ORES
Научно-практический электронный журнал
Оригинальные исследования (ОРИС), *ISSN 2222-0402*
www.ores.su
original-research.ru

УДК 004.4

ПРЕИМУЩЕСТВА ИСПОЛЬЗОВАНИЯ РЕАКТИВНОГО ФРЕЙМВОРКА ПРИ НАПИСАНИИ ДИНАМИЧЕСКИХ ОДНОСТРАНИЧНАХ ВЕБ-ПРИЛОЖЕНИЙ

Онуфриева Татьяна Александровна,
кандидат технических наук, доцент Калужского филиала ФГБОУ ВПО «Московский государственный технический университет им. Н.Э. Баумана» (национальный исследовательский университет)», 248000, Россия, г. Калуга, ул. Баженова, д. 2. onufrievata@mail.ru.

Голубев Андрей Сергеевич
студент 2-го курса магистратуры группы ИУК2-31М Калужского филиала ФГБОУ ВПО «Московский государственный технический университет им. Н.Э. Баумана» (национальный исследовательский университет) », 248000, Россия, г. Калуга, ул. Баженова, д. 2. golandroser@gmail.com

Федоров Виктор Олегович
Кандидат технических наук, доцент Калужского филиала ФГБОУ ВПО «Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)», 248000, Россия, г. Калуга, ул. Баженова, д. 2. fedorov_vo@bmstu.ru

Аннотация

В научной статье раскрывается современный компонентно-реактивный подход к написанию динамических SPA приложений, описываются основы организации системы реактивности в фреймворке Vue.js.

На примере конкретного приложения (Программный модуль системы автоматизации ведомственной АЗС), клиентская составляющая которого написана с использованием фреймворка Vue, проанализированы преимущества использования данного фреймворка и его системы реактивности.

Результаты статьи могут быть использованы при создании различного рода одностраничных веб-приложений с использованием фреймворка Vue.js, а также помогут разобраться в основах его реактивности.

Ключевые слова: SPA, парадигма реактивного программирования, Vue.JS, привязка данных, система автоматизации ведомственной АЗС.

ADVANTAGES OF USING A REACTIVE FRAMEWORK WHEN WRITING DYNAMIC SINGLE-PAGE WEB APPLICATIONS

Tatiana A. Onufrieva
Candidate of Technical Sciences, Docent of Bauman Moscow State Technical University (Kaluga Branch), 248000, Russian Federation, Kaluga, Bazhenova st., 2.
onufrievata@mail.ru

Andrey S. Golubev
2nd year master's student of group IUK2-31M of Bauman Moscow State Technical University (Kaluga Branch), 248000, Russian Federation, Kaluga, Bazhenova st., 2.
golandroser@gmail.com

Viktor O. Fedorov
Candidate of Technical Sciences, Docent of Bauman Moscow State Technical University (Kaluga Branch), 248000, Russian Federation, Kaluga, Bazhenova st., 2. fedorov_vo@bmstu.ru

ABSTRACT

The scientific article reveals a modern component-reactive approach to writing dynamic SPA applications and describes the basics of organizing a reactivity system in the Vue.js framework.

Using the example of a specific application (software module for a departmental gas station automation system), the client component of which is written using the Vue framework, the advantages of using this framework and its reactivity system are analyzed.

The results of the article can be used to create various kinds of single-page web applications using the Vue.js framework, and will also help you understand the basics of its reactivity.

Keywords: SPA, reactive programming paradigm, Vue.JS, data binding, departmental gas station automation system.

ВВЕДЕНИЕ

В последние годы в связи с бурным ростом технологий и техники, работающей на различной аппаратной и программной платформе, удешевлением доступа к интернету достаточно часто для обеспечения кроссплатформенности при разработке приложений используются веб-технологии.

Одна из главных тенденций при написании многофункционального веб-приложения – использование одностраничной архитектуры его построения (SPA – single page application). При реализации таких приложений часто используется реактивная парадигма программирования.

Преимущества и недостатки SPA-приложений.

Одностраничность в буквальном смысле обуславливается наличием в приложении всего лишь одной html-страницы, а подгрузка и переиспользование элементов обеспечивается динамически с использованием механизмов языка JavaScript. Преимущество использование SPA – подхода заключается в следующем:

- Повышение общей скорости работы приложения, связанное с тем, что основные данные, необходимые для работы приложения, подгружаются при его запуске и кэшируются для дальнейшей работы в автономном режиме [1].

- Снижение нагрузки на сервер благодаря тому, что логика отображения элементов переносится на браузер клиентской стороны. В связи с этим также упрощается внесение изменений в код логики приложения.
- Адаптация SPA - приложений к устройствам разных платформ проще, потому что не требуется каждый раз вносить изменения в клиентскую составляющую приложения.

Несмотря на существенные преимущества, имеются у такого подхода и недостатки, которые не позволяют использовать его повсеместно:

- Отсутствие встроенного механизма SEO-оптимизации, то есть механизма ранжирования (предложения страниц). Поисковику достаточно сложно предлагать различные страницы сайта, так как физически она только одна.
- Приложение не будет работать в случае, если у пользователя отключена возможность работы с JavaScript, так как основные механизмы для воспроизведения такого сайта становятся недоступными.

Условный цикл работы одностраничного веб-приложения изображён на рисунке 1. Клиентская часть на начальном этапе инициализирует запрос, в ответ на который сервер выдаёт единичный html-файл. В случае необходимости обновления данных на странице, клиентская часть инициализирует специализированный Ajax-запрос, в ответ на который сервер возвращает данные, например, в формате Json [2].

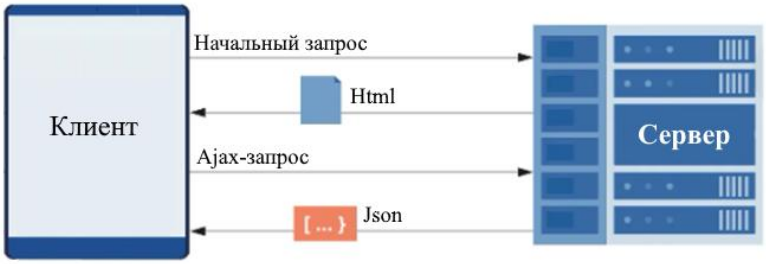

Рисунок 1. Цикл работы SPA-приложения

Способы создания SPA-приложений, использование реактивной парадигмы

Создание одностраничных web - приложений невозможно без JavaScript, так как именно этот язык программирования служит фундаментом для SPA, обработку и отрисовку элементов страницы, осуществляет виртуальные переходы между «страницами» сайта, расположенными в пределах одного html-документа. При этом имеется два варианта разработки:

- Использование чистого языка JavaScript;
- Использование реактивного фреймворка.

Первый вариант требует очень хорошего знания всех тонкостей языка JavaScript, что в некоторых случаях может очень замедлить процесс разработки. Однако невозможно не отметить, что программы на чистом языке будут работать эффективнее и занимать меньше места.

Фреймворк же является надстройкой над языком, использующим абстракцию, упрощающую и ускоряющую программирование, а, следовательно, и создание нового приложения.

В основном все фреймворки, позволяющие создавать одностраничные веб – приложения, основаны на реактивной парадигме программирования.

В чём заключается реактивность фреймворка и каким образом применение этой реактивности сказывается на работе приложения?

Реактивная парадигма в программировании рассматривает любые данные как асинхронный поток, реактивность определяет способ автоматически обновлять систему в зависимости от изменения потока данных [3].

Реактивное программирование реализует поведенческий шаблон «Наблюдатель», представленный на рисунке 2 в виде UML-диаграммы классов: у нас есть конкретный объект - поток данных (ObservableObs), подписываясь на поток (метод Add), мы будем получать о возникших в нём изменениях (метод Notify), а в зависимости от этого каким-либо образом обновлять систему (метод HandleEvent) [4].

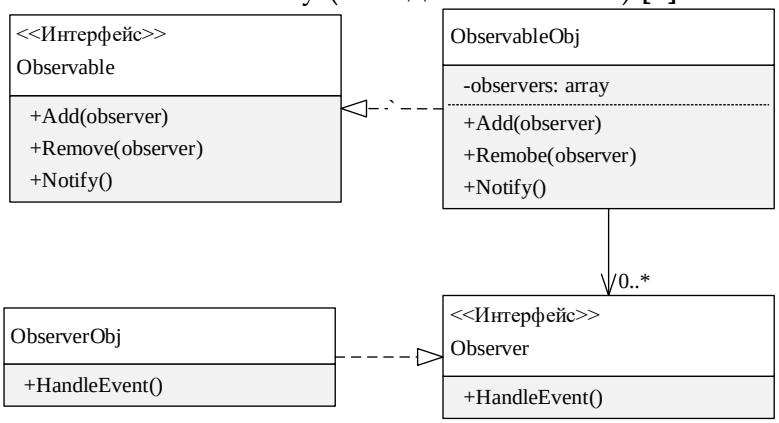

Рисунок 2. Шаблон поведения «Наблюдатель»

Основы реактивности фреймворка JavaScript – фреймворка Vue.js

Рассмотрим, каким образом данный шаблон реализуется в популярном фреймворке, используемом при построении многофункциональных одностраничных веб-приложения – Vue.js.

Реактивность в фреймворке Vue второй версии реализуется в соответствии со схемой, приведённой на рисунке 3.

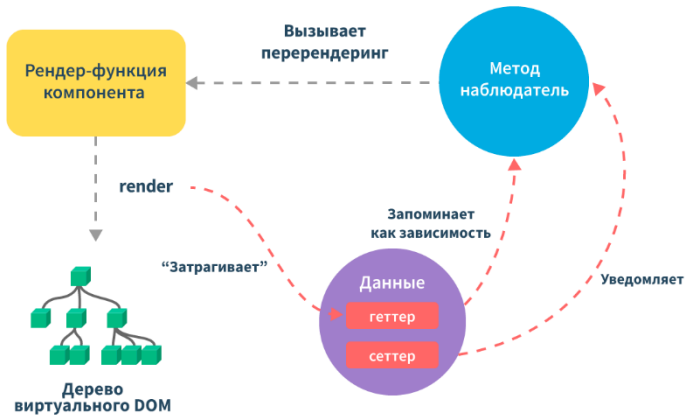

Рисунок 3. Схема обеспечения реактивности Vue 2

Взаимодействие с реальным деревом объектов на странице (DOM) осуществляется не напрямую. DOM API имеет ограничения: его методы дороги и ресурсоёмки. Они показывают приемлемую производительность только для небольшого количества изменений.

С этой целью Vue использует виртуальный DOM. В некотором приближении виртуальный DOM — это набор JS – объектов (узлов) (VNode). Этапы рендеринга и перерендеринга Vue-компонента представлены на рисунке 4.

Для получения набора виртуальных узлов Vue.js использует специализированную render-функцию. Далее виртуальные JS - объекты посредством библиотеки vdom переводятся в реальную браузерную модель DOM.

При обновлении элементов Vue анализирует изменения виртуальной модели, вызывая перерендеринг не всего документа, а лишь значимых узлов.

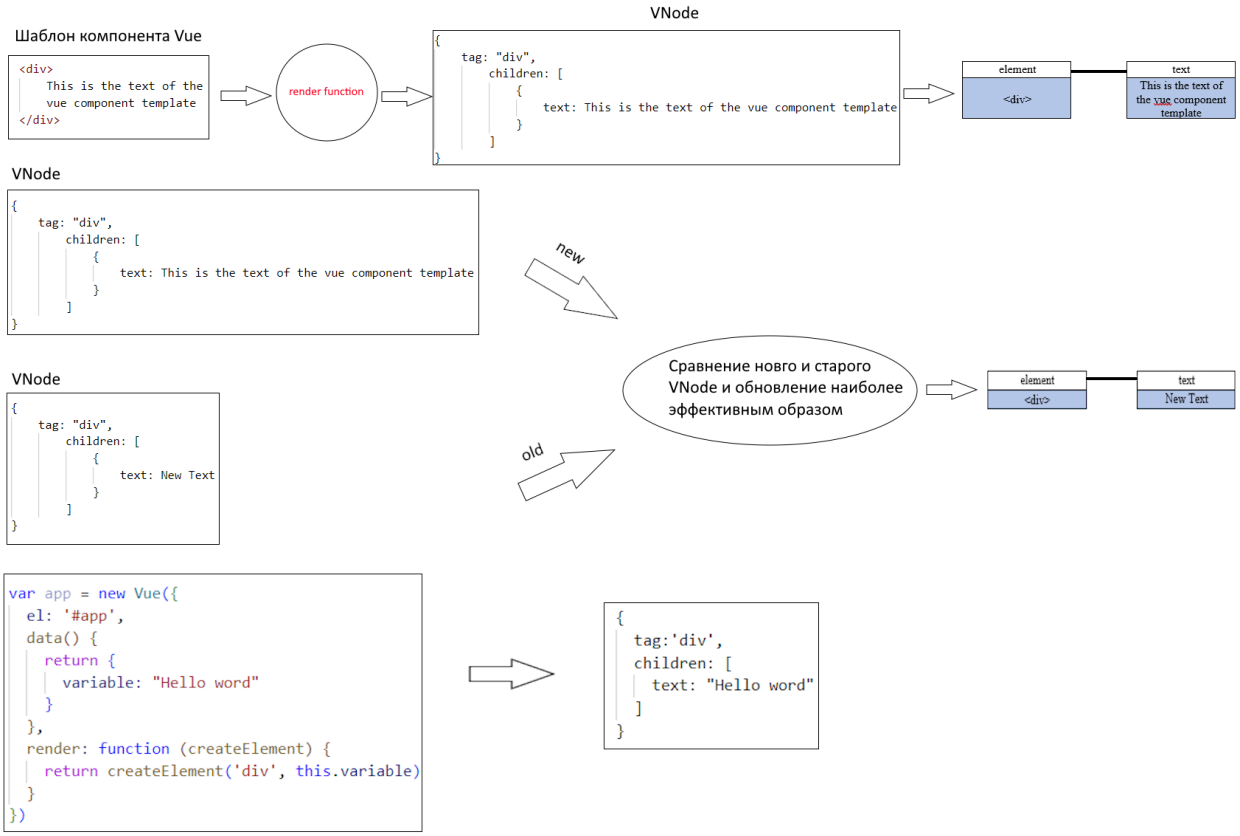

Рисунок 4. Этапы рендеренга и перерендерингра Vue-компонента

Определив базовые принципы обновления элементов на странице, рассмотрим модель реактивного обновления данных.

Для наглядности и понимания реализуем данную схему на упрощённом примере.

Для реализации наблюдения Vue использует Object.defineProperty добавляя getter и setter каждому свойству передаваемого объекта, таким является объект data внутри отдельного компонента Vue [5].

```
Object.defineProperty(obj, key, {
  get() {
     return value;
….
  },
  set(newValue) {
  value = newValue;
….
  }
```

Одного лишь геттера и сеттера недостаточно для того, чтобы система реактивности работала. Для наблюдения также необходим класс-зависимости, который реализует методы интерфейса «Observable», показанного на рисунке 2. В Vue таким объектом является класс Dep. Он реализует 3 основных метода интерфейса:

```
let target=null

class Dep {
   constructor() {
      this.subs = [];
   }
   depend() {
      if (target && !this.subscribers.includes(tagrget) {
         this.subs.push(target);
      }
   }
   notify() {
      this.jobs.forEach(run => {
         run();
      });
   removeSub(){
this.subs[this.subs.indexOf(sub)] = null
}
   }
}
```

Такой класс создаётся для каждого реактивного свойства, метод depen() помещается в getter Object.defineProperty, notify() помещается в setter. subs – массив - хранилище для «подписанных» функций-задач.

```
Object.keys(data).forEach(key => {
   let val = data[key]
   const dep = new dep()
   Object.defineProperty(data, key, {
      get() {
         dep.depend()
         return val;
      },
      set(newValue) {
         val = newValue;
         dep.notify()
      }

   })
})
```

Третий важный элемент – метод – наблюдатель, которому в качестве аргумента передаётся функция анонимная функция.

```
function watcher(Myfunc) {
   target = Myfunc
   target()
   target = null
}
```

Допустим функция, передаваемая в watcher представляет из себя следующее:

Watcher(()=>{elem3=react_elem1+react_elem2}),

где react_elem1, react_elem2 – реактивные переменные, находящиеся в объекте data.

Когда мы обращаемся к реактивным элементам react_elem1 и react_elem2, то, соответственно, активируются их геттеры, а в классы Dep, связанные с этими элементами, добавляется новый подписчик-наша анонимная функция.

Теперь, изменив значение любого из реактивных элементов, мы вызовем метод notify(), тем самым запустив функцию подписчика – нашу анонимную функцию и все значения будут пересчитаны.

В нашем упрощённом случае watcher всего лишь один, в vue таких watcher'ов может быть большое количество. Например, Vue имеет watcher, который отслеживает перерендеринг страницы (render watcher). Общий алгоритм начинается с вызова функции рендеринга компонента во время его монтирования:

- Функцию рендеринга компонента можно рассматривать как анонимную функцию, передаваемую наблюдателю внутри нашей собственной системы. На момент вызова этой функции, данные объекта становятся реактивными.
- При попытке доступа к переменной вызывается функция depend() для объекта Dep, связанного с этим свойством.
- Далее, когда свойство будет изменено, вызывается сеттер. Сеттер не только устанавливает новое значение свойства, вызывает функию notify(). Вызов notify() в свою очередь приводит к вызову всех зарегистрированных обратных вызовов, хранящихся во внутреннем массиве объекта Dep, включая функцию рендеринга [5, 6].

Таким образом, на простом примере были рассмотрены главные принципы организации рективности в фреймворке Vue.

Стоит отметить, что приведённый выше код – это лишь основополагающий принцип, сама система реактивности Vue организована гораздо сложнее, обрабатываются исключительные ситуации, имеется возможность асинхронной очереди реактивных команд, чтоб ограничить ненужные многократные обновления страницы и составляющих её элементов.

Всё описываемое скрыто под оболочкой. Программисту, использующему Vue, не приходится прописывать классы отслеживания, зависимости, функции уведомления.

Анализ преимуществ использования реактивного фреймворка Vue.js при реализации клиентской части приложения на примере «Программного модуля системы автоматизации ведомственной АЗС».

Рассмотрим применение системы реактивности в конкретном проекте: «Система автоматизации ведомственных АЗС», программный модуль которой предназначен для сбора информации и управлением функционированием терминалов автоматизации ведомственных ТРК.

Интерфейс прогрмаммы написан полностью на Vue JS с использованием архитектуры одностраничного построения web-приложения.

В качестве примера рассмотрим работу фреймворка по выводу карточек терминалов ТРК (Рисунок 5, Рисунок 6), а также возможности реактивного фреймворка по двухсторонней привязке данных (Рисунок 7, Рисунок 8).

Простота рендеринга массивов больших массивов данных обеспечивается благодаря специальной директиве (v-for), работа которой очень напоминает цикл «foreach» в языках программирования [7]. И теперь вместо того, чтоб для каждого терминала прописывать интерфейсную составляющую, делается это один раз кодом разметки, повторение кода реализуется механизмами Vue.

Второй ключевой момент, непосредственно связанный с реактивностью – это возможность условной отрисовки: в примере ниже для активного терминала (v-else) отрисуется зелёный кружок статуса, а для неактивного (v-if=!item.online) – серый.

Поле item.online – реактивно, его значение отслеживается, а изменения приводят к перерисовке элемента на странице). Вместо условного рендеринга Vue позволяет показывать и скрывать элементы (директива v-show и реактивное свойство item.isConfigurationChanged) (Рисунок 5).

```
<div class="col-12 col-lg-3 col-md-4 col-sm-6 py-1 px-2 "
    v-for="item in AllTerminalsFiltered" :key="item.guid">
    <b-card no-body class="terminal border">
       <b-icon v-if="!item.online" icon="circle-fill"
         style="color: #4F4F4F;" font-scale="2"
         :title="OffTime(item.lastConnectionDateTime)">
       </b-icon>
       <b-icon v-else icon="circle-fill" style="color: #15891B;"font-scale="2"
       title="Активен">
       </b-icon>
      <span class="m-2">{{ item.name }}</span>
            ...
              <div class="first_info d-flex"
                  style="min-height: 21px; margin-left: 38px;">
                  <b-iconstack v-show="item.isConfigurationChanged"
                      title="Конфигурация на изменении" font-scale="1.5">
                      <b-icon icon="border-style" stacked variant="warning "
                          scale="0.6"></b-icon>
                      <b-icon stacked icon="circle" variant="warning ">
                          </b-icon>
                  </b-iconstack
                   </div>
              ...
</template>
        <b-card-body class="d-flex flex-column card-body-terminal"
            ...
        </b-card-body>
    </b-card>
</div>
```

Рисунок 5. Вырезка кода для формирования карточек терминалов управления функционированием ТРК

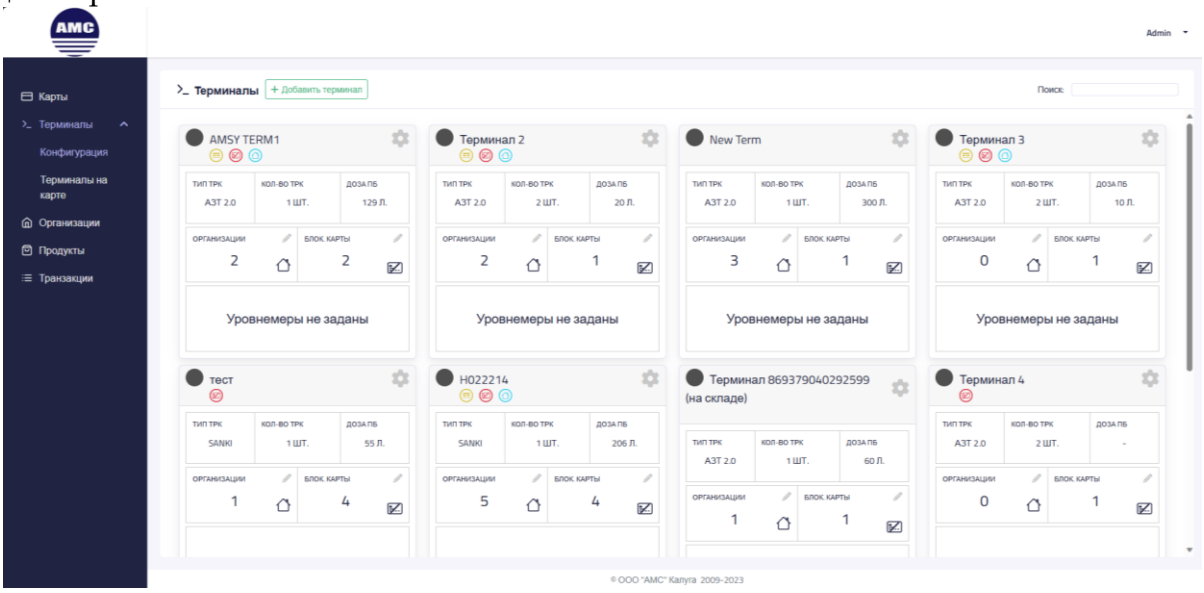

Рисунок 6. Карточки терминалов управления функционированием ТРК

Двухсторонняя привязка реактивных данных осуществляется во Vue с помощью директивы (v-model), директива используется для связывания полей ввода и реактивных переменных [8].

При изменении полей ввода будет изменено значение реактивной переменной, при изменении реактивной переменной будет изменено значение поля ввода – этим определяется двусторонность взаимосвязи. Например, таким образом связано поле hourOffset в рассматриваемом коде (Рисунок 7).

Такой подход к данным позволяет очень эффективно и быстро изменять их, а благодаря механизму реактивности отображение интерфейса будет эффективно перерисован.

```
terminalConfiguration: {
    hourOffset: null,
    fullTankVolume: 50,
    gaugeAddr: 1,
    gaugeChannels: [],
    gaugeType: 0,
    pumps_col: 1,
    gauge_col: 0,
    fuels: [],
    pumps: [{
        addr: 1,
        type: 0,
        nozzles_col: 1,
        nozzles: [{
            adr: 1,
            productID: null
        },
        ]
    },
    ],
},
```

```
<legend>Общие параметры</legend>
<div class="row mb-2">
    <div class="col-sm-4 col-12 align-self-center"><label
            class="form-label font-weight-bold">Наименование</label></div>
    <div class="col-sm-8 col-12 align-self-center"><input type="text"
            placeholder="Наименование терминала" :disabled="loading" v-model="name"
            class="form-control" required />
    </div>
</div>
<div class="row mb-2">
    <div class="col-sm-4 col-12 align-self-center"><label
            class="form-label font-weight-bold">IMEI</label></div>
    <div class="col-sm-8 col-12 align-self-center"><input type="text"
            @keypress="onlyNumber" :disabled="loading || edit" pattern="[0-9]{15}"
            title="15 цифр" maxlength="15" placeholder="xxxxxxxxxxxxxxx"
            v-model="imei" class="form-control" required />
    </div>
</div>
<div class="row mb-2">
    <div class="col-sm-4 col-12 align-self-center">
        <label class="form-label">Часовой пояс</label>
    </div>
    <div class="col-sm-8 col-12 align-self-center">
        <multiselect trackBy="text" valueProp="value" :searchable="true" locale="ru"
            :canClear="false" :canDeselect="false" :allowEmpty="true"
            placeholder="Выберите часовой пояс" id="time_zone"
            :close-on-select="true" label="text" required
            v-model="terminalConfiguration.hourOffset" :options="TimeZones">
            <template v-slot:nooptions><div class="d-flex w-100 align-items-center justify-content-center my-2">Нет элементов</div></template>
            <template v-slot:noresults><div class="d-flex w-100 align-items-center justify-content-center my-2"> Не найдено</div></template>
        </multiselect>
    </div>
</div>
```

Рисунок 7. Вырезка кода и данных для двухсторонне привязки в модальном окне настройки конфигурации терминала управления функционированием ТРК

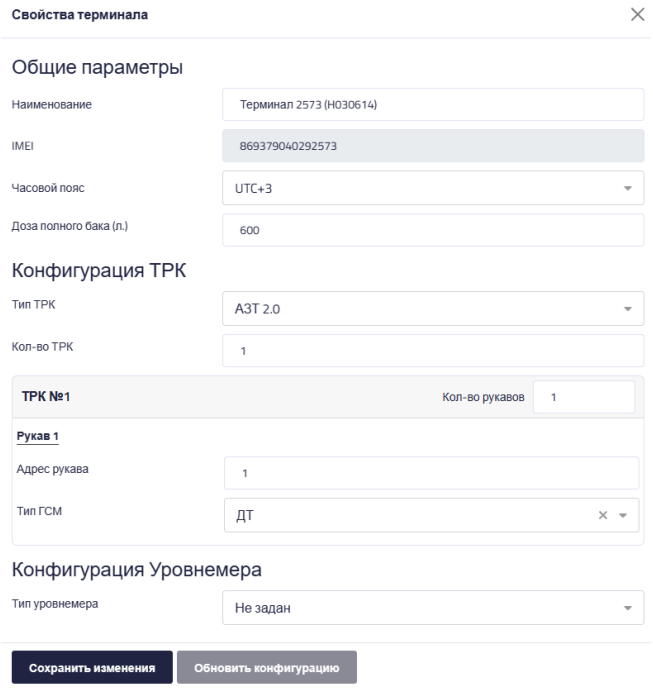

Рисунок 8. Модальное окно настройки конфигурации

Заключение

В статье были рассмотрены преимущества использования SPA-технологии при построении одностраничных веб – приложений, рассмотрена реактивная парадигма программирования и её применение в веб -программировании. Описаны основы реализации системы реактивности и обновления данных во фреймворке Vue.JS.

На конкретном примере программы, использующей в интерфейсной составляющей фреймворк Vue, были продемонстрированы преимущества использования реактивного подхода при разработке.

Описанные положения могут быть применимы при разработке различного рода многофункциональных одностраничных веб – приложений.

Список литературы:

1. Сычев, А. В. Теория и практика разработки современных клиентских веб-приложений : учебное пособие / А. В. Сычев. — 3-е изд. — Москва : Интернет-Университет Информационных Технологий (ИНТУИТ), Ай Пи Ар Медиа, 2021. — 482 с. — ISBN 978-5-4497-0943-1. — Текст : электронный // Цифровой образовательный ресурс IPR SMART : [сайт]. — URL: https://www.iprbookshop.ru/102067.html
2. Ардейчик С.М., шербаф А.И. SPA-архитектура мультифункционального веб-приложения // ВЕСЦІ БДПУ. СЕРЫЯ 3. ФІЗІКА. МАТЭМАТЫКА. ІНФАРМАТЫКА. БІЯЛОГІЯ. ГЕАГРАФІЯ, 2018, № 3(97), С. 85 – 91.
3. Кононов Н.А. Исследование причин появления и формирование определения концепции реактивности, применяемой в веб-разработке // Сборник статей XXIV Международной научно-практической конференции. Пенза, 2021, С. 59-62.
4. Романов, Е. Л. Программная инженерия: учебное пособие: Е. Л. Романов; Новосибирский государственный технический университет. – Новосибирск: Новосибирский государственный технический университет, 2017. – 395 с.: табл., схем., ил. – (Учебники НГТУ). – Режим доступа: по подписке. – URL: https://biblioclub.ru/index.php?page=book&id=573945
5. Подробно о реактивности [Электронный ресурс]. URL: https://ru.vuejs.org/v2/guide/reactivity.html
6. Хэнчетт Эрик, Листуон Бенджамин Vue.js в действии.-СПБ.: Санкт-Петербург, 2019.-304 с.: ил.-(Серия «Библиотека программиста»). ISBN 978-5-4461-1098-8.: [Электронный ресурс]. URL: http://web.spt42.ru/files/vue/Vue_js_v_deystvii_-_E_Khenchett_2019.pdf
7. Сафронов, А. И. Проектирование типовой информационной системы управления с использованием технологии web-программирования на базе фреймворка Vue.js: учебно-методическое пособие / А. И. Сафронов, А. И. Котова. — Москва : РУТ (МИИТ), 2019. — 97 с. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/175692
8. Харитонов Ю.Е., Павлов М.В. Основные концепции и возможности фреймворка Vue.js // Сборник докладов XIII Международной конференции, РОССИЙСКИЕ РЕГИОНЫ В ФОКУСЕ ПЕРЕМЕН. Сборник докладов XIII Международной конференции. Екатеринбург, 2019, Т. 1. С. 550-555.

References:

1. Sychev, A. V. Theory and practice of developing modern client web applications: textbook / A. V. Sychev. — 3rd ed. — Moscow: Internet University of Information Technologies (INTUIT), IP Ar Media, 2021. — 482 p. — ISBN 978-5-4497-0943-1. — Text: electronic // Digital educational resource IPR SMART: [website]. — URL: https://www.iprbookshop.ru/102067.html
2. Ardeychik S.M., Sherbaf A.I. SPA architecture of a multifunctional web application // VESCI BDPU. GRAY 3. PHYSICS. MATH. INFORMATION. BIALOGY. GEAGRAPHY, 2018, No. 3(97), pp. 85 – 91.
3. Kononov N.A. Study of the reasons for the emergence and formation of a definition of the concept of reactivity used in web development // Collection of articles of the XXIV International Scientific and Practical Conference. Penza, 2021, pp. 59-62.
4. Romanov, E. L. Software engineering: textbook: E. L. Romanov; Novosibirsk State Technical University. – Novosibirsk: Novosibirsk State Technical University, 2017. – 395 pp.: table, diagrams, illus. – (NSTU textbooks). – Access mode: by subscription. – URL: https://biblioclub.ru/index.php?page=book&id=573945
5. Details about reactivity [Electronic resource]. URL: https://ru.vuejs.org/v2/guide/reactivity.html
6. Eric Hanchett, Listuon Benjamin Vue.js in action. - St. Petersburg: St. Petersburg, 2019. - 304 p.: ill. - (Series “Programmer’s Library”). ISBN 978-5-4461-1098-8.: [Electronic resource]. URL: http://web.spt42.ru/files/vue/Vue_js_v_deystvii_-_E_Khenchett_2019.pdf
7. Safronov, A. I. Design of a standard information management system using web programming technology based on the Vue.js framework: educational manual / A. I. Safronov, A. I. Kotova. — Moscow: RUT (MIIT), 2019. — 97 p. — Text: electronic // Lan: electronic library system. — URL: https://e.lanbook.com/book/175692
8. Kharitonov Yu.E., Pavlov M.V. Basic concepts and capabilities of the Vue.js framework // Collection of reports of the XIII International Conference, RUSSIAN REGIONS IN THE FOCUS OF CHANGE. Collection of reports of the XIII International Conference. Ekaterinburg, 2019, T. 1. P. 550-555.