САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Фронт-энд разработка

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ №3

Выполнила: Лапшина Екатерина

Группа: К3440

Проверил: Добряков Д. И.

Санкт-Петербург

2025 г.

Задание

Мигрировать ранее разработанное приложение (в рамках ЛР1 и ЛР2) на фреймворк Vue.js.

Требования:

- Подключение роутера (Vue Router)
- Работа с внешним API посредством Axios
- Компонентный подход к разработке
- Использование Composables для выделения логики

Проект: RentApp — веб-сервис для поиска и бронирования жилья (квартиры, дома, студии).

Ход работы

При миграции проекта на Vue.js структура приложения была организована следующим образом. Все страницы разбиты на представления (views), а переиспользуемые элементы вынесены в компоненты (components). Логика работы с данными и API вынесена в composables.

1. Структура проекта

Проект организован по следующей структуре директорий:

```
src/
├── assets/ # Стили и ресурсы
├── components/ # Переиспользуемые компоненты
│      ├── AppHeader.vue
│      ├── AppFooter.vue
│      └── PropertyCard.vue
├── composables/  # Логика (Composition API)
│      ├── useApi.js
│      ├── useAuth.js
│      ├── useBookings.js
│      └── useProperties.js
├── router/ # Маршрутизация
│      └── index.js
├── views/  # Страницы
│      ├── Login.vue
│      ├── Messages.vue
│      ├── Object.vue
│      ├── Profile.vue
│      ├── Register.vue
│      └── Search.vue
├── App.vue # Корневой компонент
└── main.js # Точка входа
```

2. Реализация роутинга

Настроена маршрутизация для основных страниц приложения: поиск, авторизация, регистрация, страница объекта, профиль и сообщения. Роутер создан с использованием history mode для чистых URL-адресов.

```
import { createRouter, createWebHistory } from 'vue-router'
import Search from '../views/Search.vue'
import Login from '../views/Login.vue'
import Register from '../views/Register.vue'
import ObjectDetails from '../views/Object.vue'
import Profile from '../views/Profile.vue'
import Messages from '../views/Messages.vue'

const router = createRouter({
      history: createWebHistory(import.meta.env.BASE_URL),
      routes: [
            { path: '/', name: 'home', component: Search },
            { path: '/login', name: 'login', component: Login },
            { path: '/register', name: 'register', component: Register },
            { path: '/property/:id', name: 'property', component: ObjectDetails,
      props: true },
            { path: '/profile', name: 'profile', component: Profile },
            { path: '/messages', name: 'messages', component: Messages }
      ]
})

export default router
```

3. Работа с API и Composables

3.1. **Конфигурация Axios.** Создан базовый экземпляр Axios с настройкой базового URL для всех запросов к API:

```
import axios from 'axios'

 const api = axios.create({
   baseURL:
   'http://localhost:3000'
 })

 export function useApi() {
   return { api }
 }
```

3.2. **Логика работы со свойствами.** Реализована загрузка списка объектов недвижимости с возможностью фильтрации по городу, типу объекта и ценовому диапазону

```
import { ref } from 'vue'
import { useApi } from './useApi'

export function useProperties() {
      const { api } = useApi()
      const properties = ref([])
      const loading = ref(false)

      async function fetchProperties(filters = {}) {
            loading.value = true
            try {
                  let query = '?'
                  if (filters.city) query += `city=${filters.city}&`
                  if (filters.type) query += `type=${filters.type}&`
                  if (filters.minPrice) query += `minPrice=${filters.minPrice}&`
                  if (filters.maxPrice) query += `maxPrice=${filters.maxPrice}&`

                  const response = await api.get(`/properties${query}`)
                  properties.value = response.data
            } finally {
                  loading.value = false
            }
      }

      return { properties, loading, fetchProperties }
 }
```

3.3. **Логика авторизации.** Реализована система аутентификации с глобальным состоянием текущего пользователя:

```
import { ref, computed } from 'vue'
import { useApi } from './useApi'


const currentUser = ref(null)
const { api } = useApi()

export function useAuth() {
    const isAuthenticated = computed(() => !!currentUser.value)

    async function login(email, password) {
          const response = await api.get(
                `/users?email=${email}&password=${password}`
          )
          if (response.data.length > 0) {
                currentUser.value = response.data[0] return true
          }
          return false
    }

    function logout() {
          currentUser.value = null
    }
```

```
    return { currentUser, isAuthenticated, login, logout }
  }
```

4. Компоненты и страницы

4.1. **Карточка объекта.** Компонент PropertyCard отображает краткую информацию об объекте недвижимости с изображением, названием и ценой:

```
<script setup>
  defineProps({
      property: { type: Object, required: true }
  })
</script>

<template>
      <div class="card h-100">
            <img :src="property.image" class="card-img-top" :alt="property.title">
            <div class="card-body">
                  <h5 class="card-title">{{ property.title }}</h5>
                  <p class="card-text">{{ property.price }} ₽/night</p>
                  <RouterLink :to="{ name: 'property', params: {id:property.id}}"
                        class="btn btn-primary">
                        View Details
                  </RouterLink>
            </div>
      </div>
</template>
```

4.2. **Страница поиска.** Главная страница приложения интегрирует composable useProperties и отображает форму фильтров и сетку карточек объектов:

```
<script setup>
import { ref, onMounted } from 'vue'
import { useProperties } from '../composables/useProperties'
import PropertyCard from '../components/PropertyCard.vue'

const { properties, loading, fetchProperties } = useProperties()
const filters = ref({
  city: '',
  type: '',
  minPrice: '',
  maxPrice: ''
})

onMounted(() => {
  fetchProperties()
})

function applyFilters() {
  fetchProperties(filters.value)
}
</script>

<template>
  <div class="row">
    <div class="col-md-3">
      <form @submit.prevent="applyFilters">
        <!-- Поля для фильтров -->
        <input v-model="filters.city" placeholder="City">
        <select v-model="filters.type">
          <option value="">All Types</option>
          <!-- ... -->
        </select>
        <button type="submit" class="btn
          btn-primary"> Apply
        </button>
      </form>
    </div>

    <div class="col-md-9">
      <div v-if="loading">Loading...</div>
      <div class="row" v-else>
        <div class="col-4"
             v-for="prop in properties"
             :key="prop.id">
          <PropertyCard :property="prop" />
        </div>
      </div>
    </div>
  </div>
</template>
```

Вывод

В ходе выполнения лабораторной работы была осуществлена миграция проекта RentApp на фреймворк Vue.js. Проект логически структурирован с выделением переиспользуемых компонентов и бизнес-логики через Composition API. Реализовано взаимодействие с мок-сервером через Axios, настроена маршрутизация с использованием Vue Router. Применение composables позволило эффективно разделить логику приложения и обеспечить её переиспользование в различных компонентах. Полученная архитектура обеспечивает масштабируемость и поддерживаемость кода.