САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Фронт-энд разработка

Отчет

Лабораторная работа №3

Выполнил:

Цой Степан

Группа
К3440

Проверил:
Добряков Д. И.

Санкт-Петербург

2025 г.

Задачи и цели

Цель работы — освоить разработку одностраничных приложений (SPA) с использованием фреймворка Vue 3 с Composition API, vue-router для маршрутизации и паттерна composables для переиспользования логики.

- Создать Vue 3 проект с помощью Vite и настроить его структуру
- Реализовать компонентный подход: NavBar, RecipeCard и View-компоненты
- Освоить реактивность Vue: ref(), computed(), onMounted()
- Настроить клиентскую маршрутизацию через vue-router с навигационным гардом
- Применить composables useApi и useAuth для разделения ответственности
- Использовать все основные директивы: v-if, v-for, v-model, :bind, @events

Ход работы

Структура проекта

```
vue-app/src/
├── main.js             # createApp + use(router) + mount
├── App.vue             # Корневой компонент: <NavBar> + <RouterView>
├── router/index.js     # Маршруты + beforeEach гард
├── composables/
│   ├── useApi.js       # HTTP-запросы, loading/error состояния
│   └── useAuth.js      # Авторизация, синглтон user ref
├── components/
│   ├── NavBar.vue      # Шапка с навигацией и переключателем темы
│   └── RecipeCard.vue   # Переиспользуемая карточка рецепта
└── views/
    ├── HomeView.vue    # Главная: поиск, фильтры, список рецептов
    ├── RecipeView.vue   # Страница рецепта: детали, комментарии
    ├── LoginView.vue   # Форма входа
    ├── RegisterView.vue # Форма регистрации
    └── ProfileView.vue  # Профиль с сохранёнными рецептами
```

Composable useApi.js

```
import { ref } from 'vue'
import axios from 'axios'

const API = 'http://localhost:3000'

export function useApi() {
  const loading = ref(false)
  const error = ref(null)

  async function request(method, url, data = null) {
   loading.value = true; error.value = null
   try {
   const res = await axios({ method, url: API + url, data })
   return res.data
   } catch (e) {
   error.value = e.message; return null
   } finally {
   loading.value = false  // Выполняется ВСЕГДА
   }
  }

  return {
   loading, error,
   getRecipes:  (params) => request('GET', '/recipes' + (params||'')),
   getRecipe:   (id)   => request('GET', `/recipes/${id}`),
   addComment:  (body)  => request('POST', '/comments', body),
  }
}
```

Маршрутизация с навигационным гардом

```
import { createRouter, createWebHistory } from 'vue-router'

const routes = [
  { path: '/',          component: HomeView },
  { path: '/recipe/:id',  component: RecipeView },   // :id — параметр
  { path: '/login',     component: LoginView },
  { path: '/profile',   component: ProfileView,
   meta: { requiresAuth: true } },                // Защищённый маршрут
]

const router = createRouter({ history: createWebHistory(), routes })

// Навигационный гард — выполняется перед каждым переходом
router.beforeEach((to, from, next) => {
  if (to.meta.requiresAuth && !localStorage.getItem('user')) {
   next('/login')  // Редирект неавторизованных
  } else {
   next()        // Разрешить переход
  }

})
```

HomeView — директивы шаблона

```
<template>
  <div>
    <!-- v-model: двустороннее связывание с ref -->
    <input v-model="searchQuery" placeholder="Поиск рецептов...">


    <!-- v-if/v-else: условный рендеринг, удаляет из DOM -->
    <div v-if="loading">⏳ Загрузка...</div>
    <div v-else-if="error">❌ {{ error }}</div>
    <div v-else>
    <!-- v-for с :key — список, key для алгоритма сравнения -->
    <RecipeCard v-for="r in filteredRecipes" :key="r.id"
                :recipe="r" />  <!-- :recipe — передача prop -->
    </div>
  </div>
</template>
```

Вывод

В ходе работы реализовано полноценное SPA-приложение на Vue 3. Composition API с ref() и computed() обеспечивает декларативную реактивность — DOM обновляется автоматически при изменении данных без ручного манипулирования. Паттерн composables (useApi, useAuth) разделил ответственность: компоненты содержат только UI-логику, вся работа с данными вынесена в переиспользуемые функции. Навигационный гард beforeEach реализовал защиту маршрутов без дублирования проверки в каждом компоненте. Директивы v-for с :key, v-model, v-if демонстрируют декларативный подход: разработчик описывает что должно отображаться, а не как это сделать.