Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

Нижегородский государственный технический

университет им. Р.Е.Алексеева

Институт радиоэлектроники и информационных технологий

Кафедра прикладной математики и информатики

Отчёт № 10

«Реализация принципа SPA»

по дисциплине

web-разработка

Выполнил:

студент группы 23-ПМ-1

Блинов А. С.

Проверил:

Курушин Е. А.

Нижний Новгород

2025 год

Оглавление

Цели и задачи 3

Теоретическая справка 4

Реализация 8

Вывод 11

Приложение 12

Цели и задачи

В ходе данной работы были поставлены следующая цель:

Преобразовать все страницы системы в компоненты и реализовать между ними клиентскую маршрутизацию.

Для из реализации нужно выполнить следующие задачи:

1. Оформите в виде компонентов оставшиеся страницы ИС.

2. Добавьте роутинг.

Теоретическая справка

Современная веб-разработка все чаще отдает предпочтение компонентно-ориентированной архитектуре и SPA-подходу, что позволяет создавать высокопроизводительные и интерактивные пользовательские интерфейсы. Ключевыми аспектами такой разработки являются эффективная маршрутизация, модульная структура и продуманные способы взаимодействия между компонентами. В данном разделе рассматриваются фундаментальные концепции, необходимые для построения многостраничных Vue-приложений с четкой организацией кода.

1. В чём заключается принцип SPA?

SPA (Single Page Application) — это современный подход к веб-разработке, при котором все необходимые ресурсы загружаются единожды при первом обращении к приложению. Последующая навигация между разделами происходит динамически, без полной перезагрузки страницы, за счет асинхронной подгрузки данных и замены компонентов. Это обеспечивает мгновенный отклик интерфейса и создает ощущение работы с нативным приложением.

2. Что такое роутинг?

Роутинг в контексте SPA — это система навигации, которая сопоставляет URL-адреса с соответствующими компонентами приложения. Во Vue.js для этих целей используется библиотека Vue Router, предоставляющая:

- Динамическую подгрузку компонентов
- Вложенные маршруты
- Навигационные хуки
- Анимацию переходов

Пример конфигурации:

```
const routes = [
```

```
  { path: '/dashboard', component: Dashboard },
  { path: '/profile', component: UserProfile }
]
```

3. Что такое пайпы?

В экосистеме Vue (ранее известные как фильтры) — это функции для трансформации данных непосредственно в шаблонах. Хотя в Vue 3 они были удалены из ядра, их аналоги можно реализовать через:

- Вычисляемые свойства (computed)
- Глобальные методы
- Специальные компоненты-форматтеры

Пример форматирования даты:

```
// В компоненте
computed: {
  formattedDate() {
    return new Date(this.rawDate).toLocaleString()
  }
}
```

4. Что такое модули?

Модули представляют собой автономные блоки функциональности, которые:

- Инкапсулируют логику предметной области
- Минимизируют межфайловые зависимости
- Упрощают тестирование

В Vue-экосистеме модулями могут быть:

- Компоненты (один файл = один компонент)
- Vuex-модули для управления состоянием
- Плагины и миксины
- API-сервисы

5. В чём отличие разработки клиентской части ИС с Node.js и без неё?

Различия представлены в таблице 1.

Таблица 1 – Отличие разработки клиентской части ИС с Node.js и без неё

Критерий	**С Node.js**	**Без Node.js**
Сборка	Автоматизированная	Ручная
Зависимости	NPM/Yarn	CDN-подключение
Инструменты	Vue CLI, Vite	Базовые скрипты
Оптимизация	Tree-shaking, chunking	Нет
Тестирование	Jest, Mocha	Ограниченное

6. Как осуществляется связь между компонентами приложения?

В Vue предусмотрена многоуровневая система взаимодействия:

1. Иерархическая связь
   - props (нисходящий поток)
   - $emit (восходящий поток)
2. Глобальное состояние
   - Vuex/Pinia для сложных сценариев
3. Контекстная инъекция
   - provide/inject для глубоких цепочек
4. Шина событий
   - mitt или аналоги для слабосвязанных компонентов

Пример использования provide/inject:

```
// Родительский компонент
provide('userData', user)

// Дочерний компонент
const user = inject('userData')
```

Реализация

Была реализована система маршрутизации с использованием библиотеки vue-router. Ранее была создана главная страница в виде Vue-компонента. По аналогии с ней были оформлены остальные страницы — «Courses», «Info» каждая из которых представлена отдельным .vue-файлом. Далее был настроен роутинг, позволяющий переключаться между этими страницами без перезагрузки.

1. Создание компонентов страниц

Компоненты для каждой страницы были оформлены в папке src/views:

- MainView.vue — главная страница
- CourseView.vue — страница с курсами
- InfoView.vue — страница инфо

Вид со всеми страницами представлен на рисунке 1.

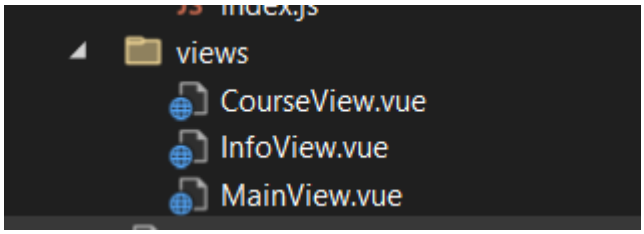

Рисунок 1 – Папка со всеми страницами

Каждый компонент содержит стандартную структуру Vue: блок <template> с HTML-разметкой, <script> с логикой и <style scoped> для индивидуальных стилей.

2. Установка и настройка маршрутизатора

Для организации маршрутов была использована библиотека vue-router:

В каталоге src/router/ был создан файл index.js, где были описаны маршруты для всех страниц.

3. Подключение роутера к приложению

В главном файле main.js была произведена инициализация Vue-приложения с использованием созданного маршрутизатора.

4. Обновление App.vue

Все обновлённые блоки кода представлены в приложении.

5. Проверка работоспособности

Маршруты успешно отрабатывали: при переходе по ссылкам содержимое страницы обновлялось без перезагрузки, благодаря механизму SPA. Это обеспечило плавную и логичную навигацию между разделами интерфейса информационной системы.

Таким образом, был реализован полноценный роутинг во Vue-приложении, что позволило пользователю интуитивно перемещаться между различными страницами, не покидая интерфейс.

На рисунке 2 представлен работающий сайт.

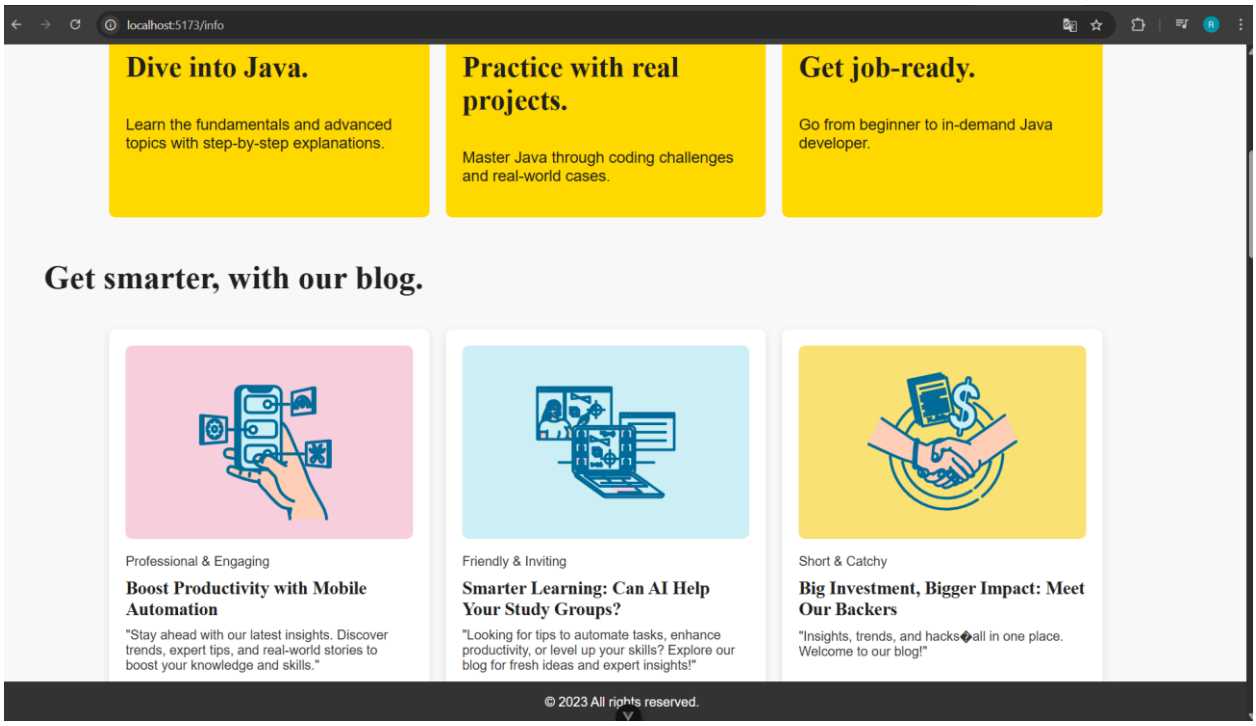

Рисунок 2 – Итоговый сайт

Вывод

В ходе работы выполнено преобразование всех страниц системы в компоненты Vue.js с реализацией клиентской маршрутизации через Vue Router. Это позволило создать структурированное SPA-приложение с плавными переходами между разделами, обеспечив основу для дальнейшего развития проекта.

Приложение

```
import { createRouter, createWebHistory } from 'vue-router'
import MainView from '@/views/MainView.vue'
import CourseView from '@/views/CourseView.vue'
import InfoView from '@/views/InfoView.vue'

const router = createRouter({
    history: createWebHistory(import.meta.env.BASE_URL),
    routes: [
        {
            path: '/',
            name: 'home',
            component: MainView
        },
        {
            path: '/courses',
            name: 'courses',
            component: CourseView
        },
        {
            path: '/info',
            name: 'info',
            component: InfoView
        }
    ]
})

export default router
```