност. React дава повече гъвкавост и свобода да бъдат инсталирани пакети или библиотеки, които са необходими за конкретния проект. Angular предлага ясно как трябва да бъде структурирано приложението, докато при React в различните проекти е възможна различна йерархия и организация на папките.

Виртуален DOM срещу реален DOM.

Angular си служи единствено с реалния DOM на браузъра, а React използва уникално рендиране във виртуален DOM[3] изобразен на Фигура 1.



Фигура 1. Виртуалният DOM намалява повторното рендиране

[2] Model View Controller, Wikipedia, 2020 <https://en.wikipedia.org/wiki/Model-view-controller> (15.02.2020)
[3] Document Object Model, Wikipedia, 2020 <https://en.wikipedia.org/wiki/Document_Object_Model> (15.02.2020)

Червените кръгове представляват променените възли. Тези възли са елементи на потребителския интерфейс, които са променили състоянието си. Изчислява се разликата между предишната версия на виртуалното DOM дърво и текущото виртуално DOM дърво. Цялото родителско под-дърво се рендира, за да даде обновения потребителски интерфейс. Обновеното дърво се актуализира пакетно до реалния DOM. Следователно това води до повишена производителност, а актуализациите към реалния DOM се изпращат на групи, вместо да се изпращат актуализации за всяка промяна в състоянието. Например, ако е необходимо да се актуализира възрастта на потребител в HTML елемент, виртуалния DOM ще види разликите между предходната и текущата версия на елемента и ще обнови само него. В същата ситуация реалния DOM ще обнови цялата HTML структура, докато достигне елемента с възрастта на потребителя. Естествено този пример е тривиален, но в едно реално приложение е възможно да има стотици заявки за данни на една и съща страница. Това би се отразило драстично на общата производителност.

Компонентно-базирана архитектура.

React е известна сред разработчиците със своята висока гъвкавост и ефикасност. Изгражда добре капсулирани компоненти, които управляват собствено вътрешно състояние. Дава възможност за комбиниране на тези компоненти и създаването на сложни потребителски интерфейси. Angular също разполага с компонентно-базирана архитектура. В двете технологии е много важно, компонентите лесно да могат да бъдат използвани отново за създаване на други компоненти или дори да бъдат използвани в други проекти.

Крива на обучението.

Съществено значение при избора на нова технология е нейната крива на обучение. Отговорът зависи от предишен опит и познаване на свързани с технологията понятия. Необходимо е да бъде направена оценка на новите неща, които трябва да бъдат научени.

В React трябва да бъдат изучени елементарен JSX синтаксис, как да бъдат написани компоненти, как да се управлява тяхното вътрешно състояние и как да бъдат използвани props. Не е необходимо изучаване на нови логически структури

или цикли, тъй като всичко това е обикновен JavaScript. След като бъдат изучени основите е нужно запознаване с react-router-dom, както и библиотека за управление на състоянието на компонентите като Redux.

В Angular входната бариера е много по-висока. Основните неща са: директиви, модули, декоратори, компоненти, услуги, инжектиране на зависимости, темплейти и др. След това е нужно запознаване със зони, AoT компилации, Rx.js и др.

Като заключение може да се добави, че React притежава по-гладка крива на обучение в сравнение с Angular.

JSX.

За разлика от Angular, React използва JSX[4] в темплейтите. Той е предпроцесор, който добавя XML синтактично разширение към JavaScript. JSX прави React по-гъвкав за разработчиците като комбинира темплейти на потребителския интерфейс, JavaScript логика и подобен на XML език. На Фигура 2 е декларирана променлива name и е използвана в JSX, обвита във фигурни скоби.

```
const name = 'Jordan';
const element = <h1>Hello, {name}</h1>;
```

Фигура 2. JSX синтаксис

Angular предлага свои разширения на HTML, като “ng-if”, “ng-for”, които изискват изучаване на специфичен синтаксис.

Мобилни решения.

Ionic е работна рамка за разработване на хибридни мобилни приложения. Тя използва контейнер Cordova, който е включен в Angular. Ionic предоставя компонентна библиотека на потребителски интерфейси, която е лесна за настройка и разработване на хибридни мобилни приложения. Полученото приложение за мобилно устройство, обаче е просто уеб приложение поставено в Cordova контейнер. Поради това е възможно да бъде по-бавно.

React Native, от друга страна, е платформа, разработена от Facebook за създаване на истински native мобилни приложения, използващи React.

[4] JavaScript XML, W3Schools, 2020 < https://www.w3schools.com/react/react_jsx.asp> (15.02.2020)

Синтаксисът е малко по-различен, но има много повече прилики, отколкото разлики. React Native създава наистина native потребителски интерфейс. Също така позволява да бъдат създадени свои собствени компоненти и да бъдат свързани с native код, написан в Objective-C, Java или Swift.

Тестване.

Jest се използва от Facebook за тестване на кода на React. Той е включен във всеки проект на React и изисква нулева конфигурация за да се използва. Понякога Jest се използва в комбинация с Enzyme (програма за тестване на JavaScript, използвана в Airbnb).

Jasmine е тестова работна рамка, която се използва в Angular. Тя е по-сложна от Jest. Често резултата е твърде раздут и труден за разчитане.

Интеграция.

Главен недостатък на React е интеграцията. React не е MVC работна рамка и както бе споменато по-рано представлява единствено "V" слоя от MVC модела. За това се използват инструменти за интегриране на React компонентите в традиционна MVC работна рамка. За да бъде това възможно са необходими задълбочени познания. При Angular не съществува този проблем, защото е пълнофункционална MVC работна рамка.

Популярност.

В момента React има 144000 звезди в Github срещу 57800 на Angular. Сътрудниците в Github на React са 1362 срещу 1089 на Angular.

II. Изисквания, инсталация и структура на приложението

За да се изучава React са необходими познания по HTML, CSS и JavaScript (ES5, ES6). Всеки разработчик се нуждае от инструменти за да създава приложения. Задължителни са интегрирана среда за разработка (IDE) и terminal (Command Prompt / PowerShell в Windows). Първият инструмент се използва за да се организира и пише програмен код, а втория – за да се изпълняват команди, като стартиране на приложението, тестове или инсталиране на други библиотеки.

Не на последно място е необходима инсталация на Node.js[5], който съдържа в себе си NPM[6] и Yarn[7]. Това са пакетни мениджъри, чрез които могат да бъдат инсталирани външни пакети. Последните биват набор от функционалности, библиотеки или цели работни рамки и са така наречените „зависимости“ на бъдещото приложение. Могат да бъдат инсталирани глобално в папката на node пакетите или локално в папката на самия проект.

Съществуват няколко подхода при създаването на React приложение:

1. Чрез използване на CDN[8], като трябва да бъдат добавени два скрипта в <head> таговете на index.html. Съществуват различни скриптове за разработка и за използване в завършеното приложение.

```
<script src="https://unpkg.com/react@16/umd/react.development.js"></script>
<script src="https://unpkg.com/react-dom@16/umd/react-dom.development.js"></script>
```

```
<script src="https://unpkg.com/react@16/umd/react.production.min.js"></script>
<script src="https://unpkg.com/react-dom@16/umd/react-dom.production.min.js"></script>
```

2. С помощта на terminal / Command Prompt и по-рано инсталирания NPM:

```
npm install –save react react-dom
```

Този подход е често използван, когато трябва да бъде добавен React към вече съществуващо приложение. За съжаление това не е всичко и са необходими допълнителни настройки и конфигуриране на Babel[9] компилатор, което често се явява непосилна задача за начинаещите React разработчици.

3. Поради сложността на втория подход, през 2016 г. Facebook представя create-react-app[10] като решение с нулева конфигурация.

Според допитване[11] в Twitter на Дан Абрамов, програмист в екипа на React, 96% от потребителите на create-react-app биха го препоръчали на начинаещи, а 84,6% на напреднали разработчици.

[5] Node.js Official Webpage, 2020 <https://nodejs.org> (15.02.2020)
[6] Node Package Manager, Wikipedia, 2020 <https://en.wikipedia.org/wiki/Npm_(software)> (15.02.2020)
[7] Yarn Package Manager Official Webpage, 2020 <https://yarnpkg.com> (15.02.2020)
[8] Content Delivery Network, Wikipedia, 2020 <https://en.wikipedia.org/wiki/Content_delivery_network> (15.02.2020)
[9] Babel Compilator Official Webpage, 2020 <http://babeljs.io> (15.02.2020)
[10] create-react-app Repository, Github, 2020 <https://github.com/facebook/create-react-app> (15.02.2020)
[11] Dan Abramov Twitter Profile, 2020 <https://twitter.com/dan_abramov/status/806985854099062785> (15.02.2020)

Командата, с която create-react-app може да бъде инсталиран в глобалните node пакети е:

```
npm install -g create-react-app
```

Проверка дали create-react-app е инсталиран успешно се извършва с командата:

```
create-react-app --version
```

За да бъде създадено приложение с име “portfolio” е нужно да бъде изпълнена командата:

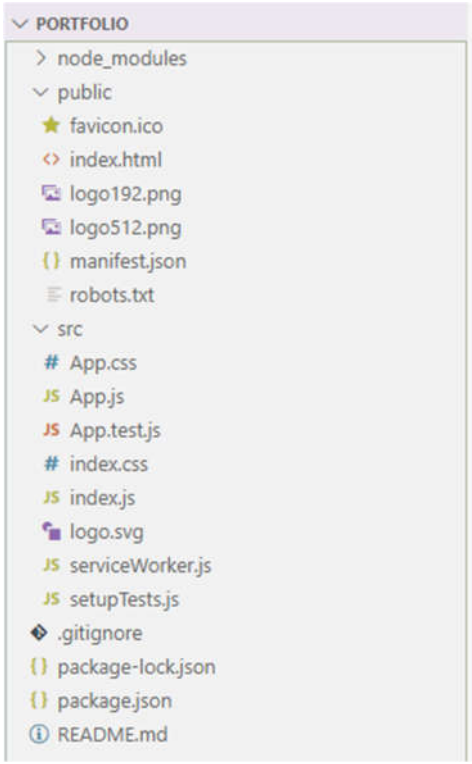
```
create-react-app portfolio
```

Следващата команда, която ни отвежда в главната директория на приложението е:

```
cd portfolio
```

За нуждите на настоящия реферат и приложение ще бъде използван популярния за JavaScript и React, редактор на програмен код Visual Studio Code[12].

След отваряне на папката на приложението в редактора се разкрива структурата на приложението. (Фигура 3)



Фигура 3. Структура на приложение създадено с create-react-app

[12] Visual Studio Code Official Webpage, 2020 < https://code.visualstudio.com> (15.02.2020)

Всичко, което е необходимо за начало се намира в папката src. Главният фокус е върху src/App.js. Този файл се използва за имплементация на React приложението. Допълнително могат да бъдат открити src/App.test.js за тестове и src/index.js, като входна точка в света на React, src/index.css и src/App.css за стилизиране на приложението и компонентите му. Папката node_modules съдържа node пакетите, а в package.json са описани всички зависимости, които са инсталирани и нужни за нормалното функциониране на приложението.

Приложенията създадени с create-react-app имат няколко вградени npm скрипта:

1. **`npm start`** за стартиране на приложението в http://localhost:3000.
2. **`npm test`** за тестове.
3. **`npm run build`** за изграждане на приложението, когато е готово за употреба.

III. Разработка на уеб приложение с React

Преди всичко е нужно да бъде разгледано съдържанието на src/index.js на Фигура 4 и public/index.html на Фигура 5.

```
import React from 'react';
import ReactDOM from 'react-dom';
import './index.css';
import App from './App';
import * as serviceWorker from './serviceWorker';
//Next row renders App component into <div id="root"></div>
ReactDOM.render(<App />, document.getElementById('root'));
serviceWorker.unregister();
```

Фигура 4. Съдържание на src/index.js

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <title>Portfolio</title>
  </head>
  <body>
    <div id="root"><!--App content goes here --></div>
  </body>
</html>
```

Фигура 5. Съдържание на public/index.html

В src/index.js се импортират библиотеката react, react-dom, css стилове от index.css и главния App компонент. Импортира се също serviceWorker. Той е скрипт изпълняван от браузъра във фонов режим и няма да бъде разглеждан в настоящия реферат. В index.js се изпълнява код, който рендира главния App компонент в блоков елемент с идентификатор “root”, намиращ се в public/index.html.

Тъй като React създава SPA приложения, задължително е да се използва рутер за да имат отделните страници свои собствени url адреси. За целта е нужно да се инсталира пакета react-router-dom:

```
npm install react-router-dom --save
```

Това ще бъде и първият компонент в приложението. В папката src трябва да бъде създадена нова папка с име components, а в нея нов файл с наименование Router.js. В него се импортират всички страници, които от своя страна са също компоненти и им се описват url адресите. Съдържанието на Router.js е изобразено на Фигура 6.

```
import React from 'react';
import { Switch, Route} from 'react-router-dom';
import Landing from './Landing';
import About from './About';
import Projects from './Projects';
import Contact from './Contact';
import NotFound from './NotFound';

const Router = () => (
    <Switch>
        <Route exact path="/" component={Landing} />
        <Route path="/about" component={About} />
        <Route path="/projects" component={Projects} />
        <Route path="/contact" component={Contact} />
        <Route component={NotFound} />
    </Switch>
)

export default Router;
```

Фигура 6. Съдържание на src/components/Router.js

За да работи рутера коректно е нужно да се импортира в src/index.js, както и да се извършат корекции в кода намиращ се в render() метода. (Фигура 7)

```
import React from 'react';
import ReactDOM from 'react-dom';
import App from './App';
import * as serviceWorker from './serviceWorker';
import { BrowserRouter as Router } from 'react-router-dom';
ReactDOM.render(
    <Router>
        <App />
    </Router>
    , document.getElementById('root')
);
serviceWorker.unregister();
```

Фигура 7. Импортиране на рутер в src/index.js

При следващата стъпка трябва да бъдат поставени главните компоненти в return() метода на функцията App, находяща се в src/App.js, като последователно се добавят и импортират компонентите Navbar, Router и Footer, изграждащи гръбнака на приложението. (Фигура 8)

```
import React from 'react';
import './App.css';
import Navbar from './components/Navbar';
import Router from './components/Router';
import Footer from './components/Footer';

function App() {
  return (
    <React.Fragment>
      <Navbar />
      <Router />
      <Footer />
    </React.Fragment>
  );
}

export default App;
```

Фигура 8. Импортиране на навигационна лента, рутер и футър в src/App.js

Създаване на компонент src/components/Navbar.js, представляващ навигационната лента на приложението. (Фигура 9)

```
import React, { Component } from 'react';
import { Link } from 'react-router-dom';

class Navbar extends Component{
    render(){
        return(
            <React.Fragment>
                <nav className="navbar">
                    <Link className="navbar-brand" to="/">
                        Portfolio
                    </Link>
                    <ul className="navbar-nav">
                        <li className="nav-item">
                            <Link className="nav-link" to="/">
                                Home
                            </Link>
                        </li>
                        <li className="nav-item">
                            <Link className="nav-link" to="/about">
                                About
                            </Link>
                        </li>
                        <li className="nav-item">
                            <Link className="nav-link" to="/projects">
                                Projects
                            </Link>
                        </li>
                        <li className="nav-item">
                            <Link className="nav-link" to="/contact">
                                Contact
                            </Link>
                        </li>
                    </ul>
                </nav>
            </React.Fragment>
        );
    }
}

export default Navbar;
```

Фигура 9. Навигационна лента - src/components/Navbar.js

Както се вижда, кодът в return() метода силно наподобява на HTML, но има някои дребни различия. Поради това, че думата **class** е резервирана в JavaScript, няма как да бъде използвана и е заместена с **className**. За да се използва рутера, традиционните анкор тагове са заместени с Link тагове:

```
<a class="nav-link" href="/about">About</a>
<Link className="nav-link" to="/about">About</Link>
```

Създаване на компонент src/components/Footer.js (Фигура 10)

```
import React, { Component } from 'react';

class Footer extends Component{
    getYear() {
        return new Date().getFullYear();
    }

    render(){
        return(
            <React.Fragment>
            <hr />
            <footer>
                <ul>
                    <li>
                        <a href="https://twitter.com/profile">Twitter</a>
                    </li>
                    <li>
                        <a href="https://www.facebook.com/profile">Facebook</a>
                    </li>
                    <li>
                        <a href="https://www.linkedin.com/in/profile">LinkedIn</a>
                    </li>
                    <li>
                        <a href="https://github.com/profile">GitHub</a>
                    </li>
                </ul>
                <p>Copyright &copy; Portfolio {this.getYear()}</p>
                <p>Created by Jordan Valkov</p>
            </footer>
            </React.Fragment>
        );
    }
}

export default Footer;
```

Фигура 10. Футър - src/components/Footer.js

Компонентите Landing.js, About.js, Portfolio.js и Contact.js са страници с различно съдържание, но много сходен JSX код. Поради това, в настоящия реферат ще бъде разгледана една от тях, а именно src/components/About.js. Съдържанието на страницата е изобразено на Фигура 11.

```
import React, { Component } from 'react';

class About extends Component{
    render(){
        return(
            <React.Fragment>
                <header className="header bg-about">
                    <div className="overlay"></div>
                    <div className="page-heading">
                        <h1>About Me</h1>
                        <span className="subheading">This is what I do.</span>
                    </div>
                </header>

                <div className="container">
                    <div className="row">
                        <div className="col">
                            <p>Lorem ipsum dolor sit amet!</p>
                            <p>Consectetur quia natus ducimus</p>
                            <p>Totam invent asper repellendus!</p>
                        </div>
                    </div>
                </div>
            </React.Fragment>
        );
    }
}

export default About;
```

Фигура 11. Страница About - src/components/About.js

Последният компонент в приложението се използва, когато трябва да бъде върната грешка 404 за това, че не съответства такава страница на въведения в браузъра url адрес. Това е компонентът src/components/NotFound.js. (Фигура 12)

```
import React, { Component } from 'react';
class NotFound extends Component{
    render(){
        return(
            <React.Fragment>
                <header className="header bg-notfound">
                    <div className="overlay"></div>
                    <div className="page-heading">
                        <h1>404</h1>
                        <span className="subheading">Page Not Found</span>
                    </div>
                </header>
            </React.Fragment>
        );
    }
}
export default NotFound;
```

Фигура 12. Страница 404 Page Not Found - src/components/NotFound.js

Заключение

В заключение, могат да се направят някои изводи по темата, а именно, че React позволява да бъдат разработени съвременни уеб приложения, които могат да се използват, както като статични страници, така и да бъдат интегрирани в MVC работни рамки. React е част от популярния JavaScript фулстак пакет MERN, зад който стоят MongoDB, Express.js, React и Node.js. MERN позволява цялостната JavaScript разработка, от базата данни до клиентската и сървърната страна на уеб приложенията.

Решаващите фактори за това React да бъде предпочитана библиотека са нейните предимства. Тя притежава бързина на приложенията, постигната чрез виртуален DOM. Кривата на обучение на React е значително по-гладка, отколкото на Angular. Документацията е подробна и изчерпателна. React има огромна общност от разработчици и много източници на информация, като различни уебсайтове, форуми, блогове и литература. Инсталацията е лесна и с нулева конфигурация. Извършва се лесно миграция на стари React проекти към актуалната версия на библиотеката. Компонентно-базираната архитектура и опростения JSX синтаксис позволяват добра организация и преизползваемост на кода. Това от своя страна намалява времето за разработка, тестове и отстраняване на грешки, а от там и на разходите за разработка.

Използвана литература

1. Banks A. & Porcello E., Learning React. O'Reilly, 2018.
2. Thomas M. T., React in action. Manning, 2018.
3. Angular Official Webpage: <https://angular.io> (15.02.2020)
4. Babel Compilator Official Webpage, 2020 <http://babeljs.io> (15.02.2020)
5. Content Delivery Network, Wikipedia, 2020 <https://en.wikipedia.org/wiki/Content_delivery_network> (15.02.2020)
6. create-react-app Repository, Github, 2020 <https://github.com/facebook/create-react-app> (15.02.2020)
7. Dan Abramov Twitter Profile, 2020 <https://twitter.com/dan_abramov/status/806985854099062785> (15.02.2020)
8. Document Object Model, Wikipedia, 2020 <https://en.wikipedia.org/wiki/Document_Object_Model> (15.02.2020)
9. JavaScript XML, W3Schools, 2020 < https://www.w3schools.com/react/react_jsx.asp> (15.02.2020)
10. Model View Controller, Wikipedia, 2020 <https://en.wikipedia.org/wiki/Model-view-controller> (15.02.2020)
11. Node.js Official Webpage, 2020 <https://nodejs.org> (15.02.2020)
12. Node Package Manager, Wikipedia, 2020 <https://en.wikipedia.org/wiki/Npm_(software)> (15.02.2020)
13. React Library Repository, Github, 2020 <https://github.com/facebook/react> (15.02.2020)
14. React Official Webpage: <http://reactjs.org> (15.02.2020)
15. Single Page Application, Wikipedia, 2020 <https://en.wikipedia.org/wiki/Single-page_application> (15.02.2020)
16. Visual Studio Code Official Webpage, 2020 < https://code.visualstudio.com> (15.02.2020)
17. Vue JS Official Webpage: <https://vuejs.org> (15.02.2020)
18. Yarn Package Manager Official Webpage, 2020 <https://yarnpkg.com> (15.02.2020)